Notes de cours
OPT5 : Apprentissage par renforcement

Adrien Pavao

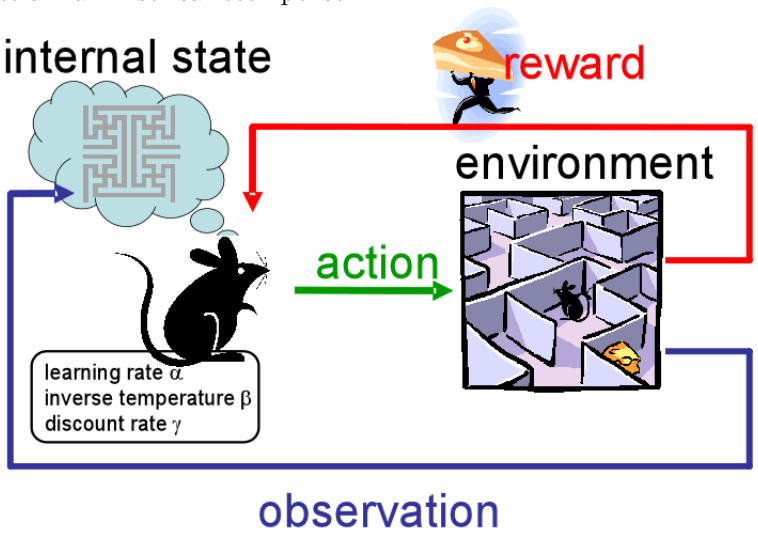
Novembre 2017

1 Introduction

L'apprentissage par renforcement consiste à apprendre à partir d'expériences, plutôt qu'à partir de données pré-établies, de façon à optimiser une récompense quantitative au cours du temps.

Un agent est plongé au sein d'un environnement. Il doit choisir ses actions (prise de décision) en fonction de son état courant. En retour, l'agent reçoit une récompense de l'environnement, qui peut être positive ou négative.

L'agent cherche à améliorer sa politique de décision au fur et à mesure de sorte à maximiser sa récompense.



Dans l'illustration ci-dessus, on voit :

- **L'environnement :** Le labyrinthe.
- **L'agent :** La souris. C'est l'observation qu'elle fait de l'environnement que l'on nomme son état courant. On nomme tous les états possibles l'**espace des états**.

- **Les actions :** Les actions possibles de la souris en réaction à l'état. Avancer, tourner à gauche, revenir en arrière, etc. Toutes les actions réalisables par l'agent constituent l'**espace des actions**.
- **La récompense :** Le fromage. La souris souhaite tirer profit de sa perception et de sa stratégie de décision afin d'accéder à cette récompense, si possible en un temps minimal.

2 Processus de décision markovien

Un processus de décision markovien (MDP) est un modèle stochastique où un agent prend des décisions et où les résultats de ses actions sont aléatoires.[1]

Formellement, un MDP est un quadruplet S, A, T, R :

- Espace des états S (dénombrable ou continu)
- Espace des actions A (dénombrable ou continu)
- Fonction de transition $p(s, a, s') \rightarrow [0, 1]$ (déterministe ou probabiliste)
- Récompense r(s)

On cherche une stratégie π qui maximise l'espérance des récompenses cumulées.
L'horizon temporelle est notée H et peut être fini (problème épisodique) ou infini (problème coninu).

3 Equation de Bellman

- **La fonction d'évaluation V** décrit les meilleurs valeurs possibles de l'objectif selon l'état. Le calcul de V nous permet d'avoir également la fonction a(x) decrivant l'action optimale selon l'état : la fonction de politique.
- **L'équation de Bellman** est une condition nécessaire d'optimalité. Elle permet de connaitre la fonction d'évaluation V optimale et la politique optimale.

Voici l'équation générale[2] :

$$V(x) = \max_{a \in \Gamma(x)} \{F(x, a) + \beta V(T(x, a))\}$$

Dans le cas des processus de décision de Markov, l'équation de Bellman est une récursion des récompenses attendues :

[1] https://fr.wikipedia.org/wiki/Processus_de_décision_markovien
[2] https://en.wikipedia.org/wiki/Bellman_equation

$$V(s) = r(s) + \gamma \sum_{s'} p(s, \pi(s), s')V(s')$$

Il s'agit d'une équation fonctionnelle, dont l'inconnue est la fonction V. On voit que cela consiste en la somme de la récompense à l'état s et de γ fois la somme des évaluations des états suivants (par la politique π et pondéré par les probabilités dans le cas d'un problème non-déterministe). La récursion s'arrête aux états finaux. La **stratégie optimale** est celle qui maximise V ?

Hypothèse. Dans le cas déterministe :

$$V(s) = r(s) + \gamma V(s')$$

4 Dilemme Exploration Exploitation

Exemple du bandit manchot. Doit-on exploiter le bras que l'on estime être le meilleur ou bien explorer plus pour peut-être trouver un bras plus rentable ?

Regret

Dans un problème de bandit manchot, le regret après K essais est défini comme la différence entre la récompense que l'on obtiendrait en utilisant K fois la meilleur machine et l'espérance de la récompense après K essais :

$$r_K = K\mu^* - \sum_{k=1}^{K} E(\mu_{I_k})$$

Inégalité de Hoeffding

Soit r_1, ..., r_n i.i.d. dans [0, 1] ... Voir cours.

5 Les deux grandes approches

- **Model-free :** L'agent connait S et A mais pas P et R. On ne peut pas faire de programmation dynamique. Les valeurs sont estimées à partir d'interactions directes avec l'environnement (only rely on actual observations).
- **Model-based :** L'agent connaît S, A, P et dans certains cas R aussi.

Model-free	Model-based
Q-Learning	Value Iteration
Monte Carlo	Policy Iteration/Improvement
Temporal Difference	Policy Evaluation

TD peut se faire sur V ou sur Q (SARSA)

L'algorithme Dyna combine à la fois une approche model-free et une approche model-based.

6 Méthode UCB1

Algorithme sur les bandits manchots, Upper Confidence Bound.

1. On tire chaque machine une fois.
2. On calcule le score UCB1 de chaque machine à l'aide de la formule suivante :

$$UCB1_j = m_j + \sqrt{\frac{2ln(n)}{n_j}}$$

   avec m_j le score moyen de la machine j, n_j le nombre de fois qu'elle a été tirée et n le nombre de tirages totale.
   On tire la machine avec le plus grand score.
3. Étape 2 en boucle.

Le premier terme de la formule représente donc l'exploitation et le second l'exploration.

Cette méthode garanti une borne à la probabilité que la moyenne empirique dévie de l'espérance selon :

$$P(y + a < \mu) \leq e^{-2na^2}$$

Ou un truc du genre.
Avec Y la moyenne des tirages, ou je sais pas quoi.

7 Monte Carlo Tree Search

- **Selection:** start from root R and select successive child nodes down to a leaf node L. The section below says more about a way of choosing child nodes that lets the game tree expand towards most promising moves, which is the essence of Monte Carlo tree search.
- **Expansion:** unless L ends the game with a win/loss for either player, create one (or more) child nodes and choose node C from one of them.
- **Simulation:** play a random playout from node C. This step is sometimes also called playout or rollout.
- **Backpropagation:** use the result of the playout to update information in the nodes on the path from C to R.

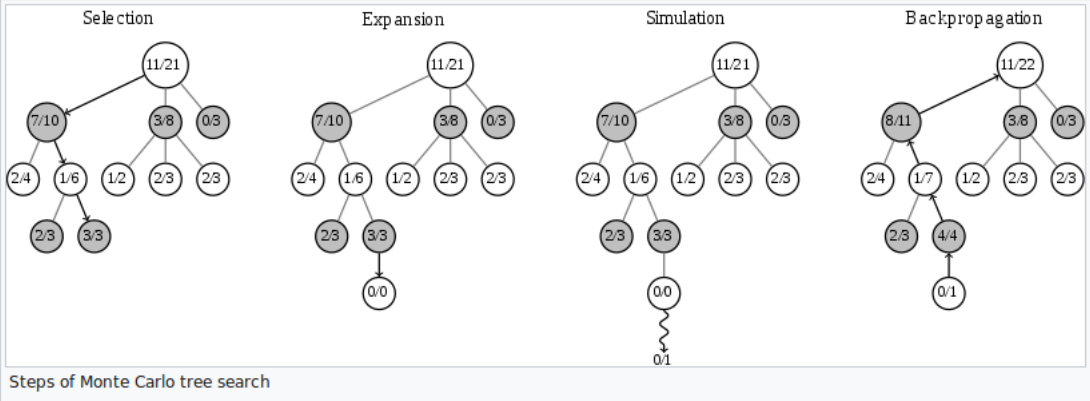


Figure 1: Schéma explicatif du MCTS