Assignment 1
Linear Regression

Due Date: 31st March, 2022

Guidelines

- Submit all of your code and results in a single zip file with name FirstName_RollNumber_04.zip
- Submit single zip file containing a) code (tas01.ipynb, tas) b) report
- There should be a Report.pdf detailing your experience and highlighting any interesting result.
- Email instructor or TAs if there are any questions.

Overview

In this assignment, you will implement linear regression and get to see it work on data. Linear regression is a linear approach to model the relation b/w a dependent variable and one or more independent variables. Mathematically,

$$\bar{y} = mX + c$$

In the above equation X is an independent variable and Y is dependent, m is slope and c is the y-intercept. The loss is the error in our predicted values of m and c. You will be using Mean Squared error to calculate the loss between true and predicted values.

$$E = \frac{1}{n} \sum_{i=0}^{n} (y_i - \bar{y}_i)^2$$

Here y_i is the actual value and $\bar{y}_i$ is the predicted value and n is the number of training examples. Next step is to find optimal values of m and c using the gradient descent algorithm. Gradient descent algorithm can be broken down to following steps:

- Randomly initialize the values of m and c
- Set a value of learning rate (L), epochs.
- For each epoch:
    - Find $\bar{y}$ and then compute derivative of E w.r.t to m (dE/dm) and c (dE/dc).
    - Update the values of m and c using following formulae:

      m= m - L (dE/dm)

      c= c - L(dE/dc)
    - Repeat until epoch < epochs.

You will be using a subset of house price prediction dataset available on kaggle. **For now, we would encourage you to implement linear regression from scratch using numpy. Please refrain from using tools like scikit-learn.**

Task 01: Train/Validation/Test Split (5 Marks)

- Perform train/validation/test (70%/15%/15%) random split on both task01.csv and task02.csv separately. (You are not allowed to use sklearn train_test_split function for this task)

Task 02: Linear Regression with One Variable (25 Marks)

In this task, you will implement linear regression with one variable to predict the apartment price. Download the file **task01.csv**. This file contains the dataset for task 1. The csv file contains two columns Area(sq_ft) and price and has 1460 entries. Area(sq_ft) is your independent variable and price is dependent variable.

Step 1: Read the data files using pandas.
Step 2: Randomly initialize the m and c. Set the learning rate to 0.001 and set the number of epochs to 100 (You are encouraged to experiment with a different set of learning rate and number of epochs).
Step 3: Write a function *loss_func(y_ture, y_pred)* that takes predicted apartment price and original apartment price as input and returns the mean squared error b/w the true and predicted price.
Step 4: Write a function grad_descent(learning_rate, epochs, m, c) that performs the gradient descent to reach optimal values of m and c for the given dataset. The algorithm is explained in the overview section. The function should return final values of m, c and an array containing the loss values at each epoch for both training and validation.
Step 5: Plot the epoch vs training and validation loss using matplotlib.
Step 6: Plot the training data with linear regression fit and find test loss.

Task 03: Linear Regression with Multiple Variables (60 Marks)

In this task, you will implement linear regression with multiple variables to predict the apartment price. Download the file **task02.csv**. This file contains the dataset for task 2. The csv file contains five columns Area(sq_ft), Bedrooms, Kitchen, YearBuilt and price and has 1460 entries. Area(sq_ft), Bedrooms, Kitchen, YearBuilt are your independent variables and price is dependent variable. Zero value in the Bedroom and kitchen column indicates that there is no dedicated area for them (example studio apartment). **For this task you need to report results for both with and without feature normalization.** Normalization is done by subtracting the mean and dividing by standard deviation.

Step 1: Read the data file using pandas
Step 2: Randomly initialize the m (Hint: for each feature/variable you will have to compute separate m) and c. Set the learning rate to 0.001 and set the number of epochs to 100 (You are encouraged to experiment with a different set of learning rate and number of epochs).

Step 3: Write a function *loss_func(y_ture, y_pred)* that takes predicted apartment price and original apartment price as input and returns the mean squared error b/w the true and predicted price.

Step 4: Write a function grad_descent(learning_rate, epochs, m, c) that performs the gradient descent to reach optimal values of m and c for the given dataset. The algorithm is explained in the overview section. The function should return final values of m, c and an array containing the loss values at each epoch.

Step 5: Plot the epoch vs training and validation loss values using matplotlib and find test loss.

For task 2 and 3: If you are getting NaN values for m and c on the given parameters try decreasing the learning rate or number of epochs. Another possible option could be to remove outliers from data (you can check z-score, Inter quartile range (IQR) outlier removal method or simply decide outliers by looking at scatter plot). Remember best practice is to remove outliers before train/test/val split.

Report (10 Marks)

Task 01:

- Briefly describe the outlier removal technique you used (if any)

Task 02:

- Describe any important decision you made and any interesting insights you gained.
- Report the parameters which gave the minimum loss along with values of slope, intercept, training loss, validation loss and test loss.
- Comment on training and validation loss curves.

Task 03:

- Describe any important decision you made and any interesting insights you gained.
- Report the parameters which gave the minimum loss along with values of slope, intercept, training loss, validation loss and test loss.
- Comment on training and validation loss curves.
- Which method gave you the best results with normalization or without normalization? And why? Briefly explain.