Homework6

Steven Black

3/27/2019

IS 677: Introduction to Data Science Spring 2019

Homework Assignment 6 (Due: March 24, 2019, midnight EST)

1. Load the crime data from the website: http://www.andrew.cmu.edu/user/achoulde/94842/data/crime_simple.txt, and store it into a dataframe. Check the structure of the data. (10 points)

[hint:] you can use the command below to load the data.

[`read.table("http://www.andrew.cmu.edu/user/achoulde/94842/data/crime_simple.txt", sep = "\t", header = TRUE)`]

Here are the explanations of the columns of the data:

- R: Crime rate: # of offenses reported to police per million population
- Age: The number of males of age 14-24 per 1000 population
- S: Indicator variable for Southern states (0 = No, 1 = Yes)
- Ed: Mean # of years of schooling x 10 for persons of age 25 or older
- Ex0: 1960 per capita expenditure on police by state and local government
- Ex1: 1959 per capita expenditure on police by state and local government
- LF: Labor force participation rate per 1000 civilian urban males age 14-24
- M: The number of males per 1000 females
- N: State population size in hundred thousands
- NW: The number of non-whites per 1000 population
- U1: Unemployment rate of urban males per 1000 of age 14-24
- U2: Unemployment rate of urban males per 1000 of age 35-39
- W: Median value of transferable goods and assets or family income in tens of $
- X: The number of families per 1000 earning below 1/2 the median income

```
crime_dataframe <- read.table("http://www.andrew.cmu.edu/user/achoulde/94842/data/crime_simple.txt", sep

head(crime_dataframe)
```

```
##       R Age S  Ed Ex0 Ex1  LF    M   N  NW  U1 U2   W   X
## 1  79.1 151 1  91  58  56 510  950  33 301 108 41 394 261
## 2 163.5 143 0 113 103  95 583 1012  13 102  96 36 557 194
## 3  57.8 142 1  89  45  44 533  969  18 219  94 33 318 250
## 4 196.9 136 0 121 149 141 577  994 157  80 102 39 673 167
## 5 123.4 141 0 121 109 101 591  985  18  30  91 20 578 174
## 6  68.2 121 0 110 118 115 547  964  25  44  84 29 689 126
```

2. Create training data using 40 randomly selected rows and test data using the rest. (20 points)

```
crime_dataframe_sample <- crime_dataframe[sample(nrow(crime_dataframe), 40), ]

print(paste("The dimensions of the dataframe are: ", dim(crime_dataframe_sample)))
```

```
## [1] "The dimensions of the dataframe are:  40"
## [2] "The dimensions of the dataframe are:  14"
```

```
head(crime_dataframe_sample)
```

```
##        R Age S  Ed Ex0 Ex1  LF    M   N  NW  U1 U2   W   X
## 16  94.6 142 1  88  81  77 497  956  33 321 116 47 427 247
## 33 107.2 147 1 104  63  64 560  972  23  95  76 24 462 233
## 3   57.8 142 1  89  45  44 533  969  18 219  94 33 318 250
## 11 167.4 124 0 105 121 116 580  966 101 106  77 35 657 170
## 26 199.3 131 0 121 160 143 631 1071   3  77 102 41 674 152
## 35  65.3 123 0 102  97  87 526  948 113  76 124 50 572 158
```

3. The first column in the data (R) is the target variable (the one that we want to predict, while the rest are independent variables that will be used for predicting the target variable. Write R code to Create the independent and target variables. (10 points)

```
target_variable <- crime_dataframe_sample["R"]
dependent_variables <- subset(crime_dataframe_sample, select=-c(R))
```

4. Normalize the data so that they are mean centered and scaled between -1 to 1. (10 points)

```
dependent_variables_normalized <- scale(dependent_variables,
                                        center=apply(dependent_variables, 2, mean),
                                        scale=apply(dependent_variables, 2, sd)
                                        )
summary(dependent_variables_normalized)
```

```
##       Age                S                 Ed                Ex0         
##  Min.   :-1.5424   Min.   :-0.7246   Min.   :-1.8001   Min.   :-1.3743  
##  1st Qu.:-0.7119   1st Qu.:-0.7246   1st Qu.:-0.4299   1st Qu.:-0.7892  
##  Median :-0.2373   Median :-0.7246   Median : 0.1843   Median :-0.2528  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.6921   3rd Qu.: 1.3456   3rd Qu.: 0.6567   3rd Qu.: 0.7063  
##  Max.   : 3.0452   Max.   : 1.3456   Max.   : 1.5072   Max.   : 2.5593  
##       Ex1                LF                  M                 N          
##  Min.   :-1.3218   Min.   :-1.69965   Min.   :-1.7133   Min.   :-0.91030  
##  1st Qu.:-0.7403   1st Qu.:-0.81680   1st Qu.:-0.5929   1st Qu.:-0.67945  
##  Median :-0.2977   Median :-0.05166   Median :-0.1634   Median :-0.27390  
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000  
##  3rd Qu.: 0.5789   3rd Qu.: 0.78541   3rd Qu.: 0.2568   3rd Qu.: 0.05678  
##  Max.   : 2.6011   Max.   : 2.06718   Max.   : 3.4032   Max.   : 3.20758  
##        NW                 U1                 U2                 W          
##  Min.   :-0.9975   Min.   :-1.4346   Min.   :-1.65126   Min.   :-2.4038  
##  1st Qu.:-0.7410   1st Qu.:-0.8438   1st Qu.:-0.71545   1st Qu.:-0.5955  
##  Median :-0.2755   Median :-0.1318   Median : 0.03924   Median : 0.1915  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000  
##  3rd Qu.: 0.3657   3rd Qu.: 0.5196   3rd Qu.: 0.52225   3rd Qu.: 0.8815  
##  Max.   : 3.0019   Max.   : 2.9284   Max.   : 2.93725   Max.   : 1.5913  
##        X          
##  Min.   :-1.6344  
##  1st Qu.:-0.6900  
##  Median :-0.4324  
##  Mean   : 0.0000  
##  3rd Qu.: 0.8126  
##  Max.   : 2.0454
```

```
target_variable_normalized <- scale(target_variable,
                                    center=apply(target_variable, 2, mean),
```

```
                                    scale=apply(target_variable, 2, sd)
                                    )
summary(target_variable_normalized)
```

```
##        R
##  Min.   :-1.4848
##  1st Qu.:-0.6828
##  Median :-0.2338
##  Mean   : 0.0000
##  3rd Qu.: 0.3987
##  Max.   : 2.6627
```

5. Build a network with two dense hidden layers. Choose the number of units to avoid overfitting. Select 50 epochs. (20 points)

```
library(keras)

build_model <- function() {
  model <- keras_model_sequential() %>%
    layer_dense(units = 64, activation = "relu",
                input_shape = dim(dependent_variables_normalized)[[2]]
                ) %>%
    layer_dense(units = 64,
                activation = "relu"
                ) %>%
    layer_dense(units = 1)

  model %>% compile(
    optimizer = "rmsprop",
    loss = "mse",
    metrics = c("mae")
  )
}
```

6. Validate the model using k-fold cross validation where k=4. (20 points)

```
k <- 4
indices <- sample(1:nrow(dependent_variables_normalized))
folds <- cut(indices, breaks = k, labels = FALSE)

num_epochs <- 50
all_mae_histories <- NULL
for (i in 1:k) {
  cat("processing fold #", i, "\n")

  val_indices <- which(folds == i, arr.ind = TRUE)
  val_data <- dependent_variables_normalized[val_indices,]
  val_targets <- target_variable_normalized[val_indices]

  partial_train_data <- dependent_variables_normalized[-val_indices,]
  partial_train_targets <- target_variable_normalized[-val_indices]

  model <- build_model()

  history <- model %>% fit(
```

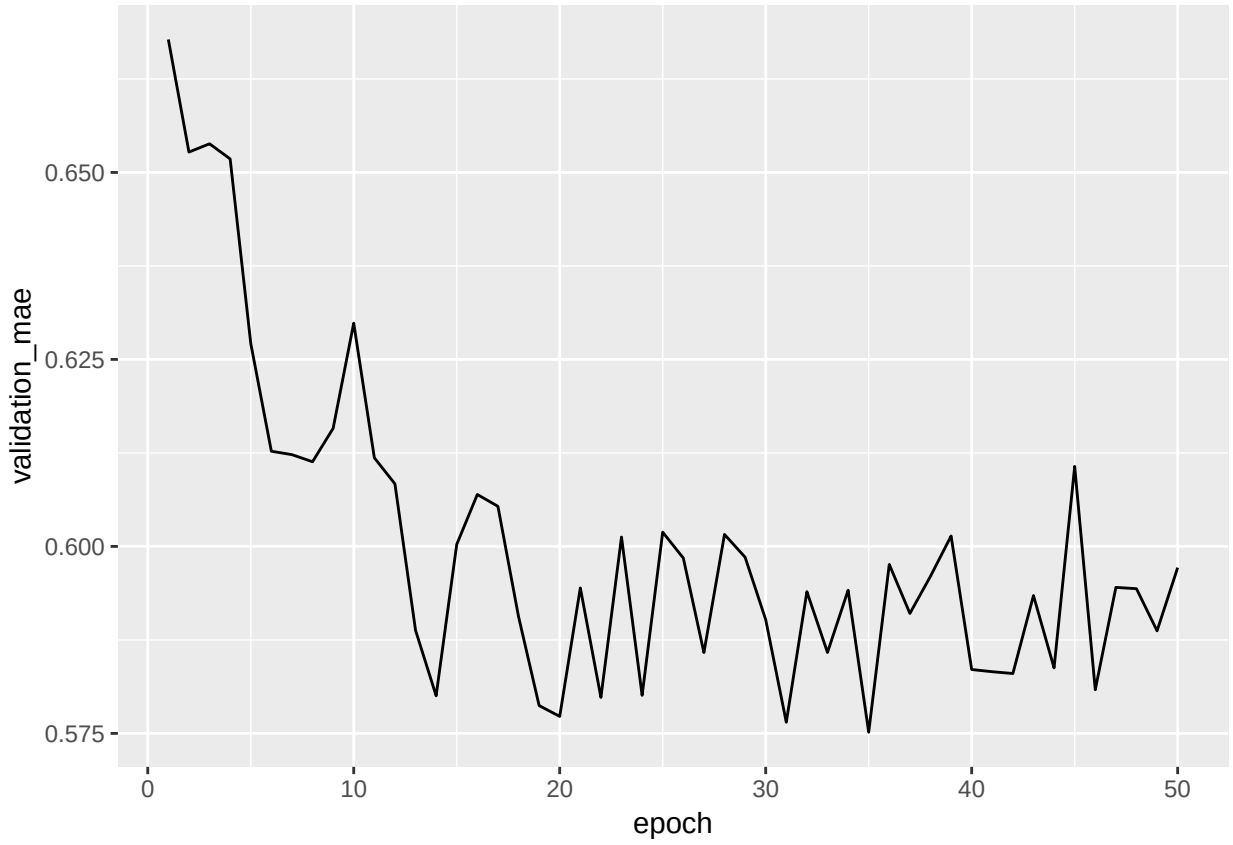
```
    partial_train_data,
    partial_train_targets,
    validation_data = list(val_data, val_targets),
    epochs = num_epochs, batch_size = 1, verbose = 0
  )
  mae_history <- history$metrics$val_mean_absolute_error
  all_mae_histories <- rbind(all_mae_histories, mae_history)
}
```

```
## processing fold # 1
## processing fold # 2
## processing fold # 3
## processing fold # 4
```

7. Compute the average of the per-epoch MAE scores for all folds. Plot the MAEs. When does the model start overfitting? (Note that you should clear previous sessions to avoid overestimating the model performance.) (10 points)

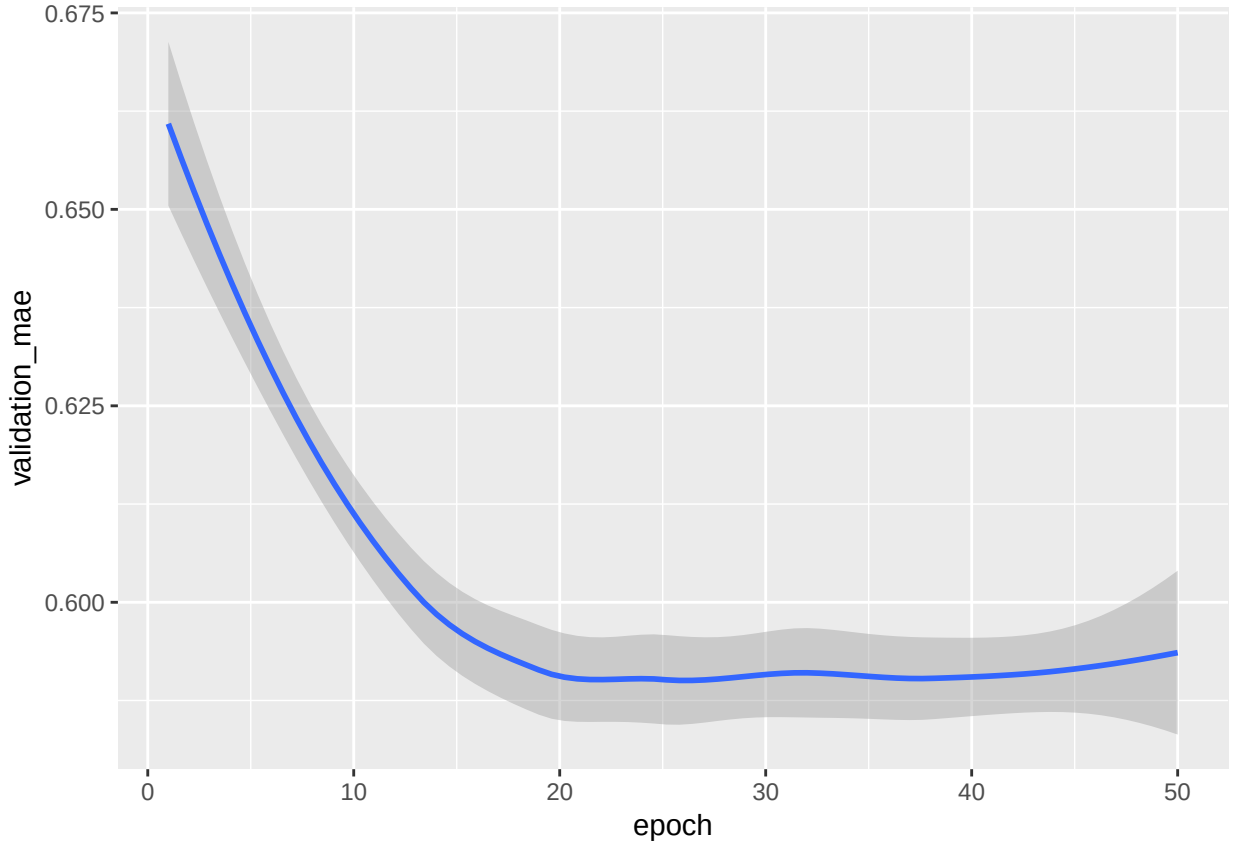
```
k_clear_session()

average_mae_history <- data.frame(
  epoch = seq(1:ncol(all_mae_histories)),
  validation_mae = apply(all_mae_histories, 2, mean)
)
```

```
library(ggplot2)
ggplot(average_mae_history, aes(x = epoch, y = validation_mae)) + geom_line()
```



```
ggplot(average_mae_history, aes(x = epoch, y = validation_mae)) + geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



According to this plot, validation MAE stops improving significantly after the below value. Past that point, you start overfitting.

```
with(average_mae_history, epoch[validation_mae == min(validation_mae)])
```

```
## [1] 35
```