Ensemble learning, Bagging, Boosting

Somak Aditya,
Sudeshna Sarkar

Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav Gogate, Rob Schapire, and Tommi Jaakkola

Ensemble methods

Machine learning competition with a $1 million prize

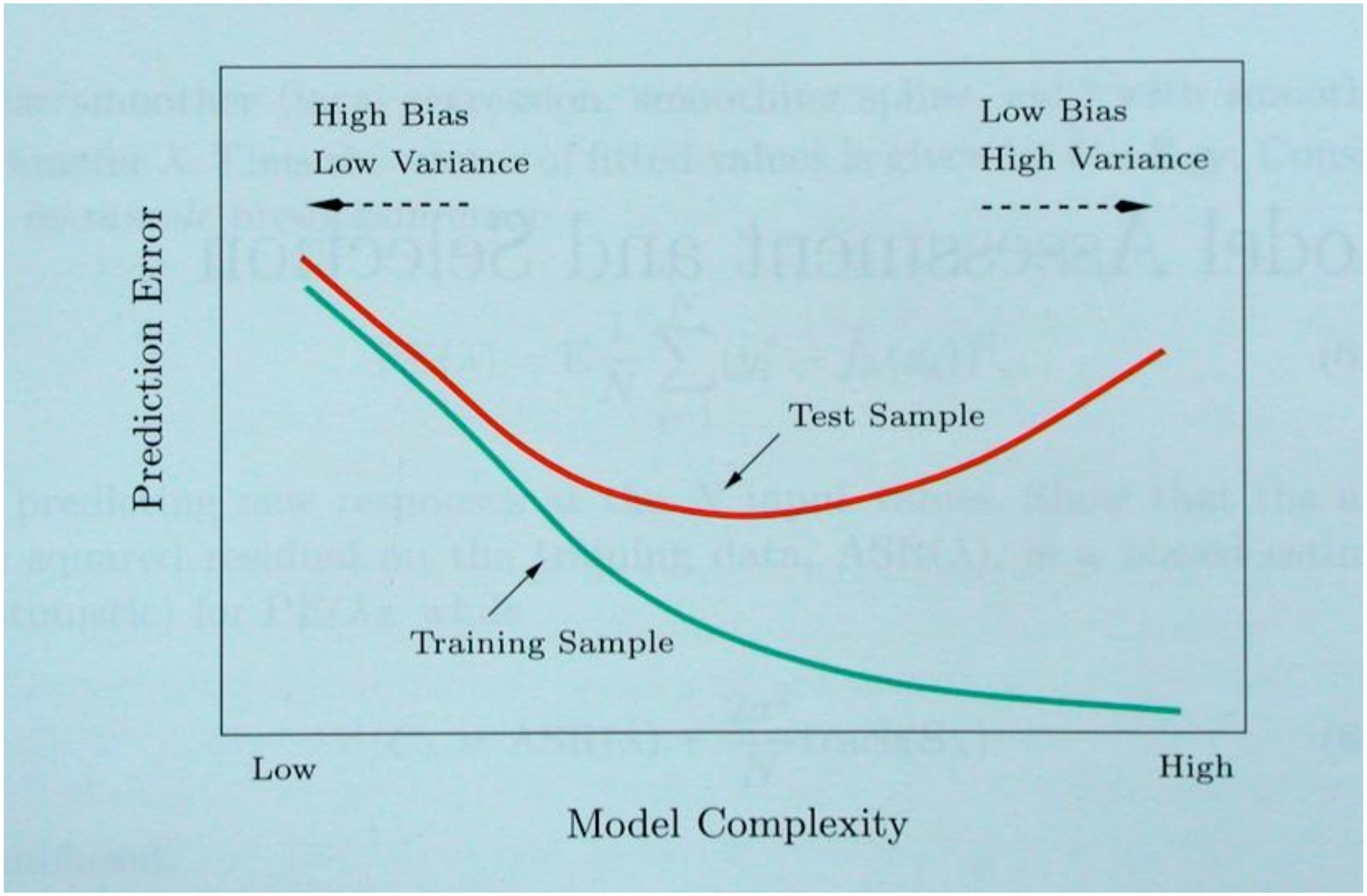
Leaderboard

Display top 20 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	The Ensemble	0.8553	10.10	2009-07-26 18:38:22
2	BellKor's Pragmatic Chaos	0.8554	10.09	2009-07-26 18:18:28
Grand Prize - RMSE <= 0.8563				
3	Grand Prize Team	0.8571	9.91	2009-07-24 13:07:49
4	Opera Solutions and Vandelay United	0.8573	9.89	2009-07-25 20:05:52
5	Vandelay Industries !	0.8579	9.83	2009-07-26 02:49:53
6	PragmaticTheory	0.8582	9.80	2009-07-12 15:09:53
7	BellKor in BigChaos	0.8590	9.71	2009-07-26 12:57:25
8	Dace_	0.8603	9.58	2009-07-24 17:18:43
9	Opera Solutions	0.8611	9.49	2009-07-26 18:02:08
10	BellKor	0.8612	9.48	2009-07-26 17:19:11
11	BigChaos	0.8613	9.47	2009-06-23 23:06:52
12	Feeds2	0.8613	9.47	2009-07-24 20:06:46
Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8633	9.26	2009-07-21 02:04:40
14	Gravity	0.8634	9.25	2009-07-26 15:58:34
15	Ces	0.8642	9.17	2009-07-25 17:42:38
16	Invisible Ideas	0.8644	9.14	2009-07-20 03:26:12
17	Just a guy in a garage	0.8650	9.08	2009-07-22 14:10:42
18	Craig Carmichael	0.8656	9.02	2009-07-25 16:00:54
19	J Dennis Su	0.8658	9.00	2009-03-11 09:41:54
20	acmehill	0.8659	8.99	2009-04-16 06:29:35
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell				
Cinematch score on quiz subset - RMSE = 0.9514				

	users					
movies	1		?	3	5	?
	?	1				2
		4		4	5	?

Bias/Variance Tradeoff



Hastie, Tibshirani, Friedman "Elements of Statistical Learning" 2001

Reduce Variance Without Increasing Bias

- We can sample m independent training sets. Then, we can compute y by averaging all the m predictions from m models $y = \frac{1}{m}\sum_{i=1}^{m} y_i$

  - **Bias: unchanged**, since the averaged prediction has the same expectation

$$\mathbb{E}[y] = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = \mathbb{E}[y_i]$$

  - **Variance: reduced**, since we're averaging over independent samples

$$\mathrm{Var}[y] = \mathrm{Var}\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = \frac{1}{m^2}\sum_{i=1}^{m}\mathrm{Var}[y_i] = \frac{1}{m}\mathrm{Var}[y_i].$$

- Averaging reduces variance:

$$Var(\bar{X}) \quad \frac{Var(X)}{N}$$

When predictions are independent

Average models to reduce model variance

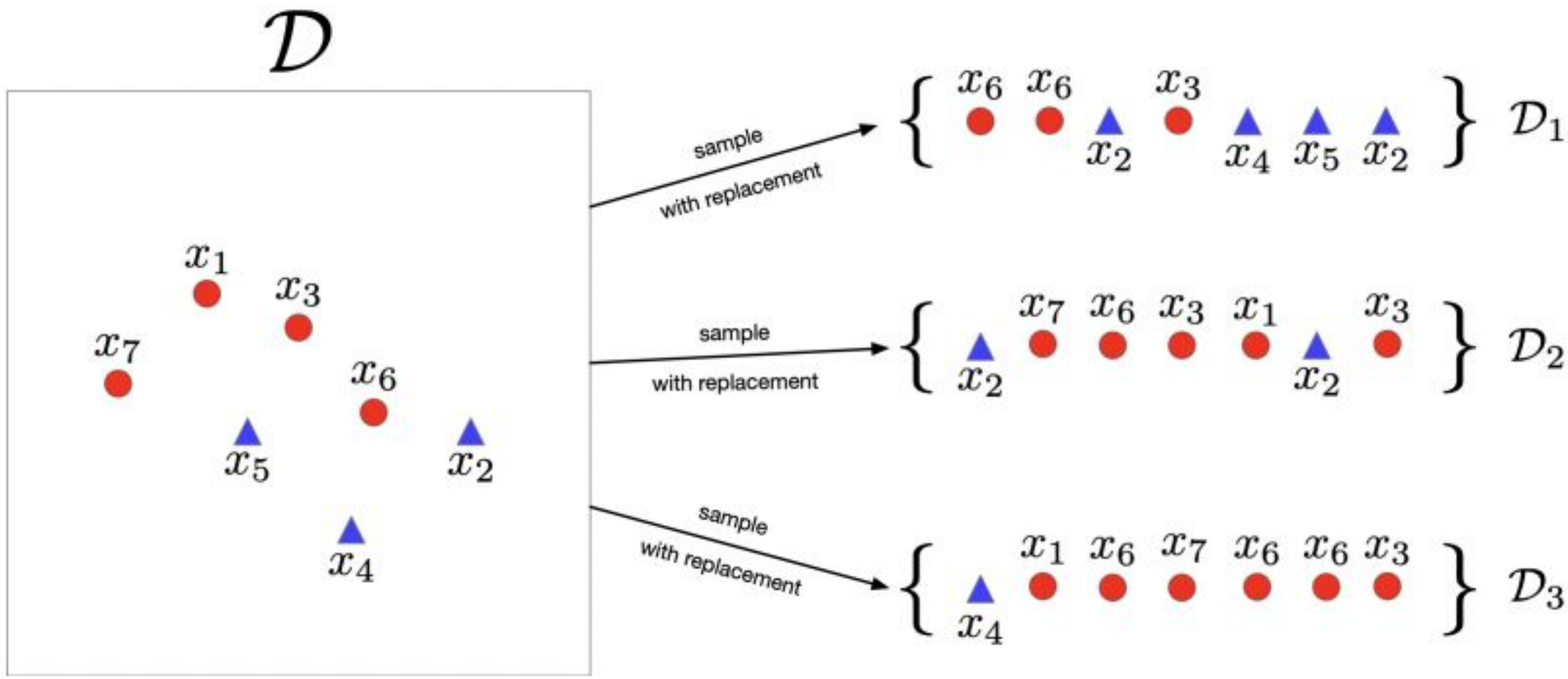
One problem: only one training set, where do multiple models come from?

Bagging: Bootstrap Aggregation

- Leo Breiman (1994)
- Take repeated bootstrap samples from training set *D*.
- *Bootstrap sampling*: Given set *D* containing *N* training examples, create *D*' by drawing *N* examples at random with replacement from *D*.

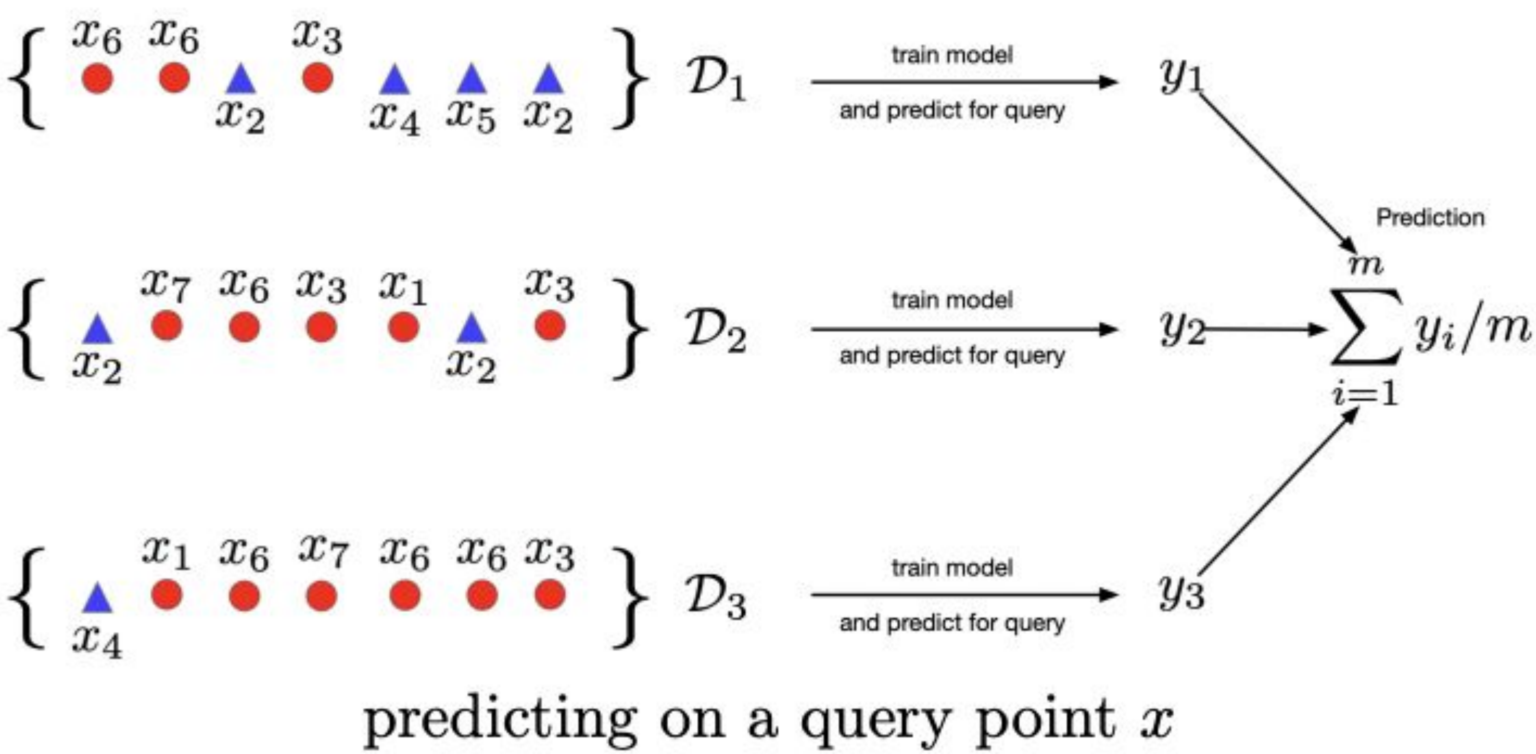
- Bagging:
  - Create *k* bootstrap samples $D_1 \ldots D_k$.
  - Train distinct classifier on each D_i.
  - Classify new instance by majority vote / average.

Bagging



in this example $n = 7$, $m = 3$

Bagging



x_6 x_6 x_3
x_2 x_4 x_5 x_2
$\mathcal{D}_1$
train model
and predict for query
y_1
x_7 x_6 x_3 x_1 x_3
x_2 x_2
$\mathcal{D}_2$
train model
and predict for query
y_2
Prediction
$\sum_{i=1}^{m} y_i/m$
x_1 x_6 x_7 x_6 x_6 x_3
x_4
$\mathcal{D}_3$
train model
and predict for query
y_3
predicting on a query point x

Bagging

- Best case:

$$Var(Bagging(L(x, D))) \quad \frac{Variance(L(x, D))}{N}$$

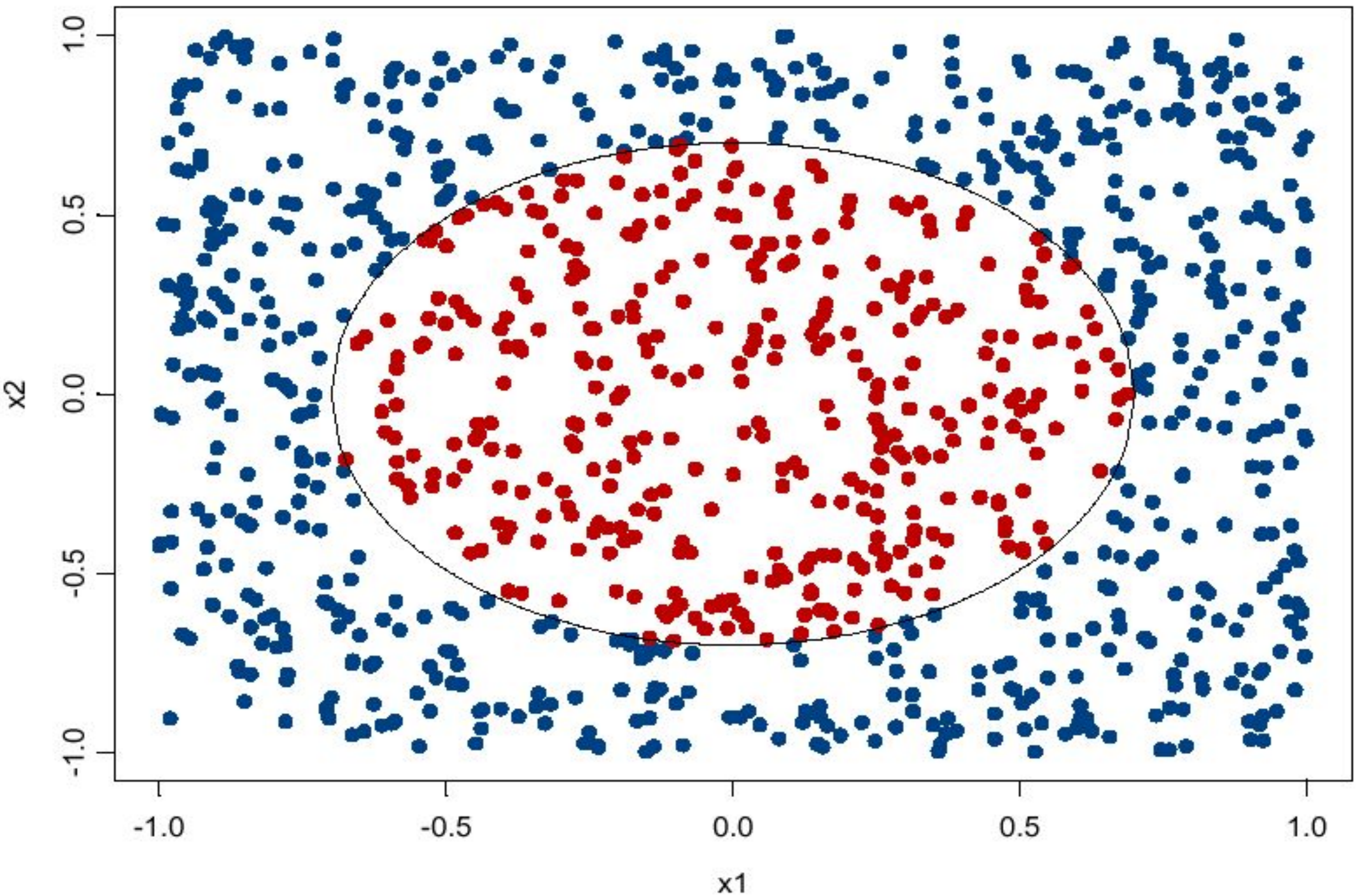
In practice:

models are correlated, so reduction is smaller than 1/N

variance of models trained on fewer training cases

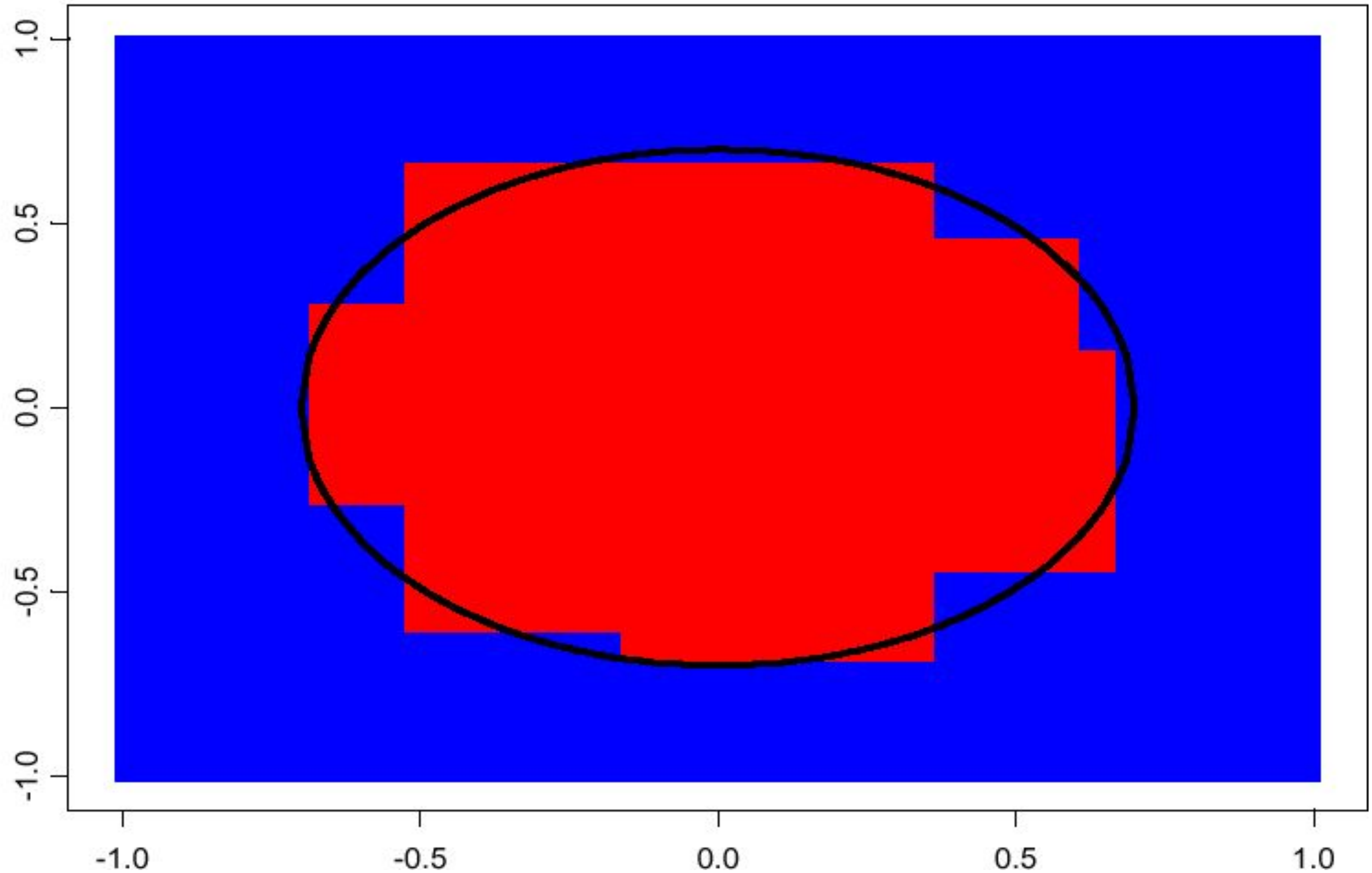
usually somewhat larger

Bagging Example

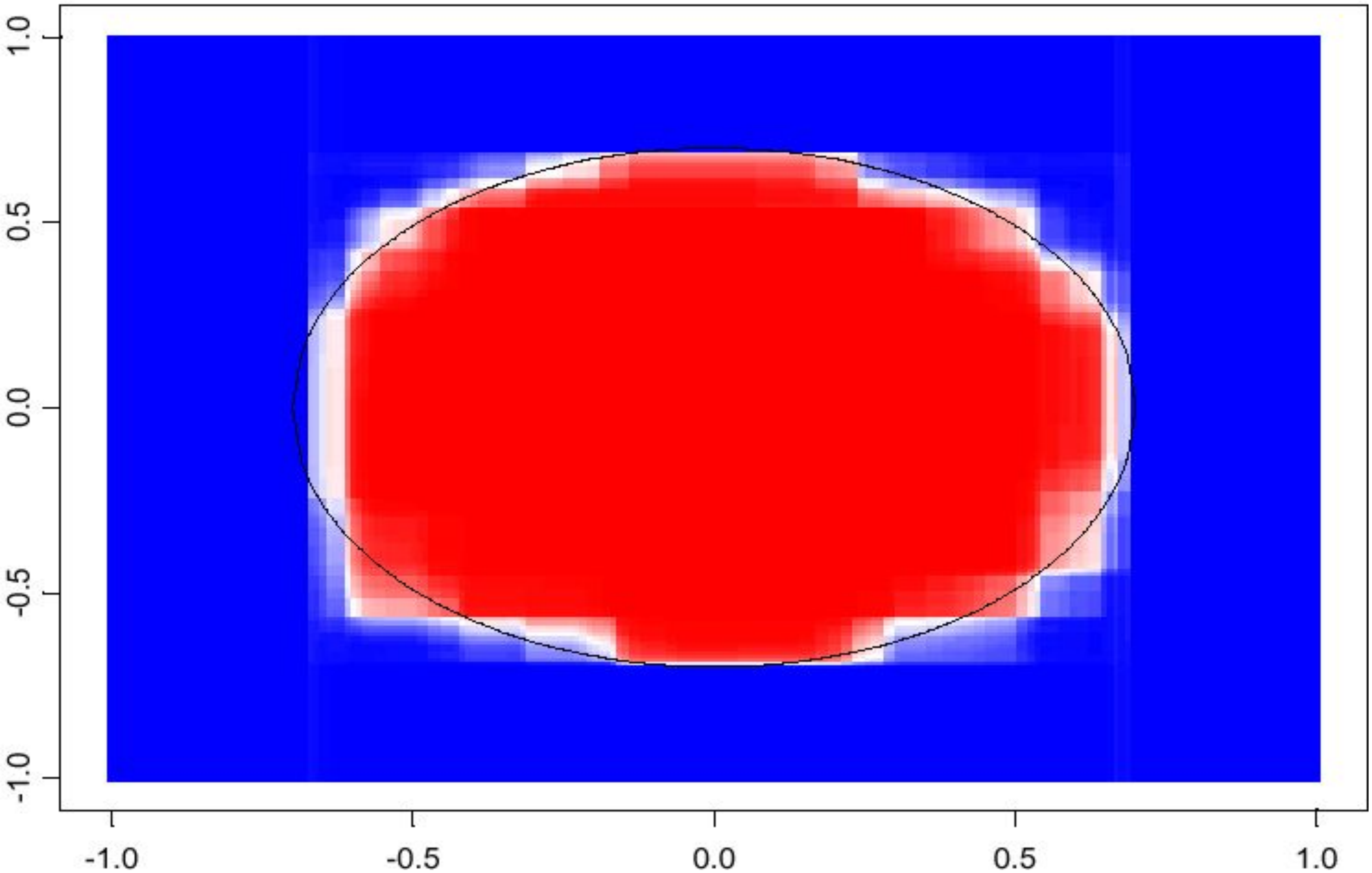


decision tree learning algorithm; very similar to ID3

CART decision boundary



100 bagged trees



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

Reduce Bias^2 and Decrease Variance?

- Bagging reduces variance by averaging
- Bagging has little effect on bias
- Can we average *and* reduce bias?
- Yes:

- Boosting

Theory and Applications of Boosting

Rob Schapire

Example: "How May I Help You?"

[Gorin et al.]

- goal: automatically categorize type of call requested by phone customer (Collect, CallingCard, PersonToPerson, 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)
  - `I just called a number in sioux city and I musta rang the wrong number because I got the wrong party and I would like to have that taken off of my bill` (BillingCredit)
- observation:
  - easy to find "rules of thumb" that are "often" correct
    - e.g.: "IF '`card`' occurs in utterance
      THEN predict 'CallingCard' "
  - hard to find single highly accurate prediction rule

The Boosting Approach

- devise computer program for deriving rough rules of thumb
- apply procedure to subset of examples
- obtain rule of thumb
- apply to 2nd subset of examples
- obtain 2nd rule of thumb
- repeat T times

Key Details

- how to choose examples on each round?
- concentrate on “hardest” examples
  - (those most often misclassified by previous rules of thumb)
- how to combine rules of thumb into single prediction rule?
  - take (weighted) majority vote of rules of thumb

Boosting

- boosting = general method of converting rough rules of thumb into highly accurate prediction rule
- technically:
  - assume given "weak" learning algorithm that can consistently find classifiers ("rules of thumb") at least slightly better than random, say, accuracy ≥ 55%
    - (in two-class setting) ["weak learning assumption"]
  - given sufficient data, a boosting algorithm can provably construct single classifier with very high accuracy, say, 99%

Strong and Weak Learnability

- boosting's roots are in "PAC" learning model [Valiant '84]
- get random examples from unknown, arbitrary distribution
- strong PAC learning algorithm:
  - for any distribution
    with high probability
    given polynomially many examples (and polynomial time)
    can 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?

If Boosting Possible, Then...

- can use (fairly) wild guesses to produce highly accurate predictions
- if can learn “part way” then can learn “all the way”
- should be able to improve any learning algorithm
- for any learning problem:
  - either can always learn with nearly perfect accuracy
  - or there exist cases where cannot learn even slightly better than random guessing

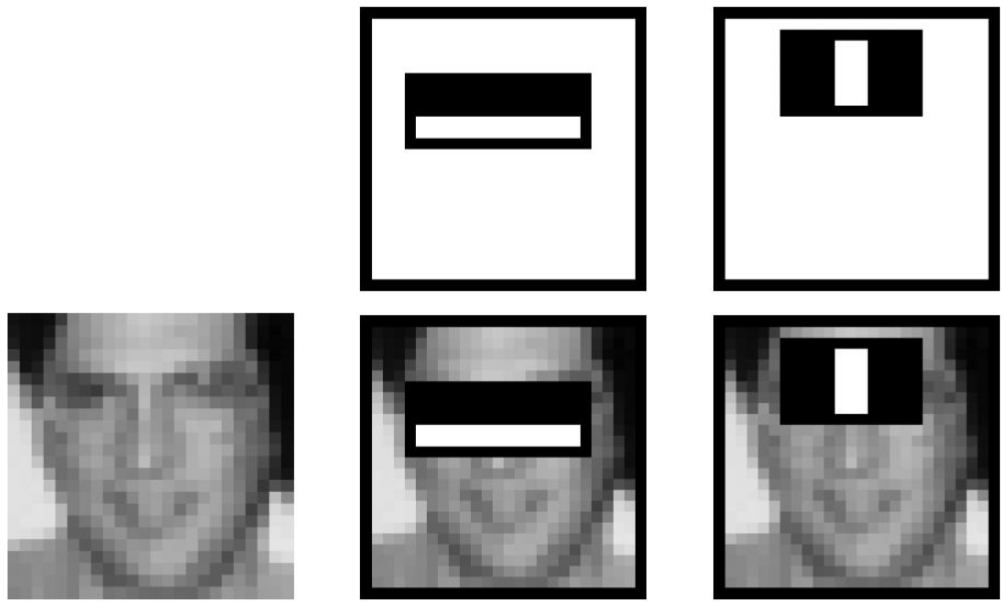
First Boosting Algorithms

- [Schapire ’89]:
  - first provable boosting algorithm
- [Freund ’90]:
  - “optimal” algorithm that “boosts by majority”
- [Drucker, Schapire & Simard ’92]:
  - first experiments using boosting
  - limited by practical drawbacks
- [Freund & Schapire ’95]:
  - introduced “AdaBoost” algorithm
  - strong practical advantages over previous boosting algorithms

Application: Detecting Faces

[Viola & Jones]

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



- many clever tricks to make extremely fast and accurate

Basic Algorithm and Core Theory

- introduction to AdaBoost
- analysis of training error
- analysis of test error and the margins theory
- experiments and applications

Adaboost

Given: a class $\mathcal{F} = \{f : \mathcal{X} \mapsto \{-1, 1\}\}$ of weak learners and the data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, $y_i \in \{-1, 1\}$. Initialize the weights as $w_1(i) = 1/n$.
For $t = 1, \ldots T$:

1. Find a weak learner f_t based on weights $w_t(i)$;

2. Compute the *weighted* error $\epsilon_t = \sum_{i=1}^{n} w_t(i) I(y_i \neq f_t(x_i))$;

3. Compute the *importance* of f_t as $\alpha_t = 1/2 \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$;

4. Update the distribution $w_{t+1}(i) = \frac{w_t(i) e^{-\alpha_t y_i f_t(x_i)}}{Z_t}$,
   $Z_t = \sum_{i=1}^{n} w_t(i) e^{-\alpha_t y_i h_t(x_i)}$.

A Formal Description of Boosting

- given training set $(x_1, y_1), \ldots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \ldots, T$:
    - construct distribution D_t on $\{1, \ldots, m\}$, Initialize $D_t = \frac{1}{m}$
    - find weak classifier ("rule of thumb") based on weights D_t
        - $h_t : X \to \{-1, +1\}$
    - with error ϵ_t on D_t:
        - $\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i=1}^{m} D_t(i) I[y_i \neq h_t(x_i)]$
    - Compute importance of h_t as $\alpha_t = \frac{1}{2}\ln((1-\epsilon_t)/\epsilon_t)$
    - Update the data distribution $D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$, $Z_t = \sum_{i=1}^{m} D_t(i) e^{-\alpha_t y_i h_t(x_i)}$
- output final/combined classifier H_{final} (weighted mix of all $h_t's$)

AdaBoost

[with Freund]

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

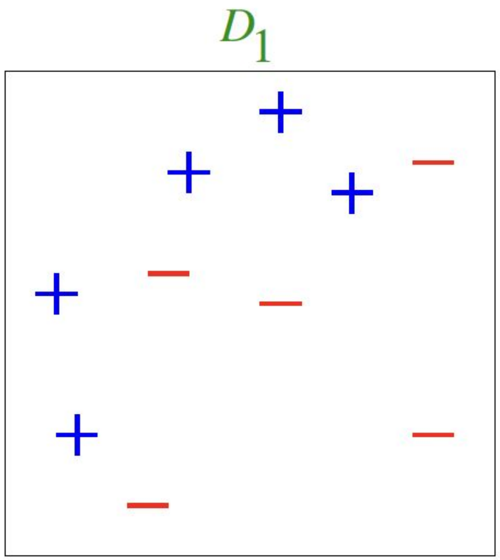
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$
$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i))$$

where Z_t = normalization factor

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) > 0$$

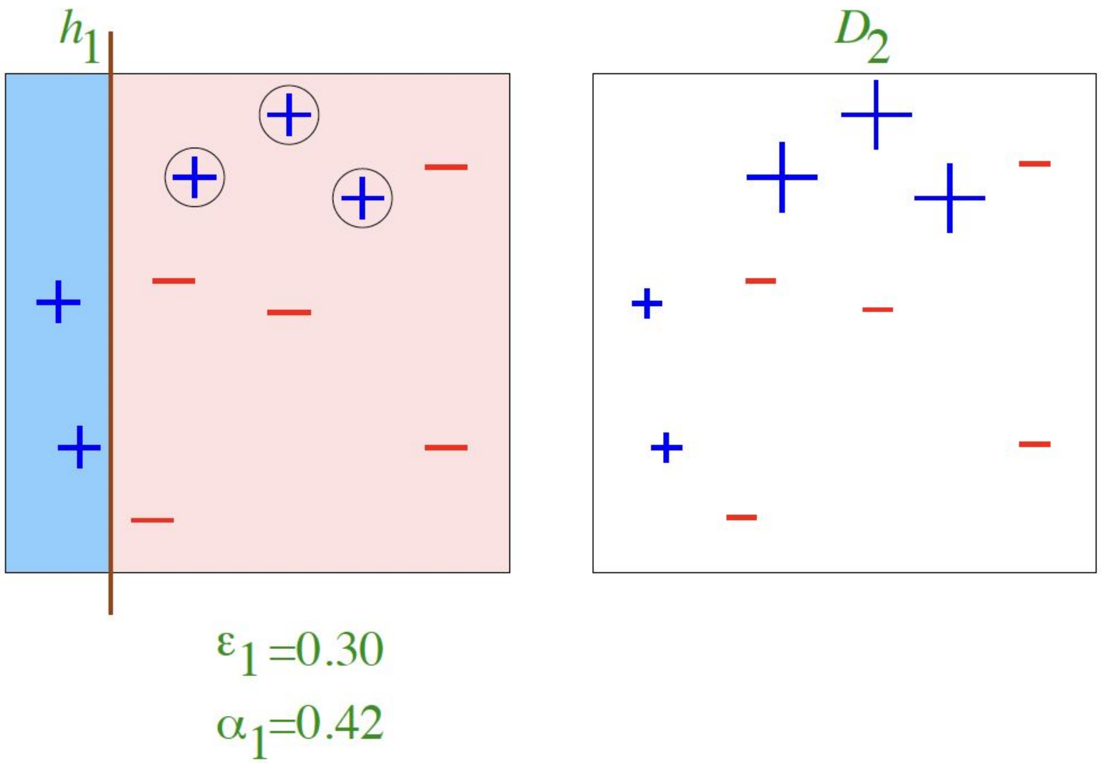
- final classifier:
  - $H_{\text{final}}(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right)$

Toy Example

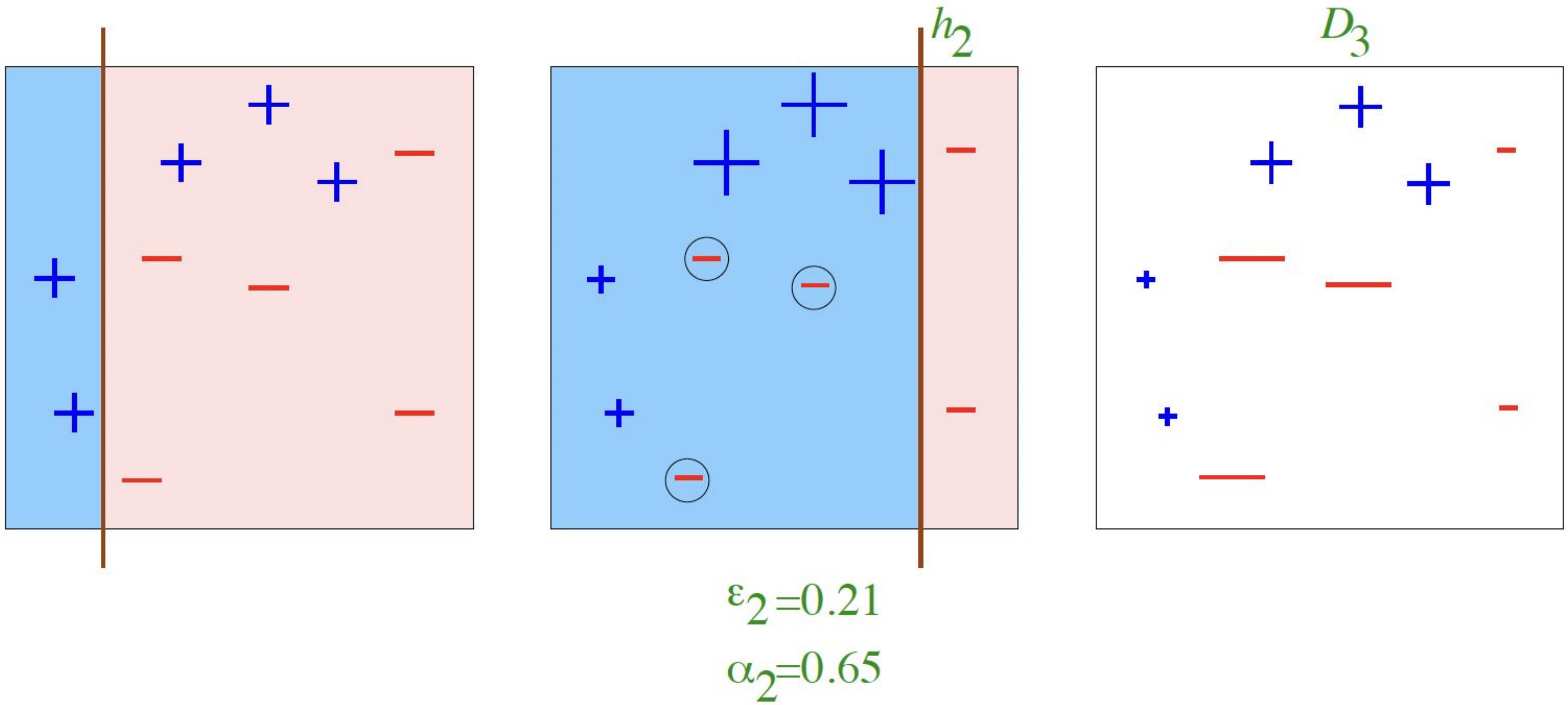


weak classifiers = vertical or horizontal half-planes

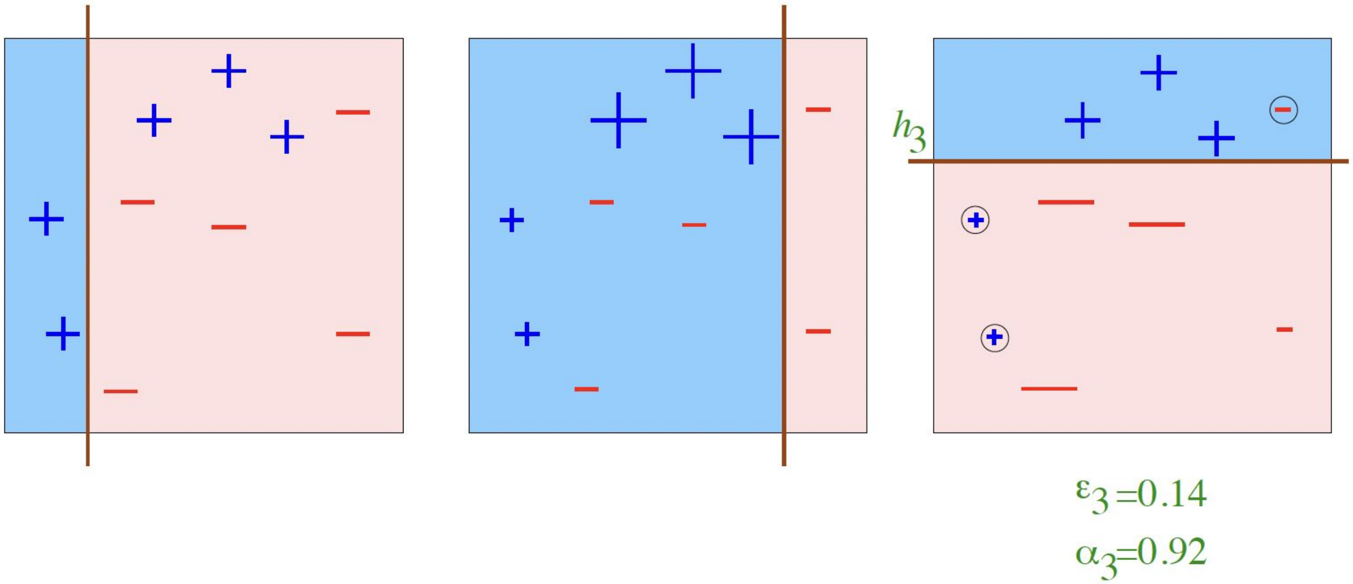
Round 1



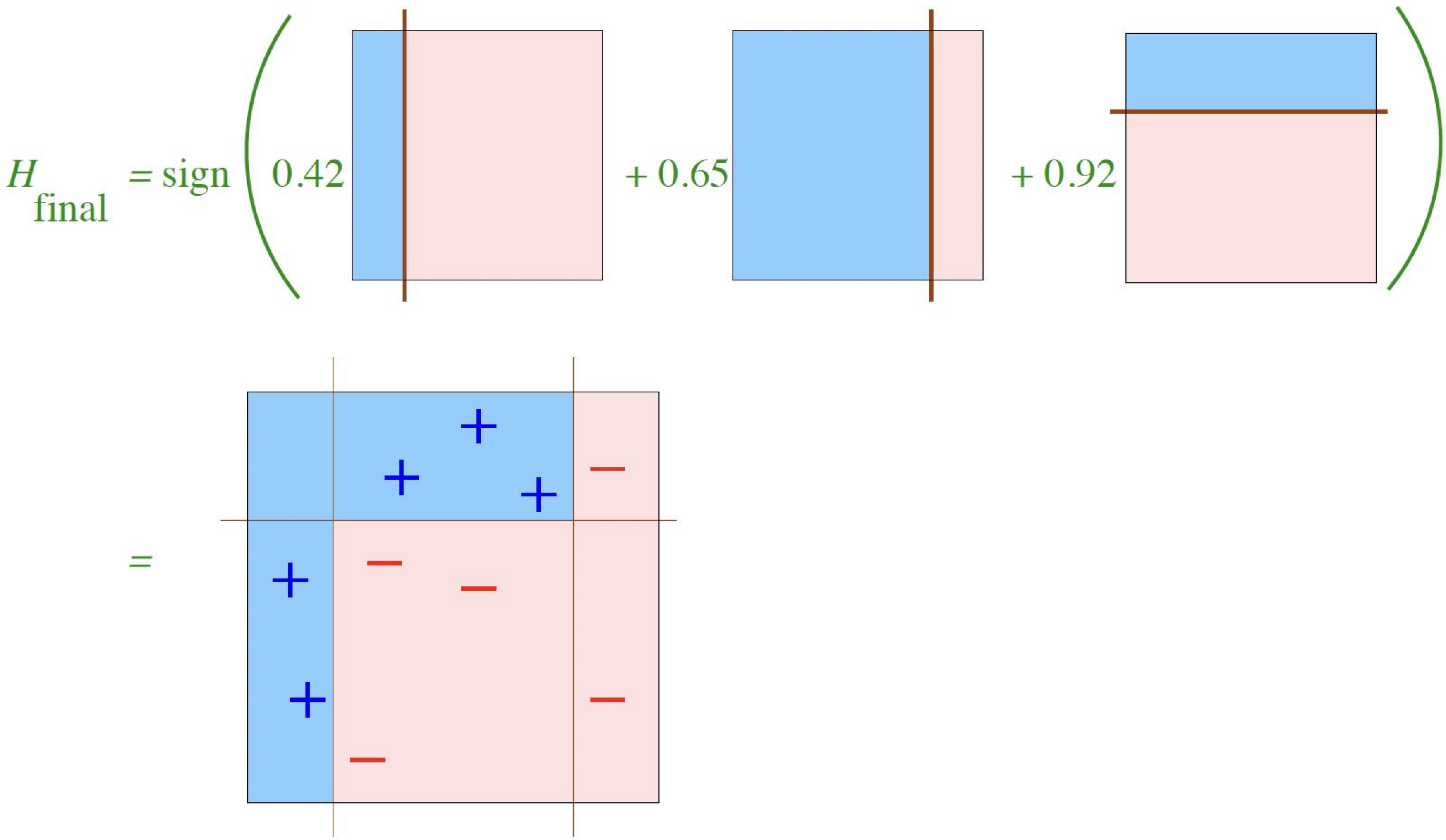
Round 2



Round 3



Final Classifier



Voted combination of classifiers

- The general problem here is to try to combine many simple "weak" classifiers into a single "strong" classifier
- We consider voted combinations of simple binary ± 1 comp

$$h_m(\mathbf{x}) = \alpha_1 \, h(\mathbf{x}; \theta_1) + \ldots + \alpha_m \, h(\mathbf{x}; \theta_m)$$

where the (non-negative) votes α_i can be used to emphasize component classifiers that are more reliable than others

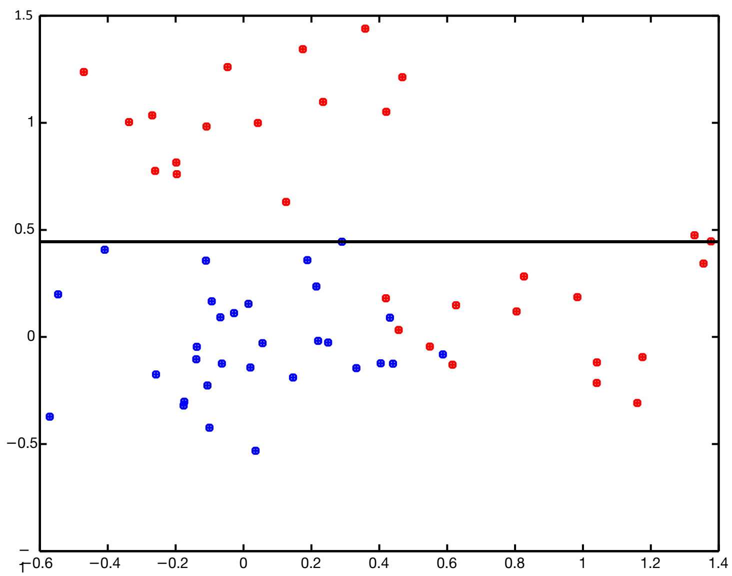
Components: decision stumps

- Consider the following simple family of component classifiers generating ±1 labels:

$$h(\mathbf{x}; \boldsymbol{\theta}) = sign(w_1 x_k - w_0)$$

where $\theta = \{k, w_1, w_0\}$. These are called *decision stumps*.

- Each decision stump pays attention to only a single component of the input vector



Voted combination cont'd

- We need to define a loss function for the combination so we can determine which new component $h(\boldsymbol{x};\ \theta)$ to add and how many votes it should receive

$$h_m(\mathbf{x}) = \alpha_1\, h(\mathbf{x}; \theta_1) + \ldots + \alpha_m\, h(\mathbf{x}; \theta_m)$$

- While there are many options for the loss function we consider here only a simple exponential loss

$$exp\{\, -y\, h_m\, (\boldsymbol{x})\, \}$$

Modularity, errors, and loss

- Consider adding the m^{th} component:

$$\sum_{i=1}^{n} \exp\{\, -y_i[h_{m-1}(\mathbf{x}_i) + \alpha_m h(\mathbf{x}_i;\theta_m)]\,\}$$

$$= \sum_{i=1}^{n} \exp\{\, -y_i h_{m-1}(\mathbf{x}_i) - y_i \alpha_m h(\mathbf{x}_i;\theta_m)\,\}$$

Modularity, errors, and loss

- Consider adding the m^{th} component:

$$\sum_{i=1}^{n} \exp\{ -y_i[h_{m-1}(\mathbf{x}_i) + \alpha_m h(\mathbf{x}_i;\theta_m)] \}$$

$$= \sum_{i=1}^{n} \exp\{ -y_i h_{m-1}(\mathbf{x}_i) - y_i\alpha_m h(\mathbf{x}_i;\theta_m) \}$$

$$= \sum_{i=1}^{n} \underbrace{\exp\{ -y_i h_{m-1}(\mathbf{x}_i)\}}_{\text{fixed at stage } m} \exp\{-y_i\alpha_m h(\mathbf{x}_i;\theta_m) \}$$

Modularity, errors, and

.

- Consider adding the m^{th} component:

$$\sum_{i=1}^{n} \exp\{\, -y_i[h_{m-1}(\mathbf{x}_i) + \alpha_m h(\mathbf{x}_i;\theta_m)]\,\}$$

$$= \sum_{i=1}^{n} \exp\{\, -y_i h_{m-1}(\mathbf{x}_i) - y_i\alpha_m h(\mathbf{x}_i;\theta_m)\,\}$$

$$= \sum_{i=1}^{n} \underbrace{\exp\{\, -y_i h_{m-1}(\mathbf{x}_i)\}}_{\text{fixed at stage } m} \exp\{-y_i\alpha_m h(\mathbf{x}_i;\theta_m)\,\}$$

$$= \sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i\alpha_m h(\mathbf{x}_i;\theta_m)\,\}$$

So at the m^{th} iteration the new component (and the votes) should optimize a weighted loss (weighted towards mistakes).

Empirical exponential loss cont'd

- To increase modularity we'd like to further decouple the optimization of $h(\mathbf{x};\theta_m)$ from the associated votes α_m
- To this end we select $h(\mathbf{x};\theta_m)$ that optimizes the rate at which the loss would decrease as a function of α_m

$$\frac{\partial}{\partial \alpha_m}\Big|_{\alpha_m=0} \sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i\alpha_m h(\mathbf{x}_i;\theta_m)\} =$$

$$\left[\sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i\alpha_m h(\mathbf{x}_i;\theta_m)\} \cdot \big(-y_i h(\mathbf{x}_i;\theta_m)\big)\right]_{\alpha_m=0}$$

$$= \left[\sum_{i=1}^{n} W_i^{(m-1)} \big(-y_i h(\mathbf{x}_i;\theta_m)\big)\right]$$

Empirical exponential loss cont'd

- We find $h(\mathbf{x}; \hat{\theta}_m)$ that minimizes

$$-\sum_{i=1}^{n} W_i^{(m-1)} y_i h(\mathbf{x}_i; \theta_m)$$

We can also normalize the weights:

$$-\sum_{i=1}^{n} \frac{W_i^{(m-1)}}{\sum_{j=1}^{n} W_j^{(m-1)}} y_i h(\mathbf{x}_i; \theta_m)$$

$$= \; -\sum_{i=1}^{n} \tilde{W}_i^{(m-1)} y_i h(\mathbf{x}_i; \theta_m)$$

so that $\sum_{i=1}^{n} \tilde{W}_i^{(m-1)} = 1$.

Selecting a new component: summary

- We find $h(\mathbf{x}; \hat{\theta}_m)$ that minimizes

$$-\sum_{i=1}^{n} \tilde{W}_i^{(m-1)} y_i h(\mathbf{x}_i; \theta_m)$$

  where $\sum_{i=1}^{n} \tilde{W}_i^{(m-1)} = 1$.

- α_m is subsequently chosen to minimize

$$\sum_{i=1}^{n} \tilde{W}_i^{(m-1)} \exp\{-y_i \alpha_m h(\mathbf{x}_i; \hat{\theta}_m)\}$$

The AdaBoost algorithm

0) Set $\tilde{W}_i^{(0)} = 1/n$ for $i = 1, \ldots, n$

1) At the m^{th} iteration we find (any) classifier $h(\mathbf{x}; \hat{\theta}_m)$ for which the *weighted classification error* ϵ_m

$$\epsilon_m = 0.5 - \frac{1}{2}\left(\sum_{i=1}^{n} \tilde{W}_i^{(m-1)} y_i h(\mathbf{x}_i; \hat{\theta}_m)\right)$$

is better than chance.

2) The new component is assigned votes based on its error:

$$\hat{\alpha}_m = 0.5 \log(\, (1 - \epsilon_m)/\epsilon_m \,)$$

3) The weights are updated according to (Z_m is chosen so that the new weights $\tilde{W}_i^{(m)}$ sum to one):

$$\tilde{W}_i^{(m)} = \frac{1}{Z_m} \cdot \tilde{W}_i^{(m-1)} \cdot \exp\{\, -y_i \hat{\alpha}_m h(\mathbf{x}_i; \hat{\theta}_m) \,\}$$