Q1

February 11, 2018

ML Assignment 1
Madhur Singhal
2015CS10235

0.1 Q1 Linear Regression

In this problem, we will implement least squares linear regression to predict density of wine based on its acidity. We will also use 3D plotting and contour plotting to observe how our gradient descent converges. The loss function and update equations are shown below.

$$J(\theta) = \frac{1}{2}[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2]$$

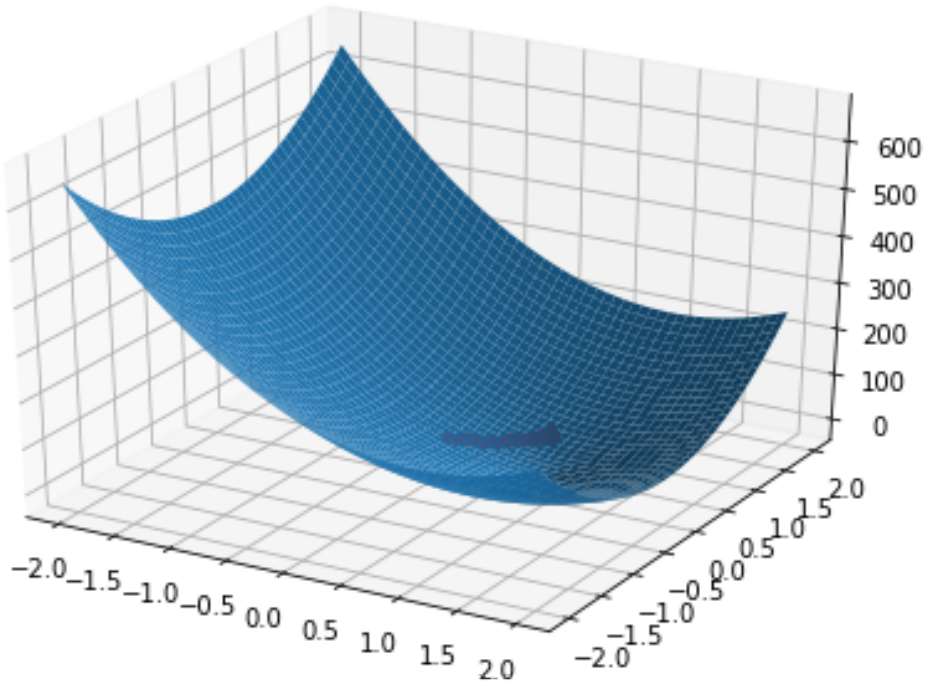
$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{1}{m}[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}]$$

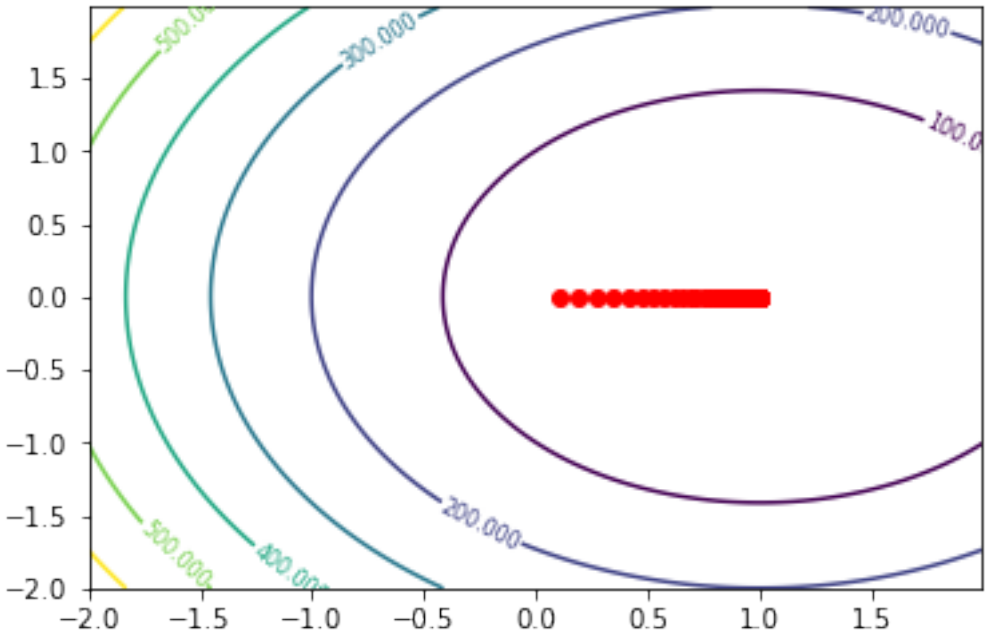
We use gradient descent to optimize the cost function and find out the values of parameters.

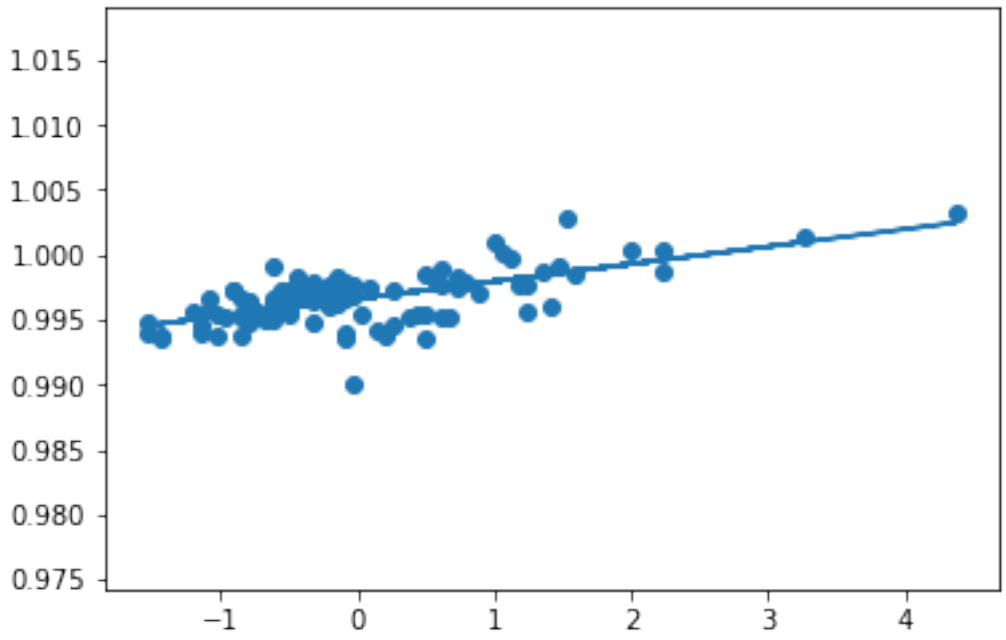
Since this is just a report so proper animation is not possible here, thus to view the plots interactively I have provided a separate python file in the folder (plots.py). In this report I have shown the final images produced only.

Below are the three plots produced for learning rate of 0.0001 and the parameters learnt.

```
Out[6]: [<matplotlib.lines.Line2D at 0x7effa0253c88>]
```

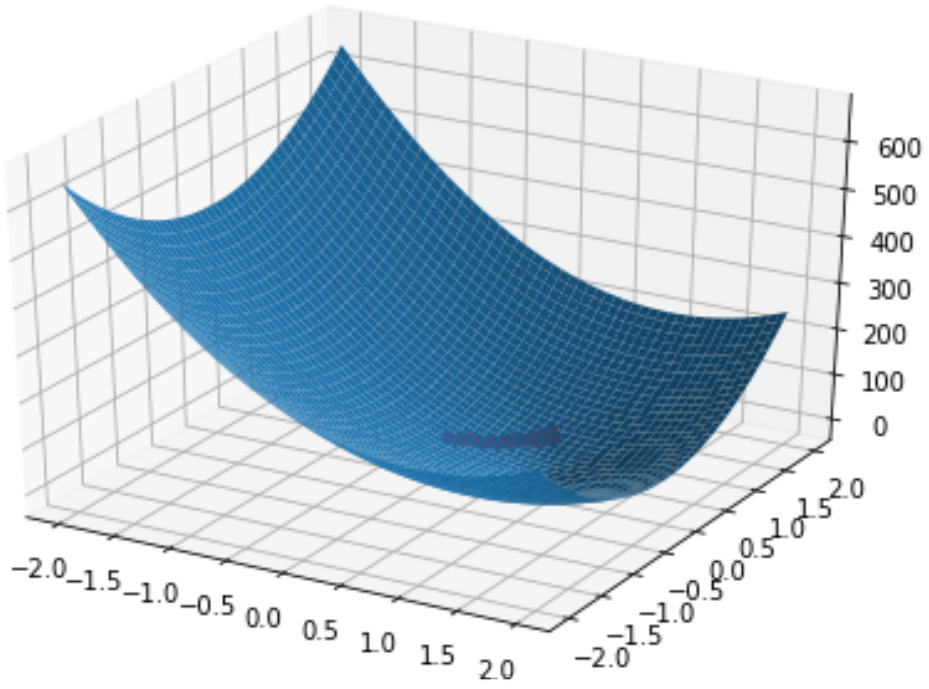

600
500
400
300
200
100
0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

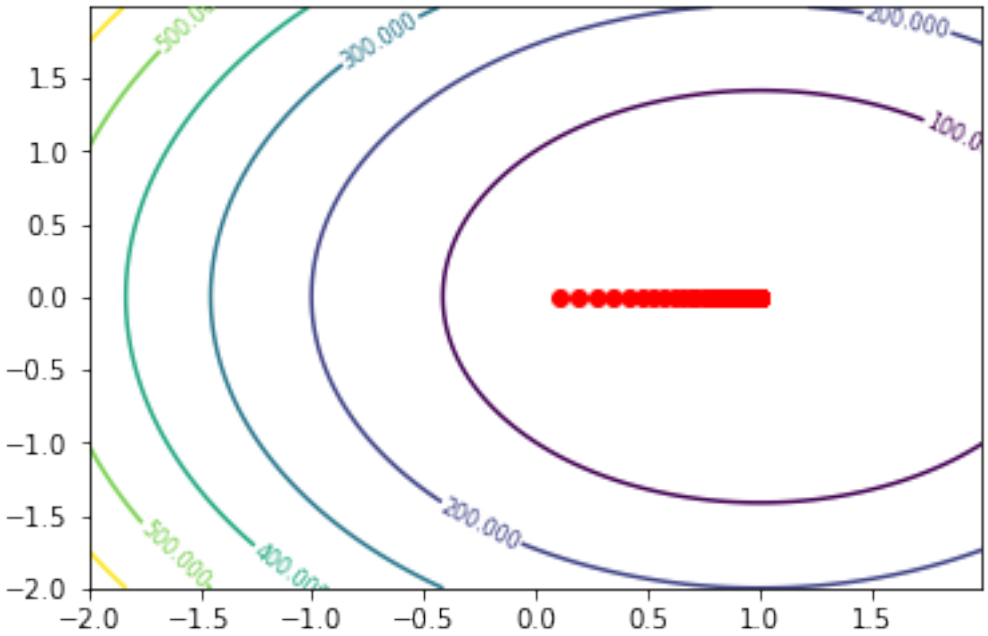

500.000
300.000
200.000
100.0
400.000
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0

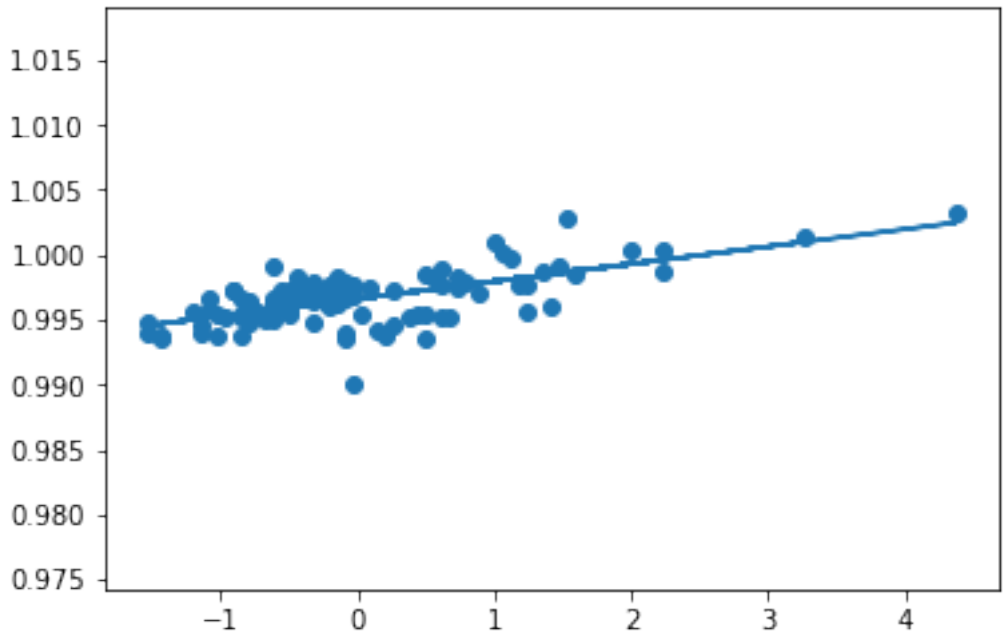


Finally we do the last part and see the contours for different learning rates.
For LR = 0.001, following are the stats and the plots obtained.

```
---------------
LR -  0.001
Iterations needed -  122
Loss -  0.00011947940058425674
Change in Loss -  9.839792527869054e-11
Theta --  [0.00134019 0.9966172 ]
----------------
```

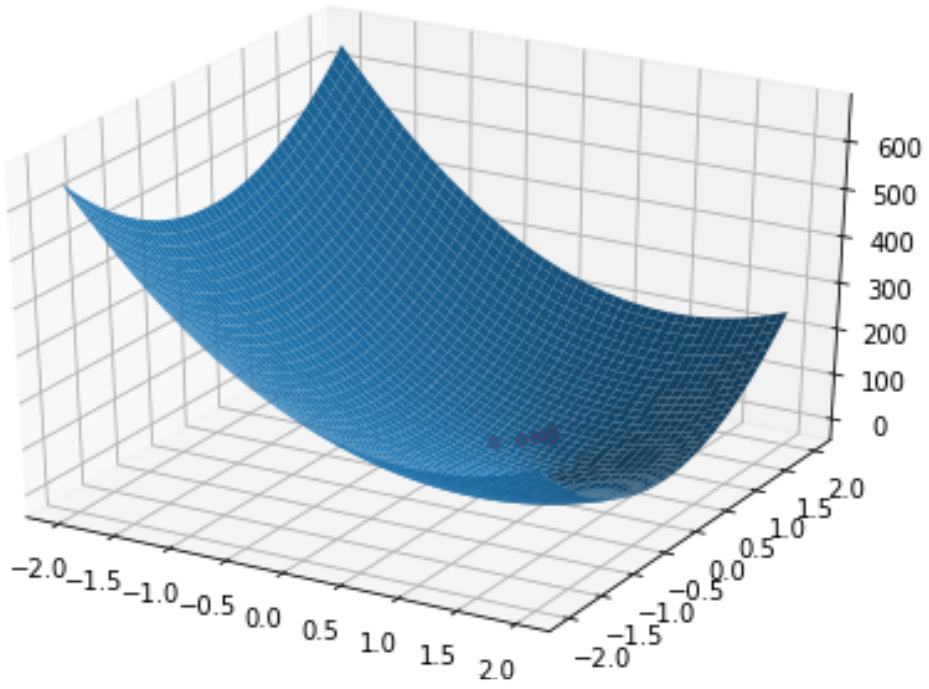

600
500
400
300
200
100
0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

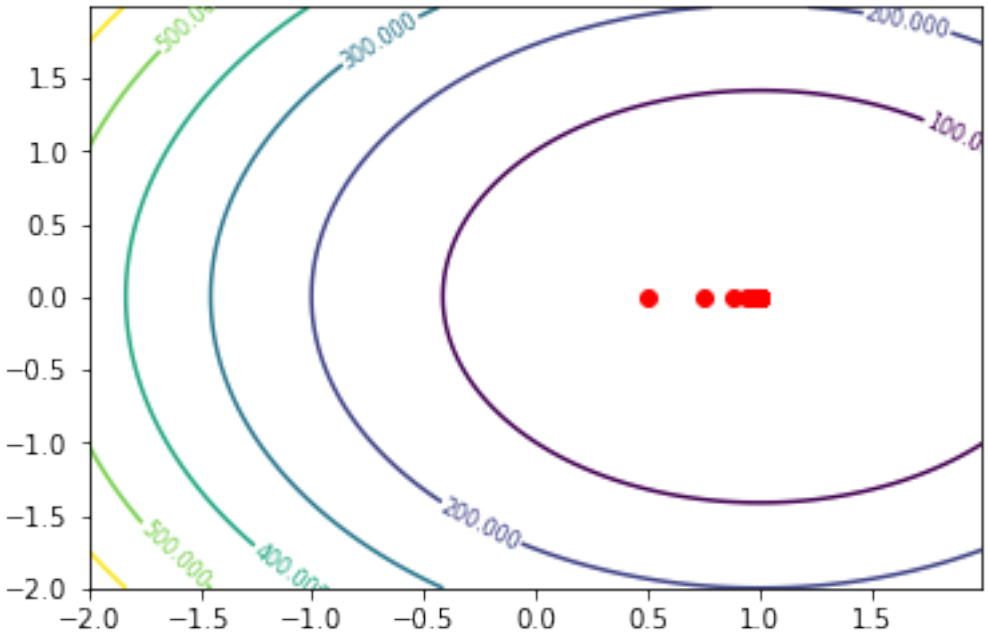

500.000
300.000
200.000
100.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

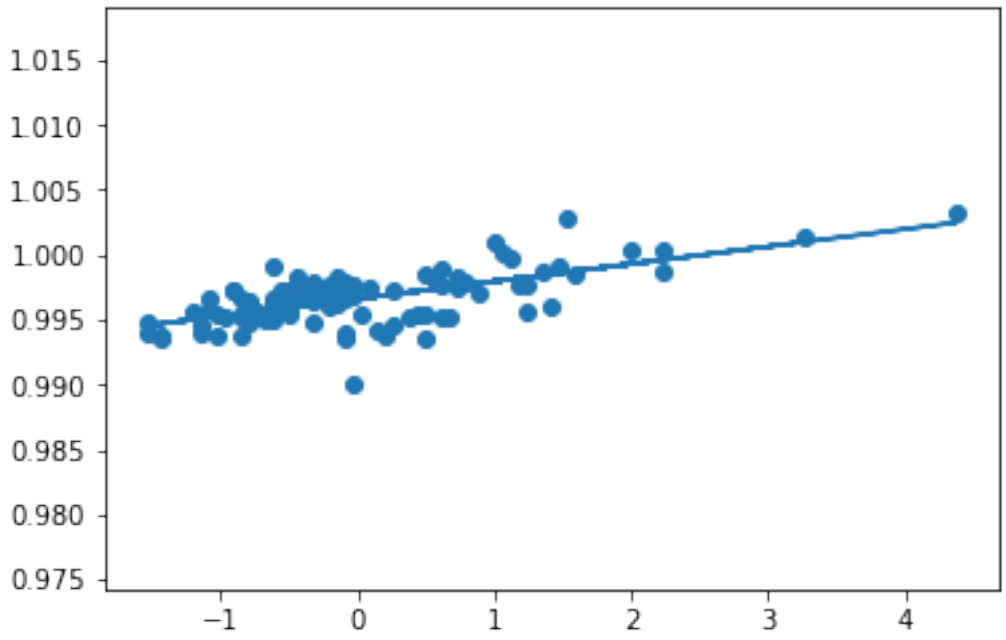


For LR = 0.005, following are the stats and the plots obtained.

```
---------------
LR -  0.005
Iterations needed -  22
Loss -  0.00011947899239035003
Change in Loss -  3.3875952444670926e-11
Theta --  [0.0013402  0.99661962]
----------------
```

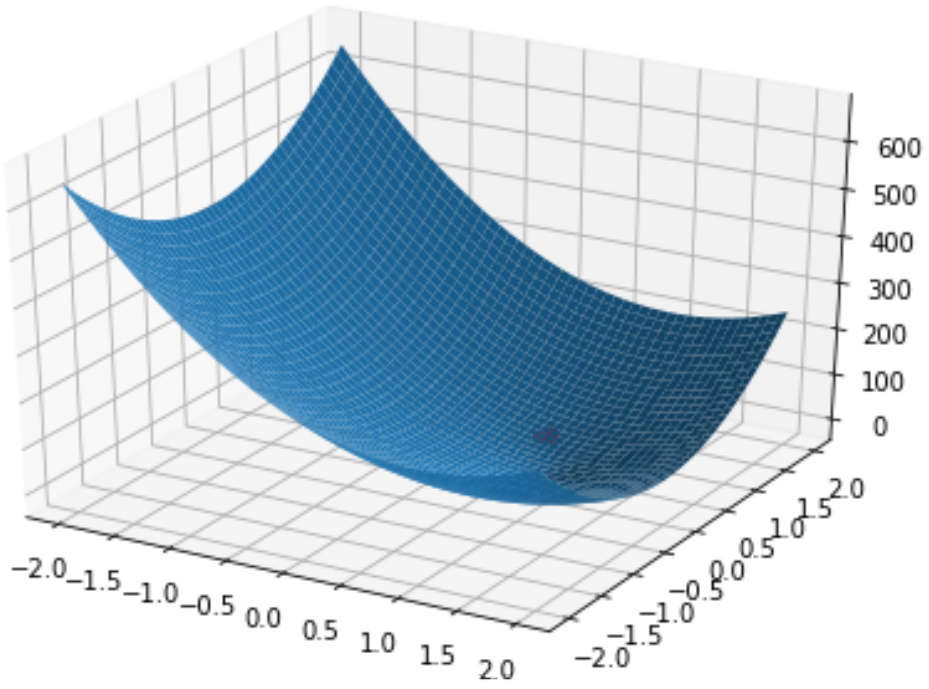

600
500
400
300
200
100
0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

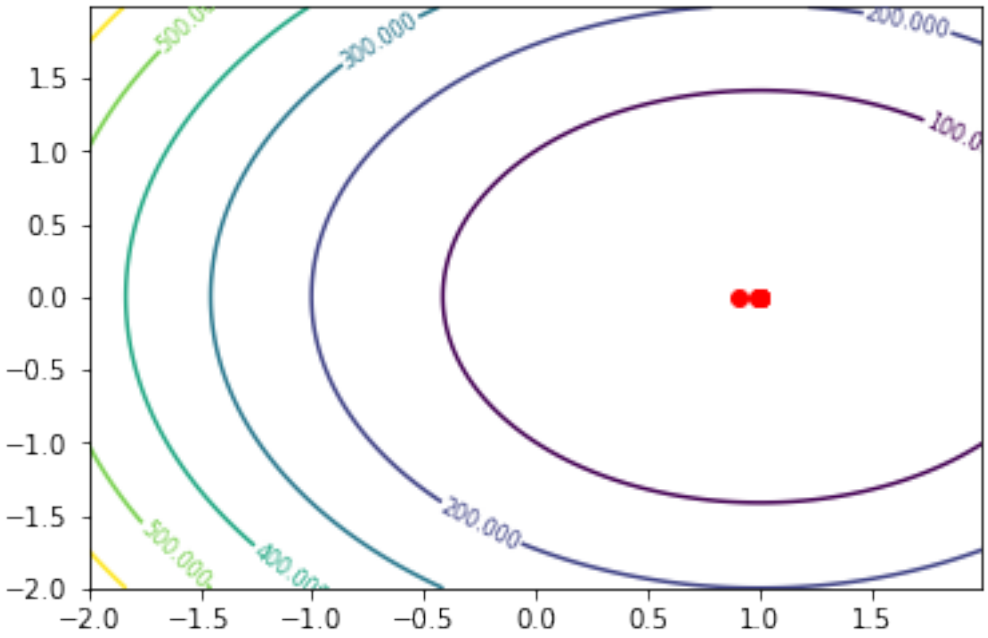

300.000
200.000
100.0
500.000
400.000
200.000
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

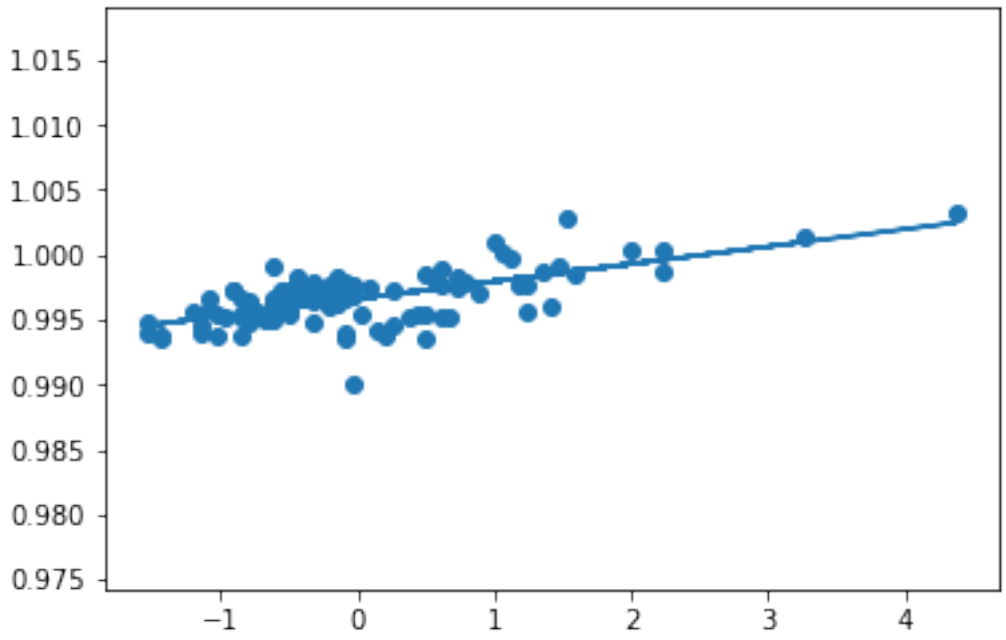


For LR = 0.009, following are the stats and the plots obtained.

```
---------------
LR -  0.009
Iterations needed -  8
Loss -  0.000119478981594993
Change in Loss -  4.916604441824492e-11
Theta --  [0.0013402 0.99662  ]
----------------
```

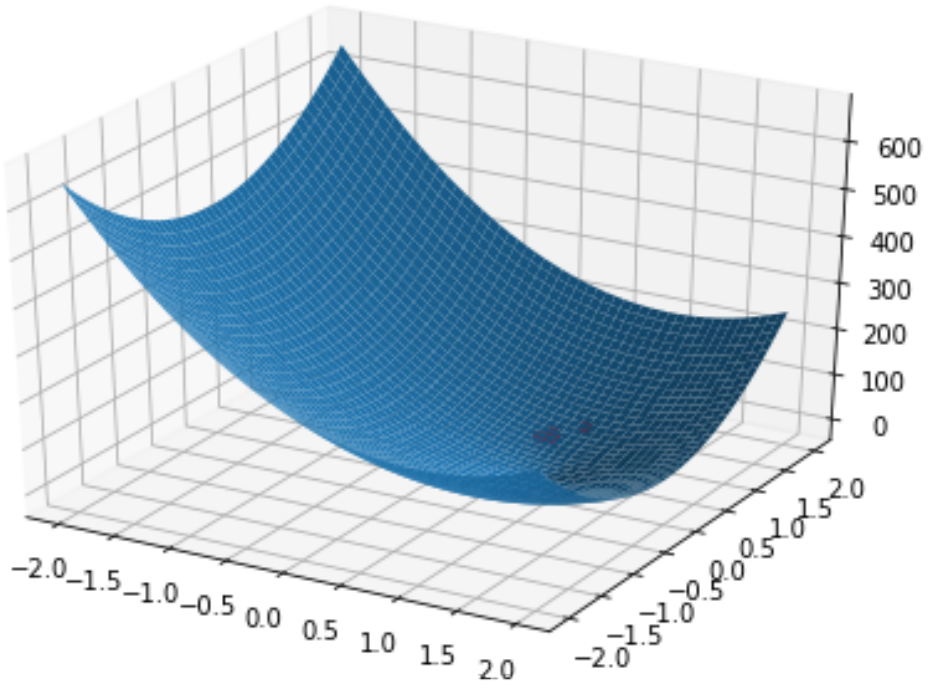

600
500
400
300
200
100
0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

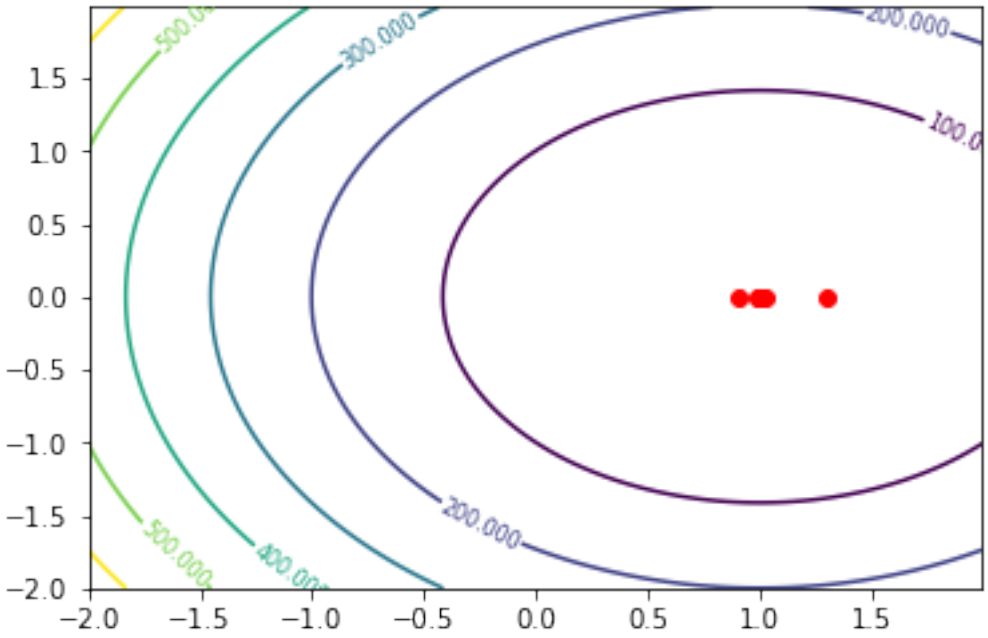

300.000
200.000
100.0
500.000
400.000
200.000
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

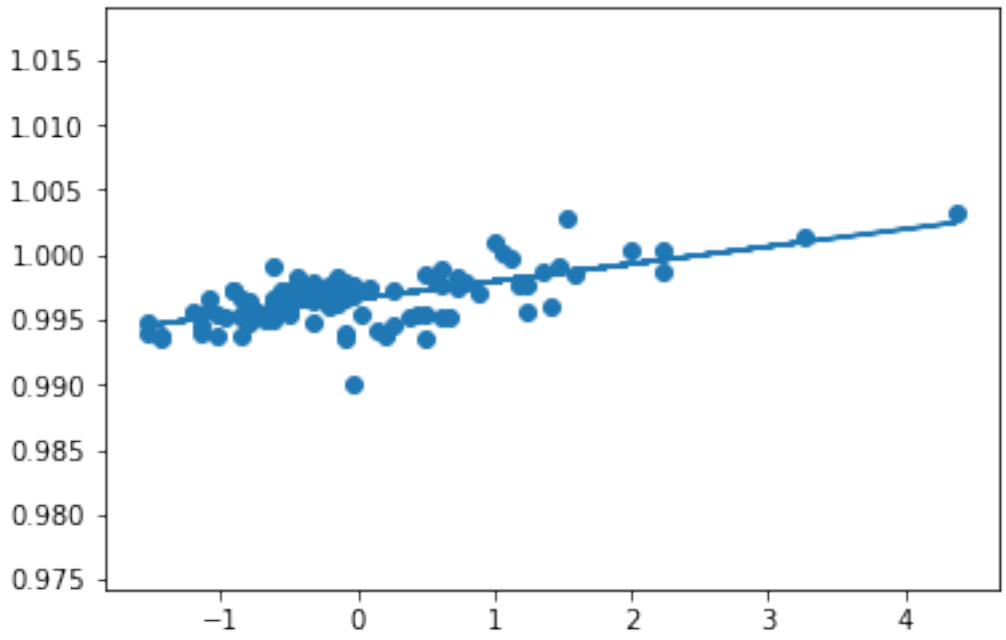


For LR = 0.013, following are the stats and the plots obtained.

```
---------------
LR -  0.013
Iterations needed -  14
Loss -  0.00011947898236072468
Change in Loss -  1.2763846511838761e-11
Theta --  [0.0013402  0.99662026]
----------------
```

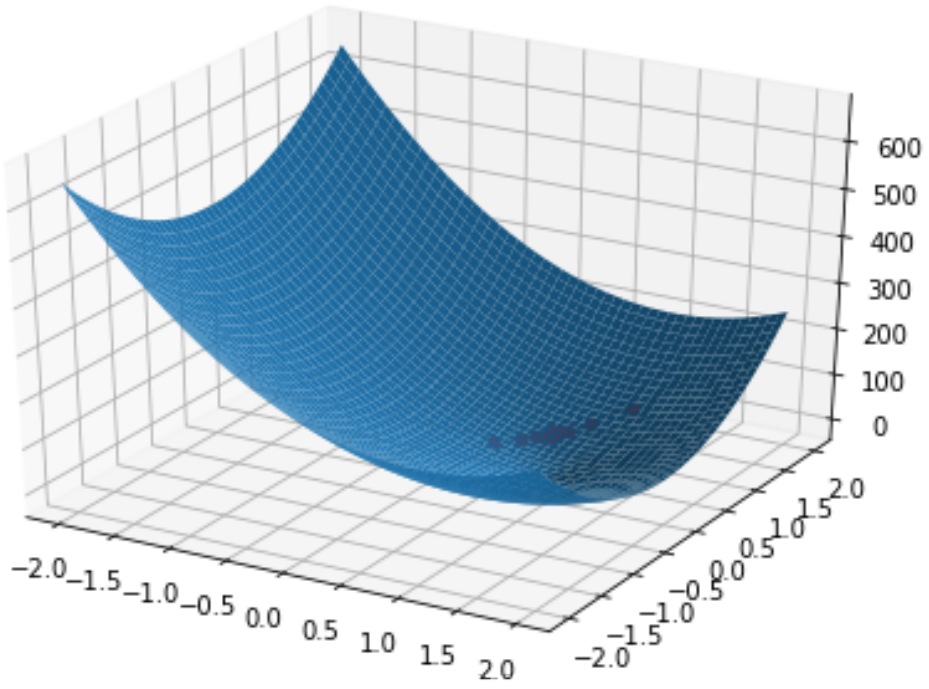

600
500
400
300
200
100
0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

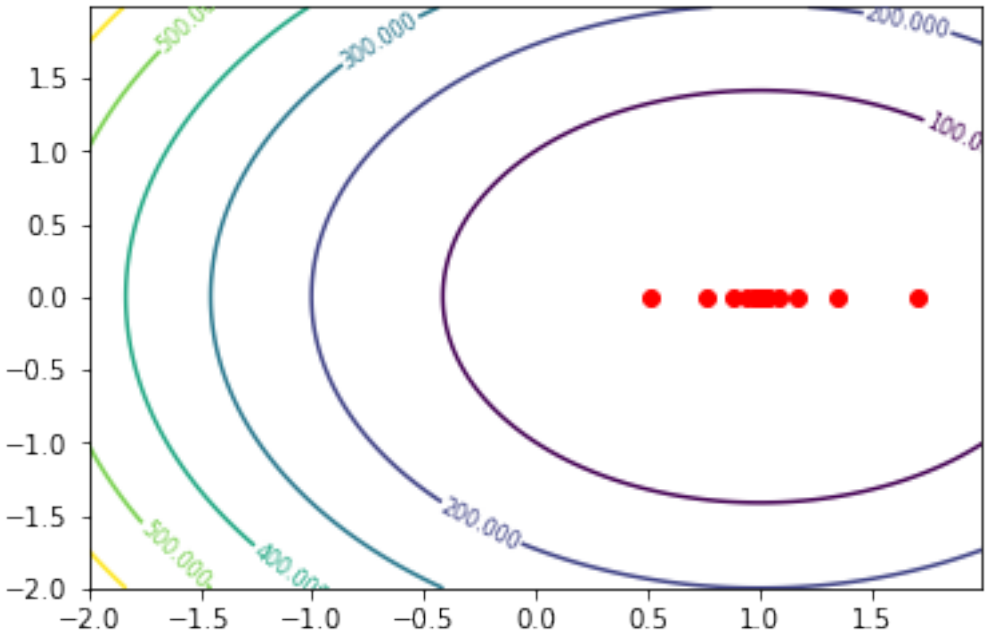

500.000
400.000
300.000
200.000
100.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0

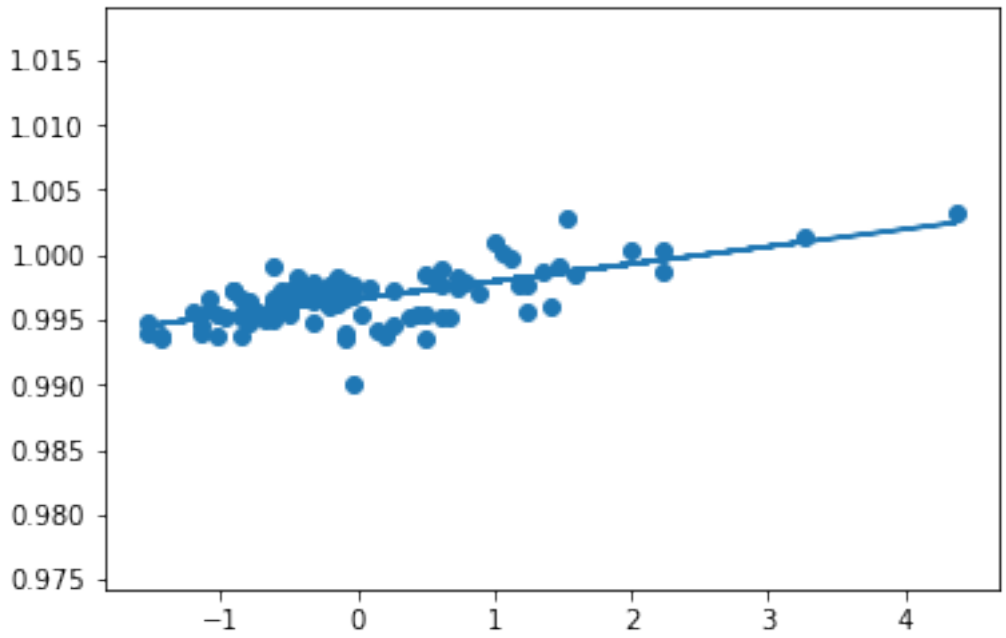


For LR = 0.017, following are the stats and the plots obtained.

```
---------------
LR -  0.017
Iterations needed -  39
Loss -  0.00011947906494412838
Change in Loss -  8.72680353545055e-11
Theta --  [0.00134019 0.99661881]
----------------
```


600
500
400
300
200
100
0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


300.000
200.000
100.0
500.000
400.000
200.000
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5



For the other two values of LR given, gradient descent diverges and python gives an overflow error after some iterations.

The main observation is that a moderate value of learning rate (0.009 here) is the best as there are problems on both sides of the spectrum. For large learning rates gradient descent can oscillate about the minimum for quite some time as seen here with LR = 0.017. Also gradient descent can diverge too if the learning rate is too high (more than 0.2 here). For small learning rates, gradient descent takes a large time to converge as can be seen by comparing the different number of iterations taken using different learning rates. With LR = 0.009 our gradient descent converges in just 8 iterations with a final loss of 10^{-4}.