# Homework 7

Jane Downer

```
#devtools::install_github("jabiru/tictoc")
#remove.packages("tensorflow") # May complain if not installed
#install.packages("tensorflow")
#install_tensorflow()
#tf$constant("Hello Tensorflow")
#install.packages("keras")
#install_tensorflow(method = "conda", envname = "py3.6", conda_python_ver
sion = "3.6")
# enable eager execution
# the argument device_policy is needed only when using a GPU
#library(keras)
#install_tensorflow(version = "2.3.0")
library(tensorflow)
library(keras)
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)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:tensorflow':
##
##     train
```

```
library(stats)
library(tictoc)

rm(list=ls())

# Set working directory as needed
setwd("/Users/user/Desktop")
df <- read.csv("activity-small.csv")

# Seed the PRNG
set.seed(1122)
df <- df[sample(nrow(df)), ]

# --- Your code goes below ---
```

## Part 2.1-A

```
indx <- sample(1:nrow(df), 0.20*nrow(df))
test.df  <- df[indx, ]
train.df <- df[-indx, ]

label.test <- test.df$label
test.df$label <- NULL
test.df <- as.data.frame(scale(test.df))
test.df$label <- label.test
rm(label.test)

label.train <- train.df$label
train.df$label <- NULL
train.df <- as.data.frame(scale(train.df))
train.df$label <- label.train
rm(label.train)
rm(indx)

X_train <- select(train.df, -label)
y_train <- train.df$label
y_train.ohe <- to_categorical(y_train)

X_test <- select(test.df, -label)
y_test <- test.df$label
y_test.ohe <- to_categorical(test.df$label)
```

```
create_model <- function(e, batch, split, shape, HL, vary_batch = F,
                         L1_units, L1_act,
                         L2_units, L2_act,
                         L3_units, L3_act)
  {
  tic("  Time taken to train neural network")
  if (HL == 1) {
    model <- keras_model_sequential() %>%
    layer_dense(units = L1_units, activation= L1_act, input_shape=c(shap
e)) %>%
    layer_dense(units = L2_units, activation= L2_act)
  } else if (HL == 2) {
    model <- keras_model_sequential() %>%
    layer_dense(units = L1_units, activation= L1_act, input_shape=c(shap
e)) %>%
    layer_dense(units = L2_units, activation= L2_act)  %>%
    layer_dense(units = L3_units, activation= L3_act)
  }

  model %>% compile(loss = "categorical_crossentropy",
                    optimizer="adam",
                    metrics=c("accuracy"))

  model %>% fit(
              data.matrix(X_train),
              y_train.ohe,
              epochs = e,
              batch_size = batch,
              validation_split = split,
              verbose = 0)

  acc = (model %>% evaluate(as.matrix(X_test), y_test.ohe))[2][[1]]
  acc = round(acc, 3)

  pred.class <- model %>% predict_classes(as.matrix(X_test))
  pred.prob <- model %>% predict(as.matrix(X_test)) %>% round(3)

  m <- confusionMatrix(as.factor(y_test), as.factor(pred.class))
  m_stats = as.data.frame(m[[4]])
  c_0 = c(round(m_stats[1,1],3), round(m_stats[1,2],3), round(m_stats[1,1
1],3))
  c_1 = c(round(m_stats[2,1],3), round(m_stats[2,2],3), round(m_stats[2,1
1],3))
  c_2 = c(round(m_stats[3,1],3), round(m_stats[3,2],3), round(m_stats[3,1
1],3))
  c_3 = c(round(m_stats[4,1],3), round(m_stats[4,2],3), round(m_stats[4,1
1],3))

  toc(log = TRUE, quiet = TRUE)
  log.txt <- tic.log(format = TRUE)
  tic.clearlog()

  elapsed = unlist(log.txt)

  line_one = ""
  if (vary_batch == F) {
    line_one = "Batch gradient descent:\n"
  } else {
    line_one = paste("Batch size: ", toString(batch), "\n",
                     "  Time taken to train neural network: ", toString(e
lapsed), "\n",
                     sep = "", collapse = NULL)
  }

  s = paste(line_one,
            "  Overall accuracy: ", toString(acc), "\n",
            "  Class 0: Sens. = ", toString(c_0[1]), ", Spec. = ",
               toString(c_0[2]), ", Bal.Acc. = ", toString(c_0[3]), "\n",
            "  Class 1: Sens. = ", toString(c_1[1]), ", Spec. = ",
               toString(c_1[2]), ", Bal.Acc. = ", toString(c_1[3]), "\n",
            "  Class 2: Sens. = ", toString(c_2[1]), ", Spec. = ",
```

```
              toString(c_2[2]), ", Bal.Acc. = ", toString(c_2[3]), "\n",
            "  Class 3: Sens. = ", toString(c_3[1]), ", Spec. = ",
              toString(c_3[2]), ", Bal.Acc. = ", toString(c_3[3]), "\n",
            sep = "", collapse = NULL)

  ret = c(s, acc)
  return(ret)
}
```

```
output = create_model(e = 100, batch = 1, split = 0.2, shape = dim(X_trai
n)[2], HL = 1, vary_batch = F,
                      L1_units = 8,                   L1_act = "relu",
                      L2_units = dim(y_train.ohe)[2], L2_act = "softmax",
                      L3_units = NULL,                L3_act = NULL)
```

## Part 2.1-A-i

```
cat(paste0("Accuracy: ", output[2]))
```

```
## Accuracy: 0.81
```

## Part 2.1-A-ii

```
cat(output[1])
```

```
## Batch gradient descent:
##   Overall accuracy: 0.81
##   Class 0: Sens. = 0.931, Spec. = 0.979, Bal.Acc. = 0.955
##   Class 1: Sens. = 0.721, Spec. = 0.935, Bal.Acc. = 0.828
##   Class 2: Sens. = 0.76, Spec. = 0.973, Bal.Acc. = 0.867
##   Class 3: Sens. = 0.839, Spec. = 0.87, Bal.Acc. = 0.854
```

## 2.1-B

```
output_1   = create_model(e = 100, batch = 1, split = 0.2, shape = dim(X_
train)[2], HL = 1, vary_batch = T,
                      L1_units = 8,                   L1_act = "relu",
                      L2_units = dim(y_train.ohe)[2], L2_act = "softmax",
                      L3_units = NULL,                L3_act = NULL)

output_32  = create_model(e = 100, batch = 32, split = 0.2, shape = dim(X
_train)[2], HL = 1, vary_batch = T,
                      L1_units = 8,                   L1_act = "relu",
                      L2_units = dim(y_train.ohe)[2], L2_act = "softmax",
                      L3_units = NULL,                L3_act = NULL)

output_64  = create_model(e = 100, batch = 64, split = 0.2, shape = dim(X
_train)[2], HL = 1, vary_batch = T,
                      L1_units = 8,                   L1_act = "relu",
                      L2_units = dim(y_train.ohe)[2], L2_act = "softmax",
                      L3_units = NULL,                L3_act = NULL)

output_128 = create_model(e = 100, batch = 128, split = 0.2, shape = dim
(X_train)[2], HL = 1, vary_batch = T,
                      L1_units = 8,                   L1_act = "relu",
                      L2_units = dim(y_train.ohe)[2], L2_act = "softmax",
                      L3_units = NULL,                L3_act = NULL)

output_256 = create_model(e = 100, batch = 256, split = 0.2, shape = dim
(X_train)[2], HL = 1, vary_batch = T,
                      L1_units = 8,                   L1_act = "relu",
                      L2_units = dim(y_train.ohe)[2], L2_act = "softmax",
                      L3_units = NULL,                L3_act = NULL)
```

```
cat(output_1[1],"\n")
```

```
## Batch size: 1
##   Time taken to train neural network:   Time taken to train neural net
work: 148.111 sec elapsed
##   Overall accuracy: 0.795
##   Class 0: Sens. = 0.93, Spec. = 0.972, Bal.Acc. = 0.951
##   Class 1: Sens. = 0.698, Spec. = 0.934, Bal.Acc. = 0.816
##   Class 2: Sens. = 0.787, Spec. = 0.967, Bal.Acc. = 0.877
##   Class 3: Sens. = 0.758, Spec. = 0.862, Bal.Acc. = 0.81
##
```

```
cat(output_32[1],"\n")
```

```
## Batch size: 32
##   Time taken to train neural network:   Time taken to train neural net
work: 8.893 sec elapsed
##   Overall accuracy: 0.765
##   Class 0: Sens. = 0.875, Spec. = 0.993, Bal.Acc. = 0.934
##   Class 1: Sens. = 0.679, Spec. = 0.884, Bal.Acc. = 0.782
##   Class 2: Sens. = 0.809, Spec. = 0.974, Bal.Acc. = 0.891
##   Class 3: Sens. = 0.639, Spec. = 0.848, Bal.Acc. = 0.743
##
```

```
cat(output_64[1],"\n")
```

```
## Batch size: 64
##   Time taken to train neural network:   Time taken to train neural net
work: 7.517 sec elapsed
##   Overall accuracy: 0.77
##   Class 0: Sens. = 0.862, Spec. = 0.993, Bal.Acc. = 0.927
##   Class 1: Sens. = 0.672, Spec. = 0.914, Bal.Acc. = 0.793
##   Class 2: Sens. = 0.818, Spec. = 0.962, Bal.Acc. = 0.89
##   Class 3: Sens. = 0.7, Spec. = 0.841, Bal.Acc. = 0.771
##
```

```
cat(output_128[1],"\n")
```

```
## Batch size: 128
##   Time taken to train neural network:   Time taken to train neural net
work: 5.363 sec elapsed
##   Overall accuracy: 0.75
##   Class 0: Sens. = 0.836, Spec. = 0.992, Bal.Acc. = 0.914
##   Class 1: Sens. = 0.7, Spec. = 0.88, Bal.Acc. = 0.79
##   Class 2: Sens. = 0.756, Spec. = 0.948, Bal.Acc. = 0.852
##   Class 3: Sens. = 0.658, Spec. = 0.858, Bal.Acc. = 0.758
##
```

```
cat(output_256[1])
```

```
## Batch size: 256
##   Time taken to train neural network:   Time taken to train neural net
work: 5.001 sec elapsed
##   Overall accuracy: 0.65
##   Class 0: Sens. = 0.789, Spec. = 0.992, Bal.Acc. = 0.89
##   Class 1: Sens. = 0.513, Spec. = 0.887, Bal.Acc. = 0.7
##   Class 2: Sens. = 0.786, Spec. = 0.943, Bal.Acc. = 0.864
##   Class 3: Sens. = 0.182, Spec. = 0.757, Bal.Acc. = 0.469
```

## Part 2.1-C-i

```
cat(paste0("Increasing branch sizes results in more data entries being pr
ocessed between the weights being updated. As a result, weights are updat
ed less frequently and execution time is shorter."))
```

```
## Increasing branch sizes results in more data entries being processed b
etween the weights being updated. As a result, weights are updated less f
requently and execution time is shorter.
```

## Part 2.1-C-ii

```
cat(paste0("With a larger batch size, the variance within batches is larg
er. As a result, when updating the weights, the changes vary more broadly
than they do for smaller batch sizes. This leads to lower accuracy -- bot
h balanced and overall. Note: in my results, this effect is not linear an
d seems to disappear around the 128-batch mark."))
```

```
## With a larger batch size, the variance within batches is larger. As a
result, when updating the weights, the changes vary more broadly than the
y do for smaller batch sizes. This leads to lower accuracy -- both balanc
ed and overall. Note: in my results, this effect is not linear and seems
to disappear around the 128-batch mark.
```

## Part 2.1-D

```
output_d_1 = create_model(e = 100, batch = 1, split = 0.2, shape = dim(X_
train)[2], HL = 1, vary_batch = F,
                      L1_units = 8,                    L1_act = "relu",
                      L2_units = dim(y_train.ohe)[2], L2_act = "softmax",
                      L3_units = NULL,                 L3_act = NULL)

output_d_2 = create_model(e = 100, batch = 1, split = 0.2, shape = dim(X_
train)[2], HL = 2, vary_batch = F,
                      L1_units = 8,                    L1_act = "relu",
                      L2_units = dim(y_train.ohe)[2], L2_act = "softmax",
                      L3_units = dim(y_train.ohe)[2], L3_act = "softplus"
)

output_d_3 = create_model(e = 100, batch = 1, split = 0.2, shape = dim(X_
train)[2], HL = 1, vary_batch = F,
                      L1_units = 8,                    L1_act = "relu",
                      L2_units = dim(y_train.ohe)[2], L2_act = "softmax",
                      L3_units = NULL,                 L3_act = NULL)

output_d_4 = create_model(e = 100, batch = 1, split = 0.2, shape = dim(X_
train)[2], HL = 2, vary_batch = F,
                      L1_units = 8,                    L1_act = "relu",
                      L2_units = dim(y_train.ohe)[2], L2_act = "softmax",
                      L3_units = dim(y_train.ohe)[2], L3_act = "softplus"
)
```

## Part 2.-D-i and 2.1-D-ii

```
cat(output_d_1[1])
```

```
## Batch gradient descent:
##   Overall accuracy: 0.795
##   Class 0: Sens. = 0.903, Spec. = 0.993, Bal.Acc. = 0.948
##   Class 1: Sens. = 0.688, Spec. = 0.934, Bal.Acc. = 0.811
##   Class 2: Sens. = 0.792, Spec. = 0.974, Bal.Acc. = 0.883
##   Class 3: Sens. = 0.808, Spec. = 0.845, Bal.Acc. = 0.826
```

```
cat(output_d_2[1])
```

```
## Batch gradient descent:
##   Overall accuracy: 0.72
##   Class 0: Sens. = 0.914, Spec. = 0.972, Bal.Acc. = 0.943
##   Class 1: Sens. = 0.571, Spec. = 0.927, Bal.Acc. = 0.749
##   Class 2: Sens. = 0.776, Spec. = 0.974, Bal.Acc. = 0.875
##   Class 3: Sens. = 0.562, Spec. = 0.788, Bal.Acc. = 0.675
```

```
cat(output_d_3[1])
```

```
## Batch gradient descent:
##   Overall accuracy: 0.775
##   Class 0: Sens. = 0.917, Spec. = 0.986, Bal.Acc. = 0.951
##   Class 1: Sens. = 0.672, Spec. = 0.901, Bal.Acc. = 0.787
##   Class 2: Sens. = 0.776, Spec. = 0.974, Bal.Acc. = 0.875
##   Class 3: Sens. = 0.697, Spec. = 0.85, Bal.Acc. = 0.774
```

```
cat(output_d_4[1])
```

```
## Batch gradient descent:
##   Overall accuracy: 0.81
##   Class 0: Sens. = 0.932, Spec. = 0.986, Bal.Acc. = 0.959
##   Class 1: Sens. = 0.732, Spec. = 0.917, Bal.Acc. = 0.824
##   Class 2: Sens. = 0.776, Spec. = 0.974, Bal.Acc. = 0.875
##   Class 3: Sens. = 0.778, Spec. = 0.878, Bal.Acc. = 0.828
```

## Part 2.1-D

```
cat(paste0("The best model had two hidden layers and batch size 1. The nu
mber of neurons per layers were 8, 4, and 4, and these layers' activation
funtions were 'relu', 'softmax', and 'softplus'. The overall accuracy was
0.825, which is not all that much greater than our initial accuracy of 0.
810. I noticed that changing the number of layers did not guarantee bette
r or worse results -- in the four setups I tried in the above step, two h
idden layers was associated with lower accuracy when the initial units we
re 16, but associated with higher accuracy when the initial units were
 8." ))
```

```
## The best model had two hidden layers and batch size 1. The number of n
eurons per layers were 8, 4, and 4, and these layers' activation funtions
were 'relu', 'softmax', and 'softplus'. The overall accuracy was 0.825, w
hich is not all that much greater than our initial accuracy of 0.810. I n
oticed that changing the number of layers did not guarantee better or wor
se results -- in the four setups I tried in the above step, two hidden la
yers was associated with lower accuracy when the initial units were 16, b
ut associated with higher accuracy when the initial units were 8.
```