ELEN 4903: Machine Learning

Week 7, Lecture 1, 2/28/2018

Prof. John Paisley

Department of Electrical Engineering
& Data Science Institute

Columbia University

BOOSTING

Robert E. Schapire and Yoav Freund, *Boosting: Foundations and Algorithms*, MIT Press, 2012.
See this textbook for many more details. (I borrow some figures from that book.)

BAGGING CLASSIFIERS

Algorithm: Bagging binary classifiers

Given $(x_1, y_1), \ldots, (x_n, y_n)$, $x \in \mathcal{X}$, $y \in \{-1, +1\}$

- For $b = 1, \ldots, B$
  - Sample a bootstrap dataset $\mathcal{B}_b$ of size n. For each entry in $\mathcal{B}_b$, select (x_i, y_i) with probability $\frac{1}{n}$. Some (x_i, y_i) will repeat and some won't appear in $\mathcal{B}_b$.
  - Learn a classifier f_b using data in $\mathcal{B}_b$.
- Define the classification rule to be

$$f_{bag}(x_0) = \text{sign}\left(\sum_{b=1}^{B} f_b(x_0)\right).$$

- With bagging, we observe that a *committee* of classifiers votes on a label.
- Each classifier is learned on a *bootstrap sample* from the data set.
- Learning a collection of classifiers is referred to as an *ensemble method*.

BOOSTING

How is it that a committee of blockheads can somehow arrive at highly reasoned decisions, despite the weak judgment of the individual members?

- Schapire & Freund, "Boosting: Foundations and Algorithms"

Boosting is another powerful method for ensemble learning. It is similar to bagging in that a set of classifiers are combined to make a better one.

It works for any classifier, but a "weak" one that is easy to learn is usually chosen. (weak = accuracy a little better than random guessing)

Short history

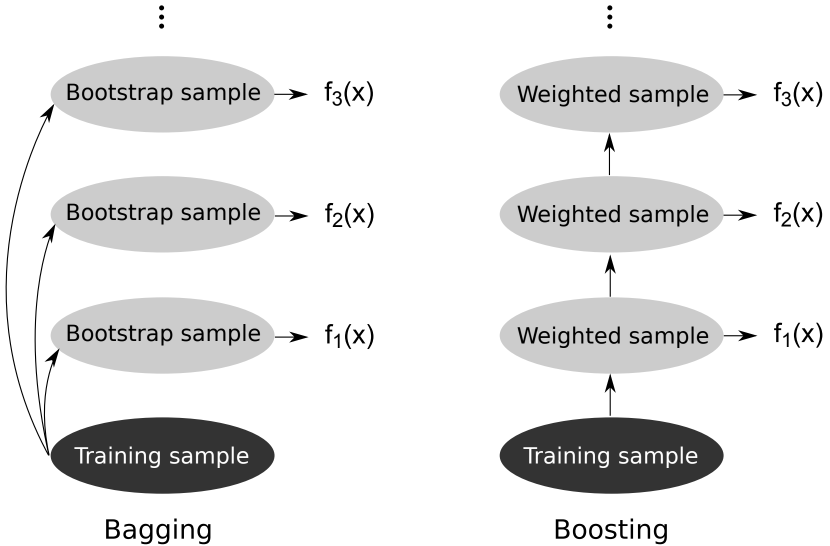
1984 : Leslie Valiant and Michael Kearns ask if "boosting" is possible.

1989 : Robert Schapire creates first boosting algorithm.

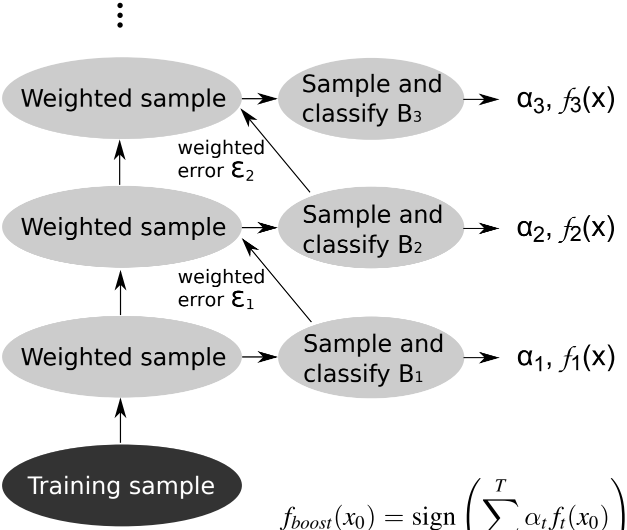
1990 : Yoav Freund creates an optimal boosting algorithm.

1995 : Freund and Schapire create AdaBoost (Adaptive Boosting), the major boosting algorithm.

Bagging vs Boosting (overview)



The AdaBoost Algorithm (sampling version)



$$f_{boost}(x_0) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x_0)\right)$$

Boosting

The AdaBoost Algorithm (sampling version)

Algorithm: Boosting a binary classifier

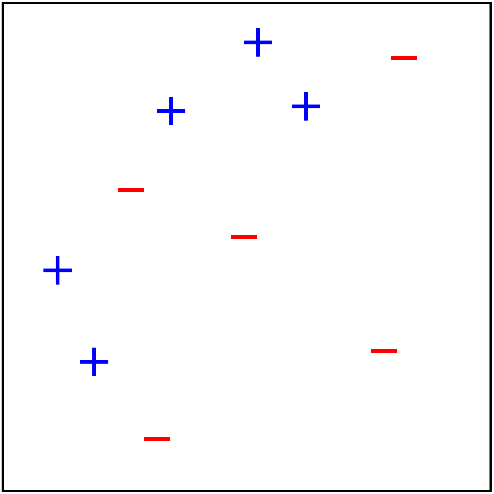
Given $(x_1, y_1), \dots, (x_n, y_n)$, $x \in \mathcal{X}$, $y \in \{-1, +1\}$, set $w_1(i) = \frac{1}{n}$ for $i = 1:n$

- For $t = 1, \dots, T$
  1. Sample a bootstrap dataset $\mathcal{B}_t$ of size n according to distribution w_t. Notice we pick (x_i, y_i) with probability $w_t(i)$ and not $\frac{1}{n}$.
  2. Learn a classifier f_t using data in $\mathcal{B}_t$.
  3. Set $\epsilon_t = \sum_{i=1}^{n} w_t(i) \mathbb{1}\{y_i \neq f_t(x_i)\}$ and $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$.
  4. Scale $\hat{w}_{t+1}(i) = w_t(i) e^{-\alpha_t y_i f_t(x_i)}$ and set $w_{t+1}(i) = \frac{\hat{w}_{t+1}(i)}{\sum_j \hat{w}_{t+1}(j)}$.
- Set the classification rule to be

$$f_{boost}(x_0) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x_0) \right).$$

Comment: Description usually simplified to "learn classifier f_t using distribution w_t."

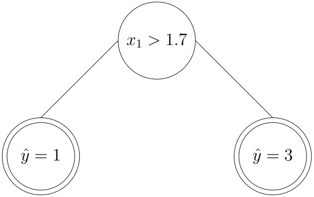
Boosting a decision stump (example 1)



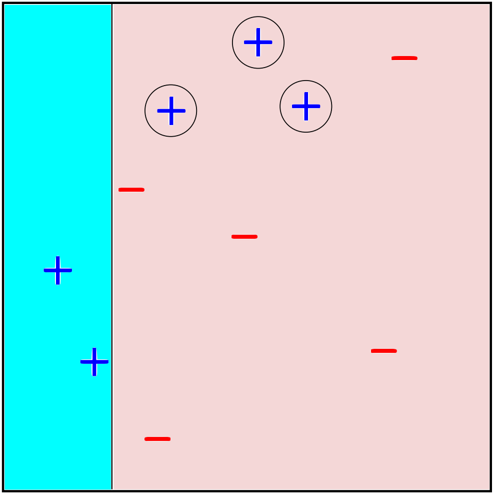
Original data

Uniform distribution, w_1
Learn *weak classifier*

Here: Use a decision stump



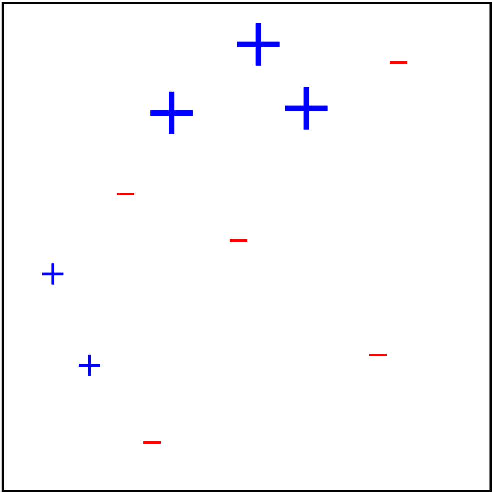
Boosting a decision stump (example 1)



Round 1 classifier

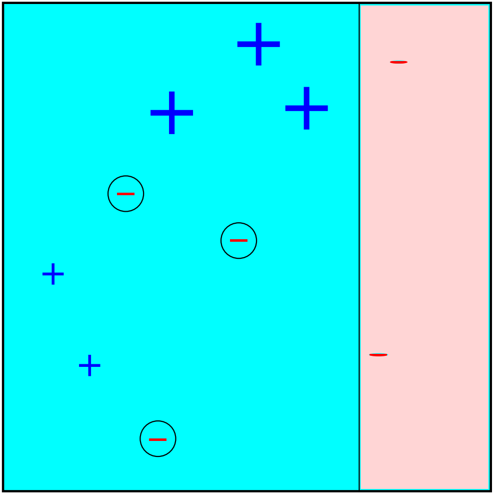
Weighted error: $\epsilon_1 = 0.3$
Weight update: $\alpha_1 = 0.42$

Boosting a decision stump (example 1)



Weighted data

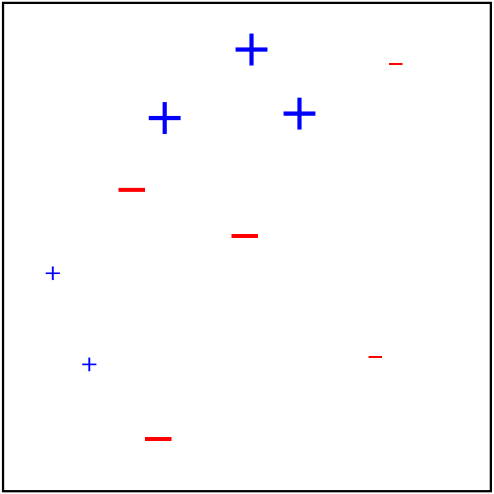
After round 1



Round 2 classifier

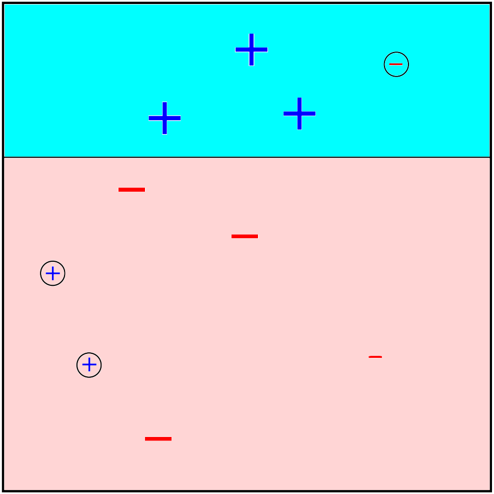
Weighted error: $\epsilon_2 = 0.21$
Weight update: $\alpha_2 = 0.65$

Boosting a decision stump (example 1)



Weighted data

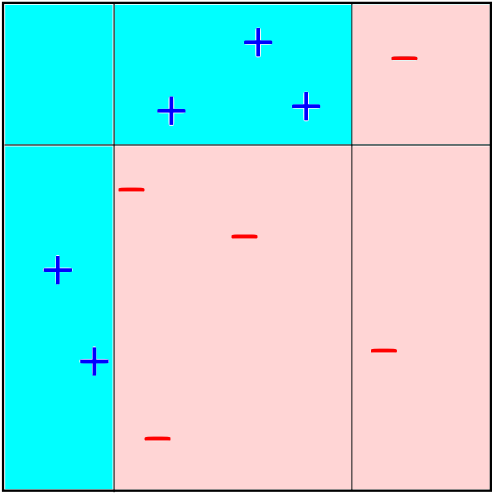
After round 2



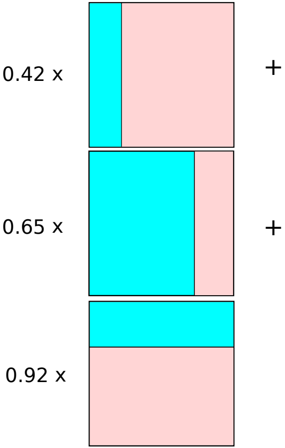
Round 2 classifier

Weighted error: $\epsilon_3 = 0.14$
Weight update: $\alpha_3 = 0.92$

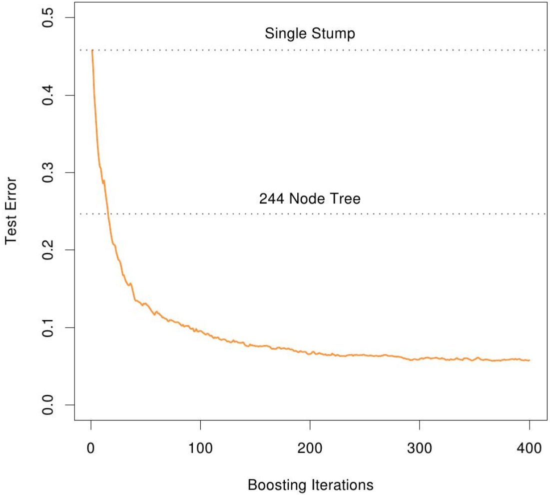
BOOSTING A DECISION STUMP (EXAMPLE 1)



Classifier after three rounds



BOOSTING A DECISION STUMP (EXAMPLE 2)



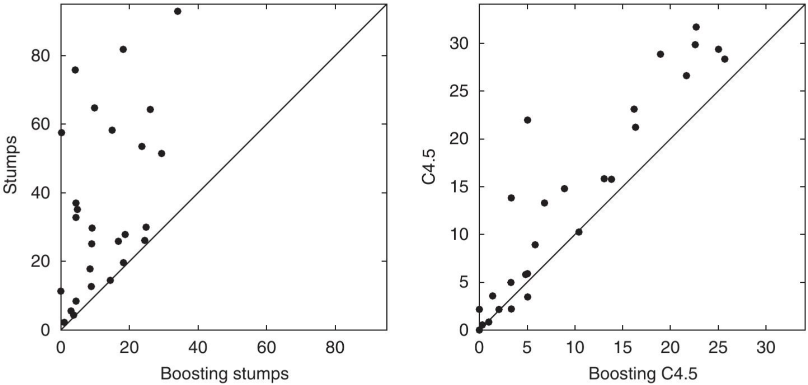
Example problem

Random guessing
50% error

Decision stump
45.8% error

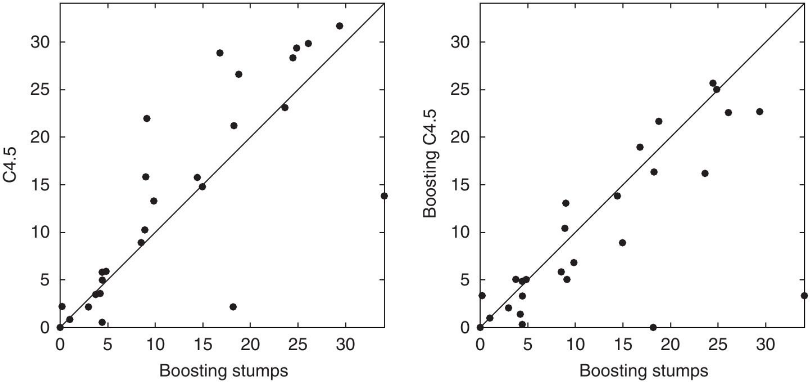
Full decision tree
24.7% error

Boosted stump
5.8% error



Point = one dataset. Location = error rate w/ and w/o boosting. The boosted version of the same classifier almost always produces better results.

BOOSTING



(left) Boosting a bad classifier is often better than not boosting a good one.
(right) Boosting a good classifier is often better, but can take more time.

Boosting and feature maps

Q: What makes boosting work so well?

A: This is a well-studied question. We will present one analysis later, but we can also give intuition by tying it in with what we've already learned.

The classification for a new x_0 from boosting is

$$f_{boost}(x_0) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x_0)\right).$$

Define $\phi(x) = [f_1(x), \ldots, f_T(x)]^\top$, where each $f_t(x) \in \{-1, +1\}$.

- We can think of $\phi(x)$ as a high dimensional feature map of x.
- The vector $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_T]^\top$ corresponds to a hyperplane.
- So the classifier can be written $f_{boost}(x_0) = \text{sign}(\phi(x_0)^\top \boldsymbol{\alpha})$.
- Boosting learns the feature mapping and hyperplane simultaneously.

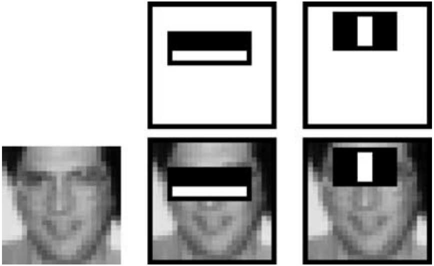
APPLICATION: FACE DETECTION

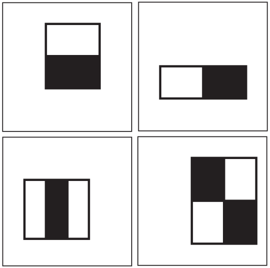
FACE DETECTION (VIOLA & JONES, 2001)

Problem: Locate the faces in an image or video.

Processing: Divide image into patches of different scales, e.g., 24×24, 48×48, etc. Extract *features* from each patch.

Classify each patch as face or no face using a *boosted decision stump*. This can be done in real-time, for example by your digital camera (at 15 fps).





- One patch from a larger image. Mask it with many “feature extractors.”
- Each pattern gives one number, which is the sum of all pixels in black region minus sum of pixels in white region (total of 45,000+ features).

Face detection (example results)

Analysis of boosting

Training error theorem

We can use *analysis* to make a statement about the accuracy of boosting *on the training data*.

Theorem: Under the AdaBoost framework, if ϵ_t is the weighted error of classifier f_t, then for the classifier $f_{boost}(x_0) = \text{sign}(\sum_{t=1}^{T} \alpha_t f_t(x_0))$,

$$\text{training error } = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{y_i \neq f_{boost}(x_i)\} \;\leq\; \exp\Big(-2\sum_{t=1}^{T}(\tfrac{1}{2} - \epsilon_t)^2\Big).$$

Even if each ϵ_t is only a little better than random guessing, the sum over T classifiers can lead to a large negative value in the exponent when T is large.

For example, if we set:

$$\epsilon_t = 0.45,\; T = 1000 \;\rightarrow\; \text{training error } \leq\; 0.0067.$$

Setup

We break the proof into three steps. It is an application of the fact that

$$\text{if} \quad \underbrace{a < b}_{\text{Step 2}} \quad \text{and} \quad \underbrace{b < c}_{\text{Step 3}} \quad \text{then} \quad \underbrace{a < c}_{\text{conclusion}}$$

- Step 1 calculates the value of b.
- Steps 2 and 3 prove the two inequalities.

Also recall the following step from AdaBoost:

- Update $\hat{w}_{t+1}(i) = w_t(i)e^{-\alpha_t y_i f_t(x_i)}$.
- Normalize $w_{t+1}(i) = \dfrac{\hat{w}_{t+1}(i)}{\sum_j \hat{w}_{t+1}(j)} \quad \longrightarrow \quad$ Define $Z_t = \sum_j \hat{w}_{t+1}(j)$.

PROOF OF THEOREM $(a \le \boldsymbol{b} \le c)$

Step 1

We first want to expand the equation of the weights to show that

$$w_{T+1}(i) = \frac{1}{n} \frac{e^{-y_i \sum_{t=1}^{T} \alpha_t f_t(x_i)}}{\prod_{t=1}^{T} Z_t} := \frac{1}{n} \frac{e^{-y_i h_T(x_i)}}{\prod_{t=1}^{T} Z_t} \; \rightarrow \; h_T(x) := \sum_{t=1}^{T} \alpha_t f_t(x_i)$$

Derivation of Step 1:

Notice the update rule: $w_{t+1}(i) = \dfrac{1}{Z_t} w_t(i) e^{-\alpha_t y_i f_t(x_i)}$

Do the same expansion for $w_t(i)$ and continue until reaching $w_1(i) = \frac{1}{n}$,

$$w_{T+1}(i) = w_1(i) \frac{e^{-\alpha_1 y_i f_1(x_i)}}{Z_1} \times \cdots \times \frac{e^{-\alpha_T y_i f_T(x_i)}}{Z_T}$$

The product $\prod_{t=1}^{T} Z_t$ is "b" above. We use this form of $w_{T+1}(i)$ in Step 2.

Step 2

Next show the training error of $f_{boost}^{(T)}$ (boosting after T steps) is $\leq \prod_{t=1}^{T} Z_t$.
Currently we know

$$w_{T+1}(i) = \frac{1}{n}\frac{e^{-y_i h_T(x_i)}}{\prod_{t=1}^{T} Z_t} \;\Rightarrow\; w_{T+1}(i)\prod_{t=1}^{T} Z_t = \frac{1}{n}e^{-y_i h_T(x_i)} \quad \& \quad f_{boost}^{(T)}(x) = \text{sign}(h_T(x))$$

Derivation of Step 2:

Observe that $0 < e^{z_1}$ and $1 < e^{z_2}$ for any $z_1 < 0 < z_2$. Therefore

$$\begin{aligned}
\underbrace{\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\{y_i \neq f_{boost}^{(T)}(x_i)\}}_{a} \quad &\leq \quad \frac{1}{n}\sum_{i=1}^{n} e^{-y_i h_T(x_i)} \\
&= \quad \sum_{i=1}^{n} w_{T+1}(i)\prod_{t=1}^{T} Z_t \quad = \quad \underbrace{\prod_{t=1}^{T} Z_t}_{b}
\end{aligned}$$

"a" is the training error – the quantity we care about.

Step 3

The final step is to calculate an upper bound on Z_t, and by extension $\prod_{t=1}^{T} Z_t$.

Derivation of Step 3:

This step is slightly more involved. It also shows why $\alpha_t := \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.

$$
\begin{aligned}
Z_t &= \sum_{i=1}^{n} w_t(i) e^{-\alpha_t y_i f_t(x_i)} \\
&= \sum_{i\,:\,y_i = f_t(x_i)} e^{-\alpha_t} w_t(i) + \sum_{i\,:\,y_i \neq f_t(x_i)} e^{\alpha_t} w_t(i) \\
&= e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t
\end{aligned}
$$

Remember we *defined* $\epsilon_t = \sum_{i\,:\,y_i \neq f_t(x_i)} w_t(i)$, the probability of error for w_t.

Derivation of Step 3 (continued):

Remember from Step 2 that

$$\text{training error} \;=\; \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{y_i \neq f_{boost}(x_i)\} \;\leq\; \prod_{t=1}^{T} Z_t\,.$$

and we just showed that $Z_t = e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t$.

We want the training error to be small, so we pick α_t to *minimize* Z_t. Minimizing, we get the value of α_t used by AdaBoost:

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

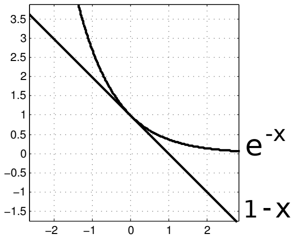
Plugging this value back in gives $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$.

PROOF OF THEOREM $(a \leq \boldsymbol{b} \leq \boldsymbol{c})$

Derivation of Step 3 (continued):

Next, re-write Z_t as

$$\begin{aligned} Z_t &= 2\sqrt{\epsilon_t(1-\epsilon_t)} \\ &= \sqrt{1-4(\frac{1}{2}-\epsilon_t)^2} \end{aligned}$$



Then, use the inequality $1-x \leq e^{-x}$ to conclude that

$$Z_t = \left(1-4(\tfrac{1}{2}-\epsilon_t)^2\right)^{\frac{1}{2}} \leq \left(e^{-4(\frac{1}{2}-\epsilon_t)^2}\right)^{\frac{1}{2}} = e^{-2(\frac{1}{2}-\epsilon_t)^2}.$$

Concluding the right inequality $(a \leq \boldsymbol{b} \leq \boldsymbol{c})$

Because both sides of $Z_t \leq e^{-2(\frac{1}{2}-\epsilon_t)^2}$ are positive, we can say that

$$\prod_{t=1}^{T} Z_t \;\leq\; \prod_{t=1}^{T} e^{-2(\frac{1}{2}-\epsilon_t)^2} = e^{-2\sum_{t=1}^{T}(\frac{1}{2}-\epsilon_t)^2}.$$

This concludes the "$b \leq c$" portion of the proof.

Combining everything

$$\text{training error} \;=\; \overbrace{\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{y_i \neq f_{boost}(x_i)\}}^{a} \;\leq\; \overbrace{\prod_{t=1}^{T} Z_t}^{b} \;\leq \overbrace{e^{-2\sum_{t=1}^{T}(\frac{1}{2}-\epsilon_t)^2}}^{c}.$$

We set out to prove "$a < c$" and we did so by using "b" as a stepping-stone.

TRAINING VS TESTING ERROR

Q: Driving the training error to zero leads one to ask, does boosting overfit?

A: Sometimes, but very often it doesn't!