Policy Gradients

Andreas Orthey

The core idea behind policy gradients is to parameterize the policy π (the mapping from states to actions) using a parameter vector θ. This is advantageous for continuous state spaces and for problems with large state and action spaces. The policy π_θ can be thought of as a functional approximation to the policy. An example of a parameterized policy would be a linear weighting of basis functions like $\pi_\theta = \theta^T F(s,a)$ whereby $F(s,a)$ is a set of basis functions [Basis function(2018)]. Another option would be the weights of a neural network. In any case, our goal is to select the best policy by selecting the best parameter vector.

To select the best parameter vector, we first need to define a cost function over policies. We call this cost function $J(\pi_\theta)$. While many cost functions are possible, we like to select the cost function which maximizes our expected reward over policies. The concrete cost function we choose here is the cumulative reward over trajectories as

$$J(\pi_\theta) = \mathbb{E}_{\xi \sim P_{\pi_\theta}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right], \tag{1}$$

whereby γ is the discount factor, $R(s,a)$ is the reward, ξ is a trajectory $\xi = (s_{0:T+1}, a_{0:T})$ (which is the set of states and actions we reach if we play a certain policy) and P_{π_θ} is the probability distribution over trajectories (think about it as a set from which we can sample). To evaluate a policy, we thereby evaluate trajectories. Each trajectory gives us a certain cumulative return. The trick with policy gradients is to differentiate our cost function with respect to θ. This differentiation gives us the steepest descent in parameter space (and thereby in policy space). What we can do now is to follow this steepest descent until we converge to a local minimal solution. Concrete, this means that we make small steps in the steepest descent direction as

$$\theta' \leftarrow \theta - \alpha \nabla_\theta J(\pi_\theta), \tag{2}$$

with α being the step size and $\nabla_\theta J(\pi_\theta)$ being the derivative of the cost function with respect to θ. We continue this procedure until we converge (measured by error between θ' and θ) to a local minimum (or fixpoint) of the parameter landscape, which minimizes the cost function $J(\pi_\theta)$. The major difficulty in this approach is to evaluate efficiently the derivative of the cost function.

1 Derivation of Policy Gradients

Our goal is therefore to differentiate the cost function $J(\pi_\theta)$. To do this, we need to find a numerical representation of the gradient $\nabla_\theta J(\pi_\theta)$ which we can compute. One possible way is to rewrite the gradient in the following way (each step is explained in detail below):

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta \, \mathbb{E}_{\xi \sim P_{\pi_\theta}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] && \text{(Definition)} \\
&= \nabla_\theta \int_\xi P_{\pi_\theta}(\xi) R(\xi) d\xi && \text{(Expand the Expectation)} \\
&= \int_\xi P_{\pi_\theta}(\xi) \nabla_\theta \log P_{\pi_\theta}(\xi) R(\xi) d\xi && \text{(Log-Derivative Trick)} \\
&= \mathbb{E}_{\xi|\theta} \{ \nabla_\theta \log P_{\pi_\theta}(\xi) R(\xi) \} && \text{(Return to Expectation)} \\
&= \mathbb{E}_{\xi|\theta} \{ \sum_{t=0}^{T_\xi} \nabla_\theta \log \pi(a_t|s_t) R(\xi) \} && \text{(Grad-log-prob)}
\end{aligned}$$

Before continuing with detailed descriptions of each step, please note that the last expression of the gradient can be numerically calculated. To do that, you would take your start policy π and follow it until termination. At each step, you compute the log and gradient of the policy and multiply by the reward you obtain. Doing this iteratively, gives you a way to (locally) minimize the cost over policies by applying gradient descent as in Eq. 2.

We now will discuss each of the five steps in the derivation. The first step is the definition, which was already motivated in the last section. We thus need to detail the next four steps.

1.1 Expand the Expectation

In the second step, we expand the expectation. To do this, we first make the probability distribution over trajectories explicit as

$$P_\pi(\xi) = P(s_0) \prod_{t=0}^{T} \pi(a_t, s_t) P(s_{t+1} \mid s_t, a_t). \tag{3}$$

Note that in the non-stochastic case, every policy has a unique trajectory which provides a unique cumulative return (cost). However, in the stochastic case, we need to compute the probability distributions over trajectories and estimate the expected return.

In the expansion of the expectation, we next take the sum (integral) over all possible trajectories and weigh their cumulative return by their probability of occurrence under the application of the policy. A trajectory which is likely to occur will count more than a trajectory which is less likely to occur.

1.2 Log-Derivative Trick

Next, we bring the gradient inside the integral. Since $R(\xi)$ does not depend on θ, we can apply the gradient to the probability distribution. We then apply the log-derivative trick which uses the logarithm identity from calculus: $\nabla_x \log(f(x)) = \frac{\nabla_x f(x)}{f(x)}$ [Logarithmic derivative(2018)]. If we put in P_{π_θ} and rearrange we get

$$\nabla_\theta P_\pi(\xi) = P_\pi(\xi) \nabla_\theta \log P_\pi(\xi). \tag{4}$$

The reason for this rearrangement is that the gradient of the logarithm is often much easier to compute.

1.3 Return to Expectation

In the next step, we return again from the integral representation to the expectation.

1.4 Grad-log-prob

Finally, we rewrite the gradient logarithm as

$$\nabla_\theta \log P_\pi(\xi) = \nabla_\theta \log P(s_0) + \sum_{t=0}^{T} (\nabla_\theta \log P(s_{t+1}|s_t, a_t) + \nabla_\theta \log \pi_\theta(a_t|s_t)) \tag{5}$$

$$= \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t), \tag{6}$$

whereby we took advantage of the fact that both $P_(s_0)$ and $P(s_{t+1}|s_t, a_t)$ are independent of θ and therefore evaluate to zero under derivation with respect to θ.

2 Additional Material

Here are some additional resources to learn about policy gradients.

- Good introduction by OpenAI `https://spinningup.openai.com/en/latest/spinningup/rl_intro3.html`
- `https://danieltakeshi.github.io/2017/03/28/going-deeper-into-reinforcement-learning-fundamentals-of-policy-gradients/`
- Helpful material `https://jonathan-hui.medium.com/rl-policy-gradients-explained-9b13b688b146`

References

[Basis function(2018)] Basis function. Basis function — Wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/wiki/Basis_function.

[Logarithmic derivative(2018)] Logarithmic derivative. Logarithmic derivative — Wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/wiki/Logarithmic_derivative.