

A2DI: Apprentissage de politiques par épisodes

John Klein

Lille1 Université - CRIStAL UMR CNRS 9189





UFR IEEA
Informatique, Electronique
Electrotechnique, Informatique

- d'après un document d'Alessandro Lazaric -

Visitez sa [homepage](#)

[Home](#) [Publications](#) [Teaching](#) [Activities](#) [Projects](#)

Alessandro Lazaric



Alessandro Lazaric
Junior Researcher
SequeL Team

Welcome to my site

I received my PhD from the Electronic and Informatics Department of Politecnico di Milano, under the supervision of [Andrea Bonarini](#) and [Marcello Restelli](#).

I'm currently a Junior Researcher (CR1) at INRIA Lille - Nord Europe in the SequeL team led by [Philippe Preux](#) and [Rémi Munos](#).

You can find my (almost) updated CV [here](#).

My main research topics are:

- *Reinforcement Learning*
- *Transfer Learning*
- *Multi-arm Bandit*
- *Online Learning*

I also keep on eye on:

- *Multiagent Learning*
- *Game Theory*
- *Mechanism Design*

Apprentissage par renforcement :

- MDP est un modèle puissant pour représenter un mécanisme d'apprentissage par renforcement.
- Une politique peut être évaluée à l'aide de sa fonction de valeur.
- Les fonctions de valeur obéissent aux équations de Bellman.
- La programmation dynamique permet de calculer une politique optimale.

Apprentissage par renforcement :

- MDP est un modèle puissant pour représenter un mécanisme d'apprentissage par renforcement.
- Une politique peut être évaluée à l'aide de sa fonction de valeur.
- Les fonctions de valeur obéissent aux équations de Bellman.
- La programmation dynamique permet de calculer une politique optimale.

Apprentissage par renforcement :

- MDP est un modèle puissant pour représenter un mécanisme d'apprentissage par renforcement.
- Une politique peut être évaluée à l'aide de sa fonction de valeur.
- Les fonctions de valeur obéissent aux équations de Bellman.
- La programmation dynamique permet de calculer une politique optimale.

Apprentissage par renforcement :

- MDP est un modèle puissant pour représenter un mécanisme d'apprentissage par renforcement.
- Une politique peut être évaluée à l'aide de sa fonction de valeur.
- Les fonctions de valeur obéissent aux équations de Bellman.
- La programmation dynamique permet de calculer une politique optimale.

Problème : la prog. dynamique requiert une expression explicite des probabilités de transition et de la fonction de récompense.

Apprentissage par renforcement :

- MDP est un modèle puissant pour représenter un mécanisme d'apprentissage par renforcement.
- Une politique peut être évaluée à l'aide de sa fonction de valeur.
- Les fonctions de valeur obéissent aux équations de Bellman.
- La programmation dynamique permet de calculer une politique optimale.

Problème : la prog. dynamique requiert une expression explicite des probabilités de transition et de la fonction de récompense.

→ Dans ce chapitre, nous allons étudier des solutions pour effectuer un apprentissage par renforcement sans avoir recours à ces éléments!

Plan du chapitre

- 1 Apprentissage par épisodes
- 2 Différences temporelles
- 3 Q-learning
- 4 Conclusions

Idée de base :

- *Apprendre à l'aide d'un modèle* ré-utilisable à souhait : soit *le simulateur f* (boîte noire) de l'environnement. Pour tout (x, a) , il fournit

$$f(x, a) = \{y, r\} \text{ with } y \sim p(\cdot|x, a), r = r(x, a).$$

- *Apprentissage par épisodes* : grâce à f , on génère plusieurs *trajectoires* débutant en l'état x et s'achevant quand une condition d'*arrêt* est atteinte :

$$(x_0^{(i)} = x, x_1^{(i)}, \dots, x_{T_i}^{(i)})_{i=1}^n.$$

- *Apprentissage incrémental* : à chaque temps t , l'agent est à l'état x_t , effectue l'action a_t , observe une transition vers l'état x_{t+1} , et reçoit une récompense r_t . On *suppose* que $x_{t+1} \sim p(\cdot|x_t, a_t)$ et $r_t = r(x_t, a_t)$ (hyp. MDP).

Idée de base :

- *Apprendre à l'aide d'un modèle* ré-utilisable à souhait : soit *le simulateur f* (boîte noire) de l'environnement. Pour tout (x, a) , il fournit

$$f(x, a) = \{y, r\} \text{ with } y \sim p(\cdot | x, a), r = r(x, a).$$

- *Apprentissage par épisodes* : grâce à f , on génère plusieurs *trajectoires* débutant en l'état x et s'achevant quand une condition d'*arrêt* est atteinte :

$$(x_0^{(i)} = x, x_1^{(i)}, \dots, x_{T_i}^{(i)})_{i=1}^n.$$

- *Apprentissage incrémental* : à chaque temps t , l'agent est à l'état x_t , effectue l'action a_t , observe une transition vers l'état x_{t+1} , et reçoit une récompense r_t . On *suppose* que $x_{t+1} \sim p(\cdot | x_t, a_t)$ et $r_t = r(x_t, a_t)$ (hyp. MDP).

Idée de base :

- *Apprendre à l'aide d'un modèle* ré-utilisable à souhait : soit *le simulateur f* (boîte noire) de l'environnement. Pour tout (x, a) , il fournit

$$f(x, a) = \{y, r\} \text{ with } y \sim p(\cdot | x, a), r = r(x, a).$$

- *Apprentissage par épisodes* : grâce à f , on génère plusieurs *trajectoires* débutant en l'état x et s'achevant quand une condition d'*arrêt* est atteinte :

$$(x_0^{(i)} = x, x_1^{(i)}, \dots, x_{T_i}^{(i)})_{i=1}^n.$$

- *Apprentissage incrémental* : à chaque temps t , l'agent est à l'état x_t , effectue l'action a_t , observe une transition vers l'état x_{t+1} , et reçoit une récompense r_t . On *suppose* que $x_{t+1} \sim p(\cdot | x_t, a_t)$ et $r_t = r(x_t, a_t)$ (hyp. MDP).

Rq : on se concentre sur

un problème à horizon infinie (sans affaiblissement) et avec états terminaux. Pour une politique donnée π , on cherche d'abord à déterminer sa fonction de valeur

$$V^{\pi}(x) = \mathbb{E} \left[\sum_{t=0}^{T-1} r^{\pi}(x_t) \mid x_0 = x; \pi \right],$$

avec $r^{\pi}(x_t) = r(x_t, \pi(x_t))$ et T la date aléatoire où un état terminal est atteint.

Générateur de trajectoires :

L'agent interagit avec f (protocole RL) selon l'algorithme suivant :

Générer n trajectoires

Initialiser $t \leftarrow 0$ et $x_0 \leftarrow x$ [éventuellement aléatoirement].

for i de 1 à n **do**

while x_t pas terminal [générer 1 traj.] **do**

 Effectuer l'action a_t .

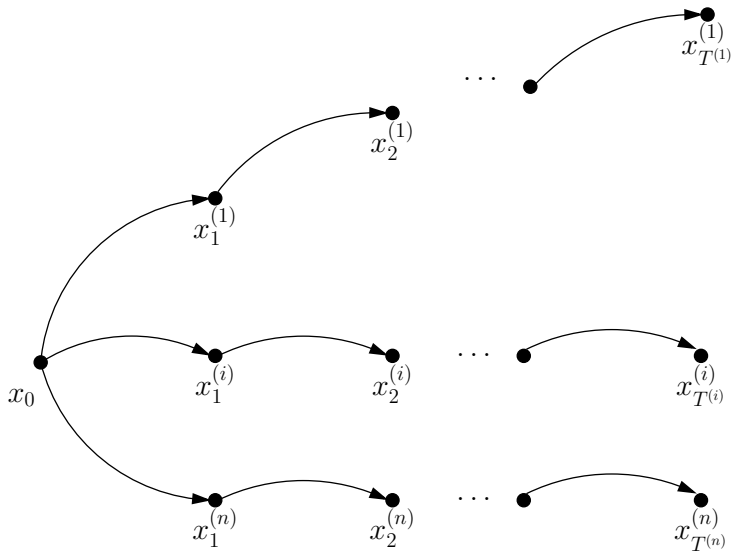
 Appeler f et observer x_{t+1} et r_t .

$t \leftarrow t + 1$

end while

end for

Générateur de trajectoires : résultats



Générateur de trajectoires *pour évaluer* π :

L'agent interagit avec f (protocole RL) selon l'algorithme suivant :

Générer n trajectoires

Initialiser $t \leftarrow 0$ et $x_0 \leftarrow x$ [éventuellement aléatoirement].

for i de 1 à n **do**

while x_t pas terminal [générer 1 traj.] **do**

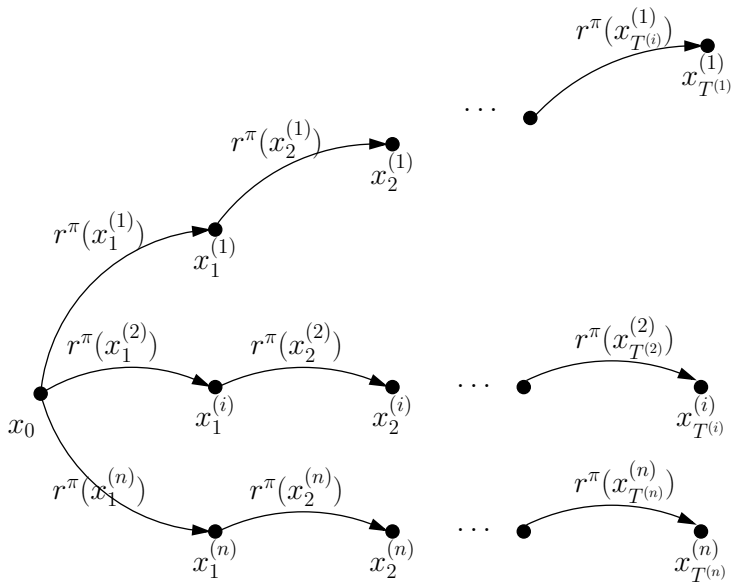
 Effectuer l'action $a_t = \pi(x_t)$.

 Appeler f et observer x_{t+1} et $r_t = r^\pi(x_t)$.

$t \leftarrow t + 1$

end while

end for

Générateur de trajectoires *pour évaluer* π : résultats

Que faire de ces n trajectoires ? → du Monte-Carlo !



Pause stats !

Monte Carlo en 2 mots :

- Il s'agit d'un outil permettant de calculer une quantité $I(X)$ où X est une variable aléatoire.
- Souvent $I(X)$ ne peut être calculée (loi de X mal connue, complexité déraisonnable, etc.).
- Si on peut tirer n échantillons indépendants $x^{(i)} \sim p_X$ alors

$$I(X) \approx \frac{1}{n} \sum_{i=1}^n I(x^{(i)}).$$

- Monte Carlo repose à 99% sur la loi des grands nombres.

Que faire de ces n trajectoires ? → du Monte-Carlo !



Pause stats !

Monte Carlo en 2 mots :

- Il s'agit d'un outil permettant de calculer une quantité $I(X)$ où X est une variable aléatoire.
- Souvent $I(X)$ ne peut être calculée (loi de X mal connue, complexité déraisonnable, etc.).
- Si on peut tirer n échantillons indépendants $x^{(i)} \sim p_X$ alors

$$I(X) \approx \frac{1}{n} \sum_{i=1}^n I(x^{(i)}).$$

- Monte Carlo repose à 99% sur la loi des grands nombres.

Que faire de ces n trajectoires ? → du Monte-Carlo !



Pause stats !

Monte Carlo en 2 mots :

- Il s'agit d'un outil permettant de calculer une quantité $I(X)$ où X est une variable aléatoire.
- Souvent $I(X)$ ne peut être calculée (loi de X mal connue, complexité déraisonnable, etc.).
- Si on peut tirer n échantillons indépendants $x^{(i)} \sim p_X$ alors

$$I(X) \approx \frac{1}{n} \sum_{i=1}^n I(x^{(i)}).$$

- Monte Carlo repose à 99% sur la loi des grands nombres.

Que faire de ces n trajectoires ? \rightarrow du Monte-Carlo !



Pause stats !

Monte Carlo en 2 mots :

- Il s'agit d'un outil permettant de calculer une quantité $I(X)$ où X est une variable aléatoire.
- Souvent $I(X)$ ne peut être calculée (loi de X mal connue, complexité déraisonnable, etc.).
- Si on peut tirer n échantillons indépendants $x^{(i)} \sim p_X$ alors

$$I(X) \approx \frac{1}{n} \sum_{i=1}^n I(x^{(i)}).$$

- Monte Carlo repose à 99% sur la loi des grands nombres.

Que faire de ces n trajectoires ? → du Monte-Carlo !

Monte Carlo dans notre cas :

- La quantité à estimer est $V^\pi(x_0)$.
- Récompense cumulée sur la trajectoire n° i :

$$\hat{R}_i(x_0) = \sum_{t=0}^{T(i)} \gamma^t r^\pi(x_t^{(i)})$$

- Fonction de valeur estimée :

$$V^\pi(x_0) \approx \hat{V}_n^\pi(x_0) = \frac{1}{n} \sum_{i=1}^n \hat{R}_i(x_0)$$

Que faire de ces n trajectoires ? → du Monte-Carlo !

Monte Carlo dans notre cas :

- La quantité à estimer est $V^\pi(x_0)$.
- Récompense cumulée sur la trajectoire n° i :

$$\hat{R}_i(x_0) = \sum_{t=0}^{T^{(i)}} \gamma^t r^\pi(x_t^{(i)})$$

- Fonction de valeur estimée :

$$V^\pi(x_0) \approx \hat{V}_n^\pi(x_0) = \frac{1}{n} \sum_{i=1}^n \hat{R}_i(x_0)$$

Que faire de ces n trajectoires ? → du Monte-Carlo !

Monte Carlo dans notre cas :

- La quantité à estimer est $V^\pi(x_0)$.
- Récompense cumulée sur la trajectoire n° i :

$$\hat{R}_i(x_0) = \sum_{t=0}^{T^{(i)}} \gamma^t r^\pi(x_t^{(i)})$$

- Fonction de valeur estimée :

$$V^\pi(x_0) \approx \hat{V}_n^\pi(x_0) = \frac{1}{n} \sum_{i=1}^n \hat{R}_i(x_0)$$

Monte-Carlo : propriétés

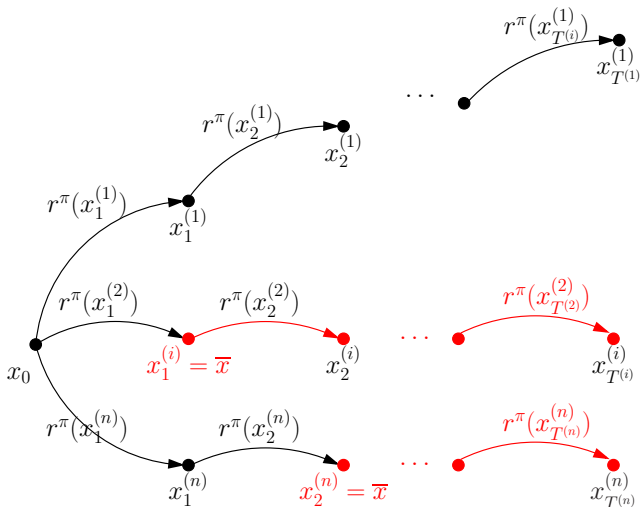
- Chaque récompense cumulée sur une trajectoire est un estimateur **non-biaisé** de $V^\pi(x)$

$$\mathbb{E}[\hat{R}^{(i)}(x_0)] = \mathbb{E}\left[r^\pi(x_0^{(i)}) + \gamma r^\pi(x_1^{(i)}) + \cdots + \gamma^{T^{(i)}} r^\pi(x_{T^{(i)}}^{(i)})\right] = V^\pi(x)$$

- d'où

$$\hat{V}_n^\pi(x_0) \xrightarrow{\text{p.s.}} V^\pi(x_0) \text{ quand } n \longrightarrow \infty.$$

- Des garanties quand $n < \infty$ existent aussi.

Monte-Carlo : *Sous-trajectoires*

Les *sous-trajectoires* débutant en \bar{x} peut être utilisées pour estimer $V^\pi(\bar{x})$!

Monte-Carlo : *Sous-trajectoires* - 2 approches

Toute trajectoire $(x_0, x_1, x_2, \dots, x_T)$ contient aussi la sous-trajectoire $(x_t, x_{t+1}, \dots, x_T)$ dont la récompense cumulée $\hat{R}(x_t) = r^\pi(x_t) + \dots + r^\pi(x_{T-1})$ pourrait être utilisée pour construire un estimateur de $V^\pi(x_t)$.

- *First-visit MC* : Pour chaque état x , on prend en compte seulement la sous-trajectoire quand x est atteint en 1^{er}. *Estimateur non-biaisé, Seulement un échantillon par trajectoire.*
- *Every-visit MC* : Etant donné une trajectoire $(x_0 = x, x_1, x_2, \dots, x_T)$, on identifie les m sous-trajectoires débutant en x et se terminant en x_T , puis on les moyenne pour obtenir un estimé. *Plusieurs échantillons par trajectoire, Estimateur biaisé.*

Monte-Carlo : *Sous-trajectoires* - 2 approches

Toute trajectoire $(x_0, x_1, x_2, \dots, x_T)$ contient aussi la sous-trajectoire $(x_t, x_{t+1}, \dots, x_T)$ dont la récompense cumulée $\bar{R}(x_t) = r^\pi(x_t) + \dots + r^\pi(x_{T-1})$ pourrait être utilisée pour construire un estimateur de $V^\pi(x_t)$.

- *First-visit MC* : Pour chaque état x , on prend en compte seulement la sous-trajectoire quand x est atteint en 1^{er}. **Estimateur non-biaisé, Seulement un échantillon par trajectoire.**
- *Every-visit MC* : Etant donné une trajectoire $(x_0 = x, x_1, x_2, \dots, x_T)$, on identifie les m sous-trajectoires débutant en x et se terminant en x_T , puis on les moyenne pour obtenir un estimé. **Plusieurs échantillons par trajectoire, Estimateur biaisé.**

Monte-Carlo : *Sous-trajectoires* - 2 approches

Toute trajectoire $(x_0, x_1, x_2, \dots, x_T)$ contient aussi la sous-trajectoire $(x_t, x_{t+1}, \dots, x_T)$ dont la récompense cumulée $\bar{R}(x_t) = r^\pi(x_t) + \dots + r^\pi(x_{T-1})$ pourrait être utilisée pour construire un estimateur de $V^\pi(x_t)$.

- *First-visit MC* : Pour chaque état x , on prend en compte seulement la sous-trajectoire quand x est atteint en 1^{er}. *Estimateur non-biaisé, Seulement un échantillon par trajectoire.*
- *Every-visit MC* : Etant donné une trajectoire $(x_0 = x, x_1, x_2, \dots, x_T)$, on identifie les m sous-trajectoires débutant en x et se terminant en x_T , puis on les moyenne pour obtenir un estimé. *Plusieurs échantillons par trajectoire, Estimateur biaisé.*

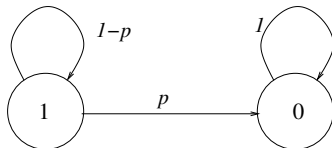
Monte-Carlo : *Sous-trajectoires* - 2 approches

Qui préférer ?

rappel : → *un estimateur biaisé peut être plus efficace (à condition qu'il soit consistant) !*

Monte-Carlo : *Sous-trajectoires* - 2 approches

Exemple : chaîne de Markov à 2 états



La récompense est 1 tant qu'on est dans l'état 1 (elle vaut 0 dans l'état 0 terminal). Toutes les trajectoires sont $(x_0 = 1, x_1 = 1, \dots, x_T = 0)$. D'après les *équations de Bellman*, on a

$$V(1) = 1 + (1 - p)V(1) + 0 \cdot p = \frac{1}{p},$$

puisque $V(0) = 0$.

Monte-Carlo : *Sous-trajectoires* - 2 approches

On mesure la qualité de l'estimation de V au sens de la perte quadratique moyenne ou *mean square error* (MSE).

On obtient alors un résultat bien connu en ML, appelé **décomposition Biais-Variance** :

$$\mathbb{E}[(\hat{V} - V)^2] = \underbrace{(\mathbb{E}[\hat{V}] - V)^2}_{\text{Biais}^2} + \underbrace{\mathbb{E}[(\hat{V} - \mathbb{E}[\hat{V}])^2]}_{\text{Variance}}$$

Monte-Carlo : *Sous-trajectoires* - 2 approches

Exemple : chaîne de Markov à 2 états

First-visit Monte-Carlo. Toutes les trajectoires débutent par l'état 1, donc la récompense cumulée sur une unique trajectoire est exactement T , i.e., $\hat{V} = T$.

On suppose que le temps final T suit une loi *geometrique* d'espérance

$$\mathbb{E}[\hat{V}] = \mathbb{E}[T] = \frac{1}{p} = V^\pi(1) \Rightarrow \textit{estimateur non-biaisé}.$$

Ainsi, la MSE de \hat{V} coïncide avec la variance de T , qui vaut

$$\mathbb{E}\left[\left(T - \frac{1}{p}\right)^2\right] = \frac{1}{p^2} - \frac{1}{p}.$$

Monte-Carlo : *Sous-trajectoires* - 2 approches

Exemple : chaîne de Markov à 2 états

Every-visit Monte-Carlo. Etant donné une trajectoire, nous pouvons construire $T - 1$ sous-trajectoires (nombre de visites de l'état 1), où la $t^{\text{ème}}$ trajectoire produit la récompense cumulée $T - t$.

$$\hat{V} = \frac{1}{T} \sum_{t=0}^{T-1} (T - t) = \frac{1}{T} \sum_{t'=1}^T t' = \frac{T+1}{2}.$$

L'espérance est

$$\mathbb{E}\left[\frac{T+1}{2}\right] = \frac{1+p}{2p} \neq V^{\pi}(1) \Rightarrow \textit{estimateur biaisé}.$$

Monte-Carlo : *Sous-trajectoires* - 2 approches

Soit n *trajectoires indépendantes*, chacune de longueur T_i .

Le nombre total d'échantillons vaut $\sum_{i=1}^n T_i$ et l'estimateur \hat{V}_n est

$$\begin{aligned}\hat{V}_n &= \frac{\sum_{i=1}^n \sum_{t=0}^{T_i-1} (T_i - t)}{\sum_{i=1}^n T_i} = \frac{\sum_{i=1}^n T_i(T_i + 1)}{2 \sum_{i=1}^n T_i} \\ &= \frac{1/n \sum_{i=1}^n T_i(T_i + 1)}{2/n \sum_{i=1}^n T_i} \\ &\xrightarrow{a.s.} \frac{\mathbb{E}[T^2] + \mathbb{E}[T]}{2\mathbb{E}[T]} = \frac{1}{p} = V^\pi(1) \Rightarrow \text{estimateur consistant.}\end{aligned}$$

La MSE de l'estimateur est

$$\mathbb{E}\left[\left(\frac{T+1}{2} - \frac{1}{p}\right)^2\right] = \frac{1}{2p^2} - \frac{3}{4p} + \frac{1}{4} \leq \frac{1}{p^2} - \frac{1}{p}.$$

Monte-Carlo : *Sous-trajectoires* - 2 approches

En général

- *Every-visit MC* : estimateur *biaisé* mais *consistant*.
- *First-visit MC* : estimateur *non-biaisé* avec potentiellement une *plus grosse MSE*.

Rq : quand l'espace d'état est grand, la probabilité de visiter plusieurs fois le même état est faible, alors les performances des 2 méthodes sont semblables.

Monte-Carlo : *Sous-trajectoires* - 2 approches

En général

- *Every-visit MC* : estimateur *biaisé* mais *consistant*.
- *First-visit MC* : estimateur *non-biaisé* avec potentiellement une *plus grosse MSE*.

R_q : quand l'espace d'état est grand, la probabilité de visiter plusieurs fois le même état est faible, alors les performances des 2 méthodes sont semblables.

Plan du chapitre

- 1 Apprentissage par épisodes
- 2 Différences temporelles**
- 3 Q-learning
- 4 Conclusions

Monte-Carlo : Limites

- Le **problème** majeur de l'approche MC est qu'il faut **toutes les trajectoires** pour calculer \hat{V} .
- Ne pourrait-on pas estimer V **au fur et à mesure** qu'on reçoit des trajectoires ?
- Les algorithmes de type **différences temporelles** (TD) répondent à ce besoin.

Monte-Carlo : Limites

- Le **problème** majeur de l'approche MC est qu'il faut **toutes les trajectoires** pour calculer \hat{V} .
- Ne pourrait-on pas estimer V **au fur et à mesure** qu'on reçoit des trajectoires ?
- Les algorithmes de type **différences temporelles** (TD) répondent à ce besoin.

Monte-Carlo : Limites

- Le **problème** majeur de l'approche MC est qu'il faut **toutes les trajectoires** pour calculer \hat{V} .
- Ne pourrait-on pas estimer V **au fur et à mesure** qu'on reçoit des trajectoires ?
- Les algorithmes de type **différences temporelles** (TD) répondent à ce besoin.

Différences Temporelles : TD(1)

TD(1)

Soit $(x_0^{(i)}, \dots, x_{T^{(i)}}^{(i)})$ la $i^{\text{ème}}$ trajectoire échantillonnée et $\hat{R}^{(i)}$ la fonction de récompense cumulée correspondante. Pour tout état $x_t^{(i)}$ ($t < T^{(i)}$) appartenant à cette trajectoire, mettre à jour la fonction de valeur estimée selon

$$\hat{V}(x_t^{(i)}) \leftarrow (1 - \eta(x_t^{(i)})) \hat{V}(x_t^{(i)}) + \eta(x_t^{(i)}) \hat{R}^{(i)}(x_t^{(i)}).$$

Différences Temporelles : TD(1)

TD(1)

Soit $(x_0^{(i)}, \dots, x_{T^{(i)}}^{(i)})$ la $i^{\text{ème}}$ trajectoire échantillonnée et $\hat{R}^{(i)}$ la fonction de récompense cumulée correspondante. Pour tout état $x_t^{(i)}$ ($t < T^{(i)}$) appartenant à cette trajectoire, mettre à jour la fonction de valeur estimée selon

$$\hat{V}(x_t^{(i)}) \leftarrow (1 - \eta(x_t^{(i)})) \hat{V}(x_t^{(i)}) + \eta(x_t^{(i)}) \hat{R}^{(i)}(x_t^{(i)}).$$

→ même idée que SGD !! (même si ici pas de gradient)

Différences Temporelles : TD(1)

- Chaque récompense cumulée est un estimateur **non-biaisé** de $V^\pi(x_t)$

$$\mathbb{E}[\hat{R}^{(i)}(x_t)] = \mathbb{E}\left[r^\pi(x_t^{(i)}) + r^\pi(x_{t+1}^{(i)}) + \cdots + r^\pi(x_{T(i)}^{(i)})\right] = V^\pi(x_t)$$

- Si en plus, les coefficients η vérifient les conditions de Robbins-Monroe pour tout état x

$$\sum_{i=1}^{\infty} \eta(x) = \infty \text{ et } \sum_{i=1}^{\infty} \eta(x)^2 < \infty,$$

- alors

$$\hat{V}(x_t) \xrightarrow{\text{p.s.}} V^\pi(x_t) \text{ quand } n \rightarrow \infty.$$

Différences Temporelles : TD(0)

- Soit l'opérateur de **Bellman bruité** $\hat{\mathcal{T}}^\pi$:

$$\hat{\mathcal{T}}^\pi V(x_t) = r^\pi(x_t) + V(x_{t+1}).$$

- $\hat{\mathcal{T}}^\pi V(x_t)$ est un estimateur **non-biaisé** de $\mathcal{T}^\pi V(x_t)$:

$$\begin{aligned} \mathbb{E} \left[\hat{\mathcal{T}}^\pi V(x_t) | x_t = x \right] &= \mathbb{E} [r^\pi(x_t) + V(x_{t+1}) | x_t = x], \\ &= r^\pi(x_t) + \sum_y p(y|x, \pi(x)) V(y), \\ &= \mathcal{T}^\pi V(x_t). \end{aligned}$$

- Le **bruit** est **borné** :

$$|\hat{\mathcal{T}}^\pi V(x_t) - \mathcal{T}^\pi V(x_t)| \leq \|V\|_\infty.$$

Différences Temporelles : TD(0)

TD(0)

Soit $(x_0^{(i)}, \dots, x_{T^{(i)}}^{(i)})$ la $i^{\text{ème}}$ trajectoire échantillonnée et $\hat{R}^{(i)}$ la fonction de récompense cumulée correspondante. Pour tout état $x_t^{(i)}$ ($t < T^{(i)}$) appartenant à cette trajectoire, mettre à jour la fonction de valeur estimée selon

$$\hat{V}(x_t^{(i)}) \leftarrow (1 - \eta(x_t^{(i)})) \hat{V}(x_t^{(i)}) + \eta(x_t^{(i)}) \hat{r}^\pi \hat{V}(x_t^{(i)}).$$

Différences Temporelles : TD(0)

TD(0)

Soit $(x_0^{(i)}, \dots, x_{T^{(i)}}^{(i)})$ la $i^{\text{ème}}$ trajectoire échantillonnée et $\hat{R}^{(i)}$ la fonction de récompense cumulée correspondante. Pour tout état $x_t^{(i)}$ ($t < T^{(i)}$) appartenant à cette trajectoire, mettre à jour la fonction de valeur estimée selon

$$\hat{V}(x_t^{(i)}) \leftarrow (1 - \eta(x_t^{(i)})) \hat{V}(x_t^{(i)}) + \eta(x_t^{(i)}) \hat{r}^\pi \hat{V}(x_t^{(i)}).$$

→ Toujours la même idée que SGD !

Différences Temporelles : TD(0)

TD(0)

Soit $(x_0^{(i)}, \dots, x_{T^{(i)}}^{(i)})$ la $i^{\text{ème}}$ trajectoire échantillonnée et $\hat{R}^{(i)}$ la fonction de récompense cumulée correspondante. Pour tout état $x_t^{(i)}$ ($t < T^{(i)}$) appartenant à cette trajectoire, mettre à jour la fonction de valeur estimée selon

$$\hat{V}(x_t^{(i)}) \leftarrow (1 - \eta(x_t^{(i)})) \hat{V}(x_t^{(i)}) + \eta(x_t^{(i)}) \hat{r}^\pi \hat{V}(x_t^{(i)}).$$

- Toujours la même idée que SGD !
- Si Robbins-Monroe ok, alors convergence ok !

Différences Temporelles : définition

Définition

A l'itération i , étant donné l'estimateur actuel \hat{V} (obtenu en $i - 1$), on définit la **différence temporelle** pour la paire (x_t, x_{t+1}) par

$$d^i(x_t, x_{t+1}) = r^\pi(x_t) + \hat{V}(x_{t+1}) - \hat{V}(x_t).$$

Différences Temporelles : définition

Définition

A l'itération i , étant donné l'estimateur actuel \hat{V} (obtenu en $i - 1$), on définit la **différence temporelle** pour la paire (x_t, x_{t+1}) par

$$d^i(x_t, x_{t+1}) = r^\pi(x_t) + \hat{V}(x_{t+1}) - \hat{V}(x_t).$$

On note que cette définition évalue la cohérence de l'estimateur pour la transition $x_t \rightarrow x_{t+1}$ vis à vis des **équations de Bellman**.

Différences Temporelles : TD(1) vs TD(0)

- TD(1) tient compte des dépendances à long terme :

$$\hat{V}\left(x_t^{(i)}\right) \leftarrow \hat{V}\left(x_t^{(i)}\right) + \eta\left(x_t^{(i)}\right)\left(d^{(i)}\left(x_t^{(i)}, x_{t+1}^{(i)}\right) + \dots + d^{(i)}\left(x_{T-1}^{(i)}, x_T^{(i)}\right)\right)$$

- TD(0) tient compte des dépendances à court terme :

$$\hat{V}\left(x_t^{(i)}\right) \leftarrow \hat{V}\left(x_t^{(i)}\right) + \eta\left(x_t^{(i)}\right)\left(d^{(i)}\left(x_t^{(i)}, x_{t+1}^{(i)}\right)\right)$$

Différences Temporelles : TD(λ)

L'algorithme TD(λ) offre une solution intermédiaire entre TD(1) et TD(0).

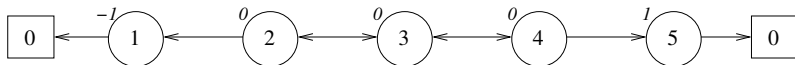
TD(λ)

Soit $(x_0^{(i)}, \dots, x_{T^{(i)}}^{(i)})$ la $i^{\text{ème}}$ trajectoire échantillonnée et $d^{(i)}$ les différences temporelles. Pour tout état $x_t^{(i)}$ ($t < T^{(i)}$) appartenant à cette trajectoire, mettre à jour la fonction de valeur estimée selon

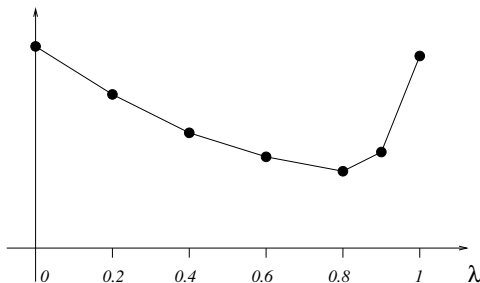
$$\hat{V}(x_t^{(i)}) \leftarrow \hat{V}(x_t^{(i)}) + \eta(x_t^{(i)}) \sum_{s=t}^{T^{(i)}-1} \lambda^{s-t} d^{(i)}(x_s^{(i)}, x_{s+1}^{(i)}).$$

Différences Temporelles : TD(λ)

Exemple : chaîne linéaire



La MSE de \hat{V} par rapport à V^π après $n = 100$ trajectoires :



Plan du chapitre

- 1 Apprentissage par épisodes
- 2 Différences temporelles
- 3 Q-learning**
- 4 Conclusions

Q-learning :

Les algorithmes TD ont encore un **gros problème** :

- Grâce à eux, je calcule incrémentalement \hat{V}^{π_k} .
- Pour faire évoluer ma politique, le choix naturel est la **politique gloutonne** vis à vis de \hat{V}^{π_k} :

$$\pi_{k+1}(x) \in \arg \max_a \left[r(x, a) + \sum_y p(y|x, a) \hat{V}^{\pi_k}(y) \right].$$

Q-learning :

Les algorithmes TD ont encore un **gros problème** :

- Grâce à eux, je calcule incrémentalement \hat{V}^{π_k} .
- Pour faire évoluer ma politique, le choix naturel est la **politique gloutonne** vis à vis de \hat{V}^{π_k} :

$$\pi_{k+1}(x) \in \arg \max_a \left[r(x, a) + \sum_y p(y|x, a) \hat{V}^{\pi_k}(y) \right].$$

... mais les $p(y|x, a)$ sont inconnues !

Q-learning :

Les algorithmes TD ont encore un **gros problème** :

- Grâce à eux, je calcule incrémentalement \hat{V}^{π_k} .
- Pour faire évoluer ma politique, le choix naturel est la **politique gloutonne** vis à vis de \hat{V}^{π_k} :

$$\pi_{k+1}(x) \in \arg \max_a \left[r(x, a) + \sum_y p(y|x, a) \hat{V}^{\pi_k}(y) \right].$$

... mais les $p(y|x, a)$ sont inconnues !

→ On va utiliser les **Q-fonctions**.

Q-learning :

Q-learning

Soit $(x_0^{(i)}, \dots, x_{T^{(i)}}^{(i)})$ la $i^{\text{ème}}$ trajectoire échantillonnée et \hat{Q} notre estimée actuelle de Q . Pour toute paire état-action $(x_t^{(i)}, a_t)$ ($t < T^{(i)}$) appartenant à cette trajectoire, mettre à jour la Q-fonction estimée selon

$$\hat{Q}(x_t^{(i)}, a_t) \leftarrow (1 - \eta(x_t^{(i)}, a_t)) \hat{Q}(x_t^{(i)}, a_t) + \eta(x_t^{(i)}, a_t) \left[r + \max_{b \in A} \hat{Q}(x_{t+1}^{(i)}, b) \right].$$

Q-learning :

Q-learning

Soit $(x_0^{(i)}, \dots, x_{T^{(i)}}^{(i)})$ la $i^{\text{ème}}$ trajectoire échantillonnée et \hat{Q} notre estimée actuelle de Q . Pour toute paire état-action $(x_t^{(i)}, a_t)$ ($t < T^{(i)}$) appartenant à cette trajectoire, mettre à jour la Q-fonction estimée selon

$$\hat{Q}(x_t^{(i)}, a_t) \leftarrow (1 - \eta(x_t^{(i)}, a_t)) \hat{Q}(x_t^{(i)}, a_t) + \eta(x_t^{(i)}, a_t) \left[r + \max_{b \in A} \hat{Q}(x_{t+1}^{(i)}, b) \right].$$

Si Robbins-Monroe ok et que toute paire (x, a) est visitée **infiniment souvent**, alors convergence ok !

Plan du chapitre

- 1 Apprentissage par épisodes
- 2 Différences temporelles
- 3 Q-learning
- 4 Conclusions**

Messages importants du chapitre :

- Les solutions de type Monte Carlo permettent d'apprendre une fonction de valeur sans avoir besoin explicitement des valeurs des probabilités de transition ou de l'équation de la fonction de récompense.
- Les algorithmes de type TD offrent en plus la possibilité d'apprendre la fonction de valeur de manière plus incrémentale.
- Q-learning fait de même et permet d'en déduire la politique gloutonne associée.

Messages importants du chapitre :

- Les solutions de type Monte Carlo permettent d'apprendre une fonction de valeur sans avoir besoin explicitement des valeurs des probabilités de transition ou de l'équation de la fonction de récompense.
- Les algorithmes de type TD offrent en plus la possibilité d'apprendre la fonction de valeur de manière plus incrémentale.
- Q-learning fait de même et permet d'en déduire la politique gloutonne associée.

Messages importants du chapitre :

- Les solutions de type Monte Carlo permettent d'apprendre une fonction de valeur sans avoir besoin explicitement des valeurs des probabilités de transition ou de l'équation de la fonction de récompense.
- Les algorithmes de type TD offrent en plus la possibilité d'apprendre la fonction de valeur de manière plus incrémentale.
- Q-learning fait de même et permet d'en déduire la politique gloutonne associée.