Tarea 2 - Keras prediction

Madrid rain

Exercise done by Cedric Prieels for the Machine Learning I course of the master in Data Science.

The objective of this exercise is to design a neural network able to predict the precipitation in Madrid using large scale variables. The training dataset has been given to us, and consists of 8766 observations of 450 predicting variables, and the actual value of the precipitation measured in Madrid. Different parameters need to be used (number of layers, neurons, early stopping, regularization,...) to try and define the best neural network possible, with the least error and the best generalization capacity possible. This neural network should be able to give us if the day is rainy or not (precipitation $<$ 1mm), and additionally, the quantity of rain expected.

Open and study the dataset

First of all, as always, let's open the dataset we have been given, called Madrid_Alumno.rda and which contains several thousands of observations of the the precipitation in Madrid. We can also load the packages we are going to need at this stage.

```
library(keras)
library(magrittr) #To get access to the %>% operator to define the DNN
load('~/Downloads/Madrid_Alumno.rda') #Set the correct working path

print(dim(xTrain)) #8766 observations of 450 different variables
```

```
## [1] 8766  450
```

```
head(yTrain) #Value of the precipitation, obviously measured as a positive number
```

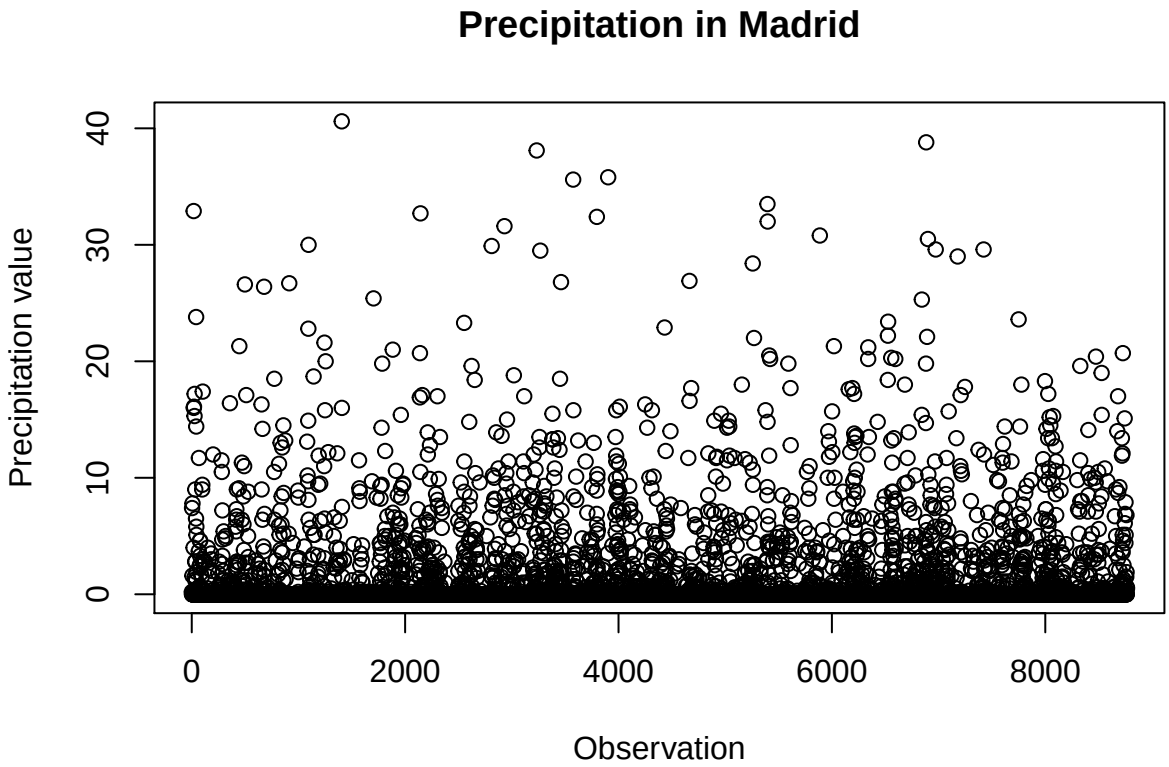
```
## [1] 0.2 0.0 7.4 0.0 0.0 1.6
```

```
print(dim(yTrain)) #8766 observations in this case as well
```

```
## [1] 8766
```

We can then see that we have successfully loaded 8766 observations of 450 different variables (xTrain) and the value of the precipitation for any given observation (yTrain), measured as a positive number in the range [0, inf[. We can try plotting this precipitation value in order to understand it a bit better.

```
plot(yTrain, xlab="Observation", ylab="Precipitation value", main="Precipitation in Madrid")
```



This allows us to see that for these 8766 observations, the maximum value of the precipitation measured is around 40mm. We now define a rainy day as a day having more than 1 mm of rain, while the day will be considered to be non-rainy if the precipitation measured is smaller than 1 mm. We can already see with the previous plot that most days are non-rainy, which might be a problem we will need to deal with when training a DNN since we clearly have an imbalanced dataset.

We already notice at this point that we need to be careful, since the precipitation is a semicontinous variable in the sense that many days are expected to have a value of precipitation exactly equal to 0. Now, we are going to rescale the data to avoid any scale issue, by substracting the mean and dividing by the standard deviation.

```
for (i in 0:dim(xTrain)[2]) { #Rescale each column
  xTrain[,i] <- (xTrain[,i] - mean(xTrain[,i]))/(sd(xTrain[,i]))
  xTest[,i] <- (xTest[,i] - mean(xTest[,i]))/(sd(xTest[,i]))
  yTrain[i] <- (yTrain[i] - mean(yTrain))/(sd(yTrain))
}
sum(is.na(xTrain)) #Check if na are present, which might be a problem later on
```

```
## [1] 0
```

Binary classification

The goal here is to predict whether a given day will be rainy or not. A day is considered to be rainy if the precipitation value is larger than 1mm.

```
rainyDays <- (yTrain > 1)
head(rainyDays)
```

```
## [1] FALSE FALSE  TRUE FALSE FALSE FALSE
```

Let's start defining our keras model for this particular case. We use the relu activation function for the hidden layers because it is more stable under derivation, which is always something interesting when defininig

multiple hidden layers.

Several combinations have been tested for this model, keeping the one resulting in the highets validation accuracy (0.8957): - For the number of layers, the combinations (1000, 500, 1), (1000, 500, 200, 100, 100, 1), (500, 400, 300, 200, 100, 1) and (200, 200, 200, 200, 1) have been tested, resulting in small differences. - For the learning rate, values = 1e-4, 5e-4, 0.001, 0.002 and 0.005 have been tested. - The early stopping has been set and removed. - The patience has been set to 1, 5 and 10. - The epochs has always been set to 200 but the early stopping, when set, usually prevented to reach this value anyway. - The batch size has been set to 32, 64, 128 and 256.

```
k_clear_session() #Reset all the keras models

myModel <- keras_model_sequential()
myModel %>%
  layer_dense(units = 1000, input_shape = dim(xTrain)[2], activation = "relu") %>%
  layer_dense(units = 500, activation = "relu") %>%
  layer_dense(units = 200, activation = "relu") %>%
  layer_dense(units = 100, activation = "relu") %>%
  layer_dense(units = 100, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")
str(myModel)
```

```
## Model
## Model: "sequential"
## _________________________________________________________________________
## Layer (type)                         Output Shape                     Param #
## =========================================================================
## dense (Dense)                        (None, 1000)                     451000
## _________________________________________________________________________
## dense_1 (Dense)                      (None, 500)                      500500
## _________________________________________________________________________
## dense_2 (Dense)                      (None, 200)                      100200
## _________________________________________________________________________
## dense_3 (Dense)                      (None, 100)                      20100
## _________________________________________________________________________
## dense_4 (Dense)                      (None, 100)                      10100
## _________________________________________________________________________
## dense_5 (Dense)                      (None, 1)                        101
## =========================================================================
## Total params: 1,082,001
## Trainable params: 1,082,001
## Non-trainable params: 0
## _________________________________________________________________________
```

Now let's compile this model.

```
myModel %>% compile(
  optimizer = optimizer_sgd(lr=0.002),
  loss = "binary_crossentropy",
  metrics = "accuracy"
)
```

Train this model, using the early_stopping parameter. The data has been splitted into test and validation using a 0.2 splitting.

```
callbacks <- list(callback_early_stopping(monitor='val_loss', patience=10, mode='auto'))
history <- myModel %>% fit(xTrain, rainyDays, epochs = 200, batch_size = 256,
```

```
                          validation_split = 0.2, callbacks = callbacks)
plot(history, main="History of the model training")
```

In these plots, the top corresponds to the loss while the bottom corresponds to the accuracy ; the dots correspond to the train dataset and the crosses to the validation dataset. Finally, let's predict and display the results obtained.

```
yTest_ocu <- myModel$predict_classes(xTest)
which(yTest_ocu > 0.5) #Which days are going to be categorized as rainy?
```

```
##   [1]    5    7   20   50   53   55   56   57   84   85   86   88   89  101  102
##  [16]  103  104  105  109  110  112  114  115  125  154  155  273  274  275  285
##  [31]  289  290  292  298  299  300  301  304  312  313  314  319  326  327  330
##  [46]  334  338  339  340  343  352  361  417  418  419  420  421  437  451  453
##  [61]  454  455  457  477  487  488  495  496  497  507  508  553  612  613  659
##  [76]  666  668  701  704  769  811  813  815  816  818  823  860  862  863  864
##  [91]  867 1014 1015 1026 1032 1034 1048 1049 1066 1067 1091 1102 1103 1111 1112
## [106] 1125 1130 1152 1153 1159 1172 1173 1174 1175 1178 1179 1191 1192 1207 1208
## [121] 1219 1256 1261 1351 1360 1385 1386 1387 1388 1389 1391 1392 1394 1395 1403
## [136] 1404 1407 1408 1415 1416 1423 1424 1425 1432 1435 1483 1490 1491 1495 1500
## [151] 1512 1552 1553 1558 1559 1560 1576 1577 1578 1579 1581 1583 1601 1603 1604
## [166] 1605 1606 1626 1629 1698 1734 1736 1758 1784 1785 1815 1816 1828 1829 1839
## [181] 1860 1875 1876 1903 1924 1925 1926 1927 1933 1934 1935 1936 1937 1954 1955
## [196] 1961 1962 1963 1969 1970 1972 1976 1977 1978 2079 2118 2122 2128 2131 2160
## [211] 2168 2169 2174 2188 2189
```

Precipitation prediction

Now, let's create another model able to predict the quantity of precipitation. Several combinations have been tested for this model as well, keeping the one resulting in the highest validation accuracy (): - For the number of layers, the combinations (1000, 500, 1), (1000, 500, 200, 100, 100, 1), (500, 400, 300, 200, 100, 1) and (200, 200, 200, 200, 1) have been tested, resulting in small differences. - For the learning rate, values = 1e-4, 5e-4, 0.001, 0.002 and 0.005 have been tested. - The early stopping has been set and removed. - The patience has been set to 5, 10 and 20. - The epochs has always been set to 200 but the early stopping, when set, usually prevented to reach this value anyway. - The batch size has been set to 64, 128, 256 and 512.

```
k_clear_session() #Reset all the keras models

#Define the model
myModel <- keras_model_sequential()
myModel %>%
  layer_dense(units = 1000, input_shape = dim(xTrain)[2], activation = "relu") %>%
  layer_dense(units = 500, activation = "relu") %>%
  layer_dense(units = 100, activation = "relu") %>%
  layer_dense(units = 1, activation = "linear")
str(myModel)
```

```
## Model
## Model: "sequential"
## _________________________________________________________________________________
## Layer (type)                        Output Shape                    Param #
## =================================================================================
## dense (Dense)                       (None, 1000)                    451000
## _________________________________________________________________________________
## dense_1 (Dense)                     (None, 500)                     500500
## _________________________________________________________________________________
## dense_2 (Dense)                     (None, 100)                     50100
## _________________________________________________________________________________
## dense_3 (Dense)                     (None, 1)                       101
## =================================================================================
## Total params: 1,001,701
## Trainable params: 1,001,701
## Non-trainable params: 0
## _________________________________________________________________________________
```

```
#Compile our model
myModel %>% compile(
  optimizer = optimizer_sgd(lr = 0.0003),
  loss = "mse",
  metrics = "accuracy"
)

#Train the model
callbacks <- list(callback_early_stopping(monitor='val_loss', patience=10, mode='auto'))
history <- myModel %>% fit(xTrain, yTrain, epochs = 200, batch_size = 512, validation_split = 0.2, calll
plot(history, main="History of the model training")
```

Using this model, predictions can be made as well using xTest.

```
yTest_reg <- myModel$predict(xTest)
head(yTest_reg, 10)
```

```
##             [,1]
##  [1,]  0.3753992
##  [2,]  0.7222450
##  [3,]  1.2125062
##  [4,]  0.5017618
##  [5,]  5.0969496
##  [6,] -0.1258958
##  [7,]  3.2623580
##  [8,]  1.5753803
##  [9,]  2.5725410
## [10,]  0.4174339
```

We can finally save both these predictions in a separate file.

```
save(yTest_ocu,yTest_reg,file="yTest.rda")
```