CS 613 - Machine Learning

Assignment 2 - Linear Regression
Fall 2016

Introduction

In this assignment you will perform linear regression on a dataset and using cross-validation to analyze your results. In addition to computing and applying the close-form solution, you will also implement from scratch a gradient descent algorithm for linear regression.

You may **not** use any function from the Matlab ML library in your code. And as always your code should work on any dataset that has the same general form as the provided one.

Grading

Part 1 (theory)	10pts
Part 2 (closed-form LR)	20pts
Part 3 (s-folds LR)	10pts
Part 4 (Local LR)	20pts
Part 5 (GD LR)	30pts
Report	10pts
TOTAL	100
Code doesn't generalize	-10pts

Table 1: Grading Rubric

Datasets

Fish Length Dataset (x06Simple.csv) This dataset consists of 44 rows of data each of the form:

1. Index
2. Age (days)
3. Temperature of Water (degrees Celsius)
4. Length of Fish

The first row of the data contains header information.

Data obtained from: http://people.sc.fsu.edu/ jburkardt/datasets/regression/regression.html

1 Theory

1. Consider the following data:

$$\begin{bmatrix} -2 & 1 \\ -5 & -4 \\ -3 & 1 \\ 0 & 3 \\ -8 & 11 \\ -2 & 5 \\ 1 & 0 \\ 5 & -1 \\ -1 & -3 \\ 6 & 1 \end{bmatrix}$$

   (a) Compute the coefficients for the linear regression using global least squares estimate (LSE) where the second value is the dependent variable (the value to be predicted). Show your work and remember to add a bias feature and to standardize the features (10pts).

2 Closed Form Linear Regression

Download the dataset *x06Simple.csv* from Blackboard. This dataset has header information in its first row and then all subsequent rows are in the format:

$$ROWId, x_{i,1}, x_{i,2}, y_i$$

Your code should work on any CSV data set that has the first column be header information, the first column be some integer index, then D columns of real-valued features, and then ending with a target value.

Write a script that:

1. Reads in the data, ignoring the first row (header) and first column (index).
2. Randomizes the data
3. Selects the first 2/3 (round up) of the data for training and the remaining for testing
4. Standardizes the data (except for the last column of course) using the training data
5. Computes the closed-form solution of linear regression
6. Applies the solution to the testing samples
7. Computes the *root mean squared error* (RMSE): $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$. where $\hat{y}_i$ is the predicted value for obseration x_i.

Implementation Details

1. Seed the random number generate with zero prior to randomizing the data
2. Don't forget to add in the offset feature!

In your report you will need:

1. The final model in the form $y = w_0 + w_1 x_{:,1} + ...$
2. The root mean squared error.

3 S-Folds Cross-Validation

Cross-Validation is a technique used to get reliable evaluation results when we don't have that much data (and it is therefore difficult to train and/or test a model reliably).

In this section you will divide your data up into 5 parts and train/test 5 different models using the 5-Folds Cross-Validation. We can then look at the root mean squared error using all the errors.

Write a script that:

1. Reads in the data, ignoring the first row (header) and first column (index).
2. Randomizes the data
3. Creates S folds (for our purposes $S = 5$, but make your code generalizable, that is it should work for any legal value of S)
4. For $i = 1$ to S
   (a) Select fold i as your testing data and the remaining $(S - 1)$ folds as your training data
   (b) Standardizes the data (except for the last column of course) based on the training data
   (c) Train a closed-form linear regression model
   (d) Compute the squared error for each sample in the current testing fold
5. Compute the RMSE using all the errors.

Implementation Details

1. Seed the random number generate with zero prior to randomizing the data
2. Don't forget to add in the offset feature!

In your report you will need:

1. The root mean squared error.

4 Locally-Weighted Linear Regression

Next we'll do locally-weighted closed-form linear regression.

Write a script to:

1. Read in the data, ignoring the first row (header) and first column (index).
2. Randomize the data
3. Select the first 2/3 (round up) of the data for training and the remaining for testing
4. Standardize the data (except for the last column of course) using the training data
5. Then for each *testing sample*
   (a) Compute the necessary distance matrices relative to the training data in order to compute a local model.
   (b) Evaluate the testing sample using the local model.
   (c) Compute the squared error of the testing sample.
6. Compute the *root mean squared error* (RMSE): $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$. where $\hat{y}_i$ is the predicted value for obseration x_i using the squared errors of the testing samples.

Implementation Details

1. Seed the random number generate with zero prior to randomizing the data
2. Don't forget to add in the offset feature!
3. Use the L1 distance when computing the distance matrices.
4. Let $k = 1$ in the similarity function $\beta(a, b) = e^{-d(a,b)/k^2}$.
5. Use *all* training instances when computing the local model.

In your report you will need:

1. The root mean squared error.

5 Gradient Descent

As discussed in class Gradient Descent (Ascent) is a general algorithm that allows us to converge on local minima (maxima) when a closed-form solution is not available or is not feasible to compute.

In this section you are going to implement a gradient descent algorithm to find the parameters for linear regression on the same data used for the previous sections. You may **NOT** use any function for a ML library to do this for you.

Implementation Details

1. Seed the random number generator prior to your algorithm.
2. Don't forget to add in the offset feature!
3. Initialize the parameters of W using random values in the range [-1, 1]
4. Do **batch** gradient descent
5. Terminate when absolute value of the percent change in the RMSE on the **training** data is less than the Matlab defined *eps* or after $1,000,000$ iterations have passed (whichever occurs first).
6. Use a learning rate $\alpha = 0.01$.
7. Make sure that you code can work for an arbitrary number of observations and an arbitrary number of features.

Write a script that:

1. Reads in the data, ignoring the first row (header) and first column (index).
2. Randomizes the data
3. Selects the first 2/3 (round up) of the data for training and the remaining for testing
4. Standardizes the data (except for the last column of course) base on the training data
5. While the termination criteria (mentioned above in the implementation details) hasn't been met
   (a) Compute the RMSE of the *training* data
   (b) While we can't let the testing set affect our training process, also compute the RMSE of the testing error at each iteration of the algorithm (it'll be interesting to see).
   (c) Update each parameter using *batch* gradient descent
6. Compute the RMSE of the testing data.

What you will need for your report

1. Final model
2. A graph of the RMSE if the *training* and *testing* sets as a function of the iteration
3. The final RMSE *testing* error.

Your graph should look similar to Figure 1.

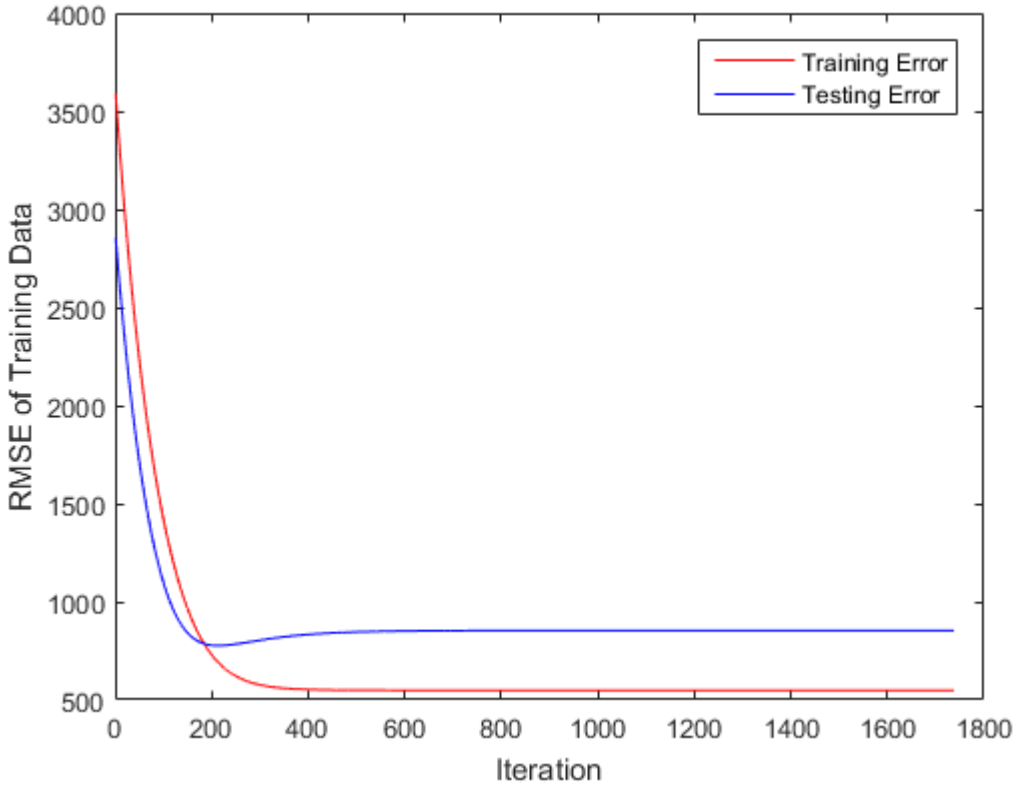

Figure 1: Gradient Descent Progress

Submission

For your submission, upload to Blackboard a single zip file containing:

1. PDF Writeup
2. Source Code
3. readme.txt file

The readme.txt file should contain information on how to run your code to reproduce results for each part of the assignment.

The PDF document should contain the following:

1. Part 1:
   (a) Your solutions to the theory question
2. Part 2:
   (a) Final Model
   (b) RMSE
3. Part 3:
   (a) RMSE
4. Part 4:
   (a) RMSE
5. Part 5:
   (a) Final Model
   (b) RMSE
   (c) Plot of RMSE for Training and Testing Data vs Gradient Descent iteration number