Boosting

This is an optional assignment, and if you choose to submit a solution your final grade will be determined based on best six out of the seven assignments. If you submit, you should be able to do the codewalk on or before April,30.

1   Boosting

Algorithm-1 lists a generic boosting algorithm. Boosting relies on training a weak (base) classifier on weighted data. In general, boosting provides a clean interface between boosting and the underlying weak learning algorithm: in each round, the boosting algorithm provides a weighted data set to the weak learner, and the weak learner provides a predictor in return. You may choose to keep this clean interface (which would allow you to run boosting using any weak learning algorithm) or you may choose to more simply incorporate the weak learning algorithm inside your boosting code. As boosting remains agnostic to the weak learner, it is known as a *meta-learning* algorithm.

The weighting scheme (how α is defined in Algorithm-1) can be varied to modify the overall boosting algorithms. In this question you will implement AdaBoost which chooses α as:

$$\alpha_t = \frac{1}{2} \ln \frac{1 + \gamma_t}{1 - \gamma_t}$$

where γ is the so-called *edge* of $h_t(x)$, and defined as:

$$\gamma_t = \sum_{i=1}^{N} D_t(i) y_i h_t(x_i)$$

Data: Training data: $(x_1, y_1), \ldots, (x_N, y_N)$, where $x_i \in \mathbb{R}^m$ and $y_i \in \{-1, 1\}$
Result: Boosted Classifier: $H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$
Initialize $D_1(i) \leftarrow \frac{1}{N}$;
for *t=1 to T* **do**

    Train base classifier using distribution D_t;
    Get base classifier, $h_t : X \rightarrow \{-1, 1\}$;
    Choose $\alpha_t \in \mathbb{R}$;
    Update: $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$, where Z_t is chosen so that $\sum_i D_{t+1}(i) = 1$;

end

Algorithm 1: A generic algorithm for boosting.

1.1 Decision Stumps

Each predictor will correspond to a decision stump, which is just a feature-threshold pair. Note that for each feature f_i, you may have many possible thresholds which we shall denote as τ. Given an input instance to classify, a decision stump corresponding to feature f_i and threshold τ will predict $+1$ if the input instance has a feature f_i value greater than the threshold τ; otherwise, it predicts -1.

Determining suitable threshold: To create the various thresholds for each feature f_i, you should:

1. sort the training examples by their f_i values

2. remove duplicate values

3. construct thesholds that are midway between successive feature values

You should also add two thresholds for each feature: one below all observed values for that feature and one above all values for that feature. By removing duplicate values we are trying to cut down the number of thresold we need to check. (*Note:* The search procedure is the same as used for selecting a threshold for defining a binary split on a contiuous feature in decision trees, except we are maximizing the training error instead of information gain or Gini index.)

Optimal Decision Stump: The optimal threshold τ^* for feature f_i corresponds to the threshold that maximizes:

$$\tau^* = \operatorname*{argmax}_{\tau} |\frac{1}{2} - error(\tau)|$$

where, $error(.)$ is calculated as the sum of the weights $(D_t(i))$ of the misclassified instances. In this case since we are dealing with absolute values if a threshold has an error of 0.2 and another threshold produces an error of 0.9 we will prefer the latter one since it has a higher difference from 0.5 (for such decision stumps you would need to invert the prediction so that for instances with a feature value greater than τ you will predict -1 and $+1$ otherwise. You would need to remember this when you combine such decision stumps with others to evaluate $H(x)$).

1.2 Programming Tasks

1. Implement the AdaBoost algorithm, using optimal decision stumps.

2. Implement the AdaBoost algorithm, using random decision stumps (see below).

Random decision stumps on the other hand are completely random, where you will select the feature and threshold randomly.

Run your AdaBoost implementation on the Spambase, Diabetes and Breast Cancer datasets. For each dataset partition it into a training set (80%) and a test set (20%). After each round, you should compute:

1. The local "round" error for the decision stump returned, i.e., the error using h_t measured using D_t

2. The training error for the linear combination of decision stumps, i.e., the error using H_t measured on training data (ignoring D_t)

3. The test error for the linear combination of decision stumps, i.e., the error using H_t measured on test data

1.3 Deliverables:

1. *Boosting with optimal decision stumps:* For each dataset provide three plots that show the local error, training error and test error.

2. *Boosting with random decision stumps:* For each dataset provide three plots that show the local error, training error and test error.

3. In both cases how did you determine that boosting has converged i.e., how did you set the parameter T in Algorithm-1.