IIT BOMBAY

Lab Assignment #2
Points: 5

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

Course: *CS 335* – Instructor: *Preethi Jyothi*

Due date: *August 14, 2023*

General Instructions

- Download lab2.tgz from Moodle for CS 335, and extract the file to get `lab2`. The data files are also up on Kaggle's "Data" tab.
- The folder `q1_q3/` has template code for both questions Q1 and Q3. You should solve Q1 first, and Q3 can be implemented after.
- The folder `q2/` has template code for Q2.
- The folder `splits/` contains all the CSV files required for this task. Make sure you update the path files in the template code to load the CSV files properly.
- For your final submission, create [rollnumber].tgz with the following internal directory structure:

```
[rollnumber]/
  |
  +-q1\_q3/
  |  |
  |  +- template.py
  +-q2/
  |  +- template.py
  |  +- figures/
```

  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 14, 2023**.

- You will get 3 points if your closed form solution exactly matches your solution using gradient descent. You will get an additional 2 points for Q3 with a basis function that improves over the solution in Q1. Your roll number **HAS TO** appear on the Kaggle leaderboard to get full points for this lab assignment. You can get up to 3 extra credit points if you top the Kaggle private leaderboard. More details appear at the end.

(Unregularized) Linear Regression

This lab will familiarize you with training and evaluating linear regression predictors. For this task, your linear regression model should predict the release year of a song from a set of **90 timbre-based audio features** extracted from the song. The release year ranges between 1922 and 2011. Training and development sets are in `train_data.csv` and `dev_data.csv`, respectively. `test_data.csv` and `test_labels.csv` contain the test features and test labels, respectively. You need to generate predictions for the unseen test instances in `hidden_test_data.csv` and upload on Kaggle.

Q1: Closed Form Solution

For a training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i \in \mathbb{R}^{(d+1)}$ represented by a feature matrix $\mathbf{X}$ and a label vector $\mathbf{y}$, the least squares solution $\mathbf{w}^*$ can be computed by:

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

where

$$\mathbf{X} = \begin{bmatrix} \longleftarrow & \mathbf{x}_1{}^\top & \longrightarrow \\ \longleftarrow & \mathbf{x}_2{}^\top & \longrightarrow \\ & \vdots & \\ \longleftarrow & \mathbf{x}_n{}^\top & \longrightarrow \end{bmatrix}_{n \times (d+1)}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n \times 1}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}_{(d+1) \times 1}$$

1. Datasets are loaded using the `load_data` function.
2. Features and target values are extracted using the `prepare_data` function.

Implement the closed-form solution for linear regression within `train_model` in `q1_q3/template.py`. Use the learned weights to predict the target values in the `predict` function. Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are calculated using the `calculate_errors` function. Code to tune your model using the development set (`dev_data.csv`) and the test set (`test_data.csv`), and code to print the corresponding MSE and RMSE values are provided in the template code.

Q2: Gradient Descent

Find the least squares solution using *Batch Gradient Descent*. `batch_size` indicates the number of data points in a batch. If `batch_size` is `None`, this function implements **Gradient Descent** and computes the gradient over all training examples. The code for loading datasets, computing batches, evaluating on the `test_set`, and saving the predictions on `hidden_test_data` are already implemented in `q2.py` (and accompanied with descriptions).

Complete the following functions:

1. `fit()`: There are two loops inside this function. You have to complete the inner `for` loop and use `compute_gradient()` to calculate the gradient of the loss w.r.t. the weights. Next, you must calculate the "validation loss" using `compute_rmse_loss()` and store these losses across training epochs in `error_list`.

2. `compute_gradient()`: This function should return the gradient of the loss w.r.t weights of the model. (Normalize the values before returning, or it may cause gradients to explode.)

3. `compute_rmse_loss()`: This function should return the *Root Mean Square Error* loss ($\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2}$) between the target labels and predicted labels.

4. `predict()`: This function should return the predicted values of the model on the given set of feature values passed as an argument to it.

5. `plot_loss()`: This function is used to plot the losses stored in `error_list` of the model.

6. You have to make a folder named `figures` in the present working directory to save the plots generated by `plot_loss()`. Names should be in the format `loss_<batch_size>.png`.

Note:

1. Make sure that you get roughly the same test RMSE loss using the weights learned by the closed form solution in Q1 and gradient descent in Q2. This will be part of the autograder check. **3/5 points** is for passing this check.

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

3. You can tweak the values of batch size when training the model.

Q3: Basis Functions

We will use basis functions to try and improve the model's fit from Q1. For a datapoint $\mathbf{x}_i$, we can transform it using a Radial Basis Function (RBF) as follows:

$$\Phi(\mathbf{x}_i) = \exp\left\{-\frac{(\mathbf{x}_i - \boldsymbol{\mu}_i)^2}{2\sigma^2}\right\}$$

where exp is an elementwise operation and the dimensionality of $\Phi(\mathbf{x}_i)$ is the same as $\mathbf{x}_i$. The RBF transformation can be concatenated to the original features, so that the transformed input features $\hat{\mathbf{x}}_i = [\mathbf{x}_i \quad \Phi(\mathbf{x}_i)]$.

Complete the following:

1. There are two sections in `q1_q3/template.py` that you have to uncomment. One transforms the features as explained above, and the other saves a csv file with predictions on `hidden_test_set` that you can submit on **Kaggle**.
2. Compute `mu` ($\boldsymbol{\mu}$) as the mean of training features and set `s` (σ) to 1.
3. `transform_features(X,mu,s)`: This function takes the design matrix `X` and parameters `mu` (i.e μ) and `s` (i.e σ) and returns the concatenated features `X_tf`.

NOTE

1. You can tune the value of σ to improve the quality of your fit.
2. The test RMSE of your regression model with basis functions should be less than the test RMSE using the closed form solution in Q1. The autograder will check for this and you will get the remaining **2/5 points** for passing this check.

❊ Extra Credit: Climb the Leaderboard on Kaggle

The objective for this last part is to build the best possible linear regression model using any enhancements you like. Submit your target predictions for the instances in `hidden_test_data.csv` to Kaggle so that your roll number appears on both the "Public Leaderboard" (and eventually the "Private Leaderboard" after the assignment concludes). Top-scoring performers on the "Private Leaderboard" (with a suitable threshold determined after the deadline passes) will be awarded up to 3 extra credit points. The exact breakdown of the three points for this question will be announced later this week.