6.842 Randomness and Computation April 25, 2022

Lecture 22

Lecturer: Ronitt Rubinfeld *Scribe: Kai Zheng*

Definition 1 *An algorithm $\mathcal{A}$* weakly PAC learns *concept class $\mathcal{C}$ if there is a $\gamma > 0$ such that, for any $c \in \mathcal{C}$, distribution $\mathcal{D}$, and $\delta > 0$, with probability at least $1-\delta$, given examples of c from the distribution $\mathcal{D}$, $\mathcal{A}$ outputs a function h such that* $\mathrm{err}_{\mathcal{D}}(h) \equiv \Pr_{\mathcal{D}}[h(x) \neq c(x)] \leq \frac{1}{2} - \frac{\gamma}{2}$.

Notice that in the above definition, the learned function is only required to be correct a $\gamma/2$ fraction of time more than simply guessing. Our objective for these notes is to show that distribution-free weak learning implies strong learning, which is defined as follows.

Definition 2 *An algorithm $\mathcal{A}$* strongly PAC learns *concept class $\mathcal{C}$ if for any $c \in \mathcal{C}$, distribution $\mathcal{D}$, and $\epsilon, \delta > 0$, with probability at least $1-\delta$, given examples of c from the distribution $\mathcal{D}$, $\mathcal{A}$ outputs a function h such that* $\mathrm{err}_{\mathcal{D}}(h) \equiv \Pr_{\mathcal{D}}[h(x) \neq c(x)] \leq \epsilon$.

Theorem 3 *$\mathcal{C}$ is weak learnable $\implies$ $\mathcal{C}$ is strong learnable.*

1 Part 1: Modest Boosting

We will show that through a modest accuracy boosting algorithm, we may use an algorithm $\mathcal{A}$ that weakly PAC learns $\mathcal{C}$, to strongly learn the concept class. The algorithm works as follows.

Suppose that we are given labelled samples of a function $f \in \mathcal{C}$, $(x_1, f(x_1)), (x_2, f(x_2)), \ldots$, where the x_i are drawn from $\mathcal{D}$. Our goal is to strongly learn f using $\mathcal{A}$.

1. Run $\mathcal{A}$ on $\mathcal{D}$ for f, output a function h_1.
2. Create an example oracle $\mathcal{D}_2$ as follows, so that $\mathcal{D}_2$ outputs an x such that $h(x) = f(x)$ with probability 1/2. Run $\mathcal{A}$ on $\mathcal{D}_2$, for f and output a function h_2.
3. Create an example oracle $\mathcal{D}_3$ that only outputs x such that $h_1(x) \neq h_2(x)$. Run $\mathcal{A}$ on $\mathcal{D}_3$ for f, output a function h_3.
4. Output $h \equiv \mathrm{maj}(h_1, h_2, h_3)$.

Note that we may generate $\mathcal{D}_2$ by flipping a coin, and if heads, drawing samples from $\mathcal{D}$ until we obtain an x such that $h(x) = f(x)$, and if tails drawing samples from $\mathcal{D}$ until we obtain an x such that $h(x) \neq f(x)$. We will show later that if h_1 is not already close to f, then this will not take too many samples. This allows us to efficiently sample from $\mathcal{D}_2$.

We first show the following lemma, which quantifies the modest boost. It will help to define the following quantities.

- $\beta_1 = \Pr_{\mathcal{D}}(h_1(x) \neq f(x))$
- $\beta_2 = \Pr_{\mathcal{D}_2}(h_2(x) \neq f(x))$
- $\beta_3 = \Pr_{\mathcal{D}_3}(h_3(x) \neq f(x))$

By construction, for x such that $h(x) = f(x)$, $\mathcal{D}_2(x) = \frac{1}{2} \Pr_{\mathcal{D}}[x \mid h(x) = f(x)] = \frac{\mathcal{D}(x)}{2(1-\beta_1)}$, or equivalently $\mathcal{D}(x) = 2(1-\beta_1)\mathcal{D}_2(x)$. A similar conditional expectation yields that for x such that $h(x) = f(x)$, $\mathcal{D}(x) = 2\beta_1 \mathcal{D}_2(x)$.

Lemma 4 *Let $\beta = \max(\beta_1, \beta_2, \beta_3)$. Then,* $\mathrm{err}(h) \leq g(\beta) = 3\beta^2 - 2\beta^3$.

Proof The function h can err if $h_1(x) \neq f(x)$ and $h_2(x) \neq f(x)$, or $h_1(x) \neq h_2(x)$ and $h_3(x) \neq f(x)$. Formally,

$$\begin{aligned}\mathrm{err}_{\mathcal{D}}(H) &= \Pr_{\mathcal{D}}[h_1(x) \neq f(x), h_2(x) \neq f(x)] + \Pr_{\mathcal{D}}[h_3(x) \neq f(x) \mid h_1(x) \neq h_2(x)] \\ &\leq \Pr_{\mathcal{D}}[h_1(x) \neq f(x), h_2(x) \neq f(x)] + \beta_3 \Pr_{\mathcal{D}}[h_1(x) \neq h_2(x)].\end{aligned}$$

To manipulate this expression, we split the error of h_2 into two values,

$$\begin{aligned}\alpha_1 &= \Pr_{\mathcal{D}_2}[h_2(x) \neq f(x), h_1(x) = f(x)] \\ \alpha_2 &= \Pr_{\mathcal{D}_2}[h_2(x) \neq f(x), h_1(x) \neq f(x)].\end{aligned}$$

Note that $\beta_2 = \alpha_1 + \alpha_2$, and $\Pr_{\mathcal{D}}[h_1(x) \neq f(x), h_2(x) \neq f(x)] = \beta_1\alpha_2$. Furthermore, by the observation that $\mathcal{D}(x) = 2(1-\beta_1)\mathcal{D}_2(x)$ for all x such that $h(x) = f(x)$,

$$\Pr_{\mathcal{D}}[h_2(x) \neq f(x), h_1(x) = f(x)] = 2(1-\beta_1)\Pr_{\mathcal{D}_2}[h_2(x) \neq f(x), h_1(x) = f(x)] = 2(1-\beta_1)\alpha_1.$$

By construction, $\Pr_{\mathcal{D}_2}[h_1(x) \neq f(x)] = \frac{1}{2}$, so $\Pr_{\mathcal{D}_2}[h_1(x) \neq f(x), h_2(x) = f(x)] = \frac{1}{2} - \alpha_2$, and by the observation that $\mathcal{D}(x) = 2\beta_1\mathcal{D}_2(x)$ for all x such that $h(x) \neq f(x)$,

$$\Pr_{\mathcal{D}}[h_2(x) = f(x), h_1(x) \neq f(x)] = 2\beta_1 \Pr_{\mathcal{D}_2}[h_2(x) = f(x), h_1(x) \neq f(x)] = 2\beta_1(\frac{1}{2} - \alpha_2).$$

Putting it all together, we get that, $\Pr_{\mathcal{D}}[h_1(x) \neq h_2(x)] \leq 2(1-\beta_1)\alpha_1 + 2\beta_1(\frac{1}{2} - \alpha_2) = 2\alpha_1 + \beta_1 - 2\beta_1\beta_2$. Recall $\beta = \max(\beta_1, \beta_2, \beta_3)$. It will be necessary for the next part to note that by the distribution free learning guarantee of $\mathcal{A}$, we get the same bound of $\frac{1}{2} - \gamma$ for each of β_1, β_2, and β_3. For now, we simply conclude:

$$\mathrm{err}_{\mathcal{D}}(H) \leq 2\beta_1\alpha_2 + \beta_3(2\alpha_1 + \beta_1 - 2\beta_1\beta_2) \leq 3\beta^2 - 2\beta^3$$

■

2 Part 2: Recursive Accuracy Boosting

The boosting algorithm above can take an error of $\beta < 1/2$, guaranteed by $\mathcal{A}$, and reduce this error to $3\beta^2 - 2\beta^3$. We now describe how to achieve strong learning through recursion.

Algorithm stronglearn$(\rho, \mathcal{D}')$:

- If $\rho < \frac{1}{2} - \frac{\gamma}{2}$ return $\mathcal{A}$ on $\mathcal{D}'$.
- Else, set $\beta = g^{-1}(\rho)$:
- Define, $\mathcal{D}_2', \mathcal{D}_3'$ as in modest boost and let $\mathcal{D}_1' = \mathcal{D}'$.
- Set $h_i \leftarrow$ stronglearn$(\beta, \mathcal{D}_i')$ for $i = 1, 2, 3$.
- return $h \equiv \mathrm{maj}(h_1, h_2, h_3)$.

We analyze the sample complexity of this algorithm. For simplicity assume that the advantage of $\mathcal{A}$ is at least 1/2, so $\gamma \geq 1/2$. Then $\beta < 1/4$ always, and $g(\beta) \leq 3\beta^2 = \frac{1}{3}(3\beta)^2$. Thus, after k recursive calls, the error is at most $\frac{1}{3}(3\beta)^{2^k}$ and $k = \Theta(\log\log(\frac{1}{\epsilon}))$ suffices to get error ϵ. In other words, for $k = \Theta(\log\log(\frac{1}{\epsilon}))$, $g^{-k}(\epsilon) \geq \frac{1}{2} - \frac{\gamma}{2}$, for $\gamma > 1/2$. Moreover, this results in an output hypothesis of size $O(s\log(1/\epsilon)$, describable, for example, by a circuit.

Moreover, it does not take too many samples from $\mathcal{D}'$ to sample from the distributions $\mathcal{D}'_2$ and $\mathcal{D}'_3$. We will not show this explicitly, but the intuition is that in order to find samples such that $h_1(x) = f(x)$, more than half of the samples should satisfy this requirement. For samples such that $h_1(x) \neq f(x)$, if we cannot find such samples efficiently, then h_1 is already a good approximation of f. Likewise, if samples such that $h_1(x) \neq h_2(x)$ are hard to find, then we do not need h_3 to define $h \equiv \text{maj}(h_1, h_2, h_3)$ anymore. Altogether, this shows the following theorem.

Theorem 5 *If $\mathcal{C}$ is weakly learnable and size at most s, then there exists an efficient algorithm using $\frac{poly(n,s,\log(1/\epsilon),\log(1/\delta))}{\epsilon}$ samples that outputs hypotheses of size $poly(n,s,\log(1/\epsilon))$ that has error at most ϵ with probability at least $1-\delta$.*