AdaBoost

Meir and Rätsch wrote a very nice (though a bit old) introduction to boosting and leveraging [10]. Some of our materials are based on that survey.

1 Bagging and Boosting, High-level Idea

The main idea of "boosting" is to combine a set of weak hypotheses into a stronger hypothesis. The question of whether weak hypotheses can be combined to form a strong hypothesis has its root in PAC-learning [16]. Kearns and Valiant [8] proved that a weak PAC-learner can be "boosted" to form strong leaner (with arbitrary small error). Schapire [14] was the first to show that weak PAC-learning and strong PAC-learning are essentially equivalent: the boosting algorithm can be done in polytime. Finally, Freund and Schapire [6] designed AdaBoost (**Ada**ptive **Boost**) as we know it today, which is considered the first step toward practical boosting.

To be more precise, recall the PAC setting where we have an unknown distribution $\mathcal{D}$ on Ω, a target concept $c \in \mathcal{H}$. A *strong PAC-learner* is an algorithm which, given any $\epsilon, \delta > 0$, takes $m = \text{poly}(1/\epsilon, 1/\delta)$ samples $S = \{(\mathbf{x}_i, c(\mathbf{x}_i))\}_{i=1}^m$ where $\mathbf{x}_i \sim \mathcal{D}$ and outputs a hypothesis h satisfying

$$\Pr_S[\text{err}(h) \leq \epsilon] \geq 1 - \delta.$$

The algorithm runs in time $\text{poly}(1/\delta, 1/\epsilon, n = |\mathbf{x}_i|)$. A *weak PAC-learner* is a learner as above which works only for a specific value of $\epsilon = \epsilon_0$.

Exercise 1. Suppose we have a learner which works efficiently for a specific pair $0 < \epsilon_0, \delta_0 < 1/2$, namely it always outputs a hypothesis h in time $\text{poly}(1/\epsilon_0, 1/\delta_0, n)$ such that

$$\Pr_S[\text{err}(h) \leq \epsilon_0] \geq 1 - \delta_0.$$

Show how to use this learner to design a new learner which works efficiently for any small δ, i.e. the new learner always outputs a hypothesis h where

$$\Pr_S[\text{err}(h) \leq \epsilon_0] \geq 1 - \delta$$

and it runs in time $\text{poly}(1/\epsilon_0, 1/\delta, n)$.

The above exercise shows that boosting the confidence is easy. Boosting the error is another story though. Since the PAC-setting is quite limited, we will work on the inconsistent hypothesis model (IHM) directly.

Under the IHM model, there is an unknown distribution $\mathcal{D}$ on $\Omega \times \{-1, 1\}$. A weak learner is an algorithm $\mathbf{A}$ for which there exists an $\epsilon_0 < 1/2$ such that given any $0 < \delta < 1/2$, $\mathbf{A}$ takes m samples S from $\mathcal{D}$ and produces a hypothesis $h \in \mathcal{H}$ which satisfies the following

$$\Pr_S[\text{err}(h) \leq \epsilon_0] \geq 1 - \delta.$$

Recall that

$$\text{err}(h) = \text{err}_{\mathcal{D}}(h) = \operatorname*{Prob}_{(\mathbf{x},y)\sim\mathcal{D}}[h(\mathbf{x}) \neq y].$$

(Note that we are not concerned with the poly-size of m just yet, because our results hold for any m.)

AdaBoost is an *ensemble* method which outputs a "composite hypothesis" which is a linear combination of the weak hypotheses. The hope (and reality confirms) is that weak hypotheses with errors close to $1/2$ (but not equal to $1/2$) are easy to find, and boosting gives us a meta-learning algorithm to produce good learners. The main question is how to combine the weak classifiers.

Possibly the simplest type of "ensemble learning" is called *Bagging* (**B**oostrap **Agg**regat**ing**) by the Berkeley statistician Breiman [2]. In bagging we take m training samples, then take $m' \leq m$ uniform samples with replacement from the training samples and feed m' samples to a learner to fetch a hypothesis h_1. Do this independently T times to get hypotheses $h_1, \ldots, h_T$. Finally, output a composite hypothesis which always takes the majority vote of the hypotheses $h_1, \ldots, h_T$. Bagging often can reduce the variance of the output hypothesis. (Roughly speaking, the error of a hypothesis on an example $\mathbf{x}$ comes from 3 sources: noise, bias, and variance. Noise is unavoidable. This is intrinsic uncertainty over $\mathbf{x}$'s true label. Bias measures how close the prediction is to the real label. Variance measures how much a prediction changes if the samples change.) Bagging does not improve the bias.

In boosting, instead of uniform sampling from the training samples we will focus more on the samples which were wrongly classified in the previous rounds. The voting does not treat all (weak) hypotheses equally. We take a weighted vote.

2 AdaBoost and its empirical error bound

In the following algorithm and analysis it is natural to include the notion of "soft classifiers." As we have discussed so far, the hypotheses are functions from $\Omega \to \{-1, 1\}$. These types of classifiers are called *hard classifiers*. Soft classifiers are functions from $\Omega \to \mathbb{R}$. If h is a soft classifier then the label of $\mathbf{x} \in \Omega$ is 1 if $h(\mathbf{x}) > 0$ and -1 if $h(\mathbf{x}) \leq 0$. In other words, the label of $\mathbf{x}$ under h is $\text{sign}(h(\mathbf{x}))$. Hard classifiers are a special case of soft classifiers, certainly.

The generalization error of a soft classifier h cannot be defined to be $\text{Prob}[h(\mathbf{x}) \neq y]$ any more. To formalize the notion of a "loss" for a soft classifier, we define a *01-loss function* $\lambda : \mathbb{R}^2 \to \{0, 1\}$ by

$$\lambda(y', y) = \mathbf{1}_{\{y'y \leq 0\}}.$$

Hence, $\lambda(h(\mathbf{x}), y)$ is 1 if $h(\mathbf{x})$ has a different sign from y or $h(\mathbf{x}) = 0$, in which case we count the label of h given to $\mathbf{x}$ as an error. Then, the performance of a soft classifier h is measured by the *risk* or the *generalization error*

$$L^{01}_{\mathcal{D}}(h) = \int \lambda(h(\mathbf{x}), y) d\mathcal{D}(\mathbf{x}, y) = \mathop{\mathbb{E}}_{\mathcal{D}}[\lambda(h(\mathbf{x}), y)].$$

Since $\mathcal{D}$ is often implicit, we write L^{01} instead of $L^{01}_{\mathcal{D}}$, except for cases when we want to emphasize the underlying distribution. If h was a hard classifier, then this is precisely $\text{err}(h)$ has we have been working on. In a couple of lectures we will look at other types of loss functions. Naturally, the *empirical risk* or *empirical loss* of h is defined by

$$\hat{L}^{01}(h) = \frac{1}{m}\sum_{i=1}^{m} \lambda(h(\mathbf{x}_i), y_i) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}_{\{h(\mathbf{x}_i)y_i \leq 0\}}.$$

Algorithm 1 AdaBoost

Require: A weak learner (also called *base learner*)

1: Take m training samples $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ from the unknown distribution $\mathcal{D}$ on $\Omega \times \{-1, 1\}$.
2: $\mathcal{D}_1(i) = 1/m$, for all $i \in [m]$
3: **for** $t = 1$ to T **do**
4: Train the base learner using $\mathcal{D}_t$ on S
5: Get a base classifier h_t from the weak learner
6: Select "weight" $\alpha_t \in \mathbb{R}$ // more below
7: Update for each $i \in [m]$: $\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i)e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$
8: // Z_t is a normalizing factor so that $\mathcal{D}_{t+1}$ is a distribution on S
9: **end for**
10: $f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$
11: **Output** $H(\mathbf{x}) = \text{sign}\,(f(\mathbf{x}))$

Algorithm 2 shows the skeleton of the Adaptive Boost algorithm. We first take m samples S from the unknown distribution $\mathcal{D}$. Then, we use S to train our weak learner using different distributions on S: $\mathcal{D}_1, \ldots, \mathcal{D}_m$. The first distribution $\mathcal{D}_1$ is simply the uniform distribution on S. In the second distribution ($t = 2$), we "re-weight" the probability masses of points in S so that points which were wrongly classified by h_1 get (exponentially) higher weight. In particular, if $y_i \neq h_1(\mathbf{x}_i)$ which means $y_i h(\mathbf{x}_i) \leq 0$ then $\mathcal{D}_2(i) \propto \mathcal{D}_1(i)e^{\alpha_1}$ and if $y_i = h(\mathbf{x}_i)$ then $\mathcal{D}_2(i) \propto \mathcal{D}_1(i)e^{-\alpha_1}$. The weight α_1 is a parameter which we will choose later analytically. Later rounds are done similarly. Finally, we obtain a weighted average of all the weak classifiers and output the weighted "vote." Figure 1, taken from the survey by Meir and Rätsch, illustrates the main idea.

How do we choose the α_t? That is the main question. We use the Empirical Risk Minimization strategy (a form of induction) which aims to minimize the empirical error of the ensemble hypothesis H. Thus, we need to bound $\widehat{\text{err}}(H)$, or equivalently in this case $\hat{L}^{01}(f)$. Now, noting that $\mathbf{1}_{\{y_i f(\mathbf{x}_i) \leq 0\}} \leq e^{-y_i f(\mathbf{x}_i)}$, we have

$$\begin{aligned}
\widehat{\text{err}}(H) = \hat{L}^{01}(f) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}_{\{y_i f(\mathbf{x}_i) \leq 0\}} &\leq \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(\mathbf{x}_i)} \\
&= \frac{1}{m}\sum_{i=1}^{m} e^{-y_i \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i)} \\
&= \sum_{i=1}^{m} \mathcal{D}_1(i) \prod_{t=1}^{T} e^{-y_i \alpha_t h_t(\mathbf{x}_i)} \\
&= \sum_{i=1}^{m} \mathcal{D}_1(i) e^{-y_i \alpha_1 h_1(\mathbf{x}_i)} \prod_{t=2}^{T} e^{-y_i \alpha_t h_t(\mathbf{x}_i)} \\
&= \sum_{i=1}^{m} Z_1 \mathcal{D}_2(i) \prod_{t=2}^{T} e^{-y_i \alpha_t h_t(\mathbf{x}_i)} \\
&= \prod_{t=1}^{T} Z_t.
\end{aligned}$$

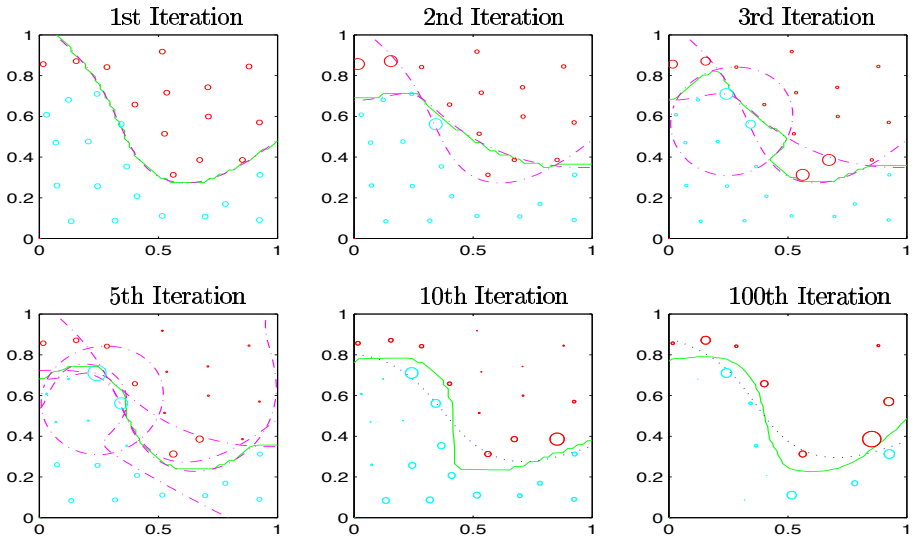


Fig. 1. Illustration of AdaBoost on a 2D toy data set: The color indicates the label and the diameter is proportional to the weight of the examples in the first, second, third, 5th, 10th and 100th iteration. The dashed lines show the decision boundaries of the single classifiers (up to the 5th iteration). The solid line shows the decision line of the combined classifier. In the last two plots the decision line of Bagging is plotted for a comparison. (Figure taken from [153].)

Figure 1: Illustration of AdaBoost, Taken from Meir and Rätsch's survey

So we pick the α_t to minimize the product $\prod_{t=1}^{T} Z_t$. Let us do the analysis for the case where all h_t are hard classifiers, i.e. $h_t(\mathbf{x}) \in \{-1, 1\}, \forall \mathbf{x} \in \Omega$. Define

$$\epsilon_t = \mathrm{err}_{\mathcal{D}_t}(h_t) = \sum_{i=1}^{m} \mathbf{1}_{\{y_i \neq h_t(\mathbf{x}_i)\}} \mathcal{D}_t(i) = \sum_{y_i \neq h_t(\mathbf{x}_i)} \mathcal{D}_t(i). \tag{1}$$

Thus, by definition

$$\begin{aligned} Z_t &= \sum_{i=1}^{m} \mathcal{D}_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} \\ &= \sum_{y_i \neq h_t(\mathbf{x}_i)} \mathcal{D}_t(i) e^{\alpha_t} + \sum_{y_i = h_t(\mathbf{x}_i)} \mathcal{D}_t(i) e^{-\alpha_t} \\ &= \epsilon_t e^{\alpha_t} + (1 - \epsilon_t) e^{-\alpha_t} \\ &\geq 2\sqrt{\epsilon_t (1 - \epsilon_t)}, \end{aligned}$$

where equality holds if only if we chose

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}.$$

This is the value that we "plug in" to the algorithm. If all base classifiers h_t are weak-classifiers in the sense that there is some constant $\gamma > 0$ such that $\epsilon_t \leq 1/2 - \gamma$, then the empirical risk of the output is bounded by

$$\hat{L}^{01}(H) \leq \prod_t 2\sqrt{\epsilon_t (1 - \epsilon_t)} = \prod_t 2\sqrt{1/4 - (\epsilon_t - 1/2)^2} \leq \prod_t \sqrt{1 - 4\gamma^2} \leq \prod_t e^{-2\gamma^2} = e^{-2T\gamma^2}.$$

This is beautiful, because the empirical risk goes to zero as $T \to \infty$. The problem is, as always, driving down the empirical risk might "risk" driving up the generalization error due to overfitting.

Exercise 2 (When base classifiers are soft)**.** The analysis above doesn't quite work if the $h_t : \Omega \to [-1, 1]$ are soft classifiers. We can make it work by following the following reasoning. Define

$$r_t = \sum_{i=1}^{m} \mathcal{D}_t(i) y_i h_t(\mathbf{x}_i).$$

Then r_t basically measures "how wrong" h_t is.

(a) Show that, for any real number w, and any $\alpha \geq 0$

$$e^{-\alpha w} \leq \frac{1+w}{2} e^{-\alpha w} + \frac{1-w}{2} e^{\alpha w}.$$

(b) Now define $w_i = y_i h_t(\mathbf{x}_i)$, derive that

$$Z_t \leq \sum_{i=1}^{m} \mathcal{D}_t(i) \left(\frac{1+w_i}{2} e^{-\alpha_t w_i} + \frac{1-w_i}{2} e^{\alpha_t w_i} \right).$$

(c) Show that the right hand side above is minimized at $\alpha_t = \frac{1}{2} \ln \frac{1+r_t}{1-r_t}$.

(d) How does r_t relate to $L^{01}_{\mathcal{D}_t}(h_t)$?

(e) What can you say about the empirical risk of H in this case?

3 Generalization error bound

So we have a bound on the empirical error of the ensemble hypothesis H. What about its generalization error? First of all, let $\mathcal{H}$ denote the function class of the weak classifiers h_t. Define

$$\Theta_T(\mathcal{H}) = \left\{ \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t \right) \mid \alpha_t \geq 0, h_t \in \mathcal{H} \right\}.$$

Then clearly $H \in \Theta_T(\mathcal{H})$. From the Rademacher complexity lectures we know that with probability at least $1 - \delta$ we have

$$L^{01}(H) \leq \hat{L}^{01}(H) + \widehat{\mathcal{R}}_S(\Theta_T(\mathcal{H})) + 3\sqrt{\frac{\ln(2/\delta)}{2m}},$$

and

$$L^{01}(H) \leq \hat{L}^{01}(H) + \mathcal{R}_m(\Theta_T(\mathcal{H})) + \sqrt{\frac{\ln(1/\delta)}{2m}}.$$

In particular, applying Massart's lemma or applying VC Theorem directly we have with probability at least $1 - \delta$

$$L^{01}(H) \leq \hat{L}^{01}(H) + O\left(\sqrt{\frac{\text{VCD}(\Theta_T(\mathcal{H})) \ln m + \ln(1/\delta)}{m}} \right).$$

Thus, in order to estimate the generalization error bound we will need to either bound $\text{VCD}(\Theta_T(\mathcal{H}))$ or bound the Rademacher complexities of $\Theta_T(\mathcal{H})$ directly.

Let us first bound $\text{VCD}(\Theta_T(\mathcal{H}))$. Freund and Schapire in their original AdaBoost paper bounded this quantity using a simple observation from Baum and Haussler [1]. We will work on $\Omega = \mathbb{R}^n$ as this is probably the most common usage. A *feedforward net* is a directed acyclic graph with n inputs and one output. Each input corresponds to a coordinate from $\mathbb{R}^n$. Each non-input node v is associated with a function f_v which belongs to a function class $\mathcal{F}_v$. These nodes are called *computation nodes*. The in-degree of a computation node v is denoted by $\text{INDEG}(v)$. The function f_v is from $\mathbb{R}^{\text{INDEG}(v)} \to \{-1, 1\}$. The net itself is associated with a function $f : \mathbb{R}^n \to \{-1, 1\}$ defined in the obvious way. Let $\mathcal{F}$ denote the class of functions f formed by this *feedforward architecture*.

It should be obvious that $\Theta_T(\mathcal{H})$ is a feedforward architecture with $T + 1$ computation nodes. The first T nodes correspond to functions $h_1, \ldots, h_T$. The last node is the sign function, which is basically a linear threshold function:

$$\text{sign}\left(\sum_t \alpha_t y_t\right) = \begin{cases} 1 & \sum_t \alpha_t y_t > 0 \\ 0 & \sum_t \alpha_t y_t \leq 0. \end{cases}$$

We also know that the class of linear threshold functions on $\mathbb{R}^T$ have VC-dimension $T + 1$.

Going back to the general setting of Baum and Hassler. We will bound $\text{VCD}(\mathcal{F})$ in terms of $\text{VCD}(\mathcal{F}_v)$ for computation nodes v. Let's order all computation nodes $v_1, \ldots, v_N$ so that the inputs of f_{v_i} are only dependent on the actual inputs or the outputs of f_{v_j} for $j < i$. This is possible since the architecture is acyclic. Let's use $\mathcal{F}_i$ instead of $\mathcal{F}_{v_i}$ to denote the function class corresponding to computation node v_i. Consider a set S of m sample points. $\mathcal{F}_1$ can label S in $\Pi_{\mathcal{F}_1}(m)$ different ways. Then, for each of these labelings $\mathcal{F}_2$ can label S in $\Pi_{\mathcal{F}_2}(m)$ different ways. The process goes on. Hence, overall S can be labeled by $\mathcal{F}$ in $\Pi_{\mathcal{F}}(m) \leq \prod_{i=1}^N \Pi_{\mathcal{F}_i}(m)$ different ways.

Now, let $d_i = \text{VCD}(\mathcal{F}_i)$ and $d = \sum_{i=1}^N d_i$. We assume that all d_i are finite. Suppose $m \geq d \geq d_i$. Then,

$$\begin{aligned} \prod_{i=1}^N \Pi_{F_i}(m) &\leq \prod_{i=1}^N \left(\frac{me}{d_i}\right)^{d_i} \\ &= (me)^d \prod_{i=1}^N \left(\frac{1}{d_i}\right)^{d_i} \\ &\leq (me)^d (N/d)^d. \end{aligned}$$

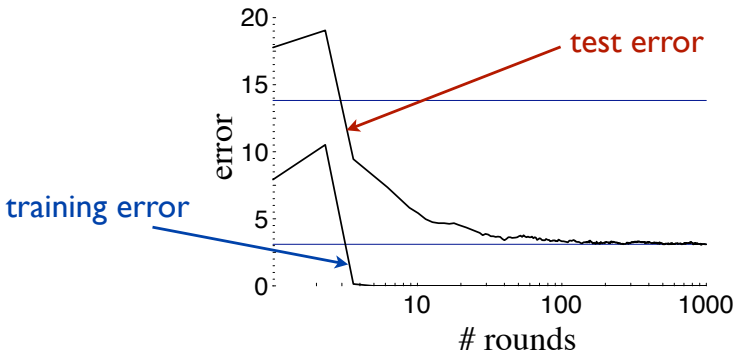
To see the last inequality, we prove its equivalent form

$$\sum_{i=1}^N \frac{d_i}{d} \log(1/d_i) \leq \log(N/d).$$

If we set $p_i = d_i/d$ and $X = 1/d_i$ with probability p_i, then the left hand side is $\mathbb{E}[\log(X)]$ over this distribution. Since $\log$ is concave, by Jensen's inequality

$$\mathbb{E}[\log(X)] \geq \log \mathbb{E}[X] = \log\left(\sum_{i=1}^n (1/d_i)(d_i/d)\right) = \log(N/d).$$

We just proved the following lemma by Baum and Haussler.



C4.5 decision trees (Schapire et al., 1998).

Figure 2: Sometime AdaBoost does not overfit

Lemma 3.1 (Baum and Haussler)**.** *Let $\mathcal{F}$ be a feedforward architecture with $N \geq 2$ computation nodes. Let $\mathcal{F}_i$ be the function class associated with the ith computation node. Let $d_i = \text{VCD}(\mathcal{F}_i)$ and $d = \sum_{i=1}^{N} d_i$. Then,*

$$\Pi_{\mathcal{F}}(m) \leq \prod_{i=1}^{N} \Pi_{\mathcal{F}_i}(m) \leq (Nme/d)^d$$

for $m \geq d$.

Exercise 3. Let $\mathcal{H}$ be the hypothesis class of the (hard) base classifiers, and T be the number of iterations in the AdaBoost algorithm. Then,

$$\text{VCD}(\Theta_T(\mathcal{H})) \leq 2(d+1)(T+1)\log_2(e(T+1)).$$

From the exercise (which follows from Baum and Haussler's lemma), there is a potential issue: when T increases we are able to drive down the empirical error but the VCD of the ensemble function class increases which potentially leads to overfitting. Some experiments did not confirm this worry (Figure 2). Is there another explanation?

4 Boosting the Margin

There is a way to partly explain the fact that AdaBoost does not seem to overfit in some situation [15]. When bounding the generalization error of H we use the empirical error of H. However, as H is only the sign of f, we lost some information about f. In particular, the magnitude of the value of $f(\mathbf{x})$ should give us some indication about the degree of confidence that f has on its label. While H is a hard classifier, f is a soft classifier. To assess the confidence f has, let us define a couple of other loss functions which try to capture confidence levels.

The 01-*margin loss function* is defined to be

$$\lambda_\rho^{01}(y, y') = \mathbf{1}_{yy' \leq \rho}$$

where $1 \geq \rho \geq 0$ is called the *margin*. If $\rho = 0$ then the 01-margin loss function is just the normal 01-loss function. For $\rho > 0$, $\lambda_\rho^{01}(f(\mathbf{x}), y)$ is 1 (i.e. counted as a loss) if $f(\mathbf{x})y \leq \rho$. Now, if $f(\mathbf{x})$ and y have

different signs then it is a loss as before. For our problem $y \in \{-1, 1\}$. Hence, even when $f(\mathbf{x})y > 0$ we would still have a loss if $f(\mathbf{x})$ has the same sign as y but its magnitude (i.e. margin/confidence) is smaller than ρ.

The *margin loss function* is defined by

$$\lambda_\rho(y, y') = \begin{cases} 1 & yy' \leq 0 \\ 1 - yy'/\rho & 0 < yy' \leq \rho \\ 0 & yy' > \rho. \end{cases}$$

So in this case we "ease" the loss linearly from 1 down to 0 as yy' approaches ρ. When $\rho = 1$ this is called the *Hinge loss* function, an important loss function used in SVM. There are many types of loss functions such as square loss, log loss, etc.

The *margin loss* of function f is defined to be

$$L_\rho(f) := \mathbb{E}[\lambda(f(\mathbf{x}), y)]$$

and the empirical margin loss of f is

$$\hat{L}_\rho(f) := \frac{1}{m} \sum_{i=1}^{m} \lambda_\rho(f(\mathbf{x}_i), y_i).$$

It should be noted that

$$\lambda \leq \lambda_\rho \leq \lambda_\rho^{01}.$$

Thus

$$L^{01} \leq L_\rho \leq L_\rho^{01},$$

and

$$\hat{L}^{01} \leq \hat{L}_\rho \leq \hat{L}_\rho^{01}.$$

The margin loss is an upper bound for the real loss, both of which are bounded by the 01-margin loss.

What do we gain with these new notions of loss? We will prove the following theorem.

Theorem 4.1 (General margin bound, Koltchinskii-Panchenko)**.** *Let $\mathcal{F}$ be a class of real valued functions. Fix a margin $1 \geq \rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ we have*

$$\begin{aligned} L^{01}(f) &\leq \hat{L}_\rho(f) + \frac{2}{\rho}\mathcal{R}_m(\mathcal{F}) + \sqrt{\frac{\log(1/\delta)}{2m}} \\ L^{01}(f) &\leq \hat{L}_\rho(f) + \frac{2}{\rho}\widehat{\mathcal{R}}_S(\mathcal{F}) + 3\sqrt{\frac{\log(2/\delta)}{2m}}. \end{aligned}$$

In the context of AdaBoost, we will be able to show that as long as $\rho < \gamma$ then $\hat{L}_\rho(f) \to 0$ as $T \to \infty$. Furthermore, we will show that $D = \text{VCD}(\mathcal{F})$ is independent of T. From $\mathcal{R}_m(\mathcal{F}) = O\sqrt{(D \ln m)/m}$ we can now infer that when $\rho \gg 1/\sqrt{m}$ the other two terms goes to 0 as well. To summarize, in order to show that in some cases AdaBoost does not overfit we have to do the following:

- Prove Theorem 4.1

- Prove that $\hat{L}_\rho(f)$ tends to 0 for some range of ρ as T tends to ∞
- Prove that $\text{VCD}(\mathcal{F})$ is independent of T
- Conclude that if we are able to make $\rho \gg 1/\sqrt{m}$ then all terms in the upperbound for $L^{01}(f)$ tends to 0.

These steps are implemented in the following four subsections.

4.1 Proof of Theorem 4.1

To prove this theorem, we need two lemmas. The first lemma we already proved in the Rademacher complexity lecture.

Lemma 4.2 (Koltchinskii-Panchenko, 2002 [9]). *Let $\mathcal{G}$ be a class of functions from $\mathcal{Z}$ to $[0,1]$, and $\mathcal{D}$ be an arbitrary distribution on $\mathcal{Z}$. For notational convenience, we write*

$$\begin{aligned} \mathbb{E}[g] &= \mathop{\mathbb{E}}_{z\in\mathcal{D}}[g(z)] \\ \widehat{\mathbb{E}}_S[g] &= \frac{1}{m}\sum_{i=1}^{m} g(z_i), \text{ where } S = \{z_1,\ldots,z_m\}. \end{aligned}$$

Then, for any $\delta > 0$,

$$\mathop{Prob}_{S\sim\mathcal{D}^m}\left[\sup_{g\in\mathcal{G}}\{\mathbb{E}[g] - \widehat{\mathbb{E}}_S[g]\} \le 2\mathcal{R}_m(\mathcal{G}) + \sqrt{\frac{\ln(1/\delta)}{2m}}\right] \ge 1-\delta, \tag{2}$$

and

$$\mathop{Prob}_{S\sim\mathcal{D}^m}\left[\sup_{g\in\mathcal{G}}\{\mathbb{E}[g] - \widehat{\mathbb{E}}_S[g]\} \le 2\widehat{\mathcal{R}}_S(\mathcal{G}) + 3\sqrt{\frac{\ln(2/\delta)}{2m}}\right] \ge 1-\delta. \tag{3}$$

The second lemma is called the *Talagrand Contraction lemma* []. A function $\varphi : \mathbb{R} \to \mathbb{R}$ is called *L-Lipschitz* if for any $x, y \in \mathbb{R}$ $|\varphi(x) - \varphi(y)| \le L \cdot |x - y|$.

Lemma 4.3 (Talagrand's Contraction Lemma). *Let $\varphi : \mathbb{R} \to \mathbb{R}$ be an L-Lipschitz function with $\varphi(0) = 0$. Let $\mathcal{F}$ be any family of real-valued functions from $\mathcal{Z}$ to $\mathbb{R}$. Then*

$$\widehat{\mathcal{R}}_S(\varphi\circ\mathcal{F}) \le L\cdot\widehat{\mathcal{R}}_S(\mathcal{F}).$$

Here, S is a set of m (sample) points from $\mathcal{Z}$.

Before proving Talagrand's contraction lemma, let us finish the proof of Theorem 4.1 using the above two lemmas.

Proof of Theorem 4.1. Let $\mathcal{F}$ be the family of real-valued functions given in Theorem 4.1. Suppose functions in $\mathcal{F}$ are from Ω to $\mathbb{R}$ for some domain Ω. Define $\mathcal{Z} = \Omega \times \{-1, 1\}$. Define a new class of functions $\tilde{\mathcal{F}}$ which consists of one function $\tilde{f} : \mathcal{Z} \to \mathbb{R}$ for each $f \in \mathcal{F}$. The function $\tilde{f}$ is defined by

$$\tilde{f}(z) = \tilde{f}(\mathbf{x}, y) = yf(\mathbf{x}).$$

(Roughly, $\tilde{f}$ measures the "margin/confidence" of f. If f was a hard classifier then $\tilde{f}$ tells us whether f was right or wrong on the point $(\mathbf{x}, y)$.)

Next, define a function $\varphi_\rho : \mathbb{R} \to [0,1]$ as follows.

$$\varphi_\rho(x) = \begin{cases} 1 & x \leq 0 \\ 1 - x/\rho & 0 < x \leq \rho \\ 0 & x > \rho \end{cases}$$

Then obviously $\lambda_\rho(y', y) = \varphi_\rho(y'y)$ and φ_ρ is a $(1/\rho)$-Lipschitz function. Now, define another function class

$$\mathcal{G} = \{\varphi_\rho \circ \tilde{f} \mid \tilde{f} \in \tilde{\mathcal{F}}.\}.$$

Thus functions in $\mathcal{G}$ measures the margin loss of functions in f. Now, we apply Koltchinskii-Panchenko's lemma on $\mathcal{G}$. Let $S = \{z_1, \ldots, z_m\}$ be m points from $\mathcal{Z}$ and $\mathcal{D}$ be an arbitrary distribution on $\mathcal{Z}$. We have, with probability at least $1 - \delta$, the following holds for all $g \in \mathcal{G}$ (we are proving both statements in parallel)

$$\begin{aligned} \mathop{\mathbb{E}}_{z \sim \mathcal{D}}[g(z)] &\leq \frac{1}{m}\sum_{i=1}^m g(z_i) + 2\mathcal{R}_m(\mathcal{G}) + \sqrt{\frac{\log(1/\delta)}{2m}} \\ \mathop{\mathbb{E}}_{z \sim \mathcal{D}}[g(z)] &\leq \frac{1}{m}\sum_{i=1}^m g(z_i) + 2\widehat{\mathcal{R}}_S(\mathcal{G}) + 3\sqrt{\frac{\log(2/\delta)}{2m}}. \end{aligned}$$

Now, to convert the above statements in to statements about $\mathcal{F}$, note the following

- We can bound the 01-loss of f by g. Let $z = (\mathbf{x}, y)$, then

  $$\lambda(f(\mathbf{x}), y) = \mathbf{1}_{f(\mathbf{x})y \leq 0} \leq \varphi_\rho(f(\mathbf{x})y) = \varphi_\rho(\tilde{f}(z)) = g(z).$$

  Hence,

  $$L^{01}(f) = \mathbb{E}[\lambda(f(\mathbf{x}), y)] \leq \mathbb{E}[g(z)].$$

- Next, it is obvious that

  $$\frac{1}{m}\sum_{i=1}^m g(z_i) = \hat{L}_\rho(f).$$

- Finally, we bound the Rademacher complexity of $\mathcal{G}$ by the Rademacher complexity of $\mathcal{F}$:

$$\begin{aligned} \widehat{\mathcal{R}}_S(\mathcal{G}) &= \widehat{\mathcal{R}}_S(\varphi_\rho \circ \tilde{\mathcal{F}}) \\ \text{(Rade. comp. invariant under a constant shift)} &= \widehat{\mathcal{R}}_S((\varphi_\rho - 1) \circ \tilde{\mathcal{F}}) \\ \text{(Talagrand's lemma)} &\leq \frac{1}{\rho}\widehat{\mathcal{R}}_S(\tilde{\mathcal{F}}) \\ &= \frac{1}{\rho}\mathop{\mathbb{E}}_{\sigma}\left[\sup \frac{1}{m}\sum_{i=1}^m \sigma_i y_i f(\mathbf{x}_i) \mid S\right] \\ &= \frac{1}{\rho}\mathop{\mathbb{E}}_{\sigma}\left[\sup \frac{1}{m}\sum_{i=1}^m \sigma_i y_i f(\mathbf{x}_i) \mid S\right] \\ (y_i \in \{-1, 1\}) &= \frac{1}{\rho}\mathop{\mathbb{E}}_{\sigma}\left[\sup \frac{1}{m}\sum_{i=1}^m \sigma_i f(\mathbf{x}_i) \mid S\right] \\ &= \widehat{\mathcal{R}}_S(\mathcal{F}) \end{aligned}$$

From that, $\mathcal{R}_m(\mathcal{G}) = \mathbb{E}[\widehat{\mathcal{R}}_S(\mathcal{G})] \leq \mathbb{E}[\widehat{\mathcal{R}}_S(\mathcal{F})] = \mathcal{R}_m(\mathcal{F})$ and the proof is completed.

□

We next prove Talagrand's contraction lemma.

Proof of Lemma 4.3. We start with the left-hand-side.

$$\begin{aligned}
\widehat{\mathcal{R}}_S(\varphi \circ \mathcal{F}) &= \mathop{\mathbb{E}}_{\sigma}\left[\sup_{f\in\mathcal{F}} \frac{1}{m}\sum_{i=1}^{m} \sigma_i(\varphi\circ f)(z_i)\right] \\
&= \mathop{\mathbb{E}}_{\sigma_1,\ldots,\sigma_{m-1}}\left[\mathop{\mathbb{E}}_{\sigma_m}\left[\sup_{f\in\mathcal{F}} \frac{1}{m}\sum_{i=1}^{m} \sigma_i(\varphi\circ f)(z_i)\right]\right] \\
&= \mathop{\mathbb{E}}_{\sigma_1,\ldots,\sigma_{m-1}}\left[\mathop{\mathbb{E}}_{\sigma_m}\left[\sup_{f\in\mathcal{F}} \frac{1}{m}u_m(f)\right]\right],
\end{aligned}$$

where to simplify notations we have defined

$$u_k(f) := \sum_{i=1}^{k} \sigma_i(\varphi\circ f)(z_k).$$

Next, we bound the inner expectation:

$$\mathop{\mathbb{E}}_{\sigma_m}\left[\sup_{f\in\mathcal{F}} \frac{1}{m}u_m(f)\right] = \frac{1}{2}\sup_{f\in\mathcal{F}}\left\{\frac{1}{m}u_{m-1}(f) + \frac{1}{m}(\varphi\circ f)(z_m)\right\} + \frac{1}{2}\sup_{f\in\mathcal{F}}\left\{\frac{1}{m}u_{m-1}(f) - \frac{1}{m}(\varphi\circ f)(z_m)\right\}$$

From the definition of sup, for every $\epsilon > 0$ there are two functions $f_1, f_2 \in \mathcal{F}$ such that

$$\begin{aligned}
\mathop{\mathbb{E}}_{\sigma_m}\left[\sup_{f\in\mathcal{F}} \frac{1}{m}u_m(f)\right] &\leq \frac{1}{2}\left[\frac{1}{m}u_{m-1}(f_1) + \frac{1}{m}(\varphi\circ f_1)(z_m)\right] + \frac{1}{2}\left[\frac{1}{m}u_{m-1}(f_2) - \frac{1}{m}(\varphi\circ f_2)(z_m)\right] + \epsilon \\
&\leq \frac{1}{2}\left[\frac{1}{m}u_{m-1}(f_1) + \frac{1}{m}u_{m-1}(f_2) + \frac{1}{m}(\varphi(f_1(z_m)) - \varphi(f_2(z_m))\right] + \epsilon \\
(\bar\sigma \in \{-1,1\}) \quad &\leq \frac{1}{2}\left[\frac{1}{m}u_{m-1}(f_1) + \frac{1}{m}u_{m-1}(f_2) + \frac{L}{m}\bar\sigma(f_1(z_m) - f_2(z_m)\right] + \epsilon \\
&= \frac{1}{2}\left[\frac{1}{m}u_{m-1}(f_1) + \frac{L}{m}\bar\sigma f_1(z_m)\right] + \frac{1}{2}\left[\frac{1}{m}u_{m-1}(f_2) - \frac{L}{m}\bar\sigma f_2(z_m)\right] + \epsilon \\
&\leq \mathop{\mathbb{E}}_{\sigma_m}\left[\sup_{f\in\mathcal{F}}\left\{\frac{1}{m}u_{m-1}(f) + \sigma_m L f(z_m)\right\}\right] + \epsilon
\end{aligned}$$

Since the inequality is true for any $\epsilon > 0$, it must hold that

$$\mathop{\mathbb{E}}_{\sigma_m}\left[\sup_{f\in\mathcal{F}} \frac{1}{m}u_m(f)\right] \leq \mathop{\mathbb{E}}_{\sigma_m}\left[\sup_{f\in\mathcal{F}}\left\{\frac{1}{m}u_{m-1}(f) + \sigma_m L f(z_m)\right\}\right].$$

Keep doing the same trick $m-1$ more times and we obtain the desired inequality. □

4.2 Empirical margin loss for AdaBoost

The next task is to show that $\hat{L}_\rho(f)$ tends to 0 for some range of ρ. We actually will bound an upper bound of $\hat{L}_\rho(f)$. Note that

$$\hat{L}_\rho(f) \le \hat{L}_\rho^{01}(f) = \text{Prob}[f(\mathbf{x})y \le \rho].$$

We already had an analysis of AdaBoost which bounds $\text{Prob}[f(\mathbf{x})y \le 0]$. This analysis will be similar, and thus we leave it as an exercise.

Exercise 4. Recall the definition of ϵ_t in (1). Prove that if $\epsilon_t \le 1/2 - \gamma_t$ for some $\gamma_t > 0$ in each round $t \in [T]$ of the AdaBoost algorithm, then

$$\hat{L}_\rho(f) \le \hat{L}_\rho^{01}(f) \le \prod_{t=1}^{T} (1-2\gamma_t)^{\frac{1-\rho}{2}} (1+2\gamma_t)^{\frac{1+\rho}{2}}.$$

From the exercise, if $\rho < \gamma_t$ for all t then the factor $(1-2\gamma_t)^{\frac{1-\rho}{2}}(1+2\gamma_t)^{\frac{1+\rho}{2}}$ is strictly less than 1. (Why?) Thus, when $T \to \infty$ the empirical 01-margin loss goes to 0. In particular, the empirical margin loss goes to 0.

4.3 Rademacher complexities of the convex hull of classifiers

Let $\mathcal{H}$ be a class of functions from Ω to $\mathbb{R}$. The *convex hull* of $\mathcal{H}$ is defined by

$$\text{CONV}_T(\mathcal{H}) := \left\{ \sum_{t=1}^{T} \mu_T h_t \mid \mu_t \ge 0, \sum_t \mu_t \le 1, h_t \in \mathcal{H} \right\}.$$

The soft classifier f output by the AdaBoost algorithm has the form $f = \sum_t \alpha_t h_t$. Since we are only interested in the sign of f, we can change f to be $\sum_t \frac{\alpha_t}{\|\alpha\|_1} h_t$. In this case the coefficients $\mu_t = \frac{\alpha_t}{\|\alpha\|_1}$ sums to 1 and thus $f \in \text{CONV}_T(\mathcal{H})$.

Theorem 4.4. *Let $\mathcal{H}$ be any class of functions from Ω to $\mathbb{R}$. Let S be any set of m points $\mathbf{x}_i$ from Ω. Then*

$$\widehat{\mathcal{R}}_S(\text{CONV}_T(\mathcal{H})) = \widehat{\mathcal{R}}_S(\mathcal{H}),$$

and thus

$$\mathcal{R}_m(\text{CONV}_T(\mathcal{H})) = \mathcal{R}_m(\mathcal{H}).$$

Proof. From definition,

$$\begin{aligned}
\widehat{\mathcal{R}}_S(\text{CONV}_T(\mathcal{H})) &= \frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{\mu\ge\mathbf{0},\|\mu\|_1\le 1, h_t\in\mathcal{H}} \sum_{i=1}^{m}\sigma_i \sum_{t=1}^{T}\mu_t h_t(\mathbf{x}_i)\right] \\
&= \frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h_t\in\mathcal{H}}\sup_{\mu\ge\mathbf{0},\|\mu\|_1\le 1} \sum_{t=1}^{T}\mu_t\left(\sum_{i=1}^{m}\sigma_i h_t(\mathbf{x}_i)\right)\right] \\
&= \frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h_t\in\mathcal{H}}\max_{t\in[T]}\left(\sum_{i=1}^{m}\sigma_i h_t(\mathbf{x}_i)\right)\right] \\
&= \frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^{m}\sigma_i h(\mathbf{x}_i)\right] \\
&= \widehat{\mathcal{R}}_S(\mathcal{H}).
\end{aligned}$$

□

4.4 Conclusions

Combining everything, let $\mathcal{H}$ be the function class of the base classifiers, from Theorem 4.1 we are able to bound (with probability at least $1-\delta$)

$$\text{err}(H) = L^{01}(f) \leq \hat{L}^{01}_{\rho}(f) + \frac{2}{\rho}\mathcal{R}_m(\mathcal{H}) + \sqrt{\frac{\log(1/\delta)}{2m}}$$

and

$$\text{err}(H) = L^{01}(f) \leq \hat{L}^{01}_{\rho}(f) + \frac{2}{\rho}\widehat{\mathcal{R}}_S(\mathcal{H}) + 3\sqrt{\frac{\log(2/\delta)}{2m}}.$$

The second bound is data-dependent. Hence, we might be able to use it for model selection (beyond the scope of this course). But in any case, since the Rademacher complexities are bounded by about $O(\sqrt{d/m})$, the last two terms of the upper bound will tend to 0 as $m \to \infty$.

The first term is exactly $\text{Prob}[f(\mathbf{x})y \leq \rho]$ and it goes to 0 if $\rho < \gamma$. Hence, overall we will need $\gamma \gg 1/\sqrt{m}$ which depends on how good the base classifiers are.

The expression $\text{Prob}[f(\mathbf{x})y \leq \rho]$ suggests that AdaBoost should try to maximize the margin (the confidence) of its composite hypothesis. AdaBoost does not explicitly attempt to do so, and indeed it was shown in 2004 that AdaBoost does always converge to a composite hypothesis with maximum possible margin [12]. However, it was also shown that under certain conditions AdaBoost can achieve half of the maximum possible margin [11]. There are many other boosting algorithms which explicitly aim to maximize the margin: AdaBoost* [11], arg-gv [3], Coordinate Ascent Boosting and Approximate Coordinate Ascent Boosting [13], linear programming-based boosting algorithms such as LP-AdaBoost [7] and LPBoost [5]. However, in practice AdaBoost still performs very well and in some experiments yield better margins than the other algorithms [4, 7].

Support Vector Machines do try to explicitly maximize the margin. SVM is our next subject.

References

[1] E. B. BAUM AND D. HAUSSLER, *What size net gives valid generalization?*, in NIPS, 1988, pp. 81–90.

[2] L. BREIMAN, *Bagging predictors*, Machine Learning, 24 (1996), pp. 123–140.

[3] L. BREIMAN, *Arcing classifiers*, Ann. Statist., 26 (1998), pp. 801–849. With discussion and a rejoinder by the author.

[4] L. BREIMAN, *Prediction games and arcing algorithms*, Neural Computation, 11 (1999), pp. 1493–1517.

[5] A. DEMIRIZ, K. P. BENNETT, AND J. SHAWE-TAYLOR, *Linear programming boosting via column generation*, Machine Learning, 46 (2002), pp. 225–254.

[6] Y. FREUND AND R. E. SCHAPIRE, *A decision-theoretic generalization of on-line learning and an application to boosting*, in Proceedings of the Second European Conference on Computational Learning Theory, London, UK, 1995, Springer-Verlag, pp. 23–37.

[7] A. J. GROVE AND D. SCHUURMANS, *Boosting in the limit: Maximizing the margin of learned ensembles*, in AAAI/IAAI, 1998, pp. 692–699.

[8] M. KEARNS AND L. VALIANT, *Cryptographic limitations on learning boolean formulae and finite automata*, J. ACM, 41 (1994), pp. 67–95.

[9] V. KOLTCHINSKII AND D. PANCHENKO, *Empirical margin distributions and bounding the generalization error of combined classifiers*, Ann. Statist., 30 (2002), pp. 1–50.

[10] R. MEIR AND G. RÄTSCH, *An introduction to boosting and leveraging*, in Machine Learning Summer School, 2002, pp. 118–183.

[11] G. RÄTSCH AND M. K. WARMUTH, *Efficient margin maximizing with boosting*, J. Mach. Learn. Res., 6 (2005), pp. 2131–2152.

[12] C. RUDIN, I. DAUBECHIES, AND R. E. SCHAPIRE, *The dynamics of adaboost: Cyclic behavior and convergence of margins*, Journal of Machine Learning Research, 5 (2004), pp. 1557–1595.

[13] C. RUDIN, R. E. SCHAPIRE, AND I. DAUBECHIES, *Boosting based on a smooth margin*, in Learning theory, vol. 3120 of Lecture Notes in Comput. Sci., Springer, Berlin, 2004, pp. 502–517.

[14] R. E. SCHAPIRE, *The strength of weak learnability*, Mach. Learn., 5 (1990), pp. 197–227.

[15] R. E. SCHAPIRE, Y. FREUND, P. BARLETT, AND W. S. LEE, *Boosting the margin: A new explanation for the effectiveness of voting methods*, in ICML, D. H. Fisher, ed., Morgan Kaufmann, 1997, pp. 322–330.

[16] L. G. VALIANT, *A theory of the learnable*, Commun. ACM, 27 (1984), pp. 1134–1142.