Ensemble Learning

UCLA Math 156

Spring 2016

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. (4) Complete and turn in the function `boostlearn.m`. The function takes two sets of locations, `X0` and `X1`, corresponding to samples of the two classes. We call the observation locations to be trained on x_n with class assignments $y_n \in \{-1, 1\}$ ($+1$ for samples in `X0`, -1 for samples in `X1`). Initially, each observation is given the same weight w_n, normalized so that $\sum_n w_n = 1$. Thus the weight of each observation will be $1/N$ where N is the number of observations.

   At the m^{th} iteration of the algorithm, perform the following steps:

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

First, use `weaklearn.m` with the weights (uniform in the first iteration) to learn the parameters for a weak classifier. This is the m^{th} weak classifier which the function will output.

Second, calculate the weighted fraction ϵ_m of observations mislabeled by this new classifier. That is, if the new weak classifier predicts the labels $\hat{y}_m(x_n)$ then

$$\epsilon_m = \sum_{n:\hat{y}_m(x_n)\neq y_n} w_n.$$

Third, compute $\alpha_m = 0.5 \log \frac{1-\epsilon_m}{\epsilon_m}$ and update the observation weights by the equation

$$w_n \leftarrow \frac{1}{C} w_n e^{-\hat{y}_m(x_n) y_n \alpha_m} \quad \forall\, n.$$

Here C is chosen so that $\sum_i w_i = 1$ remains true; in practice, update the weights then re-normalize them to sum to 1.

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 α_m.

2. Complete the function `boosteval.m` which accepts the parameters for the learned classifiers and weights from `boostlearn.m`, then returns labels for each input observation at locations `X`. Recall that the label is found using the sign of $\sum_m \alpha_m \hat{y}_m(x)$ where $\hat{y}_m(x)$ is the label given to the observation x by the m^{th} weak classifier.

3. (4) Use `make_cloud.m` to generate data sampled from two point clouds, and use `weaklearn.m` to classify the data directly (with uniform weights). 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?

4. (4) 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 (for example, use `scatter`).

5. (4) Try larger values of M in the range 1 to 100 and study how the classifier changes. 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?

6. (4) Use `make_cloud.m` to generate more observations and test values for M. That is, generate and turn in a plot with M on the x-axis and the mis-classification rate for new data on the y-axis. Describe the shape of this graph.