Abstract

1 BACKGROUND

A reinforcement learning environement is modelled as a Markov decision process (MDP) $M = \langle \mathcal{S}, \mathcal{A}, r, P, P_0, \gamma \rangle$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the set of available actions. At time $t = 0$ a state s_0 is sampled from the distribution $P_0(\cdot)$. At each timestep an action $a_t \sim \pi(\cdot|s_t)$ is selected and the agent transitions to a new state state $s_{t+1} \sim P(\cdot|s_t, a_t)$. A scalar reward $r_{t+1} = r(\cdot|s_{t+1}, s_t, a_t)$ is observed. As the agent and environement interact in a sequence of time steps, a history of observations $\mathcal{H}_t = (s_0, a_0, s_1, r_1, a_1, s_2, r_2, \ldots, s_t, r_t)$ is collected. The goal is to find a policy $\pi^\star$, such that sampling actions $a \sim \pi^*(\cdot|s)$ maximizes the expected future reward, $J^\pi := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. An efficient agent must be able to learn from the data it collects, but since the data is dependent on the policy, it must also prioritize to explore states and actions that the agent can learn a lot from.

The Q-function, $Q^\pi(s, a)$, is defined as the expected reward of taking action a in state s and then following policy π thereafter: $Q^\pi := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = 0, a_0 = 0\right]$. The Bellman operator on Q^π is defined as $\mathcal{B}[Q^\pi](s_t, a_t) := \mathbb{E}_{P(s_{t+1}|s_t,a_t)\pi(a_{t+1}|s+t)} \left[r(s_{t+1}, s_t, a) + \gamma Q^\pi(s_{t+1}, a_{t+1})\right]$. With the Q-function we can define a policy $\pi(a|s) = \arg\max_a Q(s, a)$. In deep RL, we model Q with a neural network $\hat{Q}_\omega$. One way to facilitate exploration in this policy is to introduce uncertainty in the Q-value. Fellows et al. [2021] does this by defining the Bayesian Bellman operator (BBO). They define $P_B(b|s, a, \omega)$ as the distribution over Bellman functions such that for a noisy sample $b_i \sim P_B(b|s, a, \omega)$, $b_i = \mathcal{B}[\hat{Q}_\omega](s_i, a_i) + \eta_i$. Then we approximate this with a distribution parameterized by ϕ such that $P(b|s, a, \phi) \approx P(b|s, a, \omega)$.

$$\mathcal{B}^\star_{\omega,N}(s, a) := \mathbb{E}_{P(\phi|\mathcal{D}^N_\omega)} \left[\hat{B}_\phi(s, a)\right]. \tag{1}$$

Here, $\hat{B}_\phi(s, a) = \mathbb{E}_{P(b|s,a,\phi)}[b]$, is the conditional mean of $P(b|s, a, \phi)$, and the data $\mathcal{D}^N_\omega$ is defined as the collection of sampled Q-values, states and actions $\mathcal{D}^N_\omega := \{b_i, s_i, a_i\}_{i=1:N}$. Under the assumption that each state s_i is drawn i.i.d from a distribution with support over $\mathcal{S}$, or from an ergodic Markov chain defined over a σ-algebra that is countably generated from $\mathcal{S}$ they find that $\omega^\star$ such that $\hat{Q}_{\omega^\star} = \mathcal{B}^\star_{\omega^\star,N}$ can be found by minimising the mean squared Bayesian Bellman error (MSBBE):

$$\mathrm{MSBBE}_N(\omega) := \|\hat{Q}_\omega - \mathcal{B}^\star_{\omega,N}\|^2_{\rho,\pi}, \tag{2}$$

And that an unbiased estimate of $\nabla_\omega \mathrm{MSBBE}_N(\omega)$ can be calculated as long as we can sample from $P(\phi|\mathcal{D}^N_\omega)$.

Nonlinear function approximators such as neural networks typically have an intractable posterior, so we cannot calculate the posterior analytically. Instead we use a posterior approximation $q(\phi|\mathcal{D}^N_\omega) \approx P(\phi|\mathcal{D}^N_\omega)$, together with an algorithm for approximate inference. Fellows et al. [2021] presents the Bayesian Bellman Actor-Critic using randomized priors [Osband et al., 2018], and show state of the art exploratory behaviour.

For most approximate inference methods in deep learning, it is difficult to incorporate domain knowledge into the prior. Typically the prior distribution is defined on the parameters as $P(\phi)$, while the knowledge exists in the function space $P(a|s)$.

Functional variational Bayesian neural networks [Sun et al., 2019] lets us define a prior on the *function space*. In a Q-learning algorithm, this would be a function $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. For now we generalize, and say that $f : \mathcal{X} \to \mathcal{Y}$. Sun et al. [2019] show that for two stochastic processes P and Q:

$$\mathrm{KL}[P\|Q] = \sup_{n\in\mathbb{N}, \mathbf{X}\in\mathcal{X}^n} \mathrm{KL}\left[P_\mathbf{X}\|Q_\mathbf{X}\right], \tag{3}$$

where $\mathbf{X} \in \mathcal{X}^n$ is a finite measurement set and $P_\mathbf{X}, Q_\mathbf{X}$ are the marginal distributions at $\mathbf{X}$. They further show that if $\mathbf{f}^\mathbf{X}$ are the function values at points $\mathbf{X}$, then

$$\nabla_\phi \mathrm{KL}[q_\phi(\mathbf{f}^X) \| p(\mathbf{f}^{\mathbf{X}})] = \mathbb{E}\left[\nabla_\phi \mathbf{f}^{\mathbf{X}} (\nabla_{\mathbf{f}} \log q(\mathbf{f}^{\mathbf{X}}) - \nabla_{\mathbf{f}} \log p(\mathbf{f}^{\mathbf{X}}))\right]. \quad (4)$$

The difficult part in (4) is to estimate $\nabla_{\mathbf{f}} \log q(\mathbf{f}^{\mathbf{X}})$ and $\nabla_{\mathbf{f}} \log p(\mathbf{f}^{\mathbf{X}})$. To estimate these log-density gradients, they use a spectral Stein gradient estimator [Shi et al., 2018]. Given enough function value samples, the spectral Stein gradient estimator can estimate score functions for both in-distribution and out-of-distribution samples. This means that we can use it to estimate both gradients. $\nabla_{\mathbf{f}} \log q(\mathbf{f}^{\mathbf{X}})$ is likely intractable intractable, considering q_ϕ is a neural network with stochastic weights. Depending on how we define the prior, however, $\nabla_{\mathbf{f}} \log p(\mathbf{f}^{\mathbf{X}})$ can be easy to compute analytically. To reduce variance in the gradients we aim to define tractable priors.

2 METHOD

Instead of a prior on the parameter space, we would like to have a prior on the Q-function space. Using the functional variational neural network framework explored earlier, our maximization target becomes:

$$\log p(\mathcal{D}_\omega^N | f) - \mathrm{KL}\left[q(\mathbf{f}^{\mathcal{D}_\omega^N}, \mathbf{f}^M) \| p(\mathbf{f}^{\mathcal{D}_\omega^N}, \mathbf{f}^M)\right], \quad (5)$$

where $\mathbf{f}^{\mathcal{D}_\omega^N}$ is f applied to the dataset $\mathcal{D}_\omega^N$, and $\mathbf{f}^M$ is f applied to a set of i.i.d. random points $M = \{m_i \sim c \mid i = 1, \ldots, k\} \subseteq \mathcal{S} \times \mathcal{A}$ with full support. We will let

$$M = \{(s_i, a_j)\ \forall s_i \in \mathcal{D}_\omega^N,\ \forall a_j \in \mathcal{A}\}. \quad (6)$$

To justify this we need to show that $supp(c) = \mathcal{S} \times \mathcal{A}$. Under the same assumption about the distribution of each state s_i from BBO, $supp(c) = \mathcal{S} \times \mathcal{A}$ follows trivially.

Since we are modeling q_ϕ with a Bayesian neural network, and Q_ω with a neural network, we have a bilevel optimization problem. To solve this, we employ a similar strategy to Fellows et al. [2021], where we have a two-timescale gradient update. ϕ and ω are updated using stochastic gradient descent at different timescales to ensure stable convergence.

Algorithm 1 Functional Bayesian RL

$\mathcal{D} \leftarrow \emptyset$
$s \sim P_0$
Initialize ϕ, ω
while not converged **do**
  $f \sim q_\phi$
  $a \sim \arg\max_a f(s, a)$
  $s' \sim P(\cdot|s, a)$
  $r = r(s', a, s)$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{s, a, r, s'\}$
  UPDATEPOSTERIOR($\phi, \omega, \mathcal{D}$)
end while

function UPDATEPOSTERIOR($\phi, \omega, \mathcal{D}$)
  while not converged **do**
    $T \sim \mathcal{D}$
    $\mathbf{f} \leftarrow \{f_i \sim q_\phi \text{ for } i = 1, \ldots, k\}$
    $\mathcal{F} \leftarrow \emptyset$
    $\Delta_{\mathcal{L}} \leftarrow 0$
    for $\{s, a, r, s'\} \in T$ **do**
      $\hat{\boldsymbol{a}} \leftarrow \arg\max_a \mathbf{f}(s, a)$
      $\mathcal{F} \leftarrow \mathcal{F} \cup \mathbf{f}(s, a)$
      $G \leftarrow r + \gamma Q_\omega(s, a)$
      $\Delta_{\mathcal{L}} \leftarrow \Delta_{\mathcal{L}} - \frac{1}{k} \frac{1}{|T|} \nabla_\phi \log p(\hat{\boldsymbol{a}}|G)$
    end for
    $\Delta_{\mathrm{KL}} \leftarrow SSGE(p, \mathcal{F})$
    $\phi \leftarrow \phi + \eta_\phi(\Delta_{\mathcal{L}} - \Delta_{\mathrm{KL}})$
    $\hat{B}_\phi \leftarrow \frac{1}{k} \frac{1}{|T|} \sum_{i=1}^k \sum_{b \in \mathcal{F}} b$
    $\Delta_\omega \leftarrow \sum_{\{s,a,r,s'\} \in T} \nabla_\omega \| \hat{B}_\phi(s, a) - Q_\omega(s, a) \|_2^2$
    $\omega \leftarrow \omega - \eta_\omega \Delta_\omega$
  end while
end function

References

Matthew Fellows, Kristian Hartikainen, and Shimon Whiteson. Bayesian Bellman Operators. *arXiv:2106.05012 [cs]*, June 2021. URL http://arxiv.org/abs/2106.05012. arXiv: 2106.05012.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized Prior Functions for Deep Reinforcement Learning. *arXiv:1806.03335 [cs, stat]*, November 2018. URL

http://arxiv.org/abs/1806.03335. arXiv: 1806.03335.

Jiaxin Shi, Shengyang Sun, and Jun Zhu. A spectral approach to gradient estimation for implicit distributions. jun 2018. URL https://arxiv.org/abs/1806.02925v1.

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional Variational Bayesian Neural Networks. *arXiv:1903.05779 [cs, stat]*, March 2019. URL http://arxiv.org/abs/1903.05779. arXiv: 1903.05779.