Computational Learning Theory **Fall Semester, 2010**

Lecture 6: November 21, 2010

Lecturer: Yishay Mansour
Scribe: Yoel Sharf, Livnat Jerby, Alon Ardenboim [1]

6.1 Weak and Strong Learners

In the PAC model there is a distribution D on X. Random examples $< x, c_t(x) >$ are drawn according to the distribution D and the target function $c_t \in C$. The goal is to find a hypothesis $h \in H$ such that $error(h, c_t) \leq \epsilon$, with probability $1 - \delta$.
This is a *strong learning model*, since ϵ and δ can be arbitrarily small.

Recall that ϵ is the error rate of the algorithm and $1 - \delta$ represents the confidence. However, suppose we have an algorithm with low error rate but also low confidence, say confidence 50%, or alternatively an algorithm with an error rate of 49% (slightly better than flipping a coin) but high confidence level.

Is it possible to drive those *weak* algorithms to be *strong learners* ? Intuitively, it is easier to find hypothesis that is correct only 51 percent of the time, rather than a hypothesis that is correct 99 percent of the time.

6.1.1 Do we need to require "for all δ"?

Suppose algorithm A attains confidence $\delta = \frac{1}{2}$, is it possible to build a PAC learning algorithm A' (from A)? The answer is Yes.

Algorithm A':

1. Run A $k = log\frac{2}{\delta}$ times(on fresh data each time) with parameter $\epsilon' = \frac{\epsilon}{2}$.
   Algorithm A outputs hypothesis $h_1 \dots h_k$.

2. Draw a new sample S of size $m = \frac{2}{\epsilon^2} \ln \frac{4k}{\delta}$ and test each hypothesis h_i on it.
   Return $h_i^* = min(error(h_i(S)))$.

[1] Based on scribes written by Tamir Berler and Ron Moshe Hecht (May 18, 2002)

Analysis of Algorithm A'

After the first stage of the algorithm, with probability at most $(\frac{1}{2})^k$, $\forall i : error(h_i) > \frac{\epsilon}{2}$. Hence, with probability at least $1 - (\frac{1}{2})^k$, $\exists i : error(h_i) \leq \frac{\epsilon}{2}$.
If we set $k = log\frac{2}{\delta}$, we will get that with probability $1-\frac{\delta}{2}$ for one of $h_1 \dots h_k$ holds $error(h_i) \leq \frac{\epsilon}{2}$.

Now we will show that after the second stage of the algorithm A', with probability $1 - \frac{\delta}{2}$, outputs the hypothesis h_i^* (with minimum errors on S) such that,

$$error(h_i^*) \leq \frac{\epsilon}{2} + min(error(h_i)) \leq \epsilon \quad .$$

Proof: First, we use Chernoff Bound to bound the probability for "bad" event, i.e., the difference between the empirical error of any h_i and its real error is grater than $\frac{\epsilon}{2}$:

$$Pr[|\widehat{error(h_i)} - error(h_i)| \geq \frac{\epsilon}{2}] \leq 2e^{-2(\frac{\epsilon}{2})^2 m}$$

Second, we will bound the probability that such bad event will happen to any of those h_is by $\frac{\delta}{2}$,

$$2e^{-(\frac{\epsilon}{2})^2 m} k \leq \frac{\delta}{2} \quad .$$

Then, by isolating m, we will get:

$$\frac{1}{e^{\frac{\epsilon^2}{2}m}} \leq \frac{\delta}{4k}$$
$$\frac{4k}{\delta} \leq e^{\frac{\epsilon^2}{2}m}$$
$$\ln \frac{4k}{\delta} \leq \frac{\epsilon^2}{2} m$$
$$\frac{2}{\epsilon^2} \ln \frac{4k}{\delta} \leq m \quad .$$

We have that sample with size at least m, then with probability $1 - \frac{\delta}{2}$, for each of those h_i will hold:

$$\widehat{error(h_i)} - error(h_i) < \frac{\epsilon}{2}$$

thus

$$error(h_i^*) - min(error(h_i)) < \frac{\epsilon}{2} \quad .$$

From the first stage of the algorithm we already know that $min(error(h_i)) \leq \frac{\epsilon}{2}$, hence:

$$error(h_i^*) < \frac{\epsilon}{2} + min(error(h_i)) \leq \epsilon$$

□

6.1.2 Do we need to require "for all ϵ"?

One question we can ask: given an algorithm that outputs hypothesis with $\epsilon = \frac{1}{2}$, can we drive it to learn PAC? The answer is No, because such an algorithm will do exactly like flipping a coin.

Definition: Weak learning

Algorithm A learn Weak-PAC a concept class C with H if:
$\forall c_t \in C$,
$\forall D$,
$\forall \delta > \frac{1}{2}$,
$\exists \gamma > 0$,
Algorithm A outputs hypothesis $h \in H$ and with probability $1-\delta$, such that $error(h) \leq \frac{1}{2} - \gamma$.

Intuitively, A will guarantee an error rate of 49% instead of 1%. We show, that if a concept class has a weak learning algorithm, then there is a PAC learning algorithm for the class.
Note that running A multiple times, does not work because A might return the same hypothesis over and over again.

Example

Suppose the following target function c_t (over bits):

if $x_1 = x_2 = 1 \Longrightarrow$ some very hard function

otherwise $\Longrightarrow c_t = 0$

(e.g. it all depends on the first and the second bits.)

Then it's very easy to get error rate of 12.5% over uniform distribution examples. We will simply predict always zero.
This is true because the probability for the event $x_1 = x_2 = 1$ is 0.25, and then we predict half the time correctly. For other values of x_1 and x_2 we are correct.
Although it's easy to achieve 87.5% accuracy, it is very difficult to achieve more, from our assumption on the hard function.

Conclusion: An important requirement in weak learning model is: *for all distribution.* (in the example we assumed a specific distribution)

Proof of Equivalence

Preliminary: We're going to concentrate on the case of a sample with finite size, so that, given $x_1 \dots x_m$ goal is to find a hypothesis classifies them all right.

Given: $x_1 \dots x_m$ and their labels,

H - A weak hypothesis class,

We will use a Regret Minimization algorithm RM.

The Algorithm:

1. The algorithm basic action: to choose a distribution over $x_1 \dots x_m$.
2. For each step t, the RM(Regret Minimization) algorithm chooses a distribution (over $x_1 \dots x_m$).
3. For each step t, the opponent returns loss (for each x_i) such that, $error(h_t, D_t) \leq \frac{1}{2} - \gamma$. The loss to be returned will be 1 for correct classification and 0 for error (opposite of what we expect).
4. After T steps, we will require that $MAJ(h_1(x) \dots h_T(x))$ classify all the sample correctly.

Notes:

1. Since at each stage we return a weak-learner h, that is classified correctly at least $\frac{1}{2} + \gamma$ fraction, then the loss of RM is at least $(\frac{1}{2} + \gamma)T$.
2. Suppose that there is x_i such that MAJ does not classify it correctly, then the loss of x_i is at most $\frac{T}{2}$.
3. $loss(RM) = (\frac{1}{2} + \gamma)T \leq \frac{T}{2} + 2\sqrt{T \log m}$.
   ($2\sqrt{T \log m}$ is the regret of RM)
   For this inequality to hold, it is required that:

$$\gamma T \leq 2\sqrt{T \log m}$$

$$T \leq \frac{4 \log m}{\gamma^2}$$

4. Conclusion: If you execute the above RM algorithm, after $\frac{4\log m}{\gamma^2}$ iterations you will get a consistent hypothesis.
   Thus, by *Occam Razor Theorem*, we can PAC learn the class.

5. RM will naturally concentrate on examples with the "most errors".

6.2 Recursive Construction

6.2.1 Algorithm Description

Let A be a weak learning algorithm, and p the error probability of A.
Step I: Run A with the initial distribution D_1 to obtain h_1. The idea is to define a new distribution D_2, such that

$$S_c = \{x|h_1(x) = c_t(x)\}, S_i = \{x|h_1(x) \neq c_t(x)\}$$
$$P_{D_2}[x|h_1(x) = c_t(x)] = \frac{1}{2}, P_{D_2}[x|h_1(x) \neq c_t(x)] = \frac{1}{2}$$

To do so we will define D_2 as follows:

$$D_2(x) = \begin{cases} \frac{0.5}{1-p} \cdot D_1(x) & x \in S_c \\ \frac{0.5}{p} \cdot D_1(x) & x \in S_i \end{cases}$$

To obtain h_2 we will run A with D_2. The distribution D_3 would be defined on examples x for which $h_1(x) \neq h_2(x)$, and we would obtain h_3, i.e. $P_{D_3}[x|h_1(x) \neq h_2(x)] = P_{D_1}[x|h_1(x) \neq h_2(x)]$. Our combined hypothesis would be:

$$H(x) = \begin{cases} h_1(x) & h_1(x) = h_2(x) \\ h_3(x) & otherwise \end{cases}$$

6.2.2 Estimation of the Error

Suppose each hypothesis h_i errors with a probability of p, independently. What would be the error of the majority of h_1, h_2, h_3?

$$Error = 3p^2(1-p) + p^3 = 3p^2 - 2p^3$$

We would like to show that this is the error probability without assuming the hypotheses are independent. To do so we would divide the space into four subspaces:

$$S_{cc} = \{x|h_1(x) = c_t(x) \wedge h_2(x) = c_t(x)\}, S_{ii} = \{x|h_1(x) \neq c_t(x) \wedge h_2(x) \neq c_t(x)\}$$
$$S_{ci} = \{x|h_1(x) \neq c_t(x) \wedge h_2(x) = c_t(x)\}, S_{ic} = \{x|h_1(x) = c_t(x) \wedge h_2(x) \neq c_t(x)\}$$

The error probability, with respect to the initial distribution D_1, is $P_{ii} + (P_{ic} + P_{ci})p$. Let us define $\alpha = D_2(S_{ci})$. Therefore, in terms of D_1 we get $P_{ci} = 2(1-p)\alpha$.We have,

$$D_2(S_{ii}) = p - \alpha$$
$$P_{ii} = 2p(p - \alpha).$$

From the construction of D_2,

$$D_2(S_{ic}) = \frac{1}{2} - (p - \alpha)$$
$$P_{ic} = 2p(\frac{1}{2} - p + \alpha).$$

Therefore the error is: $P_{ii} + (P_{ic} + P_{ci})p = 2p(p-\alpha) + p(2p(\frac{1}{2} - p + \alpha) + 2(1-p)\alpha) = 3p^2 - 2p^3$. Given that the initial probability is $p_0 = \frac{1}{2} - \gamma_0$, each step improves upon the previous step:

$$\frac{1}{2} - \gamma_{t+1} = 3(\frac{1}{2} - \gamma_t)^2 - 2(\frac{1}{2} - \gamma_t)^3$$
$$\frac{1}{2} - \gamma_{t+1} \geq \frac{1}{2} - \gamma_t(\frac{3}{2} - \gamma_t^2)$$

The termination condition of the recursion would be to obtain an error of ϵ as required. For $\gamma_t > \frac{1}{4}$ we get that $p < \frac{1}{4}$, and therefore

$$p_{t+1} \leq 3p_t^2 - 2p_t^3 \leq 3p_t^2 < \frac{1}{2}$$

The recursion depth is therefore $O(log\frac{1}{\gamma} + log(log\frac{1}{\epsilon}))$, and the number of nodes in the recursion tree is $3^{depth} = poly(\frac{1}{m}, log\frac{1}{\epsilon})$.□

6.3 AdaBoost

The AdaBoost algorithm is an iterative boosting algorithm that enables us to create a strong learning algorithm from a weak learning algorithm. The general idea of this algorithm is to maintain a distribution on the input sample, and increase the weight of the harder to classify examples so the algorithm would focus on them.

6.3.1 Algorithm Description

Input: A set of m classified examples: $S = \{< x_1, y_1 >, < x_2, y_2 >, \cdots, < x_m, y_m >\}$ where $y_i \in \{0, 1\} \quad \forall i \in \{1, \cdots, m\}$.

Let's denote D_t the distribution of weights of the examples at time t, and $D_t(i)$ the weight of example x_i at time t.
At the beginning, we initialize:

$$D_1(i) = \frac{1}{m} \quad \forall i \in \{1, \cdots, m\}$$

At each iteration we'll look for a classifier $h_t : X \mapsto \{-1, +1\}$ that minimizes the error on the current distribution (defined as $\epsilon_t = \Pr_{D_t}[h_t \neq c_t]$ where c_t is the target function).
At time $t + 1$ we update the weights in the following manner:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & y_i = h_t(x_i) \\ e^{-\alpha_t} & y_i \neq h_t(x_i) \end{cases} \\ &= \frac{D_t(i)}{Z_t} \cdot e^{-y_i \alpha_t h_t(x_i)} \end{aligned}$$

where Z_t is a normalizing factor to keep D_{t+1} a distribution and $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$. The hypothesis we return after running the algorithm for T iterations is:

$$H(x) = \text{Sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

An advantage using the AdaBoost algorithm is that it removes the need of knowing the parameter γ. Another advantage is that it's easy to implement and runs efficiently.

6.3.2 Bounding the Error

Theorem 6.1 *Let H be the output hypothesis of AdaBoost. Then:*

$$\begin{aligned} \widehat{error}(H) &\leq \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)} \\ &= \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \\ &\leq e^{-2\sum_t \gamma_t^2} \end{aligned}$$

where the last line is obtained from the inequality $1 + x \leq e^x$. *Thus, if* $\gamma_t \geq \gamma \ \forall t$, *then:*

$$\widehat{error} \leq e^{-2\gamma^2 T}$$

so that the error dies exponentially fast in T.

Proof: The proof follows in three steps:
1. First, obtain the following expression for $D_{T+1}(i)$:

$$D_{T+1}(i) = \frac{D_1(i)e^{-y_i f(x_i)}}{\prod_t Z_t}$$

where $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$.

Proof: Since $D_{t+1}(i)$ is given by:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} e^{-y_i \alpha_t h_t(x_i)}$$

we can unravel the recurrence to obtain:

$$\begin{aligned} D_{T+1}(i) &= \frac{1}{m} \prod_{t=1}^{T} \frac{e^{-y_i \alpha_t h_t(x_i)}}{Z_t} \\ &= \frac{e^{-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)}}{m \prod_{t=1}^{T} Z_t} \\ &= \frac{e^{-y_i f(x_i)}}{m \prod_{t=1}^{T} Z_t} \end{aligned}$$

□

2. Second, we bound the training error of H by the product of the normalizing weights Z_t:

$$\widehat{error}(H) \leq \prod_{t=1}^{T} Z_t$$

Proof:

$$\begin{aligned}
\widehat{error}(H) &= \frac{1}{m}\sum_{i=1}^{m} I(y_i \neq H(x_i)) \\
&= \frac{1}{m}\sum_{i=1}^{m} I(y_i f(x_i) \leq 0) \\
&\leq \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(x_i)} \\
&= \frac{1}{m}\sum_{i=1}^{m} m\left(\prod_{t=1}^{T} Z_t\right) D_{T+1}(i) \\
&= \left(\prod_{t=1}^{T} Z_t\right)\sum_{i=1}^{m} D_{T+1}(i) \\
&= \prod_{t=1}^{T} Z_t,
\end{aligned}$$

where I is the indicator function. The third line follows from the observation that when $I(y_i f(x_i) \leq 0) = 1$, then $y_i f(x_i) \leq 0$ and so $e^{-y_i f(x_i)} \geq 1 = I(y_i f(x_i) \leq 0)$. (Also, clearly when $I(y_i f(x_i) \leq 0) = 0$, then $e^{-y_i f(x_i)} \geq 0$). The fourth line follows from step 1. The last line is obtained from the fact that D_{T+1} is a probability distribution over the examples. □

3. Now that the training error has been bounded in step 2 by the product of the normalizing weights Z_t, the last step is to express Z_t in terms of ϵ_t:

$$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Proof: By definition,

$$\begin{aligned}
Z_t &= \sum_{i=1}^{m} D_t e^{-y_i \alpha_t h_t(x_i)} \\
&= \sum_{i:y_i = h_t(x_i)} D_t(i) e^{-\alpha_t} + \sum_{i:y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t} \\
&= (1-\epsilon_t)e^{-\alpha_t} + \epsilon_t e^{\alpha_t},
\end{aligned}$$

where the last step follows from the fact that:

$$\sum_{i:y_i \neq h_t(x_i)} D_t(i) = \epsilon_t,$$

since the sum is taken over misclassified examples, which gives the error for the t^{th} round. Since the expression above for Z_t is valid for all α_t, minimizing Z_t with respect to α_t for each t will produce the minimum training error $\widehat{error}(H)$. Taking the derivative of Z_t with respect to α_t and setting equal to zero, we obtain:

$$-(1-\epsilon_t)e^{-\alpha_t} + \epsilon_t e^{\alpha_t}$$

Solving, we find:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right).$$

Using this value of α_t in the expression for Z_t, and then plugging that into the bound on the training error for H, we end up with:

$$\widehat{error}(H) \leq \prod_{t=1}^{T}\left(2\sqrt{\epsilon_t(1-\epsilon_t)}\right)$$

which proves the theorem. □

□