Lab Assignment #4
Points: 5

Main TAs: *Tejpalsingh Baljeetsingh Siledar, Vishal Tapase, Sanjeev Kumar*

Course: *CS 335* – Instructor: *Preethi Jyothi*
Due date: *August 28, 2023*

General Instructions

- Download lab4.tgz (URL:`https://drive.google.com/file/d/1yo1jLBm7CHkqZvL1tiRxTHe9bm648y5G/view?usp=sharing`) from Moodle for CS 335, and extract the file to get `lab4`.
- The data files are in `task1_data.csv`, `task2_data.csv` and `task3_data.csv` for Q1, and in `train_data.csv`, `test_data.csv` (with labels in `test_labels.csv`) and `hidden_test_data.csv` for Q2. 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/
  |  |
  |  +- lab4_q1_template.py
  |  +- plot_1.png
  |  +- plot_2.png
  |  +- plot_3_lambda0.png
  |  +- plot_3_lambda1.png
  |  +- plot_3_lambda10.png
  +-Q2/
  |  +- lab4_q2_template.py
  |  +- hidden_test_out.csv
```

- You will get 3 points if you share the correct plots for Q1. You will get 2 points if your predictions for the instances in `test_data.csv` using your code in `lab4_q2_template.py` matches or outperforms the baseline accuracy (i.e. 74.39%). NOTE: 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.

Q1: Simple Logistic Regression and Decision Boundaries

This problem involves implementing logistic regression (LR) classifiers (with and without L2-regularization) for three toy tasks by maximizing conditional likelihood using gradient descent. You will also plot the decision boundaries learned by the logistic regression classifiers.

All the code for this question is in the template file `lab4_q1_template.py`.

There are three tasks specified in this question, all using 2-dimensional data points. Each task has its respective function, `task_1`, `task_2` and `task_3`, and uses the data files `q1_data.txt`, `q2_data.txt` and `q3_data.txt`, respectively. The new code you need to write is marked with TODO in the comments.

Before filling in the required functions and implementing the LR classifier, it will help to visualize the data using the scatterplot function that is already available in `visualize_data`. This will give you a sense of what feature transformations would be needed to the 2-dimensional data points to make them separable by an LR classifier.

Complete the following functions:

1. `loss_grad_function()`: Compute the binary cross-entropy loss and the gradient of the loss function using vectorized operations. The expressions for the loss $L(\mathbf{w})$ and gradient $G(\mathbf{w})$ from a logistic regression classifier over a dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ are given below:

$$L(\mathbf{w}, \mathcal{D}) = -\frac{1}{n} \sum_{i=1}^{n} \Big(y_i \log \sigma(\mathbf{w}^T \mathbf{x}_i) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i)) \Big)$$

$$G(\mathbf{w}, \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} \Big(\sigma(\mathbf{w}^T \mathbf{x}_i) - y_i \Big) \mathbf{x}_i$$

   Here, $\sigma(x) = \frac{1}{1+\exp(-x)}$ refers to the sigmoid function. `self.lambda_value` initialized in the class `LogisticRegression` refers to a scaling factor for a regularizer term. If `self.lambda_value` is non-zero, modify $L(\mathbf{w})$ and $G(\mathbf{w})$ to implement L2-regularized logistic regression. That is, $L(\mathbf{w}, \mathcal{D})$ will contain an additional $\lambda||\mathbf{w}||_2^2$.

2. `gradient_descent()`: Implement gradient descent over a fixed number of epochs. Compute the full gradient over the entire dataset for this problem.

3. `visualize_data()`: Plot decision boundaries for each of the toy tasks, overlayed on top of the scatter plot of the dataset (which is already implemented). For task 3, generate three plots for $\lambda = 0$, $\lambda = 1$ and $\lambda = 100$.

Look for other **TODO** markers in `lab4_q1_template.py` and appropriately fill in the remaining gaps in the template code.

Q2: Logistic Regression on Real Data

Implement a logistic regression classifier with L2-regularization using batched gradient descent and train on a real dataset specified in `train_data.csv`. The dataset is the same one you have worked with in previous labs that takes a set of 90 timbre-based audio features as input. Instead of predicting the release year of a song from these features as in previous assignments, in this lab, you will map each training instance to a binary label signifying whether or not the song is old (0) or new (1).

`create_batches()` is already implemented. You should complete the following two functions:

- `loss_grad_function()`
- `batch_gradient_descent()`

You can reuse code from Q1 to implement these two functions.

❄ Extra Credit: Climb the Leaderboard on Kaggle

The objective for this extra credit part is to build the best possible regularized logistic regression model using any enhancements you like. Submit your target predictions for the instances in `hidden_test_data.csv` to Kaggle at `https://www.kaggle.com/competitions/lab4-cs335` 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.