CSE 142 – Machine Learning, Fall 2021

Assignment #4
Due Wednesday, December 1st, 2021

Notes:

- *This assignment is to be done individually. You may discuss the problems at a general level with others in the class (e.g., about the concepts underlying the problem, or what lecture or reading material may be relevant), but the work you turn in must be solely your own.*
- Be sure to re-read the "Policy on Academic Integrity" on the course syllabus.
- Be aware of the late policy in the course syllabus – i.e., *late submissions will not be accepted*, so turn in what you have by the due time.
- Any updates or corrections will be posted on the Canvas and Piazza, so check there occasionally.
- Your code must be well commented – graders should not have to work to figure out the purpose or method of any section or line or code.
- Turning in the assignment:
  - *Include your Codalab username in your report* and submit it through Gradescope
  - Submit your code on Codalab. See the problem description for details

---

For this assignment, we will hold 1 Codalab competition.

Problem #1 [100 points]

For this assignment, we publish a CodaLab competition. The competition link is:
https://competitions.codalab.org/competitions/36348?secret_key=16f533ca-ebd8-4099-b80d-f15e338f2fbe

The late submission competition link is:
https://competitions.codalab.org/competitions/36349?secret_key=ac3d2f75-3c3a-45b3-97d0-dcb8298d8aba

In this problem you'll create a binary classifier using boosting, based on a variation of the simple linear classifier (similar to what you implemented in HW2, problem 6, but with two classes instead of three).

Write Python code that implements boosting for binary classification, using the following modification of the basic linear classifier as the model. Instead of computing each class exemplar by taking the mean of the class training points, use the weighted mean. I.e., instead of

$$\text{class\_exemplar} = \frac{1}{k}\sum_{i=1}^{k} x_i \text{ (where } k \text{ is the number of training points in the class)}$$

use

$$\text{class\_exemplar} = \frac{1}{\sum w_i}\sum_{i=1}^{k} w_i x_i \text{ (where } w_i \text{ is the weight associate with point } x_i\text{)}$$

As described in the boosting algorithm, these weights w_i are initially evenly distributed among all of the training points (each weight is initialized to 1/|D|, where |D| is the number of training points), and then are updated at each iteration of the boosting algorithm. As before, the discrimination surface of each model is halfway between the two class exemplars, perpendicular to the vector connecting the exemplars. Once the boosting algorithm is run on the training points, classify the test points based on the weighted average of the T models $M_t(x)$. For classifying test points, use $M(x) > 0$ to predict the positive class (and thus $M(x) \leq 0$ to predict the negative class).

Implementation

Implement a BoostingClassifier class in the boostit.py. There are two interface function: fit(X_train, y_train) and predict(X_test).
The fit() function includes the training process given the training data X_train and y_train. The predict() function includes the process of generating predictions.
For loading the dataset, use the following code as shown in local_evaluation.py:

```
import numpy as np
with open('train.npy', 'rb') as f:
    X_train = np.load(f)
    Y_train = np.load(f)
```

You can assume that the two classes are not linearly separable, so you will never get an error of zero for an iteration over the training data. (Otherwise, you'll get a divide-by-zero error!) After you implement the BoostingClassifier class in boostit.py, you can run

```
$ python3 local_evaluation.py
```

to evaluate the performance on your local dataset. Please make sure your dataset file is on the correct path. Note we are using Python3 on Codalab.

Submission

You only need to zip the **boostit.py** file, as well as other source code you used in your implementation, into your submission file to Codalab. The online evaluation script on Codalab is the same as the **local_evaluation.py** except for the path to datasets.

Dataset

There are 3 datasets in total. Dataset 1 is used for your offline evaluation, and dataset 2 and dataset3 are used on the online Codalab competition. **You should submit your code to both dataset2 and dataset3 on Codalab.** The datasets are in the same format, except for the dimensions of feature vectors. The feature vectors in training and testing data sets all have the same format, each comprising M N-dimensional points:

```
P11 P12 … P1N
P21 P22 … P2N
…
PM1 PM2 … PMN
```

where M is the number of instances, N is the dimensions of feature vectors, Pij is real-valued, describing the j^{th} component of point i. The values of N must be the same in a train-test split of the same dataset.

Codalab performance [60/100 points]

$$final\_score = 50 \times accuracy + 50 \times F\_score$$

Report [40/100 point]

Report part1: What you did: [10/40 points]
Write a description of your code, how you made it work, what issues did you face. Or anything special about your code that instructors need to know about.

Report part2: How is your code's performance locally: [10/40 points]
Run the program on the provided data sets with T=5 on Dataset, basically your local performance. Save the output from these examples in different text files to be turned in with your report on Gradescope. Note that your CodaLab submission does not need to print this.

As output, your program should print out information about the boosting iterations and about the classification of the test data, as in this example:

```
% python run.py
Iteration 1:
Error = 0.15
Alpha = 0.8673
Factor to increase weights = 3.3333
Factor to decrease weights = 0.5882
Iteration 2:
Error = 0.22
```

```
Alpha = 0.6328
Factor to increase weights = 2.2727
Factor to decrease weights = 0.6410
(...etc... for all iterations of boosting)

Testing:
False positives: 10
False negatives: 7
Error rate: 9%
```

Report part3: Compare this and linear classifier: [20/40 points]
You need to compare this boosting classifier with the original linear classifier and also to compare results using different values of T. You are also encouraged to visualize the data and results (training data points, linear classifiers trained, and training data points and their classification).

Common issues:

1. **Use the same username as your CruzID and include your username in the report.**
2. There is a time difference between the CodaLab time. The time zone we use is still based on the California Pacific time zone.
3. This time the data will be loaded from the eval script, students will no longer need to do that in the fit() or predict() function.