

Q-Learning

Processus d'Apprentissage basé sur un MDP

rivo.link@gmail.com

1 Markov Decision Process (MDP)

D'après Wikipédia: Un processus de décision markovien est un processus de contrôle stochastique discret. À chaque étape, le processus est dans un certain état \mathbf{s} et l'agent choisit une action \mathbf{a} . La probabilité que le processus arrive à l'état \mathbf{s}' est déterminée par l'action choisie. Plus précisément, elle est décrite par la fonction de transition d'états $\mathbf{T}(\mathbf{s}, \mathbf{a}, \mathbf{s}')$. Donc, l'état \mathbf{s}' dépend de l'état actuel \mathbf{s} et de l'action \mathbf{a} sélectionnée par le décideur. Cependant, pour un état \mathbf{s} et une action \mathbf{a} , le prochain état est indépendant des actions et états précédents. On dit alors que le processus satisfait la propriété de Markov.

Quand le processus passe de l'état \mathbf{s} à l'état \mathbf{s}' avec l'action \mathbf{a} , l'agent gagne une récompense \mathbf{r} .

2 Q-Learning

Le Q-Learning, basé sur le processus de décision markovien, est une technique d'apprentissage automatique utilisée en intelligence artificielle, plus particulièrement en apprentissage par renforcement.

2.1 Q-Function

La Q-Function est la base du Q-Learning. Pour un état \mathbf{s} et une action \mathbf{a} , elle donne la récompense esperée $\mathbf{Q}(\mathbf{s}, \mathbf{a})$.

2.2 Policy

La Politique π à l'état \mathbf{s} est la façon de choisir l'action \mathbf{a} qui maximise la récompense esperée $\mathbf{Q}(\mathbf{s}, \mathbf{a})$. C'est à dire, si on est à l'état \mathbf{s} et qu'on choisi l'action \mathbf{a} selon la politique π , alors, on pourrai espérer une récompense optimale.

$$\pi(s) = \operatorname{argmax}_a(Q(s,a))$$

2.3 Rewards

La récompense esperée \mathbf{R}_t à l'instant t pour un état \mathbf{s}_t et une action \mathbf{a}_t est la somme des récompenses futures.

$$R_t = Q(s_t, a_t)$$

$$R_t = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \dots + r_n$$

Mais, sachant que le processus est stochastique, alors plus on va dans le future, moins les récompenses sont évidentes, ainsi, on introduit un **facteur de discontinuite** γ .

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots + \gamma^{n-t} r_n$$

Ainsi,

$$\begin{aligned} R_t &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots + \gamma^{n-t} r_n \\ R_t &= r_t + \gamma(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots + \gamma^{n-(t+1)} r_n) \\ R_t &= r_t + \gamma R_{t+1} \end{aligned} \quad (1)$$

2.4 Optimal Reward

La recompense esperée à l'instant \mathbf{t} et à l'état \mathbf{s}_t , en choisissant l'action \mathbf{a}_t suivant la politique π est donc optimale et a la valeur $Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$.

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = r_t + \gamma \max_{a_{t+1}} Q(\mathbf{s}_{t+1}, a_{t+1})$$

On peut aussi écrire:

$$Q(\mathbf{s}_t, \mathbf{a}_t) = r_t + \gamma \max_{a_{t+1}} Q(\mathbf{s}_{t+1}, a_{t+1}), \quad \mathbf{a}_t = \pi(\mathbf{s}_t)$$

Note,

$$\begin{aligned} Q^\pi(\mathbf{s}_t, \mathbf{a}_t) &= \max R_t \\ &= \max(r_t + \gamma R_{t+1}) \\ &= r_t + \gamma \max(R_{t+1}) \\ Q^\pi(\mathbf{s}_t, \mathbf{a}_t) &= r_t + \gamma \max_{a_{t+1}} Q(\mathbf{s}_{t+1}, a_{t+1}) \end{aligned} \quad (2)$$

3 Q-Function

Sachant que, pour un état \mathbf{s} et une action \mathbf{a} , la q-fonction donne la récompense esperée $Q(\mathbf{s}, \mathbf{a})$, alors comment est construite cette fonction?

3.1 Q-Table

Pour un nombre fini, assez petit, d'états et d'actions, la q-fonction peut être représentée par un tableau ou une matrice, dont les états forment les lignes, et les actions forment les colonnes.

Chaque cellule du tableau est initialisée aléatoirement. Puis, mis à jour à chaque étape \mathbf{t} du processus selon la formule ci-dessous, où α est appelée **facteur d'apprentissage**.

$$Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q(\mathbf{s}_t, \mathbf{a}_t) + \alpha(r_t + \gamma \max_{a_{t+1}} Q(\mathbf{s}_{t+1}, a_{t+1}) - Q(\mathbf{s}_t, \mathbf{a}_t))$$

Selon ce formule, noter bien qu'à l'état \mathbf{s}_t , pour tous les actions possibles, il y a une qui, étant choisie, donne une récompense qui tend vers la récompense optimale. On construit donc en même temps la politique π à l'état \mathbf{s}_t .

3.2 Deep Q-Function

La plus part du temps, le nombre d'états et d'actions est assez conséquent tel qu'utiliser la q-table demande trop de ressources et n'est plus optimal.

Dans ce cas, un réseau de neurones artificiels sera un moyen plus efficace de représenter la q-fonction. Ce réseau demandera un couple état-action en entrée, et prédira la valeur de la récompense espérée en sortie.

Chaque poids des neurones du réseau sera initialisé aléatoirement et, comme dit précédemment, les targets sont les récompenses espérées. Et à l'étape t du processus, le **loss** sera alors défini comme suit:

$$loss = \left(\underbrace{r_t + \gamma \max_{a_{t+1}} Q'(s_{t+1}, a_{t+1})}_{target} - \underbrace{Q(s_t, a_t)}_{prediction} \right)^2$$

Un variant de ce réseau est un réseau qui attendra un état en entrée, et prédira en sortie un vecteur où chaque composante représente la récompense espérée d'une action. L'action optimale sera alors associée à la composante maximale.