Appendix A:
Basics of Monte Carlo

Mathematical Foundations of Deep Neural Networks

Fall 2022

Department of Mathematical Sciences

Ernest K. Ryu

Seoul National University

Monte Carlo

We quickly cover some basic notions of Monte Carlo simulations.

These concepts will be used with VAEs.

These ideas are also extensively used in reinforcement learning (although not a topic of this course).

Monte Carlo estimation

Consider IID data $X_1, \ldots, X_N \sim f$. Let $\phi(X) \geq 0$ be some function*. Consider the problem of estimating

$$I = \mathbb{E}_{X\sim f}[\phi(X)] = \int \phi(x) f(x)\, dx$$

One commonly uses

$$\hat{I}_N = \frac{1}{N} \sum_{i=1}^{N} \phi(X_i)$$

to estimate I. After all, $\mathbb{E}\left[\hat{I}_N\right] = I$ and $\hat{I}_N \to I$ by the law of large numbers.#

*The assumption $\phi(X) \geq 0$ can be relaxed.

#Convergence in probability by weak law of large numbers and almost sure convergence by strong law of large numbers.

Monte Carlo estimation

We can quantify convergence with variance:

$$\mathrm{Var}_{X\sim f}\left(\hat{I}_N\right) = \sum_{i=1}^{N} \mathrm{Var}_{X_i\sim f}\left(\frac{\phi(X_i)}{N}\right) = \frac{1}{N}\mathrm{Var}_{X\sim f}\left(\phi(X)\right)$$

In other words

$$\mathbb{E}\left[\left(\hat{I}_N - I\right)^2\right] = \frac{1}{N}\mathrm{Var}_{X\sim f}\left(\phi(X)\right)$$

and

$$\mathbb{E}\left[\left(\hat{I}_N - I\right)^2\right] \to 0$$

as $N \to \infty$.[#]

[#] So $\hat{I}_n \to I$ in L^2 provided that $\mathrm{Var}_{X\sim f}\left(\phi(X)\right) < \infty$.

Empirical risk minimization

In machine learning and statistics, we often wish to solve

$$\underset{\theta \in \Theta}{\text{minimize}} \quad \mathcal{L}(\theta)$$

where the objective function

$$\mathcal{L}(\theta) = \mathbb{E}_{X \sim p_X}[\ell(f_\theta(X), f_\star(X))]$$

Is the (true) *risk*. However, the evaluation of $\mathbb{E}_{X \sim p_X}$ is impossible (if p_X is unknown) or intractable (if p_X is known but the expectation has no closed-form solution). Therefore, we define the proxy loss function

$$\mathcal{L}_N(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(f_\theta(X_i), f_\star(X_i))$$

which we call the *empirical risk,* and solve

$$\underset{\theta \in \Theta}{\text{minimize}} \quad \mathcal{L}_N(\theta)$$

Empirical risk minimization

This is called *empirical risk minimization* (ERM). The idea is that

$$\mathcal{L}_N(\theta) \approx \mathcal{L}(\theta)$$

with high probability, so minimizing $\mathcal{L}_N(\theta)$ should be similar to minimizing $\mathcal{L}(\theta)$.

Technical note) The law of large numbers tells us that

$$\mathbb{P}(|\mathcal{L}_N(\theta) - \mathcal{L}(\theta)| > \varepsilon) = \mathrm{small}$$

for any given θ, but we need

$$\mathbb{P}\left(\sup_{\theta \in \Theta} |\mathcal{L}_N(\theta) - \mathcal{L}(\theta)| > \varepsilon\right) = \mathrm{small}$$

for all compact Θ in order to conclude that the argmins of the two losses to be similar. These types of results are established by a *uniform law of large numbers*.

Importance sampling

Importance sampling (IS) is a technique for reducing the variance of a Monte Carlo estimator.

Key insight of important sampling:

$$I = \int \phi(x) f(x)\, dx = \int \frac{\phi(x) f(x)}{g(x)} g(x)\, dx = \mathbb{E}_{X\sim g}\left[\frac{\phi(X) f(X)}{g(X)}\right]$$

(We do have to be mindful of division by 0.) Then

$$\hat{I}_N = \frac{1}{N}\sum_{i=1}^{N} \phi(X_i) \frac{f(X_i)}{g(X_i)}$$

with $X_1, \dots, X_N \sim g$ is also an estimator of I. Indeed, $\mathbb{E}[\hat{I}_N] = I$ and $\hat{I}_N \to I$. The weight $\frac{f(x)}{g(x)}$ is called the *likelihood ratio* or the Radon–Nikodym derivative.

So we can use samples from g to compute expectation with respect to f.

IS example: Low probability events

Consider the setup of estimating the probability

$$\mathbb{P}(X > 3) = 0.00135$$

where $X \sim \mathcal{N}(0,1)$. If we use the regular Monte Carlo estimator

$$\hat{I}_N = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\{X_i > 3\}}$$

where $X_i \sim \mathcal{N}(0,1)$, if N is not sufficiently large, we can have $\hat{I}_N = 0$. Inaccurate estimate.

If we use the IS estimator

$$\hat{I}_N = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\{Y_i > 3\}} \exp\left(\frac{(Y_i - 3)^2 - Y_i^2}{2}\right)$$

where $Y_i \sim \mathcal{N}(3,1)$, having $\hat{I}_N = 0$ is much less likely. Estimate is much more accurate.

Importance sampling

Benefit of IS quantified by with variance:

$$\mathrm{Var}_{X\sim g}\left(\hat{I}_N\right) = \sum_{i=1}^{N} \mathrm{Var}_{X\sim g}\left(\frac{\phi(X_i)f(X_i)}{ng(X_i)}\right) = \frac{1}{N}\mathrm{Var}_{X\sim g}\left(\frac{\phi(X)f(X)}{g(X)}\right)$$

If $\mathrm{Var}_{X\sim g}\left(\frac{\phi(X)f(X)}{g(X)}\right) < \mathrm{Var}_{X\sim f}\left(\phi(X)\right)$, then IS provides variance reduction.

We call g the *importance* or *sampling distribution*. Choosing g poorly can increase the variance. What is the best choice of g?

Optimal sampling distribution

The sampling distribution

$$g(x) = \frac{\phi(x)f(x)}{I}$$

makes $\mathrm{Var}_{X\sim g}\left(\frac{\phi(X)f(X)}{g(X)}\right) = \mathrm{Var}_{X\sim g}(I) = 0$ and therefore is optimal. (I serves as the normalizing factor that ensures the density g integrates to 1.)

Problem: Since we do not know the normalizing factor I, the answer we wish to estimate, sampling from g is usually difficult.

Optimized/trained sampling distribution

Instead, we consider the optimization problem

$$\underset{g\in\mathcal{G}}{\text{minimize}} \quad D_{\text{KL}}\left(g \| \frac{\phi f}{I}\right)$$

and compute a suboptimal, but good, sampling distribution within a class of sampling distributions $\mathcal{G}$. (In ML, $\mathcal{G} = \{g_\theta | \theta \in \Theta\}$ is parameterized by neural networks.)

Importantly, this optimization problem does not require knowledge of I.

$$\begin{aligned} D_{\text{KL}}(g_\theta \| \phi f / I) &= \mathbb{E}_{X\sim g_\theta}\left[\log\left(\frac{I g_\theta(X)}{\phi(X) f(X)}\right)\right] \\ &= \mathbb{E}_{X\sim g_\theta}\left[\log\left(\frac{g_\theta(X)}{\phi(X) f(X)}\right)\right] + \log I \\ &= \mathbb{E}_{X\sim g_\theta}\left[\log\left(\frac{g_\theta(X)}{\phi(X) f(X)}\right)\right] + \text{constant independent of } \theta \end{aligned}$$

How do we compute stochastic gradients?

Log-derivative trick

Generally, consider the setup where we wish to solve

$$\underset{\theta \in \mathbb{R}^p}{\text{minimize}} \quad \mathbb{E}_{X \sim f_\theta}[\phi(X)]$$

with SGD.

(Previous slide had θ-dependence both on and inside the expectation. For now, let's simplify the problem so that ϕ does not depend on θ.)

Incorrect gradient computation:

$$\nabla_\theta \mathbb{E}_{X \sim f_\theta}[\phi(X)] \stackrel{?}{=} \mathbb{E}_{X \sim f_\theta}[\nabla_\theta \phi(X)] = \mathbb{E}_{X \sim f_\theta}[0] = 0$$

Log-derivative trick

Correct gradient computation:

$$\nabla_\theta \mathbb{E}_{X\sim f_\theta}[\phi(X)] = \nabla_\theta \int \phi(x) f_\theta(x)\, dx = \int \phi(x) \nabla_\theta f_\theta(x)\, dx$$

$$= \int \phi(x) \frac{\nabla_\theta f_\theta(x)}{f_\theta(x)} f_\theta(x)\, dx = \mathbb{E}_{X\sim f_\theta}\left[\phi(X)\frac{\nabla_\theta f_\theta(X)}{f_\theta(X)}\right]$$

$$= \mathbb{E}_{X\sim f_\theta}\left[\phi(X)\nabla_\theta \log(f_\theta(X))\right]$$

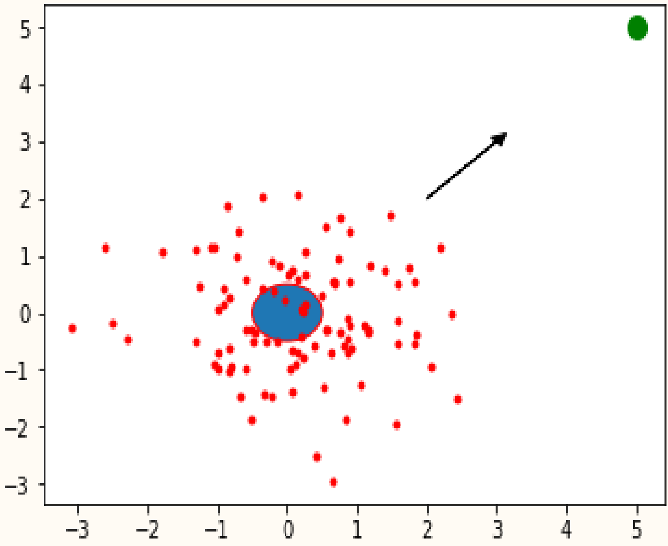
Therefore, $\phi(X)\nabla_\theta \log(f_\theta(X))$ with $X \sim f_\theta$ is a stochastic gradient of the loss function. This technique is called the *log-derivative trick*, the *likelihood ratio gradient*[#], or *REINFORCE*[*].

Formula with the log-derivative ($\nabla_\theta \log(\cdot)$) is convenient when dealing with Gaussians, or more generally exponential families, since the densities are of the form

$$f_\theta(x) = h(x)\exp(\text{function of } \theta)$$

[#]P. W. Glynn, Likelihood ratio gradient estimation for stochastic systems, *Communications of the ACM*, 1990.
[*]R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

Log-derivative trick example



Learn $\mu \in \mathbb{R}^2$ to minimize the objective below.

$$\underset{\mu \in \mathbb{R}^2}{\text{minimize}} \quad \mathbb{E}_{X \sim \mathcal{N}(\mu, I)} \left\| X - \begin{pmatrix} 5 \\ 5 \end{pmatrix} \right\|^2$$

Then the loss function is

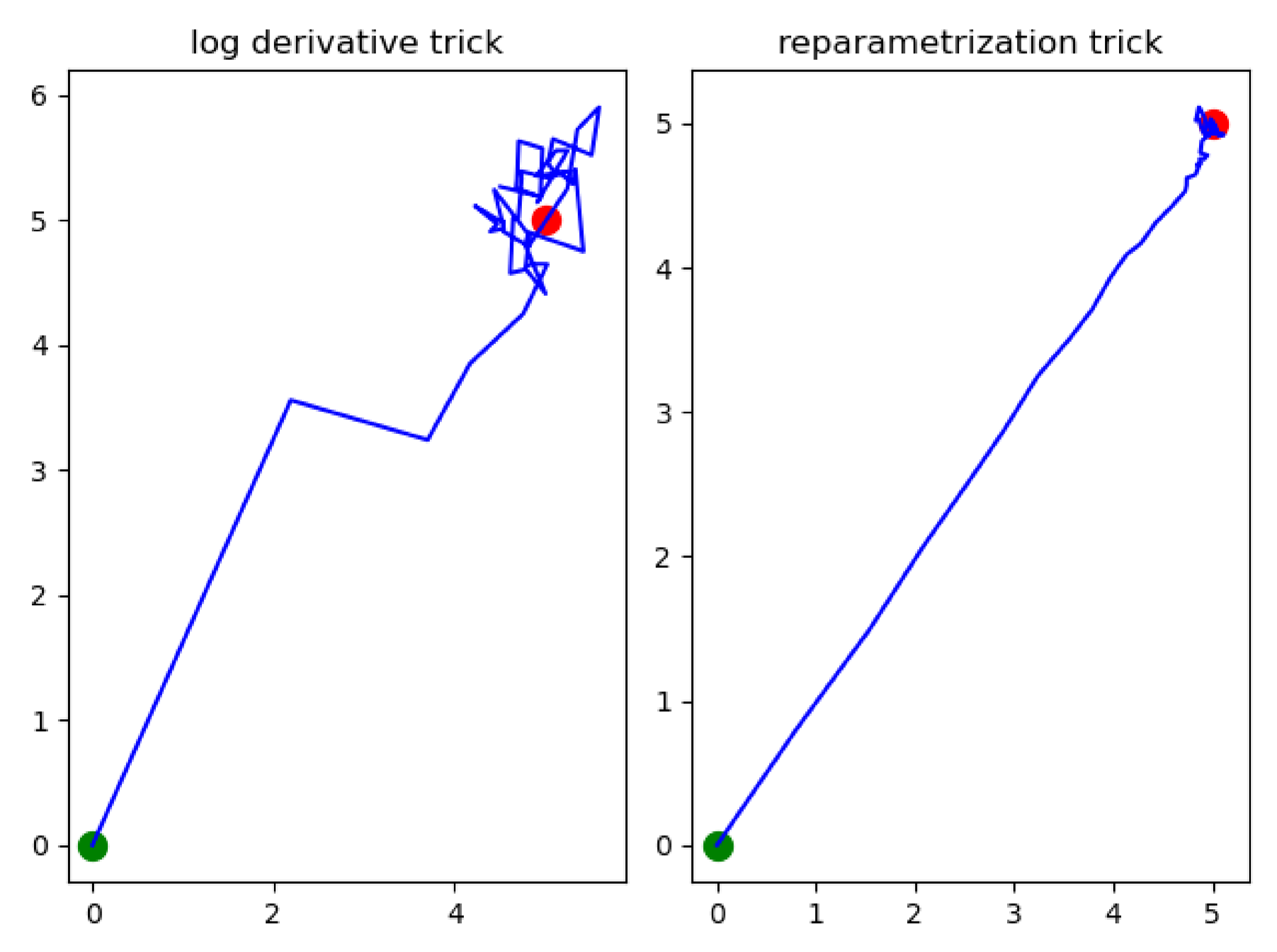
$$\mathcal{L}(\mu) = \mathbb{E}_{X \sim \mathcal{N}(\mu, I)} \left\| X - \begin{pmatrix} 5 \\ 5 \end{pmatrix} \right\|^2 = \int \left\| x - \begin{pmatrix} 5 \\ 5 \end{pmatrix} \right\|^2 \frac{1}{2\pi} \exp\left(-\frac{1}{2} \|x - \mu\|^2 \right) \, dx$$

And, using $X_1, \ldots, X_B \sim \mathcal{N}(\mu, I)$, we have stochastic gradients

$$\nabla_\mu \mathcal{L}(\mu) = \mathbb{E}_{X \sim q_\mu} \left[\left\| x - \begin{pmatrix} 5 \\ 5 \end{pmatrix} \right\|^2 \nabla_\mu \left(-\frac{1}{2} \|x - \mu\|^2 \right) \right] \approx \frac{1}{B} \sum_{i=1}^{B} \left\| X_i - \begin{pmatrix} 5 \\ 5 \end{pmatrix} \right\|^2 (X_i - \mu)$$

These stochastic gradients have large variance and thus SGD is slow.

Log-derivative trick example



Reparameterization trick

The *reparameterization trick* (RT) or the *pathwise derivative* (PD) relies on the key insight.

$$\mathbb{E}_{X\sim\mathcal{N}(\mu,\sigma^2)}\left[\phi(X)\right] = \mathbb{E}_{Y\sim\mathcal{N}(0,1)}\left[\phi\left(\mu+\sigma Y\right)\right]$$

Gradient computation:

$$\begin{aligned}\nabla_{\mu,\sigma}\mathbb{E}_{X\sim\mathcal{N}(\mu,\sigma^2)}\left[\phi(X)\right] = \mathbb{E}_{Y\sim\mathcal{N}(0,1)}\left[\nabla_{\mu,\sigma}\phi(\mu+\sigma Y)\right] &= \mathbb{E}_{Y\sim\mathcal{N}(0,1)}\left[\phi'(\mu+\sigma Y)\begin{bmatrix}1\\ Y\end{bmatrix}\right]\\ &\approx \frac{1}{B}\sum_{i=1}^{B}\phi'(\mu+\sigma Y_i)\begin{bmatrix}1\\ Y_i\end{bmatrix}, \qquad Y_1,\dots,Y_B\sim\mathcal{N}(0,I)\end{aligned}$$

RT is less general than log-derivative trick, but it usually produces stochastic gradients with lower variance.

Reparameterization trick example

Consider the same example as before

$$\mathcal{L}(\mu) = \mathbb{E}_{X\sim\mathcal{N}(\mu,I)} \left\| X - \begin{pmatrix} 5 \\ 5 \end{pmatrix} \right\|^2 = \mathbb{E}_{Y\sim\mathcal{N}(0,I)} \left\| Y + \mu - \begin{pmatrix} 5 \\ 5 \end{pmatrix} \right\|^2$$

Gradient computation:

$$\nabla_\mu \mathcal{L}(\mu) = \mathbb{E}_{Y\sim\mathcal{N}(0,I)} \nabla_\mu \left\| Y + \mu - \begin{pmatrix} 5 \\ 5 \end{pmatrix} \right\|^2 = 2\mathbb{E}_{Y\sim\mathcal{N}(0,I)} \left(Y + \mu - \begin{pmatrix} 5 \\ 5 \end{pmatrix} \right)$$

$$\approx \frac{2}{B} \sum_{i=1}^{B} \left(Y_i + \mu - \begin{pmatrix} 5 \\ 5 \end{pmatrix} \right), \qquad Y_1, \ldots, Y_B \sim \mathcal{N}(0, I)$$

These stochastic gradients have smaller variance and thus SGD is faster.

Reparameterization trick example