CS 613 - Machine Learning Assignment 1 - Regression

Avani Jain (14386342)

01/22/20

1 Theory

1. (10 pts) Consider the following data:

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

   (a) Compute the coefficients for the linear regression using least squares estimate (LSE), where the second value (column) is the dependent variable (the value to be predicted) and the first column is the sole feature. Show your work and remember to add a bias feature and to standardize the features. Compute this model using **all** of the data (don't worry about separating into training and testing sets).

   Solution:- We are using linear regression to find out the coefficients using LSE (Least Square Estimate). The final model equation can be written as :-

$$\hat{y} = 1.4 + -1.42639 * x_1$$

   (b) Confirm your coefficient and intercept term using the sklearn.linear_model LinearRegression function.

   Solution:- To confirm our answer we are using sklearn.linear_model LinearRegression function . The final model equation can be written as :-

$$\hat{y} = 1.4 + -1.42639 * x_1$$

2. For the function $g(x) = (x-1)^4$, where x is a single value (not a vector or matrix):

   (a) (3pts) What is the gradient with respect to x? Show your work to support your answer.

Solution:- The gradient with respect to x is the first order partial derivative of $g(x)$. We can find the first order partial derivative as follows:

$$g'(x) = \frac{\partial g(x)}{\partial x}$$
$$g'(x) = \frac{\partial (x-1)^4}{\partial x}$$
$$g'(x) = 4 * (x-1)^3$$

(b) (3pts) What is the global minima for $g(x)$? Show your work to support your answer.
Solution:- The global minima for $g(x)$ can be found as:-
We assume $g'(x) = 0$.

$$g'(x) = 0$$
$$\implies 4 * (x-1)^3 = 0$$
$$\implies (x-1)^3 = 0$$
$$\implies x = 1$$

The second partial derivative of $g(x)$ is :-

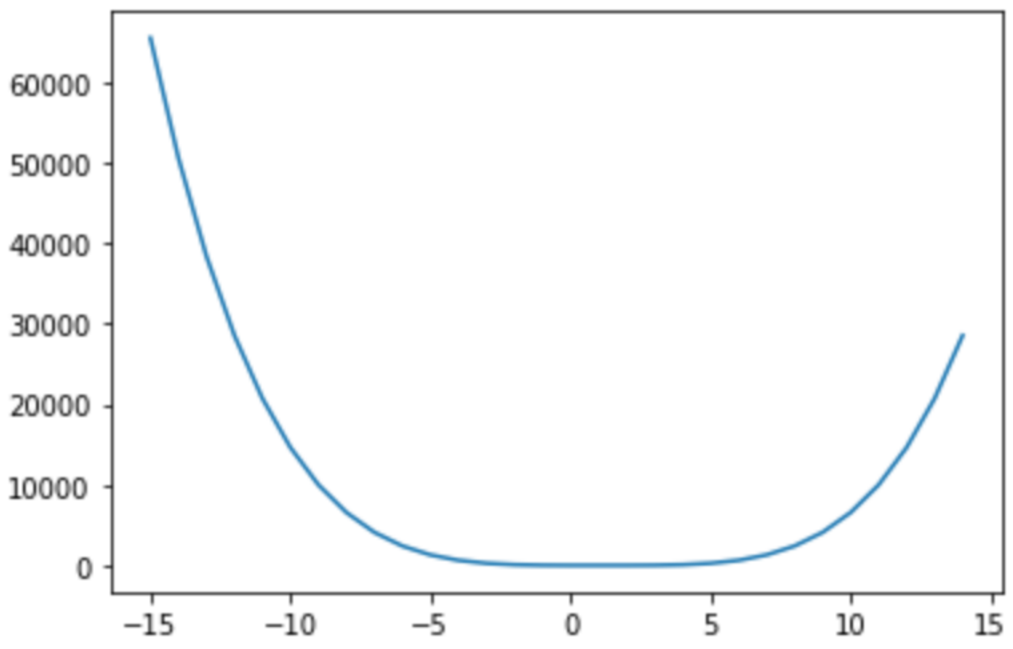
$$g''(x) = \frac{\partial g'(x)}{\partial x}$$
$$g''(x) = \frac{\partial 4(x-1)^3}{\partial x}$$
$$g''(x) = 12 * (x-1)^2$$

We are applying this function on the critical points:

$$g''(1) = 0$$

Hence, $x = 1$ is the gobal minima of $g(x)$.

(c) (3pts) Plot x vs $g(x)$ using matplotlib and use this image in your report.
Solution:
We plot $g(x)$ with respect to x where x values range from -15 to 15 and obtain the following graph.



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 observation X_i.

Solution:-
We use closed form linear regression to obtain the final model and RMSE value which are:-

$$Final\ model : \hat{y} = -674.1844 + 1086.2673 * x_1 + (-4812.1560) * x_2$$

$$RMSE = 703.6080$$

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 do S-Folds Cross-Validation for a few different values of S. For each run you will divide your data up into S parts (folds) and test S different models using S-Folds Cross-Validation and evaluate via root mean squared error. In addition, to observe the affect of system variance, we will repeat these experiments several times (shuffling the data each time prior to creating the folds). We will again be doing our experiment on the provided fish dataset. **You may use sklearn KFold** to perform this task.

Write a script that:

1. Reads in the data, ignoring the first row (header) and first column (index).
2. 20 times does the following:
   (a) Randomizes the data
   (b) Creates S folds.
   (c) For $i = 1$ to S
      i. Select fold i as your testing data and the remaining $(S - 1)$ folds as your training data
      ii. Standardizes the data (except for the last column of course) based on the training data
      iii. Train a closed-form linear regression model
      iv. Compute the squared error for each sample in the current testing fold
   (d) You should now have N squared errors. Compute the RMSE for these.
3. You should now have 20 RMSE values. Compute the mean and standard deviation of these. The former should give us a better "overall" mean, whereas the latter should give us feel for the variance of the models that were created.

Solution:-
We are obtaining the following value after using the cross validation technique:-

S	Mean	Standard Deviation
3	601.8675	101.7718
5	627.5888	111.9579
20	553.4061	286.4298
N	493.4732	385.3478

4 Locally-Weighted Linear Regression

Next we'll do locally-weighted closed-form linear regression. You may use **sklearn train_test_split** for this part.

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 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. 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 observation X_i.

   Solution:-
   We are using locally-weighted closed form linear regression to obtain the RMSE value:

$$RMSE = 406.0320$$

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, except **sklearn train_test_split** for the data.

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.

Solution:
We use gradient descent to converge on a local minima:

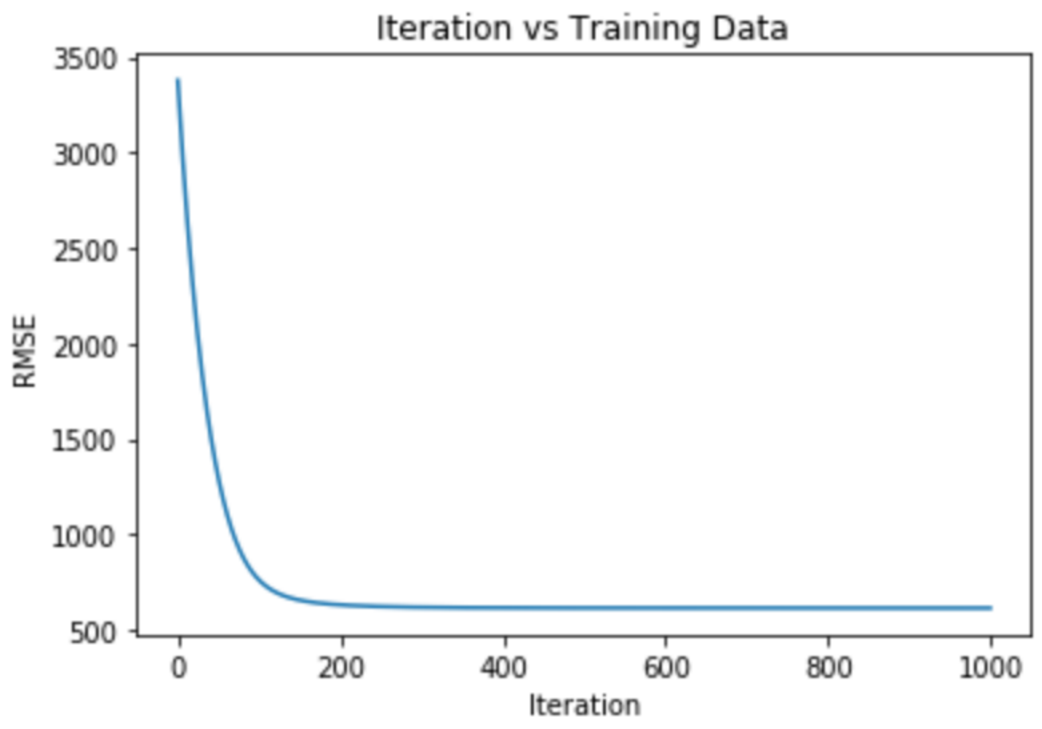
$$Final\ model : \hat{y} = 1632.9323 + 1178.4108 * x_1 + (-1114.0121) * x_2$$

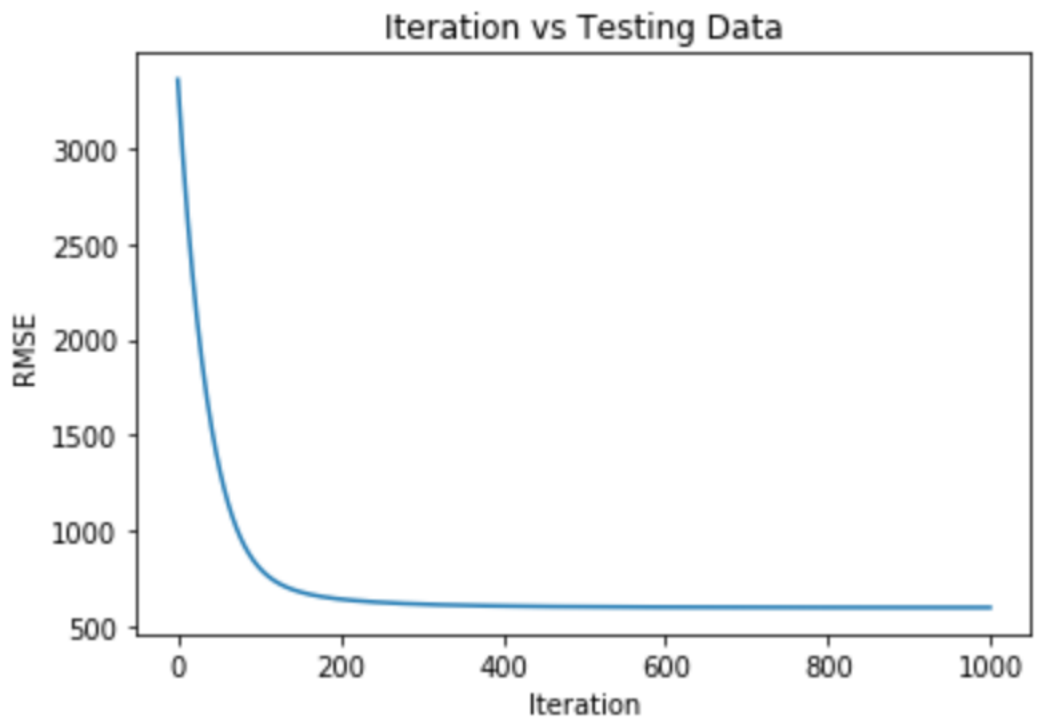
We observe that the value of RMSE converges at the following points:

$$RMSE\ of\ training\ data = 613.7909$$

$$RMSE\ of\ testing\ data = 597.3681$$

We plot the graph of training RMSE and testing RMSE as a function of the iterations and obtain the following output :


Iteration vs Training Data
RMSE
Iteration
3500
3000
2500
2000
1500
1000
500
0
200
400
600
800
1000


Iteration vs Testing Data
RMSE
Iteration
3000
2500
2000
1500
1000
500
0
200
400
600
800
1000