Apprentissage par renforcement

Olivier Sigaud

ISIR

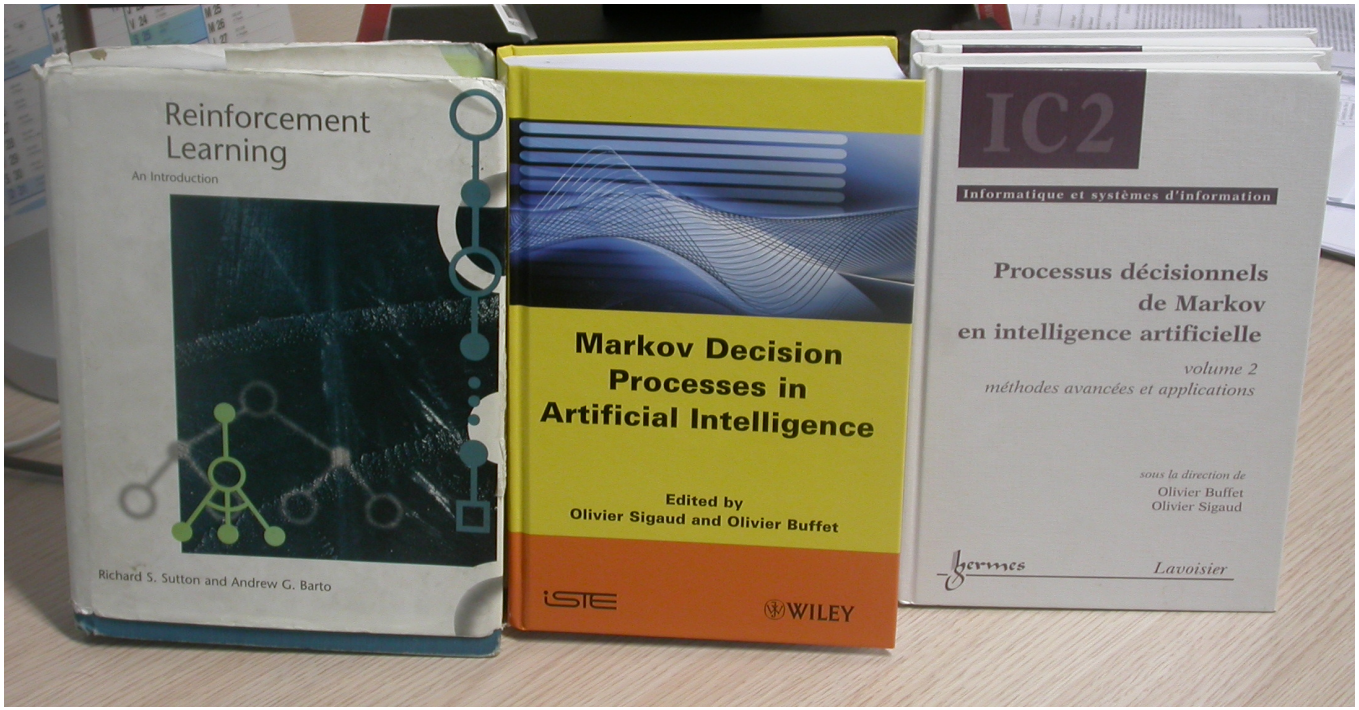
21 septembre 2012

Plan

Objectifs du tutoriel

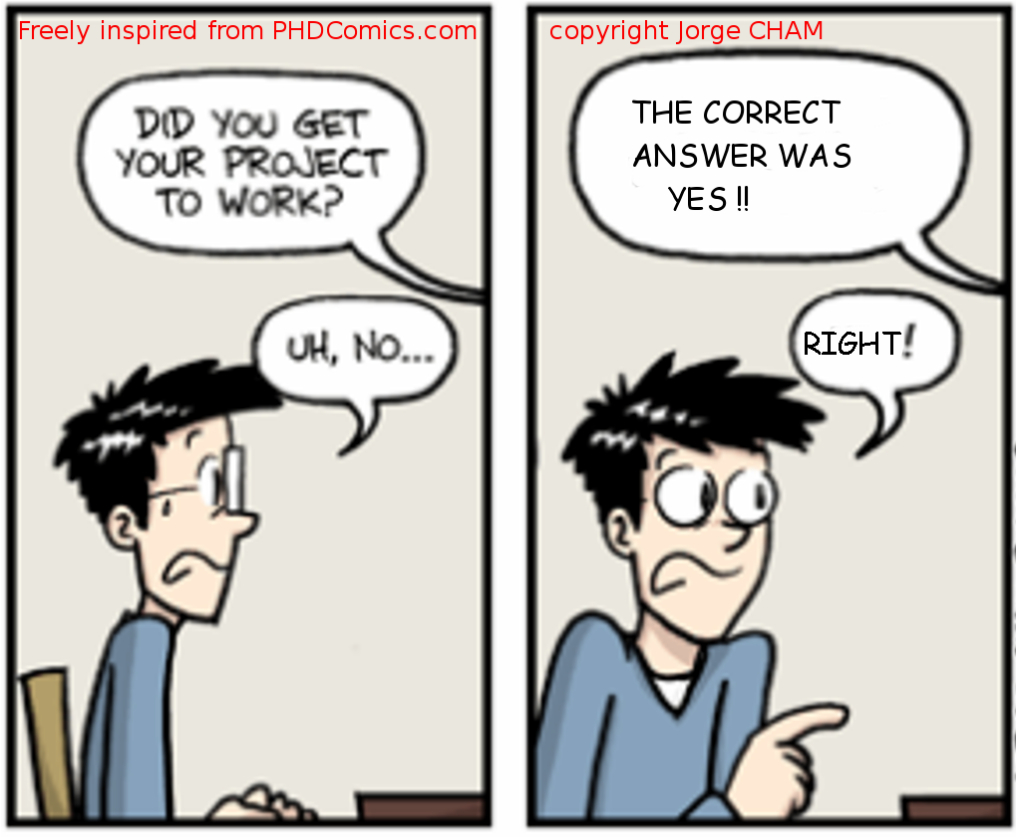
- Présenter les bases de :
  - Programmation dynamique
  - Apprentissage par renforcement direct avec Q-fonctions
  - Apprentissage par renforcement indirect
- avec un éclairage sur les méthodes acteur-critique
- Dans le cadre de cet exposé, on considère des états et actions discrètes

Livres introductifs



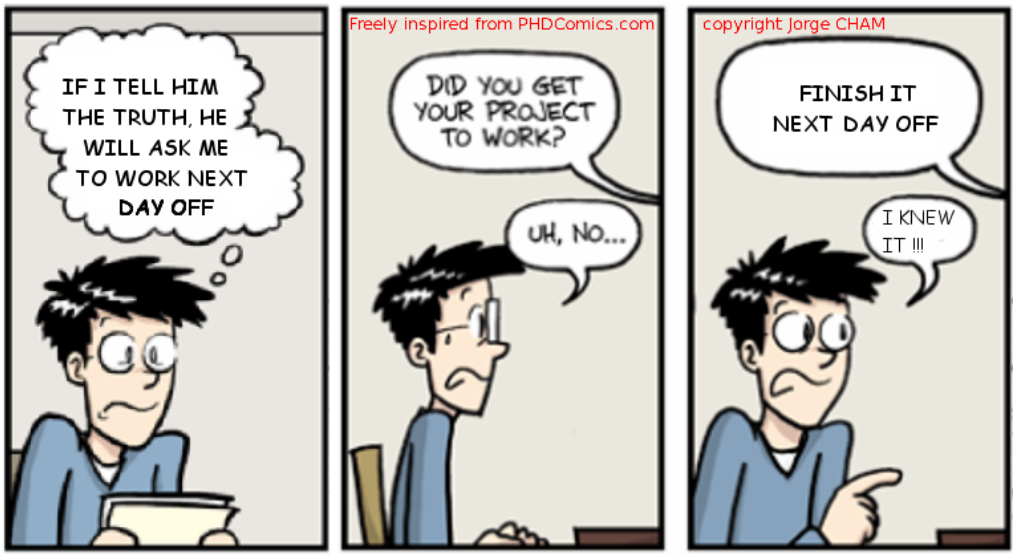
1. [Sutton & Barto, 1998] : la bible du domaine
2. [Buffet & Sigaud, 2008] : en français
3. [Sigaud & Buffet, 2010] : traduction (améliorée) de 2

Apprentissage supervisé



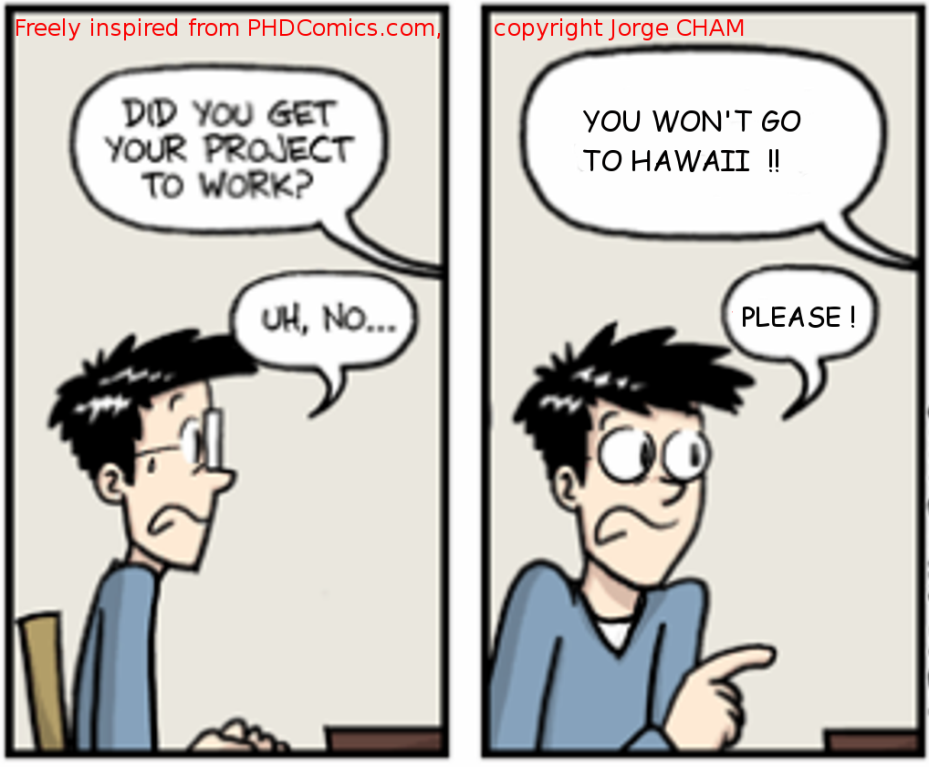
- Le superviseur indique à l'agent la réponse qu'il fallait donner
- Mécanisme typique : rétro-propagation du gradient, RLS
- Applications : classification, régression, approx. de fonctions

Apprentissage auto-supervisé



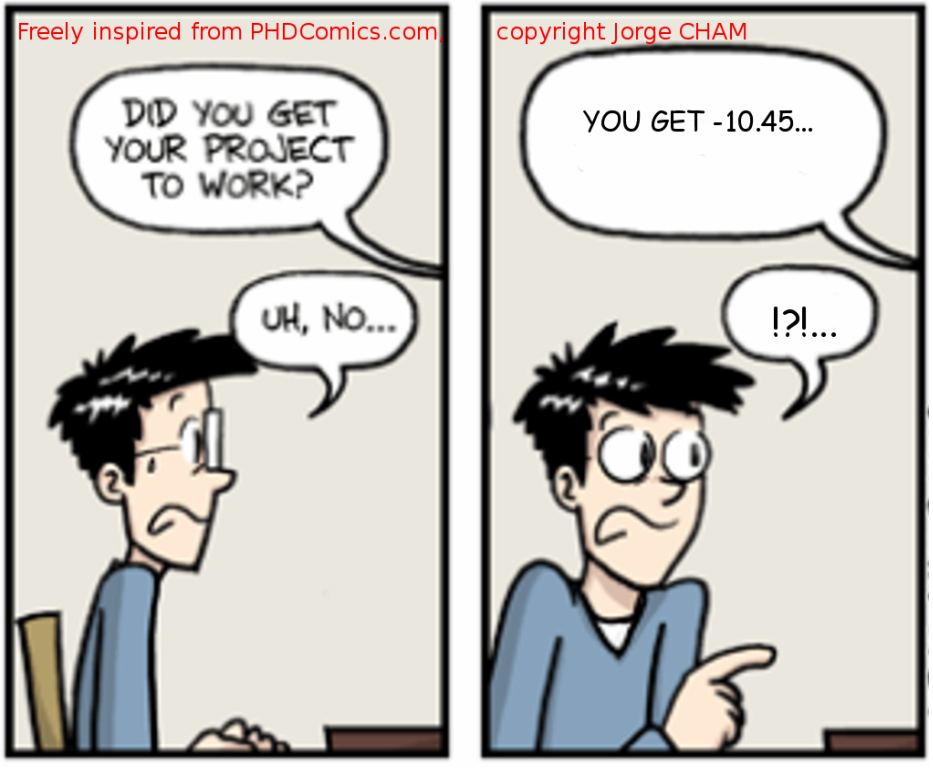
- L'agent prédit l'état suivant
- Correction donnée par l'environnement → Pas besoin de superviseur
- Application : apprentissage de prédictions

Apprentissage par renforcement



- L'environnement indique la valeur de l'action (récompense ou punition)
- Application : optimisation du comportement

Apprentissage par renforcement : modèle



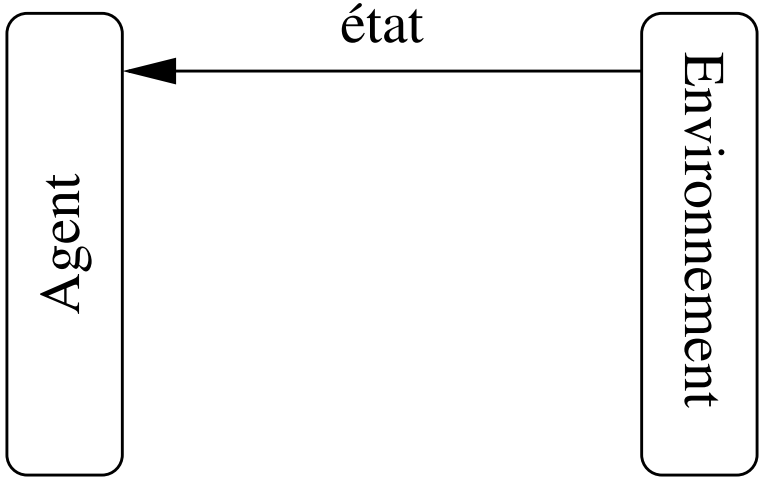
- En pratique, la valeur est un scalaire
- -10.45, c'est bien ou pas ?
- Il faut explorer pour le savoir

Compromis exploration/exploitation

- L'exploration peut être (très) dangereuse
- Plutôt exploiter ce que je sais ou explorer ?
- Suis-je optimal ? Faut-il encore explorer ?
- Explorer de moins en moins au cours du temps
- ϵ-*greedy* : choisir la meilleure action la plupart du temps, une action au hasard sinon

Processus Décisionnel de Markov

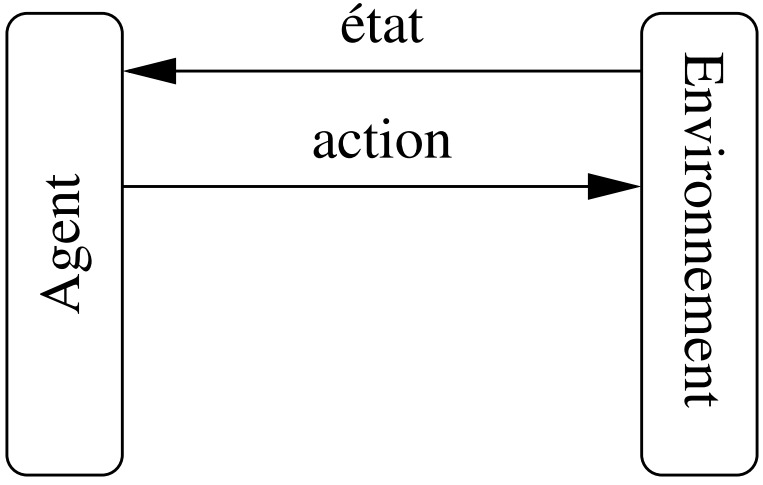
- S : espace d'états
- A : espace d'actions
- $T : S \times A \rightarrow \Pi(S)$: fonction de transition (dynamique du système)
- $r : S \times A \rightarrow \mathbb{R}$: fonction de renforcement



- Le formalisme des PDM définit s^{t+1} et r^{t+1} en $f(s_t, a_t)$
- Repose sur l'hypothèse de Markov (cf. Alain Dutech)
- Un PDM décrit un problème, pas une politique

Processus Décisionnel de Markov

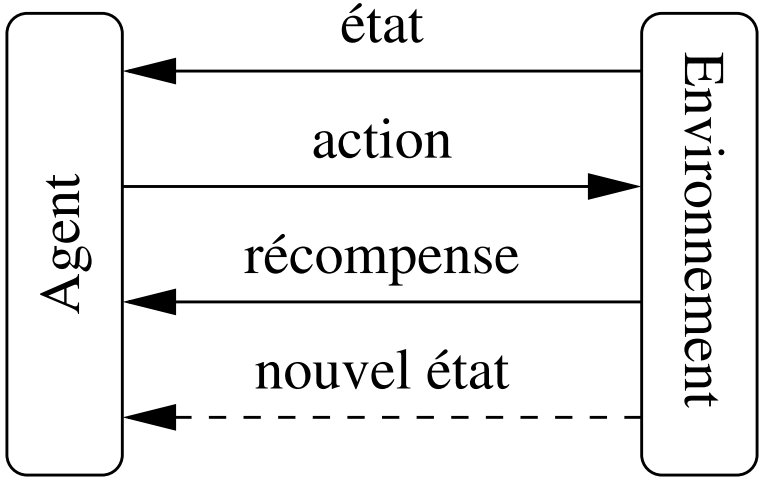
- S : espace d'états
- A : espace d'actions
- $T : S \times A \rightarrow \Pi(S)$: fonction de transition (dynamique du système)
- $r : S \times A \rightarrow \mathbb{R}$: fonction de renforcement



- Le formalisme des PDM définit s^{t+1} et r^{t+1} en $f(s_t, a_t)$
- Repose sur l'hypothèse de Markov (cf. Alain Dutech)
- Un PDM décrit un problème, pas une politique

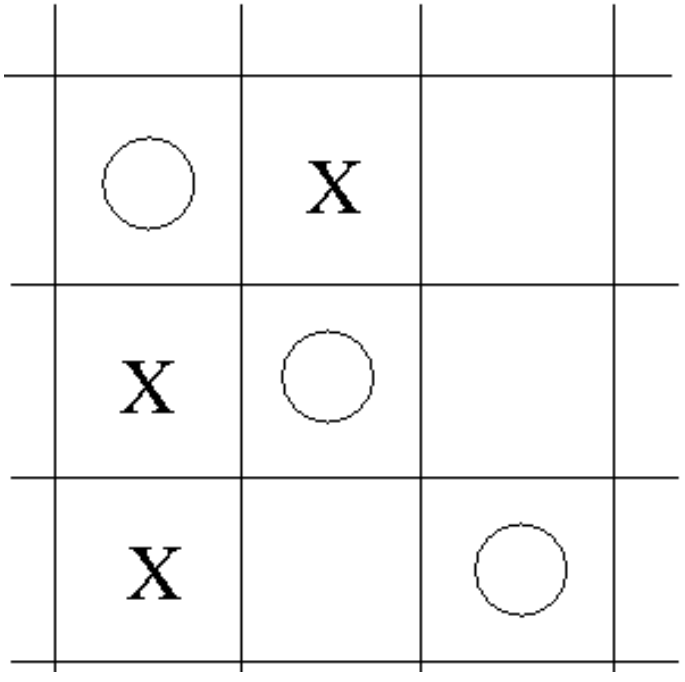
Processus Décisionnel de Markov

- S : espace d'états
- A : espace d'actions
- $T : S \times A \rightarrow \Pi(S)$: fonction de transition (dynamique du système)
- $r : S \times A \rightarrow \mathbb{R}$: fonction de renforcement



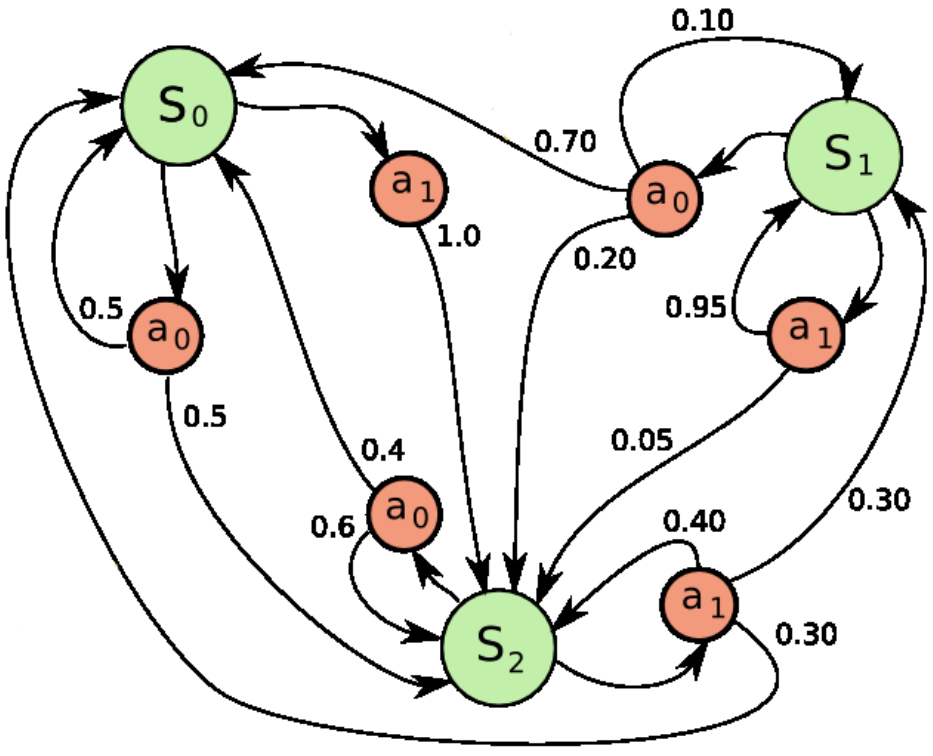
- Le formalisme des PDM définit s^{t+1} et r^{t+1} en $f(s_t, a_t)$
- Repose sur l'hypothèse de Markov (cf. Alain Dutech)
- Un PDM décrit un problème, pas une politique

Exemple : tic-tac-toe



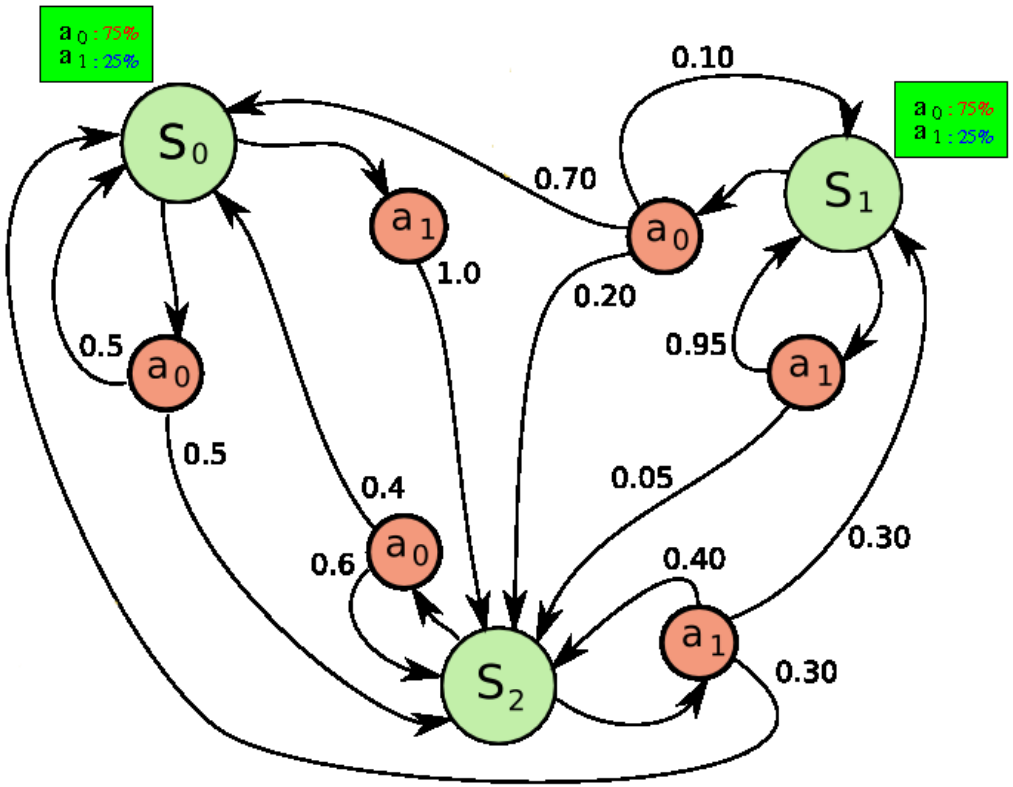
- L'état ne correspond pas toujours à une position
- Le coup de l'adversaire peut être caché dans la fonction de transition (stochastique)

Problème stochastique



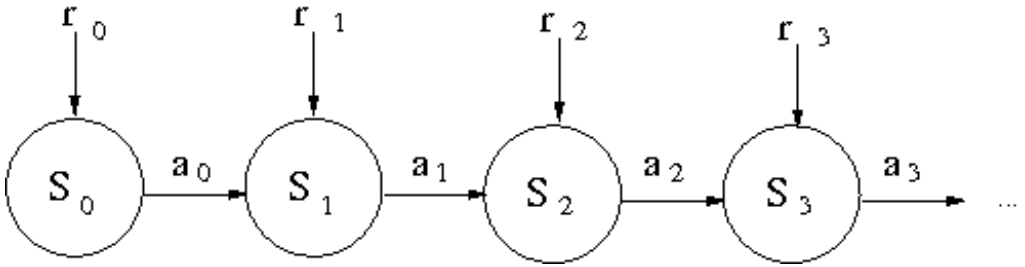
- Cas déterministe = cas particulier du stochastique
- $T(s^t, a^t, s^{t+1}) = p(s'|s, a)$

Politique stochastique

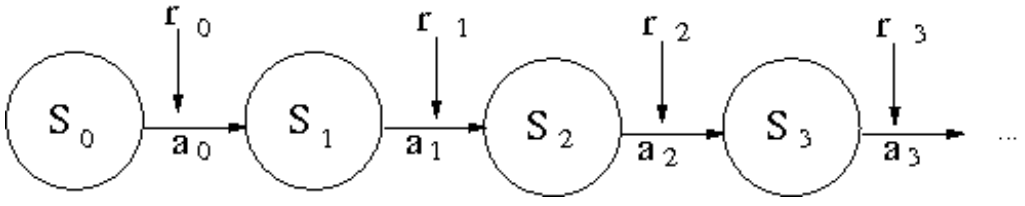


- Pour tout PDM, il existe une politique déterministe optimale
- Ce n'est pas vrai en multi-critère, par exemple

Chaîne de Markov et récompense : sur état ou action ?

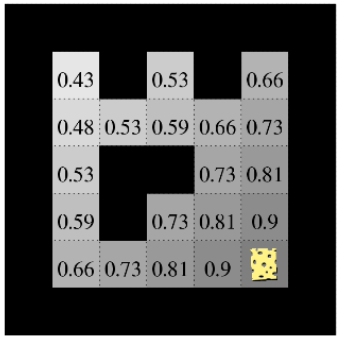


- Récompense sur les états



- Récompense sur les actions
- Dans la suite, on suppose des récompenses sur les actions (notées $r(s, a)$)

Fonction de valeur

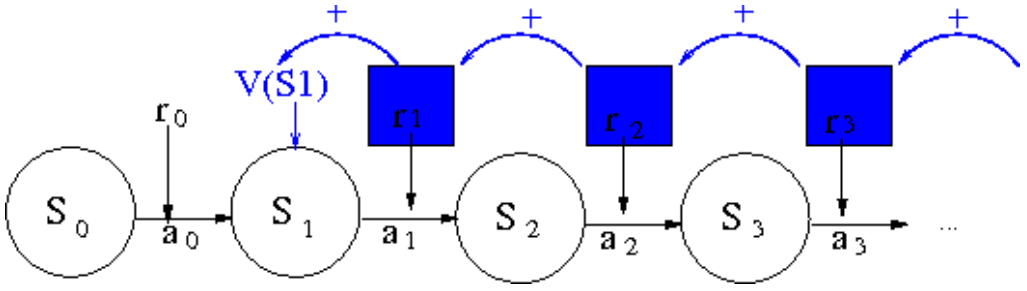


state / action	a_0	a_1	a_2	a_3
e_0	0.66	0.88	0.81	0.73
e_1	0.73	0.63	0.9	0.43
e_2	0.73	0.9	0.95	0.73
e_3	0.81	0.9	1.0	0.81
e_4	0.81	1.0	0.81	0.9
e_5	0.9	1.0	0.0	0.9

- But de la planification (et de l'apprentissage) : trouver une politique (stratégie) $\pi : S \rightarrow A$ qui maximise le cumul des utilités au long terme
- La fonction de valeur $V^\pi : S \rightarrow \mathbb{R}$ représente à partir de chaque état le cumul des utilités futures en suivant π. C'est un vecteur avec une valeur par état.
- La fonction de valeur d'action $Q^\pi : S \times A \rightarrow \mathbb{R}$ représente le cumul des utilités au long terme de faire chaque action en chaque état (puis en suivant π). C'est une matrice avec une valeur par état et par action.
- Exposé centré sur la fonction V, tout peut être étendu à la fonction Q

Critères d'agrégation

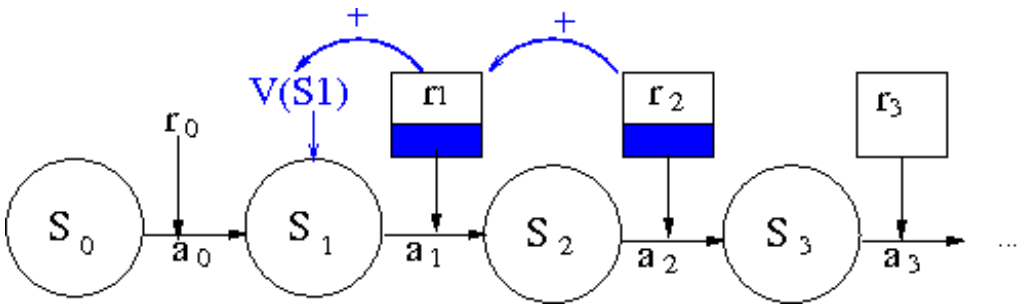
- Cumul des utilités au long terme : suppose de définir un critère agrégeant les utilités immédiates r.



- Simple somme (horizon fini) : $V^{\pi}(S_0) = r_0 + r_1 + r_2 + \ldots + r_N$
- Equivalent : critère moyen

Critères d'agrégation

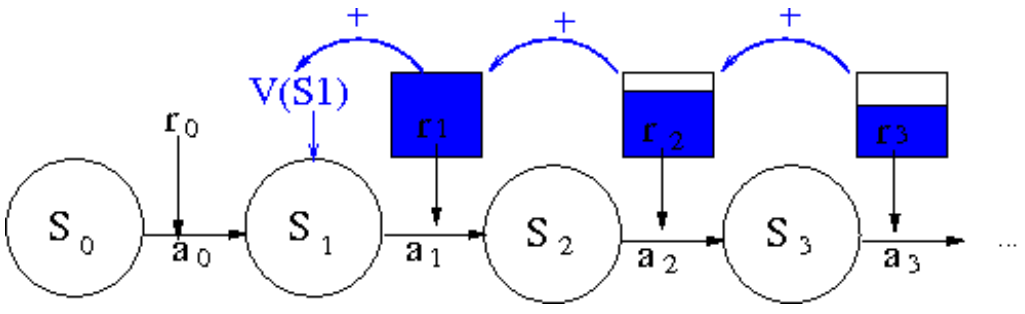
- Cumul des utilités au long terme : suppose de définir un critère agrégeant les utilités immédiates r.



- Critère moyen (ou somme) sur une fenêtre : $V^{\pi}(S_0) = \frac{r_0+r_1+r_2}{3}$...

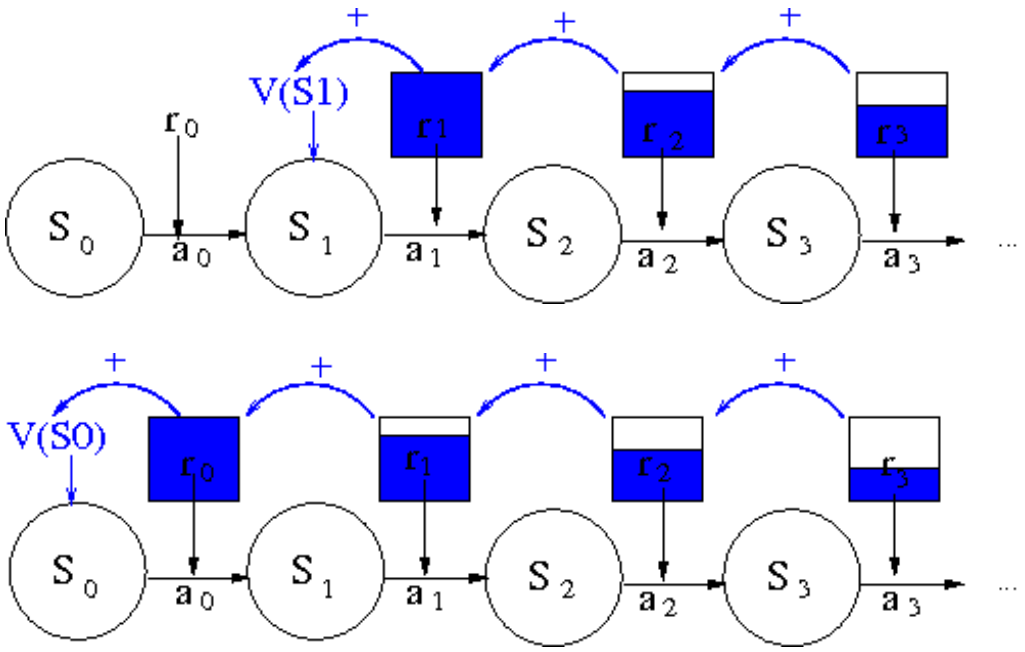
Critères d'agrégation

- Cumul des utilités au long terme : suppose de définir un critère agrégeant les utilités immédiates r.



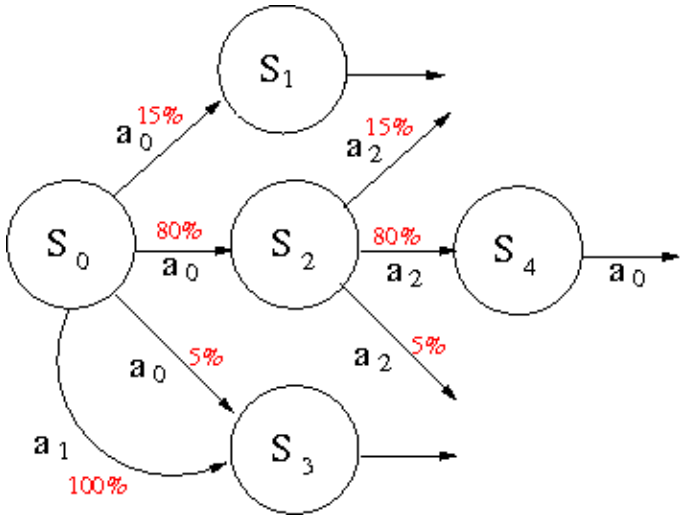
- Critère actualisé (γ-pondéré) : $V^{\pi}(s_{t_0}) = \sum_{t=t_0}^{\infty} \gamma^t r(s_t, \pi(s_t))$
- $\gamma \in [0, 1]$: facteur d'actualisation
  - si $\gamma = 0$, seule l'utilité immédiate est importante
  - si $\gamma = 1$, les utilités futures sont aussi importantes que l'utilité immédiate
- On continue sur le cas actualisé

Equation de Bellman sur une chaîne de Markov : récursion



- Etant donné le critère actualisé (γ-pondéré)
- $V(s_0) = r_0 + \gamma V(s_1)$

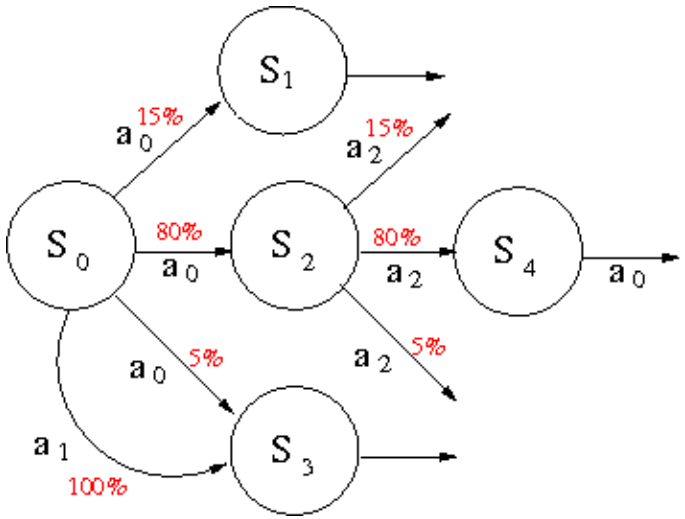
Equation de Bellman : cas général



- Généralisation de la récursion $V(s_0) = r_0 + \gamma V(s_1)$ au cas à plusieurs successeurs possibles
- cas π déterministe :

$$V^{\pi}(s) = r(s, \pi(s)) + \gamma \sum_{s'} p(s'|s, \pi(s)) V^{\pi}(s')$$

Equation de Bellman : cas général



- Généralisation de la récursion $V(s_0) = r_0 + \gamma V(s_1)$ au cas à plusieurs successeurs possibles
- cas π stochastique :

$$V^{\pi}(s) = \sum_{a} \pi(s,a)[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V^{\pi}(s')]$$

- La politique optimale se déduit de la fonction de valeur optimale :

$$\pi^*(s) = \arg\max_{a \in A} \Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V^*(s')]$$

Opérateur de Bellman et Programmation Dynamique

- On a $V^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s'} p(s'|s, \pi(s)) V^\pi(s')$
- On appelle opérateur de Bellman (noté T^π) l'application

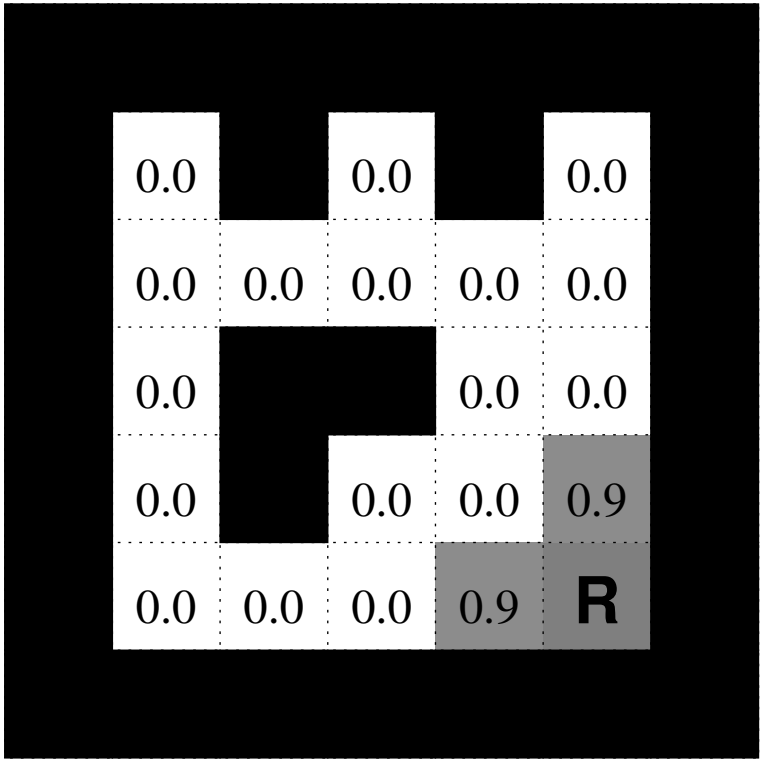
$$V^\pi(s) \leftarrow r(s, \pi(s)) + \gamma \sum_{s'} p(s'|s, \pi(s)) V^\pi(s')$$

- On appelle opérateur d'optimalité de Bellman (noté T^*) l'application

$$V(s) \leftarrow \max_{a \in A} \left[r(s, a) + \gamma \sum_{s'} p(s'|s, a) V(s') \right]$$

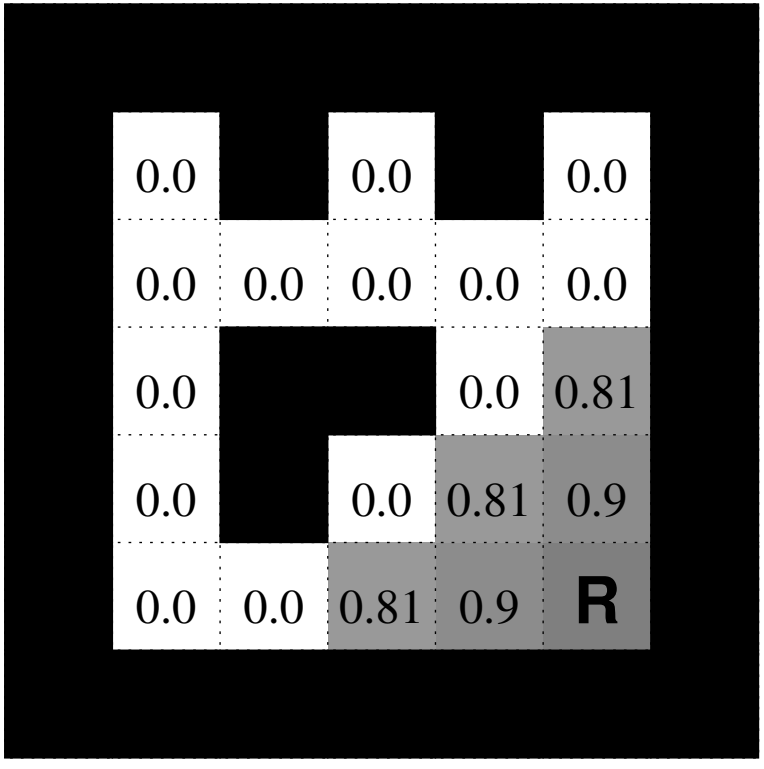
- La fonction de valeur optimale est un point fixe de l'opérateur d'optimalité de Bellman T^* : $V^* = T^* V^*$
- *Value Iteration* : trouver V^* en itérant $V_{i+1} \leftarrow T^* V_i$
- *Policy Iteration* : *policy evaluation* (avec $V^\pi_{i+1} \leftarrow T^\pi V^\pi_i$) + *policy improvement* avec
  $\forall s \in S, \pi'(s) \leftarrow \arg\max_{a \in A} \sum_{s'} p(s'|s, a)[r(s, a) + \gamma V^\pi(s')]$

Value Iteration en pratique



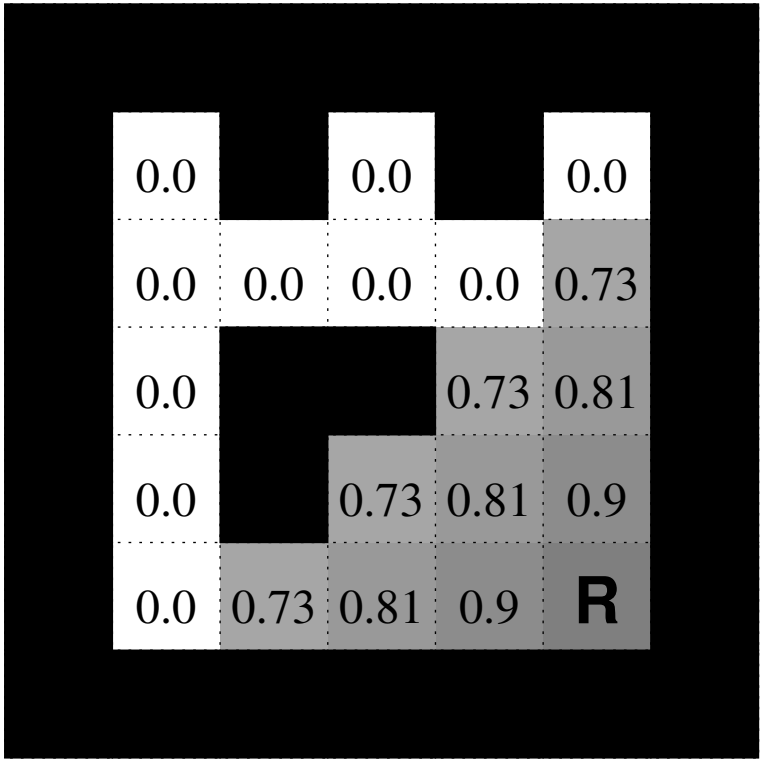
$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \right]$$

Value Iteration en pratique



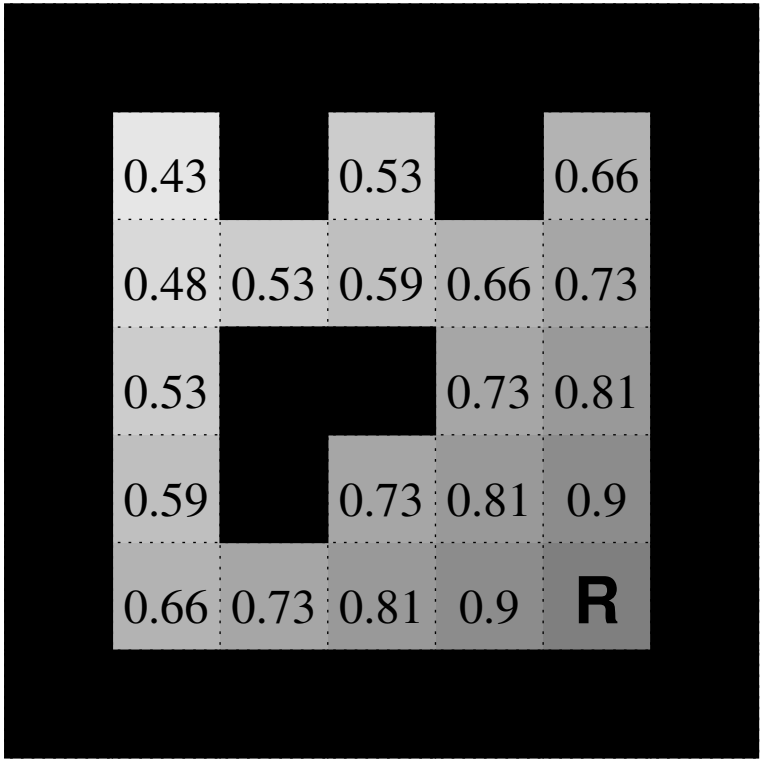
$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \right]$$

Value Iteration en pratique



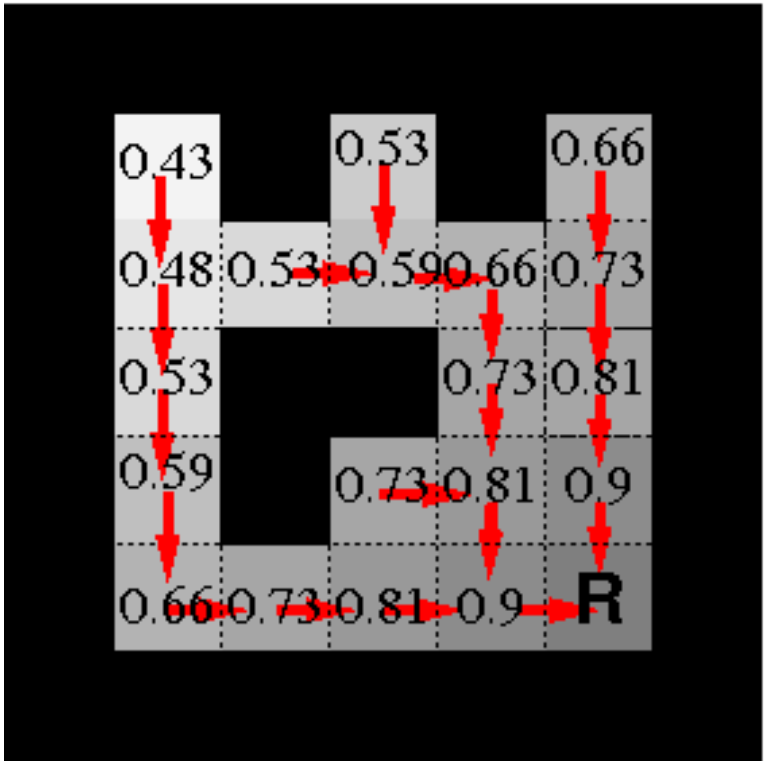
$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \right]$$

Value Iteration en pratique



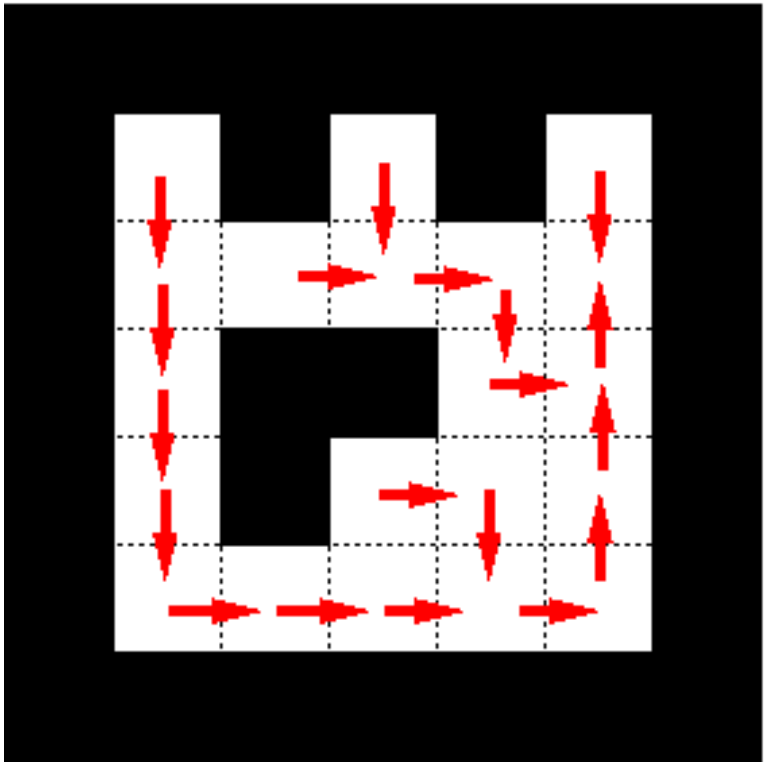
$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \right]$$

Value Iteration en pratique



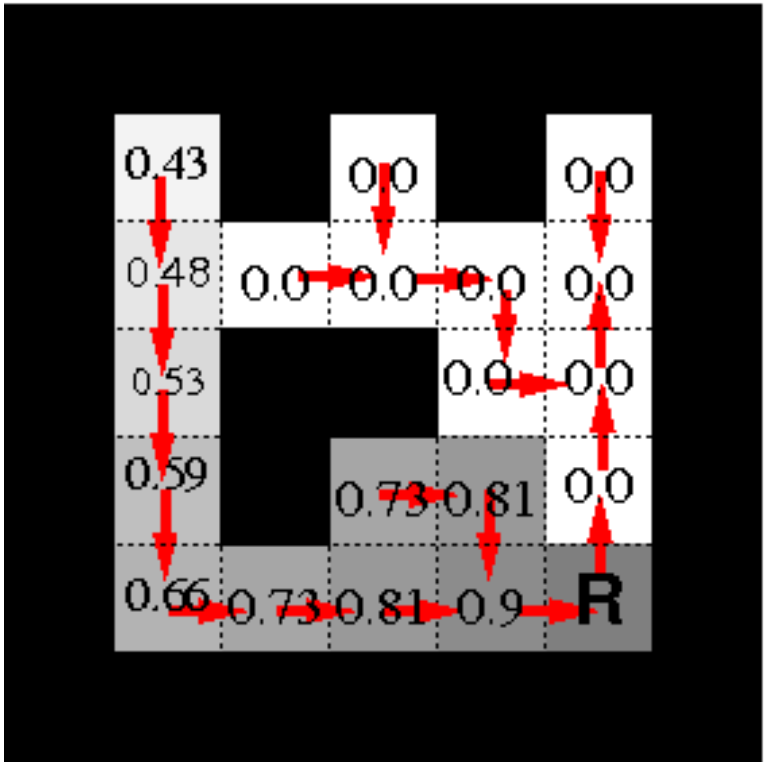
$$\pi^*(s) = \arg\max_{a \in A} \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V^*(s') \right]$$

Policy Iteration en pratique



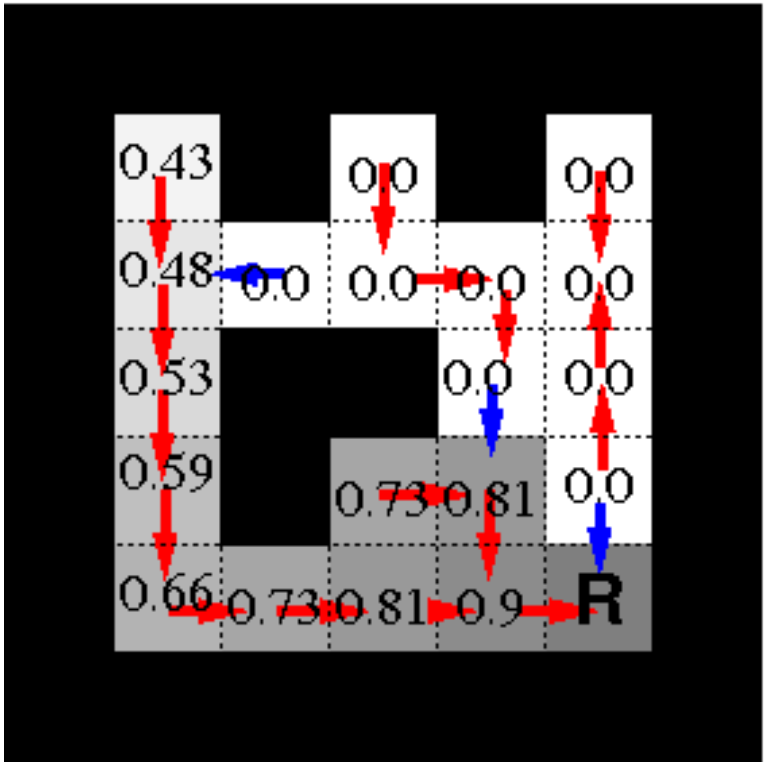
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

Policy Iteration en pratique



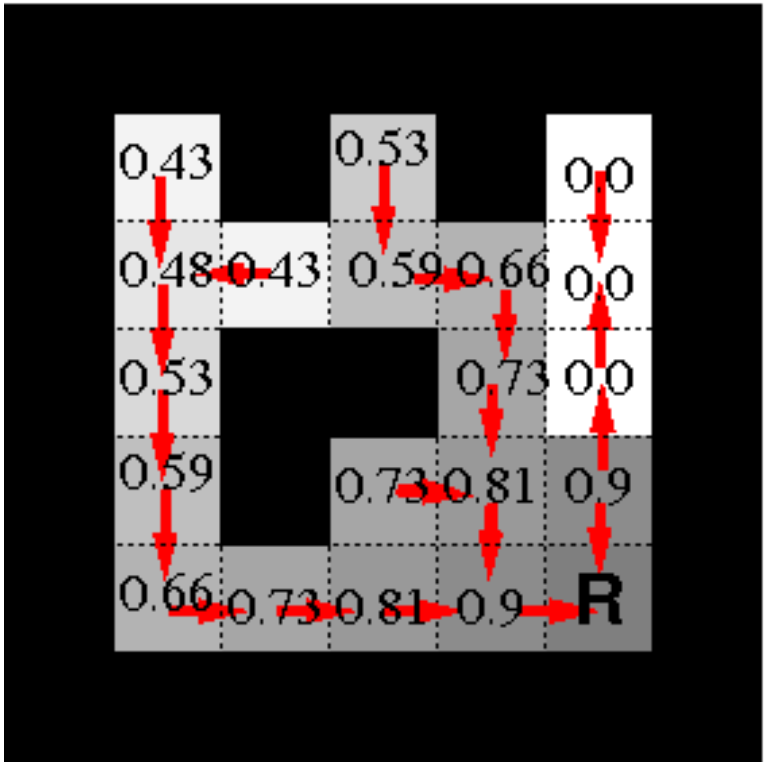
$$\forall s \in S, \pi_{i+1}(s) \leftarrow improve(\pi_i(s), V_i(s))$$

Policy Iteration en pratique



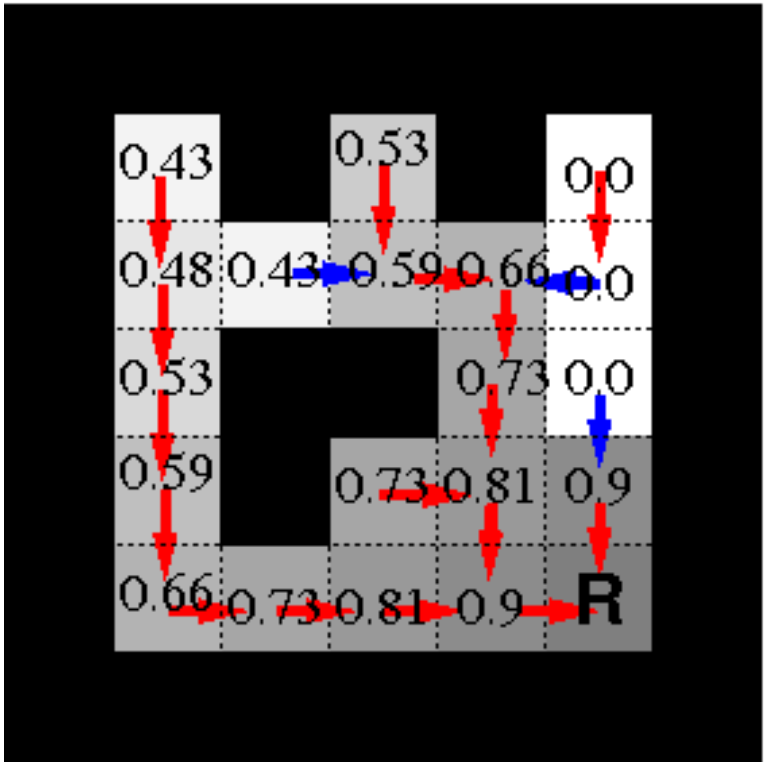
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

Policy Iteration en pratique



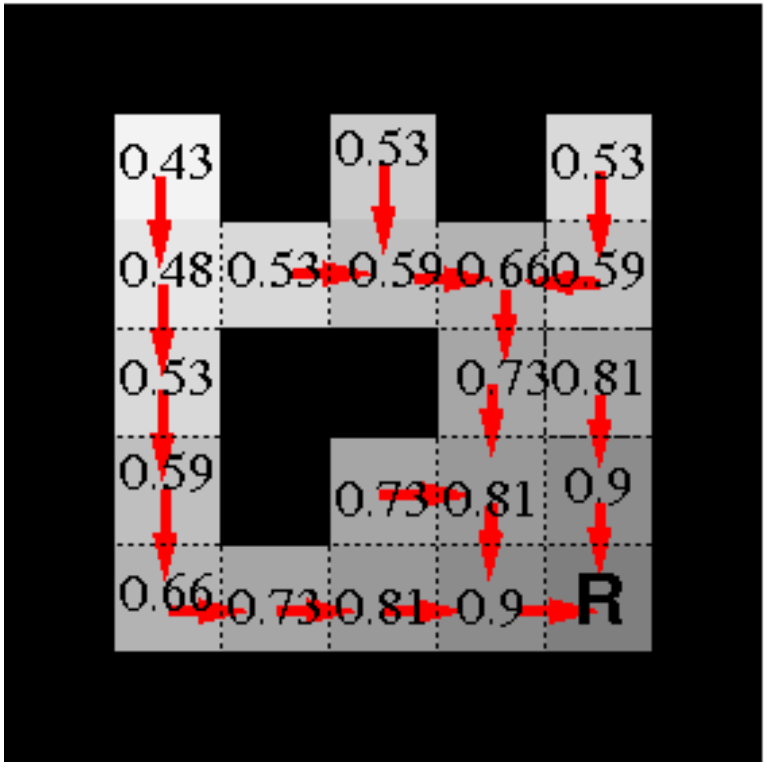
$$\forall s \in S, \pi_{i+1}(s) \leftarrow improve(\pi_i(s), V_i(s))$$

Policy Iteration en pratique



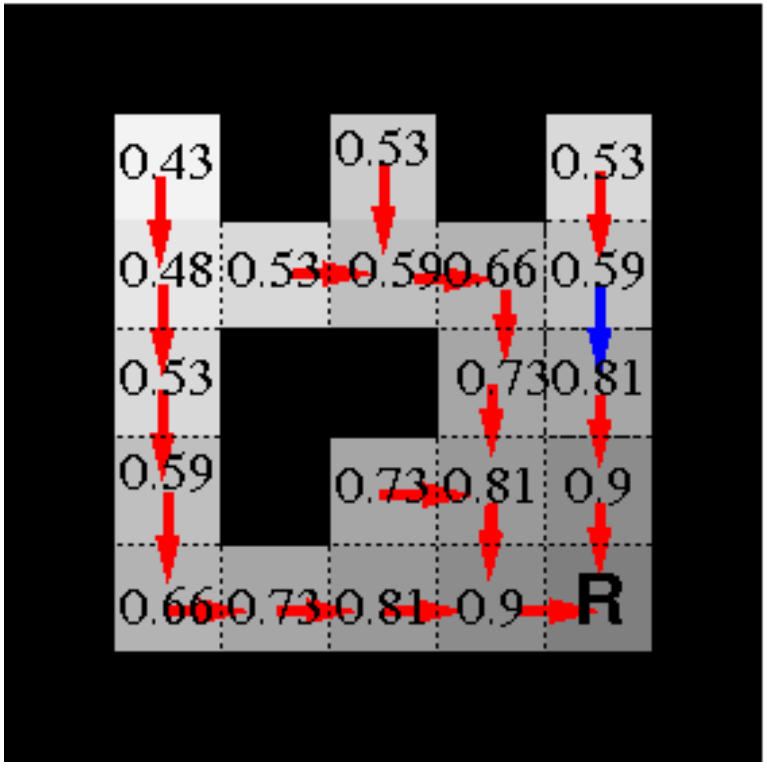
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

Policy Iteration en pratique



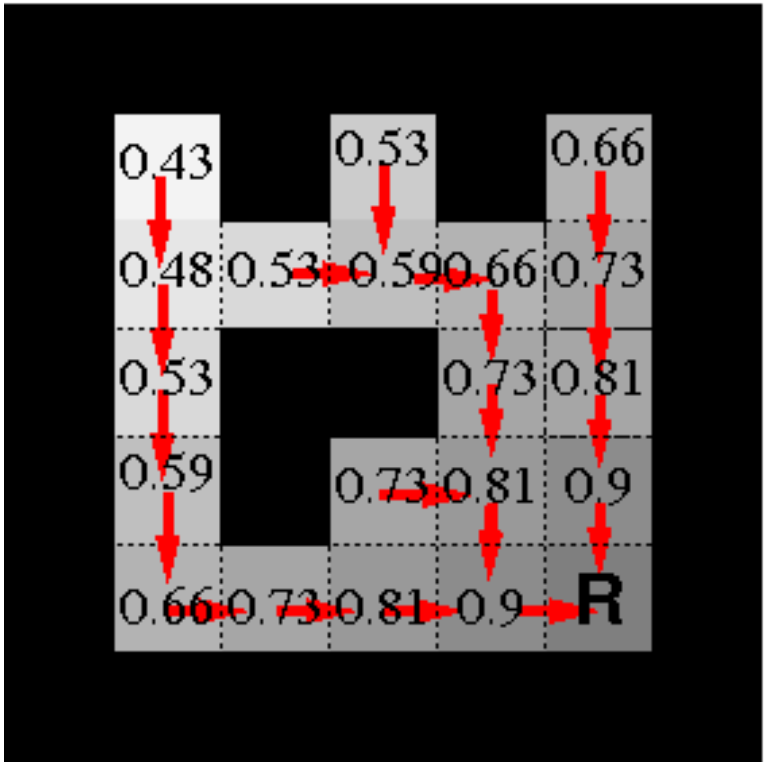
$$\forall s \in S, \pi_{i+1}(s) \leftarrow improve(\pi_i(s), V_i(s))$$

Policy Iteration en pratique



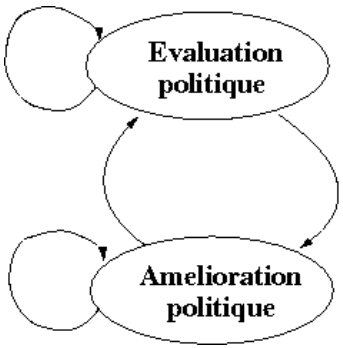
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

Policy Iteration en pratique



$$\forall s \in S, \pi_{i+1}(s) \leftarrow improve(\pi_i(s), V_i(s))$$

Modified Policy Iteration



- Pour obtenir une bonne évaluation de π, itérer l'opérateur de Bellman jusqu'à ce que $\max_s |V_k(s) - V_{k+1}(s)| < \epsilon$.
- De même, pour améliorer π à partir d'une fonction de valeur V^π donnée, itérer jusqu'à ce que la plus grande amélioration de π soit inférieure à ϵ'.
- Au lieu de réaliser un grand nombre d'étapes d'évaluation entre chaque série d'étapes d'amélioration de π, on peut entrelacer un nombre quelconque de ces deux étapes : l'algorithme converge
- Méthode généralisée d'itération sur les politiques [Howard, 1971, Puterman & Shin, 1978].
- *Value Iteration* : cas particulier à une étape.

Programmation linéaire

- résoudre

$$\min_{V \in \mathcal{V}} \sum_{s \in S} V(s)$$

- avec

$$V(s) \geq r(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V(s'), \quad \forall s \in S,\ a \in A$$

- Pour $s \in S$
  - $\pi(s) \leftarrow \text{argmax}_{a \in A} \{ r(s,a) + \gamma \sum_{s'} p(s'|s,a) V(s') \}$
- Retourner V, π

Apprentissage par renforcement

- En programmation dynamique, les fonctions T et r sont données
- Apprentissage par renforcement : construire π^* en ne connaissant *a priori* ni r ni T
  - AR direct (*model-free*) : construire π^* directement, sans construire ni r ni T
  - acteur-critique : cas particulier d'AR direct
  - AR indirect (*model-based*) : construire un modèle de r et T, et s'appuyer sur cette connaissance pour calculer π^*

Estimation incrémentale (ici, de l'utilité immédiate)

- Estimer l'utilité immédiate moyenne reçue en s (problème stochastique)
- $E_k(s) = (r_1 + r_2 + ... + r_k)/k$
- $E_{k+1}(s) = (r_1 + r_2 + ... + r_k + r_{k+1})/(k+1)$
- Donc $E_{k+1}(s) = k/(k+1)E_k(s) + r_{k+1}/(k+1)$
- Ou $E_{k+1}(s) = (k+1)/(k+1)E_k(s) - E_k(s)/(k+1) + r_{k+1}/(k+1)$
- Ou $E_{k+1}(s) = E_k(s) + 1/(k+1)[r_{k+1} - E_k(s)]$
- Suppose encore de stocker k.
- On approxime par $E_{k+1}(s) = E_k(s) + \alpha[r_{k+1} - E_k(s)]$
- Permet de converger vers la moyenne plus ou moins vite selon α sans rien stocker.
- On retrouve cette formule partout.

Erreur de différence temporelle

- Si les estimations $V(s_t)$ et $V(s_{t+1})$, étaient exactes, on aurait :
- $V(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + ...$
- $V(s_{t+1}) = r_{t+1} + \gamma(r_{t+2} + \gamma^2 r_{t+3} + ...$
- Donc on aurait : $V(s_t) = r_t + \gamma V(s_{t+1})$
- $\delta_k = r_k + \gamma V(s_{k+1}) - V(s_k)$ mesure l'erreur entre les valeurs effectives des estimations $V(s)$ et les valeurs qu'elles devraient avoir.
- Pour corriger cette erreur, il faut faire évoluer $V(s_k)$ dans le sens de δ_k

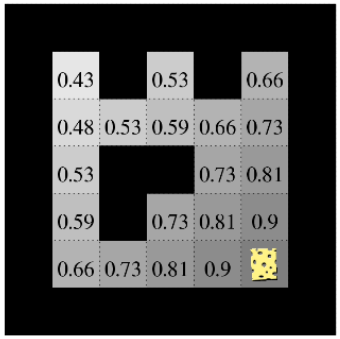
TD(0)

- L'agent réalise une séquence $s_0, a_0, r_1, \cdots, s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, \cdots$ déterminée par sa politique π
- $V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha[r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)]$ (ou $V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha\delta$)
- Les mises à jour se font localement à partir de s_t, s_{t+1} et r_{t+1}.
- Preuve de convergence : [Dayan & Sejnowski, 1994]
- Deux processus de convergence couplés, le premier estime de plus en plus précisément $r(s_t)$, le second approche de mieux en mieux $V^\pi(s_t)$ en propageant de proche en proche.

Limites de TD(0)

- TD(0) estime $V^{\pi}(s)$
- On ne peut pas en déduire π sans connaître T : il faudrait réaliser un pas de regard en avant pour déterminer l'action a qui maximise $V(s')$.
- Trois approches :
  - Travailler sur $Q(s, a)$ plutôt que $V(s)$.
  - Apprendre un modèle de T : apprentissage par renforcement indirect
  - Acteur-critique (tenir à jour une politique en parallèle)

Fonction de valeur et fonction de valeur d'action



state / action	a_0	a_1	a_2	a_3
e_0	0.66	0.88	0.81	0.73
e_1	0.73	0.63	0.9	0.43
e_2	0.73	0.9	0.95	0.73
e_3	0.81	0.9	1.0	0.81
e_4	0.81	1.0	0.81	0.9
e_5	0.9	1.0	0.0	0.9

- La fonction de valeur $V^\pi : S \to \mathbb{R}$ représente la valeur sur le long terme de chaque état (en suivant la politique π). C'est un vecteur avec une valeur par état.
- La fonction de valeur d'action $Q^\pi : S \times A \to \mathbb{R}$ représente la valeur sur le long terme de faire chaque action en chaque état (puis en suivant la politique π). C'est une matrice avec une valeur par état et par action.
- Tous les algorithmes qui travaillent sur la fonction V peuvent être étendus à la fonction Q ($Q^* = TQ^*$, VI, PI).

Sarsa

- Rappel : (TD) $V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$
- Pour chaque $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ observé :
  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
- Suppose de connaître exactement a_{t+1}, donc de figer l'exploration (méthode Off-policy)
- Problème pour inclure de l'exploration aléatoire
- Complique les preuves de convergence [Singh et al., 2000]

Q-Learning

- Pour chaque $(s_t, a_t, r_{t+1}, s_{t+1})$ observé :

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

- $\max_{a \in A} Q(s_{t+1}, a)$ remplace $Q(s_{t+1}, a_{t+1})$
- Méthode Off-policy : Plus besoin de connaître a_{t+1} [Watkins, 1989]
- Politique avec exploration (e.g. ϵ-*greedy*)
- Convergence assurée si suffisamment d'exploration

Q-Learning en pratique

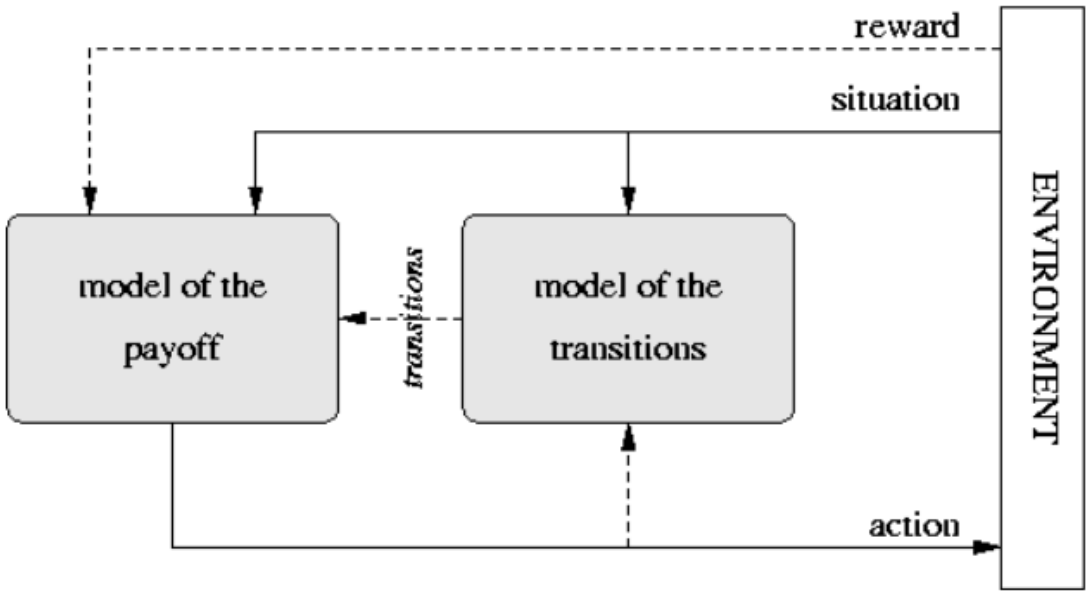
(Q-learning : le film...)

- Construire une table états×actions (*Q-Table*)
- L'initialiser intelligemment
- Appliquer l'équation de mise à jour après chaque action
- Eventuellement, utiliser une table incrémentale
- Problème : très lent en pratique

Traces d'éligibilité

- But : accélérer l'apprentissage
- Procédé naïf : stocker tous les (s, a) et propager les utilités en marche-arrière (compromis quantité de mémoire/vitesse d'apprentissage),
- Problème : compromis mémoire-vitesse, horizon fini
- TD(λ), sarsa (λ) et Q(λ) : plus sophistiqué, horizons infinis.
- Un indicateur $e(s)$ est augmenté de 1 (ou réinitialisé) à chaque fois que s est visité et diminué d'un facteur $\gamma\lambda$ à chaque pas de temps pour tout s.
- TD(λ) : $V(s) \leftarrow V(s) + \alpha\delta e(s)$, (similaire pour sarsa (λ) et Q(λ)),
- Si $\lambda = 0$, l'indicateur s'annule instantanément donc on retombe sur TD(0), sarsa ou Q-learning
- TD(1) = Monte-Carlo...

Apprentissage par Renforcement indirect



- Idée : apprendre un modèle de T et r, puis planifier sur ce modèle en réalisant des mises à jour « dans la tête de l'agent » ([Sutton, 1990a, Sutton, 1990b])
- Les processus d'approximation de T et r sont des apprentissages incrémentaux et auto-supervisés
- Différentes approches :
  - tirer des transitions au hasard dans le modèle et appliquer la différence temporelle
  - utiliser les algorithmes *Policy Iteration* (Dyna-PI) or *Q-learning* (Dyna-Q)

ADyna : algorithme général

Paramètre(s) : α, N

Initialisation : $\forall a \in A, \forall s \in S$ définir $Q_a(s)$ de façon arbitraire

À chaque pas de temps : pour un état s :

Décision :

1. Choisir une action a en fonction des $Q_a(s)$ (par exemple en utilisant ϵ-greedy)
2. Exécuter a, observer s' et r

Apprentissage :

3. $Q_a(s) \leftarrow Q_a(s) + \alpha(r + \gamma \max_{a'} [Q_{a'}(s')] - Q_a(s))$
4. Mettre à jour $\hat{T}(s,a)$ à partir de $\langle s, a, s' \rangle$ et $\hat{R}(s,a)$ à partir de $\langle s, a, r \rangle$

Planification :

5. Répéter N fois :
   (a) $s \leftarrow$ un état observé choisi aléatoirement
   (b) $a \leftarrow$ une action déjà exécutée dans s et choisie aléatoirement
   (c) Déterminer s' tel que $\hat{P}(s'|s,a) = 1$ (l'environnement est supposé déterministe)
   (d) $r \leftarrow \hat{R}(s,a)$
   (e) $Q_a(s) \leftarrow Q_a(s) + \alpha(r + \gamma \max_{a'} [Q_{a'}(s')] - Q_a(s))$

ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

Architectures Dyna et généralisation

(exemple avec un modèle erroné)
(exemple avec un bon modèle)

- Grâce au modèle des transitions, Dyna peut propager plus souvent
- Problème : dans le cas stochastique, le modèle des transitions est en $card(S) \times card(S) \times card(A)$
- Intérêt de l'apprentissage par renforcement indirect compact
- MACS [Gérard et al., 2005] : Dyna avec généralisation (systèmes de classeurs)
- SPITI [Degris et al., 2006] : Dyna avec généralisation (MDP factorisés)

De $Q-$ *learning* aux approches acteur-critique (1)

état / action	a_0	a_1	a_2	a_3
e_0	0.66	0.88	0.81	0.73
e_1	0.73	0.63	0.9	0.43
e_2	0.73	0.9	0.95	0.73
e_3	0.81	0.9	1.0	0.81
e_4	0.81	1.0	0.81	0.9
e_5	0.9	1.0	0.0	0.9

- Dans $Q-$ *learning*, il faut chercher un max dans la $Q-$ *Table* à chaque pas
- C'est très cher quand le nombre d'actions augmente (optimisation si continu)

De Q – *learning* aux approches acteur-critique (2)

état / action	a_0	a_1	a_2	a_3
e_0	0.66	0.88*	0.81	0.73
e_1	0.73	0.63	0.9*	0.43
e_2	0.73	0.9	0.95*	0.73
e_3	0.81	0.9	1.0*	0.81
e_4	0.81	1.0*	0.81	0.9
e_5	0.9	1.0*	0.0	0.9

- On peut stocker la meilleure valeur pour chaque état
- Du coup, on calcule moins de max (seulement dans un cas)

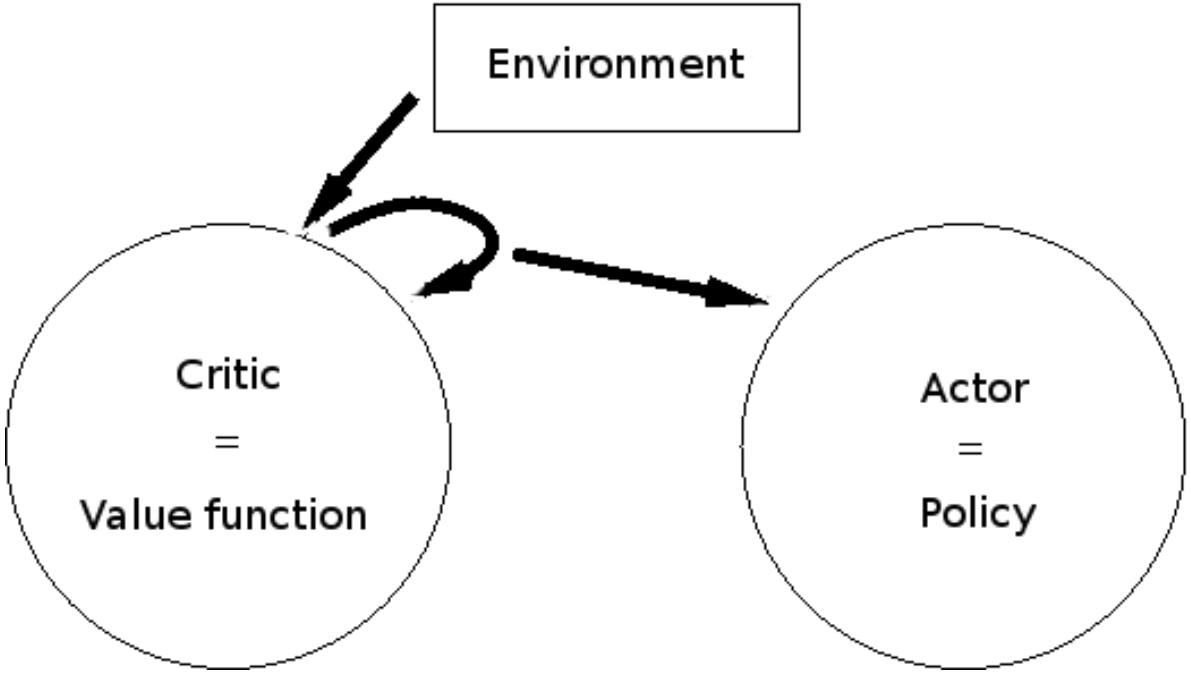
De Q – *learning* aux approches acteur-critique (3)

état / action	a_0	a_1	a_2	a_3
e_0	0.66	0.88*	0.81	0.73
e_1	0.73	0.63	0.9*	0.43
e_2	0.73	0.9	0.95*	0.73
e_3	0.81	0.9	1.0*	0.81
e_4	0.81	1.0*	0.81	0.9
e_5	0.9	1.0*	0.0	0.9

état	action choisie
e_0	a_1
e_1	a_2
e_2	a_2
e_3	a_2
e_4	a_1
e_5	a_1

- Stocker le max est équivalent à stocker une politique
- Mise à jour de la politique en fonction de la mise à jour de la valeur
- Schéma de base des architectures acteur-critique

Architecture acteur-critique naïve

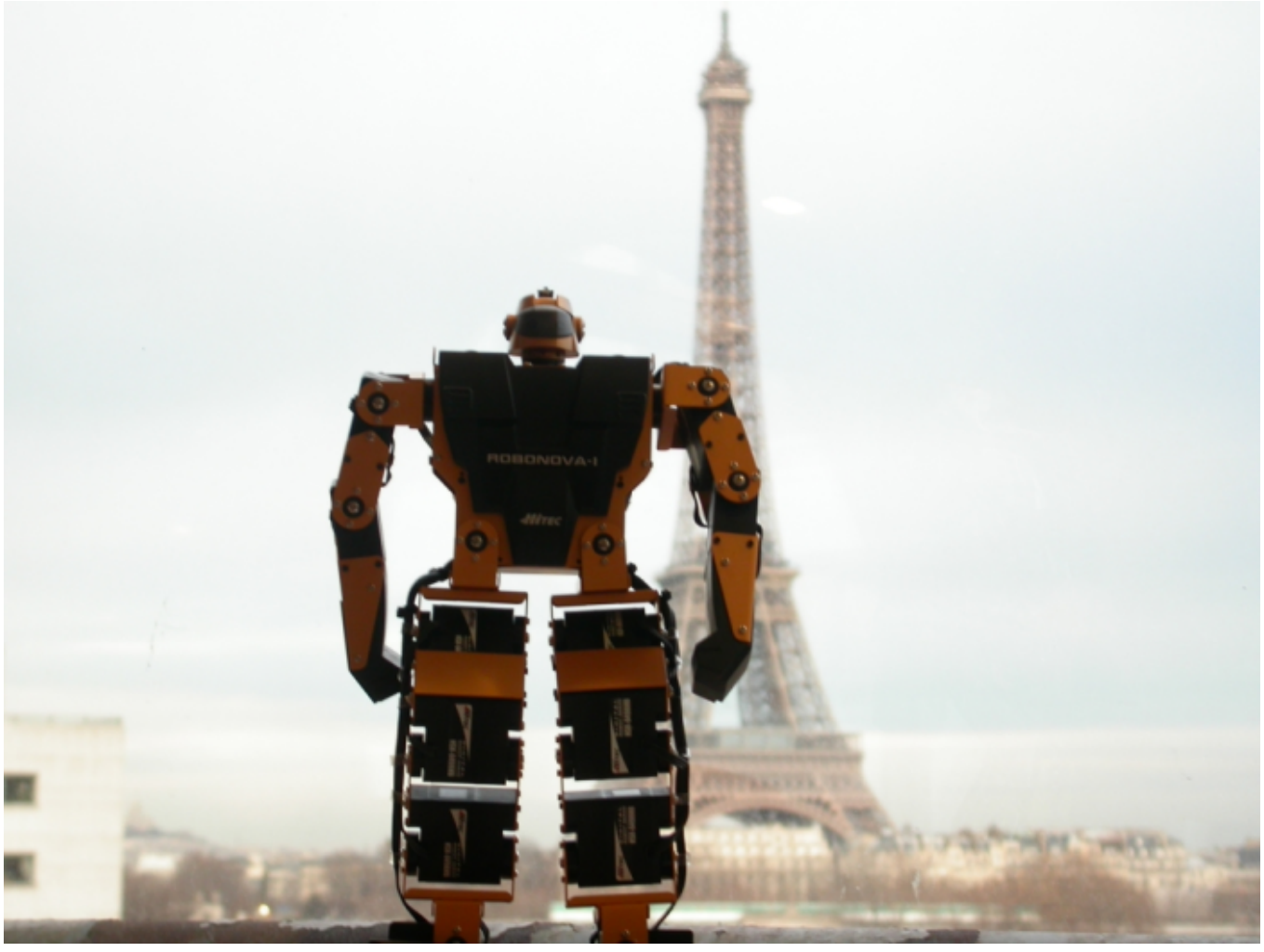


- Proche de *Policy iteration*
- Etats et actions discrets, politique stochastique
- Une mise à jour du critique engendre une mise à jour de l'acteur
- Critique : calcule δ et met à jour $V(s)$ avec $V_k(s) \leftarrow V_k(s) + \alpha_k \delta_k$
- Acteur : $P^\pi(a|s) = P^\pi(a|s) + \alpha_k \prime \delta_k$
- Plus besoin de max sur les actions
- Par contre, il faut savoir tirer une action à partir d'une politique probabiliste (pas évident pour des actions continues)

Messages

- La programmation dynamique (*value iteration*, *policy iteration*), c'est quand on connaît r et T
- Les algorithmes d'apprentissage par renforcement sont tous basés sur l'erreur de différence temporelle δ_k
- Ce sont des algorithmes basés sur des statistiques incrémentales
- L'apprentissage par renforcement indirect combine apprentissage de modèle et programmation dynamique
- Les architectures acteur-critique sont des approches directes, proches de *policy iteration*

Des questions ?



Buffet, O. & Sigaud, O. (2008).

Processus décisionnels de Markov en intelligence artificielle.
Lavoisier.

Dayan, P. & Sejnowski, T. (1994).

Td(lambda) converges with probability 1.
Machine Learning, 14(3).

Degris, T., Sigaud, O., & Wuillemin, P.-H. (2006).

Learning the Structure of Factored Markov Decision Processes in Reinforcement Learning Problems.
Edité dans *Proceedings of the 23rd International Conference on Machine Learning (ICML'2006)*, pages 257–264, CMU, Pennsylvania.

Gérard, P., Meyer, J.-A., & Sigaud, O. (2005).

Combining latent learning with dynamic programming in MACS.
European Journal of Operational Research, 160 :614–637.

Howard, R. A. (1971).

Dynamic Probabilistic Systems.
Wiley.

Puterman, M. L. & Shin, M. C. (1978).

Modified Policy Iteration Algorithms for Discounted Markov Decision Problems.
Management Science, 24 :1127–1137.

Sigaud, O. & Buffet, O. (2010).

Markov Decision Processes in Artificial Intelligence.
iSTE - Wiley.

Singh, S. P., Jaakkola, T., Littman, M. L., & Szepesvari, C. (2000).

Convergence Results for Single-Step On-Policy Reinforcement Learning Algorithms.
Machine Learning, 38(3) :287–308.

Sutton, R. S. (1990a).

Integrating architectures for learning, planning, and reacting based on approximating dynamic programming.
Edité dans *Proceedings of the Seventh International Conference on Machine Learning ICML'90*, pages 216–224, San Mateo, CA. Morgan Kaufmann.

Sutton, R. S. (1990b).

Planning by incremental dynamic programming.
Edité dans *Proceedings of the Eighth International Conference on Machine Learning*, pages 353–357, San Mateo, CA. Morgan Kaufmann.

Sutton, R. S. & Barto, A. G. (1998).

Reinforcement Learning : An Introduction.
MIT Press.

Watkins, C. J. C. H. (1989).

Learning with Delayed Rewards.
Thèse de doctorat, Psychology Department, University of Cambridge, England.