**Machine Learning**

# Lab 5: Ensemble Methods for Classification

## 1 Overview

> "The AdaBoost algorithm proposed by Yoav Freund and Robert Schapire is one of the most important ensemble methods, since it has solid theoretical foundation, very accurate prediction, great simplicity (Schapire said it needs only 'just 10 lines of code'), and wide and successful applications." [1]

For this lab you will be programming an ensemble learning algorithm known as AdaBoost to solve a two-class data classification problem. The idea behind the algorithm is to combine multiple "weak learners", or small models with poor but above-random performance, in ways which ultimately construct a robust and general classification algorithm. The weak learner in this case is provided: it searches for an axis-aligned hyperplane that best separates two classes of data. The AdaBoost algorithm alternates between weighing controversial observations more heavily and finding classifiers which, when combined, begin to learn the geometry of the class label divisions. Here you will study this algorithm in a synthetic setting where a single weak learner has poor classification accuracy but the ensemble is capable of building an accurate classifier.

## 2 Provided Resources

- `learnweak.m` - The provided weak learning algorithm for this lab. The function will fit to data in two classes with the provided weights, and return parameters for a hyperplane which separates the classes with at least 50% accuracy.
- `evalweak.m` - After learning parameters for a weak classifier, this function will calculate the predicted class labels for data.
- `boostlearn.m` - To be completed, a function that generates a sequence of weak classifiers and a sequence of weights that describe the final boosted classifier.
- `boosteval.m` - To be completed, a function which uses the learned sequence of weak classifiers and classifier weights from `boostlearn.m` to calculate class labels for input data.
- `make_cloud.m` - Function that samples observations of data from two classes in two dimensions.

## 3 Guide

1. Complete and turn in the function `boostlearn.m` as follows:
   (a) Observation locations are given in $x_i$ with class assignments $t_i \in \{-1, 1\}$.
   (b) Initially, each observation is given the same weight $v_i$, normalized so that $\sum_i v_i = 1$. Thus the weight of each observation will be $1/N$ where $N$ is the number of observations.
   (c) At the $k^{\text{th}}$ iteration of the algorithm (looping from 1 to $M$), perform the following steps:
      i. First, use `weaklearn.m` with the weights (uniform in the first iteration) to learn the parameters for a weak classifier. This is the $k^{\text{th}}$ weak classifier which the function will output.
      ii. Calculate the weighted fraction $\varepsilon$ of observations mislabeled by this new classifier. That is, if the new weak classifier predicts the labels $\hat{t}_i$ then
      $$\varepsilon = \sum_{i:\, \hat{t}_i \neq t_i} v_i \Big/ \sum_i v_i$$

[1] "Top 10 algorithms in data mining" by X. Wu et al. DOI 10.1007/s10115-007-0114-2

iii. Compute $\alpha_k = \ln \frac{1-\varepsilon}{\varepsilon}$.

iv. Update the observation weights by the equation

$$\forall i\colon \hat{t}_i \neq t_i\colon \quad v_i \leftarrow v_i e^{\alpha_k}$$

(d) After performing each of the above steps a fixed number times $M$, the algorithm returns the parameters for the $M$ weak classifiers and $M$ classifier weights $\alpha_k$.

2. Complete the function `boosteval.m`. This function accepts the parameters for the learned classifiers and weights from `boostlearn.m`, then returns labels, by committee vote, for each input observation. Recall that the label is found using the sign of $\sum_k \alpha_k f_k$ where $f_k$ is the label given to the observation by the $k^{\text{th}}$ weak classifier.

3. Use `make_cloud.m` to generate training data sampled from two point clouds.

4. First, let's establish the performance of a single weak learner: use `weaklearn.m` to classify the data directly (with uniform weights). Plot the data (for example, use `scatter`) and distinguish between classes, and whether we classify correctly, or not (e.g. 'x' vs 'o' for the true class, and blue versus red for correct/wrong classification). How well did this classifier perform? Could a different single linear classifier perform better on this data? If so, what would the best case look like?

5. Use `boostlearn.m` to classify the same data as above with $M = 5$ weak classifiers. Generate and turn in a figure showing the classification result.

6. Try larger values of $M$ in the range 1 to 100 and study how the classifier changes. Based on visual inspection of the results: At what point do you believe the classifier is sufficiently reflecting the data? At what point do you believe the classifier might be over-fitting?

7. Use `make_cloud.m` to generate a second set of observations (validation data). Test values for $M$, systematically. That is, for varying values of $M$, train AdaBoost on the training data, and evaluate its performance on the validation data. Generate and turn in a plot with $M$ on the x-axis and the misclassification rate for validation data on the y-axis. Describe the shape of this graph.