An Introduction to Boosting

Abraham Horowitz

May 8, 2017

1 Introduction

Boosting is a meta-learning algorithm, which means that is a function of other functions, which are in turn transformations of the original datasets. Its objective is to maximize predictability over the entire distribution of outputs, and it is characterized by the individual weak algorithms that only learn a portion of the output distribution.

At each iteration, LogitBoost optimizes a learning algorithm, and computes the error for each example with a logistic loss function. The loss is then designated as a weight for that example in the next iteration, when the distribution is re-weighted or re-sampled using the weights.

2 The Algorithm Explained

In our example, the jth iteration of LogitBoost creates one feedforward neural network. After optimization, it computes the logistic loss, which acts as a weight, for the ith example:

$$w_j^i = \epsilon_j^i = \frac{1}{1 + e^{\lambda_j y_i h_j(x_i)}} \tag{1}$$

where $h_j(x_i)$ is the predicted output for the ith example by the jth learner, and y_i is the actual output. The λ parameter is for regularization. For learner j, it is computed as:

$$\lambda_j = \frac{1}{2} \ln \frac{1 - \epsilon_j}{\epsilon_j} \tag{2}$$

Regularizing the example weight vector prevents overfitting residuals by the next learner, but allows for the next iterator to learn from a slightly re-focused distribution.

Equation (2) computes the log-likelihood of model accuracy. The intuition is as follows: If a model has a high error rate, the value of λ will be low and consequently, the example weights for high-cost examples will be high. The latter is necessary to achieve boosting's objective and is acceptable in this case because if model quality is poor, there will be a large amount of high-weight examples and thus a low risk of overfitting to them.

Additionally, if the model has high accuracy, weight of high-cost examples will be **relatively low**. This is very important because if $\Pr(h_j(x_i) = y_i)$ is high, then the error has probably converged and reweighting the distribution to learn a small set of residuals will be overfitting and will exponentially increase model error. The λ parameter thus exists to perform regularizing transformations on equation (1) that are dependent on local model error.

Moving forward, the distribution D is then re-weighted for the next iteration using the example weights.

$$D_{j+1} = \sum_i w_j^i D(i) \tag{3}$$

The algorithm performs this process J times and combines models additively into strong forecast $H(x_i)$, again employing the vector of $J\lambda$ parameters as model weights:

$$H(x_i) = \sum_j \lambda_j h_j(x_i) \tag{4}$$