Module : Apprentissage Automatique

L'apprentissage par Renforcement

Mr. Lakhmissi CHERROUN

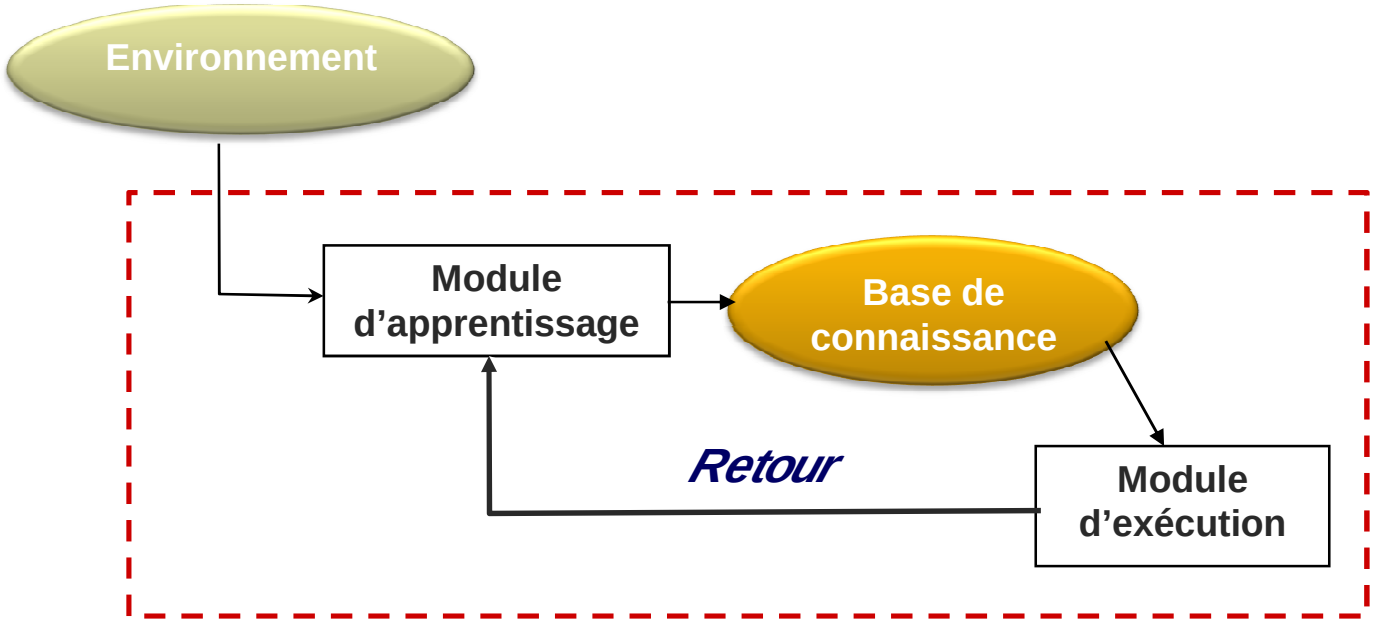
Département de Mathématique et Informatique

Université de Djelfa

2011-2012

L’apprentissage

A pour but d’améliorer les performances du système en tenant compte des ressources et des compétences dont il dispose.



L'AR: Apprentissage par l'interaction avec son environnement, pour maximiser ses récompenses. L'agent se charge de l'apprendre par lui même en renforçant les actions qui s'avèrent les meilleures.

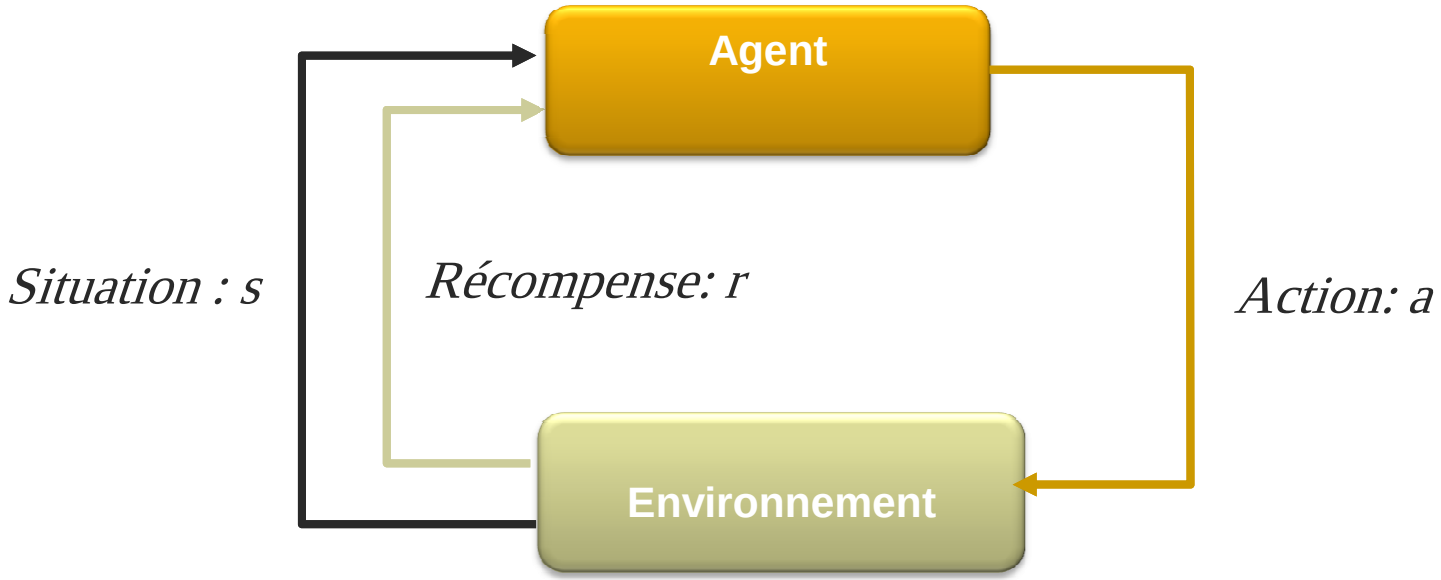
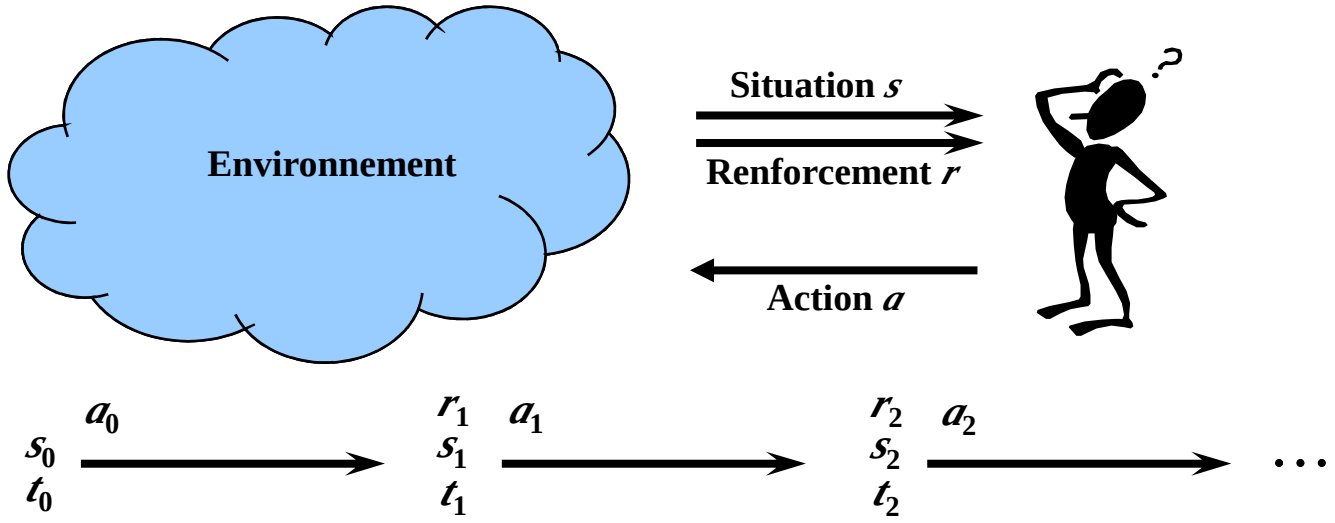


Figure.2 *L'apprentissage par Renforcement*

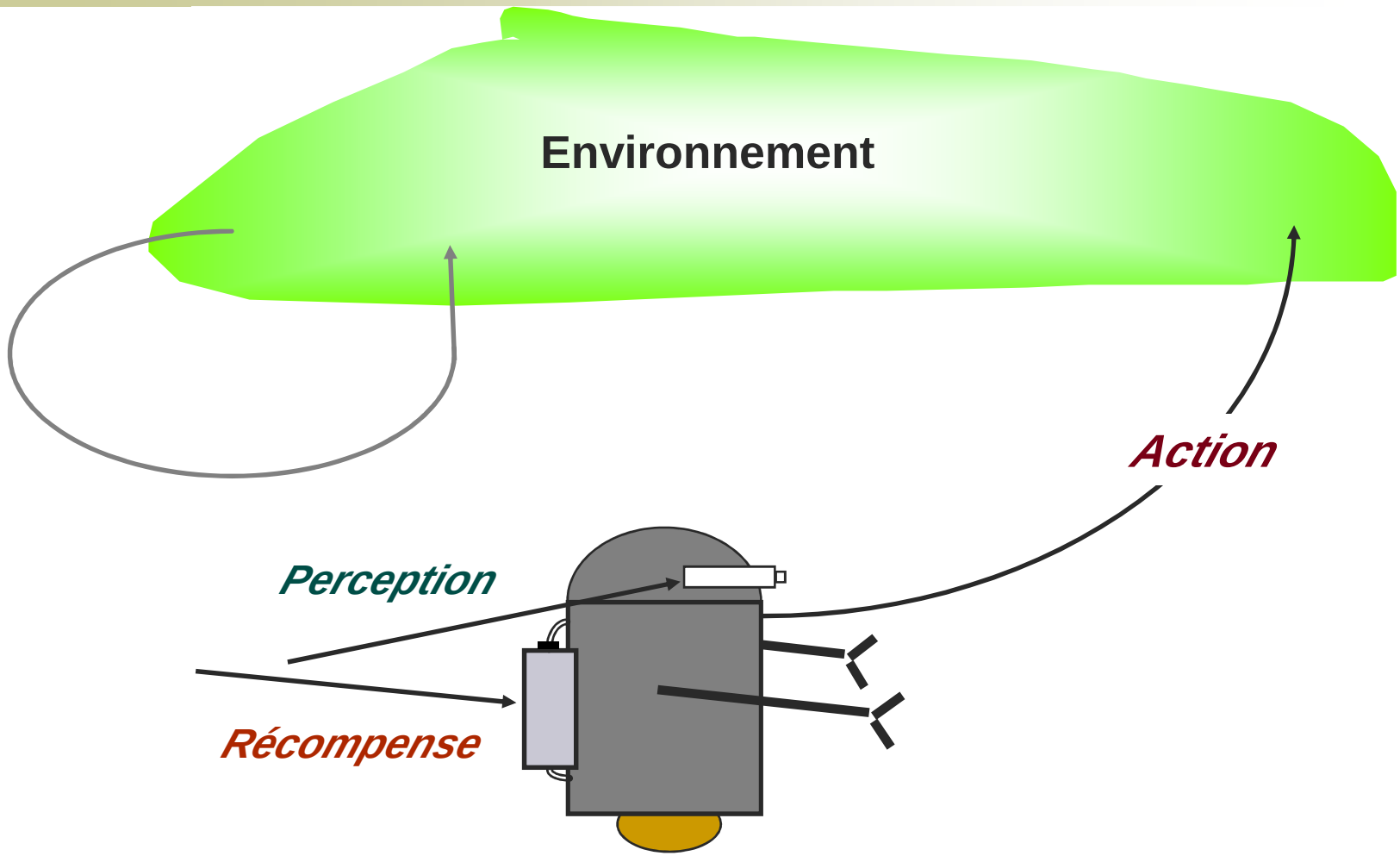
Apprentissage par renforcement (AR)

[Samuel 1959]…[Sutton et Barto 1998]…



- L'agent apprend à se rapprocher d'une stratégie comportementale optimale par des interactions répétitives avec l'environnement.
- Les décisions sont prises séquentiellement à des intervalles de temps discrets.
- L'environnement peut être stochastique et inconnu.

AR



Pourquoi l'AR ?

- Il est très utile dans le cadre de problèmes où :
  - des stratégies comportementales efficaces sont inconnues a priori ou sont difficilement automatisables
  - lorsqu'il y a de l'incertain dans la manière dont l'environnement évolue

- L'AR se distingue des autres approches d'apprentissage par plusieurs aspects :
  - L'apprentissage se fait sans supervision
  - Il repose sur le principe d'essai/erreur
  - Il s'appuie sur une estimation anticipée d'un renforcement

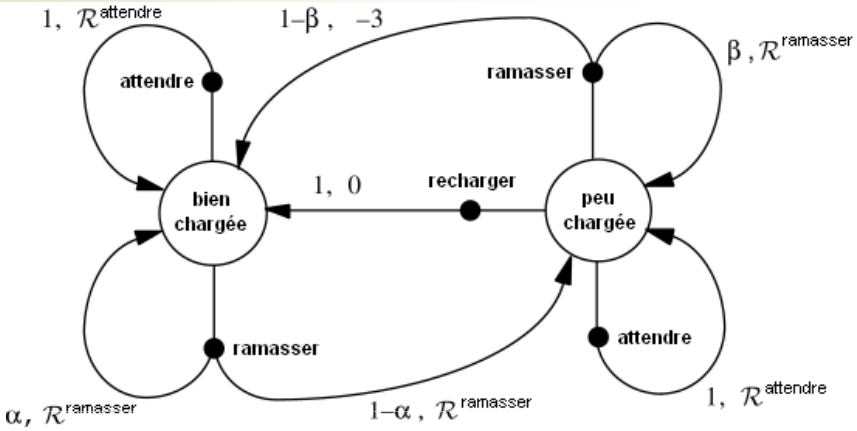
- Il a déjà obtenu de très bons résultats pratiques dans le cadre de problèmes complexes pour les méthodes classiques d'IA
  - L'exemple le plus célèbre est celui de l'application TD-Gammon [Tesauro 1992, 1994, 1995, 2002] qui est devenue le meilleur joueur du jeu Backgammon au monde
    - L'espace d'états de l'ordre de 10^{20} et l'espace d'actions de l'ordre de 10^{2}

Les bases de l'AR

- Les théories qui regroupent les différents aspects utiles à une formalisation de l'AR
  - Théorie des probabilités
    - Représentation et évolution des états d'un environnement au travers du formalisme des chaînes de Markov
  - Théorie de la décision
    - Le problème fondamental de la prise de décision d'un agent
  - Théorie de l'utilité
    - Évaluation d'une décision (correspondance avec le notion de performance)

- Un problème d'AR peut être généralement représenté par le formalisme des processus de décision Markoviens (PDM)
  - On considère que l'environnement peut être observé
  - Les situations observées correspondent à des états réels de l'environnement

Processus de décision Markovien (PDM)



Exemple d'un robot ramasseur

- Un PDM est défini par un tuple (S, A, T, R)
  - S est un ensemble fini d'états
  - A est un ensemble fini d'actions
  - $T: S \times A \times S \rightarrow [0,1]$ est une fonction de probabilité de transition des états
    - $T(s, a, s') = P\{s_{t+1} = s' \mid s_t = s, a_t = a\}$
  - $R: S \times A \times S \rightarrow \Re$ est une fonction de renforcement à valeurs réelles
    - $R(s, a, s') = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\}$

- Les fonctions T et R représentent le modèle de l'environnement
  - Elles sont généralement stochastiques
  - Elles sont sujettes à la propriété de Markov
    - Il est possible de déterminer l'évolution au prochain état de l'environnement en considérant uniquement l'état actuel et l'action choisie (T est indépendante des interactions passées)
      - $P\{s_{t+1} = s' \mid s_t, a_t\} = P\{s_{t+1} = s' \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0\}$

La fonction de renforcement R

- La fonction de renforcement correspond à un « feedback » (récompense ou punition) de l'environnement
  - Elle permet d'évaluer le comportement de l'agent
  - Le choix de cette fonction est à la charge du concepteur
  - Elle est critique pour le succès de l'apprentissage

- Types de fonction de renforcement
  - Renforcement immédiat
    - Chaque action conduit à une récompense ou une punition
  - Renforcement immédiat en minimisant le temps nécessaire pour atteindre un but
    - Chaque action conduit à une punition, sauf celle qui amène à l'état final
  - Renforcement retardé
    - Aucune action ne conduit à une récompense ou punition, sauf celle qui amène à l'état final

Le but de l'agent

- Apprendre une **politique** π (stratégie comportementale) qui maximise une mesure R de renforcement cumulatif à long terme en allant d'un état initial à un état final:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \text{, où } 0 \leq \gamma \leq 1$$

γ est un facteur de décompte pour les renforcements futurs

- La politique π est une fonction qui associe une distribution de probabilités sur les actions $a \in A$ à des états $s \in S$:

$$\pi : S \rightarrow \prod(A)$$

- Une politique optimale π^* est celle qui optimise une fonction d'évaluation V^π ou Q^π

Les fonctions d'évaluation V^{π} et V^{*}

- La fonction d'évaluation $V^{\pi}(s)$ associe à chaque état $s \in S$ une mesure du renforcement cumulé que l'agent reçoit lorsqu'il suit une politique π à partir de l'état s

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

$$= E_{\pi}\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\right\}$$

$$= E_{\pi}\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s\}$$

$$= \sum_a \pi(s,a) \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V^{\pi}(s')\right]$$ **(équation de Bellman)**

s	$V(s)$
s_0	R_t
s_1	…
…	…
s_n	…

- La fonction d'évaluation optimale $V^{*}(s)$

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) = \max_{a} \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V^{*}(s')\right]$$

Les fonctions de qualité Q^{π} et Q^{*}

- La fonction d'évaluation $Q^{\pi}(s,a)$ est définie de façon similaire à $V^{\pi}(s)$
  - Elle associe à chaque couple état/action $s \in S$ et $a \in A$ une mesure du renforcement cumulé que l'agent reçoit lorsqu'il suit une politique π en exécutant l'action a à partir de l'état s

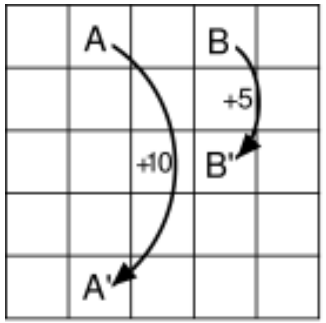
$$Q^{\pi}(s,a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\}$$

s	a	Q(s,a)
s_0	a_0	R_t
s_0	a_1	…
…	…	…
s_n	a_0	…
s_n	a_1	…

- La fonction d'évaluation optimale $Q^{*}(s,a)$

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma \max_{a'} Q^{*}(s',a')\right]$$

Exemple des fonctions V et π



Tableau

- Problème de déplacement dans un tableau
  - L'espace d'états $\mathcal{S}$ de l'environnement = les cellules du tableau
    - L'état initial est une cellule choisie aléatoirement
  - L'espace d'actions $\mathcal{A}$ de l'agent = {nord, sud, est, ouest}
  - La fonction de transition $\mathcal{T}$ est déterministe et connue
    - Un action exécutée dans l'état A transfère l'environnement à l'état A'
    - Une action exécutée dans l'état B transfère l'environnement à l'état B'
    - Une action qui amène l'agent dehors du tableau ne change pas l'état de l'environnement
    - Les autres actions avancent l'agent sur le tableau
  - La fonction de renforcement $\mathcal{R}$
    - Un action exécutée dans l'état A conduit à une récompense de +10
    - Une action exécutée dans l'état B conduit à une récompense de +5
    - Une action qui amène l'agent dehors du tableau conduit à une punition de -1
    - Les autres actions conduisent à un renforcement nul

Méthodes d'optimisation

- Programmation dynamique [Bellman 1957][Bertsekas et Tsitsiklis 1996]…
  - Des méthodes incrémentales
    - Réalisation d'itérations successives qui se rapprochent petit à petit de la fonction de valeur optimale
  - Cependant, l'agent doit connaître parfaitement le modèle de l'environnement (les fonctions T et R)
    - Résolution de problèmes de planification plutôt que de problèmes d'apprentissage

- Monte Carlo [Michie et Chambers 1968][Rubinstein 1981]…
  - Des méthodes qui s'appuient sur l'expérience
    - L'agent n'est pas obligé de connaître parfaitement le modèle de l'environnement
      - Un problème d'apprentissage est posé (T et R sont ainsi apprises par l'expérience)
    - Elles permettent un apprentissage en ligne
  - Cependant, elles ne sont pas incrémentales
    - Une évaluation ne se fonde pas sur d'autres évaluations, ce qui nécessite donc un apprentissage décomposé en une succession d'épisodes de longueur finie

Les méthodes d'apprentissage par différences temporelles

Vont réunir les avantages des méthodes de PD et MC et constituent les méthodes les plus utilisées en pratique.

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

$$V_{t+1}^{\pi} \leftarrow \begin{cases} V_t^{\pi}(s) + \alpha\left[r_t + \gamma V_t^{\pi}(s_{t+1}) - V_t^{\pi}(s)\right] & \text{pour} \quad s = s_t \\ V_t^{\pi}(s) & \text{sinon} \end{cases}$$

Convergence : La convergence vers les politiques optimale à condition que:

- chaque état soit visité une infinité de fois,
- Le taux d'apprentissage respecte les conditions suivantes :

$$\sum_t \alpha_t(s) = +\infty \qquad \sum \alpha_t^2(s) < +\infty, \forall s \in S$$

$$\alpha_t(s) = \frac{1}{1 + \textit{nombre de visites à l'état s depuis le début de l'épisode}}$$

1. **L'algorithme TD(0):** Est un algorithme d'évaluation d'une politique.
2. **L'algorithme SARSA** : est similaire à l'algorithme *TD(0)*

$$Q(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \alpha\left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$$

Méthodes de différence temporelle

[Samuel 1959][Klopf 1972][Holland 1976, 1986][Sutton 1988]…

- Combinaison
  - de l'aspect incrémental des méthodes de programmation dynamique
  - du recours à l'expérience des méthodes de Monte Carlo
- Des méthodes qui ne nécessitent pas de modèle de la dynamique de l'environnement
- Elles évaluent par l'expérience l'intérêt
  - d'être dans un état donné
  - d'une action à partir dans un état donné
- Elles nécessitent un bon compromis d'exploration/exploitation

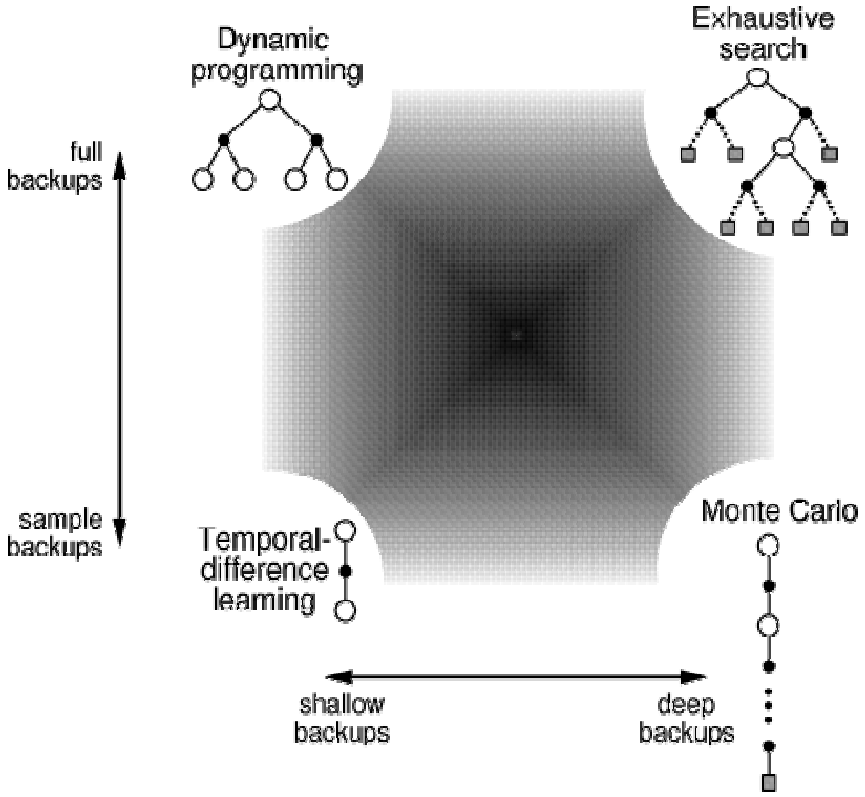


Figure récupérée de [Sutton et Barto 1998]

Algorithme de différence temporelle (TD)

- TD est l'algorithme de base de l'apprentissage par renforcement. Il consiste à comparer :
  - la récompense que l'agent reçoit effectivement de l'environnement
  - la récompense qu'il s'attend à recevoir en fonction des estimations $V(s)$ précédentes
- Définition de l'équation de mise à jour incrémentale

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

$$V(s_t) \leftarrow V(s_t) + \alpha\delta_t$$

$V(s)$	: Fonction d'évaluation des états
α	: Taux d'apprentissage
δ_t	: Erreur de différence temporelle
r_{t+1}	: Récompense immédiate
γ	: Facteur de décompte temporel

- Dans le cadre de cet algorithme, l'agent ne peut pas déterminer en avance quel est l'état suivant de meilleure valeur estimée
  - Il est donc incapable de déduire quelle politique il doit suivre
  - C'est pourquoi les algorithmes qui prennent en compte la valeur des couples état/action (s,a) sont préférés

Algorithme Sarsa $(s_t,a_t,r_{t+1},s_{t+1},a_{t+1})$

- Sarsa est semblable au TD, sauf que la fonction d'évaluation des états $V(s)$ est remplacée par la fonction d'évaluation des actions $Q(s,a)$
  - On estime la valeur d'une action à partir d'un état plutôt que d'être dans un état
- Définition de l'équation de mise à jour incrémentale

$$\delta_t = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\delta_t$$

$Q(s,a)$	: Fonction d'évaluation des actions
α	: Taux d'apprentissage
δ_t	: Erreur de différence temporelle
r_{t+1}	: Récompense immédiate
γ	: Facteur de décompte temporel

- Le fait de toujours déterminer à l'instant t l'action qui sera exécutée à l'instant $t+1$ signifie que l'agent suit la stratégie d'exploration adoptée

Algorithme Q-learning [Watkins 1989]

- Q-learning est une simplification de Sarsa du fait qu'il n'est pas nécessaire de déterminer à l'instant t l'action qui sera exécutée à l'instant $t+1$
- Définition de l'équation de mise à jour incrémentale

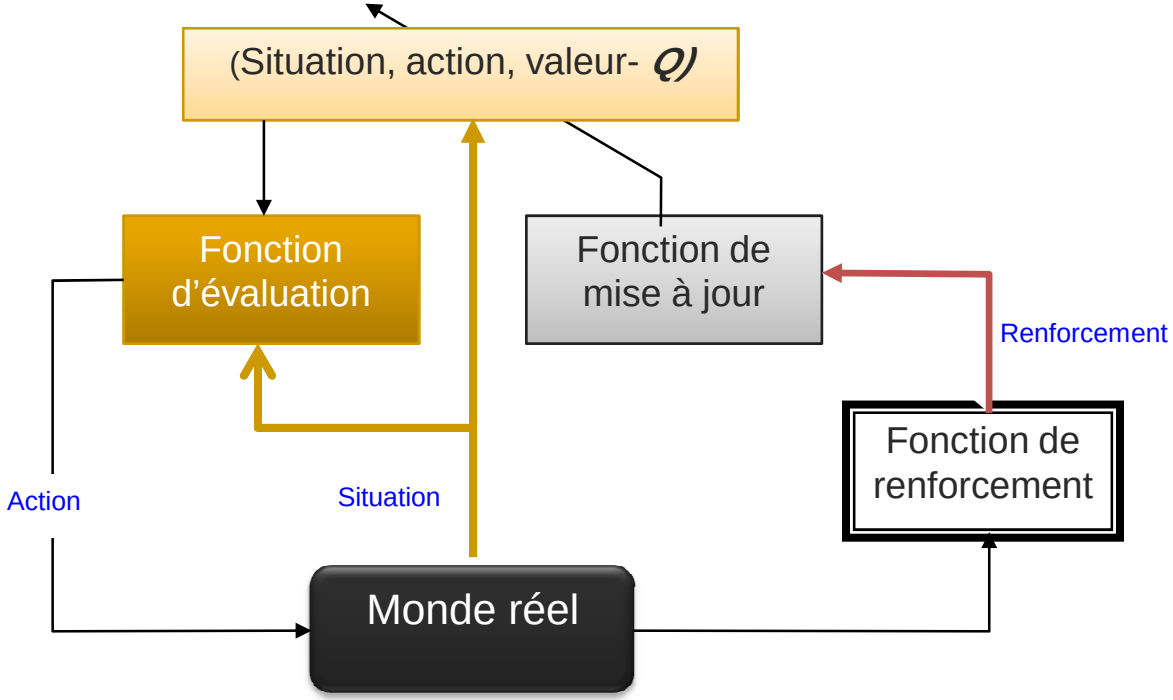
$$\delta_t = r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\delta_t$$

$Q(s,a)$	: Fonction d'évaluation des actions
α	: Taux d'apprentissage
δ_t	: Erreur de différence temporelle
r_{t+1}	: Récompense immédiate
γ	: Facteur de décompte temporel

- Au lieu de déterminer l'action d'avance, l'agent choisit celle qui a la meilleure valeur estimée.
  - L'agent ne suit pas la stratégie d'exploration adoptée
- Q-learning est l'algorithme d'AR le plus connu en raison de ses preuves formelles de convergence [Watkins et Dayan 1992]

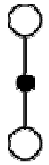
Q-learning [Watkins 1989]



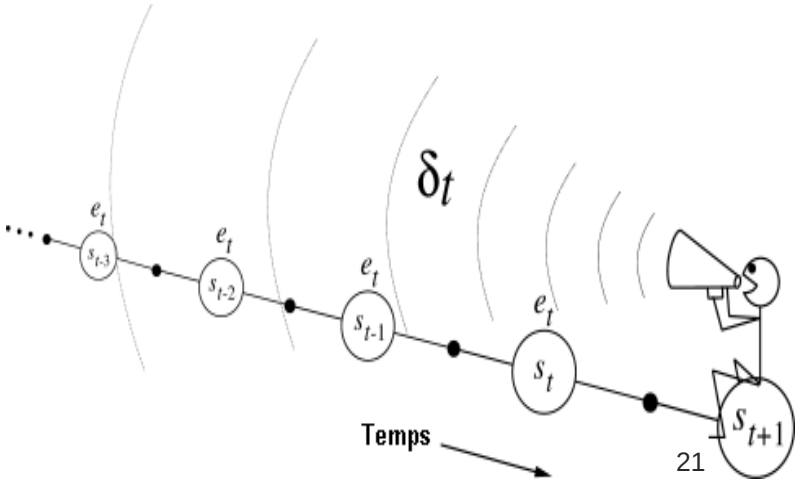
L'algorithme Q-Learning

TD(λ), Sarsa(λ) et Q(λ)

- Un défaut des algorithmes précédents est qu'ils ne mettent à jour qu'une valeur par pas de temps
  - La procédure de mise à jour est particulièrement lente

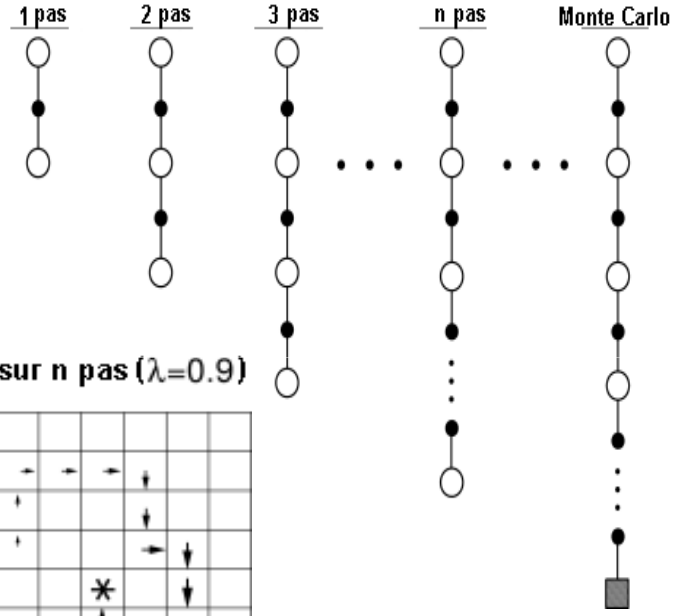


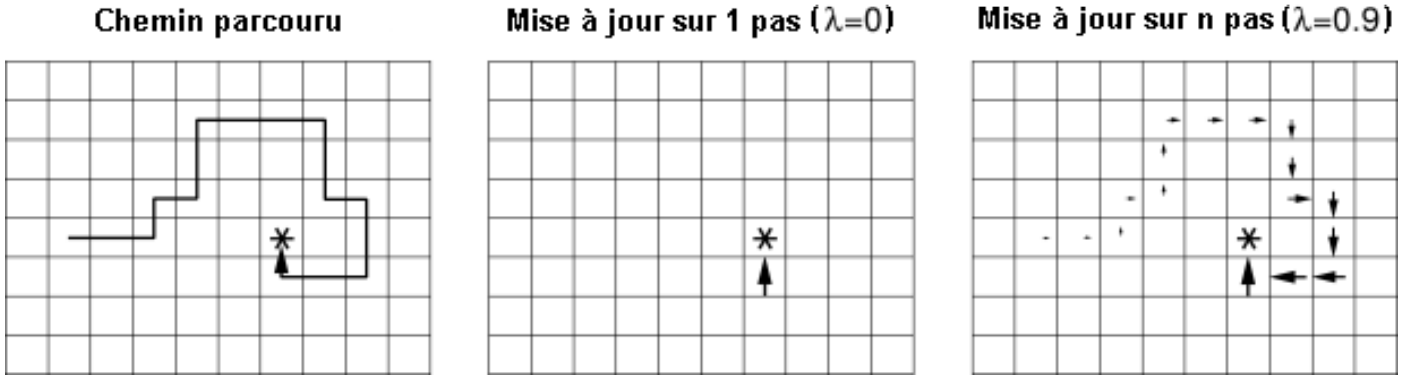
- On les dote donc d'une mémoire de transitions (ou historique)
  - Chaque état est associé à un indicateur $e(s)$ qui mesure l'écoulement du temps depuis sa dernière visite
  - L'indicateur $e(s)$ est
    - affecté de 1 à chaque fois que l'état est visité
    - diminué d'un facteur $\gamma\lambda$ pour les autres états
  - L'erreur de différence temporelle corrige simultanément l'estimation des états en fonction de la valeur de $e(s)$



TD(λ), Sarsa(λ) et Q(λ)

- Le facteur λ introduit un mécanisme d'amorçage
  - Si λ=0, on retombe sur TD(0), Sarsa et Q-learning
  - Si λ=1, les algorithmes s'apparentent à Monte Carlo





- Les algorithmes dotés d'historique sont plus efficaces que leur version de base, cependant ils requièrent plus de mémoire
  - Un compromis entre la vitesse d'apprentissage et la mémoire utilisée est nécessaire

Architecture Acteur Critique

c'est une structure plus complexe que SARSA et Q-Learning ou la politique et la valeur d'état sont stockées séparément.

Choix d'action

1- Gloutonne :

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

2- *eps* Gloutonne :

$$a_{\varepsilon-gloutonne} = \begin{cases} \arg\max_{a \in A(s_t)} Q(s_t, a) \text{ avec probabilté } \varepsilon \\ \text{action prise au hasard dans } A(s_t) \text{ avec probabilité } 1-\varepsilon \end{cases}$$

3- Softmax :

$$\Pr[a_t | s_t] = \frac{Q(s_t, a_t)}{\sum_{a \in A(s_t)} Q(s_t, a)}$$

4- Boltzmann :

$$\Pr[a_t | s_t] = \frac{e^{\frac{Q(s_t, a_t)}{\tau}}}{\sum_{a \in A(s_t)} e^{\frac{Q(s_t, a_t)}{\tau}}}$$

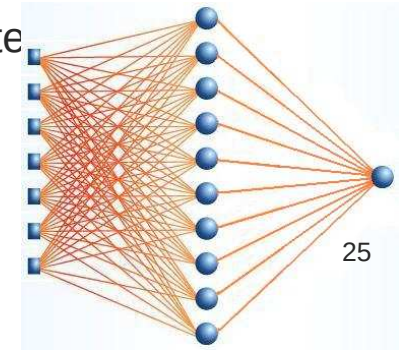
Limitations des PDMs

- Quand la propriété de Markov n'est plus assurée
  - Les environnements partiellement observables
    - L'agent n'a pas de connaissance parfaite sur l'état de son environnement
  - Les systèmes multi-agents
    - Les environnements sont non stationnaires
- Quand les espaces d'états et d'actions sont continus
  - La finitude et discrétisation est perdue
- Quand les transitions des états se déroulent dans des intervalles de temps continus

- **Il existe une infinité d'extensions dans le cadre des processus de décision non Markoviens pour traiter ces limitations**

D’autres limitations du cadre de l’AR

- Quand les espaces d’états et d’actions sont très grands
  - Mémoire nécessaire pour représenter une table
  - Temps nécessaire pour pouvoir remplir efficacement cette table
    - La convergence est assurée lorsque chaque ligne est visitée infiniment
- Voies envisageables afin de écarter ces limitations
  - Utiliser des connaissances disponibles pour
    - Structurer la prise de décision
    - Abstraire les données afin de trouver des granularités pertinentes
    - Mais il n’y pas de recette générale
      - Chaque application requiert une expertise particulière pour être représentée de la façon la plus adéquate possible
  - Utiliser des méthodes de généralisation de la fonction d’évaluation
    - Algorithmes d’AR fondées sur des méthodes de descente du gradient
    - Par exemple, un réseau de neurones artificiels



Itération de la valeur

Initialisation :
- un PDM : (S,A,R,P)
- un seuil de précision

Initialiser V aléatoirement

$i \leftarrow 0$

Répéter

Pour tout $s \in S$ faire

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

Fin pour

$i \leftarrow i+1$

Jusque $\left\|V_i - V_{i-1}\right\| \leq \varepsilon$

Pour tout état $s \in S$ faire

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

Fin pour

Itération de la politique

Initialisation :
- un PDM : (S,A,R,P)
- un seuil de précision

Initialiser π_0 aléatoirement

$k \leftarrow 0$

Répéter

Initialiser V_0^{π} aléatoirement

$i \leftarrow 0$

Répéter

Pour tout $s \in S$ faire

$$V_{i+1}(s) \leftarrow \sum_{s'} P(s,\pi(s),s')\left[R(s,\pi(s),s') + \gamma V_i(s')\right]$$

Fin pour

$i \leftarrow i+1$

Jusqu $\left\|V_i - V_{i-1}\right\| \leq \varepsilon$

Pour tout état $s \in S$ faire

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

Fin pour

$k \leftarrow k+1$

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

Algorithme de Monte Carlo

Initialisation :

- une politique π_0

pour chaque état s **faire**

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

Générer une trajectoire en suivant la politique π_0 à partir de l'état $s_0 = s$. On note (s_t, a_t, r_t) ; la suite des triples état, action, retour immédiat de cette trajectoire. On suppose la trajectoire de taille bornée T.

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

Fin pour

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

Fin pour

L'algorithme SARSA

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

Pour un grand nombre **faire**

Initialiser l'état initial s_0

$t \leftarrow 0$

Choisir l'action à émettre a_t en fonction de la politique de Q ($\varepsilon\_gloutonne$ par exemple) et l'émettre .

Répéter

Emettre a_t

Observer r_t et s_{t+1}

Choisir l'action à émettre a_t en fonction de la politique de Q ($\varepsilon\_gloutonne$)

$Q(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$

$t \leftarrow t+1$

Jusque $s_t \in F$

Fin pour

Q-Learning

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

Pour un grand nombre ∞ **faire**

Initialiser l'état initial s_0

$t \leftarrow 0$

Répéter

Choisir l'action à émettre a_t en fonction de la politique de Q ($\varepsilon\_gloutonne$) et l'émettre.

Observer r_t et s_{t+1}

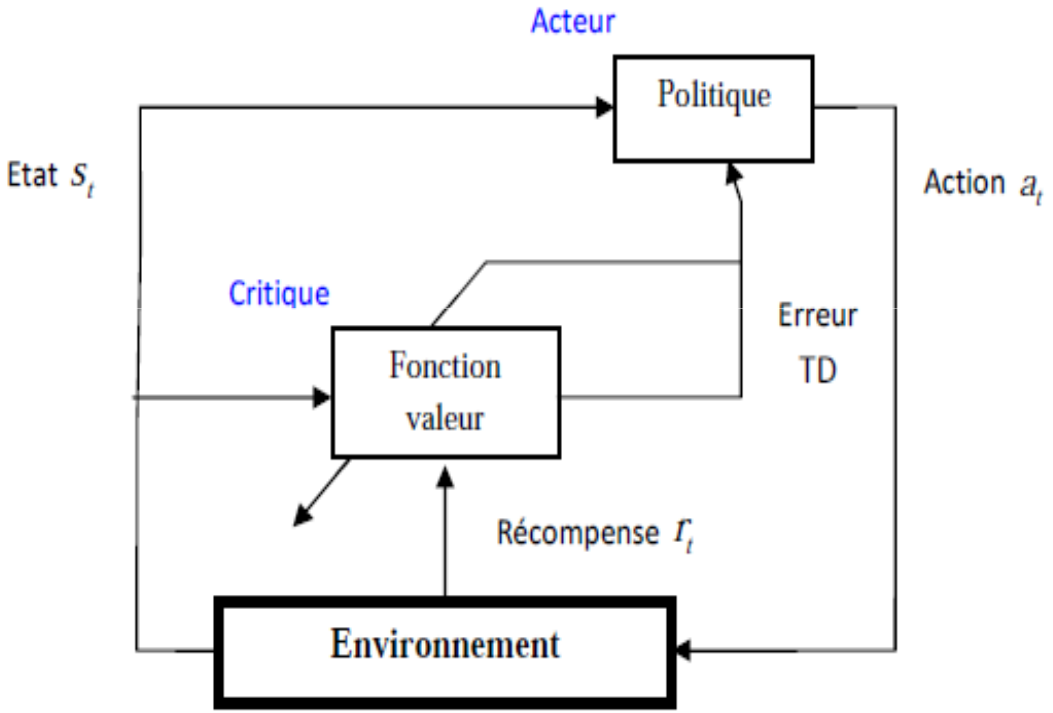
$$Q(s_t,a_t) \leftarrow Q^{\pi}(s_t,a_t) + \alpha\left[r_t + \gamma \max_{a \in A(s_{t+1})} Q(s_{t+1},a') - Q(s_t,a_t)\right]$$

$t \leftarrow t+1$

Jusque $s_t \in F$

Fin pour

ACRL Algorithm



L'algorithme TD(L)

Initialisation :

- une politique π

$V^{\pi} \leftarrow 0$ (Valeur arbitraire)

Pour ∞ **faire**

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

Initialiser l'état initial s_0

$t \leftarrow 0$

Répéter

Emettre l'action $a_t = \pi(s_t)$

Observer r_t et s_{t+1}

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

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

Pour $s \in S$ **faire**

$V^{\pi}(s) = V^{\pi}(s) + \alpha \delta e(s)$

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

Fin pour

$t \leftarrow t + 1$

Jusque $s_t \in F$

Fin pour

Quelques considérations

- Il faut employer une méthode de généralisation de la fonction d'évaluation
- Il faut définir ou redéfinir
  - La fonction de renforcement
  - La stratégie de recommandation
    - La stratégie d'exploration
  - La stratégie d'enregistrement
- Il faut réfléchir au problème de la sur-spécialisation
  - Le système se limite à ne recommander que des items similaires à ceux qui sont le plus appréciés par l'utilisateur