Beryl Sin

PeopleSoft ID: 4522433

01/24/2024

Problem Set 2

1. See computeCost.py for the computeCost function and ps2.py for other code

```
Question 1 (i) Cost :  [[8.4375]]
Question 1 (ii) Cost :  [[50.]]
Question 1 (iii) Cost :  [[15.]]
```

2. See gradientDescent.py for the gradientDescent function and ps2.py for other code

```
Question 2 results :
Cost after each iteration:
 [[8.340886109302165]
 [8.340661225397694]
 [8.340443037411477]
 [8.340231345972551]
 [8.340025957646217]
 [8.339826684757302]
 [8.339633345218653]
 [8.339445762364766]
 [8.339263764790356]
 [8.339087186193725]
 [8.338915865224806]
 [8.338749645337737]
 [8.338588374647806]
 [8.338431905792667]
 [8.33828009579769]]

Estimated Theta:
 [[1.76405235]
 [0.39553624]
 [0.97411702]]
```

3. See normalEqn.py for the normalEqn function and ps2.py for other code

```
Question 3 theta:
 [[0.        ]
 [0.66666667]
 [0.66666667]]
```

There is a noticeable difference between the two estimates for theta because the estimate obtained from gradient descent changes gradually over iterations whereas the estimate obtained from using the normal equation approach does not use a learning rate and is more of an analytical approach. To achieve similar results, more iterations of gradient descent should be used.

4. a) See ps2.py for code

b) See ps2-4-b.png

c) See ps2.py for code

```
Question 4C X.shape:  (179, 2)
Question 4C y.shape:  (179, 1)
```

(row, col)

d) See ps2.py for code

e) See ps2-4-e.png

```
Question 4E theta :
 [[-5.894261105176994]
 [17.799745118092844]]
```

theta = [theta_0; theta_1]

f) See ps2-4-f.png

g) See ps2.py for code

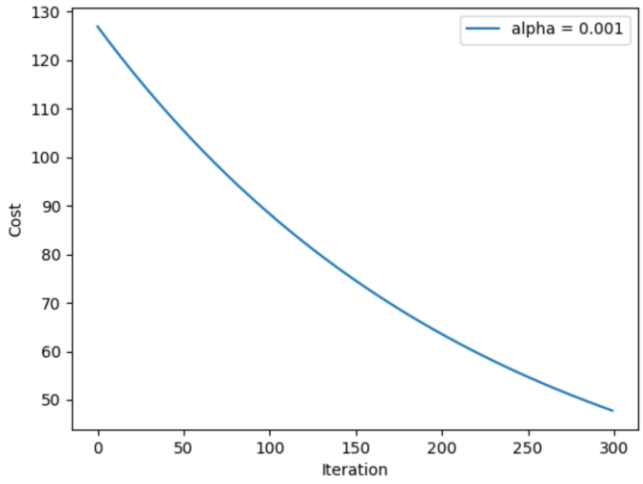
```
Question 4G Cost :  [[4.562787008188487]]
```

h) See ps2.py for code

```
Question 4H Cost :  [[4.562917221858439]]
```

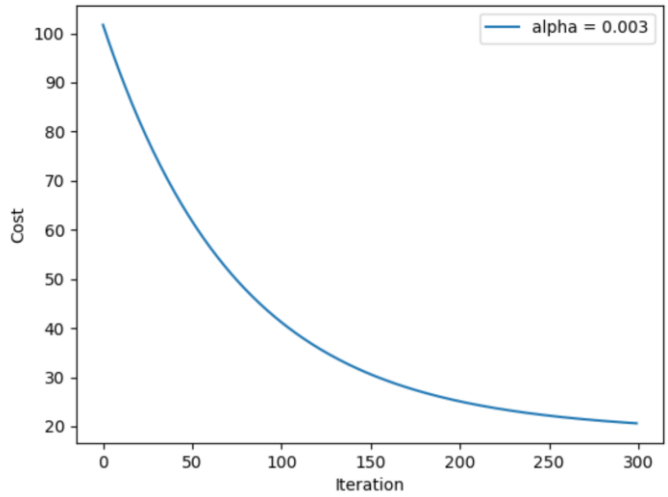
i)

ps2-4-i-1.png



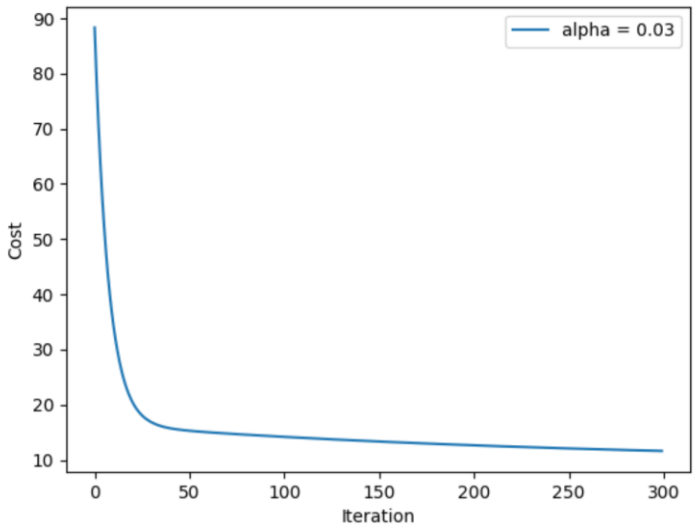
Gradually decreases cost (not very fast, learning rate too low), will take very long to converge.

ps2-4-i-2.png



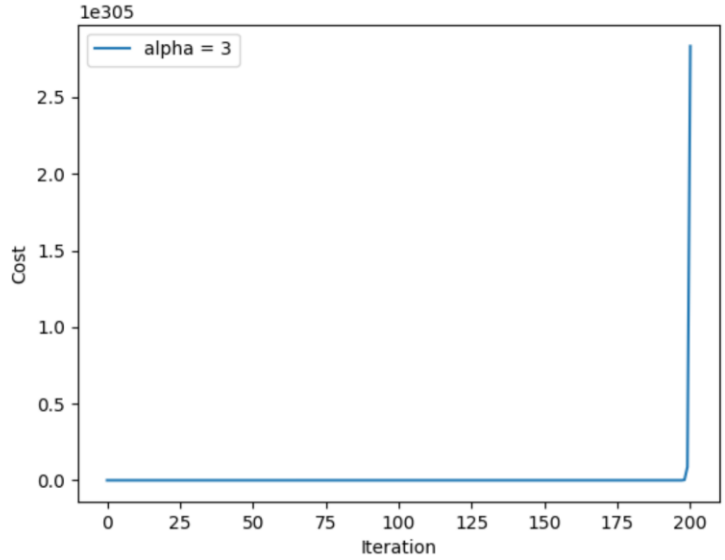
Cost decreases faster with a slightly higher alpha (good).

ps2-4-i-3.png



Cost decreases even faster with a higher alpha, runs the risk of overshooting the minima

ps2-4-i-4.png



Cost increases, gradient descent diverges with very high alpha, learning rate too high.

5. a) See ps2.py for code

```
Question 5A Engine Size Statistics :
Question 5A Mean :  1611.111111111111
Question 5A Standard Deviation :  383.5345687576268
Question 5A Weight Statistics:
Question 5A Mean :  1292.2777777777778
Question 5A Standard Deviation :  238.73737443185826
Question 5A X.shape :  (36, 3)
Question 5A y.shape :  (36, 1)
```

b) See ps2.py for code and also ps2-5-b.png

```
Question 5B theta :
 [[101.97392098930213]
 [2.682315797335005]
 [2.1139571306056393]]
```

c) See ps2.py for code

```
Question 5C Prediction :  [106.86016366245306]
```