CS 4375 Machine Learning

Assignment 1:
Linear Regression using Gradient Descent

Due Date: Mentioned in E-Learning

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 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

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.

Suppose there are n data points $X^{(1)}, X^{(2)}, \dots, X^{(n)}$, where each data point has m dimensions i.e. $X^{(i)} = \{x_0^{(i)}, x_1^{(i)}, \dots, x_m^{(i)}\}$. The *true value* of output for each data point is $y^{(1)}, y^{(2)}, \dots, y^{(n)}$.

We make a model for a single data point $X^{(i)}$ as:

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

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

$$\begin{align} H &= XW \tag{2} \\ &= \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \dots & x_m^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \dots & x_m^{(2)} \\ \dots & \dots & \dots & \dots \\ x_0^{(n)} & x_1^{(n)} & \dots & x_m^{(n)} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_m \end{bmatrix} \tag{3} \\ &= \begin{bmatrix} h^{(1)} \\ h^{(2)} \\ \dots \\ h^{(n)} \end{bmatrix} \tag{4} \end{align}$$

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

Error is defined as:

$$
\begin{aligned}
E &= H - Y && (5)\\
&= \begin{bmatrix} w_0x_0^{(1)} + w_1x_1^{(1)} + \cdots + w_mx_m^{(1)} \\ w_0x_0^{(2)} + w_1x_1^{(2)} + \cdots + w_mx_m^{(2)} \\ \cdots \\ w_0x_0^{(n)} + w_1x_1^{(n)} + \cdots + w_mx_m^{(n)} \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \cdots \\ y^{(n)} \end{bmatrix} && (6)\\
&= \begin{bmatrix} \sum_{i=0}^m w_ix_i^{(1)} \\ \sum_{i=0}^m w_ix_i^{(2)} \\ \cdots \\ \sum_{i=0}^m w_ix_i^{(n)} \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \cdots \\ y^{(n)} \end{bmatrix} && (7)\\
&= \begin{bmatrix} e^{(1)} \\ e^{(2)} \\ \cdots \\ e^{(n)} \end{bmatrix} && (8)
\end{aligned}
$$

Mean Squared Error is defined as:

$$
\begin{aligned}
\text{MSE} &= \frac{1}{2n}(\ [e^{(1)}]^2 + [e^{(2)}]^2 + \cdots + [e^{(n)}]^2\ ) && (9)\\
&= \frac{1}{2n}E^TE && (10)
\end{aligned}
$$

where n is the number of data points.

After random initialization, gradient descent update rule for any weight is defined as:

$$
w_i^{\text{new}} = w_i^{\text{old}} - \alpha\frac{\partial(\text{MSE})}{\partial w_i} \qquad (11)
$$

where α is the learning rate.

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

$$
\begin{aligned}
\text{MSE} &= \frac{1}{2n}\sum_{j=1}^{n}(h^{(j)} - y^{(j)})^2 && (12)\\
&= \frac{1}{2n}\sum_{j=1}^{n}(\sum_{i=0}^{m} w_ix_i^{(j)} - y^{(j)})^2 && (13)
\end{aligned}
$$

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

$$
\begin{aligned}
\frac{\partial(\text{MSE})}{\partial w_i} &= \frac{1}{n}\sum_{j=1}^{n}(h^{(j)} - y^{(j)})x_i^{(j)} && (14)\\
&= \frac{1}{n}\sum_{j=1}^{n}e^{(j)}x_i^{(j)} && (15)
\end{aligned}
$$

The term $e^{(j)}$ is the error for the j^{th} point, as defined previously. We can then use equation (11) to find the weight update rule.

1.2 Coding in Python (75 points)

For this part, you will write your own code in Python for implementing the gradient descent algorithm, and apply it to a linear regression problem. You are free to use any data loading, pre-processing, and graphing library, such as numpy, pandas, graphics. **However, you cannot use any library that implements gradient descent or linear regression.**

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.php`. If the above link doesn't work, you can go to the main page:
   `https://archive.ics.uci.edu/ml/index.php` and choose "view all datasets option". Host the dataset on a public location e.g. UTD web account. Please do not hard code paths to your local computer.
   Please note you can not use any toy dataset, for example the ones that come with scikit-learn

2. Pre-process 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.

3. After pre-processing split the dataset into training and test parts. It is up to you to choose the train/test ratio, but commonly used values are 80/20, 90/10, etc.

4. Use the training dataset to construct a linear regression model using the equations in the previous part and develop a model. **Note again: you cannot use a library that implements gradient descent or linear regression.**

   There are various parameters such as *learning rate*, *number of iterations* or other *stopping condition*, etc. You need to tune these parameters to achieve the optimum error value. Tuning involves testing various combinations, and then using the best one. You need to create a log file that indicates parameters used and error (MSE) value obtained for various trials.

5. Apply the model you created in the previous step to the test part of the dataset. Report the test dataset error values for the best set of parameters obtained from previous part. If you are not satisfied with your answer, you can repeat the training step.

6. Answer this question: Are you satisfied that you have found the best solution? Explain.

2 Linear Regression using ML libraries (25 points)

In the second part of this assignment, you will use any ML library that performs linear regression from Scikit Learn package. `https://scikit-learn.org` on the **same dataset that you used in part 1**.

Use similar workflow like in part 1. The difference would be that the package will build the model for you. Report output similar to earlier part.

3 Additional Requirements

- You will be judged on the basis of your code. Write clean and elegant code that should be in the form of Python classes, with appropriate constructors, and other methods.
- Parameters used such as number of iterations, learning rate, should be optimized. Keep a log file of your trials indicating parameters used, training error, and test error.
- 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. Use your judgement and be creative.
- Output as many evaluation statistics as possble. 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
- Be sure to answer the following question: Are you satisfied that the package has found the best solution. How can you check. Explain.

4 What to Submit

- For parts 1 and 2, completed Python code file(s). Remember to properly name your files such as part1.py and part2.py.

- README file indicating how to build and run your code. For part 2, indicate which libraries you have used. If the TA cannot run your code, you don't get any credit.
- **Please do not submit any dataset files.** Host your data on a public web source, such as your UTD web account, AWS, etc.
- Do not hardcode paths on your local computer.
- A report file containing log of your trials with different parameters, answer to questions, and plots.