CS 596 Machine Learning

**Homework Assignment 8**

Due: 11:59pm, May 10, 2018

In this mini-project, you will be instructed to implement the Adaboost algorithm, an ensemble machine learning model, and use it for a binary classification problem. We will work on a randomly generated dataset where every sample is described with two features.

**Overview of Adaboost**

Adaboost is a type of bagging method, aiming to create a strong classifier through combining multiple weak classifiers. One can use any machine learning models (e.g. decision tree, SVMM, neural networks) to create weak classifiers. In this project, we will use *decision stumps* as weak classifiers. This is a threshold-based decision rule using one of the sample features. Let  denote the feature vector of the sample i, and denote the d-th element of , and C denote a constant for thresholding. A decision stump  has the form:

or

Therefore, a decision stump has three parameters: feature index *d*, threshold value C, and direction.

**Overview of the Script**

A starter script *main\_adaboost\_ha8.py* is provided to facilitate your programming. You will need to understand the whole script and modify PLACEHOLDER code lines accordingly to accomplish the pipeline.

Thee starter codes include three major steps: load training and testing data, training (boosting), and testing. You don’t need to change the codes for data loading.

The main script includes the following functions:

* *def func\_readData(filename,option):* read training and testing data from files
* *def* *stumpClassify(datamat,d,threshold,inequal)*: implementation of a decision stump. This function takes as inputs data matrix (*datamat*), index of feature (*d*) and a threshold. Note that if ‘inequal’ is ‘lt’, the weak classifier is defined as ; otherwise, .
* *def buildWeakStump(data,label,D):* find the optimal weak classifier (decision stumps) with the least training error. Arguments: data, label, and sample weights (i.e. D)
* *def train(data,label,numIt = 1000):* implementation of the AdaBoosting algorithm: retrieving a set of weak classifiers and their weights.
* *def adaboostClassify(dataTest,classifier):* classifying a single data sample using the strong classifier (i.e., a set of weak classifiers)
* *def test(dataSet,classifier):* classify a set of data samples using the strong classifier.

**Implementation of train()**

This function implements the AdaBoosting algorithm, and returns a set of weak classifiers with classifier weights. This iterative function includes three major steps:

**Step 1**: Select the optimal weak classifiers.

To do so, the starter codes provide a function bestStump, err, classEstimate = buildWeakStump(data,label,D). You don’t need to change the codes for this step – it is strongly recommended to read and understand this function for your best interest. This function will return three variables.

* *bestStump*: the selected weak classifier, being composed of three fields: dim (feature index), threshold, and ineq (indicate if use ‘<=’ or ‘>’ to define weak classifiers).
* *estimateclass*: predictions made by the selected weak classifier;
* *err*: error rate of the selected weak classifier;

**Step 2:** Calculate the weight of a weak classifier according to its error rate.

You will need to implement this step in PLACEHOLDER 1. The weight of a weak classifier (or a decision stump) is calculated based on its error rate:

where is a small constant (e.g., 1e-16) and t represents the current iteration.

**Step 3:** Update sample weights.

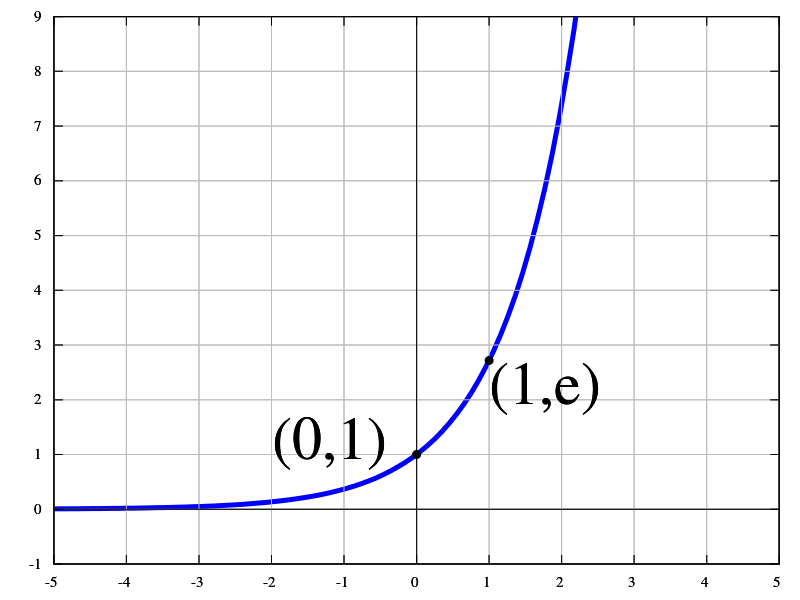
You will need to implement this step in PLACEHOLDER 2.

At each iteration of an Adaboost algorithm, a sample misclassified by the currently selected decision stump will be assigned with a larger weight; a correctly classified sample will be assigned with a smaller weight.

Let represent the current weight of a sample , let (+1 or -1) denote its label, denote the currently selected weak classifier. The new weight of this sample can be calculated as:

where m is the number of training samples. The bottom equation is used to normalize all sample weights so that their sum is equal to 1.

*How does the above update equations work to implement the boosting idea?* To answer this question, you will need to know the shape of exp():



For a training sample , if returns +1, will return a positive number between 0 and 1. Therefore, the update equation will decrease the value of the sample weight.

The provided codes in train() will print the current training error rate of each selected weak classifier. You might include these in your report and observe how it changes over iterations.

**Implementation of test()**

You will need to complete the PLACEHOLDER 3 in this function.

To obtain the label for an unseen testing sample, we first apply each weak classifier, i.e., call the function stumpClassify(), which returns +1 or -1.

Let *x* denote the feature of a testing sample, the final prediction is a linear combination of the outputs of all weak classifiers ():

where  returns +1 or -1.

Please apply your strong classifiers over testing samples and include the predicated labels (+1 or -1) in your report.

**Extra Credits (0.2-2 credits)**

Extra credits will be given to submissions which apply this Adaboost algorithm over the dataset provided in ha6, i.e. crab gender classification. You might use 2 or more features per crab sample. In your report, please describe how you split the training/testing data, and include your result (accuracy or error rate).