TD-Learning, Q-Learning et Équations de Bellman

Haythem REHOUMA

1 TD(0)

TD(0) est la forme la plus simple de TD-Learning dans la famille TD(n) telle que présentée ici. On effectue une mise à jour par *bootstrap immédiat*, en utilisant uniquement la valeur estimée du prochain état sans tenir compte de la récompense immédiate.

$$V(S_t) \leftarrow (1-\alpha)\, V(S_t) + \alpha\, \gamma\, V(S_{t+1}) \tag{1}$$

où :
— $V(S_t)$ est la valeur estimée de l'état courant S_t,
— α est le taux d'apprentissage (*learning rate*),
— γ est le facteur d'actualisation,
— $V(S_{t+1})$ est la valeur estimée de l'état futur immédiat.

2 TD(1)

TD(1) est la variante la plus utilisée : on utilise la récompense immédiate puis on effectue le bootstrap sur la valeur du prochain état.

$$V(S_t) \leftarrow (1-\alpha)\, V(S_t) + \alpha\big[R_{t+1} + \gamma V(S_{t+1})\big] \tag{2}$$

Ici, on prend en compte :
— la récompense immédiate R_{t+1},
— puis la valeur actualisée du prochain état $V(S_{t+1})$.

3 TD(2)

TD(2) utilise deux récompenses futures, puis la valeur estimée de l'état situé deux pas plus loin.

$$V(S_t) \leftarrow (1-\alpha)\, V(S_t) + \alpha\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})\big] \tag{3}$$

4 TD(3)

De manière analogue, TD(3) utilise trois récompenses futures avant de faire le bootstrap sur $V(S_{t+3})$.

$$V(S_t) \leftarrow (1-\alpha)\, V(S_t) + \alpha\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 V(S_{t+3})\big] \tag{4}$$

5 TD(4)

TD(4) utilise quatre récompenses, puis la valeur estimée de l'état S_{t+4}.

$$V(S_t) \leftarrow (1-\alpha)\, V(S_t) + \alpha\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \gamma^4 V(S_{t+4})\big] \tag{5}$$

6 TD(n) – Généralisation

La forme générale TD(n) est donnée par :

$$V(S_t) \leftarrow (1-\alpha)\, V(S_t) + \alpha \left[\sum_{k=1}^{n} \gamma^{k-1} R_{t+k} + \gamma^n V(S_{t+n})\right] \tag{6}$$

Dans ces équations :
— R_{t+k} est la récompense reçue k pas dans le futur,
— γ^{k-1} pondère chaque récompense selon sa distance temporelle,
— $V(S_{t+n})$ est la valeur estimée de l'état n pas dans le futur.
Plus n est grand, plus la méthode exploite d'informations futures. Cela peut améliorer la précision de l'estimation, mais augmente aussi la variance et la complexité numérique.

7 Q-Learning

Q-Learning est une méthode *off-policy* qui apprend les valeurs des paires *état–action*. La mise à jour de Q-Learning est :

$$Q(S_t, A_t) \leftarrow (1-\alpha)\, Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a)\right] \tag{7}$$

où :
— $Q(S_t, A_t)$ est la valeur estimée de la paire état–action courante,
— $\max_a Q(S_{t+1}, a)$ est la meilleure valeur d'action dans l'état suivant S_{t+1}.
Q-Learning cherche ainsi à approximer la *fonction de valeur optimale* des paires état–action, indépendamment de la politique effectivement suivie pendant l'exploration.

8 Équations de Bellman

Les équations de Bellman expriment la valeur d'un état (ou d'une paire état–action) comme la récompense immédiate attendue plus la valeur actualisée des futurs états, sous une politique π donnée ou à l'optimum.

8.1 Valeur d'un état sous une politique π

$$V^\pi(s) = \mathbb{E}_\pi\big[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s\big] \tag{8}$$

8.2 Valeur d'une paire état–action sous une politique π

$$Q^\pi(s,a) = \mathbb{E}_\pi\big[R_{t+1} + \gamma Q^\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\big] \tag{9}$$

8.3 Forme optimale (état)

$$V^*(s) = \max_a \mathbb{E}\big[R_{t+1} + \gamma V^*(S_{t+1}) \mid S_t = s, A_t = a\big] \tag{10}$$

8.4 Forme optimale (état–action)

$$Q^*(s,a) = \mathbb{E}\big[R_{t+1} + \gamma \max_{a'} Q^*(S_{t+1}, a') \mid S_t = s, A_t = a\big] \tag{11}$$

9 Comparaison des méthodes

Méthode	**Type**	**Mise à jour**	**Avantages**	**Inconvénients**
TD(0)	On-policy	Bootstrap immédiat (aucune récompense utilisée)	Très simple, calcul rapide	Sous-utilise l'information des récompenses, biais élevé
TD(n)	N-step, on-policy	n récompenses + 1 valeur future $V(S_{t+n})$	Peut améliorer la précision de l'estimation de V	Variance et complexité croissantes avec n
Q-Learning	Off-policy	Mise à jour sur paires état–action, avec $\max_a Q(S_{t+1}, a)$	Permet d'apprendre une politique optimale sans suivre cette politique pendant l'exploration	Nécessite plus de mémoire et une exploration suffisante de l'espace des actions