CS-584 – Assignment 1

Parametric Regression
Shalin A. Chopra
A20355181

1. Problem Statement

This assignment aims at solving the Parametric Regression using two different techniques, Single variable regression and Multivariate regression. The Regression problem is to fit the models according to the datasets. All the models have numerical responses. The implementation tries to design a model in such a way that it will try to fit datasets and produces low testing data errors. The implementation includes different datasets, 10-Fold Cross validation for training and test datasets, high dimensional feature space mapping and solving the iterative and Kernel methods. The report includes complete and comparative analysis of the different models in different techniques.

2. Proposed Solution

Solution includes different models to try and fit the given data. The datasets include single and multiple features. The different models used are Linear Model and Polynomial model for Single variable and for Multivariate we have implemented Explicit Solution, Iterative Solution using Stochastic Gradient Descent and solved the Dual Regression problem using Gaussian Kernel Method. The Training and Testing error are calculated at each Fold for 10- Fold Cross validation. The metric used for calculation is Residual Sum of Errors (RSE) and Mean Square Error (MSE).

3. Implementation Details

The assignment is implemented using Python v. 3.5 and using PyCharm IDE. The dataset are loaded directly from the URL of those datasets hence, requiring internet connectivity. For all the datasets I have implemented 10-Fold Cross validation which divides the dataset into training and test set. The Training data is used for Training the Model and using the same model the test data is tried to fit over it. The cross validation take 90% training data and 10% test data. **Linear model** I have implemented simple linear model which takes the Objective function H (Ɵ) minimizes it to get the Ɵ values and taking into account these value the Target values for particular feature value. **Polynomial Model** I have solved polynomial model by increasing the feature vector by polynomial degree and tried to model the data in such a way that it doesn't tries to overfit the data. Here, I have checked for different polynomial degree values. For Multivariate, **Linear regression** in high dimensional feature space is performed. For, this the number of features (n) are mapped to higher dimension (N) such that N>n, but taking into account the constraint that the value for N < m (Number of Samples) so that the predicted target values are correct. Different mappings are performed and the test errors (MSE) are recorded and compared. The regression model using **iterative solution** is performed using **Stochastic Gradient Descent** Algorithm, the cost function for the same is calculated and the graph for the

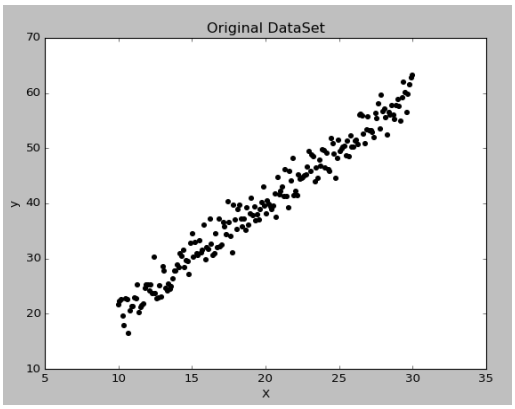
decreasing cost function is plotted accordingly. Here, the main thing is to determine what should be the value of **Learning Rate (α)** and when will the Gradient Descent Algorithm converge i.e. Number of Iterations. I tried various values and the one which makes the value of **Cost Function J(Θ)** decrease as number of iterations increases is selected. The dual regression problem is solved using the Kernel tricks. Kernel implemented is **Gaussian Kernel**; it takes into account calculation of similarity between feature vectors and predicts the target value accordingly. Firstly, similarity is found between the features and **Gram Matrix** is calculated, from the Gram Matrix we calculate α, this is used to calculate the Target values along with the Kernel Function **K (X^i, x)**.

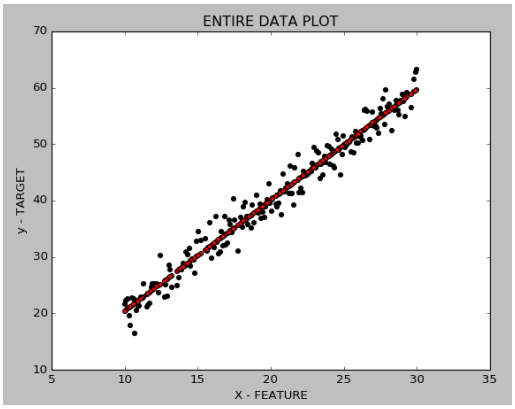
Execution of program:

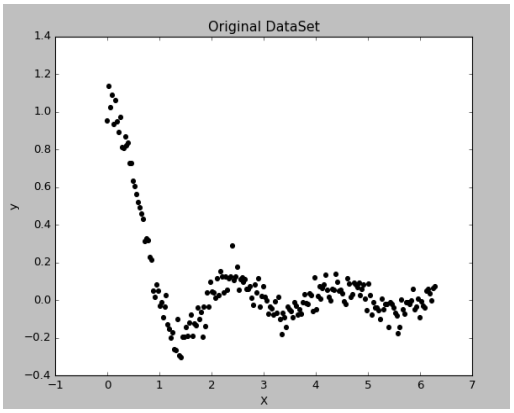
1. The program for polynomial Model accepts a user input for Degree of polynomial.
2. The program for High Dimensional Feature Space accepts Dimension from the user.
3. The datasets are loaded directly from the URL's, Internet Connectivity is must.
4. The Multivariate Regression has 2 real datasets taken from UCI Machine Learning Library.

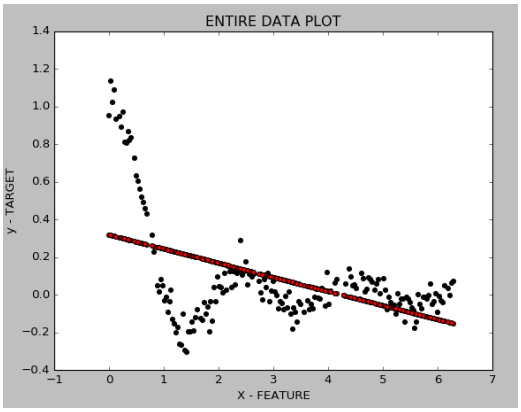
4. Results & Discussions

a. Linear Regression (Single Feature)

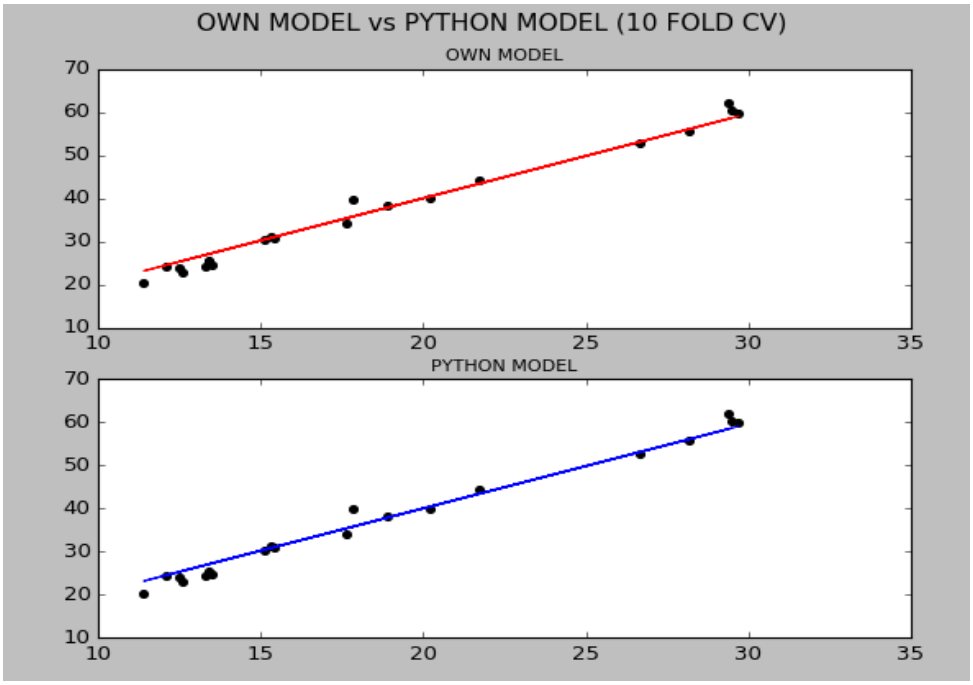








The above plots are for Linear Model using single feature. We observe that when the data is linear the Model tries to fit the data set. But, in the 2nd plot the linear model doesn't fit the data entirely. Since, it's a linear model it forms the equation of a line, thus plotting a linear line with coef and intercept (taking the slope-intercept form)



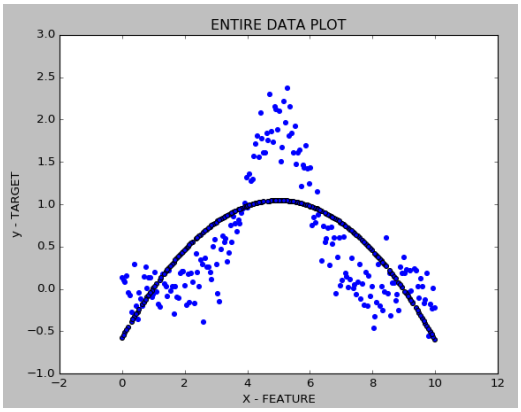
The above plot compares the 10-Fold CV output with the read-made python function for Linear Regression. The values are same for both plots thus giving the same output for the same. The following table shows the values for Θ and RSE & MSE errors for Training and Testing Data.

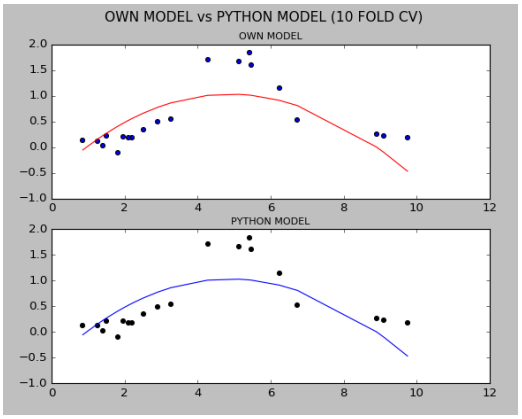
Θ = 0.17343865, 1.98698647 (same for both)
Test Data RSE = 0.861909469546
Test Data MSE = 5.13549946273
Training Data RSE = 0.956428280857
Training Data MSE = 4.13773234262

b. Polynomial Regression Model

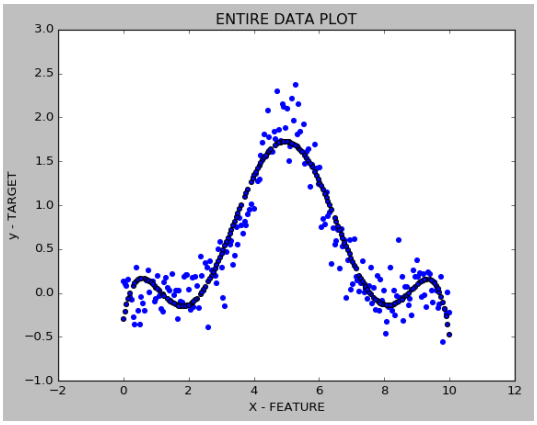
The below plots shows the polynomial model plots for different degrees of polynomial. I have considered two different plots for Degree = 3 and Degree = 6. As seen the degree 3 plot tries to fit the data but doesn't quite fit it better when compared with degree 6 model. This might be case of data **underfitting** the model. The train error is also high when compared with test error. The Model with degree 6 almost fits the data nicely and a good data model is obtained. The errors tell that the degree 6 model is a good option to fit the data with low test and train errors. When the degree is increased above 7, the model tries to fit every point in the dataset thus leading to **Overfitting** of data to model, though the error might pretty small put there has been a lot of computation cost involved.

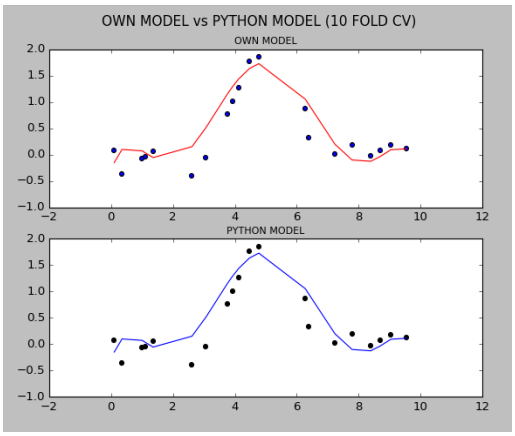
Degree = 3





Degree = 6





	Degree = 3	**Degree = 6**
RSE Test Data	0.501254313888	0.109158960947
RSE Training Data	7.75584974496	2.85788034442
MSE Test Data	0.421972136449	0.0745412460684
MSE Training Data	0.236165757894	0.0623044719302

c. Reduced Data

For this experiment, I have decreased the Training Data from 90% to 30% and 50% and observed the difference between them for different testing data. The results are shown below:

Linear Model:

	Training = 30%	**Training = 50%**
Training Data Samples	60	100
Test Data Samples	20	20
RSE Test Data	3.80396442687	0.151395764275
RSE Training Data	4.22109769721	2.26786227586
MSE Test Data	3.01165316646	5.60118092672
MSE Training Data	4.19177852706	4.21967832175

Polynomial Model (Degree 3):

	Training = 30%	**Training = 50%**
Training Data Samples	60	100
Test Data Samples	20	20
RSE Test Data	2.86551610071	36.4757035616
RSE Training Data	19.7432426653	5.23312685057
MSE Test Data	0.247814214062	0.207286617838
MSE Training Data	0.269537177342	0.275125747077

As observed when the Training data is pretty small, the model prepared doesn't make good predictions. The error rate is quite high for 30% of data which reflects on test data predictions. When training data is 50% the model shows the same effect instead just the test data RSE is less when compared with 30% data. Thus, we can conclude that a large training dataset is required which covers all aspects and tries to prepare a model which produces good results on any Test dataset case.

d. Multivariate Regression

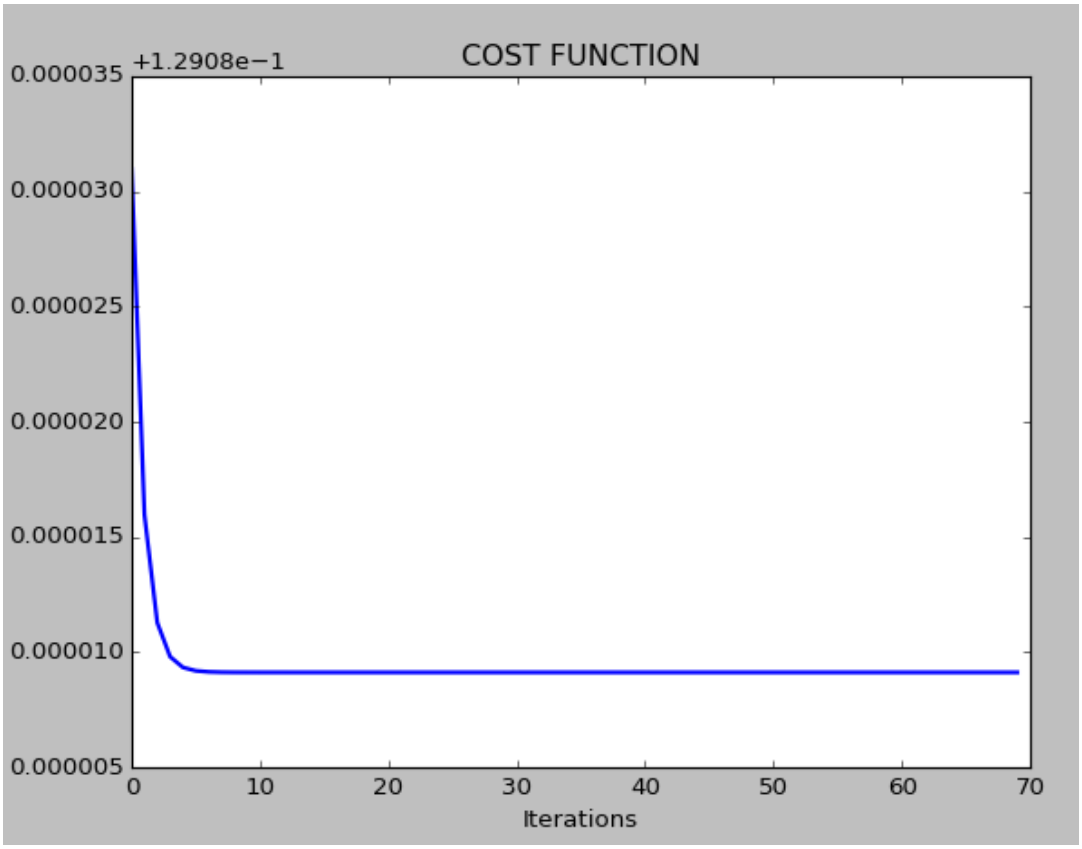
Dimension of Map	**3**	**4**	**5**
Original No. of Features (n)	2	2	2
Mapped Feature (N)	10	15	21
MSE Test Data	0.2164697	0.2141677	0.2140714
MSE Training Data	0.2622920	0.26128243	0.2610334
Time to execute (in sec)	0.17230	0.24515	0.3412

The above result is for dataset 1, given by professor. I have mapped the data to 3 different high dimensional feature space. As observed by there is not much difference when the feature vector is 2 originally. The values obtained are very close to each other, and thus producing accurate results. But when the case is of data with larger features and those when mapped to higher space produces inaccurate as then the N becomes larger than m (i.e. number of samples). The only thing that we observe increases is time, as computation increases involving different matrices of different sizes. I have also tested real dataset, the results obtained for the Boston Housing data (506 samples) set is below when mapped to higher space:

Dimension of Map	**2**	**3**
Original No. of Features (n)	13	13
Mapped Feature (N)	105	560
MSE Test Data	9.17924746967	5525208827.24
MSE Training Data	6.0947747462	0.011068334
Time to execute (in sec)	0.3302	1.5002

The above table shows that when the data is mapped for 3-dimension the errors increase drastically this is due to the fact that, the number of samples i.e. 506 are less that number of features i.e. 560. So, to compute better results the m > N. Thus, mapping data always to higher dimension doesn't quite increase the prediction quality of the model.

e. Iterative Solution – Stochastic Gradient Descent



I have implemented the Stochastic gradient descent for comparison of Iterative solution with the Explicit solution. The reason for selecting the Stochastic gradient descent is that it pretty fast when compared with Batch G.D as it has to compute the Θ for the entire feature at once and then moves forward. In Stochastic approach the initial Θ is selected, and for every iteration a new Θ is calculated by subtracting it from the previous value along with the learning rate and cost function. The selection of learning rate plays a major role in calculating the cost. The above graph shows the correctness of my implementation and selection of learning rate as the cost function goes on decreasing with the increase in number of iterations. The values for error are given below. If the dataset2 is mapped to higher space feature, the Cost function goes on increasing and thus the selection for Learning rate needs to be changed to 0.00001 or less, but this isn't a good learning rate as it must be between $0.001 < L.R < 10$.[1]

MSE Test Data = 0.216731467472
MSE Train Data = 0.263385296099

f. Dual Regression Problem – Gaussian Kernel

For dual regression problem, Gaussian Kernel method is applied. To calculate the distances or similarity between features. For Dataset1 and Dataset2, the error rates ae given below:

Dataset 1:
MSE Test Data = 0.827612805365
Time = 11.5265 seconds

Dataset 2:
MSE Test Data = 0.0289384252232
Time = 14.3841 seconds

The time taken for Gaussian method is pretty high when compared with primal solution problem solved ealier. This is due to the fact that we need to calculate similarity between every feature with every other features and samples, thus involving a lot of computation. The matrices are of m X m form, and thus produces very high computation time.
The time taken for primal problem is low as there we don't calculate the value of similarity, since we have direct equation to get the value for Θ, and then compute target values.
For, dual problem we have eliminated the cost of calculating Θ, by replacing it with Gram Matrix and Kernel Method.

For dataset3 and 4, which has 5 features each and about 100000 samples, when I ran the program the numpy.dot function went into memory error thus casuing abrupt close of the program.

5. References

1. http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=MachineLearning&doc=exercises/ex3/ex3.html
2. https://archive.ics.uci.edu/ml/
3. http://cs229.stanford.edu/materials.html
4. http://www.numpy.org/
5. http://www.mathworks.com/help/stats/introduction-to-parametric-regression-analysis.html?requestedDomain=www.mathworks.com