IIIS 2018 Spring: ATCS - Learning and Optimization - theory and practice
Lecture date: Jun 11, 2018
Instructor: Jian Li **Scribe: Yu Shi**

1 Introduction

In this lecture, we first introduce policy gradient. The basic version of policy gradient algorithm is based on Monte Carlo estimate, which has high variance. We show how the variance can be reduced by using baseline. Then we introduce the actor-critic method, to further reduce variance and accelerate learning. Finally, we briefly introduce some advanced topics related to policy gradient.

2 Policy Gradient

With infinite states in a reinforcement learning problem, we can use machine learning models to approximate the value function. The idea of policy gradient is to parametrize the policy in a machine learning model, and learn the policy using gradient methods. The input of the model is the states and the output is actions. Specifically, let θ be the parameters of the model for policy, then $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)]$ is the expected return by following the policy π_θ specified by θ, where τ is the trajectory $(s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$ generated by following π_θ. We can use gradient ascent to update θ by $\theta \leftarrow \theta + \eta\nabla_\theta J(\theta)$. By expanding $J(\theta)$, we have

$$J(\theta) = \int_\tau r(\tau)p_\theta(\tau)\,\mathrm{d}\tau = \int_\tau r(\tau)p(s_0)\prod_{t=0}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|a_t, s_t)\,\mathrm{d}\tau$$

Thus

$$\begin{aligned}\nabla_\theta J(\theta) &= \int_\tau r(\tau)\nabla_\theta p_\theta(\tau)\,\mathrm{d}\tau = \int_\tau r(\tau)\frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)}p_\theta(\tau)\,\mathrm{d}\tau \\ &= \int_\tau r(\tau)p_\theta(\tau)\nabla_\theta \ln p_\theta(\tau)\,\mathrm{d}\tau = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[r(\tau)\nabla_\theta \log p_\theta(\tau)]\end{aligned}$$

By using Monte Carlo estimate, we have

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} r(\tau_i)\nabla_\theta \log_\theta p_\theta(\tau_i)$$

Note that

$$\begin{aligned}\nabla_\theta \log p_\theta(\tau) &= \nabla_\theta\left(\log p(s_0) + \sum_t \log p(s_t|s_{t-1}, a_{t-1}) + \sum_t \log \pi_\theta(a_t, s_t)\right) \\ &= \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)\end{aligned}$$

Let $\pi_\theta(\tau) = \prod_t \pi_\theta(a_t|s_t)$, then we have $\nabla_\theta \log p_\theta(\tau) = \nabla_\theta \log \pi_\theta(\tau)$. Thus

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} r(\tau_i)\nabla_\theta \log \pi_\theta(\tau_i) \tag{1}$$

which can be easily computed by back-propagation on the policy model. However, the Monte Carlo estimate requires repeatedly sampling trajectories and the values $\nabla_\theta \log \pi_\theta(\tau_i)$ often have large variance, which is difficult for training. Also, equation (1) favors trajectories with large reward.

3 Reduce Variance With Baseline Advantage

To solve the problems above, we can reduce the variance of the Monte Carlo estimate by introducing a baseline b into (1)

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau_i) \left(r(\tau_i) - b\right)$$

here b is a baseline which can depend on states $s = (s_0, s_1, \ldots, s_T)$. Note that this modification does not change the expectation of (1), since

$$\begin{aligned}
\mathop{\mathbb{E}}_{\tau}\left[\nabla_\theta \log \pi_\theta(\tau) b\right] &= \int \pi_\theta(\tau) \frac{\nabla \pi(\tau)}{\pi_\theta(\tau)} b \, \mathrm{d}\tau = \int b \nabla_\theta \pi_\theta(\tau) \, \mathrm{d}\tau \\
&= \nabla_\theta \int b \pi_\theta(\tau) \, \mathrm{d}\tau = \nabla_\theta \int b \prod_t \pi_\theta(a_t|s_t) \, \mathrm{d}\tau \\
&= \nabla_\theta \int b \prod_t \int \left(\pi_\theta(a_t|s_t) \mathrm{d}a_t\right) \, \mathrm{d}s = \nabla_\theta \int b \, \mathrm{d}s = 0
\end{aligned}$$

One common choice of b is the average return of all trajectories

$$b = \frac{1}{N} \sum_{i=1}^{N} r(\tau_i)$$

The variance of $\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)$ can be expressed as

$$\begin{aligned}
\mathrm{Var}\left[\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)\right] &= \mathop{\mathbb{E}}_{\tau}\left[\left(\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)\right)^2\right] - \mathop{\mathbb{E}}_{\tau}\left[\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)\right]^2 \\
&= \mathop{\mathbb{E}}_{\tau}\left[\left(\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)\right)^2\right] - \mathop{\mathbb{E}}_{\tau}\left[\nabla_\theta \log \pi_\theta(\tau) r(\tau)\right]^2
\end{aligned}$$

Set the gradient of b to 0

$$2 \mathop{\mathbb{E}}_{\tau}\left[\nabla_\theta \log \pi_\theta(\tau)^T \nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)\right] = 0$$

which gives the optimal b that minimizes the variance

$$b = \frac{\mathbb{E}_\tau[g^2(\tau) r(\tau)]}{\mathbb{E}_\tau[g^2(\tau)]}$$

where $g^2(\tau) = \nabla_\theta \log \pi_\theta(\tau)^T \nabla_\theta \log \pi_\theta(\tau)$.

4 Actor-Critic Method

The above algorithms based on Monte Carlo method use the return value $r(\tau_i)$ for each trajectory, which means the update of θ can be done only an episode is ended. Thus they are inconvenient for continuing problems. The actor-critic method introduce the idea of bootstrap to further reduce the variance and make the parameter update available for every step in a trajectory. First, we do some further approximation on $\nabla_\theta J(\theta)$,

$$\begin{aligned}
\nabla_\theta J(\theta) &\approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\left(\sum_{t'=1}^{T} r(s_{t'}^i, a_{t'}^i)\right)\right) \\
&\approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\left(\sum_{t'=t}^{T} r(s_{t'}^i, a_{t'}^i)\right)\right) \\
&\approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i) Q^{\pi_\theta}(s_t^i, a_t^i)\right)
\end{aligned}$$

The intuition is that the reward before step t has no correlation with the gradients $\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)$. Thus $\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\left(\sum_{t'=1}^{t-1} r(s_{t'}^i, a_{t'}^i)\right)\right) \approx 0$. Here

$$Q^{\pi_\theta}(s_t^i, a_t^i) = \sum_{t'=t}^{T}\mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[r(s_i^t, a_i^t)|s_i^t, a_i^t\right]$$

is the true expectation of the reward to go after step t. By using the state-value function $V(s_t^i) = \mathbb{E}_{a\sim\pi_\theta(a|s_t^i)}$ as the baseline

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\left(Q^{\pi_\theta}(s_t^i, a_t^i) - V^{\pi_\theta}(s_t^i)\right)\right)$$

If we can estimate the advantage $A^{\pi_\theta}(s_t^i, a_t^i) = Q^{\pi_\theta}(s_t^i, a_t^i) - V^{\pi_\theta}(s_t^i)$, then we can avoid going through the whole episode for a update. First, we fit a value function $\hat{V}^{\pi_\theta}(s)$ by machine learning model, and estimate $Q^{\pi_\theta}(s_t^i, a_t^i)$ using

$$\hat{Q}^{\pi_\theta}(s_t^i, a_t^i) = r(s_t^i, a_t^i) + \hat{V}^{\pi_\theta}(s_{t+1}^i)$$

Now the advantage is estimated by

$$\hat{A}^{\pi_\theta}(s_t^i, a_t^i) = r(s_t^i, a_t^i) + \hat{V}^{\pi_\theta}(s_{t+1}^i) - \hat{V}^{\pi_\theta}(s_t^i)$$

To learn $\hat{V}^{\pi_\theta}$, we use $\left\{(s_i^t, \sum_{t'=t}^{T} r(s_i^{t'}, a_i^{t'}))\right\}_i$ as training data and fit the parameters ϕ of the state-value model using supervised learning. For example let $y_i^t = \sum_{t'=t}^{T} r(s_i^{t'}, a_i^{t'})$, then we can fit ϕ by minimize $\mathcal{L}(\phi) = \sum_{i,t}\|\hat{V}^{\pi_\theta}(s_i^t) - y_i^t\|^2$.

By using the idea of bootstrap, the training target can be further simplified

$$y_i^t = \sum_{t'=t}^{T} r(s_i^{t'}, a_i^{t'}) \approx r(s_i^t) + \hat{V}^{\pi_\theta}(s_i^{t+1})$$

Now we can update the policy parameters inside an episode. Algorithm 1 and 2 shows the details of batch version and online version of the actor-critic algorithm.

Algorithm 1: Batch Actor-Critic

repeat forever:
  sample a series $\{s_i, a_i\}$ by following $\pi_\theta(a|s)$
  use supervised learning to fit the parameters ϕ of $\hat{V}^{\pi_\theta}$
  update $\hat{A}^{\pi_\theta}(s_i, a_i) \leftarrow r(s_i, a_i) + \hat{V}^{\pi_\theta}(s'_i) - \hat{V}^{\pi_\theta}(s_i)$, where s'_i is the next state of s_i
  calculate $J(\theta)$
  update $\theta \leftarrow \theta + \eta \nabla_\theta J(\theta)$

Algorithm 2: Online Actor-Critic

repeat forever:
  sample one step (s_i, a_i) by following $\pi_\theta(a|s)$
  update the parameters ϕ of $\hat{V}^{\pi_\theta}$ using the one step reward
  update $\hat{A}^{\pi_\theta}(s_i, a_i) \leftarrow r(s_i, a_i) + \hat{V}^{\pi_\theta}(s'_i) - \hat{V}^{\pi_\theta}(s_i)$, where s'_i is the next state of s_i
  calculate $J(\theta)$
  update $\theta \leftarrow \theta + \eta \nabla_\theta J(\theta)$

5 Other Topics Related to Policy Gradient

1. Share parameters of the state-value model and the policy model, or even using a single model to produce both the state-value $\hat{V}(s)$ and the policy $\pi(a|s)$. The idea is that the two models may need some common features extracted from the state.

2. Parallelism of policy gradient. Each core in a multi-core machine samples a trajectory, then update the parameters together, can be asynchronous. For example, the A3C[2] algorithm.

3. Advanced Policy Gradient Methods: [1], TRPO[3] and PPO [4]

References

[1] Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

[2] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[3] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.