

Boosting Algorithms
Weak and Strong Learning

PAC Learning model

- Some distribution D over domain X
- Examples: $\langle x, c^*(x) \rangle$
 - c^* is the target function
- Goal:
 - With high probability $(1-\delta)$
 - find h in H such that
 - $error(h, c) < \epsilon$
- δ and ϵ are arbitrarily small.

Weak Learning - Confidence

- Assume we are given a learning algorithm
 - with confidence $\delta = \frac{1}{2}$
 - but arbitrary small accuracy $\epsilon > 0$
- Can we boost the confidence?
 - How?!

BoostConfidence Algorithm

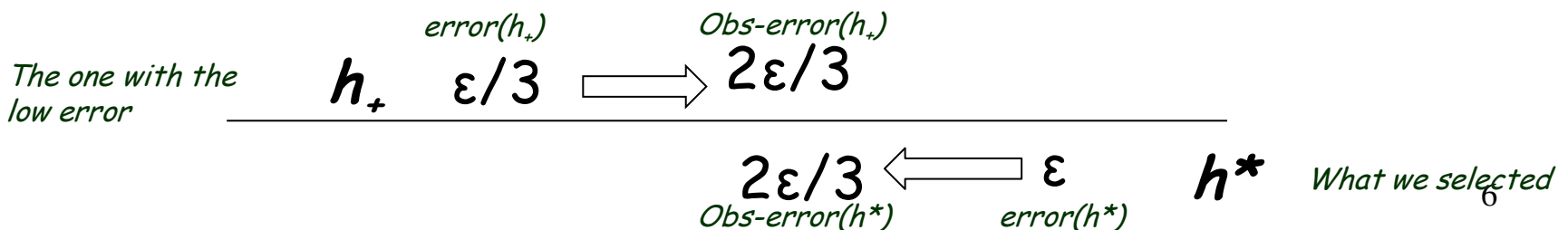
- **Input:** Algorithm \mathbf{A} and parameter δ
- Create $k = \log(2/\delta)$ independent problems
 - Sample S_i for i -th copy
 - Run Algorithm \mathbf{A} on S_i with accuracy $\varepsilon/3$
 - Let h_i be the hypothesis that \mathbf{A} outputs on S_i
- Take a new sample S of size $m = \varepsilon^{-2} \log(2k/\delta)$
 - Return the best hypothesis h_i on S
 - call it h^*

BoostConfidence Analysis S_i

- Fix copy i :
 - The probability that $error(h_i) < \epsilon/3$ at least $1/2$
- The probability that
 - some h_i has $error(h_i) < \epsilon/3$
 - at least $1 - 2^{-k} = 1 - \delta/2$
 - Holds for $k = \log(2/\delta)$
- Assume this holds!
 - Namely, some h_i has $error(h_i) < \epsilon/3$
 - denote it by h_+

Boost Confidence Analysis S

- With probability at least $1 - \delta/2$
 - for every h_i : $|error(h_i) - obst-error(h_i)| \leq \epsilon/3$
 - Chernoff bound using $m = \epsilon^{-2} \log(2k/\delta)$
- Assume this holds!
- Together, with probability at least $1 - \delta$
 - $error(h^*) \leq \epsilon$



Weak Learning - Accuracy

- Assume: $error(h,c) < \frac{1}{2} - \gamma$
- The parameter $\gamma > 0$ is small
 - constant
- Intuitively: a much easier task
- Question:
 - Assume \mathbf{C} is weak learnable,
 - Is \mathbf{C} PAC (strong) learnable?

Weak Learning - Definition

Weak Learning

- Algo A weak learns C using H if
- exists $\gamma > 0$
- for all c in C
- **for any distribution D**
- for all $\delta > 1/2$
- Outputs h in H such that
 - with prob $1 - \delta$
 - $error(c, h) < 1/2 - \gamma$

Strong Learning

- Algo A strong learns C using H if
- for all $\epsilon > 0$
- for all c in C
- for any distribution D
- for all $\delta > 1/2$
- Outputs h in H such that
 - with prob $1 - \delta$
 - $error(c, h) < \epsilon$

Weak learning – definition

- Why do we need **ANY** distribution
- Example:
 - Consider the following distribution over bits
 - if $x_1=x_2=0$ then $c^*(x)=$ some hard function
 - otherwise $c^*(x)=0$
- Uniform Distribution
 - Predicting random for $x_1=x_2=0$, 0 elsewhere will be correct 87.5%
 - Getting above that is impossible.
- The hard distribution:
 - Sample from the set where $x_1=x_2=0$

Three Weak Learners

- One weak learner
 - only one thing to do !
- Two weak learners
 - what to do if they disagree?
- Three weak learners
 - Can we improve accuracy?

- Example

x	- y	x	- y
11110	- 1	10111	- 0
11110	- 1	01101	- 0
10011	- 1	11011	- 0
01001	- 1	01100	- 0
10001	- 1	00000	- 0

Weak learners: single bits

Three weak learners

- First weak learner
 - use the distribution D
 - get h_1
- Second weak learner
 - How can we force new h ?
 - Set D_2 s.t.
 - h_1 has error $1/2$
 - Get h_2
 - why $h_2 \neq h_1$?
- Third weak learner
 - What are “interesting inputs”?
 - $h_2(x) \neq h_1(x)$
 - let D_3 be such inputs
 - get h_3
- How will we predict?
 - If $h_2(x) \neq h_1(x)$ using $h_3(x)$
 - Else $h_2(x)$ (or $h_1(x)$)
 - **majority**

3 Weak Learners - Performance

- Define D_2 and D_3
- Dist D_2 :
 - Select random $b \in \{0,1\}$
 - If $b=0$
 - Sample x from D_1 until $h_1(x) = c^*(x)$
 - Else ($b=1$)
 - Sample x from D_1 until $h_1(x) \neq c^*(x)$
- Dist D_3
 - if $h_1(x) \neq h_2(x)$ sample

- Formally
$$D_2(x) = \begin{cases} \frac{0.5}{1-p} D_1(x) & \text{if } h_1(x) = c^*(x) \\ \frac{0.5}{p} D_1(x) & \text{if } h_1(x) \neq c^*(x) \end{cases}$$
$$p = \Pr[h_1(x) \neq c^*(x)]$$

$$D_3(x) = \begin{cases} \frac{D_1(x)}{Z} & \text{if } h_1(x) \neq h_2(x) \\ 0 & \text{if } h_1(x) = h_2(x) \end{cases}$$

$$Z = \Pr[h_1(x) \neq h_2(x)]$$

3 Weak Learners - Analysis

- Assume all WL have error rate $p = \frac{1}{2} - \gamma$
- If all errors are independent, the error of the majority is
- ***Error = $3p^2(1-p) + p^3 = 3p^2 - 2p^3$***
- We show next that this holds even if the errors are dependent

3 Weak Learners - Analysis

- Assume all WL are $p = 1/2 - \gamma$

$$D_2(x) = \begin{cases} \frac{0.5}{1-p} D_1(x) & \text{if } h_1(x) = c^*(x) \\ \frac{0.5}{p} D_1(x) & \text{if } h_1(x) \neq c^*(x) \end{cases}$$

$$p = \Pr[h_1(x) \neq c^*(x)]$$

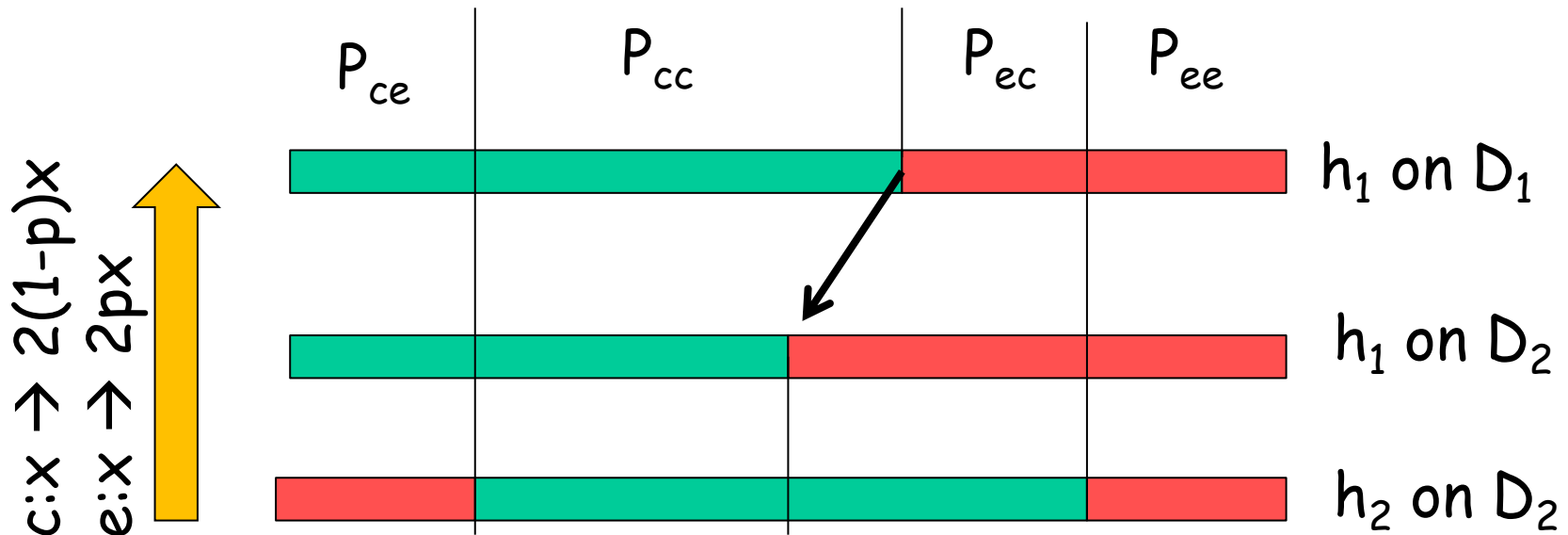
- To convert from a probability in D2 to D1

Correct in h_1 : $x \rightarrow 2(1-p)x$

Error in h_1 : $x \rightarrow 2px$

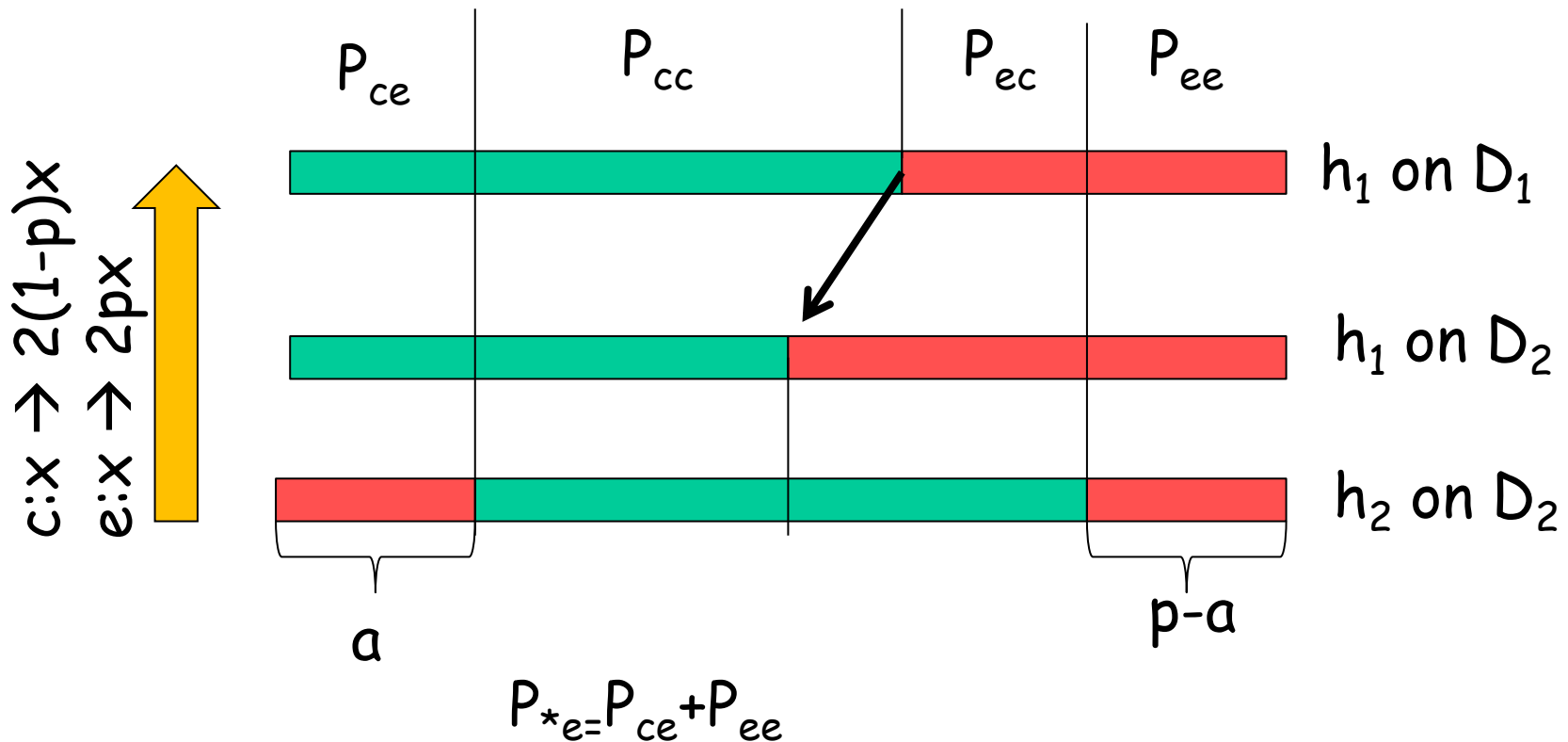
3 Weak Learners - Analysis

- Assume all WL are $p = 1/2 - \gamma$



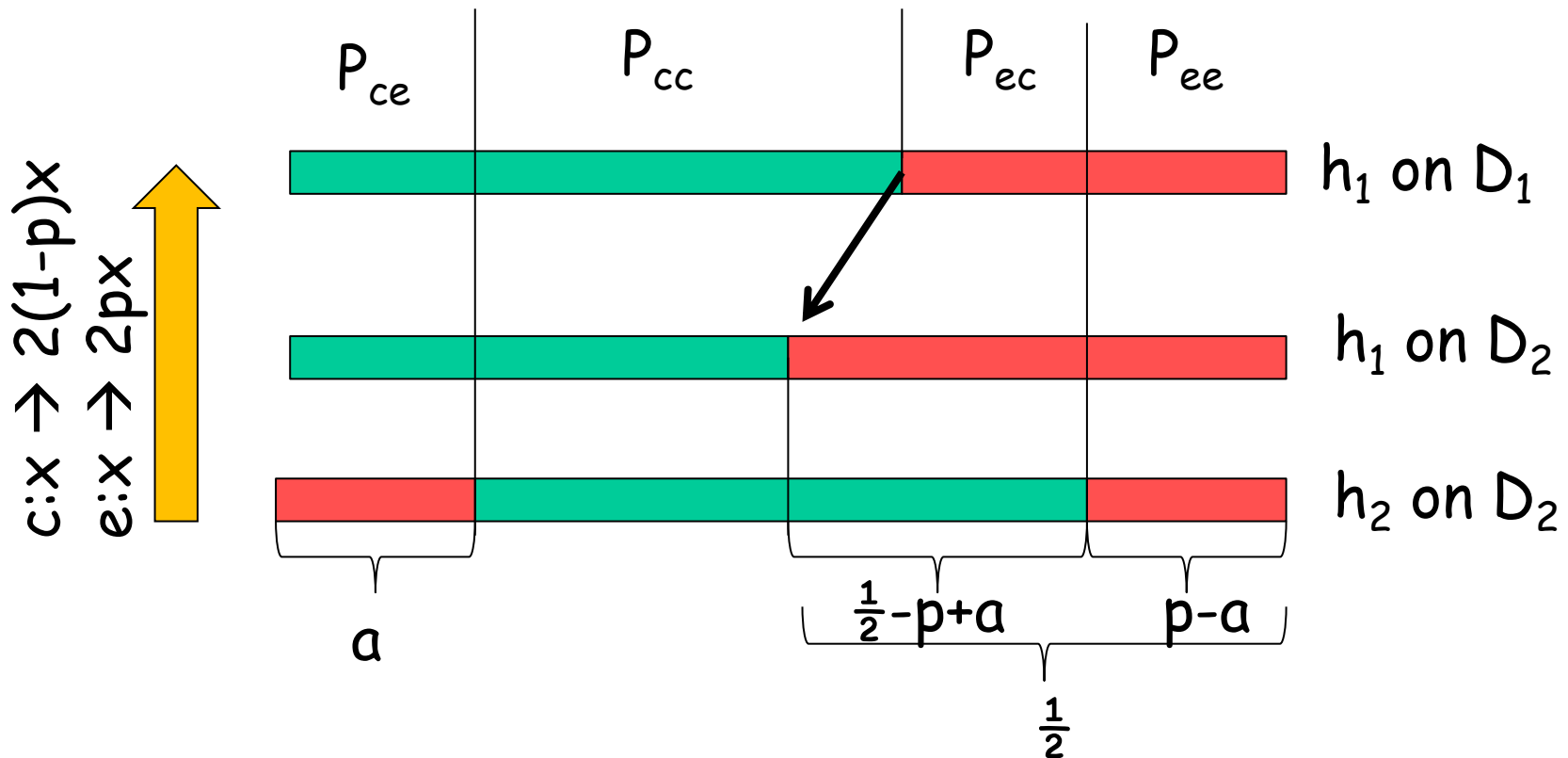
3 Weak Learners - Analysis

- Assume all WL are $p = 1/2 - \gamma$



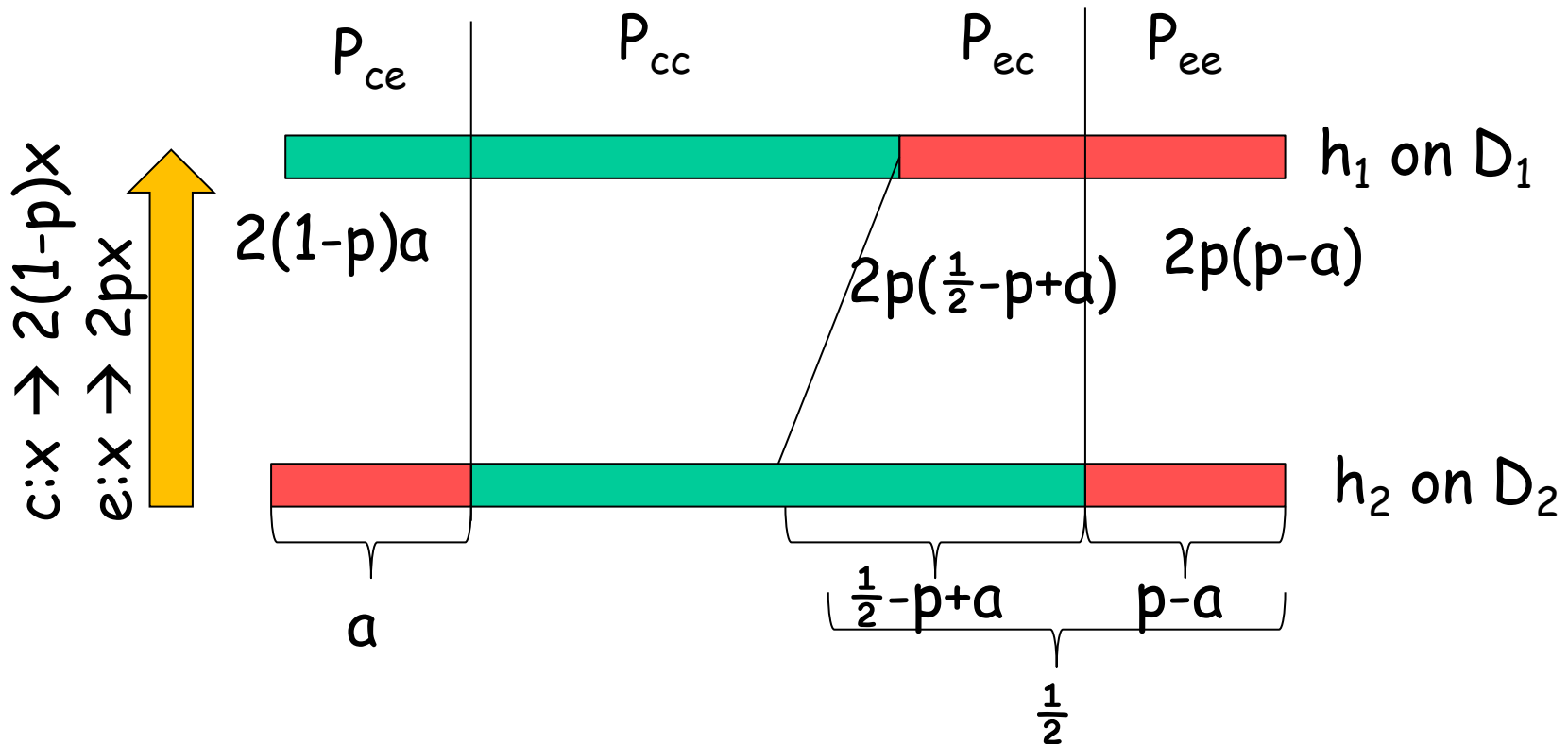
3 Weak Learners - Analysis

- Assume all WL are $p = \frac{1}{2} - \gamma$



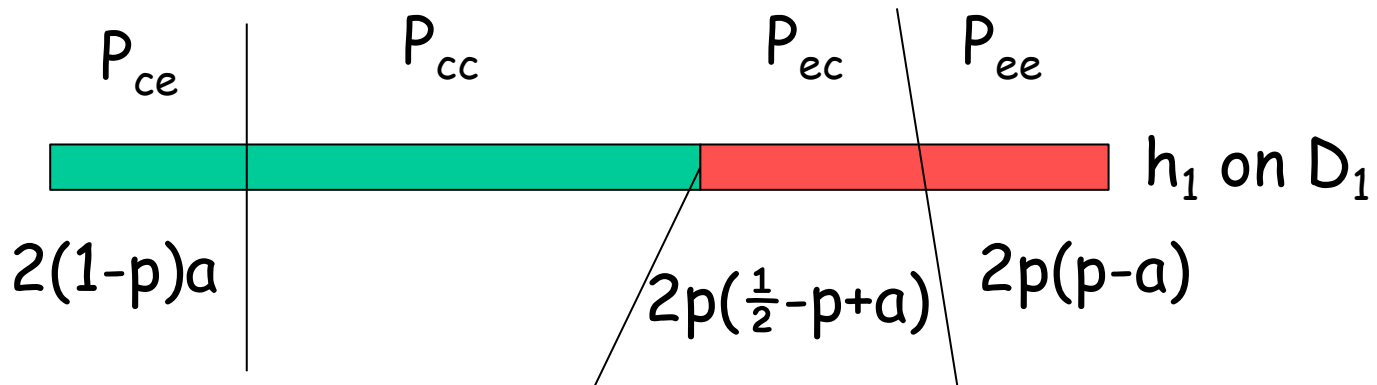
3 Weak Learners - Analysis

- Assume all WL are $p = \frac{1}{2} - \gamma$



3 Weak Learners - Analysis

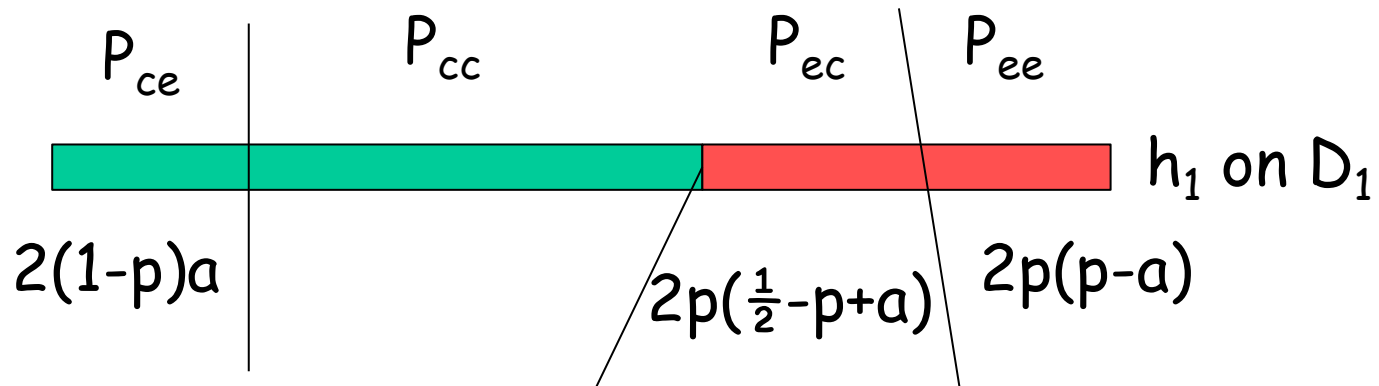
- Assume all WL are $p = \frac{1}{2} - \gamma$



$$\text{Error} = P_{ee} + p(P_{ec} + P_{ce}) = 3p^2 - 2p^3$$

3 Weak Learners - Analysis

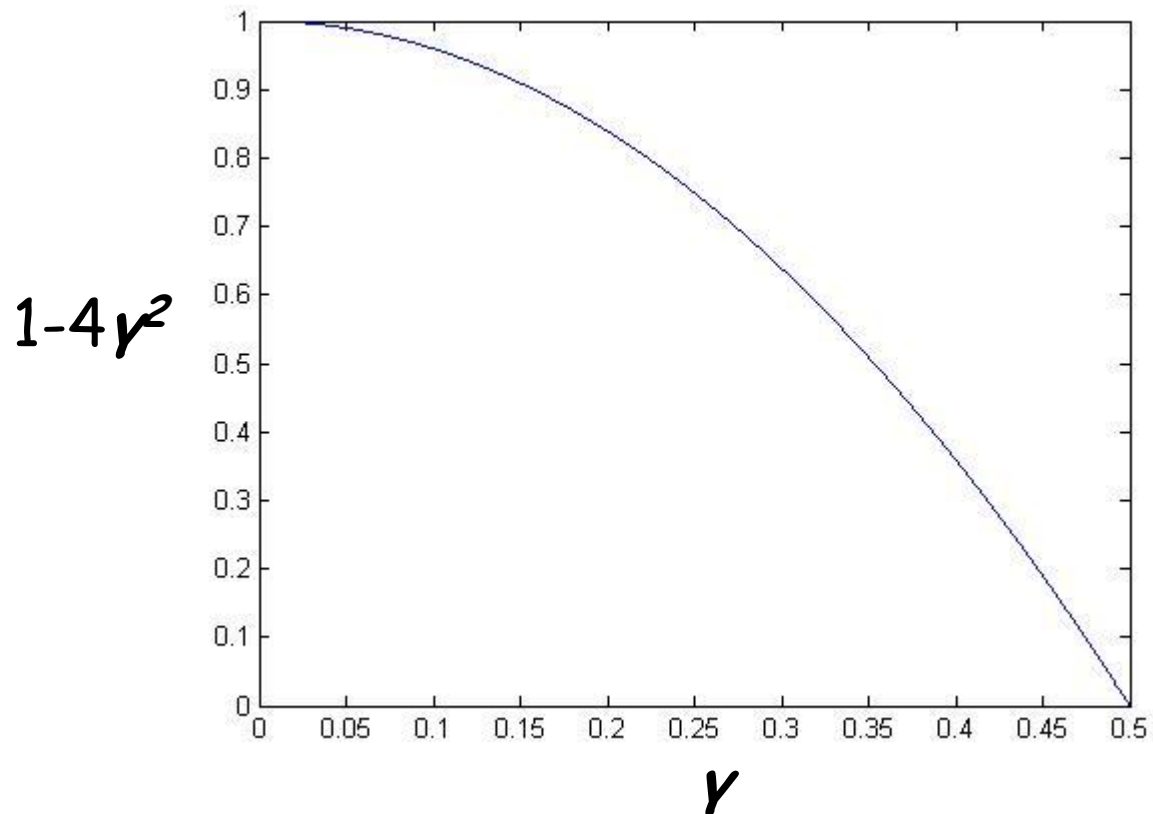
- Assume all WL are $p = \frac{1}{2} - \gamma$



$$\text{Error} = P_{ee} + p(P_{ec} + P_{ce}) = 3p^2 - 2p^3 = p(3p - 2p^2) = p(1 - 4\gamma^2)$$

3 Weak Learners - Analysis

$$\text{New Error} = \text{Old Error} * (1 - 4\gamma^2)$$



What about more hypothesis

- The CS way
 - Do it recursively
 - Can push down the error arbitrarily
- We show a more constructive way

ADABOOST: ADAPTIVE BOOSTING

AdaBoost: Overview

- Build a linear classifier
 - basic elements, weak learners
 - Ensemble method
- An “online” approach
 - each time add one more classifier
- Fit the sample \mathbf{S}
- Each time step t
 - have a hypothesis F_t
 - Select a distribution D_t
 - on \mathbf{S}
 - Find a weak learner h_t
 - w.r.t D_t
 - Add h_t to the hypothesis
 - decide on weight a_t
 - $f_{t+1}(x) = f_t(x) + a_t h_t(x)$
 - predict $\text{sign}(f_{t+1}(x))$

AdaBoost: Algorithm

- Weak Learners H

$$h: X \rightarrow \{+1, -1\}$$

- Initialization:

- Sample

- $S = \{(x_1, y_1) \dots, (x_m, y_m)\}$

- $D_1(x_i) = 1/m$

- Prediction

$$f(x) = \text{sign}(\sum \alpha_t h_t(x))$$

- Step $t = 1, \dots, T$

- Receive h_t

- WL w.r.t. D_t

- Define

- $\varepsilon_t = \Pr[h_t(x) \neq c^*(x)]$

- $\alpha_t = \frac{1}{2} \log(1 - \varepsilon_t) / \varepsilon_t$

- Define D_{t+1} ,

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

$$\propto D_t(x_i) e^{-\alpha_t y_i h_t(x_i)}$$

$$Z_t = \sum_i D_{t+1}(x_i)$$

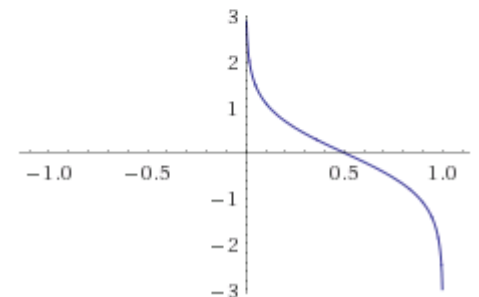
AdaBoost: Intuition

- How do we change the distribution?
 - Error \rightarrow weight increases
 - Correct \rightarrow weight decreases
 - Focus on the “hard” examples
- What are the parameters?
 - The weak learning class H
 - The number of iterations T
 - Assume to be inputs

Input:

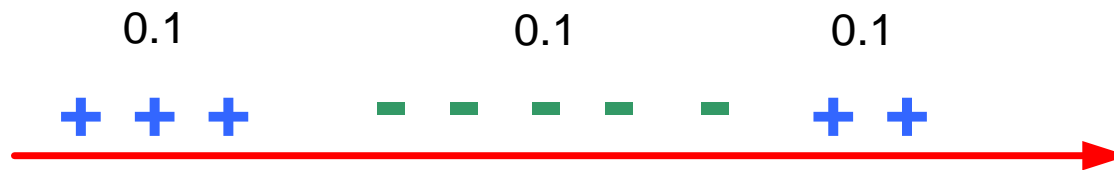
$$0.5 \log\left(\frac{1-x}{x}\right)$$

Plots:

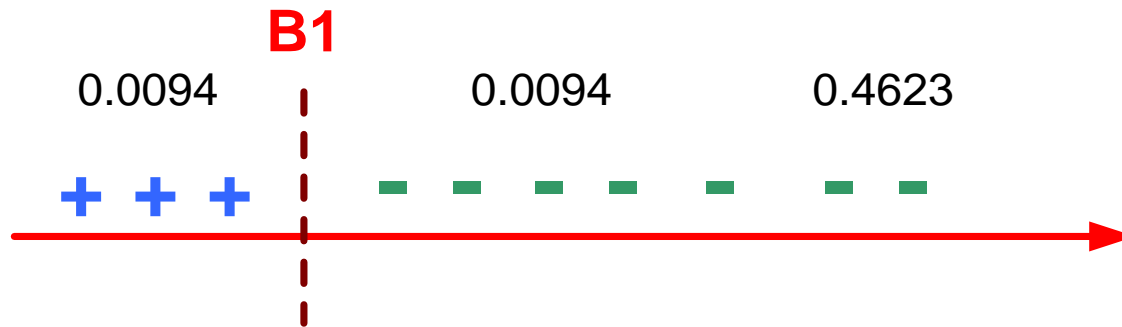


Illustrating AdaBoost

Original
Data

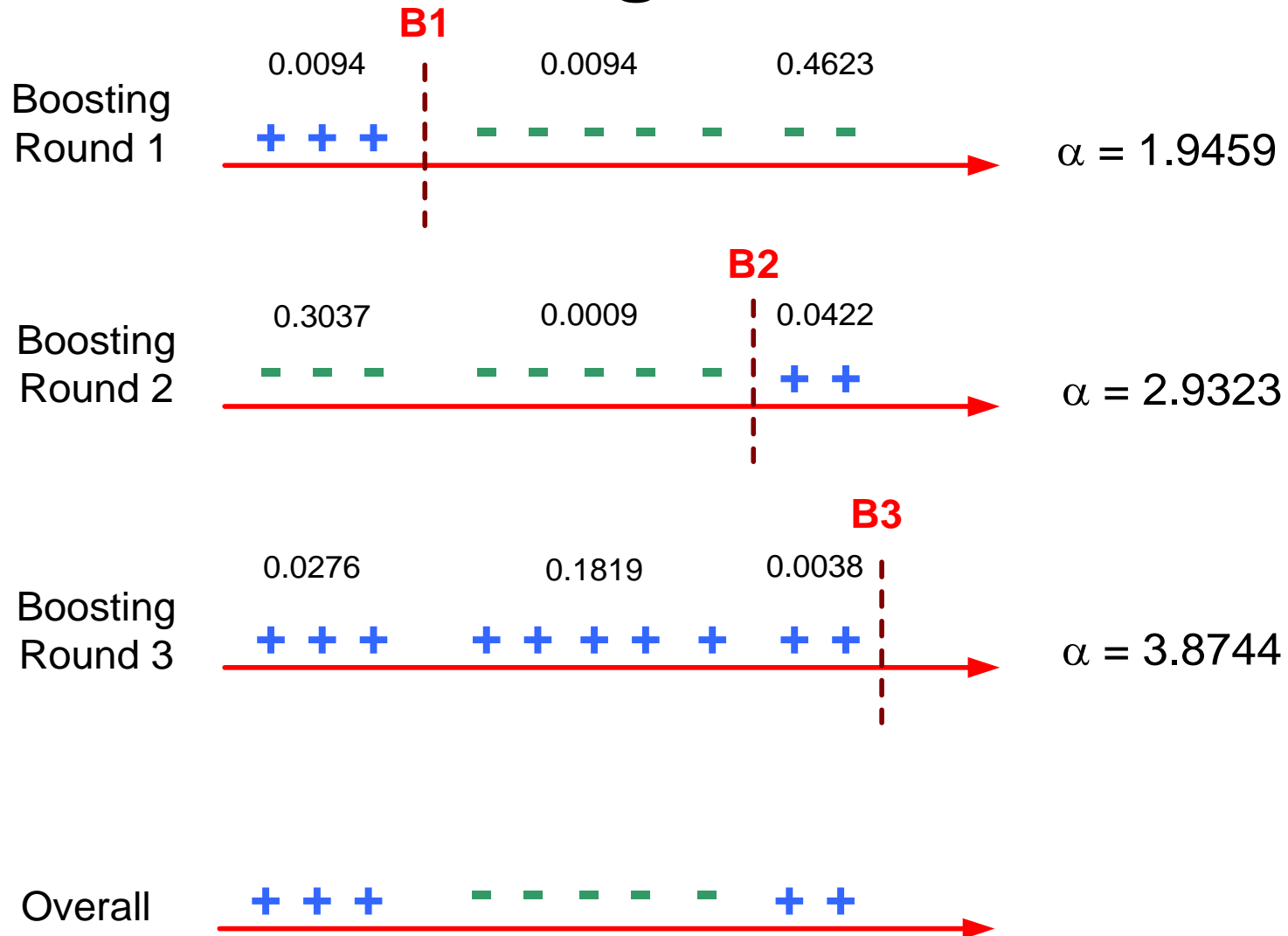


Boosting
Round 1



$$\alpha = 1.9459$$

Illustrating AdaBoost



AdaBoost: Analysis

- **Theorem:**
 - Given $\varepsilon_1, \dots, \varepsilon_T$
 - the training error ε of f is bounded by

$$\varepsilon \leq 2^{-T} \prod_{t=1}^T \sqrt{\varepsilon_t (1 - \varepsilon_t)}$$

- Proof based on three claims

AdaBoost: Analysis

Claim 1:

$$D_{T+1}(x_i) = \frac{D_1(x_i)e^{-y_i f(x_i)}}{\prod_t Z_t}$$

where $f(x) = \sum \alpha_t h_t(x)$

Corollary 1: since $D_1(x_i) = 1/m$

$$e^{-y_i f(x_i)} = m D_{T+1}(x_i) \prod_t Z_t$$

Proof

- For $t+1$ we have

$$D_{t+1}(x_i) = \frac{D_t(x_i)}{Z_t} e^{-\alpha_t y_i h_t(x_i)}$$

- unravel recurrence

$$\begin{aligned} D_{T+1}(x_i) &= D_1(x_i) \prod_{t=1}^T \frac{e^{-y_i \alpha_t h_t(x_i)}}{Z_t} \\ &= D_1(x_i) \frac{e^{-y_i \sum_{t=1}^T \alpha_t h_t(x_i)}}{\prod_t Z_t} \\ &= D_1(x_i) \frac{e^{-y_i f(x_i)}}{\prod_t Z_t} \end{aligned}$$

AdaBoost: Analysis

Claim 2:

$$\text{error}(f, S) \leq \prod_{t=1}^T Z_t$$

Proof

$$\begin{aligned} \text{error}(f, S) &= \frac{1}{m} \sum_{i=1}^m I(y_i \neq f(x_i)) \\ &= \frac{1}{m} \sum_{i=1}^m I(y_i f(x_i) \leq 0) \\ &\leq \frac{1}{m} \sum_{i=1}^m e^{-y_i f(x_i)} \\ &= \frac{1}{m} \sum_{i=1}^m m D_{T+1}(x_i) \prod_{t=1}^T Z_t \\ &= \left(\prod_{t=1}^T Z_t \right) \sum_{i=1}^m D_{T+1}(x_i) \\ &= \prod_{t=1}^T Z_t \end{aligned}$$

$$z < 0 \rightarrow e^{-z} > 1$$

Corollary 1

AdaBoost: Analysis

Claim 3:

$$Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$$

Proof

$$\begin{aligned} Z_t &= \sum_{i=1}^m D_t(x_i) e^{-y_i \alpha_t h_t(x_i)} \\ &= \sum_{i: y_i = h_t(x_i)} D_t(x_i) e^{-\alpha_t} + \sum_{i: y_i \neq h_t(x_i)} D_t(x_i) e^{\alpha_t} \\ &= (1 - \varepsilon_t) e^{-\alpha_t} + \varepsilon_t e^{\alpha_t} \end{aligned}$$

Recall

$$\sum_{i: y_i \neq h_t(x_i)} D_t(x_i) = \varepsilon_t$$

AdaBoost: Analysis

- Completing the proof
 - $Z_t = (1 - \varepsilon_t)e^{-\alpha_t} + \varepsilon_t e^{\alpha_t}$ valid for any α_t
 - Minimize Z_t

$$\frac{dZ_t}{d\alpha_t} = -(1 - \varepsilon_t)e^{-\alpha_t} + \varepsilon_t e^{\alpha_t}$$

$$e^{2\alpha_t} = \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$$

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right) \Rightarrow Z_t = 2\sqrt{\varepsilon_t(1 - \varepsilon_t)}$$

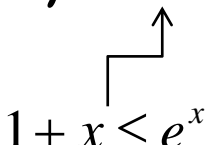
AdaBoost: Analysis

- **Theorem:**

- Given $\varepsilon_1, \dots, \varepsilon_T$

- the error ε of f is bounded by $\prod_{t=1}^T Z_t$

$$\varepsilon \leq 2^T \prod_{t=1}^T \sqrt{\varepsilon_t (1 - \varepsilon_t)} = \prod_{t=1}^T \sqrt{1 - 4\gamma_t^2} \leq e^{-2 \sum_t \gamma_t^2}$$

$1 + x \leq e^x$ 

AdaBoost: Analysis

- **Theorem:**

- Given $\varepsilon_1, \dots, \varepsilon_T$

- the error ε of f is bounded by $\prod_{t=1}^T Z_t$

$$\varepsilon \leq 2^T \prod_{t=1}^T \sqrt{\varepsilon_t(1-\varepsilon_t)} = \prod_{t=1}^T \sqrt{1-4\gamma_t^2} \leq e^{-2\sum_t \gamma_t^2}$$

$1+x \leq e^x$

- For $\gamma_t > \gamma$

$$\varepsilon \leq e^{-2T\gamma^2}$$

Additional Issues

- Boosting the margin
- Overfitting
- When to stop
- Advantages
- Weaknesses
- Implementation
- Bagging/ Stacking/ Boosting
- Decision stumps