CS 146 - HW2 - Q4

Jie-Yun Cheng
004460366

////////////////////////////
//   Problem #4 //
////////////////////////////

(a)

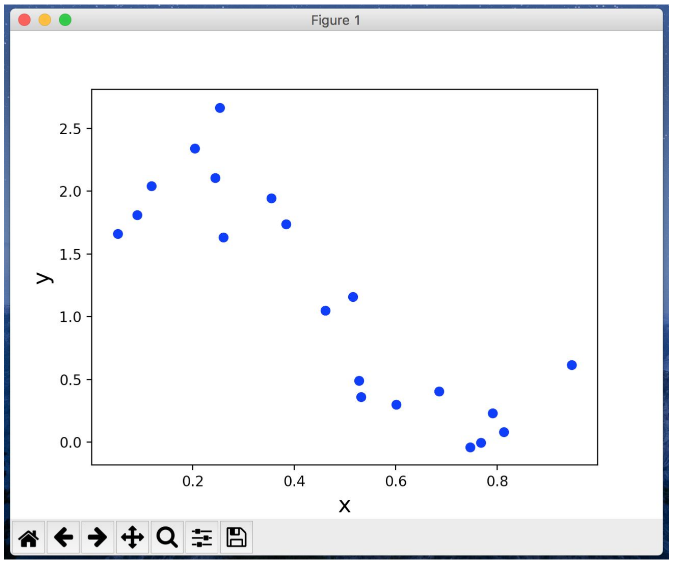

Figure 1: plot of train_data

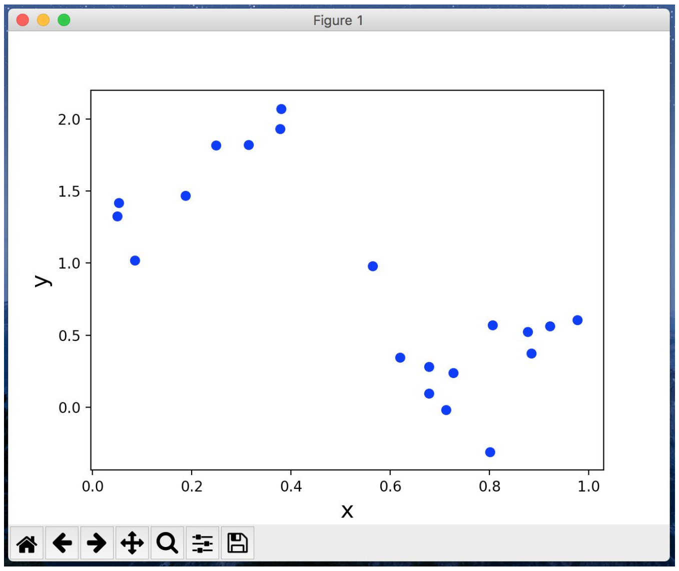

Figure 2: plot of test_data

In general, the data points in both figures seem to show a negative correlation between x and y, but the test_data seems to have two different clusters of points. Linear regression probably would not do well in predicting the data, although polynomial regression may work.

(b)
Done. Check code on my github:
https://github.com/chengjieyun59/UCLA_CS146_projects/tree/master/hw2/code/src

(c)
Done. Check code on my github:
https://github.com/chengjieyun59/UCLA_CS146_projects/tree/master/hw2/code/src

(d)
If the step size is too large, the algorithm would not converge, like the 10000 iterations for when step size is 0.0407. This would result in a large error. If the step size is too small, then it would take a very long time to converge, as evident in the 7021 iterations for when step size is 0.001.

Printed from the terminal:

```
code snippet cost: 40.233847409671
Investigating gradient descent...
number of iterations: 10000
cost, J(theta): 4.0863970367957645
step size, eta: 0.0001
vector, theta: [ 2.27044798 -2.46064834]
number of iterations: 7021
cost, J(theta): 3.9125764057919437
step size, eta: 0.001
vector, theta: [ 2.4464068 -2.816353 ]
number of iterations: 765
cost, J(theta): 3.912576405791486
step size, eta: 0.01
vector, theta: [ 2.44640703 -2.81635347]
number of iterations: 10000
cost, J(theta): 2.710916520014198e+39
step size, eta: 0.0407
vector, theta: [-9.40470931e+18 -4.65229095e+18]
```

Converted into a table:

step size, η	vector, θ	iterations	cost, $J(\theta)$
0.0001	[2.27044798; -2.46064834]	10000	4.0863970367957645
0.001	[2.4464068; -2.816353]	7021	3.9125764057919437
0.01	[2.44640703; -2.81635347]	765	3.912576405791486
0.0407	[-9.40470931e+18; -4.65229095e+18]	10000	2.710916520014198e+39

(e)
Printed from the terminal:

```
Investigating closed form solution...
vector, theta: [ 2.44640709 -2.81635359]
```

The coefficient vector, θ, is [2.44640709; -2.81635359] with a cost of 3.912576405791486, the same as part (d). When the model parameter, step size, for the gradient descent is small, the coefficients are closer to that of the closed form solution. Though closed form solution is faster amd takes less run time than gradient descent in this case.

(f)
Printed from the terminal:

```
Investigating different step sizes...
number of iterations: 1679
cost, J(theta): 3.912576405792008
vector, theta: [ 2.44640678 -2.81635296]
```

It takes 1678 iterations with the cost of 3.912576405792008 to converge to [2.44640678; -2.81635296]. In this case, $\eta_k = \frac{1}{1+k}$, while k = number of iterations of the outer loop. This gradient descent has the same solution as the closed form solution.

(g)
Done. Check code on my github:
https://github.com/chengjieyun59/UCLA_CS146_projects/tree/master/hw2/code/src

(h)
Using $J(\theta)$ is not that good at comparing different data sets when they are of different sizes. On the other hand, RMSE can compare different sizes of data sets by normalizing all the points with the normalization term, N, and the square root makes it so that RMSE would have the same standard deviation as the data predictions.

(i)

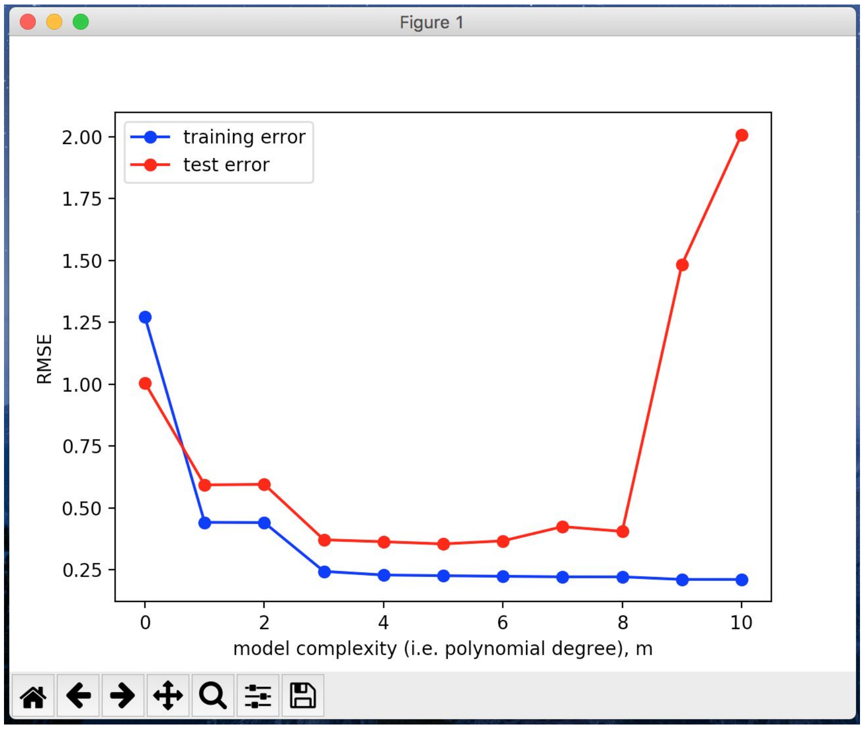

Figure 3: training error and test error calculated with RMS method for each polynomial degree

When m is too large, which in this case is when m > 8, there is overfitting, as evident in the figure that shows that the test error increases dramatically while training error keeps decreasing. When m is too small, which is when m < 3 in this case, there is underfitting, where training error and test error are both high. The best way to fit the data is with a 4th, 5th, or 6th degree polynomial, where error is the lowest and there is no overfitting or underfitting.