Apprentissage par renforcement

Cours 2: MDPs & Plannification

Sylvain Lamprier

UE RLD - Master DAC

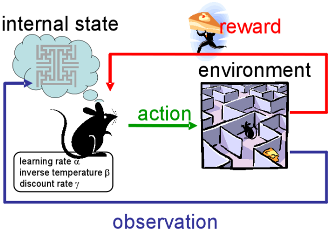
2020

Principe

Lexique

- Agent, Environnement
- Etat : ce qui conditionne le futur (hypothèse de Markov)
- Observation : ce que perçoit l'agent
- Action : interaction de l'agent avec l'environnement
- Récompense (*reward*) : quantité perçue après chaque action
- Politique (*policy*) : fonction de sélection de l'action selon l'état

Objectif : trouver une politique qui permet de maximiser l'ensemble des récompenses reçues

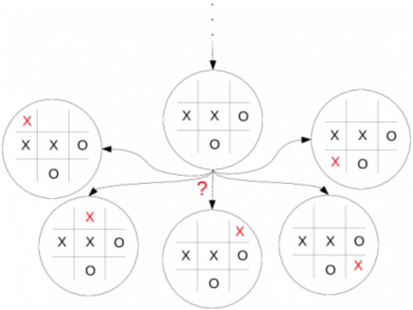


Objectif : adaptation du système à son environnement

Reproduction artificielle du “conditionnement”

Comment :

- enseigner un comportement à l’aide de récompenses ?
- réagir à une situation donnée ?
- agir de manière à maximiser les récompenses ?

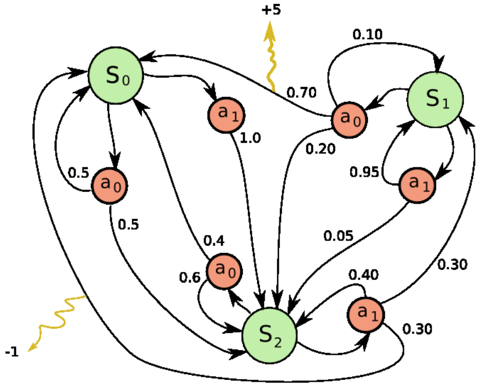


Markov Decision Process

Nous définissons un MDP comme un quadruplet $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$:

- $\mathcal{S}$ est l'ensemble d'états (*states*)
- $\mathcal{A}$ est l'ensemble des actions. On note $\mathcal{A}(s)$ l'ensemble des actions dans l'état s tel que $\mathcal{A}(s) \in \mathcal{A}$
- $\mathcal{P}$ est la fonction de transition : $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0; 1]$. Cette fonction définit une distribution de probabilité sur les transitions $\mathcal{P}(s, a, s') = P(s' = s_{t+1} | s = s_t, a_t = a)$.
- $\mathcal{R}$ est la fonction de récompense (*reward*). $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow R$ telle que $R(s, a, s') = E[r_t | s_t = s, a_t = a, s_{t+1} = s']$

MDP



+5
0.10
S_0
S_1
a_1
a_0
0.70
1.0
0.20
0.5
a_0
0.95
a_1
0.5
0.4
0.05
a_0
0.30
0.6
0.40
a_1
S_2
0.30
-1

Hypothèse de Markov

On a fait l'hypothèse que :

$$P(s_{t+1}|s_0, a_0, s_1, a_1, ..., s_t, a_t) = P(s_{t+1}|s_t, a_t) \quad (1)$$

- L'état suivant ne dépend que de l'état courant (et de l'action)
- Hypothèse de Markov d'ordre 1

L'hypothèse est fausse si :

- L'état ne contient pas toute l'information nécessaire
- Les transitions dépendent du temps
- Les transitions dépendent d'un "joueur adverse" qui appartient à l'environnement...
- ...mais bon, commençons par le plus simple

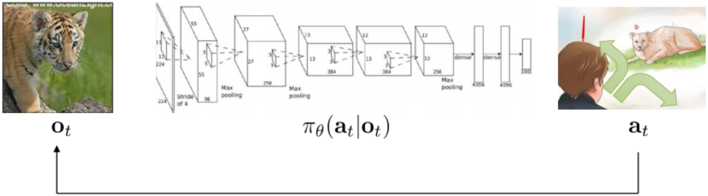
Politique

On définit une politique (stratégie ou *policy*) π comme étant la fonction :

$$\begin{aligned} \pi : \mathcal{S} \times \mathcal{A} &\to [0;1] \\ (s,a) &= P(a_t = a | s_t = s) \end{aligned}$$

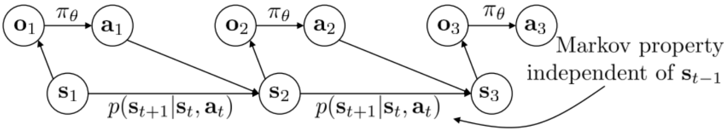
Une politique est *Non déterministe*. Par abus de notation, nous noterons $\pi(s)$ l'action choisie pour l'état s.

Observations vs States



$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

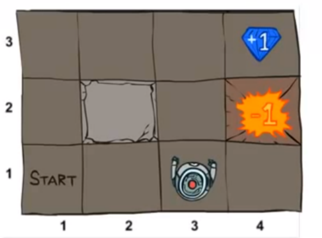
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)



Définitions

- Si le MDP possède un ensemble d'états finaux $\mathcal{F} \subset \mathcal{S}$ alors le problème est un problème *à horizon fini*.
- Nous considérons ici que le processus est *stationnaire* i.e : $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$ ne varient pas dans le temps.
- Notons aussi que nous considérons pour le moment que les états et les actions sont discrètes et en nombre fini.
- On considère aussi pour l'instant que le modèle de transitions est entièrement connu
- et que les états sont toujours observables (i.e., $o_t = s_t$)

Exemple : Grid World



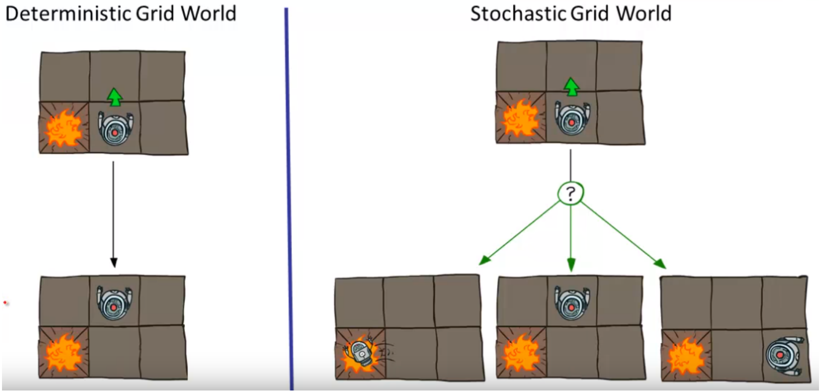
Rewards :

- $+1$ sur la case diamant (état final)
- -1 sur la case feu (état final)

Déplacements :

- Nord, Est, Sud, Ouest
- Pas de déplacement si mur (l'action n'a alors pas d'effet)
- Mouvements bruités : 80% de déplacements dans la direction demandée, 10% dans chacune des deux directions à 90 degrés de la direction choisie

Exemple : Grid World



A*

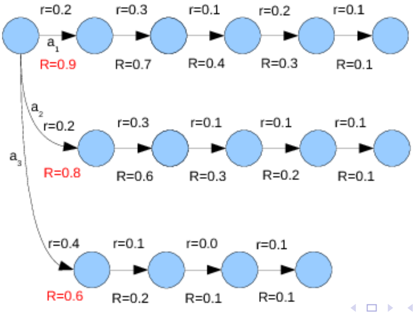
RL

Problématique

Comment choisir une action ?

- Regarder la récompense liée à chaque action
- Mais aussi les récompenses futures !

$\Rightarrow$ **fonction de valeur** d'états (ou d'état/action) : indication sur le long terme des récompenses attendues ($\neq$ récompenses immédiates)

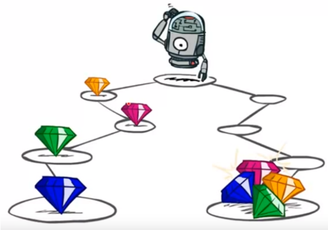


Récompense

On définit la récompense R_t comme étant la somme des retours obtenus à partir de l'instant t :

$$R_t = \sum_{i=0}^{i=+\infty} \gamma^i r_{t+i}$$

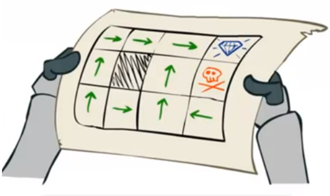
- r_t est la récompense immédiate au temps t
- γ est appelé le facteur d'amortissement des récompenses (discount) et permet de prendre en compte les récompenses à plus ou moins long terme.



But de l'apprentissage par renforcement

Le but de l'apprentissage par renforcement consiste à chercher une politique qui permet d'obtenir le plus de récompense. Cette politique est appelé **politique optimale** et est notée π^*. Par définition :

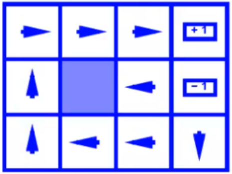
$$\pi^* = argmax_\pi R_0$$



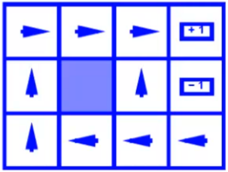
Optimal policy when R(s, a, s') = -0.03
for all non-terminals s

Exemple : Grid World

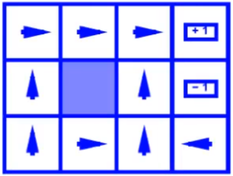
Politiques optimales pour différentes configurations ($\gamma = 1$)



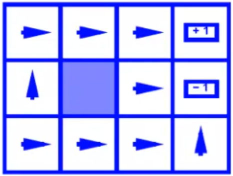
R(s) = -0.01



R(s) = -0.03



R(s) = -0.4



R(s) = -2.0

Evaluation d'une politique

On définit la fonction de valeur V (*value function*) comme la fonction qui, étant donnée une politique π, nous renseigne sur l'espérance de la récompense si l'on suit cette politique :

$$V^{\pi}(s_t) = E_{\pi}[R_t | s_t = s] = E_{\pi}[\sum_{i=0}^{i=+\infty} \gamma^i r_{t+i} | s_t = s]$$

Equation de Bellman

L'équation de Bellman définit le problème de l'apprentissage par renforcement.

$$
\begin{aligned}
V^\pi(s) &= E[R_t | s_t = s] \\
&= E[\sum_{i=0}^{i=+\infty} \gamma^i r_{t+i} | s_t = s] \\
&= E[r_t + \sum_{i=1}^{i=+\infty} \gamma^i r_{t+i} | s_t = s] \\
&= E[r_t + \gamma \sum_{i=1}^{i=+\infty} \gamma^{i-1} r_{t+i} | s_t = s] \\
&= E[r_t] + E[\gamma \sum_{i=1}^{i=+\infty} \gamma^{i-1} r_{t+i} | s_t = s]
\end{aligned}
$$

or :

$$E[r_t] = \sum_{a \in \mathcal{A}(s)} P(a|s) \sum_{s'} P(s'|a,s) \mathcal{R}(s,a,s')$$
$$= \sum_{a \in \mathcal{A}(s)} \pi(a,s) \sum_{s'} \mathcal{P}(s,a,s') \mathcal{R}(s,a,s')$$

et :

$$E[\gamma \sum_{i=1}^{i=+\infty} \gamma^{i-1} r_{t+i} | s_t = s] = \sum_{a \in \mathcal{A}(s)} \pi(a,s) \sum_{s'} \mathcal{P}(s,a,s') \gamma V^{\pi}(s')$$

On en déduit :

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}(s)} \pi(a,s) \sum_{s'} \mathcal{P}(s,a,s') \left(\mathcal{R}(s,a,s') + \gamma V^{\pi}(s') \right)$$

C'est un système de $|S|$ équations linéaires à —S— inconnues.

Evaluation d'une politique

Afin d'évaluer une politique, on construit une suite récurrente V_i^π qui converge vers V^π de la manière suivante :

$$V_{i+1}^\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a,s) \sum_{s'} \mathcal{P}(s,a,s') \left(\mathcal{R}(s,a,s') + \gamma V_i^\pi(s')\right)$$

Algorithme 1 L'algorithme d'évaluation de la politique.

Nécessite: un PDM : $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$
Nécessite: une politique π
Nécessite: un seuil de précision ϵ
  initialiser V_0^π aléatoirement
  $i \leftarrow 0$
  répéter
    pour tout état $s \in \mathcal{S}$ **faire**
      $V_{i+1}^\pi(s) \leftarrow \sum_{a \in \mathcal{A}(s)} \pi(s,a) \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s')[\mathcal{R}(s,a,s') + \gamma V_i^\pi(s')]$
    fin pour
    $i \leftarrow i + 1$
  jusque $||V_i^\pi - V_{i-1}^\pi|| \leq \epsilon$

Lorsque $0 < \gamma < 1$, alors l'algo converge vers un point fixe (Preuve)

Policy Iteration

Une politique π est **meilleure** qu'une politique π' :

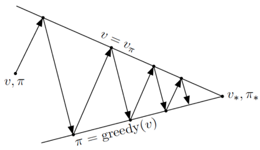
$$\pi \geq \pi' \text{ ssi } \forall s : V^{\pi}(s) \geq V^{\pi'}(s)$$

Algorithme 2 L'algorithme d'itération de la politique.

Nécessite: un PDM : $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$
Nécessite: un seuil de précision ϵ
  initialiser π_0 aléatoirement
  $k \leftarrow 0$
  répéter
    initialiser V_0^{π} aléatoirement
    $i \leftarrow 0$
    répéter
      pour tout état $s \in \mathcal{S}$ **faire**
        $V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{P}(s, \pi_k(s), s')[\mathcal{R}(s, \pi(s), s') + \gamma V_i^{\pi_k}(s)]$
      fin pour
      $i \leftarrow i + 1$
    jusque $||V_i^{\pi_k} - V_{i-1}^{\pi_k}|| \leq \epsilon$
    pour tout état $s \in \mathcal{S}$ **faire**
      $\pi_{k+1}(s, a) \leftarrow \arg\max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} \mathcal{P}(s, a, s')[\mathcal{R}(s, a, s') + \gamma V^{\pi_k}(s')]$
    fin pour
    $k \leftarrow k + 1$
  jusque $\pi_k = \pi_{k-1}$

L'algorithme Policy iteration :

- Converge de manière certaine vers la politique stationnaire optimale en un temps fini (preuve)



- Mais à chaque itération, évaluation complète de la politique...
- ... qui peut s'avérer très coûteuse pour de grands graphes

Selon l'équation d'optimalité de Bellman on observe (preuve) :

$$\begin{aligned} V^*(s) &= max_\pi V^\pi(s) \\ &= max_a \sum_{s'} P(s,a,s') \left(\mathcal{R}(s,a,s') + \gamma V^*(s')\right) \end{aligned}$$

$\Rightarrow$ Algorithme Value Iteration

Value iteration

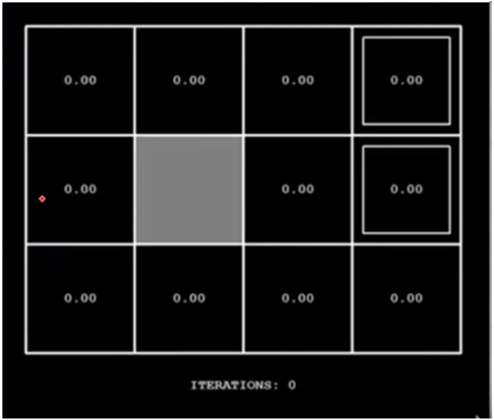
Algorithme 3 L'algorithme d'itération de la valeur.

Nécessite: un PDM : $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$
Nécessite: un seuil de précision ϵ
  initialiser V_0 aléatoirement
  $i \leftarrow 0$
  répéter
    pour tout état $s \in \mathcal{S}$ **faire**
      $V_{i+1}(s) \leftarrow \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} \mathcal{P}(s, a, s')[\mathcal{R}(s, a, s') + \gamma V_i(s')]$
    fin pour
    $i \leftarrow i + 1$
  jusque $||V_i - V_{i-1}|| \leq \epsilon$
  pour tout état $s \in \mathcal{S}$ **faire**
    $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} \mathcal{P}(s, a, s')[\mathcal{R}(s, a, s') + \gamma V(s')]$
  fin pour

Convergence asymptotique vers π^* (preuve)

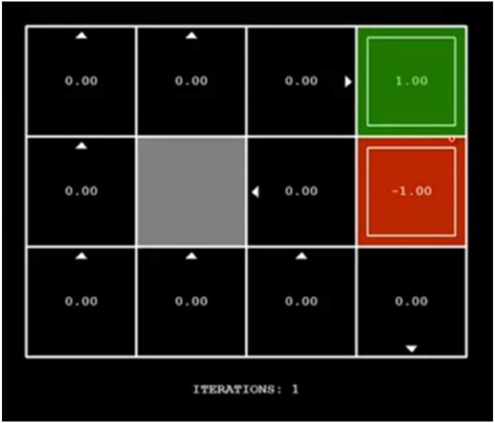
Value iteration on Grid World

Value Iteration Algorithm ($\gamma = 0.99$)



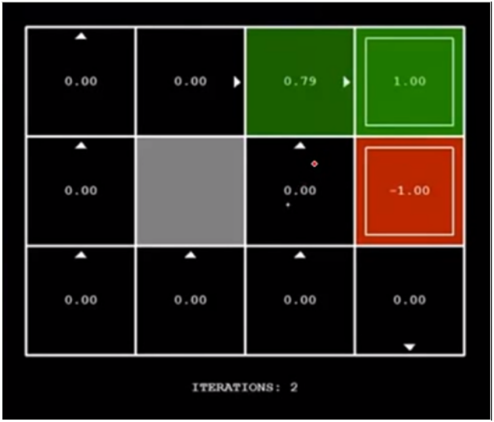
Value iteration on Grid World

Value Iteration Algorithm ($\gamma = 0.99$)



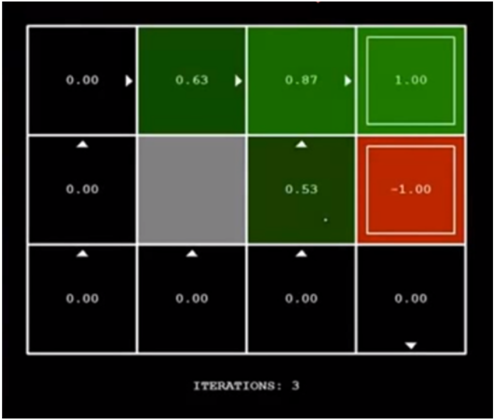
Value iteration on Grid World

Value Iteration Algorithm ($\gamma = 0.99$)



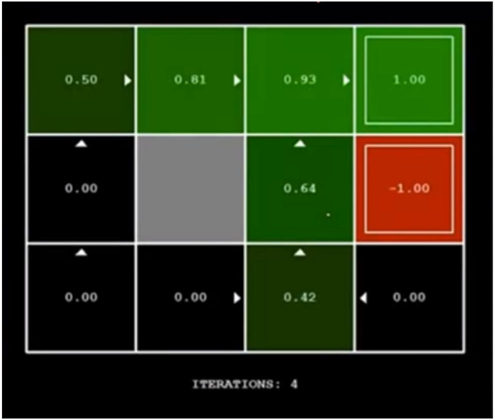
Value iteration on Grid World

Value Iteration Algorithm ($\gamma = 0.99$)



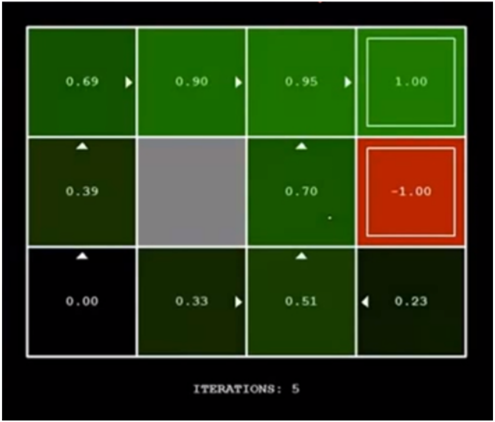
Value iteration on Grid World

Value Iteration Algorithm ($\gamma = 0.99$)



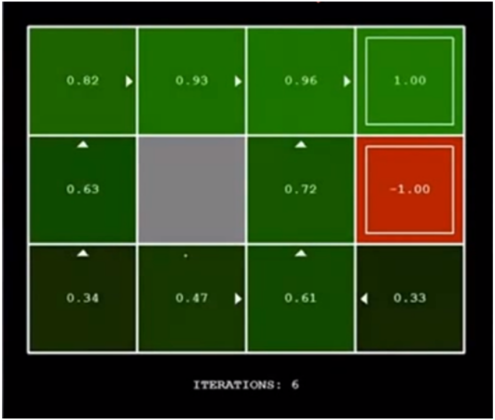
Value iteration on Grid World

Value Iteration Algorithm ($\gamma = 0.99$)

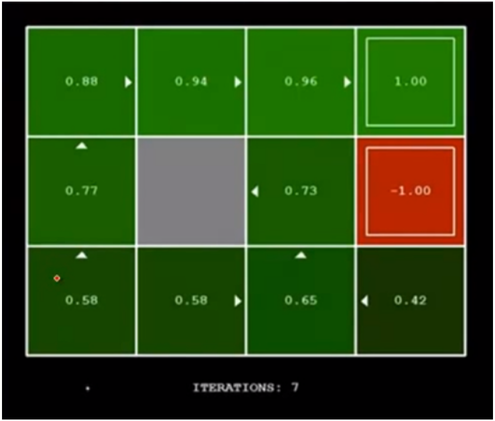


Value iteration on Grid World

Value Iteration Algorithm ($\gamma = 0.99$)

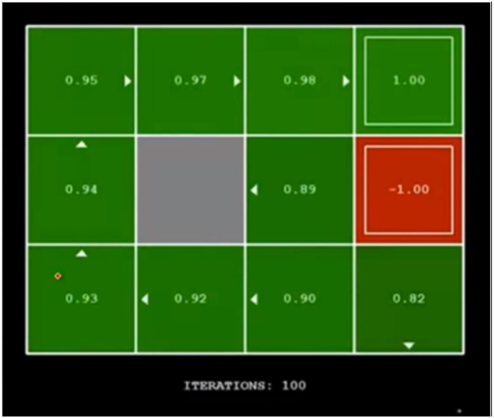


Value Iteration Algorithm ($\gamma = 0.99$)



Value iteration on Grid World

Value Iteration Algorithm ($\gamma = 0.99$)



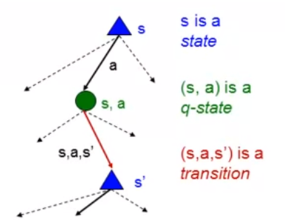
Q-values

La fonction de valeur V (*value function*) est définie comme la fonction qui, étant donnée une politique π, nous renseigne sur l'espérance de la récompense si l'on suit cette politique :

$$V^{\pi}(s_t) = E_{\pi}[R_t|s_t = s] = E_{\pi}[\sum_{i=0}^{i=+\infty} \gamma^i r_{t+i}|s_t = s]$$

De la même manière, on peut définir la Q-fonction comme la fonction qui permet de connaitre la qualité de faire une certaine action dans un certain état :

$$Q^{\pi}(s,a) = E_{\pi}[R_t|s_t = s, a_t = a]$$



Il existe une relation entre Q et V :

$$Q^{\pi}(s,a) = \sum_{s'} P(s_{t+1} = s|s,a)\left(R(s,a,s') + \gamma V^{\pi}(s')\right)$$

Q-value Iteration

- – Plus de valeurs à stocker
- \+ Extraction de la politique facilitée
- \+ Permet la construction de méthodes model-free lorsque l'on ne connaît pas le MDP

Sources

- Ludovic Denoyer (cours à l'ENSIIE)
- Pieter Abbeel (UC Berkeley, Spring 2013)
- Sergey Levine (UC Berkeley, Spring 2017)
- Jonathan Hui : `https://medium.com/@jonathan_hui/rl-value-learning-24f52b49c36d`