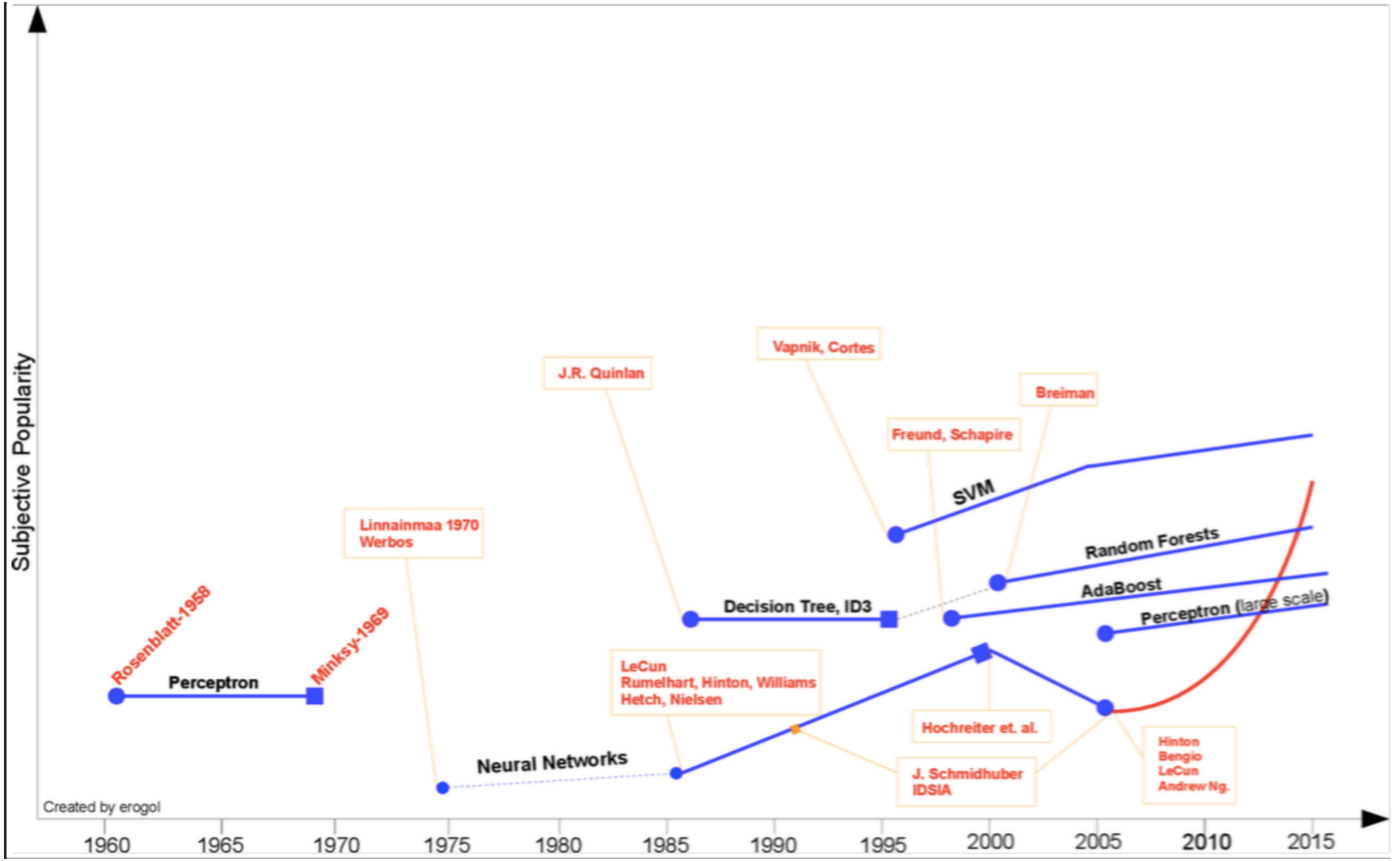


Subjective Popularity
Rosenblatt-1958
Perceptron
Minksy-1969
Linnainmaa 1970
Werbos
Neural Networks
J.R. Quinlan
Decision Tree, ID3
LeCun
Rumelhart, Hinton, Williams
Hetch, Nielsen
Vapnik, Cortes
SVM
Freund, Schapire
Breiman
Random Forests
AdaBoost
Perceptron (large scale)
Hochreiter et. al.
J. Schmidhuber
IDSIA
Hinton
Bengio
LeCun
Andrew Ng.
Created by erogol
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015

Random Forests

The Bootstrap Sample and Bagging

Simple ideas to improve any model via ensemble

Bootstrap Samples

- Random samples of your data *with replacement* that are the same size as original data.
- Some observations will not be sampled. These are called *out-of-bag observations*

<u>Example:</u> Suppose you have 10 observations, labeled 1-10

Bootstrap Sample Number	Training Observations	Out-of-Bag Observations
1	{1,3,2,8,3,6,4,2,8,7}	{5,9,10}
2	{9,1,10,9,7,6,5,9,2,6}	{3,4,8}
3	{8,10,5,3,8,9,2,3,7,6}	{1,4}

(Efron 1983) (Efron and Tibshirani 1986)

Bootstrap Samples

- Can be proven that a bootstrap sample will contain approximately 63% of the observations.
- The sample size is the same as the original data as some observations are repeated.
- Some observations left out of the sample (~37% out-of-bag)

- Uses:
  - Alternative to traditional validation/cross-validation
  - **Create Ensemble Models using different training sets (Bagging)**

Bagging

(Bootstrap Aggregating)

- Let k be the number of bootstrap samples
- For each bootstrap sample, create a classifier using that sample as training data
  - Results in k different models
- Ensemble those classifiers
  - A test instance is assigned to the class that received the highest number of votes.

Bagging Example

input variable

target

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y	1	1	1	-1	-1	-1	-1	1	1	1

- 10 observations in original dataset
- Suppose we build a decision tree with only 1 split.
- The best accuracy we can get is 70%
  - Split at x=0.35
  - Split at x=0.75
- A tree with one split called a **decision stump**

Bagging Example

Let's see how bagging might improve this model:

1. Take 10 Bootstrap samples from this dataset.
2. Build a decision stump for each sample.
3. Aggregate these rules into a voting ensemble.
4. Test the performance of the voting ensemble on the whole dataset.

Bagging Example
Classifier 1

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Best decision stump splits at x=0.35

First bootstrap sample:

Some observations chosen multiple times.

Some not chosen.

Bagging Example

Classifiers 1-5

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

x <= 0.65 ==> y = 1
x > 0.65 ==> y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ==> y = 1
x > 0.3 ==> y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Example

Classifiers 6-10

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ==> y = -1
x > 0.05 ==> y = 1

Bagging Example

Predictions from each Classifier

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1
True Class	1	1	1	-1	-1	-1	-1	1	1	1

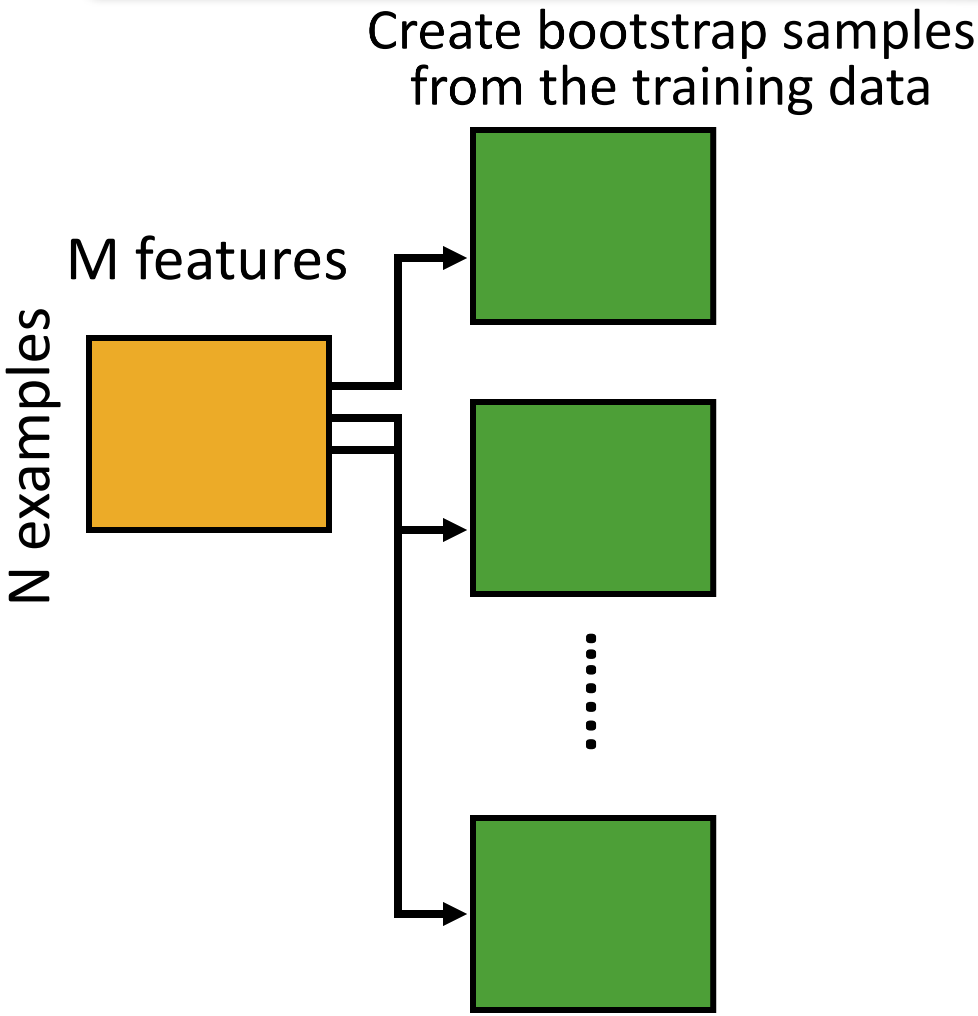
Ensemble Classifier has 100% Accuracy

Bagging

- Bagging or *bootstrap aggregation* a technique for reducing the variance of an estimated prediction function.

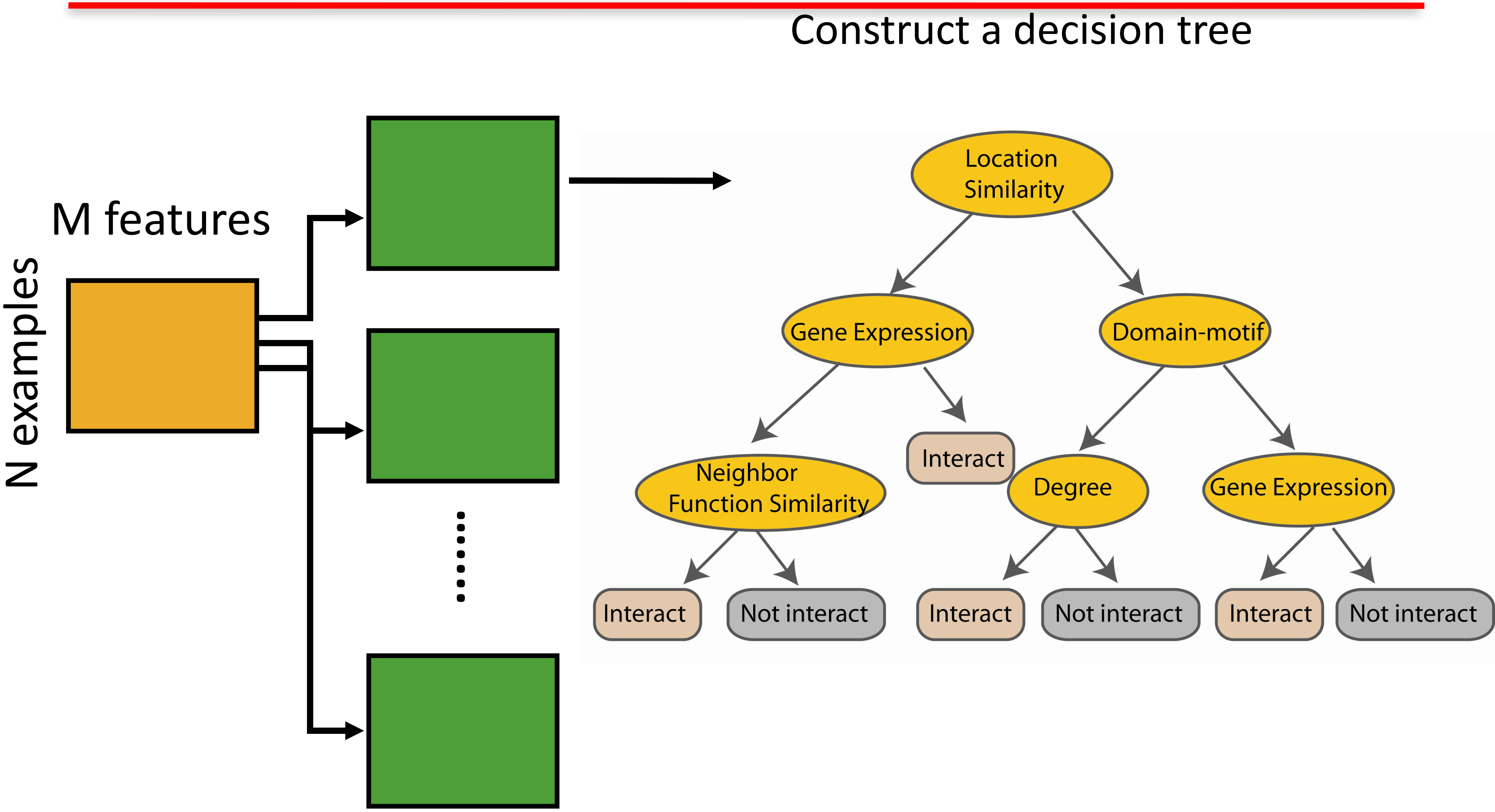
- For classification, a *committee* of trees each cast a vote for the predicted class.

Bagging

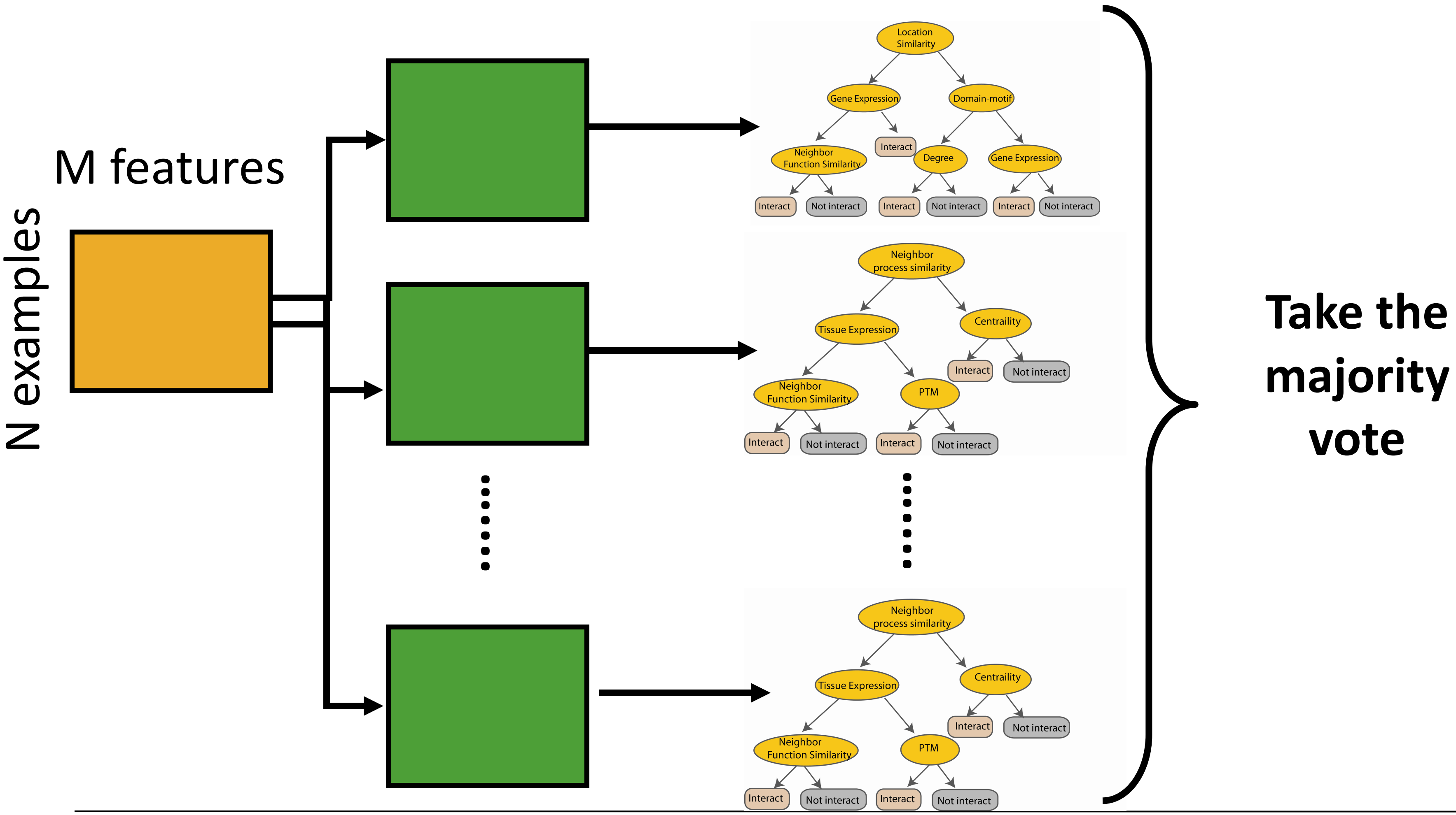


Create bootstrap samples
from the training data
M features
N examples

Bagging


Construct a decision tree
M features
N examples
Location Similarity
Gene Expression
Domain-motif
Neighbor Function Similarity
Interact
Degree
Gene Expression
Interact
Not interact
Interact
Not interact
Interact
Not interact

Bagging


M features
N examples
Location Similarity
Gene Expression
Domain-motif
Neighbor Function Similarity
Interact
Degree
Gene Expression
Interact
Not interact
Interact
Not interact
Interact
Not interact
Neighbor process similarity
Tissue Expression
Centraility
Interact
Not interact
Neighbor Function Similarity
PTM
Interact
Not interact
Interact
Not interact
Neighbor process similarity
Tissue Expression
Centraility
Interact
Not interact
Neighbor Function Similarity
PTM
Interact
Not interact
Interact
Not interact
Take the majority vote

Bagging Summary

- Improves generalization error on models with high variance
- Bagging helps reduce errors associated with random fluctuations in training data (high variance)
- If base classifier is stable (not suffering from high variance), bagging can actually make it worse

- Bagging does not focus on any particular observations in the training data (unlike boosting)

Random Forests

Tin Kam Ho (1995, 1998)

Leo Breiman (2001)

Random Forests

- Random Forests are ensembles of decision trees similar to the one we just saw

- Ensembles of decision trees work best when their predictions are not correlated – they each find different patterns in the data

- Problem: Bagging tends to create correlated trees
- Two Solutions: (a) Randomly subset features considered for each split. (b) Use unpruned decision trees in the ensemble.

Random Forests

- A collection of unpruned decision or regression trees.
- Each tree is build on a bootstrap sample of the data ***and*** a subset of features are considered at each split.
  - The number of features considered for each split is a parameter called $mtry$.
  - Brieman (2001) suggests $mtry = \sqrt{p}$ where p is the number of features
  - I'd suggest setting $mtry$ equal to 5-10 values evenly spaced between 2 and p and choosing the parameter by validation
  - Overall, the model is relatively insensitive to values for $mtry$.
- The results from the trees are ensembled into one voting classifier.

Random Forests

Based on slides by Oznur Tastan et.al

Basic idea of Random Forests

Grow a forest of many trees.

Each tree is a little different (slightly different data, different choices of predictors).

Combine the trees to get predictions for new data.

Idea: most of the trees are good for most of the data and make mistakes in different places.

tree A — $p = \text{accuracy} = \frac{2}{3}$

tree B — $p = \frac{2}{3}$

tree C — $p = \frac{2}{3}$

majority vote: what's chance that none of the three trees or just one of the trees predict correctly?

$(1-p) = \frac{1}{3}$, $(1-p)\ \frac{1}{3}$, $p\ \frac{2}{3}$, $(1-p)\ \frac{1}{3}$ ...

$(1-p)\ \frac{1}{3}$, $(1-p)\ \frac{1}{3}$, $(1-p)\ \frac{1}{3}$, $p\ \frac{2}{3}$

$(1-p)\ \frac{1}{3}$ $\frac{1}{3^3}$, $p\ \frac{2}{3}$ $\frac{2}{3^3}$, $(1-p)\ \frac{1}{3}$ $\frac{2}{3^3}$, $(1-p)\ \frac{1}{3}$ $\frac{2}{3^3}$

$\Sigma = \frac{7}{27}$

or: 2 or 3 trees predict correctly

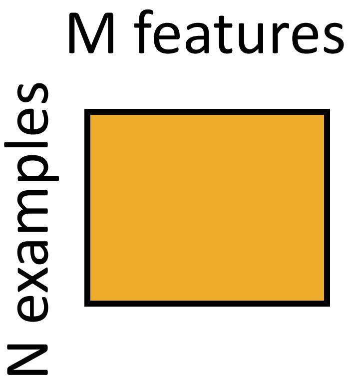
$$C_3^2 \cdot p^2(1-p) + p^3 = 3 \times \frac{2^2}{3^2} \times \frac{1}{3} + \frac{2^3}{3^3} = \frac{20}{27} > \frac{18}{27} = \frac{2}{3}$$

Random forest classifier

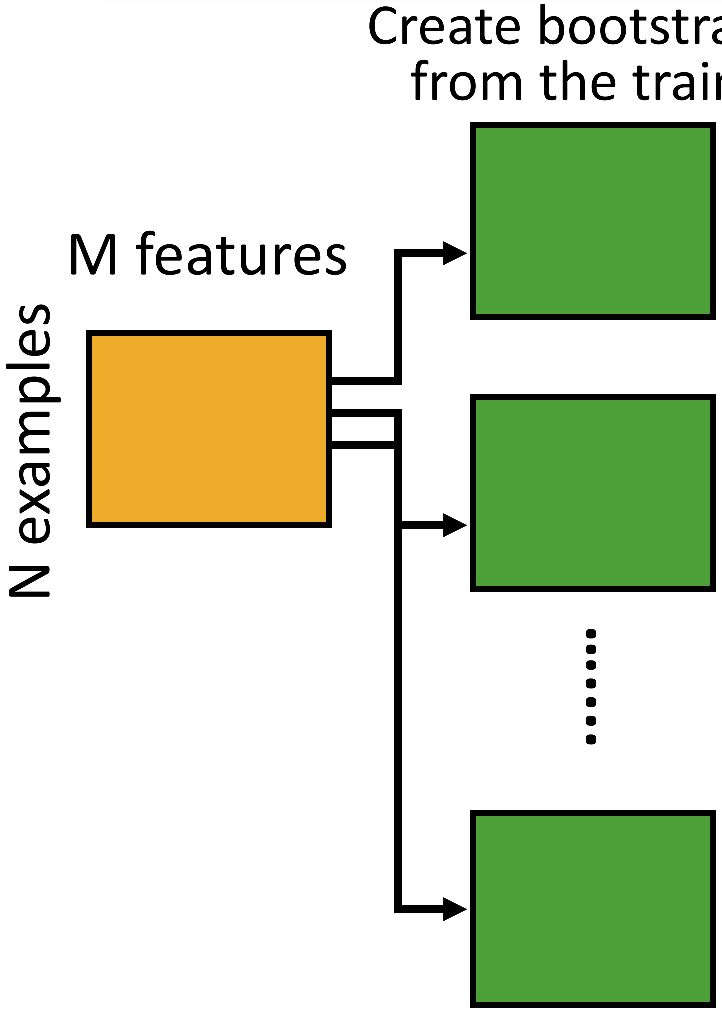
Random forest classifier, an extension to bagging which uses *de-correlated* trees.

Random Forest Classifier

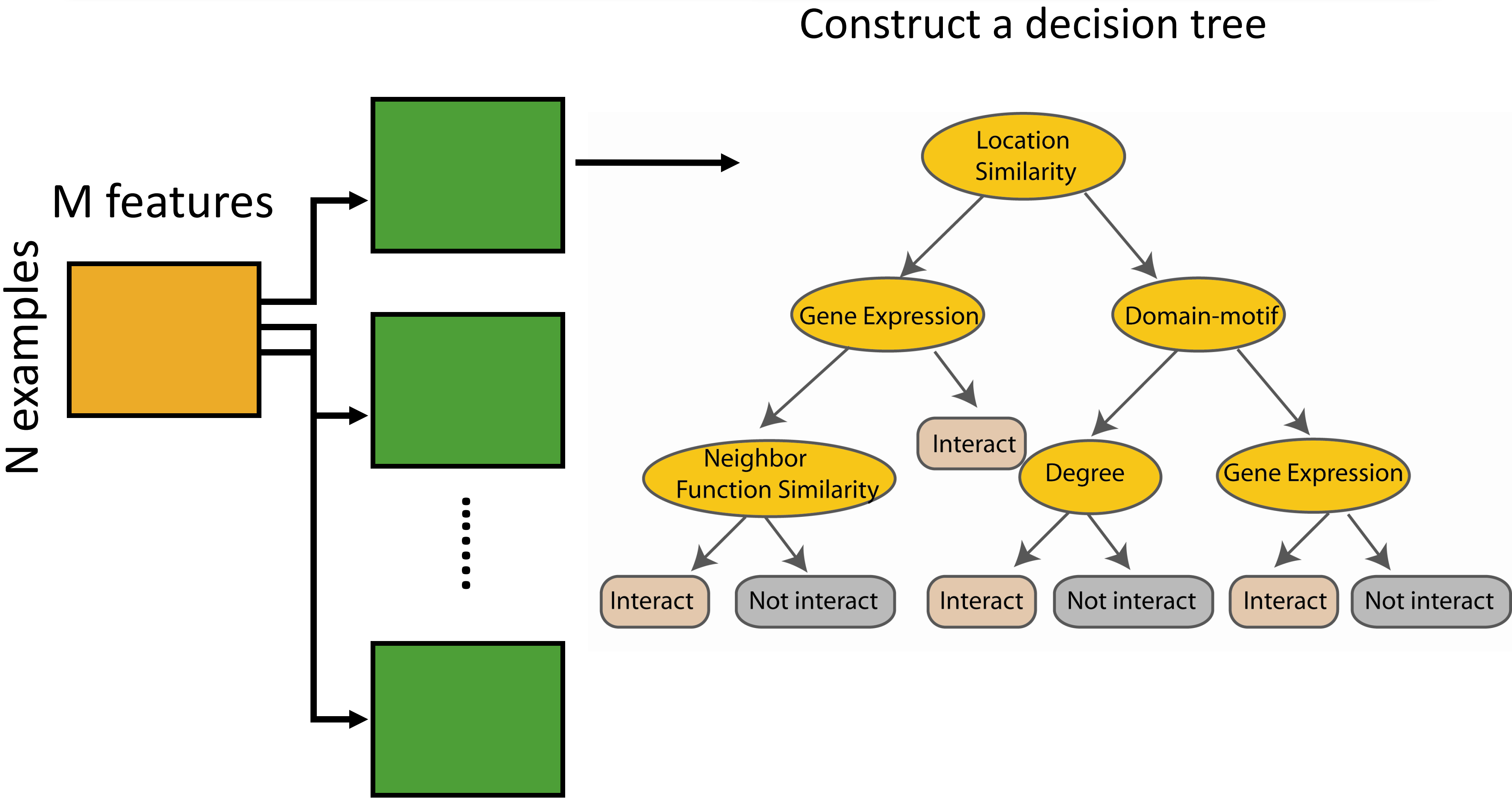
Training Data



Random Forest Classifier

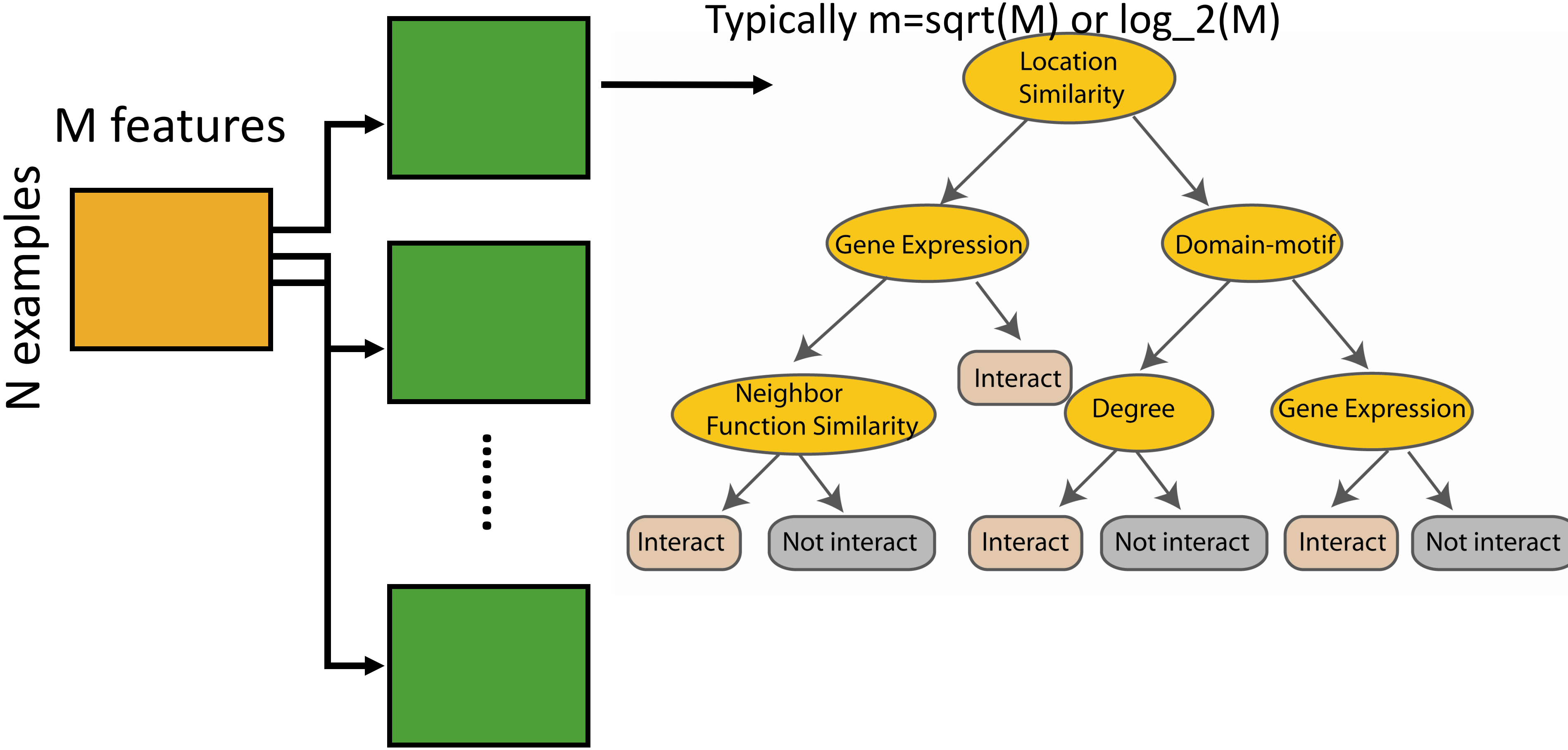


Random Forest Classifier

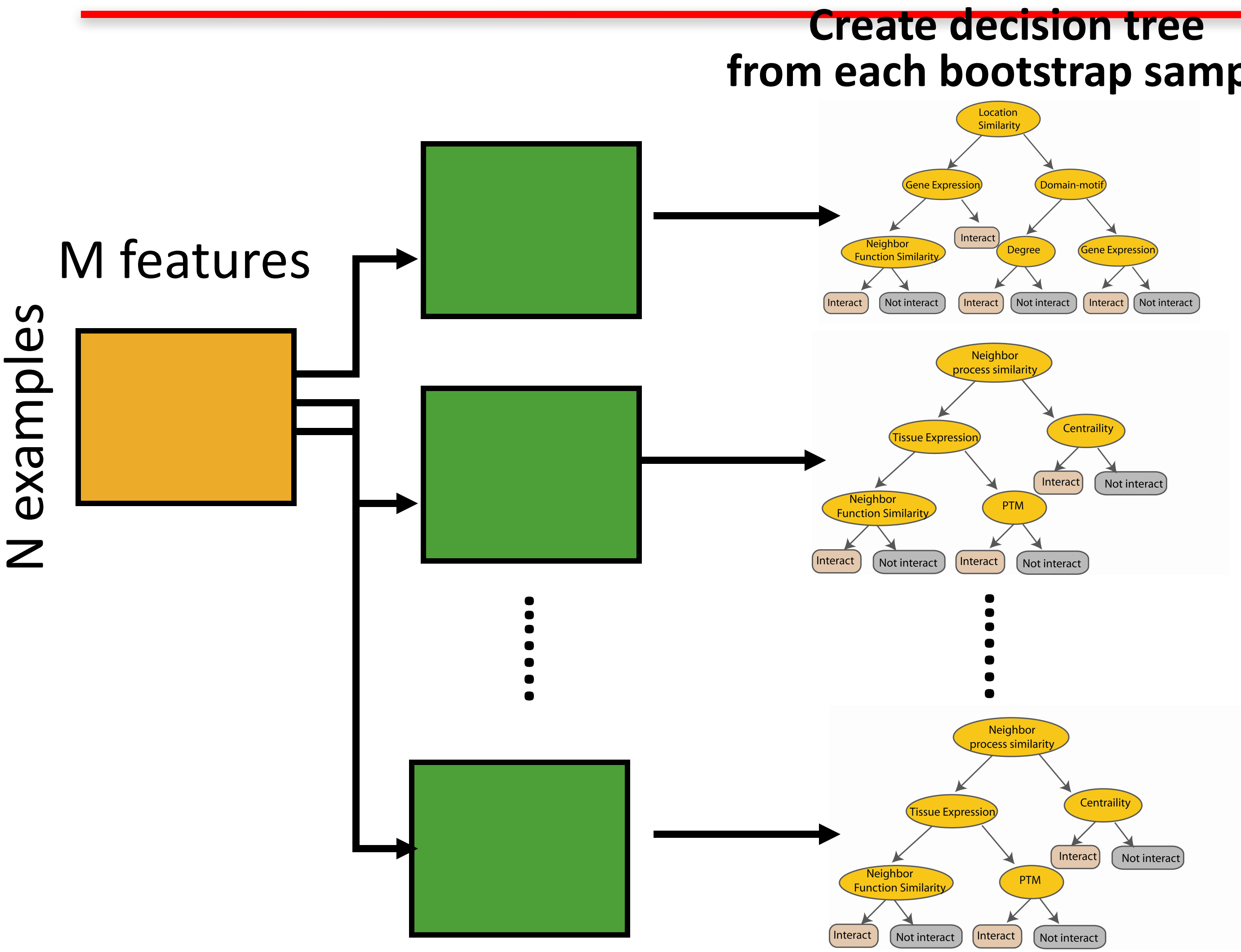


Random Forest Classifier

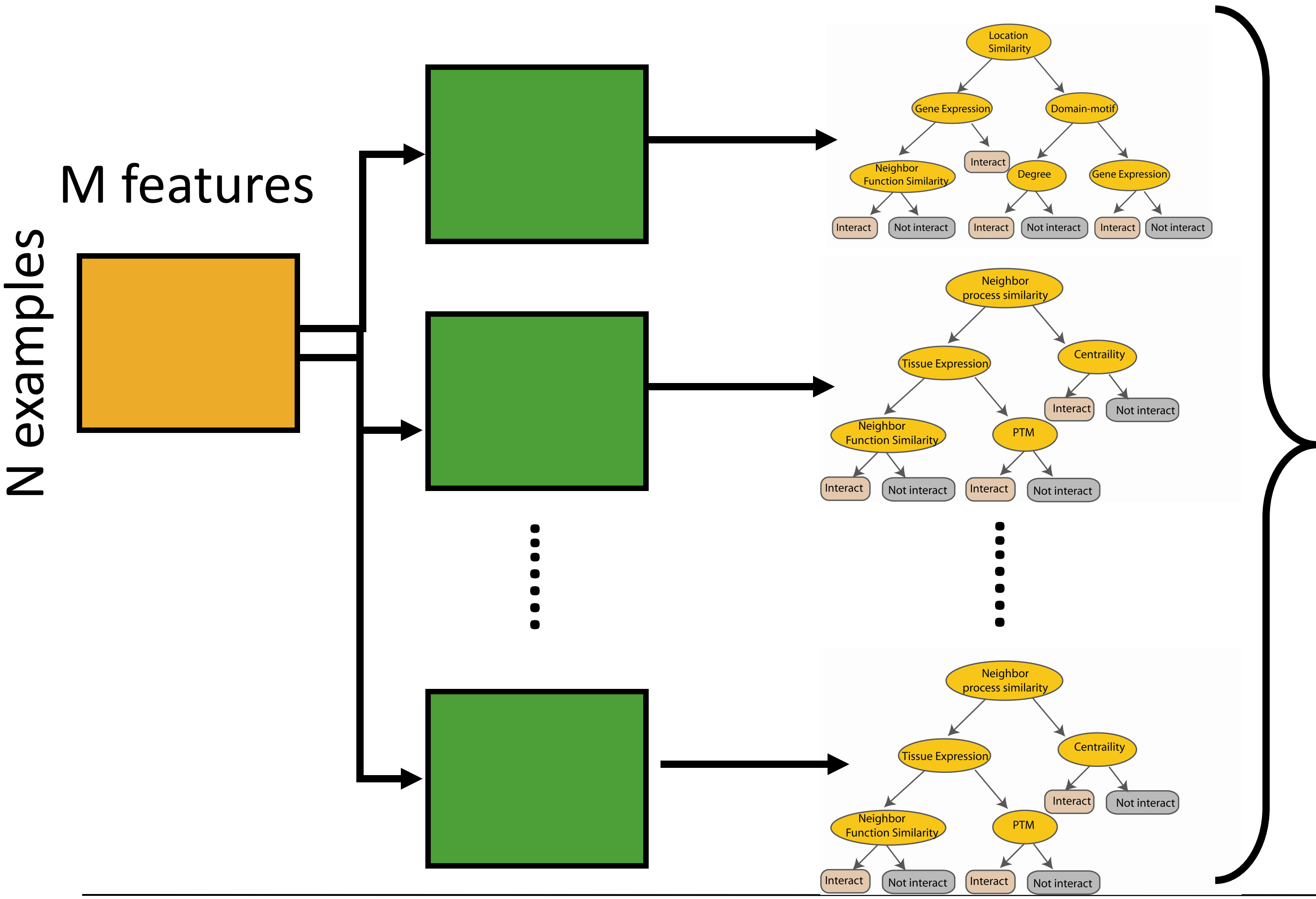
At each node in choosing the split feature
choose only among *m*<*M* features
Typically m=sqrt(M) or log_2(M)



Random Forest Classifier



Random Forest Classifier



Take he majority vote

Random Forests Summary

- Advantages
  - Computationally Fast – can handle thousands of input variables
  - Trees can be trained simultaneously
  - Exceptional Classifiers – one of most accurate available
  - Provide information on variable importance for the purposes of feature selection
  - Can effectively handle missing data
- Disadvantages
  - No interpretability in final model aside from variable importance
  - Prone to overfitting --- **solution: limiting maximum tree depth**
  - Lots of tuning parameters like the number of trees, the depth of each tree, the percentage of variables passed to each tree