Stochastic Gradient Descent

Gradient Descent: A Recap

Recall gradient descent: the new weights are the old weights, but multiplied by a small amount (the step size, η) such that the loss function is a little bit better (found by taking its gradient). This is done until convergence (the difference between the new and old weights is 10^{-8}) or we hit around 10,000 iterations– whichever comes first. Too large of a step size can cause divergence.

Gradient descent learns really quickly initially, but often takes **fricking forever** to converge towards the end. That's because we don't know the "best" step size when we start.

We can remedy this by stepping as far as the gradient says we should– this scales the step size, at each step, to an appropriate amount. Recalling high school calculus and physics, it necessarily entails taking the derivative of the gradient, called the **Hessian**. This is a quadratic approximation (rather than linear) and lets you use Newton's method (recall SICP, or the Fast Inverse Square Root algorithm).

So there's a timsort (mergesort for large instances, insertion sort for small)-like method for fitting: do gradient descent first, then Newton's method (via the Hessian) afterwards.

What is SGD?

Stochastic gradient descent sacrifices a little bit of correctness for a lot of speed gain. (I know, didn't answer my own question.) You take **a few** terms in weights and step towards **just that subset's** gradient. This is because the number of weights could be in the millions.

$$\nabla l(w) = \frac{1}{m} \sum_{i=1}^{m} x_i (f(x_i, w) - y_i)$$

So, computing the "gradient" with $m - 1$ terms should be "good enough" for real-life m. With $m = 3$ it's a bad idea.

So pseudocode might be:

Repeat forever:

- Sample $x_i, y_i \sim D$
- $w^{k+1} = w^k - \eta \nabla l_i(w)$

So η_k could be $\propto \frac{1}{k}$; it doesn't really matter as long as it gets smaller over time. Actually, in practice, decaying step sizes are slow and otherwise suck.

Step Size

The change in our function's output is proportional to η, prediction error, and squared feature norm. If $\|x\|_2^2$ increases (more, or larger features), then η must decrease.

$$\eta = \frac{\eta_0}{max_x \|x\|_2^2}$$

where η_0 is the rate of decrease for prediction error.