

# UVA CS 6316: Machine Learning

## Lecture 18b: Random Forest / Boosting

Dr. Yanjun Qi

University of Virginia  
Department of Computer Science

# Course Content Plan →

## Six major sections of this course

- Regression (supervised)
- Classification (supervised)
- Unsupervised models
- Learning theory
- Graphical models
- Reinforcement Learning

Y is a continuous

Y is a discrete

NO Y

About  $f()$

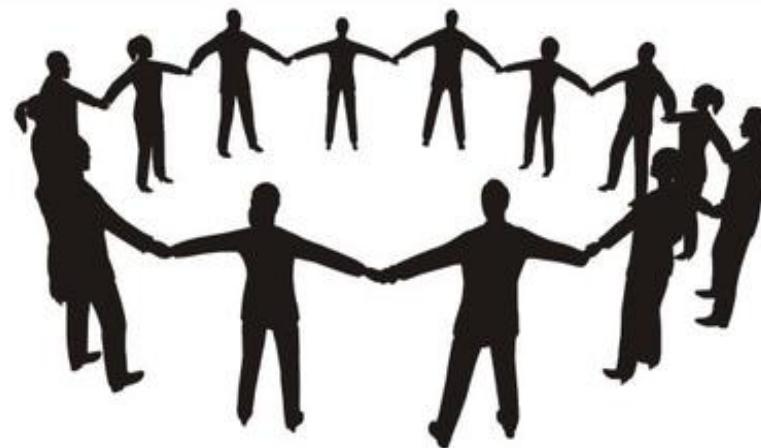
About interactions among  $X_1, \dots, X_p$

Learn program to Interact with its environment

# Three major sections for classification

- We can divide the large variety of classification approaches into **roughly three major types**
  1. Discriminative
    - directly estimate a decision rule/boundary
    - e.g., ~~support vector machine, decision tree, logistic regression,~~  
~~e.g. neural networks (NN), deep NN~~
  2. Generative:
    - build a generative statistical model
    - e.g., ~~Bayesian networks, Naïve Bayes classifier~~
  3. ~~Instance based classifiers~~
    - Use observation directly (no models)
    - e.g. K nearest neighbors

# Today: Ensemble



# Today: Ensemble

- Framework of Ensemble:

- 1. Get a set of classifiers  $f_1(x), f_2(x), f_3(x), \dots, f_B(x)$

They should be diverse.

- 2. Aggregate the classifiers (*properly*)

blend

# Today: Ensemble



- Bagging / reduce Variance
- Bagged Decision Tree
- Random forests:
- Boosting / reduce bias , reduce Variance
  - Adaboost
  - Xgboost / Gradient Boosting
- Stacking

# Bagging

- Bagging or *bootstrap aggregation*
  - a technique for reducing the variance of an estimated prediction function.
- For instance, for classification, a *committee of decision trees*
  - Each tree casts a vote for the predicted class.

# Bootstrap

The basic idea:

randomly draw datasets *with replacement* (i.e. allows duplicates) from the training data, each samples *the same size as the original training set*

$f_1, f_2 \dots, f_B \quad \#|Tr|$   
① change training set  $\{s_1, s_2, \dots, s_B\}$

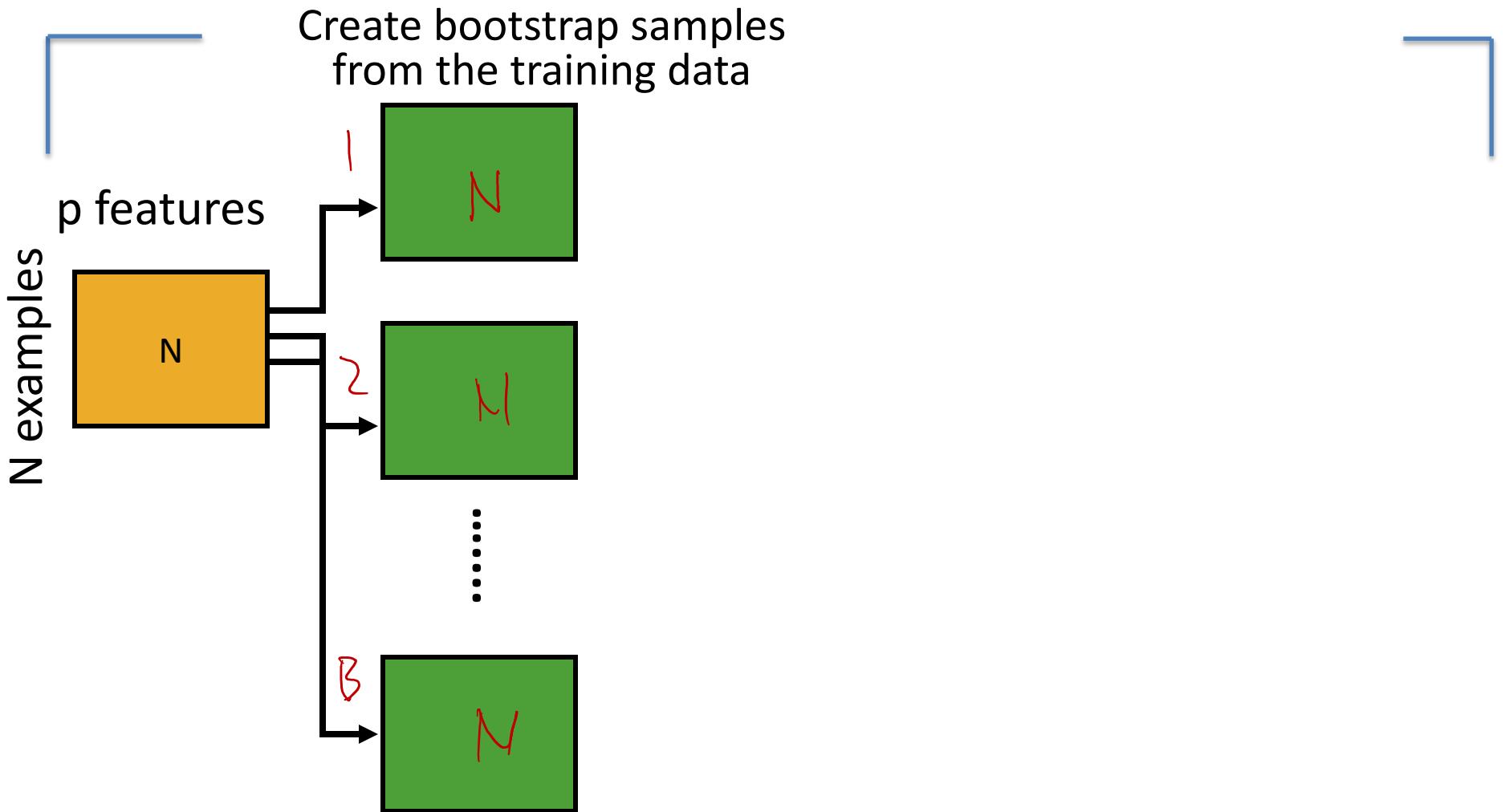
# With vs Without Replacement

- 
- **Bootstrap with replacement** can keep the sampling size the same as the original size for every repeated sampling. The sampled data groups are independent on each other.

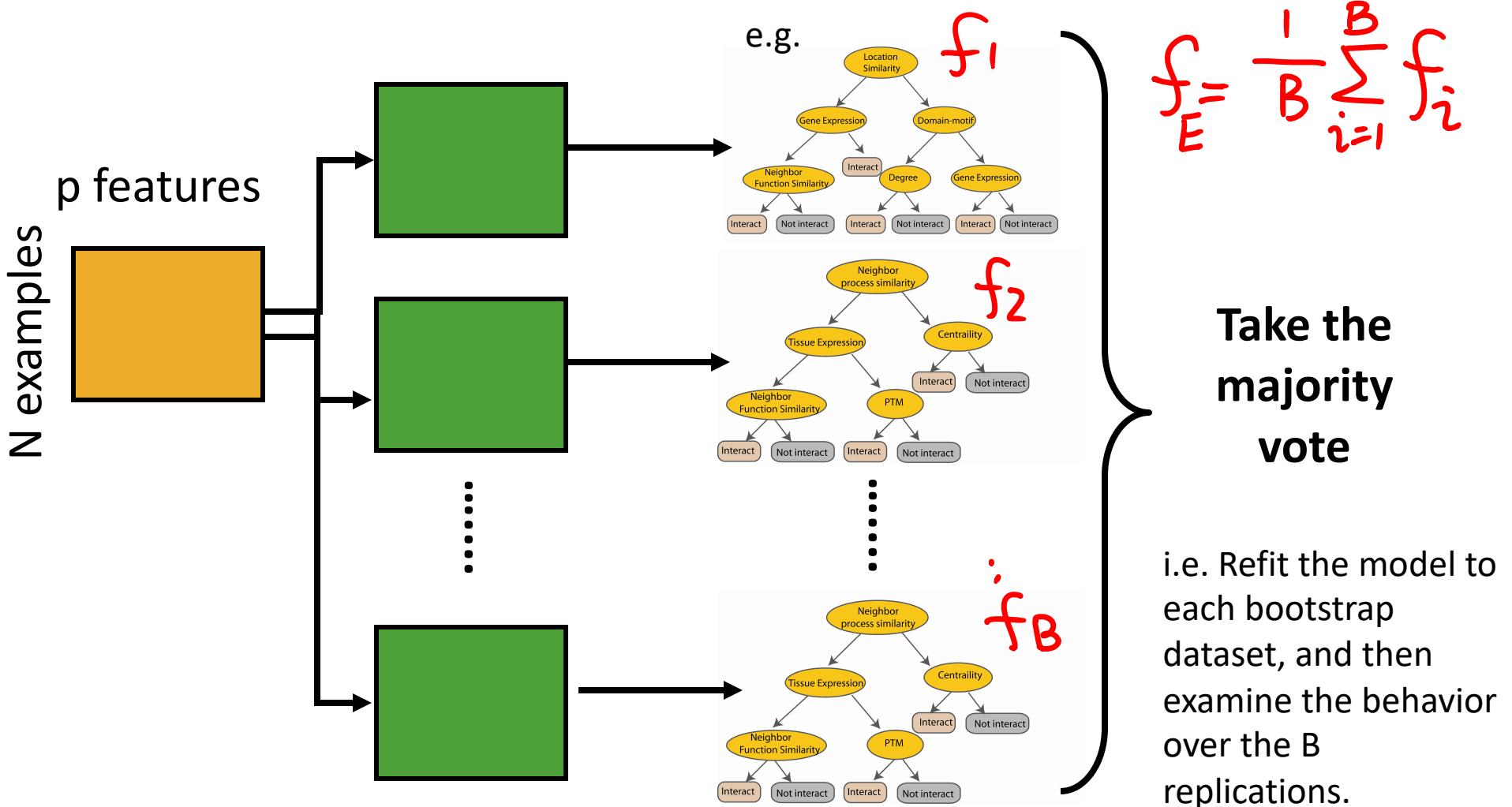
$$S_1, S_2, \dots, S_B$$

- **Bootstrap without replacement** cannot keep the sampling size the same as the original size for every repeated sampling. The sampled data groups are dependent on each other.

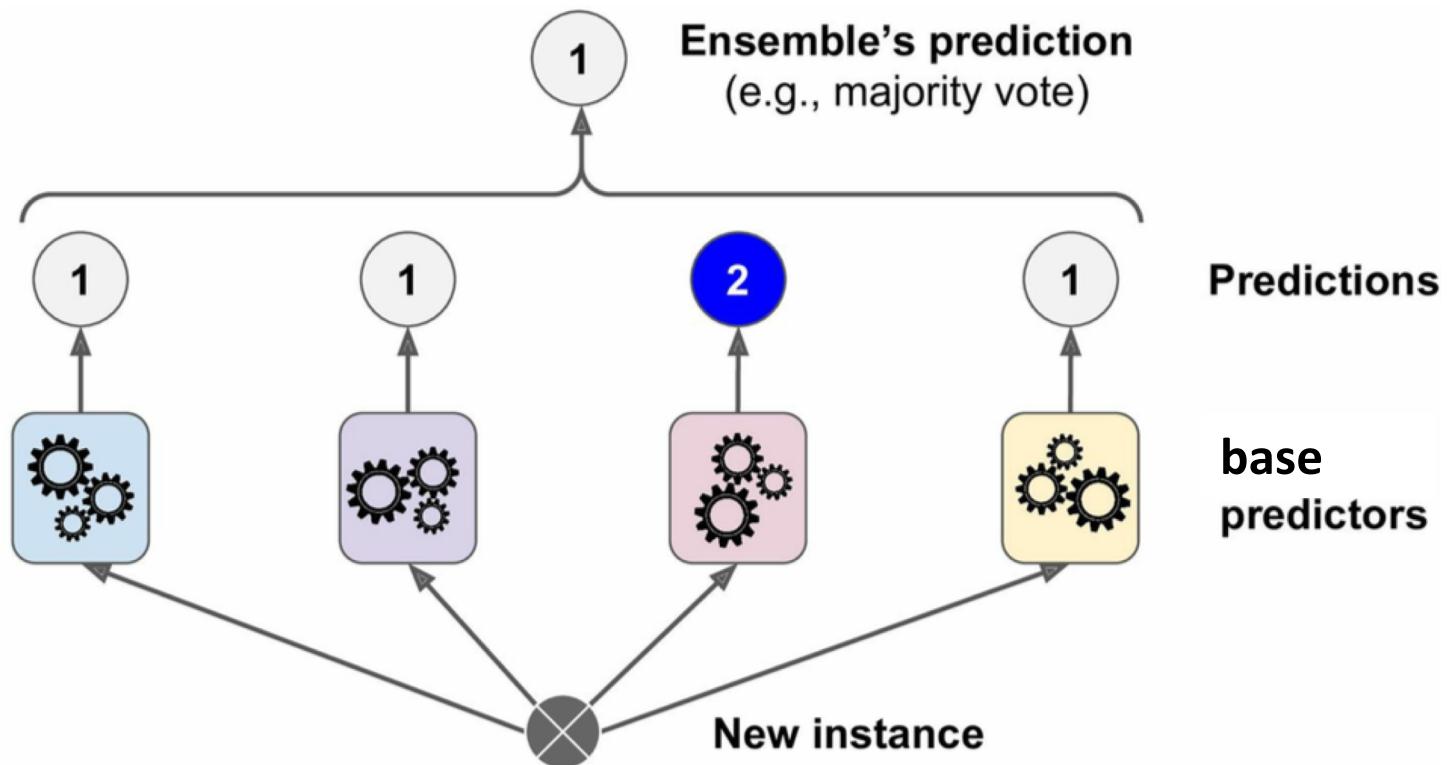
# Bagging



# Bagging of DT Classifiers



# E.g., Predict by Hard voting



$$\text{Var}(f_i)$$

vs.

$$\text{Var}\left(\frac{1}{B} \sum_{i=1}^B f_i\right)$$

## Decision Boundary Comparison

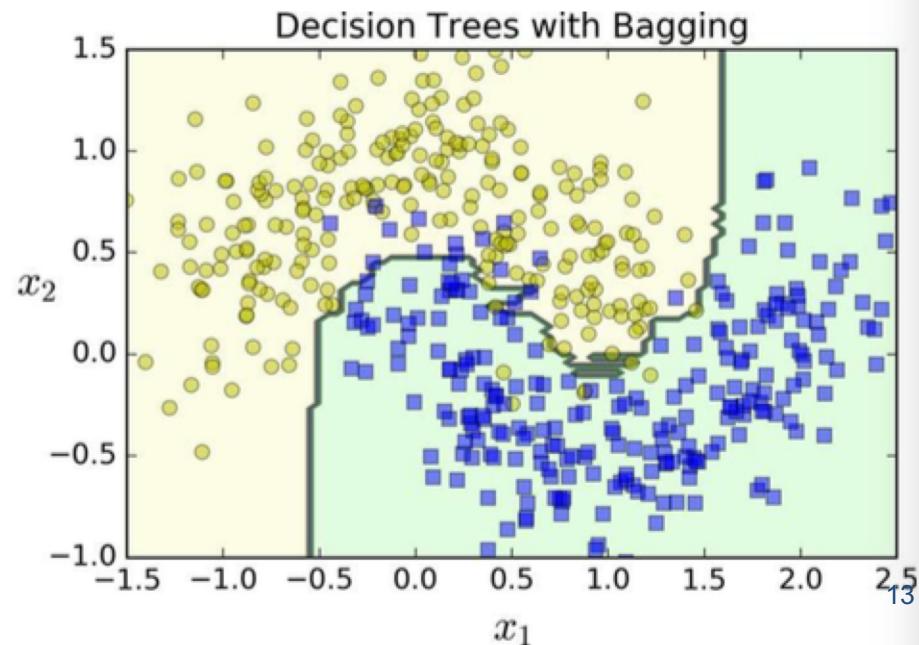
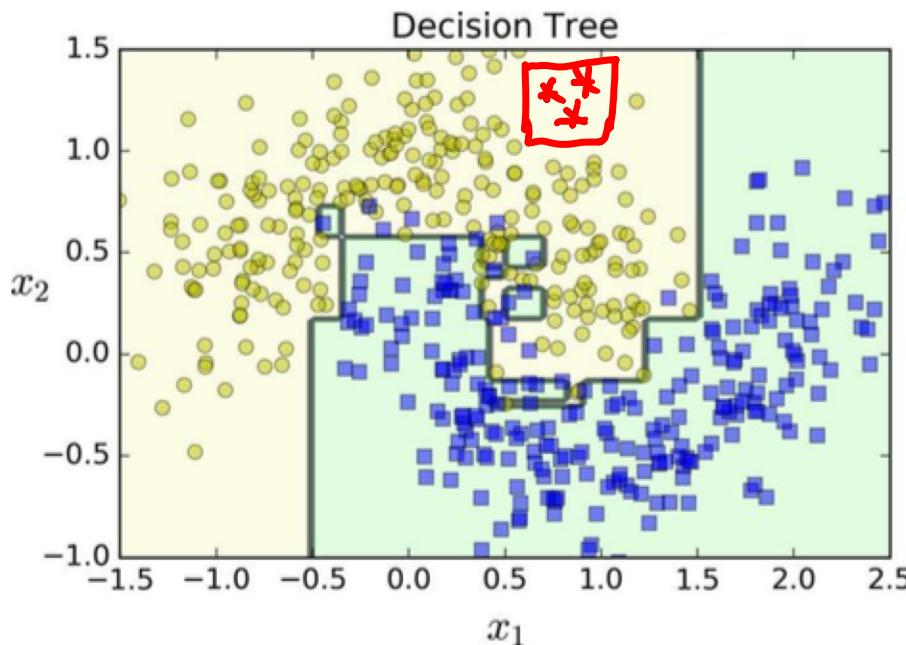
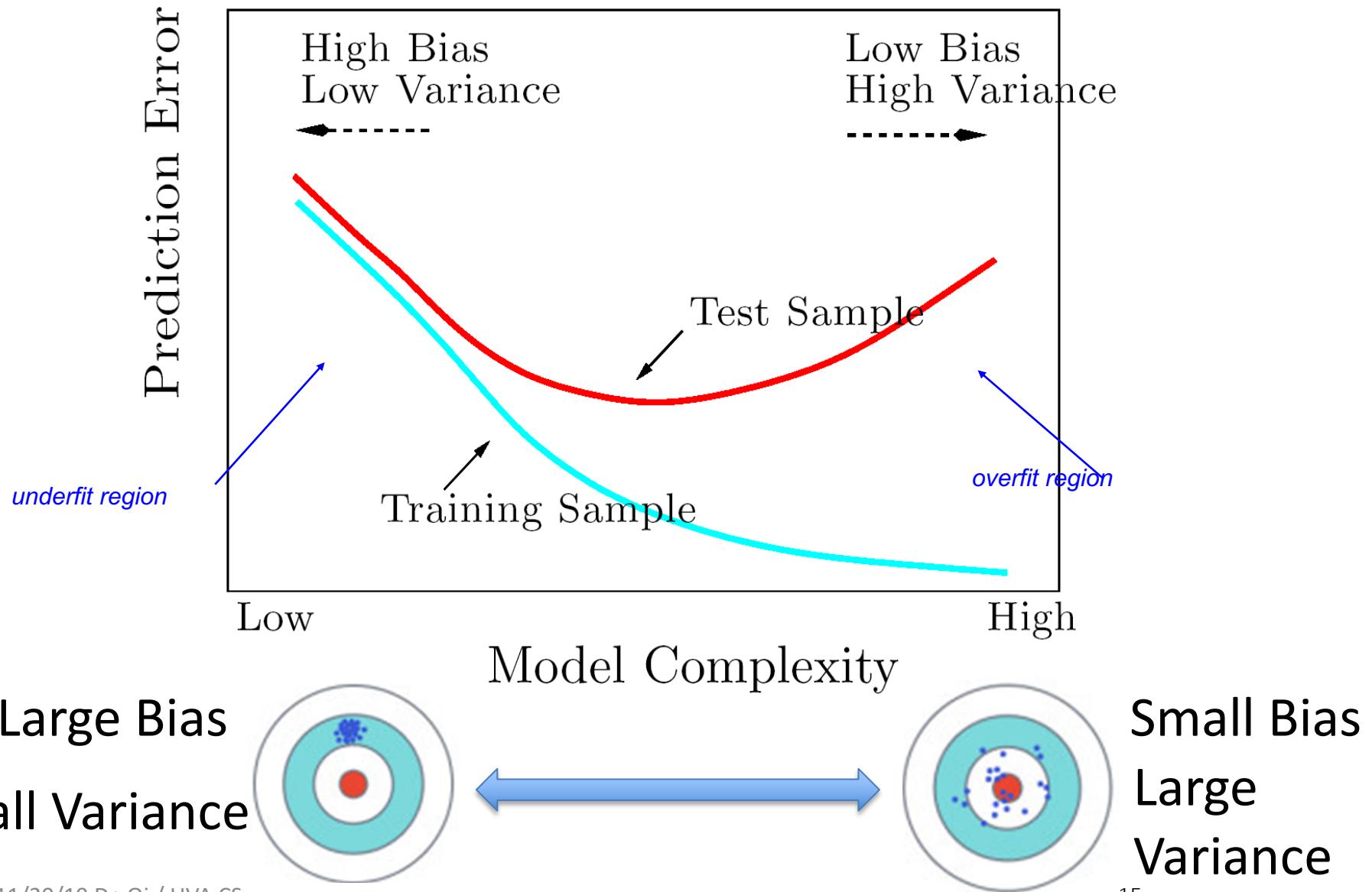


Figure 7-5. A single Decision Tree versus a bagging ensemble of 500 trees

# Peculiarities of Bagging

- Model Instability is good when bagging
  - The more variable (unstable) the basic model is, the more improvement can potentially be obtained
  - Low-Variability methods (e.g. LDA) improve less than High-Variability methods (e.g. decision trees)

# Recap: Bias-Variance Tradeoff / Model Selection



classifiers  $f_1(x)$ ,

$s_1$

$f_2(x)$ ,

$s_2$

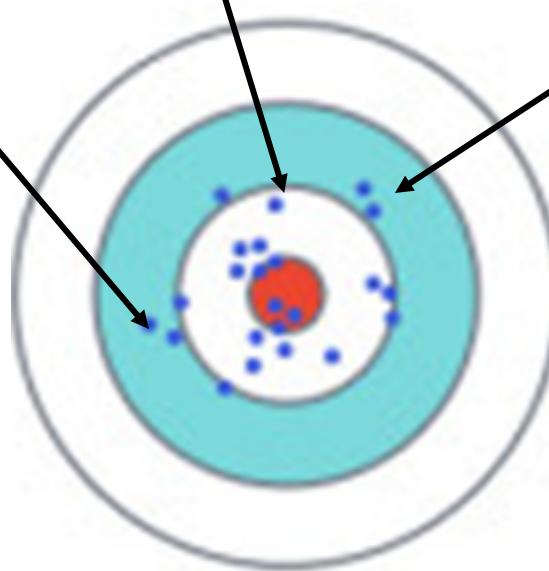
$f_3(x), \dots$

$f_B(x)$

$s_B$

A complex model will have large variance.

We can average complex models to reduce variance.



If we average all the  $f_i$ , is it close to  $f^*$

$$E[\hat{f}] = f^*$$

$$\text{Var}(aX + bY) = a^2 \text{Var}(X) + b^2 \text{Var}(Y) + 2ab\text{Cov}(X, Y)$$

When  $\text{Cov}(\hat{f}_i, \hat{f}_j) = 0$

$$Var(\hat{f}) = E((\hat{f} - \bar{f})^2) = Var\left(\frac{1}{B} \sum_{i=1}^B \hat{f}_i\right) = \frac{1}{B^2} \sum_{i=1}^B Var(\hat{f}_i) =$$

Base classifiers  $f_1(x), f_2(x), f_3(x), \dots, f_B(x)$

$$Var(\hat{f}) = Var\left(\frac{1}{B} \sum_{i=1}^B \hat{f}_i\right)$$

$$= \frac{1}{B^2} \sum_{i=1}^B Var(f_i) \quad \text{when } \forall i, j, \text{Cov}(f_i, f_j) = 0$$

$$= \frac{1}{B} Var(f_i)$$

Because  
 $|S_1| = |S_2| = \dots = |S_B|$

$Var(f_i) \approx Var(f_i)$

# Bagging : an extreme case study using simulated data (with correlated features)

$N = 300$  training samples,  $y \in \{0, 1\}$ ,  $x_1, x_2, \dots, x_5$

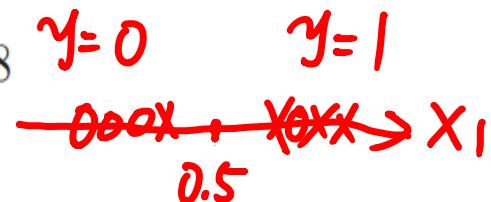
$Y$ : Two classes and  $X$ :  $p = 5$  features,

Each feature  $N(0, 1)$  distribution and pairwise correlation .95

Response  $Y$ : generated according to:

$$\Pr(Y = 1|x_1 \leq 0.5) = 0.2$$

$$\Pr(Y = 1|x_1 > 0.5) = 0.8$$

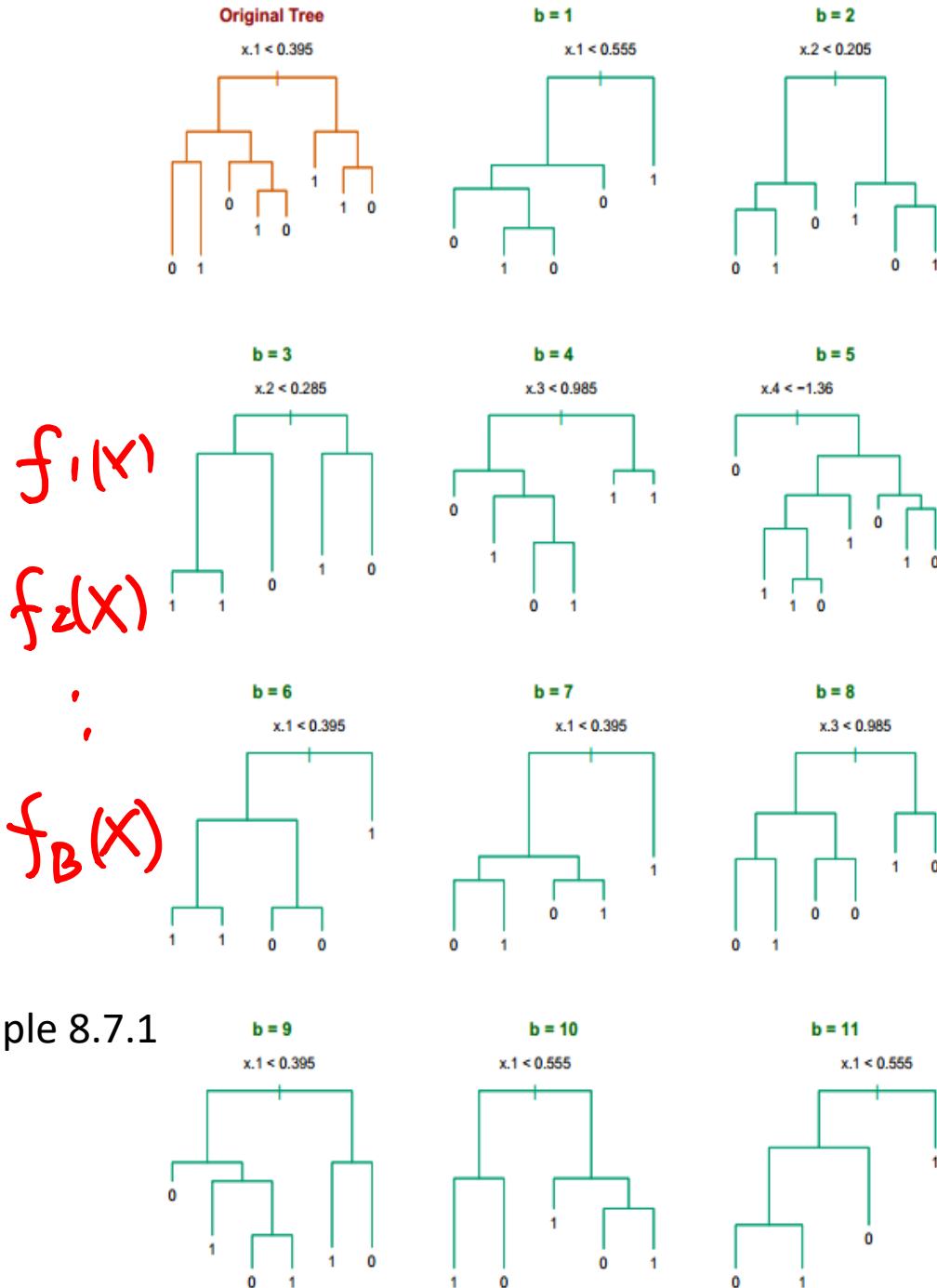


Test sample size of 2000

Fit classification trees to training set and bootstrap samples

$B = 200$

Notice the bootstrap trees look quite different from the original tree



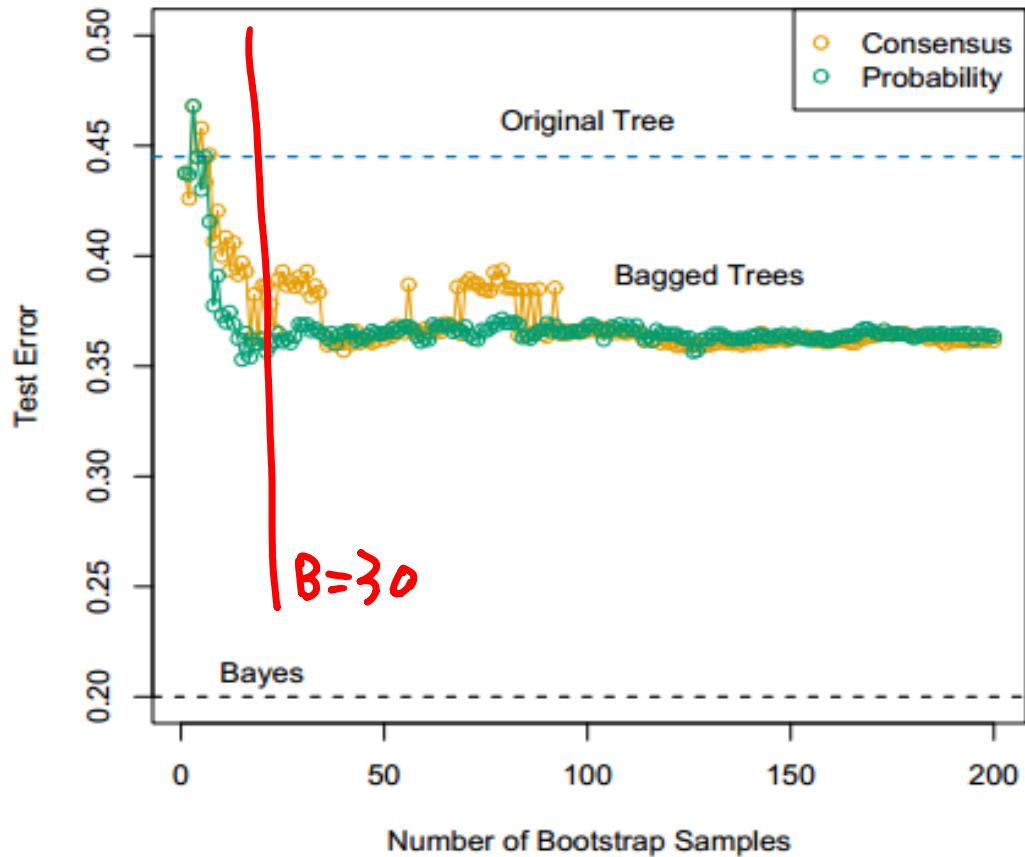
Five features highly correlated with each other

→ No clear difference with picking up which feature to split

→ Small changes in the training set will result in different tree

→ But these trees are actually quite similar wrt output classification

ESL book / Example 8.7.1



→ For  $B > 30$ , more trees do not improve the bagging results

→ Since the trees correlate highly to each other and give similar classifications

$B$

$$\text{Var}\left(\frac{1}{B} \sum f_i\right)$$

Consensus: Majority vote

Probability: Average distribution at terminal nodes

# Bagging

- Slightly increases model complexity
  - Cannot help when greater enlargement of model diversity is needed
- Bagged trees are correlated
  - Use random forest to reduce correlation between trees

$$\text{Var}(aX + bY) = a^2\text{Var}(X) + b^2\text{Var}(Y) + 2ab\text{Cov}(X, Y)$$

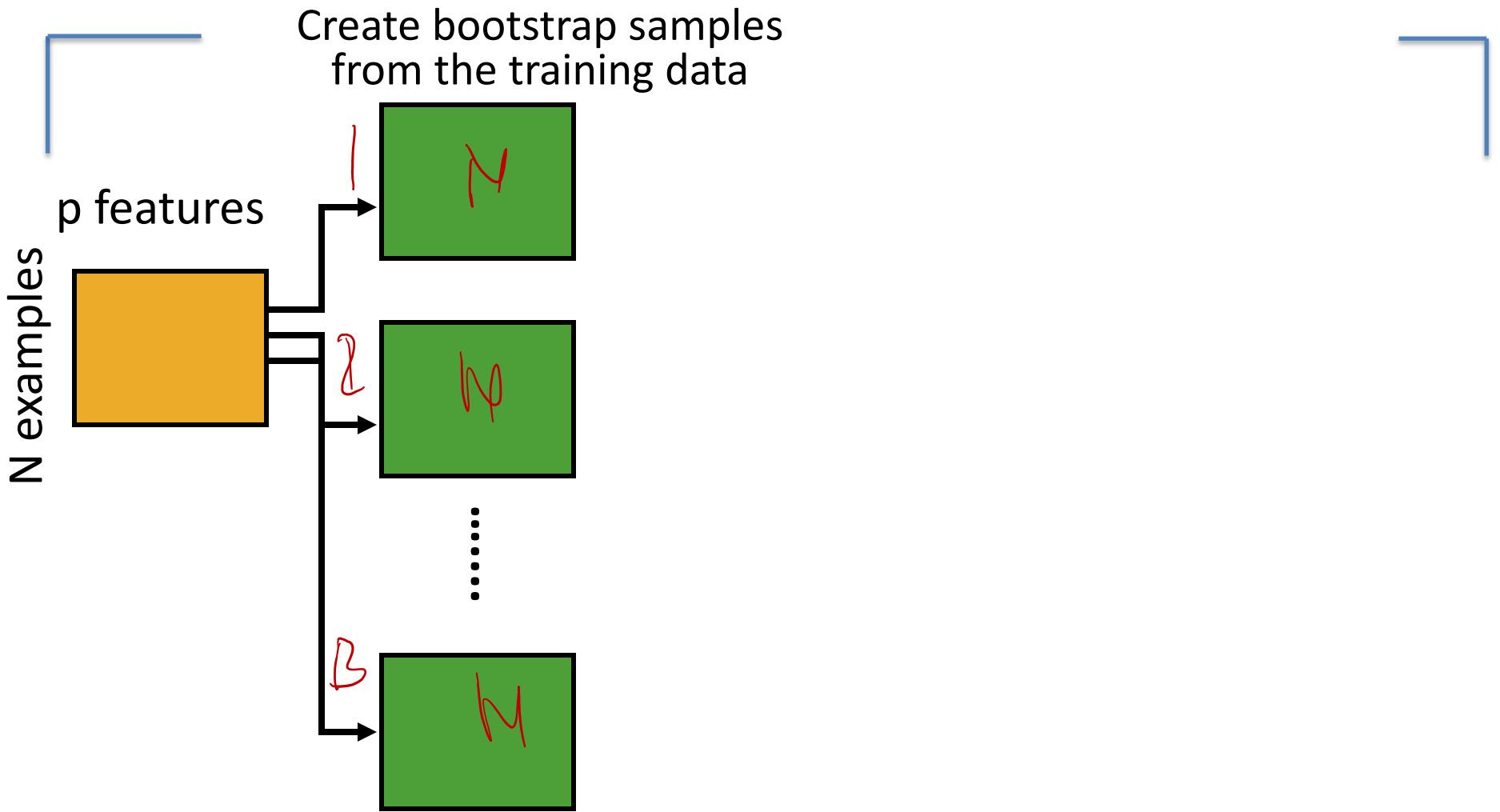
# Today: Ensemble

- Bagging
  - Bagged Decision Tree
- Random forests:
  - Boosting
    - Adaboost
    - Xgboost
- Stacking

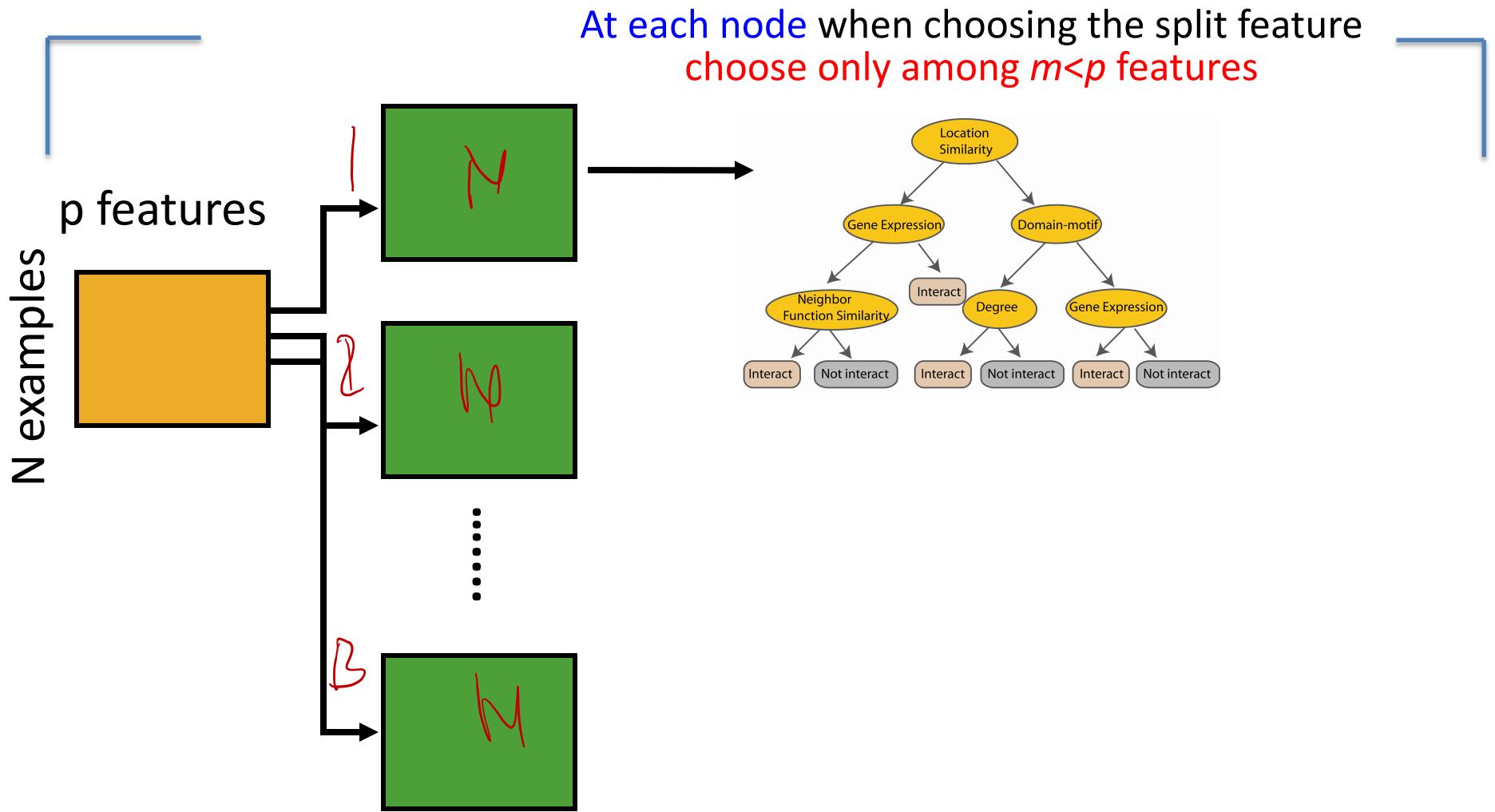
# Random forest classifier

- Random forest classifier,
  - an extension to bagging
  - which *uses de-correlated trees.*

# Random Forest Classifier

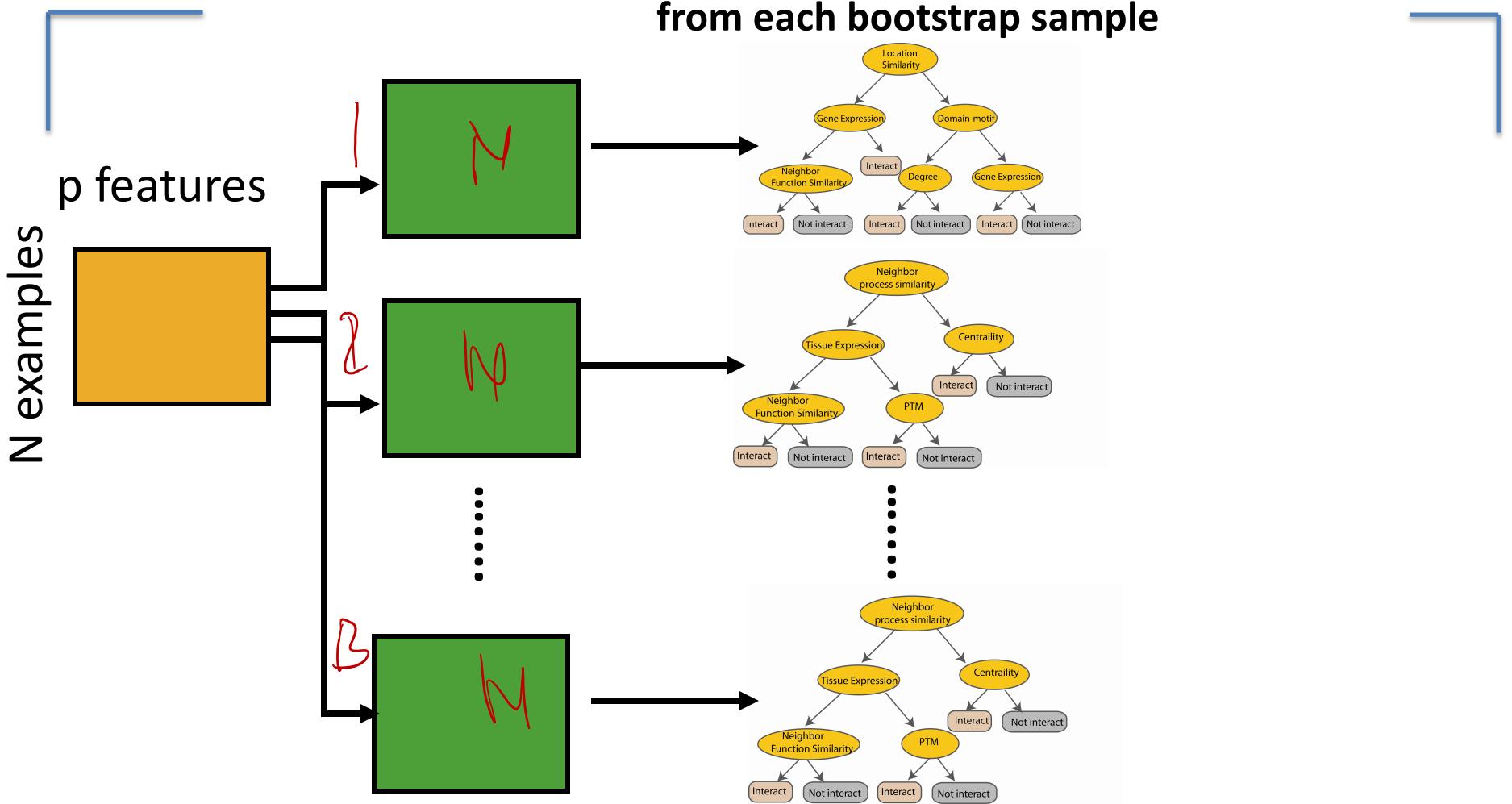


# Random Forest Classifier

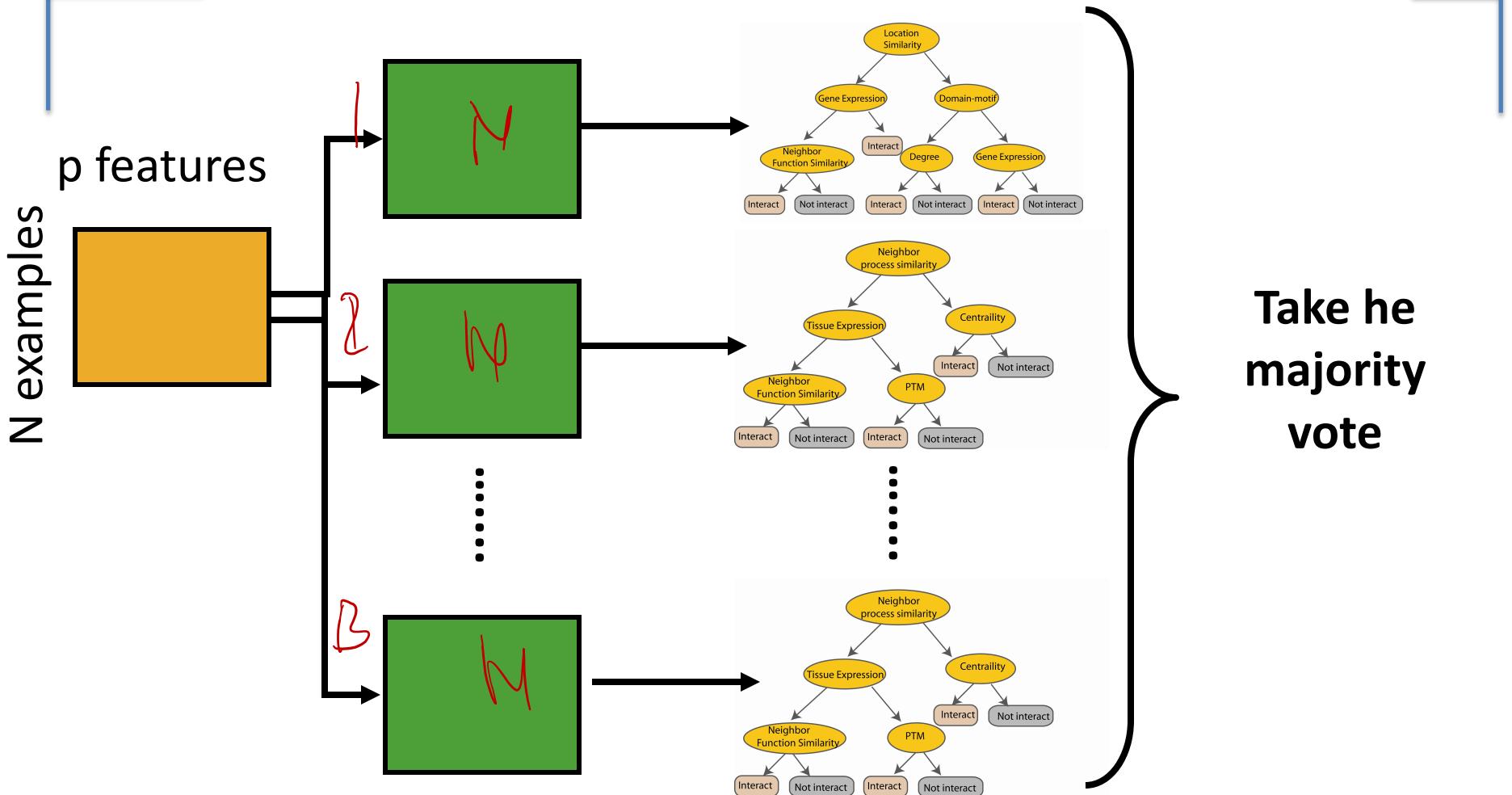


# Random Forest Classifier

Create decision tree  
from each bootstrap sample



# Random Forest Classifier



Take the  
majority  
vote

# Random Forests

For each of our  $B$  bootstrap samples

Form a tree in the following manner

i: Given  $p$  dimensions, **pick  $m$  of them**

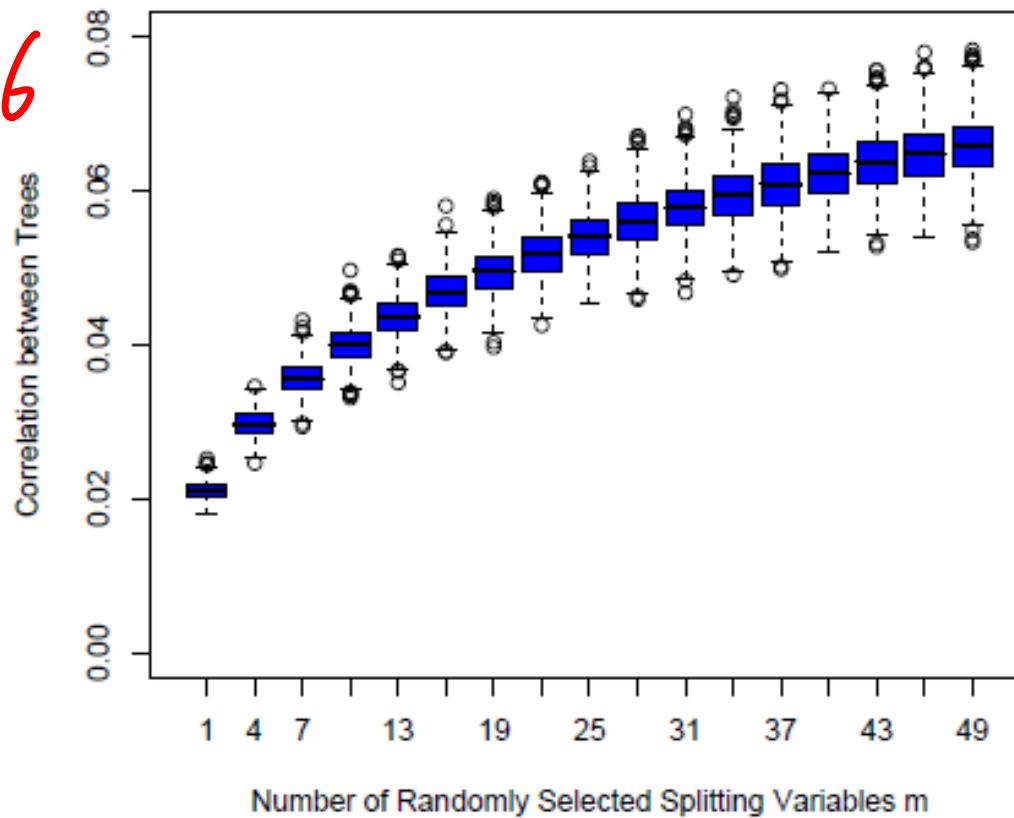
ii: **Split only according to these  $m$  dimensions**

(we will NOT consider the other  $p-m$  dimensions)

Repeat the above steps i & ii for each split

Note: we pick a different set of  $m$  dimensions for each split  
on a single tree

$P = 0.06$



**FIGURE 15.9.** *Correlations between pairs of trees drawn by a random-forest regression algorithm, as a function of  $m$ . The boxplots represent the correlations at 600 randomly chosen prediction points  $x$ .*

Page 598-599 In ESL book

# Random Forests

Random forest can be viewed as a refinement of bagging with a tweak of **decorrelating** the trees:

At each tree split, a random subset of **m** features out of all **p** features is drawn to be considered for splitting

Some guidelines provided by Breiman, but be careful to choose m based on specific problem:

- {
  - m = p amounts to bagging
  - m = p/3 or log2(p) for regression
  - m = sqrt(p) for classification

# **Why correlated trees are not ideal ?**

Random Forests try to reduce correlation between the trees.

Why?

# Why correlated trees are not ideal ?

Assuming each tree has variance  $\sigma^2$

If trees are independently identically distributed, then average variance is  $\sigma^2/B$

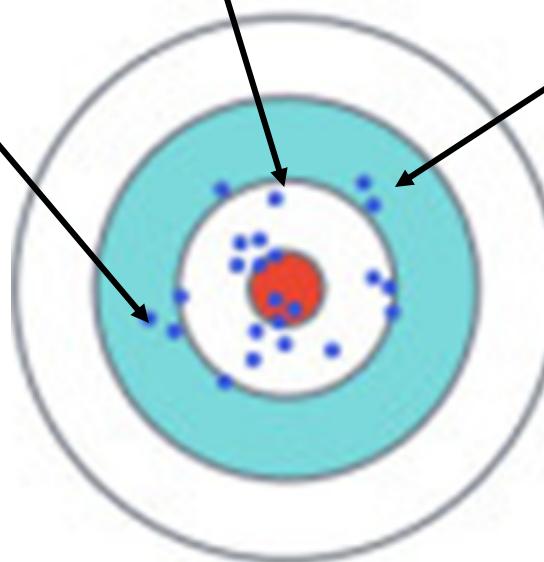
$$\text{Var}(aX + bY) = a^2 \text{Var}(X) + b^2 \text{Var}(Y) + 2ab\text{Cov}(X, Y)$$

$$\text{Var}(\hat{f}) = E((\hat{f} - \bar{f})^2) = \text{Var}\left(\frac{1}{B} \sum \hat{f}_i\right) = \frac{1}{B^2} \sum \text{Var}(\hat{f}_i) = \frac{1}{B} \sigma^2$$

classifiers  $f_1(x), f_2(x), f_3(x), \dots, f_B(x)$

A complex model will have large variance.

We can average complex models to reduce variance.



If we average all the  $f_i$ , is it close to  $f^*$

$$E[\hat{f}] = f^*$$

# Why correlated trees are not ideal ?

Assuming each tree has variance  $\sigma^2$

If simply identically distributed, then average variance is

$$\rho_{ij} = \text{Corr}(f_i, f_j) = \rho \sigma^2 + \frac{1 - \rho}{B} \sigma^2$$

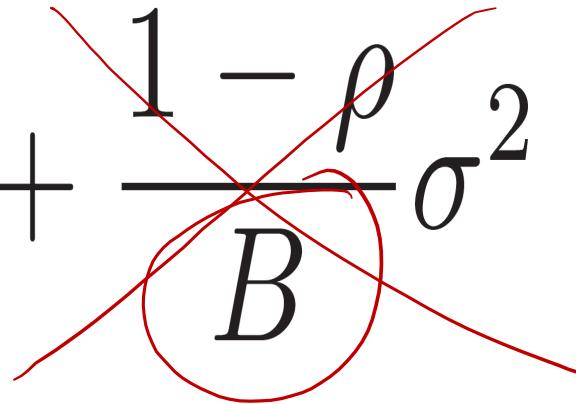
As  $B \rightarrow \infty$ , second term  $\rightarrow 0$

Thus, the pairwise correlation always affects the variance

# Why correlated trees are not ideal ?

Assuming each tree has variance  $\sigma^2$

If simply identically distributed, then average variance is

$$\Rightarrow \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$


RAM x B  
trainTim x B  
testTime x B

As  $B \rightarrow \infty$ , second term  $\rightarrow 0$

Thus, the pairwise correlation always affects the variance

# Why correlated trees are not ideal ?

How to deal?

If we reduce  $m$  (the number of dimensions we actually consider in each splitting ),

then we reduce the pairwise tree correlation

Thus, variance will be reduced.

# More about Random Forests

1. Construct subset  $(x_1^*, y_1^*), \dots, (x_n^*, y_n^*)$  by sampling original training set with replacement.
2. Build tree-structured learners  $h(x, \Theta_k)$ , where at each node, **m predictors at random** are selected before finding the best split.
  - Gini Criterion.
  - No pruning.
3. Combine the predictions (average or majority vote) to get the final result.

# Today: Ensemble

- Bagging
  - Bagged Decision Tree
  - Random forests:
- Boosting
  - Adaboost
  - Xgboost
- Stacking



# e.g. Ensembles in practice



Oct 2006 -  
2009

Each rating/sample:

+ <user, movie, date of grade, grade>

Training set (100,480,507 ratings)

Qualifying set (2,817,131 ratings) → winner

- Training data is a set of users and ratings (1,2,3,4,5 stars) those users have given to movies.
- Predict what rating a user would give to any movie
- \$1 million prize for a 10% improvement over Netflix' s current method ( $MSE = 0.9514$ )

# Ensemble in practice

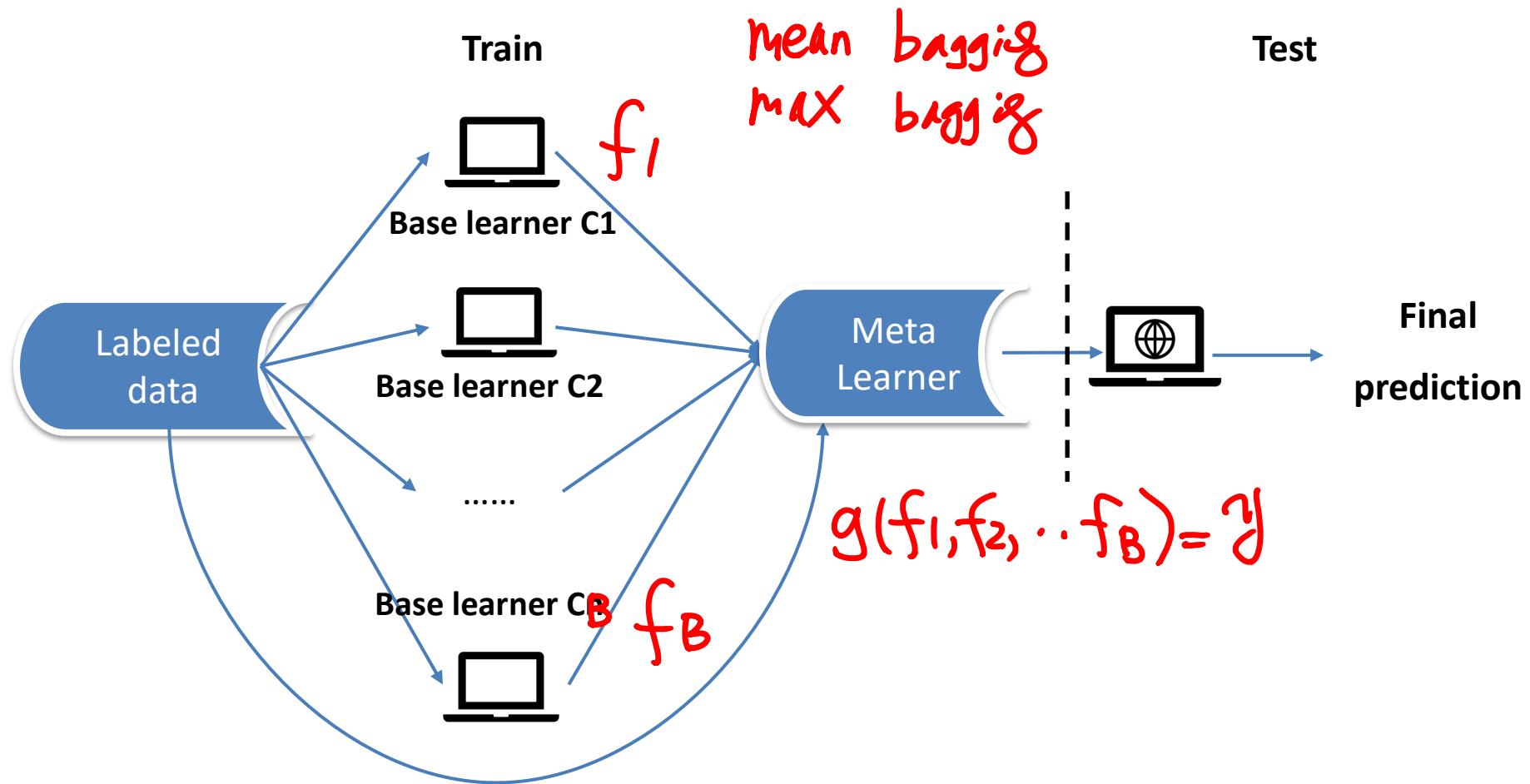
Team “Bellkor’s Pragmatic Chaos” defeated the team “ensemble” by submitting just 20 minutes earlier! → 1 million dollar !

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
<b><u>Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos</u></b>				
1	<a href="#">BellKor's Pragmatic Chaos</a>	0.8567	10.06	2009-07-26 18:18:28
2	<a href="#">The Ensemble</a>	0.8567	10.06	2009-07-26 18:38:22
3	<a href="#">Grand Prize Team</a>	0.8582	9.90	2009-07-10 21:24:40
4	<a href="#">Opera Solutions and Vandelay United</a>	0.8588	9.84	2009-07-10 01:12:31
5	<a href="#">Vandelay Industries!</a>	0.8591	9.81	2009-07-10 00:32:20
6	<a href="#">PragmaticTheory</a>	0.8594	9.77	2009-06-24 12:06:56
7	<a href="#">BellKor in BigChaos</a>	0.8601	9.70	2009-05-13 08:14:09
8	<a href="#">Dace</a>	0.8612	9.59	2009-07-24 17:18:43
9	<a href="#">Feeds2</a>	0.8622	9.48	2009-07-12 13:11:51
10	<a href="#">BigChaos</a>	0.8623	9.47	2009-04-07 12:33:59
11	<a href="#">Opera Solutions</a>	0.8623	9.47	2009-07-24 00:34:07
12	<a href="#">BellKor</a>	0.8624	9.46	2009-07-26 17:19:11

The ensemble team → blenders of multiple different methods

# Stacking

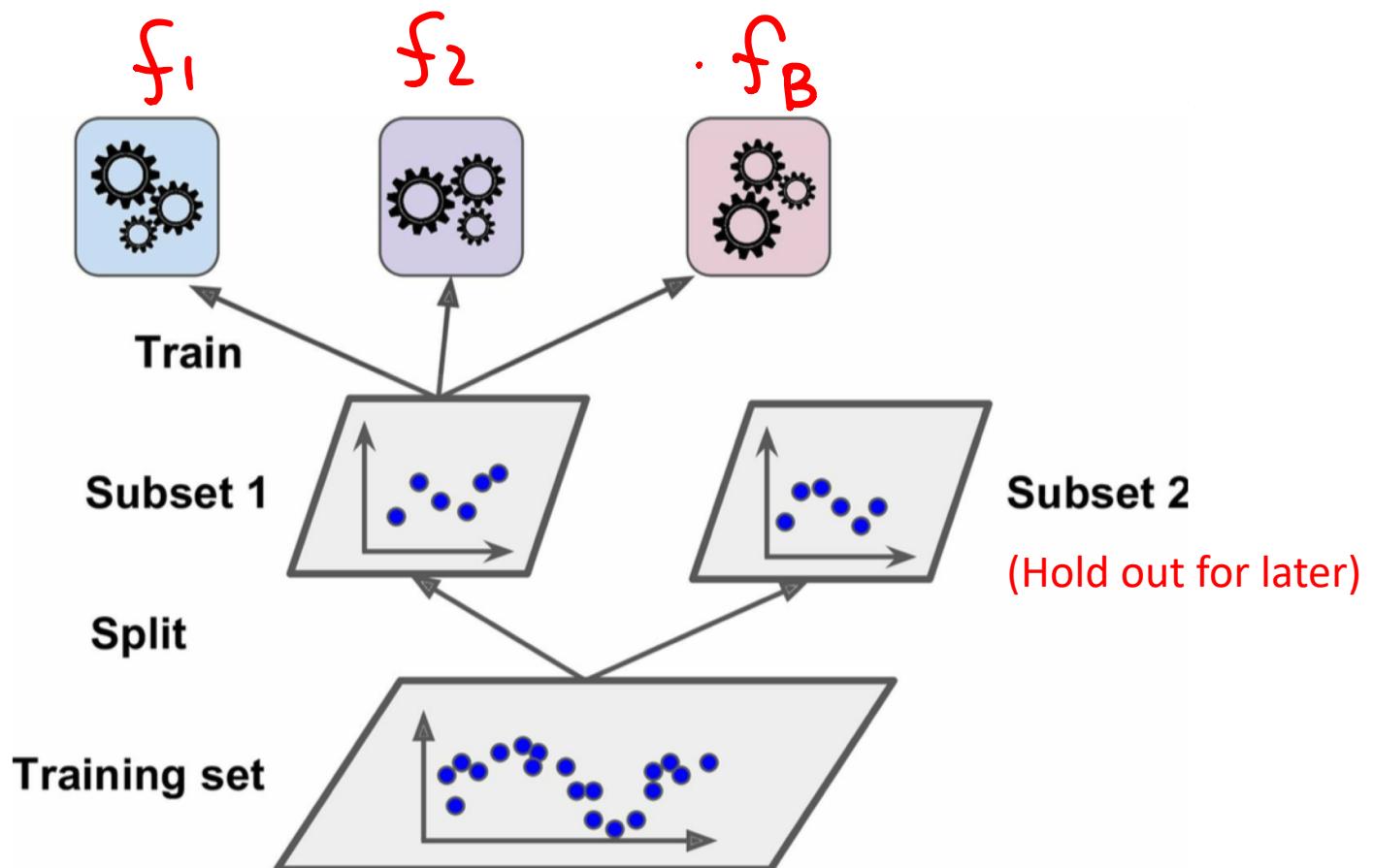
- Main Idea: Learn and combine multiple classifiers



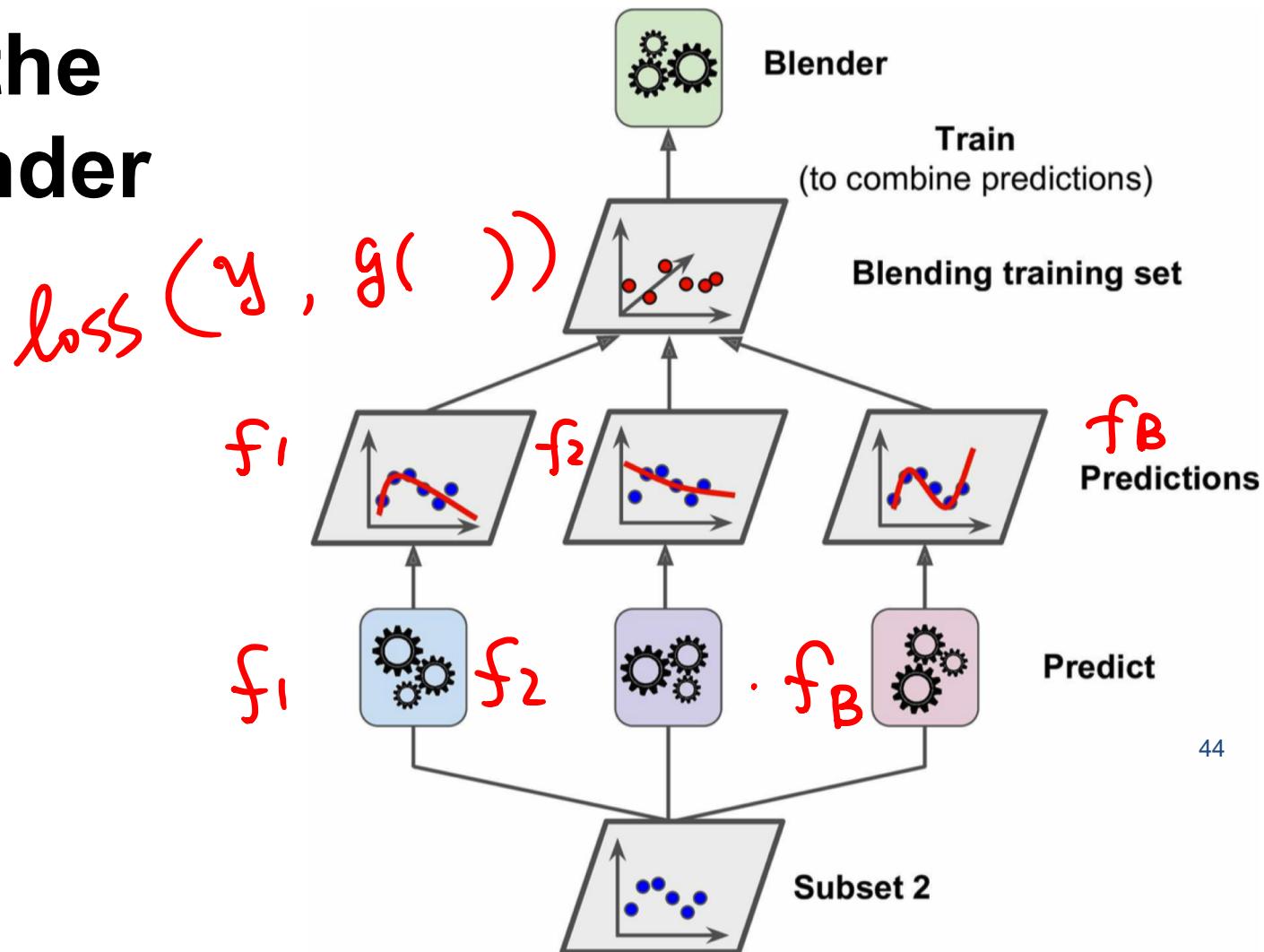
# Generating Base and Meta Learners

- **Base model—efficiency, accuracy and diversity**
    - Sampling training examples
    - Sampling features
    - Using different learning models
  - **Meta learner**
    - Majority voting
    - Weighted averaging
    - .....
    - Higher level classifier — Supervised (e.g. Xgboost as blender)
- 
- Unsupervised

# Training the base predictors



# Training the meta blender



# Today: Ensemble

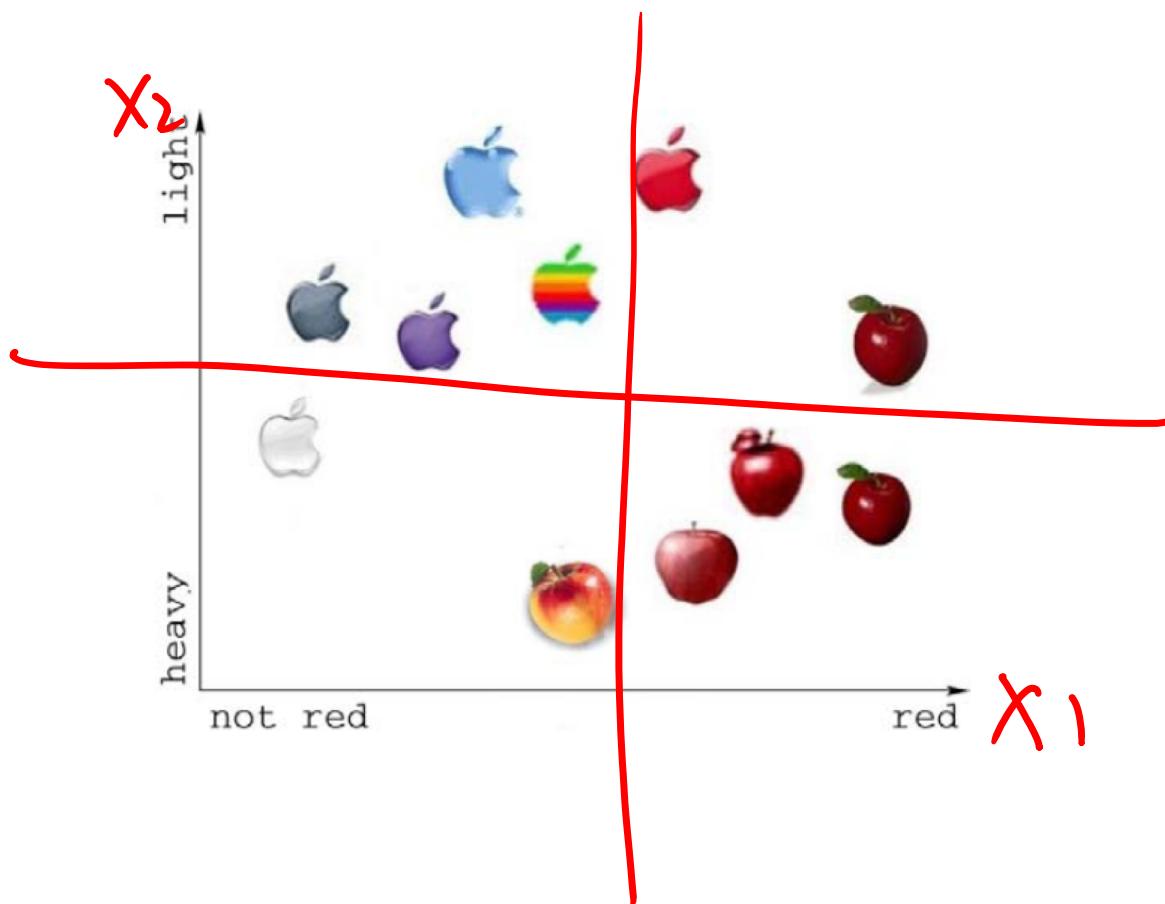
- Bagging
  - Bagged Decision Tree
  - Random forests:
- Boosting
  - Adaboost
  - Xgboost
- Stacking

# Boosting Strategies

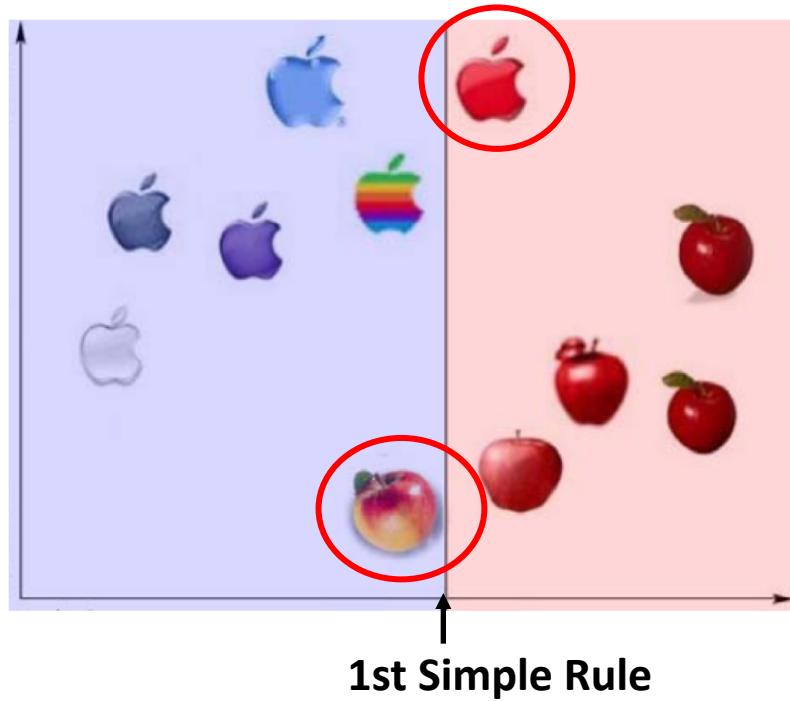
base learners : shallow DT  
High bias

1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules

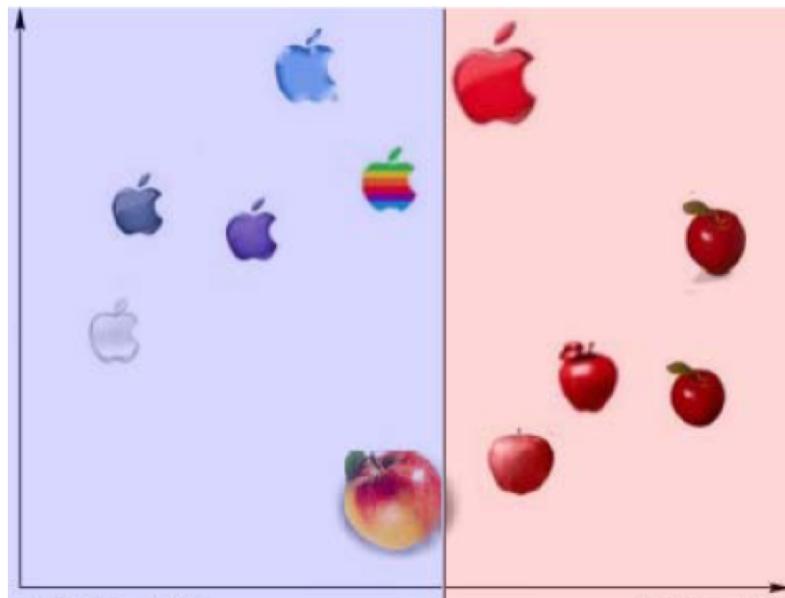
- Recognizing apples:
- (1) Collect a set of real apples and plastic apples
- (2) Observe some rules to tell them apart based on their characteristics



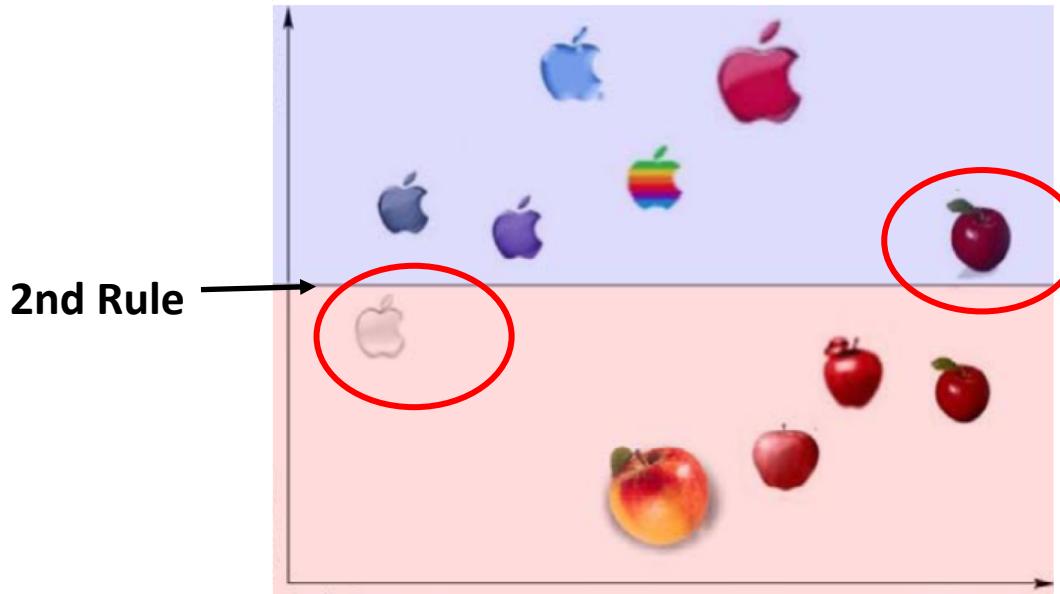
1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



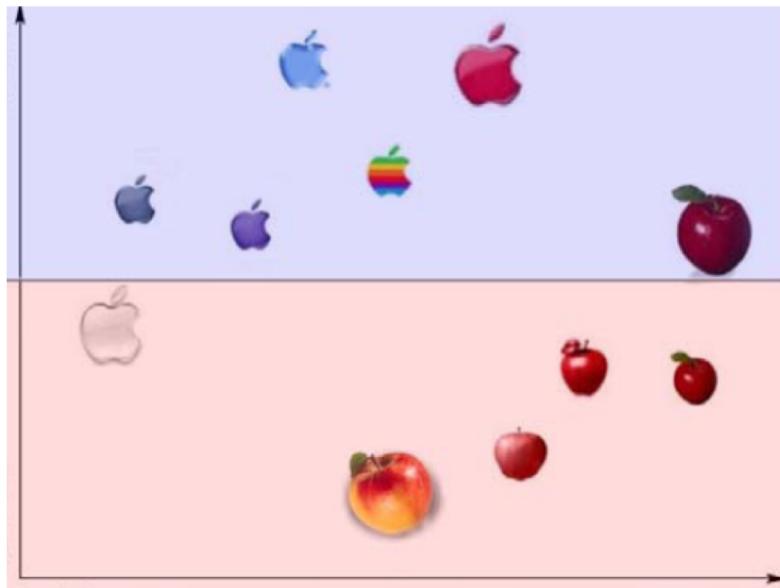
1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



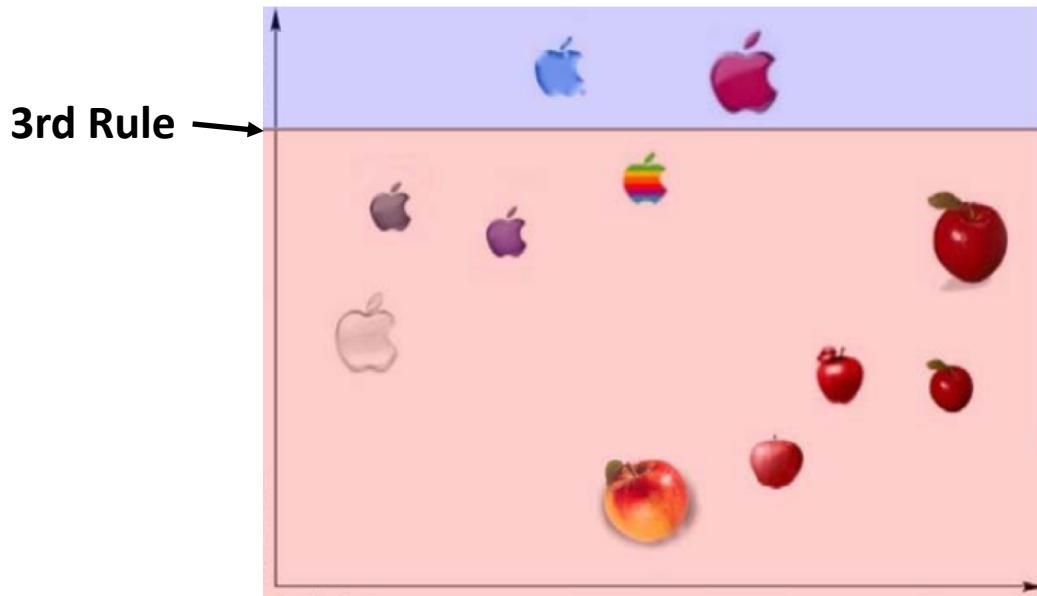
1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



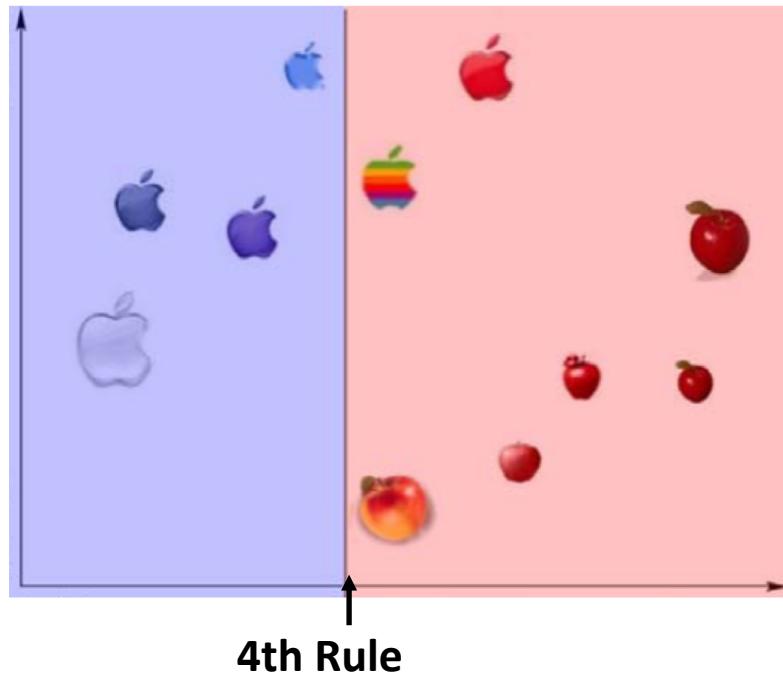
1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules

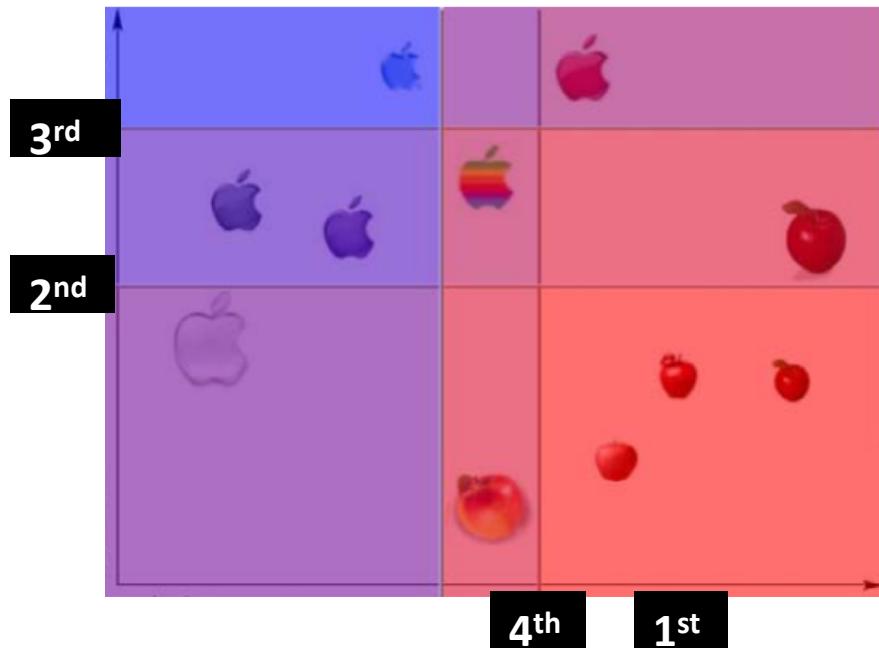


1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules

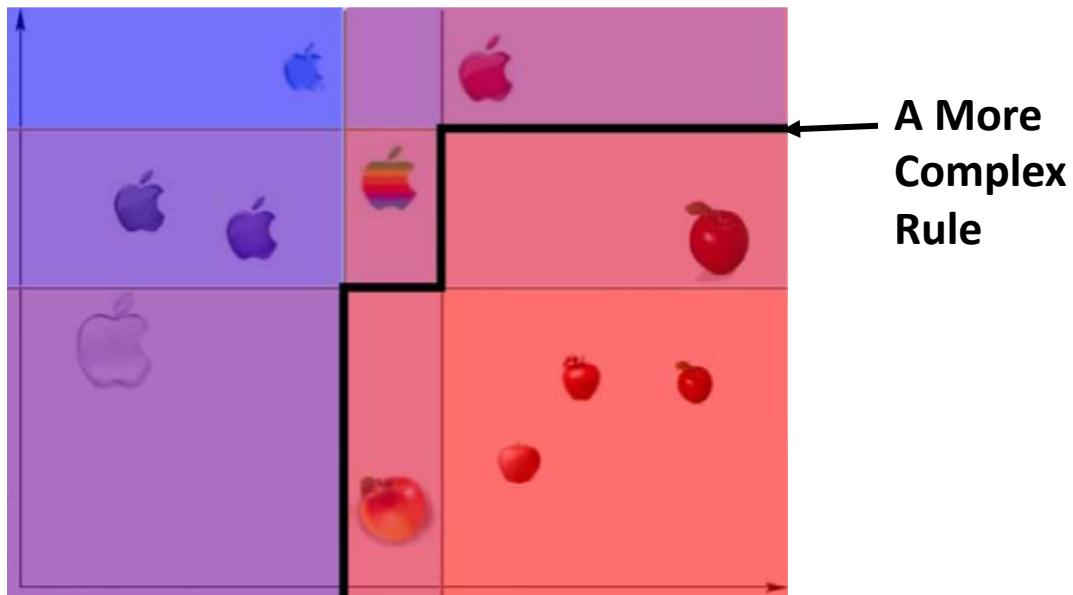


53

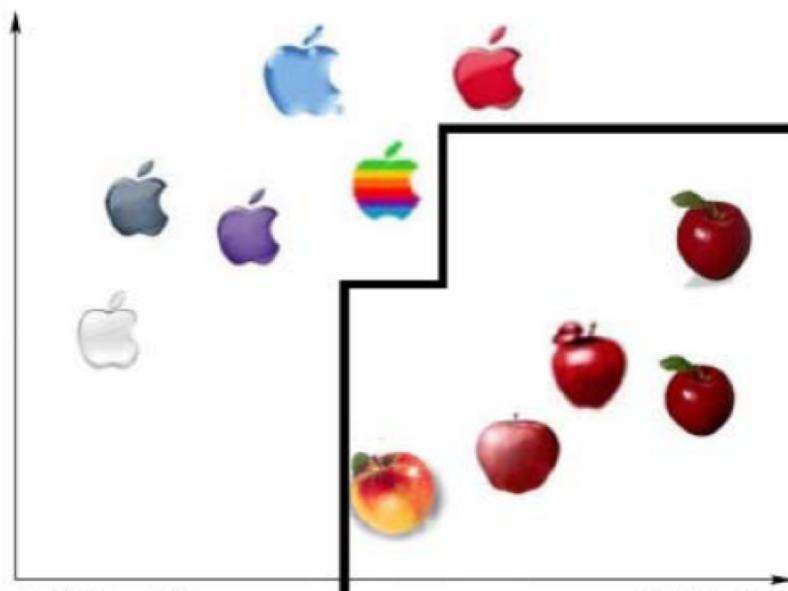
1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



Final Classifier is the additive combination of base rules:



# Adaboost Algorithm (Proposed by Robert Schapire)

**Training Data:**  $\mathbf{D} = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \mathcal{R}^n, y_i \in \{-1, 1\}, 1 \leq i \leq m\}$

Set uniform example weight  $w_i, 1 \leq i \leq m$

**For  $t = 1$  to  $T$  iterations:**

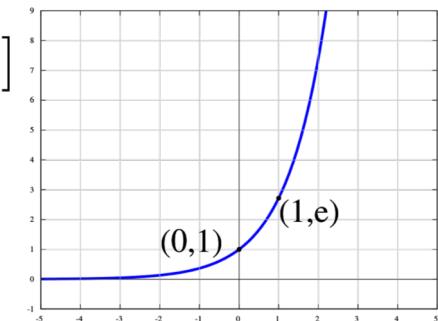
Select a base classifier:  $h_t(\mathbf{x}_i) = \arg \min(\epsilon_t)$

$$\epsilon_t = \sum_{i=1}^m w_i [y_i \neq h_t(\mathbf{x}_i)]$$

Set classifier weight:  $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$

Update example weight:  $w_i = w_i e^{-\alpha_t y_i h_t(\mathbf{x}_i)}$

**Final Classifier:**  $\hat{f} = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(\mathbf{x}_i) \right)$



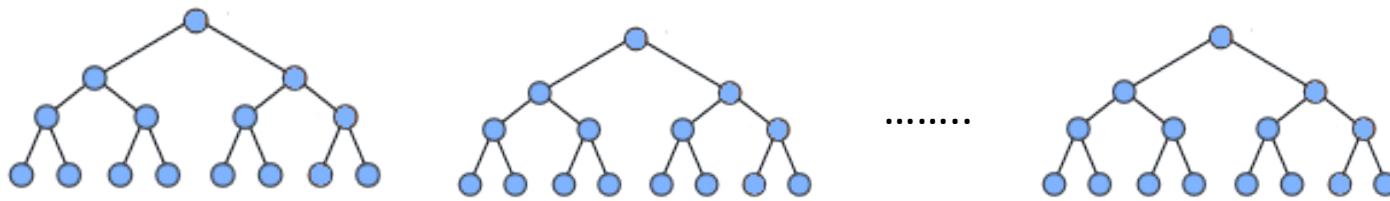
# Boosting vs. Bagging

- Similar to bagging, boosting combines a weighted sum of many classifiers, thus **it reduces variance**.
- One key difference: unlike bagging, boosting fit the tree to the entire training set, and adaptively weight the examples.
- Boosting tries to do better at each iteration, (by making model a bit more complex), thus **it reduces bias**.

# XGBoost

- Additive tree model: add new trees that complement the already-built ones
- Response is the optimal linear combination of all decision trees
- Popular in Kaggle Competitions for efficiency and accuracy

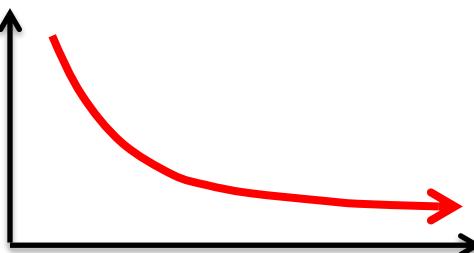
Additive tree model



More in 18c-  
extraBoosting  
Slides

Greedy Algorithm

Error



Number of Tree

# XGBoost

- XGBoost is a very efficient Gradient Boosting Decision Tree implementation with some interesting features:
- **Regularization:** Can use L1 or L2 regularization.
- **Handling sparse data:** Incorporates a sparsity-aware split finding algorithm to handle different types of sparsity patterns in the data.
- **Weighted quantile sketch:** Uses distributed weighted quantile sketch algorithm to effectively handle weighted data.
- **Block structure for parallel learning:** Makes use of multiple cores on the CPU, possible because of a block structure in its system design. Block structure enables the data layout to be reused.
- **Cache awareness:** Allocates internal buffers in each thread, where the gradient statistics can be stored.
- **Out-of-core computing:** Optimizes the available disk space and maximizes its usage when handling huge datasets that do not fit into memory.

# References

- Prof. Tan, Steinbach, Kumar's "Introduction to Data Mining" slide
- ESLbook : Hastie, Trevor, et al. *The elements of statistical learning*. Vol. 2. No. 1. New York: Springer, 2009.
- Dr. Oznur Tastan's slides about RF and DT
- Dr. Rich's slides about boosting