University of Manitoba

ECE 3790: Engineering Algorithms **Winter 2021**

Assignment 2: Gradient Descent

April 6, 2021

The purpose of this assignment is to give you some experience with Gradient Descent.

Important: This assignment can be performed in groups of two and feel free to interact with your classmates.

Each group is responsible for turning in an assignment report that includes the answers requested at the end of the Problem Statement and a titlepage.

Important: Please include the following signed statement with your submission.

I (We), [insert name(s)] attest that the work I am (we are) submitting is my (our) own work and that it has not been copied/plagiarized from online or other sources. Any sourced material used for completing this work has been properly cited. [Signature(s)]

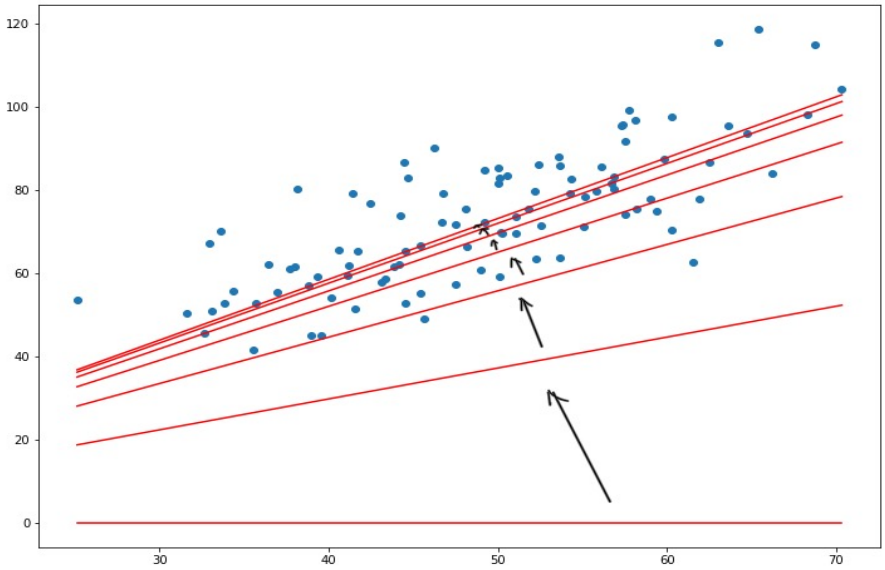
Also Important: Assignments done in pairs must be submitted with a short paragraph outlining each partner's contribution.

Objective Functions, Gradients, and Step Sizes

I, Emma Tran attest that the work I am submitting is my own work and that it has not been copied/plagiarized from online or other sources. Any sourced material used for completing this work has been properly cited.

Discussion Questions

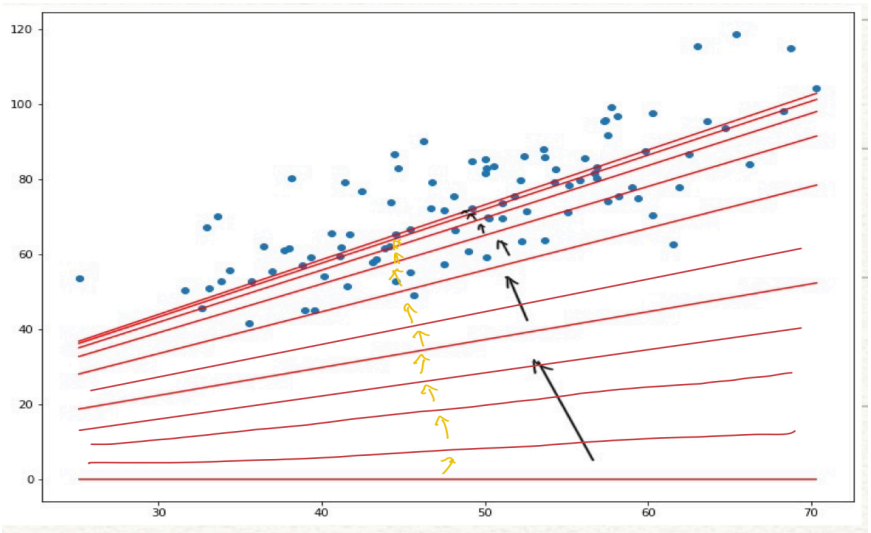
1. What validation have you provided to show that your gradient descent implementation works? Discuss the plots you are submitting.



From the plot we can see that the regression line started at a location really far from the points of data set. Then after each iteration, it got closer and closer to the center of these points. By visualizing, we can stated that the program worked since the regression line was converging to be the best-fit line.

2. Discuss the difference in performance between the fixed step-size α and the analytic α in terms of the numbers of iterations to converge to the solution. How did you choose the fixed step-size?

The iterations when we use the analytic alpha is more efficient (i.e its looped in a smaller number of time). Because when we use the fixed step-size the line only can move in a small step of the d direction instead of go all the way along that direction.



We can see the concept of fixed steps demonstrated in yellow arrow compared to black arrow.

I started at alpha = 0.1 and tried different steps size until it get to a nice plot. I also print out a0 and a1 along side so I know I should increase or decrease the step size for next try.

3. Did your algorithm converge for every choice of fixed step-size? If not, what causes the algorithm to diverge?
   My algorithm did not converge for every choice of fixed step-size. The algorithm can be diverge when the step size is either too small or too big. The reason is that, if the steps size get too big then it could just pass through the right region without noticing and cannot go back. On another hand, if the step size is too small, the change will be minor and it also can get to the wrong direction.

4. Can we always use an analytic value of α when minimizing a general function $f(\underline{x})$? Why or why not?
   In some complicated situation, we cannot always use an analytic value of α when minimizing a general function $f(\underline{x})$. If analytic value of alpha is provided then we can use it to minimize the general function f(x). However, there will be some situation when the function is too complicated too find an analytic value of alpha.

5. Suppose you didn't have a closed-form expression for the objective function, that is you only have a black box that you can pass inputs into and get function evaluations out of. Can you still apply Gradient Descent? If so, how? (Hint: can you approximate derivatives numerically? If so how?).
   If we only have a black box, we will still be able to apply the Gradient Descent. We can approximated derivatives using the definition of derivative method where we have the derivative can be calculated as[1]

$$\frac{f(x+h)-f(x)}{h}.$$

Appendix:

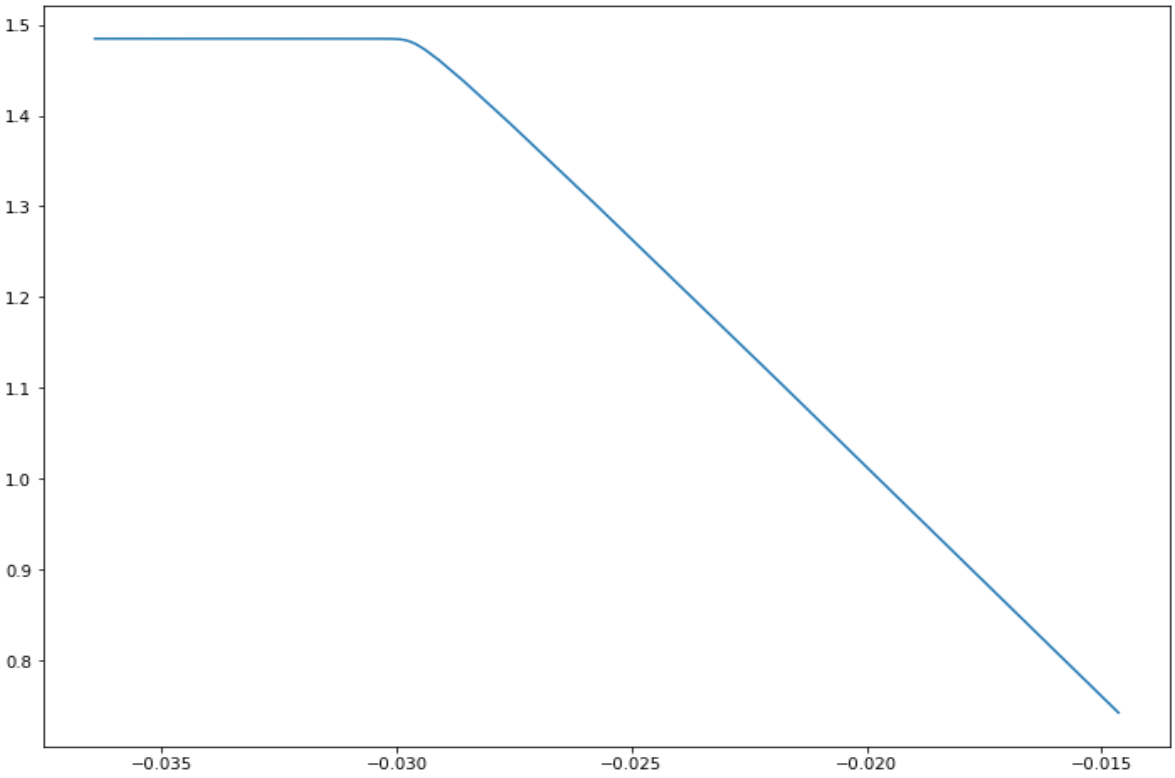
- I denoted my $\underline{x}$ as $x\_$, $\underline{b}$ as $b\_$ and A^T as At and r^T as Rt
- I have 2 version of code since one of them is not work properly so I try different way.

Reference:

1. https://en.wikipedia.org/wiki/Numerical_differentiation#/media/File:Derivative.svg
2. https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931

```
import csv
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

#Input data
data = list(csv.reader(open("data.csv")))
data = pd.read_csv('data.csv')
A = data.iloc[:, :2]
a = data.iloc[:,1:2]
print(type(a))
b = data.iloc[:,-1:]

#initialize guess a1 and a0 x_ = [a0 a1]

x_ = np.array([[0],[0]])
a0 = x_[0]
a1 = x_[1]

#calculate d
Ax = np.dot(A,x_)
r = Ax - b #r_ = Ax_ - b_
At = np.transpose(A) #A^T
gradient = 2 * np.dot(At,r) #2.A^T.r_
d = -gradient #d=gradient at x_ = [0 0]

# The fixed step size alpha
alpha = 0.00000001
max_iteration = 1000  # The max number of iteration

#analytic alpha
Art = np.dot(np.transpose(r),A) #A.r^T
alpha = np.dot(Art,d) / (np.dot(A,d))**2

for i in range(max_iteration):
    Y_pred = -a0 + a1*a #keep track with the regression line
    d = -2 * np.dot(At,np.dot(A,x_)-b)              # The predicted value of Y
    x_ = x_ + alpha*d #update x_ aka [a0 a1]
    plt.plot([min(a), max(a)], [min(Y_pred), max(Y_pred)], color='red')  # regression line
    plt.plot(x_[0],x_[1])

plt.scatter(a, b)
print(-x_[0],x_[1])


plt.show()
```

```
import pandas as pd
import matplotlib.pyplot as plt

#Input data
data = pd.read_csv('data.csv')
X = data.iloc[:,1]
Y = data.iloc[:,2]


#initialize a1 and a0
a0 = 0
a1 = 0

alpha = 0.0001  # The fixed step size alpha
max_iteration = 6  # The max number of iteration

n = float(len(X)) # length of X
al1=[]
al0=[]
for i in range(max_iteration):
    Y_pred = a0 + a1*X  # The predicted value of Y

    #sum (yi - (a0+a1x)**2) derivative wrt to a1
    D_a1 = (2/n) * sum(X * (Y - Y_pred))  # Gradient - Derivative wrt a0
    #sum (yi - (a0+a1x)**2) derivative wrt to a0
    D_a0 = -(2/n) * sum(Y - Y_pred)  # Gradient - Derivative wrt a1
    a0 = a0 + alpha * D_a0  # Update a0
    al0.append(a0)
    a1 = a1 + alpha * D_a1  # Update a1
    al1.append(a1)
    plt.plot([min(X), max(X)], [min(Y_pred), max(Y_pred)], color='red')  # regression line


print(a0)
print(a1)
#plt.plot(al0,al1)
Y_pred = a0 + a1*X
plt.scatter(X, Y)
#plt.plot([min(X), max(X)], [min(Y_pred), max(Y_pred)], color='red')  # regression line
plt.show()
```


1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
−0.035
−0.030
−0.025
−0.020
−0.015