Improved Boosting Algorithms Using Confidence-rated Predictions

Robert Schapire Yoram Singer

(Summery by feng jiao)

This paper is mainly on the algorithm of boosting, which is one of the most important recent developments in the classification methodology. Boosting works by sequentially applying a classification algorithm to reweighted versions of the training data, and then taking a weighted majority vote of the sequence of classifiers thus produced.

This paper has main three parts. The first parts refer to how to improve boosting algorithm using a theorem which proved in the paper. And the second part mainly introduces several multiclass algorithms and the third part give corresponding experiments to compare them.

Part I Improve boosting algorithm.

This paper gives a simply analysis of Adaboost algorithm, especially on how to achieve the upper bound of the training error. The concluding is listed below.(The detailed prove can see the paper)

$$\frac{1}{m}\sum_i [\![H(x_i) \neq y_i]\!] \leq \prod_t Z_t$$

The left of this above inequation is the training error, and the right is the production of the normalization factor of each bound.

It is gives us a way to minimize the training error, which normally is a good criterion to judge a classification algorithm. That is to be greedily minimize the bound given in the theorem by **minimizing Z_t at each round of boosting**. It applies this idea both in the choice of α_t and as a general criterion for the choice of weak hypothesis of h_t.

First see how it use this idea to choose α_t

$$Z = \sum_i D(i) e^{-\alpha u_i} \leq \sum_i D(i) \left(\frac{1+u_i}{2} e^{-\alpha} + \frac{1-u_i}{2} e^{\alpha} \right)$$

Where $u_i = y_i h_t(x_i)$

When $\alpha = \frac{1}{2}\ln\left(\frac{1+r}{1-r}\right)$ where $r = \sum_i D(i) u_i$

Z is minimized with the value of $Z \leq \sqrt{1-r^2}$

The same theorem had been proved by Freund and Scapire(1997).

The paper also give a general numerical method for exactly minimizing Z with respect to α (the detailed can see in paper) and get the following conclusion

1. Normally there exists a unique choice of α_t which minimizes Z_t

2. For that choice of α_t, it has that $E_{i\sim D_{t+1}[y_i h_t(x_i)]} = 0$

Meanwhile it gives an analytic method for weak hypotheses that abstain. "abstain" means that the weak hypothesis don't give any prediction by taking the value of 0, instead of –1 or +1.
And in this case, it can be proved when

$$\alpha = \frac{1}{2}\ln\left(\frac{W_+}{W_-}\right), \text{ where } W_b = \sum_{i:u_i=b} D(i) \quad b \in \{-1, 0, +1\}$$

Z is minimized with $Z = W_0 + 2\sqrt{W_- W_+}$

We have showed how to use that idea to choose α_t, next we show how to apply that idea to guide in the design of weak learning algorithms which can be combined more powerfully with boosting.
Here it only considers weak hypotheses which make their predictions based on a partitioning of the domain. X. To be more specific, each such weak hypothesis is associated with a partition of X into disjoint blocks X$_1$, X$_2$ …X$_n$ which cover all of X and for which $h(x) = h(x')$ for $x, x' \in X_j$. In other words, h's prediction depends on only on which block X_j a given instance falls into.
In this case, the paper concludes that

If $c_j = h(x)$ when $c_j = \frac{1}{2}\ln\left(\frac{W_+^j}{W_-^j}\right)$ where $W_b^j = \sum_{i:x_i \in X_j \wedge y_i = b} D(i) = \Pr_{i\sim D}\left[x_i \in X_j \wedge y_i = b\right]$

Z is minimized and $Z = 2\sum_j \sqrt{W_+^j W_-^j}$

But sometime W_+^j and W_-^j maybe very small or even zero, which will make c_j will be very large or infinite in magnitude. In practice, such large predictions may cause numerical problems.
We suggest using instead the "smoothed" values

$$c_j = \frac{1}{2}\ln\left(\frac{W_+^j + \varepsilon}{W_-^j + \varepsilon}\right)$$

where $Z \le 2\sum_j \sqrt{W_-^j W_+^j} + \sqrt{2N\varepsilon}$

we can see if we chose $\varepsilon \ll \frac{1}{2N}$, Z will not be greatly degraded. In the actual experiment, it uses ε on the order of 1/*m*, where *m* is the number of training examples.

Part II Multiclass algorithm

This paper also introduce several method how to apply boosting algorithm into multiclass problem. The detailed algorithm can be found in the paper

1 Using **Hamming Loss** for multiclass problems

Using Mapping $H : X \to 2^{y}$

Hamming Loss $\frac{1}{k} E_{(x,Y)\sim D[|h(x)\Delta Y|]}$ where Δ denotes symmetric difference.

and its Domain-partitioning weak hypotheses is as the below

When $c_{jl} = \frac{1}{2} \ln\left(\frac{W_{+}^{jl}}{W_{-}^{jl}}\right)$

Z is minimized and $Z = 2\sum_{j}\sum_{l}\sqrt{W_{+}^{jl}W_{-}^{jl}}$

2 Using **Output Coding** for multiclass problems

Using Mapping $H : X \to 2^{y}$ is the simplest and most obvious way. However, it maybe a more effective to use a more sophisticated mapping. It can define a one-to-one mapping $\lambda : Y \to 2^{Y'}$.

λ maps to the subsets of an unspecified label set Y' which need not to be the same as Y.

3 Using **Ranking loss** for multiclass problems

It only care about the crucial pars l_0, l_1 and $l_0 \notin Y, l_1 \in Y$. When f misorders a crucial pair if $f(x,l_1) \le f(x,l_0)$.

The ranking loss algorithm is to minimize the expected fraction of crucial pairs which are misordered.

The upper bound of training error is $rloss(f) \le \prod_{t=1}^{T} Z_t$

Part III Experiment.

The paper run two sets of experiment.

Set1. Compares the algorithms on a set of learning benchmark problems from the UCI repository.

Set2. Comparison on a large text categorization task.

And for multiclass problems, it compared three of the boosting algorithms

Discrete AdaBoost.MH

Real AdaBoost.MH

Discrete AdaBoost.MR

Set1

Experiment 1

Comparison of discrete Adaboost.MH and discrete Adaboost.MR on 11 multiclass benchmark problems from the UCI repository. Each point in each scatter plot show the error rate of the two competing algorithms on a single benchmark

Conclusion

Generally quite evenly matched with a possible slight advantage to AdaBoost.MH

Set1

Experiment 2

Comparison of discrete and real AdaBoost.MH on 26 binary and multiclass benchmark problems from the UCI repository.

Conclusion

Real version(with confidences) is overall more effective at driving down the training error and also test error rate. But after 1000 rounds, these differences largely disappear.

Set1

Experiment 3

Comparison of discrete and real Adaboost.MH on 16 binary problems from UCI repository

Set1

Experiment 4

Comparison of discrete AdaBoost.MR, discrete AdaBoost.MH and real Adaboost.MH on 11 multiclass problem from UCI repository

Conclusion of the above two

The potential for improvement of Adaboost with real-valued prediction seems to be greatest on larger problems

Set 2

Experiment 5

Large text-categorization problem with Six class and 142,727 train and 66,973 test documents.

Conclusion

Real Adaboost.MH dramatically outperform the other two methods

Discrete Adaboost.MH seems to consistently outperform discrete Adaboost.MR