assignment

Bonam

2024-07-31

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:

```r
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:

```r
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)
```

```r
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)
```

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

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

```r
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)
ris<-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          6.1         2.9          4.7         1.4 versicolor
## 2          5.0         3.4          1.5         0.2     setosa
## 3          5.7         3.0          4.2         1.2 versicolor
## 4          7.7         2.8          6.7         2.0  virginica
## 5          5.1         3.7          1.5         0.4     setosa
```

```
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  
##                 
##                 
##                 
```

```
# Train and test split
set.seed(254)
data_rows<-floor(0.80 * nrow(iris))
data_rows
```

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

```
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
```

```
train_data<-iris[train_indices, ]
sample_train_data<-sample_n(train_data,5)
sample_train_data
```

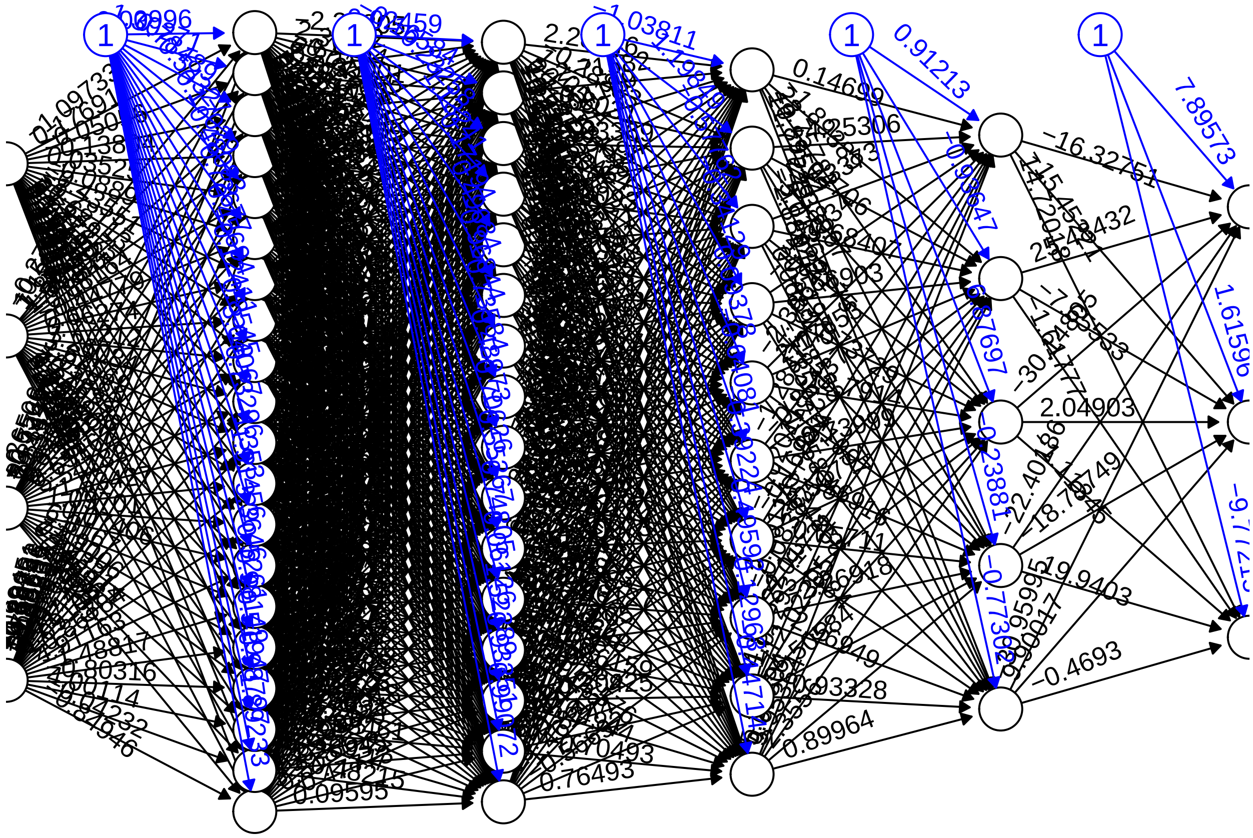
```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          5.0         2.0          3.5         1.0 versicolor
## 2          5.7         3.8          1.7         0.3     setosa
## 3          5.3         3.7          1.5         0.2     setosa
## 4          7.1         3.0   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

```
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 20,16,14,12,10,5
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)
```

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

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

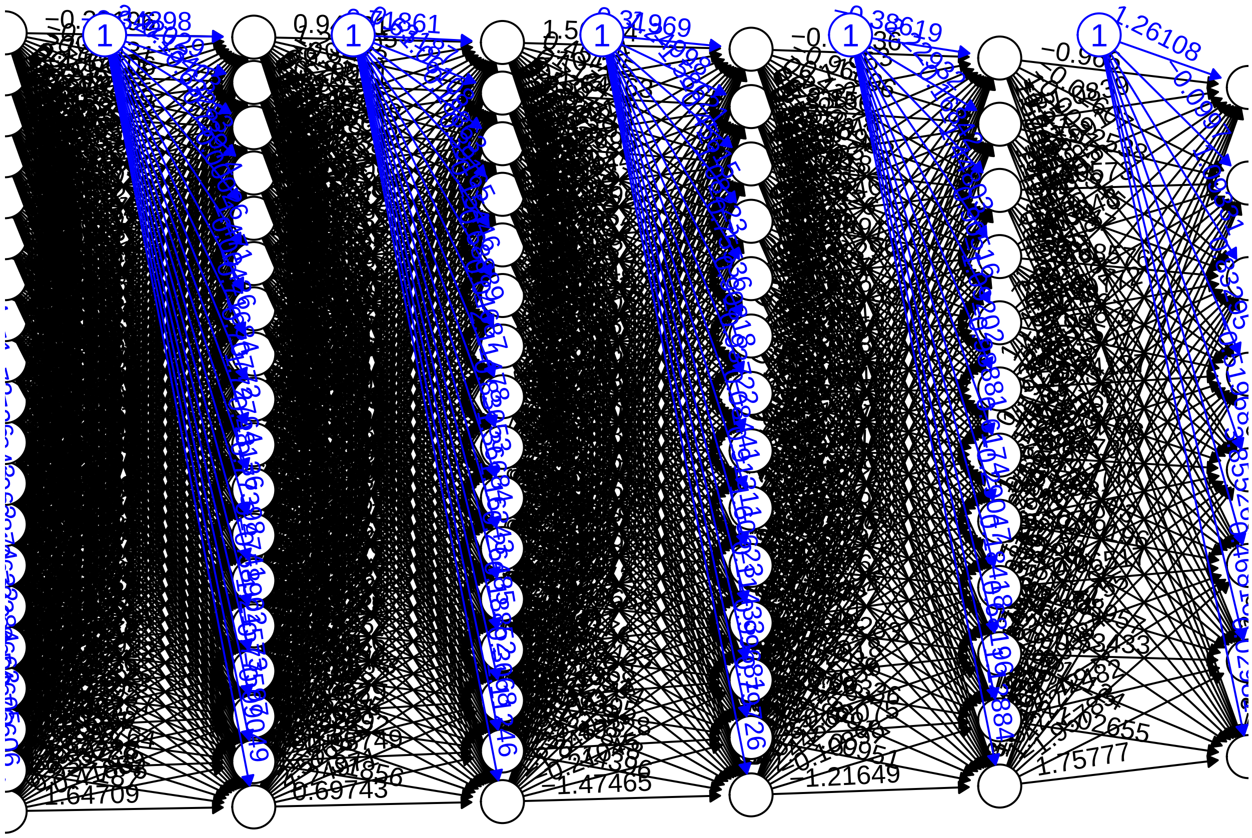
```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

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

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

```r
#for the second test with configuration of c(30,24,20,18,16,14,12,8,6,3)
model<-neuralnet( Species ~ Sepal.Length +Sepal.Width+Petal.Length +Petal.Width, data = train_data, hidd

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



```
#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"
```

```
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
```

```
summary(test_data)
```

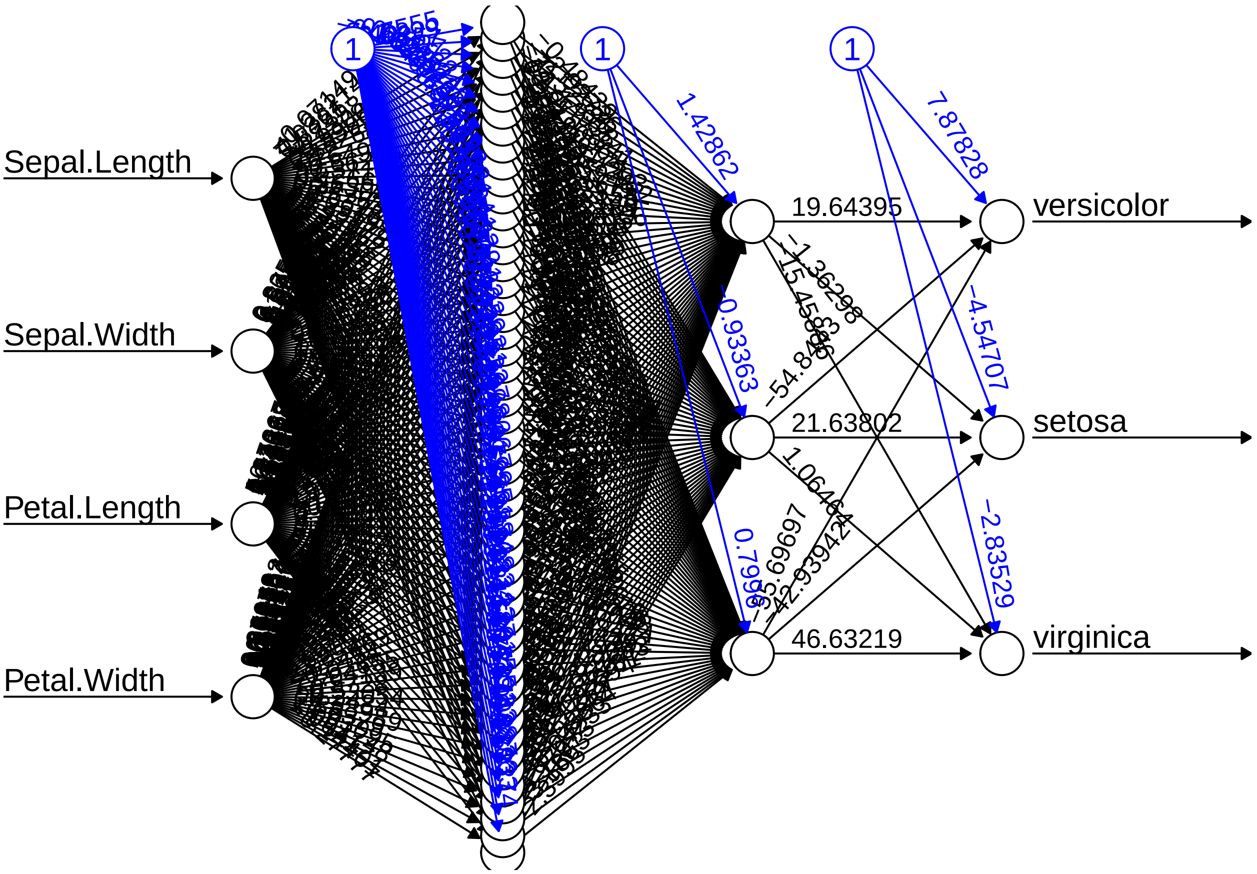
```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.700   Min.   :2.200   Min.   :1.200   Min.   :0.200
##  1st Qu.:5.425   1st Qu.:2.900   1st Qu.:1.600   1st Qu.:0.250
##  Median :6.050   Median :3.100   Median :4.500   Median :1.400
##  Mean   :6.043   Mean   :3.143   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] 96.66667
```

```
#The plot of 50,3
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
```

```
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
```

configuration	accuracy
c(50,3)	100%
c(20,16,10,5)	100%
c(30,24,20,18,16,14,12,8,6,3)	96.66%

My analysis suggests that increasing the number of hidden layers of the model decreases the accuracy. This is because for relatively simpler problems like this one putting a lot of hidden layers leads to overfitting. Adding more hidden layers will only increase the accuracy for complex problems but will reduce the accuracy for simpler problems because of overfitting. My accuracy score was 100 for c(50,3), 100% for c(20,16,10,5) and 96.66% for c(30,24,20,18,16,14,12,8,6,3).

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