Apprentissage par renforcement

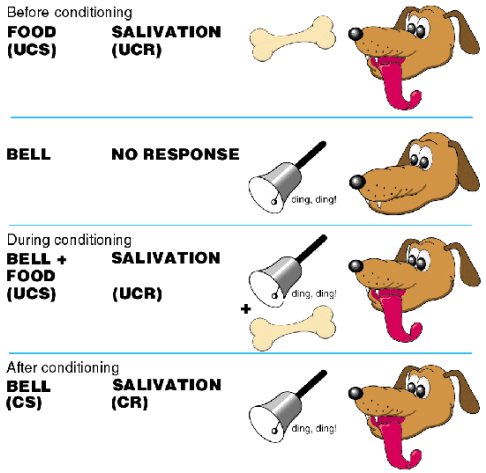
Apprendre en simulant :

Apprentissage par renforcement

- Apprentissage d'interactions avec un environnement
- Environnement pouvant être dynamique
- Pas d'exemples "tout fait"
- Exploration / Exploitation

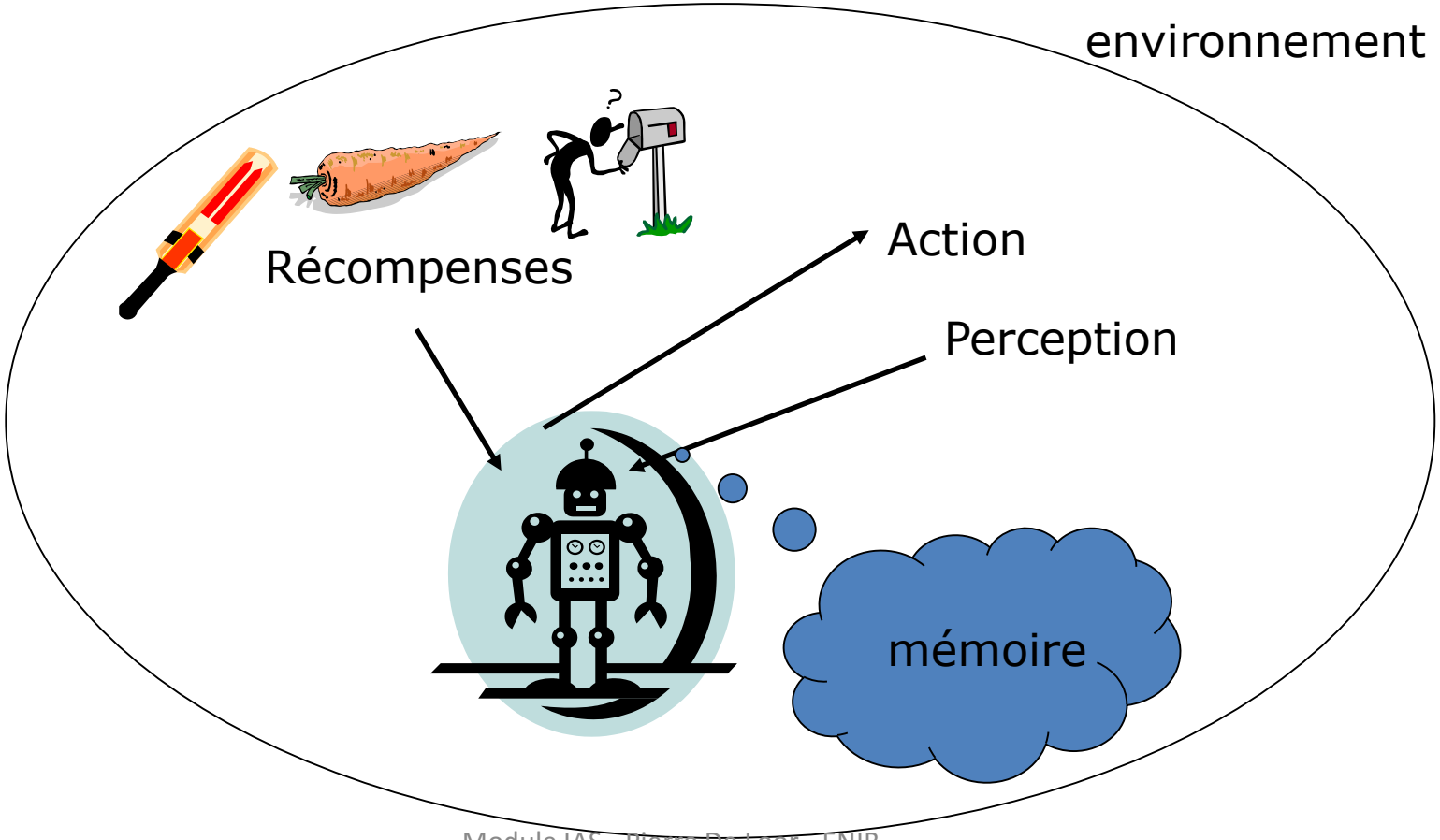
Les origines

- Le conditionnement …
  - [Thorndike 1911],
  - [Pavlov 1927],
  - [Skinner 1938],
  - [Tolman 1930]

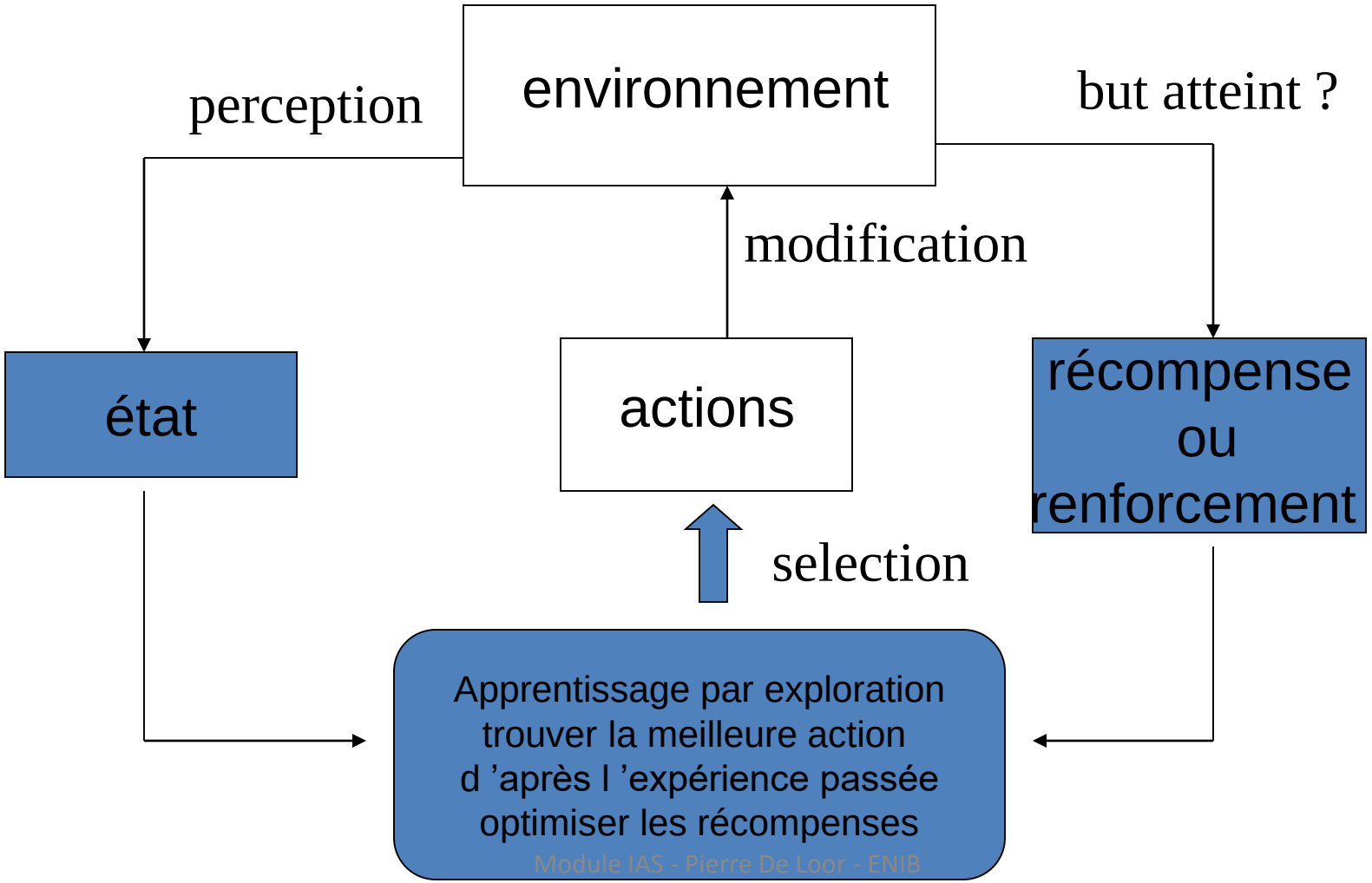


Principe

Apprentissage artificiel par renforcement:
La carotte, le bâton et le silence



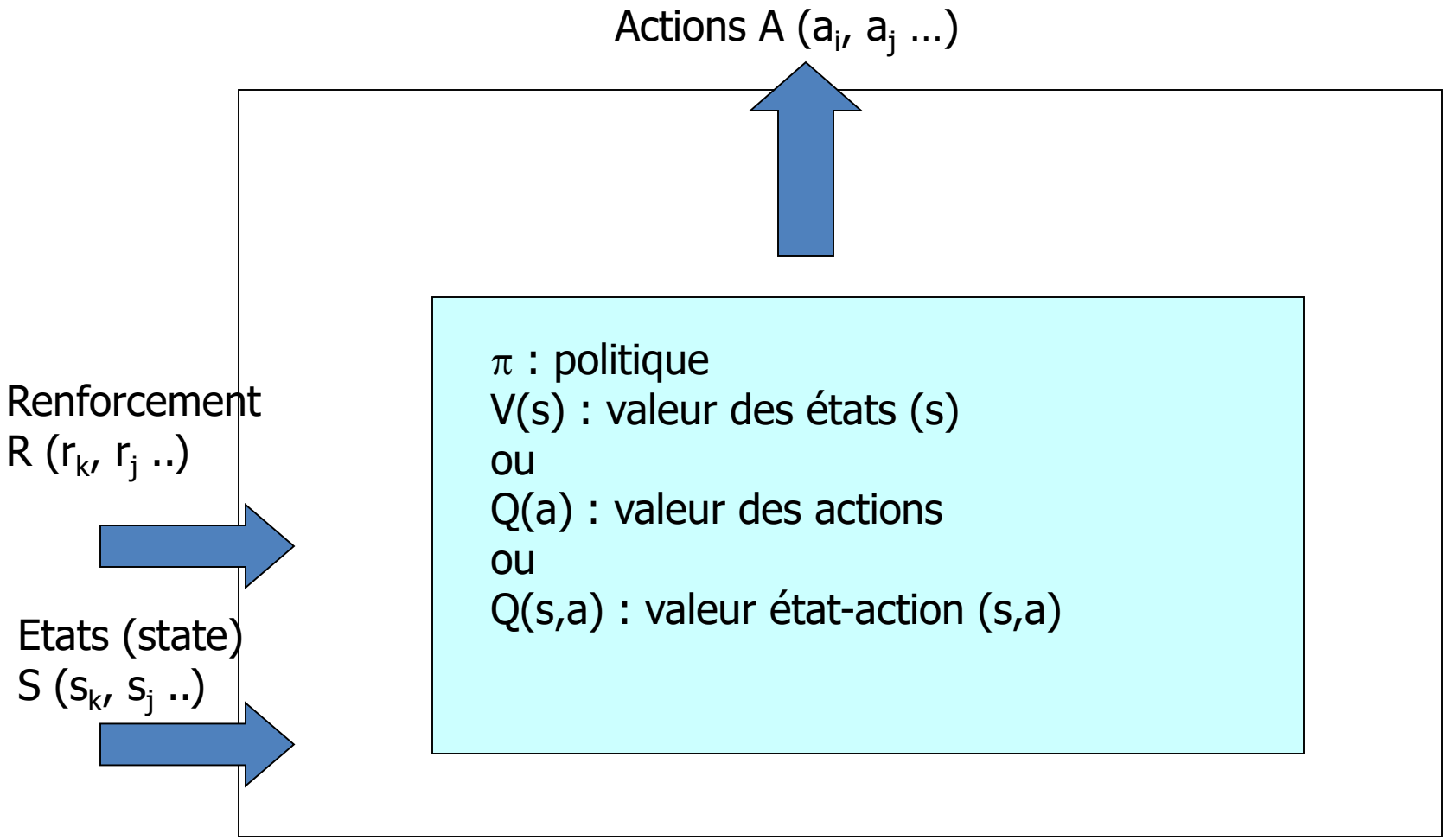
Principe


environnement
perception
but atteint ?
modification
état
actions
récompense
ou
renforcement
selection
Apprentissage par exploration
trouver la meilleure action
d 'après l 'expérience passée
optimiser les récompenses

Bien comprendre le renforcement

- But atteint = récompense
- mauvaise solution = punition
- Entre les deux = rien
    - c'est à l'algorithme de trouver "comment" atteindre le but, pas à nous de lui dire "chaud" ou "froid" car il va apprendre à atteindre un sous objectif.
    - Il n'est pas nécessaire (ni souhaitable) de mesurer le degrès de satisfaction du but, ou d'imaginer les étapes intermédiaires

Variables importantes

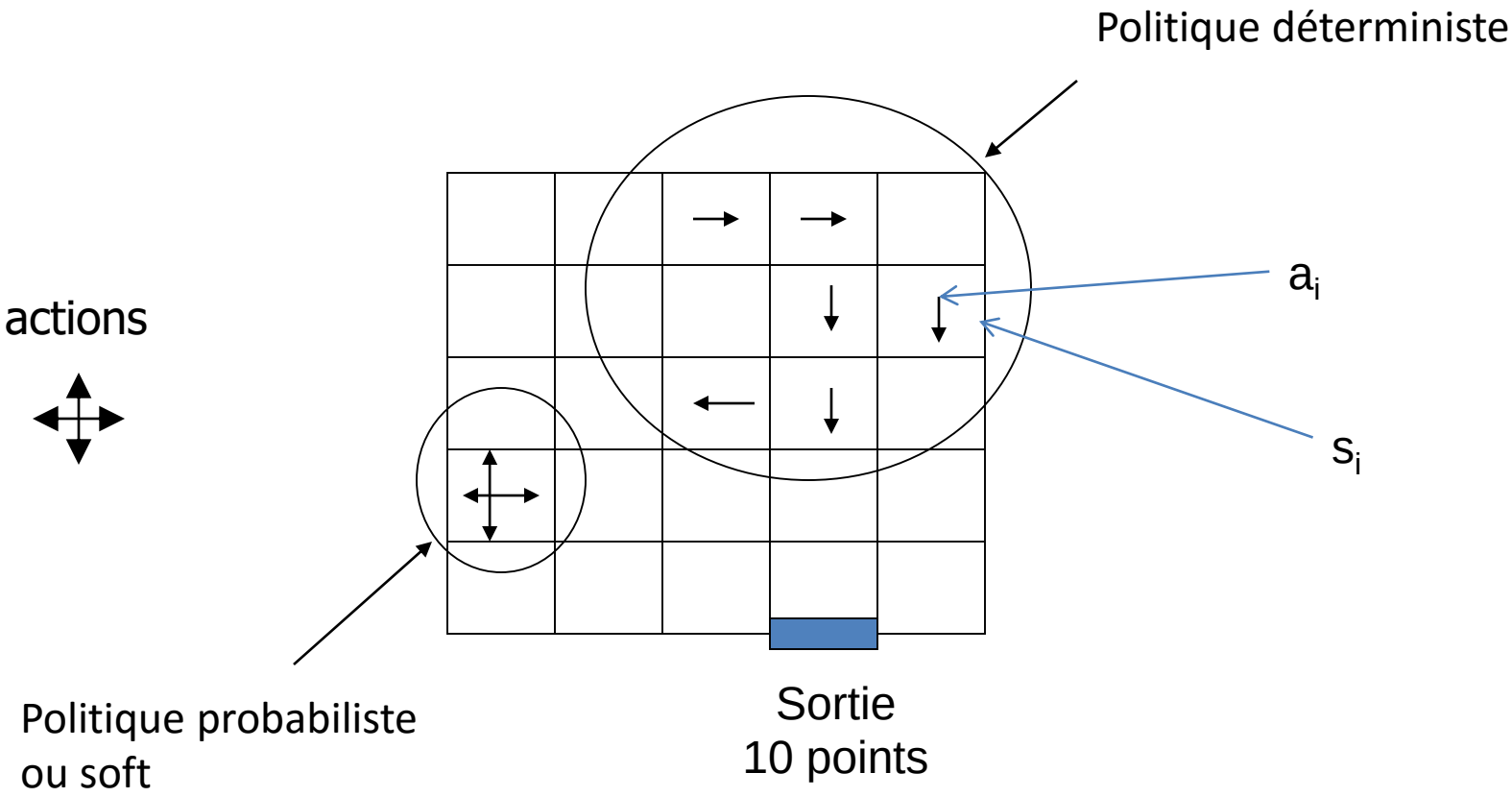


π : Politique(s)

- Caractérise ce qu'il faut faire ? s->a
- probabiliste :
  - π(s,a) : probabilité de choix (dure/permissive)
- déterministe :
  - π(s) : action, pas une probabilité
- π^* optimale : permet d 'obtenir le plus de gains
  - compromis court/long terme , infini ? épisode ?
  - C 'est ce que l 'on cherche à apprendre

Politique π

- exemple



V : Valeur des états (s->$\Re$)

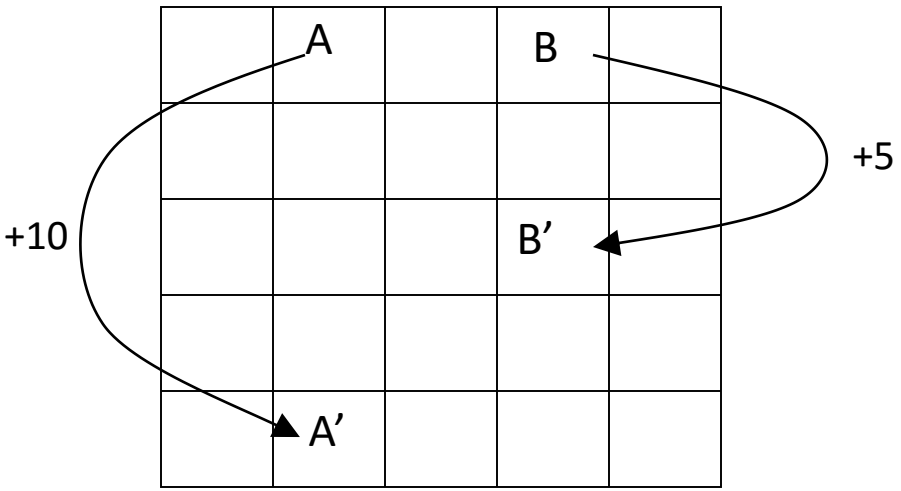
États :	s_1	s_2	s_3	…
Valeur	V(S1)	V(S2)	V(S3)	

- V(s) Gains que je peux espérer en s
  - sur n coups (épisodes) : V(s) est borné
    - Reward =r_{t+1}+ r_{t+2} + … + r_{t+n}
  - à l ’infini : pondération -> remise γ
    - Reward = r_{t+1} + γr_{t+2} + $\gamma^2 r_{t+3}$ +…
- V* : valeur optimale (inconnue)
- V^{π} : valeur pour la politique π

Autre Exemple

- L ’environnement

actions



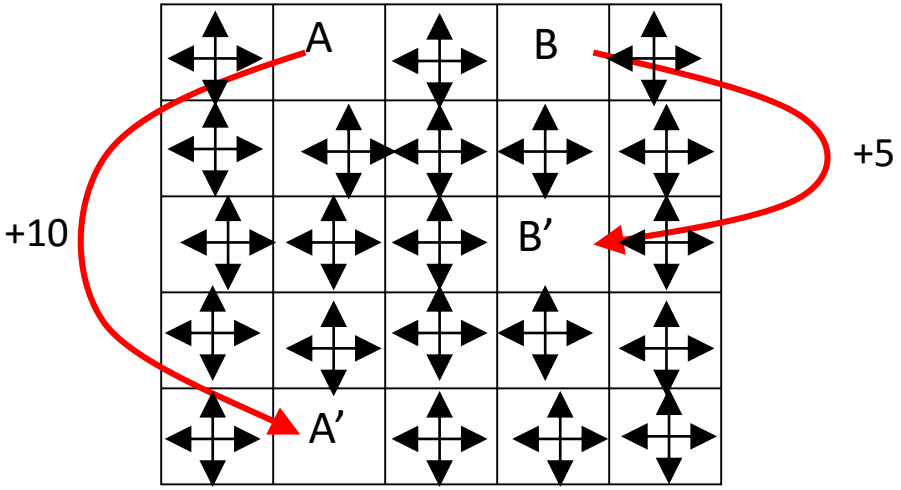
Renforcements
- bords : -1

- A->A ’ : 10
- B->B ’ : 5
(pour toute action)

- ailleurs : 0

Gains des états V^{π} pour une politique π

- politique π : probabilité de choix des 4 directions identique partout $\pi(s_i,\text{droite}) = \pi(s_i,\text{haut}) = \ldots = 0.25$
- Horizon infini, remise : γ=0.9

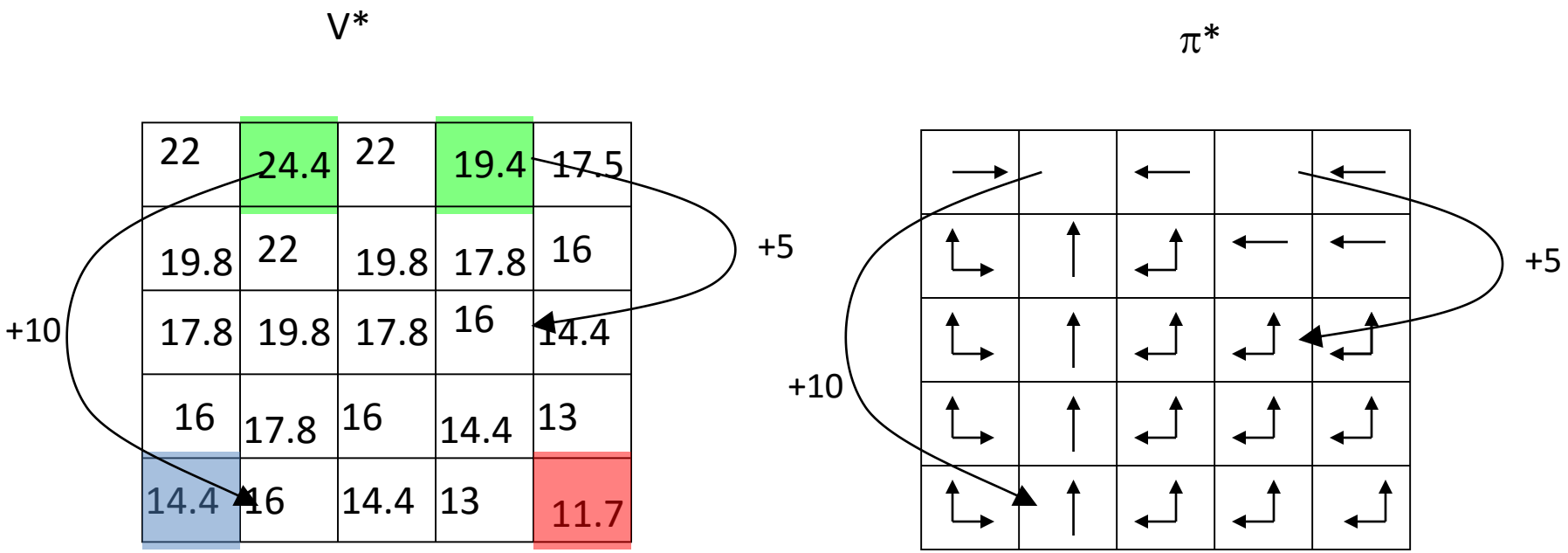


V^{π}

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2

Gain et politique optimale

- V* : gains optimums pour la politique optimale π^*



Plusieurs actions (Q(s,a))

- Association état/action
- Politique π

États :	s_1	s_2	s_3	…
Actions : a1 a2 a3 …	$Q_{(s1,a1)}$ … …	$Q_{(a1,s2)}$	$Q_{(a3,s3)}$	…

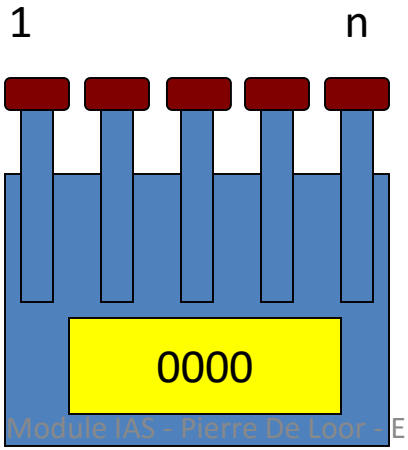
Q (s,a)

- Comme V mais …
  - sur une action : Q(a) : a -> $\Re$
  - sur un couple état/action : Q(s,a) : (s,a) -> $\Re$
  - Q(s,a) : gains espérés en exécutant a à partir de s.
- Q* valeur optimale (inconnue)
- $Q^{\pi}(s,a)$ valeur pour une politique π

Le problème de l'exploration

Bandit Manchot

- pas d'états (ou plutôt un seul)
  - pas de V
- n bras : n actions $a_1 \ldots a_n$ -- > $Q(a_i)$
- chaque bras permet de gagner une somme r distribuée aléatoirement (proba inconnue)



Bandit Manchot

- Explorer : jouer au pif
- Exploiter : jouer le meilleur bras estimé
  - Valeur d'une action a (bras) : Q(a) (= Q*(a))
    - Reflète la moyenne des gains obtenue par a
  - Valeur estimée d'une action après qu'elle soit choisie k_a fois :

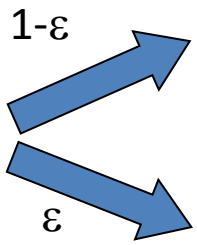
$$Q_t(a) = \frac{r1+r2+\ldots\ldots+rk_a}{k_a}$$

  - si $k_a \rightarrow \infty$ alors $Q_t(a) \rightarrow Q(a)$

Bien apprendre : explorer/exploiter

- Taux d'exploration : ε (ε-Greedy methods)
- ε variable aléatoire ($0 \leq \varepsilon \leq 1$)

probabilité



Jouer le meilleur « greedy action »
$\max_a(Q(a))$

Explorer

- valeur de ε ?
  - ↗ apprentissage lent, prudent, performant à long terme
    versus
  - ↘ apprentissage rapide mais moins performant en moyenne

  – si ε est toujours > 0 on parle de politique ε-soft

Explorations plus fines

- Softmax
  - Dépend de la valeur des actions $Q_t(a_i)$
  - plus une action est “valable” plus elle a de chance d’être choisie (distribution de Boltzman, fixer la température ?)

probabilité
de choisir « a »
parmi n actions

$$\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}}$$

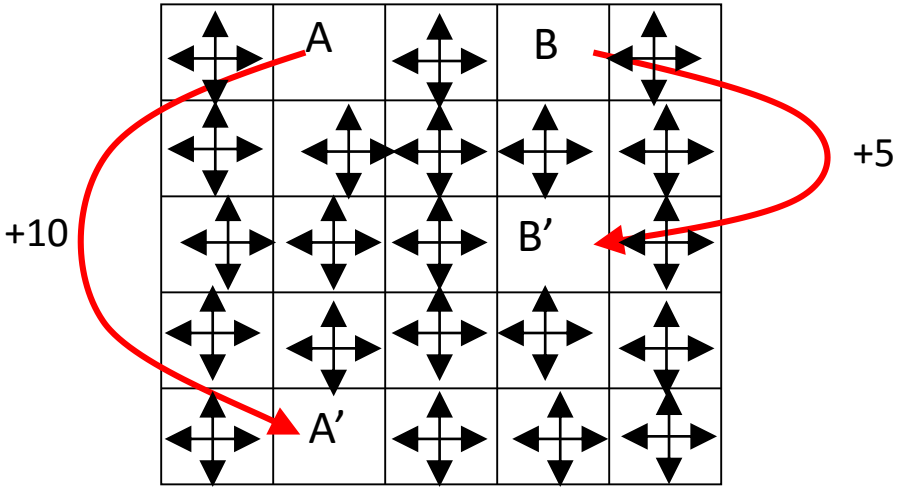
$\tau \to 0$: greedy

$\tau \to \infty$: equiprobable

Le problème de l'apprentissage

Gains des états V^{π} pour une politique π

- $Q_t(a) = \frac{r1+r2+\ldots\ldots+rk_a}{k}$
- Mais quand k-> ∞ (Horizon infini) , il faut une version récursive/incrémentale



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2

Il est possible d'obtenir une définition incrémentale Q(a)

$$Q_t(a) = \frac{r_1 + r_2 + \ldots + r_{ka}}{ka}$$

$$Q_{k+1} = \frac{\sum_{i=1}^{k+1} r_i}{k+1}$$

$$Q_{k+1} = \frac{\sum_{i=1}^{k} r_i + r_{t+1}}{k+1}$$

Q_k

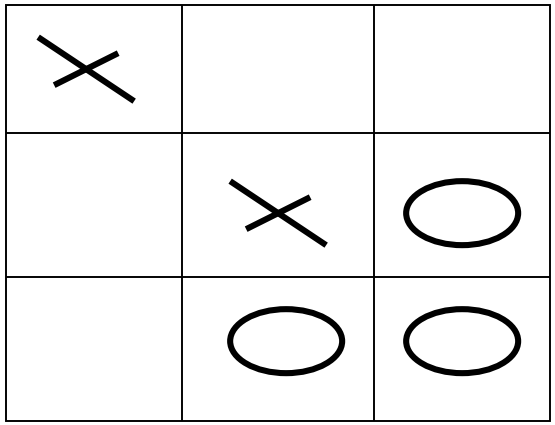
$$Q_{k+1} = Q_k + \frac{1}{k+1} * [r_{k+1} - Q_k]$$

Voir annexes

$$Q_{k+1} = Q_k + \alpha.[erreur]$$

Apprentissage par différence temporelle : Exemple : Tic Tac Toe

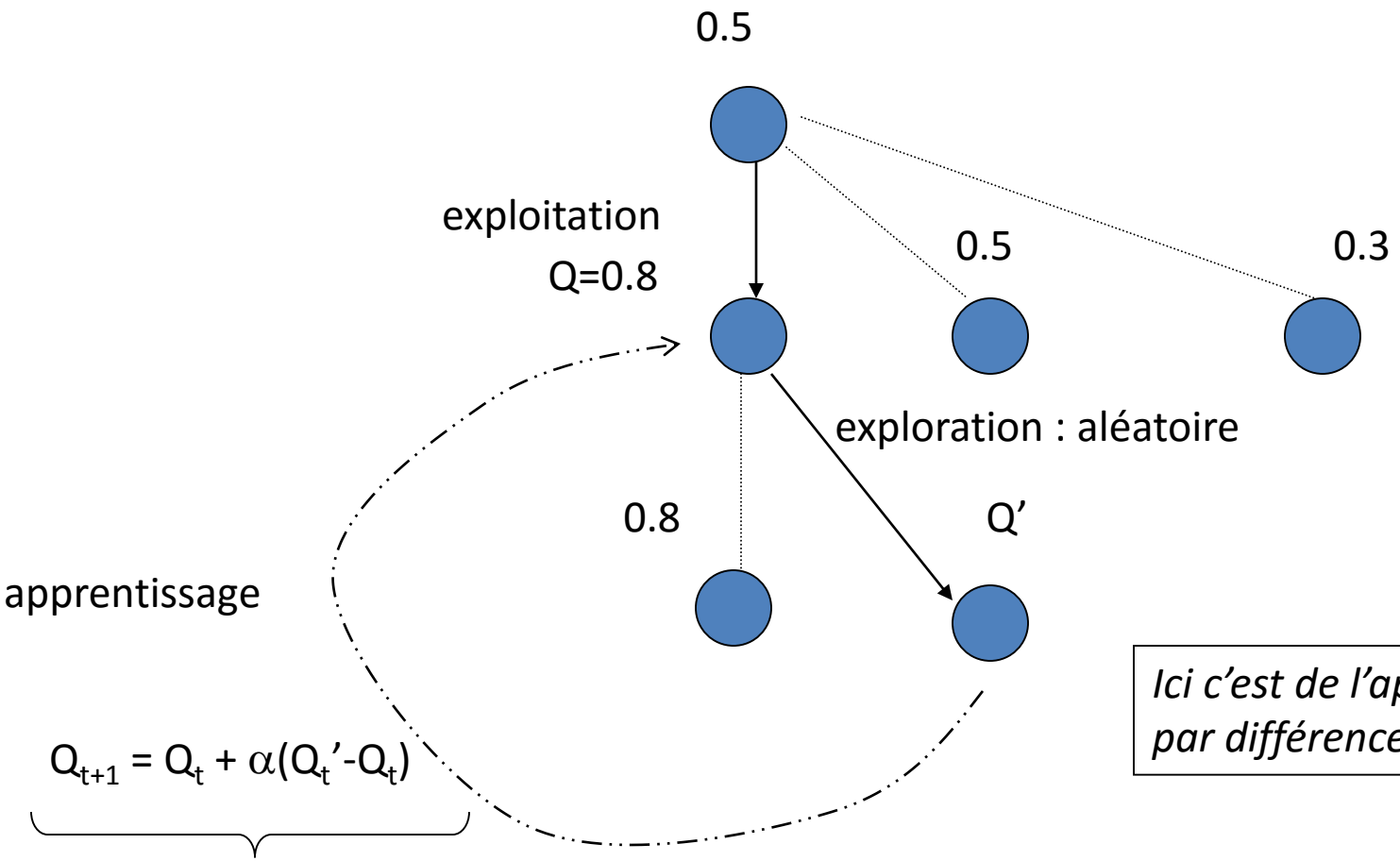
- Joueur parfait ne perd jamais
- Joueur imparfait “pas logique”
  - modèle parfais du joueur (minmax) exclu



Apprentissage par différence temporelle : Exemple : Tic Tac Toe

- Pondérer les états du jeu
  - valeur de l'état = récompense
  - pondération initiale :
    - 1 pour trois croix alignées
    - 0 pour trois rond alignés
    - 0.5 ou aléatoire pour les autres
- exploiter : *aller vers les états de grande valeur*
- explorer : *partir au hasard*
- apprendre : *faire remonter un partie des valeurs de l 'état courant vers les états précédents*

Apprentissage par différence temporelle : Exemple : Tic Tac Toe



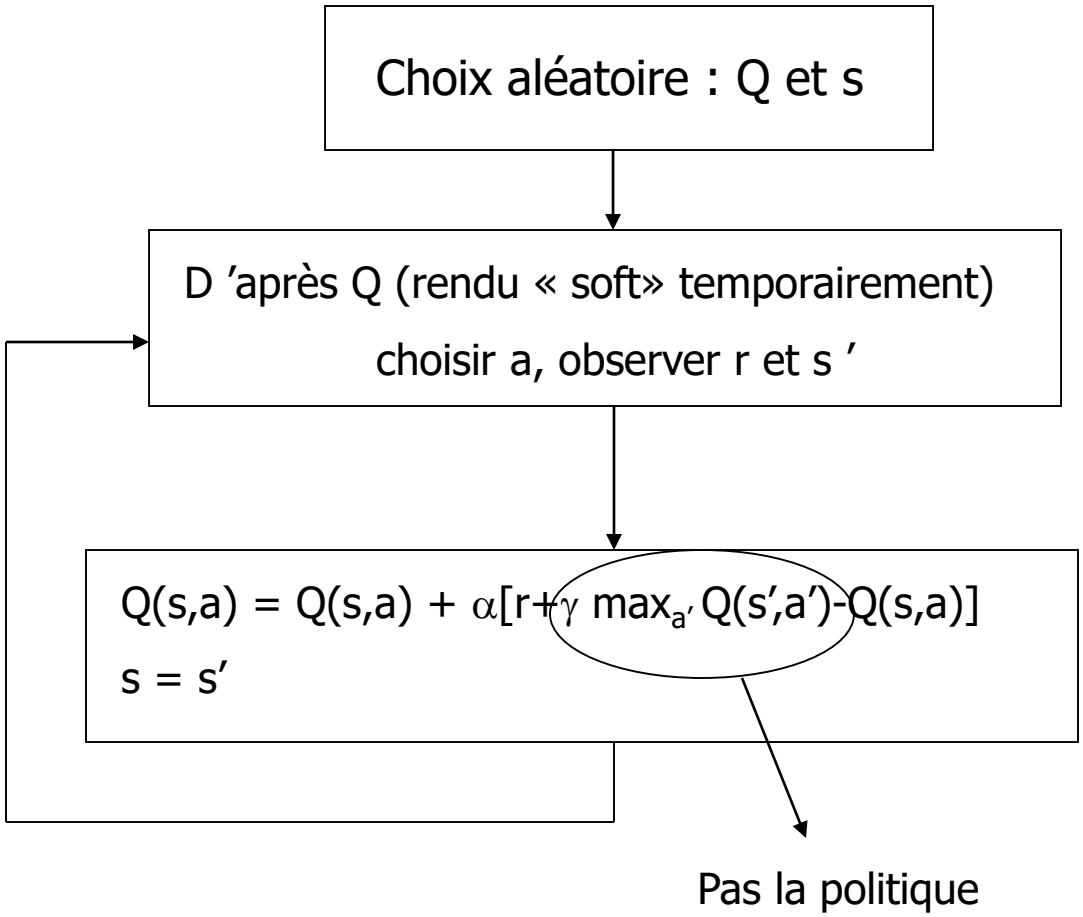
Ici c'est de l'apprentissage par différence temporelle

Bootstrap : l 'estimation de Q dépend de l 'estimation de Q'

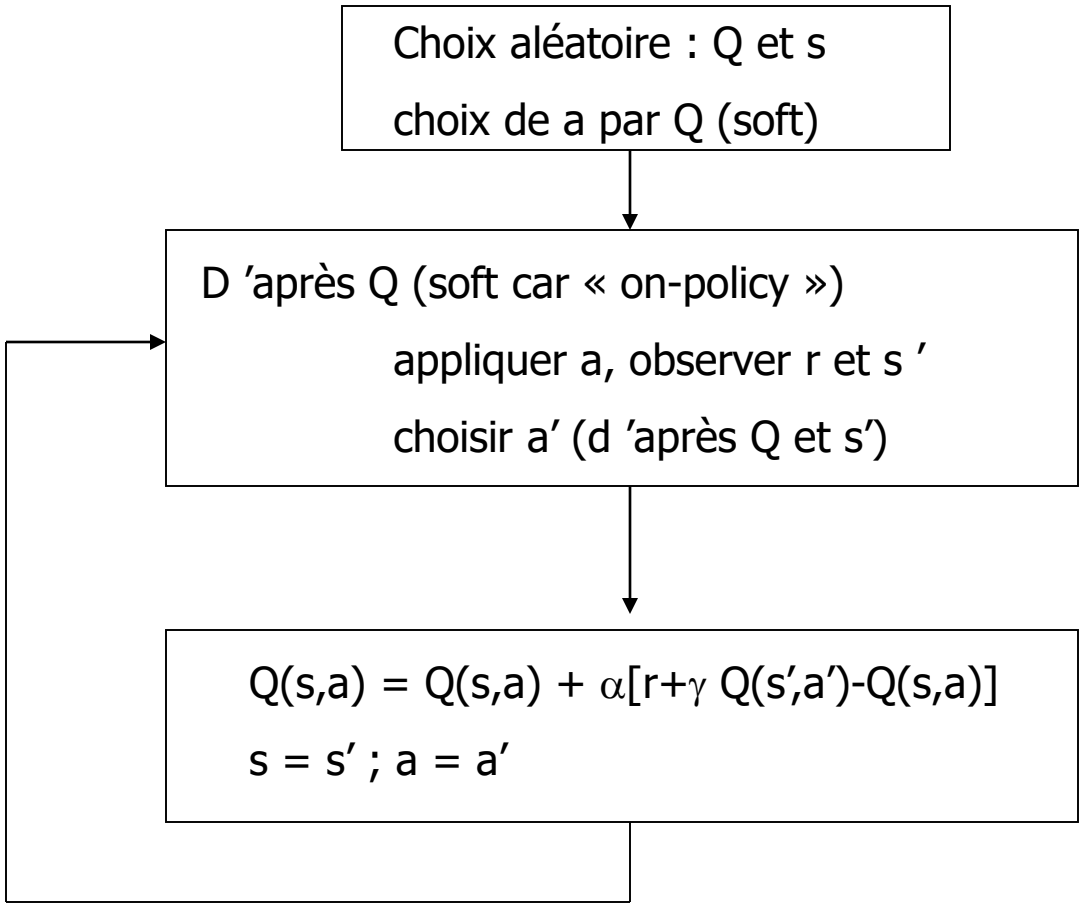
Q-learning : off-policy

- Pendant une simulation à chaque pas :
- $Q(s_t,a_t) = Q(s_t,a_t)+\alpha*[r_{t+1}+\gamma*max_a Q(s_{t+1},a)-Q(s_t,a_t)]$
- max_a recherche l ’action a qui permet d ’obtenir le meilleur Q
- indépendant de la politique que l ’on « joue »

Q-learning : off-Policy



Sarsa : On-Policy (s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})
Apprentisage de Q(s,a)

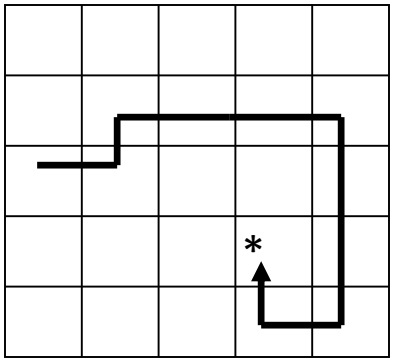


Contrôle du système

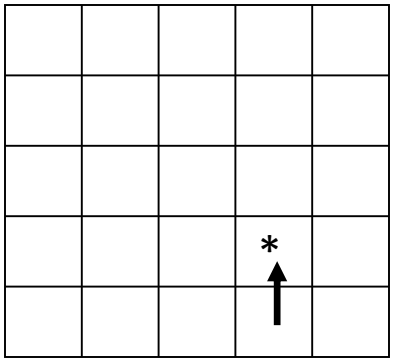
Amélioration de la politique

Sarsa(λ) sur un labyrinthe

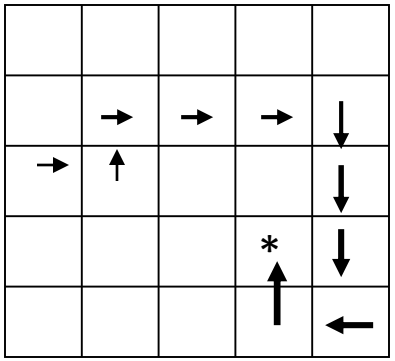
- Les premiers pas sont utilisés pour évaluer Q



chemin



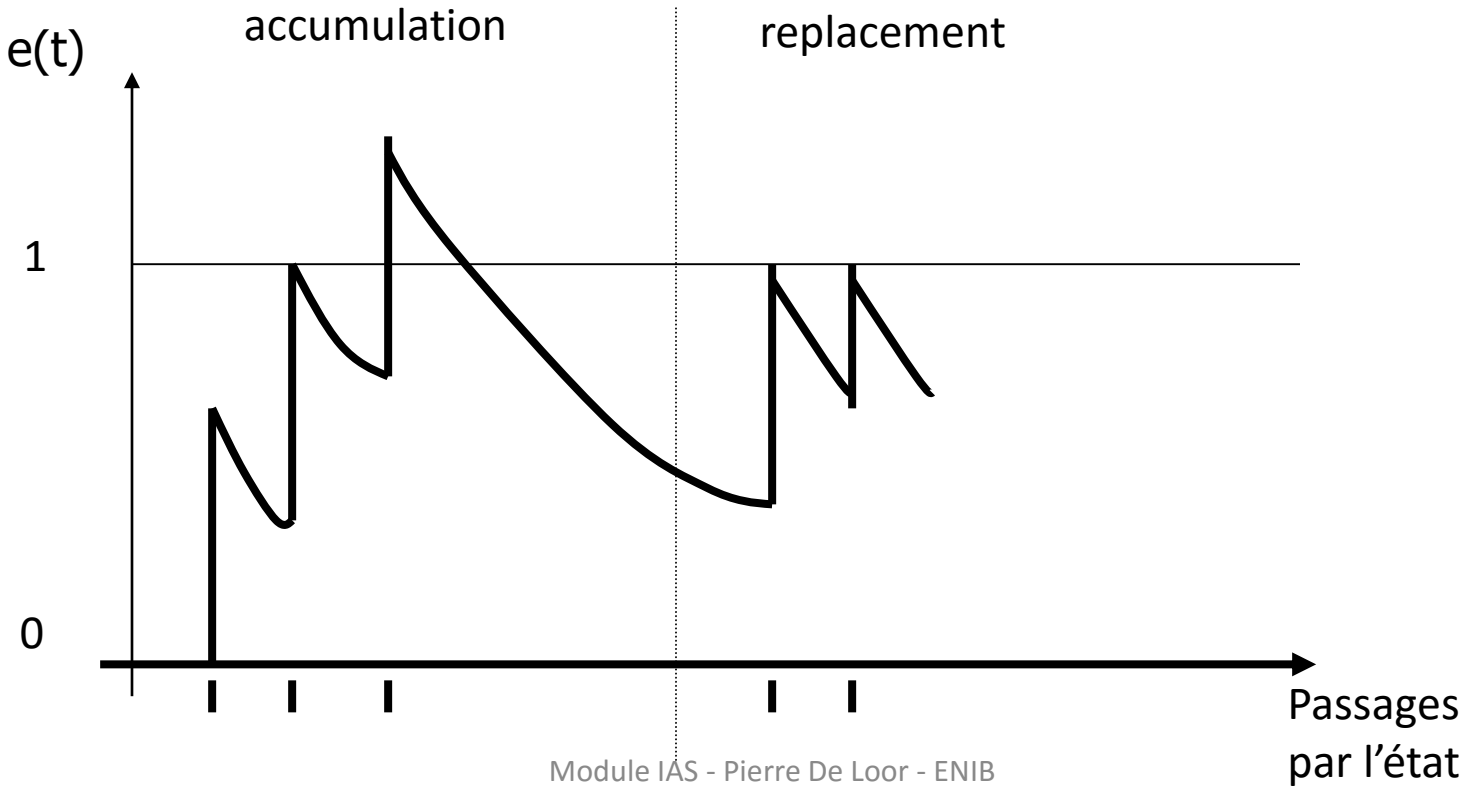
Sarsa



Sarsa(λ)

e(t) trace d ’elligibilité d ’un état (ou état-action)

- Sorte de facteur d’oubli
- Deux possibilités : accumulation ou replacement



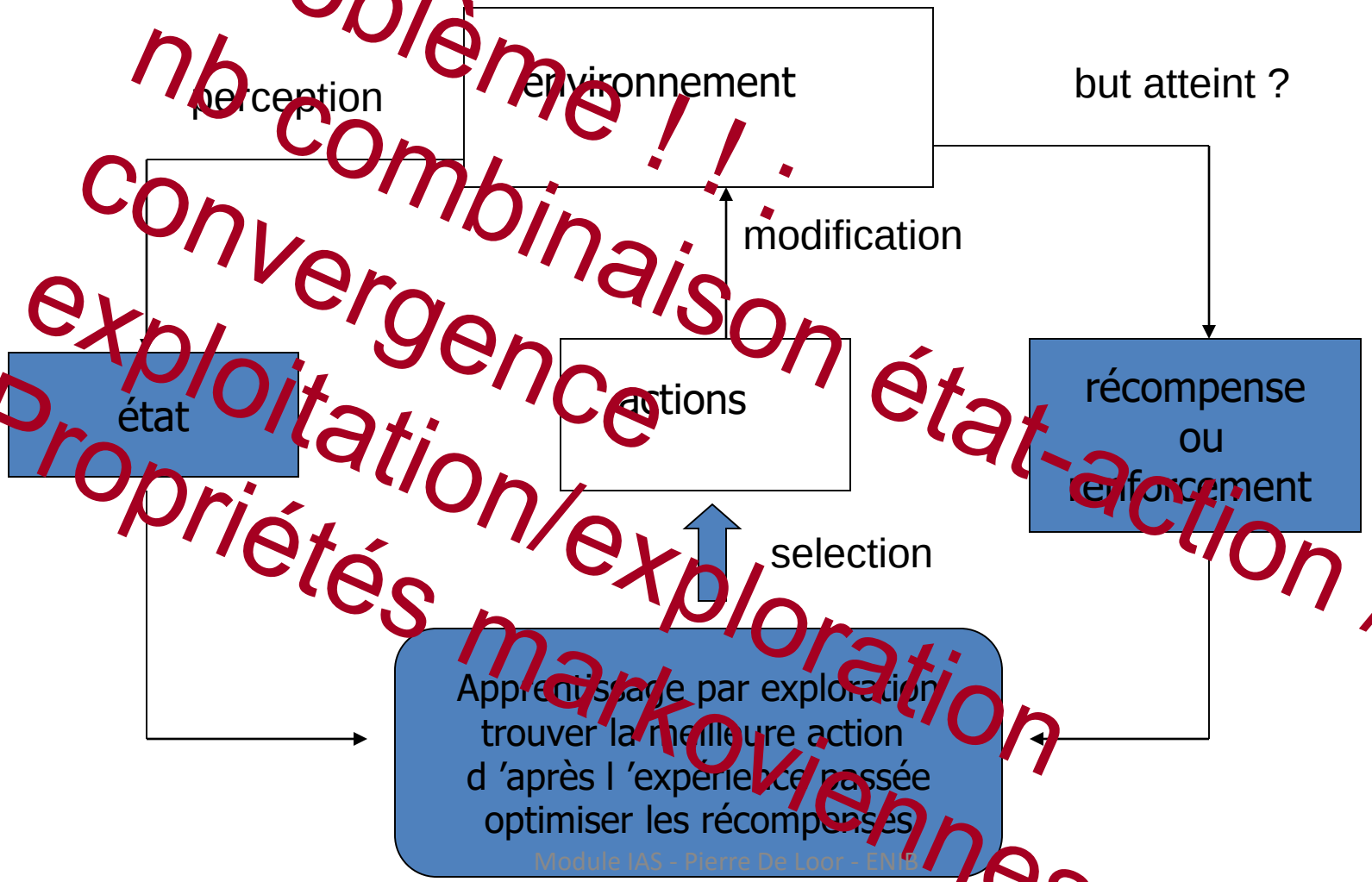
SARSA(λ) : l ’algo complet (en cadeau gratuit)

Initialize Q(s,a) arbitrarily and e(s,a)=0, for all s,a
Repeat (for each episode)
    Initialize s,a
    Repeat (for each step of episode)
        Take action a, observe r, s’
        Choose a’ from s’ using policy derived from Q (e.g., ε-greedy)
        $\delta \leftarrow r + \gamma Q(s,a') - Q(s,a)$
        $e(s,a) \leftarrow e(s,a)+1$ ← accumulation
        For all s, a:
            $Q(s,a) \leftarrow Q(s,a) + \alpha\ \delta\ e(s,a)$ ← Ça c’est intéressant
            $e(s,a) \leftarrow \gamma\lambda e(s,a)$
        $s \leftarrow s'$; $a \leftarrow a'$
    until s is terminal

Le problème d'une approche markovienne

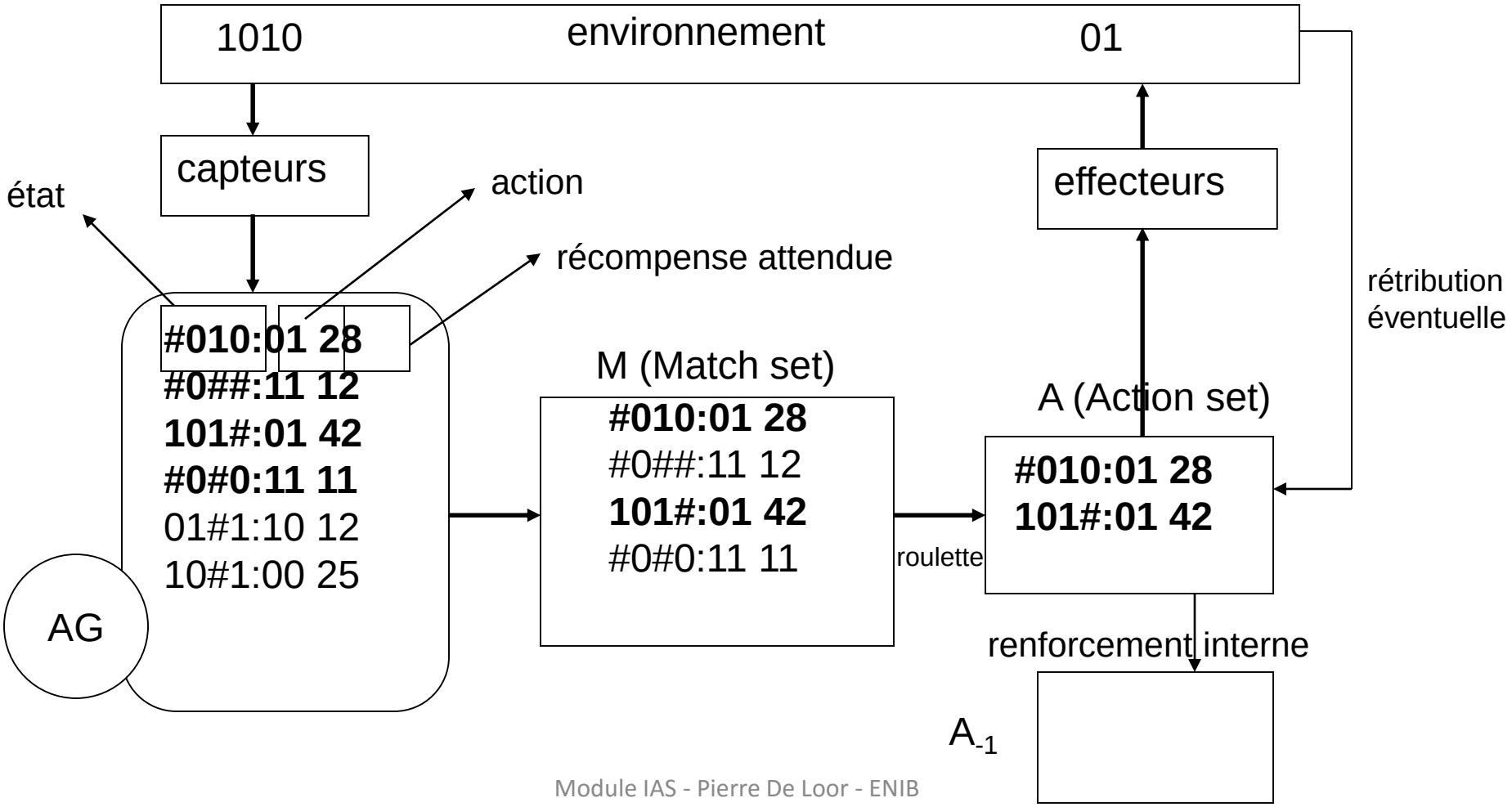
- Chaque état :
  - représentent le passé de façon condensé
  - suffisant pour prendre les décisions
  - propriété Markovienne
    - le choix de la prochaine action à exécuter dépend juste de l'état présent
  - pas d'historique
  - exemple : situation d'un jeu d'echec

Apprentissage par renforcement

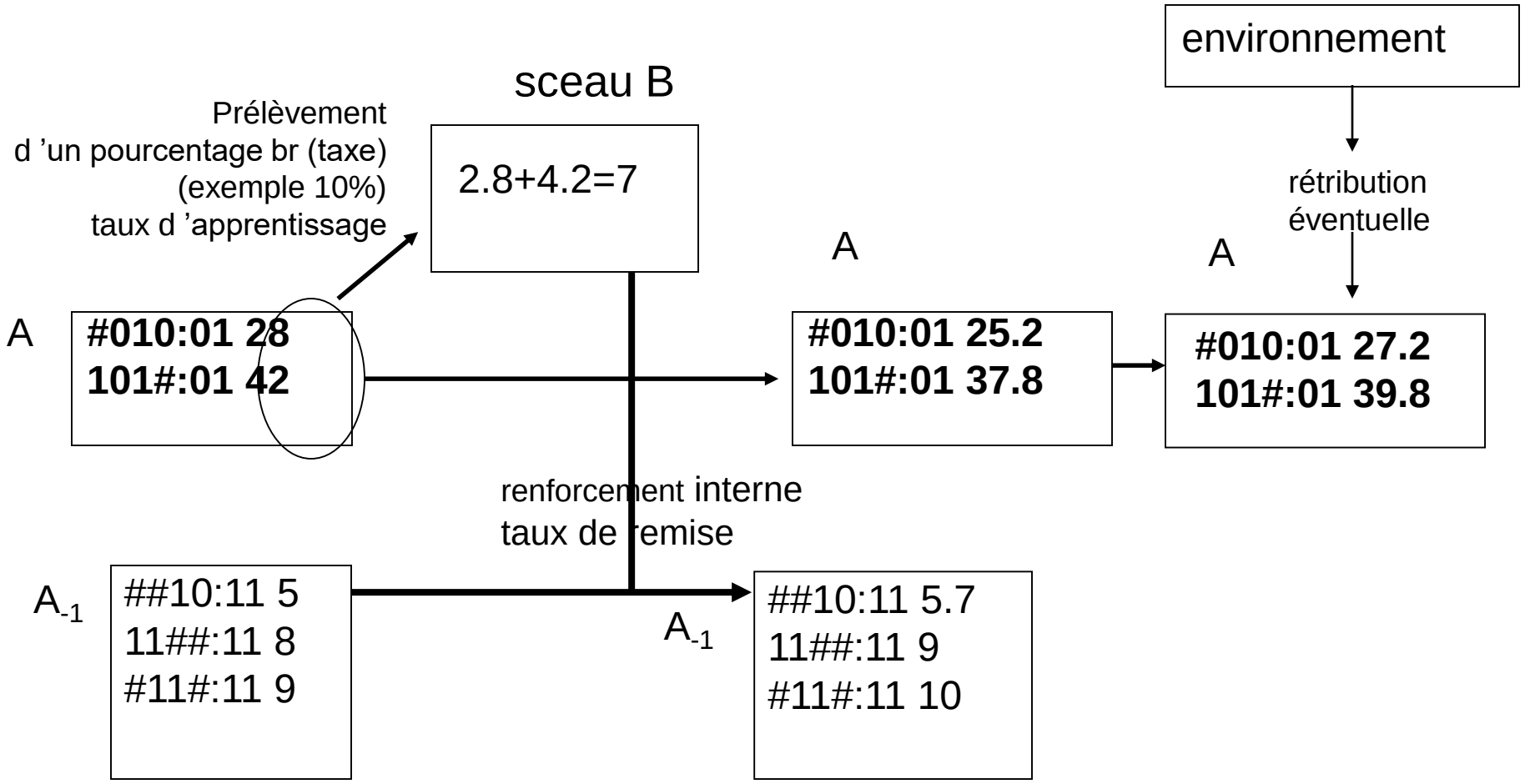


Problème ! ! .
nb combinaison état-action /
convergence
exploitation/exploration
Propriétés markoviennes

Système de classeurs



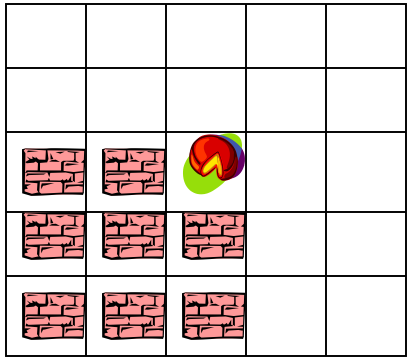
Renforcement des ZCS : Algorithm « Bucket Brigade »

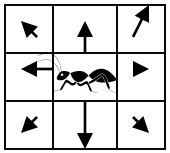


Variante pour convergence : diminution des poids des classifieurs de M-A

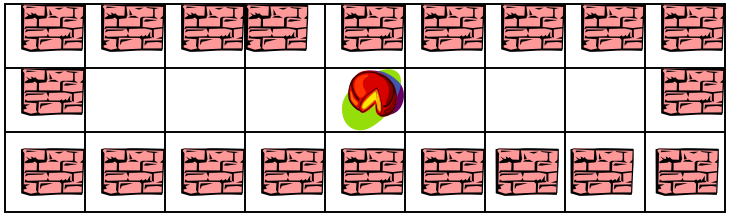
ZCS

- privilégie les règles rapportant plus
- élimine les autres (oubli = surprise = « covering » = pif)
- quelle que soit leur « précision »
- n ’apprend que des environnements markovien





Perception états

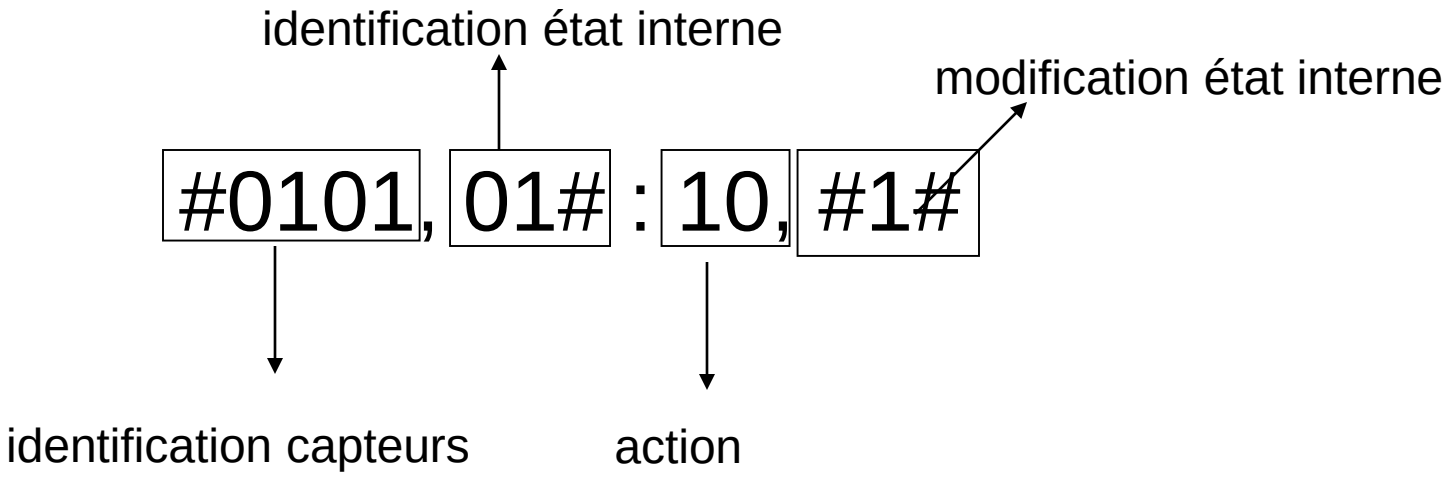


Woods1 : markovien

Woods100 : non markovien

Environnement non markovien : renforcer l ’historique : ZCSM

- Codage d ’une règle



Préserver les classifieurs précis : XCS

- Privilégier les règles les plus « préçises »
- Une règle #010:01
  - + 3 attributs
    - prédiction (de rétribution) : utilisé pour la sélection (M->A)
    - erreur de prédiction
    - fitness (qualité/précision) : utilisé par l ’AG pour la reproduction (faite sur A (niche))
  - mise à jour : Q-learning

Systèmes de classifieurs XCS

- Apprentissage
  - Prédiction : pour toutes les régles de l ’ensemble M (Match set)

$$P(a_i) = \frac{\sum P_i . F_i}{\sum F_i}$$

  - Définition de l ’ensemble A (roulette ou autre d ’après $P(a_i)$)
  - Application de l ’action et réception de la rétribution P
  - Pour tous les éléments de A, mise à jour de la prédiction :

    $p = p + \beta.(P-p)$ [$0<\beta<1$: taux d ’apprentissage]
  - Mise à jour de l ’erreur de prédiction :
    - $e = e + \beta.(|P-p|-e)$
  - Mise à jour de la fitness :
    - $F = F + \beta.(k-F)$ [k fonction décroissante de e (e↘ : F↗)]

Applications

- Jeux vidéos
- Robotique
- Finances / Systèmes experts

Références

- [Cor2002] Antoine Cornuéjols et Laurent Miclet, **Apprentissage artificiel, concepts et algorithmes**, éditions Eyrolles, 2002.
- [Ger2002] P. Gérard, **Systèmes de classeurs : étude de l'apprentissage latent**, thèse de doctorat de l'université de Paris 6.
- « Machine Learning » *Tom Mitchell, McGraw-Hill,* international edition, 1997.
- [pes99] Peshkin, Leonid and Meuleau, Nicolas and Kaelbling, Leslie P., **Learning Policies with External Memory,ml99**, editor : Bratko, I. and Dzeroski, S, pp 307-314, 1999.
- [Ria2000] **Algorithmes d 'apprentissages et applications**, revue d 'intelligence artificielle, volume 14, n°3, édition hermes, 2000
- Cédric Sanza, **Evolution d 'entités virtuelles cooperatives par système de classifieurs**. These de doctorat de l 'université de Toulouse, 2001.
- [Sut2000]R.S.Sutton and A.G.Barto*,* **Reinforcement Learning,** MIT Press, 2000.
- [Ski38] B.F. Skinner, **The Behavior of organisms**. Appleon Century Croft, New York, 1938.
- [Thor11] Thorndike, **Animal Intelligence**, 1911.
- [Tol30] E.C. Tolman et C.H. Honzik. **Insight in Rats**. University of California Publications in Psychology, 4: 215-232, 1930.
- [Whi91] Whitehead and Ballard, **Learning to Perceive and Act by Trial and Error**, Machine Learning, 7, 45-83, 1991.
- **Apprentissage automatique et évolution artificielle**, revue extraction des connaissances et apprentissage, Volume1, n°3, éditions hermes, 2001.
- Algorithmes d 'apprentissages et applications, revue d 'intelligence artificielle, volume 14, n°3, édition hermes, 2000.