AS - Apprentissage par renforcement (simple)

Ludovic Denoyer et Nicolas Baskiotis

Notions de base

Processus de Décision Markovien partiellement observables (PO-MDP)

- Un ensemble d'états $\mathcal{S}$ associé à un espace d'observations $\mathcal{X}$ tel que $P(x|s)$ est inconnu
  - Si $\mathcal{X} == \mathcal{S}$, alors, on a affaire à un MDP classique
- Un ensemble possible d'actions $\mathcal{A}$, discret dans notre cas
- Une fonction de récompense immédiate $r(s, a)$
- Un modèle (inconnu) du monde $P(s'|s, a)$

Politique

Dans ce contexte, un agent est défini par une **politique** π telle que $\pi = P(a|s)$, la probabilité de choisir une action dans un état donné. Dans le cadre des PO-MDP, la politique est défine par $P(a|x_t, ..., x_1)$ où $x_t, a_{t-1}, x_{t-1}, ..., x_1$ est la séquence des observations+actions précédentes.

Politique optimale

Soit un facteur de *discount* $\gamma \leq 1$, la récompense globale obtenue sur une trajectoire $s_1, a_1, s_2, a_2, ...a_{T-1}, s_T$ est définie par :

$$R(s_1, a_1, s_2, a_2, ...a_{T-1}, s_T) = \sum_t \gamma^{t-1} r(s_t, a_t)$$

La politique optimale π^* est la politique qui maximise l'espérance de la récompense :

$$\pi^* = \arg\max_\pi E_\pi[R(s_1, a_1, ..., s_{+\infty})]$$

où $s_1, a_1, ..., s_{+\infty}$ est échantilloné selon π. Il est prouvé que cettte politique existe.

Value Function et Q-Function

La *Value Function* est définie comme :

$$V^\pi(s_t) = E_\pi[R(s_t, a_t, ..., s_{+\infty})]$$

La *Q-function* est définie comme :

$$Q^\pi(s_t, a_t) = E_\pi[R(s_t, a_t, ..., s_{+\infty})]$$

La *Value function* et la *Q function* entretiennent des liens étroits :

$$Q^\pi(s_t, a_t) = \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t)(r(s_t, a_t) + \gamma V^\pi(s_{t+1}))$$

Q-Learning

Soit Q^* la Q-value optimale :

$$Q^*(s, a) = \max_\pi Q^\pi(s, a)$$

On peut définir la politique optimale comme :

$$\pi^*(s) = \max_a Q^*(s, a)$$

On a donc :

$$Q^*(s, a) = r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})$$

Et on peut montrer que :

$$Q^*(s, a) = E_{s'}[r + \gamma \max_{a'} Q^*(s', a')|s, a]$$

D'où l'algorithme de Q-learning de la Figure 1

Q-Learning approché

On va maintenant considérer que $Q^*(s, a)$ est approximé par un modèle paramétrique $Q(s, a, w)$ de paramètres w (voir Figure 2)

On peut utiliser la descente de gradient pour résoudre l'équation de Bellman :

– En considérant que $r + \gamma \max_{a'} Q^*(s', a', w)$ est la cible a atteindre pour Q(s,a,w)

– En utilisant un coût des moindres carrés : $(r + \gamma \max_{a'} Q^*(s', a', w) - Q(s, a, w))^2$

Mais, en pratique, cela diverge due aux corrélations dans les exemples d'apprentissage. La solution passe par l'implémentation d'une mémoire (*Experience Replay*) permettant de supprimer les corrélations (et la non stationnarité) dans la base d'apprentissage.

Algorithme 6 L'algorithme *Q-Learning.*

$Q(s,a) \leftarrow 0, \forall (s,a) \in (\mathcal{S}, \mathcal{A})$
pour ∞ **faire**
  initialiser l'état initial s_0
  $t \leftarrow 0$
  répéter
    choisir l'action à émettre a_t et l'émettre
    observer r_t et s_{t+1}
    $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_t + \gamma \max_{a' \in \mathcal{A}(s_{t+1})} Q(s_{t+1},a') - Q(s_t,a_t)]$
    $t \leftarrow t+1$
  jusque $s_t \in \mathcal{F}$
fin pour

Figure 1 – Q-Learning (tabulaire)

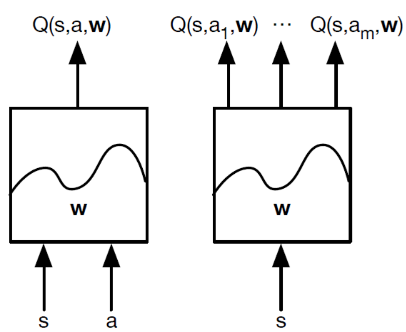

Figure 2 – Approximation de la Q-function

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ **do**
        With probability ϵ select a random action a_t
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for

FIGURE 3 – Deep Q-Learning with Experience Replay