Machine Learning 10-601

Tom M. Mitchell
Machine Learning Department
Carnegie Mellon University

November 27, 2017

Today:

- Ensemble learning
- Mistake bounds on learning
- Weighted majority algorithm
- Boosting
- AdaBoost and Logistic Regr.

Recommended reading:

- Schapire's tutorial on Boosting
- Wikipedia Ensemble Learning page

some slides courtesy of Maria Balcan
some slides courtesy of Rob Shapire
some slides courtesy of Ziv Bar-Joseph

Ensemble Methods

- Key idea: k hypotheses might be better than one
- Examples:
  - Candidate elimination algorithm (aka Halving Algorithm)
  - Bayes optimal classifier
  - Weighted Majority algorithm
  - AdaBoost
  - Decision forests

Candidate Elimination Algorithm

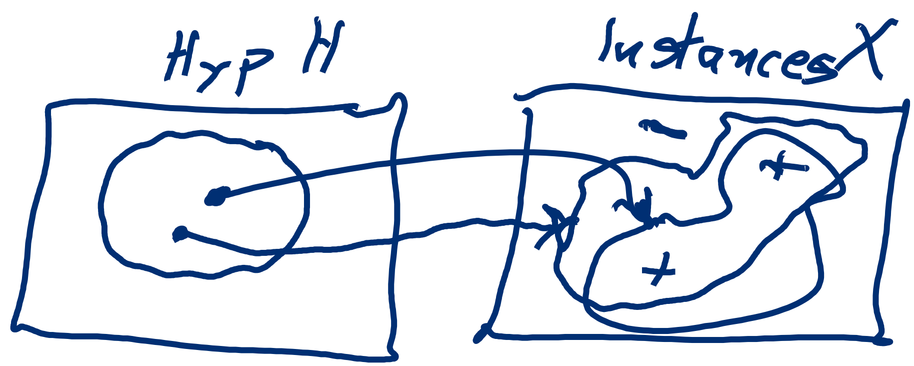
Candidate Elimination Algorithm (aka Halving Algorithm)

- Initialize Version Space VS ← H
- For each training example <x,y>
  - remove from VS every hypothesis that misclassifies <x,y>

Note remaining candidate hypotheses have zero training error

Classify new example x:

- every hypothesis remaining in VS votes equally
- y ← majority vote



Mistake Bounds: Halving Algorithm

1. Initialize VS ← H
2. For each training example,
   - remove from VS every hypothesis that misclassifies this example

Consider the Halving Algorithm:

- Learn concept using version space CANDIDATE-ELIMINATION algorithm
- Classify new instances by majority vote of version space members

How many mistakes before converging to correct h?

- ... in worst case? initially $|VS| = |H|$
- ... in best case? 0

after one mistake $|VS| \leq \frac{1}{2}|H|$

after k mistakes $|VS| \leq \left(\frac{1}{2}\right)^k |H|$

$k \leq \lfloor \log_2 |H| \rfloor$

Optimal Mistake Bounds

possible concepts to be learned $\sim H$

Let $M_A(C)$ be the max number of mistakes made by algorithm A to learn concepts in C. (maximum over all possible $c \in C$, and all possible training sequences)

$$M_A(C) \equiv \max_{c \in C} M_A(c)$$

Definition: Let C be an arbitrary non-empty concept class. The **optimal mistake bound** for C, denoted $Opt(C)$, is the minimum over all possible learning algorithms A of $M_A(C)$.

$$Opt(C) \equiv \min_{A \in learning\ algorithms} M_A(C)$$

$$\boxed{VC(C) \leq Opt(C) \leq M_{Halving}(C) \leq log_2(|C|).}$$

Weighted Majority Algorithm

a_i denotes the i^{th} prediction algorithm in the pool A of algorithms. w_i denotes the weight associated with a_i.

vote mass for $y=0$

for $y=1$

- For all i initialize $w_i \leftarrow 1$
- For each training example $\langle x, c(x) \rangle$
  - $*$ Initialize q_0 and q_1 to 0
  - $*$ For each prediction algorithm a_i
    - $\cdot$ If $a_i(x) = 0$ then $q_0 \leftarrow q_0 + w_i$

      If $a_i(x) = 1$ then $q_1 \leftarrow q_1 + w_i$
  - $*$ If $q_1 > q_0$ then predict $c(x) = 1$

    If $q_0 > q_1$ then predict $c(x) = 0$

    If $q_1 = q_0$ then predict 0 or 1 at random for $c(x)$
  - $*$ For each prediction algorithm a_i in A do

    If $a_i(x) \neq c(x)$ then $w_i \leftarrow \beta w_i$

when β=0, equivalent to the Halving algorithm…

Weighted Majority

Even algorithms that learn or change over time…

[Relative mistake bound for WEIGHTED-MAJORITY] Let D be any sequence of training examples, let A be any set of n prediction algorithms, and let k be the minimum number of mistakes made by any algorithm in A for the training sequence D. Then the number of mistakes over D made by the WEIGHTED-MAJORITY algorithm using $\beta = \frac{1}{2}$ is at most

$$2.4(k + \log_2 n)$$

$\beta = 1/2$ init $w_i = 1$

let M = number of mistakes by WtdMaj

let a_j be secretly the best (fewest mistakes) alg

a_j makes k mistakes

A is collection of n algs.

① what is final weight for a_j? $\boxed{\left(\frac{1}{2}\right)^k}$

② let $W = \sum_{i=1}^{n} w_i$ at end of training

initially $\boxed{W = n}$

after one mistake $W \leq \frac{3}{4} W = \frac{3}{4} n$

0.5 W vote wrong → 0.25 W

after M mistakes $W \leq \left(\frac{3}{4}\right)^M n$

③ final wt of $a_j \leq$ final wt mass

$$\left(\frac{1}{2}\right)^k \leq \left(\frac{3}{4}\right)^M n$$

Boosting

- Weighted Majority algorithm learns weight for each given predictor/hypothesis
  - It's an example of an ensemble method: combines predictions of *multiple* hypotheses

- Boosting learns weight, and also the hypotheses

- Leads to one of the most popular learning methods in practice: Decision Forests

Boosting: Key Idea

[Rob Schapire]

- Use a learner that produces better-than-chance h(x)'s
- Train it multiple times, on reweighted training examples
  - Each time, upweight the incorrectly classified examples, downweight the correctly classified examples
- Final prediction: weighted vote of the multiple $h_i(x)$'s

- Practically useful
- Theoretically interesting

AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$$

training error

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

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

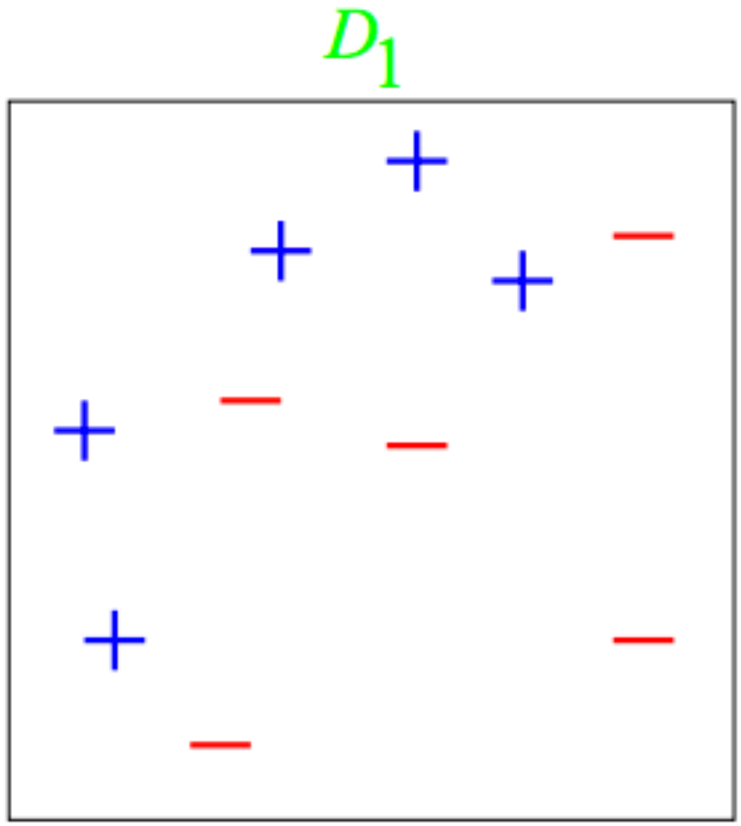
where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

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

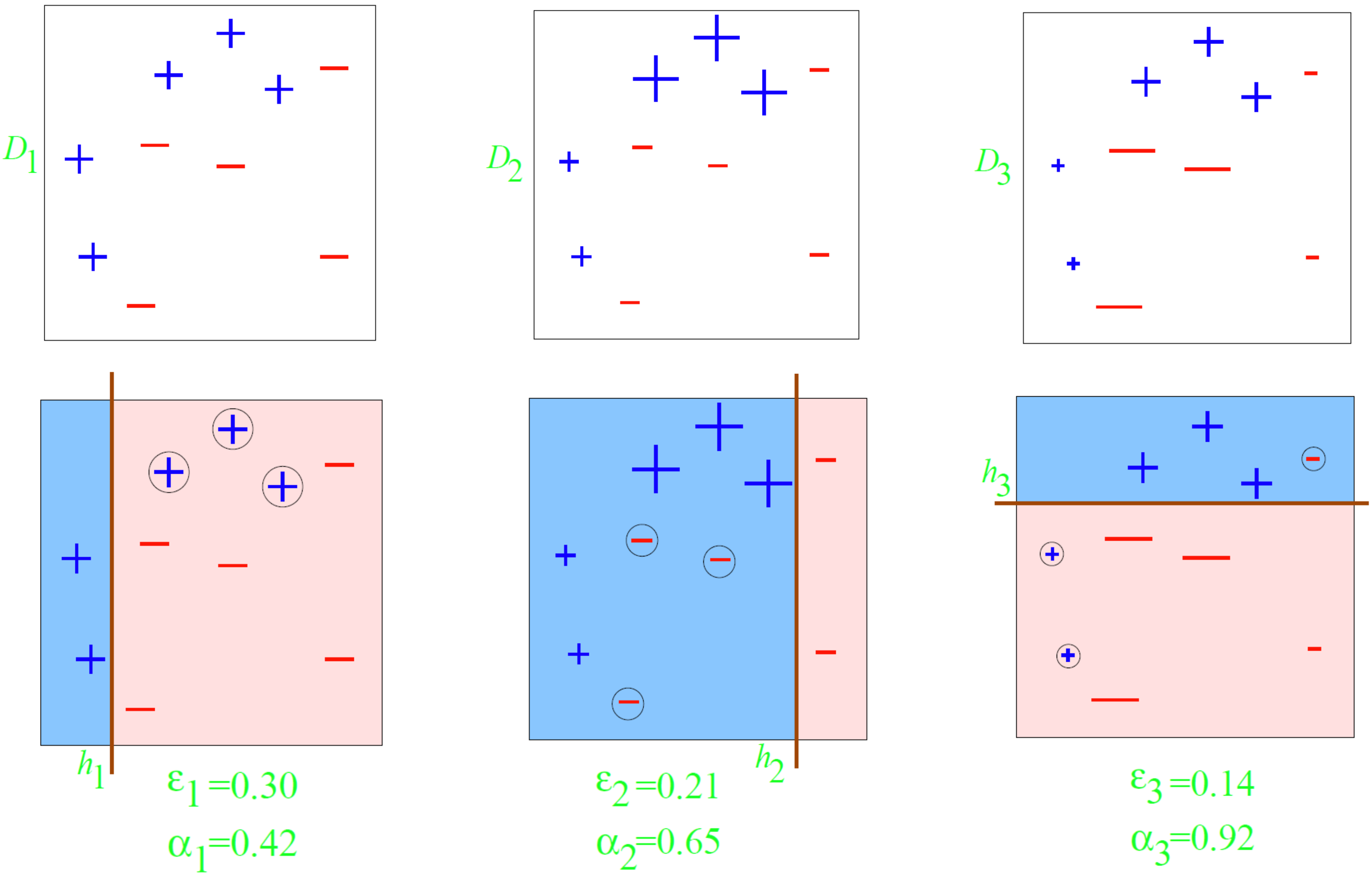
AdaBoost: A toy example

Weak classifiers: vertical or horizontal half-planes (a.k.a. decision stumps)

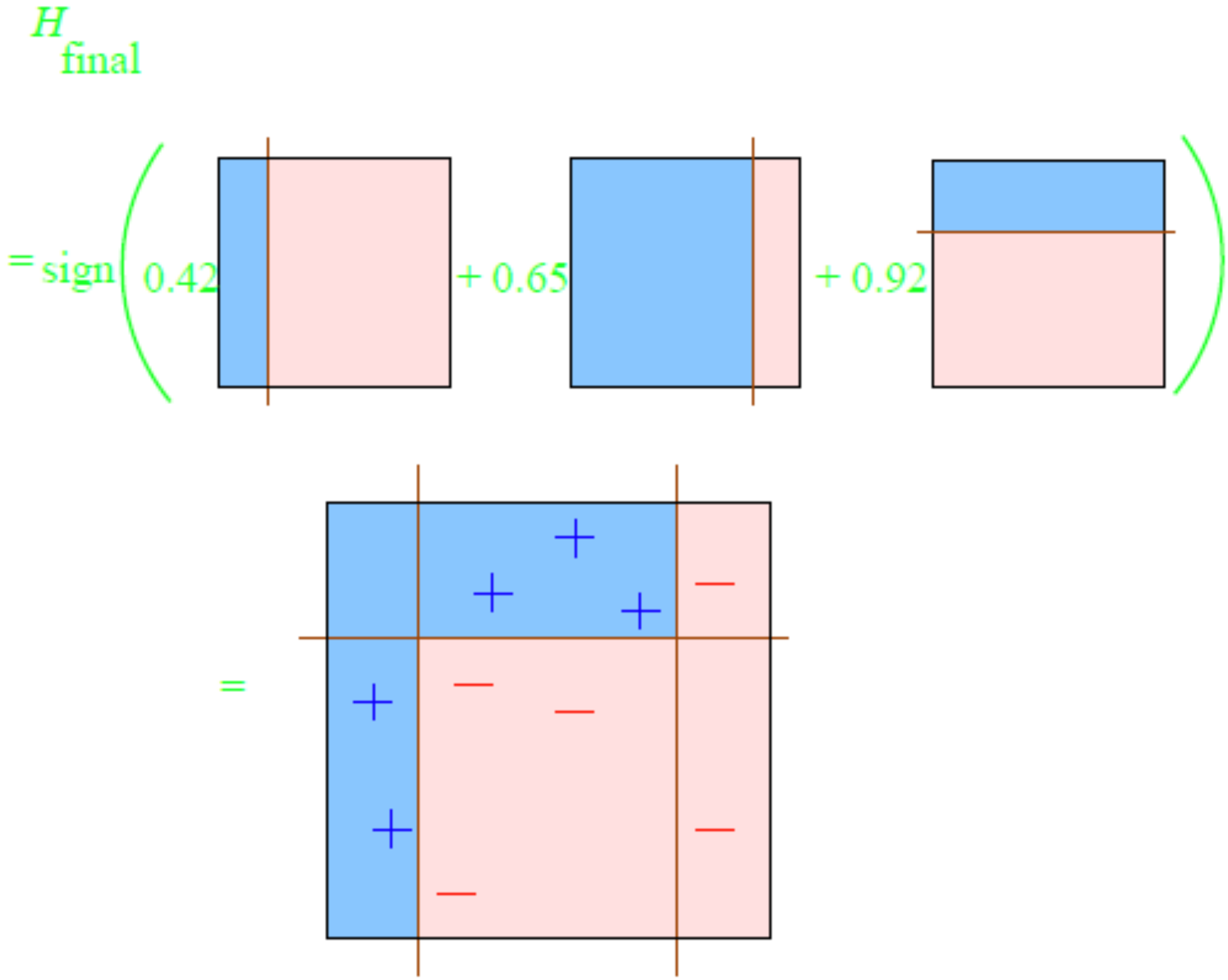


[Rob Schapire]

AdaBoost: A toy example



AdaBoost: A toy example



[Rob Schapire]

AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$$

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

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

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

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

Training Error

note this is > 1 if
classification error

Training error of final classifier is bounded by:

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

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

Theoretical Result 1: Training Error

Training error of final classifier is bounded by:

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

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

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

Training error of final classifier is bounded by:

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

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

If we minimize $\prod_t Z_t$, we minimize our training error

We can tighten this bound greedily, by choosing α_t and h_t on each iteration to minimize Z_t.

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

Theoretical Result 1: Training Error

We can minimize this bound by choosing α_t on each iteration to minimize Z_t.

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

For boolean target function, this is accomplished by [Freund & Schapire '97]:

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

Where:

$$\boxed{\epsilon_t = \sum_{i=1}^{m} D_t(i) \delta(h_t(x_i) \neq y_i)}$$

AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$$

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

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

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

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

Bound on True Error

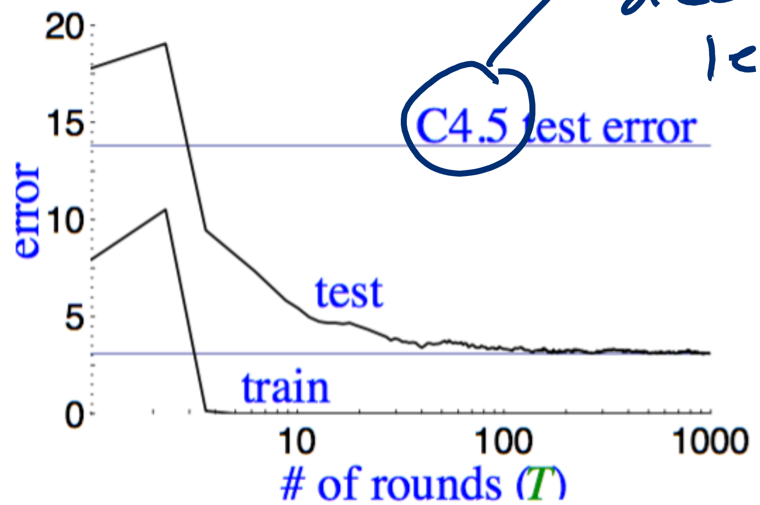
[Freund & Shapire, 1999]

With high probability:

$$error_{true}\left(sign(\sum_t \alpha_t h_t(x))\right) \leq error_{train}\left(sign(\sum_t \alpha_t h_t(x))\right) + O\left(\sqrt{\frac{T \cdot VCdim(H)}{m}}\right)$$

Actual Typical Run

standard dec. tree learning alg.



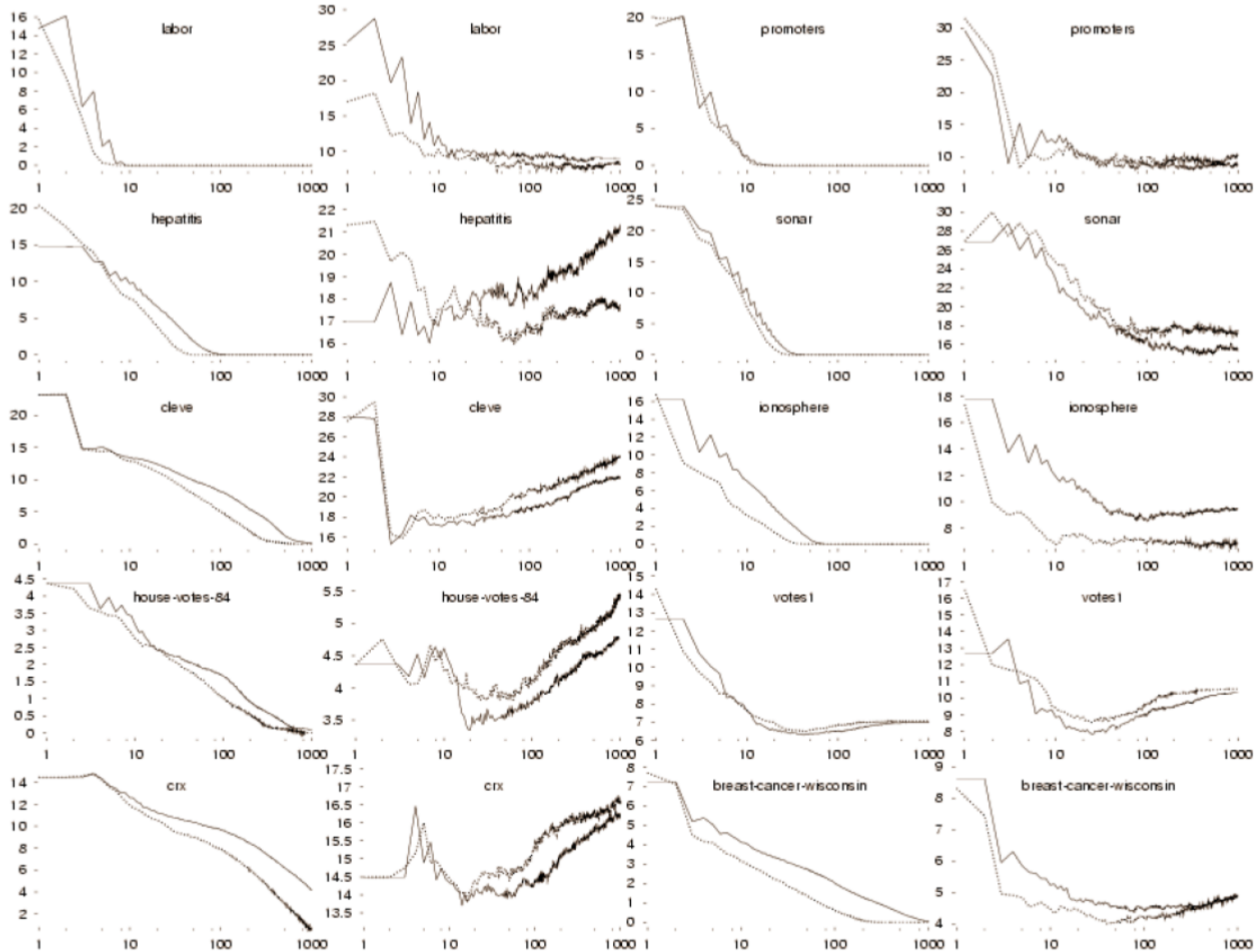
(boosting C4.5 on "letter" dataset)

- test error does not increase, even after 1000 rounds
  - (total size > 2,000,000 nodes)
- test error continues to drop even after training error is zero!

	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1

[Rob Shapire]

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



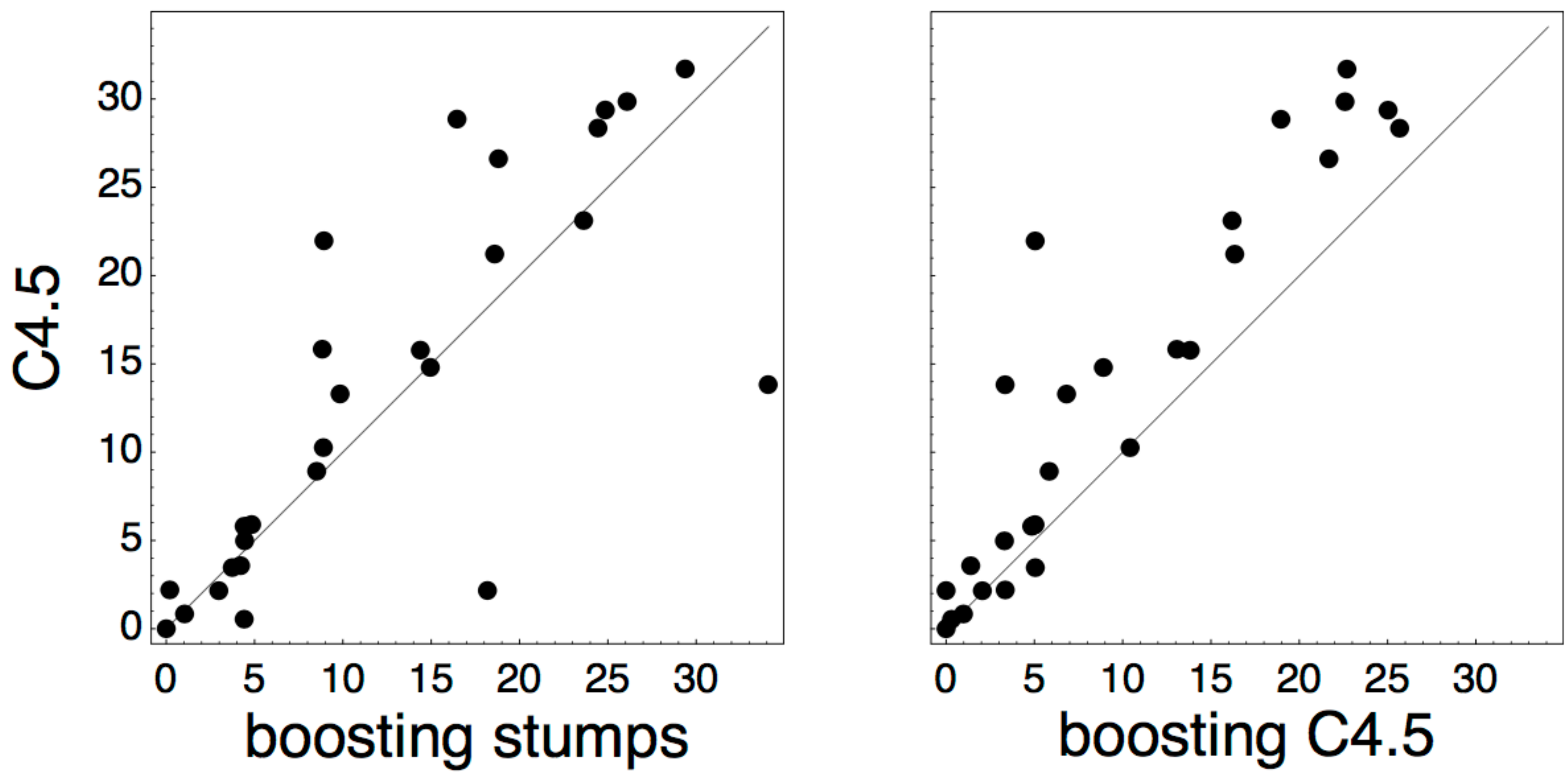
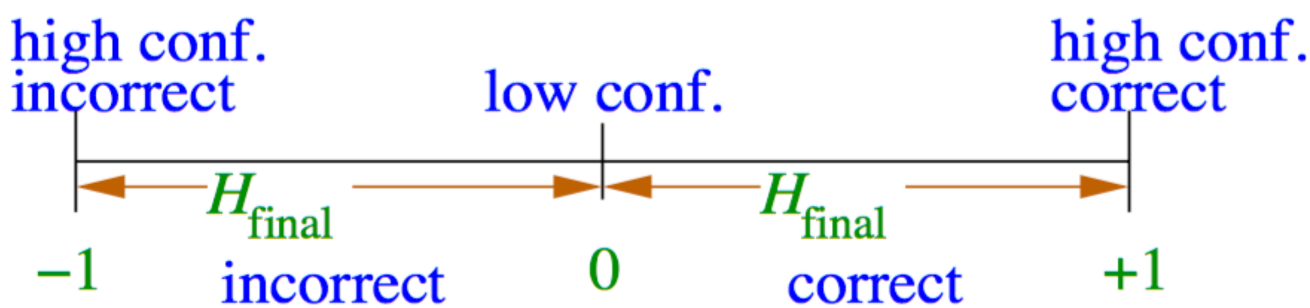


Figure 3: Comparison of C4.5 versus boosting stumps and boosting C4.5 on a set of 27 benchmark problems as reported by Freund and Schapire [21]. Each point in each scatterplot shows the test error rate of the two competing algorithms on a single benchmark. The y-coordinate of each point gives the test error rate (in percent) of C4.5 on the given benchmark, and the x-coordinate gives the error rate of boosting stumps (left plot) or boosting C4.5 (right plot). All error rates have been averaged over multiple runs.

A Better Story: Theory of Margins

[with Freund, Bartlett & Lee]

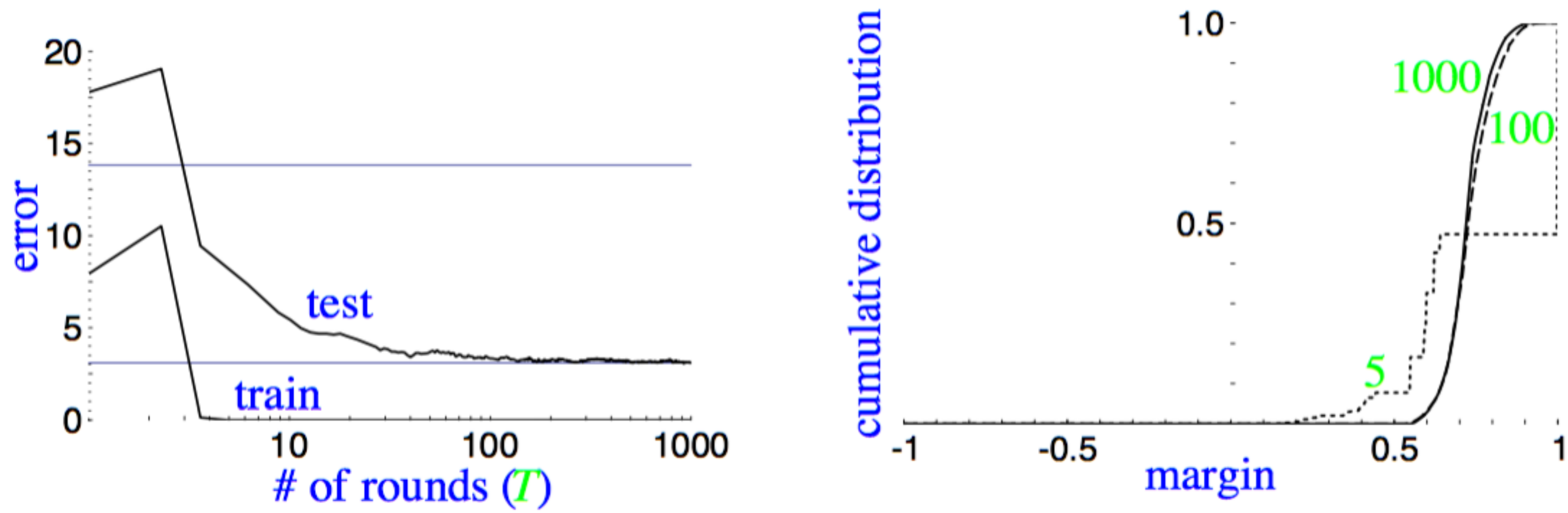
- key idea:
  - training error only measures whether classifications are right or wrong
  - should also consider confidence of classifications
- recall: H_{final} is weighted majority vote of weak classifiers
- measure confidence by margin = strength of the vote
  = (fraction voting correctly) − (fraction voting incorrectly)



[Rob Schapire]

Empirical Evidence: The Margin Distribution

- margin distribution
  = cumulative distribution of margins of training examples



	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1
% margins ≤ 0.5	7.7	0.0	0.0
minimum margin	0.14	0.52	0.55

[Rob Schapire]

Bounds on Generalization Error in Boosting

[Freund & Shapire, 1999]

With high probability

$$error_{true}\left(sign(\sum_t \alpha_t h_t(x))\right) \leq error_{train}\left(sign(\sum_t \alpha_t h_t(x))\right) + O\left(\sqrt{\frac{T \cdot VCdim(H)}{m}}\right)$$

[Shapire, et al., 1999]

For all $\theta > 0$, with high probability:

$$error_{true}\left(sign(\sum_t \alpha_t h_t(x))\right) \leq P_{train}[margin_f(x,y) \leq \theta] + O\left(\sqrt{\frac{VCdim(H)}{m\theta^2}}\right)$$

Margin based: Independent of T !!

Boosting and Logistic Regression

Logistic regression assumes:

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

And tries to maximize data likelihood:

$$P(\mathcal{D}|H) = \prod_{i=1}^{m} \frac{1}{1 + \exp(-y_i f(x_i))}$$

Equivalent to minimizing log loss

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

Boosting and Logistic Regression

Logistic regression equivalent to minimizing log loss

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

Boosting minimizes similar loss function!!

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

Both smooth approximations of 0/1 loss!

Logistic regression and Boosting

Logistic regression:

- Minimize loss fn

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

- Define

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

where x_j predefined

Boosting:

- Minimize loss fn

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

- Define

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

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

(not a linear classifier)

- Weights α_j learned incrementally

Slack variables – Hinge loss

Complexity penalization

$$\xi_j = \mathrm{loss}(f(x_j), y_j)$$

$$f(x_j) = \mathrm{sgn}(\mathbf{w} \cdot \mathrm{x_j} + \mathrm{b})$$

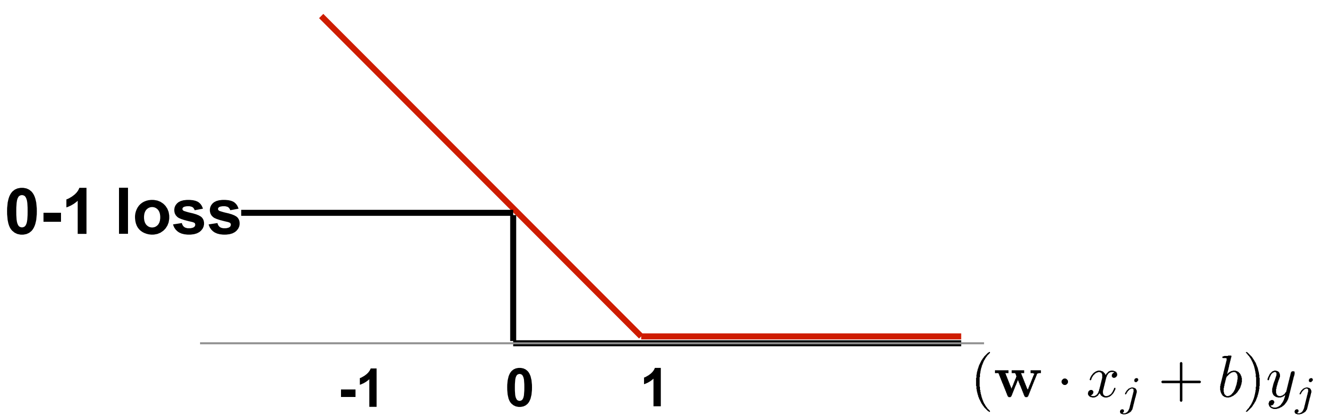
$$\min_{\mathbf{w},b} \ \mathbf{w}^T\mathbf{w} + C \sum_j \xi_j$$

$$\text{s.t. } (\mathbf{w}^T\mathbf{x}_j + b)\, y_j \geq 1 - \xi_j \quad \forall j$$

$$\xi_j \geq 0 \quad \forall j$$

$$\xi_j = (1 - (\mathbf{w} \cdot x_j + b) y_j))_+$$

Hinge loss



SVM vs. Logistic Regression

SVM : **Hinge loss**

$$\text{loss}(f(x_j), y_j) = (1 - (\mathbf{w} \cdot x_j + b) y_j))_+$$

Logistic Regression : **Log loss** (negative log conditional likelihooc

$$\text{loss}(f(x_j), y_j) = -\log P(y_j \mid x_j, \mathbf{w}, b) = \log(1 + e^{-(\mathbf{w} \cdot x_j + b) y_j})$$

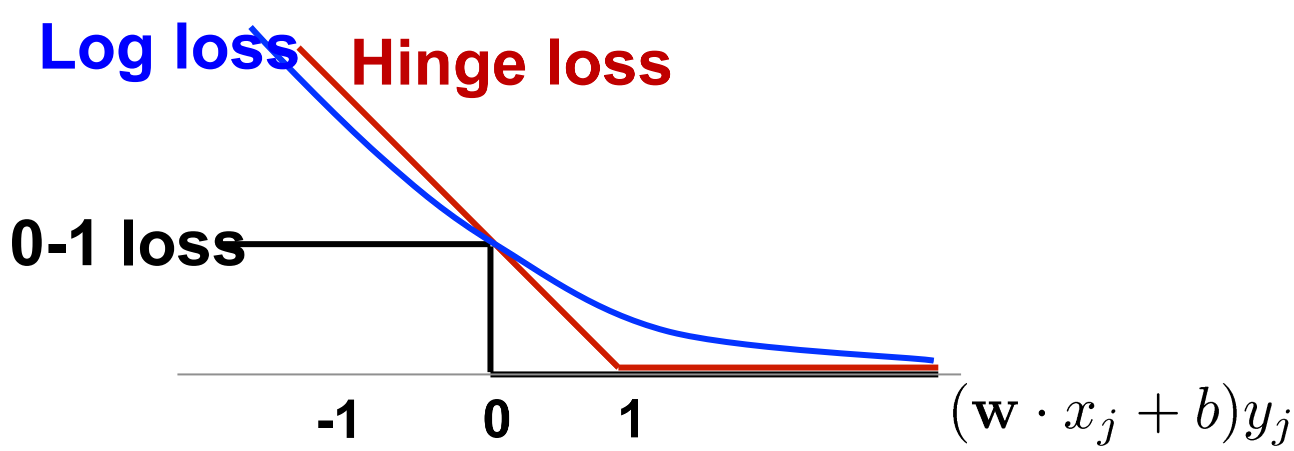
Boosting : $loss(f(x_j), y_j) = \exp(-y_j f(x_j))$

SVM : **Hinge loss**

$$\text{loss}(f(x_j), y_j) = (1 - (\mathbf{w} \cdot x_j + b)y_j))_+$$

Logistic Regression : **Log loss** (negative log conditional likelihooc

$$\text{loss}(f(x_j), y_j) = -\log P(y_j \mid x_j, \mathbf{w}, b) = \log(1 + e^{-(\mathbf{w} \cdot x_j + b)y_j})$$



What You Should Know

- Ensemble methods

- Weighted Majority
  - Learns weights for a given pool of hypotheses
  - Mistake bound relative to best hypothesis in the pool
  - …

- Boosting
  - Learns weigths and hypotheses
  - Theory: training error, true error, correspondence to Log. Regression
  - Practice: Boosted decision trees (and stumps) very popular!

- Many variants of ensemble methods
  - Resample training data to generate variety
  - Randomize learning algorithm to generate variety
  - Active learning – choose examples where vote is closest to tie