ANN

2024-07-30

Installing required packages

Here, I am installing the necessary packages and loading them.

```r
install.packages(c('neuralnet', 'kera','tensorflow'), dependencies = T)
```

```
## Installing packages into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)

## Warning: package 'kera' is not available for this version of R
##
## A version of this package for your version of R might be available elsewhere,
## see the ideas at
## https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages
```

```r
install.packages("tidyverse")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
library(neuralnet)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::compute() masks neuralnet::compute()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

Data and processing

Here we get the data set for the model. We are converting the iris data set column into a factor. This is important for the neural network.

```r
iris<-iris%>%mutate_if(is.character, as.factor)
iris_sample <- iris[sample(nrow(iris), 10), ]
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
```

```
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

Splitting the data

Now we split the data into training and testing. We will calculate the the number of rows that represent 80% of the dataset. Then randomly select the indices that will be used for training. Then, split the data.

Finally, we print the head for the test and train datasets.

```r
set.seed(254)
data_rows<-floor(0.80*nrow(iris))
data_rows
```

```
## [1] 120
```

```r
train_indices<-sample(c(1:nrow(iris)), data_rows)
train_indices
```

```
##   [1]  55  37 146  70  45 124  20  76 144   3  88  10 136 126 102 125  64 111
##  [19] 122  32 147 123  95 101 149 143  94 150  11  83  54  57  61  48  29  69
##  [37] 130 115 145  17  50  96  35  93  49  12  14  60  18  97 109 134  62 113
##  [55]  75 119  41  27  25  89 100  91  19 137  46 103  85   6  44  86  71  36
##  [73] 104  42 139 118 106   9  43  84  66  39   7  72 117 108   4  38 138  65
##  [91]   5   2  87  82  40  77 128  67  92 131  74  56  59 120  23  13  33 107
## [109] 127  24 116  34  68  58  73  80   8  99 121 133
```

```r
train_data<-iris[train_indices, ]
head(train_data)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 55           6.5         2.8          4.6         1.5 versicolor
## 37           5.5         3.5          1.3         0.2     setosa
## 146          6.7         3.0          5.2         2.3  virginica
## 70           5.6         2.5          3.9         1.1 versicolor
## 45           5.1         3.8          1.9         0.4     setosa
## 124          6.3         2.7          4.9         1.8  virginica
```

```r
test_data<-iris[-train_indices, ]
head(test_data)
```

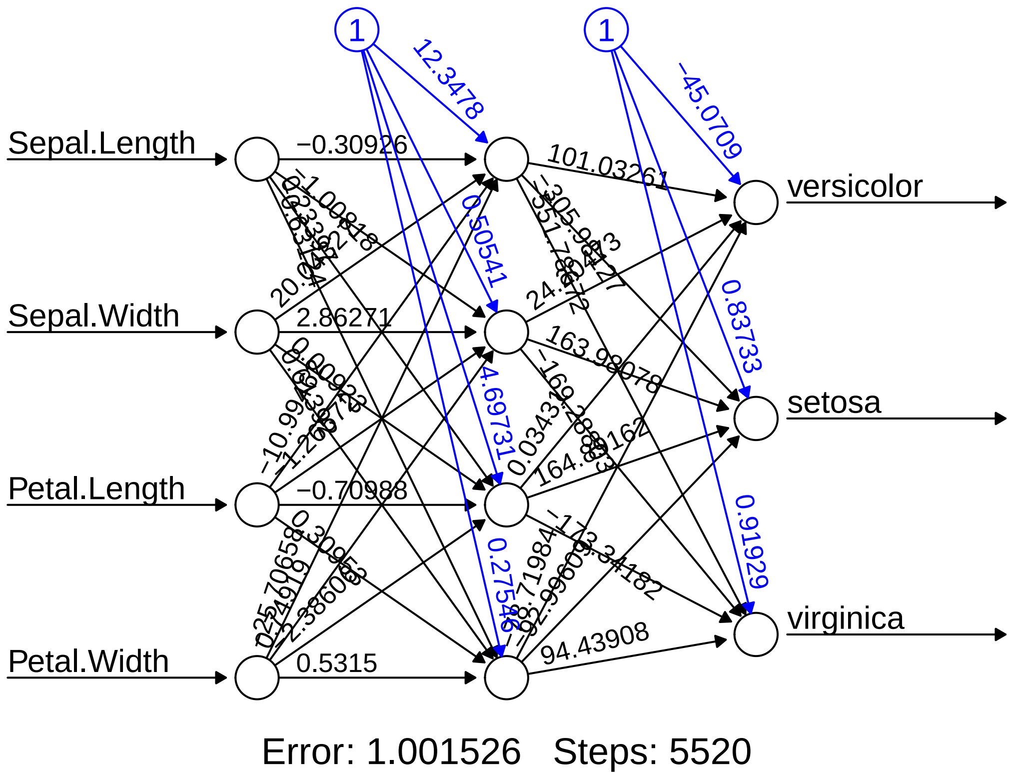
```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5          1.4         0.2  setosa
## 15          5.8         4.0          1.2         0.2  setosa
## 16          5.7         4.4          1.5         0.4  setosa
## 21          5.4         3.4          1.7         0.2  setosa
## 22          5.1         3.7          1.5         0.4  setosa
## 26          5.0         3.0          1.6         0.2  setosa
```

ANN model

Now we create a neural network model that will predict the species based on the features in the dataset.

Once we create the model, we then plot it

```
model<-neuralnet(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data = train_data, l
plot(model, rep='best')
```



Evaluation

Now that the model has been created, we need to test and evaluate it, and see the accuracy

```
pred<-predict(model, test_data)
pred
```

```
##              [,1]         [,2]         [,3]
## 1   1.000000e+00 1.710135e-10 0.000000e+00
## 15  1.000000e+00 2.387817e-07 0.000000e+00
## 16  1.000000e+00 1.266236e-12 0.000000e+00
## 21  1.000000e+00 1.461364e-11 0.000000e+00
## 22  1.000000e+00 5.634459e-13 0.000000e+00
## 26  1.000000e+00 1.708495e-14 0.000000e+00
## 28  1.000000e+00 1.290007e-10 0.000000e+00
## 30  1.000000e+00 1.310230e-13 0.000000e+00
## 31  1.000000e+00 9.248574e-14 0.000000e+00
## 47  1.000000e+00 2.327558e-12 0.000000e+00
## 51  1.352465e-36 1.000000e+00 3.550262e-18
## 52  1.563677e-38 1.000000e+00 1.843794e-18
## 53  7.215021e-38 9.999999e-01 6.124045e-08
## 63  8.697065e-34 1.000000e+00 2.815586e-39
```

```
## 78  1.306723e-39 2.364684e-04 9.998535e-01
## 79  9.099311e-40 1.000000e+00 2.123716e-11
## 81  1.289886e-36 1.000000e+00 4.157010e-37
## 90  2.246086e-38 1.000000e+00 5.893895e-25
## 98  1.967700e-37 1.000000e+00 1.246661e-25
## 105 1.018099e-44 8.961082e-24 1.000000e+00
## 110 2.371386e-45 3.950570e-23 1.000000e+00
## 112 1.279637e-41 7.501421e-16 1.000000e+00
## 114 1.632900e-43 4.829642e-18 1.000000e+00
## 129 1.359505e-43 7.985934e-22 1.000000e+00
## 132 1.836300e-42 7.712546e-20 1.000000e+00
## 135 2.040331e-41 1.755987e-07 9.999999e-01
## 140 7.101872e-42 3.463004e-17 1.000000e+00
## 141 2.513769e-44 1.260010e-22 1.000000e+00
## 142 1.305158e-41 5.417382e-16 1.000000e+00
## 148 2.823033e-42 5.220259e-15 1.000000e+00
```

```r
labels<-c("setosa", "versicolor", "virginica")
labels
```

```
## [1] "setosa"     "versicolor" "virginica"
```

```r
prediction_label <- data.frame(max.col(pred)) %>% mutate(pred=labels[max.col.pred.]) %>% select(2) %>% u
table(test_data$Species, prediction_label)
```

```
##             prediction_label
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0          8         1
##   virginica       0          0        11
```

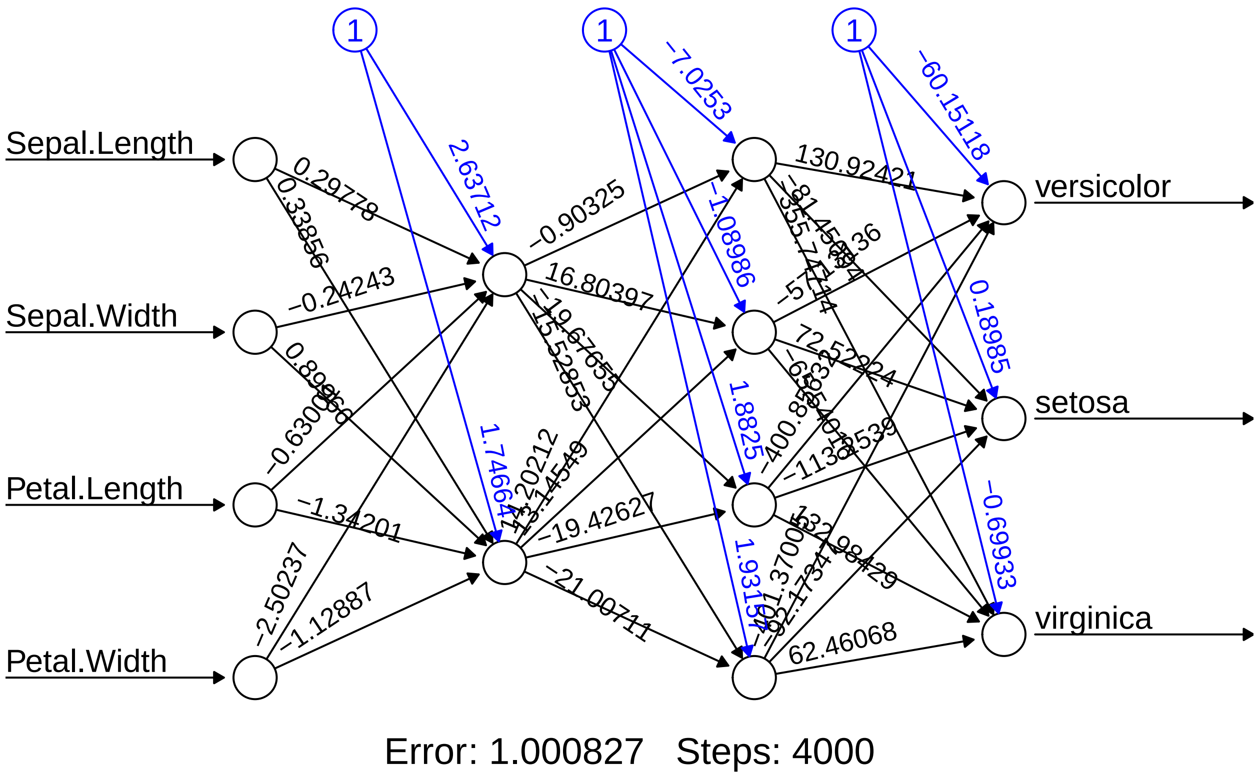
```r
check = as.numeric(test_data$Species) == max.col(pred)
accuracy = (sum(check)/nrow(test_data))*100
print(accuracy)
```

```
## [1] 96.66667
```

Layers

Now to see the effect the layers have. Currently, the model has one hidden layer with 4 neurons. But let's see different layers and how if affects the accuracy.

```r
model<-neuralnet(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data = train_data, h
plot(model, rep='best')
```




```r
pred<-predict(model, test_data)
labels<-c("setosa", "versicolor", "virginica")
prediction_label <- data.frame(max.col(pred)) %>% mutate(pred=labels[max.col.pred.]) %>% select(2) %>%
table(test_data$Species, prediction_label)
```

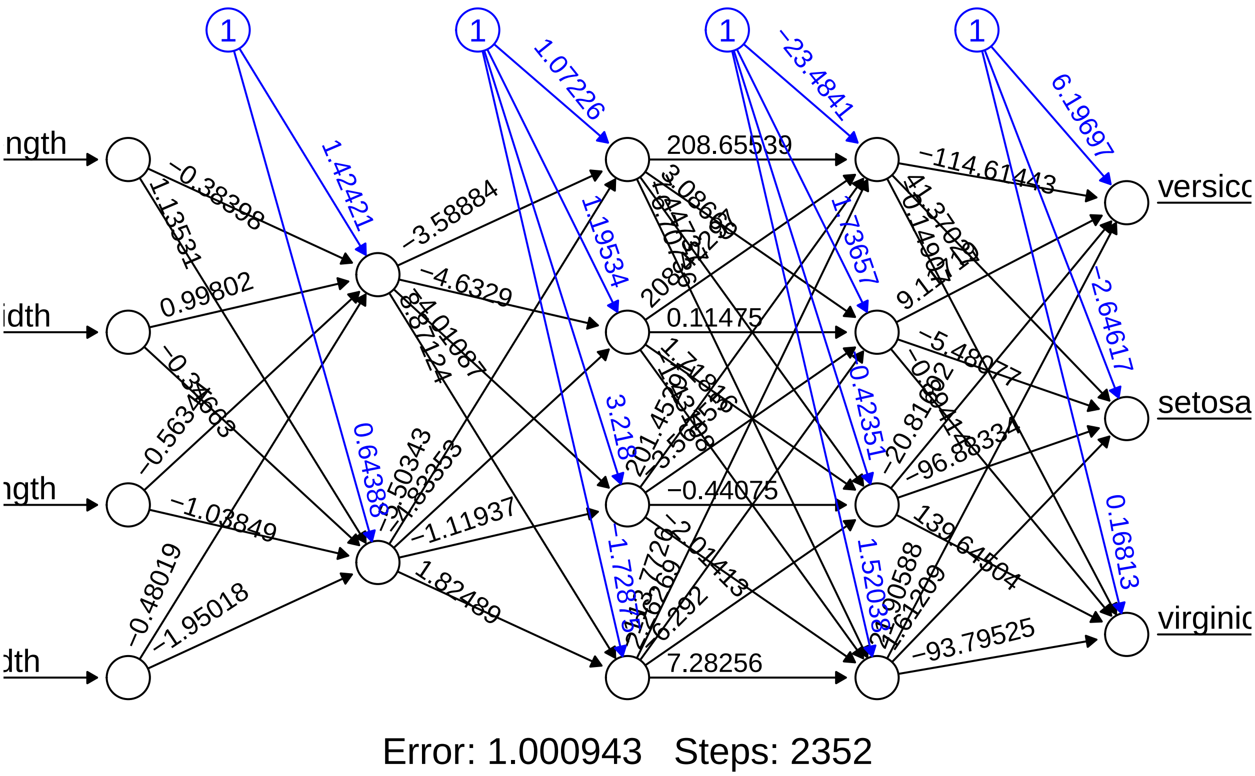
```
##             prediction_label
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0          9         0
##   virginica       0          0        11
```

```r
check = as.numeric(test_data$Species) == max.col(pred)
accuracy = (sum(check)/nrow(test_data))*100
print(accuracy)
```

```
## [1] 100
```

The above neural network has two hidden layers, with 2 and 4 neurons respectively

```r
model<-neuralnet(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data = train_data,
plot(model, rep='best')
```




```
pred<-predict(model, test_data)
labels<-c("setosa", "versicolor", "virginica")
prediction_label <- data.frame(max.col(pred)) %>% mutate(pred=labels[max.col.pred.]) %>% select(2) %>% u
table(test_data$Species, prediction_label)
```

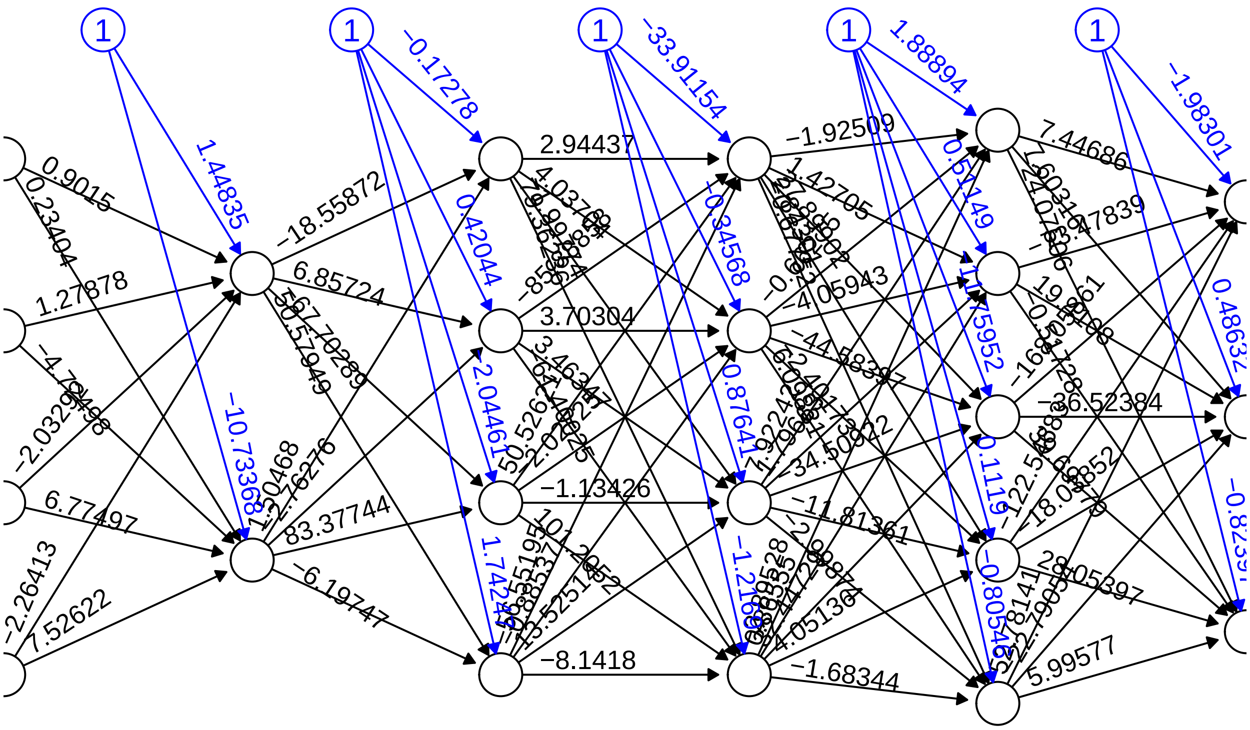
```
##             prediction_label
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0          9         0
##   virginica       0          0        11
```

```
check = as.numeric(test_data$Species) == max.col(pred)
accuracy = (sum(check)/nrow(test_data))*100
print(accuracy)
```

```
## [1] 100
```

The above neural network has three hidden layers, with 2, 4 and 4 neurons respectively

```
model<-neuralnet(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data = train_data, h
plot(model, rep='best')
```



Error: 1.000086   Steps: 2256

```r
pred<-predict(model, test_data)
labels<-c("setosa", "versicolor", "virginica")
prediction_label <- data.frame(max.col(pred)) %>% mutate(pred=labels[max.col.pred.]) %>% select(2) %>% u
table(test_data$Species, prediction_label)
```

```
##             prediction_label
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0          9         0
##   virginica       0          0        11
```

```r
check = as.numeric(test_data$Species) == max.col(pred)
accuracy = (sum(check)/nrow(test_data))*100
print(accuracy)
```

```
## [1] 100
```

The above neural network has three hidden layers, with 2, 4, 4 and 5 neurons respectively

Conclusion

Increasing the layers increases the accuracy for sure, but it also is unnecessary with this dataset. It is not that complex.