Shorabh Dhandharia
A20361514

HW1 - Parametric Regression

Problem Statement

In this assignment I have implemented different techniques of parametric regression for single and multiple features dataset. To test performance of my algorithm I have used the 10 fold cross validation. I have also compared the performance of my algorithm with Python ready-made functions.

Proposed Solution

For Single Variable Regression,

- To load each of the single feature data sets and plotted the graph to see the complexity of the problem
- To fit a linear model to the data and compute the training and test error. Plot the regression model implemented by me on test data.
- To test different polynomial models on different subsets of data and justify one optimum model.
- To reduce the amount of training data and observe the effect on performance of linear and polynomial models

For Multivariate Regression,

- Load multiple feature data sets and map them to higher dimension feature space using combination of features
- Perform Linear Regression in high dimensional space and evaluate different mappings in terms of testing error and choose a single model and justify. Also compute Root Squared Error (RSE) and Mean Squared Error (MSE) for training and testing data
- Solve regression problem using iterative solution and compare it with the above implemented explicit solution
- Solve dual linear regression problem using Gaussian Kernel function and compare time performance and accuracy with primal regression problem.

Implementation Details

I have implemented all the above proposed solutions using Python 3. In PyCharm 5.04.

I am reading the dataset from the file and getting it into different lists and then converting them into arrays. I am then applying the 10 fold cross validation to divide training and test data to implement the algorithm.

Single Variable Regression

In Single feature linear model I am calculating the two matrices A and B and solving them to compute Thetha. Using Thetha I am calculating y hat for training and test data. And in the last step I am calculating the RSE and MSE for training and test data.

I have then fit my dataset into a linear model using Python ready-made function and calculated RSE and MSE and compared them with my algorithm.

For Single Feature Polynomial model I am taking user input for polynomial degree and then using that I am concatenating my training data into another matrix Z. Using Z matrix I have calculate my Thetha. Rest I have performed all the steps mentioned above.

For evaluation, I have then tested different polynomial models on different subsets of data and also reduce the training data to observe the effects of performance for linear and polynomial models

Multivariate Regression

In multivariate regression, I am reading the datasets from the file and getting it into two lists X and Y. In X, I am getting the entire feature set. I am then converting them into array and applying the 10 fold cross validation. I am creating a Z matrix by transferring the training data to higher dimensional space. I have then calculated Thetha with shape equals to number of columns in Z. Using Thetha, I have calculated y_hat, RSE and MSE for training and test data.

I have implemented the iterative solution using Stochastic Gradient Descent with similar steps as above. I am taking initial assumption for Alpha and Thetha. And then applying the formula, I have computed Gradient and successive Thethas and iterated for predefined no. I have calculated the cost for each iterations and plotted the graph. Also, I have compared the iterative solution with the earlier one which is the explicit solution.

Lastly I have also implemented the Gaussian kernel to solve the dual problem of linear regression and compared the time performance and accuracy with the primal regression problem.

Results and Discussions

Single Feature Linear Regression

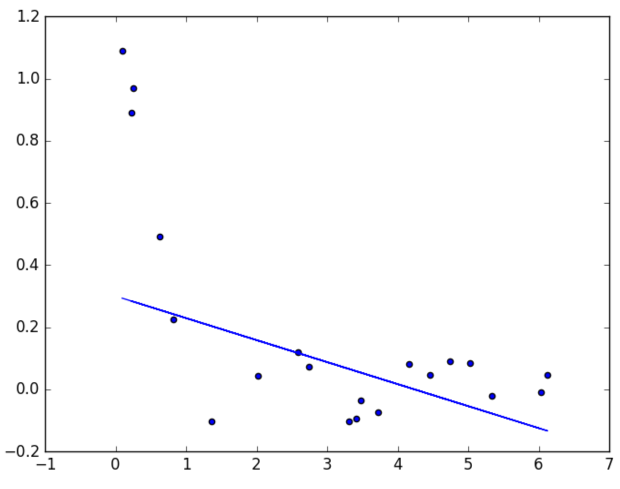
The below table represents the Thetha values received from the model which I implemented and the Python's ready-made function. The graph represents the plot of testing data.

I received same Thetha values in both the case along with MSE and RSE values as I have implemented the same linear model which python's inbuilt function does.

For comparison, it can be seen that Test RSE is much smaller than Train RSE whereas MSE is remains almost the same in both the case.

	Thetha 0	Thetha 1
Manual Thetha	0.30030491	0.300305
Python Thetha	-0.070789	-0.07079

	Manual	Python
TestRSE	0.5963431	0.596343
TestMSE	0.09011061	0.090111
TrainRSE	42.460806	42.46081
TrainMSE	0.05647698	0.056477



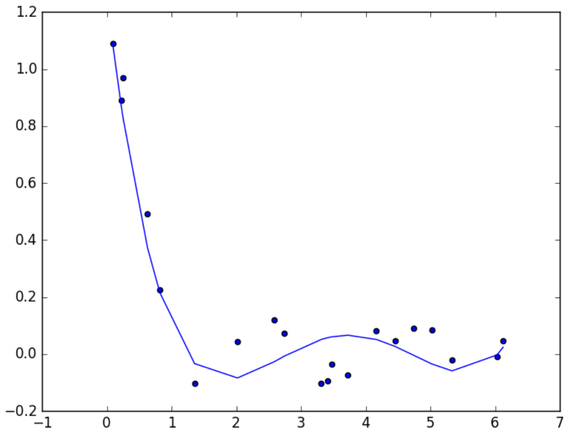
Single Feature Polynomial Regression

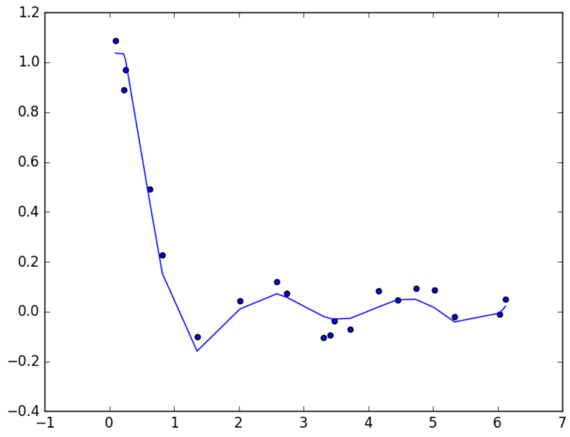
The below table represents Thetha values from the linear model which I implemented and Python's ready-made function. After evaluating for different polynomial degrees I have seen that as degree increases, more no. of training points can be plotted on the graph and more accurate result I get. The only concern in increasing degree is the computation cost since it has to analysis more no. of training samples and compute Thetha based on that. I have chosen the linear model with degree 4 and I got the same result as Python's inbuilt polyfit() function. The accuracy can be seen from the RSE and MSE which is very small.

For comparison I have shown tables and graphs for degree 8 as well from which it can be seen that the Test MSE is much lesser than what I got with degree 4. Also, it tries to connect more no. of training points. As mentioned above even here RSE is smaller for test data whereas MSE remains almost the same

	Thetha 0	Thetha 1	Thetha 2	Thetha 3	Thetha 4
Manual Thetha	1.26100608	-1.94169	0.960312	-0.18843	0.012738
PolyFit Thetha	1.26100608	-1.94169	0.960312	-0.18843	0.012738

Degree = 4			Degree = 8		
	Manual	PolyFit		Manual	PolyFit
TestRSE	0.45559242	0.455592	TestRSE	0.30578859	0.305789
TestMSE	0.00925643	0.009256	TestMSE	0.00329904	0.003299
TrainRSE	14.7925779	14.79258	TrainRSE	6.99802293	6.998023
TrainMSE	0.01177223	0.011772	TrainMSE	0.00390952	0.00391





Reduction of Training examples

As it can be inferred from the table below, as we reduce the no. of training samples RSE for datasets tends to increase while MSE remains almost the same. The reason is obvious since we predicting result with use of less no. of examples it will not give me an optimum result and error will increase.

	Number of Training Examples			
	180	160	140	120
TestRSE	0.596343	4.533236265	66.57366	150.7019905
TestMSE	0.090111	0.051883919	0.054145	0.044359871
TrainRSE	42.46081	640.6675676	7407.781	16110.44599
TrainMSE	0.056477	0.060501553	0.061405	0.063920502

Multivariate Linear Regression

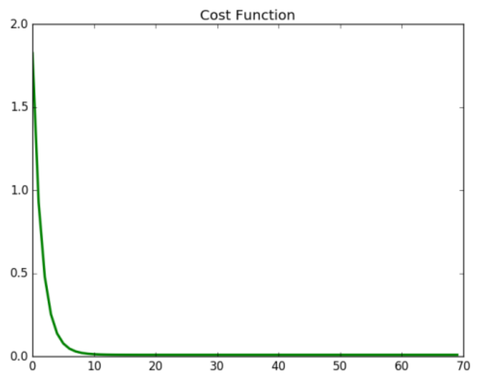
I implemented the multivariate linear regression by transferring the dataset into a higher dimension. The below table shows MSE calculated for degree up to 4 which gives me almost same result.

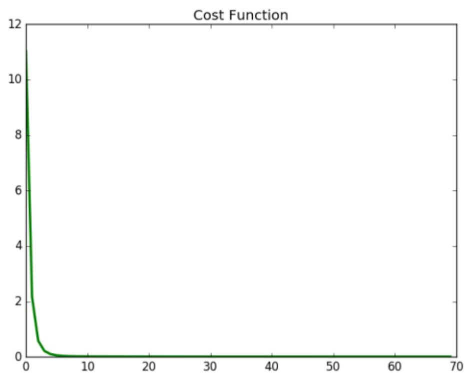
	Degree = 2	Degree = 3	Degree = 4
TestMSE	0.017810419	0.075720132	0.075602755
TrainMSE	0.020146571	0.082780938	0.082896444

Stochastic Gradient Descent

	Degree = 1	Degree = 2
TrainMSE	0.01995574	0.019968321
TestMSE	0.019537748	0.01950896

I didn't implement Batch Gradient Descent algorithm since it computes Thetha for each feature which increases the computation cost.
From my evaluation of Stochastic Gradient Descent algorithm the important factors are alpha and polynomial degree. I used alpha as 0.00001 to implement degrees upto 2, after that the cost of the iteration starts increasing and the graph gets a mirror reflection. But when I increased the value of alpha to 0.0005, I can get the result graph for degree 1 only. The below left hand side graph is for degree 1 and right hand side is for degree 2.





Gauss Kernel

I have implemented the algorithm to solve the Dual regression problem using the Gauss kernel using the formula which was explained in the lecture.

As per my evaluation, the result is highly dependent on the value of Sigma. Currently I have used sigma value as 1 and I got MSE of 0.77 whereas for primal problem I got MSE around 0.07.

The time taken to solve this algorithm on my computer was around 20 seconds since there are more than 2K values which will be tested against 250 test data whereas primal problem takes hardly around one second to implement the algorithm.

I was able to run this code successfully for first two data sets only since for other two it throws me Memory Error, since there are 100K values.

References

http://docs.scipy.org/doc/numpy/reference/

https://www.cs.cmu.edu/~tom/10701_sp11/slides/Kernels_SVM_04_7_2011-ann.pdf

http://pythonhosted.org/PyQt-Fit/Param_tut.html

Note:

All the codes has been executed and evaluated for the datasets provided with the problem statement.

To run the code successfully you need to change the file name and location in all the codes, references are provided below.

Gauss.py:14, SinglePolynomial.py:6, SingleLinear.py:6, MultivariateRegression.py:6, IterativeSolution.py:6