Lecture 11 – Ensembles: Boosting

Prof. Maggie Makar

Announcements

- Midterm content:
  - Covers everything up to (but not including) neural networks
  - You will be allowed one **double sided 8.5 × 11** inch paper with notes prepared by **you**
- Midterm times
  - Main midterm: 3/6 at 7pm
  - Conflict midterm: 3/6 at 3pm
  - Accommodations: 3/6 at 7pm
  - Accommodations + conflict: TAC
- Midterm locations:
  - Main midterm: Look out for a Canvas post
  - All others, we have reached out
- HW2 due this Tuesday 2/20 at 10pm
- Quiz 6 due Friday 2/23 at 10pm

– Midterm review: 3/4
– Midterm sample: Before EOW

Outline

- Recap
  - Ensembles: bagging

- Continue bagging: why does it work?
- Ensembles: Boosting
  - Setup and high-level overview
  - The algorithm
  - Properties of AdaBoost
    - Properties of $\hat{\epsilon}_m, \alpha_m$
    - It minimizes the exponential loss
    - It has a boosting property
    - It keeps the variance in check
  - AdaBoost example

Majority rule notation

$y_i \in \{0,1\}$

$$\mu_m = \arg\max\left(\boxed{\sum_i (1-y_i)[\![\bar{x}_i \in R_m]\!]}, \boxed{\sum_i y_i[\![\bar{x}_i \in R_m]\!]}\right) = \text{argmax}\left(\underbrace{\#\text{ of examples in } R_m \text{ with } y_i=0}_{0}, \underbrace{\#\text{ of ex. in } R_m \text{ with } y_i=1}_{1}\right)$$

argmax $[a, b]$

position 0 position 1

$a > b \to 0$

$a < b \to 1$

$$\mu_m = \frac{1}{n_m}\sum_i y_i \geq 0.5$$

Ensemble methods

missclass ≤ 0.5

- **Idea:** Create a set of weak/base models whose individual decisions are combined in some way
- **Main advantage:** Reduces variance without increasing bias
- Describes a set of approaches that differ in training and combination methods
- **Two main types of ensembles**
  - Bagging
    - "Vanilla" bagging
    - Random Forests
  - Boosting (Adaboost)

Ensemble methods: bagging

DTs are typically high variance

Bagging = **B**ootstrap **agg**regat**ing**

Algorithm:

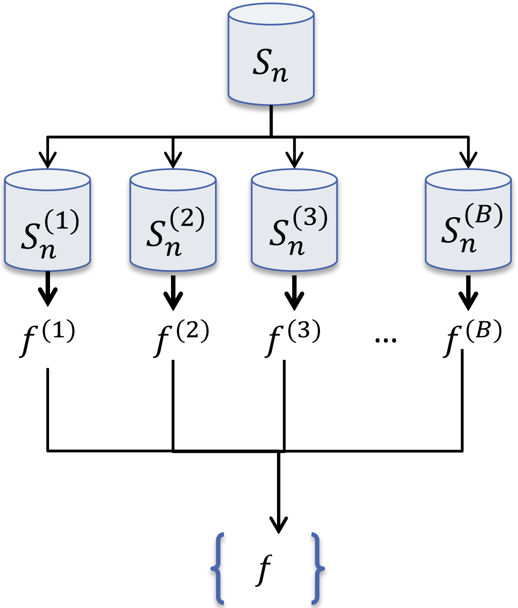
1. Sample n points B times with replacement
2. Build B decision trees using each of the B bootstrap replicates
3. Aggregate their prediction

$$f(\bar{x}) = \arg\max_{y} \sum_{b=1}^{B} [\![f^{(b)}(\bar{x}) = y]\!]$$

For binary predictions, with $y \in \{-1, +1\}$

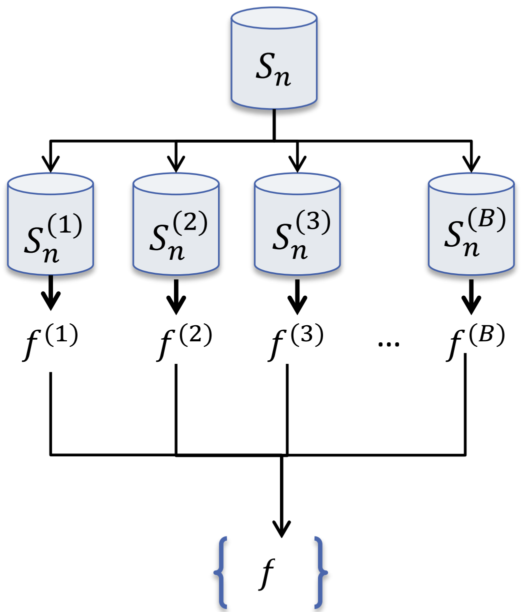
$$f(\bar{x}) = \operatorname{sign}\left(\sum_{b=1}^{B} f^{(b)}(\bar{x}) \right)$$

$= f(\bar{x})$



Ensemble methods: bagging

- Bagging = **B**ootstrap **agg**regat**ing**



Assumptions:

- Each decision tree has a misclassification rate better than 50%
- Classifiers are independent

If assumptions are satisfied:
As $B \to \infty$, misclassification rate $\to 0$

Why does bagging reduce variance?

$\in \mathbb{R}^{n \times d}$

- For a fixed test set $\bar{X}$, use σ^2 to denote the variance of a single decision tree $f^{(b)}(\bar{X})$

- Define $\tilde{f}(\bar{x}) = \frac{1}{B}\sum_{b=1}^{B} f^{(b)}(\bar{x})$ $f(\bar{x}) = \text{sign}(\tilde{f}(\bar{x}))$

Random forest
(1) Bagging
(2) sample $m < d$ features

- The variance of the bagged ensemble is:

$V(x) = \mathbb{E}[x^2]$
$V(ax) = \mathbb{E}[a^2x^2]$
$= a^2\,\mathbb{E}[x]$

$$
\begin{aligned}
\mathbb{V}(\tilde{f}) &= \mathbb{V}\left(\frac{1}{B}\sum_{b=1}^{B} f^{(b)}(\bar{X})\right) \quad \leftarrow \text{by definition.}\\
&= \frac{1}{B^2}\mathbb{V}\left(\sum_{b=1}^{B} f^{(b)}(\bar{X})\right) \quad \leftarrow \text{kick } \tfrac{1}{B} \text{ (constant) out}\\
&= \frac{1}{B^2}\left[\mathbb{V}(f^{(1)}(\bar{X})) + \ldots + \mathbb{V}(f^{(b)}(\bar{X})) + \ldots + \mathbb{V}(f^{(B)}(\bar{X}))\right] + \text{Cor}(\cdots)\\
&= \frac{1}{B^2}\left[B\sigma^2\right] = \frac{\sigma^2}{B} < \sigma^2
\end{aligned}
$$

Boosting

- Combining simple weak classifiers into a more complex/strong classifier
- Decrease bias (structural error) **and** variance (estimation error)
- **Sequentially** build classifiers, each one improves on the previous classifiers
- General form: weighted sum of weak classifier.

final classifier

$$\underbrace{h_M(\bar{x})} = \sum_{m=1}^{M} \alpha_m \underbrace{h(\bar{x}^{(i)}; \bar{\theta}_m)}$$

single weak classifier

weight assigned to the mth classifier.

- Different boosting algorithms differ on loss function, and how to pick α
- We'll study **Ada**ptive **Boost**ing (AdaBoost) by Freund and Schapire

Setup

- Training data:

$$S_n = \{\bar{x}^{(i)}, y^{(i)}\}_{i=1}^{n}, \bar{x} \in \mathbb{R}^d, y \in \{-1, +1\}$$

- A set of weak classifiers
  - Stumps with $\leq$ 50% misclassification rate

decision trees with depth 1

$$h(\bar{x}; \bar{\theta}_m) = \text{sign}(\theta_{1,m}(x_d - \theta_{0,m}))$$

$$\text{where } \bar{\theta}_m = [d_m, \theta_{0,m}, \theta_{1,m}]^\top$$

Split dimension

Split value

+/- direction

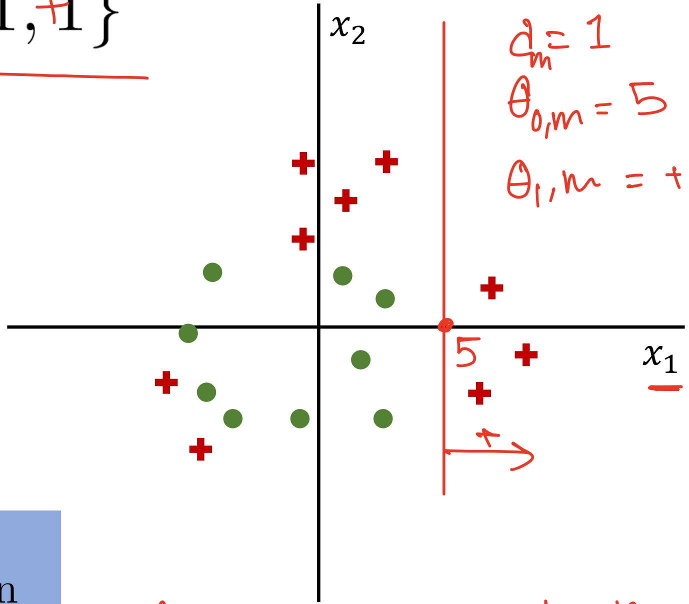
do I want x values that are greater than the split value $\theta_{0,m}$ to be +ve or -ve?

$\bar{\theta}_m = [1, 5, +1]^T$

$d_m = 1$

$\theta_{0,m} = 5$

$\theta_{1,m} = +1$

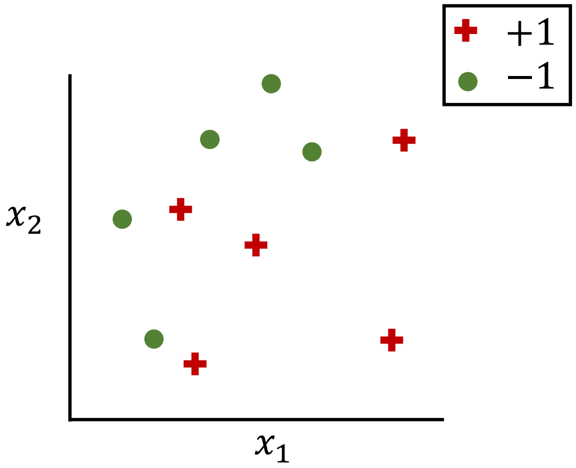


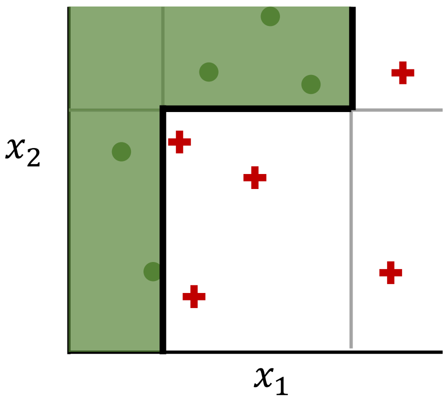
- Final classifier

$$h_M(\bar{x}) = \sum_{m=1}^{M} \alpha_m h(\bar{x}; \bar{\theta}_m)$$

$H_M(\bar{x}) = \text{sign}(h_M)$

AdaBoost

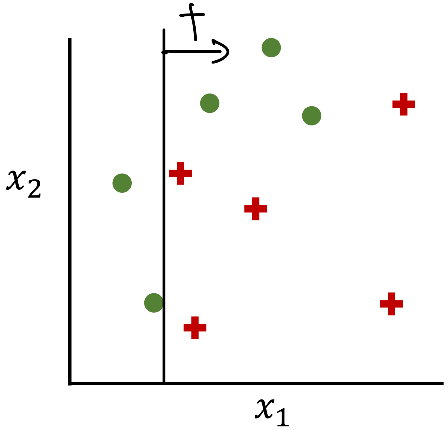




AdaBoost: high level idea

$m = 1$

$\hat{\epsilon}_1 = \frac{3}{10}$



$+ \quad +1$

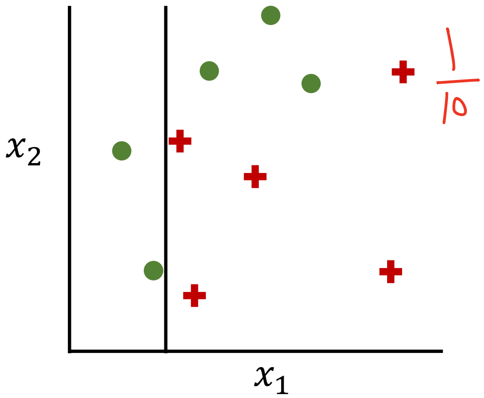
$\circ \quad -1$

$\frac{3}{10}$

AdaBoost: high level idea

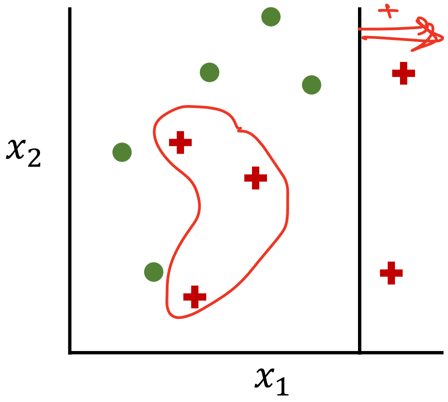
$m = 1$

$\hat{\epsilon}_1 = 0.3, \alpha_1 = 0.42$



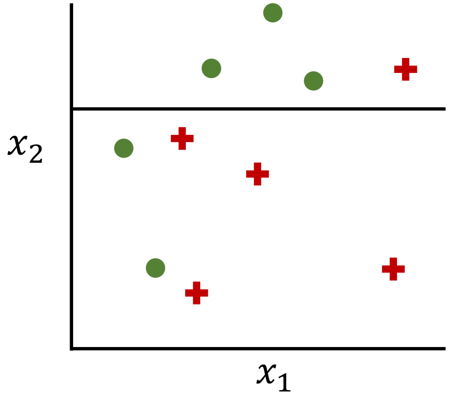
$m = 2$

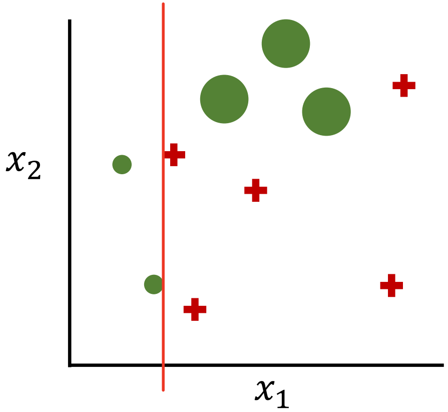
$\hat{\epsilon}_2 = 0.21, \alpha_2 = 0.65$

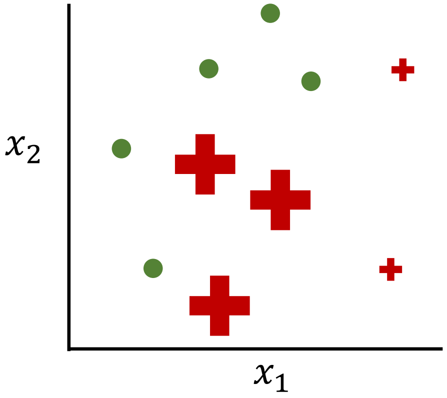


$m = 3$

$\hat{\epsilon}_3 = 0.14, \alpha_3 = 0.92$





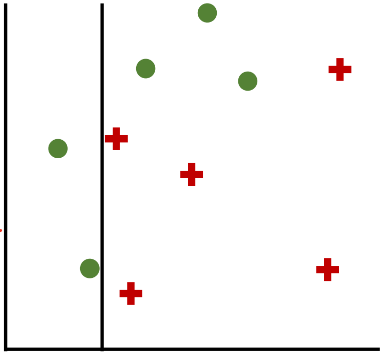


$$h_M(\bar{x}) = \sum_{m=1}^{M} \alpha_m h(\bar{x}; \bar{\theta}_m)$$

AdaBoost: high level idea

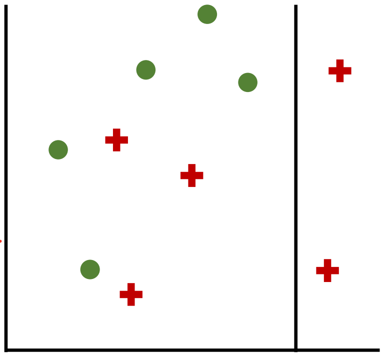
$h_M = \Big[\, \alpha_1$

0.42



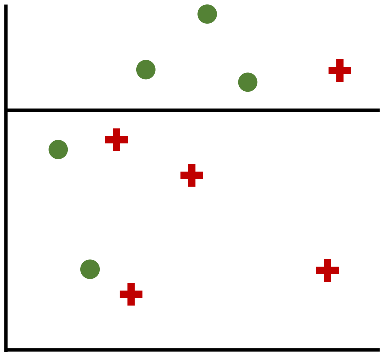
$+\alpha_2$

0.62



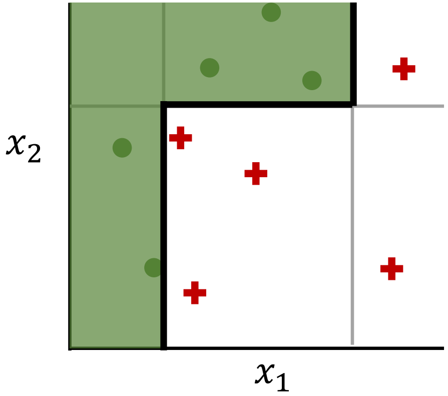
$+\alpha_3$

0.92



$\Big]$

$$H_M = \text{sign}(h_M) =$$



TL;DPA:

1. AdaBoost is an ensemble method that reduces variance without increasing bias

2. It works sequentially: at every iteration, the model tries to fix the mistakes of the previous models

AdaBoost: the algorithm

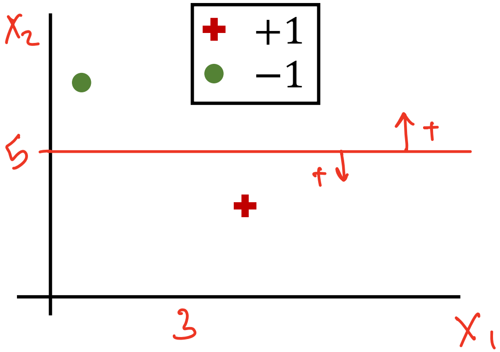
AdaBoost

1. Initialize the observation weights $\widetilde{w}_0^{(i)} = \frac{1}{n}$, for all $i \in [1 \ldots n]$
2. For $m = 1$ to M: via XV
   (a) Find: $\bar{\theta}_m = \arg\min_{\bar{\theta}} \sum_{i=1}^{n} \widetilde{w}_{m-1}^{(i)} [\![y^{(i)} \neq h(\bar{x}^{(i)}; \bar{\theta})]\!]$
   (b)
   (c)
   (d)
   (e)
3.

At initialization: the cost (aka weight) of misclassifying any point is the same

For every possible stump: evaluate the sum the weights of misclassified points

Find the stump that minimizes the total weights of misclassified points



$\bar{\theta}_1^{(1)} = [1, 3, +1]$

$\bar{\theta}_1^{(2)} = [1, 3, -1]$

$\bar{\theta}_1^{(3)} = [2, 5, +1]$

$\bar{\theta}_1^{(4)} = [2, 5, -1]$

AdaBoost: the algorithm

AdaBoost

1. Initialize the observation weights $\tilde{w}_0^{(i)} = \frac{1}{n}$, for all $i \in [1 \dots n]$
2. For $m = 1$ to M:
   (a) Find: $\bar{\theta}_m = \arg\min_{\bar{\theta}} \sum_{i=1}^{n} \tilde{w}_{m-1}^{(i)} [\![y^{(i)} \neq h(\bar{x}^{(i)}; \bar{\theta})]\!]$
   (b) Given $\bar{\theta}_m$, compute: $\hat{\epsilon}_m = \sum_{i=1}^{n} \tilde{w}_{m-1}^{(i)} [\![y^{(i)} \neq h(\bar{x}^{(i)}; \bar{\theta}_m)]\!]$
   (c) Compute $\alpha_m = \frac{1}{2} \ln \left(\frac{1-\hat{\epsilon}_m}{\hat{\epsilon}_m} \right)$. → miss class rate ≤ 0.5
   (d) Update un-normalized weights for all $i \in [1 \dots n]$:
   $$w_m^{(i)} \leftarrow \tilde{w}_{m-1}^{(i)} \cdot \exp\left[-y^{(i)} \alpha_m h(\bar{x}^{(i)}; \bar{\theta}_m) \right],$$
   $y^{(i)} \in \{-1, 1\}$ $\quad h(\bar{x}^{(i)}; \bar{\theta}_m) \in \{-1, 1\}$
   (e) Normalize weights to sum to 1:
   $$\tilde{w}_m^{(i)} \leftarrow \frac{w_m^{(i)}}{\sum_i w_m^{(i)}} := Z_m$$
   $$Z_m = \frac{1}{\sum_i w_m^{(i)}}$$
3. Output the final classifier: $h_M(\bar{\theta}) = \sum_{m=1}^{M} \alpha_m h(\bar{x}; \bar{\theta})$

Compute the resulting weighted misclassification rate

α_m which controls how much we "value" $h(\bar{x}, \bar{\theta}_m)$ is inversely related to $h(\bar{x}, \bar{\theta}_m)$'s error ≥ 0

If i is correctly classified:

$$w^{(i)} \rightarrow \tilde{w}_{m-1}^{(i)} \exp(-\alpha_m) \quad < 1$$

If i is incorrectly classified:

$$w^{(i)} \rightarrow \tilde{w}_{m-1}^{(i)} \exp(\alpha_m) \quad > 1$$

TL;DPA:

1. We went through the details of the AdaBoost algorithm

2. The stump built at each iteration is weighted by α_m, which is inversely related to the error of the stump

3. If a data point is misclassified at iteration m, it will have a higher weight at iteration $m+1$. And vice versa.

Properties of AdaBoost: $\hat{\epsilon}_m, \alpha_m$

Recall that:

$$\hat{\epsilon}_m = \sum_{i=1}^{n} \widetilde{w}_{m-1}^{(i)} [\![y^{(i)} \neq h(\bar{x}^{(i)}; \bar{\theta}_m)]\!]. \in [0, \tfrac{1}{2}]$$

weights are normalized

weak classifiers

$$\alpha_m = \frac{1}{2} \ln \left(\frac{1 - \hat{\epsilon}_m}{\hat{\epsilon}_m} \right).$$

Extreme case #1: if $\hat{\epsilon}_m = \frac{1}{2}$, $\alpha_m = 0$

Extreme case #2: if $\hat{\epsilon}_m = 0$, $\alpha_m = \infty$

Properties of AdaBoost: the boosting property

- $h(\bar{x};\bar{\theta}_m)$ is the best classifier given $\left\{\widetilde{w}_{m-1}^{(i)}\right\}_{i=1}^{n}$

- We then updated the weights to $\left\{\widetilde{w}_{m}^{(i)}\right\}_{i=1}^{n}$ to chose $h(\bar{x};\bar{\theta}_{m+1})$

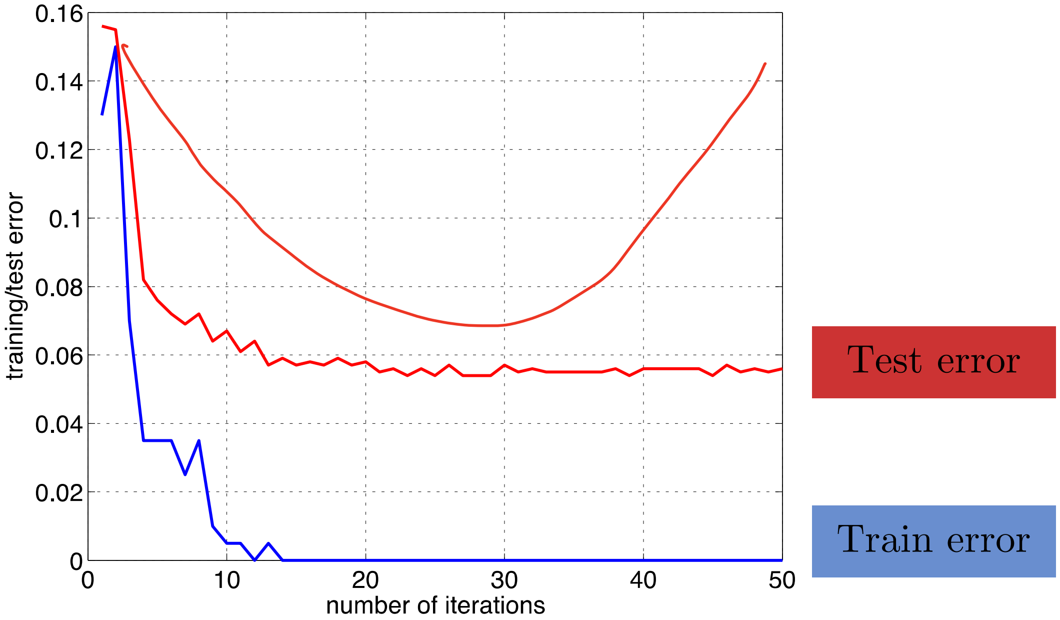
- How does $h(\bar{x};\bar{\theta}_m)$ perform given $\left\{\widetilde{w}_{m}^{(i)}\right\}_{i=1}^{n}$

- $$\sum_{i=1}^{n} \widetilde{w}_m^{(i)} [\![h(\bar{x};\bar{\theta}_m) \neq y^{(i)}]\!] = 0.5$$

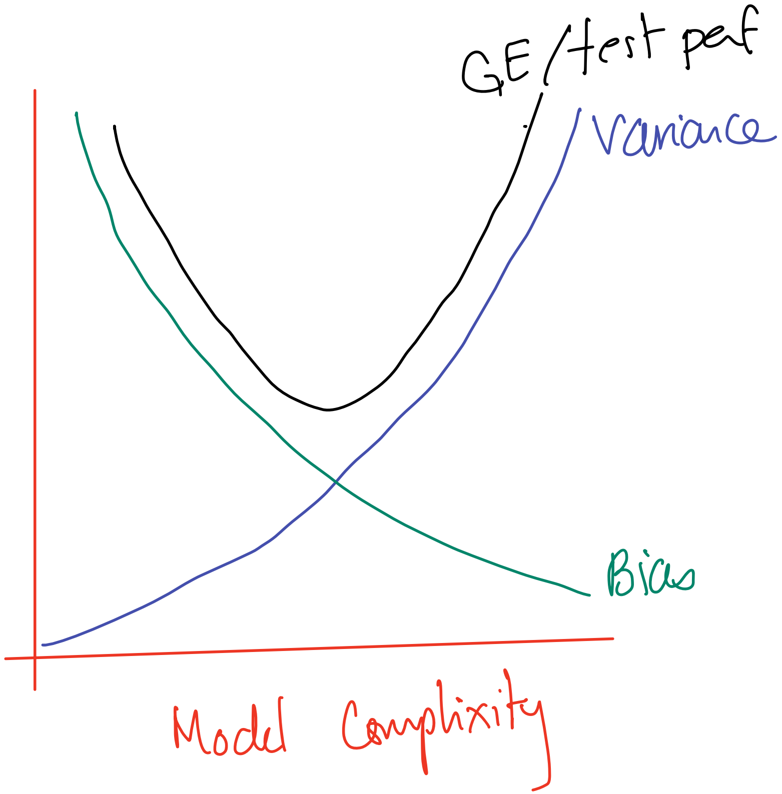
$$\hat{\epsilon}_m = \sum_{i=1}^{n} \tilde{w}_{m-1}^{(i)} [\![h(\bar{x};\bar{\theta}_m \neq y^{(i)}]\!]$$

Properties of AdaBoost: it keeps variance in check

$$\frac{\text{Model complexity} \uparrow}{\text{Margin} \uparrow}$$



Complexity.



Properties of AdaBoost: it minimizes the exponential loss

- Exponential loss in general: $\text{Loss}_{\exp}(z) = \exp(-z)$
- Exponential loss for $(\bar{x}, y)$: $R_n(h_M) = \exp(-y h_M(\bar{x}))$
- AdaBoost *greedily* minimizes the exponential loss

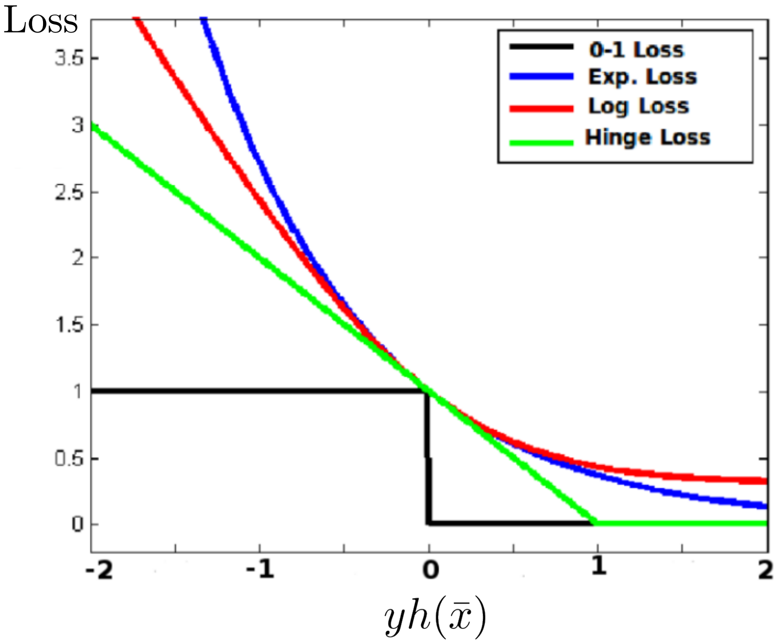
h_M $\quad$ h_m

$$R_n(\alpha_m, \bar{\theta}_m; \{\alpha_j\}_{j=1}^{m-1}, \{\bar{\theta}_j\}_{j=1}^{m-1}) =$$

$$= \frac{1}{n} \sum_i \exp\left[-y^{(i)} h_m(\bar{x}^{(i)})\right]$$

$$= \frac{1}{n} \sum_i \exp\left[-y^{(i)} \left[\alpha_m h(\bar{x}^{(i)}; \bar{\theta}_m) + \sum_{j=1}^{m-1} \alpha_j h(\bar{x}^{(i)}; \bar{\theta}_j)\right]\right]$$

$$\frac{\partial R_n}{\partial \bar{\theta}_m} \qquad \frac{\partial R_n}{\partial \alpha_m}$$



TL;DPA: Important properties of AdaBoost:

1. Really low accuracy stumps get tossed out, and perfect stumps get a influence of ∞ on the final prediction

2. The boosting property ensures that we never use the same stump twice *in a row*

3. AdaBoost keeps the variance of the model low without increasing bias

4. AdaBoost greedily minimizes the exponential loss (an upper bound on the 0-1 loss)