CSE446: Ensemble Learning - Bagging and Boosting
Winter 2015

Luke Zettlemoyer

Slides adapted from Carlos Guestrin, Nick Kushmerick, Padraig Cunningham

Voting (Ensemble Methods)

- Instead of learning a single classifier, learn **many weak classifiers** that are **good at different parts of the data**
- **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???**
  - force classifiers to learn about different parts of the input space? different subsets of the data?
  - weigh the votes of different classifiers?

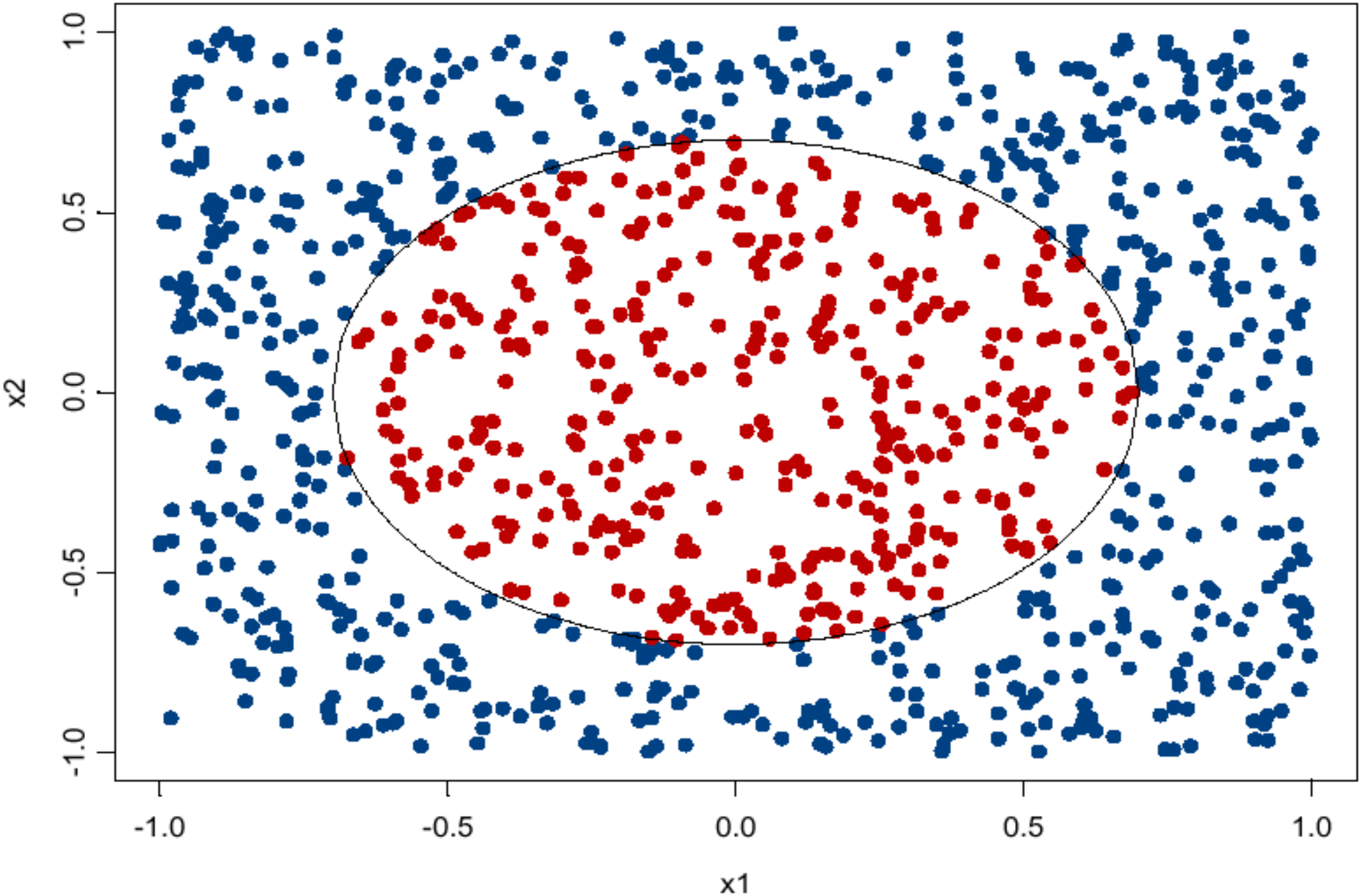
BAGGing = Bootstrap AGGregation

(Breiman, 1996)

- for i = 1, 2, …, K:
  - $T_i \leftarrow$ randomly select M training instances with replacement
  - $h_i \leftarrow$ learn(T_i) *[Decision Tree, Naive Bayes, …]*

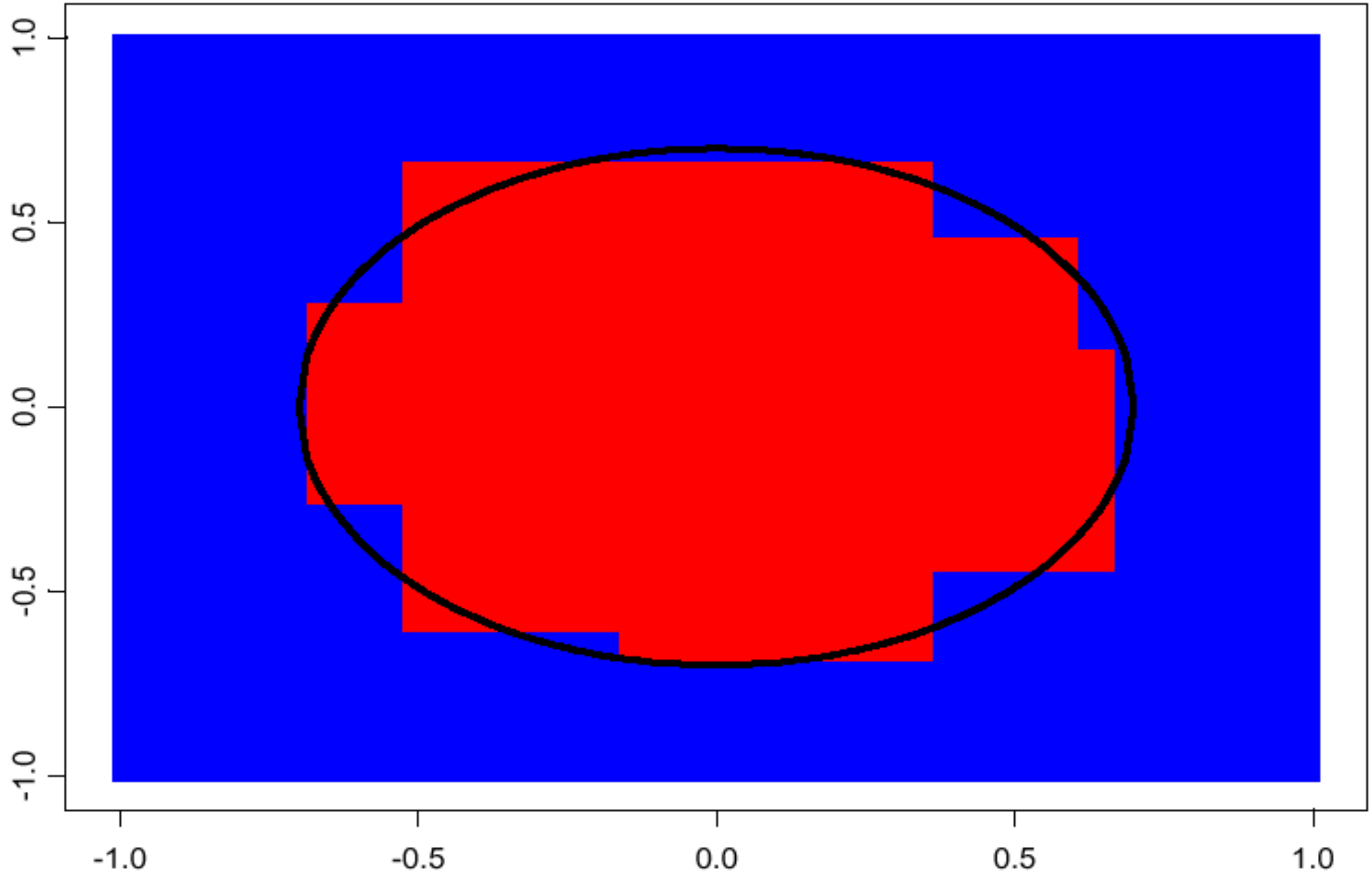
- Now combine the h_i together with uniform voting (w_i=1/K for all i)

Bagging Example

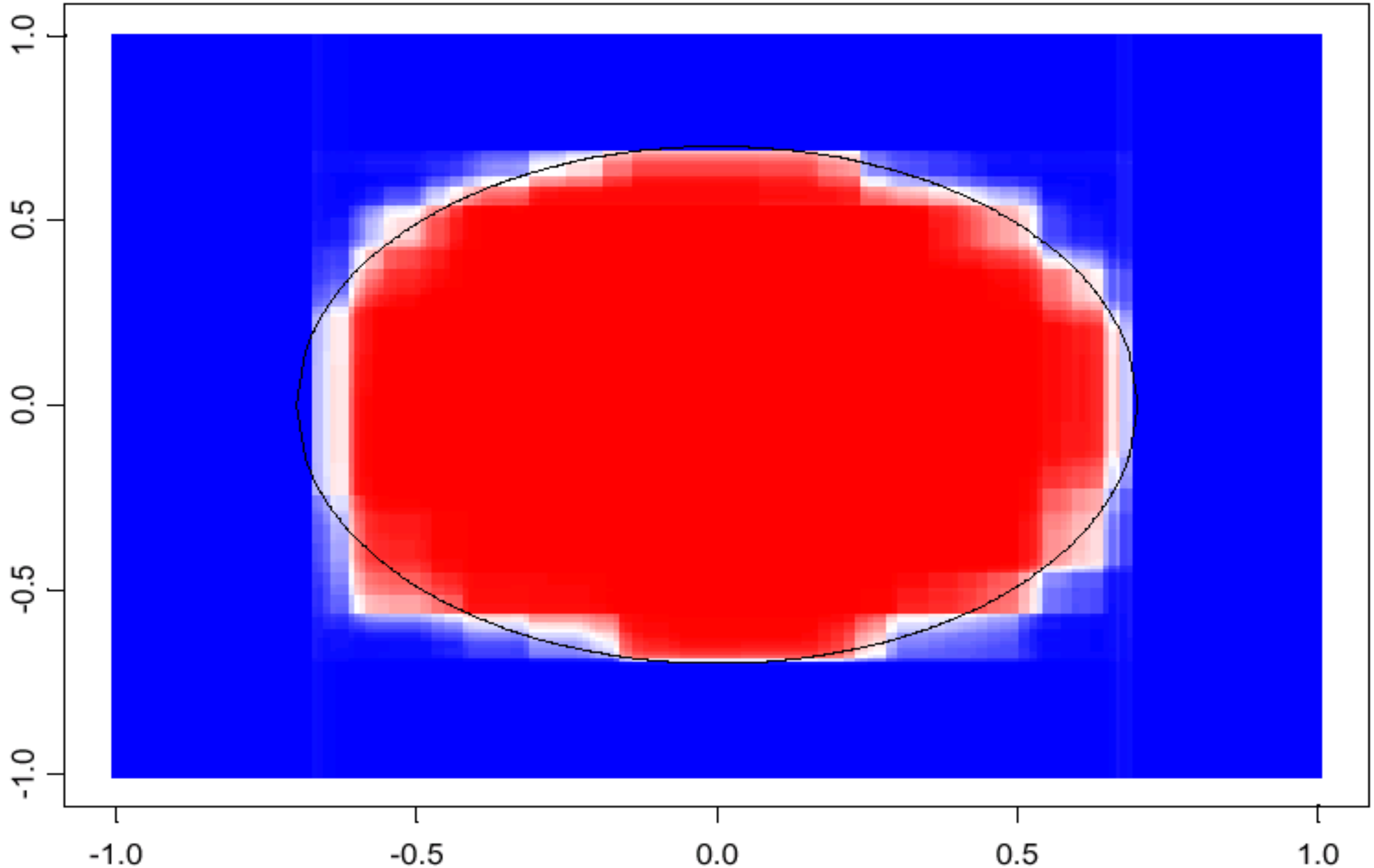


decision tree learning algorithm; very similar to version in earlier slides

CART decision boundary



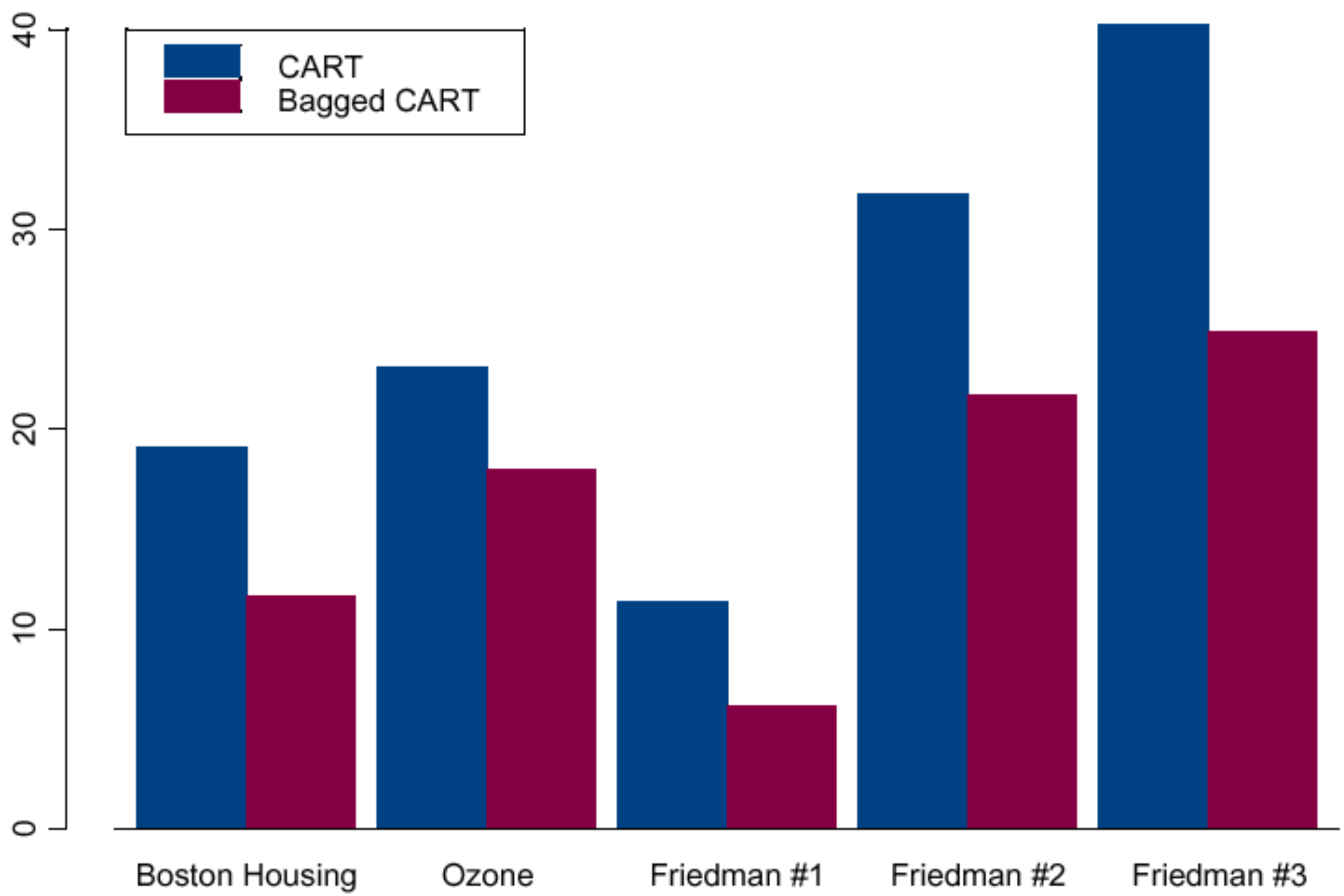
100 bagged trees



shades of blue/red indicate strength of vote for particular classification

Regression results

Squared error loss



Fighting the bias-variance tradeoff

- **Simple (a.k.a. weak) learners are good**
  - e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)
  - Low variance, don't usually overfit
- **Simple (a.k.a. weak) learners are bad**
  - High bias, can't solve hard learning problems
- Can we make weak learners always good???
  - **No!!!**
  - **But often yes…**

Boosting [Schapire, 1989]

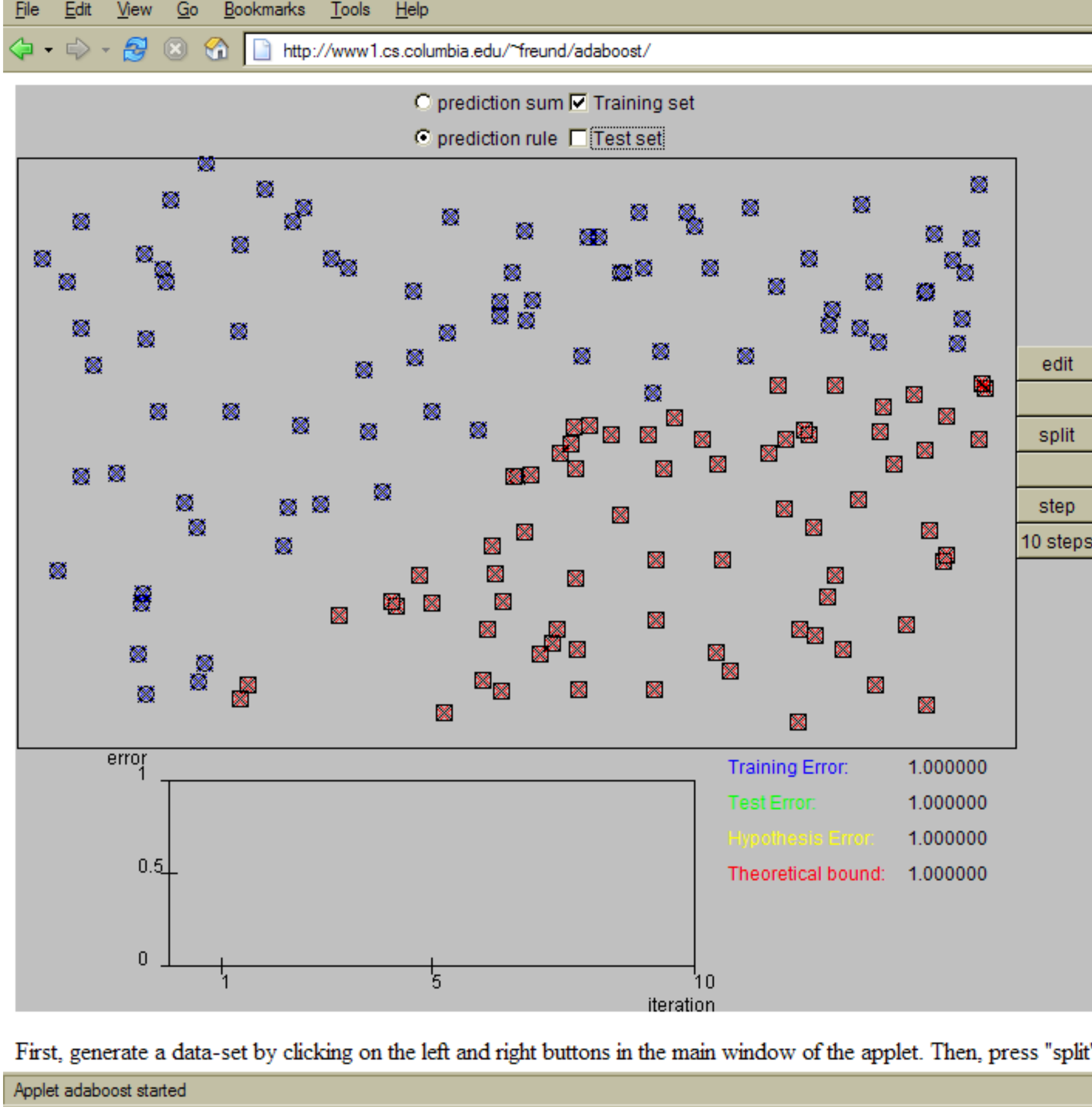
- Idea: given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote

- On each iteration *t*:
  - weight each training example by how incorrectly it was classified
  - Learn a hypothesis – h_t
  - A strength for this hypothesis – α_t

- Final classifier:

$$\boxed{h(x) = \text{sign}\left(\sum_i \alpha_i h_i(x)\right)}$$

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

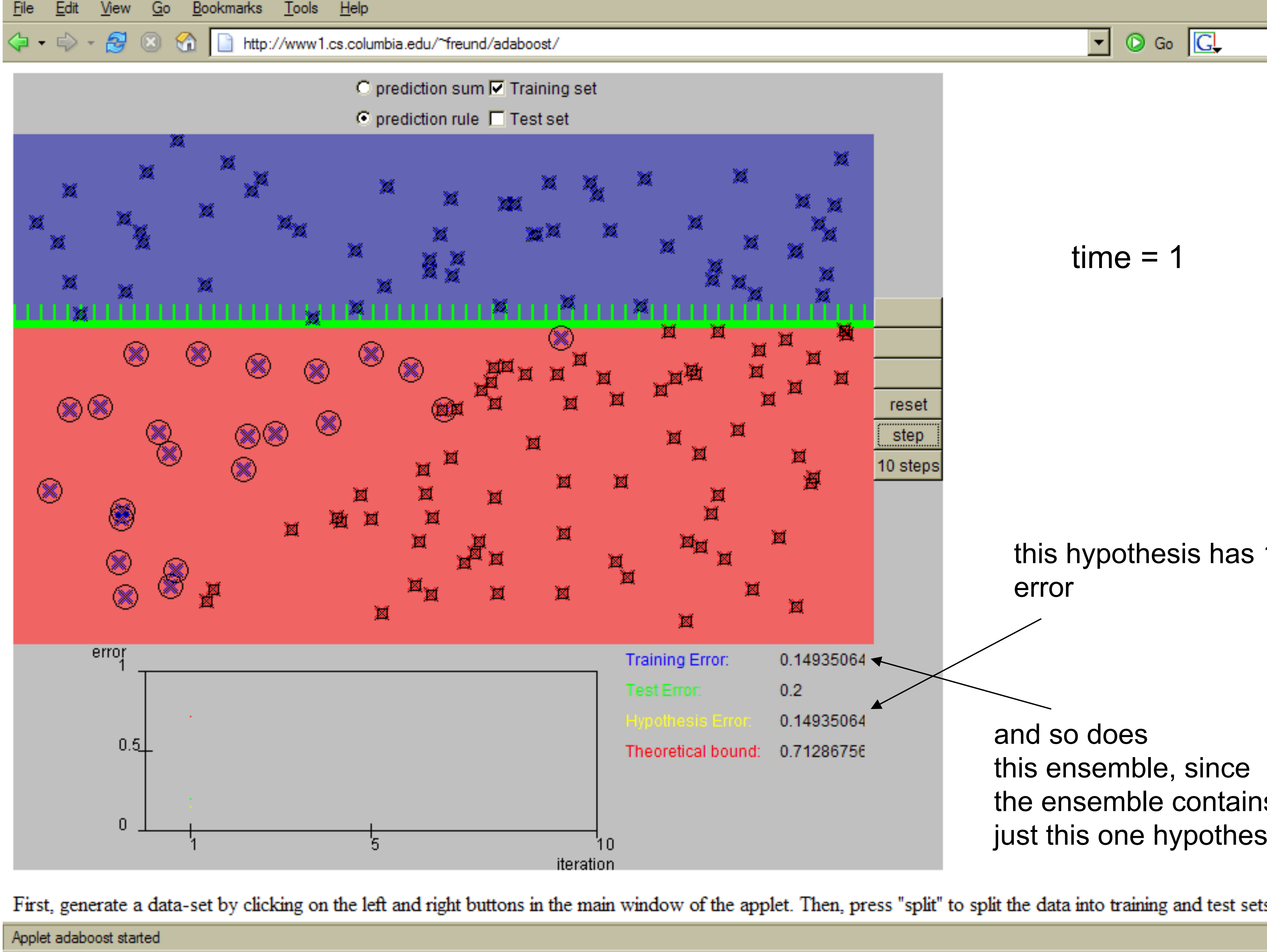


time = 0

blue/red = class

size of dot = weight

weak learner =
Decision stub:
horizontal or vertica



time = 1

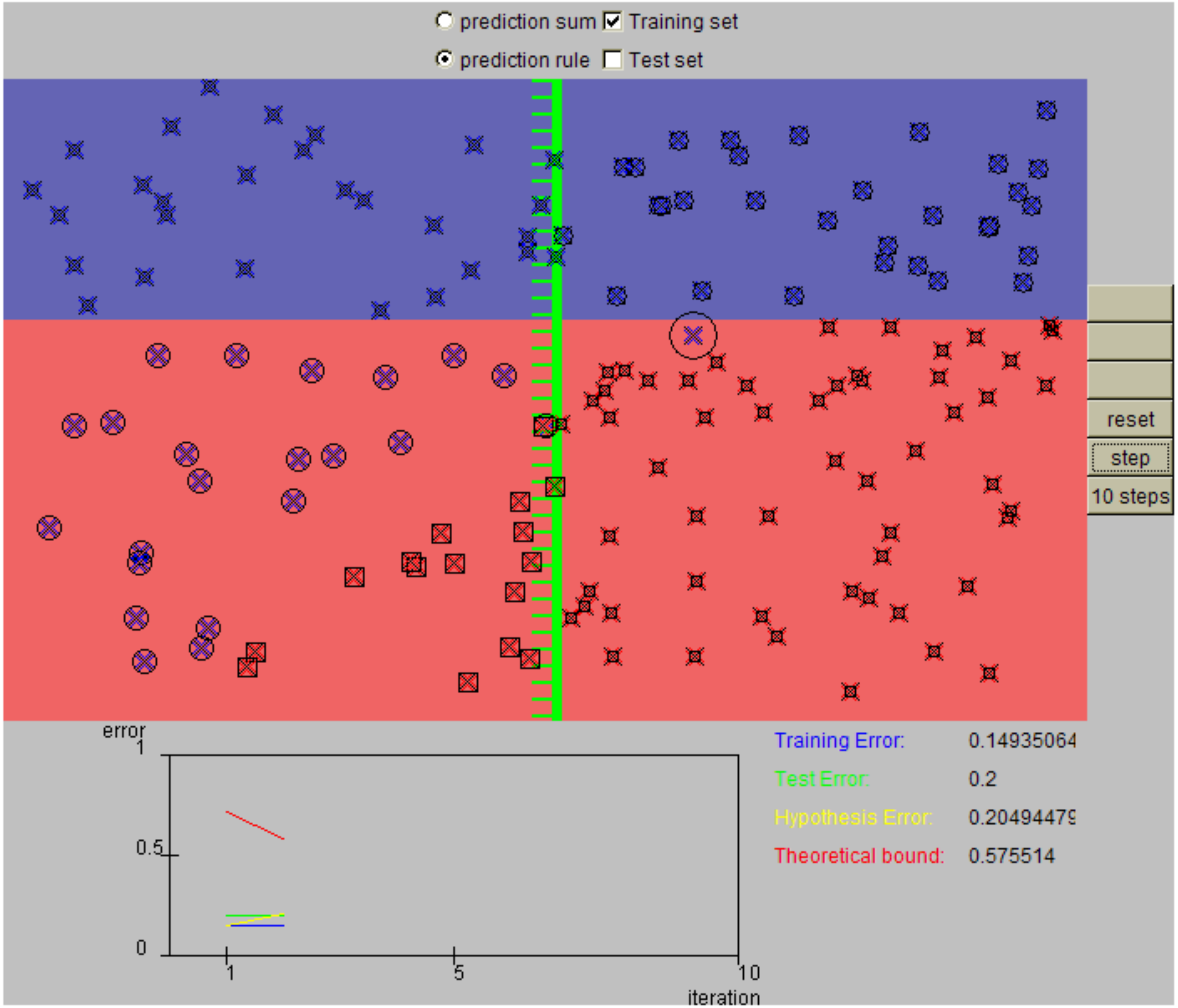
this hypothesis has 1
error

and so does
this ensemble, since
the ensemble contains
just this one hypothes

File Edit View Go Bookmarks Tools Help

http://www1.cs.columbia.edu/~freund/adaboost/

Go



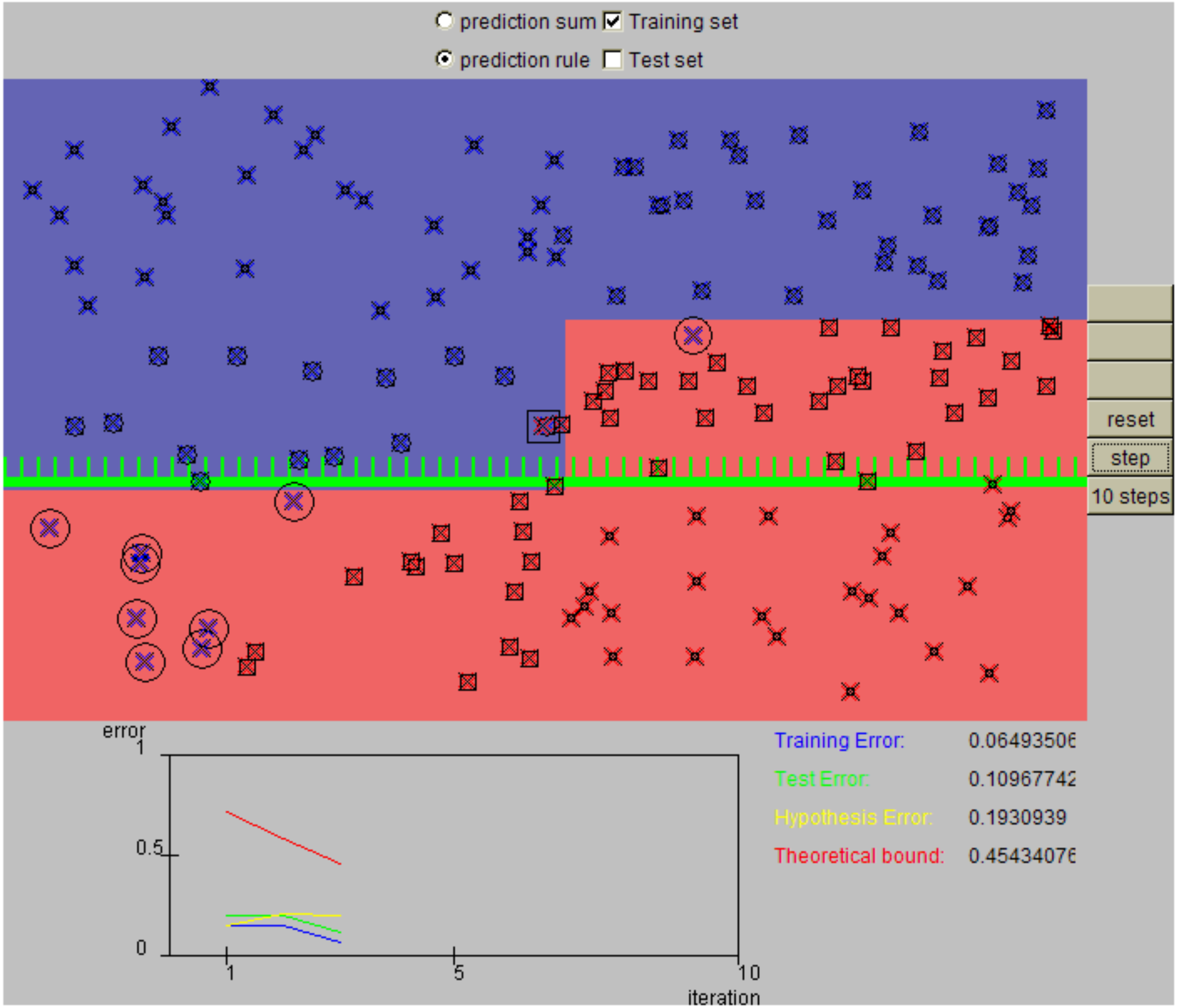
time = 2

First, generate a data-set by clicking on the left and right buttons in the main window of the applet. Then, press "split" to split the data into training and test sets

Applet adaboost started

File Edit View Go Bookmarks Tools Help

http://www1.cs.columbia.edu/~freund/adaboost/



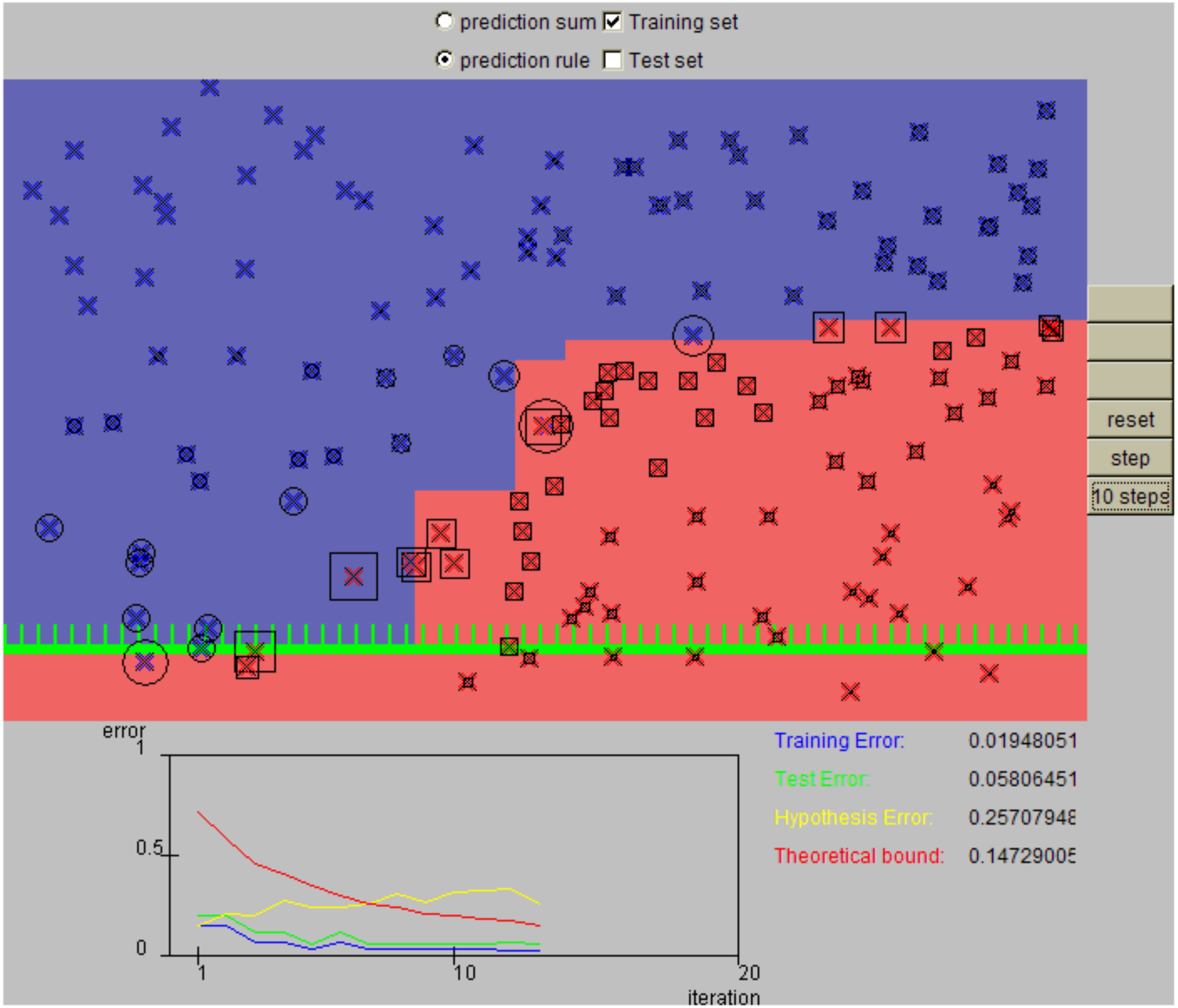
time = 3

First, generate a data-set by clicking on the left and right buttons in the main window of the applet. Then, press "split" to split the data into training and test sets

Applet adaboost started

File Edit View Go Bookmarks Tools Help

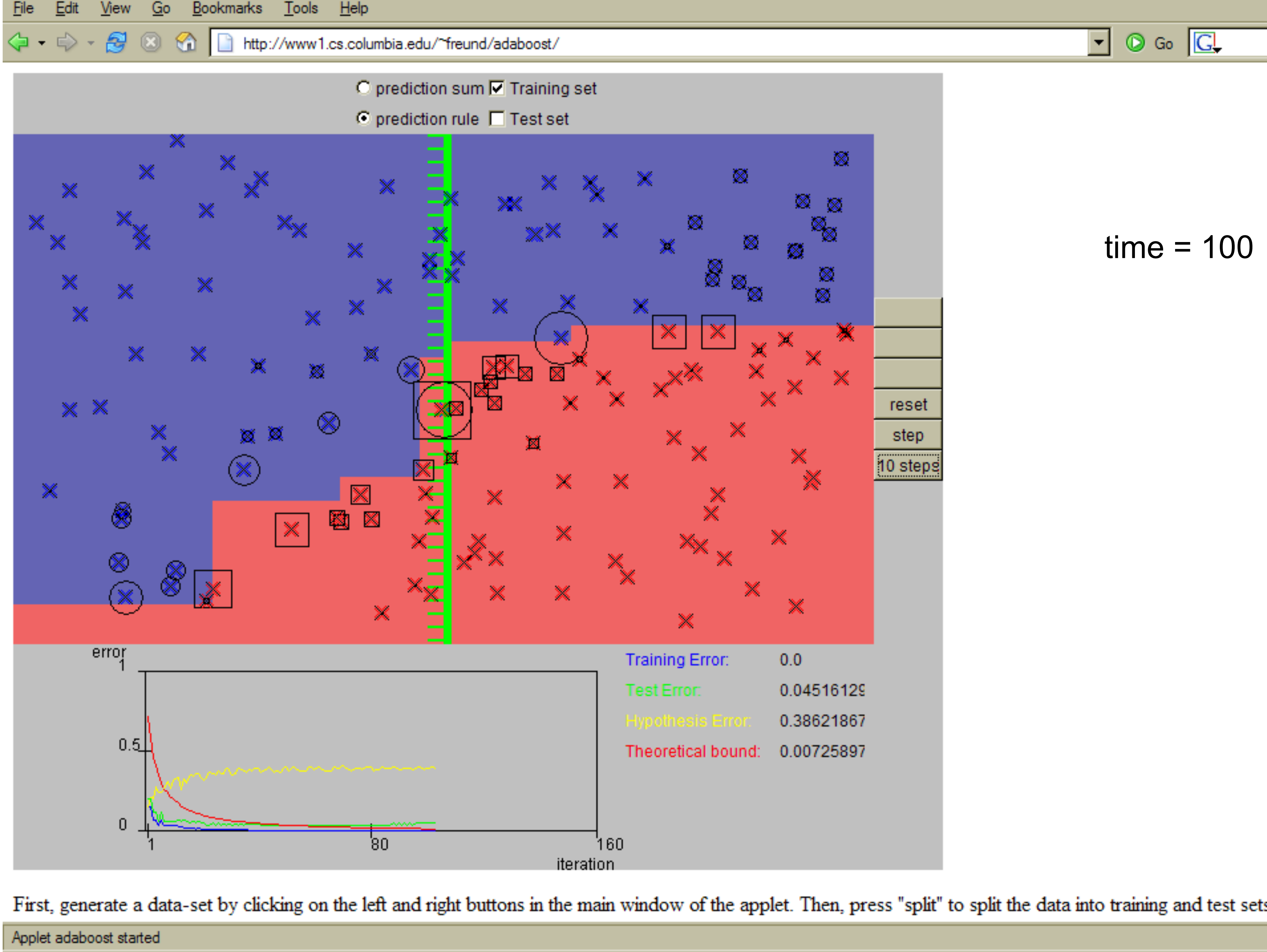
http://www1.cs.columbia.edu/~freund/adaboost/



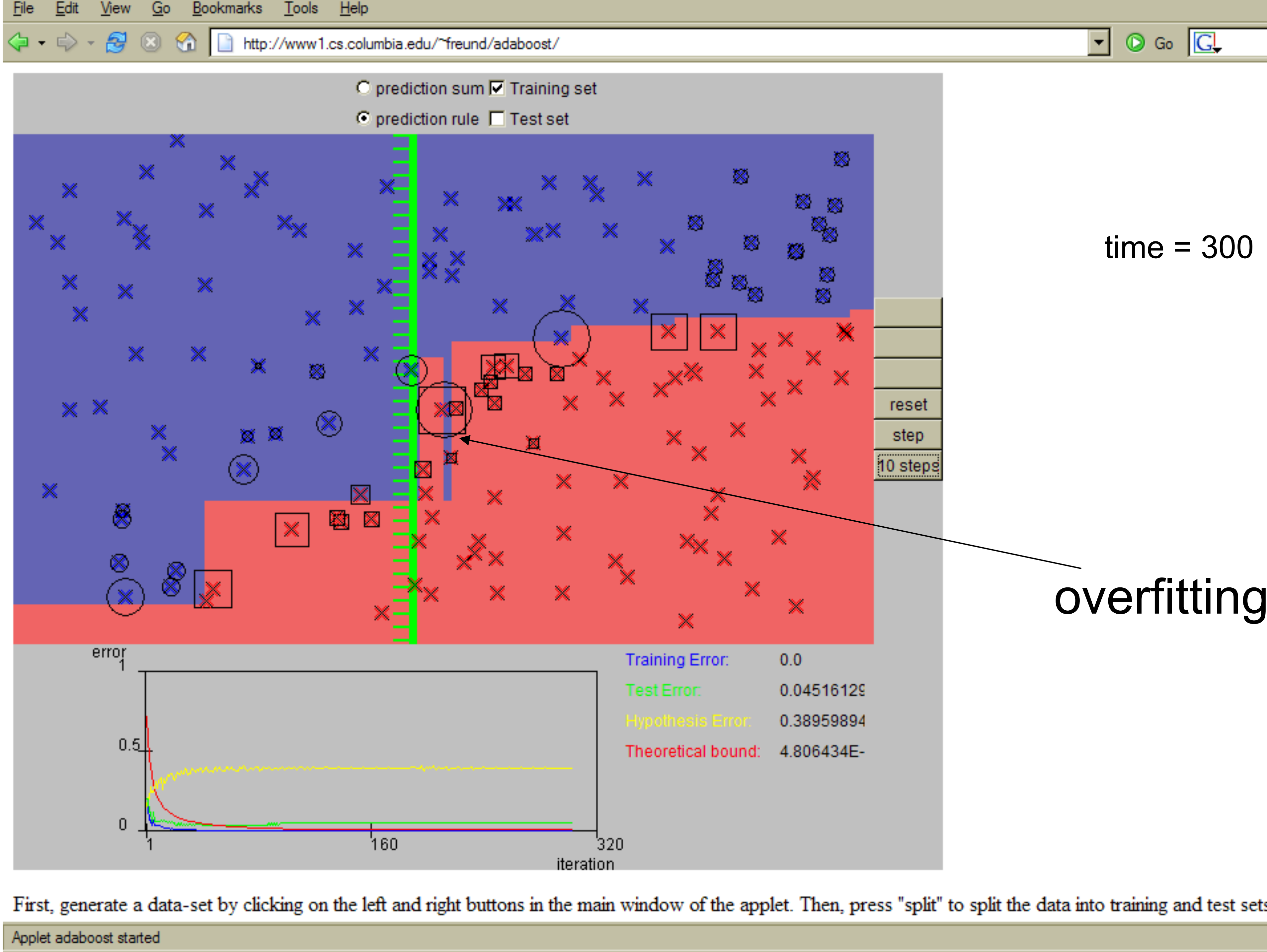
time = 13

First, generate a data-set by clicking on the left and right buttons in the main window of the applet. Then, press "split" to split the data into training and test sets

Applet adaboost started


File Edit View Go Bookmarks Tools Help
http://www1.cs.columbia.edu/~freund/adaboost/
Go
prediction sum
Training set
prediction rule
Test set
reset
step
10 steps
error
1
0.5
0
1
80
160
iteration
Training Error: 0.0
Test Error: 0.04516129
Hypothesis Error: 0.38621867
Theoretical bound: 0.00725897
First, generate a data-set by clicking on the left and right buttons in the main window of the applet. Then, press "split" to split the data into training and test sets
Applet adaboost started

time = 100



time = 300

overfitting

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 if it occurred D(i) times
    - If I were to "resample" data, I would get more samples of "heavier" data points
- **Now, always do weighted calculations:**
  - e.g., MLE for Naïve Bayes, redefine *Count(Y=y)* to be weighted count:

$$Count(Y = y) = \sum_{j=1}^{n} D(j)\delta(Y^j = y)$$

  - setting D(j)=1 (or any constant value!), for all j, will recreates unweighted case

Given: $(x^1, y^1), \ldots, (x^m, y^m)$ where $x^i \in \mathbb{R}^n, y^i \in \{-1, +1\}$

Initialize: $D_1(i) = 1/m, \text{ for } i = 1, \ldots, m$

How? Many possibilities. Will see one shortly!

For t=1…T:

- Train base classifier $h_t(x)$ using D_t
- Choose α_t
- Update, for i=1..m:

Why? Reweight the data: examples i that are misclassified will have higher weights!

$$D_{t+1}(i) \propto D_t(i) \exp(-\alpha_t y^i h_t(x^i))$$

with normalization constant:

$$\sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y^i h_t(x^i))$$

- $y^i h_t(x^i) > 0$ → h_i correct
- $y^i h_t(x^i) < 0$ → h_i wrong
- h_i correct, $\alpha_t > 0$ → $D_{t+1}(i) < D_t(i)$
- h_i wrong, $\alpha_t > 0$ → $D_{t+1}(i) > D_t(i)$

Output final classifier:

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

Final Result: linear sum of "base" or "weak" classifier outputs.

Given: $(x^1, y^1), \ldots, (x^m, y^m)$ where

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

Initialize: $D_1(i) = 1/m, \;$ for $i = 1, \ldots,$

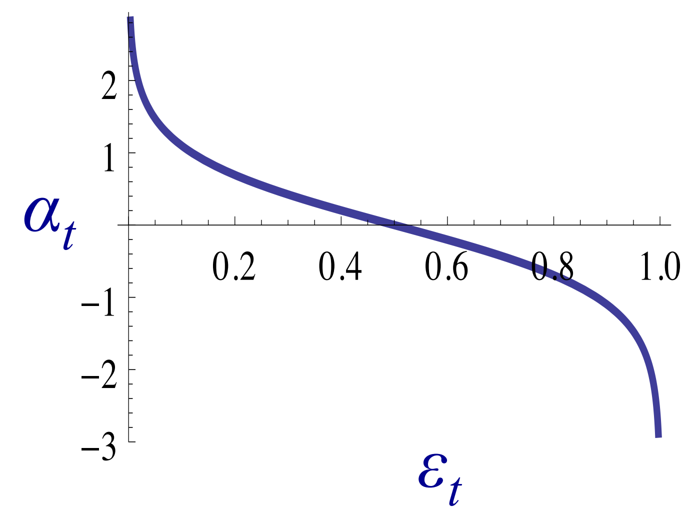
For t=1…T:

- Train base classifier $h_t(x)$ using D_t
- Choose α_t
- Update, for i=1..m:

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

$$D_{t+1}(i) \propto D_t(i) \exp(-\alpha_t y^i h_t(x^i))$$

- ε_t : error of h_t, weighted by D_t
  - $0 \leq \varepsilon_t \leq 1$
- α_t :
  - No errors: ε_t=0 → α_t=∞
  - All errors: ε_t=1 → α_t=−∞
  - Random: ε_t=0.5 → α_t=0



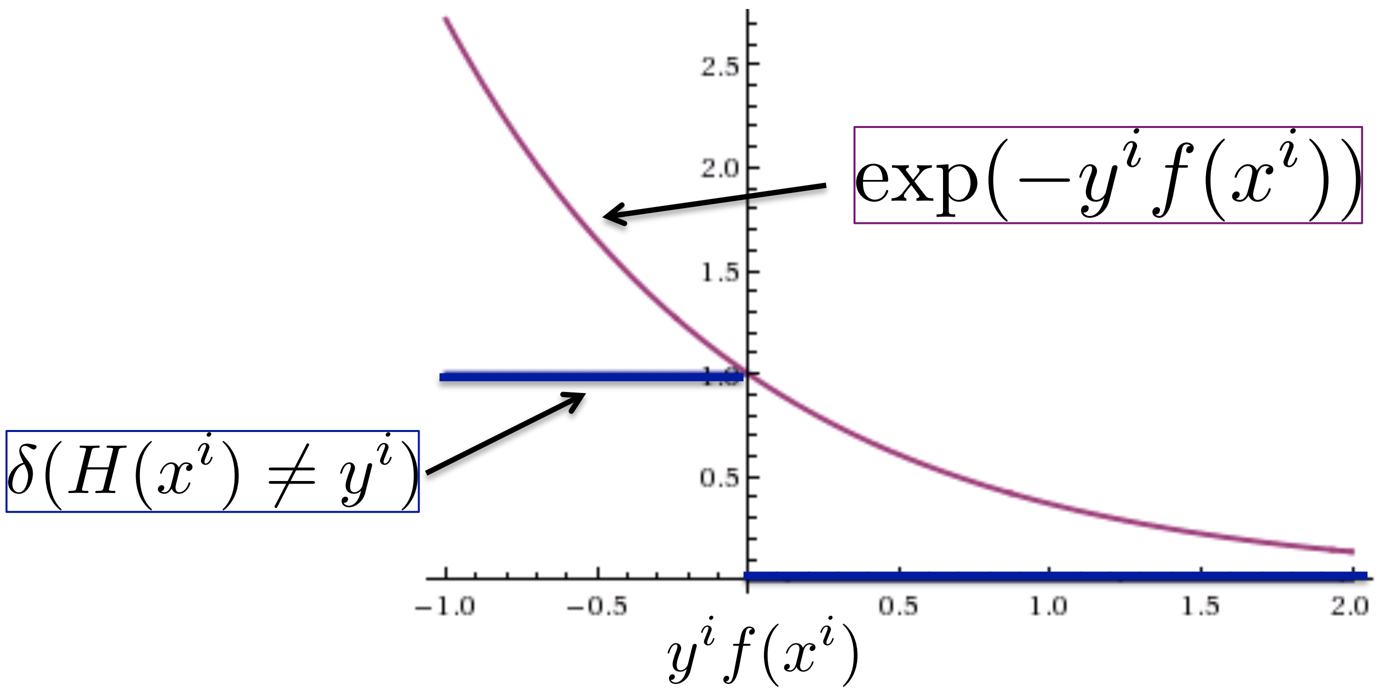
What α_t to choose for hypothesis h_t?

[Schapire, 1989]

Idea: choose α_t to minimize a bound on training error!

$$\sum_{i=1}^{m} \delta(H(x^i) \neq y^i) \leq \sum_{i=1}^{m} D_t(i) \exp(-y^i f(x^i))$$

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



What α_t to choose for hypothesis h_t?

[Schapire, 1989]

Idea: choose α_t to minimize a bound on training error!

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

Where

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

And

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

This equality isn't obvious! Can be shown with algebra (telescoping sums)!

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 $Z_{t.}$
- h_t is estimated as a black box, but can we solve for α_t?

Summary: choose α_t to minimize *error bound*

[Schapire, 1989]

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

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

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) \delta(h_t(x^i) \neq y^i)$$

For boolean Y: differentiate, set equal to 0, there is a closed form solution! [Freund & Schapire '97]:

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

Initialize: $D_1(i) = 1/m, \text{ for } i = 1, \ldots, m$

For t=1…T:

- Train base classifier $h_t(x)$ using D_t
- Choose α_t

$$\epsilon_t = \sum_{i=1}^{m} D_t(i)\delta(h_t(x^i) \neq y^i)$$

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

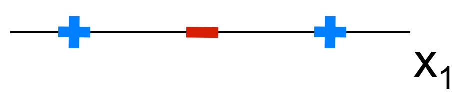
- Update, for i=1..m:

$$D_{t+1}(i) \propto D_t(i)\exp(-\alpha_t y^i h_t(x^i))$$

Output final classifier:

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

x_1	y
-1	1
0	-1
1	1



$H(x) = \text{sign}(0.35 \times h_1(x))$

- $h_1(x)=+1$ if $x_1>0.5$, -1 otherwise

Use decision stubs as base classifier

Initial:

- $D_1 = [D_1(1), D_1(2), D_1(3)] = [.33,.33,.33]$

t=1:

- Train stub [work omitted, breaking ties randomly]
  - $h_1(x)=+1$ if $x_1>0.5$, -1 otherwise
- $\varepsilon_1=\Sigma_i D_1(i)\ \delta(h_1(x^i)\neq y^i)$
  $= 0.33\times1+0.33\times0+0.33\times0=0.33$
- $\alpha_1=(1/2)\ \ln((1-\varepsilon_1)/\varepsilon_1)=0.5\times\ln(2)= 0.35$
- $D_2(1)\ \alpha\ D_1(1)\times\exp(-\alpha_1 y^1 h_1(x^1))$
  $= 0.33\times\exp(-0.35\times1\times-1) = 0.33\times\exp(0.35) = 0.46$
- $D_2(2)\ \alpha\ D_1(2) \times \exp(-\alpha_1 y^2 h_1(x^2))$
  $= 0.33\times\exp(-0.35\times-1\times-1) = 0.33\times\exp(-0.35) = 0.23$
- $D_2(3)\ \alpha\ D_1(3) \times \exp(-\alpha_1 y^3 h_1(x^3))$
  $= 0.33\times\exp(-0.35\times1\times1) = 0.33\times\exp(-0.35) =0.23$
- $D_2 = [D_1(1), D_1(2), D_1(3)] = [0.5,0.25,0.25]$

t=2

- Continues on next slide!

Initialize: $D_1(i) = 1/m, \text{ for } i = 1, \ldots, m$

For t=1…T:

- Train base classifier $h_t(x)$ using D_t
- Choose α_t

$$\epsilon_t = \sum_{i=1}^{m} D_t(i)\delta(h_t(x^i) \neq y^i)$$

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

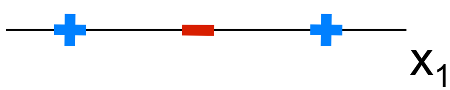
- Update, for i=1..m:

$$D_{t+1}(i) \propto D_t(i)\exp(-\alpha_t y^i h_t(x^i))$$

Output final classifier:

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

x_1	y
-1	1
0	-1
1	1



$H(x) = \text{sign}(0.35 \times h_1(x) + 0.55 \times h_2(x))$

- $h_1(x)=+1$ if $x_1>0.5$, -1 otherwise
- $h_2(x)=+1$ if $x_1<1.5$, -1 otherwise

- $D_2 = [D_1(1), D_1(2), D_1(3)] = [0.5, 0.25, 0.25]$

t=2:

- Train stub [work omitted; different stub because of new data weights D; breaking ties opportunistically (will discuss at end)]
  - $h_2(x)=+1$ if $x_1<1.5$, -1 otherwise
- $\varepsilon_2=\Sigma_i D_2(i)\ \delta(h_2(x^i)\neq y^i)$
  $= 0.5\times0+0.25\times1+0.25\times0=0.25$
- $\alpha_2=(1/2)\ \ln((1-\varepsilon_2)/\varepsilon_2)=0.5\times\ln(3)= 0.55$
- $D_2(1)\ \alpha\ D_1(1)\times\exp(-\alpha_2 y^1 h_2(x^1))$
  $= 0.5\times\exp(-0.55\times1\times1) = 0.5\times\exp(-0.55) = 0.29$
- $D_2(2)\ \alpha\ D_1(2)\times\exp(-\alpha_2 y^2 h_2(x^2))$
  $= 0.25\times\exp(-0.55\times-1\times1) = 0.25\times\exp(0.55) = 0.43$
- $D_2(3)\ \alpha\ D_1(3)\times\exp(-\alpha_2 y^3 h_2(x^3))$
  $= 0.25\times\exp(-0.55\times1\times1) = 0.25\times\exp(-0.55) = 0.14$
- $D_3 = [D_3(1), D_3(2), D_3(3)] = [0.33, 0.5, 0.17]$

t=3

- Continues on next slide!

Initialize: $D_1(i) = 1/m, \text{ for } i = 1, \ldots, m$

For t=1…T:

- Train base classifier $h_t(x)$ using D_t
- Choose α_t

$$\epsilon_t = \sum_{i=1}^{m} D_t(i)\delta(h_t(x^i) \neq y^i)$$

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

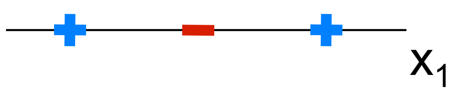
- Update, for i=1..m:

$$D_{t+1}(i) \propto D_t(i)\exp(-\alpha_t y^i h_t(x^i))$$

Output final classifier:

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

x_1	y
-1	1
0	-1
1	1



H(x) = sign($0.35 \times h_1(x) + 0.55 \times h_2(x) + 0.79 \times h_3(x)$)

- $h_1(x)$=+1 if x_1>0.5, -1 otherwise
- $h_2(x)$=+1 if x_1<1.5, -1 otherwise
- $h_3(x)$=+1 if x_1<-0.5, -1 otherwise

- $D_3 = [D_3(1), D_3(2), D_3(3)] = [0.33, 0.5, 0.17]$

t=3:

- Train stub [work omitted; different stub because of new data weights D; breaking ties opportunistically (will discuss at end)]
  - $h_3(x)$=+1 if x_1<-0.5, -1 otherwise
- $\varepsilon_3 = \Sigma_i D_3(i)\ \delta(h_3(x^i) \neq y^i)$
  $= 0.33 \times 0 + 0.5 \times 0 + 0.17 \times 1 = 0.17$
- $\alpha_3 = (1/2)\ \ln((1-\varepsilon_3)/\varepsilon_3) = 0.5 \times \ln(4.88) = 0.79$

- Stop!!! How did we know to stop?

Strong, weak classifiers

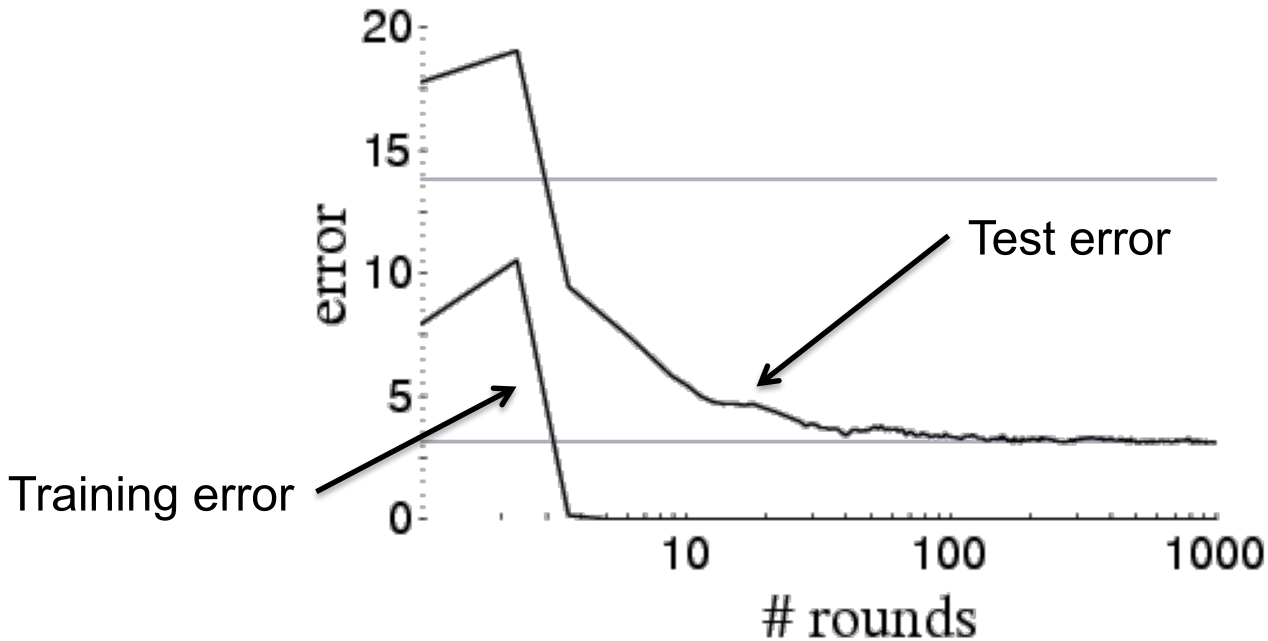
- If each classifier is (at least slightly) better than random: $\varepsilon_t < 0.5$
- Another bound on error:

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

- What does this imply about the training error?
  - Will reach zero!
  - Will get there exponentially fast!
- Is it hard to achieve better than random training error?

Boosting results – Digit recognition

[Schapire, 1989]



- Boosting:
  - Seems to be robust to overfitting
  - Test error can decrease even after training error is zero!!!

Boosting generalization error bound

[Freund & Schapire, 1996]

$$error_{true}(H) \quad \leq \quad error_{train}(H) + \tilde{\mathcal{O}}\left(\sqrt{\frac{Td}{m}}\right)$$

Constants:

- *T*: number of boosting rounds
  - Higher T → Looser bound, *what does this imply?*
- *d*: VC dimension of weak learner, measures complexity of classifier
  - Higher d → bigger hypothesis space → looser bound
- *m*: number of training examples
  - more data → tighter bound

Boosting generalization error bound

[Freund & Schapire, 1996]

$$error_{true}(H) \quad \leq \quad error_{train}(H) + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$$

Constants:

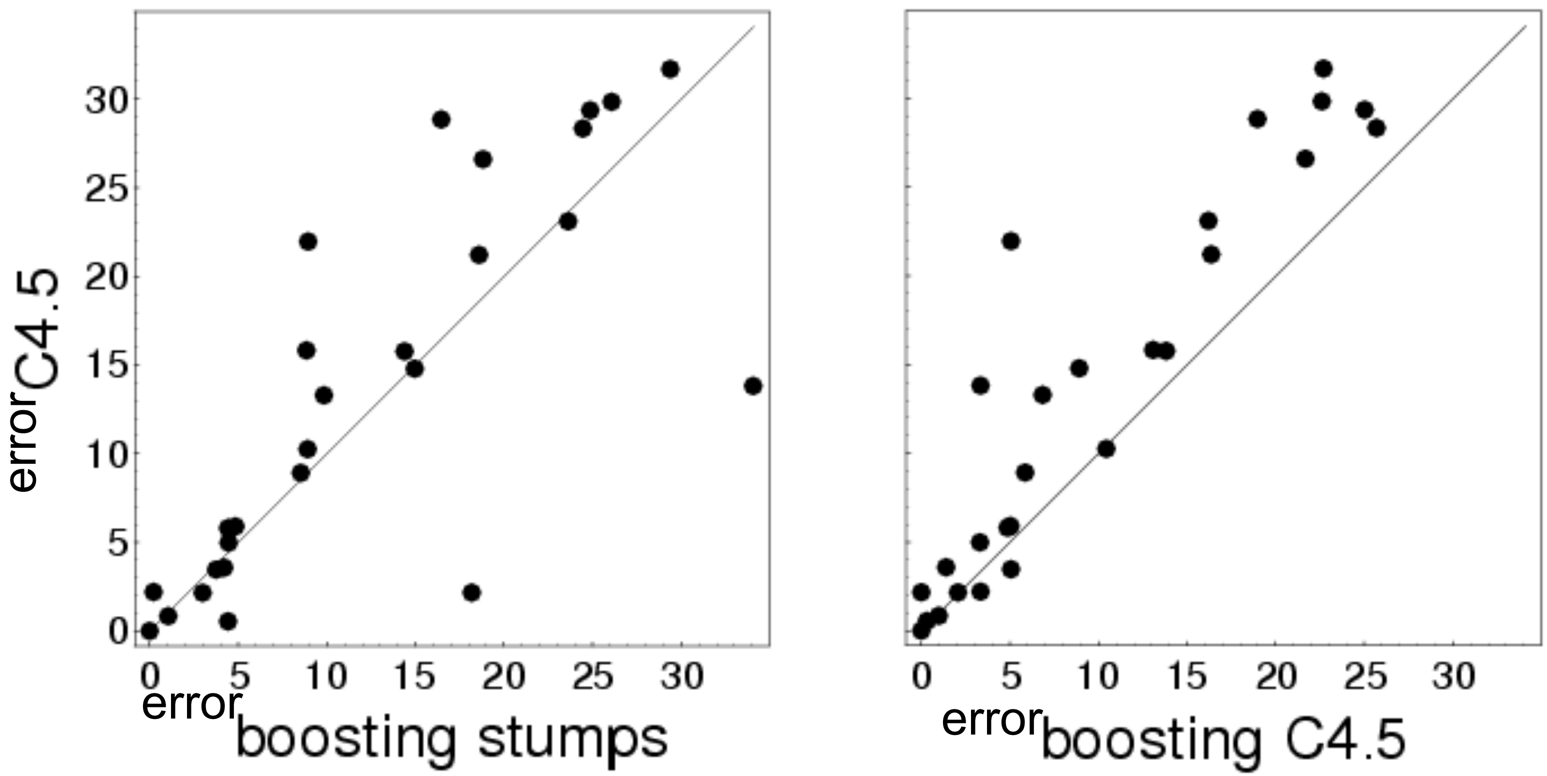
- **Theory does not match practice**:
  - Robust to overfitting
  - Test set error decreases even after training error is zero
- **Need better analysis tools**
  - we'll come back to this later in the quarter

– more data → tighter bound

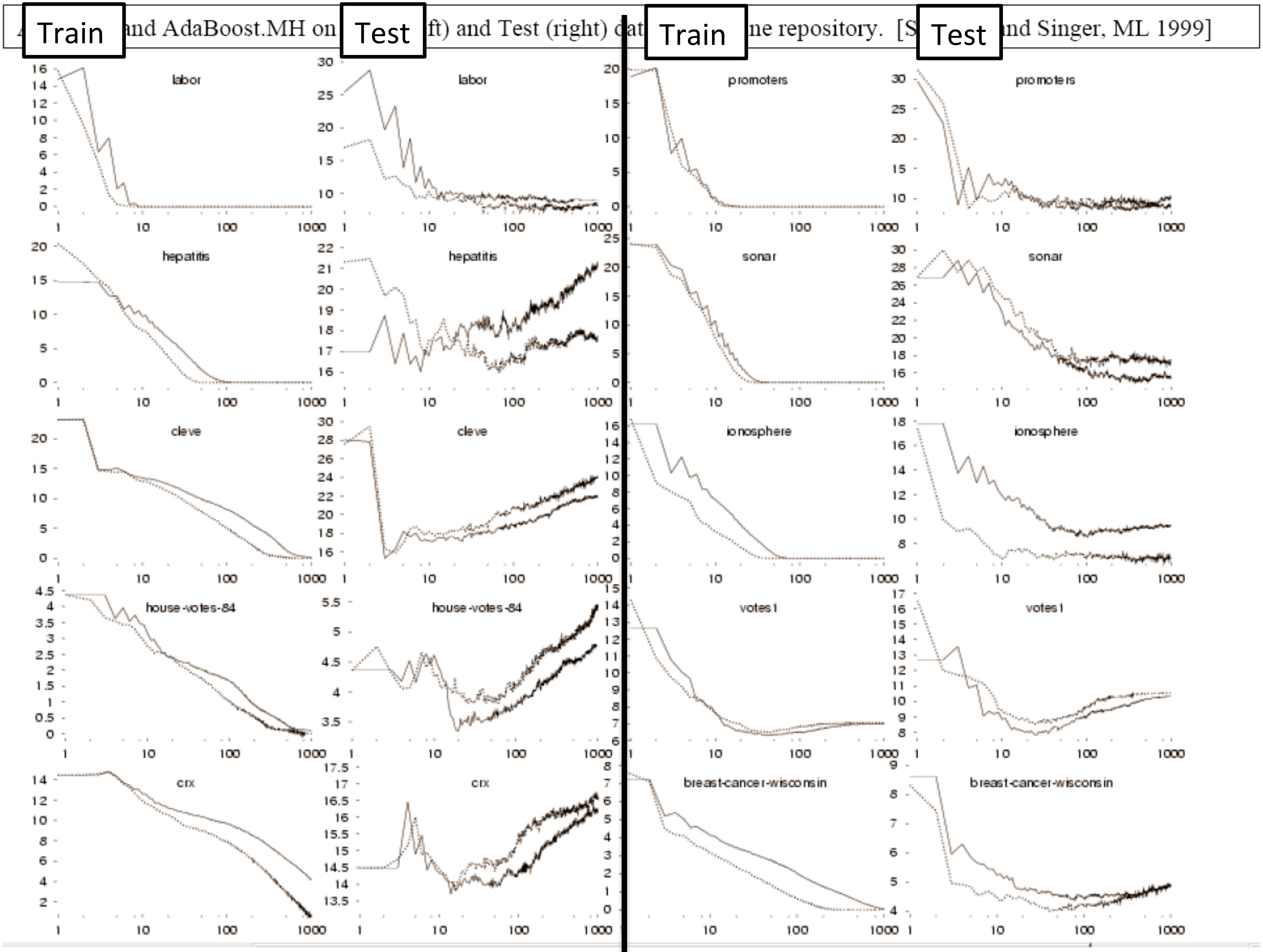
Boosting: Experimental Results

[Freund & Schapire, 1996]

Comparison of C4.5, Boosting C4.5, Boosting decision stumps (depth 1 trees), 27 benchmark datasets



A… and AdaBoost.MH on … ft) and Test (right) dat… ne repository. [S… nd Singer, ML 1999]



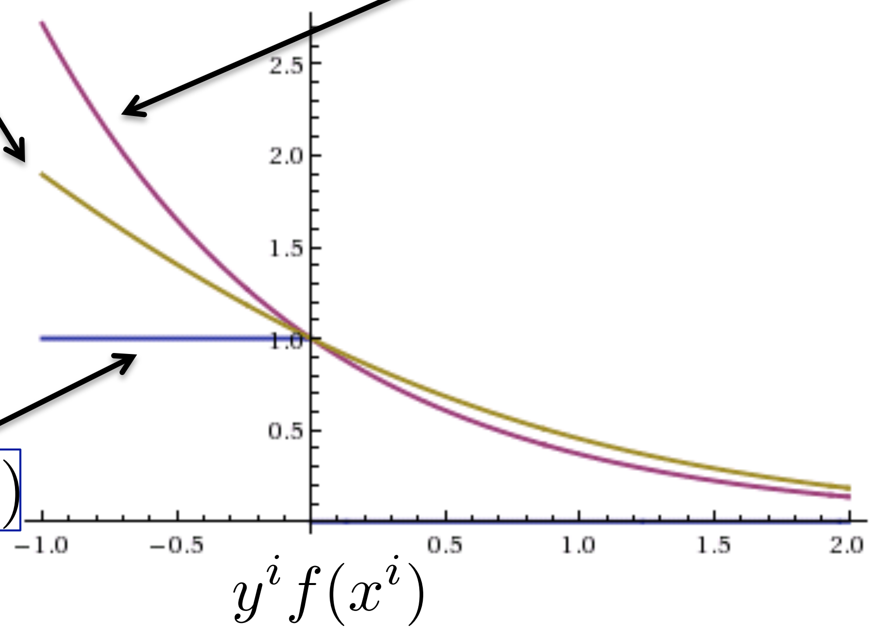
Boosting and Logistic Regression

Logistic regression equivalent to minimizing log loss:

Boosting minimizes similar loss function:

$$\boxed{\ln(1 + \exp(-y^i f(x^i)))}$$

$$\boxed{\exp(-y^i f(x^i))}$$



$$\boxed{\delta(H(x^i) \neq y^i)}$$

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 each feature x_j is predefined

- Jointly optimize parameters w_0, w_1, … w_n via gradient ascent.

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 $h_t(x)$ learned to fit data

- Weights α_j learned incrementally (new one for each training pass)

What you need to know about Boosting

- Combine weak classifiers to get very strong classifier
  - Weak classifier – slightly better than random on training data
  - Resulting very strong classifier – can get zero training error
- AdaBoost algorithm
- Boosting v. Logistic Regression
  - Both linear model, boosting "learns" features
  - 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