MIS 637– Data analytics and Machine Learning

FINAL TEAM PROJECT

ON NEURAL NETWORK

Guided By
Prof. Foad Pajouh

TEAM 1-

Asad Sajid Awan

Kanika Yadav

Dipesh Shah

Krishna Sai Vikas Padmala

Gurudatta Sai Prasad Akkala

Praneeth Kumar Bodolla

What are neural networks?

A neural network is a collection of algorithms that aims to identify underlying links in a set of data using a method that imitates how the human brain functions. In this context, neural networks are systems of neurons that can be either organic or synthetic in origin. Since neural networks can adapt to changing input, the network can produce the best outcome without having to change the output criterion. In other words, A neural network functions in a manner like that of the human brain. In a neural network, a "neuron" is a mathematical function that gathers and categorizes data in accordance with a particular architecture. The network is quite like statistical techniques of regression analysis.

It is a multi-layered perceptron (MLP), which has layers that are connected to one another. Input patterns are gathered by the input layer. Input patterns may map to classifications or output signals in the output layer. For instance, a list of values for technical indicators regarding security could be included in the patterns.

Building a Neural Network Model -

Overview of the dataset:

- There are no categorical variables in this dataset; it is solely numerical.
- The code contains 50 observations with 1 dependent variable, and 4 independent variables, and we believe they are all normalized.
- We did secondary research to decide which programming language we should use. In textbook of the course, we found a mention of the R Zone in the Neural networks chapter. Accordingly, we used “neuralnet” package in R-studio.
- The data that was provided to us was written in a horizontal format, but for the sake of convenience, we changed it into a vertical format so that the Excel or CSV file, on which we conducted our data analysis, can be simply read in the R studio.
- We have used the given a learning rate.

Implementation:

- The Logical Activation function from the neural-net package is the sigmoid function that is the result of the package based on R and the sigmoid function are the same.
- The command for hidden layer takes a vector in which the first number represent hidden layers and second number represents the number of nodes hence the vector is (2,2) that represents layers and nodes respectively. It’s a 2/2 vector in which the first vector has 2 hidden layers. Whereas the second vector defines 2 hidden layers having 2 neurons each.
- The algorithm we choose is a backpropagation.

- By considering Y as our dependent variable the linear output is true and we also took $X_{1,}$ X_2, X_3, and X_4 as our independent variables.

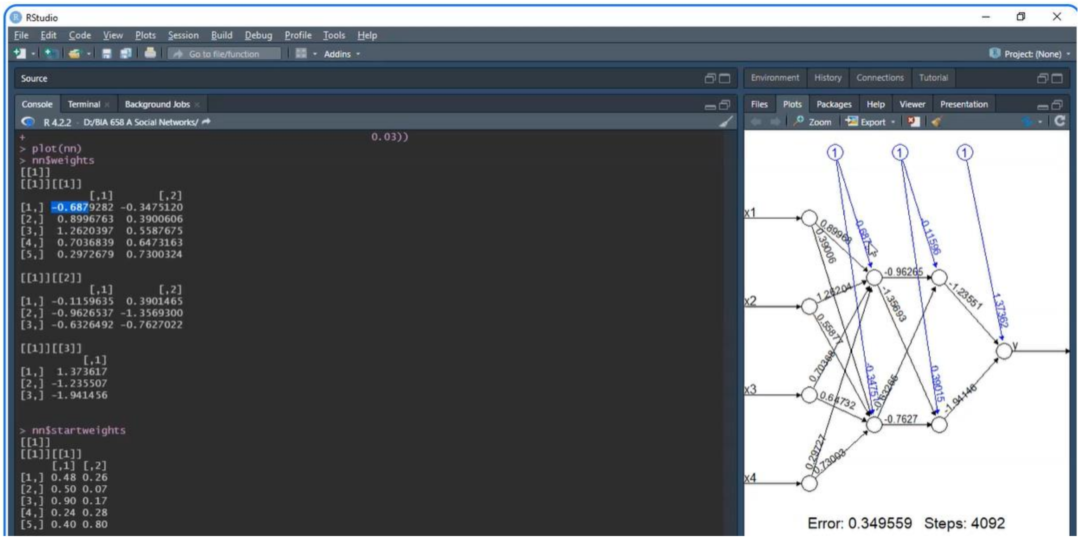
Learning Experience in building the neural network:

Challenge 1: The “neuralnet” package of R gave us the options to use “Logisitic” as the activation function, while in the course, we have been taught to use sigmoid function. At the time of implementation, we didn’t know that Logistic and Sigmoid are both the same. Therefore, we made a sigmoid function of our own using the exponent. It is mentioned in the R script.

Challenge 2: As per the problem diagram we had 2 hidden layers, each of them having two neurons, so how to build that into the formula used in programming was another challenge. Using YouTube videos, we found that we can use a vector in the formula to solve this problem. We solved it by setting Hidden = c(2,2) in which the first value tells us there are 2 hidden layers and 2nd value tells us number of neurons.

Challenge 3: The biggest challenge was to assign the weights provided in the excel dataset as the initial weights. We first ran the algorithm with the initial weights in the same order as they were given to us. The confusion was, we were not sure if we have written W1A first in the order, it will be indeed taken as W1A and not in place of some other weights. We solved this by re-engineering the code. We first ran the algorithm with the weights as they were provided, created the model, looked at the plot, and discovered the first value of weights we gave as W1A is being read as W0A. And the order in which R is reading the weights is such that W0A, W1A, W2A, W3A, W4A and then for node B, C, D, and Z. So, we re-arranged our weights in the formula as this assortment in the “startweights”.

Looking at the below screenshot, as per convention this is W_{OA}, which is 0.687 but the first weight that we have been given is 0.48. So, there is some mismatch. We had to sort the weights because as per excel we had to give W_{1A}.

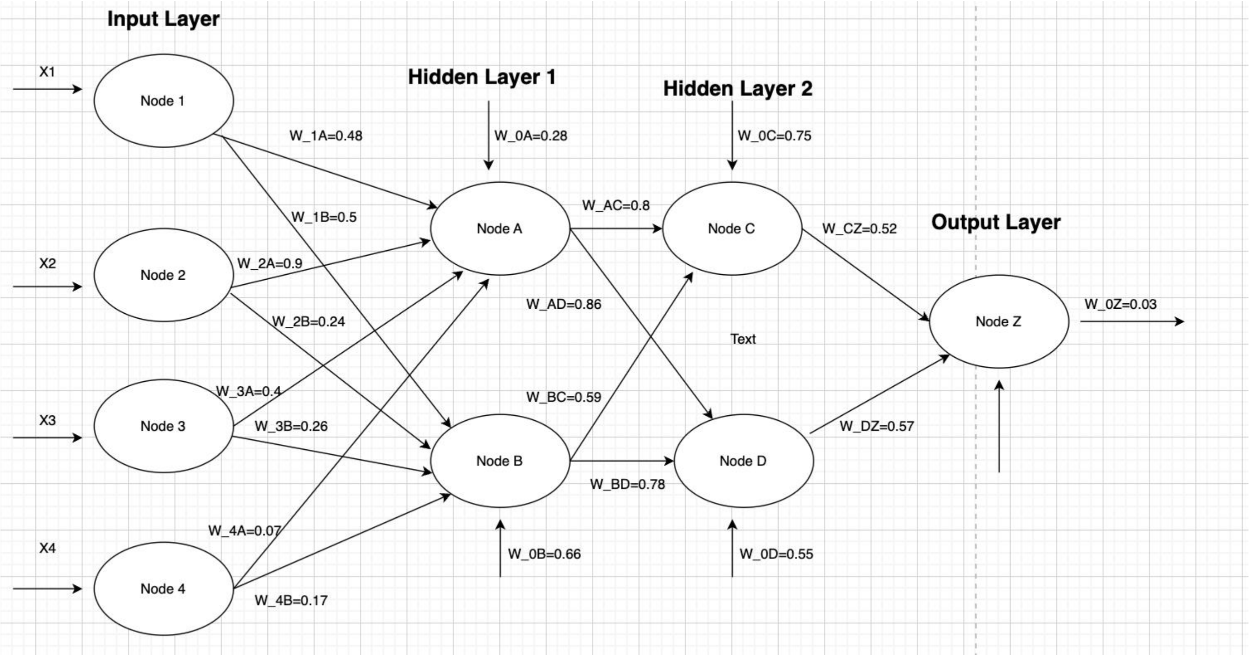


```
+                                                          0.03))
> plot(nn)
> nn$weights
[[1]]
[[1]][[1]]
           [,1]       [,2]
[1,] -0.6879282 -0.3475120
[2,]  0.8996763  0.3900606
[3,]  1.2620397  0.5587675
[4,]  0.7036839  0.6473163
[5,]  0.2972679  0.7300324

[[1]][[2]]
           [,1]       [,2]
[1,] -0.1159635  0.3901465
[2,] -0.9626537 -1.3569300
[3,] -0.6326492 -0.7627022

[[1]][[3]]
          [,1]
[1,]  1.373617
[2,] -1.235507
[3,] -1.941456


> nn$startweights
[[1]]
[[1]][[1]]
     [,1] [,2]
[1,] 0.48 0.26
[2,] 0.50 0.07
[3,] 0.90 0.17
[4,] 0.24 0.28
[5,] 0.40 0.80
```

Error: 0.349559 Steps: 4092

The wrong weights (initial wrong assignment of weights) were giving us 0.349 as error, the correct weight sorting gave us error 0.347508 which clearly shows improved error rate.

Here, the R neuralnet package was used to work on this problem. Following diagram shows the neural network with the initial weights.



Formula used -

```
nn = neuralnet (y ~ x1+x2+x3+x4+x4, data = data_nn, hidden = c(2,2),
                learningrate = 0.02, algorithm = ‘backprop’,
                err.fct = ‘sse’, act.fct = sigmoid, linear output = TRUE,
                starweights = c(0.48, 0.5, 0.9, 0.24,
                                0.4, 0.26, 0.07, 0.17,
                                0.28, 0.8, 0.86,
                                0.66, 0.59, 0.78,
                                0.75, 0.52, 0.55, 0.57,
                                0.03))
```

Where:

Inputs independent variables - x1,x2,x3,x4

Response Variable - y

Dataset - c(2,2) – 2 hidden layer with 2 neurons in each.

Learning rate - 0.02

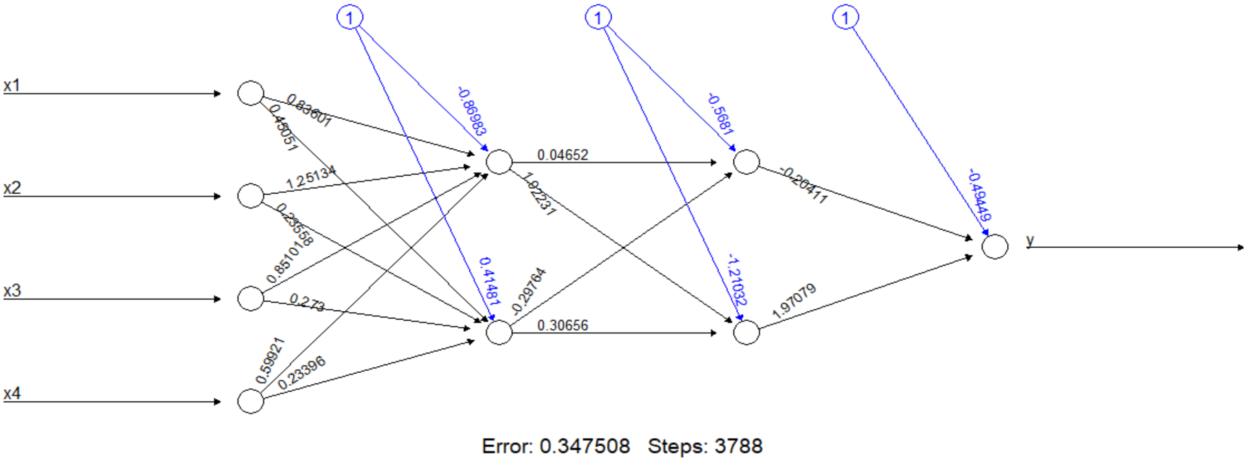
Algorithm - backpropogation,

Activation function – sigmoid / Logical activation form R

Output - ‘sse’ is sum of square error

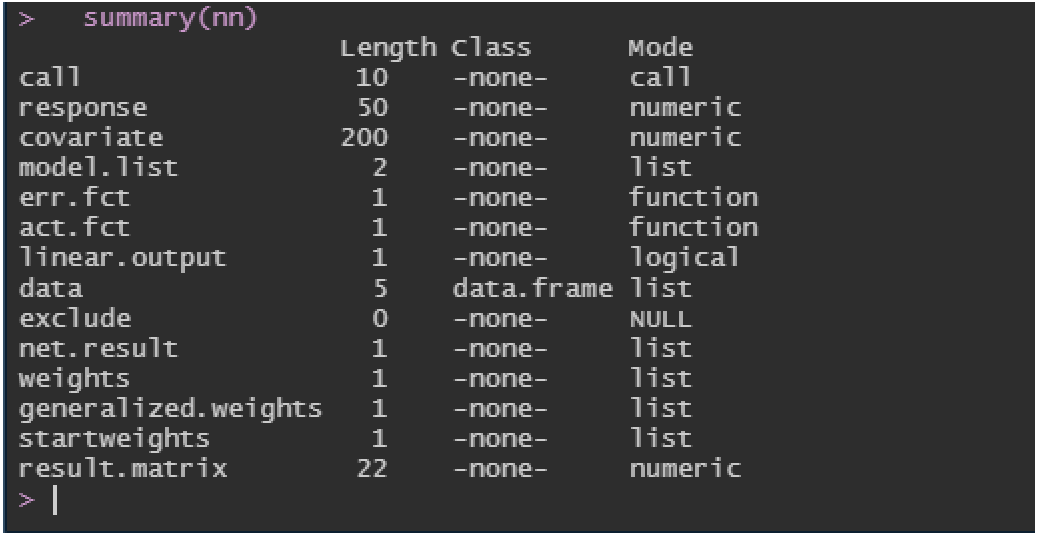
The output was linear because y variable is the linear variable.

plot(nn)



RESULTS—

summary(nn)



```
>   summary(nn)
                    Length Class      Mode
call                 10     -none-    call
response             50     -none-    numeric
covariate           200     -none-    numeric
model.list            2     -none-    list
err.fct               1     -none-    function
act.fct               1     -none-    function
linear.output         1     -none-    logical
data                  5     data.frame list
exclude               0     -none-    NULL
net.result            1     -none-    list
weights               1     -none-    list
generalized.weights   1     -none-    list
startweights          1     -none-    list
result.matrix        22     -none-    numeric
> |
```

nn$result.matrix

error	**0.347508**
reached.threshold	0.009994
steps	3788
Intercept.to.1layhid1	-0.86983
x1.to.1layhid1	0.836011
x2.to.1layhid1	1.251342
x3.to.1layhid1	0.851006
x4.to.1layhid1	0.599214
Intercept.to.1layhid2	0.414807
x1.to.1layhid2	0.450509
x2.to.1layhid2	0.235579
x3.to.1layhid2	0.272997
x4.to.1layhid2	0.233965
Intercept.to.2layhid1	-0.5681
1layhid1.to.2layhid1	0.046523
1layhid2.to.2layhid1	-0.29764
Intercept.to.2layhid2	-1.21032
1layhid1.to.2layhid2	1.922312
1layhid2.to.2layhid2	0.306559
Intercept.to.y	-0.49449
2layhid1.to.y	-0.20411
2layhid2.to.y	1.97079

nn$weights

Weights	Final Values
W_1A	0.8360114
W_1B	0.4505089
W_2A	1.2513425
W_2B	0.2355793
W_3A	0.8510065
W_3B	0.2729971
W_4A	0.5992144
W_4B	0.2339646
W_0A	-0.8698283
W_AC	0.04652263
W_AD	1.922312

W_0B	0.4148067
W_BC	-0.29764154
W_BD	0.306559
W_0C	-0.56810022
W_CZ	-0.2041074
W_0D	-1.210320
W_DZ	1.9707904
W_0Z	-0.4944936

nn$response

Predicted y	Actual y
0.43232	0.51
0.6054704	0.68
0.4961026	0.52
0.681639	0.75
0.3924518	0.45
0.5059119	0.41
0.6164064	0.43
0.4075457	0.47
0.588539	0.66
0.5100204	0.54
0.6902865	0.56
0.6670288	0.84
0.5915714	0.69
0.5647863	0.5
0.4415837	0.27
0.6591599	0.5
0.6142728	0.39
0.6866821	0.69
0.5258144	0.34
0.6512591	0.61
0.6814465	0.7
0.6559931	0.53
0.6313622	0.65
0.7027865	0.78
0.4701083	0.54
0.4153917	0.24
0.5545563	0.68
0.678102	0.79

0.5970299	0.57
0.3913403	0.14
0.6444901	0.8
0.6695461	0.51
0.4877302	0.63
0.4495037	0.45
0.6737581	0.75
0.6445068	0.76
0.5102506	0.58
0.5375437	0.59
0.3409057	0.5
0.5674986	0.65
0.6442892	0.57
0.6951318	0.65
0.5548011	0.41
0.6136304	0.39
0.5983255	0.62
0.6380193	0.73
0.5019952	0.48
0.6287658	0.79
0.3850688	0.46
0.6276271	0.77