IIT BOMBAY

Lab Assignment #1
Points: 5

Main TAs: *P S V N Bhavani Shankar, Dhiraj Sah, Saurabh Kumar*

Course: *CS 335* – Instructor: *Preethi Jyothi*
Due date: *August 7, 2023*

General Instructions

- Download the file lab1.tgz from Moodle for CS 335, and extract the file. You will see three directories, one for each question, with the necessary files within each directory.
- For your final submission, create a final submission file [rollnumber].tgz with the following internal directory structure:

```
lab1/
  |
  +-q1/
  |  |
  |  +- q1.py
  |  +- histograms/
  +-q2/
  |  +- q2.py
```

  where `q1.py` and `q2.py` have the appropriate code filled in. Compress your submission directory using the command: `tar -cvzf [rollnumber].tgz rollnumber` and upload [rollnumber].tgz to Moodle. This lab submission is due on or before **11.59 pm on Aug 7, 2023**.
- You will get the full 5 points if the code you've written in `q1.py` and `q2.py` successfully passes through the autograder. Small errors will not be penalized, since it is the first lab.

Question 1

Say you toss 100 fair coins simultaneously and want to programmatically record the total number of heads. One simultaneous toss of the 100 coins counts as a single trial. `q1/q1.py` contains a list `num_trials_list` of varying numbers of trials. For every value in this list, we want to count the number of heads and plot a histogram. The histogram should take the shape of a binomial distribution and get more accurate with larger numbers of trials.

`q1.py` has two functions:

1. `toss(num_trials)`: Takes the number of times the experiment is to be performed as an argument and returns a numpy[a] array of the same size with the number of heads obtained in each trial.

2. `plot_hist(trial)`:- Takes an array of the number of heads obtained for k trials from `toss(`k`)` and plots a histogram based on these counts. Generate plots for each value in `trials_list` and save these plots in a directory called `histograms`.

Complete the following tasks:

1. Complete the function definitions in `q1.py`.

2. Create a directory named `histograms` inside the directory `q1/`.

3. Save the histograms obtained by `plot_hist()` in `histograms` with names formatted as `hist_<num_trials>.png`. For example, for `num_trials` = 1000, the filename would be `hist_1000.png"`.

Notes:

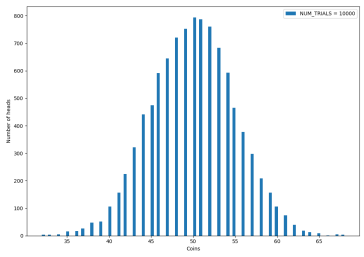
1. Use `plt.savefig()` function to save the histograms.

2. Use the given template only. **DO NOT** change the names of the given functions as it will cause the autograder to fail.

3. Use `for` loops to simulate the 100 coin tosses for a given `num_trials` value. Do not use predefined functions to calculate the numpy array in *''toss()"* function else marks will be deducted.

Here is an example histogram for **num_trials** = 10000.

[a] `https://numpy.org/doc/stable/user/index.html#user`



Question 2

The dataset in the CSV file `q2_dataset.csv` contains 1024 instances, where each instance has 10 features and a binary 0/1 label associated with it. Complete the following tasks:

1. Sequentially group the 1024 instances into smaller batches of 16 instances each and compute the mean instance for each batch.
2. Compute the L2 (Euclidean) distance of each instance with every other instance within a batch.
3. Return the label of the most similar instance (barring itself) for each instance in each batch.

Explanations of the functions to be completed within `q2.py` are detailed below:

- `compute_batch_means`: Takes the instances and batch size as input, and outputs the mean input vector for each batch. (Ignore the label when computing the mean vector.) You should implement this function using only matrix/vector operations in numpy, and without the use of `for` loops. Make sure your implementation does not shuffle the instances or use any form of randomness.
- `compute_l2_distance`: Takes a batch of instances as input, and outputs a distance matrix where the entry (i, j) represents the L2 distance between the i-th and j-th instance in the batch. Try to implement this function using `for` loops. We will guide you to implement this function using numpy operations without `for` loops. Make sure your implementation does not shuffle the instances or use any form of randomness.
  - `compute_l2_distance` should contain code with `for` loops.
  - `compute_l2_distance_numpy` computes the same using numpy operations on matrices/vectors.
- `compute_most_similar`: Takes the instances, labels, and batch size as input, and output the label of the most similar instance (based on `compute_l2_distance`) for each instance in each batch. Check that replacing `compute_l2_distance` with `compute_l2_distance_numpy` gives the same outputs. Be sure to exclude the instance itself when computing the most similar instance.

Notes:

1. Complete the definitions of `compute_batch_means`, `compute_l2_distance` and `compute_most_similar`. `compute_l2_distance_numpy` can be done at the end.
2. The time taken by `compute_most_similar` is printed as one of the outputs to reiterate the importance of using matrix operations when dealing with large amounts of data.
3. The template already includes a function to load the dataset from a CSV file. Do not change the names of the functions or their arguments in the template, as this will cause the autograder to fail.
4. Batch size is fixed to **16**; do not change that.

Question 3

This question aims to familiarize you with Kaggle competitions for upcoming labs. For this question, you do not need to submit any files and only need to ensure that your name appears on the Kaggle leaderboard.

Important Todo for Q3

Please sign up on Kaggle using your IITB LDAP email ID, with your Kaggle "Display Name" set to <your_roll_number>.

You are given a training dataset `q3_train_data.csv` containing 5058 instances and a test file `hidden_test.csv` containing 560 instances. Run the code in `q3.py` to train a logistic regression classifier, and tune hyperparameters to improve performance on the test data.

Logistic Regression is a binary classification algorithm. Unlike linear regression that predicts continuous values, logistic regression predicts the probability of an input belonging to one of two classes (usually represented as 0 or 1). It uses the logistic (sigmoid) function to transform the linear output into a probability value between 0 and 1.

The file `q3.py` contains the entire ML pipeline for fitting a logistic regression classifier on the given dataset. Please complete the following steps:

1. **Hyperparameter tuning**: In `perform_grid_search`, set appropriate values for the given hyperparameters in *param_grid*[a] . This will be the set from which the best hyperparameters get selected for the model.

2. **Plot training Loss**: Run `plot_and_save_training_loss` to create a set of plots based on hyperparameters generated by *grid_search*.

3. **Test predictions**: Use the already implemented functions to get predictions for test instances in `hidden_test.csv` snf create a two-column csv file with the column headers **ID** & **Bankrupt?**. Submit this file on Kaggle at the following link and confirm that your roll number appears on the leaderboard.

Note:

1. Be sure to submit the output CSV file on Kaggle for this question to be considered complete.

[a]Descriptions of hyperparameters are in `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`.