Artificial Neural Network Assigment

Muna Said 664331

2024-08-01

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

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

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

```
install.packages(c("neuralnet", "keras", "tensorflow"), dependencies = T)
```

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

```
library(neuralnet)
install.packages("tidyverse")
```

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

```
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 errors
```

```
iris<-iris %>%mutate_if(is.character, as.factor)
iris<-iris %>%mutate_if(is.character, as.factor)
sample_iris<-sample_n(iris,5)
sample_iris
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          6.7         3.3          5.7         2.1  virginica
## 2          6.1         2.8          4.0         1.3 versicolor
```

```
## 3          5.7         2.9          4.2         1.3 versicolor
## 4          6.8         3.2          5.9         2.3  virginica
## 5          4.7         3.2          1.6         0.2     setosa
```

```r
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  
##                 
##                 
## 
```

```r
# Train and test split
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, ]
sample_train_data<-sample_n(train_data,5)
sample_train_data
```

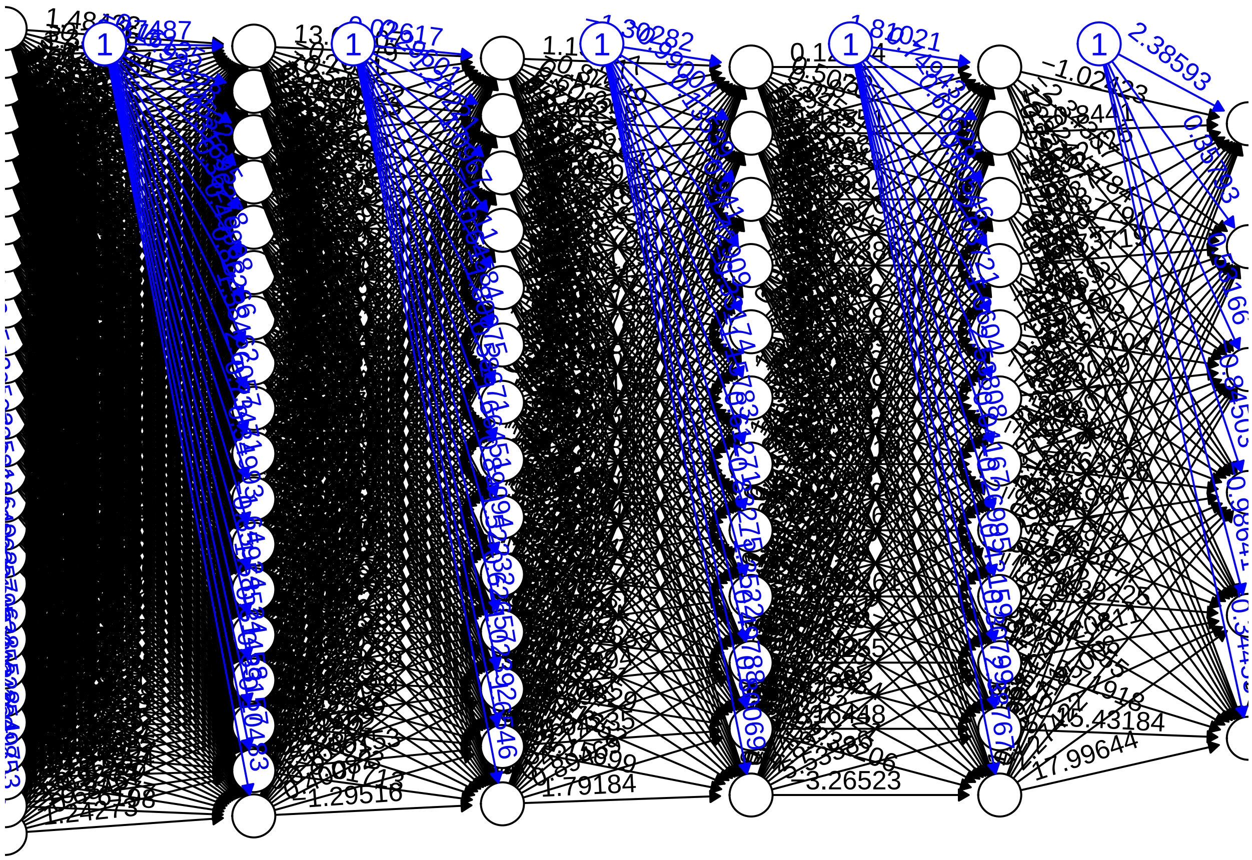
```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          5.0         2.0          3.5         1.0 versicolor
## 2          5.7         3.8          1.7         0.3     setosa
## 3          5.3         3.7          1.5         0.2     setosa
## 4          7.1         3.0          5.9         2.1  virginica
## 5          5.2         3.4          1.4         0.2     setosa
```

```r
test_data<-iris[-train_indices,]
sample_test_data<-sample_n(test_data,5)
sample_test_data
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          6.5         3.0          5.2         2.0  virginica
## 2          5.7         4.4          1.5         0.4     setosa
## 3          5.1         3.5          1.4         0.2     setosa
```

```
## 4          7.0         3.2          4.7         1.4 versicolor
## 5          6.7         3.1          5.6         2.4  virginica
```

```
#The plot of 30,18,14,12,12,6
model<-neuralnet( Species ~ Sepal.Length +Sepal.Width+Petal.Length +Petal.Width, data = train_data, hid
plot(model, rep = 'best')
```



```
# Model evaluation
#predict categories - test dataset
#list of category names
#dataframe
# table - actual and predicated
pred<-predict(model, test_data)

labels<-c("setosa", "versicolor","virginca")
labels
```

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

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

```
##             prediction_label
##              setosa versicolor virginca
##   setosa         10          0        0
##   versicolor      0          9        0
```

```
##   virginica       0          0       11
```

```r
summary(test_data)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.700   Min.   :2.200   Min.   :1.200   Min.   :0.200  
##  1st Qu.:5.425   1st Qu.:2.900   1st Qu.:1.600   1st Qu.:0.250  
##  Median :6.050   Median :3.100   Median :4.500   Median :1.400  
##  Mean   :6.043   Mean   :3.143   Mean   :3.867   Mean   :1.253  
##  3rd Qu.:6.650   3rd Qu.:3.475   3rd Qu.:5.275   3rd Qu.:2.000  
##  Max.   :7.900   Max.   :4.400   Max.   :6.400   Max.   :2.500  
##        Species  
##  setosa    :10  
##  versicolor: 9  
##  virginica :11  
##                 
##                 
## 
```

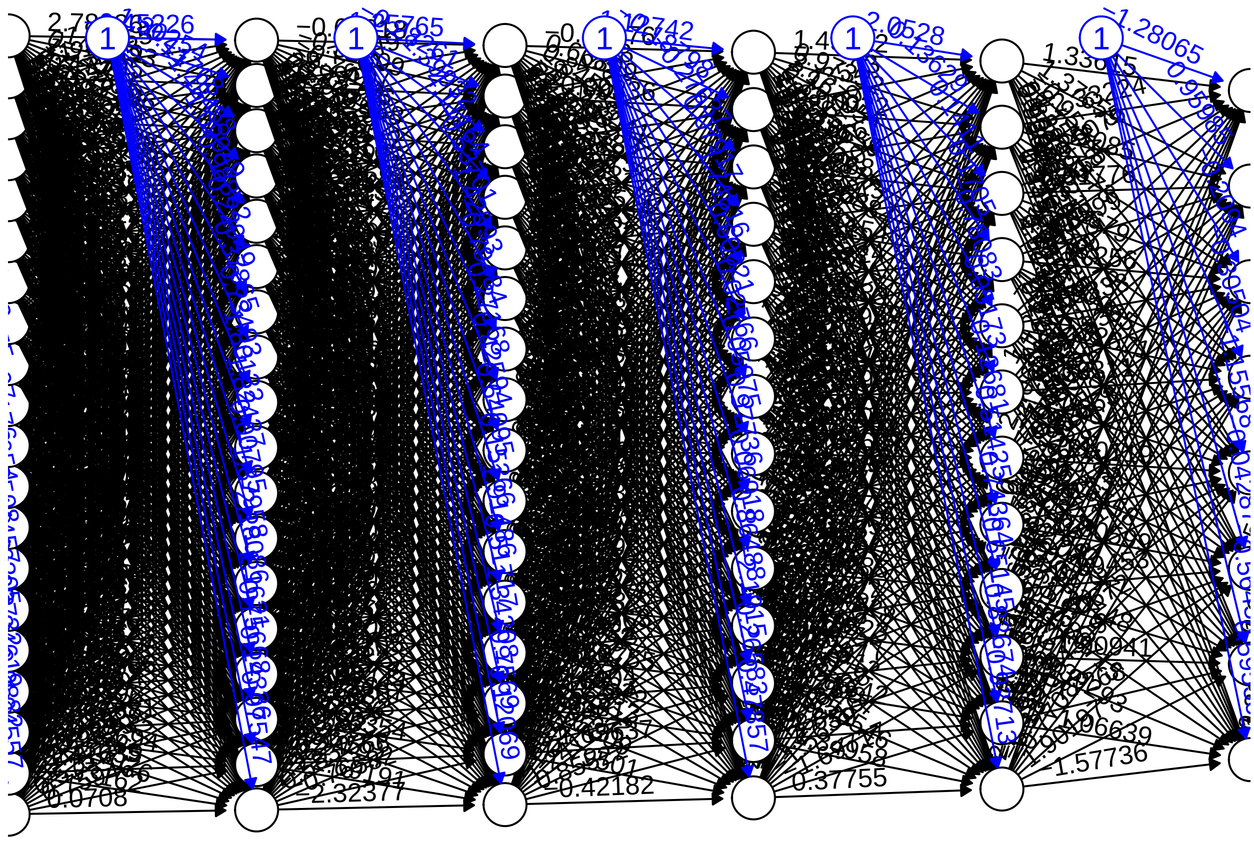
```r
check= as.numeric(test_data$Species) == max.col(pred)

accuracy<-(sum(check)/nrow(test_data))*100
print(accuracy)
```

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

```r
#The plot of 30,24,20,18,16,14,12,8,6,3
model<-neuralnet( Species ~ Sepal.Length +Sepal.Width+Petal.Length +Petal.Width, data = train_data, hid

plot(model, rep = 'best')
```



```r
#second test
# Model evaluation
#predict categories - test dataset
#list of category names
#dataframe
# table - actual and predicated

pred<-predict(model, test_data)

labels<-c("setosa", "versicolor","virginca")
labels
```

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

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

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

```r
summary(test_data)
```

```
##    Sepal.Length     Sepal.Width      Petal.Length     Petal.Width
```

```
##  Min.   :4.700   Min.   :2.200   Min.   :1.200   Min.   :0.200
##  1st Qu.:5.425   1st Qu.:2.900   1st Qu.:1.600   1st Qu.:0.250
##  Median :6.050   Median :3.100   Median :4.500   Median :1.400
##  Mean   :6.043   Mean   :3.143   Mean   :3.867   Mean   :1.253
##  3rd Qu.:6.650   3rd Qu.:3.475   3rd Qu.:5.275   3rd Qu.:2.000
##  Max.   :7.900   Max.   :4.400   Max.   :6.400   Max.   :2.500
##        Species
##  setosa    :10
##  versicolor: 9
##  virginica :11
##
##
##
```

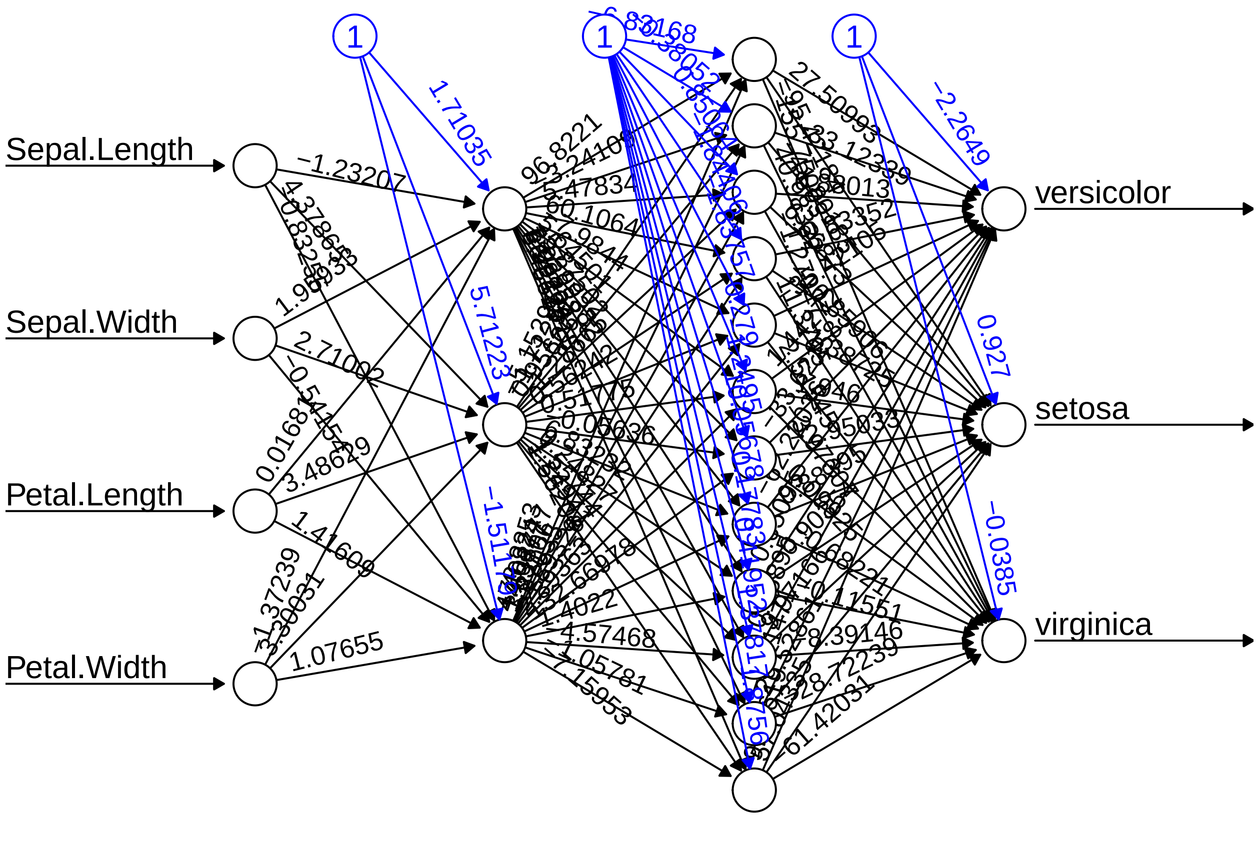
```
check= as.numeric(test_data$Species) == max.col(pred)

accuracy<-(sum(check)/nrow(test_data))*100
print(accuracy)
```

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

```
#third test
#The plot of 3,12
model<-neuralnet( Species ~ Sepal.Length +Sepal.Width+Petal.Length +Petal.Width, data = train_data, hid

plot(model, rep = 'best')
```



```r
# Model evaluation
#predict categories - test dataset
#list of category names
#dataframe
# table - actual and predicated

pred<-predict(model, test_data)

labels<-c("setosa", "versicolor","virginca")
labels
```

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

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

```
##             prediction_label
##              setosa versicolor virginca
##   setosa         10          0        0
##   versicolor      0          9        0
##   virginica       0          1       10
```

```r
summary(test_data)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.700   Min.   :2.200   Min.   :1.200   Min.   :0.200
##  1st Qu.:5.425   1st Qu.:2.900   1st Qu.:1.600   1st Qu.:0.250
##  Median :6.050   Median :3.100   Median :4.500   Median :1.400
##  Mean   :6.043   Mean   :3.143   Mean   :3.867   Mean   :1.253
##  3rd Qu.:6.650   3rd Qu.:3.475   3rd Qu.:5.275   3rd Qu.:2.000
##  Max.   :7.900   Max.   :4.400   Max.   :6.400   Max.   :2.500
##        Species
##  setosa    :10
##  versicolor: 9
##  virginica :11
##
##
##
```

```r
check= as.numeric(test_data$Species) == max.col(pred)

accuracy<-(sum(check)/nrow(test_data))*100
print(accuracy)
```

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

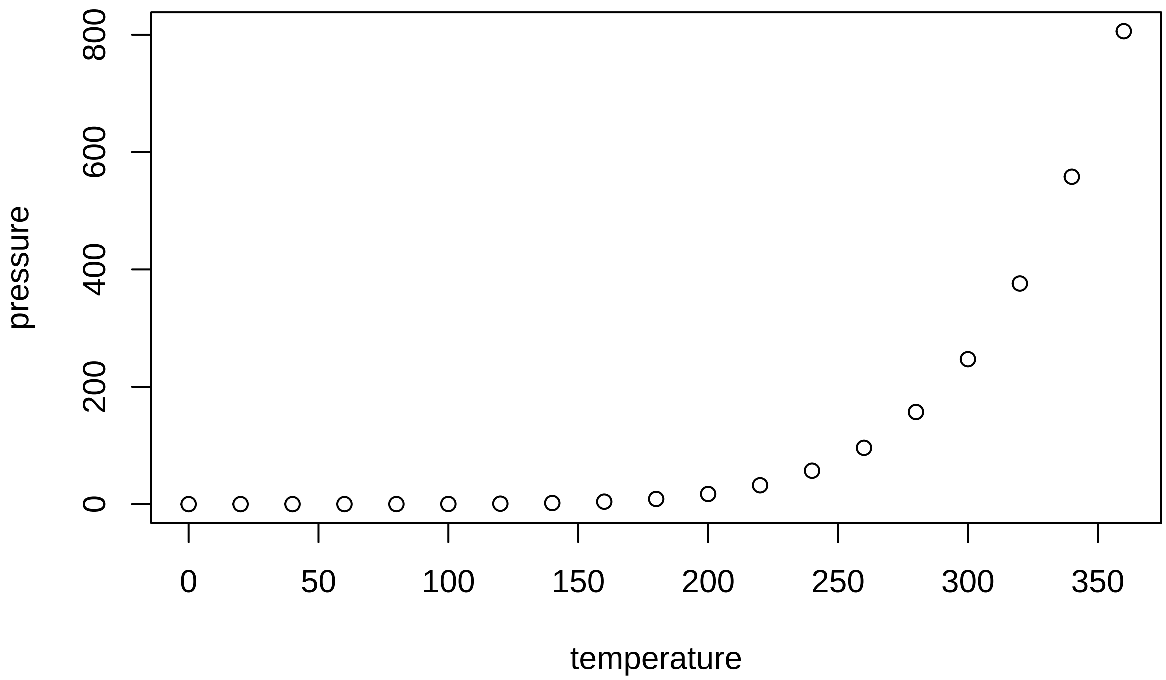
Including Plots

Configuration	Accurcay
c(30,18,14,12,12,6)	100
c(30,24,20,18,16,14,12,8,6,3)	100
c(3,12)	96

Analysis Layer Depth: Using more layers allows it to learn more complex patterns. For the Iris dataset, which is relatively simple, deeper networks (like the ones in Configurations 1 and 2) did a great job without running into overfitting issues. meaning they were able to understand the data well without getting too specific or memorizing it.

Layer Configuration: How you arrange the layers and their sizes impacts how well the network learns. Networks with lots of layers and varying sizes (like Configuration 2) can capture detailed and complex patterns. On the other hand, simpler setups (like Configuration 3) can also work well, showing that you don't always need a lot of layers to achieve great results. I have noted that increasing the number of layers in a neural network can enhance its ability to learn complex features but it also brings challenges such as overfitting. Alternatively decreasing the number of layers simplifies the model, which may improve generalization and reduce the risk of overfitting, but it may limit the model's learning capacity as shown with the c(3,12) with 96% accuracy.

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.