Computational Learning Theory - Hilary Term 2018
4 : Boosting

Lecturer: Varun Kanade

1 Weak Learnability

Let us revisit the definition of PAC-learning. The requirements for a concept class to be PAC-learnable are quite stringent. The learning algorithm has to work for all target concepts in the class, for all input distributions, and for any setting of accuracy (ϵ) and confidence (δ) parameters. It is worthwhile considering what happens when we relax some of these requirements. We have already seen that fixing the confidence paramter to be a constant, *e.g.,* $\delta = 1/4$, leaves the notion of PAC-learnability unchanged. On the other hand, if we only require the learning algorithm to succeed with respect to certain input distributions, then PAC-learning is possible for concept classes that are provably not PAC-learnable in the distribution-free sense, *i.e.,* algorithms that have to work with respect to all distributions.[1] For example, the concept class of convex subsets of $[0, 1]^2$ is not (distribution-free) PAC-learnable, but when restricted to the uniform distribution over $[0, 1]^2$, it is learnable. In this lecture, we focus on the accuracy parameter, ϵ. The problem of learning is trivial if $\epsilon \geq 1/2$ as we can make a random prediction on an input $x \in X$ and achieve an error of $1/2$.[2] The question we are interested in is what happens when $\epsilon = 1/2 - \gamma$, for some $\gamma > 0$? For example, is it possible to learn some concept classe up to error $1/4$, but not to an arbitrarily small ϵ?

Surprisingly, the answer to this question is no, *i.e.,* if we can learn a concept class up to error at most $1/2 - \gamma$, then we can learn this class up to error bounded by any $\epsilon > 0$. This method is known as *boosting*, as we take a "weak learning" algorithm and boost it to produce a "strong learning" algorithm.

γ-Weak Learner

Let us define the notion of weak learning formally. We will let the parameter γ for weak learning be a function of the instance size n, and the representation size of the target concept, $\mathrm{size}(c)$.

Definition 1 (γ-Weak Learning). *For $\gamma(\cdot, \cdot)$ with $\gamma > 0$, we say that L is a γ-weak PAC learning algorithm for concept class C using hypothesis class H, if for any $n \geq 0$, any $c \in C_n$, any D over X_n, and $0 < \delta < 1/2$, L given access to $\mathsf{EX}(c, D)$ and inputs $\mathrm{size}(c)$, δ and γ, outputs $h \in H_n$ that with probability at least $1 - \delta$, satisfies, $\mathrm{err}(h) \leq \frac{1}{2} - \gamma(n, \mathrm{size}(c))$.*

We say that L is an efficient γ-weak PAC learner if H is polynomially evaluatable, $1/\gamma(n, \mathrm{size}(c))$ is bounded by some polynomial in n and $\mathrm{size}(c)$, and the running time of L is polynomial in n, $1/\delta$, and $\mathrm{size}(c)$.

[1] What is most important here is the order of quantifiers. The notion of PAC-learning requires a single learning algorithm to work regardless of the input distribution. Of course, the learning algorithm may be adaptive in the sense that depending on what examples it has received it can change its behaviour.

[2] If the output hypothesis is allowed to be randomised, that is it takes as input $x \in X$, and also has access to random coin tosses, when making a prediction, then it is immediately clear that the outlined approach works. Otherwise, this approach relies on being able to construct "pseudo-random" functions, and as such relies on an unproven, but widely believed conjecture in complexity theory. If $\epsilon > 1/2$, then we know that one of the two constant hypotheses, always predicting 1, or always predicting 0, gives error at most $1/2$, and we can with high confidence determine which one.

Boosting: A Short History

Boosting has had an interesting history and is a prominent example of how a suitable theoretical question has led to some very practical algorithms. The notion of weak learning first appeared in the work of Kearns and Valiant (1989), who showed that certain concept classes were hard to learn even when the requirement was only to output a hypothesis that was slightly better than random guessing. Shortly thereafter, Freund (1990) and Schapire (1990) showed that in the distribution-free setting weak and strong learning are in fact equivalent. The early boosting algorithms were not easy to implement in practice; Freund and Schapire (1995) designed an improved boosting algorithm, called Adaboost, which while retaining strong theoretical guarantees was very easy to implement in practice. Adaboost has enjoyed remarkable practical success and implementation of Adaboost and its variants appear in most machine learning libraries.

2 The AdaBoost Algorithm

The central idea of the boosting approach is the following. Initially, we can use a weak learning algorithm that gives us a hypothesis that performs slightly better than random guessing. We could repeatedly run this weak learning algorithm, though it may return the same hypothesis. However, if we modify the distribution so that the hypothesis already returned is no longer valid, *i.e.,* under the new distribution it has error exactly $1/2$, then the weak learning algorithm is required to provide us with a different hypothesis.[3] By doing this repeatedly, we can combine several hypotheses to produce one that has low error. All boosting algorithms make use of this high-level approach. The AdaBoost (for adaptive boosting) algorithm exploits the fact that some hypotheses may be much better than others and aggressively modifies the distribution to account for this. Initially, we will concentrate on proving that AdaBoost succeeds in finding a hypothesis that has training error 0 on a given sample.

The AdaBoost algorithm is described in Alg. 1. We assume that AdaBoost has access to the weak learning algorithm, WeakLearn. AdaBoost receives a training sample of m examples drawn from $\mathsf{EX}(c, D)$. It defines a distribution D_t over this sample at each iteration and hence can simulate the example oracle for the weak learning algorithm. To make the mathematical analysis simpler, we will assume that the labels y_i are in $\{-1, 1\}$ rather than $\{0, 1\}$. This is a transformation that is frequently used in machine learning and students should convince themselves that this does not make any difference.

Remark 2. *We will also make the simplifying assumption that the weak learning algorithm has failure probability* 0. *This is not entirely unrealistic as the weak learning algorithm is given a distribution with finite (and small) support. Thus, in principle the entire distribution could be provided to the weak learning algorithm and hence there is no reason for failure, unless the algorithm itself is randomised. In any case, a slightly more cumbersome analysis of the AdaBoost algorithm can easily be carried out, taking into account the possible failures at each stage and using the union bound to bound the combined probability of all the bad events.*

Theorem 3. *Assuming that* WeakLearn *is a γ-weak learner for the concept class C, after T iterations the training error of the hypothesis output by AdaBoost (Alg. 1) is* 0*, provided* $T \geq \frac{\log 2m}{2\gamma^2}$.

Proof. We will use the convention that $\text{sign}(z) = 1$ if $z \geq 0$ and $\text{sign}(z) = -1$ otherwise. Let $\mathbb{1}(\cdot)$ be the indicator of the predicate inside the parantheses, which takes the value 1 if the

[3]If the error of h is much larger than $1/2$ under the modified distribution, then the weak learning algorithm may simply return $1 - h$, which is not of much use, since we already have h.

Algorithm 1 AdaBoost

```
1: Input: Training data $(x_1, y_1), \ldots, (x_m, y_m)$ drawn from $\mathsf{EX}(c, D)$
2: $D_1(i) = 1/m$        ▷ Uniform initial distribution over the training data
3: for $t \leftarrow 1, \ldots, T$ do
4:     Obtain $h_t \leftarrow$ WeakLearn$(D_t)$ ▷ Examples drawn from $D_t$ are passed to WeakLearn
5:     Set $\epsilon_t = \mathbb{P}_{(x,y)\sim D_t}\left[h_t(x) \neq y\right]$        ▷ $\epsilon_t \leq 1/2 - \gamma$
6:     Set $\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
7:     Update $D_{t+1}(i) = D_t(i) \cdot \exp(-\alpha_t y_i h_t(x_i))/Z_{t+1}$        ▷ $Z_{t+1}$ is the normalizing constant
8: end for
9: Set $H = \sum_{i=t}^{T} \alpha_t h_t$
10: Output: Hypothesis $\mathrm{sign}(H(\cdot))$
```

predicate is true and 0 otherwise. Observe that $\mathbb{1}(\mathrm{sign}(H(x)) \neq y) \leq e^{-yH(x)}$ for $y \in \{-1, 1\}$.

$$\mathbb{P}_{(x,y)\sim D_1}\left[\mathrm{sign}(H(x)) \neq y\right] = \sum_{i=1}^{m} D(i) \cdot \mathbb{1}(\mathrm{sign}(H(x_i) \neq y_i) \leq \sum_{i=1}^{m} D_1(i) \cdot e^{-y_i H(x_i)} \tag{1}$$

We introduce some additional notation. Let $H_t = \sum_{s=t}^{T} \alpha_s h_s$ be the weighted sum of the hypotheses returned in iterations t through T; and thus, $H_t = \alpha_t h_t + H_{t+1}$. Then consider the following:

$$\begin{aligned}
\sum_{i=1}^{m} D_1(i) \cdot e^{-y_i H(x_i)} &= \sum_{i=1}^{m} D_1(i) \cdot e^{-y_i H_1(x_i)} && \text{As } H = H_1 \\
&= \sum_{i=1}^{m} D_1(i) \cdot e^{-\alpha_1 y_i h_1(x_i)} \cdot e^{-y_i H_2(x_i)} && \text{As } H_1 = \alpha_1 h_1 + H_2 \\
&= Z_2 \cdot \sum_{i=1}^{m} D_2(i) \cdot e^{-y_i H_2(x_i)} && \text{As } D_2(i) = D_1(i) \cdot e^{-\alpha_1 y_i h(x_i)}/Z_2 \\
&= Z_2 \cdot \sum_{i=1}^{m} D_2(i) \cdot e^{-\alpha_2 y_2 h_2(x_i)} \cdot e^{-y_i H_3(x_i)} && \text{As } H_2 = \alpha_2 h_2 + H_3 \\
&= Z_2 \cdot Z_3 \cdot \sum_{i=1}^{m} D_3(i) \cdot e^{-y_i H_3(x_i)} && \text{As } D_3(i) = D_2(i) \cdot e^{-\alpha_2 y_i h(x_i)}/Z_3
\end{aligned}$$

Continuing this way, we obtain,

$$\sum_{i=1}^{m} D_1(i) \cdot e^{-y_i H(x_i)} = Z_2 \cdot Z_3 \cdots Z_T \cdot \sum_{i=1}^{m} D_T(i) \cdot e^{-y_i H_T(x_i)}$$

And thus,

$$\sum_{i=1}^{m} D_1(i) \cdot e^{-y_i H(x_i)} = \prod_{t=2}^{T+1} Z_t \tag{2}$$

Let us now obtain a bound on Z_{t+1}, for $t = 1, \ldots, T$. We have,

$$\begin{aligned}
Z_{t+1} &= \sum_{i:h_t(x_i)=y_i} D_t(i) \cdot e^{-\alpha_t} + \sum_{i:h_t(x_i)\neq y_i} D_t(i) \cdot e^{\alpha_t} \\
&= (1-\epsilon_t)e^{-\alpha_t} + \epsilon_t e^{\alpha_t} = 2\sqrt{\epsilon_t(1-\epsilon_t)} && \text{Substituting } \alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}
\end{aligned}$$

Letting $\gamma_t = \frac{1}{2} - \epsilon_t$ and using the fact that $\sqrt{1-x} \le e^{-x/2}$, we get,

$$Z_{t+1} = \sqrt{1 - 4\gamma_t^2} \le e^{-2\gamma_t^2} \tag{3}$$

Now, by the guarantee on the weak learning algorithm, $\gamma_t \ge \gamma$ for $t = 1, \ldots, T$. Thus, $\prod_{t=2}^{T+1} Z_t \le e^{-2T\gamma^2}$. Provided $T \ge \log(2m)/(2\gamma^2)$, the training error is at most $1/(2m)$ and hence must in fact be 0 (as error on any point causes the error to be at least $1/m$). □

2.1 Bounding the Generalization Error

In order to bound the generalization error, we need to ensure that the VC-dimension of the hypothesis class used by the weak learning algorithm is small. Suppose the weak learning algorithm, WeakLearn, outputs hypotheses from H and $\mathsf{VCD}(H) = d$. Denote by $\mathsf{TH}_k(H)$ the class of functions given by:

$$\mathsf{TH}_k(H) = \{x \mapsto \text{sign}\left(\sum_{i=1}^{k} \alpha_k h_k(x)\right) \mid h_i \in H, \alpha_i \in \mathbb{R}\}.$$

Lemma 1. *If* $\mathsf{VCD}(H) = d$, *then* $\mathsf{VCD}(\mathsf{TH}_k(H)) = O(kd)$.

The proof of the lemma is left as an exercise. Let us see how to use this lemma to give a bound on the generalisation error of the hypothesis output by AdaBoost. To do this, we'll modify the result of Theorem 3 slightly. We consider T to be the number of iterations required to ensure that the training error is at most $\epsilon/2$, that is at most $\epsilon m/2$ examples in the training data are classified incorrectly. It is an easy exercise to show that this can be achieved provided $T \ge \frac{1}{2\gamma^2} \log \frac{2}{\epsilon}$. We have already see that if a hypothesis from a class that has bounded VC dimension is consistent with a sample drawn from a distribution, it has low error with respect to the underlying distribution. We will use a slightly stronger result that holds even when a hypothesis makes mistakes on a sample. The following theorem asserts that provided the sample is large enough, the *empirical error*, *i.e.,* the error observed on a sample is not that different from the error under the actual distribution. Note that this theorem does not even require the data to be labelled correctly according to any target concept from a concept class.

Theorem 4. *Let H be a class of boolean functions over some domain X with* $\mathsf{VCD}(H) = d$. *Let D be any distribution over the pairs $X \times \{0,1\}$. For any $h \in H$, define $\text{err}(h; D) = \mathbb{P}_{(x,y)\sim D}\left[h(x) \neq y\right]$. Let $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ be a sample of size m drawn independently from D. Define, $\widehat{\text{err}}(h; S) = \frac{1}{m}|\{i \mid (x_i, y_i) \in S, y_i \neq h(x_i)\}|$ to be the empirical error of h on the sample S. Then for any $\delta > 0$,*

$$\mathbb{P}\left[\exists h \in H : |\widehat{\text{err}}(h; S) - \text{err}(h; D)| > \sqrt{\frac{8d \ln \frac{2em}{d} + 8 \ln \frac{4}{\delta}}{m}}\right] \le \delta$$

where the probability is computed with respect to the random choice of the sample S.

The above theorem together with Lemma 1 implies that provided we start with a sample of size m that is large enough, yet polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$ and $\frac{1}{\gamma}$, the hypothesis output by AdaBoost has error at most ϵ with probability at least $1 - \delta$.

References

Yoav Freund. Boosting a weak learning algorithm by majority. In *COLT*, volume 90, pages 202–216, 1990.

Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, 1989.

Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.