The mathematics of boosting

Philippe Fullsack

March 10 2020

1 The boosting idea

Boosting is an idea proposed by Robert Schapire and Yoav Freund. It evolved, as good ideas often do, from a simple concept into a family of powerful algorithms. In this section, we present a basic idea and show how it allows us to improve the quality of prediction of pretty much any algorithm. In the next section, we will further improve on this idea. Suppose we some access to some data. We have seen in our presentation of the Random Forest algorithm how we can re-sample data with replacement (this is called bootstrap sampling) and use this method to create synthetic data-sets to generate many models (e.g. CART trees) and elicit a good response by averaging or voting. In bootstrap sample, the basic idea was that all records were exchangeable and had the exact same weight. In boosting, we proceed differently. We are going to play with 're-weighting of the data'. By this, we mean that the algorithm will attach a weight (importance) to each data record, and it will do this so as to improve the performance of classification. Assume that we have a baseline algorithm that does 'a bit better' than chance guessing, hence has a probability of error $p < 1/2$. Our goal is to embed this baseline (or weak) learner into an 'boosted' algorithm that has a better performance. Of course, the key reason why we might be able to do this is that we have access to labels and can therefore validate the predictions of the weak learner.

Here is the first idea of Schapire-Freund (pre-AdaBoost, if you wish to call it so). Note $D_1[x]$ be the original sampling probability on x. This gives a uniform weight $1/|S|$ to all records/data points. Applying our base learner produces a first hypothesis (tentative prediction of classes), that we note h_1. This makes mistakes on incorrect samples (set S_I) and is correct on the subset S_C. Hence verification of labels (which is possible because we build our model from a labelled training set, as we usually do in supervised learning algorithms) leads to a first partition of of S. The nominal/baseline algorithm has a probability of error equal to $p = \frac{|S_I|}{|S|}$, and the probability of making no error is $1 - p = \frac{|S_C|}{|S|}$. Now we want in the next round the algorithm to re-weight the data point so as to *balance* the hard and easy examples, so to speak. This is done by modifying the original uniform sampling distribution on data points so as to give equal weights to the sample points that have been correctly and incorrectly classified in the first round of iteration. Hence, we want to reduce the probability of correct data point from $1 - p$ to $1/2$. This 'un-boosting' (by a value $\gamma = 1/2 - p$) of correctly classified examples will by the same token boost the hard example. Hence, chances are that the algorithm will spend more iterations working at classifying the hard examples. So we create a distribution $D_2(x)$ that is $D_2(x) = D_1(x)/(2(1-p))$ (reduced sampling probability) on S_C and $D_2(x) = D_1(x)/(2p)$ (boosted sampling probability) on S_I. This creates a hypothesis h_2 with accuracy $1 - p$ on the D_2 distribution. Now, we describe how the algorithm proceeds in the second round. It picks data (examples) with the re-weighted distribution $D_2(x)$. Then the same algorithm (base learner) $\mathcal{A}$ is applied to this sample. Then $\mathcal{A}$ is fed to those sample in S where the two hypotheses disagree ($h_1(x) \neq h_2(x)$). Over this set, the sampling distribution

is taken to be uniform. This gives a third hypothesis $h_3(x)$, that has the baseline accuracy $1-p$ over this set. Finally, we take a majority vote between the 3 hypotheses. If a set of covariates, x, is such that $h_1(x) = h_2(x)$, the algorithm returns $h(x) = h_1(x) = h_2(x)$, or else, returns $h_3(x)$.

Let us analyze this algorithm. The following lemma states the result.

Lemma 1. *The accuracy Q of the boosted algorithm is:*

$$Q = 3p^2 - 2p^3 \tag{1}$$

Proof S is partitioned into 4 regions, $S_{CC}, S_{CI}, S_{IC}, S_{II}$ by the two rounds of this algorithm, according to the status of the classification of data points by hypotheses h_1, h_2 during the first and second rounds. Let us note $p_{CC}, p_{CI}, p_{IC}, p_{II}$ the probabilities that data points belong to these regions.

As no mistake is made on S_{CC}, and all samples are incorrect in S_{II}, and $1-p$ samples are correct on S_{CI} or S_{IC}, the error rate is:

$$Q = p(p_{CI} + p_{IC}) + p_{II}$$

Let us calculate each term. First, $p_{CI} = 2(1-p)\gamma$. Second, by definition of h_2:

$$D_2(S_{II}) = p - \gamma$$

so that $p_{II} = 2p(p-\gamma)$. Third,by construction of D_2, $D_2(S_{IC}) = 1/2 - (p-\gamma)$. so that $p_{IC} = 2p(1/2 - (p-\gamma))$. Hence the overall error rate is:

$$p(2(1-p)\gamma + 2p(1/2 - p + \gamma))) + 2p(p-\gamma)$$

If we know develop and substitute the expression $\gamma = 1/2 - p$, we find:

$$Q = p(2(1-p)(1/2-p) + 2p(1/2 - p + 1/2 - p)) + 2p(p - 1/2 + p)$$

or $Q = 3p^2 - 2p^3$ □

Remark For the second round of the algorithm, a naive idea would have been to only look at examples in S that are misclassified by h_1, and learn from this subset only. I let you establish the reason why this idea cannot work.

Remark If h_1, h_2, h_3 were independent classifiers, all with error rate p, then the error rate of the majority vote would be:

$$p^3 + 3p^2(1-p) = 3p^2 - 2p^3$$

This formula is obtained by reasoning that 2 hypotheses at least must be incorrect in order for the majority vote to be incorrect. We note that the lemma matches this result. So, assuming for example that $p = 1/4$, the boosted algorithm makes now a much lower error of $5/32$. Re-weighting has acted so as to force hypotheses to be as useful as if they were independent.

2 Adaboost

The previous section has revealed that weak learners can be boosted. This is an important finding. Let's move now to a cleaner and more efficient algorithm to do the boosting: Adaboost. Note that Adaboost comes into different flavors. We will comment on that later.

For each round or iteration t, let f_t be the (approximate) minimizer of the loss:

$$l_t = \sum_{i=1}^{n} D_t(i) 1(f_t(x_i) \neq y_i)$$

Let $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$, and consider a modification of the current (boosted) classifier by an additive perturbation:

$$F_t = F_{t-1} + \alpha_t f_t$$

This implies that the decision rule of Adaboost is a properly weighted linear combination of weak predictor functions fitted onto iteratively re-weighted sample records.

Now comes the time to update the weights. For correctly classified data points, weights are decreased:

$$D_{t+1}(i) = e^{-\alpha_t} \frac{D_t(i)}{Z_t}$$

while for incorrectly classified data points, weights are increased:

$$D_{t+1}(i) = e^{\alpha_t} \frac{D_t(i)}{Z_t}$$

The partition function Z_t normalizes D_t so that it sums to one. This turns out to be

$$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)} \tag{2}$$

Theorem 2 (Schapire-Freund)**.** *The probability of making error at stage T of Adaboost is upper-bounded by:*

$$Q = \hat{\mathbb{P}}(YF_T(x) \leq 0) \leq E_T$$

with $E_T = \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)}$

Proof

The key observation is that the event $YF_T(X) \leq 0$ is equivalent to $e^{-YF_T(X)} \geq 1$. Therefore

$$\hat{\mathbb{P}}(YF_T(x) \leq \hat{\mathbb{E}}(e^{-YF_T(X)}) = E_T$$

But each point i contributes $\frac{1}{n}e^{-y_i F_T(x_i)}$ to the expectation, therefore

$$E_T = \frac{1}{n}\sum_{i=1}^{n} e^{-y_i \sum_{t=1}^{T} \alpha_t f_t(x_i)}$$

$$E_T = \frac{1}{n}\sum_{i=1}^{n}\prod_{t=1}^{T} e^{-y_i \alpha_t f_t(x_i)}$$

Now, note that regardless of the sign of y_i, $D_{t+1} = \frac{D_t(i)}{Z_t}e^{-y_i\alpha_t f_t(x_i)}$, which implies that:

$$E_T = \frac{1}{n}\sum_{i=1}^{n}\prod_{t=1}^{T} Z_t \frac{D_{t+1}(i)}{D_t(i)}$$

All D_t cancel each other, except:

$$E_T = \frac{1}{n} \sum_{i=1}^{n} \prod_{t=1}^{T} e^{-y_i \alpha_t f_t(x_i)}$$

As $\sum_i D_{T+1}(i) = 1$ and $D_1(i) = 1/n$, we get:

$$E_T = \prod_{t=1}^{T} Z_t$$

How should the weights α_t be chosen? Let us analyze the partition function Z_t. All incorrectly classified points i (such that $y_i \neq f_t(x_i)$ contribute $e^{\alpha_t} D_t(i)$ and all correctly classified points i (such that $y_i = f_t(x_i)$ contribute $e^{-\alpha_t} D_t(i)$. If we note $\epsilon_t = \sum_{i, y_i \neq f_t(x_i)} D_t(i)$, we have:

$$Z_t = (1 - \epsilon_t) e^{-\alpha_t} + \epsilon_t e^{\alpha_t}$$

The idea is to minimize this expression in α_t, which is easily found by zeroing the derivative. This results in $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$, which in turns, gives:

$$Z_t = 2\sqrt{\epsilon_t (1 - \epsilon_t)}$$

□

Corollary 3. *If we can uniformly bound the weak learner error with some* $0 < \gamma < 1/2$

$$\forall t, \epsilon_t \leq \frac{1}{2} - \gamma$$

then

$$Q \leq (1 - 4\gamma^2)^{T/2}$$

Remark Therefore, Adaboost decreases the error as in a geometric progression. The ratio of this progression is dictated by the margin that we can save/maintain uniformly on the prediction error of the weak learner, i.e. $1/2 - p$. However, I am not too sure how we would be able to prove a uniform bound. The weak learner would have to provide error guarantees that do not depend on the sample. This is a point to examine further.

3 Extension

It does not take too much sweating to arrive at useful generalizations of Adaboost. One popular extension is Gradient boosting (XGBoost). Gradient Boosting is a generic algorithm that finds approximate solutions to additive modeling problem. AdaBoost is a special case with a particular (exponential) loss function. Gradient boosting is much more flexible and accepts a wide range of loss functions.

On the other hand, AdaBoost is a much more intuitive perspective its implementation does not require gradients. We have seen how it instead use the very intuitive idea of reweighting of training samples based on classifications from previous learners.

4 Conclusion

This document is not entirely complete. I have other topics in mind that I will paste here when I can. For the meantime, I hope to have left you with the Take Home message that Boosting is very different from Bagging, and quite elegant.

You will also understand why Robert Schapire and Yoav Freund have received the Godel Prize in 2003 for the invention of this beautiful algo rithm.

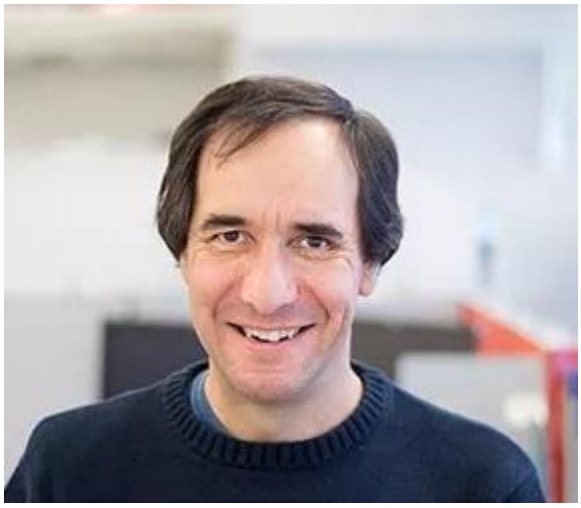

Figure 1: Robert Schapire.

Figure 2: Yoav Freund.

References