DSE 6211 Module 04 Lab 04

Joseph Annand

2024-02-10

## Load libraries

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(caret)

## Warning: package 'caret' was built under R version 4.3.2

## Loading required package: ggplot2

## Loading required package: lattice

## Import data

data <- read.csv("lab\_4\_data/lab\_4\_data.csv")  
training\_ind <- createDataPartition(data$lodgepole\_pine,  
 p = 0.75,  
 list = F,  
 times = 1)

## Data Pre-processing

training\_set <- data[training\_ind, ]  
test\_set <- data[-training\_ind, ]  
  
top\_20\_soil\_types <- training\_set %>%  
 group\_by(soil\_type) %>%  
 summarise(count = n()) %>%  
 arrange(desc(count)) %>%  
 select(soil\_type) %>%  
 top\_n(20)

## Selecting by soil\_type

training\_set$soil\_type <- ifelse(training\_set$soil\_type %in% top\_20\_soil\_types$soil\_type,  
 training\_set$soil\_type,  
 "other")  
  
training\_set$wilderness\_area <- factor(training\_set$wilderness\_area)  
training\_set$soil\_type <- factor(training\_set$soil\_type)  
  
  
onehot\_encoder <- dummyVars(~ wilderness\_area + soil\_type,  
 training\_set[, c("wilderness\_area", "soil\_type")],  
 levelsOnly = T,  
 fullRank = T)  
  
onehot\_enc\_training <- predict(onehot\_encoder,  
 training\_set[, c("wilderness\_area", "soil\_type")])  
  
training\_set <- cbind(training\_set, onehot\_enc\_training)  
  
  
test\_set$soil\_type <- ifelse(test\_set$soil\_type %in% top\_20\_soil\_types$soil\_type,  
 test\_set$soil\_type,  
 "other")  
  
test\_set$wilderness\_area <- factor(test\_set$wilderness\_area)  
test\_set$soil\_type <- factor(test\_set$soil\_type)  
  
onehot\_enc\_test <- predict(onehot\_encoder, test\_set[, c("wilderness\_area", "soil\_type")])  
test\_set <- cbind(test\_set, onehot\_enc\_test)  
  
test\_set[, -c(11:13)] <- scale(test\_set[, -c(11:13)],  
 center = apply(training\_set[, -c(11:13)], 2, mean),  
 scale = apply(training\_set[, -c(11:13)], 2, sd))  
training\_set[, -c(11:13)] <- scale(training\_set[, -c(11:13)])

## Create R arrays for neural network

training\_features <- array(data = unlist(training\_set[, -c(11:13)]),  
 dim = c(nrow(training\_set), 33))  
training\_labels <- array(data = unlist(training\_set[, 13]),  
 dim = c(nrow(training\_set)))  
  
test\_features <- array(data = unlist(training\_set[, -c(11:13)]),  
 dim = c(nrow(test\_set), 33))  
test\_labels <- array(data = unlist(training\_set[, 13]),  
 dim = c(nrow(test\_set)))

## Load libraries and Tensorflow workspace

library(reticulate)

## Warning: package 'reticulate' was built under R version 4.3.2

library(tensorflow)

## Warning: package 'tensorflow' was built under R version 4.3.2

##   
## Attaching package: 'tensorflow'

## The following object is masked from 'package:caret':  
##   
## train

library(keras)

## Warning: package 'keras' was built under R version 4.3.2

use\_virtualenv("my\_tf\_workspace")

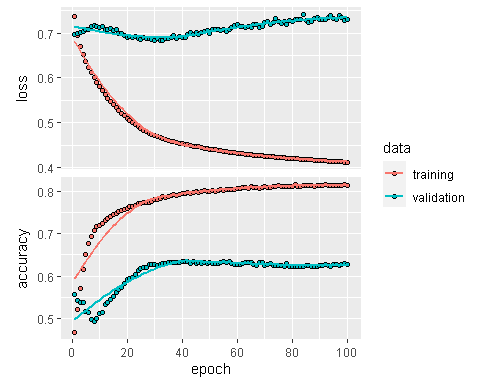
## Dense Feed-forward Neutral Network

model <- keras\_model\_sequential(list(  
 layer\_dense(units = 20, activation = "relu"),  
 layer\_dense(units = 10, activation = "relu"),  
 layer\_dense(units = 1, activation = "sigmoid")))  
  
compile(model,  
 optimizer = "rmsprop",  
 loss = "binary\_crossentropy",  
 metrics = "accuracy")  
  
history <- fit(model, training\_features, training\_labels,  
 epochs = 100, batch\_size = 512, validation\_split = 0.33)

## Exercises

### Question 1

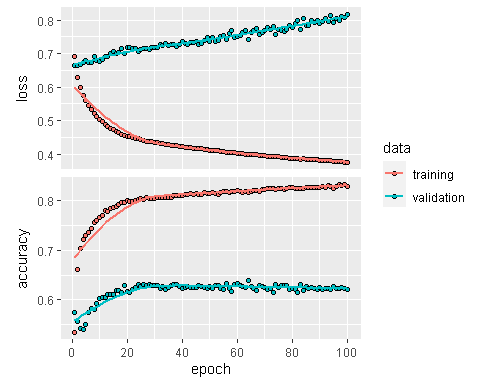
plot(history)



### Question 2

model\_2 <- keras\_model\_sequential(list(  
 layer\_dense(units = 50, activation = "relu"),  
 layer\_dense(units = 25, activation = "relu"),  
 layer\_dense(units = 1, activation = "sigmoid")))  
  
compile(model\_2,  
 optimizer = "rmsprop",  
 loss = "binary\_crossentropy",  
 metrics = "accuracy")  
  
history\_2 <- fit(model\_2, training\_features, training\_labels,  
 epochs = 100, batch\_size = 512, validation\_split = 0.33)

plot(history\_2)



### Question 3

The second model that uses more nodes in the hidden layers better minimizes the training loss but yields a larger validation loss than the first model. The two models generate similar accuracy curves for both training and validation data. Based on the plots, it appears that the second model suffers from overfitting the training data and that the first model results in slightly better performance

### Question 4

# Model 1  
predictions <- predict(model, test\_features)

## 69/69 - 0s - 155ms/epoch - 2ms/step

predicted\_class <- (predictions[, 1] >= 0.5) \* 1  
  
table(predicted\_class, test\_labels)

## test\_labels  
## predicted\_class 0 1  
## 0 329 596  
## 1 365 889

model\_1\_accuracy <- mean(predicted\_class == test\_labels)  
model\_1\_accuracy

## [1] 0.558972

# Model 2  
predictions\_2 <- predict(model\_2, test\_features)

## 69/69 - 0s - 113ms/epoch - 2ms/step

predicted\_class\_2 <- (predictions\_2[, 1] >= 0.5) \* 1  
  
table(predicted\_class\_2, test\_labels)

## test\_labels  
## predicted\_class\_2 0 1  
## 0 302 488  
## 1 392 997

model\_2\_accuracy <- mean(predicted\_class\_2 == test\_labels)  
model\_2\_accuracy

## [1] 0.596145

The first model, which has less nodes in the hidden layers, has an accuracy of about 55.9% while the second model, which has more nodes in the hidden layers, has a slightly improved accuracy of 59.6%. The second model yields better results for making predictions on the test set.