

The following content is provided under a Creative Commons license. Your support will help MIT OpenCourseWare continue to offer high-quality educational resources for free. To make a donation or to view additional materials from hundreds of MIT courses, visit MIT OpenCourseWare at [ocw.mit.edu](https://ocw.mit.edu).

**PHILIPPE**

So I apologize. My voice is not 100%. So if you don't understand what I'm saying, please ask me. So we're going to be analyzing-- actually, not really analyzing. We described a second-order method to optimize the log likelihood in a generalized linear model, when the parameter of interest was beta. So here, I'm going to rewrite the whole thing as a beta. So that's the equation you see.

**RIGOLLET:**

But we really have this beta. And at iteration  $k + 1$ , beta is given by beta  $k$ . And then I have a plus sign. And the plus, if you think of the Fisher information at beta  $k$  as being some number-- if you were to say whether it's a positive or a negative number, it's actually going to be a positive number, because it's a positive semi-definite matrix. So since we're doing gradient ascent, we have a plus sign here. And then the direction is basically gradient  $\ln$  at beta  $k$ . OK?

So this is the iterations that we're trying to implement. And we could just do this. At each iteration, we compute the Fisher information, and then we do it again and again. All right. That's called the Fisher-scoring algorithm. And I told you that this was going to converge. And what we're going to try to do in this lecture is to show how we can re-implement this, using iteratively re-weighted least squares, so that each step of this algorithm consists simply of solving a weighted least square problem.

All right. So let's go back quickly and remind ourselves that we are in the Gaussian-- sorry, we're in the exponential family. So if I look at the log likelihood for one observation, so here it's  $\ln$ -- sorry. This is the sum from  $i = 1$  to  $n$  of  $y_i$  minus-- OK, so it's  $y_i$  times  $\theta_i$ , sorry, minus  $b$  of  $\theta_i$ . Then there's going to be some parameter. And then I have plus  $c$  of  $y_i$   $\phi$ . OK. So just the exponential went away when I took the log of the likelihood. And I have  $n$  observations, so I'm summing over all  $n$  observations.

All right. Then we had a bunch of formulas that we came up to be. So if I look at the expectation of  $y_i$ -- so that's really the conditional of  $y_i$ , given  $x_i$ . But like here, it really doesn't matter. It's just going to be different for each  $i$ . This is denoted by  $\mu_i$ . And we showed that this was beta prime of  $\theta_i$ . Then the other equation that we found was that.

And so what we want to model is this thing. We want it to be equal to  $x_i$  transpose beta-- sorry

g of this thing. All right. So that's our model. And then we had that the variance was also given by the second derivative. I'm not going to go into it.

What's actually interesting is to see, if we want to express  $\theta_i$  as a function of  $x_i$ , what we get, going from  $x_i$  to  $\mu_i$  by  $g$  inverse, and then to  $\theta_i$  by  $b$  inverse, we get that  $\theta_i$  is equal to  $h$  of  $x_i$  transpose  $\beta$ ,  $h$  of  $x_i$  transpose  $\beta$ , where  $h$  is the inverse-- so which order is --this? Is the inverse of  $g$ , and then the compose would be prime. OK?

So we remember that last time, those are all computations that we've made, but they're going to be useful in our derivation. And the first thing we did last time is to show that, if I look now at the derivative of the log likelihood with respect to one coordinate of  $\beta$ , which is going to give me the gradient if I do that for all the coordinates, what we ended up finding is that we can rewrite it in this form, some of  $y_i$  tilde minus  $\mu_i$  tilde. So let's remind ourselves that-- so  $y_i$  tilde is just  $y$  divided-- well, OK  $y_i$  tilde is  $y_i$ -- is it times or divided-- times  $g$  prime of  $\mu_i$ .  $\mu_i$  tilde is  $\mu_i$  times  $g$  prime of  $\mu_i$ .

And then that was just an artificial thing, so that we could actually divide the weights by  $g$  prime. But the real thing that built the weights are this  $h$  prime. And there's this normalization factor. And so if we read it like that-- so if I also write that  $w_i$  is  $h$  prime of  $x_i$  transpose  $\beta$  divided by  $g$  prime of  $\mu_i$  times  $\phi$ , then I could actually rewrite my gradient, which is a vector, in the following matrix form, the gradient  $\ln$  at  $\beta$ . So the gradient of my log likelihood of  $\beta$  took the following form. It was  $x$  transpose  $w$ , and then  $y$  tilde minus  $\mu$  tilde.

And here,  $w$  was just the matrix with  $w_1, w_2$ , all the way to  $w_n$  on the diagonal and 0 on of the up diagonals. OK? So that was just taking the derivative and doing a slight manipulations that said, well, let's just divide whatever is here by  $g$  prime and multiply whatever is here by  $g$  prime. So today, we'll see why we make this division and multiplication by  $g$  prime, which seems to make no sense, but it actually comes from the Hessian computations.

So the Hessian computations are going to be a little more annoying. Actually, let me start directly with the coordinate  $y$ 's derivative, right? So to build this gradient, what we used, in the end, was that the partial derivative of  $\ln$  with respect to the  $g$ th coordinate of  $\beta$  was equal to the sum over  $i$  of  $y_i$  tilde minus  $\mu_i$  tilde times  $w_i$  times the  $g$ th coordinate of  $x_i$ . OK?

So now, let's just take another derivative, and that's going to give us the entries of the Hessian. OK, so we're going to the second derivative. So what I want to compute is the derivative with respect to  $\beta_j$  and  $\beta_k$ . OK.

So where does beta j-- so here, I already took the derivative with respect to beta j. So this is just the derivative with respect to beta k of the derivative with respect to beta j. So what I need to do is to take the derivative of this guy with respect to beta k. Where does beta k show up here? It's set in, in two places.

**AUDIENCE:** In the y's?

**PHILIPPE** No, it's not in the y's. The y's are my data, right? But I mean, it's in the y tildes. Yeah, because  
**RIGOLLET:** it's in mu, right? Mu depends on beta. Mu is g inverse of xi transpose beta. And it's also in the w's.

Actually, everything that you see is directly-- well, OK, w depends on mu n on beta explicitly. But the rest depends only on mu. And so we might want to be a little-- well, we can actually use the-- did I use the chain rule already? Yeah, it's here. But OK, well, let's go for it. Oh yeah, OK.

Sorry, I should not write it like that, because that was actually-- right, so I make my life miserable by just multiplying and dividing by this g prime of mu. I should not do this, right? So what I should just write is say that this guy here-- I'm actually going to remove the g prime of mu, because I just make something that depends on theta appear when it really should not.

So let's just look at the last but one equality. OK. So that's the one over there, and then I have xi j. OK, so here, it make my life much more simple, because yi does not depend on beta, but this guy depends on beta, and this guy depends on beta.

All right. So when I take the derivative, I'm going to have to be a little more careful now. But I just have a derivative of a product, nothing more complicated. So this is what? Well, the sum is going to be linear, so it's going to come out. Then I'm going to have to take the derivative of this term.

So it's just going to be 1 over psi. Then the derivative of mu i with respect to beta k, which I will just write like this, times h prime of xi transpose beta xi j. And then I'm going to have the other one, which is yi minus mu i over 5 times the second derivative of h of xi transpose beta.

And then I'm going to take the derivative of this guy with respect to beta j with beta k, which is just xi k. So I have xi j times xi k. OK. So I still need to compute this guy. So what is the partial derivative with respect to beta k of g? So mu is g of-- worry, it's g inverse of xi transpose beta.

OK?

So what do I get? Well, I'm going to get definitely the second derivative of  $g$ . Well, OK, that's actually not a bad idea. Well, no, that's OK. I can make the second-- what makes my life easier, actually? Give me one second. Well, there's no one that actually makes my life so much easier. Let's just write it. Let's go with this guy.

So it's going to be  $g''(\xi^T \beta) \xi^k$ . OK? So now, what do I have if I collect my terms? I have that this whole thing here, the second derivative is, well, I have the sum from  $1$  equal  $1$  to  $n$ . Then I have terms that I can factor out, right? Both of these guys have  $\xi^j$ , and this guy pulls out an  $\xi^k$ . And it's also here,  $\xi^j \xi^k$ , right? So everybody here is  $\xi^j \xi^k$ .

And now, I just have to take the terms that I have here. The  $1$  over  $\phi$ , I can actually pull out in front. And I'm left with the second derivative of  $g$  times the first derivative of  $h$ , both taken at  $\xi^T \beta$ . And then, I have this  $y_i - \mu_i$  times the second derivative of  $h$ , taken at  $\xi^T \beta$ . OK.

But here, I'm looking at Fisher scoring. I'm not looking at Newton's method, which means that I can actually take the expectation of the second derivative. So when I start taking the expectation, what's going to happen-- so if I take the expectation of this whole thing here, well, this guy, it's not-- and when I say expectation, it's always conditionally on  $\xi$ . So let's write it--  $x_1 \dots x_n$ . So I take conditional. This is just deterministic.

But what is the conditional expectation of  $y_i - \mu_i$  times this guy, conditionally on  $\xi^T \beta = 0$ , right? Because this is just the conditional expectation of  $y_i$ , and everything else depends on  $\xi$  only, so I can push it out of the conditional expectation. So I'm left only with this term. OK. So now I need to-- sorry, and I have  $\xi^j \xi^k$ . OK

So now, I want to go to something that's slightly more convenient for me. So maybe we can skip that part here, because this is not going to be convenient for me anyway. So I just want to go back to something that looks eventually like this. OK, that's what I'm going to want. So I need to have my  $\xi$  show up with some weight somehow. And the weight should involve  $h'$  divided by  $g'$ .

Again, the reason why I want to see  $g'$  coming back is because I had  $g'$  coming in the original  $w$ . This is actually the same definition as the  $w$  that I used when I was computing

the gradient. Those are exactly these  $w$ 's, those guys. So I need to have  $g$  prime that shows up. And that's where I'm going to have to make a little bit of computation here. And it's coming from this kind of consideration. OK?

So this thing here-- well, actually, I'm missing the  $\phi$  over there, right? There should be a  $\phi$  here. OK. So we have exactly this thing, because this tells me that, if I look at the Hessian-- so this was entry-wise, and this is exactly the form of something of the form of the  $k$ . This is exactly the  $j$ th  $k$ th entry of  $x_i x_i^T$ . Right? We've used that before.

So if I want to write this in a vector form, this is just going to be the sum of something that depends on  $i$  times  $x_i x_i^T$ . So this is  $\frac{1}{\phi} \sum_{i=1}^n g'(\mu_i) x_i x_i^T$ . OK? And that's for the entire matrix. Here, that was just the  $j$   $k$ th entries of this matrix. And you can just check that, if I take this matrix, the  $j$   $k$ th entry is just the product of the  $j$ th coordinate and the  $k$ th coordinate of  $x_i$ . All right.

So now I need to do my rewriting. Can I write this? So I'm missing something here, right? Oh, I know where it's coming from.  $\mu$  is not  $g'$  of  $x$  beta.  $\mu$  is  $g$  inverse of  $x$  beta, right? So the derivative of  $x$  prime is not  $g'$  prime. It's like this guy-- no,  $\frac{1}{g'(\mu)}$ , right? Yeah. OK? The derivative of  $g$  inverse is  $\frac{1}{g'(\mu)}$ . I need you guys, OK? All right.

So now, I'm going to have to rewrite this. This guy is still going to go away. It doesn't matter, but now this thing is becoming  $h'$  over  $g'$  of  $g$  inverse of  $x_i x_i^T$  beta, which is the same here, which is the same here. OK? Everybody approves? All right. Well, now, it's actually much nicer.

What is  $g$  inverse of  $x_i x_i^T$  beta? Well, that was exactly the mistake that I just made, right? It's  $\mu_i$  itself. So this guy is really  $g'$  of  $\mu_i$ . Sorry, just the bottom, right? So now, I have something which looks like a sum from  $i=1$  to  $n$  of  $h'$  of  $x_i x_i^T$  beta, divided by  $g'$  of  $\mu_i$  times  $x_i x_i^T$ , which I can certainly write in matrix form as  $x^T w x$ , where  $w$  is exactly the same as before.

So it's  $w_1 \dots w_n$ . And  $w_i$  is  $h'$  of  $x_i x_i^T$  beta divided by  $g'$  of  $\mu_i$ . There's a prime here times  $\phi$ , which is the same that we had here. And it's supposed to be the same that we have here, except the  $\phi$  is in white. That's why it's not there. OK. All right? So it's actually simpler than what's on the slides, I guess. All right.

So now, if you pay attention, I actually never force this  $g'$  of  $\mu_i$  to be here. Actually, I even tried to make a mistake to not have it. And so this  $g'$  of  $\mu_i$  shows up completely naturally. If I had started with this, you would have never questioned why I actually didn't multiply by  $g'$  and divided by  $g'$  completely artificially here. It just shows up naturally in the weights. But it's just more natural for me to compute the first derivative first than the second derivative second, OK?

And so we just did it the other way around. But now, let's assume we forgot about everything. We have this. This is a natural way of writing it,  $x^T w$ . If I want something that involves some weights, I have to force them in by dividing by  $g'$  of  $\mu_i$  and therefore, multiplying  $y_i$  by  $w_i$ . OK?

So now, if we recap what we've actually found, we got that-- let me write it here. We also have that the expectation of  $H$  in of  $\beta^T x$  is  $w$ . So if I go back to my iterations over there, I should actually update  $\beta_{k+1}$  to be equal to  $\beta_k$  plus the inverse. So that's actually equal to negative  $i$  of  $\beta_k$ -- well, yeah. That's negative  $i$  of  $\beta$ , I guess.

Oh, and  $\beta$  here shows up in  $w$ , right?  $w$  depends on  $\beta$ . So that's going to be  $\beta_k$ . So let me call it  $w_k$ . So that's the diagonal of  $H'$   $x^T \beta_k$ , this time, divided by  $g'$  of  $\mu_i$   $\phi$ . OK? So this  $\beta_k$  induces a  $\mu$  by looking at  $g'$  inverse of  $x^T \beta_k$ . All right. So  $\mu_i$  is  $g'$  inverse of  $x^T \beta_k$ . So that's 2 to the-- sorry, that's an iteration.

And so now, if I actually write these things together, I get minus  $x^T w$  inverse. So that's  $w_k$ . And then I have my gradient here that I have to apply at  $k$ , which is  $x^T w_k$ . And then I have  $y_k$  minus  $\mu_k$ , where, again, the indices-- I mean the superscript  $k$  are pretty natural.  $y_k$  just means that-- so that's just  $y_i$ . So that's just  $y_i$  times  $g'$  of  $\mu_i$ . And  $\mu_k$  is, if I look at the  $i$  coordinate, it's just going to be  $\mu_i$  times  $g'$  of  $\mu_i$ . OK?

So I just add superscripts  $k$  to everything. So I know that those things get updated real time, right? Every time I make one iteration, I get a new value for  $\beta$ , I get a new value for  $\mu$ , and therefore, I get a new value for  $w$ . Yes?

**AUDIENCE:** [INAUDIBLE] the Fisher equation [INAUDIBLE]?

**PHILIPPE**  
**RIGOLLET:**

Yeah, that's a good point. So that's definitely a plus, because this is a positive, semi-definite matrix. So this is a plus. And well, that's probably where I erased it. OK. Let's see where I made my mistake. So there should be a minus here. There should be a minus here. There should be a minus even at the beginning, I believe. So that means that what is my-- oh, yeah, yeah.

So you see, when we go back to the first, so what I erased was basically this thing here,  $y_i$  minus  $\mu_i$ . And when I took the first derivative-- so it was the derivative with respect to  $H$  prime. So the derivative with respect to the second term-- I mean, the derivative of the second term was actually killed, because we took the expectation of this guy. But when we took the derivative of the first term, which is the only one that stayed, this guy went away. But there was a negative sign from this guy, because that's the thing we took the negative off.

So it's really, when I take my second derivative, I should carry out the minus signs everywhere. OK? So it's just I forget this minus throughout. You see the first term went away, on the first line there. The first term went away, because the conditional expectation of  $y_i$ , given  $x_i = 0$ . And then I had this minus sign in front of everyone, and I forgot it. All right. Any other mistake that I made? We're good? All right.

So now, this is what we have, that  $x_k$ -- sorry, that  $\beta_{k+1}$  is equal to  $\beta_k$  plus this thing. OK? And if you look at this thing, it sort of reminds us of something. Remember the least squares estimator. So here, I'm going to actually deviate slightly from the slides. And I will tell you how. The slides take  $\beta_k$  and put it in here, which is one way to go. And just think of this as a big least square solution.

Or you can keep the  $\beta_k$ , solve another least squares, and then add it to the  $\beta_k$  that you have. It's the same thing. So I will take the different routes. So you have the two options, all right? OK.

So when we did the least squares-- so parenthesis least squares-- we had  $y = X\beta + \epsilon$ . And our estimator  $\hat{\beta}$  was  $(X^T X)^{-1} X^T y$ , right? And that was just solving the first order condition, and that's what we found.

Now look at this--  $(X^T X)^{-1} X^T$  bleep something. OK? So this looks like, if this is the same as the left hand side, if  $W_k$  is equal to the identity matrix, meaning we don't see it, and  $y$  is equal to  $\tilde{y}_k - \mu_k$ -- so those similarities, the fact that we just squeeze in-- so the fact that the response variable is different is really not a problem.

We just have to pretend that this is equal to  $y$  tilde minus  $\mu$  tilde. I mean, that's just the least squares. When you call a software that does least squares for you, you just tell it what  $y$  is, you tell it with  $x$  is, and it makes the computation. So you would just lie to it and say all the actual  $y$  I want is this thing.

And then we need to somehow incorporate those weights. And so the question is, is that easy to do? And the answer is yes, because this is a setup where this would actually arise.

So one of the things that's very specific to what we did here and with least squares, we assume that  $\epsilon$ , when we did at least the inference, we assumed that  $\epsilon$  was normal 0 and the covariance matrix was the identity, right? What if the covariance matrix is not the identity? If the covariance matrix is not the identity, then your maximum likelihood is not exactly these least squares.

If the covariance matrix is any matrix you have another solution, which involves the inverse of the covariance matrix that you have, but if your covariance matrix, in particular, is diagonal-- which would mean that each observation that you get in this system of equations is still independent, but the variances can change from one line to another, from one observation to another-- then it's called heteroscedastic. "Hetero" means "not the same." "Scedastic" is "scale." And a heteroscedastic case, you would have something slightly different. And it makes sense that, if you know that some observations have much less variance than others, you might want to give them more weight. OK?

So if you think about your usual drawing, and maybe you have something like this, but the actual line is really-- OK, let's say you have this guy as well, so just a few here. If you start drawing this thing, if you do least squares, you're going to see something that looks like this on those points.

But now, if I tell you that, on this side, the variance is equal to 100, meaning that those points are actually really far from the true one, and here on this side, the variance is equal to 1, meaning that those points are actually close to the line you're looking for, then the line you should be fitting is probably this guy, meaning do not trust the guys that have a lot of variance.

And so you need somehow to incorporate that. If you know that those things have much more variance than these guys, you want to weight this. And the way you do it is by using weighted least squares. OK. So we're going to open in parentheses on weighted least squares. It's not a



fundamental statistical question, but it's useful for us, because this is exactly what's going to spit out-- something that looks like this with this matrix  $w$  in there.

OK. So let's go back in time for a second. Assume we're still covering least squares regression. So now, I'm going to assume that  $y$  is  $x$  beta plus epsilon, but this time, epsilon is a multivariate Gaussian in, say,  $p$  dimensions with mean 0. And covariance matrix, I will write it as  $w$  inverse, because  $w$  is going to be the one that's going to show up. OK?

So this is the so-called heteroscedastic. That's how it's spelled, and yet another name that you can pick for your soccer team or a capella group. All right. So the maximum likelihood, in this case-- so actually, let's compute the maximum likelihood for this problem, right? So the log likelihood is what? Well, we're going to have the term that tells us that it's going to be-- so OK.

What is the density of a multivariate Gaussian? So it's going to be a multivariate Gaussian in  $p$  dimension with mean  $x$  beta and covariance matrix  $w$  inverse, right? So that's the density that we want. Well, it's of the form  $1$  over determinant of  $w$  inverse times  $2\pi$  to the  $p/2$ . OK? And times exponential, and now, what I have is  $x$  minus  $x$  beta transpose  $w$ -- so that's the inverse of  $w$  inverse--  $x$  minus  $x$  beta divided by 2. OK?

So this is  $x$  minus  $\mu$  transpose  $\sigma$  inverse  $x$  minus  $\mu$  divided by 2. And if you want a sanity check, just assume that  $\sigma$ -- yeah?

**AUDIENCE:** Is it  $x$  minus  $x$  beta or  $y$ ?

**PHILIPPE**  
**RIGOLLET:** Well, you know, if you want this to be  $y$ , then this is  $y$ , right? Sure. Yeah, maybe it's less confusing. So if you should do  $p$  is equal to 1, then what does it mean? It means that you have this mean here. So let's forget about what it is. But this guy is going to be just 1 sigma squared, right? So what you see here is the inverse of sigma squared. So that's going to be 2 over 2 sigma squared, like we usually see it.

The determinant of  $w$  inverse is just the product of the entry of the 1 by 1 matrix, which is just sigma square. OK? So that should be actually-- yeah, no, that's actually-- yeah, that's sigma square. And then I have this  $2\pi$ . So square root of this, because  $p$  is equal to 1, I get sigma square root  $2\pi$ , which is the normalization that I get.

This is not going to matter, because, when I look at the log likelihood as a function of beta-- so I'm assuming that  $w$  is known-- what I get is something which is a constant. So it's minus  $p$  minus  $n$  times  $p/2$  times log that  $w$  inverse times  $2\pi$ . OK? So this is just going to be a

constant. It won't matter when I do the maximum likelihood.

And then I'm going to have what? I'm going to have plus  $1/2$  of  $y$  minus  $x$  beta transpose  $w$   $y$  minus  $x$  beta. So if I want to take the maximum of this guy-- sorry, there's a minus here. So if I want to take the maximum of this guy, I'm going to have to take the minimum of this thing. And the minimum of this thing, if you take the derivative, you get to see-- so that's what we have, right? We need to compute the minimum of  $y$  minus  $x$  beta transpose  $w$  minus  $y$  minus  $x$  beta.

And the solution that you get-- I mean, you can actually check this for yourself. The way you can see this is by doing the following. If you're lazy and you don't want to redo the entire thing-- maybe I should keep that guy.  $W$  is diagonal, right? I'm going to assume that so  $w$  inverse is diagonal, and I'm going to assume that no variance is equal to 0 and no variance is equal to infinity, so that both  $w$  inverse and  $w$  have only positive entries on the diagonal. All right?

So in particular, I can talk about the square root of  $w$ , which is just the matrix, the diagonal matrix, with the square roots on the diagonal. OK? And so I want to minimize in beta  $y$  minus  $x$  beta transpose  $w$   $y$  minus  $x$  beta.

So I'm going to write  $w$  as square root of  $w$  times square root of  $w$ , which I can, because  $w$ -- and it's just the simplest thing, right? If  $w$  is  $w_1$   $w_n$ , so that's my  $w$ , then the square root of  $w$  is just square root of  $w_1$  square root of  $w_n$ , and then 0 is elsewhere. OK? So the product of those two matrices gives me definitely back what I want, and that's the usual matrix product.

Now, what I'm going to do is I'm going to push one on one side and push the other one on the other side. So that gives me that this is really the minimum over beta of-- well, here I have this transposed, so I have to put it on the other side.  $w$  is clearly symmetric and so is square root of  $w$ . So the transpose doesn't matter. And so what I'm left with is square root of  $w$   $y$  minus square root of  $w$   $x$  beta transpose, and then times itself. So that's square root  $w$   $y$  minus square root  $w$ -- oh, I don't have enough space--  $x$  beta. OK, and that stops here.

But this is the same thing that we've been doing before. This is a new  $y$ . Let's call it  $y$  prime. This is a new  $x$ . Let's call it  $x$  prime. And now, this is just the least squares estimator associated to a response  $y$  prime and a design matrix  $x$  prime. So I know that the solution is  $x$  prime transpose  $x$  prime inverse  $x$  prime transpose  $y$  prime.

And now, I'm just going to substitute again what my  $x$  prime is in terms of  $x$  and what my  $y$  prime is in terms of  $y$ . And that gives me exactly  $x$  square root  $w$  square root  $w$   $x$  inverse. And

then I have  $x^T w$  for this guy. And then I have  $w^T y$  for that guy. And that's exactly what I wanted. I'm left with  $x^T w^{-1} x^T w y$ . OK?

So that's a simple way to take into account the  $w$  that we had before. And you could actually do it with any matrix that's positive semi-definite, because you can actually talk about the square root of those matrices. And it's just the square root of a matrix is just a matrix such that, when you multiply it by itself, it gives you the original matrix. OK?

So here, that was just a shortcut that consisted in saying, OK, maybe I don't want to recompute the gradient of this quantity, set it equal to 0, and see what  $\hat{\beta}$  had should be. Instead, I am going to assume that I already know that, if I did not have the  $w$ , I would know how to solve it. And that's exactly what I did. I said, well, I know that this is the minimum of something that looks like this, when I have the primes. And then I just substitute back my  $w$  in there. All right. So that's just the lazy computation. But again, if you don't like it, you can always take the gradient of this guy. Yes?

**AUDIENCE:** Why is the solution written in the slides different?

**PHILIPPE** Because there's a mistake. Yeah, there's a mistake on the slides. How did I make that one?

**RIGOLLET:** I'm actually trying to parse it back. I mean, it's clearly wrong, right? Oh, no, it's not. No, it is. So it's not clearly wrong.

Actually, it is clearly wrong. Because if I put the identity here, those are still associative, right? So this product is actually not compatible. So it's wrong, but there's just this extra thing that I probably copy-pasted from some place. Since this is one of my latest slide, I'll just color it in white. But yeah, sorry, there's a mis-- this parenthesis is not here. Thank you.

**AUDIENCE:** [INAUDIBLE].

**PHILIPPE** Yeah. OK?

**RIGOLLET:**

**AUDIENCE:** So why not square root [INAUDIBLE]?

**PHILIPPE** Because I have two of them. I have one that comes from the  $x^T$  prime that's here, this guy. And  
**RIGOLLET:** then I have one that comes from this guy here. OK, so the solution-- let's write it in some place that's actually legible-- which is the correction for this thing is  $x^T w^{-1} x^T w y$ . OK? So you just squeeze in this  $w$  in there. And that's exactly what we had

before,  $x^T w$  inverse  $x^T w$  some  $y$ . OK?

And what I claim is that this is routinely implemented. As you can imagine, heteroscedastic linear regression is something that's very common. So every time you a least squares formula, you also have a way to put in some weights. You don't have to put diagonal weights, but here, that's all we need.

So here on the slides, again, I took the beta  $k$ , and I put it in there, so that I have only one least square solution to formulate. But let's do it slightly differently. What I'm going to do here now is I'm going to say, OK, let's feed it to some least squares. So let's do weighted least squares on a response,  $y$  being  $y_k - \mu_k$ , and design matrix being, well, just the  $x$  itself. So that doesn't change.

And the weights-- so the weights are what? The weights are the  $w_k$  that I had here. So  $w_k$  is  $\frac{1}{\sigma_k^2}$ . OK, and so this, if I solve it, will spit out something that I will call a solution. I will call it  $u_{k+1}$ . And to get  $\beta_{k+1}$ , all I need to do is to do  $\beta_k + u_{k+1}$ -- sorry,  $\beta_k$ -- yeah. OK?

And that's because-- so here, that's not clear. But I started from there, remember? I started from this guy here. So I'm just solving a least squares, a weighted least square that's going to give me this thing. That's what I called  $u_{k+1}$ . And then I add it to  $\beta_k$ , and that gives me  $\beta_{k+1}$ . So I just have this intermediate step, which is removed in the slides. OK?

So then you can repeat until convergence. What does it mean to repeat until convergence?

**AUDIENCE:** [INAUDIBLE]?

**PHILIPPE RIGOLLET:** Yeah, exactly. So you just set some threshold and you say, I promise you that this will converge, right? So you know that at some point, you're going to be there. You're going to go there, but you're never going to be exactly there. And so you just say, OK, I want this accuracy on my data. Actually, the machine is a little strong. Especially if you have 10 observations to start with, you know you're going to have something that's going to have some statistical error. So that should actually guide you into what kind of error you want to be making.

So for example, a good rule of thumb is that if you have  $n$  observations, you just take some within-- if you want the L2 distance between the beta-- the two consecutive beta to be less than  $1/n$ , you should be good enough. It doesn't have to be that machine precision. And so it's clear how we do this, right?

So here, I just have to maintain a bunch of things, right? So remember, when I want to recompute-- at every step, I have to recompute a bunch of things. So I have to recompute the weights. But if I want to recompute the weights, not only do I need to previous iterate, but I need to know how the previous iterate impacts my means. So at each step, I have to recalculate  $\mu_i^k$  by doing  $g'$ , rate? Remember  $\mu_i^k$  was just  $g$  inverse of  $x_i^T \beta^k$ , right? So I have to recompute that.

And then I use this to compute my weights. I also use this to compute my  $y$ , right? so my  $y$  depends also on  $g'$  of  $\mu_i^k$ . I feed that to my weighted least squares engine. It spits out the  $\hat{u}^k$ , that I add to my previous  $\beta^k$ . And that gives me my new  $\beta^{k+1}$ . OK. So here's the pseudocode, if you want to take some time to parse it. All right.

So here again, the trick is not much. It's just saying, if you don't feel like implementing Fisher scoring or inverting your Hessian at every step, then a weighted least squares is actually going to do it for you automatically. All right. Then that's just a numerical trick. There's nothing really statistical about this, except the fact that this calls for a solution for each of the step reminded us of sum of the squares, except that there was some extra weights. OK.

So to conclude, we'll need to know, of course,  $xy$ , the link function. Why do we need the variance function? I'm not sure we actually need the variance function. No, I don't know why I say that. You need  $\phi$ , not the variance function.

So where do you start actually, right? So clearly, if you start very close to your solution, you're actually going to do much better. And one good way to start-- so for the beta itself, it's not clear what it's going to be. But you can actually get a good idea of what beta is by just having a good idea of what mu is. Because mu is  $g$  inverse of  $x_i^T \beta$ .

And so what you could do is to try to set mu to be the actual observations that you have, because that's the best guess that you have for their expected value. And then you just say, OK, once I have my mu, I know that my mu is a function of this thing. So I can write  $g$  of mu and solve it, using your least squares estimator, right? So  $g$  of mu is of the form  $x \beta$ . So you just solve for-- once you have your mu, you pass it through  $g$ , and then you solve for the beta that you want. And then that's the beta that you initialize with. OK?

And actually, this was your question from last time. As soon as I use the canonical link, Fisher scoring and Newton-Raphson are the same thing, because the Hessian is actually

deterministic in that case, just because when you use the canonical link,  $H$  is the identity, which means that its second derivative is equal to 0. So this term goes away even without taking the expectation.

So remember, the term that went away was of the form  $y_i - \mu_i$  divided by  $\phi$  times  $h'$  prime prime of  $x_i$  transpose beta, right? That's the term that we said, oh, the conditional expectation of this guy is 0. But if  $h'$  prime prime is already equal to 0, then there's nothing that changes. There's nothing that goes away. It was already equal to 0. And that always happens when you have the canonical link, because  $h$  is  $\eta$  prime inverse. And the canonical link is  $\eta$  prime inverse, so this thing is the identity. So the second derivative of  $f$  of  $x$  is equal to  $x$  is 0. OK.

My screen says end of show. So we can start with some questions.

**AUDIENCE:** I just wanted to clarify. So iterative-- what is it say for iterative--

**PHILIPPE** Reweighted least squares.

**RIGOLLET:**

**AUDIENCE:** Reweighted least squares is an implementation of the Fisher scoring [INAUDIBLE]?

**PHILIPPE** That's an implementation that's just making calls to weighted least squares oracles. It's called  
**RIGOLLET:** an oracle sometimes. An oracle is what you assume the machine can do easily for you. So if you assume that your machine is very good at multiplying by the inverse of a matrix, you might as well just do Fisher scoring yourself, right? It's just a way so that you don't have to actually do it.

And usually, those things are implemented-- and I just said routinely-- in statistical software. But they're implemented very efficiently in statistical software. So this is going to be one of the fastest ways you're going to have to solve, to do this step, especially for large-scale problems.

**AUDIENCE:** So the thing that computers can do well is the multiplier [INAUDIBLE]. What's the thing that the computers can do fast and what's the thing that [INAUDIBLE]?

**PHILIPPE** So if you were to do this in the simplest possible way, your iterations for, say, Fisher scoring is  
**RIGOLLET:** just multiply by the inverse of the Fisher information, right?

**AUDIENCE:** So finding that inverse is slow?

**PHILIPPE** Yeah, so it takes a bit of time. Whereas, since you know you're going to multiply directly by something, if you just say-- those things are not as optimized as solving least squares.

**RIGOLLET:** Actually, the way it's typically done is by doing some least squares. So you might as well just do least squares that you like.

And there's also less-- well, no, there's no-- well, there is less recalculation, right? Here, your Fisher, you would have to recompute the entire matrix of Fisher information. Whereas here, you don't have to. Right? You really just have to compute some vectors and the vector of weights, right?

So the Fisher information matrix has, say,  $n$  choose two entries that you need to compute, right? It's symmetric, so it's order  $n$  squared entries. But here, the only things you update, if you think about it, are this weight matrix. So there is only the diagonal elements that you need to update, and these vectors in there also. There's two inverses  $n$  squared. So that's much less thing to actually put in there. It does it for you somehow. Any other question? Yeah?

**AUDIENCE:** So if I have a data set [INAUDIBLE], then I can always try to model it with least squares, right?

**PHILIPPE** Yeah, you can.

**RIGOLLET:**

**AUDIENCE:** And so this is like setting my weight equal to 1-- the identity, essentially, right?

**PHILIPPE** Well, not exactly, because the  $g$  also shows up in this correction that you have here, right?

**RIGOLLET:**

**AUDIENCE:** Yeah.

**PHILIPPE** I mean, I don't know what you mean by--

**RIGOLLET:**

**AUDIENCE:** I'm just trying to say, are there ever situations where I'm trying to model a data set and I would want to pick my weights in a particular way?

**PHILIPPE** Yeah.

**RIGOLLET:**

**AUDIENCE:** OK.

**PHILIPPE** I mean--

**RIGOLLET:**

**AUDIENCE:** [INAUDIBLE] example [INAUDIBLE].

**PHILIPPE** Well, OK, there's the heteroscedastic case for sure. So if you're going to actually compute

**RIGOLLET:** those things-- and more generally, I don't think you should think of those as being weights. You should really think of those as being matrices that you invert. And don't think of it as being diagonal, but really think of them as being full matrices.

So if you have-- when we wrote weighted least squares here, this was really-- the  $w$ , I said, is diagonal. But all the computations really never really use the fact that it's diagonal. So what shows up here is just the inverse of your covariance matrix. And so if you have data that's correlated, this is where it's going to show up.