

# DS 4400

## Machine Learning and Data Mining I Spring 2021

Alina Oprea

Associate Professor

Khoury College of Computer Science

Northeastern University

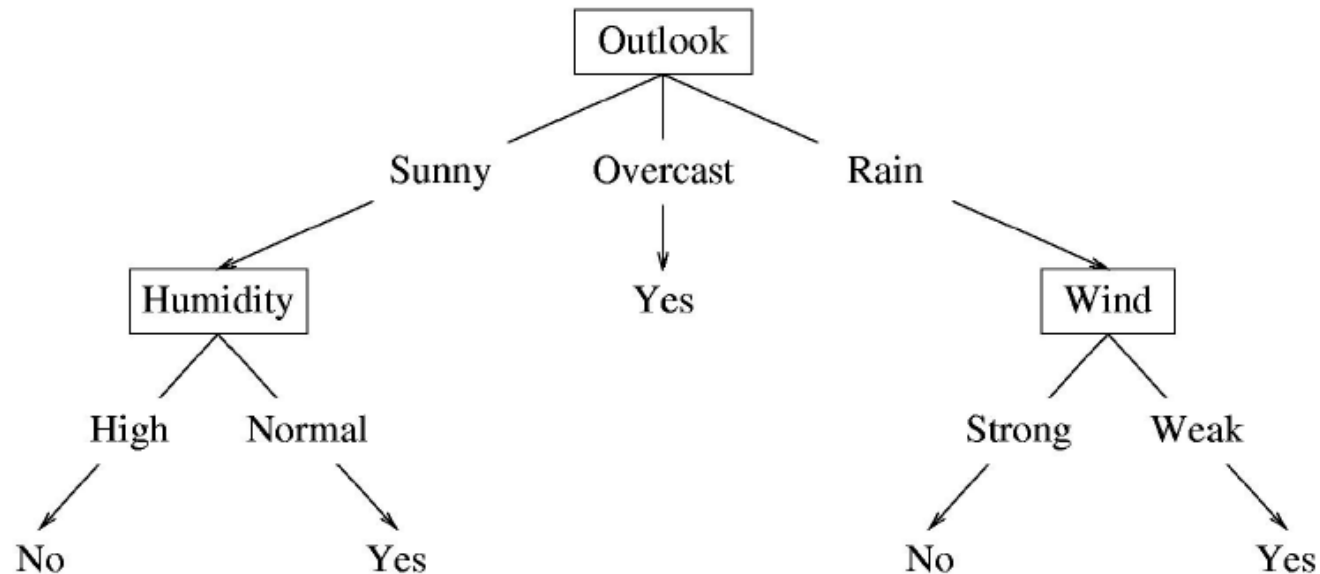
March 9 2021

# Outline

- Decision Trees
  - Regression trees
- Ensemble models
  - Majority vote over multiple models
- Bagging
  - Bootstrap samples
  - Random forest
- Boosting
  - AdaBoost

# Decision Tree

- A possible decision tree for the data:



- Each internal node: test one attribute  $X_i$
- Each branch from a node: selects one value for  $X_i$
- Each leaf node: predict  $Y$  (or  $p(Y \mid x \in \text{leaf})$  )

# Learning Decision Trees

- Start from empty decision tree GREEDY.
- Split on **next best attribute (feature)**
  - Use, for example, information gain to select attribute:

$$\arg \max_i IG(X_i) = \arg \max_i H(Y) - H(Y | X_i)$$

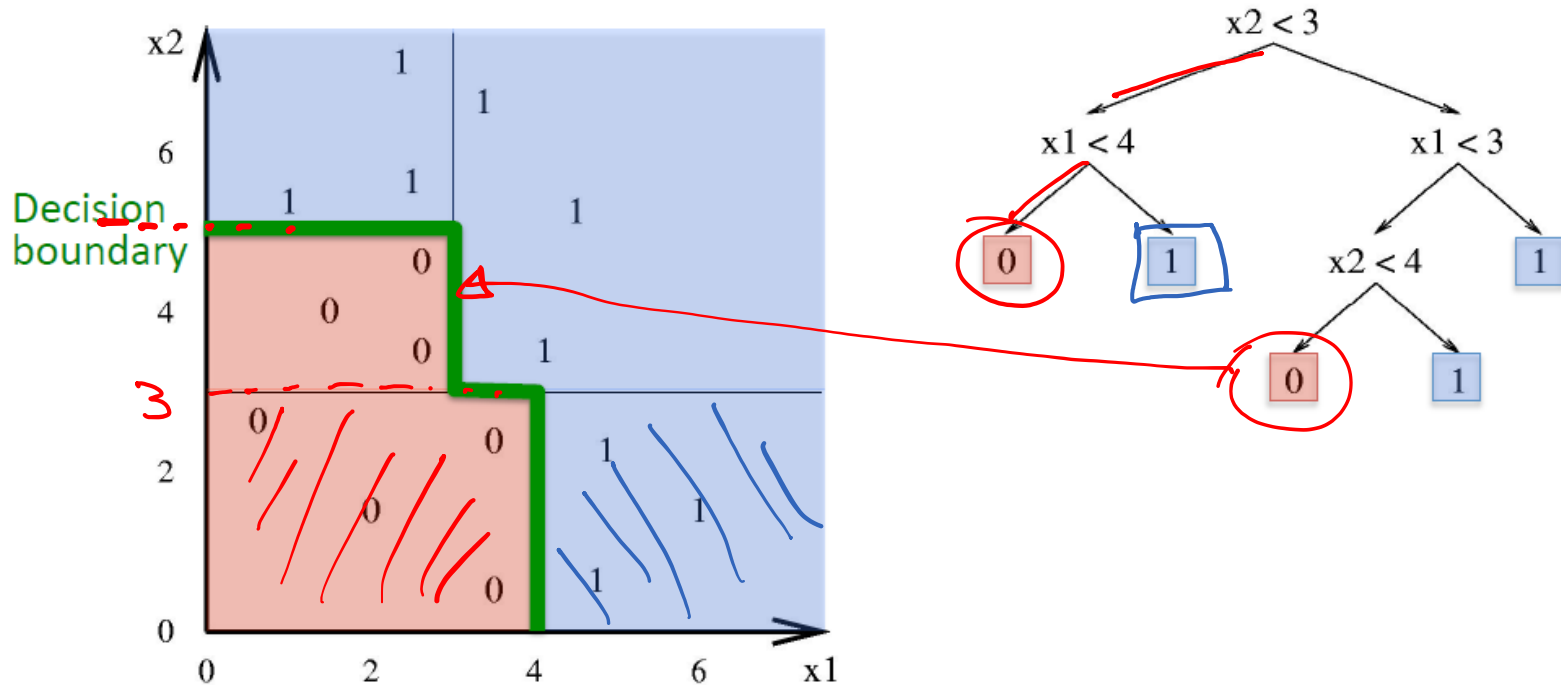
- Recurse

STOPPING CONDITIONS

ID3 algorithm uses Information Gain  
Information Gain reduces uncertainty on Y

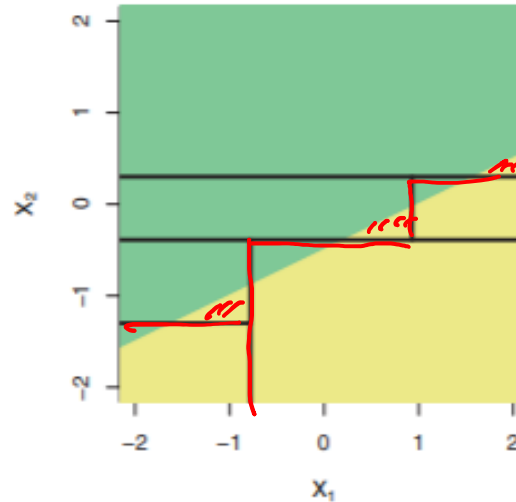
# Decision Boundary

- Decision trees divide the feature space into axis-parallel (hyper-)rectangles
- Each rectangular region is labeled with one label

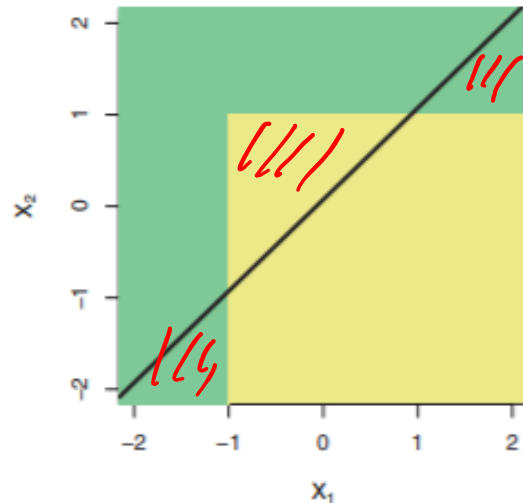


# Decision Trees vs Linear Models

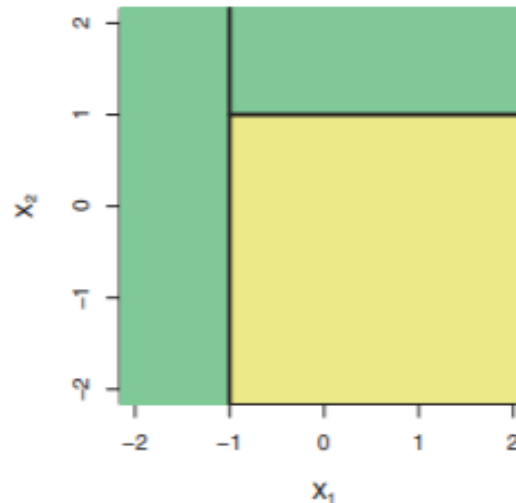
LOGISTIC  
REG  
LDA



DATA  
LINEARLY  
SEPARABLE



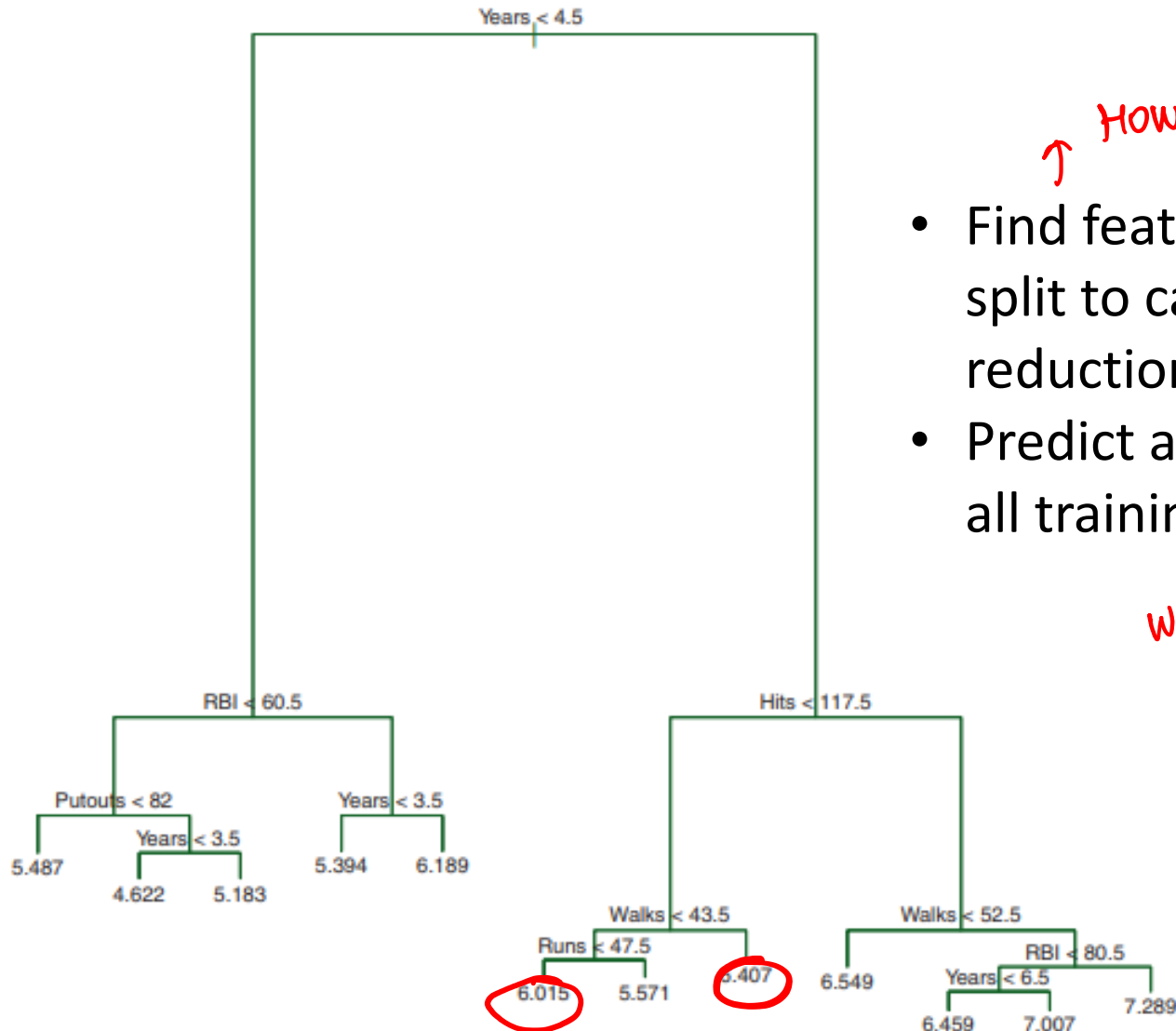
Linear model



NON-LINEAR  
MODEL

Decision tree

# Regression Trees



↑ HOW TO SPLIT

- Find feature and value to split to cause the maximum reduction in MSE
- Predict average response of all training data at each leaf

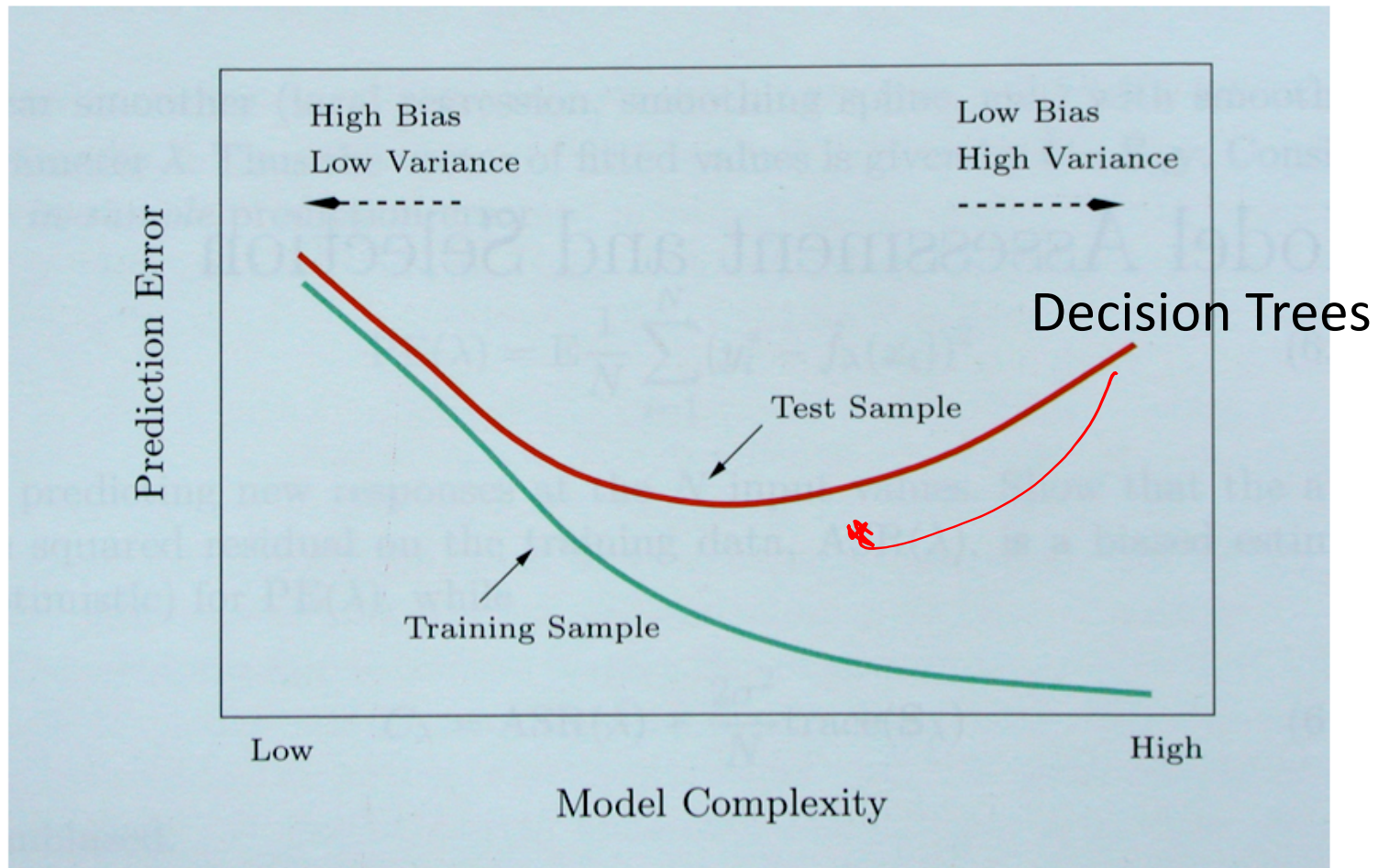
↓ WHAT TO PREDICT

# Summary Decision Trees

- Representation: decision trees
- Bias: prefer small decision trees
- Search algorithm: greedy
- Heuristic function: information gain or information content or others
- Overfitting / pruning



# Bias/Variance Tradeoff



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

How to reduce variance of single decision tree?

# Ensemble Learning

Consider a set of classifiers  $h_1, \dots, h_L$

**Idea:** construct a classifier  $H(\mathbf{x})$  that combines the individual decisions of  $h_1, \dots, h_L$

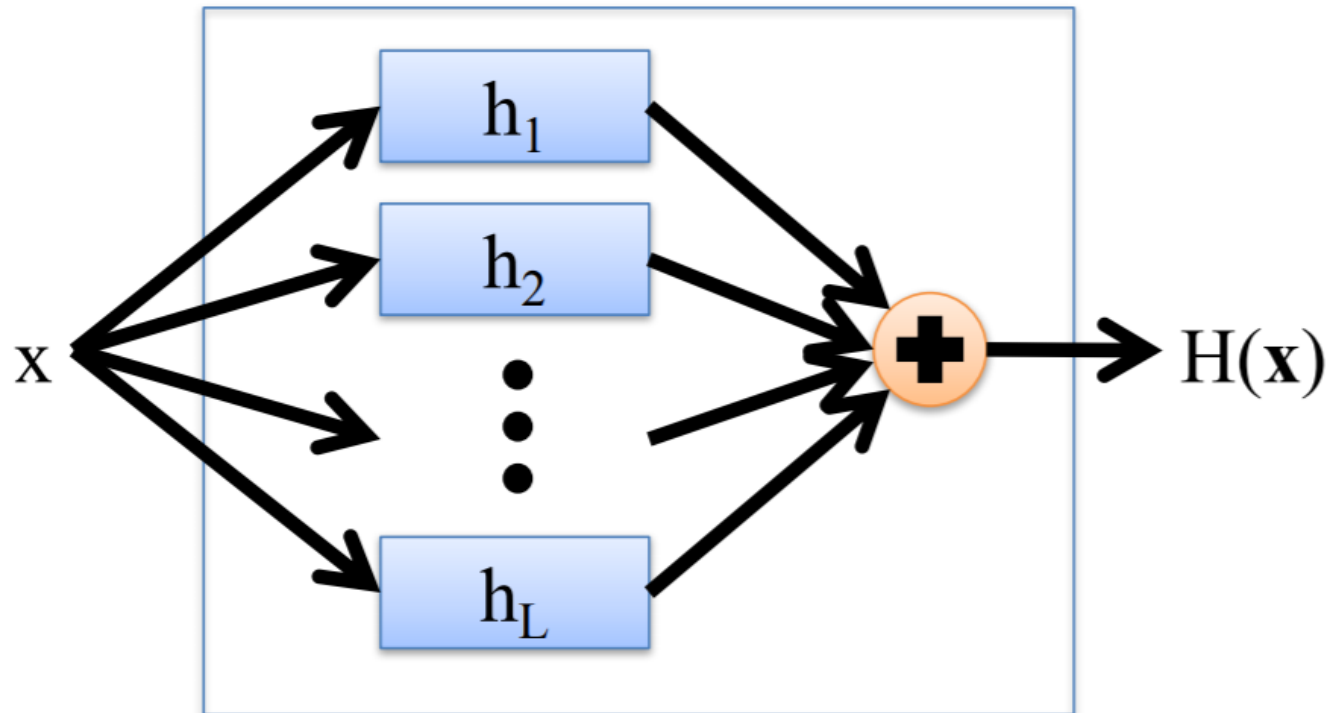
- e.g., could have the member classifiers vote, or
- e.g., could use different members for different regions of the instance space

Successful ensembles require **diversity**

- Classifiers should make different mistakes
- Can have different types of base learners

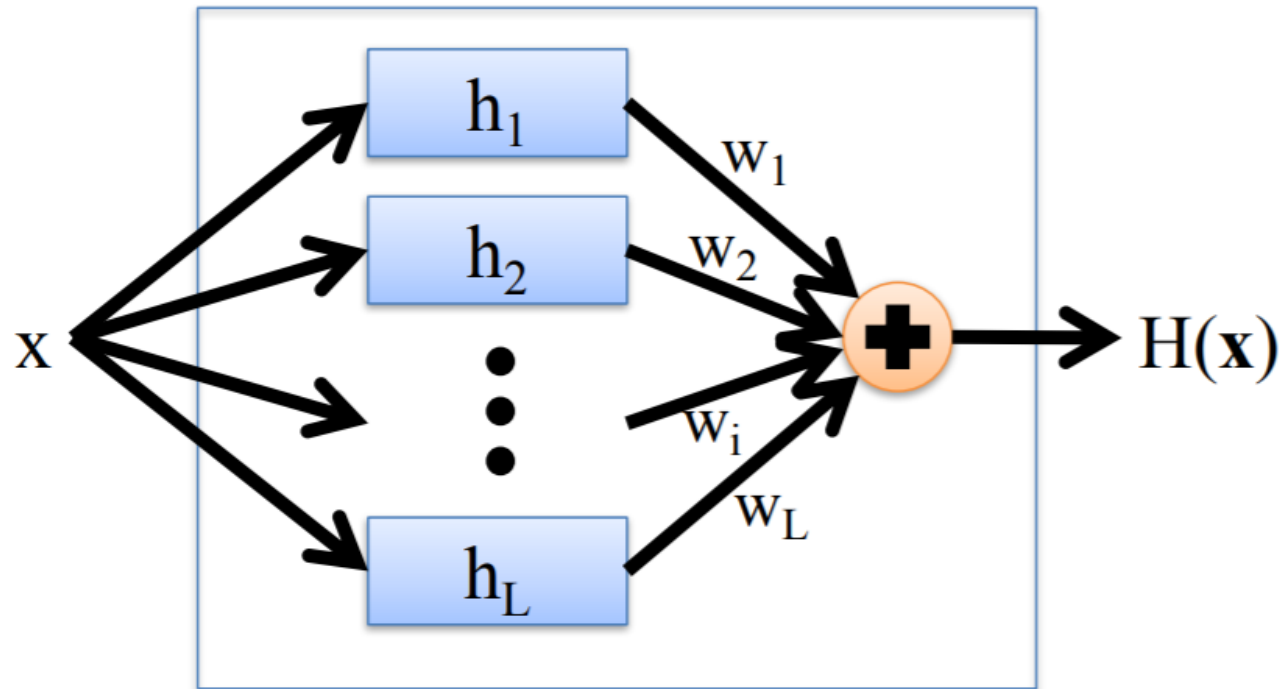
*- VARY TRAINING (SUBSET)  
- VARY MODELS / ALG FOR TRAINING  
- VARY FEATURES*

# Combining Classifiers: Averaging



- Final hypothesis is a simple vote of the members

# Combining Classifiers: Weighted Averaging



- Coefficients of individual members are trained using a validation set

# Practical Applications

**Goal:** predict how a user will rate a movie

- Based on the user's ratings for other movies
- and other peoples' ratings
- with no other information about the movies



This application is called “collaborative filtering”

**Netflix Prize:** \$1M to the first team to do 10% better than Netflix' system (2007-2009)

**Winner:** BellKor's Pragmatic Chaos – an ensemble of more than 800 rating systems

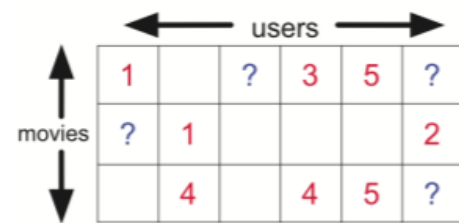
# Netflix Prize

Machine learning competition with a \$1 million prize

## Leaderboard

Display top 20 leaders.

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





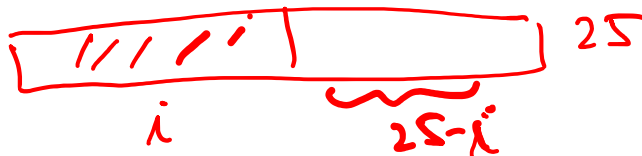
# Reduce error

- Suppose there are 25 base classifiers
- Each classifier has error rate,  $\epsilon = 0.35$
- Assume independence among classifiers
- Probability that the ensemble classifier makes a wrong prediction:

MAJORITY PRED.

$\Rightarrow$  13 CLASSIFIERS MAKE AN ERROR  
 EXACTLY  $i$  CLAS. ARE WRONG,  $i \geq 13$   

$$\sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} = 0.06$$



# Reduce Variance

$X_1, \dots, X_N$  INDEPENDENT RV

$\mu \in [0,1]$  ;  $X_i \in [0,1]$  ;  $E[X_i] = \mu$

$$\bar{X} = \frac{X_1 + \dots + X_N}{N}$$

$$\text{Var}[X_i] = \sigma^2$$

$$E[\bar{X}] = \frac{E[X_1] + \dots + E[X_N]}{N} = \mu$$


$$\begin{aligned} \text{Var}[\bar{X}] &= \text{Var}\left[\frac{X_1 + \dots + X_N}{N}\right] = \frac{1}{N^2} \sum_{i=1}^N \text{Var}(X_i) \\ &= \frac{\sigma^2}{N} \end{aligned}$$

$$\text{Var}(X+Y) = \text{Var}(X) + \text{Var}(Y)$$

$$\text{Var}(a \cdot X) = a^2 \text{Var}(X)$$



# How to Achieve Diversity

- 
- Avoid overfitting
    - Vary the training data
  - Features are noisy
    - Vary the set of features

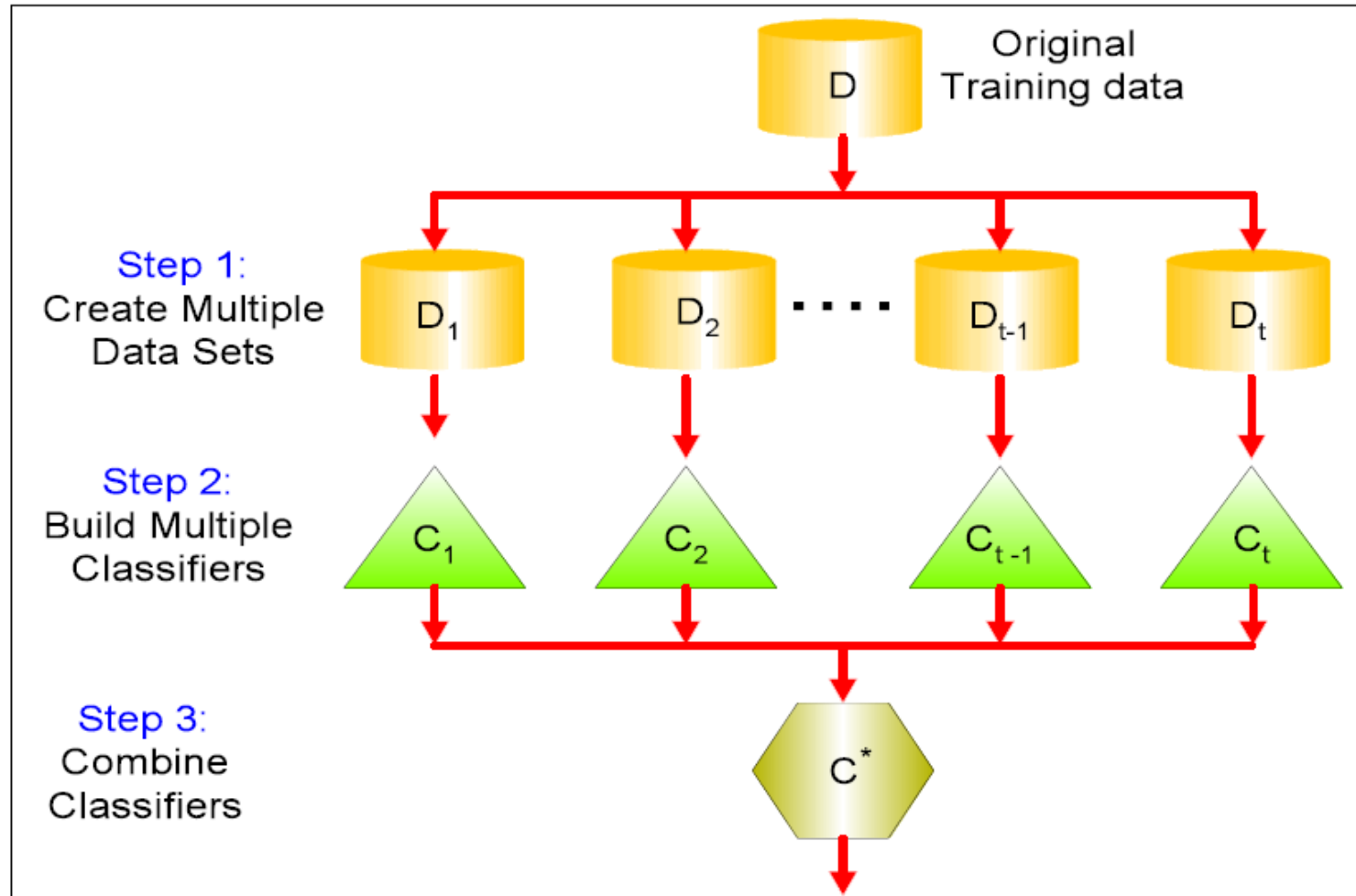
Two main ensemble learning methods

- **Bagging** (e.g., Random Forests)
- **Boosting** (e.g., AdaBoost)

# Bagging

- 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  $D_1 \dots D_k$ .
  - Train distinct classifier on each  $D_i$ .
  - Classify new instance by majority vote / average.

# General Idea



Majority Votes

# Example of Bagging

$$\left(1 + \frac{1}{n}\right)^n \rightarrow e$$

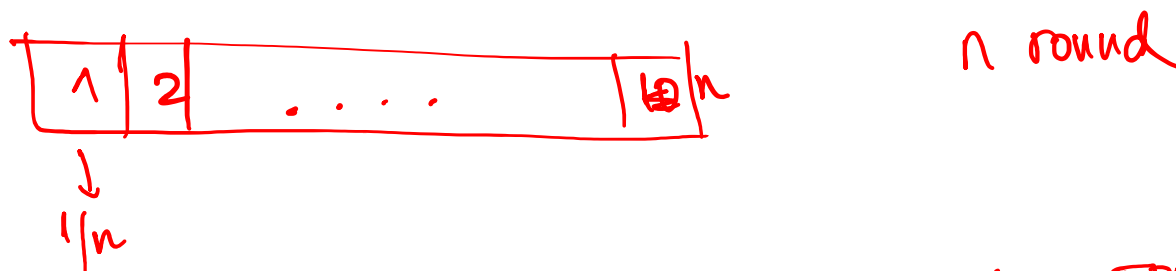
$$\left(1 - \frac{1}{n}\right)^n \rightarrow \frac{1}{e}$$

- Sampling with replacement

Training Data

Data ID	1	2	3	4	5	6	7	8	9	10
Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

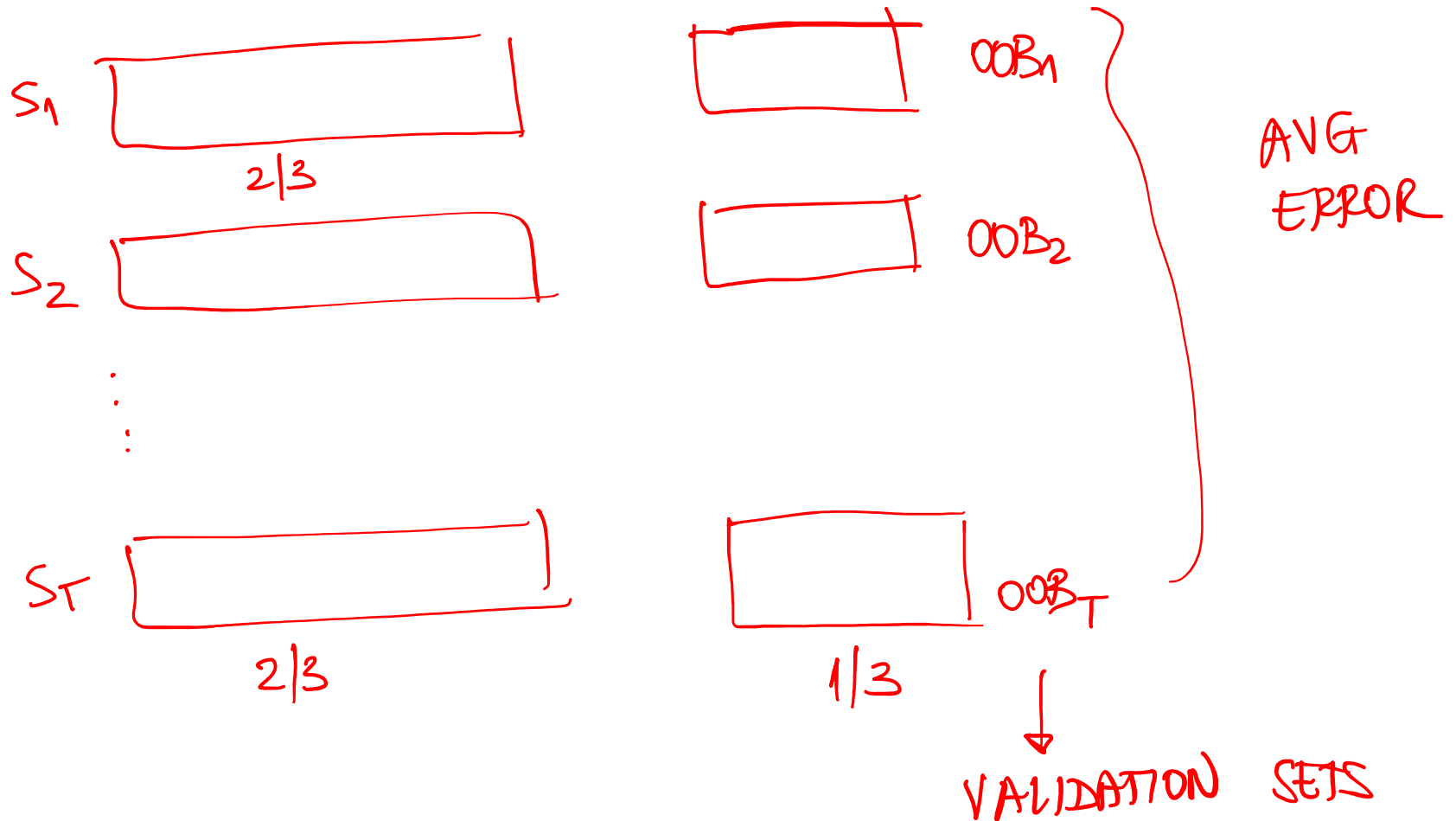
- Sample each training point with probability  $1/n$
- Out-Of-Bag (OOB) observation:** point not in sample



PROB. TRAINING EXAMPLE NOT IN BOOTSTRAP SAMPLE

$$\left(1 - \frac{1}{n}\right)^n \approx \frac{1}{e} \approx \frac{1}{3}$$

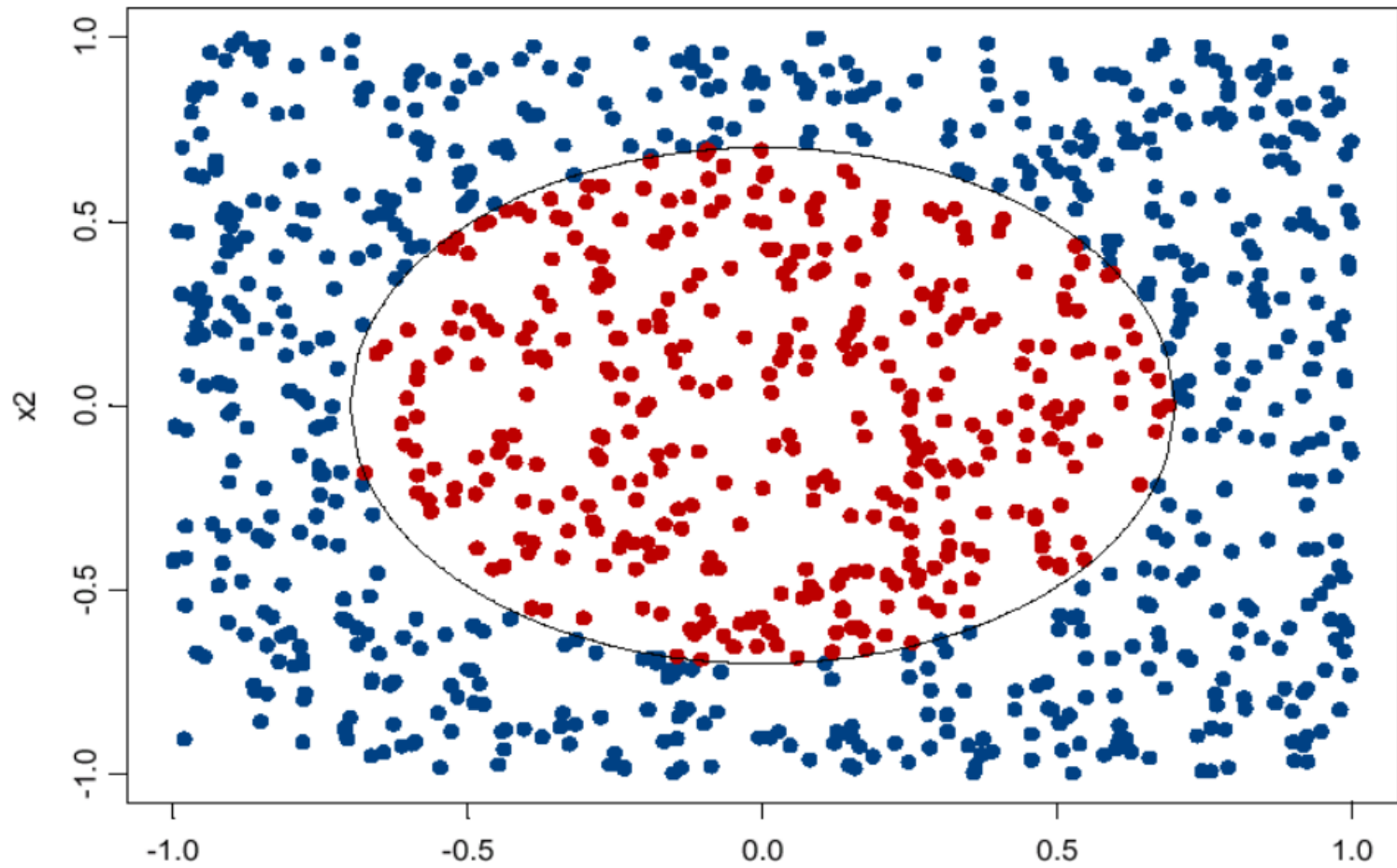
# Bootstrap Samples



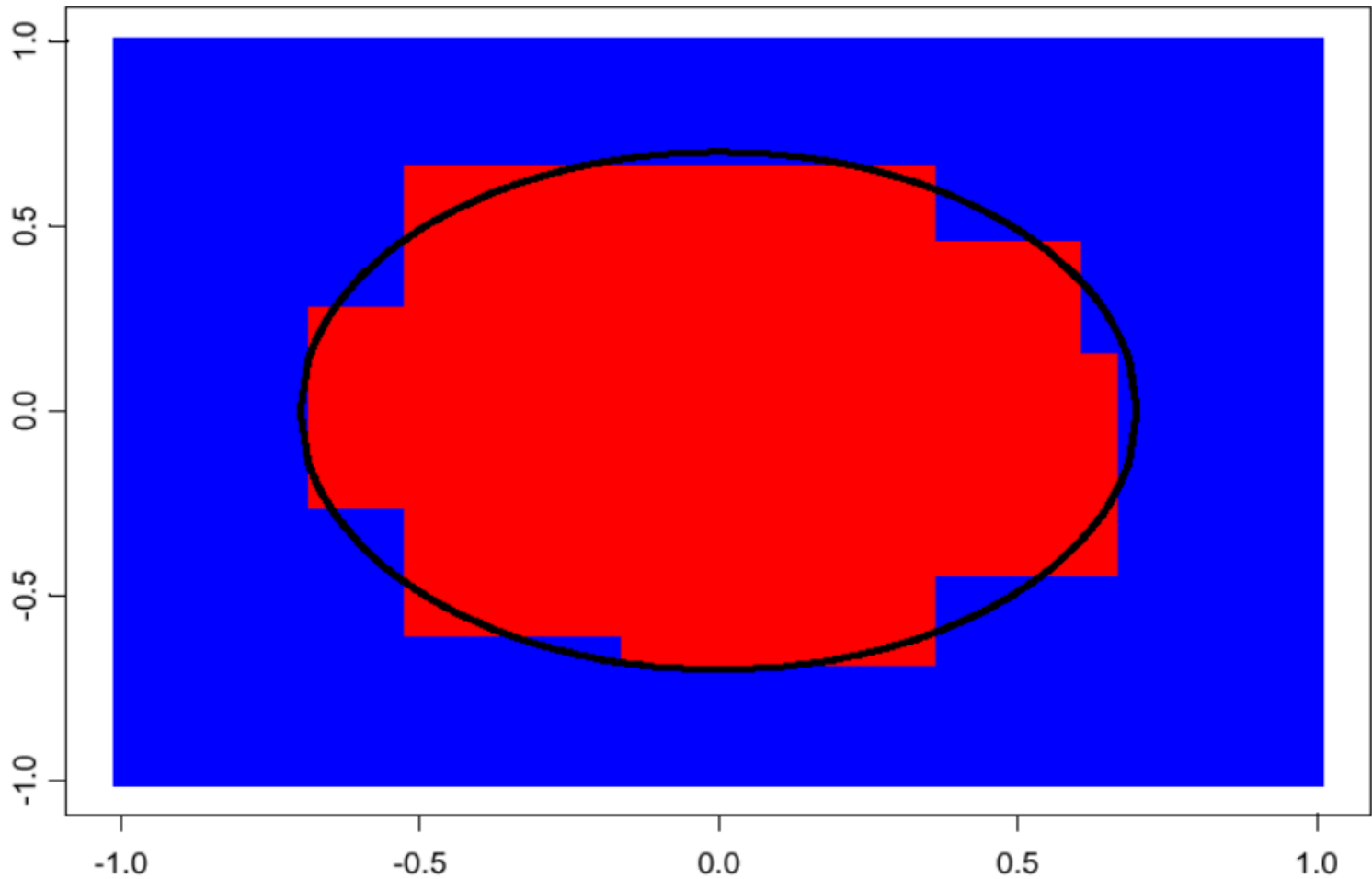
# Bagging

- Can be applied to multiple classification models
- Very successful for decision trees
  - Decision trees have high variance
  - Don't prune the individual trees, but grow trees to full extent
  - Precision accuracy of decision trees improved substantially
- OOB average error used instead of Cross Validation

# Example Distribution

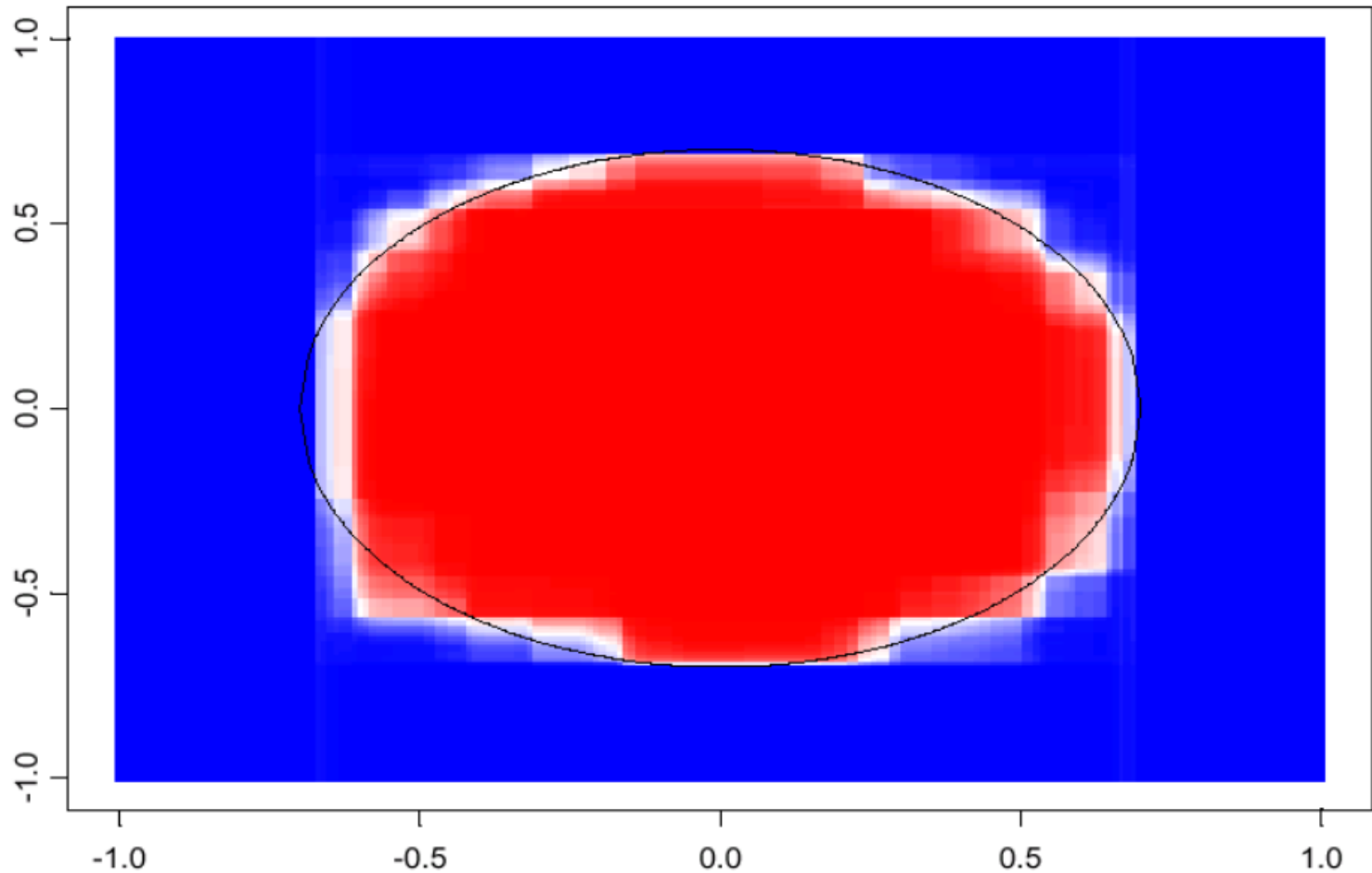


# Decision Tree Decision Boundary





# 100 Bagged Trees



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

# Random Forests

- Ensemble method specifically designed for decision tree classifiers
- Introduce two sources of randomness: “Bagging” and “Random input vectors”
  - **Bagging method**: each tree is grown using a bootstrap sample of training data
  - **Random vector method**: **At each node**, best split is chosen from a random sample of  $m$  attributes instead of all attributes

# Random Forests

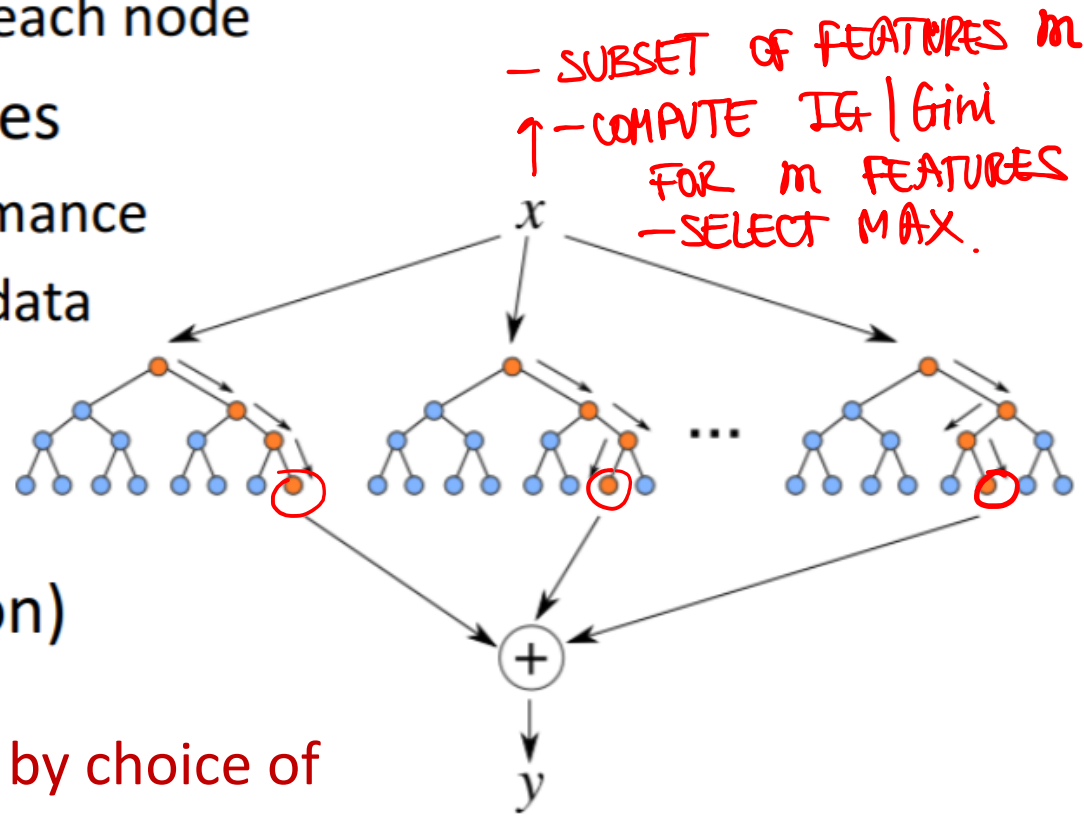
## TRAIN

- Construct decision trees on bootstrap replicas
  - Restrict the node decisions to a small subset of features picked randomly for each node
- Do not prune the trees
  - Estimate tree performance on out-of-bootstrap data

## TEST

- Average the output of all trees (or choose mode decision)

Trees are de-correlated by choice of random subset of features



# Random Forest Algorithm

TRAIN

1. For  $b = 1$  to  $B$ :  $\rightarrow$  # TREES

- (a) Draw a **bootstrap sample**  $\mathbf{Z}^*$  of size  $N$  from the training data.
- (b) Grow a random-forest tree  $T_b$  to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size  $n_{min}$  is reached.
  - i. Select  **$m$  variables at random** from the  $d$  variables.
  - ii. Pick the best variable/split-point among the  $m$ .
  - iii. Split the node into two daughter nodes.

2. Output the ensemble of trees  $\{T_b\}_1^B$ .

TEST

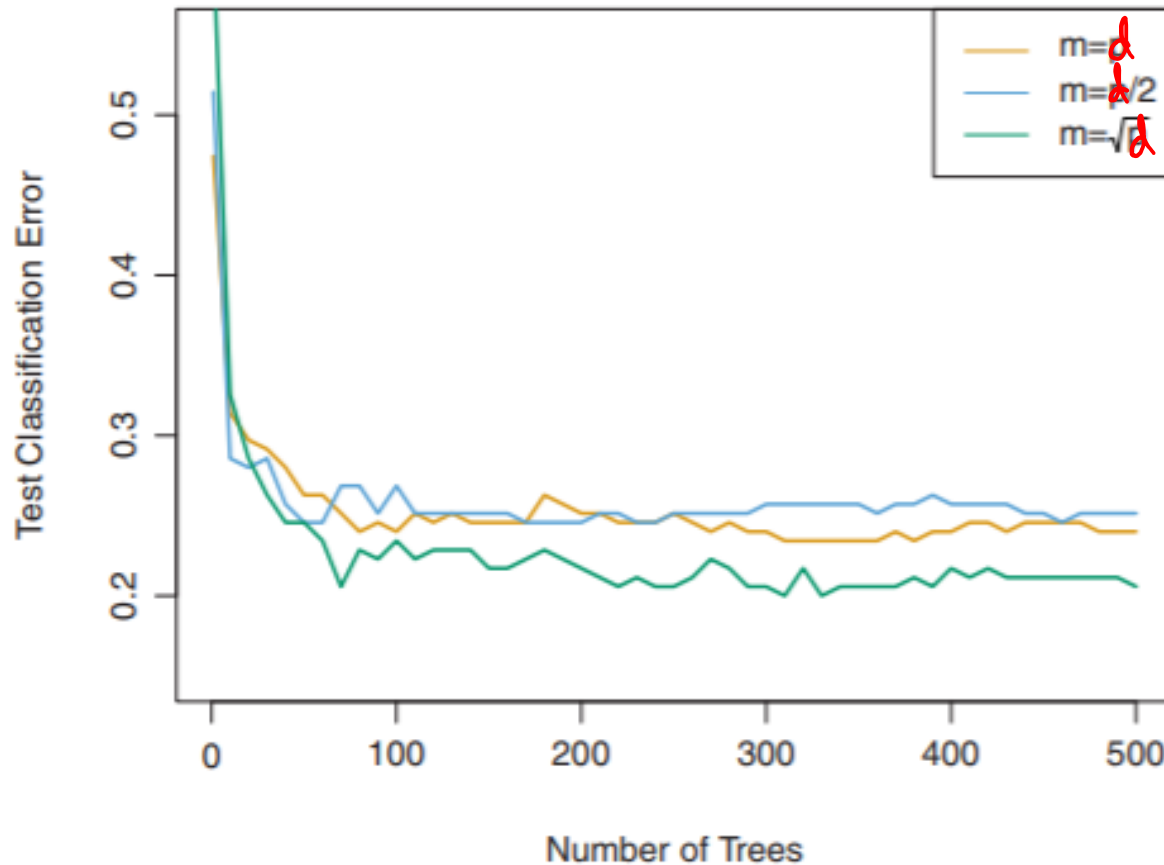
To make a prediction at a new point  $x$ :

*Regression:*  $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B} \sum_{b=1}^B T_b(x)$ .

*Classification:* Let  $\hat{C}_b(x)$  be the class prediction of the  $b$ th random-forest tree. Then  $\hat{C}_{\text{rf}}^B(x) = \text{majority vote } \{\hat{C}_b(x)\}_1^B$ .

If  $m=p$ , this is equivalent to **BAGGING**  
If  $m=d$ , this is equivalent to **BOOSTING**  
with Decision Trees as base learner

# Effect of Number of Predictors

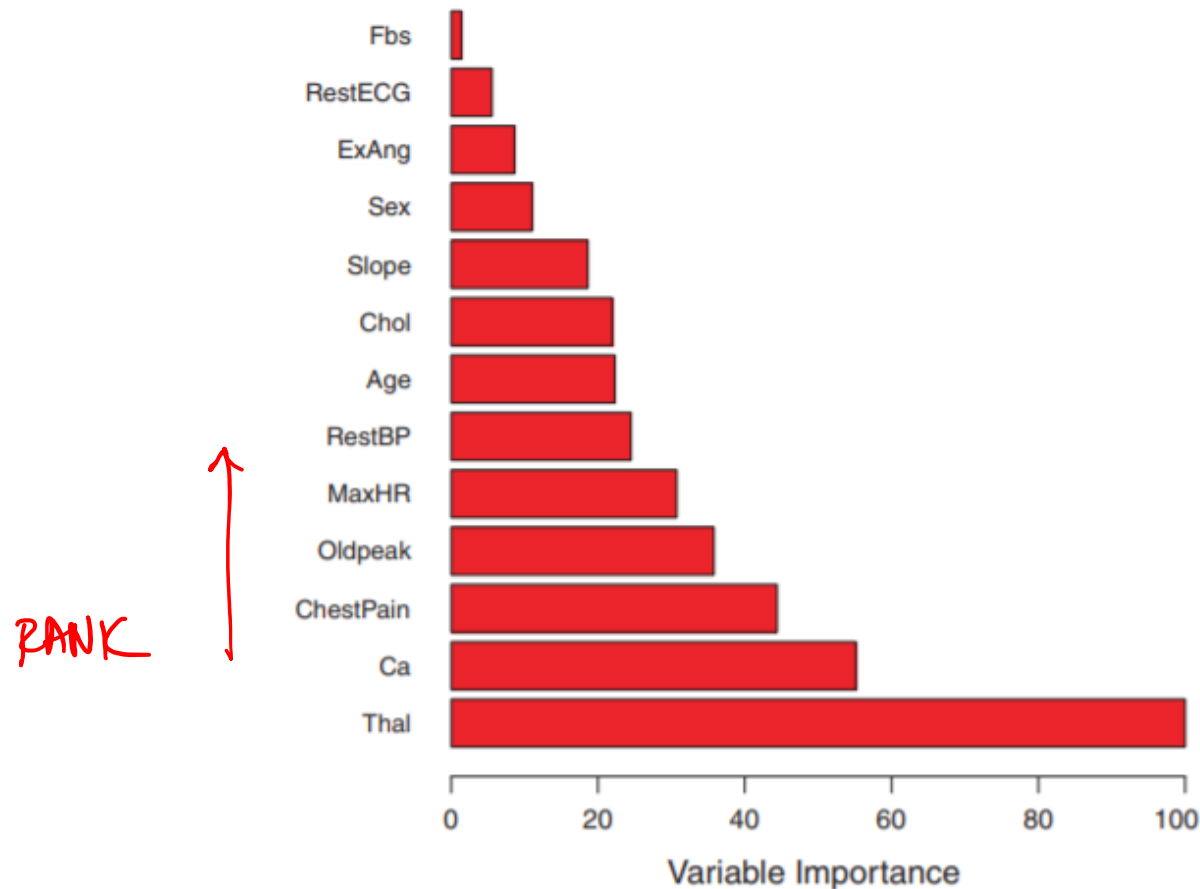


- $d$  = total number of predictors;  $m$  = predictors chosen in each split
- Random Forests uses  $m = \sqrt{d}$

# Variable Importance

- Ensemble of trees loses somewhat interpretability of decision trees
- Which variables contribute mostly to prediction?
- Random Forests computes a Variable Importance metric per feature
  - For each tree in the ensemble, consider the split by the particular feature
  - How much information gain / Gini index decreases after the split
  - Average over all trees

# Variable Importance Plots



**FIGURE 8.9.** A variable importance plot for the **Heart** data. Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.

IG

# Variable Importance

- Ensembles of trees loose in interpretability
  - Variable importance helps with determining important features
- Can be used as a filter method for feature selection
  - Train Random Forest model
  - Compute variable importance
  - Select top k features by highest important
  - Train other models with the k features



# How to Achieve Diversity

- Avoid overfitting
  - Vary the training data
- Features are noisy
  - Vary the set of features

Two main ensemble learning methods

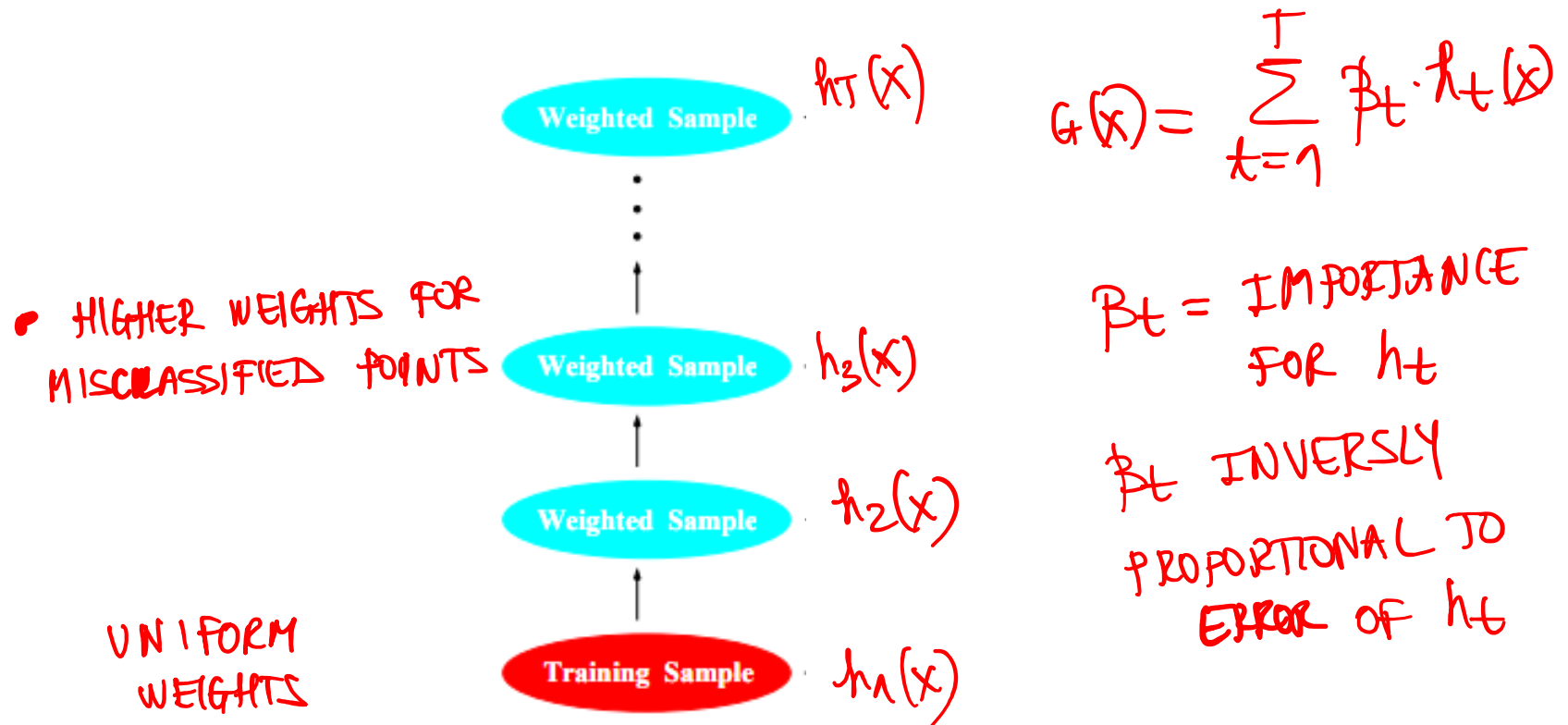
- **Bagging** (e.g., Random Forests)
- **Boosting** (e.g., AdaBoost)

# AdaBoost

- A meta-learning algorithm with great theoretical and empirical performance
- Turns a base learner (i.e., a “weak hypothesis”) into a high performance classifier
- Creates an ensemble of weak hypotheses by repeatedly emphasizing mispredicted instances

Adaptive Boosting  
Freund and Schapire 1997

# Overview of AdaBoost



**FIGURE 10.1.** Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.

# Boosting [Shapire '89]

- **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 weak hypothesis –  $h_t$
  - A strength for this hypothesis –  $\beta_t$
- Final classifier:  $H(x) = \text{sign}(\sum \beta_t h_t(x))$

$$h_t \in \{-1, 1\}$$

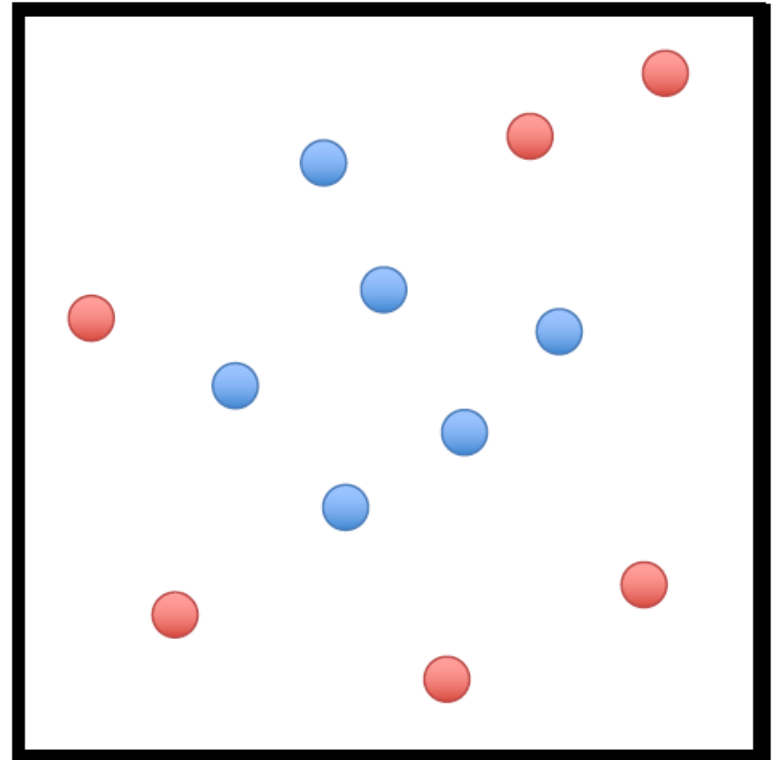
## Convergence bounds with minimal assumptions on weak learner

If each weak learner  $h_t$  is slightly better than random guessing ( $\epsilon_t < 0.5$ ), then training error of AdaBoost decays exponentially fast in number of rounds  $T$ .

# AdaBoost

- 1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$
- 2: **for**  $t = 1, \dots, T$
- 3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$
- 4:   Compute the weighted training error of  $h_t$
- 5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)$
- 6:   Update all instance weights:  
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

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



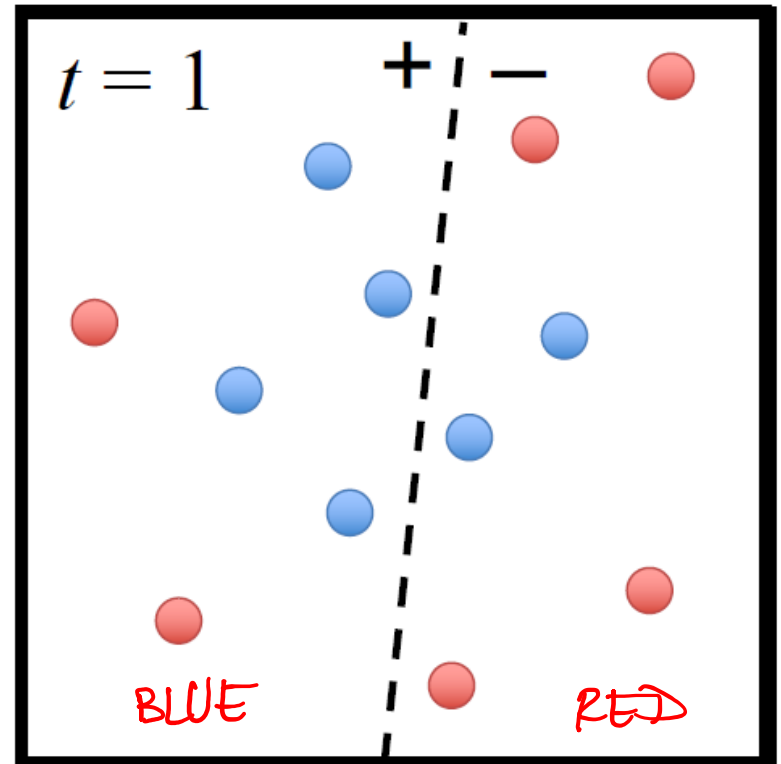
- Size of point represents the instance's weight

# AdaBoost

$T = \# \text{ MODELS}$

- 1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$
- 2: **for**  $t = 1, \dots, T$
- 3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$
- 4:   Compute the weighted training error of  $h_t$
- 5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)$
- 6:   Update all instance weights:  
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

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



$$\epsilon_1 = \frac{4}{12} = \frac{1}{3}$$

$$\beta_1 = \frac{1}{2} \ln \left( \frac{1 - \epsilon_1}{\epsilon_1} \right)$$

# Acknowledgements

- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
  - Andrew Moore
- Thanks!