Distributed Optimization and Machine Learning

Prof. Mayank Baranwal

Computer Science & Engineering, Electrical Engineering, Mathematics

Indian Institute of Technology Bombay

Week-1

Lecture - 2: Analyzing optimization algorithms in continuous time domain

Alright. So, let us move back to journal. So, when we talk about optimization right. So, if I try to write the gradient descent algorithm, the standard gradient descent which is xk plus 1 is xk minus eta gradient f of xk this is when we want to minimize f of x right

$$\underbrace{x_{k+1} = x_k - \eta \nabla F(x_k)}_{\text{Gradient Descent (GD)}} \qquad \min_{x} F(x)$$

and this is the simple gradient descent algorithm ok. So one way is to directly analyze these algorithms in discrete setting or in discretized setting. But what I can do is, I can really view this algorithm. So I can pretty much rewrite it like this.

$$\frac{x_{k+1} - x_k}{\eta} = -\nabla F(x_k)$$

And I can consider the limiting case when the step size is really small, right. So the limiting case when eta goes to 0, what does this left hand side converge to? So so this

$$\lim_{\eta \to 0} \left(\frac{x_{k+1} - x_k}{\eta} \right) = -\nabla F(x_k)$$

$$\boxed{\dot{x} = -\nabla F(x)}$$

left-hand side in the limiting case eta goes to 0 this is nothing but your x dot ok and the right-hand side is gradient of f of x so this basically becomes a continuous time variant of your gradient descent it is also called gradient flow ok? So, this is just basically studying the continuous time limit of your discretized algorithm right? I mean in practice we are always going to implement algorithm like the way we have seen over here right? I mean you are going to be implementing everything in computer. So, it has to be discretized, but from the point of view of analysis.

So, it is much easier to analyze continuous-time variance of algorithms right. And the reason being let us say I want to optimize. So, if let us say f of x happens to be half x square ok what is gradient of f of x x so what is the continuous time variant of it x what is negative x what is the equilibrium of this particular so this is called this is a dynamical system now right so what is the equilibrium of this dynamical system 0 right which is the optimal solution right so x star is equal to 0, this is the equilibrium point, also the optimal solution.

$$F(x) = \frac{1}{2}x^2 \qquad \boxed{\dot{x} = -x}$$

$$\nabla F(x) = x \qquad x^* = 0 \text{ Equilibrium Pt/}$$

So, can we say anything about the stability of this particular equilibrium point? It is exponentially stable. So, x of t is like e to the negative t right.

So, it becomes much easier to analyze these algorithms. So, we know that this algorithm is now exponentially fast at least in continuous time it is exponentially fast convergent to the optimal solution. So, this is equilibrium is exponentially stable. So, it tells you how quickly this particular algorithm converges to the optimal or this particular dynamical system or trajectories of this dynamical system converge to the equilibrium point. So, it becomes much easier to analyze than maybe looking at the discretized version of an algorithm.

Just from the analysis point of view it makes things much easier. So, for folks who have had courses in control theory and stability theory, you would have heard of something called Lyapunov stability right? So, there are lot of results in Lyapunov stability which are related to continuous time dynamical systems and we are going to make use. So, in since I mean it is a syscon course. So, we are definitely going to make use of all the related concepts in stability theory and we will try and analyze these algorithms as well as we will try and develop new algorithms which are at least in continuous time probably faster than simple gradient descent or gradient flow. So, one example could be So, let us say I choose I define my dynamical system which looks something like this divided by the norm of this gradient.

$$\dot{x} = -\frac{\nabla F(x)}{\|\nabla F(x)\|}$$

So, for f of x which is half x square, what does this right hand side look like for this particular dynamical system? x dot is minus sign of x right. is like this particular. So, when x is positive, sign of x is 1. So, x dot is always negative 1 when x is positive and x dot is 1 when x is negative and it is 0 when it is 0 or may be not well defined when it is 0 when x is equal to 0. But the point is when you look at function like x square or half x square, if you are approaching closer to the optimal solution, the gradient, the value of the gradient also becomes smaller and smaller right.

So x dot is negative. So the gradient of f of x is simply x right. So the gradient of f of x is x. And the dynamical system that we were working with was x dot is negative x. Now as you approach closer to the optimal solution, the value of x also becomes smaller.

That means you are making smaller and smaller updates towards the optimal solution. And that basically slows down your convergence speed. On the other hand, this particular dynamical system or this particular dynamical system, this always has a gradient of plus one or minus one, no matter where you are. So this becomes particularly useful when you are within the minus one to one range. If you are within minus one to one range, you still have a large gradient value as compared to this particular dynamical system.

And therefore you can make faster updates towards your optimal solution. This particular dynamical system is useful if let's say x is more than 1 then I mean then you wouldn't want to restrict your particular update to just plus 1 or minus 1 right. But then if you are basically in the regime from minus 1 to 1 you can make better updates or you can make faster updates by choosing this particular dynamical system right? So that gives you an idea of how to design dynamical system which first of all have the same equilibrium point this one this also has the same equilibrium point right gradient f of x is 0 when x is equal to x star right. So this has the same equilibrium point but by just simply redesigning the vector field or the right-hand side of the vector field you can come up with algorithms which are maybe faster even in discretized version. So if I were to discretize this algorithm I would write this as xk plus 1 is xk minus step size times something like this.

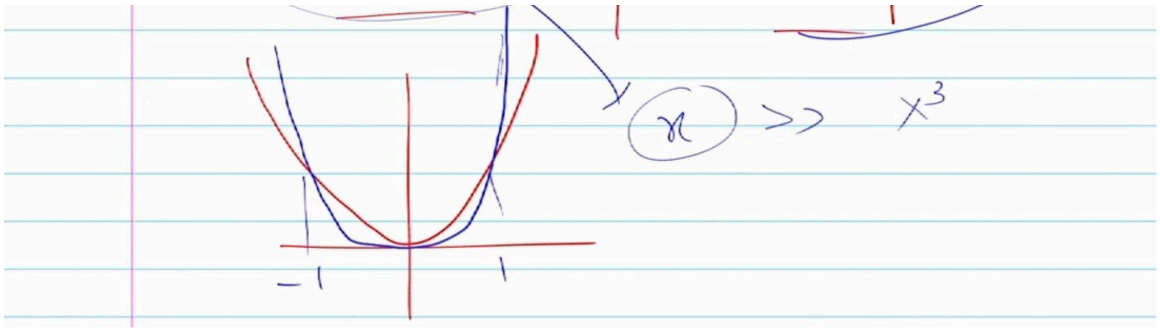
$$x_{k+1} = x_k - \eta \frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}$$

And if you recall algorithms like Adam or Adagrad, you would see that there is some kind of gradient normalization in those commonly used algorithms. the ones that are most popularly used for training deep neural networks you would see that they have used some form of gradient normalization and this is one of the this is the reason why we use gradient normalization because we can actually closer to the optimal solution we can make better for faster updates then is this clear? all right. So, what is the difference between optimizing functions let us say I have a function of the form f1 x is half x

$$\underline{F_1(x) = \frac{1}{2}x^2} \quad \Big| \quad F_2(x) = \frac{1}{4}x^4$$

square like this and f2 x one-fourth x power 4 ok. So, both these functions have the same optimal solutions x equal to 0 right. So, which one do you think is more sort of suitable function to work? Suitable in the sense which I mean for which particular function we can arrive at the optimal solution faster.

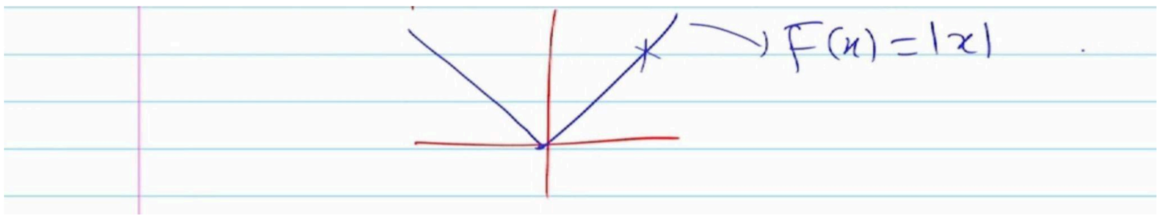
why first one right one to one right right. So, let us say this is your half x square right and if I look at maybe one fourth x to the power four possibly looks something like this, something like this right, this range is minus 1 to 1. So, outside this range you have better gradients with this particular function, but the moment you are in this range of minus 1 to 1, the gradient values are basically in this case x cube right, whereas this one has gradient



of the form x. Now for x between negative 1 and 1, this is much much greater than x cube and you would be making faster sort of updates towards the optimal solution than working with functions of the form 1 fourth x to the 4 and this is precisely why we tend to work with functions which are mean square laws or something mean square and not quartic kind of function because we can optimize those functions much faster than something like this. So these class of functions are called strongly convex function.

Does everyone know what convex function is? So we will also again I mean as I said like we will start with optimization first. So we will be reviewing the basics of optimization followed by basics of stability theory and that's when like once we have covered those is when we are going to move to study distributed optimization. But this class of function is called strongly convex function and because of the strong convexity nature, you continue to have like a gradient value which does not diminish as fast as may be something like one fourth quartic function of this form. And therefore it's better to work with these class of functions where you can provide accelerated convergence guarantees than with working with functions like these where it's difficult to provide accelerated convergence guarantees. The question is can all functions be accelerated or optimization of all the functions be accelerated? So strongly convex is one class of function that we looked at which looks like, I mean you can design accelerated optimization algorithm.

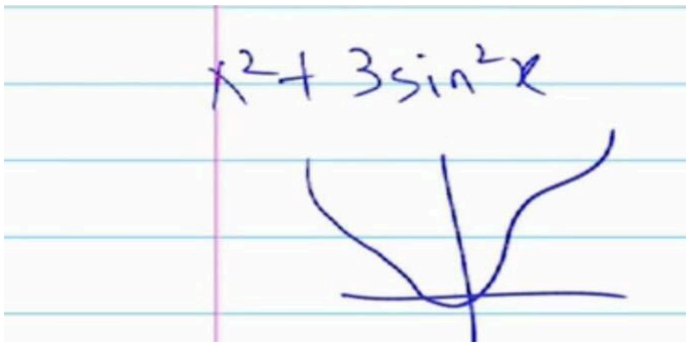
We just looked at the normalization form of it as well. If I look at functions of this form something like mod x or a smooth version of it where I mean you do not you have a well-defined gradient. So, you can make it arbitrarily smooth over here. So, this f of x is mod x. So, what is the gradient of or the magnitude of the gradient at any point plus 1 right magnitude of the gradient is 1 right? So just by knowing the gradient value at any



point, you cannot ascertain how far you are from the optimal solution.

Whereas with x square if you know the gradient value you can I mean at least if you know the optimal solution and the gradient value you know that how far you are from the optimal solution. So just by knowing the gradient value here if you use any gradient-based optimization There is no way for you to know how far you are from the optimal solution and therefore these class of functions which do not have sort of this form of gradient, a strongly convex form of gradient, these class of functions cannot be accelerated in general using simple gradient-based optimization. So, there is a specific class of functions, strongly convex function happens to be one of those. The other functions which I mean again we will study later, functions which are called, functions which satisfy something called PL inequality. which are some generalization of strongly convex function.

So, the function that satisfy PL inequality need not be convex. For instance, if I look at x square plus 3 sin square x. So, the graph of this would look something like this. So, this is



not a convex function. But then it still has a unique minimum which is at x equal to 0.

So, these class of functions again can be accelerated. So, we are also going to look at different classes of functions that can whose optimization can be accelerated and so that is again something that we are going to look at in this course. Thank you.