

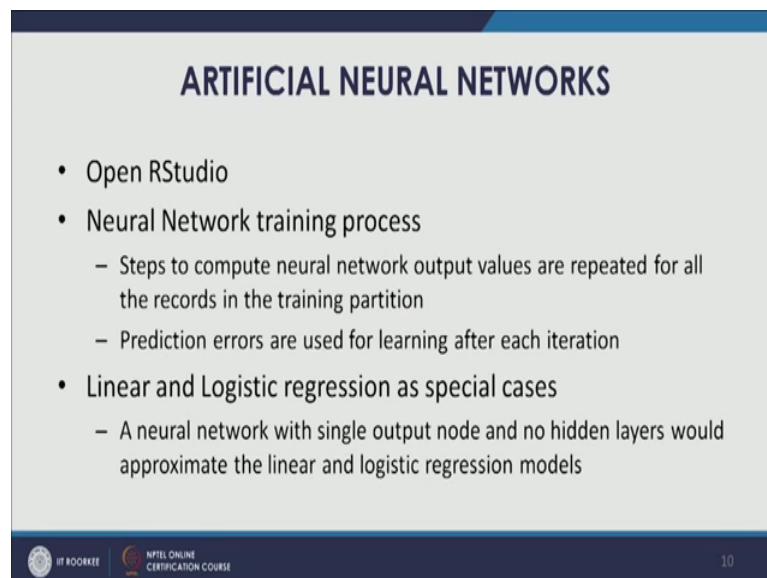
Business Analytics & Data Mining Modeling Using R
Dr. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology, Roorkee

Lecture - 55
Artificial Neural Network-Part III

Welcome to the course Business Analytics and Data Mining Modeling Using R. So, in previous lecture previous few lectures; we have been discussing artificial neural networks. So, we have been able to cover the background, we also did a small exercise, we understood the architecture different layers, we also we also have gone through a few more details related to input layer computations that, we are required to perform in input layers or hidden layers and output layers.

We also understood some of the expression the computations, the transfer function all those things we have gone through the bias values weights connection weights. So, all that discussion we have completed.

(Refer Slide Time: 01:04)



ARTIFICIAL NEURAL NETWORKS

- Open RStudio
- Neural Network training process
 - Steps to compute neural network output values are repeated for all the records in the training partition
 - Prediction errors are used for learning after each iteration
- Linear and Logistic regression as special cases
 - A neural network with single output node and no hidden layers would approximate the linear and logistic regression models

IT ROORKEE NPTEL ONLINE CERTIFICATION COURSE 10



We also discussed how linear regression and logistic regression Can be considered as special cases of neural network.

(Refer Slide Time: 01:09)

ARTIFICIAL NEURAL NETWORKS

- Linear and Logistic regression as special cases
 - If a linear transfer function ($g(x) = bx$) is used, output would be

$$y = \theta + \sum_{i=1}^p w_i x_i$$
 - A formulation equivalent to multiple linear regression equation
 - However, estimation method (least squares) is different from neural network (back propagation)




IIT ROORKEE

NPTEL ONLINE
CERTIFICATION COURSE
11

(Refer Slide Time: 01:13)

ARTIFICIAL NEURAL NETWORKS

- Linear and Logistic regression as special cases
 - If a logistic transfer function ($g(x) = 1/(1+e^{-bx})$) is used, output would be

$$P(y = 1) = \frac{1}{1 + e^{\theta + \sum_{i=1}^p w_i x_i}}$$
 - A formulation equivalent to logistic regression equation
 - However, estimation method (maximum-likelihood method) is different from neural network (back propagation)


IIT ROORKEE

NPTEL ONLINE
CERTIFICATION COURSE
12

(Refer Slide Time: 01:16)

ARTIFICIAL NEURAL NETWORKS

- Normalization

- Scale of [0,1] is typically recommended for neural network models for performance purposes

- For numeric variables,

$$V_{norm} = \frac{V - \min(V)}{\max(V) - \min(V)}$$



Now, let us move forward. So, we also talked about in previous lecture the normalization. So, we talked about that scale of 0, 1 is typically recommended for a neural network models for performance purposes, neural network models converge quite quickly, when the scale is this is scale is used 0 to 1 scale is used.

And the performance is also improved. How, so how this is kind of normalization can be achieved. So, we discussed this particular expression formula V_{norm} . So, any normalized variable in this to bring it to 0 to 1 scale; So, we can sub in the numerator we can subtract it is value with the minimum value of that particular variable and then divided by difference of maximum and minimum value for that particular variable.

(Refer Slide Time: 02:07)

ARTIFICIAL NEURAL NETWORKS

- Normalization

- Binary variables (categorical variables with two classes)

- Create dummy variables: set of values {0, 1}

- Nominal variables with m (>2) classes

- Create m-1 dummy variables: set of values {0, 1}

- Ordinal variables with m (>2) classes

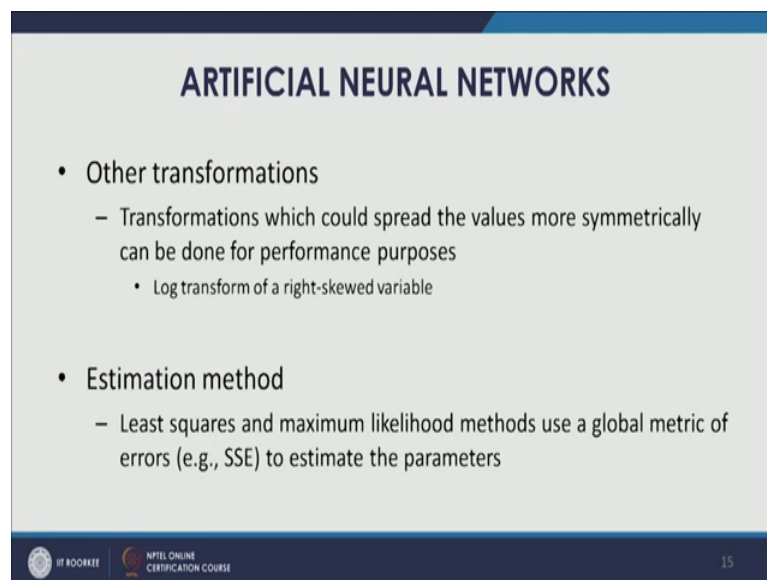
- Map the values to the set {0, 1/(m-1), 2/(m-1), ..., (m-2)/(m-1), 1}



So, that will give us the values in 0 to 1 scale. We also discussed that binary variable you know they are not much to we are not required to do much, because they would already be 0 to 1, because we would typically be used dummy variables. So, they are anyway they are 0; they take two values 0 and 1, nominal variables also no such problem, we will have $m - 1$ dummy variables.

So, they will also have value 0 or 1. So, that would be also in the same range, ordinal variables we talked about this particular mapping right 0 or 1 divided by $m - 1$, 2 divided by $m - 1$, and up to $m - 2$, $m - 1$ and $m - 2$ divided by $m - 1$ and then 1. So, this is just mapping and we would be required to you know; so this is mapping can then be used for ordinal variables.

(Refer Slide Time: 03:05)



ARTIFICIAL NEURAL NETWORKS

- Other transformations
 - Transformations which could spread the values more symmetrically can be done for performance purposes
 - Log transform of a right-skewed variable
- Estimation method
 - Least squares and maximum likelihood methods use a global metric of errors (e.g., SSE) to estimate the parameters

IT ROOKIE NPTEL ONLINE CERTIFICATION COURSE 15

Now, there are a few other transformations that can help. For example, if we are if we have a right skewed variable, then probably log transform that can be done and that can help us our neural network modelling. So, as you can see in the first point that transformation, which could spread the values more symmetrically can be done for performance purposes.

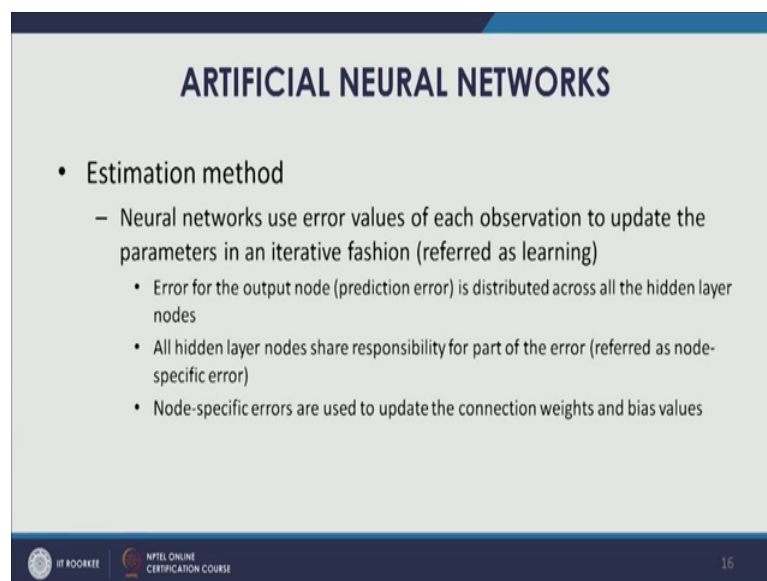
So, just like you know the normalization 0 to 1, and then this is log transformed we are if we have a right skewed variable. So, that the values are are more evenly spread. So, those kind of transformation can be done to improve the performance of our neural network model.

Let us discuss the estimation method. So, as we are familiar about the linear regression and logistic regression models these techniques and we know that least squares and maximum likelihood methods are used as estimation technique in these methods.

So, least square is used in linear regression and the maximum likelihood method is used in the logistic regression. However, as you would; as we have discussed while you know, when we were discussing these technique linear and logistic that a global metric of errors; for example, SSE sum of sum of square errors that is typically used to estimate the parameter.

So, these techniques least square and maximum likelihood; so their estimation procedure their estimation steps use this particular global matrix SSE or some other global matrix to estimate the parameters that are betas that, because this is we you know model the predictors as linear function and therefore, those beta values are estimated using these methods and typically a global metric of this kind SSE is used there.

(Refer Slide Time: 05:08)



ARTIFICIAL NEURAL NETWORKS

- Estimation method
 - Neural networks use error values of each observation to update the parameters in an iterative fashion (referred as learning)
 - Error for the output node (prediction error) is distributed across all the hidden layer nodes
 - All hidden layer nodes share responsibility for part of the error (referred as node-specific error)
 - Node-specific errors are used to update the connection weights and bias values

IT ROOKIE NPTEL ONLINE CERTIFICATION COURSE 16

However, we in neural network, we do not use any such you know global metric to compute the parameters. So, we will understand, so what is the technique estimation method, that is use in neural network.

So, neural networks use error values of each observation to update the parameters in an iterative fashion. So, it is not that we compute a you know we use a global matrix and the error computation based on those global matrix, they are either optimized there.

So, that error that error is minimized. So, that typically happens in linear and logistic as we discussed. So, in case of neural network it is for each observation, whatever error value using the typical formula actual value minus updates predicted value, so that error value that is used to update the parameters in an iterative fashion. So, after each observation we would be updating. So, this is particular process this particular step is referred as learning.

So, when we say that you know especially for data driven techniques, data driven model for example, classification and regression tree, neural network here, in this particular case. So, these techniques these models they learn from the data. So, in particular case, in this particular case artificial neural network the learning is based on the error values right. So, after each observation the learning comes from the error value for that particular observation and this happens in an iterative fashion and all the observations are used in this learning process.

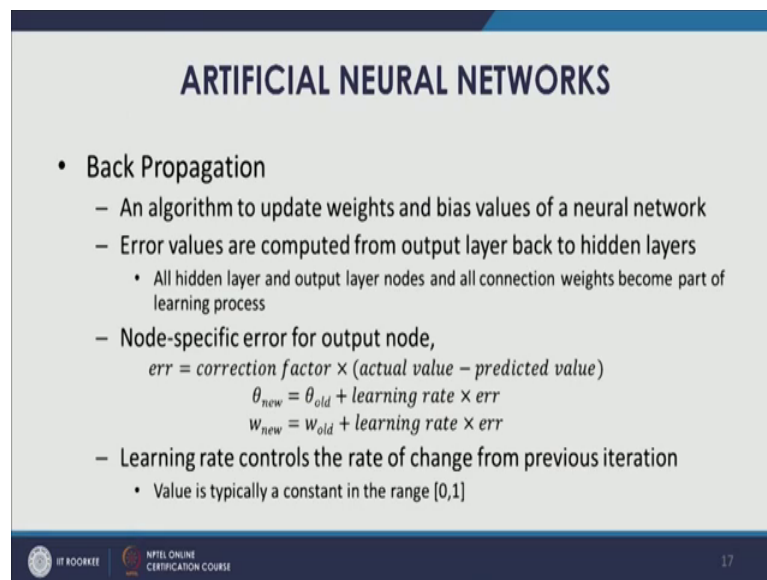
So, as you can see in few more points in the slide error for the output node, that is you know prediction error. So, the is typically distributed across all the hidden layer node. So, that that is what happens in neural network; then all hidden layer nodes share responsibility for part of the error. So, you would from this it would be clear that all the nodes that we talk about for example, hidden layer nodes or hidden layer nodes and output layer nodes.

So, all those nodes they share they are part of the learning process, they share response and learning happens through error values after each observation. So, they also; so these nodes also share responsibility for part of the error. So, this is this particular is referred as node specific error. So when the part of the error that is shared by hidden layer nodes that is called node specific error.

Now, these nodes specific errors are used to update the connection weights and bias values right. So, we are going to have connections between input layer nodes and output layer nodes and similarly for between hidden layer nodes to output layer nodes. So, all those connections, weights and the bias values for hidden layer nodes and the output

layer nodes. So, all those things all those parameters are going to be updated using these node is specific errors. So, the actual algorithm that is actually used to perform this process that is back propagation so as the anyway we can say that estimation method that is using neural network is actually back propagation algorithm.

(Refer Slide Time: 08:35)



ARTIFICIAL NEURAL NETWORKS

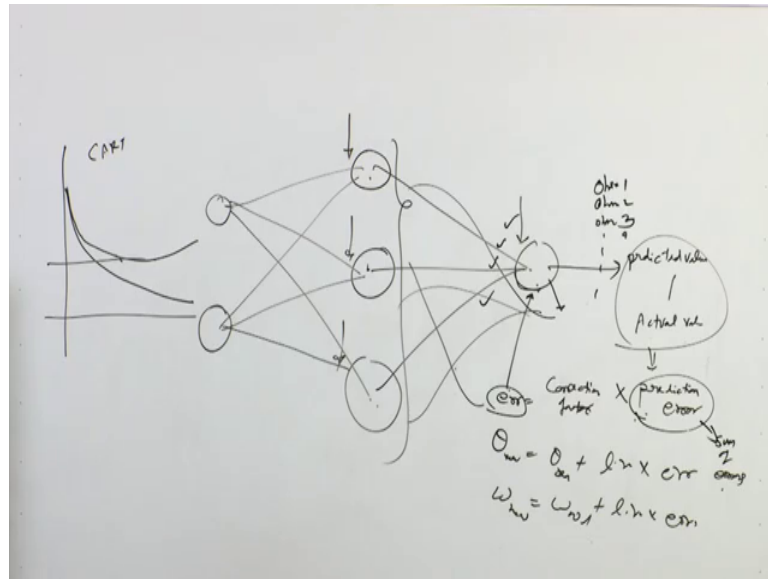
- **Back Propagation**
 - An algorithm to update weights and bias values of a neural network
 - Error values are computed from output layer back to hidden layers
 - All hidden layer and output layer nodes and all connection weights become part of learning process
 - Node-specific error for output node,
$$err = correction\ factor \times (actual\ value - predicted\ value)$$
$$\theta_{new} = \theta_{old} + learning\ rate \times err$$
$$w_{new} = w_{old} + learning\ rate \times err$$
 - Learning rate controls the rate of change from previous iteration
 - Value is typically a constant in the range [0,1]

IT ROORKEE | NPTEL ONLINE CERTIFICATION COURSE 17

So, this particular algorithm is used to update weights and biased values of a neural network.

So, error values they are computed from output layer back to hidden layers that is why you would see that it is called back propagation. If we have to understand this we can understand the same thing through a diagram; for example, in previous lecture; we had used this particular neural network.

(Refer Slide Time: 09:04)



So, we had two nodes here; so we had two nodes in input layer, then we had three nodes hidden layer, and then we had one node in output layer. So, the error values would be computed here, and then they would be propagated back to hidden layer nodes right. So, that is why this algorithm is called back propagation; so error values are computed from output layer back to hidden layer nodes.

The next point as you can see in the slide all hidden layer and output layer nodes and all connection weights become part of learning process right. So, since this particular value, that is computed here the error value that is computed here. Now, this is propagated back to the other layers the previous layers of the network.

So, all the connections that are going to be here all the connections and the corresponding weight the bias values they are going to be part of the learning process. Similarly here; so since all the connections all the weights and vast values are going to be updated using this particular error you know back in a back propagation manner. So, all of all these parameters are going to be all these nodes and the connections are going to be part of the learning process.

So, typically how do we compute these errors? So, let us talk about the node is specific error, how it is computed. So, you can see that first we will discuss for the output node. So, for example, in this case we have one output node. So, if we consider this particular network diagram, then we can see that for this output node error can be computed in this

fashion. So, we have a correction factor, that is multiplied to the predicted error, that is actual value minus predicted value. So, this is prediction error. So, this prediction error that is going to be computed here; so the predicted scores, so the output value that comes from this particular node output node, that is going to be the predicted value as we talked about in the previous lecture.

And now we will have the actual value, now the difference actual value minus predicted value. So, this is going to be our predicted value, so this is typical definition of prediction error. So, this is a typical definition for any error computation. Now, this particular value is going to be multiplied by a correction factor, this particular value is going to be multiplied by a correction factor and that; so now, we will get a new value, now this value this value would be assigned to the output layer node. So, now, this particular value is going to be used to update the parameters related to this particular node.

So, as we can see that the bias value we can see that in the next equation, we can see that bias value θ_{new} is θ_{old} plus learning rate into error. So, the error value that has been computed in the previous step; So, we will multiplied with learning rate and then add it to the old value of old bias value. So, that will give us the new bias value to be used for the networks. So, any new observation that is run through the network it will use this particular new bias value. So, what is learning rate?

So, learning rate controls the rate of change from previous iteration. So, you can see what part of; what amount of the computed error value that we just saw; what part is actually used for the learning process that is controlled by learning rate. So, typically the learning rate value is typically it is a constant value and in the range 0 to 1. So, that will determine what; how much; what amount of error part of the error is going to be used for the learning process the updating process.

So these are the steps, so node specific error from the prediction error we can compute the prediction error that we get for a particular observation. So, as I talked about for every observation we will learn through this particular network. And in previous lecture, we talked about that all these through an example also in R as well we understood this that all the connection weights and bias values they are initialized to random numbers. Now, once the first observation is passed through this particular network run through this

network these randomly initialized you know weight values and bias values are going to be used to produce the predicted value.

And once that predicted value for that particular objects observation number 1, is actually the core number one is actually computed, then it can be subtracted from actual value and we will get the prediction error and from there we can compute the error and we will know the values as given in the you know slide that we can use the learning rate and the control the amount of change and we will get the new values and new values for thetas and w weights and those can be updated.

So, the bias values here for output node and the weights can be updated. So, this was for the output node and; however, the process the steps are quite similar for the hidden layer nodes as well; however, at this point I would like to point out that this was for the first observation. Now, second observation also once these weights and you know connection weights and bias values have been updated for the whole network, then the second observation will pass through these updated values right.

And again we will reach to this value will learn through the network, we will have the predicted value again the same process will continue and again the updation and learning will happen. And in this fashion observation 2, 3, 4 and in this fashion will keep running these observation through the network and the network will keep on learning in this fashion right. In the prediction the correction factor will get the error and that would be updated as we talked about right $\theta_{\text{new}} = \theta_{\text{old}} + \text{learning rate} \times \text{error}$ value.

Similarly, $w_{\text{new}} = w_{\text{old}} + \text{learning rate} \times \text{error}$ value; So, in this fashion now the new values will be updated. So, we talked about how the values are computed for a output node, these values are being computed for the output node. Similarly, the same type of process similar steps are performed for to compute node specific error for hidden nodes. So, the error value that we have computed for output node this one.

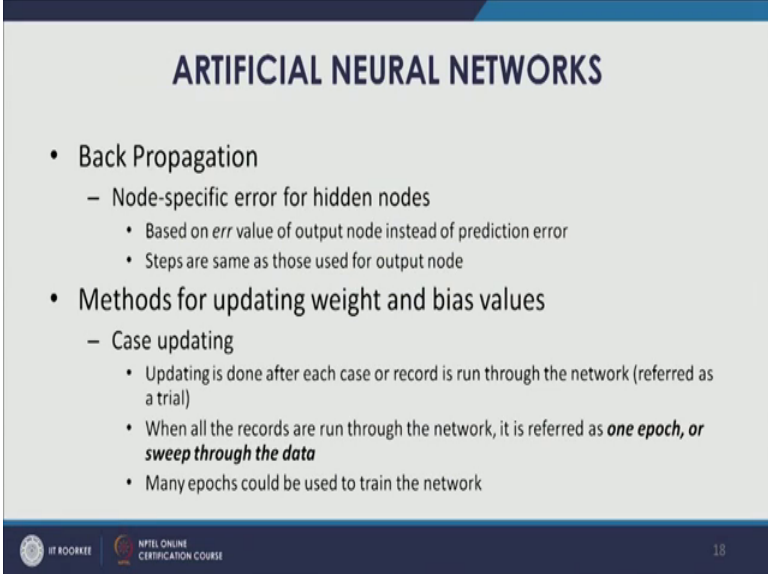
Now, this is going to be used by hidden layer nodes, this is going to be used by hidden layer nodes to perform these steps similar to what we have here. So, this particular error, now this is going to be used here; and these steps are going to be quite similar again we will you know; so in place of prediction error now we will have this error value, now we will have this error value for hidden nodes and other steps are going to be similar. So, we

will use a correction factor and that correction factor is, then going to be used for all these nodes.

Now, the correction factor is specific to the node. So, the value that is the that has been the output value, that is there for a particular node. So, whether it is for the output node or the hidden layer nodes that; so correction factor could be based on these output values and therefore, it is it could it is going to be different it is going to be different for all these nodes.

So, that correction factor then would be for output node it would be multiplied with the prediction error, for hidden layer node it would be multiplied by the error value that is computed for the output node. So, in this fashion all the weight and bias values are going to be updated.

(Refer Slide Time: 18:32)



The slide is titled "ARTIFICIAL NEURAL NETWORKS" in a bold, dark blue font. Below the title, there are two main bullet points. The first is "Back Propagation", which has a sub-bullet "Node-specific error for hidden nodes" containing two further points: "Based on err value of output node instead of prediction error" and "Steps are same as those used for output node". The second main bullet point is "Methods for updating weight and bias values", which has a sub-bullet "Case updating" containing three points: "Updating is done after each case or record is run through the network (referred as a trial)", "When all the records are run through the network, it is referred as *one epoch, or sweep through the data*", and "Many epochs could be used to train the network". At the bottom of the slide, there are logos for "IIT ROORKEE" and "NPTEL ONLINE CERTIFICATION COURSE", along with the page number "18".

- **Back Propagation**
 - Node-specific error for hidden nodes
 - Based on err value of output node instead of prediction error
 - Steps are same as those used for output node
- **Methods for updating weight and bias values**
 - Case updating
 - Updating is done after each case or record is run through the network (referred as a trial)
 - When all the records are run through the network, it is referred as *one epoch, or sweep through the data*
 - Many epochs could be used to train the network

Now, let us talk about the few more things about updating weight and bias values. So, there are two main approaches for this updation. So, for example, what we have discussed till now is; actually you know case updating. So, what is case updating? So, updating is done after each case or record is run through the network referred as a trial.

So, this is what we have discussed. After each observation this case updating you know this is called case updating, after each observations the bias and weight values are updated and this happens for all the observations in the data set.

And when all the records are run through the network it is referred as one epoch or sweep through the data. So, that is referred as a one sweep through the data, if we are using training partition data set, then it is going to be one sweep through the training partition data. So, in the training process learning process of the network we might have to run many such epochs. So, as you can see here one point here last point here in case updating many epochs could be used to train the network.

Now, the another approach for updating weight and bias value is called a batch updating. So, this is different from what we have discussed in you know case updating here. So, what happens in batch updating? The updating is done after all the records are run through the network right. Till now what we have been discussing is observation is run through; so the weights and biased values are randomly initialized for the network.

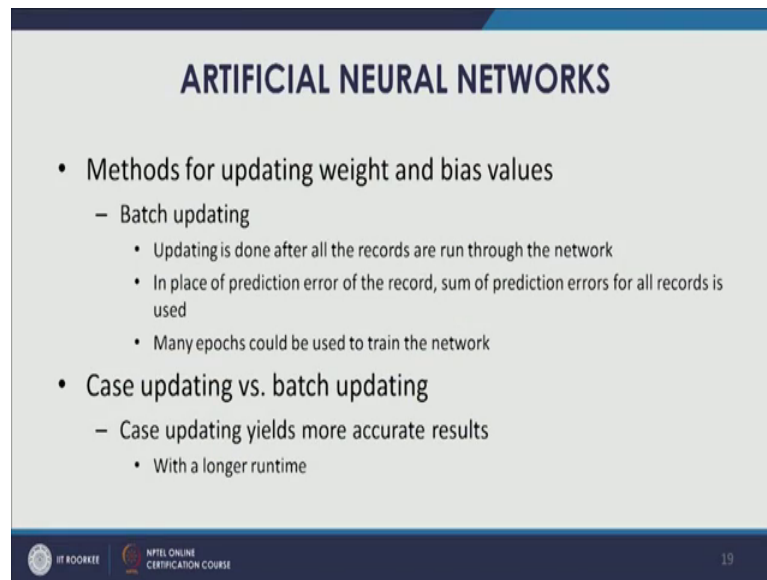
So, once first we decide the network architecture, the network structure once that is decided, then we will in randomly initialize the connection weights and bias values, then we will run the first observation through the network, then the second observation through the network, third observation through the network, and when all the observations have been run that is one epoch. So, we might have to you know execute many such epochs and this is, what we are discussing under case updating this is called case updating.

And, when we talk about batch updating first all the observations; so you would see that in case updating, after one observation has been run immediately these computations are done and the bias values and weight values they are updated; however, in batch updating all the observations are run through the network, and then this particular these kind of computations are performed. So, what is the difference in these computations?

So, in place of prediction error, because the prediction error was a specific to the observation, so for every observation you know we had a prediction error in the case updating. Now, in place of that prediction error we will use some of prediction errors for all records, you know that is for all records.

So, in place of prediction error some of these errors is going to be used right. So, this is going to be used and accordingly the biased values and weight values are going to be updated. So, again in batch updating also even you know many epochs, we would have to run to train the network.

(Refer Slide Time: 22:12)



The slide is titled "ARTIFICIAL NEURAL NETWORKS" in a bold, dark blue font. Below the title, there are two main bullet points. The first is "Methods for updating weight and bias values", which has a sub-bullet "Batch updating". Under "Batch updating", there are three sub-bullets: "Updating is done after all the records are run through the network", "In place of prediction error of the record, sum of prediction errors for all records is used", and "Many epochs could be used to train the network". The second main bullet point is "Case updating vs. batch updating", which has a sub-bullet "Case updating yields more accurate results". Under this, there is one sub-bullet: "With a longer runtime". At the bottom of the slide, there are logos for "IIT ROORKEE" and "NPTEL ONLINE CERTIFICATION COURSE", and the number "19" in the bottom right corner.

- **Methods for updating weight and bias values**
 - **Batch updating**
 - Updating is done after all the records are run through the network
 - In place of prediction error of the record, sum of prediction errors for all records is used
 - Many epochs could be used to train the network
- **Case updating vs. batch updating**
 - **Case updating yields more accurate results**
 - With a longer runtime

Now, in terms of performance in terms of performance and modeling, how case updating and batch updating what are the advantages or disadvantages; so the one important difference that you can understand through the process itself; case updating is more rigorous right. So, in case updating every time when an observation is done through the network, the updation of these values the back propagation of these values that take place and that takes place for each record. However, in batch updating this happens once for all the observations in the data set.

So, you would see that case updating is more rigorous and therefore, it would require more runtime as well. So, the accuracy of model would be much better in case updating; however, it will take it will come at the cost of longer runtime.

Now, let us talk about the stopping criteria for updating.

(Refer Slide Time: 23:11)

ARTIFICIAL NEURAL NETWORKS

- Stopping Criteria for updating
 - Small incremental change in bias and weight values from previous iteration
 - Rate of change of error function values reaches a required threshold
 - Limit on no. of runs is reached



So, when should be stopped; we talk about that to train the network to learn from the data; to learn from the data, we will have to execute many epochs will have to perform many sweeps through the data. So, when do we stop? So, what should be the; what are the some of them you know key stopping criteria's for this learning process.

So, few points are discussed here, as you can see small incremental change in bias and weight values from previous iteration. So, if the bias and weight values the change that is happening; so that is that you can see learning rate into error. So, that component the second component; that is being added to the previous value this is very incremental there is not significant change.

So, when we run through; when we run our first observation, then this is this particular component would be quite significant as we go through other observations, this value might decrease and you know as we go through all the observation and we you know one epoch is completed, we go through second epoch still there would be some you know significant you know component here.

However, as we move as we do more computations of this kind this is you know that significant that significance significant value that is being added here you know added or subtracted here might that that magnitude of that might decrease and it might just remain an incremental value very small change might be happening.

So, probably that is the indication that the model or network has saturated and therefore, we should stop the learning process. So, this is what is mentioned here. The small

incremental change in bias and weight values might indicate that probably we should stop the learning process.

Other stopping criteria could be a rate of change of error function values as there is a required threshold. So, overall; over overall performance of the model, so some error function could be used for example, SSE could be used to check the overall performance of the model. So, therefore, we can see for you know different different sweep. So, that we that we execute different sweeps, that we execute we can have the; you know that you can see; we can check the change of error values there that is for the model error. And we see when that is you know that is reaching the required threshold.

So, if the rate of change is you know has these there is required so; that means, the rate of change in terms of model performance is not much. So, that is when we talk about this error function that is with respect to the model performance. So, that error, so it could be SSE or some other metric, and if the rate of change is not much then probably all has reached to a established specified value threshold value, then probably we should stop the learning process.

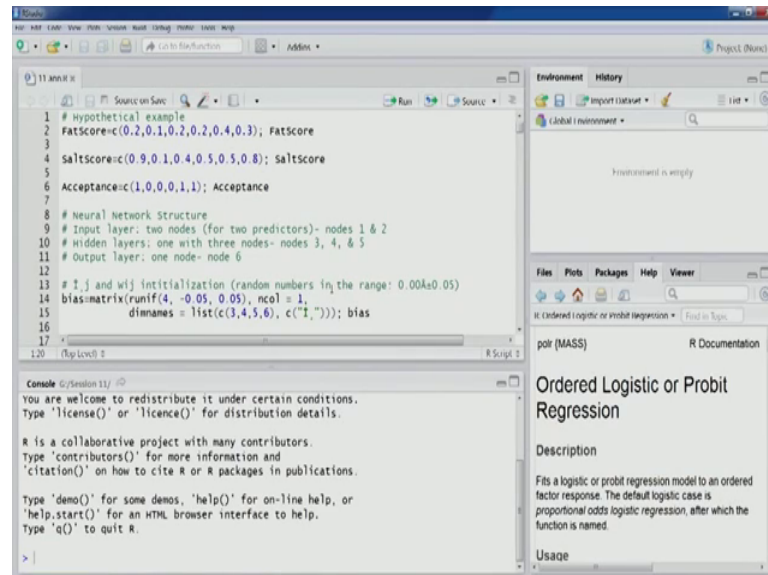
Now, so this is second point is quite similar to what we have been discussing in previous techniques as well. Now, for example, in the cart algorithm also we talked about that; you know the classification, misclassification rate on training partition and the same on you know validation partition and somewhere when the value is minimized probably you know that is the point where we should stop the tree growth. So, this is second point is quite similar to this.

Now, there could be another criteria, that is limit on number of runs is released. So, we can also specify the number of runs the maximum limits. So, this is going to be the probably the last result to stop the network is not able to converse, then probably we would like to set a limit at the point this particular training process learning process would be stopped.

Typically this is the last result, when we are not able to you know stop the stop the learning process the updating process either from first criteria, second criteria, whatever is being used and then probably we should stop at some point and too many runs have been done and still no conversion has taken place. And probably we should we can do

that by specifying the limit on number of runs. After this discussion let us understand some of these concepts through a modeling exercise in R.

(Refer Slide Time: 28:02)



So, in the previous lecture, we had used this particular example that is we had this fat score and salt score, and this was for different experimentation with respect to cheese samples and whether those cheese samples and that combination of fat and salt score, whether that is being accepted or rejected by the experts. So, we had this hypothetical later a few a values also and that was used to understand the computations that we do in a neural network. So, we are going to use this same example to build our to build our to train our neural network model.

So, let us execute this code. So, we will have fat score, as you can see just 6 values are there.

(Refer Slide Time: 28:51)


```

1 # Hypothetical example
2 Fatscore=c(0.2,0.1,0.2,0.2,0.4,0.3); Fatscore
3
4 saltscore=c(0.9,0.1,0.4,0.5,0.5,0.8); saltscore
5
6 Acceptance=c(1,0,0,1,1); Acceptance
7
8 # Neural Network Structure
9 # Input layer: two nodes (for two predictors)- nodes 1 & 2
10 # Hidden layers: one with three nodes- nodes 3, 4, & 5
11 # Output layer: one node- node 6
12
13 # i, j and wij initialization (random numbers in the range: 0.00&0.05)
14 bias=matrix(runif(4, -0.05, 0.05), ncol = 1,
15             dimnames = list(c(3,4,5,6), c("1",""))); bias
16
17
41 (Top Level) 2
R Script 2

```

Console G:/Session 11/

```

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> Fatscore=c(0.2,0.1,0.2,0.2,0.4,0.3); Fatscore
[1] 0.2 0.1 0.2 0.2 0.4 0.3
>

```

Environment History

Global Environment

Values

```

Fatscore num [1:6] 0.2 0.1 0.2 0.2 ...

```

Files Plots Packages Help Viewer

Ordered Logistic or Probit Regression

Description

Fits a logistic or probit regression model to an ordered factor response. The default logistic case is proportional odds logistic regression, after which the function is named.

Usage

So, these sample size is going to be quite a small just 6 values; however, we are going to go going through this exercise for the illustration purpose. So, let us look at the salt score.

(Refer Slide Time: 29:06)

```

33 x=bias[i,1]*weightsHO[1,j]*output[1]+weightsHO[2,j]*output[2]+
34 weightsHO[3,j]*output[3]
35 output[4]=1/(1+exp(-x))
36
37 # classify first record using cutoff value=0.5
38 ifelse(output[4]>0.5, 1, 0) # predicted class
39 Acceptance[1] # actual value
40
41 # Model for hypothetical data
42 library(neuralnet)
43
44 df=data.frame(Fatscore, saltscore, Acceptance)
45 str(df)
46
47 # Neural Network Model
48 # startweights vector: no. of all bias (4) and connection weight values (9)
49
441 (Top Level) 2
R Script 2

```

Console G:/Session 11/

```

'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> Fatscore=c(0.2,0.1,0.2,0.2,0.4,0.3); Fatscore
[1] 0.2 0.1 0.2 0.2 0.4 0.3
> saltscore=c(0.9,0.1,0.4,0.5,0.5,0.8); saltscore
[1] 0.9 0.1 0.4 0.5 0.5 0.8
> Acceptance=c(1,0,0,1,1); Acceptance
[1] 1 0 0 1 1
> library(neuralnet)
Error in library(neuralnet) : there is no package called 'neuralnet'
>

```

Environment History

Global Environment

Values

```

Acceptance num [1:6] 1 0 0 1 1
Fatscore num [1:6] 0.2 0.1 0.2 0.2 ...
saltscore num [1:6] 0.9 0.1 0.4 0.5 ...

```

Files Plots Packages Help Viewer

Ordered Logistic or Probit Regression

Description

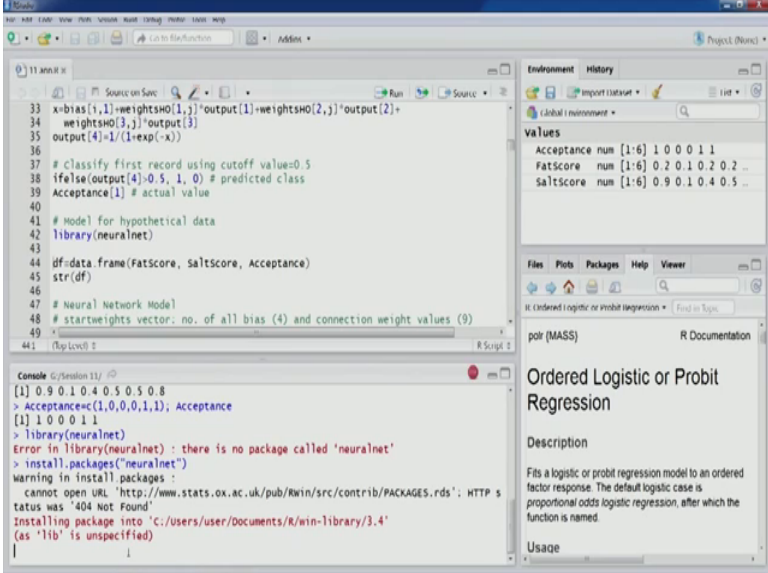
Fits a logistic or probit regression model to an ordered factor response. The default logistic case is proportional odds logistic regression, after which the function is named.

Usage

So, this is second variable than acceptance, then some of these computation that you see and we have gone through in previous lecture. So, let us skip through; I will stop here. So, this is the package that we require neural net.

So, this is the packet that we are going to use for our neural network modeling exercise. So, let us load this particular library a neural network; so this is probably not installed. So, let us install this package.

(Refer Slide Time: 29:49)



The screenshot shows the R Studio interface. The script editor contains R code for a neural network model. The console shows the execution of the code, which results in an error: "Error in library(neuralnet) : there is no package called 'neuralnet'". The message pane on the right shows the documentation for the 'poir (MASS)' package, which is an Ordered Logistic or Probit Regression model.

```
33 x=bias[i,1]*weightsHO[1,j]*output[1]+weightsHO[2,j]*output[2]+
34 weightsHO[3,j]*output[3]
35 output[4]=1/(1+exp(-x))
36
37 # Classify first record using cutoff value=0.5
38 ifelse(output[4]>0.5, 1, 0) # predicted class
39 Acceptance[i] # actual value
40
41 # Model for hypothetical data
42 library(neuralnet)
43
44 df=data.frame(Fatscore, Saltscore, Acceptance)
45 str(df)
46
47 # Neural Network Model
48 # startweights vector: no. of all bias (4) and connection weight values (9)
49
441 (Top Level) 2
```

Console

```
G:/Session 11/
[1] 0.9 0.1 0.4 0.5 0.5 0.8
> Acceptance=c(1,0,0,1,1); Acceptance
[1] 1 0 0 0 1 1
> library(neuralnet)
Error in library(neuralnet) : there is no package called 'neuralnet'
> install.packages("neuralnet")
Warning in install.packages :
cannot open URL 'http://www.stats.ox.ac.uk/pub/Rwin/src/contrib/PACKAGES.rds': HTTP s
tatus was '404 Not Found'
Installing package into 'C:/Users/user/Documents/R/win-library/3.4'
(as 'lib' is unspecified)
```

Environment History

Libraries in environment

Values

Acceptance	num	[1:6]	1	0	0	1	1				
Fatscore	num	[1:6]	0	2	0	1	0	2	0	2	..
Saltscore	num	[1:6]	0	9	0	1	0	4	0	5	..

Files Plots Packages Help Viewer

Ordered Logistic or Probit Regression

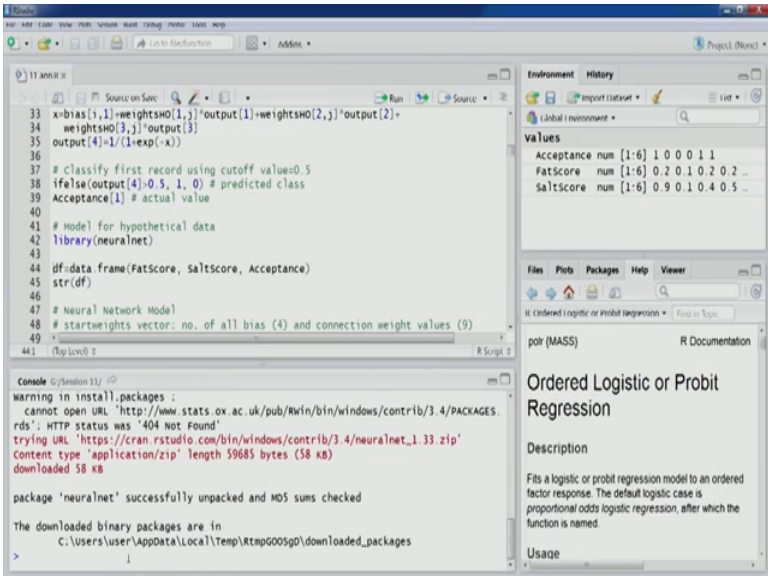
Description

Fits a logistic or probit regression model to an ordered factor response. The default logistic case is proportional odds logistic regression, after which the function is named.

Usage

So, once; so this is so there are many many packages which are available in R that can be used for neural network modeling exercise.

(Refer Slide Time: 29:56)



The screenshot shows the R Studio interface. The script editor contains the same R code as the previous screenshot. The console shows the execution of the code, which results in a warning message: "Warning in install.packages : cannot open URL 'http://www.stats.ox.ac.uk/pub/Rwin/bin/windows/contrib/3.4/PACKAGES.rds': HTTP status was '404 Not Found'". The message pane on the right shows the documentation for the 'poir (MASS)' package, which is an Ordered Logistic or Probit Regression model.

```
33 x=bias[i,1]*weightsHO[1,j]*output[1]+weightsHO[2,j]*output[2]+
34 weightsHO[3,j]*output[3]
35 output[4]=1/(1+exp(-x))
36
37 # Classify first record using cutoff value=0.5
38 ifelse(output[4]>0.5, 1, 0) # predicted class
39 Acceptance[i] # actual value
40
41 # Model for hypothetical data
42 library(neuralnet)
43
44 df=data.frame(Fatscore, Saltscore, Acceptance)
45 str(df)
46
47 # Neural Network Model
48 # startweights vector: no. of all bias (4) and connection weight values (9)
49
441 (Top Level) 2
```

Console

```
G:/Session 11/
Warning in install.packages :
cannot open URL 'http://www.stats.ox.ac.uk/pub/Rwin/bin/windows/contrib/3.4/PACKAGES.
rds': HTTP status was '404 Not Found'
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.4/neuralnet_1.33.zip'
Content type 'application/zip' length 59685 bytes (58 kB)
downloaded 58 kB
package 'neuralnet' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:/Users/user/AppData/Local/Temp/RtmpG00SPD/downloaded_packages
```

Environment History

Libraries in environment

Values

Acceptance	num	[1:6]	1	0	0	1	1				
Fatscore	num	[1:6]	0	2	0	1	0	2	0	2	..
Saltscore	num	[1:6]	0	9	0	1	0	4	0	5	..

Files Plots Packages Help Viewer

Ordered Logistic or Probit Regression

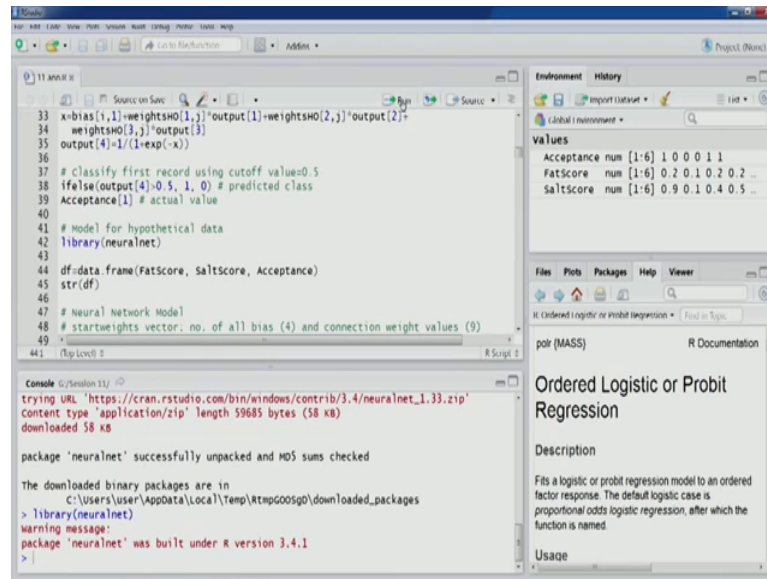
Description

Fits a logistic or probit regression model to an ordered factor response. The default logistic case is proportional odds logistic regression, after which the function is named.

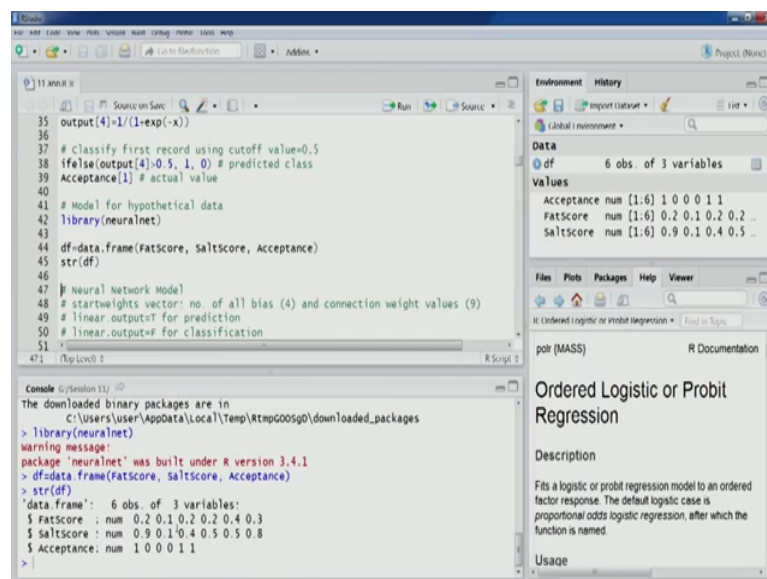
Usage

So, neural network being one of the most used package. So, that is why we are using this one. So, otherwise for this is applicable for other techniques as well, so there could be

(Refer Slide Time: 30:29)



(Refer Slide Time: 30:41)



So, let us look at the structure of this data frame. So, you can see fat score is the 6 values and all our numerical as of now; so as you can see probably this particular process is actually for a classification task. We would like to classify whether a particular key sample is acceptable or not based on the fat score and salt score.

However, you would see that pack the package that we are going to use neural network that restrict that does not require us to change the change the variable type to for example, acceptance is a categorical outcome variable, but the package does not allow us to do that; so all the computations are done internally within the function that are available in this package.

So, now let us talk about the model. So, structure neural network is structured as we talked about that; that is the first thing that we need to decide and of course. So, we can do certain experimentation with the neural network structure. However, for our illustration we will use this particular neural network structure for this example; two nodes in the input layer and then three nodes in the hidden layers the one node in the output layer so in this fashion, because our output variable is binary variable.

So, we can use now a neural network, as you can see this function is going to be used to build the model and we would see that linear output is one argument there, and this argument has to be specified as false for classification, if we are building model for classification task that it has to be specified as true, if we are building model for the prediction task.

So, I we will stop here and we will continue our discussion on this particular model size in R in the next lecture.

Thank you.