6.858/18.428 Machine Learning	**Lecture 12: October 24, 1994**
Lecturer: Ron Rivest	*Scribe: Lillian Lee*

12.1 Outline

- Definition of weak learnability
- Weak learnability implies PAC-learnability

  –Boosting confidence

  –Boosting accuracy

12.2 Weak Learning

Notice that the definition of PAC-learning imposes some rather stringent conditions, among them the requirement that a PAC-learning algorithm be able to output hypotheses that have arbitrarily low error rates, and be able to do so with arbitrarily high confidence. In his Ph.D. thesis, Schapire considered (among other topics) the *weak learning* model, which differs from the PAC-learning model in that the learning algorithm need only achieve a fixed (but small) error rate and confidence. We go through his proof of the rather surprising fact that if a concept class is weakly learnable, then it is PAC-learnable.

12.2.1 Definition of Weak Learnability

Let $EX(c, \mathcal{D})$ be as before.

Definition 1 *A concept class $\mathcal{C}$ is* **weakly learnable** *if there exists a learning algorithm L and constants $\epsilon_0 < \frac{1}{2}$ and $\delta_0 < 1$ such that for every concept $c_* \in \mathcal{C}$ and every distribution $\mathcal{D}$ on X, algorithm L, with access to $EX(c, \mathcal{D})$, returns a hypothesis h such that with probability at least $1 - \delta_0$, $err(h)$ is at most ϵ_0.*

Notice that we can define "weakly learnable using $\mathcal{H}$" and "efficiently weakly learnable" in the same fashion as we did for PAC-learnability. Also note that this definition differs slightly from that given in the book (and in the next lecture).

12.2.2 Weak learnability implies PAC-learnability

Our main theorem is the following:

Theorem 1 *If a concept class $\mathcal{C}$ is weakly learnable by $\mathcal{H}$, then it is PAC-learnable by $\mathcal{H}'$, where $\mathcal{H}'$ is the closure of $\mathcal{H}$ under majority operations.*

We prove this theorem by first showing that we can boost the confidence of certain learning algorithms, and then showing that we can boost the accuracy of certain learning algorithms.

12.2.3 Boosting confidence

Suppose we have a weak learning algorithm L that achieves arbitrarily low error rates, but only does so with confidence at most δ_0. Then, we can raise the confidence to any δ we want (thereby creating a PAC-learning algorithm) by running the algorithm many times and picking the output hypothesis with the lowest estimated error rate.

Suppose we run L k times with parameters $\epsilon/2$ and δ_0. Let $h_1, h_2, \ldots, h_k$ be the independent hypotheses that L outputs. Then we have

$$\begin{aligned} (\forall i)\Pr\{err(h_i) \le \frac{\epsilon}{2}\} &\ge 1-\delta_0 \\ \Rightarrow \Pr\{(\forall i) err(h_i) > \frac{\epsilon}{2}\} &\le \delta_0^k \\ \Rightarrow \Pr\{(\exists i) err(h_i) \le \frac{\epsilon}{2}\} &\ge 1-\delta_0^k. \end{aligned}$$

Choose k so that $\delta_0^k \le \delta/2$, that is, $k \ln \delta_0 \le \ln(\delta/2) \Rightarrow k \ge \log_{\delta_0}(\delta/2)$, since $\delta_0 < 1$. Then, with probability at least $1-\delta/2$, we have some hypothesis h_i such that $err(h_i) \le \epsilon/2$.

Now it remains to figure out *which* of our h_i's has the lowest error rate; we do this by drawing a sample of size m for each h_i to estimate each h_i's error rate. It's sufficient that our estimate be accurate within $\epsilon/4$ because our aim is to output a hypothesis with error rate at most ϵ: if $err(h_i) \leq \epsilon/2$ then our error estimate will be at most $3\epsilon/4$, whereas if $err(h_i) > \epsilon$, our error estimate will be above $3\epsilon/4$. The additive form of the Chernoff bound gives us that $GE(p, m, (p+\epsilon/4)) \leq \exp(-2m(\epsilon/4)^2)$; setting the latter quantity less than or equal to $\delta/(2k)$ implies that $m \geq \frac{8}{\epsilon^2}\ln(2k/\delta)$.

What is the chance that we output a hypothesis with error greater than ϵ with the above procedure? Well, either among our k tries we never get a hypothesis with error rate $\leq \epsilon/2$ or we make a bad mistake when estimating error rates. But the chance of either of these event occurring is $\leq \delta/2 + \delta/2 = \delta$, as desired.

We remark that there is also a method for taking an algorithm with fixed ϵ_0 and δ_0 and constructing an algorithm which, on inputs ϵ and δ, outputs with probability at least $1-\delta$ a hypothesis with error rate at worst only slightly larger than ϵ.

12.2.4 Boosting accuracy

Now suppose we have a weak learning algorithm L that outputs hypotheses with error rate at most $\beta < 1/2$ but does so with probability at least $1-\delta$ for any input δ. We then take advantage of the fact that weak learning algorithms can learn from *any* distribution to find hypotheses h_1, h_2, and h_3, formed on three different distributions. Our output is $h = \text{majority}(h_1, h_2, h_3)$ (this is why we learn using $\mathcal{H}'$); h will have a lower error rate with high probability.

Let c_* be the concept we are trying to learn, and let $\mathcal{D}_1 = \mathcal{D}$. The first step is to run L with $EX(C, \mathcal{D})$ to get a hypothesis h_1 such that $err(h_1) = \beta_1 \leq \beta < 1/2$. Then, we want to focus L's attention where it does poorly by running L on a new distribution that gives greater weight to examples on which h_1 errs.

A first attempt to do this would be to only give L examples where the target concept c_* differs from h_1. However, the result of this might be that L outputs exactly the complement of h_1, which is not necessarily any better than h_1. So instead, let us consider a new oracle which flips a coin: on heads, it returns an example x such that $h_1(x) \neq c_*(x)$, and on tails, it returns an example x such that $h_1(x) = c_*(x)$ (in the next lecture, we will discuss what to do if it is hard to get such a distribution). Let $\mathcal{D}_2$ be this new distribution. Then, run L using the oracle $EX(c, \mathcal{D}_2)$; let h_2 be the output. Let $\beta_2 \leq \beta$ be the error rate of h_2 on $\mathcal{D}_2$.

The next step is to find a hypothesis h_3 to act as a tiebreaker. We will learn this hypothesis by using yet another distribution $\mathcal{D}_3$ that consists entirely of examples where h_1 and h_2 disagree (if drawing examples from $\mathcal{D}_3$ is too hard (which means that h_1 and h_2 are "too good"), we output h_1 or h_2). Let $\beta_3 \leq \beta$ be h_3's error rate on $\mathcal{D}_3$.

Finally, we output $h = \text{majority}(h_1, h_2, h_3)$.

Claim 1 *$err_{\mathcal{D}}(h) \leq g(\beta)$, where $g(\epsilon) = 3\epsilon^2 - 2\epsilon^3$.*

Proof:

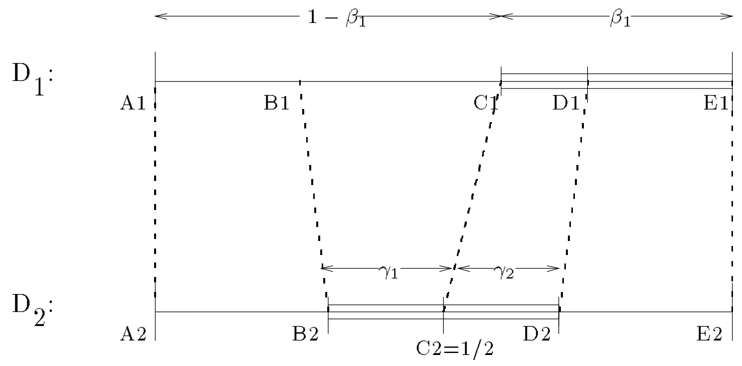

Figure 12.1: Mapping from $\mathcal{D}_1$ to $\mathcal{D}_2$

Consider the figure below. C1-E1 is the region where h_1 is wrong, B2-D2 is the region where h_2 is wrong, and C1-D1 is the region where both are wrong. B2-C2 has weight γ_1 under $\mathcal{D}_2$; C2-D2 has weight γ_2 under $\mathcal{D}_2$. Notice that regions to the right of C1 are mapped to regions with probability $1/(2\beta_1)$ times more under $\mathcal{D}_2$ than they had under $\mathcal{D}_1$, whereas regions to the left of C1 are mapped to regions with probability $1/(2(1-\beta_1))$ times less under $\mathcal{D}_2$ than they had under $\mathcal{D}_1$. Applying this mapping backwards gives us the following information:

Region	Weight under $\mathcal{D}_1$
A1-B1	$2(1-\beta_1)(1/2-\gamma_1)$
B1-C1	$2(1-\beta_1)\gamma_1$
C1-D1	$2\beta_1\gamma_2$
D1-E1	$2\beta_1(1/2-\gamma_2)$

From now on, assume all weights are with respect to $\mathcal{D}$. We have:

$$\begin{aligned} err(h) &\leq \Pr\{h_1 \neq c_* \text{ and } h_2 \neq c_*\} + \Pr\{h_1 \neq h_2 \text{ and } h_3 \text{ wrong}\} \\ &\leq \text{weight(C1-D1)} + (\text{weight(B1-C1)} + \text{weight(D1-E1)})(\beta_3) \\ &\leq 2\beta_1\gamma_2 + (2(1-\beta_1)\gamma_1 + 2\beta_1(1/2-\gamma_2))\beta, \text{ since } \beta_3 \leq \beta \quad (*) \end{aligned}$$

$$
\begin{aligned}
&= \beta_1(2\gamma_2 - 2\beta\gamma_1 + 2\beta(1/2 - \gamma_2)) + 2\beta\gamma_1 \\
&= \beta_1(2\gamma_2 - \beta(2(\gamma_1 + \gamma_2) - 1)) + 2\beta\gamma_1 \\
&= \beta_1(2\gamma_2 + \beta(1 - 2(\gamma_1 + \gamma_2)) + 2\beta\gamma_1.
\end{aligned}
$$

But $\gamma_1 + \gamma_2 = \beta_2 \leq \beta < 1/2$ so $1 - 2(\gamma_1 + \gamma_2)$ is positive $\Rightarrow (2\gamma_2 + \beta(1 - 2(\gamma_1 + \gamma_2))$ is positive. So $err(h)$ is an increasing function of β_1, which means we can replace β_1 by β in (*). Thus,

$$
\begin{aligned}
err(h) &\leq 2\beta\gamma_2 + \beta(2(1-\beta)\gamma_1 + 2\beta(1/2 - \gamma_2)) \\
&= 2\beta\gamma_2 + 2\beta\gamma_1 - 2\beta^2\gamma_1 + \beta^2 - 2\beta^2\gamma_2 \\
&= 2\beta\gamma_2(1-\beta) + 2\beta\gamma_1(1-\beta) + \beta^2 \\
&= \beta^2 + 2\beta(1-\beta)(\gamma_1 + \gamma_2) \\
&\leq \beta^2 + 2\beta(1-\beta)\beta \\
&= 3\beta^2 - 2\beta^3 \\
&= g(\beta)
\end{aligned}
$$

■

12.2.5 Next lecture

To finish the proof, we still need to show that we can simulate $EX(c, \mathcal{D}_2)$ and $EX(c, \mathcal{D}_3)$; after all, we would run into trouble if it were never the case that $h_1(x) \neq h_2(x)$. Also, it remains to be shown that we can iterate the above procedure to get the error rate down to arbitrary levels in a reasonable amount of time.

References

[1] R. E. Schapire. *The Design and Analysis of Efficient Learning Algorithms.* MIT Press, Cambridge, MA, 1992.