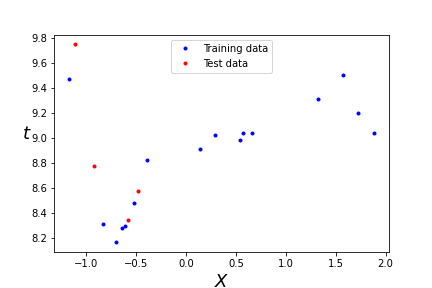
ELL409   
Assignment 1

Vansh Gupta  
2019EE10143

**Part 1A**

* Using only 20 data points

  
Fig 1. Randomly sampled 20 data points

* Using Moore-Penrose pseudoinverse:

I run my code for different values of (regularization constant) and epochs (number of iterations). [Fig 2.]

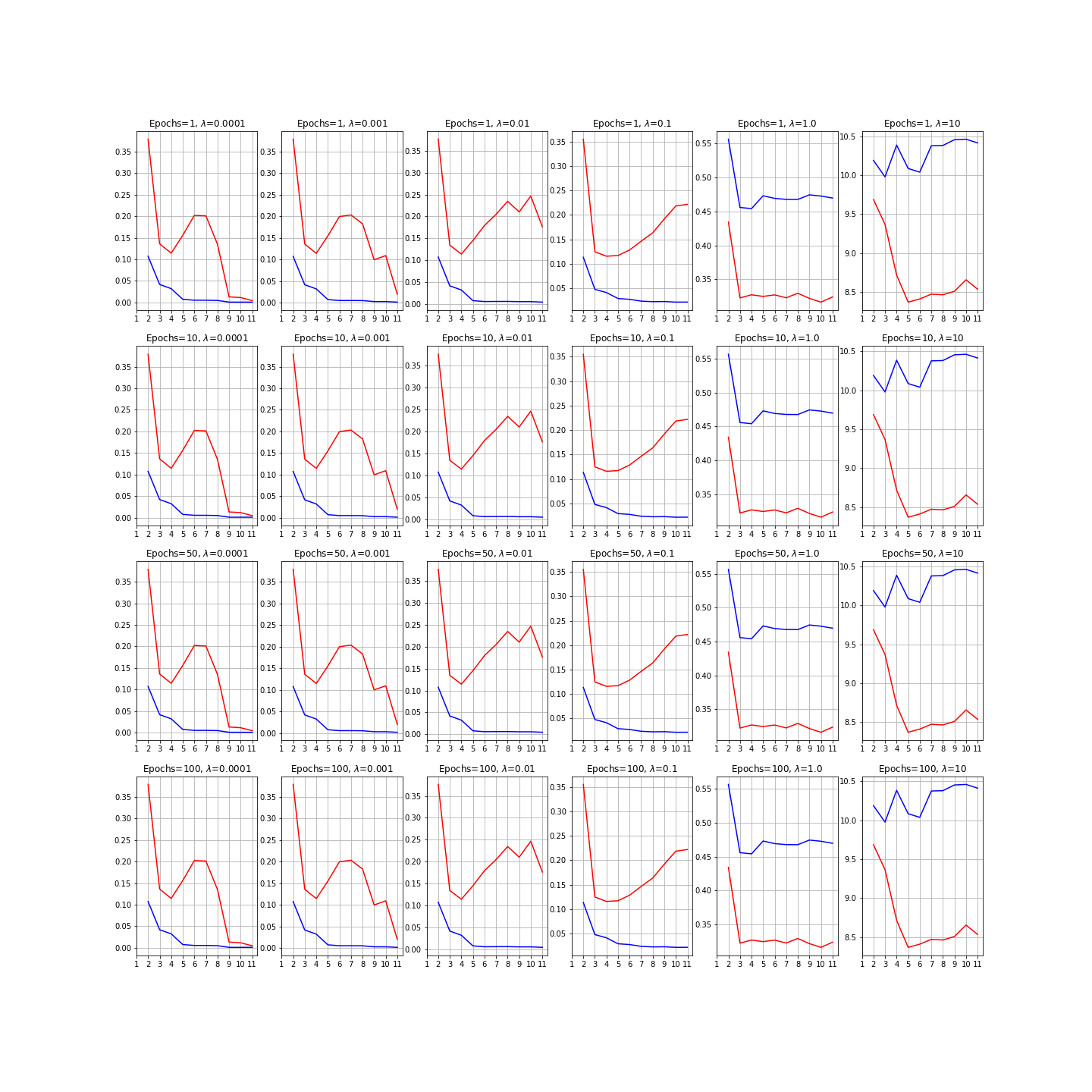
After that, I observed that the number of epochs have no effect on the losses, which is also expected since our dataset is very small for the model to have to learn the same data point again. Thus, I iterate over different . We can also see that while the training error has an elbow point at m=5 (m is the highest degree of polynomial), the test set is truly reduced for m=9. Thus, for further iterations, I will fix m at 9. [Fig 3]

Once again, we see that since number of data points is only 20, there is no over-fitting in the first place that should be fixed with regularization

A final plot for = 0, number of epochs = 1, over different values of m is shown in Fig. 4

Therefore, the best guess of the polynomial, for m = 9 and = 0 is:

8.99654324 0.32089103 0.30898259**2** 4.58312801**3** 3.91859305 6.12614492**5** 7.53175079**6** 0.885168573.20999694**8** 0.88288049

  
Fig 2. The red line depicts the test error and the blue line shows the training error

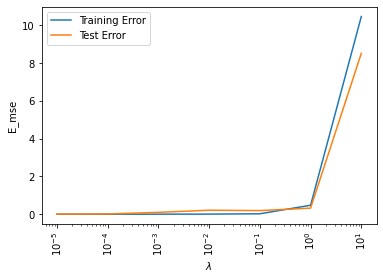
  
Fig 3.

  
Fig 4.

* Using Gradient Descent:

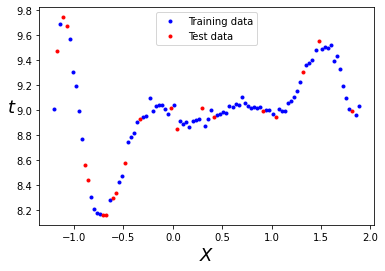
I followed the exact same procedure as listed for the Moore-Penrose pseudoinverse.

The inference for epoch and was absolute and not related to the method, therefore for this set of experiments, I keep my number of epochs to be 1 and to be 0.

Thus, here my main focus is on the maximum degree of the polynomial (which should be the same), the learning rate and the size of our mini-batch.

For the order of polynomial, I simply take learning rate value of 0.02 and first do stochastic gradient descent on it for 1 epoch, and then batch gradient descent for 100 epochs. The results are in Fig. 5 and Fig. 6 respectively.

* Using all 100 data points:

  
Fig. x

* Using Moore-Penrose pseudoinverse:

I run my code for different values of (regularization constant) and epochs (number of iterations). [Fig xx.]

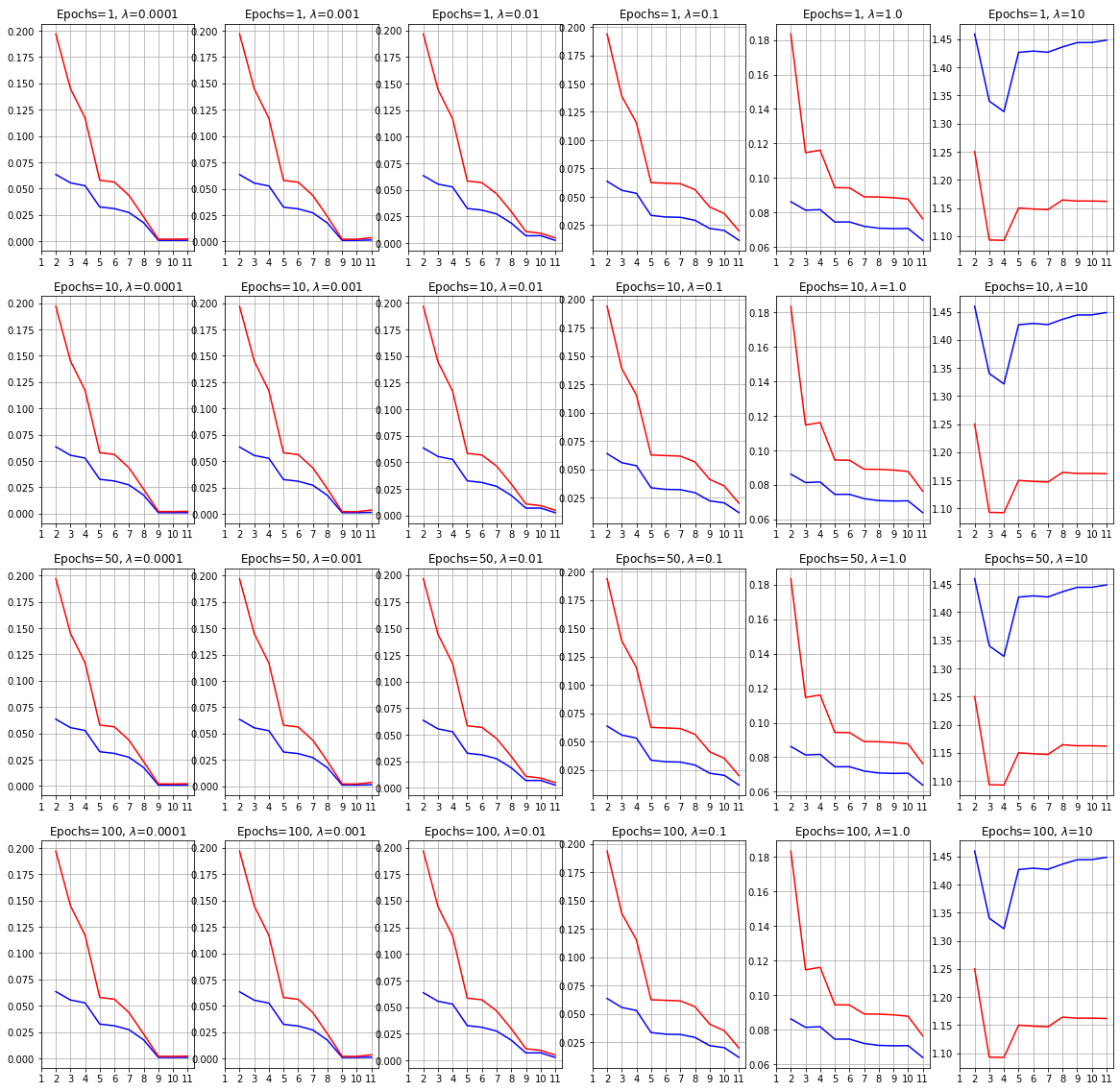
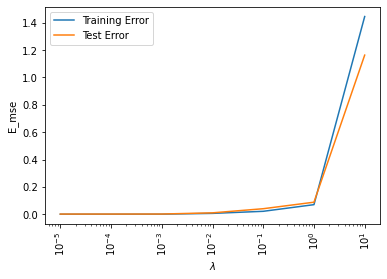
Once again, we observed that the number of epochs have no effect on the losses. Thus, I iterate over different . We can also see that while the training error has an elbow point at m=5 (m is the highest degree of polynomial), the test set is truly reduced for m=9. Thus, for further iterations, I will fix m at 9. [Fig xxx]

This time, we see that since number of data points is 100, there is some improvement with regularization at = 0.01 after which it starts under-fitting.

A final plot for = 0.01, number of epochs = 1, over different values of m is shown in Fig. xxxx

Therefore, the best guess of the polynomial, for m = 9 and = 0.01 is:

9.0140234 0.01824327 0.97251592**2** 2.73209287**3** 1.05359651 4.21682231**5** 4.04789536**6** 0.86240564 1.9421145**8** 0. 51181432

  
Fig xx. The red line depicts the test error and the blue line shows the training error  
  
Fig. xxx

