


Course Name: Optimization Theory and Algorithms
Professor Name: Dr. Uday K. Khankhoje
Department Name: Electrical Engineering
Institute Name: Indian Institute of Technology Madras
Week - 07
Lecture - 49

Intro to Newton methods

Okay. So then this allows us to move to the next part of set of methods in optimization. We have looked at first, these are all first order methods because we are only computing gradient at best, right? So the other big elephant in the optimization room is second order methods. So Newton and quasi-Newton methods. So that is what we are going to look at next. Okay.

So those of you reading the book, this is roughly chapters three, six, and seven of Nocedal and Wright. We've already discussed the Newton direction when we started with line search methods, right? So everyone kind of knows the motivation. So what was the motivation to go towards Newton methods? Rate of convergence was quadratic, right? Not linear. Rate of convergence was quadratic.

Okay. And there is no free lunch. What is the price? Compute Hessians, right? Why is it expensive to compute a Hessian? Because I have to find the gradient of gradient in some sense. So if I take, for example, let's take function $f(x, y)$. So if I do ∇f , what all is there in ∇f ? What is the expression for ∇f ? $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}$.




ch 3,6,7 of NW

Motivation \rightarrow Rate of convg quadratic

Price \rightarrow Compute Hessian.

$f(x,y) \rightarrow \nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$

$\frac{f(x+h,y) - f(x,y)}{h}$



OPTIMIZATION THEORY AND ALGORITHMS

Let us say I was evaluating this by finite differences. Finite difference is simply $f(x + h, y) - f(x, y)$ divided by h . So how many function evaluations did I need over here to calculate ∇f ?

Two. At least three, no? Because there is $f(x + h, y)$, there is $f(x + h, y)$, $f(x, y + h)$, and $f(x, y)$. So I need at least three function evaluations.

If I am doing, this is what is called a one-sided difference. There is also a two-sided difference or a central difference. How do I write a central difference? This is one-sided. $f(x + h, y) - f(x - h, y)$ divided by $2h$. Here how many function evaluations would I need? Four function evaluations to get the gradient because there is no common point now, right.

It is $x + h, x - h, y + h, y - h$. What is the advantage of a two-sided? It is much more accurate, right. Numerically you can prove this is actually. If you take the Taylor series expansion of $f(x + h)$, what is the order of the error? So $f(x)$, let's do this. So error, if I write $f(x + h)$, the Taylor series of this, this is going to be $f(x, y) + h \frac{\partial f}{\partial x}(x, y) + O(h^2)$.

NPTEL

$f(x, y) \rightarrow \nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} \rightarrow 3 \text{ fn evals}$

one sided $\left(\frac{f(x+h, y) - f(x, y)}{h} \right)$ two sided $\left(\frac{f(x+h, y) - f(x-h, y)}{2h} \right)$

error: $f(x+h, y) \approx f(x, y) + h \frac{\partial f}{\partial x} + O(h^2)$

error: $O(h^3)$

7/7

OPTIMIZATION THEORY AND ALGORITHMS

So when I take this guy to the left-hand side, my error is $O(h^2)$. On the other hand if I take this guy and you write the Taylor series expansion for $f(x + h)$ and $f(x - h)$. What will happen to the quadratic terms? They cancel out exactly because h^2, h^2 has the same sign and subtracting the two. So what will be the order of error? h^3 . This has order h^3 error. Simple Taylor series or Taylor expansion.

So that is why the two-sided difference is much more accurate because the error is now going as order h^3 . All there is a price to be paid. The price is I need four function evaluations instead of three function evaluations. So this was just to calculate ∇f . Now what is the Hessian of f ? $f_{xx}, f_{xy}, f_{yx}, f_{yy}$.

So what is f_{xx} ? What do I mean by f_{xx} ? $\frac{\partial^2 f}{\partial x^2}$. And this is going to be $\frac{\partial^2 f}{\partial x \partial y}$. So each of these, I mean this I can write as this, $\frac{\partial}{\partial x}, \frac{\partial f}{\partial y}$. So I need many more function evaluations than just 3 for each of these terms. Because I am going to get this is going to be a finite difference in y .

And together this whole thing is going to be FD in x . This is all just to get one guy. Similarly, I have to repeat it for f_{xx}, f_{yy} . So it's easy enough to calculate how many function evaluations you will need over here. So that is the price that you're paying.

And this is a simple toy illustration of 2D. Real-life problems will have, let's say, a few 10,000 or a few lakh or a million dimensions of f . Good luck, right? You don't find the use of Newton methods much in the machine learning or data science, big data applications for this reason.

Okay. All right. Okay. So let's do a bit of revision of this. What are the requirements of using or doing a Newton search method? Okay. Is the Newton method a line search method? It is a line search method, right. So it is a line search method which is great because we have already studied one type of line search method which was gradient descent, right.

So this is going to be simply $x_{k+1} = x_k + \alpha_k p_k$. We already have a recipe for calculating α which is inexact line search. So α_k can come from inexact line search. What was the Newton direction equal to? Does anyone recall? There was a negative Hessian inverse multiplied by ∇f_k . So this is the Newton direction and just to contrast it, this is my steepest descent was $-\nabla f_k$.

But this was not all, right? It was not, it is not enough to just write this over here. There was one further requirement on the Hessian.

NPTEL

error: $f(x+h, y) \approx f(x, y) + h \frac{\partial f}{\partial x} + O(h^2)$

$\nabla^2 f = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{bmatrix}$


error: $O(h^3)$

$\frac{\partial^2 f}{\partial x^2}$ $\frac{\partial^2 f}{\partial x \partial y} \rightarrow \frac{\partial}{\partial x} \left(\frac{\partial f}{\partial y} \right)$

FD in y

The Hessian needed to be positive definite, right? So do you recall why that is, how can we derive that this is necessary? How did we do it? We needed to show that it leads to a legitimate

descent direction. For that what was the expression? I needed to look at the cosine of the angle between these guys. So between $-\nabla f_k$ and my p_k , this angle θ_k . So, cosine θ between these two vectors would simply be

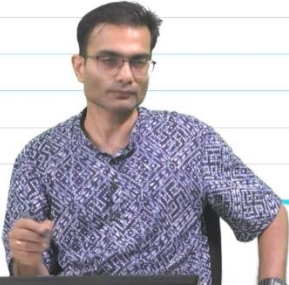


It is a line search method:

$$x_{k+1} = x_k + \alpha_k p_k.$$

↓ inexact L.S.

$$p_k^N = -(\nabla^2 f_k)^{-1} \nabla f_k \quad \text{Newton}$$

$$p_k^{SD} = -(\mathbf{I}) \nabla f_k.$$


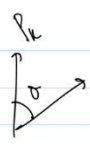
OPTIMIZATION THEORY AND ALGORITHMS

$$\cos(\theta_k) = \frac{-\nabla f_k^T p_k}{\|\nabla f_k\| \|p_k\|}$$

No problem with the denominator because that is always positive.

Over here, if I substitute the numerator, p_k is coming from here, the Newton expression. This became ∇f_k^T . And this needed to be, cosine of this needed to be what? Greater than 0, less than 0. Greater than 0. So this has to be like this.

So therefore this expression has to be positive. So you can see therefore that if the inverse Hessian is positive definite, then p_k^n is a legitimate descent direction. This is a stronger requirement. Why is it a slightly stronger requirement but a very practical requirement? Because when I say positive definite, what am I saying? If the matrix A is positive definite, I am saying that $z^T A z > 0$ for all z .



$$0 < \cos \theta = \frac{-\nabla f_k^T p_k}{\|\nabla f_k\| \|p_k\|} = \frac{\nabla f_k^T (\nabla^2 f_k)^{-1} \nabla f_k^T}{(\quad)}$$

A is P.D. Δ^0 x^N
 \Rightarrow If $(\nabla^2 f)^{-1}$ is P.D. the p_k^N is a legit D.D.

For all z . Here what do I need? I am always only going to multiply it by ∇f^T , right? So it is a little bit of a weaker thing that I require for making sure the Newton direction is a legitimate descent direction. But in practice, there is no way for me to know what set of values this ∇f is going to take. So since there is no way for me to practically ensure that I will just say let it be a positive definite matrix. The consequence of this is that in practice when you implement the Newton method, it may happen that your matrix is not positive definite.

But this product is positive. So you can see what has happened over there. There are some vectors for which this product is positive. You just got lucky. And ideally, when we are solving research problems, we do not want to rely on luck any more than necessary, right? So that's why this is standard requirement, okay? Now you will find that in many places it is not written that the inverse of the Hessian is positive definite. Instead, it is written that the Hessian is positive definite.



$$p_k = -B_k^{-1} \nabla f_k$$

If $B_k = \nabla^2 f_k$ Newton
if B_k is an approx of $\nabla^2 f_k$,
quasi-Newton method.



OPTIMIZATION THEORY AND ALGORITHMS

Are these two the same things? Because what is the simple linear algebra concept that tells us that it is the same thing? The eigenvalue decomposition, right? So if I write this as $Q\Lambda Q^T$, right. These are orthogonal vectors Q and what is going to be this inverse? It is going to be the same Q is going to come here, this is going to get inverted, Q^T . These are greater than 0 diagonal values, these are also therefore greater than 0 diagonal values, right. So, again linear algebra over here. So I have mentioned Newton and Newton-like methods.

Okay. So now we will just, I will just introduce a little bit more of terminology. So what you will find in most of the literature is that this p_k is written in the following way. Instead of writing the Hessian explicitly, it is written as the term given for it is. If this B_k is equal to the Hessian, we say of course it is a Newton method. If B_k is an approximation of the Hessian, it is called a Newton-like method. Or the more precise word would be quasi-Newton.

Okay. So you will find the same word, the same term B_k being used throughout the discussion that will follow in the subsequent lectures. Which means that I can switch between Newton and quasi-Newton depending on how I compute my B_k . Okay. And the whole point of a quasi-Newton method is that I want to approximate the Hessian. Means I do not want to calculate it because we have seen what is the cost involved.

I want to approximate this Hessian. And this quasi-Newton method gives us again there are several cousins of quasi-Newton methods. Different tricks to estimate the Hessian. Not calculate the Hessian but estimate the Hessian. So this gives rise to like we had a family of non-linear CG methods. We have a family of quasi-Newton methods, okay.

Which since they do not directly require calculating the Hessian, they are very, very competitive with your preconditioned CG, nonlinear CG, all of that, okay. So that is about it I think that we want to talk about. In terms of introducing, it is more like a revision of what you already knew for the requirements for CG and, sorry, for the Newton methods and the quasi-Newton methods,

okay. We will have to ensure that this approximation is also positive definite. Otherwise it will not be a descent direction, okay.

So next week we will look at further analysis of the Newton method, okay. For a lot of engineering problems which are not high in dimension, Newton methods are very, very competitive because of the much faster rate of convergence.