Apprentissage par renforcement

Antoine Cornuéjols

AgroParisTech – INRA MIA Paris-Saclay

antoine.cornuejols@agroparistech.fr

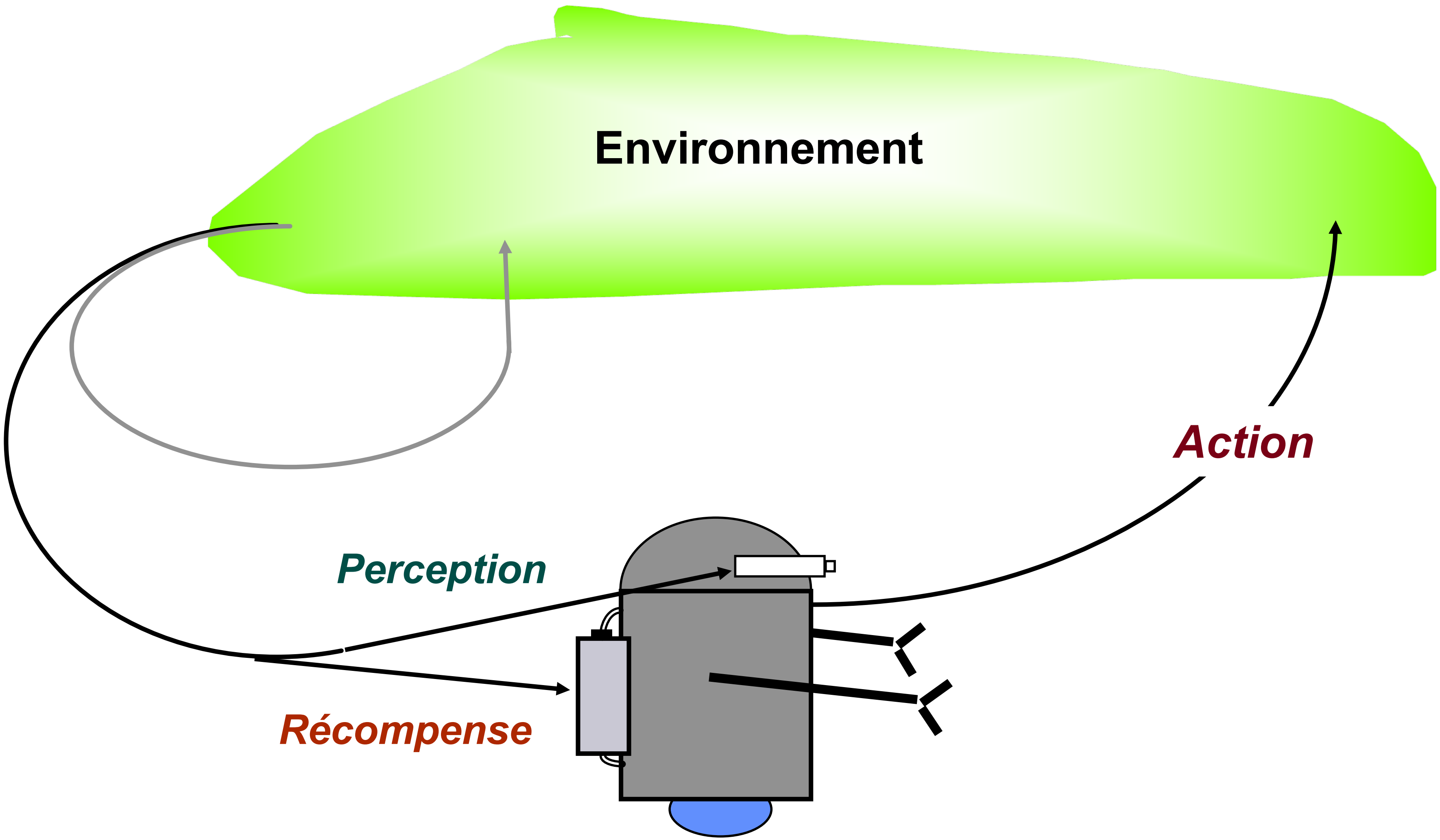
Plan du cours

1. Introduction : motivation, problèmes, notions et principes
2. Notions d'utilité et de politique
3. Apprentissage des fonctions d'utilité en **environnement connu** (et monde fini)
4. **Univers inconnu** (et monde fini)
   - Méthode de Monte-Carlo
   - Apprentissage par différences temporelles
   - SARSA
   - Méthode du Q-Learning
5. La **généralisation** dans l'apprentissage par renforcement
6. Exemples d'applications
7. Bilan et perspectives

Plan du cours

1. Introduction : motivation, problèmes, notions et principes
2. Notions d'utilité et de politique
3. Apprentissage des fonctions d'utilité en **environnement connu** (et monde fini)
4. **Univers inconnu** (et monde fini)
   - Méthode de Monte-Carlo
   - Apprentissage par différences temporelles
   - SARSA
   - Méthode du Q-Learning
5. La **généralisation** dans l'apprentissage par renforcement
6. Exemples d’applications
7. Bilan et perspectives

Introduction : schéma général



Comme dans MCTS … MAIS …

- Comme dans MCTS
  - **Choix itératif d'actions** à l'issue incertaine

- MAIS
  - Souvent, **pas de « fin de partie »**, donc pas d'exploration possible jusqu'aux feuilles
    - Un signal de **renforcement** de temps en temps
  - On ne joue **pas** contre **un adversaire** de type **MIN**. Donc pas de pire cas
    - On veut **maximiser une espérance de gain** dans un environnement mal connu (au début)

Introduction : Les **notations** de base

- *Temps* discret: t
- *États* : $s_t \in S$
- *Actions* : $a_t \in \mathcal{A}(s_t)$
- *Récompenses* : $r_t \in \mathcal{R}(s_t)$
- L'agent : $s_t \rightarrow a_t$
- L'environnement : $(s_t, a_t) \rightarrow s_{t+1}, r_{t+1}$
- *Politique* : $\pi_t : S \rightarrow \mathcal{A}$ ***T, R***
  - Avec $\pi_t(s,a)$ = Prob que $a_t = a$ si $s_t = s$
- Les transitions et récompenses ne dépendent **que** de **l'état** et de **l'action précédents** : processus Markovien

$\mathcal{R}^a_{ss'}$ **Récompense** reçue en passant de l'état s à l'état s' par l'action a

$\mathcal{P}^a_{ss'}$ **Probabilité** de passer de l'état s à l'état s' sous l'action a

Introduction : critères de gain

- ***Horizon fini***

$$\sum_{t=0}^{k} r_t = r_0 + r_1 + \ldots + r_k$$

- ***Horizon infini avec intérêt***

$$\sum_{t=0}^{\infty} \gamma^t r_t = r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ldots$$

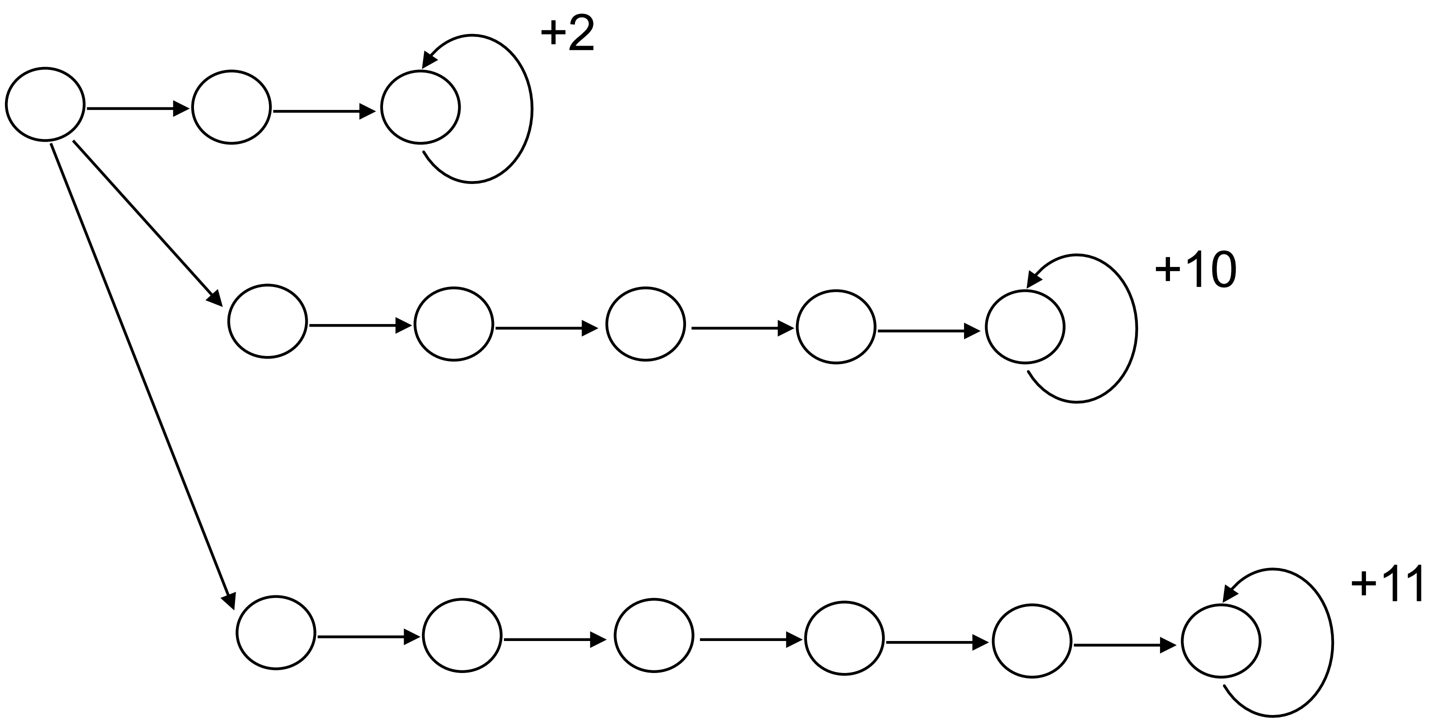
- ***En moyenne***

$$\lim_{k \to \infty} \frac{1}{k} \sum_{t=0}^{k} r_t$$

Comparaison des comportements induits

k=4, γ=0.9

Quelle est la meilleure stratégie ?



$\sum_{t=0}^{k} r_t$	$\sum_{t=0}^{\infty} \gamma^t r_t$	$\lim_{k\to\infty} \frac{1}{k} \sum_{t=0}^{k} r_t$
6	16.0	2
0	**59.0**	10
0	58.4	**11**

Plan du cours

1. Introduction : motivation, problèmes, notions et principes
2. Notions d'utilité et de politique
3. Apprentissage des fonctions d'utilité en **environnement connu** (et monde fini)
4. **Univers inconnu** (et monde fini)
   - Méthode de Monte-Carlo
   - Apprentissage par différences temporelles
   - SARSA
   - Méthode du Q-Learning
5. La **généralisation** dans l'apprentissage par renforcement
6. Exemples d'applications
7. Bilan et perspectives

*Deux **questions** essentielles*

1. Comment orienter les choix par **examen seulement** des successeurs directs ?
    - Donc **comment apprendre** une fonction d'évaluation ?

2. On cherche à maximiser une espérance de gain, donc face à une certaine distribution de probabilité correspondant à l'environnement et à ses réponses à mes choix
    - **Comment identifier** une politique optimale si l'espérance de gain que je dois évaluer **change** dès que je **modifie** ma politique ?

Remarque

- Cadre d'apprentissage à partir de très peu d'information
  - Va nécessairement impliquer de **très nombreuses** expériences
    - Souvent appel à des simulations avant de passer dans le monde réel

- Pour **simplifier le problème**, dans un premier temps on suppose
  - Un **espace d'états** discret
  - Des **transitions** dans le temps discrètes
  - Des **états** entièrement observables par l'agent

Introduction : Eléments de base

- *Politique* :

  ensemble d'associations *situation* → *action* (une application)

  Une simple table ... un algorithme de recherche intensive

  Eventuellement stochastique

- *Fonction de renforcement* :
  - Définit implicitement le <u>but poursuivi</u>
  - Une fonction : (*état*, *action*) → *récompense* $\in \Re$

- *Fonction d'évaluation* $V(s)$ *ou* $Q(s,a)$:
  - Récompense accumulée sur le <u>long-terme</u> (pas d'exploration en avant nécessaire)

- *Modèle de l'environnement* :
  - Fonctions *T* et *R* : (*état(t)*, *action*) → (*état(t+1)*, *récompense)*

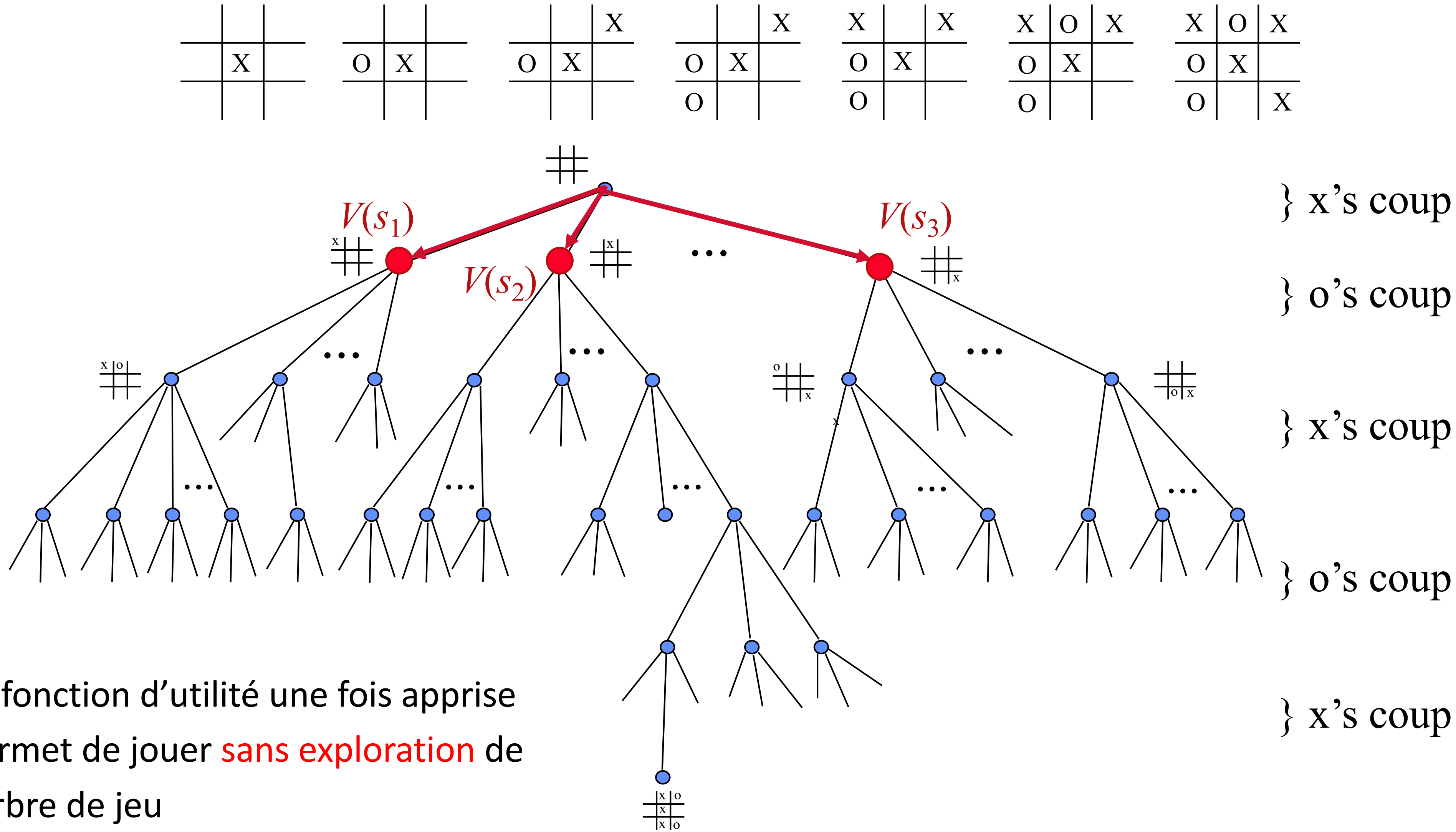
La notion d'utilité

Principe :

Choisir une action sans avoir besoin de faire une exploration (simulée) en avant

- Il faut donc disposer d'une **fonction d'évaluation locale** résumant une espérance de gain si l'on choisit cette action : fonction d'utilité
- Il faut **apprendre** cette fonction d'utilité : apprentissage par renforcement

Notion d'utilité. Exemple : Tic-Tac-Toe



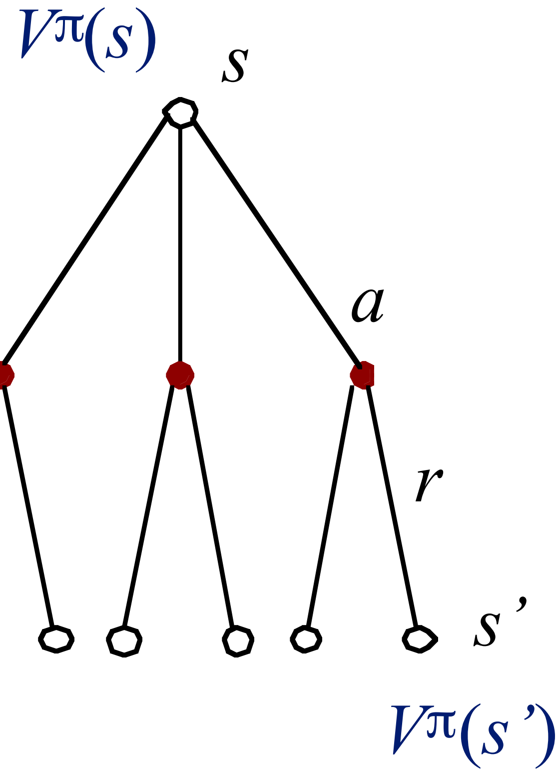
La fonction d'utilité une fois apprise permet de jouer sans exploration de l'arbre de jeu

Fonctions d'utilité : $V^{\pi}(s)$ et $Q^{\pi}(s,a)$

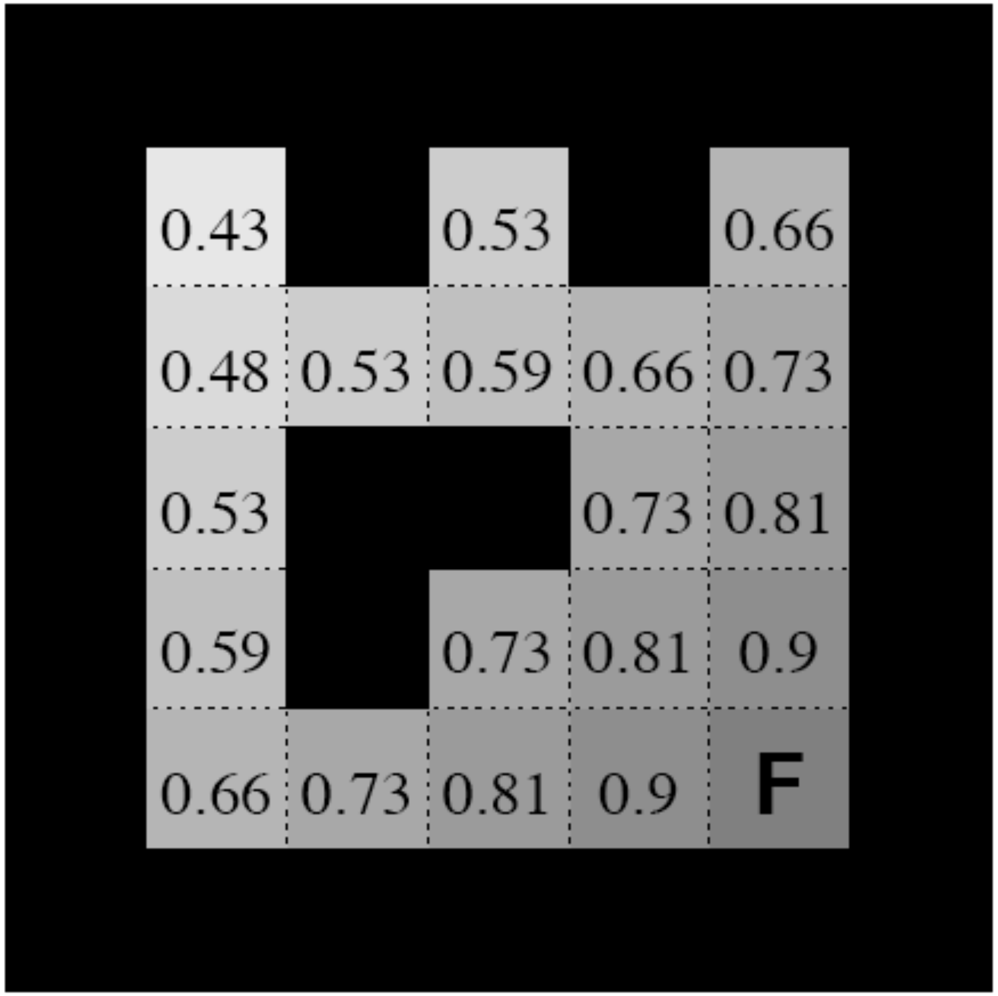
$$V^{\pi}(s) = E_{\pi}\left\{R_t \mid s_t = s\right\}$$
$$= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')\right]$$

$$Q^{\pi}(s,a) = E_{\pi}\left\{R_t \mid s_t = s, a_t = a\right\}$$
$$= \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')\right]$$

$$V^{\pi}(s) = \sum_a \pi(s,a)\, Q^{\pi}(s,a)$$



Notion d'utilité



Utilisation : avec la fonction d'utilité $V^*(s)$

- Une *politique* est une application $\pi \; : \; S \rightarrow A$

- Valeur optimale d'un état :

$$V^*(s) \;=\; \max_{\pi} V_{\pi}(s) \;=\; \max_{\pi} E_{\pi}\left(\sum_{t=0}^{\infty} \gamma^t \, r^t\right)$$

- La fonction de valeur optimale V^* est unique

$$V^\star(s) \;=\; \max_{a \in \mathcal{Z}} \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{s_t s'} + \gamma V^*(s')\right]$$

- Une politique stationnaire optimale existe :

$$\boxed{\pi^\star(s) \;=\; a^* \;=\; \operatorname*{ArgMax}_{a \in \mathcal{Z}} \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{s_t s'} + \gamma V^*(s')\right]}$$

Utilisation : avec la fonction d'utilité V*(s)

Attention

Pour appliquer la politique, il faut connaître :

$$\boxed{\pi^\star(s) \;=\; a^* \;=\; \underset{a \in \mathcal{Z}}{\mathrm{ArgMax}} \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{s_t s'} \;+\; \gamma V^*(s') \right]}$$

Utilisation : avec la fonction d'utilité $Q^*(s,a)$

- Fonction d'évaluation d'action $Q_\pi(s,a)$

- Valeur optimale d'une action (dans un état) :

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) = E\left\{r_{t+1} + \gamma V^*(s_{t+1}) \middle| s_t = s, a_t = a\right\}$$

$$Q^\star(s,a) = \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{s_t s'} + \gamma \max_{a' \in \mathcal{Z}} Q^*(s',a')\right]$$

- **Théorème :**

$$\boxed{\pi^\star(s) = \underset{a}{\text{ArgMax}}\ Q^\star(s,a)}$$

est une politique optimale

Utilisation : avec la fonction d'utilité Q(s,a)*

Ici :

Pour appliquer la politique, **rien à connaître** sur l'environnement !!!

$$\boxed{\pi^\star(s) \; = \; \underset{a}{\text{ArgMax}} \; Q^\star(s,a)}$$

Plan du cours

1. Introduction : motivation, problèmes, notions et principes
2. Notions d'utilité et de politique
3. Apprentissage des fonctions d'utilité en **environnement connu** (et monde fini)
4. **Univers inconnu** (et monde fini)
   - Méthode de Monte-Carlo
   - Apprentissage par différences temporelles
   - SARSA
   - Méthode du Q-Learning
5. La **généralisation** dans l'apprentissage par renforcement
6. Exemples d'applications
7. Bilan et perspectives

Quand le monde est **connu**

T(*s*,*a*,*s'*) et *r*(*s*)

$$\mathcal{P}^{a}_{ss'} \qquad \mathcal{R}^{a}_{ss'}$$

Algorithme d'évaluation de **politique**

Programmation dynamique : Évaluation de politique

Évaluation de politique : Pour une **politique** donnée π, calculer la fonction d'utilité d'état $V^{\pi}(s)$:

Rappel :

State - value function for policy π :

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

Bellman equation for V^{π} :

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'}\left[R^{a}_{ss'} + \gamma V^{\pi}(s')\right]$$

— a system of $|S|$ simultaneous linear equations

Évaluation itérative d'une politique

<u>Principe</u> : l'équation de point fixe de Bellman peut fournir

une ***procédure itérative d'approximation***

de la fonction d'utilité V^{π}

$$\boxed{V_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s') \right]}$$

$$V_0 \longrightarrow V_1 \longrightarrow \ldots \longrightarrow V_k \longrightarrow V_{k+1} \longrightarrow \ldots \longrightarrow V^{\pi}$$

une "propagation" : visite de tous les états et mise à jour des valeurs

*Algorithme d'**évaluation** itérative d'une politique*

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
    $\Delta \leftarrow 0$
    Loop for each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Algorithme d'itération de valeur pour Q(s,a)

Algorithme 1 *Value Iteration*

Répéter

  Pour tout $s \in S$ **Faire**

    Pour tout $a \in A$ **Faire**

      $q \leftarrow Q(s,a)$

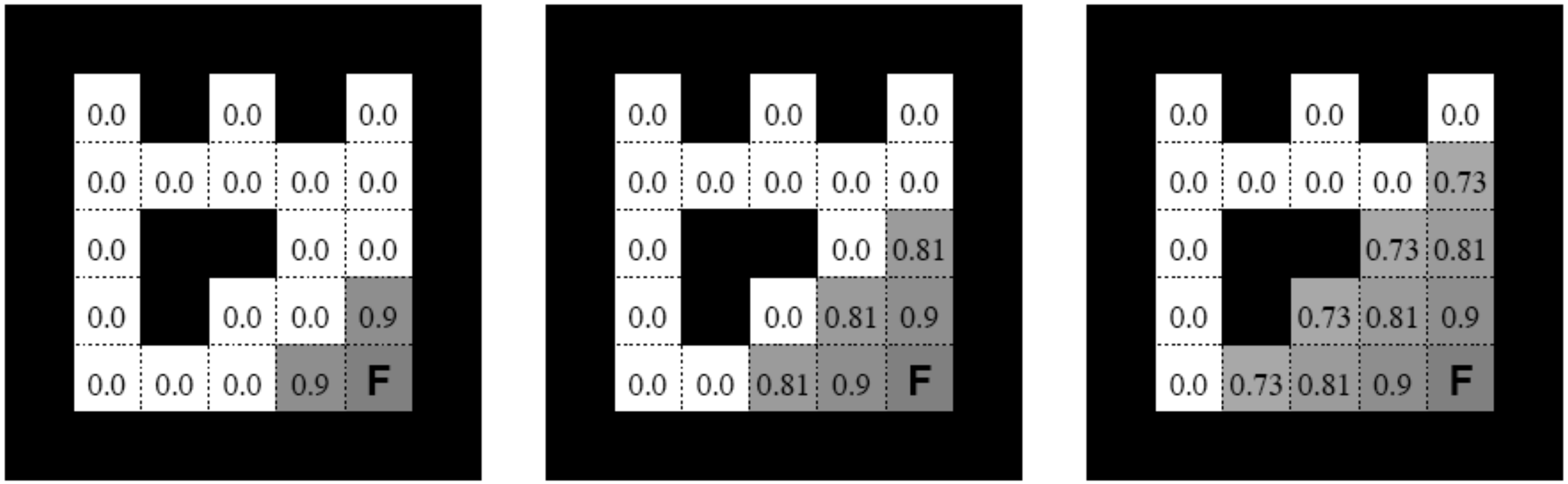
      $Q(s,a) \leftarrow R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') max_{a' \in A} Q(s',a')$

      $\Delta \leftarrow max(\Delta, |q - Q(s,a)|)$

    Fin Pour

  Fin Pour

Jusqu'à $\Delta < \epsilon$



Algorithme d’amélioration de **politique**

Comment améliorer une politique

Relation d'ordre sur les politiques :

Soient π et π' deux politiques déterministes, tq $\forall s \in \mathcal{S}$

$$Q^{\pi}(s, \pi'(s)) \;\geq\; V^{\pi}(s) \qquad (1)$$

Alors la politique π' est au moins aussi bonne que π :

$$V^{\pi'}(s) \;\geq\; V^{\pi}(s)$$

Comment améliorer une politique

Relation d'ordre sur les politiques :

Soient π et π' deux politiques déterministes, tq $\forall s \in \mathcal{S}$

$$Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s) \qquad (1)$$

Alors la politique π' est au moins aussi bonne que π :

$$V^{\pi'}(s) \geq V^{\pi}(s)$$

- Si l'on trouve une modification π' de la politique π vérifiant l'inégalité (1), alors on obtient une **meilleure politique**

Amélioration de politique

Supposons fait le calcul de V^{π} pour une politique déterministe π.

Pour un état donné s,
serait-il meilleur de faire l'action $a \neq \pi(s)$?

L'utilité de l'action a dans l'état s est :

$$Q^{\pi}(s,a) = E_{\pi}\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \middle| s_t = s, a_t = a\right\}$$
$$= \sum_{s'} P^{a}_{ss'}\left[R^{a}_{ss'} + \gamma V^{\pi}(s')\right]$$

Il est préférable de choisir l'action a dans l'état s si :

$$Q^{\pi}(s,a) \; > \; V^{\pi}(s)$$

Amélioration de politique (Cont.)

Il suffit de faire cela pour tous les états pour obtenir une nouvelle politique π' qui est gloutonne par rapport à V^{π} :

$$\begin{aligned} \pi'(s) &= \arg\max_{a} Q^{\pi}(s,a) \\ &= \arg\max_{a} \sum_{s'} P^{a}_{ss'}\left[R^{a}_{ss'} + \gamma V^{\pi}(s')\right] \end{aligned}$$

➢ Alors $V^{\pi'} \geq V^{\pi}$

Amélioration de politique (Cont.)

What if $V^{\pi'} = V^{\pi}$?

i.e., for all $s \in S$, $V^{\pi'}(s) = \max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^{\pi}(s') \right]$?

But this is the Bellman Optimality Equation.

So $V^{\pi'} = V^*$ and both π and π' are optimal policies.

P* is the only **fixed point** of the policy iteration process!!

Itération de politique

$$\pi_0 \xrightarrow{\mathbf{E}} V^{\pi_0} \xrightarrow{\mathbf{A}} \pi_1 \xrightarrow{\mathbf{E}} V^{\pi_1} \xrightarrow{\mathbf{A}} \ldots \longrightarrow \pi^* \xrightarrow{\mathbf{E}} V^* \xrightarrow{\mathbf{A}} \pi^*$$

Évaluation
de politique

Amélioration de politique
"gloutonne"

Algorithme d'itération de politique

Initialisation arbitraire de π

Faire

calcul de la fonction de valeur avec π

$$V_\pi(s) = R(s,\pi(s)) + \gamma \sum_{s' \in S} T(s,\pi(s),s')\, V_\pi(s')$$

Amélioration de la politique à chaque état

$$\pi'(s) \leftarrow \max_a \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')\, V_\pi(s') \right)$$

$\pi' := \pi$

Jusqu'à ce qu'aucune amélioration ne soit possible

- **Garantie de convergence** vers une politique optimale

Policy Iteration

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\quad \Delta \leftarrow 0$
   $\quad$ Loop for each $s \in \mathcal{S}$:
   $\qquad v \leftarrow V(s)$
   $\qquad V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
   $\qquad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad$ *old-action* $\leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
   $\quad$ If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Algorithme d'itération de **valeur**

Combine évaluation de politique et amélioration de politique

Policy iteration

- Drawbacks

Each time a policy is modified,
a phase of **iterative policy evaluation** is triggered, which can be costly

- **Do we have to wait until convergence** of this evaluation before modifying again the policy?
  - Answer : NO

In *Value Iteration*, policy evaluation is stopped after just one *sweep* (visiting all states)

- The **convergence** of policy iteration is **still guaranteed**

*Value iteration: for estimating π**

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

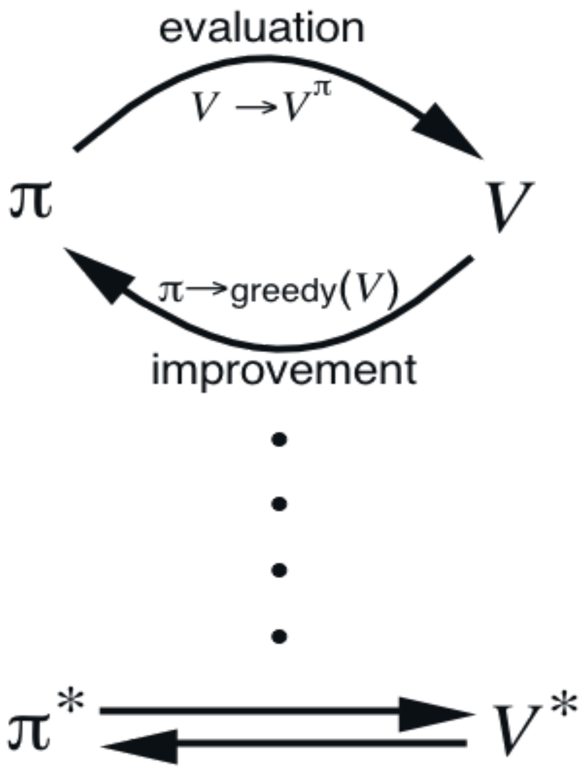
- Utilise l'équation de Bellman sur la politique optimale

$$\boxed{V^\star(s) \;=\; \max_{a \in \mathcal{Z}} \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{s_t s'} + \gamma V^*(s')\right]}$$

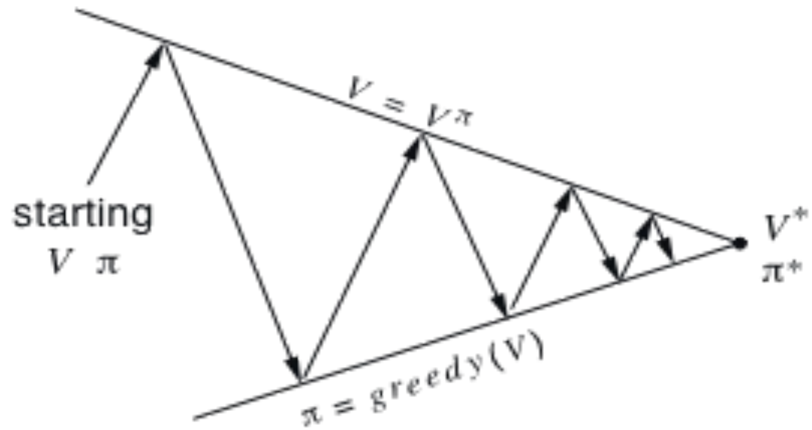
Itération généralisée de politique

Generalized Policy Iteration (GPI):

Toute interaction d'**étape d'évaluation** de politique et d'**étape d'amélioration** de politique indépendamment de leur granularité :



Métaphore géométrique pour la convergence de GPI :



Plan du cours

1. Introduction : motivation, problèmes, notions et principes
2. Notions d'utilité et de politique
3. Apprentissage des fonctions d'utilité en **environnement connu** (et monde fini)
4. Univers **inconnu** (et monde fini)
   - Méthode de Monte-Carlo
   - Apprentissage par différences temporelles
   - SARSA
   - Méthode du Q-Learning
5. La **généralisation** dans l'apprentissage par renforcement
6. Exemples d'applications
7. Bilan et perspectives

Comment faire ?

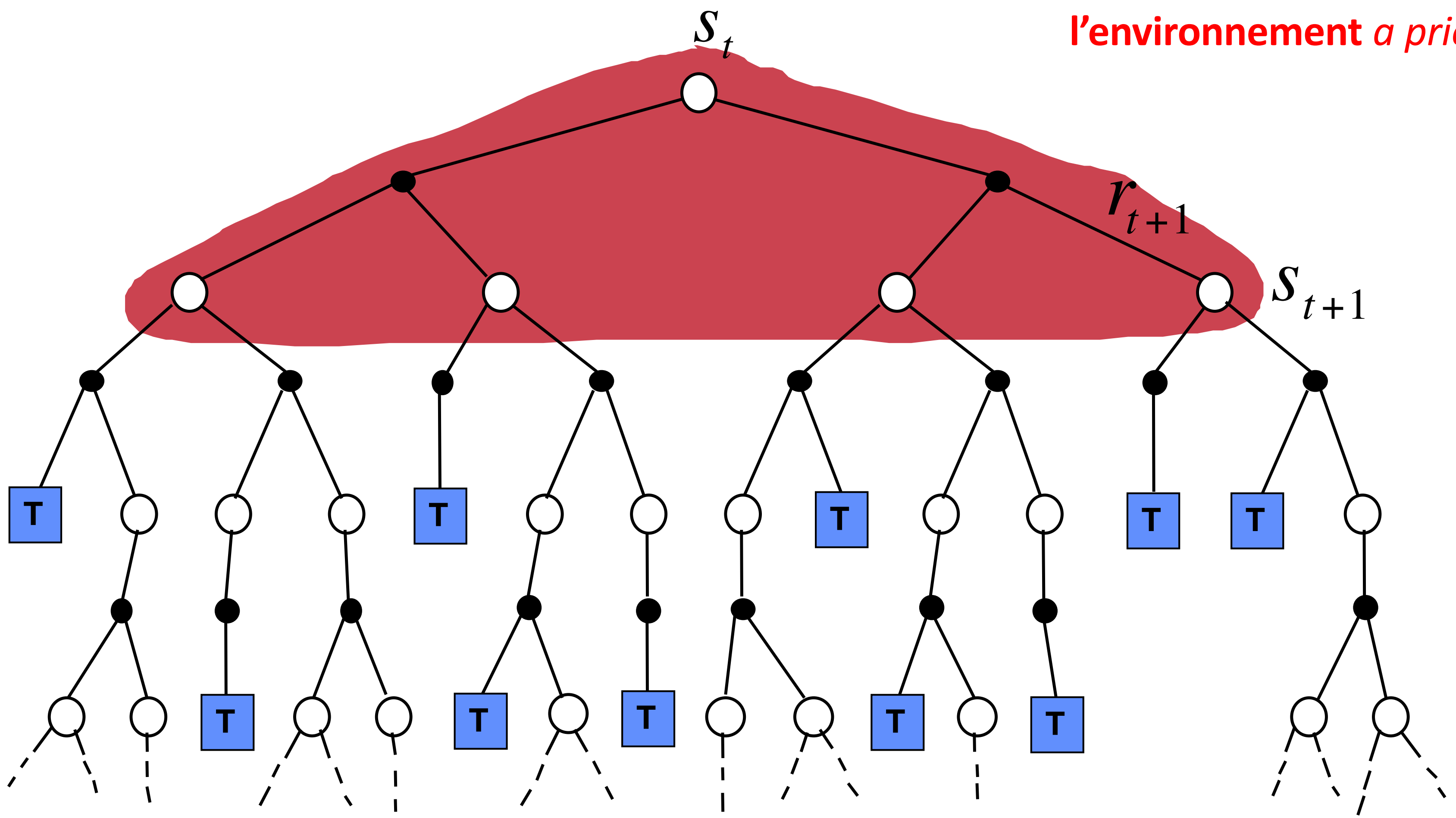
Apprentissage par renforcement

Le modèle du monde est **inconnu**

TD learning : cf. Dynamic Programming

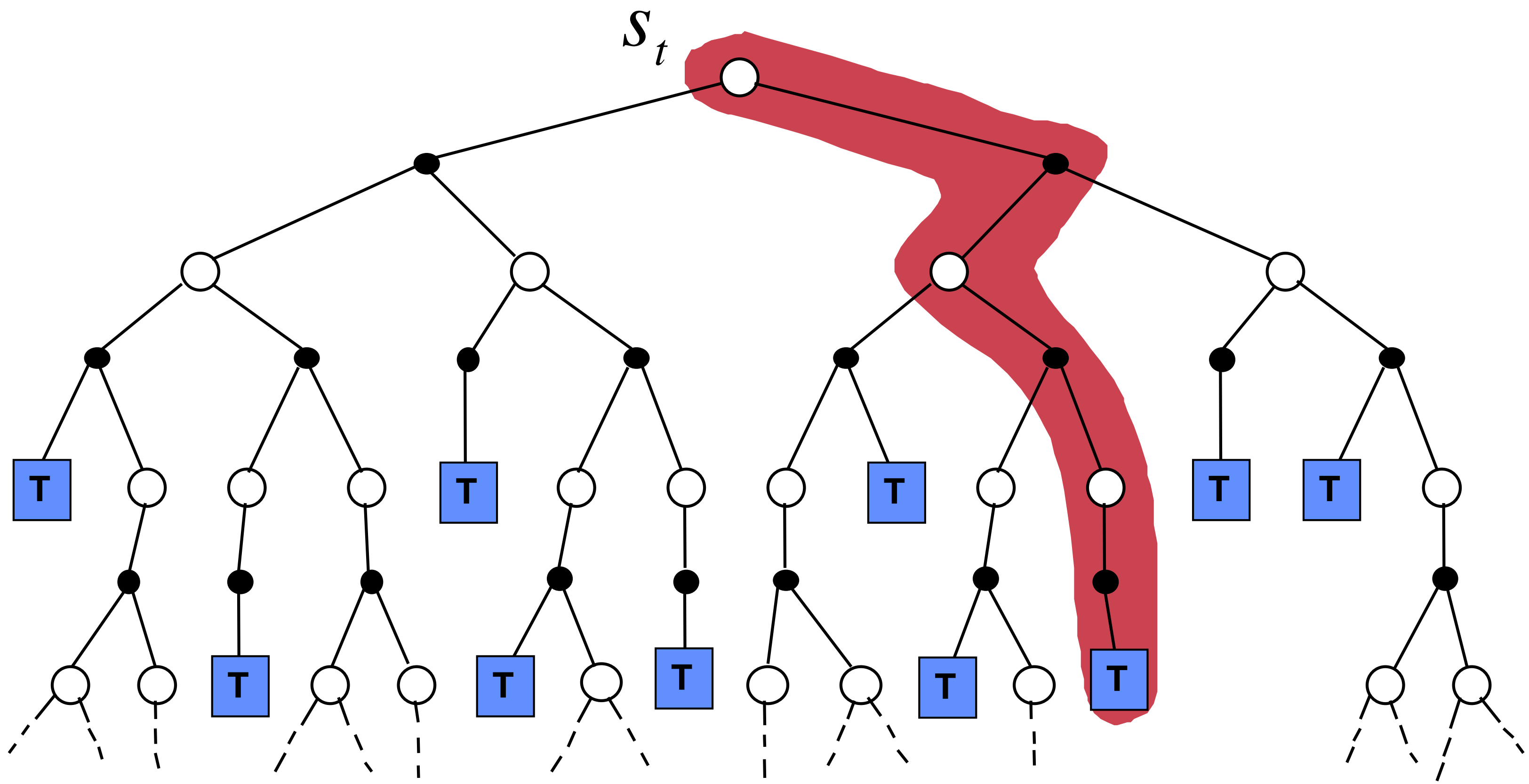
$$V(s_t) \leftarrow E_\pi \left\{ r_{t+1} + \gamma \; V(s_t) \right\}$$

On calcule l'espérance.
MAIS il **faut connaître l'environnement** *a priori*.



Monte Carlo method

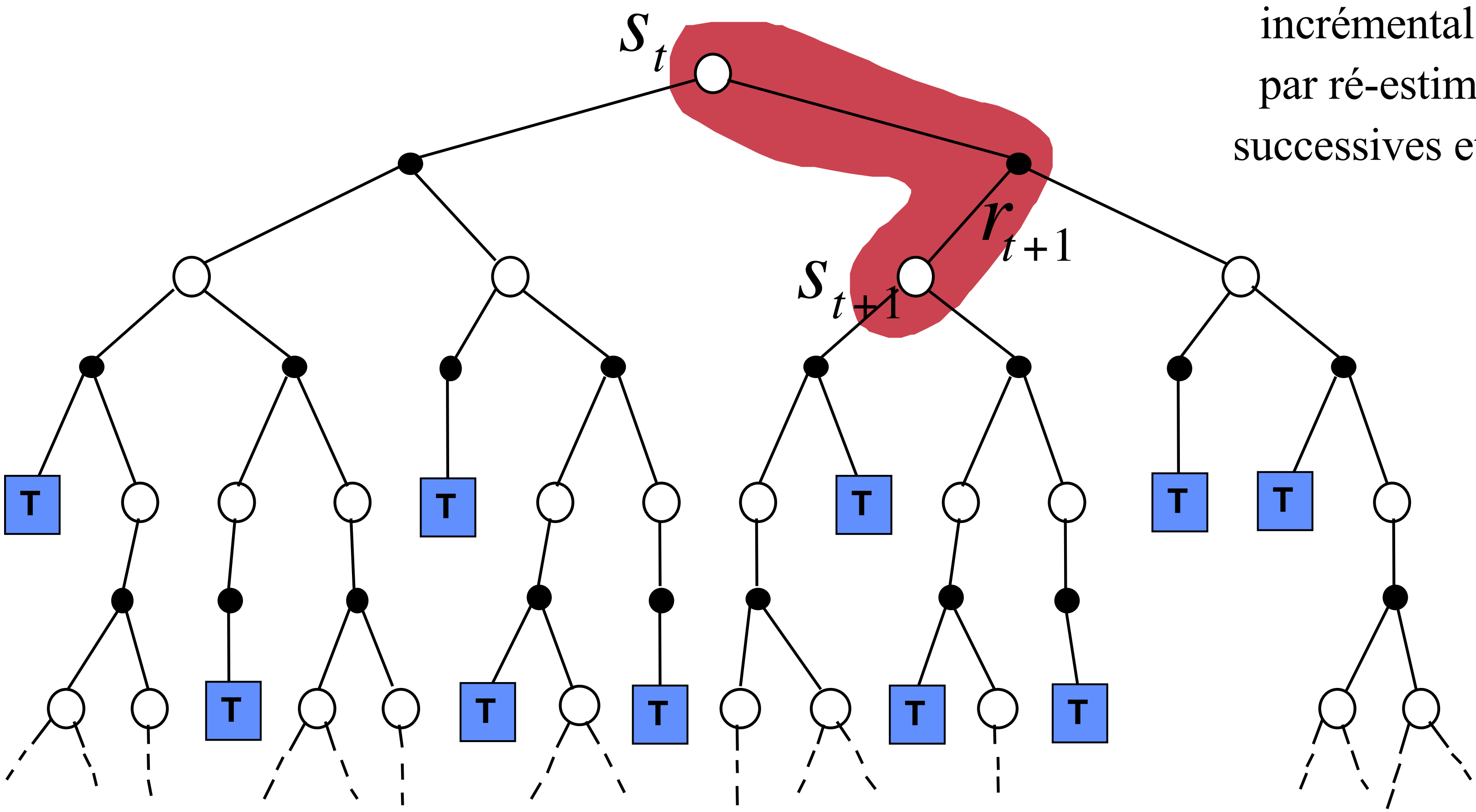
$$\underbrace{V(S_t)}_{\text{New value of state t}} \leftarrow \underbrace{V(S_t)}_{\text{Former estimation of value of state t (= Expected return starting at that state)}} + \underbrace{\alpha}_{\text{Learning Rate}}[\underbrace{G_t}_{\text{Return at timestep t}} - \underbrace{V(S_t)}_{\text{Former estimation of value of state t (= Expected return starting at that state)}}]$$



TD learning : Simplest Temporal Difference Method

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

On met à jour incrémentalement par ré-estimations successives et locales



TD learning : évaluation par méthode des différences temporelles

Évaluation de politique :

pour une politique donnée π, calculer la fonction d'utilité V^{π}

Simple every - visit $\boxed{\text{Monte Carlo method}}$:

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

cible: le **vrai** gain sur une durée t

The simplest $\boxed{\text{TD method, TD(0)}}$:

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

cible: une **estimation** du gain

Monte Carlo method

Algorithm parameter: small $\varepsilon > 0$

$\pi(s|a) > 0, \ \ \forall s \in \mathcal{S}, a \in \mathcal{A}$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$\boxed{G \leftarrow \gamma G + R_{t+1}}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

Modifies π

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

exploration

ε-greedy

TD learning : algo d'évaluation par différences temporelles

Initialisation :

$\pi \leftarrow$ politique à évaluer

$V \leftarrow$ une fonction arbitraire d'évaluation

Répéter (pour chaque pas de l'épisode) :

$a \leftarrow$ action préconisée par π pour s

Faire a; recevoir r; voir état suivant s'

$V(s) \leftarrow V(s) + \alpha\ [r + \gamma V(s') - V(s)]$

$s \leftarrow s'$

Jusqu'à s terminal

Le principe des différences temporelles

Soit la méthode d'estimation par moyennage :

La moyenne des premiers k renforcements est (en ignorant la dépendance sur a):

$$X_k = \frac{r_1 + r_2 + \ldots + r_k}{k}$$

Peut-on faire le même calcul incrémentalement ?

Oui :

$$X_{k+1} = X_k + \frac{1}{k+1}\left[r_{k+1} - X_k\right]$$

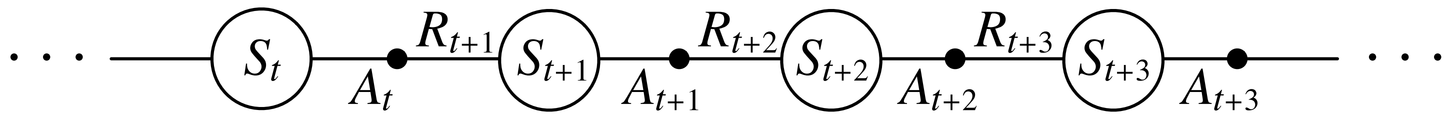
Règle classique d'amélioration :

NouvelleEstimation = *AncienneEstimation* + *Pas*[***Cible*** – *AncienneEstimation*]

TD learning : Learning An Action-Value Function Q(s,a)

Algorithme SARSA

Estimer Q^π pour la politique courante π



Pour chaque transition à partir d'un noeud non terminal *s*, mettre à jour :

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[\mathcal{R}_{ss'}^{a} + \gamma \, Q(s', a') - Q(s,a) \right]$$

Si *s'* est terminal, alors : $Q(s', a') = 0$

Action effectivement choisie dans l'état atteint *s'*

TD learning : Learning An Action-Value Function Q(s,a)

Algorithme SARSA

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize S
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Loop for each step of episode:
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ε-greedy)
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

Algorithme « **sur politique** » : on s'appuie sur la politique courante (choix de a') pour améliorer

L'apprentissage Q (Q-learning)

- Idée [Watkins,89] : Estimer les valeurs Q "en-ligne", en trouvant à la fois la politique et la fonction d'évaluation d'action :

$$\boxed{Q(s,a) \leftarrow Q(s,a) + \alpha \left[\mathcal{R}^{a}_{ss'} + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]}$$

MAJ avec l'action a de **valeur Q maximale** dans S_t.

Algorithme « **hors politique** » : on ne s'appuie pas sur la politique courante pour améliorer

L'apprentissage Q (Q-learning)

- Idée [Watkins,89] : Estimer les valeurs Q "en-ligne", en trouvant à la fois la politique et la fonction d'évaluation d'action :

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

MAJ avec l'action a de **valeur Q maximale** dans S_t.

- Théorème :

Si chaque action est exécutée un nombre infini de fois dans chaque état, les valeurs Q calculées convergent vers Q^*, conduisant à une politique optimale.

TD learning : Q-Learning

Algorithme SARSA

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[\mathcal{R}^a_{ss'} + \gamma \, Q(s', a') - Q(s,a) \right]$$

Q-learning

$$\boxed{Q(s,a) \leftarrow Q(s,a) + \alpha \left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q(s', a') - Q(s,a) \right]}$$

Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

Q-learning est « hors politique » (off policy algorithm)

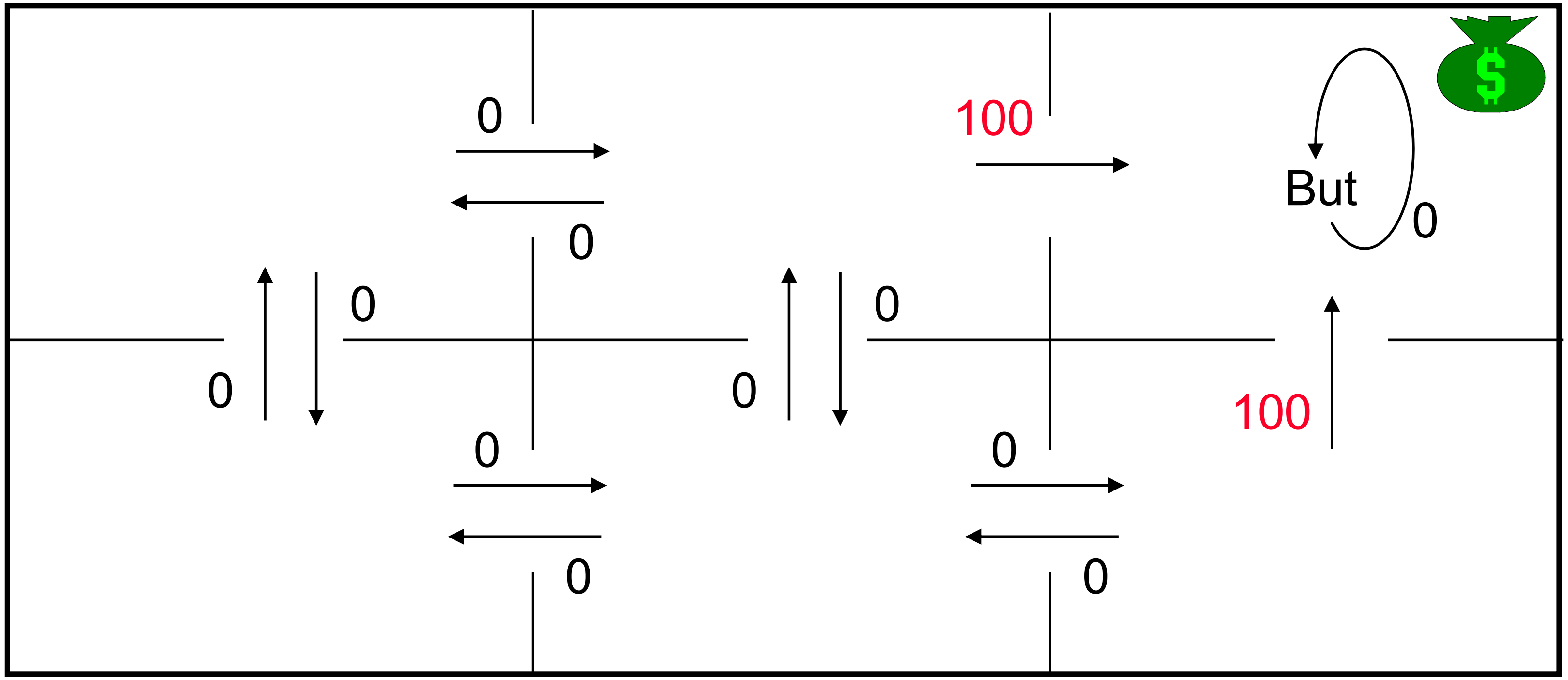
- Algorithme « **sur politique** » (on-policy)
  - La **même politique** est utilisée pour agir et mettre à jour les valeurs

- Algorithme « **hors politique** » (off-policy)
  - La politique pour agir (e.g. choisie par ε-greedy)
  - Est **différente** de la politique pour mettre à jour les valeurs (e.g. l'action maximisant Q(s',a') dans le Q-learning)

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

Exemple (1/4)

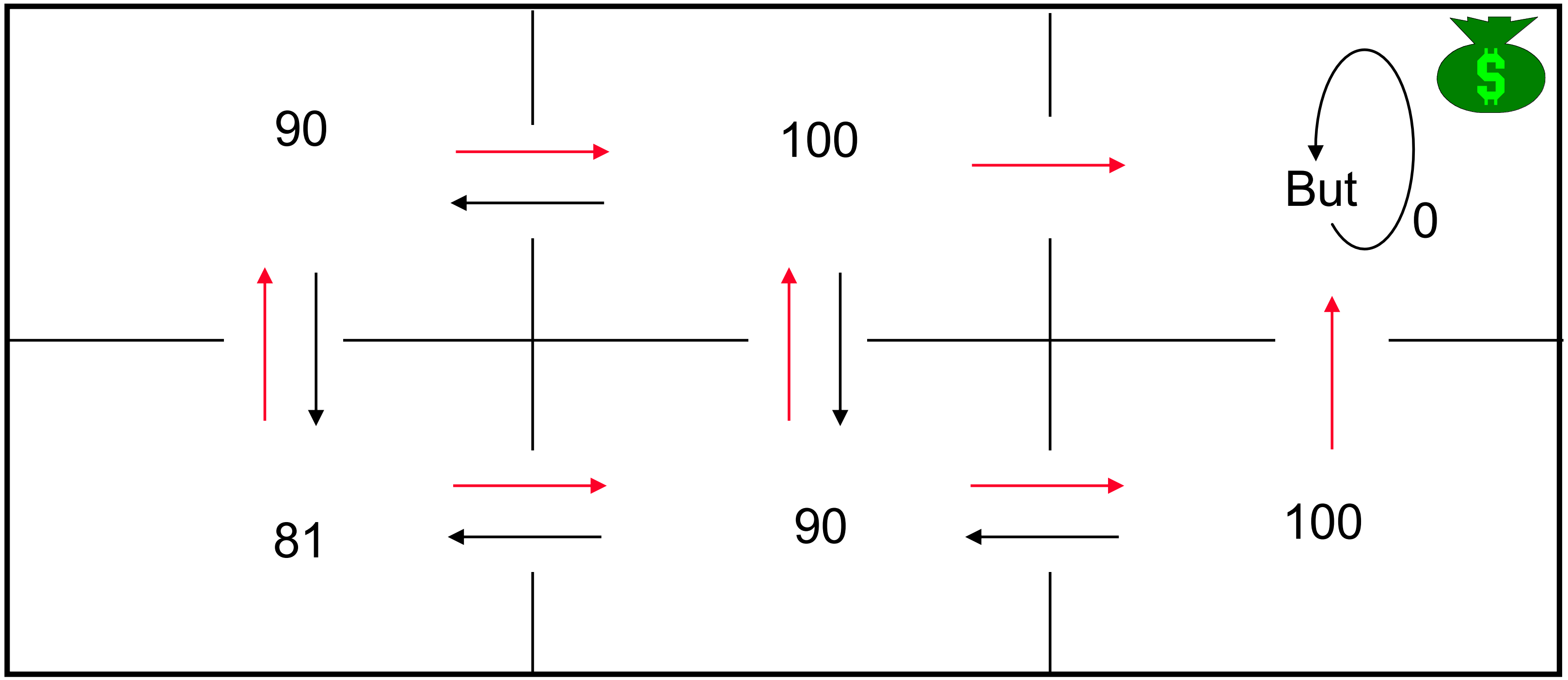
r(s,a) récompense immédiate



- Rq: La dernière étape assure la récompense (jeux, monde des blocs, etc.)
- <u>Tâche</u>: apprendre la meilleure stratégie

Exemple (2/4)

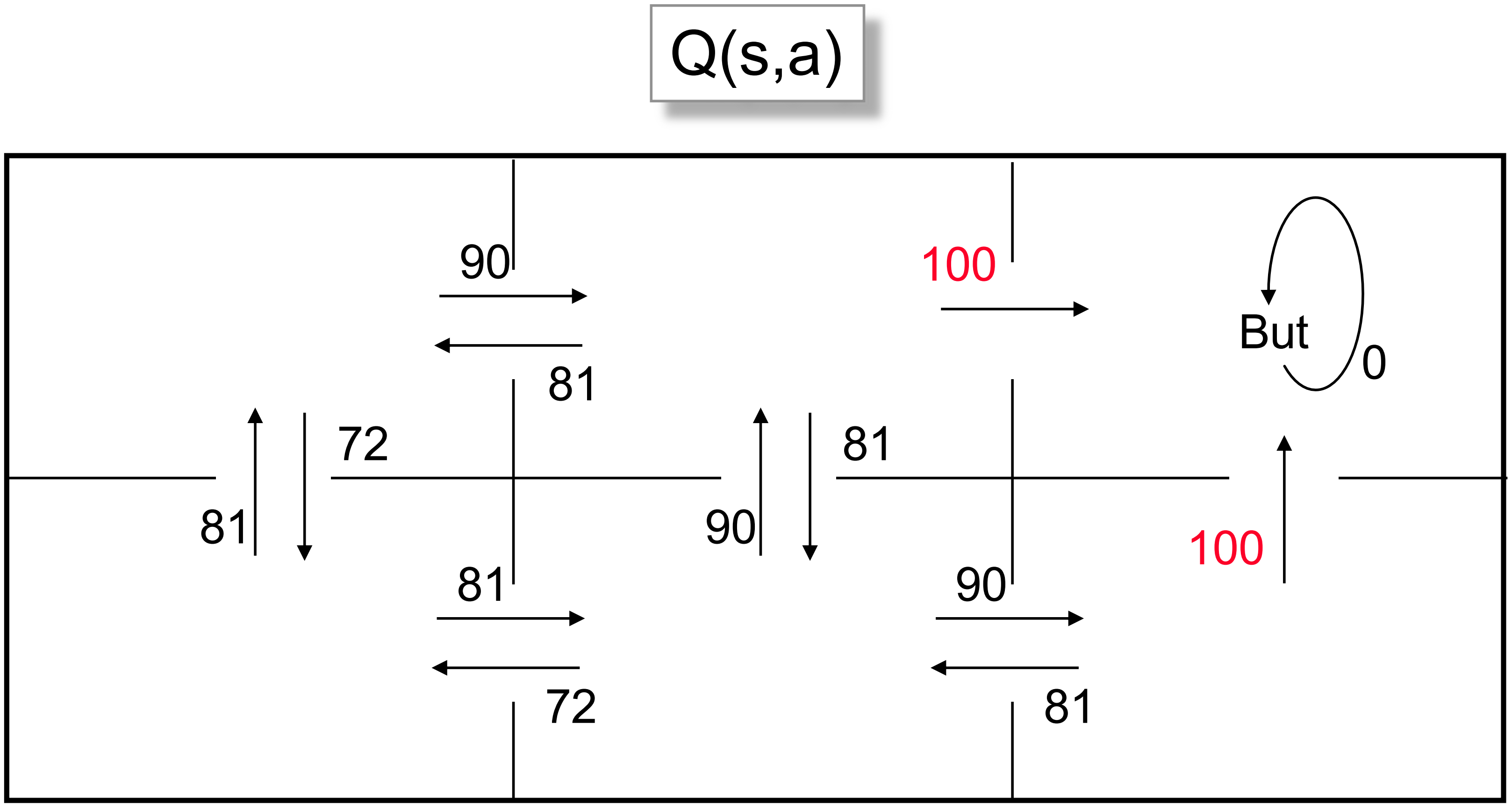
- On définit la récompense cumulée $V^{\pi}(s_t) = \sum_{t=0}^{\infty} \gamma^t r_t$
- <u>Le problème</u>: trouver $\pi^* = \underset{\pi}{\operatorname{argmax}}(V^{\pi}(s))$



V*(s)=Vπ*(s) récompense cumulée optimale

Exemple *(3/4)*

- La fonction Q est définie comme étant LA fonction qui résume en UN nombre toute l'info nécessaire sur le gain cumulé d'une action a, prise dans l'état s.



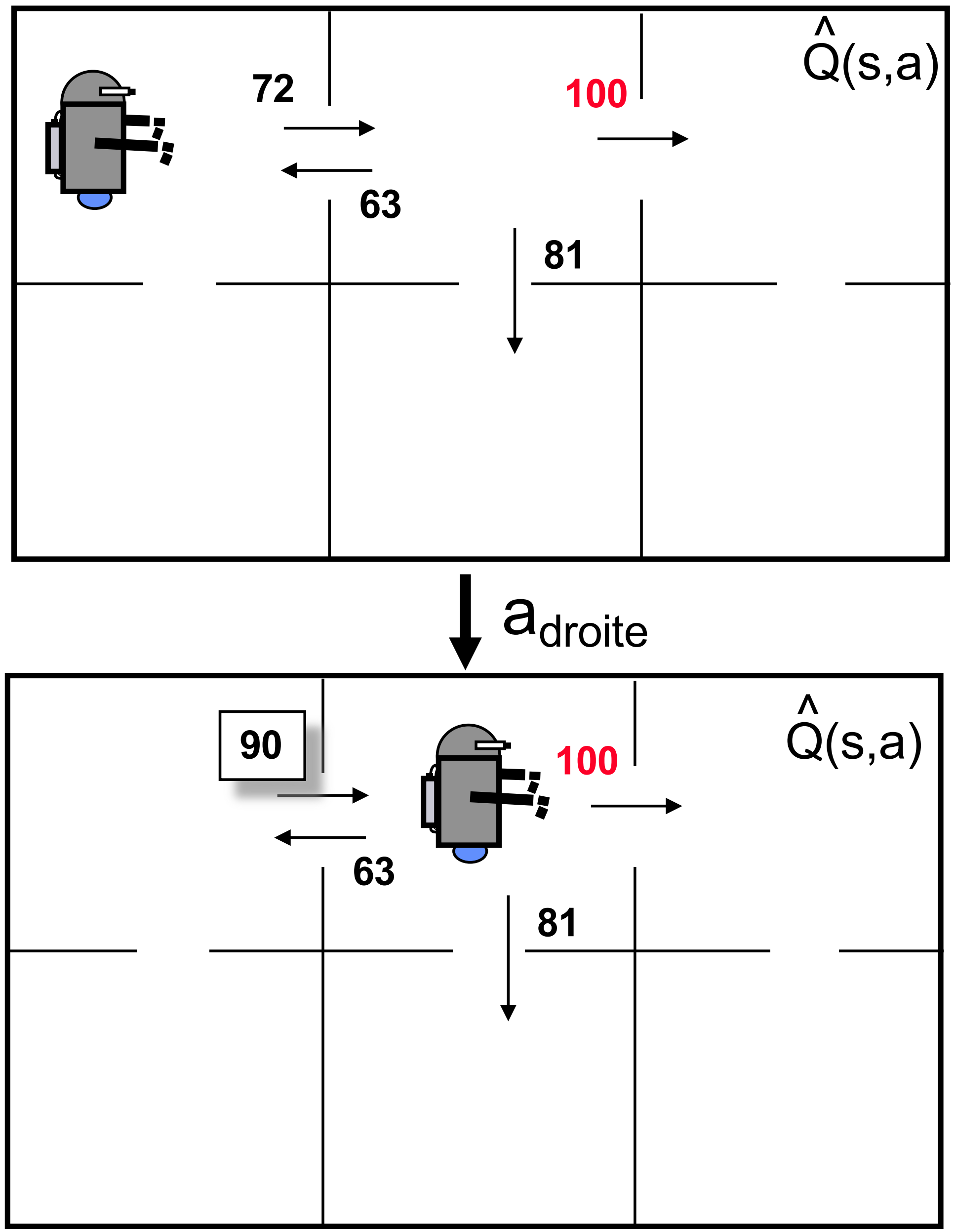
Exemple *(4/4)*

On Prend α = 1

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(\delta(s,a),a'))$$

$$\leftarrow 0 + 0.9 \max\{63,\ 81,\ 100\}\}$$

$$\leftarrow 90$$



Plan du cours

1. Introduction : motivation, problèmes, notions et principes
2. Notions d'utilité et de politique
3. Apprentissage des fonctions d'utilité en **environnement connu** (et monde fini)
4. **Univers inconnu** (et monde fini)
   - Méthode de Monte-Carlo
   - Apprentissage par différences temporelles
   - SARSA
   - Méthode du Q-Learning
5. La **généralisation** dans l'apprentissage par renforcement
6. Exemples d'applications
7. Bilan et perspectives

La généralisation dans l' apprentissage par renforcement

Apprentissage avec *généralisation*

- Si l'espace S (ou S x A) est trop important pour l'utilisation d'une table mémorisant les prédictions

- Deux options :

  - Utilisation d'une technique de généralisation dans l'espace S ou l'espace S x A (e.g. réseau de neurones, ...)

  - Utilisation d'une technique de regroupement d'états en classes d'équivalence (même prédiction et même action générée).

Généralisation : Approximation de la fonction V(s)

Comme avant : Évaluation de politique :

pour une politique donnée π, calculer la fonction d'utilité V^{π}

Mais avant, les fonctions d'utilité étaient stockées dans des tables.

Maintenant, l'estimation de la fonction d'utilité au temps t, V_t, dépend d'un vecteur de paramètres $\vec{\theta}_t$, et seul ce vecteur de paramètres est mis à jour.

e.g., $\vec{\theta}_t$ pourrait être le vecteur de poids de connexions d'un réseau de neurones.

Généralisation : Backups as Training Examples

e.g., the TD(0) backup :

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

As a training example:

$$\left\{\text{description of } \underbrace{s_t}_{\text{input}},\ \underbrace{r_{t+1} + \gamma\, V(s_{t+1})}_{\text{target output}}\right\}$$

Généralisation : n'importe quelle méthode inductive ?

- En principe, oui :
  - Réseaux de neurones artificiels
  - Arbres de décision
  - Méthodes de régression multivariées
  - etc.
- Mais l'App. par R. a des exigences particulières :
  - Apprendre tout en agissant
  - S'adapter à des mondes non stationnaires

Plan du cours

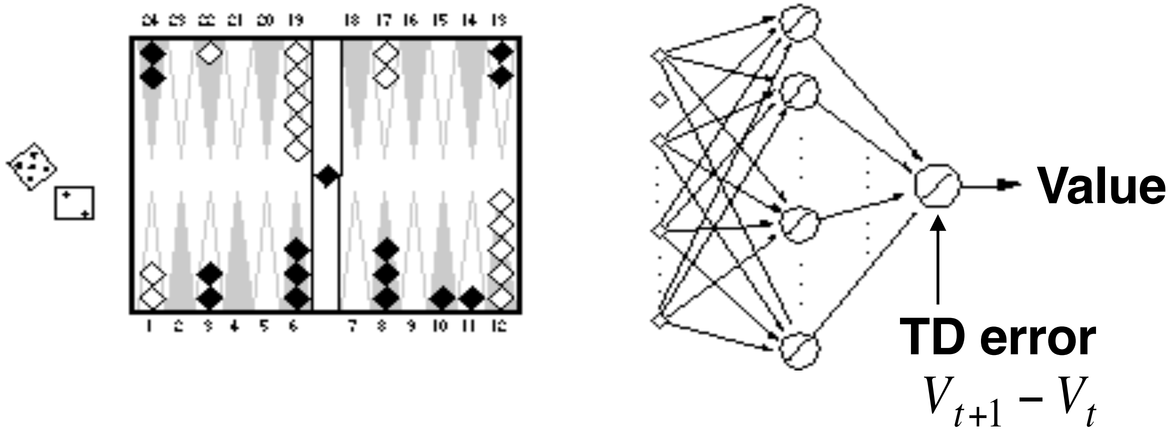
1. Introduction : motivation, problèmes, notions et principes
2. Notions d'utilité et de politique
3. Apprentissage des fonctions d'utilité en **environnement connu** (et monde fini)
4. **Univers inconnu** (et monde fini)
   - Méthode de Monte-Carlo
   - Apprentissage par différences temporelles
   - SARSA
   - Méthode du Q-Learning
5. La **généralisation** dans l'apprentissage par renforcement
6. Exemples d'applications
7. Bilan et perspectives

Some Notable RL Applications

- TD-Gammon: Tesauro
  - world's best backgammon program
- Elevator Control: Crites & Barto
  - high performance down-peak elevator controller
- Inventory Management: Van Roy, Bertsekas, Lee&Tsitsiklis
  - 10–15% improvement over industry standard methods
- Dynamic Channel Assignment: Singh & Bertsekas, Nie & Haykin
  - high performance assignment of radio channels to mobile telephone calls

TD-Gammon

Tesauro, 1992–1995



Action selection by 2–3 ply search

Start with a random network

Play very many games against self

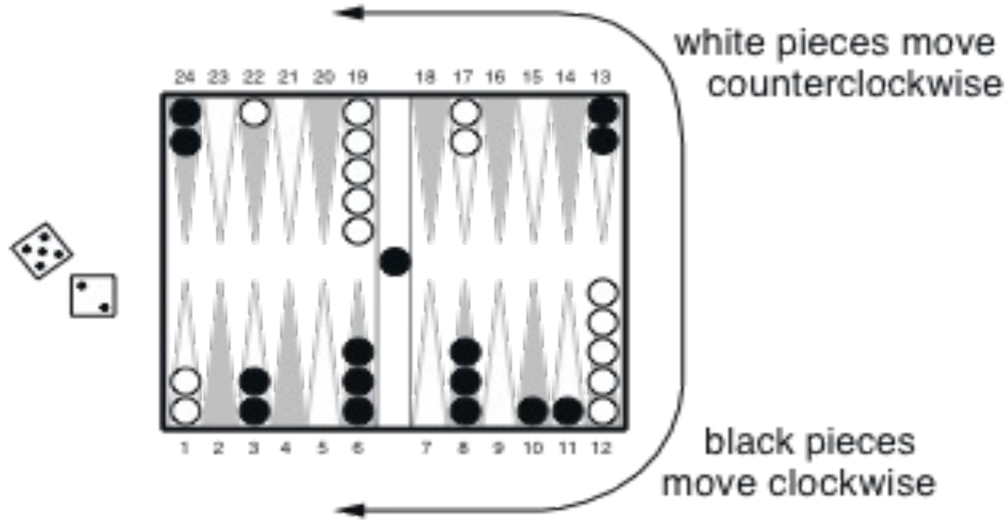
Learn a value function from this simulated experience

This produces arguably the best player in the world

Realizations : TD Gammon

Tesauro 1992, 1994, 1995, ...

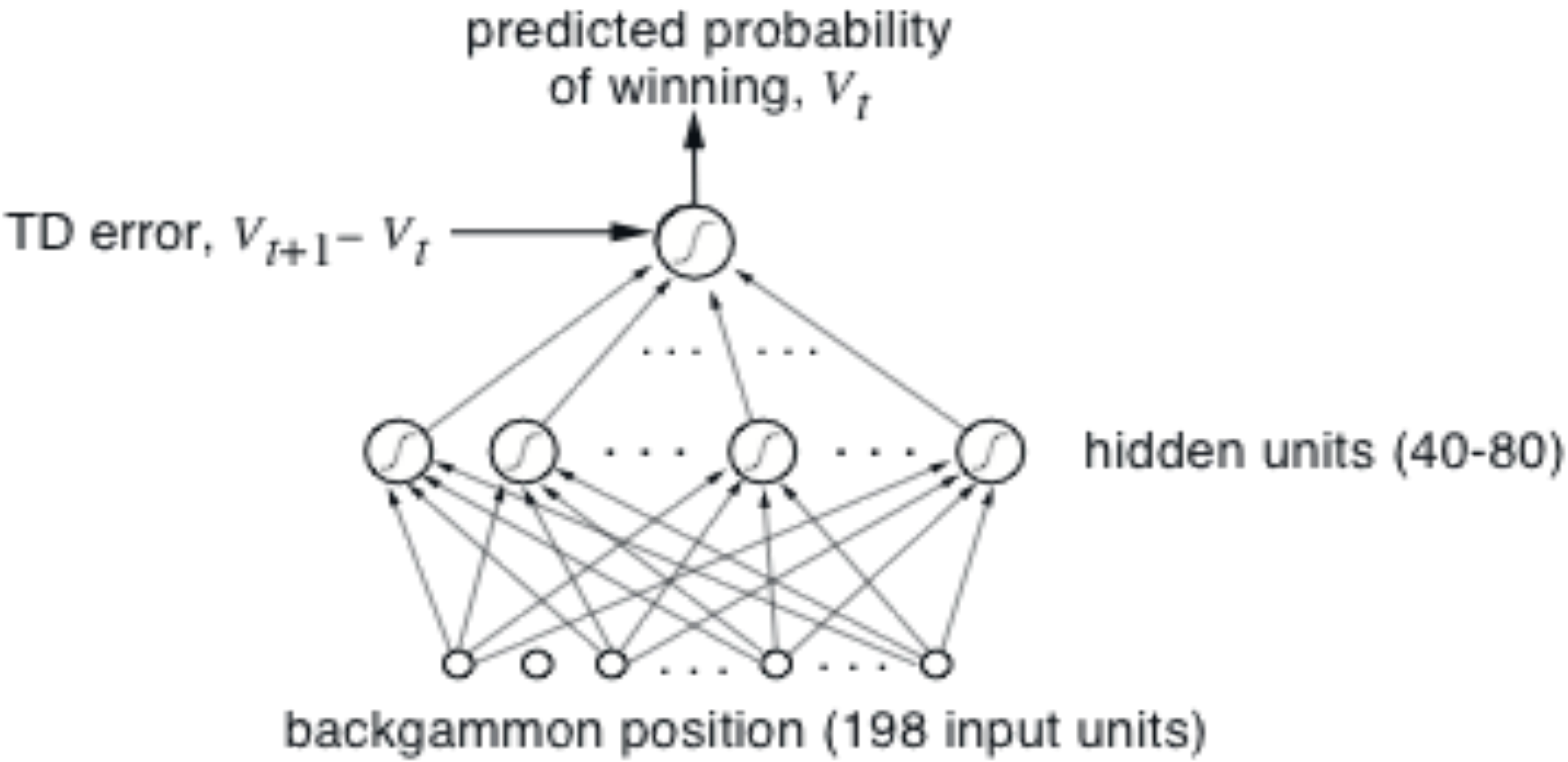
- White has just rolled a 5 and a 2 so can move one of his pieces 5 and one (possibly the same) 2 steps
- Objective is to advance all pieces to points 19-24
- Hitting
- Doubling
- 30 pieces, 24 locations implies enormous number of configurations
- Effective branching factor of 400



Realizations: A Few Details

- Reward: 0 at all times except those in which the game is won, when it is 1
- Episodic (game = episode), undiscounted
- Gradient descent TD(λ) with a multi-layer neural network
  - weights initialized to small random numbers
  - backpropagation of TD error
  - four input units for each point; unary encoding of number of white pieces, plus other features
- Use of afterstates
- Learning during self-play

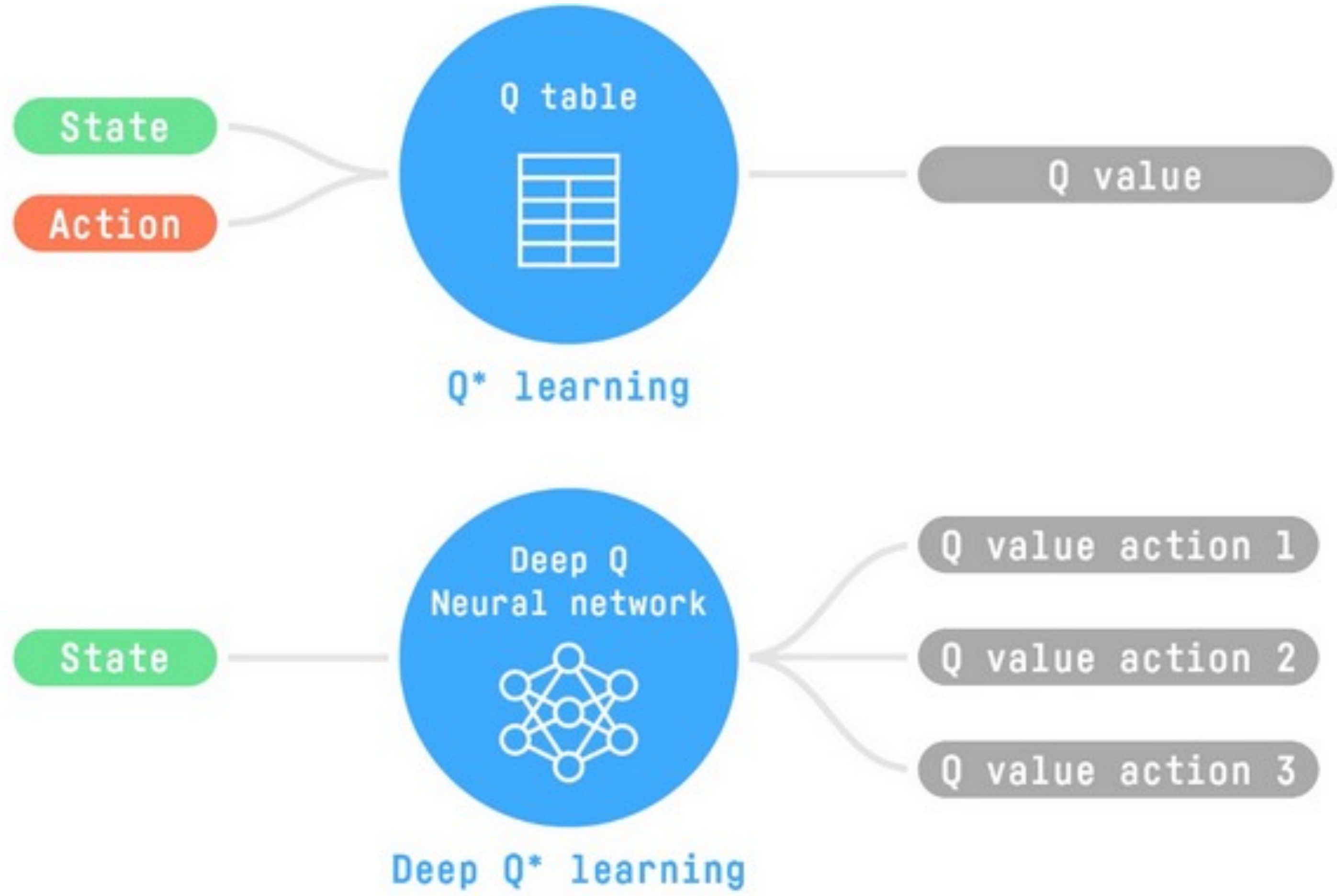
Realizations: Multi-layer Neural Network



Realizations: Summary of TD-Gammon Results

Program	Hidden Units	Training Games	Opponents	Results
TD-Gam 0.0	40	300,000	other programs	tied for best
TD-Gam 1.0	80	300,000	Robertie, Magriel, . . .	−13 points / 51 games
TD-Gam 2.0	40	800,000	various Grandmasters	−7 points / 38 games
TD-Gam 2.1	80	1,500,000	Robertie	−1 point / 40 games
TD-Gam 3.0	80	1,500,000	Kazaros	+6 points / 20 games

Deep Reinforcement Learning based on the Q value



https://huggingface.co/blog/deep-rl-intro

Deep Reinforcement Learning

- Les valeurs Q*(s,a) obéissent aux **équations de Bellman**

$$Q^{\star}(s,a) \;=\; \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^{\star}(s',a') \,|s,a\right]$$

- Traiter la valeur $r + \gamma \max_{a'} Q(s',a',\mathbf{w})$ comme une cible

- **Minimiser** la perte MSE (Minimum Square Error) par gradient stochastique

$$\ell(\mathbf{w}) \;=\; \left(r + \gamma \max_{a'} Q(s',a',\mathbf{w}) \,-\, Q(s,a,\mathbf{w})\right)^2$$

➢ Converge vers Q* **si monde fini** (représentation tabulaire)

➢ Mais diverge (!) si Q est réalisé par un réseau de neurones, car :

1. **Corrélation** entre les exemples
2. Et cible **non stationnaire**

Deep Q-network (DQN): Experience replay

...

To remove correlations, build data-set from agent's own experience

s_1, a_1, r_2, s_2
s_2, a_2, r_3, s_3
s_3, a_3, r_4, s_4
$\ldots$
$s_t, a_t, r_{t+1}, s_{t+1}$

$s_t, a_t, r_{t+1}, s_{t+1} \rightarrow$ (into the data-set) $\rightarrow s, a, r, s'$

Et rejouer des expériences passées pour éviter **l'oubli catastrophique**

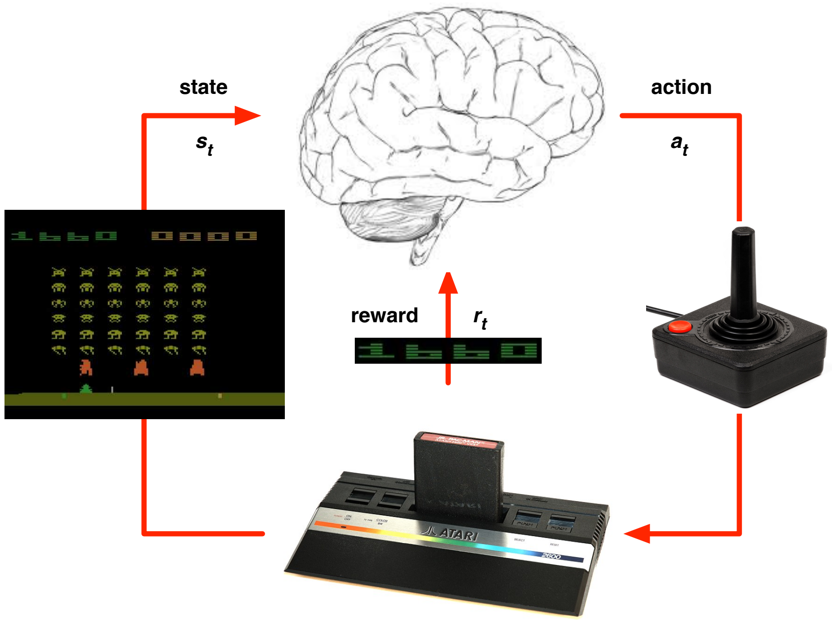
Sample experiences from data-set and apply update

$$l = \left(r + \gamma \max_{a'} Q(s', a', \mathbf{w}^-) - Q(s, a, \mathbf{w}) \right)^2$$

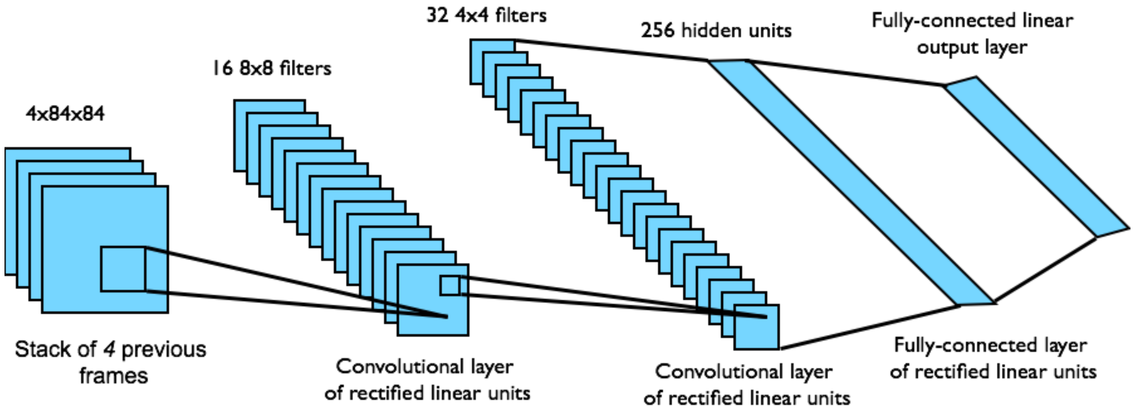
To deal with non-stationarity, target parameters $\mathbf{w}^-$ are held fixed

Deep RL in Atari

…



- End-to-end learning of values $Q(s, a)$ from pixels s
- Input state s is stack of raw pixels from last 4 frames
- Output is $Q(s, a)$ for 18 joystick/button positions
- Reward is change in score for that step



Network architecture and hyperparameters fixed across all games

Deep RL in Atari

…

DQN paper
www.nature.com/articles/nature14236

DQN source code:
sites.google.com/a/deepmind.com/dqn/

Deep RL in Go

…

AlphaGo paper:
www.nature.com/articles/nature16961

AlphaGo resources:
deepmind.com/alphago/

Bilan : trois idées principales

1. Le passage par des fonctions d'utilité

2. La rétro-propagation de ces valeurs le long de trajectoires réelles ou simulées

3. Itération généralisée de politique :

    1. Calculer continuellement une estimation de la **fonction d'utilité** optimale et

    2. Chercher **une politique** optimale grâce à cette estimation, qui, en retour, s'adapte en conséquence

Les grandes approches

- RL basé sur l'apprentissage de valeurs d'utilité
  - E.g. apprendre Q*(s,a)

- RL basé sur l'apprentissage de politique
  - Explorer directement l'espace des politiques
  - E.g. par algorithme génétique

- RL basé sur l'apprentissage d'un modèle du monde
  - Construire un modèle de l'environnement
  - Construire un plan en utilisant ce modèle

Sources documentaires

- Ouvrages / articles
  - Sutton & Barto (2018) : *Reinforcement Learning: an introduction*. MIT Press, 2018.
  - Plaat, Aske (2022) : *Deep Reinforcement Learning*. Springer.
  - Kaelbling, Littman & Moore (96) : *Reinforcement learning : A survey*. Journal of Artificial Intelligence Research, 4:237-285.

- Sites web
  - http://http://www-anw.cs.umass.edu/~rich/RL-FAQ.html

(FAQ maintenue par Rich Sutton et point dentrée pour de nombreux sites)