

Course Name: Optimization Theory and Algorithms
Professor Name: Dr. Uday K. Khankhoje
Department Name: Electrical Engineering
Institute Name: Indian Institute of Technology Madras
Week - 06
Lecture - 44

Discussion on doubts

Alright, let us get started. So, I will just go over the doubt sheets, not many, but there was one question about when will the project topics be released. So, this is about your course project. So, I am not going to release project topics, you are going to come up with project topics yourself and then I will let you know if that is in the right direction. So, the initiative is from your side. We can start talking about this maybe a few weeks down the line after we have finished for example, the non-linear CG method.

So, then you have at least some basic tools by which you can understand broad set of papers ok. But what is what we have done in the past is that you look up research papers that are somewhat recent by somewhat recent let us say last 10 years or so. and you try to explain them using what you have learned in the course in the language of that and you present like how is this being put to use in the research industry whatever right it could be your own research problem also so that is that should be something relevant, newish in the last 10 years. It should not be some 50 year old problem which is long been solved.


We are not looking for textbook illustrations of you know there are so many topics in optimization which we will not be able to cover. So, you do not go to some old optimization book and pull out some chapter which is not being covered that that does not count should be based on a paper of the last say 10 years. That's roughly the motivation in the project topic. So I will not be releasing the project topic. I'll have a spreadsheet shared with you.

You can enter your title and reference paper. I'll have a look at it and then give you a thumbs up, thumbs down, whatever. Any questions about this? Anything. I mean, what did you have in mind? Is there something you would like to do different? OK. I guess you will start thinking about it from today.

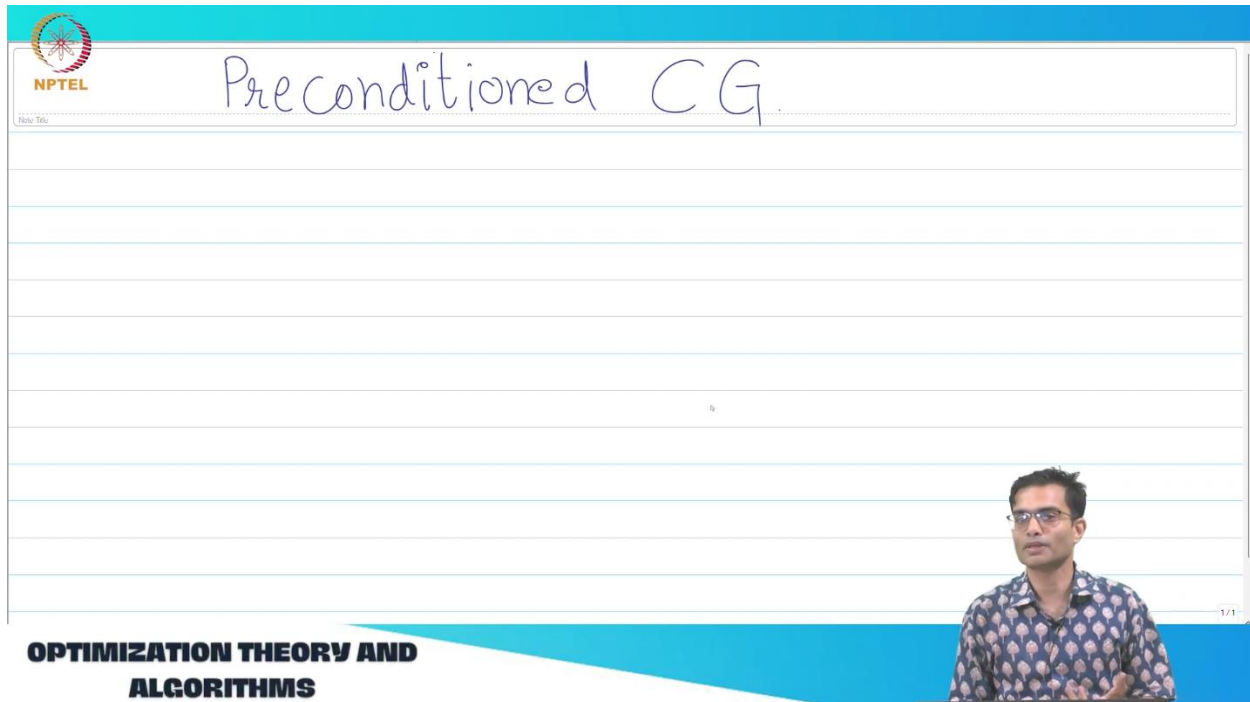
So, then student has not written the name. What does β_k mean in the conjugate gradient method? Is it a step length like α ? So, what was where did β_k come in? To calculate the new conjugate direction I use the old conjugate direction and the steepest descent and I added some coefficient to one of them as a linear coefficient of linear combination that is how I came up with β . Right. Then how would I use or how I mean how would I define β then like how would I calculate the value of β not a formula what is the concept to have the conjugacy be satisfied between p_k and p_{k-1} , I left multiplied by whatever $p_k^T A p_k$ and that gave me a certain expression for β . Right. So, I got that is how I calculated β . So, β is not a step length for which I am going to do line search or anything, it is a parameter which helps to ensure conjugacy of steps ok.

So, that is what is the motivation of β . Then I got like several doubts on the same theme. So, I will take them up all together first is why is this why steepest descent method is dying for bad

condition number how to interpret it ok. Now, why is this why was the when we increase the condition number the steepest descent convergence became worse and worse.



Preconditioned CG



OPTIMIZATION THEORY AND ALGORITHMS

Right. That we already saw from the when we did proof of the steepest descent method we saw that the rate depended on the condition number. So, mathematically we can understand that as the condition number is getting worse it takes longer time to converge. The other way of thinking about it is what was the motivation of condition number? It was like an amplification factor if there is an error in the right hand side how much error do I get in the solution right. So, as I am going through the iterations my solution is not accurate obviously I have not reached the solution. So, if the condition number is worse I am making some progress in the right direction, but my error is also increasing because of this is blowing up because of this condition number.

So, every step I am making some progress and I am being thrown away by the condition number that again make progress again I am thrown away. So, that is why if the condition number is 1 perfect I never make a mistake whatever is you know there I achieve it. So, that is the intuitive field for condition number. So, whenever you want to understand anything related to condition number keep in mind it is like an amplifier. but how much is it amplifying the error.

And that same condition number also appeared in the convergence analysis of the conjugate gradient method except it appeared with a $\sqrt{\kappa}$ in steepest descent it appeared with a κ . So, it is not blowing up that badly. So, that is why condition number comes up. Why is most of deep learning steepest descent based or their modifications. If CGM has guaranteed convergence in n steps and has faster convergence what is stopping us from using it that is one question.

The second question is why would you ever use gradient descent when CGM seems to obviously be faster and better, legitimate questions right. We saw that CGM was just beating steepest descent right. Now, it is a so there are few answers to it. One is the steepest descent method the way we formulated it, what did we need? For example, if you wanted to calculate do

backtracking line search, everyone remembers what is steepest what are the ingredients of steepest descent? I pick a negative gradient direction that is the direction to go into and then I need to know how much to walk that is the step length. In a real life problem I cannot do exact step length.

So, what do I do? inexact line search. An example of implementing inexact line search was with our backtracking line search. and to know whether the step length is good enough or not what did I do in backtracking line search I said sufficient decrease or curvature condition the wolf condition that is how I got a good enough step length that is all that I needed in steepest descent. So, nowhere did I say that f has to be a $\|Ax - b\|^2$ where A is positive definite. So, steepest descent is applicable to a broader variety of problems.

So far what we have done for conjugate gradient is we are trying to solve $Ax = b$ where A is positive symmetric positive definite. So, the class is restricted, but then as we will see today hopefully that there is a version of CG which can be adapted to the non-linear non-linear function it need not be $\|Ax - b\|^2$ it can be some general function. you should compare steepest descent with non-linear CG if you wanted a fair comparison, right. A non-linear CG again will you will notice will be will have a line search, will have all of these things. So, the convergence may not be as good, ok.

The other thing is that steepest descent the way I have shown you is not how it is implemented in ML applications. do plane steepest descent, they do something called accelerated gradient descent or there are momentum based steepest descent. There are different cousins of it which have better convergence properties than steepest descent right. Now, if we start going into covering all those different applications, we will spend a whole semester you know discussing cousins of steepest descent, but having understood steepest descent you can easily jump to and understand accelerated gradient descent or whatever right how to do it. So, that would be a more fair comparison between accelerated versions of steepest descent and CG.

Ok. The other issue is I mean the other second level answer to this is I do not know enough, this is actually a very active area of research, how people apply or adapt CG in the machine learning world is something that you can actually explore as part of the course project ok. Similar to this you will again have another question when we do the next module of the course on Newton methods and Newton like methods, they have even faster conversions. Yet its use in ML is close to non-existent. Now, why is that a big strong requirement is if any method requires a Hessian it is very expensive. So, no large scale problem will use Hessian then you step down to Newton like methods which are called quasi Newton methods which do not compute a Hessian.

So, there you will find some work in the research in world where there is Newton like methods quasi Newton methods being applied to AI ML ok. This can be one more of the projects which you explore in the course project because it is I mean if we start doing all of that we will not cover other important aspects of the course. For example, very important thing we will cover probably after quiz 2 is constraint optimization. So, with limited time this is what we have to deal with and this is going to be of interest to everyone.

How can CG and quasi Newton methods be made applicable to AI ML applications? I mean really speaking you guys can be the ones that write papers on this. So, we are just laying the

foundations over here, so that you can springboard into that it is all up for grabs at this point.
Any questions? Okay.