Adaptive Weak Learner Boosting

Application Note

Asher Hensley

I. Introduction

This application note describes a machine learning algorithm called Adaptive Boosting (or AdaBoost) using decision stumps; a simple yet powerful technique for both feature mining and pattern classification problems. AdaBoost is an ensemble classifier which combines several weak classifiers into a single strong classifier using an iterative reweighting scheme on the training data. This results in a stronger emphasis being placed on samples that harder to classify as learning proceeds. This technique is very robust to overfitting, it can learn the best discriminating features automatically, and it is considered one of the best "off-the-shelf" pattern classification algorithms available.

II. Algorithm

Here's how it works: given a set of m training examples $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)\}$ where $\mathbf{x}_i \in \mathbb{R}^{n\times 1}$ is the i^{th} feature vector and $y_i \in \{-1, +1\}$ is the i^{th} binary truth label, the first step is to initialize a training example weight vector using[1]:

$$D_1(i) = \frac{1}{m} \quad i = 1, 2, ..., m \tag{1}$$

The next step is to specify a weak learner $h_t(\mathbf{x})$ which, for the decision stump, is given by:

$$h_t(\mathbf{x}) = c_t \times \{I(x_k \geq X_t) - I(x_k < X_t)\} \tag{2}$$

where x_k is the k^{th} feature, X_t is the decision boundary, $c_t \in \{-1, +1\}$ is the class label being detected, and $I(u)$ is the indicator function which is equal to 1 when its argument is true (zero else). The weighted training error for boosting round t is then given by:

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) I(h_t(\mathbf{x}_i) \neq y_i) \tag{3}$$

Classifier training then proceeds as follows:

Algorithm: AdaBoost

1. **for** $t = 1$ to T
2. Determine $h_t(\mathbf{x})$ which minimizes ϵ_t
3. Compute $\alpha_t = \frac{1}{2}(\ln(1 - \epsilon_t) - \ln(\epsilon_t))$
4. Update $D_{t+1}(i) = Z_{t+1}^{-1} D_t(i) \exp(-\alpha_t y_i h_t(\mathbf{x}_i))$
5. **return** $\{\alpha_1, ..., \alpha_T\}, \{h_1(\mathbf{x}), ..., h_T(\mathbf{x})\}$

where Z_t is a normalizing constant given by:

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(\mathbf{x}_i)) \tag{4}$$

After training is complete, new data points are classified according to the following hypothesis function:

$$H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right) \tag{5}$$

[1]This assumes an equal cost of misclassification. A different weighting scheme could be used (for example) to give a larger cost of misclassifying a target than misclassifying clutter.

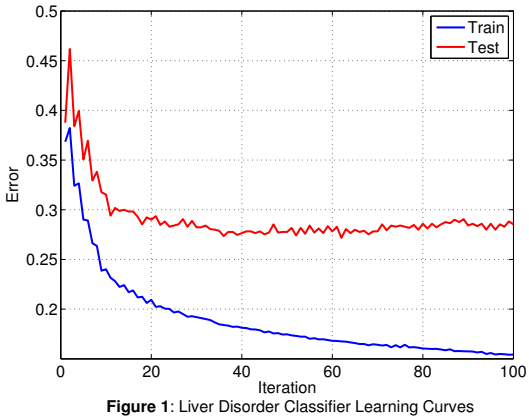

Figure 1: Liver Disorder Classifier Learning Curves

III. Error Bounds

The AdaBoost algorithm has been well studied theoretically and it has been shown the training error on the final classifier can be made arbitrarily small with an upper bound of:

$$e_{train}(H) = \frac{1}{m} \sum_{i=1}^{m} I(H(\mathbf{x}_i) \neq y_i) \leq \prod_{t=1}^{T} Z_t \tag{6}$$

which says the training error goes to zero exponentially. Based on this, a theoretical upper bound on the true error is claimed to be:

$$e_{true}(H) \leq e_{train}(H) + \mathcal{O}\left(\sqrt{\frac{Td}{m}}\right) \tag{7}$$

where d is the Vapnik-Chernovenkis (VC) dimension of the weak learner. This result suggests that with too many boosting rounds T, the classifier will overfit. However there is experimental evidence showing that AdaBoost is very robust to overfitting even after thousands of boosting rounds. In some cases the hold out test error has been shown to decrease even after the training error has been driven to zero.

IV. Analysis: Liver Disorder Classifier

As an example, consider the performance of the AdaBoost algorithm on the BUPA liver disorder data set. The data set consists of 345 patients each with the following features:

1. mean corpuscular volume
2. alkaline phosphotase
3. alamine aminotransferase
4. aspartate aminotransferase
5. gamma-glutamyl transpeptidase
6. number of alcoholic beverages/day

A Matlab implementation of the AdaBoost algorithm was run 50 times on this data set, each with a random split of 90% training data and 10% test data. The resulting training and test classification errors were then averaged over each run and plotted versus boosting round in **Figure 1**. From this result we can expect a classification error of around 27% on unseen patents using roughly 40 boosting rounds and be confident we have not overfit.