Gradient Cost Function

Jose Carlos Montes

8/11/2019

```
library(readr)
library(ggplot2)
```

We will initiate by importing the libraries and data required:

```
mdata <- read_csv("C:/Users/josec/Dropbox/_CUNEF/Machine Learning/machine_learning_basics/data/4_1_data
```

```
## Parsed with column specification:
## cols(
##   `score-1` = col_double(),
##   `score-2` = col_double(),
##   label = col_double()
## )
```

```
head(mdata)
```

```
## # A tibble: 6 x 3
##   `score-1` `score-2` label
##       <dbl>     <dbl> <dbl>
## 1      34.6      78.0     0
## 2      30.3      43.9     0
## 3      35.8      72.9     0
## 4      60.2      86.3     1
## 5      79.0      75.3     1
## 6      45.1      56.3     0
```

After giving a quick look to the data we can appreciate that we have 2 scores and one label, which is our target variable.

Now we will create the Sigmoid and the Gradient Descent test funtions which we are using to optimize the parameters:

```
set.seed(123)
Sigmoid <- function(x) {
  1 / (1 + exp(-x))
}
Cost_function <- function(parameters, X, Y) {
  n <- nrow(X)
  g <- Sigmoid(X %*% parameters)
  J <- (1/n) * sum((-Y * log(g)) - ((1 - Y) * log(1 - g)))
  return(J)
}
Gradient_Descent_test <- function(iterations = 1200, X, Y) {

  parameters <- rep(x = 0, times = 3)
```

```
  print(paste("Initial Cost Function value: ",
              convergence <- c(Cost_function(parameters, X, Y)), sep = ""))

  parameters_optimization <- optim(par = parameters, fn = Cost_function, X = X, Y = Y,
                                   control = list(maxit = iterations))


  parameters <- parameters_optimization$par


  print(paste("Final Cost Function value: ",
              convergence <- c(Cost_function(parameters, X, Y)), sep = ""))
 return(parameters)
}
```

Additionally we now create the funtion which will help us evaluate the confusion matrix:

```
Matrix <- function(X, Y, parameters, cutoff = 0.70) {
 tabla.res <- NA


 for (i in 1:nrow(X)) {
    res <- Sigmoid(t(as.numeric(X[i,])) %*% parameters)
    tabla.res <- rbind(tabla.res, res)
 }
 tabla.res <- tabla.res[-1,]


 matrix_table <- table(Y, ifelse(tabla.res > cutoff, 1, 0))

 return(matrix_table)
}
```

1. Test the TestGradientDescent function with the training set (4_1_data.csv). Obtain the confusion matrix.

```
X <- as.matrix(mdata[, c(1,2)])
X <- cbind(rep(1, nrow(X)), X)
Y <- as.matrix(mdata$label)
par.optimos <- Gradient_Descent_test(X = X, Y = Y)
```

```
## [1] "Initial Cost Function value: 0.693147180559945"
## [1] "Final Cost Function value: 0.203497704580909"
```

```
Matrix(X = X, Y = Y, par.optimos, cutoff = 0.80)
```

```
##
## Y    0  1
##   0 37  3
##   1 11 49
```

We are creating a function that will make 500 iterations in order to find the optim cost:

```
Cost_graph <- function(max_iterations = 500, X, Y) {

  graph.table <- data.frame(cbind(seq(from = 1, to = max_iterations, by = 1), NA))
  names(graph.table) <- c("Iteration", "Cost")

  parameters <- rep(x = 0, times = 3)
  iterations <- 0

  for (i in 1:max_iterations) {
    parameters_optimization <- optim(par = parameters, fn = Cost_function, X = X, Y = Y,
                                     control = list(maxit = iterations))
    iterations <- iterations + 1
    graph.table[i, 2] <- Cost_function(parameters_optimization$par, X, Y)
  }
  plot <- ggplot(data = graph.table, aes(x = Iteration, y = Cost)) +
    geom_smooth(se = FALSE)
  return(plot)
}
```

Now we can graph the corresponding iterations:

```
Cost_graph(max_iterations = 500, X = X, Y = Y)
```