Lecture #: Topic

Instructor: Aditya Bhaskara *Scribe: Ian Mogensen*

CS 5966/6966: Theory of Machine Learning

March 1st, 2022

Abstract

In this lecture we primarily discuss the concept of Preconditioning, and touch on Paralellism as it relates to Gradient Descent and Stability as a result of strongly convex functions.

1 Parallelism

Computing gradients can usually be parallelized, with parallelism taking place with computing the derivatives for each i (each row). This is generally what is referred to when people mention parallelism in Deep Learning, because epochs must run in series, and as such parallelism across time is infeasible without prior knowledge of where gradient descent will end up after a certain number of steps.

It is for this reason that there is a large focus on making the dependence on T very good (as in convergence with error $O(\frac{1}{\sqrt{T}})$ after T steps, seen in the previous class) - the lack of an ability to parallelize across time leaves one with little choice but to minimize the number of steps needed.

Second-order methods, seen later on, are methods sometimes preferred to what we have seen thus far, because they result in a much smaller number of iterations necessary. However, this smaller number of steps can come at the cost of heavier, more complicated computation.

2 Preconditioning

Broadly speaking, preconditioning in gradient descent is used to adjust for different curvature in different directions. To gain intuition, we consider the following example

$$f(x_1, x_2, x_3) = x_1^2 + \frac{x_2^2}{4} + \frac{x_3^2}{9}$$

which yields the gradient

$$\nabla f = \begin{pmatrix} 2x_1 \\ \frac{2x_2}{4} \\ \frac{2x_3}{9} \end{pmatrix}$$

and can be verified to have a Lipschitz constant of 2. From previous classes, we know that the step-size of gradient descent is $\eta = \frac{1}{2M}$ for an M-smooth function, hence we have a step size $< \frac{1}{4}$.

It follows that with $x^{(0)} = (1, 1, 1)$,

$$\nabla f(x^{(0)}) = \begin{pmatrix} 2 \\ 1/2 \\ 2/9 \end{pmatrix}$$

Moving to the next iteration yields

$$x^{(1)} = x^{(0)} - \frac{1}{4} \cdot \nabla f(x^{(0)}) = \begin{pmatrix} 1/2 \\ 7/8 \\ 17/18 \end{pmatrix}$$

We can see that while the first element of the vector makes decent progress towards approaching 0, the other two elements remain close to 1. It is clear that the other elements would progress very slowly because the Lipschitz constant used to determine the step-size is calculated based off the first gradient, which in this case is an upper-bound across all gradients. In other words, there is a Lipschitz constant that is much better specifically in the directions of x_2 and x_3.

A Hessian matrix is key to this idea of moving differently along different directions. A Hessian matrix is a matrix of second-order partial derivatives denoted as:

$$\mathbf{H} = \nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}$$

Much like assuming the function for regular gradient descent to behave linearly, for determining how much to move along different directions, we assume the function to behave quadratically. This quadratic approximation would be

$$(1) \quad f(x + \delta) = f(x) + \langle \delta, \nabla f(x) \rangle + \frac{1}{2} \delta^T (\mathbf{H}(x)) \cdot \delta$$

Note the similarity between this equation (1), and the one-dimensional quadratic $f(x) + \delta f'(x) + \delta/2! f''(x)$

where $\mathbf{H}(x) = \nabla^2 f(x)$ is the Hessian matrix.

To minimize the quadratic, we have

$$(2) \quad \delta = -(\nabla^2 f(x))^{-1} \nabla f(x)$$

The minimization in equation (2) also shows resemblance to that of a one-dimensional quadratic: $\delta = -f'(x)/f''(x)$

There are many methods that improve or are generalizations on this notion of preconditioning. AdaGrad, for example, can be thought of as approximating second-order methods with only first-order information.

3 Stability

Along with faster optimization, strong convexity also adds stability, which means that given some strongly convex function $f(x)$, it holds that with

$$g(x) = f(x) + \delta(x)$$

$\arg\min_x f(x)$ is close to $\arg\min_x g(x)$ for some "small" δ.

Intuitively, one can think of this as a lack of sudden fluctuations (just as the term "stability" implies). If there was sporadicness in the function, then the small movement $\delta(x)$ from $f(x)$ could result in a function value $g(x)$ very different from $f(x)$.

Note that a loss function that is more stable also generalizes well.