IIT BOMBAY

Lab Assignment #12
Points: 5

Main TAs: *Ashish Agrawal, Tejpalsingh Baljeetsingh Siledar, Ashwin Ramachandran*

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

General Instructions

1. Download lab12.tgz(`https://drive.google.com/file/d/1H6GESOCxaIJHfJOLoiQ4j8LARPt8lcgS/view?usp=sharing`) from Moodle, and extract the file to get `lab12`.

2. For your final submission, you need to submit three `.py` files with all the TODOs implemented: `q1.py`, `q2.py` and `q3.py`. Submit these three files together as a single .tgz file named `[rollnumber].tgz`. (If you attempt the optional problem in Q2, you will also have a file titled `q2_parallel.py`.)

3. Your final submission is due on Moodle on or before **11.59 pm on Nov 7, 2023**.

4. For Q1, you will get 2 points if your training error is 0 and the plot of the learned decision boundary using Adaboost is correct. For Q2, you will get 2 points if you match the test precision from the TA's solution (≈ 0.65). For Q3, you will get 1 point if you match the TAs system on Kaggle. You can get up to 2 extra credit points for Q1 and Q2: 1 point for visualizing the progress of Adaboost and 1 point for parallelizing the random forest classifier.

Q1. Adaboost

Implement the Adaboost algorithm from scratch. Adaboost ensembles weak classifiers (that perform better than chance) and make it stronger by focussing on examples that are hard to classify.

Given training instances $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \{1, -1\}$, recall the Adaboost algorithm is as follows:

1: Initialize uniform weights for each training example $\forall i\ \mathrm{w}_t(i) = \frac{1}{N}$
2: **for** t = 1 ... T **do** ▷ loop over T iterations where T is a hyperparameter
3: Train a weak classifier $h_t(\mathbf{x})$ on training data weighted by $\mathrm{w}_t(i)$
4: Compute the weighted error of $h_t(\mathbf{x})$:

$$\epsilon_t = \sum_i \mathrm{w}_t(i)\ \mathbb{1}[h_t(\mathbf{x}_i) \neq y_i]$$

5: Compute importance scores (proportional to the error ϵ_t):

$$\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

6: Update weights of the training examples:

$$\mathrm{w}_{t+1}(i) = \frac{\mathrm{w}_t(i) \exp(-\alpha_t y_i h_t(\mathbf{x}_i))}{\sum_j \mathrm{w}_t(j) \exp(-\alpha_t y_j h_t(\mathbf{x}_j))}$$

7: **end for**
8: The final prediction is: sign $(\sum_t \alpha_t h_t(\mathbf{x}))$

Complete the following TODOs outlined in the source file `q1.py`:

1. Implement both `fit` and `predict` functions within the class `AdaBoostClassifier`. The weak classifiers are decision stumps (i.e., a single decision tree node). `train_err` printed at the very end should be 0 for the given dataset.

2. Within `plot_adaboost`, printing a scatter plot of the dataset is already implemented. Write code to plot the learned decision boundary via the classifier passed as the command line argument `clf` to `plot_adaboost`.

3. ❄ At each iteration of Adaboost, you can plot both the weak learner $h_t(\mathbf{x})$ and the strong learner estimated thus far (i.e., $\sum_{j=1}^t \alpha_t h_j(\mathbf{x})$). This will help visualize how Adaboost progressively learns strong learners that aim to perfectly learn the training data instances. This is an optional TODO worth 1 extra credit point. If you are attempting this part, implement the function `visualize_stepwise_adaboost` and uncomment the line calling this function in `q1.py`.

Q2. Bagging with Random Forests

Bagging or (Bootstrap Aggregating) refers to an ensembling technique by first bootstrap sampling (i.e., creating multiple datasets by repeatedly sampling from the original dataset with replacement), training a classifier on each bootstrap sample and aggregating predictions via a majority vote from the classifiers in the ensemble.

In this problem, you will implement a random forest (RF). An RF is just bagging with decision-tree classifiers; additionally, for every split in every DT, only a random subset of k features are considered instead of all the original features. The algorithm can be summarized as follows:

1: **Input:** Dataset $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ of N training samples. Each $\mathbf{x}_i$ consists of d attributes.
2: Do bootstrap sampling on D and create m different datasets $D_1, \ldots, D_m$ each of size N by sampling from D with replacement.
3: For each D_i, train a decision tree (DT) classifier. `decision_tree.py` contains an implementation of a simple decision tree classifier. Random forests adopt an additional trick beyond standard bagging. Each DT only uses a subset of $k < d$ attributes randomly sampled from the original set of d attributes. Before fitting a DT to the data in D_i, pick a random subset of k features and use them to train the DT. (k is set to 7 in the template code.)
4: Aggregate the predictions from the k different DTs using a majority vote.

Within the class `RandomForest` in `q2.py`, complete the definitions of `fit` and `predict`.

❄ **Parallelization.** Apart from superior performance, another desirable feature of Random Forests is that its training is parallelizable (unlike Adaboost). Try to parallelize your code in `q2.py` in a separate file `q2_parallel.py`. Print `training_time` in `q2_parallel.py`, as was done in `q2.py`. This is an optional question that will earn you 1 extra credit point if your parallelized code is at least 2 or 3 times faster than the run in `q2.py`. (*Hint*: The library `multiprocessing` (`https://docs.python.org/3/library/multiprocessing.html`) might be of use.)

Q3. Climb the Leaderboard on Kaggle

For this part, you are given data for a real problem and need to fit a high-performing ensemble classifier to this data. `q3.py` contains code to preprocess the instances in the folder `lab12/data_q3/` and create train/test splits. You should fit an ensemble classifier to the training data and submit test predictions (for the instances in `data_q3/test_new.csv` to Kaggle (`https://www.kaggle.com/t/587a86ca147d4d92aaf5488857726c3e`)). The file format of `submission.csv` is id, label on each line where label = {0,1}.

You can use ensemble classifiers from the `sklearn` library such as `AdaBoostClassifier`, `RandomForestClassifier`, etc. Your roll number should appear on both the "Public Leaderboard" (and eventually the "Private Leaderboard" after the assignment concludes). The current TA submission on the leaderboard uses a random forest classifier with `n_estimators`=100, `min_samples_split`=2, `min_samples_leaf`=1 (random seed of 42). 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.