Aide mémoire MDP

Hervé Frezza-Buet

30 avril 2004

1 Reinforcement Learning

S et A sont les espaces d'états et d'actions, $\pi(s,a)$ la probabilité de choisir l'action a dans l'état s, c'est la politique de comportement. Si on en dispose, on notera les régularités du monde $T^a_{ss'}$ et $r^a_{ss'}$ les probabilités de transition d'état et de récompense, quand on passe de s en s' par l'action a. on pose :

$$\begin{aligned} V^\pi(s) &= \mathcal{E}^\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\,\middle|\, s_t = s\right\} \\ Q^\pi(s,a) &= \mathcal{E}^\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\,\middle|\, s_t = s, a_t = a\right\} \end{aligned}$$

2 Consistance et optimalité

L'équation de Bellman définissant le jeu des valeurs V correspondant effectivement à une politique.

$$V^\pi(s) = \sum_a \pi(s,a)\sum_{s'} T^a_{ss'}[r^a_{ss'} + \gamma V^\pi(s')] \quad (1)$$

Il existe des politiques optimales qui ont toutes les mêmes V et Q, supérieurs à tous les V^π et Q^π. On a les équations d'optimalité de Bellman.

$$\begin{aligned} V^\star(s) &= \max_a Q^\star(s,a) \\ V^\star(s) &= \max_a \sum_{s'} T^a_{ss'}[r^a_{ss'} + \gamma V^\star(s')] \quad (2) \end{aligned}$$

3 Programmation dynamique

3.1 Policy evaluation

On itère une suite V_k, pour tous les s à chaque pas (full backup), dont Bellmann (1) est le point fixe.

$$V^\pi_{k+1}(s) = \sum_a \pi(s,a)\sum_{s'} T^a_{ss'}[r^a_{ss'} + \gamma V^\pi_k(s')]$$

3.2 Policy improvement

À partir des V^{π_k} d'une politique π_k, on peut en trouver une meilleure : celle qui est gloutonne sur ces valeurs. La meilleure de toutes est gloutonne sur ces propres valeurs (point fixe du processus d'amélioration, et de l'équation d'optimalité de Bellman (2)).

$$\begin{aligned} \pi^{k+1}(s) &= \operatorname*{argmax}_a \sum_{s'} T^a_{ss'}[r^a_{ss'} + \gamma V^{\pi_k}(s')] \\ V^{\pi_{k+1}}(s') &= \max_a \sum_{s'} T^a_{ss'}[r^a_{ss'} + \gamma V^{\pi_k}(s')] \end{aligned}$$

3.3 Value iteration

On mélange évaluation et amélioration. On peut faire des mises à jour synchrones *et asynchrones*.

$$V_{k+1}(s) = \max_a \sum_{s'} T^a_{ss'}[r^a_{ss'} + \gamma V_k(s')]$$

4 Méthodes de Monte Carlo

On estime les V et Q à partir de plusieurs tests[1], parce qu'on *ne peut plus les calculer* à partir des $T^a_{ss'}$ et $r^a_{ss'}$, ceux-ci étant inconnus.
On applique une politique π pour générer un épisode jusqu'à l'état terminal. Après la première visite de s, on mesure la somme pondérée[2] des récompenses reçues : le profit mesuré. On met à jour V^π où Q^π à partir de ce profit mesuré pour en estimer la moyenne. Ayant confiance en ces estimations, on peut améliorer π en définissant une politique gloutonne[3] sur Q^π.
Le problème est de s'assurer de visiter tous les états/actions. On peut le résoudre avec des politiques ϵ-soft et ϵ-gloutonnes.

[1] épisodes
[2] par les γ^k.
[3] Sans les $T^a_{ss'}$, les V^π ne sont plus utilisables !

5 Différences temporelles

Avec Monte Carlo, on estime la moyenne des profits R_t, mais il faut attendre la fin de l'épisode pour avoir R_t :

$$\Delta V^\pi(s_t) = \alpha\big[\underline{R_t} - V^\pi(s_t)\big]$$

5.1 TD-learning

On fait pareil, mais on utilise un estimateur pour ne pas avoir à attendre la fin de l'épisode.

$$\Delta V^\pi(s_t) = \alpha\Big[\underline{r_{t+1} + \gamma V^\pi(s_{t+1})} - V^\pi(s_t)\Big]$$

5.2 Sarsa

C'est ma même chose sur les Q.

$$\Delta Q^\pi(s_t, a_t) = \alpha\Big[\underline{r_{t+1} + \gamma Q^\pi(s_{t+1}, a_{t+1})} - Q^\pi(s_t, a_t)\Big]$$

5.3 Q-Learning

C'est Off-Policy, on évalue directement la politique optimale alors qu'on suit une autre politique.

$$\Delta Q^\star(s_t, a_t) = \alpha\Big[\underline{r_{t+1} + \gamma \max_a Q^\star(s_{t+1}, a)} - Q^\star(s_t, a_t)\Big]$$

6 Traces

6.1 TD(λ), vue en avant, théorique

On peut utiliser une autre <u>cible</u>.

$$R_t^n = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n})$$

C'est une vue *en avant* de la qualité de la politique. Ça décrit de TD ($n = 1$) jusqu'à Monte Carlo ($n = \infty$). On peut même utiliser comme <u>cible</u> des combinaisions linéaires de R_t^n.

$$R_t^\lambda = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^n$$

Là aussi, on passe de TD ($\lambda = 0$) à Monte Carlo ($\lambda = 1$).

6.2 TD(λ), vue en arrière, pratique

En pratique, on fait un calcul basé sur des traces pour réaliser un $TD(\lambda)$. C'est la section précédente qui permet de donner un sens à ce que calculent ces traces, car on montre que les calculs avant et arrière sont équivalents. A tout moment t, pour tous les états s, on tient à jour une trace d'éligibilité $e_t(s)$:

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{si } s \neq s_t \\ \gamma\lambda e_{t-1}(s) + 1 & \text{si } s = s_t \end{cases}$$

On définit alors la mise à jour suivante, *pour tous les états à chaque pas de temps*, à l'aide des traces :

$$\Delta V_t(s) = \alpha e_t(s)\Big[\underline{r_{t+1} + \gamma V_t(s_{t+1})} - V_t(s)\Big]$$