ECE 763 Project 2 Report

Jie Hu

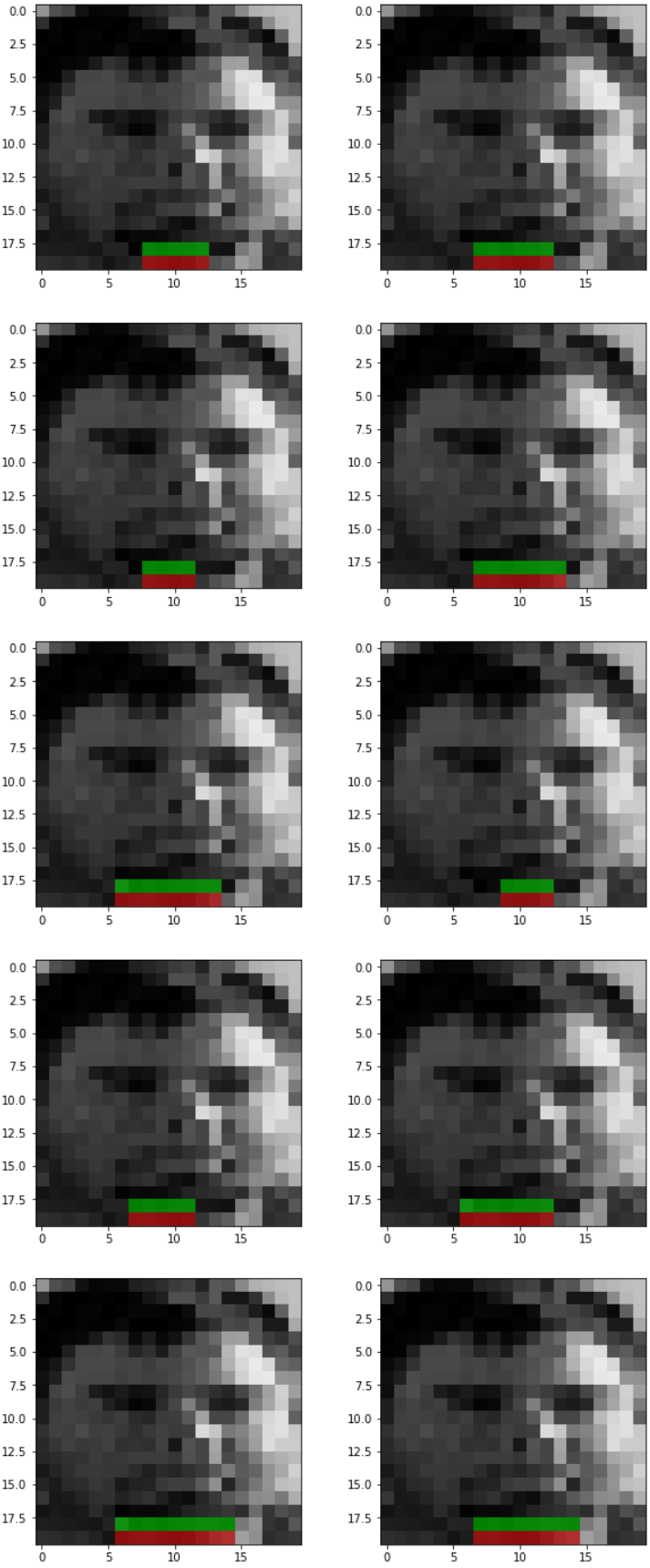
03/28/2022

Setup

In this project, I reuse the dataset I created in project 1, which includes train dataset with 1000 face images and 1000 non-face images, and test dataset with 100 face images and 100 non-face images. The images in the original dataset are of size 20x20 with RGB format. To proceed this project, I convert the RGB images to gray images with opencv package. I also implement parallel computation to speed up the training process, e.g., using *njit* from numba package and *delayed* from dask package.

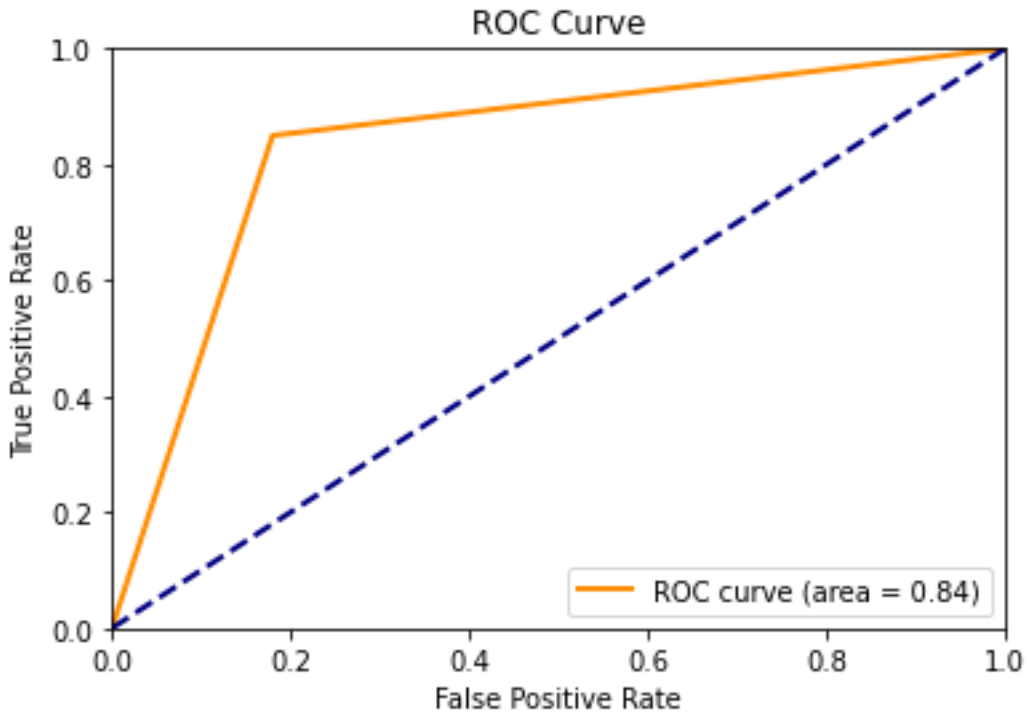
Task 1: Calculate the classification error for each weak learner and draw the best ten features (before boosting)

To extract Haar feature, I use the existing function *haar_like_feature* from scikit-image package, and I obtain 78460 features. The corresponding coordinations are generated by function *haar_like_feature_coord* from scikit-image package. Each feature is considered as a weak classifier with attributes: threshold, sign (flipped the sign if error is larger than 0.5), error and the index of the current feature in the feature array. Then, I have 78460 weak classifiers before boosting. By sorting the error of each weak classifier in ascending order, I select the top ten features, and visualize the Haar features with function *draw_haar_like_feature*. The figure of the top ten features is given as follows.


0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
5
10
15

Task 2: Implementing the Adaboost Algorithm and test it in the test dataset you created. Plot the ROC based on thresholding $H(X)$.

Even using parallel computation is too slow for the training process because of the number of Haar features. To facilitate the simulation, I only choose the top ten features, which are derived before boosting in task 1, in the Adaboost algorithm. Following the Adaboost algorithm and train the model for 100 iterations, I find the model achieves 83.5% accuracy and the ROC plot is given in the following figure.



Summary of the observations of the refactoring process in terms of coding

Previously, my code is hard to adjust different parameters or dataset because each method is written separately and there is no connection between methods (reusable parameters). The image patch processing is also written such that parameters are defined at the beginning of the code. Now, I have reconstructed each method as a class so that they are easy to call and are able to receive arguments from the input, and display the result accordingly. Besides, after doing task 1 and 2, I optimize the code for project 1 where the parallel computation is used to accelerate the simulation.