machine-learning-ex1

Instructions

This file contains code that helps you get started on the linear exercise. You will need to complete the following functions in this exericse:

1. warmUpExercise.R
2. plotData.R
3. gradientDescent.R
4. computeCost.R
5. featureNormalize.R
6. normalEqn.R

x refers to the population size in 10,000s
y refers to the profit in $10,000s

Initialization

```
rm(list=ls())
Rfunctions <- c("warmUpExercise.R","plotData.R","gradientDescent.R","computeCost.R","featureNormalize.R
for (i in 1:length(Rfunctions)) {
  source(Rfunctions[i])
}
```

———— Part 1: Basic Function ————

Complete warmUpExercise.R

Running warmUpExercise 5x5 Identity Matrix:

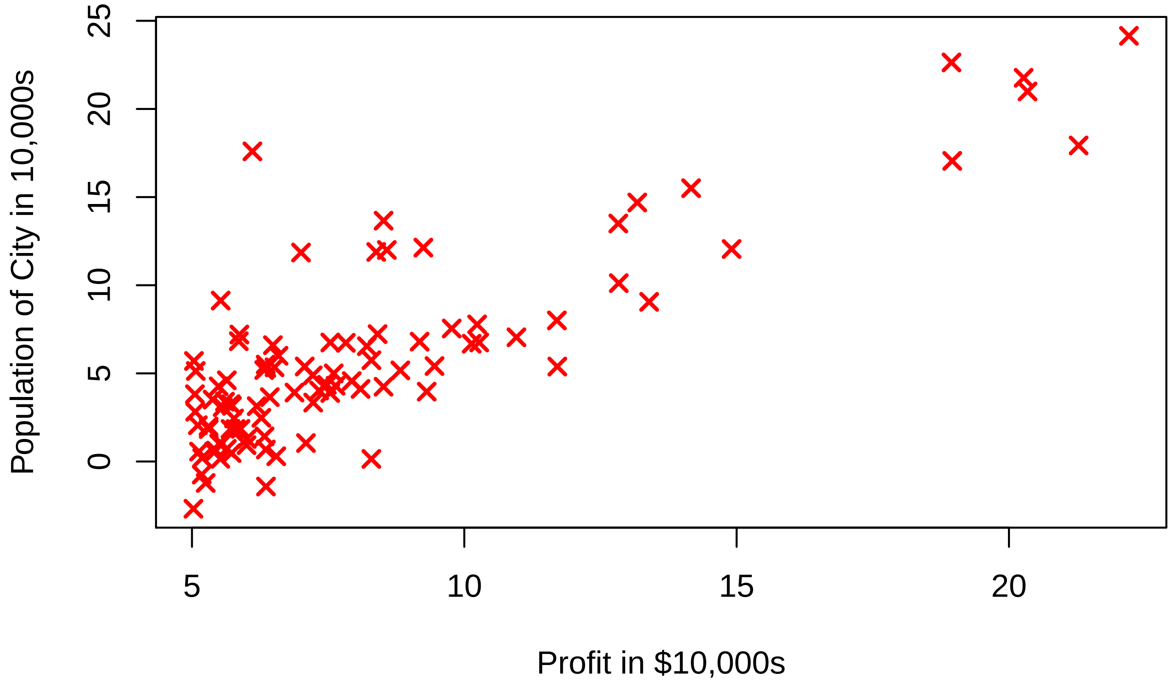
```
warmUpExercise()
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    1    0    0    0
## [3,]    0    0    1    0    0
## [4,]    0    0    0    1    0
## [5,]    0    0    0    0    1
```

———— Part 2: Plotting ————

Plotting Data …

```
data <- read.table("/home/jayendra/iProjects/MOOCs/Machine Learning/Solutions/machine-learning-ex1/ex1/
X <- data[, 1]
y <- data[, 2]
m <- length(y) # number of training examples
plotData(X, y)
```



————————- Part 3: Gradient descent ————————-

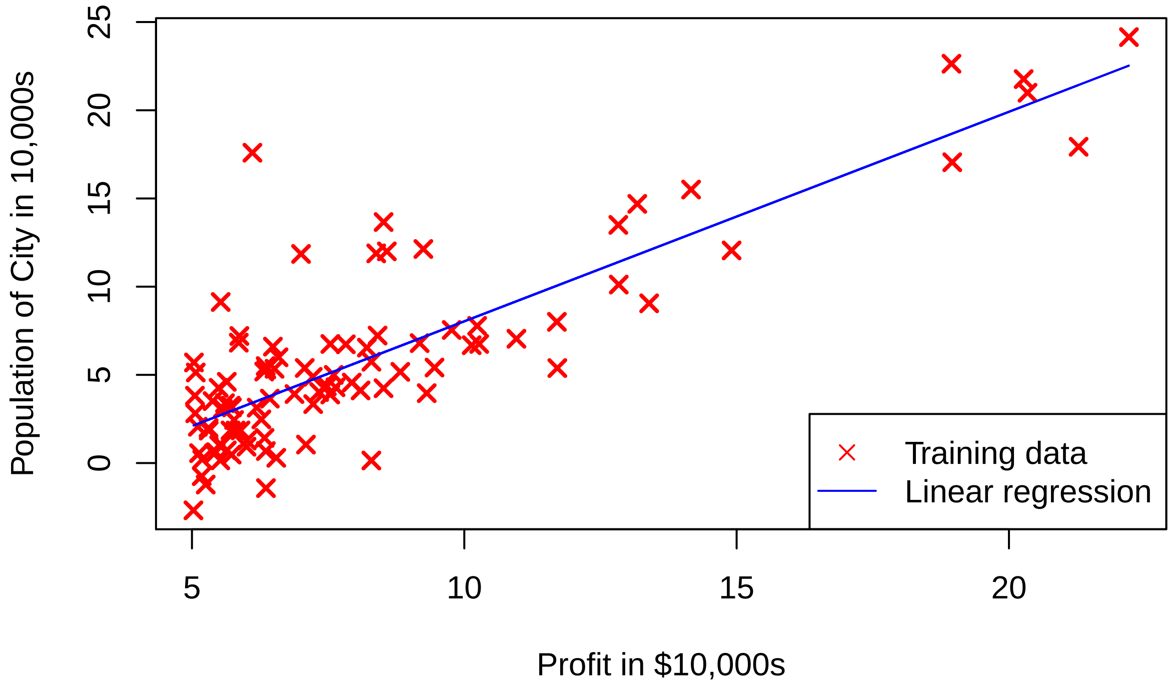
```
X <- cbind(rep(1,m),X) # Add a column of ones to x
X <- as.matrix(X)
# initialize fitting parameters
theta <- c(8,3)
# Some gradient descent settings
iterations <- 1500
alpha <- 0.02
# compute and display initial cost
computeCost(X, y, theta)
```

```
##          [,1]
## [1,] 383.526
```

```
# run gradient descent
gd <- gradientDescent(X, y, theta, alpha, iterations)
#Decompose list (gd) variables into global env variables
theta <- gd$theta
J_history <- gd$J_history
theta_history <- gd$theta_history
rm(gd)
# print theta to screen
#'Theta found by gradient descent:
cat(sprintf('%f %f \n', theta[1], theta[2]))
```

```
## -3.844313 1.187863
```

```
# Plot the linear fit
plotData(X[,2], y)
lines(X[, 2], X  %*% theta, col="blue")
legend("bottomright", c('Training data', 'Linear regression'), pch=c(4,NA),col=c("red","blue"), lty=c(NA
```



```
# Predict values for population sizes of 35,000 and 70,000
predict1 <- c(1, 3.5) %*% theta
cat(sprintf('For population = 35,000, we predict a profit of %f\n',predict1*10000))
```

```
## For population = 35,000, we predict a profit of 3132.080736
```

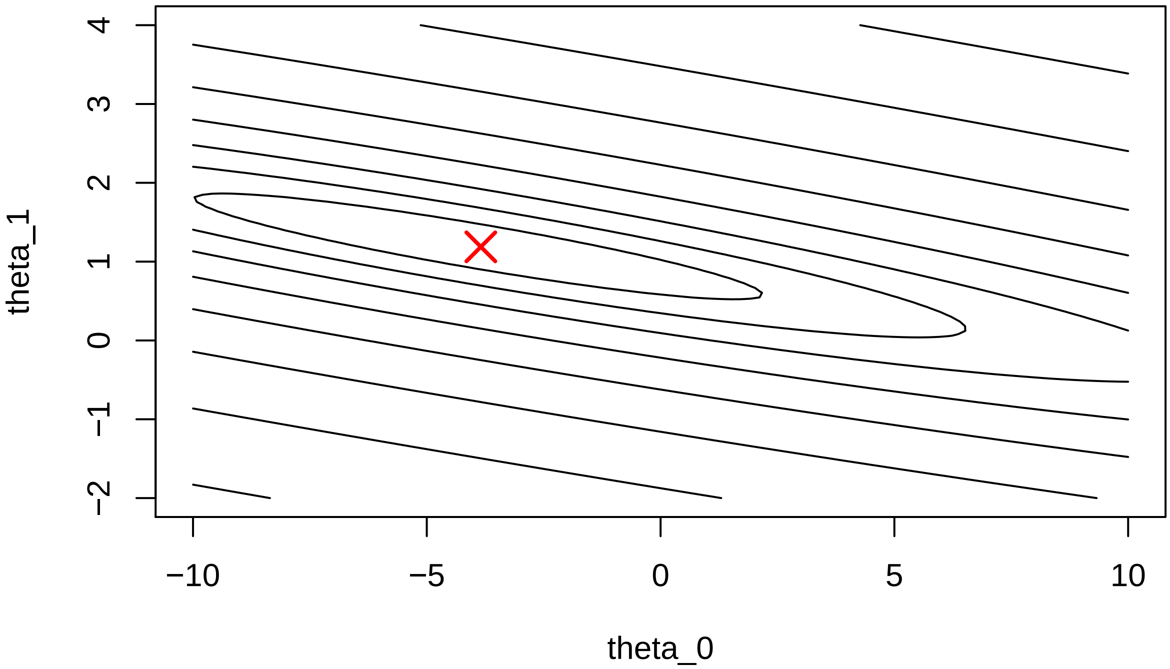
```
predict2 <- c(1, 7) %*% theta
cat(sprintf('For population = 70,000, we predict a profit of %f\n',predict2*10000))
```

```
## For population = 70,000, we predict a profit of 44707.290050
```

————- Part 4: Visualizing J(theta0,theta1) ————-

```
# Grid over which we will calculate J
theta0_vals <- seq(-10, 10, length.out=100)
theta1_vals <- seq(-2, 4, length.out=100)

# initialize J_vals to a matrix of 0's
J_vals <- matrix(0,length(theta0_vals), length(theta1_vals))

# Fill out J_vals
for (i in 1:length(theta0_vals)) {
    for (j in 1:length(theta1_vals)) {
```

```r
      J_vals[i,j] <- computeCost(X, y, c(theta0_vals[i], theta1_vals[j]))
    }
}

#interactive 3D plot
#install.packages("rgl")
library(rgl)
#open3d()

nbcol = 100
color = rev(rainbow(nbcol, start = 0/6, end = 4/6))
J_vals_col  = cut(J_vals, nbcol)

persp3d(theta0_vals, theta1_vals, J_vals,col = color[J_vals_col],
        xlab=expression(theta_0),ylab=expression(theta_1),
        zlab="Cost",main = "Gradient Descent")
points3d(theta_history[, 1], theta_history[, 2], J_history+10,
         col="red",size=3.5)
lines3d(theta_history[, 1], theta_history[, 2], J_history+10, col="red")

# Contour plot
# Plot J_vals as 20 contours spaced logarithmically between 0.01 and 100
# logarithmic contours are denser near the center
logspace <- function( d1, d2, n)
            return(exp(log(10)*seq(d1, d2, length.out=n)))
            #or return(10^seq(d1, d2, length.out=n))

contour(theta0_vals, theta1_vals, J_vals, levels = logspace(-2, 3, 20),
        xlab=expression(theta_0),
        ylab=expression(theta_1),
        drawlabels = FALSE)

points(theta[1], theta[2], pch=4, cex=2,col="red",lwd=2)
```


theta_1
4
3
2
1
0
−1
−2
−10
−5
0
5
10
theta_0