
CHAPTER 1

DEEP REINFORCEMENT LEARNING

1 Neural Networks

...

2 Distributional Reinforcement Learning

Remember that we defined $\mathbb{P}_{s,a}^\pi := \mathbb{P}^\pi \otimes \delta_{S_0}(s) \otimes \delta_{A_0}(a)$ as the probability measure of the Markov reward process (S, A, R) started in (s, a) . We define the random variable of the return under policy π as

$$Z^\pi := \sum_{t=0}^{\infty} \gamma^t R_t, \quad \gamma \in (0, 1).$$

Unlike the methods before, where we were interested in the expected reward $Q^\pi(s, a) = \mathbb{E}_{s,a}^\pi[Z^\pi]$, we are now interested in the distribution of these cumulative rewards. For that define

$$\eta_{s,a}^\pi(B) := \mathbb{P}_{s,a}^\pi(Z^\pi \in B), \quad B \in \mathcal{B}(\mathbb{R}).$$

In analogy to weak solutions for PDEs, i.e. in sense of distribution, the return distribution $\eta_{s,a}^\pi$ satisfies the Bellman equation in sense of distribution:

$$\forall \phi \in C_b(\mathbb{R}) : \quad \int_{\mathbb{R}} \phi(z) d\eta_{s,a}^\pi(z) = \mathbb{E}_{s,a}^\pi \left[\int_{\mathbb{R}} \phi(R + \gamma z) d\eta_{S',A'}^\pi(z) \right].$$

We define $f_{r,\gamma}(z) := r + \gamma z$ and the push forward

$$((\eta_{s,a}^\pi)_{f_{r,\gamma}})(B) := \eta_{s,a}^\pi(f_{r,\gamma}^{-1}(B)), \quad B \in \mathcal{B}(\mathbb{R}).$$

then, the above can be written as

$$\begin{aligned} \forall \phi \in C_b(\mathbb{R}) : \quad & \int_{\mathbb{R}} \phi(z) d\eta_{s,a}^\pi(z) = \mathbb{E}_{s,a}^\pi \left[\int_{\mathbb{R}} \phi(R + \gamma z) d\eta_{S',A'}^\pi(z) \right] = \mathbb{E}_{s,a}^\pi \left[\int_{\mathbb{R}} \phi(z) d(\eta_{S',A'})_{f_{R,\gamma}}(z) \right] \\ \iff \forall \phi \in C_b(\mathbb{R}) : \quad & \langle \phi, \eta_{s,a}^\pi \rangle = \mathbb{E}_{s,a}^\pi [\langle \phi, (\eta_{S',A'})_{f_{R,\gamma}} \rangle]. \end{aligned}$$

For any $\phi \in C_b(\mathbb{R})$ and any measure ν we have

$$\begin{aligned} (\nu * \delta_r)(\phi) &= \int_{\mathbb{R}} \phi(z) d(\nu * \delta_r)(z) = \int_{\mathbb{R}} \int_{\mathbb{R}} \phi(x + y) d\nu(x) d\delta_r(y) \\ &= \int_{\mathbb{R}} \phi(x + r) d\nu(x) = (\nu(\cdot - r))(\phi). \end{aligned}$$

Next, define $D_\gamma(x) := \gamma x$, then if $Z \sim \eta^\pi(x, a)$ then $\gamma Z \sim (\eta^\pi(s, a))_{D_\gamma}$. In total

$$Z\gamma + r \sim (\eta^\pi(s, a))_{D_\gamma}(\cdot - r) = ((\eta^\pi(s, a))_{D_\gamma} * \delta_r)$$

holds in sense of distribution. Now define the distributional Bellman operator in sense of distribution as

$$\mathcal{T}^\pi Z^\pi(s, a) = R(s, a) + \gamma Z^\pi(S', A'),$$

i.e. the distribution

$$\mathcal{T}^\pi : (\mathcal{P}(\mathbb{R}))^{\mathcal{S} \times \mathcal{A}} \rightarrow (\mathcal{P}(\mathbb{R}))^{\mathcal{S} \times \mathcal{A}}, (\nu_{s,a})_{(s,a) \in \mathcal{S} \times \mathcal{A}} \mapsto ((\mathcal{T}^\pi \nu)_{s,a})_{(s,a) \in \mathcal{S} \times \mathcal{A}}$$

is point wise defined for all $\phi \in C_b(\mathbb{R})$ as

$$(\mathcal{T}^\pi \nu)_{s,a}(\phi) := \mathbb{E}_{s,a}^\pi[((\nu(S', A'))_{D_\gamma} * \delta_R)(\phi)] = \sum_{s', a', r} \pi(a'; s') p(s', r; s, a) ((\nu_{s', a'})_{D_\gamma} * \delta_r)(\phi)$$

3 Deep Q-Networks

Deep Q-Networks (DQN) is a Q-learning variant that utilizes deep neural networks to approximate the Q-value. This algorithm addresses how to deal with large, continuous state action spaces $(\mathcal{S}, \mathcal{A})$. The idea is to reduce dimensionality by plugging in the state space \mathcal{S} (e.g. pixels in an Atari game) into a convolutional neural layer, which is fed into a feed forward neural network that outputs the Q-value for each action $a \in \mathcal{A}$. Problems with simple neural networks are

- They can forget
- Are unstable: small changes in Q-values can lead to big changes in the policy, i.e. the action distribution
- Correlation: In a trajectory τ_i of a rollout, the consecutive states s_t, s_{t+1} are highly correlated, which violates the i.i.d. assumption for estimating the expectation.
- The loss is non-stationary, since the target values depend on the parameters of the network itself, meaning

$$\mathcal{L}(\theta) := \mathbb{E}[(\mathbb{E}[R + \gamma \max_{a'} Q^\theta(S', a')] - Q^\theta(S, A))^2],$$

where both Q terms depend on θ .

In DQN these issues were solved by the following

- Experience Replay: A behavioral policy π^b samples Rollouts, that are stored in a buffer $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^N$. In order to break the correlation between consecutive states, mini batches are sampled uniformly from this buffer to update the Q-network.
- Target Network: A second network with parameters θ^- is introduced, which is bootstrapped, i.e. every C steps the parameters of the Q-network are copied to the target network $\theta^- \leftarrow \theta$. This delay stabilizes the training.

4 Proximal Policy Gradient

5 Advantage Estimation

5.1 Variants to PPO

6 Stable Baselines3