Apprentissage par renforcement

Laëtitia Matignon

LIRIS

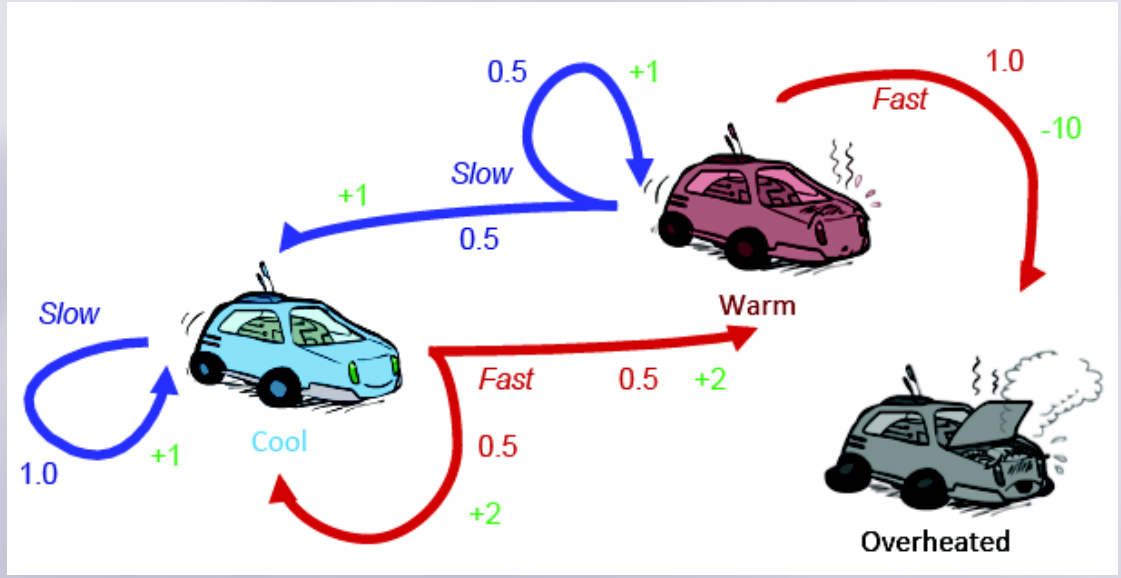
Lyon 1

Plan

Rappels : Planification (*offline* MDP)

Planification

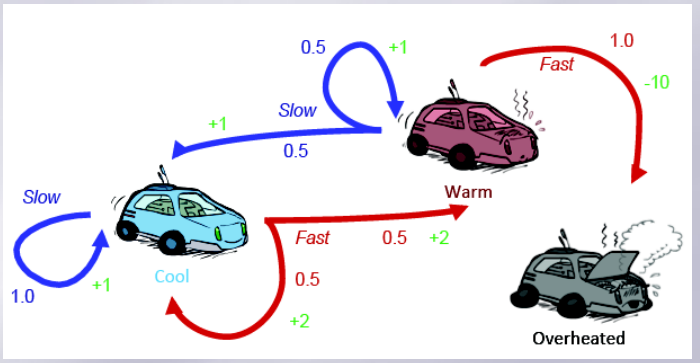
- Modèle MDP $< S, A, T, R >$ complètement connu
  - ensemble fini d'états : S
  - ensemble fini d'actions : A ou $A(s)$
  - fonction de transition $T : S \times A \times S \to [0; 1]$
  - fonction de renforcement $R : S \times A \times S \to \mathbb{R}$

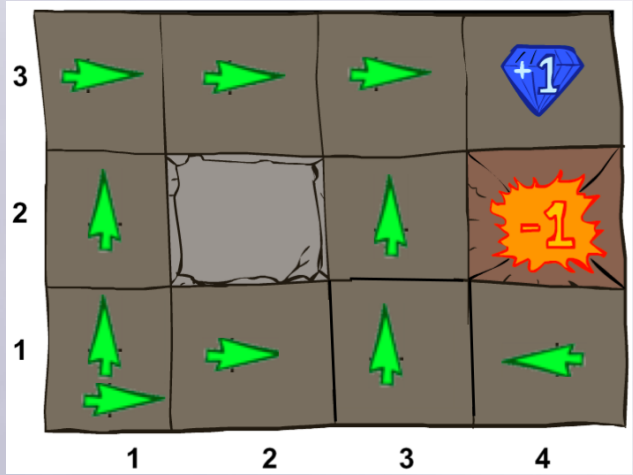


Rappels : Planification (*offline* MDP)

Planification

- Modèle MDP $< S, A, T, R >$ **complètement connu**
- Solution au MDP : une politique π





Exemple d'une politique $\pi : S \rightarrow A$

Rappels : Planification (*offline* MDP)

Planification

- Modèle MDP $< S, A, T, R >$ complètement connu
- Solution au MDP : une politique π
- Objectif : trouver une politique optimale π^*

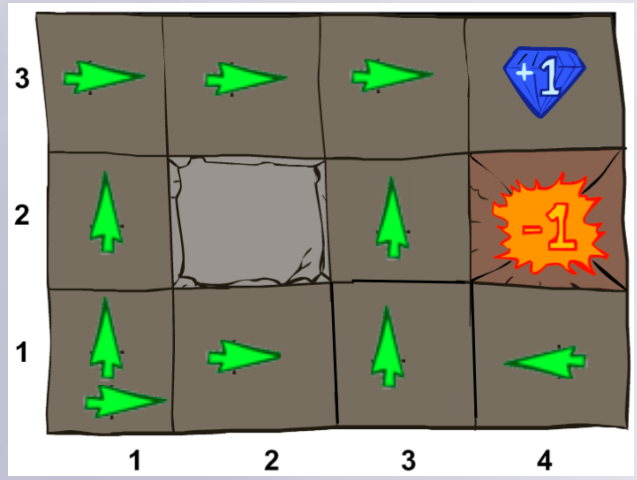
π^* donne pour tout état, la ou les action(s) permettant de maximiser les récompenses que l'on espère obtenir à travers la séquence d'états futurs.
π^* maximise le critère γ pondéré :
$G_t = r_1 + \gamma r_2 + \gamma^2 r_3 + ... = \sum_{t=0}^{\infty} \gamma^t r_{t+1}$

Rappels : Planification (*offline* MDP)

Planification

- Objectif : trouver une politique optimale π^*
- Moyen : fonction de valeur $V : S \rightarrow \Re$

La valeur $V^\pi(s)$ d'un état s est la **somme des récompenses espérées** si on suit π depuis s : $V^\pi(s) = E\{G^\gamma|\pi\} = E\{\sum_{t=0}^{\infty} \gamma^t r_{t+1}|\pi, s_0 = s\}$

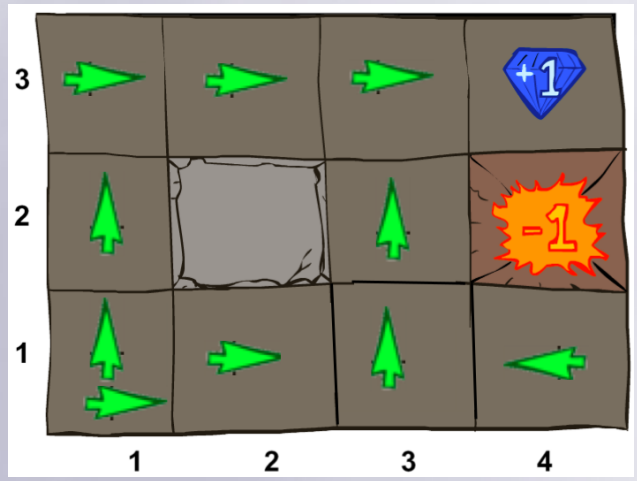


Fonction de valeur V associée à la politique π dans un environnement déterministe

Rappels : Planification (*offline* MDP)

Planification

- Objectif : trouver une politique optimale π^*
- Moyen : fonction de valeur $V : S \rightarrow \Re$
- Algorithme *value iteration* : calcul itératif de V^* :
  $\forall s \in S \; V_{k+1}(s) = \max_{a \in A} \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V_k^{\pi}(s')]$

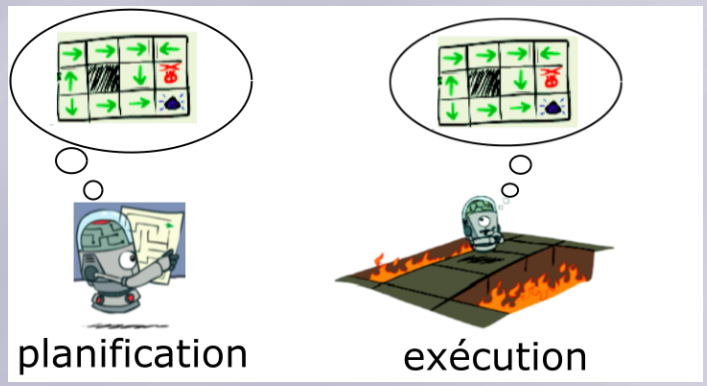


Fonction de valeur V associée à la politique π dans un environnement déterministe

Rappels : Planification (*offline* MDP)

Planification **hors-ligne**

- Objectif : trouver une politique optimale π^*
- Moyen : fonction de valeur $V : S \rightarrow \Re$
- Algorithme *value iteration* : calcul itératif de V^*
- Problème de planification dans l'incertain :
  - Phase **hors ligne** : calcul itératif de V^* **sans être placé dans l'environnement** (*value iteration*)
  - Puis phase d'exécution : exécution de la politique **dans l'environnement**



Plan

Apprentissage en-ligne

Apprentissage en-ligne

- Modèle MDP $< S, A, T, R >$
- Objectif : trouver la politique optimale π^*
- Nouvelle hypothèse : T et R inconnus de l'agent
  - L'agent ne connait pas à l'avance les effets de ses actions
  - L'agent ne connait pas à l'avance les actions et/ou états récompensés

$\rightarrow$ Impossible de planifier !

MDP partiellement connu

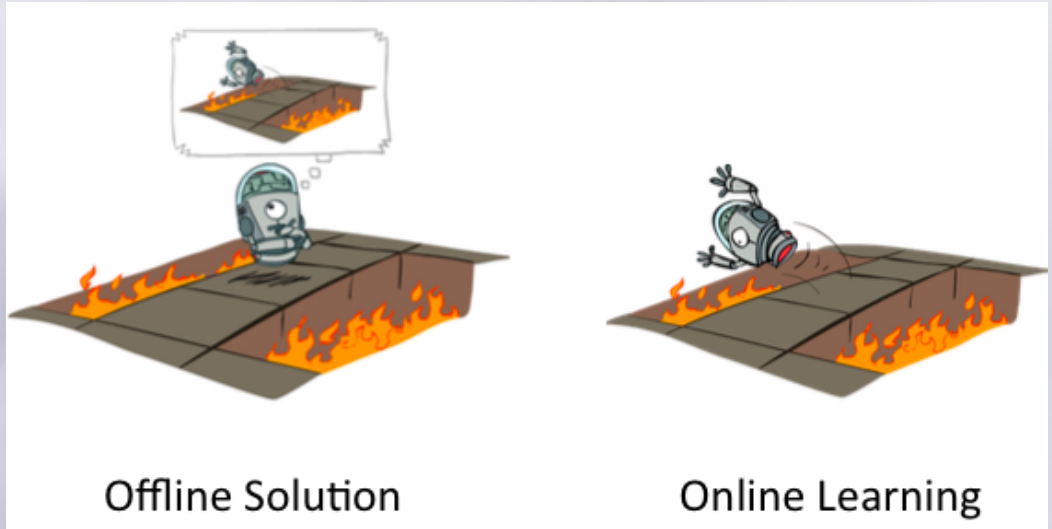
Apprentissage en-ligne

Apprentissage en-ligne

Modèle MDP partiellement connu de l'agent (T et R inconnus) :

- L'agent doit interagir avec l'environnement pour compenser ce manque d'information

$\rightarrow$ problème d'apprentissage par renforcement



Planification vs. Apprentissage

Différentes formes d'apprentissage

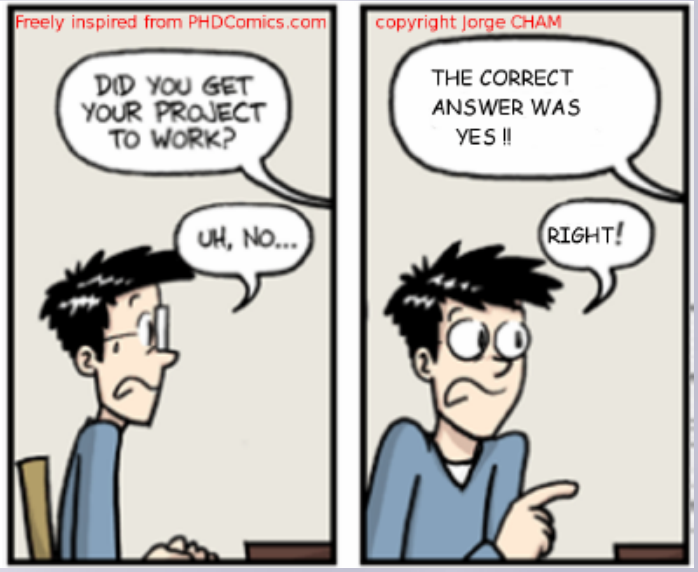
	Supervisé	Non supervisé	Par renforcement
Apprend Information Méthode			

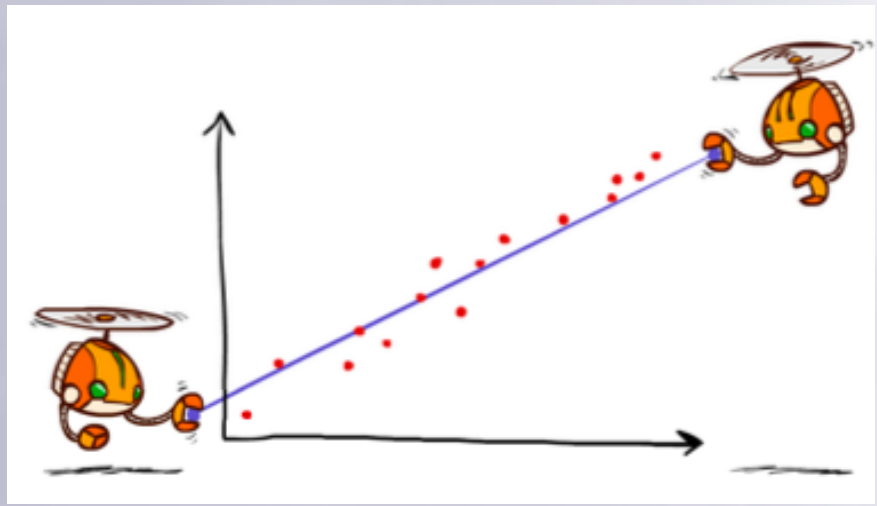
Différentes formes d'apprentissage

	Supervisé	Non supervisé	Par renforcement
Apprend Information Méthode	relations sortie désirée par instruction		

Apprentissage supervisé

- Ensemble d'exemples (X_k, Y_k) où $Y = f(X)$ avec f fonction cible (inconnue) à apprendre.
- Application : approximation de fonction, classification, régression ...



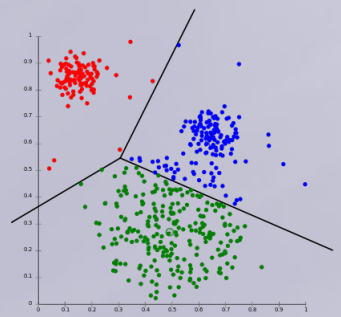


L'apprentissage c'est quoi ?

	Supervisé	Non supervisé	Par renforcement
Apprend	relations	structures ou patterns	
Information	sortie désirée	rien	
Méthode	par instruction	par observation	

Apprentissage non supervisé

- Identifier des structures dans des données $(X_k)_k$ (ex. des classes)
- Regrouper les données selon des critères de ressemblance inconnus *a priori*
- Application : clustering (agrégation)



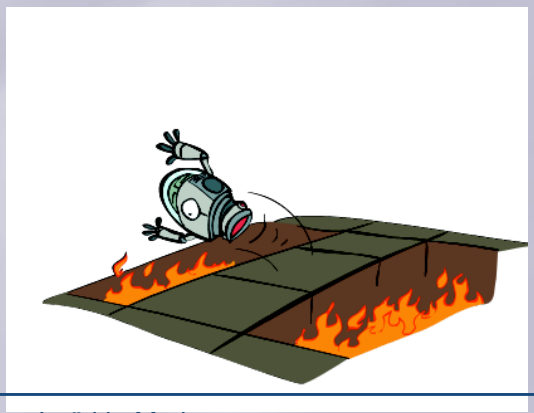
k-means

L'apprentissage c'est quoi ?

	Supervisé	Non supervisé	Par renforcement
Apprend	relations	structures ou patterns	politique
Information	sortie désirée	rien	récompense
Méthode	par instruction	par observation	par essai/erreur

Apprentissage par renforcement

- Apprentissage basé sur l'interaction avec l'environnement : l'apprenant est actif
- Apprendre par l'expérience en fonction des échecs ou succès constatés (les **renforcements** ou **récompenses**).



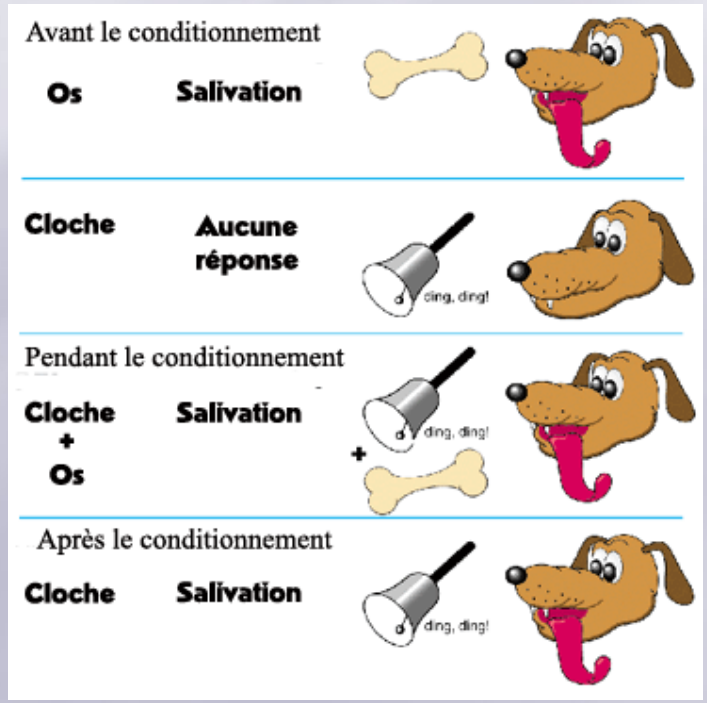
L'apprentissage par renforcement ?

- Apprendre un comportement par l'expérience en fonction des échecs ou succès constatés (les **renforcements** ou **récompenses**).
- Apprendre à associer des actions à des situations pour **maximiser la récompense.**

Origines de l'apprentissage par renforcement

C'est la rencontre de . . .

- Psychologie expérimentale
- Modèles de conditionnement animal :
  - conditionnement classique/pavlovien : association entre stimulus neutre (cloche) et inconditionnel (induit réflexe) donne stimulus conditionnel



Chien de pavlov

Origines de l'apprentissage par renforcement

C'est la rencontre de . . .

- Psychologie expérimentale

- Modèles de conditionnement animal :
  - conditionnement opérant : apprendre par essais-erreurs une action conditionnée par une situation
    - La récompense renforce l'action "presser le bouton"
    - Modifier la récompense modifie le comportement appris

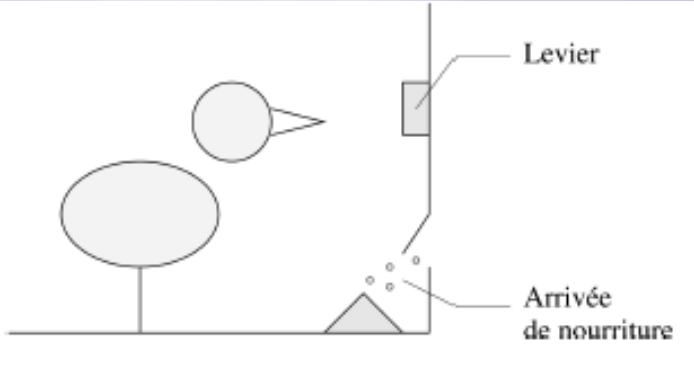


FIGURE – Boite de Skinner

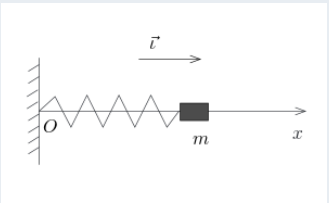
Loi de l'effet ([Thorndike,1911]) :

Une action suivie d'un stimulus agréable sera renforcée et une action suivie d'un stimulus désagréable tendra à diminuer.

Origines de l'apprentissage par renforcement

C'est la rencontre de . . .

- Psychologie expérimentale : le courant béhavioriste étudie la manière dont une association entre des stimuli et des réflexes ou actions peut être renforcée
  renforcement = satisfaction, plaisir ou inconfort, douleur
- Théorie du contrôle optimal : concevoir des contrôleurs capables de maximiser un critère de performance d'un système dynamique
  renforcement = critère à maximiser.



- Processus décisionnel markovien (formalisation)

Reproduction du conditionnement animal pour le contrôle de systèmes.

Objectifs de l'Apprentissage par Renforcement

Objectifs de l'AR

Comment faire acquérir un comportement en lui distribuant des récompenses ?

- Trouver π^* par un processus d'essais et d'erreurs, sans connaissances *a priori* de R et T
- Apprentissage orienté objectif : contrôleur optimal pour la tâche spécifiée par les récompenses

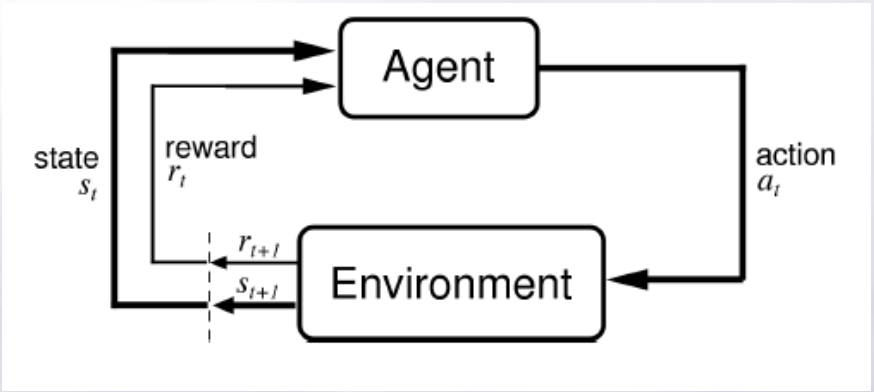
Types d'AR

- AR indirect (*model-based*) :
  - apprendre R et T par interactions (forme d'apprentissage supervisé)
  - s'appuyer sur cette connaissance pour calculer π^* avec des méthodes de planification
- AR direct (*model-free)* : construire π^* en apprenant des fonctions spécifiques mais sans apprentissage direct de R et T

Exemple

Exemple [Kohl and Stone, ICRA 2004]

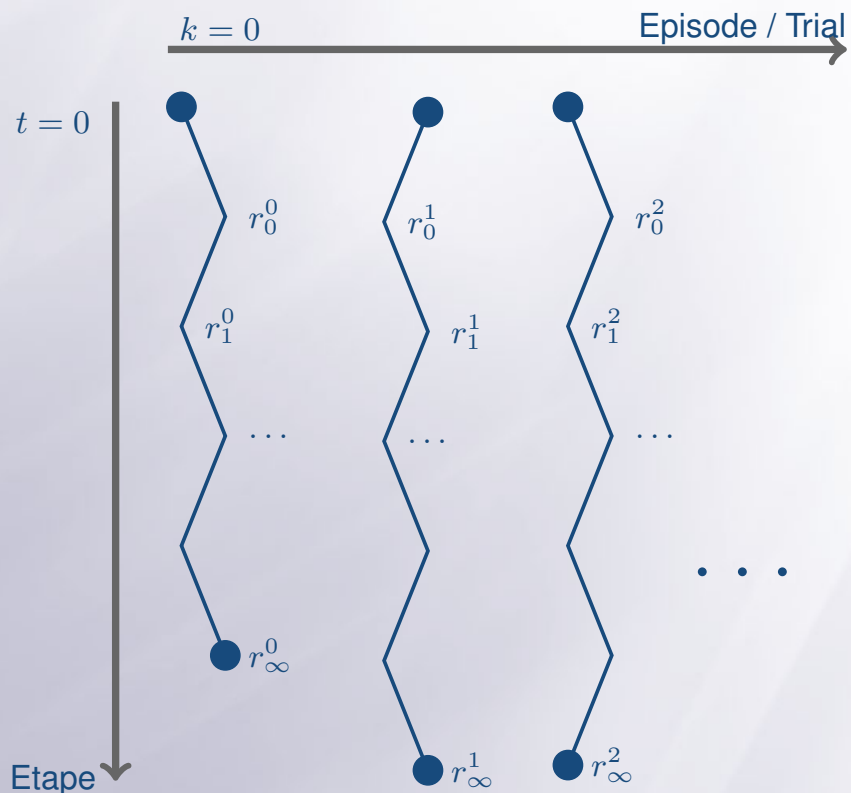
Temporalité en Apprentissage par Renforcement



Agent plongé dans un MDP sans connaissance de R et T.

Double temporalité :

- *Étape t (une action est effectuée)*
- *Épisode k (ensemble d'étapes/actions de l'état initial à un état absorbant)*



Plan

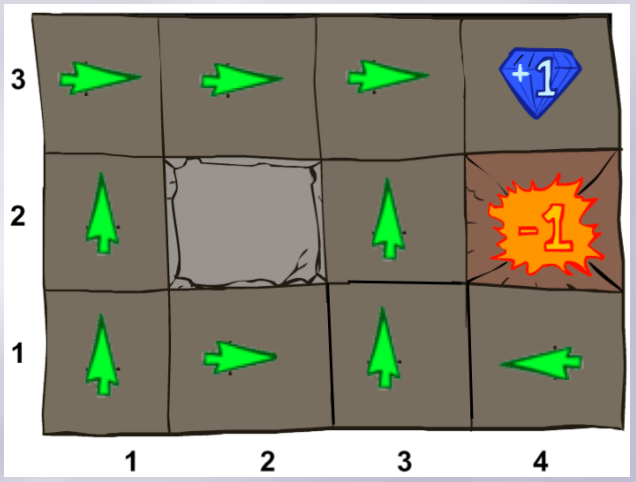
Plan

RL passif

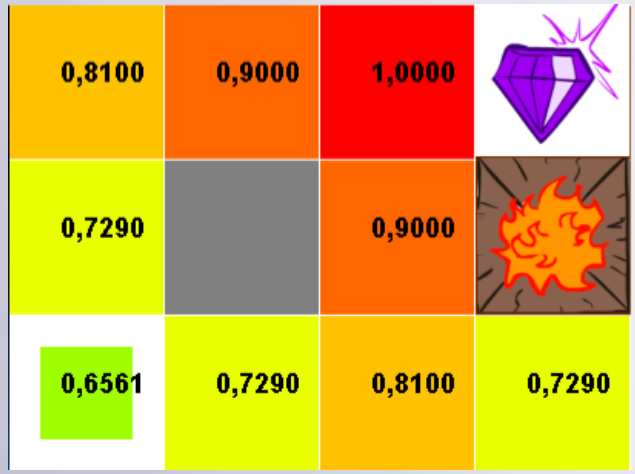
Tâche simplifiée : évaluation de politique

- L'agent exécute une politique π fixée, T et R sont inconnus

Comment évaluer $V^\pi(s)$ pour chaque état s, i.e. évaluer la somme des récompenses espérées depuis chaque état ?



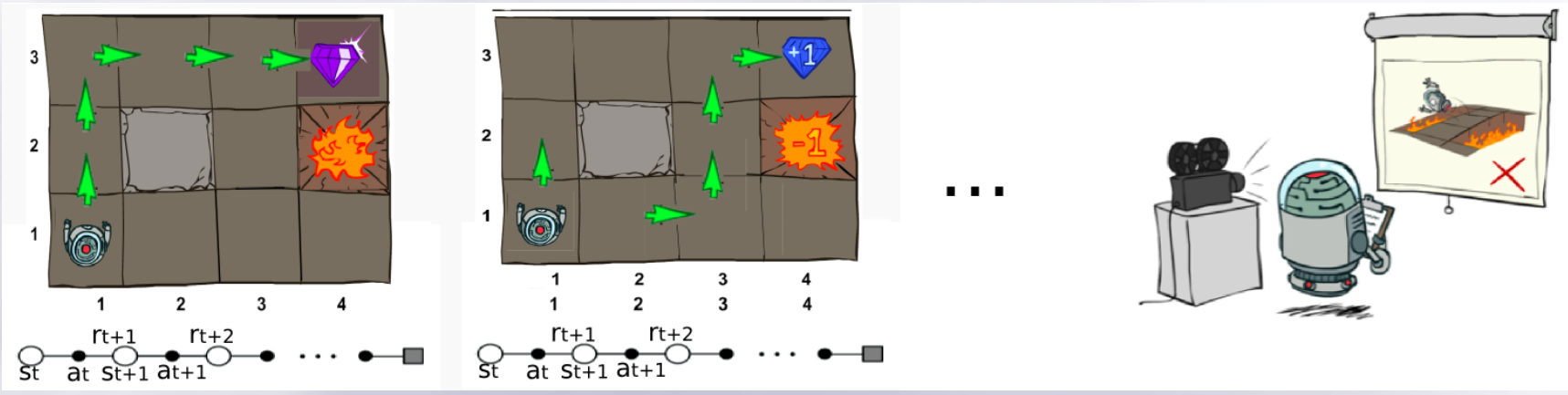
π fixée



V^π

RL passif

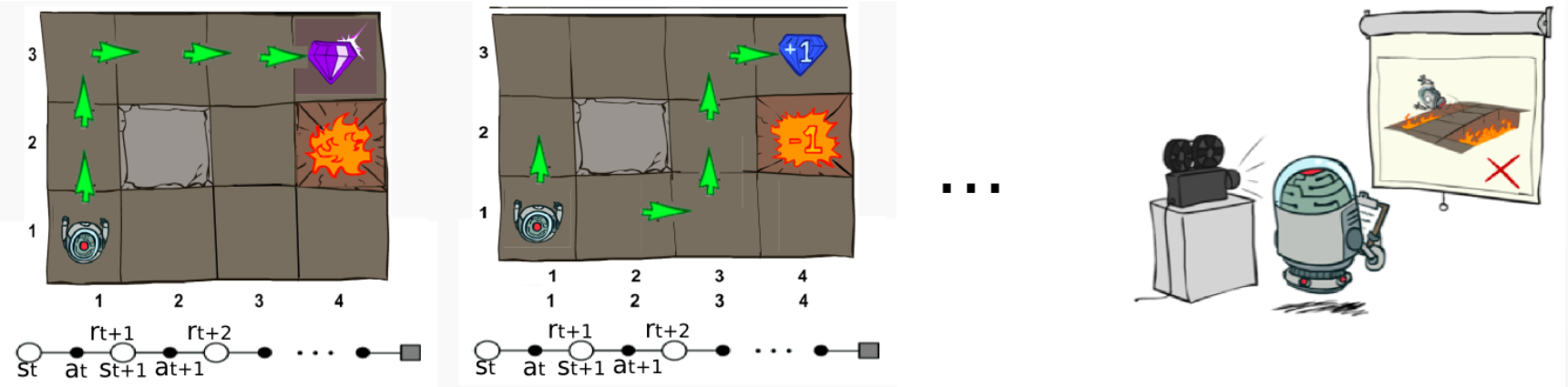
Comment évaluer $V^{\pi}(s)$ pour chaque état s, i.e. évaluer la somme des récompenses espérées depuis chaque état ?



- l'agent interagit directement avec l'environnement : réalisation de plusieurs **épisodes** en suivant π
- **moyenne** des résultats obtenus sur ces épisodes

$\rightarrow$ différentes méthodes pour la moyenne : évaluation directe (moyenne classique), différence temporelle (moyenne incrémentale + amorçage)

Evaluation directe de politique

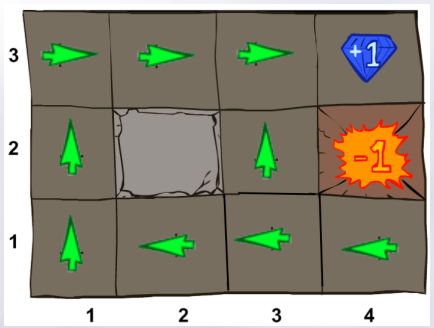


Méthode par moyenne non incrémentale (Monte Carlo)

- on stocke des épisodes (trajectoires) complets réalisés dans l'environnement en suivant la politique π
- on utilise une **moyenne** pour estimer la **valeur** de chaque état :
  - on calcule la **valeur** de chaque état s pour chaque épisode k : $G^\gamma(s_t^k) = r_{t+1}^k + \gamma r_{t+2}^k + \gamma^2 r_{t+3}^k + \ldots$
  - la valeur d'un état est la moyenne de ses valeurs sur les épisodes : $V^\pi(s) = \frac{1}{n} \sum_{i=1}^{n} \{G^\gamma(s_t^i) | s_t^i = s\}$

Evaluation directe de politique : Exemple

π fixée :



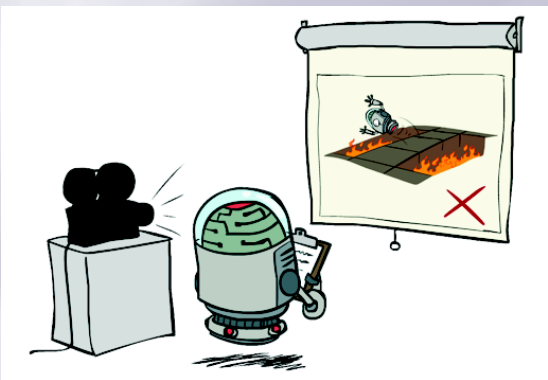
$\gamma = 1, R(*, *, s) = -0.1$

Episode $k = 1$:

t=0 $(1,1)$ NORD -0.1
t=1 $(1,2)$ NORD -0.1
t=2 $(1,2)$ NORD -0.1
t=3 $(1,3)$ EST -0.1
t=4 $(2,3)$ EST -0.1
t=5 $(3,3)$ EST -0.1
t=6 $(3,2)$ NORD -0.1
t=7 $(3,3)$ EST $+1$

Episode $k = 2$:

t=0 $(1,1)$ NORD -0.1
t=1 $(1,2)$ NORD -0.1
t=2 $(1,3)$ EST -0.1
t=3 $(2,3)$ EST -0.1
t=4 $(3,3)$ EST -0.1
t=5 $(3,2)$ NORD -1



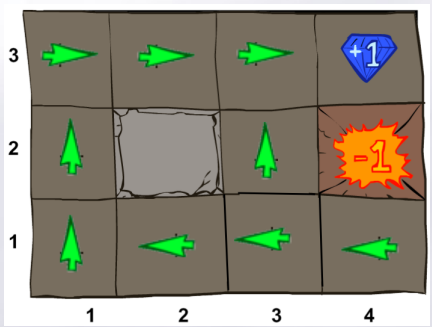
- valeur de état s_t pour chaque épisode k : $G^\gamma(s_t^k) = r_{t+1}^k + \gamma r_{t+2}^k + \gamma^2 r_{t+3}^k + \dots$
- moyenne des valeurs

$$V^\pi(s) = \frac{1}{n} \sum_{k=1}^{n} \{G^\gamma(s_t^k) | s_t^k = s\}$$

$V(< 2,3 >) =? \; V(< 3,3 >) =?$

Evaluation directe de politique : Exemple

π fixée :



$\gamma = 1, R(*, *, s) = -0.1$

Episode $k = 1$:

t=0 $(1,1)$ NORD -0.1
t=1 $(1,2)$ NORD -0.1
t=2 $(1,2)$ NORD -0.1
t=3 $(1,3)$ EST -0.1
t=4 $(2,3)$ EST -0.1
t=5 $(3,3)$ EST -0.1
t=6 $(3,2)$ NORD -0.1
t=7 $(3,3)$ EST $+1$

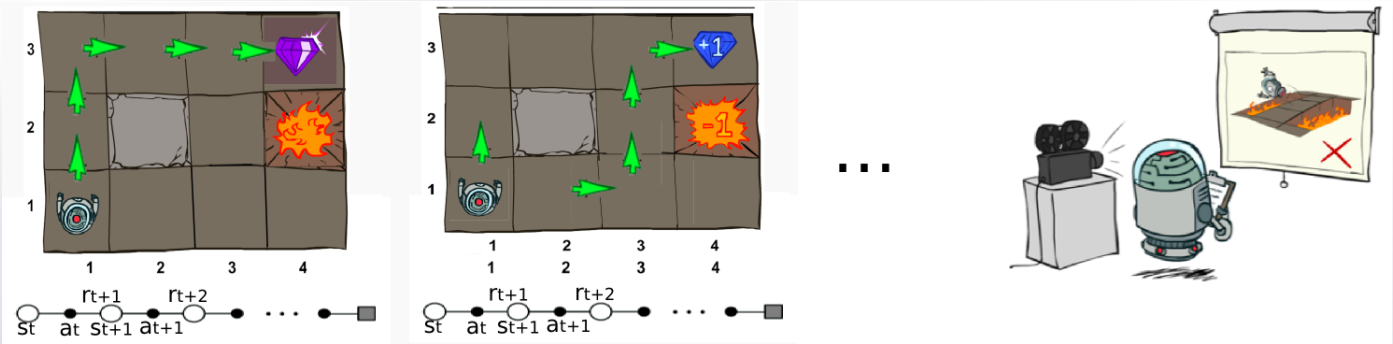
Episode $k = 2$:

t=0 $(1,1)$ NORD -0.1
t=1 $(1,2)$ NORD -0.1
t=2 $(1,3)$ EST -0.1
t=3 $(2,3)$ EST -0.1
t=4 $(3,3)$ EST -0.1
t=5 $(3,2)$ NORD -1

- valeur de état s_t pour chaque épisode k :
  $G^\gamma(s_t^k) = r_{t+1}^k + \gamma r_{t+2}^k + \gamma^2 r_{t+3}^k + ...$
- moyenne des valeurs $V^\pi(s) = \frac{1}{n} \sum_{k=1}^{n} \{G^\gamma(s_t^k) | s_t^k = s\}$

$$V(< 2,3 >) = \frac{1}{2} \times [(1 - 0.1 \times 3) + (-1 - 0.1 \times 2)] = -0.25$$
$$V(< 3,3 >) = \frac{1}{3} \times [(1 - 0.1 \times 2) + (1) + (-1 - 0.1)] = 0.23$$

Evaluation directe de politique



L'évaluation directe de politique nécessite de conserver tous les épisodes $i \in [1, k]$ pour calculer la moyenne :

$$V^{\pi}(s) = \frac{1}{k} \sum_{i=1}^{k} G_i^{\gamma}(s)$$

Comment évaluer itérativement cette moyenne ?

Calcul itératif de moyenne

- Comment calculer la moyenne V des 3 valeurs $v_i = \{1, 4, 7\}$?
- Méthode non incrémentale : $V = \frac{1+4+7}{3} = 4$

Calcul itératif de moyenne

- Comment calculer la moyenne V des 3 valeurs $v_i = \{1, 4, 7\}$?
- Méthode incrémentale : Estimation incrémentale de V en calculant V_{k+1} $(new\ estimation)$ à partir de V_k $(old\ estimation)$.
  A chaque k, on reçoit un nouvel échantillon v_k

$$V_{k+1} = \frac{1}{k+1} \sum_{i=1}^{k+1} v_i \tag{1}$$

$$= \frac{1}{k+1} (v_{k+1} + \sum_{i=1}^{k} v_i) \tag{2}$$

$$= \frac{1}{k+1} (v_{k+1} + kV_k + V_k - V_k) \tag{3}$$

$$V_{k+1} = V_k + \frac{1}{k+1} [v_{k+1} - V_k] \tag{4}$$

$$new\ estimation = old\ estimation +$$
$$\alpha_k [new\ echantillon - old\ estimation]$$

Calcul itératif de moyenne

- Comment calculer la moyenne V des 3 valeurs $v_i = \{1, 4, 7\}$?
- Méthode incrémentale : Estimation incrémentale de V en calculant V_{k+1} $(new\ estimation)$ à partir de V_k $(old\ estimation)$.
  A chaque k, on reçoit un nouvel échantillon v_k

$$\underbrace{V_{k+1}}_{new\_estim} = \underbrace{V_k}_{old\_estim} + \underbrace{\frac{1}{k+1}}_{\alpha_k} \underbrace{[v_{k+1} - V_k]}_{new\_echantillon - old\_estim}$$

- $k = 0\ V_0 = 0$
- $k = 1\ v_1 = 1 : V_1 = V_0 + \frac{1}{1}[v_1 - V_0] = 0 + \frac{1}{1}[1 - 0] = 1$
- $k = 2\ v_2 = 4 : V_2 = V_1 + \frac{1}{2}[v_2 - V_1] = 1 + \frac{1}{2}[4 - 1] = 2.5$
- $k = 3\ v_3 = 7 : V_3 = V_2 + \frac{1}{3}[v_3 - V_2] = 2.5 + \frac{1}{3}[7 - 2.5] = 4$

α variable : les anciennes et nouvelles valeurs v_i sont pondérées de la même façon.

Calcul itératif de moyenne

- Méthode incrémentale : Estimation incrémentale de V en calculant V_{k+1} à partir de V_k

$$\underbrace{V_{k+1}}_{new\_estim} = (1-\alpha) \underbrace{V_k}_{old\_estim} + \alpha \underbrace{(v_{k+1})}_{new\_echantillon}$$

- Si on utilise α constant :
- $n = 0\ V_0 = 0$
- $n = 1\ V_1 = v_1$
- $n = 2\ V_2 = (1-\alpha)V_1 + \alpha v_2 = (1-\alpha)v_1 + \alpha v_2$
- $n = 3\ V_3 = (1-\alpha)V_2 + \alpha v_3 = (1-\alpha)^2 v_1 + \alpha(1-\alpha)v_2 + \alpha v_3$

α constant : les anciennes et nouvelles valeurs v_i ne sont plus pondérées de la même façon : oubli exponentiel des anciennes valeurs v_i.

Mise à jour incrémentale de V

Comment appliquer le calcul itératif de la moyenne pour évaluer la valeur $V^\pi(s)$ d'un état selon une politique π ?

- $V^\pi(s)$ est la **somme des récompenses espérées** si on suit π depuis s
- Evaluation iterative de $V^\pi(s)$:

$$\underbrace{V^\pi_{t+1}(s)}_{new\_estim} = (1-\alpha) \underbrace{V^\pi_t(s)}_{old\_estim} + \alpha \underbrace{(v_{t+1}(s))}_{new\_echantillon}$$

Quand mettre à jour $V^\pi(s)$?
Quel est le nouvel échantillon ?

Mise à jour incrémentale de V

- $V^\pi(s)$ est la **somme des récompenses espérées** si on suit π depuis s

$$\underbrace{V^\pi_{t+1}(s)}_{new\_estim} = (1-\alpha) \underbrace{V^\pi_t(s)}_{old\_estim} + \alpha \underbrace{(v_{t+1}(s))}_{new\_echantillon}$$

Quand mettre à jour $V^\pi(s)$? A chaque étape

- Une étape $< s, a, s', r >$: l'agent est dans l'état s, agit ($a = \pi(s)$), reçoit r, arrive dans état s'

Quel est le nouvel **échantillon** à chaque étape ?

- une évaluation de la somme des récompenses espérées depuis s pour une étape $< s, a, s', r >$ est

$$v(s) = r + \gamma V^\pi(s')$$

($V(s')$ estime la somme des récompenses depuis s')

Mise à jour incrémentale par étape

Comment appliquer le calcul itératif de la moyenne pour évaluer la valeur $V^\pi(s)$ d'un état selon une politique π ?

Mise à jour par différence temporelle (TD)

Après chaque action exécutée par l'agent depuis un état s (étape $< s, \pi(s), s', r >$), on applique la mise à jour suivante à l'état s :

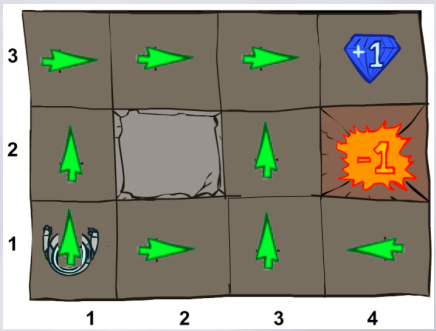
$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$

- coefficient d'apprentissage $\alpha \in [0; 1]$

Mise à jour TD : Exemple

Après chaque action exécutée par l'agent depuis un état s, mise à jour de :

$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$



La politique à évaluer avec $\gamma = 1.0$ et $\alpha = 0.1$

Episode 1 :

t=0 $(1,1)$ NORD 0

t=1 $(1,2)$ NORD 0

t=2 $(1,3)$ EST 0

t=3 $(2,3)$ EST 0

t=4 $(3,3)$ EST $+1$

Episode 2 :

t=0 $(1,1)$ NORD 0

t=1 $(1,2)$ NORD 0

t=2 $(1,3)$ EST 0

t=3 $(2,3)$ EST 0

t=4 $(3,3)$ EST 0

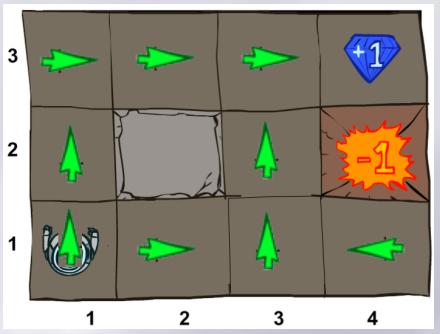
t=5 $(3,2)$ NORD -1

Précisez pour chaque étape de chaque épisode, l'état mis à jour et sa valeur.

Mise à jour TD : Exemple

Après chaque action exécutée par l'agent depuis un état s, mise à jour de :

$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$



La politique à évaluer avec $\gamma = 1.0$ et $\alpha = 0.1$

Episode 1 :

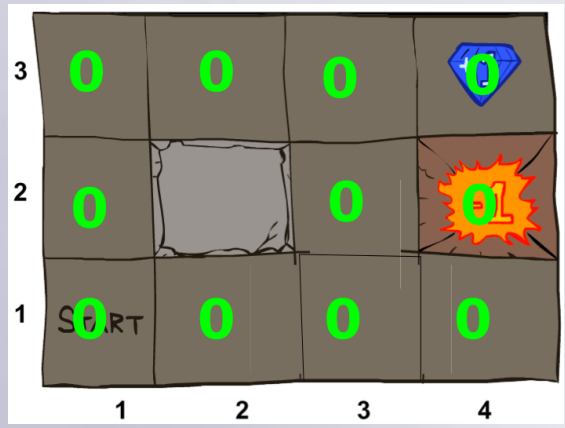
t=0 $(1,1)$ NORD 0

t=1 $(1,2)$ NORD 0

t=2 $(1,3)$ EST 0

t=3 $(2,3)$ EST 0

t=4 $(3,3)$ EST $+1$

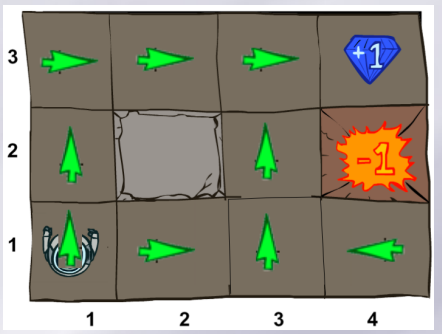


V init à 0 $\forall s$

Mise à jour TD : Exemple

Après chaque action exécutée par l'agent depuis un état s, mise à jour de :

$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$



La politique à évaluer avec $\gamma = 1.0$ et $\alpha = 0.1$

Episode 1 :

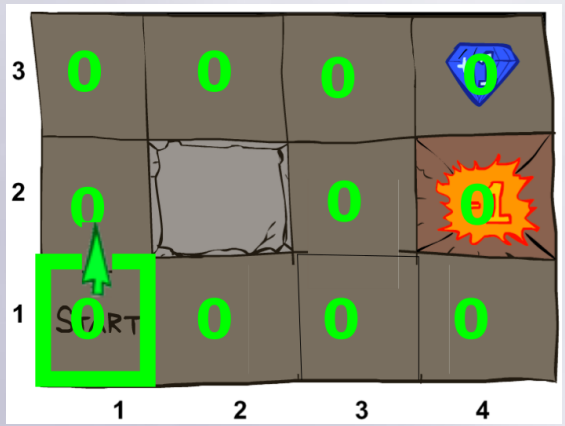
t=0 $(1,1)$ NORD 0

t=1 $(1,2)$ NORD 0

t=2 $(1,3)$ EST 0

t=3 $(2,3)$ EST 0

t=4 $(3,3)$ EST $+1$

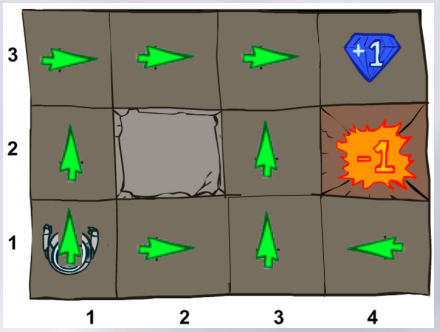


Episode 1, $t = 0$,
$s =< 1,1 >, s' =< 1,2 >$
$V^\pi(s) \leftarrow 0 + \alpha[0 + \gamma 0 - 0]$

Mise à jour TD : Exemple

Après chaque action exécutée par l'agent depuis un état s, mise à jour de :

$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$



La politique à évaluer avec $\gamma = 1.0$ et $\alpha = 0.1$

Episode 1 :

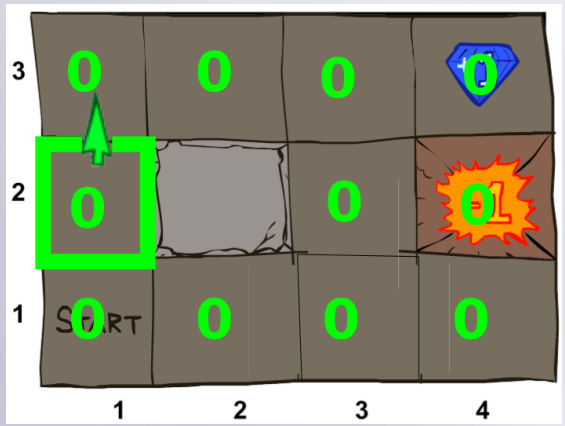
t=0 $(1,1)$ NORD 0

t=1 $(1,2)$ NORD 0

t=2 $(1,3)$ EST 0

t=3 $(2,3)$ EST 0

t=4 $(3,3)$ EST $+1$

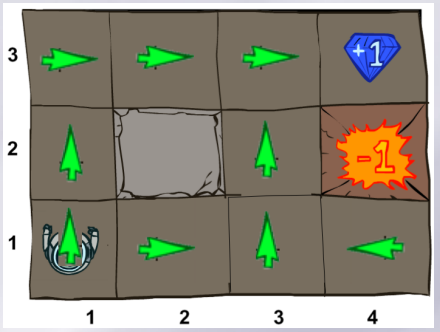


Episode 1, $t = 1$,
$s =< 1, 2 >, s' =< 1, 3 >$
$V^\pi(s) \leftarrow 0 + \alpha[0 + \gamma 0 - 0]$

Mise à jour TD : Exemple

Après chaque action exécutée par l'agent depuis un état s, mise à jour de :

$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$



La politique à évaluer avec $\gamma = 1.0$ et $\alpha = 0.1$

Episode 1 :

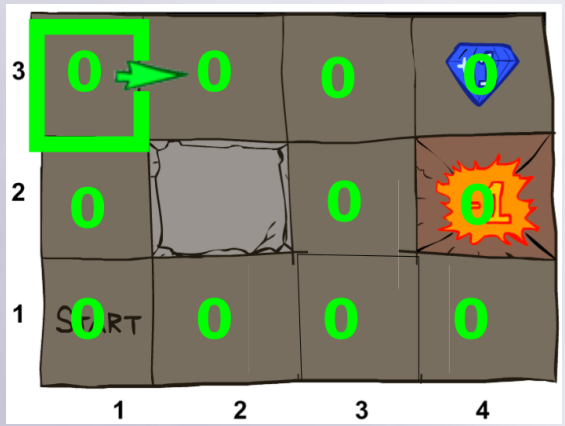
t=0 $(1,1)$ NORD 0

t=1 $(1,2)$ NORD 0

t=2 $(1,3)$ EST 0

t=3 $(2,3)$ EST 0

t=4 $(3,3)$ EST $+1$

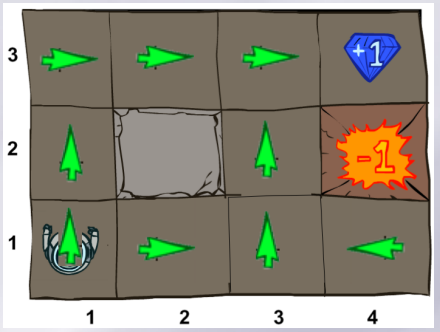


Episode 1, $t = 2$,
$s =< 1,3 >, s' =< 2,3 >$
$V^\pi(s) \leftarrow 0 + \alpha[0 + \gamma 0 - 0]$

Mise à jour TD : Exemple

Après chaque action exécutée par l'agent depuis un état s, mise à jour de :

$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$



La politique à évaluer avec $\gamma = 1.0$ et $\alpha = 0.1$

Episode 1 :

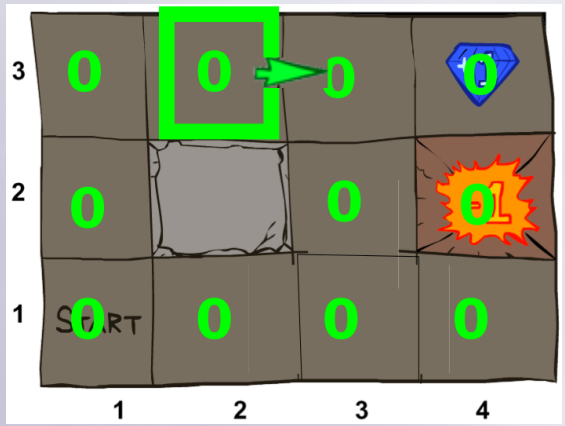
t=0 $(1,1)$ NORD 0

t=1 $(1,2)$ NORD 0

t=2 $(1,3)$ EST 0

t=3 $(2,3)$ EST 0

t=4 $(3,3)$ EST $+1$

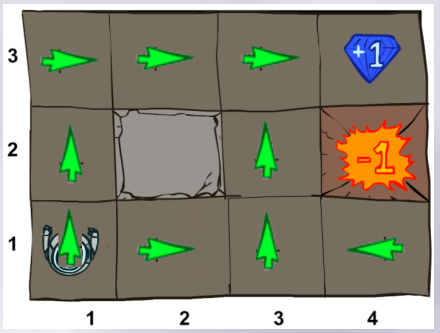


Episode 1, $t = 3$,
$s =< 2, 3 >, s' =< 3, 3 >$
$V^\pi(s) \leftarrow 0 + \alpha[0 + \gamma 0 - 0]$

Mise à jour TD : Exemple

Après chaque action exécutée par l'agent depuis un état s, mise à jour de :

$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$



La politique à évaluer avec $\gamma = 1.0$ et $\alpha = 0.1$

Episode 1 :

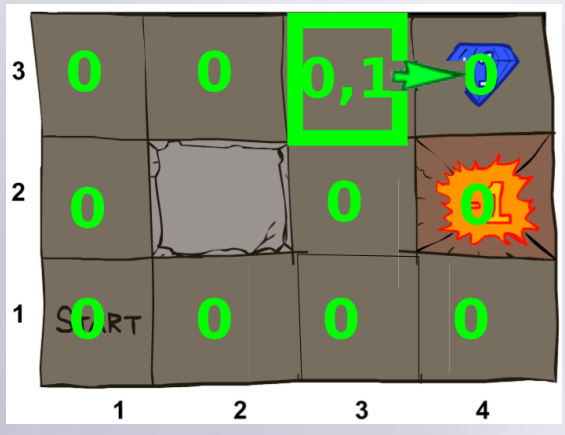
t=0 $(1,1)$ NORD 0

t=1 $(1,2)$ NORD 0

t=2 $(1,3)$ EST 0

t=3 $(2,3)$ EST 0

t=4 $(3,3)$ EST $+1$

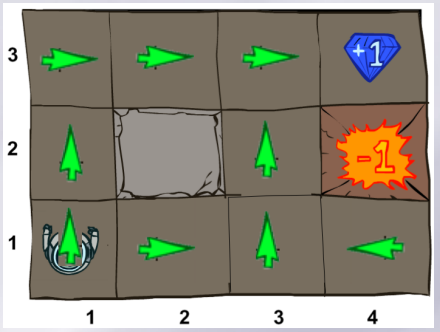


Episode 1, $t = 4$,
$s =< 3,3 >, s' =< 4,3 >$
$V^\pi(s) \leftarrow 0 + \alpha[1 + \gamma 0 - 0] = 0.1$

Mise à jour TD : Exemple

Après chaque action exécutée par l'agent depuis un état s, mise à jour de :

$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$



La politique à évaluer avec $\gamma = 1.0$ et $\alpha = 0.1$

Episode 2 :

t=0 $(1,1)$ NORD 0

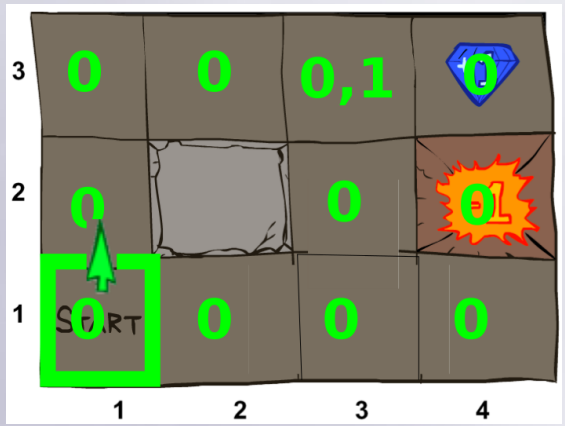
t=1 $(1,2)$ NORD 0

t=2 $(1,3)$ EST 0

t=3 $(2,3)$ EST 0

t=4 $(3,3)$ EST 0

t=5 $(3,2)$ NORD -1

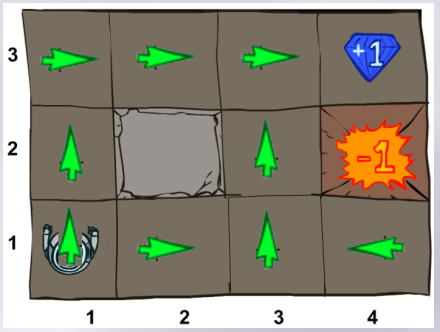


Episode 2, $t = 0$,
$s =< 1,1 >, s' =< 1,2 >$
$V^\pi(s) \leftarrow 0 + \alpha[0 + \gamma 0 - 0]$

Mise à jour TD : Exemple

Après chaque action exécutée par l'agent depuis un état s, mise à jour de :

$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$



La politique à évaluer avec $\gamma = 1.0$ et $\alpha = 0.1$

Episode 2 :

t=0 $(1,1)$ NORD 0

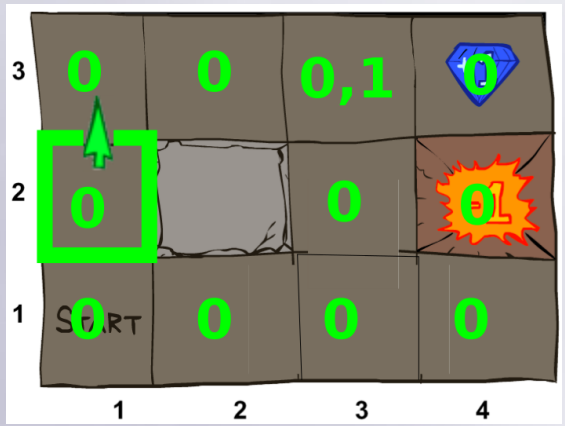
t=1 $(1,2)$ NORD 0

t=2 $(1,3)$ EST 0

t=3 $(2,3)$ EST 0

t=4 $(3,3)$ EST 0

t=5 $(3,2)$ NORD -1

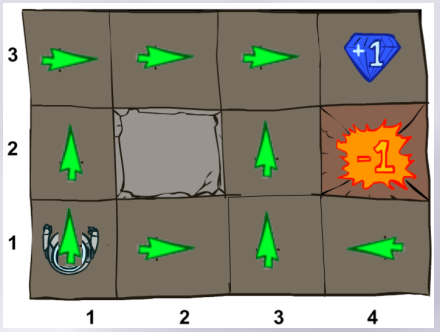


Episode 2, $t = 1$,
$s =< 1,2 >, s' =< 1,3 >$
$V^\pi(s) \leftarrow 0 + \alpha[0 + \gamma 0 - 0]$

Mise à jour TD : Exemple

Après chaque action exécutée par l'agent depuis un état s, mise à jour de :

$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$



La politique à évaluer avec $\gamma = 1.0$ et $\alpha = 0.1$

Episode 2 :

t=0 $(1,1)$ NORD 0

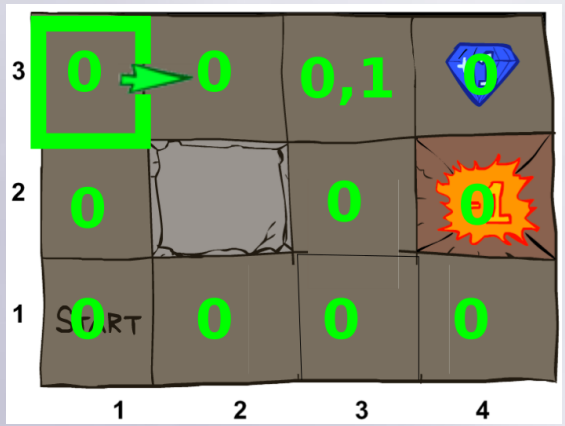
t=1 $(1,2)$ NORD 0

t=2 $(1,3)$ EST 0

t=3 $(2,3)$ EST 0

t=4 $(3,3)$ EST 0

t=5 $(3,2)$ NORD -1

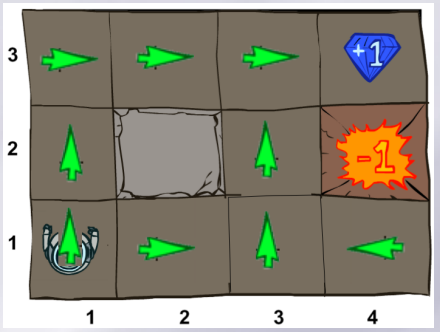


Episode 2, $t = 2$,
$s =< 1,3 >, s' =< 2,3 >$
$V^\pi(s) \leftarrow 0 + \alpha[0 + \gamma 0 - 0]$

Mise à jour TD : Exemple

Après chaque action exécutée par l'agent depuis un état s, mise à jour de :

$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$



La politique à évaluer avec $\gamma = 1.0$ et $\alpha = 0.1$

Episode 2 :

t=0 $(1,1)$ NORD 0

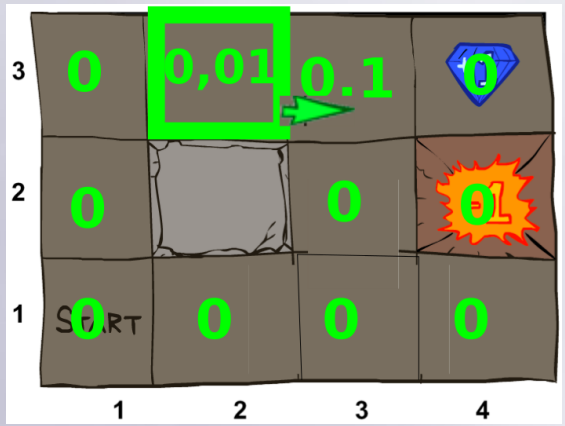
t=1 $(1,2)$ NORD 0

t=2 $(1,3)$ EST 0

t=3 $(2,3)$ EST 0

t=4 $(3,3)$ EST 0

t=5 $(3,2)$ NORD -1

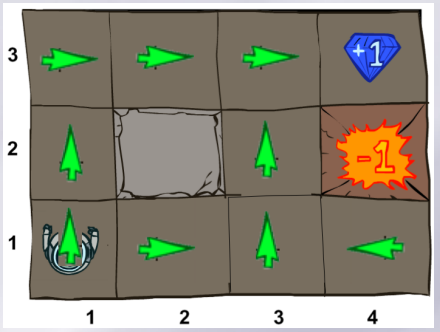


Episode 2, $t = 3$,
$s =< 2,3 >, s' =< 3,3 >$
$V^\pi(s) \leftarrow 0 + \alpha[0 + \gamma 0.1 - 0]$

Mise à jour TD : Exemple

Après chaque action exécutée par l'agent depuis un état s, mise à jour de :

$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$



La politique à évaluer avec $\gamma = 1.0$ et $\alpha = 0.1$

Episode 2 :

t=0 $(1,1)$ NORD 0

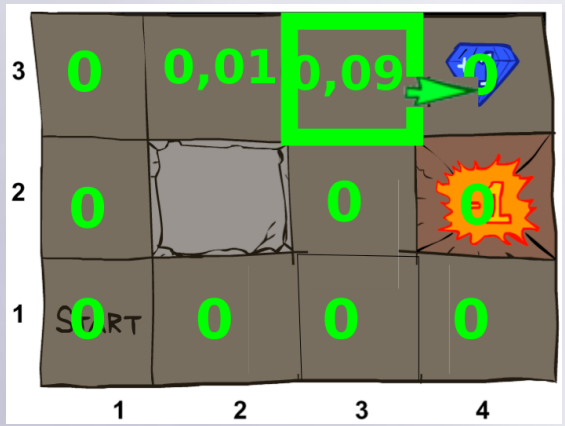
t=1 $(1,2)$ NORD 0

t=2 $(1,3)$ EST 0

t=3 $(2,3)$ EST 0

t=4 $(3,3)$ EST 0

t=5 $(3,2)$ NORD -1

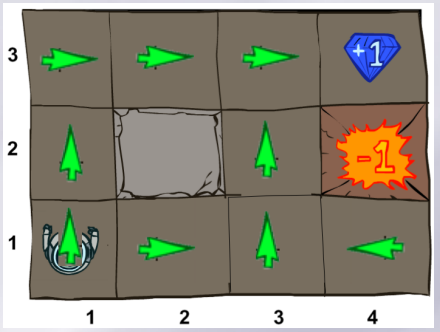


Episode 2, $t = 4$,
$s =< 3,3 >, s' =< 3,2 >$
$V^\pi(s) \leftarrow 0.1 + \alpha[0 + \gamma 0 - 0.1]$

Mise à jour TD : Exemple

Après chaque action exécutée par l'agent depuis un état s, mise à jour de :

$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$



La politique à évaluer avec $\gamma = 1.0$ et $\alpha = 0.1$

Episode 2 :

t=0 $(1,1)$ NORD 0

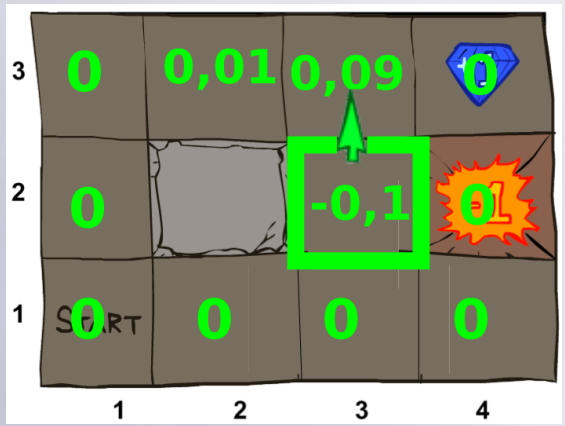
t=1 $(1,2)$ NORD 0

t=2 $(1,3)$ EST 0

t=3 $(2,3)$ EST 0

t=4 $(3,3)$ EST 0

t=5 $(3,2)$ NORD -1



Episode 2, $t = 5$,
$s =< 3,2 >, s' =< 4,2 >$
$V^\pi(s) \leftarrow 0 + \alpha[-1 + \gamma 0 - 0]$

Résumons ...

- Un agent suit une politique fixée
- Il évalue cette politique sans connaitre le modèle avec :

TD

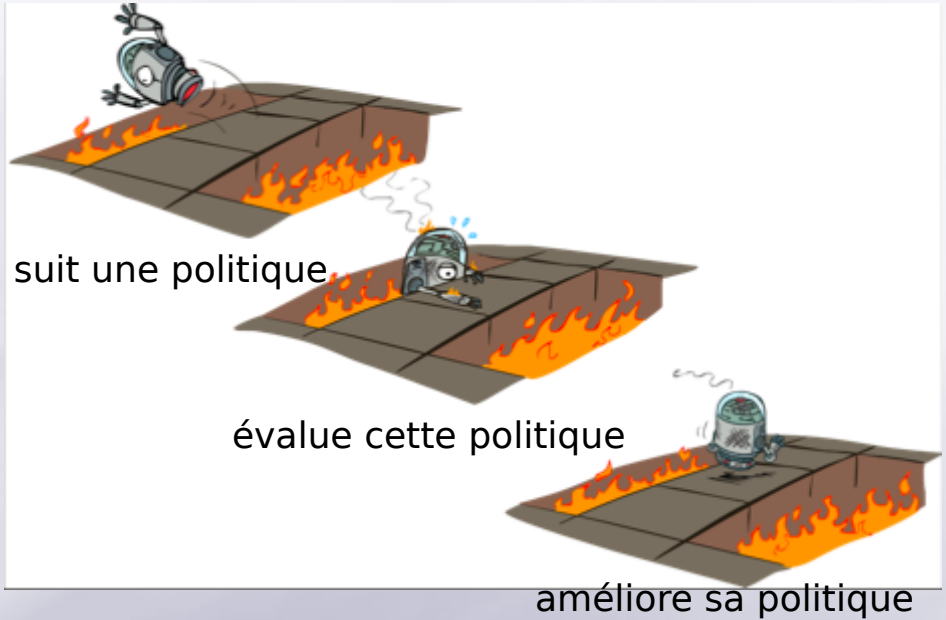
Mise à jour après chaque action de l'état s où il a exécuté une action :

$$\underbrace{V^\pi(s)}_{new\_estim} \leftarrow (1-\alpha) \underbrace{V^\pi(s)}_{old\_estim} + \alpha \times (\underbrace{r + \gamma V^\pi(s')}_{new\_echantillon})$$

- L'objectif de l'agent apprenant est de trouver π^*
- Ici l'agent n'est pas actif !

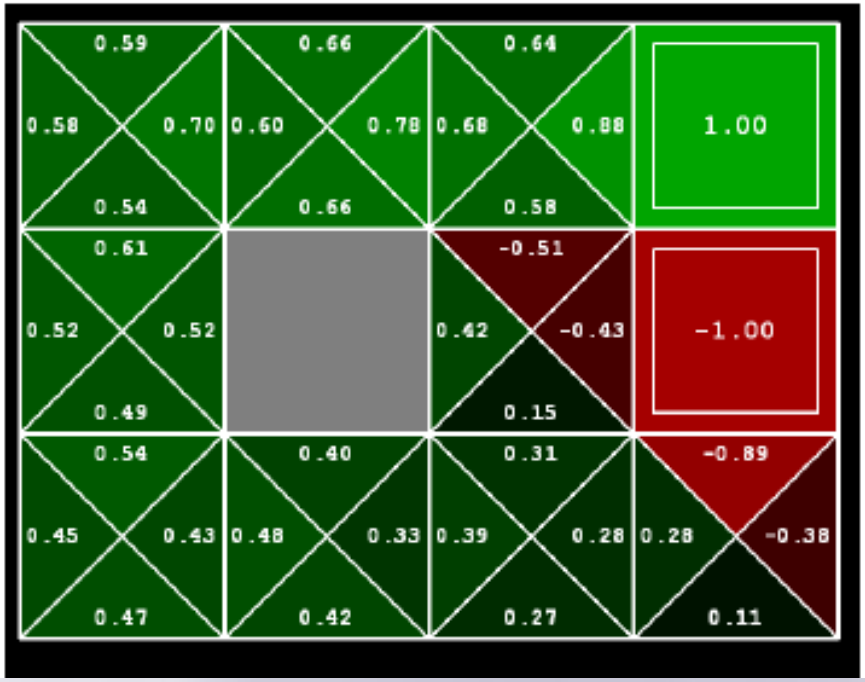
Plan

Agent glouton (*greedy*)



- π n'est plus figée mais l'agent la modifie
- Objectif de l'agent : trouver la politique optimale π^* sans connaitre T et R
  - L'agent suit une politique π
  - Il évalue cette politique (mise à jour de $V^\pi(s)$ selon TD)
  - Après chaque évaluation, il améliore la politique π suivie.

Fonction de valeur d'action Q



$$Q : S \times A \to \Re$$

- $Q^\pi(s,a)$ évalue le retour espéré lorsque l'on effectue l'action a dans l'état s puis que l'on suit π : $Q^\pi(s,a) = E\{\sum_{t=0}^{\infty} \gamma^t r_{t+1} | \pi, s_t = s, a_t = a\}$
- Liens entre V et Q : $\forall s \in S \quad V(s) = \max_a Q(s,a)$

Amélioration de politique en-ligne

Politique gloutonne : action de plus grande valeur dans s

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

Apprentissage par renforcement actif

Agent glouton

A chaque étape :

- L'agent dans s suit sa politique $a = \pi(s)$
- Il évalue cette politique : mise à jour de $Q^\pi(s,a)$ selon TD après chaque action :

$$\underbrace{Q^\pi(s,a)}_{new\_estim} \leftarrow (1-\alpha)\underbrace{Q^\pi(s,a)}_{old\_estim} + \alpha \times \underbrace{(r + \gamma \max_b Q^\pi(s',b))}_{new\_echantillon}$$

- Après chaque mise à jour, il améliore sa politique π :

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

Apprentissage par renforcement actif

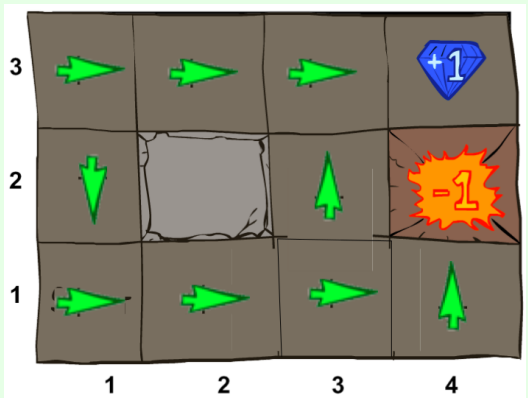
Agent glouton

- L'agent dans s suit sa politique $a = \pi(s)$
- Il évalue cette politique :

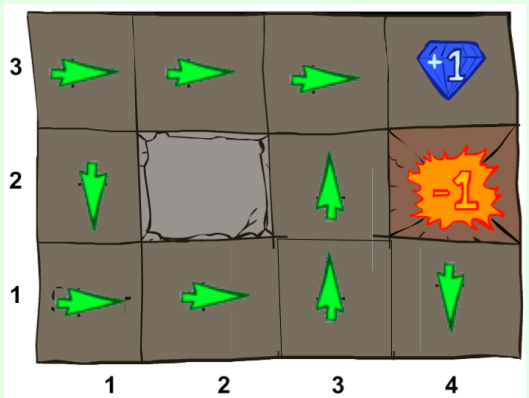
$$Q^\pi(s,a) \leftarrow (1-\alpha)Q^\pi(s,a) + \alpha \times (r + \gamma \max_b Q^\pi(s',b))$$

- Il améliore sa politique π : $\pi(s) = \arg\max_{a \in A} Q^\pi(s,a)$

Example



π initiale



π après 500 itérations

Apprentissage par renforcement actif

Agent glouton

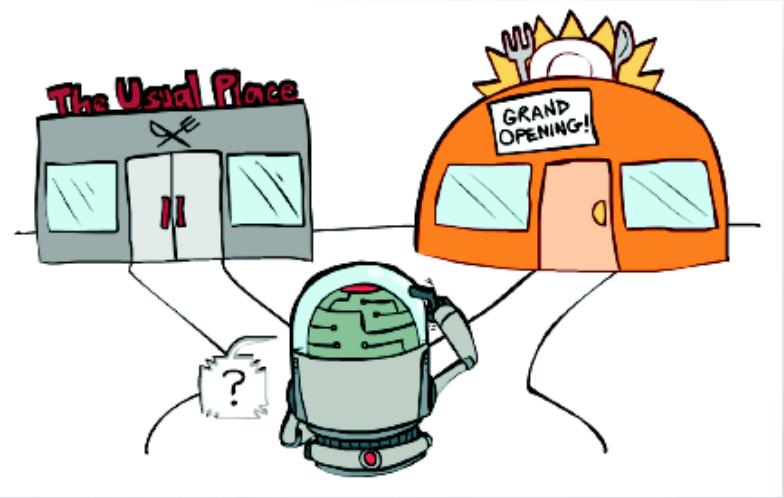
- L'agent dans s suit sa politique $a = \pi(s)$
- Il évalue cette politique :

$$Q^\pi(s,a) \leftarrow (1-\alpha)Q^\pi(s,a) + \alpha \times (r + \gamma \max_b Q^\pi(s',b))$$

- Il améliore sa politique π : $\pi(s) = \arg\max_{a \in A} Q^\pi(s,a)$

Approche limitée par le manque d'exploration.

Exploration et Exploitation



Trouver le compromis entre exploration et exploitation.

Exploitation / Glouton

Choisir la meilleure action en fonction de ce que l'on connait jusqu'à présent (politique gloutonne)

Exploration

Essayer une action sous-optimale jusqu'à présent
$\rightarrow$ gain de connaissance pour améliorer Q tant que Q^* n'a pas été trouvée : accélération de l'apprentissage
$\rightarrow$ perte de temps si l'agent a déjà trouvé Q^*

Stratégies d'exploration

Politique ϵ-*greedy*

Explorer avec une probabilité ϵ :

$\pi_{greedy}(s)$ =

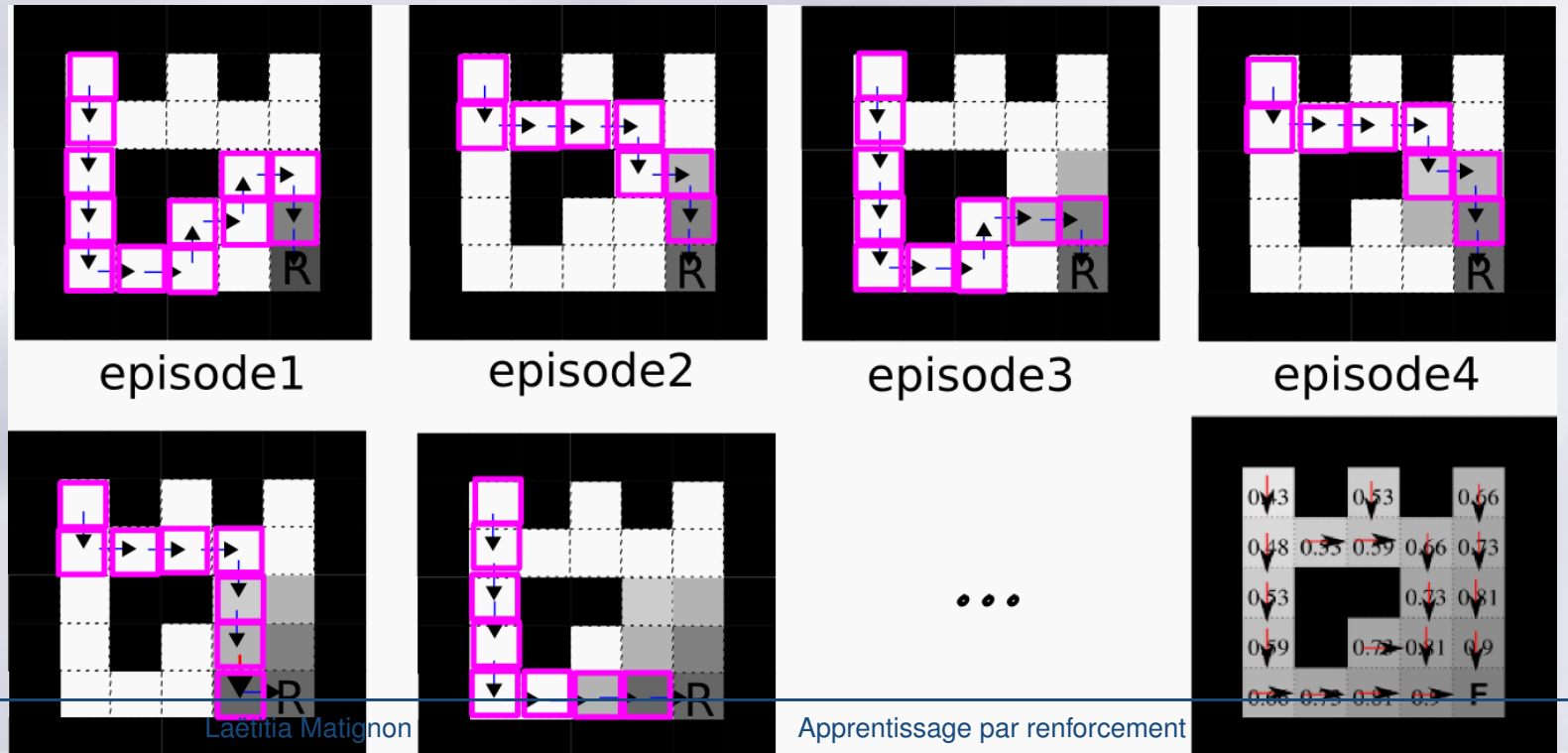
- action d'exploration (aléatoire) avec la probabilité ϵ
- action gloutonne ($a = \arg\max_a Q(s,a)$) avec la probabilité $1 - \epsilon$

ϵ-*greedy* traite toutes les actions, sauf la meilleure, de façon équivalente.

Algorithme du Q-learning

Construire une table des Q-valeurs (dimension $|S| \times |A|$) et répéter pour chaque épisode :

- $s \leftarrow$ état initial
- Répéter pour chaque étape dans l'épisode
  - Choisir a selon une stratégie d'exploration (apprentissage actif)
  - Exécuter a, observer r et s'
  - $Q(s,a) \leftarrow (1-\alpha_k)Q(s,a) + \alpha_k[r + \gamma \max_b Q(s',b)]$ (mise à jour TD)
  - $s \leftarrow s'$



Algorithme du Q-learning

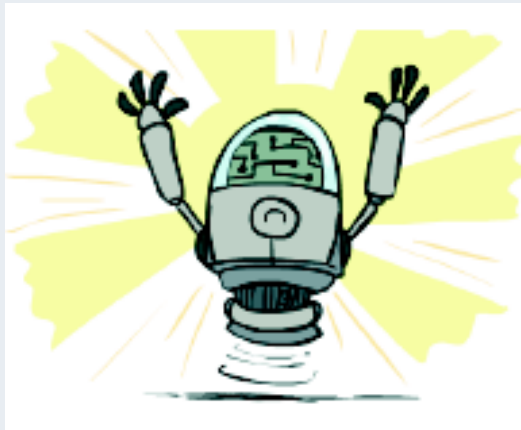
Propriétés du Q-learning

Q-learning converge vers Q^* si [Jaakkola,1994] :

- S et A finis
- tous les couples (s, a) doivent être visités un nombre infini de fois

  $\rightarrow$ la stratégie d'exploration doit permettre assez d'exploration
- la suite $\alpha_k(s, a)$ doit décroitre à chaque nouvelle visite de (s, a)

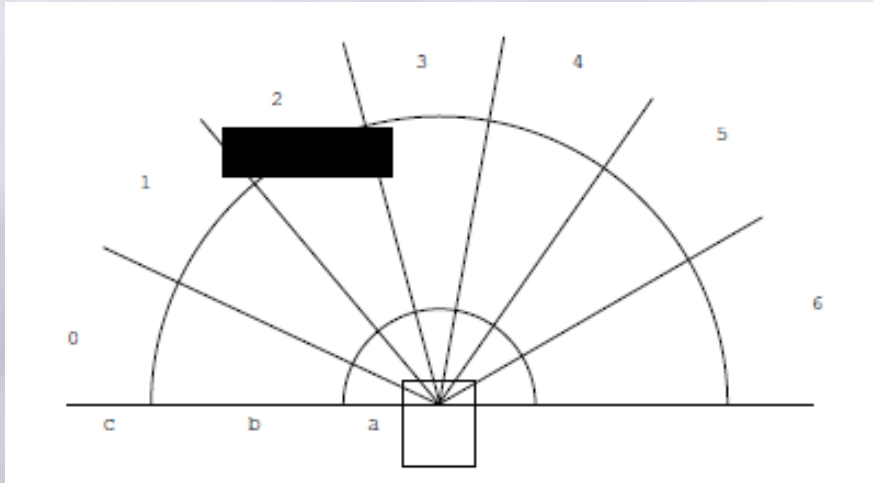
  $\rightarrow$ En pratique, α est choisi constant (adaptation permanente)



Exemple de mise en oeuvre

Apprendre à un robot à éviter des obstacles :

- télémètre laser et base mobile
- discrétisation de l'état : $|S| = 2187$
- 3 actions (avancer, tourner à droite, tourner à gauche)
- $r = -10$ si robot percute un obstacle (remis à son point de départ)
- $r = 3$ lorsque le robot choisit l'action d'avancer

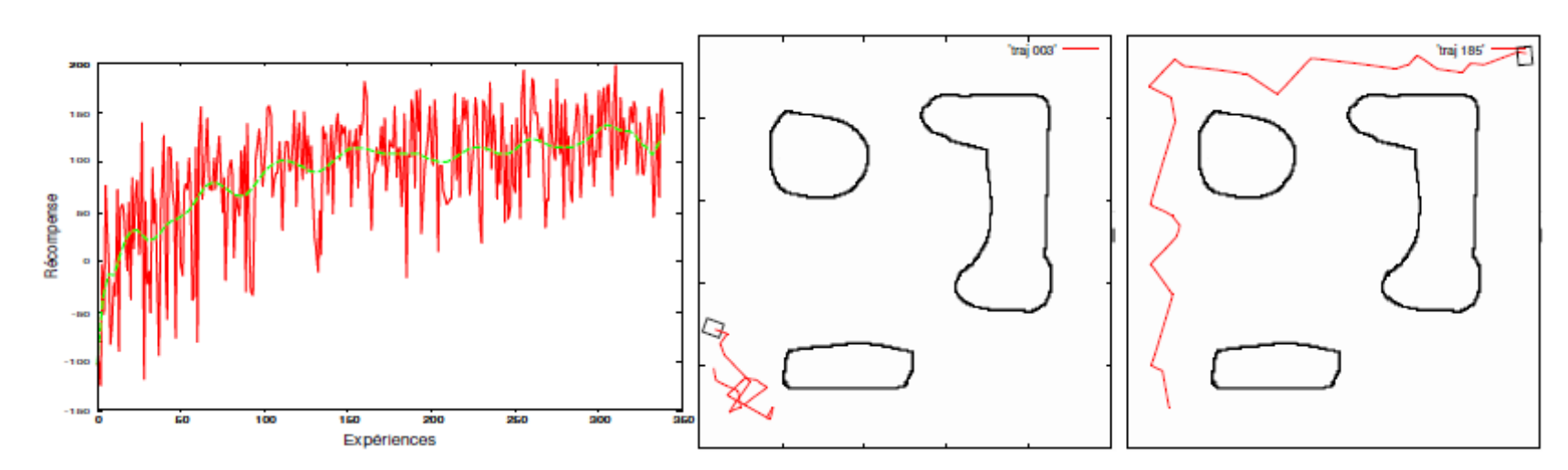


Discrétisation de l'espace d'états : 7 secteurs, pour chacun, O_i donne présence ou non d'obstacles dans les zones a et b ($O_i = 0$ si obstacle dans a, $O_i = 1$ si obstacle dans b, $O_i = 2$ sinon). L'état s est la valeur en base 3 fournie par les O_i : $s = \sum_k 3^k O_k$. Ici,

$$O_0 = O_1 = O_4 = O_5 = O_6 = 2, O_2 = O_3 = 1 \text{ donc } s = 2150.$$

Exemple de mise en oeuvre

Q-learning avec une table de taille 2187×3.



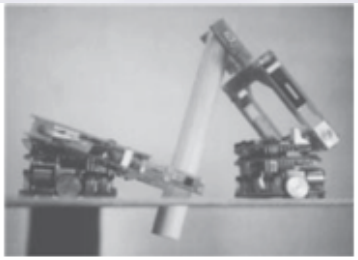
épisode initial ($k = 0$) épisode $k = 185$

Un épisode correspond à 100 étapes (actions).

Plan

Systèmes distribués de manipulation

[Hirata *et al.* 02]



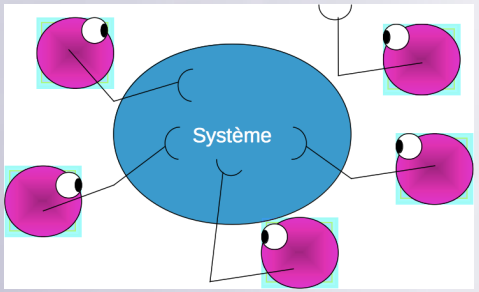
[Martinoli *et al.* 95]

[Adouane 05]

Plusieurs entités physiquement indépendantes

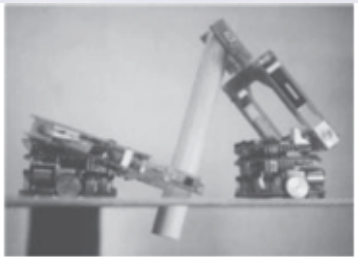
Systèmes Multi-Agent (SMA) :

- Regroupement d'entités
- En interaction
- Emergence d'un comportement collectif complexe



Systèmes distribués de manipulation

[Hirata *et al.* 02]



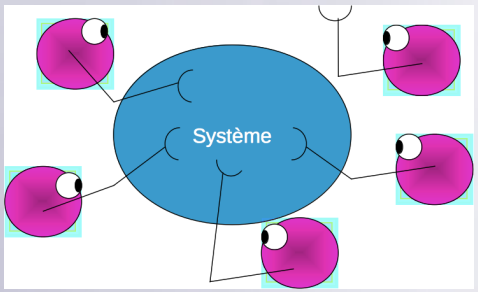
[Martinoli *et al.* 95]

[Adouane 05]

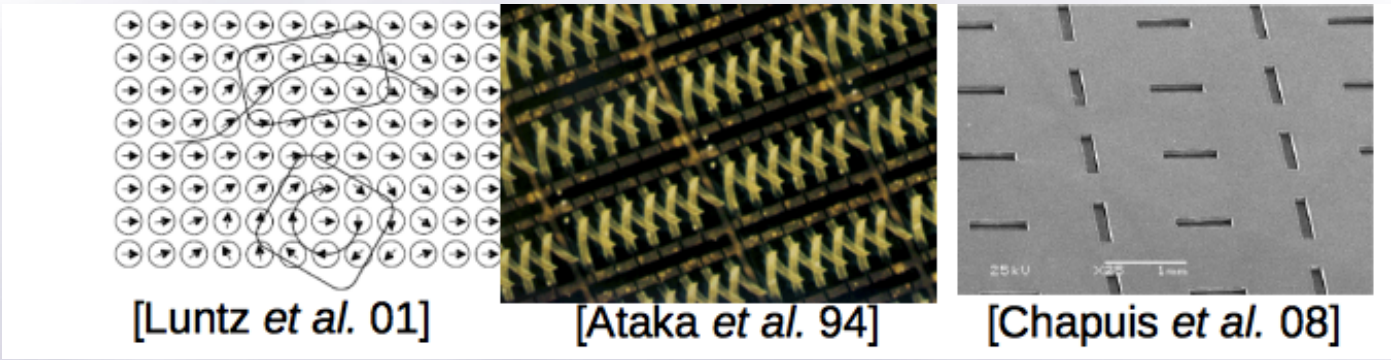
Plusieurs entités physiquement indépendantes

SMA avec situation de coopération :

- Objectifs compatibles mais ressources ou compétences d'un ou plusieurs agents insuffisantes
- Coordination des actions individuelles



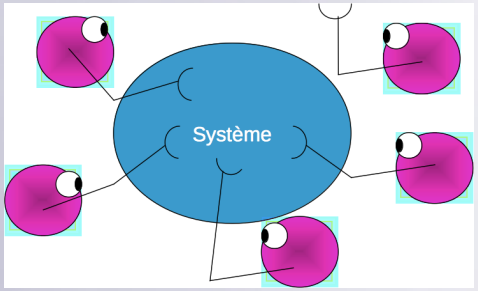
Systèmes distribués de manipulation



Matrice d'actionneurs = matrice d'agents

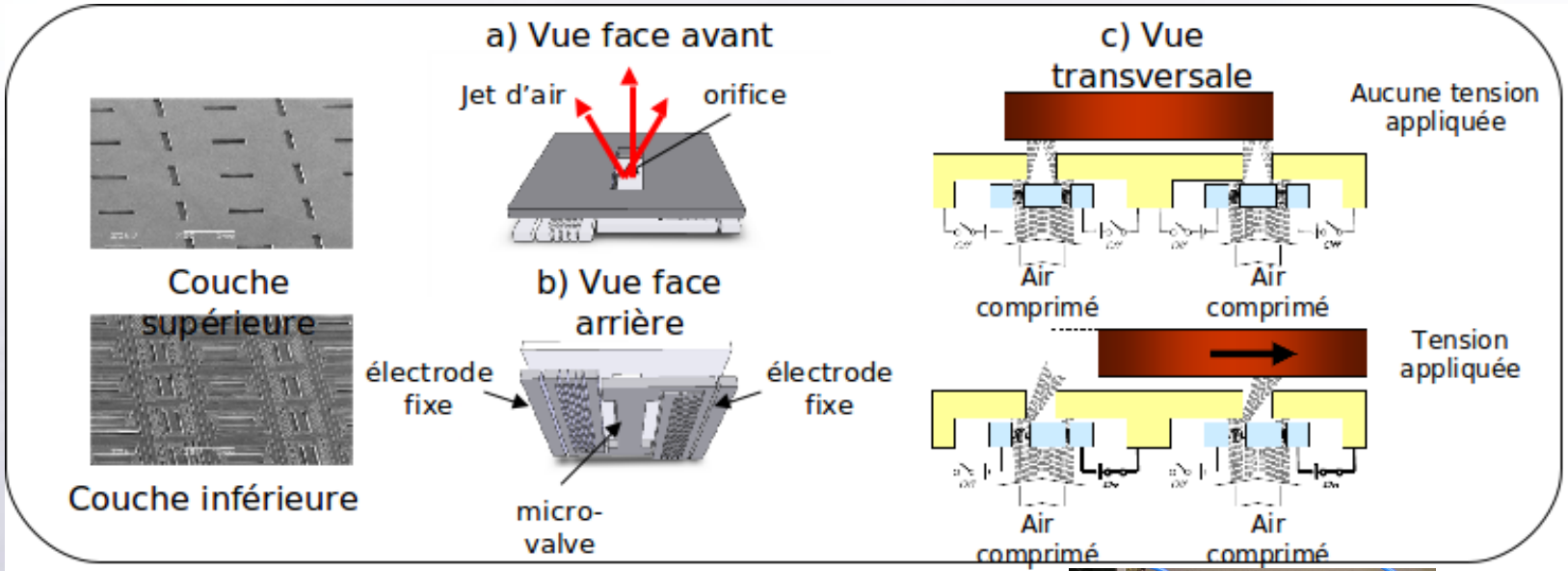
SMA avec situation de coopération :

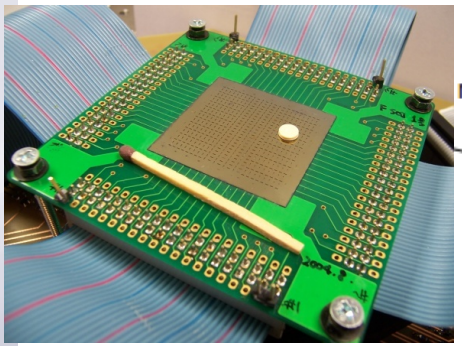
- Objectifs compatibles mais ressources ou compétences d'un ou plusieurs agents insuffisantes
- Coordination des actions individuelles



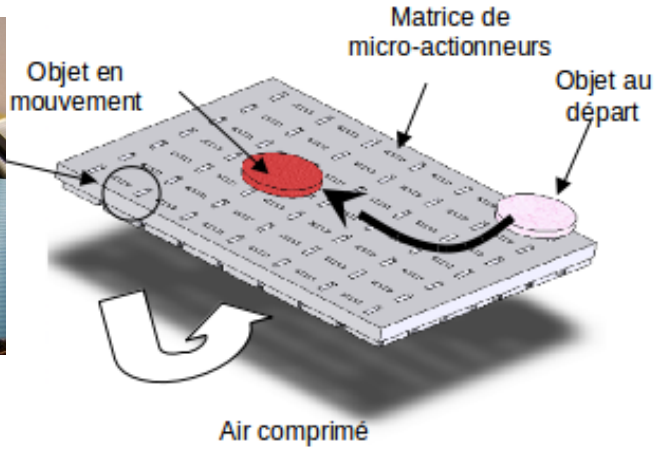
Contexte : projet ANR Smart surface

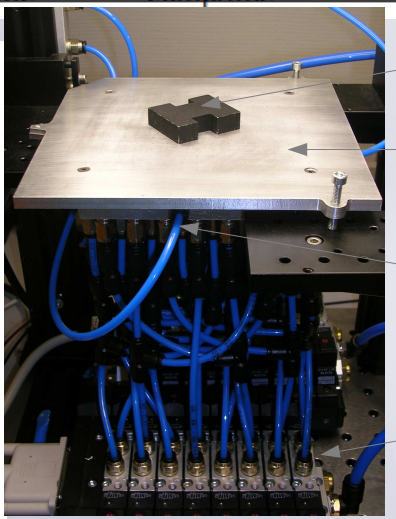
Surface active pneumatique [Fukuta *et al.* 06] [Chapuis *et al.* 08]





prototype micro de surface active



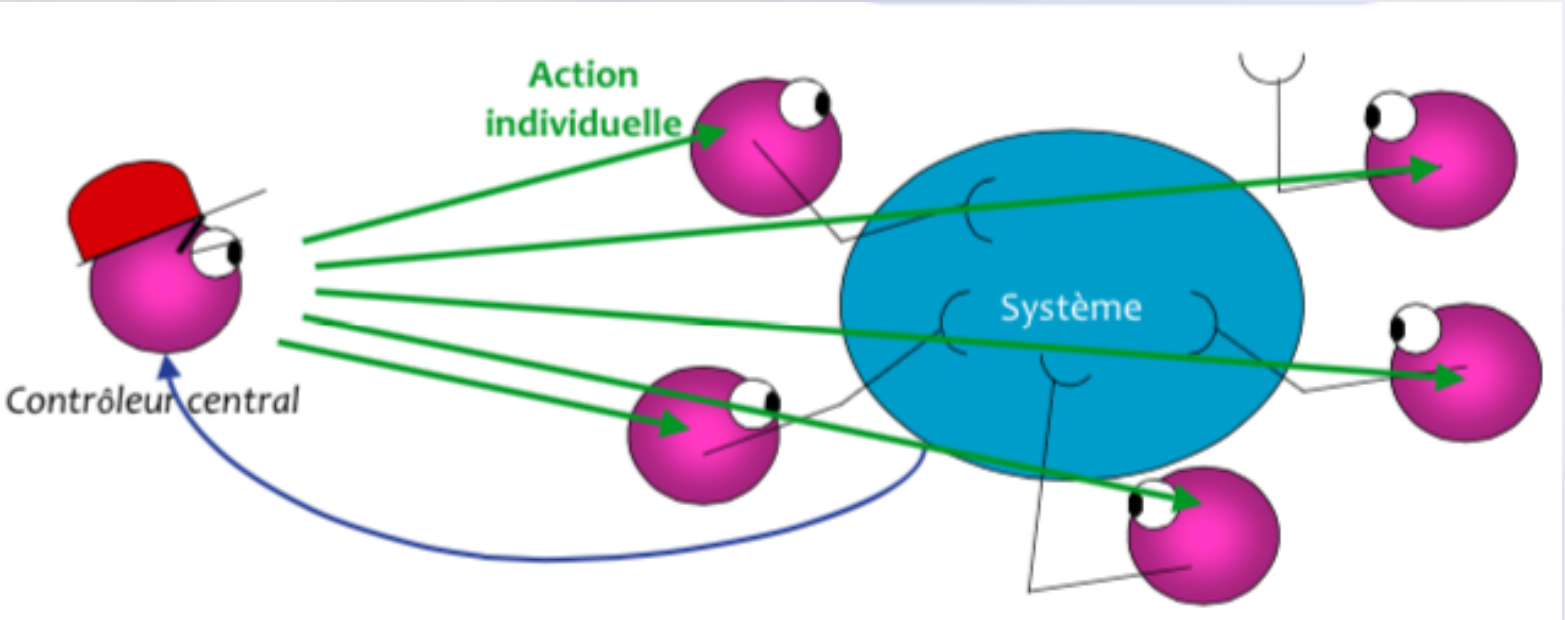


prototype macro de surface active

Systèmes distribués de manipulation

Architecture de commande centralisée

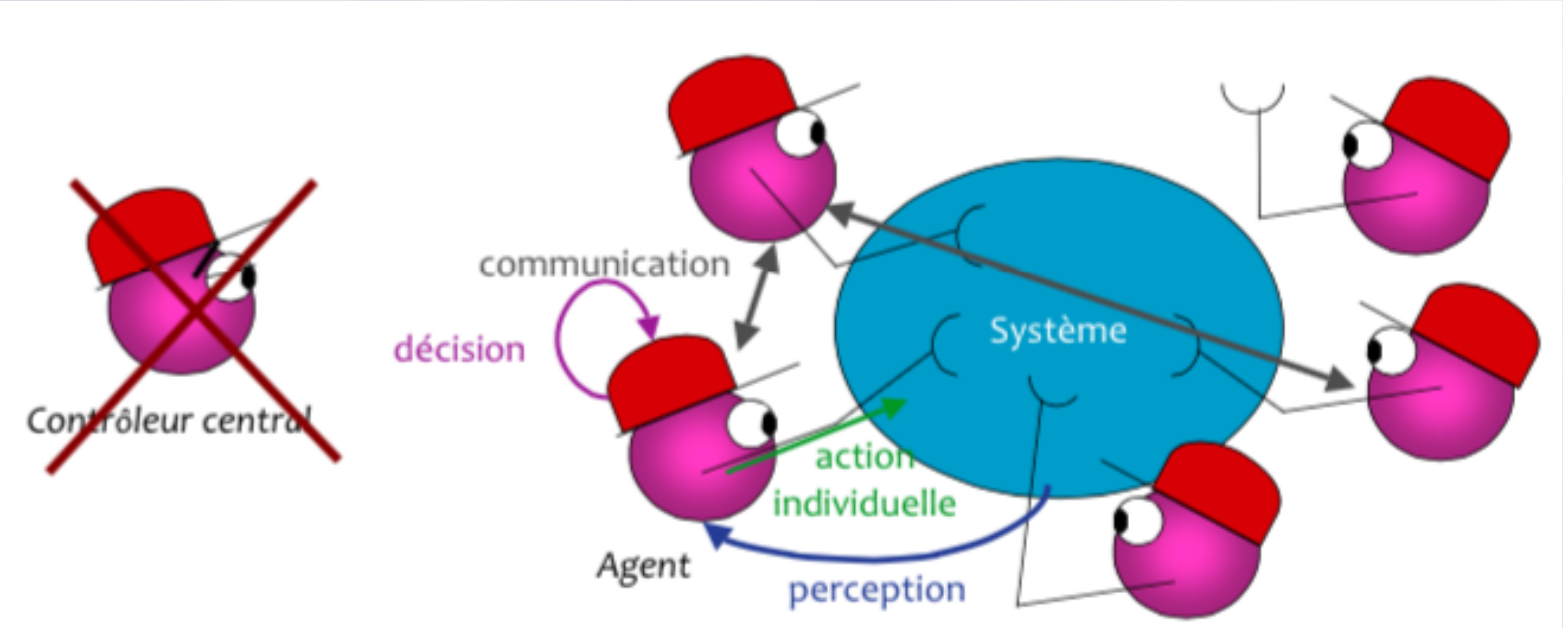
- Pas de difficultés de coordination
- Peu robuste
- Non modulable
- Beaucoup de communications



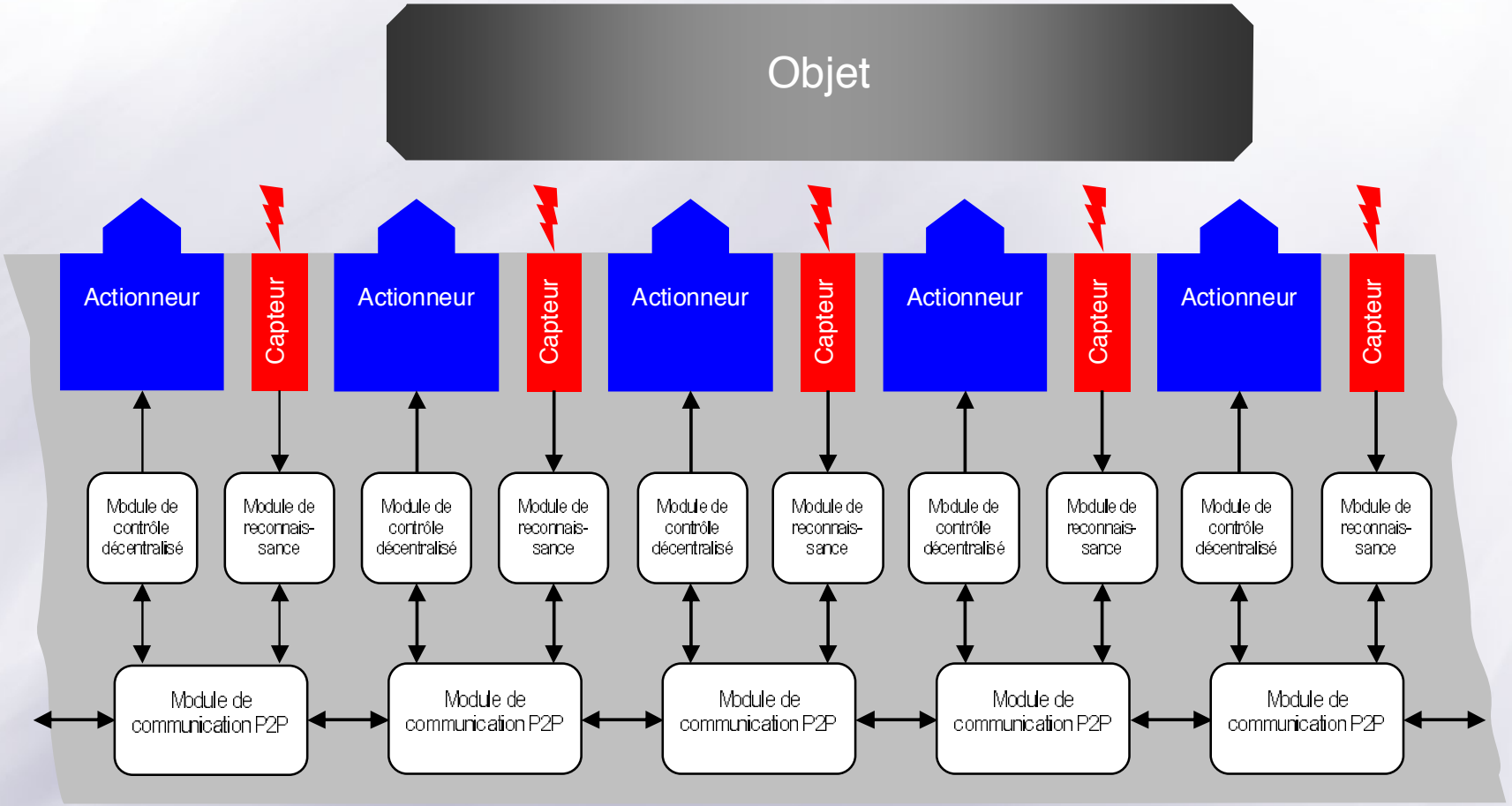
Systèmes distribués de manipulation

Architecture de commande décentralisée

- Agents autonomes
- Robuste
- Modulable
- Difficultés de coordination



Systèmes distribués de manipulation intégré

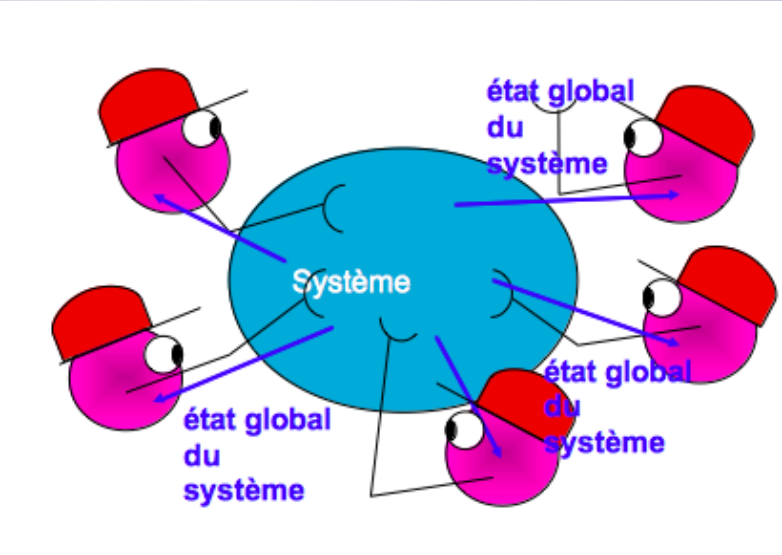


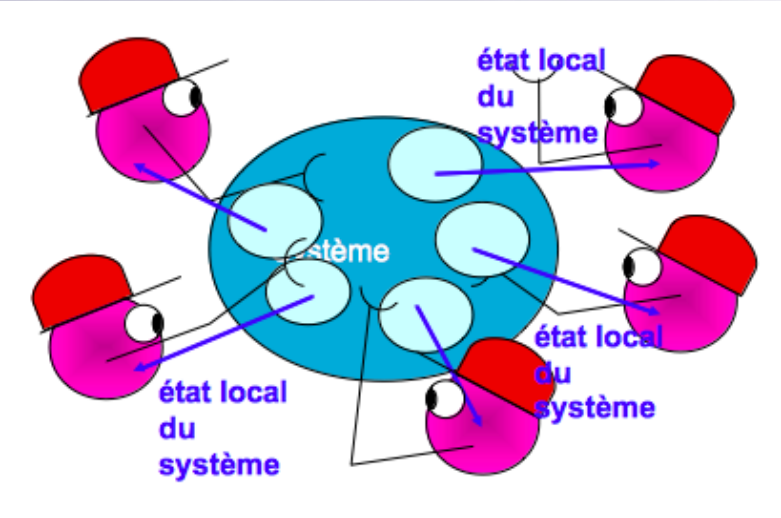
Systèmes distribués de manipulation

Quelle est la perception de chaque agent ?

Perception de l'état

- Observabilité individuelle totale : un agent perçoit l'état global du système à partir de sa perception individuelle
- Observabilité individuelle partielle : un agent ne connait pas l'état global du système à partir de sa perception individuelle

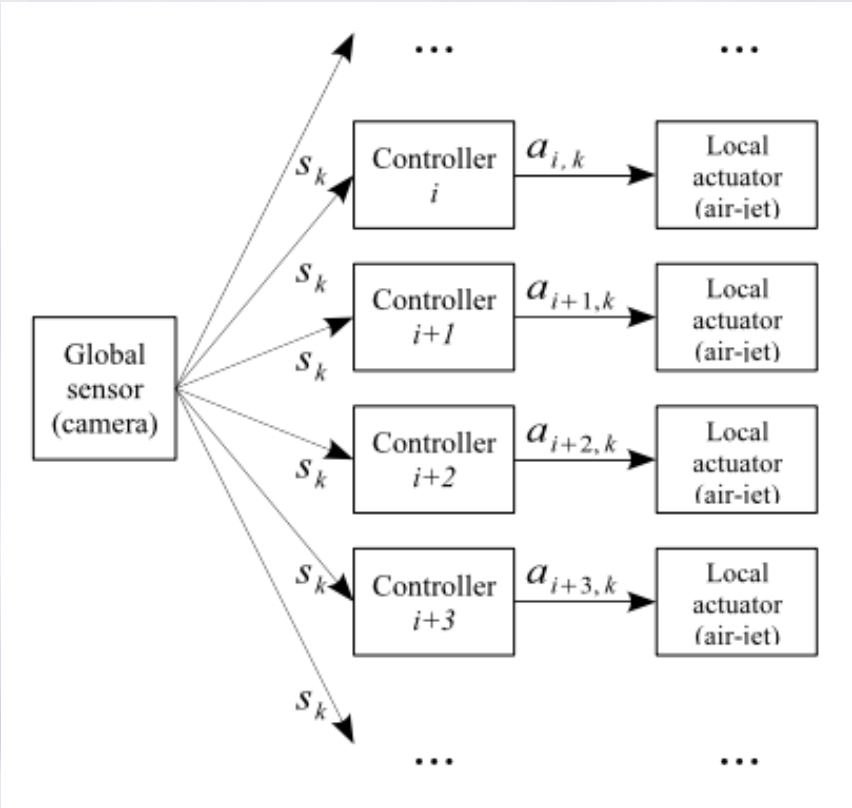


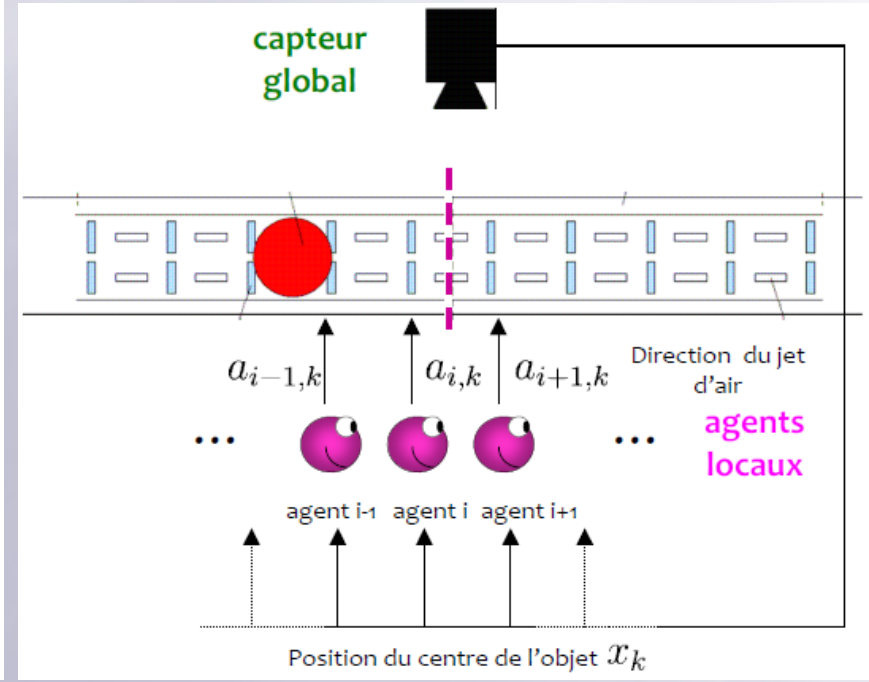


Systèmes distribués de manipulation

Observabilité individuelle totale :

- capteur global
- chaque agent perçoit la position de la pièce.

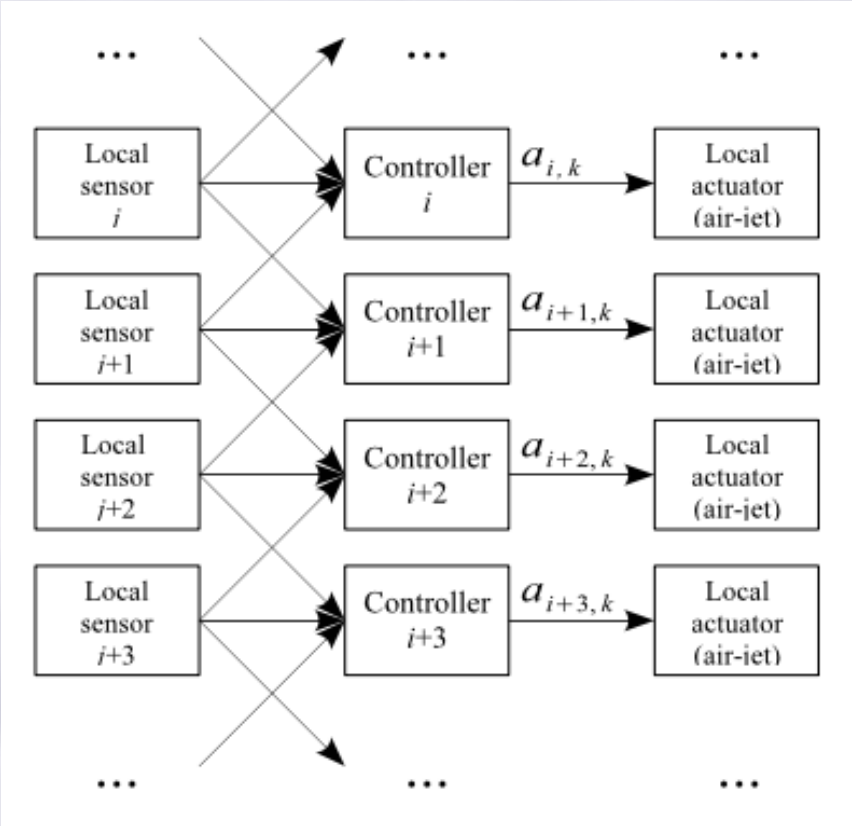


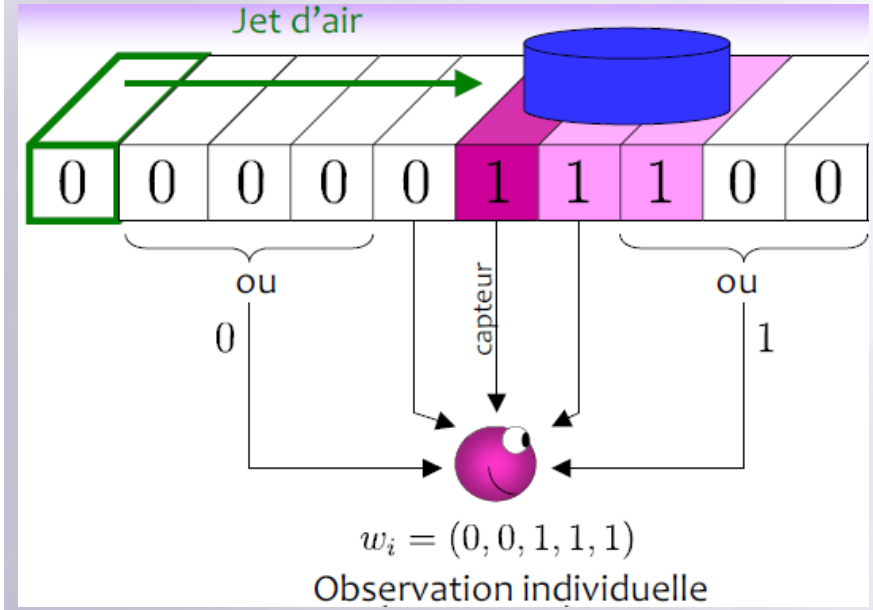


Systèmes distribués de manipulation

Observabilité individuelle partielle :

- capteur local
- chaque agent perçoit si la pièce est au-dessus de lui et de ses voisins





Systèmes distribués de manipulation

Quelle est la perception de chaque agent ?

Perception des actions

- Agent indépendant : n'a pas accès aux actions individuelles des autres
- Agent percevant l'action jointe : a accès aux actions individuelles des autres agents

Systèmes distribués de manipulation

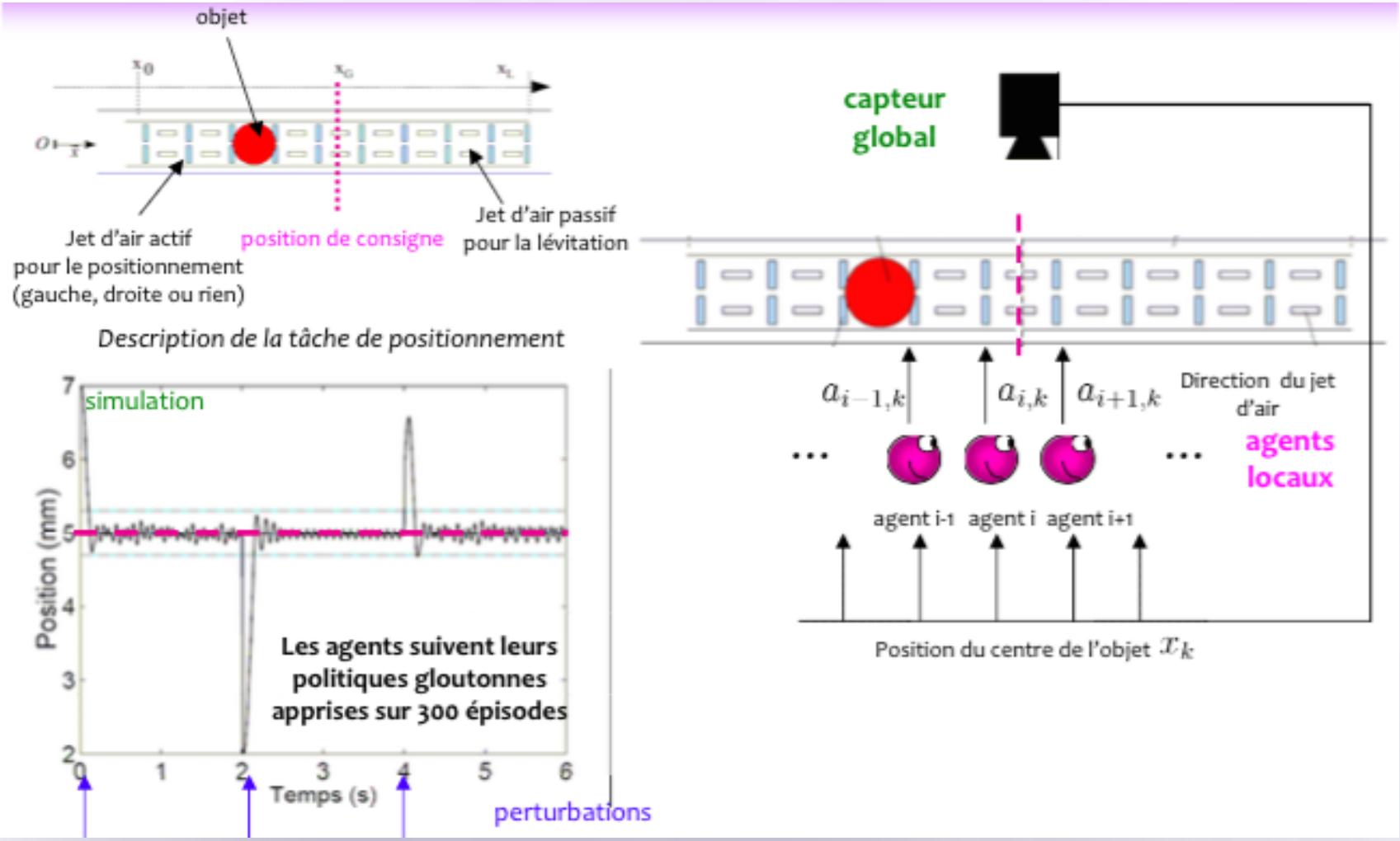
Nos hypothèses

- Agents coopératifs
- Agent indépendant : se coordonner sans savoir ce que les autres agents vont décider
- Observabilité individuelle totale : mais en observant l'effet de l'action jointe sur le système
- Dynamique du système inconnue

Comment des agents indépendants et coopératifs peuvent-ils apprendre à se coordonner ?

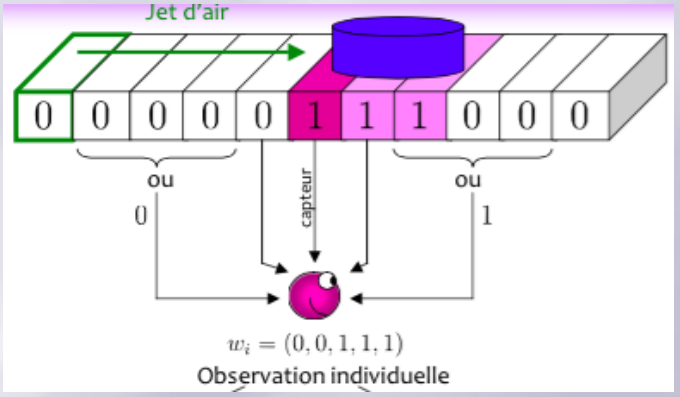
$\rightarrow$ Modèles décisionnels multi-agents issus des MDP et de la théorie des jeux

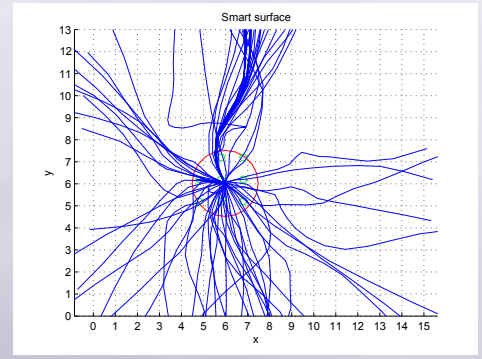
Contrôle adaptatif semi-décentralisé



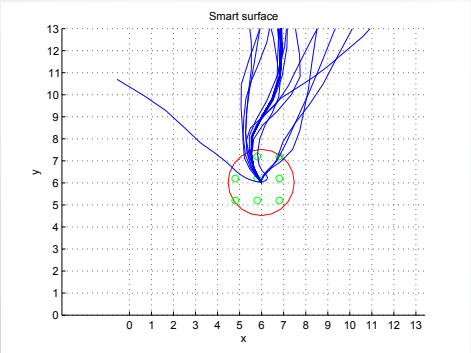
Contrôle adaptatif décentralisé

- point de vue local : observabilité individuelle partielle
- partage limité des informations capteurs : observabilité de + en + complète

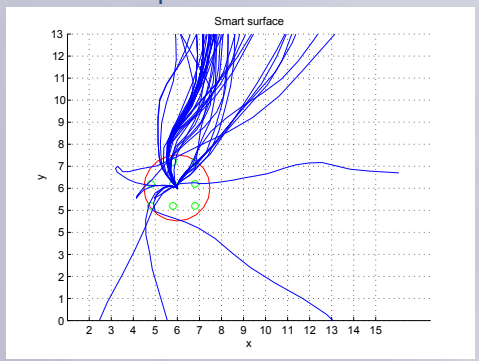




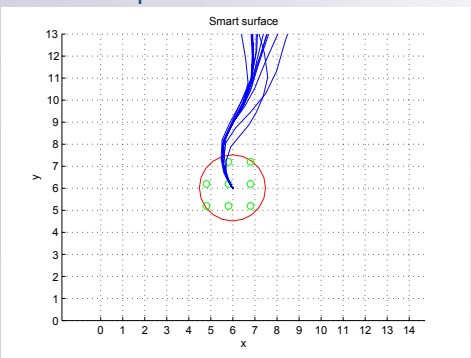
épisode 1 à 100



épisode 100 à 200



épisode 200 à 300



épisode 300 à 400

Plan

Problèmes

Les algorithmes décrits jusqu'ici sont des algorithmes dits tabulaires

- Ils supposent des espaces d'états et d'actions discrets et finis
- Ils stockent les Q-valeurs dans des tables

Espace d'états de grande dimension : *curse of dimensionality*

Quantité de données nécessaire pour un bon apprentissage augmente exponentiellement avec la dimension de S : explosion combinatoire de la taille de la table Q

- problème de lenteur de l'AR (trop d'états à visiter)
- problème d'espace mémoire

Problèmes

Les algorithmes décrits jusqu'ici sont des algorithmes dits tabulaires

- Ils supposent des espaces d'états et d'actions discrets et finis
- Ils stockent les Q-valeurs dans des tables

Espace d'états de grande dimension : *curse of dimensionality*

Quantité de données nécessaire pour un bon apprentissage augmente exponentiellement avec la dimension de S : explosion combinatoire de la taille de la table Q

- problème de lenteur de l'AR (trop d'états à visiter)
- problème d'espace mémoire

Solution

Qualité de la représentation

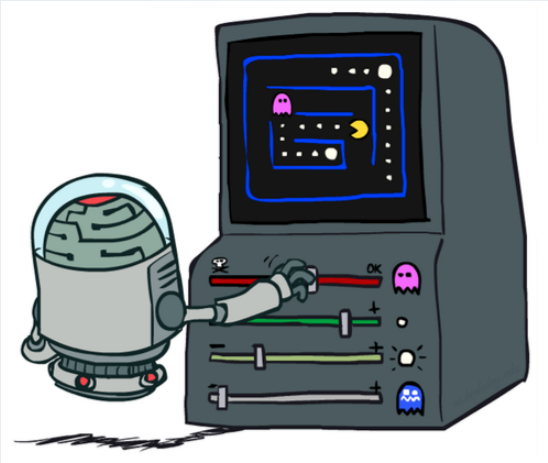
Utilisation de *features* ou **fonctions caractéristiques** expertes :

- chaque *feature* contient une information spécifique sur les données
- dimension du vecteur de features $\ll$ dimension de l'espace d'états
- choix des *features* de sorte à obtenir une politique pertinente

Solution : qualité de la représentation

Features experte :

Exemple Jeu Pac-Man



Agent Pac-Man qui apprend par renforcement :

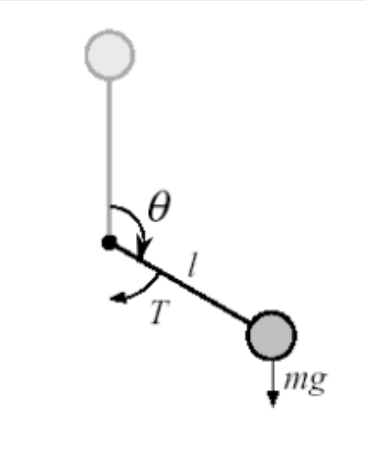
- information brute dans l'état du jeu contient beaucoup d'éléments (score, position des murs et pac-gommes, position et direction des agents et fantômes, ...)

- quelles informations conserver dans l'état du MDP pour faire fonctionner un Q-learning ? (cf. TP2 §2.2)
- la politique définie sur cet espace d'état doit permettre à l'agent de prendre une décision markovienne.

Solution : qualité de la représentation

Features experte :

Exemple du pendule inversé (discrétisation)



Espace d'états en 2D : $\mathbf{s} = (\theta, \dot{\theta})$
9 actions (*torque*)
$R(\mathbf{s}) = cos(\theta)$

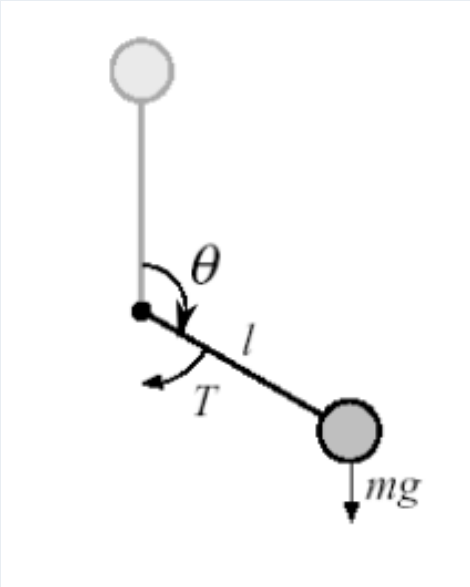
Comment choisir une discrétisation adaptée si espace perceptif continu ?

- si trop détaillée, la discrétisation sera computationnellement ingérable
- si trop faible, la discrétisation pourrait conduire à une politique non pertinente

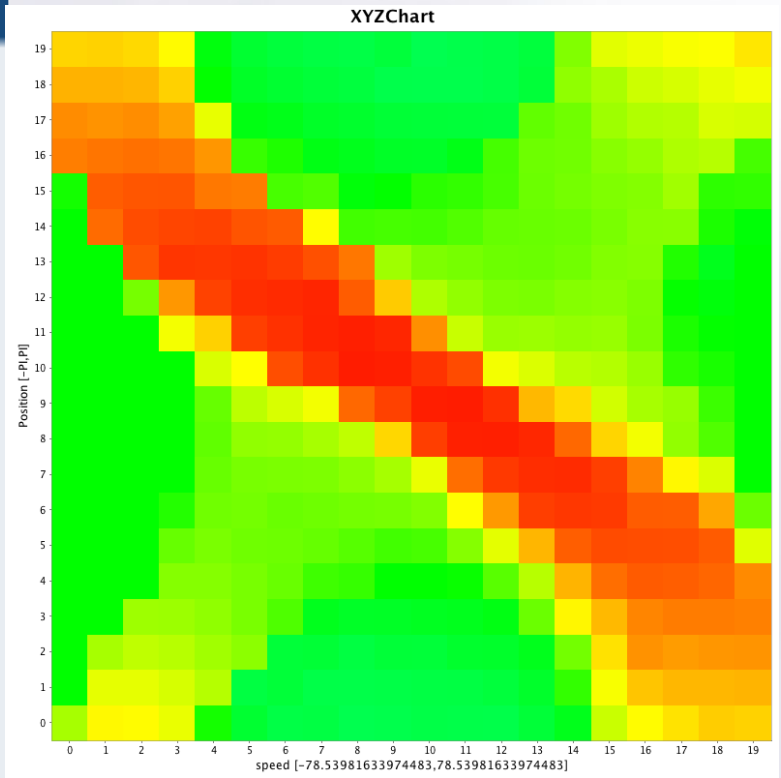
Solution : qualité de la représentation

Features experte :

Pendule inversé



Espace d'états en 2D :
$\mathbf{s} = (\theta, \dot{\theta})$
9 actions (*torque*)
$R(\mathbf{s}) = cos(\theta)$



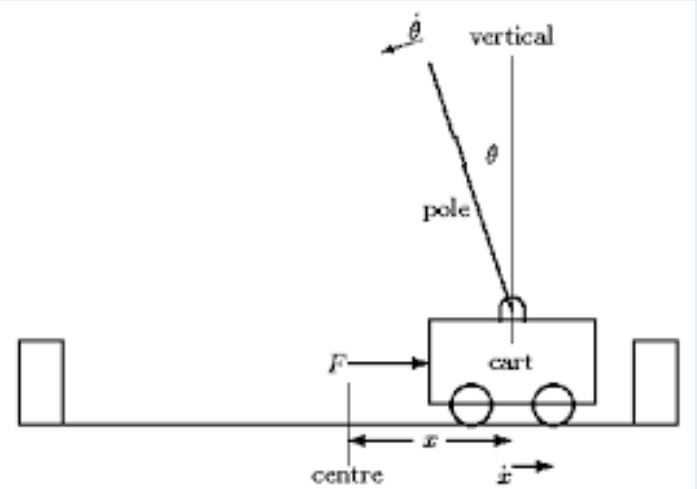
$V(s)$

Fonctionne avec un Q-learning et discrétisaiton uniforme de S (30×30)

Solution : qualité de la représentation

Features experte :

Pendule inversé sur chariot



Espace d'états en 4D : $\mathbf{s} = (x, \dot{x}, \theta, \dot{\theta})$

3 actions : $a = \{-F, 0, F\}$

$R(\mathbf{s}) = cos(\theta) - d$ avec d pénalité si proche du bord

Fonctionne avec un Q-learning et discrétisation non-uniforme de S :

- $x = \{-0.8, 0.8\}$; $\dot{x} = \{-0.5, 0.5\}$
- $\theta = \{-0.105, -0.017, 0, 0.017, 0.105\}$; $\dot{\theta} = \{-0.873, 0, 0.873\}$

Solution

Qualité de la représentation

Utilisation de *features*.

Généralisation entre états

- Apprendre par l'expérience sur un sous-espace de S
- Généraliser les expériences apprises à de nouvelles situations jamais rencontrées

Solution : Généralisation entre états

Exemple Jeu Pac-Man

On décrit l'état du MDP avec les positions 2D des éléments (pacman, fantômes, dots, ...).

L'agent apprend par l'expérience que cet état est mauvais

Avec le Q-learning classique, l'agent ne saura rien sur cet état

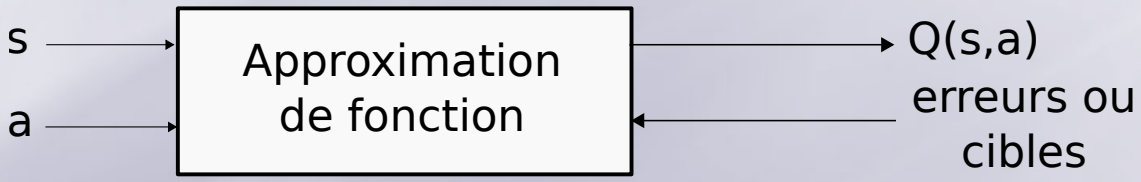
Ou même celui-ci

Solution : Généralisation entre états

Généralisation entre états

- Apprendre par l'expérience sur un sous-espace de S
- Généraliser les expériences apprises à de nouvelles situations jamais rencontrées

Problème classique en *Machine Learning* d'approximation de fonction.



- table
- **Approximation linéaire et vecteur de *features* expertes** : Approximate Q-learning (cf. TP2 §2.3)
- ...

Solution : Généralisation entre états

Approximate Q-learning

- La fonction de valeur Q est approximée par une fonction linéaire de n fonctions caractéristiques f_i :

$$Q_w(s,a) = \sum_i w_i f_i(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + ... + w_n f_n(s,a)$$

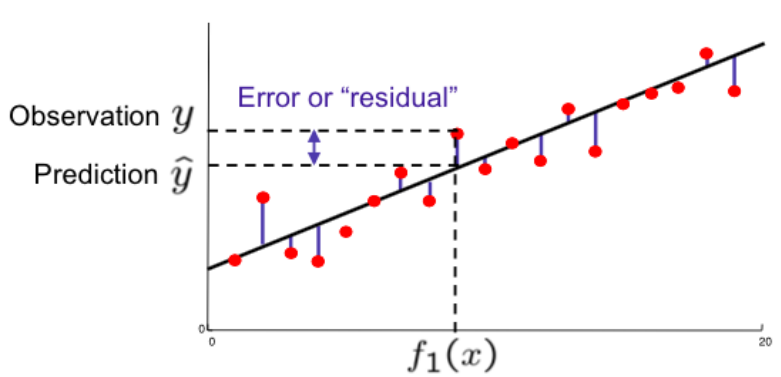
- Les *features* $f_i : S \times A \to R$ sont choisies (expertes)
- L'agent apprend les poids w_i permettant d'avoir la meilleure approximation de la fonction Q

Approximate Q-learning : exemple sur le Pacman

On décrit un état en utilisant un vecteur de caractéristiques, composés de fonctions caractéristiques f_i :

- f_i associe une valeur à tout état ou état-action
- f_i capture une ou plusieurs propriétés importantes d'un état :
  - Distance au fantôme le plus proche f_{GST}
  - Nombre de fantômes
  - 1/Distance au pac-gomme le plus proche f_{DOT}
  - 1 si une action peut amener à être dévoré
  - ...

Solution : Généralisation entre états



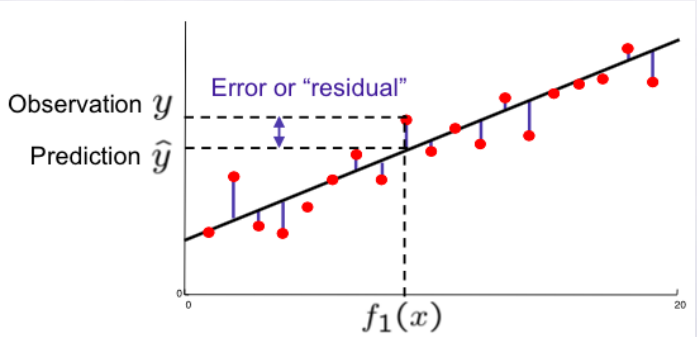
Approximate Q-learning

Apprentissage supervisé pour approximer la fonction Q : à chaque nouvelle étape (s, a, s', r) :

- la prédiction pour la valeur de (s, a) est $\hat{y} = Q_w(s, a) = \sum_i w_i f_i(s, a)$
- la valeur cible (estimée par amorçage) de (s, a) est $y = r + \gamma \max_b Q_w(s', b)$

Trouver le vecteur de poids $\vec{w}$ qui minimise une fonction de l'erreur entre prédiction et valeur cible.

Rappels : descente de gradient et approximation linéaire



- Ensemble d'échantillons (x_k, y_k)
- Prédiction $\hat{y}_k = w_0 + w_1 * x_k$
- $E(\vec{w})$ est l'erreur sur un échantillon d'exemples

$$E(\vec{w}) = 1/2 \sum_k (y_k - \hat{y}_k)^2$$

- Objectif : Trouver $\vec{w}$ qui minimise $E(\vec{w})$

- Trouver le vecteur de poids $\vec{w}$ qui minimise $E(\vec{w})$.
- Descente de gradient : algorithme itératif pour trouver le minimum local d'une fonction.

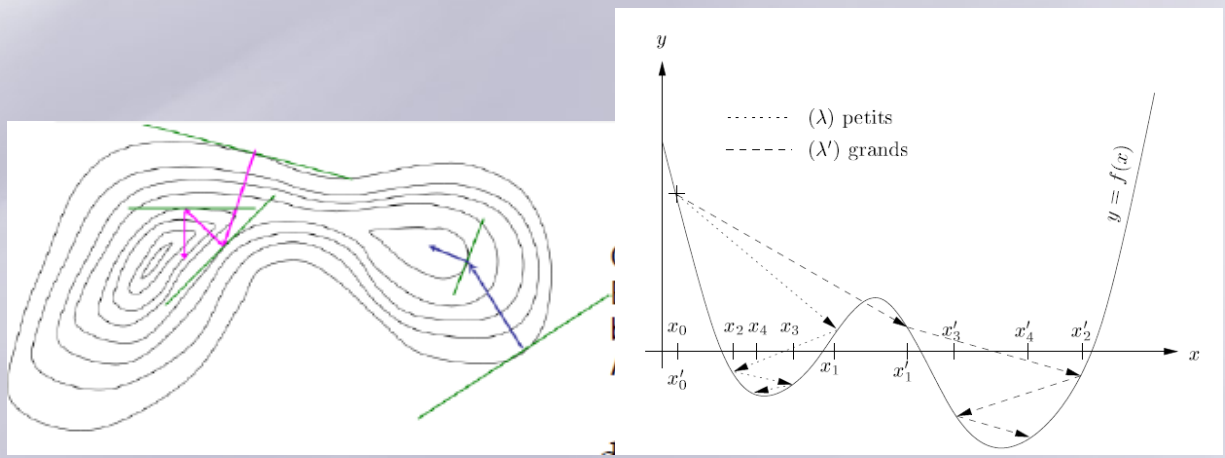
Rappels : descente de gradient

Principe :

- A t on part du point $\vec{x_t}$,
- on suit le vecteur donné par le gradient de f en $\vec{x_t}$ noté $\vec{\bigtriangledown} f(\vec{x_t})$
- Le gradient indique la direction de plus grande pente de f
- $\vec{\bigtriangledown} f(x_1, ..., x_K) = (\frac{\partial f(x_1,...,x_K)}{\partial x_1}, \frac{\partial f(x_1,...,x_K)}{\partial x_2}, ...)^T$
- On passe à

$$\vec{x}_{t+1} = \vec{x_t} - \lambda \vec{\bigtriangledown} f(\vec{x_t})$$

$\lambda > 0$ pas d'apprentissage permettant de progresser rapidement au début (en sortant si possible de zones d'optima locaux) puis à affiner le résultat par des pas plus petits.



Approximate Q-learning

Pour une étape (s, a, s', r), $Q_w(s, a) = \sum_i w_i f_i(s, a)$,

- erreur sur l'exemple (y cible, $\hat{y}$ prédiction)

$$E(\vec{w}) = 1/2(y - \hat{y})^2 = 1/2(y - \sum_i w_i f_i(s, a))^2$$

- gradient

$$\frac{\partial E(\vec{w})}{\partial w_m} = -(y - \sum_i w_i f_i(s, a)) f_m(s, a)$$

- mise à jour de chaque poids :

$$w_m \leftarrow w_m - \alpha \frac{\partial E(\vec{w})}{w_m}$$
$$w_m \leftarrow w_m + \alpha(y - \hat{y}) f_m(s, a)$$
$$w_m \leftarrow w_m + \alpha(r + \gamma \max_b Q(s', b) - Q(s, a)) f_m(s, a)$$

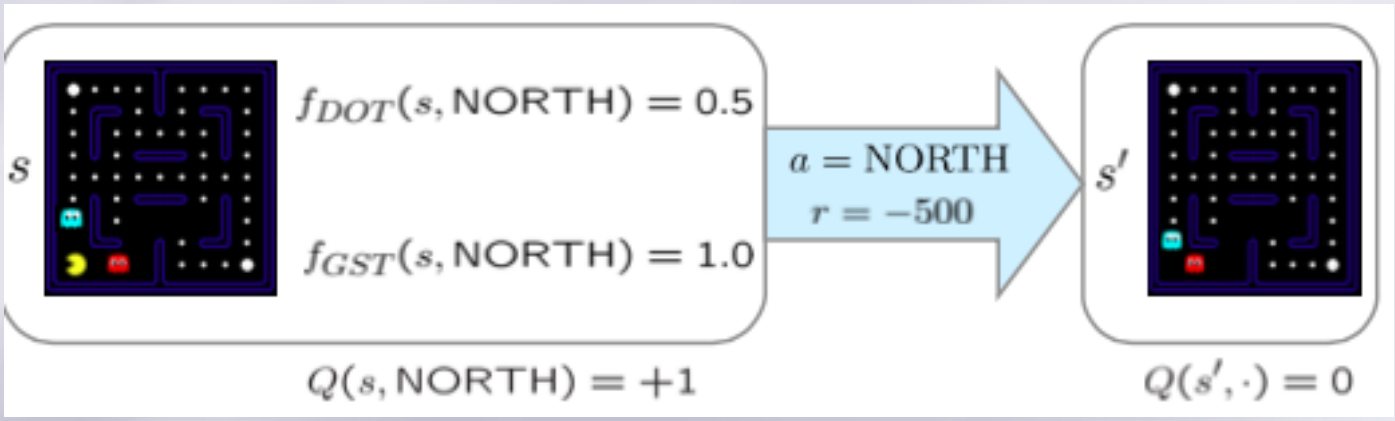
- Interprétation : si mauvaise prédiction, ajustement des poids.

Approximate Q-learning : exemple sur le Pacman

Après chaque étape (s, a, s', r) :

$$w_m \leftarrow w_m + \alpha(r + \gamma \max_b Q(s', b) - Q(s, a)) f_m(s, a)$$

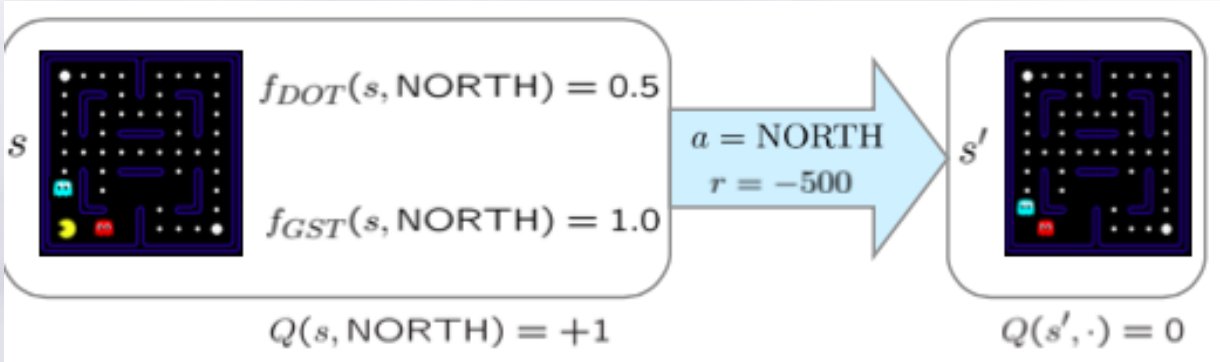
$$Q(s, a) = 4.0 f_{DOT}(s, a) - 1.0 f_{GST}(s, a)$$



Que vaut $Q(s, a)$ après la mise à jour ($\alpha = 0.004$) ?

Approximate Q-learning : exemple sur le Pacman

$$Q(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$$



$$\delta = r + \gamma \max_b Q(s', b) - Q(s,a) = -500 + 0 + 1$$

$$w_{DOT} = 4.0 + \alpha \delta f_{DOT} = 4.0 + 0.004 \times -501 \times 0.5 = 3$$

$$w_{GST} = -1.0 + \alpha \delta f_{GST} = -1.0 + 0.004 \times -501 \times 1.0 = -3$$

$$Q(s,a) = 3.0 f_{DOT}(s,a) - 3.0 f_{GST}(s,a)$$

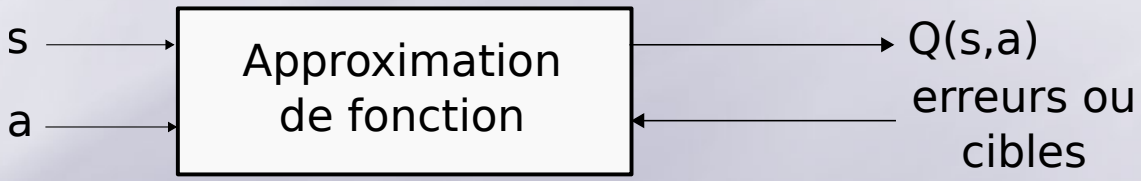
Solution

Qualité de la représentation

Utilisation de *features*.

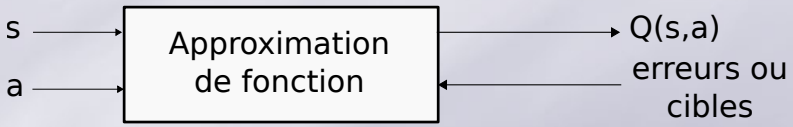
Généralisation entre états

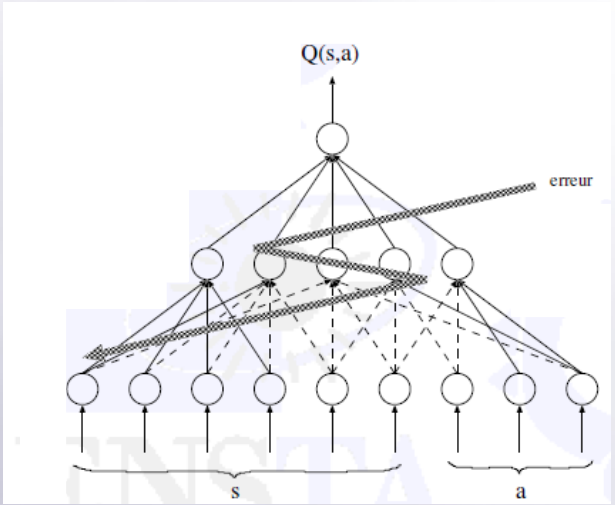
Problème classique en *Machine Learning* d'approximation de fonction.



- table
- Approximation linéaire et vecteur de *features* expertes : Approximate Q-learning (cf. TP2 §2.3)
- **Approximation non linéaire (réseaux de neurones) : deep RL** (cf. cours M2 IA en IBI)
- ...

Approximation de la fonction Q avec NN



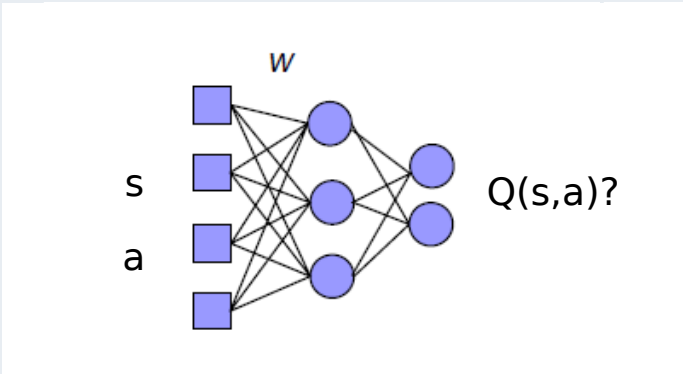


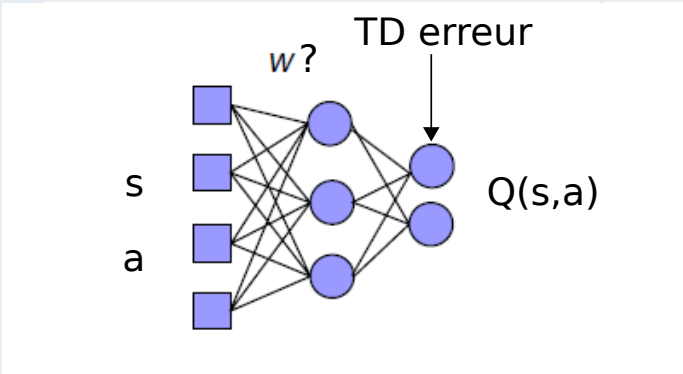
- *features* non-expertes : elles sont « apprises »par le réseau de neurones
- Q paramétrée par un vecteur de paramètres $\vec{\omega}$ qui sont les poids du NN : $Q(s,a) = f(s,a,\vec{\omega})$
- On fait varier Q en modifiant $\vec{\omega}$

Approximation de la fonction Q avec NN

Q-learning avec réseau de neurones

- Raisonnement = Activation :
  - Obtenir une estimation de Q pour une action a étant donné un état s
  - $\rightarrow$ Propagation vers l'avant
- Apprentissage = Retro propagation :
  - Déterminer les poids du réseau pour un état s, une action a et sa valeur cible
  - $\rightarrow$ Propagation vers l'arrière





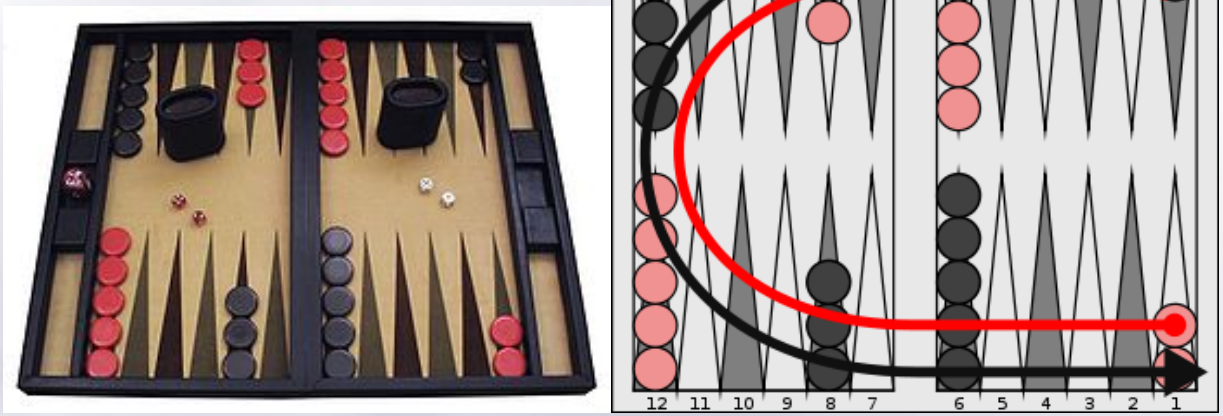
Retro propagation = Ajustement successifs des paramètres dans la direction qui réduira l'erreur TD : $\delta = r + \gamma \max_b Q(s', b) - Q(s, a)$

$$\vec{\omega}_{t+1} \leftarrow \vec{\omega}_t + \alpha \delta \Delta_{\vec{\omega}_t} Q_t(s_t, a_t)$$

$\Delta_{\vec{\omega}_t} f(x) = (\frac{\partial f(x)}{\partial \omega_t(1)}, \frac{\partial f(x)}{\partial \omega_t(2)}, \ldots)^T$ gradient

Plan

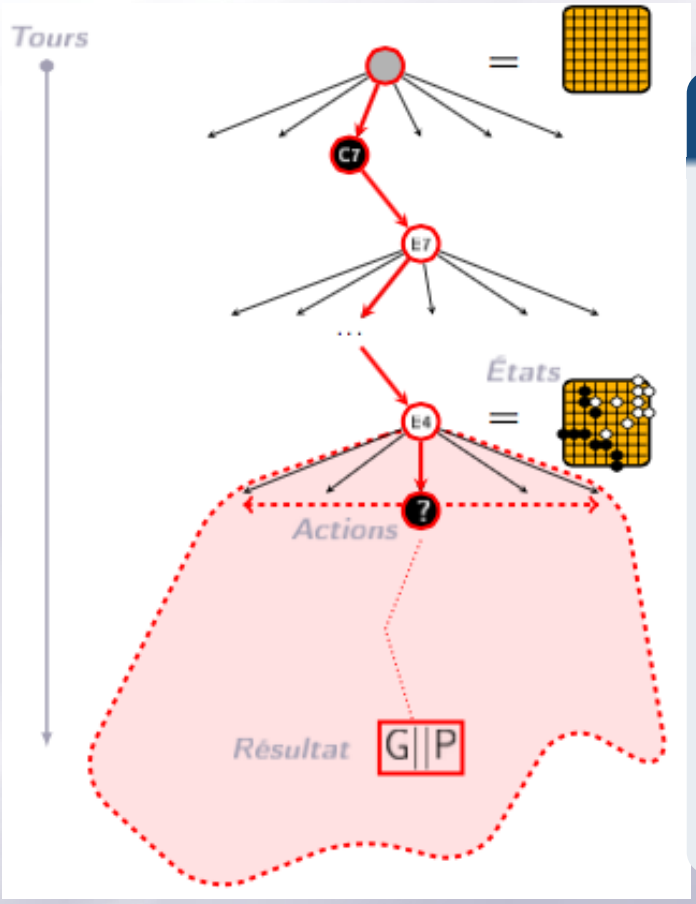
TD-Gammon [Tesauro, 1992]



Backgammon

- jeu à 2 joueurs sur plateau, avec dés
- 2 ensembles de 15 dames
- joueur gagne lorsqu'il sort toutes ses dames du plateau (pour cela, il faut amener toutes ses dames dans le dernier quadran)
- à chaque jet de dés, le joueur choisit parmi un grand nombre d'options pour déplacer ses dames (20 en moyenne)

TD-Gammon [Tesauro, 1992]



Backgammon : Difficultés

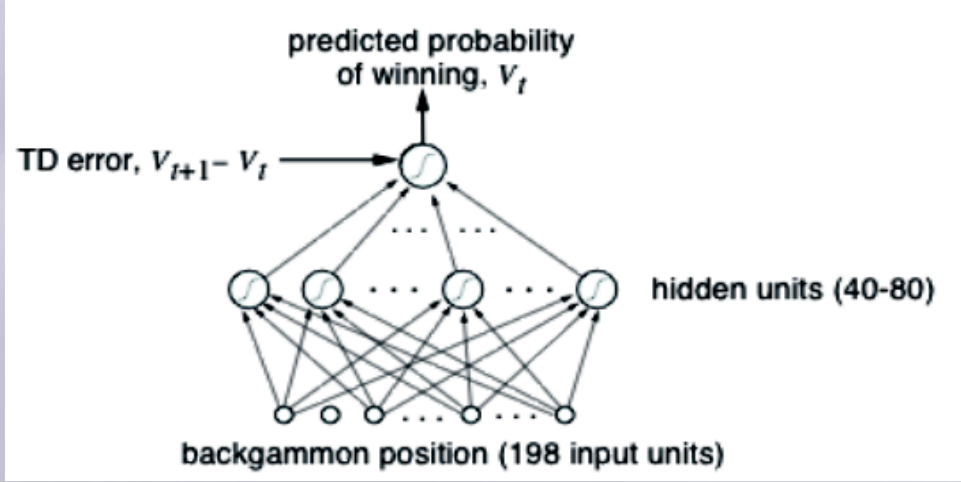
- Grand nombre de positions sur le plateau (30 pièces et 26 cases)
  $\rightarrow$ exclut les méthodes tabulaires
- Beaucoup d'options pour chaque jet de dés
- Pour considérer les mouvements futurs de l'adversaire, il faut considérer les jets de dés possibles.
  $\rightarrow$ Facteur de branchement d'environ 400

TD-Gammon [Tesauro, 1992]

AR appliqué au Backgammon

- TD(λ) avec retro propagation dans un réseau de neurones
- une couche d'entrée avec 198 entrées pour coder des informations sur la position des dames sur le plateau
- une couche cachée de 40 neurones
- une couche de sortie avec 3 valeurs pour estimer la probabilité de gagner de façon ordinaire, gammon ou backgammon.

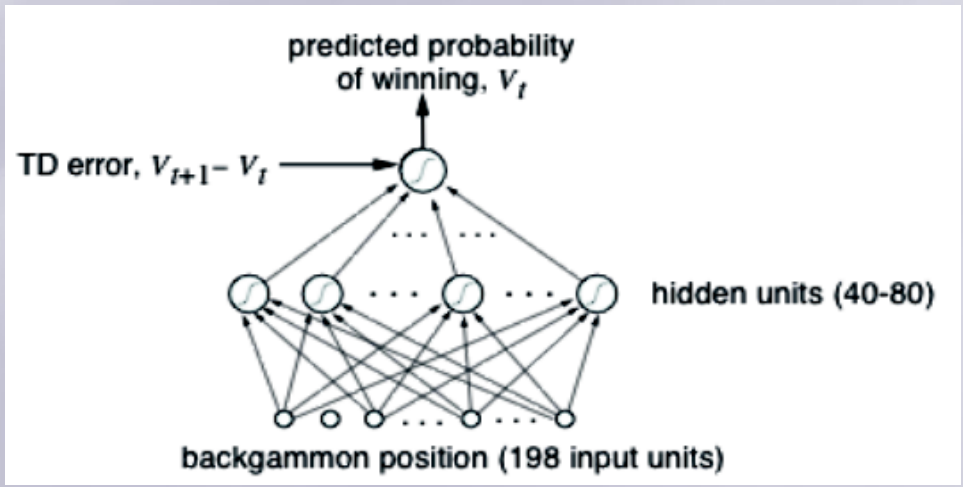
$V_t(s)$ utilisé pour estimer la probabilité de gagner depuis s.



TD-Gammon [Tesauro, 1992]

AR appliqué au Backgammon

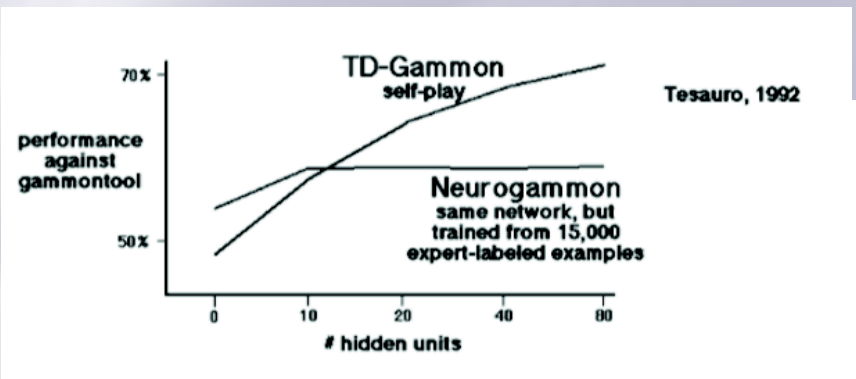
- TD(λ) avec retro propagation dans un réseau de neurones
- récompenses immédiates : +100 si gagne, -100 si perd, 0 sinon
- après chaque jet de dés, le réseau évalue toutes les positions possibles depuis la position courante. Celle de plus forte valeur est choisie. La récompense reçue est alors rétro-propagée dans le réseau.
- Apprend en jouant contre soi-même 300000 parties
  $\rightarrow$ joue aussi bien que le meilleur joueur du monde



TD-Gammon [Tesauro, 1992]

Plusieurs versions

- Neurogammon : apprentissage supervisé sur corpus expert + informations caractéristiques en entrées
- TD Gammon 0.0 : apprentissage contre soi-même en mode TD
- TD Gammon 1.0 : apprentissage contre soi-même en mode TD + informations caractéristiques en entrées
- TD Gammon 2 : *2-ply search* : prise en compte des mouvements possibles de l'adversaire
- TD Gammon 3 : *3-ply search*



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

Jeux Atari : DeepMind's Deep RL

https://www.youtube.com/watch?v=V1eYniJ0Rnk

Human-level control through deep reinforcement learning, Nature, 2015.
https://storage.googleapis.com/deepmind-data/assets/papers/DeepMindNature14236Paper.pdf

Robocup soccer

Neural Fitted Q-learning [Stone et al., Riedmiller et al., 2004]

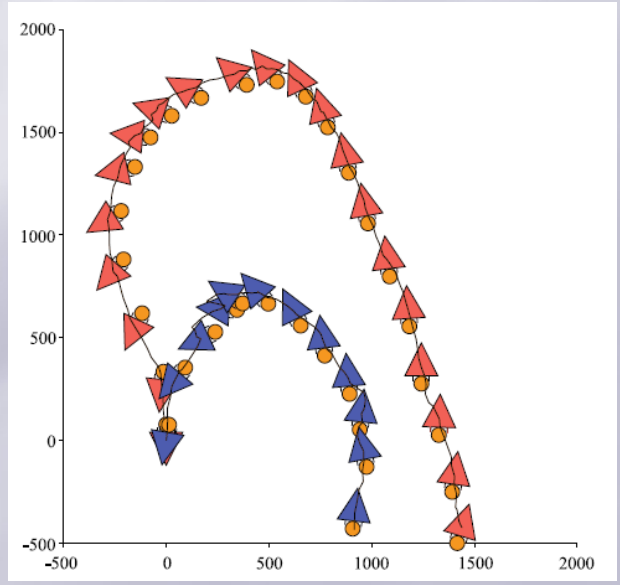
- Avantages de neural Q-learning : généralisation pour accélérer l'apprentissage
- Inconvénients de neural Q-learning : apprentissage "global" implique modifications incontrôlées de valeurs pour des couples (s, a) qui ne sont pas ceux pour lesquels on réalise la mise à jour $\rightarrow$ apprentissage peut être long
- *fitted Q Iteration* [Enrst,2005] : Enregistrer et réutiliser les transitions des expériences précédentes à chaque mise à jour de Q (Batch RL)
- *Neural Fitted Q Iteration* : utilise perceptron multi-couches pour Q

Robocup soccer

Neural Fitted Q-learning [Stone et al., Riedmiller et al., 2004]

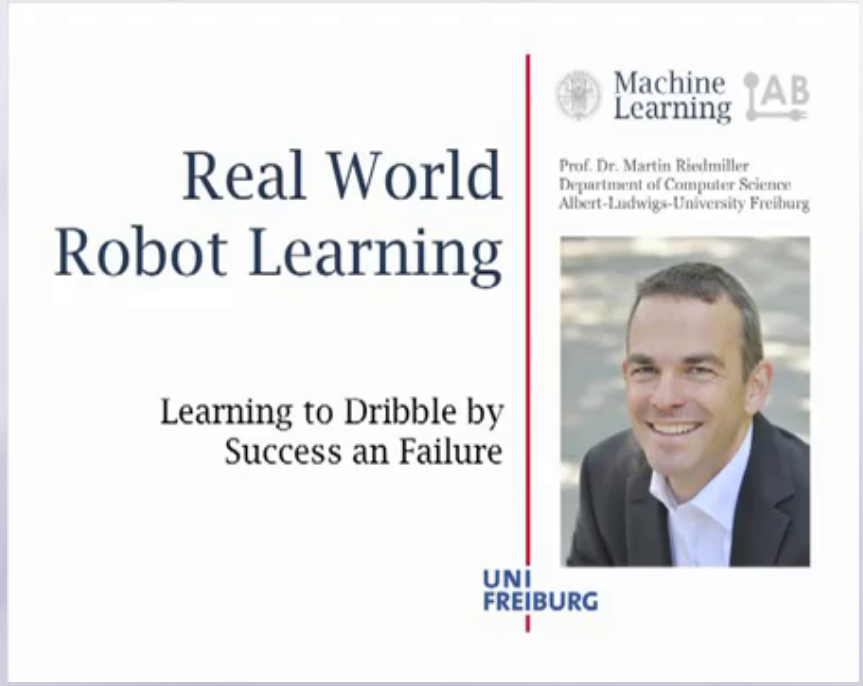
Premier comportement complètement appris sur le robot réel par équipe Brainstormers Tribots (Machine Learning Lab, University of Freiburg) :

- Apprendre à dribbler : MLP avec 9 entrées (6 pour l'état, 3 pour les actions), 2 couches cachées de 20 neurones chacune, un neurone de sortie. Phase d'apprentissage d'environ 1h30.



hand-coded vs. neural FQ

Robocup soccer



Robocup soccer

	'00	'01	'02	'03	'04	'05	'06	'07	'08
Simulation League									
NeuroKick	●	●	●	●	●	●	●	●	●
NeuroIntercept	●	●	●	●		○	○		
NeuroGo2Pos	●	●	●	●	●				
NeuroADB								●	●
NeuroAttack	○	●	●	●	●		●	●	●
NeuroPenalty				●	●	●	●	●	●
Rank	2	2	3	3	2	1	2	1	1
MidSize League									
NeuroMotorSpeed								○	○
NeuroGo2Pos							○	○	○
LmapIntercept							●	●	●
NeuroDribble								●	●
Rank							1	1	3

Comportements appris par AR neuronal dans l'équipe Brainstormers Tribots

Recherche directe de politique

Une autre solution pour traiter des problèmes de grandes tailles ou continus est d'effectuer une recherche directe de politique :

- Apprendre une politique qui maximise les récompenses plutôt qu'une fonction de valeur
- la politique prend la forme d'un contrôleur paramétré qui définit un sous-espace de l'espace des politiques stochastiques :

$$\pi = h(\vec{\theta})$$

  $\vec{\theta}$ vecteur de paramètres.
- le choix du contrôleur permet de faire un compromis entre l'expression d'une grande variété de π et une taille mémoire réduite
- problème d'optimisation des paramètres du contrôleur
- Intuition :
  - l'algorithme modifie itérativement les paramètres du contrôleur pour obtenir N contrôleurs "voisins" ;
  - la performance de ces nouveaux contrôleurs est évaluée en ligne

Recherche directe de politique

[Kohl and Stone, ICRA 2004]

[Ng. *et al.*, 2004]