Stochastic Gradient Estimation

Michele Garibbo

1 Overview

Here, I discuss the two main approaches/tricks to stochastic gradient estimation in the machine learning literature, 1) The log-trick and 2) the Reparameterization-trick. The key insight behind both approaches is to re-write the gradient of the problematic expectation as the expectation of the gradient. As a result, we can then use a Monte Carlo estimate of the expectation of the gradient (i.e., by computing the gradient at different samples). Before introducing the two approaches, I briefly show under what circumstances computing the gradient of an expectation is problematic.

2 Introduction of the problem

To start, assume you need optimise the following objective relative to the parameters θ:

$$J(\theta) = \mathbb{E}_{p_X}[f_\theta(x)] \tag{1}$$

To optimise this objective, we can compute its gradient relative to the parameters we want to optimise:

$$\nabla_\theta J = \nabla_\theta \mathbb{E}_{p_X}[f_\theta(x)] \tag{2}$$

we can then expand this to:

$$= \nabla_\theta \int p_X(x)\ f_\theta(x)\ dx \tag{3}$$

under certain conditions we can re-write this as (Leibniz integral rule):

$$= \int \nabla_\theta\ p_X(x)\ f_\theta(x)\ dx \tag{4}$$

since p_X doesn't depend on θ we have:

$$= \int p_X(x)\ \nabla_\theta f_\theta(x)\ dx \tag{5}$$

$$= \mathbb{E}_{p_X}[\nabla_\theta f_\theta(x)] \tag{6}$$

We have transformed the gradient of an expectation into the expectation of a gradient, which is great. This is great because we can now sample gradients of f_θ to estimate this expectation through Monte Carlo estimates.
Now let's see what happens in case the underlying distribution p_X also depends on θ,

$$\nabla_\theta J = \nabla_\theta \mathbb{E}_{p_\theta(x)}[f_\theta(x)] \tag{7}$$

$$= \int \nabla_\theta[\ p_\theta(x)\ f_\theta(x)]\ dx \tag{8}$$

$$= \int \nabla_\theta p_\theta(x)\ f_\theta(x)\ dx\ + \int p_\theta(x)\ \nabla_\theta f_\theta(x)\ dx \tag{9}$$

Now we can see the issue: $\int \nabla_\theta p_\theta(x)\ f_\theta(x)\ dx$. If the integral cannot be solved analytically, we no longer have the option to re-write this integral in terms of an expectation of a gradient, from which we can sample. Two distinct well-known approaches can be used to solve this issue, the "log-trick" and the "reparameterization-trick".

3 The Log-trick

We can multiply the first (problematic) integral term in Eq.9 by $\frac{p_\theta(x)}{p_\theta(x)}$, which gives:

$$\int \nabla_\theta p_\theta(x)\ f_\theta(x)\ dx = \int \frac{p_\theta(x)}{p_\theta(x)} \nabla_\theta p_\theta(x)\ f_\theta(x)\ dx \tag{10}$$

Now we can exploit the gradient of the logarithm function to re-write this as

$$= \int p_\theta(x) \nabla_\theta \log(p_\theta(x))\ f_\theta(x)\ dx \tag{11}$$

$$= \mathbb{E}_{p_\theta(x)}[\nabla_\theta \log(p_\theta(x))\ f_\theta(x)] \tag{12}$$

Now we can put this back with the second integral term in Eq.9 to obtain (i.e. re-joined the two integrals terms into one integral, and expressed it as an expectation):

$$\nabla_\theta \mathbb{E}_{p_\theta(x)}[f_\theta(x)] = \mathbb{E}_{p_\theta(x)}[\nabla_\theta \log(p_\theta(x))\ f_\theta(x) + \nabla_\theta f_\theta(x)] \tag{13}$$

Once again, we transformed a problematic gradient of an expectation into the expectation of a gradient, which we can sample. Note that in case $f(x)$ does not depend on θ, this simplifies to:

$$\nabla_\theta \mathbb{E}_{p_\theta(x)}[f(x)] = \mathbb{E}_{p_\theta(x)}[\nabla_\theta \log(p_\theta(x))\ f(x)] \tag{14}$$

3.1 RL example: REINFORCE

In reinforcement learning, the REINFORCE algorithm is often associated to the log-trick, since this RL algorithm makes use of it. However, REINFORCE also relies on the (key) policy gradient theorem, which enables to greatly simplify the policy gradient computation, before applying the log-trick. Consider the gradient of the objective in RL,

$$\nabla_\theta J = \nabla_\theta \mathbb{E}_{s\sim d^\pi}[V^\pi(s)] \tag{15}$$

where d^π is stationary distribution of the Markov chain induced by π_θ,

$$= \nabla_\theta \mathbb{E}_{s\sim d^\pi}\left[\int_a \pi_\theta(a \mid s) Q^\pi(s,a)\right] \tag{16}$$

$$= \nabla_\theta \int_s d^\pi(s) \int_a \pi_\theta(a \mid s) Q^\pi(s,a) \tag{17}$$

Computing $\nabla_\theta d^\pi(s)$ is virtually impossible since we don't have explicit access to this distribution and it is very hard to determine how a change in the policy may affect this distribution. Fortunately, the policy gradient theorem shows we can re-write this complex gradient as (e.g., see my notes available at link),

$$\approx \int_s d^\pi(s) \int_a Q^\pi(s,a) \nabla_\theta \pi_\theta(a \mid s) \tag{18}$$

At this stage, we apply the log-trick,

$$= \int_s d^\pi(s) \int_a Q^\pi(s,a) \nabla_\theta \pi_\theta(a \mid s) \frac{\pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \tag{19}$$

which enables us to write the expectation of the gradient also over the policy (by also merging the two integrals),

$$\mathbb{E}_{s\sim d^\pi; a\sim\pi_\theta}[Q^\pi(s,a) \nabla_\theta \log(\pi_\theta(a \mid s))] \tag{20}$$

4 The Reparameterization-trick

Consider again the objective:

$$\nabla_\theta J = \nabla_\theta \mathbb{E}_{p_\theta(x)}[f_\theta(x)] \tag{21}$$

To compute this through another approach, we can reparametrize x (more on this later) such that x becomes a deterministic function of θ and the stochasticity comes from a separate random variable ϵ, independent of θ.

$$= \nabla_\theta \mathbb{E}_{p_{(\epsilon)}}[f_\theta(g(\epsilon, \theta))] \tag{22}$$

In this way, the underlying distribution $p()$, no longer depends on θ so that we can bring the gradient operator inside the expectation as we did in the first example where the expectation did not depend on θ, Eq.6

$$= \mathbb{E}_{p_{(\epsilon)}}[\nabla_\theta f_\theta(g(\epsilon, \theta))] \tag{23}$$

While the "log-trick" can be applied under most conditions (i.e. p_θ has to be a continuous function of θ), the "reparameterization-trick" requires the underlying distribution p_θ to be reparametrized into a distribution that we can sample from and that's independent of θ. To better grasp this, let's see how the reparameterization-trick is actually computed. The reparameterization-trick relies on the change of variable.

$$x \sim p_X(x) \rightarrow \epsilon = h(x) \qquad\qquad x = g(\epsilon) \text{ where } g = h^{-1} \tag{24}$$

$$p_\epsilon(\epsilon) = p_X(g(\epsilon))|\frac{dg}{d\epsilon}| \tag{25}$$

4.1 Example

Assume we want to reparametrize a Gaussian distribution, $p_X(x) = \mathcal{N}(x \mid \mu, \sigma)$ to the variable $\epsilon = \frac{x-\mu}{\sigma} = h(x)$ (i.e. a unit Gaussian), which implies $x = \mu + \epsilon\sigma = g(\epsilon)$. Based on Eq.25, we have $p_\epsilon(\epsilon) = p_X(\mu + \epsilon\sigma)\ \sigma$ since $\frac{dg}{d\epsilon} = \sigma$ and σ is always positive. This gives us:

$$p_X(\mu + \epsilon\sigma)\ \sigma = \frac{\sigma}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2}\ \frac{(\mu + \epsilon\sigma - \mu)^2}{2\sigma^2}\right\} \tag{26}$$

Which simplifies to:

$$= \mathcal{N}(\epsilon \mid 0, 1) \tag{27}$$

This suggests that we can sample $\epsilon \sim \mathcal{N}(0, 1)$ and then transforms it back to x based on $x = \mu + \epsilon\sigma$. This is great since sampling ϵ from $\mathcal{N}(0, 1)$ is super easy and the underlying distribution no longer depends on (μ, σ). For instance, if we have $p_\theta = \mathcal{N}(x \mid \theta, \sigma)$ and we need to optimize:

$$\nabla_\theta \mathbb{E}_{p_\theta(x)}[f(x)] \tag{28}$$

we can re-write this as:

$$\nabla_\theta \mathbb{E}_{p_\epsilon}[f(\theta + \epsilon\sigma)] \tag{29}$$

This is exactly why when we have a Gaussian policy in RL and we are trying to estimate its expected return, we can simply sample actions according to $\mu + \epsilon\sigma$ where $\epsilon \sim \mathcal{N}(0, 1)$ and allow Pytorch to backpropagate trough the sample actions (see example below). However, no all distributions can be reparametrised into simple distribution from which we can easily sample. That is why the log-trick is more applicable.

4.2 RL example: 'DPG' with a stochastic policy

Assume you're in a RL setting where you need to compute the following (stochastic) policy gradient (i.e. equivalent to deterministic policy gradient, but with a stochastic policy):

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[Q(s, a)] = \nabla_\theta \int_a \pi_\theta(a \mid s) Q(s, a) da \tag{30}$$

where θ parameterise your (stochastic) Gaussian policy $\pi = \mathcal{N}(a \mid \mu_\theta, \sigma_\theta)$ (for simplicity, we used short-hand notation $\mu_\theta = \mu_\theta(s)$ and $\sigma_\theta = \sigma_\theta(s)$). Here, we run in the same problem of needing to compute a difficult integral term, which we cannot re-write in terms of an expectation (i.e. $\int_a \nabla_\theta(\pi_\theta(a \mid s) Q(s, a)) da$). As in the example above, we can re-parametrize a such that it becomes a deterministic function of θ. Since, a follows a Gaussian distribution, we can use the same re-parametrization as in the example above, $\epsilon = \frac{a - \mu_\theta}{\sigma_\theta}$ and thus, $a = \mu_\theta + \sigma_\theta \epsilon = g(\epsilon)$. If we apply this change of variable to the Gaussian policy

we get the re-parametrized distribution:

$$p(\epsilon) = \pi_\theta(g(\epsilon) \mid s)|\frac{da}{d\epsilon}| \tag{31}$$

$$= \frac{\sigma_\theta}{\sqrt{2\pi\sigma_\theta^2}} \exp\left\{-\frac{1}{2} \frac{(\mu_\theta + \epsilon\sigma_\theta - \mu_\theta)^2}{2\sigma_\theta^2}\right\} \tag{32}$$

where π denotes the actual quantity and not the policy (apologies for the overloaded notation). Note how all the terms depending on θ cancel out, giving a distribution independent of θ

$$= \mathcal{N}(\epsilon \mid 0, 1) \tag{33}$$

Thanks to this re-parametrisation of the policy, we can re-write:

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[Q(s,a)] = \int_a \mathcal{N}(\epsilon \mid 0,1) \nabla_\theta Q(s, \mu_\theta + \sigma_\theta \epsilon) d\epsilon \tag{34}$$

$$= \mathbb{E}_{p_\epsilon}[\nabla_\theta Q(s, \mu_\theta + \sigma_\theta \epsilon)] \tag{35}$$