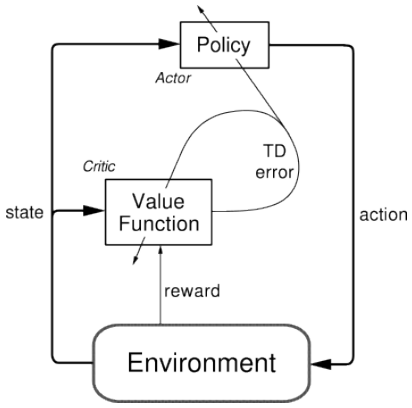
Soft Actor-Critic:
Off-Policy Maximum Entropy Deep Reinforcement

Oliver Chmurzynski, Leon Büttinghaus, Thilo Röthemeyer

Sommersemester 2021

1 Actor-Critic Ansatz

Der Actor nimmt die Aktionsauswahl für einen bestimmten Zustand vor, der Critic lernt die Q-function und "kritisiert" die Entscheidungen des Actors.



2 Entropie Maximierung

Während standard RL Algorithmen nur die Belohnung maximieren, maximiert Soft Actor-Critic zusätzlich die Entropie der policy. Die Entropie ist höher, je ungenauer festzustellen ist, welche Aktion die optimale ist. Dies regt somit mehr zum Erforschen der Umgebung an. Die Entropie-Maximierung wird mit dem temperature-Wert α reguliert.

$$\sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))] \tag{1}$$

Der temperature-Wert kann entweder initial gesetzt werden, oder über gradient descent gelernt werden. Zweiteres ist zu bevorzugen, da die Einstellung von α zur Laufzeit angepasst werden müsste, um sich der verändernden policy anzupassen. Das Lernverfahren erfolgt über folgende Fehlerfunktion:

$$J(\alpha) = \mathbb{E}_{a_t\sim\pi_t}\left[-\alpha\log\pi_t(a_t|s_t) - \alpha\mathcal{H}\right] \tag{2}$$

3 Update Regeln der Netze

Die Optimierung des Critic- und Actor-Netzwerkes erfolgt ebenfalls über gradient descent. Das Critic-Netzwerk versucht die folgende Fehlerfunktion zu minimieren:

$$J_Q(\theta) = \tag{3}$$

$$\mathbb{E}_{(s_t,a_t)\sim D}\left[\frac{1}{2}(Q_\theta(s_t,a_t) - (r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}[V_{\overline{\theta}}(s_{t+1})]))^2\right]$$

Dabei ist

$$V_{\overline{\theta}}(s_t) = \mathbb{E}_{a_t\sim\pi}[Q_{\overline{\theta}}(s_t,a_t) - \alpha\log\pi(a_t|s_t)] \tag{4}$$

der value eines gegebenen Zustandes, mit Parametern $\overline{\theta}$ des target-Netzwerkes

Die Fehlerfunktion des Actors lautet:

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim D}\left[\mathbb{E}_{a_t\sim\pi_\phi}\left[\alpha\log(\pi_\phi(a_t|s_t)) - Q_\theta(s_t,a_t)\right]\right] \tag{5}$$

4 Reparameterization Trick

Das Ziel des reparameterization tricks ist es, das stochastische Actor-Netzwerk durch Reparametrisierung in ein deterministisches umzuwandeln. Die geschieht über eine Abbildung auf den Aktionsraum, welcher über einen noise Vektor ϵ die Aktionen sampelt. Dadurch wird die Gradientenberechnung und somit das Training erleichtert.

$$a_t = f_\phi(\epsilon_t; s_t) \tag{6}$$

Die Fehlerfunktion des Actors kann dann wie folgt umgeschrieben werden:

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim D,\epsilon_t\sim N}\left[\alpha\log\pi_\phi(\phi_\phi(\epsilon_t;s_t)|s_t) - Q_\theta(s_t,\phi_\phi(\epsilon_t;s_t))\right] \tag{7}$$

5 Algorithmus

Algorithm 1: Soft Actor-Critic

Input θ_1,θ_2 ,ϕ
$\overline{\theta_1} \leftarrow \theta_1$, $\overline{\theta_2} \leftarrow \theta_2$, $\mathcal{D} \leftarrow \emptyset$
for *each iteration* **do**
  for *each environment step* **do**
    $a_t \sim \pi_\phi(a_t|s_t)$, $s_{t+1} \sim p(s_{t+1}|s_t,a_t)$
    $D \leftarrow D \cup \{(s_t,a_t,r(s_t,a_t),s_{t+1})\}$
  end
  for *each gradient step* **do**
    $\theta_i \leftarrow \theta_i - \lambda_Q\hat{\nabla}_{\theta_i}J_Q(\theta_i)$ for $i \in \{1,2\}$
    $\phi \leftarrow \phi - \lambda_\pi\hat{\nabla}_\phi J_\pi(\phi)$
    $\alpha \leftarrow \alpha - \lambda\hat{\nabla}_\alpha J(\alpha)$
    $\overline{\theta_i} \leftarrow \tau\theta_i + (1-\tau)\overline{\theta_i}$ for $i \in \{1,2\}$
  end
end