Some notes on Policy Gradients for Reinforcement Learning

David Meyer

dmm@{1-4-5.net,uoregon.edu,...}

Last update: September 11, 2017

1 Introduction

Policy gradients for Reinforcement Learning (RL) are really beautiful in form and function. In these notes I'll try to point out a few of these amazing features[1].

Here we consider methods that learn a parameterized policy that can select actions without consulting a value function. A value function may still be used to learn the policy weights, but that is not required for action selection. The notation $\theta \in \mathbb{R}^n$ is typically used for the primary learned weight vector, and $\pi(a \mid s, \theta) = Pr(A_t = a \mid s_t = s, \theta_t = \theta)$ is the probability that action a is taken at time t given that the agent is in state s at time t with weight vector θ. If a method uses a learned value function as well, then the value function's weight vector is denoted w to distinguish it from θ, such as in $\hat{v}(s, w)$.

We wish to consider methods for learning the policy weights θ based on the gradient of some performance measure $\eta(\theta)$) with respect to the policy weights. These methods seek to maximize performance, so their updates approximate gradient ascent in η. As is typical for gradient descent/ascent,

$$\boldsymbol{\theta}_{t+1} \coloneqq \boldsymbol{\theta}_t + \alpha \widehat{\nabla \eta(\boldsymbol{\theta}_t)} \tag{1}$$

where, as is typical with gradient ascent/decent methods, $\widehat{\nabla \eta(\boldsymbol{\theta}_t)}$ is a stochastic estimate whose expectation approximates the gradient of the performance measure η with respect to its argument θ_t. As we will see, $\widehat{\nabla \eta(\boldsymbol{\theta}_t)}$ will turn out to be something like $\nabla_\theta \mathbb{E}_\tau[R(\tau)]$, where τ is a *trajectory* and $R(\tau)$ is the return for path τ (see Section 2).

[1] Here we follow the notation used in [1].

2 Score Function/Likelihood Ratio Estimators

First notices that Likelihood Ratio methods only change the probabilities of experienced paths, and further, these methods do not try to change the actions taken in a given path. That said, recall that our problem here is to solve the following:

$$\max_{\theta} \mathbb{E}[\sum_{t=0}^{H} R(s_t) \mid \pi_\theta] \tag{2}$$

where $\pi_\theta(u \mid s)$ is the probability of action u in state s. Now, because ultimately we'll want to compute the gradient $\nabla_\theta \mathbb{E}[R \mid \pi_\theta]$, we need to prove one identity, namely (imagine that $f(x) = R$)

$$\nabla_\theta \mathbb{E}_{x \sim p(x|\theta)}[f(x)] = \mathbb{E}_x[f(x) \nabla_\theta \log p(x \mid \theta)] \tag{3}$$

So the proof of this is amazingly simple given the log derivative trick[2]. First, consider a function $f(x)$ for which we wish to find $\nabla_\theta \mathbb{E}_{x \sim p(x|\theta)}[f(x)]$. Then

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{x \sim p(x|\theta)}[f(x)] &= \nabla_\theta \int_x p(x \mid \theta) f(x) dx && \text{\# definition} && (4)\\
&= \int_x \nabla_\theta p(x \mid \theta) f(x) dx && \text{\# Lebesgue Integral} && (5)\\
&= \int_x p(x \mid \theta) \frac{\nabla_\theta p(x \mid \theta)}{p(x \mid \theta)} f(x) dx && \text{\# multiply by } \frac{p(x \mid \theta)}{p(x \mid \theta)} && (6)\\
&= \int_x p(x \mid \theta) \nabla_\theta \log p(x \mid \theta) f(x) && \text{\# log-derivative trick} && (7)\\
&= \mathbb{E}_{x \sim p(x|\theta)}[f(x) \nabla_\theta \log p(x \mid \theta)] && \text{\# defn expectation} && (8)
\end{aligned}$$

Define $\hat{g}_i = f(x_i) \nabla_\theta \log p(x_i \mid \theta)$. An empirical estimate the gradient for m samples is

$$\nabla_\theta \mathbb{E}_{x \sim p(x|\theta)}[f(x)] = g \approx \frac{1}{m} \sum_{i=1}^{m} \hat{g}_i = \frac{1}{m} \sum_{i=1}^{m} f(x_i) \nabla_\theta \log p(x_i \mid \theta) \tag{9}$$

Aside: The "log derivative trick", which is also called the "likelihood ratio trick" since $\frac{\nabla_\theta p(x|\theta)}{p(x|\theta)}$ is called the "likelihood ratio" and $\frac{\nabla_\theta p(x|\theta)}{p(x|\theta)} = \nabla_\theta \log p(x \mid \theta)$, is sometimes framed

[2] `http://blog.shakirm.com/2015/11/machine-learning-trick-of-the-day-5-log-derivative-trick`

up like this: The gradient of something divided by something is the gradient of log something. Also frequently pronounced "grad something divided by something is grad log something".

So $\nabla_\theta E_{x\sim p(x|\theta)}[f(x)] = E_{x\sim p(x|\theta)}[f(x)\nabla_\theta \log p(x \mid \theta)]$. This gives us an unbiased gradient estimator; to compute the gradient estimate, just sample $x_i \sim p(x \mid \theta)$ and then compute the gradient estimate $\hat{g}_i = f(x_i)\nabla_\theta \log p(x_i \mid \theta)$. Now, let the trajectory (sometimes path) τ be the state-action sequence $(s_0, a_0, r_0, s_1, a_1, r_1, \cdots, a_{T-1}, a_{T-1}, r_{T-1})$ and suppose $f(x) = R(\tau)$, the total return for *path* τ. Then

$$\nabla_\theta \mathbb{E}_\tau[R(\tau)] = E_\tau[\nabla_\theta \log p(\tau \mid \theta) R(\tau)] \tag{10}$$

so that all we really need to do is see that we can compute $p(\tau \mid \theta)$:

$$p(\tau \mid \theta) = \mu_0(s_0) \prod_{t=0}^{T-1} \underbrace{\pi(a_t \mid s_t, \theta)}_{\text{policy}} \underbrace{P(s_{t+1}, r_t) \mid s_t, a_t)}_{\text{dynamics}} \qquad \#\text{definition} \tag{11}$$

$$\log p(\tau \mid \theta) = \log \mu_0(s_0) + \sum_{t=0}^{T-1} \log \pi(a_t \mid s_t, \theta) + \log P(s_{t+1}, r_t) \mid s_t, a_t) \tag{12}$$

$$\nabla_\theta \log p(\tau \mid \theta) = \nabla_\theta \sum_{t=0}^{T-1} \log \pi(a_t \mid s_t, \theta) \tag{13}$$

$$= \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \tag{14}$$

So $\nabla_\theta \mathbb{E}_\tau[R] = \mathbb{E}_\tau[R(\tau)\nabla_\theta \sum_{t=0}^{T-1} \log \pi(a_t \mid s_t, \theta)]$, which gives a direct way of computing a gradient we can use with standard Stochastic Gradient Ascent (or Descent), where the parameter update rule would be something like $\theta \coloneqq \theta + \alpha \nabla_\theta \mathbb{E}_\tau[R(\tau)]$. Essentially move θ in the direction of better expected return. Note also that the model dynamics, $P(s_{t+1}, r_t) \mid s_t, a_t)$, conveniently drops out of the gradient (good thing given that we're trying to learn model-free control).

3 Derivation of the Gradient via Importance Sampling

It turns out that almost all on-policy control methods utilize importance sampling, which is a general technique for estimating expected values under one distribution given samples

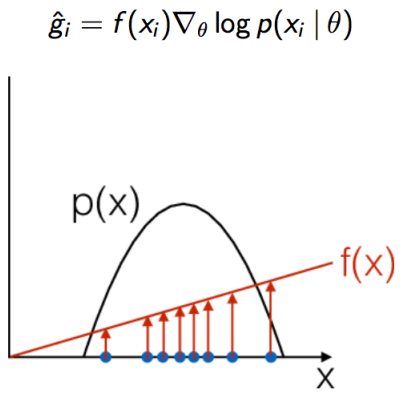

Figure 1: Score/likelihood Gradient Estimator Intuition 1. Image courtesy John Schulman `https://learning.mpi-sws.org/mlss2016/slides/2016-MLSS-RL.pdf`

from another[1]. The basic approach to applying importance sampling to on-policy learning is by weighting returns according to the relative probability of their trajectories occurring under the target and behavior policies; this is called the importance-sampling ratio. Now given a starting state S_t, the probability of the subsequent state-action trajectory $\tau = A_t, S_{t+1}, A_{t+1} \cdots, S_T$ occurring under any policy π is

$$p_t(\tau) = \prod_{k=t}^{T-1} \pi(A_k \mid S_k) p(S_{k+1} \mid S_k, A_k) \tag{15}$$

where p is the state-transition probability function. The relative probability of the trajectory under the target and behavior policies (the importance-sampling ratio) is then

$$\rho_t^T = \frac{\prod_{k=t}^{T-1} \pi(A_k \mid S_k) p(S_{k+1} \mid S_k, A_k)}{\prod_{k=t}^{T-1} u(A_k \mid S_k) p(S_{k+1} \mid S_k, A_k)} = \frac{\prod_{k=t}^{T-1} \pi(A_k \mid S_k)}{\prod_{k=t}^{T-1} u(A_k \mid S_k)} \tag{16}$$

One of the many interesting things here is that it turns out that ρ_t^T doesn't depend on the MDP's transition probability function (which is what we want since we want to solve the model-free problem). In the following example, consider $\rho_t^T = \frac{p(x|\theta)}{p(x|\theta_{old})}$.

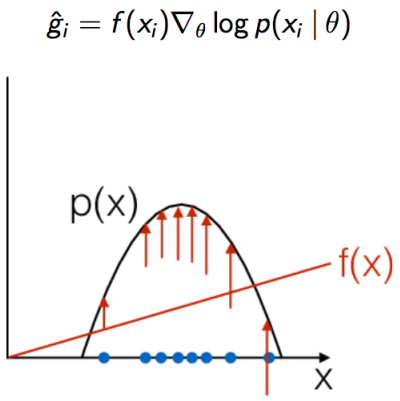

Figure 2: Score/likelihood Gradient Estimator Intuition 2. Image courtesy John Schulman `https://learning.mpi-sws.org/mlss2016/slides/2016-MLSS-RL.pdf`

$$\mathbb{E}_{x\sim\theta}\big[f(x)\big] = \mathbb{E}_{x\sim\theta_{old}}\left[\frac{p(x \mid \theta)}{p(x \mid \theta_{old})} f(x)\right] \tag{17}$$

$$\nabla_\theta \mathbb{E}_{x\sim\theta}\big[f(x)\big] = \mathbb{E}_{x\sim\theta_{old}}\left[\frac{\nabla_\theta p(x \mid \theta)}{p(x \mid \theta_{old})} f(x)\right] \tag{18}$$

$$\nabla_\theta \mathbb{E}_{x\sim\theta}\big[f(x)\big]\big|_{\theta=\theta_{old}} = \mathbb{E}_{x\sim\theta_{old}}\left[\frac{\nabla_\theta p(x \mid \theta)\big|_{\theta=\theta_{old}}}{p(x \mid \theta_{old})} f(x)\right] \tag{19}$$

$$= \mathbb{E}_{x\sim\theta_{old}}\left[\nabla_\theta \log p(x \mid \theta)\big|_{\theta=\theta_{old}} f(x)\right] \tag{20}$$

4 A Bit of Intuition

What we've learned so far is that we can compute the an estimate of the gradient for sample i is $\hat{g}_i = f(x_i)\nabla_\theta \log p(x_i \mid \theta)$. Now, suppose $f(x)$ somehow measures how good the sample x is. Then we can conclude a few things about $\hat{g}_i$:

- Moving in the direction $\hat{g}_i$ pushes up the log probability ("logprob") of the sample in proportion to how good it is (i.e., $f(x_i)$), remembering that the SGD update rule is

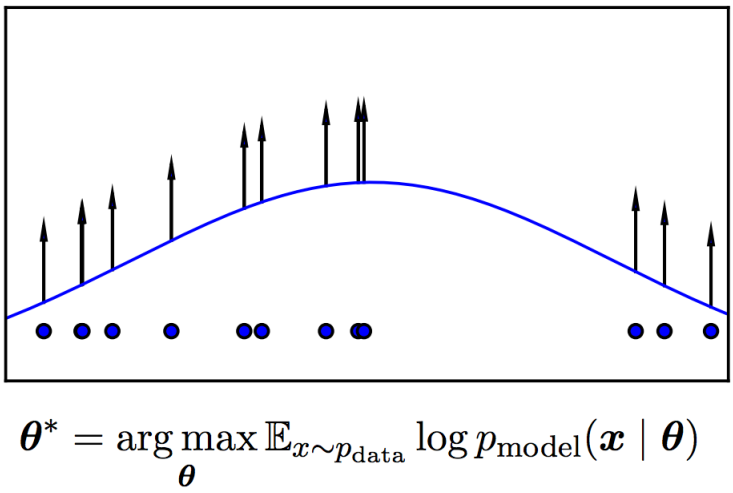

Figure 3: Maximum Likelihood Estimation. Image courtesy Ian Goodfellow [2].

something like $\theta \coloneqq \theta + \alpha \nabla_\theta \mathbb{E}_\tau[R(\tau)]$. This pushing up is depicted in Figures 1 and 2.

- Note that this is true even if $f(x)$ is discontinuous, and unknown, or the sample space (containing x) is a discrete set!
- More generally, Maximum Likelihood Estimation (MLE) consists of taking several samples from the Data Generating Distribution (DGD) to form a training set, then pushing up on the probability the model assigns to those points, thus maximizing the likelihood of the training data. Figure 3 shows how different data points push up on different parts of the density function for a Gaussian model applied to 1-D data. The fact that the density function must sum to one means that we cannot simply assign infinite likelihood to all points; as one point pushes up in one place it inevitably pulls down in other places. The resulting density function balances out the upward forces from all the data points in different locations.

Basically, the gradient tries to Increase probability of paths with positive return R and decrease the probability of paths with negative return.

4.1 Reorganizing Equation 10

Equation 10 can be rewritten as follows:

$$\nabla_\theta \mathbb{E}_\tau[R(\tau)] = E_\tau[\nabla_\theta \log p(\tau \mid \theta) R(\tau)] \tag{21}$$

$$= \mathbb{E}\left[\left(\sum_{t=0}^{T-1} r_t\right)\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta)\right)\right] \tag{22}$$

But notice that we can limit the trajectory to t steps and derive gradient estimator for one reward term $r_{t'}$. In particular

$$\nabla_\theta \mathbb{E}\big[r_{t'}\big] = \mathbb{E}\left[r_{t'} \sum_{t=0}^{t} \nabla_\theta \log \pi(a_t \mid s_t, \theta)\right] \tag{23}$$

Now, we can sum over t to get

$$\nabla_\theta \mathbb{E}\big[R\big] = \mathbb{E}\left[\sum_{t'=0}^{T-1} r_{t'} \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta)\right] \tag{24}$$

$$= \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \sum_{t'=t}^{T-1} r_{t'}\right] \tag{25}$$

5 Adding a Baseline: Deriving an Unbiased Estimator

Here's the problem: suppose that $f(x) \geq 0 \; \forall x$. Then for every x_i, the gradient estimator $\hat{g}_i$ tries to push up up its density (recall that $\boldsymbol{\theta}_{t+1} \coloneqq \boldsymbol{\theta}_t + \alpha\widehat{\nabla\eta(\boldsymbol{\theta}_t)}$). Better would be to have a (new) unbiased estimator that avoids this problem; this estimator would only push up the density for better-than-average x_i. What would this look like?

$$\nabla_\theta \mathbb{E}_x\big[f(x)\big] = \nabla_\theta \mathbb{E}_x\big[(f(x) - b)\big] \tag{26}$$

$$= \mathbb{E}_x\big[\nabla_\theta \log p(x \mid \theta)(f(x) - b)\big] \tag{27}$$

Note that a near optimal choice for b is always $\mathbb{E}\big[f(x)\big]$; however $\mathbb{E}\big[f(x)\big]$ must itself be estimated.

To derive our estimate with baseline, recall that

$$\nabla_\theta \mathbb{E}_\tau\big[R\big] = \left[\sum_{t'=0}^{T-1} r_{t'} \sum_{t=t'}^{t'} \nabla_\theta \log \pi(a_t \mid s_t, \theta)\right] \tag{28}$$

$$= \mathbb{E}_\tau\left[\sum_{t=0}^{T-1} \log \pi(a_t \mid s_t, \theta)\left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)\right] \tag{29}$$

for any "baseline" function $b : \mathcal{S} \to \mathbb{R}$. See Schulman, J. , et al. [3] for details of the proof.

References

[1] R. S. Sutton and A. G. Barto., *Reinforcement learning: An introduction*, vol. 1. MIT Press, 1998.

[2] I. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," 01 2017.

[3] J. Schulman, N. Heess, T. Weber, and P. Abbeel, "Gradient estimation using stochastic computation graphs," 06 2015.