Edison Chen, ebc5802
10/5/2023

Problem 1

1) $J(w) = \frac{1}{2m} \sum_{i=1}^{m} [w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + w_3 x_3^{(i)} - y^{(i)}]^2$

$m = 4 \rightarrow$ 4 data points

$$\begin{aligned} J(w) = \frac{1}{8}\Big[&(w_0 + 1600w_1 + 1770w_2 + 3w_3 - 330)^2 \\ &+ (w_0 + 2400w_1 + 2740w_2 + 3w_3 - 369)^2 \\ &+ (w_0 + 1416w_1 + 1634w_2 + 2w_3 - 232)^2 \\ &+ (w_0 + 3000w_1 + 3412w_2 + 4w_3 - 540)^2 \Big] \end{aligned}$$

2) $\frac{\partial J(w)}{\partial w_j} = \frac{1}{m} \sum_{i=1}^{m} [w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + w_3 x_3^{(i)} - y^{(i)}] x_j^{(i)}$

$w_j = w_j - \alpha \frac{\partial J(w)}{\partial w_j}$

Initialized to 0

$= 0 - 0.1 \frac{\partial J(w)}{\partial w_j}$

$$\begin{aligned} w_0 = (-0.1) \cdot \frac{1}{4}\Big[&(w_0 + 1600w_1 + 1770w_2 + 3w_3 - 330) \\ &+ (w_0 + 2400w_1 + 2740w_2 + 3w_3 - 369) \\ &+ (w_0 + 1416w_1 + 1634w_2 + 2w_3 - 232) \\ &+ (w_0 + 3000w_1 + 3412w_2 + 4w_3 - 540) \Big] \end{aligned}$$

$$\begin{aligned} w_1 = (-0.1) \cdot \frac{1}{4}\Big[&(w_0 + 1600w_1 + 1770w_2 + 3w_3 - 330)\,1600 \\ &+ (w_0 + 2400w_1 + 2740w_2 + 3w_3 - 369)\,2400 \\ &+ (w_0 + 1416w_1 + 1634w_2 + 2w_3 - 232)\,1416 \\ &+ (w_0 + 3000w_1 + 3412w_2 + 4w_3 - 540)\,3000 \Big] \end{aligned}$$

$$\begin{aligned} w_2 = (-0.1) \cdot \frac{1}{4}\Big[&(w_0 + 1600w_1 + 1770w_2 + 3w_3 - 330)\,1770 \\ &+ (w_0 + 2400w_1 + 2740w_2 + 3w_3 - 369)\,2740 \\ &+ (w_0 + 1416w_1 + 1634w_2 + 2w_3 - 232)\,1634 \\ &+ (w_0 + 3000w_1 + 3412w_2 + 4w_3 - 540)\,3412 \Big] \end{aligned}$$

$$\begin{aligned} w_3 = (-0.1) \cdot \frac{1}{4}\Big[&(w_0 + 1600w_1 + 1770w_2 + 3w_3 - 330)\,3 \\ &+ (w_0 + 2400w_1 + 2740w_2 + 3w_3 - 369)\,3 \\ &+ (w_0 + 1416w_1 + 1634w_2 + 2w_3 - 232)\,2 \\ &+ (w_0 + 3000w_1 + 3412w_2 + 4w_3 - 540)\,4 \Big] \end{aligned}$$

Problem 2

Part A

a) $\text{Mean} = \bar{x} = \frac{1}{m}(x_1 + x_2 + x_3 + \ldots + x_m)$

$$\text{Sample Standard Deviation} = \sqrt{\frac{\sum_{i=1}^{m} (x_i - \bar{x})^2}{m}}$$

b) The mean and standard deviation of [1, 2, 3, 4, 5] is (3.0, 1.4142135623730951

```python
def mean_and_std(x):
    n = len(x)
    mean = sum(x) / n
    squared_diff = [(element - mean) ** 2 for element in x]
    sample_std = math.sqrt(sum(squared_diff) / n)
    return mean, sample_std
```

c)

```
area,bedrooms,price
0.13141542202104753,-0.22609336757768828,0.48089022542412296
-0.5096406975906851,-0.22609336757768828,-0.08498337959656438
0.5079086986184144,-0.22609336757768828,0.23109744835070525
-0.743677058718778,-1.5543919020966084,-0.8763980357612113
1.2710707457752388,1.1022051669412318,1.6126374354654975
-0.019945050665056006,1.1022051669412318,-0.32750063889114467
-0.5935885227779358,-0.22609336757768828,-0.20624200924385452
-0.7296857545209029,-0.22609336757768828,-1.1431751048938927
-0.7894667815481874,-0.22609336757768828,-1.038076208624265
-0.6444659925883908,-0.22609336757768828,-0.7915169950081082
-0.07718220420181784,1.1022051669412318,-0.8117348505246331
-0.0008659994861353915,-0.22609336757768828,0.053251458201346386
-0.14077904146488657,-0.22609336757768828,-0.08418307264089227
3.15099325527155,2.430503701460152,2.9060628183699255
-0.9319236970174614,-0.22609336757768828,-0.6508569846172517
0.38071502409227687,1.1022051669412318,0.8850856575817567
-0.8657829862638698,-1.5543919020966084,-0.32750063889114467
-0.9726256728658254,-0.22609336757768828,-1.1358915032064123
0.7737434783780416,1.1022051669412318,1.2900733127864195
1.3105007848783414,1.1022051669412318,2.090396436275821
-0.29722726113203557,-0.22609336757768828,-0.7074443451193204
-0.1433229149554093,-1.5543919020966084,-0.6904681369686998
-0.5045529506096396,-0.22609336757768828,-0.7882834315508472
-0.049199595806067614,1.1022051669412318,-0.6508569846172517
2.403094449057862,-0.22609336757768828,1.8874903293326886
-1.1456090702213721,-0.22609336757768828,-0.7316960710487784
-0.6902557154178003,-0.22609336757768828,1.0031107237717858
0.6681727285213475,-0.22609336757768828,1.0394883126659729
0.2535213495661395,-0.22609336757768828,1.0879917645248889
0.80935770724536,-0.22609336757768828,-0.32750063889114467
-0.20564781547321664,-1.5543919020966084,0.07669479326648915
-1.2728027447475097,-2.8826904366155284,-1.3784087625009924
0.05001147032431958,1.1022051669412318,-0.20624200924385452
1.4453260798760472,-0.22609336757768828,1.9359937811916046
-0.24126204434053514,1.1022051669412318,-0.4406753598952821
-0.7169663870682891,-0.22609336757768828,-0.7316960710487784
```

Part B

B

a) $J(w) = \frac{1}{2m} \sum_{i=1}^{m} [w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} - y^{(i)}]^2$

b) Testing alpha = 0.01!
Cycle #10 Loss: 0.4190631611865087
Cycle #20 Loss: 0.3584493240307891
Cycle #30 Loss: 0.31280298354690234
Cycle #40 Loss: 0.27820281599944013
Cycle #50 Loss: 0.25177563171317424
Cycle #60 Loss: 0.2314144109083103
Cycle #70 Loss: 0.21557234626074678
Cycle #80 Loss: 0.20311239064687756
Total # of Cycles (Alpha = 0.01): 80
Gradient Time Elapsed (alpha = 0.01): 0.0011870861053466797

Testing alpha = 0.03!
Cycle #10 Loss: 0.3111284832782255
Cycle #20 Loss: 0.23005399557327416
Cycle #30 Loss: 0.19233243026847216
Cycle #40 Loss: 0.17278233972198603
Cycle #50 Loss: 0.1613869107507196
Cycle #60 Loss: 0.15402830358982716
Cycle #70 Loss: 0.14891520392518032
Cycle #80 Loss: 0.1451979361027959
Total # of Cycles (Alpha = 0.03): 80
Gradient Time Elapsed (alpha = 0.03): 0.0011720657348632812

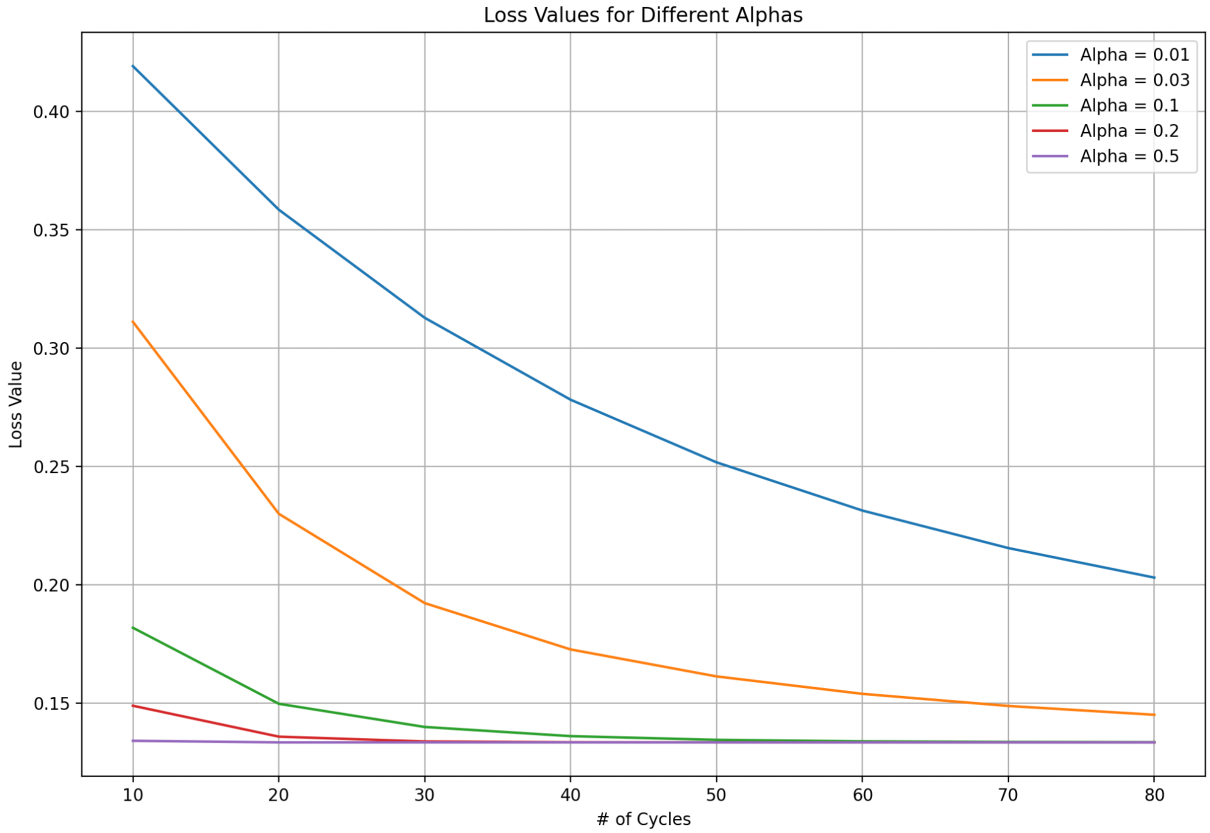
Testing alpha = 0.1!
Cycle #10 Loss: 0.1819471912821587
Cycle #20 Loss: 0.14982962332600686
Cycle #30 Loss: 0.14004150868932377
Cycle #40 Loss: 0.13617202131412695
Cycle #50 Loss: 0.13460253082449805
Cycle #60 Loss: 0.13396455827082202
Cycle #70 Loss: 0.133705186306677

Cycle #80 Loss: 0.13359973535610564
Total # of Cycles (Alpha = 0.1): 80
Gradient Time Elapsed (alpha = 0.1): 0.0011498928070068836

Testing alpha = 0.2!
Cycle #10 Loss: 0.14901233231420197
Cycle #20 Loss: 0.13595669278251765
Cycle #30 Loss: 0.13391233307823933
Cycle #40 Loss: 0.13358846107189612
Cycle #50 Loss: 0.13353715040681297
Cycle #60 Loss: 0.13352902131664288
Cycle #70 Loss: 0.1335277334341545
Cycle #80 Loss: 0.13352752939640006
Total # of Cycles (Alpha = 0.2): 80
Gradient Time Elapsed (alpha = 0.2): 0.0012159347534179688

Testing alpha = 0.5!
Cycle #10 Loss: 0.1341996763788764
Cycle #20 Loss: 0.13353215971298305
Cycle #30 Loss: 0.13352752341263244
Cycle #40 Loss: 0.1335274912107683
Cycle #50 Loss: 0.13352749098710717
Cycle #60 Loss: 0.1335274909855538
Cycle #70 Loss: 0.13352749098554298
Cycle #80 Loss: 0.1335274909855429
Total # of Cycles (Alpha = 0.5): 80
Gradient Time Elapsed (alpha = 0.5): 0.0011789798736572266

b + c)



The loss function for alpha = 0.5 gives the best result. It converges much faster than the other alpha. Thus, 0.5 is the best alpha out of these options.

Part C

Predicted price of a house w/ 2650 square feet and 4 bedrooms: 423554.11927749857
w: [-9.094380095333727e-17, 0.8847659867635496, -0.05317881857187676]

Part D

Stochastic Gradient Descent
Testing stochastic gradient descent for alpha = 0.05!
Stochastic loss after cycle 1: 0.1613492997302047
Stochastic loss after cycle 2: 0.14108004783908412
Stochastic loss after cycle 3: **0.13421848094422878**
Stochastic Time Elapsed: **0.00017786026000976562**

Regular Gradient Descent
Testing alpha = 0.5!
Cycle #10 Loss: 0.1341996763788764

Cycle #20 Loss: 0.13353215971298305
Cycle #30 Loss: 0.13352752341263244
Cycle #40 Loss: 0.1335274912107683
Cycle #50 Loss: 0.13352749098710717
Cycle #60 Loss: 0.1335274909855538
Cycle #70 Loss: 0.13352749098554298
Cycle #80 Loss: **0.1335274909855429**
Gradient Time Elapsed (alpha = 0.5): **0.0015249252319335938**

The stochastic gradient descent approach was 8.5x faster than the traditional gradient descent approach and produced a similar J(w) after just 3 cycles compared to the J(w) after 80 traditional cycles. The stochastic method converges on the solution after doing fewer computations. However, one thing I noticed was that between trials, because of the randomness, the stochastic gradient descent would sometimes increase its loss between cycles.