E0 370 Statistical Learning Theory **Lecture 15 (Oct 20, 2011)**

Boosting

Lecturer: Shivani Agarwal *Scribe: Anil C R*

1 Introduction

We start by discussing some variants of the learnability model we saw in the last few lectures, and consider whether efficient learnability in these new models implies efficient learnability in the original model, and vice-versa. We then discuss boosting and the popular AdaBoost algorithm.

2 Equivalence of Models for Polynomial (Efficient) Learnability

Recall the following definition of efficient learnability. Let $\mathcal{H} = \cup_{n=1}^{\infty} \mathcal{H}_n$, where $\mathcal{H}_n \subseteq \{\pm 1\}^{\mathcal{X}_n}$ (with $\mathcal{X}_n$ being a domain whose instances can be encoded using a representation of size polynomial in n). Let size : $\mathcal{H} \to \mathbb{N}$, and let $\mathcal{H}_{n,s} = \{h \in \mathcal{H}_n \mid \text{size}(h) \leq s\}$. Then $\mathcal{H}$ is *efficiently learnable* w.r.t. domain dimension n and target complexity 'size' if there exists an algorithm $\mathcal{A} : \cup_{n=1}^{\infty} \cup_{t \in \mathcal{H}_n} \cup_{m=1}^{\infty} (\mathcal{X}_n \times t)^m \to \cup_{n=1}^{\infty} \{\pm 1\}^{\mathcal{X}_n}$, which given a training sample $S \in (\mathcal{X}_n \times t)^m$ returns as output a function $h_S \in \{\pm 1\}^{\mathcal{X}_n}$, such that for all $\epsilon, \delta \in (0, 1]$, $n \in \mathbb{N}$, and $s \in \mathbb{N}$, $\exists m_0 = m_0(\epsilon, \delta, n, s)$ such that for all distributions μ on $\mathcal{X}_n$ and $t \in \mathcal{H}_{n,s}$, for all $m \geq m_0$:

$$\mathbf{P}_{S \sim (\mu \times t)^m} \left(\text{er}_{\mu \times t}[h_S] \geq \epsilon \right) \leq \delta \,, \tag{1}$$

and moreover, $m_0(\epsilon, \delta, n, s) = \text{poly}(\frac{1}{\epsilon}, \frac{1}{\delta}, n, s)$ and the running time of $\mathcal{A}$ on any $S \in (\mathcal{X}_n \times t)^m$ is $\text{poly}(m, n)$. Note that here we allow the learning algorithm to return *any* function in $\{\pm 1\}^{\mathcal{X}_n}$, not necessarily a function in $\mathcal{H}_n$; in other words, we do not insist on *proper* learnability of $\mathcal{H}$, but rather allow for learnability of $\mathcal{H}$ by $\cup_{n=1}^{\infty} \{\pm 1\}^{\mathcal{X}_n}$. Consider the following variants of this model:

1. **Logarithmic dependence on $\frac{1}{\delta}$.** Suppose we require $m_0(\epsilon, \delta, n, s) = \text{poly}(\frac{1}{\epsilon}, \ln(\frac{1}{\delta}), n, s)$, i.e. the sample complexity is required to have a poly-*logarithmic* dependence on $\frac{1}{\delta}$ rather than a polynomial dependence. Clearly, if a function class $\mathcal{H}$ is efficiently learnable in this new model, then it is also efficiently learnable in the original model. In fact, it is not hard to show that the two models are equivalent: efficient learnability in the original model also implies efficient learnability in the new model. (To see this, start with an (ϵ, δ)-learning algorithm with sample complexity $\text{poly}(\frac{1}{\epsilon}, \frac{1}{\delta}, n, s)$, and set $\delta_0 = \frac{1}{2}$; this gives an $(\epsilon, \frac{1}{2})$-learning algorithm with sample complexity $\text{poly}(\frac{1}{\epsilon}, n, s)$. Now run this algorithm $O(\ln(\frac{1}{\delta}))$ times, and return the function with the smallest empirical error on a new sample of size $O(\frac{1}{\epsilon} \ln(\frac{1}{\delta}))$; the total sample complexity is $\text{poly}(\frac{1}{\epsilon}, \ln(\frac{1}{\delta}), n, s)$ as desired, and it can be verified that this yields an $(O(\epsilon), \delta)$-learning algorithm.)

2. **Two-distribution model.** Now suppose we replace the single distribution μ on $\mathcal{X}_n$ by two distributions μ_+, μ_-, representing draws of positive and negative examples respectively (i.e. μ_+ is obtained by conditioning μ on the event $\{x \in \mathcal{X}_n \mid t(x) = 1\}$, and similarly for μ_-), and require that there exist sample sizes $m_+ = m_+(\epsilon, \delta, n, s)$ and $m_- = m_-(\epsilon, \delta, n, s)$, both $\text{poly}(\frac{1}{\epsilon}, \frac{1}{\delta}, n, s)$, such that

$$\mathbf{P}_{S \in (\mu_+ \times 1)^{m_+} \times (\mu_- \times -1)^{m_-}} \left(\text{er}_{(\mu_+ \times 1)}[h_S] \geq \epsilon \right) \leq \delta \tag{2}$$

and

$$\mathbf{P}_{S \in (\mu_+ \times 1)^{m_+} \times (\mu_- \times -1)^{m_-}} \left(\text{er}_{(\mu_- \times -1)}[h_S] \geq \epsilon \right) \leq \delta \,. \tag{3}$$

It turns out that again, this new model of efficient learnability is equivalent to the original, single-distribution model. We leave the proof as an exercise (see [5] if you need help!).

3. **Weak learnability model.** Consider now a variant where we require only that there exist an algorithm $\mathcal{A}$ and a constant $\gamma \in (0, \frac{1}{2})$ such that for all $\delta \in (0, 1]$, $n \in \mathbb{N}$, and $s \in \mathbb{N}$, $\exists m_0 = m_0(\delta, n, s) = \text{poly}(\frac{1}{\delta}, n, s)$ such that for all distributions μ on $\mathcal{X}_n$, $t \in \mathcal{H}_n$, and $m \geq m_0$,

$$\mathbf{P}_{S \sim (\mu \times t)^m} \left(\text{er}_{\mu \times t}[h_S] \geq \frac{1}{2} - \gamma \right) \leq \delta \,, \tag{4}$$

with running time $\text{poly}(m, n)$ as before; in other words, the algorithm is required to learn (with confidence $1 - \delta$) a given target function with accuracy only slightly higher than $\frac{1}{2}$, not arbitrarily close to 1 as required in the original model. This model is referred to as the *weak learnability* model. Clearly, any function class that is efficiently learnable in the original model (sometimes said to be *strongly* learnable) is also efficiently learnable in this new model (*weakly* learnable). The question is, does weak learnability also imply strong learnability? While it can be shown that this is not true in the distribution-specific setting (i.e. one can construct examples of function classes that are weakly learnable w.r.t. a specific distribution but not strongly learnable w.r.t. the same distribution), surprisingly, it turns out that any function class $\mathcal{H}$ that is weakly learnable in the distribution-free setting as above is also strongly learnable, thus making the two models equivalent. This remarkable result, which involves giving a *boosting* algorithm that can boost the accuracy of any weak learning algorithm to produce a strong learning algorithm, was first shown by Schapire [6]. A more efficient boosting algorithm was provided by Freund [2]. While beautiful from a theoretical standpoint, both these boosting algorithms had a shortcoming from a practical standpoint: namely, both algorithms required as input the accuracy parameter γ associated with the weak learner. Later, Freund and Schapire gave a more practical boosting algorithm that does not require knowledge of γ; this algorithm, known as AdaBoost, has since been extensively studied and is widely used in practice [4, 3, 7].

3 AdaBoost Algorithm

Let $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{X}}$ be a class of 'weak' or 'base' (real-valued) classifiers. Assume a 'weak' or 'base' learning algorithm that given as input a 'weighted' sample (S, D), where $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \{\pm 1\})^m$ and D is a distribution on S, returns a function $f \in \mathcal{F}$ (ideally, with low error w.r.t. D, but as we will see, this is not a hard requirement[1]). Then the basic AdaBoost algorithm works as follows (we use here the form given in [7], which is widely used in practice):

Algorithm **AdaBoost**

Inputs: Training sample $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \{\pm 1\})^m$
Number of iterations T

Initialize: $D_1(i) = \frac{1}{m} \quad \forall i \in [m]$

For $t = 1, \ldots, T$:

- Train weak learner on weighted sample (S, D_t); get weak classifier $f_t : \mathcal{X} \to \mathbb{R}$
- Choose $\alpha_t \in \mathbb{R}$
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i f_t(x_i))}{Z_t}$$

where

$$Z_t = \sum_{j=1}^{m} D_t(j) \exp(-\alpha_t y_j f_t(x_j))$$

Output final hypothesis:

$$h_S(x) \equiv h_T(x) = \text{sign} \left(\underbrace{\sum_{t=1}^{T} \alpha_t f_t(x)}_{F_T(x)} \right)$$

---

[1] In fact, for binary classification, 'weak' classifiers with high error are as good as those with low error; as we will see, the key requirement is that the errors be different from those associated with random guessing.

In order to understand the algorithm, note that

$$\begin{aligned} D_{T+1}(i) &= \frac{D_T(i)\exp(-\alpha_T y_i f_T(x_i))}{Z_T} && (5)\\ &= \frac{D_{T-1}(i)\exp(-\alpha_{T-1} y_i f_{T-1}(x_i))\cdot\exp(-\alpha_T y_i f_T(x_i))}{Z_{T-1}\cdot Z_T} && (6)\\ &\vdots \\ &= \frac{D_1(i)\exp(-y_i\sum_{t=1}^T \alpha_t f_t(x_i))}{\prod_{t=1}^T Z_t} && (7)\\ &= \frac{\frac{1}{m}\exp(-y_i F_T(x_i))}{\prod_{t=1}^T Z_t}\,. && (8) \end{aligned}$$

Observe first that this gives $D_{T+1}(i) \propto \exp(-y_i F_T(x_i))$; indeed for any τ, we have similarly $D_{\tau+1}(i) \propto \exp(-y_i F_\tau(x_i))$. This gives that the distribution $D_{\tau+1}(i)$ assigns weights to examples in S according to the margins $y_i F_\tau(x_i)$ of the function $F_\tau(x) = \sum_{t=1}^\tau \alpha_t f_t(x)$ obtained so far: the smaller the margin (the poorer the classification), the larger the weight, thus forcing the weak learner on the $(\tau+1)$-th iteration to focus on examples not classified accurately by F_τ. Now, since $\sum_{i=1}^m D_{T+1}(i) = 1$, we have from the above equation

$$\sum_{i=1}^m \frac{\frac{1}{m}\exp(-y_i F_T(x_i))}{\prod_{t=1}^T Z_t} \;=\; 1 \qquad (9)$$

or

$$\underbrace{\frac{1}{m}\sum_{i=1}^m \exp(-y_i F_T(x_i))}_{\mathrm{er}_S^{\exp}[F_T]} \;=\; \prod_{t=1}^T Z_t\,. \qquad (10)$$

Recall that the exponential loss $\ell_{\exp} : \{\pm 1\}\times\mathbb{R}\to[0,\infty)$, given by $\ell_{\exp}(y,\hat{y}) = e^{-y\hat{y}}$, forms a convex upper bound on the zero-one loss $\ell_{0\text{-}1}(y,\hat{y}) = \mathbf{1}(\mathrm{sign}(\hat{y}) \neq y)$, which yields

$$\mathrm{er}_S^{0\text{-}1}[h_T] \;\le\; \mathrm{er}_S^{\exp}[F_T] \;=\; \prod_{t=1}^T Z_t\,. \qquad (11)$$

Therefore, one can try to minimize $\prod_{t=1}^T Z_t$, which is equivalent to minimizing the empirical exponential error $\mathrm{er}_S^{\exp}[F_T]$, which in turn forms a convex upper bound on the empirical zero-one error $\mathrm{er}_S^{0\text{-}1}[h_T]$. In particular, in a greedy/coordinate-descent type approach, one chooses α_t so as to minimize Z_t on each iteration t. We discuss this choice of α_t below, first in the special case $f_t(x) \in [-1,1]$, and then in the general case $f_t(x) \in \mathbb{R}$.

3.1 Special Case: $f_t(x) \in [-1,1]$

Let

$$r_t \;=\; \mathbf{E}_{i\sim D_t}\big[y_i f_t(x_i)\big] \;=\; \sum_{i=1}^m D_t(i)\, y_i f_t(x_i)\,. \qquad (12)$$

Now Z_t can be approximated as

$$\begin{aligned} Z_t &= \sum_{i=1}^m D_t(i)\exp(-\alpha_t y_i f_t(x_i)) \\ &\le \sum_{i=1}^m D_t(i)\left(\left(\frac{1+y_i f_t(x_i)}{2}\right)e^{-\alpha_t} + \left(\frac{1-y_i f_t(x_i)}{2}\right)e^{\alpha_t}\right) \\ &= \left(\frac{1+r_t}{2}\right)e^{-\alpha_t} + \left(\frac{1-r_t}{2}\right)e^{\alpha_t}\,, \end{aligned}$$

where the inequality follows from the fact that $f_t(x_i) \in [-1,1]$ and convexity of $\phi(z) = e^{-\alpha_t z}$ (taking $u_i = y_i f_t(x_i)$ and $\lambda = \frac{1+u_i}{2} \in [0,1]$ we get $\phi(u_i) = \phi(\lambda\cdot 1 + (1-\lambda)\cdot(-1)) \le \lambda\phi(1) + (1-\lambda)\phi(-1)$). Note

that for $f_t(x_i) \in \{\pm 1\}$ the inequality is actually an equality and the approximation is exact. Minimizing the RHS as an approximation to minimizing Z_t then gives

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 + r_t}{1 - r_t} \right) . \tag{13}$$

With this choice of α_t, the RHS becomes equal to $\sqrt{1 - r_t^2}$. Therefore, for $f_t(x) \in [-1, 1]$ $\forall t, x$ and $\alpha_t = \frac{1}{2} \ln(\frac{1+r_t}{1-r_t})$, we have that

$$\mathrm{er}_S^{0\text{-}1}[h_T] \;\leq\; \prod_{t=1}^{T} \sqrt{1 - r_t^2} \, . \tag{14}$$

Now note that

$$1 - r_t \;=\; \sum_{i=1}^{m} D_t(i)(1 - y_i f_t(x_i)) \;=\; \sum_{i=1}^{m} D_t(i)|y_i - f_t(x_i)| \;=\; \mathrm{er}_{D_t}^{\mathrm{abs}}[f_t] \;=\; 2\,\mathrm{er}_t \, , \quad \text{say,} \tag{15}$$

where $\mathrm{er}_t \in [0, 1]$. Then

$$\mathrm{er}_S^{0\text{-}1}[h_T] \;\leq\; \prod_{t=1}^{T} \sqrt{1 - r_t^2} \;=\; 2^T \prod_{t=1}^{T} \sqrt{\mathrm{er}_t(1 - \mathrm{er}_t)} \, , \tag{16}$$

which means that whenever $\mathrm{er}_t \neq \frac{1}{2}$ (so that $\mathrm{er}_t(1 - \mathrm{er}_t) < \frac{1}{4}$), (the upper bound on) the empirical error of the learned function reduces. In particular, if $\mathrm{er}_t \leq \frac{1}{2} - \gamma$ $\forall t$ (which corresponds to an accuracy slightly better than random guessing), then we have

$$\mathrm{er}_S^{0\text{-}1}[h_T] \;\leq\; 2^T \prod_{t=1}^{T} \sqrt{\left(\frac{1}{2} - \gamma\right)\left(\frac{1}{2} + \gamma\right)} \;=\; 2^T \prod_{t=1}^{T} \sqrt{\frac{1}{4} - \gamma^2} \;=\; (1 - 4\gamma^2)^{T/2} \;\leq\; e^{-2T\gamma^2} \to 0 \text{ as } T \to \infty \, . \tag{17}$$

Thus in this case, by taking T large enough, the empirical error $\mathrm{er}_S^{0\text{-}1}[h_T]$ can be made zero; applying VC-dimension based bounds (assuming finite capacity of the base class $\mathcal{F}$) or margin-based bounds then provides a high-confidence bound on the generalization error of h_T, which can then be made arbitrarily small for sufficiently large sample size m. (Of course, these bounds can also be applied when there is no known bound $\frac{1}{2} - \gamma$ on the errors er_t, in which case one can bound the generalization error of h_T in terms of the empirical error of h_T as usual.)

3.2 General Case: $f_t(x) \in \mathbb{R}$

In order to minimize $Z_t \equiv Z_t(\alpha) = \sum_{i=1}^{m} D_t(i) \exp(-\alpha y_i f_t(x_i))$ over $\alpha \in \mathbb{R}$, consider the derivative of $Z_t(\alpha)$ w.r.t. α:

$$\frac{dZ_t(\alpha)}{d\alpha} \;=\; -\sum_{i=1}^{m} D_t(i) y_i f_t(x_i) \exp(-\alpha y_i f_t(x_i)) \;=\; -Z_t(\alpha) \sum_{i=1}^{m} D_{t+1}(\alpha, i) y_i f_t(x_i) \;=\; -Z_t(\alpha) \cdot r_{t+1}(\alpha) \, . \tag{18}$$

Setting the derivative to zero yields $r_{t+1}(\alpha) = 0$, i.e. $\mathbf{E}_{i \sim D_{t+1}(\alpha)} [y_i f_t(x_i)] = 0$. Moreover, it can be shown that unless $y_i f_t(x_i) = 0$ $\forall i$, the second derivative $\frac{d^2 Z_t(\alpha)}{d\alpha^2}$ is strictly positive for all $\alpha \in \mathbb{R}$, which gives that $Z_t(\alpha)$ is a convex function of α. In particular, if $\exists i_1$ such that $y_{i_1} f_t(x_{i_1}) > 0$ (which gives $Z_t'(\alpha) \to -\infty$ as $\alpha \to -\infty$) and $\exists i_2$ such that $y_{i_2} f_t(x_{i_2}) < 0$ (which gives $Z_t'(\alpha) \to \infty$ as $\alpha \to \infty$), then there exists a unique choice of $\alpha_t \in \mathbb{R}$ that minimizes $Z_t(\alpha)$, which can be found numerically. At this choice of α_t, we have from the above that $D_{t+1} = D_{t+1}(\alpha_t)$ and $r_{t+1} = r_{t+1}(\alpha_t)$ satisfy

$$r_{t+1} = \mathbf{E}_{i \sim D_{t+1}} [y_i f_t(x_i)] = 0 \, . \tag{19}$$

4 Extensions and Applications

We have described above the basic AdaBoost algorithm for binary classification. Extensions exist for a variety of other learning problems, including regression, multiclass classification, and ranking; see for example [4, 7, 1]. Boosting algorithms have been used in a variety of applications, ranging from face detection to gene ranking.

5 Next Lecture

In the next lecture, we will return to the question of statistical consistency of learning algorithms. In particular, we will consider statistical consistency of the k-nearest neighbors classification algorithm.

References

[1] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[2] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[3] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning (ICML)*, 1996.

[4] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[5] David Haussler, Michael Kearns, Nick Littlestone, and Manfred K. Warmuth. Equivalence of models for polynomial learnability. *Information and Computation*, 95(2):129–161, 1991.

[6] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[7] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.