Square Roots Neural Network

Jeff Shelton Okang'a

23/07/2020

Introduction

In this document, we will develop a neural network model that will learn from a training data set of number 0 to 805. We will use a comma separated value file that contains two columns of, numbers and their respective square roots. We will visualize the model on different levels of hidden layers and determine the most accurate model. We will use the trained model to then predict the efficiency of the model in a real world scenario with the numbers 806 to 810.

Implementing the Neural Network

Data Preparation

In this section of the document we will show how we import and explore the nature of our data set.

```
theDataset <- read.csv("SquareRoots.csv")
```

Adding our preferred columns to the dataset

```
names(theDataset) <- c('Number', 'SQRT')
```

After importing the data set and naming its columns, it is necessary that we explore the nature of our data set to see what we're working with.

```
head(theDataset)
```

```
##   Number     SQRT
## 1      0 0.000000
## 2      1 1.000000
## 3      2 1.414214
## 4      3 1.732051
## 5      4 2.000000
## 6      5 2.236068
```

```
tail(theDataset)
```

```
##     Number     SQRT
## 801    800 28.28427
## 802    801 28.30194
## 803    802 28.31960
## 804    803 28.33725
## 805    804 28.35489
## 806    805 28.37252
```

From what we have observed, our data set has the columns Number and SQRT(for squareroots) and 806 records. Now that we are well aware of what we're working with we can proceed to create the neural network model. The data set has a single feature and a label, the Number is the feature and the SQRT is the label.

Creating the Neural Network Model

The neural network model we are going to create is of the form $SQRT \sim Number$. The model is to learn from a training set(our current data set 0 to 805) and we test it against a new dat set of 806 to 810. To test the benefits of using different layers we will create 5 different neural network models to represent different layers. However we first need to imoprt the neuralnet library to utilise the neuralnet function to build the model.

```
library(neuralnet)
```

```
## Warning: package 'neuralnet' was built under R version 4.0.2
```

Creating the different layered models.

```
neuron_1<-neuralnet(SQRT~Number, data=theDataset, hidden=2, act.fct="logistic", linear.output = FALSE)
neuron_2<-neuralnet(SQRT~Number, data=theDataset, hidden=3, act.fct="logistic", linear.output = FALSE)
neuron_3<-neuralnet(SQRT~Number, data=theDataset, hidden=10, act.fct="logistic", linear.output = FALSE)
neuron_4<-neuralnet(SQRT~Number, data=theDataset, hidden=15, act.fct="logistic", linear.output = FALSE)
neuron_5<-neuralnet(SQRT~Number, data=theDataset, hidden=20, act.fct="logistic", linear.output = FALSE)
```

Discussion on layers

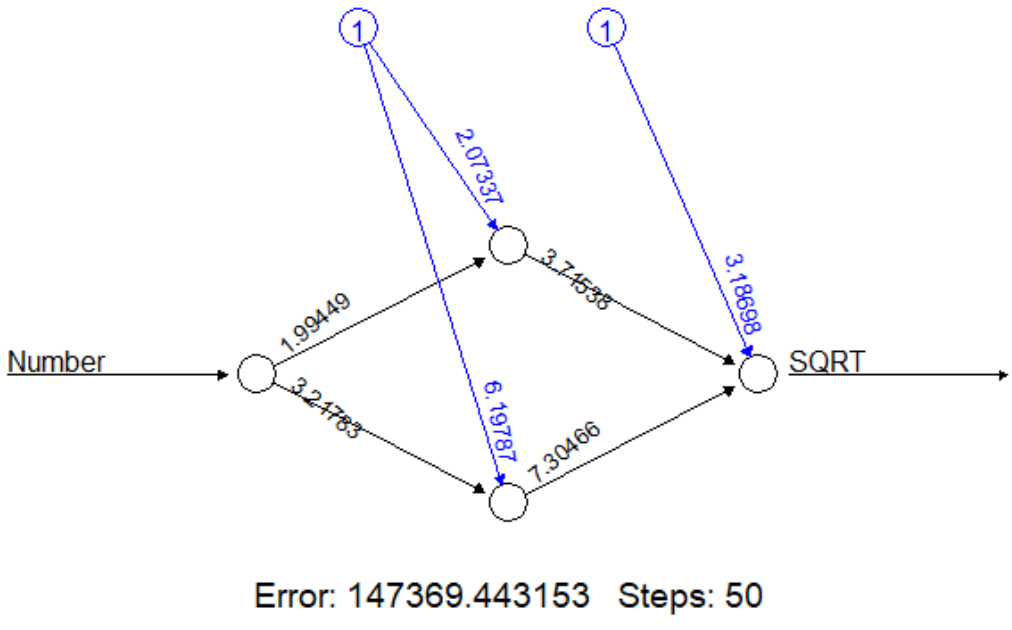
The amount of layers we use on our model will determine how well our model performs on unseen data(the testDataset). A higher number of hidden layers could result to the model being overfit and only perform well on the training set and poorly on the test set.

Visualizing the models

We will have to plot the neural network models created above to see the hidden layers.

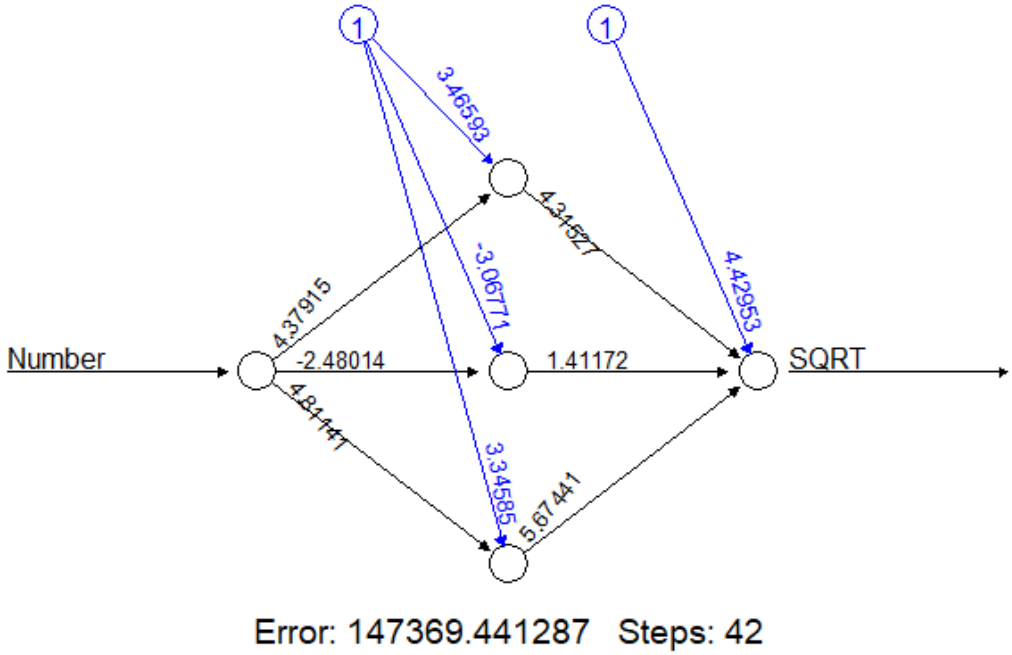
The first model with two hidden layers:

```
plot(neuron_1)
```



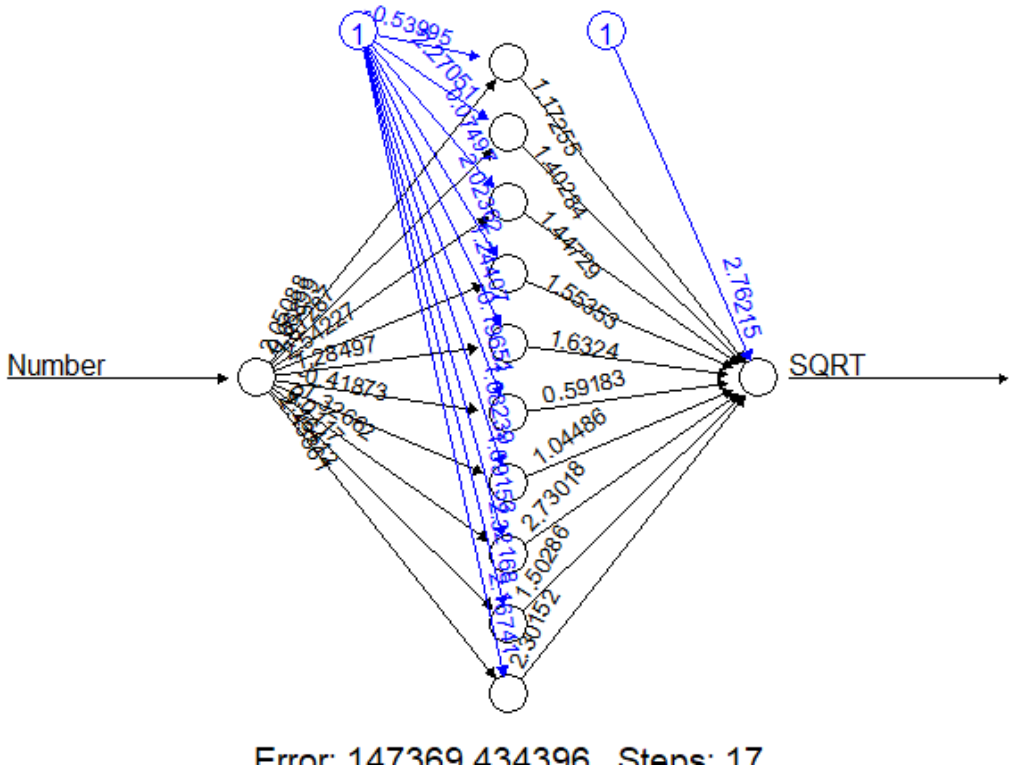
The second model with three hidden layers:

```
plot(neuron_2)
```



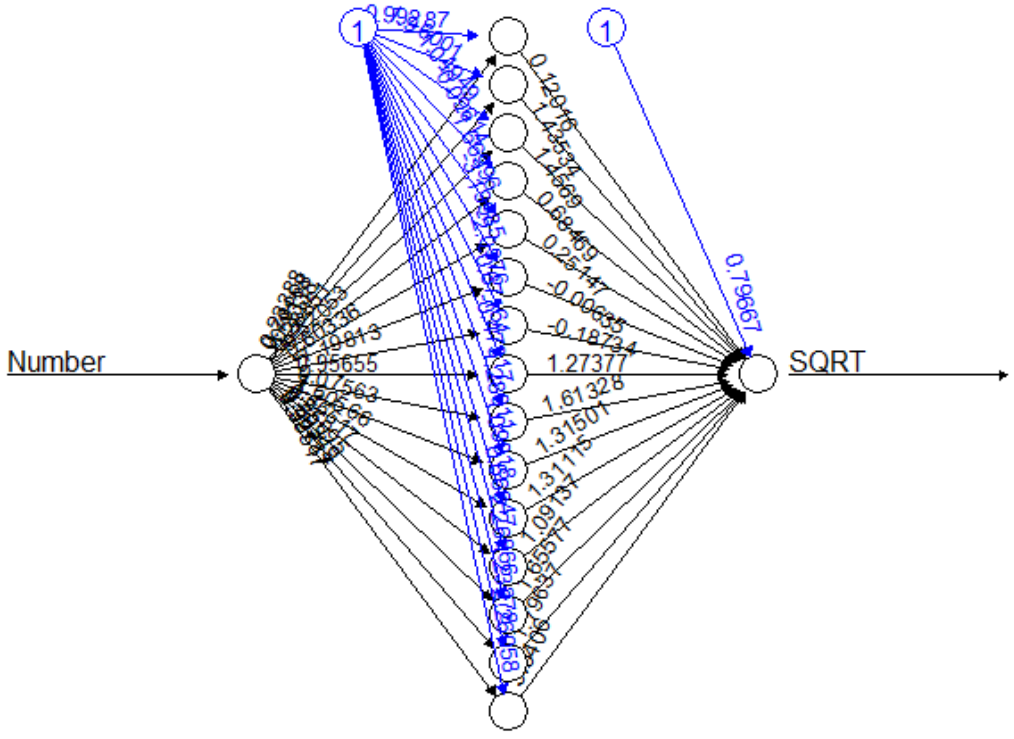
The third model with ten hidden layers:

```
plot(neuron_3)
```



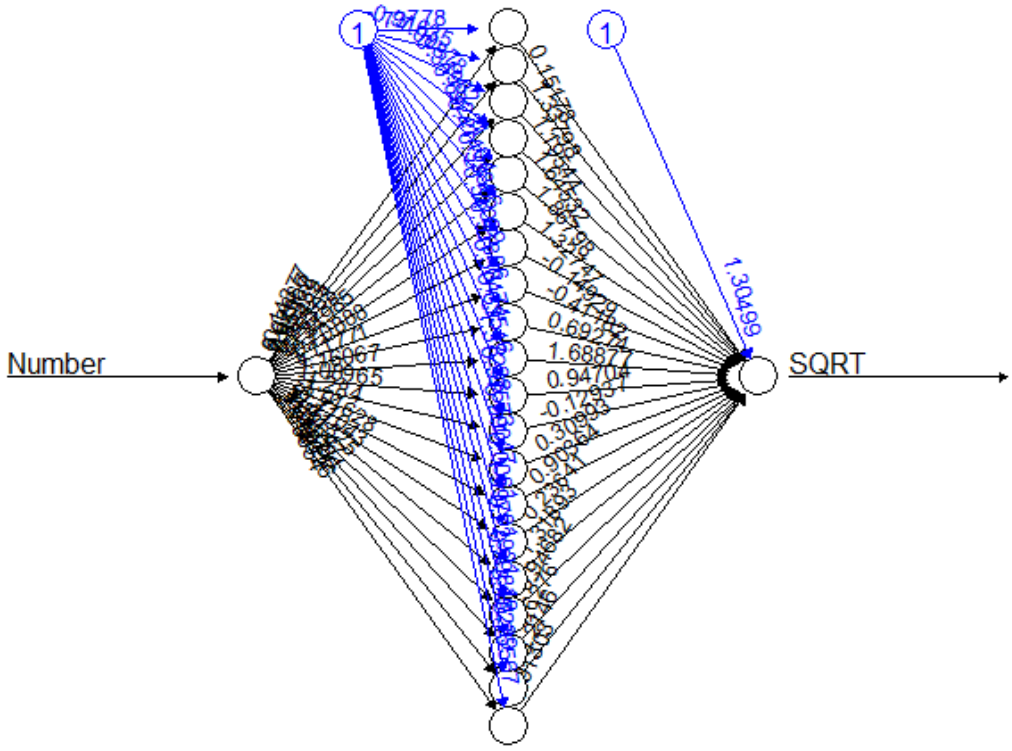
The fourth model with fifteen hidden layers:

```
plot(neuron_4)
```



The fifth model with twenty five hidden layers:

```
plot(neuron_5)
```



Testing the Model

We will test the data set using numbers 806 t0 810

```
Number<-c(806:810)
testData<- data.frame(Number, SQRT=sqrt(Number))
testData
```

```
##   Number     SQRT
## 1    806 28.39014
## 2    807 28.40775
## 3    808 28.42534
## 4    809 28.44293
## 5    810 28.46050
```

Now we use our 5 models and fit it on the test set and see how it performs compared to the actual results of the square roots of the numbers 806 to 810. The following will showw the accuracies of the varried models:

First Model Results

```
predict<-compute(neuron_1,testData)
predict$net.result
```

```
##           [,1]
## [1,] 0.9999993
## [2,] 0.9999993
## [3,] 0.9999993
## [4,] 0.9999993
```

```
## [5,] 0.9999993
```

Second Model Results

```
predict<-compute(neuron_2,testData)
predict$net.result
```

```
##           [,1]
## [1,] 0.9999993
## [2,] 0.9999993
## [3,] 0.9999993
## [4,] 0.9999993
## [5,] 0.9999993
```

Third Model Results

```
predict<-compute(neuron_3,testData)
predict$net.result
```

```
##           [,1]
## [1,] 0.9999997
## [2,] 0.9999997
## [3,] 0.9999997
## [4,] 0.9999997
## [5,] 0.9999997
```

Fourth Model Results

```
predict<-compute(neuron_4,testData)
predict$net.result
```

```
##           [,1]
## [1,] 0.9999995
## [2,] 0.9999995
## [3,] 0.9999995
## [4,] 0.9999995
## [5,] 0.9999995
```

Fifth Model Results

```
predict<-compute(neuron_5,testData)
predict$net.result
```

```
##           [,1]
## [1,] 0.9999993
## [2,] 0.9999993
## [3,] 0.9999993
## [4,] 0.9999993
## [5,] 0.9999993
```

Conclusion

From the tests we've done with our model,we can assert that our model performs very well when tested against unseen data getting very high accuracy percentages on the test data of numbers from 806 to 810.

The model performs exceptionally across all different hidden layers with minimal and negligible change across the layers of 2,3,10,15 and 20.