

# Apprentissage par renforcement

Olivier Sigaud  
<http://people.isir.upmc.fr/sigaud>

UE Animat, M2 IAD

15 février 2013



# Plan

## Introduction

Différents types d'apprentissage

## Programmation dynamique

Processus Décisionnel de Markov

Méthodes itératives

## Apprentissage par renforcement

Généralités

Différences temporelles

Approches fondées sur la fonction de valeur d'action

Approche acteur-critique

AR indirect

## Action continue

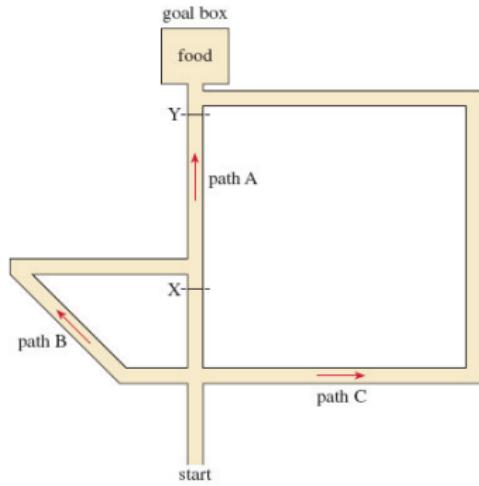
Approches acteur pur

Approches acteur-critique

Approches EM-based

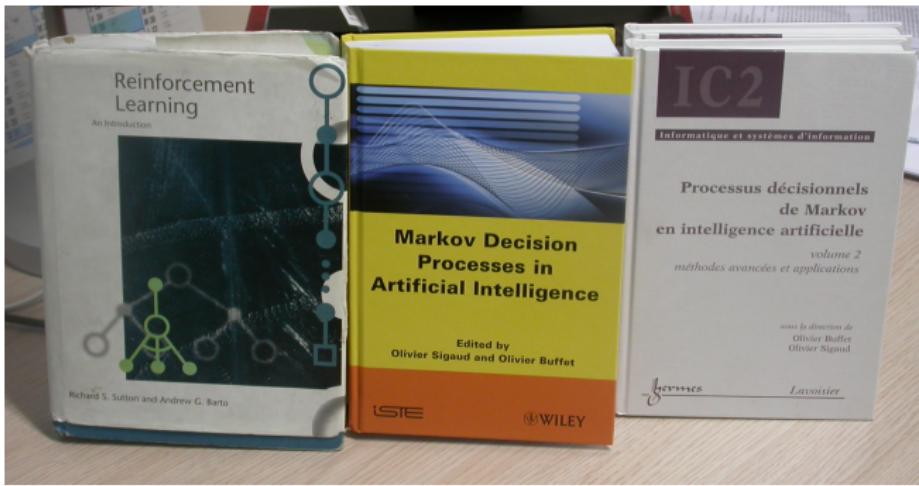
## Références

## Sélection de l'action/planification



- ▶ Apprentissage par essai et erreur (modèle principal : Apprentissage par renforcement)
  - ▶ Les maths comme lien entre la psycho et les neurosciences

## Livres introductifs

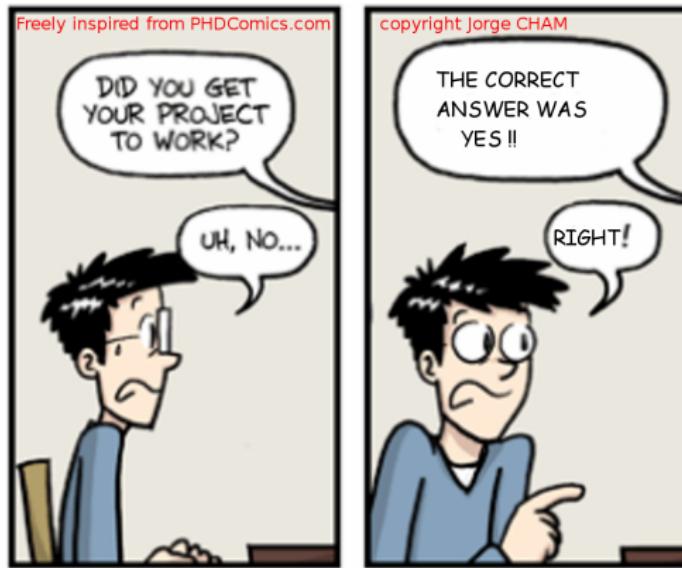


1. [Sutton & Barto, 1998] : la bible du domaine
2. [Buffet & Sigaud, 2008] : en français
3. [Sigaud & Buffet, 2010] : traduction (améliorée) de 2



Reinforcement Learning : An Introduction. MIT Press.

## 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



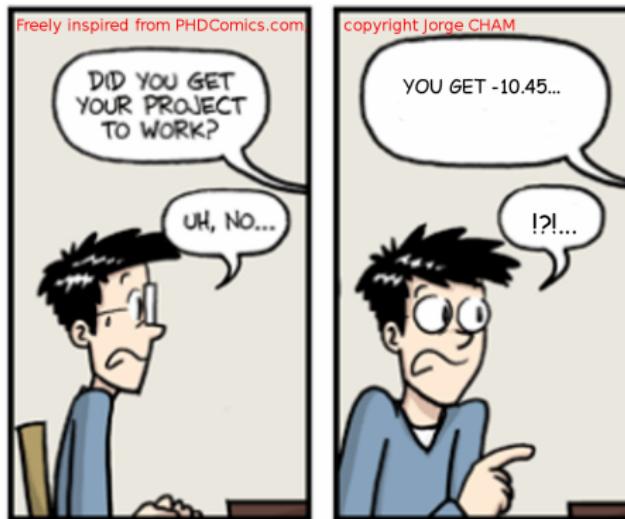
- ▶ L'agent prédit l'état suivant
  - ▶ Correction donnée par l'environnement → Pas besoin de superviseur
  - ▶ Application : apprentissage de prédictions

## 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

## Different mécanismes d'apprentissage : synthèse

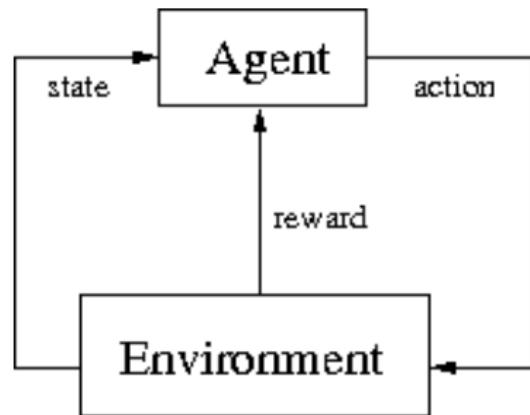
- ▶ Apprentissage supervisé : pour une entrée, l'apprenti reçoit la sortie qu'il aurait dû produire
- ▶ Apprentissage par renforcement : pour une entrée, l'apprenti reçoit un scalaire qui représente la valeur immédiate de sa sortie
- ▶ Apprentissage non-supervisé : pour une entrée, l'apprenti ne reçoit rien, il extrait simplement des corrélations
- ▶ NB : les apprentissages auto-supervisé et non-supervisé sont difficiles à distinguer

## Objectifs du cours

- ▶ Présenter les bases de l'A/R discret et de la programmation dynamique
- ▶ Bien distinguer :
- ▶ Programmation dynamique
- ▶ Apprentissage par renforcement direct
- ▶ Approches acteur-critique
- ▶ Apprentissage par renforcement indirect

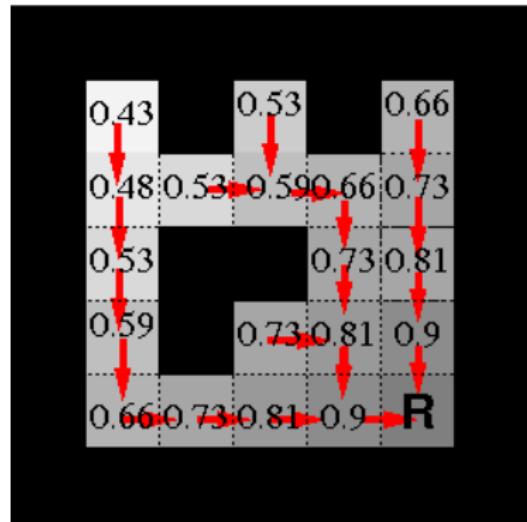
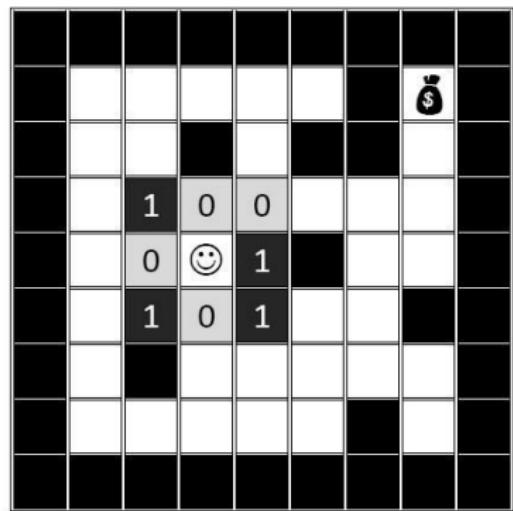
## Processus Décisionnel de Markov

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



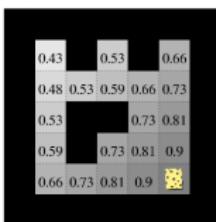
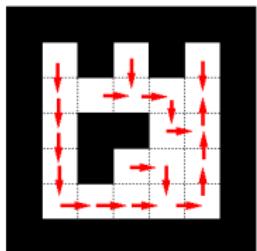
- ▶ Le formalisme des PDM définit  $s^{t+1}$  et  $r^{t+1}$  en  $f(s_t, a_t)$
- ▶ Un PDM décrit un problème, pas une politique

## Limites des PDM



- ▶ La propriété de Markov est violée si :
  - ▶ l'état ne contient pas toute l'information
  - ▶ l'état suivant dépend de l'action de plusieurs agents...
  - ▶ les transitions dépendent du temps...

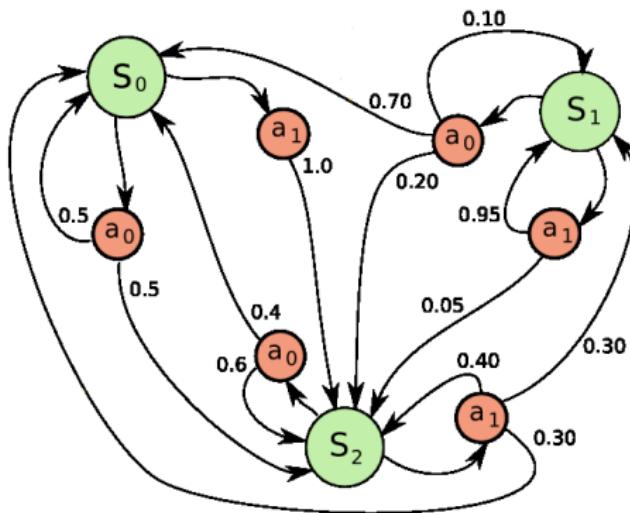
## Fonction de valeur



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

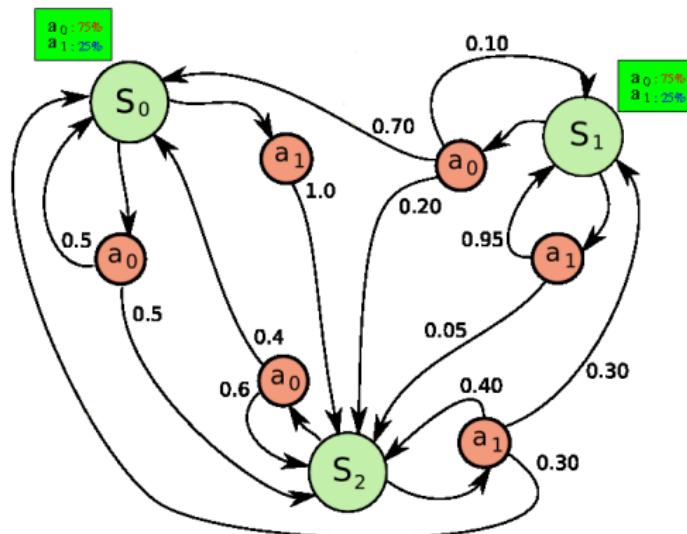
- ▶ Une **politique** (ou stratégie) est une fonction  $\pi : S \rightarrow A$ . Dans le cas déterministe, c'est un **vecteur** avec une action par état.
- ▶ But de la planification (et de l'apprentissage) : trouver une politique qui maximise le cumul des utilités au long terme
- ▶ La **fonction de valeur**  $V^\pi : S \rightarrow \mathbb{R}$  représente à partir de chaque état le cumul des utilités futures en suivant  $\pi$ . 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  $\pi$ ). 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  $Q$

## Problème stochastique



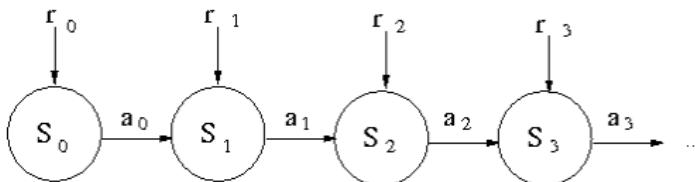
- Le cas déterministe est un cas particulier du stochastique
- $T(s^t, a^t, s^{t+1}) = p(s'|s, a)$

## Politique stochastique

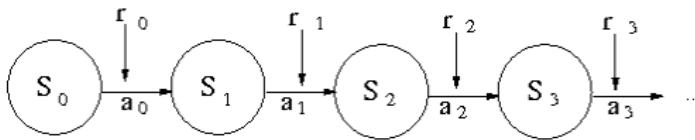


- ▶ Pour tout MDP, il existe au moins une politique déterministe optimale

## Récompenses sur une chaîne de Markov : sur les états ou les actions ?



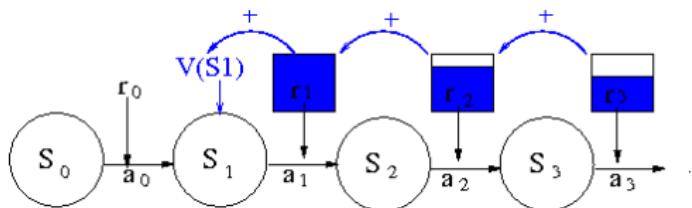
- ▶ Récompenses sur les états



- ▶ Récompenses sur les actions dans les états
- ▶ Ici, on utilise les récompenses sur les actions dans les états (notées  $r(s, a)$ )

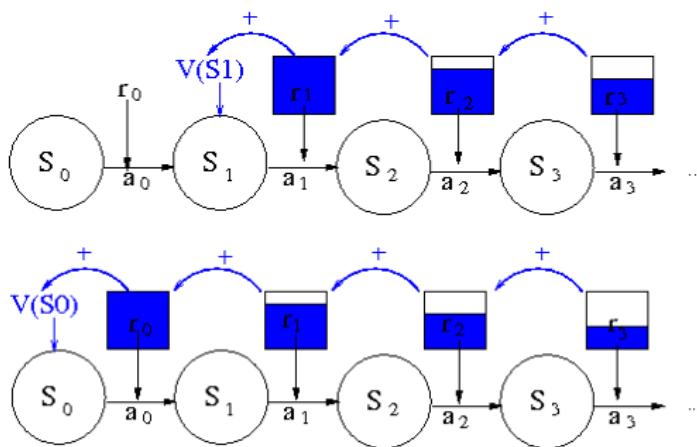
## Critères d'agrégation

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



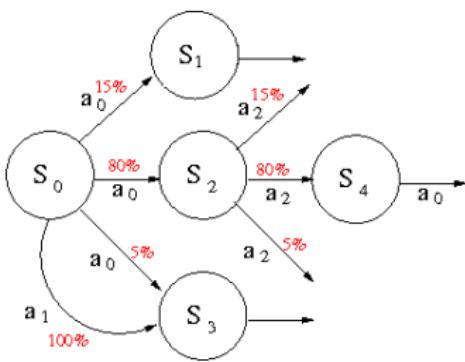
- ▶ Simple somme ou critère moyen (horizon fini), critère moyen sur une fenêtre
- ▶ Critère actualisé ( $\gamma$ -pondéré) :  $V^\pi(s_{t_0}) = \sum_{t=t_0}^{\infty} \gamma^t r(s_t, \pi(s_t))$
- ▶  $\gamma \in [0, 1]$  : facteur d'actualisation
  - ▶ si  $\gamma = 0$ , seule l'utilité immédiate est importante
  - ▶ 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



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

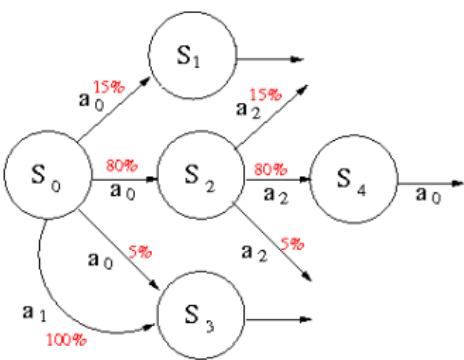
## Equation de Bellman : cas général



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

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

## Opérateur de Bellman

- ▶ 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  $T^\pi$ ) l'application qui consiste à propager la valeur associée à une politique sur un pas

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

- ▶ On appelle **opérateur d'optimalité de Bellman** (noté  $T^*$ ) l'application qui consiste à propager la valeur associée à la meilleure action possible sur un pas

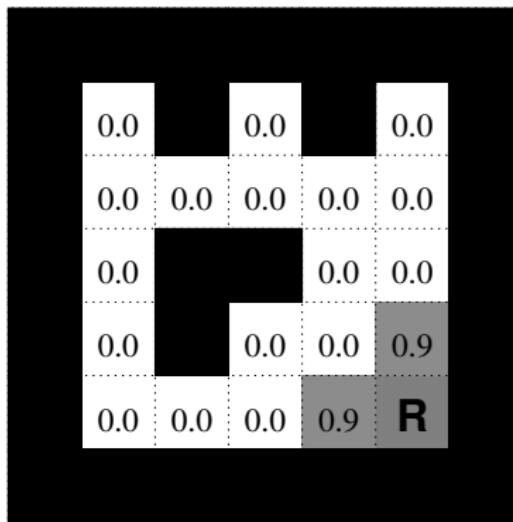
$$V(s) \leftarrow \max_{a \in A} \left[ r(s, a) + \gamma \sum_{s'} p(s'|s, a) V(s') \right]$$

- ▶ Si  $\gamma \in ]0, 1[$ , les deux opérateurs de Bellman sont contractants, donc ils ont un point fixe

## Opérateur de Bellman et Programmation Dynamique

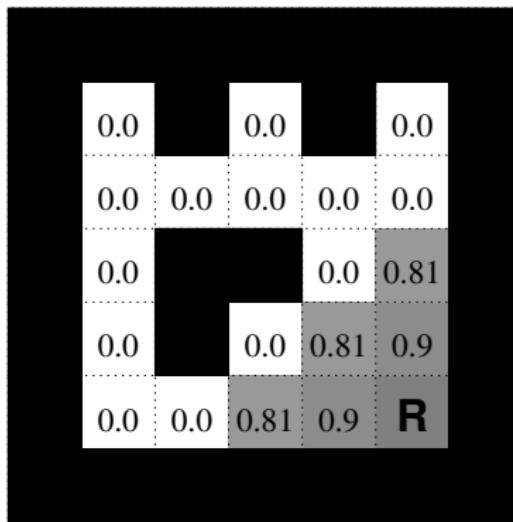
- ▶ On appelle fonction de valeur optimale (notée  $V^*$ ) la fonction de valeur associée à la politique qui rapporte le plus
- ▶ On montre que la fonction de valeur optimale est un point fixe de l'opérateur d'optimalité de Bellman  $T^* : V^* = T^* V^*$
- ▶ *Value Iteration* : trouver  $V^*$  en itérant  $V_{i+1} \leftarrow T^* V_i$
- ▶ *Policy Iteration* : *policy evaluation* (avec  $V_{i+1}^\pi \leftarrow T^\pi V_i^\pi$ ) + *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



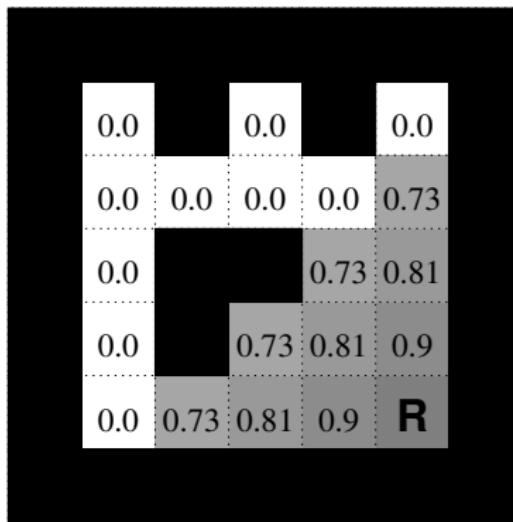
$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} [r(s, a) + \gamma \sum_{s'} p(s'|s, a) V_i(s')]$$

## Value Iteration en pratique



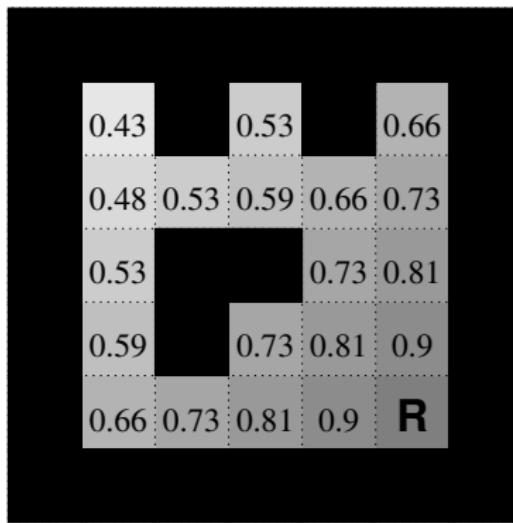
$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} [r(s, a) + \gamma \sum_{s'} p(s'|s, a) V_i(s')]$$

## Value Iteration en pratique



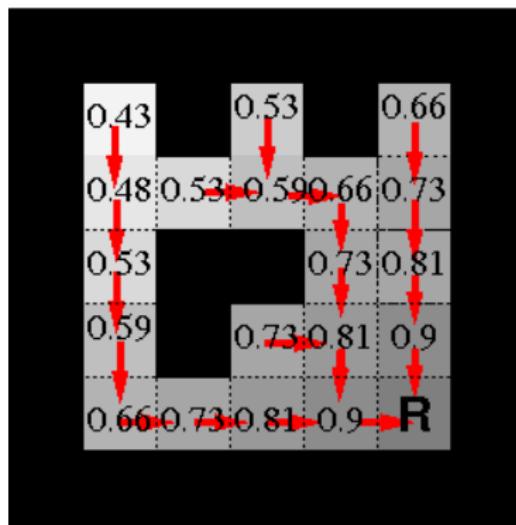
$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} [r(s, a) + \gamma \sum_{s'} p(s'|s, a) V_i(s')]$$

## Value Iteration en pratique



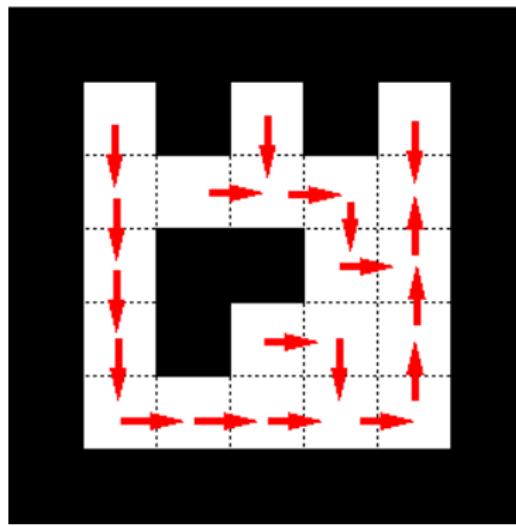
$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} [r(s, a) + \gamma \sum_{s'} p(s'|s, a) V_i(s')]$$

## Value Iteration en pratique



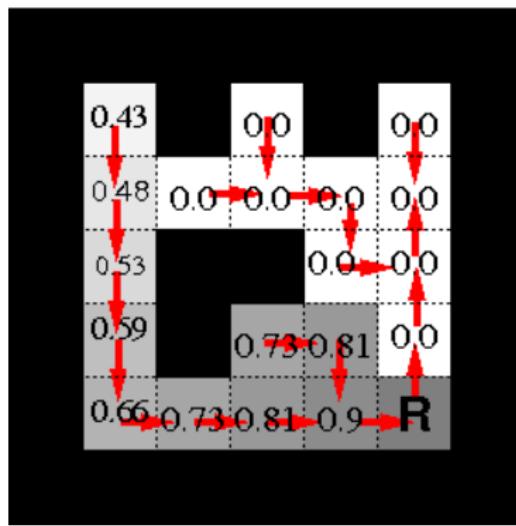
$$\pi^*(s) = \arg \max_{a \in A} [r(s, a) + \gamma \sum_{s'} p(s'|s, a) V^*(s')]$$

## Policy Iteration en pratique

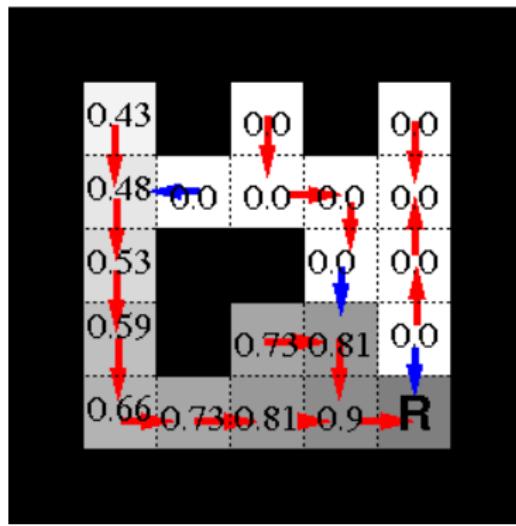


$$\forall s \in S, V_i(s) \leftarrow \text{evaluate}(\pi_i(s))$$

## Policy Iteration en pratique

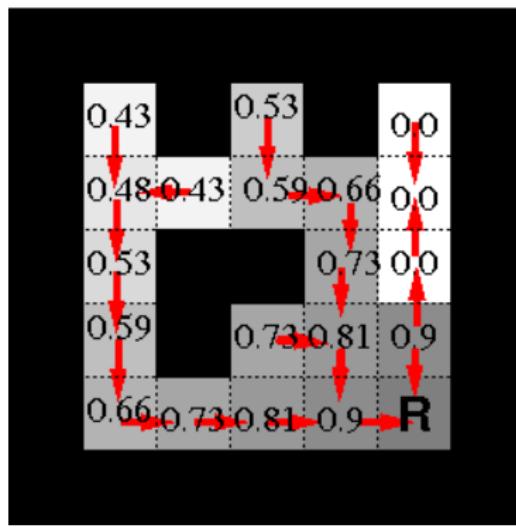

$$\forall s \in S, \pi_{i+1}(s) \leftarrow \text{improve}(\pi_i(s), V_i(s))$$

## Policy Iteration en pratique

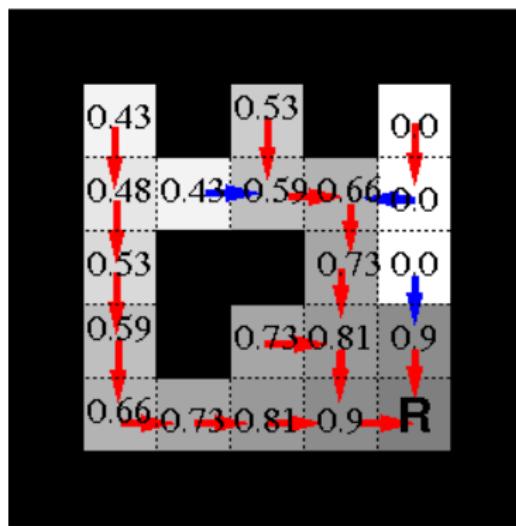


$$\forall s \in S, V_i(s) \leftarrow \text{evaluate}(\pi_i(s))$$

## Policy Iteration en pratique

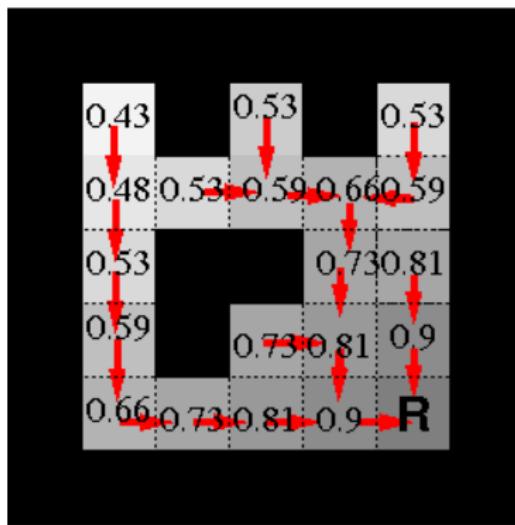

$$\forall s \in S, \pi_{i+1}(s) \leftarrow \text{improve}(\pi_i(s), V_i(s))$$

## Policy Iteration en pratique

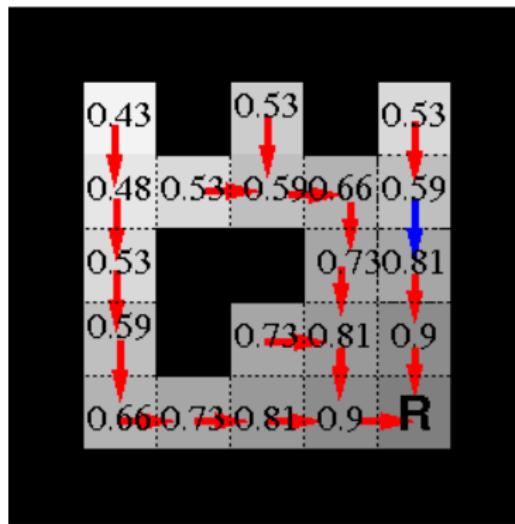


$$\forall s \in S, V_i(s) \leftarrow \text{evaluate}(\pi_i(s))$$

## Policy Iteration en pratique

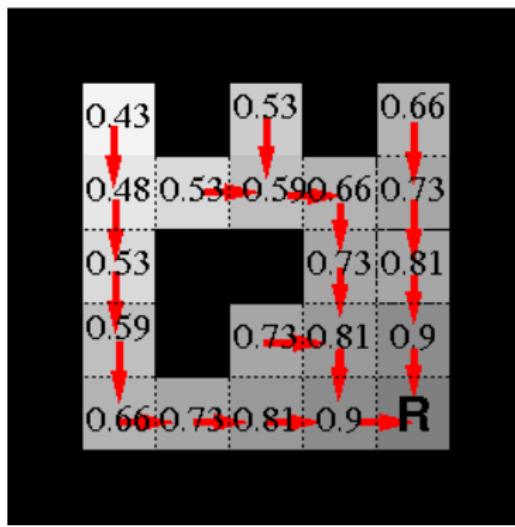

$$\forall s \in S, \pi_{i+1}(s) \leftarrow \text{improve}(\pi_i(s), V_i(s))$$

## Policy Iteration en pratique



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

## Policy Iteration en pratique


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

## Familles de méthodes

- ▶ Critique : fonction de valeur (d'action) → évaluation de la politique
- ▶ Acteur : la politique elle-même
- ▶ *Value iteration* est une méthode “critique pur” : elle itère sur la fonction de valeur (d'action) jusqu'à convergence, puis elle en déduit une politique optimale
- ▶ *Policy iteration* est une méthode “acteur-critique” : elle met à jour en parallèle une politique et une fonction de valeur (d'action)
- ▶ Dans le cas continu, on s'intéresse aussi aux méthodes “acteur pur” (descente de gradient de performance sur les politiques)

## Apprentissage par renforcement

- ▶ En programmation dynamique, les fonctions  $T$  et  $r$  sont données
- ▶ Apprentissage par renforcement : construire  $\pi^*$  en ne connaissant *a priori* ni  $r$  ni  $T$ 
  - ▶ AR direct (*model-free*) : construire  $\pi^*$  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  $\pi^*$

## Estimation incrémentale (ici, de l'utilité immédiate)

- ▶ Estimer l'utilité immédiate moyenne reçue en  $s$  (problème stochastique)
- ▶  $E_k(s) = (r_1 + r_2 + \dots + r_k)/k$
- ▶  $E_{k+1}(s) = (r_1 + r_2 + \dots + r_k + r_{k+1})/(k + 1)$
- ▶ 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

$$E_{k+1}(s) = E_k(s) + \alpha[r_{k+1} - E_k(s)] \quad (1)$$

- ▶ Permet de converger vers la moyenne plus ou moins vite selon  $\alpha$  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 :
- ▶  $V(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots$
- ▶  $V(s_{t+1}) = r_{t+1} + \gamma(r_{t+2} + \gamma^2 r_{t+3} + \dots)$
- ▶ 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  $\delta_k$

## Méthodes de Monte Carlo

- ▶ Très utilisées dans les jeux (Go...) pour évaluer un état
- ▶ Engendrer un grand nombre de trajectoires :  $s_0, s_1, \dots, s_N$  et observer les récompenses  $r_0, r_1, \dots, r_N$
- ▶ Mettre à jour la valeur des états  $V(s_k)$ ,  $k = 0, \dots, N - 1$  avec :

$$V(s_k) \leftarrow V(s_k) + \alpha(s_k)(r_k + r_{k+1} + \dots + r_N - V(s_k))$$

- ▶ Utilise la méthode d'estimation incrémentale de la moyenne (1)

## Méthode d'évaluation : TD(0)

- ▶ L'agent réalise une séquence  $s_0, a_0, r_1, \dots, s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, \dots$  déterminée par sa politique  $\pi$
- ▶  $V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha[r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)]$  (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  $r_{t+1}$ .
- ▶ Deux processus de convergence couplés qui estiment de plus en plus précisément  $V^\pi(s_t)$ , le premier localement au fil des essais, le second en propageant depuis les états suivants.



TD(lambda) converges with probability 1. *Machine Learning*, 14(3) :295-301

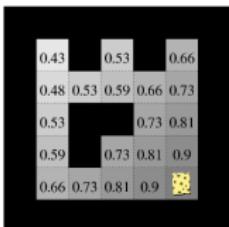
## Méthode de différences temporelles

- ▶ Les méthodes TD combinent les propriétés de la programmation dynamique et de Monte Carlo :
- ▶ dans Monte Carlo,  $T$  et  $r$  sont **inconnus** (il faut des trajectoires), mais la mise à jour de la valeur est **globale**,
- ▶ en programmation dynamique,  $T$  et  $r$  sont **connus**, mais la mise à jour est **locale**
- ▶ TD : comme en PD,  $V(s_t)$  est mis à jour **localement** à partir d'une estimation de  $V(s_{t+1})$ , sachant que  $T$  et  $r$  sont inconnus
- ▶ NB : Les méthodes de Monte Carlo peuvent être reformulées incrémentalement en utilisant l'erreur de différence temporelle  $\delta_k$

## Limites de TD(0)

- ▶ TD(0) estime  $V^\pi(s)$
- ▶ On ne peut pas en déduire  $\pi$  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  $V(s)$ .
  - ▶ Acteur-critique (tenir à jour une politique en parallèle)
  - ▶ Apprendre un modèle de  $T$  : apprentissage par renforcement indirect

## Rappel : fonction de valeur / 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  $V^\pi : S \rightarrow \mathbb{IR}$  représente la valeur sur le long terme de chaque état (en suivant la politique  $\pi$ )
- ▶ La fonction de valeur d'action  $Q^\pi : S \times A \rightarrow \mathbb{IR}$  représente la valeur sur le long terme de faire chaque action en chaque état (puis en suivant la politique  $\pi$ )
- ▶ Tous les algorithmes qui travaillent sur la fonction  $V$  peuvent être étendus à la fonction  $Q$  ( $Q^* = TQ^*$ , VI, PI)
- ▶ Travailler sur la fonction  $Q$  permet de savoir quelle action choisir

# Sarsa

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

- ▶ 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 **On-policy**)
- ▶ Problème pour inclure de l'exploration aléatoire
- ▶ Complique les preuves de convergence



Singh, S. P., Jaakkola, T., Littman, M. L., and Szepesvari, C. (2000). Convergence Results for Single-Step On-Policy Reinforcement Learning Algorithms. *Machine Learning*, 38(3) :287-308.

## *Q-Learning*

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

- ▶ 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  $a_{t+1}$
- ▶ Politique avec exploration (e.g.  $\epsilon$ -greedy)
- ▶ Convergence assurée si suffisamment d'exploration



Watkins, C. J. C. H. (1989). Learning with Delayed Rewards. *Thèse de doctorat, Psychology Department, University of Cambridge, England*

## *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

## Architecture acteur-critique naïve

- ▶ Etats et actions discrets, politique stochastique
- ▶ Une mise à jour du critique engendre une mise à jour de l'acteur
- ▶ Critique : calcule  $\delta$  et met à jour  $V(s)$  avec  $V_k(s) \leftarrow V_k(s) + \alpha_k \delta_k$
- ▶ Acteur :  $P^\pi(a|s) = P^\pi(a|s) + \alpha_k / \delta_k$
- ▶ 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)

## 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)

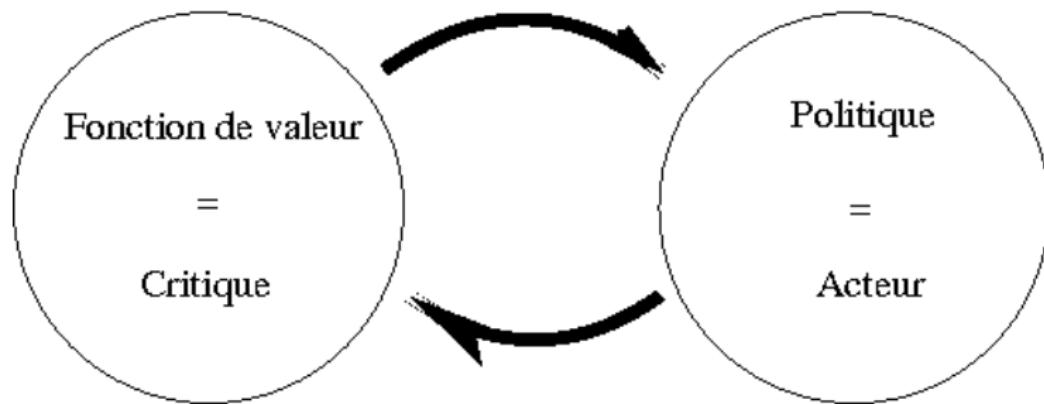
## De $Q$ – learning aux approches acteur-critique (3)

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
$e_0$	$a_1$
$e_1$	$a_2$
$e_2$	$a_2$
$e_3$	$a_2$
$e_4$	$a_1$
$e_5$	$a_1$

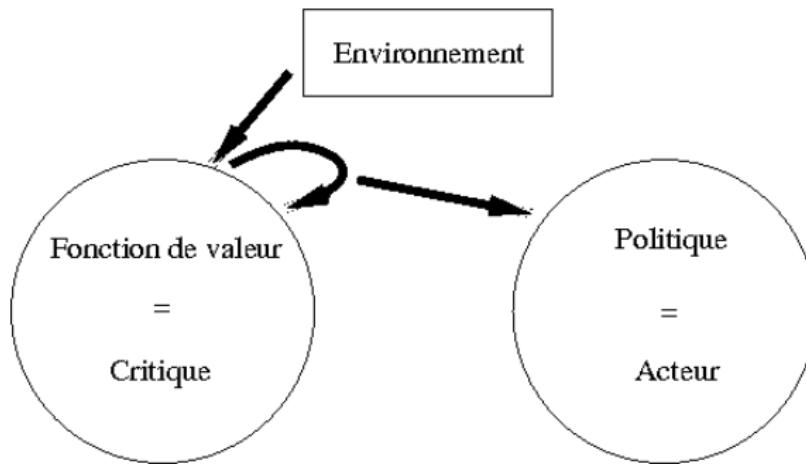
- ▶ Stocker le max est équivalent à stocker une politique
- ▶ Mise à jour de la politique en fonction de la mise à jour de la valeur
- ▶ Schéma de base des architectures acteur-critique

## Programmation dynamique et acteur-critique (1)



- ▶ Les méthodes de *policy iteration* alternent deux étapes :
  1. Evaluation : mettre à jour par différences temporelles une valeur à chaque état ( $V(s)$ ) ou chaque couple état-action ( $Q(s, a)$ ) en fonction de la politique courante
  2. Amélioration de la politique : monter le gradient de la fonction de valeur

## Programmation dynamique et acteur-critique (2)

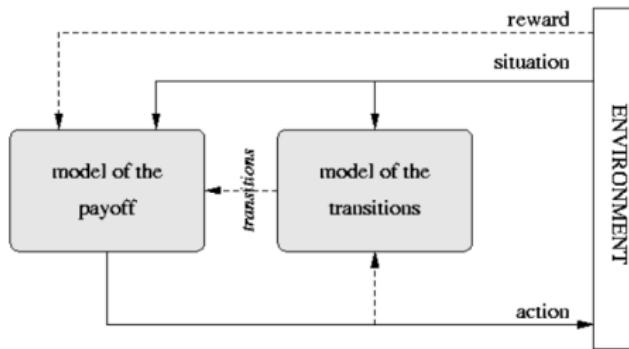


- ▶ Apprentissage par renforcement : cas où on ne connaît pas les fonctions de transition et de récompense
- ▶ On tient à jour en parallèle une représentation de la fonction de valeur et une représentation de la politique
- ▶ Une mise à jour de la fonction de valeur induit une mise à jour locale de la politique
- ▶ La prise d'information sur l'environnement entraîne des mises à jour du critique et de l'acteur

## 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( $\lambda$ ), sarsa ( $\lambda$ ) et Q( $\lambda$ ) : 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( $\lambda$ ) :  $V(s) \leftarrow V(s) + \alpha \delta e(s)$ , (similaire pour sarsa ( $\lambda$ ) et Q( $\lambda$ )),
- ▶ 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 par des mises à jour « dans la tête de l'agent » ([Sutton, 1990a, Sutton, 1990b])
- ▶ Approcher  $T$  et  $r$  par des apprentissages incrémentaux et **auto-supervisés**
- ▶ Différentes approches :
  - ▶ tirer des transitions au hasard et appliquer Bellman dans le modèle
  - ▶ utiliser *Policy Iteration* (Dyna-P) or *Q-learning* (Dyna-Q) pour planifier
  - ▶ plus fin : propagation “raisonnée” : *Prioritized Sweeping* [Moore, 1994, ?]



# ADyna : algorithme général

---

**Paramètre(s) :**  $\alpha, N$

---

**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  $\epsilon$ -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)  $Q_a(s) \leftarrow Q_a(s) + \alpha(r + \gamma \max_{a'} [Q_{a'}(s')] - Q_a(s))$
-

## 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  $\text{card}(S) \times \text{card}(S) \times \text{card}(A)$
- ▶ 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)



Learning the Structure of Factored Markov Decision Processes in Reinforcement Learning Problems,  
Proceedings ICML 23, 257-264

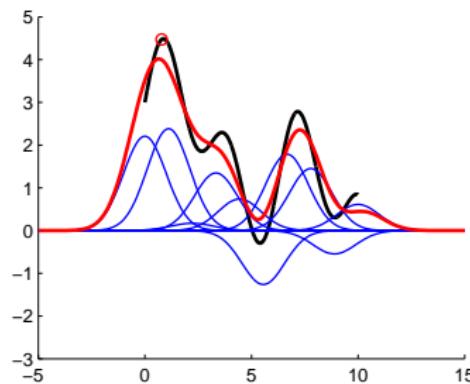
## Messages

- ▶ 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  $\delta_k$
- ▶ Ce sont des algorithmes basés sur des statistiques incrémentales
- ▶ Les architectures acteur-critique sont des approches directes, proches de *policy iteration*
- ▶ L'apprentissage par renforcement indirect combine apprentissage de modèle et programmation dynamique

## Les problèmes à action continue

- ▶ Pour faire de l'apprentissage de la commande, il faut faire de l'apprentissage par renforcement avec des actions continues
- ▶ L'apprentissage par renforcement avec espace d'état continu a connu des développements considérables en 10 ans
- ▶ Le cas de l'action continue est beaucoup plus difficile, bien que la programmation dynamique puisse ses racines historiques dans la commande optimale
- ▶ La difficulté provient du fait que, avec les méthodes de différences temporelles "classiques", il faut faire un max sur les actions, ce qui est un problème d'optimisation dans le cas continu
- ▶ Les méthodes de gradient et acteur-critique atténuent le problème, au prix de ne fournir qu'un maximum local (méthodes de Pontryagin plutôt que de Bellman)

## Paramétrisation : approcher le continu



- ▶ Les fonctions continues ne peuvent pas être énumérées
- ▶ Solution générale : *features* paramétrées, régler un vecteur de paramètres
- ▶ Pour l'apprentissage par renforcement avec actions continues, deux approches :
  - ▶ Méthodes « acteur pur » : représentation paramétrée de la politique...
  - ▶ Méthodes « acteur-critique » : ... ET de la fonction de valeur

## Politiques paramétrées

- ▶ On paramètre des politiques stochastiques notées  $\pi_\theta(u|x)$  par un vecteur  $\theta$
- ▶ Note : on trouve aussi  $\pi(u|x; \theta)$
- ▶ On cherche le gradient d'une performance sur le long terme  $J(\pi_\theta)$
- ▶ On note  $\nabla_\theta = \frac{\partial J(\pi_\theta(u|x))}{\partial \theta}$
- ▶  $J$  s'exprime différemment selon qu'on considère une récompense moyenne, actualisée par un facteur  $\gamma$ , etc.

## Différences finies

- ▶ On veut maximiser  $J(\pi_\theta)$  par une descente (ou montée) de gradient
- ▶ Pour estimer  $J(\pi_\theta)$ , on utilise des échantillons de performance récoltés le long de trajectoires du système
- ▶ Approche naïve : fixer différentes valeurs de  $\theta$ , mesurer la performance, en déduire  $\nabla_\theta$  et faire varier  $\theta$  selon la pente
- ▶ Intérêt : aucun présupposé sur la forme de la politique, donc très général
- ▶ Inconvénient : très lent, très grande variance, donc besoin de beaucoup d'échantillons et réglage délicat

## REINFORCE

- ▶ Si on connaît la forme explicite de  $\pi_\theta(u|x)$ , on peut calculer analytiquement la dérivée par rapport à  $\theta$
- ▶ On peut alors utiliser cette dérivée pour estimer à moindre coût le gradient de performance le long de trajectoires
- ▶ C'est plus efficace, mais moins général

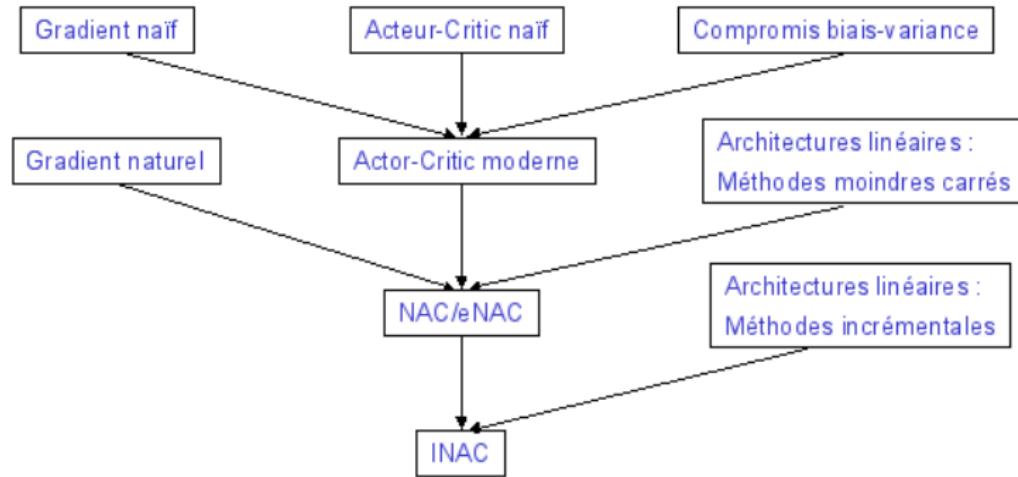
## D'acteur pur à acteur-critique

- ▶ Sachant que la fonction de valeur  $V^{\pi_\theta}$  (ou  $Q^{\pi_\theta}$ ) « résume » la performance de  $\pi_\theta$  en chaque état
- ▶ Peut-on utiliser des valeurs estimées de  $V^{\pi_\theta}$  ou  $Q^{\pi_\theta}$  pour calculer le gradient plus efficacement ?
- ▶ [Sutton et al., 2000] montre que oui : des méthodes qui utilisent la valeur atteignent la même qualité que REINFORCE avec moins d'échantillons



Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *NIPS 12*

## Articulation des contributions successives



## Aperçu

- ▶ Les méthodes acteur-critique continues stockent une approximation linéaire de la fonction avantage  $A^\pi(x, u) = Q^\pi(x, u) - V^\pi(x)$
- ▶ Cette fonction est utilisée pour calculer un gradient de performance
- ▶ Ce gradient est utilisé pour mettre à jour l'acteur = la représentation paramétrique de la politique.
- ▶ Pour que ça marche, il faut une condition de compatibilité entre les *features* du critique et celles de l'acteur
- ▶ [Peters & Schaal, 2008] : ça marche mieux avec le gradient naturel qu'avec le gradient naïf

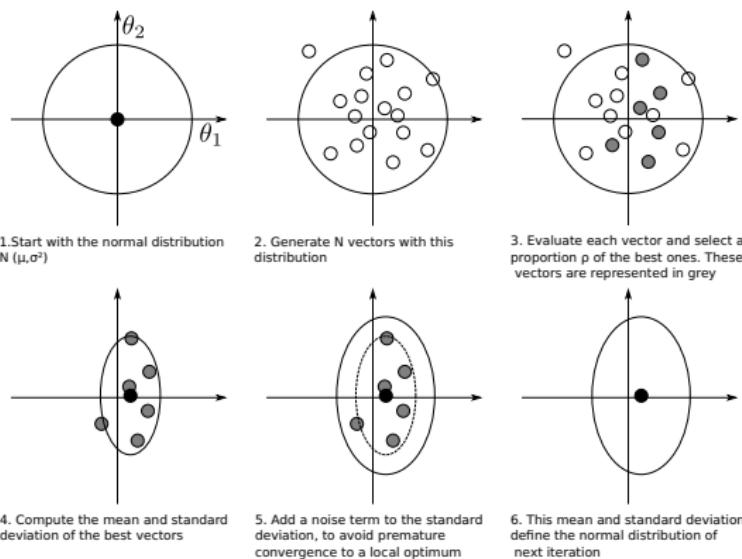


Peters, J. and Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural networks* 21(4) :682-697.

## Evolution récente

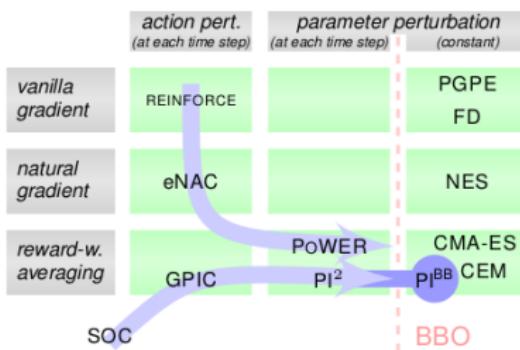
- ▶ La nécessité d'une approximation linéaire de  $A^\pi(x, u) = Q^\pi(x, u) - V^\pi(x)$  fait perdre en efficacité
- ▶ De plus, l'estimation du gradient est local et très dépendante d'un « pas d'apprentissage » qui la rend instable
- ▶ Les méthodes *EM-based* suppriment cette sensibilité au pas d'apprentissage
- ▶ Ces méthodes reformulent des problème d'apprentissage par renforcement en problèmes d'inférence probabiliste
- ▶ Permettent d'unifier l'apprentissage par démonstration (ou imitation) et l'apprentissage par renforcement
- ▶ Méthodes efficaces en apprentissage robotique
- ▶ Méthodes du moment : optimisation stochastique : PoWER,  $PI^2$ , CEPS, CMA-ES

## CEPS



Marin, and Sigaud, O. (2012) Towards fast and adaptive optimal control policies for robots : A direct policy search approach, *Proceedings conference Robotica*, pp. 21-26

## Sélection de l'action versus optimisation



- ▶ Convergence progressive des méthodes d'A/R vers l'optimisation stochastique
- ▶ CEM  $\approx$  CMA-ES  $\approx$  PI<sup>B</sup> > PoWER > eNAC
- ▶ Cette évolution est très liée au domaine d'application (apprentissage pour la robotique)
- ▶ Et à l'utilisation de « Dynamic Motor Primitives » (DMPs)
- ▶ La suite dans le cours suivant...



Stulp, F. and Sigaud, O. (2012) Path integral policy improvement with covariance matrix adaptation, *Proceedings ICML*



Stulp, F. and Sigaud, O. (submitted) Policy Improvement Methods : Between Black-Box Optimization and Episodic Reinforcement Learning, *Journal of Machine Learning Research*

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) :295–301.
-  Degris, T., Sigaud, O., & Willemin, P.-H. (2006).  
**Learning the Structure of Factored Markov Decision Processes in Reinforcement Learning Problems.**  
Édité 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.
-  Marin, D. & Sigaud, O. (2012).  
**Towards fast and adaptive optimal control policies for robots : A direct policy search approach.**  
Édité dans *Proceedings Robotica*, pages 21–26, Guimaraes, Portugal.
-  Moore, A. W. (1994).  
**The Parti-Game Algorithm for Variable Resolution Reinforcement Learning in Multidimensional State-Spaces.**  
Édité dans Hanson, S. J., Cowan, J. D., & Giles, C. L., éditeurs, *Advances in neural information processing systems*. Morgan Kaufmann, San Mateo, CA.
-  Peters, J. & Schaal, S. (2008).  
**Reinforcement learning of motor skills with policy gradients.**  
*Neural networks : the official journal of the International Neural Network Society*, 21(4) :682–97.
-  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.
-  **Stulp, F. & Sigaud, O. (2012).**  
**Path integral policy improvement with covariance matrix adaptation.**  
Édité dans *Proceedings of the 29th International Conference on Machine Learning (ICML'2012)*, page to appear, Edinburgh, Scotland.
-  **Stulp, F. & Sigaud, O. (submitted).**  
**Policy improvement methods : Between black-box optimization and episodic reinforcement learning.**  
*Journal of Machine Learning Research*, page to appear.
-  **Sutton, R. S. (1990a).**  
**Integrating architectures for learning, planning, and reacting based on approximating dynamic programming.**  
Édité 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.**  
Édité 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.
-  **Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000).**  
**Policy gradient methods for reinforcement learning with function approximation.**  
Édité dans *Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press.
-  **Watkins, C. J. C. H. (1989).**  
**Learning with Delayed Rewards.**  
Thèse de doctorat, Psychology Department, University of Cambridge, England.