CS 383 - Machine Learning

Assignment 1 - Regression

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.

As with all homeworks, you cannot use any functions that are against the "spirit" of the assignment, unless explicitly told to do so. For this assignment that would mean any linear regression functions. You *may* use statistical and linear algebra functions to do things like:

- mean
- std
- cov
- inverse
- matrix multiplication
- transpose
- etc...

Grading

Part 1 (Theory)	19pts
Part 2 (Closed-form LR)	25pts
Part 3 (Local LR)	20pts
Part 4 (GD LR)	25pts
Report	11pts
TOTAL	100

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. (10pts) Consider the following supervised dataset:

$$X = \begin{bmatrix} -2 \\ -5 \\ -3 \\ 0 \\ -8 \\ -2 \\ 1 \\ 5 \\ -1 \\ 6 \end{bmatrix}, Y = \begin{bmatrix} 1 \\ -4 \\ 1 \\ 3 \\ 11 \\ 5 \\ 0 \\ -1 \\ -3 \\ 1 \end{bmatrix}$$

   (a) Compute the coefficients for the linear regression using least squares estimate (LSE). Show your work and remember to add a bias feature. You can use numpy.linalg.inv to compute the inverse. Compute this model using **all** of the data (don't worry about separating into training and testing sets).

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

2. For the function $J = (x_1 + x_2 - 2)^2$, where x_1 and x_2 are a single valued variables (not vectors):

   (a) What are the partial gradients, $\frac{\partial J}{\partial x_1}$ and $\frac{\partial J}{\partial x_2}$? Show work to support your answer. (4pts)

   (b) Create a 2D plot of x_1 vs J matplotlib, for fixed values of x_2 at 0,1, and 2. (4pts)

   (c) Based on your plots, what are the values of x_1 and x_2 that minimize J? (2pts)

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.

Implementation Details

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

In your report you will need:

1. The final model in the form $y = \theta_0 + \theta_1 x_{:,1} + ...$
2. The root mean squared error.

3 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.

Implementation Details

1. Seed the random number generate with zero prior to randomizing the data
2. Don't forget to add in the bias feature!
3. Use the L1 distance when computing the distances $d(a, b)$.
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.

4 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.

Implementation Details

1. Seed the random number generator prior to your algorithm.
2. Don't forget to a bias feature!
3. Initialize the parameters of θ 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 2^{-23}, or after $1,000$ iterations have passed (whichever occurs first).
6. Use a learning rate $\eta = 0.01$.
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.

Submission

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

1. PDF Writeup and PDF of Jupyter Notebook
2. Source Code Jupyter Notebook

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) Final Model
   (b) RMSE
   (c) Plot of RMSE for Training and Testing Data vs Gradient Descent iteration number