Boosting

Can we make dumb learners smart?

Aarti Singh

Mar 13, 2023
Machine Learning 10-701

Slides Courtesy: Carlos Guestrin, Freund & Schapire

Why boost weak learners?

Goal: Classify movie review sentiment

“I'm a fan of TV movies in general and this was one of the **good** ones”

“Long, **boring**. Never have I been so glad to see ending credits roll”

“I don’t know why I **like** this movie, but I never get tired.”

- **Easy to find “rules of thumb” that are better than random chance.**

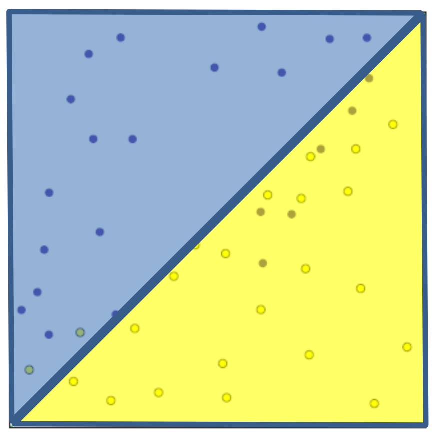
  E.g. If ‘good’ occurs in utterance, then predict ‘positive’

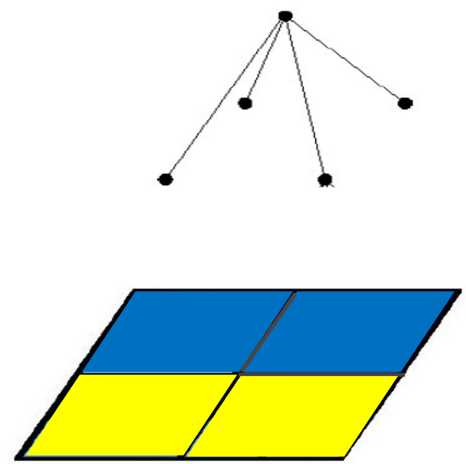
- **Hard to find single highly accurate prediction rule.**

  e.g. “This movie is terrible but it has some **good** effects”

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

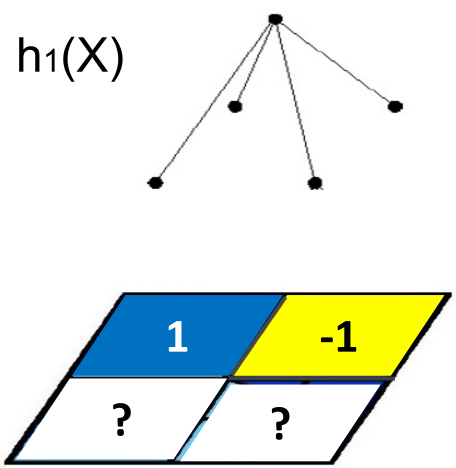
Are bad ☹ - can't solve hard learning problems

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

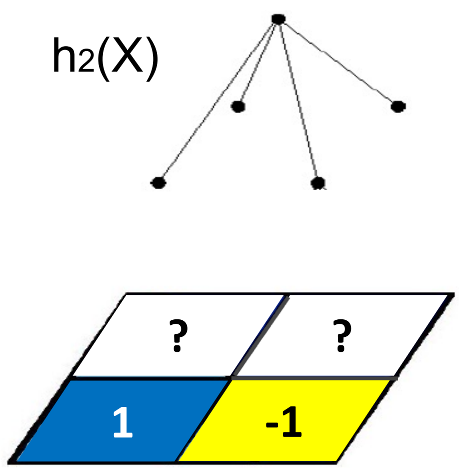
Voting (Ensemble Methods)

- 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!

$h_1(X)$



$h_2(X)$



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

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

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

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? α_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 – α_t

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

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

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

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

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**
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

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

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 α_t to choose for hypothesis h_t?

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

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

[Freund & Schapire'95]

Weighted training error

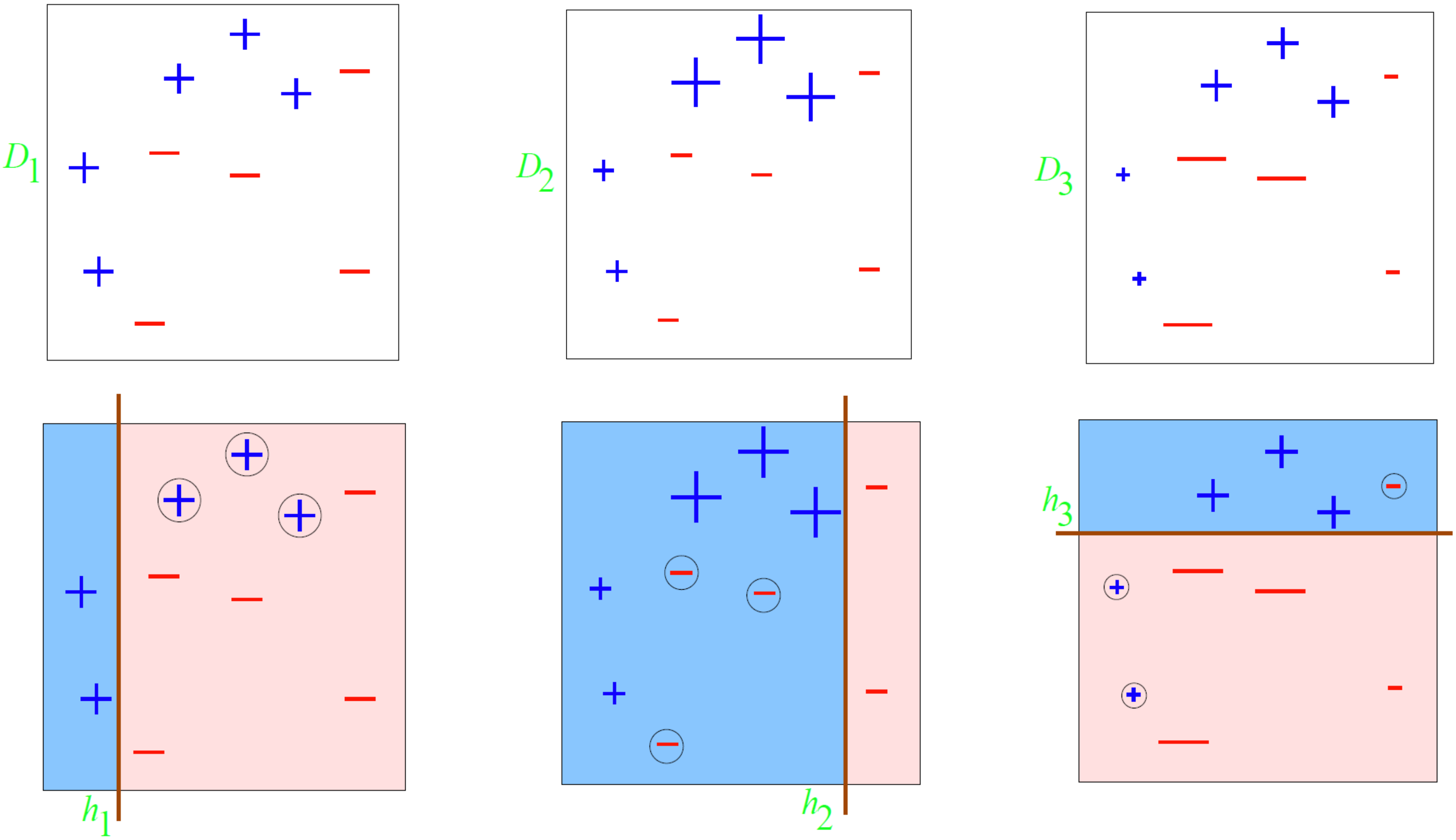
$$\epsilon_t = P_{i\sim D_t(i)}[h_t(\mathbf{x}^i) \neq y^i] = \sum_{i=1}^{m} D_t(i)\underbrace{\delta(h_t(x_i) \neq y_i)}_{\text{Does } h_t \text{ get } i^{th} \text{ point wrong}}$$

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

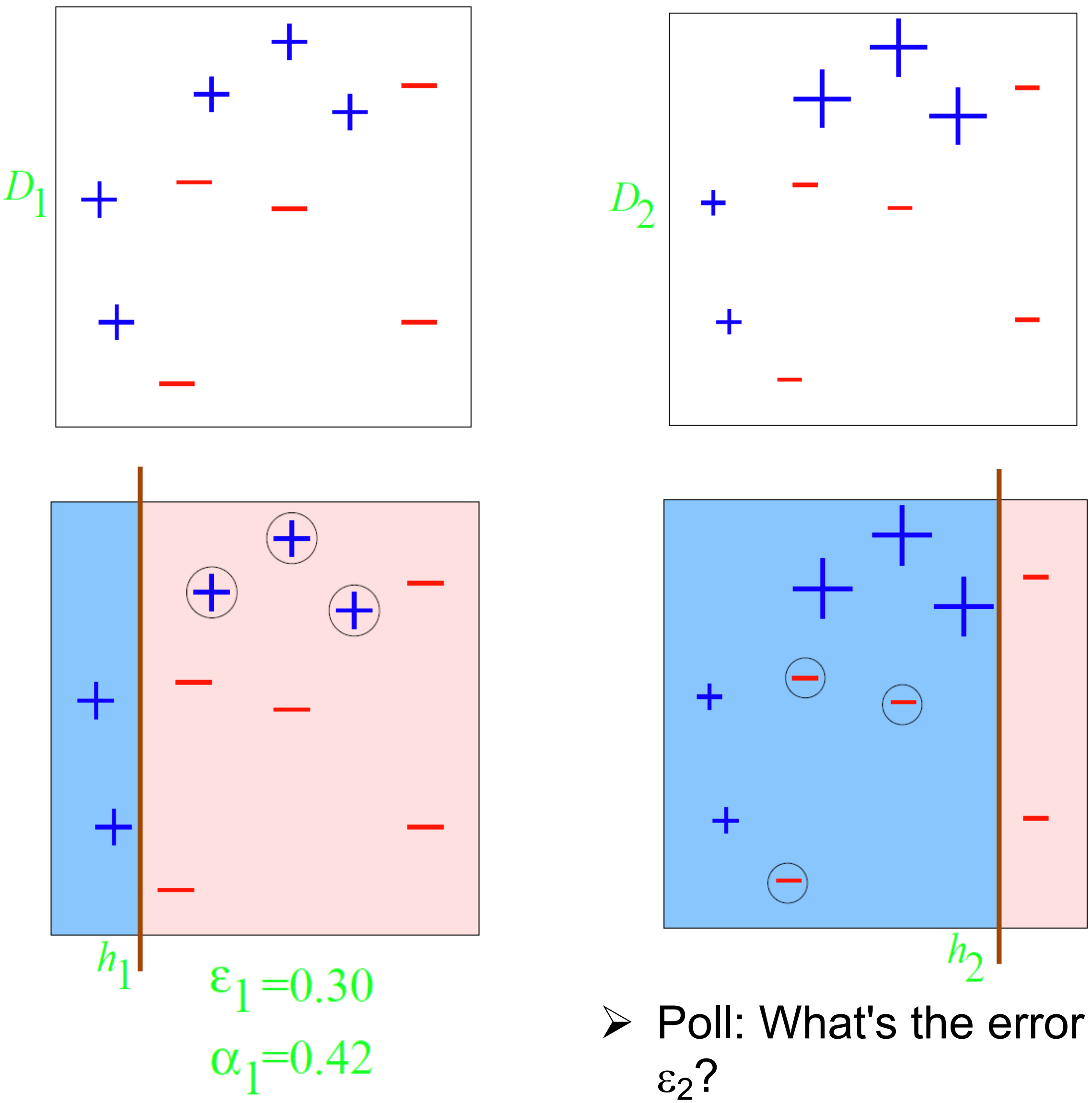
ε_t = 1 if h_t perfectly wrong => $-h_t$ perfectly right — $\alpha_t = -\infty$

$\varepsilon_t = 0.5$ — $\alpha_t = 0$

Boosting Example (Decision Stumps)

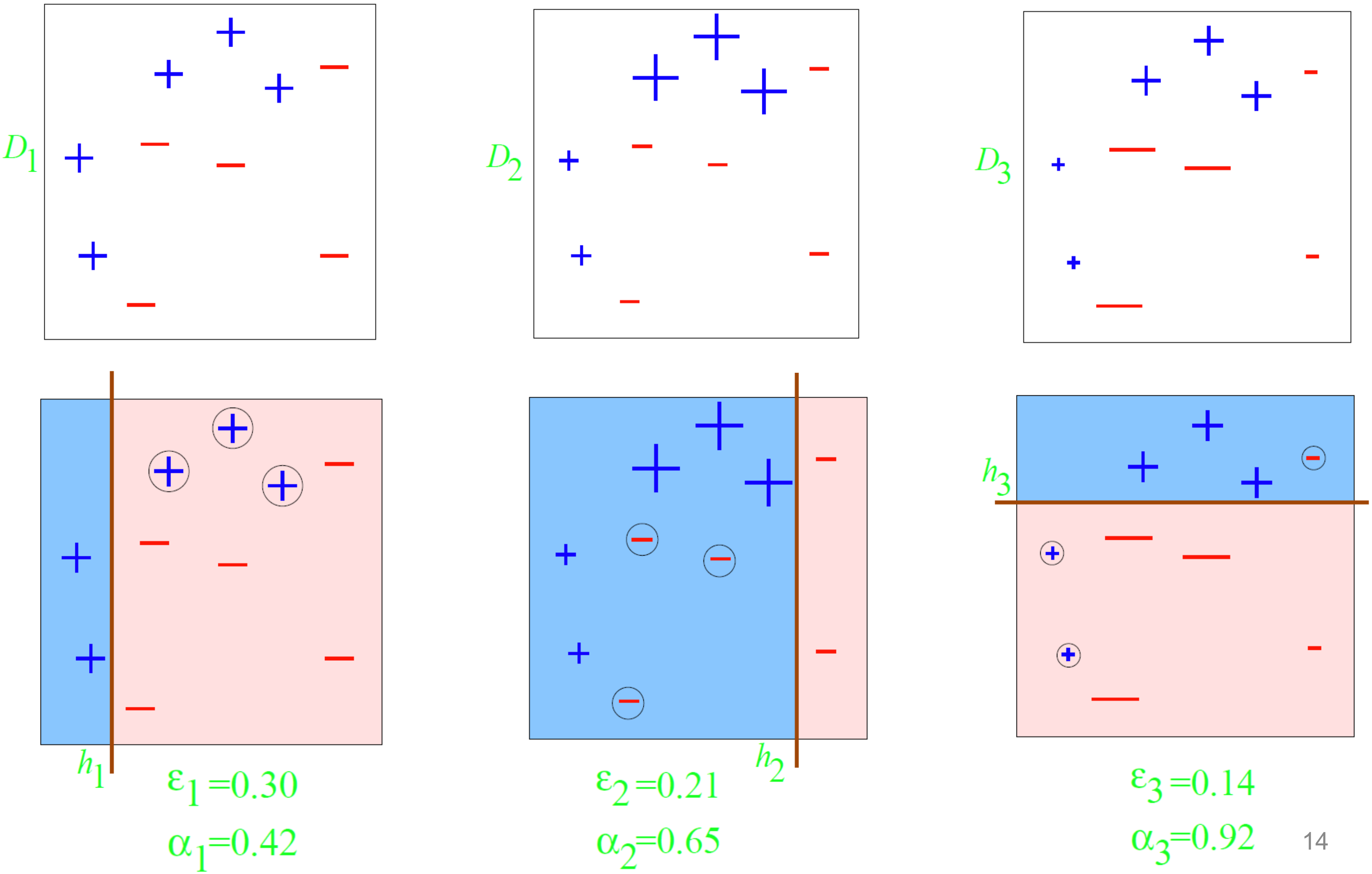


Boosting Example (Decision Stumps)

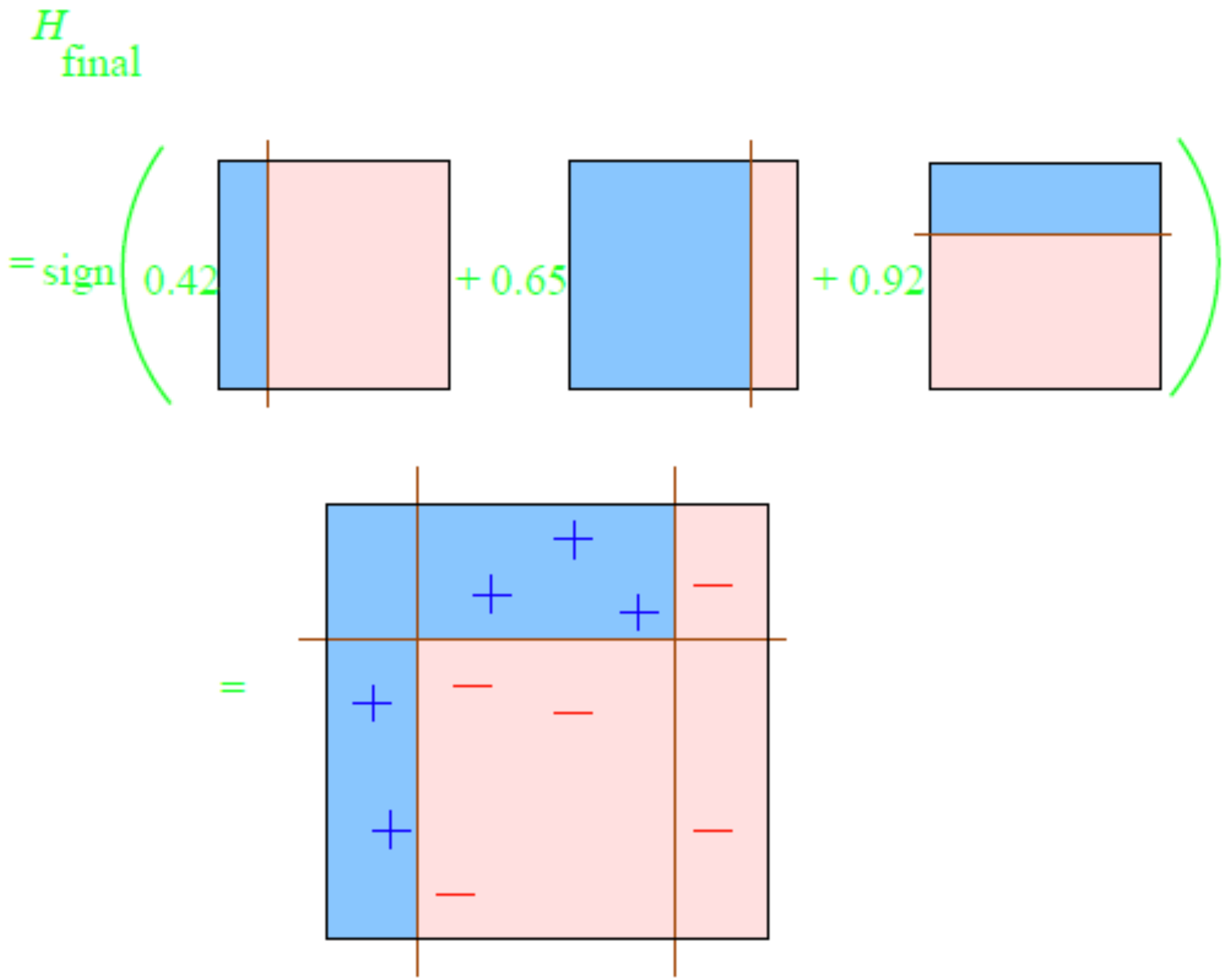


- Poll: What's the error on the weighted training data, ε_2?

Boosting Example (Decision Stumps)

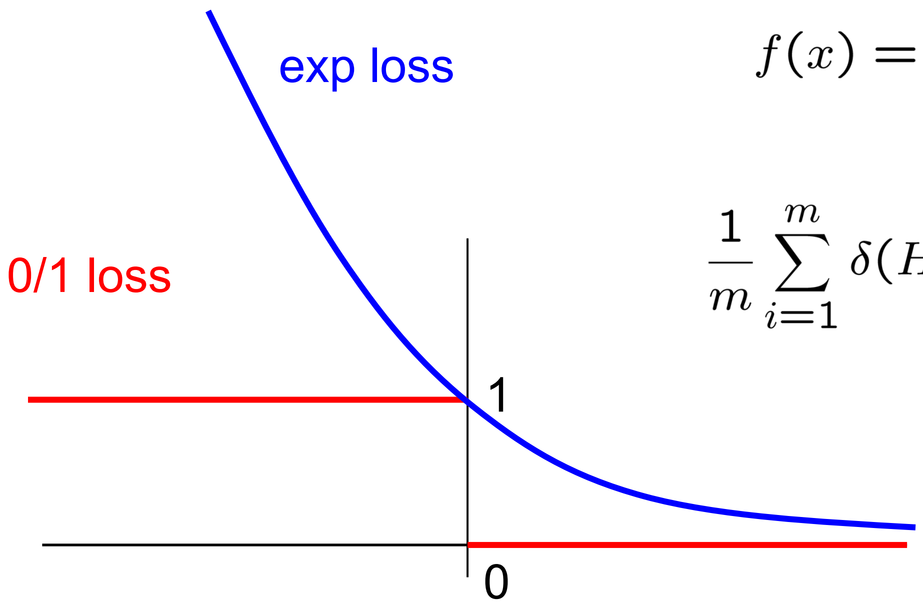


Boosting Example (Decision Stumps)



Analysis for Boosting

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



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

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

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.

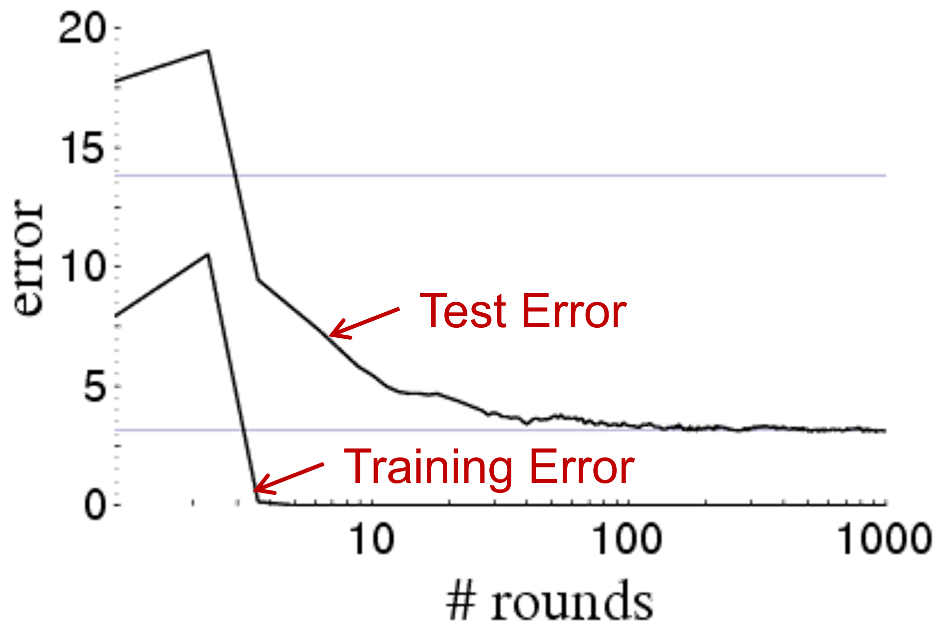
$$\frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i) \neq y_i) \leq \exp\left(-2\sum_{t=1}^{T}(1/2-\epsilon_t)^2\right)$$

Training Error

What about test error?

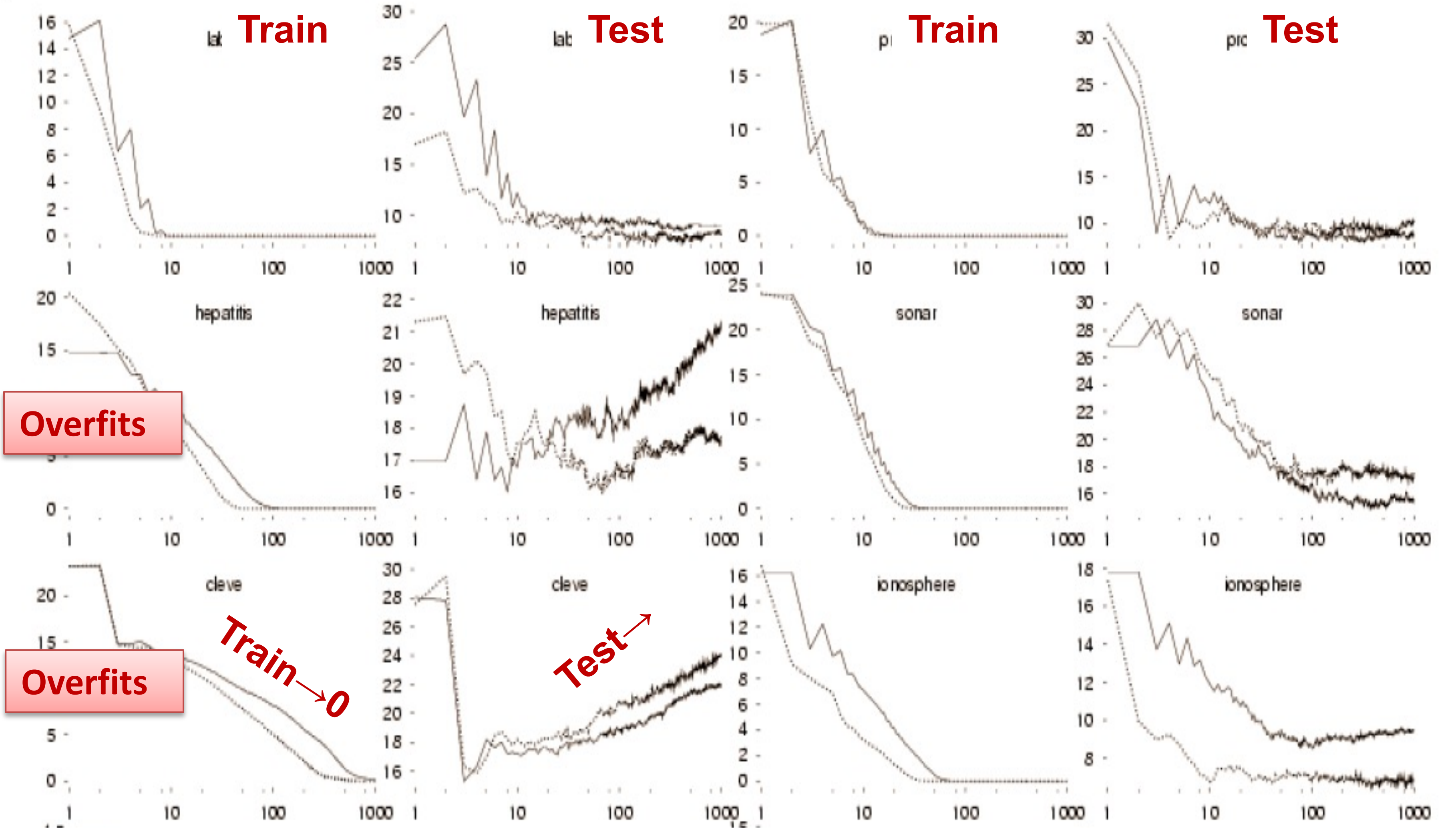
Boosting results – Digit recognition

[Schapire, 1989]



- Boosting often,
  - Robust to overfitting
  - Test set error decreases even after training error is zero
- If classes are well-separated, 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 classes not well separated (high label noise) or weak learners are too complex.

Boosting and Logistic Regression

Logistic regression equivalent to minimizing log loss

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

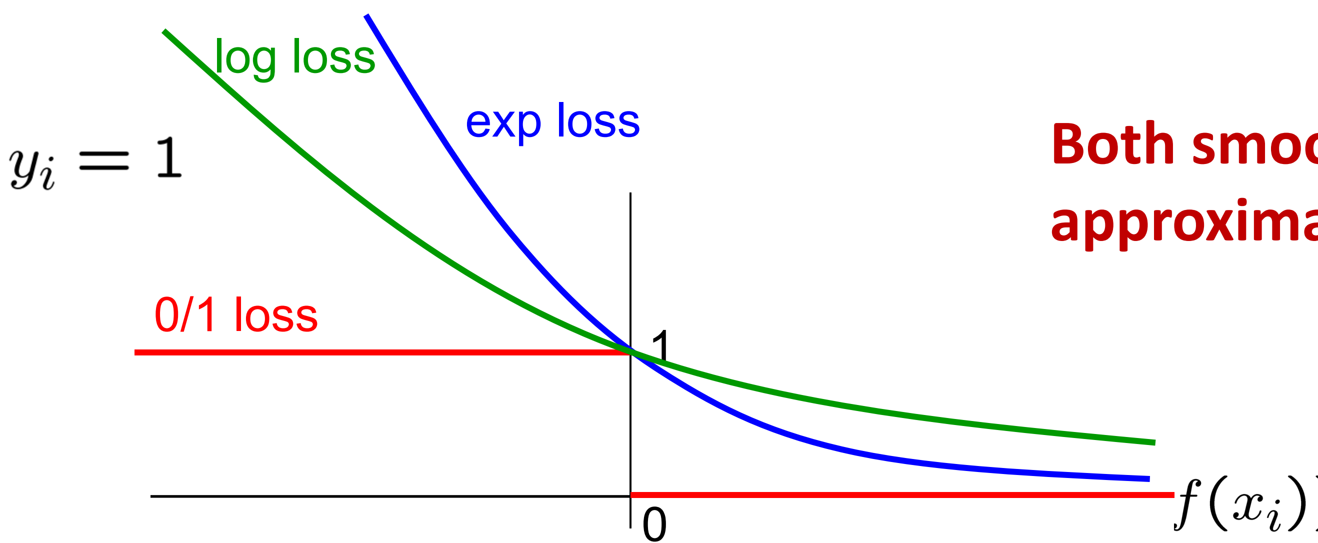
$$f(x) = w_0 + \sum_j w_j x_j$$

Boosting minimizes similar loss function!!

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

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

Weighted average of weak learners



Both smooth and convex approximations of 0/1 loss!

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 $w_0, w_1, w_2 \ldots$

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

(not a linear classifier)

- Weights α_t learned per iteration t incrementally

Hard & Soft Decision

Weighted average of weak learners $f(x) = \sum_t \alpha_t h_t(x)$

Hard Decision/Predicted label: $H(x) = sign(f(x))$

Soft Decision:
(based on analogy with
logistic regression)

$$P(Y = 1|X) = \frac{1}{1 + \exp(-f(x))}$$

Bagging (Bootstrap aggregating)

[Breiman, 1996]

Related approach to combining classifiers:

1. Run independent weak learners on subsampled data (sample with replacement) from the training set
2. Average/vote over weak hypotheses

Bagging vs.	Boosting
Resamples data points	Reweights data points (modifies their distribution)
Weight of each classifier is the same	Weight is dependent on classifier's accuracy
Only variance reduction	Both bias and variance reduced – learning rule becomes more complex with iterations
Can be trained in parallel	Trained sequentially

Random Forest

Related approach to combining decision trees:

1. Train decision trees on subsampled data (sample with replacement) from the training set + using **feature bagging** (random subset of features considered at each node)
2. Average/vote over decision trees

Random forest	vs. Boosted decision trees
Resamples data points	Reweights data points (modifies their distribution)
Weight of each classifier is the same	Weight is dependent on classifier's accuracy
Only variance reduction	Both bias and variance reduced – learning rule becomes more complex with iterations
Typically complex decision trees	Typically uses decision stumps
Can be trained in parallel	Trained sequentially

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