CS 584 - Machine Learning - Assignment Report

Rakesh Adhikesavan

Master's in Data Science, Illinois Institute of Technology, Chicago

Abstract

Parametric Regression is the process of fitting models to data where the models have a numerical response [1]. In this assignment, various techniques for parametric regression was implemented. The performance of each algorithm was evaluated and tested on four univariate and four multivariate data sets, using a 10 fold cross validation method [2]. The algorithms were implemented in Python and all the results and conclusions are summarized below.

----------***----------

1. Introduction

The purpose of this assignment is to get an idea of how parametric regression works and to identify how some models are more suitable to certain datasets than others.

First, four individual single feature datasets were loaded into the Python working environment. A linear model was used to fit each of this dataset and the training and testing errors were computed. The results obtained were compared to the inbuilt functions provided by Python package sklearn [3]. Then, the amount of training data used was reduced and the effect of the performance of the linear and polynomial models were recorded.

Secondly, four individual multiple feature datasets were loaded into the Python working environment. The features were mapped to a higher dimension feature space using a combination of features. Linear regression was performed in the higher dimension space. Different mappings were evaluated in terms of the testing error that they produced. The best model was chosen, the choice and justification was recorded. Gradient descent algorithm was used to solve the regression problem and the solution was compared to the explicit solution. The Dual regression problem was solved using a Gaussian Kernel function, the time performance and accuracy was recorded.

2. Implementation Details

All the algorithms were implemented in Python. Some of the important dependencies are numpy, sklearn, matplotlib and scipy.

2.1 readData.py

This program reads all the four single feature datasets. The Program looks for the files in the path *Data/svr/*

After loading the datasets into the Python working environment, the program generates a scatterplot of all four data sets and saves the plot in path *Plots/scatterplots.png*

2.2 svr.py

This program works with the single feature datasets. First, a linear model is used to fit the data and a scatterplot of the original and predicted values is generated. Then, sklearn's inbuilt function is used to fit a linear model to the data and the predicted values are plotted. Next, polynomial models of degrees from 2 through 6 is generated and the best model is selected. The programs runs a for-loop and calls all the functions on all the datasets. One of the design issues is that the plots couldn't be generated in a

matrix form. Every plot is generated as an individual plot. It would be more desirable to generated plots related to one another as subplots in a matrix form.

2.3 mvr.py

This program loads all the datasets in the path *Data/mvr/*

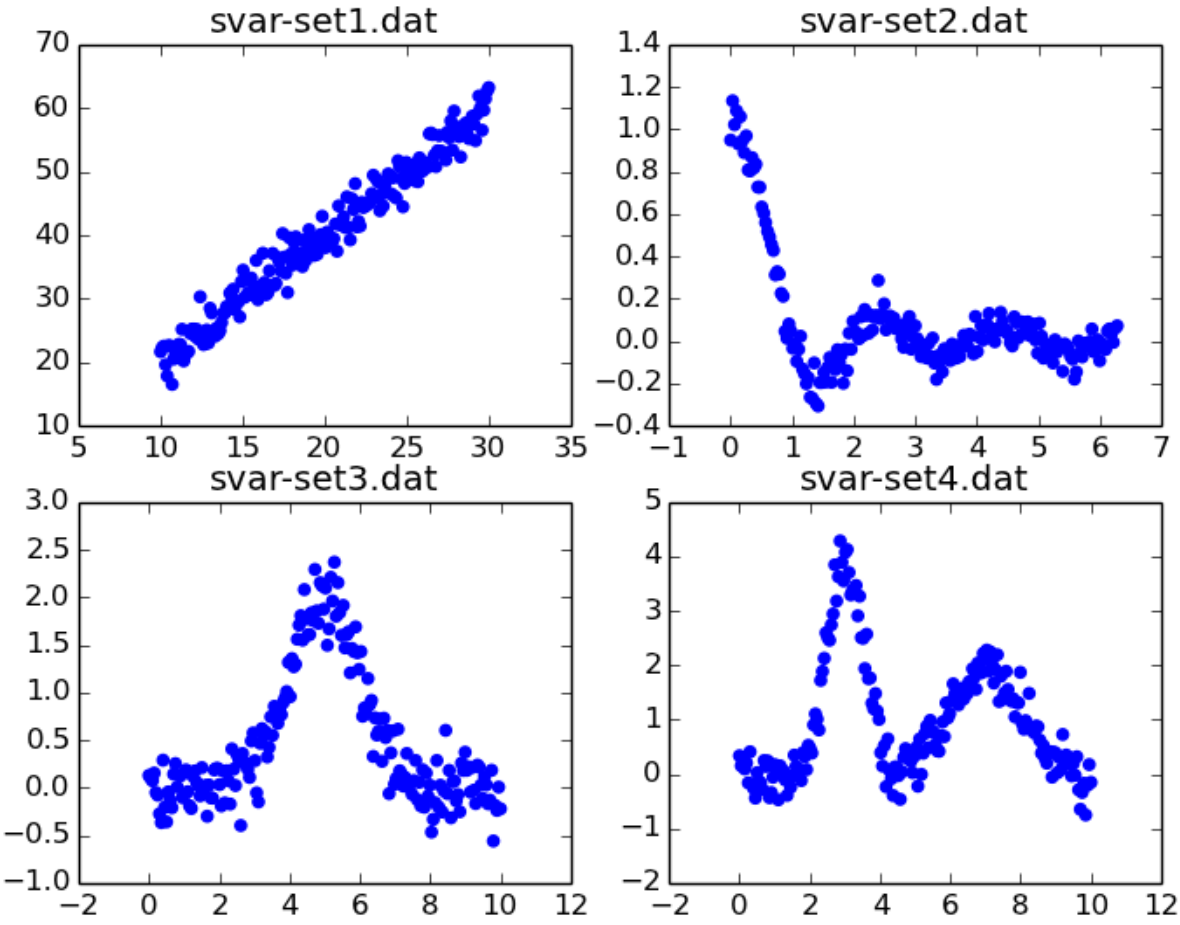
The Gradient Descent Algorithm uses a learning weight of *0.000001* and a threshold *of 0.000001.*

The initial theta values are generated using a random number generator in the range *[0,1].*

3. Results and Discussion

3.1 Single Feature Datasets

Four datasets from four individual files were loaded into the Python working environment. A

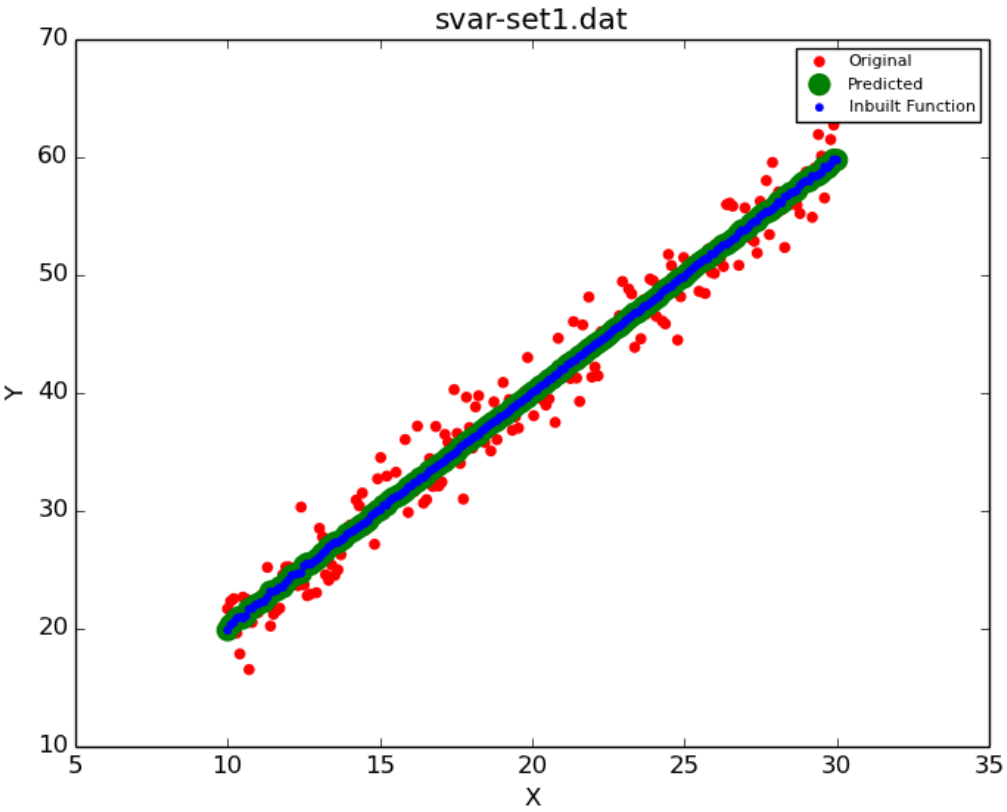


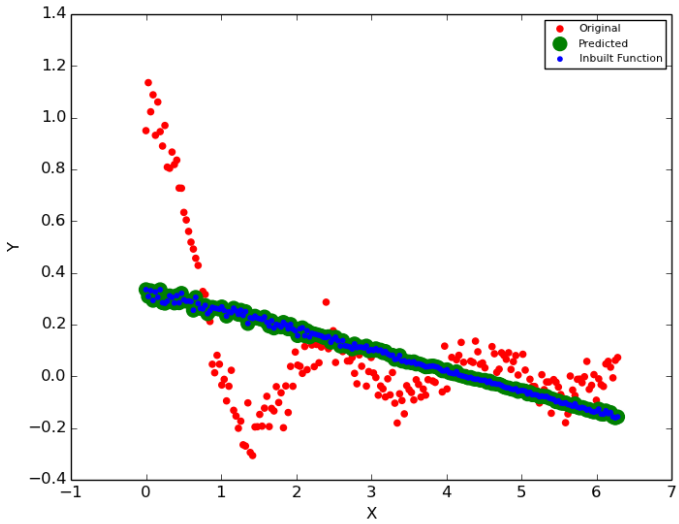
scatter plot of the data was generated to understand the complexity of the data.

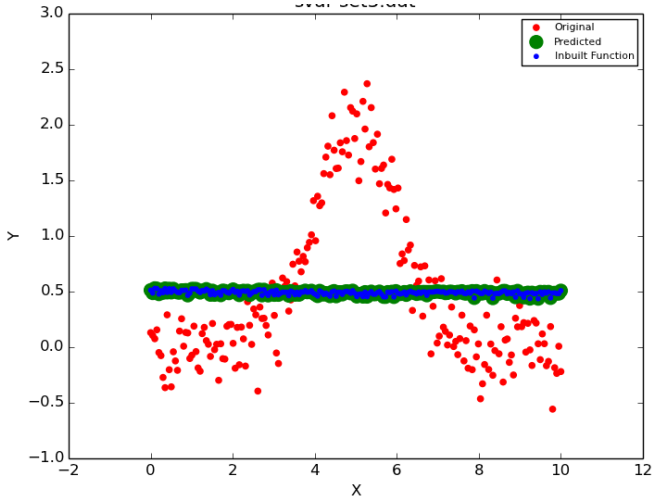
3.1.1 Linear model

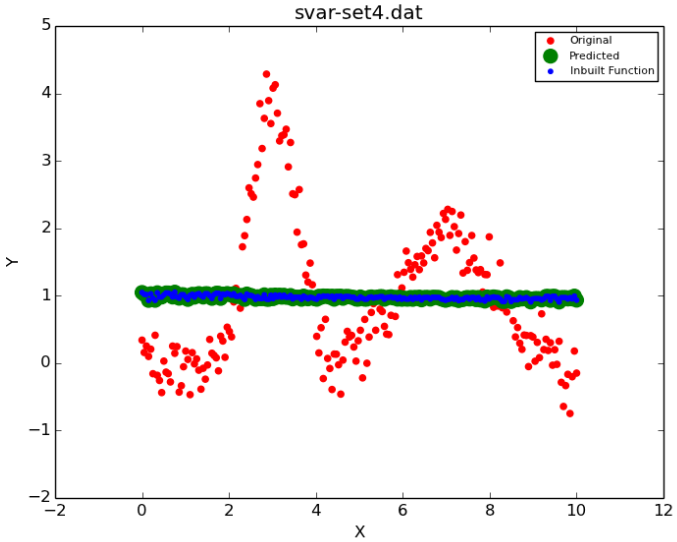
A linear model was used to fit the data. The predicted values were plotted along with the scatter plot of the original data. An inbuilt function from sklearn [3] was used to predict the values for the same inputs and these results were also plotted. It was observed that the values predicted by the inbuilt function and by the function implemented for the purpose of this assignment overlapped.

The First image has been enlarged for better viewing


svar-set1.dat
Original
Predicted
Inbuilt Function
X
Y


Original
Predicted
Inbuilt Function
X
Y


Original
Predicted
Inbuilt Function
X
Y


svar-set4.dat
Original
Predicted
Inbuilt Function
X
Y

Sample output for Dataset: svar-set2.dat

	Intercept	Coefficient	Testing Error	Training Error
My Function	0.332477456568	-0.0777048948454	0.061200692676	0.0594871873194
Inbuilt Function	0.33244798	-0.07768946	0.0622020605087	0.0595726065787

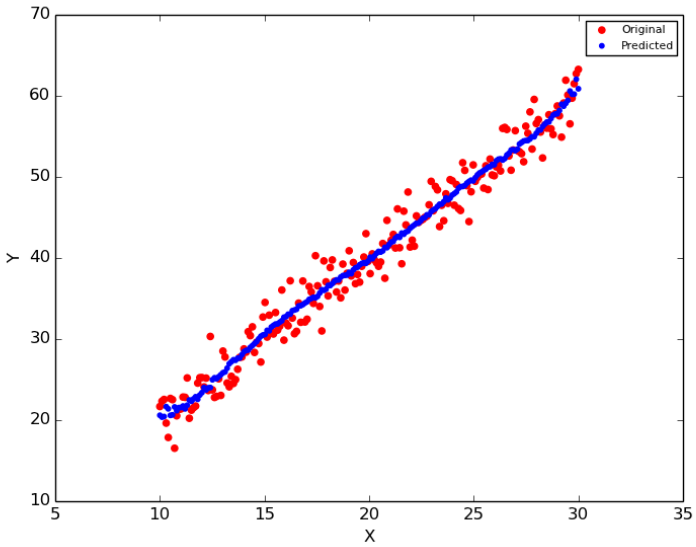
It was observed for all data sets that the predicted values were very similar. This is apparent from the plots above.

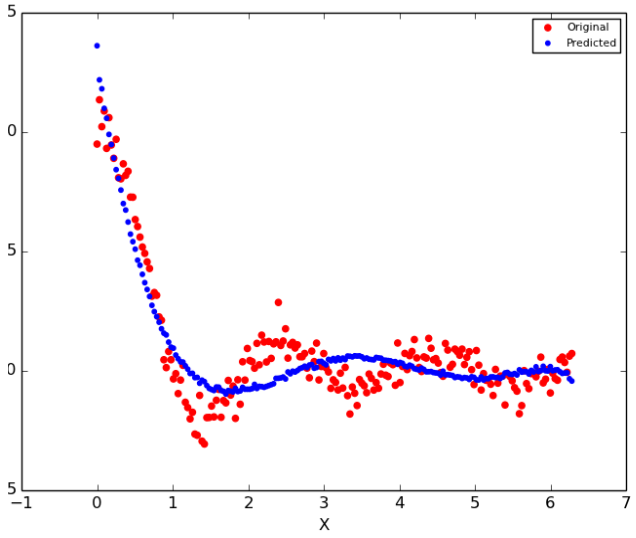
3.1.2 Conclusions so far

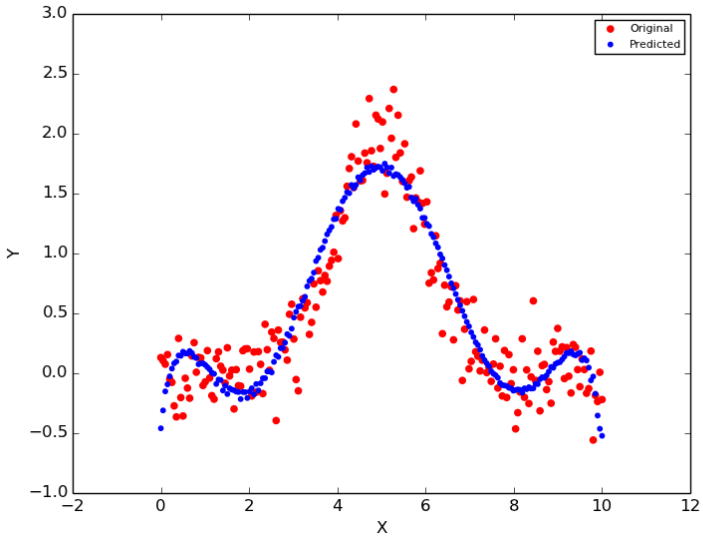
It was observed that a linear model fails to do an ideal job in fitting the data, except in the case of dataset 1 where the input x and output y seem to have a linear relationship. This can be observed in the scatter plot. In datasets 2, 3 and 4, the linear model does not fit very well, it is very general and under-fitted. The errors are high for these files. The increase in the error is not apparent, because this error value is an absolute measure and does not take into consideration the scale of the data. A better measure would be Relative Mean Square error.

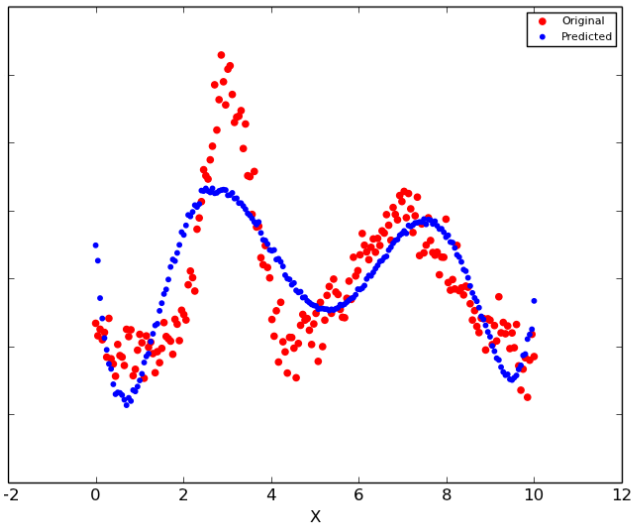
3.1.3 Non-Linear Model

Since Linear Models don't fit the data very well (under-fitting), Different polynomial models were tried on different subsets of the data and the best model was chosen in terms of testing error. Testing error is a better criteria than training error here because, it was observed that as we increase the degree of the polynomial, the training error decreases. The model would fit the data better. But, the model would fit the data too well, causing over-fitting. This is the case where the training error would be less but testing error would increase because of less generalization. The predicted values were plotted along with the original values.







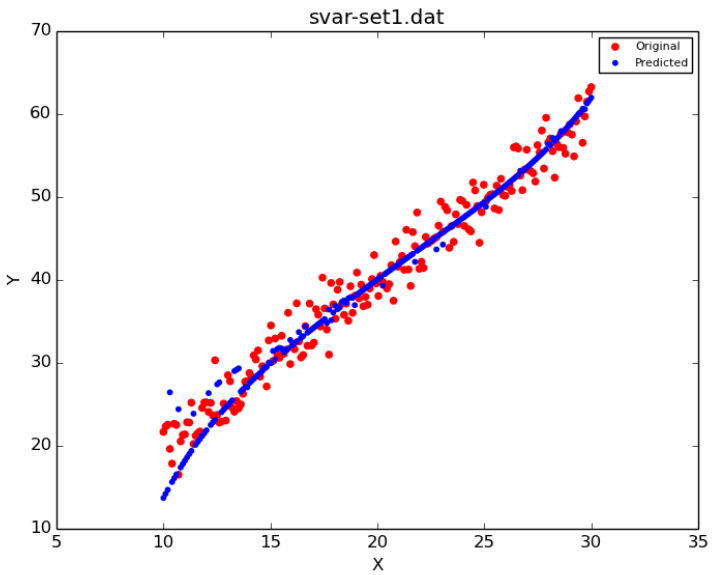


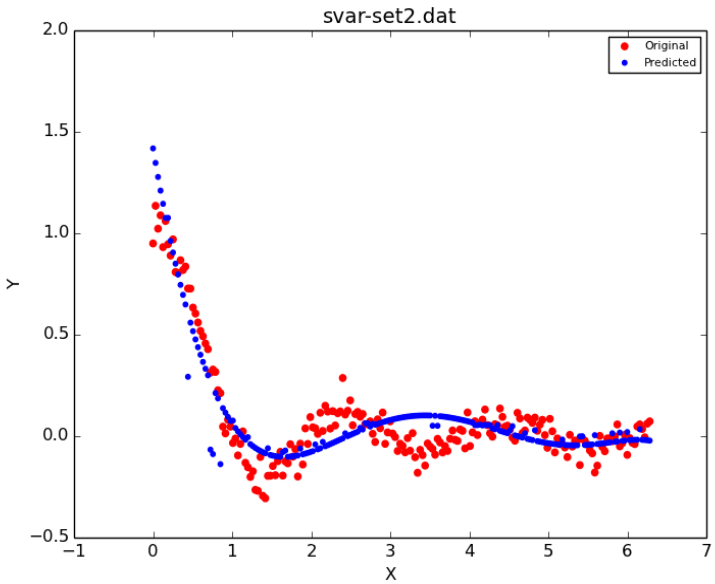
It can be observed that polynomial models fit the data better.

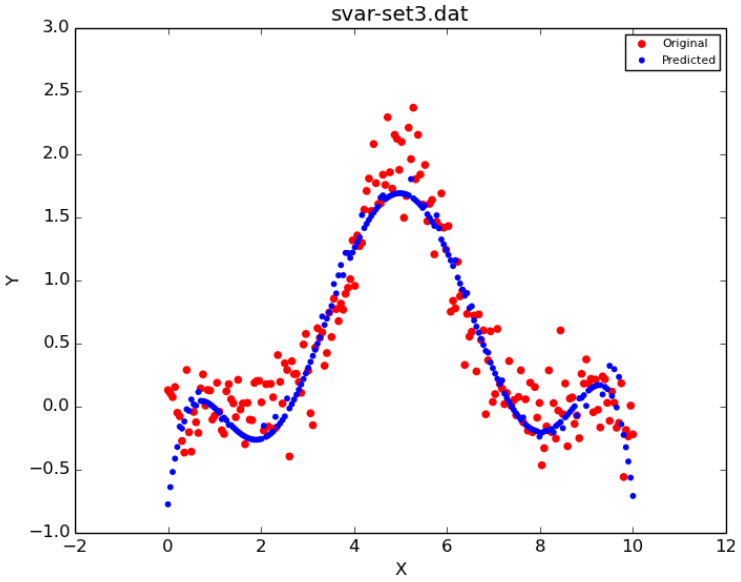
Sample output:

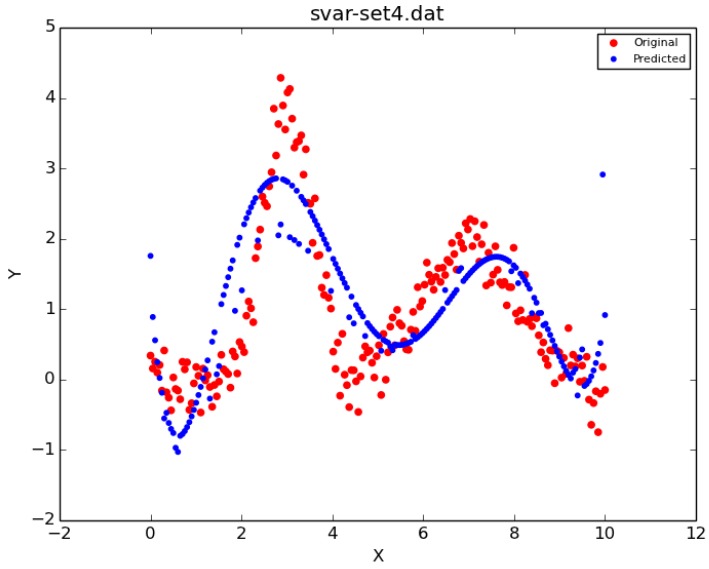
FileName : *svar-set1.dat*
Fitting Polynomial model
Optimal Degree : 3
Intercept : 885.288362471
Coefficients: [-2.94641234e+02 , 4.02609024e+01 , -2.83845940e+00 , 1.09890966e-01, -2.22017046e-03, 1.83236287e-05]
Testing Error : 4.11927853594

Next, the amount of training data was reduced to 20 percent of the original amount to notice the effects on performance. As expected, the performance deteriorated. The scatter plot showed more outliers and naturally the errors also increased.









Sample Output:
FileName : svar-set4.dat
Fitting Polynomial model

Optimal Degree : 6
**Intercept : ** 1.39468863515
**Coefficients : **[-7.78105242, 8.98236102, -3.65561706e, 6.76600159e-01, -5.82068152e-02, 1.88914744e-03]
**Testing Error : **0.60192187035

Reducing training data

Optimal Degree : 6
**Intercept : ** 0.460274779187
Coefficients : [-3.38611528e+00 5.84914343e+00 -2.68619096e+00 5.24229505e-01 -4.63080480e-02 1.52385922e-03]
Testing Error : 0.75822496709

We can see that the coefficients are different after reducing the training dataset and the testing error increases. In the scatterplot for svar-set4.dat, we can observe that there are more outliers after reducing the training data.

However, if we take the example of dataset 1, the testing error decreases from *0.0064* to *0.0054* after reducing the training data. This can be explained. When the size of the training set is decreased, the model becomes more general, this helps to reduce training error.

3.2 Multiple Features Dataset

3.2.1 Regression in higher dimensional feature space

All the four datasets are loaded and mapped to a higher dimensional feature space using sklearn's inbuilt function PolynomialFeatures [4]. The data is mapped to dimensions of 2, 3 and 4. Linear

regression is performed in each of these higher dimensions and a single model is chosen based the testing error that each model produces.

Sample output:
File Name: mvar-set1.dat
Mapping to higher dimension of 2

Intercept : 1.02230218338
Coefficients : [0.99752589 0.99048461 -0.01260975 -0.00847393 -0.00643214]

Mapping into higher dimension of 3

Intercept : 1.02230218338
Coefficients : [1.00248539e+00 1.00537442e+00 -1.26097513e-02 -8.47393334e-03
-6.43213699e-03 -2.31159943e-03 -1.64092663e-02 5.84897669e-04
3.15715316e-03]

Mapping into higher dimension of 4

Intercept : 1.06730280003
Coefficients : [1.00248539e+00 1.00537442e+00 -3.57546839e-02 -2.55666363e-02
-8.68183352e-02 -2.31159943e-03 -1.64092663e-02 5.84897669e-04
3.15715316e-03 5.49722890e-03 -3.80671172e-03 2.56108986e-03
1.06530316e-02 2.15593024e-02]

Optimal Dimension : 2, **Testing Error :** 0.259616633312

In each dimension space, the testing error is computed and the model with the least error is chosen as the best. In the above example, the optimal dimension is 2.

3.2.2 Gaussian Kernel Function

To solve the dual regression problem a Gaussian Kernel function was used. The idea to calculate the Gaussian Kernel in Python using numpy was obtained from stat.stackexchange.com [5].

The Kernel method was applied only on dataset 1.

File Name : mvar-set1.dat
Intercept : 2.50094197e+09
Coefficients : -2.08515572e+09

3.2.3 Gradient Descent Algorithm

The following sample output is for dataset **mvar-set4.dat**
Explicit Solution
Intercept : 0.0101203623075
Coefficients : [4.66764413e-05 1.09406142e-04 3.95241972e-05 2.39375741e-04
-1.83675810e-06]

Iterative Solution
Intercept : 0.0101676493
Coefficients : [4.66204117e-05, 1.09829000e-04, 3.97418302e-05, 2.39602639e-04, -1.35885240e-06]

It was observed that the Iterative algorithm produced a theta matrix that was very close to the explicit solution. As expected the testing error was also very similar.

Sample output of testing error of explicit solution and iterative solution
File Name : mvar-set4.dat
Iterative Testing Error : 0.00418950358788
Explicit Testing Error : 0.00418950268543

References

[1] Parametric Regression definition: Link http://www.mathworks.com/help/stats/introduction-to-parametric-regression-analysis.html?requestedDomain=www.mathworks.com

[2] k- Fold Cross validation function documentation: http://scikit-learn.org/stable/modules/cross_validation.html

[3] Sklearn inbuilt function documentation: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[4] Sklearn Polynomial Features: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

[5] Helo from stackExchange: http://stats.stackexchange.com/a/15817