Lectures on Machine Learning (Fall 2017)

Hyeong In Choi Seoul National University

Lecture 11: Boosting (I)
AdaBoost (Draft: version 0.9.2)

Topics to be covered:

- Plain vanilla AdaBoost
- Multi-class AdaBoost
- Generalized versions of AdaBoost
- Multi-label classification
  - What it is and why
  - Multi-label ranking problem
- Generalization error

AdaBoost (Adaptive Boosting) is a generic name (rather, meta-algorithm) that stands for a group of algorithms developed by Schapire and Freund and others that combine a host of rather weak (not so accurate) classifiers to create a strong (accurate) classifier. In this lecture we first present the original form of AdaBoost and then proceed to more general ones. In particular we start with Binary AdaBoost and AdaBoost.M1 for multiclass classification. We then introduce a slightly more general form of Binary AdaBoost and then cover AdaBoost.MH and AdaBoost.MR. For a general reference, one is referred to [1].

1 Binary AdaBoost

1.1 Standard form of binary AdaBoost

The following version of Binary AdaBoost, which is equivalent to the original one given in Section 1.3, is the most direct and easiest to understand. All other versions of AdaBoost are essentially its variations.

Binary AdaBoost
Input:

- Data: $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, where $x^{(i)} \in \mathfrak{X}$ and $y^{(i)} \in \mathfrak{Y} = \{-1, 1\}$ for $i = 1, \cdots, N$
- Total number of rounds: T

Initialize: Define the initial probability distribution $D_1(i) = \dfrac{1}{N}$ for $i = 1, \cdots, N$.
Do for $t = 1, \cdots, T$:

- Train using the probability distribution D_t on $\{1, \cdots, N\}$.
- Get a hypothesis (classifier) $h_t : \mathfrak{X} \to \mathfrak{Y}$ that has a training error rate ε_t (with respect to D_t) given by

$$\varepsilon_t = \sum_{i: h_t(x^{(i)}) \neq y^{(i)}} D_t(i).$$

- If $\varepsilon_t > \dfrac{1}{2}$, then set $T = t - 1$ and abort the loop.
- Set $\alpha_t = \dfrac{1}{2} \log\left(\dfrac{1 - \varepsilon_t}{\varepsilon_t}\right)$.
- Define the new probability distribution D_{t+1} by setting

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y^{(i)} = h_t(x^{(i)}) \\ e^{\alpha_t} & \text{if } y^{(i)} \neq h_t(x^{(i)}) \end{cases} \\ &= \frac{D_t(i)}{Z_t} \exp(-\alpha_t y^{(i)} h_t(x^{(i)})), \end{aligned}$$

where

$$Z_t = \sum_{i=1}^{N} D_t(i) \exp(-\alpha_t y^{(i)} h_t(x^{(i)})).$$

Output: Define $g(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$. The final hypothesis (classifier) $H(x)$ is

$$H(x) = \operatorname{sgn}(g(x)).$$

Note that ε_t is an estimator of $\text{Prob}_{i\sim D_t}(h_t(x^{(i)}) \neq y^{(i)})$, and Z_t that of $E_{i\sim D_t} \exp(-\alpha_t y^{(i)} h_t(x^{(i)}))$. The following theorem is the key to analyzing Binary AdaBoost and its subsequent generalizations.

Theorem 1. *The training error $E_{train}(H)$ of H satisfies*

$$E_{train}(H) = \frac{1}{N}\big|\{i : H(x^{(i)}) \neq y^{(i)}\}\big| \leq \prod_{t=1}^{T} Z_t.$$

Proof. Note that by repeatedly applying the definition of Z_t, we have

$$\begin{aligned} D_{T+1}(i) &= \frac{D_T(i)}{Z_T} e^{-\alpha_T y^{(i)} h_T(x^{(i)})} \\ &\vdots \\ &= D_1(i) \frac{e^{-y^{(i)} \sum_{t=1}^{T} \alpha_t h_t(x^{(i)})}}{\prod_{t=1}^{T} Z_t} \\ &= \frac{1}{N} \frac{e^{-y^{(i)} g(x^{(i)})}}{\prod_{t=1}^{T} Z_t}. \end{aligned} \tag{1}$$

Now if $H(x) \neq y$, then $yg(x) \leq 0$. Thus $e^{-yg(x)} \geq 1$. Therefore

$$\begin{aligned} E_{train}(H) &= \frac{1}{N} \sum_i \begin{cases} 1 & \text{if } H(x^{(i)}) \neq y^{(i)} \\ 0 & \text{else} \end{cases} \\ &\leq \frac{1}{N} \sum_i e^{-y^{(i)} g(x^{(i)})} \\ &= \sum_i D_{T+1}(i) \prod_t Z_t, \qquad \text{(by (1))} \\ &= \prod_t Z_t, \end{aligned}$$

where the last equality is due to the fact that D_{T+1} is a probability distribution. □

This theorem suggests that in order to make the training error small, one has to successively make Z_t as small as possible.

Corollary 1. *Let α_t and ε_t be given as above. Then*

$$Z_t = \sqrt{4\varepsilon_t(1-\varepsilon_t)}. \tag{2}$$

Thus

$$E_{train}(H) \leq \prod_{t=1}^{T} \sqrt{4\varepsilon_t(1-\varepsilon_t)}. \tag{3}$$

Let $\varepsilon_t = 1/2 - \delta_t$ with $0 < \delta_t < 1/2$. Then

$$E_{train}(H) \leq e^{-2\sum_t \delta_t^2}. \tag{4}$$

Assume further that there is a postive constant δ such that $\delta_t \geq \delta$. Then

$$E_{train}(H) \leq e^{-2\delta^2 T}. \tag{5}$$

Proof. Note that $h_t(x) = y$ if and only if $yh_t(x) = 1$; and $h_t(x) \neq y$ if and only if $yh_t(x) = -1$. Thus

$$\begin{aligned} Z_t &= \sum_{i:h_t(x^{(i)})\neq y^{(i)}} D_t(i)e^{\alpha_t} + \sum_{i:h_t(x^{(i)})=y^{(i)}} D_t(i)e^{-\alpha_t} \\ &= \varepsilon_t e^{\alpha_t} + (1-\varepsilon_t)e^{-\alpha_t} \\ &= \sqrt{4\varepsilon_t(1-\varepsilon_t)}, \end{aligned}$$

where the last equality is due to the definition of α_t. Inequality (3) is the consequence of Theorem 1 and (2). The rest of the proof follows immediately once we use the fact that $\sqrt{1-x} \leq e^{-x/2}$. □

Remark.

(1). Note that

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y^{(i)} = h_t(x^{(i)}) \\ e^{\alpha_t} & \text{if } y^{(i)} \neq h_t(x^{(i)}). \end{cases}$$

This means that the incorrect incidences are assigned higher probabilities than the correct ones. So it has the effect of inducing the new weak learner to focus more on the previously wrong cases at the next round $t+1$.

(2). Let us see why α_t *was set as* $\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$. *In the proof of Corollary 2, we have shown that*

$$Z_t = e^{-\alpha_t}(1-\varepsilon_t) + e^{\alpha_t}\varepsilon_t.$$

With ε_t *fixed,* Z_t *as a function of* α_t *takes on the minimum value when its derivative with respect to* α_t *is zero. In other words,* α_t *must satisfy:*

$$-e^{-\alpha_t}(1-\varepsilon_t) + e^{\alpha_t}\varepsilon_t = 0.$$

Solving for α_t, *we get what we want. In other words,* α_t *is chosen to minimize* Z_t, *which is part of the upper bound of the training error as in Theorem 1.*

(3). In this version of AdaBoost, the iterative round aborts if $\varepsilon_t > 1/2$. *But it is not necessary as we can see in the generalized version of Binary AdaBoost presented in Section 3.*

1.2 A worked-out example

Let us now look at a simple example and work out the details of Binary AdaBoost. This way, one can best understand the inner workings of AdaBoost.

Let the data $\mathfrak{D}$ consist of five points in the plane as given in Figure 1. Two points $(0.5, 1.5)$ and $(1.5, 1.5)$ are labeled with $+$ mark and three points $(1.5, 0.5)$, $(2.5, 1.5)$ and $(2.5, 2.5)$ are labeled with $-$ mark.

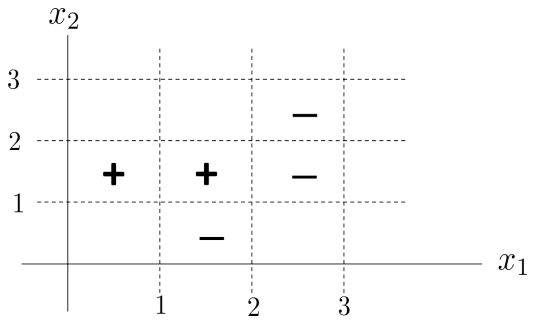

Figure 1: Data

In this example, we assume that the only weak learners available to us are the four binary classifiers whose classification boundary is either the

horizontal line $x_2 = 1$ or $x_2 = 2$ or the vertical line $x_1 = 1$ or $x_1 = 2$. Let us suppose we chose the weak learner h_1 with $x_1 = 2$ as its classification boundary as shown in Figure 2.

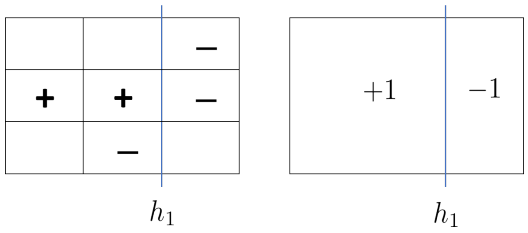

Figure 2: Weak learner h_1

If we assign $+1$ to the left of the line $x_1 = 2$ and -1 to the right as in the right figure of Figure 2, the classification error occurs only at $(1.5, 0.5)$. Therefore we have

$$\begin{aligned} \varepsilon_1 &= \frac{1}{5} \\ \alpha_1 &= \frac{1}{2}\log\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right) = \log 2 = 0.69. \end{aligned}$$

Figure 3 depicts $\alpha_1 h_1$. Since $\alpha_1 = \log 2$, $e^{\alpha_1} = 2$ and $e^{-\alpha_1} = 1/2 = 0.5$.

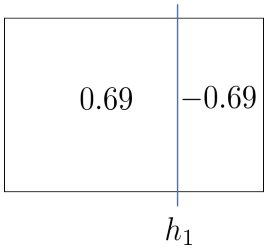

Figure 3: $\alpha_1 h_1$

Therefore we have

$$D_2(i) \;\propto\; D_1(i) \begin{cases} e^{\alpha_1} & \text{at } (1.5, 0.5) \\ e^{-\alpha_1} & \text{elsewhere,} \end{cases}$$

where the symbol $\propto$ indicates the proportionality, i.e., the equality up to a constant. Using this and the fact that $D_1(i) = 1/5$, one can easily calculate

		1/8
1/8	1/8	1/8
	4/8	

Figure 4: Distribution D_2

D_2, which is depicted in Figure 4. Let us now set the next weak learner h_2 to be the one with the classification boundary $x_2 = 1$ whose value is set as $+1$ in the region above the line $x_2 = 1$ and -1 below it. This situation is depicted in Figure 5. Then the misclassified points are $(2.5, 1.5)$ and $(2.5, 2.5)$. Each

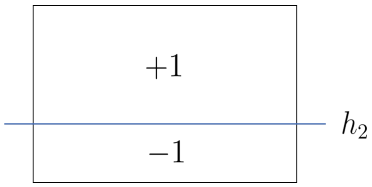

Figure 5: Weak learner h_2

has D_2 probability 1/8, and the error rate of h_2 is 1/4. Thus we have

$$\begin{aligned} \varepsilon_2 &= \frac{1}{4} \\ \alpha_2 &= \frac{1}{2}\log\left(\frac{1-\varepsilon_2}{\varepsilon_2}\right) = \log\sqrt{3} = 0.55. \end{aligned}$$

Figure 6 shows $\alpha_2 h_2$. If we were to stop the AdaBoost here and declare

$$g = \alpha_1 h_1 + \alpha_2 h_2,$$

then the values of g should be as shown in Figure 7. Note that with this g the final classifier $H(x_1) = \mathrm{sgn}(g(x_1))$ still has an error at $(1.5, 0.5)$. So let us go one more round. To do that, we need to first calculate D_3. Note that

$$\begin{aligned} e^{\alpha_2} &= e^{\log\sqrt{3}} = \sqrt{3} = 1.732 \\ e^{-\alpha_2} &= 1/\sqrt{3} = 0.577. \end{aligned}$$

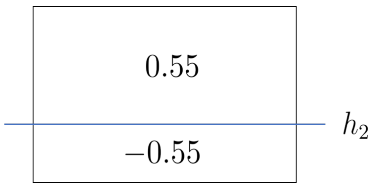

Figure 6: $\alpha_2 h_2$

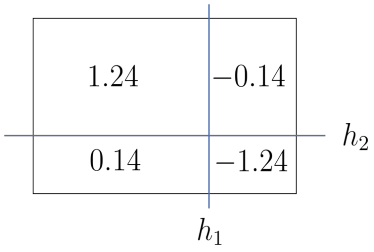

Figure 7: $\alpha_1 h_1 + \alpha_2 h_2$

Therefore we have

$$D_3(i) \quad \sim \quad D_2(i) \begin{cases} e^{\alpha_2} & \text{at } (2.5, 1.5) \text{ or } (2.5, 2.5) \\ e^{-\alpha_2} & \text{elsewhere.} \end{cases}$$

With this, it is easy to calculate D_3, which is depicted in Figure 8. Now let us take another weak classifier h_3 that has the classification boundary $x_1 = 1$ with $+1$ on the left of the line $x_1 = 1$ and -1 on the right. This h_3 is depicted in Figure 9. This h_3 misclassifies one point, that is, $(1.5, 1.5)$. The

		3/12
1/12	1/12	3/12
	4/12	

Figure 8: Distribution D_3

misclassification rate of h_3 with respect to the distribution D_3 is then 1/12.

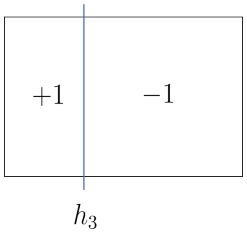

Figure 9: h_3

Therefore $\alpha_3 h_3$ is as depicted in Figure 10

$$\begin{aligned} \varepsilon_3 &= 1/12 = 0.0833 \\ \alpha_3 &= \frac{1}{2}\log\left(\frac{1-\varepsilon_3}{\varepsilon_3}\right) \approx 1.2. \end{aligned}$$

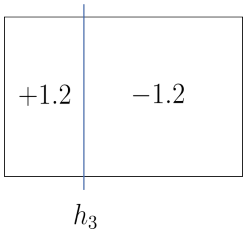

Figure 10: $\alpha_3 h_3$

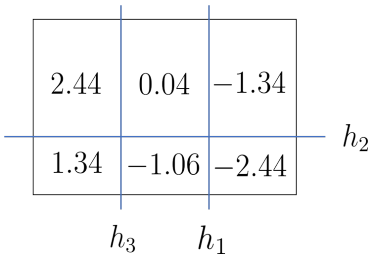

Figure 11: $g = \alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3$

Combining all of them together we can get

$$g = \alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3, \tag{6}$$

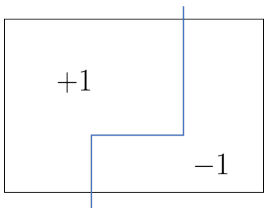

Figure 12: Final classifier $H(x)$

which is depicted in Figure 11. Taking the sign of g, the final classifier $H(x_1)$ is obtained, which is depicted in Figure 12.

This worked-out example illustrates how AdaBoost works and how a seemingly linear combination of linear boundaries as in (6) can produce a nonlinear classification boundary. All the subsequent generalizations of AdaBoost work more or less in the same spirit.

1.3 Original form of AdaBoost

AdaBoost in the original formalism of Schaprie and Freund in [3] has the following form.

Binary AdaBoost (in original form)
Input:

- Data: $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, where $x^{(i)} \in \mathfrak{X}$ and $\{0, 1\}$.
- Total number of rounds: T

Initialize: the weight vector $w_{1,i} = \dfrac{1}{N}$ for $i = 1, \cdots, N$.

- Set $p_{t,i} = \dfrac{w_{t,i}}{\sum_{i=1}^{N} w_{t,i}}$
- Train $h_t : \mathfrak{X} \to \{0, 1\}$ using the probability p_t
- Calculate the error (w.r.t. p_t)

$$\varepsilon_t = \sum_i p_{t,i} |h_t(x^{(i)}) - y^{(i)}|$$

- (If $\varepsilon_t > \dfrac{1}{2}$, set $T = t-1$ and abort the loop.)
- Set $\beta_t = \dfrac{\varepsilon_t}{1-\varepsilon_t}$
- Set the weight vector w_{t+1} by setting

$$w_{t+1,i} = w_{t,i}\, \beta_t^{1-|h_t(x^{(i)})-y^{(i)}|}$$

Output: The final hypothesis (classifier) is

$$H(x) \;=\; \begin{cases} 1 & \text{if } \sum_t \log(1/\beta_t)\, h_t(x) \geq \dfrac{1}{2}\sum_t \log(1/\beta_t) \\ 0 & \text{else.} \end{cases}$$

Let us see that this formalism is equivalent to the Standard form of binary AdaBoost given in Section 1.1. First, check that $\log(1/\beta_t) = 2\alpha_t$. Note that the hypothesis h_t in this case has value $\{0,1\}$, while the hypothesis h_t in the version given in Section 1.1 has value $\{-1,1\}$. To distinguish them, let us use the notation $\tilde{h_t}$ for this version, while keeping the same h_t for the version given in Section 1.1. So we have $\tilde{h_t} = (1+h_t)/2$, with which it is easy to see that the formula

$$\sum_t \log(1/\beta_t)\, \tilde{h_t}(x) \geq \frac{1}{2}\sum_t \log(1/\beta_t)$$

is equivalent to

$$\sum_t \alpha_t h_t(x) \geq 0.$$

So the outputs of the two versions are equivalent. Now the weight adjustment scheme of this version is

$$w_{t+1,i} = \begin{cases} w_{t,i}\beta_t = w_{t,i}e^{-2\alpha_t} & \text{if } y^{(i)} = h_t(x^{(i)}) \\ w_{t,i} & \text{if } y^{(i)} \neq h_t(x^{(i)}). \end{cases}$$

On the other hand, in the version given in Section1.1, the distribution adjustment scheme is

$$\begin{aligned} D_{t+1}(i) \;&\propto\; D_t(i) \cdot \begin{cases} e^{-\alpha_t} & \text{if } y^{(i)} = h_t(x^{(i)}) \\ e^{\alpha_t} & \text{if } y^{(i)} \neq h_t(x^{(i)}) \end{cases} \\ &\propto\; D_t(i) \cdot \begin{cases} e^{-2\alpha_t} & \text{if } y^{(i)} = h_t(x^{(i)}) \\ 1 & \text{if } y^{(i)} \neq h_t(x^{(i)}). \end{cases} \end{aligned}$$

Therefore these two schemes are seen to be equivalent. In fact we can check that

$$p_{t,i} = D_t(i),$$

after normalization

$$p_{t,i} = \frac{w_{t,i}}{\sum_{i=1}^{N} w_{t,i}}.$$

2 Multi-class AdaBoost

Binary AdaBoost can be easily generalized. The most direct method is as follows:

AdaBoost.M1
Input:

- Data: $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, where $x^{(i)} \in \mathfrak{X}$ and $y^{(i)} \in \mathfrak{Y}$ for $i = 1, \cdots, N$. Here $\mathfrak{Y}$ is a set of K elements, which, as usual, is identified with $\{1, \cdots, K\}$
- Total number of rounds: T

Initialize: Define the initial probability $D_1(i) = \dfrac{1}{N}$ for $i = 1, \cdots, N$.
Do for $t = 1, 2, \cdots, T$:

- Train using the probability distribution D_t on $\{1, \cdots, N\}$.
- Get a hypothesis (classifier) $h_t : \mathfrak{X} \to \mathfrak{Y}$ that has a training error rate ε_t (with respect to D_t) given by

$$\varepsilon_t = \sum_{i:h_t(x^{(i)}) \neq y^{(i)}} D_t(i).$$

- If $\varepsilon_t > \dfrac{1}{2}$, then set $T = t - 1$ and abort the loop.
- Set $\alpha_t = \dfrac{1}{2} \log \left(\dfrac{1 - \varepsilon_t}{\varepsilon_t} \right)$.

- Define the new probability distribution D_{t+1} by setting

$$\begin{aligned} D_{t+1(i)} &= \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y^{(i)} = h_t(x^{(i)}) \\ e^{\alpha_t} & \text{if } y^{(i)} \neq h_t(x^{(i)}) \end{cases} \\ &= \frac{D_t(i)}{Z_t} \exp(-\alpha_t S(h_t(x^{(i)}), y^{(i)})), \end{aligned}$$

  where $Z_t = \sum_{i=1}^{N} D_t(i) \exp(-\alpha_t S(h_t(x^{(i)}), y^{(i)}))$, and $S(a,b) = \mathbb{I}(a = b) - \mathbb{I}(a \neq b)$ so that $S(a,b) = 1$ if $a = b$ and $S(a,b) = -1$ if $a \neq b$.

Output: the final hypothesis (classifier)

$$H(x) = \operatorname*{argmax}_{y \in \mathfrak{Y}} \sum_{t:h_t(x)=y} \alpha_t.$$

Remark. *In case $\mathfrak{Y} = \{-1, 1\}$, this AdaBoost.M1 is identical to Binary AdaBoost presented above. To see this, first note that in either case all the inputs and algorithmic steps are identical. So we have only to check that the final hypotheses are identical. For Binary AdaBoost, $H(x) = \operatorname{sgn}(g(x))$, where*

$$g(x) = \sum_t \alpha_t h_t(x) = \sum_{t:h_t(x)=1} \alpha_t - \sum_{t:h_t(x)=-1} \alpha_t.$$

Thus $H(x) = 1$ if and only if

$$\sum_{t:h_t(x)=1} \alpha_t \geq \sum_{t:h_t(x)=-1} \alpha_t.$$

This can be seen to hold true if and only if

$$\operatorname*{argmax}_{y \in \mathfrak{Y}} \sum_{t:h_t(x)=y} \alpha_t = 1.$$

The case for $H(x) = -1$ can be proved in exactly the same way.

Theorem 2. *The training error $E_{train}(H)$ of H satisfies*

$$E_{train}(H) = \frac{1}{N} \big| \{i : H(x^{(i)}) \neq y^{(i)}\} \big| \leq \prod_{t=1}^{T} Z_t.$$

Proof. The proof is essentially the same as that of Theorem 1 with some minor modification. First, note that by repeatedly applying the definition of Z_t as before, we can prove that

$$D_{T+1}(i) = \frac{1}{N} \frac{\exp(-\sum_t \alpha_t S(h_t(x^{(i)}), y^{(i)}))}{\prod_{t=1}^T Z_t}. \tag{7}$$

Now assume $H(x^{(i)}) = \widehat{y}^{(i)} \neq y^{(i)}$. Then by the definition of H,

$$\sum_{t:h_t(x^{(i)})=\widehat{y}^{(i)}} \alpha_t \geq \sum_{t:h_t(x^{(i)})=y^{(i)}} \alpha_t. \tag{8}$$

Note that if $H(x^{(i)}) = \widehat{y}^{(i)} \neq y^{(i)}$, there are three possibilities: (1) $h_t(x^{(i)}) = y^{(i)}$, (2) $h_t(x^{(i)}) = \widehat{y}^{(i)}$, and (3) $h_t(x^{(i)}) \neq y^{(i)}, \widehat{y}^{(i)}$. Therefore, using the fact that $S(h_t(x^{(i)}), y^{(i)}) = 1$ if $h_t(x^{(i)}) = y^{(i)}$ and $S(h_t(x^{(i)}), y^{(i)}) = -1$ if $h_t(x^{(i)}) \neq y^{(i)}$, we have

$$\begin{aligned} \sum_t \alpha_t S(h_t(x^{(i)}), y^{(i)}) &= \sum_{t:h_t(x^{(i)})=y^{(i)}} \alpha_t - \sum_{t:h_t(x^{(i)})=\widehat{y}^{(i)}} \alpha_t - \sum_{t:h_t(x^{(i)})\neq y^{(i)},\widehat{y}^{(i)}} \alpha_t \\ &\leq \sum_{t:h_t(x^{(i)})=y^{(i)}} \alpha_t - \sum_{t:h_t(x^{(i)})=\widehat{y}^{(i)}} \alpha_t. \end{aligned} \tag{9}$$

(Here, we used the fact that $\varepsilon_t \leq 1/2$, which in turn implies that $\alpha_t \geq 0$.) By (8), (9) is seen to be non-positive, which implies that $e^{-\sum_t \alpha_t S(h_t(x^{(i)}),y^{(i)})} \geq 1$. Therefore, we can estimate the training error:

$$\begin{aligned} E_{train}(H) &= \frac{1}{N} \sum_i \begin{cases} 1 & \text{if } H(x^{(i)}) \neq y^{(i)} \\ 0 & \text{else} \end{cases} \\ &\leq \frac{1}{N} \sum_i e^{-\sum_t \alpha_t S(h_t(x^{(i)}),y^{(i)})} \\ &= \sum_i D_{T+1}(i) \prod_t Z_t, \qquad \text{by (7)} \\ &= \prod_t Z_t, \end{aligned}$$

where the last equality is due to the fact that D_{T+1} is a probability. □

This theorem also suggests that in order to make the training error small, one has to successively make Z_t as small as possible.

Corollary 2. *Let α_t and ε_t be given as above. Then*

$$Z_t = \sqrt{4\varepsilon_t(1-\varepsilon_t)}.$$

Thus

$$E_{train}(H) \leq \prod_{t=1}^{T} \sqrt{4\varepsilon_t(1-\varepsilon_t)}.$$

Let $\varepsilon_t = 1/2 - \delta_t$ with $0 < \delta_t < 1/2$. Then

$$E_{train}(H) \leq e^{-2\sum_t \delta_t^2}.$$

Assume further that there is a positive constant δ such that $\delta_t \geq \delta$. Then

$$E_{train}(H) \leq e^{-2\delta^2 T}.$$

Proof. Recall that $h_t(x^{(i)}) = y^{(i)}$ if and only if $S(h_t(x^{(i)}), y^{(i)}) = 1$ and $h_t(x^{(i)}) \neq y^{(i)}$ if and only if $S(h_t(x^{(i)}), y^{(i)}) = -1$. Thus

$$\begin{aligned} Z_t &= \sum_{i=1}^{N} D_t(i) \exp(-\alpha_t S(h_t(x^{(i)}), y^{(i)}) \\ &= \sum_{i:h_t(x^{(i)})\neq y^{(i)}} D_t(i) e^{\alpha_t} + \sum_{i:h_t(x^{(i)})=y^{(i)}} D_t(i) e^{-\alpha_t} \\ &= \varepsilon_t e^{\alpha_t} + (1-\varepsilon_t) e^{-\alpha_t} \\ &= 2\sqrt{\varepsilon_t(1-\varepsilon_t)}, \end{aligned}$$

where the last equality follows from the definition of α_t. The rest of the proof proceeds as in that of Corollary 1. □

3 Generalized versions of AdaBoost

Schapire, Freund and others have further developed and generalized AdaBoost in various contexts. In this lecture we present some of the generalizations done by Schapire and Singer [2]. First, the original AdaBoost can be written in the following slightly general form in which h_t is allowed to take on real values rather than the binary values of $\{-1, 1\}$.

Binary AdaBoost (generalized version)
Input:

- Data: $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, where $x^{(i)} \in \mathfrak{X}$ and $y^{(i)} \in \mathfrak{Y} = \{-1, 1\}$ for $i = 1, \cdots, N$
- Total number of rounds: T

Initialize: Define the initial probability distribution $D_1(i) = \dfrac{1}{N}$ for $i = 1, \cdots, N$.
Do for $t = 1, \cdots, T$:

- Train using the probability distribution D_t on $\{1, \cdots, N\}$.
- Get a hypothesis (classifier) $h_t : \mathfrak{X} \to \{-1, 1\}$.
- Choose α_t.
- Define the new probability distribution D_{t+1} by setting

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y^{(i)} = h_t(x^{(i)}) \\ e^{\alpha_t} & \text{if } y^{(i)} \neq h_t(x^{(i)}) \end{cases} \\ &= \frac{D_t(i)}{Z_t} \exp(-\alpha_t y^{(i)} h_t(x^{(i)})), \end{aligned}$$

  where

$$Z_t = \sum_{i=1}^{N} D_t(i) \exp(-\alpha_t y^{(i)} h_t(x^{(i)})).$$

Output: Define $g(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$. The final hypothesis (classifier) $H(x)$ is

$$H(x) = \operatorname{sgn}(g(x)).$$

The analysis of training error for this generalized version of Binary AdaBoost is very similar to what was done before. Most of all, the following theorem, the proof of which is verbatim the same as that of Theorem 1, is the most basic.

Theorem 3. *The training error $E_{train}(H)$ of the generalized version of Binary AdaBoost satisfies*

$$E_{train}(H) = \frac{1}{N} \big| \{i : H(x^{(i)}) \neq y^{(i)}\} \big| \leq \prod_{t=1}^{T} Z_t.$$

In here, we only deal with the case where $|h_t| \leq 1$. First, using the convexity of the function $y = e^{-\alpha x}$ for any fixed $\alpha \in \mathbb{R}$, we have the following lemma.

Lemma 1. *For any $\alpha \in \mathbb{R}$ and for any $x \in [-1, 1]$, the following inequality holds:*

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

This lemma then implies that

$$\begin{aligned} Z_t &\leq \sum_{i=1}^{N} D_t(i) \left(\frac{1 + y^{(i)} h_t(x^{(i)})}{2} e^{-\alpha_t} + \frac{1 - y^{(i)} h_t(x^{(i)})}{2} e^{\alpha_t} \right). \\ &= \frac{1+r_t}{2} e^{-\alpha_t} + \frac{1-r_t}{2} e^{\alpha_t}, \end{aligned} \tag{10}$$

where

$$r_t = \sum_i D_t(i) y^{(i)} h_t(x^{(i)}). \tag{11}$$

It is easily seen that the right hand side of (10) has its minimum value as a function of α_t if

$$\alpha_t = \frac{1}{2} \log \left(\frac{1+r_t}{1-r_t} \right). \tag{12}$$

Plugging (12) into (10) and using the fact that $\sqrt{1-x} \leq e^{-x/2}$, we have the following corollary:

Corollary 3. *Suppose the generalized version of AdaBoost is utilized and assume $|h_t| \leq 1$. Then with α_t defined by (12) and r_t by (11), we have*

$$Z_t \leq \sqrt{1 - r_t^2}.$$

Therefore the training error of H has the following bound:

$$E_{train}(H) \leq \prod_{t=1}^{T} \sqrt{1 - r_t^2} \leq e^{-\sum_t r_t^2/2}.$$

Remark.

(1). The upper bound of Z_t given above is not as tight as that given for the version in Section 1.1. Say, compare it with (2) in Corollary 1. There are many other ways of improving its bound. But as long as $r_t \neq 0$, the result of the t-th round contributes to the decrement of the (upper bound of) training error.

(2). Note also that there is no requirement that $\alpha_t \geq 0$, i.e. it is not required that $\varepsilon_t \leq 1/2$, the violation of which is the cause for the halt of the loop in the original version of AdaBoost.

(3). In particular, this generalized version of AdaBoost can be used for the binary h_t, meaning the original setup of Binary AdaBoost, with no halt in the middle of the loop.

4 Multi-label classification

4.1 What it is and why

We now present two generalizations of AdaBoost that are most popular. Both of them deal with the case in which the output is not only multi-class but also multi-label. To understand why and how such a situation naturally arises, let us look at the following example.

Business is booming for FunMovies, an online movie rental company. It wants to develop a movie recommendation system. The list of movie genre categories FunMovies deals with consists of: Action, Romance, Comedy, Sci-Fi, Thriller, and War. So the set of outputs (labels) can be defined as

$$\mathfrak{Y} = \{\text{Action, Comedy, Romance, Sci-Fi, Thriller}\}.$$

Each customer has one or more of such categories as his/her favorites. The movie recommendation system's objective is to predict which categories each customer likes. For example, a customer named Jinsu is a 35 year-old male programmer who likes action movies and sci-fi movies. For Jinsu, the input data may look like

$$x^{(i)} = (35, \text{male, programmer}, \cdots),$$

and its label $Y^{(i)}$ is

$$Y^{(i)} = \{\text{Action, Sci-Fi}\}.$$

Another customer, Sujin, who is a 26 year-old female artist, likes romance, comedy and sci-fi movies. The input data for her would look like

$$x^{(j)} = (26, \text{female}, \text{artist}, \cdots),$$

and its label $Y^{(j)}$ is

$$Y^{(j)} = \{\text{Comedy}, \text{Romance}, \text{Sci-Fi}\}.$$

One salient aspect of this setting is that the label $Y^{(i)}$ is not a single value but a set. Then the question arises as to how to measure the accuracy of this movie recommendation system and what the error measure has to be.

In order to address this issue, let us formalize the situation. First, the input (feature) space is as usual denoted by $\mathfrak{X}$. The output (label) space is a finite set $\mathfrak{Y}$. Let $2^{\mathfrak{Y}}$ be the set of subsets of $\mathfrak{Y}$. A classifier then is a map h given by

$$\begin{aligned} h \quad : \quad & \mathfrak{X} \to 2^{\mathfrak{Y}} \\ & x \mapsto h(x) = Y \in 2^{\mathfrak{Y}}. \end{aligned}$$

It should be noted that its value $h(x)$ is a *subset* of $\mathfrak{Y}$. Now let $(x^{(i)}, Y^{(i)})$ be a data point in the training set $\mathfrak{D}$, and let $h(x^{(i)}) \in 2^{\mathfrak{Y}}$ be the value of a classifier h. Figure 13 depicts the relationship between them. In particular, it is reasonable to assert that the error (misclassification) occurs when $h(x^{(i)})$ misses points in $Y^{(i)}$ or $h(x^{(i)})$ contains points that are not in $Y^{(i)}$. On the other hand the points that are simultaneously contained in both $Y^{(i)}$ and $h(x^{(i)})$ and the points that are contained in neither $Y^{(i)}$ nor $h(x^{(i)})$ should be regarded as correctly classified. This is also depicted in Figure 13. For this reason, it is reasonable to define the error of $h(x^{(i)})$ as the size of the symmetric difference between the two sets $h(x^{(i)})$ and $Y^{(i)}$. Namely,

$$e_H(h(x^{(i)}), Y^{(i)}) = |h(x^{(i)}) \triangle Y^{(i)}| = |(h(x^{(i)}) \setminus y^{(i)}) \cup (Y^{(i)} \setminus h(x^{(i)}))|.$$

It is reminiscent of the so-called Hamming distance between two bit vectors. For instance let $x = 1\ 0\ 0\ 1\ 1\ 1\ 0$ and $y = 1\ 1\ 0\ 1\ 0\ 0\ 0$ be bit vectors. The bitwise exclusive OR of x and y is $x \oplus y = 0\ 1\ 0\ 0\ 1\ 1\ 0$. The Hamming distance between x and y is defined to be the number of $1's$ in $x \oplus y$. For this reason, we call $e_H(h(x^{(i)}), Y^{(i)})$ the *Hamming error* of h at $x^{(i)}$.

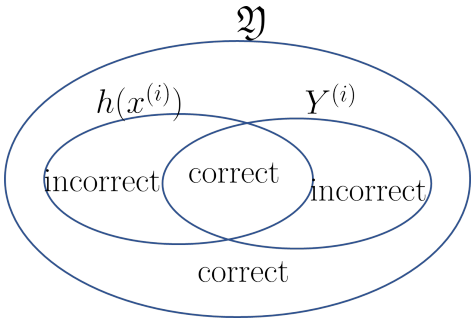

Figure 13: $h(x^{(i)})$ and $Y^{(i)}$

The basic strategy for multi-class, multi-label AdaBoost is to reduce the problem to Binary AdaBoost. To do that, let us introduce the following notation. Now let $\mathfrak{Y} = \{1, 2, \cdots, K\}$. For $Y \in 2^{\mathfrak{Y}}$, define, by abuse of notation, an operator Y by

$$Y[\ell] \quad = \quad \begin{cases} 1 & \text{if } \ell \in Y \\ -1 & \text{if } \ell \notin Y. \end{cases}$$

For $Y^{(i)} \in 2^{\mathfrak{Y}}$ define $y_\ell^{(i)}$ by

$$y_\ell^{(i)} = Y^{(i)}[\ell] \quad = \quad \begin{cases} 1 & \text{if } \ell \in Y^{(i)} \\ -1 & \text{if } \ell \notin Y^{(i)}. \end{cases}$$

The meaning of this notation is that instead of treating $Y^{(i)}$ as a single set, it is broken down into a series of questions like "Is the label ℓ in $Y^{(i)}$?" for every $\ell \in \mathfrak{Y}$. This way, the data set $\mathfrak{D} = \{(x^{(i)}, Y^{(i)})\}_{i=1}^n$ can be regarded as equivalent to the new data set

$$\tilde{\mathfrak{D}} = \{(x^{(i)}, \ell), y_\ell^{(i)}\}_{i=1,\cdots,N, \ell=1,\cdots,K}.$$

It should be noted that this new data set $\tilde{\mathfrak{D}}$ has the merit of being *single-label, binary* data.

For a multi-class, multi-label classifier $h : \mathfrak{X} \to 2^{\mathfrak{Y}}$, define the corresponding single-label, binary classifier $\tilde{h}$ by

$$\begin{array}{rcl} \tilde{h} : \mathfrak{X} \times \mathfrak{Y} & \to & \{-1, 1\} \\ (x, \ell) & \mapsto & \begin{cases} 1 & \text{if } \ell \in h(x) \\ -1 & \text{if } \ell \notin h(x). \end{cases} \end{array}$$

Thus for each data point $(x^{(i)}, Y^{(i)})$, the Hamming error of h at $x^{(i)}$ is easily seen to be

$$e_H(h(x^{(i)}), Y^{(i)}) = |\{\ell : y_\ell^{(i)} \neq \tilde{h}(x^{(i)}, \ell)\}| = |\{\ell : y_\ell^{(i)} \tilde{h}(x^{(i)}, \ell) = -1\}|.$$

We are now ready to present AdaBoost.MH, which is a multi-class, multi-label classification algorithm. (Here, 'H' stands for Hamming.)

AdaBoost.MH
Input:

- Data: $\mathfrak{D} = \{(x^{(i)}, Y^{(i)})\}_{i=1}^N$, where $x^{(i)} \in \mathfrak{X}$ and $Y^{(i)} \in 2^{\mathfrak{Y}}$ for $i = 1, \cdots, N$, where $\mathfrak{Y} = \{1, \cdots, K\}$ is a set of K elements.
- Total number of rounds: T

Initialize: Define the initial probability distribution $D_1(i) = \dfrac{1}{NK}$ for $i = 1, \cdots, N, \ell = 1, \cdots, K$.

Do for $t = 1, 2, \cdots, T$:

- Train using the probability distribution D_t on $\{1, \cdots, N\} \times \{1, \cdots, K\}$.
- Get a hypothesis (classifier) $\tilde{h} : \mathfrak{X} \times \mathfrak{Y} \to \{-1, 1\}$, equivalently $h : \mathfrak{X} \to 2^{\mathfrak{Y}}$
- choose $\alpha_t \in \mathbb{R}$
- update the probability distribution

$$D_{t+1}(i, \ell) = \frac{D_t(i) \exp(-\alpha_t y_\ell^{(i)} \tilde{h}_t(x^{(i)}, \ell))}{Z_t},$$

where $Z_t = \sum_{t,\ell} D_t(i, \ell) \exp(-\alpha_t y_\ell^{(i)} \tilde{h}_t(x^{(i)}, \ell))$
(Note: $e_H(h_t(x^{(i)}), Y^{(i)}) = |\{\ell : y_\ell^{(i)} \tilde{h}_t(x^{(i)}, \ell) = -1\}|$)

Output: the final hypothesis (classifier)

$$H(x, \ell) = \operatorname{sgn}\left(\sum_t \alpha_t \tilde{h}_t(x, \ell)\right).$$

We have seen above how this multi-class, multi-label classification problem can be reduced to a single-label, binary classification problem. Then, invoking the results on the general version of Binary AdaBoost, the below results follow immediately.

Theorem 4. *Assuming $|h_t| \leq 1$, the training error satisfies:*

$$E_{train}(H) \leq \prod_{t=1}^{T} Z_t.$$

Corollary 4.

$$E_{train}(H) \leq \prod_{t} \sqrt{1 - r_t^2} \leq e^{-\frac{1}{2}\sum_t r_t^2},$$

where $r_t = \sum_{i,\ell} D_t(i,\ell) y_\ell^{(i)} \tilde{h}_t(x^{(i)}, \ell)$.

4.2 Multi-label ranking problem

Let us continue our story on FunMovies. Recall that Sujin likes romance, comedy and sci-fi movies. Assume further that she in fact hates those categories not in her favorites . Then any recommendation system should try not to recommend her action or thriller movies, since doing so would be an egregious mistake. AdaBoost.MR (rank boost) is a boosting method that tries to minimizes such errors. With this in mind, let us give the following definition.

Definition 1. *Given $Y \in 2^{\mathfrak{Y}}$, we say an ordered pair $(\ell_0, \ell_1) \in \mathfrak{Y} \times \mathfrak{Y}$ is a critical pair with respect to Y if $\ell_0 \notin Y$ and $\ell_1 \in Y$.*

Thus for instance, the order pair (Action, Romance) is a critical pair for Sujin. In summary, we give the following definition.

Definition 2. *Let $\mathfrak{X}$ be the input (feature) space and let $\mathfrak{Y} = \{1, \cdots, K\}$ be the output (label) space. A function $f : \mathfrak{X} \times \mathfrak{Y} \to \mathbb{R}$ is called a ranking function. Given x, we say ℓ_2 has higher ranking over ℓ_1 with f if $f(x, \ell_1) < f(x, \ell_2)$. We say a critical pair (ℓ_0, ℓ_0) is a critical error given x with respect to f, if $f(x, \ell_0) \geq f(x, \ell_1)$.*

In AdaBoost.MR, the focus is on minimizing such critical errors. Let $\mathfrak{D} = \{(x^{(i)}, Y^{(i)})\}_{i=1}^{n}$ be the usual data set with $x^{(i)} \in \mathfrak{X}, y^{(i)} \in 2^{\mathfrak{Y}}$. Define the initial probability distribution

$$D_1 : \{1, \cdots, N\} \times \mathfrak{Y} \times \mathfrak{Y} \to [0, 1]$$

by setting

$$D_1(i, \ell_0, \ell_1) = \begin{cases} \dfrac{1}{N\,|\mathfrak{Y} - Y^{(i)}|\,|Y^{(i)}|} & \text{if } \ell_0 \notin Y^{(i)} \text{ and } \ell_1 \in Y^{(i)} \\ 0 \quad \text{else.} \end{cases}$$

In other words, $D_1(i, \ell_0, \ell_1) = 0$ if (ℓ_0, ℓ_1) is not critical with respect to $Y^{(i)}$; and for each $x^{(i)}$ an equal probability is given to all critical pairs. It is then easy to see that the critical error rate (with respect to D_1) of the ranking function f can be computed as

$$\sum_{i,\ell_0,\ell_1} D_1(i, \ell_0, \ell_1) \mathbb{I}[f(x^{(i)}, \ell_0) \geq f(x^{(i)}, \ell_1)]. \tag{13}$$

With all these preparatory discussions, we are now ready to present AdaBoost.MR.

AdaBoost.MR
Input:

- Data: $\mathfrak{D} = \{(x^{(i)}, Y^{(i)})\}_{i=1}^{N}$, where $x^{(i)} \in \mathfrak{X}$ and $Y^{(i)} \in 2^{\mathfrak{Y}}$ for $i = 1, \cdots, N$. Here $\mathfrak{Y}$ is a set of K elements usually identified with $\{1, \cdots, K\}$.
- Total number of rounds: T

Initialize: Define the initial probability distribution D_1 by

$$D_1(i, \ell_0, \ell_1) = \begin{cases} \dfrac{1}{N\,|\mathfrak{Y} - Y^{(i)}|\,|Y^{(i)}|} & \text{if } \ell_0 \notin Y^{(i)} \text{ and } \ell_1 \in Y^{(i)} \\ 0 \quad \text{else} \end{cases}$$

Do for $t = 1, 2, \cdots, T$:

- Train using D_t

- Get a hypothesis (ranking function) $h_t : \mathfrak{X} \times \mathfrak{Y} \to \mathbb{R}$
- choose α_t
- update the probability distribution

$$D_{t+1}(i, \ell_0, \ell_1) = \frac{D_t(i, \ell_0, \ell_1) \exp\left\{\frac{1}{2}\alpha_t[h_t(x^{(i)}, \ell_0) - h_t(x^{(i)}, \ell_1)]\right\}}{Z_t},$$

where $Z_t = \sum_{i,\ell_0,\ell_1} D_t(i, \ell_0, \ell_1) \exp\left\{\frac{1}{2}\alpha_t[h_t(x^{(i)}, \ell_0) - h_t(x^{(i)}, \ell_1)]\right\}$.

Output: the final hypothesis

$$f(x, \ell) = \sum_t \alpha_t h_t(x, \ell).$$

Theorem 5. *The training error for AdaBoost.MR satisfies*

$$E_{train}(f) \le \prod_{t=1}^{T} Z_t.$$

Proof. Recall by unraveling the definition of D_t, we get

$$\begin{aligned} D_{T+1}(i) &= \frac{D_1(i, \ell_0, \ell_1) \exp\left\{\frac{1}{2}\sum_t \alpha_t[h_t(x^{(i)}, \ell_0) - h_t(x^{(i)}, \ell_1)]\right\}}{\prod_{t=1}^{T} Z_t} \\ &= \frac{D_1(i, \ell_0, \ell_1) \exp\left\{\frac{1}{2}[f(x^{(i)}, \ell_0) - f(x^{(i)}, \ell_1)]\right\}}{\prod_{t=1}^{T} Z_t}. \end{aligned} \tag{14}$$

By (13), the training error satisfies

$$\begin{aligned} E_{train}(f) &= \sum_{i,\ell_0,\ell_1} D_1(i, \ell_0, \ell_1) \mathbb{I}[f(x^{(i)}, \ell_0) \ge f(x^{(i)}, \ell_1)] \\ &\le \sum_{i,\ell_0,\ell_1} D_1(i, \ell_0, \ell_1) \exp\left\{\frac{1}{2}[f(x^{(i)}, \ell_0) - f(x^{(i)}, \ell_1)]\right\} \\ &= \sum_{i,\ell_0,\ell_1} D_{T+1} \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} Z_t, \end{aligned}$$

where the inequality above is due to the fact that for any $t, \mathbb{I}(t \geq 0) \leq e^{t/2}$ and the last equalities are due to (14) and that D_{T+1} is a probability distribution. □

Assume now $|h_t| \leq 1$ for $t = 1, \cdots, T$. Then using the same argument that leads to (10), we can easily show

$$Z_t \leq \frac{1 - r_t}{2} e^{\alpha_t} + \frac{1 + r_t}{2} e^{-\alpha_t},$$

where $r_t = \frac{1}{2} \sum_{i,\ell_0,\ell_1} D_1(i, \ell_0, \ell_1)[h_t(x^{(i)}, \ell_1) - h_t(x^{(i)}, \ell_0)]$. Since the right hand side of the above equation is minimum when $\alpha_t = \frac{1}{2} \log \left(\frac{1 + r_t}{1 - r_t} \right)$, plugging this back, we have the following Corollary:

Corollary 5. *Assume $|h_t| \leq 1$. Then the training error for AdaBoost.MR satisfies*

$$E_{train}(f) \leq \prod_t \sqrt{1 - r_t^2} \leq e^{-\frac{1}{2} \sum_t r_t^2},$$

where $r_t = \sum_{i,\ell} D_t(i, \ell) y_\ell^{(i)} h_t(x^{(i)}, \ell)$.

5 Generalization error

Let us now discuss the generalization error of AdaBoost. Since the training error decreases as the total number of rounds, T, of training increases, one normally expects overfitting. Figure 14 shows such case.

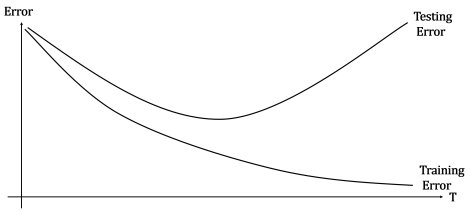

Figure 14: Training and testing errors in usual learning situation

But in many, if not all, of AdaBoost practices it has been reported that such overfitting rarely occurs. Rather, the situation looks more like as in

Figure 15, where the training error stabilizes (not increase much) as T gets very big. Of course, there are AdaBoost practices in which the behavior in

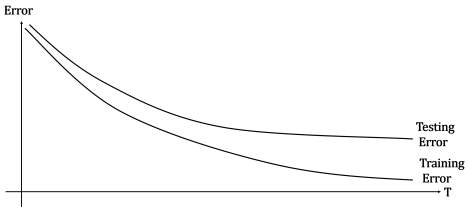

Figure 15: Training and testing errors in typical AdaBoost practice

Figure 14 is observed. In fact, Freund and Schapire have cooked up such example [1]. But in the majority of cases, the behavior in Figure 15 is more prevalent, hence the motto: "AdaBoost almost never overfits." But then the question arises as to "why."

One explanation is through the concept of margin. For example, for Binary AdaBoost $H = \text{sgn}(f(x))$, where $f = \sum_t \alpha_t h_t(x)$. The margin is defined as

$$\text{Margin} = \Big| \frac{\sum_t \alpha_t h_t(x)}{\sum_t \alpha_t} \Big|.$$

In [4], Schapire et al. utilize the Vapnik theory on margin to show that boosting increases the margin of the training set. In a different vein, Breiman observed that the boosting procedure is like a dynamical system and he speculated that it is perhaps ergodic, which means that there is a limit probability distribution to which the dynamics converge. Recently Belanch and Ortiz presented a very interesting argument along this line [7].

On the other hand, it has been known that boosting methods work rather poorly when the input data is noisy. In fact, Long and Servedio show that any convex potential booster suffer from the same problem [6]. It is a serious hindrance to using AdaBoost for noisy inputs. See also the discussion by Schapire in [5]. One is also referred to [1] for more extensive discussion on the generalization error of AdaBoost.

Homework. *Suppose there are nine points on the xy-plane marked with* $+$ *or* $-$ *signs. The points with the* $+$ *sign are* $(0.5, 0.5), (0.5, 1.5), (0.5, 2.5)$ *and* $(1.5, 2.5)$*; the points with the* $-$ *sign are* $(1.5, 0.5), (1.5, 1.5)$*,* $(2.5, 0.5)$*,* $(2.5, 1.5)$ *and* $(2.5, 2.5)$*. They form a training set and are marked in Figure 1.*

Use the Binary AdaBoost algorithm to find a classification boundary that separates the + and − points. You are only to use the weak learners (classifiers) that are of the form:

- *the classifiers with the vertical line $x = 1$ or 2 as the classification boundary;*
- *the classifiers with the horizontal line $y = 1$ or 2 as the classification boundary.*

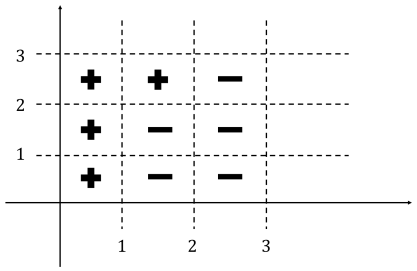

Figure 16: The training set

References

[1] Schapire, R.E. and Freund, Y., *Boosting: Foundations and Algorithms (Adaptive Computation and Machine Learning series)*, The MIT Press (2012)

[2] Schapire, R.E. and Singer, Y., *Improved Boosting Algorithms Using Confidence-rated Predictions. Machine Learning*, 37, 297-336 (1999)

[3] Freund, Y. and Schapire, R.E., *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*, J. of Computer and System Sciences 55, 119-139 (1997)

[4] Schapire, R.E. and Freund, Y., Bartlett, P., Lee, W.S., *Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods*, The Annals of Statistics 26, 1651-1686 (1998)

[5] Schapire, R.E., *Explaining AdaBoost*, in Bernhard Scholkopf, Zhiyuan Luo, Vladimir Vovk, editors, Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik, Springer (2013).

[6] Long, P.M. and Servedio, R.A., *Random Classification Noise Defeats All Convex Potential Boosters*, Machine Learning 78, 287-304 (2010)

[7] Belanch, J. and Ortiz, L.E., *On the Convergence Properties of Optimal AdaBoost*, arXiv:1212.1108v2 [cs.LG] 11Apr2015