Assignment 1: Linear Regression using Gradient Descent

Spring 2020

Due Date: Tuesday Feb 11, 2020 Midnight

Instructions

- There are two parts to this assignment. The first part requires you to write code that uses gradient descent for linear regression. In the second part, you will use a ML library on the same dataset and compare your results.
- For the programming part, it's your responsibility to find the best set of parameters. Please include a README file detailing how to compile and run your program.
- All work submitted must be your own. Do not copy from online sources. If you use any references, please list them.
- You should use a cover sheet, which can be downloaded from: `http://www.utdallas.edu/~axn112530/cs6375/CS6375_CoverPage.docx`
- You are allowed to work in pairs i.e. a group of two students is allowed. Please write the names of the group members on the cover page.
- **You have a total of 4 free late days for the entire semester. You can use at most 2 days for any one assignment. After four days have been used up, there will be a penalty of 10% for each late day. The submission for this assignment will be closed 2 days after the due date.**
- Please ask all questions on Piazza, not via email.

1 Linear Regression using Gradient Descent (50 points)

1.1 Background

In this question, we will use gradient descent to perform linear regression analysis. Before we start coding, let's make sure we understand all the notation and vector format of equations:
We make a model for a single data point ($x^{(i)}$) as:

$$h^{(i)} = w_0 x_0^{(1)} + w_1 x_1^{(1)} + \cdots + w_n x_n^{(1)} + \tag{1}$$

We can write vector form of the above equation for all data points as:

$$H = XW \tag{2}$$

$$= \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \dots & x_m^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \dots & x_m^{(2)} \\ \dots & \dots & \dots & \dots \\ x_0^{(n)} & x_1^{(n)} & \dots & x_m^{(n)} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_m \end{bmatrix} \tag{3}$$

$$= \begin{bmatrix} h^{(1)} \\ h^{(2)} \\ \dots \\ h^{(n)} \end{bmatrix} \tag{4}$$

Here W is the weights vector X is matrix with one row per data point. The superscript denotes the data number and subscript denotes the attribute number. Error is defined as:

$$E = H - Y \tag{5}$$

$$= \begin{bmatrix} w_0 x_0^{(1)} + w_1 x_1^{(1)} + \cdots + w_m x_m^{(1)} \\ w_0 x_0^{(2)} + w_1 x_1^{(2)} + \cdots + w_m x_m^{(2)} \\ \dots \\ w_0 x_0^{(n)} + w_1 x_1^{(n)} + \cdots + w_m x_m^{(n)} \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(n)} \end{bmatrix} \tag{6}$$

$$= \begin{bmatrix} e^{(1)} \\ e^{(2)} \\ \dots \\ e^{(n)} \end{bmatrix} \tag{7}$$

Mean Squared Error is defined as:

$$\text{MSE} = \frac{1}{2n} \sum_{i=1}^{n} [e^{(1)}]^2 + [e^{(2)}]^2 + \cdots + [e^{(n)}]^2 \tag{8}$$

$$= \frac{1}{2n} E^T E \tag{9}$$

where n is the number of data points.
Gradient descent update for a single weight is defined as:

$$w_i^{\text{new}} = w_i^{\text{old}} - \alpha \frac{\partial(\text{MSE})}{\partial w_i} \tag{10}$$

MSE can be written as below. The variable j loops over all data points.

$$\text{MSE} = \frac{1}{2n} \sum_{j=1}^{n} (\sum_{i=0}^{m} w_i x_i - y_i)^2 \tag{11}$$

Its derivative wrt to weight w_i can be evaluated as below. Note that the variable j loops over all data points.

$$\frac{\partial(\text{MSE})}{\partial w_i} = \frac{1}{n} \sum_{j=1}^{n} (\sum_{i=0}^{m} w_i x_i - y_i) x_i \tag{12}$$

$$= \frac{1}{n} \sum_{j=1}^{n} e_i x_i \tag{13}$$

The term e_i is the error, as defined previously. Using vector definition, we can write the derivative for each point as:

$$\frac{\partial(\text{MSE})}{\partial W} = \frac{1}{n} \begin{pmatrix} \sum_{j=1}^{n} e_0 x_0 \\ \sum_{j=1}^{n} e_1 x_1 \\ \dots \\ \sum_{j=1}^{n} e_m x_m \end{pmatrix} \tag{14}$$

$$= \frac{1}{n} X^T E \tag{15}$$

The gradient descent weight update rule can be written in vector format as:

$$W^{\text{new}} = W^{\text{old}} - \frac{\alpha}{n} X^T E \tag{16}$$

1.2 Coding in Python

I have provided you a starter code in Python that uses the equations above. You will need to perform the following:

1. Choose a dataset suitable for regression from UCI ML Repository: - `https://archive.ics.uci.edu/ml/datasets.html`
2. Complete the code for pre-processing your dataset. Pre-processing includes the following activities:
   - Remove null or NA values

- Remove any redundant rows
- Convert categorical variables to numerical variables
- If you feel an attribute is not suitable or is not correlated with the outcome, you might want to get rid of it.
- Any other pre-processing that you may need to perform.

There is a **preProcess** method in the code. You need to complete it.

3. The starter code trains on the *train.csv* file and is tested on the *test.csv* dataset file. You need to change them to your dataset locations and values. You can create train/test datasets by splitting a dataset into 80:20 (or any other suitable ratio).
4. It is your responsibility to find the best value of *learning rate* and number of *epochs* parameters.
5. Output the training and testing mean squared error (MSE) for your dataset.
6. Answer this question: Are you satisfied that you have found the best solution. Explain.

2 Linear Regression using ML libraries (50 points)

In the second part of this assignment, you are free to use any ML library on the **same dataset that you used in part 1**.
Some suggested libraries are:

- Scikit Learn's linear regression or SGD linear regression package . Details are available at:
  `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html` or
  `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html`
- R's base package:
  `https://www.statmethods.net/stats/regression.html`
  or R's glm function: `https://www.statmethods.net/advstats/glm.html`
- Any other library that you find suitable. Make sure to list details in README file.

Some requirements are:

- You need to provide as many plots as possible. They could be MSE vs number of iterations, the output variable plotted against one or more of important attributes

- Parameters used such as number of iterations, learning rate, should be optimized
- Output as many evaluation statistics as possible. Some examples are weight coefficients, MSE, R^2 value, `https://en.wikipedia.org/wiki/Coefficient_of_determination`, Explained Variance `https://en.wikipedia.org/wiki/Explained_variation`, and any other
- Answer the following question: Are you satisfied that the package has found the best solution. How can you check. Explain.

What to Submit

- For part 1, completed Python code file.
- README file indicating details of your dataset, how to compile, answer to questions.etc.
- Do not hardcode paths on your local computer.
- If your dataset file is large, please post it online and read it from URL
- README file indicating which package you used, how to compile, answer to questions, etc
- Plots from part 2