CS584 Assignment 1: Report

Gady Agam

Department of Computer Science

Illinois Institute of Technology

February 18, 2016

Abstract

This report is for assignment in CS584. The problems that we confront is Linear Regression and its variants like Single Feature, Multi Feature, Polynomial Models, usage of Primal and Dual Solutions. We also implement ways to map features of lower degrees to higher degrees and the computation issues of them and the way Gaussian Kernel function. Their performance in terms of accuracy and time is computed.

1. Parametric Regression
   1. Problem Statement:

Fit a Linear model to a complex dataset and do parametric regression, compute the training and testing error. Use different polynomial models and compare their performance with the in-built Scikit Learn functions. Load other datasets with multiple features, map them to higher dimension and evaluate its performance. Solving the linear regression with by both implicit and explicit solution and analyze their performance. Solve the dual linear regression problem using Gaussian Kernel functions. Compare the results of both Primal and Dual Solutions.

* 1. Proposed Solution:

The above problem statements can be solved my implementing variants of parametrized Linear Regression algorithms, for this case, we implement Linear Regression with Single Feature, Single Feature Polynomial model, Multiple Feature with Polynomial model. All these algorithms are implemented in Python 2.7 using Ipython Notebooks. Packages like numpy, matplot are widely used in this implementation.

* 1. Implementation Details.

1. The given datasets are loaded and a scatter graph is plotted with the all features marked as green dots and the regressed values as blue dots.

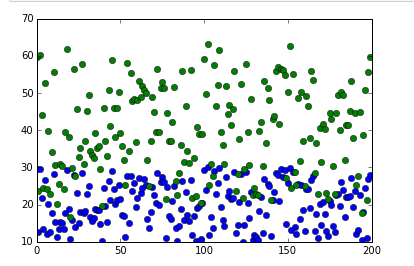
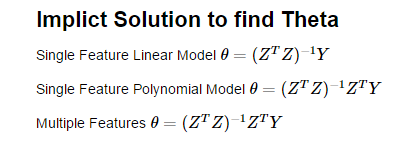


Fig.1 Features: Green Dots, Regressed Value: Blue Dots.

1. Parametrized Regression is achieved by supervised learning, a dataset ‘svar-set1.dat’ is taken and it is split into Training and Testing data. Using the training data (feature, regressed value)



The matrix Z is created from the feature vectors with 1’s as the first column, it is done to computation complexity.

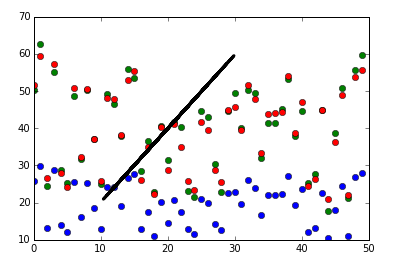
1. From the learnt theta we can make the prediction for new data. We call the Ŷ and it is computed using the formula Ŷ = ƟT . Z
2. From the above steps we can do single feature linear regression, but we need to find how accurate the results, this is split into two stages (i.e) Training Error and Testing Error.

The Training is calculated using a technique called **Cross Validation**. A variant called KFOLD Cross Validation is implemented, it is done by splitting the Training data into train and test data for k-folds. If the length of data is 10 and folds required is 2 then indexes would be split as ([5, 6, 7, 8, 9] - Train, [0, 1, 2, 3, 4]- Test) and ([0, 1, 2, 3, 4]-Train, [5, 6, 7, 8, 9]-Test). This would give an insight on how the model will generalize to a new or unknown dataset

1. Using the formula Ŷ = ƟT . Z we have the predicted value with this a regression model can be plot as graph.

Cross Validation Error with 10 folds: 4.38419193846

Mean Squared Test Errors : 4.29856341131



**Blue Dots: Features, Green Dots: Given regressed values, Red Dots: Predicted regression values. Black Line is the plot between the Features and predicted values**

1. Comparing the results of the implementation with Scikit Learn’s built in function

By Scikit Learn Mean Squared Error: 4.29856341131

Time Taken by Scikit Learn: 0.0010516007078

By implemented Model Mean Square Error 4.29856341131

Time Taken by Implemented model: 0.00252315745456

We can see that the mean square error for the predicted data is same but scikit learn’s implementation takes lesser time. It is twice as fast as the implemented algorithm.

1. An Experiment is conducted to identify the degree of the polynomial functions that needs to be used, We try different degrees from 1 to 5 and find the Training and Test Error. We take both Training and Testing errors into consideration in order eliminate the scenario of overfitting models

For i = 1 Training CV Error 4.38419193846 Test Error 4.29856341131

For i = 2 Training CV Error 4.43687922999 Test Error 4.34355934721

For i = 3 Training CV Error 4.49829478324 Test Error 4.21425175253

For i = 4 Training CV Error 4.5128074272 Test Error 4.17825535149

For i = 5 Training CV Error 4.56445239708 Test Error 4.17958158238

For i = 6 Training CV Error 4.42132648852 Test Error 4.24161652809

For i = 7 Training CV Error 4.47729185832 Test Error 4.23941757661

For i = 8 Training CV Error 4.50407572994 Test Error 4.2563135219

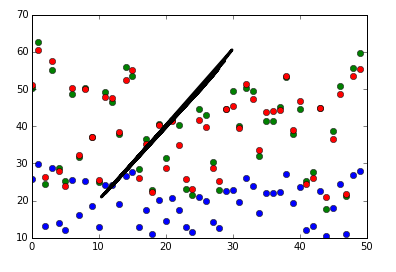
For i = 9 Training CV Error 4.55680027476 Test Error 4.26733904806

We choose polynomial with degree as 4 because it has optimal Training and Testing Error, So our model doesn’t overfit and has better generalization.

1. For a Polynomial Model with degree 4 we have Test errors as shown below.

Mean Squared Error using Single Feature Polynomial Model: 4.17825535149

Residual Sum Square Error using Single Feature Polynomial Model: 208.912767574



**Blue Dots: Features, Green Dots: Given regressed values, Red Dots: Predicted regression values. Black Line is the plot between the Features and predicted values**

1. When we do the analysis by varying the Training Data Size we see that the Testing Errors increases when training size decreases. E.g. With training size of 50 we get the highest Testing error

Mean Squared Error using Single Feature Polynomial Model: 4.7143245382

1. Using the “PolynomialFeatures “we can map feature matrix to higher dimensions. PolynomialFeatures takes in the feature matrix and the higher dimension power to which the matrix needs to be raised.
2. As the degree of the Polynomial function is by choice, we do an experiment to find the best degree for this data set.

Where ‘I’ is the degree of the polynomial, we compute the training Cross validation and testing error,

For i = 1 Training CV Error 0.259956250375 Test Error 0.258454399093

For i = 2 Training CV Error 0.261044149799 Test Error 0.257994505951

For i = 3 Training CV Error 0.263784664564 Test Error 0.257551911174

For i = 4 Training CV Error 0.26476956965 Test Error 0.256361459492

For i = 5 Training CV Error 0.267055630264 Test Error 0.25827542428

For i = 6 Training CV Error 0.267519335005 Test Error 0.260915363766

For i = 7 Training CV Error 0.269367296557 Test Error 0.261745622607

For i = 8 Training CV Error 0.271184639754 Test Error 0.265173658319

For i = 9 Training CV Error 0.273559623058 Test Error 0.265508735814

We choose degree = 4 by considering the trade-off between the Bias-Variance.

1. For Polynomial model with degree 4 with Training set size 1500 Test Set Size 1000, Mean Square Training error: 0.257770245339 Testing Error 0.256139174876, Time Taken for primal solution 0.000837346151457
2. The iterative solution implemented is Gradient Descent with the formula,

 where ƞ is the learning rate. It plays a key role in getting to the minima of the Objective function, in this implementation, multiple ƞ’s were tried and learning rate of 0.002 gets to the minima quickest. When the ƞ = 0.002 we get the theta as found in the explicit solution. So the testing errors is based on the Theta that is found by Gradient Descent.

Say, for ƞ =0.4, Mean Squared Testing Error using Gradient Descent 0.270962288976 and the time taken to compute the thetas by Gradient Descent and Predict the regression values takes more time that the explicit solution.

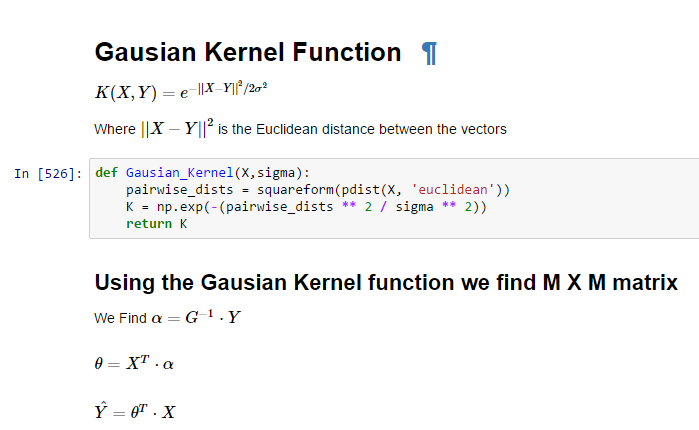
For ƞ =0.4,

Mean Squared Testing Error using Gradient Descent is 0.270962288976

Time Taken (Gradient Descent): 0.00105373897713

Mean Squared Testing Error using Primal Solution is 0.256361459492

Time Taken (Primal Solution): 0.00100712471243



The Mean Squared Testing Error using Dual Solution (i.e.) Gaussian Kernel function is 2.5265505948e+17

And the Time taken (Gaussian Kernel) is 0.463033605036.

The Mean Squared Testing Error using Primal Solution is 0.256361459492, Time Taken for the Primal Solution is 0.00100712471243

So, the Dual Solution takes more time and produces results with less accuracy for this data set.

**From the mean square errors listed above we can see that Primal Solution has a better accuracy than Dual Solution. This result can be justified, as Dual solution is more suited to the cases with less data and more features and Primal Solution for problems involving more data and lesser features.**

1. External Data Set

[**http://archive.ics.uci.edu/ml/machine-learning-databases/forest-fires/**](http://archive.ics.uci.edu/ml/machine-learning-databases/forest-fires/)

From the UCI Repository, I have chosen Forest Fire data set, column 8 to 10 containing data regarding temperature, RH, wind. The last column is the regression variable. A Polynomial model for degree 1 is fit and the Forest Fire values are predicted.