Lecture 3: Theoretical Analysis of Boosting Methods

Tianpei Xie

Feb. 6th., 2023

Contents

1 Boosting Algorithm **2**
- 1.1 AdaBoost . . . 2
- 1.2 Functional Gradient Descent . . . 5
- 1.3 Gradient Boost . . . 6

2 Theoretical Guarantee for Boosting **8**
- 2.1 Weak Learner . . . 9
- 2.2 Training Error Bounds . . . 9
- 2.3 Generalization Error Bounds for Finite Hypothesis Class . . . 11
- 2.4 Generalization Error Bounds via VC Dimension . . . 13
- 2.5 Generalization Error Bounds via Margin Theory . . . 14

3 Fundamental Perspectives **20**
- 3.1 Game Theory . . . 20
- 3.2 Online Learning . . . 20
- 3.3 Maximum Entropy Learning . . . 20
- 3.4 Bregman Iterative Projection Algorithms . . . 20

Algorithm 1.1
The boosting algorithm AdaBoost

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t to minimalize the weighted error:

$$\epsilon_t \doteq \mathbf{Pr}_{i \sim D_t}[h_t(x_i) \neq y_i].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

$$= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t},$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Figure 1: AdaBoost Algorithm [Schapire and Freund, 2012]

1 Boosting Algorithm

1.1 AdaBoost

- ***The AdaBoost algorithm*** is the first boosting algorithms proposed by Freund and Schapire [Schapire and Freund, 2012]. Its basic idea is to combine multiple *weak learners* to form a *strong learner*. This strategy is called ***ensemble learning***. Let $h_t \in \mathcal{H}$ be base hypothesis, and α_t be the corresponding weight at iteration $t \in [1, T]$. The *combined learner* after T iteration is

$$H(x) := \text{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

- **Remark** There are several other *ensemble learning algorithms* such that ***bagging algorithms***, which is based on ***boostrap resampling stretegy***. The idea is to *sample with replacement* on the existing dataset to create *duplicated records*. The most frequent seen records in the existing data are more likely be sampled thus this idea is equivalent to ***Monte Carlo sampling*** according to the empirical distribution. For each resampled dataset, we can train a classifier that reflect our understanding on the duplicated sample. Then we combine

multiple classifier to build a single classifier via ***decision fusion***. The typical bagging-based algorithm is ***the random forest algorithm***, which build decision trees for each boostrapped dataset and combine them.

- **Remark** (***Characteristic of AdaBoost***)
  AdaBoost is an *ensemble learning method*. The followings are several key characteristics:

  1. ***AdaBoost*** trains ***multiple weak learners*** in ***sequential manner***. Unlike *bagging methods*, *boosting methods* build weak learners *sequentially*. ***The performance of the previous learners*** *will* ***affect how the new weak-learner is trained***. In the case of AdaBoost, it will be reflected in *the sample weights*, with misclassified samples having increased weight.

     In this way, it is described as a ***functional gradient descent algorithm*** which *instead of computing the gradient*, it *learns* ***a new base hypothesis*** *that* ***resembles the gradient of functional***.

  2. ***The performance measure*** in each step of ***AdaBoost*** is ***the training error rate*** of new base learner relative to ***a weighted sample distribution***; i.e.

     $$\epsilon_t := \widehat{\mathbb{E}}_{\mathcal{D}_t}\left[h_t(X) \neq Y\right] = \sum_{i=1}^{m} \mathcal{D}_t(i) \mathbb{1}\left\{h_t(X_i)\right\}$$

     The training error rate under $\mathcal{D}_t$ has two roles:

     (a) it deterime *the* ***weight*** α_t *for the base hypothesis* h_t. In AdaBoost,

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

     It is clear that when the classifier h_t is no more than a random guess with $\epsilon_t = 1/2$, the corresponding weight $\alpha_t = 0$. In other situation, the *smaller* the ϵ_t the *larger* the α_t. Note that $\alpha_t > 0$ ***if and only if*** $\epsilon_t < 1/2$ meaning that the learned hypothesis h_t is a *weak-learner*.

     (b) it determines ***the multiplicative factor*** *in the exponential reweight strategy for each sample*. In particular, the factor is $\exp\left(-\alpha_t y_i h_t(x_i)\right)$.

  3. ***AdaBoost*** *apply an* ***exponential reweighting strategy*** *at each iteration*. In particular,

     $$\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

     where $Z_t := \sum_{i=1}^{m} \mathcal{D}_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)$ is the partition function that normalized the sample distribution $\mathcal{D}_{t+1}$. The exponential reweighting strategy ***shrinks*** the weight by $e^{-\alpha_t} < 1$ when *the sample is correctly labeled*, while ***enlarges*** the weight by $e^{\alpha_t} > 1$ when *the sample is* ***incorrectly labeled***. This means that *misclassified samples will have* ***higher weights*** in next iteration.

  4. *AdaBoost* promote the idea of ***adversarial learning*** that is shown great success in deep learning models such as *Generative Adversarial Network (GAN)*. The key idea comes from the ***two player zero-sum game***. Unlike GAN, AdaBoost did not continually

re-train the same hypothesis but instead move on to build a new hypothesis for a few misclassified samples.

The change of sample distribution $\mathcal{D}_t \to \mathcal{D}_{t+1}$ helps the AdaBoost to ***boost the impact*** *of* ***adversarial samples*** for the *new* learner so that it will ***overemphasize on the past mistakes***. The consequence is that ***the later learned*** *base hypothesis* is ***more specialized*** *on a few* ***difficult samples*** while ***the early learned*** *base hypothesis* is *more generalized for a majority of easy samples.*

In the end, when *the error rate* of new classifier *decreased*, the corresponding *hypothesis weight* α_t will *decrease* and the multiplicative factor will *tends to* 1, which leades to the sample distribution ***converge to some stationary distribution***.

$$h_t \to \epsilon_t \downarrow \qquad \Rightarrow e^{\alpha_t} \to 1 \qquad \Rightarrow \mathcal{D}_t \approx \mathcal{D}_{t+1}$$

5. *One of main reason behind the* ***popularity*** *of boosting* is *its high* ***computational efficiency***. Boosting methods are highly ***scalable*** for large dataset with high dimensions. *AdaBoost* provides ***performance guarantee*** *both* ***theorectically*** *and* ***practically*** when the base learner is implemented with simple learning algorithm such as ***decision trees***. Specifically, when using decision stumps (1-layer decision tree), the time complexity of each round of boosting is in $\mathcal{O}(mn)$ where n is the feature dimension and m is the sample size.

- **Remark** *AdaBoost* is a well-studied algorithm and it provides ***theorectical guarantee*** based on *statistical learning theory.* This chapter focus on various aspects of theorectial guarantee of AdaBoost algorithm and its ***connections*** to other algorithms. In particular, we focus on following aspects:

  1. We show that ***the training error*** of AdaBoost ***will converge to zero*** as iteration T increases, even if each single hypothesis only slightly better than random guess with error rate $\epsilon_t = \frac{1}{2} - \gamma_t$.

  2. We develop ***the generalization error bound*** for AdaBoost using ***VC dimension*** *of base hypothesis class* $\mathcal{H}$. This allows us to provides ***Probably Approximately Correct (PAC) learnablity*** guarantee for given hypothesis class $\mathcal{H}$ and it helps to *quantitatively* describe the ***sample complexity*** of the algorithm. VC dimension theory also helps us to build intuition on the ***tradeoff*** between lower training error and overfitting as the number of iterations increases (i.e. ***the Bias-Complexity tradeoff***).

  3. The VC dimension theory is not sufficient to explain the performance of AdaBoost, esp. when the generalization error continues to improve even if the training error is zero. An alterinative theory in statistical learning is called ***the large margin theory***. In particular, it associated the performance of classifier with the ***margin*** between *the decision boundary and samples.* The idea is that for binary classification, a good classifier not only make correct decision but also make decision that is ***robust to small perturbation*** of samples. Leaving a margin between decision boundary and samples allows the classifier to ***avoid making mistakes with highly confidence***. The idea of learning with maximal margin motivates the development of *support vector machines (SVM).* Boosting and SVM do share some similarities here. However, ***boosting is not directly optimizing the margin***. Although in practice and theory, it is observed that the learned hypothesis from AdaBoost do have a large margin, it can also be shown that

AdaBoost's success cannot be ***fully explained by large margin*** either.

4. One important aspect for AdaBoost is its connection to ***game theory*** via its ***adversarial training style***. *The min-max theorem* helps us to understand that ***the weak learnablity assumption*** implies *a* ***strong assumption*** *that* ***the dataset is linearly separable with a margin***.

5. Other algorithms have connections with AdaBoost include

   (a) ***online learning*** [Cesa-Bianchi and Lugosi, 2006], esp. when ***the exponential reweighting strategy*** is used; and

   (b) ***Bregman iterative projections*** [Peyr and Cuturi, 2019], a generic algorithms that at each iteration projects to a subspace that is closer to the target by minimizing the ***Bregman divergence***.

   (c) The way when the sample distribution is optimized is also close to ***maximum entropy learning*** [Cover and Thomas, 2006].

1.2 Functional Gradient Descent

- The boosting models are summarized as a ***stage-wise additive model*** [Hastie et al., 2009],

$$F_M(x) := \sum_{t=1}^{M} \alpha_t h_t(x).$$

- The learning algorithm, at each iteration t, choose a base hypothesis $h_t \in \mathcal{H}$ and its weight $\alpha_t \in \mathbb{R}$ that *minimizes* the loss, given *the additive model in previous iterations*:

$$\min_{h_t \in \mathcal{H}, \alpha_t \in \mathbb{R}} \sum_{i=1}^{m} L(y_i, F_{t-1}(x_i) + \alpha_t h_t(x_i))$$

  After (h_t, α_t) is selected, it *merges* with existing additive model $F_t(x) = F_{t-1}(x) + \alpha_t h_t$.

- Given sample $\mathcal{D}$, we treat $h_t \in \mathcal{H}$ on $\mathcal{D}$ as a vector $h_{\mathcal{D}} := (h_t(x_1), \ldots, h_t(x_m))$. Then we can find the gradient of loss function with respect to the vector $h_{\mathcal{D}}$ evaluated at F_{t-1}

$$\nabla L_{\mathcal{D}}(F_{t-1}) := \left[\frac{\partial L(y, h)}{\partial h_t(x_i)} \Big|_{h=F_{t-1}} \right]_{i=1,\ldots,m} \tag{1}$$

- This step is close to the classical *gradient descent algorithm* for *(unconstrained) optimization.* ***The major difference*** is that instead of computing *the numerical gradient* ∇L, we choose to ***learn a new base hypothesis*** $h_t \in \mathcal{H}$ that ***matches*** the ***negative functional gradient***

$$h_t := \arg\min_{h \in \mathcal{H}} Diss(h_{\mathcal{D}}, -\nabla L_{\mathcal{D}}(F_{t-1})), \tag{2}$$

  where $Diss$ is some ***distance/dissimilarity measure*** between two functions on given data set $\mathcal{D}$. Thus the new base hypothesis h_t plays role of $-\nabla L(F_{t-1})$ and it is then merging with existing function F_{t-1} to *make corrections*.

Algorithm 7.3
AnyBoost, a generic functional gradient descent algorithm

Goal: minimization of $\mathcal{L}(F)$.
Initialize: $F_0 \equiv 0$.
For $t = 1, \ldots, T$:

- Select $h_t \in \mathcal{H}$ that maximizes $-\nabla\mathcal{L}(F_{t-1}) \cdot h_t$.
- Choose $\alpha_t > 0$.
- Update: $F_t = F_{t-1} + \alpha_t h_t$.

Output F_T.

Figure 2: A generic boosting algorithm based on functional gradient descent [Hastie et al., 2009]

- It can be shown that for AdaBoost, the loss functional is the exponential loss:

$$L(F) \equiv L(y, F) := \exp\left(-yF(x)\right) \tag{3}$$

and the functional gradient

$$\nabla L_{\mathcal{D}}(F_{t-1}) := \left[\frac{-y_i \exp\left(-y_i F_{t-1}(x_i)\right)}{m}\right]_{i=1,\ldots,m}. \tag{4}$$

Choose similarity measure as *the cosine similarity*, which means that the goal is to optimize

$$\begin{aligned}\max_{h\in\mathcal{H}} S(h_{\mathcal{D}}, -\nabla L_{\mathcal{D}}(F_{t-1})) &= \max_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} y_i h_t(x_i) \exp\left(-y_i F_{t-1}(x_i)\right) \\ &= \max_{h\in\mathcal{H}} \sum_{i=1}^{m} D_t(i) y_i h_t(x_i) := 1 - 2\epsilon_t\end{aligned}$$

So the goal of the maximizing the cosine similarity is equivalent to minimizing the error rate ϵ_t on weighted sample distribution $\mathcal{D}_t$. The hypothesis weight α_t is chosen to minimize the exponential loss (3).

1.3 Gradient Boost

- The ***gradient boosting methods*** uses the negative functional gradient on samples $\mathcal{D}$ as

$$-\nabla L_{\mathcal{D}}(F_{t-1}) := -\left[\frac{\partial L(y,h)}{\partial h_t(x_i)}\Big|_{h=F_{t-1}}\right]_{i=1,\ldots,m} \equiv [r_{i,t}]_{i=1,\ldots,m} \tag{5}$$

- At each iteration, it learns a base hypothesis that minimize *the mean squared error loss*. In other word, it treats *the negative functional gradient vector* $[r_{i,t}]_{i=1}^{m}$ as *the* ***residual*** *and perform* ***regression tasks iteratively***. That is,

$$\min_{h_t\in\mathcal{H}} \sum_{i=1}^{m} \left(r_{i,t} - h_t(x_i)\right)^2 = \min_{h_t\in\mathcal{H}} \sum_{i=1}^{m} \|-\nabla L_{\mathcal{D}}(F_{t-1}) - h_t\|_2^2 \tag{6}$$

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute
   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$
   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$.
   (c) For $j = 1, 2, \ldots, J_m$ compute
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$
   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.

Figure 3: Gradient Boost Tree Algorithm [Hastie et al., 2009]

Finally, choose α_t that minimize the loss function L:

$$\alpha_t := \sum_{i=1}^{m} r_{i,t} \frac{h_t(x_i)}{\|h_t\|_2^2}$$

- Equivalently, ***the negative functional gradient*** becomes ***the pseudo-label*** for the new hypothesis during the learning.
- Compared to *AdaBoost*, ***the Gradient Boost*** has several differences:
  1. Gradient boost generalized the AdaBoost by *choosing a* ***general loss function*** in the learning task, which could be more efficient and more ***flexible*** for some tasks.
  2. Unlike AdaBoost, ***no sample reweighting is needed for Gradient Boost*** since the role of error rate ϵ_t and sample distribution $\mathcal{D}_t$ is fulfilled by *the functional gradient* $\nabla L_{\mathcal{D}}(F_{t-1})$. In particular, ***correctly labeled samples*** have ***smaller functional gradients*** while ***misclassified*** *samples have* ***larger functional gradients***.

     In Gradient Boosting, '*shortcomings*' (of existing weak learners) are identified by ***gradients***. In *AdaBoost*, '*shortcomings*' are identified by *high-weight data points.*
- It tends to say that the main differences are that *Gradient Boosting is a* ***generic algorithm*** *to* ***find approximate solutions*** *to* ***the additive modeling problem***, while AdaBoost can be seen as a special case with a particular loss function. Hence, *Gradient Boosting is* ***much more flexible***.

  However, as pointed by Freund and Schapire [Schapire and Freund, 2012], ***optimizing the exponential loss alone cannot explain the performance of AdaBoost***. It is likely due to ***large margin property*** and ***the adversarial training procedure*** that the AdaBoost outperforms its counterpart in optimization only approach. It is critical to take into account *the particular* ***dynamics*** *of the algorithm*, not just the objective function.

2 Theoretical Guarantee for Boosting

- **Remark** (***Data***)
  Define an ***observation*** as a d-dimensional vector x. The *unknown* nature of the observation is called a ***class***, denoted as y. The domain of observation is called an ***input space*** *or* ***feature space***, denoted as $\mathcal{X} \subset \mathbb{R}^d$, whereas the domain of class is called the ***target space***, denoted as $\mathcal{Y}$. For ***classification task***, $\mathcal{Y} = \{1, \ldots, M\}$; and for ***regression task***, $\mathcal{Y} = \mathbb{R}$. A ***concept*** $c : \mathcal{X} \to \mathcal{Y}$ is the *input-output association* from the nature and is *to be learned* by ***a learning algorithm***. Denote $\mathcal{C}$ as *the set of all concepts* we wish to learn as the ***concept class***. The learner is requested to output a *prediction rule*, $h : \mathcal{X} \to \mathcal{Y}$. This function is also called a ***predictor***, a ***hypothesis***, or a ***classifier***. The predictor can be used to predict the label of new domain points. Denote a collection of n ***samples*** as

$$\mathcal{D} \equiv \mathcal{D}_n = ((X_1, Y_1), \ldots, (X_n, Y_n)) \equiv ((X_1, c(X_1)), \ldots, (X_n, c(X_n))).$$

  Note that $\mathcal{D}_n$ is a finite ***sub-sequence*** in $(\mathcal{X} \times \mathcal{Y})^n$.

- **Definition** (***Generalization Error in Deterministic Scenario***) [Mohri et al., 2018]
  Under *a deterministic scenario*, ***generalization error*** or the ***risk*** or simply ***error*** for the *classifier* $h \in \mathcal{H}$ is defined as

$$L(h) \equiv L_{\mathcal{P},c}(h) = \mathcal{P}\{h(X) \neq c(X)\} \equiv \mathbb{E}_X[\mathbb{1}\{h(X) \neq c(X)\}] \tag{7}$$

  with respect to the concept $c \in \mathcal{C}$ and *the feature distribution* $\mathcal{P} \equiv \mathcal{P}_X$.

- **Definition** (***Empirical Error or Training Error***)
  Given the data $\mathcal{D}$, the ***training error*** or the ***empirical error/risk*** of a hypothesis $h \in \mathcal{H}$ is defined as

$$\widehat{L}(h) \equiv \widehat{L}_{\mathcal{D}}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{h(X_i) \neq Y_i\} = \frac{1}{n}|\{i : h(X_i) \neq Y_i\}| := \widehat{\mathbb{E}}[\mathbb{1}\{h(X) \neq Y\}]$$

  where either $Y = c(X)$ or Y is a random variable associated with X.

- **Definition** (***The Realizable Assumption***)
  There exists $h^* \in \mathcal{H}$ s.t. $L_{\mathcal{P},c}(h^*) = 0$.

- **Definition** (***PAC Learnability***)
  A hypothesis class $\mathcal{H}$ is ***PAC learnable*** if there exist a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $\epsilon, \delta \in (0,1)$, *for every distribution* $\mathcal{P}$ *over* $\mathcal{X}$, and *for every labeling function* $c : \mathcal{X} \to \{0,1\}$, if *the realizable assumption* holds with respect to $\mathcal{H}$, $\mathcal{P}$, c, then when running the learning algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. examples generated by $\mathcal{P}$ and labeled by c, the algorithm returns a hypothesis h such that, with probability of at least $1 - \delta$ (over the choice of the examples),

$$L_{\mathcal{P},c}(h) \leq \epsilon.$$

2.1 Weak Learner

- **Definition** (γ-***Weak Learnability***) [Schapire and Freund, 2012, Shalev-Shwartz and Ben-David, 2014]

A learning algorithm, $\mathcal{A}$, is a ***γ-weak-learner*** for a class $\mathcal{H}$ if there exists a function $m_{\mathcal{H}} : (0,1) \to \mathbb{N}$ such that for ***every*** $\delta \in (0,1)$, ***for every distribution*** $\mathcal{P}$ *over* $\mathcal{X}$, and ***for every labeling function*** $c : \mathcal{X} \to \{-1, +1\}$, if *the realizable assumption holds* with respect to $\mathcal{H}$, $\mathcal{P}$, c, then when running the learning algorithm on $m \geq m_{\mathcal{H}}(\delta)$ i.i.d. examples generated by $\mathcal{P}$ and labeled by c, the algorithm returns a hypothesis h such that, with probability of at least $1 - \delta$,

$$L_{\mathcal{P},c}(h) \leq \frac{1}{2} - \gamma.$$

A hypothesis class $\mathcal{H}$ is ***γ-weak-learnable*** if there exists a γ-*weak-learner* for that class.

- **Remark** We call PAC learnalbe ***the strong learnable***.
- **Remark** (***Weak Learner Without Accuracy Guarantee***)
  Unlike *the PAC learner*, who guarantees that *with high probability* the generalization error rate is less than ϵ ***for all*** ϵ, a ***γ-weak-learner*** guarantees that with high probability, the error rate is less than ϵ ***for some*** $\epsilon = 1/2 - \gamma$, i.e. *less than half with a margin* γ.

  In other word, *under the realizablity assumption*, it is expected that ***with more data***, ***a PAC learner*** can learn the "*true*" labeling function behind the data, (i.e. ***zero generalization error*** with high probability). While a ***γ-weak-learner*** can only get ***slightly better than random guess*** and it is ***not*** *expected to have* ***lower error rate*** even if more data are available.
- **Remark** (***Weak Learner is as Hard as PAC Learner***)
  The fundamental theorem of learning states that if a hypothesis class $\mathcal{H}$ has a VC dimension d, then *the sample complexity of PAC learning* $\mathcal{H}$ satisfies $m_{\mathcal{H}}(\epsilon, \delta) \geq C_1(d + \log(1/\delta))/\epsilon$, where C_1 is a constant. Applying this with $\epsilon = 1/2 - \gamma$ we immediately obtain that ***if*** $d = \infty$ ***then*** $\mathcal{H}$ ***is not*** ***γ-weak-learnable***.

  This implies that from ***the statistical perspective*** (i.e., if we ignore *computational complexity*), ***weak learnability*** *is also characterized by the VC dimension of* $\mathcal{H}$ *and therefore is just* ***as hard as PAC (strong) learning***. However, when we do consider ***computational complexity***, the potential advantage of weak learning is that maybe there is *an algorithm that satisfies the requirements of weak learning and* ***can be implemented efficiently***.

2.2 Training Error Bounds

- **Remark** Recall that $h_t \in \mathcal{H}$ are base learners for $t \in [1, T]$, and $(\alpha_1, \ldots, \alpha_T) \in \Sigma_T$. The *combined learner* is

$$H(x) := \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

- The space of all such combined classifiers is defined as below:

  Definition (***Ensemble Hypothsis Class***)
  Define the class of T ***linear combinations of base hypotheses*** from $\mathcal{H}$ as

$$L(\mathcal{H}, T) := \left\{ \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(\cdot)\right) : \alpha \in \mathbb{R}^T, h_t \in \mathcal{H}, t = 1, \ldots, T \right\} \tag{8}$$

- **Definition** (***Linear Threshold Functions***)
  Define Σ_n as the space of all ***linear threshold functions***

$$\Sigma_n := \{\operatorname{sgn}(\langle w\,,\,x\rangle) : w \in \mathbb{R}^n\}\,.$$

  Thus $L(\mathcal{H}, T) = \{\sigma(h_1(x), \ldots, h_T(x)) : \sigma \in \Sigma_T\}$

- **Proposition 2.1** *(**Training Error Bound for AdaBoost**) [Schapire and Freund, 2012] Given the notation of Adaboost algorithm, let $\gamma_t = 1/2 - \epsilon_t$, and let $\mathcal{D}_1$ be an arbitrary initial distribution over the training set. Then **the weighted training error** of the combined classifier $\mathcal{H}$ with respect to $\mathcal{D}_1$ is bounded as*

$$\widehat{L}_{\mathcal{D}_1}(H) \le \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \le \exp\left(-2\sum_{t=1}^{T} \gamma_t^2\right). \tag{9}$$

  Proof: Consider the linear combination of base classifiers:

$$f(x) := \sum_{t=1}^{T} \alpha_t h_t(x).$$

  By definition of distribution $\mathcal{D}_T$ via $\mathcal{D}_{T-1}$ we have

$$\begin{aligned}
\mathcal{D}_{T+1}(i) &= \frac{\exp(-\alpha_T y_i h_T(x_i))}{Z_T}\mathcal{D}_{T-1}(i) \\
&= \ldots \\
&= \frac{\exp\left(-\sum_{t=1}^{T} \alpha_t y_i h_t(x_i)\right)}{\prod_{t=1}^{T} Z_t}\mathcal{D}_1(i) = \frac{\exp(-y_i f(x_i))}{\prod_{t=1}^{T} Z_t}\mathcal{D}_1(i)
\end{aligned}$$

  Note that $H(x) \neq y$ if and only if $yf(x) < 0$, and $\mathbb{1}_{(-\infty,0]}(x) \le \exp(-x)$ so

$$\mathbb{1}\{H(x_i) \neq y_i\} = \mathbb{1}_{(-\infty,0]}(y_i f(x_i)) \le \exp(-y_i f(x_i))\,.$$

  By definition of training error,

$$\begin{aligned}
\widehat{L}_{\mathcal{D}_1}(H) &= \sum_{i=1}^{m} \mathbb{1}\{H(x_i) \neq y_i\}\,\mathcal{D}_1(i) \\
&\le \sum_{i=1}^{m} \exp(-y_i f(x_i))\,\mathcal{D}_1(i) \\
&= \left(\sum_{i=1}^{m} \mathcal{D}_{T+1}(i)\right)\prod_{t=1}^{T} Z_t \\
&= \prod_{t=1}^{T} Z_t
\end{aligned} \tag{10}$$

Finally, by our choice of $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t} = \frac{1}{2}\log\frac{1/2+\gamma_t}{1/2-\gamma_t}$, we have that

$$\begin{aligned}
Z_t &= \sum_{i=1}^{m} \mathcal{D}_t(i)\exp\left(-\alpha_t y_i h_t(x_i)\right) \\
&= \exp\left(-\alpha_t\right) \sum_{i:H(x_i)=y_i} \mathcal{D}_t(i) + \exp\left(\alpha_t\right) \sum_{i:H(x_i)\neq y_i} \mathcal{D}_t(i) \\
&= \exp\left(-\alpha_t\right)\left(1-\epsilon_t\right) + \exp\left(\alpha_t\right)\epsilon_t \qquad \text{(by definition of error } \epsilon_t\text{)} \\
&= \exp\left(-\alpha_t\right)\left(1/2+\gamma_t\right) + \exp\left(\alpha_t\right)\left(1/2-\gamma_t\right) \qquad \text{(substitute } \alpha_t\text{)} \\
&= \sqrt{\left(1-2\gamma_t\right)\left(1+2\gamma_t\right)} \qquad (11)
\end{aligned}$$

Substituting (11) into (10), we have the result. ∎

2.3 Generalization Error Bounds for Finite Hypothesis Class

- **Definition** (***Restriction of $\mathcal{H}$ to $\mathcal{D}$***).
  Let $\mathcal{H}$ be a class of functions from $\mathcal{X}$ to $\{0,1\}$ and let $\mathcal{D} = \{x_1, \ldots, x_m\} \subset \mathcal{X}$.

  The restriction of $\mathcal{H}$ to $\mathcal{D}$ is *the set of functions* from $\mathcal{D}$ to $\{0,1\}$ that can be *derived from* $\mathcal{H}$. That is,

  $$\mathcal{H}_{\mathcal{D}} := \{(h(x_1), \ldots, h(x_m)) : h \in \mathcal{H}\},$$

  where we ***represent*** each function from $\mathcal{X}$ to $\{0,1\}$ as a ***vector*** in $\{0,1\}^{|\mathcal{D}|}$.

- **Definition** (***Shattering***).
  A hypothesis class $\mathcal{H}$ ***shatters*** a finite set $\mathcal{D} \subset \mathcal{X}$ if ***the restriction of $\mathcal{H}$ to $\mathcal{D}$*** is the set of ***all functions*** from $\mathcal{D}$ to $\{0,1\}$. That is,

  $$|\mathcal{H}_{\mathcal{D}}| = 2^{|\mathcal{D}|}.$$

- **Definition** (***Growth Function***).
  Let $\mathcal{H}$ be a hypothesis class. Then ***the growth function of $\mathcal{H}$***, denoted $\tau_{\mathcal{H}} : \mathbb{N} \to \mathcal{N}$, is defined as

  $$\tau_{\mathcal{H}}(m) := \max_{\mathcal{D}\subset\mathcal{X}:|\mathcal{D}|=m} |\mathcal{H}_{\mathcal{D}}|.$$

  In words, $\tau_{\mathcal{H}}(m)$ is ***the number of different functions*** from a set $\mathcal{D}$ of ***size*** m to $\{0,1\}$ that can be obtained by ***restricting $\mathcal{H}$ to $\mathcal{D}$***.

- **Lemma 2.2** ***(Sauer's Lemma)***. *[Shalev-Shwartz and Ben-David, 2014, Mohri et al., 2018] Let $\mathcal{H}$ be a hypothesis class with $VCdim(\mathcal{H}) \leq d < \infty$. Then, for all $m \geq d+1$,*

  $$\tau_{\mathcal{H}}(m) \leq \sum_{i=0}^{d} \binom{m}{i} \leq \left(\frac{em}{d}\right)^d. \qquad (12)$$

- **Proposition 2.3** ***(Generalization Bound via Growth Function)*** *[Mohri et al., 2018] Let $\mathcal{H}$ be a family of functions taking values in $\{-1, +1\}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, for any $h \in \mathcal{H}$,*

$$L(h) \leq \widehat{L}_m(h) + \sqrt{\frac{2 \log \tau_{\mathcal{H}}(m)}{m}} + \sqrt{\frac{\log(1/\delta)}{2m}} \tag{13}$$

Growth function bounds can be also derived directly (without using Rademacher complexity bounds first). The resulting bound is then the following:

$$\mathcal{P}\left\{\exists h \in \mathcal{H}, \left|L(h) - \widehat{L}_m(h)\right| > \epsilon\right\} \leq 4\tau_{\mathcal{H}}(2m)\ \exp\left(-\frac{m\epsilon^2}{8}\right) \tag{14}$$

which only differs from (13) *by constants.*

- The following lemma shows that the VC dimension of Σ_T is T.

  Lemma 2.4 *[Schapire and Freund, 2012, Mohri et al., 2018]*
  The space Σ_n of ***linear threshold functions*** *over $\mathbb{R}^n$*

$$\Sigma_n := \{sgn\left(\langle w\,,\, x\rangle\right) : w \in \mathbb{R}^n\}$$

  has ***VC-dimension*** *n.*

- **Remark** Note that the class Σ_n is ***the class of half-spaces*** $\{\langle w\,,\, x\rangle : w \in \mathbb{R}^n\}$ whose VC dimension is n.

- **Lemma 2.5** ***(Growth Number of Ensemble Hypothesis Class, Finite Hypothesis Class)*** *[Schapire and Freund, 2012, Shalev-Shwartz and Ben-David, 2014]*
  Assume $\mathcal{H}$ is ***finite****. Let $m \geq T \geq 1$. For any set $\mathcal{D}$ of m points, the number of dichotomies realizable by $L(\mathcal{H}, T)$ is bounded as follows:*

$$|L(\mathcal{H}, T)_{\mathcal{D}}| \leq \tau_{L(\mathcal{H},T)}(m) \leq \left(\frac{em}{T}\right)^T |\mathcal{H}|^T\,. \tag{15}$$

  Proof: Consider a new sample $\mathcal{D}' := \{Z_1\,, \ldots, Z_m\}$ where

$$Z_i := \left(h_1(X_i)\,, \ldots, h_T(X_i)\right).$$

  Given that the VC dimension of Σ_T is T, the bound on the growth number becomes

$$|L(\mathcal{H}, T)_{\mathcal{D}'}| \leq \left(\frac{em}{T}\right)^T. \tag{16}$$

  That is, for *fixed* $h_1\,, \ldots, h_T$, *the number of dichotomies* defined by functions of the form $\sigma(h_1(x)\,, \ldots, h_T(x))$ for $\sigma \in T$ is bounded as in equation (16). Since the number of choices for $h_1\,, \ldots, h_T$ is equal to $|\mathcal{H}|^T$, and since for each one of these, the number of dichotomies is as in equation (16), we thus obtain the bound stated in the lemma. ∎

- **Theorem 2.6** ***(Generalization Bound for AdaBoost, Finite Hypothesis)*** *[Schapire and Freund, 2012]*
  Suppose ***AdaBoost*** *is run for T rounds on $m \geq T$ random examples, using base classifiers from a* ***finite space*** *$\mathcal{H}$. Then, with probability at least $1-\delta$, the combined classifier H satisfies*

$$L_{\mathcal{P},c}(H) \leq \widehat{L}_m(H) + \sqrt{\frac{2T\left(\log|\mathcal{H}| + \log(em/T)\right)}{m}} + \sqrt{\frac{\log(1/\delta)}{2m}} \tag{17}$$

Furthermore, with probability at least $1-\delta$*, if* $\mathcal{H}$ *is realizable with the training set (i.e.* $\widehat{L}_m(h) \equiv 0$*), then*

$$L_{\mathcal{P},c}(H) \leq \frac{2T\left(\log|\mathcal{H}| + \log(2em/T)\right) + 2\log(2/\delta)}{m}. \tag{18}$$

2.4 Generalization Error Bounds via VC Dimension

- **Lemma 2.7** *(**Growth Number of Ensemble Hypothesis Class, VC Class**). [Schapire and Freund, 2012]*
  Assume $\mathcal{H}$ *has* ***finite VC-dimension*** $d \geq 1$*. Let* $m \geq \max\{T, d\} \geq 1$*. For any set* $\mathcal{D}$ *of* m *points, the number of dichotomies realizable by* $L(\mathcal{H}, T)$ *is bounded as follows:*

$$|L(\mathcal{H},T)_{\mathcal{D}}| \leq \tau_{L(\mathcal{H},T)}(m) \leq \left(\frac{em}{T}\right)^T \left(\frac{em}{d}\right)^{dT}. \tag{19}$$

- **Lemma 2.8** *(**VC-Dimension of Ensemble Hypothesis Class, VC Class**). [Schapire and Freund, 2012, Shalev-Shwartz and Ben-David, 2014]*
  Assume $\mathcal{H}$ *has* ***finite VC-dimension*** $\nu(\mathcal{H}) = d$ *and* $\min\{T, d\} \geq 3$*. Then the VC dimension of combined hypothesis class is bounded by*

$$\nu(L(\mathcal{H},T)) \leq T(d+1)\left(3\log\left(T(d+1)\right)+2\right) = \mathcal{O}(Td\log(Td)). \tag{20}$$

- **Remark** (***Lower Bound on VC Dimension***). [Shalev-Shwartz and Ben-David, 2014]
  For some base hypothesis class $\mathcal{H}$, the VC-dimension of ensemble is at least Td. For instance, for $\mathcal{H}_n$ be the class of *decision stumps* over $\mathbb{R}^n$, we can show that $\log(n) \leq d = \nu(\mathcal{H}) \leq 2\log(n) + 5$. In this example, for all $T \geq 1$,

$$\nu(L(\mathcal{H}_n,T)) \geq 0.5T\log(n) \asymp \Omega(Td).$$

- **Theorem 2.9** *(**Generalization Bound for AdaBoost via VC Dimension**). [Schapire and Freund, 2012]*
  Suppose ***AdaBoost*** *is run for* T *rounds on* $m \geq \max\{T, d\}$ *random examples, using base classifiers from a* ***finite space*** $\mathcal{H}$*. Then, with probability at least* $1-\delta$*, the combined classifier* H *satisfies*

$$L_{\mathcal{P},c}(H) \leq \widehat{L}_m(H) + \sqrt{\frac{2T\left(d\log(em/d) + \log(em/T)\right)}{m}} + \sqrt{\frac{\log(1/\delta)}{2m}} \tag{21}$$

  Furthermore, with probability at least $1-\delta$*, if* $\mathcal{H}$ *is realizable with the training set (i.e.* $\widehat{L}_m(h) \equiv 0, \forall h \in \mathcal{H}$*), then*

$$L_{\mathcal{P},c}(H) \leq \frac{2T\left(d\log(2em/d) + \log(2em/T)\right) + 2\log(2/\delta)}{m}. \tag{22}$$

- **Remark** (***Limit of VC Dimension Analysis***)
  The upper bound grows as $\mathcal{O}(dT\log(dT))$, thus the bound suggests that ***AdaBoost*** *could* ***overfit*** *for* ***large values of*** T, and indeed this can occur. See Figure 4. However, in many cases, it has been observed empirically that *the generalization error of AdaBoost* ***decreases*** as a function of *the number of rounds* of boosting T.

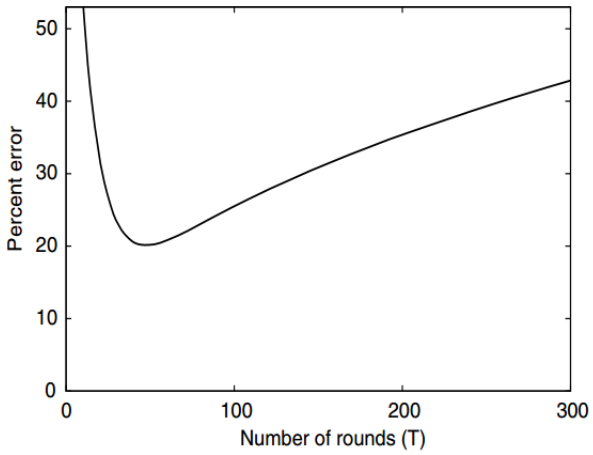

Figure 4.1
A plot of the bound on the generalization error given in equation (4.6) as a function of the number of rounds T, using the constants from theorem 4.3 with $\gamma = 0.2$, $m = 10^6$, $\ln |\mathcal{H}| = 10$, and $\delta = 0.05$.

Figure 4: AdaBoost may overfit if the number of rounds T is too large. [Schapire and Freund, 2012]

- **Corollary 2.10** *[Schapire and Freund, 2012]*
  Assume, in addition to the assumptions of theorem 2.9, that each base classifier has weighted error $\epsilon_t \leq 1/2 - \gamma$ for some $\gamma > 0$. Let the number of rounds T be equal to

$$\inf \left\{ T \in \mathbb{N} : T \geq \frac{\log(m)}{2\gamma^2} \right\}$$

  Then, with probability at least $1 - \delta$, the generalization error of the combined classifier H will be at most

$$\mathcal{O}\left(\frac{1}{m}\left[\frac{\log(m)}{\gamma^2}\left(\log(m) + d\log\left(\frac{m}{d}\right)\right) + \log\left(\frac{1}{\delta}\right)\right]\right)$$

- **Remark** Ignoring the log factor, the generalization error bound (21) can be summarized as

$$L_{\mathcal{P},c}(H) \leq \widehat{L}_m(H) + \mathcal{O}\left(\sqrt{\frac{T\mathcal{C}_{\mathcal{H}}}{m}}\right)$$

  where $\mathcal{C}_{\mathcal{H}}$ is some complexity measure of base class $\mathcal{H}$.

- **Theorem 2.11** *(**Strong Learnable** $\Leftrightarrow$ **Weak Learnable**) [Schapire and Freund, 2012]*
  *A target class $\mathcal{H}$ is (efficiently) **weakly** PAC learnable **if and only if** it is (efficiently) **strongly** PAC learnable.*

2.5 Generalization Error Bounds via Margin Theory

- **Definition** (L_1-***Margin***) [Mohri et al., 2018, Schapire and Freund, 2012]
  The L_1***-margin*** $\rho(x)$ of a point $x \in \mathcal{X}$ with label $y \in \{-1, +1\}$ for *a linear combination of base classifiers* $g = \sum_{t=1}^{T} \alpha_t h_t = \langle \alpha\,,\, h \rangle$ with $\alpha \neq 0$ and $h_t \in \mathcal{H}$ for all $t \in [1, T]$ is defined as

$$\rho(x) := y\frac{\langle \alpha\,,\, h(x) \rangle}{\|\alpha\|_1} = \frac{\sum_{t=1}^{T} \alpha_t y h_t(x)}{\|\alpha\|_1} \tag{23}$$

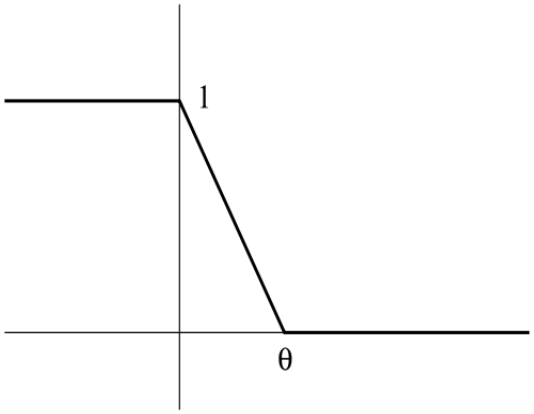

Figure 5.3
A plot of the piecewise-linear function ϕ given in equation (5.27).

Figure 5: The piecewise linear function φ_ϕ. [Schapire and Freund, 2012]

The L_1***-margin*** of a linear combination classifier g ***with respect to a sample*** $\mathcal{D}$ is ***the minimum margin*** *of the points within the sample*:

$$\rho := \min_{i=1,\dots,m} y_i \frac{\langle \alpha\,,\, h(x_i)\rangle}{\|\alpha\|_1} = \min_{i=1,\dots,m} \frac{\sum_{t=1}^{T} \alpha_t y_i h_t(x_i)}{\|\alpha\|_1} \tag{24}$$

- **Definition** (***Margin Loss Function***) [Mohri et al., 2018, Schapire and Freund, 2012]
  For any $\rho > 0$, ***the*** ρ***-margin loss*** $L_\rho : \mathbb{R}\times\mathbb{R} \to \mathbb{R}_+$ is defined for all $y, y' \in \mathbb{R}$ by $L_\rho(y, y') = \varphi_\rho(yy')$ where φ is defined as a piecewise-linear function,

$$\varphi_\rho(x) := \begin{cases} 1 & \text{if } x \le 0 \\ 1 - x/\rho & \text{if } 0 \le x \le \rho \\ 0 & \text{if } x \ge \rho \end{cases}$$

  This function is ***Lipschitz*** with $L_\varphi = 1/\rho$.

- **Definition** (***Empirical Margin Loss***) [Schapire and Freund, 2012, Mohri et al., 2018]
  Given a sample $\mathcal{D}_m$ and a hypothesis h, ***the empirical margin loss*** is defined by

$$\widehat{L}_{m,\rho}(h) = \frac{1}{m}\sum_{i=1}^{m} \varphi_\rho\left(Y_i h(X_i)\right) \tag{25}$$

  Note that for any $i \in [1, m]$, $\mathbb{1}\{y_i h(x_i) \le 0\} \le \varphi_\rho(y_i h(x_i)) \le \mathbb{1}\{y_i h(x_i) \le \rho\}$. Thus, the empirical margin loss can be bounded as follows:

$$\begin{aligned} \widehat{L}(h) &= \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\{h(X_i) \ne Y_i\} = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\{Y_i h(X_i) \le 0\} \\ &\le \widehat{L}_{m,\rho}(h) \le \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\{Y_i h(X_i) \le \rho\}\,. \end{aligned} \tag{26}$$

- **Remark** In all the results that follow, the empirical margin loss can be replaced by this upper bound, which admits a simple interpretation: it is *the fraction of the points in the training sample* $\mathcal{D}$ *that have been* ***misclassified*** *or classified* ***with confidence less than*** ρ.

- **Remark** When the coefficients α_t are ***non-negative***, as in the case of *AdaBoost*, $\rho(x)$ is ***a convex combination*** of the base classifier values $h_t(x)$. In particular, if the base classifiers h_t take values in $[-1, +1]$, then $\rho(x)$ is in $[-1, +1]$. *The absolute value* $|\rho(x)|$ can be interpreted as ***the confidence*** *of the classifier* g in that label.

- **Definition** (***Convex Hull of Hypothesis Class***)
  For any hypothesis class $\mathcal{H}$, ***the convex hull*** of set $\mathcal{H}$, denoted as $\text{conv}(\mathcal{H})$, is defined as

$$\text{conv}(\mathcal{H}) := \left\{ \sum_{k=1}^{T} \lambda_k h_k(\cdot) : T \geq 1, \forall k \in [1, T], \lambda_k \geq 0, h_k \in \mathcal{H}, \sum_{k=1}^{T} \lambda_k = 1 \right\}.$$

- **Remark** Let $\mathcal{H}$ be our space of base classifiers, and let $\mathcal{M}$ be the space of all "***margin functions***" of the form $yf(x)$ where f is any convex combination of base classifiers:

$$\mathcal{M} := \{(x, y) \to yf(x) : f \in \text{conv}(\mathcal{H})\}$$

  Note that $\widehat{\mathfrak{R}}_{\mathcal{D}}(\mathcal{M}) = \widehat{\mathfrak{R}}_{\mathcal{D}}(\text{conv}(\mathcal{H}))$ since $y_i\sigma_i$ has the same distribution as σ_i.

- **Definition** (***Empirical Rademacher Complexity***)
  Let $\mathcal{G}$ be a family of functions mapping from $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$ to $[a, b]$ and $\mathcal{D} = (z_1, \ldots, z_n)$ a fixed *sample* of size n with elements in $\mathcal{Z}$. Then, ***the empirical Rademacher complexity*** of $\mathcal{G}$ *with respect to the sample* $\mathcal{D}$ is defined as:

$$\widehat{\mathfrak{R}}_{\mathcal{D}}(\mathcal{G}) = \mathbb{E}_\sigma \left[\sup_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i g(z_i) \right] \tag{27}$$

  where $\sigma := (\sigma_1, \ldots, \sigma_n)$ are ***independent uniform random variables*** taking values in $\{-1, +1\}$. The random variables σ_i are called ***Rademacher variables***.

- **Proposition 2.12** *(**Empirical Rademacher Complexity of a Convex Hull of Function Class***
  Let $\mathcal{H}$ be a set of functions mapping from $\mathcal{X}$ to $\mathbb{R}$. Then, for any sample $\mathcal{D}$, the empirical Rademacher complexity

$$\widehat{\mathfrak{R}}_{\mathcal{D}}(conv(\mathcal{H})) = \widehat{\mathfrak{R}}_{\mathcal{D}}(\mathcal{H}) \tag{28}$$

  where conv($\mathcal{H}$) is ***the convex hull*** *of set $\mathcal{H}$.*

- **Theorem 2.13** *(**Uniform Bound via Rademacher Complexity**) [Mohri et al., 2018]*
  Let $\mathcal{G}$ be a family of functions mapping from $\mathcal{Z}$ to $[0, 1]$. Then, for any $\delta > 0$, ***with probability at least*** *$1 - \delta$, each of the following holds for all $g \in \mathcal{G}$:*

$$\mathbb{E}\left[g(Z)\right] \leq \frac{1}{m} \sum_{i=1}^{m} g(Z_i) + 2\mathfrak{R}_m(\mathcal{G}) + \sqrt{\frac{\log(1/\delta)}{2m}} \tag{29}$$

  and

$$\mathbb{E}\left[g(Z)\right] \leq \frac{1}{m} \sum_{i=1}^{m} g(Z_i) + 2\widehat{\mathfrak{R}}_m(\mathcal{G}) + 3\sqrt{\frac{\log(2/\delta)}{2m}} \tag{30}$$

- Based on the theorem above, we can have the generalization error bound via margin:

Theorem 2.14 *(**Ensemble Rademacher Margin Bound**) [Schapire and Freund, 2012, Mohri et al., 2018]*
Let $\mathcal{H}$ denote a set of real-valued functions. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, each of the following holds for all $h \in conv(\mathcal{H})$:

$$L(h) \leq \widehat{L}_{m,\rho}(h) + \frac{2}{\rho}\mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log(1/\delta)}{2m}} \tag{31}$$

$$L(h) \leq \widehat{L}_{m,\rho}(h) + \frac{2}{\rho}\widehat{\mathfrak{R}}_m(\mathcal{H}) + 3\sqrt{\frac{\log(2/\delta)}{2m}} \tag{32}$$

Proof: Consider the family of functions taking values in $[0, 1]$:

$$\varphi_\rho \circ \mathcal{M} := \{\varphi_\rho \circ f : f \in \mathcal{M}\}$$

where $\mathcal{M} := \{(x, y) \to yh(x) : h \in \text{conv}(\mathcal{H})\}$. By the generalization bound via Rademacher complexity,

$$\mathbb{E}\left[\varphi_\rho(Yh(X))\right] \leq \frac{1}{m}\sum_{i=1}^{m}\varphi_\rho\left(Y_i h(X_i)\right) + 2\mathfrak{R}_m(\varphi_\rho \circ \mathcal{M}) + \sqrt{\frac{\log(1/\delta)}{2m}}$$

By inequality (26)

$$L(h) = \mathbb{E}\left[\mathbb{1}\left\{Y \neq h(X)\right\}\right] \leq \mathbb{E}\left[\varphi_\rho(Yh(X))\right]$$

thus

$$L(h) \leq \widehat{L}_{m,\rho}(h) + 2\mathfrak{R}_m(\varphi_\rho \circ \mathcal{M}) + \sqrt{\frac{\log(1/\delta)}{2m}}$$

Note that φ_ρ is $(\frac{1}{\rho})$-Lipschitz function.

$$\begin{aligned}
\mathfrak{R}_m(\varphi_\rho \circ \mathcal{M}) &\leq \frac{1}{\rho}\mathfrak{R}_m(\mathcal{M}) && (\text{ by contraction principle}) \\
&= \frac{1}{\rho}\mathfrak{R}_m(\text{conv}(\mathcal{H})) && (\text{ since } y_i \text{ is absorbed by } \sigma_i) \\
&= \frac{1}{\rho}\mathfrak{R}_m(\mathcal{H}). && (\text{ by (28)})
\end{aligned}$$

This complete the proof. ∎

- **Theorem 2.15** *(**Ensemble VC-Dimension Margin Bound**) [Schapire and Freund, 2012, Mohri et al., 2018]*
Let $\mathcal{H}$ be a family of functions taking values in $\{+1, -1\}$ with VC-dimension d. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for all $h \in conv(\mathcal{H})$:

$$L(h) \leq \widehat{L}_{m,\rho}(h) + \frac{2}{\rho}\sqrt{\frac{2d\log(em/d)}{m}} + \sqrt{\frac{\log(1/\delta)}{2m}} \tag{33}$$

- **Remark** Note that from the point of view of binary classification, g and $g/\left\|\alpha\right\|_1$ are equivalent since $\text{sgn}\left(g\right) = \text{sgn}\left(g/\left\|\alpha\right\|_1\right)$, thus $L(g) = L(g/\left\|\alpha\right\|_1)$, but their empirical margin loss are

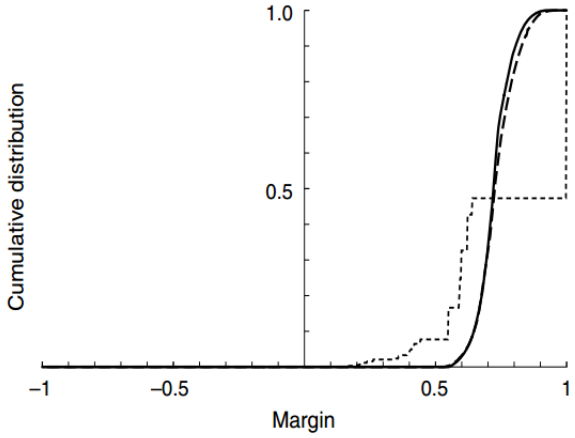

Figure 5.2
The margin distribution graph for boosting C4.5 on the letter dataset showing the cumulative distribution of margins of the training instances after 5, 100, and 1000 iterations, indicated by short-dashed, long-dashed (mostly hidden), and solid curves, respectively. (Reprinted with permission of the Institute of Mathematical Statistics.)

Figure 6: Distribution of margin for AdaBoost on one example dataset. [Schapire and Freund, 2012]

distinct. Let $g = \sum_{t=1}^{T} \alpha_t h_t$ denote the function defining the classifier returned by AdaBoost after T rounds of boosting when trained on sample $\mathcal{D}$. Then, in view of (31), for any $\delta > 0$, with probability at least $1 - \delta$

$$L(g) \leq \widehat{L}_{m,\rho}(g/\|\alpha\|_1) + \frac{2}{\rho}\mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log(1/\delta)}{2m}} \tag{34}$$

- **Remark** (***Generalization Guarantee by Large Margin Only***)
  Remarkably, ***the number of rounds of boosting*** T ***does not appear*** *in the generalization bound* (34). The bound depends only on *the margin* ρ*, the sample size* m*, and the Rademacher complexity of the family of base classifiers* $\mathcal{H}$. Thus, the bound guarantees an effective generalization if the margin loss $\widehat{L}_{m,\rho}(g/\|\alpha\|_1)$ is small for a relatively large ρ.

- **Proposition 2.16** ***(Empirical Margin Loss Bound for AdaBoost)*** *[Schapire and Freund, 2012, Mohri et al., 2018]*
  Let $f = \sum_{t=1}^{T} \alpha_t h_t$ *denote the function defining the classifier returned by AdaBoost after* T *rounds of boosting and assume for all* $t \in [1, T]$ *that* $\epsilon_t < 1/2$*, which implies* $\alpha_t > 0$*. Then, for any* $\rho > 0$*, the following holds:*

$$\widehat{L}_{m,\rho}(f) \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t^{1-\rho}(1-\epsilon_t)^{1+\rho}} \leq \left[(1-2\gamma)^{1-\rho}(1+2\gamma)^{1+\rho}\right]^{T/2} \tag{35}$$

where $\gamma \leq 1/2 - \epsilon_t$ *for all* t*. Note that* $\gamma > 0$ *so* $\left[(1-2\gamma)^{1-\rho}(1+2\gamma)^{1+\rho}\right] < 1$.

Proof: Consider the linear combination of base classifiers:

$$f(x) := \sum_{t=1}^{T} \alpha_t h_t(x).$$

Note that $yf(x) \leq \rho$ if and only if

$$y \sum_{t=1}^{T} \alpha_t h_t(x) \leq \rho \sum_{t=1}^{T} \alpha_t.$$

This implies that

$$\exp\left(-y \sum_{t=1}^{T} \alpha_t h_t(x) + \rho \sum_{t=1}^{T} \alpha_t\right) \geq 1 \geq \mathbb{1}\left\{yf(x) \leq \rho\right\}$$

Thus

$$\begin{aligned}\widehat{L}_{m,\rho}(f) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\left\{y_i f(x_i) \leq \rho\right\} &\leq \exp\left(\rho \sum_{t=1}^{T} \alpha_t\right) \left[\frac{1}{m} \sum_{i=1}^{m} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)\right)\right] \\ &= \exp\left(\rho \sum_{t=1}^{T} \alpha_t\right) \left(\prod_{t=1}^{T} Z_t\right) \quad \text{(See proof in Proposition 2.1)}\end{aligned}$$

Plugging in the values of $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$ and $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$ and the derivation follows the same as in Proposition 2.1, which gives the final result. ∎

- **Remark** This bound implies that the fraction of training examples with $yf(x) \leq \rho$ decreases to zero ***exponentially fast*** with T, and must actually be equal to zero at some point since this fraction must always be a multiple of $1/m$.

- **Remark** (***AdaBoost Maximize the Margin?***)
  The margin bounds combined with the bound on the empirical margin loss suggest that under some conditions, ***AdaBoost*** *can achieve* ***a large margin on the training sample***. They could also serve as a theoretical explanation of the empirical observation that *in some tasks* ***the generalization error decreases*** *as a function of* T *even after the error on the training sample is zero*: *the margin would continue to increase.*

  But does *AdaBoost* maximize the L_1-margin? ***No***. It has been shown that *AdaBoost may* ***converge to a margin*** *that is* ***significantly smaller than the maximum margin***. However, under some general assumptions, when the data is *separable* and *the base learners satisfy particular conditions*, it has been proven that *AdaBoost* can ***asymptotically*** *achieve* ***a margin*** *that is* ***at least half the maximum margin***, $\rho_{max}/2$.

- **Remark** (***Limit for Margin Theory***)
  We can directly maximize the L_1-margin by solving a *Linear Programming (LP) problem.* By definition, the solution of the LP just described admits an L_1-margin that is larger or equal to that of the *AdaBoost* solution. However, empirical results do not show a systematic benefit for the solution of the LP. In fact, it appears that in many cases, *AdaBoost outperforms that algorithm.* ***The margin theory*** described ***does not seem sufficient*** to explain that performance.

3 Fundamental Perspectives

3.1 Game Theory

3.2 Online Learning

3.3 Maximum Entropy Learning

3.4 Bregman Iterative Projection Algorithms

References

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006. ISBN 978-0-471-24195-9.

Trevor Hastie, Robert Tibshirani, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

Gabriel Peyr and Marco Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019. ISSN 1935-8237.

Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012. ISBN 0262017180.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.