L'apprentissage par renforcement en machine learning

Nélia BOUZID, Clotilde CARROTTE, Émilie GARIAZZO, Lucile LAPRAY, Leslie LESAUVAGE, Aurélien VALDECASA

2 mars 2025

Résumé

En apprentissage par renforcement, les équations de Bellman stipulent qu'une politique optimale existe lorsque l'on se situe dans un processus de décision markovien (PDM). Ces PDM induisent un problème de décision markovien qui formalise la recherche d'une politique optimale, c'est-à-dire le choix d'actions à réaliser en fonction d'états successifs. L'optimisation d'une politique est l'objet de nombreux algorithmes, tel que le Q-learning. [8] Nous revenons dans cet écrit sur l'aspect théorique du processus de décision markovien ainsi que sur le problème d'optimisation tel que rendu possible par les équations de Bellman et par la propriété de convergence des algorithmes d'apprentissage par renforcement, et sur quelques exemples d'application.

Introduction

L'intelligence artificielle repose en grande partie sur l'apprentissage machine, qui permet d'entraîner une machine à atteindre un objectif sans lui fournir un protocole précis. Trois grands paradigmes existent : l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement, sur lequel nous nous concentrerons ici.

Inspiré des mécanismes d'apprentissage biologiques, l'apprentissage par renforcement repose sur un principe d'essais-erreurs. [1] Il vise à modéliser et automatiser la prise de décision en plaçant un agent dans un environnement où il peut se trouver dans diverses situations et entreprendre différentes actions, chacune modifiant son état et lui attribuant une récompense. L'objectif est d'apprendre à associer des situations aux actions qui maximisent cette récompense.

Les bases théoriques du domaine remontent aux années 1950 avec Richard Bellman, qui introduit la programmation dynamique et l'équation éponyme pour optimiser les décisions séquentielles (Bellman, 1957). En 1960, Ronald Howard développe l'itération sur les politiques pour résoudre les processus de décision markoviens (PDM). Paul Werbos, dans les années 1970, établit un lien entre programmation dynamique et apprentissage en proposant une approche heuristique du contrôle optimal (Werbos, 1977). Chris Watkins introduit en 1989 l'algorithme du Q-learning, qui permet d'apprendre une fonction de valeur sans modèle explicite de l'environnement. Enfin, l'ouvrage de Sutton et Barto, *Reinforcement Learning : An Introduction* (2018) [8], synthétise ces avancées et demeure une référence majeure du domaine.

Nous aborderons l'apprentissage par renforcement sous l'angle du contrôle optimal des processus de décision markoviens. Après avoir défini ces derniers et introduit les notions de processus stochastique, politique, valeur et qualité, nous examinerons les équations de Bellman et la propriété nécessaire de convergence des algorithmes d'optimisation avant de se pencher sur l'algorithme de Q-learning. Enfin, nous illustrerons ces concepts par plusieurs applications, comme le problème des bandits et l'exemple d'un aspirateur robot.

1 Vocabulaire et rappels : le processus de décision markovien

1.1 Processus de Markov

Définition. Un processus stochastique $X = (X_t)_{t \in T}$ est une famille de variables aléatoires X_t indexée par un ensemble T. Il représente une évolution discrète (ou à temps continu) d'une variable

aléatoire. [4] [10]

Propriété. Soit $X = (X_t)_{t \in T}$ un processus stochastique et E un espace d'états dénombrables. Le processus satisfait la propriété de Markov si pour tout $t \geq 0$, pour toute suite d'états $(i_0, \ldots, i_n, j) \in E^{t+2}$,

$$P(X_{t+1} = j | X_0 = i_0, X_1 = i_1, \ldots, X_t = i_t) = P(X_{t+1} = j | X_t = i_t).$$

On dit alors que le processus stochastique X est un processus de Markov. [12]

Remarque. Cette propriété stipule que le futur ne dépend pas du passé, sachant le présent.

1.2 Processus de décision markovien

Un processus de décision markovien (PDM), ou de Markov, est un cadre mathématique permettant de modéliser la prise de décision par un agent "intelligent", comme un robot ou un algorithme, dans un environnement où les résultats sont incertains. Autrement dit, lorsque l'agent effectue une action, il ne peut pas s'assurer avec certitude de ce que sera le résultat. Ainsi, dans un environnement stochastique, une même action peut produire des résultats différents. Le PDM est souvent utilisé en apprentissage par renforcement. En effet, c'est un domaine de l'intelligence artificielle où un agent apprend à prendre des décisions optimales en interagissant avec son environnement. En voici une définition formelle :

Rappel. Si une variable aléatoire possède une densité de probabilité, l'espérance est la moyenne des valeurs pondérées par cette densité. On la note $\mathbb{E}$.

Définition. Le processus de décision de Markov est défini par un quintuplet (S, A, P, R, γ) où [3] :

- S est l'ensemble des états du processus, c'est-à-dire l'ensemble des configurations possibles dans lesquelles le système peut se trouver à un instant donné. S est fini ; son cardinal est le nombre d'états.
- A est l'ensemble des actions possibles sur ce processus. A est fini.
- P est la fonction de transition : pour chaque couple (état, action), cette fonction indique la probabilité de passer d'un état $s \in S$ à un état futur $s' \in S$ en prenant une action $a \in A$.

$$S \times A \times S \to [0, 1]$$

$$(s, a, s') \mapsto P(s' \mid s, a)$$

- R est la fonction de récompense, qui mesure la qualité des transitions. À valeurs réelles, cette fonction formalise les conséquences d'une action émise dans un état. Elle représente les récompenses que l'agent obtient dans un état, ou après avoir mené une action. Par exemple, dans un jeu vidéo, si un agent ramasse une pièce, il reçoit une récompense de $+10$. S'il tombe dans un piège, il reçoit une pénalité de -50.
- γ est la valeur d'escompte et est comprise dans l'intervalle $[0, 1]$. C'est un scalaire qui permet de pondérer l'importance des récompenses futures qu'obtient un agent dans un environnement donné selon si la politique choisie est plus ou moins court-termiste (c'est-à-dire ne maximise pas les gains sur le long terme). En ajustant γ, on équilibre entre optimisation à court terme et vision à long terme. En pondérant, on stabilise l'évaluation de la politique, car les récompenses futures sont moins certaines que les récompenses immédiates. En reprenant l'exemple du jeu vidéo, si $\gamma = 0.9$, l'agent privilégiera le fait de ramasser plusieurs pièces sur le long terme plutôt que de prendre un raccourci risqué pour une récompense immédiate. Si $\gamma = 0.1$, il maximisera les gains immédiats sans se soucier du futur. γ sera important plus tard, lorsqu'on voudra savoir si l'état dans lequel l'agent se trouve est un bon état, ou si l'action qu'il mène est bonne : on pourra ainsi pondérer les récompenses futures qu'il obtiendra dans cet état / par cette action, pour voir si sa situation est pérenne sur le long terme.

Définition. On peut ajouter à ces définitions une indexation séquentielle. [8] Soit $T \subseteq \mathbb{N}$ l'ensemble des instants de décision. Alors on a les notations suivantes :

- On prend $s \in S$ un état indexé par $t \in T$, alors s_t est l'état du système à l'instant t.

— De même, si l'on prend $a \in A$ un état indexé par t, alors a_t est l'action prise d'après l'état s_t.
— Indexée par t, on a la fonction de transition : $P(s_{t+1} \mid s_t, a_t)$ qui donne la probabilité que le système soit dans l'état s_{t+1} étant donné l'état courant s_t et l'action réalisée $a_t \in A$.
— Indexée par t, on a la fonction de retour qui associe une récompense liée à l'action prise à un instant t, ou au fait d'être dans un état s à l'instant t.

Remarque. La succession d'états $(s_0, \ldots, s_t, \ldots)$ visités par le système se nomme une trajectoire. La trajectoire peut aussi se définir comme la séquence des états et des actions $(s_0, a_0, s_1, a_1, \ldots)$.

Définition. La dernière définition nous permet de mieux définir la fonction récompense R. Elle peut être utilisée et définie de plusieurs façons [3] :
— **Récompense attendue pour un état donné** : cela représente la récompense moyenne obtenue en étant dans l'état s. Elle est formalisée par l'espérance de la récompense immédiate R_t obtenue dans l'état actuel $S_t = s$:

$$R : S \to \mathbb{R}$$

$$R(s) = \mathbb{E}[R_t \mid S_t = s]$$

— **Récompense attendue pour une action donnée dans un état** : cela représente la récompense moyenne obtenue lorsqu'on prend l'action a dans l'état s. On la formalise comme l'espérance de la récompense immédiate liée à prendre l'action $A_t = a$ dans l'état $S_t = s$:

$$R : S \times A \to \mathbb{R}$$

$$R(s, a) = \mathbb{E}[R_t \mid S_t = s, A_t = a]$$

— **Récompense associée à une transition complète** : cette formulation prend en compte non seulement l'état et l'action, mais aussi l'état résultant. Il s'agit de la récompense moyenne associée au fait d'être, à l'instant t, dans un état s, et de passer à l'instant $t+1$ à un état s'. On la formalise ainsi :

$$R : S \times A \times S \to \mathbb{R}$$

$$R(s, a, s') = \mathbb{E}[R_t \mid S_t = s, A_t = a, S_{t+1} = s']$$

Remarque. Dans un PDM, l'agent prend des décisions en fonction de l'état actuel du système. L'état influence les actions possibles et les transitions vers d'autres états.

Exemple. Dans un jeu d'échecs, un état pourrait être la disposition à un certain instant de toutes les pièces présentes sur l'échiquier. Dans cet exemple l'agent prendrait la place de l'un des joueurs d'échecs. Il aurait ainsi, pour chaque instant t, l'information de la disposition des pièces sur l'échiquier, mais pas l'information du prochain coup de son adversaire. En effet, il est important de comprendre que l'état de l'agent correspond aux observations qu'il fait sur sa situation, mais la connaissance d'un état ne garantit pas que l'agent ait en sa possession toutes les informations.

1.3 Politique

Définition. Une politique, notée π, spécifie la manière dont l'agent choisit son action [3]. Mathématiquement, il s'agit d'une fonction qui associe à chaque état s une action a, soit de manière déterministe, soit de manière probabiliste Elle est formalisée par la fonction $\pi : S \times A \to [0, 1]$ où S est l'ensemble des états possibles, A est l'ensemble des actions disponibles et $\pi(a \mid s)$ est la probabilité de choisir l'action $a \in A$ lorsque l'agent est dans l'état $s \in S$. Voici le principe de quelques types de politiques :
— **Politique déterministe** : l'agent choisit toujours la même action pour un état donné.
— **Politique stochastique** : l'action est choisie selon une distribution de probabilités.
— **Politique aléatoire** : l'agent choisit l'action de manière aléatoire, quel que soit l'état.
— **Politique non stationnaire** : la politique évolue au cours du temps.

Définition. La fonction valeur, ou valeur, est une application qui associe à tout état sa valeur pour une certaine politique. On la note V^π. Elle représente le retour espéré si l'agent commence en

s et suit la politique π, et répond ainsi à la question "Est-il bon de se trouver dans l'état s ?" [5]. La fonction de valeur V^π est donc définie mathématiquement comme l'espérance des récompenses futures cumulées en suivant la politique π à partir de l'état s. Elle prend en entrée une valeur s et donne en sortie une valeur réelle, qui indique s'il est bon ou non de se trouver dans l'état s [8]. On a ainsi $V^\pi : S \to \mathbb{R}$ et

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid s_0 = s\right]$$

où R_t est la récompense obtenue au temps t, et l'espérance est prise sur toutes les trajectoires possibles induites par la politique π. La somme accumule les récompenses obtenues à chaque instant t, et le facteur d'escompte γ pondère l'importance des récompenses futures : si γ est proche de 1, les récompenses futures comptent presque autant que les récompenses immédiates, si γ est proche de 0, seules les récompenses immédiates sont prises en compte. La somme des récompenses pondérées $\sum_{t=0}^{\infty} \gamma^t R_t$ permet donc de savoir si, à long terme, la politique choisie est la meilleure : une politique très court-termiste aura une moins bonne somme des récompenses pondérées car γ s'approchera très rapidement de 0. On peut donc lire cette somme comme la moyenne des futures récompenses en suivant la politique π, étant plus ou moins une stratégie à long terme (modélisée par γ), lorsque l'on se trouve dans l'état s.

Définition. $Q^\pi(s, a)$ est la qualité de l'état s quand on y effectue l'action a puis que l'on applique la politique π. Dans la littérature, on trouve les appellations de "qualité" et de "valeur d'état". Dans cet écrit, nous utiliserons plutôt le terme de qualité afin d'éviter les ambiguïtés avec la fonction valeur. Ainsi, la qualité est la valeur d'une action spécifique dans un état s, avant de continuer selon π. La valeur rendue par cette application répond à la question "Est-il bon de mener l'action a dans l'état s ?" [5]. Il s'agit ainsi de la moyenne pondérée de la récompense obtenue après avoir exécuté l'action a alors qu'on est dans l'état s, et qu'on suit une politique π. Effectivement, puisque l'on se trouve dans un environnement stochastique, une même action a ne mène pas toujours au même état s' : on pondère donc la récompense d'arriver en l'état s' après avoir mené l'action a par la probabilité de passer de s à s' en exécutant a. On peut écrire cela comme l'espérance de la récompense immédiate $R(s)$ lorsque l'agent commence dans l'état s, prend l'action a, et suit ensuite la politique π pour les actions futures [3] :

$$Q^\pi : S \times A \to \mathbb{R}$$

$$Q^\pi(s, a) = \mathbb{E}[R(s) \mid s_0 = s, a_0 = a, a_{t>0} \sim \pi]$$

ou comme l'espérance de la somme des récompenses futures, pondérées par le facteur d'escompte γ, lorsque l'agent commence dans l'état s, prend l'action a, et suit la politique π pour toutes les actions futures [8] :

$$Q^\pi : S \times A \to \mathbb{R}$$

$$Q^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s, A_0 = a\right]$$

Remarque. La première définition mathématique équivaut à la deuxième : la première est l'estimation de la récompense obtenue en performant l'action a (ce qui mène à obtenir $R(s)$) puis en suivant la politique π, ce qui revient à l'estimation de la somme des récompenses obtenues en suivant la politique π (ce qui correspond à la deuxième définition).

Remarque. Calculer la qualité de l'état s quand on effectue l'action a en suivant la politique π revient à calculer la récompense future attendue en prenant l'action a depuis l'état s en suivant la politique π.

Remarque. Alors que la valeur d'un état indique s'il est bon de passer par cet état pour optimiser la fonction objectif, la qualité indique s'il est bon d'exécuter une certaine action dans un certain état.

2 Résultats

2.1 Équations de Bellman [2] [3]

Définition. V^π et Q^π vérifient chacune une équation qui permet de calculer exactement leur valeur.

$$V^\pi(s) = \sum_{a \in A} \pi(s,a) \sum_{s' \in S} P(s,a,s') \left[R(s,a,s') + \gamma V^\pi(s') \right] \tag{1}$$

et

$$Q^\pi(s,a) = \sum_{s' \in S} P(s,a,s') \left[R(s,a,s') + \gamma \sum_{a' \in A} \pi(s',a') Q^\pi(s',a') \right] \tag{2}$$

Par ailleurs, la politique optimale est la meilleure possible en tout état. En notant V^* la fonction valeur d'une politique optimale, on peut donc écrire :

$$V^*(s) = \max_\pi V^\pi(s), \quad \forall s \in S$$

Ainsi, en maximisant V^* sur les actions possibles, on trouve :

$$V^*(s) = \max_{a \in A(s)} \sum_{s' \in S} P(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right] \tag{3}$$

$Q^*(s,a)$ est la valeur optimale de la paire état-action (s,a). Elle représente l'espérance du retour cumulé (somme des récompenses) que l'on peut obtenir en prenant l'action a dans l'état s et en suivant ensuite la politique optimale. $Q^*(s,a)$ est essentiellement la valeur de $Q^\pi(s,a)$ lorsque π est la politique optimale. En d'autres termes, $Q^*(s,a)$ est la valeur maximale que peut atteindre $Q^\pi(s,a)$ quelle que soit la politique π.

$$Q^*(s,a) = \sum_{s' \in S} P(s,a,s') \left[R(s,a,s') + \gamma \max_{a' \in A} Q^*(s',a') \right] \tag{4}$$

Ces quatre équations sont nommées « équations de Bellman ».

Démonstration :
(1) Partons de la définition de V^π :

$$V(s) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R_t \mid s_0 = s \right].$$

En suivant la définition de $V^\pi(s)$, on peut réécrire la somme des récompenses futures comme :

$$V^\pi(s) = \mathbb{E}_\pi \left[R_0 + \gamma \sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid s_0 = s \right].$$

En utilisant la linéarité de l'espérance :

$$V^\pi(s) = \mathbb{E}_\pi[R_0 \mid s_0 = s] + \gamma \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid s_0 = s \right].$$

Mais le terme à droite de γ est juste la valeur attendue à partir de l'état suivant S_1 sous la politique π, donc :

$$V^\pi(s) = \mathbb{E}_\pi[R_0 \mid s_0 = s] + \gamma \mathbb{E}_\pi[V^\pi(s_1) \mid s_0 = s].$$

L'espérance des récompenses est obtenue en sommant sur toutes les actions possibles a et les états suivants s' :

1. La politique $\pi(s,a)$ donne la probabilité de choisir l'action a.
2. La dynamique de l'environnement $P(s,a,s')$ donne la probabilité d'atteindre s' depuis s avec a.
3. La récompense immédiate est $R(s,a,s')$.

Donc :

$$\mathbb{E}_\pi[R_0 \mid s_0 = s] = \sum_{a \in A} \pi(s,a) \sum_{s' \in S} P(s,a,s') R(s,a,s')$$

et

$$\mathbb{E}_\pi[V^\pi(s_1) \mid s_0 = s] = \sum_{a \in A} \pi(s,a) \sum_{s' \in S} P(s,a,s') V^\pi(s').$$

En combinant ces deux résultats dans l'équation précédente, on retrouve :

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

(2) Par définition, on a :

$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid s_0 = s, a_0 = a \right].$$

On extrait la première récompense obtenue après avoir pris $a_0 = a$:

$$Q^\pi(s,a) = \mathbb{E}\left[R_0 + \sum_{t=1}^{\infty} \gamma^t R_t \mid s_0 = s, a_0 = a \right].$$

En effectuant un changement d'indice, on obtient :

$$Q^\pi(s,a) = \mathbb{E}\left[R_0 + \sum_{t=0}^{\infty} \gamma^{t+1} R_{t+1} \mid s_0 = s, a_0 = a \right].$$

D'où

$$Q^\pi(s,a) = \mathbb{E}\left[R_0 + \gamma \sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid s_0 = s, a_0 = a \right].$$

Or, la somme restante correspond exactement à la valeur d'un état futur s' sous la politique π, donc on reconnaît :

$$Q^\pi(s,a) = \mathbb{E}\left[R_0 + \gamma V^\pi(s') \mid s_0 = s, a_0 = a \right].$$

On a déjà vu en (1) comment exprimer l'espérance de ces variables. Ici, on applique ces expressions à une action a particulière, car $Q^\pi(s,a)$ exprime s'il est bien ou non de mener une action a dans l'état s. On trouve :

$$Q^\pi(s,a) = \sum_{s' \in S} P(s,a,s') \left[R(s,a,s') + \gamma V^\pi(s') \right].$$

Par définition, la fonction valeur V^π peut s'écrire en fonction de Q^π, puisque la politique π détermine quelle action a' sera jouée dans s' :

$$V^\pi(s') = \sum_{a' \in A} \pi(s',a') Q^\pi(s',a').$$

On injecte donc cette équation dans la précédente et on obtient l'équation de Bellman :

$$Q^\pi(s,a) = \sum_{s' \in S} P(s,a,s') \left[R(s,a,s') + \gamma \sum_{a' \in A} \pi(s',a') Q^\pi(s',a') \right].$$

(3) et (4) sont démontrés en prenant le maximum des V^π et Q^π. □

2.2 Q-learning

Dans cette partie nous introduisons l'algorithme Q-learning, qui est un résultat majeur de l'apprentissage par renforcement. Nous présenterons d'abord le critère de Robbins-Monro, qui est important pour l'efficacité de l'algorithme Q-learning.

2.2.1 Critère de Robbins-Monro

Le critère de Robbins-Monro est une condition essentielle pour la convergence des algorithmes de mise à jour incrémentale (qui mettent à jour leurs estimations d'une certaine valeur ou les paramètres d'un modèle de manière progressive), comme ceux utilisés dans l'apprentissage par renforcement et en particulier pour le Q-learning. Il assure que la suite des mises à jour ne diverge pas et converge vers une valeur optimale.

Définition. On note α le taux d'apprentissage d'un algorithme d'optimisation de politique. Le paramètre α détermine l'influence des nouvelles observations sur la mise à jour des calculs de valeurs ou qualités utilisés pour l'optimisation. On peut indexer séquentiellement α afin qu'il diminue [8] (ou augmente, mais cette pratique est moins courante) au fil du temps, en prenant par exemple $\alpha_t = \frac{1}{t}$.

Définition. On appelle fonction de mise à jour la fonction qui ajuste les paramètres d'un modèle en fonction des nouvelles observations reçues. En apprentissage par renforcement, elle est utilisée pour améliorer progressivement l'estimation des valeurs d'état $V(s)$, des qualités d'action $Q(s, a)$ ou des paramètres d'une politique $\pi(a \mid s)$.

Définition. Une mise à jour itérative est définie par : $\theta_{t+1} = \theta_t + \alpha_t F(\theta_t, X_t)$, où α est le taux d'apprentissage et $F(\theta_t, X_t)$ est une fonction de mise à jour basée sur la nouvelle observation X_t.

Remarque. On cherche ainsi un paramétrage optimal θ^* tel que $\mathbb{E}[F(\theta^*, X_t)] = 0$. En effet, on souhaite que la mise à jour tende vers une valeur stable où l'espérance de l'ajustement est nulle.

Critère de Robbins-Monro [14]. Pour garantir la convergence de θ_t, il faut que α_t respecte les deux conditions suivantes :

- Sommation divergente : $\sum_{t=1}^{\infty} \alpha_t = \infty$ Cela garantit que l'algorithme continue d'apprendre et ne s'arrête pas trop tôt.
- Sommation quadratique convergente : $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$ Cela empêche les mises à jour d'être trop grandes et de provoquer des oscillations instables.

Remarque. Si α est grand ($\alpha \approx 1$), l'algorithme s'adapte rapidement aux nouvelles expériences, mais peut être instable. À l'inverse, si α est petit ($\alpha \approx 0$), l'apprentissage est plus lent, mais les valeurs apprises sont plus stables. [8] Enfin, selon le critère de Robbins-Monro, si α décroît avec le temps, on garantit presque sûrement la convergence de la séquence d'ajustement des paramètres.

2.2.2 Introduction au Q-learning

Le Q-learning est un algorithme d'apprentissage par renforcement qui permet à un agent d'apprendre à prendre des décisions optimales dans un environnement incertain.

Comme nous l'avons vu en section 1 sur les processus de décision markoviens et en section 2.1 sur les équations de Bellman, l'optimisation de la politique optimale repose sur des fondements théoriques solides. Le Q-learning s'inscrit naturellement dans ce cadre : il offre une méthode d'apprentissage par renforcement hors-politique, qui permet d'estimer directement la fonction de valeur optimale Q^* sans disposer d'un modèle explicite de l'environnement.

L'objectif principal du Q-learning est de déterminer la meilleure politique de décision pour un agent, c'est-à-dire la séquence d'actions qui maximise la récompense cumulée au fil du temps. Pour ce faire, le Q-learning estime la fonction de valeur optimale $Q^*(s, a)$, qui représente la qualité d'une action a dans un état donné s. En effet, l'estimation est une meilleure stratégie que le calcul direct des équations de Bellman qui n'est souvent pas possible, car les informations sur les fonctions de transition (P) et de récompense (R) sont généralement inconnues ou difficiles à modéliser dans les environnements réels. De plus, si l'environnement est vaste, la puissance de calcul nécessaire à une connaissance exhaustive du système devient trop coûteuse. Ainsi, le Q-learning utilise une approche itérative d'exploration/exploitation, plus efficace que le calcul direct.

2.2.3 Fonctionnement du Q-learning [3]

Le Q-learning est un algorithme basé sur les différences temporelles (TD). Il repose sur la même intuition que les équations de Bellman présentées précédemment : pour tout état s, la valeur optimale est obtenue en maximisant sur les actions futures. Cependant, le Q-learning améliore l'estimation de Q^* au fil des interactions de l'agent avec l'environnement : cette démarche est particulièrement utile dans des contextes où l'environnement est inconnu ou sujet à des variations, comme c'est le cas dans plusieurs applications (cf section 3). Dans un algorithme de Q-learning, l'agent commence sans connaissance sur son environnement. Il cherche donc à l'explorer, et à en extraire des connaissances qu'il pourra ensuite utiliser (ou, dans les termes du Q-learning, exploiter) pour affiner sa politique.

- Il associe à chaque situation (état) et action possible une valeur initiale arbitraire (souvent 0). Ensuite, il explore et teste différentes actions, recevant à chaque fois une récompense plus ou moins forte.
- À chaque étape, l'agent doit décider entre l'exploration ou l'exploitation. Explorer consiste à essayer une action au hasard pour découvrir de nouvelles possibilités. À l'inverse, exploiter consiste à choisir l'action qui semble la meilleure en fonction des expériences passées.
- Pour équilibrer ces deux stratégies, il suit une règle mixte, appelée stratégie ϵ-gloutonne : avec une probabilité ϵ, il choisit une action au hasard pour explorer. Sinon, il choisit l'action qui a donné les meilleurs résultats jusqu'à présent (avec probabilité $1 - \epsilon$).
- Il s'agit ensuite d'apprendre en ajustant ses choix. Quand l'agent effectue une action, il observe la récompense immédiate, ainsi que l'état dans lequel il se retrouve après cette action. Il met alors à jour sa manière d'évaluer cette action en ajustant progressivement sa perception des bonnes décisions.
- Au début, les choix de l'agent sont très aléatoires, mais au fil des essais, il affine ses décisions. Progressivement, il construit une stratégie efficace, où il sait quelles actions choisir pour obtenir les meilleures récompenses. Une fois que ses évaluations se stabilisent, on dit qu'il a convergé : il a appris une stratégie optimale et peut prendre des décisions intelligentes sans tâtonner.

2.2.4 Définition formelle et algorithme de Q-learning [3] [8]

Nous proposons ici une définition formelle de l'algorithme Q-learning et une description de l'algorithme utilisant les notations mathématiques :

Définition. Considérons un PDM (S, A, P, R, γ) tel que défini dans la section 1. Le Q-learning met à jour la fonction d'action $Q(s, a)$ selon la règle :

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

avec :

- $\alpha \in (0, 1]$ le taux d'apprentissage,
- $\gamma \in [0, 1]$ le facteur d'escompte,
- $\max_{a'} Q(s', a')$ représentant l'estimation de la valeur optimale pour l'état suivant.

L'idée est de mettre à jour $Q(s, a)$ après chaque interaction avec l'environnement en réduisant l'erreur entre l'estimation actuelle et une meilleure approximation. Cette erreur est définie par :

$$\delta = R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)$$

où $R(s, a)$ est la récompense immédiate et $\gamma \max_{a'} Q(s', a')$ l'estimation actualisée des récompenses futures maximales pondérée par γ. Plutôt que de remplacer directement $Q(s, a)$, on l'ajuste progressivement avec le taux d'apprentissage. L'algorithme s'exécute de la manière suivante :

1. Initialiser $Q(s, a)$ de façon arbitraire pour tout $s \in S$ et $a \in A(s)$, en imposant par convention $Q(s, a) = 0$ pour tout $s \in S$ et pour tout $a \in A$.
2. Pour chaque itération :
   - Initialiser l'état S.
   - Pour chaque étape de l'itération :
     - Sélectionner une action A depuis S selon une stratégie ϵ-gloutonne.
     - Observer la récompense R et l'état suivant S'.
     - Mettre à jour la valeur $Q(s, a)$ selon la règle ci-dessus.
     - Affecter $s \leftarrow s'$.
   - Continuer jusqu'à atteindre un état terminal.

2.2.5 Convergence et garanties théoriques

Dans le cadre du Q-learning, on a la convergence de l'algorithme sous les conditions suivantes [8] :

- Les ensembles S et A, d'états et d'actions, sont finis.
- Une exploration suffisante assurant que toutes les paires (s, a) sont visitées un nombre infini de fois ;
- Un taux d'apprentissage α respectant les critères de Robbins-Monro ;
- Un environnement stationnaire avec des récompenses bornées, c'est-à-dire que les probabilités de transition et les récompenses ne changent pas au cours du temps et les récompenses que l'agent peut recevoir sont comprises dans un intervalle fini.

On a le résultat de l'algorithme Q-learning qui converge presque sûrement vers la fonction de valeur optimale Q^*, c'est-à-dire que les valeurs $Q(s, a)$ apprises par l'algorithme vont se rapprocher de $Q^*(s, a)$ pour chaque état et action.

Remarque. Il existe d'autres algorithmes dont le résultat converge vers Q^* ou V^*. Leur convergence repose généralement sur plusieurs hypothèses [3] :

- On contraint les valeurs du facteur d'escompte à $\gamma \in [0, 1]$. En effet, le facteur d'escompte assure que les récompenses futures ont un poids décroissant et empêche une explosion des valeurs.
  - Généralement, on impose $\gamma \in [0, 1[$ pour garantir la convergence.
  - Si $\gamma = 1$, la somme des récompenses futures peut diverger, sauf si les récompenses sont bornées et que l'horizon temporel est fini.
- On suppose que S et A, les ensembles d'états et d'actions, sont de cardinaux finis. Cette hypothèse est suffisante si l'on se trouve dans un cadre discret, mais des hypothèses supplémentaires sont requises dans un cadre continu.
- On considère aussi que l'exploration de l'environnement par l'agent est suffisante. Cela signifie que les paires état-action (s, a) doivent être visitées un nombre suffisant de fois pour garantir une approximation correcte des valeurs $V(s)$ et des qualités $Q(s, a)$.

3 Applications

3.1 Un robot aspirateur

Un robot aspirateur doit nettoyer une pièce divisée en plusieurs zones. Chaque zone correspond à un état (S), et le robot peut effectuer plusieurs actions (A) comme avancer, tourner ou aspirer la poussière.

$$S = \{\text{Départ}, \text{Zone1}, \text{Zone2}, \text{Obstacle}, \text{Destination}\}$$

$$A = \{\text{Avancer}, \text{Tourner à gauche}, \text{Tourner à droite}, \text{Attendre}\}$$

Lorsqu'il choisit une action, il ne sait pas toujours exactement où il se retrouvera à cause d'éventuels obstacles ou erreurs de navigation. La fonction de transition $P(s' \mid s, a)$ représente cette incertitude.

- Si le robot avance depuis Zone1, il arrive à Zone2 avec 80 % de chances, mais rencontre un Obstacle avec 20 % de chances. On a :

  $$P(\text{Zone2} \mid \text{Zone1}, \text{Avancer}) = 0.8$$

  et

  $$P(\text{Obstacle} \mid \text{Zone1}, \text{Avancer}) = 0.2$$

  .
- Si le robot tourne à gauche, il réussit avec 90 % de chances mais reste bloqué avec 10 % de chances. On a :

  $$P(\text{Zone2} \mid \text{Zone1}, \text{Tourner à gauche}) = 0.9$$

  et

  $$P(\text{Zone1} \mid \text{Zone1}, \text{Tourner à gauche}) = 0.1$$

La fonction de récompense $R(s, a, s')$ quantifie l'utilité de ces actions :

- $R(\text{Zone1}, \text{Avancer}, \text{Zone2}) = -1$
- $R(\text{Zone1}, \text{Avancer}, \text{Obstacle}) = -10$
- $R(\text{Zone2}, \text{Avancer}, \text{Destination}) = +100$

Le facteur d'escompte $\gamma \in [0, 1]$ contrôle l'importance des récompenses futures.

— Si γ est proche de 1, le robot prend des décisions en pensant à long terme.
— Si γ est proche de 0, il privilégie les récompenses immédiates.
— Par exemple, si $\gamma = 0.9$, alors les récompenses futures sont comptées avec une pondération décroissante : $V(s) = R(s, a, s') + \gamma R(s', a', s'') + \gamma^2 R(s'', a'', s''') + \dots$

Le but du robot aspirateur est d'apprendre une politique optimale qui lui permet de maximiser ses points en nettoyant efficacement tout en évitant les obstacles. Autrement dit, le robot cherche une politique optimale $\pi^*(s)$ qui maximise la valeur des états $V(s)$. $V(s)$ est la valeur d'un état s sous une politique π. Elle représente l'espérance (c'est-à-dire la moyenne attendue) des récompenses futures si l'agent commence en s et suit la politique π. Elle est définie par :

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s\right].$$

L'espérance mathématique indique qu'on prend la moyenne pondérée de toutes les séquences possibles de récompenses, en tenant compte de leur probabilité d'occurrence. On somme toutes les récompenses futures R_t que l'agent va obtenir en suivant π. Chaque récompense est pondérée par γ, qui réduit l'importance des récompenses lointaines. Par ailleurs, la qualité d'une action dans un état donné est évaluée par la fonction :

$$Q^\pi(s, a) = R(s, a, s') + \gamma \sum_{t=0}^{\infty} P(s' \mid s, a) V^\pi(s').$$

$P(s' \mid s, a)$ est la probabilité d'atteindre s' depuis s en prenant a, et $V(s')$ est la valeur de l'état suivant. Pour maximiser ses récompenses, l'agent peut utiliser des algorithmes comme le Q-learning ou les réseaux de neurones profonds (DQN) afin d'améliorer ses décisions à chaque nouvelle expérience.

3.2 Le problème des bandits : un cas élémentaire de PDM

Les problèmes de bandits illustrent simplement les concepts clés des PDM tout en introduisant le dilemme exploration vs exploitation. Considérons un casino avec K machines à sous. Chaque machine (ou "bras") donne une récompense aléatoire d'espérance μ_k inconnue. L'objectif est de maximiser les gains cumulés sur T essais.

3.2.1 Modélisation comme PDM dégénéré

Ce problème se formalise comme un PDM (S, A, P, R) où :
— $S = \{s\}$ (un seul état : le joueur reste devant les machines),
— $A = \{1, \dots, K\}$ (actions = choix de machine),
— $P(s'|s, a) = 1$ avec s' = s (l'état ne change jamais),
— $R(a) \in [0, 1]$ suit une loi de probabilité inconnue.

Contrairement aux PDM généraux, il n'y a pas de transitions d'états. La fonction valeur se simplifie alors en l'espérance de gain pour chaque action :

$$Q^*(a) = \mathbb{E}[R(a)].$$

3.2.2 Estimation incrémentale des récompenses

À chaque essai t, on maintient pour chaque action a :
— $N_t(a)$: nombre de fois où a a été choisie,
— $Q_t(a) = \frac{1}{N_t(a)} \sum_{i=1}^{N_t(a)} R_i(a)$: récompense moyenne observée.

Après avoir choisi l'action a_t et reçu R_t, on met à jour :

$$Q_{t+1}(a_t) = Q_t(a_t) + \frac{1}{N_{t+1}(a_t)}(R_t - Q_t(a_t)).$$

Cette formule rappelle l'équation de Bellman simplifiée, où la "valeur" d'une action converge vers son espérance réelle.

3.2.3 Exemple : Machines à sous

Imaginons des machines à sous où chaque victoire allume une lumière rouge (échec) ou verte (succès). Chaque machine a a une probabilité inconnue μ_a de produire une lumière verte (récompense $R(a) = 1$). Comment répartir les essais entre les machines pour maximiser le nombre total de lumières vertes, tout en apprenant leurs caractéristiques ?

3.2.4 Stratégies fondamentales

Pour résoudre le dilemme exploration/exploitation, trois approches classiques existent :

— **Gloutonne** ($\epsilon = 0$) : Choix systématique de l'action au $Q_t(a)$ maximal. Risque de se bloquer sur un sous-optimum.

— **ϵ-gloutonne** : Avec probabilité ϵ, exploration aléatoire. Par exemple pour $\epsilon = 0.1$:

$$a_t = \begin{cases} \text{Uniforme}(1, K) & \text{avec proba } 0.1, \\ \arg\max_a Q_t(a) & \text{avec proba } 0.9. \end{cases}$$

— **UCB (Upper Confidence Bound)** : Choix optimiste de l'action maximisant :

$$Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}},$$

où c contrôle le niveau de confiance. Cette borne supérieure favorise les actions peu explorées mais potentiellement bonnes.

3.2.5 Application numérique

Prenons $K = 2$ machines avec $\mu_1 = 0.6$, $\mu_2 = 0.7$. Une stratégie gloutonne pourrait faire :

1. Essai 1 : Choix aléatoire (machine 1, gain 1). $Q_1(1) = 1.0$.
2. Essai 2 : Exploitation de la machine 1 (gain 0). $Q_2(1) = 0.5$.
3. Essai 3 : Machine 1 encore (gain 1). $Q_3(1) = 0.66$.
4. ... Jamais de test de la machine 2 !

Une stratégie ϵ-gloutonne ($\epsilon = 0.3$) aurait permis d'essayer occasionnellement la machine 2 et découvrir sa meilleure performance.

3.2.6 Convergence et lien avec les PDM généraux

Sous des conditions d'exploration suffisantes (comme $\epsilon_t \to 0$ ou le critère de Robbins-Monro), on garantit que $Q_t(a) \to \mu_a$. Ce résultat découle directement de la convergence des équations de Bellman (Section 2.1) dans ce PDM simplifié.

3.2.7 Apprentissage incrémental des valeurs d'actions

L'estimation des valeurs $Q_t(a)$ suit une démarche similaire à la résolution itérative des équations de Bellman (Section 2.2.5). La mise à jour :

$$Q_{t+1}(a) = Q_t(a) + \alpha_t \left(R_t - Q_t(a)\right)$$

correspond à une version stochastique de l'équation de Bellman pour un PDM à état unique. Le pas d'apprentissage $\alpha_t = \frac{1}{N_t(a)}$ satisfait les conditions de Robbins-Monro :

$$\sum_{t=1}^{\infty} \alpha_t = \infty \quad \text{et} \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty,$$

garantissant la convergence vers $Q^*(a) = \mathbb{E}[R(a)]$. Ce mécanisme est analogue à la mise à jour du Q-learning où $Q(s, a)$ évolue par essais-erreurs.

3.2.8 Environnements non stationnaires et adaptation

Dans des contextes réels (publicité en ligne), les espérances μ_a peuvent évoluer dans le temps. On remplace alors le pas décroissant $\alpha_t = 1/N_t(a)$ par un pas constant $\alpha \in (0, 1]$:

$$Q_{t+1}(a) = (1 - \alpha)Q_t(a) + \alpha R_t.$$

Cette règle donne plus de poids aux récentes observations, permettant de suivre des variations lentes. Par exemple, avec $\alpha = 0.1$, une récompense $R_t = 1$ mettrait à jour $Q_t(a)$ comme :

$$Q_{t+1}(a) = 0.9 \times Q_t(a) + 0.1 \times 1.$$

Conclusion

L'apprentissage par renforcement repose sur la formalisation des processus de décision markoviens, permettant à un agent d'optimiser ses actions en fonction d'un environnement incertain. Les équations de Bellman garantissent l'existence d'une politique optimale que l'on peut calculer explicitement si on a une connaissance exhaustive de cet environnement. Néanmoins, cette condition est très rarement respectée, ce qui rend le processus d'optimisation plus complexe. Ainsi, les algorithmes d'apprentissage par renforcement consistent généralement à estimer une politique optimale. Afin d'assurer un résultat cohérent et éviter que les algorithmes ne calculent à l'infini, il est nécessaire que leur recherche converge vers la valeur optimale. Cette convergence est assurée de diverses manières, comme par le respect du critère de Monro-Robbins dans l'algorithme de Q-learning. À travers les exemples étudiés, nous avons illustré comment ces concepts s'appliquent dans divers contextes. L'apprentissage par renforcement continue d'évoluer, notamment avec l'essor des méthodes profondes qui améliorent la capacité des agents à généraliser leurs décisions dans des environnements complexes. On peut également noter que le paradigme est passé d'une optimisation par maximisation de récompense, à optimisation par minimisation des coûts. [1]

Références

[1] Mehdi Khamassi, *How can we assure AI systems' alignment with human values ?* 2025. Colloque Embed-days, ENS-PSL, 28/02/25.

[2] Kurtis Pykes, *Comprendre l'équation de Bellman dans l'apprentissage par renforcement* 2025. Datacamp. 02/2025 : https://www.datacamp.com/fr/tutorial/bellman-equation-reinforcement-learning?utm_source=chatgpt.com.

[3] Philippe Preux, *Digest AR*, 2024. *Notes de cours d'apprentissage par renforcement* 02/2025 : https://philippe-preux.github.io/Documents/digest-ar.pdf.

[4] Jean-Christophe Breton, *Processus stochastiques* 2024. Université de Rennes. Cours M2 Mathématiques. 02/2025 : https://perso.univ-rennes1.fr/jean-christophe.breton/Fichiers/processus_M2.pdf

[5] Ajay Halthor, *Bellman Equation - Explained !* 2023. CodeEmporium. 02/2025 : https://www.youtube.com/watch?v=9JZID-h6ZJ0

[6] E. Franck, Y. Privat, *Contrôler, optimiser, décider*, 2022. Gazette de la société mathématique de France, n°174. (Partie 4 - p.17 à 23). 02/25 : https://smf.emath.fr/system/files/filepdf/G174-BD.pdf.

[7] Anon, *Matchbox Educable Noughts and Crosses Engine* 2021. Wikipédia. 02/2025 : https://en.wikipedia.org/wiki/Matchbox_Educable_Noughts_and_Crosses_Engine

[8] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning : An Introduction* 2nd edition, 2018. A Bradford Book. The MIT Press. ISBN-10 0262039249.

[9] Jérôme Mengin, *Apprentissage par renforcement* 2017. Institut de Recherche en Informatique de Toulouse (IRIT). 02/2025 : https://www.irit.fr/~Jerome.Mengin/teaching/mach-learn/ap-renforc_diapos.pdf

[10] Anon, *Processus stochastique* 2015. Wikipédia. 02/2025 : https://fr.wikipedia.org/wiki/Processus_stochastique

[11] Hugo Larochelle et Froduald Kabanza, *Intelligence Artificielle [11.2] : Processus de décision markovien - définitions* 2013. Université de Sherbrooke. 02/2025 : https://www.youtube.com/watch?v=mxHhrP4fVmM

[12] Pierre-Loïc Méliot, *Théorie des chaînes de Markov* S.-D. Laboratoire de Mathématiques d'Orsay, Université Paris-Saclay. 02/2025 : https://www.imo.universite-paris-saclay.fr/~pierre-loic.meliot/markov/markov_theory.pdf

[13] Donald Michie, *Experiments on the mechanization of game-learning. Part I. Characterization of the model and its parameters* 1963. *The Computer Journal*, Volume 6, Issue 3, p. 232-236. 02/2025 : https://people.csail.mit.edu/brooks/idocs/matchbox.pdf

[14] Herbert Robbins, Sutton Monro, *A Stochastic Approximation Method* 1951. *The Annals of Mathematical Statistics*, 22(3). p.400-407. 02/2025 : https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-22/issue-3/A-Stochastic-Approximation-Method/10.1214/aoms/1177729586.full