Assignment 1: Enhanced 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 front page of each submitted file.
- **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 Regression using Enhanced Gradient Descent

1.1 Background

In class, we studied the basic ("vanilla") form of gradient descent applied to a linear regression problem. There have been various enhancements proposed to the basic algorithm over the years to obtain better performance.

You can read about some of the enhancement at the following links:

- `https://towardsdatascience.com/improving-vanilla-gradient-descent-f9d91031ab1d`
- `https://towardsdatascience.com/10-gradient-descent-optimisation-algorithms-86989510b5e9`
- `https://hackernoon.com/demystifying-different-variants-of-gradient-descent-optimization-algorithm-19ae9ba2e9bc`

For this assignment, you will use gradient descent with at least one enhancement chosen from above links. **Note that you cannot use stochastic or mini-batch versions of gradient descent.** These are very similar to the basic algorithm.

Before starting to code, make sure you understand the enhanced algorithm. Remember to write the equations for error function, learning rate, weight update.

1.2 Coding in Python (75 points)

For this part, you will write your own code in Python for implementing the enhanced gradient descent algorithm that you selected, 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, its variations or linear regression.**

You will need to perform the following:

1. Decide on the enhancement you will use on the basic gradient descent.
2. 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.
3. 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.

4. 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.

5. 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.

6. 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.

7. 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 select a regression library from Scikit Learn linear regression package: `https://scikit-learn.org/stable/modules/linear_model.html`. You need to find a library that is similar to the algorithm you chose in part 1. If you cannot find any library that is exactly similar, you can choose one that is similar. Also, you need to use 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

- Submit a report indicating details of the enhanced gradient descent algorithm that you chose. Please provide equations with meaning of each symbol used.
- 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

- Brief report containing details of your algorithm.
- 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 separate file containing log of your trials with different parameters, and plots. It should also answer the question whether you have found the global best solution in both parts.