Statistical Techniques in Robotics (16-831, S20) Lecture #21 (Wednesday, April 6)

Policy Gradient Methods, Actor-Critic

Lecturer: Kris Kitani *Scribes: Xiaofeng Guo*

1 Review

In the last lectures, we talked about the topic of Function Approximators. In this lecture, we will introduce the Policy-based method: Policy Gradient and hybrid method:Actor-Critic for RL. In this section, we will review the function approximation.

1.1 Function Approximation

If we assume the a finite state and action space, it will lead to a tabular value function and we have multiple different model-free methods based on this assumption. However, when the state space is very large or continuous, we can not represent the value function as the tabular case. Therefore, we need to approximate the value function $V^\pi(s)$ by a parametric function $V_\theta(s)$ so that $V_\theta(s) \approx V^\pi(s)$. We can use multiple different approximators such as linear approximator, neural network, or deep neural network.

1.1.1 Value Function Approximation

To approximate the value function $V^\pi(s)$ by a parametric function $V_\theta(s)$, the goal is to find the optimal θ

$$\hat{\theta} = \arg\min_\theta \mathbb{E}_p[(V(s) - V_\theta(s))^2]$$

.

We can find the solution using the stochastic gradient descent method:

$$\nabla_\theta \mathcal{L}_\theta \approx -(V(s) - V_\theta(s))\frac{\partial}{\partial\theta}V_\theta(s)$$

.

Approximating the gradient requires a return estimator:

$$V(s) = \mathbb{E}[G^{(t)}] \approx G^{(t)}.$$

We can select the estimator or any prediction method we have learned, such as Monte-Carlo, TD, etc. and perform SGD to find the optimal θ.

1.1.2 Function Approximation for Control

The function approximation for control is similar as above. By replacing $V(s)$ with $Q(s,a)$, we have the goal is to approximate the value function so that $Q_\theta(s,a) \approx Q^\pi(s,a)$. We have

$$\hat{\theta} = \arg\min_\theta \mathbb{E}_p[(Q^\pi(s,a) - Q_\theta(s,a))^2]$$

.

$$\nabla_\theta \mathcal{L}_\theta \approx -(Q^\pi(s,a) - Q_\theta(s,a))\frac{\partial}{\partial\theta}Q_\theta(s,a)$$

.

$$Q^\pi(s,a) \approx G^{(t)}.$$

We can select the estimator or any prediction method we have learned to find the optimal θ.

2 Summary

2.1 Policy Gradient Method

In this subsection, we will summary the Policy Gradient Method. In valued-based methods, the goal is to estimate the value function and the optimal policy was derived from the value function. Different with that, in policy-based method, we are going to estimate the policy π directly without estimating the value function. This class of models is called policy gradient methods. For a parametrized policy $\pi_\theta(a|s)$, we want to optimize its parameter θ that maximizes the expect accumulative return:

$$\begin{aligned}\hat{\theta} &= \arg\max_\theta \mathbb{E}_{p_\theta(\zeta)}[\sum_{t=0}^{T} r^{(t)}] \\ &= \arg\max_\theta J(\theta)\end{aligned} \tag{1}$$

where

$$p_\theta(\zeta) = p(s^{(0)})\Pi_{t=0}^{T}\pi_\theta(a^{(t)}|s^{(t)})p(s^{(t+1)}|s^{(t)},a^{(t)})$$
$$\zeta = \{s^{(0)},a^{(0)},s^{(1)},a^{(1)},...,s^{(T)},a^{(T)}\}$$
$$r^{(t)} \triangleq r(s^{(t+1)},a^{(t)},s^{(t)}).$$

To handle this optimization problem, we will use gradient ascent method to find the optimal θ.

2.1.1 Gradient Ascent Method

Here we will use gradient ascent method to solve Equation 1. First, we make a linear approximation of the objective function with quadratic regularization:

$$\hat{\theta} = \arg\max_{\theta}\{\alpha(J(\theta') + \langle \theta - \theta', \nabla_{\theta'} J(\theta')\rangle) - \frac{1}{2}||\theta - \theta'||^2\}.$$

Optimizing the Lagrangian, we get:

$$\nabla_{\theta}\{\alpha(J(\theta') + \langle \theta - \theta', \nabla_{\theta'} J(\theta')\rangle) - \frac{1}{2}||\theta - \theta'||^2\} = 0$$

$$\Rightarrow \alpha\nabla_{\theta'} J(\theta') - \theta + \theta' = 0.$$

Therefore, we get the parameter θ update equation:

$$\theta \leftarrow \theta' + \alpha\nabla_{\theta'} J(\theta'). \tag{2}$$

Recall that we have:

$$\begin{aligned} J(\theta) &= \mathbb{E}_{p_\theta(\zeta)}[\sum_{t=0}^{T} r^{(t)}] \\ &= \int_{\zeta} p_\theta(\zeta) r(\zeta) d\zeta \qquad \text{by definition of expectation.} \end{aligned}$$

and

$$r(\zeta) = \sum_{t=0}^{T} (r^{(t)}).$$

Therefore, we have the gradient of objective:

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \int_{\zeta} p_\theta(\zeta) r(\zeta) d\zeta \qquad \text{by definition} \\ &= \int_{\zeta} \nabla_\theta p_\theta(\zeta) r(\zeta) d\zeta \qquad \text{linearity of gradient} \\ &= \int_{\zeta} p_\theta(\zeta) \frac{\nabla_\theta p_\theta(\zeta)}{p_\theta(\zeta)} r(\zeta) d\zeta \qquad \text{multiply by one} \\ &= \int_{\zeta} p_\theta(\zeta) \nabla_\theta \ln p_\theta(\zeta) r(\zeta) d\zeta \qquad \text{derivative of log 'trick'.} \end{aligned}$$

Then, let's take a look at the term $\nabla_\theta \ln p_\theta(\zeta)$. We can decompose it as:

$$\begin{aligned} \nabla_\theta \ln p_\theta(\zeta) &= \nabla_\theta [\ln p(s^{(0)}) + \sum_{t=1}^{T} \ln \pi_\theta(a^{(t)}|s^{(t)}) + \ln p(s^{(t+1)}|s^{(t)}, a^{(t)})] \\ &= \nabla_\theta [\sum_{t=1}^{T} \ln \pi_\theta(a^{(t)}|s^{(t)})] \qquad \text{only policy is dependent on } \theta. \end{aligned}$$

Bring it back to the gradient $\nabla_\theta J(\theta)$, we have

$$\begin{aligned}
\nabla_\theta J(\theta) &= \int_\zeta p_\theta(\zeta) \nabla_\theta \ln p_\theta(\zeta) r(\zeta) d\zeta \\
&= \int_\zeta p_\theta(\zeta) \nabla_\theta (\sum_{t=1}^{T} \ln \pi_\theta(a^{(t)}|s^{(t)})) r(\zeta) d\zeta \qquad \text{keep only terms dependent on } \theta \\
&= \mathbb{E}_{p_\theta(\zeta)}[(\sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a^{(t)}|s^{(t)}))(\sum_{t=1}^{T} r^{(t)})]
\end{aligned} \tag{3}$$

We call the term $\nabla_\theta \ln \pi_\theta(a^{(t)}|s^{(t)})$ "eligibility vector" and the term $\sum_{t=1}^{T} r^{(t)}$ "episodic return".

Let's first look at the "eligibility vector". We have

$$\nabla_\theta \ln \pi_\theta(a^{(t)}|s^{(t)}) = \frac{\nabla_\theta \pi_\theta(a^{(t)}|s^{(t)})}{\pi_\theta(a^{(t)}|s^{(t)})},$$

in which $\nabla_\theta \pi_\theta(a^{(t)}|s^{(t)})$ represents the change in policy π_θ per unit change in θ and $\pi_\theta(a^{(t)}|s^{(t)})$ represents the inverse probability weighting. Therefore, the "eligibility vector" shows the weighted sensitivity of the policy to change in parameter. Specifically, low probability actions have bigger gradient and high probability actions have smaller gradient.

For the "episodic return", we have $r(\zeta) = \sum_{t=1}^{T} r^{(t)}$. It shows the goodness of the current policy and the gradient is Weighted by this episodic return. Specifically, a good trajectory (large reward) has a bigger gradient and a bad trajectory (small reward) has a smaller gradient.

Note that it is not practical to sum over all possible trajectories when we calculate the expectation, so we approximate it with samples as Monte-Carlo estimate. Therefore, we have

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{p_\theta(\zeta)}[(\sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a^{(t)}|s^{(t)}))(\sum_{t=1}^{T} r^{(t)})] \\
&\approx \frac{1}{N} \sum_{n=1}^{N} (\sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a^{(n,t)}|s^{(n,t)}))(\sum_{t=1}^{T} r^{(n,t)}).
\end{aligned} \tag{4}$$

Therefore, we can compute the parameter update Equation 2.

2.1.2 Policy Gradient Algorithms

Based on the above equations, we get the Monte-Carlo Policy Gradient Algorithm as Algorithm 1.

Algorithm 1 MC-Policy-Gradient(π_θ, α)

for $e = 1, ..., E$ **do**
  $\{s^{(t)}, a^{(t)}, r^{(t)}\}_{t=0}^{T} \sim \mathcal{E}|\pi_\theta$
  $G^{(0)} = \sum_{t=0}^{T} r^{(t)}$
  for $t = 0, ..., T$ **do**
    $\theta \leftarrow \theta + \alpha(\nabla_\theta \log \pi_\theta(a^{(t)}|s^{(t)}))(G^{(0)})$
  end for
end for
return π_θ

However, there is a problem in Monte-Carlo Policy Gradient Method. Monte-Carlo estimation has zero bias but large variance. There are two ways to reduce its variance. In the first method, we can enforce causality. We made a simple observation that the policy decision at time t' cannot affect past rewards at $t < t'$, which means that the update to the policy at time step t should only depend on future reward. Therefore, when computing the $\nabla_\theta J(\theta)$, instead on computing the reward for the entire sequence, we only compute the future reward. Therefore, we have

$$\begin{aligned}\nabla_\theta J(\theta) &\approx \frac{1}{N}\sum_{n=1}^{N}(\sum_{t=0}^{T}\nabla_\theta \ln \pi_\theta(a^{(n,t)}|s^{(n,t)}))(\sum_{t=0}^{T} r^{(n,t)}) \\ &\approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T}\nabla_\theta \ln \pi_\theta(a^{(n,t)}|s^{(n,t)})(\sum_{t'=t}^{T} r^{(n,t')})\end{aligned}$$

The other method to reduce the variance is to remove the gradient bias with a baseline offset. We know that the reward values might be very large, which may cause very large gradients. We can adjust the gradient by subtracting some offset b:

$$\nabla_\theta J(\theta) = \int_\zeta p_\theta(\zeta)\nabla_\theta \ln p_\theta(\zeta)[r(\zeta) - b]d\zeta.$$

We can prove that this offset will not change the expected gradient as bellows.

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \int_\zeta p_\theta(\zeta)\nabla_\theta \ln p_\theta(\zeta)[r(\zeta) - b]d\zeta \qquad \text{offset version of the gradient} \\
&= \int_\zeta p_\theta(\zeta)\nabla_\theta \ln p_\theta(\zeta)r(\zeta)d\zeta - \int_\zeta p_\theta(\zeta)\nabla_\theta \ln p_\theta(\zeta)bd\zeta \\
&= \int_\zeta p_\theta(\zeta)\nabla_\theta \ln p_\theta(\zeta)r(\zeta)d\zeta - \int_\zeta p_\theta(\zeta)\frac{\nabla_\theta p_\theta(\zeta)}{p_\theta(\zeta)}bd\zeta \\
&= \int_\zeta p_\theta(\zeta)\nabla_\theta \ln p_\theta(\zeta)r(\zeta)d\zeta - \int_\zeta \nabla_\theta p_\theta(\zeta)bd\zeta \\
&= \int_\zeta p_\theta(\zeta)\nabla_\theta \ln p_\theta(\zeta)r(\zeta)d\zeta - b\nabla_\theta \int_\zeta p_\theta(\zeta)d\zeta \qquad \text{integral of probability is one} \\
&= \int_\zeta p_\theta(\zeta)\nabla_\theta \ln p_\theta(\zeta)r(\zeta)d\zeta - b\nabla_\theta(1) \qquad \text{derivative of constant is zero} \\
&= \int_\zeta p_\theta(\zeta)\nabla_\theta \ln p_\theta(\zeta)r(\zeta)d\zeta - 0 \qquad \text{get back the original gradient.}
\end{aligned}
$$

Based on the second method, we can get the Episodic-REINFORCE algorithm as Algorithm 2.

Algorithm 2 Episodic-REINFORCE(π_θ, α)

for $e = 1, ..., E$ **do**
  $\{s^{(t)}, a^{(t)}, r^{(t)}\}_{t=0}^{T} \sim \mathcal{E}|\pi_\theta$
  $G^{(0)} = \sum_{t=0}^{T} r^{(t)}$
  for $t = 0, ..., T$ **do**
    $\theta \leftarrow \theta + \alpha[G^{(0)} - b]\nabla_\theta \log \pi_\theta(a^{(t)}|s^{(t)})$
  end for
end for
return π_θ

Putting these two modifications together, we get the policy gradient algorithm as Algorithm 3.

Algorithm 3 Policy-Gradient(π_θ, α, b)

for $e = 1, ..., E$ **do**
  $\{s^{(t)}, a^{(t)}, r^{(t)}\}_{t=0}^{T} \sim \mathcal{E}|\pi_\theta$
  for $t = 0, ..., T$ **do**
    $G^{(t)} = \sum_{i=t}^{T} r^{(i)}$
    $\theta \leftarrow \theta + \alpha[G^{(t)} - b]\nabla_\theta \log \pi_\theta(a^{(t)}|s^{(t)})$
  end for
end for
return π_θ

To conclude, we summary the pros and cons of policy gradient methods.

Pros: Firstly, it doesn't require model and learns from interaction. Secondly, it is effective for high-dimensional or continuous action spaces (all methods so far assumed finite action space). Next,

it can encode prior knowledge when designing policy architecture. Finally, it finds the optimal stochastic policy and naturally explores due to stochasticity.

Cons: Firstly, the gradient is typically high variance and leads to slow convergence. Secondly, a small step size leads to slow convergence. Next, it typically converges to local minima instead of global minima. Finally, it is easy to have exploding or zero gradients.

2.2 Actor Critic

We have been discussed about the valued-based method and policy-based method. Here we will discuss the hybrid of them. When the value function estimates are used as the baseline, for policy gradient methods, this class of methods is called actor-critic. This is currently the most popular RL approach.

Recall that in the policy gradient, we need the entire episode to compute the reward $\sum_{t=1}^{T} r^{(n,t)}$. However, we can use many prediction algorithms to estimate it so that we don't need full episodes. For continuous case, we can use function approximation for estimation. Therefore, similar with the form of Equation 3, we get the gradient of objective function:

$$\nabla_\theta J(\theta) = \mathbb{E}_{p_\theta(\zeta)}[(\sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a^{(t)}|s^{(t)}))(G_\phi^{(t)})],$$

where $G_\phi^{(t)}$ is the return value estimator, which is also called "Critic". By combining the policy gradient method with different approximate estimation method, we can get different Actor-Critic methods. For example, we can combine policy gradient with Q-MC Prediction and get Q-MC Actor Critic algorithm as Algorithm 4.

Algorithm 4 Q-MC-Actor-Critic($\pi_\theta, Q_\phi, \alpha, \beta$)

for $e = 1, ..., E$ **do**
  $\{s^{(t)}, a^{(t)}, r^{(t)}\}_{t=0}^{T} \sim \mathcal{E}|\pi_\theta$
  for $t = 0, ..., T$ **do**
    $G^{(t)} = \sum_{i=t}^{T} r^{(i)}$
    $\phi \leftarrow \phi + \beta[G^{(t)} - Q_\phi(a^{(t)}, s^{(t)})]\nabla Q_\phi(a^{(t)}, s^{(t)})$
    $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a^{(t)}|s^{(t)}) \cdot Q_\phi(a^{(t)}, s^{(t)})$
  end for
end for
return π_θ

Similarly, we can get Q-TD Actor Critic algorithm as Algorithm 5

Algorithm 5 Q-TD-Actor-Critic($\pi_\theta, Q_\phi, \alpha, \beta$)

for $e = 1, ..., E$ **do**
  $\{s^{(t)}, a^{(t)}, r^{(t)}\}_{t=0}^{T} \sim \mathcal{E}|\pi_\theta$
  for $t = 0, ..., T$ **do**
    $\delta \leftarrow r^{(t)} + Q_\phi(a^{(t+1)}, s^{(t+1)})$
    $\phi \leftarrow \phi + \beta[\delta - Q_\phi(a^{(t)}, s^{(t)})]\nabla Q_\phi(a^{(t)}, s^{(t)})$
    $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a^{(t)}|s^{(t)}) \cdot Q_\phi(a^{(t)}, s^{(t)})$
  end for
end for
return π_θ

Now let's go back and discuss about the offset b in

$$\nabla_\theta J(\theta) = \mathbb{E}_{p_\theta(\zeta)}[(\sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a^{(t)}|s^{(t)}))(\sum_{i=t}^{T} r^{(i)} - b).$$

Let's denote the term $\sum_{i=t}^{T} r^{(i)} - b$ as "Adjusted Return". We can derive that the optimal baseline b for policy gradient is the value function $V_\psi(s^{(t)})$ and we can get the optimal adjusted return for the policy gradient:

$$A(a^{(t)}, s^{(t)}) = Q_\phi(a^{(t)}, s^{(t)}) - V_\psi(s^{(t)}).$$

$A(a^{(t)}, s^{(t)})$ is called the "Advantage function". The policy gradient with Advantage function is called the "Advantage Actor-Critic" and we have

$$\nabla_\theta J(\theta) = \mathbb{E}_{p_\theta(\zeta)}[(\sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a^{(t)}|s^{(t)}))(\sum_{i=t}^{T} A_\phi(a, s))].$$

Here we show two Advantage Actor-Critic algorithms as Algorithm 6 and Algorithm 7.

Algorithm 6 Advantage-MC-Actor-Critic($\pi_\theta, Q_\phi, V_\psi, \alpha, \beta, \kappa$)

for $e = 1, ..., E$ **do**
  $\{s^{(t)}, a^{(t)}, r^{(t)}\}_{t=0}^{T} \sim \mathcal{E}|\pi_\theta$
  for $t = 0, ..., T$ **do**
    $G^{(t)} \leftarrow \sum_{i=t}^{T} r^{(i)}$
    $\psi \leftarrow \psi + \kappa[G^{(t)} - V_\phi(s^{(t)})]\nabla_\psi V_\psi(s^{(t)})$
    $\phi \leftarrow \phi + \beta[G^{(t)} - Q_\phi(a^{(t)}, s^{(t)})]\nabla_\phi Q_\phi(a^{(t)}, s^{(t)})$
    $\theta \leftarrow \theta + \alpha[Q_\phi(a^{(t)}, s^{(t)}) - V_\psi(s^{(t)})]\nabla_\theta \log \pi_\theta(a^{(t)}|s^{(t)})$
  end for
end for
return π_θ

Algorithm 7 Advantage-TD-Actor-Critic($\pi_\theta, Q_\phi, V_\psi, \alpha, \beta, \kappa$)

for $e = 1, ..., E$ **do**
  $\{s^{(t)}, a^{(t)}, r^{(t)}\}_{t=0}^{T} \sim \mathcal{E}|\pi_\theta$
  for $t = 0, ..., T$ **do**
    $\delta_V \leftarrow r^{(t)} + V_\psi(s^{(t+1)})$
    $\psi \leftarrow \psi + \kappa[\delta_V - V_\psi(s^{(t)})]\nabla_\psi V_\psi(s^{(t)})$
    $\delta_Q \leftarrow r^{(t)} + Q_\phi(a^{(t+1)}, s^{(t+1)})$
    $\phi \leftarrow \phi + \beta[\delta_Q - Q_\phi(a^{(t)}, s^{(t)})]\nabla_\phi Q_\phi(a^{(t)}, s^{(t)})$
    $\theta \leftarrow \theta + \alpha[Q_\phi(a^{(t)}, s^{(t)}) - V_\psi(s^{(t)})]\nabla_\theta \log \pi_\theta(a^{(t)}|s^{(t)})$
  end for
end for
return π_θ

References

[1] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[2] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

3 Appendix

Interestingly, the term 'REINFORCE' comes from the abbreviation of

REward **I**ncrement = **N**on-negative **F**actor × **O**ffset **R**einforcement × **C**haracteristic **E**ligibility.

as shown in Figure 1.

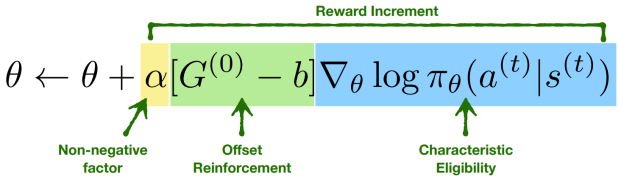

Figure 1: 'REINFORCE'

Figure 2 shows the total reward of REINFORCE without baseline performs with different step size and 3 shows reduced variance leads to faster convergence [1].

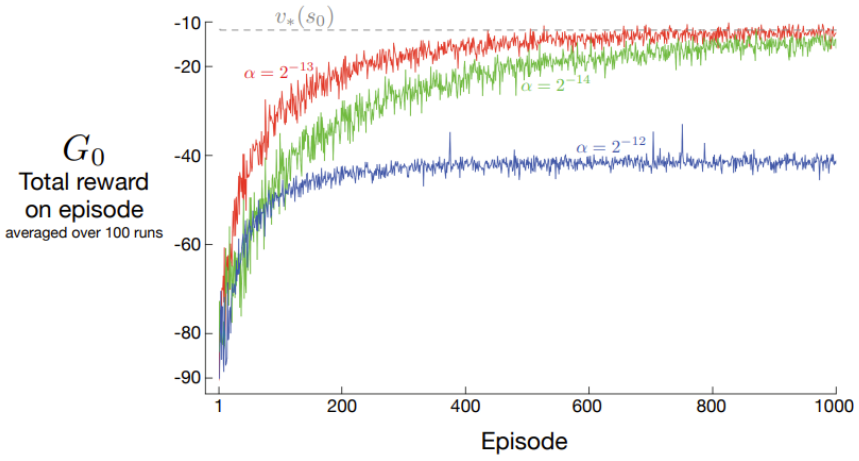

Figure 2: With a good step size, the total reward per episode approaches the optimal value of the start state, shown in [1].

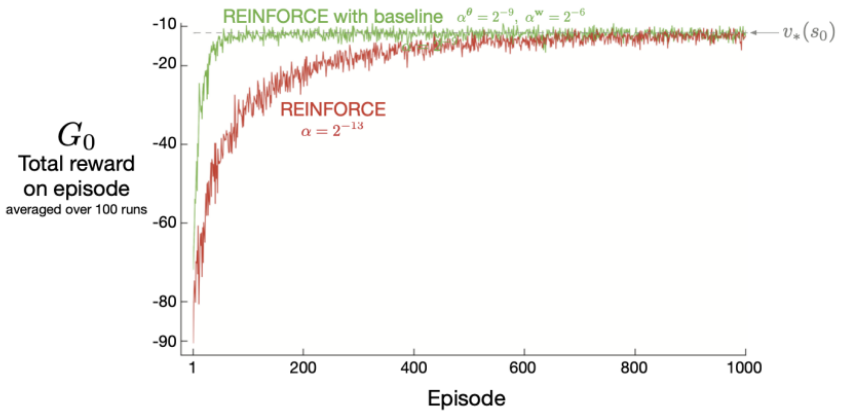

Figure 3: Adding a baseline to REINFORCE can make it learn much faster, shown in [1].