ANN Assignment

Delphine Fanwi

2024-07-31

R Markdown

http://rmarkdown.rstudio.com.

```r
# Set CRAN mirror
options(repos = c(CRAN = "https://cran.rstudio.com/"))

# Install necessary packages
install.packages(c('neuralnet','keras','tensorflow'), dependencies = TRUE)
```

```
## Installing packages into 'C:/Users/FANWI/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)

## package 'neuralnet' successfully unpacked and MD5 sums checked
## package 'keras' successfully unpacked and MD5 sums checked
## package 'tensorflow' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\FANWI\AppData\Local\Temp\RtmpgDvK8z\downloaded_packages
```

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

```
## Installing packages into 'C:/Users/FANWI/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)

## package 'neuralnet' successfully unpacked and MD5 sums checked
## package 'keras' successfully unpacked and MD5 sums checked
## package 'tensorflow' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\FANWI\AppData\Local\Temp\RtmpgDvK8z\downloaded_packages
```

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

```
## Installing package into 'C:/Users/FANWI/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)

## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\FANWI\AppData\Local\Temp\RtmpgDvK8z\downloaded_packages
```

```r
# Load libraries
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
```

```r
# Ensure iris dataset is loaded and convert character columns to factors
data(iris)
iris <- iris %>% mutate_if(is.character, as.factor)

# Display summary of the dataset
summary(iris)
```

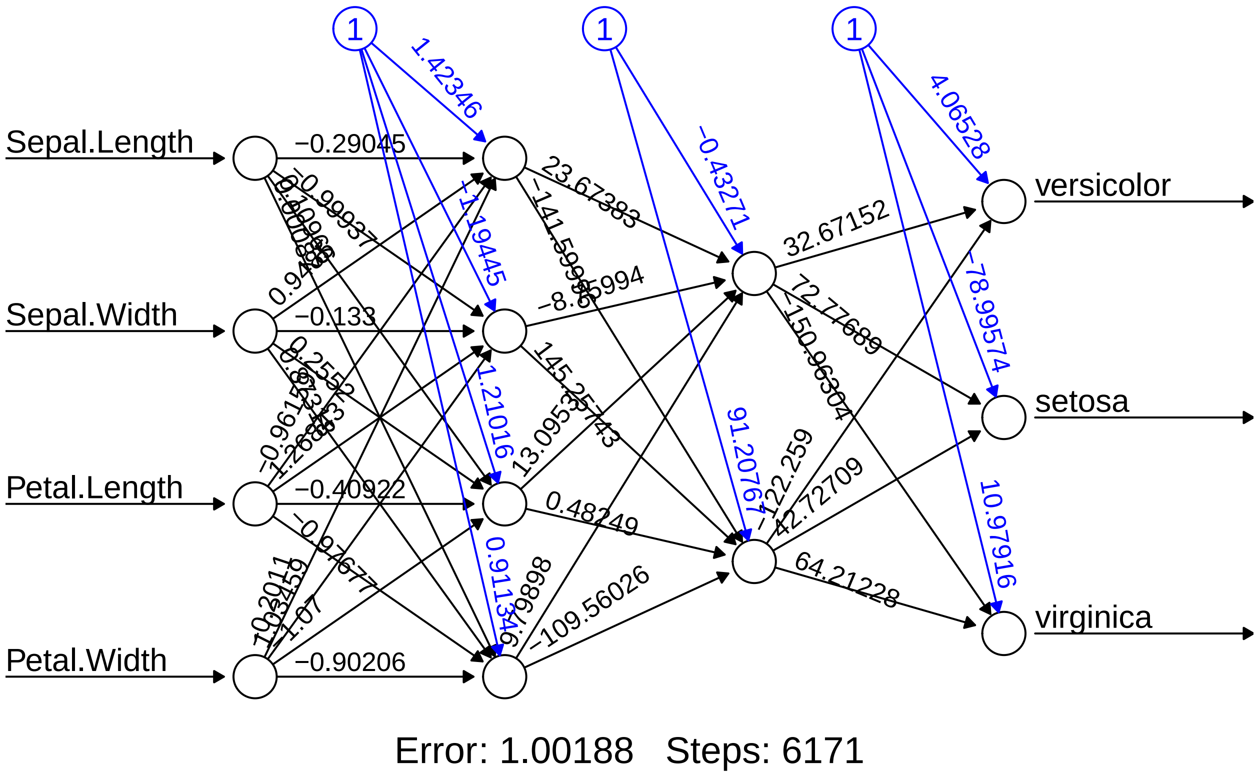
```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300  
##  Mean   :5.843   Mean   :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-Test Split
# Set seed for reproducibility
set.seed(254)
# Calculate number of rows for training data (80%)
data_rows <- floor(0.80 * nrow(iris))
# Generate random indices for training data
train_indices <- sample(1:nrow(iris), data_rows)
# Split the data into training and testing sets
train_data <- iris[train_indices, ]
test_data <- iris[-train_indices, ]

# Model Training
# Create the neural network model
```

```r
model <- neuralnet(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
                   data = train_data, hidden = c(4, 2), linear.output = FALSE)

# Display the model summary
#print(model)
# Plot the neural network
plot(model, rep = 'best')
```




```r
# Model Evaluation
# Predict species for the test dataset
pred <- predict(model, test_data)

# List of category names
labels <- c("setosa", "versicolor", "virginica")

# Convert predictions to category names
prediction_label <- data.frame(max.col(pred)) %>%
  mutate(pred = labels[max.col(pred)]) %>%
  select(pred) %>%
  unlist()

table(test_data$Species, prediction_label)
```

```
##               prediction_label
##                setosa versicolor virginica
```

```
##   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
```

##TRYING 3 MORE HIDDEN LAYERS

```r
# Load libraries
library(neuralnet)
library(tidyverse)

# Ensure iris dataset is loaded and convert character columns to factors
data(iris)
iris <- iris %>% mutate_if(is.character, as.factor)

# Display summary of the dataset
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-Test Split
# Set seed for reproducibility
set.seed(254)
# Calculate number of rows for training data (80%)
data_rows <- floor(0.80 * nrow(iris))
# Generate random indices for training data
train_indices <- sample(1:nrow(iris), data_rows)
# Split the data into training and testing sets
train_data <- iris[train_indices, ]
test_data <- iris[-train_indices, ]

# Define a function to train and evaluate a neural network model

train_and_evaluate <- function(hidden_layers) {
   set.seed(254)  # Ensure reproducibility within the function
  model <- neuralnet(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
                     data = train_data, hidden = hidden_layers, linear.output = FALSE)
  pred <- predict(model, test_data)

  # List of category names
  labels <- c("setosa", "versicolor", "virginica")

  # Convert predictions to category names
  prediction_label <- data.frame(max.col(pred)) %>%
    mutate(pred = labels[max.col(pred)]) %>%
    select(pred) %>%
    unlist()

  # Calculate accuracy
  check <- as.numeric(test_data$Species) == max.col(pred)
  accuracy <- (sum(check) / nrow(test_data)) * 100
  return(accuracy)
}

# Train and evaluate models with different hidden layer configurations
hidden_layers_list <- list(c(2,1 ), c(6,4 ),c(50,20))
accuracy_results <- sapply(hidden_layers_list, train_and_evaluate)

# Print the accuracy results for each configuration
for (i in 1:length(hidden_layers_list)) {
  cat("Hidden Layers:", hidden_layers_list[[i]], "-> Accuracy:", accuracy_results[i], "%\n")
}
```

```
## Hidden Layers: 2 1 -> Accuracy: 63.33333 %
## Hidden Layers: 6 4 -> Accuracy: 100 %
```

```
## Hidden Layers: 50 20 -> Accuracy: 100 %

##   Hidden_Layers  Accuracy
## 1           2-1  63.33333
## 2           6-4 100.00000
## 3         50-20 100.00000
```

Analysis of the result

Performance and Processing Time:

(2, 1): Underfitting resulted in a lower accuracy (63.33%) The model is simply not expressive enough to encode the intricacies of your data.Least amount of parameters thus the fastest training time.

(6, 4): Perfect accuracy. It is simple to use and captures many of the patterns within the iris dataset.When using this configuration, the training time increases compared to the (2, 1) setup, but it remains reasonable due to the relatively small size of the iris dataset.

(50,20): Well- accurate analysis measures depicts just perfect better accuracy (right) of 100%. The model is effective, but it has the potential to be overkill and slightly more at risk for overfitting on larger or richer datasets.This setup demands significantly more computational resources and results in a longer training period because of the large number of parameters involved.