



Ensemble Methods: Boosting

(Part II of Ensembles)

Rishabh Iyer

University of Texas at Dallas

Bias / Var / Noise vs Train & test

$$\text{Test error} = \text{Bias}^2 + \text{Var} + \text{Noise}.$$

"True error"

$$\text{Var} \sim \text{Test} - \text{Train}$$

$$\text{Bias}^2 + \text{Noise} \sim \text{Train}$$
$$\text{Noise} \sim \text{Minimum Possible Error}$$

Last Time

- Variance reduction via bagging
 - Generate “new” training data sets by sampling with replacement from the empirical distribution
 - Learn a classifier for each of the newly sampled sets
 - Combine the classifiers for prediction
- Today: how to reduce bias for binary classification problems

Boosting

- How to translate rules of thumb (i.e., good heuristics) into good learning algorithms
- For example, if we are trying to classify email as spam or not spam, a good rule of thumb may be that emails containing “Nigerian prince” or “Viagara” are likely to be spam most of the time

Boosting

- Freund & Schapire
 - Theory for “weak learners” in late 80’s
- **Weak Learner**: performance on **any** training set is slightly better than chance prediction
- Intended to answer a theoretical question, not as a practical way to improve learning
 - Tested in mid 90’s using not-so-weak learners
 - Works anyway!

_ Gradient - Boosted DTs

Boosting

$$\frac{1}{N} \sum_{i=1}^N L(n_i, y_i, \theta)$$

weighted:
 $\sum_{i=1}^N w_i L(n_i, y_i, \theta)$ 

1. Weight all training samples equally $(\frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N})$

2. Train model on training set

3. Compute error of model on training set

4. Increase weights on training cases model gets wrong

5. Train new model on re-weighted training set

6. Re-compute errors on weighted training set

7. Increase weights again on cases model gets wrong

Repeat until tired

Final model: weighted prediction of each model

Boosting framework

for $t=1:T$

a) $h_t = \underset{h}{\text{minimize}}$

$$\sum_{i=1}^N w_i^t \ell(h(x_i))$$

h each h is a "weak" learner.

b) Compute training err on samples

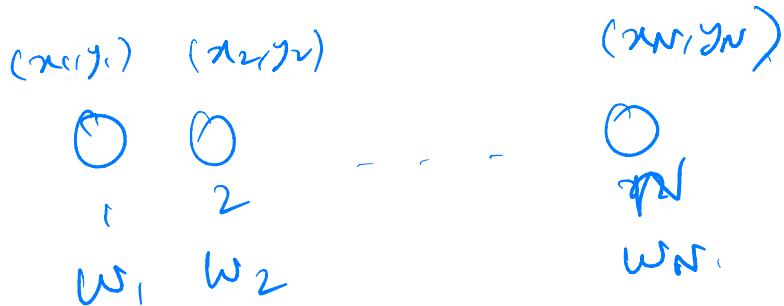
$$i=1:N$$

c) update weights using errors $\leftarrow w_i^{t+1}$

d) compute Δ_t

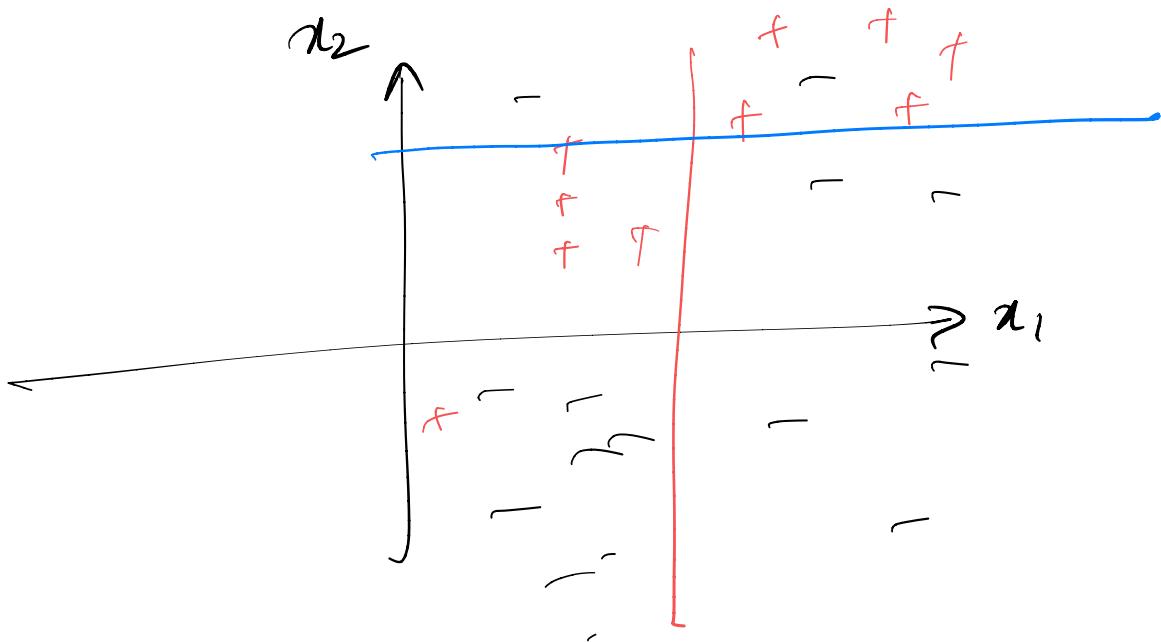
return $h = \text{sign} \left(\sum_{t=1}^T (\Delta_t h_t(x)) \right)$

Data Re-weighting

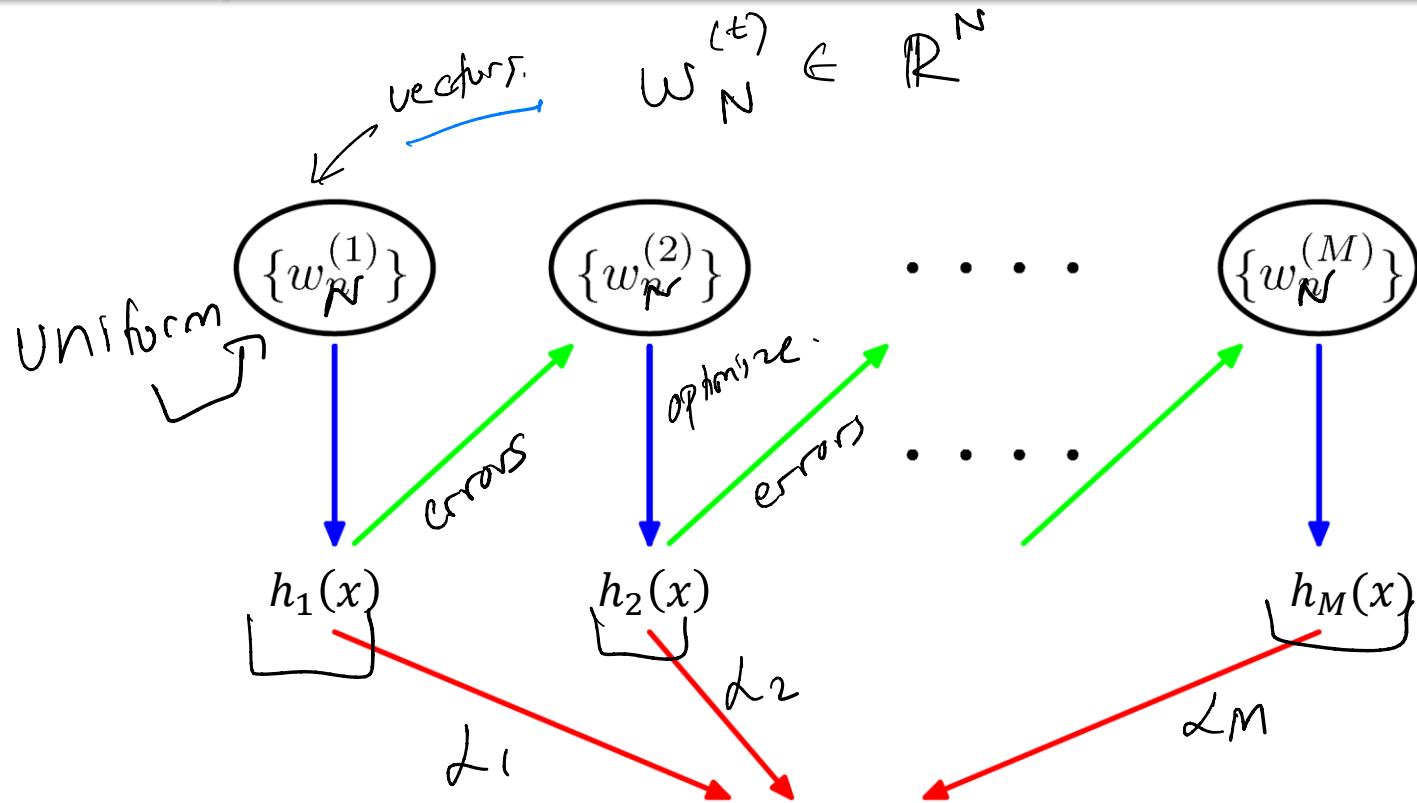


$$\min_{\theta} \sum_{i=1}^N w_i l(h_{\theta}(x_i), y_i)$$

Weak learners



Boosting: Graphical Illustration



AdaBoost

1. Initialize the data weights w_1, \dots, w_N for the first round as $w_1^{(1)}, \dots, w_N^{(1)} = \frac{1}{N}$
2. For $t = 1, \dots, T$
 - a) Select a classifier h_t for the T^{th} round by minimizing the weighted error

$$\epsilon_t = \sum_m w_m^{(t)} \mathbf{1}_{h_t(x^{(m)}) \neq y^{(m)}} \quad \text{How to phrase this?}$$

- b) Compute

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

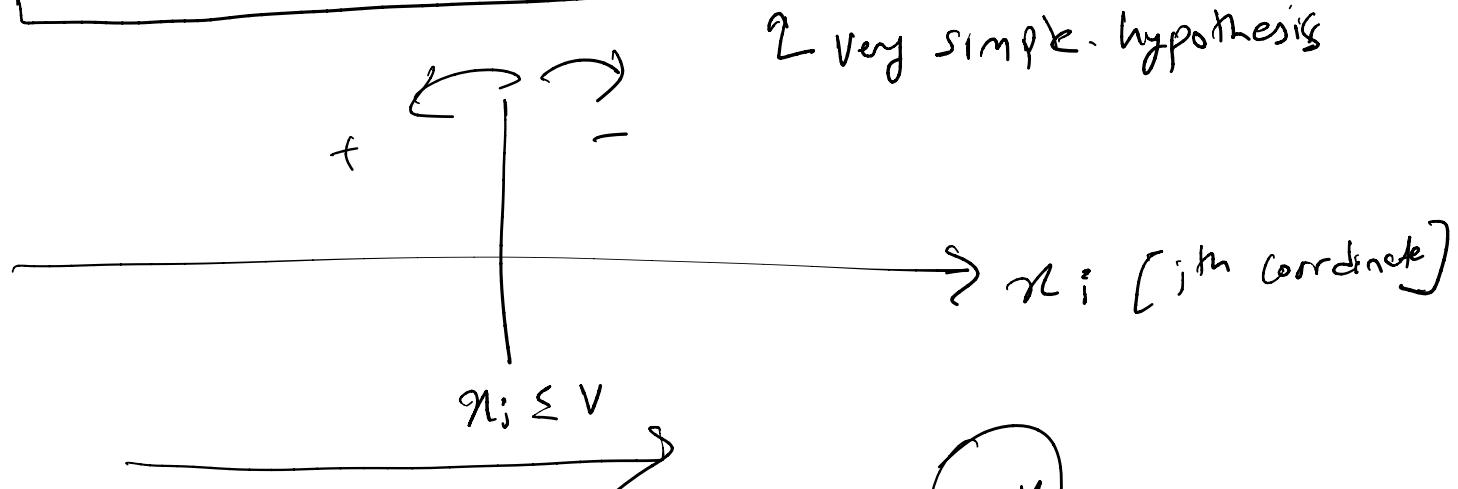
- c) Update the weights

$$w_m^{(t+1)} = \frac{w_m^{(t)} \exp(-y^{(m)} h_t(x^{(m)}) \alpha_t)}{2 \sqrt{\epsilon_t \cdot (1 - \epsilon_t)}}, \quad \forall m=1:N$$

(a) $h_t = \underset{h}{\operatorname{argmin}} \sum_{m=1}^N w_m^{(t)} \mathbf{1}_{h(x^{(m)}) \neq y^{(m)}}$

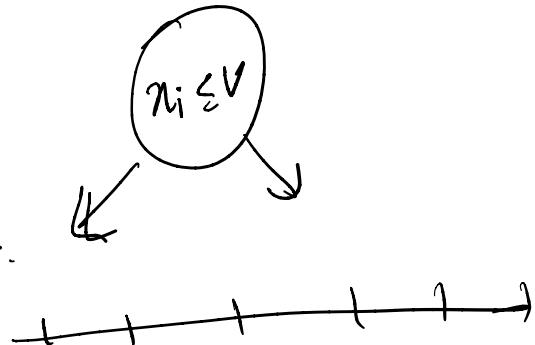
Solve the inner optimization

$$h_t = \operatorname{argmin}_h \sum_m w_m^{(t)} \mathbb{1}_{h(x^{(m)}) \neq y^{(m)}} \leftarrow$$



H coordinates x_i

- find best split V_i
- output split board with min error.



AdaBoost

1. Initialize the data weights w_1, \dots, w_M for the first round as $w_1^{(1)}, \dots, w_M^{(1)} = \frac{1}{M}$
2. For $t = 1, \dots, T$
 - a) Select a classifier h_t for the T^{th} round by minimizing the weighted error

$$\epsilon_t = \sum_m w_m^{(t)} \mathbf{1}_{h_t(x^{(m)}) \neq y^{(m)}} \quad \leftarrow \text{Inner Optimization.}$$

- b) Compute

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

Weighted number
of incorrect
classifications of
the t^{th} classifier

- c) Update the weights

$$w_m^{(t+1)} = \frac{w_m^{(t)} \exp(-y^{(m)} h_t(x^{(m)}) \alpha_t)}{2 \sqrt{\epsilon_t \cdot (1 - \epsilon_t)}}$$

AdaBoost

1. Initialize the data weights w_1, \dots, w_M for the first round as $w_1^{(1)}, \dots, w_M^{(1)} = \frac{1}{M}$
2. For $t = 1, \dots, T$
 - a) Select a classifier h_t for the T^{th} round by minimizing the weighted error

$$\epsilon_t = \sum_m w_m^{(t)} \mathbf{1}_{h_t(x^{(m)}) \neq y^{(m)}}$$

- b) Compute

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

$\epsilon_t \rightarrow 0$
 $\alpha_t \rightarrow \infty$

- c) Update the weights

$$w_m^{(t+1)} = \frac{w_m^{(t)} \exp(-y^{(m)} h_t(x^{(m)}) \alpha_t)}{2 \sqrt{\epsilon_t \cdot (1 - \epsilon_t)}} \rightarrow 0$$

↑

AdaBoost

1. Initialize the data weights w_1, \dots, w_M for the first round as $w_1^{(1)}, \dots, w_M^{(1)} = \frac{1}{M}$
2. For $t = 1, \dots, T$
 - a) Select a classifier h_t for the T^{th} round by minimizing the weighted error

$$\epsilon_t = \sum_m w_m^{(t)} \mathbf{1}_{\underline{\underline{h_t(x^{(m)})}} \neq y^{(m)}}$$

b) Compute

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

$\epsilon_t \rightarrow .5$
 $\alpha_t \rightarrow 0$

- c) Update the weights

$$w_m^{(t+1)} = \frac{w_m^{(t)} \exp(-y^{(m)} h_t(x^{(m)}) \alpha_t)}{2 \sqrt{\epsilon_t \cdot (1 - \epsilon_t)}}$$

AdaBoost

1. Initialize the data weights w_1, \dots, w_M for the first round as $w_1^{(1)}, \dots, w_M^{(1)} = \frac{1}{M}$
2. For $t = 1, \dots, T$
 - a) Select a classifier h_t for the T^{th} round by minimizing the weighted error

$$\epsilon_t = \sum_m w_m^{(t)} \mathbf{1}_{h_t(x^{(m)}) \neq y^{(m)}}$$

- b) Compute

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

Large error
 $\epsilon_t \rightarrow 1$
 $\alpha_t \rightarrow -\infty$

- c) Update the weights

$$w_m^{(t+1)} = \frac{w_m^{(t)} \exp(-y^{(m)} h_t(x^{(m)}) \alpha_t)}{2 \sqrt{\epsilon_t \cdot (1 - \epsilon_t)}}$$

AdaBoost

1. Initialize the data weights w_1, \dots, w_M for the first round as $w_1^{(1)}, \dots, w_M^{(1)} = \frac{1}{M}$
2. For $t = 1, \dots, T$

a) Select a classifier h_t for the T^{th} round by minimizing the weighted error

$$\epsilon_t = \sum_m w_m^{(t)} \mathbf{1}_{h_t(x^{(m)}) \neq y^{(m)}} \quad \leftarrow$$

b) Compute

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

c) Update the weights

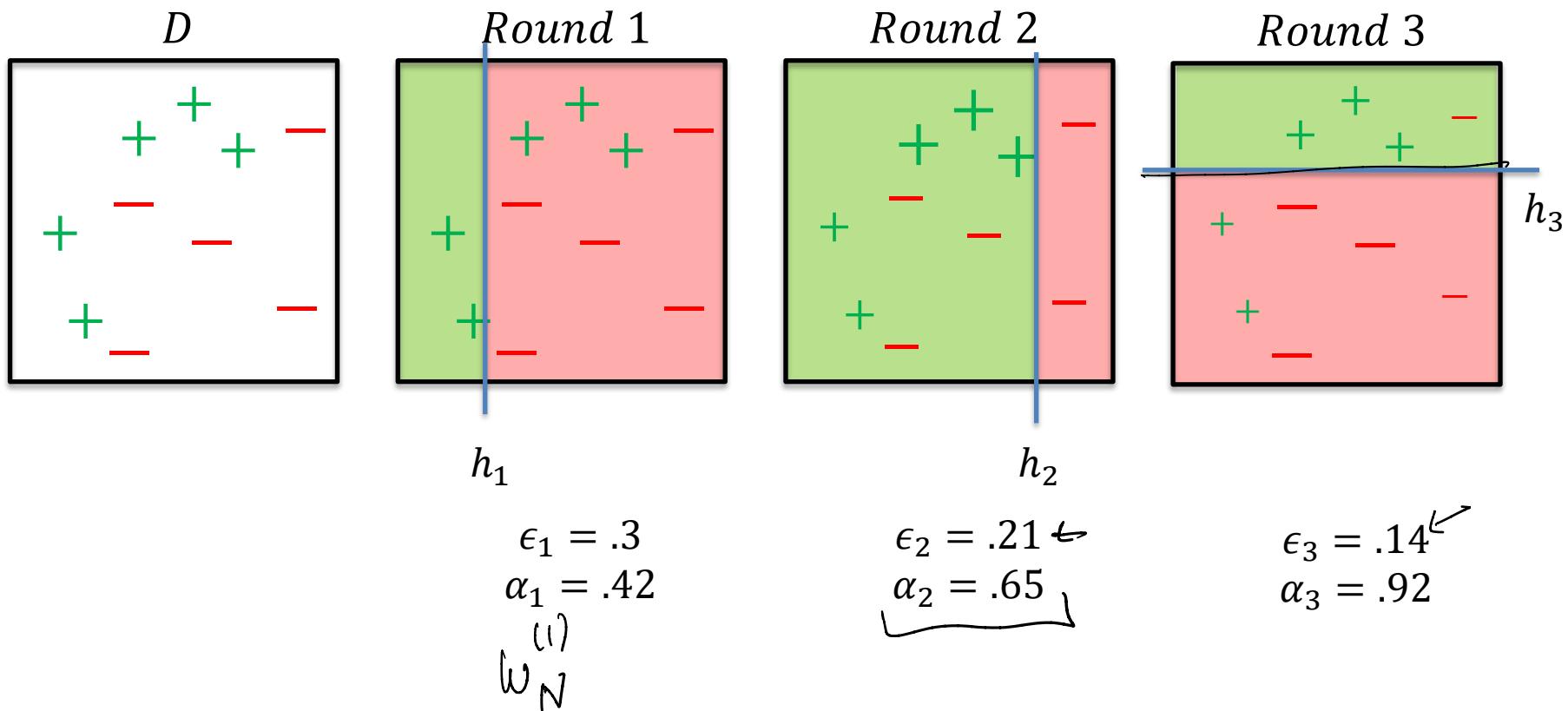
$$w_m^{(t+1)} = \frac{w_m^{(t)} \exp(-y^{(m)} h_t(x^{(m)}) \alpha_t)}{2\sqrt{\epsilon_t \cdot (1 - \epsilon_t)}} \quad \leftarrow$$

Normalization
constant

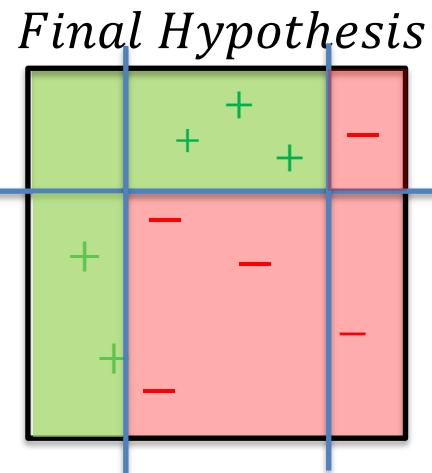
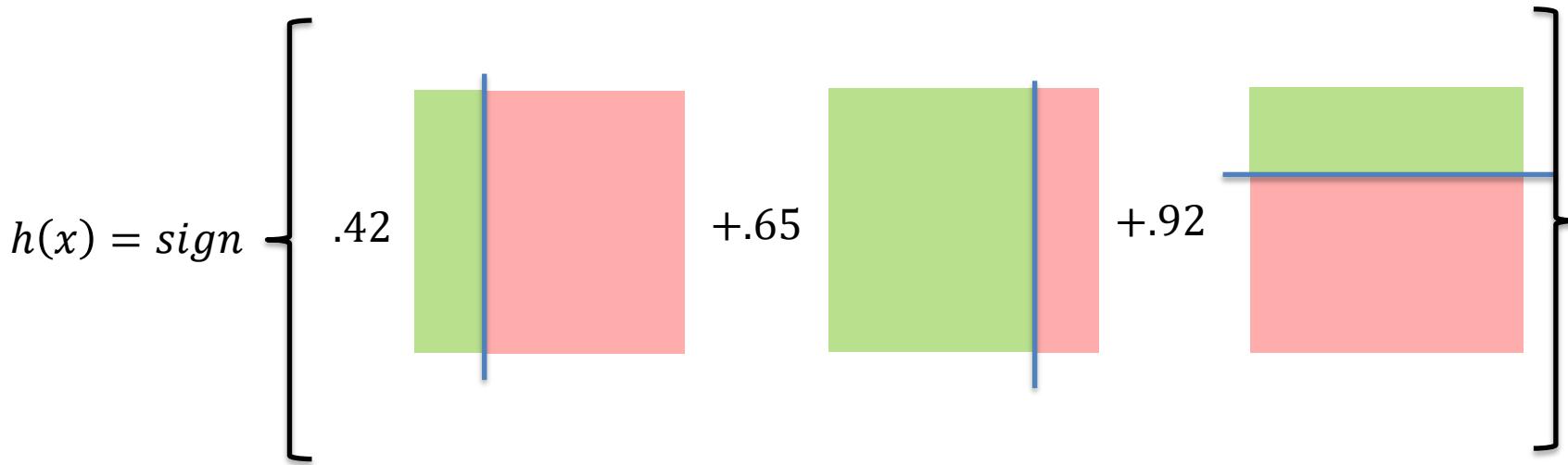
AdaBoost \Rightarrow Iterative optimization problem.

Example

- Consider a classification problem where vertical and horizontal lines (and their corresponding half spaces) are the weak learners



Final Hypothesis



Boosting

Theorem: Let $Z_t = 2\sqrt{\epsilon_t \cdot (1 - \epsilon_t)}$ and $\gamma_t = \frac{1}{2} - \epsilon_t$.

$$\frac{1}{M} \sum_m 1_{h(x^{(m)}) \neq y^{(m)}} \leq \prod_{t=1}^T Z_t = \prod_{t=1}^T \sqrt{1 - 4\gamma_t^2}$$

$\underbrace{\hspace{10em}}_{\text{LHS}}$ $\underbrace{\hspace{10em}}_{Z_t \leq 1}$

So, even if all of the γ 's are small positive numbers (i.e., can always find a weak learner), the training error goes to zero as T increases

$$Z_t \leq 1$$

$\underbrace{\hspace{10em}}_{\text{RHS}}$

1
—
2

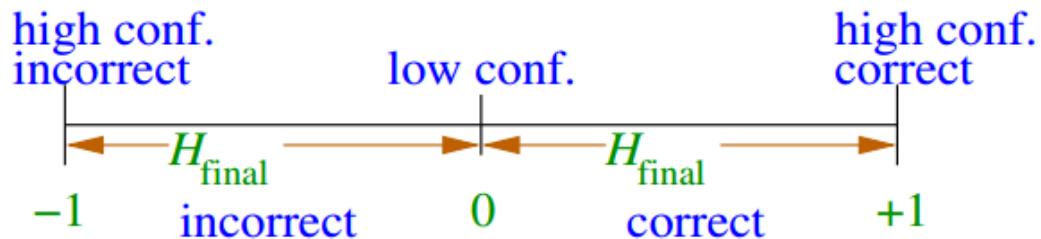


Margins & Boosting

- We can see that training error goes down, but what about test error?
 - That is, does boosting help us generalize better?
- To answer this question, we need to look at how confident we are in our predictions
 - How can we measure this?

Margins & Boosting

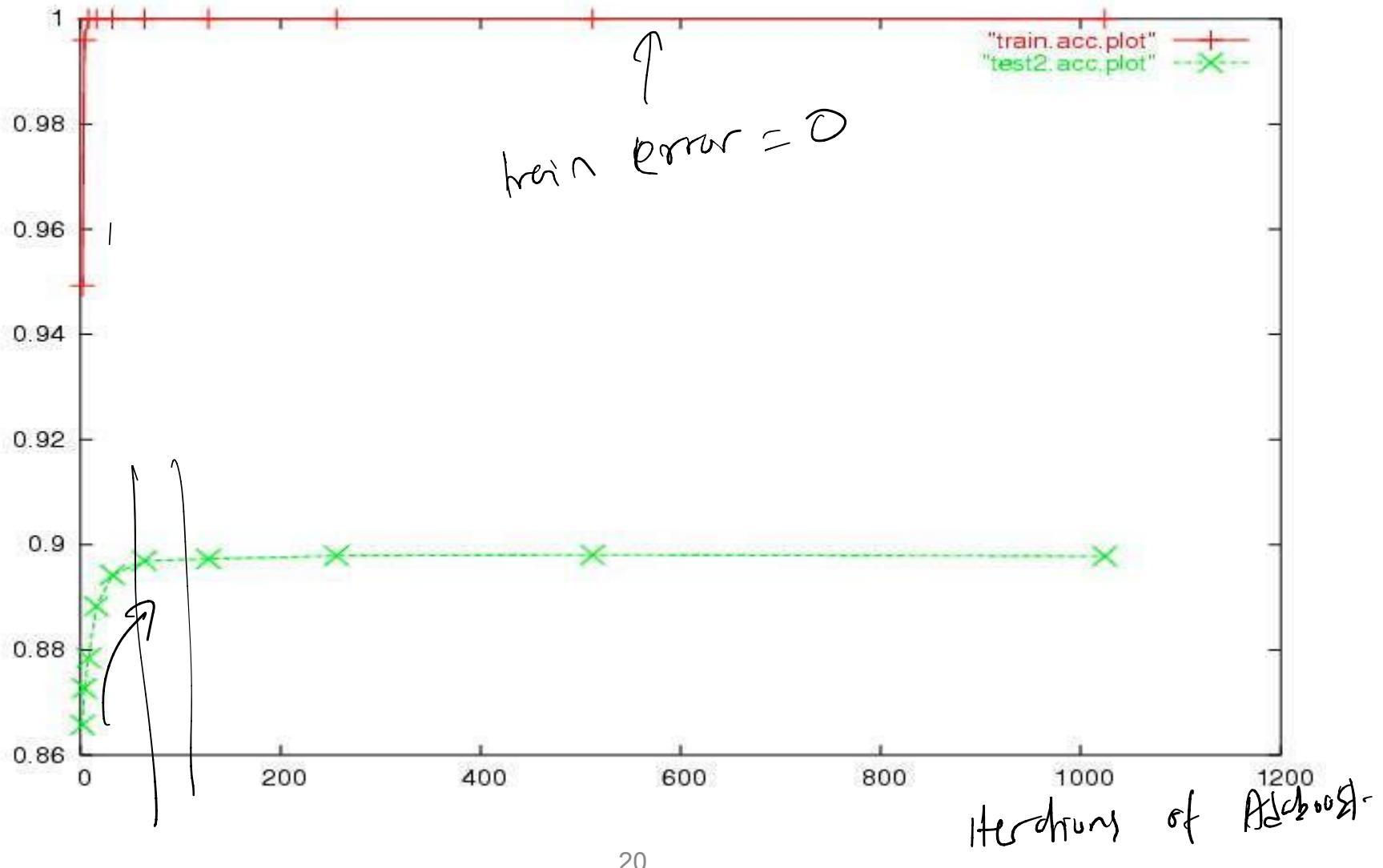
- We can see that training error goes down, but what about test error?
 - That is, does boosting help us generalize better?
- To answer this question, we need to look at how confident we are in our predictions
 - Margins!



Margins & Boosting

- Intuition: larger margins lead to better generalization (same as SVMs)
- Theorem: with high probability, boosting increases the size of the margins
 - Note: boosting does **NOT** maximize the margin, so it could still result in poor generalization performance

Boosting Performance



Boosting as Optimization

- AdaBoost can be interpreted as a coordinate descent method for a specific loss function! \leftarrow Exponential Loss
- Let $\{h_1, \dots, h_T\}$ be the set of all weak learners

$$h_1, h_2, \dots, h_T$$
- Exponential loss

$$\ell(\underbrace{\alpha_1, \dots, \alpha_T}_{\text{Factors}}) = \sum_m \exp \left(-y^{(m)} \cdot \sum_{t=1}^T \underbrace{\alpha_t h_t(x^{(m)})}_{\text{Factors}} \right)$$

- Convex in α_t
- AdaBoost minimizes this exponential loss

Coordinate Descent

- Minimize the loss with respect to a single component of α , let's pick $\alpha_{t'}$

$$\begin{aligned}
 \frac{d\ell}{d\alpha_{t'}} &= - \sum_m y^{(m)} h_{t'}(x^{(m)}) \exp\left(-y^{(m)} \cdot \sum_t \alpha_t h_t(x^{(m)})\right) \\
 &= \sum_{m: h_{t'}(x^{(m)}) = y^{(m)}} -\exp(-\alpha_{t'}) \exp\left(-y^{(m)} \cdot \sum_{t \neq t'} \alpha_t h_t(x^{(m)})\right) \\
 &\quad + \sum_{m: h_{t'}(x^{(m)}) \neq y^{(m)}} \exp(\alpha_{t'}) \exp\left(-y^{(m)} \cdot \sum_{t \neq t'} \alpha_t h_t(x^{(m)})\right) \\
 &= 0
 \end{aligned}$$

Coordinate Descent

- Solving for $\alpha_{t'}$

$$\alpha_{t'} = \frac{1}{2} \ln \frac{\sum_{m: h_{t'}(x^{(m)}) = y^{(m)}} \exp(-y^{(m)} \cdot \sum_{t \neq t'} \alpha_t h_t(x^{(m)}))}{\sum_{m: h_{t'}(x^{(m)}) \neq y^{(m)}} \exp(-y^{(m)} \cdot \sum_{t \neq t'} \alpha_t h_t(x^{(m)}))}$$

- This is similar to the adaBoost update!
 - The only difference is that adaBoost tells us in which **order** we should update the variables

Forward Stage wise Adaptive Modeling

Initialize $f_0(n_i) = 0$

For $m = 1$ to M

Compute $(\lambda_m, \gamma_m) = \underset{\lambda, \gamma}{\operatorname{argmin}} \sum_{i=1}^N L(y_i, f_{m-1}(n_i) + \lambda h(n_i, \gamma))$

→ $h(n_i, \gamma)$ is the parameterized hypothesis function

$$\rightarrow f_m(n_i) = f_{m-1}(n_i) + \lambda_m h(n_i, \gamma_m)$$

Add boost ⇒ $L(y, f(x)) = \exp(-y f(x))$

* Iterative Boosting Framework!

Inner - Optimization Problem

$$(\lambda_m, \gamma_m) = \underset{\lambda, \gamma}{\operatorname{argmax}} \sum_{i=1}^N w_i^{(m)} \exp(-\beta y_i [f_{m-1}(x_i) + \dots + h(x_i)])$$

where $w_i^{(m)} = \exp(-y_i f_{m-1}(x_i))$

$$f_{m-1}(x) = \underset{h}{\operatorname{argmin}} \sum_{i=1}^N w_i^{(m)} \mathbb{1}_{y_i \neq h(x)}$$

$$\lambda_m = \frac{1}{2} \log \left[\frac{1 - \text{err}_m}{\text{err}_m} \right]$$

"Hastie (ESL) Page 344"

AdaBoost as Optimization

- Could derive an adaBoost algorithm for other types of loss functions!
- Important to note
 - Exponential loss is convex, but not strict (may have multiple global optima)
 - In practice, adaBoost can perform quite differently than other methods for minimizing this loss (e.g., gradient descent)

Boosting in Practice

- Our description of the algorithm assumed that a set of possible hypotheses was given
 - In practice, the set of hypotheses can be built as the algorithm progress
- Example: build new decision tree at each iteration for the data set in which the \underline{m}^{th} example has weight $w_m^{(t)}$
 - When computing information gain, compute the empirical probabilities using the weights

Boosting vs. Bagging

- Bagging doesn't work well with stable models *- Fows on variance*
 - Boosting might still help → *Boosting improves Var & bias*
- Boosting might hurt performance on noisy datasets
 - Bagging doesn't have this problem
- On average, boosting improves classification accuracy more than bagging, but it is also more common for boosting to hurt performance
- Bagging is easier to parallelize
- Both ensemble methods have added overhead required to train multiple classifiers

Overhead to deploy multiple models

Boosting Beyond Binary Classification



- Slightly more complicated
 - Want to select weak learners that are better than random guessing, but there are many different ways to do better than random
 - A hypothesis space is boostable if there exists a baseline measure, that is slightly better than random, such that you can always find a hypothesis that outperforms the baseline
 - Can be boostable with respect to some baselines but not others

Additional Reading

- <http://cs229.stanford.edu/extr-notes/boosting.pdf>
- <https://web.stanford.edu/~hastie/Papers/buehlmann.pdf>
- <http://web.stanford.edu/~hastie/TALKS/boost.pdf>
- <https://cseweb.ucsd.edu/~yfreund/papers/IntroToBoosting.pdf>

"ESL" \rightarrow Boosting

Specific $L = \text{Squared Loss}$

Gradient Boosting

$$L(y, f(x)) = \frac{1}{2} \sum_{i=1}^N [y_i - f(x_i)]^2$$

Initialize $F_0(x_i) = 0, \forall i$

for $m = 1 : M$

Compute residuals $y'_i = y_i - F_{m-1}(x_i)$

Train hypothesis on $\left(x_i, y'_i\right), \forall i = 1 : N$ $\leftarrow h_m(x)$

$$F_m(x) := F_{m-1}(x) + \lambda m h_m(x)$$

Basic "Gradient Boosting Decision Trees"

$$f_{m-1}(x) = \sum_{j=1}^{m-1} h_j(x_j)$$

General Gradient Boosting?

Given any loss function $L(y, F(x))$

Initialize $F_0(x) = 0$

for $m = 1 : M$

compute $y_i^l = -\frac{\partial L}{\partial F} \Big|_{F(x_i)}$

Train $h_m(x)$ on Dataset

$$(x_i, y_i^l), i = 1 \dots N$$

$$F_m(x) = F_{m-1}(x) + \lambda_m h_m(x)$$

Differences between Adaboost & GradBoost

- ① weighted loss v/s modifying labels
 - In adaboost, we minimize weighted loss over data.
 - In grad boost, we modify labels to take residuals into account
- ② Weighing the sum of hypotheses
 - In adaboost, we obtain α_m to weigh hypotheses
 - In grad boost, α_m is a hyperparameter (often set to 1)