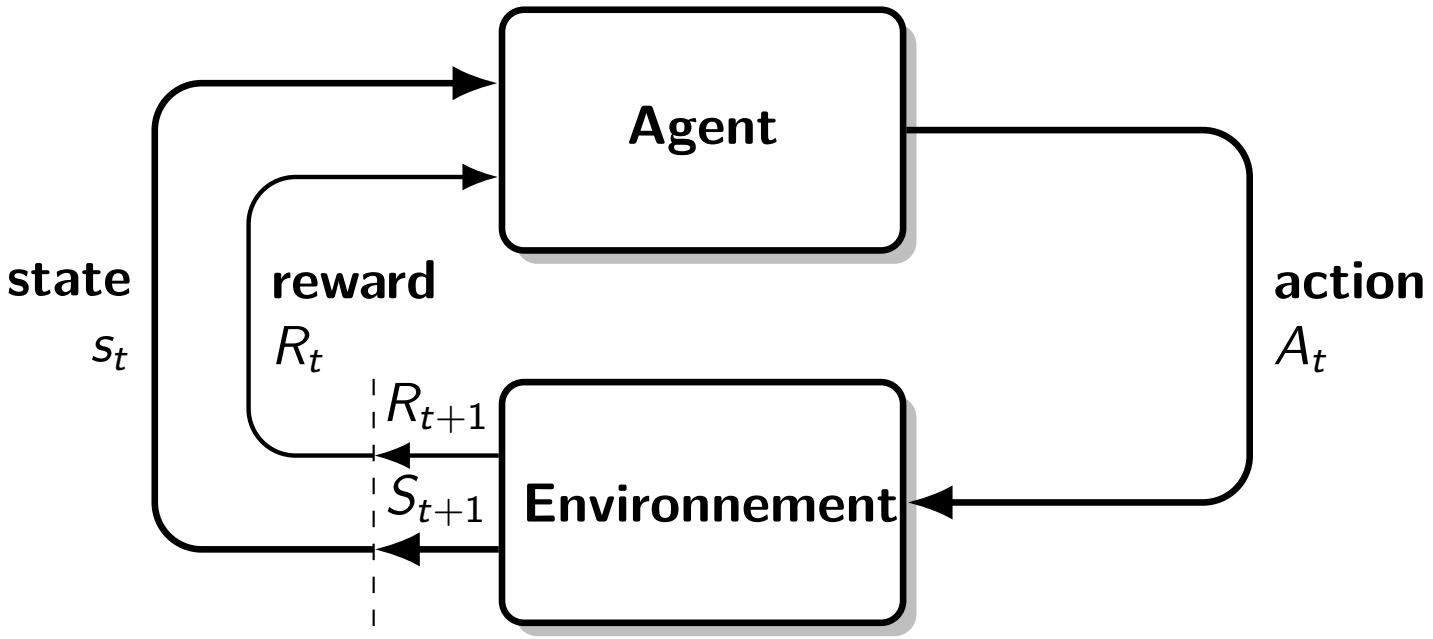
IN104 - Apprentissage par renforcement

Mathieu Dajon - Monnerat Léry

2024

- Un ensemble fini d'états S, un ensemble fini d'actions A, un ensemble de récompenses possibles.
- à chaque étape, l'agent à partir de son état courant S_t, fait une action A_t.
- il reçoit de l'environnement son nouvel état S_{t+1} et une récompense R_{t+1}.
- évolution : suite $S_0, A_0, R_0, S_1, A_1, R_1, \ldots$



But de l'apprentissage par renforcement

Calculer une politique "optimale"

$$\pi : S \rightarrow A$$

qui pour chaque état $s \in S$ donne une action $a \in A$ à effectuer par l'agent dans l'environnement.

Q-Learning

Calcul itératif de $Q : S \times A \to \mathbb{R}$, estimation de la récompense.

procedure APPRENTISSAGE($N, \gamma, \epsilon, \alpha$)
  Initialisation Q
  for all $n \in \{1, \ldots, N\}$ **do** ▷ N itérations
    Initialisation s
    repeat
      Choisir l'action a (stratégie de choix fixée)
      Faire l'action a, nouvel état s', et récompense r
      $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_a Q(s',a))$
      $s \leftarrow s'$
    until s est final
  end for
  return Q
end procedure

Convergence vers la récompense optimale : $Q_N \to Q^*$

Choix de l'action à effectuer, et apprentissage

ϵ-greedy (compromis aléatoire/glouton)

$$\begin{cases} \text{probabilié } \epsilon & a = \textit{random uniforme} & \textit{exploration} \\ \text{probalibilté } 1-\epsilon & a = \max_a Q(s,a) & \textit{exploitation} \end{cases}$$

Boltzmann ($\approx$ aléatoire pour T grand, $\approx$ glouton pour T petit)

$$p_s(a) = \frac{e^{Q(s,a)/T}}{\sum_b e^{Q(s,b)/T}}$$

Mise à jour de Q lors du changement d'état $s \rightarrow s'$:

$$Q(s,a) = (1-\alpha).Q(s,a) + \alpha(r + \gamma max_a Q(s',a))$$

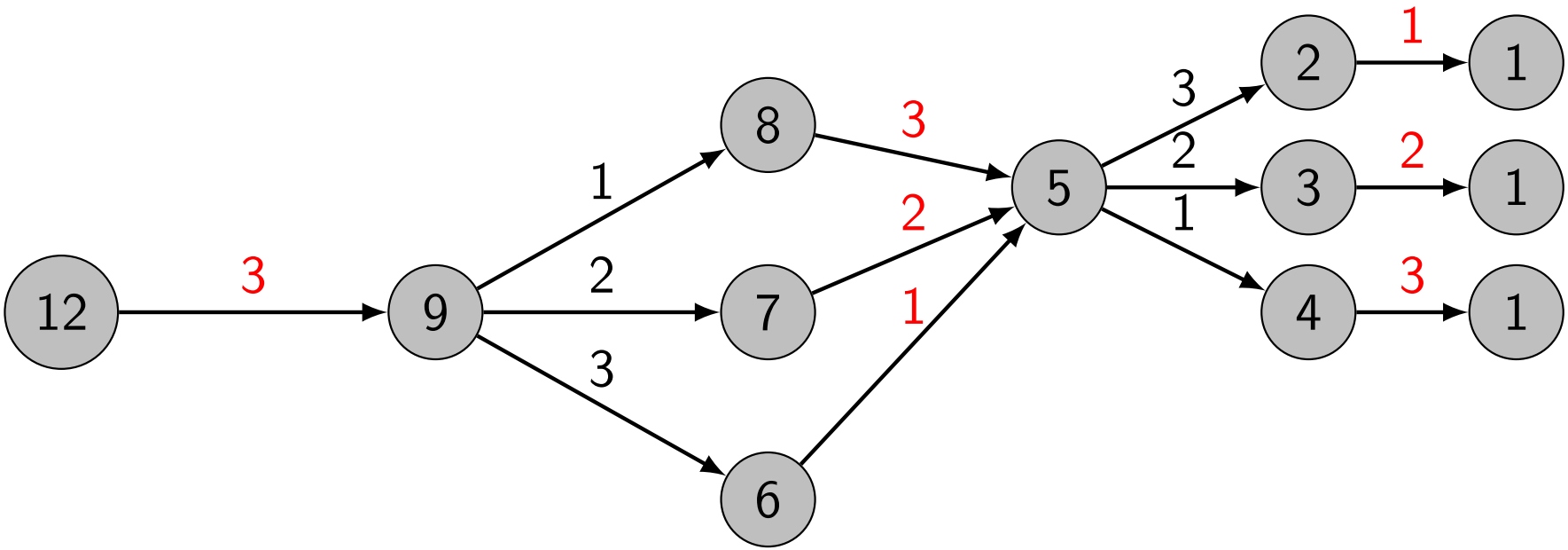
On apprend de la meilleure action possible à partir du nouvel état.

Un exemple très simple

Il s'agit d'un jeu très simple, à deux joueurs, tour par tour.

- 12 batôns posés sur une table.
- Chaque joueur, alternativement, peut en prendre 1,2 ou 3.
- Celui qui ramasse le dernier a perdu.

Il y a une stratégie gagnante pour le joueur qui débute :



Résultats

On entraîne (10000 parties, ϵ-greedy) le joueur 1 (qui débute) face à un joueur aléatoire. (récompense 0, et 1 en cas de victoire).

```
epsilon = 0.3, alpha = 0.1 gamma = 0.9
etat : actions (1,2,3)
01 :  0.00000   0.000000   0.000000
02 : >1.00000<  0.000000   0.000000
03 :  0.60558  >1.000000<  0.000000
04 :  0.66493   0.764931  >1.000000<
05 :  0.44415   0.406543   0.812027
06 : >0.90000<  0.560154   0.687911
07 :  0.79675  >0.899999<  0.636438
08 :  0.76245   0.833438  >0.899999<
09 :  0.70999   0.726325   0.804411
10 :  0.81000   0.702088   0.753197
11 :  0.00000   0.000000   0.000000
12 :  0.74281   0.770997  >0.809999<
```

Le joueur 1 "découvre" la stratégie gagnante. (améliorer la convergence en entraînant les 2 joueurs)

Le problème du labyrinthe

```
++++++++++++++++++++
+                  +
+ + + ++++ + +++++ +
+ + +++ ++ + ++ ++ +
+ +   +    + ++    +
+ +++ ++++++ +++++ +
+  s+      ++ +g+   +
+ +++++++ ++ + + + +
+ +       ++ +   + +
+ ++++++++++ +++++ +
+                  +
++++++++++++++++++++
```

Implantation et résultats

- États : position sur la grille.
- Actions : déplacement Nord, Sud, Est et Ouest.
- Récompenses : si la case où l'on se déplace (état) est
  - libre $\rightarrow$ 0,
  - un mur $\rightarrow$ -10
  - la sortie $\rightarrow$ 1000.

Apprentissage avec Q-Learning

Sarsa : différence avec Q-Learning

Supposons dans les deux cas qu'on utilise une stratégie ϵ-greedy pour le choix de l'action : On peut résumer la différence entre Sarsa et Q-learning par ce tableau :

	Sarsa	Q-learning
choix de l'action	ϵ-greedy	ϵ-greedy
mise à jour de Q	ϵ-greedy	greedy ($epsilon = 0$)

Q-Learning met à jour Q (apprentissage) avec la meilleure action possible du nouvel état (glouton). Dans Sarsa,

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

Autrement dit, Sarsa apprend de l'action choisie pour le nouvel état, alors que Q-learning apprend de l'action optimale.

Comparasion avec l'environnement du labyrinthe

Améliorations

- Adaptation des paramètres au cours de l'apprentissage :
  - Pour ϵ-greedy, faire décroitre ϵ au cours de l'apprentissage : exploration au début, stratégie groutonne à la fin.
  - Pour Boltzmann, faire décroitre T pour les mêmes raisons.
- Problème de l'arrêt de l'algorithme