



Machine Learning

Lecture 18b: Random Forest / Boosting

Dr. Beilun Wang

Southeast University
School of Computer Science
and Engineering

Course Content Plan

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

Y is a continuous

Y is a discrete


NO Y

About $f()$

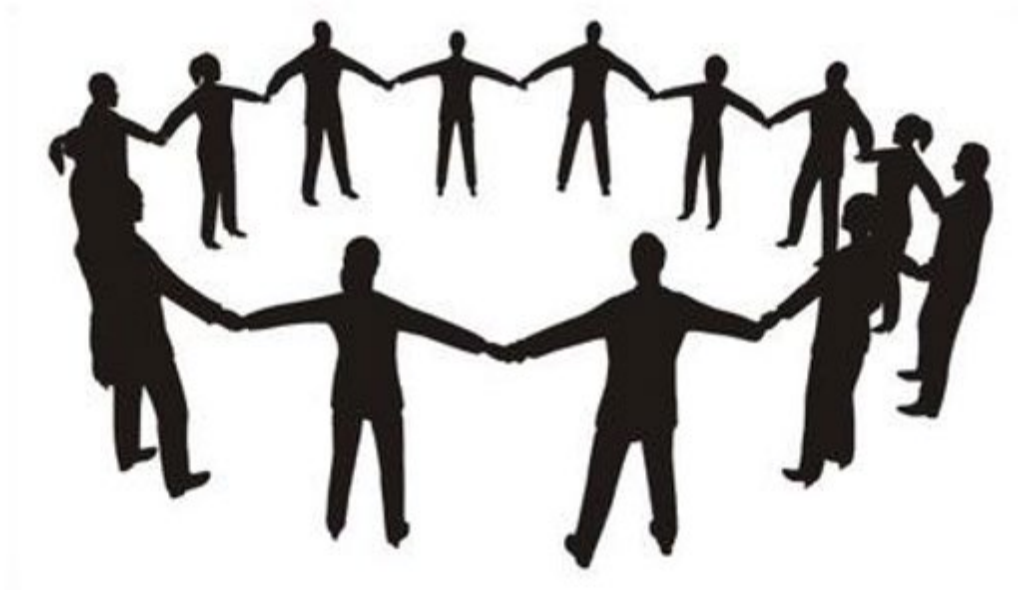
About interactions among X_1, \dots, X_p

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**
 - Discriminative
 - directly estimate a decision rule/boundary
 - ~~• e.g., support vector machine, **decision tree**, logistic regression, e.g. neural networks (NN), deep NN~~
-  • Generative:
 - build a generative statistical model
 - ~~• e.g., Bayesian networks, Naïve Bayes classifier~~
- ~~• Instance based classifiers~~
 - Use observation directly (no models)
 - ~~• e.g. K nearest neighbors~~

Today: Ensemble



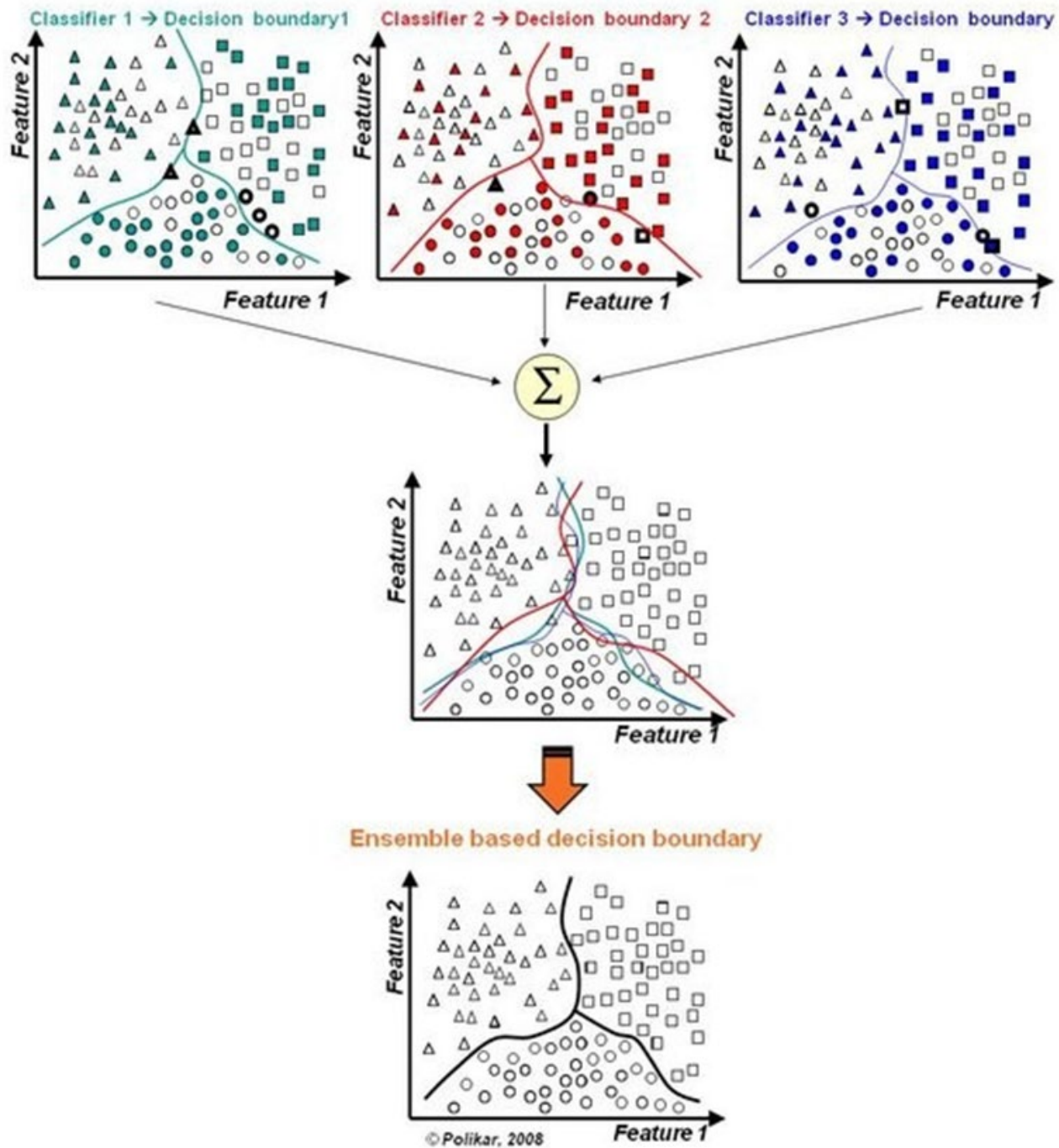
Today: Ensemble

- Framework of Ensemble:
 - Get a set of classifiers $f_1(x), f_2(x), f_3(x)...$ They should be diverse.

How to have different training data sets


- Re-sampling your training data to form a new set
- Re-weighting your training data to form a new set

- Aggregate the classifiers (*properly*)
- For example



Today: Ensemble

- Bagging

- 
- Bagged Decision Tree
 - Random forests

- Stacking

- Boosting

- Adaboost
- Xgboost

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.



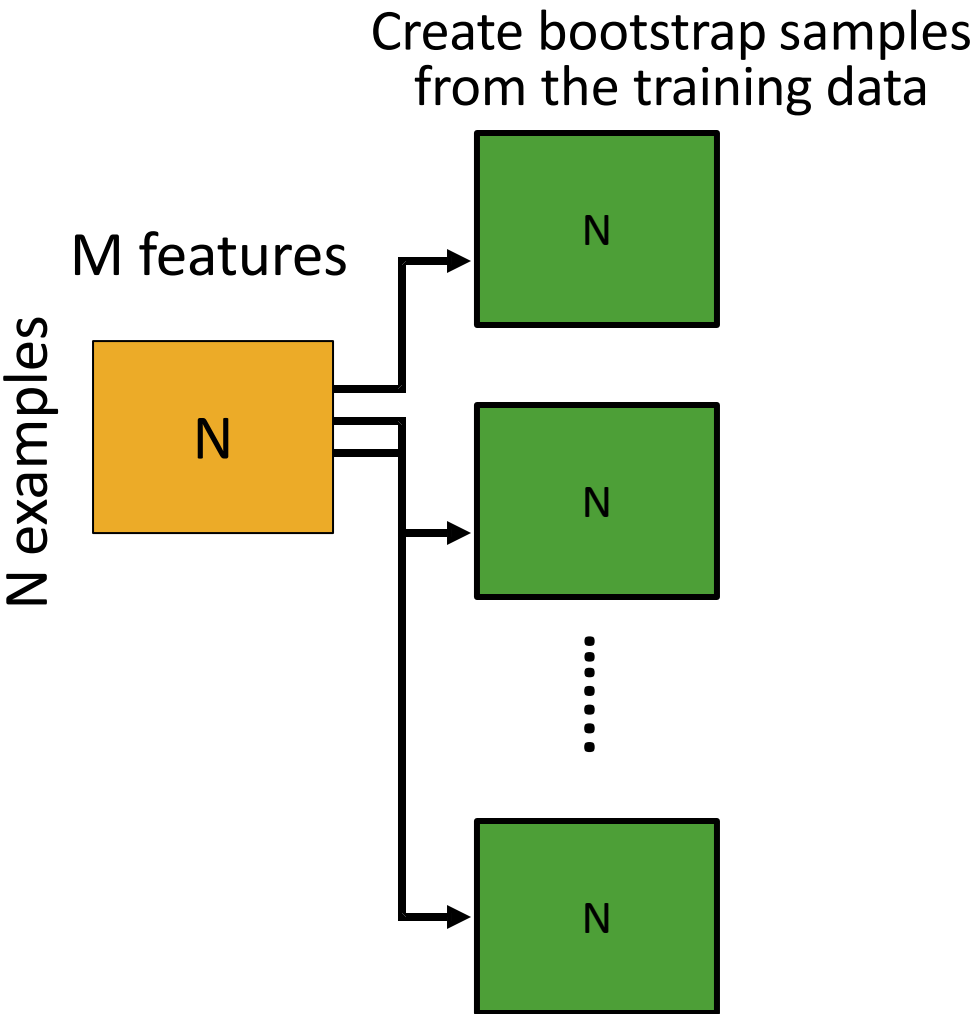
Bootstrap

- 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**

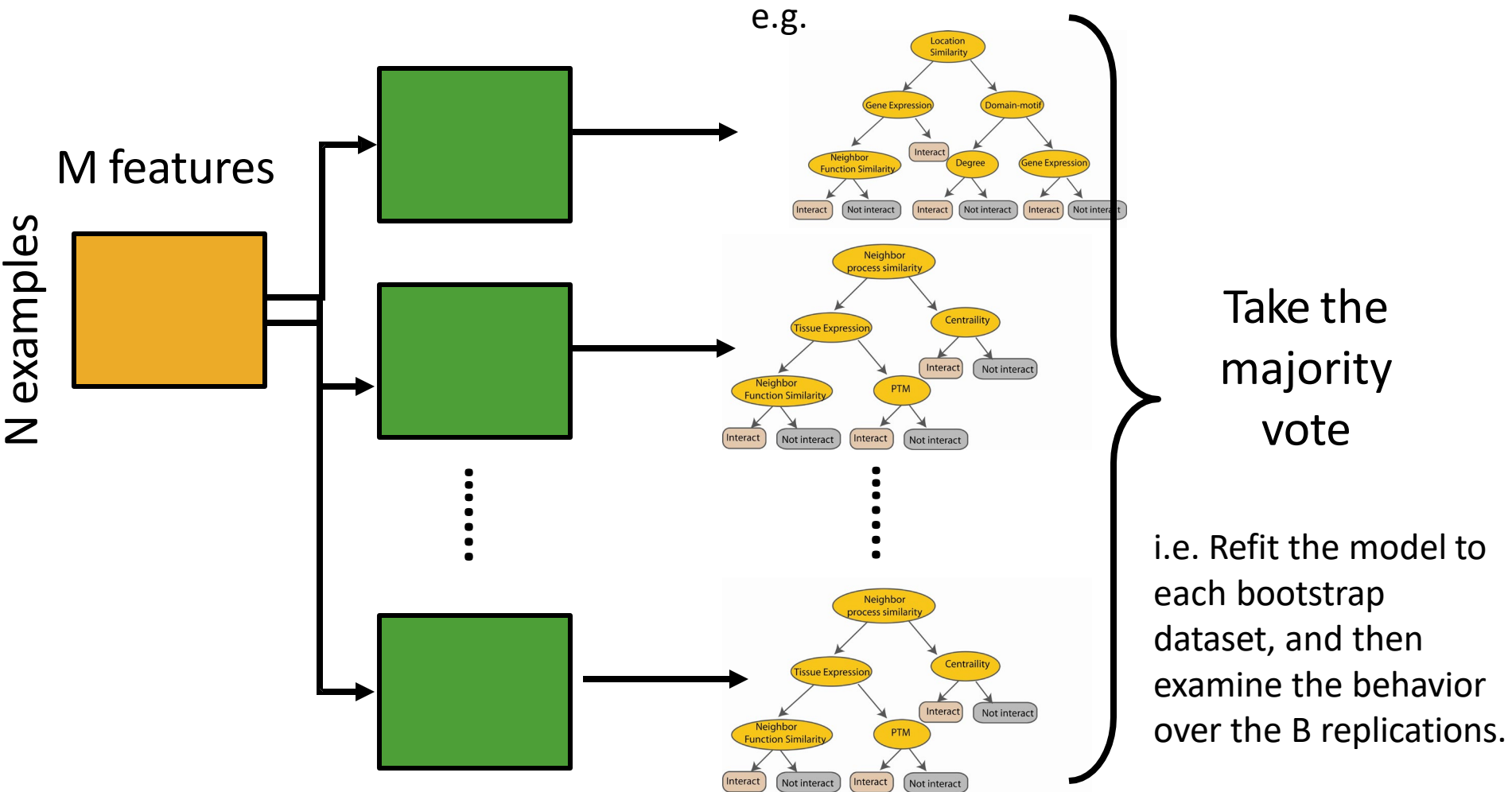
With vs Without Replacement

- **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.
- **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.

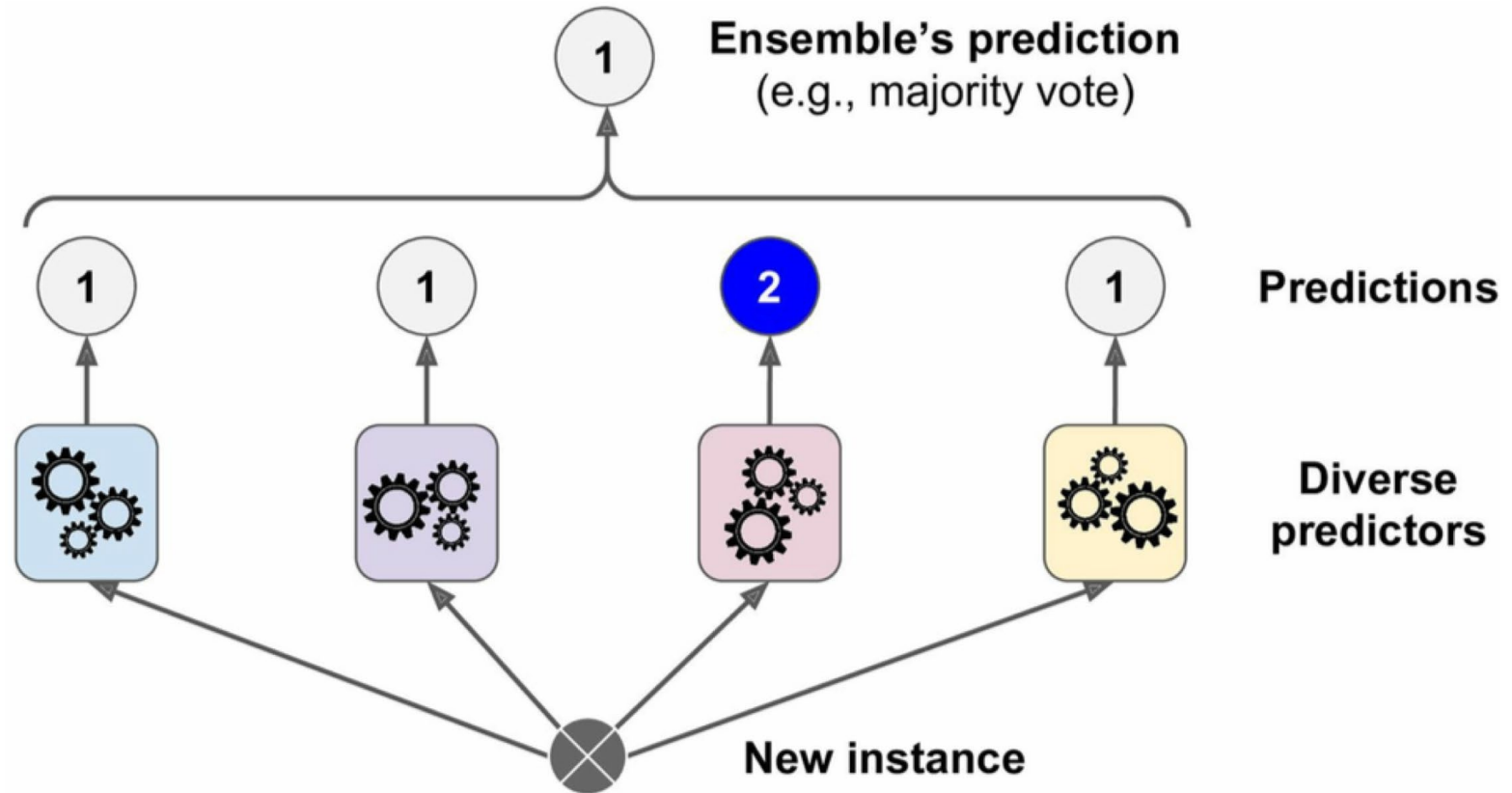
Bagging



Bagging of DT Classifiers



E.g. Predict by Hard voting



Decision Boundary Comparison

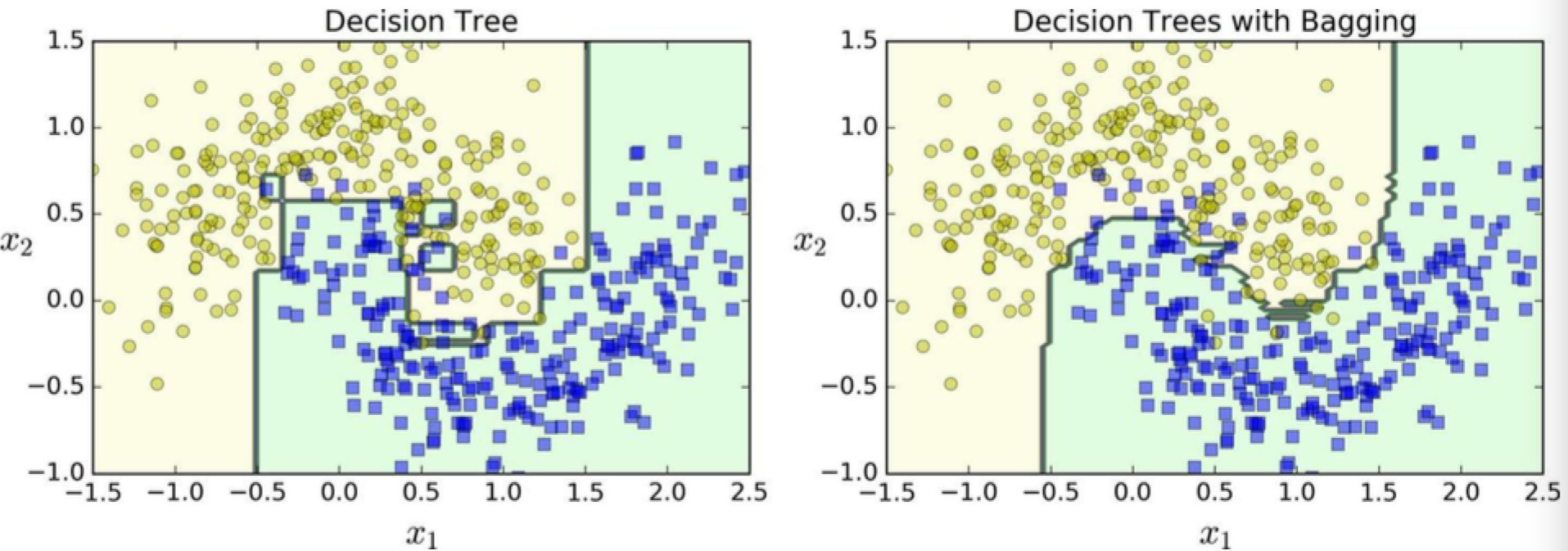
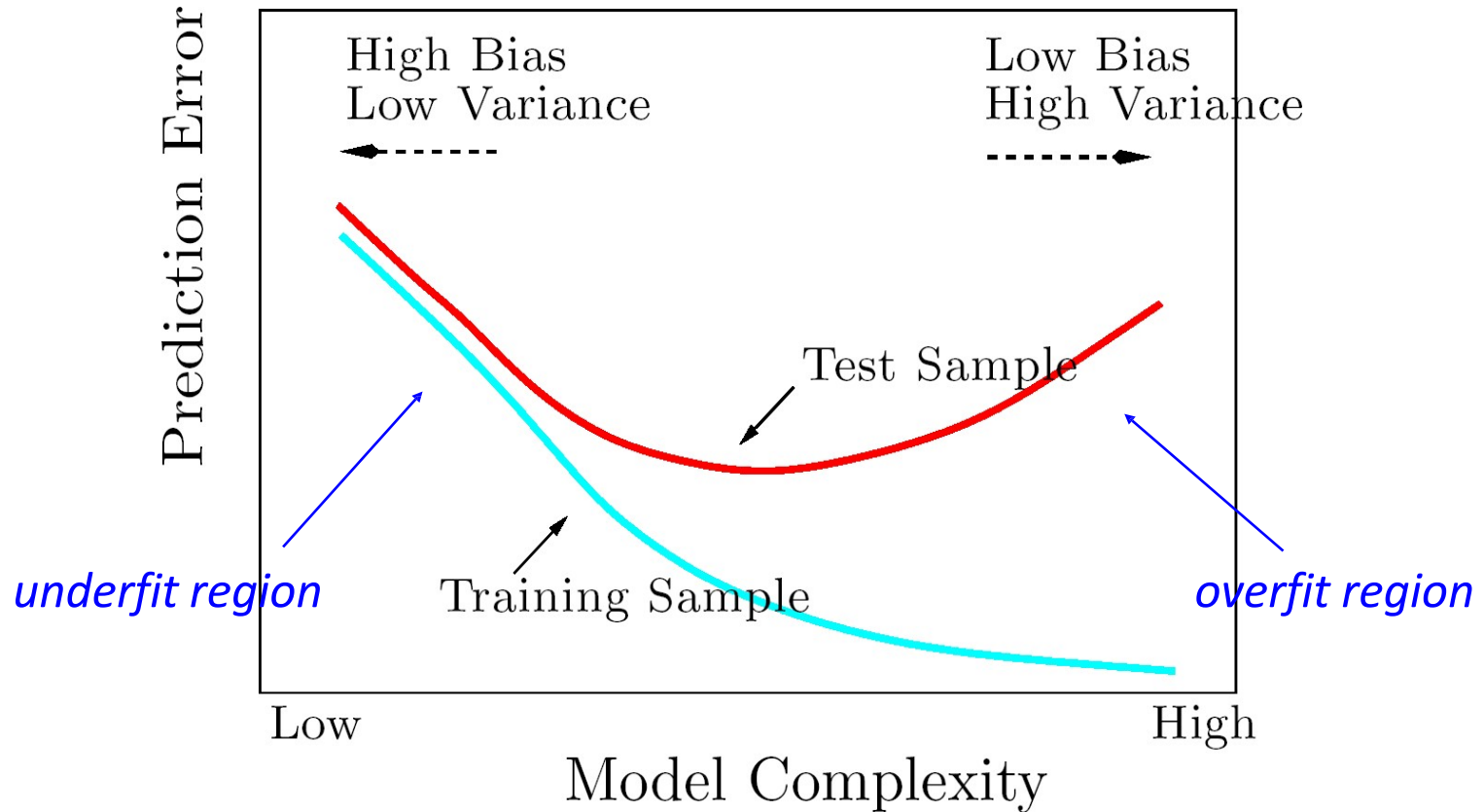


Figure 7-5. A single Decision Tree versus a bagging ensemble of 500 trees

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



Recap: Bias-Variance Tradeoff / Model Selection

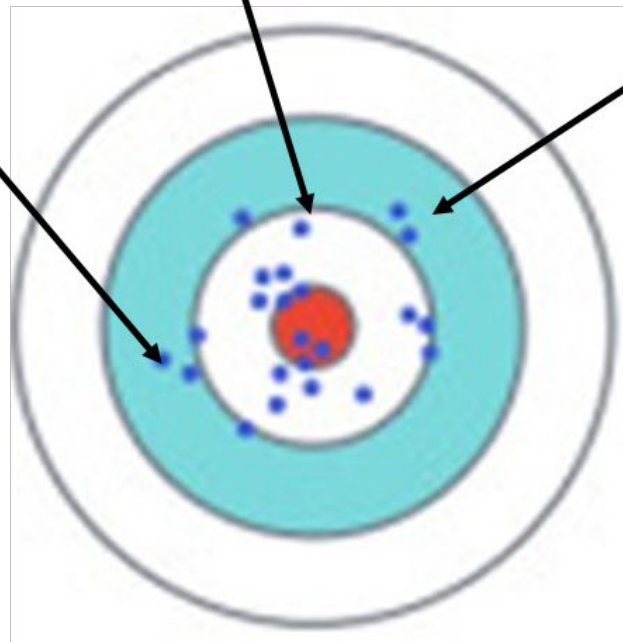
classifiers $f_1(x)$

$f_2(x)$

$f_3(x)...$

$f_B(x)$

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^*

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

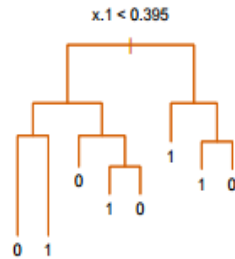
Bagging: an example with simulated data

- $N = 30$ training samples,
- two classes and $p = 5$ features,
- Each feature $N(0, 1)$ distribution and pairwise correlation .95 Response Y generated according to:

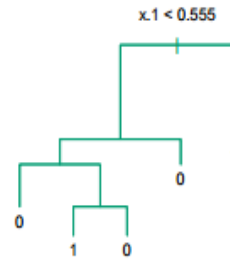
$$\Pr(Y = 1|x_1 \leq 0.5) = 0.2 \quad \Pr(Y = 1|x_1 > 0.5) = 0.8$$

- Test sample size of 2000
- Fit classification trees to training set and bootstrap samples
- $B = 200$

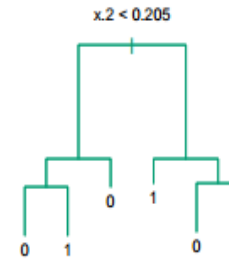
Original Tree



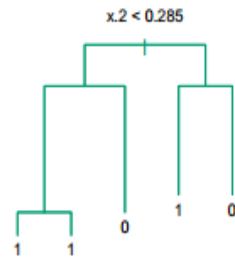
$b = 1$



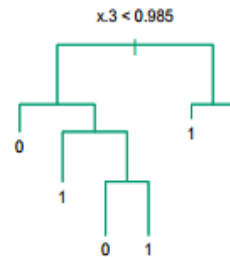
$b = 2$



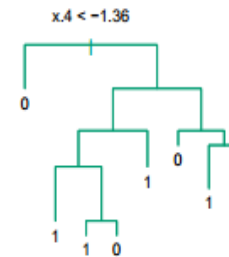
$b = 3$



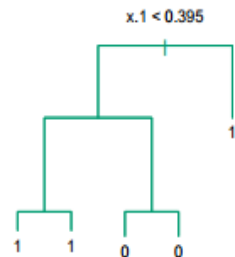
$b = 4$



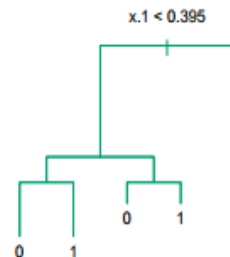
$b = 5$



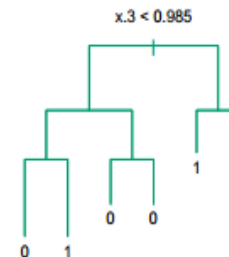
$b = 6$



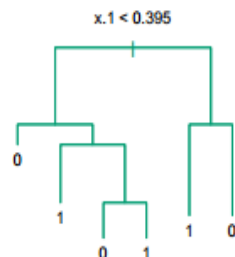
$b = 7$



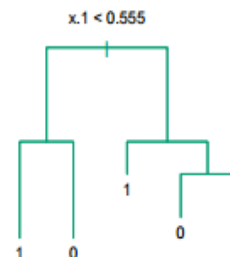
$b = 8$



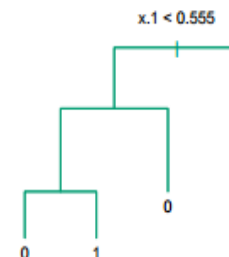
$b = 9$



$b = 10$



$b = 11$



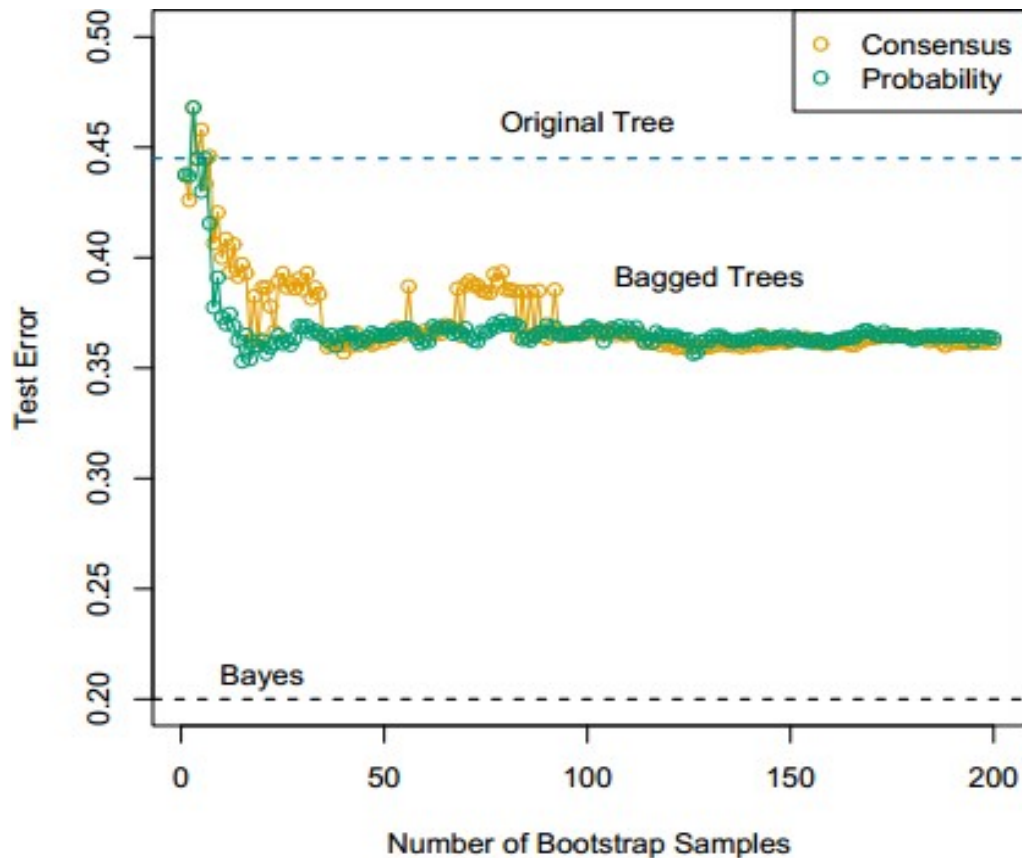
Five features
highly correlated
with each other

→ 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 for
classification

Notice the
bootstrap trees
are different
than the
original tree



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

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

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

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

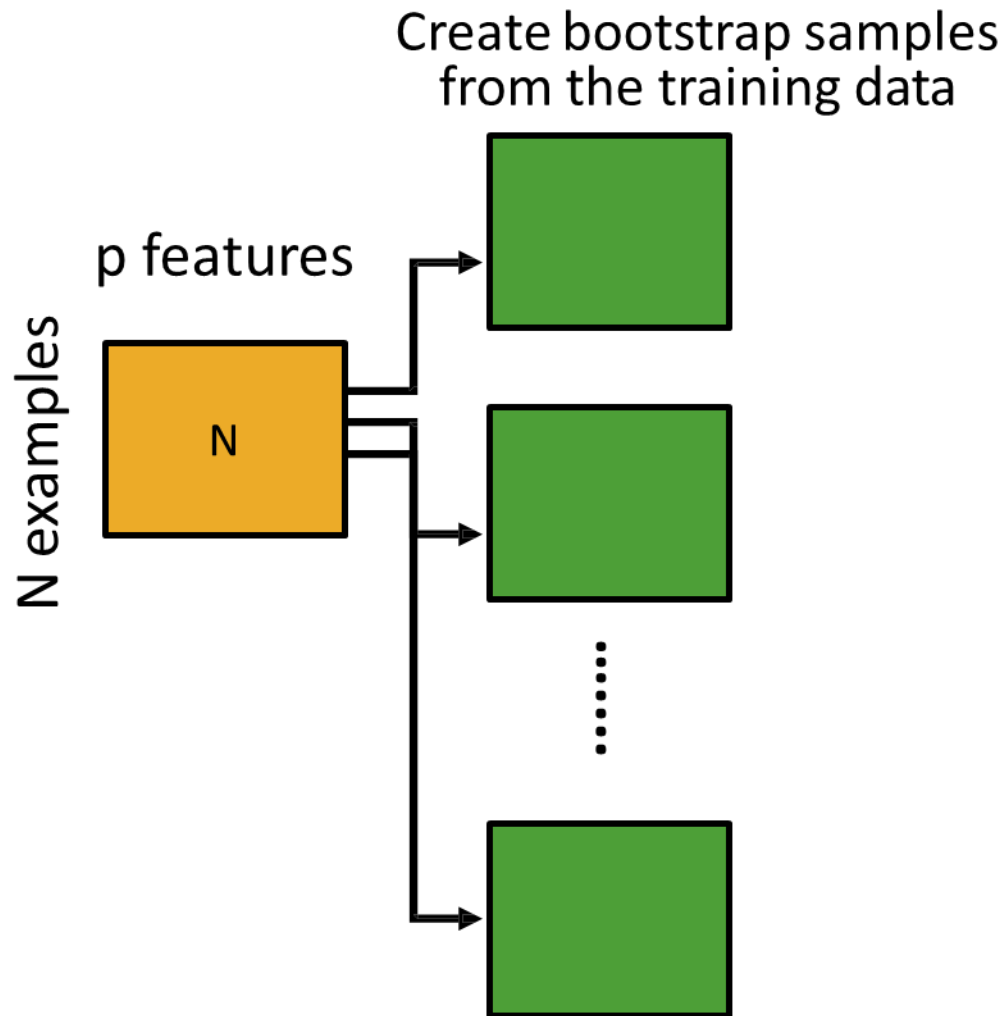
Today: Ensemble

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

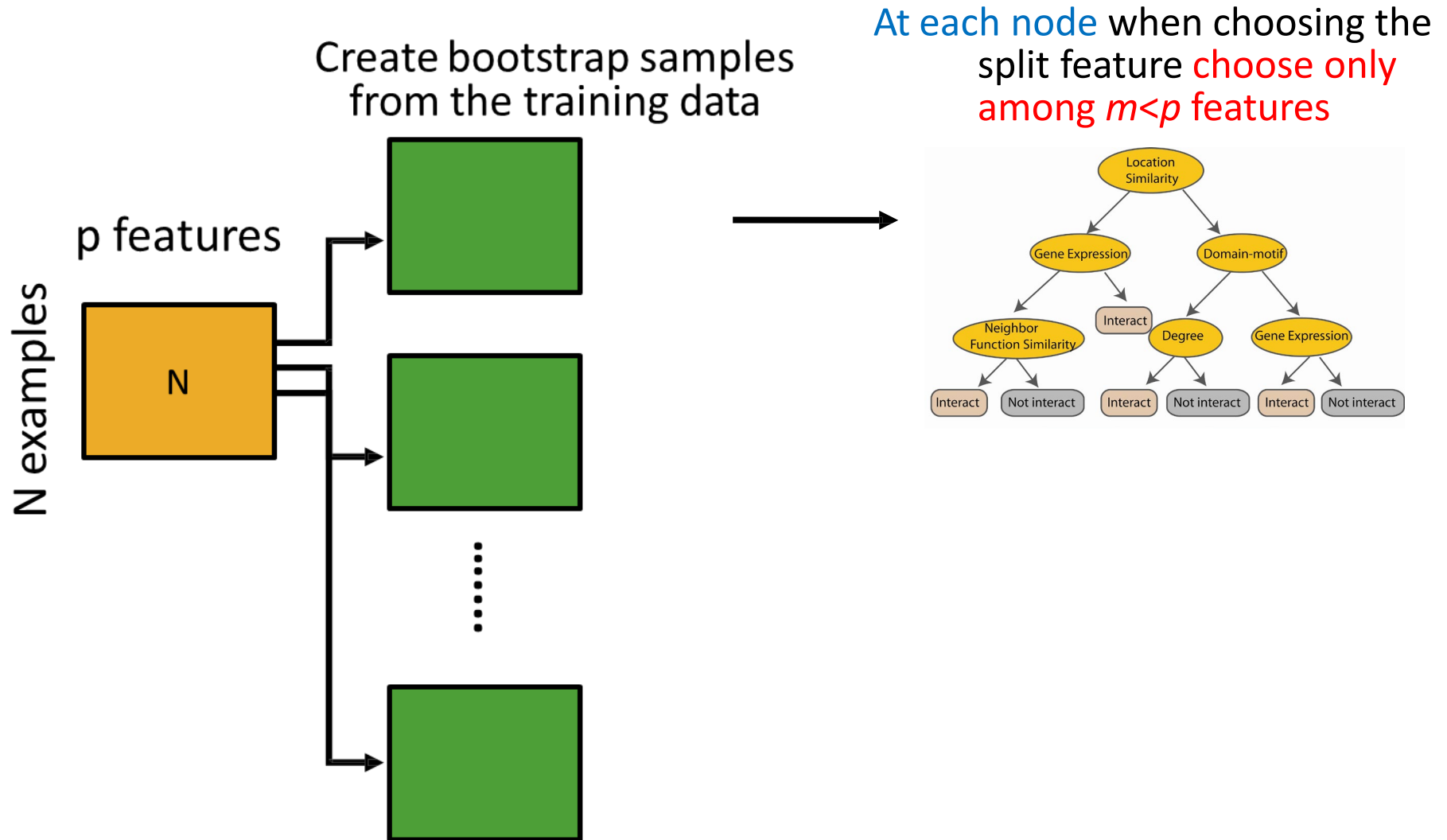
Random forest classifier

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

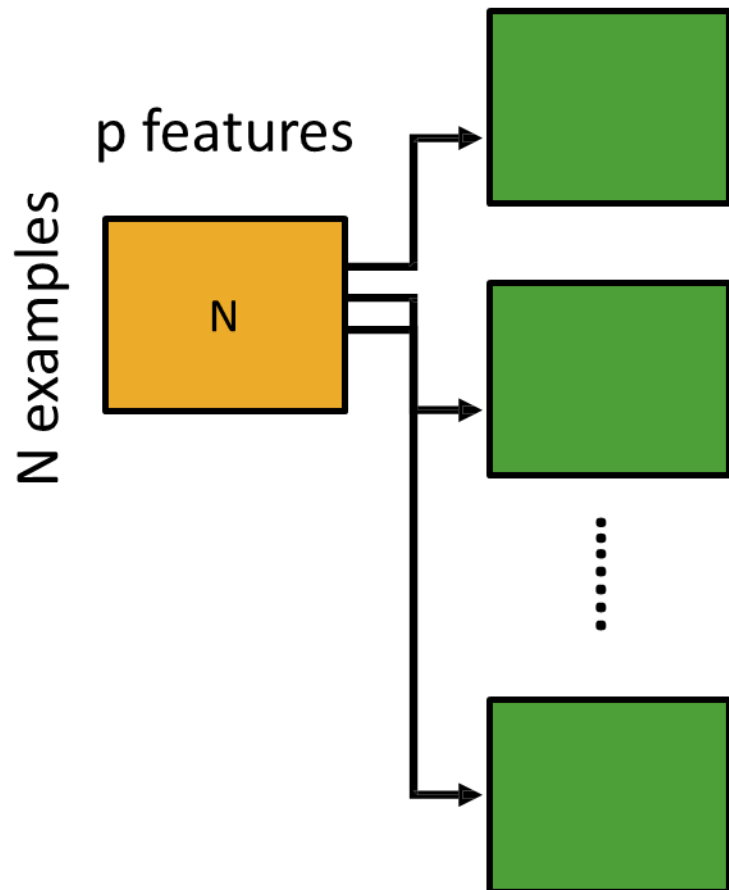
Random Forest Classifier



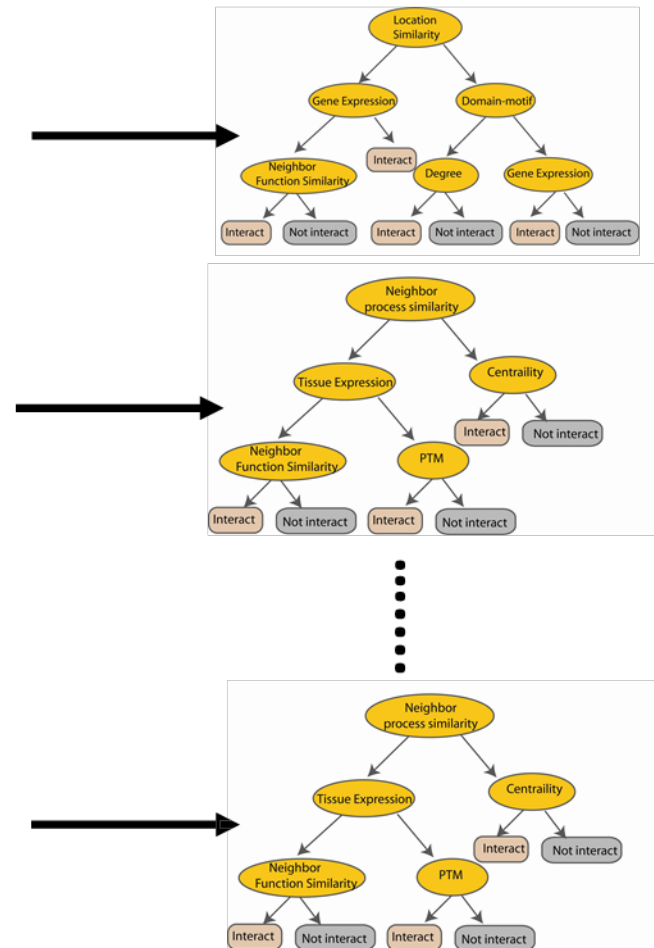
Random Forest Classifier



Random Forest Classifier



Create decision tree
from each bootstrap sample



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

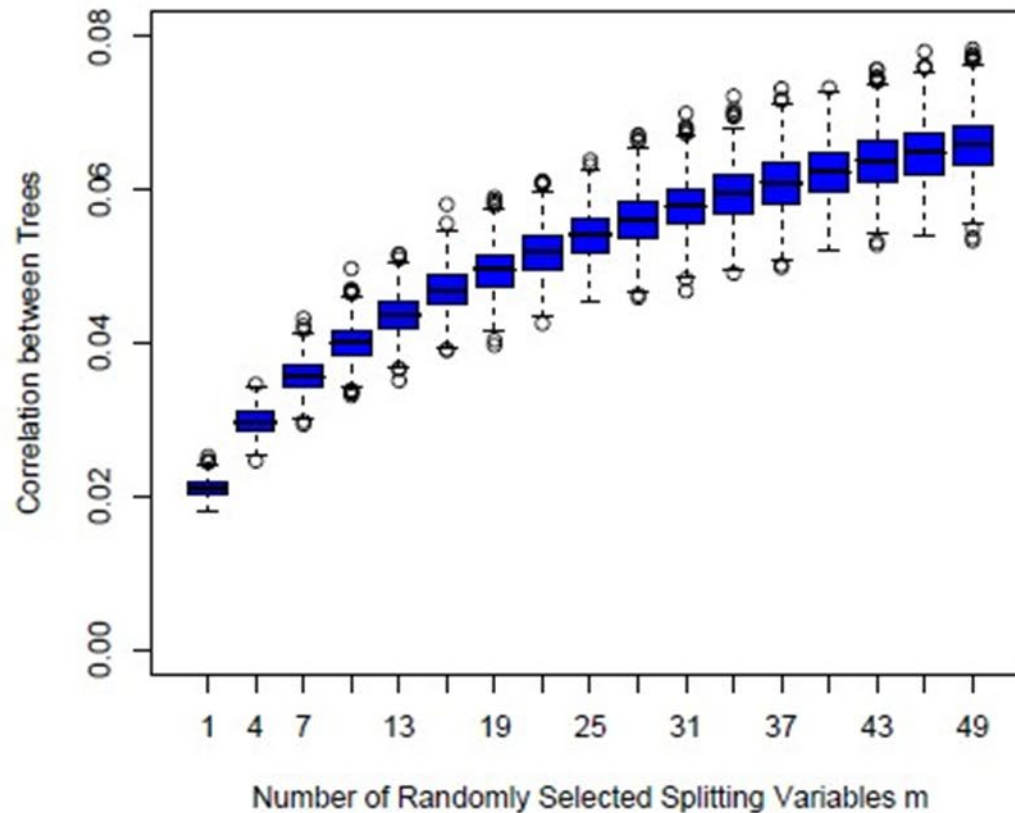


FIGURE 15.9. *Correlations between pairs of trees drawn by a random-forest regression algorithm, as a function of m . The boxplots represent the correlations at 600 randomly chosen prediction points x .*

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 $\log_2(p)$ 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?

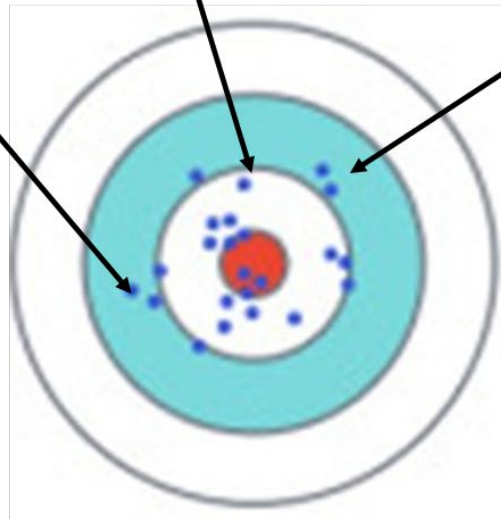
- Assuming each tree has variance σ^2
- If trees are independently identically distributed, then average variance is σ^2/B

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

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

classifiers $f_1(x)$ $f_2(x)$ $f_3(x)...$ $f_B(x)$

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 σ^2
- If simply **identically distributed**, then average variance is

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

- As $B \rightarrow \infty$, second term $\rightarrow 0$
- Thus, the pairwise correlation always affects the variance

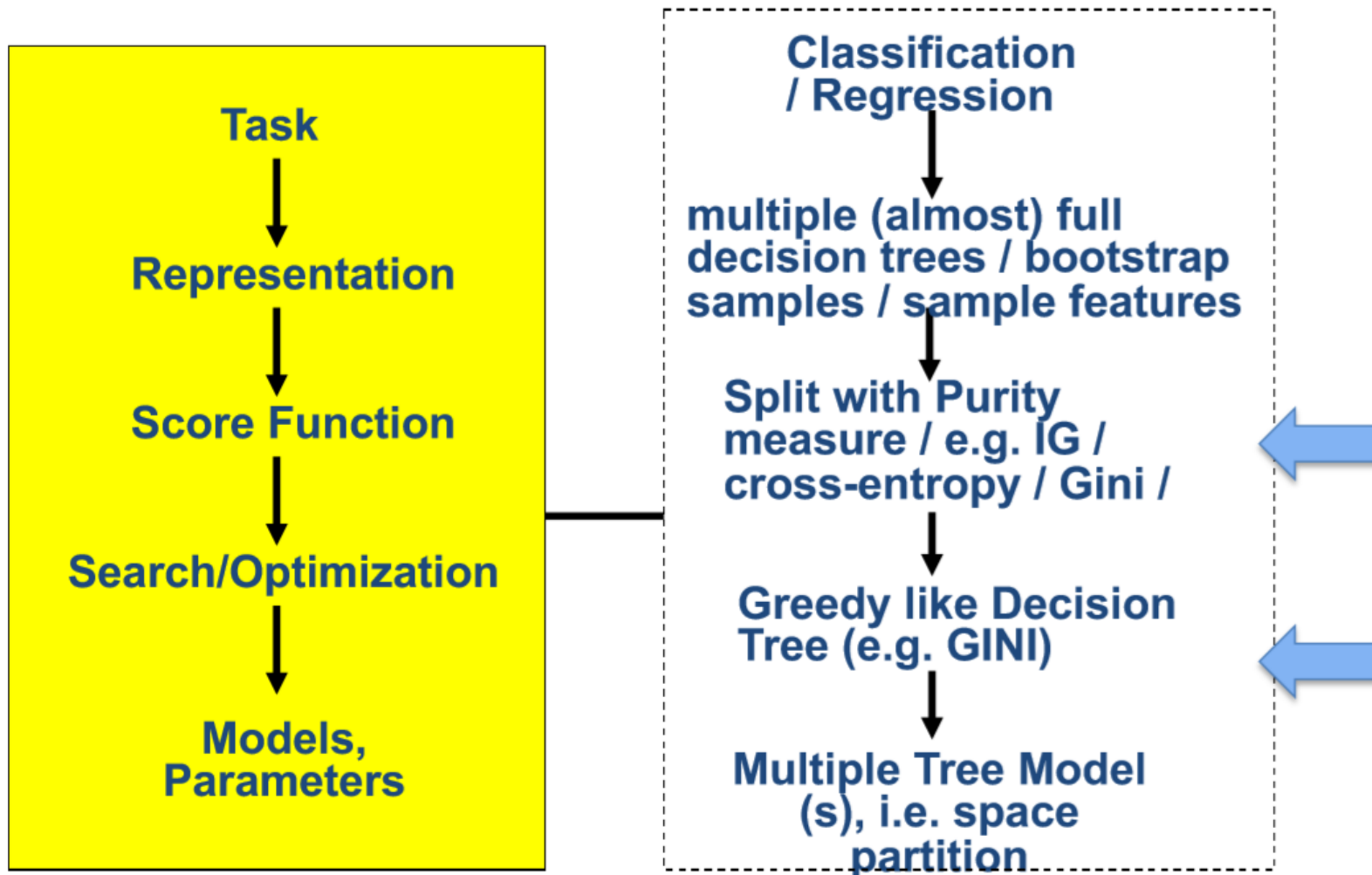
Why correlated trees are not ideal?

- 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.

Random forest



Today: Ensemble

- Bagging
 - Bagged Decision Tree
 - Random forests

• Stacking

- Boosting
 - Adaboost
 - Xgboost

e.g. Ensembles in practice



October 2006-present

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

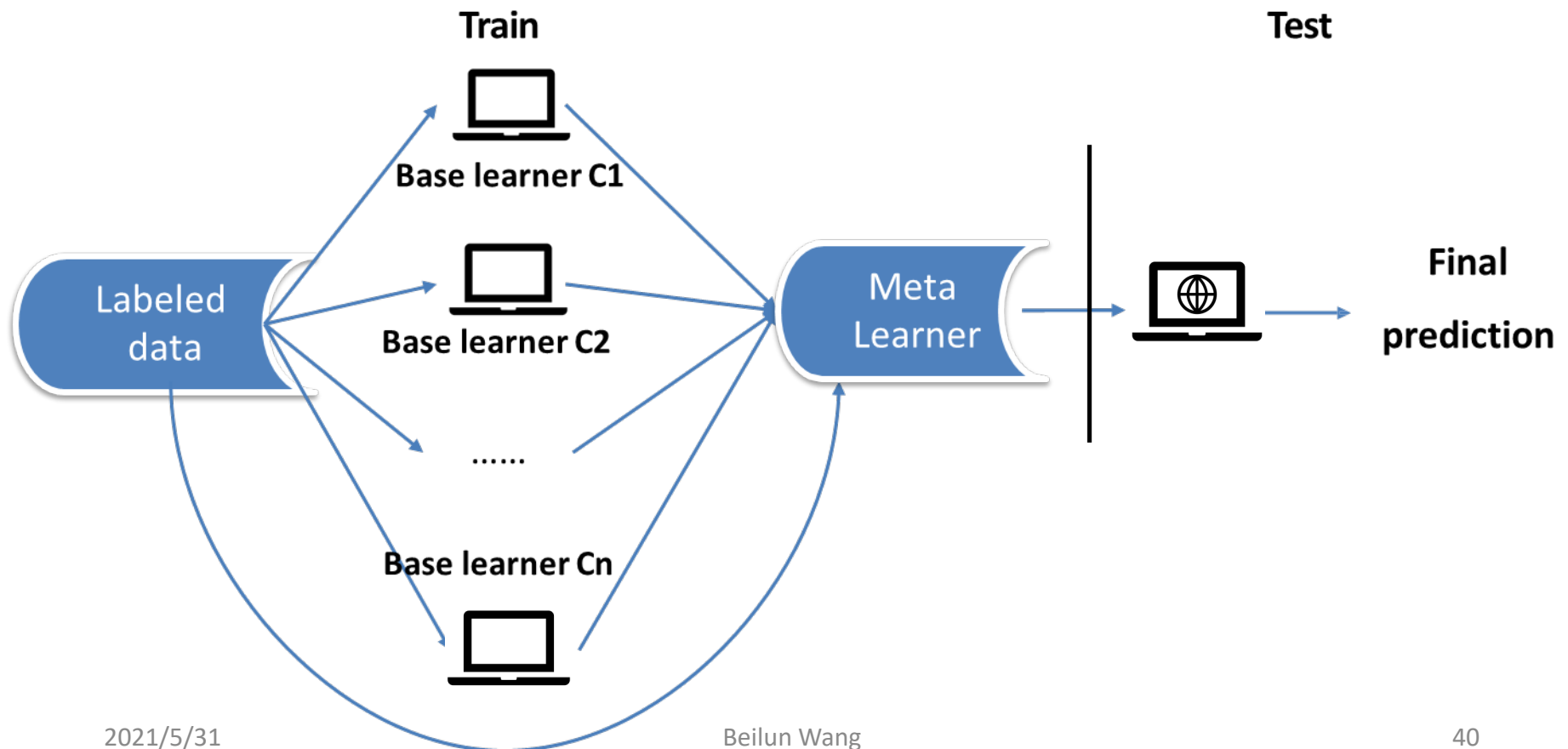
Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera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

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

The ensemble team → blenders of multiple different methods

Stacking

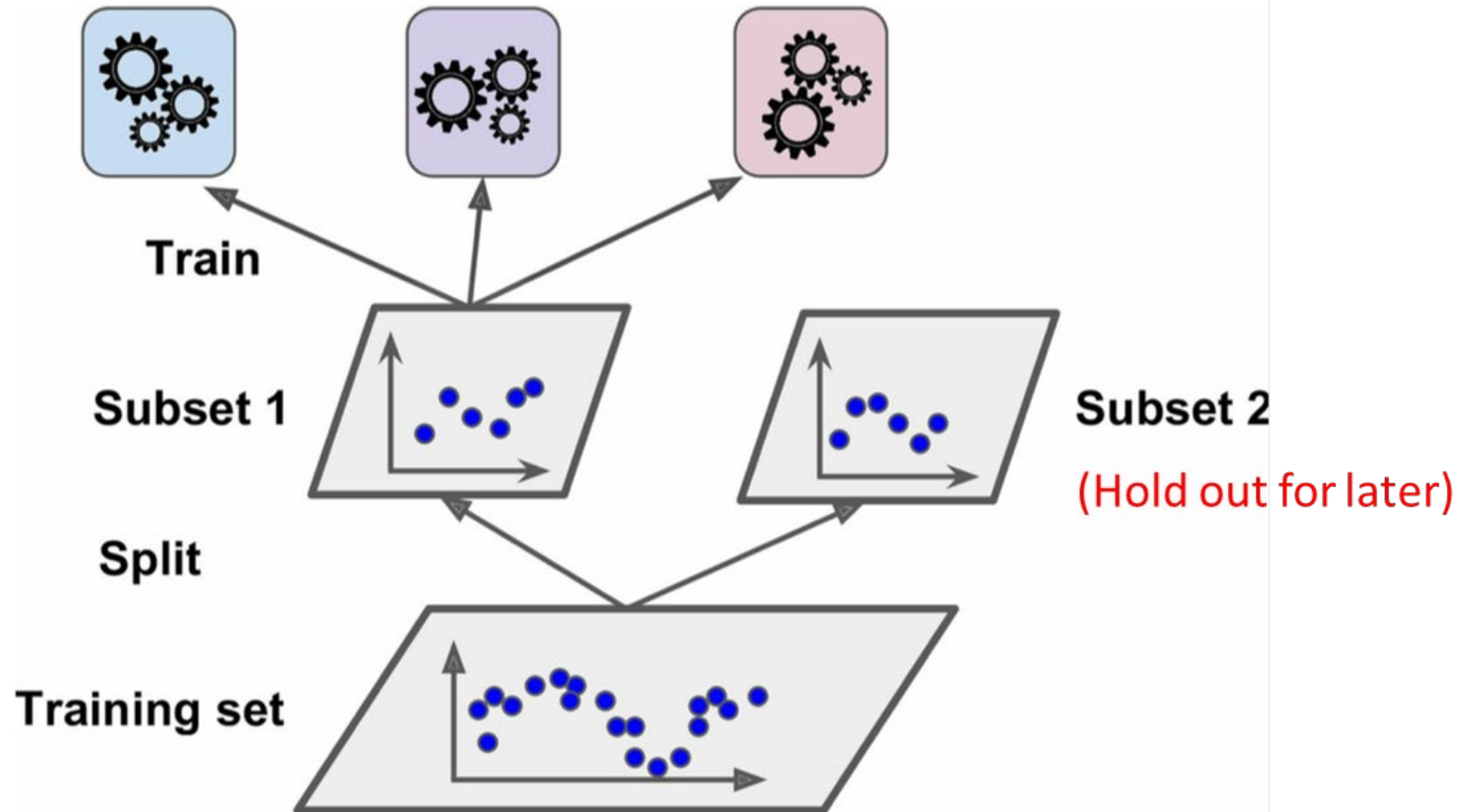
- Main Idea: Learn and combine multiple classifiers



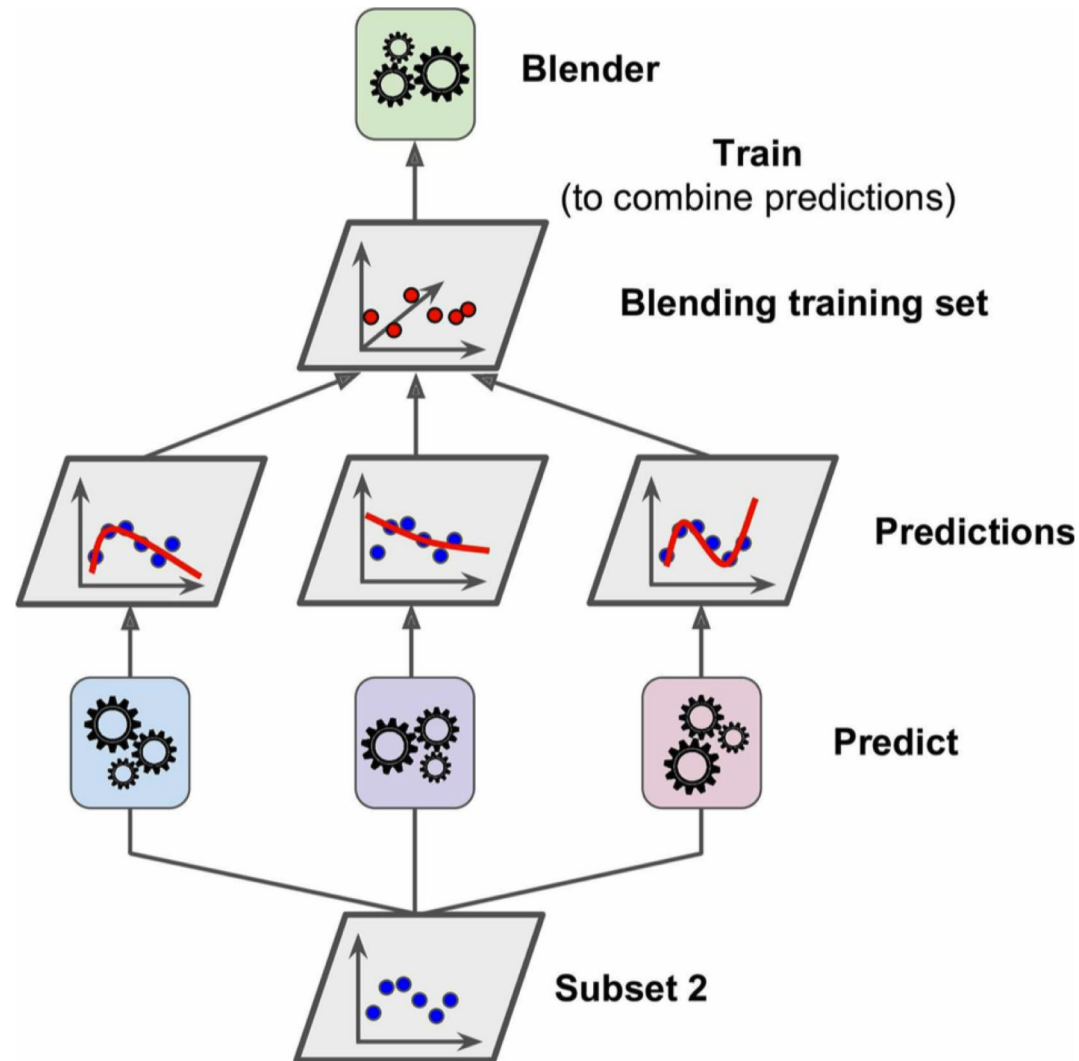
Generating Base and Meta Learners

- 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



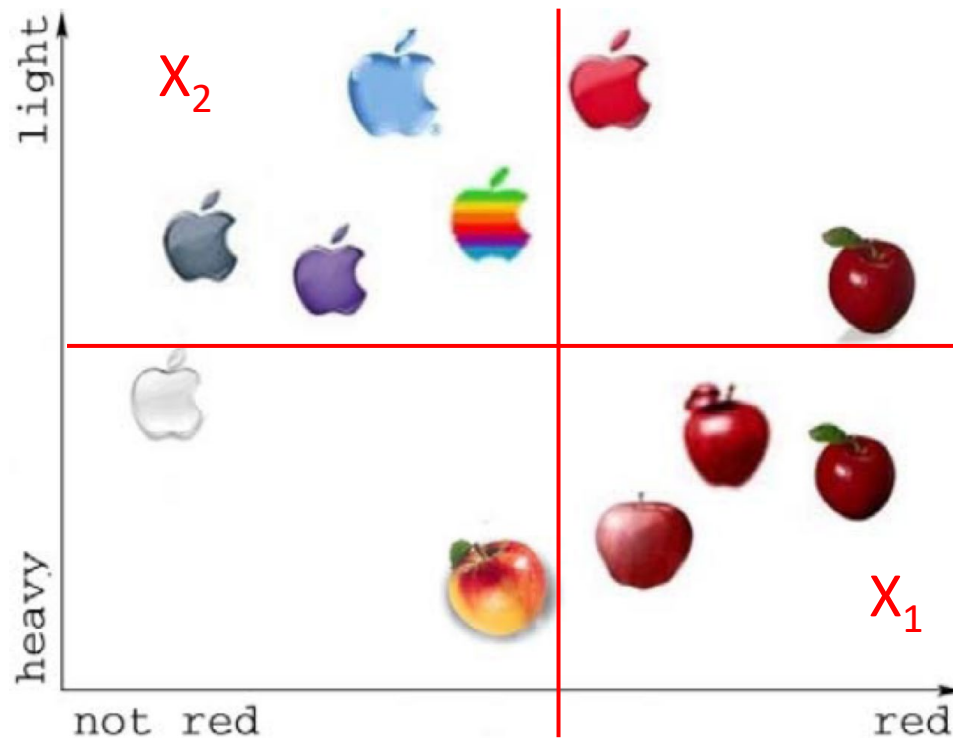
Today: Ensemble

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

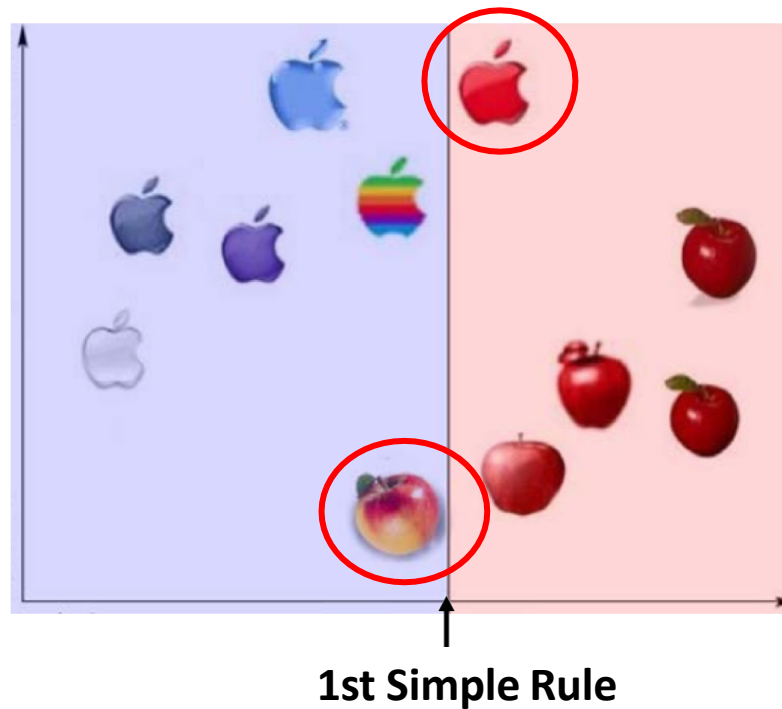
Boosting Strategies

- Have many rules (base classifiers) to **vote** on the decision
- Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
- 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



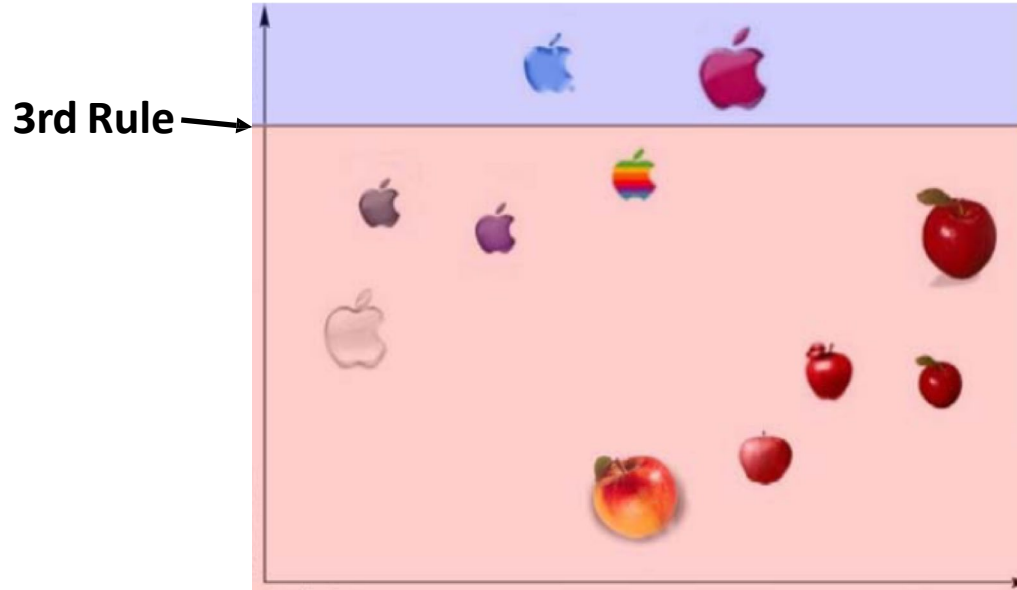
1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



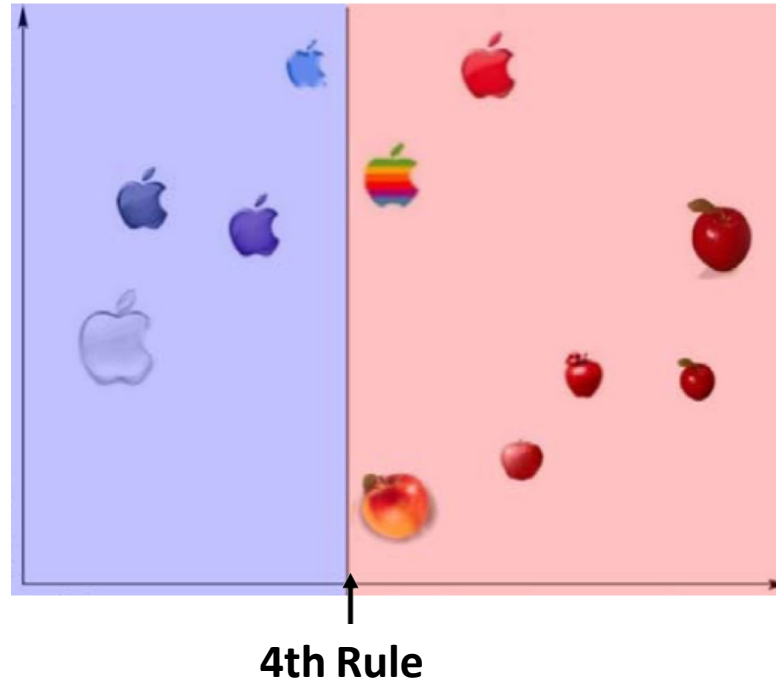
1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



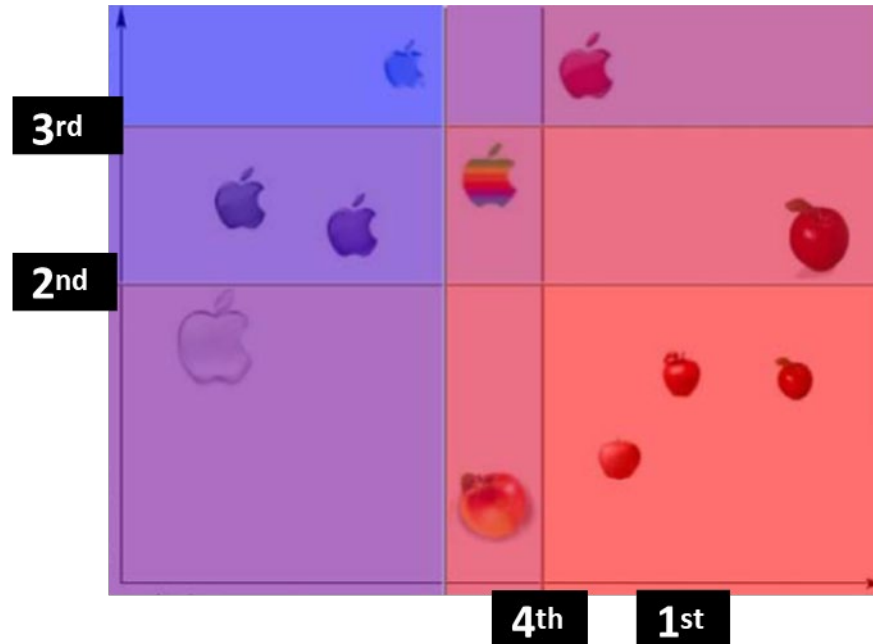
1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



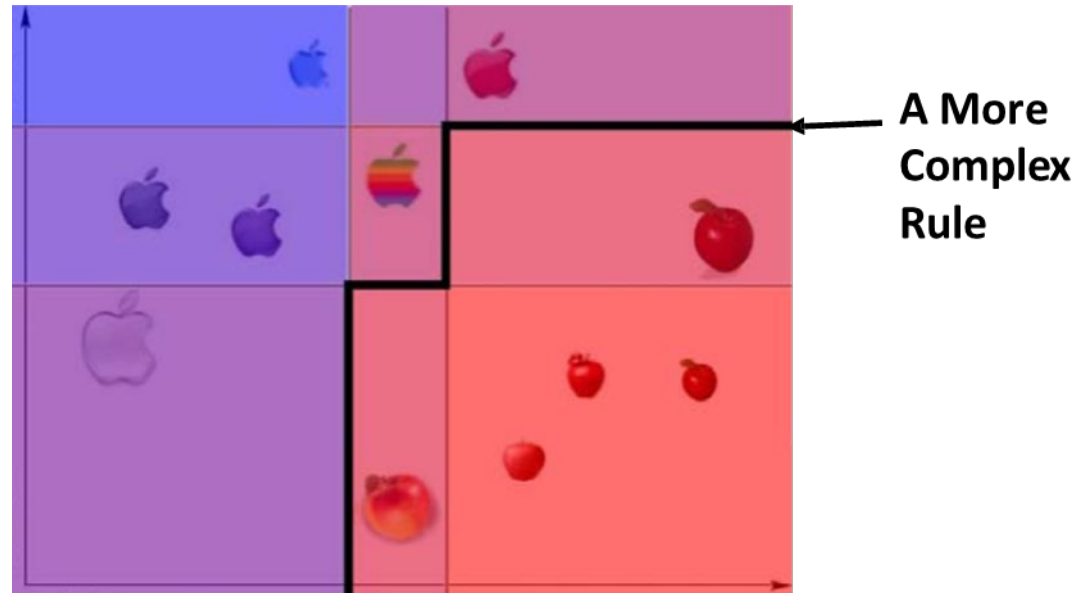
1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



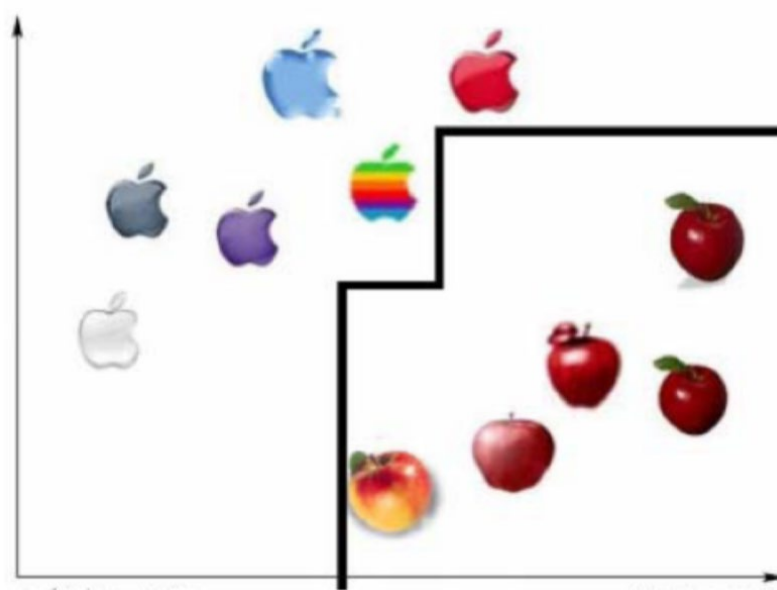
1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



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



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 $w_i, 1 \leq i \leq m$

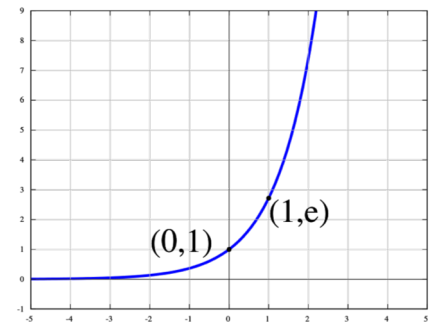
For t = 1 to T iterations:

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



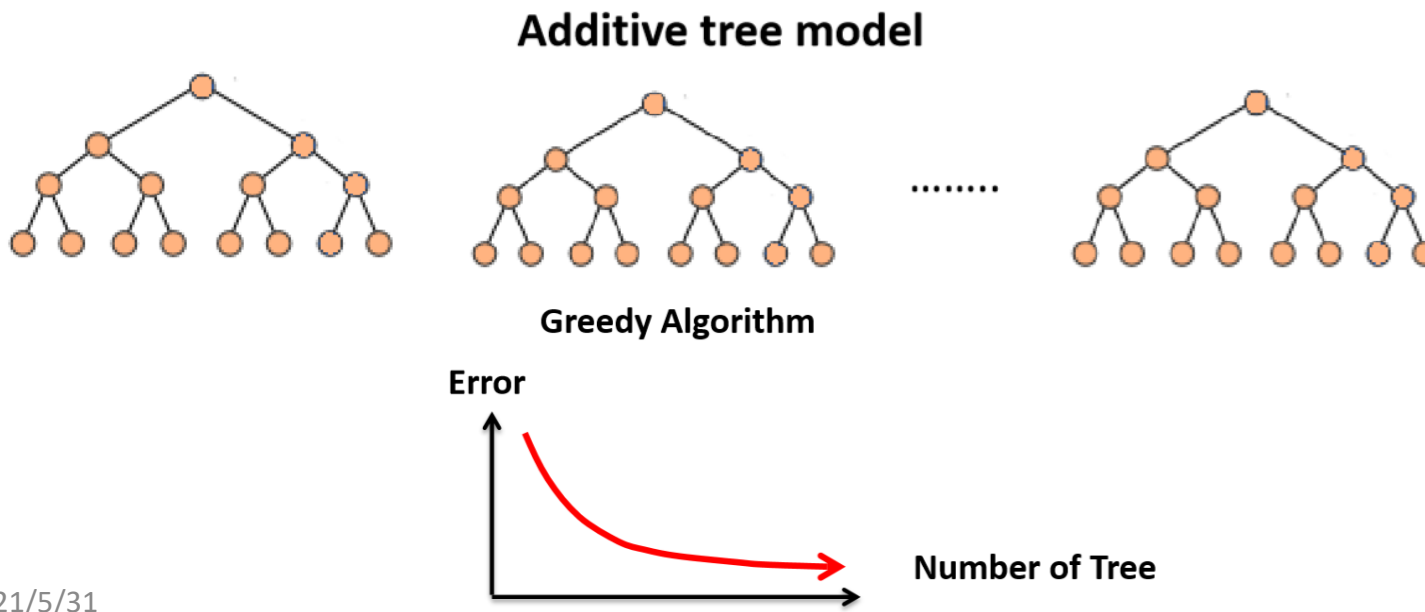
Final Classifier: $\hat{f} = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}_i) \right)$

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



XGBoost

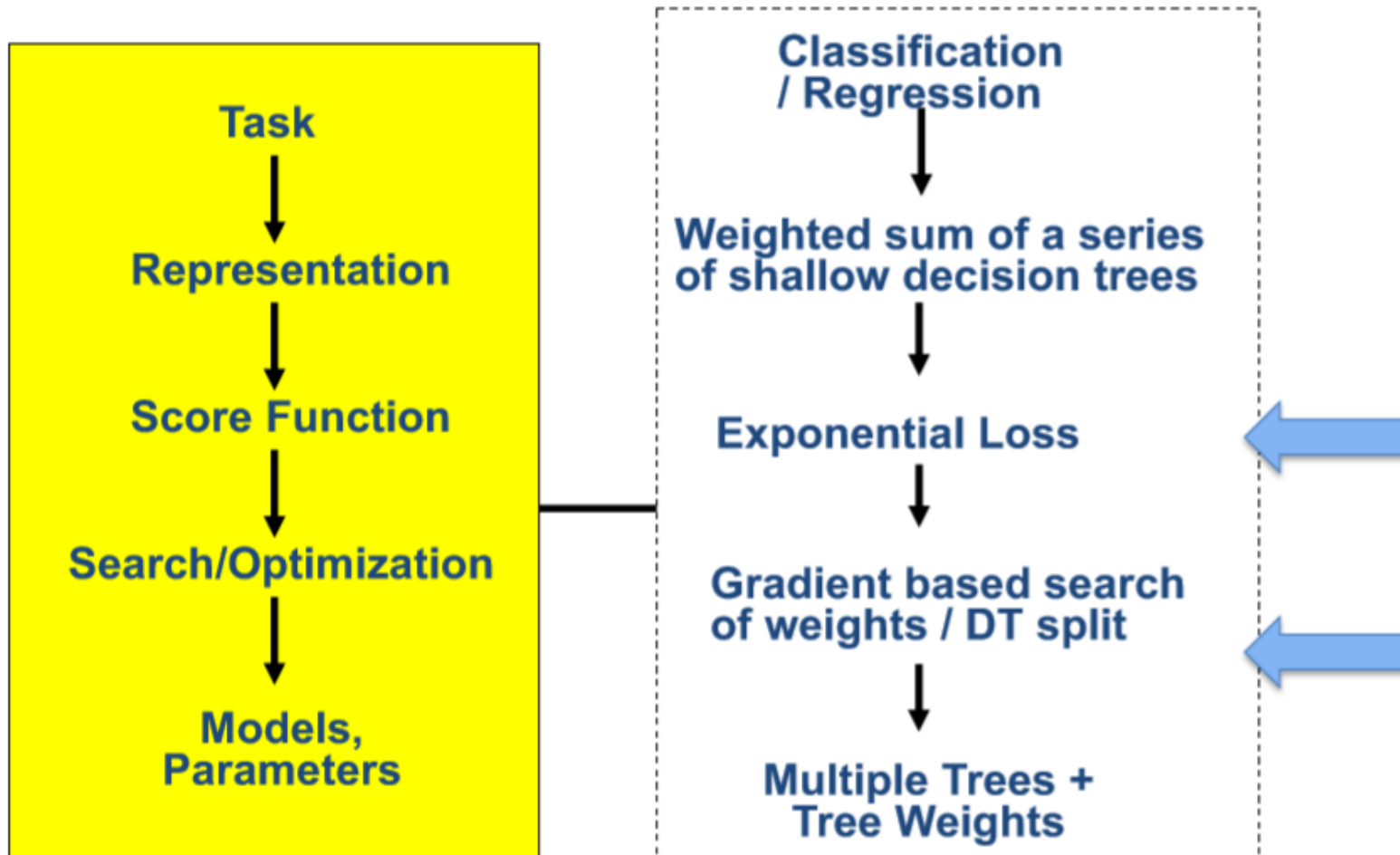
- XGBoost is a very efficient Gradient Boosting Decision Tree implementation with some interesting features:
- **Regularization:** Can use L1 or L2 regularization.
- **Handling sparse data:** Incorporates a sparsity-aware split finding algorithm to handle different types of sparsity patterns in the data.
- **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.



More about History

- Introduction of Adaboost:
 - Freund; Schapire (1999). "A Short Introduction to Boosting"
- Multiclass/Regression
 - Y. Freund, R. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", 1995.
 - Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In Proceedings of the Eleventh Annual Conference on Computational Learning Theory, pages 80–91, 1998.
- Gentle Boost
 - Schapire, Robert; Singer, Yoram (1999). "Improved Boosting Algorithms Using Confidence-rated Predictions".

Boosting



References

- <https://qiyanjun.github.io/2019f-UVA-CS6316-MachineLearning/>
- 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



Thanks for listening