1 Setting

In gradient descent, we update the parameters θ of our model according to the rule $\theta \leftarrow \theta - \varepsilon \mathrm{E}_{x,y\sim D}\left[\nabla_\theta \ell(\theta; x, y)\right]$. For standard gradient descent, we perform this update only once after each full pass through the dataset, so the expectation is computed exactly. However this is very slow if the dataset is large. An alternative approach is to use stochastic gradient descent (SGD), where at each iteration we pick one datapoint (x_i, y_i) and use $\nabla_\theta \ell(\theta; x_i, y_i)$ as an estimator for the expected gradient over the entire dataset. That is, we use the update rule $\theta \leftarrow \theta - \varepsilon \nabla_\theta \ell(\theta; x_i, y_i)$. However, the gradient at different datapoints can be very different. Variance is a way for us to measure this difference and understand what impacts it might have on training.

2 Variance

Intuitively, variance is a measure of how "spread out" a dataset is. For single-variable distribution D, it is defined as

$$\begin{aligned}
\mathrm{Var}\left[X\right] &= \mathrm{E}_{X\sim D}\left[\left(X - \mathrm{E}_{X\sim D}[X]\right)^2\right] \\
&= \mathrm{E}_{X\sim D}\left[X^2 - 2X\mathrm{E}_{X\sim D}[X] + \mathrm{E}_{X\sim D}[X]^2\right] \\
&= \mathrm{E}_{X\sim D}\left[X^2\right] - \mathrm{E}_{X\sim D}\left[2X\mathrm{E}_{X\sim D}[X]\right] + \mathrm{E}_{X\sim D}\left[\mathrm{E}_{X\sim D}[X]^2\right] \\
&= \mathrm{E}_{X\sim D}\left[X^2\right] - 2\mathrm{E}_{X\sim D}[X]^2 + \mathrm{E}_{X\sim D}[X]^2 \\
&= \mathrm{E}_{X\sim D}\left[X^2\right] - \mathrm{E}_{X\sim D}[X]^2
\end{aligned}$$

The idea with this definition is that we are measuring the *expected squared deviation* of any given sample from the mean of the distribution. (Using the square rather than the absolute value of the deviation makes analysis easier). If the dataset is more spread out, with more points farther away from the average, then the variance will be high. A very concentrated dataset will have a low variance.

Similarly, when we talk about the variance of SGD, what we are trying to measure is the average difference between the gradient computed at a *single datapoint* and the gradient *averaged over all data points*. In this case, we are measuring the variance of vectors. There are a few ways to do that, but for this class we'll adopt the following definition: for a multivariable distribution D, let

$\mu = \mathrm{E}_{X \sim D}[X]$. Then

$$\begin{aligned}
\mathrm{Var}\left[X\right] &= \mathrm{E}_{X \sim D}\left[\left\|X - \mu\right\|^2\right] \\
&= \mathrm{E}_{X \sim D}\left[\left(X_1 - \mu_1\right)^2 + \cdots + \left(X_n - \mu_n\right)^2\right] \\
&= \mathrm{E}_{X \sim D}\left[\left(X_1 - \mu_1\right)^2\right] + \cdots + \mathrm{E}_{X \sim D}\left[\left(X_n - \mu_n\right)^2\right] \\
&= \mathrm{E}_{X \sim D}\left[X_1^2\right] - \mu_1^2 + \cdots + \mathrm{E}_{X \sim D}\left[X_n^2\right] - \mu_n^2 \\
&= \left(\mathrm{E}_{X \sim D}\left[X_1^2\right] + \cdots + \mathrm{E}_{X \sim D}\left[X_n^2\right]\right) - \left(\mu_1^2 + \cdots + \mu_n^2\right) \\
&= \mathrm{E}_{X \sim D}\left[X_1^2 + \cdots + X_n^2\right] - \left\|\mu\right\|^2 \\
&= \mathrm{E}_{X \sim D}\left[\left\|X\right\|^2\right] - \left\|\mu\right\|^2
\end{aligned}$$

Intuitively, this captures the average difference (specifically, the squared Euclidean distance) between the gradient at each measured point and the average gradient. Again, if the variance is high, that indicates that our measured points are more likely to be far away from the average.

Fundamentally, with SGD we are trading variance for speed. High variance causes problems because it means the gradient estimates we use for updating our parameters can be quite different from the true gradient. This causes the learning process to become chaotic. Rather than smoothly following the gradients to a point of minimal loss, SGD often jumps around and moves in the wrong direction at times. On the other hand, these noisy gradient estimates can be computed much faster than the true gradient, which means the learning process is able to converge faster.

Aside I should point out that the above definition is not standard. More formally, the variance of a vector distribution is usually defined as

$$\mathrm{Var}[X] = \mathrm{E}_{X \sim D}\left[(X - \mu)(X - \mu)^T\right],$$

often called the variance-covariance (or simply covariance) matrix. However this definition is unweildy for our purposes, so I will use the previous definition. Note that our definition is equivalent to taking the trace of the covariance matrix, and is also equivalent to swapping the order of the multiplication:

$$\mathrm{Var}[X] = \mathrm{E}_{X \sim D}\left[(X - \mu)^T(X - \mu)\right].$$

3 Minibatches

Minibatches are an approach to reduce variance in SGD. The idea is that rather than computing the gradient at a single point, we compute the gradient over some batch of data. Specifically, we partition the dataset D into equal-sized subsets $D_1, \ldots, D_k$ and use the update rule $\theta \leftarrow \theta - \epsilon \mathrm{E}_{x,y \sim D_i}\left[\nabla_\theta \ell(\theta; x, y)\right]$.

Since batched gradient descent is also often refered to as SGD, I will use the term *pure SGD* to refer to the situation where we use a single datapoint as an estimator. Batched gradient descent represents a middle ground between standard gradient descent (in which the entire dataset is one batch) and pure SGD (in which each datapoint is its own batch). Ideally, we are looking for a batch size that allows for both minimal variance and fast computation.

The primary advantage of minibatches compared to pure SGD is that they reduce the variance of the gradient estimate. First, let's look at the variance of gradient estimates with pure SGD:

$$\mathrm{Var}_{\mathrm{SGD}}\left[\nabla_\theta \ell(\theta; x, y)\right] = \mathrm{E}_{x,y\sim D}\left[\left\|\nabla_\theta \ell(\theta; x, y)\right\|^2\right] - \left\|\nabla_\theta L(\theta)\right\|^2,$$

compared to the estimates with minibatches:

$$\mathrm{Var}_{\mathrm{Batch}}\left[\nabla_\theta \ell(\theta; x, y)\right] = \mathrm{E}_{D_i}\left[\mathrm{E}_{x,y\sim D_i}\left[\left\|\nabla_\theta \ell(\theta; x, y)\right\|\right]^2\right] - \left\|\nabla_\theta L(\theta)\right\|^2.$$

The last term is identical in both cases, so really we just need to compare the first terms,

$$\mathrm{E}_{x,y\sim D}\left[\left\|\nabla_\theta \ell(\theta; x, y)\right\|^2\right] \qquad \text{and} \qquad \mathrm{E}_{D_i}\left[\mathrm{E}_{x,y\sim D_i}\left[\left\|\nabla_\theta \ell(\theta; x, y)\right\|\right]^2\right].$$

Here we can make use of Jensen's inequality, which states that for a convex function ϕ, $\phi\left(\mathrm{E}\left[X\right]\right) \leq \mathrm{E}\left[\phi(X)\right]$. In this case, the convex function we are considering is $\phi(X) = \|X\|^2$, and we can use this to show

$$\mathrm{E}_{x,y\sim D_i}\left[\left\|\nabla_\theta \ell(\theta; x, y)\right\|\right]^2 \leq \mathrm{E}_{x,y\sim D_i}\left[\left\|\nabla_\theta \ell(\theta; x, y)\right\|^2\right].$$

From here we can add an expectation over batches to both sides to conclude

$$\mathrm{E}_{D_i}\left[\mathrm{E}_{x,y\sim D_i}\left[\left\|\nabla_\theta \ell(\theta; x, y)\right\|\right]^2\right] \leq \mathrm{E}_{D_i}\left[\mathrm{E}_{x,y\sim D_i}\left[\left\|\nabla_\theta \ell(\theta; x, y)\right\|^2\right]\right].$$

Finally, we combine the two expectations on the right side to get

$$\mathrm{E}_{D_i}\left[\mathrm{E}_{x,y\sim D_i}\left[\left\|\nabla_\theta \ell(\theta; x, y)\right\|\right]^2\right] \leq \mathrm{E}_{x,y\sim D}\left[\left\|\nabla_\theta \ell(\theta; x, y)\right\|^2\right]$$

and therefore

$$\mathrm{Var}_{\mathrm{Batch}}\left[\nabla_\theta \ell(\theta; x, y)\right] \leq \mathrm{Var}_{\mathrm{SGD}}\left[\nabla_\theta \ell(\theta; x, y)\right].$$

This result shows that minibatches decrease the variance in stochastic gradient descent. Intuitively, this makes sense because we are computing our gradient estimates using more information. The flip side is that the more data you use to compute your gradient estimates, the more time it wil take. This is the main tradeoff to keep in mind when choosing a batch size. Generally speaking, I would recommend starting with relatively small batches, usually in a power-of-2 size (for example, 32 or 64) and then experimenting from there.