Random Forests

Based on slides by Oznur Tastan et.al

Background

"Two heads are better than one."

"三个臭皮匠，顶一个诸葛亮"

↓

- Integrate results of multiple learning approaches to improve the performance

Ensemble learning

Two concepts

- Strong learner: learning algorithm with high accuracy
- Weak learner: performance on any training set is slightly better than chance prediction

$$error = ½ - \gamma$$

Can we improve a weak learner to a strong learner?

Introduction to ensemble learning

- INTUITION: *Combining Predictions of an ensemble is more accurate than a single classifier*
- Justification: (Several reasons)
  - Easy to find quite correct "rules of thumb" however hard to find single highly accurate prediction rule.
  - If the training examples are few and the hypothesis space is large then there are several equally accurate classifiers.
  - Hypothesis space does not contain the true function, but it has several good approximations.
  - Exhaustive global search in the hypothesis space is expensive so we can combine the predictions of several locally accurate classifiers.

Ensemble learning: basic idea

- Sometimes a single classifier (e.g. decision tree, neural network, …) won't perform well, but a weighted combination of them will.
- Each learner in the pool has its own weight
- When ask to predict the label for a new example
  - Each expert makes its own prediction
  - Then the master algorithm combine them using the weights for its own prediction (i.e. the "official" one)

Properties of a Tree

- Handles huge datasets
- Works for both classification and regression
- Handles categorical predictors naturally
- No formal distributional assumptions
- Can handle highly non-linear interactions and classification boundaries
- Handles missing values in the features
- Easily ignore redundant variables
- Small Tree are easy to interpret
- Large trees are hard to interpret
- Often prediction performance is poor

Random Forests

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.

Advantages of Random Forests

- Built-in estimates of accuracy.
- Automatic feature selection.
- feature importance.
- Works well "off the shelf".

Model Averaging

Classification trees can be simple, but often produce noisy (bushy) or weak (stunted) classifiers.

- Bagging (Breiman, 1996): Fit many large trees to bootstrap-resampled versions of the training data, and classify by majority vote.

- Boosting (Freund & Shapire, 1996): Fit many large or small trees to reweighted versions of the training data. Classify by weighted majority vote.

- Random Forests (Breiman 1999): Fancier version of bagging.

In general Boosting $\succ$ Random Forests $\succ$ Bagging $\succ$ Single Tree.

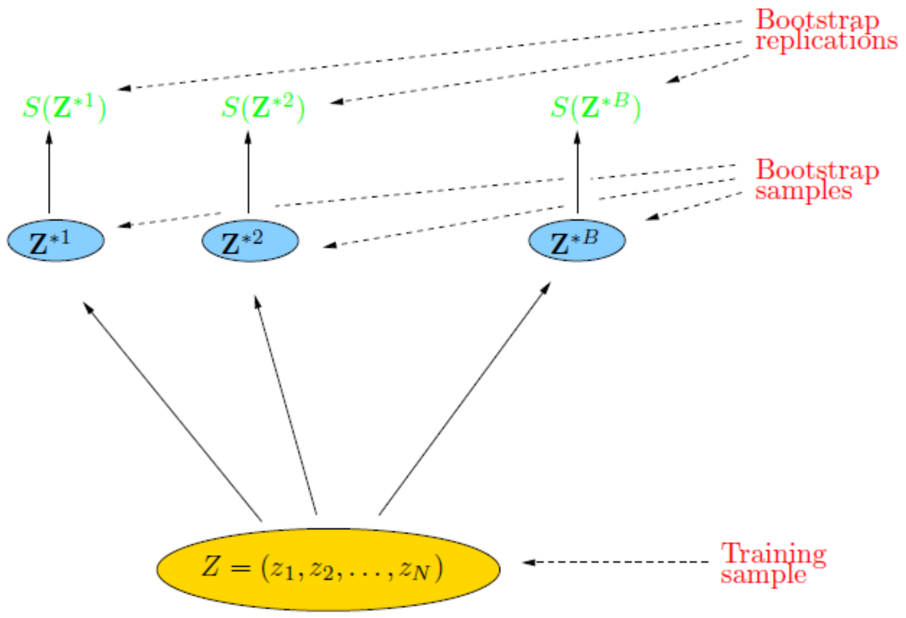
Bagging

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

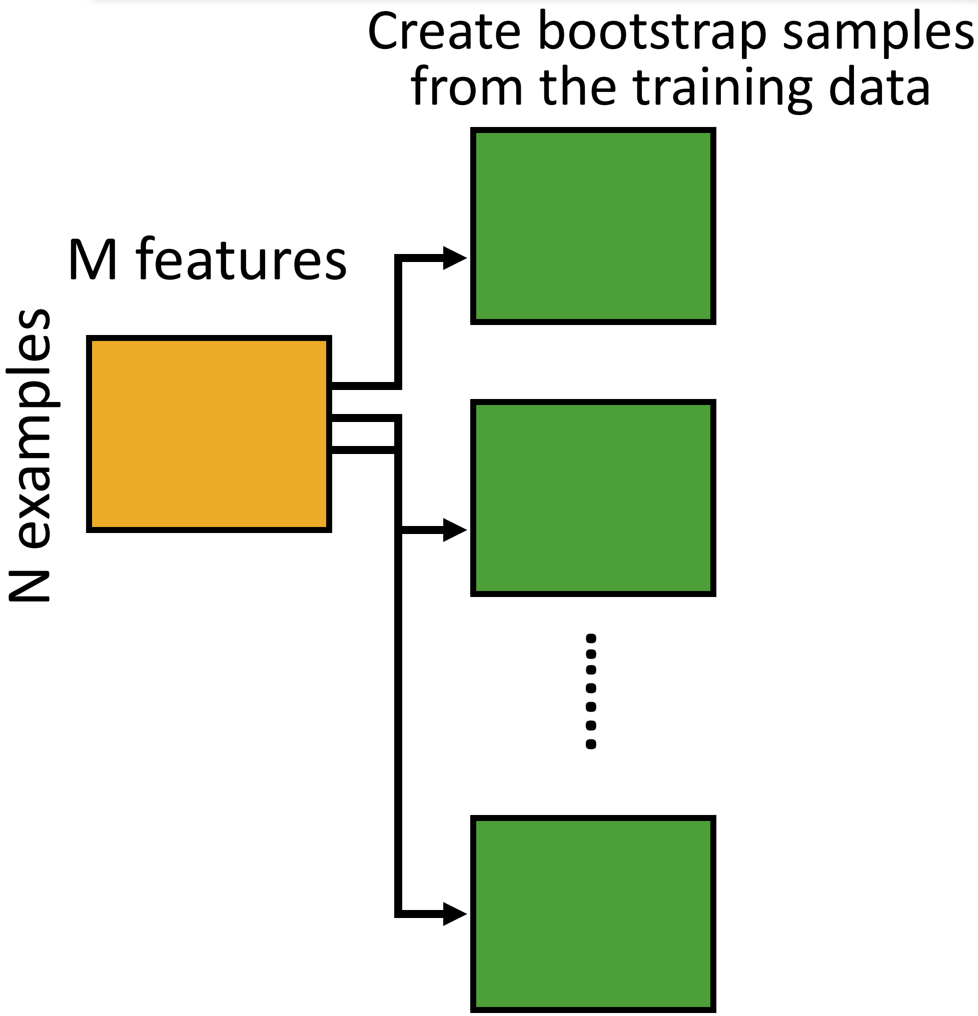
- For classification, a *committee* of trees each cast a vote for the predicted class.

Bootstrap

The basic idea:
randomly draw *B* datasets *with replacement* from the training data with size *N*, each dataset samples *the same size (N) as the original training set*



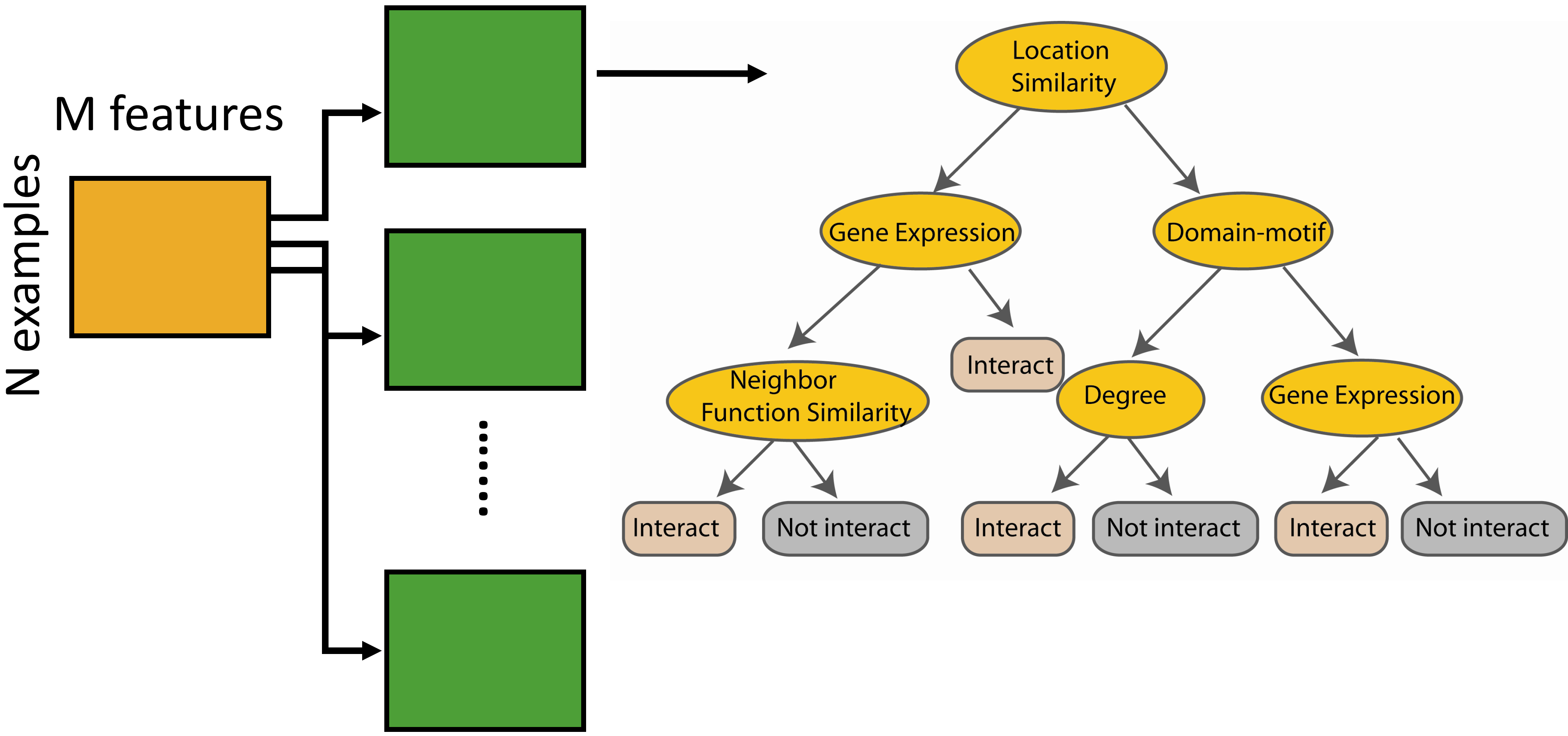
Bagging



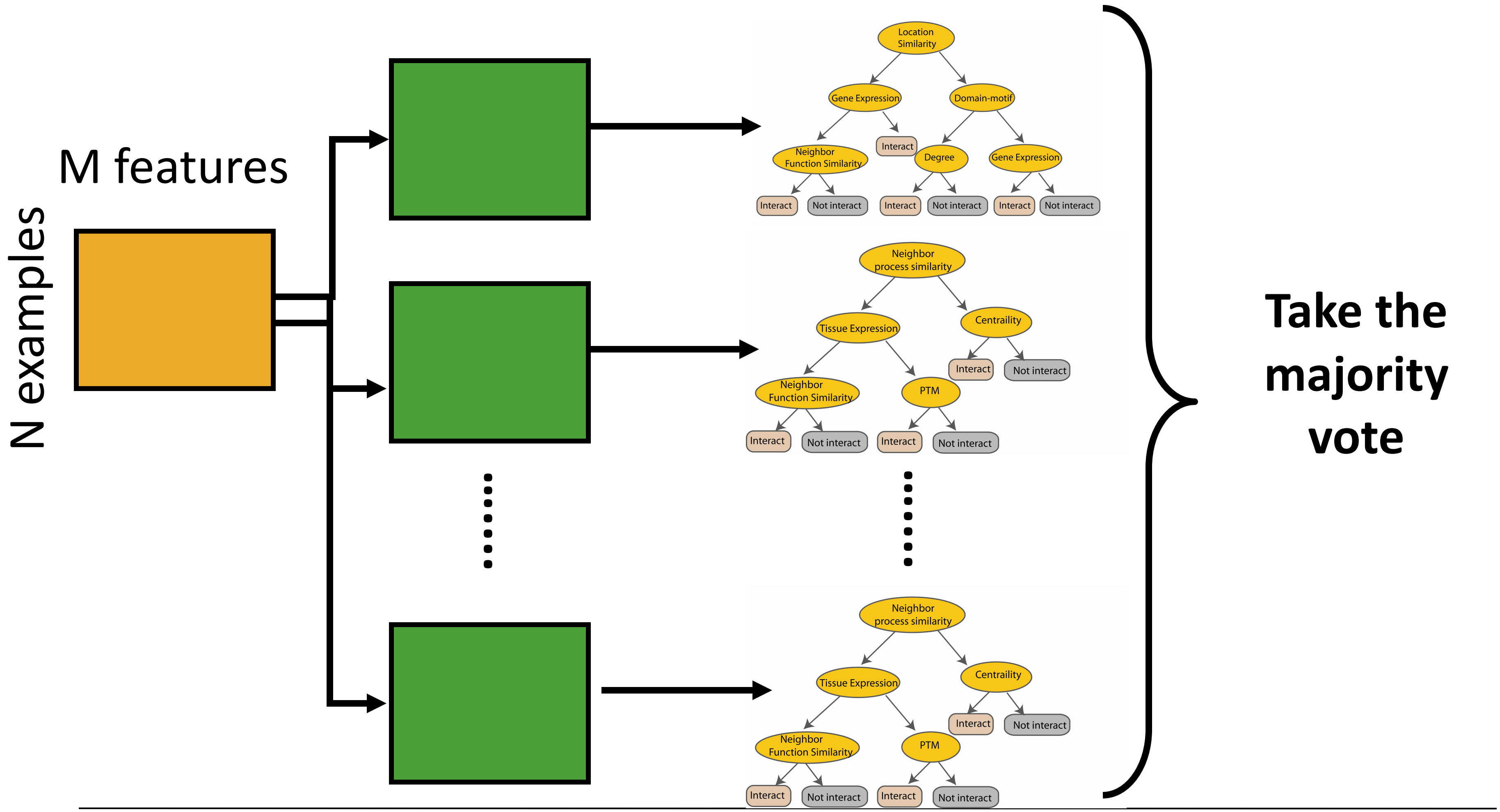
Create bootstrap samples
from the training data
M features
N examples

Random Forest Classifier

Construct a decision tree



Random Forest Classifier



Bagging

$\mathbf{Z} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ Training Sample

Z^{*b} where = 1,.., B.

The prediction at input x when bootstrap sample *b* is used for training

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

http://www-stat.stanford.edu/~hastie/Papers/ESLII.pdf (Chapter 8.7)

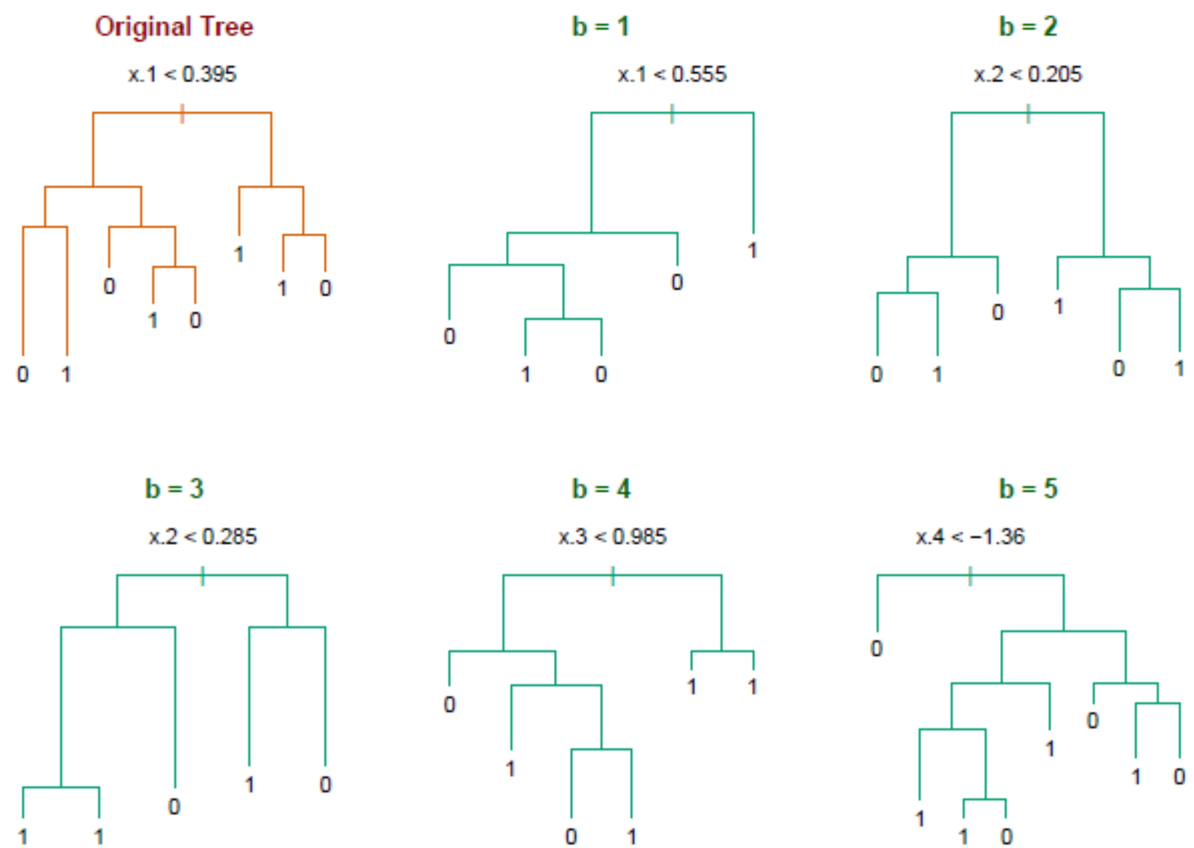
Bagging : an simulated example

Generated a sample of size $N = 30$, with two classes and $p = 5$ features, each having a standard Gaussian distribution with pairwise Correlation 0.95.

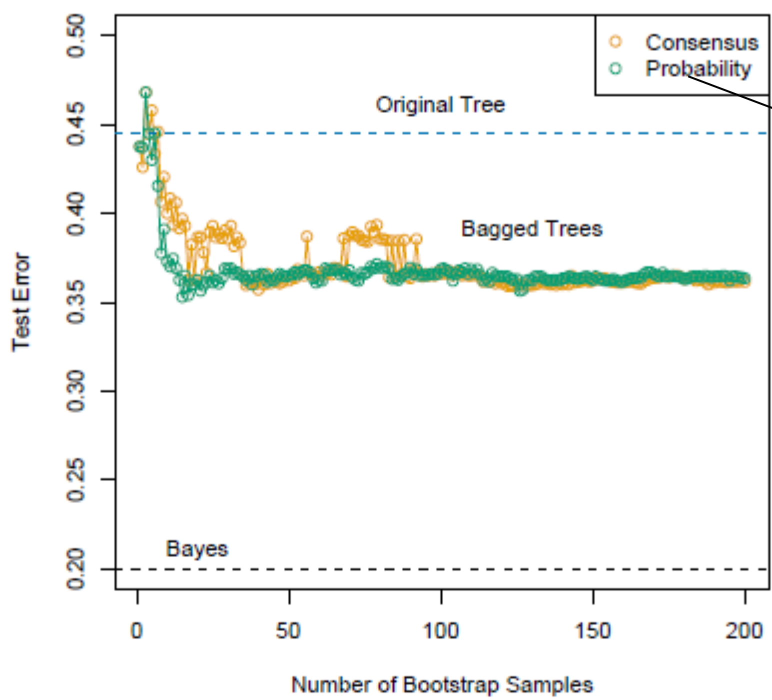
The response Y was generated according to
$\Pr(Y = 1/x1 \leq 0.5) = 0.2$,
$\Pr(Y = 0/x1 > 0.5) = 0.8$.

Bagging

Notice the bootstrap trees are different than the original tree



Bagging



Treat the voting Proportions as probabilities

FIGURE 8.10. *Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The orange points correspond to the consensus vote, while the green points average the probabilities.*

bagging helps under squared-error loss, in short because averaging reduces

Hastie

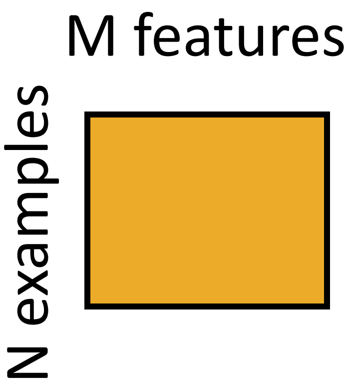
http://www-stat.stanford.edu/~hastie/Papers/ESLII.pdf Example 8.7.1

Random forest classifier

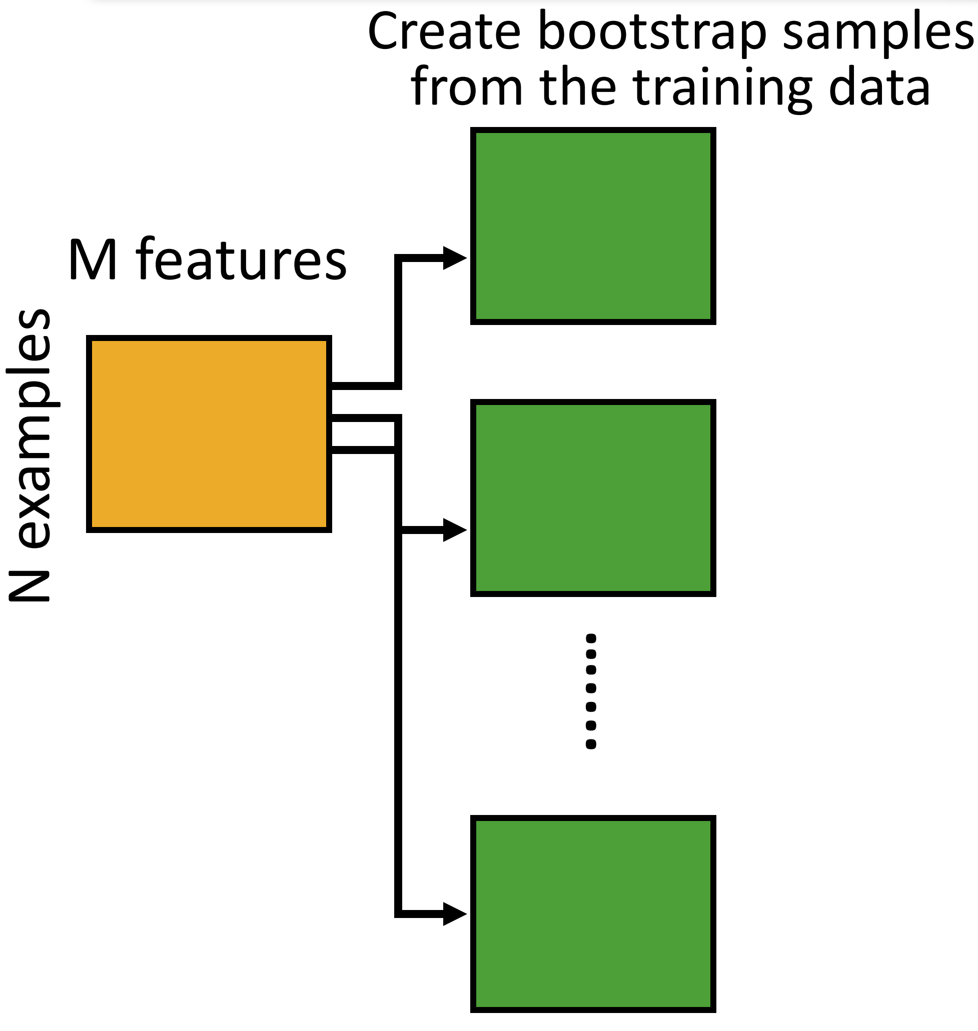
Random forest classifier, an extension to bagging which uses *de-correlated* trees.

Random Forest Classifier

Training Data

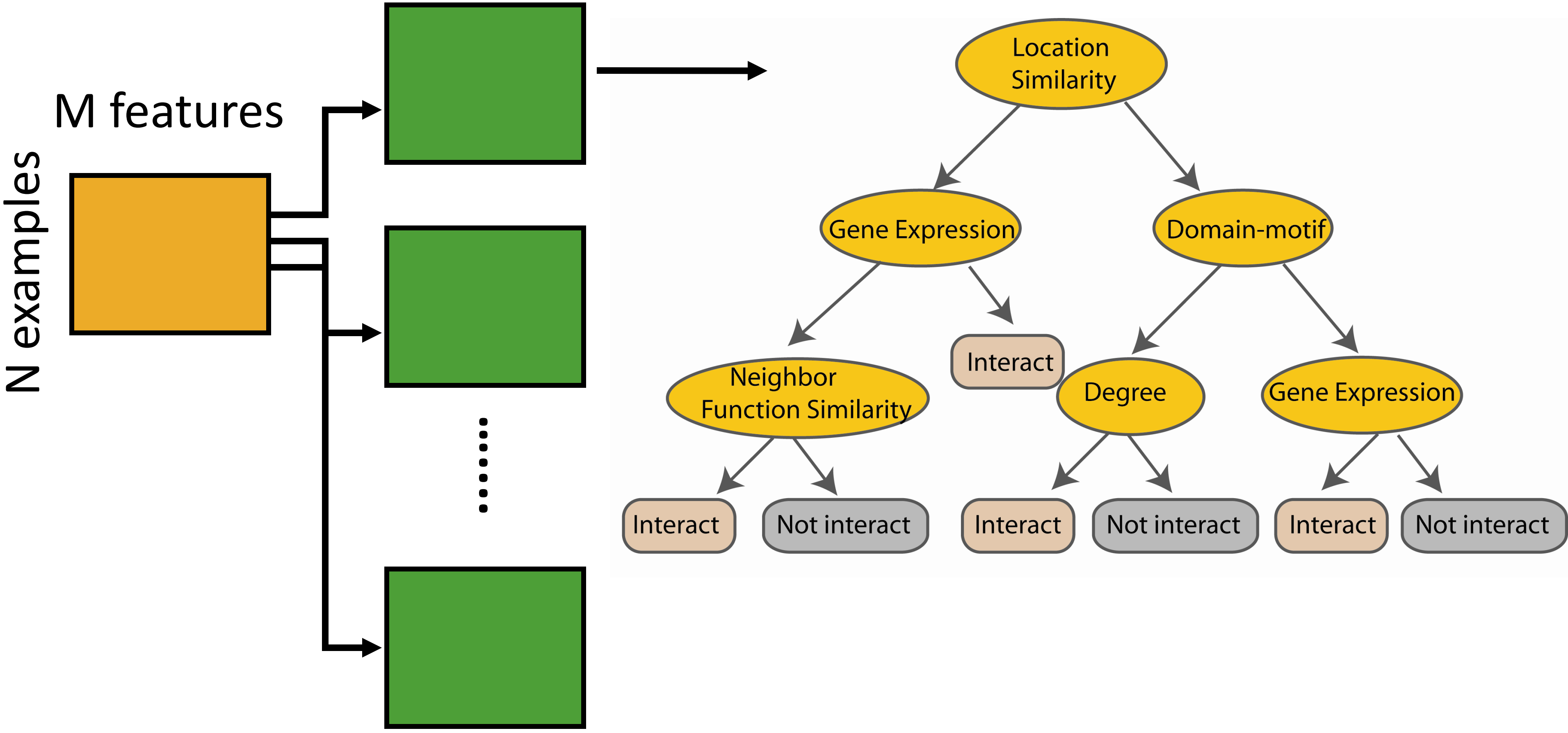


Random Forest Classifier



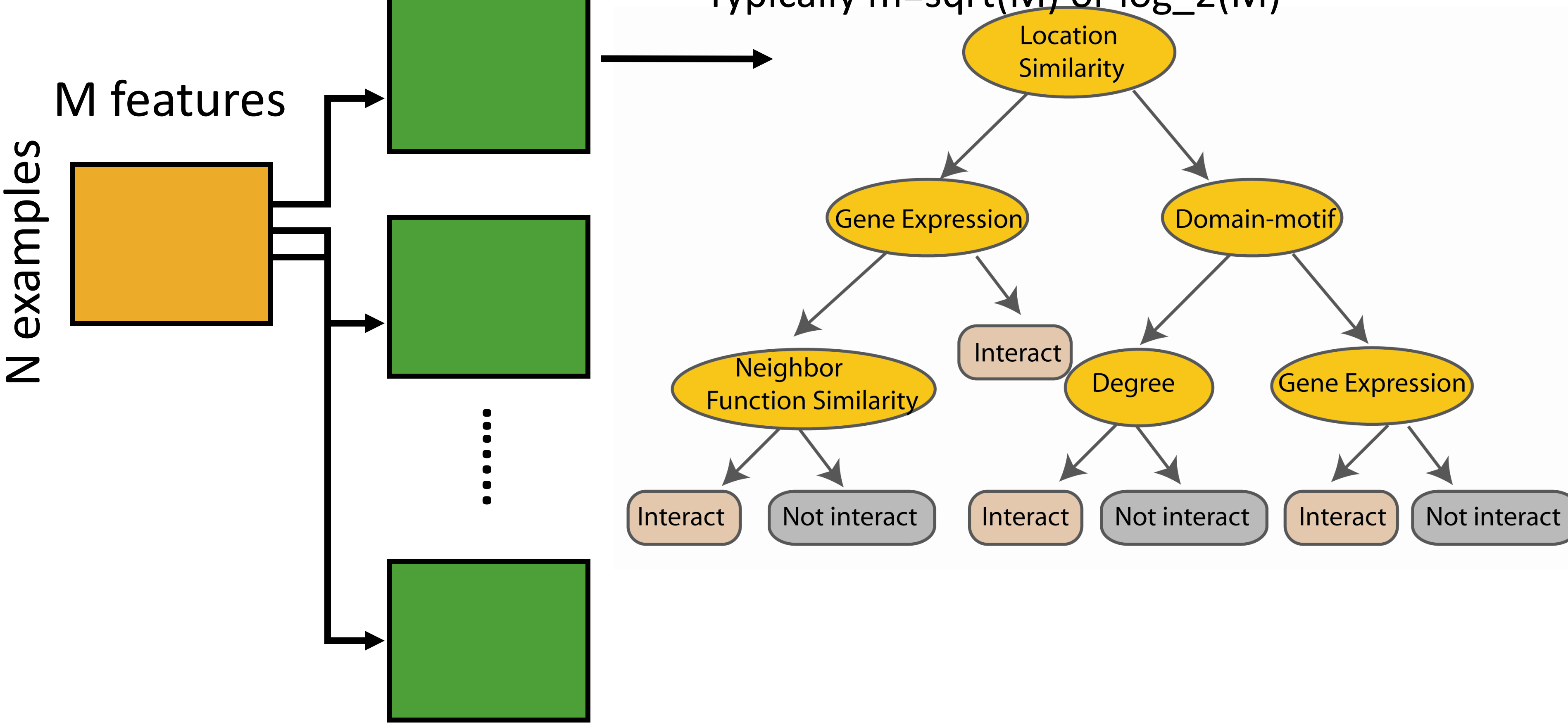
Random Forest Classifier

Construct a decision tree

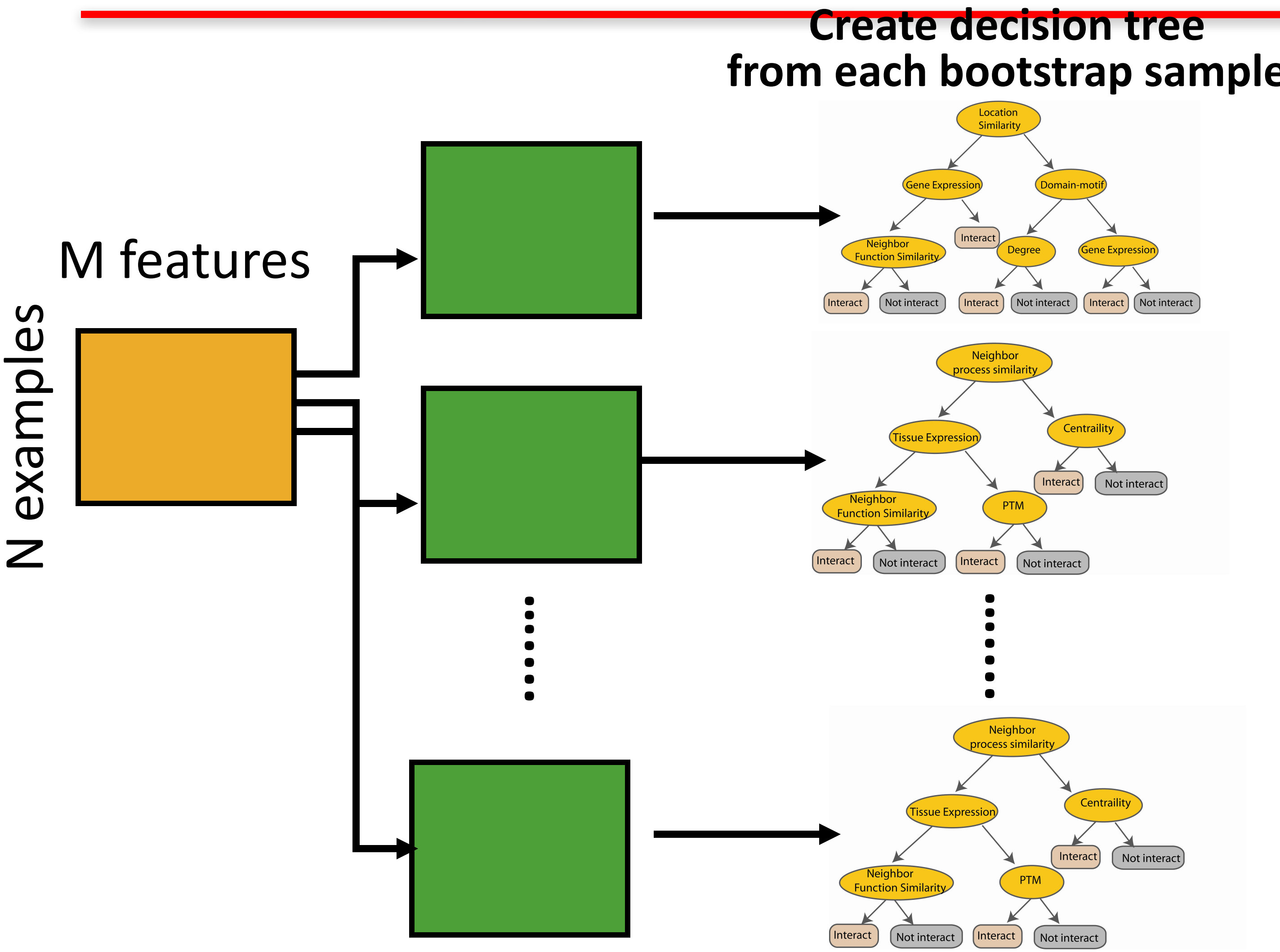


Random Forest Classifier

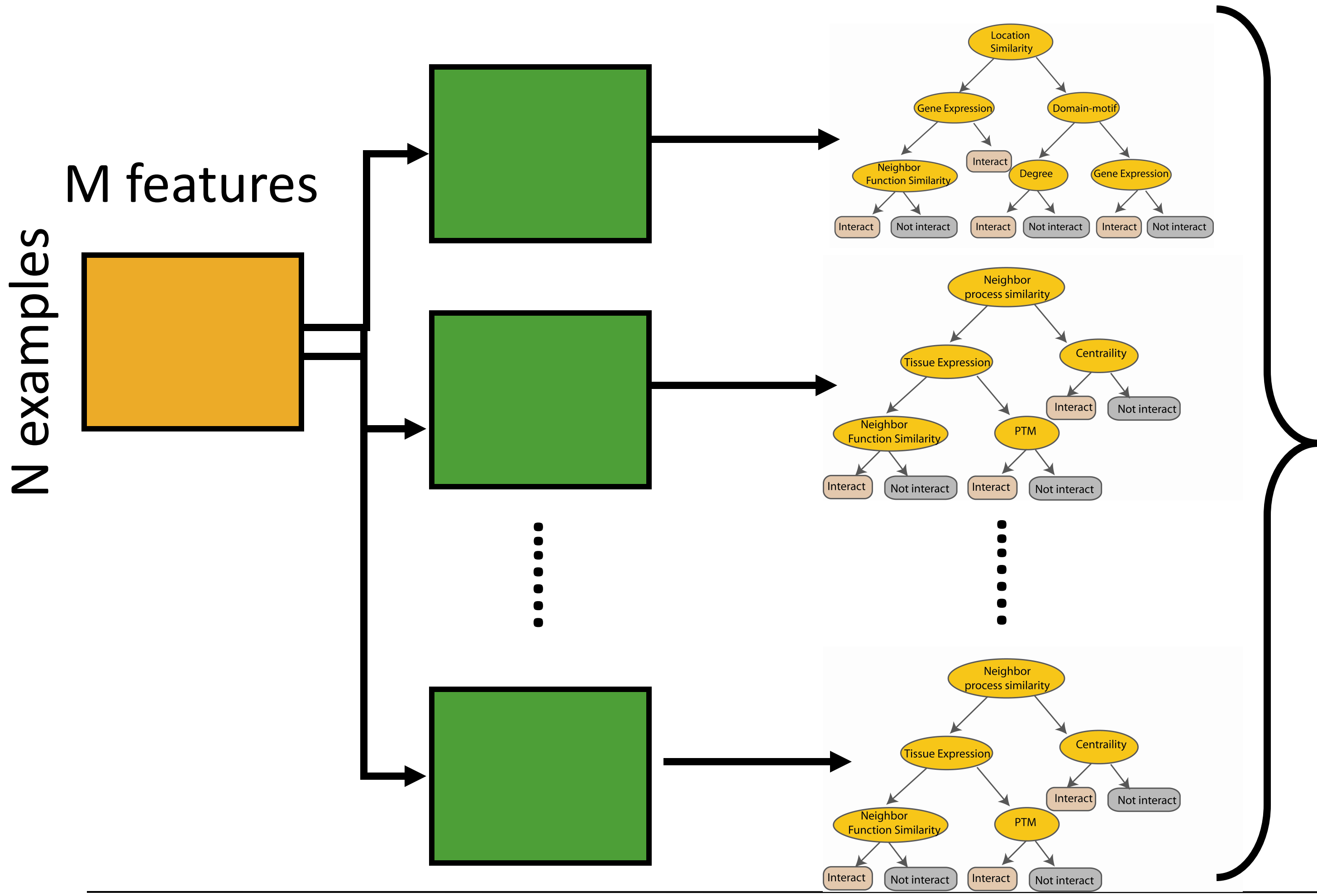
At each node in choosing the split feature
choose only among *m*<*M* features
Typically m=sqrt(M) or log_2(M)



Random Forest Classifier



Random Forest Classifier



Take he majority vote

Trees are de-correlated in random forest!

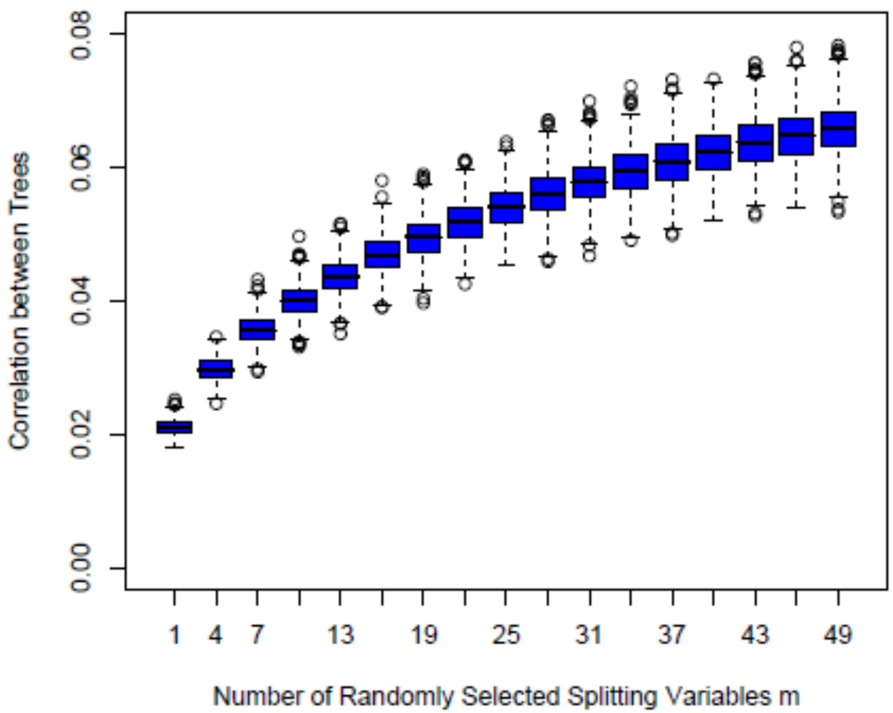


FIGURE 15.9. *Correlations between pairs of trees drawn by a random-forest regression algorithm, as a function of m. The boxplots represent the correlations at 600 randomly chosen prediction points x.*

Random forest

Available package:

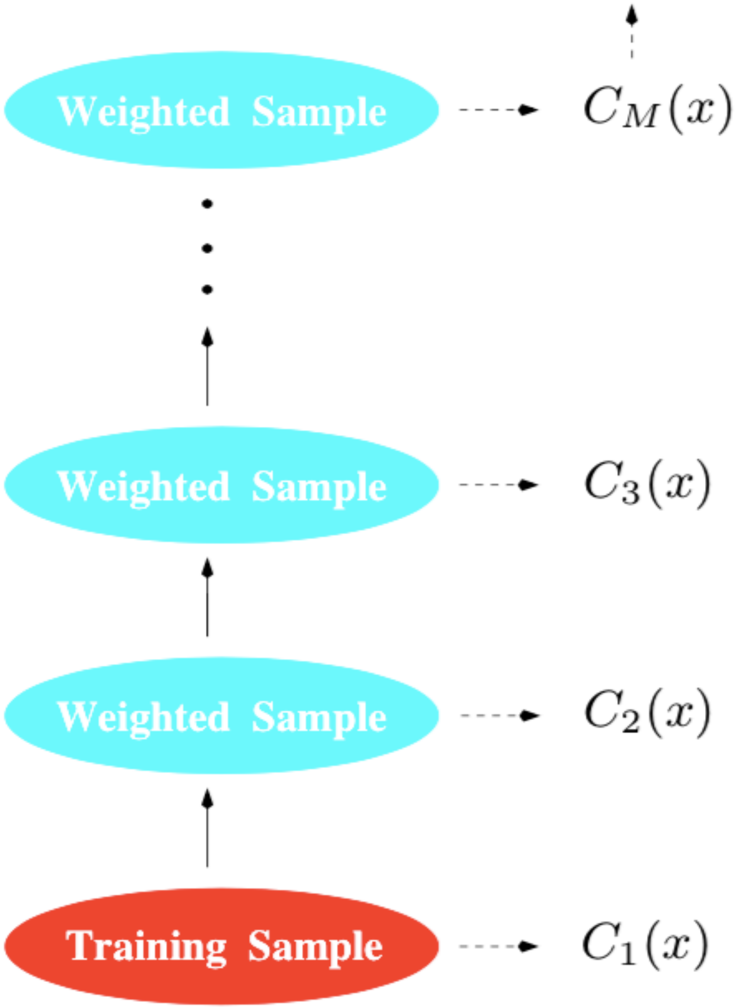
http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

To read more:

http://www-stat.stanford.edu/~hastie/Papers/ESLII.pdf

Advantages of Random Forests

- Built-in estimates of accuracy.
- Automatic feature selection.
- feature importance.
- Works well “off the shelf”.



Boosting

- Average many trees, each grown to re-weighted versions of the training data.
- Final Classifier is weighted average of classifiers:

$$C(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m C_m(x)\right]$$

AdaBoost (Freund & Schapire, 1996)

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \dots, N$.
2. For $m = 1$ to M repeat steps (a)–(d):
   (a) Fit a classifier $C_m(x)$ to the training data using weights w_i.
   (b) Compute weighted error of newest tree

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq C_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

   (c) Compute $\alpha_m = \log[(1 - \text{err}_m)/\text{err}_m]$.
   (d) Update weights for $i = 1, \dots, N$:
   $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq C_m(x_i))]$
   and renormalize to w_i to sum to 1.
3. Output $C(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m C_m(x)\right]$.

The smaller the error of a tree, the higher the weight for this tree