

Apprentissage par renforcement appliqué

Algorithmes fondamentaux [revision 3.2]

Brahim Chaib-draa

Brahim.Chaib-Draa@ift.ulaval.ca



Université Laval

2020-05-26

- 1** Concept clé en RL
- 2** Paysage algorithmique de l'apprentissage par renforcement
- 3** Modèle connu **et** parfait \implies pas d'apprentissage requis
- 4** Apprentissage par renforcement (sans modèle)
- 5** Pour aller plus loin

Concept clé en RL

1 Concept clé en RL

- Apprentissage *sans modèle* vs apprentissage *basée sur un modèle*
- Apprentissage vs Planification
- Prediction vs Contrôle
- Apprentissage EN-ligne vs HORS-ligne

2 Paysage algorithmique de l'apprentissage par renforcement

3 Modèle connu **et** parfait \implies pas d'apprentissage requis

4 Apprentissage par renforcement (sans modèle)

5 Pour aller plus loin

Concept clé en RL

Apprentissage *sans modèle* vs apprentissage *basée sur un modèle*

Concept HAUT niveau :

- Apprentissage **sans modèle** vs apprentissage **basée sur un modèle**
[*Model-free vs model-based*]
- Apprentissage vs Planification
[*Learning vs Planning*]

Concept BAS niveau :

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

Apprentissage sans modèle vs apprentissage basée sur un modèle

Concept HAUT niveau

Méthode sans modèle [*Model-free*]

- L'agent **ne dépend pas d'un modèle** pour apprendre ;
- Il apprend par essai et erreur ;



Méthode basée sur un modèle [*Model-based*]

- 1 Soit l'agent **apprend un modèle** de A à Z ;
- 2 ou l'agent **améliore un modèle** par expérience ;
- 3 ou l'agent connaît le modèle à priori ;



Concept clé en RL

Apprentissage vs Planification

Apprentissage vs Planification

Concept HAUT niveau

Apprentissage [*Learning*]

Acquérir des habiletés ou des connaissances soit par :

- 1 essai et erreur ;
- 2 imitation ;
- 3 ou sous la supervision d'un professeur *Ex. curriculum learning ;*



Planification [*Planning*]

Déterminer les étapes nécessaires à l'atteinte d'un **but** et mesurer les efforts nécessaires ;

Prérequis : avoir **un modèle de l'environnement**

Ex. un calendrier, une carte, les règles de physique, les règles d'un jeu ...



Concept clé en RL

Prediction vs Contrôle

Prediction vs Contrôle

Concept BAS niveau

Prediction [*Prediction*]

Basée sur mes **connaissances actuelles**, qu'est-ce que je peux **déduire/induire** (à propos du **futur**)¹.

Implique une notion d'incertitude.

Ex. de prédiction : estimer Q^π ou V^π



Contrôle [*Control*]

Choisir une stratégie **optimale** afin d'atteindre un **but**.

Ex. de contrôle : estimer V^* , Q^* ou π^*



1. Contrairement à son utilisation dans le domaine des statistiques où le terme prédiction n'implique pas nécessairement la notion de futur.

Concept clé en RL

Apprentissage EN-ligne vs HORS-ligne

Apprentissage EN-ligne vs HORS-ligne

Concept BAS niveau

Apprentissage EN-ligne [*ON-line learning*]

L'apprentissage se fait **au fur et à mesure** que l'agent exécute des actions dans l'environnement.



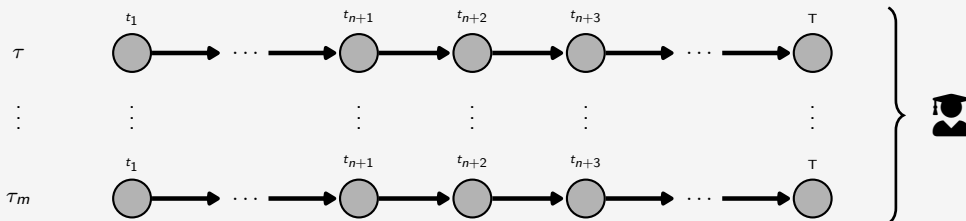
Exemple : *TD-learning*.

Apprentissage EN-ligne vs HORS-ligne

Concept BAS niveau

Apprentissage HORS-ligne [*OFF-line learning*]

La phase d'apprentissage est **retardée après la collecte d'une série de trajectoires τ** .



avec m le nombre de trajectoire collecté [$m = \text{batch size}$]

Ex : *Batch RL*. [▶ wikipédia](#).

Paysage algorithmique de l'apprentissage par renforcement

1 Concept clé en RL

2 Paysage algorithmique de l'apprentissage par renforcement

- Taxonomie du RL/DRL
- Pourquoi autant d'algorithmes ?
- Comparaison des types de méthode en RL

3 Modèle connu **et** parfait \implies pas d'apprentissage requis

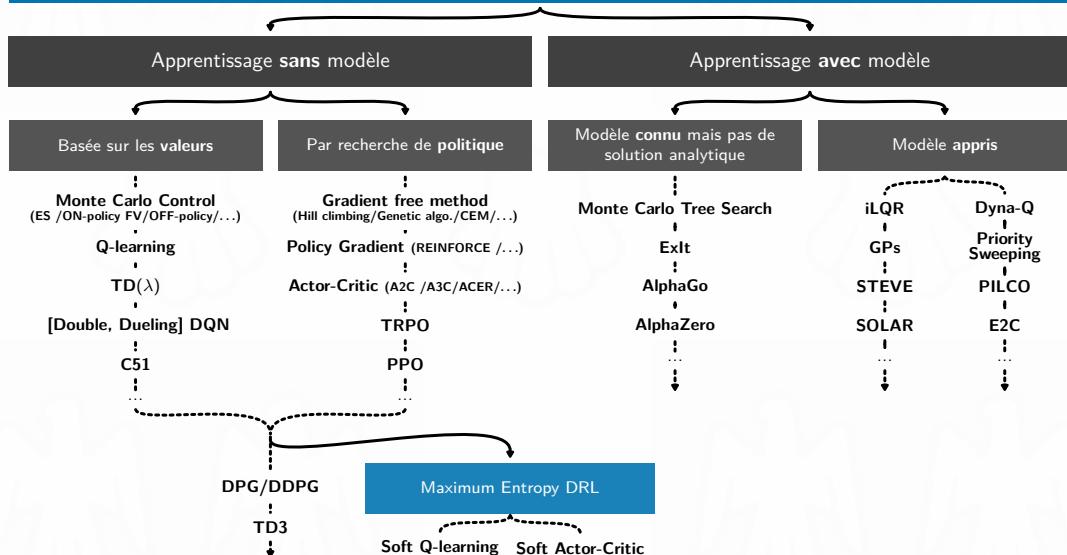
4 Apprentissage par renforcement (sans modèle)

5 Pour aller plus loin

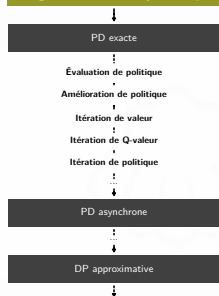
Paysage algorithmique de l'apprentissage par renforcement

Taxonomie du RL/DRL

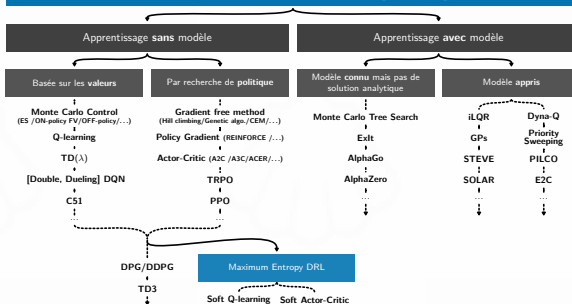
Apprentissage par renforcement [profond]



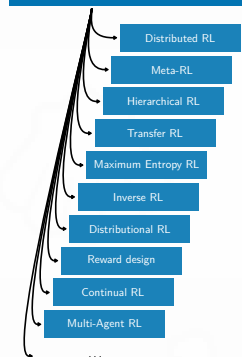
Programmation Dynamique



Apprentissage par renforcement [profond]



RL/DRL avancé



Paysage algorithmique de l'apprentissage par renforcement

Pourquoi autant d'algorithmes ?

Pourquoi autant d'algorithmes ?

Différent compromis Ex : On a peu de puissance de calcul, mais générer de nouvelles trajectoires est très rapides, donc c'est plus efficace d'utiliser un algorithme de type *Monté Carlo* qu'un par *Différence Temporelle* ;

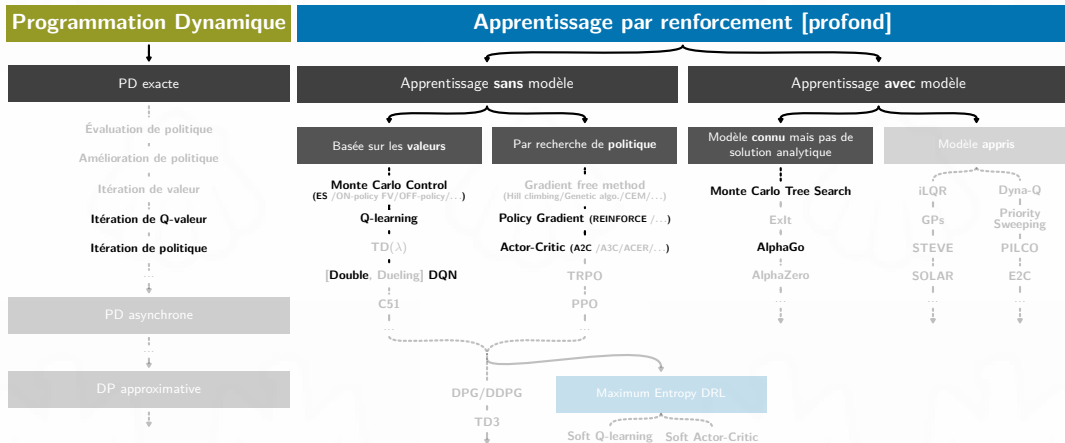
Différentes assomptions Ex : On fait l'assomption qu'on a accès à un modèle parfait, donc on peut utiliser un algorithme de *Programmation Dynamique* ;

Différente contrainte Ex : On a besoin d'un algorithme qui fonctionne sur des espaces d'actions continue, donc on ne peut pas utiliser les algorithmes de la famille *Q-learning* ;

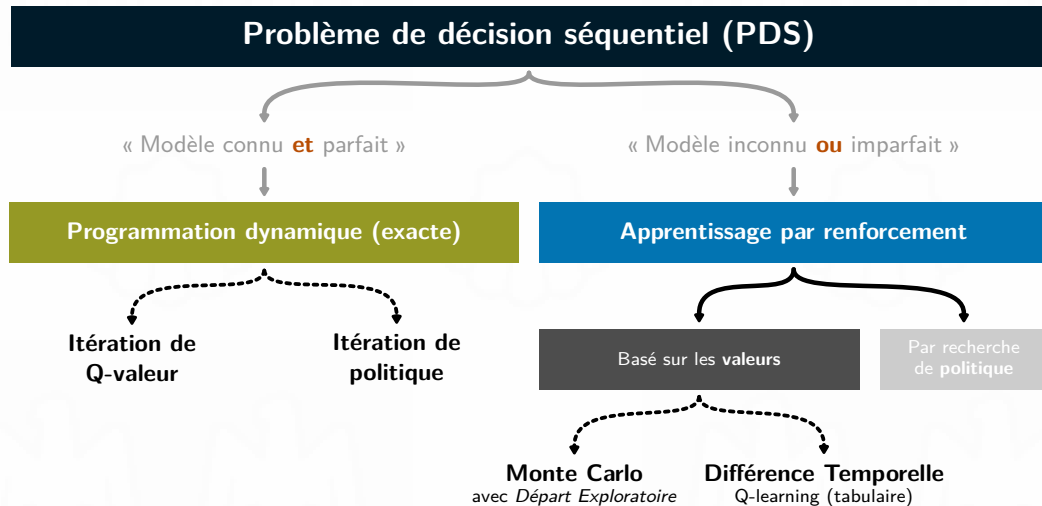
Plus de détail au fur et à mesure

Note : Voir *Reinforcement Learning coach/Selecting an Algorithm* pour un bon exemple illustrant le niveau de complexité que représente le choix d'algorithme en RL. [► RL Coach / selecting an algorithm](#)

Algorithme couvert dans le cours



Algorithme vue dans cette partie



Paysage algorithmique de l'apprentissage par renforcement

Comparaison des types de méthode en RL

Terminologie

« Backup Operation »

L'opération de **mise à jour** d'un *V-valeur* (ou *Q-valeur*) par une **estimation** de son possible successeur¹

$$V\text{-valeur} \leftarrow f_{\text{approximation du successeur possible de } V\text{-valeur}}$$

1. Voir l'excellente explication par Andre G. Barto pour plus de détail dans REINFORCEMENT LEARNING AND DYNAMIC PROGRAMMING, section 2.2 [1]

Programmation dynamique : Estimer la valeur en utilisant l'**espérance mathématique**, le **modèle** de l'environnement et l'**estimations courantes** ;

Ex. *Backup operation* pour Itération de Q-valeur

$$Q_{i+1}^{\pi}(s,a) \leftarrow \mathbb{E}_{s' \sim p} \left[r(s,a,s') + \gamma \max_{a' \in \mathcal{A}(s')} Q^{\pi}(s',a') \right]$$

RL par méthode Monté Carlo : Estimer la valeur en utilisant la **moyenne empirique** sur des **échantillons de trajectoire τ complets** ;

Ex. *Backup operation* pour Monte Carlo ES

$$Q^{\pi}(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \frac{1}{k} \left[G_t(\tau) - Q^{\pi}(s_t, a_t) \right]$$

RL par Différence Temporelle : Estimer la valeur en utilisant des **échantillons de trajectoire τ partielle + l'estimation courantes** du reste de la trajectoire ;

Ex. *Backup operation* pour Q-learning

$$Q^{\pi}(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \alpha \left[r(s_t, a_t, s_{t+1}) + \gamma \max_{a' \in \mathcal{A}(s_{t+1})} Q^{\pi}(s_{t+1}, a') - Q^{\pi}(s_t, a_t) \right]$$

Programmation dynamique : Estimer la valeur en utilisant l'**espérance mathématique**, le **modèle** de l'environnement et l'**estimations courantes** ;

Ex. *Backup operation* pour Itération de Q-valeur

$$Q_{i+1}^{\pi}(s,a) \leftarrow \mathbb{E}_{s' \sim p} \left[\underbrace{r(s,a,s')}_{\text{modèle}} + \gamma \max_{a' \in \mathcal{A}(s')} Q^{\pi}(s',a') \right]$$

RL par méthode Monté Carlo : Estimer la valeur en utilisant la **moyenne empirique** sur des **échantillons de trajectoire τ complets** ;

Ex. *Backup operation* pour Monte Carlo ES

$$Q^{\pi}(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \frac{1}{k} \left[\underbrace{G_t(\tau)}_{\text{échantillon}} - Q^{\pi}(s_t, a_t) \right]$$

RL par Différence Temporelle : Estimer la valeur en utilisant des **échantillons de trajectoire τ partielle** + l'**estimation courantes** du reste de la trajectoire ;

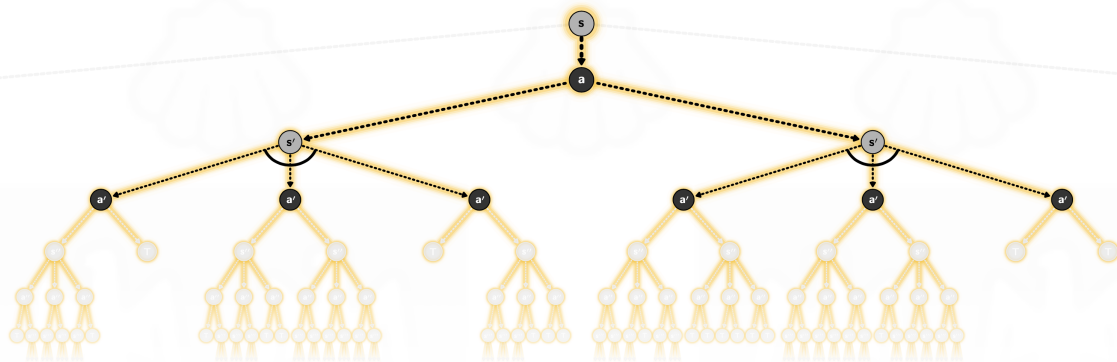
Ex. *Backup operation* pour Q-learning

$$Q^{\pi}(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \alpha \left[\underbrace{r(s_t, a_t, s_{t+1})}_{\text{échantillon}} + \gamma \max_{a' \in \mathcal{A}(s_{t+1})} Q^{\pi}(s_{t+1}, a') - Q^{\pi}(s_t, a_t) \right]$$

Backup Diagram de Programmation Dynamique

Ex.: Algorithme *Itération de Q-valeur*

$$Q_{i+1}^{\pi}(s, a) \leftarrow \mathbb{E}_{s' \sim p} \left[r(s, a, s') + \gamma \max_{a' \in \mathcal{A}(s')} Q^{\pi}(s', a') \right]$$

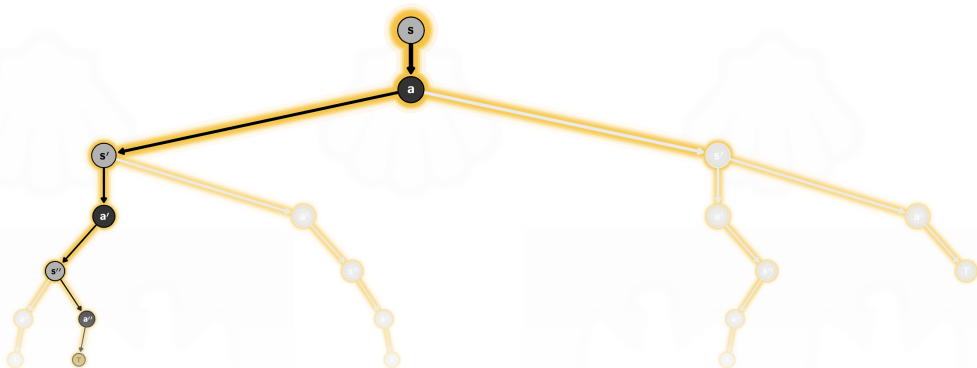


PLACEHOLDER

Backup Diagram de méthode Monté Carlo

Ex.: Algorithme *Monté Carlo ES*

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) + \frac{1}{k} \left[G_t(\tau) - Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \right]$$

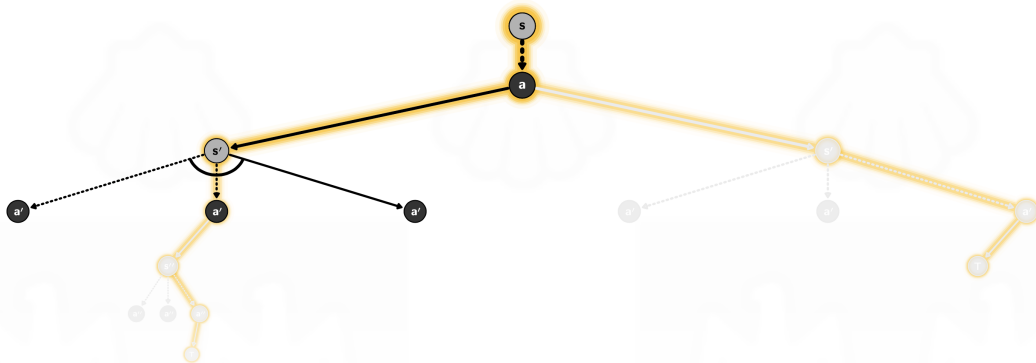


PLACEHOLDER

Backup Diagram de méthode par Différence Temporelle

Ex.: Algorithme Q-Learning

$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha \left[r(s_t, a_t, s_{t+1}) + \gamma \max_{a' \in \mathcal{A}(s_{t+1})} Q^\pi(s_{t+1}, a') - Q^\pi(s_t, a_t) \right]$$



PLACEHOLDER

Modèle connu et parfait \implies pas d'apprentissage requis

1 Concept clé en RL

2 Paysage algorithmique de l'apprentissage par renforcement

3 **Modèle connu et parfait \implies pas d'apprentissage requis**

- Méthode par Programmation Dynamique exacte
- DP : Itération de Q-valeur
- DP : Itération de politique

4 Apprentissage par renforcement (sans modèle)

5 Pour aller plus loin

Modèle connu et parfait \implies pas d'apprentissage requis

Méthode par Programmation Dynamique exacte

Programmation dynamique (exacte) et processus de décision séquentielle

« Suposont qu'après avoir lu un livre sur les cubes Rubik, on sait comment résoudre tout les cubes Rubik ... juste en réfléchissant et sans jamais en avoir pris un dans ses mains. »



Image : Cube Rubik

Caractéristique :

- Résolution par planification \Rightarrow **requiert un modèle** de l'environnement ;
- Type d'algorithme **hors-ligne** et **sans phase d'apprentissage** ;
c.-à-d. que la solution est calculée sans que l'agent n'ait à interagir avec l'environnement

Idées clés :

- 1 **Décomposer** le problème difficile **en sous-problèmes** plus simple ;
- 2 Trouver une **solution optimale** à chaque sous-problèmes ;
- 3 **Résoudre récursivement** en utilisant les solutions des sous-problèmes ;

Programmation dynamique (exacte) et processus de décision séquentielle

- 👍 **Pro :** ■ **Garantie** de trouver une solutions optimale exacte ;
- 👎 **Con :** ■ **Avoir accès au modèle** de l'environnement ;
 - Fonctionne sous l'**assomption** que le modèle est **parfait** (c.a.d. sans zone grise, ni erreur) ;
 - Victime de la **malédiction de la dimensionnalité** :
 - \implies Très gourmand en mémoire et sur le plan computationnel
 - Perd en efficacité pratique plus l'espaces d'état/action devient massif¹ ;

Pourquoi étudier ces méthodes dans un cours sur le RL alors ?

- Toutes les méthodes de RL **aspirent à des résultats similaire** à ceux obtenue par DP, **mais en se libérant des contraintes** du DP² ;
- 👍 Procure une solide **fondation** pour comprendre certains mécanismes fondamentaux de l'apprentissage par renforcement ;

1. Voir « Reinforcement Learning : An introduction », chapitre 4, section 7 « Efficiency of Dynamic Programming » par Sutton & Barto [2]

2. Quite à perde certaines propriété du DP à l'ocasion : garantie de solution optimale, production d'une solution exacte

Programmation dynamique (exacte) et processus de décision séquentielle

- 👍 **Pro :** ■ **Garantie** de trouver une solutions optimale exacte ;
- 👎 **Con :** ■ **Avoir accès au modèle** de l'environnement ;
 - Fonctionne sous l'**assomption** que le modèle est **parfait** (c.a.d. sans zone grise, ni erreur) ;
 - Victime de la **malédiction de la dimensionnalité** :
 - \Rightarrow Très gourmand en mémoire et sur le plan computationnel
 - Perd en efficacité pratique plus l'espaces d'état/action devient massif¹ ;

Pourquoi étudier ces méthodes dans un cours sur le RL alors ?

- Toutes les méthodes de RL **aspirent à des résultats similaire** à ceux obtenue par DP, **mais en se libérant des contraintes** du DP² ;
- 👍 Procure une solide **fondation** pour comprendre certains mécanismes fondamentaux de l'apprentissage par renforcement ;

1. Voir « Reinforcement Learning : An introduction », chapitre 4, section 7 « Efficiency of Dynamic Programming » par Sutton & Barto [2]

2. Quite à perde certaines propriété du DP à l'ocasion : garantie de solution optimale, production d'une solution exacte

Terminologie

« Backup Operation »

L'opération de **mise à jour** d'un *V-valeur* (ou *Q-valeur*) par une **estimation** de son possible successeur¹

$$V\text{-valeur} \leftarrow f_{\text{approximation du successeur possible de } V\text{-valeur}}$$

« Sweep »

Une execution du « Backup Operation » sur chaque *V-valeurs* (ou *Q-valeurs*) d'un espace.

1. Voir l'excellente explication par Andre G. Barto pour plus de détail dans REINFORCEMENT LEARNING AND DYNAMIC PROGRAMMING, section 2.2 [1]

Modèle connu et parfait \implies pas d'apprentissage requis

DP : Itération de Q-valeur

Itération de Q-valeur [*Q-value iteration*]

« *Think about every possible scenario ... obsessively, leave no stone unturned.* »

- Idées clés :**
- 1 **Estimer et stocker** la Q-valeur **de chaque couples** état-action de l'univers ;
 - 2 **Répéter [infiniment]** en améliorant progressivement l'estimation jusqu'à convergence vers une solution exacte ;

$$Q_i^\pi \xrightarrow{i \rightarrow \infty} Q^*$$



Pro :

- Intuitif et rapide à implémenter ;
- Convergence garantie¹ ;



Con :

- Mémoire requise pour la *Q-table*² : $2 \cdot |\mathcal{S} \times \mathcal{A}|$
- Limiter aux problèmes avec espace d'état/action de taille modeste ;
Exemple : espace d'action discrète, facteur de branchement limité, ...

1. Voir « Reinforcement Learning : An introduction », chapitre 4 (introduction, 4.4 et 4.6) par Sutton & Barto [2]

2. Version synchrone de l'algorithme : les *backup operation* sont effectués en parallèle

Itération de Q-valeur [*Q-value iteration*]

« *Think about every possible scenario ... obsessively, leave no stone unturned.* »

- Idées clés :
- 1 **Estimer et stocker** la Q-valeur **de chaque couples** état-action de l'univers ;
 - 2 **Répéter [infiniment]** en améliorant progressivement l'estimation jusqu'à convergence vers une solution exacte ;

$$Q_i^\pi \xrightarrow{i \rightarrow \infty} Q^*$$



Pro :

- Intuitif et rapide à implémenter ;
- Convergence garantie¹ ;



Con :

- Mémoire requise pour la *Q-table*² : $2 \cdot |\mathcal{S} \times \mathcal{A}|$
 - Limiter aux problèmes avec espace d'état/action de taille modeste ;
- Exemple : espace d'action discrète, facteur de branchement limité, ...

1. Voir « Reinforcement Learning : An introduction », chapitre 4 (introduction, 4.4 et 4.6) par Sutton & Barto [2]

2. Version synchrone de l'algorithme : les *backup operation* sont effectués en parallèle

↔ Algorithme | Itération de Q-valeur

Répéter avec $i = 1, 2, 3, \dots$ jusqu'à convergence vers Q^* :

$$Q_{i+1}^\pi(s, a) \leftarrow \sum_{s' \in \mathcal{S}_{t+1}} p(s'|s, a) \left[r(s, a, s') + \gamma \max_{a' \in \mathcal{A}(s')} Q_i^\pi(s', a') \right] \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$$

> En pratique | Itération de Q-valeur

On **arrête prématurément** l'algorithme au lieu d'itérer à l'infinie en implémentant un moyen d'évaluer l'estimation et en spécifiant la marge d'erreur ε visé.

Conséquence :

- ▶ L'algorithme **converge approximativement** : $Q^\pi \xrightarrow{\approx} Q^*$
- ▶ et **retourne** $\pi \approx \pi^*$ au lieu de la valeur exacte

➤ Pseudocode | Itération de Q-valeur¹ (Version synchrone)

Init: **foreach** $s \in S, a \in \mathcal{A}(s)$ **do** $Q\text{-table}[s, a] \leftarrow 0$

Init: $\text{threshold hyperparam}(\epsilon) > 0$

« Critère d'arrêt »

repeat

« sweep i »

$\Delta \leftarrow 0$

$Q\text{-table}_{old} \leftarrow Q\text{-table}$

foreach $s \in S$ **and** $a \in \mathcal{A}(s)$ **do**

$Q\text{-table}[s, a] \leftarrow \sum_{s' \in S} p(s'|s, a) \left[r(s, a, s') + \gamma \max_{a' \in \mathcal{A}(s')} Q\text{-table}_{old}[s', a'] \right]$

« Backup operation »

$\Delta \leftarrow \max \left(\Delta, \left| Q\text{-table}_{old}[s, a] - Q\text{-table}[s, a] \right| \right)$

until $\Delta < \epsilon$

« $Q^\pi \xrightarrow{\approx} Q^*$ »

Output: $\pi^*(s) \approx \arg \max_{a \in \mathcal{A}(s)} Q\text{-table}[s, a]$

« Produit une politique déterministe »

1. Pseudocode inspiré de « Reinforcement Learning : An introduction », chapitre 4.4, page 83 « Value Iteration » par Sutton & Barto [2]

Modèle connu et parfait \implies pas d'apprentissage requis



DP : Itération de politique

Itération de politique [Policy iteration]

« Think freely, see where it leads you . . . explore until you've seen it all. »

- Idées clés :
- 1 À chaque « sweep », **considérer une seule action par état** (fixer la politique) ;
 - 2 **Estimer et stocker** la V-valeur **de chaque état** de l'univers ;
 - 3 **Choisir une meilleur** politique en évaluant le **futur immédiat** (*one-step look-ahead*) ;
 - 4 **Répéter [infiniment]** en améliorant progressivement l'estimation jusqu'à convergence vers une solution exacte ;

$$V_k^\pi \xrightarrow{k \rightarrow \infty} V_k^\star \quad \text{et} \quad \pi_k \xrightarrow{k \rightarrow \infty} \pi^\star$$

-  **Pro :**
- Intuitif et rapide à implémenter ;
 - Convergence garantie¹ et plus rapide² que *Itération de Q-valeur* ;
-  **Con :**
- Mémoire requise pour la *V-table* et la *π -table*³ : $3 \cdot |S|$
 - Limiter aux problèmes avec espace d'état/action de taille modeste ;

1. Voir « Reinforcement Learning : An introduction », section 4.0, 4.3 et 4.6 par Sutton & Barto [2]

2. Sous certaines condition. Voir preuve de convergence



3. Version synchrone de l'algorithme : les *backup operation* sont effectué en parallèle

Itération de politique [Policy iteration]

« Think freely, see where it leads you . . . explore until you've seen it all. »

- Idées clés :
- 1 À chaque « sweep », **considérer une seule action par état** (fixer la politique) ;
 - 2 **Estimer et stocker** la V-valeur **de chaque état** de l'univers ;
 - 3 **Choisir une meilleur** politique en évaluant le **futur immédiat** (*one-step look-ahead*) ;
 - 4 **Répéter [infiniment]** en améliorant progressivement l'estimation jusqu'à convergence vers une solution exacte ;

$$V_k^\pi \xrightarrow{k \rightarrow \infty} V_k^\star \quad \text{et} \quad \pi_k \xrightarrow{k \rightarrow \infty} \pi^\star$$

-  **Pro :**
- Intuitif et rapide à implémenter ;
 - Convergence garantie¹ et plus rapide² que *Itération de Q-valeur* ;
-  **Con :**
- Mémoire requise pour la *V-table* et la π -table³ : $3 \cdot |\mathcal{S}|$
 - Limiter aux problèmes avec espace d'état/action de taille modeste ;

1. Voir « Reinforcement Learning : An introduction », section 4.0, 4.3 et 4.6 par Sutton & Barto [2]

2. Sous certaines condition. Voir preuve de convergence

3. Version synchrone de l'algorithme : les *backup operation* sont effectué en parallèle

↕ Algorithme | Itération de politique

$\forall s \in \mathcal{S}$, fixer arbitrairement la **politique courante** π_k tel que $\pi_0(s) = a \in \mathcal{A}(s)$

Répéter (étape 1 et 2) avec $k = 1, 2, 3, \dots$ jusqu'à convergence vers V^* et π^* :

1 Évaluer la politique courante

Répéter avec $i = 1, 2, 3, \dots$ jusqu'à convergence vers V^π :

$$V_{i+1}^\pi(s) \leftarrow \sum_{s' \in \mathcal{S}_{t+1}} p(s'|s, \pi_k(s)) [r(s, \pi_k(s), s') + \gamma V_i^\pi(s')] \quad \forall s \in \mathcal{S}$$

2 Améliorer la politique courante

$$\pi_{k+1}(s) \leftarrow \arg \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}_{t+1}} p(s'|s, a) [r(s, a, s') + \gamma V^\pi(s')] \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$$

> En pratique | Itération de politique

On **arrête prématurément** l'algorithme au lieu d'itérer à l'infinie comme dans *Itération de Q-valeur*.

Conséquence :

- L'algorithme **converge approximativement** : $V^\pi \xrightarrow{\approx} V^*$ et $\pi \xrightarrow{\approx} \pi^*$

➤ Pseudocode | Itération de politique¹ (Version synchrone)

Init: foreach $s \in S$ do $V\text{-table}[s] \leftarrow 0$ and $\pi\text{-table}[s] \leftarrow a \in \mathcal{A}(s)$ arbitrarily

Init: threshold hyperparam(ϵ) > 0

« Critère d'arrêt »

repeat

```

1  %....Étape d'évaluation de politique.....
    repeat
         $\Delta \leftarrow 0$ 
         $V\text{-table}_{old} \leftarrow V\text{-table}$ 
        foreach  $s \in S$  do
             $V\text{-table}[s] \leftarrow \sum_{s' \in S} p(s'|s, \pi\text{-table}[s]) [r(s, a, s') + \gamma V\text{-table}_{old}[s']]$ 
             $\Delta \leftarrow \max \left( \Delta, \left| V\text{-table}_{old}[s] - V\text{-table}[s] \right| \right)$ 
        until  $\Delta < \epsilon$ 
        « sweep i »
        « Backup operation »

2  %....Étape d'amélioration de politique.....
     $\pi\text{-IsStable} \leftarrow \text{true}$ 
    foreach  $s \in S$  do
         $a_{old} \leftarrow \pi\text{-table}[s]$ 
         $\pi\text{-table}[s] \leftarrow \arg \max_{a \in \mathcal{A}} \sum_{s' \in S} p(s'|s, a) [r(s, a, s') + \gamma V\text{-table}[s']]$ 
        if  $a_{old} \neq \pi\text{-table}[s]$  then  $\pi\text{-IsStable} \leftarrow \text{false}$ 
        « Backup operation »

    until  $\pi\text{-IsStable}$ 
    «  $V^\pi \xrightarrow{\approx} V^*$  et  $\pi \xrightarrow{\approx} \pi^*$  »

Output:  $\pi^*(s) \approx \pi\text{-table}[s]$ 
    « Produit une politique déterministe »
  
```

1. Pseudocode inspiré de « Reinforcement Learning : An introduction », chapitre 4.3, page 80 « Policy Iteration » par Sutton & Barto [2]

Comparaison

>_ Pseudocode | Itération de Q-valeur

Init: **foreach** $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$ **do** $Q\text{-table}[s, a] \leftarrow 0$

Init: threshold hyperparam(ϵ) > 0

repeat

$\Delta \leftarrow 0$

$Q\text{-table}_{old} \leftarrow Q\text{-table}$

foreach $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$ **do**

$Q\text{-table}[s, a] \leftarrow \sum_{s' \in \mathcal{S}} p(s'|s, a) \left[r(s, a, s') + \gamma \max_{a' \in \mathcal{A}(s')} Q\text{-table}_{old}[s', a'] \right]$

$\Delta \leftarrow \max \left(\Delta, \left| Q\text{-table}_{old}[s, a] - Q\text{-table}[s, a] \right| \right)$

until $\Delta < \epsilon$

Output: $\pi^*(s) \approx \arg \max_{a \in \mathcal{A}(s)} Q\text{-table}[s, a]$

>_ Pseudocode | Itération de politique

Init: **foreach** $s \in \mathcal{S}$ **do** $V\text{-table}[s] \leftarrow 0$ and $\pi\text{-table}[s] \leftarrow a \in \mathcal{A}(s)$ arbitrarily

Init: threshold hyperparam(ϵ) > 0

repeat

1 $\% \dots$ Étape d'évaluation de politique \dots

repeat

$\Delta \leftarrow 0$

$V\text{-table}_{old} \leftarrow V\text{-table}$

foreach $s \in \mathcal{S}$ **do**

$V\text{-table}[s] \leftarrow \sum_{s' \in \mathcal{S}} p(s'|s, \pi\text{-table}[s]) \left[r(s, a, s') + \gamma V\text{-table}_{old}[s'] \right]$

$\Delta \leftarrow \max \left(\Delta, \left| V\text{-table}_{old}[s] - V\text{-table}[s] \right| \right)$

until $\Delta < \epsilon$

2 $\% \dots$ Étape d'amélioration de politique \dots

$\pi\text{-IsStable} \leftarrow \text{true}$

foreach $s \in \mathcal{S}$ **do**

$a_{old} \leftarrow \pi\text{-table}[s]$

$\pi\text{-table}[s] \leftarrow \arg \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} p(s'|s, a) \left[r(s, a, s') + \gamma V\text{-table}[s'] \right]$

if $a_{old} \neq \pi\text{-table}[s]$ **then** $\pi\text{-IsStable} \leftarrow \text{false}$

until $\pi\text{-IsStable}$

Output: $\pi^*(s) \approx \pi\text{-table}[s]$

Comparaison

> Pseudocode | Itération de Q-valeur

```

Init: foreach  $s \in \mathcal{S}$ ,  $a \in \mathcal{A}(s)$  do  $Q\text{-table}[s, a] \leftarrow 0$ 
Init: threshold hyperparam( $\epsilon$ )  $> 0$ 

repeat
   $\Delta \leftarrow 0$ 
   $Q\text{-table}_{old} \leftarrow Q\text{-table}$ 
  foreach  $s \in \mathcal{S}$  and  $a \in \mathcal{A}(s)$  do
     $Q\text{-table}[s, a] \leftarrow \sum_{s' \in \mathcal{S}} p(s'|s, a) \left[ r(s, a, s') + \gamma \max_{a' \in \mathcal{A}(s')} Q\text{-table}_{old}[s', a'] \right]$ 
     $\Delta \leftarrow \max(\Delta, |Q\text{-table}_{old}[s, a] - Q\text{-table}[s, a]|)$ 
until  $\Delta < \epsilon$ 

Output:  $\pi^*(s) \approx \arg \max_{a \in \mathcal{A}(s)} Q\text{-table}[s, a]$ 

```

> Pseudocode | Itération de politique

```

Init: foreach  $s \in \mathcal{S}$  do  $V\text{-table}[s] \leftarrow 0$  and  $\pi\text{-table}[s] \leftarrow a \in \mathcal{A}(s)$  arbitrarily
Init: threshold hyperparam( $\epsilon$ )  $> 0$ 

repeat
  % ... Étape d'évaluation de politique ...
  1 repeat
     $\Delta \leftarrow 0$ 
     $V\text{-table}_{old} \leftarrow V\text{-table}$ 
    foreach  $s \in \mathcal{S}$  do
       $V\text{-table}[s] \leftarrow \sum_{s' \in \mathcal{S}} p(s'|s, \pi\text{-table}[s]) \left[ r(s, a, s') + \gamma V\text{-table}_{old}[s'] \right]$ 
       $\Delta \leftarrow \max(\Delta, |V\text{-table}_{old}[s] - V\text{-table}[s]|)$ 
    until  $\Delta < \epsilon$ 

  2 % ... Étape d'amélioration de politique ...
   $\pi\text{-IsStable} \leftarrow true$ 
  foreach  $s \in \mathcal{S}$  do
     $a_{old} \leftarrow \pi\text{-table}[s]$ 
     $\pi\text{-table}[s] \leftarrow \arg \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} p(s'|s, a) \left[ r(s, a, s') + \gamma V\text{-table}[s'] \right]$ 
    if  $a_{old} \neq \pi\text{-table}[s]$  then  $\pi\text{-IsStable} \leftarrow false$ 
  until  $\pi\text{-IsStable}$ 

Output:  $\pi^*(s) \approx \pi\text{-table}[s]$ 

```

Itération de valeur/politique

Exemple interactif

Évaluation de politique dans l'environnement GridWorld

► Exemple interactif



Image : REINFORCEjs, Stanford

Apprentissage par renforcement (sans modèle)

1 Concept clé en RL

2 Paysage algorithmique de l'apprentissage par renforcement

3 Modèle connu **et** parfait \Rightarrow pas d'apprentissage requis

4 **Apprentissage par renforcement (sans modèle)**

- RL par méthode Monte-Carlo [MC]
- Algorithme Monté Carlo avec départ exploratoire
- RL par Différence temporelle [TD]

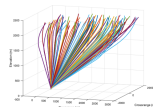
5 Pour aller plus loin

Apprentissage par renforcement (sans modèle)

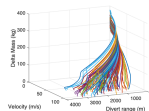
« Supposons qu'on souhaite concevoir une **fusée autonome** avec pour objectif d'atterrir sur la planète Gliese 581c¹ puis re-décoller, **sur quelle base** peut-on développer les systèmes de **navigation, de guidage et de contrôle** ?

D'un point de vue appliqué, est-ce que la **garantie d'optimalité** et la **propriété d'exactitude** sont des **condition nécessaire** ? »

Complément : Voir [Deep Reinforcement Learning for Six Degree-of-Freedom Planetary Powered Descent and Landing](#), partie I. Introduction, pour une comparaison des méthodes de RL par rapport aux méthode traditionnel par Contrôle Optimal dans le domaine de l'aérospatial. (Plus spécifiquement, dans le cadre du problème d'atterrissage autonome à une coordonné précise sur Mars)



6-DOF Trajectories



6-DOF Fuel-Mass Along the Trajectories

1. « Gliese 581c ... is classified as a super-Earth ... there are no measurements of its radius. Furthermore, the radial velocity method used to detect it only puts a lower limit on the planet's mass, which means theoretical models of planetary radius and structure can only be of limited use ... », [wikipedia](#)

Apprentissage par renforcement (sans modèle)

⚠ **Problème :** On n'a **pas de modèle** de l'environnement (Il est inconnu ou imparfait)
⇒
On ne peut pas prédire de comportement dans cette environnement ;

Concrètement : Pas de modèle
⇒
 $p(s'|s, a)$ et $r(s, a, s')$ **inconnu**
⇒
Impossible de calculer une solution optimale **exacte** à priori ;

Solution : **Interagir** avec l'environnement (réellement ou en simulateur) afin d'**apprendre** les composantes nécessaire pour trouver une solution optimale :

- $V^\pi(s)$, $Q^\pi(s, a)$, $\pi(a|s)$ pour le RL **sans** model ;
- $p(s'|s, a)$ et $V^\pi(s)$, $Q^\pi(s, a)$, $\pi(a|s)$ pour le RL **avec** model (appris) ;

Apprentissage par renforcement (sans modèle)

RL par méthode Monte-Carlo [MC]

Méthodes Monté Carlo et processus de décision séquentielle

« *Try many many time, learn from mistakes ... once in a while.* »

- Caractéristiques :**
- **Intéragit** avec l'environnement pour recueillir de l'information ;
 - Collecte des **échantillons** de trajectoire τ **complètes** ;
 - Phase d'apprentissage **hors-ligne** ;
 - Apprentissage **sans-modèle** ;
 - Pas de « Bootstrap » contrairement aux méthodes de *DP* ;
 - N'exploite pas la *propriété de Markov* ;

Idées clés : Utiliser la **moyenne empirique** de plusieurs **trajectoires τ complètes** pour approximer l'espérance mathématique des *équations de Bellman* ;

Pourquoi ça marche ?

Méthodes Monté Carlo et processus de décision séquentielle

« *Try many many time, learn from mistakes ... once in a while.* »

- Caractéristiques :**
- **Intéragit** avec l'environnement pour recueillir de l'information ;
 - Collecte des **échantillons** de trajectoire τ **complètes** ;
 - Phase d'apprentissage **hors-ligne** ;
 - Apprentissage **sans-modèle** ;
 - Pas de « Bootstrap » contrairement aux méthodes de *DP* ;
 - N'exploite pas la *propriété de Markov* ;

Idées clés : Utiliser la **moyenne empirique** de plusieurs **trajectoires τ complètes** pour approximer l'espérance mathématique des *équations de Bellman* ;

Pourquoi ça marche ?

La loi des grands nombres

Estimé de Monté Carlo (formellement)

Soit

- \mathcal{X} un espace d'évènements possible,
- x une réalisation de la variable aléatoire X ,
- f_X la distribution probabilité **continue** suivie par la v.a. X ,
- $x^{(1)}, x^{(2)}, \dots, x^{(N)}$ des échantillons tirés de f_X
- $g(\cdot)$ une fonction mesurable,

alors pour N suffisamment grand

$$\mathbb{E}[g(X)] = \int_{x \in \mathcal{X}} g(x) f_X(x) dx \approx \frac{1}{N} \sum_{n=1}^N g(x^{(n)}) = \hat{g}_N$$

avec \hat{g}_N la moyenne empirique de l'espérance mathématique $\mathbb{E}[g(X)]$

Estimé de Monté Carlo (formellement)

Soit

- \mathcal{X} un espace d'évènements possible,
- x une réalisation de la variable aléatoire X ,
- \mathbb{P}_X la distribution probabilité **discrète** suivie par la v.a. X ,
- $x^{(1)}, x^{(2)}, \dots, x^{(N)}$ des échantillons tirés de \mathbb{P}_X
- $g(\cdot)$ une fonction mesurable,

alors pour N suffisamment grand

$$\mathbb{E}[g(X)] = \sum_{x \in \mathcal{X}} g(x) \mathbb{P}(X = x) \approx \frac{1}{N} \sum_{n=1}^N g(x^{(n)}) = \hat{g}_N$$

avec \hat{g}_N la moyenne empirique de l'espérance mathématique $\mathbb{E}[g(X)]$

Estimé de Monté Carlo (considération pratique)

$$\hat{g}_N = \frac{1}{N} \sum_{n=1}^N g(x^{(n)})$$

Dans le contexte du RL :

$$Q_{k+1}^{\pi}(s_t, a_t) \leftarrow \frac{1}{k} \sum_{j=1}^k G_t^{(j)}(\tau)$$

est un « backup operation » qui **est répéter à chaque couple *état-action* rencontrer** dans une trajectoire échantillonné.

Estimé de Monté Carlo (considération pratique)

$$\hat{g}_N = \frac{1}{N} \sum_{n=1}^N g(x^{(n)})$$

Dans le contexte du RL :

$$Q_{k+1}^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow \frac{1}{k} \sum_{j=1}^k G_t^{(j)}(\tau)$$

est un « backup operation » qui **est répéter à chaque couple *état-action* rencontrer** dans une trajectoire échantillonné.

Estimé de Monté Carlo (considération pratique)

$$\hat{g}_N = \frac{1}{N} \sum_{n=1}^N g(x^{(n)})$$

Dans le contexte du RL :

$$Q_{k+1}^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow \frac{1}{k} \sum_{j=1}^k G_t^{(j)}(\tau)$$

est un « backup operation » qui **est répéter à chaque couple *état-action* rencontrer** dans une trajectoire échantillonné.

Problème : Plus le nombre d'échantillon d'épisode k devient grand, plus ce calcul devient coûteux en mémoire et en temps de calcul. **C'est inefficace ⚠**

Estimé de Monté Carlo (considération pratique)

$$\hat{g}_N = \frac{1}{N} \sum_{n=1}^N g(x^{(n)})$$

Dans le contexte du RL :

$$Q_{k+1}^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow \frac{1}{k} \sum_{j=1}^k G_t^{(j)}(\tau)$$

est un « backup operation » qui **est répéter à chaque couple état-action rencontrer** dans une trajectoire échantillonné.

Problème : Plus le nombre d'échantillon d'épisode k devient grand, plus ce calcul devient coûteux en mémoire et en temps de calcul. **C'est inefficace ⚠**

Solution : Implémentation incrémental du calcul de la moyenne

Estimé de Monté Carlo (considération pratique)

Implémentation incrémental du calcul de la moyenne :

$$\begin{aligned} Q_{k+1}^{\pi}(\mathbf{s}_t, \mathbf{a}_t) &= \frac{1}{k} \sum_{j=1}^k G_t^{(j)}(\tau) \\ &= \frac{1}{k} \left(G_t^{(k)}(\tau) + \sum_{j=1}^{k-1} G_t^{(j)}(\tau) \right) \\ &= \frac{1}{k} \left(G_t^{(k)}(\tau) + (k-1) \frac{1}{k-1} \sum_{j=1}^{k-1} G_t^{(j)}(\tau) \right) \\ &= \frac{1}{k} \left(G_t^{(k)}(\tau) + (k-1) Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \right) \\ &= \frac{1}{k} \left(G_t^{(k)}(\tau) + k Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) - Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \right) \\ &= Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) + \frac{1}{k} \left[G_t^{(k)}(\tau) - Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \right] \end{aligned}$$

Estimé de Monté Carlo (considération pratique)

Implémentation incrémental du calcul de la moyenne :

$$\begin{aligned}Q_{k+1}^{\pi}(\mathbf{s}_t, \mathbf{a}_t) &= \frac{1}{k} \sum_{j=1}^k G_t^{(j)}(\tau) \\&= \frac{1}{k} \left(G_t^{(k)}(\tau) + \sum_{j=1}^{k-1} G_t^{(j)}(\tau) \right) \\&= \frac{1}{k} \left(G_t^{(k)}(\tau) + (k-1) \frac{1}{k-1} \sum_{j=1}^{k-1} G_t^{(j)}(\tau) \right) \\&= \frac{1}{k} \left(G_t^{(k)}(\tau) + (k-1) Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \right) \\&= \frac{1}{k} \left(G_t^{(k)}(\tau) + k Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) - Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \right) \\&= Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) + \frac{1}{k} \left[G_t^{(k)}(\tau) - Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \right]\end{aligned}$$

Estimé de Monté Carlo (considération pratique)

« Update rule », forme générale¹

$$\text{Estimate}^{\text{New}} \longleftarrow \text{Estimate}^{\text{Old}} + \text{StepSize} \left[\underbrace{\text{Target} - \text{Estimate}^{\text{Old}}}_{\text{estimate « error »}} \right]$$

> En pratique | Méthode Monté Carlo

Implémentation incrémental du calcul de la moyenne

$$Q_{k+1}^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \longleftarrow Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) + \frac{1}{k} \left[G_t^{(k)}(\tau) - Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \right]$$

1. Voir explication dans « Reinforcement Learning : An introduction », chapitre 2, page 31 « Incremental Implementation » par Sutton & Barto [2]

Méthodes Monté Carlo et processus de décision séquentielle

- 👍 Pro :**
- Pas besoin de modèle ;
 - Phase d'**apprentissage** exécuté par « **batch** » \implies est un avantage lorsque les ressources computationnel sont inférieur à la capacité à produire des nouvelles trajectoires ;
 - **Aucun biais**¹ ;
 - Les estimés de chaque état sont indépendant l'un de l'autre \implies il est possible de restreindre l'échantillonnage à un sous-ensemble d'état d'intérêt ;
- 👎 Con :**
- **Dépend de la capacité à générer beaucoup de trajectoires**
 \implies est un enjeux si la production de trajectoire est longue, couteuse ou risqué ;
 - Fonctionne seulement sur les **environnements de type épisodique** ;
(chaque trajectoire doit terminer)
 - **Beaucoup de variance**² ;

1. car le retour $G_t(\tau) = \sum_{t'=t}^T \gamma^{t'-t} r(s_{t'}, a_{t'}, s_{t'+1})$ est un estimateur non biaisé de $V^\pi(s_t)$

2. car le retour $G_t(\tau)$ dépend d'une grande quantité de phénomène aléatoire (probabilité de sélection d'action, probabilité de transition, probabilité de récompense)

Méthodes Monté Carlo (Enjeux pratique)

Convergence vers Q^* et π^* garantie sous l'**assomption** que :

- 1 le processus d'*évaluation de politique* se fait sur une infinité d'épisode ;
- 2 chaque couple état-action a été visité un nombre infini de fois dans la limite d'une infinité d'épisode ;

➤ En pratique | Méthode de Monté Carlo

Enjeux → contourner ses deux assomptions d'une façon qui permet de...

- 1 relaxer le prérequis d'avoir à générer une infinité de trajectoire ;
- 2 adresser le **dilemme exploration/exploitation** (assurer l'exploration suffisante de l'espace) ;

Apprentissage par renforcement (sans modèle)

Algorithme Monté Carlo avec départ exploratoire

Algorithme Monté Carlo avec départ exploratoire [Monte Carlo ES]

- Idées clés :**
- 1 Version *Monté Carlo* de l'algorithme *Itération de politique* ;
 - 2 **Alterne épisode par épisode** entre *Évaluation* et *Amélioration de Politique* ;
 - 3 À chaque épisode, considérer **exclusivement de la première visite** fait à un *état-action* pour les calculs (*First-visit MC*)² ;
 - 4 **Choisir aléatoirement le point de départ** de chaque trajectoire échantillonné (départ exploratoire) ;

$$Q_k^\pi \xrightarrow{k \rightarrow \infty} Q_k^* \quad \text{et} \quad \pi_k \xrightarrow{k \rightarrow \infty} \pi^*$$

- 👍 Pro :**
- Adresse le *dilemme d'exploration/exploitation* en utilisant l'idée de **départ exploratoire** ;
 - Contourne le prérequis d'infinité d'épisode en acceptant le fait que chaque étape d'*évaluation de politique* provoque un **plus petit déplacement** $Q_k \rightarrow Q_k^\pi$;
- 👎 Con :**
- La preuve formelle de convergence de *Monte Carlo ES* est un problème ouvert.

2. Il existe également une version *Every-Visit MC* avec des propriétés théoriques légèrement différentes

↕ Algorithme | Monté Carlo avec *départ exploratoire*

$\forall s \in \mathcal{S}$, fixer arbitrairement la **politique courante** π_k tel que $\pi_0(s) = a \in \mathcal{A}(s)$

Répéter (pour chaque trajectoire) avec $k = 1, 2, 3, \dots$ **jusqu'à convergence** vers Q^* et π^* :

- ▶ **Choisir** (le point de départ) $s_0 \in \mathcal{S}$ et $a_0 \in \mathcal{A}(s_0)$ **aléatoirement** ;
- ▶ **Générer une trajectoire** en suivant $\pi_k(\cdot)$ jusqu'à terminaison ;

$$\tau = s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_t, a_t, r_{t+1}, \dots, s_{T-1}, a_{T-1}, r_T$$

Répéter (étape 1 à 3) pour chaque t de la trajectoire :

Seulement si le couple (s_t, a_t) est une première visite (dans cette trajectoire) :

1 $\mathcal{G}(s, a) \leftarrow \mathcal{G}(s, a) \cup \{ G_t^{(k)} \}$

2 **Évaluer la politique courante**

$$Q_{k+1}^{\pi}(s_t, a_t) \leftarrow \frac{1}{|\mathcal{G}(s, a)|} \sum_{G_t \in \mathcal{G}(s, a)} G_t^{(\cdot)}$$

3 **Améliorer la politique courante**

$$\pi_{k+1}(s_t) \leftarrow \arg \max_{a \in \mathcal{A}(s_t)} Q_{k+1}^{\pi}(s_t, a)$$

> Pseudocode | Monté Carlo avec *départ exploratoire* et *moyenne incrémentale*¹

Init: **foreach** $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$ **do** $Q\text{-table}[s, a] \leftarrow 0$ **and** $K\text{-table}[s, a] \leftarrow 0$

Init: **foreach** $s \in \mathcal{S}$ **do** $\pi\text{-table}[s] \leftarrow a \in \mathcal{A}(s)$ arbitrarily

Init: threshold hyperparam(ϵ) > 0 , $\Delta \leftarrow 0$, $\pi\text{-IsStable} \leftarrow \text{true}$

« Critère d'arrêt »

repeat

$(s_0, a_0) \leftarrow \text{random.select}(s \in \mathcal{S}), \text{random.select}(a \in \mathcal{A}(s))$

$\mathcal{T} \leftarrow \text{Generate trajectory following } \pi\text{-table}[s] \text{ from } (s_0, a_0) \text{ to terminal state}$

$G \leftarrow 0$

foreach $t \in \mathcal{T} \mid t = T-1, T-2, \dots, 2, 1, 0$ **do**

$G \leftarrow \gamma G + r_{t+1}$

if (s_t, a_t) **is not in** $\mathcal{T}[0, t-1]$ **then** it's first-visit

$Q\text{-table}_{old} \leftarrow Q\text{-table}, a_{old} \leftarrow \pi\text{-table}[s_t]$

« pour la vérification de stabilité »

1 %....Étape d'évaluation de politique.....

$K\text{-table}[s_t, a_t] \leftarrow K\text{-table}[s_t, a_t] + 1$

$Q\text{-table}[s_t, a_t] \leftarrow Q\text{-table}_{old}[s_t, a_t] + 1/\text{len}(K\text{-table}[s_t, a_t]) \left[G_t - \gamma Q\text{-table}_{old}[s_t, a_t] \right]$

« Backup operation »

2 %....Étape d'amélioration de politique.....

$\pi\text{-table}[s_t] \leftarrow \arg \max_{a \in \mathcal{A}(s_t)} Q\text{-table}[s_t, a]$

« Backup operation »

3 %....Étape de vérification de stabilité.....

$\Delta \leftarrow \max \left(\Delta, \left| Q\text{-table}_{old}[s_t, a_t] - Q\text{-table}[s_t, a_t] \right| \right)$

if $a_{old} \neq \pi\text{-table}[s]$ **then** $\pi\text{-IsStable} \leftarrow \text{false}$

until $\Delta < \epsilon$ **and** $\pi\text{-IsStable}$ for a arbitrary number of trajectories

« $Q^\pi \xrightarrow{\approx} Q^*$ et $\pi \xrightarrow{\approx} \pi^*$ »

Output: $\pi^*(s) \approx \pi\text{-table}[s]$

« Produit une politique déterministe »

1. Pseudocode inspiré de « Reinforcement Learning : An introduction », chapitre 5.3, page 99 « Monte Carlo Control » par Sutton & Barto [2]

Apprentissage par renforcement (sans modèle)

RL par Différence temporelle [TD]

Méthodes par Différence Temporelle

« Try something & learn from mistake has you go. »

- Caractéristiques :**
- **Intéragit** avec l'environnement pour recueillir de l'information ;
 - Collecte des **échantillons** de trajectoire τ **partiel** ;
 - « Bootstrap » comme avec les méthodes de *DP* ;
 - Phase d'apprentissage **en-ligne** ;
 - Apprentissage **sans-modèle** ;
 - Exploite la *propriété de Markov* ;

Idées clés : Apprendre **au fur et à mesure**, en se basant en partie sur des **échantillons** d'interaction avec l'environnement et en partie sur l'**estimation courante** (« Bootstrap ») ;

Méthodes par Différence Temporelle (considération pratique)

« Update rule », forme générale (rappel)

$$\text{Estimate}^{\text{New}} \leftarrow \text{Estimate}^{\text{Old}} + \text{StepSize} \left[\underbrace{\text{Target} - \text{Estimate}^{\text{Old}}}_{\text{estimate « error »}} \right]$$

>_ En pratique | Backup operation des méthodes par Différence Temporelle

Idées clés : Donner plus de poids aux expériences récentes ;

Comment ? Calculer la moyenne pondérée des récompenses passés ; (aka : Exponential moving average)

Exemple (Algorithme SARSA) :

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha \left[\underbrace{r(s_t, a_t, s_{t+1}) + Q_k(s_{t+1}, a_{t+1})}_{\text{TD target}} - Q_k(s_t, a_t) \right]$$

avec le « learning rate » $\alpha \in]0, 1]$, l'échantillon $r(s_t, a_t, s_{t+1})$ et Q_k un estimateur du vrai Q_k^π .

Méthodes par Différence Temporelle (considération pratique)

« Update rule », forme générale (rappel)

$$\text{Estimate}^{\text{New}} \leftarrow \text{Estimate}^{\text{Old}} + \text{StepSize} \left[\underbrace{\text{Target} - \text{Estimate}^{\text{Old}}}_{\text{estimate « error »}} \right]$$

➤ En pratique | Backup operation des méthodes par Différence Temporelle

Idées clés : Donner plus de poids aux expériences récentes ;

Comment ? Calculer la moyenne pondérée des récompenses passés ; (aka : Exponential moving average)

Exemple (Algorithme SARSA) :

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha \left[\underbrace{r(s_t, a_t, s_{t+1}) + Q_k(s_{t+1}, a_{t+1})}_{\text{TD target}} - Q_k(s_t, a_t) \right]$$

avec le « learning rate » $\alpha \in]0, 1]$, l'échantillon $r(s_t, a_t, s_{t+1})$ et Q_k un estimateur du vrai Q_k^π .

Méthodes par Différence Temporelle (considération pratique)

« Update rule », forme générale (rappel)

$$\text{Estimate}^{\text{New}} \leftarrow \text{Estimate}^{\text{Old}} + \text{StepSize} \left[\underbrace{\text{Target} - \text{Estimate}^{\text{Old}}}_{\text{estimate « error »}} \right]$$

➤ En pratique | Backup operation des méthodes par Différence Temporelle

Idées clés : Donner plus de poids aux expériences récentes ;

Comment ? Calculer la moyenne pondérée des récompenses passés ; (aka : Exponential moving average)

Exemple (Algorithme SARSA) :

$$Q_{k+1}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q_k(\mathbf{s}_t, \mathbf{a}_t) + \alpha \left[\underbrace{r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) + Q_k(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})}_{\text{TD target}} - Q_k(\mathbf{s}_t, \mathbf{a}_t) \right]$$

avec le « learning rate » $\alpha \in]0, 1]$, l'échantillon $r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$ et Q_k un estimateur du vrai Q_k^π .

Méthodes par Différence Temporelle (considération pratique)

« Update rule », forme générale (rappel)

$$\text{Estimate}^{\text{New}} \leftarrow \text{Estimate}^{\text{Old}} + \text{StepSize} \left[\underbrace{\text{Target} - \text{Estimate}^{\text{Old}}}_{\text{estimate « error »}} \right]$$

➤ En pratique | Backup operation des méthodes par Différence Temporelle

Idées clés : Donner plus de poids aux expériences récentes ;

Comment ? Calculer la moyenne pondérée des récompenses passés ; (aka : Exponential moving average)

Exemple (Algorithme SARSA) :

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha \left[\underbrace{r(s_t, a_t, s_{t+1}) + Q_k(s_{t+1}, a_{t+1})}_{\text{TD target}} - Q_k(s_t, a_t) \right]$$

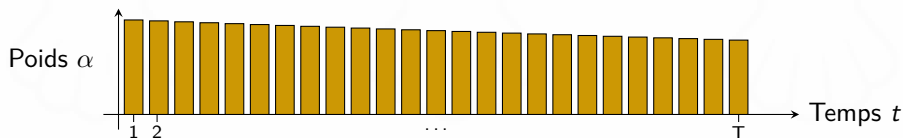
avec le « learning rate » $\alpha \in]0, 1]$, l'échantillon $r(s_t, a_t, s_{t+1})$ et Q_k un estimateur du vrai Q_k^π .

Méthodes par Différence Temporelle (considération pratique)

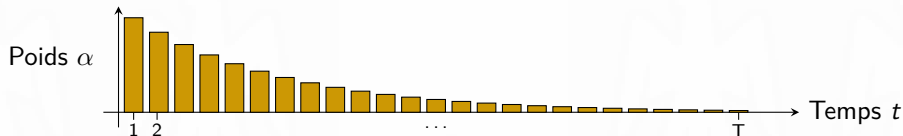
Learning rate $\alpha \in]0, 1]$

$$Q_{k+1}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q_k(\mathbf{s}_t, \mathbf{a}_t) + \alpha \left[r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) + Q_k(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - Q_k(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Exemple avec $\alpha = 0.97$: « L'agent tiens compte des expériences passée, comme si ça venait d'arrivé »



Exemple avec $\alpha = 0.5$: « L'agent est plus préoccupé par les expériences récentes »



Méthodes par Différence Temporelle

- 👍 **Pro :**
- Pas besoin de modèle ;
 - Peu apprendre **au fur et à mesure** \implies est un avantage lorsque les trajectoires sont longues à exécuter ;
 - Peu apprendre **de trajectoire incomplète** \implies est un avantage lorsque on n'a pas de garantie que les trajectoires termine toujours ;
 - **Peu de variance**¹ ;
- 👎 **Con :**
- **Légèrement biaisé**² ;
 - Les valeurs prises lors de l'initialisation peuvent avoir une incidence sur le résultat de l'apprentissage ;

1. car le « TD-target » $r(s_t, a_t, s_{t+1}) + \gamma Q(s_{t+1}, a_{t+1})$ dépend de phénomène aléatoire sur un seul « step » (sélection d'action, transition, récompense)

2. car le « TD-target » $r(s_t, a_t, s_{t+1}) + \gamma Q(s_{t+1}, a_{t+1})$ est un estimé biaisé du vrai $Q^\pi(s_t, a_t)$

Q-Learning (tabulaire) [Tabular Q-Learning]

« MOTO »

Idées clés : ■ C'est une approximation stochastique de l'algorithme *Itération de Q-valeur* ;

$$Q_i^\pi \xrightarrow{i \rightarrow \infty} Q^*$$

👍 **Pro :** ■ Fusce condimentum mi et urna aliquam, non euismod quam fermentum. ;

👎 **Con :** ■ Proin egestas urna vulputate lorem pharetra, in posuere ipsum posuere ;

☐ **ToDo:** MOTO

☐ **ToDo:** Idées clés

☐ **ToDo:** Pro and Con

↕ Algorithme | Q-Learning (tabulaire)

Répéter avec $i = 1, 2, 3, \dots$ jusqu'à convergence vers Q^* :

$$Q_{i+1}^\pi(s, a) \leftarrow \text{Fusce condimentum mi et urna aliquam, non euismod quam fermentum.}$$

> En pratique | Q-Learning (tabulaire)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam in egestas magna, vel molestie mauris.

Conséquence :

- ▶ Sed vehicula felis at felis posuere efficitur quis sit amet arcu.

□ **ToDo:** math algorithm

□ **ToDo:** applied consideration

➤ Pseudocode | Q-Learning (tabulaire)¹

Init: $Q\text{-table}[s, a] \leftarrow 0 \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$

Init: threshold hyperparam(ϵ) > 0

« Critère d'arrêt »

repeat

« sweep i »

$\Delta \leftarrow 0$

foreach $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$ **do**

$Q_{old} \leftarrow Q\text{-table}[s, a]$

$Q\text{-table}[s, a] \leftarrow \sum_{s' \in \mathcal{S}} p(s'|s, a) \left[r(s, a, s') + \gamma \max_{a' \in \mathcal{A}(s')} Q\text{-table}[s', a'] \right]$

« Backup operation »

$\Delta \leftarrow \max \left(\Delta, \left| Q_{old} - Q\text{-table}[s, a] \right| \right)$

until $\Delta < \epsilon$

« $Q^\pi \xrightarrow{\approx} Q^*$ »

Output: $\pi^*(s) \approx \arg \max_{a \in \mathcal{A}(s)} Q\text{-table}[s, a]$

« Produit une politique déterministe »

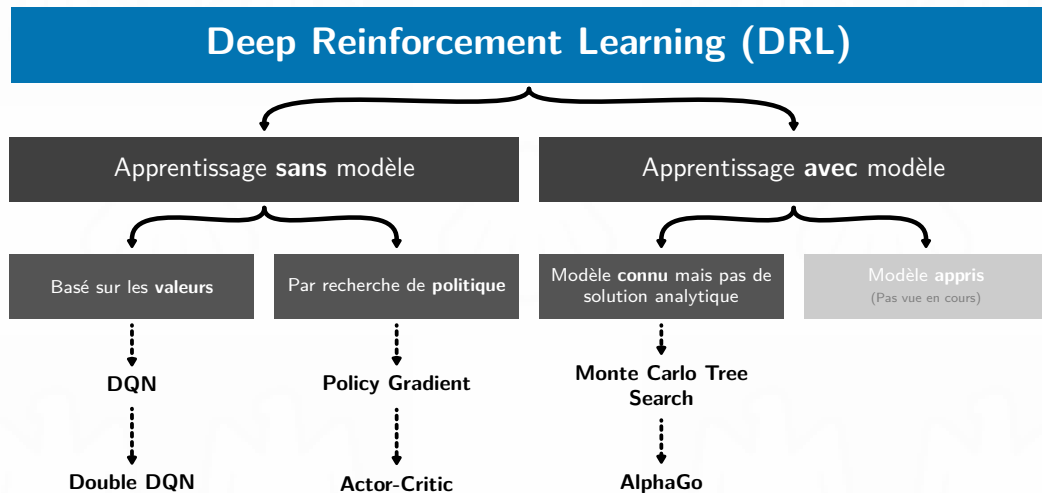
❑ **ToDo:** PSEUDOCODE

❑ **ToDo:** RÉFÉRENCE

1. Pseudocode inspiré de « PAPERTITLE », section TODO, page TODO par TODO [TODO]

Prochaine partie

Prochaine partie



Pour aller plus loin

1 Concept clé en RL

2 Paysage algorithmique de l'apprentissage par renforcement

3 Modèle connu **et** parfait \implies pas d'apprentissage requis

4 Apprentissage par renforcement (sans modèle)

5 Pour aller plus loin

- Complément théorique
- Ressource théorique
- Framework et outil d'implémentation
- Références

Pour aller plus loin

Complément théorique

Pour aller plus loin

Ressource théorique

Programmation dynamique

- « Dynamic Programming and Optimal Control », Vol. 1 et 2, 4e Edition

[► Publication](#)

par Dimitri P. Bertsekas

Abstract : ... oriented towards modeling, conceptualization, finite-horizon problems, ... mathematical analysis and computation, treats infinite horizon problems extensively, and provides an up-to-date account of approximate large-scale dynamic programming and reinforcement learning ... ;

- « Reinforcement Learning : An introduction », chapitre 4 [2]

[► Publication](#)

par Sutton & Barto

Algorithme : *Évaluation de politique, Amélioration de politique, Itération de V-valeur, Itération de politique et Itération de politique généralisé (GPI) ;*

Différence temporel [Temporal Difference (TD)]

- « The Paths Perspective on Value Learning », Distill, 2019 [3]

[► www](#)

par Greydanus & Olah

Algorithme : *Sarsa, Expected Sarsa, Q-learning et Double Q-learning ;*

Abstract : A closer look at how Temporal Difference learning merges paths of experience for greater statistical efficiency ;

Pour aller plus loin

Framework et outil d'implémentation

MDP en pratique :

- Markov Decision Process (MDP) Toolbox for Python [► Doc](#)

Pour aller plus loin

Références

References I

1. BARTO, A. G. Reinforcement Learning and Dynamic Programming. *IFAC Proceedings Volumes* **28**, 407-412. ISSN : 14746670 (1995).
2. SUTTON, R. S. & BARTO, A. G. *Reinforcement learning: An introduction*. 2^e éd. (éd. MIT PRESS) ISBN : 978-0262039246.
<http://incompleteideas.net/book/RLbook2018.pdf> (Cambridge, MA, 2018).
3. GREYDANUS, S. & OLAH, C. The Paths Perspective on Value Learning. *Distill* **4**.
<https://distill.pub/2019/paths-perspective-on-value-learning> (2019).