Reinforcement Learning

Jean-Paul Comet (projet Bioinfo Formelle)

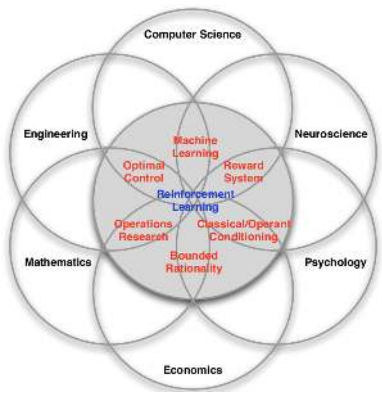
EPU dept GB-BIMB - 5ème année
Université Côte d'Azur

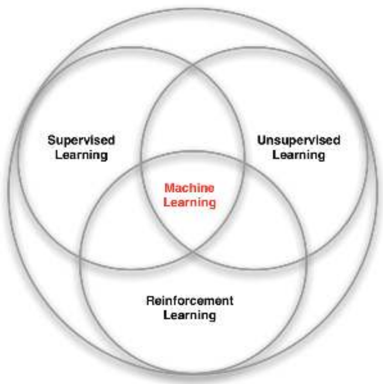
18 décembre 2023

inspired from David Silver, Antoine Cornuéjols, Sutton and Barto, 1998
http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html...

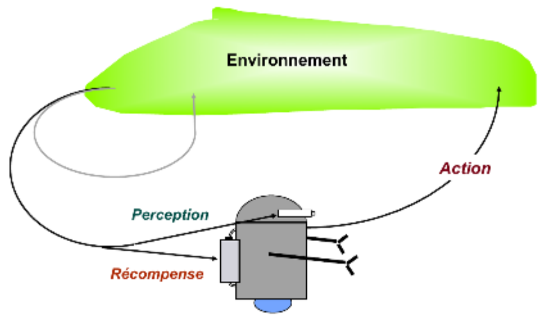
1 Introduction
2 Notions d'utilité et de politique
3 ε-greedy Algorithm
4 Markov Decision Processes : MDP
5 Quand le mode est connu
6 Quand le monde est connu (et monde fini)
7 Apprentissage avec généralisation

Introduction : general schema





Introduction : general schema



Characteristics of Reinforcement Learning

What makes reinforcement learning different from other machine learning paradigms ?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives

Introduction : general schema

Exploration vs. Exploitation Dilemma

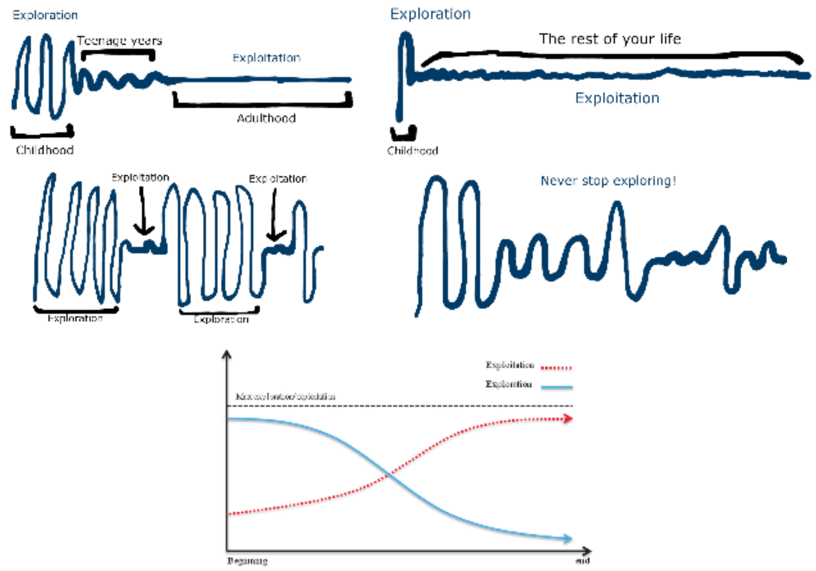
- Online decision-making involves a fundamental choice :
  Exploitation Make the best decision given current information
  Exploration Gather more information
- The best long-term strategy may involve short-term sacrifices
- Gather enough information to make the best overall decisions

Examples

- Restaurant Selection
  Exploitation Go to your favourite restaurant
  Exploration Try a new restaurant
- Online Banner Advertisements
  Exploitation Show the most successful advert
  Exploration Show a different advert
- Oil Drilling
  Exploitation Drill at the best known location
  Exploration Drill at a new location
- Game Playing
  Exploitation Play the move you believe is best
  Exploration Play an experimental move

Introduction : general schema

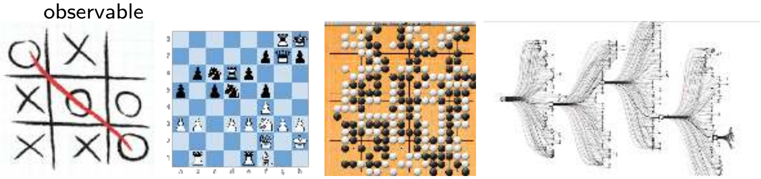
IA pour la bio
J-P Comet
Introduction
Utilité/politique
ε-greedy
MDP
Env. connu
Env. inconnu
Généralisation



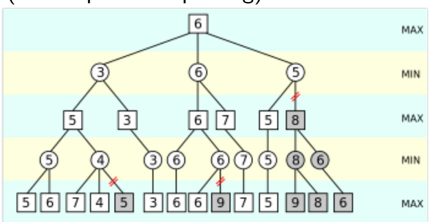
Introduction : general schema

IA pour la bio
J-P Comet
Introduction
Utilité/politique
ε-greedy
MDP
Env. connu
Env. inconnu
Généralisation

1. Application to conventional Games : All aspects of the states are fully observable



2. Technique : MinMax (with Alpha-Beta pruning) Effective for
   - Modest branching factor
   - When a good evaluation function is available



Go : branching factor 250 [Chess : 31] + no good eval. function

3. Generalisation
   - Often, no "endgame", so no exploration to the leaves possible
     A reinforcement signal from time to time
   - We don't play against a MIN type opponent. So no worst case.
     We want to maximize the expectation of gain in an unknown environment (at the beginning)

Introduction : Basic notations

IA pour la bio
J-P Comet
Introduction
Utilité/politique
ε-greedy
MDP
Env. connu
Env. inconnu
Généralisation

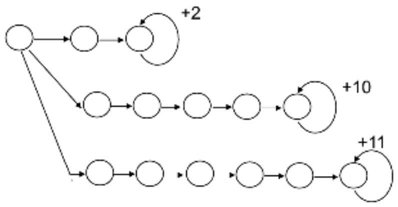
- Discrete Time : t
- States : $s_t \in S$
- Actions : $a_t \in \mathcal{A}(s_t)$
- Rewards : $r_t \in \mathcal{R}(s_t)$
- The agent : $s_t \rightarrow a_t$
- The environment : $(s_t, a_t) \rightarrow s_{t+1}, r_{t+1}$
- Policy : $\pi_t : S \rightarrow \mathcal{A}$ $\quad T$ (transitions) + R (rewards)
  with $\pi_t(s, a)$ = probability of $a_t = a$ when $s_t = s$
- Transitions and Rewards depend only on current state and action :
  Markovian process

Introduction : Critères de gains

- Horizon fini : $\sum_{t=0}^{k} r_t = r0 + r_1 + \ldots + r_k$
- Horizon infini avec intérêt $\sum_{t=0}^{\infty} \gamma^t r_t = r0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ldots$
- En moyenne : $\lim_{k \to \infty} \frac{1}{k} \sum_{t=0}^{k} r_t$

Quelle est la meilleure stratégie ?

$m = 4, \gamma = 0.9$



$\sum_{t=0}^{m} r_t$	$\sum_{t=0}^{\infty} \gamma^t r_t$	$\lim_{k \to +\infty} \frac{1}{k} \sum_{t=0}^{k} r_t$
4	16.20	2
0	**59.04**	10
0	58.45	**11**

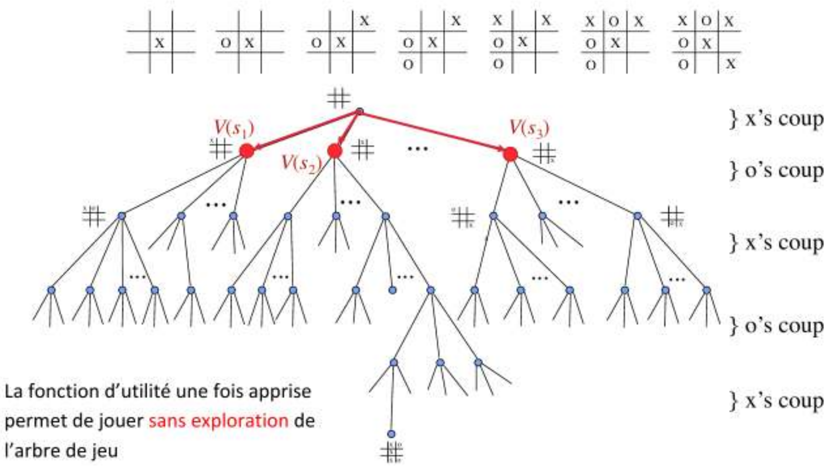
Introduction : éléments de base

- **Politique :**
  ensemble d'associations situation → action (une application)
  Une simple table ... un algorithme de recherche intensive
  Eventuellement stochastique
- **Fonction de renforcement :**
  - Définit implicitement le <u>but poursuivi</u>
  - Une fonction : (état, action) → récompense $\in \mathcal{R}$
- **Fonctions d'évaluation $V(s)$ ou $Q(s, a)$:**
  Récompense accumulée sur le long-terme
- **Modèle de l'environnement :**
  Fonctions T et R : (état(t), action) → (état(t+1), récompense)

Introduction : La notion d'utilité

Intuition de la **La notion d'utilité**

- Choisir une action sans avoir besoin de faire une exploration (simulée) en avant
- Il faut donc disposer d'une fonction d'évaluation locale résumant une espérance de gain si l'on choisit cette action : fonction d'utilité
- Il faut apprendre cette fonction d'utilité : apprentissage par renforcement



ε-greedy algorithm

The ε-greedy algorithm takes the best action most of the time, but does random exploration occasionally.

The action value is estimated according to the past experience by averaging the rewards associated with the target action a that we have observed so far (up to the current time step t) :

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{\tau=1}^{t} r_\tau \mathbb{1}[a_\tau = a]$$

where $\mathbb{1}$ is a binary indicator function and $N_t(a)$ is how many times the action a has been selected so far, $N_t(a) = \sum_{\tau=1}^{t} \mathbb{1}[a_\tau = a]$.

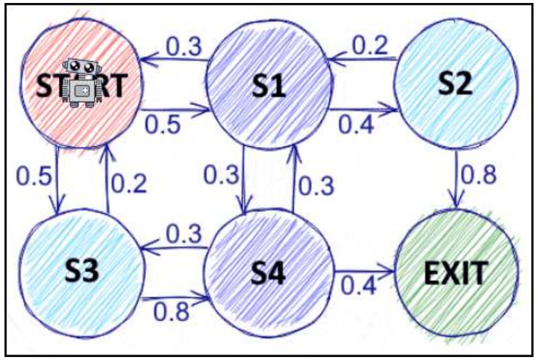
ε-**greedy algorithm :**

- with probability ε : we take a random action,
- with probability $1 - \varepsilon$: we pick the best action that we have learnt so far :

$$\hat{a}_t^* = \arg\max_{a \in \mathcal{A}} \hat{Q}_t(a)$$

TD ε-gready algorithm.

Markov Decision Processes : MDP



1. Given any state and action, s and a, the *transition probability* of each possible next state, s', is
$$\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$$
2. Given any current state and action, s and a, together with any next state, s', the expected value of the next reward is
$$\mathcal{R}^a_{ss'} = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\}$$
3. One can also define the reward dynamics in terms of the expected next reward given just the current state and action, that is, by $\mathcal{R}^a_s = E\{r_{t+1} \mid s_t = s, a_t = a\}$. This quantity is related to our as follows :
$$\mathcal{R}^a_s = \sum_{s'} \mathcal{P}^a_{ss'} \mathcal{R}^a_{ss'}$$

Value functions

Value functions : estimate how good it is for the agent to be in a given state

- "how good" : defined in terms of future rewards that can be expected, or, to be precise, in terms of expected return.
- A policy, π , is a mapping from each state, $s \in \mathcal{S}$, and action, $a \in \mathcal{A}$, to the probability $\pi(s, a)$ of taking action a when in state s.

state-value function for policy π

Value of a state s under a policy π, denoted $V^\pi(s)$, is the expected return when starting in s and following π thereafter. For MDPs, we can define formally as
$$V^\pi(s) \;=\; E_\pi\{\mathcal{R}_t \mid s_t = s\} \;=\; E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\}$$

action-value function for policy π

the value of taking action a in state s under a policy π, denoted $Q^\pi(s, a)$ is the expected return starting from s, taking the action a, and thereafter following policy π :
$$Q^\pi(s,a) \;=\; E_\pi\{\mathcal{R}_t \mid s_t = s, a_t = a\} \;=\; E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\}$$

Value functions : recursive relationships

$$\begin{aligned}
V^\pi(s) &= E_\pi\{\mathcal{R}_t \mid s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\} \\
&= E_\pi\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s\right\} \\
&= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s'\right\}\right] \\
&= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')\right]
\end{aligned}$$

We also have :

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

Optimal value function $V(s)$ & optimal policy

- Une politique est une application $\pi : S \rightarrow A$
- Valeur optimale d'un état :
$$V^*(s) = \max_\pi V_\pi(s) = \max_\pi E_\pi\left(\sum_{t=0}^{+\infty} \gamma^t r^t\right)$$
- La fonction de valeur optimale V^* est unique
$$V^*(s) = \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s')\right]$$
- Une politique stationnaire optimale existe :
$$\pi^*(s) = a^* = \underset{a}{ArgMax} \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s')\right]$$

(à connaître : $\mathcal{P}^a_{ss'}$ and $\mathcal{R}^a_{ss'}$)

Optimal value function $Q(s,a)$ & optimal policy

- Valeur optimale d'une action dans un état :

$$Q^*(s,a) = \max_\pi Q_\pi(s,a) = E_\pi\{r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a\}$$
$$= \sum_{s'} \mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s',a')\right]$$

- Théorème :

$$\pi^*(s) = \underset{a}{ArgMax}\, Q^*(s,a) \qquad \text{est une politique optimale}$$

(rien à connaître)

Quand le monde est connu : $\mathcal{P}^a_{ss'}$ and $\mathcal{R}^a_{ss'}$

Algorithme d'itération de valeur

Évaluation de politique

1. **Évaluation de politique :** Pour une politique donnée π, calculer la fonction d'utilité d'état $V_\pi(s)$:
   State-value function for policy π :

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

   Bellman equation for V_π :

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

   $\Rightarrow$ a system of $|S|$ simultaneous linear equations

2. **Évaluation itérative d'une politique (prog. dyn.)**
   Principe : l'équation de point fixe de Bellman peut fournir une procédure itérative d'approximation successive de la fonction d'utilité $V\pi$.

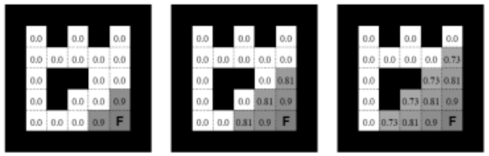
$$V_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma V_k(s')\right]$$

   Propagation :

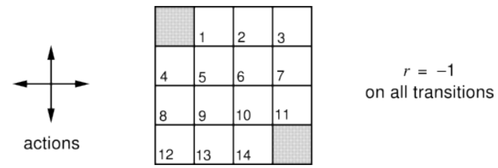
$$V_0 \to V_1 \to \ldots \to V_k \to V_{k+1} \to \ldots \to V^\pi$$

Prog. dynamique : Évaluation de politique (Algo V)

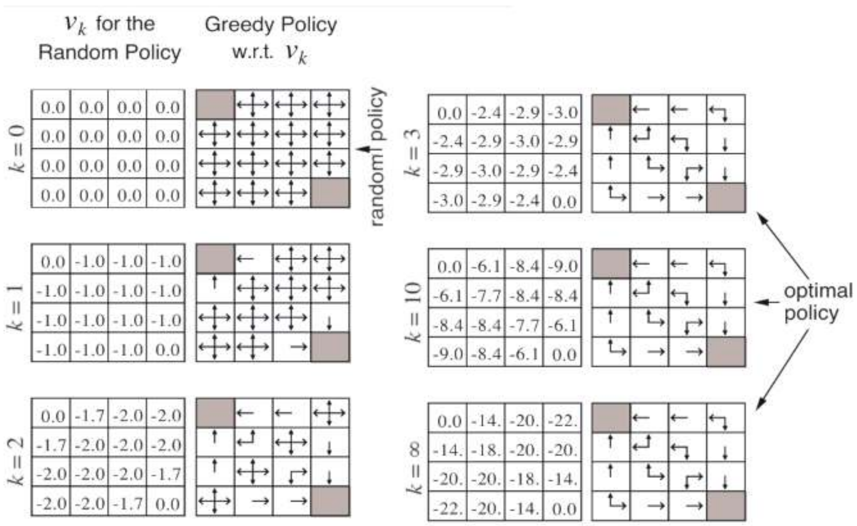
Input : π, the policy to be evaluated
Algorithm parameter :
a small threshold $\theta > 0$ determining accuracy of estimation
Initialize an array $V(s)$, for all $s \in S^+$, arbitrarily
Repeat :
$\Delta \leftarrow 0$
For each $s \in S$:
$v \leftarrow V(s)$
$V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V(s')]$
$\Delta \leftarrow max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \sim v^\pi$



DP : Evaluation d'une politique : exemple



DP : Evaluation d'une politique : exemple



Prog. dynamique : Évaluation de politique (Algo Q)

IA pour la bio
J-P Comet
Introduction
Utilité/politique
ε-greedy
MDP
Env. connu
Env. inconnu
Généralisation

Input : π, the policy to be evaluated
Algorithm parameter :
a small threshold $\theta > 0$ determining accuracy of estimation
Initialize an array $Q(s, a)$, for all $s \in S^+$, $a \in A$ arbitrarily
Repeat :
$\Delta \leftarrow 0$
For each $s \in S$:
For each $a \in A$:
$q \leftarrow Q(s, a)$
$Q(s, a) \leftarrow R(s, a) + \gamma \sum_{s'} T(s, a, s')\, max_{a'}\, Q(s', a')$
$\Delta \leftarrow max(\Delta, |q - Q(s, a)|)$
until $\Delta < \theta$
Output $Q \sim Q^\pi$

Comment améliorer une politique ?

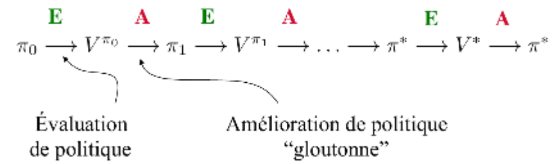
IA pour la bio
J-P Comet
Introduction
Utilité/politique
ε-greedy
MDP
Env. connu
Env. inconnu
Généralisation

Lorsqu'on change la valeur V, la politique n'est peut être pas la meilleure.
Avec les nouvelles valeurs de V, la meilleure politique est :

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

Itération de politique

IA pour la bio
J-P Comet
Introduction
Utilité/politique
ε-greedy
MDP
Env. connu
Env. inconnu
Généralisation



Initialisation arbitraire de π
Faire
calcul de la fonction de valeur avec π

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

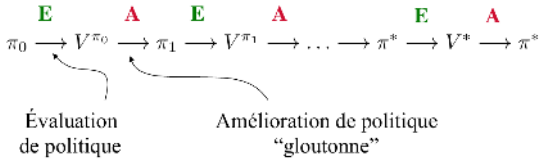
Amélioration de la politique à chaque état

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

$\pi' := \pi$
Jusqu'à ce qu'aucune amélioration ne soit possible

➥ Garantie de convergence vers une politique optimale

Itération de politique



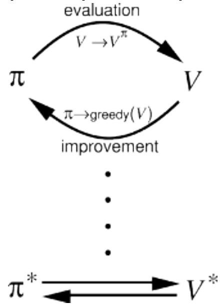
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

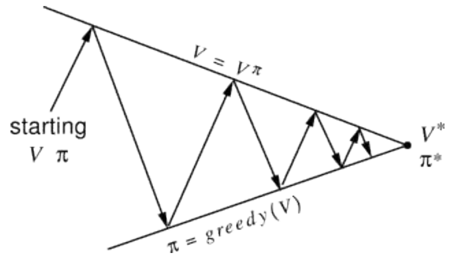
2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s',r \mid s,\pi(s))\big[r + \gamma V(s')\big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining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

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

Itération généralisée de politique

Generalized Policy Iteration (GPI) : Toute interaction d'étape d'évaluation de politique et d'étape d'amélioration de politique indépendamment de leur granularité





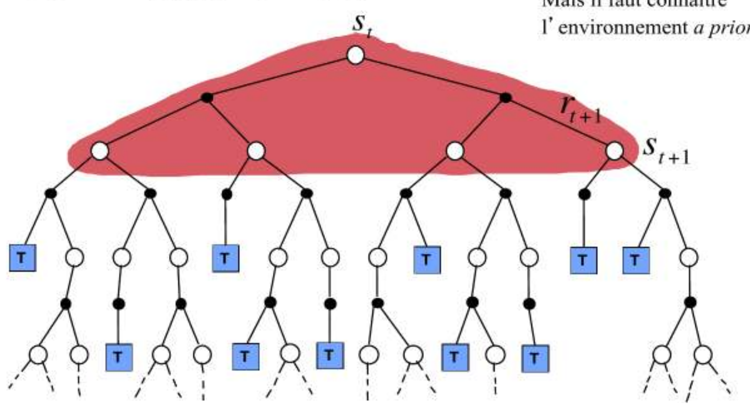
On peut se permettre aussi de ne pas calculer à chaque étape les nouvelles valeurs de V pour tous les états. Nombreux travaux, nombreuses variantes...

Le modèle du monde est inconnu

TD learning : cf. Dynamic Programming

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

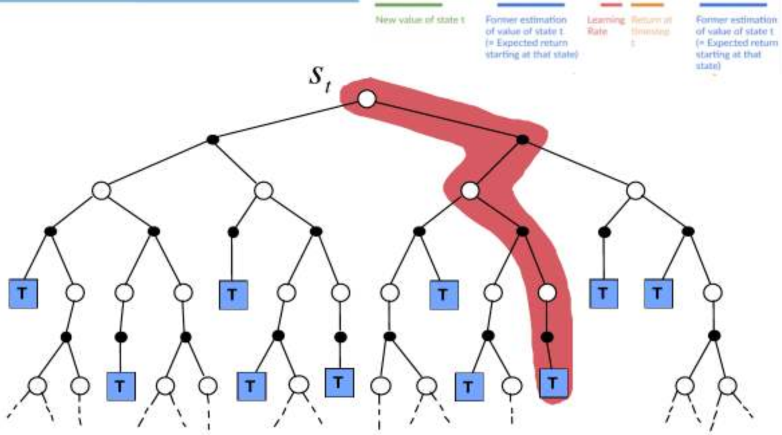
On calcule l'espérance.
Mais il faut connaître
l'environnement *a priori*.



Le modèle du monde est inconnu

Monte Carlo method

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

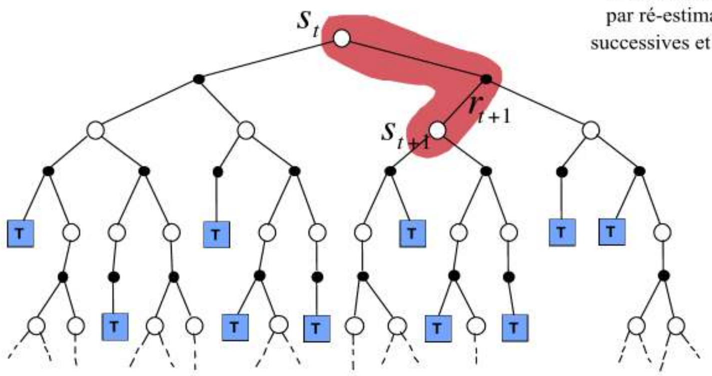


Le modèle du monde est inconnu

TD learning : Simplest TD Method

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

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



IA pour la bio
J-P Comet
Introduction
Utilité/politique
ε-greedy
MDP
Env. connu
Env. inconnu
Généralisation

TD learning

évaluation par méthode des différences temporelles

Évaluation de politique :

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

Simple every - visit Monte Carlo method :

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

R_t (cible) : le vrai gain sur une durée t

The simplest TD method, TD(0) :

$$V(s_t) \leftarrow V(s_t) + \alpha[\underbrace{r_{t+1} + \gamma V(s_{t+1})}_{\text{cible : une estimation du gain}} - V(s_t)]$$

IA pour la bio
J-P Comet
Introduction
Utilité/politique
ε-greedy
MDP
Env. connu
Env. inconnu
Généralisation

Monte Carlo method

Algorithm parameter: small $\varepsilon > 0$

Initialize:

- $\pi \leftarrow$ an arbitrary ε-soft policy
- $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
- $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

- Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
    - Append G to $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
    - Modifies π — $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
    - For all $a \in \mathcal{A}(S_t)$:

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

exploration ε-greedy

IA pour la bio
J-P Comet
Introduction
Utilité/politique
ε-greedy
MDP
Env. connu
Env. inconnu
Généralisation

Apprentissage avec généralisation

- Si l'espace S (ou $S \times A$) est trop important pour l'utilisation d'une table mémorisant les prédictions
- Deux options :
  - Utilisation d'une technique de généralisation dans l'espace S ou l'espace $S \times A$ (e.g. réseau de neurones, ...)
  - Utilisation d'une technique de regroupement d'états en classes d'équivalence (même prédiction et même action générée).

Approximation de la fonction $V(s)$:

- Évaluation de politique :
  pour une politique donnée π, calculer la fonction d'utilité V^π

Mais

- avant, les fonctions d'utilité étaient stockées dans des tables.
- Maintenant, l'estimation de la fonction d'utilité au temps t, V_t , dépend d'un vecteur de paramètres θ_t, et seul ce vecteur de paramètres est mis à jour.
  e.g., θ_t pourrait être le vecteur de poids de connexions d'un réseau de neurones.

Généralisation : Backups as Training Examples

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

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

As a training example :

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

Généralisation : n'importe quelle méthode inductive ?

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