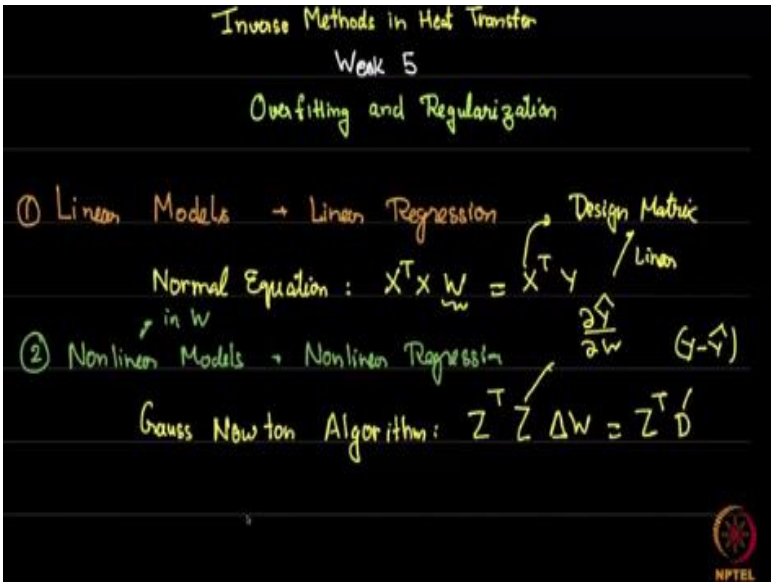
Inverse Methods in Heat Transfer
Prof. Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology, Madras

Lecture - 27
Overfitting and Regularization for Linear Models

(Refer Slide Time: 00:19)



Welcome to week five of inverse methods in heat transfer. In the last few weeks, we saw how to use linear models for forward problems and how to use linear regression in order to solve the inverse problems that occur from linear models. Secondly, we saw that using linear regression you could write the solution for the unknown parameters using the normal equations. So, the normal equations gave us,

$$X^T X W = X^T Y$$

where X is the design matrix.

Now the same idea was extended to the idea of non-linear models and for this we use non-linear regression, non-linear models mean non-linear in W, not nonlinear in X so non-linear in W. So, with this we use non-linear regression specifically we you look at the Gauss Newton algorithm and it had a form which was very similar to the normal equations except it was linear in delta W rather than being linear in W.

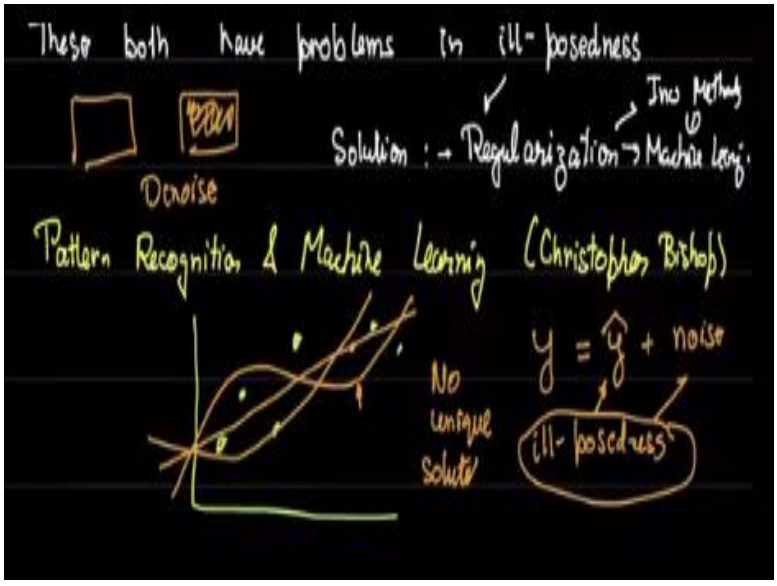
So, in this case we had the equation,

$$Z^T Z \Delta W = Z^T D$$

So, you can see that essentially if you take a sort of perturbation of the normal equation in linear you get the Gauss Newton algorithms. Now remember that Z here was $\frac{\partial \hat{Y}}{\partial W}$ in fact you can see that that would be identical to the design matrix in case $\hat{Y}$ is a linear model.

So, this is a simple check that you can do I think I talked about it also in the last week. D of course was $Y - \hat{Y}$. So, these are the two expressions that we have looked so far for inverse methods and also regression. Now there is a problem which we will see in this. So, both these models have problems in what is known as ill posedness and not get too much into ill posedness except for a short example that I am going to show right now which will give you this idea a little bit. We are going to show this example in the linear model.

(Refer Slide Time: 03:20)



Now this ill-posedness the actual solution in inverse methods and this is where inverse methods really take off is to use something called regularization. This regularization is very important both in inverse methods as well as machine learning which as I will argue later when we come to the machine learning type thing is basically, a subset. So, machine learning has actually belonged to inverse methods.

So, we are going to look at a preliminary idea of what this problem is, why this ill posedness starts, what regularization is in the context of simple linear equations. So, that is what we are going to start. So, let us go ahead and do that. So, some of the slides that I am going to show are from the book pattern recognition and machine learning by Christopher Bishop. It is an excellent book but it is a very advanced textbook as far as machine learning is concerned.

This book is available freely online, this is legally available freely online. so, this kindly been provided by I think Christopher Bishop who heads Microsoft research or some portion of Microsoft research he heads. So, he has provided the book entirely freely and the slides I am showing are actually due to a permission we have actually taken permission from him in order to provide some of these slides given a blanket permission.

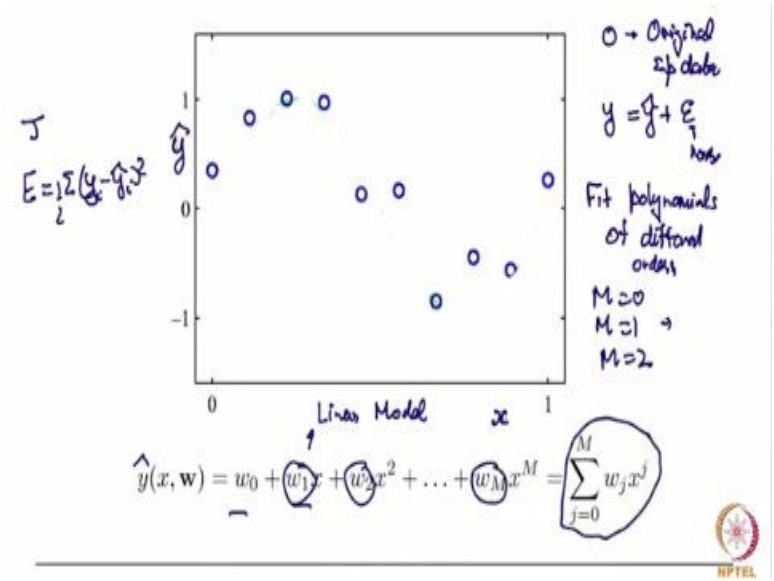
I also use this in another course of on online courses. So, this nicely illustrates this problem of both ill posedness as well as a regularization. Before we come there let me just briefly talk about the problem and then show the example that I have taken from Bishop's book. So, let us say we have a set of data points, so some points like this. Now if we don't know the physics of the problem and we do not have in mind a specific that we are going to do these points.

Now you want to find out that function which when added noise to produce this. So, for example Y actual is our model plus noise. Now the hard part and the ill-posedness comes in the fact that what do you ascribe to the function and what do you ascribe with the noise. For example, do we say as we saw in the linear case also. I showed it to you while showing let us say the overfitting not overfitting sorry the normal equations I showed you that you could fit a line you could fit a best quadratic, you could fit a best cubic.

Now how do you say which one is $\hat{Y}$ is it this which is $\hat{Y}$ and the rest of it which is noise or whether it is the quadratic which is $\hat{Y}$ and the rest of it which is noise. Some of it we can guess from physics but many times it is actually hard to sense. So, it is because of this that ill posedness starts. A similar example would be I give you an image and it is blurred due to some noise. Now if you want the original image you need to know something basic about the image.

You need to know maybe it is an image of a cat taken in an open sky or something of that sort. otherwise, it is hard for you to denoise the image uniquely. So, ill posedness happens because there is no unique solution to this problem of separating out noise from signal. There is a second problem which happens which I will kind of demonstrate to you during these which will talk about the stability issue which I briefly alluded to in the first week. So, let us see both these in the example that we are going to take a look at right now.

(Refer Slide Time: 07:32)



So, here it is. Here is a polynomial curve fitting so these blue dots here that you see here are the original experimental data. So, for example we can assume so this is here, let us call this $\hat{y}$ and this is x. So, you measure at x = 0 you got some $\hat{y}$ at x equal to some value got some $\hat{y}$. We have given a similar example which could occur within heat transfer or a semi within heat transfer within week five assignment. So, please take a look at that.

so, we have this curve which is sitting here I have written the original curve and I will show it to you later I sort of modified the slides which were given by Bishop. So, I have hidden the original curve and all you see is the data points. Now remember all these data points are being generated as your $\hat{y}$ or some model your y is $\hat{y}$ plus some noise. So, let us call this noise.

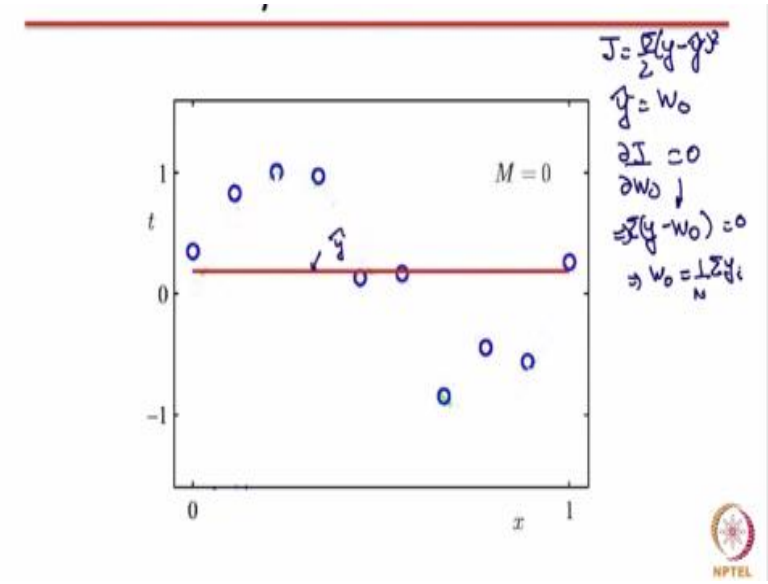
Now, what we are going to do in order to find out what the original curve is to fit polynomials of different orders. So, for example M = 0 means a straight-line polynomial, M = 1 means a linear

polynomial, M = 2 means a quadratic polynomial. We are going to try to fit different polynomials and see which one fits best. So, the hypothesis here with which we are working is the better it fits the original data, the lower the error J or let us call this E just to stop distinguish it from the cost function.

It is like the cost function with some mild difference. So, if we take the mean square error of our fit versus the original data that will tell us how good our fit is. So, here is what is expressed here same thing $\hat{y}$ of x, w is some $w_0 + w_1 x + w_2 x^2$. Remember this is still a linear model so I am showing this entire example within the linear model and we will see how to modify it for non-linear cases also.

Because this is still linear in the w's. So, in short, we can write it as this model. So, let us go step by step. We will start with the m equal to zero case.

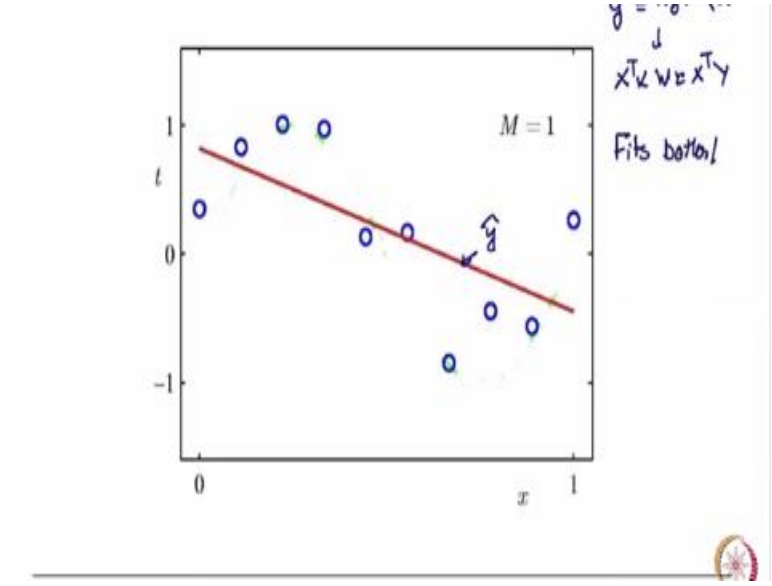
(Refer Slide Time: 10:19)



You can set in M = 0, you can how do you find this model simple find out J = half y - y-hat square sigma and let us say $\hat{y}$ is simply w_0 then you minimize J with respect to w_0. In fact, if you do this if you recall our equations, you will basically get y - y part or w_0 sigma if you actually calculate this will be 0 this will tell you that w_0 will be 1 over M sorry 1 over N in this case the number of points times sigma of y_i.

So, it will turn out that this line the constant line which you are saying is the model. This is our new model our model now is this is by hat that is we are saying. Regardless of what x is my prediction is going to be this and everything else is noise. So, the original signal was a straight line but noise random noise added to it. Now this does not look like a great model we can look at it intuitively and say maybe I can do slightly better.

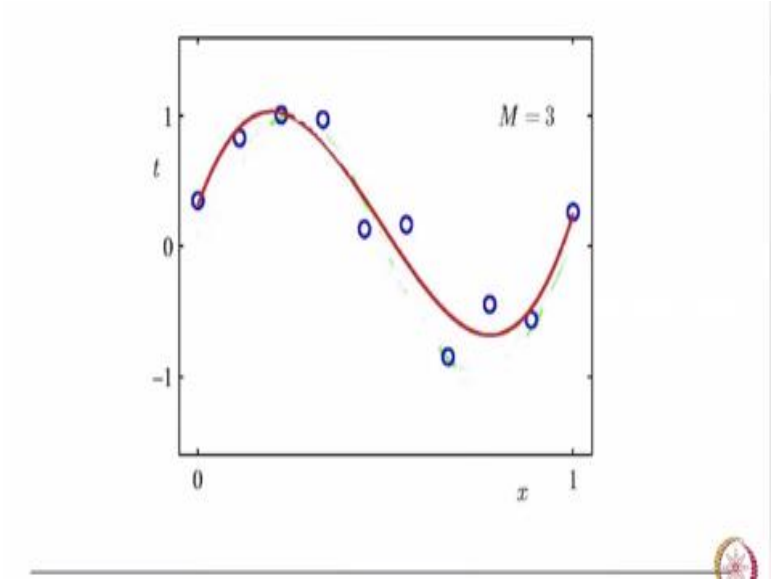
(Refer Slide Time: 11:34)



So, we go to the next case and we fit the first order polynomial. So, now remember our first order polynomial is simply a linear fit $\hat{y} = w_0 + w_1 x$ and then optimize this and in fact you can use these normal equations approach. In fact, I would encourage you try this during the exercise for this week. We have given some data set just for you to replicate this kind of effect. So, you can try this and you get some w_0 and w_1.

This looks better it looks like if you look at for example R square etcetera, it looks better lower error. Then it was before in the previous slide you can see errors are much higher you can see errors are much lower and this fits better. Overall, you can look at it and say these fits better. So, we can keep going let us say why stop at one I am not shown two but let us say we go to three.

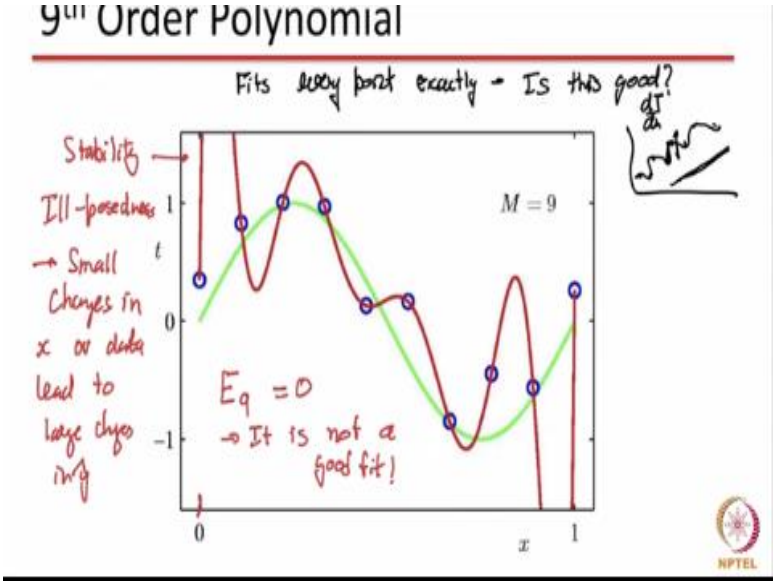
(Refer Slide Time: 12:34)



So, the green stuff that you see here is the hidden model or the hidden original data which I have kind of removed. You can see that some of it is still kind of appears here so ignore that for now. I will reveal this towards the end of this video. So, this is M = 3 this looks like it is even better it kind of captures the trend even better but you might be slightly suspicious but definitely lower error.

So, again you can see three points are almost exactly fit compared to the linear case which had more noise, this is less noise. So, now you go really greedy and say I asked add a really high order polynomial. So, I have 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 so I can always add a ninth-degree polynomial here.

(Refer Slide Time: 13:27)



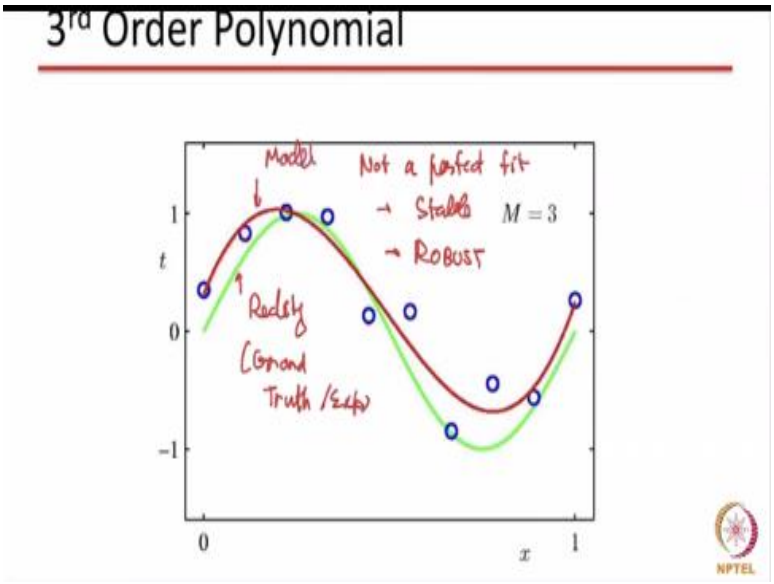
And here I put a ninth-degree polynomial. Now let us see what happens. If you look at it has actually fit every point exactly but the question is this good, should you really fit every point exactly? If I give you a slab temperature data which looks like this. You know the function is linear if I start predicting temperature like this and I find heat transfer using the gradient will not it be really bad.

So, in this case you might actually get a very high gradient when you actually want a slightly lower grade. So, see this, look at how the function varies. This is part of the stability problem that is, if I change the data a little bit so if I put slight noise to this data, you can imagine that the function will change drastically. So, small changes in x or data lead to large changes in y-hat. So, if I go from this position to a slightly different position see the change in y-hat, it is just huge.

This is typically a stability issue and especially when our thermocouples and our measurement instruments are noisy this is completely undesirable. So, now let me show you what the original function looks like. So, the original function looks like this. This is what was added noise too. You can see that let us say there was a sinusoidal temperature distribution and due to thermocouples or due to inherent noise in the system there is some variation.

So, there is some variation in the data but when you do this you can see that even though error for the ninth-degree polynomial is zero it is not a good fit. So, just driving the error to zero or driving the cost function to zero is not really as important as putting the model which is of the right order.

(Refer Slide Time: 16:00)



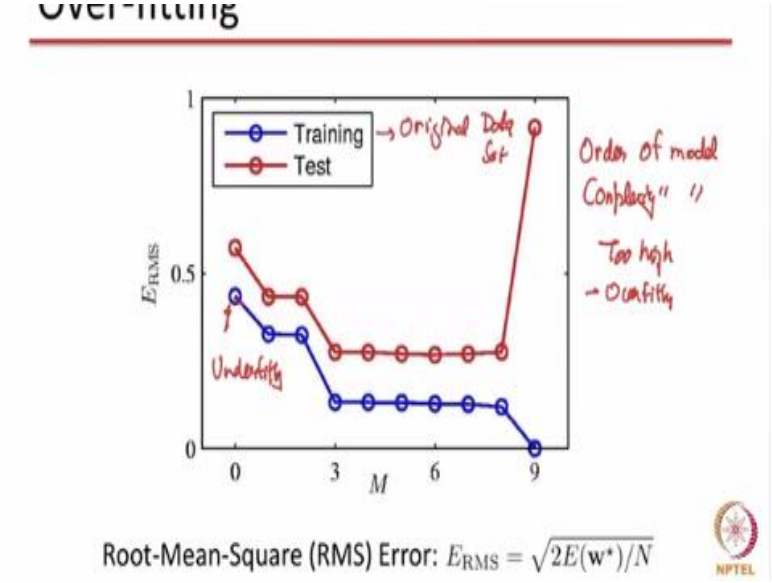
Now let us compare this with the third-degree polynomial as I promised I will now show what this looks like. Now if you compare what the third-degree polynomial looks like versus what the ninth-degree polynomial looks like. The third-degree polynomial is not a perfect fit but is stable. It is more robust. What do you mean by stable? Suppose I find dt by dx here. This is still going to be approximately the same as what my model predicts.

So, this is reality let me write that down. so, this is reality ground truth experiment whereas this is the model. So, you can see that model slopes are much closer to the original slope her. Come back to here look at this. So, if you see the slope here is much lower compared to this. So, it will seriously over predict transfer. When you see a model which E is 0 but the other quantities it actually predicts poorer. So, this is not a robust model, this is unstable.

Now why does this happen? This happens due to the inherent ill posedness of the problem, ill posedness that we do not know what the function varies like in the middle. For the same data set I can give many different function fits. So, that is the reason why ill posedness starts. Now within

this machine learning literature this problem has a particular name this is that error is zero or error is low but the fit is bad means it is called overfitting.

(Refer Slide Time: 18:00)



So, overfitting will behave in the following way, overfitting will behave like this. So, this is called training, training means original data sector. So, remember the original nine points that I had I will train I will find the error you can see that when the model is of a low order. So, on the X axis is the order of the model on the Y axis is the error that we found. So, when zeroth degree model has a high error, third degree model has a very low error and that keeps on going.

A ninth-degree model actually has completely zero error. But even though the error is zero there is something else which is high which is test. So, I will explain. The test means this you instead of having only nine points let us say you have 12 and three of them you held back in the sense just like a teacher. A teacher will show you some examples during class that is the training set and when I hold a final exam. I would have held out some few questions behind.

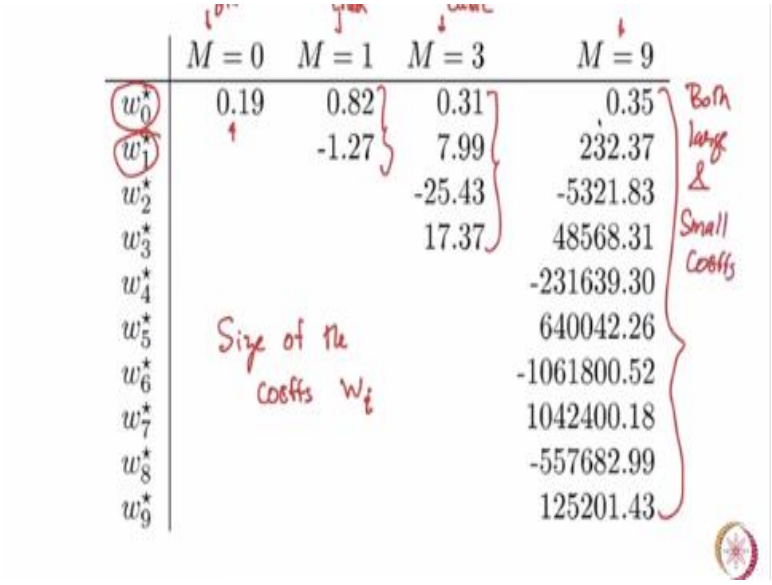
So, if somebody mugged up you know just memorized the training, they would do very well on the example problems but they will do very poorly in the test. So, how do we see the same example here maybe my original data set is at these points the original nine points that I showed you here but my test point could be here. So, I could have a test point here and I know that the original data is here but prediction is very far off.

So, this will give you a high error. So, you can see this. It was doing well in training and testing in third. For example, if I held back this data point you can see actually the wave the prediction I made and the original is actually fairly close as against once again something like this. This point here is the actual data and prediction is way down. So, that is why typically this is the graph that you will see in overfitting within machine learning.

If time permits, I will show this later but this is too even for the linear regression example that we did. So, the normal equation solution will give us these bad results for problems. In case you put too high order of model or this is called complexity of model in case of non-linear models too many parameters, too many knobs to turn was too high so this leads to over 50. This on the other hand is under fitting, I could have gone a little bit better. So, that is under fitting.

So, this is known as the classical bias variance problem and we will come to that again later on when we come to the machine learning portions of this course.

(Refer Slide Time: 21:00)

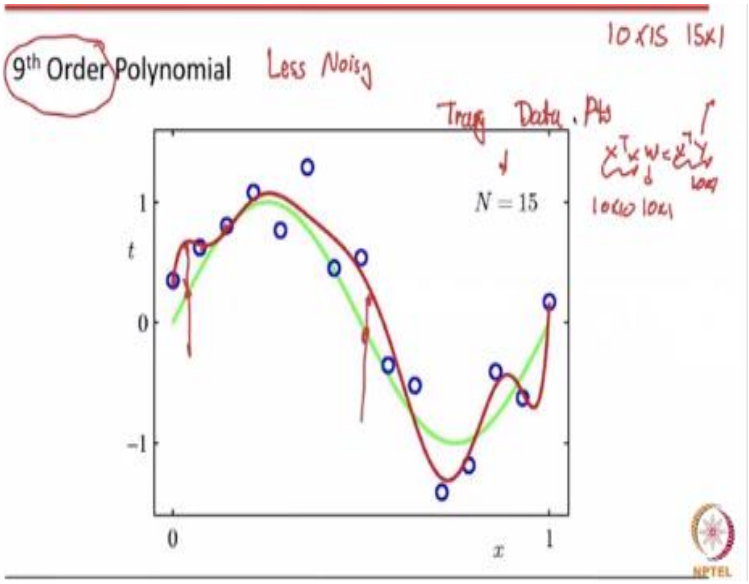


So, how do we cure this problem so for this we start looking at the coefficients of what the various degree polynomials are and this gives us a clue and we will use this idea to improve our non-linear Gauss Newton also. So, let us look at what the values of the coefficient are for the case of linear regression with various polynomial degrees. So, what you see here are the various coefficients.

So, for example w0 optimum, w1 optimum etcetera of course this is the zeroth order model. This is the first-degree linear model; this is the cubic model and this is our so-called best fit. Notice a certain phenomenon, look at the size of the coefficients or of the parameters wi. So, these two are well balanced, these slightly start varying these travels or who your huge order of magnitude. So, this was small, this is really large.

So, this has both large and small coefficients. So, this is just true observational. This always happens when there is overfitting. When there is overfitting, your w will travel through a large range of values. Now let us see if we can cure this. So, can we cure overfitting. So, I will show you two different cures the first cure is kind of a semi-practical the last cure is the one that we are coming to which is called regularization.

(Refer Slide Time: 22:52)



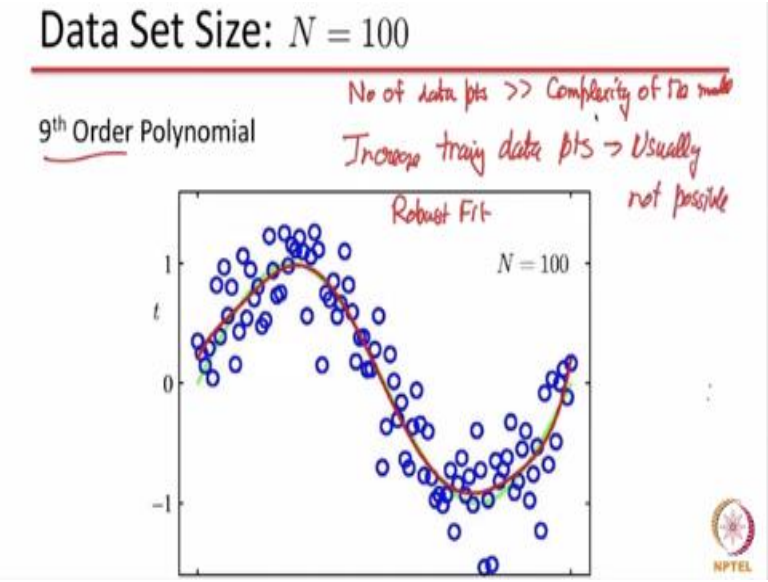
So, the first cure is this increase data size. Another way of saying this is suppose you do a few problems example problems and you do very poorly in exam obviously you are never going to know what the exam questions are but one thing you can do is solve more example problems. So, we are still going to stick with the same order polynomial same complexity as before but we are going to increase the number of data points, training data points increased.

So, we fit the ninth order polynomial for 15 data points. This is still easy because our original equation still works, our normal equation approach still works. This will be 10 cross 1, this will be 10 cross 10 and this will also be 10 cross 1, I request you to do these exercises. For example, what will be the size of y? y itself will be 15 cross 1 because there are 15 data points but x transpose will actually give you 10 cross 15. So, please check this out and do this problem.

It would take some fake examples and do this problem; it is an interesting exercise problem for you just to understand what the matrix sizes are like. So, we come here we see that the fit is slightly better. It is still overfitting it is still not as good as the original line but you can see that this is less noisy. So, the student who has done more example problems these are all example problems if I test somewhere at a new problem, it does not do so poorly as before.

So, again I introduce a new data point which was not there and check the gap between what is true and what is predicted is still not as time. So, less noisy lower test error.

(Refer Slide Time: 24:57)

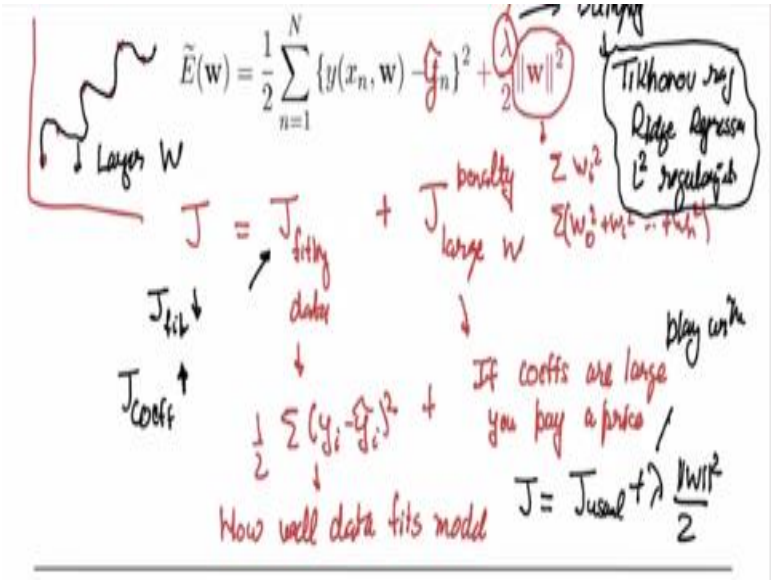


Now suppose I increase it further and put a lot of so this person has done lots and lots of exercise problems. Now you can see this is a robust fit, it is still the ninth order polynomial but you are fitting it through 100 points. So, that is like lots of data points obviously you cannot fit it perfectly, no ninth order polynomial will go through all these data points. So, this is just going to go through and this best fit is actually much better than the original ninth order polymer.

So, what we infer from this is number of data points should be large in comparison to the complexity of the model. So, we cannot just say ninth order polynomial is fine in case you have only 10 points. But it is not high in case you have 100 points in that case it actually fits really well. So, this is one way of solving overfitting that is increase training data points. But as you know we have only a few sensors in heat transfer or any such problem how are you going to increase data points.

I mean that is the number of data points you have and you want a model. So, most of the times this usually it is not possible to do this.

(Refer Slide Time: 26:30)



So, then you have the second solution called regularization. So, this is the standard solution to overfitting and the idea is very simple you penalize large coefficient values. What does that mean? There are different ways of looking at regularization, I am showing you one way which is standard and linear. I will show you a couple of more ways in the next video when we come to non-linear regression and how we actually do regularization there.

So, penalize large coefficient values means remember in our slide here we saw these w values some of these were very large whereas in the perfect model most of the w values are roughly small. So, you say I am going to make a new loss function this is called E here I am going to call it J. So,

J is the J due to fitting data so let us call this y-hat here plus some cost or penalty for large w values. So, J for fitting data looks like $\frac{1}{2}\ (y_i - \hat{y_i})^2$.

So, this only tells us how well data fits model. So, that we saw anyway. But this tells if coefficients are large you have to pay a price that is basically what the second part says. So, if coefficients are large, you pay a price and that price is given by a very simple term this I have told you earlier too this simply means sigma of wi square. For example, you can have ${w_0}^2 + {w_1}^2 + \cdots + {w_n}^2$.

So, as many coefficients there are you just add these costs and you multiply by a factor up front. This is called a regularization parameter. So, the regularization parameter lambda you can vary this and this is what is known as a hyper parameter much like our learning rate alpha. All you do is you keep on changing lambda and see what happens to your prediction as you vary lambda. Now there are specific tricks to do this and that is slightly beyond the limit of this course.

But I might touch upon this once again if I come to the machine learning portions. But the idea here is clear and it should become clearer as I show later on. Suppose you have a fit so you have these data points that you want to fit and you are choosing between multiple models. Now as the model tries to fit these data points more and more perfectly what will happen is this portion will go down. So, J fit will go down.

But what will happen invariably is if the number of data points is too few then J coefficient will go up. Why? Because it will require larger and larger W. So, the idea is somewhere in the middle you will automatically hit an optimum and this is kind of sort of like a damping term it damps out large oscillations. So, we will see this effect well not quite explicitly but we will see this effect also later on this week when I come to other the non-linear equivalence of this.

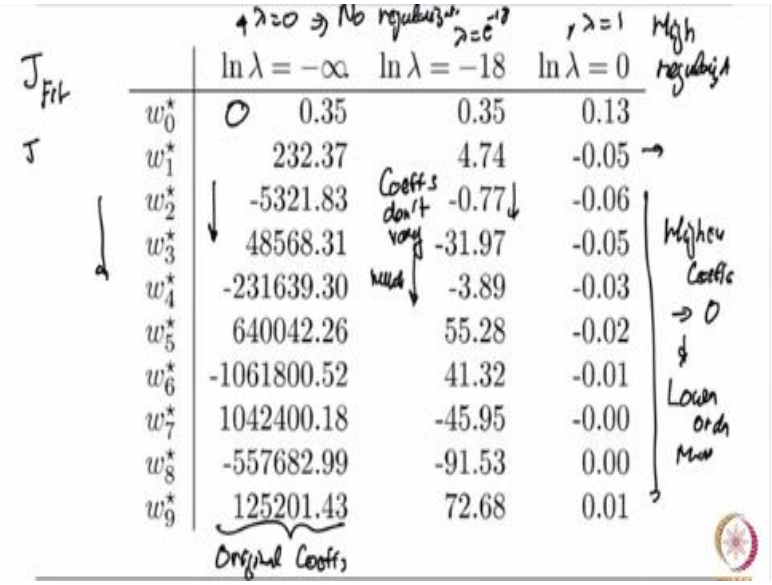
This kind of damping is called has several names, it is called Tikhonov, after the name of a scientist Tikhonov regularization. It is also called ridge regression; it has other names also it is also called L^2 regularization. So, I will come to L^1 regularization when we come to machine learning. So, all these three names are equivalent names as far as we are concerned. All it boils down to is to

changing our loss function to the usual loss function plus lambda times sigma of the weights of the coefficients divided by 2 and this you will have to play with.

Now there is a significant advantage of playing with lambda rather than playing with the order. In this case as you will see also coming to neural networks you can simply fix the non-linear model and you are just changing a small penalty term. As it turns out this have a huge number of different interpretations. We will see two to three different interpretations a couple in the next video and one later on when we come to probabilistic method.

So, this one simple term addition of one simple penalty term has several meanings within both inverse methods as well as machine learning. But let me show you the effect of doing.

(Refer Slide Time: 32:18)



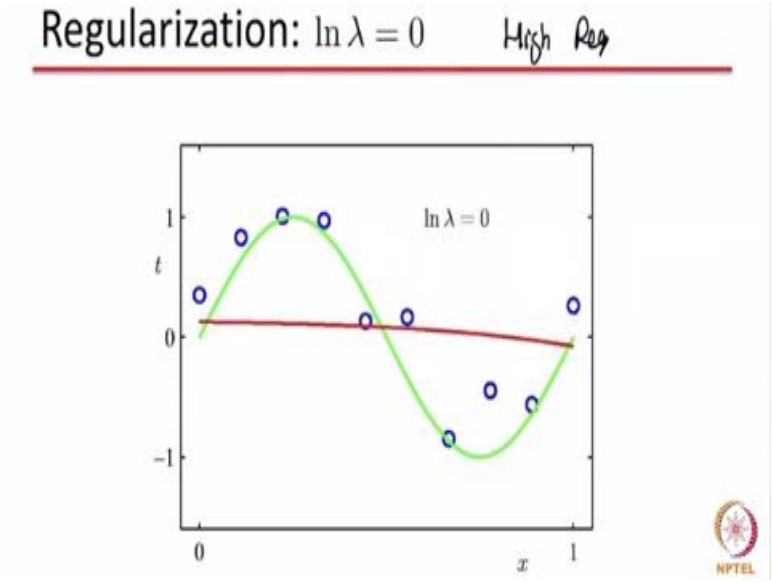
	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
w_0^*	0.35	0.35	0.13
w_1^*	232.37	4.74	-0.05
w_2^*	-5321.83	-0.77	-0.06
w_3^*	48568.31	-31.97	-0.05
w_4^*	-231639.30	-3.89	-0.03
w_5^*	640042.26	55.28	-0.02
w_6^*	-1061800.52	41.32	-0.01
w_7^*	1042400.18	-45.95	-0.00
w_8^*	-557682.99	-91.53	0.00
w_9^*	125201.43	72.68	0.01

So, let us say we set lambda = 0. So, if we set lambda = 0 this means no damping so or no regularization. So, you will get the same coefficients as before. These are the original coefficients, you saw there is a huge range of these values here. So, this is of course a bad fit it gives you low J of the fit and it gives you high coefficient. Now you come here log lambda is - 18 means lambda is e to the power - 18 when I do my experiments a couple of videos from now, I will go in powers of 10.

But since you are giving ln lambda here it makes sense to go in powers of e being our 2.7 a e power 1. So, now you see when you set log lambda = - 18 the coefficients are uniform do not vary much. So, it had be useful to see what it looks like, but currently all we are looking at is the intuition that we do not want coefficients to vary very much generally observationally, there is no direct logic that I can give you here, but observationally when the coefficients are all equally scaled.

It turns out that you are your function is not very noisy. Now here is log lambda = 0 this of course means lambda = 1. This is a high value, high regularization. Notice, higher coefficients is going to zero so which means we expected to perform about as well as a lower order model. This would be a higher degree model; this is our lower order model and this is somewhere intermediate. So, let us see what it looks like physically.

(Refer Slide Time: 34:24)

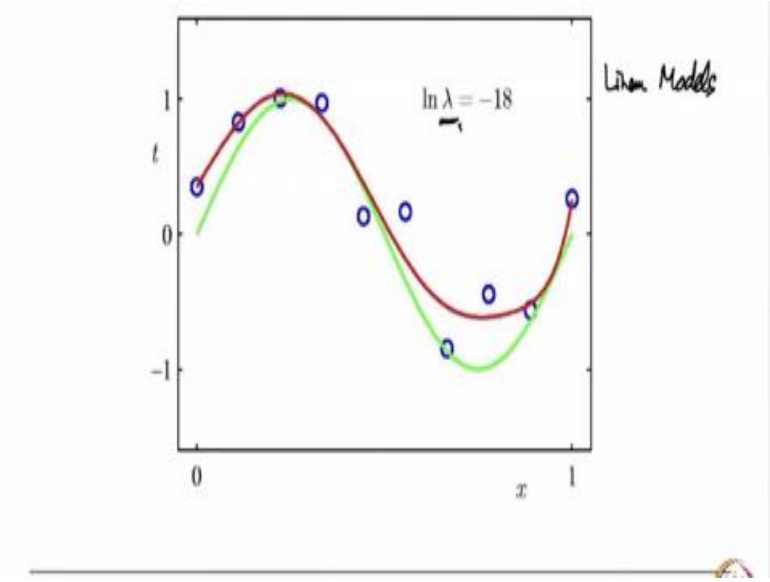


Here is the high regularization and look at what it looks like. It almost looks like the equal to zero model but we still used an M equal to nine model. Please notice we have always put we are not putting lower order models here. We are always using ninth order polynomial but the optimum w is found out by adding this extra cost. So, if you use the original full ninth order it will say no that will not work because you will have to pay too much penalty for these coefficients.

So, if the penalty is it is highly penalized then it just becomes like this function. You can compare this with the M = 0 model it will look somewhat similar. So, you can see the M = 0 model and you

can compare it with the original also. So, you will see this is approximately similar to the M = 0 model.

(Refer Slide Time: 35:40)



Now if I take that intermediate case, I already know what it looks like when I look at the lambda equal to zero case because that is just the bad fit. But if I look at the intermediate case that will look slightly better as you can see shortly. As you can see here when you use the intermediate model you see a very nice fit. So, this almost looks like the M equal to three fit that we had earlier. So, if you go back, you see the third order polynomial and the intermediate fit.

Even though here we had stopped our polynomial at third here on the other hand we let it be a ninth order polynomial but we just changed the regularization parameter. So, the moment you change the regularization parameter you are able to traverse you are able to travel the entire thing from a very bad model to a decent model to a very high model and all of it depends only on the simple one single parameter which is lambda.

So, if you vary in lambda, you can actually keep the model fixed and just vary the regularization that is the basic idea. So, this gives you intuition for why we will use this kind of damping within non-linear. So, what we looked at in this video was just this for linear models specifically polynomial models. The same trick tends to work very well in non-linear models also when there are a large number of parameters.

So, I am going to do the theory in the next video and finally in the last video I will show you a code. As I said at the beginning of this video, we are actually not going to show you a practical case where we apply it because you still have to do the calculations by hand. So, I am stopping with a small number of parameters like A and B or just two parameters but as you go to a large number of parameters this kind of damping this kind of regularization becomes inevitable and very useful as we will see when we come to the neural networks case.

So, thank you I will see you in the next video for non-linear regression how we modify it using similar regularization.