

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 - 48

Non Linear Conjugate Gradient method

OK, should we start? I'm going to start with the doubt sheets. OK, I guess this goes to the initial motivation. Maybe I should open that folder. Yesterday when we were talking about preconditioned CG, we came up with, where is it gone? Yeah. is \hat{A} symmetric positive definite? We can clearly see that it is positive definite. The question is why is this symmetric? Why is $Z^T \hat{A} Z$ symmetric? Sorry, why is \hat{A} a symmetric matrix? If I want to test what is the signature for a symmetric matrix? \hat{A} , the transpose should be equal.

So, is \hat{A}^T equal to \hat{A} ? That is the question. So, what is \hat{A} defined as? $L^T A L$. So, I need to take its transpose. So the transpose will just flip it all around, right? What will I get? $L^T A L$.

NPTEL

Nonlinear Conjugate Gradient

Usual CG

$$\phi(x) = \frac{1}{2} x^T A x - b^T x \quad \leftrightarrow \quad A x = b.$$

Can ϕ be a general convex fn.

That is straightforward. Okay. Right. If we can use the SVD to get a low rank approximation of A , remember the problem was that A had a bad condition number. So the student is asking if we use SVD to get a low rank approximation of A , it would also decrease the condition number.

Can we use this low rank approximation in the conjugate gradient method and get similar offset as effect as preconditioning? What do you think? SVD itself is what complexity? So if I am going to do an SVD to get a low rank approximation, I have already done the right. So as it is, it is not a good idea. However, there are some inexpensive techniques to get an approximation, like

you do not have to do the entire SVD. You could just get a few of the eigenvectors. So those again fall into the domain of linear algebra.

You could use those techniques to get some approximation, right, which has a better condition. Remember, we don't need something very, very exact. We just need some trick to improve the condition number so that our iterations speed up. So again, this falls in the domain of numerical linear algebra, which we are not going to go into detail. So Omkar's question is, what is the order of the difference between A and KK^T , right? The incomplete Cholesky and the actual matrix.

Can I... Can the magnitude of this difference be limited while maintaining the sparseness of K and the low computational cost? That's one question. A related question is why does incomplete Cholesky give a sparse matrix only? And another is how do you compute the incomplete Cholesky decomposition? These are all questions of numerical linear algebra. We are not going to do it in this course.

The algorithm itself of incomplete Cholesky is quite straightforward. You Wikipedia it, you get a pseudo code. If you wanted to just implement it, you can. Okay. Understanding would require reading a text in numerical linear algebra.

NPTEL

Check new procedure $\rightarrow p^{(k)} x_0$

- ① Calc $\nabla f(x_0) = r_0 = -p_0$
- ② Need $\alpha_0 \rightarrow$ inexact LS $\rightarrow x_1$
- ③ $p_{k+1} = -\nabla f_{k+1} + \beta_{k+1} p_k$

$$\beta_{k+1} = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$$

OPTIMIZATION THEORY AND ALGORITHMS

This would not be done in a first course on linear algebra. This is numerical linear algebra. I think the math department has a course for it. Calling a matrix with a bad condition number means you have calculated it, which in turn requires calculating A^{-1} .

So, what the student is saying is that the definition of the condition number is... So, how will I know the condition number unless I calculate A^{-1} , and therefore, it seems like a bad idea? So, remember this is the definition of condition number. Right? This is not in practice how you will calculate the condition number numerically. Again, it is a whole branch of numerical linear

algebra to calculate what this condition number is without obviously expending order n^3 . So no one is going to do this.

This is as good as writing the solution when I write $A = b$, sorry $b = A^{-1}x$. This is the symbolic way of writing the solution. It doesn't mean I've actually calculated A^{-1} and then multiplied by b , right? That was class 12, but that's not how it's done. So, this is just a formula. How to implement it is, again, a fairly rich field in its own, okay? And remember, in what we did last class, we are never calculating.

The image shows a whiteboard with handwritten notes. At the top left is the NPTEL logo. The main text reads: "Problem: A-conjugacy \rightarrow GRWE!". Below this, it says " P_k needs to be a legitimate descent direction!". A diagram shows a vector P_k and a vector $-\nabla f_k$ originating from the same point, with an angle θ_k between them. Below the diagram, the inequality $-\pi/2 < \theta_k < \pi/2$ is written. The update equation is given as $P_k = -\nabla f_k + \beta_k P_{k-1}$. To the right, it says "LM by ∇f_k^T ". Below the update equation, the equation $\nabla f_k^T P_k = -\|\nabla f_k\|^2 + \beta_k \nabla f_k^T P_{k-1}$ is written. In the bottom left corner, there is a small inset image of a man speaking.

We had this incomplete Cholesky decomposition, right? We never calculated any inverse anywhere. We worked that out in detail. So that cost has been saved. Any other questions not discussed just now? So what we have done so far is we have completed the basic version of the conjugate direction method. We saw how that gives the conjugate gradient method.

And then we gave a practical version of the conjugate gradient method, which is the preconditioned conjugate gradient. So this completes the description of the conjugate gradient method. And now you could implement it decently. The next version that we are going to look at is the nonlinear conjugate gradient method.

Okay. Question? So in your usual conjugate gradient method, usual CG, the cost function which we had written was a quadratic form, right? And we had written it as $\frac{1}{2}x^T Ax - b^T x$, right? And we said this is the same as effectively solving $Ax = b$. Why? Because $\nabla_x = 0$ was $Ax - b$. Now, having experienced the success of the usual CG method, the question that comes is, can ϕ be a general convex function? What do I mean by general convex function? It means not just a quadratic form.

There can be other functions which are convex but not quadratic forms. Remember the definition of convexity. I have drawn the graph and the function value lies below. So, this is one question. The other question is, can ϕ be a non-linear function?

So, this is if you pose this question, the answer turns out to be a non-linear function. Yes. But we have to obviously make some modifications for this to work. And these modifications are sort of more or less common sense, right? So, what changes will we have to make? Can you think of, looking at the recipe of the usual CG, what would be the first or one of the things that changes? So, what was the special property of the quadratic form that got used in the CG method? Positive definite, but now we no longer have positive definite. Okay, so positive definite is out.

Say that I did exact LS.

$\phi(\alpha) = f(x_{k-1} + \alpha p_{k-1})$

want best $\alpha \rightarrow \alpha_{k-1}$

$\Rightarrow \frac{d\phi(\alpha)}{d\alpha} = 0 = \nabla f(x_{k-1} + \alpha p_{k-1})^T p_{k-1}$

$0 = \nabla f_k^T p_{k-1}$

OPTIMIZATION THEORY AND ALGORITHMS

Some general function has come to us. How did we calculate the step length in CG? It was exact step length, right? Can I do that here? I cannot do that here. Why? Because, I mean, I can do it, but it's going to be very expensive. So, I have to give up exact line search and replace it with an inexact line search. So, step length, exact line search not possible.

So, I have to give up something. The other small modification that I will have to make is, remember the expression for the residual. The residual was what? $Ax - b$. Now there is no A and b , right? So, what should I, if you were in charge of affairs, what would you replace the residual expression by? $\nabla f(x)$, right? So that is the second change that you need to do. Replace $r_x = \nabla f(x)$, okay? All right, now let us say I make these two small changes.

Now let us try to just list out the modified CG method, okay? So let us see, do I have all the nuts and bolts in place for this? Let us see, right? So, check the new procedure. Step 1 was what? In our usual CG, I have picked a, let us say I have picked x_0 . That is randomly chosen. What would be the next step? $\nabla f(x_0)$, okay, I can calculate $\nabla f(x_0)$, okay. Obviously, I have access to calculating the function and gradient value.

So, I calculate $\nabla f(x)$. How does this help me? This gives me r_0 which is related to $-p_0$, right. So I have got my initial direction to go into, okay. What do I do with this next? I have got the direction in which I have to go to. What do I need to go to x_1 ? α_0 , right.

So, therefore I need α_0 . How am I going to get this? Step 1, I mean the first thing over here, right? I don't have an analytical expression for α , no problem. I replace it by a module of inexact line search. I can do that, right? So, inexact line search. Right? That took me to x_1 .

Okay? Now I need to update my p . Right. So, what was my original expression for p_{k+1} ? We said that it is related to the residual or in this case now the gradient plus some linear coefficient times the previous search direction. Now if you look at the expression for β_{k+1} , if I just replace the residual by ∇f , what will I get? So is my recipe, modified recipe now complete? Do I have everything in place? We are not getting into the proof that this is going to work or not, but will the recipe as it is work? I've got a way to calculate my α . I've got an expression for β , which gives me my new conjugate direction. I can now rinse and repeat and go through the iterations.

$d\alpha$

$$0 = \nabla f_k^T p_{k-1}$$

Then $\nabla f_k^T p_k = -\|\nabla f_k\|^2$

\Rightarrow If I do exact LS, & start with a DD

\Rightarrow all p 's are DD. ✓

practical cases, exact LS

Would you agree? Right? We were all several decades late to the party. So, two people by the name of Fletcher and Reeves made this small modification. And the first nonlinear CG version is named after them. So this is basically what I've outlined over here is Fletcher, Reeves, we're going to call this NLCG.

So it's very simple. I mean, it looks very simple because minimum modifications to the usual linear CG have been given. What is missing over here? We have no idea that this is going to work. So what is the big difference now between linear CG and non-linear CG that we have glossed over? What was critical in the case of linear CG? Positive definite. Positive definite which was used for doing what? for making the conjugate directions.


The p 's were conjugate with respect to A . Now there is no A . So the meaning of conjugacy itself is no longer valid because I have a nonlinear function in general. So these are the things that we need to look at a little bit more. But is everyone clear on the minimum modifications? I just replaced residual by $\nabla f(x)$ and the rest of the procedure goes along with the inexact line search.

So the problem is A conjugacy is gone. If conjugacy is gone, what can I do? How can I still save the story? The hint is think of gradient descent. In gradient descent, I did not have conjugacy. I had something else. What was that something else? Orthogonal.

Orthogonal to? Not orthogonal. That was a consequence. But what is the more primary property? What did I ask for the search direction? It should be a legitimate descent direction, right? In gradient descent was a special case where it was exactly in the negative gradient. But in general, remember that $\cos\theta$, as long as I am in making an acute angle with the negative gradient, I am guaranteed to first order that my function value is going to decrease. And we call that a legitimate descent direction. So since A conjugacy is gone, we at least should have something.

So that something is to fall back on ensuring that p is a legitimate descent direction. So p_k needs to be a legitimate descent direction. So, just to remind you graphically, if this was my ∇f_k , then my θ_k had what range from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$. That was the meaning of a legitimate descending direction, just as a refresher. Now let us get into a little bit more of the details. So what would, so I am just going to rewrite the update equation for the p .

Am I satisfying this at the start of iterations with p_0 ? I am trivially because I am starting with $p_0 = -r_0$ which is $-\nabla f(x)$. So I am satisfying it at step, at the starting which is graded. But how do I know that as my iterations go, my p 's continue to be legitimate descent directions? I have to ensure that otherwise the whole thing will fall apart.



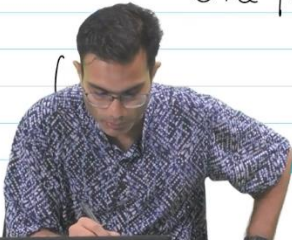
Then $\nabla f_k^T p_k = -\|\nabla f_k\|^2$

\Rightarrow If I do exact LS, & start with a DD

\Rightarrow all p 's are DD. ✓

In practical cases, exact LS X

The fix: Impose strong Wolfe Conditions
with $0 < c_1 < c_2 < \frac{1}{2}$.



So let us look at the update equation. Which was what? Let us write it for p_k . So $p_k = -\nabla f_k + \beta_k p_{k-1}$. Okay. So if I want to test a legitimate descent direction, what do I do? Dot product with $\nabla f(x)$, right? So let us left multiply by ∇f^\top , okay? What is the first term on the right-hand side? $-\|\nabla f_k\|^2$.

Is this good news or bad news? It is good news. Why? Because this term is always going to be negative. So that is already good. And then I have $\beta_k \nabla f_k^\top p_{k-1}$. So this is, on the other hand, this term could potentially be problematic. Why? Because if this exceeds $\|\nabla f_k\|^2$, then what will happen? p_k will no longer be a descent direction.

So, looking at this term is really one of the tricky parts of non-linear CG. Okay. So to study it, first we will study it in a more theoretical way. Then we will see practically what is to be done about it. So let us say, just for sake of argument, say that I did exact line search somehow.

Okay. So let us say that here I am at x_{k-1} , and this is my p_{k-1} and I arrive at x_k , okay. So I am choosing x_{k-1} because I want to evaluate this term over here. There is a p_{k-1} , right. p_{k-1} is involved in the journey from x_{k-1} to x_k .

That is why I am writing this away, right. So these are the two points over here, okay. So if I were standing over here at x_{k-1} , what is the function that I am looking at? Like I want to find out the α that gets me to x_k . So if you remember we had written it as a function of one variable. This was $f(x_{k-1} + \alpha p_{k-1})$.

And we were hunting for the best α . Right? And whatever best α I get, what will I call that? I will call that α_{k-1} . Once I find it, I will call it α_{k-1} , whether by exact or inexact line search. Let us now for sake of argument assume that I did exact line search. Now what was the defining feature of an exact line search? $\frac{d\phi}{d\alpha}$ should be what? 0, right. So if I did exact line search, it would imply that $\frac{d\phi}{d\alpha} = 0$. By the multivariate chain rule, what happens to the right-hand side? $\nabla f_k^\top p_{k-1}$.

Now, this whole expression over here, what is the short form notation for this? Is this ∇f_k or ∇f_{k-1} or nothing? This is nothing but ∇f_k . Sorry, this is ∇f_k because this expression over here is nothing but x_k .

So this is $\nabla f_k^\top p_{k-1}$, and this should be equal to 0. Does this look familiar? Right? It's the exact same character and the right-hand side of this. So simple argument, right? It assumes that I did exact line search. If I did exact line search, this term is 0. If this term is 0, then the expression on the top over here, $\nabla f_k^\top p_k$ became equal to $-\|\nabla f_k\|^2$, right? So that is great news. This term is always guaranteed to be negative, right? So if I were to say this in plain English, what is it, how would you write this observation in plain English? Question? Correct, I agree with you.

We are not doing, I am starting with the ideal case and then I will make it non-ideal. In the ideal case, if I were doing exact line search, what is the consequence over here in plain English? It is a descent direction, but it is saying that if I start with a legitimate descent direction, the next p_k is also a legitimate descent direction. So if I do exact line search and I start with a legitimate descent direction, all along the way I keep getting descent directions. So that is great.

That is giving me a clear way forward of implementing a nonlinear CG. Make sure I do exact line search, no matter how expensive it is, and pick a descent direction to start with, then every step will be this, right? So let's just note that down. If I do exact line search and start with a DD, which is a descent direction, it implies all p 's are DD. Why? Because they always make a negative, I mean, acute angle with the gradient of $-\nabla f$ direction. Now, of course, as was pointed out, if I'm doing nonlinear CG and my function is expensive to evaluate, obviously, I don't have the luxury of exact line search. So in practical cases, you can say that exact line search is not possible.

Now, thankfully for us, there is a lot of analysis that has been done on this problem. So there is, I am just going to state the theorem statement that helps us, okay. And as you can probably guess, it has something to do with the Wolfe conditions. So if you do the Wolfe conditions in a certain way, even though you do not do exact line search, you continue to have legitimate descent direction.

So I will state it here without proof, okay. So the fix is to impose the strong Wolfe conditions with the following. C_1 came from which Wolfe condition? Sufficient decrease. C_2 came from? Curvature condition. And here it is not just the curvature condition, it is the strong Wolfe conditions.

Remember we put the mod sign. So that was our C_2 . Earlier we had said when we first learned the Wolfe conditions, $C_1 < C_2 < 1$. Here it has become a little bit more conservative. It has been made half. So this is without proof being stated. If you impose this, then all directions are very, very simple to implement condition, right? So that is why it gets implemented a lot in non-linear optimization, right? So that's all we need to do.

We give up our exact line search. We replace it with an inexact line search. Make sure that the Wolfe conditions are followed in this way. And I go back to this recipe over here, the Fletcher Reeves recipe over here. Start with $-\nabla f$ Walk through the steps and everything works out. Now it is hard to, so this is one of the first variants, one of the first versions of non-linear CG. As you can imagine, there are, when people would have implemented it, there were some issues that came up.

So there are actually several versions or you can say cousins of non-linear CG which are there. I am going to cover only one more. There are several more but I will cover the two main ones. This is the first one that you should know because it is the closest related to the linear CG. What happens is when you look at this situation over here, the geometry of p_k and $-\nabla f_k$.

Legitimate descent direction means that this angle should be acute, right. Now it is in many problems observed. So I am giving you an observation. I am not proving it or anything, an empirical observation that if you take this Fletcher Reeves method, after several iterations, what starts happening is that this angle θ_k tends to $\frac{\pi}{2}$. Now if it tends to $\frac{\pi}{2}$, what's going to happen? I kind of get stuck because I'm not proceeding in a descent direction.


So I'm kind of going along the contours of the cost function. And so progress is stopped. So let's just make a note of this. Often θ_k tends to... It is still strictly less than $\frac{\pi}{2}$. So it is a legitimate descent direction.

But if that angle becomes really, really small, then I cannot progress. Okay? So knowing that this nonlinear CG has sort of been cobbled together like a, almost like a cooking recipe, if you were in charge of fixing it, how would you fix it? It does not require any fancy math. So let me keep this over here. You can in fact show that if this descent direction, if θ_k is tending to $\frac{\pi}{2}$, it can be shown that p_{k+1} and p_k , they become almost the same.

So then what can we do? Decrease β_k . That's a great idea. How much should I decrease it to? That's all there is to it, right? So supposing I make, supposing p_{k+1} is turning out to be equal to p_k , almost equal to p_k . What is top, I'm not bound to this β_k parameter. I've defined it from looking at my linear CG.


Nothing is holding me to it. So what should I do? If I said, he said, let's reduce β_k . Let's make it, take it all the way. If I make it, drive it down to 0. What happens to my p_{k+1} ? It becomes $-\nabla f_{k+1}$ which is a great descent direction, right?

So that is actually what is done. So set β_k equal to 0. Rather this is called a reset because it is resetting the descent direction, okay. Again we were late on the scene, so the guys who figured this out got their name to it. Polak-Ribiere, you can never get the I's and the E's correct. So what this method is doing, it is basically just a small variation of the Fletcher Reeves CG and I keep monitoring θ_k or $\cos\theta_k$.



One fix: impose strong Wolfe conditions
 (w/o proof) with $0 < c_1 < c_2 < \frac{1}{2}$.
 then all dirs are descent dirs!

↳ Often $\theta_k \rightarrow \frac{\pi}{2}$. Polak-Ribiere
 → Set $\beta_k = 0$ (Reset) NL CG.



As this θ_k tends to $\frac{\pi}{2}$, I do a reset and then the same recipe follows along, okay. There is this expression for β_k . In this, the only loose kind of thing is how do you define β_k , right? This has been written down by observing the linear CG. So there are people who have come up with different and more efficient expressions for β_k . And so that has given rise to a full family of nonlinear CG methods, okay.

The theoretical analysis of it is quite difficult, so people will do it heuristically. You, you know, you try several different versions and you see it will work well on a nonlinear, sorry, the Polack-Ribiere may work well on one problem, Fletcher-Reeves will work well on another problem. So, it is hard to say anything very definitive about it, but these are the options available. If you use a numerical linear algebra library, they will give you headers for all of these different versions for you to try out, okay, for that reason. So, this is in a nutshell your nonlinear CG method okay. It is the only theoretical thing of importance over here which we did not derive was the fact that this one over here.

As long as I maintain this Wolfe condition with $C_2 < \frac{1}{2}$, everything goes through. So, if you have a nonlinear method to optimize, here is a very, very simple recipe that you could do. What is the other simple recipe that we have? Gradient descent, gradient descent. Again, it is a heuristic. So it, θ_k is never going to become $\frac{\pi}{2}$, right.

So it is going to become, so you can put a threshold. If $\left| \theta_k - \frac{\pi}{2} \right|$ falls below a certain threshold, you reset. Correct. You reset it.

For one iteration you set it to 0. And then you let it, then you use the rest of the formulas. If you do always 0, yeah, yeah. This is, yeah. So this reset is, only one iteration.

Not for all iterations. If you do it for all iterations, it becomes gradient descent. Yeah, yeah, yeah. So it is a reset. I set it to 0. It is like starting from p_0 .

Now I follow the rest of the recipe. So it is like something is getting back, you reset it.

Something is getting back, you keep resetting. So yeah. For one iteration only. So only one iteration. OK. So, you know, it will be good if some of you can try to just write a simple code for this. Take a nonlinear function. In fact, there are benchmark nonlinear functions available if you Google for it, right? Functions which are known to be very notorious nonlinear functions. So people benchmark their nonlinear optimization methods against these known things.

And then when I publish my results, you can see which method is doing better. If I come up with a new method, I can say, look, hey, against this benchmark, my method works well. So I'll post a link to this library of ugly functions that you can try out your various ideas on. Any thoughts or questions on this? It's fairly descriptive. There's not much derivation involved over here. So it is what it is.