# Boosting

## Can we make dumb learners smart?

Aarti Singh

Machine Learning 10-315
Nov 3, 2021

Slides Courtesy: Carlos Guestrin, Freund & Schapire

# Why boost weak learners?

**Goal:** Automatically categorize type of call requested

(Collect, Calling card, Person-to-person, etc.)

- yes I'd like to place a collect call long distance please (Collect)
- operator I need to make a call but I need to bill it to my office (ThirdNumber)
- yes I'd like to place a call on my master card please (CallingCard)

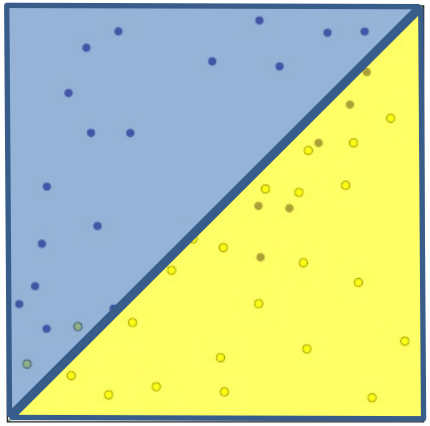
- **Easy to find "rules of thumb" that are "often" correct.**

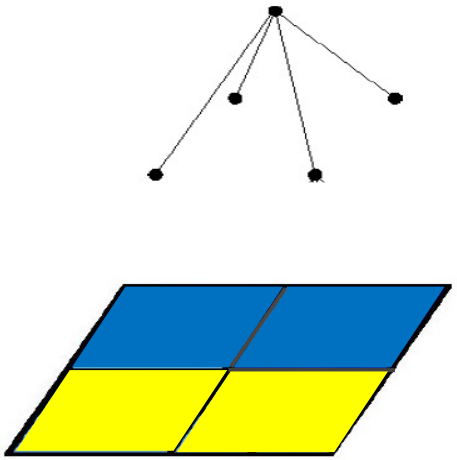
  E.g. If 'card' occurs in utterance, then predict 'calling card'

- **Hard to find single highly accurate prediction rule.**

# Fighting the bias-variance tradeoff

- **Simple (a.k.a. weak) learners** e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)





**Are good** ☺ - don't usually overfit ✓

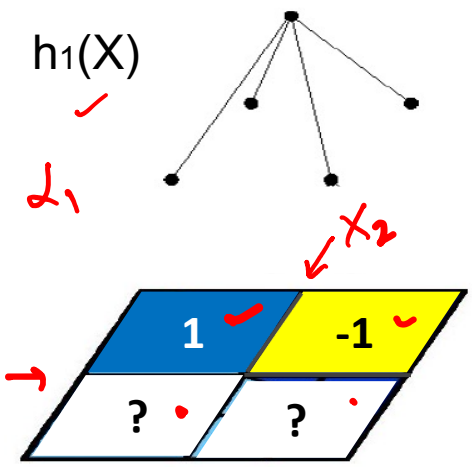
**Are bad** ☹ - can't solve hard learning problems ✓

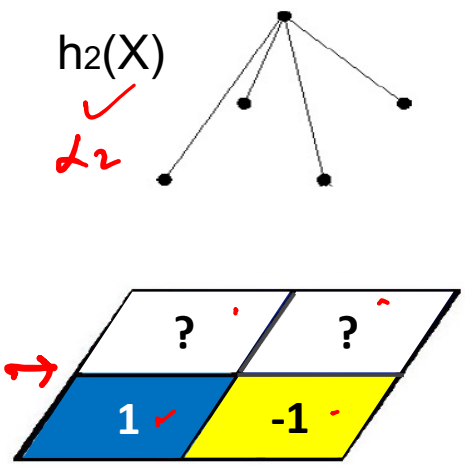
- **Can we make weak learners good???**

# Voting (Ensemble Methods)

- Instead of learning a single (weak) classifier, learn **many weak classifiers** that are **good at different parts of the input space**

- **Output class:** (Weighted) vote of each classifier
  - Classifiers that are most "sure" will vote with more conviction
  - Classifiers will be most "sure" about a particular part of the space
  - On average, do better than single classifier!




$H: X \rightarrow Y\ (-1,1)$

$H(X) = h_1(X)+h_2(X)$

$H(X) = \text{sign}(\sum_t \alpha_t h_t(X))$

weights

# Voting (Ensemble Methods)

- Instead of learning a single (weak) classifier, learn **many weak classifiers** that are **good at different parts of the input space**
- **Output class:** (Weighted) vote of each classifier
  - Classifiers that are most "sure" will vote with more conviction
  - Classifiers will be most "sure" about a particular part of the space
  - On average, do better than single classifier!
- **But how do you ???**
  - force classifiers $h_t$ to learn about different parts of the input space?
  - weigh the votes of different classifiers? $\alpha_t$

# Boosting [Schapire'89]

- **Idea:** given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote

- On each iteration *t*:
  - weight $D_t(i)$ for each training example i, based on how incorrectly it was classified
  - Learn a weak hypothesis – $h_t$
  - A weight for this hypothesis – $\alpha_t$

$\text{sign}\left(\sum_t \alpha_t h_t\right)$

- Final classifier: $H(X) = \text{sign}(\sum \alpha_t h_t(X))$

- **Practically useful**
- **Theoretically interesting**

# Learning from weighted data

- **Consider a weighted dataset**
  - D(i) – weight of *i* th training example ($\mathbf{x}^i$,$y^i$)
  - Interpretations:
    - *i* th training example counts as D(i) examples
    - If I were to "resample" data, I would get more samples of "heavier" data points

- **Now, in all calculations, whenever used, *i* th training example counts as D(i) "examples"**
  - e.g., in MLE redefine *Count(Y=y)* to be weighted count

**Unweighted data**

$$Count(Y=y) = \sum_{i=1}^{m} \mathbf{1}(Y^i=y)$$

**Weights** D(i)

$$Count(Y=y) = \sum_{i=1}^{m} D(i)\mathbf{1}(Y^i=y)$$

# AdaBoost [Freund & Schapire'95]

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

Initially equal weights

For $t = 1, \ldots, T$:

- Train weak learner using distribution $D_t$.
- Get weak classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

Naïve bayes, decision stump

Magic (+ve)

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

$$= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$y_i, h_t(x_i) \in \{-1, +1\}$

Increase weight if wrong on pt i

$y_i\, h_t(x_i) = -1 < 0$

where $Z_t$ is a normalization factor

# AdaBoost [Freund & Schapire'95]

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

**Initially equal weights**

$\rightarrow \sum_i D_1(i) = 1$

For $t = 1, \ldots, T$:

- Train weak learner using distribution $D_t$.
- Get weak classifier $h_t : X \rightarrow \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

**Naïve bayes, decision stump**

**Magic (+ve)**

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

**Increase weight if wrong on pt i**
**$y_i h_t(x_i) = -1 < 0$**

where $Z_t$ is a normalization factor

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

$\Rightarrow \sum_i D_{t+1}(i) = 1$

**Weights for all pts must sum to 1**
**$\sum_t D_{t+1}(i) = 1$**

# AdaBoost [Freund & Schapire'95]

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$. **Initially equal weights**

For $t = 1, \ldots, T$:

- Train weak learner using distribution $D_t$. **Naïve bayes, decision stump**
- Get weak classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$. **Magic (+ve)**
- Update:

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

**Increase weight if wrong on pt i**
**$y_i h_t(x_i) = -1 < 0$**

where $Z_t$ is a normalization factor

Output the final classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

# What $\alpha_t$ to choose for hypothesis $h_t$?

Weight Update Rule: $$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$\Rightarrow \boxed{\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)}$$ **[Freund & Schapire'95]**

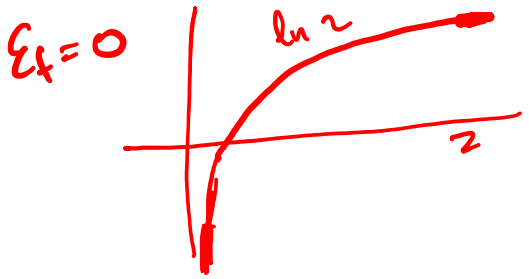
**Weighted training error**

$$\epsilon_t = P_{i\sim D_t(i)}[h_t(\mathbf{x}^i) \neq y^i] = \sum_{i=1}^{m} D_t(i)\delta(h_t(x_i) \neq y_i)$$

loss

$$= \begin{cases} 1 & h_t(x_i) \neq y_i \\ 0 & h_t(x_i) = y_i \end{cases}$$

Does $h_t$ get $i^{th}$ point wrong

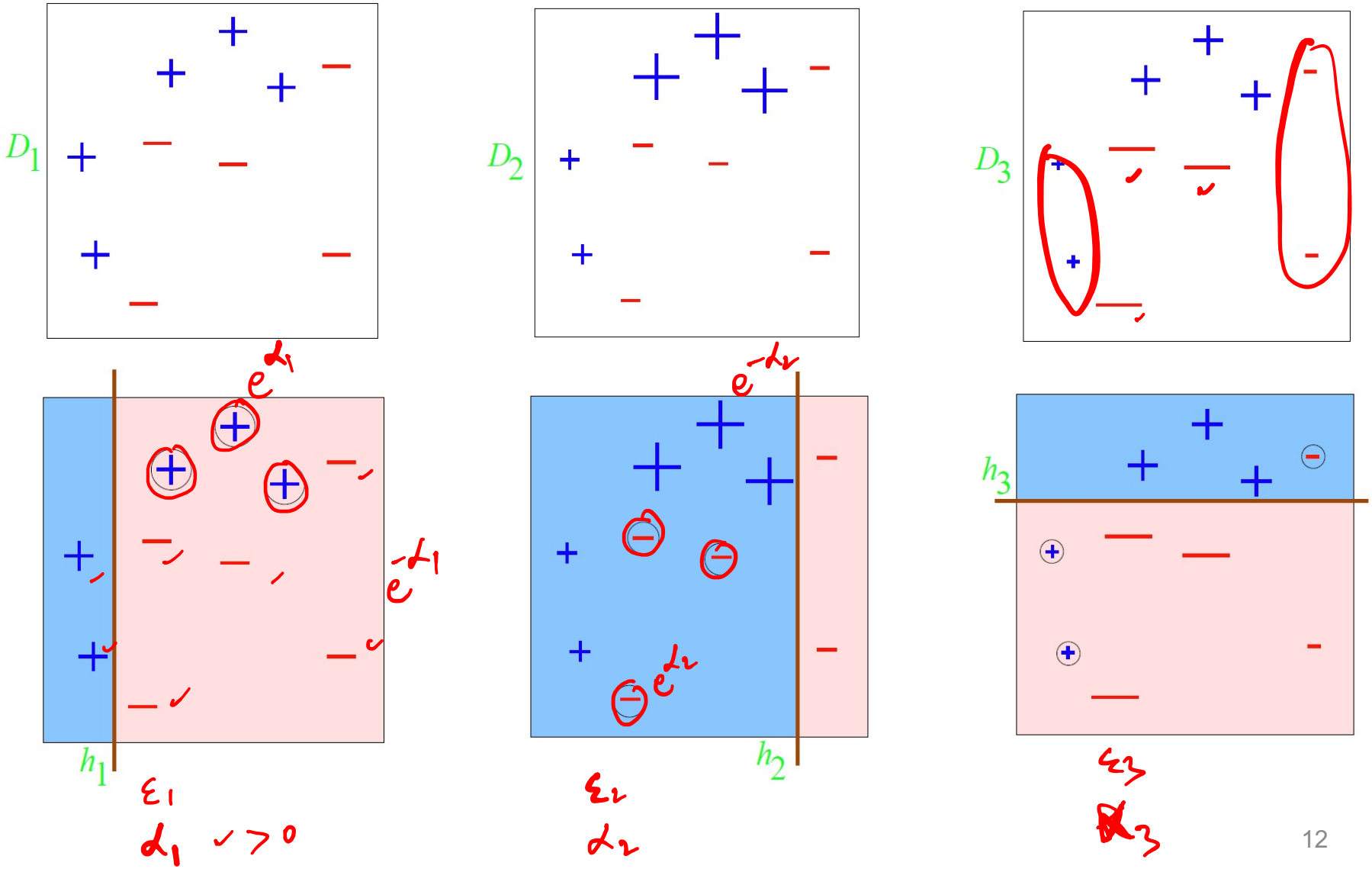



$\epsilon_t$ = 0 if $h_t$ perfectly classifies all weighted data pts — $\alpha_t = \infty$

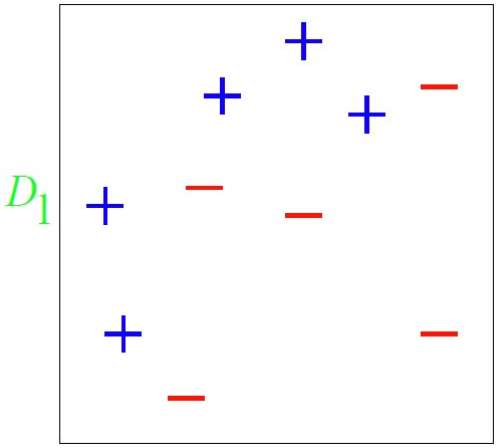
$\epsilon_t$ = 1 if $h_t$ perfectly wrong => $-h_t$ perfectly right — $\alpha_t = -\infty$

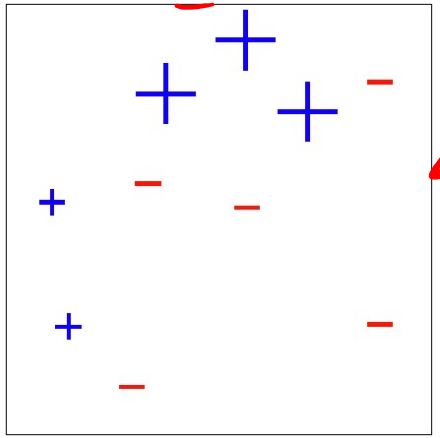
$\epsilon_t$ = 0.5 — $\alpha_t = 0$

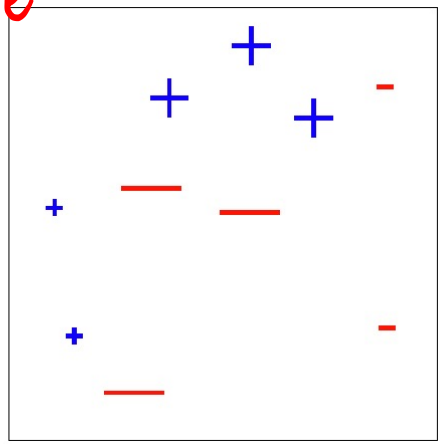
# Boosting Example (Decision Stumps)

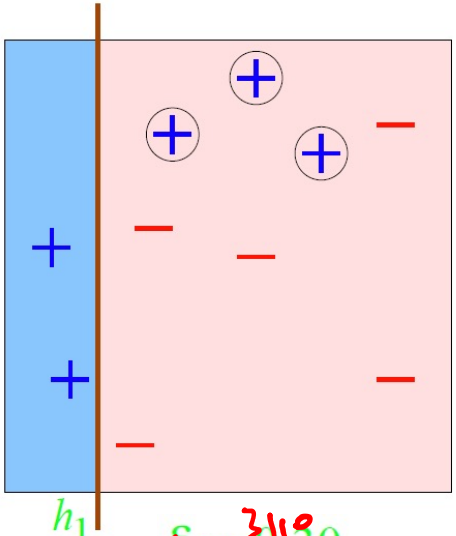


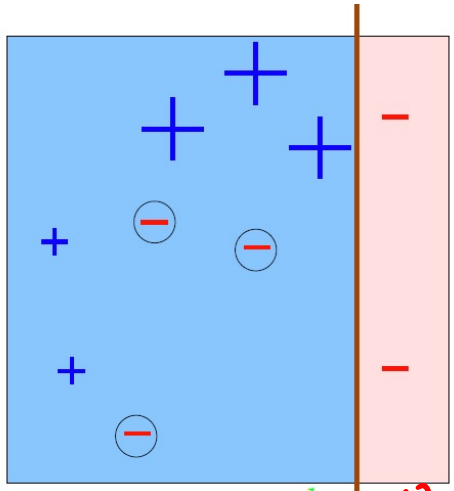
# Boosting Example (Decision Stumps)

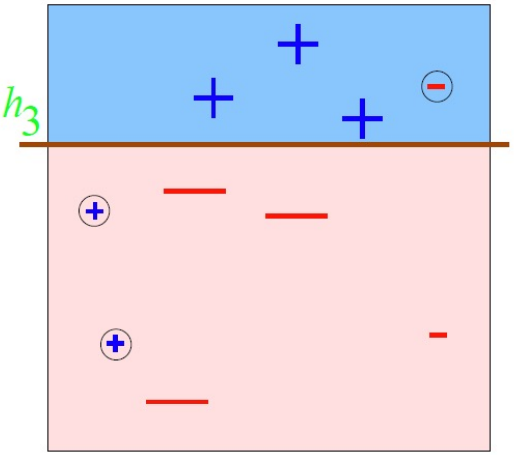


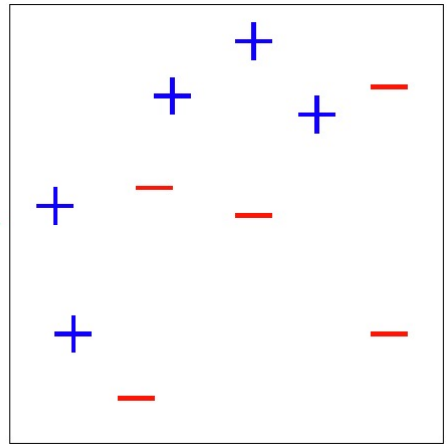



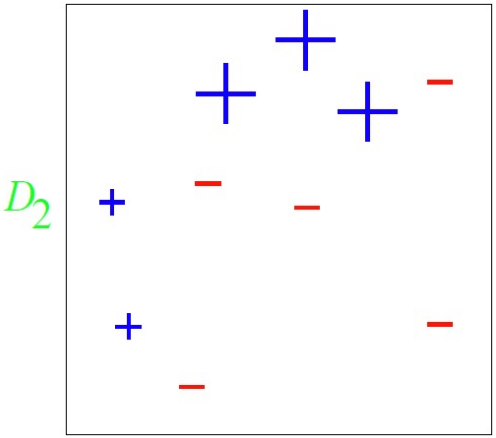

$\varepsilon_1 = 0.30$ $\alpha_1 = 0.42$

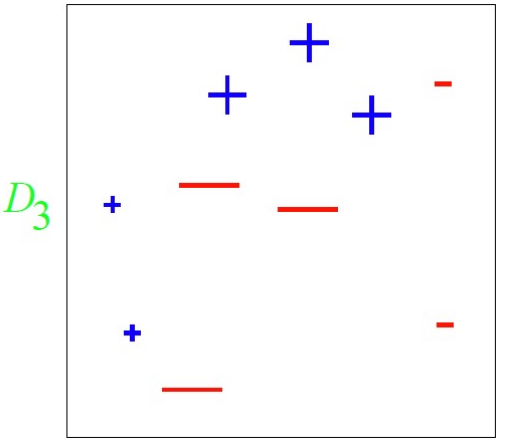
$\varepsilon_2 = 0.21$ $\alpha_2 = 0.65$

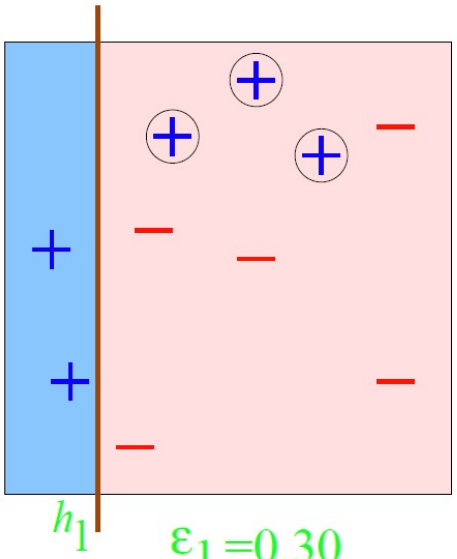
$\varepsilon_3 = 0.14$ $\alpha_3 = 0.92$

# Boosting Example (Decision Stumps)



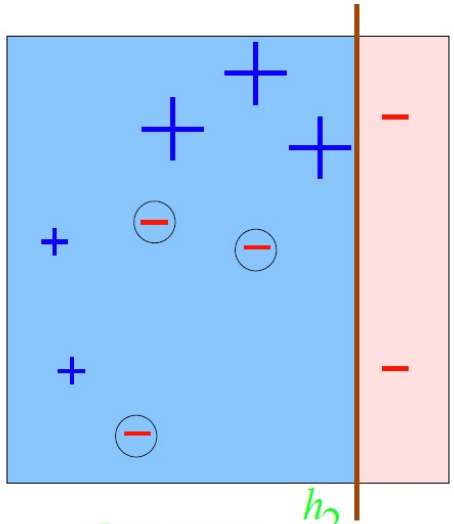



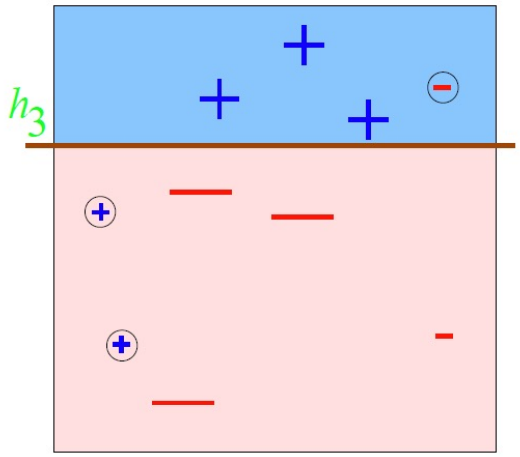

$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$




$\varepsilon_2 = 0.21$

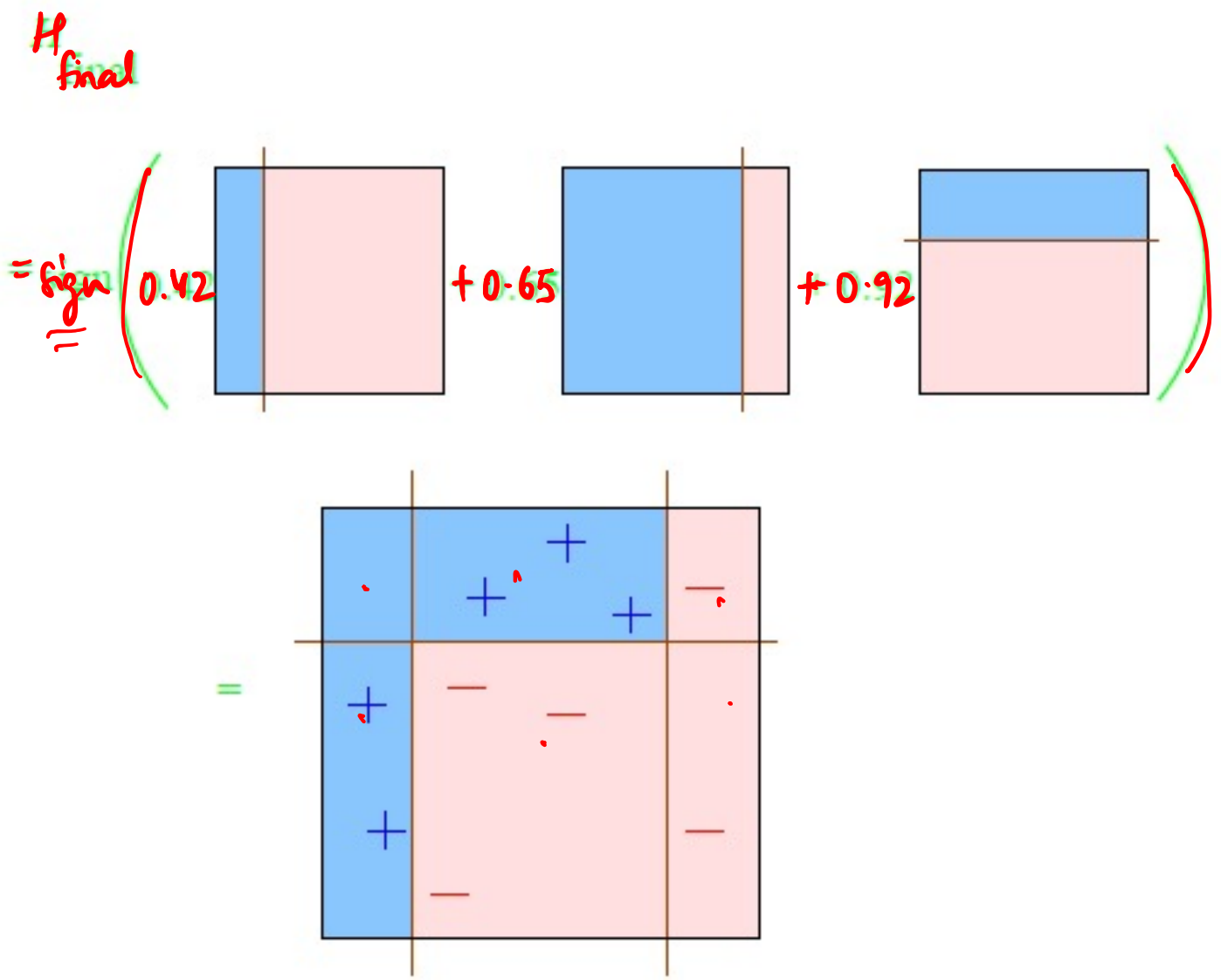
$\alpha_2 = 0.65$




$\varepsilon_3 = 0.14$

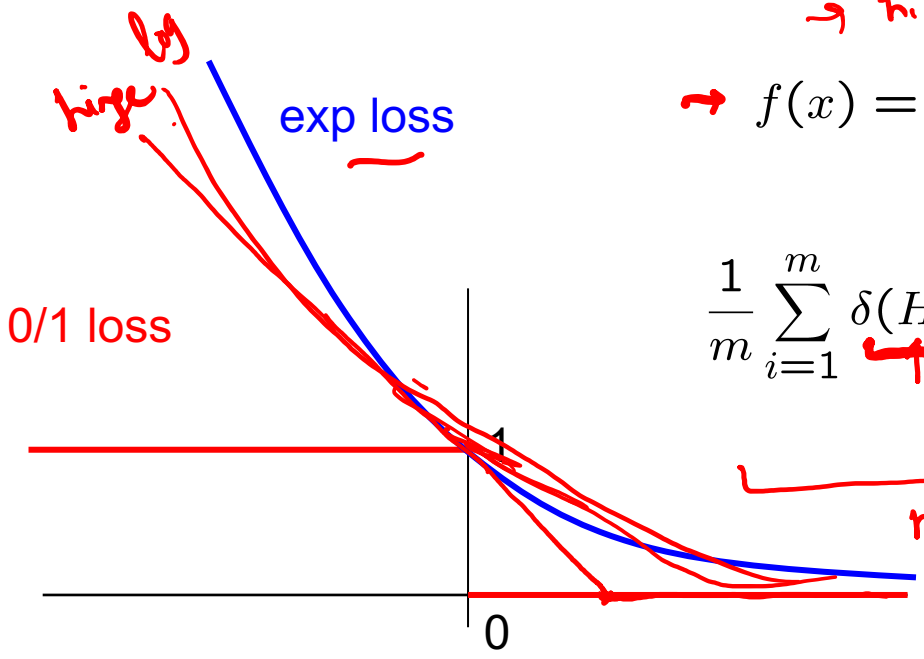
$\alpha_3 = 0.92$

# Boosting Example (Decision Stumps)



# Analysis for Boosting

- Choice of $\alpha_t$ and hypothesis $h_t$ obtained by coordinate descent on exp loss (convex upper bound on 0/1 loss)




$$f(x) = \sum_t \alpha_t h_t(x); \; H(x) = sign(f(x))$$

$$\frac{1}{m}\sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m}\sum_{i=1}^{m} \exp(-y_i f(x_i))$$

0/1 loss — exp loss

# Analysis for Boosting

Analysis reveals:

- If each weak learner $h_t$ is slightly better than random guessing ($\varepsilon_t < 0.5$), then training error of AdaBoost decays exponentially fast in number of rounds T.

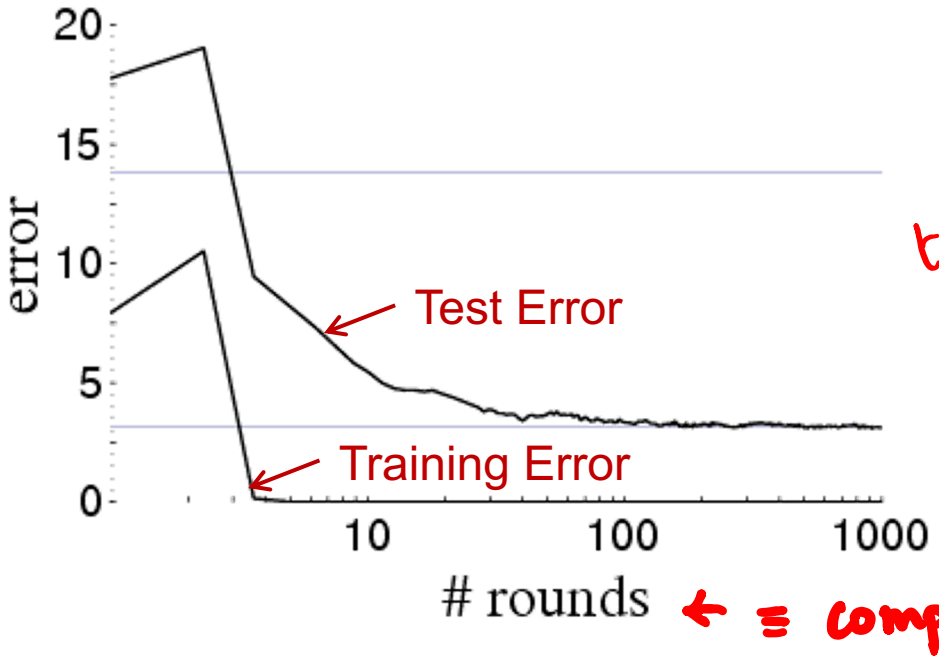
$$\frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i)\neq y_i) \leq \exp\left(-2\sum_{t=1}^{T}(1/2-\epsilon_t)^2\right)$$

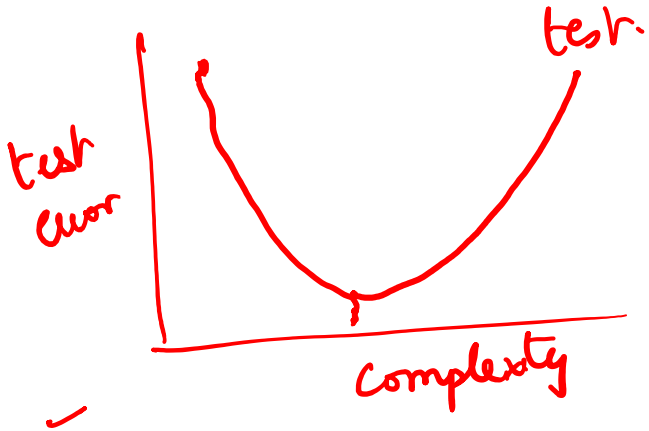
**Training Error**

**What about test error?**

# Boosting results – Digit recognition

[Schapire, 1989]

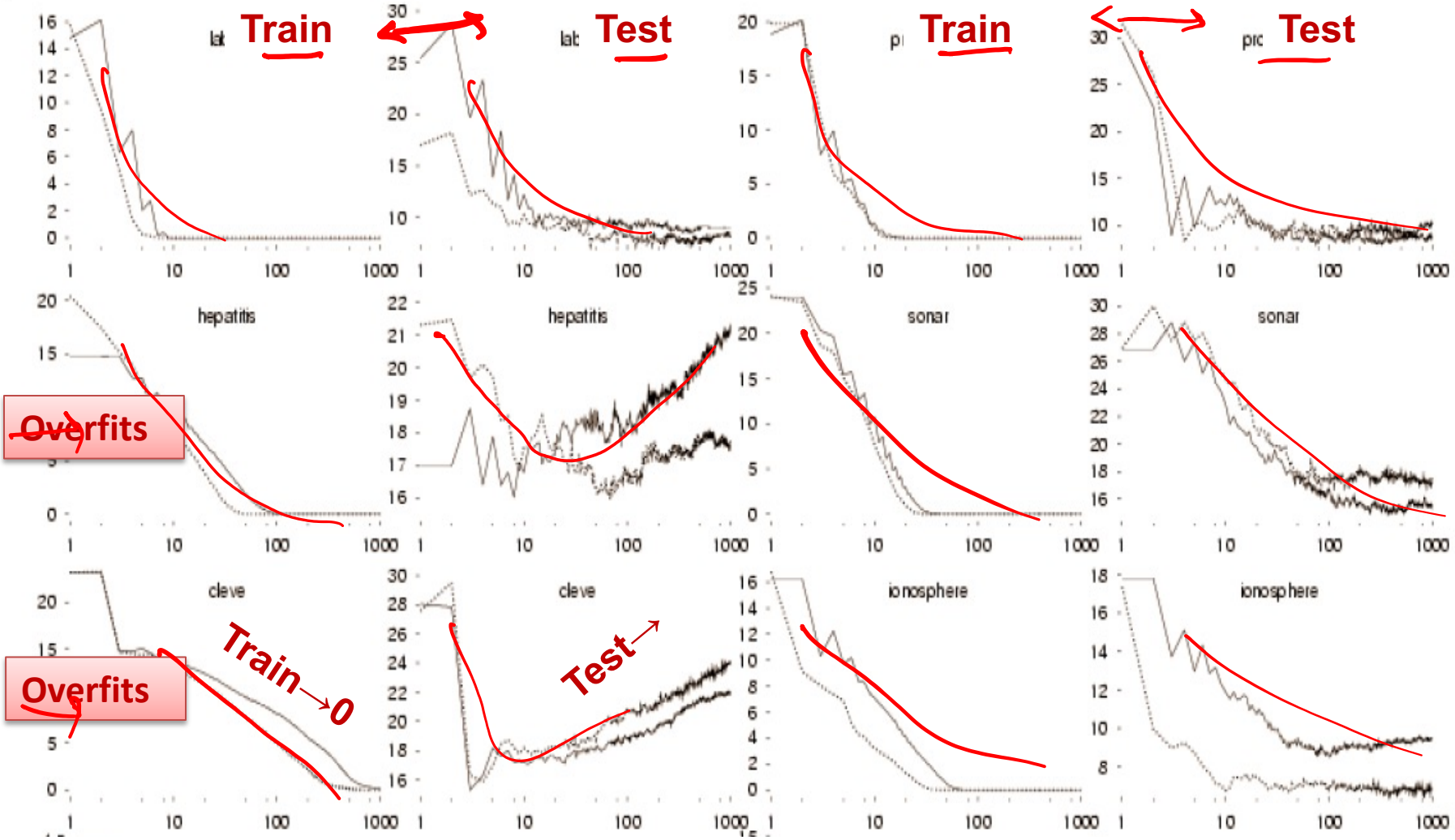



← ≡ complexity

- Boosting often,
  - Robust to overfitting
  - Test set error decreases even after training error is zero
- If margin between classes is large, subsequent weak learners agree and hence more rounds does not necessarily imply that final classifier is getting more complex.

AdaBoost and AdaBoost.MH on Train (left) and Test (right) data from Irvine repository. [Schapire and Singer, ML 1999]




Boosting can overfit if margin between classes is too small (high label noise) or weak learners are too complex.

# Boosting and Logistic Regression

Logistic regression:

- Minimize log loss

$$\sum_{i=1}^{m} \ln(1 + \exp(-y_i f(x_i)))$$

- Define

$$f(x) = \sum_j w_j x_j$$

where $x_j$ predefined features

(linear classifier)

- Jointly optimize over all weights *w0, w1, w2…*

Boosting:

- Minimize exp loss

$$\sum_{i=1}^{m} \exp(-y_i f(x_i))$$

- Define

$$f(x) = \sum_t \alpha_t h_t(x)$$

where $h_t(x)$ defined dynamically to fit data

learnt

(not a linear classifier)

- Weights $\alpha_t$ learned per iteration t incrementally

# Boosting Summary

- Combine weak classifiers to obtain strong classifier
  - Weak classifier – slightly better than random on training data
  - Resulting very strong classifier – can eventually provide zero training error
- AdaBoost algorithm
- Boosting v. Logistic Regression
  - Similar loss functions
  - Single optimization (LR) v. Incrementally improving classification (B)
- Most popular application of Boosting:
  - Boosted decision stumps!
  - Very simple to implement, very effective classifier