Boosting with decision stumps and binary features

Jason Rennie
jrennie@ai.mit.edu

April 10, 2003

1 Introduction

A special case of boosting is when features are binary and the base learner is a decision stump (which assigns examples a label based on a single feature). Text classification problems tend to have the property that the presence of a feature is more useful than its absence. But, in this special case, an algorithm like AdaBoost will assign equal magnitude weights to both positive and negative occurrences. This will make the path to convergence less direct and may cause AdaBoost to select features poorly. The alternative is to use a base learner that can abstain. But, then we lose the ability to easily weight nonoccurrences. Solutions include allowing the base learner to weight occurrences and non-occurrences separately and adjusting the bias term to compensate. We discuss such algorithms, pointing out their advantages and disadvantages.

2 Adaboost

In this section, we give the derivation of AdaBoost for the case of binary features and a decision stump base learner. Let $(x_1, \ldots, x_m)$ be the set of training examples, $x_i \in \mathcal{X}$. Let $(y_1, \ldots, y_m)$ be their labels, $y_i \in \{-1, +1\}$. Let $x_{ij} = +1$ if feature j is present in example i, $x_{ij} = -1$ otherwise. Assume that the classifier learned by AdaBoost up to round n is

$$f(x) = w_0 + \sum_{j=1}^{d} w_j x_j, \tag{1}$$

where d is the number of features and x_j is the feature value $\in \{-1, +1\}$ for the test example x. We want to modify the weight on one of the features so

as to maximally decrease the loss functional,

$$L(f) = \sum_{i=1}^{n} e^{-y_i f(x_i)}. \tag{2}$$

If we modify the weight of the k^{th} feature, the loss becomes

$$L_k(f) = \sum_{i=1}^{n} e^{-y_i f(x_i) - y_i \delta x_{ik}}, \tag{3}$$

where δ is the amount we add to w_k. Minimizing with respect to δ gives us

$$\frac{\partial L_k}{\partial \delta} = -\sum_i y_i x_{ik} e^{-y_i f(x_i) - y_i \delta x_{ik}} = 0 \tag{4}$$

$$\iff e^{\delta} \sum_{i: y_i x_{ik} = -1} e^{-y_i f(x_i)} = e^{-\delta} \sum_{i: y_i x_{ik} = 1} e^{-y_i f(x_i)} \tag{5}$$

$$\iff \delta = \frac{1}{2} \log \frac{\sum_{i: y_i x_{ik} = 1} e^{-y_i f(x_i)}}{\sum_{i: y_i x_{ik} = -1} e^{-y_i f(x_i)}} \tag{6}$$

Let $W^+ = \sum_{i: y_i x_{ik} = 1} e^{-y_i f(x_i)}$. Let $W^- = \sum_{i: y_i x_{ik} = -1} e^{-y_i f(x_i)}$. Then, the feature that maximally reduces the loss is defined by the feature, k, that maximizes

$$L(f) - L_k(f) = \sum_i e^{-y_i f(x_i)} - \sum_i e^{-y_i f(x_i) - y_i \delta x_{ik}} \tag{7}$$

$$= (1 - e^{-\delta}) \sum_{i: y_i x_{ik} = 1} e^{-y_i f(x_i)} + (1 - e^{\delta}) \sum_{i: y_i x_{ik} = -1} e^{-y_i f(x_i)} \tag{8}$$

$$= \left(\sqrt{W^+} - \sqrt{W^-} \right)^2. \tag{9}$$

Equivalently, the feature that maximally reduces the loss is the one that maximizes $\left| \sqrt{W^+} - \sqrt{W^-} \right|$.

3 ZeroOneBoost

In the special case that we have described, AdaBoost has the undesirable property that a non-occurrence of a feature contributes the same magnitude score as does an occurrence. We consider a slight variant of AdaBoost

wherein the decision stump returns +1 if the chosen feature occurs and 0 if it does not. We use the same notation as before, except that now $x_{ij} = 0$ if feature j is not present in example i.

Again, for each feature, we want to choose δ to minimize the loss. We result in the same formula for δ,

$$\delta = \frac{1}{2} \log \frac{W^+}{W^-}. \tag{10}$$

But, it is subtly different than before. Whereas before the two sets, $\{i : y_i x_{ik} = 1\}$ and $\{i : y_i x_{ik} = -1\}$, enumerated all possible values of i, they now only enumerate those for which feature k appears ($x_{ik} = 1$). We must now pay attention to the fact that W^+ or W^- can easily be zero. Smoothing is used to avoid taking log of zero or infinity. We compute δ as

$$\delta = \frac{1}{2} \log \frac{W^+ + \epsilon}{W^- + \epsilon}, \tag{11}$$

where ϵ is some small constant.

To determine the feature that maximally reduces the loss, we compute,

$$L(f) - L_k(f). \tag{12}$$

Again, the result is idential to what we found before, with the subtle difference that the two sets of examples that W^+ and W^- deal with do not cover the entire set of examples.

4 Bias Correction

In both of the above-described algorithms, weights are chosen for the features, but in the first case weights are the same for occurences & non-occurrences and in the second case, a weight of zero is always used for non-occurrences. Usually, improvements can be made by adding a bias value.

Consider AdaBoost where we have just chosen a feature, k, and weight, δ. Now we wish to choose a bias, γ, so as to minimize the loss functional. The loss is

$$L(f) = \sum_{i=1}^{n} e^{-y_i f(x_i) - y_i \gamma}. \tag{13}$$

Solving for the minimum, we get

$$\frac{\partial L}{\partial \gamma} = -\sum_{i=1}^{n} y_i e^{-y_i f(x_i) - y_i \gamma} = 0 \tag{14}$$

$$\iff \gamma = \frac{1}{2} \log \frac{\sum_{i:y_i=1} e^{-y_i f(x_i)}}{\sum_{i:y_i=-1} e^{-y_i f(x_i)}}, \tag{15}$$

or, using definitons we defined earlier, $\gamma = \frac{1}{2} \log \frac{W^+}{W^-}$. This is the same whether we are using $-1/+1$ features or $0/+1$ features and we do not need to smooth.

5 Separate occurrence and non-occurrence weights

Schapire and Singer discuss learning weights for the occurrence and non-occurrence of features separately [1]. This is similar to selecting a bias. But, whereas we have described a process of first selecting a weight and then choosing a bias, here the weights are chosen simultaneously, albeit in separate optimizations (the weights are not chosen to joinly minimize the loss functional).

Let $x_{ik} \in \{0, +1\}$. We conduct two separate minimizations. In the first, we choose δ^+, the weight for the feature's occurrence. Then, we choose δ^-, the weight for the features's non-occurrence. The formulas are identical to those already described in section 2. For δ^+, we use $0/+1$ features, so smoothing is required. For δ^-, we use $+1/0$ features. That is, $x_{ij} = 0$ when feature j occurs, $x_{ij} = +1$ when it does not. Each of the two weights will yield a separate reduction in the loss function. We choose the feature whose sum of the two loss reductions is largest.

6 NewtonBoost

None of the above algorithms finds the weight and bias term that minimizes the loss functional. The closest are the algorithms that first choose a weight, then do bias correction. An algorithm that is uniformly better than the one Schapire and Singer describe is one that picks a weight for occurrence or non-occurrence (depending on which gives lower loss) and then does bias correction. Equivalently, one could choose one weight and then choose the second weight in the context of the first.

In this section, we explore the idea of jointly optimizing the occurrence and non-occurrence weights, or jointly optimizing the weight and the bias

term. For each feature, k, consider selecting a weight, δ, and bias, γ. The new loss is

$$L_k(f) = \sum_{i=1}^{n} e^{-y_i f(x_i)} e^{-y_i \delta x_{ik} - y_i \gamma}, \tag{16}$$

Minimizing for γ and δ, we get

$$\frac{\partial L_k}{\partial \gamma} = -\sum_i y_i e^{-y_i f(x_i)} e^{-y_i \delta x_{ik}} e^{-y_i \gamma} = 0 \tag{17}$$

$$\iff \gamma = \frac{1}{2} \log \frac{\sum_{i:y_i=1} e^{-y_i f(x_i)} e^{-y_i \delta x_{ik}}}{\sum_{i:y_i=-1} e^{-y_i f(x_i)} e^{-y_i \delta x_{ik}}} \tag{18}$$

$$\frac{\partial L_k}{\partial \delta} = -\sum_i y_i x_{ik} e^{-y_i f(x_i)} e^{-y_i \delta x_{ik}} e^{-y_i \gamma} = 0 \tag{19}$$

$$\iff \delta = \frac{1}{2} \log \frac{\sum_{i:y_i x_{ik}=1} e^{-y_i f(x_i)} e^{-y_i \gamma}}{\sum_{i:y_i x_{ik}=-1} e^{-y_i f(x_i)} e^{-y_i \gamma}} \tag{20}$$

Now, assume that $x_{ij} \in \{0, +1\}$. Then

$$\delta = -\gamma + \frac{1}{2} \log \frac{W^+}{W^-}. \tag{21}$$

Now define $B^+ = \sum_{i:y_i=1} e^{-y_i f(x_i)}$, $B^- = \sum_{i:y_i=-1} e^{-y_i f(x_i)}$. Recall that $W^+ = \sum_{i:y_i x_{ik}=1} e^{-y_i f(x_i)}$ and $W^- = \sum_{i:y_i x_{ik}=-1} e^{-y_i f(x_i)}$. We simplify γ to

$$\gamma = \frac{1}{2} \log \frac{B^+ + (e^{-\delta} - 1) W^+}{B^- + (e^{\delta} - 1) W^-} \tag{22}$$

Define $N^+ = \sum_{i:y_i=1, x_{ik}=0} e^{-y_i f(x_i)}$, $N^+ = \sum_{i:y_i=-1, x_{ik}=0} e^{-y_i f(x_i)}$. Substituting for δ, we get

$$\gamma = \frac{1}{2} \log \frac{N^+ + (e^{\gamma} \sqrt{W^+ W^-})}{N^- + (e^{-\gamma} \sqrt{W^+ W^-})} \tag{23}$$

We cannot solve for γ directly. Instead, we use Newton's method. Let $y = f(\gamma)$, where

$$f(\gamma) = \gamma - \frac{1}{2} \log \frac{N^+ + e^{\gamma} \sqrt{W^+ W^-}}{N^- + e^{-\gamma} \sqrt{W^+ W^-}}. \tag{24}$$

Then,

$$f'(\gamma) = 1 - \frac{1}{2}\left(\frac{e^{\gamma}\sqrt{W^{+}W^{-}}}{N^{+} + e^{\gamma}\sqrt{W^{+}W^{-}}} + \frac{e^{-\gamma}\sqrt{W^{+}W^{-}}}{N^{-} + e^{-\gamma}\sqrt{W^{+}W^{-}}}\right) \tag{25}$$

Newton's method calculates the $n+1^{\text{st}}$ approximation to γ as

$$\gamma_{n+1} = \gamma_n - \frac{f(\gamma_n)}{f'(\gamma_n)}. \tag{26}$$

We have not been able to show that Newton's method will converge. It can be shown that f'' has one inflection point and that if we add ϵ to N^{+} and N^{-}, then $f'(\gamma) > \frac{\epsilon}{1+\epsilon}$. Visual inspection of example plots (see figure 1) over a normal range of values show cases that will converge.

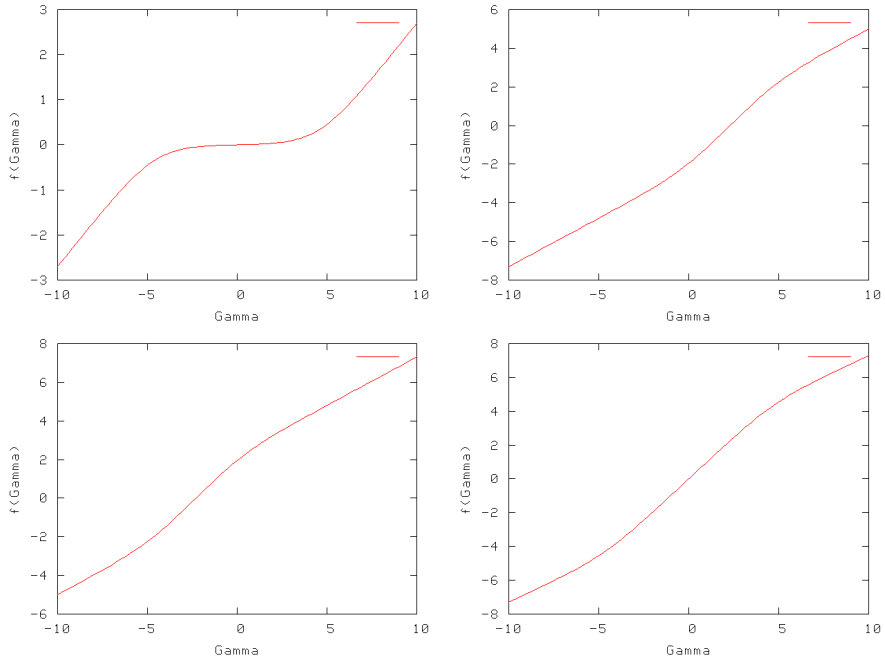

Figure 1: Example plots of $f(\gamma)$.

We can solve for the bias term, γ, via Newton's method, then calculate the weight, δ, as function of γ. We make this calculation for each feature and choose the one that yields the largest decrease in the loss functional,

$$L(f) - L_k(f) = \sum_{i=1}^{n} e^{-y_i f(x_i)}(1 - e^{-y_i x_{ik}\delta} e^{-y_i \gamma}). \tag{27}$$

This proceedure has an advantage over others described in this paper. It optimizes the weight and bias term jointly. Thus, a feature is chosen based on the real decrease in the loss functional that it can provide. The other techniques use an approximation of the loss delta. However, it is not clear that this provides significant improvement over other techniques. Schapire and Singer noted that learning separate occurrence and non-occurrence weights yielded little improvement over learning only occurrence weights. This technique may give little additional improvement.

References

[1] Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39:135–168, 2000.