Machine learning theory

Boosting

Hamid Beigy

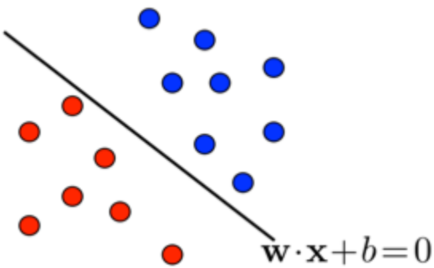
Sharif university of technology

April 2, 2023

Table of contents

Linear classifier

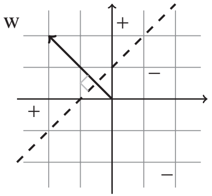
1. The family of linear classifiers is one of the most useful families of hypothesis classes.
2. Many learning algorithms that are being widely used in practice rely on linear predictors because of
   - the ability to learn them efficiently in many cases,
   - linear predictors are intuitive,
   - are easy to interpret, and
   - fit the data reasonably well in many natural learning problems.
3. A linear classifier separates different classes by a linear separator.



1. **Training data:** sample drawn iid from set $\mathcal{X} \subseteq \mathbb{R}^n$ according to some distribution $\mathcal{D}$.

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \in \mathcal{X} \times \{-1, +1\}.$$

2. **Problem:** find hypothesis $h : \mathcal{X} \mapsto \{-1, +1\}$ in H_n with small generalization error $\mathbf{R}(h)$.
3. **Hypothesis space:** $H_n = \{x \mapsto \text{sgn}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) \mid w \in \mathbb{R}^n, b \in \mathbb{R}\}$.
4. A linear classifier is defined as $h(\mathbf{x}) = \text{sgn}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right)$.
5. Vector $\mathbf{w}$ is orthogonal to the separator.



6. We shown that $VC(H_n) = n + 1$.
7. We can learn this space using ERM paradigm, as long as the sample size is $\dfrac{(n+1) + \log(1/\delta)}{\epsilon}$.
8. Implementing the ERM rule in the nonseparable case is known to be computationally hard.

1. Linear programs are problems that can be expressed as maximizing a linear function subject to linear inequalities. That is

$$\max_{\mathbf{w}\in\mathbb{R}^n} \langle \mathbf{u}, \mathbf{w} \rangle$$

$$\text{subject to } \mathbf{A}\mathbf{w} \geq \mathbf{v}.$$

   where
   - $\mathbf{w} \in \mathbb{R}^n$ is the vector of variables we wish to determine.
   - $\mathbf{A}$ is an $m \times n$ matrix.
   - $\mathbf{u} \in \mathbb{R}^n$ and $\mathbf{v} \in \mathbb{R}^n$ are vectors.
2. Linear programs can be solved efficiently.

1. Suppose that the training data is linearly separable.
2. We are interested to find $\mathbf{w}$ and b that results in zero training error.
3. Let $\mathbf{w} = (b, w_1, w_2, \ldots, w_n)$ and $\mathbf{x} = (1, x_1, \ldots, x_n)$.
4. Hence, we are looking for $\mathbf{w} \in \mathbb{R}^{n+1}$ such that for all i

$$sign(\langle \mathbf{w}, \mathbf{x}_i \rangle) = y_i$$

5. Equivalently, we are looking for $\mathbf{w} \in \mathbb{R}^{n+1}$ such that for all i

$$y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 0.$$

6. Let $\mathbf{w}^*$ be a vector that satisfies this condition.
7. Define $\gamma = \min_i (y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle)$ and let $\bar{\mathbf{w}} = \frac{\mathbf{w}^*}{\gamma}$. Therefore, for all i we have

$$y_i \langle \bar{\mathbf{w}}, \mathbf{x}_i \rangle = \frac{1}{\gamma} y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle \geq 1.$$

8. We have thus shown that there exists a vector that for all i satisfies

$$y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 1.$$

1. We have thus shown that there exists a vector that for all i satisfies

$$y_i\langle \mathbf{w}, \mathbf{x}_i\rangle > 1.$$

2. To find a vector that satisfies the above inequality,
   - Set $\mathbf{A}$ to be $m \times (n+1)$ matrix whose rows are the instances multiplied by y_i: $\mathbf{A}_{ij} = y_i \times x_{ij}$.
   - Set $\mathbf{v}$ to be $(1, 1, \ldots, 1) \in \mathbb{R}^{n+1}$.
3. Then the above inequality becomes

$$\mathbf{A}\mathbf{w} > \mathbf{v}.$$

4. The LP form requires a maximization objective, thus, we set a dummy objective, $\mathbf{u} = (0, \ldots, 0) \in \mathbb{R}^{n+1}$.
5. There are other algorithm for finding the linear classifier such as Perceptron.

Ensemble Learning

1. Ensemble methods are general techniques in machine learning for combining several predictors to create a more accurate one.
2. Two main categories of ensemble learning:
   - Boosting
   - Bagging
3. In the problem of PAC-learnability, we were trying to find learning algorithms that learned the problem really well (to within some ϵ error rate).
4. This is a strong guarantee, a strong learner is a classifier that is arbitrarily well-correlated with the true classification.

1. Two major issues in machine learning are
   - Have a good tradeoff between approximation error and estimation error.

$$\mathbf{R}(h) - \mathbf{R}^* = \underbrace{\left(\mathbf{R}(h) - \inf_{h' \in H} \mathbf{R}(h)\right)}_{\text{Estimation error}} + \underbrace{\left(\inf_{h' \in H} \mathbf{R}(h) - \mathbf{R}^*\right)}_{\text{Approximation error}}$$

   - Computational complexity of learning.
2. How do we achieve a good tradeoff between approximation error and estimation error.
   - The error of an ERM learner can be decomposed into a sum of approximation error and estimation error.
   - The more expressive the hypothesis class the learner is searching over, the smaller the approximation error is, but the larger the estimation error becomes.
   - A learner is faced with the problem of picking a good tradeoff between these two considerations.
3. Computational complexity of learning.
   - For many interesting concept classes the task of finding an ERM hypothesis may be computationally infeasible.

1. The idea behind boosting is to construct a strong learner by combining many weak learners.
2. A weak learner is defined to be a classifier that it can label examples better than random guessing.
3. Boosting is based on the question posed by Kearns and Valiant (1988, 1989):
   Can a set of weak learners create a single strong learner?
4. **Robert Schapire** answered the question of Kearns and Valiant in 1990 by introducing **Boosting algorithm**.
5. **Freund and Schapire** introduced AdaBoost algorithm in 1997.

1. **Breiman** introduced Bagging algorithm in 1994.
2. The boosting paradigm allows the learner to have smooth control over tradeoff between estimation and approximation errors.
3. The learning starts with a basic class (that might have a large approximation error), and as it progresses the class that the predictor may belong to grows richer.
4. AdaBoost enables us to control the tradeoff between the approximation and estimation errors by varying a single parameter.
5. Family of Boosting algorithms reduce variance and bias.
6. When a weak learner can be implemented efficiently, boosting provides a tool for aggregating such weak hypotheses.
7. Bagging algorithm reduces variance and helps to avoid overfitting.

1. Recall the definition of (strong) PAC learning:

Definition (Strong PAC learnability)

A concept class $\mathcal{C}$ is strongly PAC learnable using a hypothesis class H if there exists an algorithm A such that for any $c \in \mathcal{C}$, for any distribution $\mathcal{D}$ over the input space, for any $\epsilon \in (0, \frac{1}{2})$ and $\delta \in (0, \frac{1}{2})$, given access to a polynomial (in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$) number of examples drawn i.i.d. from $\mathcal{D}$ and labeled by c, A outputs a function $h \in H$ such that with probability at least $(1 - \delta)$, we have $\mathbf{R}(h) \leq \epsilon$.

2. This definition is strong in the sense that it requires that $\mathbf{R}(h)$ can be driven arbitrarily close to 0 by choosing an appropriately small value of ϵ.

1. But what happens if we can't get the error arbitrarily close to 0? Is learning all or none?
2. To answer these questions, we introduce the notion of weak PAC learning.

Definition (Weakly PAC learnability)

A concept class $\mathcal{C}$ is weakly PAC learnable using a hypothesis class H if there exists an algorithm A and a value of $\gamma > 0$ such that for any $c \in \mathcal{C}$, for any distribution $\mathcal{D}$ over the input space, for any $\delta \in (0, \frac{1}{2})$, given access to a polynomial (in $\frac{1}{\delta}$) number of examples drawn i.i.d. from $\mathcal{D}$ and labeled by c, A outputs a function $h \in H$ such that with probability at least $(1 - \delta)$, we have $\mathbf{R}(h) \leq \frac{1}{2} - \gamma$.

3. We will sometimes refer to γ as the advantage of A (over random guessing).
4. Weak learnability only requires A to return a hypothesis that does better than random guessing.

1. It's clear that strong learnability implies weak learnability.
   - Strong learnability implies the ability to find an arbitrarily good classifier with error rate at most $\epsilon > 0$.
   - In weak learnability, we only need to output a hypothesis whose error rate is at most $\left(\frac{1}{2} - \gamma\right)$.
   - The hope is that it may be easier to learn efficient weak learners than efficient strong learners.
2. The question we want to answer is whether weak learnability implies strong learnability.
   - From fundamental theorem, if $VC(H) = d$ then $m_H(\epsilon, \delta) \geq C_1 \dfrac{d + \log(1/\delta)}{\epsilon}$.
   - By setting $\epsilon = \left(\frac{1}{2} - \gamma\right)$, $d = \infty$ implies that H is not weakly learnable.
   - From the statistical perspective (ignoring computational complexity), weak learnability is characterized by $VC(H)$ and therefore is just as hard as strong learnability.
   - Computational complexity is the advantage of weak learning: the weak learning can be implemented efficiently.

The following theorem shows the learnability of weak learners.

Theorem (Weak learnability)

A class of hypothesis H is weakly learnable iff it has finite VC dimension.

Proof.

1. Finite VC $\Rightarrow$ PAC learnability $\Rightarrow$ Weak learnability
2. Weak learnability $\Rightarrow m_H(\frac{1}{2}-\gamma,\delta) \geq C_1 \dfrac{VC(H)+\log(1/\delta)}{\frac{1}{2}-\gamma}$ is finite $\Rightarrow$ Finite VC

$\square$

More formally, we might ask:

If $\mathcal{C}$ is weakly learnable using H, must there exist some H' such that $\mathcal{C}$ is (strongly) learnable using H'?

1. More formally, we might ask the following:
   If $\mathcal{C}$ is weakly learnable using H, must there exist some H' such that $\mathcal{C}$ is (strongly) learnable using H'?
2. We can think about this question as follows.
   - Fix an arbitrary $\epsilon > 0$.
   - Suppose we are given a polynomial number (in $1/\delta$ and $1/\epsilon$) of samples drawn i.i.d. from some distribution $\mathcal{D}$ and labeled by a target $c \in \mathcal{C}$, as well as a weak learning algorithm A for $\mathcal{C}$.
   - Can we incorporate A into a new algorithm that is guaranteed to produce a new function h such that with high probability, $\mathbf{R}(h) < \epsilon$?

1. A natural question to ask is whether strong and weak PAC learning algorithms are equivalent.
2. Moreover, if this is true,
   we would like to have an algorithm to convert a weak PAC learning algorithm into a strong PAC learning algorithm.
3. Boosting is an algorithm that can do the above task and defined as follows.

Definition (Boosting algorithm)

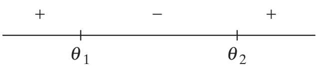
A boosting algorithm is an algorithm that converts a weak learning algorithm into a strong learning algorithm.

Example (Learning the class of 3-partitions of $\mathbb{R}$)

1. Let $H_{3p} = \left\{ h^b_{\theta_1,\theta_2} \;\middle|\; \theta_1, \theta_2 \in \mathbb{R}, b \in \{-1, +1\} \right\}$ be class of 3-partitions of $\mathbb{R}$ as

$$h^b_{\theta_1,\theta_2}(x) = \begin{cases} +b & \text{if } x < \theta_1 \\ -b & \text{if } \theta_1 \le x \le \theta_2 \\ +b & \text{if } x > \theta_2 \end{cases}$$

2. An example hypothesis is



3. By setting $\gamma = \frac{1}{6}$, we show that H_{3p} is weakly learnable by ERM over Decision Stumps (class of threshold functions) $\mathcal{B} = \{x \mapsto \operatorname{sgn}(x - \theta) \times b \mid \theta \in \mathbb{R}, b \in \{-1, +1\}\}$.
4. For every distribution $\mathcal{D}$ over $\mathbb{R}$ consistent with H_{3p}, there exists a threshold function h such that

$$\hat{\mathbf{R}}(h) \le \frac{1}{2} - \frac{1}{6} = \frac{1}{3}.$$

Example (Learning the class of 3-partitions of $\mathbb{R}$)

1. We know that $VC(\mathcal{B}) = 2$, if sample size is greater than $\Omega\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, then with probability of at least $(1-\delta)$, the $ERM_{\mathcal{B}}$ rule returns a hypothesis with an error of at most $\frac{1}{3} + \epsilon$.
2. Setting $\epsilon = \frac{1}{12}$, with probability of at least $(1-\delta)$, we have $\mathbf{R}(ERM_{\mathcal{B}}(S)) \leq \hat{\mathbf{R}}(ERM_{\mathcal{B}}(S)) + \epsilon = \frac{1}{3} + \frac{1}{12}$.
3. We see that $ERM_{\mathcal{B}}$ is a weak learner for H.

It is important to note that both strong and weak PAC learning are distribution-free.

The following example will shed more light on the importance of this.

Example (Learning with a fixed distribution)

1. Let $\mathcal{C}$ be the set of all concepts over $\{0,1\}^n \cup \{\mathbf{z}\}$, where $\mathbf{z} \notin \{0,1\}^n$.
2. Let $\mathcal{D}$ be the distribution that assigns mass $\frac{1}{4}$ to $\mathbf{z}$ and mass $\frac{3}{4}$ uniformly distributed over $\{0,1\}^n$.

$$\underset{\mathbf{x}\sim\mathcal{D}}{\mathbb{P}}[\mathbf{x}=\mathbf{k}] = \begin{cases} \frac{1}{4} & \text{if } (\mathbf{k}=\mathbf{z}) \\ \frac{3}{4} \times \frac{1}{2^n} & \text{if } \mathbf{k} \in \{0,1\}^n \end{cases}$$

3. Consider the hypothesis h that predicts

$$h(\mathbf{x}) = \begin{cases} c(\mathbf{z}) & \text{if } (\mathbf{x}=\mathbf{z}) \\ 0 & \text{with prob. of } \frac{1}{2} \text{ if } \mathbf{x} \neq \mathbf{z} \\ 1 & \text{with prob. of } \frac{1}{2} \text{ if } \mathbf{x} \neq \mathbf{z} \end{cases}$$

Example (Learning with a fixed distribution)

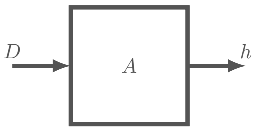
1. Consider the hypothesis h that predicts

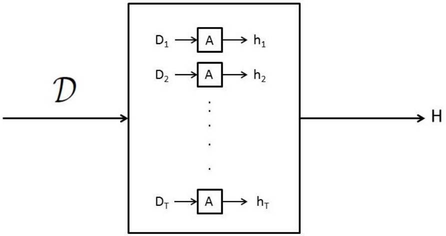
$$h(\mathbf{x}) = \begin{cases} c(\mathbf{z}) & \text{if } (\mathbf{x} = \mathbf{z}) \\ 0 & \text{with prob. of } \frac{1}{2} \text{ if } \mathbf{x} \neq \mathbf{z} \\ 1 & \text{with prob. of } \frac{1}{2} \text{ if } \mathbf{x} \neq \mathbf{z} \end{cases}$$

2. This hypothesis always correctly predict label of $\mathbf{z}$ and predict label of $\mathbf{x} \neq \mathbf{z}$ with 50% accuracy.
3. The error of this hypothesis equals to $\mathbf{R}(h) = \frac{3}{4} \times \frac{1}{2} = \frac{3}{8} < \frac{1}{2}$.
4. If we drop the distribution-freeness from the definition, $\mathcal{C}$ is weakly PAC learnable for the fixed distribution $\mathcal{D}$.
5. However, $VC(\mathcal{C}) = 2^n$, hence $\mathcal{C}$ is not strongly PAC learnable (by modifying the definition to a fixed distribution) using any algorithm.
6. This is because we would need at least $m = \Omega(2^n)$ examples, which is not polynomial.
7. Hence we cannot necessarily convert a weak into a strong learning algorithm if we fix the distribution.

Adaptive Boosting

- We are given
  - Training set $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$ drawn from distribution $\mathcal{D}$, where $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$.
  - A weak learner A which for all D (not necessarily the same as $\mathcal{D}$), given $S \sim \mathcal{D}^m$ finds a $h \in \mathcal{B}$ such that $\mathbb{P}\left[\mathbf{R}(h) \leq \frac{1}{2} - \gamma\right] \geq 1 - \delta$.
  - The goal is to find a final hypothesis $h \in H$ such that $\mathbb{P}\left[\mathbf{R}(h) \leq \epsilon\right] \geq 1 - \delta$.
- The main idea behind AdaBoost is to run the weak learning algorithm several times and combine the hypotheses from each run.
- To do this effectively, we need to force the weak algorithm to learn by giving it a different D on every run.





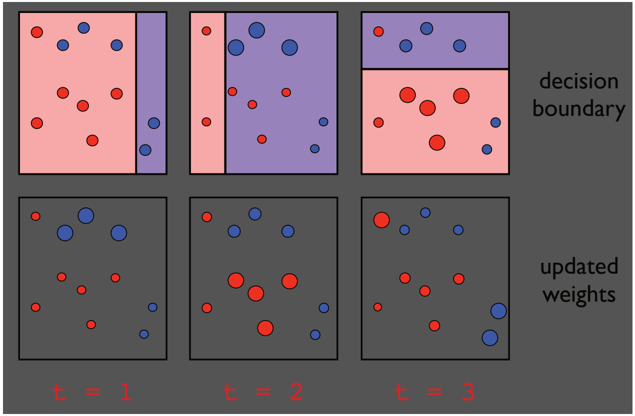
AdaBoost Algorithm

Inputs S: training set, $\mathcal{B}$: hypothesis space for weak learners, and T: number of weak learners.

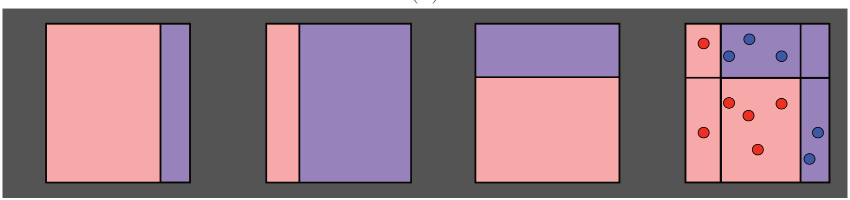
Output return a hypothesis h.

function ADABOOST(S, $\mathcal{B}$, T)
  for $i \leftarrow 1$ to m **do**
    $D_1(i) \leftarrow \frac{1}{m}$
  end for
  for $t \leftarrow 1$ to T **do**
    Let $h_t = \arg\min_{h \in \mathcal{B}} \quad \epsilon_t \triangleq \sum_{i=1}^{m} D_t(i)\, \mathbb{I}[h(\mathbf{x}_i) \neq y_1]$
    Let $\alpha_t \leftarrow \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
    Let $Z_t \leftarrow 2\sqrt{\epsilon_t(1-\epsilon_t)}$
    for $i \leftarrow 1$ to m **do**
      $D_{t+1}(i) \leftarrow \frac{D_t(i) \exp\left[-\alpha_t y_i h_t(\mathbf{x}_i)\right]}{Z_t}$
    end for
  end for
  Let $g \triangleq \sum_{t=1}^{T} \alpha_t h_t$
  return $h \triangleq \operatorname{sgn}(g)$
end function

AdaBoost algorithm (example)



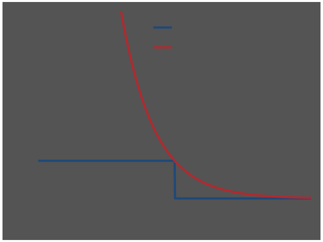
(a)



(b)

Minimizing exponential error

1. Consider the exponential loss function



2. The exponential loss function is defined by

$$\boxed{\hat{\mathbf{R}}(g_t) = \sum_{k=1}^{m} \exp\left[-y_k g_t(\mathbf{x}_k)\right],}$$

where $g_t(\mathbf{x})$ is a classifier defined in terms of a linear combination of base classifiers $h_l(\mathbf{x})$ as

$$\boxed{g_t(\mathbf{x}) = \sum_{l=1}^{t} \alpha_l h_l(\mathbf{x})}$$

1. The goal is to minimize $\hat{\mathbf{R}}$ with respect to both α_l and the parameters of the base classifiers h_l.
2. Since the base classifiers are built sequentially, we shall suppose that the base classifiers $h_1, \ldots, h_{t-1}$ and their weights $\alpha_1, \ldots, \alpha_{t-1}$ are fixed, and so we are minimizing only with respect to α_t and h_t.
3. Separating off the contribution from base classifier h_t, we can then write the $\hat{\mathbf{R}}(g_t)$ in the form

$$\begin{aligned}
\hat{\mathbf{R}}(g_t) &= \sum_{k=1}^{m} \exp\left[-y_k g_{t-1}(\mathbf{x}_k) - y_k \alpha_t h_t(\mathbf{x}_k)\right] \\
&= \sum_{k=1}^{m} D_t(k) \exp\left[-y_k \alpha_t h_t(\mathbf{x}_k)\right]
\end{aligned}$$

   where $D_t(k) = \exp\left[-y_k g_{t-1}(\mathbf{x}_k)\right]$ is constant because we optimize only w.r.t α_t and $h_t(\mathbf{x})$.

1. Let us to define
   - T_t as the set of instances that are correctly classified by $h_t(\mathbf{x})$.
   - M_t as the set of instances that are miss classified by $h_t(\mathbf{x})$.
2. We can in turn rewrite the error function in the form of

$$
\begin{aligned}
\hat{\mathbf{R}}(g_t) &= \sum_{k=1}^{m} D_t(k) \exp\left[-y_k \alpha_t h_t(\mathbf{x}_k)\right] \\
&= e^{-\alpha_t} \sum_{\mathbf{x}_k \in T_t} D_t(k) + e^{\alpha_t} \sum_{\mathbf{x}_k \in M_t} D_t(k) \\
&= e^{-\alpha_t} \sum_{\mathbf{x}_k \in T_t} D_t(k) + e^{\alpha_t} \sum_{\mathbf{x}_k \in M_t} D_t(k) + e^{-\alpha_t} \sum_{\mathbf{x}_k \in M_t} D_t(k) - e^{-\alpha_t} \sum_{\mathbf{x}_k \in M_t} D_t(k) \\
&= \left[e^{\alpha_t} - e^{-\alpha_t}\right] \sum_{\mathbf{x}_k \in M_t} D_t(k) + e^{-\alpha_t} \sum_{k=1}^{m} D_t(k) \\
&= \left[e^{\alpha_t} - e^{-\alpha_t}\right] \underbrace{\sum_{k=1}^{m} D_t(k)\, \mathbb{I}[h_t(\mathbf{x}_k \neq y_k)]}_{\epsilon_t} + e^{-\alpha_t} \sum_{k=1}^{m} D_t(k)
\end{aligned}
$$

1. The error function becomes

$$\hat{\mathbf{R}}(g_t) = \left[e^{\alpha_t} - e^{-\alpha_t}\right] \underbrace{\sum_{k=1}^{m} D_t(k)\, \mathbb{I}[h_t(\mathbf{x}_k \neq y_k)]}_{\epsilon_t} + e^{-\alpha_t} \sum_{k=1}^{m} D_t(k)$$

2. When minimizing $\hat{\mathbf{R}}(g_t)$ with respect to $h_t(\mathbf{x})$, the second term is constant, and is equivalent to minimizing ϵ_t because $\left[e^{\alpha_t} - e^{-\alpha_t}\right]$ does not affect the location of the minimum.
3. Minimizing $\hat{\mathbf{R}}(g_t)$ with respect to α_t equals to solve $\dfrac{\partial \hat{\mathbf{R}}(g_t)}{\partial \alpha_t} = 0$.

$$\begin{aligned}\frac{\partial \hat{\mathbf{R}}(g_t)}{\partial \alpha_t} &= \frac{e^{-\alpha_t} \sum_{\mathbf{x}_k \in T_t} D_t(k) + e^{\alpha_t} \sum_{\mathbf{x}_k \in M_t} D_t(k)}{\partial \alpha_t} \\ &= -e^{-\alpha_t} \sum_{\mathbf{x}_k \in T_t} D_t(k) + e^{\alpha_t} \sum_{\mathbf{x}_k \in M_t} D_t(k) = 0\end{aligned}$$

Minimizing exponential error (cont.)

1. Hence, we obtain

$$0 = -e^{-\alpha_t} \sum_{\mathbf{x}_k \in T_t} D_t(k) + e^{\alpha_t} \sum_{\mathbf{x}_k \in M_t} D_t(k)$$

2. Multiplying by e^{α_t}, becomes

$$\boxed{\underbrace{\sum_{\mathbf{x}_k \in T_t} D_t(k)}_{(1-\epsilon_t)} = e^{2\alpha_t} \underbrace{\sum_{\mathbf{x}_k \in M_t} D_t(k)}_{\epsilon_t}}$$

3. Solving this will results in

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

1. The value of $D_t(k)$ was defined as

$$D_t(k) = \exp\left[-y_k g_{t-1}(\mathbf{x}_k)\right].$$

2. Then the value of $D_{t+1}(k)$ equals to

$$\begin{aligned} D_{t+1}(k) &= \exp\left[-y_k g_t(\mathbf{x}_k)\right] \\ &= \exp\left[-y_k g_{t-1}(\mathbf{x}_k) - y_k \alpha_t h_t(\mathbf{x}_k)\right] \\ &= D_t(k) exp\left[-y_k \alpha_t h_t(\mathbf{x}_k)\right] \end{aligned}$$

3. Since $D_{t+1}(k)$ is a probability density function, then we must have $\sum_{k=1}^{m} D_{t+1}(k) = 1$. Hence, we have

$$\begin{aligned} \sum_{k=1}^{m} D_{t+1}(k) &= \sum_{k=1}^{m} D_t(k) exp\left[-y_k \alpha_t h_t(\mathbf{x}_k)\right] \\ &= \sum_{\mathbf{x}_k \in T_t} D_t(k) e^{-\alpha_t} + \sum_{\mathbf{x}_k \in M_t} D_t(k) e^{\alpha_t} \\ &= e^{-\alpha_t} \underbrace{\sum_{\mathbf{x}_k \in T_t} D_t(k)}_{1-\epsilon_t} + e^{\alpha_t} \underbrace{\sum_{\mathbf{x}_k \in M_t} D_t(k)}_{\epsilon_t} = e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t. \end{aligned}$$

- Let $Z_t = \sum_{k=1}^{m} D_{t+1}(k)$, hence we have

$$\boxed{Z_t = \sum_{k=1}^{m} D_{t+1}(k) = e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t}$$

- By substituting $\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ in the above equation, we have

$$\begin{aligned} Z_t &= \exp\left[\ln\sqrt{\frac{\epsilon_t}{1-\epsilon_t}}\right](1-\epsilon_t) + \exp\left[\ln\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}\right]\epsilon_t \\ &= \epsilon_t\sqrt{\frac{1-\epsilon_t}{\epsilon_t}} + (1-\epsilon_t)\sqrt{\frac{\epsilon_t}{1-\epsilon_t}} \\ &= 2\sqrt{\epsilon_t(1-\epsilon_t)} \end{aligned}$$

Generalization bound of AdaBoost

Theorem (Generalization bound of AdaBoost)

Let $H = \left\{ h : \mathcal{X} \mapsto \{-1, +1\} \;\middle|\; h = \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t\right), \alpha_t \in \mathbb{R}, h_t \in \mathcal{B} \right\}$ *be the hypothesis space for AdaBoost. Then, for all distribution* $\mathcal{D}$*, all training sets* $S \sim \mathcal{D}^m$*, for every* $\delta > 0$*, with probability at least* $(1-\delta)$*, for all* $h \in H$ *we have*

$$\mathbf{R}(h) \leq \hat{\mathbf{R}}(h) + \sqrt{\frac{VC(H) + \log(1/\delta)}{m}}.$$

Proof.

For proof, we must calculate

1. $\hat{\mathbf{R}}(h)$.
2. $VC(H)$.

□

Lemma

Let $g(\mathbf{x}) \triangleq \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$ *be the weighted linear combination of weak learners. Then*

$$D_{T+1}(i) = \frac{\exp\left[-y_i g(\mathbf{x}_i)\right]}{m \prod_{t=1}^{T} Z_t}$$

Proof.

1. We defined $D_{t+1}(i)$ as $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \exp\left[-\alpha_t y_i h_t(\mathbf{x}_i)\right]$.
2. We can now solve $D_{T+1}(i)$ recursively.

$$\begin{aligned}
D_{T+1}(i) &= \frac{D_T(i)}{Z_T} \exp\left[-\alpha_T y_i h_T(\mathbf{x}_i)\right] \\
&= D_{T-1}(i) \frac{\exp\left[-\alpha_{T-1} y_i h_{T-1}(\mathbf{x}_i)\right]}{Z_{T-1}} \times \frac{\exp\left[-\alpha_T y_i h_T(\mathbf{x}_i)\right]}{Z_T} \\
&= D_1(i) \frac{\exp\left[-\alpha_1 y_i h_1(\mathbf{x}_i)\right]}{Z_1} \times \frac{\exp\left[-\alpha_2 y_i h_2(\mathbf{x}_i)\right]}{Z_2} \times \ldots \times \frac{\exp\left[-\alpha_T y_i h_T(\mathbf{x}_i)\right]}{Z_T} \\
&= \frac{1}{m} \frac{\exp\left[-y_i \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i)\right]}{\prod_{t=1}^{T} Z_t} = \frac{\exp\left[-y_i g(\mathbf{x}_i)\right]}{m \prod_{t=1}^{T} Z_t}.
\end{aligned}$$

$\square$

Bounds on the empirical error of AdaBoost

Lemma

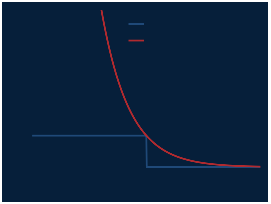
Let $g(\mathbf{x}) \triangleq \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$ *be the weighted linear combination of weak learners and* $h(\mathbf{x}) \triangleq \operatorname{sgn}(g(\mathbf{x}))$*. Then, we have*

$$\hat{\mathbf{R}}(h) \leq \prod_{t=1}^{T} Z_t.$$

Proof.

We start by the definition of empirical loss.

$$\begin{aligned}
\hat{\mathbf{R}}(h) &= \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}[h(\mathbf{x}_i) \neq y_i] = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}[y_i g(\mathbf{x}_i) \leq 0] \\
&\leq \frac{1}{m}\sum_{i=1}^{m} e^{-y_i g(\mathbf{x}_i)} = \frac{1}{m}\sum_{i=1}^{m} D_{T+1}(i) m \prod_{t=1}^{T} Z_t \\
&= \prod_{t=1}^{T} Z_t \sum_{i=1}^{m} D_{T+1}(i) = \prod_{t=1}^{T} Z_t.
\end{aligned}$$



□

Theorem (Bounds on the empirical error of AdaBoost)

Let $g(\mathbf{x}) \triangleq \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$ *be the weighted linear combination of weak learners and* $h(\mathbf{x}) \triangleq \operatorname{sgn}(g(\mathbf{x}))$. *Then, we have*

$$\hat{\mathsf{R}}(h) \leq \exp\left[-2\sum_{t=1}^{T}\left(\frac{1}{2}-\epsilon_t\right)^2\right].$$

Furthermore, if for all $t \in \{1, 2, \ldots, T\}$, *we have* $\gamma \leq \left(\frac{1}{2}-\epsilon_t\right)$, *then* $\hat{\mathsf{R}}(h) \leq e^{-2T\gamma^2}$.

Proof of Bounds on the empirical error of AdaBoost

By using the two preceding lemmas

$$\begin{aligned}\hat{\mathsf{R}}(h) &\leq \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T}\left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] = \prod_{t=1}^{T}\left[2\sqrt{\left(\frac{1}{2}-\gamma\right)\left(\frac{1}{2}+\gamma\right)}\right] = \prod_{t=1}^{T}\left[\sqrt{1-4\gamma^2}\right]\\ &\leq \prod_{t=1}^{T}\sqrt{\exp\left(-4\gamma^2\right)} = \prod_{t=1}^{T}\exp\left(-2\gamma^2\right) = \exp\left[-2\sum_{t=1}^{T}\gamma^2\right] = e^{-2T\gamma^2}.\end{aligned}$$

Proof of Bounds on the empirical error of AdaBoost (cont.)

By using the two preceding lemmas

$$\begin{aligned}\hat{\mathbf{R}}(h) &\leq \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T}\left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] = \prod_{t=1}^{T}\left[2\sqrt{\left(\frac{1}{2}-\gamma\right)\left(\frac{1}{2}+\gamma\right)}\right] = \prod_{t=1}^{T}\left[\sqrt{1-4\gamma^2}\right] \\ &\leq \prod_{t=1}^{T}\sqrt{\exp\left(-4\gamma^2\right)} = \prod_{t=1}^{T}\exp\left(-2\gamma^2\right) = \exp\left[-2\sum_{t=1}^{T}\gamma^2\right] = e^{-2T\gamma^2}.\end{aligned}$$

To derive the bound of theorem, in second equality use with $x = \epsilon_t$

$2\sqrt{x(1-x)} = \sqrt{4x-4x^2} = \sqrt{1-1+4x-4x^2} = \sqrt{1-(1-4x+4x^2)} = \sqrt{1-2\left(\frac{1}{2}-x\right)^2}$.

The second inequality follows from the inequality $1 - x \leq e^{-x}$, which is valid for all $x \in \mathbb{R}$. $\square$

Theorem (Bounds on the $VC(H)$)

Let $\mathcal{B}$ be a base class and let $H = \left\{\mathbf{x} \mapsto \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right) \,\middle|\, \alpha \in \mathbb{R}^T, \forall t \;\; h_t \in \mathcal{B}\right\}$ be the hypothesis space where the output of AdaBoost will be a member of it. Assume that both T and $VC(\mathcal{B})$ are at least 3. Then

$$VC(H) \leq T[VC(\mathcal{B}) + 1][3\log(T[VC(\mathcal{B}) + 1]) + 2].$$

Corollary (Sauer-Shelah Lemma)

Let H be a hypothesis classes with $VC(H) = d$, then for $m > d > 1$, we have

$$\Pi_H(m) \leq \left(\frac{em}{d}\right)^d = O(m^d)$$

Lemma

Let $a > 0$. Then: $x \geq 2a\log(a) \Rightarrow x \geq a\log(x)$. It follows that a necessary condition for the inequality $x < a\log(x)$ to hold is that $x < 2a\log(a)$.

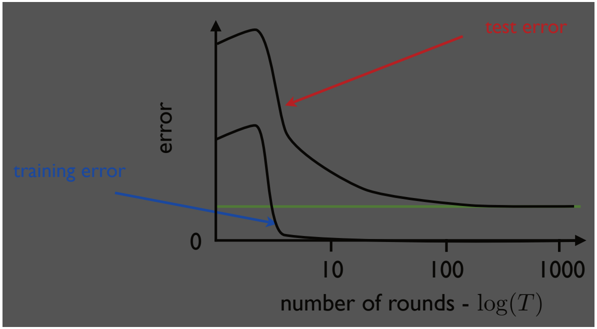
Proof of Bounds on the $VC(H)$.

1. Let $VC(\mathcal{B}) = d$ and $C = \{x_1, \dots, x_m\}$ be a set that is shattered by H.
2. Each labeling of C by $h \in H$ is obtained by first choosing $h_1, \dots, h_T \in \mathcal{B}$ and then applying a halfspace hypothesis over the vector $(h_1(\mathbf{x}), \dots, h_T(\mathbf{x}))$.
3. By Sauer's lemma, at most $(em/d)^d$ different dichotomies induced by $\mathcal{B}$ over C.
4. We need to choose T hypotheses out of at most $(em/d)^d$ different hypotheses. There are at most $(em/d)^{dT}$ ways to do it.
5. Next, for each such choice, we apply a linear predictor, which yields at most $(em/T)^T$ dichotomies.
6. Hence, the number of dichotomies is $(em/d)^{dT}(em/T)^T \leq m^{(d+1)T}$.
7. Since we assume that C is shattered, we must have $2^m \leq m^{(d+1)T}$.
8. The above lemma tells us that a necessary condition for the preceding to hold is

$$m \leq 2\frac{(d+1)T}{\log(2)} \log \frac{(d+1)T}{\log(2)} \leq T[VC(\mathcal{B}) + 1][3\log(T[VC(\mathcal{B}) + 1]) + 2].$$

□

1. Theorem Bounds on the $VC(H)$ shows that $VC(H) = O(dT \log dT)$, thus, the bound suggests that AdaBoost could overfit for large values of T.
2. The estimation error of AdaBoost grows linearly with T.
3. The empirical error of AdaBoost grows linearly with T.
4. Hence, T can be used to decrease the approximation error of AdaBoost.
5. 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.



6. These empirical results can be explained using margin-based analysis.

Margin-based analysis

1. Confidence margin of a real-valued function g at a point $\mathbf{x}$ labeled with y is $yg(\mathbf{x})$.
2. Defining geometric margin for linear hypotheses with a norm-1 constraint, such as hypotheses returned by AdaBoost, which relates to confidence margin.
3. Function $g = \sum_{t=1}^{T} \alpha_t h_t$ can be represented as $\langle \alpha, \mathbf{h} \rangle$, where $\alpha = (\alpha_1, \ldots, \alpha_T)^T$ and $\mathbf{h} = (h_1, \ldots, h_T)^T$.
4. For ensemble linear combinations such as those returned by AdaBoost, additionally, the weight vector is non-negative: $\alpha \geq 0$.
5. Geometric margin for such ensemble functions is based on norm-1 while geometric margin is based on norm-2.

Definition (L_1 geometric margin)

The L_1-geometric margin ρ_g of $g = \sum_{t=1}^{T} \alpha_t h_t$ with $\alpha \neq 0$ at a $\mathbf{x}_k \in \mathcal{X}$ defined as

$$\rho_g(\mathbf{x}) = \frac{|g(\mathbf{x})|}{\|\alpha\|_1} = \frac{|\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})|}{\|\alpha\|_1} = \frac{|\langle \alpha, \mathbf{h}(\mathbf{x}) \rangle|}{\|\alpha\|_1}$$

The L_1-margin of g over a sample $S = (x_1, \ldots, x_m)$ is its minimum margin at the points in that sample.

$$\rho_g = \min_{i \in \{1,2,\ldots,m\}} \rho_g(\mathbf{x}) = \min_{i \in \{1,2,\ldots,m\}} \frac{|\langle \alpha, \mathbf{h}(\mathbf{x}) \rangle|}{\|\alpha\|_1}$$

To distinguish this geometric margin from the geometric margin of SVM, we use the following notations

$$\rho_1(\mathbf{x}) = \frac{|\langle \alpha, \mathbf{h}(\mathbf{x}) \rangle|}{\|\alpha\|_1} \qquad L_1 - \text{margin}$$

$$\rho_2(\mathbf{x}) = \frac{|\langle \alpha, \mathbf{h}(\mathbf{x}) \rangle|}{\|\alpha\|_2} \qquad L_2 - \text{margin}$$

Lemma

For $p, q \geq 1$, p and q are conjugate, that is $\frac{1}{p} + \frac{1}{q} = 1$, then $\frac{|\langle \alpha, \mathbf{h}(\mathbf{x}) \rangle|}{\|\alpha\|_p}$ is $q-$norm distance of $\mathbf{h}(\mathbf{x})$ to the hyperplane $\langle \alpha, \mathbf{h}(\mathbf{x}) \rangle = 0$.

1. Hence, $\rho_2(\mathbf{x})$ is norm-2 distance of $\mathbf{h}(\mathbf{x})$ to the hyperplane $\langle \alpha, \mathbf{h}(\mathbf{x}) \rangle = 0$ and $\rho_1(\mathbf{x})$ is norm-∞ distance of $\mathbf{h}(\mathbf{x})$ to the hyperplane $\langle \alpha, \mathbf{h}(\mathbf{x}) \rangle = 0$.
2. Define $\bar{g} = \frac{g}{\|\alpha\|_1}$, then the confidence margin of $\bar{g}$ at $\mathbf{x}$ coincides with the L_1-geometric margin of g: $y\bar{g(\mathbf{x})} = \frac{yg(\mathbf{x})}{\|\alpha\|_1} = \rho_g(\mathbf{x})$.
3. Since $\alpha_t \geq 0$, then $\rho_g(\mathbf{x})$ is convex combination of base hypothesis values $h_t(\mathbf{x})$.

For any hypothesis set $\mathcal{H}$ of real-valued functions, $conv(\mathcal{H})$ denotes its convex hull as

$$conv(\mathcal{H}) = \left\{ \sum_{k=1}^{p} \mu_k h_k \;\middle|\; p \geq 1, \forall k \in \{1, 2, \ldots, p\}, h_k \in \mathcal{H}, \sum_{k=1}^{p} \mu_k \leq 1 \right\}$$

Lemma (Empirical Rademacher complexity of $conv(\mathcal{H})$)

Let $\mathcal{H} = \{h : \mathcal{X} \mapsto \mathbb{R}\}$. Then for any sample S, we have $\hat{\mathcal{R}}_S(conv(\mathcal{H})) = \hat{\mathcal{R}}_S(\mathcal{H})$.

Proof.

$$\begin{aligned}
\hat{\mathcal{R}}_S(conv(\mathcal{H})) &= \frac{1}{m} \mathop{\mathbb{E}}_{\sigma} \left[\sup_{h_1,\ldots,h_p \in \mathcal{H}, \|\mu\|_1 \leq 1} \sum_{i=1}^{m} \sigma_i \sum_{k=1}^{p} \mu_k h_k(\mathbf{x}_i) \right] \\
&= \frac{1}{m} \mathop{\mathbb{E}}_{\sigma} \left[\sup_{h_1,\ldots,h_p \in \mathcal{H}} \sup_{\|\mu\|_1 \leq 1} \sum_{k=1}^{p} \mu_k \sum_{i=1}^{m} \sigma_i h_k(\mathbf{x}_i) \right] \\
&= \frac{1}{m} \mathop{\mathbb{E}}_{\sigma} \left[\sup_{h_1,\ldots,h_p \in \mathcal{H}} \max_{k \in \{1,\ldots,p\}} \sum_{i=1}^{m} \sigma_i h_k(\mathbf{x}_i) \right] \\
&= \frac{1}{m} \mathop{\mathbb{E}}_{\sigma} \left[\sup_{h \in \mathcal{H}} \sum_{i=1}^{m} \sigma_i h(\mathbf{x}_i) \right] = \hat{\mathcal{R}}_S(\mathcal{H}).
\end{aligned}$$

□

The following theorem was proved for SVM.

Theorem (Ensemble Rademacher margin bound)

Let $\mathcal{H}$ be a set of real-valued functions. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $(1-\delta)$, the following hold for all $h \in conv(\mathcal{H})$:

$$
\begin{aligned}
\mathbf{R}(h) &\leq \hat{\mathbf{R}}_\rho(h) + \frac{2}{\rho}\mathcal{R}_m(H) + \sqrt{\frac{\log(1/\delta)}{2m}} \\
\mathbf{R}(h) &\leq \hat{\mathbf{R}}_\rho(h) + \frac{2}{\rho}\hat{\mathcal{R}}_S(H) + 3\sqrt{\frac{\log(1/\delta)}{2m}}
\end{aligned}
$$

Corollary (Ensemble VC-dimension margin bound)

Let $\mathcal{H} = \{\mathcal{X} \mapsto \{-1,+1\}\}$ with VC-dimension d. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $(1-\delta)$, the following holds for all $h \in conv(\mathcal{H})$

$$
\mathbf{R}(h) \leq \hat{\mathbf{R}}_{S,\rho}(h) + \frac{2}{\rho}\sqrt{\frac{2d \log \frac{em}{d}}{m}} + \sqrt{\frac{\log(1/\delta)}{2m}}
$$

These bounds can be generalized to hold uniformly for all $\rho \in (0, 1]$, at the price of an additional term of the form of $\sqrt{\dfrac{\log \log_2 \frac{2}{\delta}}{m}}$.

The given bound can not be directly applied to the function g returned by AdaBoost, since it is not a convex combination of base hypotheses, but they can be applied to its normalized version $\bar{g} \in conv(\mathcal{H})$.

Notice that from the point of view of binary classification, $\bar{g}$ and g are equivalent but their empirical margin losses are distinct.

Theorem (Bound on empirical margin loss)

Let $g = \sum_{t=1}^{T} \alpha_t h_t$ denote the function returned by AdaBoost after T rounds of boosting and assume for all $t \in \{1, \ldots, T\}$ that $\epsilon_t < \frac{1}{2}$, which implies $\alpha_t > 0$. Then for any $\rho > 0$, the following holds

$$\hat{\mathbf{R}}_{S,\rho}(h) \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t^{1-\rho}(1-\epsilon_t)^{1+\rho}}$$

Proof of Bound on empirical margin loss.

1. Recall that

$$\mathbb{I}[u \le 0] \le e^{-u}. \qquad Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

$$D_{t+1}(i) = \frac{\exp[-y_i g(\mathbf{x}_i)]}{m\prod_{t=1}^{T} Z_t} \qquad \alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

2. Then, we can write

$$\begin{aligned}
\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}\left[yg(\mathbf{x}_i) - \rho\|\alpha\|_1 \le 0\right] &\le \frac{1}{m}\sum_{i=1}^{m}\exp\left[-y_i g(\mathbf{x}_i) + \rho\|\alpha\|_1\right] \\
&= \frac{1}{m}\sum_{i=1}^{m} e^{\rho\|\alpha\|_1}\left[m\prod_{t=1}^{T} Z_t\right] D_{T+1}(i) \\
&= e^{\rho\|\alpha\|_1}\left[m\prod_{t=1}^{T} Z_t\right] = e^{\rho\sum_{t'}\alpha_{t'}}\left[m\prod_{t=1}^{T} Z_t\right] \\
&= 2^T\prod_{t=1}^{T}\left[\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}\right]^{\rho}\sqrt{\epsilon_t(1-\epsilon_t)}
\end{aligned}$$

$\square$

1. Does AdaBoost maximize L_1-geometric margin?
2. Maximum margin for a linearly separable sample $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ is

$$\rho = \max_{\alpha} \min_{i \in \{1,2,\ldots,m\}} \frac{y_i \langle \alpha, \mathbf{h}(\mathbf{x}_i) \rangle}{\|\alpha\|_1}$$

3. Then, the optimization problem can be written as

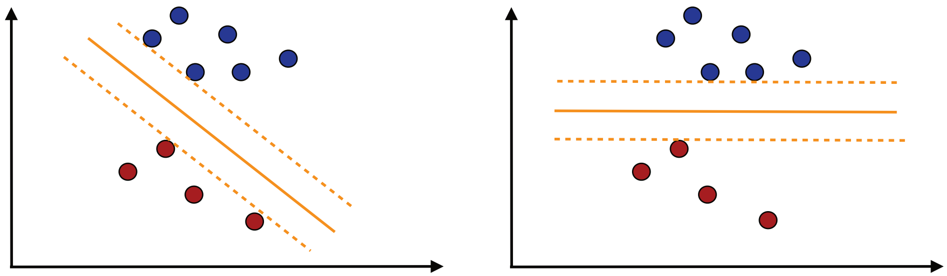
$$\begin{aligned} &\max_{\alpha} \rho \\ \text{subject to } \quad &\frac{y_i \langle \alpha, \mathbf{h}(\mathbf{x}_i) \rangle}{\|\alpha\|_1} \geq \rho \quad \forall i \in \{1, 2, \ldots, m\}. \end{aligned}$$

4. Since $\frac{\langle \alpha, \mathbf{h}(\mathbf{x}_i) \rangle}{\|\alpha\|_1}$ is invariant to scaling of α, we can restrict ourselves to $\|\alpha\|_1 = 1$.

1. Since $\frac{\langle \alpha, \mathbf{h}(\mathbf{x}_i) \rangle}{\|\alpha\|_1}$ is invariant to scaling of α, we can restrict ourselves to $\|\alpha\|_1 = 1$.
2. Then AdaBoost leads to the following optimization problem

$$\begin{aligned} &\max_{\alpha} \rho \\ \text{subject to} \quad & y_i \langle \alpha, \mathbf{h}(\mathbf{x}_i) \rangle \geq \rho \quad \forall i \in \{1, 2, \ldots, m\} \\ & \left(\sum_{t=1}^{T} \alpha_t = 1 \right) \wedge \left(\alpha_t \geq 0 \ \ \forall t \in \{1, 2, \ldots, T\} \right). \end{aligned}$$

3. The empirical results do not show a systematic benefit for the solution of the LP.
4. In many cases, AdaBoost outperforms LP algorithm.
5. The margin theory described does not seem sufficient to explain that performance.



Norm $|| \cdot ||_2$ Norm $|| \cdot ||_\infty$

Summary

1. AdaBoost offers several advantages
    - It is simple.
    - Its implementation is straightforward.
    - The time complexity of each round of boosting as a function of the sample size is rather favorable. If AdaBoost uses Decision Stumps as base classifier, the running time is $O(mnT)$.
    - AdaBoost benefits from a rich theoretical analysis.

Summary

1. There are many theoretical questions related to AdaBoost algorithm
    - The algorithm in fact does not maximize the margin.
    - The algorithms that do maximize the margin do not always outperform it.
    - The need to select the parameter T and $\mathcal{B}$. Larger values of T can lead to overfitting. In practice, T is typically determined via cross-validation.
    - We must control complexity of $\mathcal{B}$ in order to guarantee generalization; insufficiently complex $\mathcal{B}$ could lead to low margins.
    - The performance of AdaBoost in the presence of noise, at least in some tasks, degrades.

1. Section 14.3 of Christopher M Bishop Book[1].
2. Chapters 9 and 10 of Shai Shalev-Shwartz and Shai Ben-David Book[2].
3. Chapter 7 of Mehryar Mohri and Afshin Rostamizadeh and Ameet Talwalkar Book[3].

[1] Christopher M. Bishop (2006). *Pattern Recognition and Machine Learning*. Berlin, Heidelberg: Springer-Verlag.
[2] Shai Shalev-Shwartz and Shai Ben-David (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press.
[3] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar (2018). *Foundations of Machine Learning*. Second Edition. MIT Press.

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Berlin, Heidelberg: Springer-Verlag.

Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar (2018). *Foundations of Machine Learning*. Second Edition. MIT Press.

Shalev-Shwartz, Shai and Shai Ben-David (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press.

Questions?