Apprentissage par Renforcement

Apprentissage Numérique

Pierre Gérard

Université de Paris 13 - LIPN

Master MICR

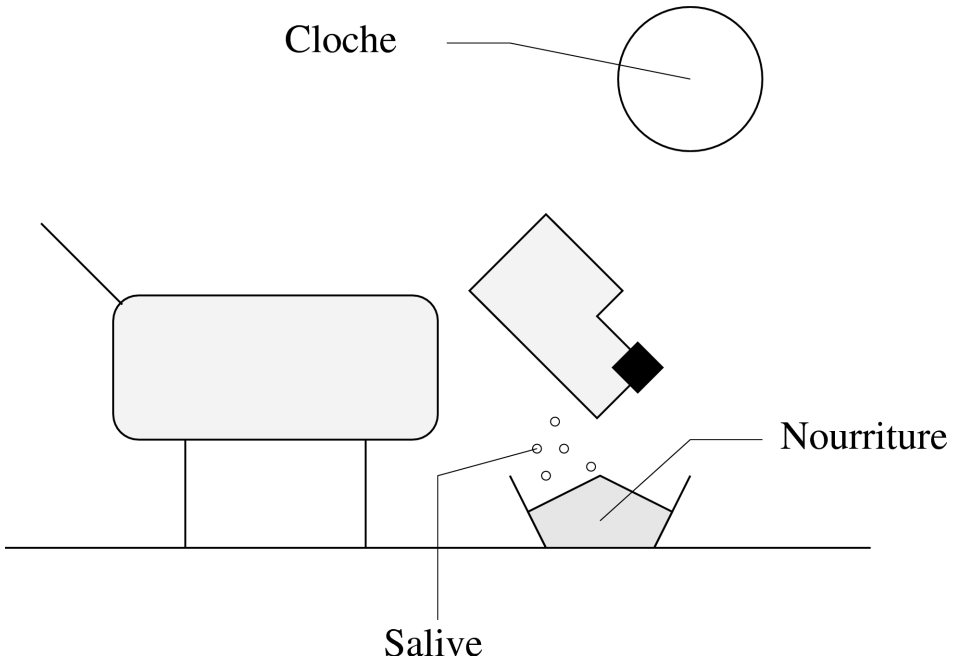
LaTeX

Plan

1. Problème d'AR
   - Introduction
   - Formalisation du problème
2. Solutions élémentaires
   - Programmation Dynamique
   - Monte Carlo
   - Différence temporelle (TD)
3. Solutions unifiées
   - Traces d'éligibilité
   - Approximation de fonctions
4. Perspectives
   - AR indirect
   - POMDP
   - Continuité du temps

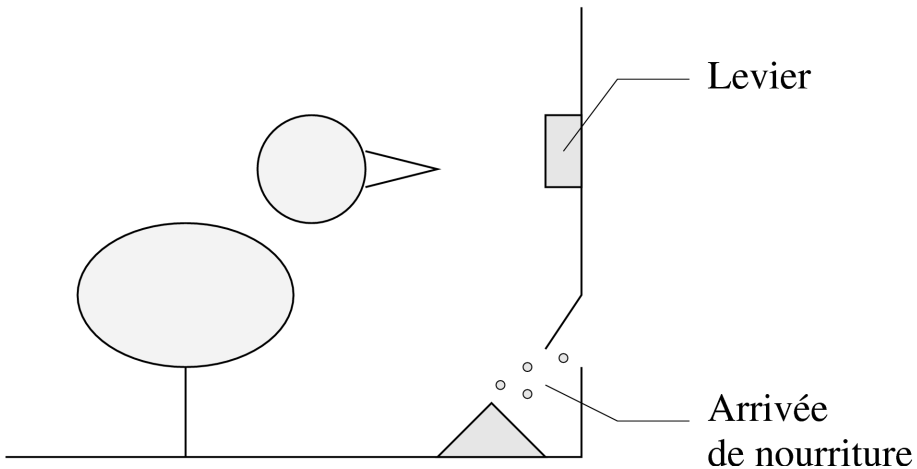
Le chien de Pavlov (conditionnement classique)

- Présenter de la nourriture à un chien provoque sa salivation
- Expérience répétée : juste après avoir présenté la nourriture, on fait sonner une cloche
- Le chien finit par saliver au simple son de la cloche



La boîte de Skinner

- Appuyer sur le levier permet de délivrer de la nourriture
- Le pigeon apprend par essais-erreurs à appuyer sur le levier pour que de la nourriture soit délivrée
- On peut aussi compliquer le dispositif pour conditionner la délivrance de nourriture par un signal sonore ou lumineux



Conditionnement opérant

- Apprentissage par essais/erreurs d'une action éventuellement conditionnée par une situation
  - Dès qu'il presse le bouton, le pigeon est **récompensé**
  - La récompense **renforce** cette action
  - Le pigeon **apprend** à presser le bouton de plus en plus souvent
- Modifier la récompense conduit à modifier le comportement que l'animal apprend

Apprentissage par Renforcement

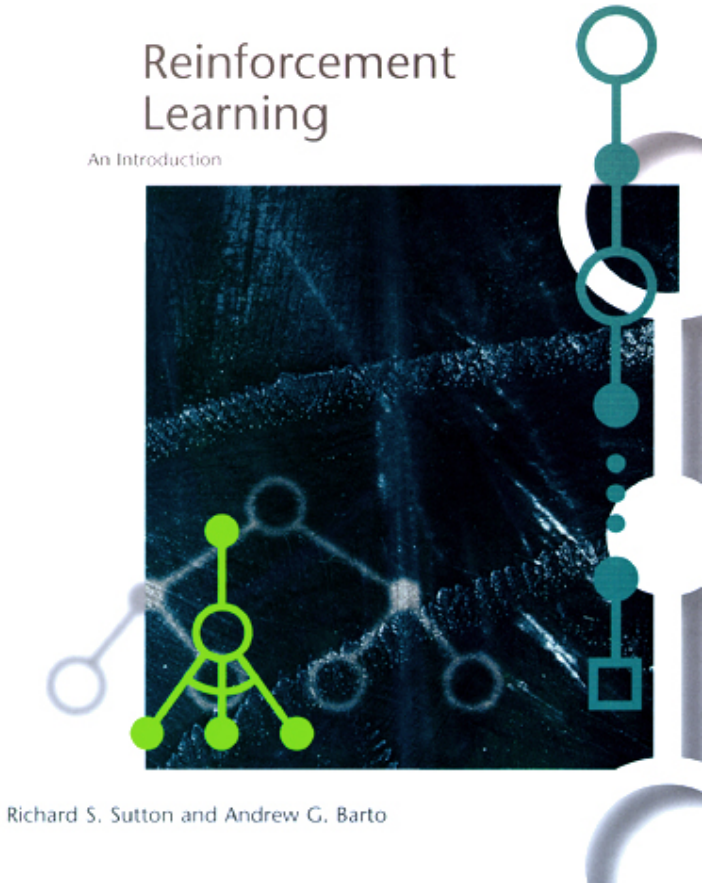
- Reproduction artificielle du conditionnement de l'animal

Objectif

Comment faire apprendre un comportement à une machine en lui distribuant des récompenses ?

- Adaptation du comportement du système à son **environnement**
- Apprentissage d'une **politique**, *càd* de règles permettant de choisir une action en fonction de la situation
- La politique doit permettre de maximiser la récompense

LE Livre



- **R. S. Sutton and A. G. Barto** *(1998)*
  Reinforcement Learning : An Introduction.
  MIT Press, 1998.

Le bandit n'est pas manchot

- Problème
  - Machine à sous à n leviers
  - A chaque essais, n actions possibles
  - On est récompensé par les gains après chaque essai
  - **Objectif** : maximiser ses gains sur un certain nombre d'essais

Exploration *vs.* exploitation

- Chaque action a une valeur, représentant l'espérance moyenne des gains en choisissant l'action en question
  - Connaissant la valeur de chaque action, il serait facile de résoudre le problème
  - Problème de **compromis exploration/exploitation**
- **Exploitation** : Exploitation des connaissances acquises jusqu'ici
- **Exploration** : Acquisition de nouvelles connaissances

Valeur d'une action

- On note :
  - $Q^\star(a)$ la valeur réelle de l'action a
  - $Q_t(a)$ la valeur de a estimée après le $t^{\text{ème}}$ essai
  - k_a le nombre de fois que a a été choisie avant le $t^{\text{ème}}$ essai
  - $r_1, r_2, \cdots, r_{k_a}$ les récompenses reçues après avoir choisi a

Estimation de la valeur d'une action

$$Q_t(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a}$$

- On suppose $Q_0(a) = 0$
- Lorsque $k_a \to \infty$, alors $Q_t(a)$ converge vers $Q^\star(a)$

Politique ϵ-greedy

- Action gloutonne (greedy) : à l'instant t, sélectionner $a^\star$ telle que

$$Q_t(a^\star) = \max_a Q_t(a)$$

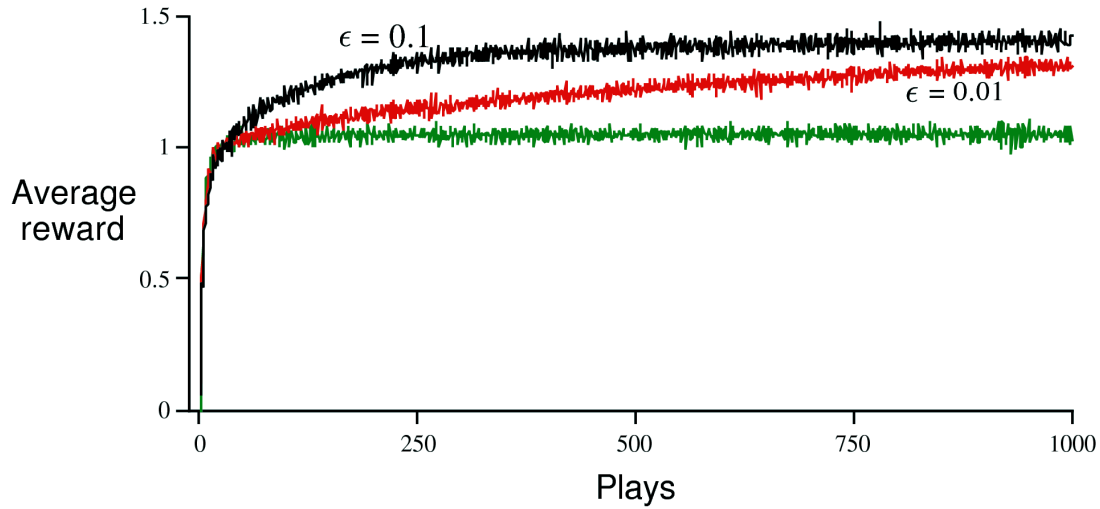
  - **Exploitation** de la connaissance courante de Q uniquement
  - Aucune exploration
- Pour ajouter simplement de l'exploration : actions ϵ-**greedy**

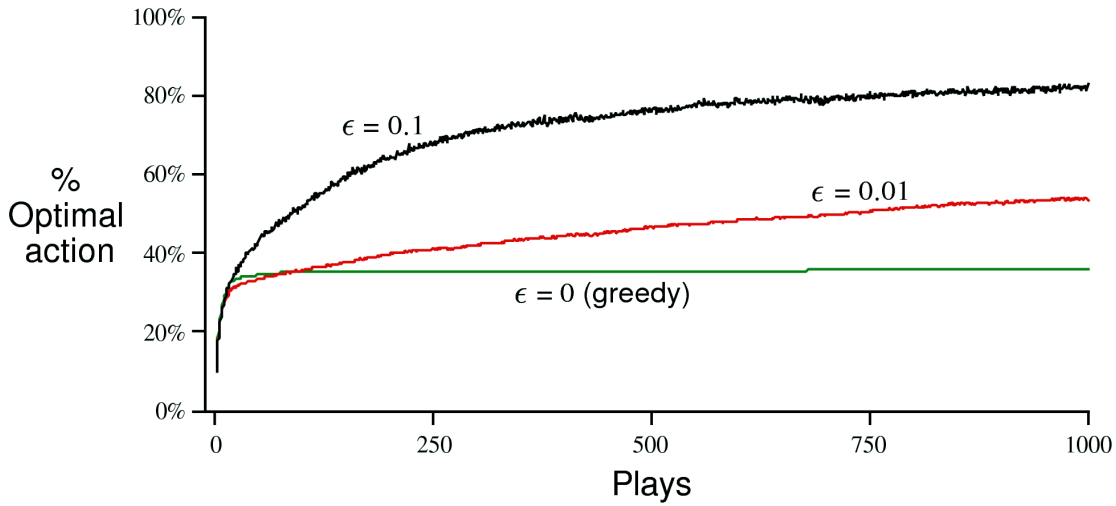
- Avec une probabilité de ϵ, on choisit une action au hasard
- Avec une probabilité de $1 - \epsilon$, on choisit l'action gloutonne $a^\star$

- Dans ce cas, si $t \longrightarrow \infty$ alors $k_a \longrightarrow \infty$ et donc $Q_t(a) \longrightarrow Q^\star(a)$

Résultats expérimentaux

- 2000 problèmes générés aléatoirement
  - $n = 10$
  - Les dix $Q^\star(a)$ sont choisies à chaque fois selon $\mathcal{N}(0, 1)$
- Pour chaque action a, les récompenses sont choisies selon $\mathcal{N}(Q^\star(a), 1)$





Softmax

- Il y a des méthodes plus sophistiquées que ϵ-greedy pour gérer le compromis exploration/exploitation
- Si les actions non gloutonnes sont très mauvaises, on peut vouloir faire varier le facteur aléatoire plus finement
- **Softmax** : au $t^{\text{ème}}$ essai, l'action a est choisie avec une probabilité définie par une distribution de Boltzmann (ou Gibbs)

$$\frac{e^{\frac{Q_t(a)}{\tau}}}{\sum_{b=1}^{n} e^{\frac{Q_t(b)}{\tau}}}$$

- τ est appelé **température**
  - Quand $\tau \longrightarrow \infty$, la politique devient aléatoire
  - Quand $\tau \longrightarrow 0$, la politique devient gloutonne

Incrémentalité de l'évaluation

- **Problème :** Evaluation de $Q^\star(a)$ pour toute action a
- De manière non incrémentale, on stocke tous les $r_1, r_2 \cdots r_{k_a}$ pour – à la fin – calculer

$$Q_t(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a}$$

  - Pas très efficace en temps de calcul
  - Pas très efficace en termes de mémoire utilisée

Incrémentalité de l'évaluation

- De manière incrémentale, on calcule Q_{k+1} en fonction de Q_k

$$
\begin{aligned}
Q_{k+1} &= \frac{1}{k+1}\sum_{i=1}^{k+1} r_i \\
&= \frac{1}{k+1}\left(r_{k+1} + \sum_{i=1}^{k} r_i\right) \\
&= \frac{1}{k+1}\left(r_{k+1} + kQ_k + Q_k - Q_k\right) \\
&= \frac{1}{k+1}\left[r_{k+1} + (k+1)Q_k - Q_k\right] \\
&= Q_k + \frac{1}{k+1}\left[r_{k+1} - Q_k\right]
\end{aligned}
$$

Correction d'erreur

$$Q_{k+1} = Q_k + \frac{1}{k+1}\left[r_{k+1} - Q_k\right]$$

- Forme générale :

$$Val_{est} \leftarrow Val_{est} + Tx_{app}\left[Val_{cible} - Val_{est}\right]$$

avec

- Val_{est} la valeur estimée et Val_{cible} la valeur cible
- $Val_{cible} - Val_{cible}$ l'erreur commise sur l'estimation
- Tx_{app} un taux d'apprentissage

- Ici, $Tx_{app} = \frac{1}{k}$. On utilise souvent le symbole α
  - Plus généralement, $\alpha_k(a) = \frac{1}{k_a}$ ou plus simplement $\alpha_k = \frac{1}{k}$

Problèmes non stationaires

- Utiliser $\alpha_k = \frac{1}{k}$ donne la valeur exacte si le problème ne change pas au cours du temps
  - Par exemple, les gains associés à chaque action bras de la machine à sous restent constants
- Un problème qui change au cours du temps est un problème **non stationnaire**
- Dans ce cas, on utilise un taux d'apprentissage **constant** $\alpha \in [0, 1]$ et

$$Q_{k+1} = Q_k + \alpha \left[r_{k+1} - Q_k \right]$$

Terme général de l'évolution de la valeur

$$
\begin{aligned}
Q_k &= Q_{k-1} + \alpha\,[r_k - Q_{k-1}] \\
&= \alpha r_k + (1-\alpha) Q_{k-1} \\
&= \alpha r_k + (1-\alpha)\alpha r_{k-1} + (1-\alpha)^2 Q_{k-2} \\
&= \alpha r_k + (1-\alpha)\alpha r_{k-1} + (1-\alpha)^2 \alpha r_{k-2} + \\
&\quad \cdots + (1-\alpha)^{k-1} \alpha r_1 + (1-\alpha)^k Q_0 \\
&= (1-\alpha)^k Q_0 + \sum_{i=1}^{k} \alpha (1-\alpha)^{k-i} r_i
\end{aligned}
$$

- C'est une **somme pondérée** puisque $(1-\alpha)^k + \sum_{i=1}^{k} \alpha(1-\alpha)^{k-i} = 1$
- Les récompenses reçues le plus récemment ont un poids plus important : on « oublie » les informations obtenues dans un passé lointain

Variation du taux d'apprentissage

- On peut vouloir faire varier le taux d'apprentissage au cours du temps
  - Apprentissage rapide au début
  - Réglages plus fins en fin d'apprentissage
- Choix d'une séquence $\{\alpha_k(a)\}$ garantissant la convergence avec une probabilité de 1

Conditions de convergence

$$\sum_{k=1}^{\infty} \alpha_k(a) = \infty$$

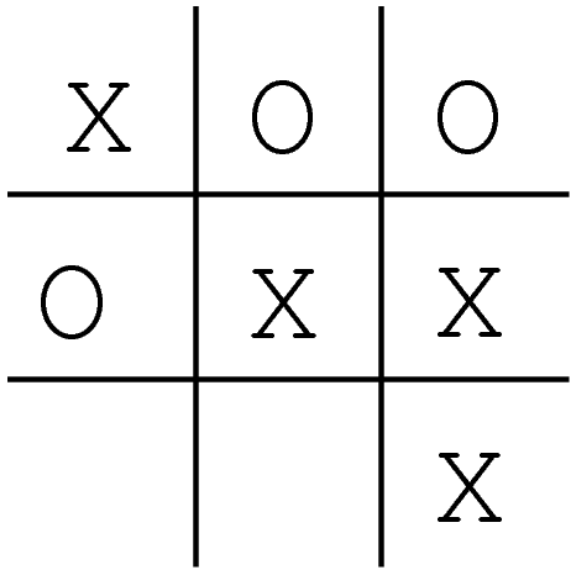
$$\sum_{k=1}^{\infty} {\alpha_k}^2(a) < \infty$$

Morpion (Tic Tac Toe)

Différence avec le Bandit-Manchot

Dans ce type de problème, le choix d'une action est conditionné par la disposition courante des X et des O sur la grille.

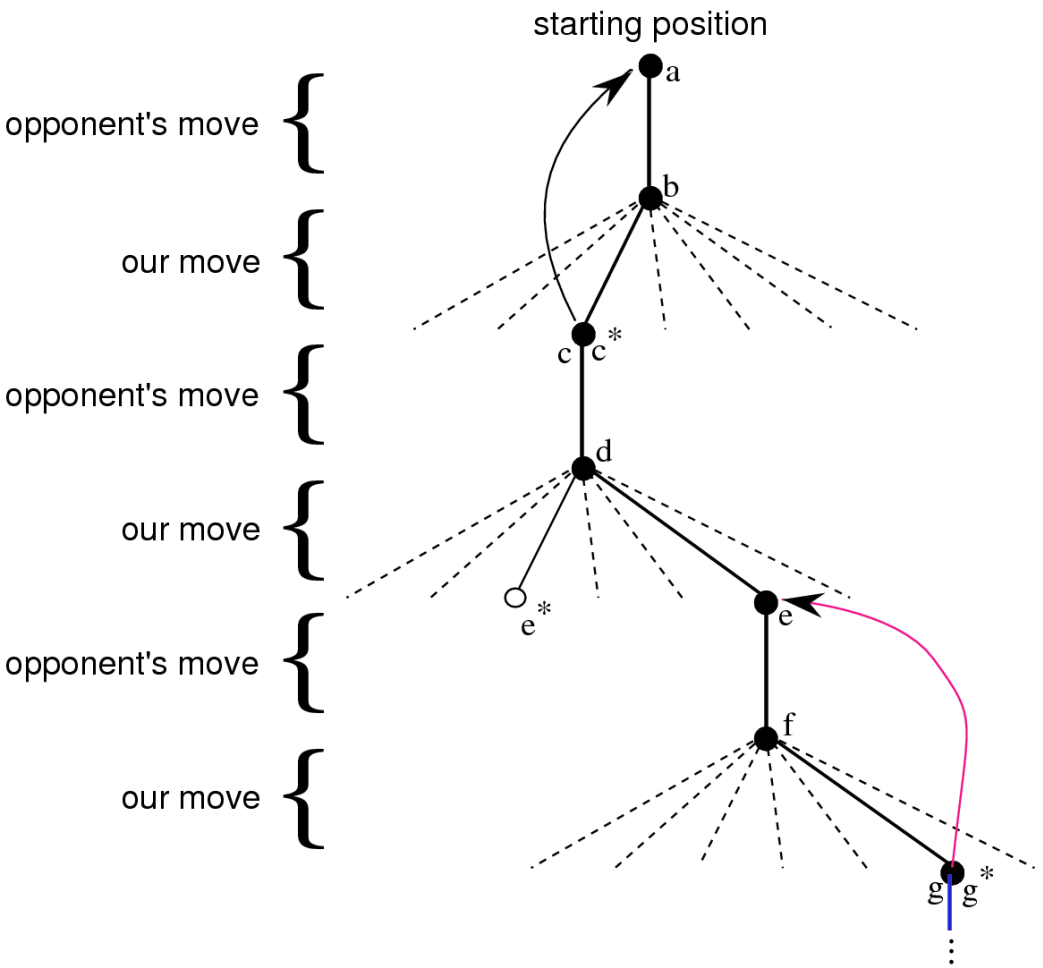
- Tour à tour, un joueur joue les X et un autre les O, il faut aligner trois X ou trois O
- Comment jouer de manière à gagner contre un opposant imparfait ?
  - **Min-max** : suppose un opposant jouant de manière particulière
  - **Programmation dynamique** : suppose que l'on connaisse parfaitement l'opposant
  - **Méthodes évolutionnistes** : parcourir l'espace de toutes les politiques possibles et sélectionner les meilleures, suppose énormément d'essais



Valeur d'un état

- Recensement des **états** possibles du jeu
  - Configurations particulières avec des X et des O dans certaines cases
- Association d'une **valeur** à chaque état
  - Si $V(a) > V(b)$ alors l'état a est considéré comme « meilleur » que l'état b
  - La valeur d'un état sera d'autant plus grande qu'à partir de cet état, on a une grande probabilité de gagner
  - Exemple si on joue les X
    - La probabilité de gagner à partir d'un état avec trois X alignés est 1
    - La probabilité de gagner à partir d'un état avec trois O alignés est 0

Actions exploratoires



- Mouvements **gloutons**
  - Les plus fréquents
  - Choisir l'action qui mène au meilleur état en regard des valeurs actuelles
- Mouvements **exploratoires**
  - Les moins fréquents
  - Choix au hasard pour aller vers des états jamais visités
  - Gain de connaissance sur la valeur « réelle » de chaque état

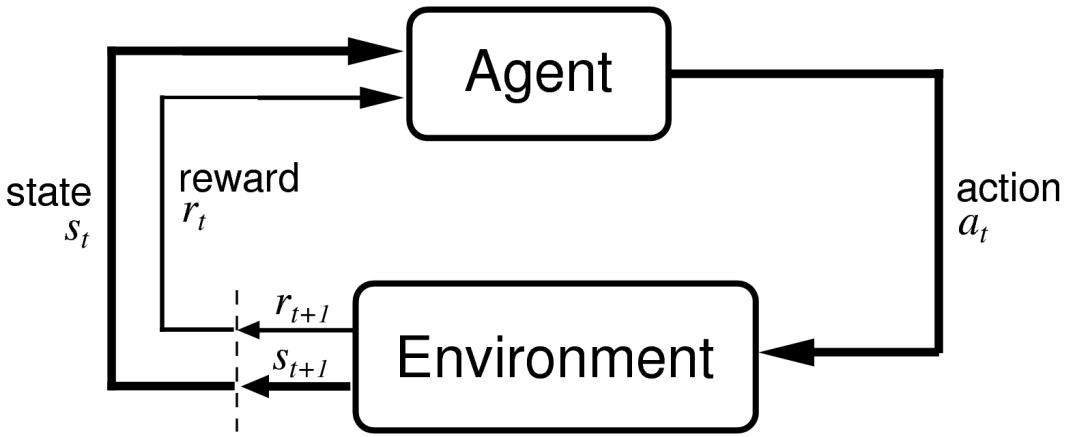
Lexique

- Récompense : **reward**
- Renforcement : **reinforcement**
- Politique : **policy**
- Programmation dynamique : **dynamic programing**
- Méthodes évolutionnistes : **evolutionary methods**
- Fonction valeur : **value function**
- Glouton : **greedy**
- Exploratoire : **exploratory**
- Compromis exploration/exploitation : **exploration/exploitation tradeoff**
- Valeur d'action : **action value**
- Stationaire : **stationary**
- Somme pondérée : **weighted sum**
- Taux d'apprentissage : **learning rate / step-size parameter**

Plan

1. Problème d'AR
   - Introduction
   - Formalisation du problème
2. Solutions élémentaires
   - Programmation Dynamique
   - Monte Carlo
   - Différence temporelle (TD)
3. Solutions unifiées
   - Traces d'éligibilité
   - Approximation de fonctions
4. Perspectives
   - AR indirect
   - POMDP
   - Continuité du temps

Interface agent/environnement



- Le temps est divisé en **pas de temps** discrets $t \in \mathbb{N}^{+\star}$
- Dans l'**état** $s_t \in \mathcal{S}$, l'agent émet l'**action** $a_t \in \mathcal{A}(s_t)$
  - $\mathcal{A}(s_t)$ est l'ensemble des actions disponibles à partir de s_t
  - L'action est choisie par l'agent selon sa **politique** courante π_t
  - $\pi_t(s, a) \in [0, 1]$ est la probabilité qu'à l'instant t, l'agent choisisse l'action a dans l'état s
- En **retour**, il reçoit de l'environnement une **récompense immédiate** $r_{t+1} \in \mathbb{R}$ et son nouvel état $s_{t+1} \in \mathcal{S}$

Récompense et buts

- A chaque pas de temps, l'agent reçoit une récompense immédiate $r_t \in \mathbb{R}$
- L'objectif de l'agent est de maximiser le **cumul** des récompenses obtenues à long terme
  - Compromis nécessaire entre récompenses immédiates et récompenses futures
- En AR, ces signaux de récompense formalisent la notion de **but**, ou d'objectif
  - Les récompenses indiquent à l'agent à **quoi** aboutir
  - Elles n'indiquent pas **comment** y parvenir
- L'AR n'est pas un apprentissage directement **supervisé**

Attention

Les récompenses sont déterminées par l'**environnement**

Retour attendu

- **Retour attendu** : cumul des récompenses obtenues à partir d'un instant t

Retour attendu

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T$$

où T est l'instant final

- Ici, la vie de l'agent est découpée en **épisodes** de plusieurs pas de temps chacun
  - Parcours de l'entrée d'un labyrinthe jusqu'à la sortie
  - Une partie de morpion
  - ...
- Chaque épisode se termine dans un **état final**

Tâches épisodiques et continues

- Les **tâches** qui peuvent être découpées en épisodes sont des tâches **épisodiques**
- Quand ce n'est pas possible, on parle de tâches **continues**
  - L'instant final serait alors $T = \infty$
  - Le retour R_t pourrait être infini
- Pour borner le retour, on introduit la notion de **retour déprécié**

Retour déprécié

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

où $\gamma \in [0, 1]$ est le **facteur de dépréciation**

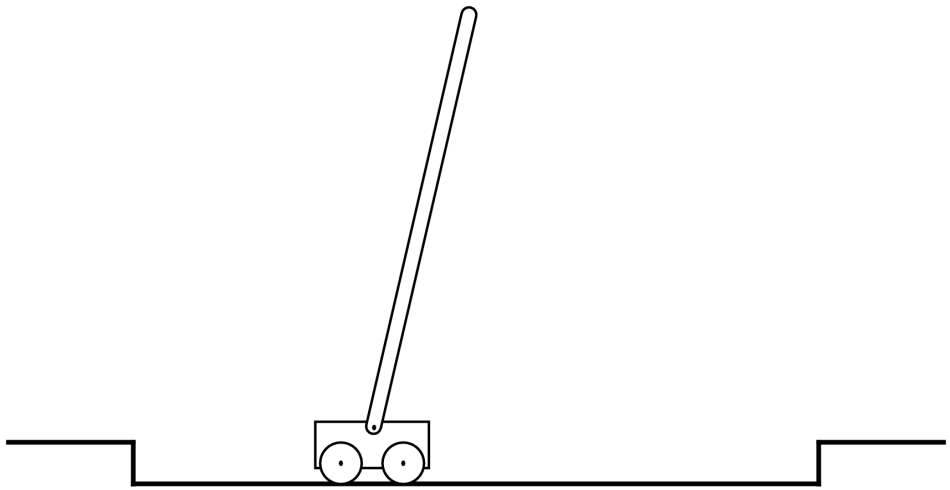
Facteur de dépréciation

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- Plus une récompense r_{t+k+1} est lointaine dans le futur (k est grand), moins son poids est important
- γ permet de régler l'importance accordée aux récompenses futures vis-à-vis des récompenses immédiates
  - Si $\gamma = 0$, les récompenses futures sont ignorées et seules les récompenses immédiates comptent (agent cigale)
  - Si $\gamma = 1$, les récompenses futures comptent autant que les récompenses immédiates (agent fourmi)
  - Le plus souvent, on choisit $\gamma \in]0, 1[$
  - Si $\gamma = 0.5$, un « tiens » vaut mieux que deux « tu l'auras »)

Exemple du pendule inversé

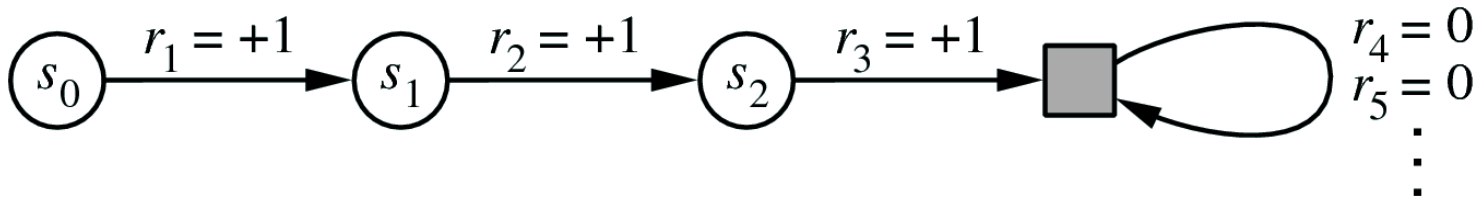
- Définition du problème comme une tâche
  - **Episodique** : $r_t = 0$ en cas d'échec et 1 tout le reste du temps
  - **Continue** : $r_t = -1$ en cas d'échec et 0 tout le reste du temps



- Avec $R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T$
  - Avec la tâche **épisodique**, le retour est plus important si la politique est efficace (si le pendule met plus de temps à tomber)
  - Avec la tâche **continue**, le retour est le même quelle que soit la politique
- Avec $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$
  - Dans tous les cas, le retour est plus important si la politique est efficace

Notation unifiée

- **État absorbant** : état à partir duquel on peut entreprendre n'importe quelle action mais qui ne génère de transitions que vers lui-même
  - Les récompenses alors obtenues sont nulles



- Pour une tâche épisodique, on peut formaliser les états finaux comme des états absorbants
- Dans tous les cas, on peut alors utiliser la notation

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$$

et ainsi, on peut avoir soit $T = \infty$ soit $\gamma = 1$, mais pas les deux

Propriété de Markov

- En règle générale, les conséquences des actions entreprises dépendent de tout ce qui s'est passé avant, et la dynamique de l'environnement peut être définie par la distribution de probabilité

$$Pr\left\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, \cdots, r_1, s_0, a_0\right\}$$

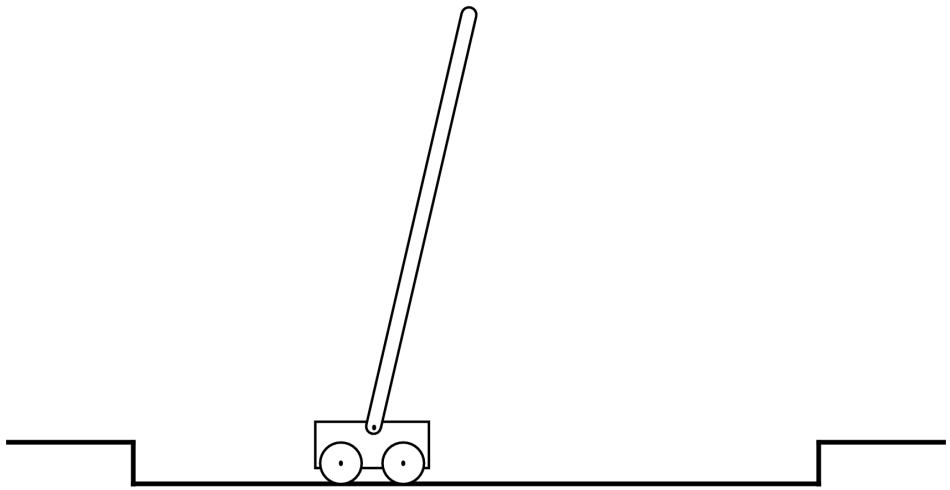
- La **propriété de Markov** est vérifiée quand cette dynamique peut être décrite par la distribution

$$Pr\left\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\right\}$$

- En d'autres termes, elle est vérifiée quand les conséquences d'une action ne dépendent que de l'état courant
- Dans ce cas, les choses sont plus simples

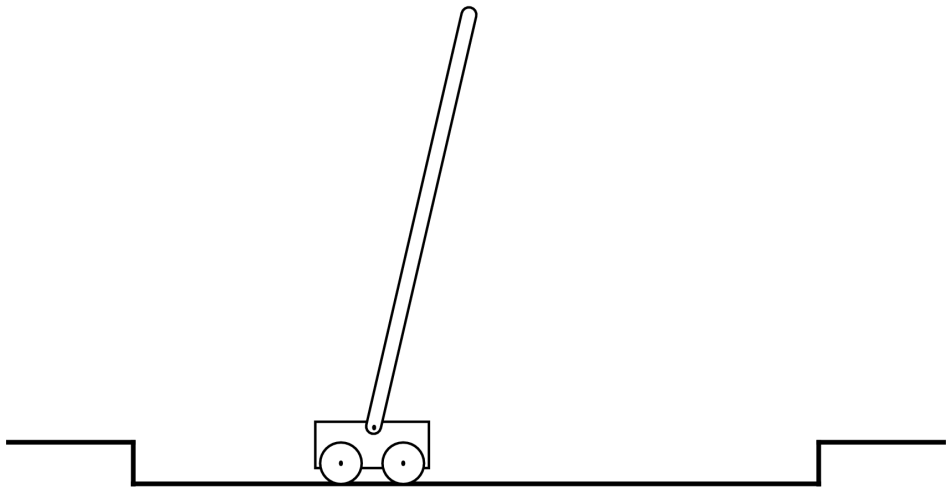
Exemple du pendule inversé

- Informations nécessaires pour que le problème soit markovien



Exemple du pendule inversé

- Informations nécessaires pour que le problème soit markovien
  - Angle du pendule
  - Position du chariot
  - Vitesse du pendule et du chariot
  - Accélération



Processus de décision markovien (PDM)

- Un PDM est défini comme un ensemble $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ avec
  - $\mathcal{S}$ un espace d'états
  - $\mathcal{A}$ un espace d'actions
  - $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ une fonction de transition

$$\mathcal{T}(s, a, s') = \mathcal{T}_{sa}^{s'} = Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$$

  - $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ une fonction de récompense

$$\mathcal{R}(s, a) = \mathcal{R}_{sa} = E\{r_{t+1} \mid s_t = s, a_t = a\}$$

- Un problème d'**apprentissage** par renforcement se définit formellement comme un **Processus de Décision Markovien** où $\mathcal{T}$ et $\mathcal{R}$ sont *a priori* inconnus

Fonction valeur

- **Fonction valeur** : fonction qui estime à quel point il est bon pour l'agent d'être dans un état particulier
- La valeur d'un état (la proximité de récompenses) dépend des actions qui seront entreprises à partir de cet état
  - Une fonction valeur est donc attachée à une politique $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$

$$
\begin{aligned}
V^{\pi}(s) &= E_{\pi}\left\{R_t \mid s_t = s\right\} \\
&= E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\}
\end{aligned}
$$

valable pour tous les $s \in \mathcal{S}$, où $E_{\pi}\left\{\cdot \mid \cdot\right\}$ désigne l'espérance en suivant π

Fonction action-valeur

- Pour tout $s \in \mathcal{S}$ et $a \in \mathcal{A}(s)$, on peut aussi définir une **fonction action-valeur**

$$\begin{aligned} Q^{\pi}(s,a) &= E_{\pi}\{R_t \mid s_t = s, a_t = a\} \\ &= E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\} \end{aligned}$$

qui définit le retour attendu en partant de l'état s, en émettant l'action a puis en suivant la politique π par la suite

Propriété fondamentale de la fonction valeur

$$
\begin{aligned}
V^{\pi}(s) &= E_{\pi}\left\{R_t \mid s_t = s\right\} \\
&= E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\} \\
&= E_{\pi}\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s\right\} \\
&= \sum_{a \in \mathcal{A}(s)} \pi(s,a) \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} E_{\pi}\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s'\right\}\right] \\
&= \sum_{a \in \mathcal{A}(s)} \pi(s,a) \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} V^{\pi}(s')\right]
\end{aligned}
$$

Équation de Bellman

- V^π est l'unique solution de l'équation

Équation de Bellman pour V^π

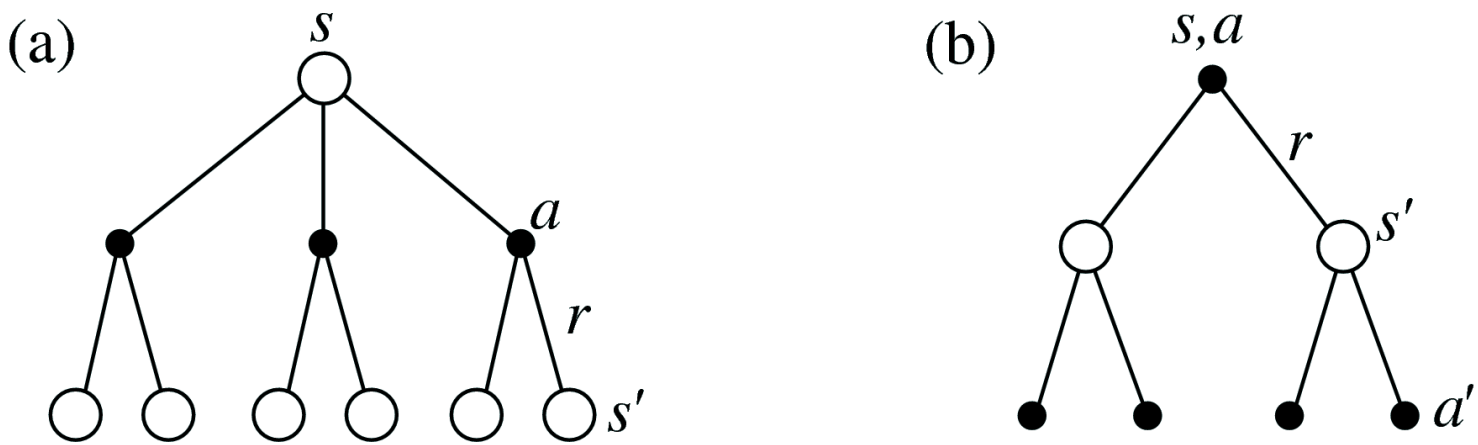
$$V^\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(s,a) \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} V^\pi(s') \right]$$

pour tout $s \in \mathcal{S}$

- L'**équation de Bellman** définit la valeur d'un état en fonction de la valeur d'autres états

Rétropropagation de la valeur

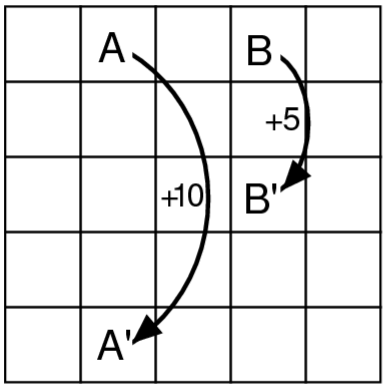
- Selon l'équation de Bellman, la valeur est retropropagée vers les états précédents
  - Tout ce qui sera étudié par la suite est fondé sur ce principe
- Diagrammes de **rétropropagation**



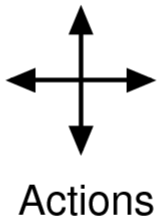
(a) pour V^π et (b) pour Q^π

Exemple de rétropropagation

- **Fonction valeur pour une politique équiprobable** (b)



(a)



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(b)

- Les actions qui devraient faire sortir l'agent du cadre échouent et une récompense de -1 est reçue
- De A, chaque action téléporte en A' avec une récompense de $+10$ (a)
- De B, chaque action téléporte en B' avec une récompense de $+5$ (b)
- Dans tous les autres cas, la récompense est nulle.

Fonction valeur optimale

- Une politique π est dite **meilleure** qu'une autre si les retours attendus en la suivant sont plus importants dans tous les états
  - Autrement dit, $\pi \geq \pi'$ ssi $\forall s \in \mathcal{S}, V^{\pi}(s) \geq V^{\pi'}(s)$
- Il existe toujours au moins une politique meilleure que les autres
- Une telle politique est dite **optimale** et on la note $\pi^{\star}$

Fonctions valeur optimales

$$\begin{aligned} V^{\star}(s) &= \max_{\pi} V^{\pi}(s) \\ Q^{\star}(s,a) &= \max_{\pi} Q^{\pi}(s,a) \\ &= E\left\{r_{t+1} + \gamma V^{\star}(s_{t+1}) \mid s_t = s, a_t = a\right\} \end{aligned}$$

Optimalité de Bellman

$$
\begin{aligned}
V^\star(s) &= \max_{a \in \mathcal{A}(s)} Q^{\pi^\star}(s,a) \\
&= \max_{a \in \mathcal{A}(s)} E_{\pi^*}\left\{ R_t \mid s_t = s, a_t = a \right\} \\
&= \max_{a \in \mathcal{A}(s)} E_{\pi^*}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right\} \\
&= \max_{a \in \mathcal{A}(s)} E_{\pi^*}\left\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s, a_t = a \right\} \\
&= \max_{a \in \mathcal{A}(s)} E\left\{ r_{t+1} + \gamma V^\star(s_{t+1}) \mid s_t = s, a_t = a \right\} \\
&= \max_{a \in \mathcal{A}(s)} \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} V^\star(s') \right]
\end{aligned}
$$

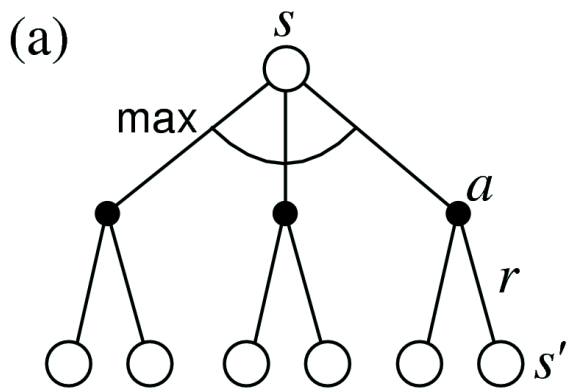
Équations de Bellman

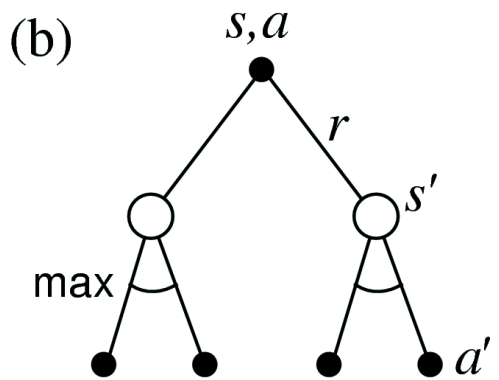
Fonction valeur optimale (a)

$$V^\star(s) = \max_{a \in \mathcal{A}(s)} \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} V^\star(s') \right]$$

Fonction action-valeur optimale (b)

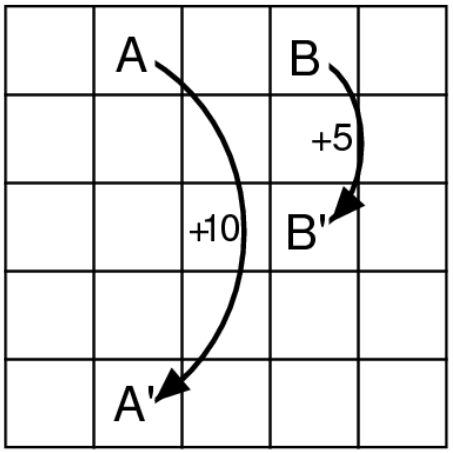
$$Q^\star(s,a) = \mathcal{R}_{sa} + \gamma \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} Q^\star(s',a')$$





Exemple de fonction valeur optimale

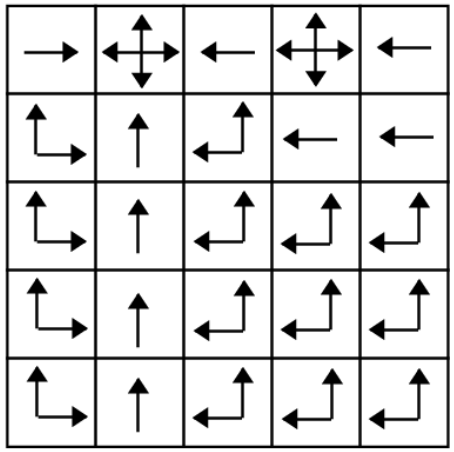
$$\gamma = 0.9$$



a) gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

b) V^*



c) π^*

- A comparer avec la fonction valeur de la politique équiprobable

Lexique

- Pas de temps : **time step**
- Récompense immédiate : **immediate reward**
- Cumul des récompenses : **cumulative reward**
- Retour attendu : **expected return**
- Tâche épisodique : **episodic task**
- Tâche continue : **continuous task**
- Retour déprécié : **discounted reward**
- Facteur de dépréciation : **discounting factor**
- État absorbant : **absorbing sate**
- Propriété de Markov : **Markov property**
- Processus de décision de markovien (PDM) : **Markov Decision Process (MDP)**
- Rétropropagation : **backup**

Plan

1. Problème d'AR
   - Introduction
   - Formalisation du problème
2. Solutions élémentaires
   - Programmation Dynamique
   - Monte Carlo
   - Différence temporelle (TD)
3. Solutions unifiées
   - Traces d'éligibilité
   - Approximation de fonctions
4. Perspectives
   - AR indirect
   - POMDP
   - Continuité du temps

PDM fini

- $\mathcal{S}$ un espace d'**états** fini, $\mathcal{A}$ un espace d'**actions fini**
- $\mathcal{T}$ une fonction de **transition**

$$\mathcal{T}_{sa}^{s'} = Pr\left\{s_{t+1} = s' \mid s_t = s, a_t = a\right\}$$

- $\mathcal{R}$ une fonction de **récompense**

$$\mathcal{R}_{sa} = E\left\{r_{t+1} \mid s_t = s, a_t = a\right\}$$

$$\begin{aligned}
V^\star(s) &= \max_{a \in \mathcal{A}(s)} E\left\{r_{t+1} + \gamma V^\star(s_{t+1}) \mid s_t = s, a_t = a\right\} \\
&= \max_{a \in \mathcal{A}(s)} \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} V^\star(s')\right] \\
Q^\star(s,a) &= E\left\{r_{t+1} + \gamma \max_{a \in \mathcal{A}(s)} Q^\star(s_{t+1}, a') \mid s_t = s, a_t = a\right\} \\
&= \mathcal{R}_{sa} + \gamma \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} Q^\star(s', a')
\end{aligned}$$

Avertissement

- En **programmation dynamique**, on suppose que $\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, et $\mathcal{R}$ sont tous connus
  - Le problème est un problème de **décision** ou d'**optimisation**
  - Ce n'est pas un problème d'**apprentissage**

Évaluation *vs.* contrôle

- Problème de l'**évaluation** d'une politique (ou **prédiction**)
  - On donne un PDM $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ et une politique π
  - On veut évaluer π en calculant V^π
- Problème du **contrôle**
  - On donne un PDM $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$
  - On veut produire une politique optimale $\pi^\star$
  - Pour ce faire, on a recours au calcul de $Q^\star$

Évaluation de la politique

- Étant donnée une politique π, et $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ étant parfaitement connu, on a un système de $|\mathcal{S}|$ équations, une par état possible

Équation de Bellman pour V^π

$$V^\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(s,a) \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} V^\pi(s') \right]$$

Résolution itérative des équations de Bellman

- On se dote d'une **suite récurrente** avec V^π pour point fixe

$$\begin{aligned} V_{k+1}(s) &= E_\pi \left\{ r_{t+1} + \gamma V_k(s_{t+1}) \mid s_t = s \right\} \\ &= \sum_{a \in \mathcal{A}(s)} \pi(s,a) \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} V_k(s') \right] \end{aligned}$$

- Méthode de résolution
  1. Initialisation arbitraire de V_0
  2. Calcul successif de $V_1, V_2, V_3, \cdots$
  3. Arrêt quand un critère de convergence est rempli
- Le plus souvent, on arrête les calculs lorsque $\max_{s \in \mathcal{S}} |V_{k+1}(s) - V_k(s)|$ est suffisamment petit

Algorithme

Évaluation itérative de la politique

entrées : un PDM $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, une politique π à évaluer

sorties : une fonction valeur V évaluant π

$\forall s \in \mathcal{S}, V(s) \leftarrow 0$

répéter

$\Delta \leftarrow 0$

pour tout $s \in S$ **faire**

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{a \in \mathcal{A}(s)} \pi(s,a) \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} V(s') \right]$

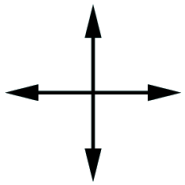
$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

fin pour

jusqu'à $\Delta < \epsilon$

Exemple d'évaluation

- Politique aléatoire



actions

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

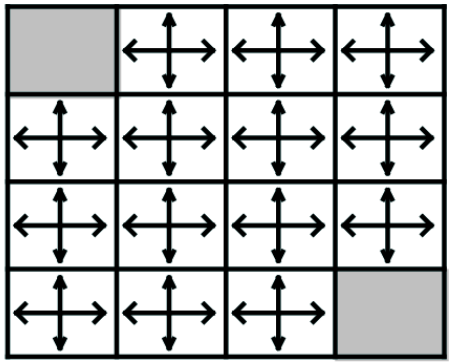
$r = -1$
on all transitions

- Politique aléatoire
- Les états non terminaux sont les états $1 \cdots 14$
- Les états grisés sont terminaux
- Les actions correspondent à des déplacements dans les 4 directions cardinales
- Les actions qui devraient mener hors du cadre laissent l'agent dans la même case

Exemple d'évaluation

$k = 0$

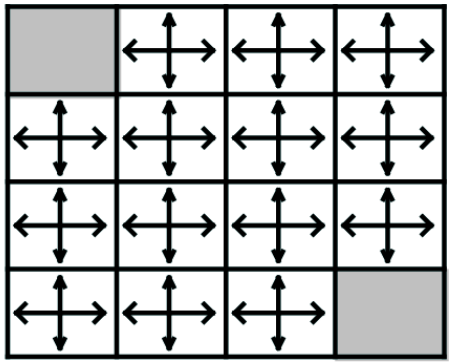
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



Exemple d'évaluation

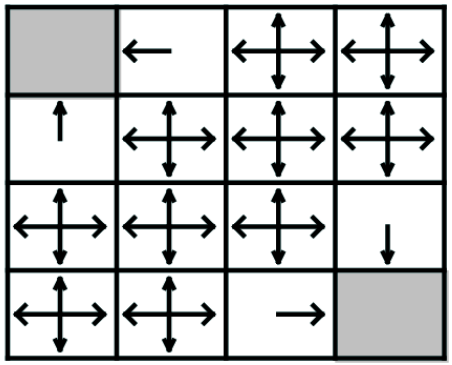
$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



$k = 1$

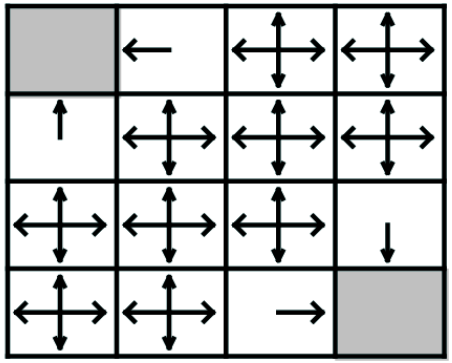
0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



Exemple d'évaluation

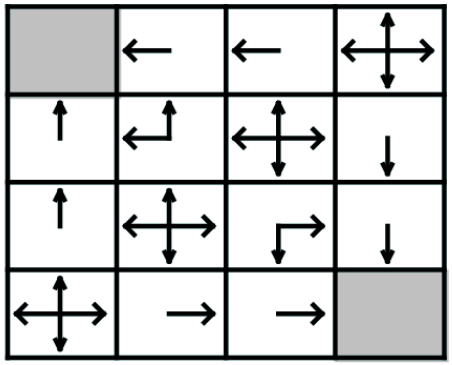
$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



$k = 2$

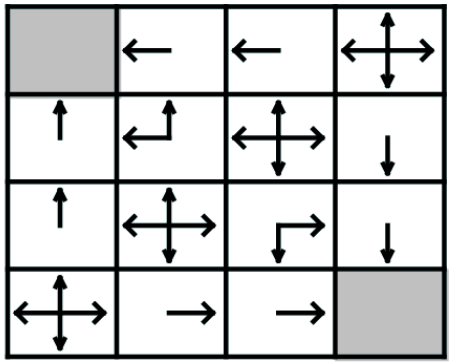
0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



Exemple d'évaluation

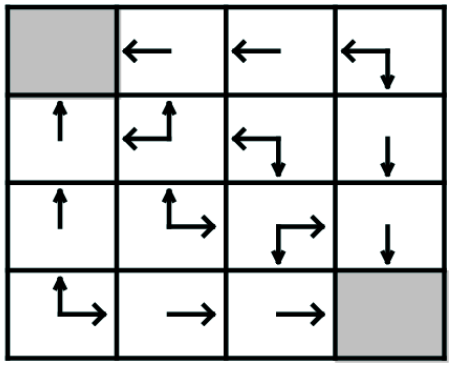
$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



$k = 3$

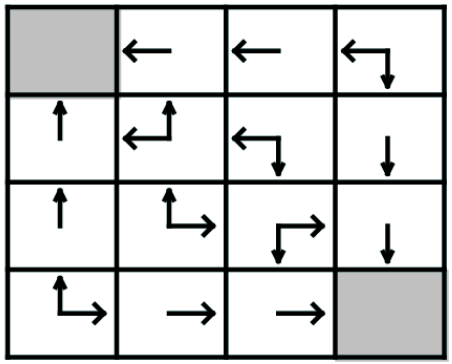
0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0



Exemple d'évaluation

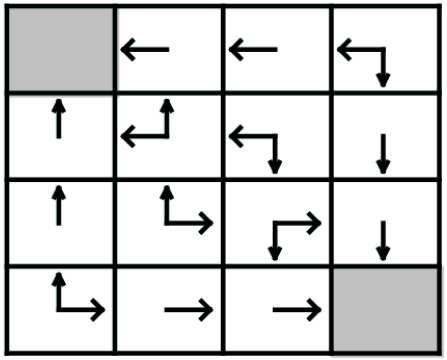
$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0



$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0



Ordre entre politiques

- **Cas déterministe** : $\pi : \mathcal{S} \rightarrow \mathcal{A}$
- Objectif de l'évaluation : l'**amélioration** de la politique
- S'il existe un a tel que $Q^{\pi}(s,a) \geq V^{\pi}(s)$ alors depuis s il vaut mieux
  - d'abord émettre a et ensuite seulement suivre π
  - plutôt que suivre π immédiatement
- Dans ce cas, il existe une politique déterministe meilleure que π, c'est celle qui en s préfère a à $\pi(s)$.
- Plus généralement,

Théorème d'amélioration de la politique

Si $\forall s \in \mathcal{S}, Q^{\pi}(s,\pi'(s)) \geq V^{\pi}(s)$, alors $\forall s \in \mathcal{S}, V^{\pi'}(s) \geq V^{\pi}(s)$, et $\pi' > \pi$

Preuve du théorème d'amélioration

$$
\begin{aligned}
V^{\pi}(s) &\leq Q^{\pi}(s, \pi'(s)) \\
&= E_{\pi'}\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s\right\} \\
&\leq E_{\pi'}\left\{r_{t+1} + \gamma Q^{\pi}(s_{t+1}, \pi'(s_{t+1})) \mid s_t = s\right\} \\
&= E_{\pi'}\left\{r_{t+1} + \gamma E_{\pi'}\left\{r_{t+2} + \gamma V^{\pi}(s_{t+2}) \mid s_{t+1}\right\} \mid s_t = s\right\} \\
&= E_{\pi'}\left\{r_{t+1} + \gamma r_{t+2} + \gamma^2 V^{\pi}(s_{t+2}) \mid s_t = s\right\} \\
&\leq E_{\pi'}\left\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V^{\pi}(s_{t+3}) \mid s_t = s\right\} \\
&\cdots \\
&\leq E_{\pi'}\left\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots \mid s_t = s\right\} \\
&= V^{\pi'}(s)
\end{aligned}
$$

Amélioration d'une politique

- Considérons une politique π et sa fonction valeur associée Q^π
- Une politique π' est **au moins aussi bonne** que π si, pour tout $s \in \mathcal{S}$,

$$\begin{aligned}
\pi'(s) &= \arg \max_{a \in \mathcal{A}(s)} Q^\pi(s,a) \\
&= \arg \max_{a \in \mathcal{A}(s)} E_\pi \left\{ r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s, a_t = a \right\} \\
&= \arg \max_{a \in \mathcal{A}(s)} \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} V^\pi(s') \right]
\end{aligned}$$

Amélioration jusqu'à l'optimalité

- Quand l'amélioration d'une politique π ne donne qu'une politique aussi bonne mais **pas strictement meilleure**, c'est que

$$V^{\pi'} = V^{\pi}$$

et donc que

$$\begin{aligned} V^{\pi'}(s) &= \max_{a \in \mathcal{A}(s)} E\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s, a_t = a\right\} \\ &= \max_{a \in \mathcal{A}(s)} \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} V^{\pi}(s')\right] \end{aligned}$$

- Ceci correspond à l'optimalité de Bellman et donc $V^{\pi'} = V^{\pi} = V^{\star}$

L'amélioration de la politique produit des politiques strictement meilleures et quand ce n'est pas le cas, c'est que la politique considérée est déjà optimale

Généralisation au cas stochastique

- Nous avons ici considéré le cas de politiques déterministes

$$\pi : \mathcal{S} \rightarrow \mathcal{A}$$

- Ces résultats se généralisent au cas des politiques **stochastiques** $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ en substituant $\sum_{a \in \mathcal{A}(s)} \pi'(s, a) Q^{\pi}(s, a)$ à $Q^{\pi}(s, \pi'(s))$

Itération de la politique

- Partant d'une politique quelconque π_0, le processus d'**itération de la politique** mène à une politique optimale $\pi^\star$

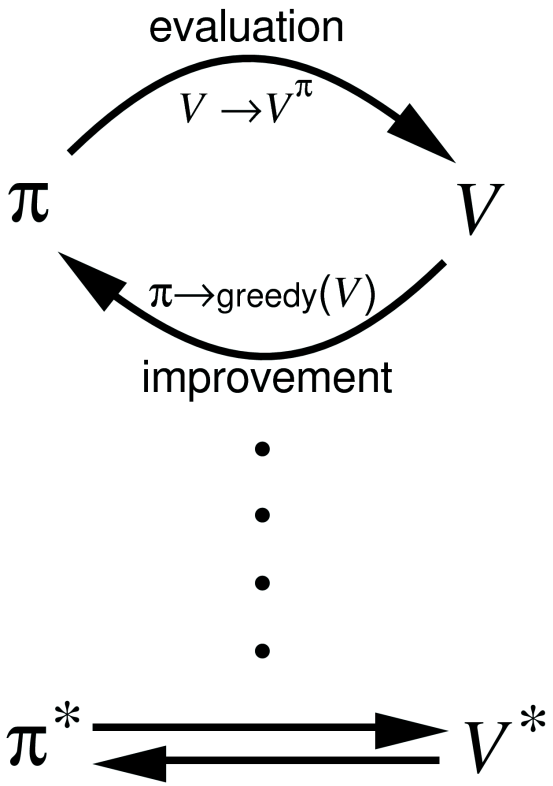
Amélioration itérative de la politique

$$\pi_0 \xrightarrow{e} V^{\pi_0} \xrightarrow{a} \pi_1 \xrightarrow{e} V^{\pi_1} \xrightarrow{a} \pi_2 \xrightarrow{e} \cdots \xrightarrow{a} \pi^\star \xrightarrow{e} V^\star$$

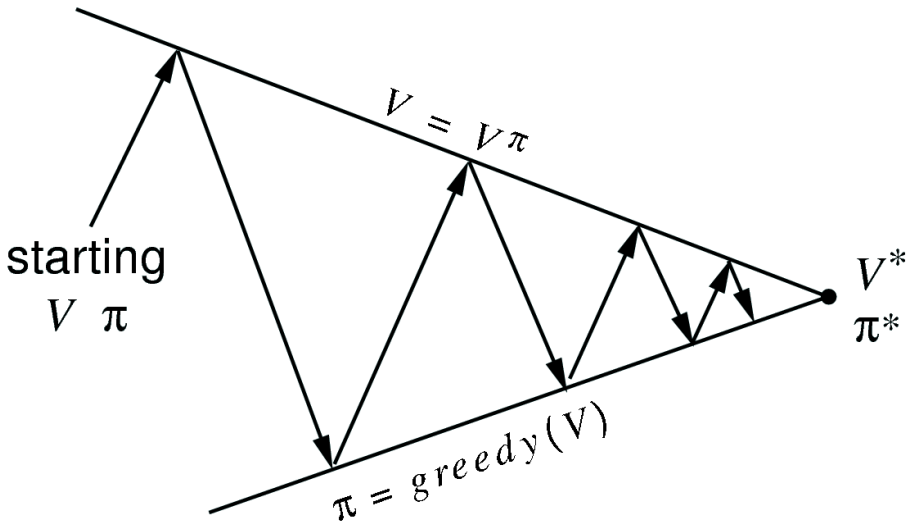
où

- $\xrightarrow{e}$ représente une **évaluation**
- $\xrightarrow{a}$ représente une **amélioration**

Itération de la politique



Itération de la politique



Algorithme

Itération de la politique

entrées : un PDM $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$
sorties : une politique π, une fonction valeur V

initialiser π aléatoirement
répéter
  répéter
    $\Delta \leftarrow 0$
    pour tout $s \in \mathcal{S}$ **faire**
      $v \leftarrow V(s)$
      $V(s) \leftarrow \mathcal{R}_{s\pi(s)} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}^{s'}_{s\pi(s)} V(s')$
      $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    fin pour
  jusqu'à $\Delta < \epsilon$
  pour tout $s \in \mathcal{S}$ **faire**
    $b \leftarrow \pi(s)$
    $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}^{s'}_{sa} V^{\pi}(s') \right]$
    $stable \leftarrow [b = \pi(s)]$
  fin pour
jusqu'à *stable*

Itération de la valeur

- L'**itération de la politique** nécessite plusieurs évaluations
  - Chaque évaluation est itérative et converge vers V^π **exactement**
- Or il n'est pas nécessaire de toujours produire **exactement** V^π
  - On peut en fait arrêter l'évaluation après la première itération
  - Dans ce cas, on produit successivement $V_0 \rightarrow V_1 \rightarrow V_2 \cdots$

$$
\begin{aligned}
V_{k+1}(s) &= \max_{a \in \mathcal{A}(s)} E\left\{ r_{t+1} + \gamma V_k(s_{t+1}) \mid s_t = s, a_t = a \right\} \\
&= \max_{a \in \mathcal{A}(s)} \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} V_k(s') \right]
\end{aligned}
$$

pour tout $s \in \mathcal{S}$

- L'itération de la valeur converge vers $V^\star$ (si elle existe) à partir de n'importe quelle fonction valeur initiale V_0

Algorithme

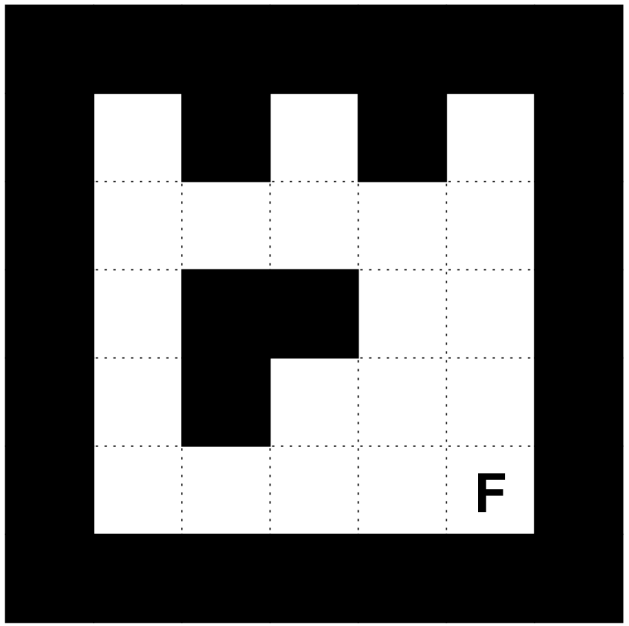
Itération de la valeur

entrées : un PDM $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$
sorties : une fonction valeur optimale V, une politique optimale π

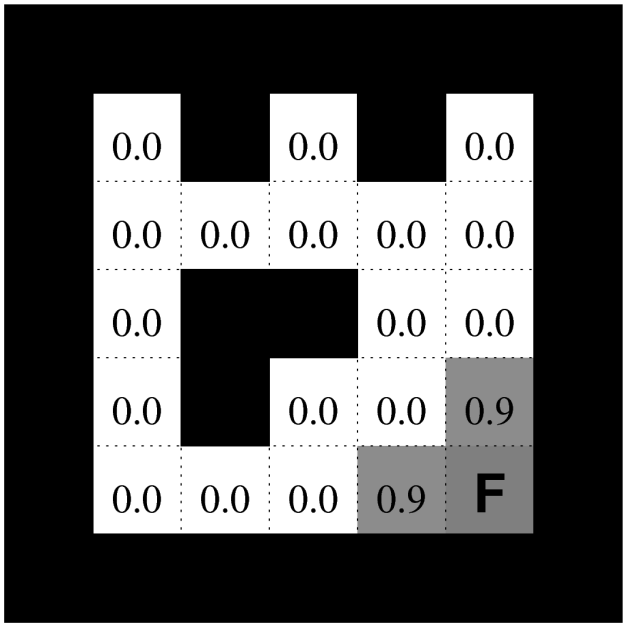
initialiser V arbitrairement
répéter
  $\Delta \leftarrow 0$
  pour tout $s \in \mathcal{S}$ **faire**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_{a \in \mathcal{A}(s)} \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} V(s') \right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  fin pour
jusqu'à $\Delta < \epsilon$
pour tout $s \in \mathcal{S}$ **faire**
  $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} \left[\mathcal{R}_{sa} + \gamma \sum_{s' \in \mathcal{S}^+} \mathcal{T}_{sa}^{s'} V(s') \right]$
fin pour

Exemple

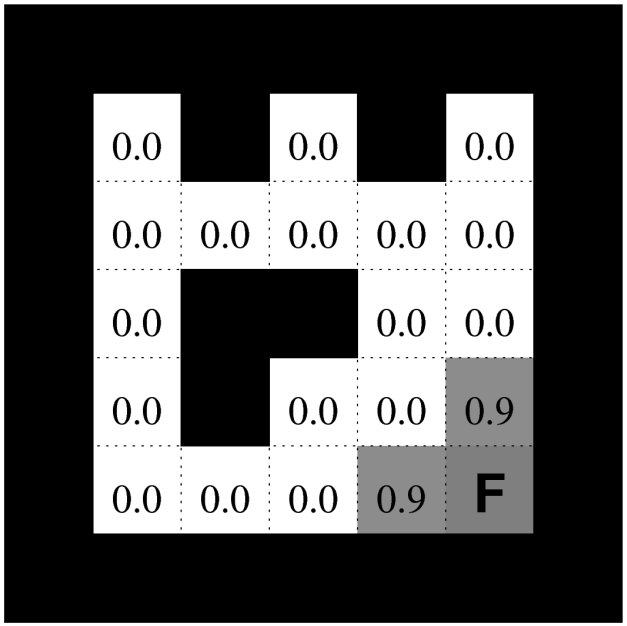
- Actions : N, S, E, O
- Transitions déterministes faisant changer de case sauf en cas de mur
- Récompense de 0.9 à l'arrivée dans la case marquée F

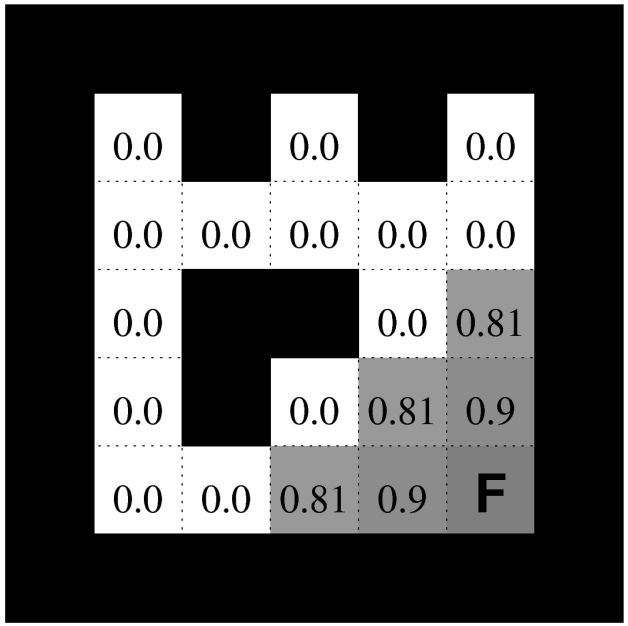


Exemple

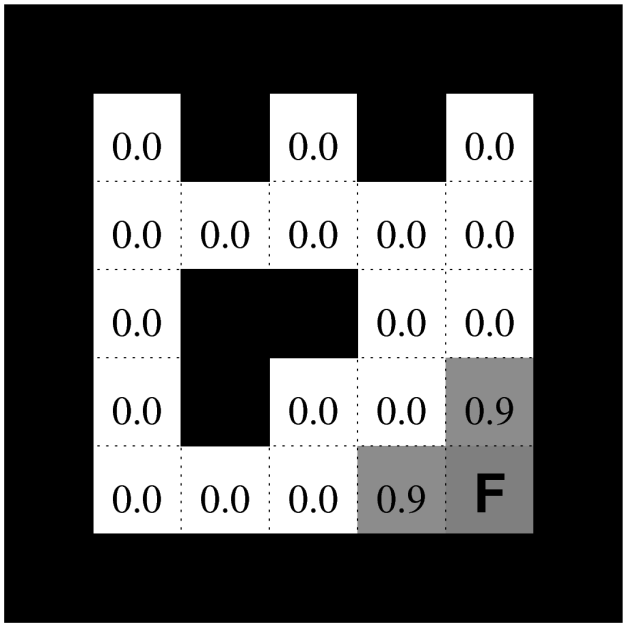


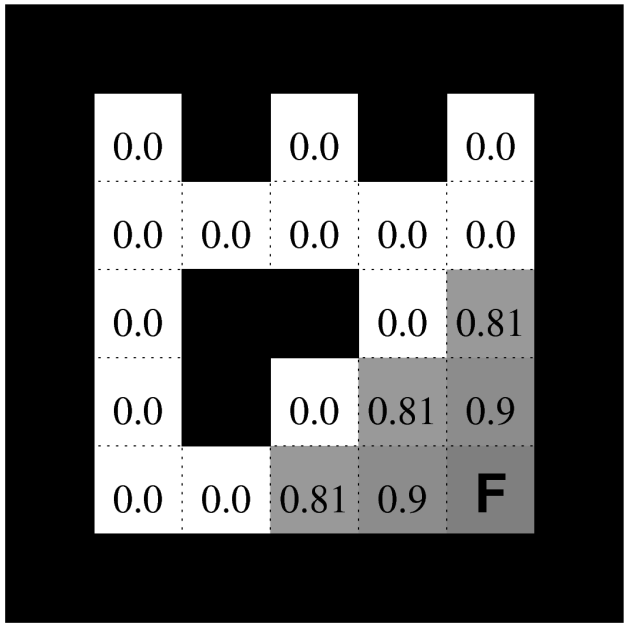
Exemple

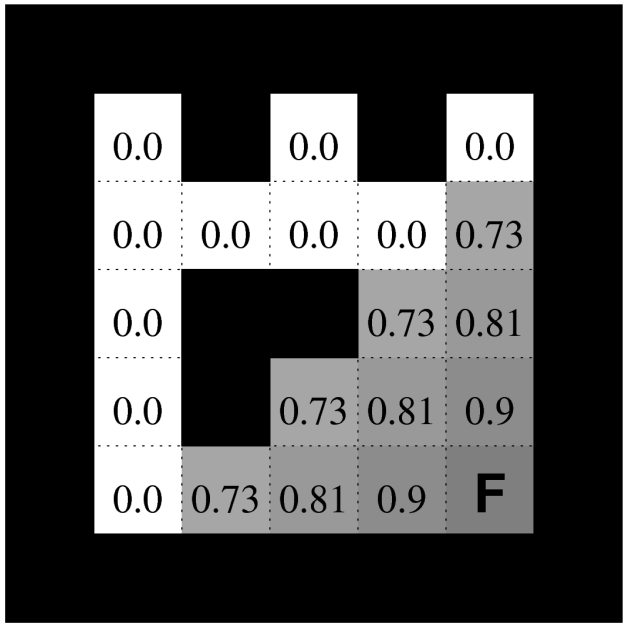

0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.9
0.0 0.0 0.0 0.9 F


0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.81
0.0 0.0 0.81 0.9
0.0 0.0 0.81 0.9 F

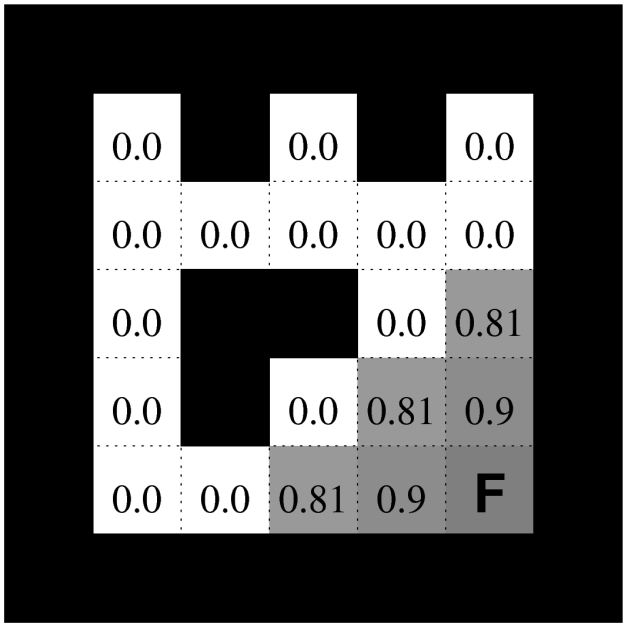
Exemple

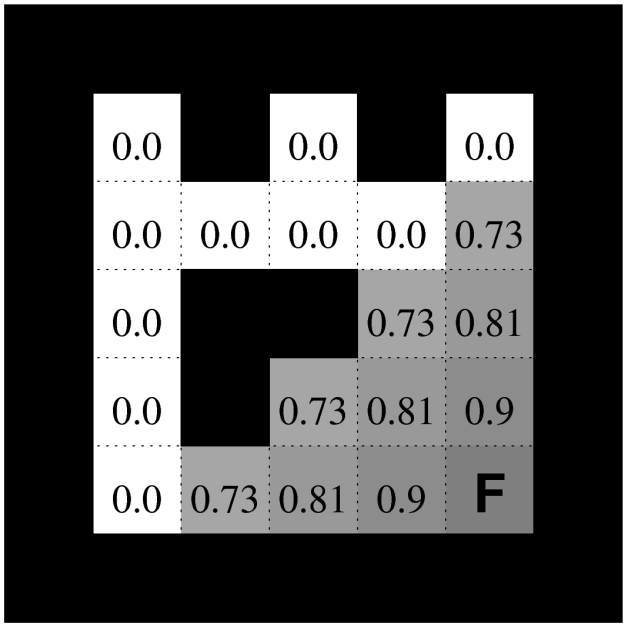

0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0 0.9
0.0 0.0 0.0 0.9 F

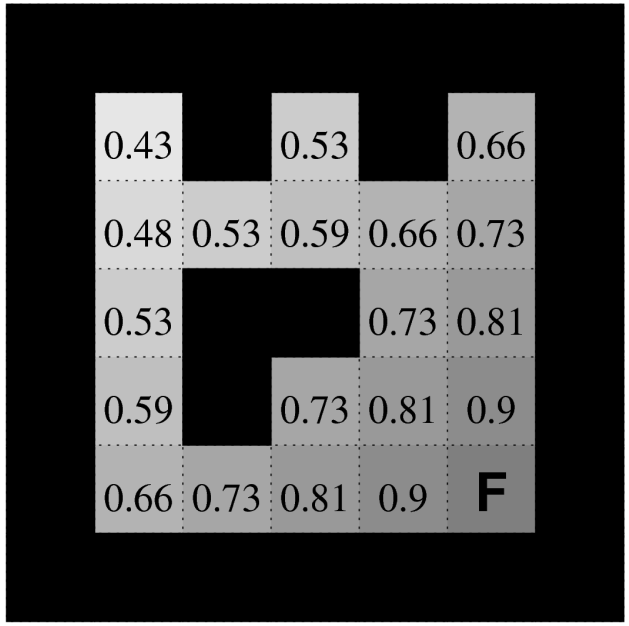

0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.81
0.0 0.0 0.81 0.9
0.0 0.0 0.81 0.9 F


0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.73
0.0 0.73 0.81
0.0 0.73 0.81 0.9
0.0 0.73 0.81 0.9 F

Exemple


0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.81
0.0 0.0 0.81 0.9
0.0 0.0 0.81 0.9 F


0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.73
0.0 0.73 0.81
0.0 0.73 0.81 0.9
0.0 0.73 0.81 0.9 F


0.43 0.53 0.66
0.48 0.53 0.59 0.66 0.73
0.53 0.73 0.81
0.59 0.73 0.81 0.9
0.66 0.73 0.81 0.9 F

Lexique

- Police déterministe : **deterministic policy**
- Police stochastique : **stochastic policy**
- Amélioration de la politique : **policy improvement**
- Itération de la politque : **policy iteration**
- Itération de la valeur : **value iteration**

Décision et apprentissage

- Dans cette partie, nous avons considéré que le PDM $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ était connu
  - Les problèmes abordés étaient des problèmes de **décision** et d'**optimisation**
- Or dans un problème d'AR, l'environnement est inconnu au départ

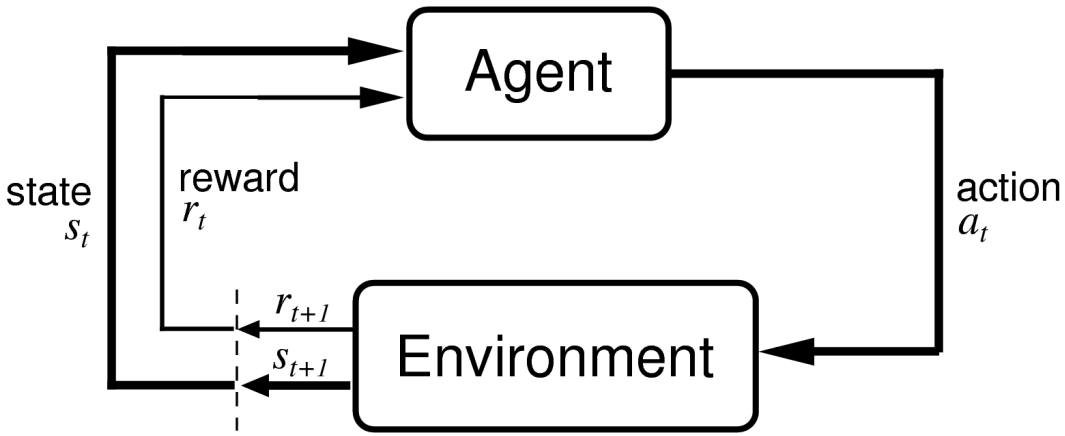
Dans la suite du cours

- En AR, on étudie l'**apprentissage** de la politique et de la fonction valeur
- Sans connaissance de $\mathcal{T}$ ni $\mathcal{R}$ au départ
- On utilise l'interaction avec l'environnement pour compenser ce manque d'information
- Cette interaction nous donne des t-uples (s_t, a_t, s_{t+1}) et (s_t, a_t, r_{t+1})

Plan

1. Problème d'AR
   - Introduction
   - Formalisation du problème
2. **Solutions élémentaires**
   - Programmation Dynamique
   - Monte Carlo
   - Différence temporelle (TD)
3. Solutions unifiées
   - Traces d'éligibilité
   - Approximation de fonctions
4. Perspectives
   - AR indirect
   - POMDP
   - Continuité du temps

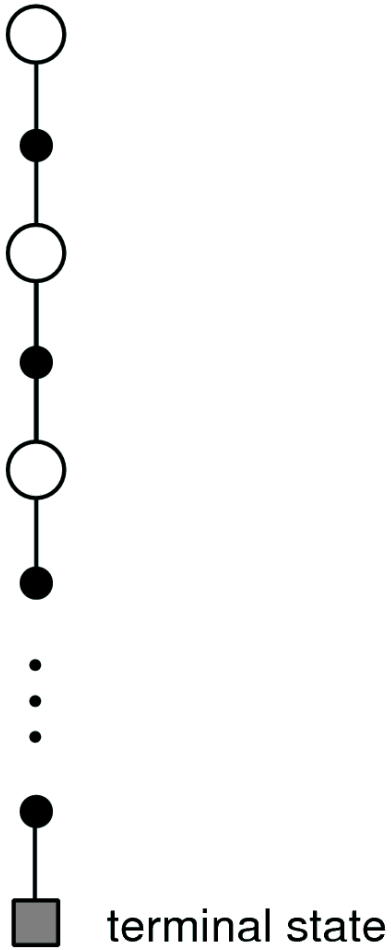
Recueil d'expérience



- L'agent ne connaît ni $\mathcal{T}$ ni $\mathcal{R}$
- En revanche, il **interagit** avec l'environnement
  - A chaque pas de temps, recueil de r_t, s_t, a_t.
- Les méthodes de Monte Carlo stockent des **séquences complètes** $s_0, a_0, r_1, s_1, a_1, \cdots, r_T, s_T$ (ou trajectoires)
- On réalise un apprentissage sur ces données observées
  - Avantage : on prend en compte l'environnement tel qu'il est, pas tel qu'on se le représente

Évaluation MC

- On dispose d'une politique π pas nécessairement optimale
- On cherche à l'évaluer en calculant V^{π}
- On dispose de trajectoires réelles



Algorithme

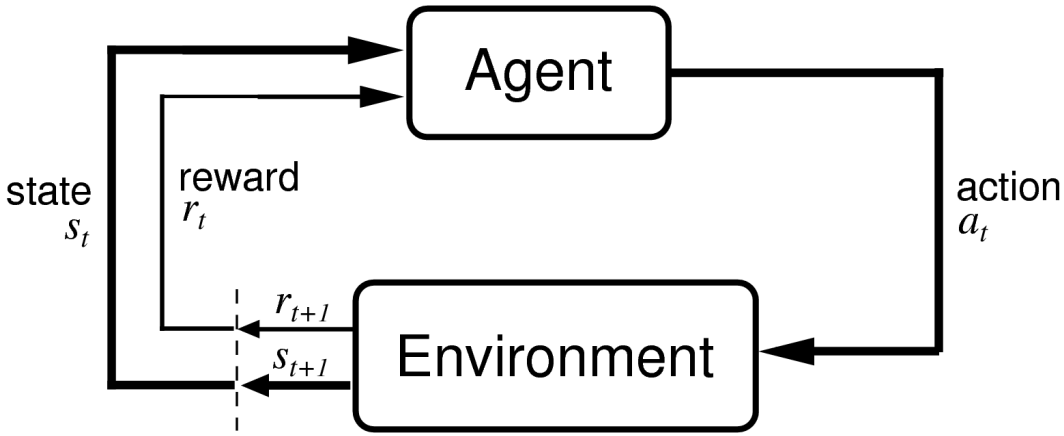
Évaluation MC de la politique (première visite)

entrées : politique π à évaluer, espaces $\langle \mathcal{S}, \mathcal{A} \rangle$
sorties : fonction valeur V de la politique

initialiser V arbitrairement
$\forall s \in \mathcal{S}, L_R(s) \leftarrow \emptyset$
boucle
  générer une trajectoire en suivant π
  pour tout $s \in \mathcal{S}$ **faire**
    $R \leftarrow$ retour suivant la première occurrence de s
    $L_R(s) \leftarrow L_R(s) \cup \{R\}$
    $V(s) \leftarrow moyenne(L_R(s))$
  fin pour
fin boucle

Contrôle MC

- On cherche une politique optimale $\pi^\star$ en interagissant avec l'environnement
- **Technique** : on part d'une politique π
  1. On interagit avec l'environnement et on évalue π en calculant Q^π
  2. On améliore π en se fondant sur Q^π et on recommence



$$\pi_0 \xrightarrow{e} Q^{\pi_0} \xrightarrow{a} \pi_1 \xrightarrow{e} Q^{\pi_1} \xrightarrow{a} \pi_2 \xrightarrow{e} \cdots \xrightarrow{a} \pi^\star \xrightarrow{e} Q^\star$$

Amélioration de la politique

- Pour l'amélioration de la politique, on choisit les actions de manière **déterministe** et **gloutonne**

$$\pi(s) = \arg \max_{a \in \mathcal{A}(s)} Q(s,a)$$

- Le **théorème d'amélioration de la politique** s'applique puisque, pour tout $s \in \mathcal{S}$

$$\begin{aligned} Q^{\pi_k}(s, \pi_{k+1}(s)) &= Q^{\pi_k}(s, \arg \max_{a \in \mathcal{A}(s)} Q^{\pi_k}(s,a)) \\ &= \max_{a \in \mathcal{A}(s)} Q^{\pi_k}(s,a) \\ &\geq Q^{\pi_k}(s, \pi_k(s)) \\ &= V^{\pi_k}(s) \end{aligned}$$

Algorithme

Contrôle MC

entrées : espaces $\langle \mathcal{S}, \mathcal{A} \rangle$
sorties : politique optimale π, fonction valeur Q associée

initialiser Q et π arbitrairement
$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, L_R(s,a) \leftarrow \emptyset$
boucle
  générer une trajectoire en suivant π
  pour tout $(s,a) \in \mathcal{S} \times \mathcal{A}$ **faire**
    $R \leftarrow$ retour suivant la première occurrence de (s,a)
    $L_R(s,a) \leftarrow L_R(s,a) \cup \{R\}$
    $Q(s,a) \leftarrow moyenne(L_R(s,a))$
  fin pour
  pour tout s de l'épisode **faire**
    $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} Q(s,a)$
  fin pour
fin boucle

Hypothèse forte

- Nécessité d'avoir des **actions exploratoires** en début d'épisodes

Politique stochastique

- Pour lever la nécessité d'**actions exploratoires** en début d'épisodes, il faut revenir sur la sélection **déterministe** et **gloutonne** des actions
  - Politiques stochastiques : telles que pour tous $s \in \mathcal{S}$ et $a \in \mathcal{A}(s)$, $\pi(s, a) > 0$
- Politiques ϵ-**greedy**
  - Probabilité minimale pour les **actions non gloutonnes** : $\frac{\epsilon}{|\mathcal{A}(s)|}$
  - Probabilité de l'**action gloutonne** : $1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}$

Amélioration de la politique dans le cas ϵ-greedy

- La politique est ϵ-**greedy** donc $\forall a \in \mathcal{A}, \pi(s,a) > \frac{\epsilon}{|\mathcal{A}(s)|}$

$$
\begin{aligned}
Q^\pi(s,\pi'(s)) &= \sum_{a\in\mathcal{A}(s)} \pi'(s,a) Q^\pi(s,a) \\
&= \frac{\epsilon}{|\mathcal{A}(s)|} \sum_{a\in\mathcal{A}(s)} Q^\pi(s,a) + (1-\epsilon) \max_{a\in\mathcal{A}(s)} Q^\pi(s,a) \\
&\geq \frac{\epsilon}{|\mathcal{A}(s)|} \sum_{a\in\mathcal{A}(s)} Q^\pi(s,a) + (1-\epsilon) \sum_{a\in\mathcal{A}(s)} \frac{\pi(s,a) - \frac{\epsilon}{|\mathcal{A}(s)|}}{1-\epsilon} Q^\pi(s,a) \\
&= \frac{\epsilon}{|\mathcal{A}(s)|} \sum_{a\in\mathcal{A}(s)} Q^\pi(s,a) - \frac{\epsilon}{|\mathcal{A}(s)|} \sum_{a\in\mathcal{A}(s)} Q^\pi(s,a) \\
&\quad + \sum_{a\in\mathcal{A}(s)} \pi(s,a) Q^\pi(s,a) \\
&= V^\pi(s)
\end{aligned}
$$

- Donc $\forall s \in \mathcal{S}, V^{\pi'}(s) \geq V^\pi(s)$, d'où $\pi' \geq \pi$ et le **théorème d'amélioration de la politique s'applique**

Algorithme

Contrôle MC stochastique

entrées : espaces $\langle \mathcal{S}, \mathcal{A} \rangle$
sorties : politique ϵ-soft π, fonction valeur Q associée

initialiser Q et π arbitrairement
$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, L_R(s,a) \leftarrow \emptyset$
boucle
générer une trajectoire en suivant π
pour tout $(s,a) \in \mathcal{S} \times \mathcal{A}$ **faire**
$R \leftarrow$ retour suivant la première occurrence de (s,a)
$L_R(s,a) \leftarrow L_R(s,a) \cup \{R\}$
$Q(s,a) \leftarrow moyenne(L_R(s,a))$
fin pour
pour tout s de l'épisode **faire**
$a^\star \leftarrow \arg\max_{a \in \mathcal{A}(s)} Q(s,a)$

$$\forall a \in \mathcal{A}, \pi(s,a) \leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|} & \text{si } a = a^\star \\ \frac{\epsilon}{|\mathcal{A}(s)|} & \text{sinon} \end{cases}$$

fin pour
fin boucle

Plan

1. Problème d'AR
   - Introduction
   - Formalisation du problème
2. **Solutions élémentaires**
   - Programmation Dynamique
   - Monte Carlo
   - Différence temporelle (TD)
3. Solutions unifiées
   - Traces d'éligibilité
   - Approximation de fonctions
4. Perspectives
   - AR indirect
   - POMDP
   - Continuité du temps

De MC à TD

- Avec MC, enregistrer des trajectoires permet de calculer R_t

$$V(s_t) \leftarrow V(s_t) + \alpha \left[R_t - V(s_t) \right]$$

- La cible pour l'apprentissage de $V(s_t)$ est R_t. En conséquence

$$\begin{aligned} V^\pi(s) &= E_\pi \left\{ R_t \mid s_t = s \right\} \\ &= E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right\} \\ &= E_\pi \left\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s \right\} \\ &= E_\pi \left\{ r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s \right\} \end{aligned}$$

- On peut donc aussi cibler $r_{t+1} + \gamma V(s_{t+1})$ et alors

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right]$$

Différence temporelle (TD)

- Comme MC, apprentissage à partir de l'**expérience**

Mise à jour TD

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right]$$

- TD ne nécessite pas de conserver des trajectoires complètes
  - Des tuples $(s_t, a_t, r_{t+1}, s_{t+1})$ suffisent
- TD est une méthode d'**amorçage**
  - On utilise une estimation pour en produire une nouvelle

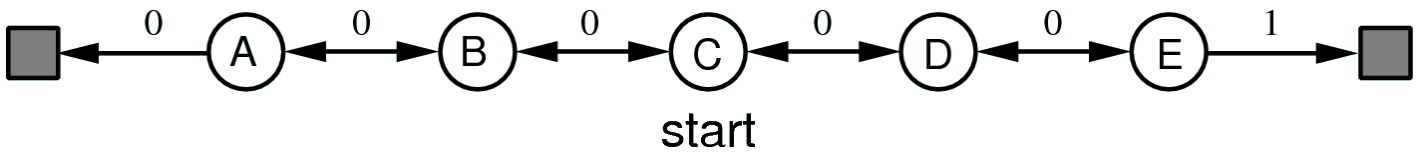
Algorithme pour l'évaluation

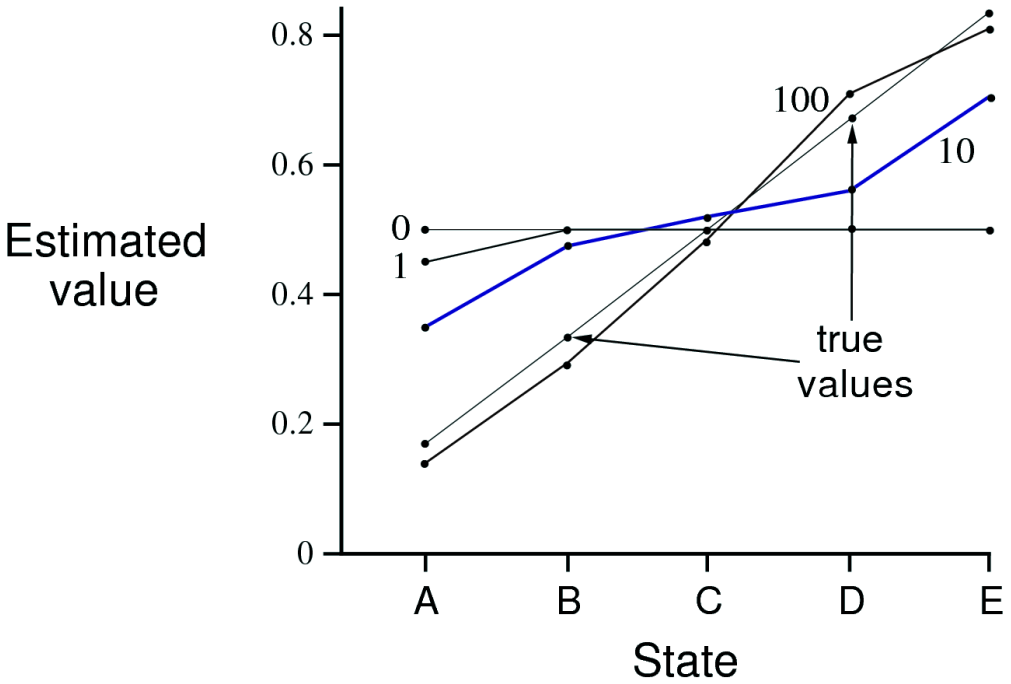
Évaluation TD tabulaire

entrées : espaces $\langle \mathcal{S}, \mathcal{A} \rangle$, politique π
sorties : fonction valeur V associée à π

initialiser V arbitrairement
boucle
initialiser l'état initial s de l'épisode
répéter
$a \leftarrow \pi(s)$
émettre a
recevoir la récompense immédiate r et le nouvel état s'
$V(s) \leftarrow V(s) + \alpha \left[r + \gamma V(s') - V(s) \right]$
$s \leftarrow s'$
jusqu'à s terminal
fin boucle

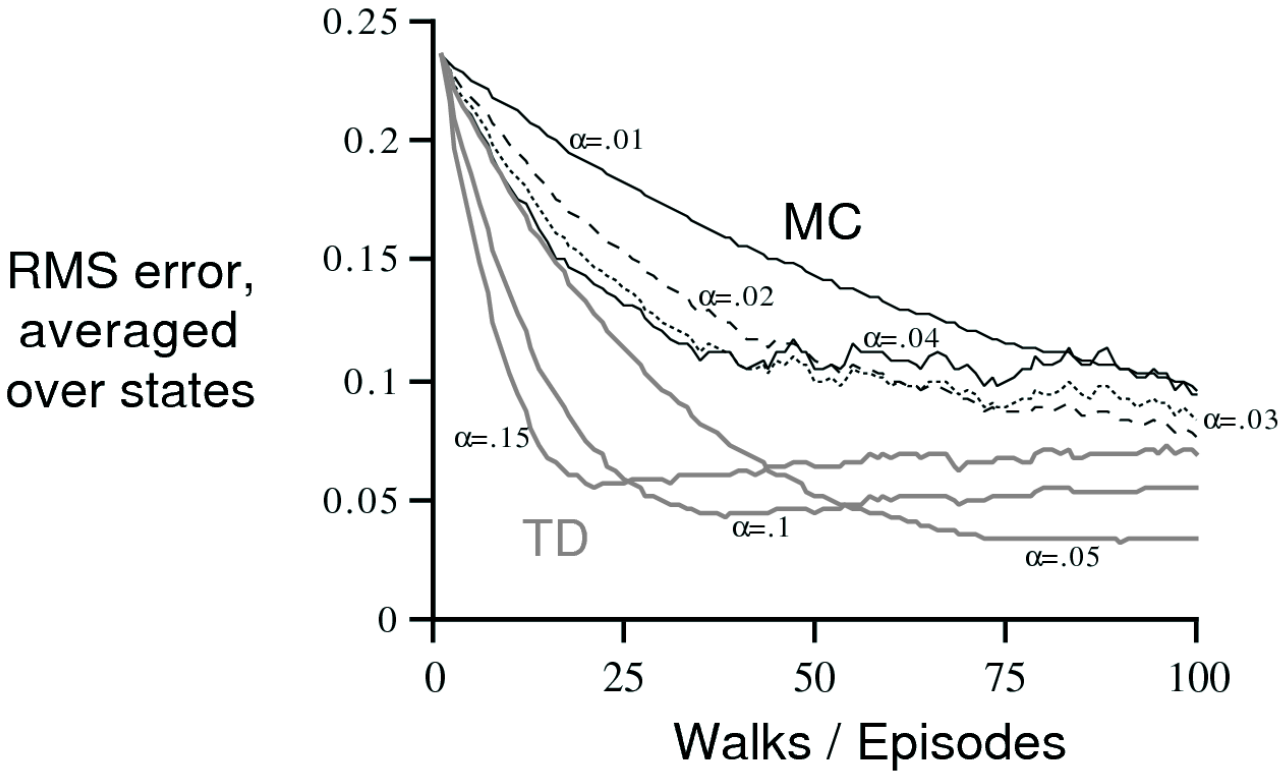
Exemple





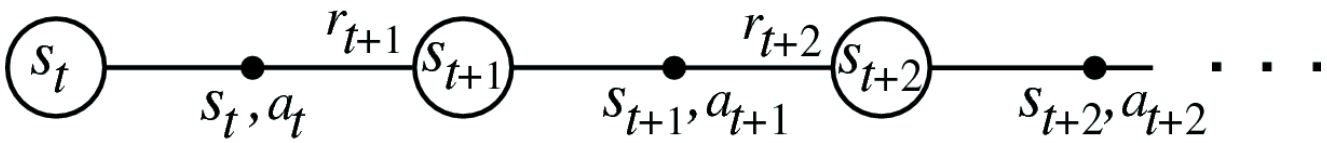
- Expérience de **marche aléatoire**
  - États A, B, C, D et E
  - Actions permettant de passer entre deux états contigus
  - Politique aléatoire
  - $\alpha = 0.1$

Comparaison expérimentale TD/MC



- Résultats moyens sur 100 séquences

Contrôle TD



- Apprentissage de Q^π plutôt que V^π pour répondre au problème du contrôle

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

Algorithme

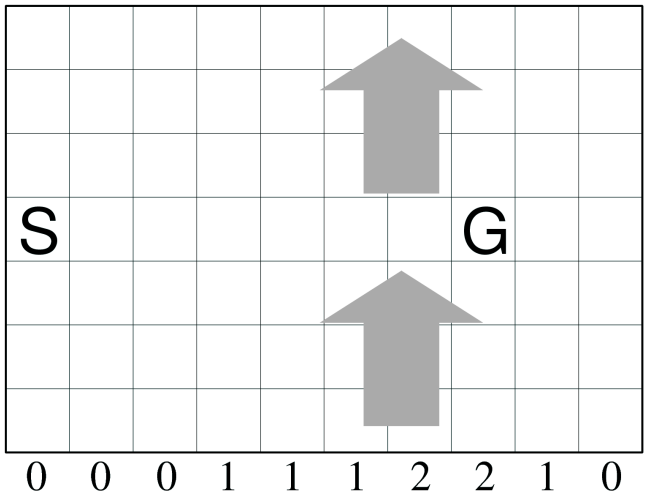
Sarsa

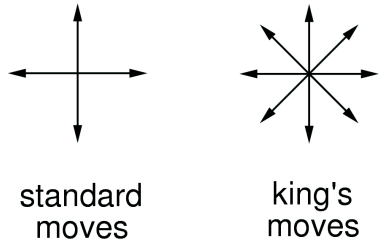
entrées : espaces $\langle \mathcal{S}, \mathcal{A} \rangle$
sorties : fonction valeur Q, actions émises

initialiser Q arbitrairement
boucle
  initialiser l'état initial s de l'épisode
  $a \leftarrow \pi^{\epsilon}_{Q}(s)$ (ϵ-greedy)
  répéter
    émettre a ; observer r et s'
    $a' \leftarrow \pi^{Q}_{\epsilon}(s')$ (ϵ-greedy)
    $Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma Q(s',a') - Q(s,a)]$
    observer le nouvel état s
    $s \leftarrow s'$; $a \leftarrow a'$
  jusqu'à s terminal
fin boucle

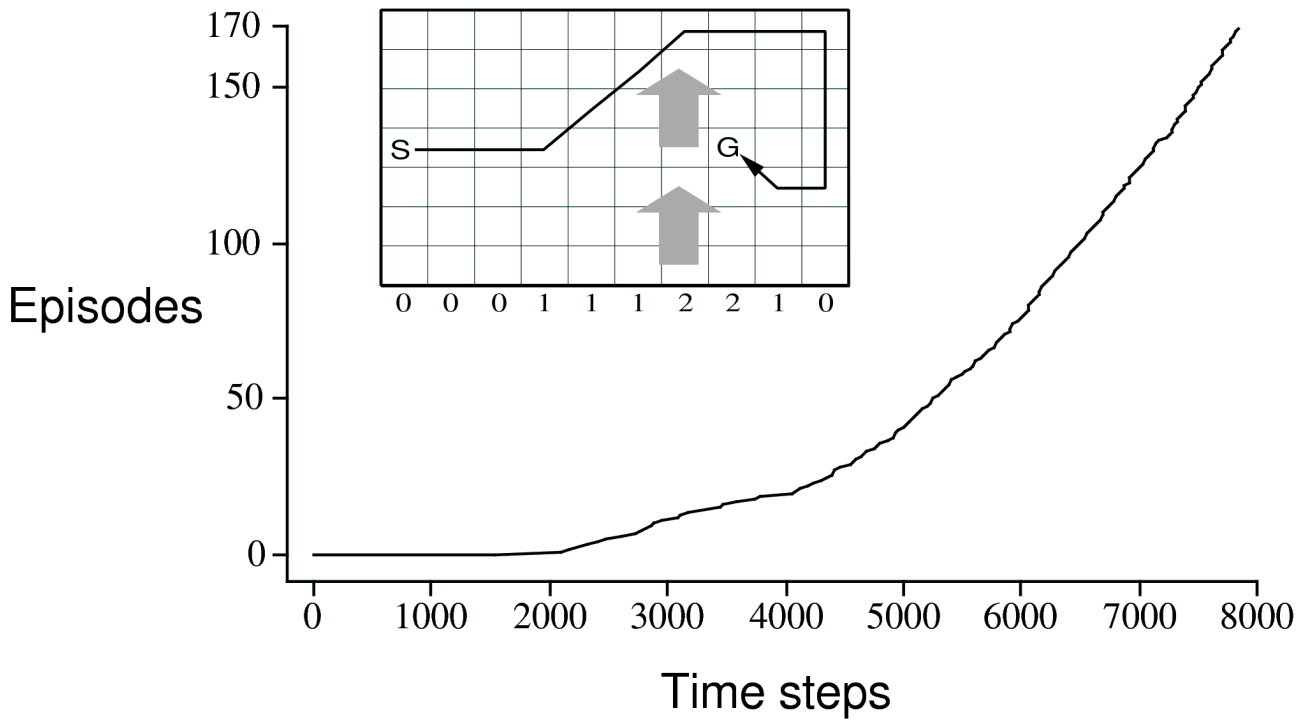
Exemple

- Les actions sont influencées par un « vent » localisé
- Une récompense de -1 est allouée à chaque pas de temps jusqu'à l'arrivée





Résultats expérimentaux

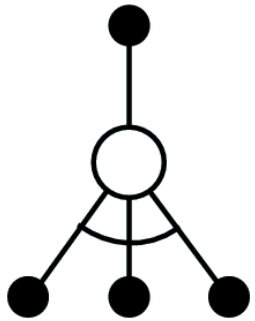


Apprentissage en ligne et hors ligne

- **Apprentissage du contrôle en ligne**
  - Ces méthodes évaluent et améliorent la politique effectivement utilisée pour les décisions
  - **Exemples** : les algos MC vus dans ce cours, Sarsa...

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

- **Apprentissage du contrôle hors ligne**
  - La politique suivie est séparée de la politique évaluée
    - Une politique **comportementale** est utilisée pour agir
    - Une politique **évaluée** est utilisée pour l'évaluation



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

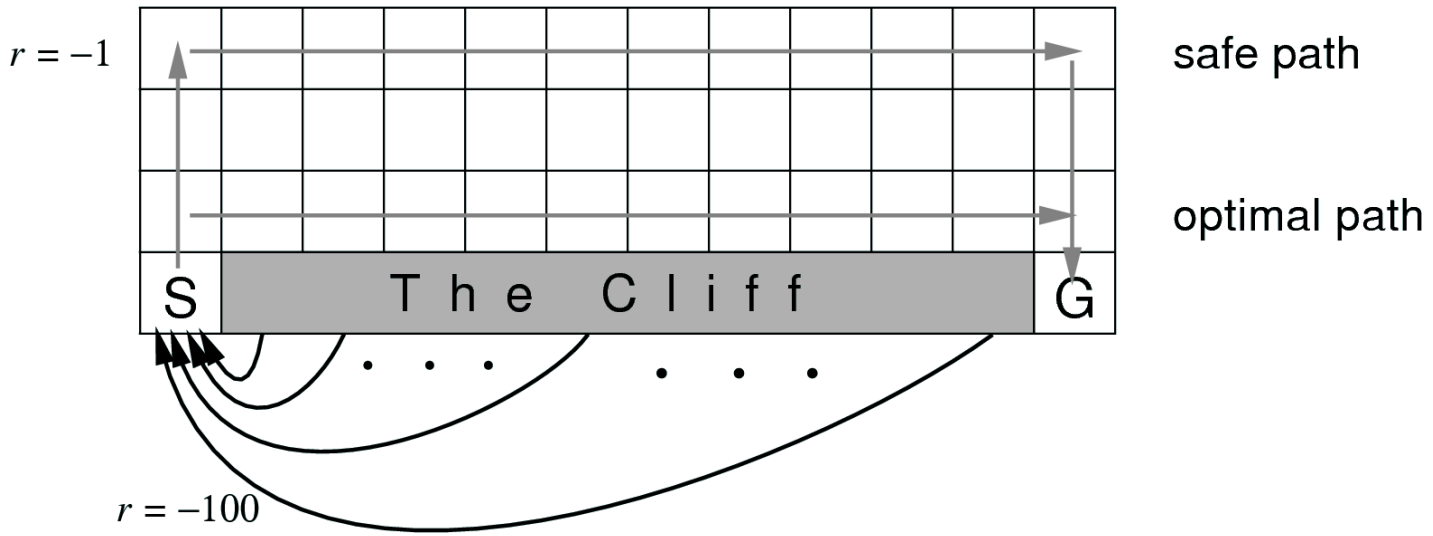
Algorithme TD hors ligne

Q-learning

entrées : espaces $\langle \mathcal{S}, \mathcal{A} \rangle$
sorties : fonction valeur Q, actions émises

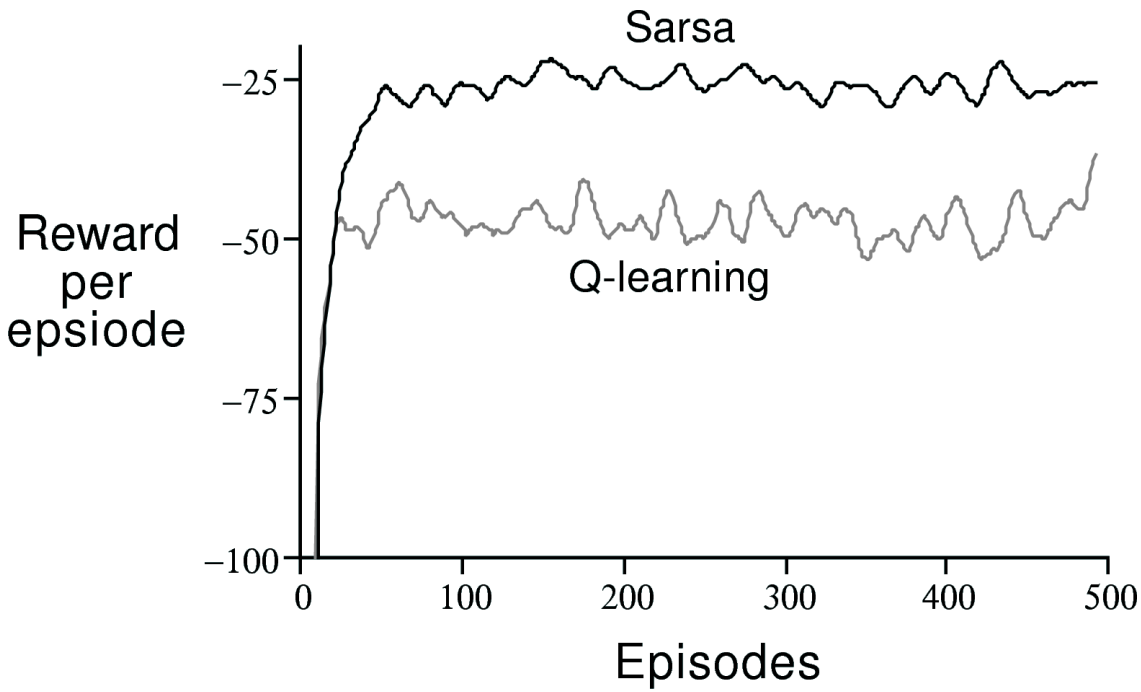
initialiser Q arbitrairement
boucle
initialiser l'état initial s de l'épisode
répéter
$a \leftarrow \pi^{\epsilon}_{Q}(s)$ (ϵ-greedy)
émettre a ; observer r et s'
$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s',a') - Q(s,a) \right]$
$s \leftarrow s'$
jusqu'à s terminal
fin boucle

Exemple



- Marche sur la falaise
  - Monde de cases normal
  - Tomber de la falaise occasionne une récompense de -100
  - Les autres mouvements occasionnent des récompenses de -1

Résultats expérimentaux



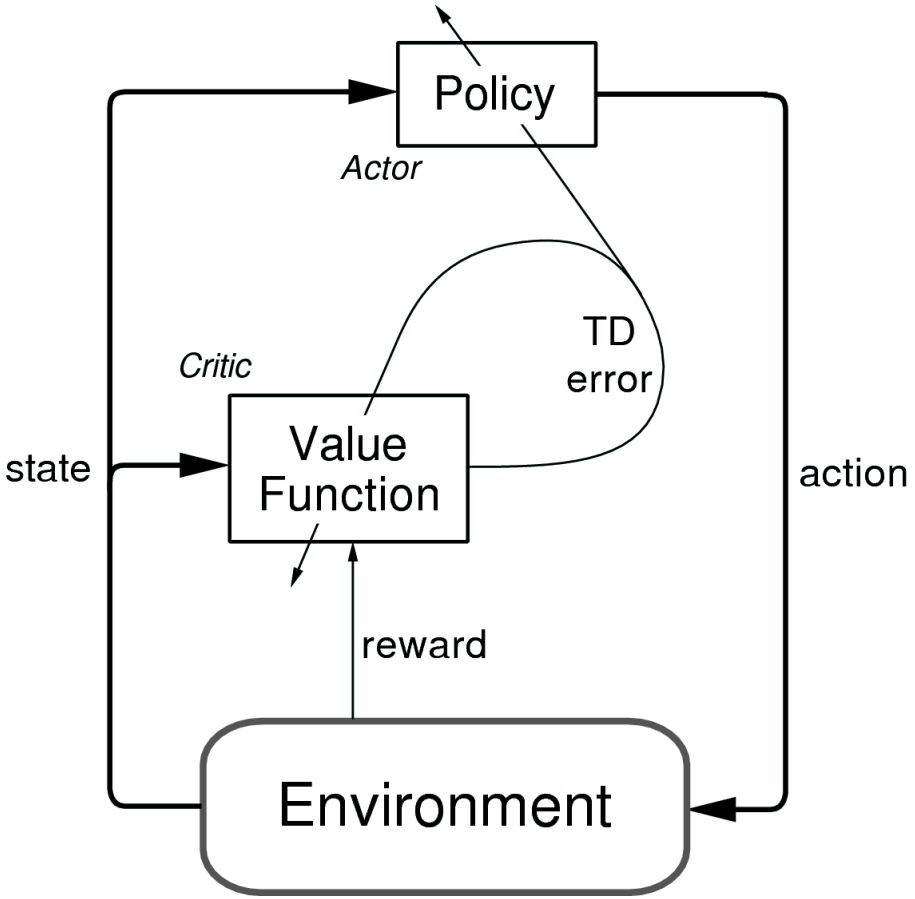
Architecture acteur/critique (A/C)

- Un **Acteur** sélectionne les actions
- Un **Critique** critique les décisions de l'acteur
  - Sur la base de l'erreur TD

Erreur TD

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

- L'apprentissage est toujours **en ligne**



Erreur TD

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

- **Si** $\delta_t > 0$, la tendance à choisir a_t quand s_t est observé devra être augmentée
- **Sinon**, cette tendance devrait être diminuée

Exemple A/C avec sélection Softmax

- Supposons que les actions sont choisies selon une méthode **softmax**

$$\pi(s,a) = Pr\{a_t = a \mid s_t = s\} = \frac{e^{p(s,a)}}{\sum_{b \in \mathcal{A}(s)} e^{p(s,b)}}$$

- Tendance à augmenter/diminuer la tendance à choisir a_t :

$$p(s_t, a_t) \leftarrow p(s_t, a_t) + \alpha\delta_t$$

avec $\beta \in [0, 1]$

- Si on voulait faire varier ce changement de tendance en fonction de la probabilité de choisir a_t, on pourrait avoir :

$$p(s_t, a_t) \leftarrow p(s_t, a_t) + \alpha\delta_t \left[1 - \pi_t(s_t, a_t)\right]$$

Lexique

- Différence temporelle : **temporal difference**
- Méthode d'amorçage : **bootstrap method**
- Apprentissage en ligne de la politique : **on-policy learning**
- Apprentissage hors-ligne de la politique : **off-policy learning**
- Politique comportementale : **behavior policy**
- Politique évaluée : **estimation policy**
- Acteur/critique : **actor/critic**
- Erreur TD : **TD error**

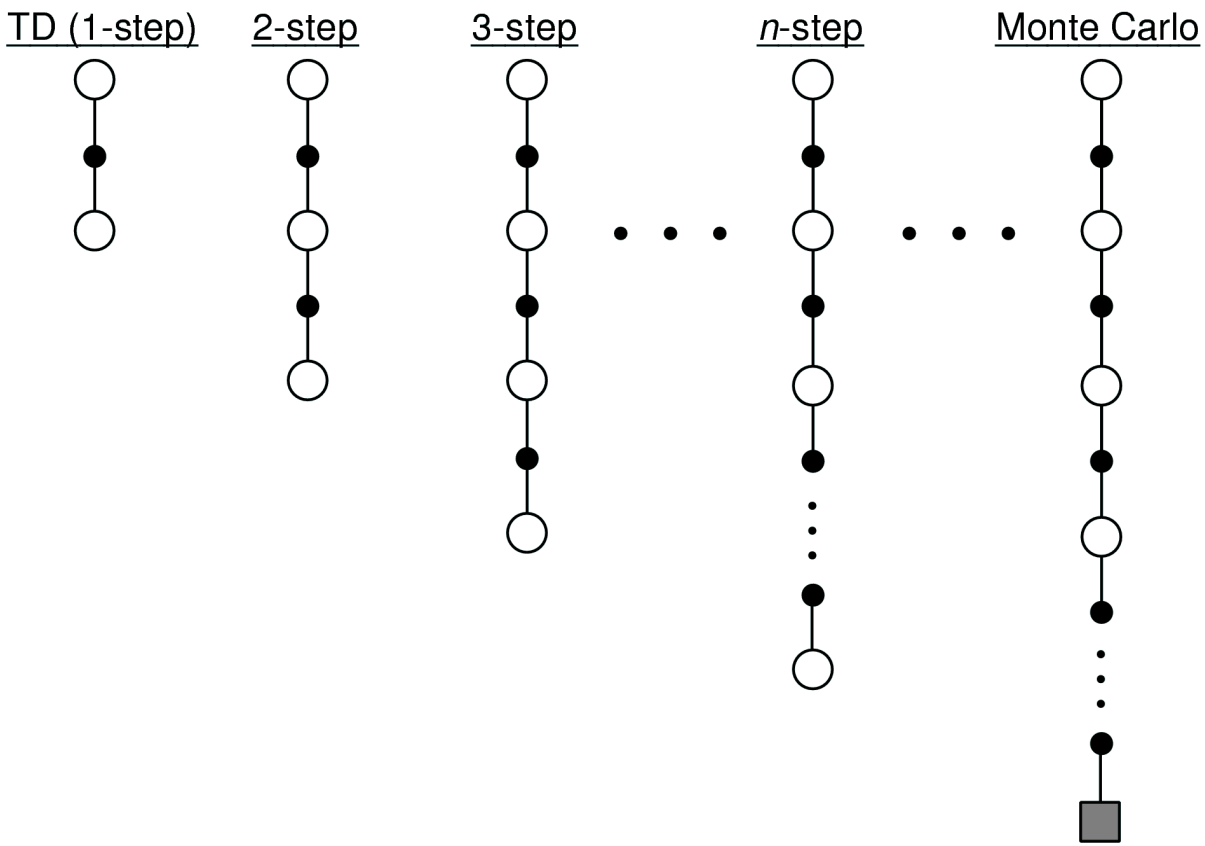
Plan

1. Problème d'AR
   - Introduction
   - Formalisation du problème
2. Solutions élémentaires
   - Programmation Dynamique
   - Monte Carlo
   - Différence temporelle (TD)
3. **Solutions unifiées**
   - Traces d'éligibilité
   - Approximation de fonctions
4. Perspectives
   - AR indirect
   - POMDP
   - Continuité du temps

Techniques d'AR unifiées

- **Objectif** : unification de
  - Monte Carlo
  - Différence temporelle
- Utilisation de **traces d'éligibilité** λ
  - Possibilité de reproduire exactement MC et TD
  - En outre, possibilité d'utiliser des **méthodes intermédiaires** souvent meilleures que TD ou MC seules

Rétropropagation à n pas de temps



- Nécessité de conserver en mémoire une trace des n pas de temps précédents

Évaluation par n-rétropropagation

- Avec MC, on utilise une mémoire des T pas de temps précédents

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T$$

- Avec TD, on ne conserve qu'un seul pas de temps et

$$R_t^{(1)} = r_{t+1} + \gamma V_t(s_{t+1})$$

- On peut poursuivre et généraliser à une n-rétropropagation :

$$R_t^{(2)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 V_t(s_{t+2})$$

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n})$$

Erreur sur l'évaluation de la politique

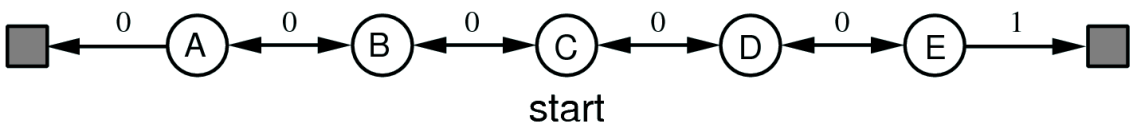
$$\Delta V_t(s_t) = \alpha \left[R_t^{(n)} - V_t(s_t) \right]$$

- Rétropropagation **en ligne** : immédiatement
  $V_{t+1}(s) = V_t(s) + \Delta V_t(s)$
- Rétropropagation **hors ligne** : après T pas de temps
  $V_{t+1}(s) = V_0(s) + \sum_{t=0}^{T-1} \Delta V_t(s)$

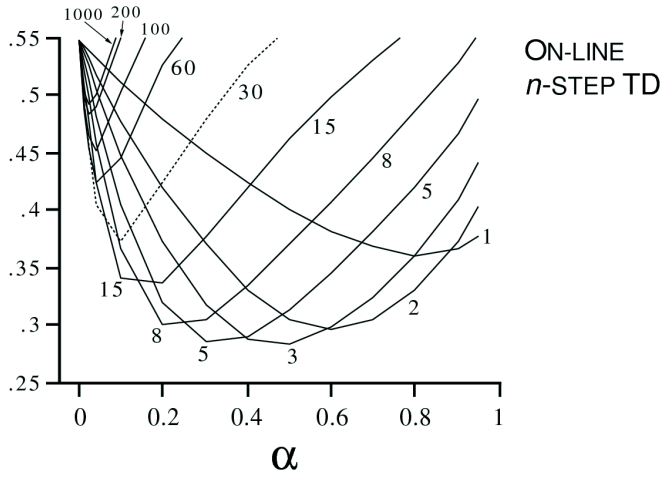
Convergence : propriété de réduction de l'erreur

$$\forall V, \max_{s \in \mathcal{S}} \left| E\left\{ R_t^{(n)} \mid s_t = s \right\} - V^\pi(s) \right| \leq \gamma^n \max_{s \in \mathcal{S}} |V(s) - V^\pi(s)|$$

Exemple

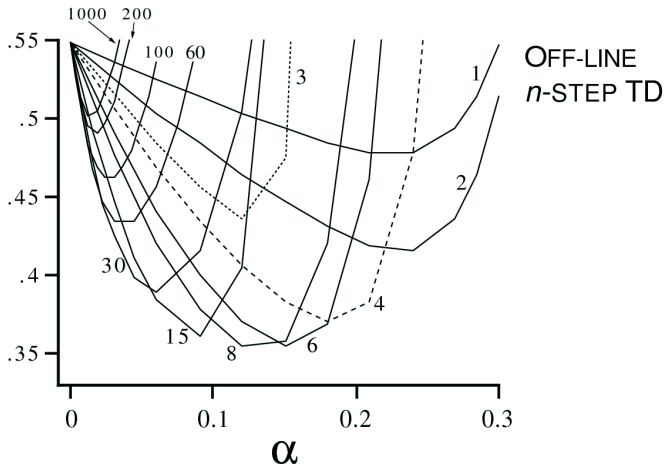


RMS error, averaged over first 10 episodes



ON-LINE
n-STEP TD

RMS error, averaged over first 10 episodes



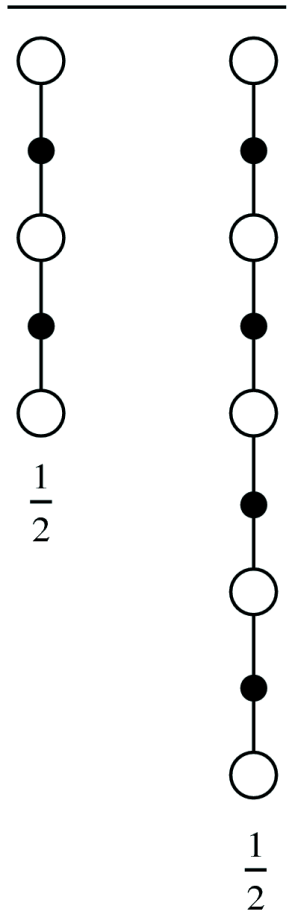
OFF-LINE
n-STEP TD

- **Marche aléatoire** avec 19 états au lieu de A, B, C, D et E
- Quel n choisir ?
- Solutions avec n intermédiaire meilleures que les solutions extrêmes TD et MC

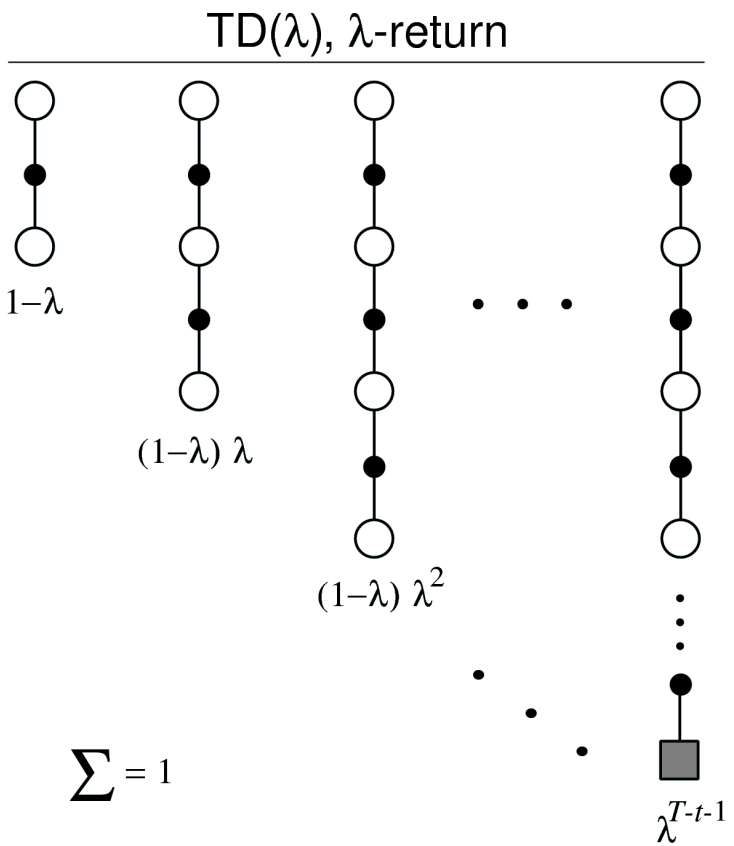
Moyenne de plusieurs n-rétropropagations

- Jusqu'ici : **rétropropagation** à n pas de temps
- Possibilité d'envisager des moyennes de n-rétropropagations
- **Exemple** : moyenne entre des rétropropagations à 2 et à 4 pas de temps :

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



Généralisation à TD(λ)

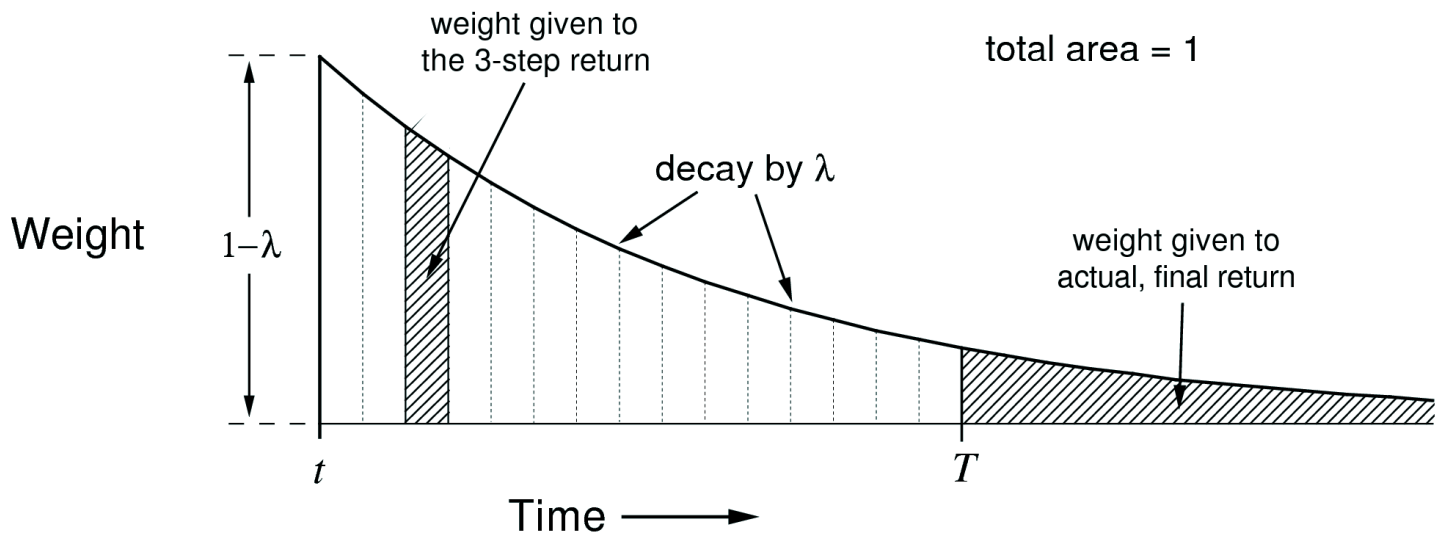


- Avec $\lambda \in [0,1]$, $(1-\lambda)$ étant un facteur de normalisation, on a $R_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)} + \lambda^{T-t-1} R_t$ avec un état terminal, et plus généralement on a :

$$R_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)}$$

- **Si** $\lambda = 0$: Différence temporelle
- **Si** $\lambda = 1$: Monte Carlo
- TD(λ) généralise TD et Monte Carlo

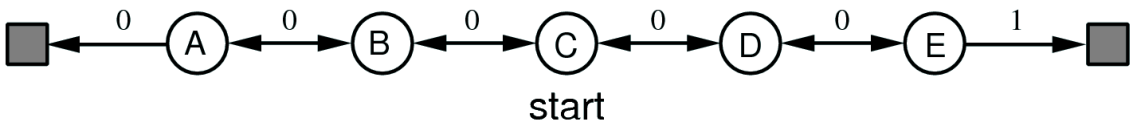
Algorithmes TD(λ)



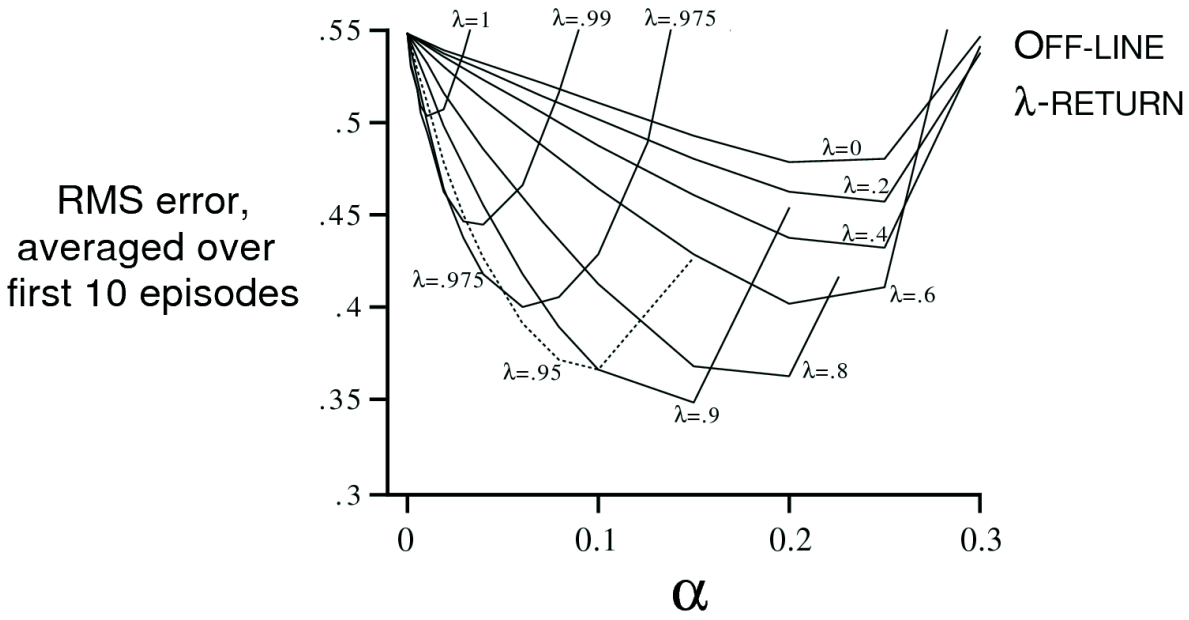
- On appelle **retour-λ** la valeur $R_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)}$
- Les algorithmes **TD(λ)** utilisent la correction de valeur suivante :

$$\Delta V_t(s_t) = \alpha \left[R_t^\lambda - V_t(s_t) \right]$$

Exemple

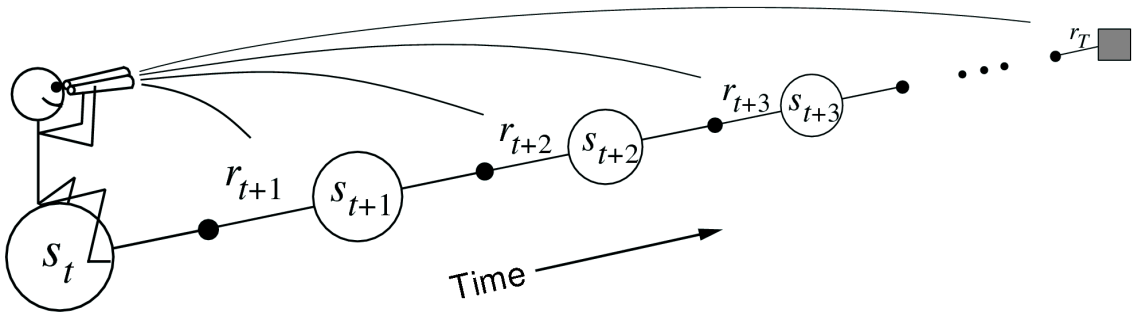


- **Marche aléatoire** avec 19 états au lieu de A, B, C, D et E
- Quel λ choisir ?
- Solutions avec λ intermédiaire meilleures que les solutions extrêmes TD et MC



Vue théorique et vue algorithmique du TD(λ)

- Nous venons de présenter TD(λ) selon sa vue **théorique** dite « **en avant** »

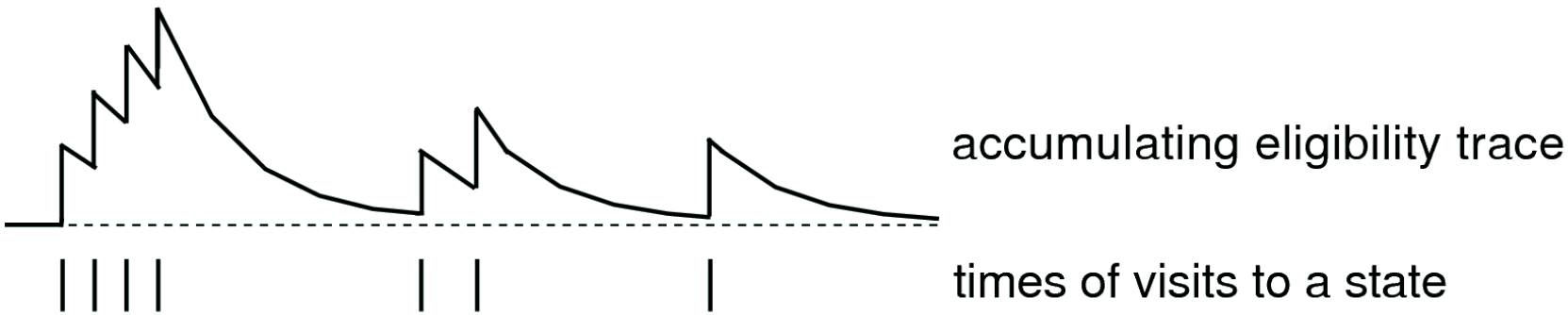


- Nous allons présenter TD(λ) de manière plus **algorithmique** selon une vue « **en arrière** »

Traces d'éligibilité

- Dans la vue « en **arrière** » du TD(λ), une variable de mémoire $e(s) \in \mathbb{R}^+$ est associée à chaque état s
- Cette trace d'éligibilité $e(s)$ est augmentée dès que s est observé, et à chaque pas de temps elle décroît sinon.

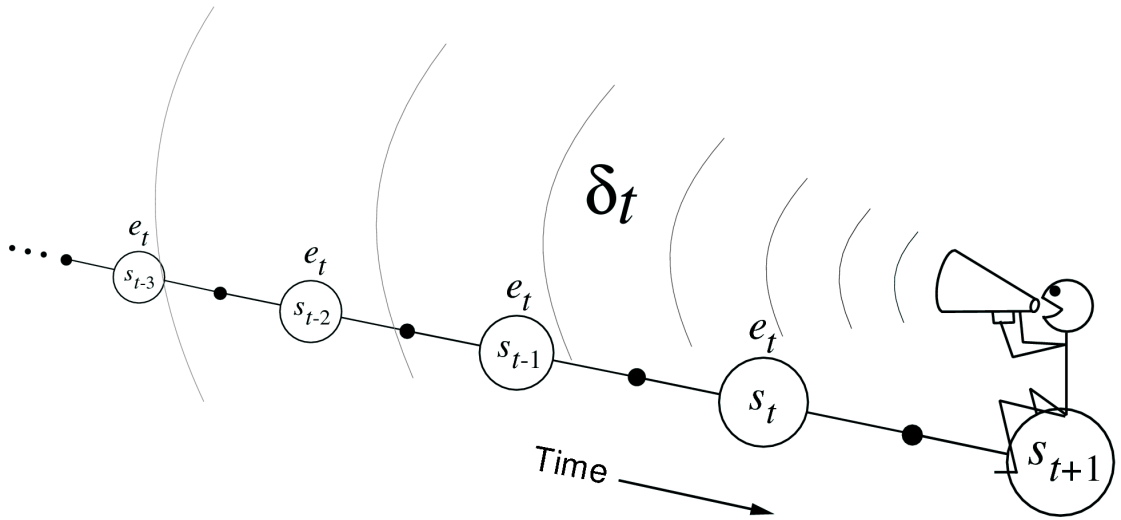
$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) + 1 & \text{si } s = s_t \\ \gamma\lambda e_{t-1}(s) & \text{sinon} \end{cases}$$



Vue « en arrière »

- Avec TD(λ), on utilise l'erreur $\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)$
- Toutefois, cette erreur sera plus ou moins prise en compte selon que l'état a été visité il y a longtemps ou pas (longtemps en termes de son **éligibilité** $e(s)$)

$$\Delta V_t(s) = \alpha \delta_t e_t(s)$$



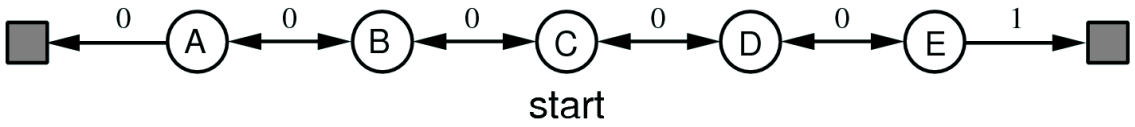
Algorithme pour l'évaluation d'une politique

Évaluation TD(λ)

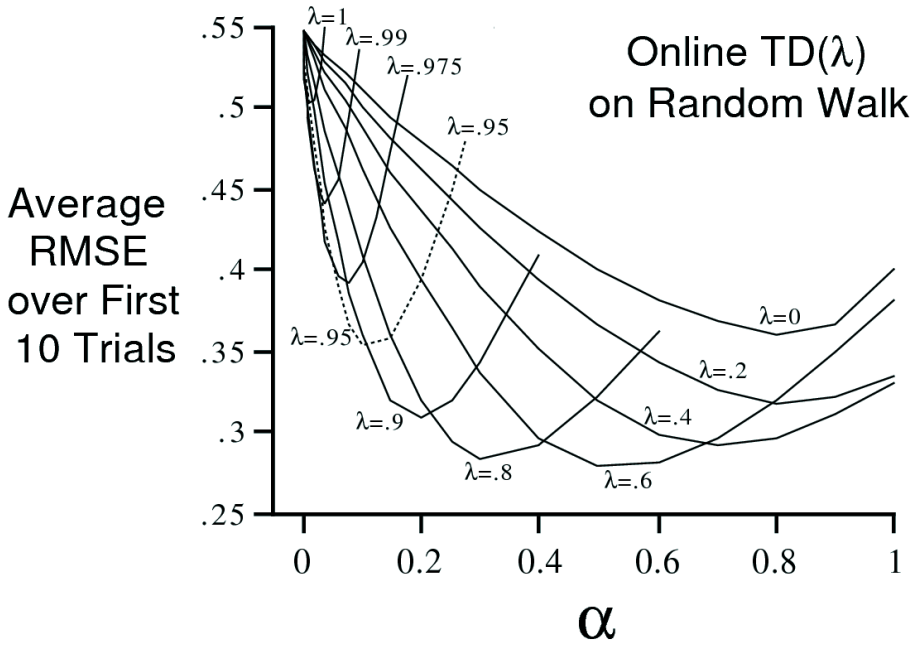
entrées : espaces $\langle \mathcal{S}, \mathcal{A} \rangle$, politique π
sorties : fonction valeur V associée à π

initialiser V arbitrairement
boucle
  $\forall s \in \mathcal{S}, e(s) \leftarrow 0$
  initialiser l'état initial s de l'épisode
  répéter
    $a \leftarrow \pi(s)$
    émettre a ; recevoir r et s'
    $\delta \leftarrow r + \gamma V(s') - V(s)$
    $e(s) \leftarrow e(s) + 1$
    pour tout $s \in \mathcal{S}$ **faire**
      $V(s) \leftarrow V(s) + \alpha \delta e(s)$
      $e(s) \leftarrow \gamma \lambda e(s)$
    fin pour
    $s \leftarrow s'$
  jusqu'à s terminal
fin boucle

Équivalence « en avant » et « en arrière »



- **Marche aléatoire** avec 19 états au lieu de A, B, C, D et E
- Algorithme d'évaluation TD(λ) « en arrière »
- Résultats **identiques** qu'avec la vue « en avant »



La preuve formelle de l'équivalence est disponible dans **LE Livre**

Contrôle TD(λ)

- Pour le problème du contrôle plutôt que de l'évaluation, on s'attache à Q plutôt qu'à V, de manière à pouvoir décider d'une action :

$$\begin{aligned} Q_{t+1}(s,a) &= Q_t(s,a) + \alpha\delta_t e_t(s,a) \\ \delta_t &= r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \\ e_t(s,a) &= \begin{cases} \gamma\lambda e_{t-1}(s) + 1 & \text{si } s = s_t \text{ et } a = a_t \\ \gamma\lambda e_{t-1}(s) & \text{sinon} \end{cases} \end{aligned}$$

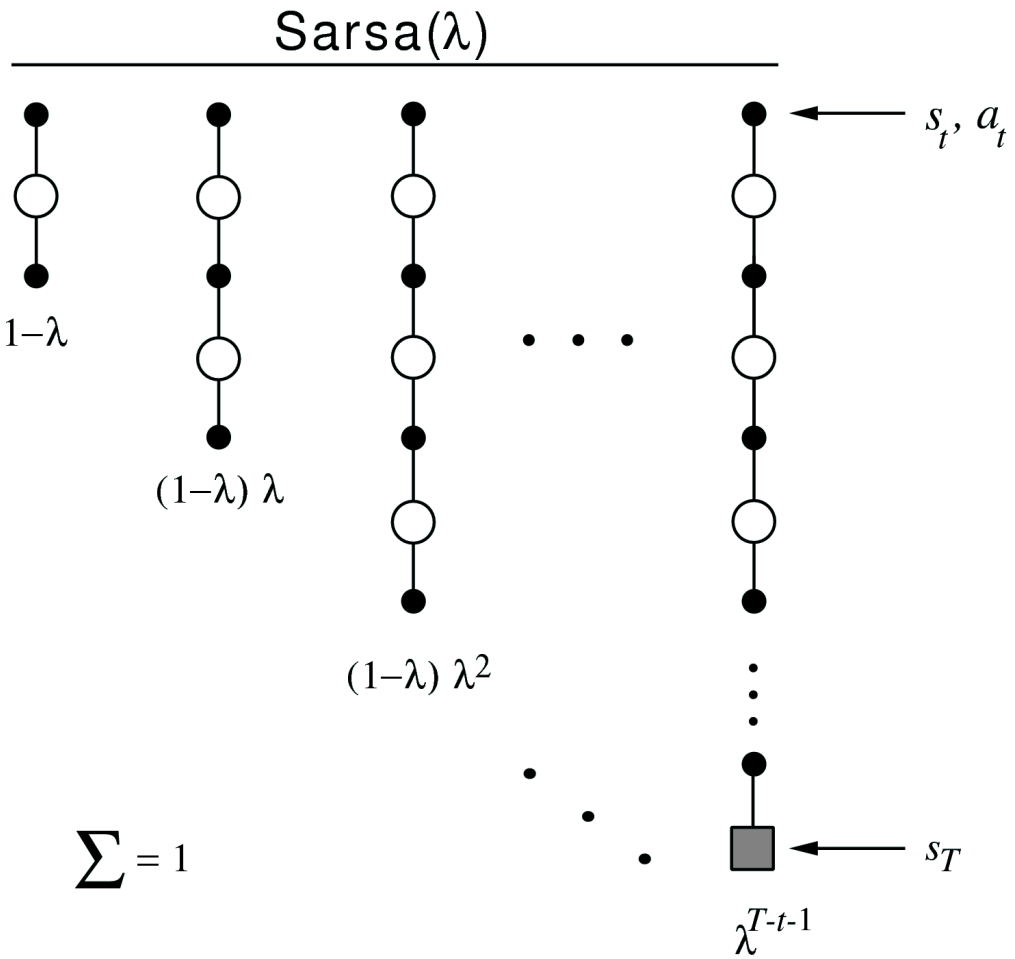
Algorithme

Sarsa(λ)

entrées : espaces $\langle \mathcal{S}, \mathcal{A} \rangle$
sorties : fonction valeur Q, actions émises

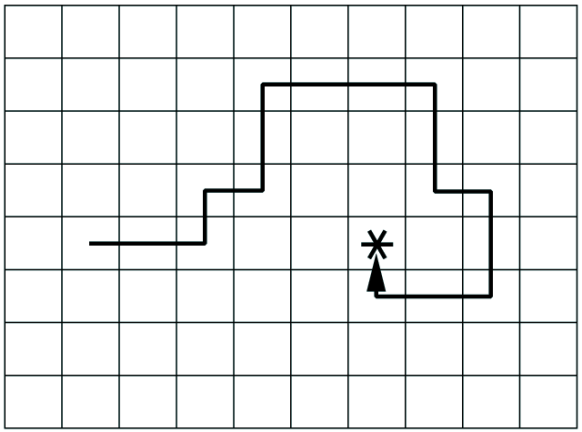
initialiser Q et e arbitrairement
boucle
$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, e(s,a) \leftarrow 0$
initialiser l'état initial s de l'épisode
$a \leftarrow \pi^{\epsilon}_{Q}(s)$ (ϵ-greedy)
répéter
émettre a ; observer r et s'
$a' \leftarrow \pi^{Q}_{\epsilon}(s')$ (ϵ-greedy)
$\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$
$e(s,a) \leftarrow e(s,a) + 1$
pour tout $(s,a) \in \mathcal{S} \times \mathcal{A}$ **faire**
$Q(s,a) \leftarrow Q(s,a) + \alpha \delta e(s,a)$
$e(s,a) \leftarrow \gamma \lambda e(s,a)$
fin pour
$s \leftarrow s'$; $a \leftarrow a'$
jusqu'à s terminal
fin boucle

Diagramme de rétropropagation de Sarsa(λ)

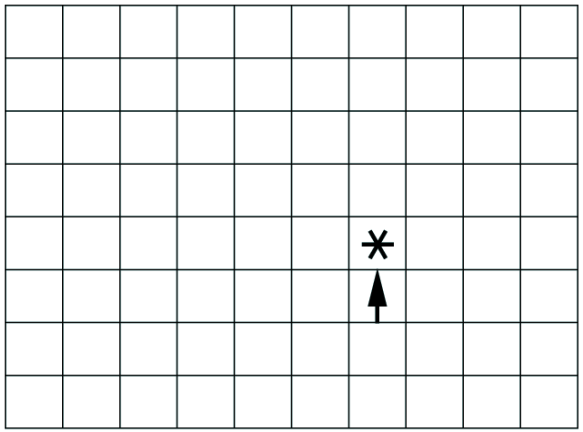


Exemple de traces Sarsa(λ)

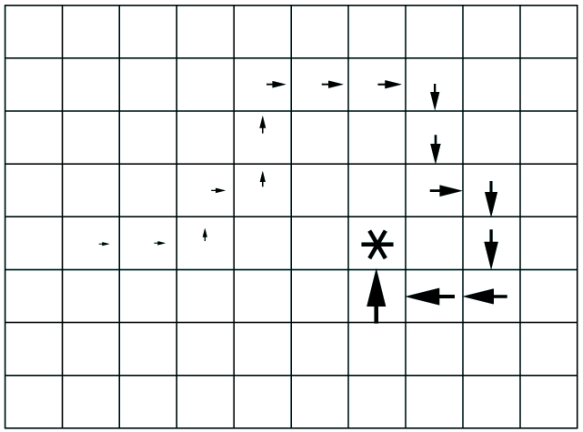
Path taken



Action values increased by one-step Sarsa



Action values increased by Sarsa(λ) with λ=0.9



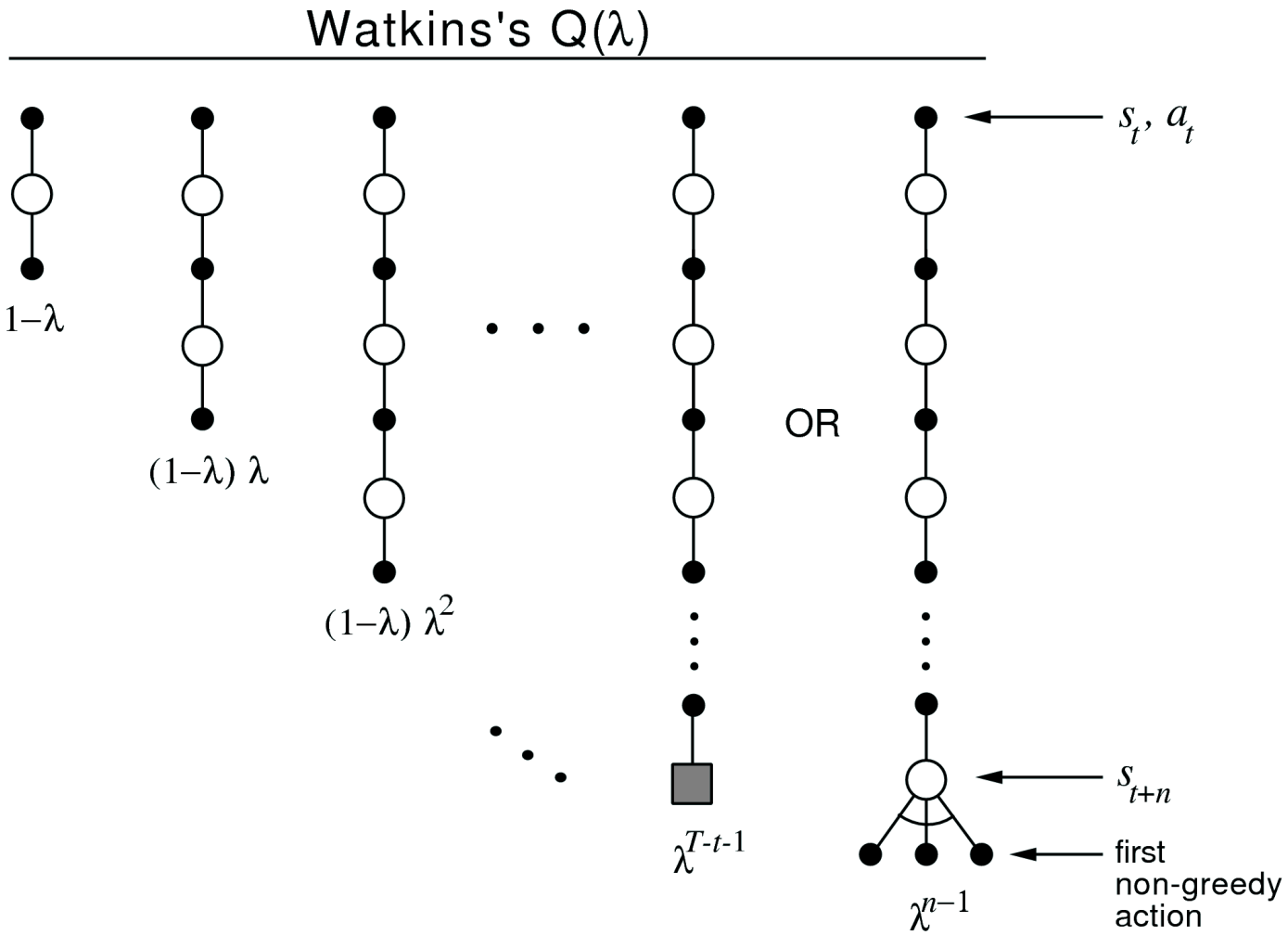
Apprentissage du contrôle TD(λ) hors ligne

- Premier algorithme : Watkins-Q(λ)
- **Problème** : Q-learning apprend une politique gloutonne alors que la politique utilisée ne l'est pas nécessairement
  - Des actions non gloutonnes sont parfois choisies
  - Quand une action exploratoire est choisie, les expériences qui la suivent ne sont plus liées à la politique apprise
  - Donc pour TD(λ), si a_{t+n} est la **première action exploratoire**, la **plus longue rétropropagation** concernera uniquement

$$r_{t+1} + \gamma r_{t+1} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n \max_{a \in \mathcal{A}(s)} Q_t(s_{t+n}, a)$$

- Ce problème limite considérablement la portée des traces d'éligibilité

Diagramme de rétropropagation de W-Q(λ)



Watkins-Q(λ)

- Les traces d'éligibilité sont semblables à celles de Sarsa(λ), mais elles sont réinitialisées dès qu'une action non exploratoire est choisie :

$$e_t = \mathcal{I}^s_{s_t}\mathcal{I}^a_{a_t} + \begin{cases} \gamma\lambda e_{t-1}(s,a) & \text{si } Q_{t-1}(s_t,a_t) = \max_{a\in\mathcal{A}(s)} Q_{t-1}(s_t,a) \\ 0 & \text{sinon} \end{cases}$$

où $\mathcal{I}^x_y$ est un facteur d'identité : $\mathcal{I}^x_y = \begin{cases} 1 & \text{si } x = y \\ 0 & \text{sinon} \end{cases}$

- Pour le reste ...

$$\begin{aligned} Q_{t+1}(s,a) &= Q_t(s,a) + \alpha\delta_t e_t(s,a) \\ \delta_t &= r_{t+1} + \gamma \max_{a'\in\mathcal{A}(s')} Q_t(s_{t+1},a') - Q_t(s_t,a_t) \end{aligned}$$

Algorithme

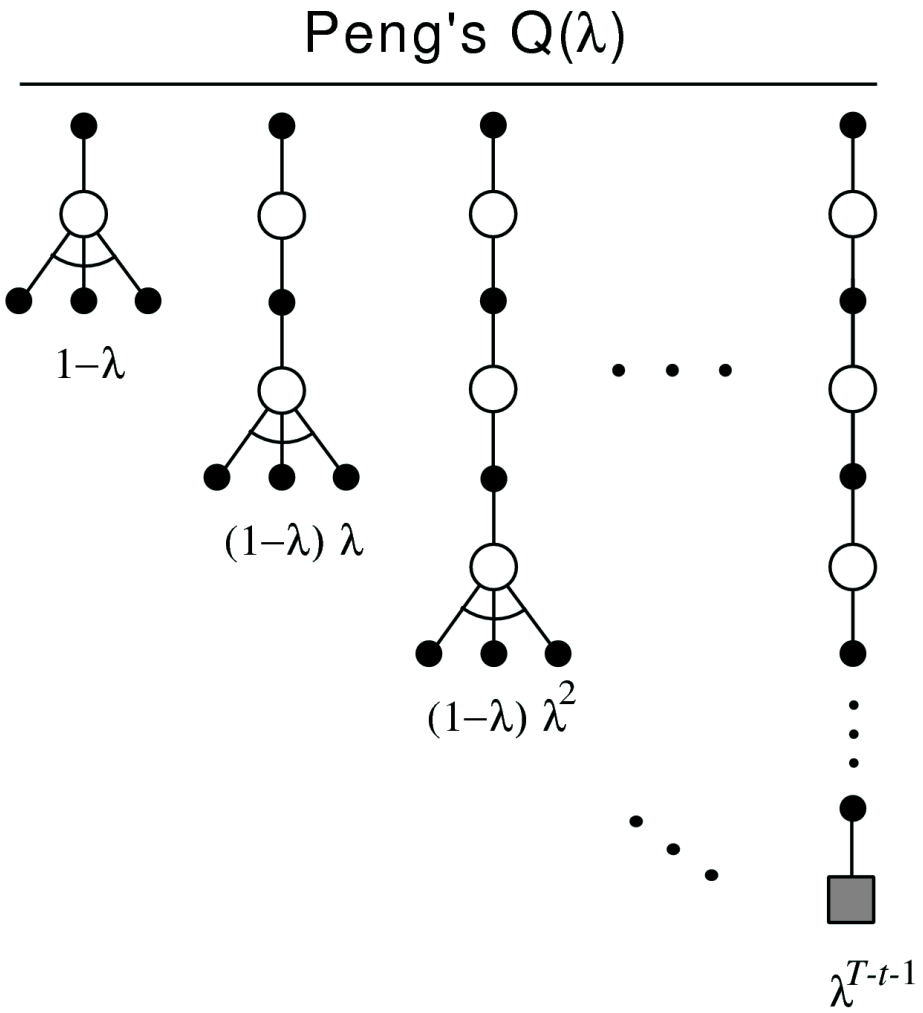
Watkins-Q(λ)

entrées : espaces $\langle \mathcal{S}, \mathcal{A} \rangle$
sorties : fonction valeur Q, actions émises

initialiser Q et e arbitrairement
boucle
$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, e(s,a) \leftarrow 0$
initialiser l'état initial s de l'épisode
$a' \leftarrow \pi_\epsilon^Q(s')$ (ϵ-greedy)
répéter
émettre a ; observer r et s'
$a' \leftarrow \pi_\epsilon^Q(s')$ (ϵ-greedy)
$a^\star = \arg\max_{b \in \mathcal{A}(s)} Q(s', b)$
$\delta \leftarrow r + \gamma Q(s', a^\star) - Q(s,a)$
$e(s,a) \leftarrow e(s,a) + 1$
pour tout $(s,a) \in \mathcal{S} \times \mathcal{A}$ **faire**
$Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
$e(s,a) \leftarrow \begin{cases} \gamma\lambda e(s,a) & \text{si } a' = a^\star \\ 0 & \text{sinon} \end{cases}$
fin pour
$s \leftarrow s'$; $a \leftarrow a'$
jusqu'à s terminal
fin boucle

Penq-Q(λ)

- Permet de remédier au problème de Watkins-Q(λ)



Traces d'éligibilité et méthodes Acteur/Critique

- **Rappel** : pour A/C(0), nous avions défini

$$p_{t+1}(s,a) = \begin{cases} p_t(s,a) + \alpha\delta_t & \text{si } a = a_t \text{ et } s = s_t \\ p_t(s,a) & \text{sinon} \end{cases}$$

- A/C(λ) généralise cette équation par :

$$p_{t+1}(s,a) = p_t(s,a) + \alpha\delta_t e_t(s,a)$$

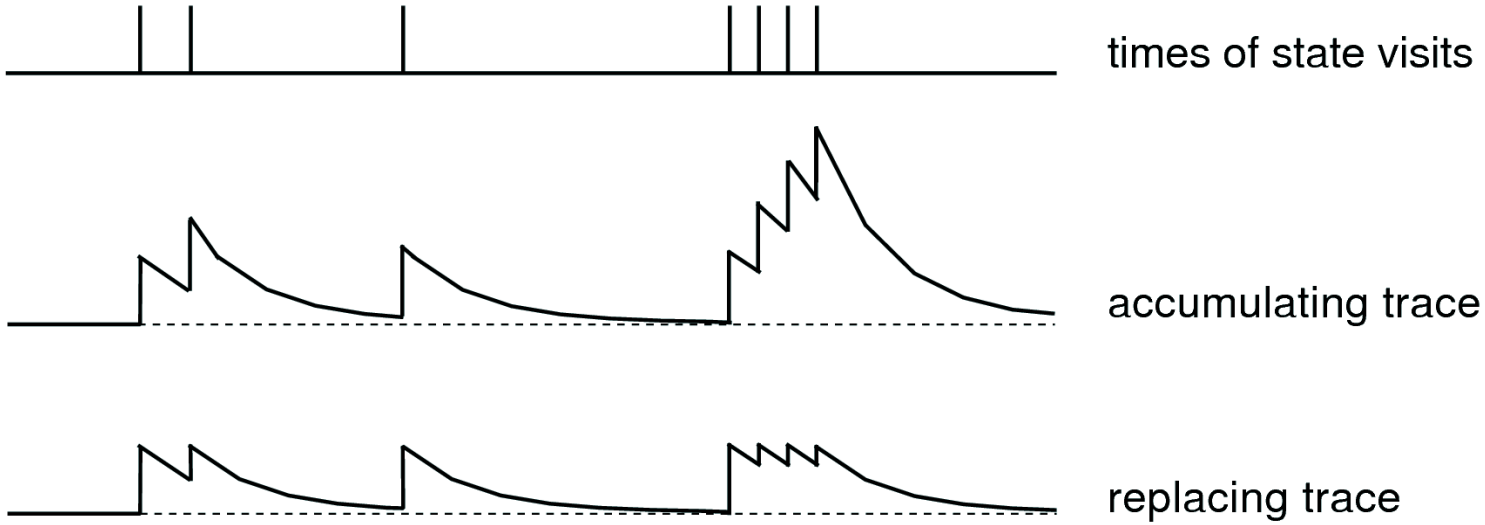
où $e(s,a)$ est une trace d'éligibilité similaire à Sarsa(λ)

- Dans le cas plus sophistiqué discuté pour A/C, nous aurons pour A/C(λ)

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 + \pi_s(s_t,a_t) & \text{si } a = a_t \text{ et } s = s_t \\ \gamma\lambda e_{t-1}(s,a) & \text{sinon} \end{cases}$$

Traces avec remplacement

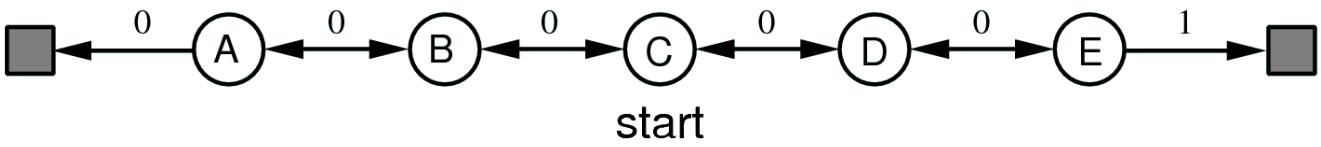
- Parfois, de meilleures performances peuvent être obtenues avec des **traces de remplacement**

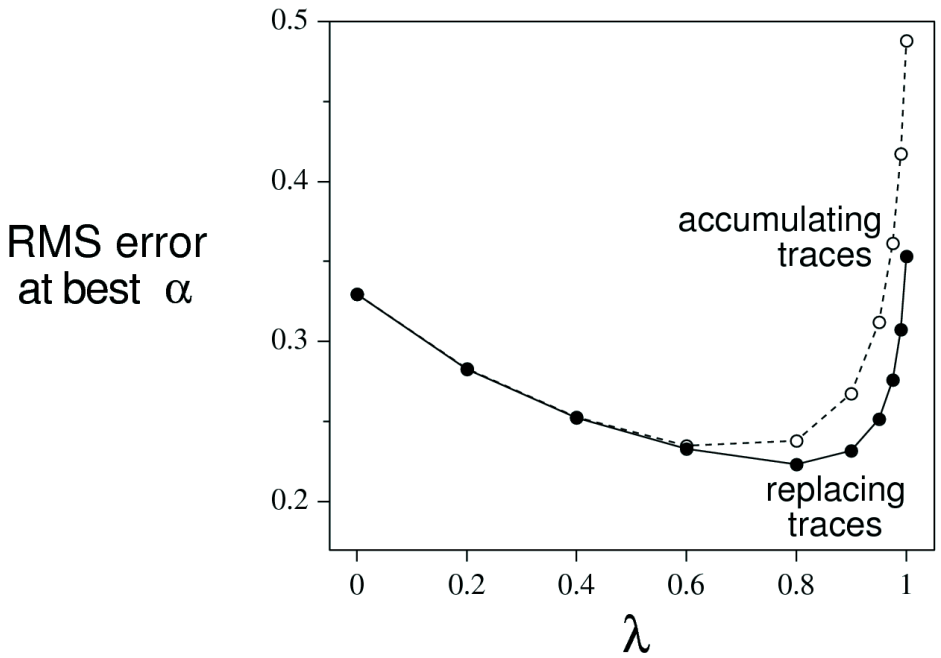


- On a alors, pour le problème d'**évaluation** :

$$e_t(s) = \begin{cases} 1 & \text{si } s = s_t \\ \gamma\lambda e_{t-1}(s) & \text{sinon} \end{cases}$$

Exemple





- **Marche aléatoire** avec 19 états au lieu de A, B, C, D et E
- Meilleure valeur de α sélectionnée

Contrôle avec les traces avec remplacement

$$e_t(s) = \begin{cases} 1 + \gamma\lambda e_{t-1}(s,a) & \text{si } s = s_t \text{ et } a = a_t \\ 0 & \text{si } s = s_t \text{ et } a \neq a_t \\ \gamma\lambda e_{t-1}(s,a) & \text{si } s = s_t \end{cases}$$

Lexique

- Trace d'éligibilité : **eligibility trace**
- Rétro-propagation : **backup**
- n-rétropropagation : n-**step backup**
- Retour-λ : λ-**return**
- Vue « en avant » ou théorique : **forward view**
- Vue « en arrière » ou algorithmique : **backward view**
- Trace avec remplacement : **replacing trace**

Plan

1 Problème d'AR
- Introduction
- Formalisation du problème

2 Solutions élémentaires
- Programmation Dynamique
- Monte Carlo
- Différence temporelle (TD)

3 Solutions unifiées
- Traces d'éligibilité
- Approximation de fonctions

4 Perspectives
- AR indirect
- POMDP
- Continuité du temps

Algorithmes tabulaires

- Les algorithmes décrits jusqu'ici sont des algorithmes dits **tabulaires**
  - Ils supposent des espaces d'états et d'actions **discrets** et **finis**
  - Ils stockent les valeurs ou les action-valeurs dans des **tables**

$Q(s,a)$	s_1	s_2	$\cdots$	s_n
a_1	0.1	0.3	$\cdots$	0.2
a_2	0.4	0.1	$\cdots$	0.4
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
a_m	0.3	0.2	$\cdots$	0.2

- Problèmes
  - **Explosion combinatoire** : quand la taille de $\mathcal{S}$ et $\mathcal{A}$ augmente, la taille des tables croît jusqu'à ce que des problèmes de performances se posent
  - **Espaces continus** : les états ne peuvent pas être représentés de manière discrète

Représentation paramétrée

- On ne peut pas toujours utiliser une **représentation tabulaire** de la fonction valeur
- Il faut alors se doter d'un moyen de **calculer** les valeurs en fonction des états
- On utilise une **fonction paramétrée** :
  - Des paramètres $\vec{\omega}$ déterminent la manière dont on calcule $V(s)$ en fonction de S
  - **Exemple :** les poids $\vec{\omega}$ d'une fonction linéaire $y = \vec{\omega} \cdot \vec{x}$ qui permettent de calculer y pour une infinité de valeurs $\vec{x}$
- L'apprentissage de la fonction valeur devient un problème d'**approximation de fonction**

Prédiction de valeur

- On se dote d'un **vecteur de paramètres** $\vec{\omega}_t$
  - V_t dépend complètement de $\vec{\omega}_t$
  - En modifiant $\vec{\omega}_t$ de pas de temps en pas de temps, on fait varier V_t
- Le vecteur $\vec{\omega}_t$ doit permettre l'**approximation d'une fonction** $s \mapsto v$
- Selon l'algorithme d'AR, la cible v de l'apprentissage par approximation sera différente
  - **DP** : $s \mapsto E_\pi \{r_{t+1} + \gamma V_t(s_{t+1}) \mid s_t = s\}$
  - **MC** : $s_t \mapsto R_t$
  - **TD** : $s_t \mapsto r_{t+1} + \gamma V_t(s_{t+1})$
  - **TD(λ)** : $s_t \mapsto R_t^\lambda$
- En identifiant clairement la cible, on se ramène à un problèmes d'**apprentissage supervisé** d'approximation de fonction

Minimisation de l'erreur

- En résolvant un problème d'approximation de fonction, on cherche à minimiser une **erreur** souvent **quadratique**

$$EQ(\vec{\omega}_t) = \sum_{s \in \mathcal{S}} P(s) \left[V^{\pi}(s) - V_t(s) \right]^2$$

où $P(s)$ est une distribution de probabilité pondérant l'erreur selon les états

- Chercher $V^{\star}$, c'est chercher $\vec{\omega}^{\star}$ tel que

$$EQ(\vec{\omega}^{\star}) \leq EQ(\vec{\omega})$$

pour tout $\vec{\omega}$ au voisinage de $\vec{\omega}^{\star}$

  - On cherche en fait des optimums locaux

Descente de gradient

- **Hypothèses**
  - $\vec{\omega}_t = (\vec{\omega}_t(1), \vec{\omega}_t(2), \cdots, \vec{\omega}_t(n))^T$
  - V_t est une fonction indéfiniment différentiable par $\vec{\omega}_t$ pour tous $s \in \mathcal{S}$
  - P est uniforme
  - A chaque pas de temps, on observe un exemple **donné** $s \mapsto V^\pi(s_t)$
- **Stratégie** : minimisation de l'erreur par ajustements successifs des paramètres

$$\begin{aligned} \vec{\omega}_{t+1} &= \vec{\omega}_t - \frac{1}{2}\alpha \nabla_{\vec{\omega}_t} \left[V^\pi(s_t) - V_t(s_t)\right]^2 \\ &= \vec{\omega}_t + \alpha \left[V^\pi(s_t) - V_t(s_t)\right] \nabla_{\vec{\omega}_t} V_t(s_t) \end{aligned}$$

- $\nabla_{\vec{\omega}_t} f(x) = \left(\frac{\partial f(x)}{\partial \vec{\omega}_t(1)}, \frac{\partial f(x)}{\partial \vec{\omega}_t(2)}, \cdots, \frac{\partial f(x)}{\partial \vec{\omega}_t(n)}\right)^T$ est un **gradient**, *càd* un vecteur de dérivées partielles

Desente de gradient généralisée

- On suppose maintenant que les exemples $s_t \mapsto v_t$ ne reflètent pas les vraies valeurs $V^\pi(s_t)$ mais seulement une **approximation** de $V^\pi(s_t)$
- On doit se contenter de v_t pour l'apprentissage de $\vec{\omega}_t$ et

$$\vec{\omega}_{t+1} = \vec{\omega}_t + \alpha \left[v_t - V_t(s_t) \right] \nabla_{\vec{\omega}_t} V_t(s_t)$$

- Si v_t est un **estimateur non biaisé** de $V^\pi(s_t)$, la convergence vers un optimum local est garantie à condition de faire décroître α
  - Non biaisé : $E\{v_t\} = V^\pi(s_t)$

Évaluation TD(λ)

- Application de la descente généralisée au retour TD(λ) : $v_t = R_t^\lambda$

Vue « en avant »

$$\vec{\omega}_{t+1} = \vec{\omega}_t + \alpha \left[R_t^\lambda - V_t(s_t) \right] \nabla_{\vec{\omega}_t} V_t(s_t)$$

- Malheureusement, pour $\lambda < 1$, R_t^λ n'est pas un estimâtes non biaisé
  - La convergence n'est pas formellement garantie
  - En pratique, c'est **efficace** et des garanties existent pour des cas particuliers
- **Vue « en arrière »**

$$\begin{aligned} \vec{\omega}_{t+1} &= \vec{\omega}_t + \alpha \delta_t \vec{e}_t \\ \delta_t &= r_{t+1} + \gamma V_t(s_{t+1}) - V(s_t) \\ \vec{e}_t &= \gamma \lambda \vec{e}_{t-1} + \nabla_{\vec{\omega}_t} V_t(s_t) \end{aligned}$$

avec $\vec{e}_0 = \vec{0}$

Algorithme

Évaluation TD(λ) avec descente de gradient

entrées : espaces $\langle \mathcal{S}, \mathcal{A} \rangle$, politique π
sorties : paramètres $\vec{\omega}$ définissant V associée à π

initialiser $\vec{\omega}$ arbitrairement
boucle
  $\vec{e} = \vec{0}$
  initialiser l'état initial s de l'épisode
  répéter
    $a \leftarrow \pi(s)$
    émettre a ; recevoir r et s'
    $\delta \leftarrow r + \gamma V(s') - V(s)$
    $\vec{e} \leftarrow \gamma\lambda\vec{e} + \nabla_{\vec{\omega}} V(s)$
    $\vec{\omega} \leftarrow \vec{\omega} + \alpha\delta\vec{e}$
    $s \leftarrow s'$
  jusqu'à s terminal
fin boucle

Méthodes linéaires

- V_t est une fonction linéaire de $\vec{\omega}_t$
- A chaque état s est associée une représentation en **attributs** numériques $\vec{\phi}_s = \left(\vec{\phi}_s(1), \vec{\phi}_s(2), \cdots, \vec{\phi}_s(n)\right)^T$
- L'approximation de la fonction valeur s'exprime comme

$$V_t(s) = \vec{\omega}_t^T \vec{\phi}_s = \sum_{i=1}^{n} \vec{\omega}_t(i)\vec{\phi}_s(i)$$

- Dans le cas linéaire, le gradient est simplement

$$\nabla_{\vec{\omega}} V(s) = \vec{\phi}_s$$

Algorithmes spécifiques

- **Convergence** pour tous les algorithmes utilisant une approximation linéaire
  - Dans le cas linéaire, la convergence est prouvée
  - Toutefois, l'algorithme ne converge pas toujours vers $\vec{\omega}^\star$, mais le point fixe $\vec{\omega}_\infty$ est borné par

$$EQ(\vec{\omega}_\infty) \leq \frac{1-\gamma\lambda}{1-\lambda} EQ(\vec{\omega}^\star)$$

  - Quand λ approche de 1, l'erreur approche de l'erreur minimale
- **Implantations**
  - Dans tous les cas, l'algorithme reste l'algorithme d'évaluation TD(λ)
  - Le gradient est simple à calculer
  - La différence entre les algorithmes tient à la manière de **définir les attributs**
    - **Il faut définir $\vec{\phi}$ astucieusement**

Fonctions de voisinage

- Pour définir $\vec{\phi}$, on peut utiliser des **fonctions de voisinage**
  - On définit n **régions** $\mathcal{R}_i$ qui forment un **ensemble de couverture** de l'espace d'états $\mathcal{S}$

$$\forall s \in \mathcal{S}, \vec{\phi}_s(i) = \begin{cases} 1 & \text{si } s \in \mathcal{R}_i \\ 0 & \text{sinon} \end{cases}$$

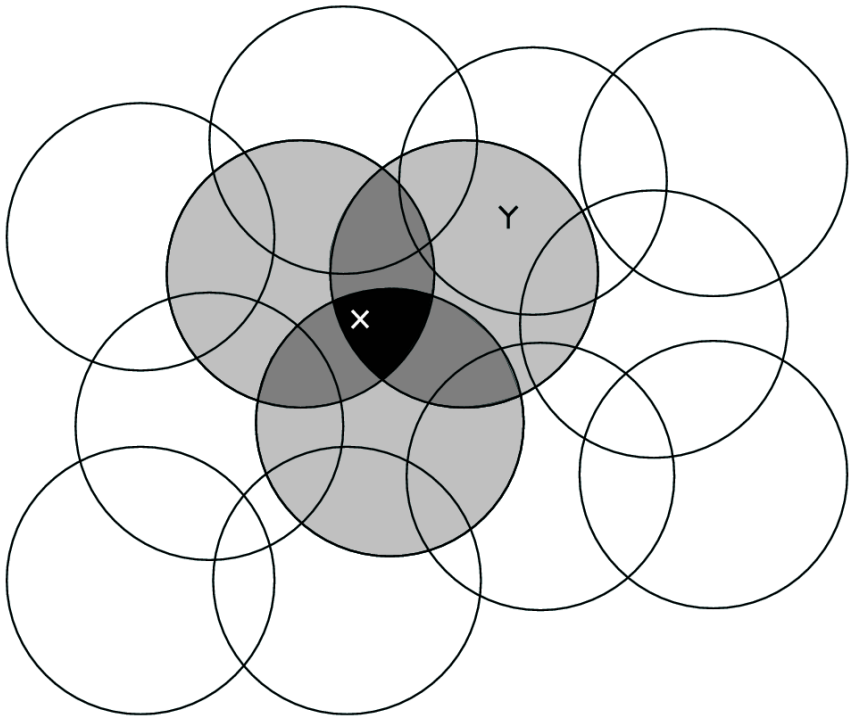
  - Les attributs d'un état représentent son appartenance à la région correspondante
  - Il y a autant d'attributs que de régions définies
- Le calcul de $V_t(s)$ se fait toujours de la même manière :

$$V_t(s) = \vec{\omega}_t^T \vec{\phi}_s = \sum_{i=1}^{n} \vec{\omega}_t(i) \vec{\phi}_s(i)$$

- Parmi les types de voisinages employés, citons
  - **Coarse coding**
  - **Tile coding**

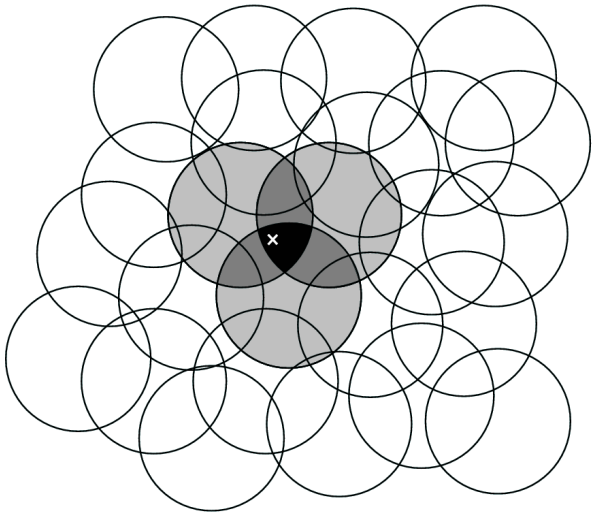
Coarse coding

- Les attributs sont définis par des **régions circulaires** (ou sphériques ou hyper-sphèriques) se recouvrant partiellement
- Une composante de $\vec{\omega}$ correspond à chaque région circulaire, et donc à chaque attribut
- La valeur d'un état est la somme des paramètres correspondant aux cercles dans lequel l'état est présent

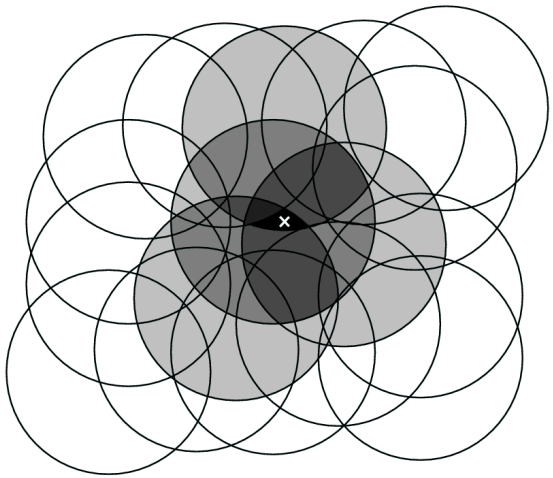


Coarse coding

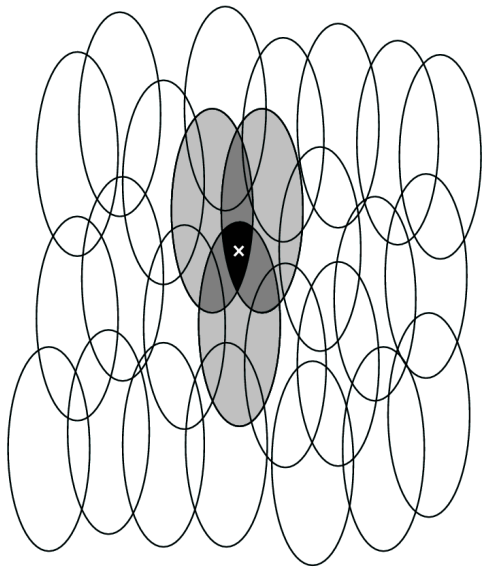
- La généralisation dépend de la taille, de la forme et de la disposition des cercles



a) Narrow generalization



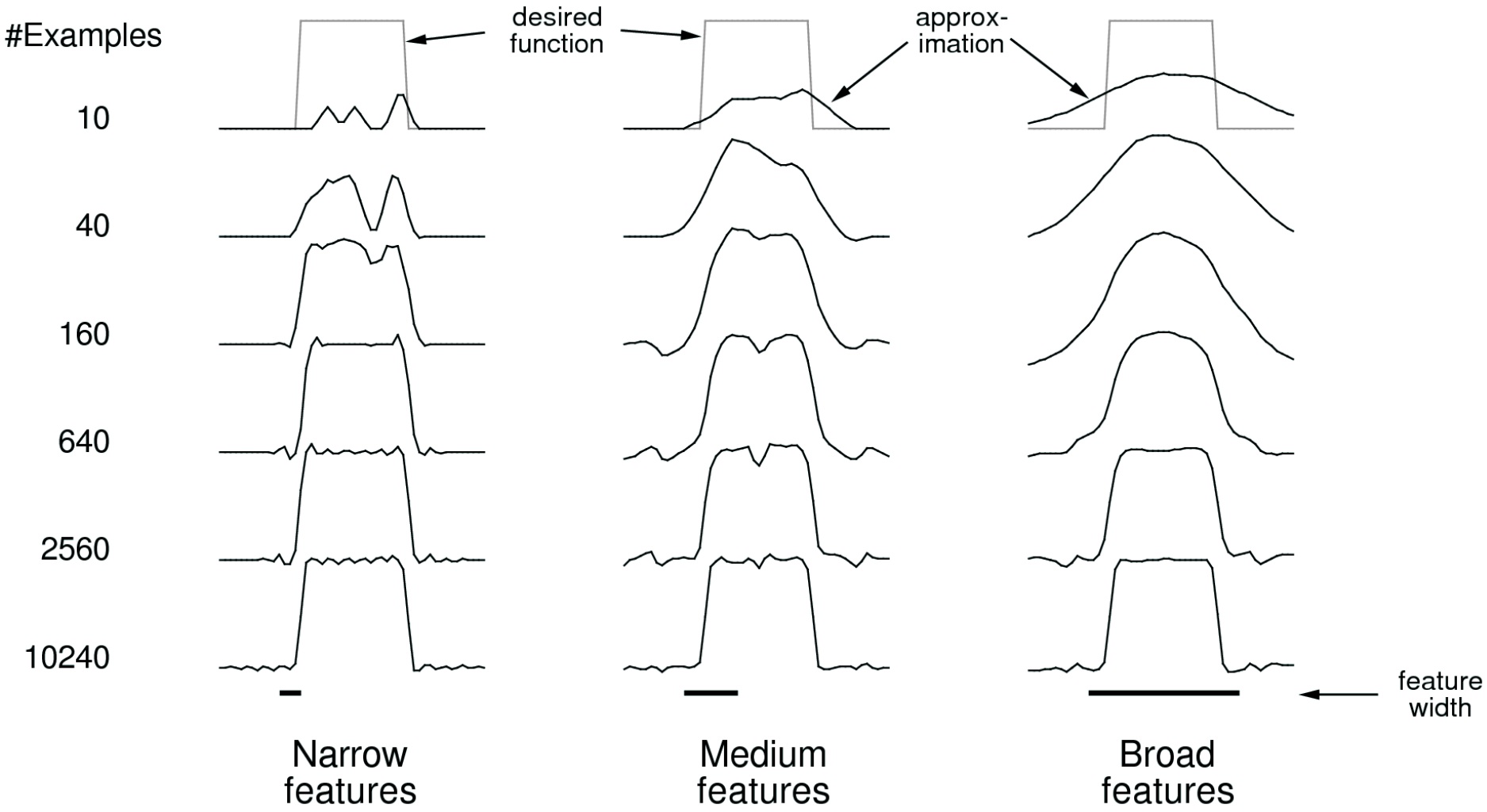
b) Broad generalization



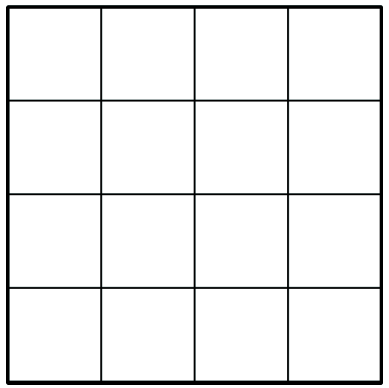
c) Asymmetric generalization

Taille des cercles et performances

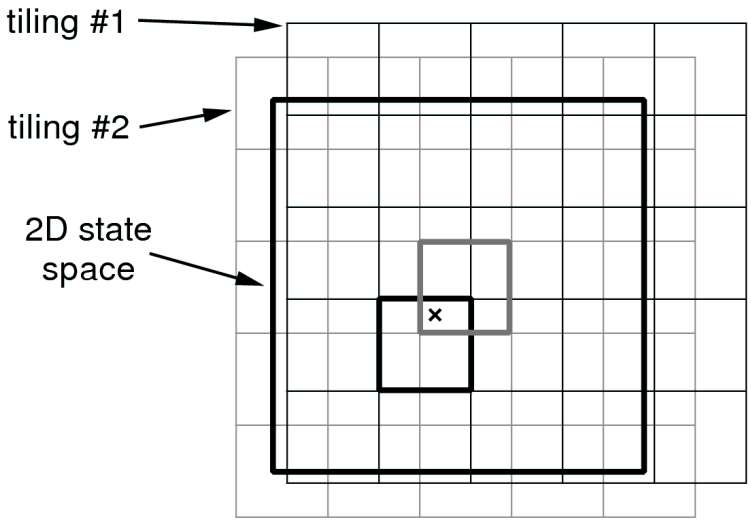
- La taille des cercles influe sur les performances et les résultats obtenus



Tile coding



- Plutôt que de définir des régions circulaires, on définit des **régions carrées** (ou cubiques ou hyper-cubiques)
- Les carrés sont organisés en grilles (**tiles**)
- Plusieurs grilles sont disposées de manière à former un entrelac

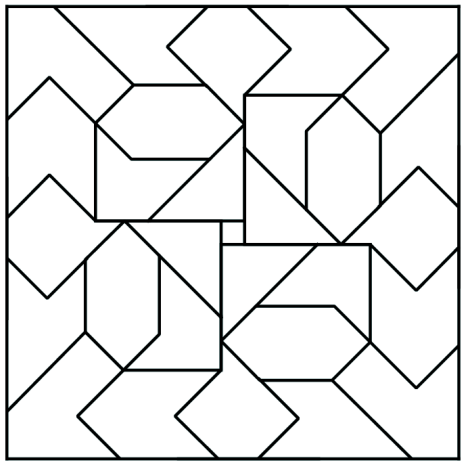


Shape of tiles ⇒ Generalization

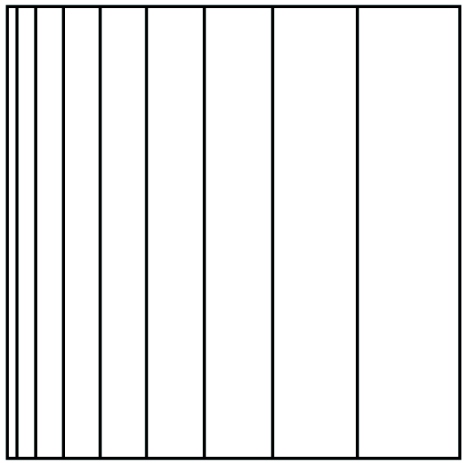
#Tilings ⇒ Resolution of final approximation

Tile coding

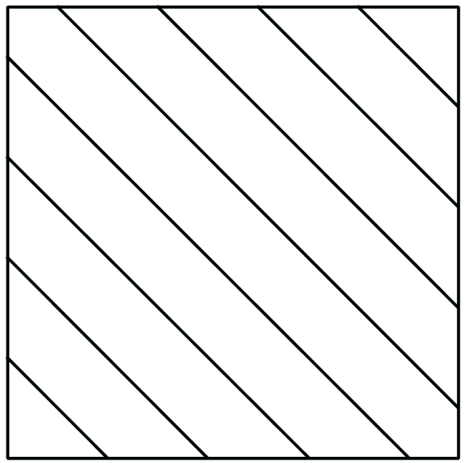
- Les régions ne sont pas obligatoirement carrées et organisées en grilles comme dans CMAC, elle peuvent prendre d'autres formes pourvu que tout l'espace d'états soit couvert



a) Irregular



b) Log stripes

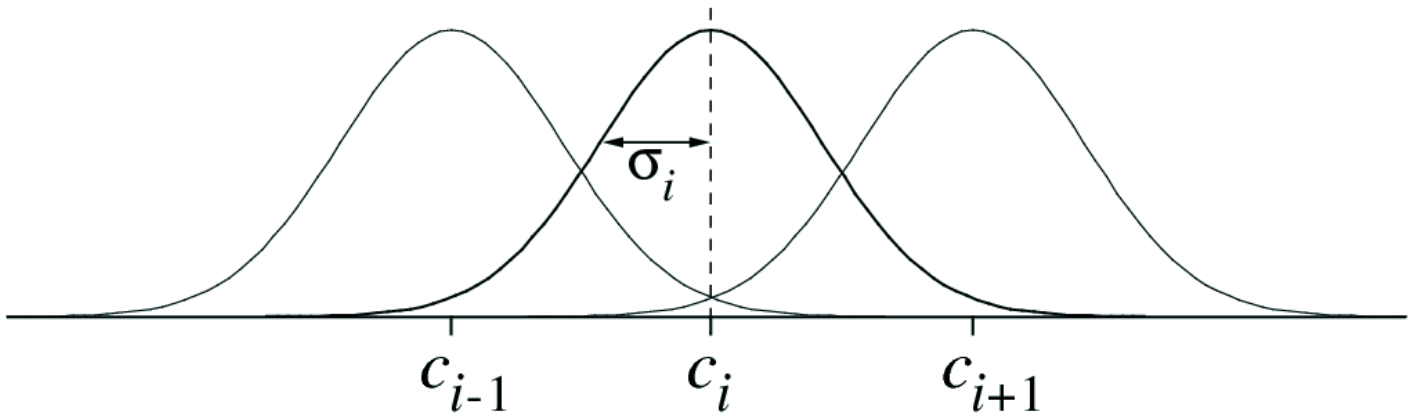


c) Diagonal stripes

Généralisation des fonctions de voisinage

- Plutôt que de valoir 0 ou 1, on veut que les $\vec{\phi}_s(i)$ prennent n'importe quelle valeur entre 0 et 1
- On définit des **fonctions à base radiale** par leur centre c_i et leur écart type σ_i

$$\vec{\phi}_s(i) = e^{-\frac{\|s-c_i\|^2}{2\sigma_i^2}}$$



Contrôle TD(λ) et approximation de fonction

- On étend le problème de l'évaluation à celui du contrôle en cherchant une approximation de $Q_t \approx Q^\pi$
  - Les exemples sont de la forme $s_t, a_t \mapsto v_t$ plutôt que $s_t \mapsto v_t$
- Avec v_t à définir, la forme générale de la **descente de gradient** pour le contrôle s'écrit

Vue « en avant »

$$\vec{\omega}_{t+1} = \vec{\omega}_t + \alpha \left[v_t - Q_t(s_t, a_t) \right] \nabla_{\vec{\omega}_t} Q_t(s_t, a_t)$$

- **Vue « en arrière »**

$$\begin{aligned} \vec{\omega}_{t+1} &= \vec{\omega}_t + \alpha \delta_t \vec{e}_t \\ \delta_t &= r_t + 1 + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \\ \vec{e}_t &= \gamma \lambda \vec{e}_{t-1} + \nabla_{\vec{\omega}_t} Q_t(s_t, a_t) \end{aligned}$$

avec $\vec{e}_0 = \vec{0}$

Algorithme d'apprentissage TD(λ) en ligne

Q(λ) avec descente de gradient linéaire, attributs booléens

entrées : espaces $\langle \mathcal{S}, \mathcal{A} \rangle$
sorties : fonction valeur Q, actions émises

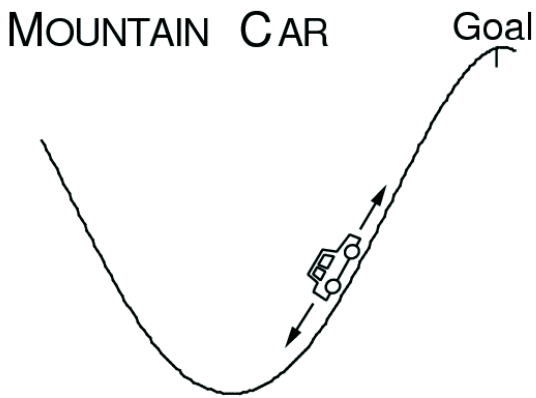
initialiser $\vec{\omega}$ arbitrairement
boucle
$\vec{e} = \vec{0}$
initialiser l'état initial s de l'épisode
$a \leftarrow \pi^{\epsilon}_{Q}(s)$ (ϵ-greedy)
$\mathcal{F}_a \leftarrow$ ensemble des attributs de (s, a)
répéter
exécuter « Épisode de Q(λ) avec descente de gradient linéaire »
jusqu'à s terminal
fin boucle

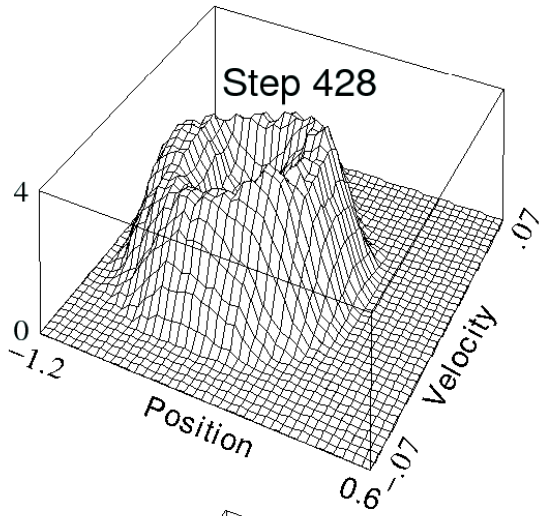
Algorithme d'apprentissage TD(λ) en ligne

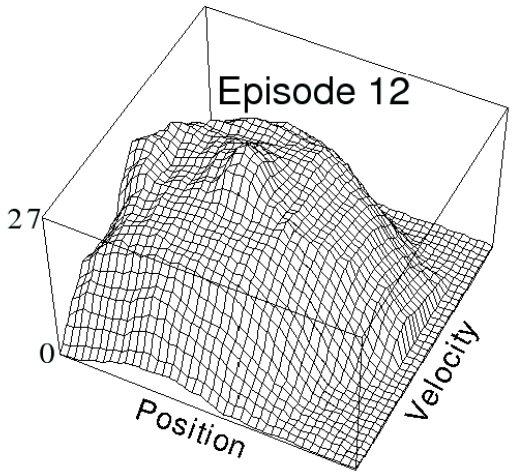
Épisode de Q(λ) avec descente de gradient linéaire, attributs booléens

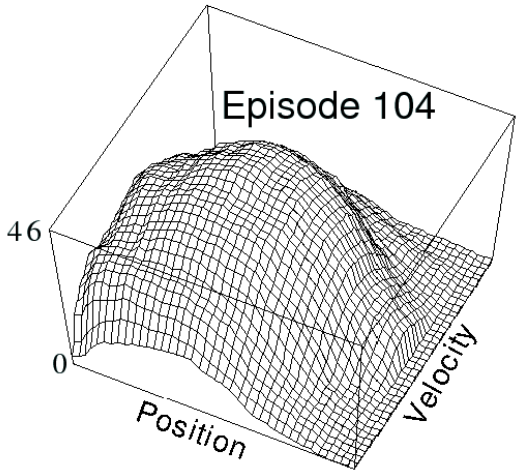
$\forall i \in \vec{\phi}_s(i), e(i) \leftarrow e(i) + 1$
émettre a ; observer r et s'
$\delta \leftarrow r - \sum_{i \in \mathcal{F}_a} \vec{\omega}(i)$
pour tout $a \in \mathcal{A}(s)$ **faire**
    $\mathcal{F}_a \leftarrow$ ensemble des attributs de (s, a)
    $Q_a \leftarrow \sum_{i \in \mathcal{F}_a} \vec{\omega}(i)$
fin pour
$\delta \leftarrow \delta + \gamma \max_{a \in \mathcal{A}(s)} Q_a$
$\vec{\omega} \leftarrow \vec{\omega} + \alpha \delta \vec{e}$
si random$(0, 1) < 1 - \epsilon$ **alors**
    $\forall a \in \mathcal{A}(s), Q_a \leftarrow \sum_{i \in \mathcal{F}_a} \vec{\omega}(i)$
    $a \leftarrow \arg\max_{a \in \mathcal{A}(s)} Q_a$
    $\vec{e} \leftarrow \gamma \lambda \vec{e}$
sinon
    $a \leftarrow$ au hasard dans$A(s)$
    $\vec{e} \leftarrow \vec{(0)}$
fin si

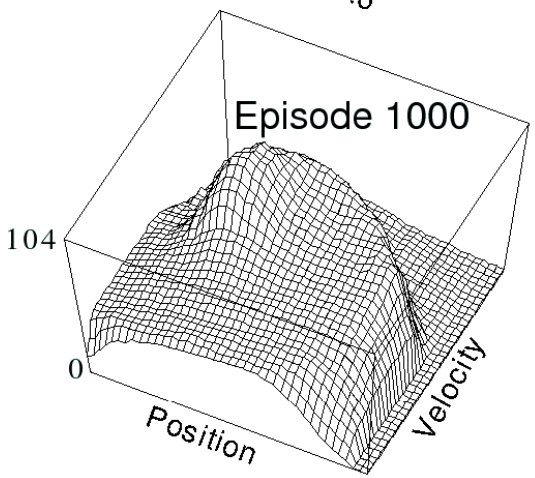
Exemple

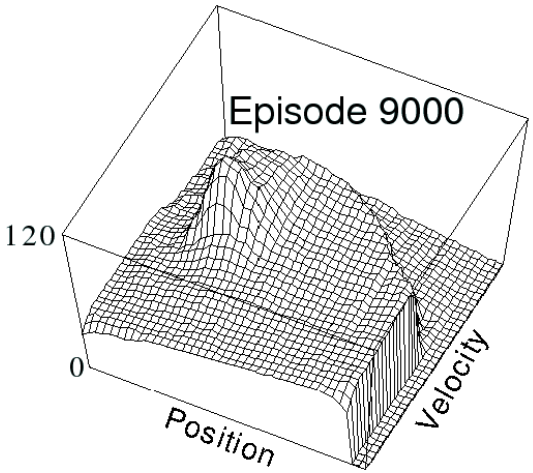

MOUNTAIN CAR
Goal


Step 428
4
0
−1.2
0.6
Position
−.07
.07
Velocity


Episode 12
27
0
Position
Velocity


Episode 104
46
0
Position
Velocity

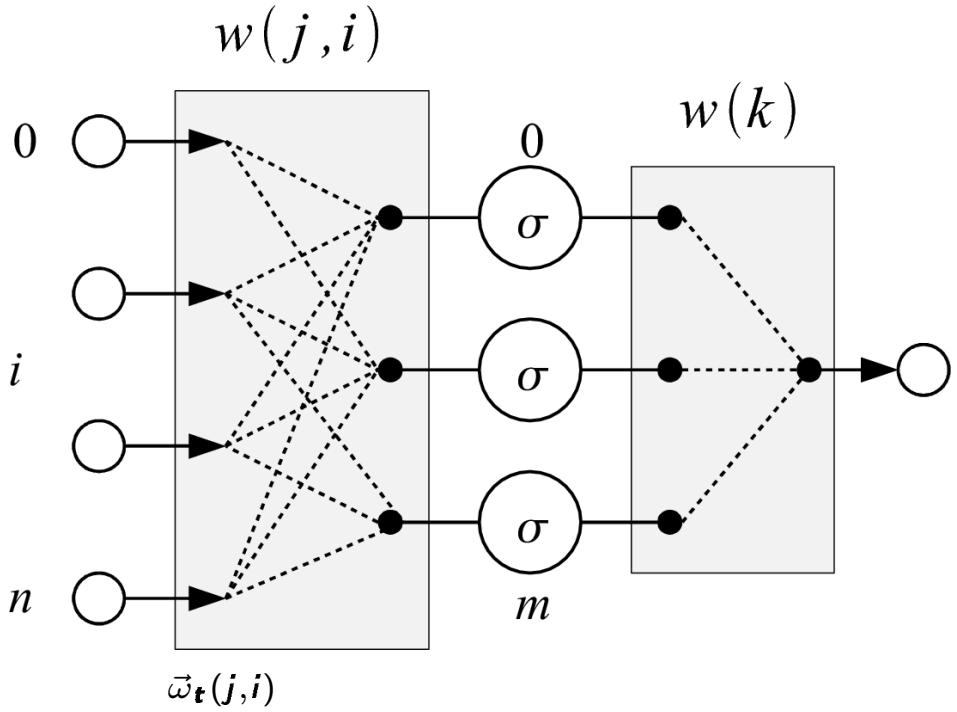

Episode 1000
104
0
Position
Velocity


Episode 9000
120
0
Position
Velocity

Approximation non linéaire

- Le gradient est plus facile à calculer dans le cas linéaire
- Néanmoins, on peut avoir besoin d'approximations non-linéaires

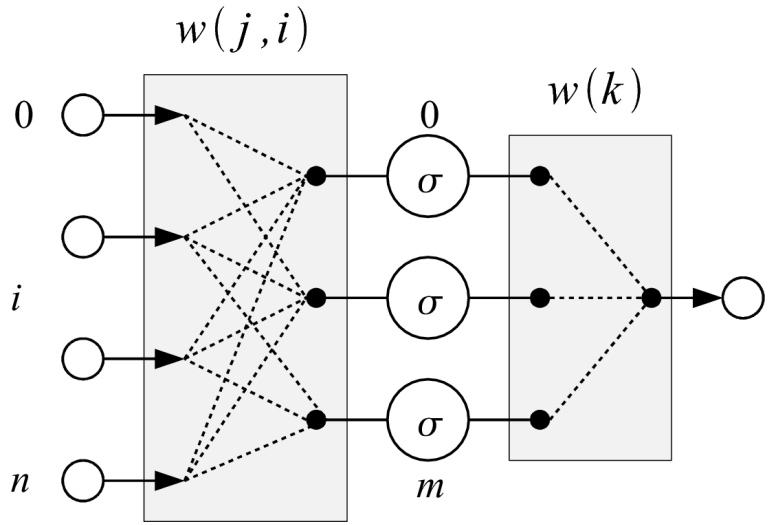
- On a toujours $\vec{\phi}_s = \left(\vec{\phi}_s(1), \vec{\phi}_s(2), \cdots, \vec{\phi}_s(n)\right)^T$ mais les paramètres $\vec{\omega}$ ne sont plus des poids dans une fonction linéaire
- **Exemple : perceptron à couche cachée** de m neurones



$$\vec{\omega}_t = (\overbrace{\vec{\omega}_t(1), \cdots, \vec{\omega}_t(n)}^{\vec{\omega}_t(k)}, \overbrace{\vec{\omega}_t(1,1), \cdots, \vec{\omega}_t(1,n)}^{\vec{\omega}_t(m,i)}, \cdots, \overbrace{\vec{\omega}_t(m,1), \cdots, \vec{\omega}_t(m,n)}^{\vec{\omega}_t(1,i)})^T$$

TD(λ) et perceptron multi-couche

- n attributs en entrée
- m neurones dans la couche cachée
- $\vec{\omega}_t$ composé de poids $\vec{\omega}_t(k)$ et $\vec{\omega}_t(j,i)$
  - $i \in [1,n]$, $j \in [1,m]$ et $k \in [1,m]$
- La sortie des neurones de la couche cachée est calculée à l'aide d'un fonction sigmoïde $\sigma(x) = \frac{1}{1+e^{-x}}$



$$V_t(s) = \sum_{k=1}^{m} \left[\vec{\omega}_t(k) \cdot \sigma \left(\sum_{i=1}^{n} \vec{\omega}_t(k,i) \vec{\phi}_s(i) \right) \right]$$

- Le calcul de $\nabla_{\vec{\omega}_t} V_t$ effectué par un algorithme de **rétropropagation du gradient**

Apprentissage de structure

- Les méthodes à base de fonctions de voisinage supposent que la fonction soit donnée avant l'apprentissage
- Le partitionnement de $\mathcal{S}$ en régions est élaboré par le concepteur avant l'apprentissage
  - Trouver le bon **partitionnement** est parfois difficile
- **Apprentissage de structure** : on laisse à un algorithme le soin de trouver un partitionnement adéquat
  - Méthodes à **base de règles** : les régions sont définies par des règles adaptatives
  - Méthodes fondées sur les **arbres de régression**

Lexique

- Algorithmes tabulaires : **tabular algorithms**
- Problème de passage à l'échelle : **scaling problem**
- Fonction paramétrée : **parametric function**
- Approximation de fonction : **function approximation**
- Apprentissage supervisé : **supervised learning**
- Erreur quadratique : **mean square error**
- Fonction indéfiniment différentiable : **smooth differentiable function**
- Estimateur non biaisé : **unbiased estimator**
- Etoit/large : **narrow/broad**
- Fonction à base radiale : **radial basis function**
- Perceptron à couche cachée : **hidden layer perceptron**

Plan

1. Problème d'AR
   - Introduction
   - Formalisation du problème
2. Solutions élémentaires
   - Programmation Dynamique
   - Monte Carlo
   - Différence temporelle (TD)
3. Solutions unifiées
   - Traces d'éligibilité
   - Approximation de fonctions
4. Perspectives
   - AR indirect
   - POMDP
   - Continuité du temps

AR direct et indirect

- Avec Sarsa, Q-learning, TD(λ)..., on apprend **directement** un modèle de Q (ou de V) à partir de l'expérience
- L'emploi de l'expérience est motivée par la méconnaissance *a priori* de l'environnement et du PDM $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ sous-jacent
  - Connaissant les fonctions $\mathcal{R}$ et $\mathcal{T}$, des méthodes de **décision** (programmation dynamique : DP) seraient employées plutôt que des méthodes d'apprentissage
- Lorsqu'on **apprend** par approximation des modèles de $\mathcal{R}$ ou de $\mathcal{T}$ pour plus facilement en déduire Q ou V, on procède à un apprentissage par renforcement **indirect**
- Les modèles appris permettent de produire des expériences simulées

$$\text{modèle} \xrightarrow{\text{simulation}} \text{expériences simulées} \xrightarrow{\text{éval}} \text{valeurs} \xrightarrow{\text{calcul}} \text{politique}$$

Planification par simulation d'action

Échantillonnage au hasard avec Q-learning

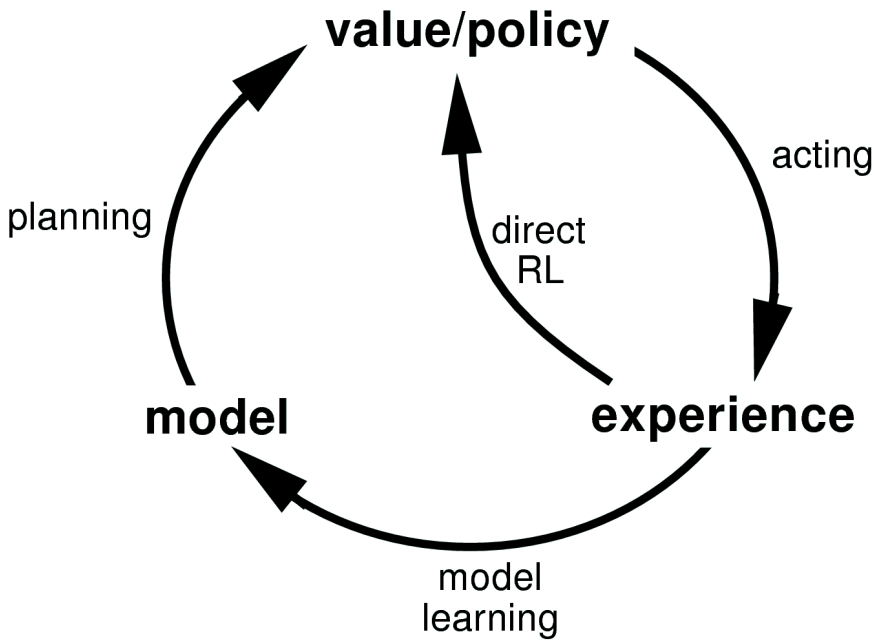
entrées : le modèle d'un environnement
sorties : une fonction valeur optimale Q

initialiser Q arbitrairement
boucle
choisir $s \in \mathcal{S}$ et $a \in \mathcal{A}(s)$ au hasard
émettre s vers le **modèle**
recevoir r **et** s' **du modèle**
$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s',a') - Q(s,a) \right]$
fin boucle

- Le modèle de l'environnement $\{\mathcal{S}, \mathcal{A}, T, R\}$ comporte
  - T un modèle de $\mathcal{T}$ obtenu avec un apprentissage par approximation
  - R un modèle de $\mathcal{R}$ obtenu par apprentissage

Intégration de la simulation et de l'expérience

- Dans la planification par simulation d'actions, on suppose qu'un apprentissage de T et/ou de R a été réalisé préalablement
- Dans les problèmes réels, on souhaite souvent intégrer les apprentissages de Q, de T et de R à l'expérience
  - **Problème :** avant que T et R soient corrects, la fonction Q déduite est inadéquate
  - **Intégration des actions simulées aux actions réelles**



Algorithme Dyna

Dyna-Q

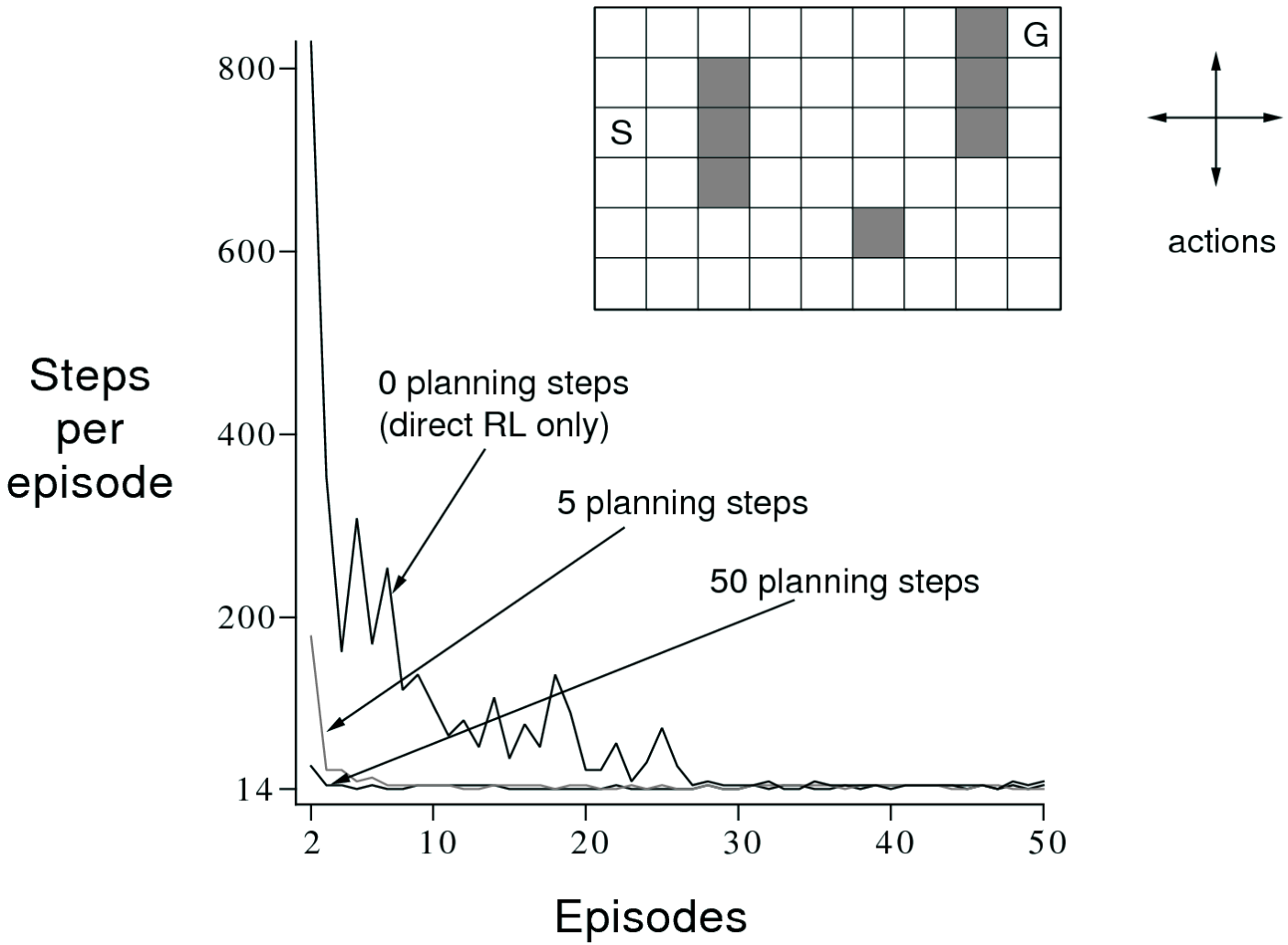
entrées : espaces $\langle \mathcal{S}, \mathcal{A} \rangle$
sorties : fonction valeur Q, modèles T et R, actions émises
initialiser Q, T et R arbitrairement
boucle
initialiser l'état initial s de l'épisode
répéter
$a \leftarrow \pi^{\epsilon}_{Q}(s)$ (ϵ-greedy)
émettre a ; observer r et s'
$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s',a') - Q(s,a) \right]$
$T(s,a) \leftarrow s'$
$R(s,a) \leftarrow r$
$s \leftarrow s'$
simulation d'actions
jusqu'à s terminal
fin boucle

Algorithme Dyna

Simulation d'actions pour Dyna-Q

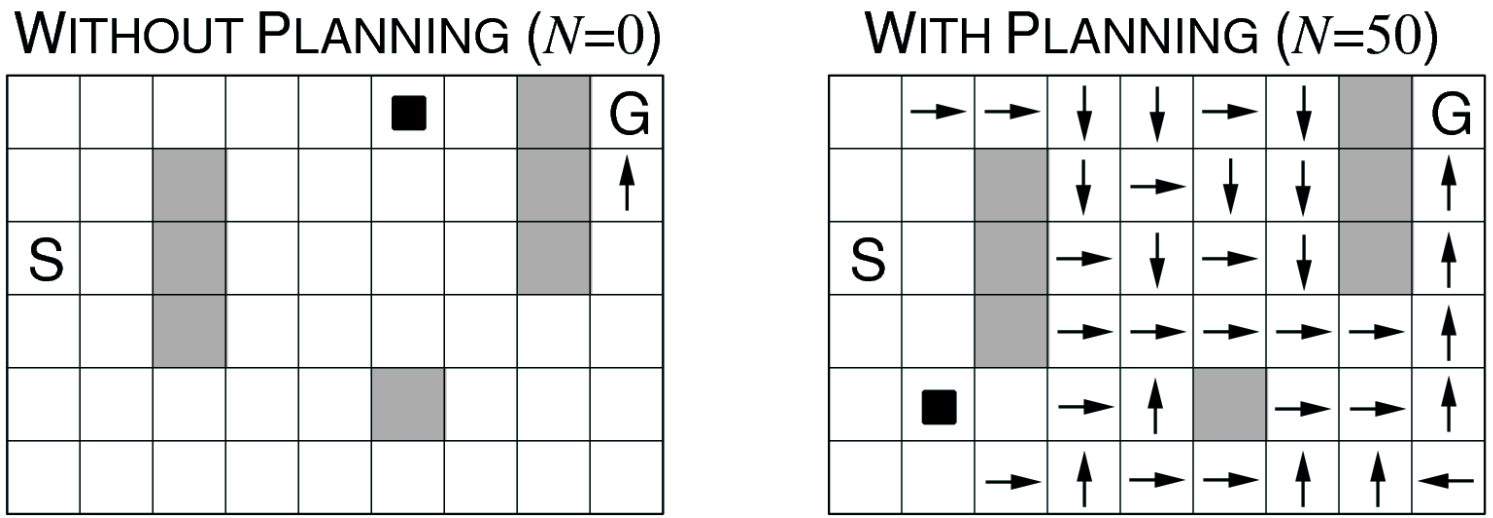
entrées : fonction valeur Q, espaces $\langle \mathcal{S}, \mathcal{A} \rangle$, fonction $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, fonction $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$
sorties : fonction valeur Q mise à jour
  pour N fois **faire**
    choisir $s_m \in \mathcal{S}$ et $a_m \in \mathcal{A}(s_m)$ au hasard
    $s'_m \leftarrow T(s_m, a_m)$
    $r_m \leftarrow R(s_m, a_m)$
    $Q(s_m, a_m) \leftarrow Q(s_m, a_m) + \alpha \left[r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s'_m, a'_m) - Q(s_m, a_m) \right]$
  fin pour

Résultats expérimentaux



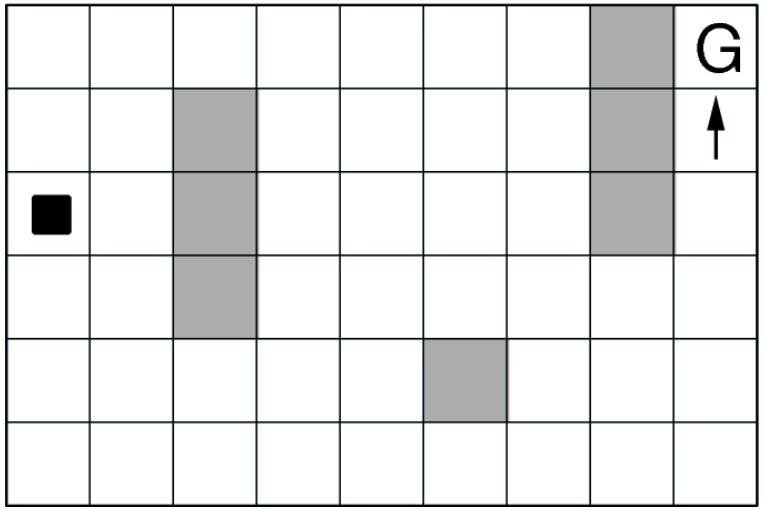
Politiques obtenues

- Politiques obtenues au milieu du deuxième épisode, en fonction du nombre N d'actions simulées



Ordonnancement des simulations d'actions

- Début du second épisode
  - Peu de paires (s, a) ont une valeur
  - Il est inutile de simuler une action qui mène dans un état avec une valeur non encore modifiée : on ne gagne aucune information
  - Il serait préférable d'**ordonner les simulations d'actions**



- Cet exemple suggère qu'il serait préférable de travailler « en arrière »
  - **On modifie d'abord la valeur d'un état menant à un état dont la valeur vient d'être modifiée**
  - L'algorithme **prioritized sweeping** exploite cette idée en utilisant des listes ordonnées pour les mises à jour
  - Les gains en performances atteignent souvent des facteurs 5 à 10

Algorithme

Prioritized sweeping

entrées : espaces $\langle \mathcal{S}, \mathcal{A} \rangle$
sorties : fonction valeur Q, modèles T et R, actions émises

initialiser Q, T et R arbitrairement
$L_p \leftarrow \emptyset$
boucle
  initialiser l'état initial s de l'épisode
  répéter
    $a \leftarrow \pi^{\epsilon}_{Q}(s)$
    émettre a ; observer r et s'
    $T(s,a) \leftarrow s'$; $R(s,a) \leftarrow r$
    $p \leftarrow \left| r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s',a') - Q(s,a) \right|$
    si $p > \theta$ **alors**
      insérer (s,a) dans L_p avec une priorité de p
    fin si
    simulation d'actions par priorité
  jusqu'à s terminal
fin boucle

Algorithme

Simulation d'actions par priorité

entrées : fonction valeur Q, espaces $\langle \mathcal{S}, \mathcal{A} \rangle$, fonction $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, fonction $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$
sorties : fonction valeur Q et L_p mises à jour

pour N fois et tant que L_p n'est pas vide **faire**
  $(s'_m, a'_m) \leftarrow$ premier élément de L_p
  $s''_m \leftarrow T(s'_m, a'_m)$
  $r'_m \leftarrow R(s'_m, a'_m)$
  $Q(s'_m, a'_m) \leftarrow Q(s'_m, a'_m) + \alpha \left[r' + \gamma \max_{a'' \in \mathcal{A}(s'')} Q(s''_m, a''_m) - Q(s'_m, a'_m) \right]$
  pour tout $(s_m, a_m) \in T^{-1}(s'_m)$ **faire**
    $r_m \leftarrow R(s_m, a_m)$
    $p_m \leftarrow \left| r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s'_m, a'_m) - Q(s_m, a_m) \right|$
    si $p_m > \theta$ **alors**
      insérer (s_m, a_m) dans L_p avec une priorité de p_m
    fin si
    enlever (s'_m, a'_m) de L_p
  fin pour
fin pour

Lexique

- Apprentissage par renforcement indirect : **indirect reinforcement learning / model based learning**

Plan

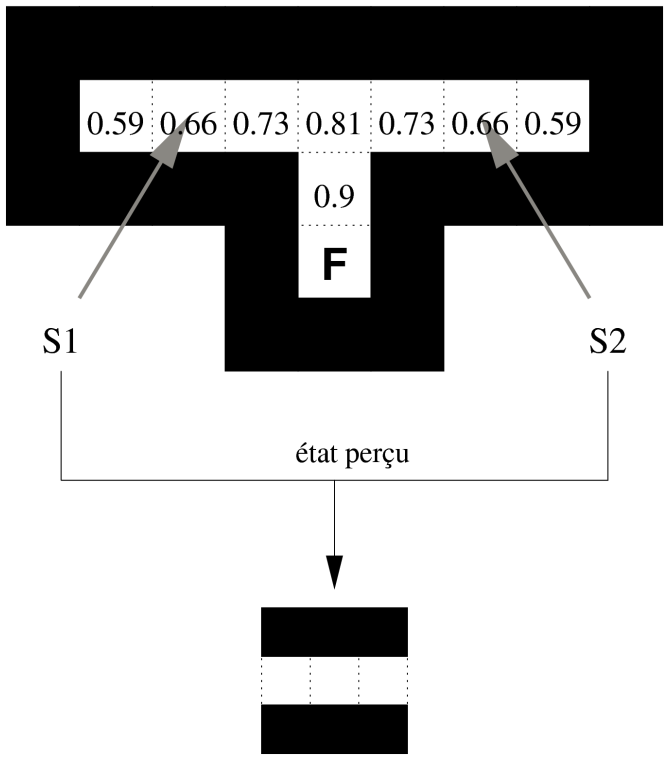
1. Problème d'AR
   - Introduction
   - Formalisation du problème
2. Solutions élémentaires
   - Programmation Dynamique
   - Monte Carlo
   - Différence temporelle (TD)
3. Solutions unifiées
   - Traces d'éligibilité
   - Approximation de fonctions
4. Perspectives
   - AR indirect
   - POMDP
   - Continuité du temps

Problèmes non markoviens

- Dans DP, TD(λ), Q-learning, Sarsa, Dyna-Q..., les conséquences d'une action ne dépendent que de l'état courant et la dynamique de l'environnement peut être décrite par la distribution de probabilité

$$Pr\left\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\right\}$$

  - La **propriété de Markov** est vérifiée

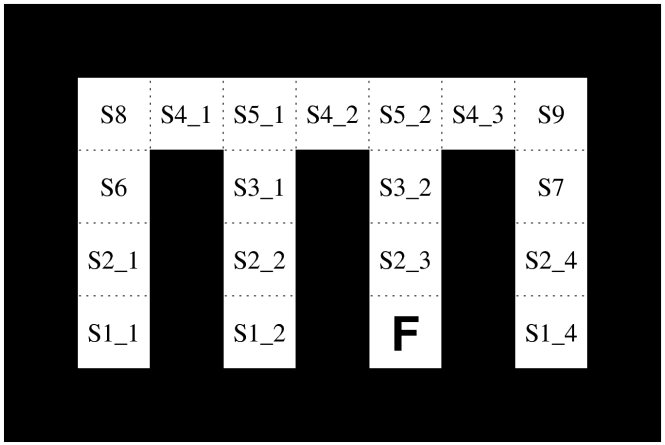


- Or, dans un grand nombre de problèmes, cette propriété n'est pas vérifiée et

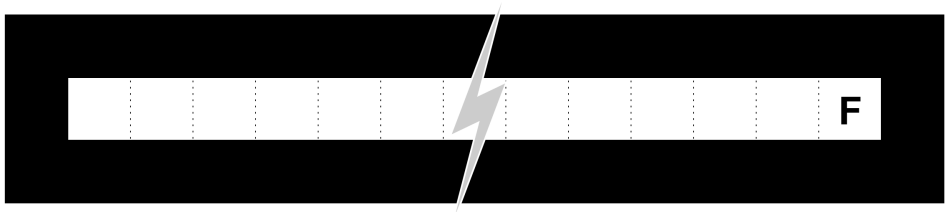
$$Pr\left\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, \cdots, r_1, s_0, a_0\right\}$$

Ambiguïtés sur les états et les récompenses

- Deux facteurs d'ambiguïté
  1. Vis à vis des états et de $\mathcal{T}$ ($S4_1$, $S4_2$ et $S4_3$)



  2. Vis à vis des récompenses et de $\mathcal{R}$ ou V



- On peut avoir ❶ sans ❷ et réciproquement

Problème partiellement observable

- **En programmation dynamique (DP)**, on substitue les problèmes markoviens partiellement observables **(POMDP)** aux problèmes de décision markoviens **(PDM)**
- Un POMDP se définit par $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O} \rangle$, où $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ est un PDM ordinaire
  - Ω est l'ensemble des observations
  - $\mathcal{O} : \mathcal{A} \times \mathcal{S} \times \Omega \rightarrow [0, 1]$ est une fonction d'observation

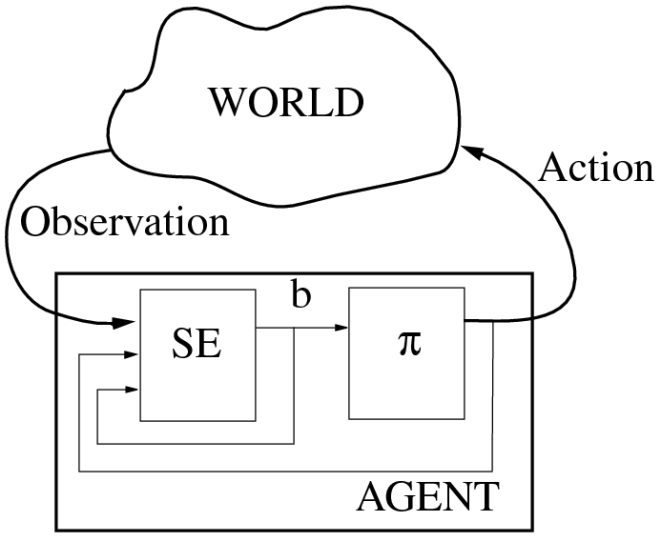
$$\mathcal{O}^{o'}_{as'} = \mathcal{O}(a, s', o') = Pr\{o_{t+1} = o' \mid a_t = a, s_{t+1} = s'\}$$

- On distingue ici ce qui est perçu par l'agent (les **observations**) de son **état réel** qui lui est inaccessible directement
- La fonction d'observation définit ce que l'agent perçoit en fonction de sa dernière action et de son état courant

Contrôle dans un POMDP

On connaît tous les éléments de $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O} \rangle$, le problème étant que l'on a seulement accès aux observation et pas aux états

- Le problème général de la **décision** est décomposé en deux modules
  - **SE :** (state estimator) pour l'estimation de l'**état** courant
  - π : pour le problème de décision proprement, et du choix d'une action en fonction de l'état estimé



- Un **état estimé** b est une distribution de probabilité sur $\mathcal{S}$
  - $b : \mathcal{S} \rightarrow [0, 1]$
  - $b(s)$ est la probabilité assignée à s
  - $\sum_{s \in \mathcal{S}} b(s) = 1$

Mise à jour de l'estimation de l'état

- SE doit permettre de donner un nouvel état estimé b' en fonction de l'ancien b, de sa dernière action a et de la nouvelle observation o'
- Pour tout $s' \in \mathcal{S}$

$$
\begin{aligned}
b'(s') &= Pr\{s' \mid o', a, b\} \\
&= \frac{Pr\{o' \mid s', a, b\}\, Pr\{s' \mid a, b\}}{Pr\{o' \mid a, b\}} \\
&= \frac{Pr\{o' \mid s', a\} \sum_{s \in \mathcal{S}} [Pr\{s' \mid a, b, s\}\, Pr\{s \mid a, b\}]}{Pr\{o' \mid a, b\}} \\
&= \frac{\mathcal{O}^{o'}_{as'} \sum_{s \in \mathcal{S}} \mathcal{T}^{s'}_{sa}\, b(s)}{Pr\{o' \mid a, b\}}
\end{aligned}
$$

- $Pr\{o' \mid a, b\}$ est un facteur de normalisation indépendant de s'
- SE estime toutes les nouvelles probabilités $b'(\cdot)$ en fonction des $b(\cdot)$

PDM estimé

- Pour calculer une politique, on définit le PDM $\langle \mathcal{B}, \mathcal{A}, \overset{\mathcal{B}}{\mathcal{T}}, \overset{\mathcal{B}}{\mathcal{R}} \rangle$
  - $\mathcal{B}$ est l'espace d'états estimés et $\mathcal{A}$ est l'espace d'actions original
  - $\overset{\mathcal{B}}{\mathcal{T}}: \mathcal{B} \times \mathcal{A} \times \mathcal{B} \rightarrow [0,1]$ est la fonction de transition entre états estimés

$$\begin{aligned} \overset{\mathcal{B}}{\mathcal{T}}{}^{b'}_{ba} = \overset{\mathcal{B}}{\mathcal{T}}(b,a,b') &= Pr\{b_{t+1} = b' \mid b_t = b, a_t = a\} \\ &= \sum_{o \in \Omega} Pr\{b' \mid b, a, o'\} Pr\{o' \mid b, a\} \end{aligned}$$

avec $Pr\{b' \mid b, a, o'\} = \begin{cases} 1 & \text{si } SE(b,a,o') = b' \\ 0 & \text{sinon} \end{cases}$

- $\overset{\mathcal{B}}{\mathcal{R}}: \mathcal{B} \times \mathcal{A} \rightarrow \mathbb{R}$ est la fonction de récompense estimée

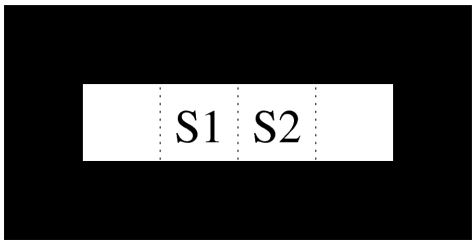
$$\overset{\mathcal{B}}{\mathcal{R}}_{ba} = \overset{\mathcal{B}}{\mathcal{R}}(b,a) = \sum_{s \in \mathcal{S}} b(s) \mathcal{R}_{sa}$$

AR dans un POMDP

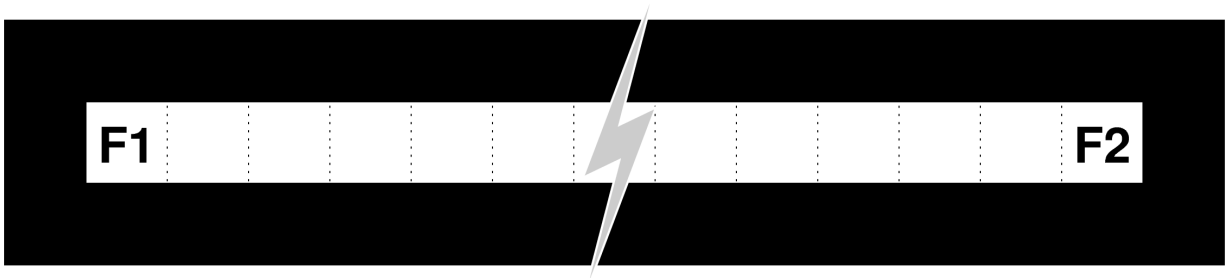
- Divers algorithmes de DP ont été développés pour résoudre le problème du **calcul d'une politique optimale**
  - Dans ce cas, le POMDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O} \rangle$ doit être connu
- Contrairement au cas markovien, aucun algorithme d'**apprentissage** n'est issu directement de ces travaux « **théoriques** » : on ne dispose pas d'approche
  - Il est difficile de construire $\mathcal{S}$ quand on n'a accès qu'aux observations
  - ...
- Les approches usuelles sont plus « **algorithmiques** »
  - Conserver une **mémoire explicite du passé**
  - Création de **séquences d'actions** (ou de chaînes d'actions)
  - **Découpage d'un POMDP** en plusieurs PDM ou augmentation des perceptions par des **états internes**

Mémoire explicite du passé

- L'état n'est plus s_t mais la **concaténation de** $s_t, s_{t-1}, \cdots, s_{t-N}$
- Les algorithmes sont les mêmes que dans le cas normal
- Permet de résoudre des problèmes dit **Markov-N**
  - Problème Markov-1



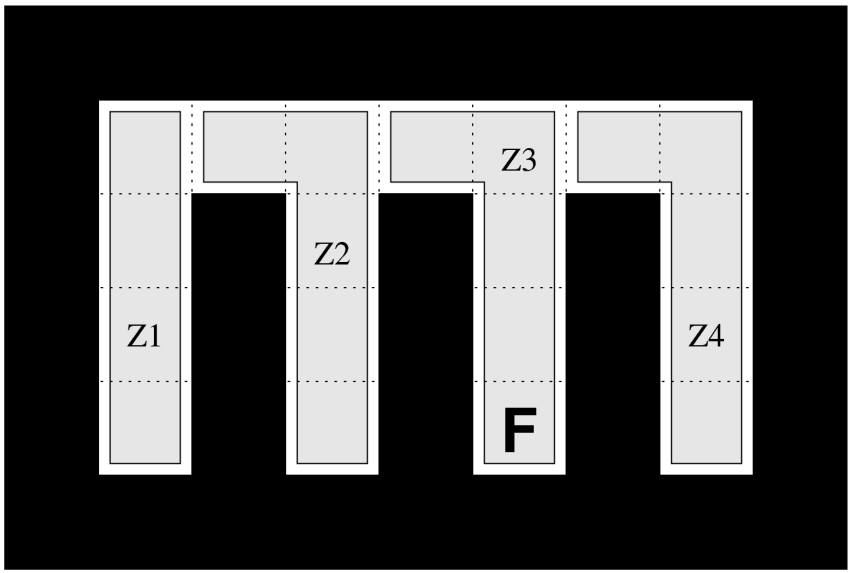
  - Problème Markov-N, N arbitraire



- **Problème :** Choix de N

Découpage d'un POMDP

- Plusieurs **modules** prennent chacun en charge un sous ensemble de $\mathcal{S}$
- Des mécanismes distribués ou centralisés permettent de passer d'un module de contrôle à un autre
- L'espace d'états est découpé en sous-espaces dans lesquels le problème est markovien



- **Problème :** Il faut définir à l'avance le nombre de modules, qui reste en outre toujours assez faible

Lexique

- Problème markovien partiellement observable : **partially observable markov problem**
- État estimé : **belief state**
- PDM estimé : **belief MDP**

Plan

1. Problème d'AR
   - Introduction
   - Formalisation du problème
2. Solutions élémentaires
   - Programmation Dynamique
   - Monte Carlo
   - Différence temporelle (TD)
3. Solutions unifiées
   - Traces d'éligibilité
   - Approximation de fonctions
4. Perspectives
   - AR indirect
   - POMDP
   - Continuité du temps

PDM continu

- Si l'espace d'états $\mathcal{S}$ ou $\mathcal{A}$ est continu, on utilise des algorithmes de descente du gradient pour l'apprentissage par approximation
- Si $\mathcal{S}$ ou $\mathcal{A}$ peuvent être continus, le temps peut l'être :

On n'a plus $t \in \mathbb{N}^+$ mais $t \in \mathbb{R}^+$

  - $t+1$ n'est plus l'état suivant immédiatement t
- Dans le cas à **temps continu**, on définit le PDM $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ avec

$$\mathcal{T}_a^s = \frac{\partial s}{\partial a}$$

pour représenter la dynamique de l'environnement en temps continu

  - $\mathcal{T}_a^s$ donne une dérivée et pas un nouvel état comme dans le cas discret

Fonction valeur dans le cas continu

- La fonction valeur est définie comme

$$V^{\pi}(s_\tau) = \int_t^{\infty} \gamma^{t-\tau} \mathcal{R}_{s_t \pi(s_t)} \, dt$$

  - On suppose une politique déterministe : $\pi : \mathcal{S} \rightarrow \mathcal{A}$

$$\begin{aligned} V^{\pi}(s_\tau) &= \int_t^{\infty} e^{-\Gamma(t-\tau)} \mathcal{R}_{s_t \pi(s_t)} \, dt \\ &= \int_t^{\tau+\Delta t} e^{-\Gamma(t-\tau)} \mathcal{R}_{s_t \pi(s_t)} \, dt + \int_{\tau+\Delta t}^{\infty} e^{-\Gamma(t-\tau)} \mathcal{R}_{s_t \pi(s_t)} \, dt \\ &= \int_t^{\tau+\Delta t} e^{-\Gamma(t-\tau)} \mathcal{R}_{s_t \pi(s_t)} \, dt + e^{-\Gamma(t-\tau)} V^{\pi}(s_{\tau+\Delta t}) \end{aligned}$$

avec $\Gamma \in \mathbb{R}^+$ et $\Gamma = -\ln \gamma$

Intégration

- **Solution 1 : faire tendre Δt vers 0**

$$V^\pi(s_\tau) \approx \mathcal{R}_{s_\tau \pi(s_\tau)} + e^{-\Gamma \delta t} V^\pi(s_{\tau+\delta t})$$

$$\Gamma V^\pi(s_\tau) = \mathcal{R}_{s_\tau \pi(s_\tau)} + \frac{\partial V^\pi}{\partial s} \mathcal{T}^s_{\pi(s_\tau)}$$

Équation de Hamilton/Jacobi/Bellman

$$\Gamma V^\star(s) = \max_{a \in \mathcal{A}(s)} \left[\mathcal{R}_{sa} + \frac{\partial V^\star}{\partial s} \mathcal{T}^s_a \right]$$

- **Solution 2 : supposer $\mathcal{R}$ constant pendant Δt**

$$\begin{aligned} V^\pi(s_\tau) &= \mathcal{R}_{s_\tau \pi(s_\tau)} \int_t^{\tau+\Delta t} e^{-\Gamma(t-\tau)}\, dt + e^{-\Gamma(t-\tau)} V^\pi(s_{\tau+\Delta t}) \\ &= \frac{1 - e^{-\Gamma \Delta t}}{\Gamma} \mathcal{R}_{s_\tau \pi(s_\tau)} + e^{-\Gamma \Delta t} V^\pi(s_{\tau+\Delta t}) \end{aligned}$$

Lexique

- AR en temps continu : **continuous time RL**