Sanket Wagh – A20330391

CS584: HW1 : Parametric Regression

Problem Statement:

To implement parametric regression. This implementation has to be performed on single variable as well as multivariate datasets. First, we implement the linear regression model by training and testing the data. Parametric regression comes into play when we change parameters such as the *size of training and testing data and by changing from linear to polynomial, by trying it out with higher dimensions of data.* For the multivariate data set there is a comparison to be done among the explicit method, iterative method and the kernel method. A comparison must be done for my logic vs the in-built python functions. The effect of the above experiments must be observed and documented.

Proposed Solution:

The main objective is to evaluate for the unknown values. The co-efficient of the feature (theta) must be determined. This can be done using the formulae given. We must use the data set and obtain a set of features and labels in the form of vectors to implement the linear regression. *Once we obtain theta we perform cross validation to analyze the performance of the model. Mean Square Error has been used as error measurement mechanism. Once this is completed, we move on and perform similar experimentation with the different forms of data as mentioned in the previous section.* The test error is of great importance since that is going to be the basis of all comparisons.

Implementation Details:

The data is first put into lists, one acts for the features (X) and the other the labels (Y). A data matrix is obtained by placing 1's in the first column of X. This is going to be used for determining the coefficient's value. All the code has been done dynamically and very few packages have been used like the PolynomialFeatures package to gain features of higher dimension.

Results and Implementation:

- Analyzing the dataset and fitting the model. I have loaded all data sets once but chose to work with svar-set2. I can see there is a sort of elbow curve here. Where the fit is high at first. Then it drops and slowly gets back up but not too much. Overall it seems to be a good fit but the data seems to form a bell curve rather than a straight line.

- The coefficient value(theta) :

Theta (using manual method.)	Theta (using in built regression function.)

`[[ 0.33244798]`
` [-0.07768946]]`
	`[[ 0.          -0.07768946]]`
	(co-efficient of actual feature and not one in 1st column of Z.)

Training error	Testing error
0.0600088113443	0.0598352197034

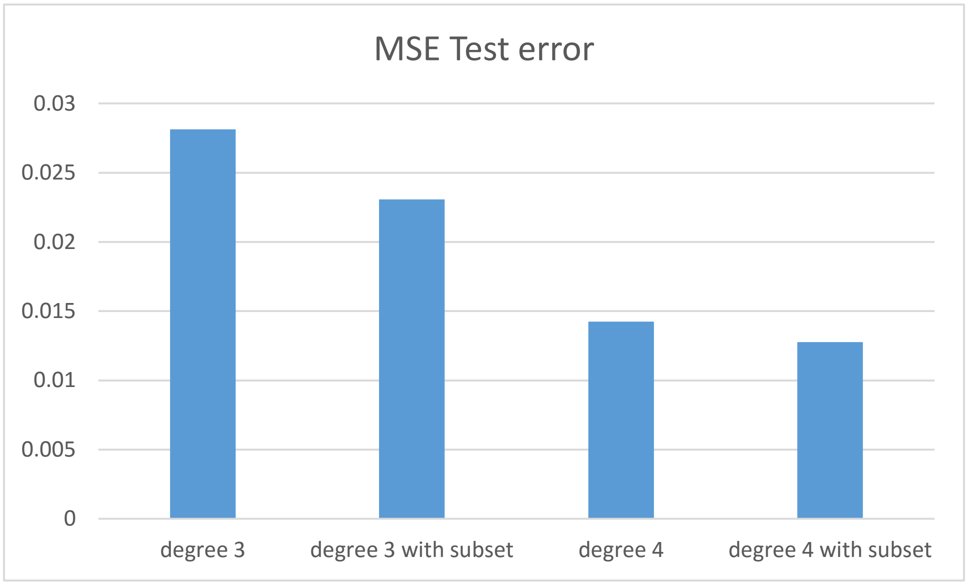
The errors are the average MSE errors which are used to determine the accuracy of the performance. We can see that the output is fairly accurate since the test error is low. The training error is greater that the test error.

- Increased Dimensions:
    - I have used 3-dimensional 4-dimensional features for this experimentation.
    - Each of the two have been experimented with using subsets of data also.

MSE Training error (degree = 3)	MSE Testing error (degree = 3)
4.80224549217	0.0281468051968
MSE Training error (degree = 4)	MSE Testing error (degree = 4)
2.88593330241	0.0142421449536

MSE Training error (degree = 3; subset of data)	MSE Testing error (degree = 3; subset of data)
3.00442324957	0.0230724717611
MSE Training error (degree = 4; subset of data)	MSE Testing error (degree = 4; subset of data)
1.79635171986	0.0127530960928

It is important to look at the test errors. This information has been represented in the form of graph as shown:

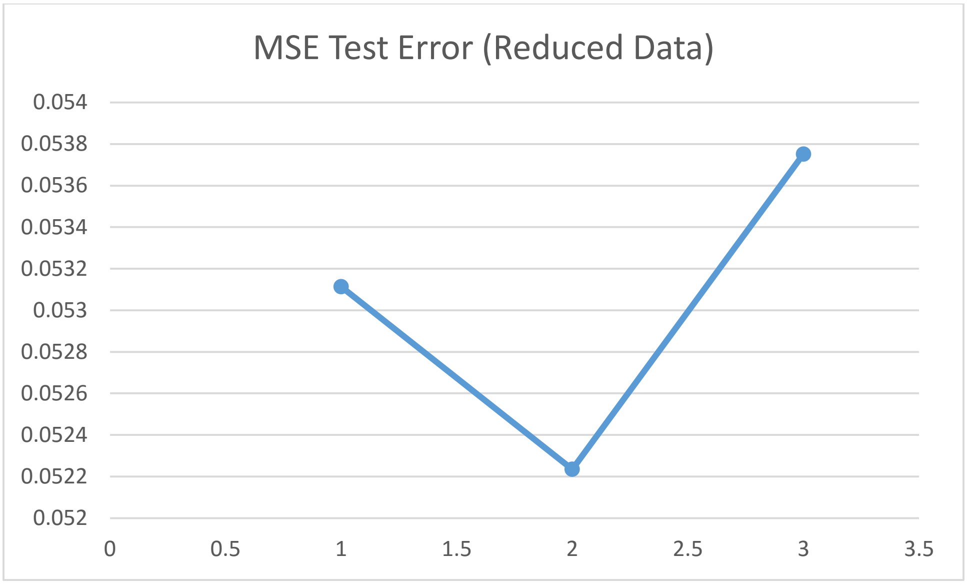


Looking at this I would choose polynomial with 4 dimensions. Since the error is lowest for it. When a subset of data is used the features with 4 dimensions still provides a better error rate. Hence I would choose it over 3 dimensions. Note: there is a possibility that data with higher dimensions may produce better or worse results but my decision to choose has been done considering the above two only.

- Reducing the data:
    - I created 3 chunks of reduced data. The feature vector would now have 150 features, 100 features and 60 features respectively for training. For the testing I chose the last 50 features. Same holds for the Labels(Y).

Reduced size (percentage)	MSE Test Error
150 vs 50 (75% of original)	0.0531134130605
100 vs 50 (50% of original)	0.0522352162825
60 vs 50 (30% of original)	0.0537518240839

Visualization:



When it comes to which of the above performed better it would be the 2nd one since it has least MSE Error.

- Multivariate Data: Dataset used: mvar-set2
    - Mapping the multivariate dataset with higher dimensions: I have mapped with 2-dimensions, 3-dimensions and 4 dimensions. Since 4-dimensions worked best with single variable polynomial function I used this for further experimentation with multivariate data set.
    - The theta values obtained are as follows:

Theta(Single dimension)	`[[ 0.06458313]`
	` [-0.00060792]]`

Theta (2 dimensions)	`[[  4.27124879e-04]`
	` [  6.45831279e-02]`
	` [ -6.07924252e-04]`
	` [  2.05812667e-05]`
	` [ -1.07308260e-03]`
	` [  3.14956092e-04]]`

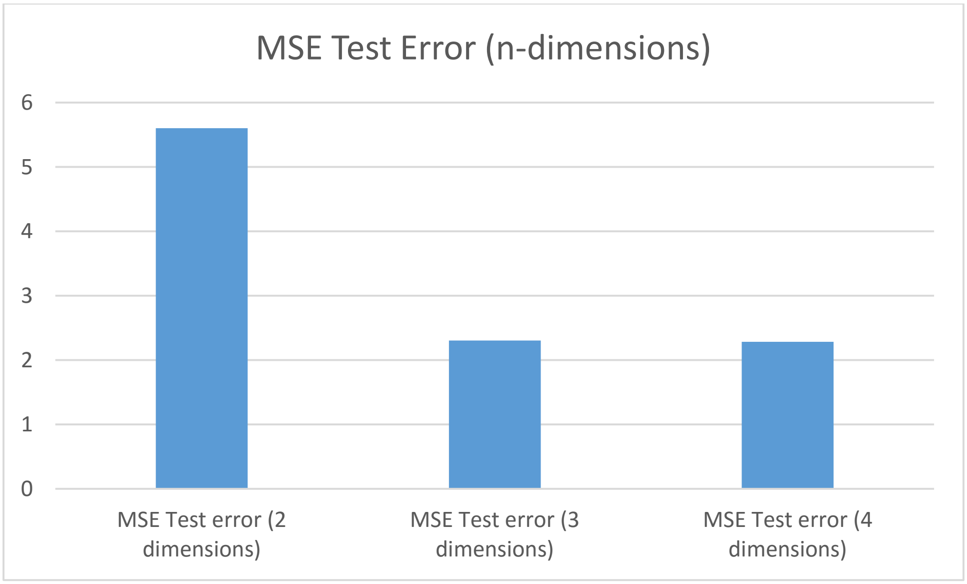
Theta (3 dimensions)	`[[  4.27124879e-04]`
	` [  2.74640229e-01]`
	` [ -4.46822095e-04]`
	` [  2.05812667e-05]`
	` [ -1.07308260e-03]`
	` [  3.14956092e-04]`
	` [ -6.29961462e-02]`
	` [ -2.02197567e-04]`
	` [ -3.80321131e-02]`
	` [  4.78640087e-05]]`

Theta (4 dimensions)	`[ -3.05395075e-03]`
	` [  2.74640229e-01]`
	` [ -4.46822095e-04]`
	` [  5.79957163e-03]`

	`[   3.89628164e-03]`
	`[   2.21490074e-03]`
	`[  -6.29961462e-02]`
	`[  -2.02197567e-04]`
	`[  -3.80321131e-02]`
	`[   4.78640087e-05]`
	`[  -1.47536279e-03]`
	`[  -8.76002123e-04]`
	`[  -3.75542840e-04]`
	`[  -1.11442954e-03]`
	`[  -3.86890751e-04]]`

Training errors are in the ipynb notebook but the test errors are worth noting, they are:

MSE Test error (2 dimensions)	5.60259447157
MSE Test error (3 dimensions)	2.30461586813
MSE Test error (4 dimensions)	2.28267885971



From this we can concur that 4 dimensions has the lowest error and hence is better than the other 2; just like in the case with single variable dataset.

Iterative approach:

Gradient descent was used in this method. Iteration count was set to 300. The theta for this was:

```
[-0.02423084  0.27476301  0.07613982  0.02659217  0.09401492  0.02451174
 -0.06303124 -0.00931468 -0.03804669 -0.0218353  -0.00615162 -0.01745814
 -0.0026346  -0.01769657 -0.00545315]
```

And the MSE test error for this was: ***[0.01283283].***

Gaussian Kernal Approach:

For this approach we use the value for alpha and the gram matrix. The formula has been implemented.

Alpha value obtained was:

```
[[-0.10825879]
 [-0.01290997]
 [-0.12720998]
 ...,
 [-0.05667539]
 [-0.04032406]
 [ 0.16316383]]
```

The MS test error for this method was: ***[3.15714045e-32].***

Among the methodologies implanted the iterative approach was better than the standard approach. But the Gaussian Kernal approach is the best of the lot. This is because the implementation for Gaussian approach was very quick as compared to the other two. This is also evident by looking at the MSE error for the three. The Gaussian model is most accuarate.

The other relatively less notable results are part of the ipynb notebooks.