CS 412 — Introduction to Machine Learning (UIC) March 18, 2025

Lecture 17

Instructor: Aadirupa Saha *Scribe(s): Gabriel Zhang*

Overview

In the last lecture, we covered the following main topics:

1. Winnow's Algorithm
2. Mistake Bounds of Winnow's Algorithm

This lecture focuses on:

1. Boosting Algorithms
2. Adaptive Boosting (AdaBoost)
3. Error Bounds of AdaBoost

1 Boosting Algorithms

Boosting is the procedure of taking many "weak" (barely-better-than-random) classifiers and combining them into a single "strong" classifier whose accuracy is substantially higher.
For dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, let $p \in \Delta_N$, where $\Delta_N = \{q \in [0,1]^n \mid \sum_{i=1}^N q_i; q_i \geq 0\}$, denote the distribution over the N examples $\{1, 2, \ldots, N\}$.

1.1 Weak Learning Oracle (WLO)

Definition 1 (Weak Learning Oracle)**.** *Let $\mathcal{X}$ be an instance space, and let $\mathcal{H}$ be a hypothesis class consisting of binary classifiers $h : \mathcal{X} \to \{-1, +1\}$. Let $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ be a set of labeled examples where $x_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$. A* Weak Learning Oracle (WLO) *is an algorithm $\mathcal{A}$ such that, for any distribution p over the training examples, it returns a hypothesis $h \in \mathcal{H}$ satisfying:*

$$P(i)_{i\sim p}[h(x_i) \neq y_i] \leq \frac{1}{2} - \gamma,$$

for some fixed constant $\gamma \in (0, \frac{1}{2}]$, known as the edge *or* advantage *of the weak learner. The parameter γ is independent of the distribution p and the sample size N, and represents the minimum advantage over random guessing that the oracle guarantees.*

Remark 1. *Notice that*

$$E_{i\sim p}\left[\mathbb{1}\left(h\left(x_i\right) \neq y_i\right)\right] \leq \frac{1}{2} - \gamma$$
$$\iff E_{i\sim p}\left[\mathbb{1}\{h(x_i) = y_i\}\right] \ = \ 1 - E_{i\sim p}[\mathbb{1}\{h(x_i) \neq y_i\}] \ \geq \ \frac{1}{2} + \gamma.$$

1.2 Formal Description of Boosting

Algorithm 1.1: General Boosting Algorithm

```
Input: Training set $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N), \}$, where $y_i \in \{-1, +1\}$ is the ground-truth label for instance $x_i \in \mathcal{X}$.
for $t = 1$ to $T$:
    Construct distribution $p$ on $\{1, 2, \ldots, N\}$
    Find WLO $h_t : X \to \{-1, +1\}$ with small error $\varepsilon_t$ on $p$:
        $\varepsilon_t = P(i)_{i \sim p} \left[ h_t(x_i) \neq y_i \right],$
Output: Final classifier $H_{\text{final}}$
```

2 Adaptive Boosting Algorithm (AdaBoost)

2.1 Minimizing Exponential Loss

A key insight into boosting is that it can be interpreted as the sequential minimization of an *exponential loss* function. Compared to naive 0-1 loss, the exponential loss makes it more tractable to optimize, while still ensuring a tight relationship to the classification error. Consider the exponential loss

$$E(F_m) = \sum_{i=1}^{N} \exp\big(-y_i F_m(x_i)\big), \tag{1}$$

where

$$F_m(x) = \sum_{\ell=1}^{m} \alpha_\ell h_\ell(x). \tag{2}$$

The goal is to find parameters $\{\alpha_\ell\}$ and weak classifiers $\{h_\ell\}$ that minimize $E(F_m)$.
Rather than minimizing this loss over all $\ell = 1, \ldots, m$ at once, boosting algorithms proceed *sequentially*. Suppose we have already chosen $\alpha_1, \ldots, \alpha_{m-1}$ and $h_1, \ldots, h_{m-1}$. Define

$$F_{m-1}(x) = \sum_{\ell=1}^{m-1} \alpha_\ell h_\ell(x). \tag{3}$$

We now wish to find the new weak classifier h_m and its coefficient α_m. Notice that

$$E(F_m) = \sum_{i=1}^{N} \exp\big(-y_i F_{m-1}(x_i)\big) \exp\big(-\alpha_m y_i h_m(x_i)\big). \tag{4}$$

For convenience, define the *weights*

$$w_i^{(m)} = \exp\big(-y_i F_{m-1}(x_i)\big). \tag{5}$$

These weights remain fixed while we choose α_m and h_m. Then

$$E(F_m) = \sum_{i=1}^{N} w_i^{(m)} \exp\bigl(-\alpha_m\, y_i\, h_m(x_i)\bigr). \tag{6}$$

Since $h_m(x_i)$ takes values in $\{-1, +1\}$, we may separate the training set into:

$$\pi_m = \bigl\{i \mid h_m(x_i) = y_i\bigr\} \quad \text{and} \quad M_m = \bigl\{i \mid h_m(x_i) \neq y_i\bigr\}, \tag{7}$$

that is, the indices of examples correctly or incorrectly classified by h_m. Hence,

$$\exp\bigl(-\alpha_m\, y_i\, h_m(x_i)\bigr) = \begin{cases} \exp(-\alpha_m), & \text{if } i \in \pi_m, \\ \exp(\alpha_m), & \text{if } i \in M_m. \end{cases} \tag{8}$$

Therefore the loss decomposes as

$$E(F_m) = \exp(-\alpha_m) \sum_{i \in \pi_m} w_i^{(m)} + \exp(\alpha_m) \sum_{i \in M_m} w_i^{(m)}. \tag{9}$$

Minimizing this expression with respect to h_m and α_m leads to:

1. Choosing h_m so that it partitions the examples into π_m and M_m in a way that best reduces the weighted error,
2. Choosing α_m by solving

$$\alpha_m = \tfrac{1}{2} \ln\Bigl(\tfrac{1-\varepsilon_m}{\varepsilon_m}\Bigr), \quad \text{where} \quad \varepsilon_m = \frac{\sum_{i \in M_m} w_i^{(m)}}{\sum_{i=1}^{N} w_i^{(m)}}. \tag{10}$$

(The quantity ε_m is the *weighted* fraction of training points misclassified by h_m.)

Finally, having determined α_m and h_m, we update the weights for the next iteration:

$$w_i^{(m+1)} = w_i^{(m)} \exp\bigl(-\alpha_m\, y_i\, h_m(x_i)\bigr), \tag{11}$$

and then normalize so that $\sum_i w_i^{(m+1)} = 1$. In this way, the training examples that were misclassified by h_m (i.e. those in M_m) receive increased weight, so they are emphasized more strongly in the subsequent round of boosting.
Repeating this process over $m = 1, 2, \ldots, M$ yields the final ensemble classifier

$$H(x) = \operatorname{Sign}\Bigl(F_M(x)\Bigr) = \operatorname{Sign}\Bigl(\sum_{\ell=1}^{M} \alpha_\ell\, h_\ell(x)\Bigr). \tag{12}$$

Because a constant factor does not change the sign, one often omits the factor of $\frac{1}{2}$ in front of the sum. This framework shows that boosting, as implemented in AdaBoost, can be seen as stagewise minimization of the exponential loss.
AdaBoost is exactly a prototypical example of this sequential exponential loss minimization approach.

2.2 Algorithm

Key components used within AdaBoost:

- x: Input features.
- y: True labels in $\{-1, +1\}$.
- $h_t(x)$: Weak classifier output at round t.
- ε_t: Weighted error of h_t with respect to p_t.
- α_t: Weight assigned to h_t, computed as $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
- $p_t(i)$: Weight of training example i at round t.
- Z_t: Normalization constant ensuring $\sum_{i=1}^{N} p_{t+1}(i) = 1$.

Step 1. Training the WLO
At round t, a weak learner produces a classifier $h_t : \mathcal{X} \to \{-1, +1\}$ using the current distribution p_t. The weighted error of h_t is defined as

$$\varepsilon_t = \sum_{i=1}^{N} p_t(i) \cdot \mathbb{1}\{h_t(x_i) \neq y_i\}, \tag{13}$$

where $\mathbb{1}\{\cdot\}$ denotes the indicator function.
Step 2. Determining the Weight α_t for WLO
The goal is to update the combined classifier $F_t(x) = F_{t-1}(x) + \alpha_t\, h_t(x)$, with the initial condition $F_0(x) = 0$. The exponential loss is given by

$$E = \sum_{i=1}^{N} \exp(-y_i\, F_t(x_i)). \tag{14}$$

After adding the new classifier h_t, the loss becomes

$$E = \sum_{i=1}^{N} \exp\Big(-y_i \Big[F_{t-1}(x_i) + \alpha_t\, h_t(x_i)\Big]\Big). \tag{15}$$

We define the weights at round t as

$$p_t(i) = \frac{\exp(-y_i\, F_{t-1}(x_i))}{Z_{t-1}}, \tag{16}$$

where Z_{t-1} is a normalization constant ensuring $\sum_{i=1}^{N} p_t(i) = 1$. Then the loss can be written as

$$E = Z_{t-1} \sum_{i=1}^{N} p_t(i) \exp(-\alpha_t\, y_i\, h_t(x_i)). \tag{17}$$

Since Z_{t-1} does not depend on α_t, we minimize

$$E(\alpha_t) = \sum_{i=1}^{N} p_t(i) \exp(-\alpha_t\, y_i\, h_t(x_i)). \tag{18}$$

Note that y_i and $h_t(x_i)$ take values in $\{-1, +1\}$, so their product is either $+1$ (if correctly classified) or -1 (if misclassified). Therefore, we can split the sum as

$$E(\alpha_t) = \exp(-\alpha_t) \sum_{i:h_t(x_i)=y_i} p_t(i) + \exp(\alpha_t) \sum_{i:h_t(x_i)\neq y_i} p_t(i). \tag{19}$$

Defining $1 - \varepsilon_t = \sum_{i:h_t(x_i)=y_i} p_t(i)$ and $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} p_t(i)$, the loss becomes

$$E(\alpha_t) = (1 - \varepsilon_t) \exp(-\alpha_t) + \varepsilon_t \exp(\alpha_t). \tag{20}$$

To minimize $E(\alpha_t)$, we differentiate with respect to α_t and set the derivative equal to zero:

$$-(1 - \varepsilon_t) \exp(-\alpha_t) + \varepsilon_t \exp(\alpha_t) = 0. \tag{21}$$

Solving for α_t, we obtain

$$\exp(2\alpha_t) = \frac{1 - \varepsilon_t}{\varepsilon_t}, \tag{22}$$

which yields

$$\alpha_t = \frac{1}{2} \ln\Big(\frac{1 - \varepsilon_t}{\varepsilon_t}\Big). \tag{23}$$

Step 3. Updating the Distribution and Normalization Constant Z_t
After computing α_t, we update the weights of the training examples as follows:

$$p_{t+1}(i) = \frac{p_t(i) \exp(-\alpha_t \, y_i \, h_t(x_i))}{Z_t}, \tag{24}$$

where the normalization constant Z_t is defined by

$$Z_t = \sum_{i=1}^{N} p_t(i) \exp(-\alpha_t \, y_i \, h_t(x_i)). \tag{25}$$

Expanding Z_t using the same split as before, we have

$$Z_t = (1 - \varepsilon_t) \exp(-\alpha_t) + \varepsilon_t \exp(\alpha_t). \tag{26}$$

Step 4. Deriving the Final Classifier
After T rounds, the prediction of the final strong classifier is given by

$$H(x) = \text{Sign}(\sum_{t=1}^{T} \alpha_t \, h_t(x)). \tag{27}$$

```
Algorithm 2.1: AdaBoost
Input: Training set $S$, weak learner $A$
Initialize: $p_1(i) = \frac{1}{N}$ for all $i \in [N]$
for $t = 1$ to $T$:
    Train weak classifier $h_t = A(S, p_t)$.
    Compute weighted error $\varepsilon_t = E_{i \sim p_t}[\mathbb{1}\{h_t(x_i) \neq y_i\}]$.
    Compute $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$.
    Update weights:
        $p_{t+1}(i) \propto p_t(i) \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{otherwise} \end{cases}$
Output: Final classifier:
    $H(x) = \text{Sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$.
```

2.3 Intuition behind AdaBoost

At any time t, we have:

- when $\varepsilon_t \to \frac{1}{2}$ $\left[\text{almost random classifier}\right]$ $\Rightarrow \alpha_t \to 0$
- when $\varepsilon_t \to 0$ $\left[\text{very good classifier}\right]$ $\Rightarrow \alpha_t \to \infty$

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$$
$$\varepsilon_t = E_{i \sim p_t}\left[\mathbb{1}\left\{h_t(x_i) \neq y_i\right\}\right]$$
$$Z_t = \sum_{i=1}^{N} p_t(i) \exp\left(-\alpha_t y_i h_t\left(x_i\right)\right).$$

As such, we have:

- if $\varepsilon_t \to \frac{1}{2} \Rightarrow \alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \to 0$, where we set w_t on $h_t \to 0$, since h_t is a nearly random classifier.
- if $\varepsilon_t \to 0 \Rightarrow$ implies $\gamma_t \to \frac{1}{2}$, which is almost an accurate classifier. Hence, $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \to \infty$, which almost putting infinite w_t on h_t.

Thus the update step suggests:

$$p_{t+1}(i) \;\propto\; \begin{cases} p_t(i)\, e^{-\alpha_t}, & \text{if } h_t(x_i) = y_i, \\ p_t(i)\, e^{+\alpha_t}, & \text{if } h_t(x_i) \neq y_i. \end{cases}$$

So, when $\alpha_t \to 0$ $\big(\Leftrightarrow$ mostly random classifier$\big)$, then

$$p_{t+1}(i) \approx p_t(i).$$

When $\alpha_t \to \infty$ $\big(\Leftrightarrow$ very good classifier$\big)$, then

$$p_{t+1}(i) \;\to\; \infty \quad \text{if misclassified,} \qquad p_{t+1}(i) \;\to\; 0 \quad \text{if correctly classified.}$$

2.4 Example

Figure 2 illustrates the AdaBoost algorithm using a subset of 30 data points taken from the toy classification dataset shown in Figure 1. Here each base learners consists of a threshold on one of the input variables. This simple classifier corresponds to a form of decision tree known as a "decision stumps", i.e., a decision tree with a single node. Thus each base learner classifies an input according to whether one of the input features exceeds some threshold and therefore simply partitions the space into two regions separated by a linear decision surface that is parallel to one of the axes.

2.5 Error Bounds

Theorem 2.1: Training Error of AdaBoost after T Rounds

After T rounds, the error rate of the final output of AdaBoost is bounded by:

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{H(x_i) \neq y_i\} \leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right), \tag{28}$$

where $\gamma_t = \frac{1}{2} - \varepsilon_t$ is the edge of classifier h_t. Under the weak learning assumption, the error rate is then bounded by $\exp\left(-2T\gamma^2\right)$ and equal to 0 as long as $T > \frac{\ln N}{2\gamma^2}$.

Under the weak learning assumption, this bound simplifies to $\exp(-2T\gamma^2)$, ensuring zero training error when:

$$T > \frac{\ln N}{2\gamma^2}. \tag{29}$$

Remark 2. *Specifically, if* $\min_{t=1}^{T}\gamma_t = \gamma$*, then*

$$\exp\Big(-2\sum_{t=1}^{T}\gamma_t^2\Big) \leq \exp\big(-2T\gamma^2\big).$$

Moreover, the training error $\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}\{H(x_i) \neq y_i\}$ *can only take values in* $0, \frac{1}{N}, \frac{2}{N}, \frac{3}{N}, \ldots$ *Hence, if we set* $T > \frac{\ln N}{2\gamma^2}$*, then*

$$\exp(-2T\gamma^2) < \exp(-2\frac{\ln N}{2\gamma^2}\gamma^2) = \frac{1}{N}.$$

This means that the training error $< \frac{1}{N}$ *but the only possibility is training error* $= 0$*. So after* $T > \frac{\ln N}{2\gamma^2}$ *rounds, the training error of AdaBoost* $\to 0$*.*

Proof. Let $F(x) = \sum_{t=1}^{T}\alpha_t h_t(x)$ so that $H(x) = \text{Sign}(F(x))$. The first step is to realize 0-1 loss is bounded by the exponential loss

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}\{H(x_i) \neq y_i\} = \sum_{i=1}^{N} p_1(i)\mathbb{1}\{y_i F(x_i) \leq 0\} \leq \sum_{i=1}^{N} p_1(i)\exp(-y_i F(x_i)).$$

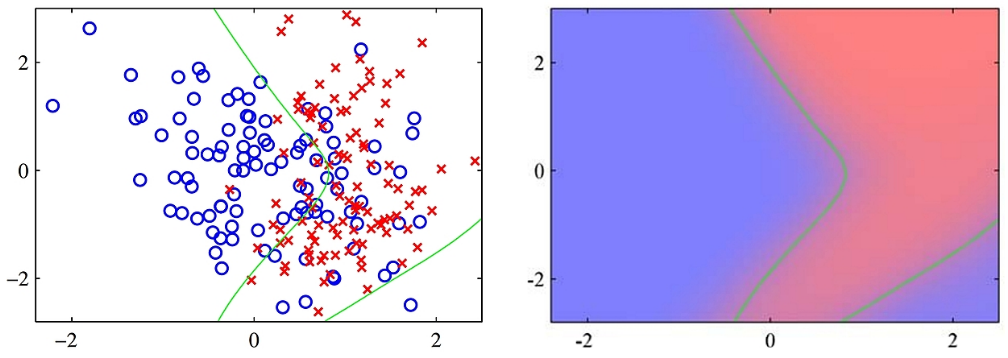

Figure 1: The left plot shows the synthetic classification data set with data from the two classes shown in red and blue. On the right is a plot of the true posterior probabilities, shown on a color scale going from pure red denoting probability of the red class is 1 to pure blue denoting probability of the red class is 0. Because these probabilities are known, the optimal decision boundary for minimizing the misclassification rate (which corresponds to the contour along which the posterior probabilities for each class equal 0.5) can be evaluated and is shown by the green curve. This decision boundary is also plotted on the left-hand figure.

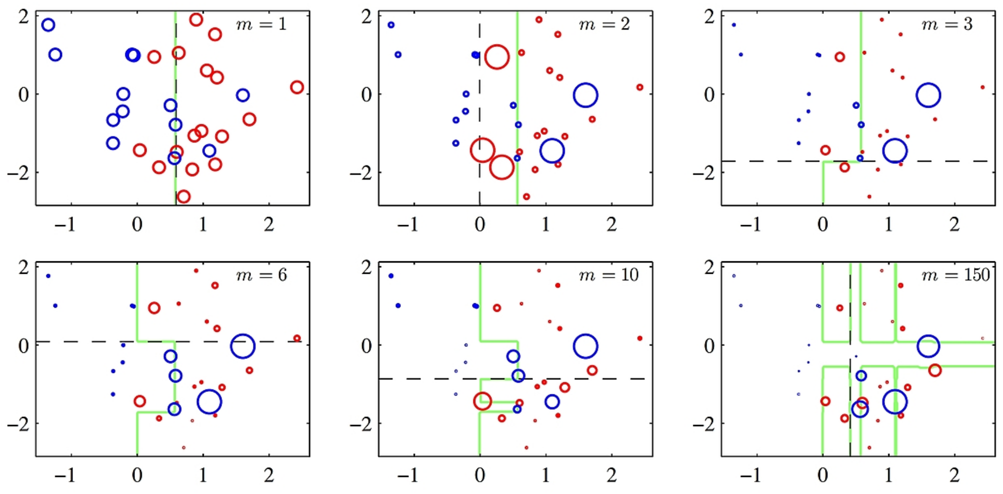

Figure 2: Illustration of boosting in which the base learners consist of simple thresholds applied to one or other of the axes. Each figure shows the number m of base learners trained so far, along with the decision boundary of the most recent base learner (dashed black line) and the combined decision boundary of the ensemble (solid green line). Each data point is depicted by a circle whose radius indicates the weight assigned to that data point when training the most recently added base learner. Thus, for instance, we see that points that are misclassified by the $m = 1$ base learner are given greater weight when training the $m = 2$ base learner.

Now notice that the update rule of p_t can be written as $p_{t+1}(i) = p_t(i)\exp(-y_i\alpha_t h_t(x_i))/Z_t$ where Z_t is the normalization factor. We then have

$$p_{T+1}(i) = p_1(i)\prod_{t=1}^{T}\frac{\exp(-y_i\alpha_t h_t(x_i))}{Z_t} = \frac{p_1(i)\exp(-y_iF(x_i))}{\prod_{t=1}^{T} Z_t},$$

and therefore the error rate is bounded by

$$\sum_{i=1}^{N}\left(p_{T+1}(i)\prod_{t=1}^{T} Z_t\right) = \prod_{t=1}^{T} Z_t.$$

It remains to bound each Z_t:

$$\begin{aligned} Z_t &= \sum_{i=1}^{N} p_t(i)\exp(-y_i\alpha_t h_t(x_i)) \\ &= \sum_{i:h_t(x_i)=y_i} p_t(i)\exp(-\alpha_t) + \sum_{i:h_t(x_i)\neq y_i} p_t(i)\exp(\alpha_t) \\ &= (1-\varepsilon_t)\exp(-\alpha_t) + \varepsilon_t\exp(\alpha_t) \end{aligned}$$

Recall that by definition, we have:

$$\alpha_t = \frac{1}{2}\ln\Big(\frac{1-\varepsilon_t}{\varepsilon_t}\Big). \tag{30}$$

Thus, we have

$$\exp(-\alpha_t) = \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} \quad \text{and} \quad \exp(\alpha_t) = \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}. \tag{31}$$

Substituting these terms into Z_t provides us with:

$$Z_t = (1-\varepsilon_t)\sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} + \varepsilon_t\sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}. \tag{32}$$

Simplify each term, we get:

$$(1-\varepsilon_t)\sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} = \sqrt{\varepsilon_t(1-\varepsilon_t)} \quad \text{and} \quad \varepsilon_t\sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} = \sqrt{\varepsilon_t(1-\varepsilon_t)}. \tag{33}$$

Therefore,

$$Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}. \tag{34}$$

Let us define the edge γ_t as:

$$\gamma_t = \frac{1}{2} - \varepsilon_t, \tag{35}$$

such that

$$\varepsilon_t = \frac{1}{2} - \gamma_t \quad \text{and} \quad 1-\varepsilon_t = \frac{1}{2} + \gamma_t. \tag{36}$$

It follows that

$$\varepsilon_t(1-\varepsilon_t) = \Big(\frac{1}{2} - \gamma_t\Big)\Big(\frac{1}{2} + \gamma_t\Big) = \frac{1}{4} - \gamma_t^2. \tag{37}$$

Substituting into the expression for Z_t gives

$$Z_t = 2\sqrt{\frac{1}{4} - \gamma_t^2} = \sqrt{1 - 4\gamma_t^2}. \tag{38}$$

By the fact that the Maclaurin series expansion of e^z is

$$e^z = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \cdots,$$

we know that:

$$1 + z \leq e^z.$$

The equality holds when $z = 0$. Continuing Eq. 38, we have:

$$\sqrt{1 + (-4\gamma_t^2)} \leq \sqrt{\exp\left(-4\gamma_t^2\right)} = \exp\left(-2\gamma_t^2\right) \tag{39}$$

As such, we have:

$$Z_t \leq \exp\left(-2\gamma_t^2\right) \tag{40}$$

This finishes the proof for Eq. 28. Under the weak learning assumption, we further have $\gamma_t \geq \gamma$ and thus the stated bound $\exp(-2T\gamma^2)$. Finally, note that as soon as the error rate drops below $1/N$, it must become zero. Therefore, as long as $\exp(-2T\gamma^2) < 1/N$, which means

$$T > \frac{\ln N}{2\gamma^2},$$

AdaBoost ensures zero training error. □

2.6 Generalization Error

According to standard VC theory, the difference between the generalization error and the training error is bounded by something like $\tilde{\mathcal{O}}\left(\sqrt{C/N}\right)$, where C is some complexity measure of the hypothesis space. This complexity for boosting grows as $\mathcal{O}(T)$, since combining more weak classifiers leads to a more complicated final classifier H. We have proved that after $T > \frac{\ln N}{2\gamma^2}$ rounds the training error of AdaBoost becomes 0. Hence, if we stop the algorithm at the right time, AdaBoost will have generalization error of order $\tilde{\mathcal{O}}\left(\sqrt{\frac{\ln N}{\gamma^2 N}}\right)$, which can be arbitrarily small when N is large enough.

Remark 3. *The above proposition indicates that under the weak learning assumption, AdaBoost does ensure arbitrarily small generalization error given enough examples, implying that "boosting" is indeed possible, or in more technical terms, weak learnability is equivalent to strong learnability.*

2.7 Prevention of Overfitting

In practice, AdaBoost tends to prevent overfitting, in the sense that even if one keeps running the algorithm for many more rounds after the training error has dropped to zero, the generalization error still keeps decreasing. It turns out that the concept of *margin* is the key for understanding this phenomenon. The margin of an example (x, y) w.r.t. the classifier H is defined as $yf(x)$ where

$$f(x) = \frac{F(x)}{\sum_{t=1}^{T} \alpha_t} = \sum_{t=1}^{T} \left(\frac{\alpha_t}{\sum_{\tau=1}^{T} \alpha_\tau}\right) h_t(x).$$

It is clear that the margin is always in $[-1, 1]$, and the sign of the margin indicates whether the final classifier H makes a mistake on x or not. Specifically we want the margin to be positive in order to have low error rate. However, we also want the margin to be a large positive (close to 1), since in this case there is a huge difference between the vote for $+1$ and -1, and the decisive win of one label makes us feel more confident about the final prediction.
Indeed, margin theory says that for any θ, the generalization error of H and the fraction of training examples with margin at most θ are related as follows:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathbb{1}\{H(x) \neq y\}] \leq \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{y_i f(x_i) \leq \theta\} + \tilde{\mathcal{O}}\left(\frac{1}{\theta}\sqrt{\frac{C_{\mathcal{H}}}{N}}\right),$$

where $C_{\mathcal{H}}$ is the complexity of the hypothesis space $\mathcal{H}$ used by the oracle (recall $h_t \in \mathcal{H}$), which is independent of the number of weak classifiers combined by H.
Therefore, if keeping running AdaBoost increases the margin even after the training error drops to zero, then it explains why overfitting does not happen. This is true because under the weak learning assumption AdaBoost guarantees

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{y_i f(x_i) \leq \theta\} \leq \left(\sqrt{(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta}}\right)^T.$$

One way to interpret this bound is to note that as long as θ is such that $(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta} < 1$, which translates to

$$\theta \leq \Gamma(\gamma) \triangleq \frac{-\ln(1-4\gamma^2)}{\ln\left(\frac{1+2\gamma}{1-2\gamma}\right)} \leq 2\gamma,$$

then the fraction of examples with margin at most θ is eventually zero when T is large enough. In other words, if we keep running AdaBoost, eventually the smallest margin is $\Gamma(\gamma)$ and the generalization error is thus

$$\tilde{\mathcal{O}}\left(\frac{1}{\Gamma(\gamma)}\sqrt{\frac{C_{\mathcal{H}}}{N}}\right).$$

Remark 4. *AdaBoost is not doing the best possible job in maximizing the smallest margin. The best possible smallest margin under the weak learning assumption can be shown to be exactly 2γ.*

Next Lecture

The next lecture will cover the following topics:
(i) Midterm solutions,
(ii) PCA.

References:

1. Freund, Yoav, 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 (1997): 119-139.

2. Schapire, Robert E. "The strength of weak learnability." Machine learning 5 (1990): 197-227.

3. Bishop, Christopher M., and Nasser M. Nasrabadi. Pattern recognition and machine learning. Vol. 4. No. 4. New York: springer, 2006.

4. Hertzmann, Aaron, David Fleet, and Marcus Brubaker. ”Machine learning and data mining lecture notes.” Computer Science Department, University of Toronto (2012).