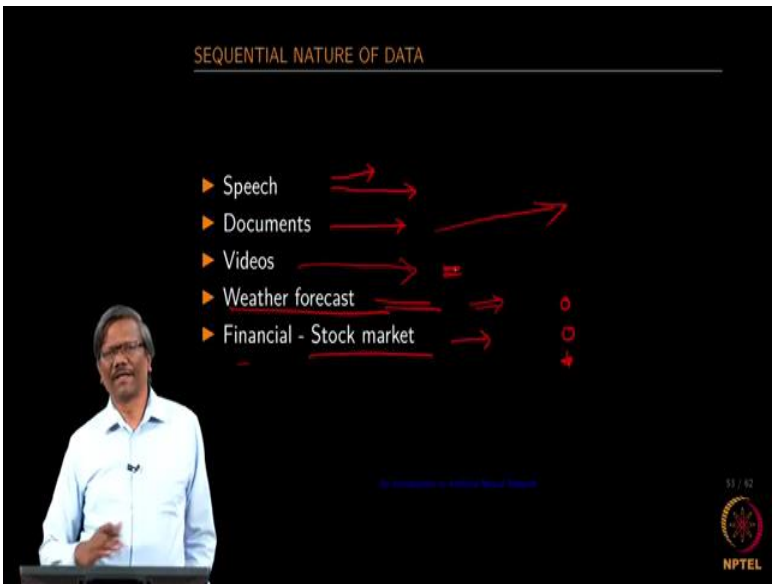
Applied Natural Language Processing
Prof. Ramaseshan Ramachandran
Department of Computer Science and Engineering
Chennai Mathematical Institute

Lecture - 38
Feedforward and Backpropagation neural network

(Refer Slide Time: 00:15)



So, we are going to continue this session into the next stage where we want to solve different kinds of problems. Earlier we were only talking about linearly separable problems and then how we can use perceptron to solve those kinds of problems, but in the real situation we get into various data sets you know speech. So, let us first talk about these speech, the one that I am talking right now is recorded and then I want to be able to recognize what I am talking. For that first of all I need to find out where is my word boundary in all of those and then take the signal values related to that of word and then process it in some way. And then finally, say that what I am saying is this let us say speech is one word ok.

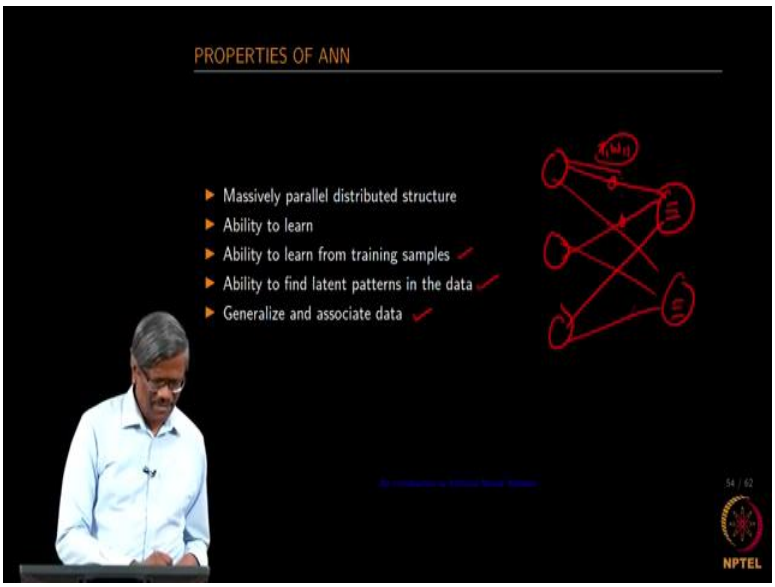
In the same fashion you know you create various word boundaries and then identify each word. And finally, take it to the application where you would like to say for example, the Alexa or the Google home or Siri or whatever you know those applications are actually listening to your speech and then figuring out what you are really talking with respect to the string of words. And finally, convert this into the sentence, maybe you search the

web or you know tries to figure out what exactly you are asking for. And finally, list those item that you are really looking for right in some fashion.

So, this is one application where the data is coming in a sequential fashion. So, I should be able to take these as the input and finally, do something with that. And, then documents we spoke about this several time you know, we been talking about tokenizing the documents and then using the word as one element for us to process, but that is not going to be the case all the time right. So, we need to be able to understand a sentence, for you to understand a sentence we need to have the string of words. And, then using those string of words you should be able to understand that as a sentence and then what that sentence means and then finally, provide what that actually is asking.

For example, if that sentence is a question asking the system to find out a various documents related to natural language processing right; related to natural language processing. So, the data is coming in as sequence of words and this contains both audio, video and so on. So, both speech and the images are coming in as a sequence. So, we should be able to process it and then provide the necessary output. It is a weather forecast again based on the historical information; it is not an isolated incident. The same with the stock market deals with at least several different variables based on which you predict what is going to be the stock price as of today for a given scrip ok.

(Refer Slide Time: 04:00)



So, how do you process these? Ok. So, before that you know we also need to mention a few things about the artificial neural networks. As we had seen earlier right it is possible to process these values for example, the computation of this the x 1 W 1 1. So, these are all independent of each one of those. So, you can see that there is a massive parallelism in this. So, it is possible to utilize the parallel processing to really compute all these values independently and then later combine them here right.

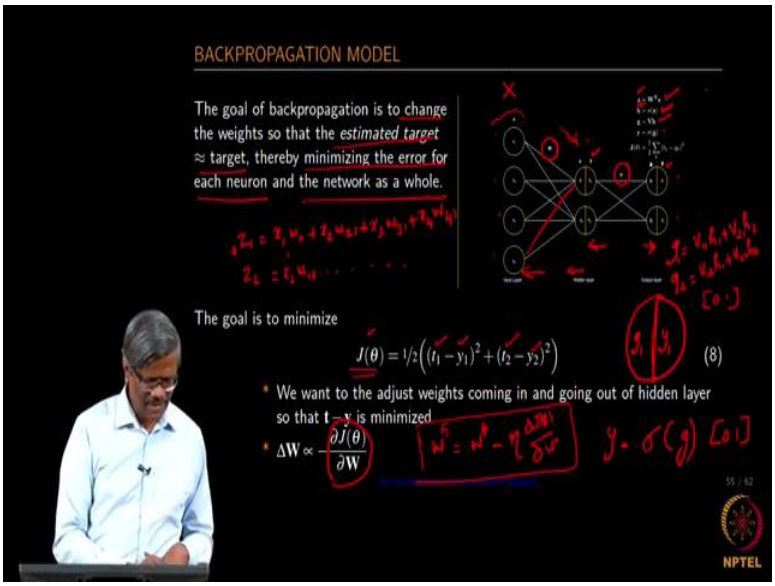
And, then we also had seen that it is possible to learn from the training samples. We were able to really figure out that even though we fed about 5,000, 6,000 sentiment words, the weights are adjusted in such a way that it really generalized the weights for all the input sentiment words right. It is not related to just one word, the weights are not related to just one word; it is related to all words that we had input. So that means, it has generalized the weight, it also figures out the latent patterns in the data.

We will talk about this and finally, it generalizes and associates the data sets in some fashion like I mentioned earlier ok; going back to this. So, in order for us to identify those patterns we require the neural net. For example so, why this scrip is down today, is there any similar situation that I am able to find out from the historical information?

Since, the number of parameters are huge in order for you to find that out, we want to use the neural network which really is good in terms of identifying those patterns which are inherent in the data set. The same fashion I want to find out whether it is going to rain today or not because, of these kinds of input parameter that I am having today with respect to the temperature, humidity all that and the pressure and so on. Is there any similar situation that existed earlier that caused rain? Then we can say with some probability that today it may rain and so on right.

So, for that again we require lots of data set and system, if it is trained using the neural network it should be able to figure out those patterns in the historical information. And, then give you with some value that you can use it to predict, same with the videos you know the data is sequential in nature ok. So, for all of this we need a neural net of different type than what we had seen in the perceptron case ok. So, we will talk about those one by one in these subsequent lectures.

(Refer Slide Time: 07:34)



So, first we will talk about artificial neural network, where we have a feed forward model and back propagation model. The feed forward is something that we had seen earlier in the perceptron as well. When the inputs are given you keep feeding the data into the hidden layer and then finally, compute the values in the output layer right. So, that is the feed forward model. Back propagation is the error is found so, I need to adjust the values of V and W. How do I do that? So, what is the mechanism through which I want to adjust these weights? Ok. Is explain I equation

$$J(\theta) = \frac{1}{2}\left((t_1 - y_1)^2 + (t_2 - y_2)^2\right.$$

So, the idea of this one is to really chain the weight so, that the estimated target is close to the target value and thereby minimizing error for the for each neuron and the network as a whole; so, that is the idea. So, this is very similar to what we saw in the XOR model, but we are extending it a little more ok. The size would dictate the complexity of the network. This is a very simple network with 4 input units, 2 hidden units and 2 output units ok. So, here we want to minimize the error using this cost function.

So, we before that let me explain the network part; I hope you can see this clearly. We have X here and then each element in this input layer has a neuron that takes the input value and then passes it to the hidden through the weights. We have X_1, X_2, X_3 and X_4

and then we have a hidden unit; there are 2 hidden units and there are 2 output units. I have actually divided the hidden unit into 2 and the output units into 2 parts so like these.

$$z_1 = x_1 w_{11} + x_2 w_{21} + b_1$$

$$z_2 = x_2 w_{12} + x_2 w_{22} + b_1$$

So, the reason for that is I am going to be computing the net values of either neuron coming from all the input units right through the weights ok. So, the net is computed using W T X. So, X is your input vector, W is your weight matrix that is connecting the input and the hidden units, to get the net value which I am calling it as Z or Z which is nothing, but the dot product of the transpose of the weight vector and the input vector. It is its nothing, but if you expand this form it is x_1 W_{11} plus x_2 W_{21} plus x_3 sorry this is 4 1 right.

So, we can write it for the this is for h_1 rather z ok. So, the whole thing is written in a compact fashion like this then I have to write it for a z 2. What is this? Ok. Like x 2 W_{12} and so on right and then we have another partition there; once the net values are computed for each of the hidden units we use a sigmoidal function to squash the values between 0 and 1. So, we use a sigma this one is nothing, but a sigmoidal function which squashes the value of a z_1 in between 0 and 1.

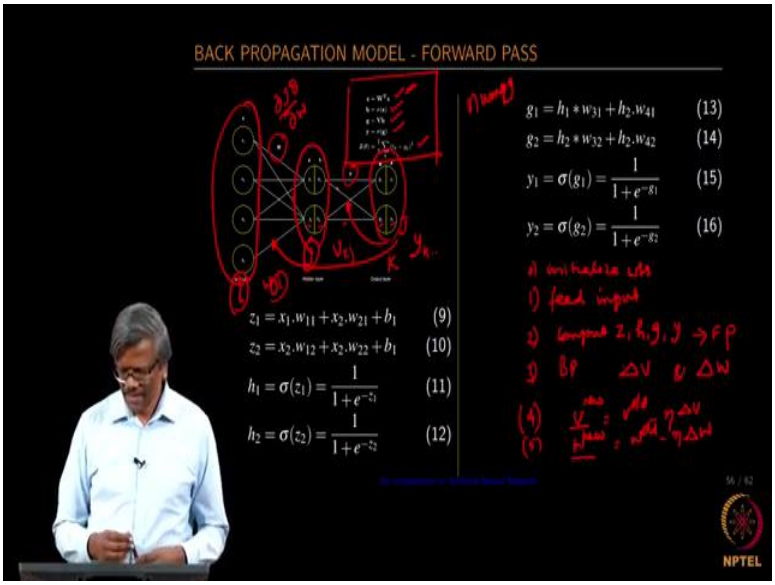
So, this computes the net and this computes the actual value that you are going to be using to compute the output right. So, we call that as h here and then again in the output layer I am going to be using the same partition. So, we have a g 1 and a y 1 there and then there is a g_2 and y_2 so, g 1 is computed using V and h. So, again V is a matrix; so, you have g equal to I am sorry V_{21} into h_2 correct this g 1. So, g 2 again you compute using V_1 to h_1 plus V 2 to h_2. So, g is computed using this and then the output again we are is going to be using a sigmoidal function so, y equal to sigma of g ok. So, again and the values of y is between 0 and 1, we are squashing the values between 0 and 1 again here.

So, there are 2 sigma that we are using one here and then one here. Since, this is going to be your supervised learning where we know what is the expected output; so, we are

going to be computing the cost function in this way ok. So, which is the t 1 is known which is the target and then y_1 is computed and then t_2 is known and then y_2 is computed ok. So, for both y 1 and y 2 we compute the cost, this cost function depends on V and W. Now, how do I change the weight? It is given by the equation here, the equation is the rate of change of cost will give you the change with which you have to update your weight ok.

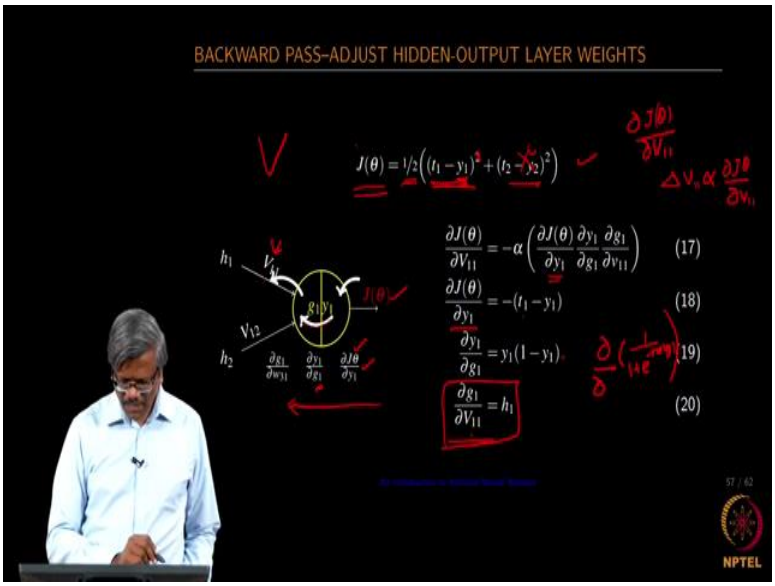
So, if you write this W let us say iteration 5 equal to so, this is how you update the weights every time ok. So, when you do the updation these weights changes on both between the hidden layer and the output layer, the input layer and the hidden layer. So, we for the simplicity sake we call this W as input weights, this V as output weights.

(Refer Slide Time: 15:59)



I think I did this earlier and this slide actually has a cleaner representation of what I wrote in the previous one ok; let us move on to the next one.

(Refer Slide Time: 16:19)



So, how do I compute the change of values right for V 1? So, we know that there is an error that is available and then I need to make some changes so, that the error becomes smaller. So, I need to start back propagating the error from the output to the output weights. So, how do I find this? Ok, is a very simple way to do that is to use a partial differential equations. And, I use a chain rule for identifying the weight changes; if now what we need to do is to find the rate of change or find the value by which we want to change a V 1 we need to start from the right.

So, we know this part, it is easy to compute because all those things are known right. So, you know the input values, initially we have the weights that are assigned some random values. So, based on that user is the net input is calculated for the hidden units and the net output for the hidden units are calculated using a sigmoidal. And, then we have again of matrix V which are the values are initialized using some random values. Now, output of the hidden units are known, now the values of V are also known.

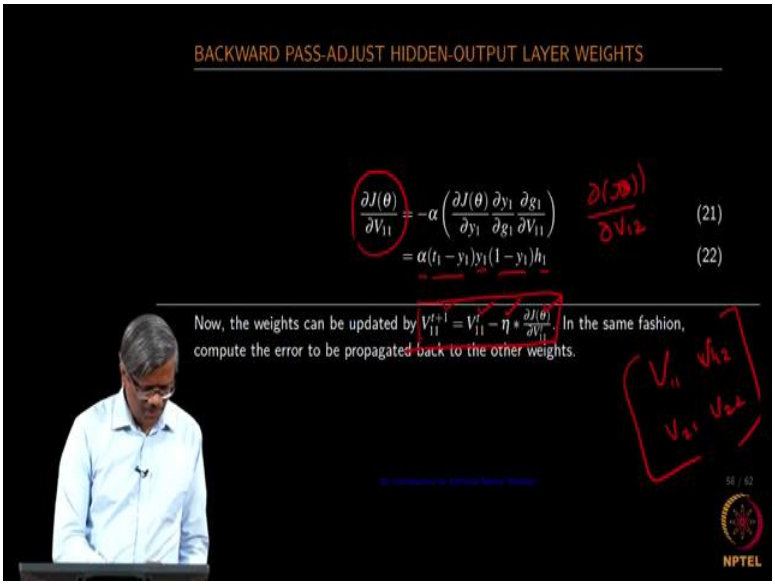
So, do the dot product you get the net output values that is equal to g, once you have the net output values you run it through your sigmoidal function. So, that the values are between 0 and 1 and now y is known that once you had run it through the sigmoidal the values of the outputs are called y right. So, since we know what is the actual target value because, this is a supervised model we can compute the error between those two output units. And, then say that there is a an error that we have received with respect to the

target that I am inputting. So, keep that change and then propagate it back into the network.

So, that the weights are adjusted in such a way that when I come back next time the J value is reduced ok. So, that is the idea correct as we had mentioned in the previous slide right. So, we need to minimize the error for each neuron and at the end we need to minimize the value for the network as a whole right; so, that is the idea here ok. So, since we know this part we can go back right. So, I can find out with respect to y 1 what is the change right, we know the value of y 1 right which we have computed using the sigmoid.

So, with respect to that what is the change that I can compute? The idea first is to find what is the rate of change rather what is the change with respect to V_{11}. So, we are going to be adjusting this weight right, for me to adjust the weight I have to come from the left to the right, I know this I can compute this. So, how do I compute this? If you take a partial derivative of this so, we have y_1 here only in this space and this will not be used ok. The half is used to cancel this part when you take the derivative and finally, what you have is $t_1 - y_1$ right. So, it is easy to compute is not it? And, this alpha is coming because this $\Delta w \propto \frac{\partial J(\theta)}{\partial w}$

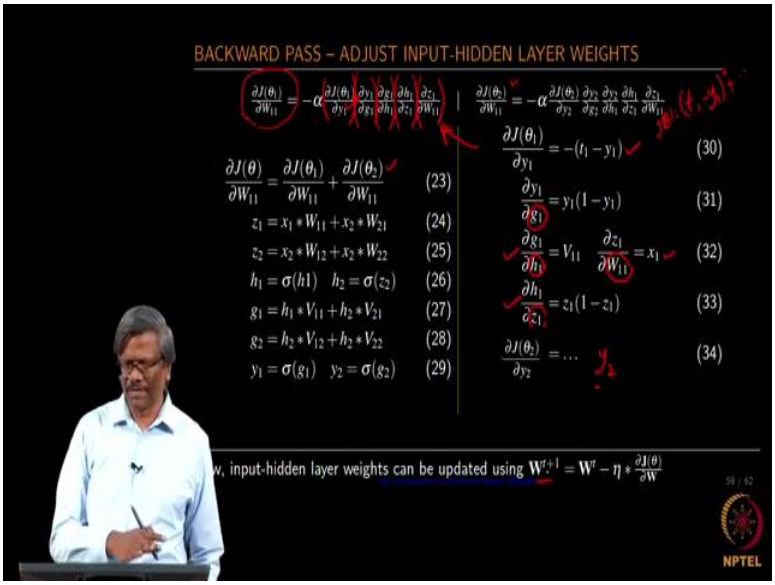
(Refer Slide Time: 22:32)



$$\frac{\partial J(\theta)}{\partial v_{11}} = -\propto \left[\frac{\partial J(\theta)}{\partial y_1}\ \frac{\partial y_1}{\partial g_1}\ \frac{\partial g_1}{\partial v_{11}}\right]$$

So, we know this, we know the eta value, we know the old value of V and then the new value is computed using this. So, this is the error due to the back propagation and the weights are adjusted. So, how many ever V you have right; so, in this case we have only about 4 values right 1 1 V_{11} V_{12} V_{21} V_{22} right. So, we are going to be adjusting all the 4 values here ok.

(Refer Slide Time: 23:43)



The next one is a little bit longer, it is not very complex you know by looking at the number of equations here do not get overwhelmed, it is pretty simple in fact. If you use the chain rule to compute the values and then if each one of those derivatives, we just have to figure out and those equations are known all the 5 question that we have here right. These equations you can make use of ok. So, do not be overwhelm initially you know it might look a very difficult part in the neural network, but it is not it is very simple ok.

So, what all you need to know is what is the chain rule, chain rule is pretty simple ok. So, once we know how to represent this using a chain rule and then you can take each one of these and then find the values. And, then for each one of those you will always have some related equation that you will find in the space here. So, what is I suggest is first try to write these equations. So, let me again erase this, this draw a network and then appropriately named those weight values like I have done and then start doing this ok; is it its very simple right. It is easy to compute all of this right.

So, these equations are very easy to write. So, once these equations are known then its only up to the chain rule to really break it up into multiple small partial derivatives. And, then for each partial derivative you get the values ok, let us let us look at this second one. Now, we are going to be going from the J is here and then J 2 this. So, we have done this, now we have to get back here. So, for us to cross this we know how y was computed, how g was computed and then we know V. We know how h 2 was computed, how these are two values are computed and we know the initial values of W and so on and we have these equations with us ok.

So, it is bit longer, but again I am emphasizing that it is not very complex. So, in this case we are going to be updating the weights, the input weights. So, input weights have to pass through rather. So, in order for us to calculate the or find the change in W_{11}, we require two right because there are two outputs that are generated. And, then the contribution of those two outputs are going to play a role in updating the W_{11} ok. So, I am just calling it as dou J theta 1 and dou J theta 2 ok.

So, let us first find for $\frac{\partial J(\theta)}{\partial j_{11}}$ and it will be very same for this as well ok. So, let me break this into multiple small partial derivatives ok. So, we start with ok; so, my related equation is here the again I will remove all this. So, I need to start with $\frac{\partial J(\theta_1)}{\partial y_1}$because there is a change of this is going to have some impact on the weight as well correct. So, I know that it is possible to compute $\frac{\partial J(\theta_1)}{\partial y_1}$ because we know the right.

So, we know this, we have computed this earlier and then we also had computed this one earlier because there is a rate of change of rather there is a change with respect to g_1 for y. So, we have this, there are two that we have to compute here; one is with respect to h_1 and then with respect to z_1 and then finally, with respect to W 1. So, if you look at the partial derivatives now here starting from the last one right and then we have $\frac{\partial y_1}{\partial g_1}$ correct and then $\frac{\partial g_1}{\partial h_1}$ and $\frac{\partial z_1}{\partial w_1}$

So, that actually leads you $\frac{\partial J(\theta)}{\partial w}$ right. So, this is easy to compute, the second one easy to compute, the third is easy by looking at the equation you can quickly figure this out and so on. Now, go through this a few times you know it is as I say keep saying this is not very complex. The total number of the equation may be overwhelming, but it is pretty simple. Your training process is going to be feed one input, compute z, h, g, y right. So, maybe the 0th step is initialize weights; so, this one is your forward pass correct.

So, you can write another function for the backward pass or the back propagation which will compute ΔV and ΔV. And, then in the training process what you do? Update these values so, the training is done until there is no more change you will find four V nu and W nu and then you stop. Or, you stop at certain number of iterations or if the values of dou J theta rather the delta theta is very small ok. So, that is how simple it is and also you have for one programming advice here is do not use this standard model for doing this.

You know instead of you writing all the loops for the i and J of the input weights and then maybe k and the l of the V here and so on. You use this numpy where you can write your instruction just as you see in this equation so it is so simple. You do not have to really worry about the looping's and then indexing problems and all that. One more thing I also want to mentioned here is if you refer to certain books or papers, they will be using various indices you get yourself familiarized with that ok.

So, normally the what they do is they use i for specifying the index in the input layer and then for J they use it for the hidden layer and then they use k for the output layer. So, that you know the weights here if they represent it will be J i and then here it would be sorry it should be V kj and then here it going to be some let us say y k and so on ok. So, we familiarize with you know what is the kind of index they are using otherwise it will confuse you. So, make sure that you understand the index part of that first when you have networks of this type and then see how they are actually changing the weights and then how they are using the indexes and so on.

So, I guess you have understood this part very well, this is very fundamental to what you are going to be doing in the next few lectures ok. This is only an extension of these the back propagation, the derivatives of the values for W and then how do I use the chain rule to find out and so on. So, instead of one hidden layer there could be multiple hidden layers too. So, you should be very thorough with the simplest once first. Once you are

very thorough with this maybe you can look at more complex networks and then understand them.