Apprentissage par renforcement

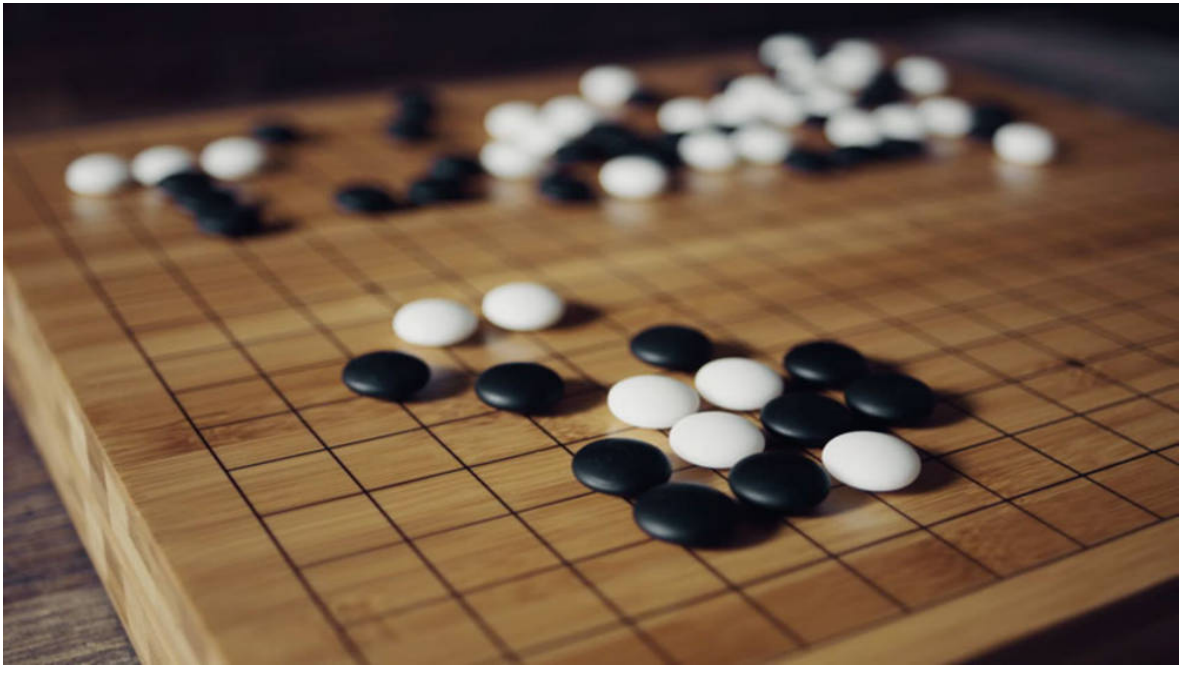
Introduction

Le jeu de Go

- Facteur de branchement très important ($\sim$200)
- Difficile de définir une fonction d'évaluation d'une position

$\Rightarrow$ Approches « arbre minmax» ne marche pas

$\Rightarrow$ Peut-on apprendre une bonne stratégie ?



Introduction

Comment apprendre à traverser un labyrinthe sans se tromper ?

Ou encore, apprendre une bonne stratégie pour :

- passer l'aspirateur dans une pièce donnée ?
- piloter les ascenseurs d'une tour ?

Introduction

Apprentissage par renforcement

= Apprentissage en environnement (partiellement) inconnu
= Apprentissage par essais et erreurs
= Apprentissage par interaction avec l'environnement (exploration)

Introduction

Apprentissage par renforcement

= Apprentissage en environnement (partiellement) inconnu
= Apprentissage par essais et erreurs
= Apprentissage par interaction avec l'environnement (exploration)

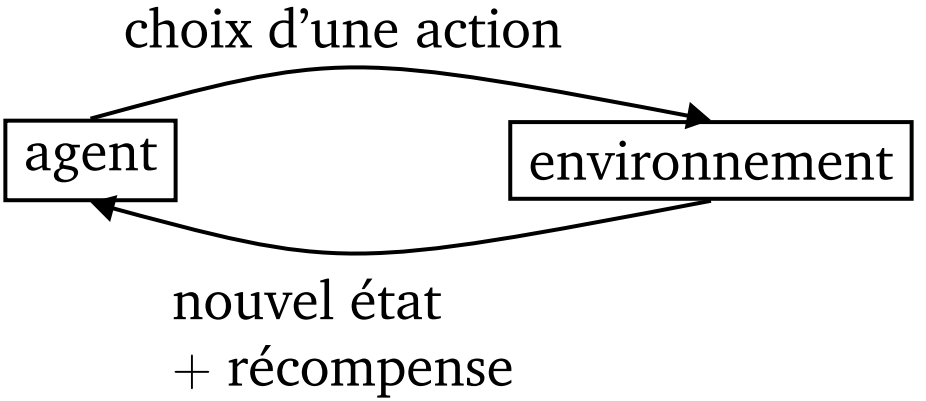
≠ Classification
car on veut ici apprendre de bonnes ***séquences*** d'actions
≠ Planification
car on n'a ici qu'une connaissance ***im***parfaite de l'environnement
car ici l'environnement peut ***changer***
car ici on n'a souvent pas un état initial et un but fixés

Introduction : Un cadre général

Un agent évolue dans un environnement donné.
Il peut effectuer certaines actions dépendant de l'état courant :

- l'état de l'environnement
- et son propre état

ce qui amène dans un nouvel état.
Certaines actions sont liées à des récompenses / coûts immédiats.

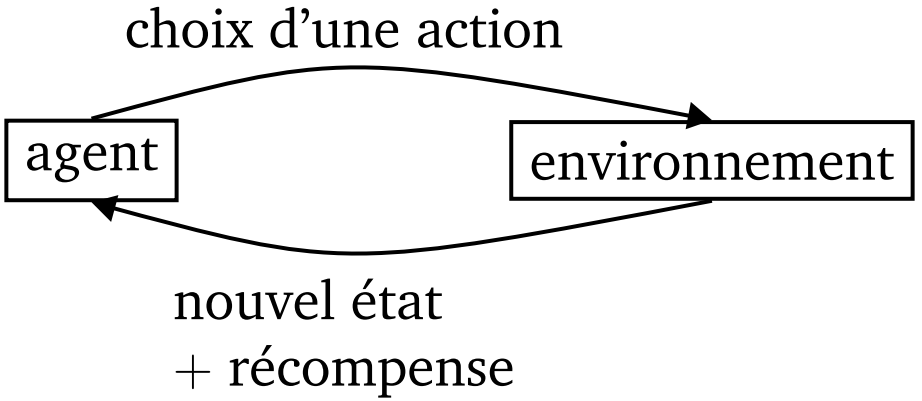


Introduction : Un cadre général

Un agent évolue dans un environnement donné.
Il peut effectuer certaines actions dépendant de l'état courant :

- l'état de l'environnement
- et son propre état

ce qui amène dans un nouvel état.
Certaines actions sont liées à des récompenses / coûts immédiats.



L'agent doit apprendre quelle action choisir dans chaque état afin de suivre une séquence d'action qui lui soit la plus favorable possible.

Introduction : Un cadre général

Apprentissage par renforcement :

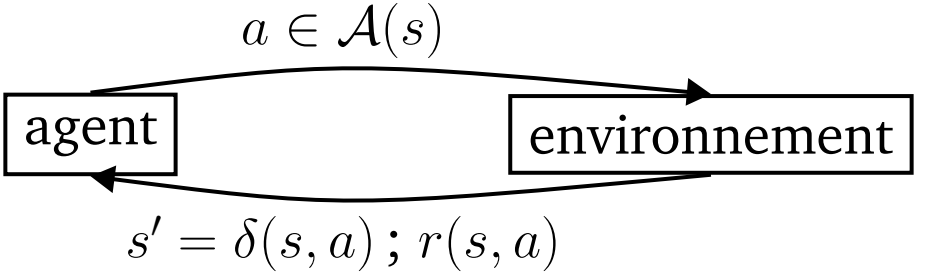
permet d'adapter un ***agent*** à un ***environnement***
en tenant compte de ***récompenses***/punitions
(les ***renforcements***)

Plan du cours

- Processus de décision markoviens (*MDP*) avec actions déterministes
- Programmation dynamique
- Apprentissage par renforcement
- Actions non déterministes

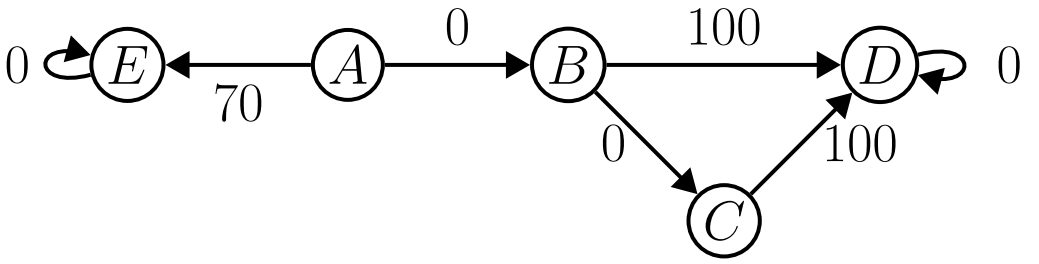
Processus de décision Markovien

$\mathcal{S}$ un ensemble d'états
$\mathcal{A}$ un ensemble d'actions
$\mathcal{A}(s)$ ensemble des actions réalisables dans l'état s
$r(s,a) \in \mathbf{R}$ récompense immédiate
$\delta(s,a) \in \mathcal{S}$ état résultant



Processus de décision Markovien : États, actions, récompenses

Exemple.



Hypothèses « markovienne»

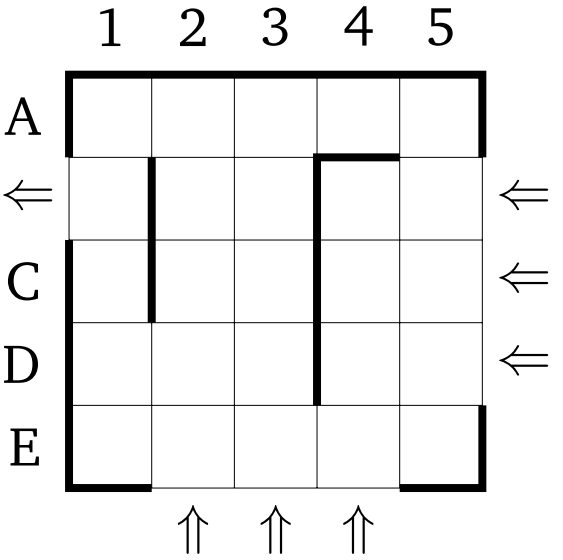
le futur ne dépend pas du passé, sachant le présent

Remarque : on se place pour le moment dans le cas déterministe

Processus de décision Markovien : États, actions, récompenses

Exemple. Un labyrinthe :

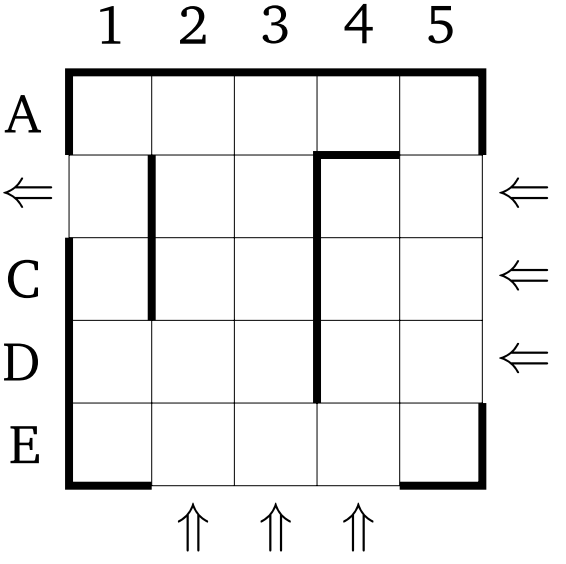
- Plusieurs entrées possibles, 1 sortie
- Déplacements possibles : $\uparrow \downarrow \leftarrow \rightarrow$ (sauf à travers les murs)
- Chaque déplacement coûte 5€
- Gain à la sortie : 100€



Processus de décision Markovien : États, actions, récompenses

Exemple. Un labyrinthe :

- Plusieurs entrées possibles, 1 sortie
- Déplacements possibles : $\uparrow \downarrow \leftarrow \rightarrow$ (sauf à travers les murs)
- Chaque déplacement coûte 5€
- Gain à la sortie : 100€

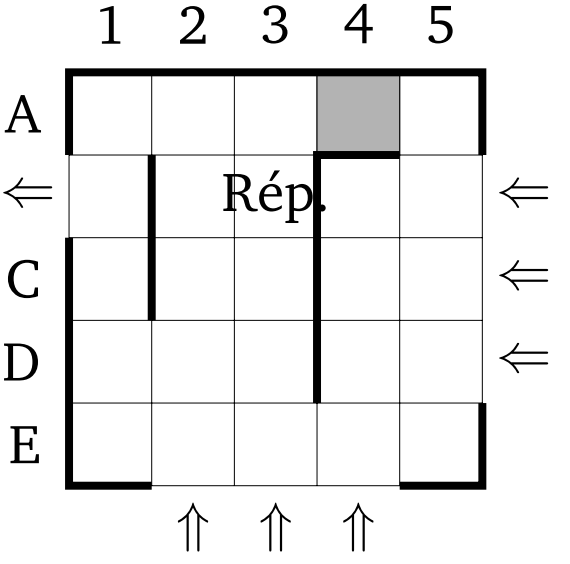


- Combien y a-t-il d'états ? $|\mathcal{S}| =$
- Quelles sont les actions ? $\mathcal{A} =$
- À quelles actions sont associés des coûts / récompenses ?

Processus de décision Markovien : États, actions, récompenses

Exemple. Un labyrinthe :

- Plusieurs entrées possibles, 1 sortie
- Déplacements possibles : $\uparrow \downarrow \leftarrow \rightarrow$ (sauf à travers les murs)
- Chaque déplacement coûte 5€
- Gain à la sortie : 100€ / ***50€ si blessé***
- ***Réparation possible en (C,2)***

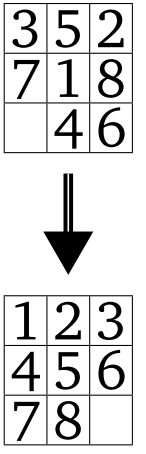


- Combien y a-t-il d'états ? $|\mathcal{S}| =$
- Quelles sont les actions ? $\mathcal{A} =$
- À quelles actions sont associés des coûts / récompenses ?

Processus de décision Markovien : États, actions, récompenses

Exemple. Le taquin :

- une grille 3×3,
  8 tuiles numérotées de 1 à 8
- un état initial quelconque,
  il faut ranger les tuiles par numéros croissants



- Combien y a-t-il d'états ? $|\mathcal{S}| =$
- Quelles sont les actions ? $\mathcal{A} =$
- À quelles actions sont associés des coûts / récompenses ?

Processus de décision Markovien : Politique

Une ***politique*** π formalise le choix des actions :

$$\pi : \mathcal{S} \to \mathcal{A} \qquad \text{telle que } \pi(s) \in \mathcal{A}(s)$$

Processus de décision Markovien : Politique

Une ***politique*** π formalise le choix des actions :

$$\pi : \mathcal{S} \to \mathcal{A} \qquad \text{telle que } \pi(s) \in \mathcal{A}(s)$$

- Professeur Tournesol : «toujours plus à l'Ouest»

Processus de décision Markovien : Politique

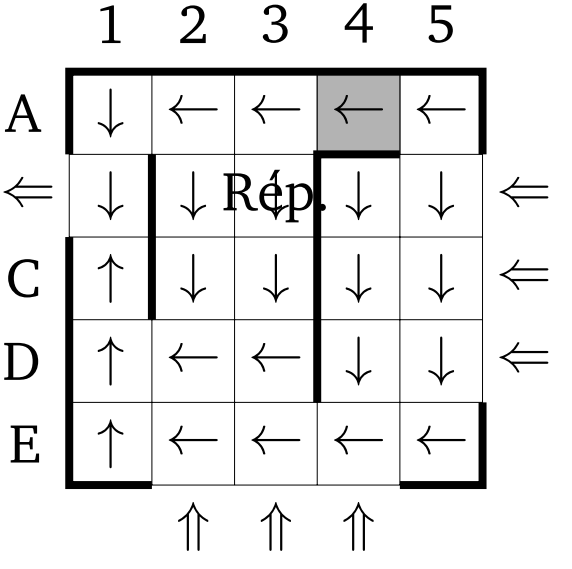
Une ***politique*** π formalise le choix des actions :

$$\pi : \mathcal{S} \to \mathcal{A} \qquad \text{telle que } \pi(s) \in \mathcal{A}(s)$$

- Professeur Tournesol : «toujours plus à l'Ouest»
- Le labyrinthe donné en exemple

  Politique 1 :
  - vers le Nord si possible ; sinon
  - vers l'Est si possible ; sinon
  - vers l'Ouest si possible ; sinon
  - vers le Sud

Politique 2 :



Processus de décision Markovien : Politique

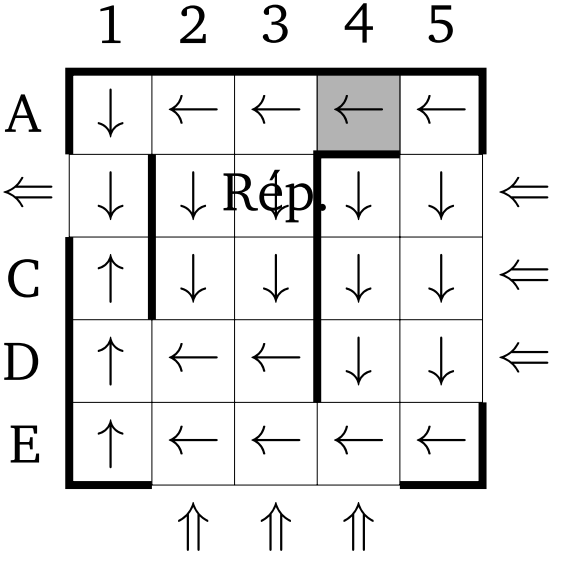
Une ***politique*** π formalise le choix des actions :

$$\pi : \mathcal{S} \to \mathcal{A} \qquad \text{telle que } \pi(s) \in \mathcal{A}(s)$$

- Professeur Tournesol : «toujours plus à l'Ouest»
- Le labyrinthe donné en exemple

  Politique 1 :
  - vers le Nord si possible ; sinon
  - vers l'Est si possible ; sinon
  - vers l'Ouest si possible ; sinon
  - vers le Sud

Politique 2 :



Combien de politiques possibles sur cet exemple ?

Processus de décision Markovien : Politique

- Le taquin 3×3 : combien de politiques possibles ?

Processus de décision Markovien : Politique

- Le taquin 3×3 : combien de politiques possibles ?
- politique $\neq$ stratégie :
  une politique donne une information très «locale» – que jouer dans un état donné
  nous parlerons plus tard de «stratégie d'apprentissage»

Processus de décision Markovien : Critères d'optimalité

Une politique dit quoi faire dans chaque état.
On cherche une politique optimale dans le long terme.

Récompense à long terme pour une séquence d'états/actions $\mathbf{s} = (s_0, a_0, s_1, a_1, \ldots)$

$$R(\mathbf{s}) = \sum_{i=0}^{+\infty} \gamma^i r(s_i, a_i)$$

où $\gamma =$ constante $\in]0, 1]$

Processus de décision Markovien : Critères d'optimalité

Une politique dit quoi faire dans chaque état.
On cherche une politique optimale dans le long terme.

Récompense à long terme pour une séquence d'états/actions $\mathbf{s} = (s_0, a_0, s_1, a_1, \ldots)$

$$R(\mathbf{s}) = \sum_{i=0}^{+\infty} \gamma^i r(s_i, a_i)$$

où $\gamma =$ constante $\in]0, 1]$

Si $\gamma < 1$: favorise les récompenses immédiates ; garantit que la série converge si les récompenses sont bornées.

Processus de décision Markovien : Critères d'optimalité

Une politique dit quoi faire dans chaque état.
On cherche une politique optimale dans le long terme.

Récompense à long terme pour une séquence d'états/actions $\mathbf{s} = (s_0, a_0, s_1, a_1, \ldots)$

$$R(\mathbf{s}) = \sum_{i=0}^{+\infty} \gamma^i r(s_i, a_i)$$

où $\gamma =$ constante $\in]0, 1]$

Si $\gamma < 1$: favorise les récompenses immédiates ; garantit que la série converge si les récompenses sont bornées.

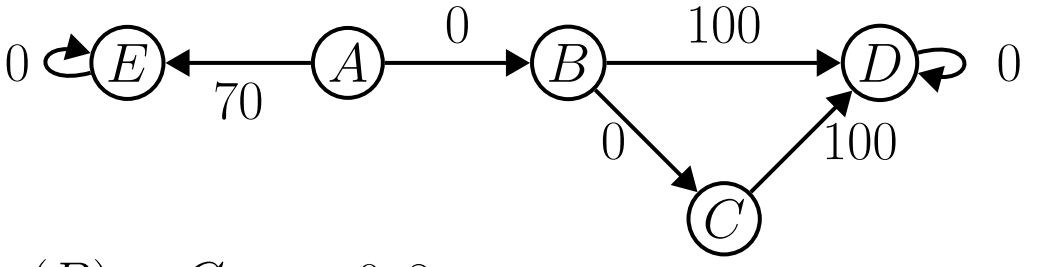
Gain suivant une politique π à partir d'un état s_0 :

$$R(\pi, s_0) = \sum_{i=0}^{+\infty} \gamma^i r(s_i, a_i) \quad \text{avec } a_i = \pi(s_i) \text{ et } s_{i+1} = \delta(s_i, a_i)$$

On l'appelle aussi la ***valeur*** de l'état s_0 pour la politique π.

Processus de décision Markovien : Critères d'optimalité

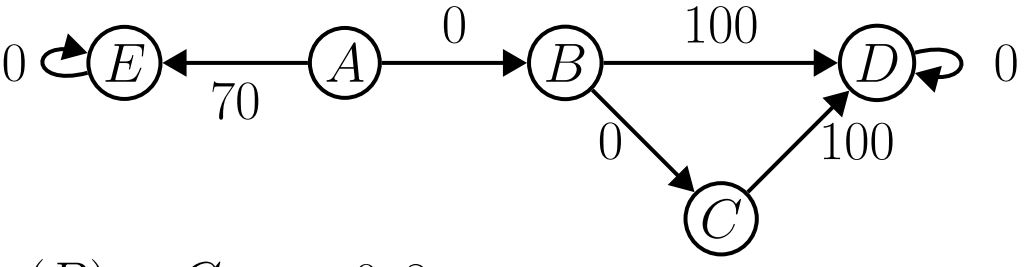
Exemple.



$\pi(A) = B$, $\pi(B) = C$, $\gamma = 0,8$

Processus de décision Markovien : Critères d'optimalité

Exemple.



$\pi(A) = B$, $\pi(B) = C$, $\gamma = 0,8$

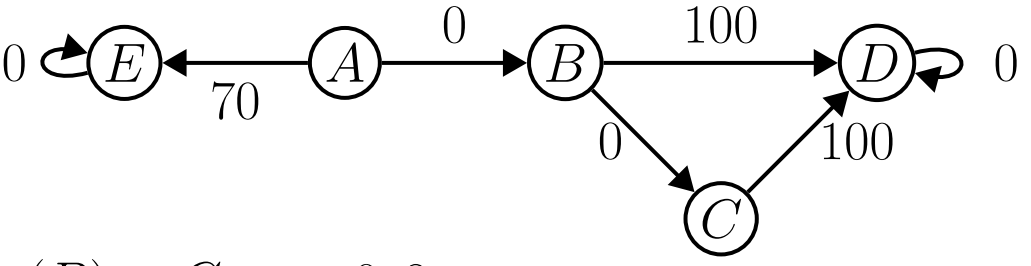
D'autres fonctions de valeur sont possibles :

Horizon fini $R(\mathbf{s}) = \sum_{i=0}^{h} r(s_i, a_i)$

Récompense moyenne $R(\mathbf{s}) = \lim_{h \to \infty} \sum_{i=0}^{h} r(s_i, a_i)$

Processus de décision Markovien : Critères d'optimalité

Exemple.



$\pi(A) = B$, $\pi(B) = C$, $\gamma = 0,8$

D'autres fonctions de valeur sont possibles :

Horizon fini $R(\mathbf{s}) = \sum_{i=0}^{h} r(s_i, a_i)$

Récompense moyenne $R(\mathbf{s}) = \lim_{h \to \infty} \sum_{i=0}^{h} r(s_i, a_i)$

Dans la suite, on ne considère que la récompense à long terme avec horizon infini et facteur de décroissance γ

Processus de décision Markovien : Critères d'optimalité

Certaines politiques sont meilleures que d'autres :

$$\pi \succeq \pi' \text{ si } R(\pi, s_0) \geq R(\pi', s_0) \text{ pour tout } s \in \mathcal{S}$$

La relation $\succeq$ est réflexive et transitive.
Relation stricte induite : $\pi \succ \pi'$ si $\pi \succeq \pi'$ et $\pi' \not\succeq \pi$.

Processus de décision Markovien : Critères d'optimalité

Certaines politiques sont meilleures que d'autres :

$$\pi \succeq \pi' \text{ si } R(\pi, s_0) \geq R(\pi', s_0) \text{ pour tout } s \in \mathcal{S}$$

La relation $\succeq$ est réflexive et transitive.
Relation stricte induite : $\pi \succ \pi'$ si $\pi \succeq \pi'$ et $\pi' \not\succeq \pi$.
Politiques optimales : π telle qu'il n'existe pas de $\pi' \succ \pi$

Processus de décision Markovien : Critères d'optimalité

Certaines politiques sont meilleures que d'autres :

$$\pi \succeq \pi' \text{ si } R(\pi, s_0) \geq R(\pi', s_0) \text{ pour tout } s \in \mathcal{S}$$

La relation $\succeq$ est réflexive et transitive.
Relation stricte induite : $\pi \succ \pi'$ si $\pi \succeq \pi'$ et $\pi' \not\succeq \pi$.

Politiques optimales : π telle qu'il n'existe pas de $\pi' \succ \pi$

Questions auxquelles on va répondre dans la suite :

- Existe-t-il une politique optimale ?
- Comment calculer / apprendre une politique optimale ?

Processus de décision Markovien : Équations de Bellman

Propriété. Pour toute politique π, pour tout état s :

$$R(\pi, s) = r(s, \pi(s)) + \gamma R(\pi, \delta(s, \pi(s))$$

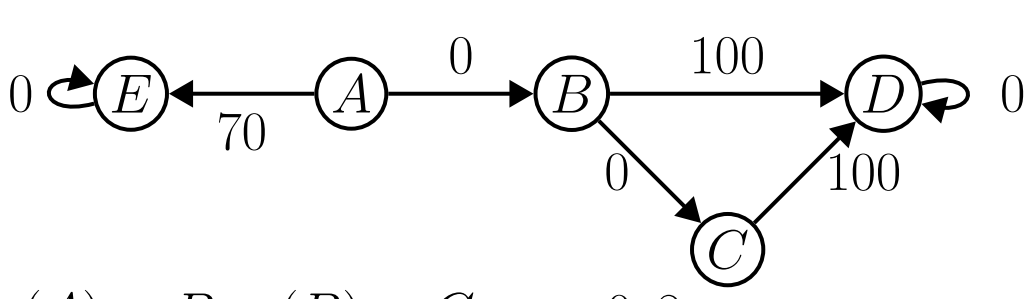
Processus de décision Markovien : Équations de Bellman

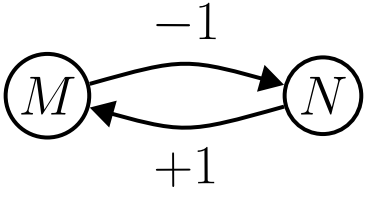
Propriété. Pour toute politique π, pour tout état s :

$$R(\pi, s) = r(s, \pi(s)) + \gamma R(\pi, \delta(s, \pi(s))$$

Cette propriété permet de calculer R pour une politique π donnée, si on a une connaissance parfaite de r et δ.

Résolution d'un système de $|\mathcal{S}|$ équations linéaires
$R(\pi, s) = r(s, \pi(s)) + \gamma R(\pi, \delta(s, \pi(s))$ pour tout $s \in \mathcal{S}$
Exemples :





$\pi(A) = B$, $\pi(B) = C$, $\gamma = 0,8$

Processus de décision Markovien : Équations de Bellman

Un algorithme itératif avec un tableau intermédiaire temp

1. $R(\pi, s) \leftarrow 0$ pour tout $s \in \mathcal{S}$;
2. Tant que pas fini faire :
   (a) pour tout $s \in \mathcal{S}$, si $a = \pi(s)$, $s' = \delta(s, a)$:
   $\mathsf{temp}(s) \leftarrow r(s, \pi(s)) + \gamma R(\pi, \delta(s, \pi(s)))$
   (b) $R(\pi, s) \leftarrow \mathsf{temp}(s)$ pour tout $s \in \mathcal{S}$

Processus de décision Markovien : Équations de Bellman

Propriété. Pour une politique π donnée, s'il existe $s \in \mathcal{S}$ et $a \in \mathcal{A}(s)$ tels que

$$R(\pi, s) < r(s, a) + \gamma R(\pi, s') \text{ (où } s' = \delta(s, a) \text{)}$$

soit π' égale à π sauf en s : on pose $\pi'(s) = a$.
Alors $\pi' \succ \pi$.

La «fameuse» fonction Q :

$$Q(\pi, s, a) = r(s, a) + \gamma R(\pi, \delta(s, a))$$

On a alors $R(\pi, s) = Q(\pi, s, \pi(s))$.

Propriété. Pour une politique π donnée, Q vérifie :

$$Q(\pi, s, a) = r(s, a) + \gamma Q(\pi, s', \pi(s') \text{ (où } s' = \delta(s, a) \text{)}$$

Processus de décision Markovien : Équations de Bellman

La fonction Q permet de détailler, pour une politique donnée, ce qui se passe si on dévie de la politique courante pour 1 transition seulement.

Processus de décision Markovien : Équations de Bellman

Système d'équations vérifié par les politiques optimales :
si π^* est optimale, alors pour tous $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$$Q(\pi^*, s, a) = r(s, a) + \underset{a' \in \mathcal{A}(s')}{\operatorname{argmax}} Q(\pi^*, s', a') \text{ (où } s' = \delta(s, a) \text{)}$$

Mais ça n'est pas un système d'équations linéaires ordinaire (argmax)
$\Rightarrow$ pas aussi facile à résoudre...

Programmation dynamique

Ce terme regroupe des méthodes qui permettent de «résoudre» un MDP lorsqu'on ***connaît le modèle***.

Amélioration itérative de politique : c'est un algorithme itératif simple pour améliorer petit à petit la politique.

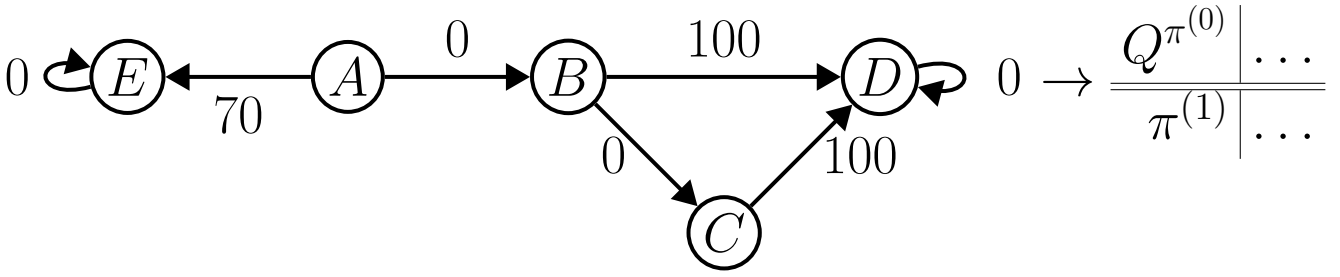
1. définir une première politique π ;
2. tant que «pas fini» faire :
   (a) pour tous $s \in \mathcal{S}, a \in \mathcal{A}(s)$: calculer $Q(\pi, s, a)$;
   (b) pour tout $s \in \mathcal{S}$:
   $$\pi'(s) \leftarrow a \in \mathcal{A}(s) \text{ qui maximise } Q(\pi, s, a) \text{ ;}$$
   (c) $\pi \leftarrow \pi'$;

Conditions d'arrêt :

- plus de temps
- peu d'écart entre deux itérations successives

Programmation dynamique

Exemple.



Programmation dynamique

Remarque. L'algorithme précédent fait évaluer complètement la politique courante à chaque itération $\Rightarrow$ coûteux !
Mais ça n'est pas nécessaire. En fait, on n'a même pas besoin de définir explicitement une politique courante.

Programmation dynamique

Remarque. L'algorithme précédent fait évaluer complètement la politique courante à chaque itération $\Rightarrow$ coûteux !
Mais ça n'est pas nécessaire. En fait, on n'a même pas besoin de définir explicitement une politique courante.

Amélioration itérative de valeurs :

1. $R(s) \leftarrow 0$ pour tout $s \in \mathcal{S}$;
2. tant que «pas fini» faire :*//(boucle principale)*
   pour tout $s \in \mathcal{S}$ faire :
   (a) pour tout $a \in \mathcal{A}(s)$: $Q(s,a) \leftarrow r(s,a) + \gamma R(\delta(s,a))$
   (b) $R(s) \leftarrow \max_{a \in \mathcal{A}(s)} Q(s,a)$;
3. pour tout $s \in \mathcal{S}$: *//(calcul de la politique finale π)*
   (a) pour tout $a \in \mathcal{A}(s)$: $Q(s,a) \leftarrow r(s,a) + \gamma R(\delta(s,a))$
   (b) $\pi(s) \leftarrow a \in \mathcal{A}(s)$ qui maximise $Q(s,a)$;

Programmation dynamique

Pour voir qui a suivi :

Pourquoi parle-t-on de «programmation dynamique» ? ? ?

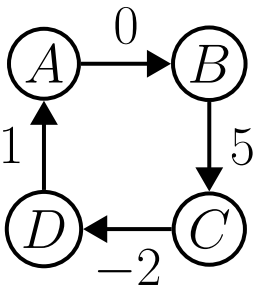
Programmation dynamique

Explication de Bellman, dans son autobiographie :
(Eye of the Hurricane : An Autobiography, 1984)

> *We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word research. . . .Hence, I felt I had to do something to shield Wilson . . .from the fact that I was really doing mathematics. . .What title, what name, could I choose ? . . .planning, is not a good word for various reasons. I decided therefore to use the word "programming". . . .Let's take a word that has an absolutely precise meaning, namely dynamic. . .It also has a very interesting property as an adjective, and that it's impossible to use the word dynamic in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible.* ***Thus, I thought dynamic programming was a good name.*** *It was something not even a Congressman could object to. So I used it as an umbrella for my activities.*

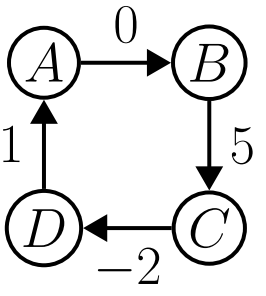
Exercices

Exercice 1. Calculer les valeurs des récompenses à long terme sur le graphe de transition ci-contre, en considérant un facteur $\gamma = 0,9$.



Exercices

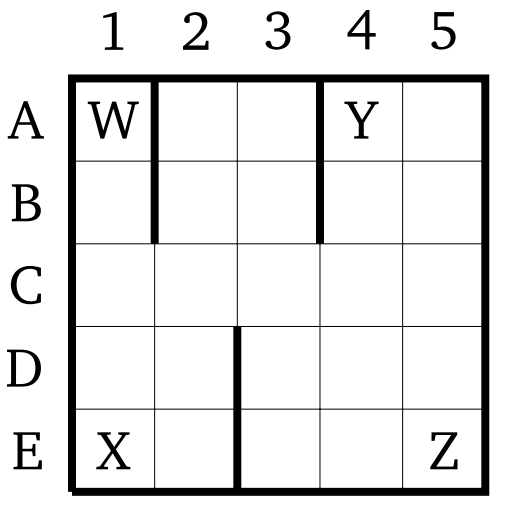
Exercice 1. Calculer les valeurs des récompenses à long terme sur le graphe de transition ci-contre, en considérant un facteur $\gamma = 0,9$.



Exercice 2. Si on augmente toutes les récompenses immédiates d'un MDP d'une constante C, de combien sont augmentées les récompenses à long terme ?

Exercices

Exercice 3. Un chauffeur de taxi d'une ville représentée par la grille 5×5 ci-contre peut prendre ou déposer des clients aux points W, X, Y et Z. Au début d'un épisode, le taxi est sur une case quelconque, et il y a un passage sur l'un des 4 points de charge / décharge, à prendre et déposer sur l'un des autres points de charge / décharge.



Les actions possibles sont « Nord», « Sud», « Est», « Ouest», « Charger», « Déposer». Il y a une récompense de -1 pour chaque déplacement, de -10 pour une tentative de charge ou décharge au mauvais endroit, et de $+20$ quand on dépose le passager à sa destination.

Exercices

Question 1. Combien y a-t-il d'états possibles ?

Question 2. Dessiner le chemin du graphe de transitions allant d'un état où le taxi est en A3 et où un client attend en W jusqu'à l'état où le client est déposé à sa destination en Y.

Question 3. Donner les valeurs des récompenses à long terme de la politique optimale, en supposant un facteur $\gamma = 1$, pour chacune des cases lorsque :

1. il y a un client à prendre en W et à déposer en Y ;
2. il y a un client dans le taxi, à déposer en Y.

Apprentissage par renforcement

Où comment résoudre un MDP qu'on ne connaît pas à l'avance !

Rappels : on utilise l'apprentissage par renforcement quand

- l'environnement est (partiellement) inconnu
- l'environnement peu changer
- on peut interagir avec l'environnement

Il va falloir ***explorer*** le MDP, l'échantillonner.
On ne suppose plus qu'on connaît les fonctions r et δ.

Apprentissage par renforcement

Un algorithme générique d'apprentissage par renforcement

1. initialiser une représentation du MDP
2. répéter
   (a) choisir un état de départ $s \in \mathcal{S}$;
   (b) répéter
      i. choisir une action $a \in \mathcal{A}(s)$;
      ii. exécuter a en observant récompense r + nouvel état s' ;
      iii. mettre à jour la représentation du MDP ;
      iv. $s \leftarrow s'$.

Remarque. On n'utilise plus de fonctions de récompense et de transition, mais des observations.

Les méthodes d'apprentissage par renforcement diffèrent les unes des autres par :

- la représentation du MDP (fonctions R, Q, ...)
- la stratégie de choix d'une action en 2(b)i
- la stratégie de mise à jours de la représentation du MDP en 2(b)iii

Apprentissage par renforcement : Q-learning

Un algorithme générique d'apprentissage par renforcement

1. initialiser une représentation du MDP
2. répéter
   (a) choisir un état de départ $s \in \mathcal{S}$;
   (b) répéter
      i. choisir une action $a \in \mathcal{A}(s)$;
      ii. exécuter a en observant récompense r + nouvel état s' ;
      iii. mettre à jour la représentation du MDP ;
      iv. $s \leftarrow s'$.

Pour Q-learning :

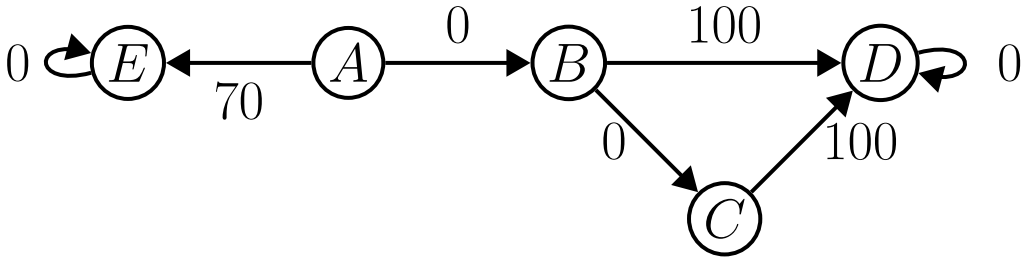
- Représente le MDP avec la fonction Q
- Règle de mise à jour :

$$Q(s,a) \leftarrow r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s',a')$$

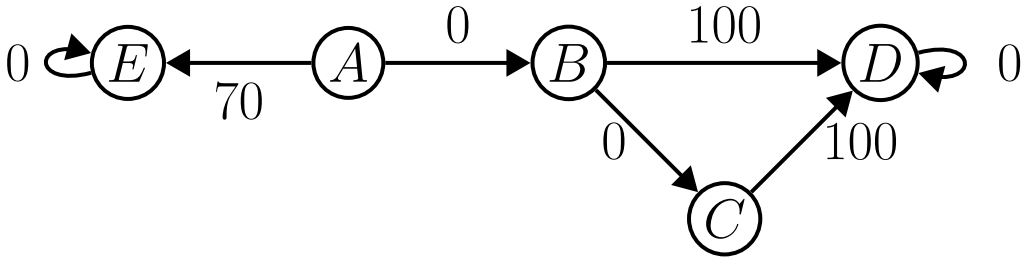
Apprentissage par renforcement : Q-learning

L'algorithme de Q-learning ne précise pas combien de fois on effectue la boucle de mise à jour de Q : en fait, l'agent va alterner des phases d'exploration, durant lesquelles il effectue cette boucle, et des phases où il utilise ce qu'il a appris, même si c'est imparfait, pour réaliser certains buts.

Apprentissage par renforcement : Q-learning



Apprentissage par renforcement : Q-learning



Apprentissage par renforcement : Convergence

Propriété.
- Si les récompenses immédiates sont bornées, et
- si $0 \leq \gamma < 1$, et
- si le choix de l'action à effectuer dans la boucle est tel que toute paire état/action sera évaluée infiniment souvent

alors la fonction Q apprise par l'algorithme converge vers la fonction $Q(\pi^*, \cdot, \cdot)$ d'une politique π^* optimale.

Preuve.....

Apprentissage par renforcement : Convergence

Amélioration possible : lorsque les récompenses apparaissent peu souvent, on peut rétropropager Q pour toutes les paires état/action rencontrées lors d'un épisode.

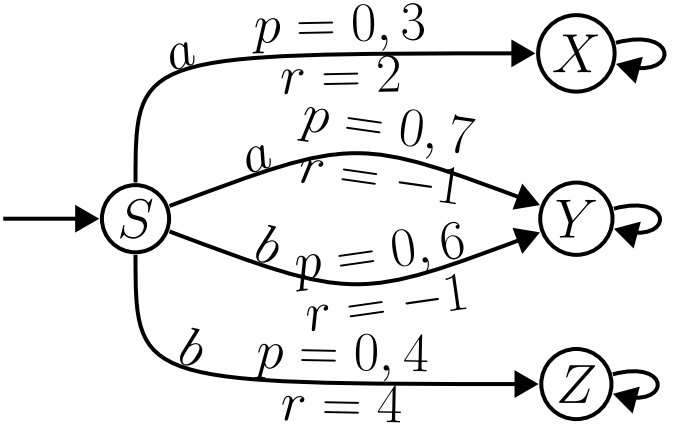
Sur l'exemple introductif, en considérant des épisodes d'apprentissages qui commencent en (A,4) : quel est le nombre minimum d'épisodes nécessaire pour apprendre complètement la fonction Q sans / avec rétropropagation ?

Transitions non déterministes

Dans de nombreuses situations, le résultat d'une action ne peut être prédit avec certitude. Par exemple :

- Jeux à plusieurs joueurs
  $\Rightarrow$ le résultat d'un coup d'un joueur dépend des coups des autres
- Robot en déplacement
  $\Rightarrow$ le résultat d'une accélération dépend de l'adhérence
  ...

- $\delta(s, a, s') \in [0, 1]$:
  probabilité d'aller en s' avec action a dans l'état s
- $r(s, a, n) \in [0, 1]$:
  probabilité d'avoir récompense n avec action a dans l'état s



Transitions non déterministes

Récompense à long terme ***moyenne*** pour une politique donnée π en partant d'un état s_0 :

$$R(\pi, s_0) = E[\sum_{i \geq 0} \gamma^i r(s_i, a_i)],$$

moyenne sur les séquences $(s_0, a_0, s_1, a_1, \ldots)$ possibles.

Transitions non déterministes

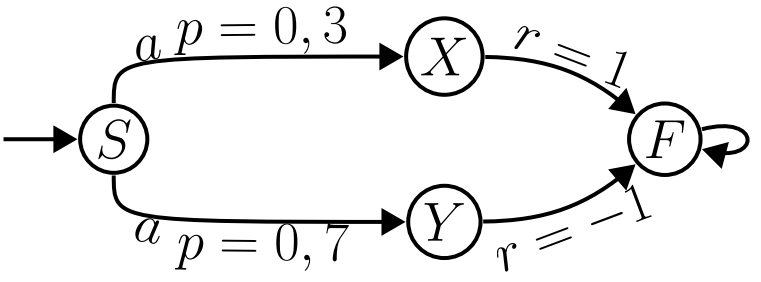
En notant $Q(s,a) = E[r(s,a) + \gamma R(\pi^*, \delta(s,a))]$, on a

$$\begin{aligned} Q(s,a) &= E[r(s,a)] + \gamma E[R(\pi^*, \delta(s,a))] \\ &= E[r(s,a)] + \gamma \sum_{s'} p(s' \mid s,a) R(\pi^*, s') \\ &= E[r(s,a)] + \gamma \sum_{s'} p(s' \mid s,a) \max_{a'} Q(s',a') \end{aligned}$$

Transitions non déterministes

Si on applique directement l'algorithme de Q-learning : risque d'oscillations

Exemple.



Transitions non déterministes : Q-learning

Un algorithme générique d'apprentissage par renforcement

1. initialiser une représentation du MDP
2. répéter
   (a) choisir un état de départ $s \in \mathcal{S}$;
   (b) répéter
      i. choisir une action $a \in \mathcal{A}(s)$;
      ii. exécuter a en observant récompense r + nouvel état s' ;
      iii. mettre à jour la représentation du MDP ;
      iv. $s \leftarrow s'$.

Pour Q-learning avec transitions non déterministes :

- Représente le MDP avec la fonction Q
- Règle de mise à jour :

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s',a'))$$

$\alpha = 1/(1 + \mathsf{nb\_visites}(s,a))$ est le ***taux d'apprentissage***

Transitions non déterministes : SARSA

Un algorithme générique d'apprentissage par renforcement

1. initialiser une représentation du MDP
2. répéter
   (a) choisir un état de départ $s \in \mathcal{S}$;
   (b) répéter
      i. choisir une action $a \in \mathcal{A}(s)$;
      ii. exécuter a en observant récompense r + nouvel état s' ;
      iii. mettre à jour la représentation du MDP ;
      iv. $s \leftarrow s'$.

SARSA : (« *on-policy* »)

- Représente le MDP avec la fonction Q et la politique courante correspondante π
- Règle de mise à jour :

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

Généralisation des renforcements

Dans la pratique : difficile / impossible de mémoriser une table de Q ($|\mathcal{S}| \times E[|\mathcal{A}(s)|] \sim 10^{29}$ pour Othello !)

$\rightarrow$ on apprend une approximation de Q.

Par exemple : pour Othello, un réseau de neurones :

- 64 unités en entrées
- une couche cachée
- 64 unités en sorties : $Q(s, a)$ pour chacun des coups possibles a

$\rightarrow$ chaque fois qu'on doit mettre Q à jour, on applique une méthode de rétropropagation.

Généralisation des renforcements

1. Initialiser Q ;
2. répéter
   (a) choisir $s \in \mathcal{S}$; $\mathcal{E} \leftarrow \emptyset$;
   (b) répéter :
      i. choisir $a \in \mathcal{A}(s)$;
      ii. $r \leftarrow r(s,a)$; $s' \leftarrow \delta(s,a)$
      iii. $\alpha \leftarrow 1/(1 + \mathsf{nb\_visites}(s,a))$;
      iv. $q \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a' \in \mathcal{A}(s)} Q(s',a'))$;
      v. $\mathcal{E} \leftarrow \mathcal{E} \cup \{((s,a),q)\}$;
      vi. $s \leftarrow s'$;
   (c) « ré-apprendre» Q à partir des exemples de $\mathcal{E}$;

Remarque : en 2c, il ne s'agit pas d'apprentissage de classifieur, mais de ***régression***.

Problème : dès lors qu'on a une telle représentation appochée de Q, la méthode risque de ne pas converger.

Apprentissage par renforcement relationnel

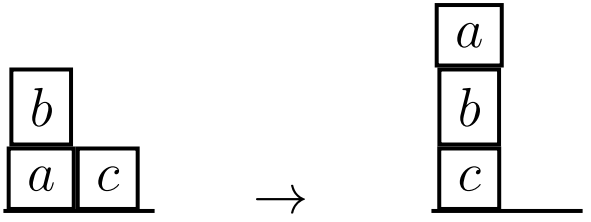
La représentation de Q, et la méthode d'apprentissage, dépendent de la représentation des états et des actions.

Lorsque les états sont représentés à l'aide de relations, et non plus de simples paires (attribut,valeur), on parle d'apprentissage par renforcement relationnel.

Apprentissage par renforcement relationnel

Exemple : le monde des cubes

État initial : Récompense :



$\mathsf{sur\_table}(a)$
$\mathsf{sur\_table}(c)$
$\mathsf{sur}(b, a)$
$\mathsf{libre}(b)$
$\mathsf{libre}(c)$

$\mathsf{sur}(a, b)$
$\mathsf{sur\_table}(c)$
$\mathsf{sur}(b, c)$
$\mathsf{libre}(a)$

Actions possibles : déplacement des cubes – si rien posé dessus ; par exemple, dans l'état initial, $\mathsf{deplacer}(b, c)$.

Apprentissage par renforcement relationnel

Exemple enregistré :

$\mathsf{but} = \mathsf{sur}(a, b)$,

$\mathsf{etat} = \{\mathsf{sur\_table}(a), \mathsf{sur\_table}(c), \mathsf{sur}(b, a), \mathsf{libre}(b), \mathsf{libre}(c)\}$,

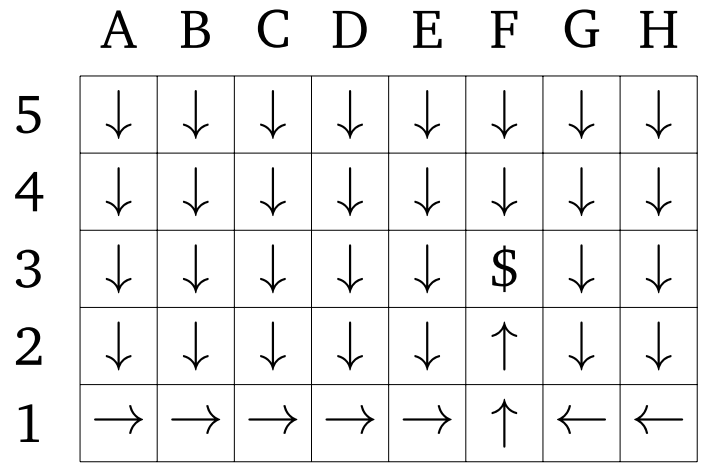
$\mathsf{action} = \mathsf{deplacer}(b, c)$,

$\mathsf{q} = 0.81$

...

Exercices

Exercice 4. Un agent se déplace dans le monde dessiné ci-contre : il y a en général quatre actions possibles (gauche, droite, haut, bas), sauf sur les bords. Chaque action à une récompense de -1 sauf les actions qui font arriver sur la case F3, marquée $, qui provoquent une récompense de 100.



On considère d'abord une politique π_0 indiquée sur dessin : il se dirige toujours vers le bas, sauf quand il est sur la ligne 1 ou quand il est sur la case F2

Question 1. Avec cette politique π_0, quelles sont, en fonction de γ, les récompenses à long terme pour les cases A5 et H1 ?

Question 2. En supposant qu'on peut calculer $Q(\pi_0, s, a)$ pour toutes les paires (état,action), comment peut-t-on améliorer la politique π_0 ?

On considère qu'il y a maintenant un vent fort du bas vers le haut sur les colonnes D, E et F : lorsque l'agent effectue une action depuis

une case de ces colonnes, il atterrit une case plus haut que la case prévue, dans la limite des cases de la grille. Par exemple, si le robot part à gauche depuis la case E3, il atterrit en fait en F4 ; mais s'il part à gauche depuis E5, il atterrit en F5 (car il ne peut sortir de la grille).

Question 3. Quelle sont maintenant les récompenses à long terme, toujours avec la politique π_0, des cases H5 et A5 ?

Question 4. En supposant un coefficient $\gamma = 1$, dessinez une politique optimale.