FML - CS 725 - 2019 Assignment 1

(TAs - Ishan, Rishabh)

Important Instructions for the assignment

1. The assignment is to be completed individually.
2. This assignment is entirely programming-based and has 2 components. The first part is implementing LinearRegression class and various loss functions (along with their gradient functions). The base code is given and you are required to fill the unimplemented functions. The second part of the assignment is to use the above part to perform well on the Kaggle leaderboard. You are given train.csv (xtrain, ytrain) and test.csv (only xtest). You are required to find ytest (predicted) and upload it on kaggle for scoring.
3. For instructions on how to access Kaggle.
    a. Go to the Kaggle site and make a new account.
    b. **Make sure your display name is your ROLL NUMBER (ie 193050000). This is very important as we'll check your submission using your roll number. No other username will be evaluated**
4. The files related to this assignment are present in **assignment1.tar.gz** . Download and extract this to get started. The directory is organised as follows

```
assignment1/
     |- main.py (all the functions that you need to write)
     |- train.csv
     |- test.csv
```

5. **Submission Details**
    a. You have to submit the following files **main.py, comparison.jpg, prediction.csv**
    b. Create a directory with the name <YourRollNumber>-assignment1 and put the above files in it. Use this command to compress the directory

        tar -czvf <YourRollNumber>-assignment1.tar.gz path/to/directory

    c. On running main.py your program should print out the predicted count for xtest on terminal. This prediction must match the prediction.csv that you have given. Also this should be the same csv which you upload on kaggle. (We will verify that the result on kaggle is reproducible by your code)
6. **Queries** - Incase of any queries regarding the assignment either email the TAs at cs725_iitb@googlegroups.com or create a new discussions on **Moodle** or **Kaggle**.
7. The entire assignment is in python. You are encouraged to use **numpy** for faster matrix operations. For generating the graph use **matplotlib**
8. Your program shouldn't take more than 10 minutes to print out the predictions on xtest.

In this assignment, you will perform **Linear Regression** by implementing **Different Regression Loss Functions** (like MSE, MAE, etc) using the method of gradient descent.
For the concept, refer to the class notes/slides.

Dataset (Bike-rental prediction)

These days the whole process of renting a bike is automated, anyone can rent a bike at a location and return it to some other location. The companies which are in bike-rental business are interested in knowing the number of bikes that will be rented given the date, time of day, weather conditions, temperature, humidity etc. Your task is to find the best linear regression model that can predict with good accuracy the number of bike rental count given the conditions.

Bike-rental Dataset
(http://bigpicture.net/article/highway-85-creative-prints-everything-motorcycle)

There are 13 columns in the dataset. The last column (“count”) is the value which you have to predict. The columns are-

1. instance (id) - id of the row of dataset
2. date - in format DD/MM/YYYY
3. season - discrete (1 - spring, 2 - summer, 3 - fall, 4 - winter)
4. hour - discrete (0-23)
5. holiday - discrete (1 - holiday, 0 - not a holiday)
6. weekday - day of the week in string
7. working day - discrete (1 - neither a holiday or weekend, 0 - otherwise)
8. weather situation - discrete (1-clear,2-mist/clouds,3-light snow/rain,4-thunderstorm/heavy rain)
9. temperature - normalised temp in °C. calculated as (t - t_min)/(t_max - t_min) (t_min = -8, t_max = 39)
10. atemperature - normalised feeling temp in °C. (t - t_min)/(t_max - t_min) (t_min = -16, t_max = 50)
11. humidity - float
12. wind speed - float
13. count - integer (This is target variable)

The train file consist **13865** rows of **13** columns. The test set consists of **3514** rows with **12** columns (the features). The last column is held out for checking.

The assignment has 2 parts

(a) **[20 Marks]** Implement the class LinearRegressor. There are 3 functions to implement **__init__**, **train** and **predict**. The primary loss function used for linear regression is mean squared error. So you are required to implement **mean_squared_loss** and **mean_squared_gradient**. We have provided the basic function to read data from csv. You need to implement **preprocess_dataset** which will process the dataset so that it can be used for training. (ie converting strings to floats or int, converting categorical variables to suitable value, or fancy things such as feature scaling etc)

(b) **[20 + 5 Marks]** Experimenting with different loss functions. In this part we ask you to implement the following 3 loss functions along with their gradient functions

```
mean_absolute_loss
mean_absolute_gradient
mean_log_cosh_loss
mean_log_cosh_gradient
root_mean_squared_loss
root_mean_squared_gradient
```

To know what these loss functions are please refer to these
https://en.wikipedia.org/wiki/Mean_squared_error
https://en.wikipedia.org/wiki/Mean_absolute_error
https://memoex.github.io/note/tech/ml/loss/ (RMSE and logcosh loss given)

After implementing the above losses you are required a plot a graph showing different losses as a function of epoch. Do the following -
Train your model using the 4 different gradient functions to update model weights. After each epoch find the mean_squared_loss of the model on the complete training dataset. Plot this mean_squared_loss as a function of epoch for each gradient function in the same graph (use legends in matplotlib). For the above task you have keep the learning rate same across all 4 gradient update functions. Choose a suitable learning rate so that none of the loss function diverges. Choose number of epochs such that we can see the losses converging. Note that even though you use different gradient function, we calculate the same loss (mean_squared_error) per epoch. This is done because these losses would differ in their absolute magnitude and hence not suitable for comparison). Save this graph with name **comparison.jpg**

(c) **[5 Marks]** For the final part you are required to tune the parameters (lr, epochs etc) to build the best classifier and perform well on the kaggle leaderboard. You can play around with **preprocess_dataset** to scale/modify/normalize features according to your wish. For example if you have a categorical variable which takes three values {“Red”,”Green”,”Blue”} you can either convert it into ordinal variables giving values {Red = 0, Green = 1, Blue = 2} or you could add more columns

and one-hot encode the categorical variables e.g Red = (1,0,0), Green = (0,1,0) and Blue = (0,0,1). You are allowed to drop or add your own features as well. Kaggle will calculate **mean squared error loss** between your predicted y and true y on test dataset. Submit main.py with the appropriate hyperparameters so that on running

```
python3 main.py --train_data train.csv --test_data test.csv
```

Your model prints out the predicted output on test data. Also make sure this runs within 10 minutes. We will cross check that your model is able to reproduce your kaggle result. The final score for this part will be dependent on your standing in the leaderboard.

May the leaderboard be with you !