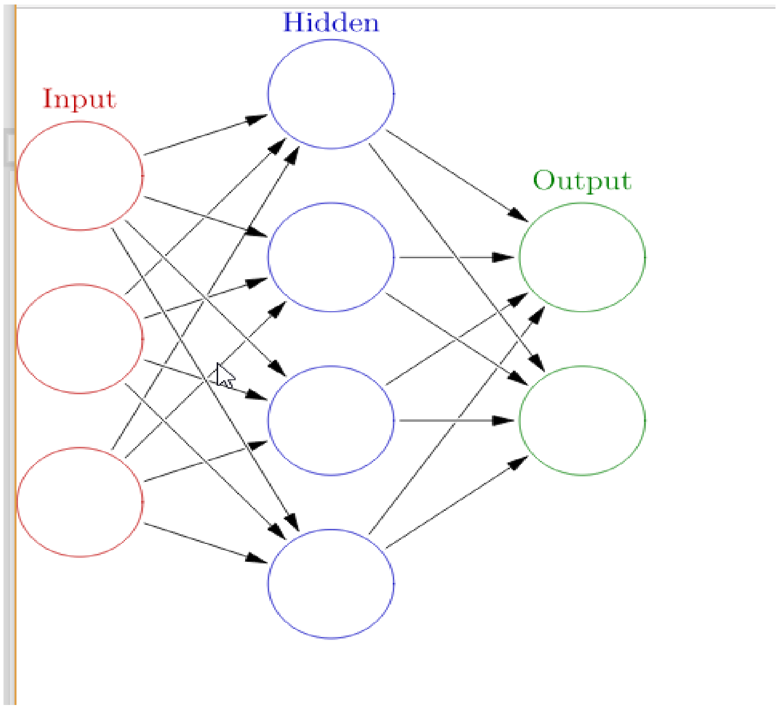
MACHINE LEARNING

Febin Zachariah – 800961027

SUPERVISED LEARNING-NEURAL NETWORKS

Neural Networks

It can be defined as a computing system made up of number of simple, highly interconnected processing elements which process information by their dynamic state response to external inputs. This approach is inspired by the biological nervous systems, the way by in which the brain computes information. Neural Networks are typically organized in layers. Layers are made up of number of interconnected nodes (like neurons in brain) which contains an activation function. Inputs data is given into neural network through the input layer and it interacts with other hidden layers to perform the actual processing. Then, hidden layers will transfer the output to the output layers. All the connections between input, hidden and output layers are interconnected by using weighted connections. Different learning rules are used in performing the neural network calculations. Most common among the learning rules is Backpropagation learning rule. In Backpropagation, forward stimulation is used to reset weights on the "front" neural units and this is sometimes done in combination with training where the correct result is known. Neural networks are used in speech recognition, computer vision and many other fields. **Following figure represents an example of neural network:**



Libraries Used

This project implements Neural Network algorithm by using the following libraries:

- **neuralnet:** This library performs the training of neural networks using backpropagation, resilient backpropagation with or without weight backtracking or the modified globally convergent version. The package allows flexible settings through custom-choice of error and activation function. Furthermore, the calculation of generalized weights is implemented.
  Ref: https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf
- **nnet:** We have used this library to extract the class variables from the output column to create different columns for each class variable.
  Ref: https://cran.r-project.org/web/packages/nnet/nnet.pdf

We have mainly used 3 functions to implement neural network algorithm in our project.

- **neuralnet ():** This function is used to train the data and create the corresponding Neural Network. Many arguments are used to get an accurate neural network and some of the relevant arguments are listed below:
  - formula: a symbolic description of the model to be fitted.
  - data: a data frame containing the variables specified in formula.
  - hidden: a vector of integers specifying the number of hidden neurons (vertices) in each layer.
  - threshold: a numeric value specifying the threshold for the partial derivatives of the error function as stopping criteria.
  - stepmax: the maximum steps for the training of the neural network. Reaching this maximum leads to a stop of the neural network's training process.
  - rep: the number of repetitions for the neural network's training.
  - learningrate: a numeric value specifying the learning rate used by traditional backpropagation. Used only for traditional backpropagation

The output of the neuralnet method is stored into an 'nn' object. The out object consists of two parameters which are relevant here: net.result (a list containing the overall result of the neural network for each replication); result.matrix (a matrix containing error, reached threshold, steps taken, estimated weights, AIC, BIC for each replication. Each column represents one replication).

- **compute ():** It is used Compute the outputs of all neurons for specific arbitrary covariate vectors given a trained neural network. We should make sure that the order of the covariates is the same in the new matrix or dataframe as in the original neural network**.** It automatically redefines the structure of the given neural network and calculates the output for arbitrary covariate combinations.
- **plot ():** It is designed for an inspection of the weights for objects of class nn, typically produced by neuralnet. The result of this method represents the plot of a trained neural network including trained synaptic weights and basic information about the training process.

Optical Recognition of Handwritten Digits Dataset

1) Introduction

- The dataset for this assignment is taken from **UCI Machine Learning repository**.
- It has handwritten digits from 43 people. 30 contributed for training set and 13 for test data set. The data taken for this assignment is already preprocessed.
- We are using Neural Network to learn the data by using training data and predict the results of test data.

2) Implementation

- For the implementation of decision tree: **R language is used**
- Packages used: **nnet, neuralnet.**
- Load the packages by using **library** command in R if it is already installed, otherwise install the libraries by using **install. packages** command.
- After that load the training data by using **read.csv** command.
- Divide the training into two sets in the ratio 80:20 for creating a cross validation data sample.
- Training data is added with one column for each class variable by using class.ind function from the nnet library
- Create the formula which shows the dependency of input attributes and output attributes.

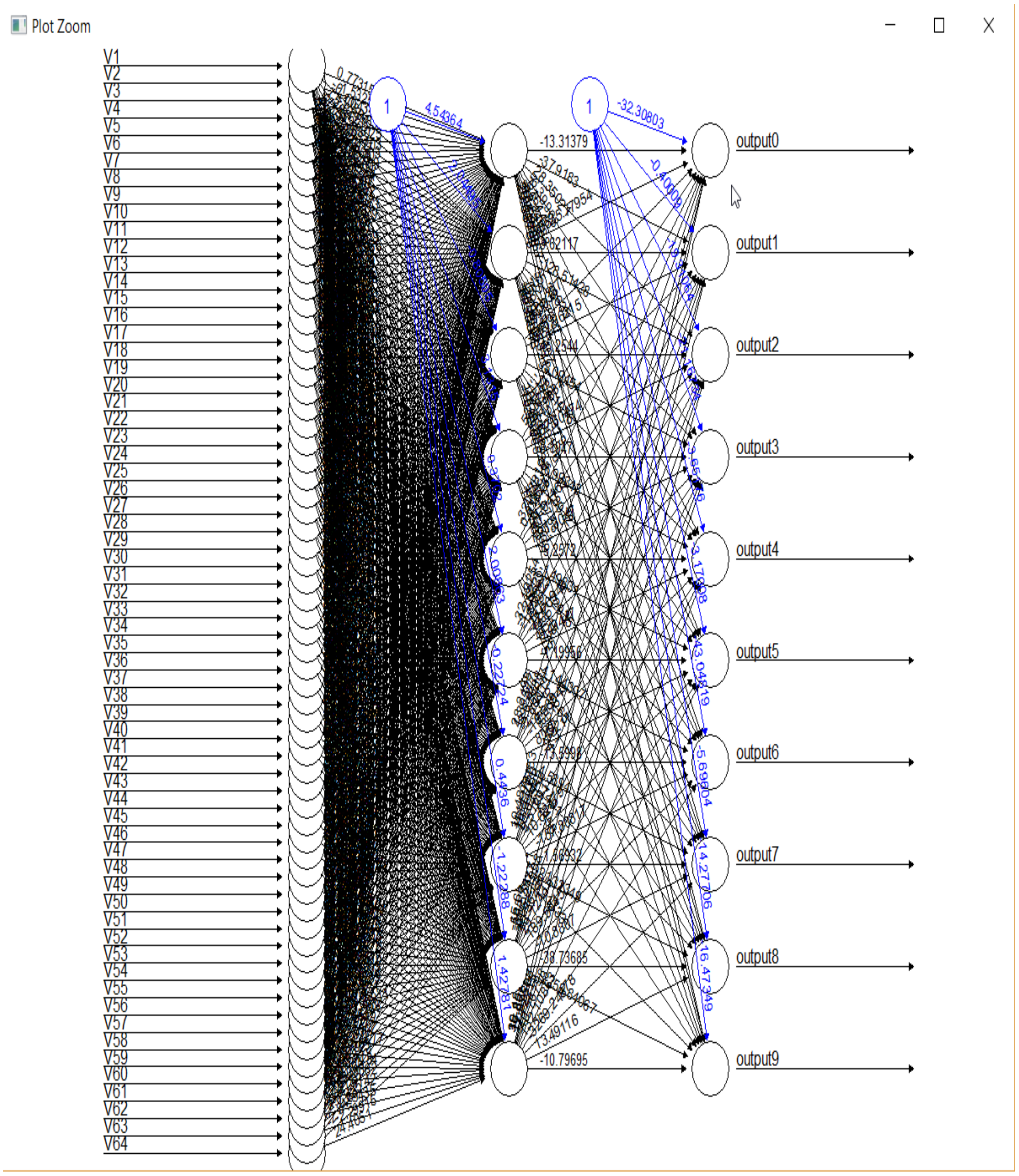
```
• output0 + output1 + output2 + output3 + output4 + output5 + outpu
  t6 +
•     output7 + output8 + output9 ~ V1 + V2 + V3 + V4 + V5 + V6 +
•     V7 + V8 + V9 + V10 + V11 + V12 + V13 + V14 + V15 + V16 +
•     V17 + V18 + V19 + V20 + V21 + V22 + V23 + V24 + V25 + V26 +
•     V27 + V28 + V29 + V30 + V31 + V32 + V33 + V34 + V35 + V36 +
•     V37 + V38 + V39 + V40 + V41 + V42 + V43 + V44 + V45 + V46 +
•     V47 + V48 + V49 + V50 + V51 + V52 + V53 + V54 + V55 + V56 +
•     V57 + V58 + V59 + V60 + V61 + V62 + V63 + V64
```

- Now create the Neural Net Work model by using neuralnet function from neuralnet library.
  nn_model <- neuralnet (fmla, data = trainset, hidden=10, threshold = 0.01, linear.output = F)

- Now, use the above decision tree to predict the values on cross validation set of data and verify our decision tree.
  outtest = compute(nn_model, test_set[,1:64])
- Use this decision tree to predict the output for the test data by using the predict command.
  outtest = compute(nn_model, test_set[,1:64])

3) Results and Observations

- **Neural Network Model**



4) Conclusions

- Successfully implemented Decision tree algorithm on R language by using neuralnet and nnet library
- Obtained an accuracy of 90% on the test data.
- Performed analysis on the data by splitting the data in different ratios.

Amazon Reviews Sentiment Analysis Dataset

1) INTRDUCTION

- This dataset consists of Amazon baby product reviews a subset of a larger amazon review collection.
- The dataset is split into training and testing subsets.
- Training data consists of **146824 observations** and test data has **36707 examples.**

2) IMPLEMENTATION

- For the implementation of decision tree: **R language is used**
- Packages used: **nnet, neuralnet, plyr, tm**
- Load the packages by using **library** command in R if it is already installed, otherwise install the libraries by using **install. packages** command.
- Load the training data and split into two subsets in the ratio 80%:20% for creating the
- Since **reviews are categorical data**, we have implemented an algorithm to convert reviews into numerical score based on the following calculation. Before calculating the score, we have used **tm library for Stemming and lemmatization** of the data.
  Score= Sum (Positive words)-Sum (Negative words)
- After getting the score for each review, we are creating additional columns to store score, number of positive words, Number of negative words.
- Create the formula which shows the dependency of input attributes and output attributes.

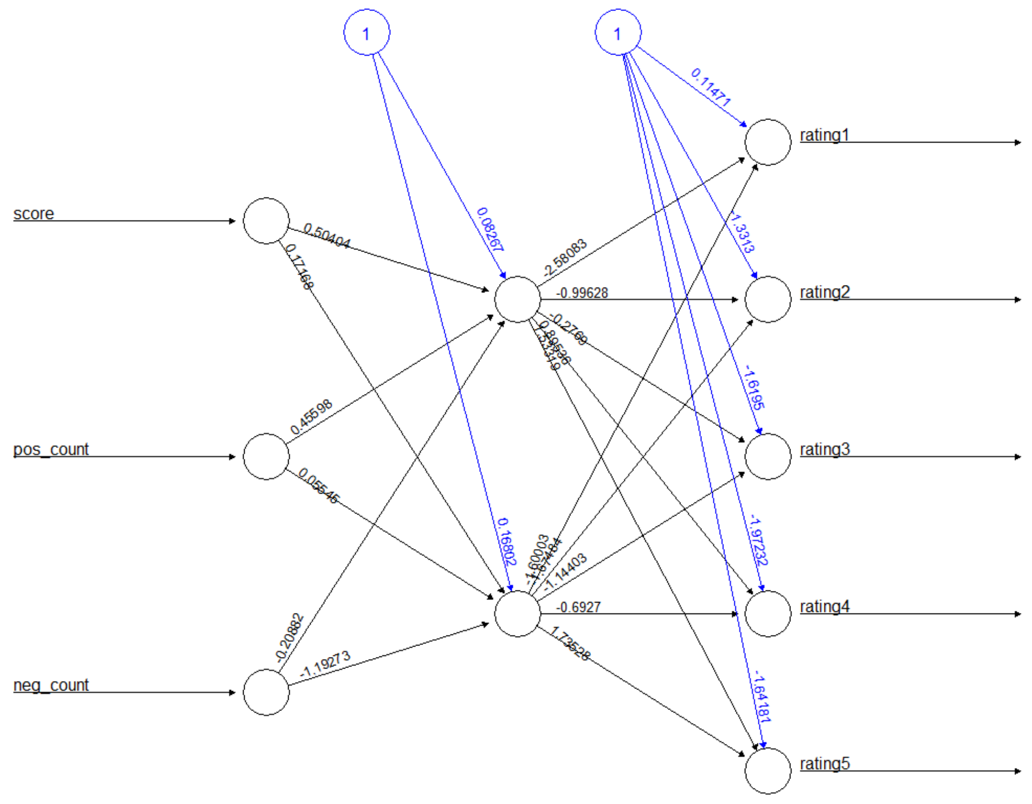
```
•  rating1 + rating2 + rating3 + rating4 + rating5 ~ score + pos_count +
•      neg_count
```

- Now create the Neural Net Work model by using neuralnet function from neuralnet library.
  nn_model <- neuralnet(fmla, data = trainset, hidden=10, threshold = 0.01, linear.output = F)

- Now, use the above decision tree to predict the values on cross validation set of data and verify our decision tree.
  outtest = compute(nn_model, test_set[,1:64])
- Use this decision tree to predict the output for the test data by using the predict command.
  outtest = compute(nn_model, test_set[,1:64])

3) Result and Observations

- **Neural Network**



4) Conclusions

- Implemented neural networks resilient backpropagation with backtracking weights algorithm to predict the rating of a review from the test data set.
- There was no improvement in the accuracy of prediction compared to decision tree algorithm (There is a significant improvement for digit recognition data using neural network algorithm over decision tree algorithm).
- On further analysis, I feel there is a need to improve the function which calculates the sentiment score for better results.
- We are now investigating on it to improve the efficiency of the sentimental analysis of the reviews further.

References:

Collaborated With: Ashwin Venkatesh prabhu

https://www.youtube.com/watch?v=JNtcslLZYsY&t=3135s

https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf

https://cran.r-project.org/web/packages/nnet/nnet.pdf

https://www.youtube.com/watch?v=ZepOMlAjuB4

https://www.r-bloggers.com/fitting-a-neural-network-in-r-neuralnet-package/

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html