Théorie des Jeux

Apprentissage par renforcement

Alexandre Aussem

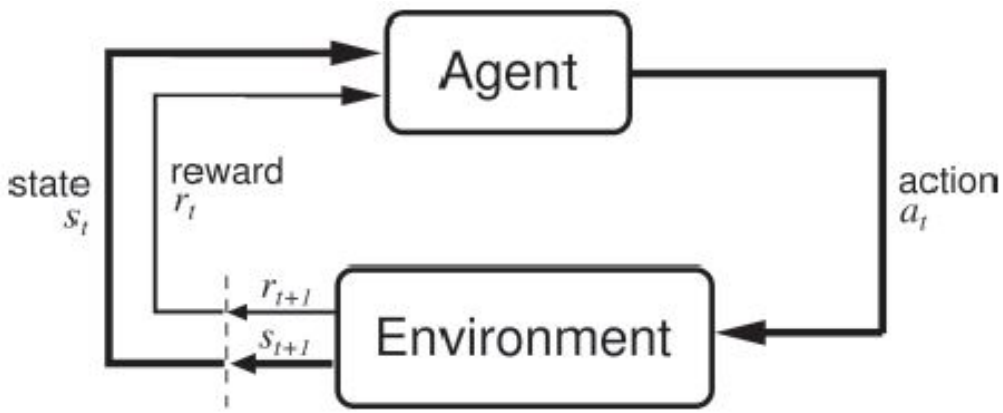
http://perso.univ-lyon1.fr/alexandre.aussem/

13 juin 2014

Interaction agent-environnement

Agent en interaction avec le monde suivant trois canaux : perception, renforcement immédiat et action instantanée.

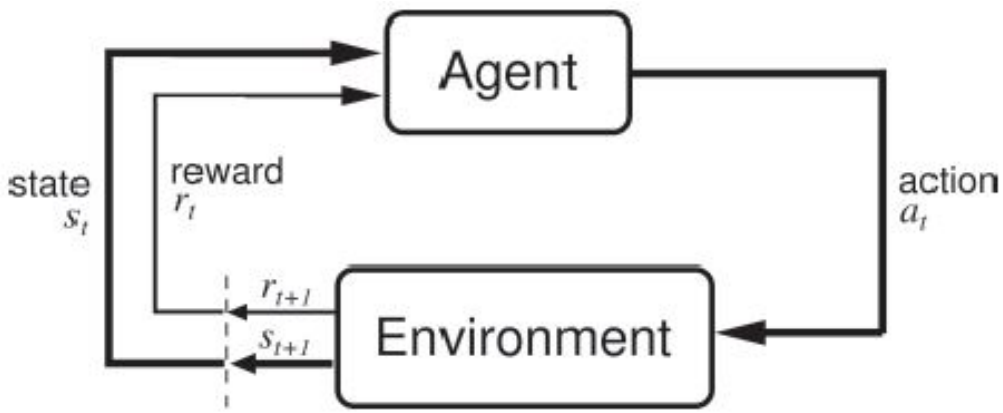
- Un canal perceptif par lequel l'agent **mesure l'état courant** s_t dans lequel il se trouve dans l'environnement.



Interaction agent-environnement

Agent en interaction avec le monde suivant trois canaux : perception, renforcement immédiat et action instantanée.

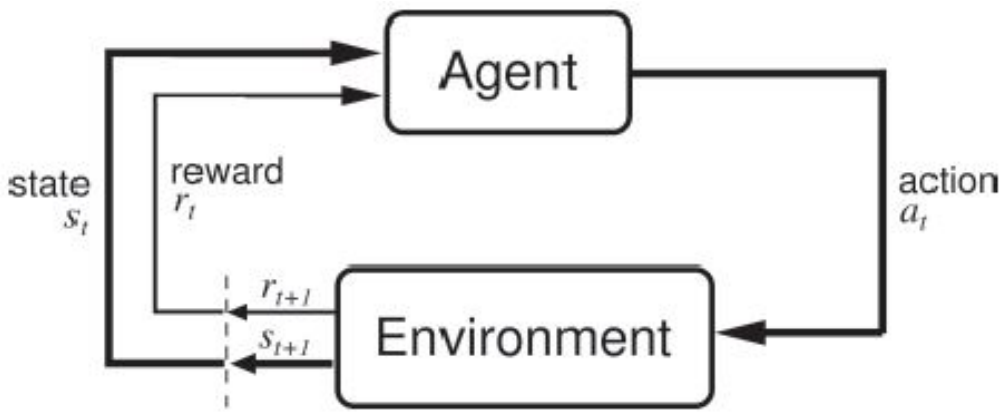
- Un canal perceptif par lequel l'agent **mesure l'état courant** s_t dans lequel il se trouve dans l'environnement.
- Un canal spécifique aux **signaux de renforcement** (ou **récompense immédiate**) renseignant l'agent sur la qualité de l'état courant, r_t.



Interaction agent-environnement

Agent en interaction avec le monde suivant trois canaux : perception, renforcement immédiat et action instantanée.

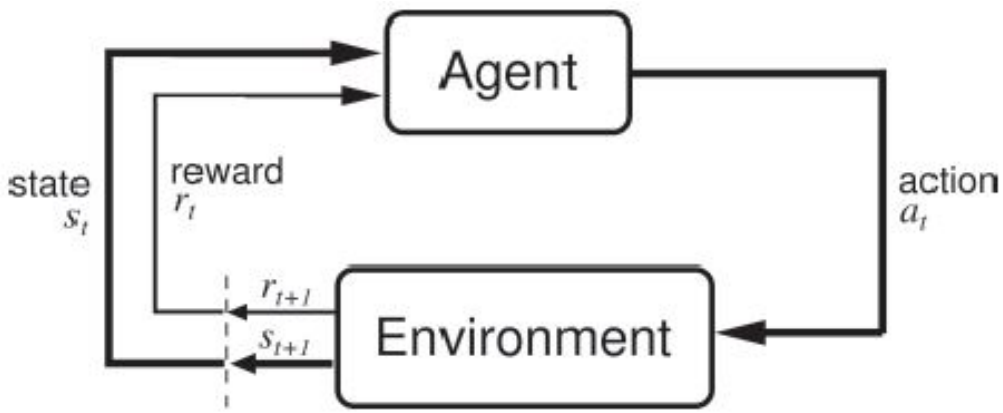
- Un canal perceptif par lequel l'agent **mesure l'état courant** s_t dans lequel il se trouve dans l'environnement.
- Un canal spécifique aux **signaux de renforcement** (ou **récompense immédiate**) renseignant l'agent sur la qualité de l'état courant, r_t.
- Un canal qui transmet à l'environnement l'action de l'agent, a_t.



Interaction agent-environnement

Agent en interaction avec le monde suivant trois canaux : perception, renforcement immédiat et action instantanée.

- Un canal perceptif par lequel l'agent **mesure l'état courant** s_t dans lequel il se trouve dans l'environnement.
- Un canal spécifique aux **signaux de renforcement** (ou **récompense immédiate**) renseignant l'agent sur la qualité de l'état courant, r_t.
- Un canal qui transmet à l'environnement l'action de l'agent, a_t.



L'interaction avec l'environnnement, grâce aux informations sur les conséquences de nos actions, nous aide à accomplir nos objectifs.

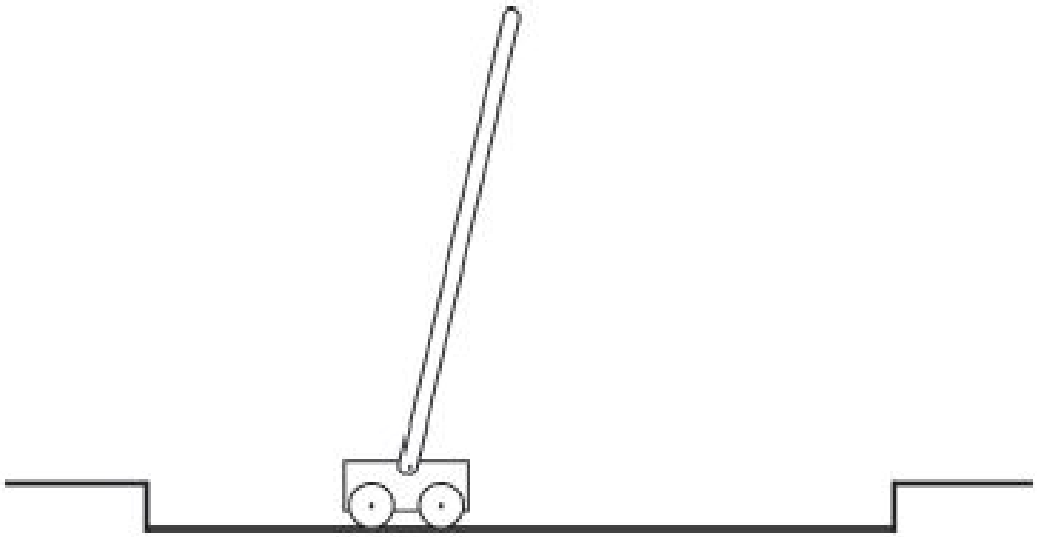
Quel objectif ?

- Maximiser un gain cummulatif sur le long terme $r_{t+1}, r_{t+2}, \ldots$
- Sur une période T si tâche épisodique (e.g, une partie, sortir d'un labyrinte), ou $T = \infty$ si tâche continue, pas de terme.
- Concept additionnel : le "discount rate" $0 \leq \gamma \leq 1$. Interprétation : un gain futur r_{t+k} vaut $\gamma^k r_{t+k}$ maintenant.
- Si $\gamma \to 0$, vision à court terme ou "myope",
- Si $\gamma \to 1$, vision à long terme,

Important : la maximisation du gain doit coïncider avec la réalisation de nos objectifs, mais ne doit pas contenir de connaissance a priori sur la façon de les réaliser.

Illustration : pole-balancing

Un wagonet sur lequel est posé en équilibre un piquet. Le wagonet peut se déplacer entre deux limietes. L'objectif est de tenir le piquet le plus longtemps à l'équilibre.

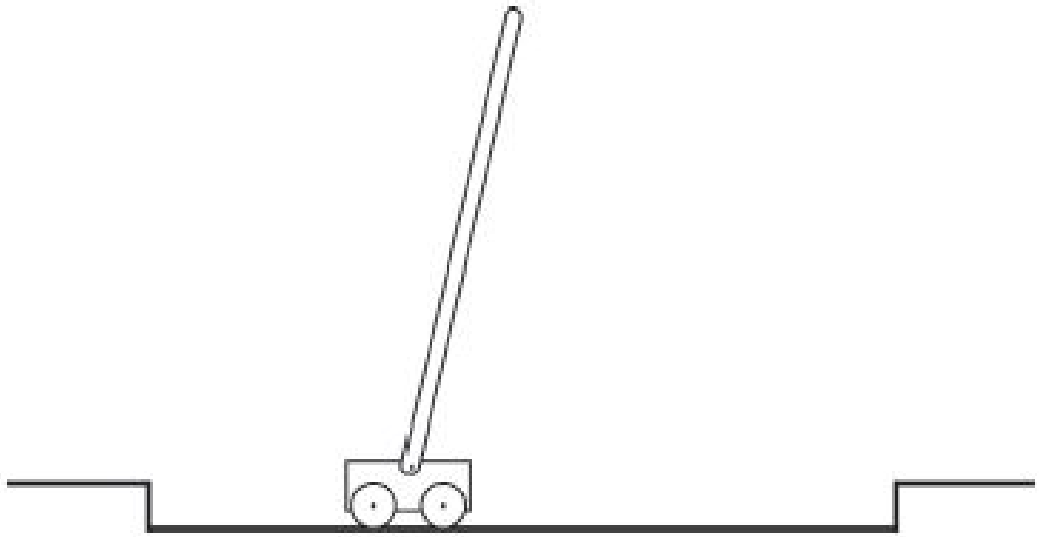


Deux alternatives :

- Tâche épisodique où chaque épisode dure le temps jusqu'à la chute du piquet avec $r_t = 1$ jusqu'à la chute.

Illustration : pole-balancing

Un wagonet sur lequel est posé en équilibre un piquet. Le wagonet peut se déplacer entre deux limietes. L'objectif est de tenir le piquet le plus longtemps à l'équilibre.



Deux alternatives :

- Tâche épisodique où chaque épisode dure le temps jusqu'à la chute du piquet avec $r_t = 1$ jusqu'à la chute.
- Tâche continue avec *discounting*. Dans ce cas, $r_t = -1$ en cas de chute et $r_t = 0$ sinon.

Les fonctions de gain

Différentes fonction de gain sont envisageables...

$$R_t = \sum_{i=t+1}^{T} r_i$$

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

$$R_t = \frac{1}{T-1} \sum_{i=t+1}^{T} r_i$$

Le choix de γ est laissé à l'experimentateur.

- On considère un temps s'écoulant de manière discrète. Un instant est noté $t \in \mathbb{N}$.
- La situation : un agent interagit avec son environnement. À chaque instant $t \in \mathbb{N}$, il se situe dans un état s_t et choisit une action a_t. Suite à cela, il perçoit un gain r_t (*return*), et se situe dans un nouvel état s_{t+1}.

Exemples de problèmes

- un agent logiciel qui explore la Toile afin de trouver des informations pertinentes sur un sujet donné ;
- un agent logiciel qui joue aux backgammon, au go, aux échecs etc.
- un robot qui ramasse les boîtes de boisson vides dans des bureaux ;
- un robot qui peint des carrosseries de voiture ;
- un robot qui explore une planète lointaine.

Def. Problèmes de décision de Markov (PDM)

On définit un problème de décision de Markov (PDM) par un quadruplet : $\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}$ où :

- $\mathcal{S}$ est un ensemble d'états fini ;

Def. Problèmes de décision de Markov (PDM)

On définit un problème de décision de Markov (PDM) par un quadruplet : $\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}$ où :

- $\mathcal{S}$ est un ensemble d'états fini ;
- $\mathcal{A}$ est un ensemble d'actions fini. On note $\mathcal{A}(s)$ l'ensemble des actions possibles dans l'état s.

Def. Problèmes de décision de Markov (PDM)

On définit un problème de décision de Markov (PDM) par un quadruplet : $\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}$ où :

- $\mathcal{S}$ est un ensemble d'états fini ;
- $\mathcal{A}$ est un ensemble d'actions fini. On note $\mathcal{A}(s)$ l'ensemble des actions possibles dans l'état s.
- $\mathcal{P}$ la fonction de transition décrivant la dynamique de l'environnement :

$$\begin{aligned} \mathcal{P} : \mathcal{S}x\mathcal{A}x\mathcal{S} &\rightarrow [0,1] \\ (s,a,s') &\rightarrow \mathcal{P}(s,a,s') = Pr[s_{t+1} = s' | s_t = s; a_t = a] \end{aligned}$$

Def. Problèmes de décision de Markov (PDM)

On définit un problème de décision de Markov (PDM) par un quadruplet : $\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}$ où :

- $\mathcal{S}$ est un ensemble d'états fini ;
- $\mathcal{A}$ est un ensemble d'actions fini. On note $\mathcal{A}(s)$ l'ensemble des actions possibles dans l'état s.
- $\mathcal{P}$ la fonction de transition décrivant la dynamique de l'environnement :

$$\begin{aligned} \mathcal{P} : \mathcal{S}x\mathcal{A}x\mathcal{S} &\rightarrow [0,1] \\ (s,a,s') &\rightarrow \mathcal{P}(s,a,s') = Pr[s_{t+1} = s' | s_t = s; a_t = a] \end{aligned}$$

- $\mathcal{R}$ la fonction de gain :

$$\begin{aligned} \mathcal{R} : \mathcal{S}x\mathcal{A}x\mathcal{S} &\rightarrow \mathbb{R} \\ (s,a,s') &\rightarrow \mathcal{R}(s,a,s') = \mathbb{E}\left[r_{t+1} | s_{t+1} = s'; s_t = s; a_t = a\right] \end{aligned}$$

Politique

Def. politique (ou stratégie)

On définit une politique π comme une application :

$$\begin{aligned} \pi : \mathcal{S}x\mathcal{A} &\rightarrow [0,1] \\ (s,a) &\rightarrow \pi(s,a) = Pr[a_t = s | s_t = s] \end{aligned}$$

On utilise également la notion de politique déterministe qui associe une action à un état (et non une probabilité d'émission des actions pour chaque état).

$$\begin{aligned} \pi : \mathcal{S} &\rightarrow \mathcal{A} \\ s &\rightarrow \pi(s) = a \end{aligned}$$

Renforcement : spécifie comment l'agent doit moifier sa politique π_t au vu de son expérience.

- Il peut exister un ensemble d'états finaux $\mathcal{F} \subset \mathcal{S}$: quand l'agent atteint l'un de ces états, sa tâche est terminée. On parle alors de tâche épisodique, ou à horizon fini.
- On supposera que $\mathcal{P}$ et $\mathcal{R}$ sont de sont pas fonction du temps : **processus stationnaire**.
- Le cas déterministe est un cas particulier de la définition précédente.
- On observe que l'état courant et le gain courant ne dépendent que de l'état précédent et de l'action qui vient d'être émise. Cette propriété fondamentale caractérise un **un problème de décision de Markov**. On parle aussi d'environnement *markovien*.

Objectif du cours : trouver la politique que doit suivre l'agent pour maximiser le gain cummulé. Cette politique est qualifiée d'optimale. On la note $\pi^\star$

Propriété

$\pi^\star$ existe pour tout PDM

On cherche donc à résoudre un **problème de contrôle optimal**. C'est un problème très général.
L'apprentissage par renforcement se compose d'un ensemble de concepts et d'algorithmes qui permettent de résoudre ce problème. Il existe d'autres méthodes.
La force de l'apprentissage par renforcement par rapport à d'autres méthodes est que la seule connaissance de S et A suffisent à résoudre ce problème : la connaissance de la dynamique de l'environnement (S et S) n'est **pas nécessaire**.
Remarquons que l'apprentissage par renforcement ne se définit pas par une certaine classe d'algorithmes mais par le problème qu'il cherche à résoudre, celui du contrôle optimal.

Il faut distinguer plusieurs cas, dans l'ordre croissant de difficulté :

- le PDM est complètement spécifié (résolution par programmation dynamique).

Domaines actuels de recherche (2013) : "Reinforcement Learning with Generalized Feedback : Beyond Numeric Rewards. Learning from qualitative feedback (e.g., ordinal MDPs or preference-based reinforcement learning), and learning from multiple feedback signals (e.g., multi-objective reinforcement learning)."

Il faut distinguer plusieurs cas, dans l'ordre croissant de difficulté :

- le PDM est complètement spécifié (résolution par programmation dynamique).
- S et A sont inconnus

Domaines actuels de recherche (2013) : "Reinforcement Learning with Generalized Feedback : Beyond Numeric Rewards. Learning from qualitative feedback (e.g., ordinal MDPs or preference-based reinforcement learning), and learning from multiple feedback signals (e.g., multi-objective reinforcement learning)."

Il faut distinguer plusieurs cas, dans l'ordre croissant de difficulté :

- le PDM est complètement spécifié (résolution par programmation dynamique).
- S et A sont inconnus
- S, A ou le temps, sont continus (prennent une infinité de valeurs). Discrétisation pour se remplacer dans le cas fini ou résolution directe dans le cas continu.

Domaines actuels de recherche (2013) : "Reinforcement Learning with Generalized Feedback : Beyond Numeric Rewards. Learning from qualitative feedback (e.g., ordinal MDPs or preference-based reinforcement learning), and learning from multiple feedback signals (e.g., multi-objective reinforcement learning)."

Il faut distinguer plusieurs cas, dans l'ordre croissant de difficulté :

- le PDM est complètement spécifié (résolution par programmation dynamique).
- S et A sont inconnus
- S, A ou le temps, sont continus (prennent une infinité de valeurs). Discrétisation pour se remplacer dans le cas fini ou résolution directe dans le cas continu.
- L'état perçu n'est pas tout à fait le reflet de l'état réel de l'environnement.

Domaines actuels de recherche (2013) : "Reinforcement Learning with Generalized Feedback : Beyond Numeric Rewards. Learning from qualitative feedback (e.g., ordinal MDPs or preference-based reinforcement learning), and learning from multiple feedback signals (e.g., multi-objective reinforcement learning)."

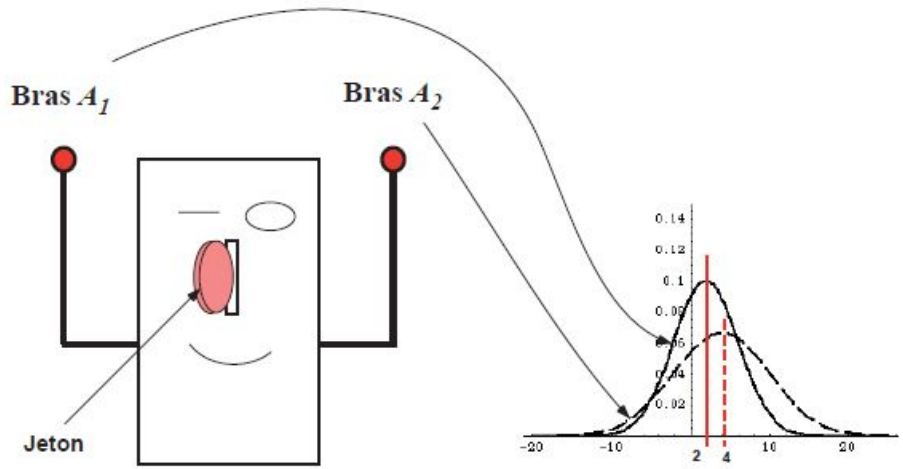
Il faut distinguer plusieurs cas, dans l'ordre croissant de difficulté :

- le PDM est complètement spécifié (résolution par programmation dynamique).
- S et A sont inconnus
- S, A ou le temps, sont continus (prennent une infinité de valeurs). Discrétisation pour se remplacer dans le cas fini ou résolution directe dans le cas continu.
- L'état perçu n'est pas tout à fait le reflet de l'état réel de l'environnement.
- L'environnement est non stationnaire.

Domaines actuels de recherche (2013) : "Reinforcement Learning with Generalized Feedback : Beyond Numeric Rewards. Learning from qualitative feedback (e.g., ordinal MDPs or preference-based reinforcement learning), and learning from multiple feedback signals (e.g., multi-objective reinforcement learning)."

Le dilemme exploration contre exploitation

Contexte : Un agent ne connaît pas son environnement et cherche à maximiser une mesure de gain cumulée.
Considérons cette machine à sous appelée "bandit à deux bras". Chacun des bras est associé à un gain aléatoire. Le gain du bras A1 suit une loi $\mathcal{N}(4, 6)$, alors que le gain de A2 suit une loi $\mathcal{N}(2, 4)$. Le joueur cherche à maximiser son gain avec ses m jetons.



Quelle est la stratégie optimale en fonction de son estimation courante des moyenne et variance de chaque bras ?

Fonctions d'évaluation

Valeur d'un état

On définit la valeur d'un état s pour une politique fixée π par :

$$V^{\pi}(s) = \mathbb{E}\left[R_t | s_t = s\right]$$

Valeur d'un état

On définit la valeur d'un état s pour une politique fixée π par :

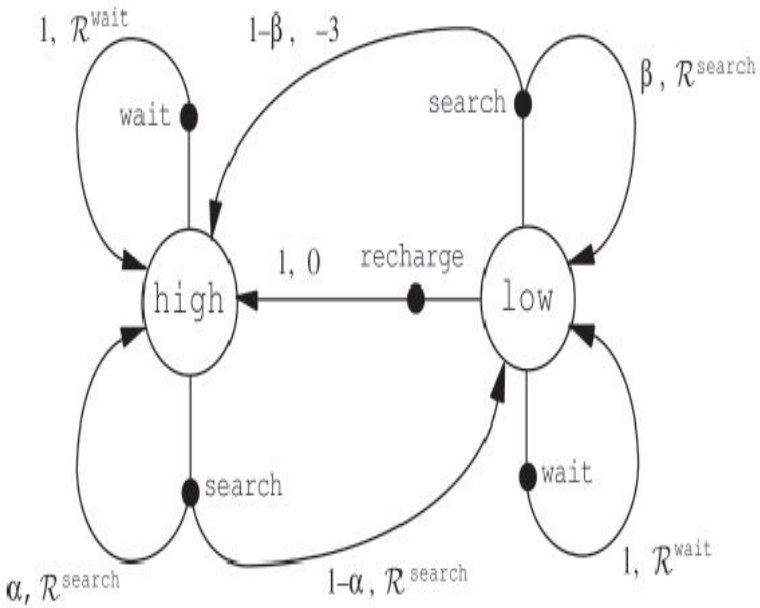
$$Q^{\pi}(s, a) = \mathbb{E}\left[R_t | s_t = s, a_t = a\right]$$

Il y a naturellement une forte relation entre ces deux fonctions

$$Q^{\pi}(s, a) = \sum_{s' \in \mathcal{S}} \mathcal{P}(s, a, s')[\mathcal{R}(s, a, s') + \gamma V^{\pi}(s')]$$

Illustration : Robot de recyclage

Un robot collecte des canettes dans des bureaux. En fonction de son niveau de batterie il a le choix entre : chercher des canettes, attendre qu'on lui apporte et rentrer se recharger.



$s = s_t$	$s' = s_{t+1}$	$a = a_t$	$\mathcal{P}^a_{ss'}$	$\mathcal{R}^a_{ss'}$
high	high	search	α	$\mathcal{R}^{search}$
high	low	search	$1-\alpha$	$\mathcal{R}^{search}$
low	high	search	$1-\beta$	-3
low	low	search	β	$\mathcal{R}^{search}$
high	high	wait	1	$\mathcal{R}^{wait}$
high	low	wait	0	$\mathcal{R}^{wait}$
low	high	wait	0	$\mathcal{R}^{wait}$
low	low	wait	1	$\mathcal{R}^{wait}$
low	high	recharge	1	0
low	low	recharge	0	0

Fonctions d'évaluation

Dans le cas d'un tâche continue, l'espérance s'exprime sur tous les couples états-actions futurs :

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}\left[R_t | s_t = s\right] \\ &= \lim_{N\to\infty} \sum_{a_t, s_{t+1}, \ldots, a_{N-1}, s_N} \prod_{j=t}^{N-1} \pi(s_j, a_j) \mathcal{P}^{a_j}_{s_j, s_{j+1}} \mathcal{R}^{a_j}_{s_j, s_{j+1}} \gamma^{j-t} \end{aligned}$$

avec la notation :

$$\begin{aligned} \mathcal{R}^{a}_{s,s'} &= \mathcal{R}(s, a, s') \\ \mathcal{P}^{a}_{s,s'} &= \mathcal{P}(s, a, s') \end{aligned}$$

Le calcul direct est rédhibitoire.

Equations de Bellman pour $V^\pi(s)$

$$
\begin{aligned}
V^\pi(s) &= \mathbb{E}\left[R_t | s_t = s\right] \\
&= \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right] \\
&= \mathbb{E}\left[r_{t+1} + \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\right] \\
&= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{s,s'} \left(\mathcal{R}^a_{s,s'} + \gamma \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s'\right]\right) \\
&= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{s,s'} [\mathcal{R}^a_{s,s'} + \gamma V^\pi(s')]
\end{aligned}
$$

Introduit une forte contrainte entre les états successifs (voisins). C'est un système de $|\mathcal{S}|$ **équations linéaires**.

Equations d'optimalité de Bellman

Relation, d'ordre

On définit une relation, d'ordre entre les stratégies π :

$$\pi \geq \pi' \iff V^{\pi}(s) \geq V^{\pi'} \quad \forall s \in \mathcal{S}$$

On peut exprimer la fonction valeur de la politique optimale, $V^{\star}$, par :

Equation d'optimalité de Bellman

$$V^{\star}(s) = \max_{\pi} V^{\pi}(s) \quad \forall s \in \mathcal{S}$$

et

$$Q^{\star}(s,a) = \max_{\pi} Q^{\pi}(s,a) \quad \forall s \in \mathcal{S},\ \forall a \in \mathcal{A}(s)$$

Equations d'optimalité de Bellman

$$
\begin{aligned}
V^\star(s) &= \max_{a \in \mathcal{A}(s)} Q^\star(s,a) \\
&= \max_{a \in \mathcal{A}(s)} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right]_{\pi^\star} \\
&= \max_{a \in \mathcal{A}(s)} \mathbb{E}\left[r_{t+1} + \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s, a_t = a\right]_{\pi^\star} \\
&= \max_{a \in \mathcal{A}(s)} \sum_{s'} \mathcal{P}^a_{s,s'}[\mathcal{R}^a_{s,s'} + \gamma V^\star(s')]
\end{aligned}
$$

$$
\begin{aligned}
Q^\star(s,a) &= \mathbb{E}\left[r_{t+1} + \gamma^k \max_{a \in \mathcal{A}(s)} Q^\star(s_{t+1}), a') | s_t = s, a_t = a\right] \\
&= \sum_{s'} \mathcal{P}^a_{s,s'}[\mathcal{R}^a_{s,s'} + \gamma \max_{a \in \mathcal{A}(s)} Q^\star(s', a')]
\end{aligned}
$$

Equations d'optimalité de Bellman

Equations d'optimalité de Bellman

$$V^\star(s) = \max_{a \in \mathcal{A}(s)} \sum_{s'} \mathcal{P}^a_{s,s'} [\mathcal{R}^a_{s,s'} + \gamma V^\star(s')]$$

$$Q^\star(s,a) = \sum_{s'} \mathcal{P}^a_{s,s'} [\mathcal{R}^a_{s,s'} + \gamma \max_{a \in \mathcal{A}(s)} Q^\star(s',a')]$$

Propriété

Pour un PDM fini, il existe une solution unique aux équations d'optimalité de Bellman associées à la fonction valeur $V^\star(s)$. C'est un système de N équations et N inconnues. De $V^\star(s)$, on en déduit rapidement le ou les actions optimales.

- Un politique qui assigne une probabilité non nulle aux actions optimales est optimale.

Equations d'optimalité de Bellman

Equations d'optimalité de Bellman

$$V^\star(s) = \max_{a \in \mathcal{A}(s)} \sum_{s'} \mathcal{P}^a_{s,s'} [\mathcal{R}^a_{s,s'} + \gamma V^\star(s')]$$

$$Q^\star(s,a) = \sum_{s'} \mathcal{P}^a_{s,s'} [\mathcal{R}^a_{s,s'} + \gamma \max_{a \in \mathcal{A}(s)} Q^\star(s',a')]$$

Propriété

Pour un PDM fini, il existe une solution unique aux équations d'optimalité de Bellman associées à la fonction valeur $V^\star(s)$. C'est un système de N équations et N inconnues. De $V^\star(s)$, on en déduit rapidement le ou les actions optimales.

- Un politique qui assigne une probabilité non nulle aux actions optimales est optimale.
- Une politique *greedy* par rapport à $V^\star$ est optimale. $V^\star$ prend déjà en compte les conséquences futures des actions.

Principe d'apprentissage par renforcement

1. Apprendre à calculer V^π pour une politique π et un PDM donnés ;

Principe d'apprentissage par renforcement

1. Apprendre à calculer V^{π} pour une politique π et un PDM donnés ;
2. A partir de là, apprendre comment trouver une politique optimale $V^{\star}$

Programmation dynamique

La **programmation dynamique** est une technique algorithmique pour optimiser des sommes de fonctions monotones croissantes sous contrainte. Elle s'appuie sur un principe simple :

> Toute solution optimale s'appuie elle-même sur des sous-problèmes résolus localement de façon optimale. Sa résolution s'effectue de "bas en haut" : on calcule d'abord les solutions des sous-problèmes les plus petits pour ensuite déduire petit à petit les solutions de l'ensemble.

Principe déjà vu dans l'**algorithme de Bellman** pour déterminer des chemins optimaux dans les graphes orientés sans circuit.

Evaluation de la politique sur un horizon fini

Sur un horizon N, les équations de Bellman pour une politique π s'écrivent :

$$\begin{aligned} V^{\pi}_{0,N}(s) &= \mathbb{E}\left[R_0 | s_0 = s\right] \\ &= \mathbb{E}\left[\sum_{k=0}^{N-1} r_{k+1} | s_t = s\right] \\ &= \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{s,s'}[\mathcal{R}^{a}_{s,s'} + V^{\pi}_{1,N}(s')] \end{aligned}$$

Principe de résolution par **programmation dynamique** pour trouver $\pi^\star$: partir de $k = N, N-1, \ldots, 0$

$$\begin{aligned} Q^{\star}_{k,N}(s,a) &= \sum_{s' \in \mathcal{S}} \mathcal{P}^{a}_{s,s'}[\mathcal{R}^{a}_{s,s'} + V^{\star}_{k+1,N}(s')], \\ \pi^{\star}_{k}(s) &= \arg \max_{a \in \mathcal{A}(s)} Q^{\star}_{k,N}(s,a) \end{aligned}$$

avec $V^{\star}_{k,N}(s) = Q^{\star}_{k,N}(s, \pi^{\star}_{k}(s))$ si $k < N$ et $V^{\star}_{k,N}(s) = 0$ si k=N.

Evaluation de la politique sur un horizon infini

Sur un horizon infini, le gain terminal n'existe pas, il n'y a donc plus d'effet de bord.
Pour calculer la fonction valeur associée à une politique π fixée, on applique directement l'équation de Bellman . Cette équation se transcrit en une équation définissant une suite récurrente V_k^π qui converge vers $V^\star$:

$$V_{k+1}^\pi(s) = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}_{s,s'}^a [\mathcal{R}_{s,s'}^a + \gamma V_k^\pi(s')]$$

L'indice k désigne ici l'itération de l'algorithme et non le temps.

Algorithme 1 L'algorithme d'évaluation de la politique (stochastique ici).

Nécessite: un PDM : $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$

Nécessite: une politique π

Nécessite: un seuil de précision ϵ

initialiser V_0^π aléatoirement

$i \leftarrow 0$

répéter

pour tout état $s \in \mathcal{S}$ **faire**

$V_{i+1}^\pi(s) \leftarrow \sum_{a \in \mathcal{A}(s)} \pi(s,a) \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s')[\mathcal{R}(s,a,s') + \gamma V_i^\pi(s')]$

fin pour

$i \leftarrow i + 1$

jusque $||V_i^\pi - V_{i-1}^\pi|| \leq \epsilon$

L'algorithme d'évaluation de la politique

Le choix de la norme ne change rien à la convergence de l'algorithme. Les normes usuelles :

- norme $\mathbf{L_1}$: $\|V_i^\pi - V_{i-1}^\pi\|_1 = \sum_{s\in\mathcal{S}} |V_i^\pi(s) - V_{i-1}^\pi(s)|$
- norme $\mathbf{L_2}$: $\|V_i^\pi - V_{i-1}^\pi\|_2 = \sum_{s\in\mathcal{S}} \left(V_i^\pi(s) - V_{i-1}^\pi(s)\right)^2$
- norme $\mathbf{L_\infty}$: $\|V_i^\pi - V_{i-1}^\pi\|_\infty = \max_{s\in\mathcal{S}} |V_i^\pi(s) - V_{i-1}^\pi(s)|$

Propriété

Pour un PDM et une politique π fixés, l'algorithme d'évaluation de la politique converge asymptotiquement vers V^π.

L'algorithme d'amélioration de la politique

Le principe de l'**amélioration de la politique** consiste à choisir pour chaque état $s \in \mathcal{S}$, l'action qui maximise l'espérance de retours calculé à partir de la politique courante π, soit :

$$\pi'(s,a) \leftarrow \arg \max_{a \in \mathcal{A}(s)} \sum_{s'} \mathcal{P}^a_{s,s'} [\mathcal{R}^a_{s,s'} + \gamma V^\pi(s')]$$

On montre (en excercice) que $V^{\pi'}(s) \geq V^\pi(s)$, $\forall s \in \mathcal{S}$ et donc que $\pi' \geq \pi$

Propriété

Lorsque $\pi = \pi'$ alors $\pi = \pi^\star$.

- démarrer avec une politique déterministe quelconque

L'algorithme d'amélioration de la politique

Le principe de l'**amélioration de la politique** consiste à choisir pour chaque état $s \in \mathcal{S}$, l'action qui maximise l'espérance de retours calculé à partir de la politique courante π, soit :

$$\pi'(s,a) \leftarrow \arg \max_{a \in \mathcal{A}(s)} \sum_{s'} \mathcal{P}^a_{s,s'} [\mathcal{R}^a_{s,s'} + \gamma V^\pi(s')]$$

On montre (en excercice) que $V^{\pi'}(s) \geq V^\pi(s)$, $\forall s \in \mathcal{S}$ et donc que $\pi' \geq \pi$

Propriété

Lorsque $\pi = \pi'$ alors $\pi = \pi^\star$.

- démarrer avec une politique déterministe quelconque
- calculer sa valeur

L'algorithme d'amélioration de la politique

Le principe de l'**amélioration de la politique** consiste à choisir pour chaque état $s \in \mathcal{S}$, l'action qui maximise l'espérance de retours calculé à partir de la politique courante π, soit :

$$\pi'(s,a) \leftarrow \arg \max_{a \in \mathcal{A}(s)} \sum_{s'} \mathcal{P}^a_{s,s'} [\mathcal{R}^a_{s,s'} + \gamma V^\pi(s')]$$

On montre (en excercice) que $V^{\pi'}(s) \geq V^\pi(s)$, $\forall s \in \mathcal{S}$ et donc que $\pi' \geq \pi$

Propriété

Lorsque $\pi = \pi'$ alors $\pi = \pi^\star$.

- démarrer avec une politique déterministe quelconque
- calculer sa valeur
- construire une politique **déterministe** meilleure que la précédente

L'algorithme d'amélioration de la politique

Le principe de l'**amélioration de la politique** consiste à choisir pour chaque état $s \in \mathcal{S}$, l'action qui maximise l'espérance de retours calculé à partir de la politique courante π, soit :

$$\pi'(s,a) \leftarrow \arg \max_{a \in \mathcal{A}(s)} \sum_{s'} \mathcal{P}^a_{s,s'} [\mathcal{R}^a_{s,s'} + \gamma V^\pi(s')]$$

On montre (en excercice) que $V^{\pi'}(s) \geq V^\pi(s)$, $\forall s \in \mathcal{S}$ et donc que $\pi' \geq \pi$

Propriété

Lorsque $\pi = \pi'$ alors $\pi = \pi^\star$.

- démarrer avec une politique déterministe quelconque
- calculer sa valeur
- construire une politique **déterministe** meilleure que la précédente
- reboucler tant que l'on arrive à produire une politique strictement meilleure.

Algorithme 2 L'algorithme d'itération de la politique.

Nécessite: un PDM : $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$

Nécessite: un seuil de précision ϵ

initialiser π_0 aléatoirement

$k \leftarrow 0$

répéter

initialiser V_0^{π} aléatoirement

$i \leftarrow 0$

répéter

pour tout état $s \in \mathcal{S}$ **faire**

$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{P}(s, \pi_k(s), s')[\mathcal{R}(s, \pi(s), s') + \gamma V_i^{\pi_k}(s)]$

fin pour

$i \leftarrow i + 1$

jusque $||V_i^{\pi_k} - V_{i-1}^{\pi_k}|| \leq \epsilon$

pour tout état $s \in \mathcal{S}$ **faire**

$\pi_{k+1}(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} \mathcal{P}(s, a, s')[\mathcal{R}(s, a, s') + \gamma V^{\pi_k}(s')]$

fin pour

$k \leftarrow k + 1$

jusque $\pi_k = \pi_{k-1}$

Convergence

Propriété

L'algorithme d'itération de la politique converge vers $\pi^\star$ au bout d'un nombre fini d'itérations

En pratique, le choix de ϵ repose sur un compromis.

- si ϵ est trop petit, convergence trop lente

'algorithme d'itération de la politique peut être optimisé de différentes manières. On peut par exemple chercher directement $V^\star$ et en déduire $\pi^\star$ plus rapidement. C'est l'algorithme d'itération de la valeur.

Convergence

Propriété

L'algorithme d'itération de la politique converge vers $\pi^\star$ au bout d'un nombre fini d'itérations

En pratique, le choix de ϵ repose sur un compromis.

- si ϵ est trop petit, convergence trop lente
- si ϵ est trop grand, on peut avoir $\pi_k = \pi_{k-1} \neq \pi^\star$

'algorithme d'itération de la politique peut être optimisé de différentes manières. On peut par exemple chercher directement $V^\star$ et en déduire $\pi^\star$ plus rapidement. C'est l'algorithme d'itération de la valeur.

Convergence

Propriété

L'algorithme d'itération de la politique converge vers $\pi^\star$ au bout d'un nombre fini d'itérations

En pratique, le choix de ϵ repose sur un compromis.

- si ϵ est trop petit, convergence trop lente
- si ϵ est trop grand, on peut avoir $\pi_k = \pi_{k-1} \neq \pi^\star$

L'algorithme d'itération de la politique peut être optimisé de différentes manières. On peut par exemple chercher directement $V^\star$ et en déduire $\pi^\star$ plus rapidement. C'est l'algorithme d'itération de la valeur.

Algorithme 3 L'algorithme d'itération de la valeur.

Nécessite: un PDM : $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$

Nécessite: un seuil de précision ϵ

initialiser V_0 aléatoirement

$i \leftarrow 0$

répéter

pour tout état $s \in \mathcal{S}$ **faire**

$V_{i+1}(s) \leftarrow \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} \mathcal{P}(s, a, s')[\mathcal{R}(s, a, s') + \gamma V_i(s')]$

fin pour

$i \leftarrow i + 1$

jusque $||V_i - V_{i-1}|| \leq \epsilon$

pour tout état $s \in \mathcal{S}$ **faire**

$\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} \mathcal{P}(s, a, s')[\mathcal{R}(s, a, s') + \gamma V(s')]$

fin pour

Implantation & difficultés de résolution

Les algorithmes que l'on vient de rencontrer sont très faciles à implanter mais reposent sur la connaissance de l'environnement $\mathcal{P}$ et $\mathcal{R}$.
La grande diffculté concerne la mise sous forme d'un PDM du problème que l'on veut résoudre.

- la recherche directe de la politique optimale est en $O(|\mathcal{S}|^{|\mathcal{A}|})$ (c'est un problème NP-difficile) ;

Implantation & difficultés de résolution

Les algorithmes que l'on vient de rencontrer sont très faciles à implanter mais reposent sur la connaissance de l'environnement $\mathcal{P}$ et $\mathcal{R}$.
La grande diffculté concerne la mise sous forme d'un PDM du problème que l'on veut résoudre.

- la recherche directe de la politique optimale est en $O(|\mathcal{S}|^{|\mathcal{A}|})$ (c'est un problème NP-difficile) ;
- en pratique : on peut appliquer ces algorithmes à des PDM ayant jusque de l'ordre de 10^6 états ;

Implantation & difficultés de résolution

Les algorithmes que l'on vient de rencontrer sont très faciles à implanter mais reposent sur la connaissance de l'environnement $\mathcal{P}$ et $\mathcal{R}$.
La grande diffculté concerne la mise sous forme d'un PDM du problème que l'on veut résoudre.

- la recherche directe de la politique optimale est en $O(|\mathcal{S}|^{|\mathcal{A}|})$ (c'est un problème NP-difficile) ;
- en pratique : on peut appliquer ces algorithmes à des PDM ayant jusque de l'ordre de 10^6 états ;
- la complexité en temps de calcul est $O(|\mathcal{A}| \cdot |\mathcal{S}|)$ pour l'algorithme d'itération de la valeur ;

Implantation & difficultés de résolution

Les algorithmes que l'on vient de rencontrer sont très faciles à implanter mais reposent sur la connaissance de l'environnement $\mathcal{P}$ et $\mathcal{R}$.
La grande diffculté concerne la mise sous forme d'un PDM du problème que l'on veut résoudre.

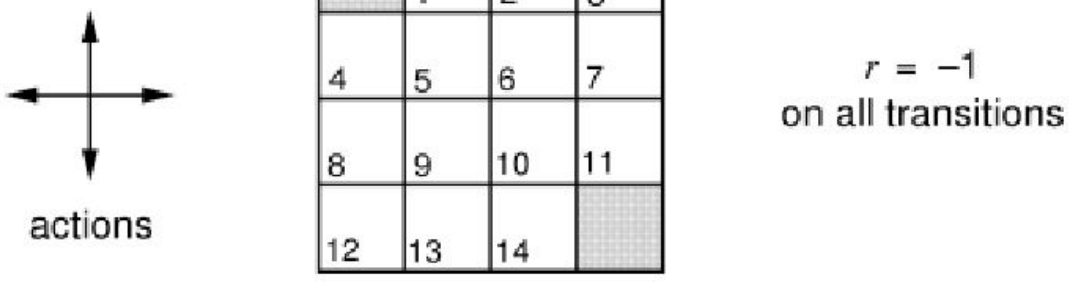
- la recherche directe de la politique optimale est en $O(|\mathcal{S}|^{|\mathcal{A}|})$ (c'est un problème NP-difficile) ;
- en pratique : on peut appliquer ces algorithmes à des PDM ayant jusque de l'ordre de 10^6 états ;
- la complexité en temps de calcul est $O(|\mathcal{A}| \cdot |\mathcal{S}|)$ pour l'algorithme d'itération de la valeur ;
- pour calculer la fonction valeur, les méthodes de programmation linéaire sont beaucoup moins efficaces que l'algorithme d'itération de la valeur ;

Implantation & difficultés de résolution

Les algorithmes que l'on vient de rencontrer sont très faciles à implanter mais reposent sur la connaissance de l'environnement $\mathcal{P}$ et $\mathcal{R}$.
La grande diffculté concerne la mise sous forme d'un PDM du problème que l'on veut résoudre.

- la recherche directe de la politique optimale est en $O(|\mathcal{S}|^{|\mathcal{A}|})$ (c'est un problème NP-difficile) ;
- en pratique : on peut appliquer ces algorithmes à des PDM ayant jusque de l'ordre de 10^6 états ;
- la complexité en temps de calcul est $O(|\mathcal{A}| \cdot |\mathcal{S}|)$ pour l'algorithme d'itération de la valeur ;
- pour calculer la fonction valeur, les méthodes de programmation linéaire sont beaucoup moins efficaces que l'algorithme d'itération de la valeur ;
- de nombreuses heuristiques sont possibles pour accélérer l'itération de la valeur. Idée : ne pas parcourir systématiquement tous les états à chaque balayage ; il faut cependant visiter tous les états suffisamment de fois.

Illustration : Grille 2D



4 actions. Tâche épisodique sans *discount*. $r_t = -1$ à chaque transition jusqu'aux deux états terminaux. Politique π équiprobable

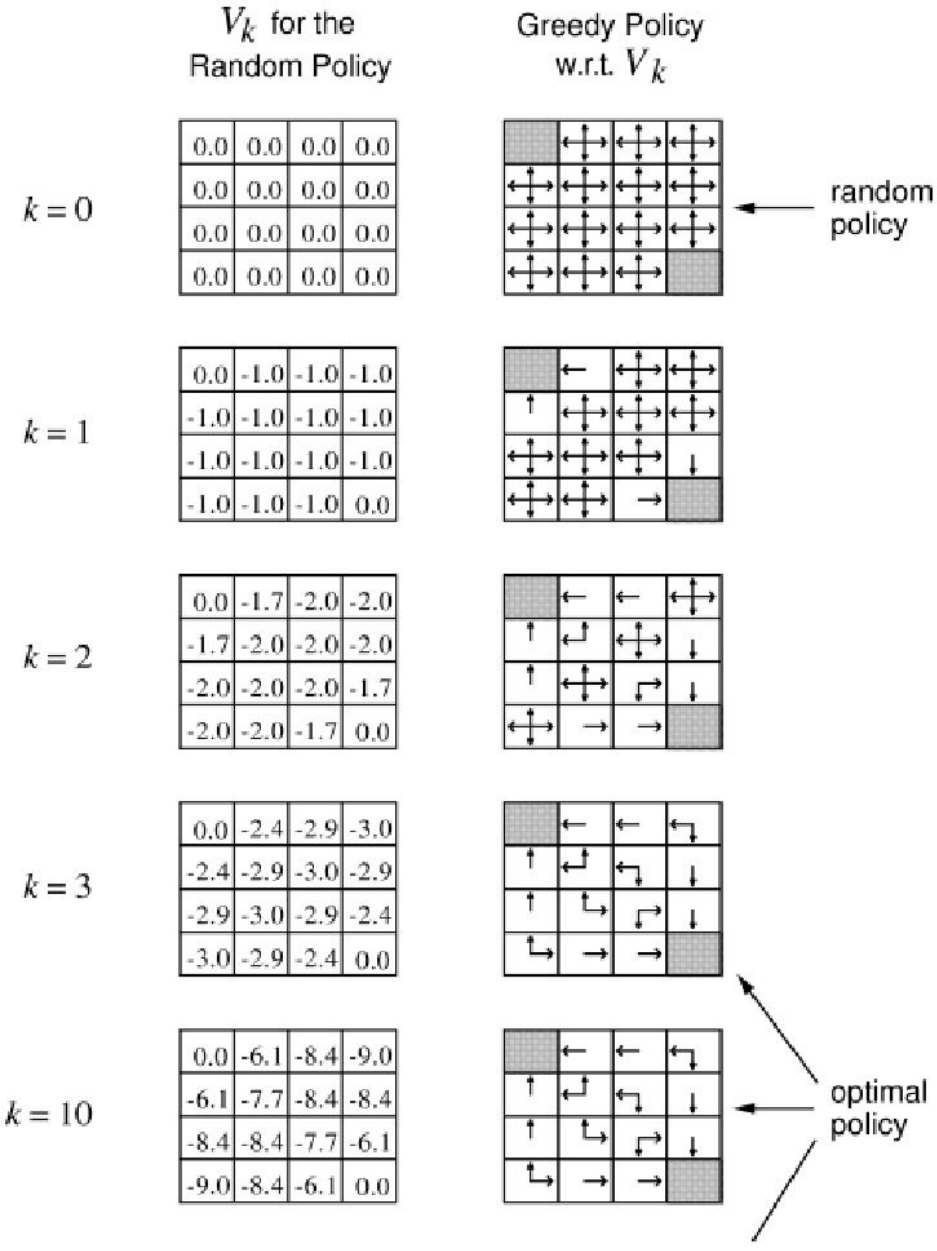

V_k for the Random Policy
Greedy Policy w.r.t. V_k
$k = 0$
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
random policy
$k = 1$
0.0 -1.0 -1.0 -1.0
-1.0 -1.0 -1.0 -1.0
-1.0 -1.0 -1.0 -1.0
-1.0 -1.0 -1.0 0.0
$k = 2$
0.0 -1.7 -2.0 -2.0
-1.7 -2.0 -2.0 -2.0
-2.0 -2.0 -2.0 -1.7
-2.0 -2.0 -1.7 0.0
$k = 3$
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
$k = 10$
0.0 -6.1 -8.4 -9.0
-6.1 -7.7 -8.4 -8.4
-8.4 -8.4 -7.7 -6.1
-9.0 -8.4 -6.1 0.0
optimal policy

Illustration : Jeu avec mise

- Un joueur mise sur le résultat d'une pièce trucquée. Il double sa mise s'il gagne et la perd sinon.

Que donne l'algorithme d'itération de la valeur ?

Illustration : Jeu avec mise

- Un joueur mise sur le résultat d'une pièce trucquée. Il double sa mise s'il gagne et la perd sinon.
- Le jeu prend fin dès qu'il a 100 euros ou s'il n'a plus de quoi miser.

Que donne l'algorithme d'itération de la valeur ?

Illustration : Jeu avec mise

- Un joueur mise sur le résultat d'une pièce trucquée. Il double sa mise s'il gagne et la perd sinon.
- Le jeu prend fin dès qu'il a 100 euros ou s'il n'a plus de quoi miser.
- $r_t = 0$ à chaque transition, exceptée lorsqu'il atteint son but auquel cas $r_t = 1$.

Que donne l'algorithme d'itération de la valeur ?

Illustration : Jeu avec mise

- Un joueur mise sur le résultat d'une pièce trucquée. Il double sa mise s'il gagne et la perd sinon.
- Le jeu prend fin dès qu'il a 100 euros ou s'il n'a plus de quoi miser.
- $r_t = 0$ à chaque transition, exceptée lorsqu'il atteint son but auquel cas $r_t = 1$.
- La fonction état-valeur retourne, par définition, la probabilité de gagner à partir de l'état courant.

Que donne l'algorithme d'itération de la valeur ?

Illustration : Jeu avec mise

- Un joueur mise sur le résultat d'une pièce trucquée. Il double sa mise s'il gagne et la perd sinon.
- Le jeu prend fin dès qu'il a 100 euros ou s'il n'a plus de quoi miser.
- $r_t = 0$ à chaque transition, exceptée lorsqu'il atteint son but auquel cas $r_t = 1$.
- La fonction état-valeur retourne, par définition, la probabilité de gagner à partir de l'état courant.
- La politique optimale maximise c'est probabilité.

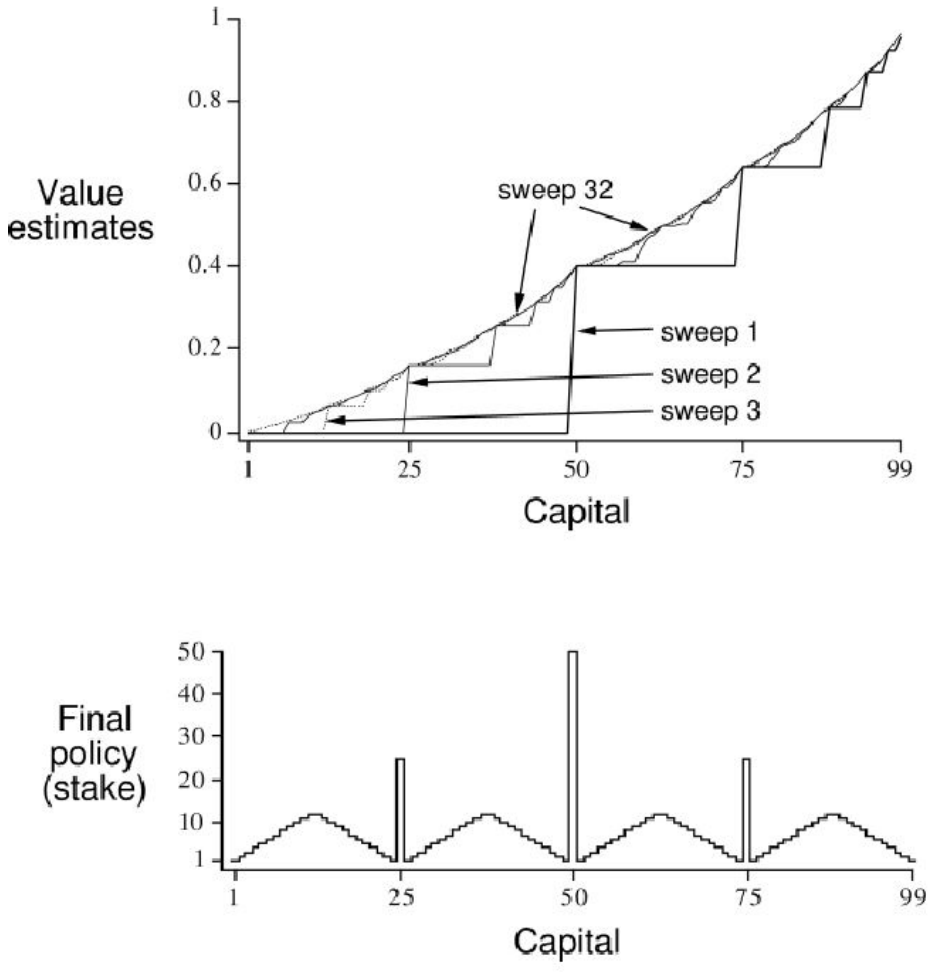
Que donne l'algorithme d'itération de la valeur ?

Illustration : Jeu avec mise

- Un joueur mise sur le résultat d'une pièce trucquée. Il double sa mise s'il gagne et la perd sinon.
- Le jeu prend fin dès qu'il a 100 euros ou s'il n'a plus de quoi miser.
- $r_t = 0$ à chaque transition, exceptée lorsqu'il atteint son but auquel cas $r_t = 1$.
- La fonction état-valeur retourne, par définition, la probabilité de gagner à partir de l'état courant.
- La politique optimale maximise c'est probabilité.
- Le joueur mise sur Pile, un événement qui survient avec $p = 0,4$.

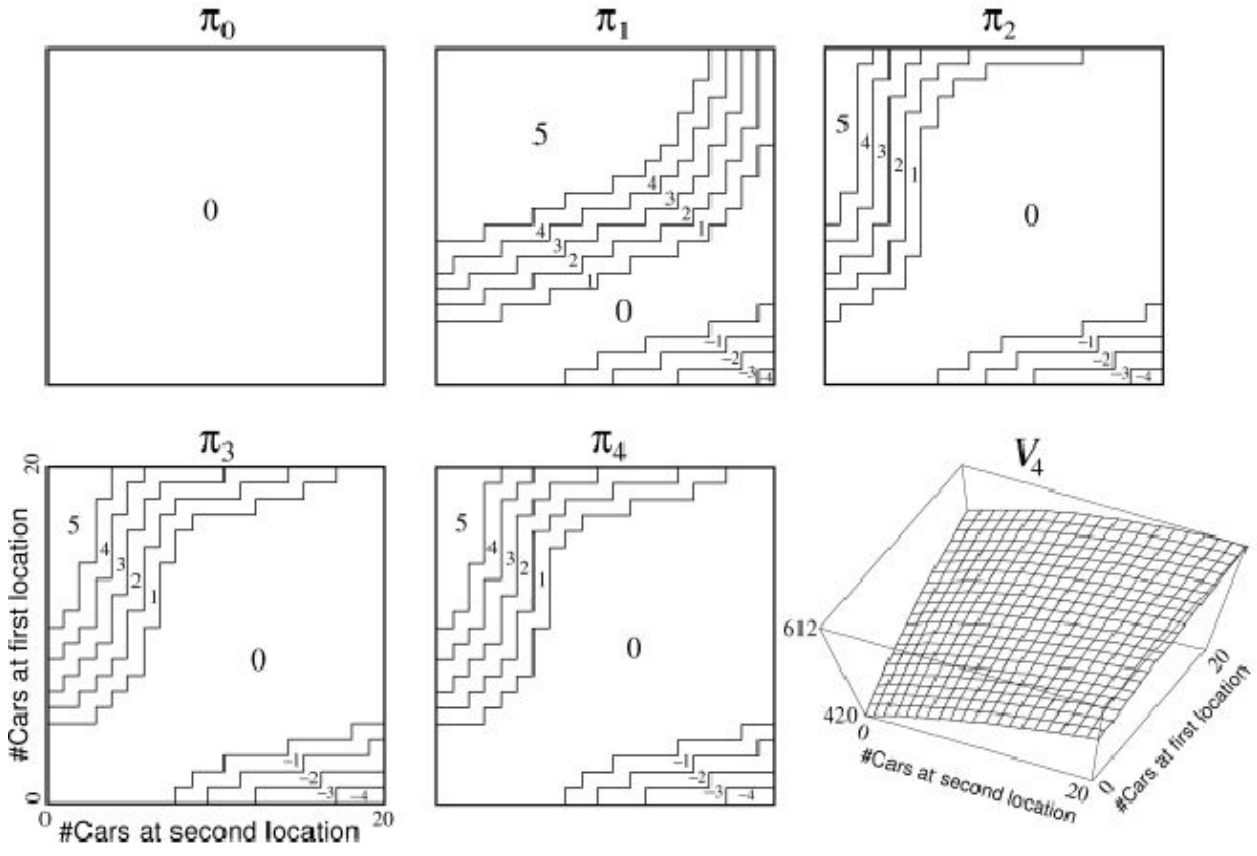
Que donne l'algorithme d'itération de la valeur ?

Illustration : Joueur



4 actions. Tâche épisodique sans *discount*. $r_t = -1$ à chaque transition jusqu'aux deux états terminaux. Politique π équiprobable

Illustration : Location voitures



Location de voiture facturée 10 euros/jour. Deux agences. Transport de voiture la nuit au coût de 2 euros. Requêtes de voitures selon des lois de Poisson.

Environnement inconnu

- On ne suppose plus la connaissance de l'environnement $\mathcal{P}$ et $\mathcal{R}$.

Environnement inconnu

- On ne suppose plus la connaissance de l'environnement $\mathcal{P}$ et $\mathcal{R}$.
- Ne disposant pas d'un modèle de la dynamique de l'environnement, l'agent va maintenant devoir inférer une politique optimale en interagissant avec son environnement.

Environnement inconnu

- On ne suppose plus la connaissance de l'environnement $\mathcal{P}$ et $\mathcal{R}$.
- Ne disposant pas d'un modèle de la dynamique de l'environnement, l'agent va maintenant devoir inférer une politique optimale en interagissant avec son environnement.
- l'apprentissage "en-ligne" alors que les algorithmes de programmation dynamique vus précédemment font de l'apprentissage "hors-ligne" (sans interagir avec leur environnement).

Méthode de Monte Carlo

- On ne s'intéresse ici qu'à des tâches épisodiques répétées.

Méthode de Monte Carlo

- On ne s'intéresse ici qu'à des tâches épisodiques répétées.
- A partir d'un grand nombre de passages par un état et d'un grand nombre de trajectoires, la moyenne des récompenses observés pour chaque état tend vers l'espérance (loi des grands nombres). Cela nous donne l'idée de l'algorithme de Monte-Carlo.

Méthode de Monte Carlo

- On ne s'intéresse ici qu'à des tâches épisodiques répétées.
- A partir d'un grand nombre de passages par un état et d'un grand nombre de trajectoires, la moyenne des récompenses observés pour chaque état tend vers l'espérance (loi des grands nombres). Cela nous donne l'idée de l'algorithme de Monte-Carlo.
- Il faut que tous les états soient visités un grand nombre de fois pour obtenir une approximation satisfaisante de V^{π}.

Algorithme 4 Un algorithme de la famille Monte Carlo.

Nécessite: une politique π

pour chaque état s **faire**

pour $j \in \{0, ..., M\}$ **faire**

générer une trajectoire en suivant la politique π à partir de l'état $s_0 = s$. On note $\{s_t, a_t, r_t\}_t$ la suite des triplets état, action, retour immédiat de cette trajectoire. On suppose la trajectoire de taille bornée T.

$v(s_0, j) \leftarrow \sum_{t=0}^{t=T} \gamma^t r_t$

fin pour

$V^\pi(s) \leftarrow \sum_{j=0}^{j=M} v(s, j)$

fin pour

Différence temporelle

$V^\pi(s)$ quantifie l'intéêt d'être dans l'état s.
En moyenne, on doit avoir :

$$V^\pi(s_t) = r_t + \gamma V^\pi(s_{t+1})$$

soit :

$$\underbrace{r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)}_{TD} = 0$$

Durant l'apprentissage, TD n'est pas nulle. Si l'estimation de $V^\pi(s)$ est trop optimiste en s, TD est négative ; TD est positive sinon. Aussi, TD peut être utilisée comme une correction **en ligne** à apporter à $V^\pi(s_t)$.

Algorithme 5 L'algorithme TD(0).

Nécessite: une politique π

$V^\pi \leftarrow 0$

pour ∞ **faire**

initialiser l'état initial s_0

$t \leftarrow 0$

répéter

émettre l'action $a_t = \pi(s_t)$

observer r_t et s_{t+1}

$V^\pi(s_t) \leftarrow V^\pi(s) + \alpha[r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)]$

$t \leftarrow t + 1$

jusque $s_t \in \mathcal{F}$

fin pour

Convergence

Propriété

Pour un PDM et une politique donnés, l'algorithme $TD(0)$ converge asymptotiquement vers V^π à condition que :

- chaque état soit visité une infinité de fois,

On peut choisir

$$\alpha_t(s) = \frac{1}{1 + \text{nombre de visites en } s}$$

Convergence

Propriété

Pour un PDM et une politique donnés, l'algorithme $TD(0)$ converge asymptotiquement vers V^π à condition que :

- chaque état soit visité une infinité de fois,
- on ait

$$\sum_t \alpha_t(s) = \infty \quad \text{et} \quad \sum_t \alpha_t^2(s) < \infty, \quad \forall s$$

On peut choisir

$$\alpha_t(s) = \frac{1}{1 + \text{nombre de visites en } s}$$

Q-Learning

L'algorithme $TD(0)$ évalue une politique. On peut chercher à l'améliorer et aboutir à une politique optimale via la technique d'amélioration de la politique déjà vue.
Cependant, on va adopter ici une autre stratégie qui fournira un algorithme plus efficace, l'algorithme **Q-Learning.**

Algorithme 6 L'algorithme *Q-Learning*.

$Q(s,a) \leftarrow 0, \forall (s,a) \in (\mathcal{S}, \mathcal{A})$

pour ∞ **faire**

initialiser l'état initial s_0

$t \leftarrow 0$

répéter

choisir l'action à émettre a_t et l'émettre

observer r_t et s_{t+1}

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

$t \leftarrow t + 1$

jusque $s_t \in \mathcal{F}$

fin pour

Q-Learning

Le choix de l'action doit garantir un équilibre entre l'exploration et l'exploitation de l'apprentissage déjà réalisé.
Intuitivement, l'exploration doit initialement être importante (quand l'apprentissage est encore très partiel, on explore) puis diminuer au profit de l'exploitation quand l'apprentissage a été effectué pendant une période suffisamment longue.
Plusieurs stratégies sont utilisées classiquement : gloutone, ϵ-gloutonne, softmax etc.

Propriété

Pour un PDM donné, l'algorithme *Q-Learning* converge asymptotiquement vers $V^\star$ à condition que :

- chaque état soit visité une infinité de fois,

Q-Learning

Le choix de l'action doit garantir un équilibre entre l'exploration et l'exploitation de l'apprentissage déjà réalisé.
Intuitivement, l'exploration doit initialement être importante (quand l'apprentissage est encore très partiel, on explore) puis diminuer au profit de l'exploitation quand l'apprentissage a été effectué pendant une période suffisamment longue.
Plusieurs stratégies sont utilisées classiquement : gloutone, ϵ-gloutonne, softmax etc.

Propriété

Pour un PDM donné, l'algorithme *Q-Learning* converge asymptotiquement vers $V^\star$ à condition que :

- chaque état soit visité une infinité de fois,
- on ait

$$\sum_t \alpha_t(s) = \infty \quad \text{et} \quad \sum_t \alpha_t^2(s) < \infty, \quad \forall s$$

Algorithme 7 L'algorithme SARSA.

$Q(s,a) \leftarrow 0, \forall (s,a) \in (\mathcal{S}, \mathcal{A})$
pour ∞ **faire**
  initialiser l'état initial s_0
  $t \leftarrow 0$
  choisir l'action à émettre a_t en fonction de la politique dérivée de Q (ϵ-gloutonne par exemple) et l'émettre
  répéter
    émettre a_t
    observer r_t et s_{t+1}
    choisir l'action à émettre a_{t+1} en fonction de la politique dérivée de Q (ϵ-gloutonne par exemple)
    $Q(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
    $t \leftarrow t + 1$
  jusque $s_t \in \mathcal{F}$
fin pour

SARSA

L'analyse de la convergence de SARSA est plus difficile que celle du *Q-Learning*.

Propriété

SARSA converge vers la politique optimale si :

- toutes les paires (état, action) sont visitées une infinité de fois ;
- la stratégie de choix de l'action tend vers une stratégie gloutonne.

En comparant SARSA et *Q-Learning*, on voit que SARSA un algorithme "*on-policy*" à la différence de *Q-Learning* qui choisit la meilleure action (celle prédite par Q) dans l'état suivant : c'est un algorithme "*off-policy*".

Traces d'éligibilités

- Lorsqu'une récompense est obtenue, elle due à la dernière transition, mais aussi, dans une moindre mesure aux précédentes.
- Les algorithmes $TD(0)$ et *Q-Learning* ne modifient que la fonction valeur de du dernier état visité à t à partir de la récompense immédiate à $t+1$.
- On peut propager la récompense à t aux transitions précédentes le long de la trajectoire.
- C'est l'idée des algorithmes $TD(\lambda)$ et $Q(\lambda)$ qui ne sont qu'une généralisation des $TD(0)$ et *Q-Learning.*

Algorithme 8 L'algorithme TD (λ).

Nécessite: une politique π

$V^\pi \leftarrow 0$ (valeur arbitraire)

pour ∞ **faire**

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

initialiser l'état initial s_0

$t \leftarrow 0$

répéter

émettre l'action $a_t = \pi(s_t)$

observer r_t et s_{t+1}

$\delta \leftarrow r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$

$e(s_t) \leftarrow e(s_t) + 1$

pour $s \in \mathcal{S}$ **faire**

$V^\pi(s) \leftarrow V^\pi(s) + \alpha\delta e(s)$

$e(s) \leftarrow \gamma\lambda e(s)$

fin pour

$t \leftarrow t + 1$

jusque $s_t \in \mathcal{F}$

fin pour

Algorithme 10 L'algorithme Q (λ).

$Q^\pi \leftarrow 0$ (valeur arbitraire)
pour ∞ **faire**
  $e(s,a) \leftarrow 0, \forall (s,a) \in \mathcal{S} \times \mathcal{A}$
  $t \leftarrow 0$
  initialiser l'état initial s_0
  choisir l'action à émettre a_0
  répéter
    émettre l'action a_t
    observer r_t et s_{t+1}
    choisir l'action qui sera émise dans s_{t+1}
    $a^* \leftarrow \arg\max_{a \in \mathcal{A}(s_{t+1})} Q(s_{t+1}, a)$
    $\delta \leftarrow r_t + \gamma Q(s_{t+1}, a^*) - Q(s_t, a_t)$
    $e(s_t, a_t) \leftarrow e(s_t, a_t) + 1$
    pour $(s,a) \in \mathcal{S} \times \mathcal{A}$ **faire**
      $Q(s,a) \leftarrow Q(s,a) + \alpha \delta e(s,a)$
      mise à jour de l'éligibilité de (s,a)
    fin pour
    $t \leftarrow t + 1$
  jusque $s_t \in \mathcal{F}$
fin pour

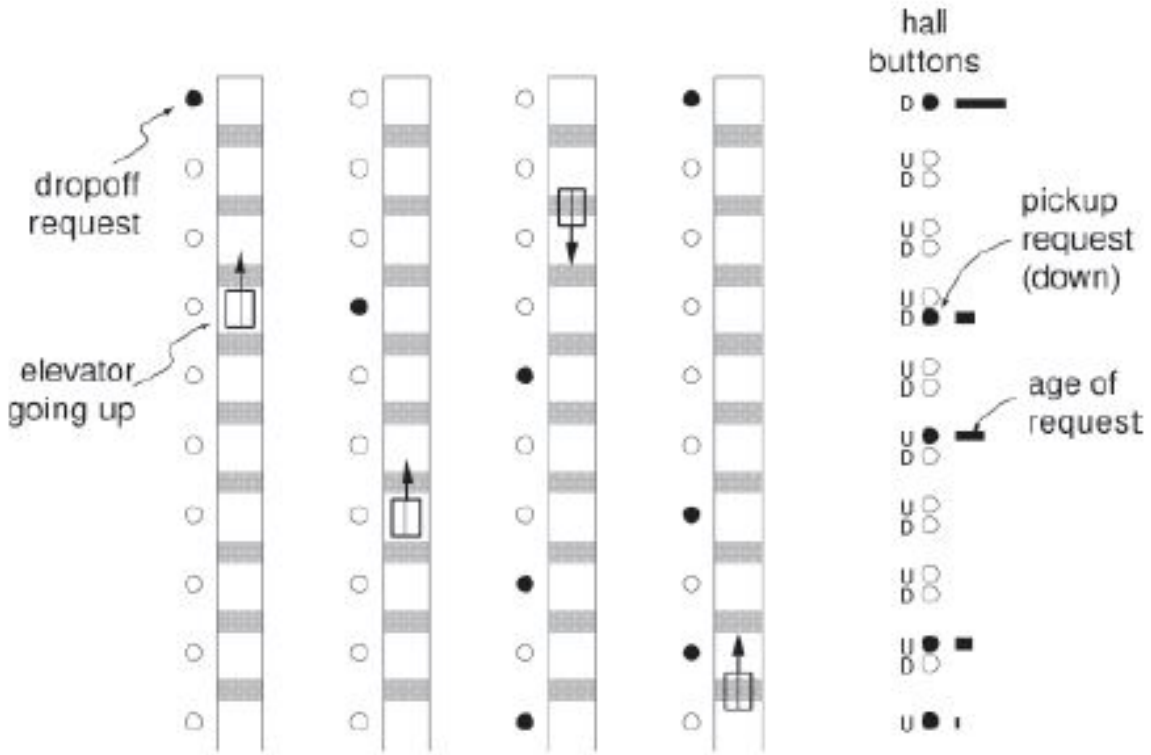
Grands espaces d'états

- Jusqu'à présent, une représentation tabulaire permettait de représenter les fonctions valeur et qualité. Chaque état devait être visité pour mettre à jour sa valeur sans impact direct sur la valeur d'autres états.
- Lorsque $|\mathcal{S}| \to \infty$, le balayage de $\mathcal{S}$ n'est plus possible.
- **Solution** : inférer une fonction paramétrique plus compacte, $\hat{V}(s) = f(\theta, s)$, où θ est un paramètre du modèle d'apprentissage et $f()$ la fonction implémentée par ce modèle, ayant des propriétés de généralisation capable d'approximer n'importe quelle fonction réelle, continue et bornée (e.g, réseau de neurones, SVM, arbre de décision).

L'objectif est de trouver $\theta^\star$ qui minimise $\|f(\theta^\star, s) - V^\star(s)\|$, $\forall s$.

Attention : en apprentissage supervisé, les exemples sont i.i.d. et on fournit la valeur idéale ; en apprentissage par renforcement, on fournit seulement une différence temporelle, donc une estimation grossière de la valeur idéale, et l'algorithme doit sélectionner les actions, donc les exemples.

Illustration : Ascenceur



4 ascenceurs et 10 étages. Position, vitesse, direction, bouttons allumés (oui/non) à chaque étage et durée durant laquelle ils sont allumés. Comment répartir les ascenceurs de façon à minimiser les temps d'attente ? Après discrétisation : 10^{22} états, soit 1000 ans de résolution pour une itération de l'algorithme d'itération de la valeur. r_t est égale à la moins la somme quadratique des temps d'attente des usagers en attente.

Lectures

- R.S. Sutton & A.G. Barto. Reinforcement Learning : An introduction. MIT Press, Cambridge, MA, 1998 A Bradford Book.
- Ph. Preux. Apprentissage par renforcement, notes de cours. Univ. Lille 3. 1998. www.grappa.univ-lille3.fr/ ppreux/Documents/ndc-ar.pdf
- A. Cornuéjols, L. Miclet, Y. Kodratoff, Apprentissage Artificiel : Concepts et algorithmes. Eyrolles, 2002
- C.M. Bishop, Pattern Recognition And Machine Learning, Springer, 2006