Homework 3: GAM and NN

Thomas Zwiller

There are four questions (30 total points) in this assignment. The minimum increment is 1 point. Please type in your answers directly in the R Markdown file. After completion, **successfully** knitr it as an html file. Submit **both** the html file and the R Markdown file via Canvas. Please name the R Markdown file in the following format: LastName_FirstName_HW3.Rmd, e.g. Zhao_Zifeng_HW3.Rmd.

Credit Balance Dataset [30 points]

The credit balance dataset contains information about 400 individuals' credit usage and other personal information. The data is stored in `Credit.csv`. It contains 8 variables, `Income`, `Limit`, `Rating`, `Cards`, `Age`, `Education`, `Student`, and `Balance`. We would like to build several statistical models to predict `Balance` (i.e. credit usage) of a person with given personal information. The data description is as follows.

- `Income`: Income in $1,000's
- `Limit`: Credit limit
- `Rating`: Credit rating
- `Cards`: Number of credit cards
- `Age`: Age in years
- `Education`: Education in years
- `Student`: A factor with levels No and Yes indicating whether the individual is a student
- `Balance`: Average credit card balance in $.

Q1 [4 points] Data Partition

Q1(a) [2 points] Let's correctly read in the data in `Credit.csv` and name it as `total_data`.

```
#reading in the data as "credit" from a csv with headers
credit <- read.csv("/Users/TomTheIntern/Desktop/Mendoza/Mod 2/Advanced Stats/Homework 3/Credit.csv", hea
```

Q1(b) [2 points] Let's partition the data in `total_data` into training **(80%)** and test data **(20%)** and store them as `R` objects `train_data` and `test_data` respectively. Use random seed **`set.seed(7)`**!

```
#setting the random seed to 7 so we can test the randomly partitioned data
set.seed(7)
#and then getting the number of observations
num_obs <- nrow(credit)
#and then we get the indices we need using the sample function
train_data_rows <- sample(1:num_obs, 0.80*num_obs)
#with our num of rows we can make training data
train_data <- credit[train_data_rows , ]
#and then testing data using the opposite indices
test_data <- credit[-train_data_rows , ]
```

Q2 [8 points] Linear Regression and GAM

Q2(a) [**3 points**] Fit a linear regression model of the **original scale** `Balance` w.r.t. all 7 predictors using the **training data**, name it `lm_full`.

```
#now we can make our linear regression model
lm_full <- lm(Balance ~ . , data = train_data)
```

Q2(b) [**5 points**] Fit a GAM of the **original scale** `Balance` w.r.t. all 7 predictors using the **training data**, name it `gam_full`. Let's use splines with **df=4** for all 6 numerical predictors, which include `Income`, `Limit`, `Rating`, `Cards`, `Age` and `Education`.

```
#importing the gam library
library(gam)
```

```
## Warning: package 'gam' was built under R version 4.4.1
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.22-5
```

```
gam_full <- gam(Balance ~ s(Income) + s(Limit) + s(Rating) + s(Cards) + s(Age)  + s(Education) + Studen
```

Q3 [10 points] Neural Networks

Fit an NN of **standardized** `Balance` w.r.t. all 7 predictors using the **training data**, name it `nn_full`. For the architecture of NN, let's use two hidden layers with 4 hidden units in the first layer and 2 hidden units in the second layer.

Q3(a) [**2 points**] Let's generate the **training dataset** that are needed for the estimation of NN using the function `model.matrix()` and store it in `x_train_nn`. In addition, use the `scale()` function to standardize the predictors by centering with mean and scaling with sd.

```
#first we make an x_train set using the model matrix function
x_train_nn <- model.matrix(  ~ Income + Limit + Rating + Cards + Age + Education + Student, data = trai
#we then can get the mean of the data
x_mean <- apply(x_train_nn, FUN = mean, MARGIN = 2)
#and the standard deviation
x_sd <- apply(x_train_nn, FUN = sd, MARGIN = 2)
#and then scale our data
x_train_nn <- scale(x_train_nn, center = x_mean, scale = x_sd)
```

Q3(b) [**2 points**] Let's further standardize the dependent variable `Balance` by dividing its maximum value. In addition, combine the standardized `Balance` with the standardized predictors `x_train_nn` generated in Q3(a).

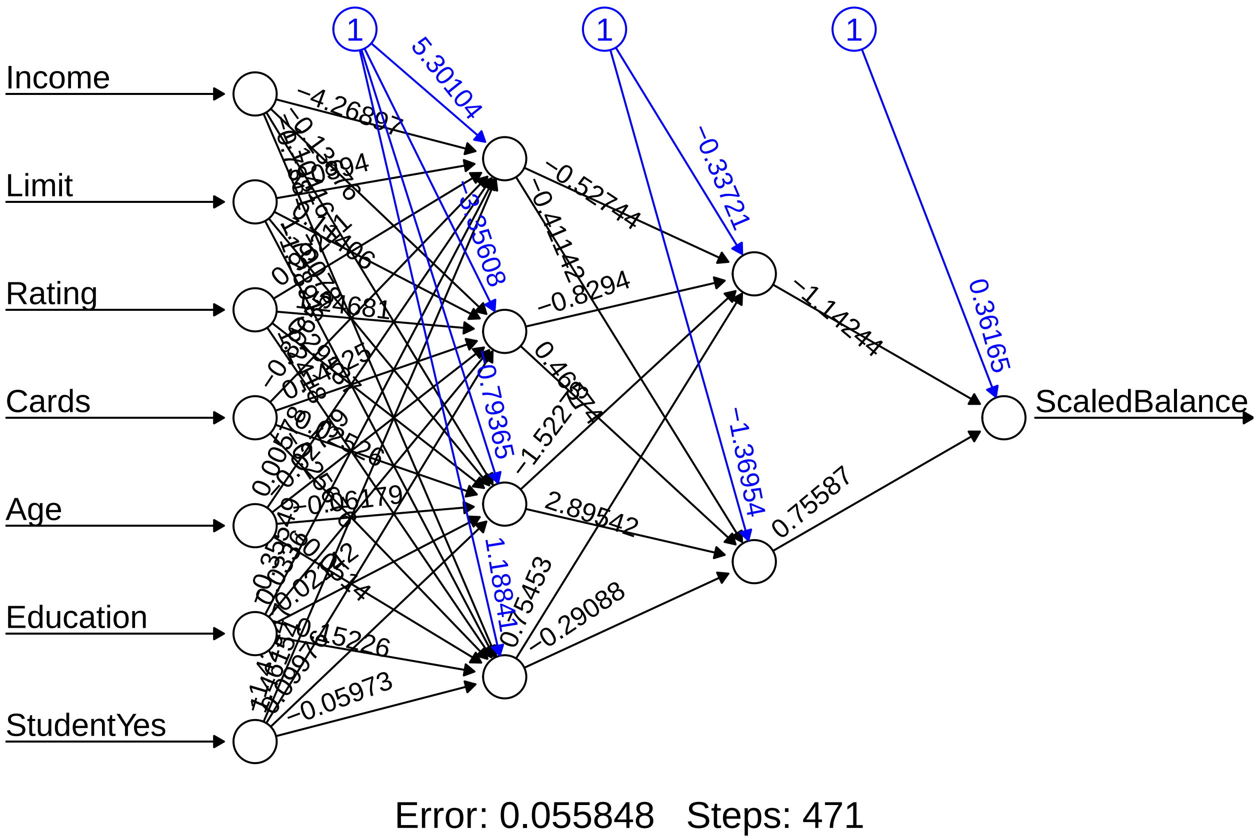
```r
#Here I get the max of the income so I can scale the data
balance_max <- max(train_data$Balance)

#then I alter the income by the max so its scaled
x_train_nn <- cbind.data.frame(train_data$Balance / balance_max, x_train_nn)

#and then input balance into the frame so that it wasn't scaled by its mean and sd
x_train_nn$Balance <- train_data$Balance

#changing the name for readability
colnames(x_train_nn)[1] <- 'ScaledBalance'
```

Q3(c) [**2 points**] Let's generate the **test dataset** that are needed for the out-of-sample prediction evaluation of NN using the function `model.matrix` and store it in `x_test_nn`. Use the `scale()` function to standardize the predictors by centering with mean and scaling with sd as in Q3(a).

```r
#creating the test training set
x_test_nn <- model.matrix( ~ Income + Limit + Rating + Cards + Age + Education + Student, data = test_da

#then scaling it using the old mean and sd
x_test_nn <- scale(x_test_nn, center = x_mean, scale = x_sd)

#then adjusting income by using the old income max
x_test_nn <- cbind.data.frame(test_data$Balance / balance_max, x_test_nn)

#and renaming it for readability
colnames(x_test_nn)[1] <- 'ScaledBalance'

#and then inputting the balance so it can be tested
x_test_nn$Balance <- test_data$Balance
```

Q3(d) [**4 points**] Let's fit an NN that has two hidden layers with 4 hidden units in the first layer and 2 hidden units in the second layer. Make sure to use random seed **`set.seed(7)`**!

```r
## Write code solution to Q3(d) here
library(neuralnet)
set.seed(7)

#training the model
nn_full <- neuralnet(ScaledBalance ~ Income + Limit + Rating + Cards + Age + Education + StudentYes, da

#and then plotting the nn
plot(nn_full, rep = 1)
```



Q4 [8 points] Model Evaluation (Prediction)

Q4(a) [4 points] Use `lm_full`, `gam_full` and `nn_full` to generate predictions for `Balance` on the test data and store the prediction in `lm_pred`, `gam_pred` and `nn_pred` respectively. Note that for prediction based on `nn_full`, make sure to transform the prediction of **standardized scale** `Balance` back to the **original scale**.

```
#prediction of the lm
lm_pred <- predict(lm_full, newdata = test_data)

#prediction of the gam
gam_pred <- predict(gam_full, newdata = test_data)

#prediction of the nn scaled to the income max
nn_pred <- as.vector(balance_max * (predict(nn_full, newdata = x_test_nn)))
```

Q4(b) [2 points] Use the R package `forecast` to evaluate the prediction performance of `lm_full`, `gam_full` and `nn_full`. What are the MAE for `lm_full`, `gam_full` and `nn_full`? (Note that MPE and MAPE may be undefined as some persons have `Balance`=0.)

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
#the accuracy of lm_pred
accuracy(lm_pred, test_data$Balance)
```

```
##                 ME     RMSE      MAE MPE MAPE
## Test set -4.077176 101.7312 84.22484 NaN  Inf
```

```r
#the accuracy of gam_pred
accuracy(gam_pred, test_data$Balance)
```

```
##                  ME     RMSE      MAE MPE MAPE
## Test set -0.4461055 70.19443 51.80754 NaN  Inf
```

```r
#the accuracy of nn_pred
accuracy(nn_pred, x_test_nn$Balance)
```

```
##                 ME     RMSE      MAE MPE MAPE
## Test set 0.4645821 37.62021 29.34589 NaN  Inf
```

Answer:

The MAE of lm_full is 84.22484.

The MAE of gam_full is 51.80754.

The MAE of nn_pred is 29.34589.

Q4(c) [2 points] Which statistical model do you prefer, `lm_full` or `gam_full` or `nn_full`? Give reasons.

Answer:

There are two criteria when we assess a model: accuracy and if the model has a parsimonious structure.

nn_full has the best MAE of 29.34, which makes it almost twice as accurate as gam_full (51.80754) and almost three times as accurate as lm_full at (84.22484)

Usually, you could make the argument that nn_full is not the best model because it has a rather complex structure and is not easily interpreted. This might lead you to choose gam_full as while it is more complicated than lm_full, it has a better performance.

However, because nn_full has a much better MAE, the added complications are worth the increase in accuracy.