Q-Learning

Processus d'Apprentissage basé sur un MDP

Cet article necessite des base en Apprentissage Automatique, et en Reseau de Neurones Artificiels

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 **s** et l'agent choisit une action **a**. La probabilité que le processus arrive à l'état **s'** est déterminée par l'action choisie. Plus précisément, elle est décrite par la fonction de transition d'états **T(s,a,s')**. Donc, l'état **s'** dépend de l'état actuel **s** et de l'action **a** sélectionnée par le décideur. Cependant, pour un état **s** et une action **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 **s** à l'état **s'** avec l'action **a**, l'agent gagne une récompense **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 etat **s** et une action **a**, elle donne la recompense esperée **Q(s,a)**.

2.2 Policy

La Politique $\boldsymbol{\pi}$ à l'etat **s** est la facon de choisir l'action **a** qui maximise la recompense esperée **Q(s,a)**. C'est a dire, si on est à l'etat **s** et qu'on choisi l'action **a** selon la politique $\boldsymbol{\pi}$, alors, on aura une recompense optimale.

$$\pi(s) = argmax_a(Q(s,a))$$

2.3 Rewards

La recompense esperée $\boldsymbol{R_t}$ à l'instant **t** pour un etat $\boldsymbol{s_t}$ et une action $\boldsymbol{a_t}$ est la somme des recompenses futures.

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

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

Mais, sachant que le processus est stochastique, alors plus on va dans le future, moins les recompenses sont evidentes, ainsi, on introduit un **facteur de discontinuite $\boldsymbol{\gamma}$**.

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

Ainsi,

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

2.4 Optimal Reward

La recompense à l'instant **t** et à l'etat $\boldsymbol{s_t}$, en choidissant l'action $\boldsymbol{a_t}$ suivant la polique $\boldsymbol{\pi}$ est donc optimale et a la valeur $\boldsymbol{Q^{\pi}(s_t,a_t)}$.

$$Q^{\pi}(s_t,a_t) = r_t + \gamma max_{a_{t+1}} Q(s_{t+1},a_{t+1})$$

On peut aussi ecrire:

$$Q(s_t,a_t) = r_t + \gamma max_{a_{t+1}} Q(s_{t+1},a_{t+1}), \quad a_t = \pi(s_t)$$

Note,

$$\begin{aligned} Q^{\pi}(s_t,a_t) &= max R_t \\ &= max(r_t + \gamma R_{t+1}) \\ &= r_t + \gamma max(R_{t+1}) \\ Q^{\pi}(s_t,a_t) &= r_t + \gamma max_{a_{t+1}} Q(s_{t+1},a_{t+1}) \end{aligned} \tag{2}$$

3 Q-Function

Sachant que, pour un etat **s** et une action **a**, la q-fonction donne la recompense esperée **Q(s,a)**, alors comment est construite cette fonction?

3.1 Q-Table

Pour un nombre fini, assez petit, d'etats et d'actions, la q-function peut etre representée par un tableau ou une matrice, dont les etats forment les lignes, et les actions forment les colonnes.

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

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_t + \gamma max_{a_{t+1}} Q(s_{t+1},a_{t+1}))$$

Selon ce formule, noter bien qu'à l'etat $\boldsymbol{s_t}$, pour tous les actions possibles, il y a une qui, etant choissie, donne une recompense qui tend vers la recompense optimale. On contruit donc en même temps la politique $\boldsymbol{\pi}$ à l'etat $\boldsymbol{s_t}$.

3.2 Deep Q-Function

La plus part du temps, le nombre d'etats 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 reseaux de neurones artificiels serai un moyen plus efficace de representé la q-function. Ce resaux demanderai un couple etat-action en entrée, et predirait la valeur de la recompense esperée en sortie.

Chaque poids des neurons du reseau sera initialisé aléatoirement et, comme dit perecedement, les targets sont les recompenses esperée. Et à l'etape **t** du processus, le **loss** sera alors defini comme suit:

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

Un variant de ce reseau est un reseau qui attendrai un etat en entrée, et predirait en sorti un vecteur où chaque composante represente la recompense esperée d'une action. L'action optimal serai alors associée à la composante maximale.