Deep Reinforcement Learning (Sp25)

Instructor: Dr. Mohammad Hossein Rohban

Summary of Lecture 9: Advanced Methods & Model-Based RL

Summarized By: Benyamin Naderi

- **PPO algorithm** in this algorithm , there are 2 fascinating tips to remember:
  - PPO uses importance sampling to evaluate the new policy using data collected from the old policy. and this importance sampling ratio is bounded.
  - The importance sampling ratio allows trajectories sampled from the old policy to be used in estimating the expected return under the new policy. However, to prevent excessive policy shifts, this ratio is clipped within a bounded range. This constraint ensures that actions chosen under the new policy do not lead to uncontrolled growth in advantage estimates, promoting stable and efficient policy learning.
- **PPO objective**

$$\theta' = \arg\max_{\theta'} \mathbb{E}_{s\sim\mu_\theta, a\sim\pi_\theta}\left[\min\left(\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}A^{\pi_\theta}(s,a), \text{clip}\left(\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}, 1-\epsilon, 1+\epsilon\right)A^{\pi_\theta}(s,a)\right)\right]$$

- In PPO we must fit 2 networks first for the policy network and the other one for value function , for the policy network learning goes on by backproping the policy network using PPO objective and for value function by using target values. note that the target value may be an approximate of reward-to-go and can be approximated by adding immediate reward and next state value(TD target)
- **Soft Actor Critic(SAC)** building up a robust algorithm , favoring stochastic policies for exploration , we make a change in the Objective function the way that the agent not only tries to maximize it's expected-discounted reward, but also promoting our policy to be stochastic . In other words The Soft Actor-Critic integrate entropy regularization into the value function, balancing reward maximization with policy entropy to encourage exploration while considering discounted future rewards. By maximizing the expected reward, while actually making policy choose more random actions to explore better the SAC objective can be seen below:

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

- **Soft Policy Improvement :** Bellman equation for SAC framework can be seen below, please note that value iteration and policy iteration is based on these 2 equations. Here we have soft Q-value and soft value function.

$$T^\pi Q(\mathbf{s}_t,\mathbf{a}_t) \triangleq r(\mathbf{s}_t,\mathbf{a}_t) + \gamma\mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[V(\mathbf{s}_{t+1})\right], \tag{2}$$

$$\text{where } V(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t\sim\pi}\left[Q(\mathbf{s}_t,\mathbf{a}_t) - \log\pi(\mathbf{a}_t|\mathbf{s}_t)\right]. \tag{3}$$

- Soft Policy is derived by a softmax operation on action distribution in output policy network.
- **In soft actor-critic we have a value network , Q network , policy network** each network is trained on replay buffer data and backpropagation on 3 loss functions

Deep Reinforcement Learning (Sp25)

Instructor: Dr. Mohammad Hossein Rohban

Summary of Lecture 9: Advanced Methods & Model-Based RL

Summarized By: Benyamin Naderi

$$J_V(\psi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\frac{1}{2} \left(V_\psi(s_t) - \mathbb{E}_{a_t \sim \pi_\phi} \left[Q_\theta(s_t, a_t) - \log \pi_\phi(a_t|s_t) \right] \right)^2 \right], \tag{1}$$

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\theta(s_t, a_t) - \hat{Q}(s_t, a_t) \right)^2 \right], \tag{2}$$

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[D_{\mathrm{KL}} \left(\pi_\phi(\cdot|s_t) \middle\| \frac{\exp\left(Q_\theta(s_t, \cdot)\right)}{Z_\phi(\cdot)} \right) \right]. \tag{4}$$

Networks update rules(SGD)

$$\psi \leftarrow \psi - \lambda_V \nabla_\psi J_V(\psi), \quad \theta \leftarrow \theta - \lambda_Q \nabla_\theta J_Q(\theta), \quad \phi \leftarrow \phi - \lambda_\pi \nabla_\phi J_\pi(\phi), \quad \bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$$

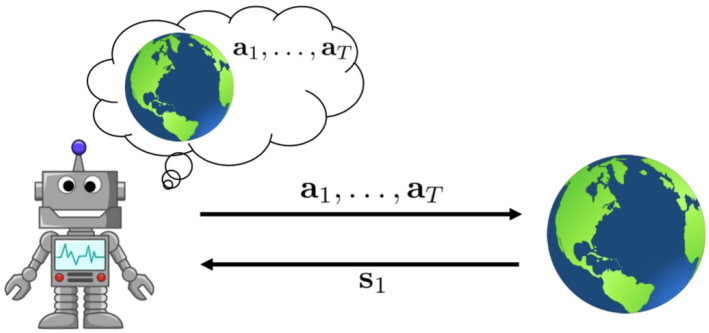
- Race condition is prevented by training a network for value function estimation and using soft bellman equation :

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p} \left[V_{\bar{\psi}}(s_{t+1}) \right] \tag{1}$$

- **Exponential moving average:** value network parameter updates slowly, since building up a stable target. **Value Network** for training this network it uses every state action in replay buffer **but for the target** we use bellman estimate of Value function and please be informed that actions taken to estimate this expectation come form **current policy** , aka the policy network action distribution.
- **Model-Based RL :** In previous methods we actually didn't try to use transition probabilities but we tried to learn the policy directly from sampling various amounts of trajectories and that's why we were completely sample inefficient , in Model-Based methods there the idea is that we can **learn transition dynamics** of our MDP, and then figure out choosing actions.
- For some cases we know transition dynamics in advance, but for some cases we can learn them.
- **In deterministic case there is an assumption that we know the transition dynamics** for this setup at first the agents captures the first state from the environment and determines the set of actions that maximizes it's reward. Note that there is no need to know internal state transitions (s2 , s3, ..) since we know the dynamics and at every moment the agent sense the reward. For more visualization over Model-Free methods take a look at this picture this case is open loop and there is no feedback to the agent since dynamics are deterministic and each state can be determined:
- **Deterministic Model-Based Optimization**

$$a_1, \ldots, a_T = \arg \max_{a_1, \ldots, a_T} \sum_{t=1}^{T} r(s_t, a_t) \quad \text{s.t.} \quad s_{t+1} = f(s_t, a_t)$$

This equation seeks a sequence of actions $(a_1, \ldots, a_T)$ that maximizes the cumulative reward $\sum_{t=1}^{T} r(s_t, a_t)$, subject to the dynamics constraint $a_{t+1} = f(s_t, a_t)$ governing action transitions.



$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg \max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} \sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \text{ s.t. } \mathbf{a}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$$

- Open loop gives us a suboptimal solution, since after choosing the first action there is a possibility that environment changes and one should choose a different sequence of actions.
- **Cross Entropy Method for stochastic planning**

  CEM can be seen as an Evolution Strategy which optimizes a stochastic function $f(\mathbf{x})$ with $f : \mathbb{R}^n \to \mathbb{R}$ by finding a suitable "individual" $\mathbf{x}$. The algorithm proceeds as follows:

  – **Population sampling**: Draw candidates $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$ from a distribution

  – **Elite selection**: Sort candidates by cost $f(\mathbf{x}^{(i)})$ and retain top-$\rho\%$ (elite set $\mathcal{E}$)

  – **Parameter update**:

$$\boldsymbol{\mu}_{\text{new}} = \frac{1}{|\mathcal{E}|} \sum_{\mathbf{x} \in \mathcal{E}} \mathbf{x}, \quad \boldsymbol{\sigma}^2_{\text{new}} = \frac{1}{|\mathcal{E}|} \sum_{\mathbf{x} \in \mathcal{E}} (\mathbf{x} - \boldsymbol{\mu}_{\text{new}})^2$$

  By iteratively refitting $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ to the elite set, the sampling distribution concentrates around low-cost regions of $\mathbb{R}^n$. After multiple iterations, CEM converges to an $\mathbf{x}^*$ near a (local/global) optimum. The method's computational cost scales with both the evaluation cost of $f(\mathbf{x})$ and the number of samples N.