Policy Gradient Methods

Federico Arangath

18 March 2024

Foundations of Reinforcement Learning

Notation

- We define $G_t := \sum_{k=0}^{\infty} \gamma^k r_{t+k}$
- We define $G_{t:T} := \sum_{k=0}^{T-t-1} \gamma^k r_{t+k}$
- We denote $J_T(\pi) := E_\pi\left[\sum_{t=0}^{T-1} \gamma^t r_t\right] = E_\pi\left[G_{0:T}\right]$
- $J(\theta) := J(\pi_\theta)$
- We denote the probability of a trajectory by

$$\Pi_\theta(\tau) = p(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t) p_{s_t \to s_{t+1}}^{a_t}$$

Recap

- **Recall**: In Q-Learning, to find the optimal (deterministic) policy, we do the following

$$\pi^*(s) = \mathrm{argmax}_{a \in A} Q^*(s, a)$$

- This is an infeasible optimization problem when the action space is very large (or continuous)!

Policy Approximation

- So how can we solve this issue?
- We approximate/parametrize the policy directly instead of estimating the corresponding value function!
- In particular, we will have

$$\pi^*(s) \approx \pi^*(s;\theta) = \pi^*_\theta(s)$$

Policy Values

- How do we optimize the parameters θ? What is our objecitve?
- **Recall**: Our original objective is to maximize the expected cumulative return, given by:

$$J(\pi) := E_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] = E_\pi\left[G_0\right]$$

Approximating Policy Values

- In practice, we have to approximate the above expectation since it is intractable
- **MC Sampling**: $J(\pi) \approx \frac{1}{n} \sum_{i=0}^{n} G_0^{(i)}$ where $G_0^{(i)}$ is the cumulative reward obtained during the i-th trajectory
- **Finite Horizon**: $J(\pi) \approx J_T(\pi)$ if T is large enough
- Hence, we will use $J(\pi) \approx \frac{1}{n} \sum_{i=0}^{n} G_{0:T}^{(i)}$ where $G_{0:T}^{(i)}$ is the cumulative reward (over T time steps) obtained during the i-th trajectory

Score Gradient Estimator I

- We now want to find the best value of θ that maximizes our objective
- To do this, we will use **gradient ascent**
- **Theorem**: It holds that

$$\nabla_\theta J_T(\theta) = \nabla_\theta E_{\tau\sim\Pi_\theta}\left[G_{0:T}\right] = E_{\tau\sim\Pi_\theta}\left[G_{0:T}\nabla_\theta \log \Pi_\theta(\tau)\right]$$

Score Gradient Estimator II

- Again, it is unfeasible to calculate the expectation
- We again use MC estimation:

$$\nabla_\theta J_T(\theta) \approx \frac{1}{n} \sum_{i=1}^{n} G_{0:T}^{(i)} \nabla_\theta \log \Pi_\theta(\tau^{(i)})$$

- There is still a problem with the estimator above. Can you see it?

Score Gradient Estimator III

- **Recall**: $\Pi_\theta(\tau) = p(s_0)\prod_{t=0}^{T-1}\pi_\theta(a_t|s_t)p_{s_t\to s_{t+1}}^{a_t}$
- However, $p_{s_t\to s_{t+1}}^{a_t}$ is unknown, hence we cannot compute $\Pi_\theta(\tau)$ directly!
- However, we can note the following:

$$\nabla_\theta \log \Pi_\theta(\tau^{(i)}) = \underbrace{\nabla_\theta \log p(s_0)}_{=0} + \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) + \sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log p_{s_t^{(i)}\to s_{t+1}^{(i)}}^{a_t^{(i)}}}_{=0}$$

Score Gradient Estimator IV

- Hence, we have that

$$\nabla_\theta J_T(\theta) \approx \frac{1}{n} \sum_{i=1}^{n} G_{0:T}^{(i)} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

- Intuition: Increasing $J(\theta)$ is equivalent to increasing the probability of policies whose corresponding return is high and decreasing probability of policies with low return

Score Gradient Estimator: Pros and Cons

- The estimators we obtained are **unbiased**
- However, as they come from a MC sampling procedure, they exhibit **High Variance**

Score Gradient Estimator with Baselines I

- To reduce variance, we introduce the concept of baselines
- **Constant Baseline**:

$$\mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\left[G_0 \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\right] = \mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\left[(G_0 - b)\boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\right]$$

- Estimator is still **unbiased**!
- It can be shown that $b \leq 2r_s^a \quad \forall s, a$, then the variance is guaranteed to be reduced

Score Gradient Estimator with Baselines II

- **Time Dependent Baselines**: Choose

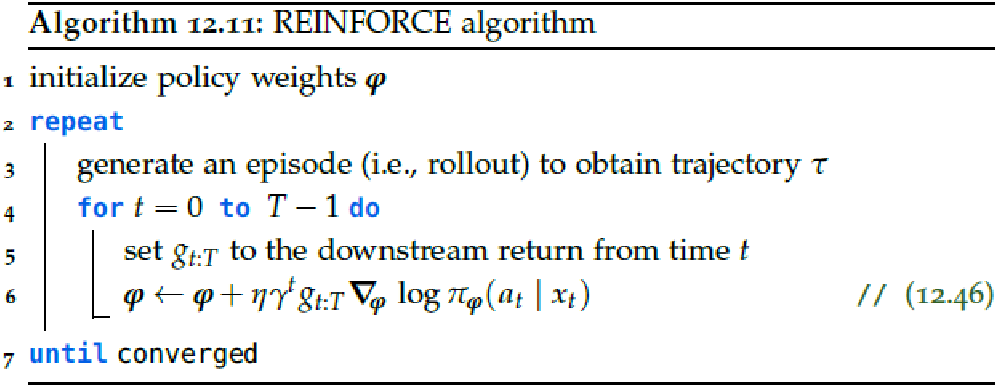
$$b(\tau_{0:t-1}) = \sum_{m=0}^{t-1} \gamma^m r_m$$

- Then, we have that:

$$G_0 - b(\tau_{0:t-1}) = \sum_{m=t}^{T-1} \gamma^m r_m = \gamma^t G_{t:T}$$

- Estimator still **Unbiased**
- This baseline helps addressing the **Credit Assignment** problem

REINFORCE



Algorithm 12.11: REINFORCE algorithm

```
initialize policy weights $\varphi$
repeat
    generate an episode (i.e., rollout) to obtain trajectory $\tau$
    for $t = 0$ to $T - 1$ do
        set $g_{t:T}$ to the downstream return from time $t$
        $\varphi \leftarrow \varphi + \eta \gamma^t g_{t:T} \nabla_\varphi \log \pi_\varphi(a_t \mid x_t)$        // (12.46)
until converged
```

REINFORCE: Remarks

- **On Policy**: to generate the estimators of the gradient, we need to do rollouts following the policy π_θ, where θ is the current estimate of the optimal parameters
- **High Variance**: even after inserting baselines, REINFORCE can still exhibit high variance. Further more advanced Variance Reduction techniques can be applied to reduce variance even more
- Due to high variance and instability, REINFORCE can often get **stuck in local optima**
- **Computationally inefficient**: due to high variance, we need to sample many trajectories to have good estimates of the gradient
- **Cannot reuse data** from previous rollouts since the method is On-policy

Actor Critic Methods

- We would now like to combine policy gradient methods with value function approximation
- This will allow us to keep the benefits of both approaches, i.e. **direct estimation of the policy** and **lower variance**
- **Note**: This comes at a cost of having to estimate the Q-function too

Policy Gradient Theorem: Idea

- The Idea is to reduce the high variance in Reinforce by using an equivalence definition of the gradient estimator involving Q-function
- This reduces variance because we won't have to sample long trajectories anymore (similar to MC vs Q-Learning we saw in Lecture 2)

Policy Gradient Theorem I

- **Theorem**:

  $$\nabla_\theta J(\theta) \propto \mathbb{E}_{s\sim\rho_\theta^\infty}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)\right]$$

  where $\rho_\theta^\infty = (1-\gamma)\sum_{t=0}^\infty \gamma^t \mathbb{P}_{\pi_\theta}(S_t = s)$ is called the **State Occupancy Measure**
- Now our gradient estimator can be approximated with trajectories of length one (only need to sample one action and one state), which has much lower variance
- Intuitively, maximizing $J(\theta)$ corresponds to **increasing the probability of policies that have high value**

Online Actor Critic I

- We will make use of two networks to estimate the Policy and the Q-function
- **Actor**: Policy Network, trained with SARSA
- **Critic**: Q-function network, trained taking samples from the gradient estimator in the previous slide

Online Actor Critic II

Algorithm 12.16: Online actor-critic

```
initialize parameters $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}$
repeat
    use $\pi_{\boldsymbol{\varphi}}$ to obtain transition $(\boldsymbol{x}, \boldsymbol{a}, r, \boldsymbol{x}')$
    $\delta = r + \gamma Q(\boldsymbol{x}', \pi_{\boldsymbol{\varphi}}(\boldsymbol{x}'); \boldsymbol{\theta}) - Q(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta})$
    // actor update
    $\boldsymbol{\varphi} \leftarrow \boldsymbol{\varphi} + \eta \gamma^t Q(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}) \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a} \mid \boldsymbol{x})$    // (12.65)
    // critic update
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \delta \boldsymbol{\nabla}_{\boldsymbol{\theta}} Q(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta})$    // (12.66)
until converged
```

Online Actor Critic III

- Since we sample the transition from the current policy estimate, the algorithm is **On Policy**
- **Note**: We neglect the dependency of the critic from the parameters of the policy to simplify training.
- This implies that **We are not guaranteed to have a valid ascent direction**

Online Actor Critic: Pros and Cons

- **Reduced variance** from REINFORCE (variance can be further decreased with methods such as A2C), although estimates are biased (due to bias of the critic)
- **Low Sample Efficiency**: like in all On Policy methods, we cannot reuse past data and hence we have to do many interactions with the environment

TRPO I

- **Goal**: we want to try to introduce Off-Policy data into the optimization to **improve sample efficiency**
- To do this, at each time step we update the policy forcing it to remain "close" to the current policy (in the so called **Trust Region**)
- In this way, subsequent policies produce similar behaviors and hence we can then reuse data from previous policies (not too far in the past)

TRPO II

Let's introduce some notation we will use in the algorithm

- **Advantage Function**:

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$$

- Surrogate Objective:

$$\tilde{J}(\theta, \theta_k) = \mathbb{E}_{s,a \sim \pi_{\theta_k}}\left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a)\right]$$

- KL Divergence:

$$KL(p||q) = \mathbb{E}_{p(x)}\left[\log \frac{p(x)}{q(x)}\right]$$

TRPO III

- At each iteration of TRPO, we solve the following optimization problem:

$$\theta_{k+1} = argmax_{\theta} \tilde{J}(\theta, \theta_k) \;\; \text{s.t.} \;\; KL(\pi_{\theta} || \pi_{\theta_k}) \leq \delta$$

- **Guarantees monotonic improvement of the original objective** $J(\theta)$, i.e $J(\theta_{k+1}) \geq J(\theta_k)$!
- Constraint on the KL-divergence ensures that two subsequent policies are similar

TRPO: Pros and Cons

- TRPO is **more sample efficient** than other Actor-Critics methods since it can partially be trained with Off-Policy data
- Constraining policy updates also provides more **stability** to the algorithm in practice
- **Downside**: Solving a constrained optimization problem at each iteration can be **computationally expensive**

PPO I

- PPO is a variant of TRPO with the same goal of introducing Off-Policy data to improve sample efficiency
- At the same time, it solves the problem of computational inefficiency by **modifying the surrogate objective getting rid of the KL divergence**
- In practice, it has shown to work really well, balancing sample and computational efficiency
- Used in RLHF to train Chat-GPT

PPO II

- The PPO surrogate objective is given by:

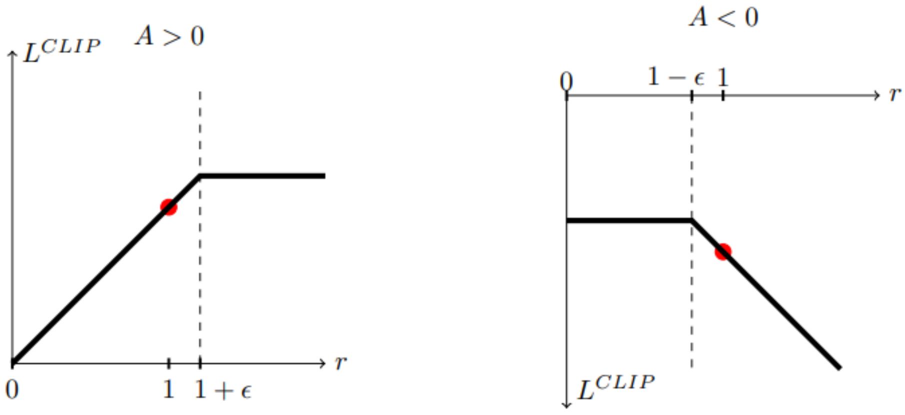
$$\tilde{J}(\theta, \theta_k) = \mathbb{E}_{\pi_{\theta_k}} \left[\min \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \text{clip} \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}}(s, a) \right) \right]$$

  where ϵ is a hyperparameter controlling the clipping.
- The update rule is again given by:

$$\theta_{k+1} = argmax_\theta \tilde{J}(\theta, \theta_k)$$

- **Note**: Often then advantage function is approximated by a critic network in practice!

PPO III



Offline Actor Critic: Idea

- It's possible to make Actor Critic fully Off Policy
- Start from DQN update:

$$\phi \leftarrow \phi + \alpha_t \left(r(s,a) + \gamma \max_{a'} Q(s',a';\phi) - Q(s,a;\phi) \right) \nabla_\phi Q(s,a;\phi)$$

- Approximate $\max_{a'} Q(s',a';\phi) \approx Q(s',\pi_\theta(s');\phi)$, i.e. we parametrize the policy and use that as a proxy for a'

References

- John Schulman, Sergey Levine, Philipp Moritz , Michael Jordan, Pieter Abbeel: Trust Region Policy Optimization, https://arxiv.org/pdf/1502.05477.pdf
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov: Proximal Policy Optimization, https://arxiv.org/pdf/1707.06347.pdf
- Ronald J. Williams: Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning, https://link.springer.com/content/pdf/10.1007/BF00992696.pdf (REINFORCE Algorithm)
- Mnih et Al.: Asynchronous Methods for Deep Reinforcement Learning, https://arxiv.org/pdf/1602.01783.pdf
- Script from Probabilistic Artificial Intelligence course, ETH, Andreas Krause and Jonas Hubotter