AdaBoost Summary

Vafa Behnam Roudsari

February 2020

1 Introduction

Paper: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting (Freund, Y., Schapire, R) [1]

Two weeks ago, it was emphasized the major contribution of A Theory of the Learnable [2] was its new framework of analysis as opposed to a technique that generates algorithms. The actual algorithms were secondary to the new technique of appraising what can be learned.

One of the key insights of this paper is that sometimes it is fruitful to start from a new framework, answer some theoretical question for that framework, and then investigate whether an algorithm inspired by said theory performs well practically or in other frameworks.

We present the definitions of Strong and Weak Learnability for some concept class C, where we allow any arbitrary distribution D over that concept class and.

We present the definitions of Strong and Weak Learnability. Let H be the hypothesis class, and D be an arbitrary distribution over the examples (x, y),

Strong Learnability: $\forall \epsilon, \delta > 0, \exists h \in H$ s.t. $Pr_{(x,y)}[|h(x) - y| > \epsilon] < \delta$

Weak Learnability: $\exists \gamma > 0, \forall \delta > 0, \exists h \in H$ s.t. $Pr_{(x,y)}[|h(x) - y| > \frac{1}{2} - \gamma] < \delta$

The difference is that, rather than having an arbitrarily small error ϵ, we can select our error γ as long as its better than random. Note the definition of weak learning is subtly quite strong itself; γ cannot depend on δ or h, so we need one constant to work for all possible probabilities δ or the ideal hypothesis h.

2 The AdaBoost Algorithm

Algorithm AdaBoost
Input: sequence of N labeled examples $\langle (x_1, y_1), ..., (x_N, y_N) \rangle$
distribution D over the N examples
weak learning algorithm **WeakLearn**
integer T specifying number of iterations
Initialize the weight vector: $w_i^1 = D(i)$ for $i = 1, ..., N$.
Do for $t = 1, 2, ..., T$

1. Set

$$\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}$$

2. Call **WeakLearn**, providing it with the distribution $\mathbf{p}^t$; get back a hypothesis $h_t: X \to [0, 1]$.
3. Calculate the error of h_t: $\varepsilon_t = \sum_{i=1}^N p_i^t |h_t(x_i) - y_i|$.
4. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$.
5. Set the new weights vector to be

$$w_i^{t+1} = w_i^t \beta_t^{1 - |h_t(x_i) - y_i|}$$

Output the hypothesis

$$h_f(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^T (\log 1/\beta_t)\, h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \log 1/\beta_t \\ 0 & \text{otherwise.} \end{cases}$$

The key player here is β_t, which plays two roles in the algorithm. Its first purpose is to ensure that in the next iteration, we target the points that we have poorly classified in the current iteration (line 5). Its second role is to, in the final output, emphasize hypotheses which have low training error. The choice of $\beta_t = \epsilon_t(1 - \epsilon_t)$ is motivated by tightening the bounds on training error, as we'll see in Theorem 1.

Furthermore, notice that we need only one WeakLearner algorithm, and the only thing that needs to be varied in every iteration is the "data" WeakLearner sees in line 2. In practice, how do we manipulate the "data" WeakLearner sees? For some techniques we can do this directly if WeakLearner can naturally incorporate some notion of importance of data points. [1] But we can also use AdaBoost for a much larger class of algorithms indirectly; we do this by simply resampling with replacement our training data, with the probability of being selected proportional to the weights, and simply feed our resampled data into WeakLearner.

Hence, this makes WeakLearner even more restrictive than at first glance, as we require our WeakLearner to do better than random on every possible "bootstrapped" sample.

[1] For example, if WeakLearner is based on minimizing a loss function of the form $\sum ||(y - X\beta)||^2$ (ordinary least squares), then we can instead use the loss function $\sum ||W(y - X\beta)||^2$ where W is not the identity matrix (weighted least squares)

3 Training Error Analysis of AdaBoost – Theorem 6

We now analyze how well AdaBoost does on its training samples.

Theorem 1. *Let h_f be the output of AdaBoost, and $\gamma_t = \frac{1}{2} - \sum_{i=1}^{N} |h_t(x_i) - y_i|$. Then*

$$\frac{\sum_i^N |h_f(x_i) - y_i|}{N} \leq \exp(-2\sum_t^T \gamma_t^2)$$

(Theorem 6 in Paper)

Corollary 1. *If $\gamma \geq \gamma_t \forall t$, then*

$$\frac{\sum_i^N |h_f(x_i) - y_i|}{N} \leq \exp(-2T\gamma^2)$$

Proof. Noting $\forall \alpha \geq 0$ and $r \in [0,1], \alpha^r \leq 1 - (1-\alpha)r$, this implies for the update rule on line 5, for $t = 1, ..., T+1$

$$\sum_i^N w_i^{t+1} = \sum_i^N w_i^t \beta_t^{1-|h_t(x_i)-y_i|} \leq \sum_i^N w_i^t (1 - (1-\beta_t)(1 - |h_t(x_i) - y_i|))$$

Letting $\epsilon_t = \sum_i^N |h_t(x_i) - y_i|)$ and seeking an upper bound for $t = T+1$,

$$\sum_i^N w_i^{T+1} \leq (\sum_i^N w_i^T)(1 - (1-\beta_t)(1-\epsilon_t))$$

$$\leq (\sum_i^N w_i^{T-1}) \prod_{t=T-1}^{T} (1 - (1-\beta_t)(1-\epsilon_t))$$

$$\leq ... \leq (\sum_i^N w_i^{1-1}) \prod_{t=1}^{T} (1 - (1-\beta_t)(1-\epsilon_t))$$

$$= \prod_{t=1}^{T} (1 - (1-\beta_t)(1-\epsilon_t))$$

The next line is saying h_f, the final output of AdaBoost, fails to classify a point correctly if and only if half or more of our weak hypotheses, adjusted for the weights of each hypothesis, have classified incorrectly.

$$\prod_t^T \beta_t^{-|h_t(x_i)-y_i|} \geq (\prod_t^T \beta_t)^{-\frac{1}{2}}$$

By definition of the weights in line 5,

$$w_i^{T+1} = w_i^T \beta_t^{1-|h_t(x_i)-y_i|} = ... = w_i^1 \prod_t^T \beta_t^{1-|h_t(x_i)-y_i|}$$

Finally, combining these equations together,

$$\sum_i^N w_i^{T+1} \geq \sum_{i|h_f(x_i)\neq y_i} w_i^{T+1} \geq (\sum_{i|h_f(x_i)\neq y_i} w_i^1)(\prod_t^T \beta_t)^{\frac{1}{2}}$$

$$= \sum_i^N w_i^1 |h_f(x_i) - y_i|(\prod_t^T \beta_t)^{\frac{1}{2}}$$

Now, exploiting the two-sided inequalities we have for w_i^{T+1}, and if our initial weights are uniform, $w_i^1 = 1/N$, then we have a bound for the training error. Let the training error be $\epsilon = \sum_i^N \frac{1}{N} |h_f(x_i) - y_i|$,

$$\epsilon \leq \frac{\sum_i^N w_i^{T+1}}{\prod_{t=1}^T \sqrt{\beta}_t} \leq \prod_{t=1}^T \frac{(1-(1-\beta_t)(1-\epsilon_t))}{\sqrt{\beta}_t}$$

Since this is an upper bound, the inequality is preserved if we differentiate the RHS, find the value of B_t that will yield the minimum (which is $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$, which is why this value was selected in the algorithm) and plug it back into the RHS. This gives us

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

Since $\gamma_t = 1 - \epsilon_t$,

$$\epsilon \leq \prod_{t=1}^T \sqrt{\gamma_t(1-\gamma_t)} = \exp(\frac{1}{2}\log(\prod_{t=1}^T 1 - 4\gamma_t^2)) = \exp(\sum_t^T \frac{1}{2}\log(1-4\gamma_t^2))$$

Since $1 - x \leq e^{-x}$, $\log(1-x) \leq -x$ and hence,

$$\leq \exp(-2\sum_t^T \gamma_t^2)$$

□

(By the way, there is no need for the Kullback-Leibler divergence stuff at all (pg 127, equation (21) of the paper), but I only realized this after proving $KL(\frac{1}{2}||\frac{1}{2} - \gamma_t) = \exp(\frac{1}{2}\log(\prod_{t=1}^T 1 - 4\gamma_t^2))$. Woops! I provided a proof in the appendix if anyone is interested.)

Let us take stock of this upper bound. First, we can clearly see that adding new learners T will never hurt training error. Second, the better each individual learner γ_t is, the lower the training error. Third, it appears even learners worse than random will still improve training error, since γ_t is squared.

4 Generalization Error & VC dimension of AdaBoost – Theorem 8

Now that we have favorable training error guarantees, how can we ensure that AdaBoost does not simply overfit to our training sample? One can conceive of the scenario where, if we allow one learner for each data point, we can simply set w_t^i to 1 if $i = t$ for $i, t = 1, ..., N(= T)$, and hence we can trivially learn any training dataset. More generally, we will see the number of learners T is a knob controlling the variance-bias tradeoff (or the degree of overfitting vs. underfitting).

Let $\epsilon(h) = Pr_{(x,y)}|h(x) - y|$ and $\hat{\epsilon}(h) = \sum_i^N \frac{|h(x_i)-y_i|}{N}$. The latter measures performance on our finite training sample, whereas the former measures the performance on an arbitrarily drawn point from the true joint distribution of (x, y). Therefore, remembering that the output of AdaBoost is governed by the function h_f, our target is to find some bound satisfying, for every $\delta > 0$,

$$Pr_{(x_i,y_i)_{i=1}^N}[|\hat{\epsilon}(h_f) - \epsilon(h_f)| > \mathcal{O}(T^{m_1}, d^{m_2}, \frac{1}{\delta})] \leq \delta$$

where m_1, m_2 are constants, N is the size of the sample, and d is a measure of the concept class.

It is important to note this measure by itself says nothing of good training performance, i.e. our goal for this chapter is to simply minimize the distance between performance on our sample and performance on future unseen samples; for example, we would do well on this measure if we have a lousy hypothesis performing very badly on both our training set and on future examples.

The authors, rather than trying to bound the particular function h_f directly, use already existing results which are far more general. We introduce notation for this end. Let $\theta(x) = 1$ for $x \geq 0$ and $\theta(x) = 0$ for $x < 0$. One natural choice of a more general hypothesis space would be the set

$$\Theta_T(H) = \{\theta(\textstyle\sum_t^T a_t(h_t - b))|b, a_1, ..., a_T \in \mathbb{R}, h_1, ..., h_T \in H\}$$

First, notice the final hypothesis outputted by AdaBoost, h_f, clearly belongs to this set (simply set $b = 1$ and $a_j = log(\frac{1}{\beta_j})$ for $j = 1, ..., T$). Hence any worst-case guarantees for the far larger set $\Theta_T(H)$ would be applicable to the AdaBoost output h_f. Second, this hypothesis set contains at least infinitely many members. Hence, upper bounding by the cardinality of this space this would be useless. We turn to the framework introduced last week, that of "capacity" or VC-dimension, to see if it can provide useful bounds on generalization. To this end, we would hope that the VC-dimension of AdaBoosting a finite hypothesis class is finite itself, which it is.

Theorem 2. *Suppose d is the VC-dimension of the WeakLearner algorithm. Then the VC-dimension of $\Theta_T(H)$ is $d_\Theta = 2(d + 1)(T + 1) \log 2(e(T + 1))$*

(Theorem 8 in paper)

Proof. The cleverness of the proof arises from the fact that it realises compositions of functions can easily be framed as feedforward neural networks, and hence the full power of the deep learning literature can be used here.

They utilize a result of Baum and Haussler [4] that states the number of realizable functions is upper bounded by $(Nem/l)^l$ where N is the number of nodes in the network, l is the sum of the VC-dimensions of the class of functions associated with each node, and $m \geq d$ is a free variable. Upper bounding the number of realizable functions would bound the VC-dimension.

A natural modeling of this problem would be a two-layer network, with the first layer consisting of the weak learners $h_1, ..., h_T$ and the second the output of the threshold function $\theta(\sum_t^T a_t(h_t - b))$. We have T nodes in the first layer and one node as the output node in the second layer, so $N = T + 1$.

Second, we have T first layer nodes, each with at most d VC-dimension. Then, we have one output node, a linear threshold function, and as linear threshold functions over $\mathbb{R}^T$ have a VC-dimension of $(T + 1)$ by result of Wenocur et Dudley [5], $l = Td + (T + 1) < (T + 1)(d + 1)$.

Hence we can now apply the theorem of Baum and Haussler [4] for these values of l and N to yield $((T + 1)em/(T + 1)(d + 1))^{(T+1)(d+1)}$. Letting $m = 2(T + 1)(d + 1)log_2(e(T + 1))$, we have the upper bound of realizable functions be less than 2^m, which implies the VC-dimension is at most m.

□

5 Generalization Error Analysis of AdaBoost – Theorem 7

Now with a finite VC-dimension, we can use general results connecting VC-dimension with generalization error. One such result is from Vapnik's Estimation of Dependences in Empirical Data (Theorem 6.7) [3]. Applying it to our current context,

Theorem 3.

$$Pr_{(x_i,y_i)_{i=1}^N}[\sup_{h \in \Theta_T(H)} |\hat{\epsilon}(h) - \epsilon(h)| > 2\sqrt{\frac{d_\Theta(ln2N/d_\Theta + 1) + ln9/\delta}{N}}] \leq \delta$$

where d_Θ is the VC dimension of the AdaBoost class (Theorem 2) and a function of the original hypothesis class d and number of learners T.

(Theorem 7 in Paper)

Therefore, to guard against overfitting, we need to either increase number of samples N, decrease the number of learners T, or decrease the VC-dimension of the concept class d. Doing the latter two however will generally increase training error.

References

[1] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, p. 119–139, Aug. 1997. [Online]. Available: https://doi.org/10.1006/jcss.1997.1504

[2] L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, pp. 1134–1142, 1984.

[3] V. Vapnik, *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Berlin, Heidelberg: Springer-Verlag, 1982.

[4] E. B. Baum and D. Haussler, "What size net gives valid generalization?" in *Proceedings of the 1st International Conference on Neural Information Processing Systems*, ser. NIPS'88. Cambridge, MA, USA: MIT Press, 1988, p. 81–90.

[5] R. S. Wenocur and R. M. Dudley, "Some special vapnik-chervonenkis classes," *Discrete Math.*, vol. 33, no. 3, p. 313–318, Jan. 1981. [Online]. Available: https://doi.org/10.1016/0012-365X(81)90274-0