Reinforcement Learning

An Introductory Note

Jingye Wang

wangjy5@shanghaitech.edu.cn

Spring 2020

Contents

9 Policy Optimization

For value-based reinforcement learning, deterministic policy is generated directly from the value function using greedy $a = \arg\max_{a'} q(a', s)$. Now instead we can parameterize the policy function as $\pi_\theta(a|s)$ where θ is the learnable policy parameter and the output is a probability over the action set.

Value-based *v.s.* Policy-based

- Value-based methods: solve RL problems through dynamic programming
  - related to classic RL and control theory;
  - learn value function;
  - generate an implicit policy based on the value function;
  - learn a deterministic policy based on the estimated action values;
  - developed by Richard Sutton, David Silver, DeepMind;
- Policy-based methods: solve RL problems mainly through learning
  - related to machine learning and deep learning;
  - do not require value function for action selection;
  - learn a stochastic policy;
  - developed by Pieter Abbeel, Sergey Levine, OpenAI, Berkeley;

The two methods can also be combined together. A popular algorithm called *Actor-Critic* entails learning both policy and value function.

Pros and cons of Policy-based

- Advantages:
  - can converge on a local optimum (worst case) or global optimum (best case);
  - is effective in high-dimensional action space;
- Disadvantages:
  - typically converges to a local optimum;
  - the policy is of high variance;

9.1 Policy Optimization Theorem

Objective of Optimization Policy: *Given a policy approximator $\pi(s, a)$ with parameter θ, find the $\theta^\star$ that gives us the optimal policy.*

One thing we care is how do we measure the quality of a policy π_θ? Let τ be a trajectory sampled from the policy function π_θ, then we defined the value of policy π_θ as

$$J(\theta) = \mathbb{E}_\tau \left[\sum_t r(s_t, a_t^\tau) \right].$$

Thus we have the goal of policy-based methods as

$$\theta^* = \arg\max_\theta \mathbb{E}_\tau \left[\sum_t r(s_t, a_t^\tau) \right].$$

However, such $J(\theta)$ may not available or handy. Hence a trick is using approximation. For example,

- In the episodic environment with discrete space, we can use the value of the starting state s_0:

$$J(\theta) \approx v^{\pi_\theta}(s_0) = \mathbb{E}_{\pi_\theta}[v(s_0)],$$

- In the environment with continuous space, we can use the average reward:

$$J(\theta) \approx \sum_s d^{\pi_\theta}(s) v^{\pi_\theta}(s) = \sum_s d^{\pi_\theta} \sum_{a \in \mathcal{A}} \pi_\theta(a|s) q^{\pi_\theta}(s, a),$$

  where d^{π_θ} is the stationary distribution of Markov chain for π_θ.

Depends on the form of $J(\theta)$, we have different methods to maximize it

- If $J(\theta)$ is differentiable, we can use gradient-based methods:
  - Gradient Ascend;
  - Conjugate Gradient;
  - Quasi-newton.
- If $J(\theta)$ is non-differentiable or hard to compute the derivative, we can use some derivative-free black-box optimization methods:
  - Cross-entropy Method (CEM);
  - Hill Climbing;
  - Evolution Algorithm;
  - Approximate Gradients by Finite Difference.

In this note, we mainly focus on gradient-based methods.

Policy Gradient Theorem: *For a policy π_θ with parameter θ, we have the gradient as*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \ln \pi(a|s;\theta) q^\pi(s, a)].$$

The proof is available in the Section 12.1 of *Reinforcement Learning: An Introduction* [1]. A detailed proof is also given in the blog [10]. The following *proof* is just for personal interest.

Proof:

For simplicity, we leave it implicit in all cases that π is a function of θ and all gradients are also implicitly with respect to θ. Notice that the gradient of the state-value function:

$$
\begin{aligned}
\nabla v^\pi(s) &= \nabla \left(\sum_a \pi(a|s) q^\pi(s,a) \right) \\
&= \sum_a \left(\nabla \pi(a|s) q^\pi(s,a) + \pi(a|s) \nabla q^\pi(s,a) \right) && \text{(Product rule of calculus)} \\
&= \sum_a \left(\nabla \pi(a|s) q^\pi(s,a) + \pi(a|s) \nabla \left(R_s^a + \sum_{s'} \mathcal{P}_{ss'}^a v^\pi(s') \right) \right) && \text{(Sec 5.2 MDP)} \\
&= \sum_a \left(\nabla \pi(a|s) q^\pi(s,a) + \pi(a|s) \nabla \sum_{s'} \mathcal{P}_{ss'}^a v^\pi(s') \right) && \text{(R_s^a is irrevalent to θ)} \\
&= \sum_a \left(\nabla \pi(a|s) q^\pi(s,a) + \pi(a|s) \sum_{s'} \mathcal{P}_{ss'}^a \nabla v^\pi(s') \right),
\end{aligned}
$$

which gives us a nice recursive form of the gradient. Therefore the future state value function $v^\pi(s')$ can be repeated unrolled by following the same equation.

Before unrolling, we define the probability of transitioning from state s to state s' in k steps under policy π as $\rho^\pi(s \to s', k)$. Thus it follows that

- when $k = 0$: obviously, we have $\rho^\pi(s \to s, k = 0) = 1$;
- when $k = 1$, it is easy to get the probability from the transition matrix as $\rho^\pi(s \to s', k = 1) = \sum_a \pi(a|s)\mathcal{P}_{ss'}^a$;
- for other cases such as going from state s to s' in $k+1$ steps, we can consider a middle state x where it takes k steps from s to x, thus we have a recursively expression as $\rho^\pi(s \to s', k+1) = \sum_x \rho^\pi(s \to x, k)\rho^\pi(s \to s', 1)$.

We now consider unrolling the recursive representation of $\nabla v^\pi(s)$. For simplicity, let $\phi(s) = \sum_a \nabla \pi(a|s) q^\pi(s,a)$. Then it follows that

$$
\begin{aligned}
\nabla v^\pi(s) &= \phi(s) + \sum_a \pi(a|s) \sum_{s'} \mathcal{P}_{ss'}^a \nabla v^\pi(s') \\
&= \phi(s) + \sum_{s'} \sum_a \pi(a|s) \mathcal{P}_{ss'}^a \nabla v^\pi(s') && \text{($\pi(a|s)$ is irrevalent to s')} \\
&= \phi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \nabla v^\pi(s') \\
&= \phi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \left(\phi(s') + \sum_{s''} \rho^\pi(s' \to s'', 1) \nabla v^\pi(s'') \right) \\
&= \phi(s) + \sum_{s'} \rho^\pi(s \to s', 1)\phi(s') + \sum_{s''} \rho^\pi(s \to s'', 2) \nabla v^\pi(s'') \\
&= \phi(s) + \sum_{s'} \rho^\pi(s \to s', 1)\phi(s') + \sum_{s''} \rho^\pi(s \to s'', 2)\phi(s'') + \sum_{s'''} \rho^\pi(s \to s''', 3) \nabla v^\pi(s''') \\
&= \ldots \\
&= \sum_{s' \in \mathcal{S}} \sum_{k=0}^{\infty} \rho^\pi(s \to s', k)\phi(s'),
\end{aligned}
$$

which finally arrives at

$$\nabla v^\pi(s) = \sum_s \sum_{k=0}^{\infty} \rho^\pi(s \to s', k) \sum_a \nabla \pi(a|s') q^\pi(s', a).$$

Thus for the $J(\theta)$ of the episodic environment, we have

$$\begin{aligned}
\nabla J(\theta) &= \nabla v^\pi(s_0) \\
&= \sum_s \sum_{k=0}^{\infty} \rho^\pi(s_0 \to s, k) \sum_a \nabla \pi(a|s) q^\pi(s, a) \\
&= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q^\pi(s, a) && (\eta(s) = \textstyle\sum_{k=0}^{\infty} \rho^\pi(s_0 \to s, k)) \\
&= \left(\sum_s \eta(s)\right) \sum_s \frac{\eta(s)}{\sum_s \eta(s)} \sum_a \nabla \pi(a|s) q^\pi(s, a) \\
&\propto \sum_s d^\pi(s) \sum_a \nabla \pi(a|s) q^\pi(s, a)
\end{aligned}$$

as $\sum_s \eta(s)$ is a constant and $d^\pi(s)$ is exactly the stationary distribution. Further, such deriving also shows the connection between the two different $J(\theta)$ we defined above. Now we rewrite the gradient as

$$\begin{aligned}
\nabla J(\theta) &\propto \sum_s d^\pi(s) \sum_a \nabla \pi(a|s) q^\pi(s, a) \\
&= \sum_s d^\pi(s) \sum_a q^\pi(s, a) \pi(a, s) \frac{\nabla \pi(a|s)}{\pi(a|s)} \\
&= \mathbb{E}_{\pi_\theta}[\nabla_\theta \ln \pi(a|s; \theta) q^\pi(s, a)].
\end{aligned}$$

□

For other general forms of policy gradient methods, one can refer to the paper [11] and the note [12] (again, many thanks to lilianweng's blog [10]).

9.2 REINFORCE: Monte-Carlo Policy Gradient

We now consider the policy gradient for the case where we can get complete episodes. According to the *policy gradient theorem*, it follows that

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}[\nabla_\theta \ln \pi(a|s; \theta) q^\pi(s, a)] \\
&= \mathbb{E}_{\pi_\theta}[\nabla_\theta \ln \pi(a|s; \theta) G_t]
\end{aligned}$$

as $q^\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$. Thus it is similar to *Monte-Carlo policy evaluation*, we can measure G_t from real complete sample episodes and use that to update our policy gradient.

Monte-Carlo Policy Gradient: *Starting from the state s_0 sampled from a distribution $d(s)$, we denote one episode as*

$$\tau = (s_0, a_0, r_1, ..., s_{T-1}, a_{T-1}, r_T) \sim (\pi_\theta, \mathcal{P}^a_{s_t s_{t+1}}).$$

Then we update the parameter in the rule:

$$\theta \leftarrow \theta + \alpha\gamma^t G_t \nabla_\theta \ln \pi(a|s;\theta).$$

Algorithm 20 REINFORCE: Monte-Carlo Policy-Gradient Control

initialize policy parameter $\theta \in \mathbb{R}^d$;
input a differentiable policy parameterization $\pi(a|s;\theta)$; the step size α;
for true **do**:
  Generate an episode $(s_0, a_0, r_1, ..., s_{T-1}, a_{T-1}, r_T)$;
  for each step of the episode $t = 0, 1, ..., T-1$ **do**:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$;
    $\theta \leftarrow \theta + \alpha\gamma^t G \nabla \ln \pi(a_t|s_t;\theta)$;
  end for
end for

The analysis for REINFORCE requires us to consider the episode level. We define the sum of rewards over the trajectory τ (for G_t baseline, the case is quite similar) as

$$R(\tau) = \sum_{t=1}^{T} r_t.$$

Let $\mathcal{D}(\tau;\theta) = d(s_0)\prod_{t=0}^{T-1}\pi(a_t|s_t;\theta)\mathcal{P}_{s_t s_{t+1}}^{a_t}$ denote the probability over trajectories when executing the policy π_θ. Then the policy gradient of $J(\theta)$ is equivalent to

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} r_t\right] \\
&= \sum_\tau \nabla_\theta \mathcal{D}(\tau;\theta) R(\tau) \\
&= \sum_\tau \mathcal{D}(\tau;\theta) R(\tau) \frac{\nabla_\theta \mathcal{D}(\tau;\theta)}{\mathcal{D}(\tau;\theta)} \\
&= \sum_\tau \mathcal{D}(\tau;\theta) R(\tau) \nabla_\theta \ln \mathcal{D}(\tau;\theta) \\
&\approx \frac{1}{m}\sum_{i=1}^{m} R(\tau_i) \nabla_\theta \ln \mathcal{D}(\tau;\theta),
\end{aligned}$$

where we suppose there are m episodes and refer to MC thought to approx the expectation. Such approximation is reasonable as long as $m \to \infty$.

We now show that such method does not need the dynamics of the model. Considering the term that matters in the gradient we got above, it follows that

$$\begin{aligned}
\nabla_\theta \ln \mathcal{D}(\tau;\theta) &= \nabla_\theta \ln \left[\mu(s_0)\prod_{t=0}^{T-1}\pi(a_t|s_t;\theta)\,\mathcal{P}_{s_t s_{t+1}}^{a_t}\right] \\
&= \nabla_\theta \left[\ln\mu(s_0) + \sum_{t=0}^{T-1}\ln\pi(a_t|s_t;\theta) + \ln\mathcal{P}_{s_t s_{t+1}}^{a_t}\right] \\
&= \sum_{t=0}^{T-1}\nabla_\theta \ln\pi(a_t|s_t;\theta)\,,
\end{aligned}$$

and thus

$$\nabla_\theta J(\theta) \approx \frac{1}{m}\sum_{i=1}^{m} R(\tau_i)\left(\sum_{t=0}^{T-1}\nabla_\theta \ln\pi(a_t^i|s_t^i;\theta)\right).$$

It shows that the dynamics, *i.e.* the transition matrix, of the model is not needed, which means policy gradient is a model-free method.

9.3 Actor-Critic Policy Gradient

Recall that in section 8 we use $\hat{v}(s;\boldsymbol{w}) \approx v^\pi(s)$. Now we combine the value approximation with policy approximation, then we will get the Actor-Critic method.

Actor: *Actor is actually a policy parameterization $\pi(a|s;\theta)$ to generate actions; it updates parameter θ in direction suggested by critic.*

Critic: *Critic is actually a value approximation $\hat{v}(s;\boldsymbol{w})$ to evaluate the reward of a state under current actor (policy); it needs to update parameter $\boldsymbol{w}$ to make accurate evaluation.*

Algorithm 21 Actor-Critic

initialize policy parameter θ; initialize the state-value function parameter $\boldsymbol{w}$;
input a differentiable policy parameterization $\pi(a|s;\theta)$; a differentiable state-value function parameterization $\hat{v}(s;\boldsymbol{w})$, the step size $\alpha_{\boldsymbol{w}}, \alpha_\theta$;
for true **do**:
    Generate a start state s;
    $I \leftarrow 1$
    while s is not terminal **do**:
        Choose action $a \leftarrow \pi(a|s;\theta)$;
        $r, s' \leftarrow environment(s,a)$;
        $\delta \leftarrow r + \gamma\hat{v}(s';\boldsymbol{w}) - \hat{v}(s;\boldsymbol{w})$;
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha_{\boldsymbol{w}}\delta\nabla\hat{v}(s;\boldsymbol{w})$;
        $\theta \leftarrow \theta + \alpha_\theta I\delta\nabla\ln\pi(a|s;\theta)$;
        $I \leftarrow \gamma I$;
        $s \leftarrow s'$;
    end while
end for

As we can see, the critic is solving a familiar problem *policy evaluation*, while the actor is doing *policy improvement*.

9.4 Extension of Policy Gradient

Nowadays, State-of-the-art RL methods are almost all policy-based.

A2C, A3C: Asynchronous Methods for Deep Reinforcement Learning, ICML'16. Representative high-performance actor-critic algorithm.

TRPO: Trust region policy optimization: deep RL with natural policy gradient and adaptive step size.

PPO: Proximal policy optimization algorithms: deep RL with importance sampled policy gradient.