67577 – Intro. to Machine Learning **Fall semester, 2003/4**

Lecture 8: Boosting Algorithms

Lecturer: Yoram Singer *Scribe: Yoram Singer* [1]

This lecture focuses on the idea of boosting. Algorithms in PAC learning models can produce hypothesis with arbitrary error rate, as long as sample complexity is satisfied. However, suppose a learning algorithm can do a little bit better than random, that is, its error rate is less than 50%, can we take this error rate and drive it down to zero? We start by repeating the (standard) definition of PAC learnability and then introduce the notion of weak-learnability.

Definition 1: $\mathcal{C}$ is learnable if $\exists$ algorithm A such that, $\forall c \in \mathcal{C}$, $\forall D$, $\forall \epsilon > 0$, $\forall \delta > 0$, whenever A is given $m = poly(\frac{1}{\epsilon}, \frac{1}{\delta})$ examples $(x_1, c(x_1)), \ldots (x_m, c(x_m))$, A returns h for which $Pr_D[err(h) > \epsilon] \leq \delta$.

Definition 2: $\mathcal{C}$ is efficiently learnable if A runs in time polynomial in $\frac{1}{\epsilon}, \frac{1}{\delta}, s, n$, where s is the size of the target concept, and n is the size of the instances. For example, n is the instance size when the domain $\mathrm{X_n}$ is $\{0,1\}^n$ or $\Re^n$.

Definition 3: $\mathcal{C}$ is weakly learnable if $\exists \gamma > 0$, $\exists$ algorithm A, such that, $\forall c \in \mathcal{C}$, $\forall D$, $\forall \delta > 0$, whenever A is given $m = poly(\frac{1}{\delta})$ examples $(x_1, c(x_1)), \ldots (x_m, c(x_m))$ A returns h for which $Pr_D[err(h) > \frac{1}{2} - \gamma] \leq \delta$

A weak learner can be trivial. For example if given a sample set of more than 60% positive examples, a learner can output a hypothesis that always predicts positive. However such a concept class is not weakly learnable since the latter is defined as having an error rate slightly smaller than 50% on *any* distribution of examples. Thus, if we use an equalized distribution (mass of positive examples is equal to the mass of negative examples), then clearly the cannot reach an error different than 50%.

Weak learnability does not necessarily mean strong learnability given a *fixed* distribution. For example, let $\mathcal{C}$ be all boolean functions on $\{0,1\}^n \cup \{Z_0\}$, let the distribution be $\frac{1}{4}$ on the point Z_0 and uniform on all the remaining points. Take a sample of size m, Z_0 is likely to be included. Also included are a tiny fraction of other points because m is small compared to the 2^n possibilities of 0/1 string. Assuming Z_0 got its label correct, the total error rate in this case is roughly $\frac{3}{4} \cdot \frac{1}{2} = \frac{3}{8} < \frac{1}{2}$ because we are getting so few samples on the other points that we are merely guessing. On the other hand, for this fixed distribution, there is really no way of driving the error rate significantly below $\frac{3}{8}$ with a polynomial size sample.

Boosting is the idea of converting weak learning to strong learning. The pseudo-code of a boosting algorithm called *AdaBoost* is given below. The code is for the case of discrete binary hypotheses of the form $h_t : \mathrm{X} \to \{-1, +1\}$. The algorithm receives a sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ where $x_i \in \mathrm{X}, \mathrm{y_i} \in \{-1, 1\}$. The boosting algorithm also gets access to a weak learning algorithm A, which $\forall D$, returns (computes) h, such that $Pr[err_D(h) \leq \frac{1}{2} - \gamma] \geq 1 - \delta$. The goal of th boosting algorithm is to achieve an arbitrary low empirical error. The key point is that the booster call A several times where on each round it feeds A with a different distribution. (Otherwise the learner A could output the same h every time.) Therefore for each running of A, we will change D to force A to learn something new every time. Each time A will output a hypothesis $h_t, t \in [1 \ldots T]$. In the

[1] Based on course notes by Rob Schapire and Jie Chen.

end we combine all the hypotheses h_t into a final resulting H. We denote by D_t the distribution constructed on given examples while $D_t(i)$ is the weight on example (x_i, y_i). The general AdaBoost of boosting goes like this:

$D_1(i) = \frac{1}{m}$
For $t = 1 \dots T$
  Run A on D_t and get $h_t : \mathrm{X} \to \{-1, 1\}$
  Compute $\epsilon_t = Pr_{D_t}[h_t(x_i) \neq y_i] = \epsilon_t = \frac{1}{2} - \gamma_t \leq \frac{1}{2} - \gamma$
  $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
  $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$
    where Z_t is a normalization constant s.t. $\sum_i D_{t+1}(i) = 1$
EndFor
Output: $H(x) = \mathrm{sign}(\sum_{t=1}^{T} h_t(x) \cdot \alpha_t)$.

Originally the algorithm A had access to all the examples and their labels. To incorporate the distribution D_t, two approaches exist in practice: (1) the algorithm picks a set of examples according to D_t; (2) the algorithm A tries to directly minimize weighted training error $\sum_{i:h_t(x_i)\neq y_i} D_t(i)$. AdaBoost is the first practical boosting algorithm. The intuition is we want to focus on the hard example, which is the example that a weak classifier misclassified. So we increase weight on a previous wrong example and lower weight on a previous correct one. Z_t in the formula is simply a normalization factor. We now analyze the empirical error of AdaBoost as a function of ϵ_t. The empirical loss theorem can be states as follows.

Theorem: $\hat{err}(H) \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}$

Based on the theorem it simple to show that,

$$\begin{aligned} \hat{err}(H) &\leq \prod_{t=1}^{T} [2\sqrt{\epsilon_t(1-\epsilon_t)}] \\ &= \exp\left(-\sum_t \mathrm{RE}(\frac{1}{2}||\epsilon_t)\right) \\ &= \prod_t \sqrt{1-4\gamma_t^2} \\ &\leq \exp\left(-2\sum_t \gamma_t^2\right). \end{aligned}$$

Furthermore, if we assume $\gamma_t \geq \gamma$ then

$$\hat{err}(H) \leq e^{-2\gamma^2 T}.$$

The theorem shows the training error would decrease exponentially as the number of times to repeat increases. Specifically, if $T > \frac{1}{2\gamma^2} \ln m$, then $\hat{err}(H) < \frac{1}{m}$, which implies the training error will be zero. Note the theorem does not have any assumptions on h_t and where samples are from. We are going to prove the theorem in 3 steps.

1. Show that
$$D_{T+1}(x_i) = \frac{\exp(-y_i f(x_i))}{m \prod_t Z_t}$$

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

Proof:

$$\begin{aligned} D_{T+1}(i) &= \frac{D_T(i) \cdot \exp(-\alpha_T y_i h_T(x_i))}{Z_T} \\ &= \frac{D_{T-1}(i) \cdot \exp(-\alpha_{T-1} y_i h_{T-1}(x_i)) \cdot \exp(-\alpha_T y_i h_T(x_i)}{Z_{T-1} \cdot Z_T} \\ &\vdots \\ &= \frac{1}{m} \cdot \frac{\exp(-y_i \cdot \sum_t \alpha_t h_t(x_i))}{\prod_t Z_t} \end{aligned}$$

2. Show that

$$\hat{err}(H) \leq \prod_t Z_t$$

Proof:

$$\begin{aligned} \hat{err}(H) &= \frac{1}{m} \cdot \sum_{i=1}^{m} [\![y_i \neq H(x_i)]\!] \\ &= \frac{1}{m} \cdot \sum_{i=1}^{m} [\![y_i f(x_i) \leq 0]\!] \\ &\leq \frac{1}{m} \sum_{i=1}^{m} e^{-y_i f(x_i)} \\ &= \frac{1}{m} \cdot \sum_{i=1}^{m} D_{T+1}(i) \cdot m \cdot \prod_t Z_t \\ &= \frac{1}{m} \prod_t Z_t \cdot \sum_{i=1}^{m} D_{T+1}(i) \cdot m \\ &= \prod_t Z_t. \end{aligned}$$

Here, $[\![\pi]\!]$ is 1 if π is true, and 0 otherwise. Note the third line follows because for each term that $y_i f(x_i) \leq 0$, the corresponding $e^{-y_i f(x_i)}$ is greater than 1 while for each term $y_i f(x_i) > 0$ the term $e^{-y_i f(x_i)}$ is greater than zero. The fourth line comes from step 1 above.

3. Finally we show that

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

Proof:

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

The value α_t was in fact chosen so that Z_t is minimized. In this case,

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

and

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

Note the third line follows the definition of weighted training error.

The remaining question is whether AdaBoost also generalizes well. We leave this question to a more advanced course on learning theory...