

Apprentissage par renforcement appliqué

Algorithmes fondamentaux

[revision 3.3]

Brahim Chaib-draa

Brahim.Chaib-Draa@ift.ulaval.ca



Université Laval

2020-06-18

- 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 *basé sur un modèle*
- Apprentissage vs Planification
- Prédiction 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 HAUT niveau :

- Apprentissage **sans mod le** vs apprentissage **bas  sur un mod le**
[*Model-free vs model-based*]
- **Apprentissage vs Planification**
[*Learning vs Planning*]

Concept BAS niveau :

- **Pr diction vs Contr le**
[*Prediction vs Control*]
- Apprentissage **EN-ligne** vs **HORS-ligne**
[*ON-line learning vs OFF-line learning*]

Concept clé en RL

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

Apprentissage sans mod le vs apprentissage bas  sur un mod le

Concept HAUT niveau

M thode sans mod le [Model-free]

- L'agent **ne d pend pas d'un mod le** pour apprendre ;
- Il apprend par essai et erreur ;



M thode bas e sur un mod le [Model-based]

- 1 Soit l'agent **apprend un mod le** de A   Z ;
- 2 ou l'agent **am liore un mod le** par exp rience ;
- 3 ou l'agent connaît le mod le   priori ;



Concept clé en RL

Apprentissage vs Planification

Apprentissage vs Planification

Concept HAUT niveau

Apprentissage [Learning]

Acqu rir des habilet s ou des connaissances soit par :

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



Planification [Planning]

D eterminer les  tapes n cessaires   l'atteinte d'un **but** et mesurer les efforts n cessaires ;

Pr requis : avoir **un mod le de l'environnement**

Ex. un calendrier, une carte, les r gles de physique, les r gles d'un jeu ...



Concept clé en RL

Prédiction vs Contrôle

Pr diction vs Contrôle

Concept BAS niveau

Prediction [Prediction]

Bas  sur mes **connaissances actuelles**, qu'est-ce que je peux d duire/induire (  propos du **futur**)¹.

Implique une notion d'incertitude.

Ex. de pr diction : estimer Q^π ou V^π



Contrôle [Control]

Choisir une strat gie **optimale** afin d'atteindre un **but**.

Ex. de contr le : estimer V^* , Q^* ou π^*



1. Contrairement   son utilisation dans le domaine des statistiques o  le terme pr diction n'implique pas n cessairement la notion de futur.

Concept clé en RL

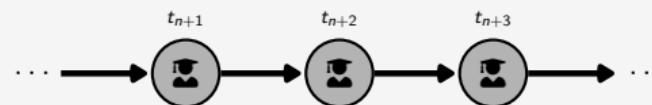
Apprentissage EN-ligne vs HORS-ligne

Apprentissage EN-ligne vs HORS-ligne

Concept BAS niveau

Apprentissage EN-ligne [ON-line learning]

L'apprentissage se fait **au fur et   mesure** que l'agent ex cute des actions dans l'environnement.



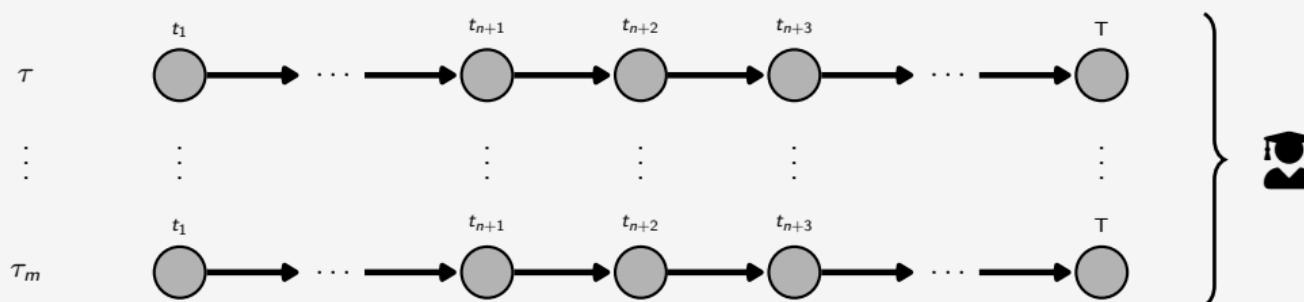
Exemple : *TD-learning*.

Apprentissage EN-ligne vs HORS-ligne

Concept BAS niveau

Apprentissage HORS-ligne [OFF-line learning]

La phase d'apprentissage est retard e apr s la collecte d'une s rie de trajectoires τ .



avec m le nombre de trajectoires collect  [$m = \text{batch size}$]

Ex : Batch RL. [wikipedia](#).

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

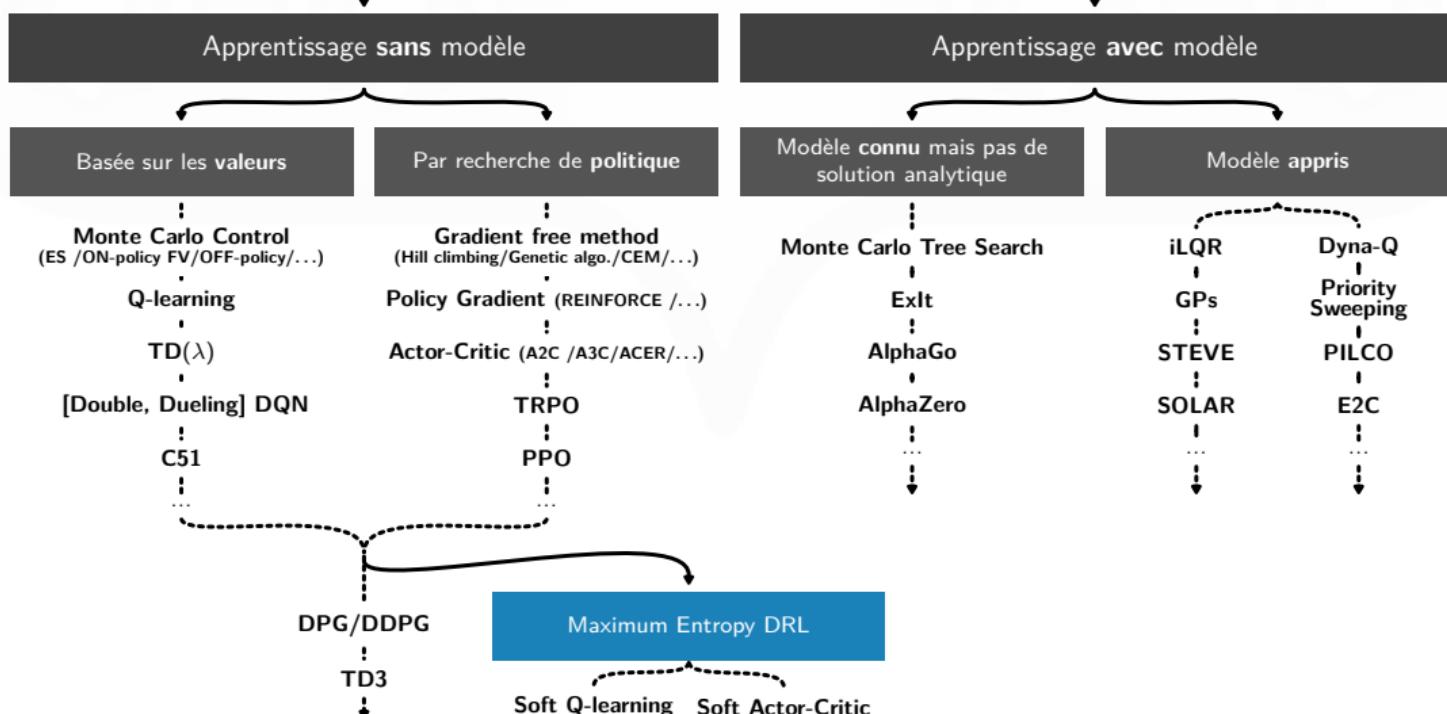
4 Apprentissage par renforcement (sans modèle)

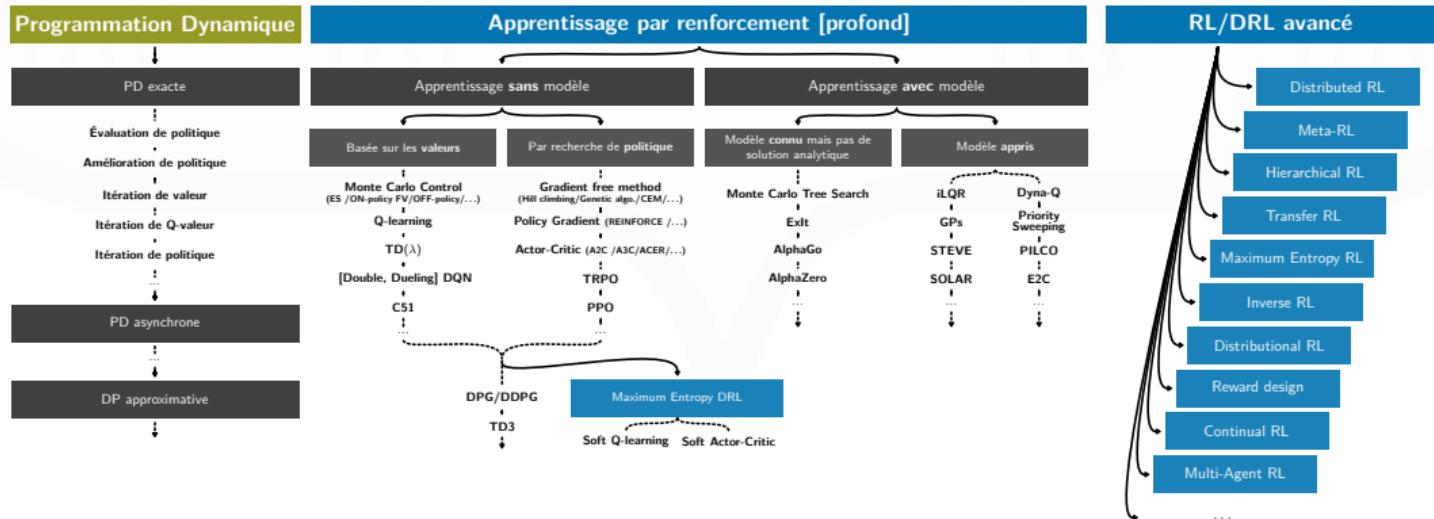
5 Pour aller plus loin

Paysage algorithmique de l'apprentissage par renforcement

Taxonomie du RL/DRL

Apprentissage par renforcement [profond]





Paysage algorithme de l'apprentissage par renforcement

Pourquoi autant d'algorithmes ?

Pourquoi autant d'algorithmes ?

Diff  ent compromis ex. : On a peu de puissance de calcul, mais g  nerer de nouvelles trajectoires est tr  s rapide, donc c'est plus efficace d'utiliser un algorithme de type *Monte-Carlo* qu'un par *Diff  rence Temporelle* ;

Diff  entes assomptions ex : On fait l'assumption qu'on a acc  s   un mod  le parfait, donc on peut utiliser un algorithme de *Programmation Dynamique* ;

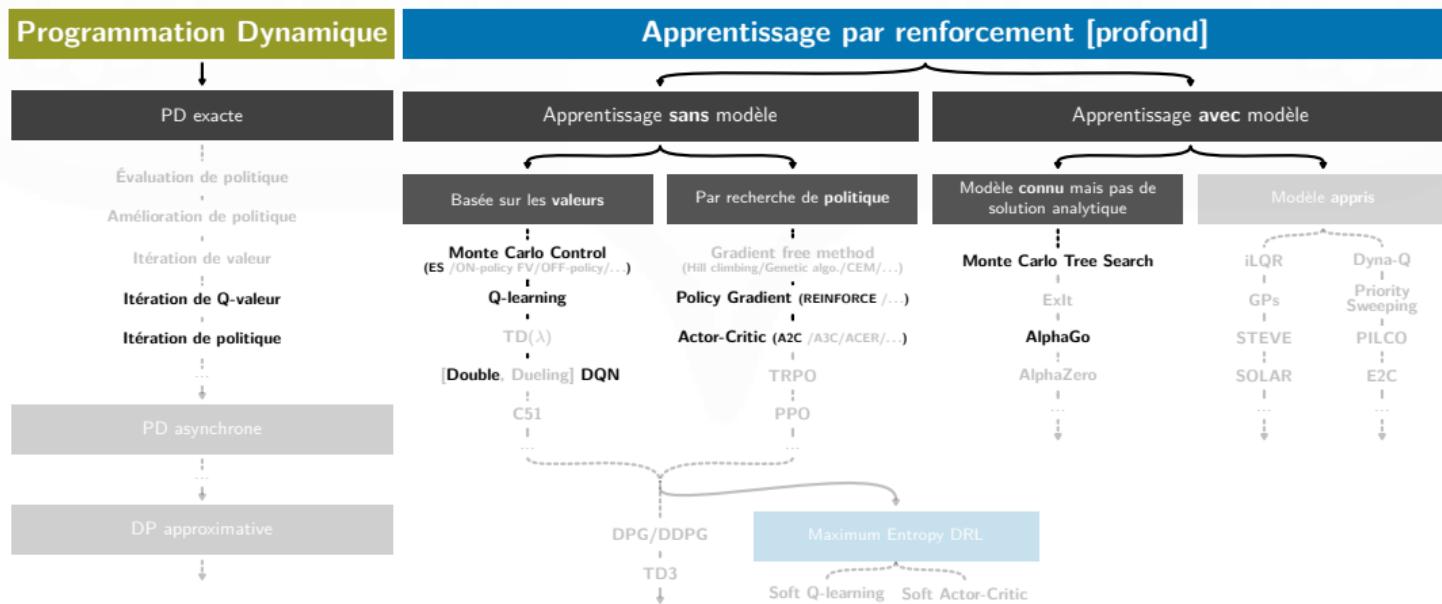
Diff  rente contrainte Ex. : On a besoin d'un algorithme qui fonctionne sur des espaces d'actions continues, donc on ne peut pas utiliser les algorithmes de la famille *Q-learning* ;

Plus de d  tail au fur et   mesure

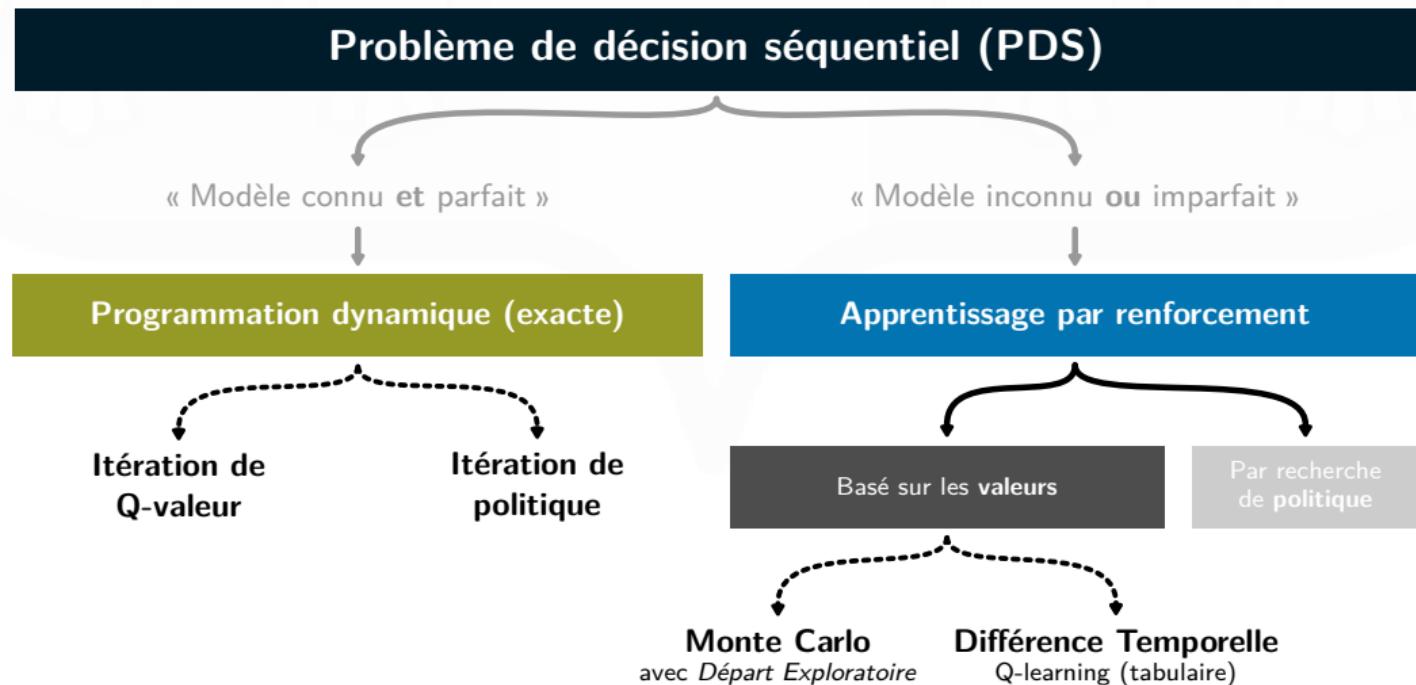
Note : Voir *Reinforcement Learning coach>Selecting an Algorithm* pour un bon exemple illustrant le niveau de complexit  que repr  sente le choix d'algorithme en RL.

▶ [RL Coach / selecting an algorithm](#)

Algorithmes couverts dans le cours



Algorithmes vu dans cette partie



Paysage algorithmique de l'apprentissage par renforcement

Comparaison des types de méthode en RL

Terminologie

« Backup Operation »

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

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

1. Voir l'excellente explication par Andre G. Barto pour plus de d tail dans REINFORCEMENT LEARNING AND DYNAMIC PROGRAMMING, section 2.2 [Barto1995]

Programmation dynamique : Estimer la valeur en utilisant l'**esp rance math matique**, le **mod le** de l'environnement et l'**estimation courantes** ;
 Ex. *Backup operation* pour It ration 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 m thode Monte-Carlo : Estimer la valeur en utilisant la **moyenne empirique** sur des ** chantillons de trajectoire τ complets** ;
 Ex. *Backup operation* pour *Monte Carlo ES*

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

RL par Diff rence Temporelle : Estimer la valeur en utilisant des ** chantillons de trajectoire τ partielle + l'estimation courante** du reste de la trajectoire ;
 Ex. *Backup operation* pour *Q-learning*

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

Programmation dynamique : Estimer la valeur en utilisant l'esp rance math matique, le mod le de l'environnement et l'estimation courantes ;

Ex. *Backup operation* pour It ration de Q-valeur

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

RL par m thode Monte-Carlo : Estimer la valeur en utilisant la moyenne empirique sur des chantillons de trajectoire τ complets ;

Ex. *Backup operation* pour Monte Carlo ES

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

RL par Diff rence Temporelle : Estimer la valeur en utilisant des chantillons de trajectoire τ partielle + l'estimation courante du reste de la trajectoire ;

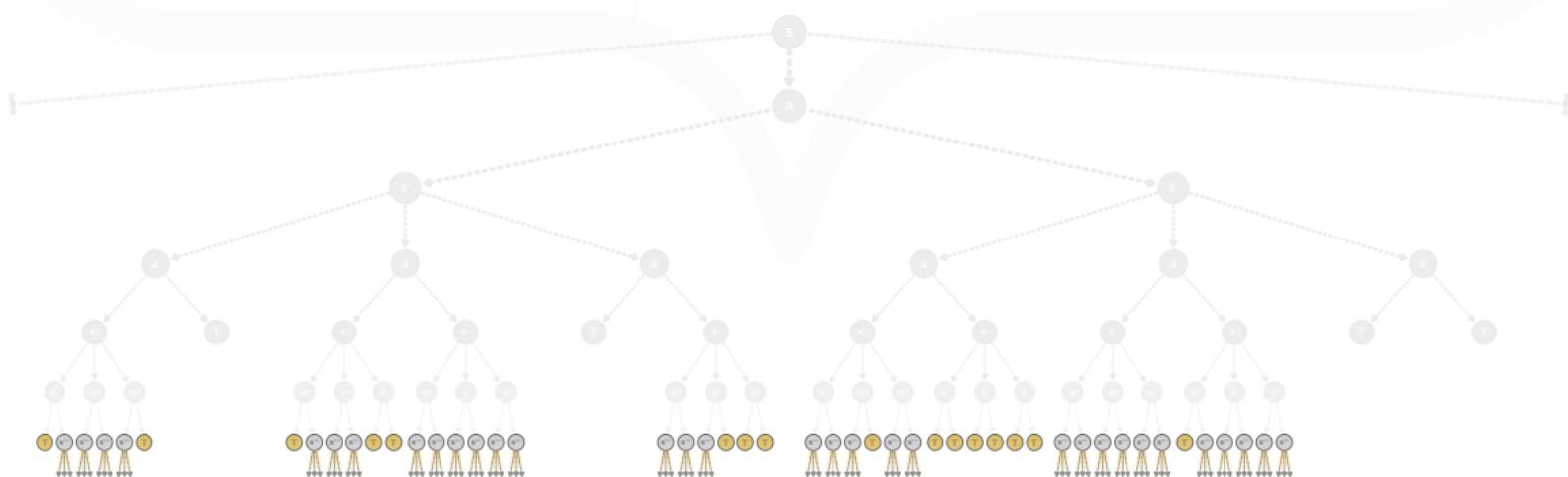
Ex. *Backup operation* pour *Q-learning*

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

Backup diagram : Programmation Dynamique

Ex.: Algorithme *It ration de Q-valeur*

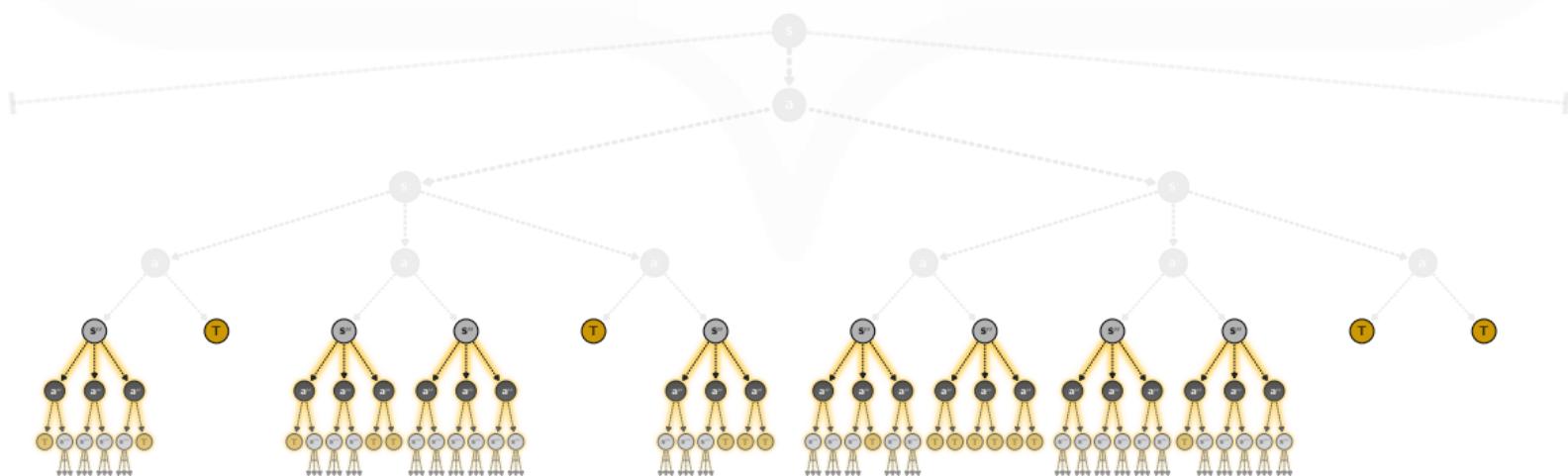
$$Q_{i+1}^{\pi}(s, a) \leftarrow \mathbb{E}_{s' \sim p} \left[r(s, a, s') + \gamma \max_{a' \in \mathcal{A}(s')} Q^{\pi}(s', a') \right]$$



Backup diagram : Programmation Dynamique

Ex.: Algorithme *It ration de Q-valeur*

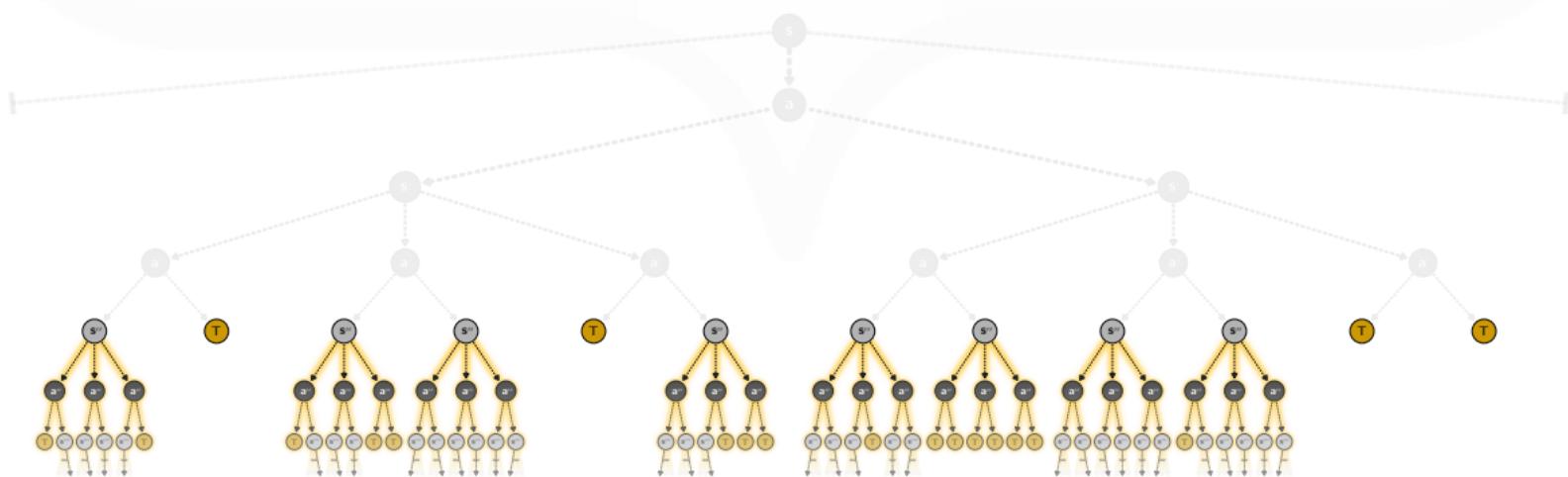
$$Q_{i+1}^{\pi}(s, a) \leftarrow \mathbb{E}_{s' \sim p} \left[r(s, a, s') + \gamma \max_{a' \in \mathcal{A}(s')} Q^{\pi}(s', a') \right]$$



Backup diagram : Programmation Dynamique

Ex.: Algorithme *It ration de Q-valeur*

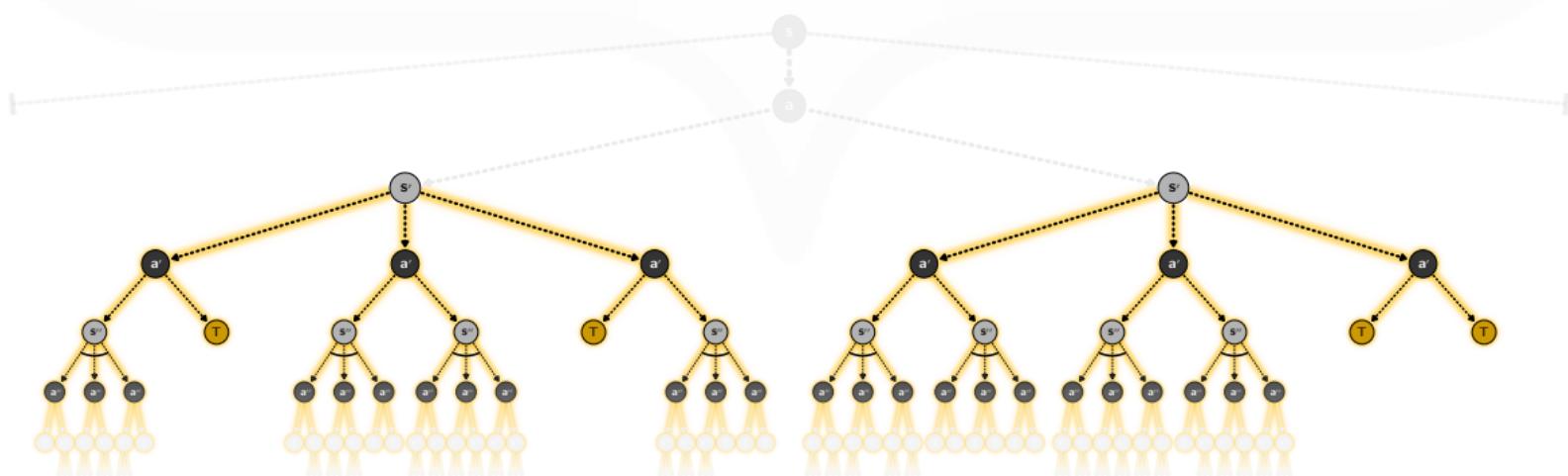
$$Q_{i+1}^{\pi}(s, a) \leftarrow \mathbb{E}_{s' \sim p} \left[r(s, a, s') + \gamma \max_{a' \in \mathcal{A}(s')} Q^{\pi}(s', a') \right]$$



Backup diagram : Programmation Dynamique

Ex.: Algorithme *It ration de Q-valeur*

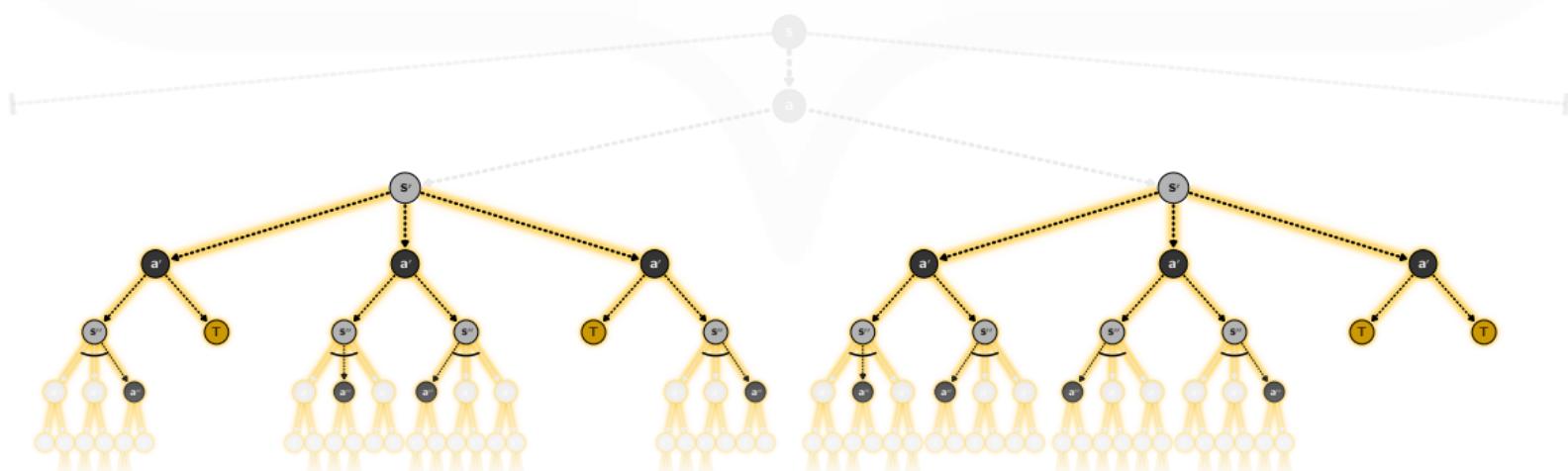
$$Q_{i+1}^{\pi}(s, a) \leftarrow \mathbb{E}_{s' \sim p} \left[r(s, a, s') + \gamma \max_{a' \in \mathcal{A}(s')} Q^{\pi}(s', a') \right]$$



Backup diagram : Programmation Dynamique

Ex.: Algorithme *It ration de Q-valeur*

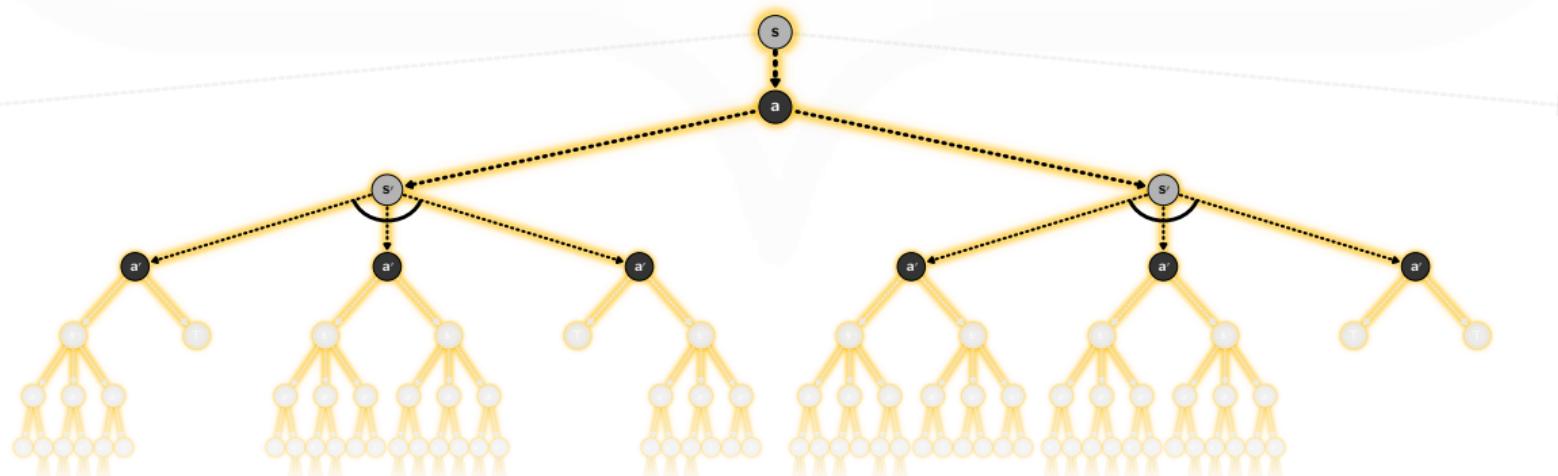
$$Q_{i+1}^{\pi}(s, a) \leftarrow \mathbb{E}_{s' \sim p} \left[r(s, a, s') + \gamma \max_{a' \in \mathcal{A}(s')} Q^{\pi}(s', a') \right]$$



Backup diagram : Programmation Dynamique

Ex.: Algorithme *It ration de Q-valeur*

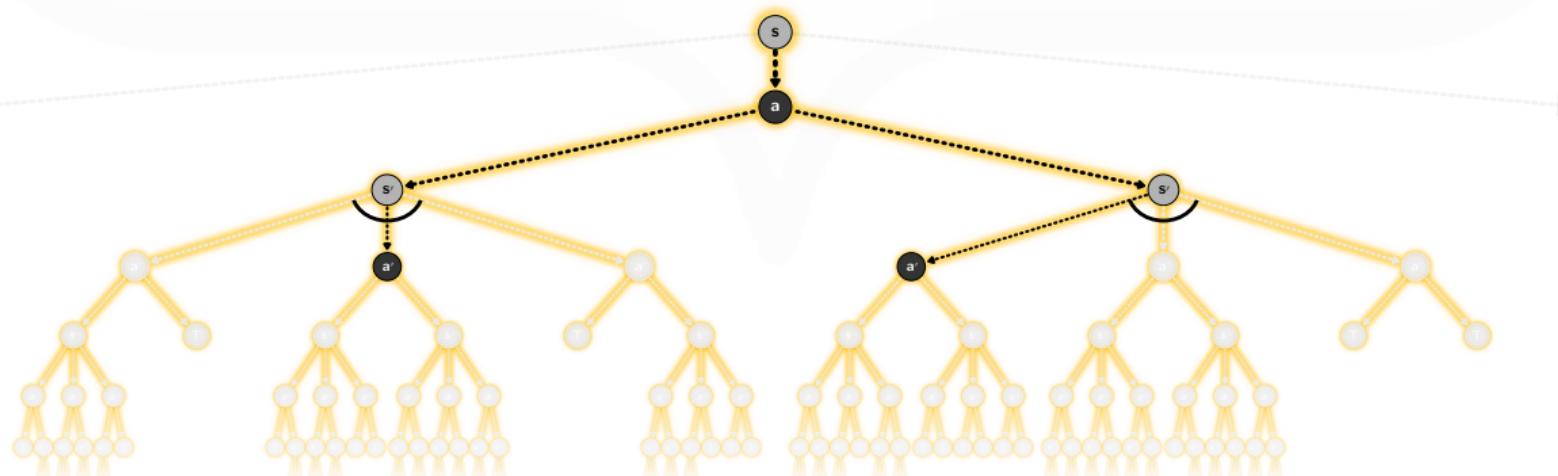
$$Q_{i+1}^{\pi}(s, a) \leftarrow \mathbb{E}_{s' \sim p} \left[r(s, a, s') + \gamma \max_{a' \in \mathcal{A}(s')} Q^{\pi}(s', a') \right]$$



Backup diagram : Programmation Dynamique

Ex.: Algorithme *It ration de Q-valeur*

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



Backup diagram : m thode Monte-Carlo

Ex.: Algorithme Monte-Carlo *ES*

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



Backup diagram : m thode Monte-Carlo

Ex.: Algorithme Monte-Carlo ES

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



Backup diagram : m thode Monte-Carlo

Ex.: Algorithme Monte-Carlo ES

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



Backup diagram : m thode Monte-Carlo

Ex.: Algorithme Monte-Carlo ES

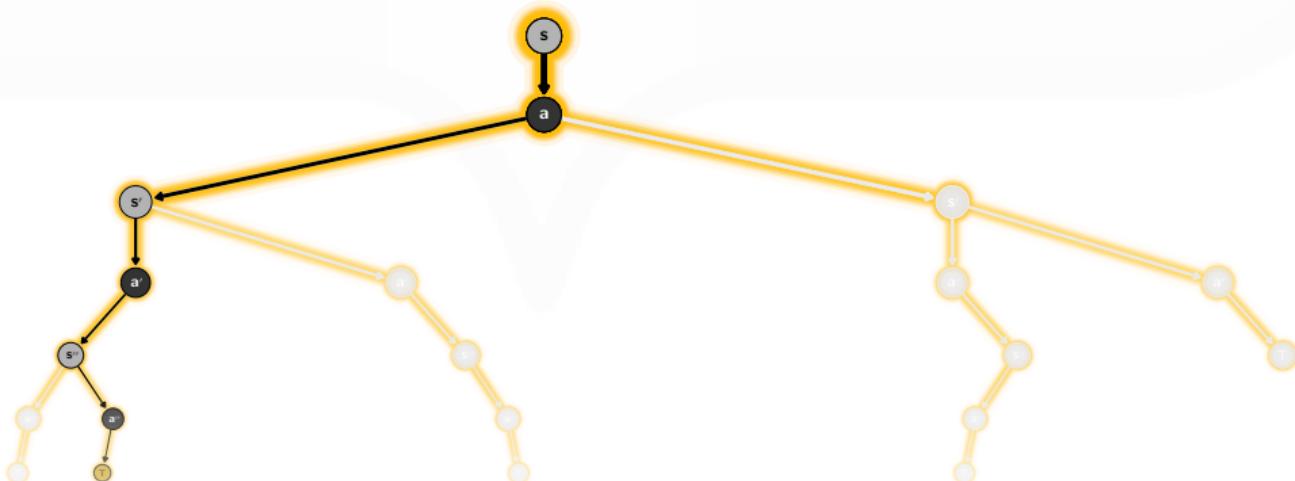
$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \frac{1}{k} \left[G_t^{(k)}(\tau) - Q^\pi(s_t, a_t) \right]$$



Backup diagram : m thode Monte-Carlo

Ex.: Algorithme Monte-Carlo ES

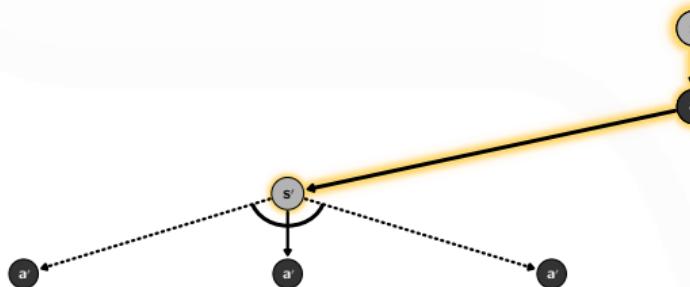
$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \frac{1}{k} \left[G_t^{(k)}(\tau) - Q^\pi(s_t, a_t) \right]$$



Backup diagram : m thode par Diff rence Temporelle

Ex.: Algorithme *Q-Learning*

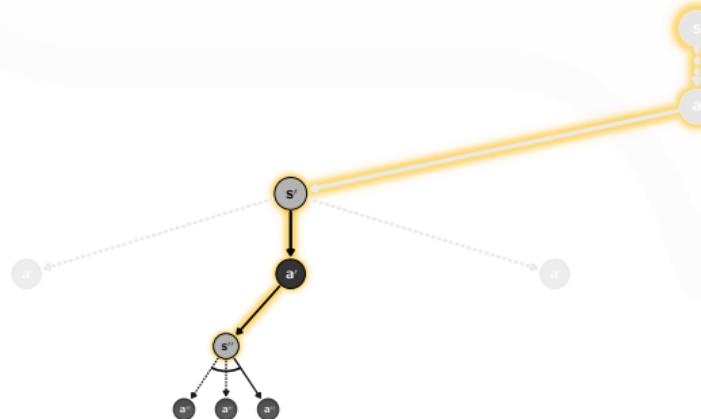
$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha \left[r^{(k)}(s_t, a_t, s_{t+1}) + \gamma \max_{a' \in \mathcal{A}(s_{t+1})} Q^\pi(s_{t+1}, a') - Q^\pi(s_t, a_t) \right]$$



Backup diagram : m thode par Diff rence Temporelle

Ex.: Algorithme *Q-Learning*

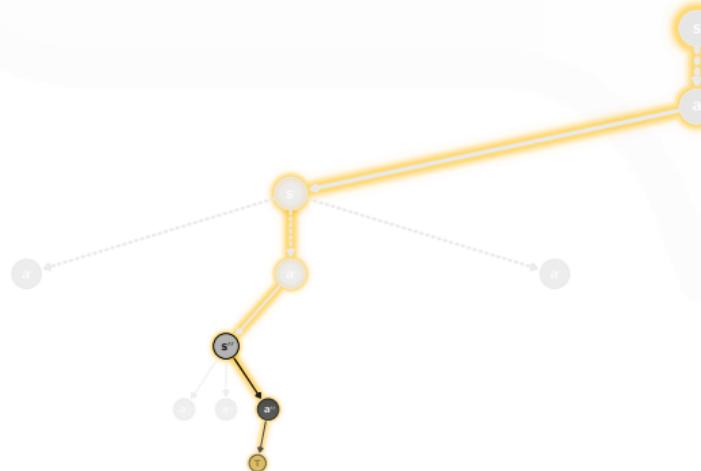
$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha \left[r^{(k)}(s_t, a_t, s_{t+1}) + \gamma \max_{a' \in \mathcal{A}(s_{t+1})} Q^\pi(s_{t+1}, a') - Q^\pi(s_t, a_t) \right]$$



Backup diagram : m thode par Diff rence Temporelle

Ex.: Algorithme *Q-Learning*

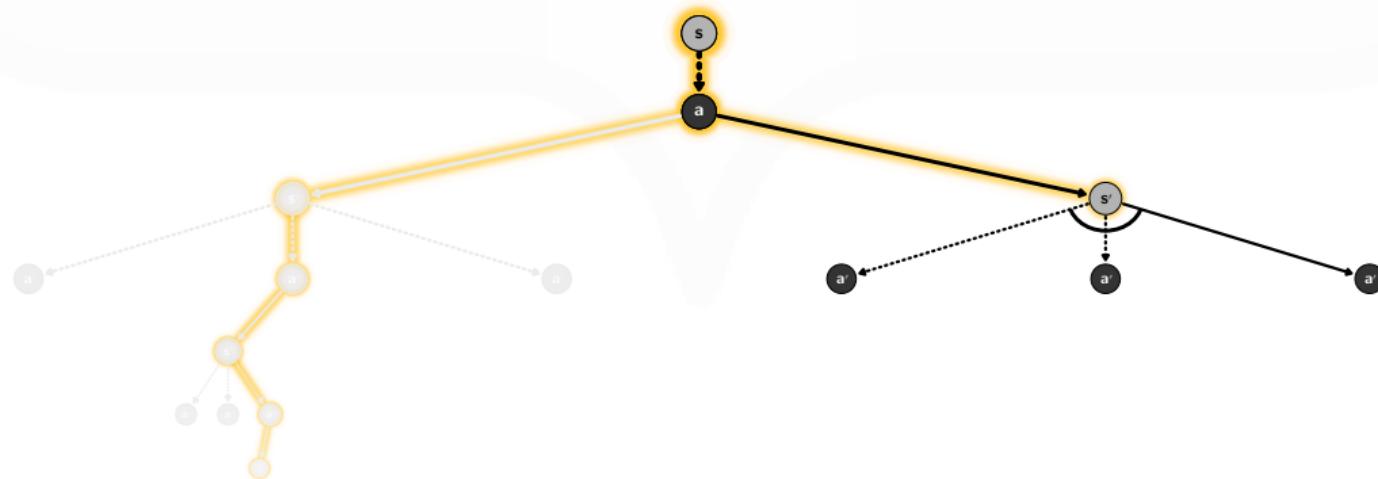
$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha \left[r^{(k)}(s_t, a_t, s_{t+1}) + \gamma \max_{a' \in \mathcal{A}(s_{t+1})} Q^\pi(s_{t+1}, a') - Q^\pi(s_t, a_t) \right]$$



Backup diagram : m thode par Diff rence Temporelle

Ex.: Algorithme *Q-Learning*

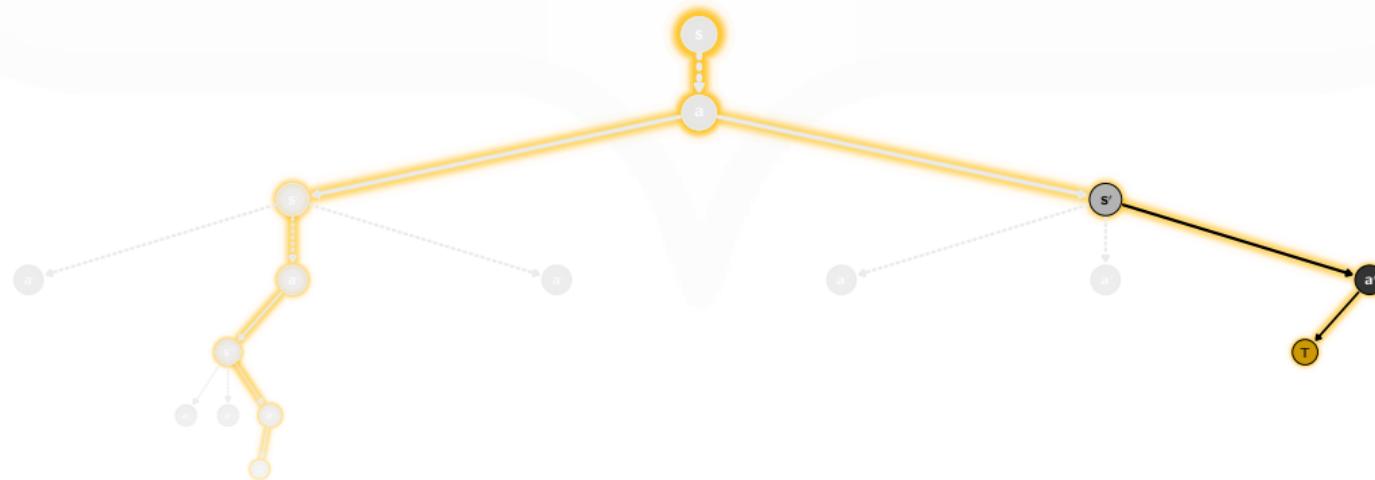
$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha \left[r^{(k)}(s_t, a_t, s_{t+1}) + \gamma \max_{a' \in \mathcal{A}(s_{t+1})} Q^\pi(s_{t+1}, a') - Q^\pi(s_t, a_t) \right]$$



Backup diagram : m thode par Diff rence Temporelle

Ex.: Algorithme *Q-Learning*

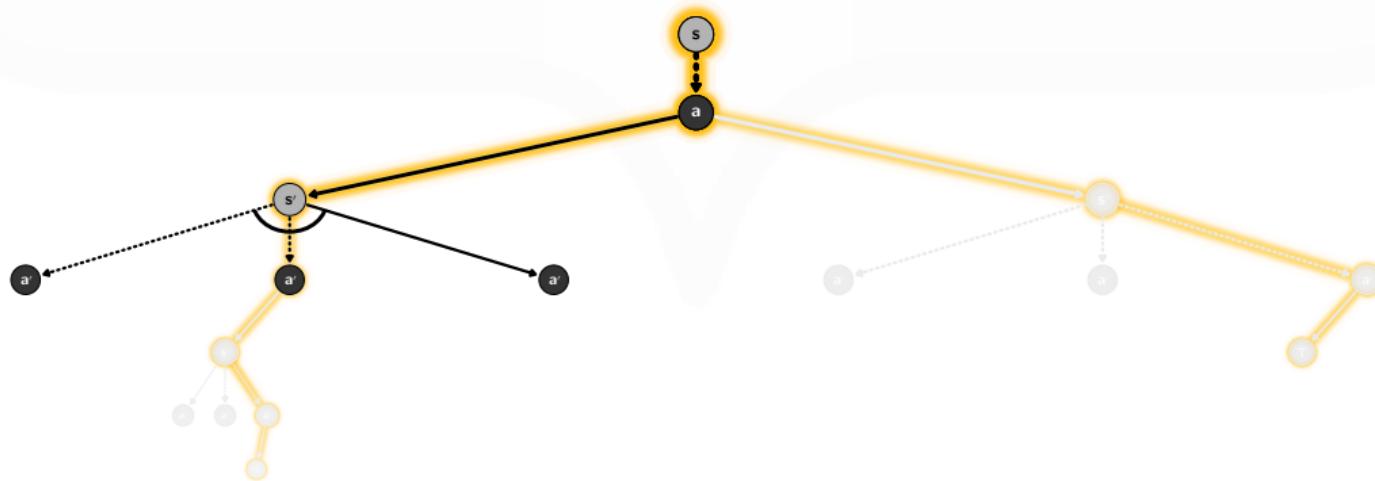
$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha \left[r^{(k)}(s_t, a_t, s_{t+1}) + \gamma \max_{a' \in \mathcal{A}(s_{t+1})} Q^\pi(s_{t+1}, a') - Q^\pi(s_t, a_t) \right]$$



Backup diagram : m thode par Diff rence Temporelle

Ex.: Algorithme *Q-Learning*

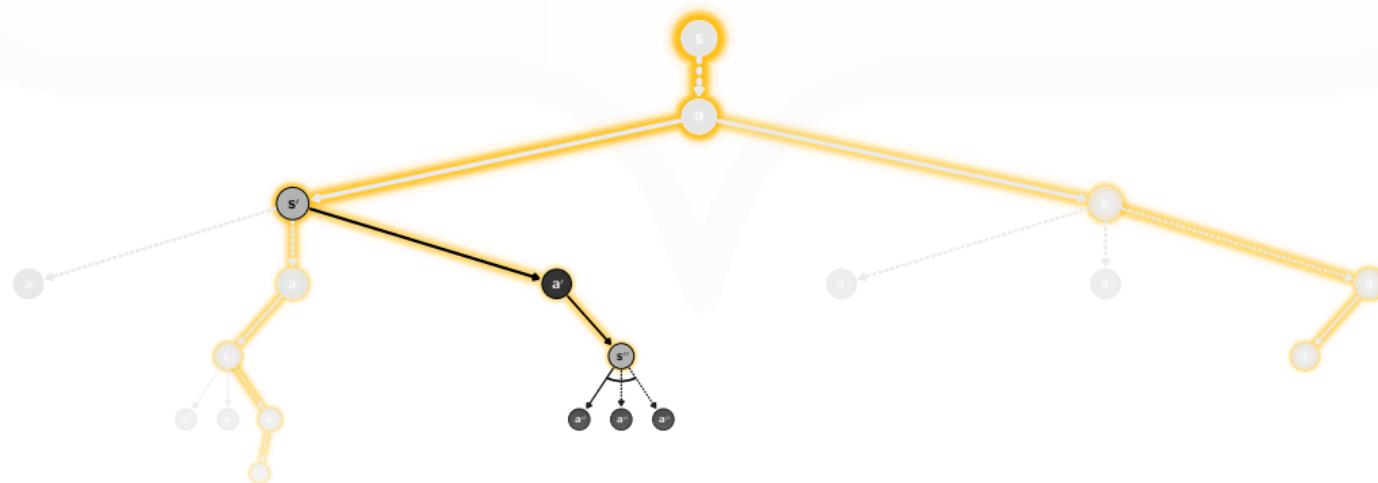
$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha \left[r^{(k)}(s_t, a_t, s_{t+1}) + \gamma \max_{a' \in \mathcal{A}(s_{t+1})} Q^\pi(s_{t+1}, a') - Q^\pi(s_t, a_t) \right]$$



Backup diagram : m thode par Diff rence Temporelle

Ex.: Algorithme *Q-Learning*

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



Backup diagram : m thode par Diff rence Temporelle

Ex.: Algorithme *Q-Learning*

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



Backup diagram : m thode par Diff rence Temporelle

Ex.: Algorithme *Q-Learning*

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



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

Méthode par **Programmation Dynamique exacte**

Programmation dynamique (exacte) et processus de d cision s quentielle

« Supposons qu'apr s avoir lu un livre sur les cubes Rubik, on sait comment r soudre tous les cubes Rubik ... juste en r f chissant et sans jamais en avoir pris un dans ses mains. »



Image : Cube Rubik

- Caract ristique :**
- R solution par planification \implies requiert un mod le de l'environnement ;
 - Type d'algorithme hors-ligne et sans phase d'apprentissage ;
c.- -d. que la solution est calcul e sans que l'agent ait   interagir avec l'environnement

- Id es cl s :**
- 1 **D ecomposer** le probl me difficile en sous-probl mes plus simples ;
 - 2 Trouver une **solution optimale**   chaque sous-probl me ;
 - 3 **R soudre r cursivement** en utilisant les solutions des sous-probl mes ;

Programmation dynamique (exacte) et processus de d cision s quentielle

- **Pro :** ■ Garantie de trouver une solution optimale exacte ;
- **Con :** ■ Dois avoir acc s au mod le de l'environnement ;
 - Fonctionne sous l'**assumption** que le mod le est **parfait** (c.- -d. sans zone grise ni erreur) ;
 - Victime de la **mal dition de la dimensionnalit ** :
 - Tr s gourmand en m moire et sur le plan computationnel
 - \implies Perd en efficacit  pratique plus l'espace d' tat/action devient massif¹ ;

Pourquoi tudier ces m thodes dans un cours sur le RL alors ?

- Toutes les m thodes de RL **aspirent   des r sultats similaires**   ceux obtenus par DP, **mais en se lib rant des contraintes** du DP² ;
- **Procure une solide fondation** pour comprendre certains m canismes fondamentaux de l'apprentissage par renforcement ;

1. Voir « Reinforcement Learning : An introduction », chapitre 4, section 7 « Efficiency of Dynamic Programming » par Sutton & Barto [8]

2. Quitte   perdre certaines propri t s du DP   l'occasion : garantie de solution optimale, production d'une solution exacte

Programmation dynamique (exacte) et processus de d cision s quentielle

- ! Pro : ■ Garantie de trouver une solution optimale exacte ;
- ! Con : ■ Dois avoir acc s au mod le de l'environnement ;
 - Fonctionne sous l'assumption que le mod le est parfait (c.-d. sans zone grise ni erreur) ;
 - Victime de la mal diction de la dimensionnalit  :
 - Tr s gourmand en m moire et sur le plan computationnel
 - \implies Perd en efficacit  pratique plus l'espace d' tat/action devient massif¹ ;

Pourquoi tudier ces m thodes dans un cours sur le RL alors ?

- Toutes les m thodes de RL aspirent  des r sultats similaires  ceux obtenus par DP, mais en se lib rant des contraintes du DP² ;
- ! Pro : Procure une solide fondation pour comprendre certains m canismes fondamentaux de l'apprentissage par renforcement ;

1. Voir « Reinforcement Learning : An introduction », chapitre 4, section 7 « Efficiency of Dynamic Programming » par Sutton & Barto [8]

2. Quitte  perdre certaines propri t s du DP  l'occasion : garantie de solution optimale, production d'une solution exacte

Terminologie

« Backup Operation »

L'op ration de **mise   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 ex cution d'un « *Backup Operation* » sur chaque *V*-valeurs (ou *Q*-valeurs) d'un espace.

1. Voir l'excellente explication par Andre G. Barto pour plus de d tail dans REINFORCEMENT LEARNING AND DYNAMIC PROGRAMMING, section 2.2 [Barto1995]

Modèle connu et parfait \Rightarrow pas d'apprentissage requis

DP : Itération de Q-valeur

It ration de Q-valeur [Q-value it ration]

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

- Id es cl s :**
- **1 Estimer et stocker** la Q-valeur **de chaque couples**  tat-action de l'univers ;
 - **2 R p ter [infiniment]** en am liorant progressivement l'estimation jusqu'  convergence vers une solution exacte ;

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

- Pro :**
- Intuitif et rapide   impl menter ;
 - Convergence garantie¹ ;

- Con :**
- M moire requise pour la *Q-table*² : $2 \cdot |\mathcal{S} \times \mathcal{A}|$
 - Limiter aux probl mes avec espace d' tat/action de taille modeste ;
Exemple : espace d'action discr te, facteur de branchement limit , ...

1. Voir « Reinforcement Learning : An introduction », chapitre 4 (introduction, 4.4 et 4.6) par Sutton & Barto [8]

2. Version synchrone de l'algorithme : les *backup operations* sont effectu  en parall le

It ration de Q-valeur [Q-value it ration]

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

- Id es cl s :**
- **Estimer et stocker** la Q-valeur **de chaque couples**  tat-action de l'univers ;
 - **R p ter [infiniment]** en am liorant progressivement l'estimation jusqu'  convergence vers une solution exacte ;

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

- Pro :**
- Intuitif et rapide   impl m enter ;
 - Convergence garantie¹ ;

- Con :**
- M moire requise pour la *Q-table*² : $2 \cdot |\mathcal{S} \times \mathcal{A}|$
 - Limiter aux probl mes avec espace d' tat/action de taille modeste ;
Exemple : espace d'action discr te, facteur de branchement limit , ...

1. Voir « Reinforcement Learning : An introduction », chapitre 4 (introduction, 4.4 et 4.6) par Sutton & Barto [8]

2. Version synchrone de l'algorithme : les *backup operations* sont effectu  en parall le

⤳ Algorithme | It ration de Q-valeur

R  p ter avec $i = 1, 2, 3, \dots$ jusqu'  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 | It ration de Q-valeur

On arr te pr matur ment l'algorithme au lieu d'it rer   l'infinie en impl mantant un moyen d' valuer l'estimation et en sp cifiant la marge d'erreur ε vis e.

Cons quence :

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

>_ Pseudocode | It ration de Q-valeur¹ (Version synchrone)

```

Init: foreach  $s \in \mathcal{S}$ ,  $a \in \mathcal{A}(s)$  do  $Q\text{-table}[s, a] \leftarrow 0$ 
Init: threshold hyperparam( $\epsilon$ )  $> 0$                                      « Crit re d'arr t »

repeat                                                                                   « sweep i »
     $\Delta \leftarrow 0$ 
     $Q\text{-table}_{old} \leftarrow Q\text{-table}$ 
    foreach  $s \in \mathcal{S}$  and  $a \in \mathcal{A}(s)$  do
         $Q\text{-table}[s, a] \leftarrow \sum_{s' \in \mathcal{S}} p(s'|s, a) \left[ r(s, a, s') + \gamma \max_{a' \in \mathcal{A}(s')} Q\text{-table}_{old}[s', a'] \right]$  « Backup operation »
         $\Delta \leftarrow \max(\Delta, |Q\text{-table}_{old}[s, a] - Q\text{-table}[s, a]|)$ 
    until  $\Delta < \epsilon$                                                                «  $Q^\pi \xrightarrow{\approx} Q^*$  »

Output:  $\pi^*(s) \approx \arg \max_{a \in \mathcal{A}(s)} Q\text{-table}[s, a]$  « Produis une politique d terministe »

```

1. Pseudocode inspir  de « Reinforcement Learning : An introduction », chapitre 4.4, page 83 « Value Iteration » par Sutton & Barto [8]

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

DP : Itération de politique

It ration de politique [Policy iteration]

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

- Id es cl s :**
- 1 A chaque « sweep », **consid rer une seule action par  tat** (fixer la politique) ;
 - 2 **Estimer et stocker** la V-valeur **de chaque  tat** de l'univers ;
 - 3 **Choisir une meilleure** politique en ´valuant le **futur imm diat** (*one-step look-ahead*) ;
 - 4 **R p ter [infiniment]** en am liorant progressivement l'estimation jusqu'  convergence vers une solution exacte ;

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

- Pro :**
- Intuitif et rapide ´ impl menter ;
 - Convergence garantie¹ et plus rapide² que *It ration de Q-valeur* ;
- Con :**
- M moire requise pour la *V-table* et la π -table³ : $3 \cdot |\mathcal{S}|$
 - Limiter aux probl mes avec espace d' tat/action de taille modeste ;

1. Voir « Reinforcement Learning : An introduction », section 4.0, 4.3 et 4.6 par Sutton & Barto [8]

2. Sous certaines conditions. Voir preuve de convergence.

3. Version synchrone de l'algorithme : les *backup operations* sont effectu s en parall le

It ration de politique [Policy iteration]

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

- Id es cl s :**
- 1 A chaque « sweep », **consid rer une seule action par  tat** (fixer la politique) ;
 - 2 **Estimer et stocker** la V-valeur **de chaque**  tat de l'univers ;
 - 3 **Choisir une meilleure** politique en ´valuant le **futur imm diat** (*one-step look-ahead*) ;
 - 4 **R p ter [infiniment]** en am liorant progressivement l'estimation jusqu'  convergence vers une solution exacte ;

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

- Pro :**
- Intuitif et rapide ´ impl menter ;
 - Convergence garantie¹ et plus rapide² que *It ration de Q-valeur* ;
- Con :**
- M moire requise pour la *V-table* et la π -table³ : $3 \cdot |\mathcal{S}|$
 - Limiter aux probl mes avec espace d' tat/action de taille modeste ;

1. Voir « Reinforcement Learning : An introduction », section 4.0, 4.3 et 4.6 par Sutton & Barto [8]

2. Sous certaines conditions. Voir preuve de convergence.

3. Version synchrone de l'algorithme : les *backup operations* sont effectu s en parall le

Algorithmes | Itération de politique

$\forall s \in S$, fixer arbitrairement la politique courante π_k telle 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

$$V_{i+1}^{\pi}(\mathbf{s}) \leftarrow \sum_{\mathbf{s}' \in S_{i+1}} p(\mathbf{s}'|\mathbf{s}, \pi_k(\mathbf{s})) [r(\mathbf{s}, \pi_k(\mathbf{s}), \mathbf{s}') + \gamma V_i^{\pi}(\mathbf{s}')] \quad \forall \mathbf{s} \in S$$

2 Améliorer la politique courante

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

>_ En pratique | Itération de politique

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

Conséquence :

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

>_ Pseudocode | It ration de politique¹ (Version synchrone)

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

« Crit re d'arr t »

repeat

1 % tape d' valuation de politique.....

repeat $\Delta \leftarrow 0$ $V\text{-table}_{old} \leftarrow V\text{-table}$ **foreach** $s \in \mathcal{S}$ **do** $V\text{-table}[s] \leftarrow \sum_{s' \in \mathcal{S}} p(s'|s, \pi\text{-table}[s]) \left[r(s, a, s') + \gamma V\text{-table}_{old}[s'] \right]$ $\Delta \leftarrow \max(\Delta, |V\text{-table}_{old}[s] - V\text{-table}[s]|)$

« sweep i »

until $\Delta < \epsilon$

2 % tape d'am lioration de politique.....

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

« Backup operation »

until $\pi\text{-IsStable}$ « $V^\pi \xrightarrow{\approx} V^*$ et $\pi \xrightarrow{\approx} \pi^*$ »**Output:** $\pi^*(s) \approx \pi\text{-table}[s]$

« Produis une politique d terministe »

1. Pseudocode inspir  de « Reinforcement Learning : An introduction », chapitre 4.3, page 80 « Policy Iteration » par Sutton & Barto [8]

Comparaison

> Pseudocode | It ration 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 | It ration 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
  1 % ....  t pe d' valuation de politique .....
     $\Delta \leftarrow 0$ 
     $V\text{-table}_{old} \leftarrow V\text{-table}$ 
    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 % ....  t pe d'am lioration 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:  $\pi^*(s) \approx \pi\text{-table}[s]$ 

```

Comparaison

> Pseudocode | It ration 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 | It ration 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
1   ... Etape d' valuation de politique
     $\Delta \leftarrow 0$ 
     $V\text{-table}_{old} \leftarrow V\text{-table}$ 
    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   ... Etape d'amplification 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:  $\pi^*(s) \approx \pi\text{-table}[s]$ 

```

It ration de valeur/politique

Exemple interactif

 valuation de politique dans l'environnement GridWorld

► Exemple interactif



Image : REINFORCEjs, Stanford

■ REINFORCEjs ► Projet ► GitHub

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 Monte-Carlo avec départ exploratoire
- RL par Différence temporelle [TD]
- Q-learning (tabulaire)

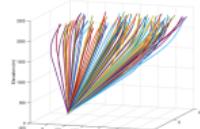
5 Pour aller plus loin

Apprentissage par renforcement (sans mod le)

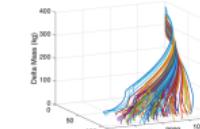
« Supposons qu'on souhaite concevoir une **fus e autonome** avec pour objectif d'atterrir sur la plan te Gliese 581c¹ puis re d閞oller, **sur quelle base** peut-on d閞velopper les syst mes de **navigation, de guidage et de contr le** ?

D'un point de vue appliqu , est-ce que la **garantie d'optimalit ** et la **propri t  d'exactitude** sont des **conditions n cessaires** ? »

Compl ment : Voir [Deep Reinforcement Learning for Six Degree-of-Freedom Planetary Powered Descent and Landing](#), partie I. Introduction, pour une comparaison des m thodes de RL par rapport aux m thodes traditionnelles par Contr le optimal dans le domaine de l'a rospatial. (Plus sp cifiquement, dans le cadre du probl me d'atterrissage autonome   une coordonn  pr cise 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 mod le)

▲ **Probl me :** On n'a **pas de mod le** de l'environnement (Il est inconnu ou imparfait)



On ne peut pas pr dire de comportement dans cet environnement ;

Concr t m nt : Pas de mod le



$p(s'|s, a)$ et $r(s, a, s')$ **inconnu**



Impossible de calculer une solution optimale **exacte** *  priori* ;

Solution : **Interagir** avec l'environnement (r ellement ou en simulateur) afin d'**apprendre** les composantes nécessaires pour trouver une solution optimale :

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

Apprentissage par renforcement (sans modèle)

RL par méthode Monte-Carlo [MC]

M thodes Monte-Carlo et processus de d cision s quentielle

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

Caract ristiques :

- **Interagit** avec l'environnement pour recueillir de l'information ;
- Collecte des ** chantillons** de trajectoires τ **compl tes** ;
- Phase d'apprentissage **hors-ligne** ;
- Apprentissage **sans-mod le** ;
- Pas de « Bootstrap » contrairement aux m thodes de *DP* ;
- N'exploite pas la *propri t  de Markov* ;

Id es cl s : Utiliser la **moyenne empirique** de plusieurs **trajectoires τ compl tes** pour approximer l'esp rance math matische des *quations de Bellman* ;

Pourquoi  a march  ?

M thodes Monte-Carlo et processus de d cision s quentielle

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

Caract ristiques :

- **Interagit** avec l'environnement pour recueillir de l'information ;
- Collecte des ** chantillons** de trajectoires τ **compl tes** ;
- Phase d'apprentissage **hors-ligne** ;
- Apprentissage **sans-mod le** ;
- Pas de « Bootstrap » contrairement aux m thodes de *DP* ;
- N'exploite pas la *propri t  de Markov* ;

Id es cl s : Utiliser la **moyenne empirique** de plusieurs **trajectoires τ compl tes** pour approximer l'esp rance math matische des *quations de Bellman* ;

Pourquoi c  marche ?

La loi des grands nombres

Estim  de Monte-Carlo (formellement)

Soit

- \mathcal{X} un espace d'閑nements possible,
- x une r閏alisation de la variable al閍toire X ,
- f_X la distribution probabilit  **continue** suivie par la v.a. X ,
- $x^{(1)}, x^{(2)}, \dots, x^{(N)}$ des chantillons tir s de f_X
- $g(\cdot)$ une fonction mesurable,

alors pour N suffisamment grand

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

avec \widehat{g}_N la moyenne empirique de l'esp rance math matique $\mathbb{E}[g(X)]$

Estim  de Monte-Carlo (formellement)

Soit

- \mathcal{X} un espace d'閑nements possible,
- x une r閏alisation de la variable al閍toire X ,
- \mathbb{P}_X la distribution probabilit  **discr te** suivie par la v.a. X ,
- $x^{(1)}, x^{(2)}, \dots, x^{(N)}$ des chantillons tir s de \mathbb{P}_X
- $g(\cdot)$ une fonction mesurable,

alors pour N suffisamment grand

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

avec \widehat{g}_N la moyenne empirique de l'esp rance math matique $\mathbb{E}[g(X)]$

Estim  de Monte-Carlo (consid ration pratique)

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

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 r p t e   chaque couple ** tat-action** rencontr  dans une trajectoire chantillonn e.

Estim  de Monte-Carlo (consid ration pratique)

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

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 r p t e   chaque couple ** tat-action** rencontr  dans une trajectoire chantillonn e.

Estim  de Monte-Carlo (consid ration pratique)

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

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 r p t e   chaque couple ** tat-action** rencontr  dans une trajectoire chantillonn e.

Probl me : Plus le nombre dchantillons dpisode k devient grand, plus ce calcule devient couteux en m moire et en temps de calcul. Cest inefficace ▲

Estim  de Monte-Carlo (consid ration pratique)

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

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 r p t e   chaque couple ** tat-action** rencontr  dans une trajectoire chantillonn e.

Probl me : Plus le nombre dchantillons dpisode k devient grand, plus ce calcule devient couteux en m moire et en temps de calcule. **C'est inefficace** 

Solution : Impl mentation incr mentale du calcule de la moyenne

Estim  de Mont  Carlo (consid ration pratique)

Impl mentation incr mental du calcul de la moyenne :

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

Estim  de Mont  Carlo (consid ration pratique)

Impl mentation incr mental du calcul de la moyenne :

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

Estim  de Monte-Carlo (consid ration pratique)

« Update rule », forme g nerale¹

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

➤ En pratique | M thode Monte-Carlo

Impl mentation incr mentale du calcul de la moyenne

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

1. Voir explication dans « Reinforcement Learning : An introduction », chapitre 2, page 31 « Incremental Implementation » par Sutton & Barto [8]

M thodes Monte-Carlo et processus de d cision s quentielle

-  **Pro :**
- Pas besoin de mod le ;
 - Phase d'**apprentissage** ex cut  par « **batch** » \implies est un avantage lorsque les ressources computationnelles sont inf rieur   la capacit    produire de nouvelles trajectoires ;
 - **Aucun biais**¹ ;
 - Les estim s de chaque  tat sont ind pendant l'un de l'autre \implies il est possible de restreindre l' chantillonnage   un sous-ensemble d' tat d'int r t ;
-  **Con :**
- **D pend de la capacit    g n rer beaucoup de trajectoires**
 \implies est un enjeu si la production de trajectoire est longue, couteuse ou risqu  ;
 - Fonctionne seulement sur les **environnements de type ´pisodique** ;
(chaque trajectoire doit terminer)
 - **Beaucoup de variances**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 biais  de $V^\pi(s_t)$

2. car le retour $G_t(\tau)$ d pend d'une grande quantit  de ph nom nes al atoires (probabilit  de s lection d'action, probabilit  de transition, probabilit  de r compense)

M thodes Monte-Carlo (Enjeux pratiques)

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

- 1 le processus d'* valuation de politique* se fait sur une infinit  d' pisodes ;
- 2 chaque couple  tat-action a  t  visit  un nombre infini de fois dans la limite d'une infinit  d' pisodes ;

➤_ En pratique | M thode de Mont  Carlo

Enjeux → contourner ses deux assumptions d'une fa on qui permet de...

- 1 relaxer le pr requis d'avoir   g n rer une infinit  de trajectoires ;
- 2 adresser le **dilemme exploration/exploitation** (assurer l'exploration suffisante de l'espace) ;

Apprentissage par renforcement (sans modèle)

Algorithme Monte-Carlo avec départ exploratoire

Algorithme Monte-Carlo avec d part exploratoire [Monte-Carlo ES]

- Id es cl s :**
- 1 Version *Monte-Carlo* de l'algorithme *It ration de politique* ;
 - 2 **Alterne  pisode par  pisode** entre * valuation et Am lioration de Politique* ;
 - 3   chaque  pisode, consid rer **exclusivement de la premi re visite** fait   un * tat-action* pour les calculs (*First-visit MC*)² ;
 - 4 **Choisir al atoirement le point de d part** de chaque trajectoire chantillonn e ; (d part exploratoire)

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

- Pro :**
- Adress  le *dilemme d'exploration/exploitation* en utilisant l'id e de **d part exploratoire** ;
 - Contourne le pr requis d'infinit  d' pisode en acceptant le fait que chaque  tape d'* valuation de politique* provoque un **plus petit d placement** $Q_k \rightarrow Q_k^\pi$;
- Con :**
- La preuve formelle de convergence de *Monte-Carlo ES* est un probl me ouvert.

2. Il existe galement une version *Every-Visit MC* avec des propri t s th oriques l g rement diff rente

⤳ Algorithme | Monte-Carlo avec *d part exploratoire*

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

R p ter pour chaque trajectoire $T_{k=1,2,3,\dots}$ jusqu'  convergence vers Q^* et π^* :

- ▶ Choisir (le point de d part) $s_0 \in S$ et $a_0 \in \mathcal{A}(s_0)$ al atoirement ;
- ▶ G n rer une trajectoire en suivant $\pi_k(\cdot)$ jusqu'  terminaison ;

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

R p ter pour chaque « timestep » $t = 0, 1, 2, \dots, t, t+1, \dots, T$ de la trajectoire T_k :

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

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

2 ** valuer 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 **Am liorer 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 | Monte-Carlo avec *d part exploratoire et moyenne incr mentale*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( $\epsilon$ ) > 0 ,  $\Delta \leftarrow 0$  ,  $\pi\text{-IsStable} \leftarrow \text{true}$                                 « Crit re d'arr t »

repeat
     $(s_0, a_0) \leftarrow \text{random.select}(s \in \mathcal{S})$ ,  $\text{random.select}(a \in \mathcal{A}(s))$ 
     $T \leftarrow \text{Generate trajectory following } \pi\text{-table}[s]$  from  $(s_0, a_0)$  to terminal state
     $G \leftarrow 0$ 
    foreach  $t \in T \mid t = T-1, T-2, \dots, 2, 1, 0$  do
         $G \leftarrow \gamma G + r_{t+1}$ 
        if  $(s_t, a_t)$  is not in  $T[0, t-1]$  then it's first-visit
             $Q\text{-table}_{old} \leftarrow Q\text{-table}$  ,  $a_{old} \leftarrow \pi\text{-table}[s_t]$                                 « pour la v rification de stabilit  »
        1
        % ....  tape d' valuation 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
        % ....  tape d'am lioration 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
        % ....  tape de v rification de stabilit  ...
         $\Delta \leftarrow \max(\Delta, |Q\text{-table}_{old}[s_t, a_t] - Q\text{-table}[s_t, a_t]|)$ 
        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 d terministe »

```

1. Pseudocode inspir  de « Reinforcement Learning : An introduction », chapitre 5.3, page 99 « Monte Carlo Control » par Sutton & Barto [8]

Apprentissage par renforcement (sans modèle)

RL par Différence temporelle [TD]

M thodes par Diff rence Temporelle

« Try something and learn from your mistakes as you go. »

Caract ristiques :

- **Int agit** avec l'environnement pour recueillir de l'information ;
- Collecte des ** chantillons** de trajectoire τ **partiels** ;
- « Bootstrap » comme avec les m thodes de *DP* ;
- Phase d'apprentissage **EN-ligne** ;
- Apprentissage **sans-mod le** ;
- Exploite la *propri t  de Markov* ;

Id es cl s : Apprendre **au fur et   mesure**, en se basant sur des ** chantillons d'interaction** avec l'environnement **et** sur l'**estimation courante** (« Bootstrap ») ;

M thodes par Diff rence Temporelle (consid ration pratique)

« Update rule », forme g n rale (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 » pour les m thodes par Diff rence Temporelle

Id es cl s : Donner plus de poids aux exp riences r centes ;

Comment ? Calculer la moyenne pond r e des r compenses pass es³

Exemple (Algorithme SARSA) :

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

avec le « learning rate » $\alpha \in]0, 1]$, l chantillon $r^{(k)}$ et Q_k^π un estimer du vrai Q^π .

3. Aussi appell  « Exponential recency-weighted average ». Voir [Compl ment th orique : D rivation de la moyenne pond r e des r compenses pass es](#)

M thodes par Diff rence Temporelle (consid ration pratique)

« Update rule », forme g n rale (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 » pour les m thodes par Diff rence Temporelle

Id es cl s : Donner plus de poids aux exp riences r centes ;

Comment ? Calculer la moyenne pond r e des r compenses pass es³

Exemple (Algorithme SARSA) :

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

avec le « learning rate » $\alpha \in]0, 1]$, l chantillon $r^{(k)}$ et Q_k^π un estimer du vrai Q^π .

3. Aussi appell  « Exponential recency-weighted average ». Voir [Compl ment th orique : D rivation de la moyenne pond r e des r compenses pass es](#)

M thodes par Diff rence Temporelle (consid ration pratique)

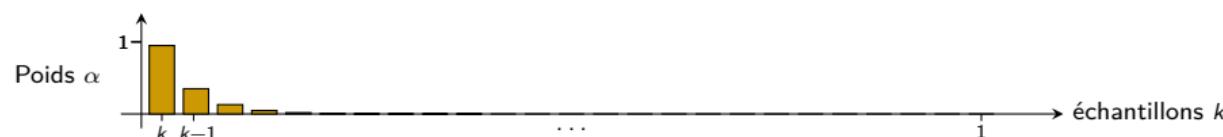
$$Q_{k+1}^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) + \alpha \left[r^{(k)}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) + \gamma Q_k^{\pi}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Exemple avec $\alpha = 0.1$: « Lointain pass , pass  ressent, pr sent ... c'est tout aussi important ! »



$$Q_6 = 0.1 \cdot (r^{(5)} + \gamma Q'_5) + 0.09 \cdot (r^{(4)} + \gamma Q'_4) + 0.081 \cdot (r^{(3)} + \gamma Q'_3) + 0.07289 \cdot (r^{(2)} + \gamma Q'_2) + 0.0656 \cdot (r^{(1)} + \gamma Q'_1) + 0.59045 \cdot Q_1$$

Exemple avec $\alpha = 0.95$: « C'est aujourd'hui qui est important parce qu'avant ... j'étais dans le champ ! »



$$Q_6 = 0.95 \cdot (r^{(5)} + \gamma Q'_5) + 0.0475 \cdot (r^{(4)} + \gamma Q'_4) + 0.00235 \cdot (r^{(3)} + \gamma Q'_3) + 0.0001 \cdot (r^{(2)} + \gamma Q'_2) + 0.0 \cdot (r^{(1)} + \gamma Q'_1) + 0.0 \cdot Q_1$$

Méthodes par Différence Temporelle

- 👍 Pro :
- Pas besoin de modèle ;
 - Peu apprendre **au fur et à mesure** \Rightarrow est un avantage lorsque les trajectoires sont longues à exécuter ;
 - Peu apprendre **de trajectoire incomplète** \Rightarrow est un avantage lorsqu'on n'a pas de garantie que les trajectoires terminent à coup sûr ;
 - **Peu de variances**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 biaisée du vrai $Q^\pi(s_t, a_t)$

Apprentissage par renforcement (sans modèle)

Q-learning (tabulaire)

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

- Id es cl s :**
- C'est une **approximation stochastique** de l'algorithme **It ration de Q-valeur** ;
 - Apprentissage **OFF-policy** \Rightarrow la politique π qui est apprise est ind pendante de la politique μ suivie 脿 l' chantillonnage ;

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

- Pro :**
- Converge vers Q^* m me si l'algorithme agit de fa on sous-optimale 脿 l' chantillonnage
 \Rightarrow l'algorithme converge 脊ventuellement, **peu importe le choix d'actions** ;
- Con :**
- La convergence vers Q^* d pend d'une **exploration suffisante** de l'espace $\mathcal{S} \times \mathcal{A}$
 \Rightarrow une strat gie exploratoire ad quate compte tenu de l'environnement est d terminante ;

⤳ Algorithme | Q-Learning (tabulaire)

Soit $\mu(\cdot|s_t)$, une politique exploratoire de votre choix, (ex. : al atoirement, « greedy », « ϵ -greedy », . . .) la politique cible $\pi(\cdot|s)$ et le « learning rate » $\alpha \in]0, 1]$

R p ter pour chaque trajectoire $T_{k=1,2,3,\dots}$ jusqu'  convergence vers π^* :

R p ter pour chaque « timestep » $t = 0, 1, 2, \dots, t, t+1, \dots, T$ de la trajectoire T_k :

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

1 $r_{t+1}^{(k)}, s_{t+1} \leftarrow$ ex cuter $a_t \sim \mu(\cdot|s_t)$ dans l'environnement et observer la r action ;

2 $\pi(\cdot|s_{t+1}) \leftarrow \arg \max_{a' \in \mathcal{A}(s_{t+1})} Q(s_{t+1}, a');$

3 $Q_{k+1}^\pi(s_t, a_t) \leftarrow Q_k^\pi(s_t, a_t) + \alpha \left[r_{t+1}^{(k)} + \gamma Q_k^\pi(s_{t+1}, \pi(\cdot|s_{t+1})) - Q_k^\pi(s_t, a_t) \right]$

➤ En pratique | Q-Learning (tabulaire)

On fait d croire tranquillement le « learning rate » α en suivant une s quence telle que

$$\{ \alpha_k \mid (\forall k \in [1, 2, \dots] \mid 0 < \alpha_{k+1} < \alpha_k \leq 1) \}$$

⤳ Algorithme | Q-Learning (tabulaire)

Soit $\mu(\cdot|s_t)$, une politique exploratoire de votre choix, (ex. : al atoirement, « greedy », « ϵ -greedy », . . .) la politique cible $\pi(\cdot|s)$ et le « learning rate » $\alpha \in]0, 1]$

R p ter pour chaque trajectoire $T_{k=1,2,3,\dots}$ jusqu'  convergence vers π^* :

R p ter pour chaque « timestep » $t = 0, 1, 2, \dots, t, t+1, \dots, T$ de la trajectoire T_k :

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

1 $r_{t+1}^{(k)}, s_{t+1} \leftarrow$ ex cuter $a_t \sim \mu(\cdot|s_t)$ dans l'environnement et observer la r action ;

2 $\pi(\cdot|s_{t+1}) \leftarrow \arg \max_{a' \in \mathcal{A}(s_{t+1})} Q(s_{t+1}, a');$

3 $Q_{k+1}^\pi(s_t, a_t) \leftarrow Q_k^\pi(s_t, a_t) + \alpha \left[r_{t+1}^{(k)} + \gamma Q_k^\pi(s_{t+1}, \pi(\cdot|s_{t+1})) - Q_k^\pi(s_t, a_t) \right]$

➤ En pratique | Q-Learning (tabulaire)

On fait d croitre tranquillement le « learning rate » α en suivant une s quence telle que

$$\{ \alpha_k \mid (\forall k \in [1, 2, \dots] \mid 0 < \alpha_{k+1} < \alpha_k \leq 1) \}$$

⤳ Algorithme | Q-Learning (tabulaire)

Soit $\mu(\cdot|s_t)$, une politique exploratoire de votre choix, (ex. : al atoirement, « greedy », « ϵ -greedy », . . .) la politique cible $\pi(\cdot|s)$ et le « learning rate » $\alpha \in]0, 1]$

R p ter pour chaque trajectoire $T_{k=1,2,3,\dots}$ jusqu'  convergence vers π^* :

R p ter pour chaque « timestep » $t = 0, 1, 2, \dots, t, t+1, \dots, T$ de la trajectoire T_k :

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

1 $r_{t+1}^{(k)}, s_{t+1}$ \longleftarrow ex cuter $a_t \sim \mu(\cdot|s_t)$ dans l'environnement et observer la r action ;

2 $\pi(\cdot|s_{t+1})$ \longleftarrow $\arg \max_{a' \in \mathcal{A}(s_{t+1})} Q(s_{t+1}, a');$

3 $Q_{k+1}^\pi(s_t, a_t)$ \longleftarrow $Q_k^\pi(s_t, a_t) + \alpha \left[r_{t+1}^{(k)} + \gamma Q_k^\pi(s_{t+1}, \pi(\cdot|s_{t+1})) - Q_k^\pi(s_t, a_t) \right]$

➤ En pratique | Q-Learning (tabulaire)

On fait d croire tranquillement le « learning rate » α en suivant une s quence telle que

$$\{ \alpha_k \mid (\forall k \in [1, 2, \dots] \mid 0 < \alpha_{k+1} < \alpha_k \leq 1) \}$$

⤳ Algorithme | Q-Learning (tabulaire)

Soit $\mu(\cdot|s_t)$, une politique exploratoire de votre choix, (ex. : al atoirement, « greedy », « ϵ -greedy », . . .) la **politique cible** $\pi(\cdot|s)$ et le « learning rate » $\alpha \in]0, 1]$

R p ter pour chaque trajectoire $T_{k=1,2,3,\dots}$ **jusqu'  convergence** vers π^* :

R p ter pour chaque « timestep » $t = 0, 1, 2, \dots, t, t+1, \dots, T$ de la trajectoire T_k :

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

1 $r_{t+1}^{(k)}, s_{t+1} \leftarrow$ ex cuter $a_t \sim \mu(\cdot|s_t)$ dans l'environnement et observer la r action ;

2 $\pi(\cdot|s_{t+1}) \leftarrow \arg \max_{a' \in \mathcal{A}(s_{t+1})} Q(s_{t+1}, a');$

3 $Q_{k+1}^\pi(s_t, a_t) \leftarrow Q_k^\pi(s_t, a_t) + \alpha \left[r_{t+1}^{(k)} + \gamma Q_k^\pi(s_{t+1}, \pi(\cdot|s_{t+1})) - Q_k^\pi(s_t, a_t) \right]$

➤ En pratique | Q-Learning (tabulaire)

On fait **d croitre tranquillement** le « learning rate » α en suivant une s quence telle que

$$\{ \alpha_k \mid (\forall k \in [1, 2, \dots] \mid 0 < \alpha_{k+1} < \alpha_k \leq 1) \}$$

Algorithmme | Q-Learning (tabulaire)

Soit $\mu(\cdot|s_t)$, une politique exploratoire de votre choix, (ex. : aléatoirement, « greedy », « ϵ -greedy », ...) la politique cible $\pi(\cdot|s)$ et le « learning rate » $\alpha \in [0, 1]$

Répéter pour chaque trajectoire $T_{k=1,2,3,\dots}$ jusqu'à convergence vers π^*

Répéter pour chaque « timestep » $t = 0, 1, 2, \dots, t, t+1, \dots, T$ de la trajectoire T_k :

$$\mathcal{T}_k \quad \equiv \quad s_{0,1}, a_{0,1}, r_1, s_{1,1}, a_{1,1}, r_2, \dots, s_{t,1}, a_{t,1}, r_{t+1}, \dots, s_{T-1,1}, a_{T-1,1}, r_T$$

- 1 $r_{t+1}^{(k)}, \mathbf{s}_{t+1} \leftarrow$ exécuter $\mathbf{a}_t \sim \mu(\cdot | \mathbf{s}_t)$ dans l'environnement et observer la réaction ;
 - 2 $\pi(\cdot | \mathbf{s}_{t+1}) \leftarrow \arg \max_{\mathbf{a}' \in \mathcal{A}(\mathbf{s}_{t+1})} Q(\mathbf{s}_{t+1}, \mathbf{a}');$ (simplification)
 - 3 $Q_{k+1}^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) + \alpha \left[r_{t+1}^{(k)} + \gamma \max_{\mathbf{a}' \in \mathcal{A}(\mathbf{s}_{t+1})} Q_k^{\pi}(\mathbf{s}_{t+1}, \mathbf{a}') - Q_k^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \right]$

>_ En pratique | Q-Learning (tabulaire)

On fait **décroître tranquillement** le « learning rate » α en suivant une séquence telle que

$$\left\{ \alpha_k \mid (\forall k \in [1, 2, \dots]) \mid 0 < \alpha_{k+1} < \alpha_k \leq 1 \right\}$$

Q-learning

Exemple interactif

Q-learning dans l'environnement GridWorld

Vous pouvez aussi changer l'algorithme pour SARSA dans les param tres.

► Exemple interactif

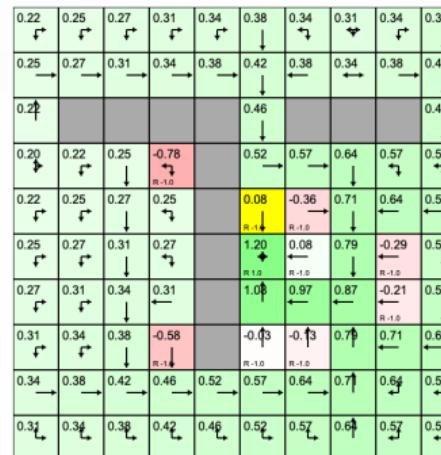


Image : REINFORCEjs, Stanford

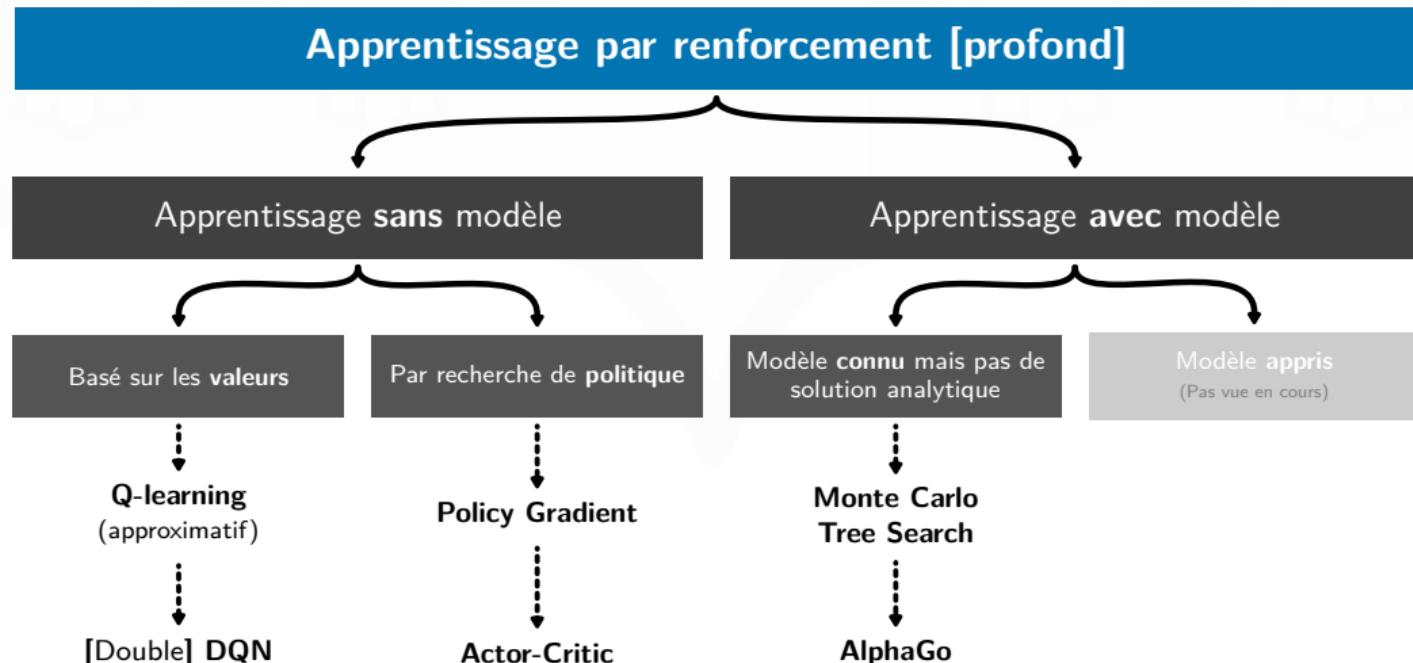
■ REINFORCEjs

► Projet

► GitHub

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 \Rightarrow pas d'apprentissage requis

4 Apprentissage par renforcement (sans modèle)

5 Pour aller plus loin

- Complément théorique
- Ressource théorique
- Références

Pour aller plus loin

Complément théorique

M thode par Diff rence Temporelle

D rivation de la moyenne pond r e des r compenses pass es :

$$\begin{aligned}
 Q_{k+1}(s_t, a_t) &= Q_k(s_t, a_t) + \frac{1}{k} \left[G_t^{(k)}(\tau) - Q_k(s_t, a_t) \right] && \left\langle \text{ avec } \alpha = \frac{1}{k} \right\rangle \\
 &= Q_k(s_t, a_t) + \alpha \left[G_t^{(k)}(\tau) - Q_k(s_t, a_t) \right] && \left\langle \text{ avec } G_t^{(k)}(\tau) \approx r^{(k)}(s_t, a_t, s_{t+1}) + Q_k(s_{t+1}, a_{t+1}) \right\rangle \\
 &= Q_k(s_t, a_t) + \alpha \left[r^{(k)}(s_t, a_t, s_{t+1}) + Q_k(s_{t+1}, a_{t+1}) - Q_k(s_t, a_t) \right] \\
 &= \alpha \left(r^{(k)}(s_t, a_t, s_{t+1}) + Q_k(s_{t+1}, a_{t+1}) \right) + (1 - \alpha)Q_k(s_t, a_t) && \left\langle \text{ avec } r^{(k)} = r^{(k)}(s_t, a_t, s_{t+1}), Q'_k = Q_k(s_{t+1}, a_{t+1}), Q_k = Q_k(s_t, a_t) \right\rangle \\
 &= \alpha \left(r^{(k)} + Q'_k \right) + (1 - \alpha)Q_k \\
 &= \alpha \left(r^{(k)} + Q'_k \right) + (1 - \alpha) \left[\alpha \left(r^{(k-1)} + Q'_{k-1} \right) + (1 - \alpha)Q_{k-1} \right] \\
 &= \alpha \left(r^{(k)} + Q'_k \right) + (1 - \alpha)\alpha \left(r^{(k-1)} + Q'_{k-1} \right) + (1 - \alpha)^2 Q_{k-1} \\
 &= \alpha \left(r^{(k)} + Q'_k \right) + (1 - \alpha)\alpha \left(r^{(k-1)} + Q'_{k-1} \right) + (1 - \alpha)^2 \left(r^{(k-2)} + Q'_{k-2} \right) + \\
 &\quad \cdots + (1 - \alpha)^{k-1} \alpha \left(r^{(1)} + Q'_1 \right) + (1 - \alpha)^k Q_1 && \left\langle \text{ avec } Q_1 \text{ l'estim  initial de } Q^\pi \right\rangle \\
 &= (1 - \alpha)^k Q_1 + \sum_{i=1}^k \alpha(1 - \alpha)^{k-i} \left(r^{(i)} + Q'_i \right)
 \end{aligned}$$

Pour aller plus loin

Ressource théorique

Programmation dynamique

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

Abstract : ... oriented towards modelling, 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 [8]
par Sutton & Barto

Algorithme : *『Evaluation de politique, Am lioration de politique, It ration de V-valeur, It ration de politique et It ration de politique g n ralis  (GPI)』;*

Diff rence temporel [Temporal Difference (TD)]

- « The Paths Perspective on Value Learning », Distill, 2019 [greydanus2019the] [▶ 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

Références

References I

1. BERSETH, G., PENG, X. B. & van de PANNE, M. Terrain RL Simulator. 1-10. arXiv : 1804.06424. <http://arxiv.org/abs/1804.06424> (2018).
2. PENG, X. B., BERSETH, G. & VAN DE PANNE, M. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics* **35**, 1-15. ISSN : 15577368 (2016).
3. MNIH, V. *et al.* Playing Atari with Deep Reinforcement Learning. 1-9. arXiv : 1312.5602. <http://arxiv.org/abs/1312.5602> (2013).
4. MNIH, V. *et al.* Human-level control through deep reinforcement learning. *Nature* **518**, 529-533. ISSN : 14764687. <https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassabis15NatureControlDeepRL.pdf> (2015).
5. VINYALS, O. *et al.* StarCraft II: A New Challenge for Reinforcement Learning. arXiv : 1708.04782. <http://arxiv.org/abs/1708.04782> (2017).
6. ABBEEL, P., COATES, A., QUIGLEY, M. & NG, A. Y. An Application of Reinforcement Learning to Aerobatic Helicopter Flight. *Advances in Neural Information Processing Systems* (2006).

References II

7. MAO, H., ALIZADEH, M., MENACHE, I. & KANDULA, S. Resource management with deep reinforcement learning. *HotNets 2016 - Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, 50-56 (2016).
8. SUTTON, R. S. & BARTO, A. G. *Reinforcement learning: An introduction*. 2^e ´ed. ( d. MIT PRESS) ISBN : 978-0262039246.
<http://incompleteideas.net/book/RLbook2018.pdf> (Cambridge, MA, 2018).
9. SCHWARZ, J. *et al.* Progress & compress: A scalable framework for continual learning. *35th International Conference on Machine Learning, ICML 2018* **10**, 7199-7208. arXiv : 1805.06370. <https://deepmind.com/research/publications/progress-compress-scalable-framework-continual-learning> (2018).
10. WENG, L. *Meta Reinforcement Learning (Lil'Log)*. 2019.
<http://lilianweng.github.io/lil-log/2019/06/23/meta-reinforcement-learning.html>.