Palmer Penguins

Prashan A. Welipitiya

4/26/2021

```r
library(pacman)
p_load(tidyverse, palmerpenguins, tidymodels, tidyr, ggplot2)
```

```r
library("keras")
```

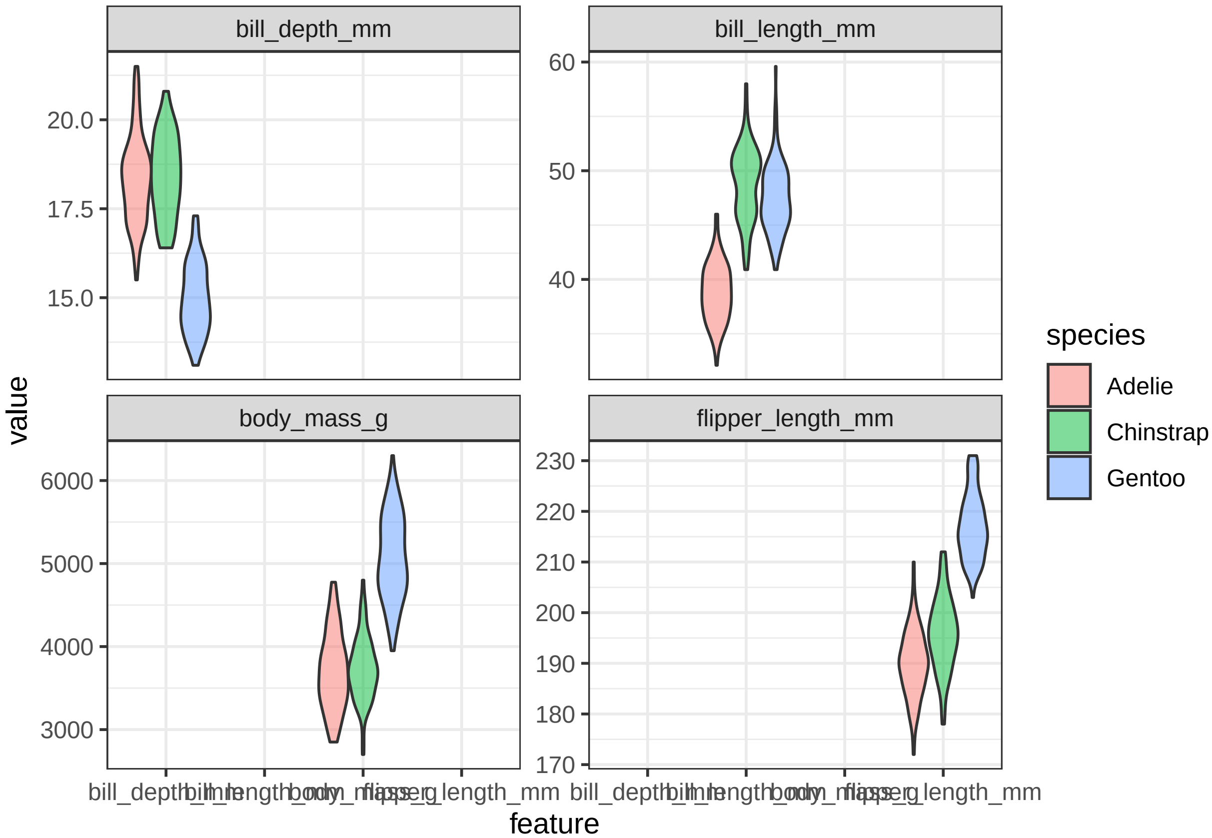
Artificial Neural Network Using the Palmer Penguins Data Set

Data

The *palmerpenguins* data set contains a total of 344 observations. We will be using 4 input features *bill_length_mm*, *bill_depth_mm*, *flipper_length_mm* and *body_mass_g* and 3 output classes for the species *Adelie Chinstrap* and *Gentoo*. The distributions of the feature values looks like so:

```r
penguins_tib <- as_tibble(penguins)
penguins_tib <- penguins_tib %>% select(species, bill_length_mm, bill_depth_mm, flipper_length_mm, body_
```

```r
penguins_tib %>% pivot_longer(names_to = "feature", values_to = "value", -species) %>%
  ggplot(aes(x = feature, y = value, fill = species)) +
  geom_violin(alpha = 0.5, scale = "width") +
  theme_bw() +
  facet_wrap(~ feature, scales = "free_y")
```



Our aim is to connect the 4 input features to the correct output class using an artificial neural network. For this task, we have chosen the following simple architecture with one input layer with 4 neurons (one for each feature), one hidden layer with 4 neurons and one output layer with 3 neurons (one for each class), all fully connected.

Our artificial neural network will have a total of 35 parameters: 4 for each input neuron connected to the hidden layer, plus an additional 4 for the associated first bias neuron and 3 for each of the hidden neurons connected to the output layer, plus an additional 3 for the associated second bias neuron, i.e. $4 \times 4 + 4 + 4 \times 3 + 3 = 35$

Prepare data

We start with slightly wrangling the iris data set by renaming and scaling the features and converting character labels to numeric.

```
nn_dat <- penguins_tib %>%
  mutate(bill_depth = scale(bill_depth_mm),
         bill_length  = scale(bill_length_mm),
         mass = scale(body_mass_g),
         flipper_length  = scale(flipper_length_mm),
         class_label  = as.numeric(species) - 1) %>%
    select(bill_depth, bill_length, flipper_length, mass, class_label)

nn_dat %>% head()
```

```
## # A tibble: 6 x 5
```

```
##   bill_depth[,1] bill_length[,1] flipper_length[,1] mass[,1] class_label
##            <dbl>           <dbl>              <dbl>    <dbl>       <dbl>
## 1          0.784          -0.883              -1.42    -0.563          0
## 2          0.126          -0.810              -1.06    -0.501          0
## 3          0.430          -0.663              -0.421   -1.19           0
## 4          1.09           -1.32               -0.563   -0.937          0
## 5          1.75           -0.847              -0.776   -0.688          0
## 6          0.329          -0.920              -1.42    -0.719          0
```

Then, we create indices for splitting the penguin data into a training and a test data set. We set aside 20% of the data for testing.

```
n <- nrow(nn_dat)
n
```

```
## [1] 342
```

```
penguin_parts <- nn_dat %>%
  initial_split(prop = 0.8)

train <- penguin_parts %>%
  training()

test <- penguin_parts %>%
  testing()

list(train, test) %>%
  map_int(nrow)
```

```
## [1] 274  68
```

```
n_total_samples <- nrow(nn_dat)

n_train_samples <- nrow(train)

n_test_samples <- nrow(test)
```

Create training and test data

Note that the functions in the keras package are expecting the data to be in a matrix object and not a tibble. So as.matrix is added at the end of each line.

```
x_train <- train %>% select(-class_label) %>% as.matrix()
y_train <- train %>% select(class_label) %>% as.matrix() %>% to_categorical()

x_test <- test %>% select(-class_label) %>% as.matrix()
y_test <- test %>% select(class_label) %>% as.matrix() %>% to_categorical()

dim(y_train)
```

```
## [1] 274   3
```

```
dim(y_test)
```

```
## [1] 68  3
```

Set Architecture

With the data in place, we now set the architecture of our neural network.

```
model <- keras_model_sequential() %>%
  layer_dense(units = 4, activation = 'relu', input_shape = 4) %>%
  layer_dense(units = 3, activation = 'softmax')
model %>% summary
```

```
## Model: "sequential"
## ________________________________________________________________________________
## Layer (type)                        Output Shape                        Param #
## ================================================================================
## dense_1 (Dense)                     (None, 4)                           20
## ________________________________________________________________________________
## dense (Dense)                       (None, 3)                           15
## ================================================================================
## Total params: 35
## Trainable params: 35
## Non-trainable params: 0
## ________________________________________________________________________________
```

Next, the architecture set in the model needs to be compiled.

```
model %>% compile(
  loss      = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics   = c('accuracy')
)
```

Train the Artificial Neural Network

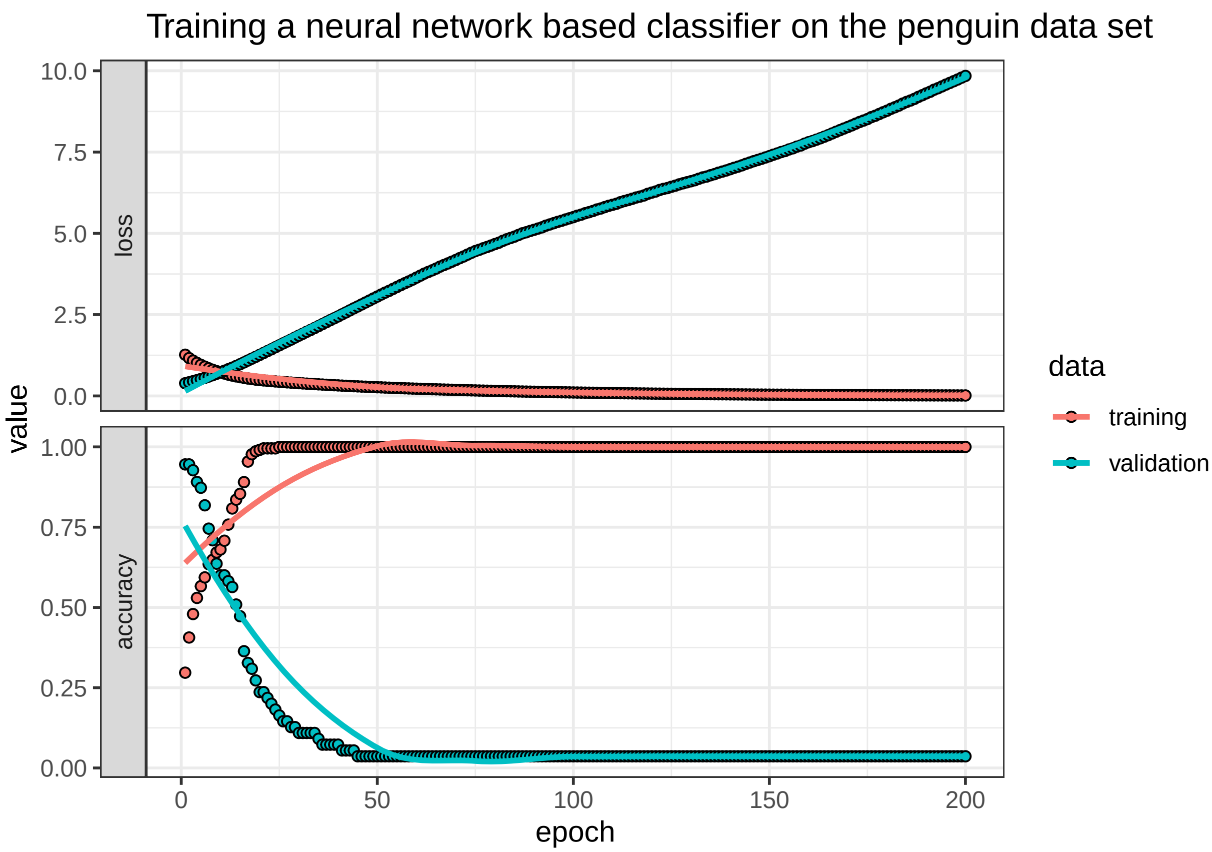
Lastly we fit the model and save the training progress in the *history* object.

Try changing the *validation_split* from 0 to 0.2 to see the *validation_loss*.

```
history <- model %>% fit(
  x = x_train, y = y_train,
  epochs = 200,
  batch_size = 20,
  validation_split = 0.2
)

plot(history) +
  ggtitle("Training a neural network based classifier on the penguin data set") +
  theme_bw()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Evaluate Network Performance

The final performance can be obtained like so.

```
perf <- model %>% evaluate(x_test, y_test)
print(perf)
```

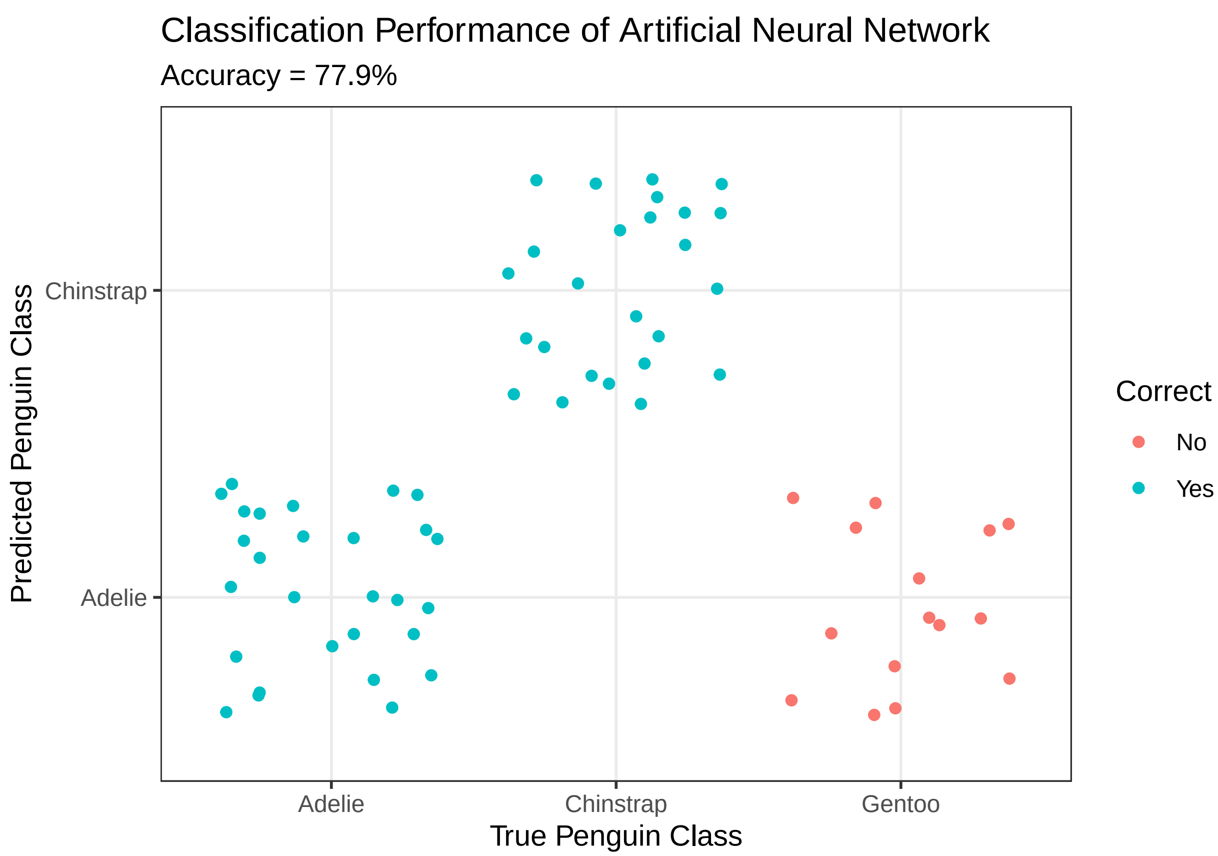
```
##      loss  accuracy
## 2.0845377 0.7794118
```

For the next plot the predicted and true values need to be in a vector. Note that the true values need to be unlisted before putting them into a numeric vector.

```
classes <- penguins %>% pull(species) %>% unique()
y_pred  <- model %>% predict_classes(x_test)
y_true  <- test %>% select(class_label) %>% unlist() %>% as.numeric()

tibble(y_true = classes[y_true + 1], y_pred = classes[y_pred + 1],
       Correct = ifelse(y_true == y_pred, "Yes", "No") %>% factor) %>%
  ggplot(aes(x = y_true, y = y_pred, colour = Correct)) +
  geom_jitter() +
  theme_bw() +
```

```r
ggtitle(label = "Classification Performance of Artificial Neural Network",
        subtitle = str_c("Accuracy = ",round(perf[2],3)*100,"%")) +
xlab(label = "True Penguin Class") +
ylab(label = "Predicted Penguin Class")
```




```r
library(gmodels)
```

```
## Warning: package 'gmodels' was built under R version 4.0.4
```

```r
CrossTable(y_pred, y_true,
           prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
           dnn = c('predicted', 'actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  68
##
```

```
##
##               | actual
##     predicted |         0 |         1 |         2 | Row Total |
## -------------|-----------|-----------|-----------|-----------|
##             0 |        28 |        15 |         0 |        43 |
##               |     1.000 |     1.000 |     0.000 |           |
## -------------|-----------|-----------|-----------|-----------|
##             2 |         0 |         0 |        25 |        25 |
##               |     0.000 |     0.000 |     1.000 |           |
## -------------|-----------|-----------|-----------|-----------|
## Column Total |        28 |        15 |        25 |        68 |
##               |     0.412 |     0.221 |     0.368 |           |
## -------------|-----------|-----------|-----------|-----------|
##
##
```

Adding Dropout Layer

```
dpt_model <- keras_model_sequential() %>%
  layer_dense(units = 4, activation = 'relu', input_shape = 4) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 3, activation = 'softmax')
dpt_model %>% summary
```

```
## Model: "sequential_1"
## ________________________________________________________________________________
## Layer (type)                        Output Shape                    Param #
## ================================================================================
## dense_3 (Dense)                     (None, 4)                       20
## ________________________________________________________________________________
## dropout (Dropout)                   (None, 4)                       0
## ________________________________________________________________________________
## dense_2 (Dense)                     (None, 3)                       15
## ================================================================================
## Total params: 35
## Trainable params: 35
## Non-trainable params: 0
## ________________________________________________________________________________
```

compile

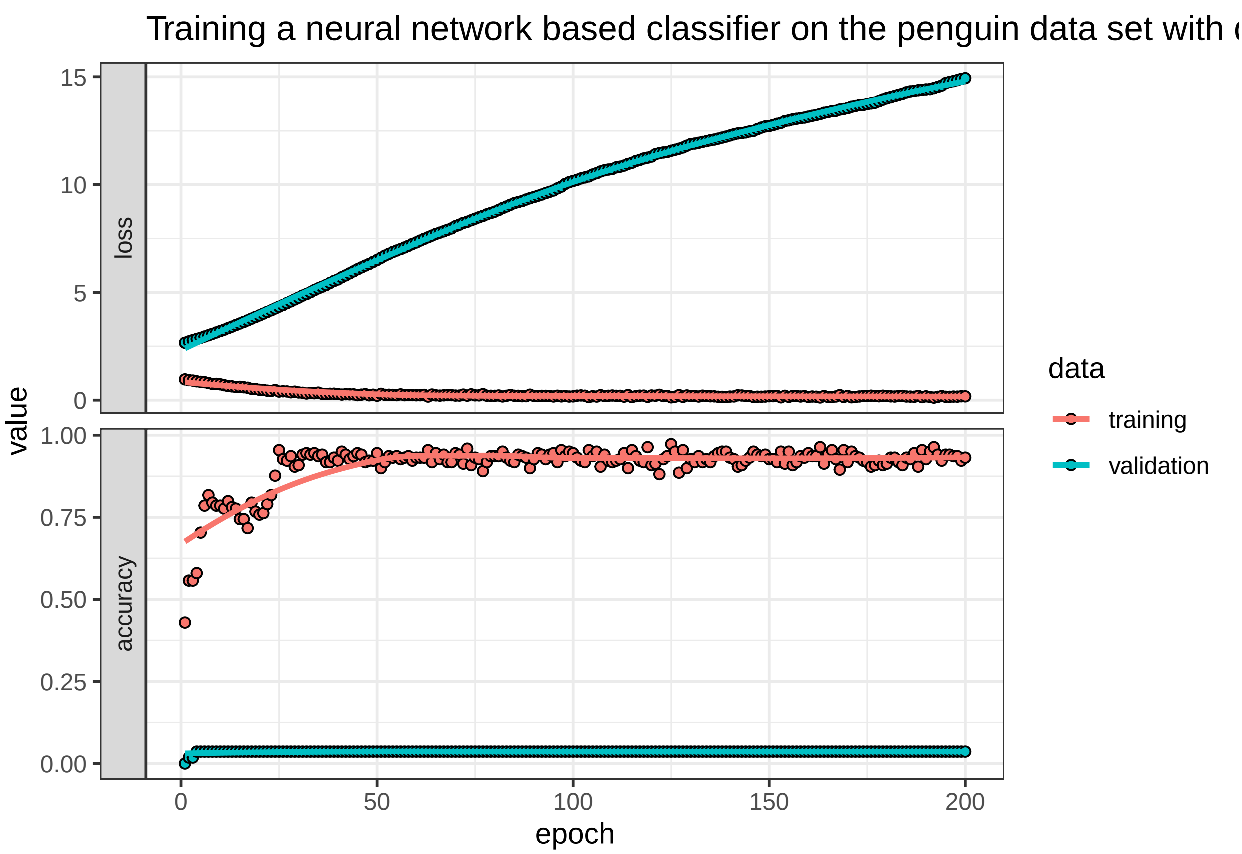
```
dpt_model %>% compile(
  loss      = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics   = c('accuracy')
)
```

Train

```
dpt_history <- dpt_model %>% fit(
  x = x_train, y = y_train,
  epochs = 200,
  batch_size = 20,
  validation_split = 0.2
)

plot(dpt_history) +
  ggtitle("Training a neural network based classifier on the penguin data set with dropout") +
  theme_bw()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



performance

```
dpt_perf <- dpt_model %>% evaluate(x_test, y_test)
print(dpt_perf)
```

```
##      loss  accuracy
## 3.1761396 0.7794118
```

```
citation("palmerpenguins")
```

```
##
```

```
## To cite palmerpenguins in publications use:
##
##   Horst AM, Hill AP, Gorman KB (2020). palmerpenguins: Palmer
##   Archipelago (Antarctica) penguin data. R package version 0.1.0.
##   https://allisonhorst.github.io/palmerpenguins/
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {palmerpenguins: Palmer Archipelago (Antarctica) penguin data},
##     author = {Allison Marie Horst and Alison Presmanes Hill and Kristen B Gorman},
##     year = {2020},
##     note = {R package version 0.1.0},
##     url = {https://allisonhorst.github.io/palmerpenguins/},
##   }
```