Cmput 466 / 551

Ensemble Methods

Readings: HTF: 8.7, 10, 16

R Greiner

University of Alberta

Some material from Tom Dietterich, E Roberto, M Botta, R Schapire

Motivation

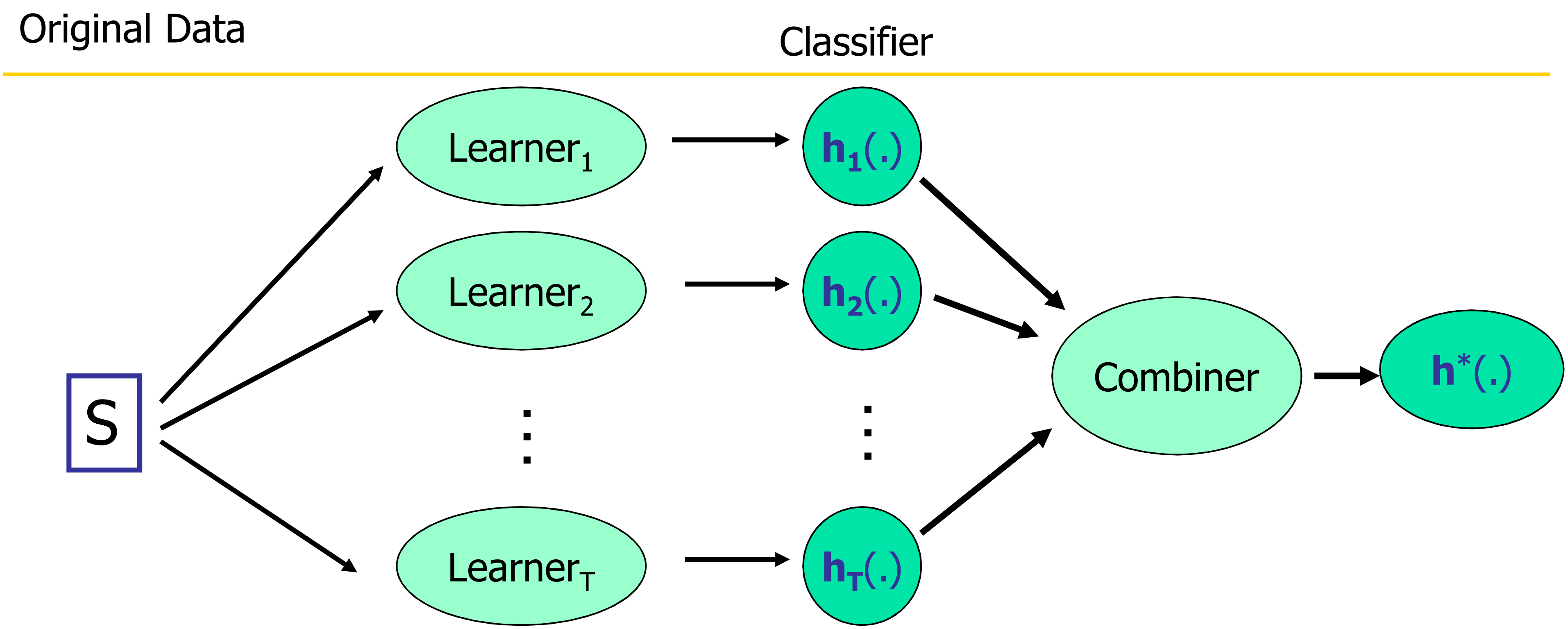
- If 1 learner is good
  - produces 1 effective classifier

  maybe many would be better

- Eg, why not learn { h_1, h_2, h_3 }, then
  - $h^*(x)$ = majority{ $h_1(x)$, $h_2(x)$, $h_3(x)$ }

- If h_i's make INDEPENDENT mistakes, h^* is more accurate!
  - Eg: If err(h_i) = ε, then err(h^*) = $3\varepsilon^2$ (0.01 $\rightarrow$ 0.0003)
  - If use majority of 2k–1 hyp, then $\boxed{err(h^*) \approx \binom{2k-1}{k} \epsilon^k}$

Learn then Combine Many Classifiers



Challenges

1. How to generate the base classifiers?
   - h_1, h_2, …
   - Different learners, Boostrap samples, …

2. How to integrate/combine them?
   - $h^*(x) = F(\ h_1(x), h_2(x), \ldots)$
   - F(…) = ?? Average, Weighted Average, Instance-specific decisions, …

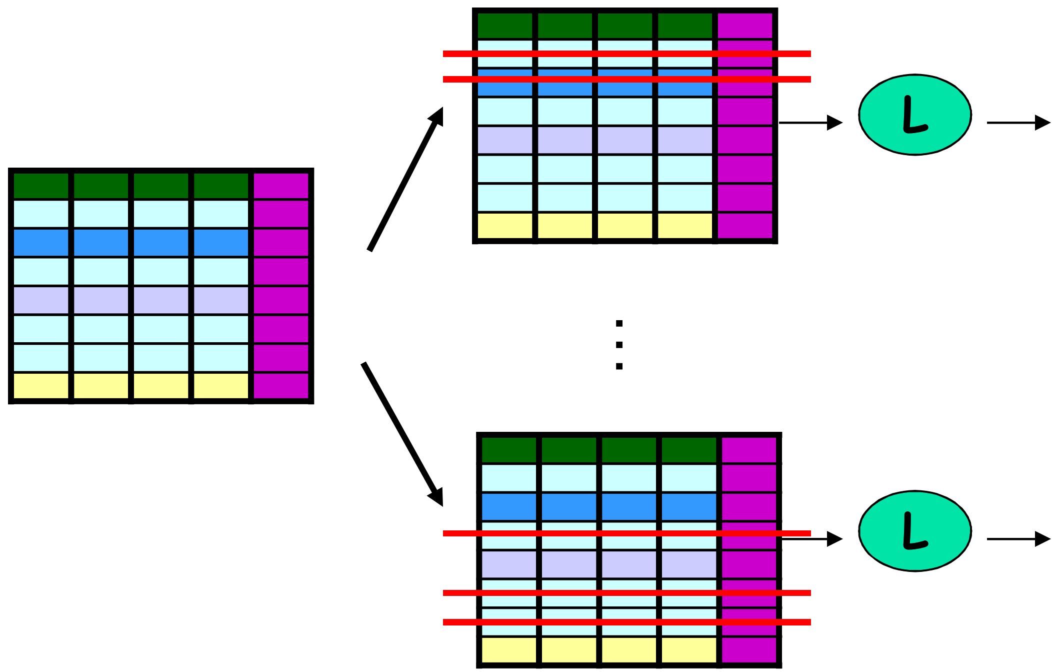
Types of Ensemble Methods

1. Subsample Training Sample
   - Bagging
   - Boosting
2. Manipulate Input Features
3. Manipulate Output Targets
   - ECOC
4. Injecting Randomness
   - Data
   - Algorithm
5. Algorithm Specific methods

- Other combinations
- Why do Ensembles work?

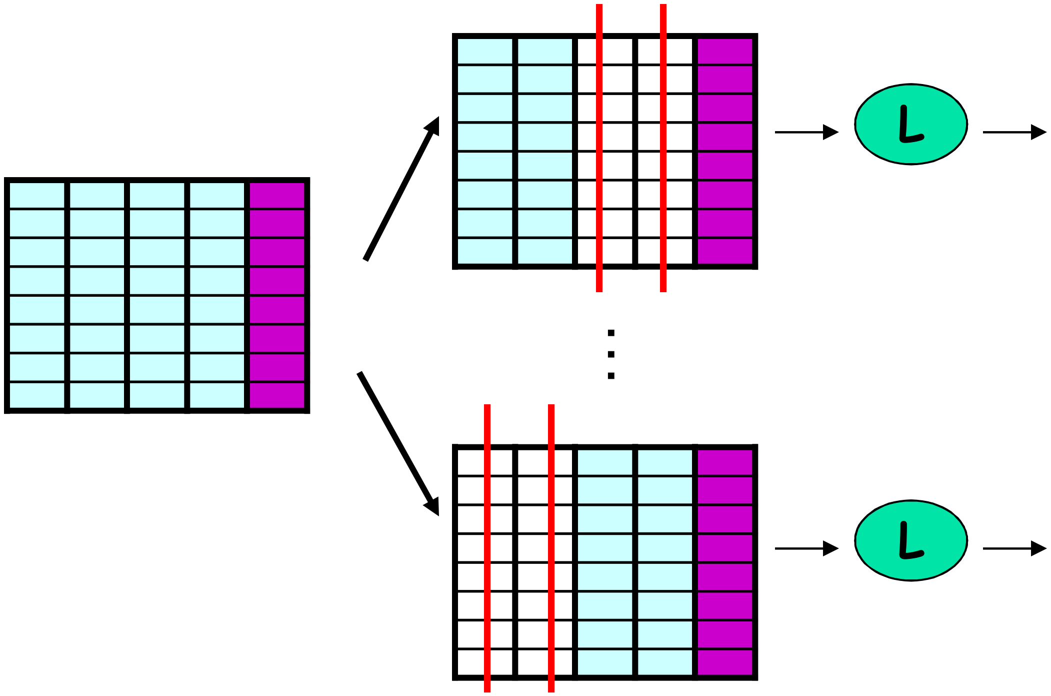
Types of Ensemble Methods

1. Subsample Training Sample
2.
3.
4.
5.



Types of Ensemble Methods

1.
2. Manipulate Input Features
3.
4.
5.



Types of Ensemble Methods

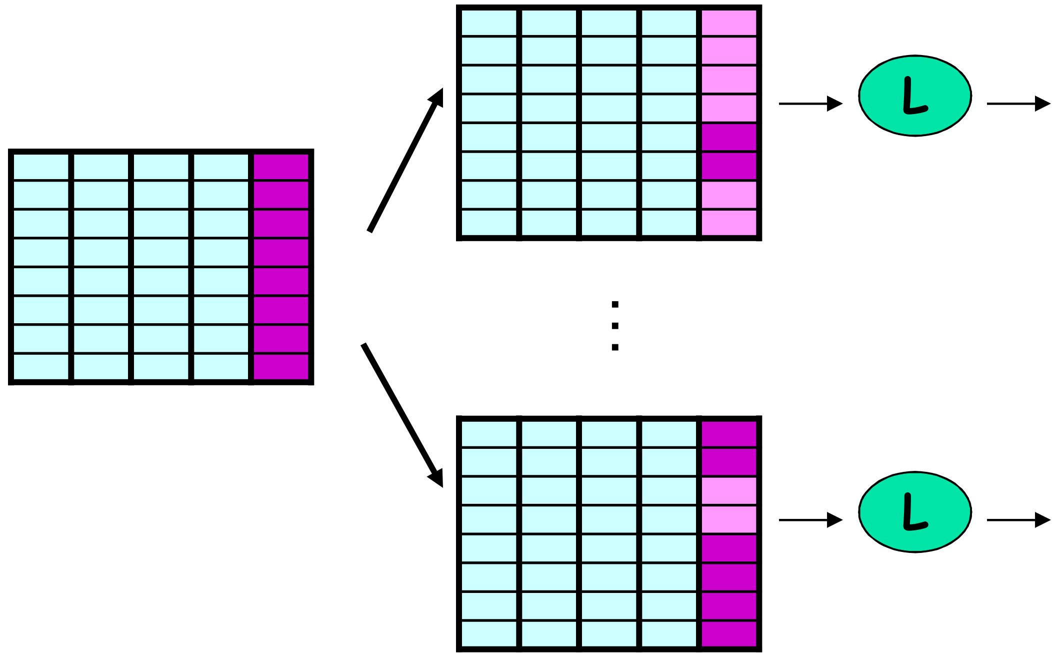
1.

2.

3. Manipulate Output Targets

4.

5.



Types of Ensemble Methods

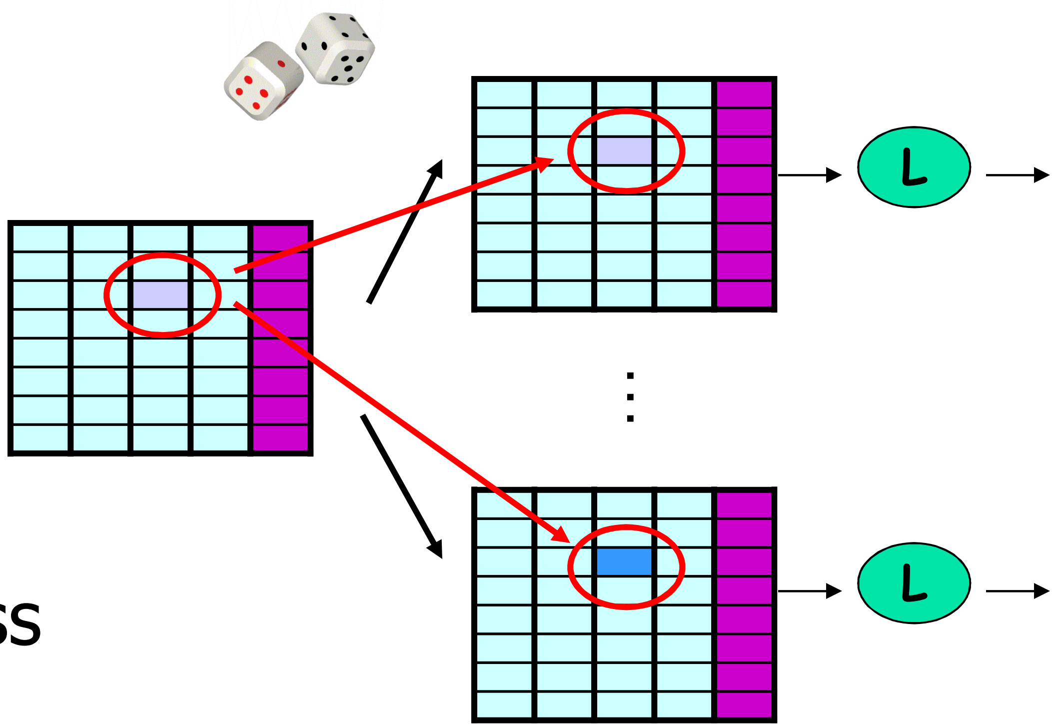
1.

2.

3.

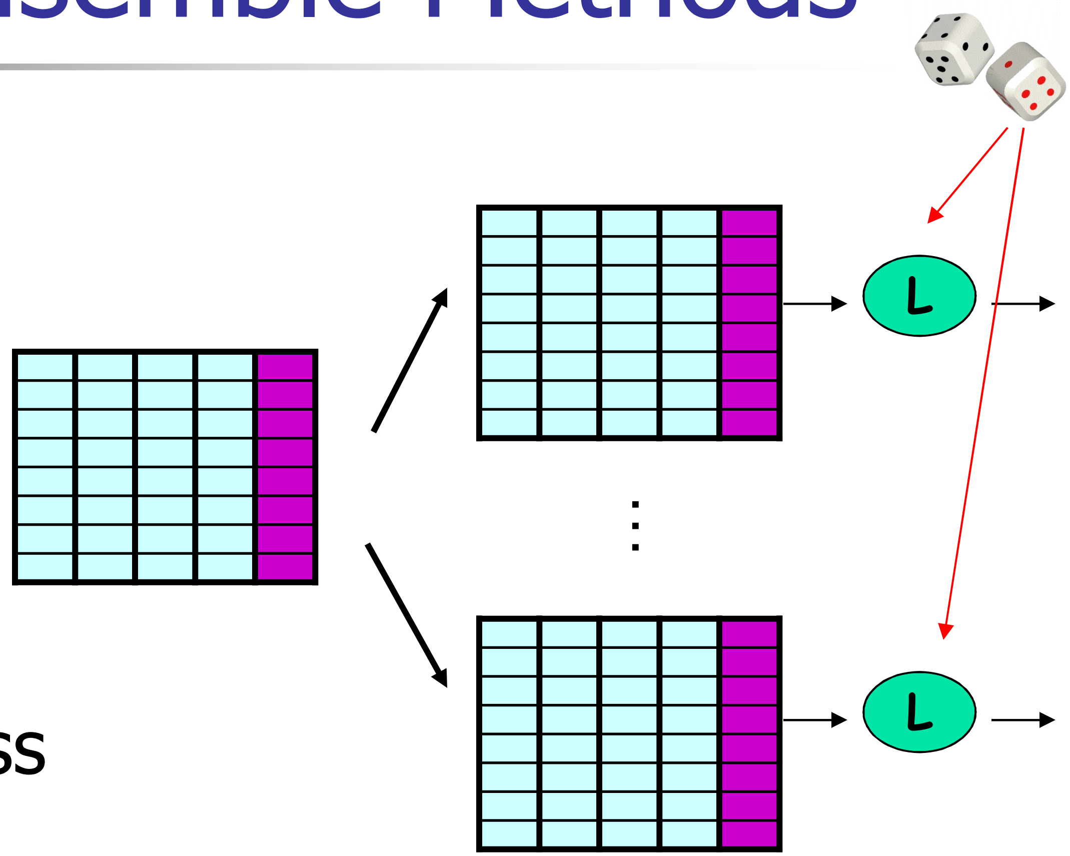
4. Injecting Randomness

5.



Types of Ensemble Methods

1.
2.
3.
4. Injecting Randomness
5.



1. Subsample Training Sample

Defn: Learner is **UNSTABLE** if
its output classifier undergoes **major** changes
in response to **small** changes in training data

- **Unstable**: Decision-tree, neural network, rule learning alg's
- **Stable**: Linear regression, nearest neighbor, linear threshold algorithms, ...

- Subsampling is best for unstable learners

- Techniques:
  - (Cross-Validated Committees)
  - Bagging
  - Boosting

1a: Bagging: **B**ootstrap **AG**gregating

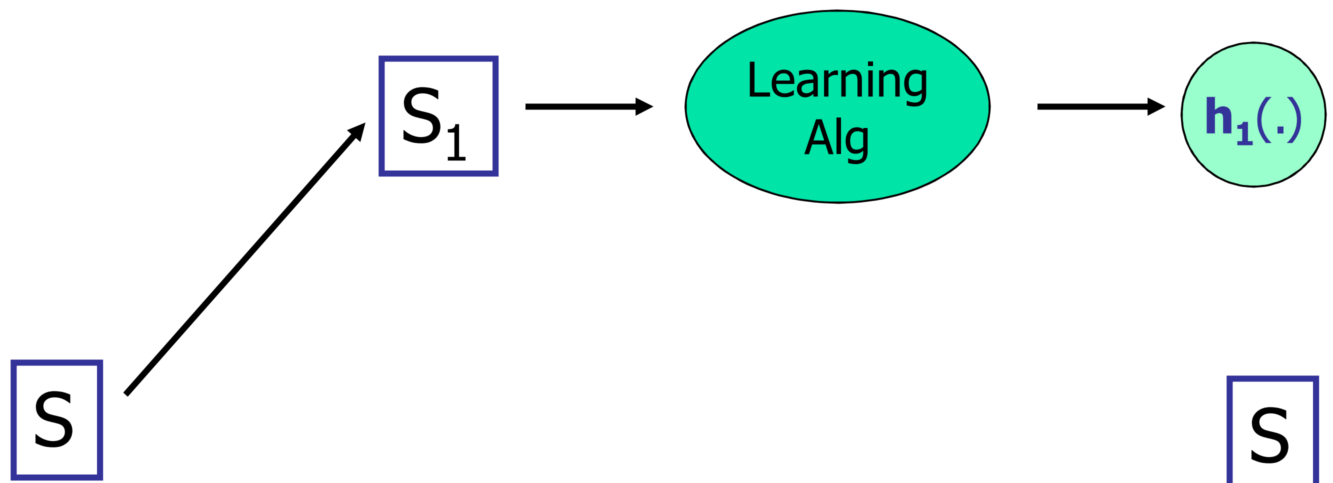
- For b = 1, …, T do
  - S_b = bootstrap replicate of S
  - Apply learning algorithm to S_b to learn h_b
- To classify new point **x**,
  using unweighted vote:

$$\boxed{\hat{h}(x) = sign\left(\frac{1}{T}\sum_{i=1}^{T} h_i(x)\right)}$$

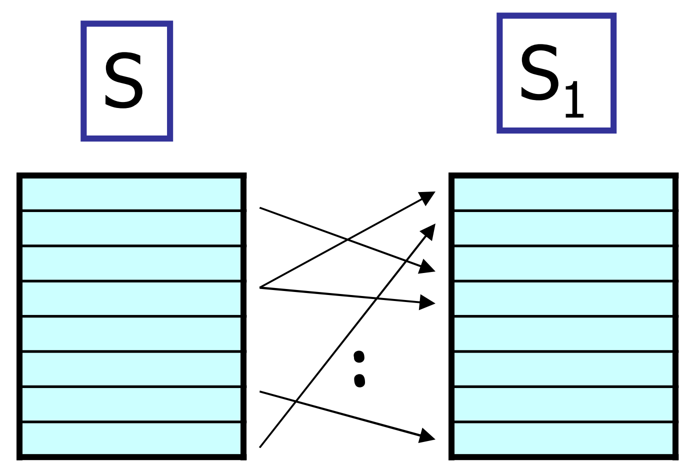
$$\boxed{\hat{h}(x) = \text{argmax}_r \{ |h_i(x) = r| \}}$$

Boostrap Replicates

Original Data | Bootstrap Replicate | Classifier

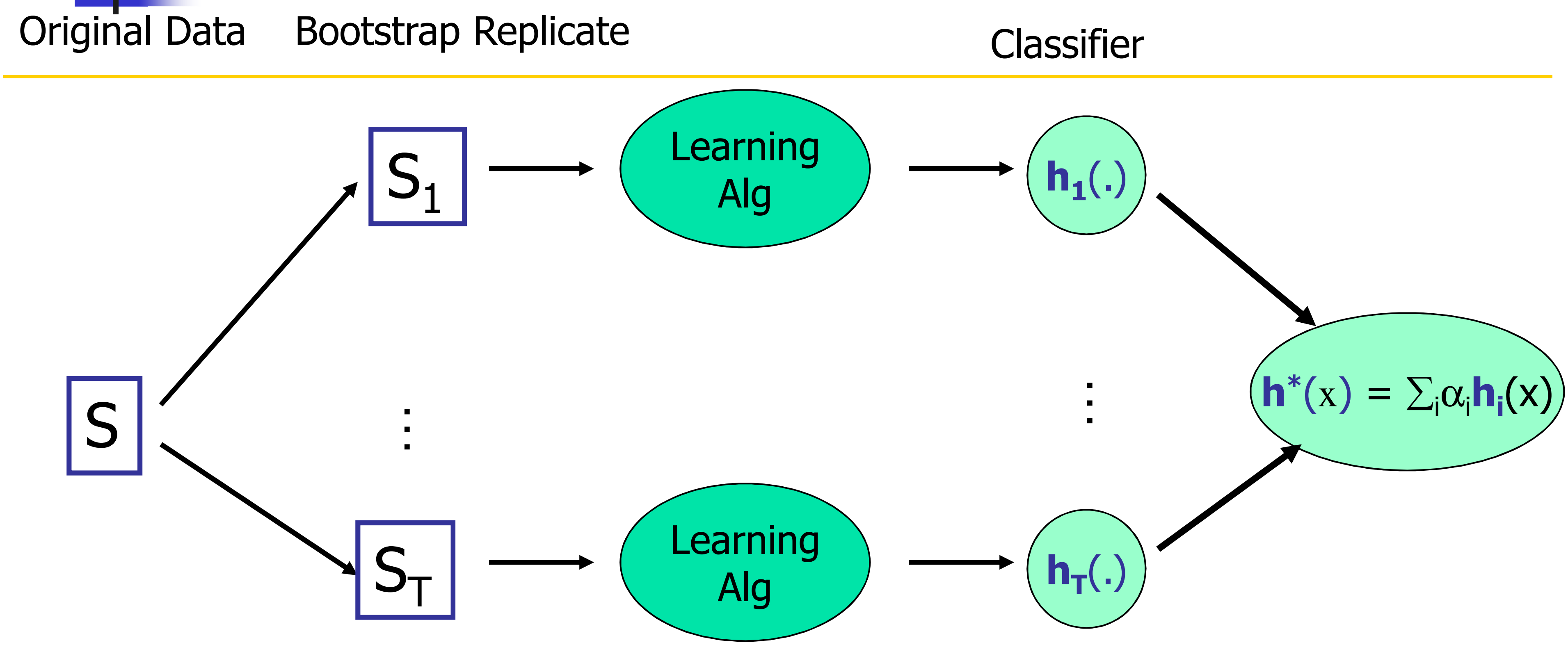


Form S_1 by drawing $|S|$ instances from S, with replacement



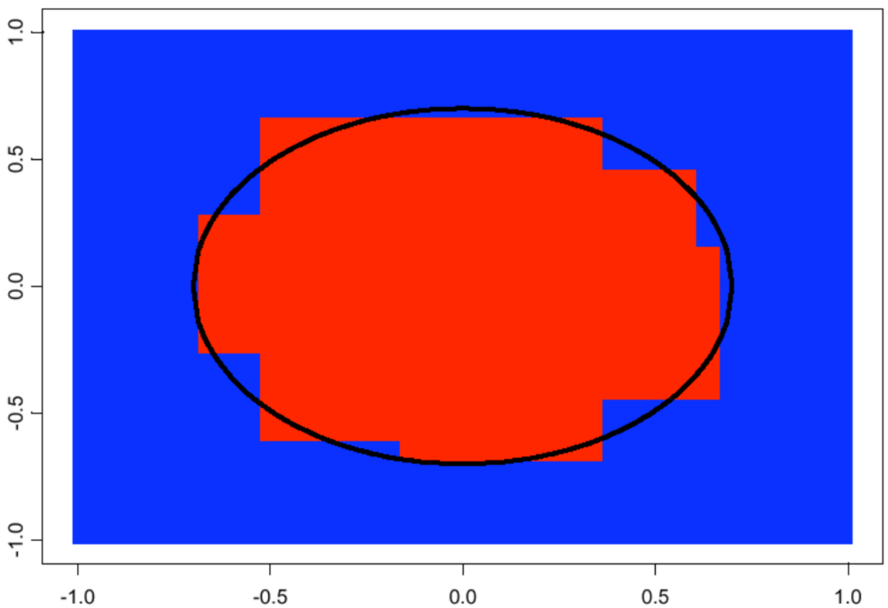
What is expected size of overlap: $|S \cap S_1|$?

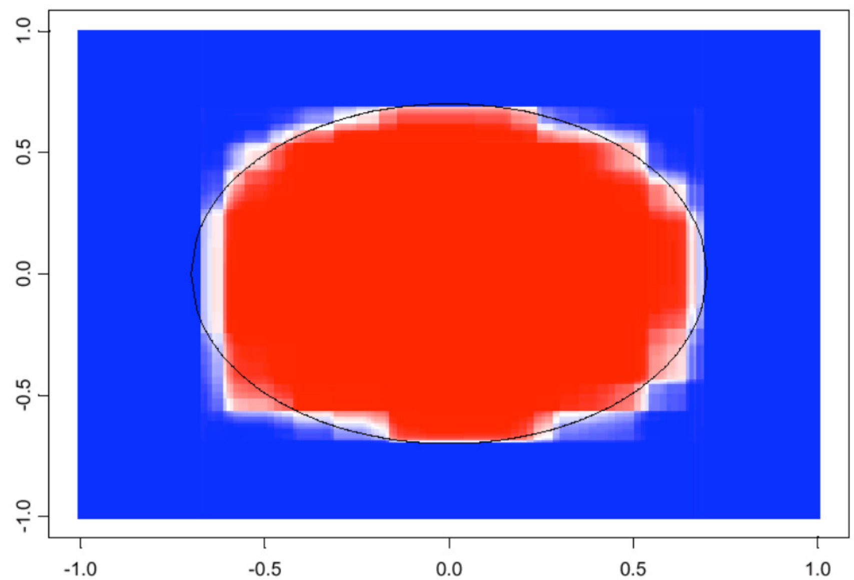
Boostrap Replicates



- Each S_i is bootstrap replicate
- h_i = classifier based on S_i
- $\alpha_i = 1/T$

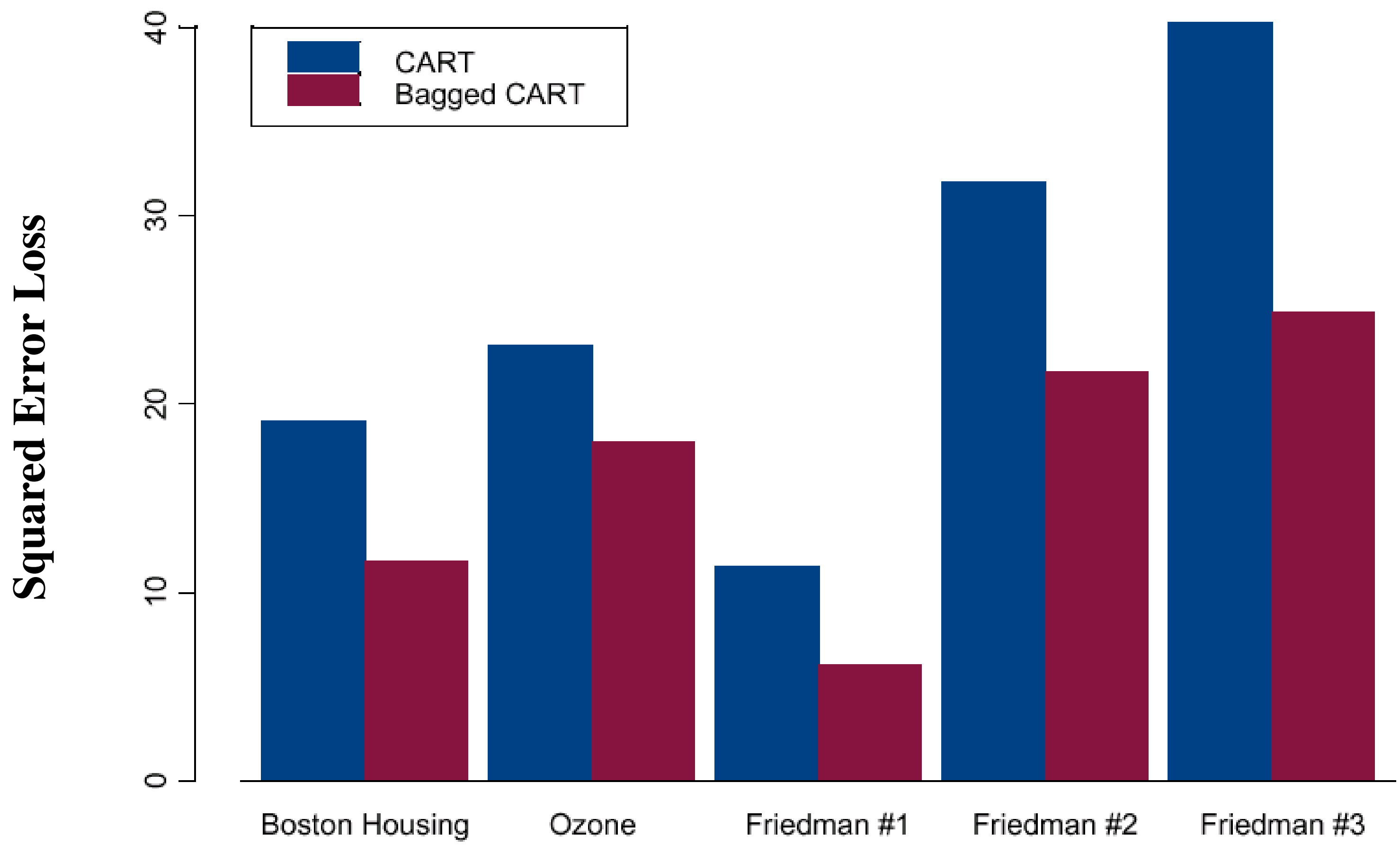
CART vs Bagged-CART



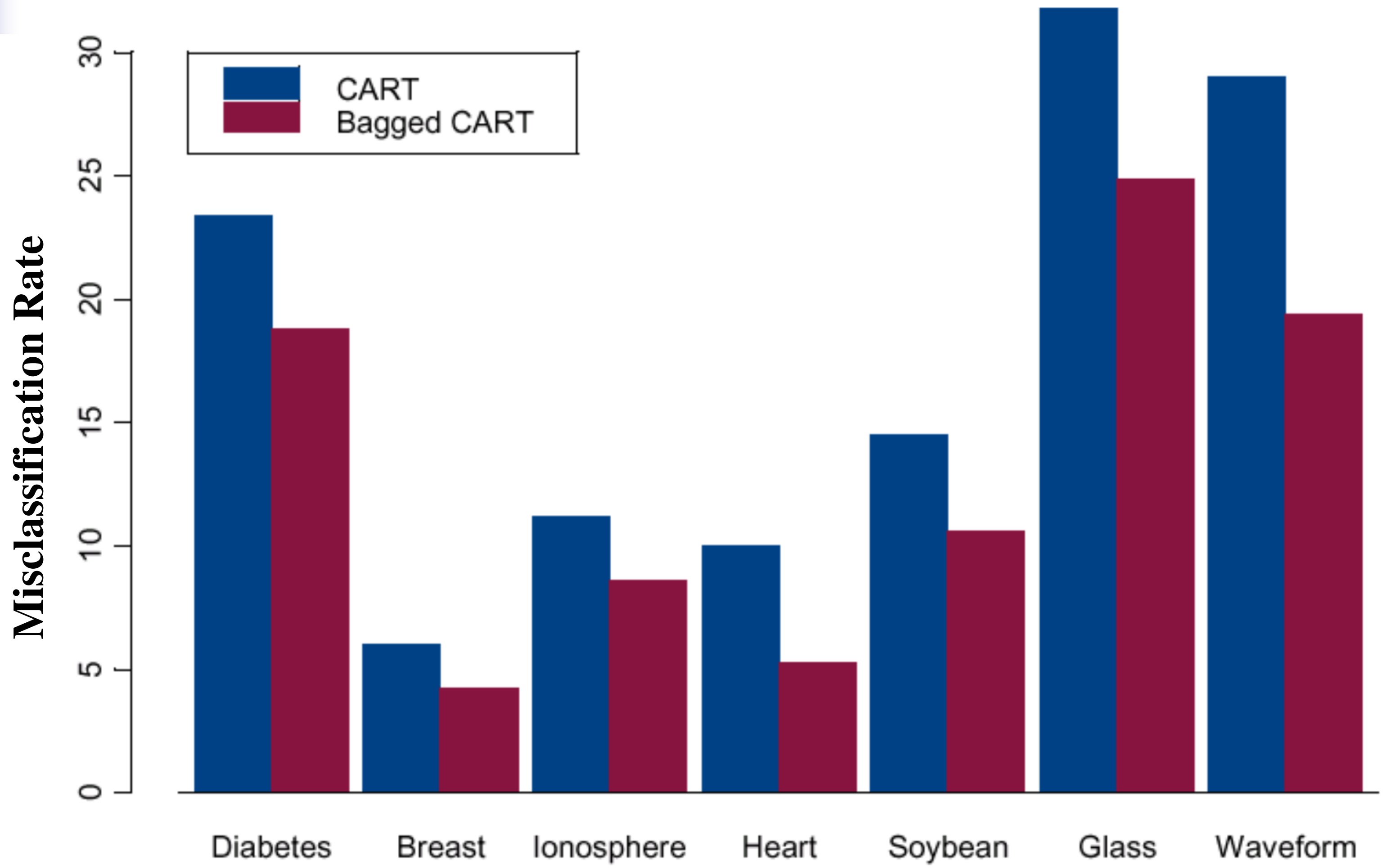


- 100 bagged trees
- shades of blue/red indicate strength of vote for particular classification

Regression Results



Classification Results



Expected Error

$$\hat{h}(x) = \frac{1}{T}\sum_{i=1}^{T} h_i(x)$$

- Assume $h_i(\mathbf{x}) = y(\mathbf{x}) + \varepsilon_i(\mathbf{x})$
- $E_x[\ (h_i(\mathbf{x}) - y(\mathbf{x})\ )^2\ ] = E_x[\ \varepsilon_i(\mathbf{x})^2\ ]$
- What is average error of $\hat{h}(\mathbf{x})$?
  - Assume $\varepsilon_i(\mathbf{x})$ each 0-mean and *uncorrelated*

$$E_x[\ \varepsilon_i(\mathbf{x})\ ] = 0$$

$$\boxed{E_x[\ \varepsilon_i(\mathbf{x})\ \varepsilon_j(\mathbf{x})\ ] = 0 \quad \text{for } i \neq j}$$

$$E_{\hat{h}} = E_x[\ y(\mathbf{x}) - \tfrac{1}{T}\textstyle\sum_i h_i(\mathbf{x})\ ]^2$$

$$= E_x[\ \tfrac{1}{T}\textstyle\sum_i \varepsilon_i(\mathbf{x})\ ]^2 \quad = \tfrac{1}{T}\ E_x[\ \varepsilon_i(\mathbf{x})^2\ ] \ \ !!$$

Got to here – Oct/2015

Types of Ensemble Methods

1. Subsample Training Sample
   - Bagging
   - Boosting
2. Manipulate Input Features
3. Manipulate Output Targets
   - ECOC
4. Injecting Randomness
5. Algorithm Specific methods

- Other combinations
- Why do Ensembles work?

1b: Boosting

- Boosting = general method of using…
  - "weak" learning algorithm L(…)
    - can reliably produce classifiers
      (at least) slightly better than random,
    - say, accuracy ≥ 55% (in two-class setting)

  to produce highly accurate predictor
  - single classifier with very high accuracy,
    - say, 99%

  … given sufficient data…

Strong vs Weak Learnability

- Boosting's roots are in "PAC" learning model (Valiant)
- Given random examples from unknown, arbitrary distribution…
- *strong* PAC learning algorithm:
  - for any distribution, with high probability, given poly # of examples, polynomial time,
  - can always find classifier with *arbitrarily small generalization error*
- *weak* PAC learning algorithm
  - same… but generalization error only needs to be slightly better than random guessing ($\frac{1}{2} - \gamma$)
- [Kearns & Valiant '88]:
  - does weak learnability imply strong learnability?

Early Boosting Algorithms

- [Kearns & Valiant ’88]:
  - does weak learnability imply strong learnability?
- YES! [Schapire ’89]:
  - provable boosting algorithm
- [Freund ’90]:
  - “optimal” algorithm that “boosts by majority”
- [Drucker, Schapire & Simard ’92]:
  - first experiments using boosting
  - limited by practical drawbacks

AdaBoost

- [Freund & Schapire '95]:
  - introduced "AdaBoost" algorithm
  - strong practical advantages over previous boosting algorithms
- experiments and applications using AdaBoost:

[Drucker & Cortes '96]	[Abney, Schapire & Singer '99]	[Tieu & Viola '00]
[Jackson & Craven '96]	[Haruno, Shirai & Ooyama '99]	[Walker, Rambow & Rogati '01]
[Freund & Schapire '96]	[Cohen & Singer' 99]	[Rochery, Schapire, Rahim & Gupta '01]
[Quinlan '96]	[Dietterich '00]	[Merler, Furlanello, Larcher & Sboner '01]
[Breiman '96]	[Schapire & Singer '00]	[Di Fabbrizio, Dutton, Gupta et al. '02]
[Maclin & Opitz '97]	[Collins '00]	[Qu, Adam, Yasui et al. '02]
[Bauer & Kohavi '97]	[Escudero, Màrquez & Rigau '00]	[Tur, Schapire & Hakkani-Tür '03]
[Schwenk & Bengio '98]	[Iyer, Lewis, Schapire et al. '00]	[Viola & Jones '04]
[Schapire, Singer & Singhal '98]	[Onoda, Rätsch & Müller '00]	[Middendorf, Kundaje, Wiggins et al. '04]
		⋮

- continuing development of theory and algorithms:

[Breiman '98, '99]	[Duffy & Helmbold '99, '02]	[Koltchinskii, Panchenko & Lozano '01]
[Schapire, Freund, Bartlett & Lee '98]	[Freund & Mason '99]	[Collins, Schapire & Singer '02]
[Grove & Schuurmans '98]	[Ridgeway, Madigan & Richardson '99]	[Demiriz, Bennett & Shawe-Taylor '02]
[Mason, Bartlett & Baxter '98]	[Kivinen & Warmuth '99]	[Lebanon & Lafferty '02]
[Schapire & Singer '99]	[Friedman, Hastie & Tibshirani '00]	[Wyner '02]
[Cohen & Singer '99]	[Rätsch, Onoda & Müller '00]	[Rudin, Daubechies & Schapire '03]
[Freund & Mason '99]	[Rätsch, Warmuth, Mika et al. '00]	[Jiang '04]
[Domingo & Watanabe '99]	[Allwein, Schapire & Singer '00]	[Lugosi & Vayatis '04]
[Mason, Baxter, Bartlett & Frean '99]	[Friedman '01]	[Zhang '04]
		⋮

A desicion-theoretic generalization of on-line learning and an application to **boosting**

Y Freund, RE **Schapire** - Computational learning theory, 1995 - Springer

Abstract We consider the problem of dynamically apportioning resources among a set of options in a worst-case on-line framework. The model we study can be interpreted as a broad, abstract extension of the well-studied on-line prediction model to a general **...**

Cited by 10097 Related articles All 64 versions Cite Save

[PDF] Experiments with a new **boosting** algorithm

Y Freund, RE **Schapire** - ICML, 1996 - public.asu.edu

Abstract In an earlier paper [9], we introduced a new "**boosting**" algorithm called AdaBoost which, theoretically, can be used to significantly reduce the error of any learning algorithm that consistently generates classifiers whose performance is a little better than random **...**

Cited by 6079 Related articles All 85 versions Cite Save More

Improved **boosting** algorithms using confidence-rated predictions

RE **Schapire**, Y Singer - Machine learning, 1999 - Springer

Abstract We describe several improvements to Freund and **Schapire's** AdaBoost **boosting** algorithm, particularly in a setting in which hypotheses may assign confidences to each of their predictions. We give a simplified analysis of AdaBoost in this setting, and we show **...**

Cited by 2916 Related articles All 33 versions Cite Save

[PDF] A short introduction to **boosting**

Y Freund, R **Schapire**, N Abe - Journal-Japanese Society For Artificial ..., 1999 - yorku.ca

Abstract **Boosting** is a general method for improving the accuracy of any given learning algorithm. This short overview paper introduces the **boosting** algorithm AdaBoost, and explains the underlying theory of **boosting**, including an explanation of why **boosting** often **...**

Cited by 1752 Related articles All 101 versions Cite Save More

Boosting the margin: A new explanation for the effectiveness of voting methods

RE **Schapire**, Y Freund, P Bartlett, WS Lee - Annals of statistics, 1998 - JSTOR

One of the surprising recurring phenomena observed in experiments with **boosting** is that the test error of the generated classifier usually does not increase as its size becomes very large, and often is observed to decrease even after the training error reaches zero. In this **...**

Cited by 2268 Related articles All 26 versions Cite Save

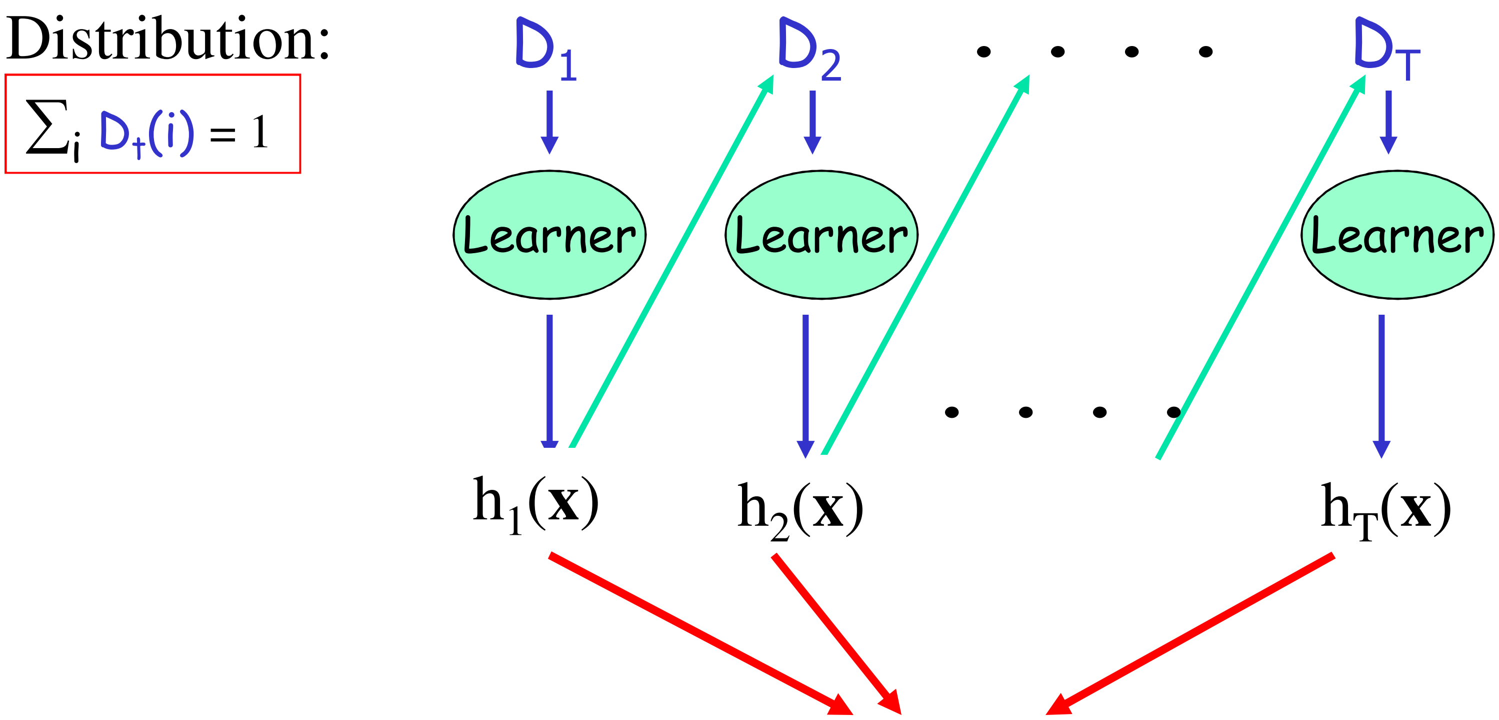
BoosTexter: A **boosting**-based system for text categorization

RE **Schapire**, Y Singer - Machine learning, 2000 - Springer

Abstract This work focuses on algorithms which learn from examples to perform multiclass text and speech categorization tasks. Our approach is based on a new and improved family of **boosting** algorithms. We describe in detail an implementation, called BoosTexter, of the **...**

Cited by 1753 Related articles All 28 versions Cite Save

Boosting Overview



$$h^*(\mathbf{x}) = \text{sign}\left(\sum_t \alpha_t \, h_t(\mathbf{x}) \right)$$

A Formal Description of Boosting

- Training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$
  - $y_i \in \{-1, +1\}$ correct label of instance $\mathbf{x}_i \in X$
- for $t = 1, \ldots, T$:
  - construct distribution D_t on $\{1, \ldots, m\}$
  - find weak classifier $h_t : X \to \{-1, +1\}$

    with small error ε_t on D_t :

    $$\varepsilon_t = \Pr_{i \in D_t}[\, h_t(\mathbf{x}_i) \neq y_i \,] = \sum_{i: y_i \neq h_t(x_i)} D_t(i)$$

- output final classifier h^* based on $\{ h_t(\mathbf{x}) \}$

[Schapire, Freund]

AdaBoost

- constructing D_t:
  - $D_1(i) = 1/m$
  - given D_t and h_t :

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

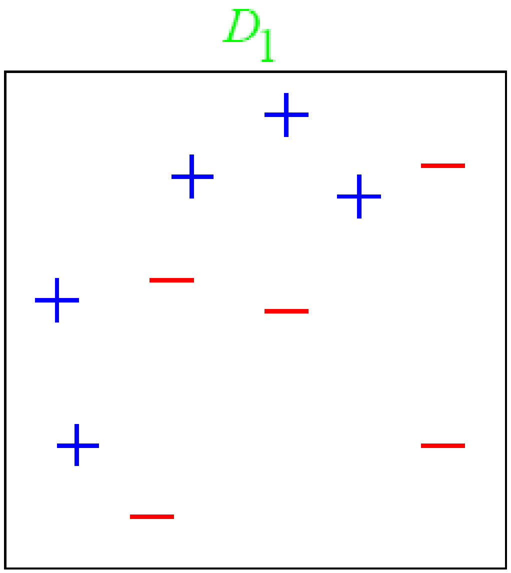
where $\boxed{\varepsilon_t = \sum_{i:y_i \neq h_t(x_i)} D_t(i)}$ $\boxed{\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)}$

Z_t = normalization constant

$\varepsilon < ½$, so $\alpha > 0$, so $e^{-\alpha} < 1$, so …
if correct, $D_{t+1}(i) < D_t(i)$ … if wrong, $D_{t+1}(i) > D_t(i)$

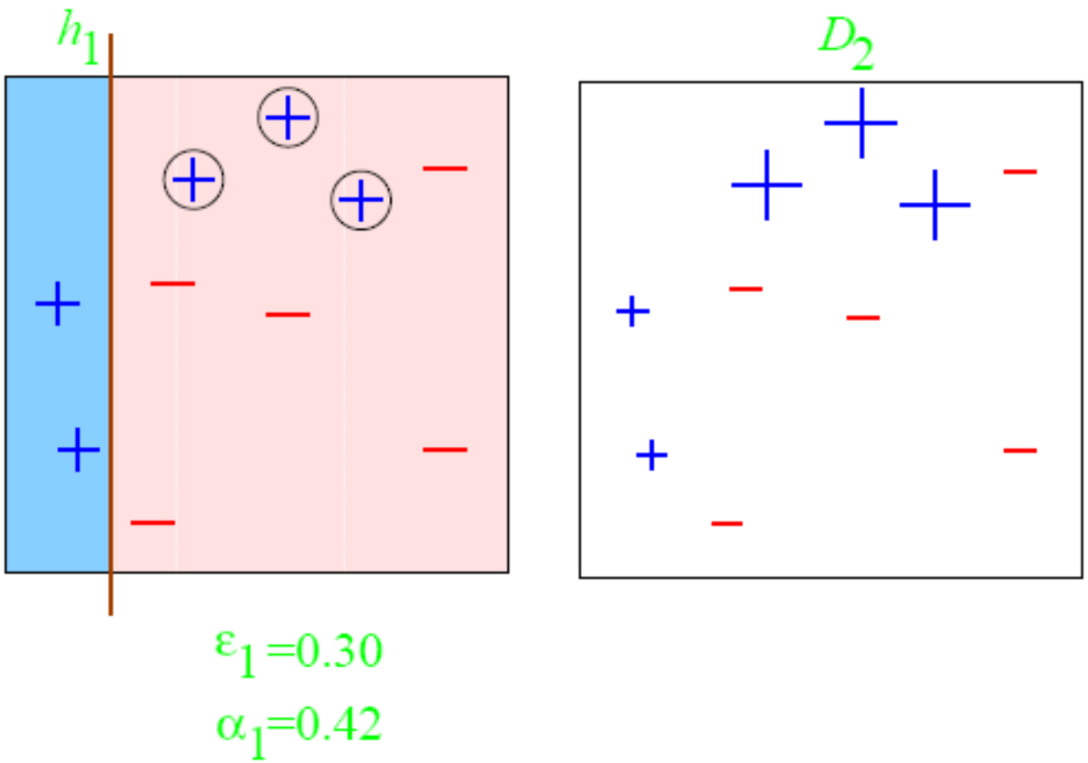
- final classifier: $h^*(\mathbf{x}) = \text{sign}\left(\sum_t \alpha_t\, h_t(\mathbf{x}) \right)$

Toy Example

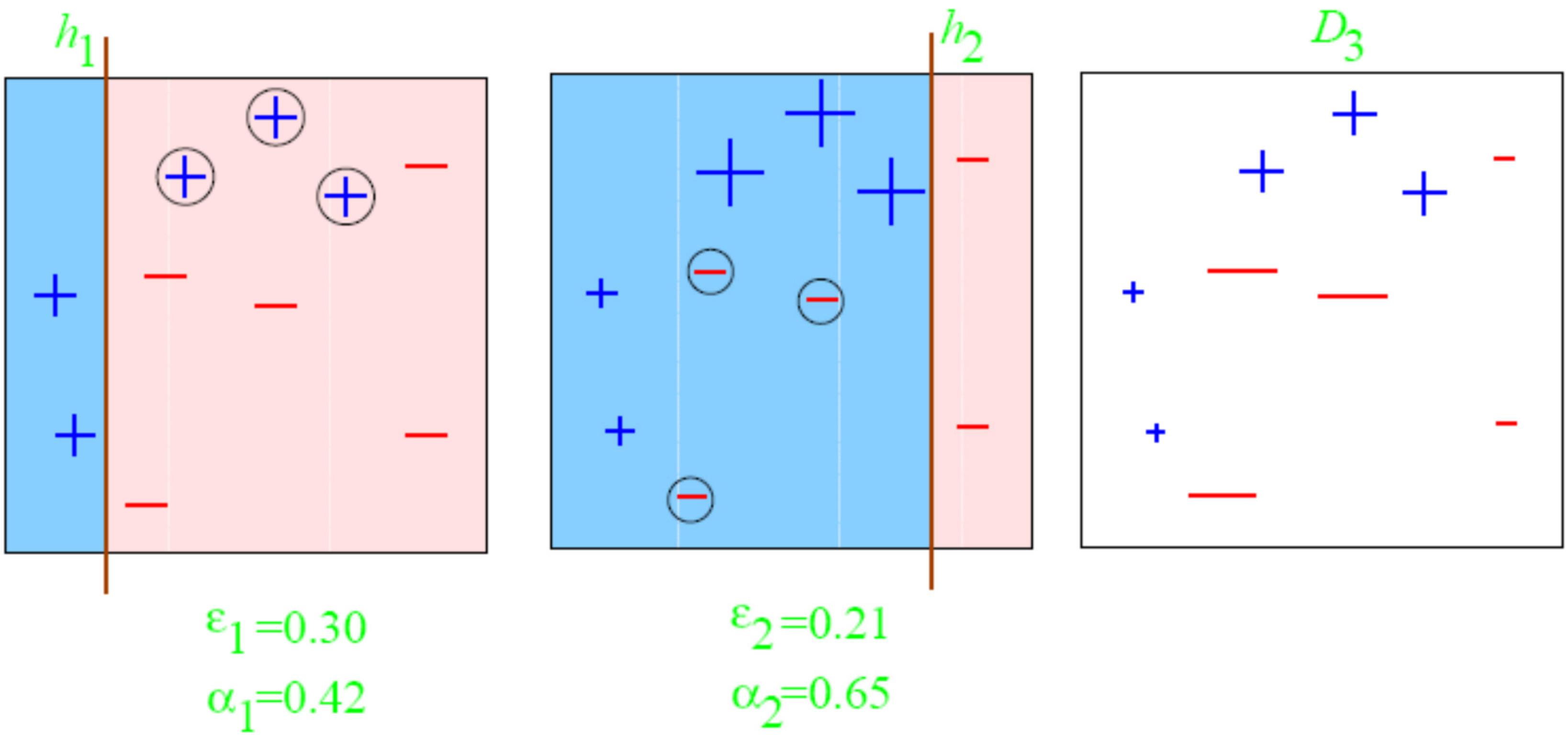


here: each weak classifiers = a vertical or horizontal half-planes

Round 1

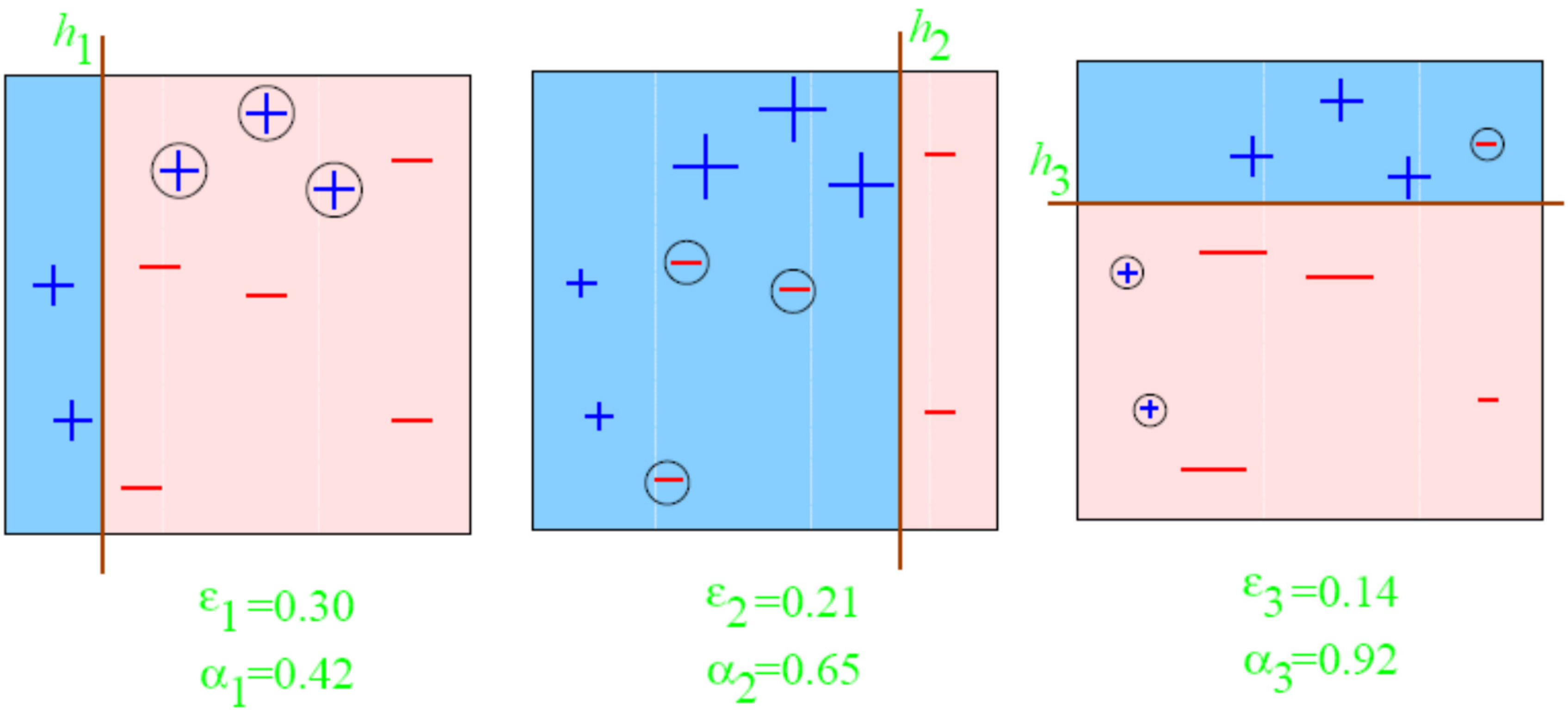


Round 2



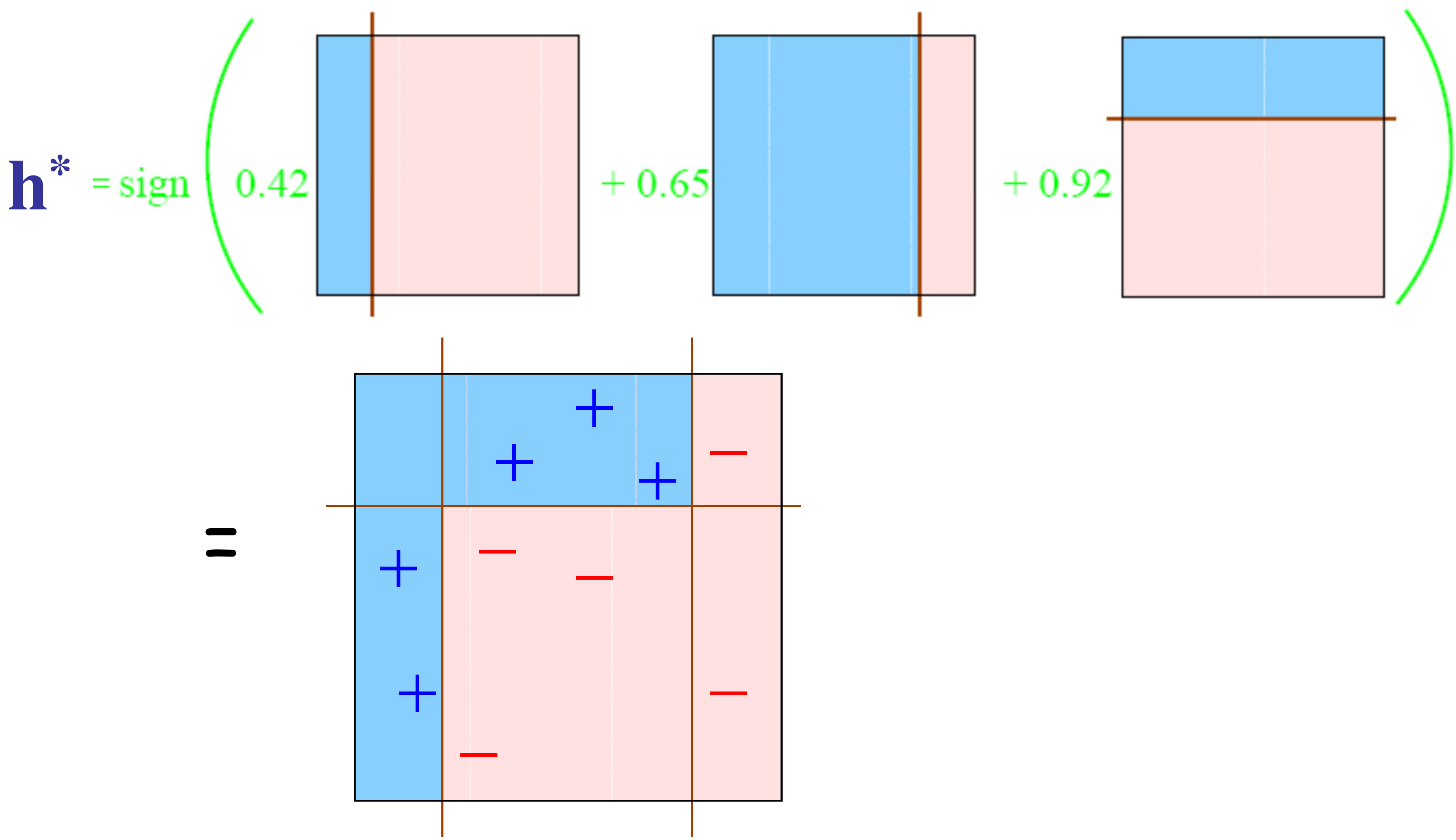
h_1
h_2
D_3
$\varepsilon_1 = 0.30$
$\alpha_1 = 0.42$
$\varepsilon_2 = 0.21$
$\alpha_2 = 0.65$

Round 3



h_1
$\varepsilon_1=0.30$
$\alpha_1=0.42$
h_2
$\varepsilon_2=0.21$
$\alpha_2=0.65$
h_3
$\varepsilon_3=0.14$
$\alpha_3=0.92$

Final Classifier



Learn from weighted instances?

- How can a learning alg use distribution D ?

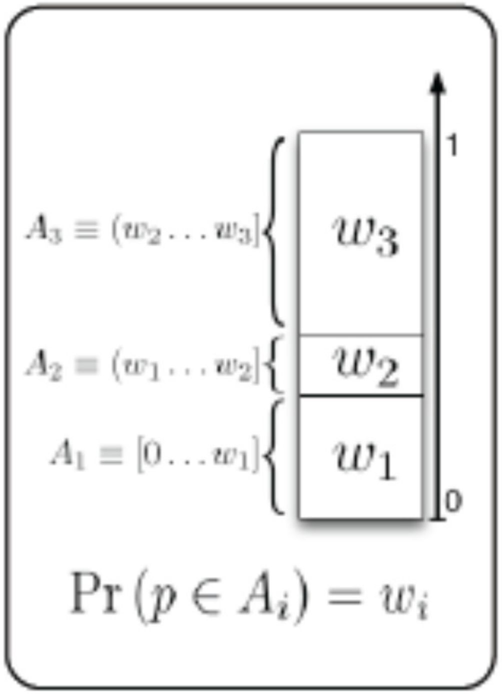
1. Reweighting

- Can modify many learning algorithms to deal with weighted instances:
  - ID3:
    - entropy, information-gain equations use COUNTs #(X=3, C=+) … assumes all weights=1
    - Modify to use *weight* of each instance
  - Naïve Bayes: ditto
  - k-NN: multiple vote from an instance by its weight

Learn from weighted instances?

Resampling

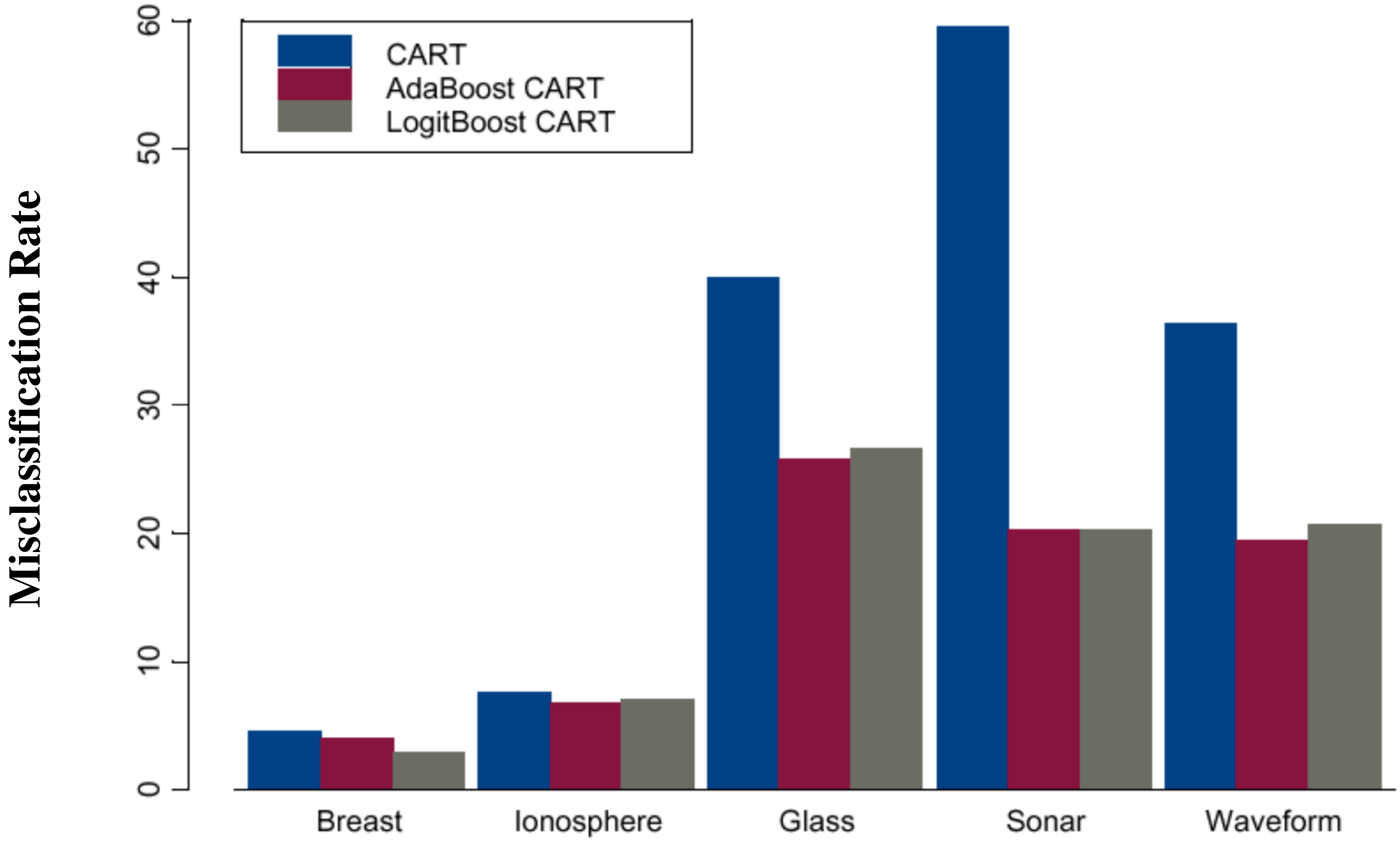
- Given dataset S and distribution D, produce new dataset S′ that embodies D
  - Stochastically
  - Using weight ratio …
- How many?
  - More is good…
  - Typically |S'| = |S|
- If possible, use Re-weighting
  - Re-sampling is only an approximation



Stochastic Resampling…

- Let S′ be the empty set
- Let D = ($w_1, \ldots, w_n$) be the weights of examples in S
  - w_i = D(i) corresponds to example $\mathbf{x}_i$
- While not-enough-samples
  - Draw $n \in [0,1]$ according to U(0,1)
  - $S' \leftarrow S' \cup \{x_k\}$ where k is such that $\sum_{i=1}^{k-1} w_i < n \leq \sum_{i=1}^{k} w_i$
- return S′

Comparison



Friedman, Hastie, Tibshirani [1998]

Analyzing the Training Error

Theorem:

- Let $\gamma_t = ½ - \varepsilon_t$
- $\text{training\_error}(h^*) \le \exp(-2 \sum_t \gamma_t^2)$

- If $\forall t : \gamma_t \ge \gamma > 0$
  then $\text{training\_error}(h^*) \le \exp(-2\gamma^2 T)$
- *AdaBoost* is adaptive:
  - does not need to know γ or T a priori
  - can exploit $\gamma_t >> \gamma$

Proof

- $f(x) = \sum_t \alpha_t h_t(x) \quad \Rightarrow \quad h^*(x) = \text{sign}(\ f(x)\ )$

- Step 1: unwrapping recurrence:

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

$$D_{\text{final}}(i) \quad = \underbrace{\frac{1}{m}}_{D_1} \frac{\exp\left(-y_i \sum_t \alpha_t h_t(x_i)\right)}{\prod_t Z_t} \quad = \quad \frac{1}{m} \frac{\exp\left(-y_i f(x_i)\right)}{\prod_t Z_t}$$

Proof (II)

$$D_{\text{final}}(i) = \frac{1}{m} \frac{\exp(-y_i f(x_i))}{\prod_t Z_t}$$

- Step 2: training_error(h*) $\leq \prod_t Z_t$

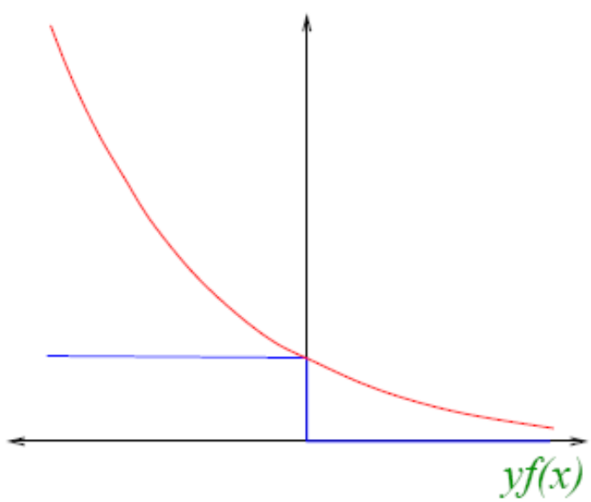
- Proof: training_error(h*) $= \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_i \neq h^*(x_i) \\ 0 & \text{else} \end{cases}$

$$= \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_i f(x_i) \leq 0 \\ 0 & \text{else} \end{cases}$$

$$\leq \frac{1}{m} \sum_i \exp(-y_i f(x_i))$$

$$= \sum_i D_{\text{final}}(i) \prod_t Z_t$$

$$= \prod_t Z_t$$



Proof (III)

- *Step 3*: $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$
- Proof:

$$\begin{aligned} Z_t &= \sum_i D_t(i)\exp(-\alpha_t\, y_i\, h_t(x_i)) \\ &= \sum_{i:y_i\neq h_t(x_i)} D_t(i)e^{\alpha_t} + \sum_{i:y_i=h_t(x_i)} D_t(i)e^{-\alpha_t} \\ &= \epsilon_t\, e^{\alpha_t} + (1-\epsilon_t)\, e^{-\alpha_t} \\ &= 2\sqrt{\epsilon_t(1-\epsilon_t)} \end{aligned}$$

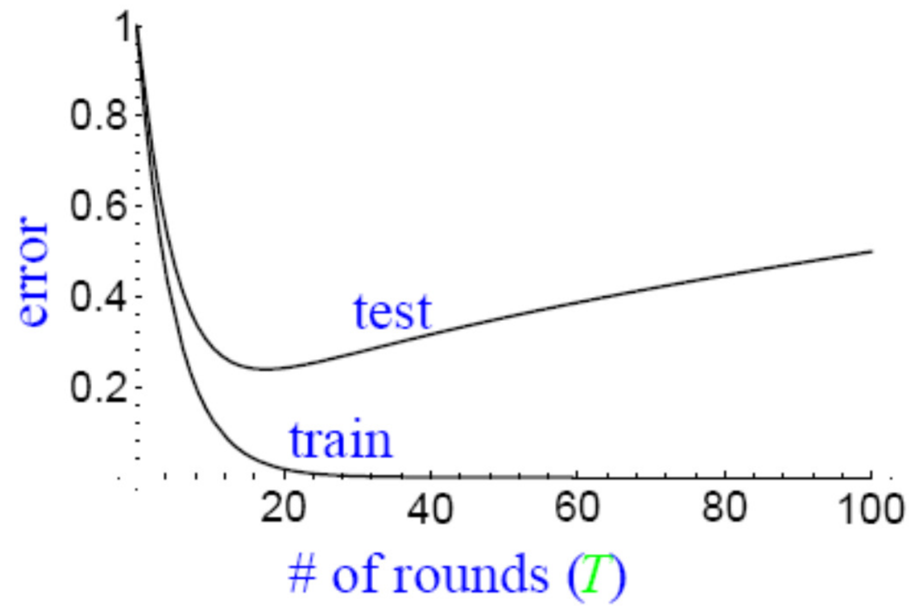
$$\boxed{\varepsilon_t = \sum_{i:y_i\neq h_t(x_i)} D_t(i)}$$

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

Proof (IV)

- Step 4: $\boxed{2\sqrt{\varepsilon(1-\varepsilon} \;\le\; \exp\left(-2\gamma^2\right)}$
- $$2\sqrt{\varepsilon(1-\varepsilon} = \sqrt{4\left(\frac{1}{2}-\gamma\right)\left(1-(\frac{1}{2}-\gamma)\right)} = \sqrt{1-4\gamma^2}$$
- Suffices to show $\boxed{\forall\, 0 \le a \le \frac{1}{4} \quad \sqrt{1-4a} \;\le\; e^{-2a}}$
- True if, for all $a \in [0, \frac{1}{4}]$
  $g(a) = (1 - 4a) - e^{-4a} \le 0$
- $g(0) = 1 - 0 - e^{0} = 0$
  $g'(a) = -4 - (-4)\, e^{-4a} = 4\,(e^{-4a} - 1) \le 0$

How Will Test Error Behave? (A First Guess)

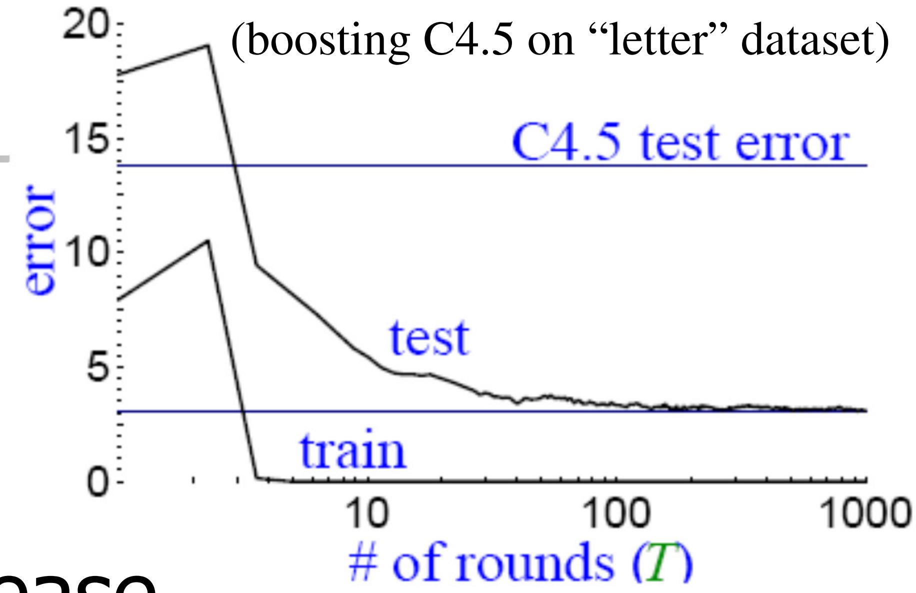


Expect…

- training error to continue to drop (or reach 0)
- test error to increase when h^* becomes "too complex"
  - "Occam's razor"
  - overfitting
    - hard to know when to stop training

Actual Typical Run

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

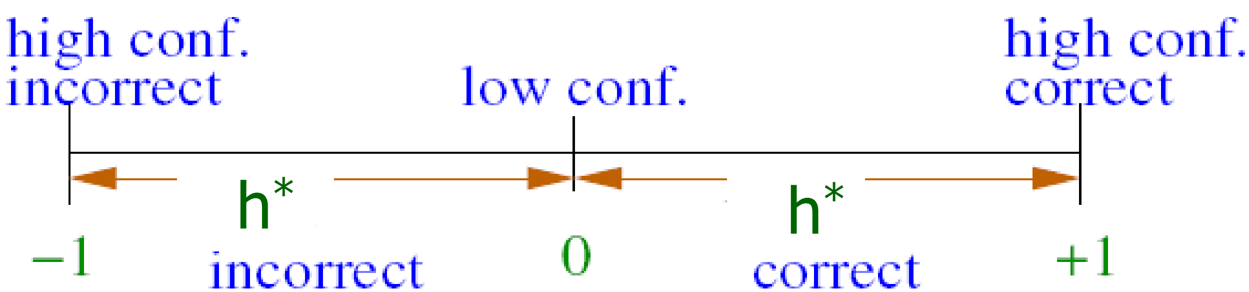


- test error does not increase, even after 1000 rounds
  - (total size > 2,000,000 nodes)
- test error continues to drop, even after training error is 0!
- Occam's razor: "simpler rule is better"... appears to not apply!

[Schapire, Freund, Bartlett & Lee]

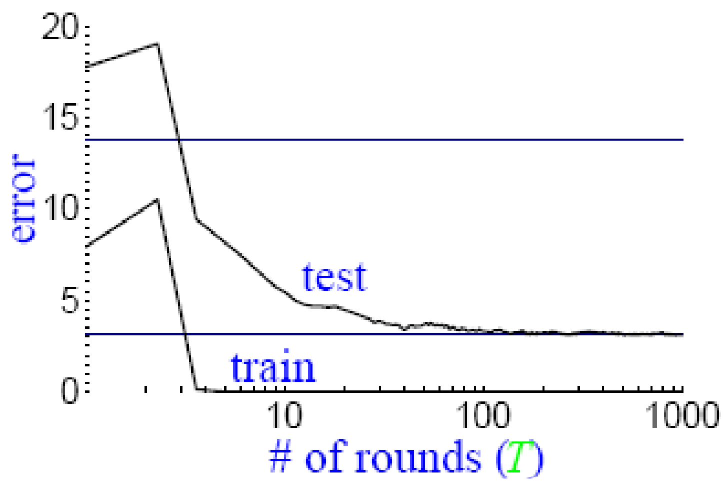
A Better Story: … using Margins

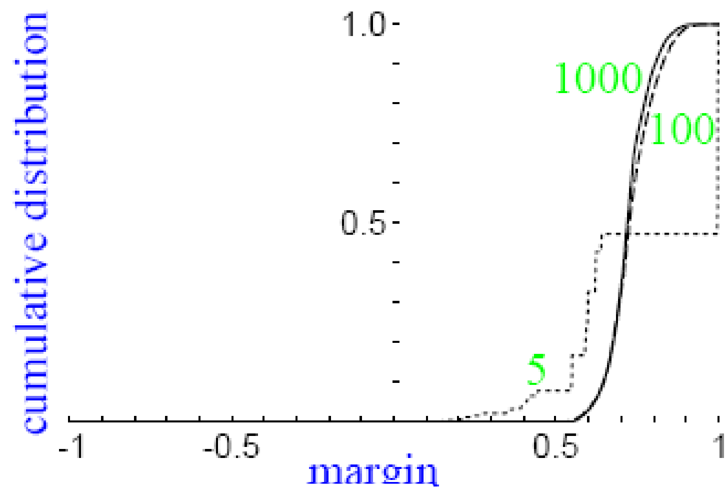
- key idea:
  - training error only measures whether classifications are right or wrong
  - should also consider confidence of classifications
- h^* is weighted majority vote of weak classifiers
- measure confidence by margin
  = strength of the vote
  = (weighted fraction voting correctly)
  – (weighted fraction voting incorrectly)



Empirical: Margin Distribution

- margin distribution
  = cumulative distribution of margins of training ex's





	# 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

Theoretical Evidence: Analyzing Boosting Using Margins

- Theorem:
  Large margins ⇒ better bound on generalization error
  - (independent of # of rounds ≈ complexity of h*)
  - proof idea: if all margins are large, then can approximate final classifier h* by a much smaller classifier
    (just as polls can predict not-too-close election)
- Theorem:
  Boosting tends to increase margins of training examples
  - (given weak learning assumption)
  - proof idea: similar to training error proof
- so:
  although final classifier h* is getting larger,
  margins are likely to be increasing,
  so final classifier h* actually getting close to a simpler classifier,
  driving down the test error

More Technically...

- with high probability, $\forall\theta > 0$:

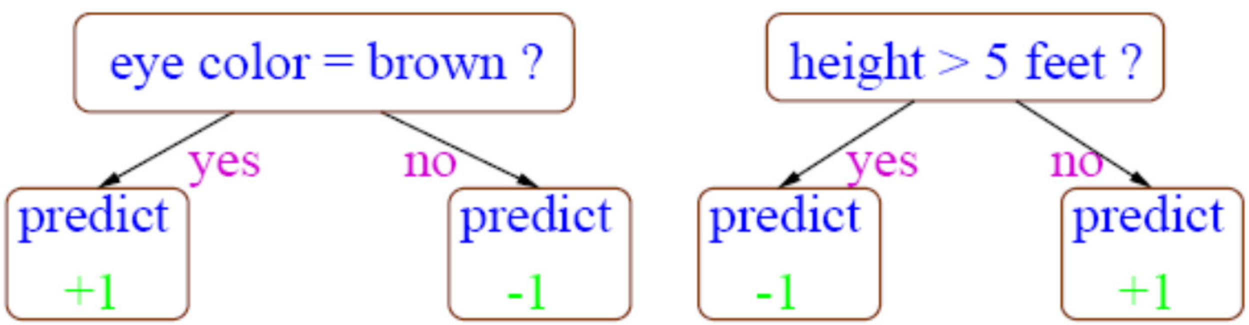
$$\text{generalization error} \le \hat{\Pr}[\text{margin} \le \theta] + \tilde{O}\left(\frac{\sqrt{d/m}}{\theta}\right)$$

($\hat{\Pr}[\ ]$ = empirical probability)

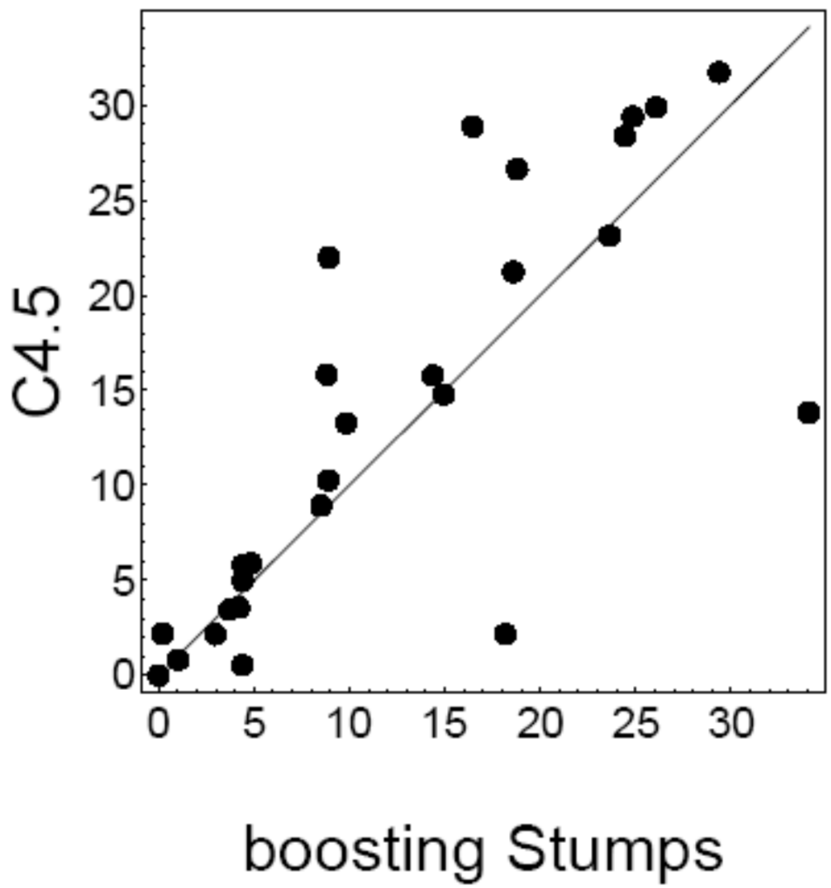
  - bound depends on
    - m = # training examples
    - d = "complexity" of weak classifiers
    - entire distribution of margins of training examples
- $\Pr[\text{margin} \le \theta] \to 0$ exponentially fast (in T) if (error of h_t on D_t) $< \frac{1}{2} - \theta$ ($\forall t$)
  - so: if weak learning assumption holds, then all examples will quickly have "large" margins

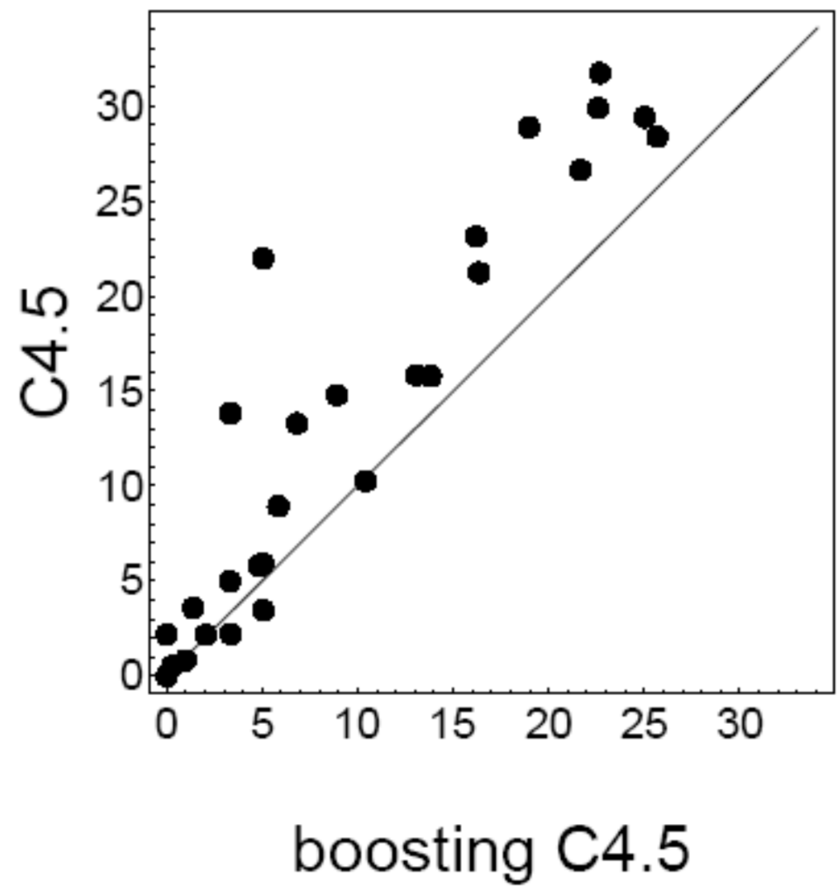
UCI Experiments

- tested AdaBoost on UCI benchmarks
- used:
  - C4.5 (Quinlan's decision tree algorithm)
  - "decision stumps": very simple rules of thumb that test on single attributes



UCI Results


C4.5
0
5
10
15
20
25
30
boosting Stumps


C4.5
0
5
10
15
20
25
30
boosting C4.5

[Schapire, Freund]

Multiclass Problems

- $y \in Y = \{1, . . . , k\}$

$$h_t : X \to Y$$

direct approach (AdaBoost.M1):

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

$$h^*(x) = \arg\max_{y \in Y} \sum_{t: h_t(x) = y} \alpha_t$$

- can prove same bound on error if $\forall t : \varepsilon_t \leq \frac{1}{2}$
  - in practice, not usually a problem for "strong" weak learners (e.g., C4.5)
  - significant problem for "weak" weak learners (e.g., decision stumps)
- instead, reduce to binary…

[Schapire, Singer]

Reducing Multiclass to Binary

- If labels = {a, b, c, d, e}
- replace each training example by five {−1,+1}-labeled examples:

$$x \quad , \quad \mathrm{c} \rightarrow \begin{cases} (x, \mathrm{a}) & , & -1 \\ (x, \mathrm{b}) & , & -1 \\ (x, \mathrm{c}) & , & +1 \\ (x, \mathrm{d}) & , & -1 \\ (x, \mathrm{e}) & , & -1 \end{cases}$$

- predict with label receiving most (weighted) votes

AdaBoost.MH

- can prove:

$$\text{training error}(\mathbf{h}^*) \leq \frac{k}{2} \cdot \prod Z_t$$

  - reflects fact that small number of errors in binary predictors can cause overall prediction to be incorrect
- extends immediately to multi-label case
  - (more than one correct label per example)

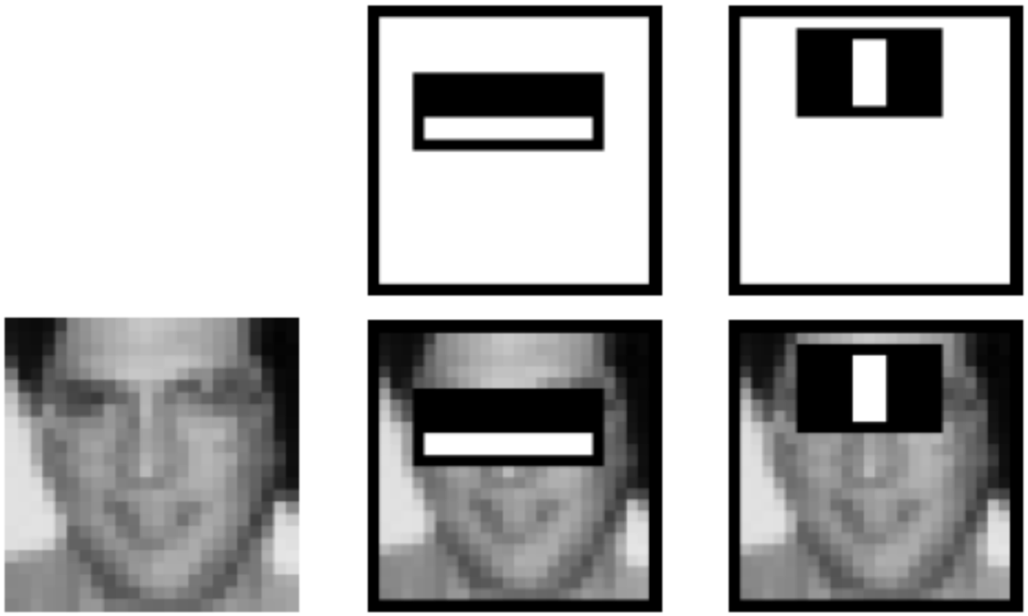
Other Uses of Boosting

- Output code
  - [Schapire, Allwein & Singer] [Dietterich & Bakiri]
- Ranking problems
  - [Schapire, Freund, Iyer & Singer]
- Confidence-rated predictions
  - [Schapire & Singer]
- Face Detection
  - [Viola & Jones]
- Active Learning
  - [Lewis & Gale] [Abe & Mamitsuka]
- Applications:
  - Text Categorization [Schapire & Singer]
  - Human-computer Spoken Dialogue [Schapire,Rahim, Di Fabbrizio, Dutton, Gupta, Hollister & Riccardi]

[Viola & Jones]

Application: Detecting Faces

- problem: find faces in photograph or movie
- weak classifiers: detect light/dark rectangles in image



- many clever tricks to make extremely fast and accurate

Practical Advantages of *AdaBoost*

- fast
- simple and easy to program
- no parameters to tune (except T, sometimes)
- flexible — can combine with any learning algorithm
- no prior knowledge needed about weak learner
- provably effective, given weak classifier
  - $\rightarrow$ shift in mind set — goal now is merely to find classifiers barely better than random guessing
- versatile
  - can use with data that is textual, numeric, discrete, etc.
  - has been extended to learning problems well beyond binary classification

Caveats

- performance of *AdaBoost* depends on data and weak learner
- consistent with theory, AdaBoost can fail if…
  - weak classifiers too complex
    $\rightarrow$ overfitting
  - weak classifiers too weak ($\gamma_t \rightarrow 0$ too quickly)
    $\rightarrow$ underfitting
    $\rightarrow$ low margins $\rightarrow$ overfitting
- empirically, *AdaBoost* seems especially susceptible to uniform noise

Conclusions wrt Boosting

Boosting is a practical tool for classification and other learning problems

- grounded in rich theory
- performs well experimentally
- often (but not always!) resistant to overfitting
- many applications and extensions

Types of Ensemble Methods

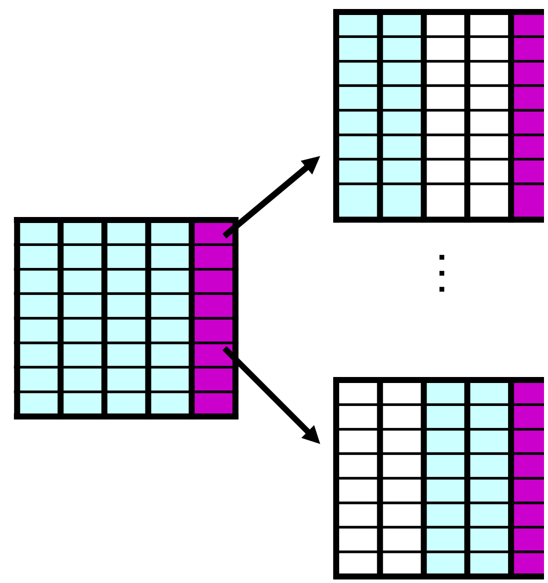
1. Subsample Training Sample
2. Manipulate Input Features
3. Manipulate Output Targets
4. Injecting Randomness
5. Algorithm Specific methods

- Other combinations
- Why do Ensembles work?

2: Manipulate INPUT FEATURES

- Different learners see different subsets of features (of each of the training instances)
- Eg: 119 features for classing volcanoes on Venus
  - Divide into 8 disjoint subsets (by hand)...
  - and use 4 networks for each
  - ... 32 NN classifiers

  Did VERY well [Cherkauer'96]



- Tried w/sonar dataset – 25 input features
  Did NOT work [Tumer/Ghost'96]

- Technique works best when
  input features highly redundant

3: Manipulate OUTPUT Targets

Spse K outputs $Y = \{ y_1, \dots y_K \}$

a. Could learn 1 classifier, into Y ($|Y|$ values)

b. Or could learn K binary classifiers:

- y_1 vs $Y - y_1$
- y_2 vs $Y - y_2$
- ...

then vote

c. Build ln K binary classifiers

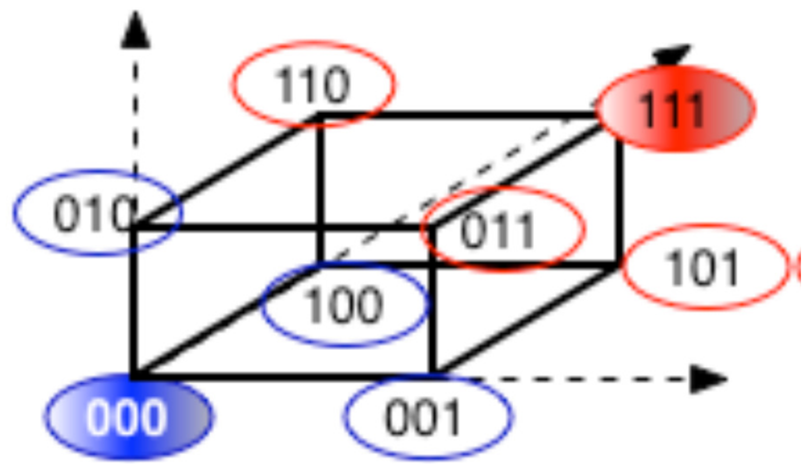
- h_i specifies i^{th} bit of index $\in \{1, 2, \dots, K\}$
- Each h_i sub-classifier splits output-values into 2 subsets
  - $h_0(x)$ is 1 if "$y_1, \dots, y_8$"; else 0
  - $h_1(x)$ is 1 if "$y_1 - y_4$; $y_9 - y_{12}$"; else 0
  - $h_2(x)$ is 1 if "y_1, y_2; y_5, y_6; y_9, y_{10}; y_{13}, y_{14}"; else 0
  - ...

Error Correcting Output Code

- Why not $> \ln K$ binary classifiers . . .
  - "Error-Correcting Codes" (some redundancy)
  - [Dietterich/Bakiri'95]
- View $[\ h_1(x), \ldots, h_m(x)\ ]$ as code-word; return label y_i with nearest codeword
- Better: can combine with AdaBoost
  - [Schapire'97]

Error Correcting Code

- Use **3** bits to encode **2** possible messages
- Codewords {000, 111}
- As differ in >2 places, can detect and correct any "single digit" error!

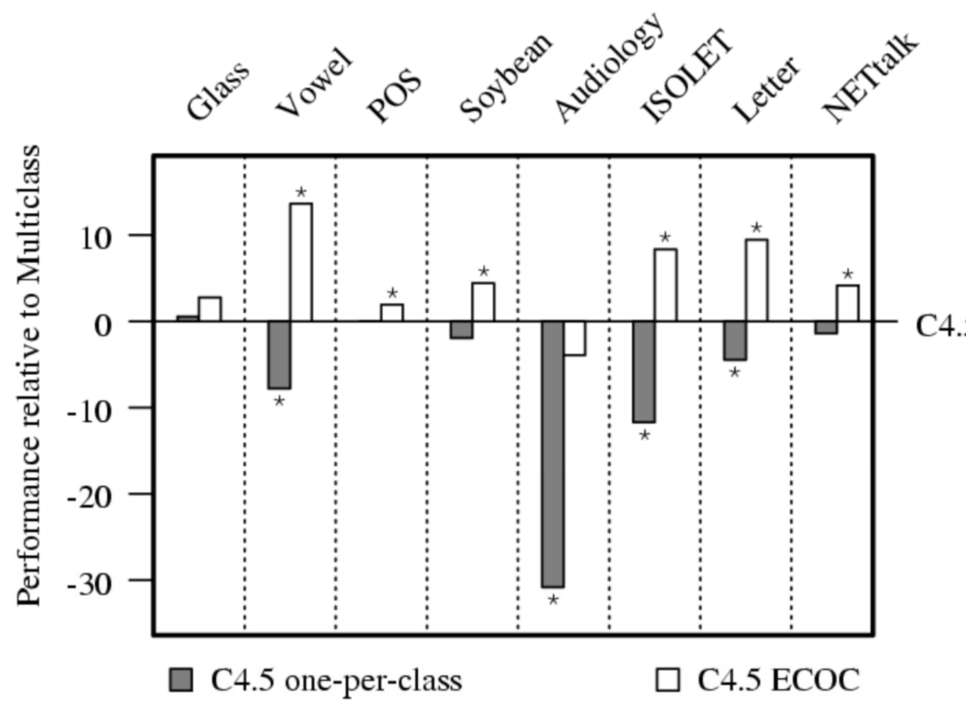


Finding Good Codes

- Lots of tricks…

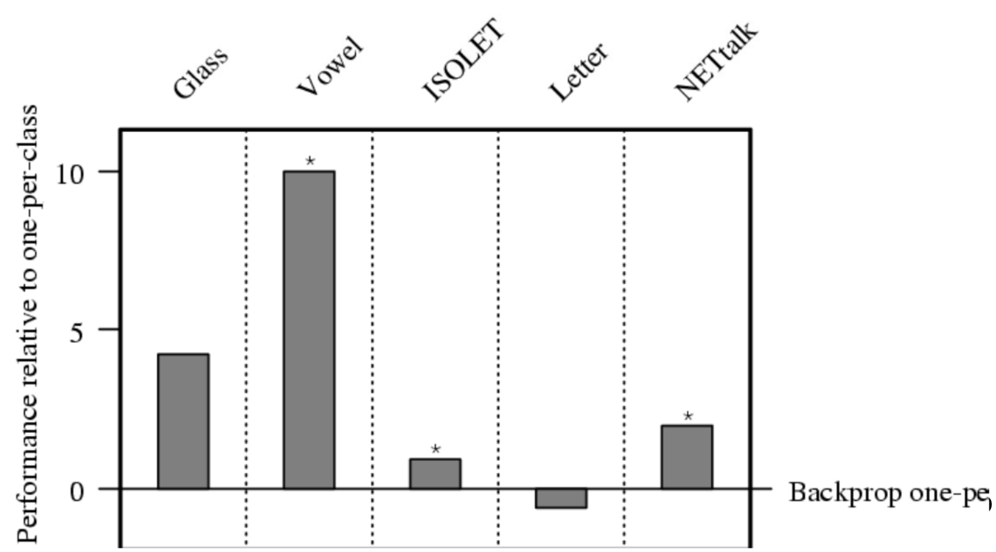
- Simple approach:
  select the codewords *at random*.
- if $2^m \gg k$,
  then obtain a "good" code with high probability
  - such codes work well in practice

Results



% decrease in error of ECOC over an ID3-like learning algorithm

(% decrease in error of ECOC over a neural network learner



Types of Ensemble Methods

1. Subsample Training Sample
2. Manipulate Input Features
3. Manipulate Output Targets
4. Injecting Randomness
   - Data
   - Learner
5. Algorithm Specific methods

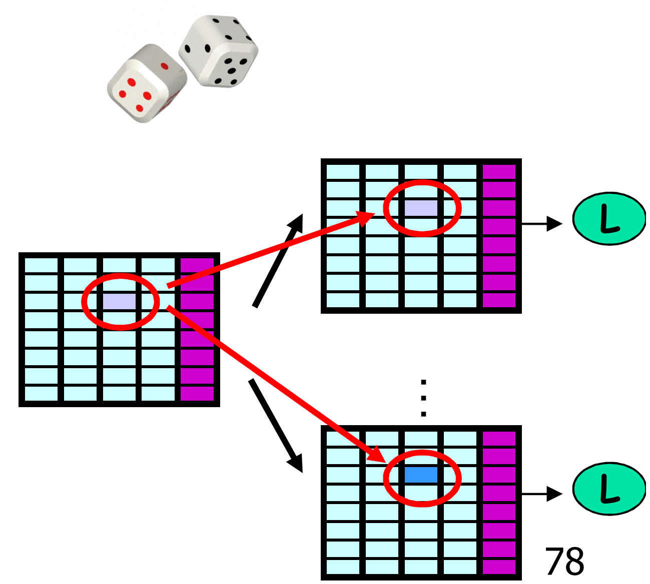
- Other combinations
- Why do Ensembles work?

4a: Injecting Randomness to Data

Add 0-mean Gaussian noise to input features

Draw w/replacement from original data, but add noise

- For Neural Nets:
  - Large improvement on
    + synthetic benchmark;
    + medical Dx
  - [Raviv/Intrator'96]



4b: Injecting Randomness to Learner

- For Neural Nets:

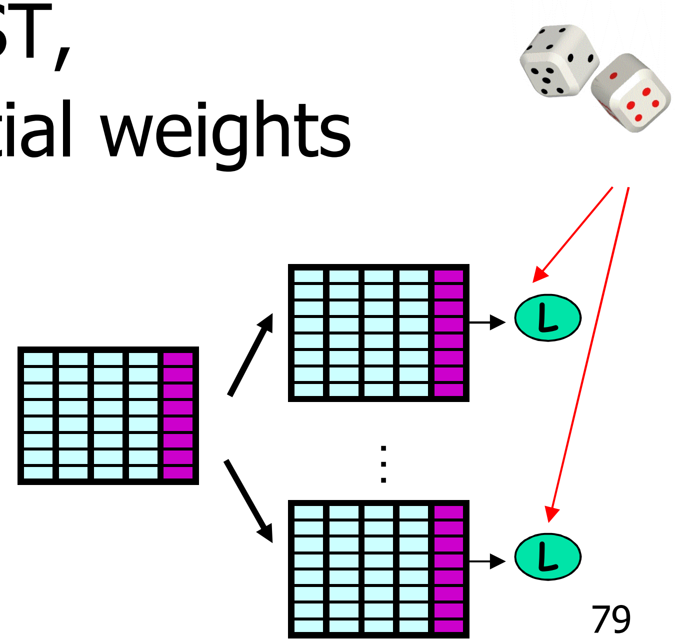
  … Different random initial values of weights

  But really independent?

  Empirical test: [Pamanto, Munro, Doyle 1996]

  Cross-validated committees BEST,

  then Bagging, then Random initial weights



Randomness – w/ C4.5

- C4.5 uses Info Gain to decide which attribute to split on
  - Why not consider top 20 attributes;
    choose one at random?

    $\Rightarrow$ Produce 200 classifiers (same data)
  - To classify new instance: Vote
  - Empirical test: [Dietterich/Kong 1995]

    Random better than bagging, than single C4.5
- FOIL (for learning Prolog-like rules)
  - Chose any test whose info gain within 80% of top
  - Ensemble of 11
    STATISTICALLY BETTER
    than 1 run of FOIL [Ali/Pazzani'96]

5: Algorithm Specific (NNs)

Seek “diverse” population of NNs

- Simultaneously train several NN's with penalty for correlations.
  Backprop minimizes error function =
  sum of MSE and correlations [Rosen'96]
- Use operators to build new structures; keep R “best”
  - DIVERSITY + ACCURACY

  (like GA [Opitz/Shavlik'96])
- Give different NNs different auxiliary tasks
  - (eg, predict one input feature)

  in addition to primary task
  Backprop use BOTH in error, so produces different nets
  [Abu-Mostafa'90; Caruana'96]
- For each $[x_i, y_i]$, re-train NN_j with
  - $[x_i, [y_i, 1]]$ if $NN_j(x_i)$ closest to y_i
  - $[x_i, [y_i, 0]]$ otherwise

  (So diff NNs get different training values, to help NN learn where it performs best) [Munro/Parmanto'97]

Algorithm Specific (NN #2)

- Person identifies which region of input space
  - (Highway, 2lane-road, dirt-road, ...)

  Train NN_i for $region_i$ ... eg, to steer, . . .
- Each NN_i also learns to reconstruct image
  - Same intermediate layer!
- When "running", each NN_i
  - proposes steering direction,
  - reconstructs of image

  Take direction from NN_i with best reconstruction [Pomerleau]
- Also: train on "bad" situation,

  by distorting image, and defining correct label

Algorithm Specific (DTs, ...)

- "Option tree":
  Decision Tree whose internal nodes have > 1 splits,
  each producing own sub-decision-tree
  - (Eval: go down each, then vote) [Buntine'90]
- Empirical: accuracy ≈ bagged C4.5 trees
  but MUCH more understandable
- Can try different modalities,
  but not clear how DIVERSE they will be
  - Use cross-validation to check for both accuracy and diversity

Combining Classifier: Linear

Linear Combination

- Unweighted: Bagging, ErrorCorrecting, Boosted (weighted)

- **Bayesian Model**
  If each h_t produces class prob. estimates
  $P(\ f(x) = y \mid h_t\ )$
  should use:
  $$P(\ f(x) = y\ ) = \sum_t P(\ f(x) = y \mid h_t\ )\ P(h_t)$$
  - Forecasting lit. suggests this is very robust [Clemen'89]
- **Variance-based**
  - Use least squares regression to find weights that max accuracy on training data
  - Uncorrelated $\Rightarrow$ h_t 's weight $\propto$ $1/\mathrm{Var}(h_t)$
    Can also deal w/ less correlated subset

Combining Classifiers: Linear, II

Linear Combination (con't)

- **Gating** [Jordan/Jacobs'94]
  - Learn classifier's { $h_1, \dots, h_T$ }
  - $\text{output}(\mathbf{x}) = \sum_t w_t \, h_t(\mathbf{x})$
  - $w_t(\mathbf{x}) = \exp(\mathbf{v}_t \, \mathbf{x}) \,/ \sum_u \exp(\mathbf{v}_u \, \mathbf{x})$
  - Problem: lot of parameters to learn: $\{\mathbf{v}_u\}$, as well as params for all h_t\`s
- **Cross-Validation** [Ali/Pazzani'96; Buntine'90]
  - Obtain weights from performance on hold-out set

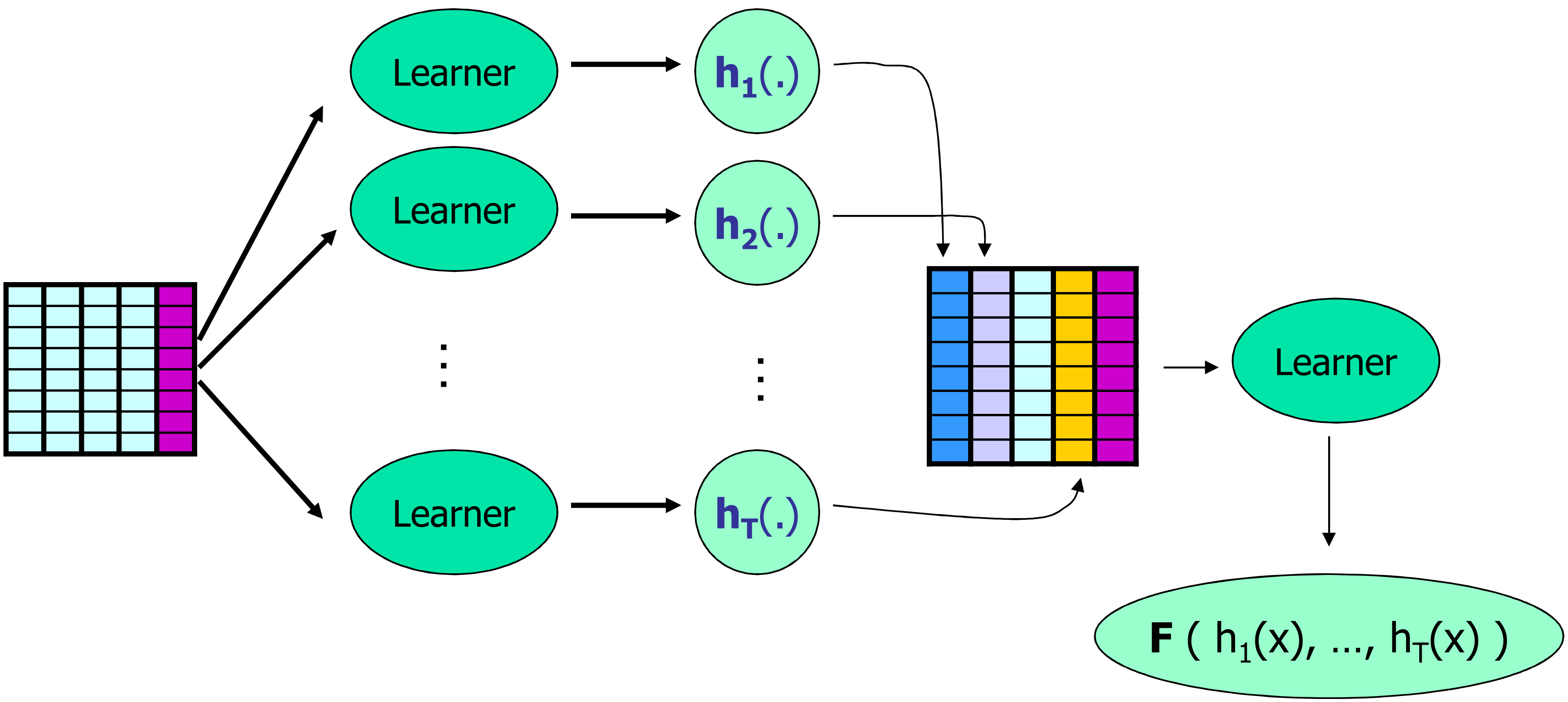
Combining Classifiers: NonLinear

Stacking [Wolper'92; Breiman'96]

- Given learners { $L_i(.)$ }, obtain $h_i = L_i(\ S\ )$
- Want classifier $h^*(x) = F(\ h_1(x), \ldots, h_T(x)\ )$
- Let $h_t^{(-i)} = L_t(S - \mathbf{x}_i)$ be classifier learned using L_t, on all but instance $\mathbf{x}_i$

  … so $T \times |S|$ classifiers
- Let $ý_i^{(t)} = h_t^{(-i)}(\mathbf{x}_i)$
- Now learn $F(\ldots)$ from $\{\ [\ [ý_i^{(1)}, ý_i^{(2)}, \ldots, ý_i^{(T)}], y_i\ ]\ \}_i$

Stacking



Why do ensemble work?

Many reasons justify ensemble approach:

- Bias/Variance decomposition
- A(nother) statistical motivation
- Representational issues
- Computational issues

Why do ensembles work? (AdaBoost)

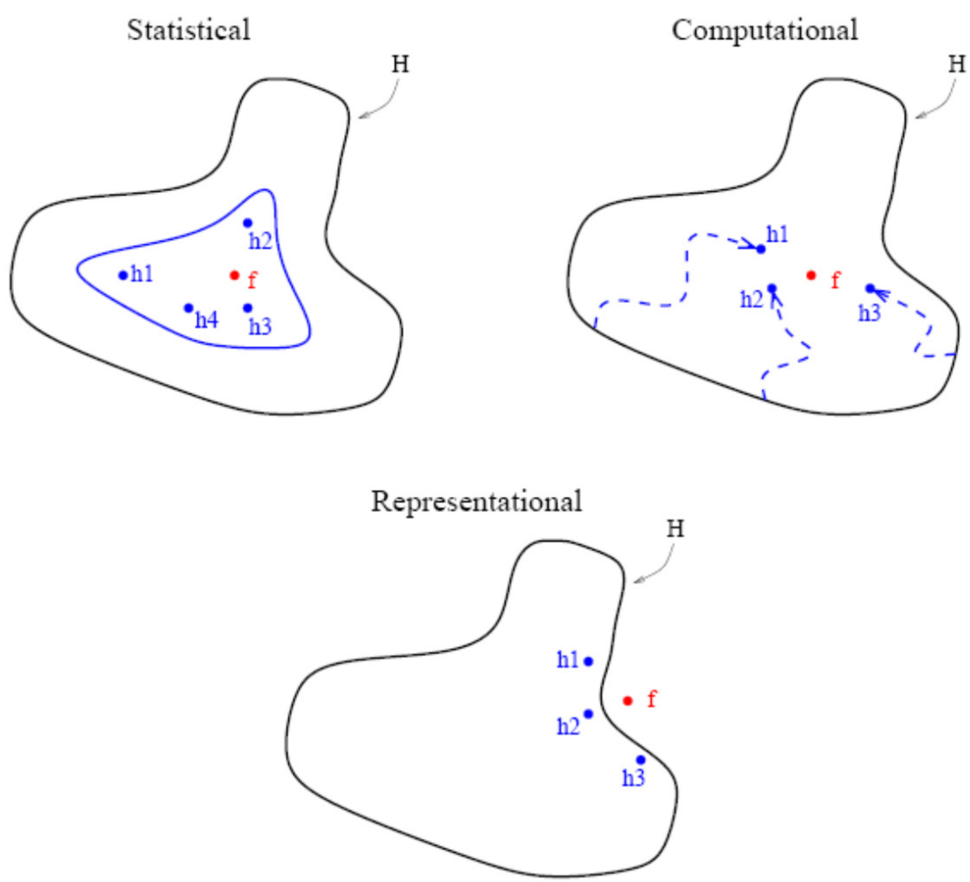
- **Empirical** evidence suggests that *AdaBoost* reduces both **bias** and **variance** part of the error
  - bias is mostly reduced in early iterations
  - while variance in later ones

Lesson learned?

- **Use Bagging with low bias and high variance classifiers**
  - e.g., decision trees, 1-nn, ...

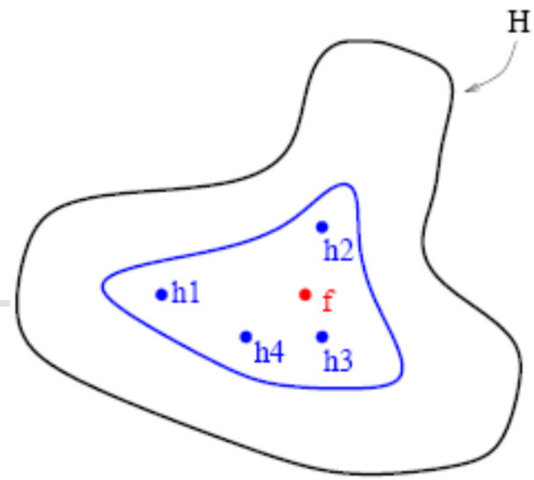
- **Always try AdaBoost** 
  - Typically produces excellent results.
  - Works especially well with very simple learners
    - eg, decision stumps

Other explanations?



[T. G. Dietterich. *Ensemble methods in machine learning*. Lecture Notes in Computer Science, 1857:1–15, 2000.]

1. Statistical

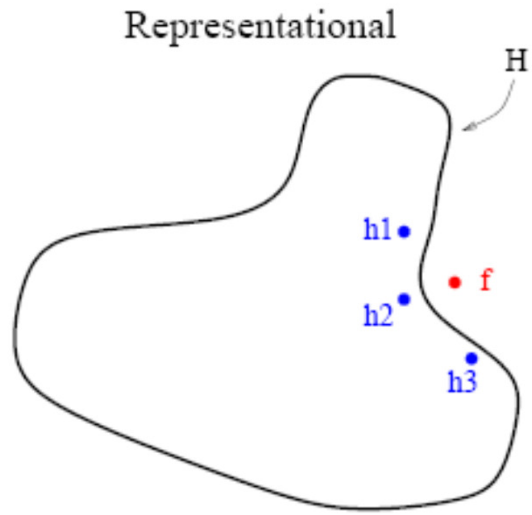


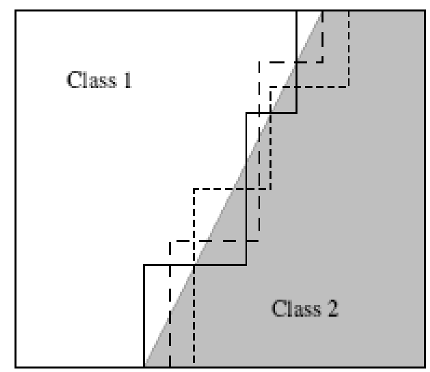
- Given a finite amount of data, many hypothesis are typically equally good. How can the learning algorithm select among them?
- **Optimal Bayes classifier recipe**:
  - take a *weighted* majority vote of *all* hypotheses,
  - weighted by their posterior probability
  - **...provably** the best possible classifier
- Ensemble learning ≈ approximation of the Optimal Bayes rule

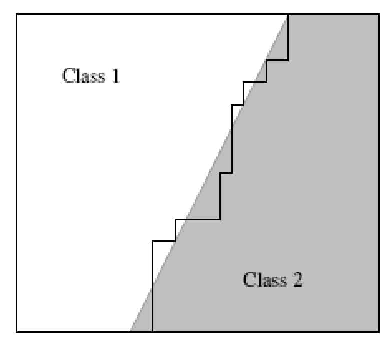
2. Representational

- Optimal target function may not be ANY individual classifier,

  but may be (approximated by) ensemble averaging
- Eg… a decision trees
  - boundaries are axis-parallel hyperplanes
  - By averaging a large number of such "staircases", can approximate diagonal decision boundary with arbitrarily good accuracy



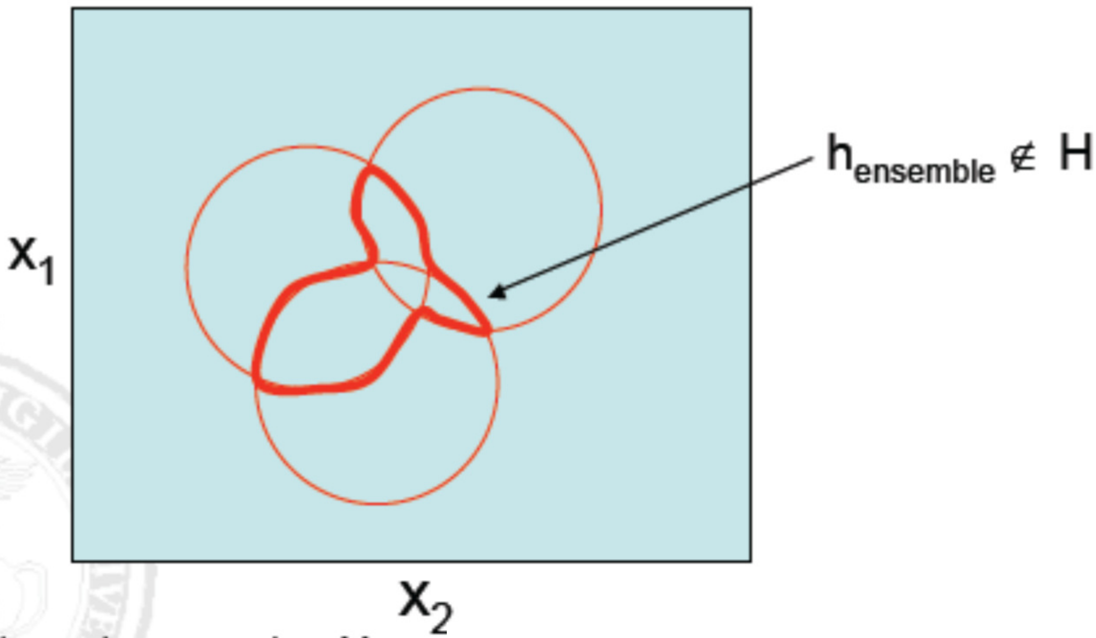




Representational (example 2)

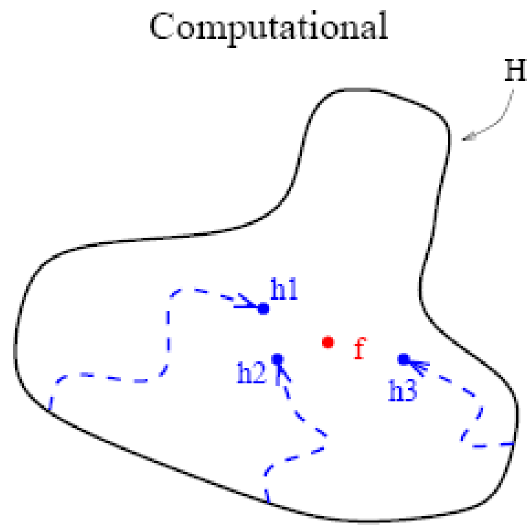
- Space: [0,1] x [0,1]
- Hypothesis space H of “discs”

$h_1, h_2, h_3 \in H$



$h_{ensemble}$ **cannot be return by “base” learner, but** $h_{ensemble}$ **can be returned by ensemble**

3. Computational



- Essentially all learning alg's search through some space of hypotheses to find one that is "good enough" for the given training data
- As many interesting hypothesis spaces are huge/infinite, heuristic search is essential
  - (eg ID3 greedily search in space of decision trees)
- Learner might get stuck in a local minimum
- One strategy for avoiding local minima: repeat the search many times with random restarts

  $\rightarrow$ bagging

Summary of Ensembles

- Ensembles: basic motivation
  creating a **committee of experts** is typically more effective than creating a single **supergenius**
- Key issues:
  - Generating base models
  - Integrating responses from base models
- Popular ensemble techniques
  - manipulate training data: bagging and boosting (ensemble of "experts", each specializing on different portions of the instance space)
  - manipulate output values: error-correcting output coding (ensemble of "experts", each predicting 1 bit of the {multibit} full class label)
- Why does ensemble learning work?