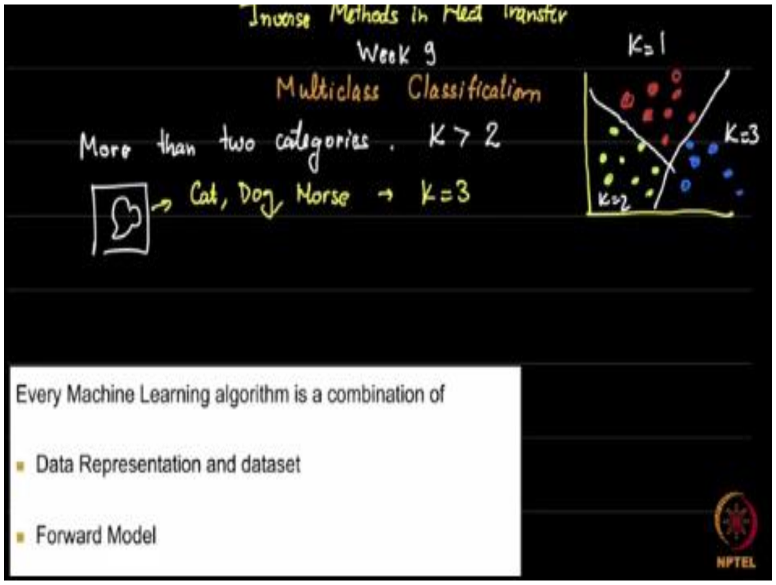
Inverse Methods in Heat Transfer

Prof. Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology, Madras

Lecture - 50
Multiclass Classification

(Refer Slide Time: 00:19)


Inverse Methods in Heat Transfer
Week 9
Multiclass Classification
More than two categories . K > 2
→ Cat, Dog, Horse → K=3
K=1
K=2
K=3
NPTEL

Every Machine Learning algorithm is a combination of

- Data Representation and dataset
- Forward Model

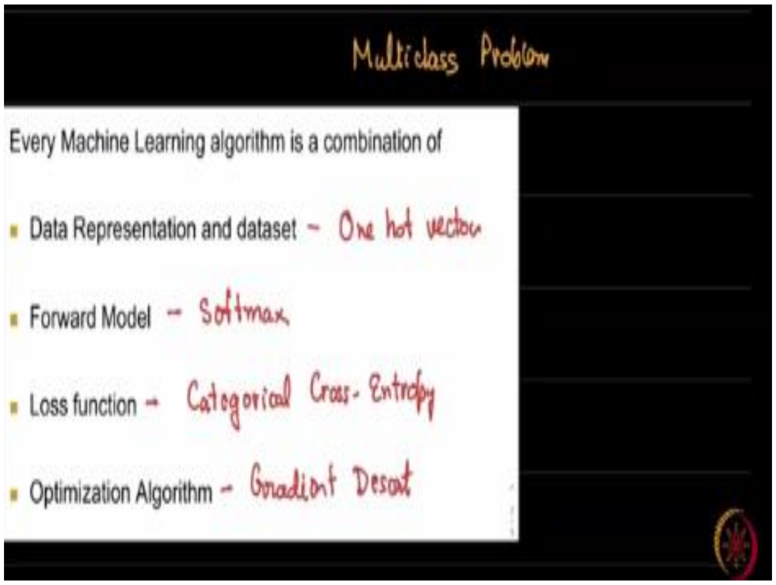
Welcome back. This is week 9 of inverse methods in heat transfer. In the last couple of videos, we saw binary classification which meant that if you have some category, which needs to be split into two, let us say cancer versus no cancer that would be the case of a binary classification problem. Now we are going to extend that here into a multi-class classification case. So, here you have more than two categories that you can split the data into.

So, let us say the number of categories is K so then K is greater than 2. For example, you have an image and this image is of something if there were only two categories typically you will use a binary classification approach. But here if it is let us say a cat or a dog or a horse, then the number of categories is three. Similarly, you could have something like laminar, turbulent versus unknown. So, that could be a possible three-class classification problem.

Or you could have many more cases for example laminar turbulent versus transition. So, this would be the example of k equal to 3 classes. In an image you would see something of the sort in case it is a binary classification problem, in case we are putting all the data at one place. They put cluster in a certain fashion we should make it appear that you have three classes where you want boundaries.

Now in all the problems that we have done so far, I have made a certain implicit assumption and I will deal with this a little bit later. But you would have classes of this sort and you can see that you can now draw boundaries that separate let us say the k equal to 1 from the k equal to 2 and the k equal to 3 cases. Now how do we do this? Here we do not have just a single decision boundary but effectively you have two decision boundaries or in some cases you can even visualize this as if it has three decision boundaries. So, how do we find that out?

(Refer Slide Time: 02:36)



Multiclass Problem

Every Machine Learning algorithm is a combination of

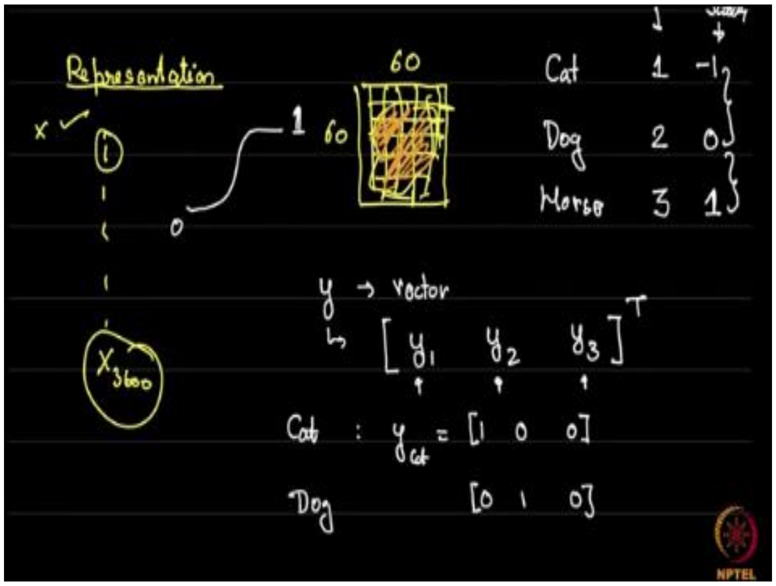
- Data Representation and dataset - One hot vector
- Forward Model - Softmax
- Loss function - Categorical Cross-Entropy
- Optimization Algorithm - Gradient Descent

So, once again we go back to our old idea about machine learning essentially being a combination of these four ideas. Data representation and data set, what the forward model is, what the loss function is and what the optimization algorithm does. So, what is it for the multi-class classification problem. So, in case you have a multi-class problem it turns out that in data set just like last time we did a scalar representation y equal to 0 or one there is something called one hot vector representation that we have to use.

The forward model will be very similar we will have a linear but we will not have sigma at the end we will have something called a soft max at the end. The loss function is what is known as a categorical cross entropy loss function and the optimization algorithm says as gradient descent. Just as before if we have lots of data you can use stochastic gradient descent or mini batch gradient descent otherwise, we are still happy with just batch gradient descent.

So, let us look at these three alone within this video and the rest of the details actually exactly happen to be exactly the same as before. So, first let us start with data representation.

(Refer Slide Time: 04:06)



So, the representation has the following problem. So, let us take this simple example you have an image again I have let us say I have a 60 cross 60 image which means there are effectively 3600 inputs which go here. So, if I write 1 and I unroll it I will have to write till X 3600 here. I put all that together and how do I get that? I simply get it as I have said multiple times before. I can unroll this and just write it out in one single column vector.

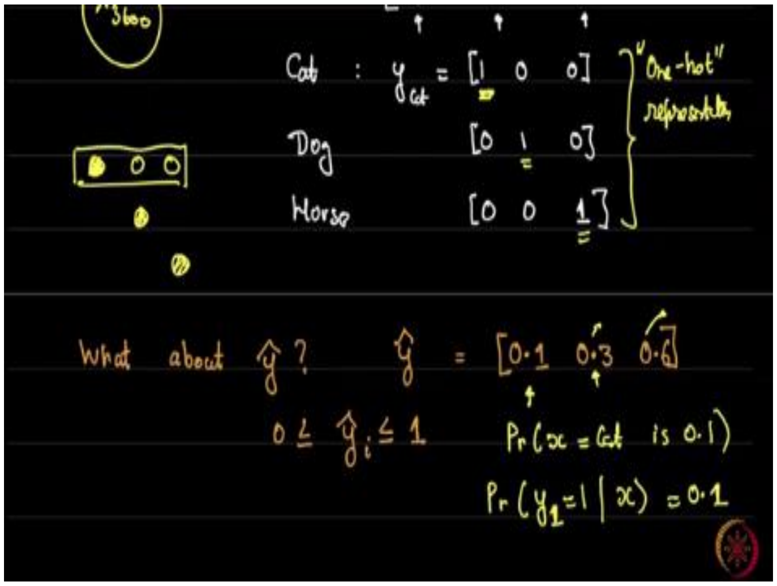
There are other ways of doing it, but we are not discussing that in this case. So, X representation is straightforward again it is as usual. Now comes the representation of Y. Let us say this image which is an image of something, it is already given to you. It is the image of a cat or a dog or a horse and you just need to figure out what it is. Now you have three classes Y is either a cat or it is a dog or it is a horse.

Now how do you represent it? Now suppose one option is I say cat is one, dog is two and horse is three. This is problematic because remember when we had just 0 or 1, we could simply make up a function that went from 0 to 1. But it is very hard to do this between 1 and 3. So, we will need to think of something else. You might think of other possibilities such as - 1, 01 and even these do not work well because you have to map the space between this and the space between this.

And we like the fact that we had a 0 to 1 and there was an interpretation as probability in the binary case. So, people came up with a clever solution. Now instead of this being a scalar that is notice here y is just one single number. This is also a scalar; we start representing y as a vector. So, y is represented as a vector as follows, y has three components I am going to write it as a row just because it is a little bit more spatial and efficient so $[y_1 \quad y_2 \quad y_3]$.

So, this is what it represents. This will say whether it is a cat or not, this will say whether it is a dog or not and this will say that it is a horse or not. Typically for an actual image if it is a cat then y will look like, $y_{\text{cat}} = [1 \quad 0 \quad 0]$ with a dog sorry if it is our dog y will look like, $y_{\text{dog}} = [0 \quad 1 \quad 0]$ and if it is a horse will look like, $y_{\text{horse}} = [0 \quad 0 \quad 1]$.

(Refer Slide Time: 07:05)



3600
Cat : $y_{cat} = [1 \quad 0 \quad 0]$
Dog $[0 \quad 1 \quad 0]$
Horse $[0 \quad 0 \quad 1]$
"One-hot" representation
What about $\hat{y}$?
$\hat{y} = [0.1 \quad 0.3 \quad 0.6]$
$0 \le \hat{y}_i \le 1$
$Pr(x = Cat$ is $0.1)$
$Pr(y_1 = 1 \mid x) = 0.1$

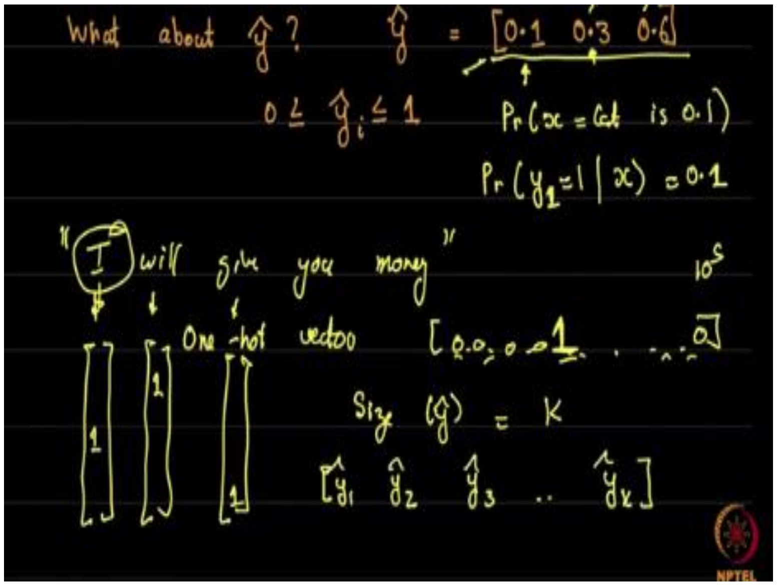
This representation notice is a vector representation and it is called a one hot representation. Whatever we did for logistic regression was a binary representation this one is a one hot representation. Why is it one hot? Only one of these is active at any point. So, you can imagine a switchboard with just three bulbs and one of them will light up. If the first one light up lights up and say cat, the second one lights up, I say dog and if the third one lights up, I say horse.

And only one of them will be lit up at any instant of time. So, this is a simple one-hot representation. but our problem does not end here. We have a problem of finding out but what about $\hat{y}$. Remember $\hat{y}$ could be any number between 0 and 1, when we dealt with binary classification. So, here we can now think of a $\hat{y}$. Let us take an example $\hat{y}$. Let us say we put the constraint that all three numbers all y-hat i have to lie between 0 and 1.

So, if we do that let us take an example, I will say $\hat{y}$ this one is 0.1, this is 0.3 and this one is 0.6 can be interpret this actually turns out there is a very natural interpretation. This says probability that this x is a cat is 0.1. Another way of saying it is probability that y_1 equal to 1 is 0.1. Remember y_1 is just the first component. This one is probability that this is a dog is 0.3 and probability that this is a horse is 0.6.

Since the greatest probability occurs for a horse, we would maybe most probably will classify this as a horse but we will say it is a horse with 60 percent probability. So, this is a very simple representation. Now notice again I am emphasizing this was represented as a vector. Even though there is one quantity it was represented as a vector of size 3 and what happens with words.

(Refer Slide Time: 09:41)


What about $\hat{y}$? $\hat{y} = [0.1 \;\; 0.3 \;\; 0.6]$

$0 \le \hat{y}_i \le 1$ $Pr(x = \text{Cat is } 0.1)$

$Pr(y_1 = 1 \mid x) = 0.1$

"I will give you money" 10^5

One-hot vector $[0 \; 0 \; 0 \; 0 \; 1 \; \ldots \; 0]$

$\text{Size}(\hat{y}) = K$

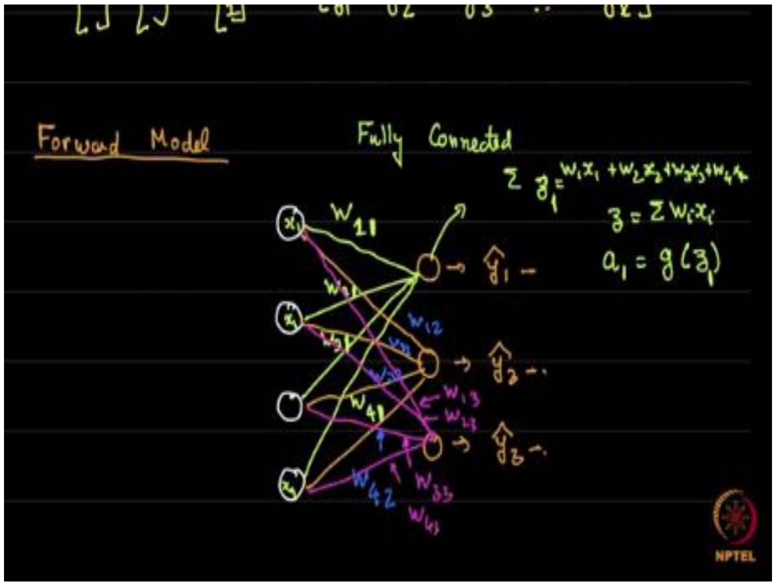
$[\hat{y}_1 \;\; \hat{y}_2 \;\; \hat{y}_3 \;\; \ldots \;\; \hat{y}_K]$

Let us say we want to find out text and we have email and something like I will give you money is some text that is sitting in an email and I want this as a vector. Now the way we do it is each word finds its dictionary place but the entire dictionary is a one hot vector. So, you give a vocabulary or a vector of size. Let us say there are 100000 words in a dictionary for example so 10 power 5 that many circles we have here.

And let us say I is the 58000 word then that alone will be one and the rest of the all the entries within this vector will be zero. So, each word is then represented as a vector of the same size of a hundred thousand size but it will be a one hot vector. So, this will be a sequence of one hot vectors and only one of them will be one. So, this is a very complicated way of doing things but nonetheless it is complicated at least for dictionary words or sentences.

But for images it is fairly straightforward. You just have as many entries as you have classes. So, the size of y-hat is exactly equal to the number of classes so you have y_1, y_2, y_3 up in y_k or $\widehat{y_1}$, $\widehat{y_2}$, up to $\widehat{y_k}$. So, that is it for the representation this is simply a one hot representation.

(Refer Slide Time: 11:27)



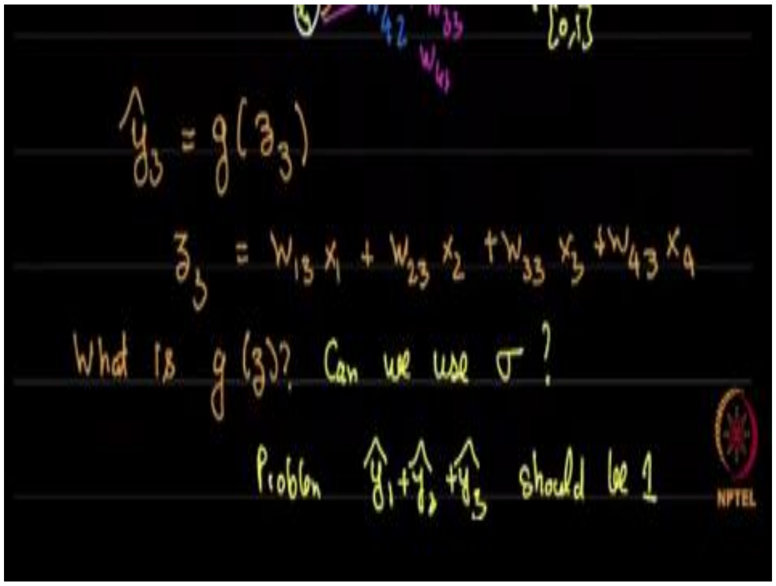
The next question was that of the forward model. So, let us make the forward model again in this case of having three entries. So, for the sake of argument let us say I have an input with four neurons and I have three neurons at the output. From here I want to take at $\widehat{y_1}$ from, here I want to take out $\widehat{y_2}$ from here $\widehat{y_3}$ that is the probability that this is a cat, probability that this is a horse, probability that this is a dog and this is a horse.

Now how do I make my weights and models in this case. So, the way we do it is this, of course, I will connect this, this is what is known as fully connected. So, we will put some weights here we will say w_1, w_2, w_3, w_4 then in this neuron there will be first a summation which basically means $w_1x_1 + w_2x_2$ so let us say this is x_1, x_2 etcetera up till $+w_3x_3 + w_4x_4$. But after that you also want a non-linearity. So, this of course is Z is this linear combination.

So, $z = \sum w_i x_i$ but then the output which I am calling $\hat{y}$ or we typically call a_1 it is the first activation is some non-linearity of z. So, $g(z_1)$, z_1 is the linear output here. Now what about the second case? In the second case I join these. Now here is a problem if I call this w_1, w_2, w_3, w_4 I will have confusion between which w_1, w_2, w_3, w_4 it is so I will call this $w_{11},\ w_{21}, w_{31}, w_{41}$.

I am going to change the rotation within the next week but this is just to make clear that there are four weights these four weights for the first neural. Similarly, another set of four weights for the second neural. Now what can we call them? Instead of calling them w_{11} we call them $w_{12},\ w_{22}$, w_{32}, w_{42}. Similarly, we have the third neural. Now you can see that it gets fairly messy but nonetheless this is really how real neural networks look pictorially. So, this is $w_{13},\ w_{23}, w_{33}, w_{43}$.

(Refer Slide Time: 14:45)


$\hat{y}_3 = g(z_3)$

$z_3 = w_{13}x_1 + w_{23}x_2 + w_{33}x_3 + w_{43}x_4$

What is $g(z)$? Can we use σ?

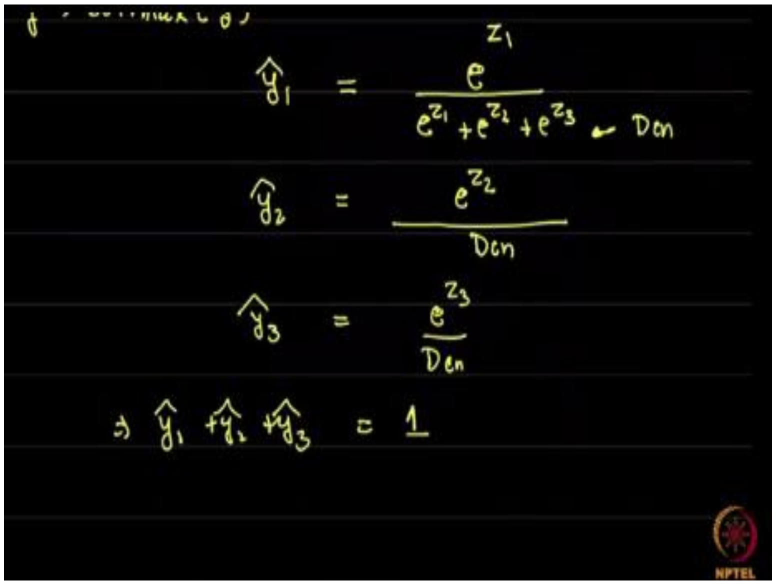
Problem $\hat{y}_1 + \hat{y}_2 + \hat{y}_3$ should be 1

So, suppose I want $\widehat{y_3}$ this quantity here $\widehat{y_3}$ we would say is a non-linear function applied on the linear activation which was z_3. Whereas z_3 itself which comes here is from a linear combination of these four. So, $z_3 = w_{13}x_1 + w_{23}x_2 + w_{33}x_3 + w_{43}x_4$ so that is it. It is a simple model it is of the binary classification or the logistic regression model. All we did really is just add more neurons at the output and just go with that.

However, we still have to figure out what is z or sorry what is this g. Now we might think we can use sigmoid can we use sigmoid. If we use sigmoid the positive thing is $\widehat{y_1}$ would lie between 0 1, $\widehat{y_2}$ would also lie between 0, 1 so would $\widehat{y_3}$. But there is a problem. The problem is $\widehat{y_1} + \widehat{y_2} + \widehat{y_3}$ should always be one. Why do we want it to be one? Because we want to interpret it as a probability.

Notice this here if the probability that the cat is 0.1 and the probability that a dog is it is 0.3, the probability that it is a horse should be 0.6 because the sum of these three should add up to one because these are the only three possibilities that exist. On the other hand, if it is just sigmoid this could be 0.5, this could be 0.5, this could be 0.5 and we would no longer be able to interpret it the way we want.

(Refer Slide Time: 16:53)



$$\hat{y}_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}} \leftarrow \text{Den}$$

$$\hat{y}_2 = \frac{e^{z_2}}{\text{Den}}$$

$$\hat{y}_3 = \frac{e^{z_3}}{\text{Den}}$$

$$\Rightarrow \hat{y}_1 + \hat{y}_2 + \hat{y}_3 = 1$$

So, we create a new function called soft max. So, soft max works like this so suppose I want $\widehat{y_1}$ I will say it is,

$$\hat{y_1} = \frac{e^{z1}}{e^{z1} + e^{z2} + e^{z3}}$$

$\widehat{y_2}$ is,

$$\widehat{y_2} = \frac{e^{z2}}{e^{z1} + e^{z2} + e^{z3}}$$

and $\widehat{y_3}$ is,

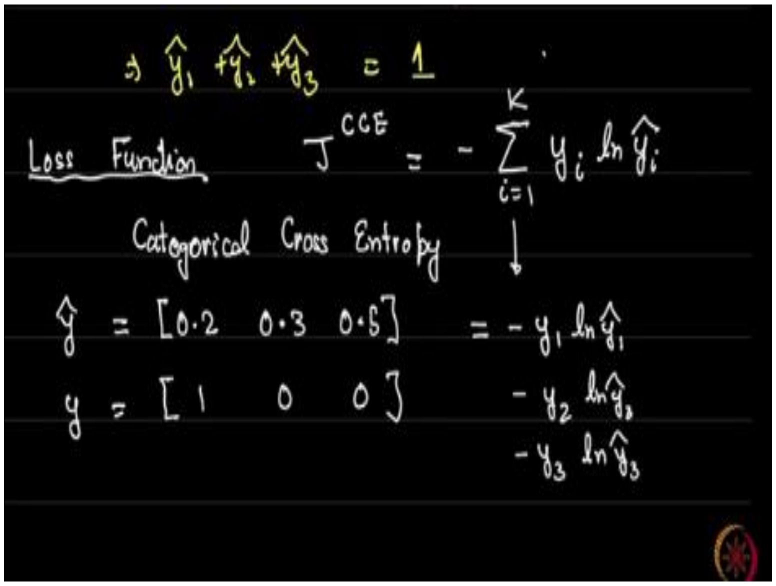
$$\widehat{y_3} = \frac{e^{z3}}{e^{z1} + e^{z2} + e^{z3}}$$

And then obviously if I sum the 3, I simply get one. This is the advantage of creating this soft max function.

It has one other nice advantage which I will talk about the next week. But the advantage of the soft max function is that we are always ensure that we can interpret the output as if it is a probability. So, what we have looked at up until now is that the data is represented as a one hot vector and the forward model is exactly the same as the logistic regression model except, we use soft max instead of sigmoid.

Now I must point out here and I will do so once more when we come to the next week that this soft max only needs to be used at the end, when you want to find out just the output. If suppose there is a calculation in the middle you can actually use sigmoid there but anyway these are greater details when we come to neural networks and I will discuss that in the next week.

(Refer Slide Time: 18:44)


$\Rightarrow \hat{y}_1 + \hat{y}_2 + \hat{y}_3 = 1$
Loss Function $J^{CCE} = -\sum_{i=1}^{K} y_i \ln \hat{y}_i$
Categorical Cross Entropy
$\hat{y} = [0.2 \quad 0.3 \quad 0.5]$
$y = [1 \quad 0 \quad 0]$
$= -y_1 \ln \hat{y}_1 - y_2 \ln \hat{y}_2 - y_3 \ln \hat{y}_3$

The third part is the question of the loss function. Now what is the loss function here? Now I will write the loss function but it should also satisfy the same property that we wanted for our binary cross entropy loss function. So, I am going to call this J^{CCE} standing for categorical cross entropy loss function. And the purpose of this loss function is exactly the same as the previous one and strangely enough it will look simpler than a binary cross entropy loss function.

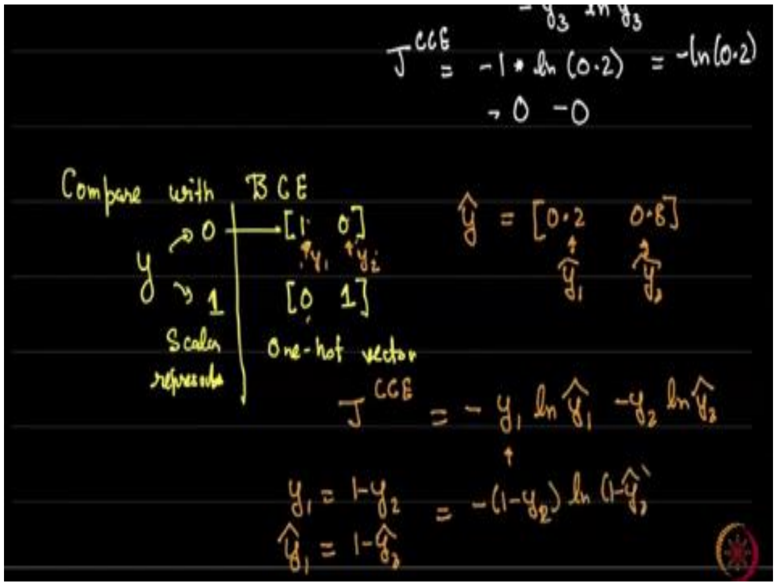
It is like this it is $-\sum_{i=1}^{k} y_i \, ln\widehat{y_i}$. This looks really simply, all it is let us say I have some examples like again I will say 0, 0.1 actually let me make this a little bit better. Let us say,

$$\hat{y} = [0.2 \quad 0.3 \quad 0.5]$$
$$y = [1 \quad 0 \quad 0]$$

Let us say we actually have a misclassification. So, we have misclassification in that the one which seems to be most probable due to our model is 0.5 but the actual truth is that it is a cat then we would calculate this in the following way.

This will be $-y_1 ln\widehat{y_1} - y_2 ln\widehat{y_2} - y_3 ln\widehat{y_3}$. Now notice y_1 is 1 so this one will be -1 * ln(0.2)-0-0. So, this is simply -ln(0.2) so that is the loss function in this case.

(Refer Slide Time: 21:07)



$- \hat{y}_3 \ln \hat{y}_3$

$J^{CCE} = -1 * \ln(0.2) = -\ln(0.2)$

$\to 0 \quad -0$

Compare with BCE

$y \to 0 \to [1 \quad 0]$, y_1, y_2

$y \to 1 \quad [0 \quad 1]$

Scalar representation | One-hot vector

$\hat{y} = [0.2 \quad 0.8]$, $\hat{y}_1$, $\hat{y}_2$

$J^{CCE} = -y_1 \ln \hat{y}_1 - y_2 \ln \hat{y}_2$

$y_1 = 1 - y_2 \qquad = -(1-y_2) \ln (1-\hat{y}_2)$

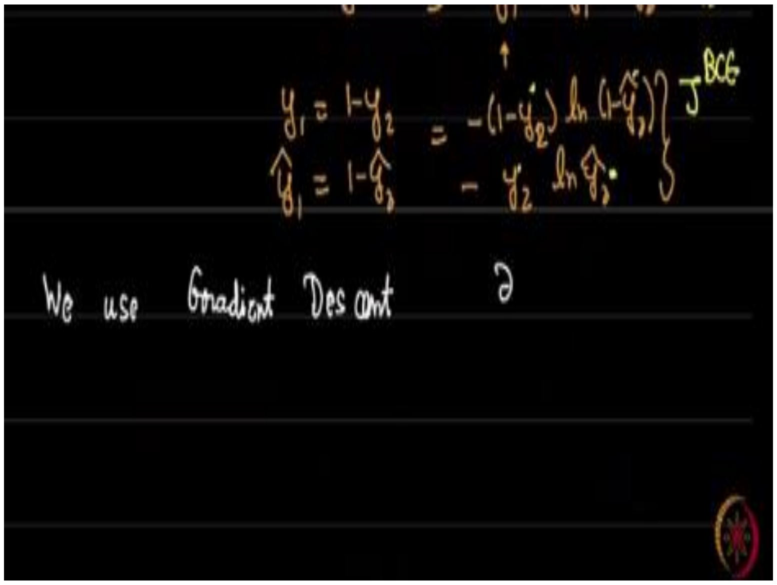
$\hat{y}_1 = 1 - \hat{y}_2$

Now I am going to argue that even though it looks less complicated it is almost exactly the same as the binary cross entropy. So, we can compare this with the binary cross entropy case. Now remember in the binary cross entropy case we had just two classes y was either 0 or 1. Now this was the scalar representation. Now instead if we represent this also as a one hot vector now instead of calling it class 0, I would have called it 1, 0 because there are two classes and it is the first class.

And instead of calling it one I would have called it 0, 1. So, let us say this is y_1 the first component of y, this is y_2. So, in the case of class 0 or in the first class I would say y_1 is 1 and y_2 is 0 the second case y_1 is 0 and y_2 is 1. Similarly, if I look at $\hat{y}$, $\hat{y}$ would look something like 0.2 and 0.8 so this one would be $\widehat{y_1}$ and $\widehat{y_2}$. Putting this together if I actually calculate the cross entropy loss function here this would be $-y_1 ln\widehat{y_1} - y_2 ln\widehat{y_2}$.

Now we know that $y_1 = 1 - y_2$ because there are only two classes. So, if y_1 is 1 then y_2 has to be 0, if y_1 is 0 then y_2 has to be 1. Similarly, we also know that $\widehat{y_1} = 1 - \widehat{y_2}$.

(Refer Slide Time: 23:13)


$y_1 = 1-y_2$
$\hat{y}_1 = 1-\hat{y}_2$
$= -(1-y_2)\ln(1-\hat{y}_2) - y_2 \ln \hat{y}_2$
J^{BCG}
We use Gradient Descent
∂

So, substituting this, this will give us $-(1-y_2)ln(1-\widehat{y_2})-y_2 ln\widehat{y_2}$ which is exactly the same as J binary cross entropy. So, you see the terms $yln\hat{y}$ and $(1-y)ln(1-\hat{y})$. So, what this shows is that what I have is the categorical cross entropy loss function is exactly the same as the binary cross entropy loss function except it has been extended to higher number of classes.

What do we do finally about the final portion of the puzzle which is the optimization algorithm, we simply use again we use gradient descent. Now it might seem like all this is fine but how do we do an actual practical case. So, we will come to that next week but all I wish to point out is it is actually fairly straightforward. We do not do anything different. So, if I go back to this image here, so let us say this has about 30 points here.

I just collect these 30 points I find the x location, the y location or x_1 and x_2, x_1x_2 class represented as a binary as a one hot vector. So, this is the way you will write it. So, let us say the first point is 0.3, 0.4 and this belongs to class 1 so this will be 1 0 0. The second point is let us say 0.1, 0.7 and this belongs to let us say 0 1 0. So, you will note this down and you have a model your model is like this.

It is a more complicated model but you simply can write down $\widehat{y_1}$ in terms of all this, $\widehat{y_2}$ in terms of all this and $\widehat{y_3}$ in terms of all this that is your forward model. Then to do back propagation you have to do $w = w\text{-}\alpha\frac{\partial J}{\partial w}$. Now how do I calculate del J del w this, I will talk about next week I will just give you the expression directly in the next week. But keep on iterating and finally you will get the weights.

So, that magically you will actually get a nice classification boundary. So, we will look at a couple of toy problems in the next week for both the logistic regression as well as well most probably not multi-class regression. But for logistic regression definitely we look at an example in the next week and we will move on to neural networks. But before we want move on to neural networks that would be a question. why exactly do, we even need neural networks and that I will answer in the next video. Thank you.