Lecture 14: Strong Convexity, Regularization and Stability

Instructor: Aditya Bhaskara *Scribe: Sai Raghavendra Sugeeth Puranam*

CS 5966/6966: Theory of Machine Learning

March 1st, 2022

Abstract

In this lecture we went over the Summary of Gradient Descent, went briefly over parallelizing gradient descent. We also discussed about Optimal bounds, Preconditioning, how we can use first order methods to approximate for second order methods and briefly went into the motivation for Strong Convexity.

1 Summary of Gradient Descent

Gradient Descent is a simple iterative first order algorithm which tries to find *argmin* of $f(x)$ over a convex domain D where $x \in D$.

If the function is a L-Lipschitz, we can say that the error after T steps is in the order of $O(\frac{1}{\sqrt{T}})$. That is formally defined as below.

$$f(\bar{x_t}) - f(x^*) \leq O(\frac{1}{\sqrt{T}})$$

where, f is the function used to calculate the error,
x_t is the point reached after T steps,
x^* is the point with minimum error

Smooth Convex function: Generally we can define a function as "smooth" if the gradient doesn't change abruptly. Formally if the gradient is M-Lipschitz.

If the step size is $< \frac{1}{2M}$ after T steps, the error will be in the range of $O(\frac{1}{T})$ as opposed to $O(\frac{1}{\sqrt{T}})$.

so formally for smooth convex function,

$$f(\bar{x_t}) - f(x^*) \leq O(\frac{1}{T})$$

μ **Strong Convex function:** A function is said to be μ Strong Convex function if

$$f(y) \geq f(x) + < \nabla f(x), y - x > + \mu ||y - x||^2$$

where y and x are points on the strong convex function.

with a strong convex function, convergence bound improves to roughly $exp(-\frac{\mu}{M}T)$ which is significantly good compared to Smooth convex function.

Strong convexity based results are good only if the condition number: $\frac{M}{\mu}$ is "small".

2 Parallelism and Complexity

Most of the optimization literature assumes that gradients are "easy" to get and Hessians are "complex"

We have a the function as $f : \mathbb{R}^d \to \mathbb{R}$ Gradients are easy because gradient is a vector in $\mathbb{R}^d$ whereas Hessians are complex because hessian is a matrix in $\mathbb{R}^{d*d}$

But sometimes gradients can also be difficult or time consuming to compute let's say in least squares regression, gradients could take time in order of d^2.

So can we parallelize gradient descent? No they cannot be parallelized across time but all gradients of variables at one step can be computed in parallel. Hence epochs must be done in series. This is the reason most of the literature tries to represent error as the dependence on T. So less the number of iterations it is better.

Second order methods: Normally we do the following update.

$$w_{t+1} = w_t - \eta \nabla f(w_t)$$

But sometimes second order update as below is preferred,

$$w_{t+1} = w_t - \frac{H^{-1}}{2} \nabla f(w_t)$$

by doing the second order update the number of iterations will be very small. But inverting the H matrix is the costly operation. So efforts have been directed in inverting a matrix faster by using parallelization.

Most recent papers like training an ImageNet under an hour etc try to use second order methods for fast convergence. They try to do matrix approximations by using tricks from numerical methods.

It is most promising to do the inversion by modelling it as an optimization problem but it shouldn't be done naively but **preconditioning** should be used to do this approximation which is discussed in the next sections.

Interest has also fizzled out in this area. If you don't have derivatives correctly these Hessian methods don't seem to work properly. They converge quickly but the guarantee is not really great. These second order methods work for certain applications but are not universal.

3 "Optimal" Bounds

With just the L-L-Lipschitz assumption for a function, the error $O(\frac{1}{\sqrt{T}})$ cannot be improved at least with "sub-gradient oracle". Hence this is a tight bound and cannot be improved.

But we can do improvement for smooth functions. Assuming the function is purely smooth, with Nesterov acceleration, we can get improve the error to $O(\frac{1}{T^2})$. Works have been done to extend this Nesterov acceleration to Stochastic gradient descent.

But all of these use the fact that there is an oracle that gives the gradient. But the question arises "what if we can use the information beyond the gradient?" This question is answered in subsequent section of **Preconditioning**.

4 Preconditioning

Hessian plays an informal role in most Gradient Descent Analysis (even M-smooth). To illustrate this, lets take an example as below.

Let us consider a simple function

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

The optimal value of this function is at value $(0, 0, 0)$

Let us say we start at some point $(1, 1, 1)$ to reach the optimal point.

The gradient of f $\nabla f = (2x_1, 2x_2/4, 2x_3/9)$. The Lipschitz constant of ∇f would be 2 because of the first co-ordinate(as its values changes by a factor of 2).

So when we start at the point $x^{(0)} = (1, 1, 1)$, $\nabla f(x^{(0)}) = (2, 1/2, 2/9)$

then considering a step size of $\frac{1}{4}$,

$$x^{(1)} = x^{(0)} - \frac{1}{4}\nabla f(x^{(0)}) = (1/2, 7/8, 8/9)$$

So the second and third co-ordinates take a while to reach 0 because the gradients are small.

So the limiting factor will be the third co-ordinate and it will lead to slower convergence.

How to solve this?: Looking back at the Lipschitz constants it is seen that for the 2nd and 3rd co-ordinate the Lipschitz constants are much lower ($\frac{2}{4} and \frac{2}{9}$ respectively). So we can move aggressively in those directions. So the idea is instead of moving along the derivative the movement can be along the derivative scaled different for each derivative. This broad idea is called as **Preconditioning**.

The Hessian is the key to move different along different directions. The optimal movement is done using second order information.

To illustrate this, let us consider a function that behaves like a quadratic in its neighborhood.

$$f(x + \delta) = f(x) + < \delta, \nabla f(x) > + \frac{1}{2}\delta^T(\nabla^2 f(x))\delta + ..$$

the $\nabla^2 f(x)$ term is the Hessian at x.

Similarly, in 1-Dimension if f is a 1-Dimensional function,

$$f(x + \delta) = f(x) + \delta f'(x) + \delta^2 f''(x)$$

which is a quadratic function in δ

This quadratic form will be minimum at

$$\delta = \frac{-f^{'}(x)}{f''(x)}$$

the higher dimension analogous of this is,

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

So when using second order methods, we jump to the above point in the next iteration. Which leads to fast convergence. But for this to be formal the quadratic approximation must hold which doesn't usually happen because of the 3rd order term.

5 Improvements, Generalizations

AdaGrad: This is like a first order approximation to the second order methods. The idea is you try to find the scale of the direction by looking at more and more samples. Essentially keeping track of approximation of Hessian by using first order information.

Polyak's "heavy ball" method(momentum): For strongly convex function we can do aggressive updates using the momentum information instead of $e^{-\frac{T}{\kappa}}$ we can have convergence bound of $e^{-\frac{T}{\sqrt{\kappa}}}$.

There are also many other methods like Newton's second order methods which try to achieve these improvements in convergence steps.

6 Motivation for Strong Convexity

We already saw that strong convexity leads to "faster" optimization.

Additional Benefit of strong convexity is the "stability" to small perturbation.

Stability to Perturbation: Let us consider a strongly convex function $f(x)$ and there is another function $g(x) = f(x) + \delta(x)$ if the $\delta(x)$ is not big the $argmin f(x)$ is "close" to $argming(x)$. This effect of strong convex functions is called stability.

Illustratively, if $f(x) = x^2, g(x) = x^2 + \alpha x$ which have min at 0 and $-\frac{\alpha}{2}$ respectively. If α is small we can have a fair idea of $g(x)$ using $f(x)$.