**PHILIPPE RIGOLLET:** All right. So let's continue talking about maximum likelihood estimation in the context of generalized linear models, all right? So in those generalized linear models, what we spent most of the past lectures working on is the conditional distribution of Y given X. And we're going to assume that this follows some distribution in the exponential family.

OK. And so what it means is that if we look at the density, say-- or the PMF, but let's talk about density to make things clearer-- we're going to assume that Y given X has distribution. So X is now fixed, because we're conditioning on it. And it has a density, which is of this form, c of Yi phi.

OK. So this c, again, we don't really need to think about it. This is something that's going to come up naturally as soon as you need normalization factor. And so here what it means, if this is the distribution of Y given Xi, so that's the density of Yi given Xi is equal to little xi.

So if it's the conditional distribution of Yi given Xi, it should depend on xi somehow. And it does not appear to depend on Xi. And here, the model is going to be on theta i, which is just a function, theta i of Xi. And we're going to take a very specific one. It's going to be a function of a linear form of the Xi.

So really we're going to take something which is of the form theta I, which is really just-- as theta does not depend on Xi-- of Xi transposed from beta. OK? So all these parts here, this is really some modeling assumptions that we're making once we've agreed on what distribution we want. OK. So to do that, our goal, of course, is going to try to understand what this beta is.

There's one beta here. What's important is that this beta does not depend on i. So if they observe pairs Xi, Yi-- let's say I observe n of them, i equals 1 to n-- the hope is that as a accumulate more and more pairs of this form where there's always the same parameter that links Xi to Yi, that's this parameter beta, that I should have a better and better estimation of this beta. Because it's always the same. And that's essentially with couples all of our distribution.

If I did not assume this, then I could have a different distribution for each pair Xi given YI. And I would not be able to do any statistics. Nothing would average in the end. But here I have the same beta, which means that I can hope to do statistics and average errors in them.

OK. So I'm going to collect, so I'll come back to this. But as usual in the linear regression model, we're going to collect all our observations Yi. So I'm going to assume that they're real valued and that my Xi's takes value in Rp just like in the regression model. And I'm going to collect all my Yi's into one big vector of Y in our n and all my X's into one big matrix in Rn times p just like for the linear regression model.

All right, so, again, what I'm interested in here is the conditional distribution of Yi given Xi. OK. I said this is this distribution. When we're talking about regression, I defined last time what the definition of regression function was. It's just one particular aspect of this conventional distribution. It's the conditional expectation of Yi given Xi.

OK. And so this conditional expectation, I will denote it by-- so I talk about the conditional, I'm going to call it, say, mu i, which is the conditional expectation of Yi given Xi equals some little xi, say. You can forget about this part if you find it confusing. It really doesn't matter. It's just that this means that this is a function of little xi. But if I only had the expectation of Yi given big Xi, this would be just a function of big Xi.

So it really doesn't change anything. It's just a matter of notation. OK. So just forget about this part. But I'll just do it like that here. OK. So this is just the conditional expectation of Yi given Xi.

It just depends on Xi, so I think it depends on i, and so I will call it mu i. But I know that since in a canonical exponential family, then I know that mu i is actually B prime of theta i. OK. So there's a 1 to 1 link between the canonical parameter of my exponential family and the mean mu i, the conditional expectation.

And the modeling assumption we're going to make is not directly-- remember, that was the second aspect of the generalized linear model. We're not going to assume that theta i itself directly depends on Xi. We're going to assume that mu i has a particular dependence on Xi through the link function.

So, again, we're back to modeling. So we have a link function g. And we assume that mu i depends on Xi as follows.

g of mu i-- and remember, all g does for us is really map the space in which mu i lives, which could be just the interval 0, 1 to the entire real line, all right? And we're going to assume that this thing that lives in the real line is just Xi transpose beta. I should maybe put a small one, Xi transpose beta. OK?

So we're making, indeed, some modeling assumption. But compared to in the linear regression model, we only assume that mu i was Xi transpose beta. So if you want to make a parallel between generalized linear models and linear model is the only difference is that g is not the identity necessarily in this case.

And all the g does for us is to just make this thing compatible, that those two things on the left and the right of equality live in the same space. So in a way, we're not making a much bigger leap of faith by assuming a linear model. The linear link is already here. We're just making things compatible, all right?

And so it's always the same link function. So now if I want to go back to beta-- right, because I'm going to want to express my likelihood-- if I were to express my likelihood from this, it would just be a function of theta, right? And so if I want to maximize my likelihood, I don't want to maximize it in theta. I want to maximize it in beta.

So if I can write my density as a function of beta, then I will be able to write my likelihood as a function of beta, and then talk about my maximum likelihood estimator. And so all they need to do is to just say, OK, how do I replace theta by-- I know that theta is a function of beta, right? I wrote it here.

So the question is, what is this function? And I actually have access to all of this. So what I know is that theta-- right, so mu is b prime of theta, which means that theta i is b prime inverse of mu i.

OK. So that's what we've got from this derivative of the log likelihood equal to 0. That give us this guy inverted. And now I know that mu i is g inverse of Xi beta.

So this composition of b prime inverse and g inverse is actually just the composition of g with b prime. Everybody's comfortable with this notation, the little circle? Any question about this?

It just means that I first applied b prime. Well, actually, it's D inverse. But if I look at a function g composed with b prime, I first applied the g b prime of x, is just g of b prime of x. OK. And then I take the inverse of this function, which is first take g inverse, and then take b prime inverse.

OK. So now I have everywhere I saw theta, now I see this function of beta. So I could technically plug that in. Of course, it's a little painful to have to write g circle beta prime all the time. So I'm going to give this guy a name.

And so you're just going to define h, which is g b prime inverse so that theta i is simply h of Xi transpose beta. OK I could give it a name, you know. But let's just call that the h function.

And something which is nice about this h function is that if g is the canonical link-- what is the canonical link? So what is it canonical to? A canonical link, it's canonical to a particular distribution in the canonical exponential family, right?

A canonical exponential family is completely characterized by the function b. Which means that if I want to talk about the canonical link, all I need to tell you is how it depends on b. So what is g as a function of b?

**AUDIENCE:** [INAUDIBLE]

**PHILIPPE RIGOLLET:** b inverse. b prime inverse, right? So this is g is equal to b prime inverse, which means that if g is composed with b prime that means that this is just the identity. So h is the identity. So h of Xi transpose beta is simply Xi transpose beta.

And it's true that the way we introduce the canonical link was just the function for which we model directly theta i as Xi transpose beta, which we can read off here, right? So theta i is simply Xi transpose beta. So now, for example, if I go back to my log-likelihood, so if I look log-likelihood, the log-likelihood is sum of the log of the densities.

So it's sum from i equal 1 to n of log of exponential Yi theta i minus b theta i divided by phi plus c of Yi phi. So this term does not depend on theta. So I have two things.

First of all, the log and the exponential are going to cancel each other. And second, I actually know that theta is just a function of beta. And it has this form. Theta i is h of Xi transpose beta.

And that's my modeling assumption. So this is actually equal to the sum from i equal 1 to n of Yi. And then here I'm going to write h of Xi transpose beta minus b of h of Xi transpose beta divided by phi. And then I have, again, this function c of Yi phi, which again won't matter.

Because when I'm going to try to maximize this thing, this is just playing the role of a constant that's shifting the entire function. In particular, your max is going to be exactly what it was. OK? So this thing is really not going to matter for me. I'm keeping track of it.

And actually, if you look here, it's gone, right? It's gone, because it does not matter. So let's just pretend it's not here, because it won't matter when I'm trying to maximize the likelihood. OK? While it's here up to constant term, it says. That's the constant term.

All right, any question? All I'm doing here is replacing my likelihood as a function of theta i's. So if I had one theta i per observation, again, this would not help me very much.

But if I assume that they are all linked together by saying that theta i is of the form Xi transpose beta or h of Xi transpose beta if I'm not using the canonical link, then I can hope to make some estimation. And so, again, if I have the canonical link, h is the identity. So I'm left only with Yi times Xi transpose beta. And then I have b of Xi transpose beta and not b composed with h, because h is the identity, which is fairly simple, right?

Why is it simple? Well, let's actually focus on this guy for one second. So let me write it down, so we know what we're talking about.

So we just showed that the log-likelihood when I use the canonical link-- so that h is equal to the identity, the log-likelihood actually takes the form ln. And it depends on a bunch of stuff. But let's just make it depend only on the parameter that we care about, which is beta, all right?

So this is of the form l of beta and that's equal to what? It's the sum from i equal 1 to n of Yi Xi transpose beta minus-- let me put the phi here. And then I'm going to have minus b of Xi transpose beta.

OK. And phi we know is some known positive term. So again, optimizing a function plus some constant or optimizing of function time as a constant, that's not going to change much either. So it won't really matter to think about whether this phi is here or not.

But let's just think about what this function looks like. I'm trying to maximize a function. I'm trying to maximize a log-likelihood. If it looked like this, that would be a serious problem. But we can do like a basic, you know, back of the envelope guess of what the variations of this function is. This first term here is-- as a function of beta, what kind of function is it?

**AUDIENCE:** Linear.

**PHILIPPE RIGOLLET:** It's linear, right? This is just Xi transpose beta. If I multiply beta by 2, I get twice. If I add something to beta, it just gets added, so it's a linear function of beta.

And so this thing is both convex and concave. In the one-dimensional case-- so think about p as being one-dimensional-- so if beta is a one-dimensional thing, those are just the function that looks like this, right? Those are linear functions. They are both convex and concave.

So this is not going to matter when it comes to the convexity of my overall function, because I'm just adding something which is just a line. And so if I started with convex, it's going to stay convex. If I started with concave, it's going to stay concave. And if I started with something which is both, it's going to stay both, meaning neither. It cannot be both.

Yeah. So if you're neither convex or concave, adding this linear-- so this will not really matter. If I want to understand when my function looks like, I need to understand would b of Xi transpose beta does. Begin, the Xi transpose beta-- no impact. It's a linear function.

In terms of convexity, it's not going to play any role. So I really need to understand what my function b looks like. What do we know about b again?

So we know that b prime of theta is equal to mu, right? Well, the mean of a random variable in a canonical exponential family can be a positive or negative number. This really does not tell me anything. That can be really anything.

However, if I look at the second the derivative of b, I know that this is what? This is the variance of Y divided by phi. That was my dispersion parameter. The variance was equal to phi times b prime prime.

So we know that if theta is not degenerate, meaning that the density does not take value infinity at only one point, this thing is actually positive. And clearly, when you have something that looks like this, unless you have some crazy stuff happening with phi being equal to 0 or anything that's not normal, then you will see that you're not degenerate. So this think is strictly positive.

And we've said several times that if b prime prime is positive, then that means that's the derivative of b prime, meaning that b prime is increasing. And b prime is increasing is just the same thing as saying that b is convex, All right? So that implies that b is strictly convex.

And the strictly comes from the fact that this is a strict sign. Well, I should not do that, because now it's no longer. So it's just a strict sign, meaning that the function, this is not strictly convex, because it's linear. Strictly convex means there's always some curvature everywhere.

So now I have this thing that's linear minus something that's convex. Something that's negative, something convex, is concave. So this thing is linear plus concave. So it is concave.

So I know just by looking at this that ln of beta, which, of course, is something that lives in Rp, but if I saw it living in R1 it would look like this. And if I saw it living in R2, it would look like a dome like this. And the fact that it's strict is also telling me that it is actually a unique maximizer.

So there's unique maximizer in Xi transpose beta, but not in beta necessarily. We're going to need extra assumptions for this. OK. So this is what I say here. The log-likelihood is strictly concave.

And so as a consequence under extra assumptions on the Xi is because, of course, if the Xi's are all the same, right? So if the entries of Xi's-- so if Xi is equal to 1, 1, 1, 1, 1, then Xi transpose beta is just the sum of the betas. And of the beta i's, I will be strictly concaving those guys, but certainly not in the individual entries.

OK. So I need extra thing on my Xi, so that this happens, just like we needed the matrix capital X in the linear regression case to be a full rank, so we could actually identify would beta was. OK. It's going to be exactly the same thing. So here, this is when we have this very specific parametrization.

And the question is-- but it may not be the case if we change the parameter beta into something else. OK. So here, the fact that we use the canonical link, et cetera, everything actually works really to our advantage, so that everything becomes strictly concave. And we know exactly what's happening.

All right, so I understand I went a bit fast on playing with convex and concave functions. This is not the purpose. You know, I could spend a lecture telling you, oh, if I add two concave functions, then the result remains concave.

If I had a concave and a strictly concave, then the result still remains strictly concave. And we could spend time proving this. This was just for you to get an intuition as to why this is correct. But we don't really have time to go into too much detail.

One thing you can do-- a strictly concave function, if it's in one dimension, all I need to have is that the second derivative is strictly negative, right? That's a strictly concave function. That was the analytic definition we had for strict concavity.

So if this was in one dimension, it would look like this, Yi times Xi times beta. Now, beta is just one number. And then I would have minus beta Xi times b. And this is all over phi.

You take second derivatives. The fact that this is linear in beta, this is going to go away. And here, I'm just going to be left with minus-- so if I take the second derivative with respect to beta, this is going to be equal to minus b prime prime Xi beta times Xi squared divided by phi.

So this is clearly positive. If Xi is 0, this is degenerate, so I would not get it. Then I have the second derivative of b prime, which I know is positive, because of the variance thing that I have here, divided by phi. And so that would all be fine.

That's for one dimension. If I wanted to do this in higher dimensions, I would have to say that the Hessian is a positive definite matrix. And that's maybe a bit beyond what this course is.

So in the rest of this chapter, I will do what I did not do when we talked about maximum likelihood. And what we're going to do is we're going to actually show how to do this maximization, right? So here, we know that the function is concave. But what it looks like specifically depends on what b is.

And for different b's, I'm going to have different things to do, Just like when I was talking about maximum likelihood estimation, if it had a concave log-likelihood function, I could optimize it. But depending on what the function is, I would actually need some algorithms that may be working better on some functions than others.

Now, here I don't have random things. I have the b is the cumulant generating function of a canonical exponential family. And there is a way for me to sort of leverage that. So not only is there the b part, but there's also the linear part.

And if I start trying to use that, I'm actually going to be able to devise very specific optimization algorithms. And the way I'm going to be able to do this is by thinking of simple black box optimization to which I can actually technically feed any function. But it's going to turn out that the iterations of this iterative algorithms are going to look very familiar when we just plug in the particular values of b, of the log-likelihood that we have for this problem.

And so the three methods we're going to talk about going from more black box-- meaning you can basically stuff in any function that's going to work, any concave function that's going to work, all the way to this is working specifically for generalized linear models-- are Newton-Raphson method. Who's already heard about the Newton-Raphson method? So there's probably some people actually learned this algorithm without even knowing the word algorithm, right?

It's a function. Typically, it's supposed to be finding roots of functions. But finding the root of a function of the derivative is the same as finding the minimum of a function. So that's the first black box method. I mean, it's pretty old.

And then there's something that's very specific to what we're doing, which is called-- so this Newton-Raphson method is going to involve the Hessian of our log-likelihood. And since we know something about the Hessian for a particular problem, we're going to be able move one to Fisher-scoring. And the word Fisher here is actually exactly coming from Fisher information. So the Hessian is going to involve the Fisher information.

And finally, we will talk about iteratively re-weighted least squares. And that's not for any function. It's really when we're trying to use the fact that there is this linear dependence on the Xi. And this is essentially going to tell us, well, you know, you can use least squares for linear regression. Here, you can use least squares, but locally, and you have to iterate.

OK. And this last part is essentially a trick by statisticians to be able to solve the Newton-Raphson updates without actually having a dedicated software for this, but just being able to reuse some least squares software. OK. So you know, we've talked about this many times. I just want to make sure that we're all on the same page here.

We have a function f. We're going to assume that it has two derivatives. And it's a function from Rm to R. So it's fist derivative is called gradient. That's the vector that collects all the partial derivatives with respect to each of the coordinates.

It's dimension m, of course. And the second derivative is an m by m matrix. It's called the Hessian. And ith row and jth column, you see the second partial derivative with respect to the ith component and the jth component. OK. We've seen that several times. This is just multi-variable calculus.

But really the point here is to maybe the notation is slightly different, because I want to keep track of f. So when I write the gradient, I write nabla sub f. And when I write Hessian, I write nabla H sub f.

And as I said, if f is strictly concave, then Hf of x is negative definite. What it means is that if I take any x in Rm, then x transpose Hf, well, that's for any X0 X, this is actually strictly negative. That's what it means to be negative definite. OK?

So every time I do x transpose-- so this is like a quadratic form. And I want it to be negative for all values of X0 and X, both of them. That's very strong, clearly. But for us, actually, this is what happens just because of the properties of b.

Well, at least the fact that it's negative, less than or equal to, if I want it to be strictly less I need some properties on X. And then I will call the Hessian map the function that maps X to this matrix Hf of X. So that's just the second derivative at x. Yeah.

**AUDIENCE:** When you what are [INAUDIBLE]?

**PHILIPPE RIGOLLET:** Where do [INAUDIBLE]? Oh, yeah. I mean, you know, you need to be able to apply Schwarz lemma. Let's say two continue derivatives that's smooth.

**AUDIENCE:** [INAUDIBLE]

**PHILIPPE RIGOLLET:** No, that's fine. OK. So how does the Newton-Raphson method work? Well, what it does is that it forms a quadratic approximation to your function. And that's the one it optimizes at every single point.

OK. And the reason is because we have a closed-form solution to defining the minimum of a quadratic function. So if I give you a function that's of the form ax squared plus b x plus c, you know exactly a closed form for its minimum. But if I give you any function or, let's say-- yeah, yeah. So here, it's all about maximum.

I'm sorry. If you're confused with me using the word minimum, just assume that it was the word maximum. So this is how it works.

OK. If I give you a function which is concave, that's quadratic. OK. So it's going to look like this. So that's of the form ax squared-- where a is negative, of course-- plus bx plus c. Then you can solve your whatever.

You can take the derivative of this guy, set it equal to 0, and you will have an exact equation into what the value of x is that it realizes this maximum. If I give you any function that's concave, that's all clear, right? I mean, if I tell you the function that we have here is that the form ax minus b of x, then I'm just going to have something that inverts b prime.

But how do I do that exactly? It's not clear. And so what we do is we do a quadratic approximation, which should be true approximately everywhere, right? So I'm at this point here, I'm going to say, oh, I'm close to being that function.

And if I'm at this point here, I'm going to be close to being that function. And for this function, I can actually optimize. And so if I'm not moving too far from one to the other, I should actually get something.

So here's how the quadratic approximation works. I'm going to write the second order Taylor expansion. OK. And so that's just going to be my quadratic approximation.

It's going to say, oh, f of x, when x is close to some point x0, is going to close to f of x0 plus the gradient of f at x0 transpose x minus x0. And then I'm going to have plus 1/2x minus x0 transpose Hf at 0x x minus x transpose, right-- x minus x0. So that's just my second order Taylor expansion multi-variate 1. And let's say x0 is this guy.

Now, what I'm going to do is say, OK, if I wanted to set this derivative of this guy equal to 0, I would just have to solve, well, you know, f prime of x equals 0, meaning that X has to be f prime inverse of 0. And really apart from like being some notation manipulation, this is really not helping me. OK. Because I don't know what f prime inverse of 0 is in many instances.

However if f has a very specific form which is something that depends on x in a very specific way, there's just a linear term and then a quadratic term, then I can actually do something. So let's forget about this approach. And rather than minimizing f, let's just minimize the right-hand side.

OK. So sorry- maximize. So maximize the right-hand side. And so how do I get this? Well, I just set the gradient equal to 0. So what is the gradient?

The first term does not depend on x. So that means that this is going to be 0 plus-- what is the gradient of this thing, of the gradient of f at x0 transpose x minus x0? What is the gradient of this guy? So I have a function of the form b transpose x. What is the gradient of this thing?

**AUDIENCE:** [INAUDIBLE]

**PHILIPPE RIGOLLET:** I'm sorry?

**AUDIENCE:** [INAUDIBLE]

**PHILIPPE RIGOLLET:** I'm writing everything in two-column form, right? So it's just b. OK. So here, what is b? Well, it's gradient of f at x0. OK. And this term here gradient of f at x0 transpose x0 is just a constant. This thing is going away as well.

And then I'm looking at the derivative of this guy here. And this is like a quadratic term. It's like H times x minus x0 squared. So when I'm going to take the derivative, I'm going to have a factor 2 that's going to pop out and cancel this one half. And then I'm going to be left only with this part times this part.

OK. So that's plus Hf x minus x0. OK. So that's just a gradient. And I want it to be equal to 0. So I'm just going to solve this equal to 0. OK?

So that means that if I want to find the minimum, this is just going to be the x* that satisfies this. So that's actually equivalent to Hf times x* is equal to Hf x0 minus gradient f at x0. Now, this is a much easier thing to solve. What is this?

This is just a system of linear equations, right? I just need to find the x* such that when I pre-multiply it by a matrix I get this vector on the right-hand side. This is just something of the form ax equals b.

And I have many ways I can do this. I could do Gaussian elimination, or I could use Spielman's fast Laplacian solvers if I had some particular properties of H. I mean, there's huge activity in terms of how to solve those systems.

But let's say I have some time. It's not a huge problem. I can actually just use linear algebra. And linear algebra just tells me that x* is equal to Hf inverse times this guy, which those two guys are going to cancel. So this is actually equal to x0 minus Hf inverse gradient f at x0.

And that's just what's called a Newton iteration. I started at some x0. I'm at some x0, where I make my approximation. And it's telling me starting from this x0, I wanted to fully optimize a quadratic approximation, I would just have to take the x*. That's this guy.

And then I could just use this guy as my x0 and do it again, and again, and again, and again. And those are called Newton iterations. And they're basically the workhorse of interior point methods, for example, a lot of optimization algorithms.

And that's what you can see here. x* is equal to x0 minus the inverse Hessian times the gradient. We briefly mentioned gradient descent. We briefly mentioned gradient decent, at some point, to optimize the convex function, right? And if I wanted to use gradient descent, again, H is a matrix. But if I wanted to think of H as being a scalar, would it be a positive or negative number? Yeah.

**AUDIENCE:** [INAUDIBLE]

**PHILIPPE RIGOLLET:** Why?

**AUDIENCE:** [INAUDIBLE]

**PHILIPPE RIGOLLET:** Yeah. So that would be this. So I want to move against the gradient to do what?

**AUDIENCE:** [INAUDIBLE]

**PHILIPPE RIGOLLET:** To minimize. But I'm maximizing here, right? Everything is maximized, right? So I know that H is actually negative definite. So it's a negative number.

So you have the same confusions they do. We're maximizing a concave function here. So H is negative. So this is something of the form x0 plus something times the gradient.

And this is what your gradient ascent, rather than descent, would look like. And all it's saying, Newton is telling you don't take the gradient for granted as a direction in which you want to go. It says, do a slight change of coordinates before you do this according to what your Hessian looks like, all right?

And those are called second order methods that require knowing what the Hessian is. But those are actually much more powerful than the gradient descent, because they're using all of the local geometry of the problem. All of the local geometry of your function is completely encoded in this Hessian. And in particular it implies that it tells you where to switch and not to go slower in some places or go faster in other places.

Now, this in practice for, say, modern large scale machine learning problems, inverting this matrix H is extremely painful. It takes too much time. The matrix is too big, and computers cannot do it.

And people resort to what's called pseudo-Newton method, which essentially tries to emulate what this guy is. And there's many ways you can do this. Some of them is by using gradients that you've collected in the past. Some of them just say, well let's just pretend H is diagonal. There's a lot of things you can do to just play around this and not actually have to invert this matrix. OK?

So once you have this, you started from edge 0. It tells you which H* you can get as a maximizer of the local quadratic approximation to your function. You can actually just iterate that, all right?

So you start at some x0 somewhere. And then once you get to some xk, you just do the iteration which is described, which is just find a k plus 1, which is the maximizer of the local quadratic approximation to your function at xk and repeat until convergence. OK. So if this was an optimization class, we would prove that convergence actually, eventually, happens for a strictly concave function.

This is a stats class, so you're just going to have to trust me that this is the case. And it's globally convergent, meaning that you can start wherever you want, and it's going to work for under minor conditions on f. And in particular, those conditions are satisfied for the log-likelihood functions we have in mind.

OK. And it converges at an extremely fast rate. Usually it's quadratic convergence, which means that every time you make one step, you improve the accuracy of your solution by two digits. If that's something you're vaguely interested in, I highly recommend that you take a class on them in your optimization. It's a fascinating topic.

Unfortunately, we don't have much time, but it starts being more and more intertwined with high dimensional statistics and machine learning. I mean, it's an algorithms class, typically. But it's very much more principled.

It's not a bunch of algorithms that solve a bunch of problems. There's basically one basic idea, which is if I have a convex function, I can actually minimize it. If I have a concave function, I can maximize it. And it evolves around a similar thing.

So let's stare at this iterative step for a second and pause. And let me know if you have any questions. OK. So, of course, in a second we will plug in for the log-likelihood.

This is just a general thing for a general function f. But then in a second, f is going to be ln. OK. So if I wanted to implement that for real, I would have to compute the gradient of ln at a point xk. And I would have to compute the Hessian at a given point and invert it.

OK. So this is just the basic algorithm. And this, as you can tell, used in no place the fact that ln was the log-likelihood associated to some canonical exponential family in a generalized linear model. This never showed up.

So can we use that somehow? Optimization for longest time was about making your problems as general as possible accumulating maybe in the interior point method theory in Koenig programming in the mid-'90s. And now what optimization is doing is that it's [INAUDIBLE] very general. It' says, OK, if I want to start to go fast, I need to exploit as much structure about my problem as I can.

And the beauty is that as statisticians are a machine learning people, we do have a bunch of very specific problem that we want optimizers to solve. And they can make things run much faster. But this did not require to wait until the 21st century. Problems with very specific structure arose already in this generalized linear model.

So what do we know? Well, we know that this log-likelihood is really one thing that comes when we're trying to replace an expectation by an average, and then doing something fancy, right? That was our statistical hammer.

And remember when we introduced likelihood maximization we just said, what do we really want to do is to minimize the KL, right? That's the thing we wanted to minimize, the KL divergence between two distributions, the true one and the one that's parameterized by some unknown theta. And we're trying to minimize that over theta.

And we said, well, I don't know what this is, because it's an expectation with respect to some known distribution. So let me just replace the expectation with respect to my unknown distribution by an average over my data points. And that's how we justified the existence of the log-likelihood maximization problem.

But here, actually, I might be able to compute this expectation, at least partially where I need it. And what we're going to do is we're going to say, OK, since at a given point xk, say, let me call it here theta, I'm trying to find the inverse of the Hessian of my log-likelihood, right? So if you look at the previous one, as I said, we're going to have to compute the Hessian H sub l n of xk, and then invert it.

But let's forget about the inversion step for a second. We have to compute the Hessian. This is the Hessian of the function we're trying to minimize. But if I could actually replace it not by the function I'm trying to minimize to maximize or the log-likelihood, but really by the function I wish I was actually minimizing, which is the KL, right?

Then that would be really nice. And what happens is that since I'm actually trying to find this at a given xk, I can always pretend that this xk that I have in my current iteration is the true one and compute my expectation with respect to that guy. And what happens is that I know that when I compute the expectation of the Hessian of the log-likelihood at a given theta and when I take the expectation with respect to the same theta, what I get out is negative Fisher information.

The Fisher information was defined in two ways-- as the expectation of the square of the derivative or negative of the expectation of the second derivative of the log-likelihood. And so now, I'm doing some sort of a leap of faith here. Because there's no way the theta, which is the current xk, that's the current theta at which I'm actually doing this optimization-- I'm actually pretending that this is the right one.

But what's going to change by doing this is that it's going to make my life easier. Because when I take expectations, we'll see that when we look at the Hessian, the Hessian as essentially the derivative of, say, a product is going to be the sum of two terms, right? The derivative of u times v is u prime v plus uv prime.

One of those two terms is actually going to have expectation 0. And that's going to make my life very easy when I take expectations and basically just have one term that's going to go away. And so in particular, my formula, just by the virtue of taking this expectation before inverting the Hessian, is going to just shrink the size of my formulas by half.

OK. So let's see how this works. You don't have to believe me. Is there any question about this slide? You guys remember when we were doing maximum estimation and Fisher information and the KL divergence, et cetera? Yeah.

**AUDIENCE:** [INAUDIBLE]

**PHILIPPE RIGOLLET:** Because that's what we're really trying to minimize.

**AUDIENCE:** [INAUDIBLE]

**PHILIPPE RIGOLLET:** Yeah. So there's something you need to trust me with, which is that the expectation of H of ln is actually H of the expectation of ln, all right? Yeah, it's true, right? Because taking derivative is a linear operator.

And we actually used that several times when we said expectation of partial of l with respect to theta is equal to 0. Remember we did that? That's basically what we used, right?

**AUDIENCE:** ln is the likelihood.

**PHILIPPE RIGOLLET:** It's the log-likelihood.

**AUDIENCE:** Log-likelihood, [INAUDIBLE] OK. When we did Fisher [INAUDIBLE], we did the likelihood of [INAUDIBLE] observation.

**PHILIPPE RIGOLLET:** Yeah.

**AUDIENCE:** Why is it ln in this case?

**PHILIPPE RIGOLLET:** So actually, ln is typically not normalized. So I really should talk about ln over n. OK. But let's see that, OK? So if I have IID observations, that should be pretty obvious.

OK. So if I have IID x1, xn, with density f theta and if I look at log f theta of Xi, sum from i equal 1 to n, as I said, I need to actually have a 1 over n here. When I look at the expectation, they all have the same expectation, right? So this is actually, indeed, equal to negative KL plus a constant. OK?

And negative KL is because this-- sorry, if I look at the expectation. So the expectation of this guy is just the expectation of one of them, all right? So I just do expectation theta. OK? Agree?

Remember, the KL was expectation theta log f theta divided by f. So that's between p theta and p theta prime. Well, no, sorry. That's the true p.

And let's call it f. p theta, right? So that's what showed up, which is, indeed, equal to minus expectation theta log f theta plus log of f, which is just a constant with respect to theta. It's just the thing that's up doesn't matter. OK?

So this is what shows up here. And just the fact that I have this 1 over n doesn't change, because they're IID. Now, when I have things that are not IID-- because what I really had was Y1 Yn, and Yi at density f theta i, which is just the conditional density given Xi, then I could still write this.

And now when I look at the expectation of this guy, what I'm going to be left with is just 1 over n sum from i equal 1 to n of the expectation of log f theta i of Yi. And it's basically the same thing, except that I have a 1 over n expectation in front. And I didn't tell you this, because I only showed you what the KL divergence was for between two distributions.

But here, I'm telling you what the KL is between two products of distributions that are independent, but not necessarily identically distributed. But that's what's going to show up, just because it's a product of things. So when you have the log, it's just going to be a sum. Other questions?

All right, so what do we do here? Well, as I said, now we know that the expectation of H is negative Fisher information. So rather than putting H inverse in my iterates for Newton-Raphson, I'm just going to put the inverse Fisher information.

And remember, it had a minus sign in front. So I'm just going to pick up a plus sign now, just because i is negative, the expectation of the Hessian. And this guy has, essentially, the same convergence properties.

And it just happens that it's easier to compute the i than H Ln. And that's it. That's really why you want to do this.

Now, you might say that, well, if I use more information, I should do better, right? But it's actually not necessarily true for several reasons. But let's say that one is probably the fact that I did not use more information.

Every step when I was computing this thing at xk, I actually pretended that at theta k the true distribution was the one distributed according to theta k. And that was not true. This is only true when theta k becomes close to the true theta.

And so in a way, what I gained I lost again by making this thing. It's just really a matter of simple computation. So let's just see it on a particular example. Actually, in this example, it's not going to look much simpler. It's actually going to be the same.

All right, so I'm going to have the Bernoulli example. All right, so we know that Bernoulli belongs to the canonical exponential family. And essentially, all I need to tell you what b is. And b of theta for Bernoulli is log 1 plus e theta, right? We computed that.

OK. And so when I look at my log-likelihood, it is going to look like the sum from i equal 1 to n of Yi of-- OK, so here I'm going to actually use the canonical link. So it's going to be Xi transpose beta minus log 1 plus exponential Xi transpose beta. And phi for this guy is equal to 1.

Is it clear for everyone what I did? OK. So remember the density, so that was really just-- so the PMF was exponential Y theta minus log 1 plus e theta. There was actually no normalization. That's just the density of a Bernoulli.

And the theta is actually log p over 1 minus p. And so that's what actually gives me what my-- since p is the expectation, this is actually giving me also my canonical link, which is the log [? at link. ?] We saw that last time.

And so if I start taking the log of this guy and summing over n and replacing theta by Xi transpose beta, which is what the canonical link tells me to do, I get this guy. Is that clear for everyone? If it's not, please redo this step on your own.

OK. So I want to maximize this function. Sorry. So I want to maximize this function over there on the first line as a function of beta. And so to do this, I want to use either Newton-Raphson or what I call Fisher-scoring. So Fisher-scoring is the second one, when you replace the Hessian by negative Fisher information.

So I replace these two things. And so I first take the gradient. OK. So let's take the gradient of ln. So the gradient of ln is going to be, well, sum-- so here, this is of the form Yi, which is a scalar, times a vector, Xi beta. That's what I erased from here.

The gradient of b transpose x is just b. So here, I have just Yi Xi. So that's of the form Yi, which is a scalar, times Xi, which is a vector.

Now, what about this guy? Well, here I have a function. So I'm going to have just the usual rule, the chain rule, right? So that's just going to be 1 over this guy.

And then I need to find the Hessian of this thing. So the 1 is going away. And then I apply the chain rule again. So I get e of Xi transpose beta, and then the Hessian of this thing, which is Xi.

So my Hessian-- my radiant, sorry, I can actually factor out all my Xi's. And it's going to look like this. My gradient is a weighted average or weighted sum of the Xi's.

This will always happen when you have a generalized linear model. And that's pretty clear. Where did the Xi show up?

Whether it's from this guy or that guy, the Xi came from the fact that when I take the gradient of Xi transpose beta, I have this vector Xi that comes out. It's always going to be the thing that comes out. So I will always have something that looks like with some sum with some weights here of the Xi's.

Now, when I look at the second derivative-- so same thing, I'm just going to take the derivative this guy. Since nothing depends on beta here or here, I'm just going to have to take the derivative of this thing. And so it's going to be equal.

So if I look now at the Hessian ln as a function of beta, I'm going to have sum from i equal 1 to n of, well, Yi-- what is a derivative of Yi with respect to beta?

**AUDIENCE:** [INAUDIBLE]

**PHILIPPE RIGOLLET:** What?

**AUDIENCE:** [INAUDIBLE]

**PHILIPPE RIGOLLET:** Yeah. 0. OK? It doesn't depend on data. I mean, this distribution does. But Y itself is just a number, right?

So this is 0. So I'm going to get the minus. And then I'm going to have, again, the chain rule that shows up. So I need to find the derivative of x over 1 plus x. What is the derivative of x over 1 plus x. Actually don't even know. So that gives me-- OK.

So that's 1 over 1 plus x squared. So that's minus e Xi transpose beta-- sorry, 1 divided by 1 plus e Xi transpose beta squared times the derivative of the exponential, which is e Xi transpose beta and again, Xi. And then I have this Xi that shows up. But since I'm looking for a matrix, I'm going to have Xi, Xi transpose, right?

OK?

**AUDIENCE:** [INAUDIBLE]

**PHILIPPE RIGOLLET:** So I know I'm going to need something that looks like a matrix in the end. And so one way you want to think about it is this is going to spit out an Xi. There's already an Xi here.

So I'm going to have something that looks like Xi. And I'm going to have to multiply by it another vector Xi. And I want it to form a matrix. And so what you need to do is to take an outer product. And that's it.

So now as a result, the updating rule is this. Honestly, this is not a result of anything. I actually rewrote everything that I had before with a theta replaced by beta, because it's just painful to rewrite this entire thing, put some big parenthesis and put minus 1 here.

And then I would have to put the gradient, which is this thing here. So as you can imagine, this is not super nice. Actually, what's interesting is at some point I mentioned there's a pseudo-Newton method. They're actually doing exactly this.

They're saying, oh, at each iteration, I'm actually going to just take those guys. If I'm at iteration k, I'm actually just going to sum those guys up to k rather than going all the way to n and look at every one. So you're just looking at your observations one at a time based on where you were before.

OK. So you have a matrix. You need to invert it. So if you want to be able to invert it, you need to make sure that the sum with those weights of Xi outer, Xi, or Xi, Xi transpose is invertible. So that's a condition that you need to have.

And well, you don't have to, because technically you don't need to invert. You just need to solve the linear system. But that's actually guaranteed in most of the cases if n is large enough.

All right, so everybody sees what we're doing here? OK. So that's for the Newton-Raphson. If I wanted to actually do the Fisher-scoring, all I would need to do is to replace the Hessian here by its expectation when I pretend that the beta have, iteration k, is the true one.

What is the expectation of this thing? And when I say expectation here, I'm always talking about conditional expectation of Y given X. The only distributions that matter, that have mattered in this entire chapter, are conditional expectation of Y given X.

The conditional expectation of this thing given X is what? It's itself. It does not depend on Y. It only depends on the X's. So conditionally on x, this thing as far as we're concerned, is completely deterministic. So it's actually equal to it's expectation.

And so in this particular example, there's no difference between Fisher-scoring and Newton-Raphson. And the reason is because the gradient no longer depends on Yi-- I'm sorry. The Hessian no longer depends on Yi. OK?

This slide is just repeating some stuff that I've said. OK. So I think this is probably-- OK, let's go through this actually. At some point, I said that Newton-Raphson-- do you have a question?

**AUDIENCE:** Yeah. When would the gradient-- sorry, the Hessian ever depend on Yi? Because it seems like Yi is just-- or at least when you have a canonical link, that the log-likelihood is just [INAUDIBLE] to Yi Xi [INAUDIBLE] theta and that's the only place Y shows up. So [INAUDIBLE] derivative [INAUDIBLE] never depend on Y?

**PHILIPPE RIGOLLET:** Not when you have a canonical link.

**AUDIENCE:** So if it's not a [INAUDIBLE] there's is no difference between--

**PHILIPPE RIGOLLET:** No.

**AUDIENCE:** OK.

**PHILIPPE RIGOLLET:** Yeah. So yeah, maybe I wanted you to figure that out for yourself. OK. So Yi times Xi transpose beta. So essentially, when I have a general family, what he's referring to is that this is just b of Xi transpose beta.

So I'm going to take some derivatives. And there's going to be something complicated coming out of this. But I'm certainly not going to have some Yi showing up. The only place where Yi shows up is here.

Now, if I take two derivatives, this thing is gone, because it's linear. The first one is going to keep on like this guy. And the second one is going to make it go on. The only way this actually shows up is when to have an H here.

And if I have an H, then I can take second derivatives. And this thing is not going to be completely independent of beta. Sorry. Yeah, this thing is still going to depend on beta, which means that this Yi term is not going to disappear.

I believe we'll see an example of that, or maybe I removed it. I'm not sure, actually. I think we will see an example.

So let us do a Iteratively Re-weighted Least Squares, or IRLS, which I've actually recently learned is a term that even though it was defined in the '50s, people still feel free to use to define to call their new algorithms which have nothing to do with this. This is really something where you actually do iteratively re-weighted least squares.

OK. Let's just actually go through this quickly what is going to be iteratively re-weighted least squares. The way the steps that we had here showed up-- let's say those guys, x*-- is this, is when were actually solving this linear system, right? That was the linear system we were trying to solve.

But solving a linear system can be done by just trying to minimize, right? If I have x a and b, it's the same as minimizing the norm of ax minus b squared over x. If I can actually find an x for which it's 0, it means that I've actually solved my problem. And so that means that I can solve linear systems by solving least square problems.

And least square problems are things that statisticians are comfortable solving. And so all I have to do is to rephrase this as at least square problem. OK? And you know, I could just write it directly like this. But there's a way to streamline it a little bit. And that's actually by using weights.

OK. So I've come in the weights-- well, not today, actually, but very soon, all right? So this is just a reminder of what we had. We have that's Yi give Xi as a distribution distributed according to some distribution in the canonical exponential family.

So that means that the log-likelihood looks like this. Again, this does not matter to us. This is the form that matters. And we have a bunch of relationships that we actually spent some time computing.

The first one is that mu is b prime of theta i. The second one is that if I take g of mu i, I get this systematic component, Xi transpose beta that's modeling. Now, if I look at the derivative mu i with respect to theta i, this is the derivative of b prime of theta i with respect to theta i. So that's the second derivative.

And I'm going to call it Vi. If phi is equal to 1, this is actually the variance. And then I have this function H, which allows me to bypass altogether the existence of this parameter mu, which says if I want to go from Xi transpose beta all the way to theta i, I have to first do g inverse, and then b prime inverse. If I stopped here, I would just have mu. OK?

OK. So now what I'm going to do is I'm going to apply the chain rule. And I'm going to try to compute the derivative of my log-likelihood with respect to beta. So, again, the log-likelihood is much nicer when I read it as a function of theta than a function of beta, but it's basically what we've been doing by hand.

You can write it as a derivative with respect to theta first, and then multiply by the derivative of theta with respect to beta. OK. And we know that theta depends on beta as H of Xi transpose beta. OK? I mean, that's basically what we've been doing for the Bernoulli case.

I mean, we used the chain rule without actually saying it. But this is going to be convenient to actually make it explicitly show up. OK. So when I first take the derivative of my log-likelihood with respect to theta, I'm going to use the fact that my canonical family is super simple.

OK. So what I have is that my log-likelihood ln is the sum from i equal 1 to n of Yi theta i minus b of theta i divided by phi plus some constant, which will go away as soon as I'm going to take my first derivative. So if I take the derivative with respect to theta i of this guy, this is actually going to be equal to Yi minus b prime theta i divided by phi. And then I need to multiply by the derivative of theta i with respect to beta.

Remember, theta is H of Xi transpose beta. So the derivative of theta i with respect to beta j, this is equal to H prime of Xi transpose beta. And then I have the derivative of this guy. Actually, let me just do the gradient of theta I at beta, right?

That's what we did. I'm just thinking of theta i as being a function of theta. So what should I add here? It was just the vector Xi, which is just the chain rule again. That's Hi prime, right? You don't see it, but there's a prime here that's derivative.

OK. We've done that without saying it explicitly. So now if I multiply those two things I have this Yi minus b prime of the theta i, which I call by its good name, which is mu i. b prime of theta i is the expectation of Yi conditionally on Xi.

And then I multiply by this thing here. So here, this thing is written coordinate by coordinate. But I can write it as a big vector when I stack them together.

And so what I claim is that this thing here is of the form Y minus mu. But here I put some tildes. Because what I did is that first I multiplied everything by g prime of mu for each mu.

OK. So why not? OK. Actually, on this slide it will make no sense why I do this. I basically multiply by g prime on one side and divide by g prime on the other side. So what I write so far is that the gradient of ln with respect to beta is the sum from i equal 1 to n of Yi minus mu i, let's call it, divide by phi times H prime of Xi transpose beta Xi. OK. So I just stacked everything that's here.

And now I'm going to start calling things. The first thing I'm going to do is I'm going to divide. So this guy here I'm going to push here.

Now, this guy here I'm actually going to multiply by g prime of mu i. And this guy I'm going to divide by g prime of mu i. So there's really nothing that happened here.

I just took g prime and multiply and divide it by g prime. Why do I do this? Well, that's actually going to be clear when we talk about iteratively re-weighted least squares.

But now, essentially I have a new mu, a Y which is-- so this thing now is going to be Y tilde minus mu tilde, so i, i. Now, this guy here I'm going to call Wi. And I have the Xi that's there, which means that now the thing that I have here I can write as follows.

Gradient ln of beta is equal to what? Well, I'm going to write it in matrix forms. So I have the sum over i of something multiplied by Xi. So I'm going to write it as x transpose. Then I'm going to have this matrix W1 Wn, and then 0 elsewhere.

And then I'm going to have my Y tilde minus mu. And remember, X is the matrix with-- sorry, it should be a bit [INAUDIBLE]. I have n, and then p. And here I have my Xi j in this matrix on row i and column j.

And this is just a matrix that has the Wi's on the diagonal. And then I have Y tilde minus mu. So this is just the matrix we're writing of this formula.

All right. So it's just saying that if I look at the sum of weighted things of my columns of Xi, it's basically the same thing. When I'm going to multiply this by my matrix, I'm going to get exactly those terms, right? Yi minus mu i tilde times Wi.

And then when I actually take this Xi transpose times this guy, I'm really just getting the sum of the columns with the weights, right? Agree? If I look at this thing here, this is a vector that has S coordinates, Wi times Yi tilde minus mu i tilde.

And I have n of them. So when I multiply X transpose by this guy, I'm just looking at a weighted sum of the columns of X transpose, which is a weighted sum of the rows of X, which are exactly my Xi's. All right, and that's this weighted sum of the Xi's.

OK. So here, as I said, the fact that we decided to put this g prime of mu i here and g prime of mu i here, we could have not done this, right? We could have just said, I forget about the tilde and just call it Yi minus mu i. And here, I just put everything I don't know into some Wi. And so why do I do this? Well, it's because when I actually start looking at the Hessian, what's going to happen?

**AUDIENCE:** [INAUDIBLE].

**PHILIPPE RIGOLLET:** Yeah. We'll do that next time. But let's just look quickly at the outcome of the computation of my Hessian. So I compute a bunch of second derivatives. And here, I have two terms, right? Well, he's gone.

So I have two terms. And when I take the expectation now, it's going to actually change, right? This thing is actually going to depend on Yi. Because I have an H which is not the identity. Oh, no, you're here, sorry.

So when I start looking at the expectation, so I look at the conditional expectation given Xi. The first term here has a Yi minus expectation. So when I take the conditional expectation, this is going to be 0. The first term is going away when I take the conditional expectation.

But this was actually gone already if we had the canonical term, because the second derivative of H when H is the identity is 0. But if H is not the identity, H prime prime may not be 0. And so I need that part to remove that term.

And so now, you know, I work a little bit, and I get this term. That's not very surprising. In the second derivative, I see I have terms in b prime prime. I have term in H prime, but squared. And then I have my Xi outer Xi, Xi, Xi transpose, which we know we would see.

OK. So we'll go through those things next time. But what I want to show you is that now once I compute this, I can actually show that if I look at this product that showed up, I had b prime prime times H prime squared. One of those terms is actually 1 over g prime.

And so I can rewrite it as one of the H primes, because I had a square, divided by g prime. And now, I have this Xi Xi transpose. So if I did not put the g prime in the W that I put here completely artificially, I would not be able to call this guy Wi, which is exactly what it is from this board.

And now that this guy is Wi, I can actually write this thing here as X transpose WX. OK? And that's why I really wanted my W to have this g prime of mu i in the denominator. Because now I can actually write a term that depends on W.

Now, you might say, how do I reconcile those two things? What the hell are you doing? And what the hell I'm doing is essentially that I'm saying that if you write beta k according to the Fisher-scoring iterations, you can actually write it as just this term here, which is of the form X transpose X inverse X transpose Y.

But I actually squeezed in these W's. And that's actually a weighted least square. And it's applied to this particular guy. So we'll talk about those weighted least squares.

But remember, least squares is of the form-- beta hat is X transpose X inverse X transpose Y. And here it's basically the same thing, except that I squeeze in some W after my X transpose.

OK. So that's how we're going to solve it. I don't want to go into the details now, mostly because we're running out of time. Are there any questions?