CS787: Advanced Algorithms

Scribe: Evan Driscoll and Dalibor Zelený
Lecturer: Shuchi Chawla
Topic: Boosting in the PAC Model
Date: November 29, 2007

26.1 Review of Last Class

26.1.1 PAC-learnability and consistency

We defined PAC-learnability as follows:

Definition 26.1.1 *A concept class C is PAC-learnable if there is an algorithm A such that for all $\epsilon > 0$ and $\delta > 0$, for all distributions D on the domain, and for all target functions $f \in C$, given a sample S drawn from D, A produces a hypothesis h in some concept class C' such that with probability at least $1 - \delta$, $\mathbf{Pr}_{x \sim D}[h(x) \neq f(x)] < \epsilon$.*

Both the running time of algorithm A and the size of S should be $\text{poly}(1/\varepsilon, 1/\delta, n, \log|C|)$. *Finally, $h(x)$ should be easy to compute.*

Recall that we do not require h to be drawn from the same concept class as C. The motivating example was that we were trying to find functions in the concept class of 3-term DNF formulas, but doing so is NP-hard. However, by instead finding a consistent 3-DNF formula, we could still get accurate results, and the problem becomes easy.

Finally, the requirement that the running time of A and the size of S should be polynomial in $\log|C|$ makes sense if one considers the fact that, since there are $|C|$ possible functions, simply recording which we chose requires at least $\log|C|$ bits. Thus outputting the function we chose gives a lower bound on the running time of $\log|C|$.

There is an easy goal that, if it is possible to reach, implies that a class is PAC-learnable:

Theorem 26.1.2 *If there is an algorithm A that, given a sample S, solves consistency in time* $\text{poly}(|S|, n)$ *(in other words, produces an $h \in C'$ consistent with S), then we can PAC-learn C if we choose a sample size of at least $|S| \geq \frac{1}{\epsilon}(\log\frac{1}{\delta} + \log|C'|)$.*

As long as C' isn't too much bigger than C, this satisfies the requirements given above for PAC-learnability. A proof of this was given last class.

26.1.2 Noisy Data

We also discussed how to compensate for noisy data. Specifically, there may not be a function f that is completely consistent with any sample. In this case, our goal is to try to learn a function that makes the fewest mistakes.

One way of looking at what we did above is as follows. There is a function $f \in C$ that classifies all data with an error rate of zero: $\text{error}_D(f) = 0$. We draw $S \sim D$ and use it to produce a hypothesis h that classifies all data in S with error zero: $\text{error}_S(h) = 0$. When we try generalize this h to the whole domain, h does well: with probability at least $1 - \delta$, we have $\text{error}_D(h) \leq \epsilon$.

With noisy data, our model changes to the following. There is a function $f \in C$ that classifies all data with an non-zero error rate of p: $\text{error}_D(f) = p$. We draw $S \sim D$ and use it to produce a hypothesis h that classifies all data in S with the lowest possible error rate of all possible h: $\text{error}_S(h) = \min_{h' \in C'} \text{error}_S(h') = p'$. When we generalize this h to the whole domain, h does well: with probability at least $1 - \delta$, we have $\text{error}_D(h) \leq p' + \sqrt{(\log \frac{1}{\delta} + \log |C|)/(2|S|)}$.

26.2 Weak PAC-learnability

We now turn our attention to another question. The conditions required for PAC-learnability seem a bit strong: algorithm A needs to work for any δ and any ϵ. However, suppose that we only had an algorithm that worked for a specific value of δ and ϵ? Is it still possible to do PAC learning? It turns out the answer is "yes," even if $1 - \delta$ is close to zero (i.e. we have very low confidence) and ϵ is just under $1/2$ (i.e. we are only doing slightly better than raw guessing).

We define a new notion, called "weak PAC-learnability," and refer to the previous definition as "strong PAC-learnability." Weak PAC-learnability is defined as follows:

Definition 26.2.1 *A concept class C is weakly PAC-learnable if there exists a $\gamma > 0$ and a $\delta' > 0$ such that there exists an algorithm A such that for all distributions D and all target functions $f \in C$, given a sample $S \sim D$ then A will produce a hypothesis h such that with probability δ', $\mathbf{Pr}_{x \sim D}[h(x) \neq f(x)] \leq \frac{1}{2} - \gamma$.*

Our question can now be rephrased to ask "does weak PAC-learnability imply strong PAC-learnability?" We will that it can by two steps. The first shows that it is possible to boost the confidence from δ' to any arbitrary δ, while not increasing the error rate "too much." The second step will boost the accuracy from $\frac{1}{2} - \gamma$ to an arbitrary ϵ while leaving the confidence unchanged.

26.2.1 Boosting the confidence

We first show how to boost the confidence. Suppose that we are given an algorithm A that gives an h with $\mathbf{Pr}_{x \sim D}[h(x) \neq f(x)] \leq \varepsilon$ with probability δ'. We will show that we can generate a h' such that $\mathbf{Pr}_{x \sim D}[h(x) \neq f(x)] \leq 2\varepsilon$ with probability at least $1 - \delta$, for any $\delta > 0$.

We will first look at an idea that doesn't work. Imagine drawing k samples $S_1, \ldots, S_k$, using A to generate hypotheses $h_1, \ldots, h_k$, then combining the h_is in some way. Taking a majority function doesn't work, because most of the h_i's can be wrong. (Recall that δ' can be close to 0.) Taking the opposite doesn't work either, because they *could* mostly be correct. Because of the very weak guarantees in our problem statement, there is essentially no way to combine the h_i's to get what we want.

However, we can use a similar process to get something that *will* work. We draw k samples and produce hypotheses as before. Note that if we draw enough, chances are that at least *one* of the hypotheses will be good; we just have to figure out which one it is. (We use the term "good" to refer to a hypothesis h for which the bounds on the error apply, and "bad" for the others.) To figure out which h_i is the best hypothesis, we test the accuracy of each of them on another sample S_{k+1}.

For this process to work with high probability, we must ensure two things:

1. We must choose k large enough that we are sufficiently assured that there will be a good hypothesis in the set of h_i's
2. We must choose the size of S_{k+1} so that it is large enough that we are sufficiently assured that the good hypothesis will do a good enough job on S_{k+1} that we will choose it over a bad hypothesis

26.2.1.1 Choosing k

This is the easy part. For each $i \in [k]$, the chance that h_i is good is at least δ'. Thus the probability that *no* h_i is good is $(1-\delta')^k$. We will set k so that this is at most $\delta/2$ for our target δ. (The division by 2 leaves us some leeway for error in the next step.) This ensures that with probability at least $1-\delta/2$, there is a good hypothesis in the set.

Setting $(1-\delta')^k = \frac{\delta}{2}$, we can solve for k, to be $\log\frac{\delta}{2}/\log(1-\delta')$. By using the approximation that $\log\frac{1}{1-x} \approx x$, we can simplify this to $\frac{1}{\delta'}\log\frac{2}{\delta}$.

26.2.1.2 Choosing $|S_{k+1}|$

For any good hypothesis h_i, we want the error rate over the sample S_{k+1} to be, with high probability, about the same as the error rate over the entire distribution. Specifically, we want it to be within $\varepsilon/2$. More formally, we want $\mathbf{Pr}\left[|\operatorname{error}_D(h_i) - \operatorname{error}_{S_{k+1}}(h_i)| > \frac{\varepsilon}{2}\right]$ to be "small enough."

We can bound the above probability using Chernoff's bound to $2\cdot\exp\left(-2\frac{\varepsilon^2}{4}|S_{k+1}|\right)$. To see this, we an alternate form of Chernoff's bound presented in Kearns and Vazirani ([1], section 9.3). Suppose that $X = \Sigma_{j=1}^{n} X_j$ with every X_j a Boolean random variable as in the original formulation, and for all j, $\mathbf{Pr}[X_j = 1] = p$. Then if we take a sample and observe that a proportion $\hat{p}$ of the drawn X_j's are equal to 1, then we can bound the difference between p and $\hat{p}$ as $\mathbf{Pr}[|\hat{p} - p| \geq \gamma] \leq 2e^{-2n\gamma^2}$.

In our example, the samples we are drawing for the X_j's are each member of S_{k+1}, and $X_j = 0$ if the hypothesis h_i is correct and $X_j = 1$ if it is incorrect. Thus n is $|S_{k+1}|$. Because the probability p of $X_j = 1$ is equal to the overall error rate of h_i, $p = \operatorname{error}_D(h_i)$; our estimator is then $\hat{p} = \operatorname{error}_{S_{k+1}}(h_i)$. Finally, $\gamma = \frac{\varepsilon}{2}$. Plugging these into the alternate form of Chernoff's inequality gives us our goal.

We now want to choose $|S_{k+1}|$ so that is is bounded by $\delta/2k$; we will see how this works out in a moment. We want to ensure that *no* good hypothesis is too far off across all the h_i's. Taking the union bound, we get that $\mathbf{Pr}\left[\exists i.\ |\operatorname{error}_D(h_i) - \operatorname{error}_{S_{k+1}}(h_i)| > \frac{\varepsilon}{2}\right] < k\cdot\exp\left(-2\frac{\varepsilon^2}{4}|S_{k+1}|\right) < \frac{\delta}{2}$. Thus with probability of at most $1-\frac{\delta}{2}$, no h_i has a difference in error of more than $\frac{\varepsilon}{2}$ between the global true error ε and the emperical error on S_{k+1}, and thus no h_i has an emperical error greater than $\frac{3}{2}\varepsilon$.

Thus when we choose the h_k with the least error, with probability $1-\frac{\delta}{2}$ it will have an emperical error of at most $\frac{3}{2}\varepsilon$. Going again to the probabilistic bound of $\frac{1}{2}\varepsilon$ between the emperical and false error, this means that the true error is at most $\frac{3}{2}\varepsilon + \frac{1}{2}\varepsilon = 2\varepsilon$.

26.2.1.3 Final confidence boosting algorithm

For us to not choose an h with low enough error, either there must have not been one present (a failure of the first part) or we chose the wrong hypothesis (a failure of the second part). Both parts have a probability of $\delta/2$ of failure, so by union bound the probability of either failing is at most δ. This means that with probability at least $1-\delta$, we will get a good enough hypothesis.

With the preceding parts in place, we can give a formal presentation of the algorithm to boost confidence:

for i from 1 to k, where $k = \frac{1}{\delta'}\log\frac{2}{\delta}$:
    draw $S_i \sim D$, run A to find h_i
Draw $S_{k+1} \sim D$ with size $O(1/\epsilon^2 \log k/\delta)$ (ensuring that $\exp\left(-2\frac{\varepsilon^2}{4}|S_{k+1}|\right) \leq \frac{\delta}{2k}$)
Find errors $\text{error}_{S_{k+1}}(h_i)$ for each i
Output $\text{argmin}_{h_i} \text{error}_{S_{k+1}}(h_i)$

26.3 Error Reduction

In the previous section, we were interested in raising the confidence in our hypothesis. In this section, we study a boosting algorithm that decreases the error probability, given that the error probability of the hypothesis produced by our algorithm is bounded away from 1/2.

Suppose that with probability $1-\delta$, some algorithm A produces a hypothesis that has an error probability of no more than $1/2-\gamma$. For $\delta = 0$, we first show how to design an algorithm that produces (with probability $1-\delta$) a hypothesis that has an error probability of $1/2-\gamma'$ with $\gamma' > \gamma$. We then apply this procedure recursively to get an error probability of ϵ for any $\epsilon > 0$.

First consider the following approach that doesn't work. Following the ideas of probability amplification, we would be tempted to get k samples $S_1, \ldots, S_k$ from a distribution D, and use A to create k different hypotheses $h_1, \ldots, h_k$. Given some x taken from our distribution D, we will then compute $h_i(x)$ for all i and side with the majority. By Chernoff's bound, the majority vote is correct with high probability.

This approach doesn't work. The sample sets $S_1, \ldots, S_k$ are independent of each other; however, the resulting hypotheses $h_1, \ldots, h_k$ are correlated. Therefore, we cannot use Chernoff's bound to say that the majority vote is correct with high probability. Other concentration bounds discussed in class don't give us a good probability of the correctness of the majority vote either, so we will have to modify our approach.

Suppose that with probability 1, A gives a hypothesis h that has error probability $p = 1/2-\gamma$. We would like to decrease this error probability to $1/2-\gamma'$ with $\gamma' > \gamma$. In order to do that, we invoke A three times, each time with a slightly different distribution. Let D be our original distribution. We let D_1 be D, and use A on a sample S_1 from D_1 to get a hypothesis h_1. In the next round, we would like to focus more on the examples x for which h_1 errs. Unfortunately, we have no quick way of sampling from the subset of the examples that are marked incorrectly by h_1. We can skew the distribution so that the total weight of the correctly marked examples is 1/2, which makes the

total weight of the incorrectly marked examples 1/2 as well. We achieve this as follows:

$$D_2(x) = \begin{cases} \frac{D_1(x)}{2(1-p)} & \text{if } h_1(x) \text{ is correct,} \\ \frac{D_1(x)}{2p} & \text{if } h_1(x) \text{ is incorrect.} \end{cases}$$

We pick our second sample, S_2, from this distribution, and use A to come up with a hypothesis h_2. Finally, our last distribution, D_3, consists only of those examples x for which $h_1(x) \neq h_2(x)$ (examples on which h_1 and h_2 disagree). We take a sample S_3 from this distribution and use A to construct a hypothesis h_3 from it. Our final hypothesis then becomes $h = Maj(h_1, h_2, h_3)$.

We now bound the error of our new hypothesis. First assume that the hypotheses h_1, h_2, and h_3 are independent. Then the probability that both h_1 and h_2 err on x is p^2, and the probability that h_3 errs and that h_1 and h_2 disagree on input x is $2p^2(1-p)$. Therefore, the total probability of error is at most $3p^2 - 2p^3$. This is strictly less than p when $p \in (0, 1/2)$. Thus there exists $\gamma' > \gamma$ such that the error probability of our new hypothesis h is at most $1 - \gamma'$.

We now show that we get the same error probability when our three intermediate hypotheses are not independent. Call the probability that h_1 and h_2 both err on x m_1, and the probability that h_1 is correct and h_2 errs on x m_2. The probability that h_1 and h_2 disagree can then be epxressed as $p - m_1 + m_2$. To see that, first notice that p is the probability that h_1 errs. If we subtract m_1 from it, we get the probability that h_1 errs and h_2 doesn't. Finally, we add the probability that h_2 errs and h_1 doesn't. We can do this because the three events we mentioned are disjoint.

By construction of D_2, $\text{error}_{D_2}(h_2) = \frac{m_1}{2p} + \frac{m_2}{2(1-p)}$, which is equal to p by assumption. Then $m_1 = 2p^2 - \frac{m_2 p}{1-p}$. Now the probability that h errs on x is $m_1 + p(p - m_1 + m_2)$. The first term is the probability that h_1 and h_2 both err. The second term is the probability that h_3 errs and that h_1 and h_2 disagree. This is equal to $(1-p)m_1 + p^2 + pm_2 = (1-p)2p^2 - m_2 p + p^2 + pm_2 = 3p^2 - 2p^3$, which is what we wanted to show.

We can repeat this process recursively to get majorities of majorities. Figure 26.3.1 shows the recursion tree. When the recursion tree becomes high enough, we will get a hypothesis that makes an error with probability at most ϵ. At each level, we recursively apply the approach described earlier and take the majority of the majorities one level lower in the tree. We obtain the hypotheses that appear in the leaves by sampling from a distribution and running the algorithm A. We obtain the hypotheses that are in the nodes (labeled by the majority function in Figure 26.3.1) of the tree recursively. At a given node, we are given a distribution D. We use the first child of that node (we make a recursive call) to produce a hypothesis h_1 using $D_1 = D$ as the distribution. After that, we skew D_1 to get D_2 as discussed before, and make another recursive call to get a hypothesis h_2 using D_2 as the distribution (this is represented by the second child of a node). Finally, we construct the distribution D_3 and make our final recursive call to get a hypothesis h_3. We output the hypothesis that takes the majority of h_1, h_2, and h_3. The order in which we find the intermediate hypotheses is determined by a preorder traversal of the tree in Figure 26.3.1.

We now show that the tree has logarithmic depth. We will do this in two stages. First, we find the number of levels sufficient to get $\gamma' \geq 1/4$. After that, we assume $\gamma \geq 1/4$ and find the number of levels necessary to get $\gamma' = 1/2 - \epsilon$.

We saw that after getting three hypotheses, our new combined hypothesis has an error probability

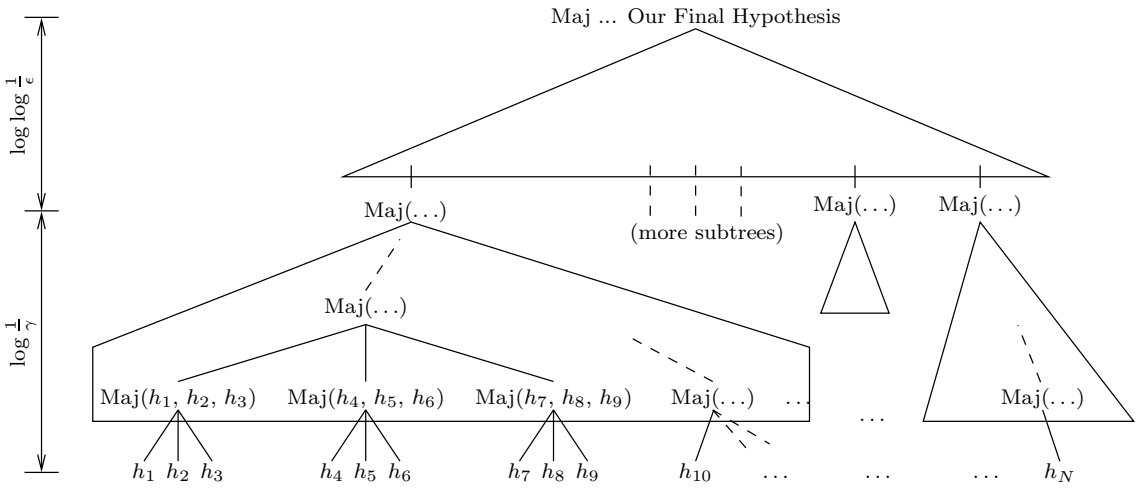

Figure 26.3.1: Recursion tree representing all the majorities we have to compute in order to create our final hypothesis. Each subtree in the bottom part corresponds to producing one hypothesis with error probability at most $1/2 - \gamma$ where $\gamma > 1/4$. The top of the tree then represents taking all these hypotheses and producing a hypothesis with error probability at most ϵ. The total height of the tree is $\mathcal{O}(\log \frac{1}{\gamma} + \log\log \frac{1}{\epsilon})$.

of $3p^2 - 2p^3 = p[3p - 2p^2]$. This means that to get an upper bound on the error probability of the new hypothesis, we multiply the error probability of the old hypothesis by $3p - 2p^2$. If we use the fact that $p = 1/2 - \gamma$, we get that the new error probability p' is at most $p \cdot 2(\frac{1}{2} - \gamma)(1 + \gamma) = (\frac{1}{2} - \gamma) \cdot 2(\frac{1}{2} - \gamma)(1 + \gamma)$. To get the new value of γ, we express $\gamma' = 1 - p'$ in terms of γ, which gives us $\gamma' = \gamma(\frac{3}{2} - 2\gamma^2)$, so we can obtain the next value of γ from the previous one if we multiply it by $\frac{3}{2} - 2\gamma^2$. Since $\gamma < 1/4$ by assumption, $\frac{3}{2} - 2\gamma^2 > \frac{11}{8}$. This means that each level of recursion increases the value of γ by a constant factor. To find the number of levels of recursion before γ is at least $1/4$, we solve the inequality $\gamma(\frac{11}{8})^k > \frac{1}{4}$ for k.

$$\begin{aligned} \gamma\left(\frac{11}{8}\right)^k &> \frac{1}{4} \\ \log\gamma + k\log\frac{11}{8} &> \log\frac{1}{4} \\ k &> \frac{\log\frac{1}{4} - \log\gamma}{\log\frac{11}{8}} \end{aligned}$$

We can rewrite the numerator of the last fraction as $\log\frac{1}{4} - \log\gamma = \log\frac{1}{4\gamma} = \log\frac{1}{\gamma} - \log 4 = \mathcal{O}(\log\frac{1}{\gamma})$. Since the denominator of that fraction is constant, we can conclude that $k = \mathcal{O}(\log\frac{1}{\gamma})$. Thus, to get $\gamma' > 1/4$ (and $p' < 1/4$), we need $\mathcal{O}(\log 1/\gamma)$ levels of recursion.

Now p is at most $1/4$, so $\gamma > 1/4$. After an additional level of recursion, we will have $p' = p(3p - 2p^2) < 3p^2$. Then let p_1 be the error probability of the hypothesis one level of recursion higher than the hypothesis with error probability $p < 1/4$, and define p_2 similarly using p_1 instead of p. In general, we have $p_i < 3p_{i-1}^2$. Then

$$
\begin{aligned}
p_1 &< 3p^2 = 3^1 p^2 < (\tfrac{3}{4})^1 p \\
p_2 &< 3p_1^2 < 3(3p^2)^2 = 3^3 p^4 < (\tfrac{3}{4})^3 p \\
p_3 &< 3p_2^2 < 3(3p_1^2)^2 = 3^3 p_1^4 < 3^3(3p^2)^4 = 3^7 p^8 < (\tfrac{3}{4})^7 p \\
&\vdots \\
p_k &< (\tfrac{3}{4})^{2^k-1} p
\end{aligned}
$$

The maximum value of p is $1/4$, and we would like $p_k < \epsilon$. To find a lower bound on k, we need to solve the inequality below.

$$
\begin{aligned}
\frac{1}{4}\left(\frac{3}{4}\right)^{2^k-1} &< \epsilon \\
(2^k-1)\log\frac{3}{4} &< \log 4\epsilon \\
2^k &> \frac{\log 4\epsilon + \log\frac{3}{4}}{\log\frac{3}{4}} \\
k &> \log\left(\frac{\log 4\epsilon + \log\frac{3}{4}}{\log\frac{3}{4}}\right) \\
&> \log\left(\frac{\log 4\epsilon + \log\frac{3}{4}}{-1}\right) \\
&> \log\left(-\log\frac{3}{4} - \log 4\epsilon\right) \\
&> \log\left(-\log\frac{3}{4} + \log\frac{1}{4\epsilon}\right) \\
&> \log\left(-\log\frac{3}{4} + \log\frac{1}{\epsilon} - \log 4\right) \\
&= \mathcal{O}\left(\log\log\frac{1}{\epsilon}\right).
\end{aligned}
$$

Therefore, we need additional $\log\log 1/\epsilon$ levels of recursion to turn γ into $1/2-\epsilon$ and get a hypothesis with error probability of at most ϵ.

We have left out the discussion of sampling from D_2 and D_3. It can be shown that we can sample from D_2 and D_3 efficiently if p is not too close to 0 or 1. Since p is at most $1/2$, the only case we have to worry about is when p is close to 0. But if p is close to zero, we can stop and use the hypothesis whose error probability is p (close to zero) as our hypothesis that has low error probability.

26.4 Closing Remarks

The recursive approach from the previous section requires us to take too many samples. It turns out that we can repeat the process of picking D_2 by creating the distribution D_{i+1} from D_i and

$h_i$using

$$D_{i+1}(x) = \begin{cases} \frac{D_i(x)}{2(1-p)} & \text{if } h_i(x) \text{ is correct,} \\ \frac{D_i(x)}{2p} & \text{if } h_i(x) \text{ is incorrect.} \end{cases} .$$

Intuitively, in each step, we give all the incorrectly marked examples more weight before we take another sample and produce another hypothesis.

After repeating this process for $(1/\epsilon^2)\log(1/\gamma)$ steps, we let our final hypothesis h be the majority vote of the hypotheses created so far. This boosting algorithm is called Adaboost. It can be shown that the hypothesis produced by Adaboost makes an error with probability at most ϵ. Adaboost also yields a much simpler hypothesis, namely a majority of one set of hypotheses, whereas the algorithm we explain in Section 26.3 produces a complex hypothesis that requires us to recursively find majorities of majorities of smaller sets of hypotheses.

26.5 Next Time

So far we have pushed the discussion of sampling from various distributions aside. In the next lecture, we will talk more about picking samples from various distributions. After that, we will talk about random walks and Markov chains.

References

[1] M. Kearns and U. Vazirani. An Introduction to Computational Learning Theory *The MIT Press*, 1994.