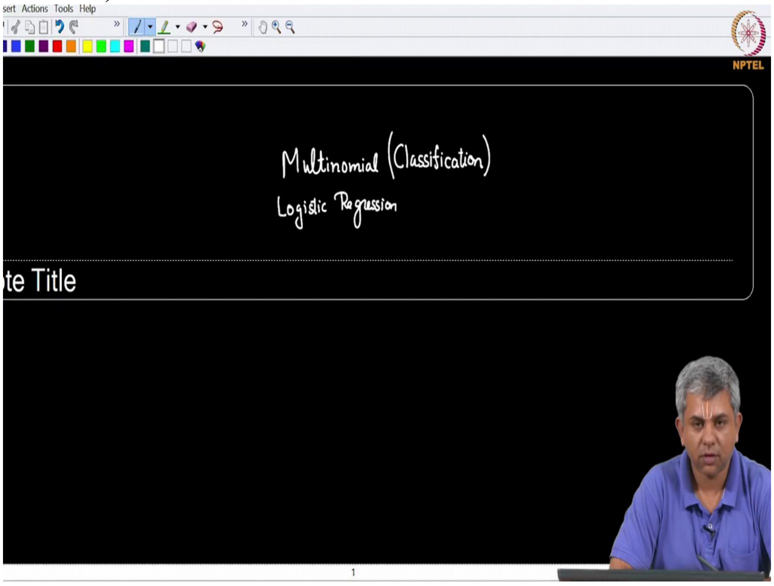
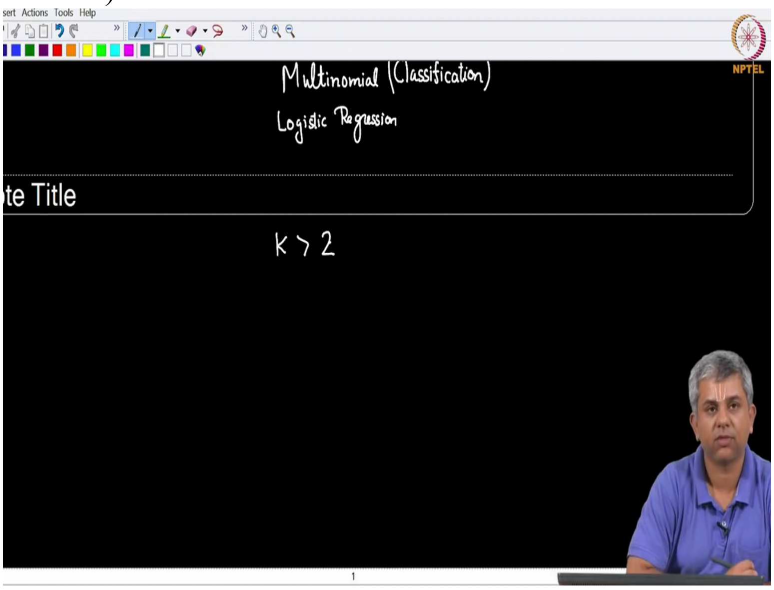
Machine Learning for Engineering and Science Applications
Professor Doctor Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology Madras
Multinomial Classification Softmax

(Refer Slide Time: 00:13)



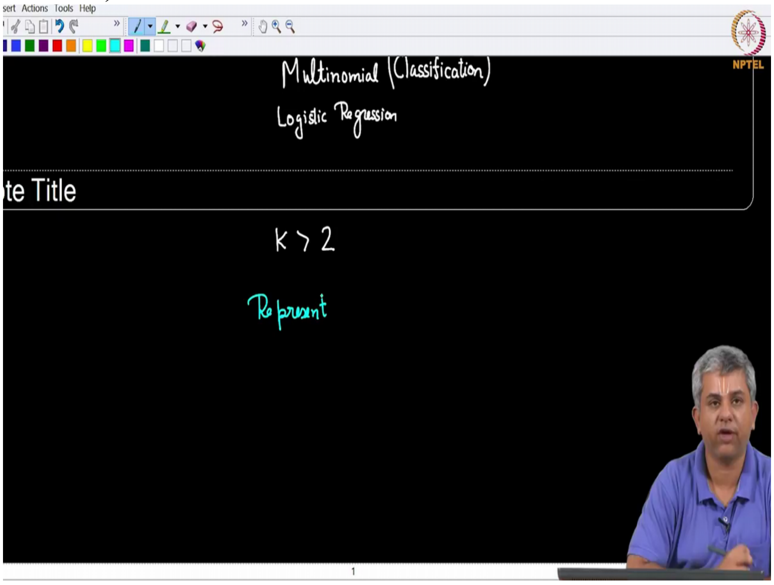
Welcome back. We will now look at some further details of multinomial logistic regression or the classification, multinomial classification algorithm. Remember that multinomial logical regression deals with, when you have k greater than 2 classes.

(Refer Slide Time: 00:31)



So in the last video we saw that in order to represent, remember we had talked about 4 different things that we need to do

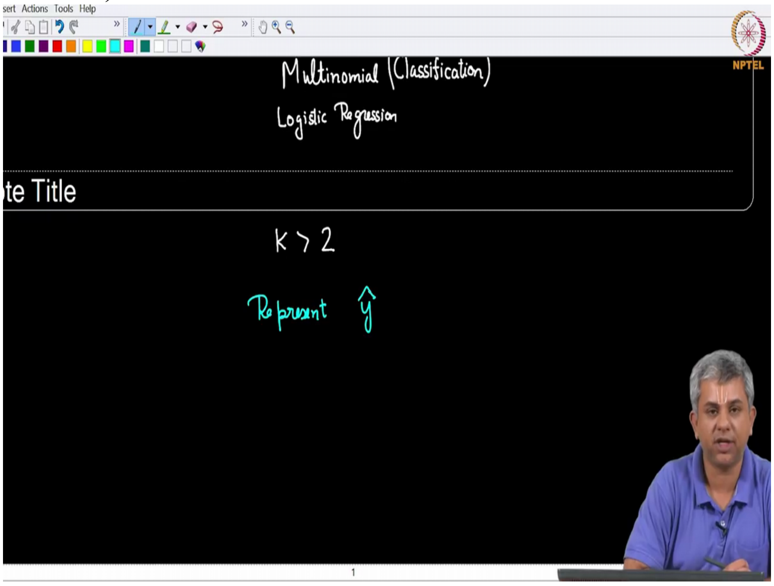
(Refer Slide Time: 00:40)



in order to establish our deep learning model.

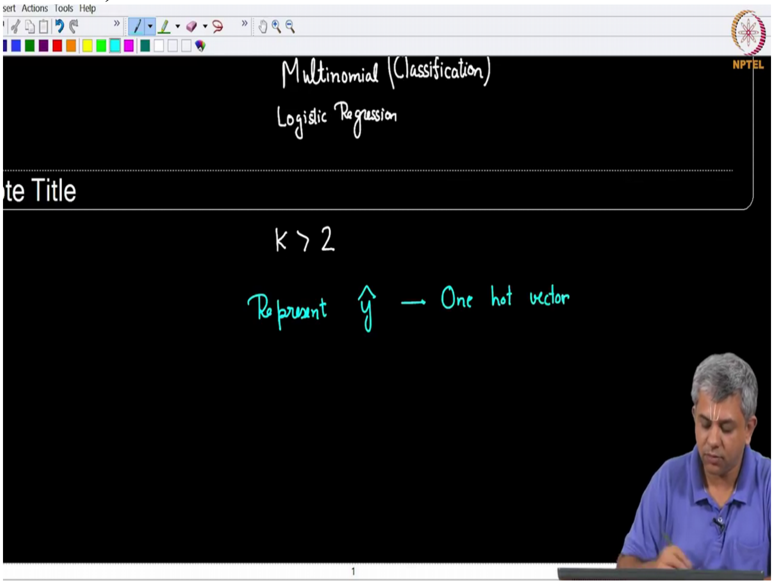
The first thing is

(Refer Slide Time: 00:45)



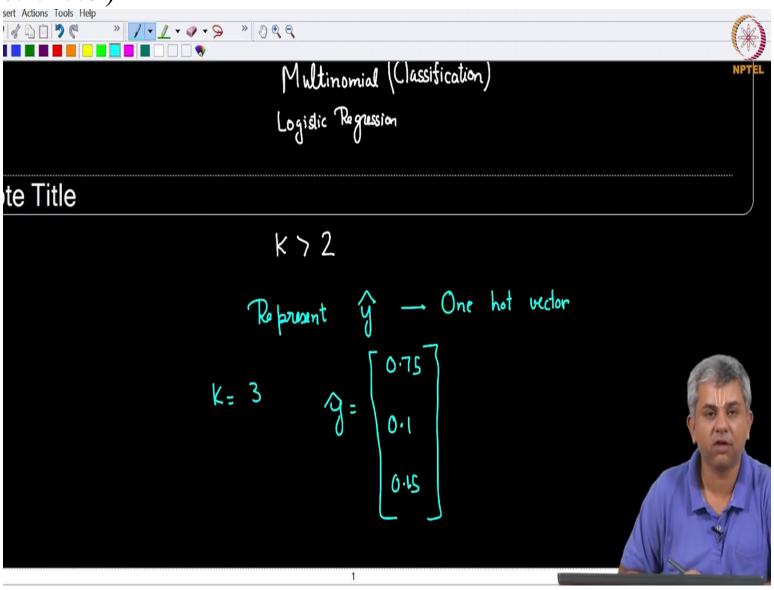
representation of y hat. This we can do with the One Hot Vector.

(Refer Slide Time: 00:55)



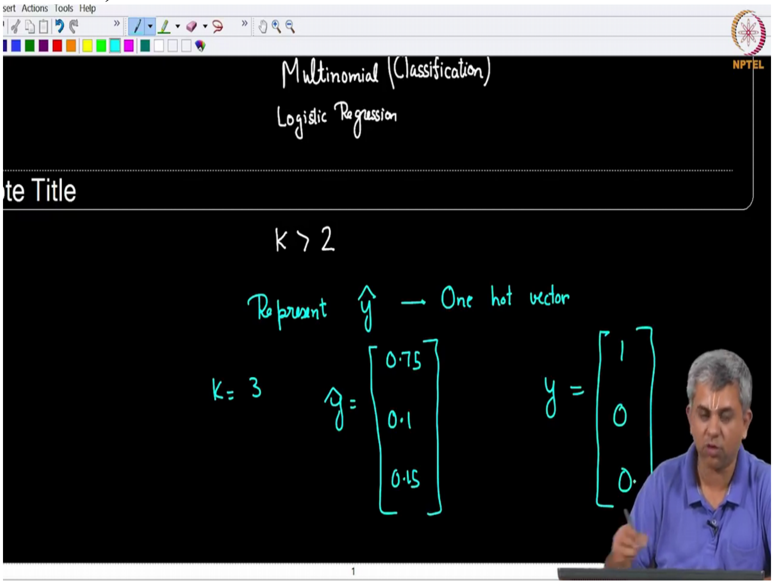
So in case, k is equal to 3, y hat will have some 3 numbers. Ok suppose it is something like point 7 5, point 1, point 1 5

(Refer Slide Time: 01:09)



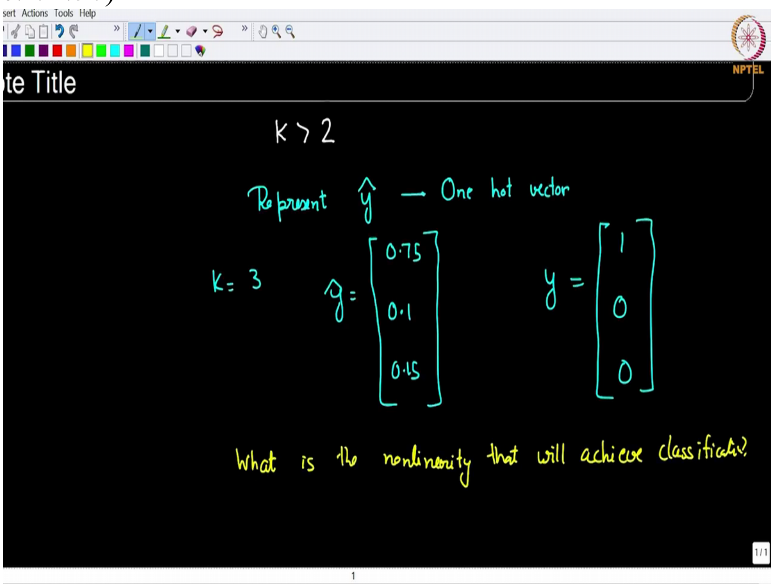
and y itself could be 1 0 0 or 0 1 0 or 0 0 1.

(Refer Slide Time: 01:16)



So you have something of this sort which represents y hat. Now what we need to do next is to find out what is the nonlinearity that will achieve the classification.

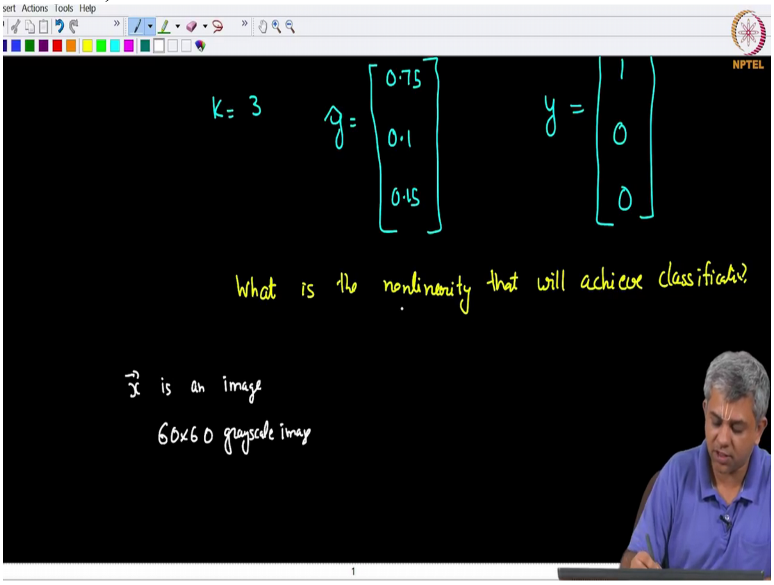
(Refer Slide Time: 01:50)



So let me briefly point out why this is important.

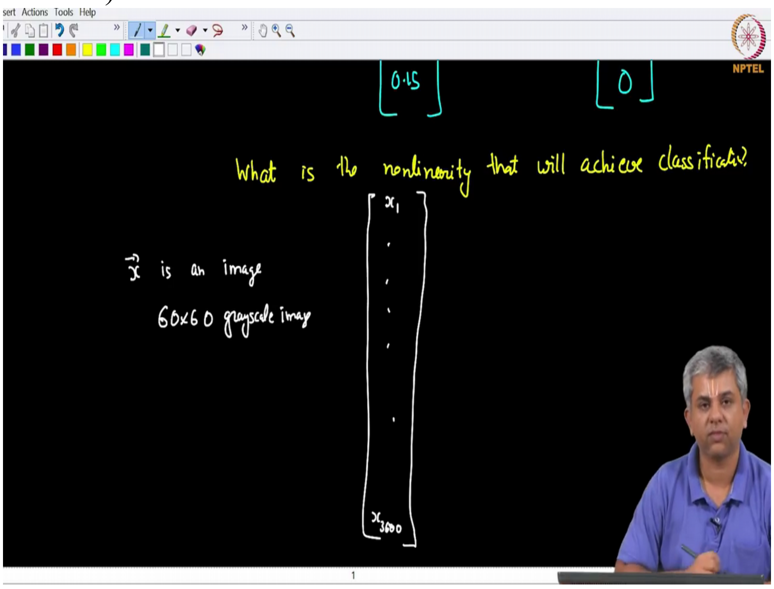
So let us say you have some input x. For the sake of this example let us say x vector is an image. Let us say it is a 60 cross 60 gray scale image,

(Refer Slide Time: 02:16)



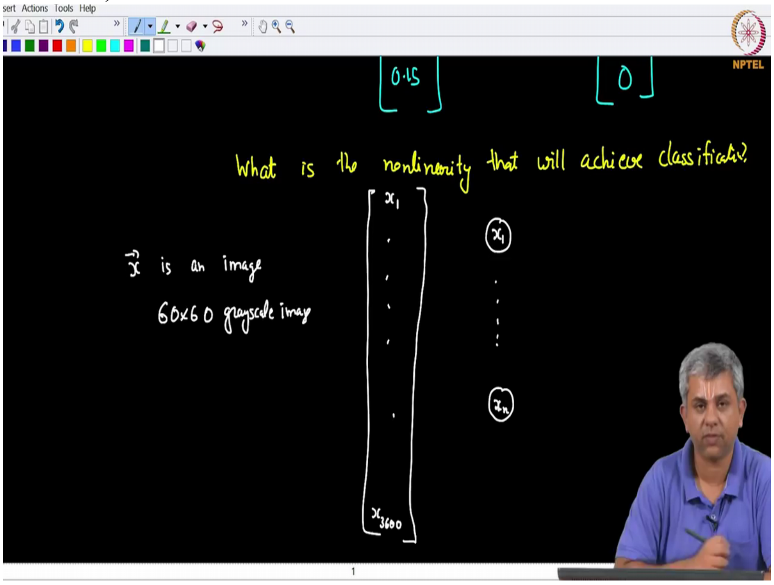
which means x vector, as I have repeated many times can be written simply as 1 unrolled, single unrolled vector which goes from x 1 to x 3600.

(Refer Slide Time: 02:29)



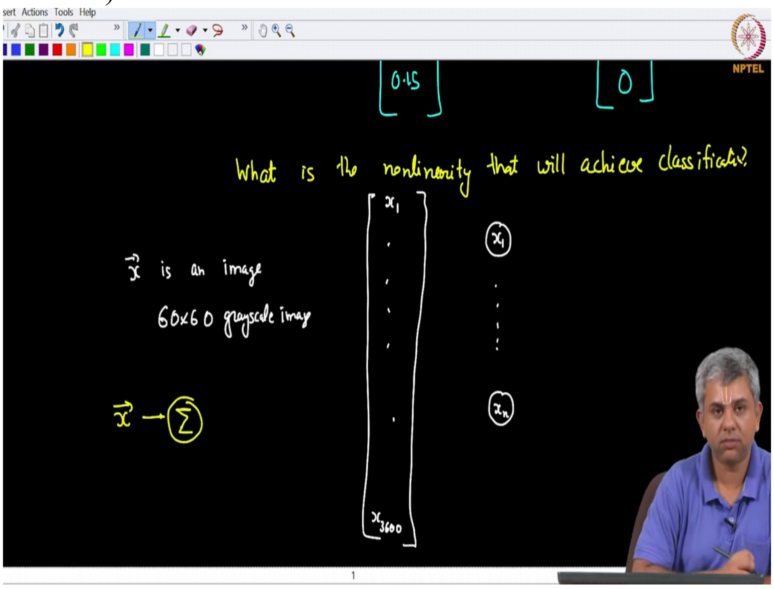
So we will just represent this as x 1 up to x n.

(Refer Slide Time: 02:38)



So all these circles representing different components of this vector, Ok. Now I have that. Now I am going to do the same thing that we did before, x vector goes through a sigma,

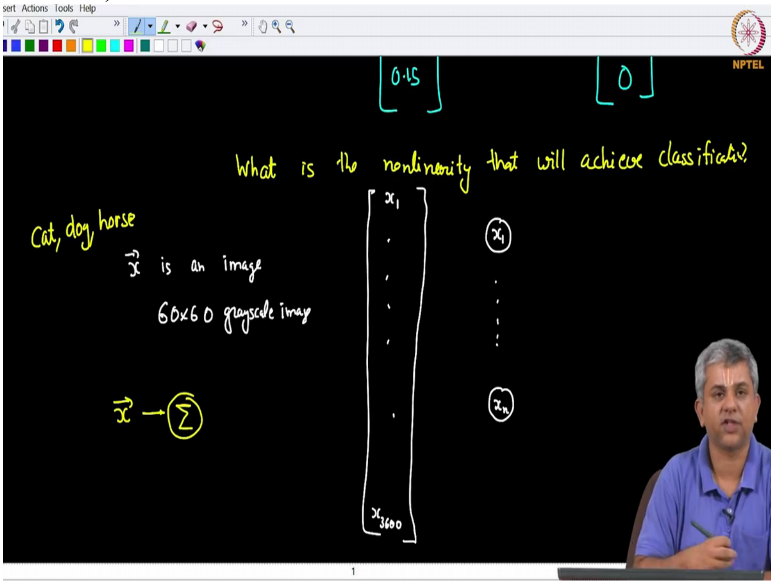
(Refer Slide Time: 02:55)



a summation and we want to classify this image,this grayscale image as one of the classes.

Let us say I will take the same example I have done several times, or I have talked about several times. Let us say this is an image which we know is either of a cat or of a dog or of a horse. You can think of

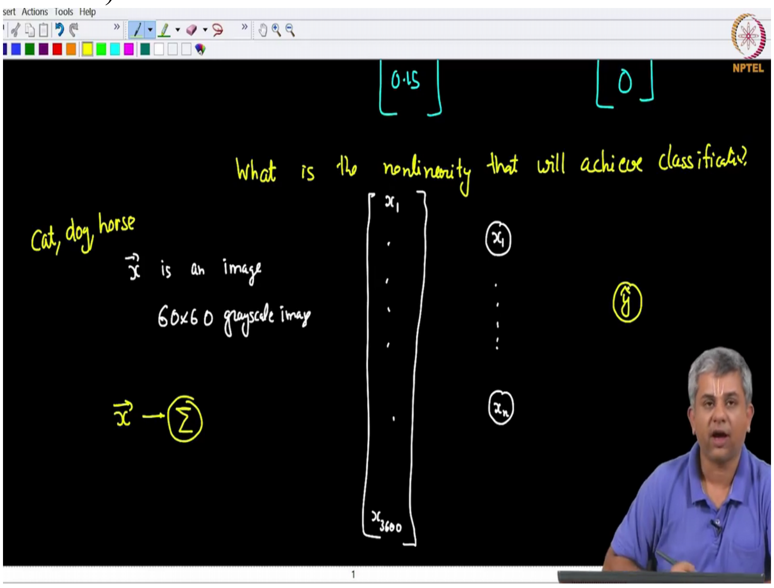
(Refer Slide Time: 03:15)



several engineering examples also.

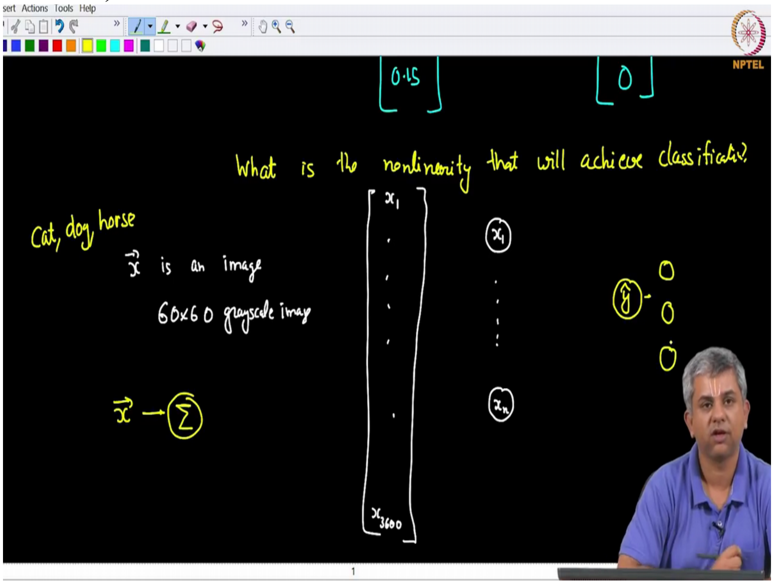
But let us say we use this because they are immediately clear to us. Ok, suppose we have to do this we need a y hat here

(Refer Slide Time: 03:26)



and y hat now is going to have three components;

(Refer Slide Time: 03:33)



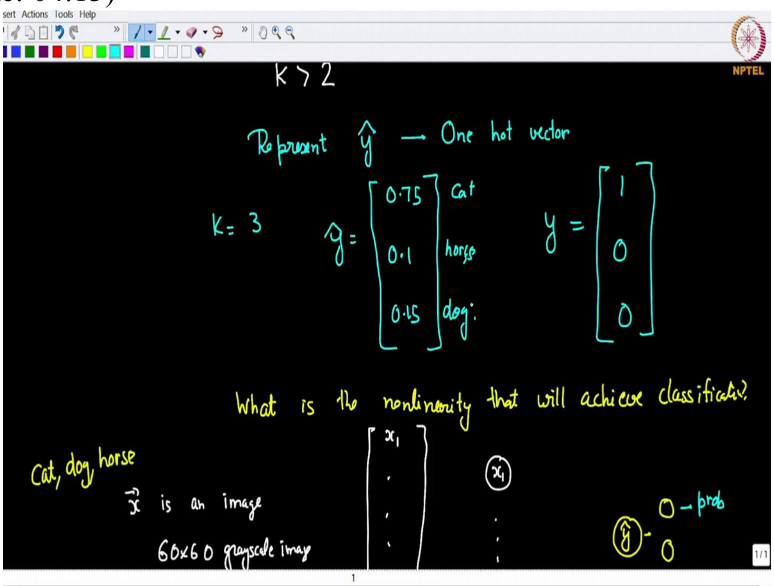
y hat 1, y hat 2, y hat 3 as I have shown above, Ok.

Ideally you would like, you know only of these to be 1, but as we have discussed several times, what you are going to get is actually some number between 0 and 1 for all these three.

Now what is the property that you would like y hat to satisfy? I had already discussed before that each of these is a probability.

So if I get something of this sort I will say that the probability that this image is a cat is point 7 5, probability that it is a horse is point 1 and probability that it is a dog is point 1 5.

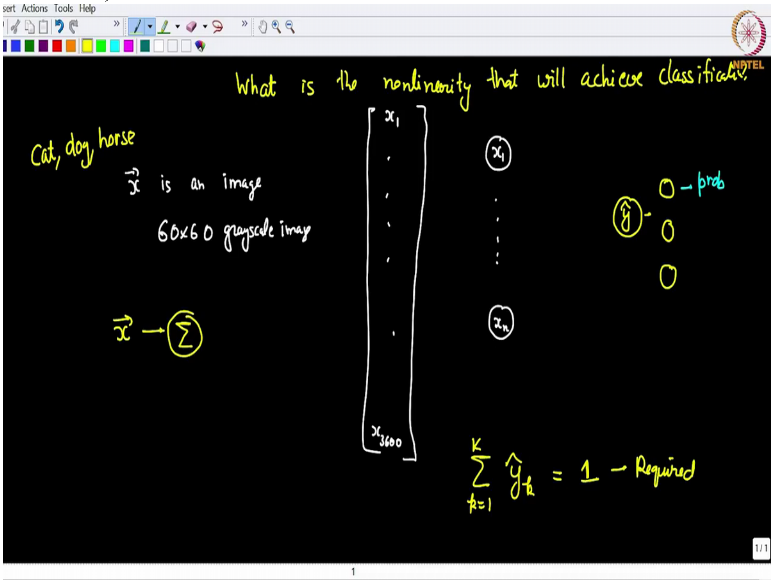
(Refer Slide Time: 04:13)



That is the way I would like to interpret my y hat, Ok.

So if I want to interpret it that way then what do I need? I need that sigma over all the classes of y hat k should be 1, Ok. This is required in case

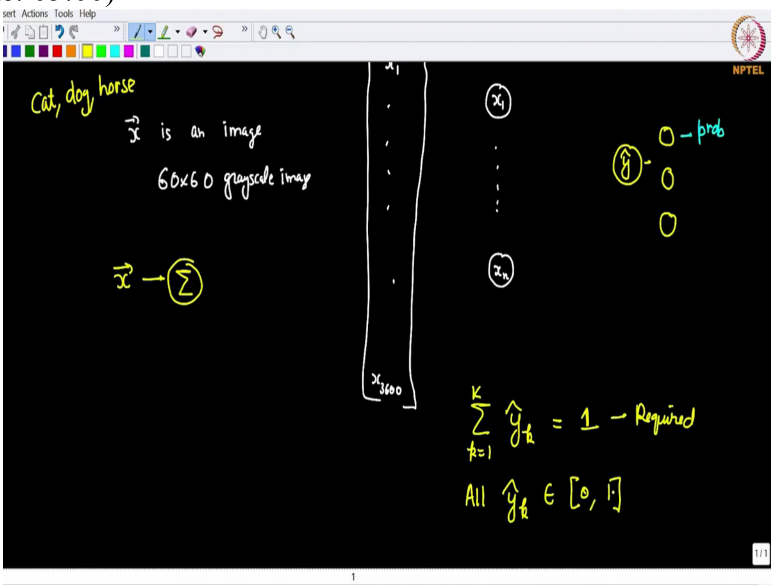
(Refer Slide Time: 04:38)



I want to interpret it in this way like a One Hot Vector. Ok there are other ways of doing it but this is the one that we will stick to for this course, Ok.

This is what we would like to do, Ok. Obviously it also means that all y hat k should also lie between 0 and 1. We do not want them

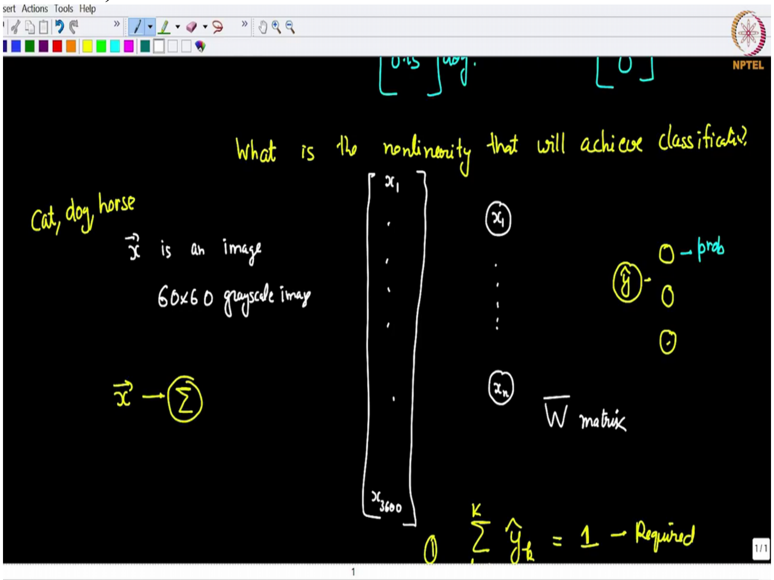
(Refer Slide Time: 05:00)



to be either negative or even greater than 1. That is what we would like to achieve. These two conditions we would like y hat to satisfy.

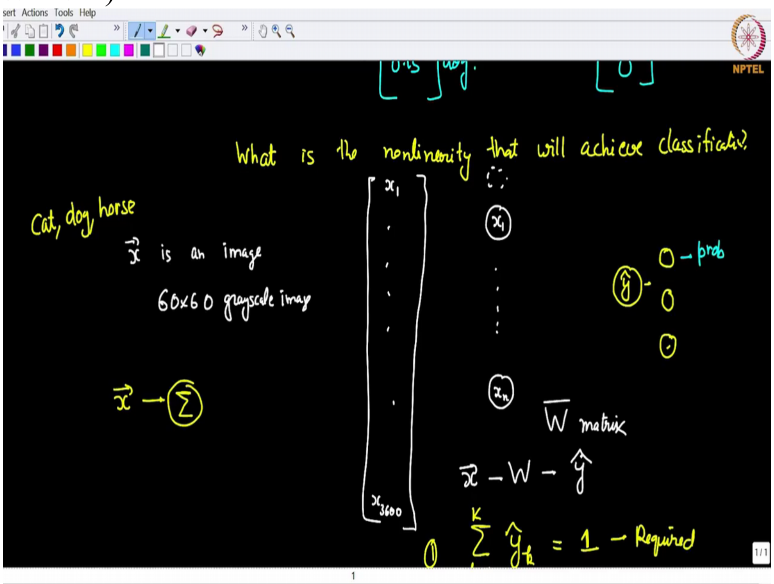
Now remember that before it goes to these 3 outputs you have a W, W matrix actually. What is the size of this matrix? So suppose I ignore the bias term,

(Refer Slide Time: 05:28)



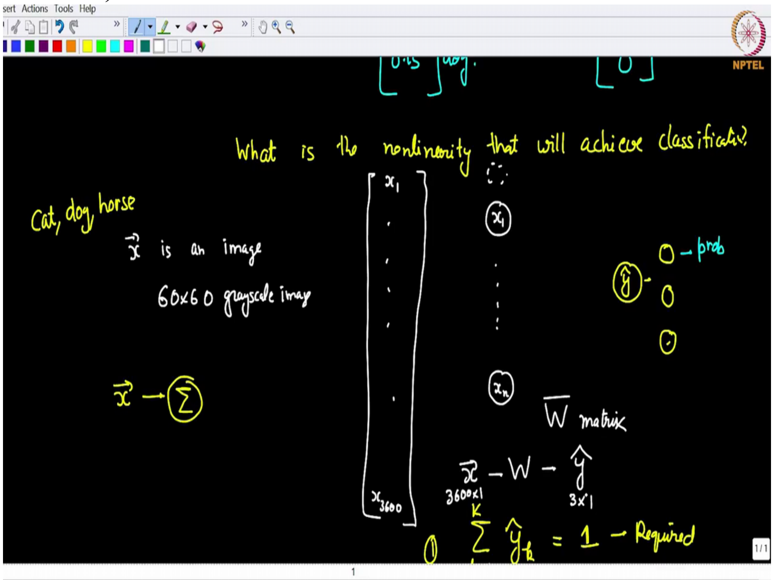
Ok, for now suppose I ignore this term which is the constant term then you will see that each x, so W has to take in x and give out y hat.

(Refer Slide Time: 05:40)



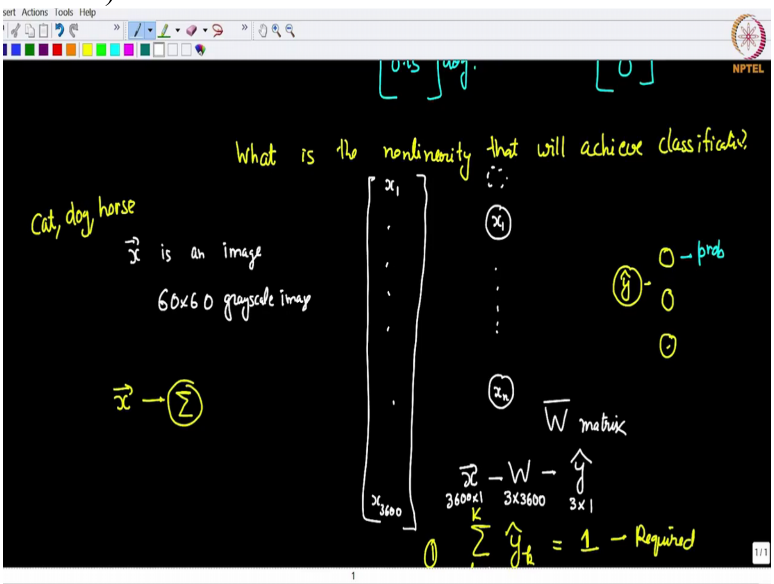
x is of the size 3600 cross 1, y is of the size 3 cross 1.

(Refer Slide Time: 05:48)



So what can you do in order to take this 3600 cross 1 to 3 cross 1? You need a weight matrix that will be of what size, Ok, this is going to be of the size

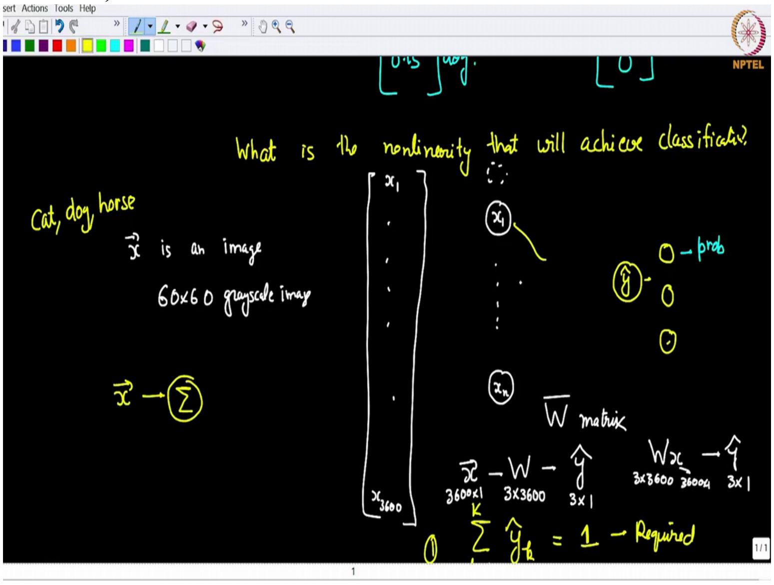
(Refer Slide Time: 06:01)



3 cross 3600. Why? Because then W x, 3 cross 3600 and x is 3600 cross 1, will give you y hat which is 3 cross 1, Ok.

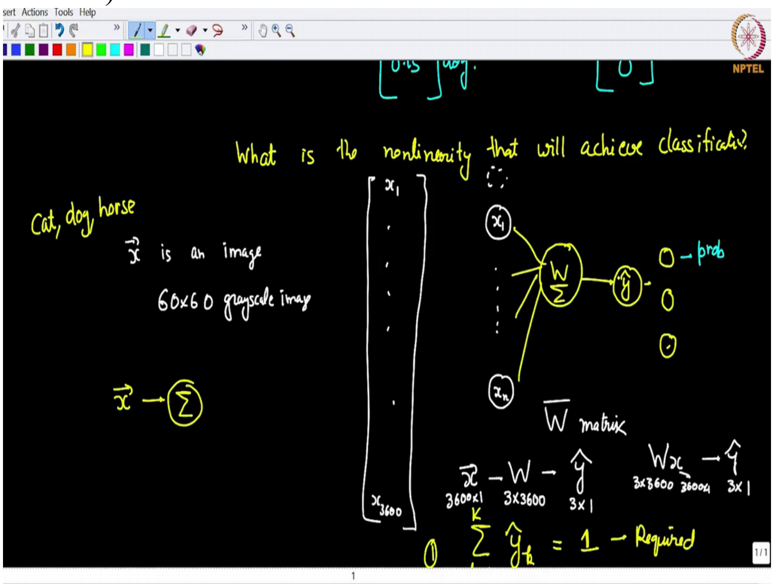
So let us put that in here.

(Refer Slide Time: 06:21)



All these get together through W. There is a summation

(Refer Slide Time: 06:31)



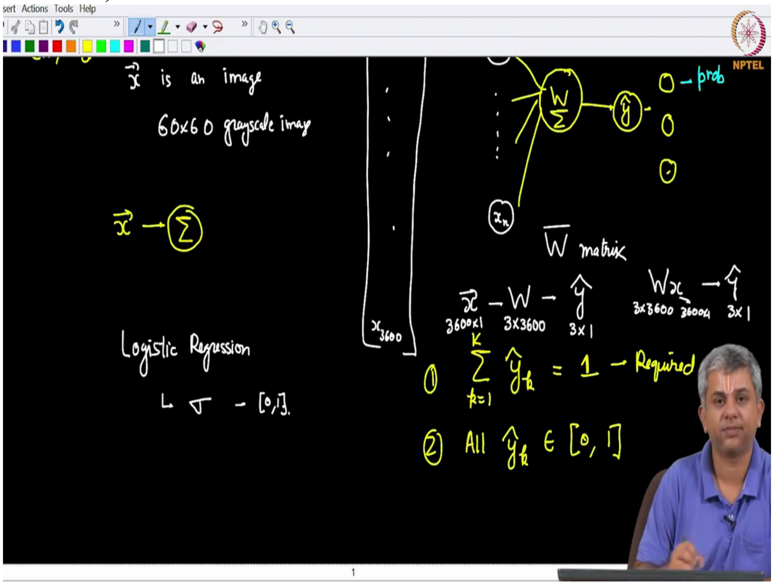
that gives you y hat. Would this be sufficient?

Obviously not because if I take some general weight matrix and just pre-multiply it by x there is no guarantee that these two conditions would be satisfied. This is the same problem that we faced while doing logistic regression also

That is W x is of the right size but now I am not sure that when I apply, when I simply apply a linear combination that it is going to give me a number between 0 and 1. Which is why we use a squeezing function just like we did in logistic regression.

So in logistic regression we use the simple squeezing function. The squeezing function was sigmoid. And sigmoid gave us between 0 and 1.

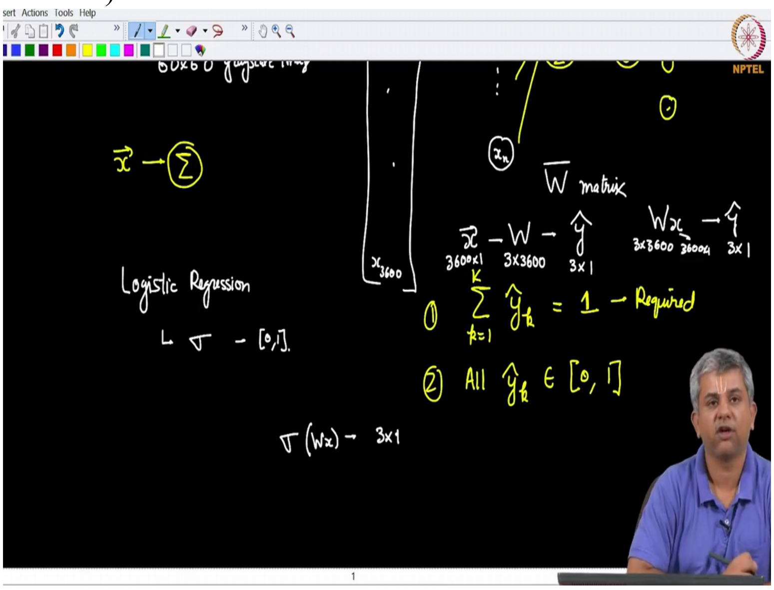
(Refer Slide Time: 07:21)



Now we could think why not do the same thing here? Ok.

So I have W x, so if I apply sigmoid of W x this will also give me a 3 cross 1

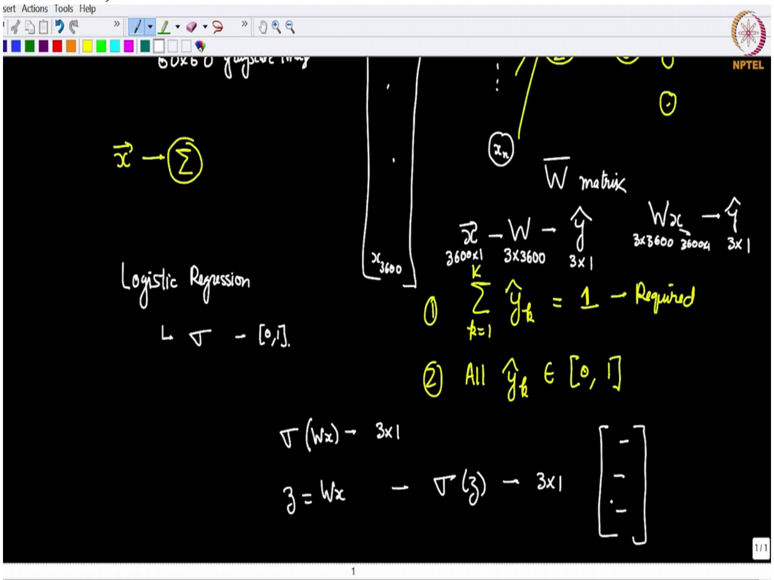
(Refer Slide Time: 07:34)



vector, each of these numbers will be between 0 and 1. So notice the operation I am doing. I find out z equal to W x. Then I do sigmoid of z. This will also be a 3 cross 1 vector.

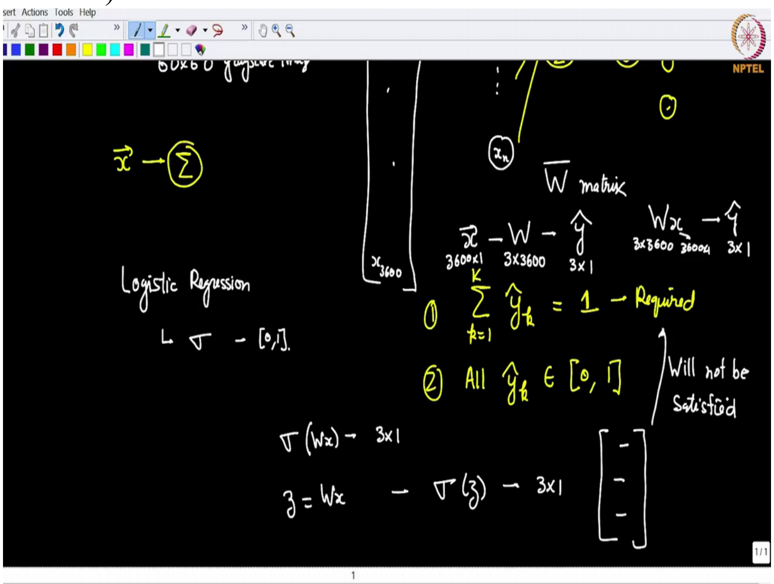
Each of these numbers will be between

(Refer Slide Time: 07:56)



0 and 1. Now why not use that? There is one small problem. The problem is this will not be satisfied,

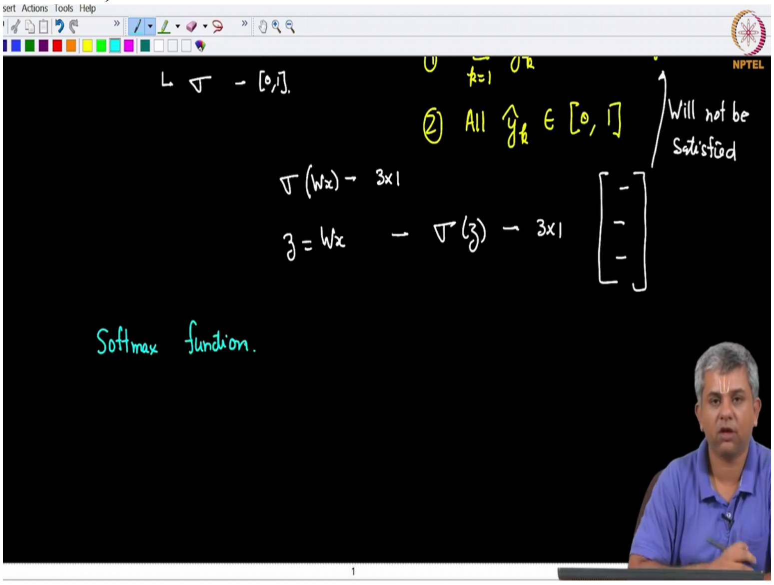
(Refer Slide Time: 08:10)



Ok. So if you arbitrarily apply sigmoid to 3 random numbers you are ascertain or you are not certain that the sum of those numbers will always stick to 1.

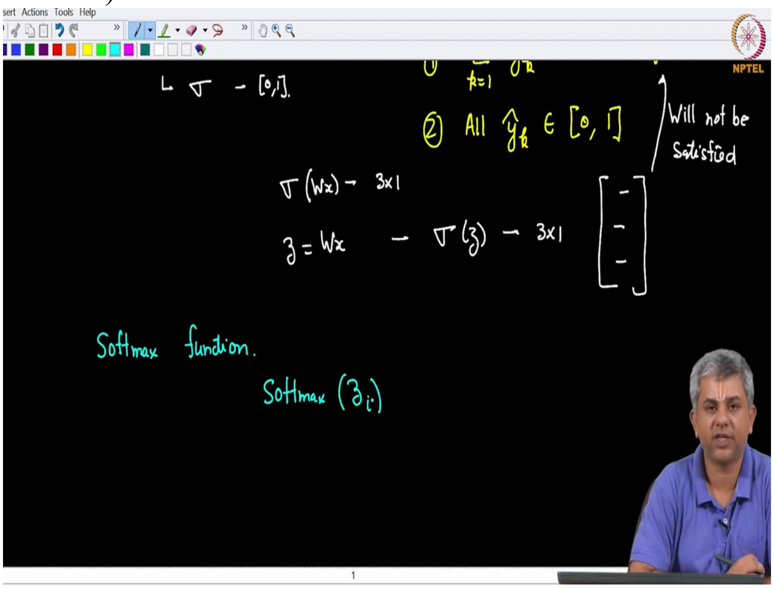
So what do we do? We do a simple function called the Softmax function.

(Refer Slide Time: 08:35)



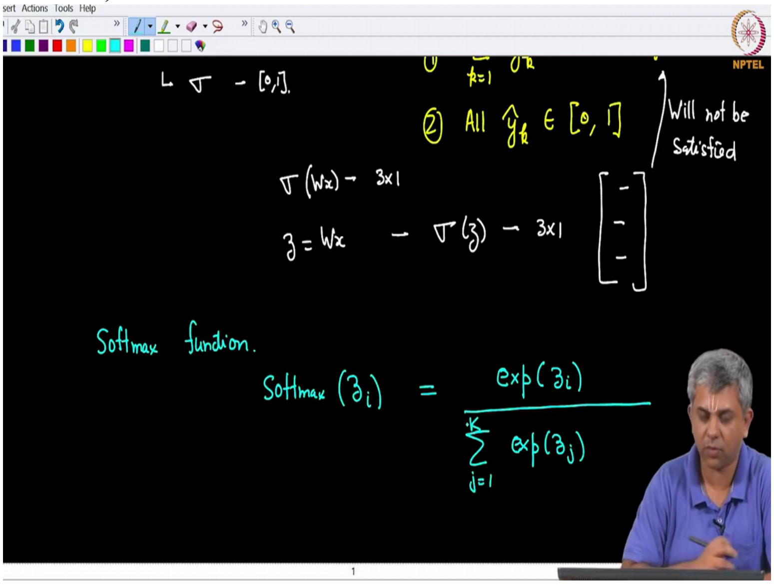
So the Softmax function works in a very simple way. Softmax

(Refer Slide Time: 08:47)



of z i is equal to exponential of z i divided by ...

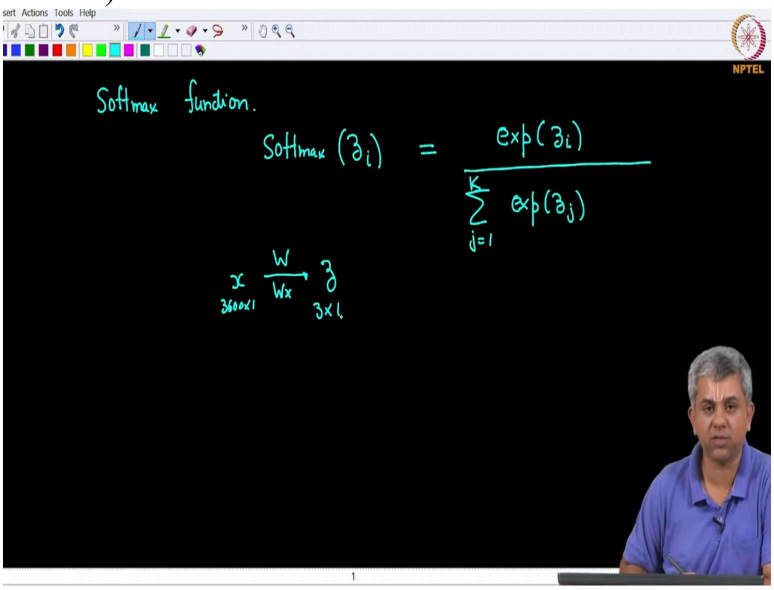
(Refer Slide Time: 09:09)



So it is simply normalizing the exponentials of all these components, Ok.

So let me show this in a simple way. So suppose you have x, once again 3600 cross 1, you apply W, you get z. z is now 3 cross 1. Remember W x becomes 3 cross 1.

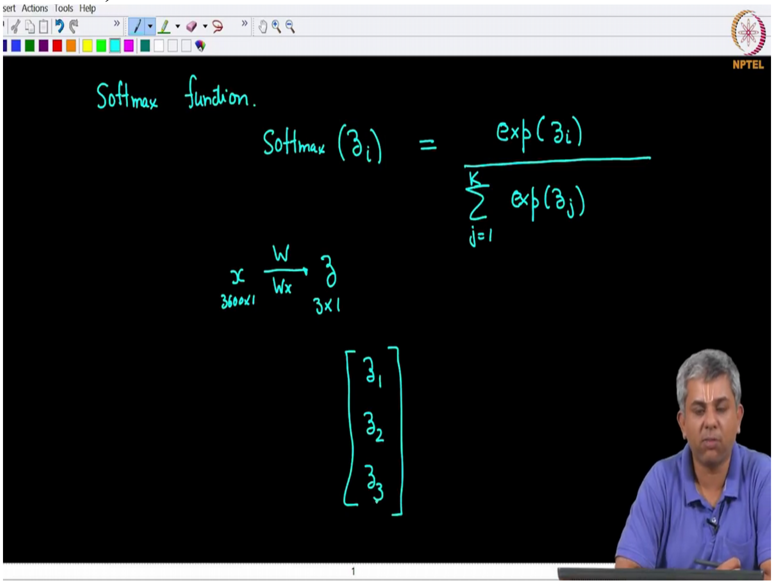
(Refer Slide Time: 09:34)



And now you have 3 numbers, z 1, z 2, z 3.

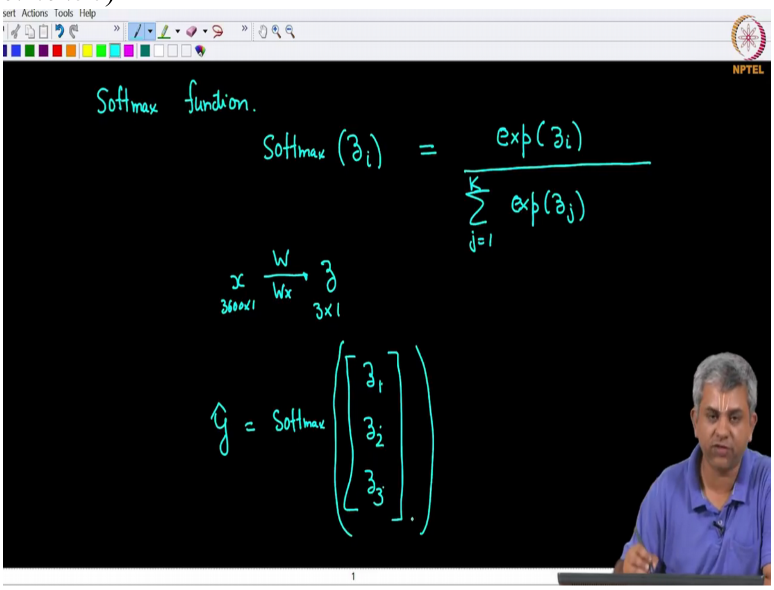
Our problem

(Refer Slide Time: 09:43)



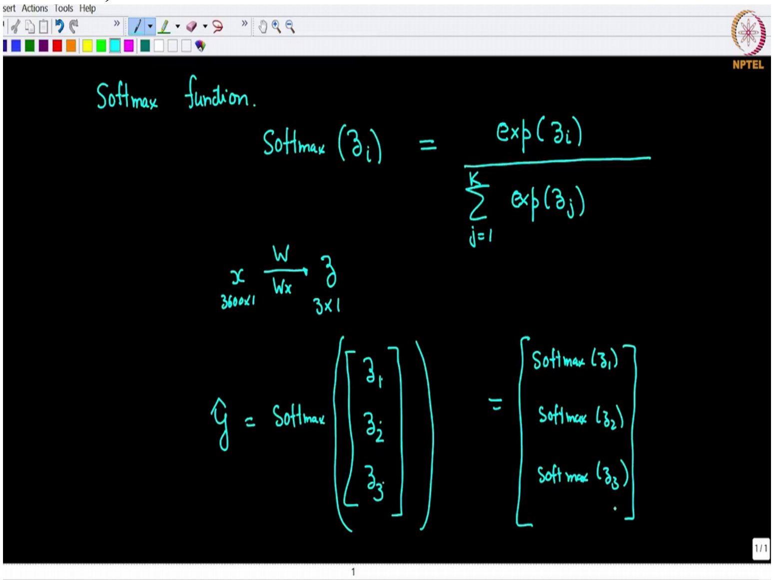
of course is z 1, z 2, z 3 are not between 1. So what do we do? We say y hat is equal to Softmax of these 3,

(Refer Slide Time: 09:56)



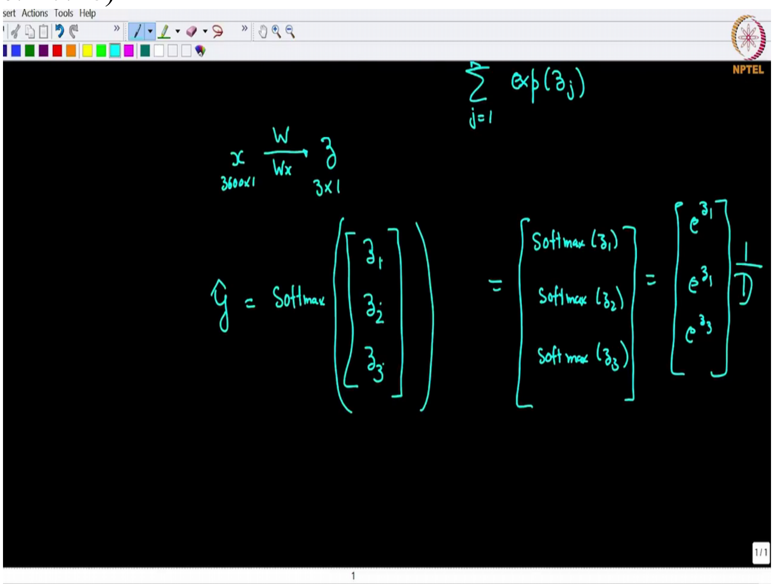
which is the same as Softmax of z 1, Softmax of z 2 and Softmax of z 3.

(Refer Slide Time: 10:14)



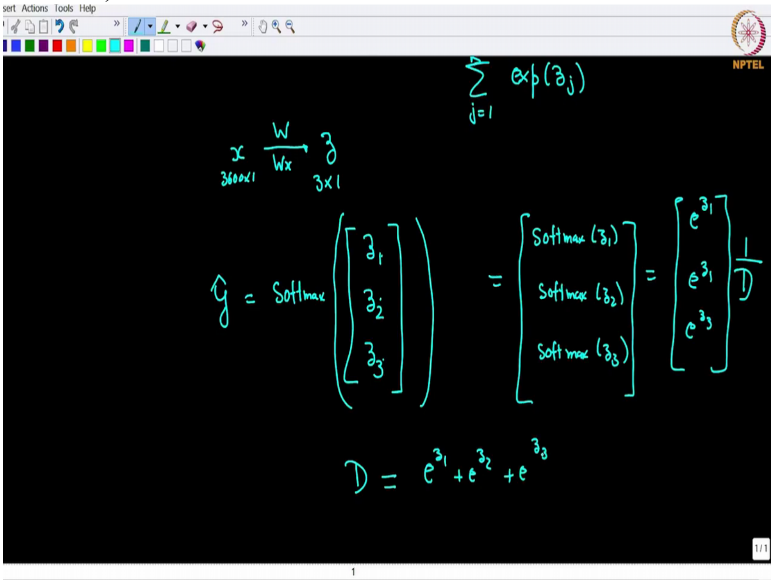
What does this do? This is equal to e power z 1, e power z 2, e power z 3; all three multiplied by 1 by some denominator where

(Refer Slide Time: 10:28)



the denominator is e power z 1 plus e power z 2 plus e power z 3.

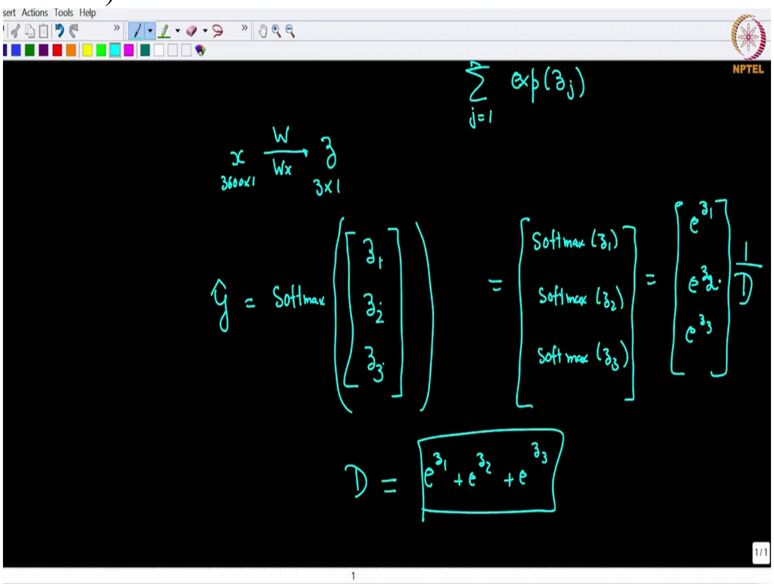
(Refer Slide Time: 10:36)



You will notice automatically that both our conditions are satisfied, Ok because e power z 1 by this sum is always going to be between 0 and 1, Ok, since the exponentials are positive functions, it is always going to be between 0 and 1.

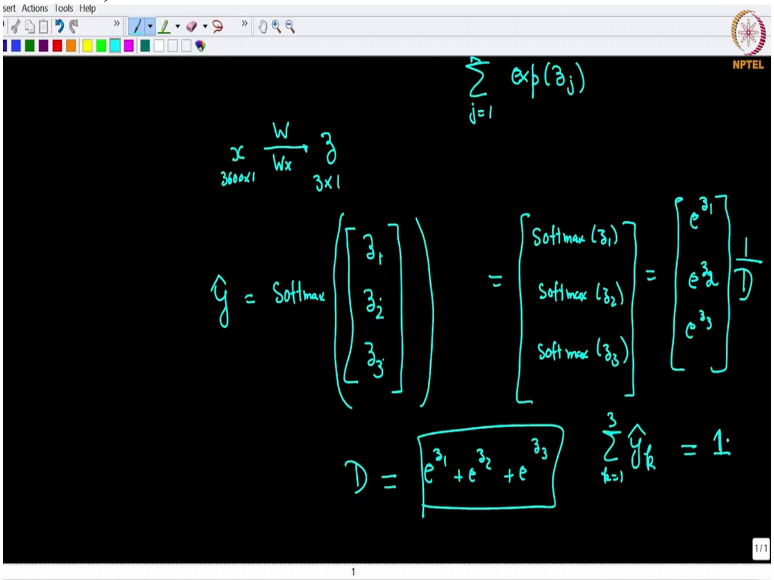
Another thing is the sum of these 3 should be z 2.

(Refer Slide Time: 10:55)



So we also get the condition that sigma of y hat k between k equal 1 to 3 is equal to 1.

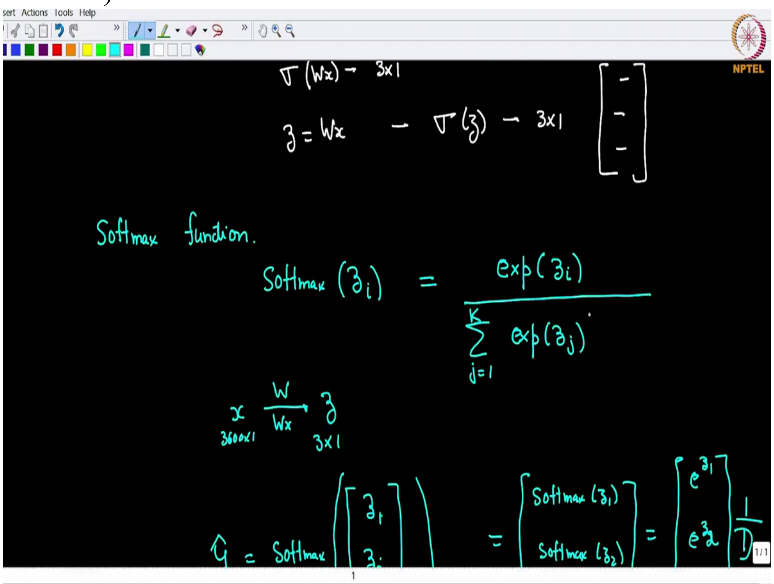
(Refer Slide Time: 11:05)



So both our conditions are satisfied. Some of you might recall that in week 3 we had seen that the practical computation of Softmax you have to be a little bit careful.

If you compute the

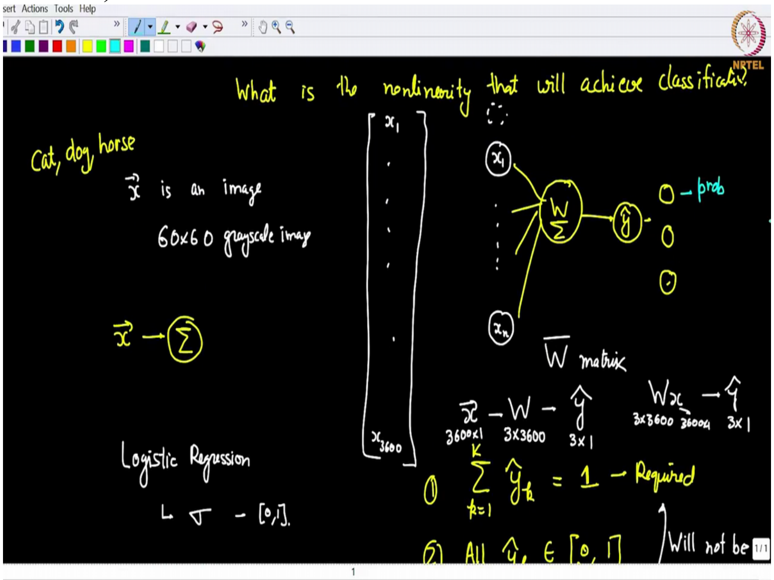
(Refer Slide Time: 11:18)



numerator and the denominator separately as I have shown here, sometimes you might run into overflow problems. We had also looked at a solution to that within week 3 itself. So I would ask you to look at that in case you have forgotten it.

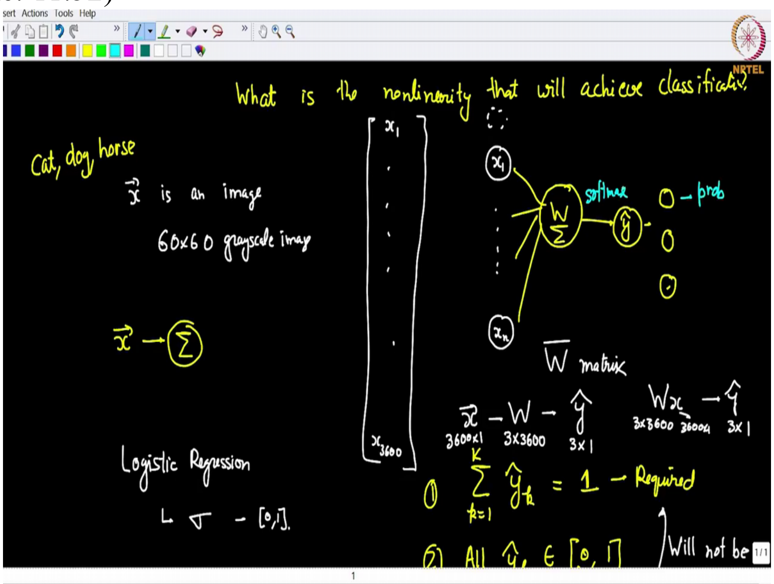
So just recapitulate what we have done in this video. It is a very simple idea. In case you have One Hot Vector

(Refer Slide Time: 11:38)



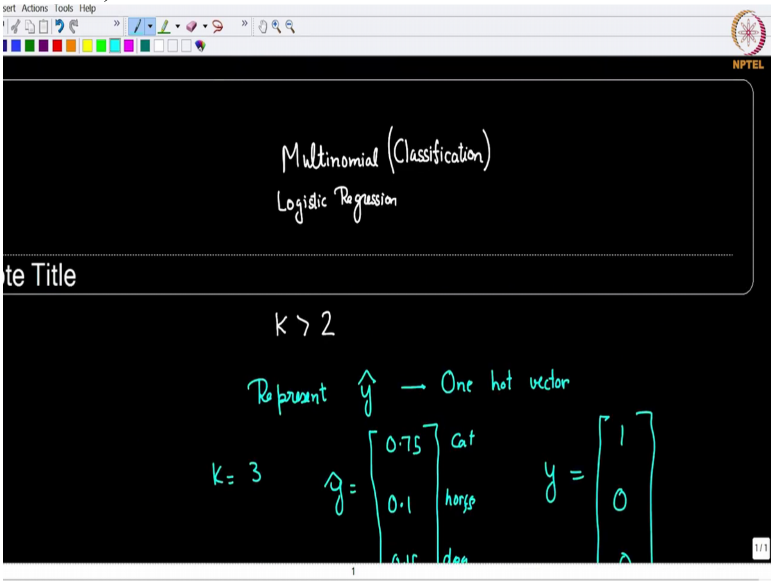
as a classification representation of your final output, all you need to do in the final layer or in the layer after the linear combination is to add a Softmax, Ok.

(Refer Slide Time: 11:52)



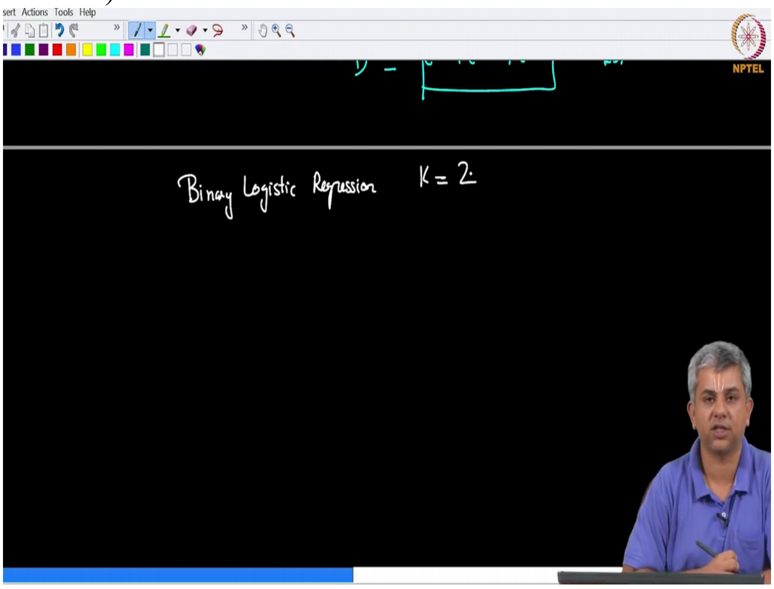
So once you add that Softmax you get a proper classification and this is your forward model for

(Refer Slide Time: 11:59)



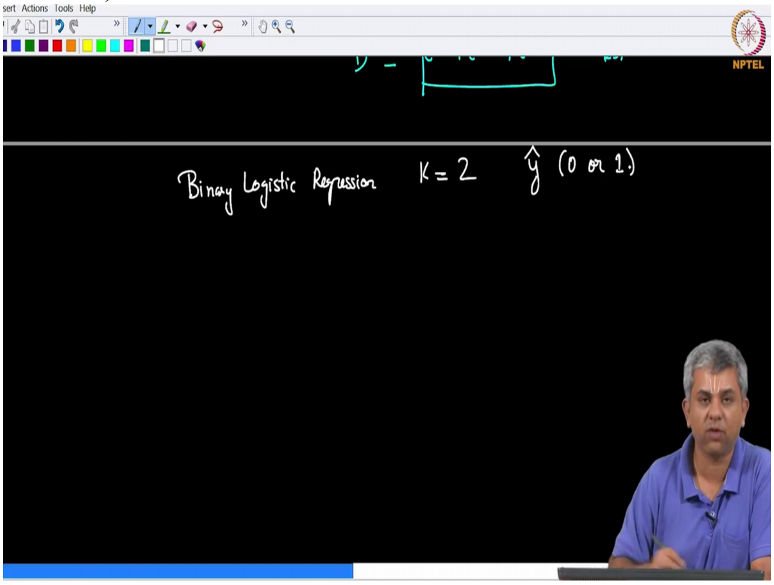
the multinomial logistic regression case. So recall that we had looked at 2 things, the binary logistic regression. In this case you have 2 classes.

(Refer Slide Time: 12:20)



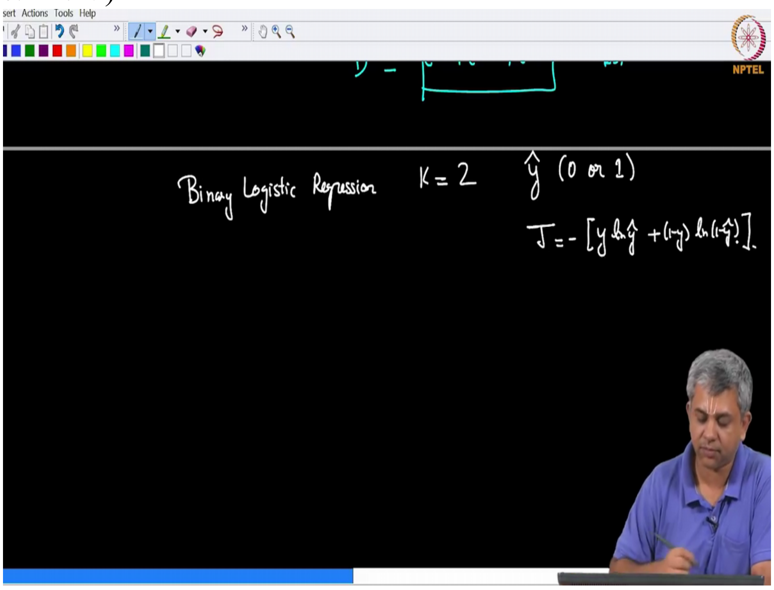
Your y hat typically, it is easier to just represent it as a scalar, a 0 or a 1, Ok.

(Refer Slide Time: 12:29)



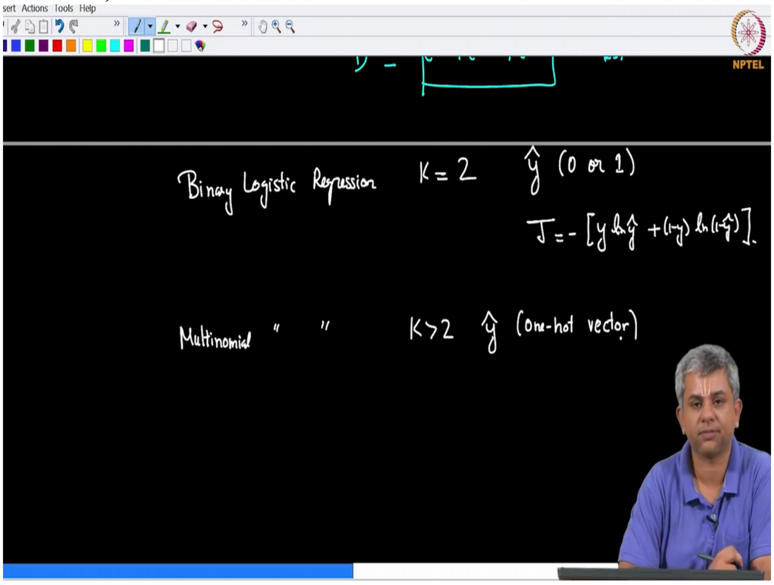
And we have our binary cross entropy loss function which was minus y l n y hat plus 1 minus y l n 1 minus y hat.

(Refer Slide Time: 12:45)



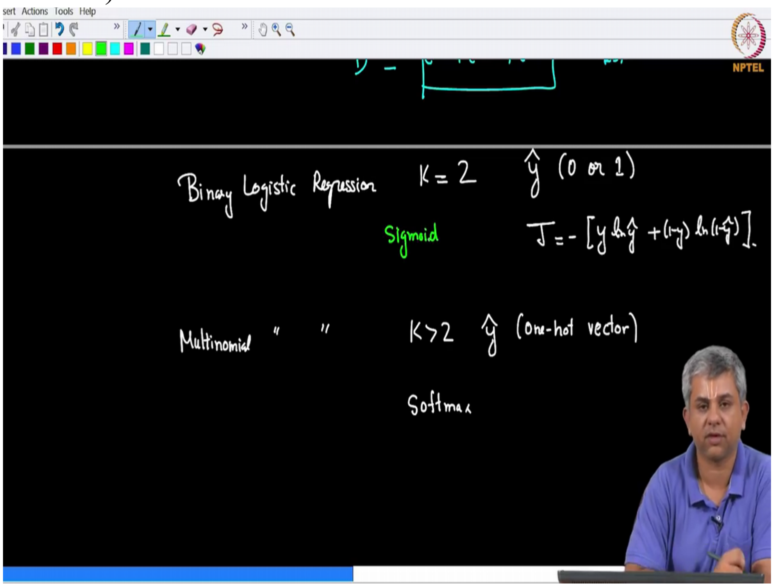
And then you have multinomial logistic regression where k is greater than 2. y hat now is a One Hot Vector

(Refer Slide Time: 13:05)



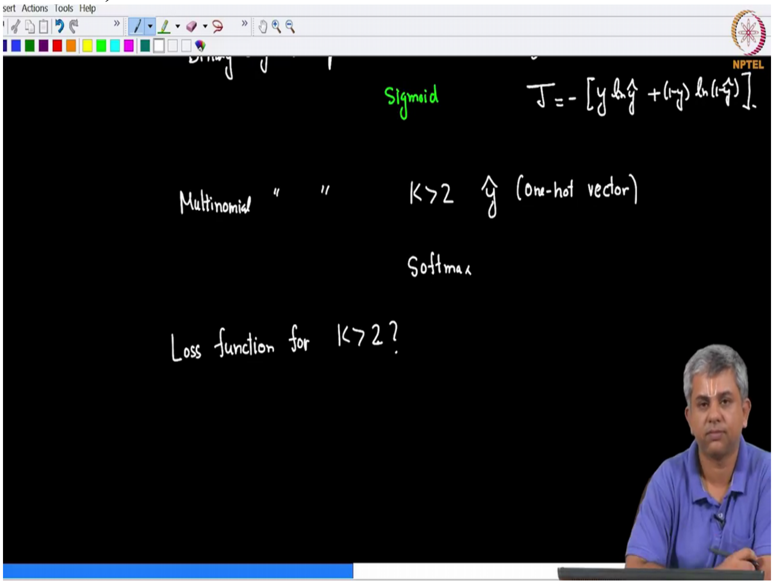
and the nonlinearity we use here is Softmax. The nonlinearity we used here was sigmoid.

(Refer Slide Time: 13:17)



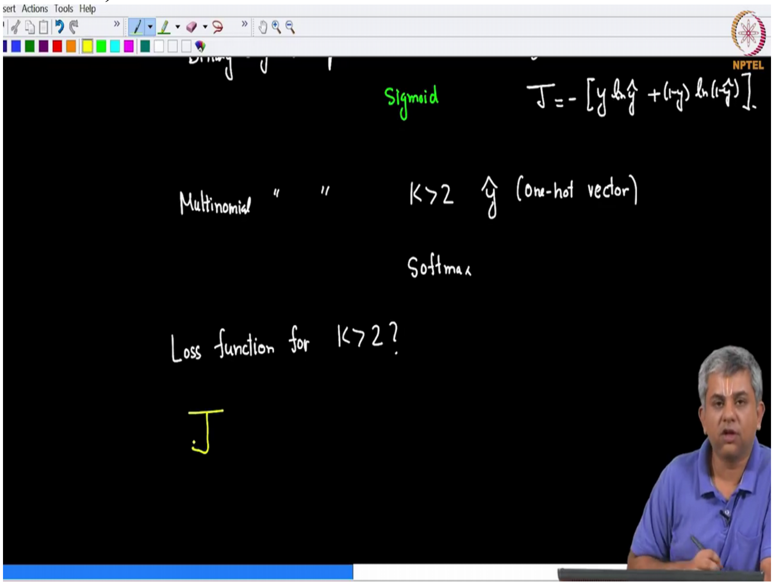
Now what do we do about J?

(Refer Slide Time: 13:32)



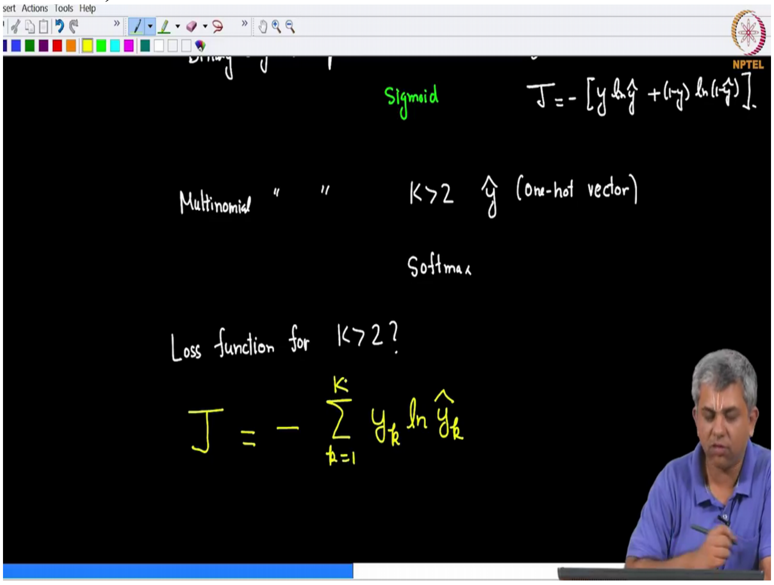
So that is the last problem that we have to solve here. As it turns out that this is also fairly straight forward. I will write it down right now.

(Refer Slide Time: 13:43)



The cost function for the multinomial case is minus sigma y k l n y k hat. This is it, for k equal to

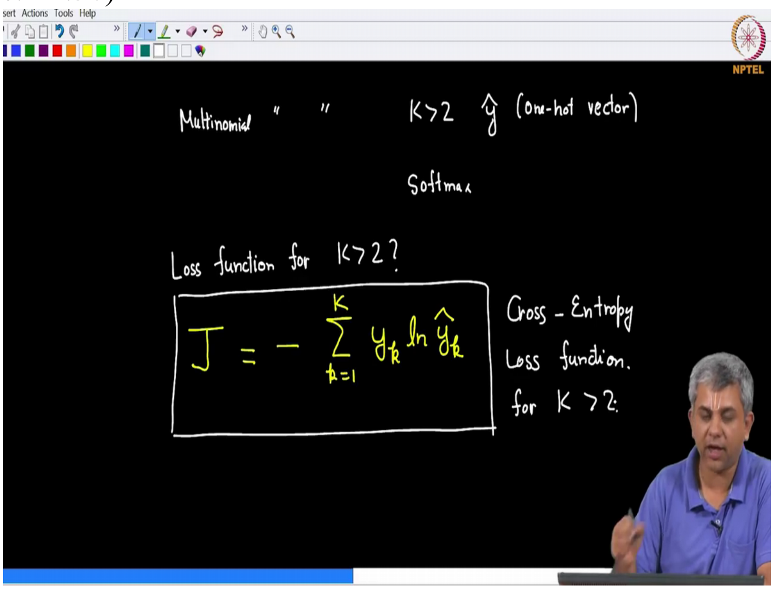
(Refer Slide Time: 14:00)



1 to capital K classes.

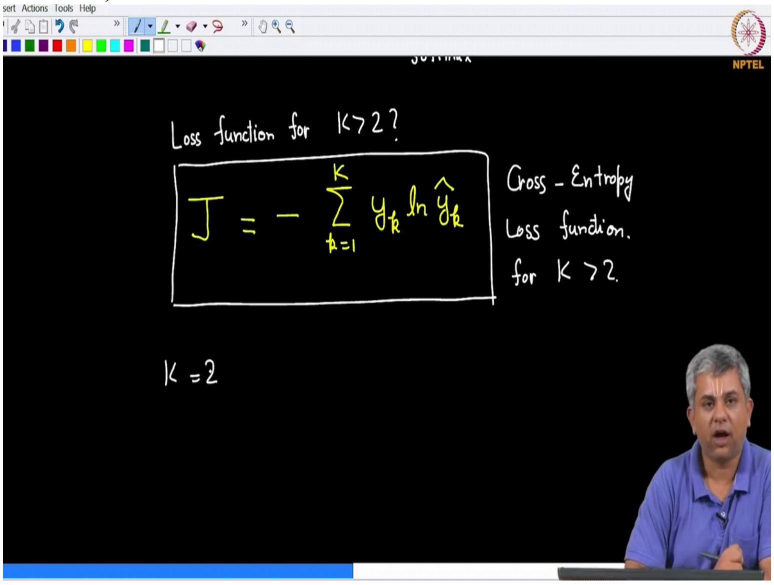
Now you might think about what happened about, you know this 1 minus y, 1 minus y hat, Ok. Why is this looking slightly different from here? This is also a cross entropy loss function for k greater than

(Refer Slide Time: 14:36)



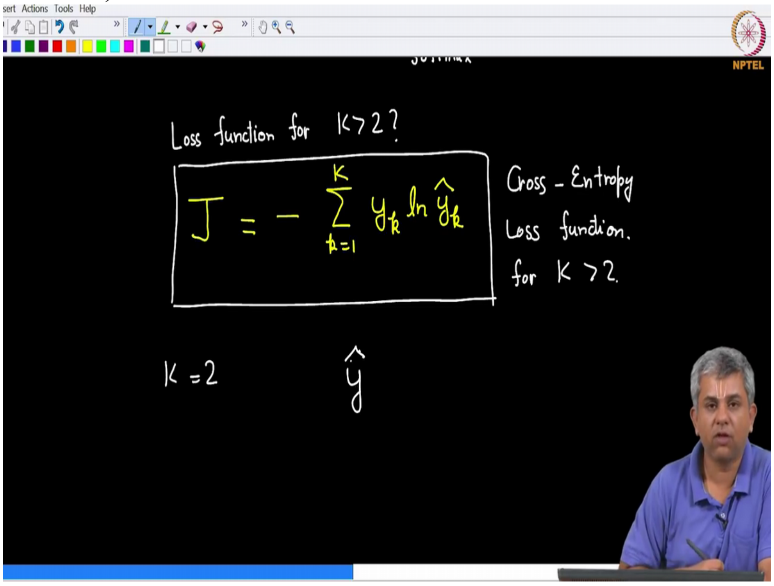
2. Now what happens at k equal to 2?

(Refer Slide Time: 14:43)



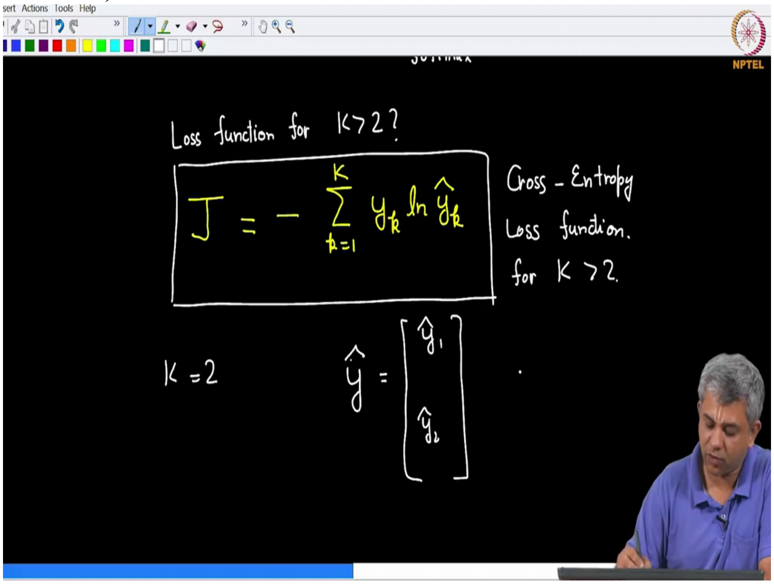
I want to show you that the binary cross entropy loss function actually becomes equivalent to this in the case of k equal to 2. So let us say you have y hat

(Refer Slide Time: 14:55)



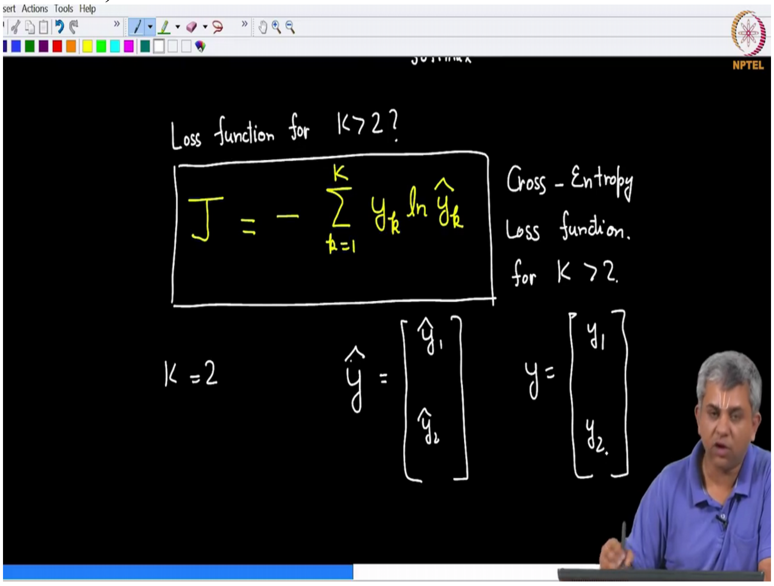
in the case of k equal to 2 and we represent it as a One Hot Vector. This is y hat 1, y hat 2.

(Refer Slide Time: 15:06)



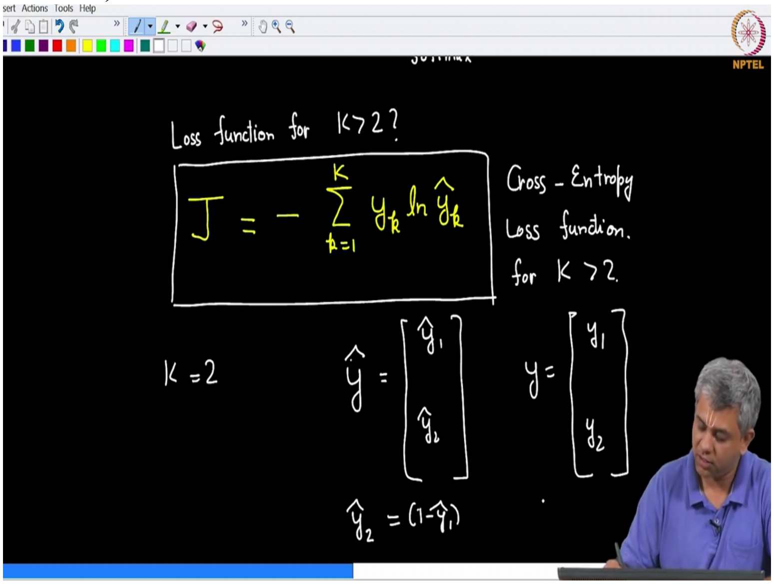
Similarly y is y 1 and y 2.

(Refer Slide Time: 15:12)



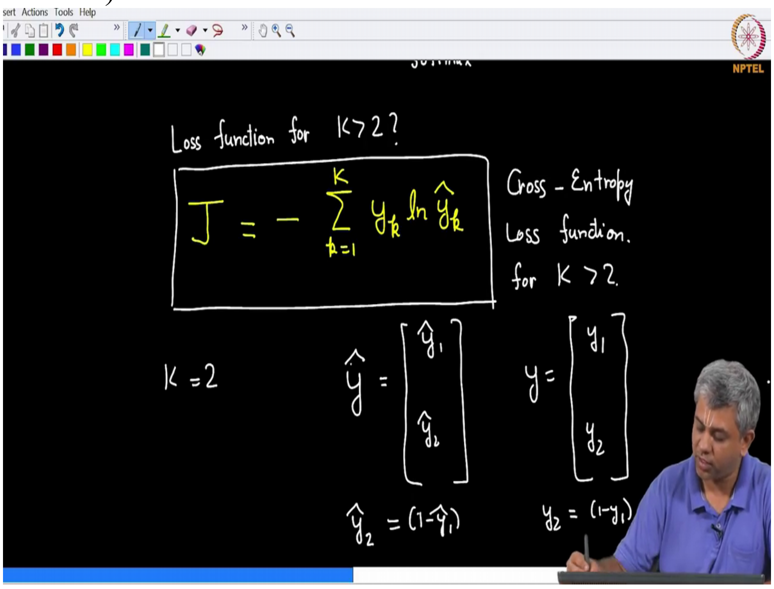
Now if it is a binary problem, it is either this or that. Therefore y hat k has to be equal to, or let me say this way; y hat 2 has to be equal to 1 minus y hat 1.

(Refer Slide Time: 15:26)



Similarly y 2 is equal to

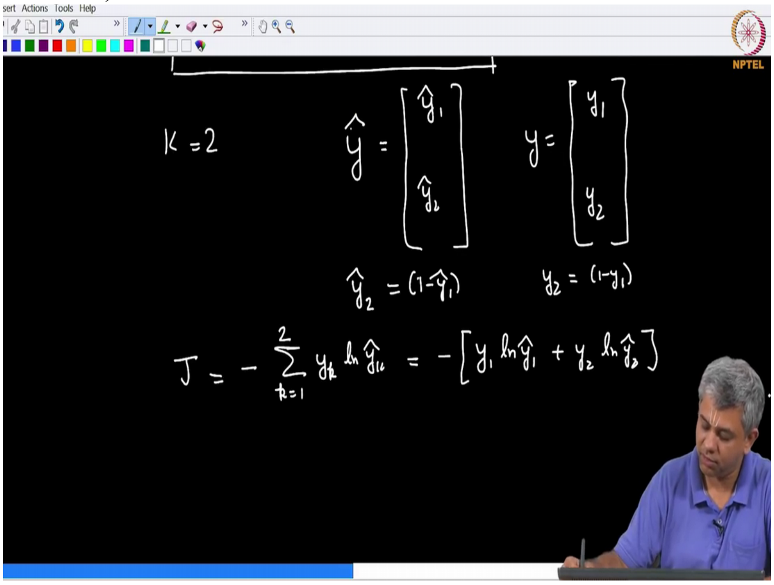
(Refer Slide Time: 15:29)



1 minus y 1.

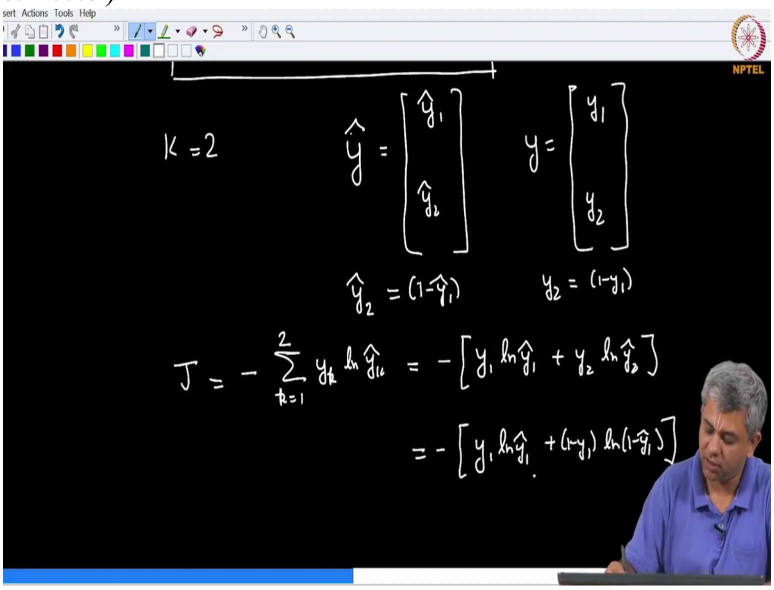
So if we run it through this formula we get J minus k equal to 1 to 2 y k l n y k hat which simply becomes minus y 1 l n y 1 hat plus y 2 l n y 2 hat

(Refer Slide Time: 15:56)



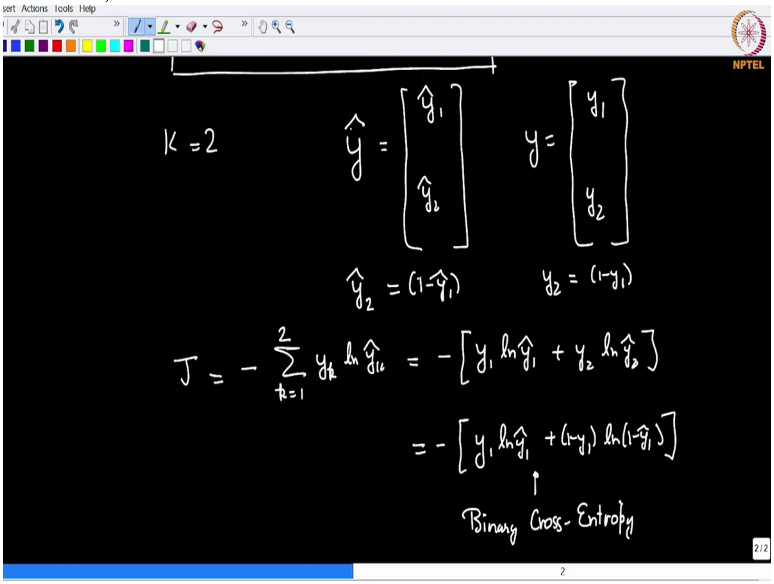
and from these two relations this is simply minus y 1 l n y 1 hat plus 1 minus y 1 l n 1 minus y 1 hat

(Refer Slide Time: 16:09)



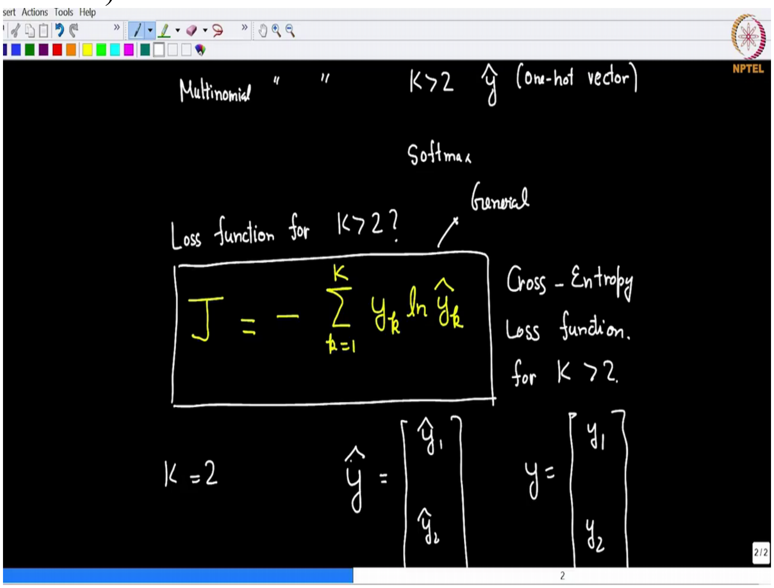
which is the same as the binary cross entropy loss function.

(Refer Slide Time: 16:18)



So this is just to say that this is a general formula. You can think of all

(Refer Slide Time: 16:27)

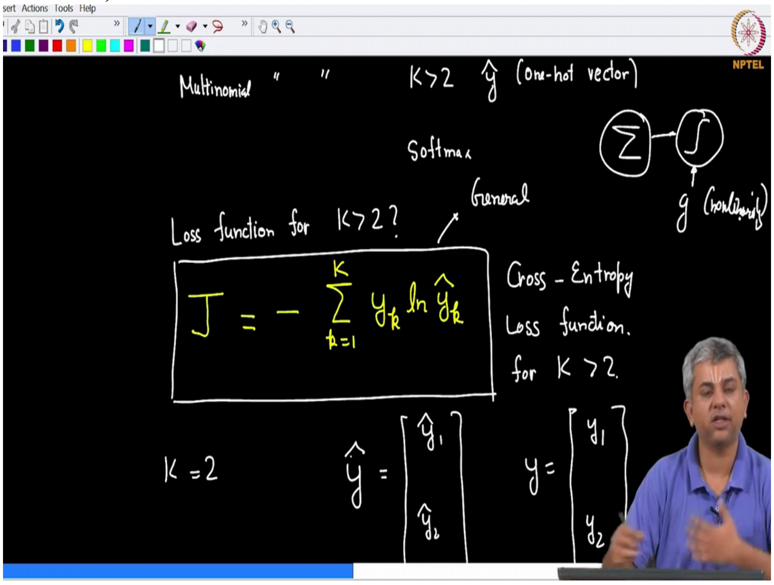


classification loss functions in this form or at least the cross entropy loss functions in this form.

To summarize, so far we have looked at the forward model and the loss function for logistic regression as well as for the multinomial logistic regression.

In both cases, all we have is a linear function followed by a nonlinearity. When you repeat the

(Refer Slide Time: 16:59)



same thing multiple times you essentially get a deep neural network as we will see in the following videos. Thank you.