Apprentissage par renforcement

Stéphane Airiau

Université Paris Dauphine

Temporal-difference Learning

Combine des idées de
la programmation dynamique (DP)
avec
les méthodes de Monte Carlo (MC)

Méthodes "Temporal-difference"

- elle apprennent directement avec l'expérience (comme MC)
- elles sont sans modèle : pas besoin de connaître les modèles de transition ou de récompenses (comme MC)
- elles peuvent apprendre d'épisodes incomplets (comme PD)
- elles utilisent des estimations pour mettre à jour son estimation (comme DP)

Méthodes "Temporal-difference"pour la fonction de valeurs

On veut estimer la valeur des états pour une politique fixe π.

- Méthode Monte Carlo "chaque visite"
  - mise à jour à l'aide du véritable gain G_t

$$v(s_t) \leftarrow v(s_t) + \alpha(G_t - v(s_t))$$

  - G_t est accessible à la fin de l'épisode ⇨ peut on éviter cette atttente ?
- Méthode la plus simple : TD(0)

$$v(s_t) \leftarrow v(s_t) + \alpha\,[r_{t+1} + \gamma v(s_{t+1}) - v(s_t)]$$

mise à jour à l'aide de $r_{t+1} + \gamma v(s_{t+1})$

$$\begin{aligned}
\text{Rappel : } v_\pi(s) &= \mathbb{E}_\pi[G_t \mid s_t = s] \\
&= \mathbb{E}_\pi[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s] \\
&= \mathbb{E}_\pi[r_{t+1} + \gamma v_\pi(s_{t+1}) \mid s_t = s]
\end{aligned}$$

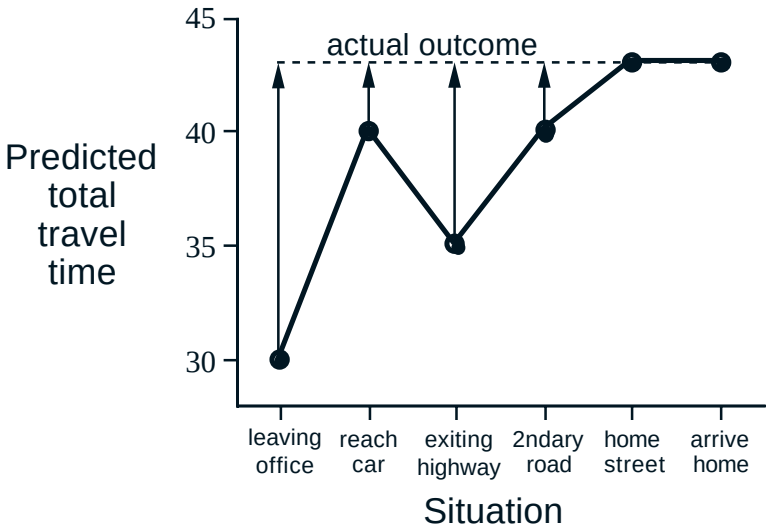
exemple du temps de trajet

Etat	temps écoulé	temps estimé	temps total prédit
départ du bureau	0	30	30
arrivée à la voiture, il pleut	5	35	40
sortie de l'autoroute	20	15	35
camion lent	30	10	40
arrivée dans le quartier	40	3	43
arrivée à la maison	43	0	43

exemple du temps de trajet

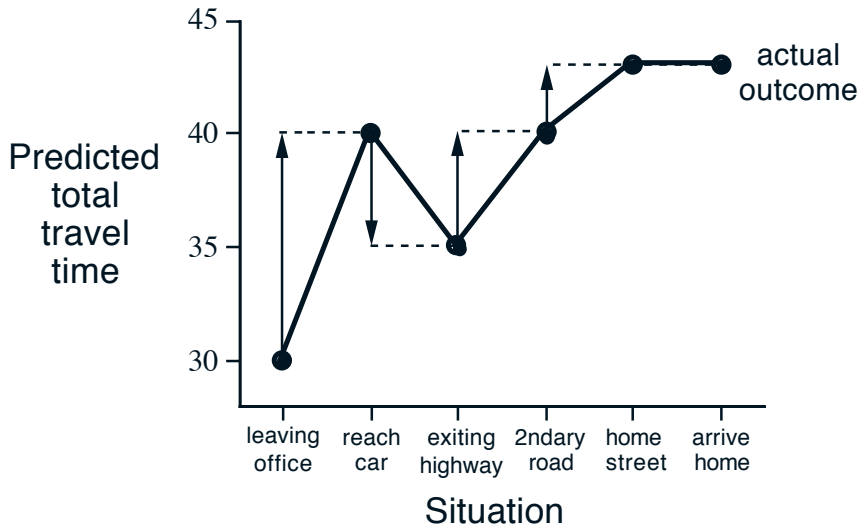
update avec méthode de Monte Carlo

$\alpha = 1$



update avec $\mathrm{TD}(0)$

$\alpha = 1$



Avantages des méthodes TD

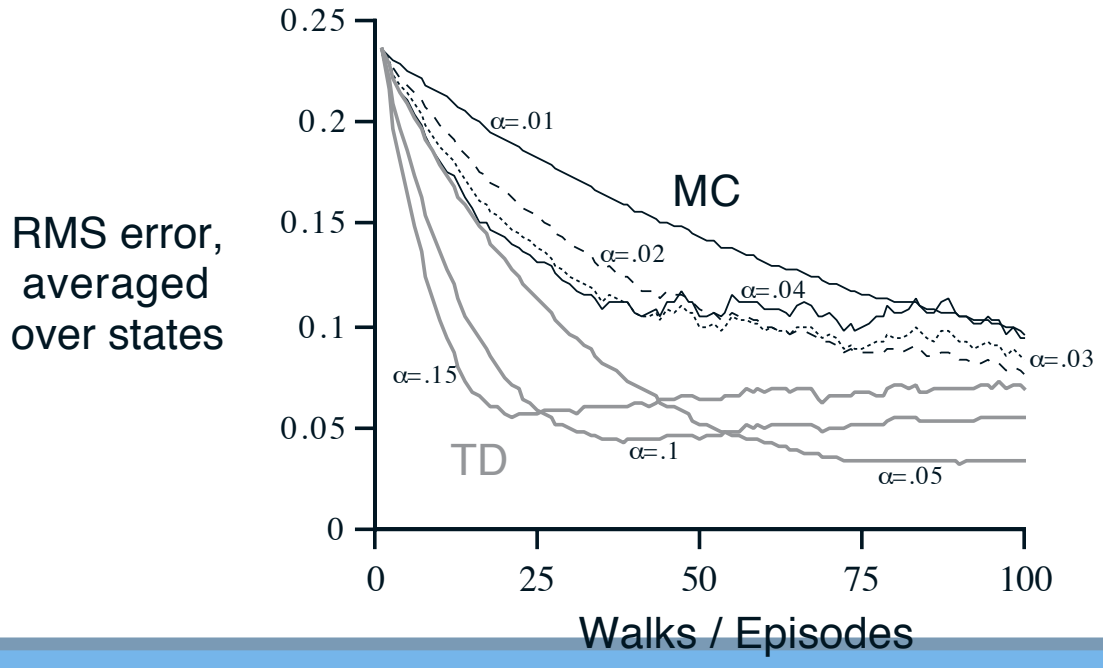
- pas besoin de connaissances des modèles
- méthode adaptée pour une utilisation ", et pas besoin d'attendre la fin de l'épisode
  les méthodes TD font une mise à jour après chaque itération
- il y a des garanties théoriques de convergence

Avantages des méthodes TD

- Pas de résultats théoriques comparant les performances des méthodes TD aux méthodes Monte Carlo.

⇨ en pratique, les méthodes TD sont plus rapides sur des problèmes stochastiques.

équiprobabilité d'aller à gauche ou à droite.



Evaluation de la politique optimale : TD "on policy"

- On apprend la fonction de valeur des actions.
- Comme pour TD(0) pour la fonction de valeurs, on a :

$$q(s_t,a_t) \leftarrow q(s_t,a_t) + \alpha\,[r_{t+1} + \gamma q(s_{t+1},a_{t+1}) - q(s_t,a_t)]$$

State–action–reward–state–action (SARSA)

Initialise $q(s) \in \Delta(A)$ arbitrairement
Répète (éternellement) pour chaque épisode
    aller à l'état initial s
    choisir action $a \in A$ pour s à l'aide d'une politique dérivée de q (ex : ϵ-greedy)
    Répète pour chaque étape de l'épisode
        Exécute action a, observe $r \in \mathbb{R}$ et état suivant $s' \in S$
        choisir action $a' \in A$ pour s' à l'aide d'une politique dérivée de q
        $q(s,a) \leftarrow q(s,a) + \alpha\,[r + \gamma q(s',a') - q(s,a)]$
        $s \leftarrow s'$
        $a \leftarrow a'$
    jusqu'à ce que s soit terminal

Evaluation de la politique optimale : TD "on policy"

Théorème

L'algorithme converge vers la fonction optimale de valeur des actions sous les conditions suivantes :

- Glouton à la Limite avec Exploration Infinie
  - toutes les paires `(état,action)` sont explorées infiniement souvent $\lim_{k\to\infty} n_k(s,a) = \infty$
  - la politique converge vers une politique gloutonne $\lim_{k\to\infty} \pi_k(a|s) = 1$ pour $a = arg\max_{a' \in A} q(s,a')$
- $\sum_{t=1}^{\infty} \alpha_t = \infty$
- $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$

ϵ-greedy est GLEI si ϵ est une fonction décroissante (ex $e_k = \frac{1}{k}$)

Evaluation de la politique optimale : TD "off policy"

Q-learning (Watkins 1989)

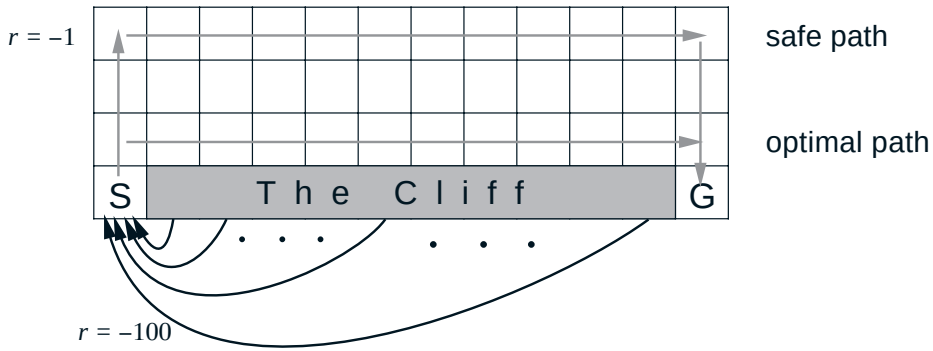
$$q(s_t,a_t) \leftarrow q(s_t,a_t) + \alpha \left[r_{t+1} + \gamma \max_{a \in A} q(s_{t+1},a) - q(s_t,a_t) \right]$$

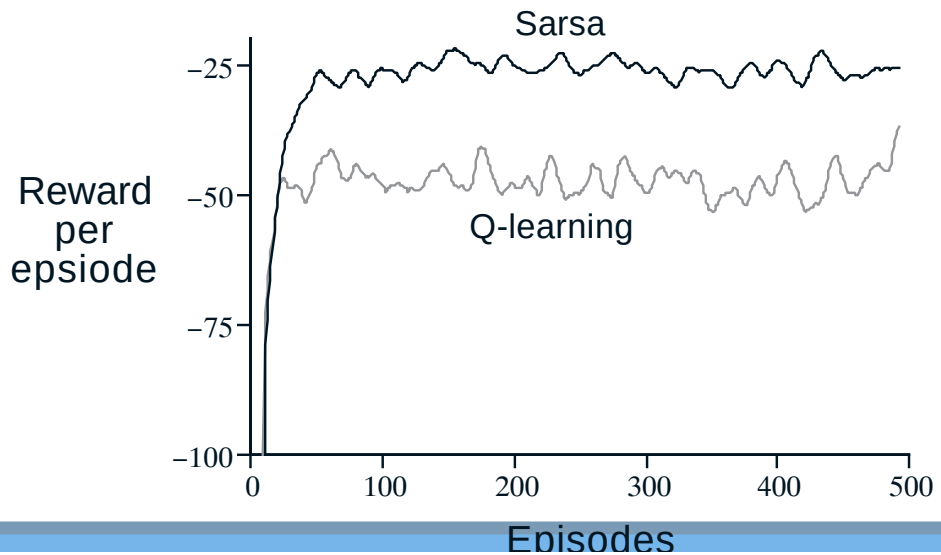
Apprend une estimation de $q^\star$ de façon indépendante à la politique suivie.

1. Initialise $q(s) \in \Delta(A)$ arbitrairement
2. Répète (éternellement) pour chaque épisode
3. aller à l'état initial s
4. choisir action $a \in A$ pour s à l'aide d'une politique dérivée de q (ex : ϵ-greedy)
5. Répète pour chaque étape de l'épisode
6. Exécute action a, observe $r \in \mathbb{R}$ et état suivant $s' \in S$
7. choisir action $a' \in A$ pour s' à l'aide d'une politique dérivée de q
8. $q(s,a) \leftarrow q(s,a) + \alpha [r + \gamma \max_{a" \in A} q(s',a") - q(s,a)]$
9. $s \leftarrow s'$
10. $a \leftarrow a'$
11. jusqu'à ce que s soit terminal

Evaluation de la politique optimale : TD "off policy"

- Pour assurer la convergence, il faut s'assurer de visiter suffisemment souvent les paires `(action,état)`.
- sous les hypothèses GLIE, Q-learning converge





Autres méthodes d'exploration

- soft max : choisir l'action a avec probabilité

$$\frac{e^{\frac{q_t(s,a)}{\tau}}}{\sum_{a' \in A} e^{\frac{q_t(s,a')}{\tau}}}$$

  - $\tau > 0$ est appelée la température
  - température haute ⇨ probabilité uniforme
  - température basse ⇨approche le comportement glouton
- initialisation optimiste : initialiser les valeurs de manière optimiste puis être glouton (Even-Dar & Mansour, NIPS 1994)
  ⇨force l'exploration à regarder les états qui semblent prometteur

Dilemme central : explorer ou exploiter

- contrairement au cas supervisé, les données sur lesquelles on travaille dépendent du comportement de l'agent !
- exploration : le but est d'apprendre le mieux possible
- exploitaiton : le but est d'optimiser au mieux ses récompenses
- défi de l'exploration : quelles actions vont améliorer au plus vite la connaissance de l'agent pour obtenir de meilleures récompenses.

⇨ l'exploration est un trait d'intelligence

- quelle politique doit suivre l'agent pour ne pas manquer les états qui donnent les bonnes récompenses (et sans passer trop de temps dans les états qui donnent de mauvaises récompenses)
- exploitation : préfère des actions qui ont mené a de "bons états"
- exploration : prendre une action qui pourrait nous mener à de bons états.

Stratégie d'exploration

- ϵ-greedy
  - facile à implémenter et très utilisée
  - convergence garantie (avec un taux d'exploration qui décroit de bonne manière)
  - il faut un nombre exponentiel d'échantillons pour garantir convergence
- Boltzmann
  - même problème pour le nombre d'échantillons

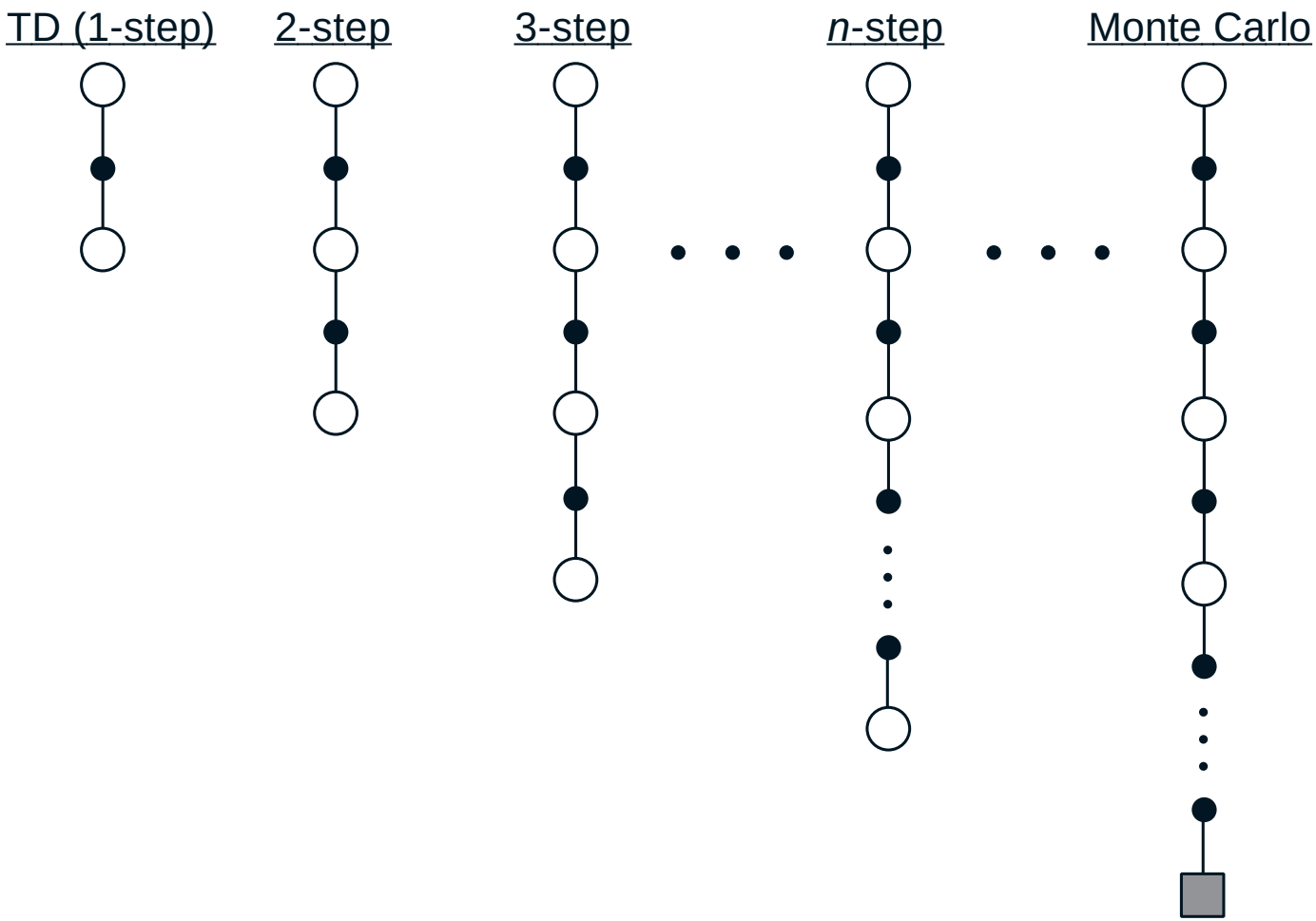
Explicit-Exploit-or-Explore E^3 (Kearns & singh 2002)

- Construit un modèle de PDM
  - optimiste
  - connait la récompense maximales
- compte le nombre de fois où chaque paire `(action,état)` a été visité pour estimer la qualité du modèle
  - ⇨ on connait l'état si on l'a visité un certain nombre de fois
  - Garantie statistique pour définir le nombre "suffisant" de viste
    $\mathcal{O}((NTG^T_{max}/\epsilon)^4 \mathrm{Var}_{max} log(\frac{1}{\delta}))$ ou Var_{max} est la variance maximale des récompenses sur tous les états
- E^3 gère deux modèles : le modèle avec les états connus, et celui avec les états par encore connu.
  connu ⇨ exploitation
  pas encore connu ⇨ exploration

R_{max} (Brafman & Tennenholtz 2002)

- utilise la même idée, mais avec un seul modèle
- initialement, tout est inconnu, on estime avec une valeur maximale R_{max} toutes les récompenses, le modèle de transition est estimé déterministe
- on compte aussi les transitions observées pour déterminer ensuite lesquelles sont connues
- on calcule une politique optimale pour ce qu'on connait jusqu'ici
  - si l'état était connu ⇨ exploitation
  - si l'état n'était pas connu ⇨ on explore l'état le plus prometteur

Entre TD(0) et Monte Carlo : TD(n)



baser la mise à jour après quelques récompenses
(entre 1 et la fin de l'épisode)

Entre TD(0) et Monte Carlo : TD(n)

$$
\begin{aligned}
R_t^{(1)} &= r_{t+1} + \gamma V_t(S_{t+1}) \\
R_t^{(2)} &= r_{t+1} + \gamma r_{t+1} + \gamma^2 V_t(S_{t+2}) \\
&\cdots \\
R_t^{(n)} &= r_{t+1} + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(S_{t+n})
\end{aligned}
$$

- pour un algorithme, à t, on devrait mettre à jour la valeur de l'état visité à $t-n$

⇨ lors des $n-1$ premières étapes, pas de mise à jour

⇨ mise à jour à chaque étape de l'état visité il y a n étapes

⇨ lorsqu'on atteint la fin de l'épisode, on met à jour les n étapes précédant la fin

Entre TD(0) et Monte Carlo : TD(n)

⇨ Monte Carlo devient un cas spécial où n est suffisemment grand.

Quel n permet d'apprendre le plus vite ?

⇨ Méthodes intéressantes théoriquement (bonnes propriétés de réduction d'erreurs), mais plus difficiles à implémenter.

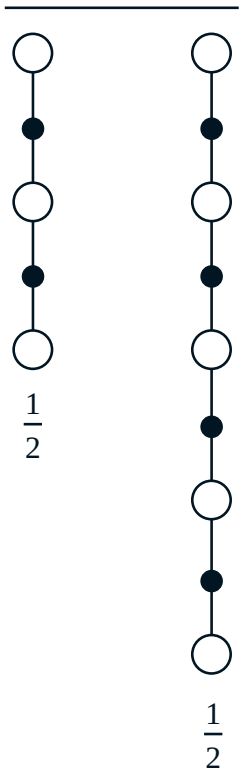
Sarsa à n étapes

On utilise cette idée pour la fonction q, en utilisant ϵ-greedy.

$$q_{t+n}(s_t, a_t) = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n \sum_{a \in A} \pi(s|S_{t+n}) q_{t+n-1}(s_{t+n}, a)$$

autre idée pour utiliser TD(0), TD(1), ..., TD(n)

On peut aussi penser à "mixer" les récompenses à plus ou moins long terme.

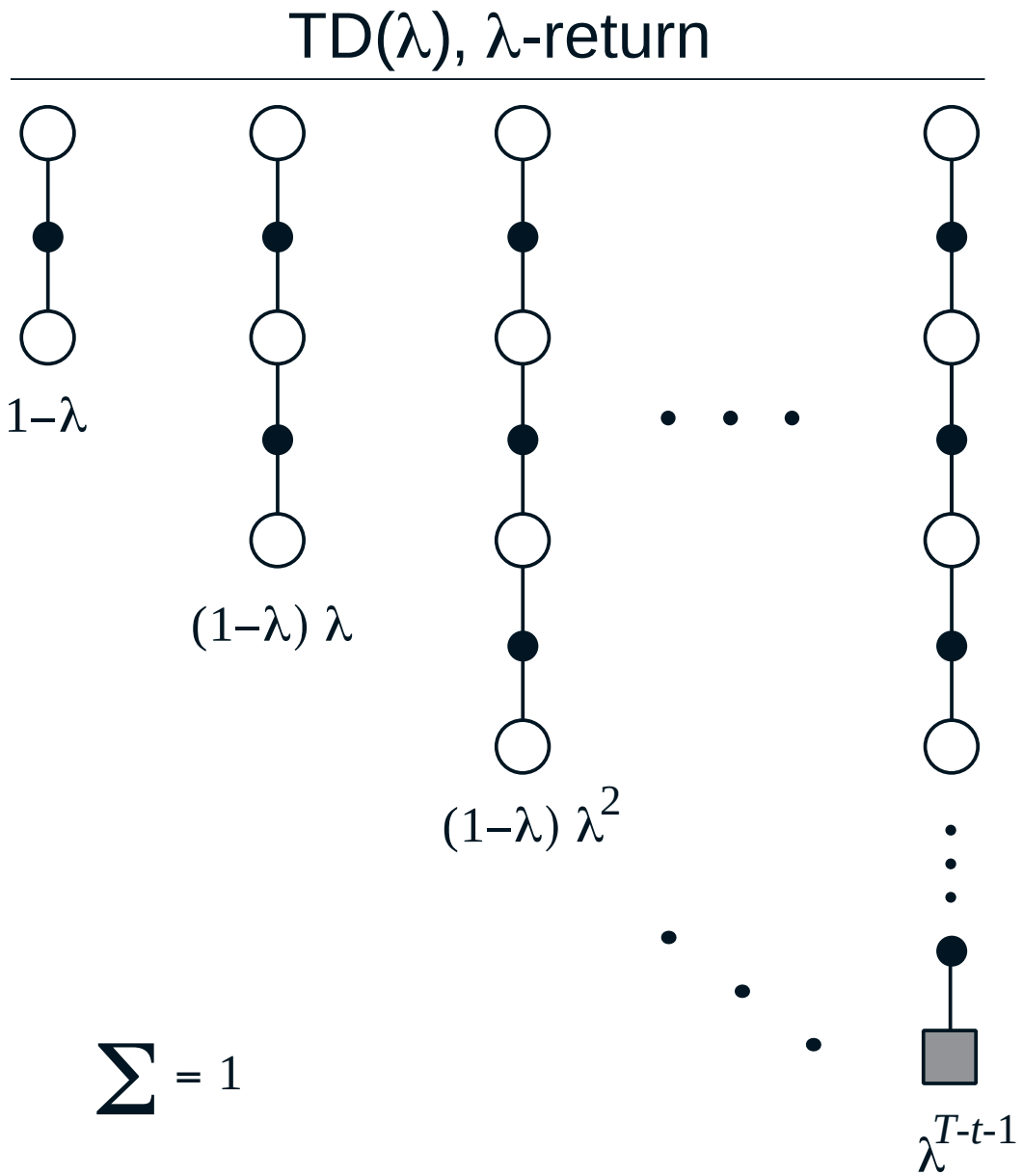


$$R_t^{moy} = \frac{1}{2} R_t^{(2)} + \frac{1}{2} R_t^{(4)}$$

autre idée pour utiliser TD(0), TD(1), …, TD(n)

- On peut donner des poids différents à chaque gain
  (ça marche même pour un ensemble infini)
- Tant que la somme des poids est 1.
- On a une propriété de réduction d'erreurs.

Un mixage particulier : TD(λ)



Un mixage particulier : TD(λ)

- $\lambda \in [0,1]$ similaire au paramètre de dévaluation.

⇨ A chaque étape suivante, le poids décroit avec le facteur λ

- le gain à n étapes possède un poids proportionnel à λ^{n-1}
- $\sum_{n=0}^{\infty} \lambda^n = \frac{1}{1-\lambda}$

⇨ facteur $1-\lambda$ pour que les poids somment à 1

- $R_t^\lambda = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)}$
- $\lambda = 1$ ⇨ Monte Carlo
- $\lambda = 0$ ⇨ TD(0)

On peut donc bâtir des algorithmes plus compliqués

mais c'est assez pour le moment !

Résumé

Types d'algorithmes pour

- évaluer une politique donnée
- trouver une politique optimale

Ce qu'on apprend

- $v_\pi : S \to \mathbb{R}$ la fonction de valeurs qui donne la valeur **à long terme** de chaque état en suivant une politique
- $\pi : A \times S \to \mathbb{R}$ la fonction qui donne la valeur **à long terme** de prendre une action puis de suivre une politique π depuis un état.

Algorithmes :

- modèle de transition et de récompenses connus ⇨ policy/value iteration
- Algorithmes pour domaines épisodique ⇨ méthodes de Monte Carlo
- Algorithme qui fonctione sans connaître ni bâtir un modèle Attention au dilemme Exploration Vs Exploitation ⇨ SARSA , Q-learning
- Algorithme qui fonctionne en bâtissant un modèle ⇨ E^3, R_{max}
- on policy : on améliore la politique que l'on suit / off policy : on utilise une stratégie pour apprendre une autre