Lecture 18

Model-Free Policy Search

In the previous chapter, we discussed sample-based trajectory optimization methods that leverage a model to generate policies without gradients. These approaches can be also applied to physical systems, but many times they are not sample efficient. Other Model-Free Policy Search approaches attempt to approximate the gradients directly. Like the gradient-free approaches, these methods have similar challenges with sample efficiency.

18.1. Likelihood Ratio Gradient

As before, our objective is to minimize the expectation of scalar value (e.g., a cost):

$$\min_{\theta} \mathbb{E}[g(x)]. \tag{18.1}$$

Assume that $x \sim p_\theta(x)$ (i.e., no dynamics). Let's take the gradient of the expected cost with respect to the parameters θ. Using LOTUS, we have:

$$\frac{\partial \mathbb{E}[g(x)]}{\partial \theta} = \frac{\partial}{\partial \theta} \int p_\theta(x) g(x) dx. \tag{18.2}$$

The main insight for this approach is to work with logarithms. Using the property

$$y = \log u \tag{18.3}$$

$$\frac{\partial y}{\partial x} = \frac{\partial u}{\partial x} \frac{1}{u} \tag{18.4}$$

we can write

$$\frac{\partial \log[p_\theta(x)]}{\partial \theta} = \frac{1}{p_\theta(x)} \frac{\partial p_\theta(x)}{\partial \theta} \tag{18.5}$$

and

$$\frac{\partial}{\partial \theta} \mathbb{E}[g(x)] = \frac{\partial}{\partial \theta} \int dx g(x) p_\theta(x) \tag{18.6}$$

$$= \int dx g(x) \frac{\partial p_\theta(x)}{\partial \theta} \tag{18.7}$$

$$= \int dx g(x) \frac{\partial \log[p_\theta(x)]}{\partial \theta} p_\theta(x) \tag{18.8}$$

$$= \mathbb{E}\left[g(x) \frac{\partial \log[p_\theta(x)]}{\partial \theta} \right]. \tag{18.9}$$

Approximate via Monte Carlo sampling:

$$\frac{\partial}{\partial \theta} \mathbb{E}[g(x)] = \frac{1}{N} \sum_{i=0}^{N} g(x_i) \frac{\partial \log[p_\theta(x_i)]}{\partial \theta}. \tag{18.10}$$

18.2. Likelihood Ratio Policy Gradient

Now consider the case where we want to minimize the following:

$$\min_\theta \mathbb{E}\Big[\sum_{k=0}^{N} g(x_k, u_k)\Big]. \tag{18.11}$$

Computing the gradient, we have

$$\frac{\partial}{\partial\theta}\mathbb{E}\Big[\sum_{k=0}^{N} g(x_k, u_k)\Big] = \sum_{k=0}^{N}\int dx_k du_k g(x_k, u_k)\frac{\partial p_\theta(x_k, u_k)}{\partial\theta} \tag{18.12}$$

$$= \mathbb{E}\Big[\sum_{k=0}^{N} g(x_k, u_k)\frac{\partial \log p_\theta(x_k, u_k)}{\partial\theta}\Big], \tag{18.13}$$

where

$$p_\theta(x_k, u_k) = p(x_0)\Big(\prod_{i=1}^{k} p(x_i|x_{i-1}, u_{i-1})\Big)\Big(\prod_{i=0}^{k} p_\theta(u_i|x_i)\Big). \tag{18.14}$$

Take the logarithm!

$$\log p_\theta(x_k, u_k) = \log p(x_0) + \sum_{i=1}^{k}\log p(x_i|x_{i-1}, u_{i-1}) + \sum_{i=0}^{k}\log p_\theta(u_i|x_i) \tag{18.15}$$

But when we take the expectation of this, only the last term depends on θ, so only the last term remains:

$$\frac{\partial}{\partial\theta}\mathbb{E}\Big[\sum_{k=0}^{N} g(x_k, u_k)\Big] = \mathbb{E}\Big[\sum_{k=0}^{N} g(x_k, u_k)\frac{\partial}{\partial\theta}\sum_{k=0}^{N}\log p_\theta(u_k|x_k)\Big]. \tag{18.16}$$

18.3. Weight Perturbation

Consider the scenario where we want to minimize some scalar function, given as

$$\min_\theta g(\theta). \tag{18.17}$$

You can estimate $\frac{\partial g}{\partial\theta}$ using finite differences as

$$\frac{\partial g}{\partial\theta_i} \approx \frac{g(\theta + \epsilon_i) - g(\theta)}{\epsilon}. \tag{18.18}$$

Parameters can be updated using

$$\hat{\theta}_{i+1} = \hat{\theta}_i - \eta\frac{\partial g}{\partial\theta}. \tag{18.19}$$

However, this requires many samples. Another way to approximate the gradient might be to randomly sample the perturbations to do the "finite difference".

Consider the update

$$\Delta\theta = -\eta[g(\theta + \beta) - g(\theta)]\beta. \tag{18.20}$$

Why does this work? Assume the function g is smooth and can be approximated as

$$g(\theta + \beta) \approx g(\theta) + \frac{\partial g}{\partial\theta}\beta. \tag{18.21}$$

$$\Delta(\theta) \approx -\eta[\frac{\partial g}{\partial \theta}\beta]\beta = -\eta\beta\beta^T\frac{\partial g}{\partial \theta}^T \tag{18.22}$$

and

$$\mathbb{E}[\Delta\theta] = -\eta\mathbb{E}\left[\beta\beta^T\right]\frac{\partial g}{\partial \theta}^T. \tag{18.23}$$

If the entries of β are independent, $\mathbb{E}\left[\beta\beta^T\right] = \sigma_\beta^2 I$, where $\mathbb{E}[\beta_i\beta_j] = \sigma_\beta^2\delta_{ij}$.

Can we do even better in terms of sample complexity? What if we use a "baseline" b for approximating the cost at θ?

$$\Delta\theta = \frac{\eta}{\sigma_\beta^2}[g(\theta+\beta) - b]\beta \tag{18.24}$$

Consider a simple form for the estimator given as:

$$b^{k+1} = \gamma g^k + (1-\gamma)b^k, \quad b^0 = 0, 0 \leq \gamma \leq 1. \tag{18.25}$$

Now compute the expected value of the update:

$$\mathbb{E}[\Delta\theta] = -\frac{\eta}{\sigma_\beta^2}\mathbb{E}[(g(\theta) + \frac{\partial g}{\partial \theta}\beta - b)\beta] \tag{18.26}$$

$$= -\frac{\eta}{\sigma_\beta^2}\mathbb{E}[(g(\theta) - b)\beta] - \frac{\eta}{\sigma_\beta^2}\mathbb{E}[\beta\beta^T]\frac{\partial g}{\partial \theta}^T \tag{18.27}$$

$$= -\eta\frac{\partial g}{\partial \theta}^T \tag{18.28}$$

The baseline doesn't impact the fact that the updates perform gradient descent on average. In fact, the above algorithm is a version of the policy gradient algorithm, where the mean of the policy is linear in θ, there is a fixed diagonal covariance, and a single sample is used to estimate the policy gradient.

18.4. Natural Policy Gradient

Traditional steepest descent can be written as

$$\min_{\Delta\theta} J(\theta + \Delta\theta) \quad \text{s.t.} \quad ||\Delta\theta|| \leq \epsilon \tag{18.29}$$

The metric used in gradient descent is $\sqrt{\Delta\theta^T\Delta\theta}$. If we want to use a more general metric, we could write

$$\min_{\Delta\theta} J(\theta + \Delta\theta) \quad \text{s.t.} \quad \Delta\theta^T M(\theta)\Delta\theta \leq \epsilon. \tag{18.30}$$

To enforce this metric, we can use Lagrange multipliers and use the Lagrangian:

$$\min_{\Delta\theta,\lambda} L(\Delta\theta, \lambda) \tag{18.31}$$

where

$$L(\Delta\theta, \lambda) = J(\theta + \Delta\theta) + \lambda(\Delta\theta^T M(\theta)\Delta\theta - \epsilon) \tag{18.32}$$

and $\lambda \geq 0$. If we approximate $L(\Delta\theta, \lambda)$ using a first order Taylor expansion of J, we have

$$L(\Delta\theta, \lambda) \approx J(\theta) + \Delta\theta^T\nabla_\theta J + \lambda(\Delta\theta^T M(\theta)\Delta\theta - \epsilon). \tag{18.33}$$

Find the minimum:

$$\frac{\partial L}{\partial \Delta\theta} = \nabla_\theta J + 2\lambda M(\theta)\Delta\theta = 0. \tag{18.34}$$

We can then write

$$\Delta\theta = -\frac{1}{2\lambda} M^{-1}(\theta) \nabla_\theta J. \tag{18.35}$$

Since M could be singular due to over-parameterization, we often use the pseudo-inverse to find $\Delta\theta$. Oftentimes, the *Fisher Information Metric* is used for M such that

$$M(\theta) = \mathbb{E}[\nabla_\theta \log(p_\theta) \nabla_\theta \log(p_\theta)^T]. \tag{18.36}$$

The Fisher Information Metric happens to be the second order approximation of the KL divergence.

$$KL(p||q) = \sum_{x \in X} p(x) \log\left(\frac{p(x)}{q(x)}\right) \tag{18.37}$$

and

$$KL(p_{\theta+\Delta\theta}||p_\theta) \approx \Delta\theta^T M(\theta) \Delta\theta. \tag{18.38}$$

where $M(\theta)$ is the Fisher Information Metric. For a policy $\pi_\theta(a|s)$ we have

$$M(\theta) = \mathbb{E}[\nabla_\theta \log(\pi_\theta(a|s)) \nabla_\theta \log(\pi_\theta(a|s)^T] \tag{18.39}$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \left[\nabla_\theta \log(\pi_\theta(a_i|s_i)) \nabla_\theta \log(\pi_\theta(a_i|s_i))^T\right]. \tag{18.40}$$

The intuition is that you move the policy in the direction that improves the policy the most, but changes the parameters the least. If you are using a neural network for your policy, you can use something like the *softmax* function on the outputs of the final layer to output a proper probability distribution.

Bibliography

[1] Russ Tedrake. *Underactuated Robotics*. 2023.

[2] Dimitri Bertsekas. *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.

[3] Drew Bagnell, Byron Boots, and Sanjiban Choudhury. *Modern Adaptive Control and Reinforcement Learning*. 2022.