Data Mining

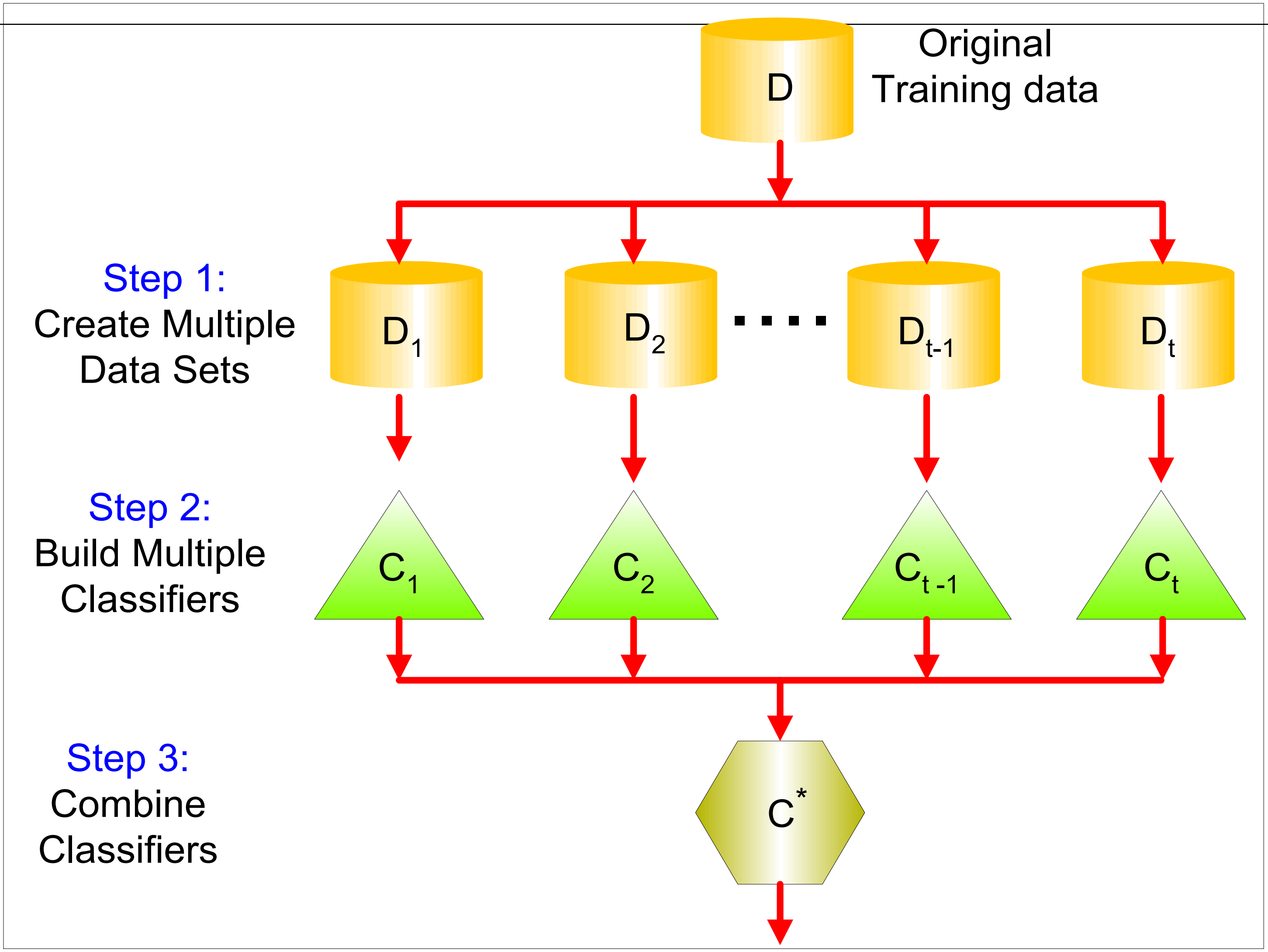
Classification:

Ensemble Methods

Ensemble Methods

- Construct a set of classifiers from the training data
- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers

General Idea



Why does it work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, ε = 0.35
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

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

Examples of Ensemble Methods

- How to generate an ensemble of classifiers?
  - Manipulating the training set
    - Bagging
    - Boosting
  - Manipulating the features
    - Random Forests

Bagging

- Sampling with replacement

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

- Build classifier on each bootstrap sample

- Each sample has probability $(1 - 1/n)^n$ of being selected

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of boosting round

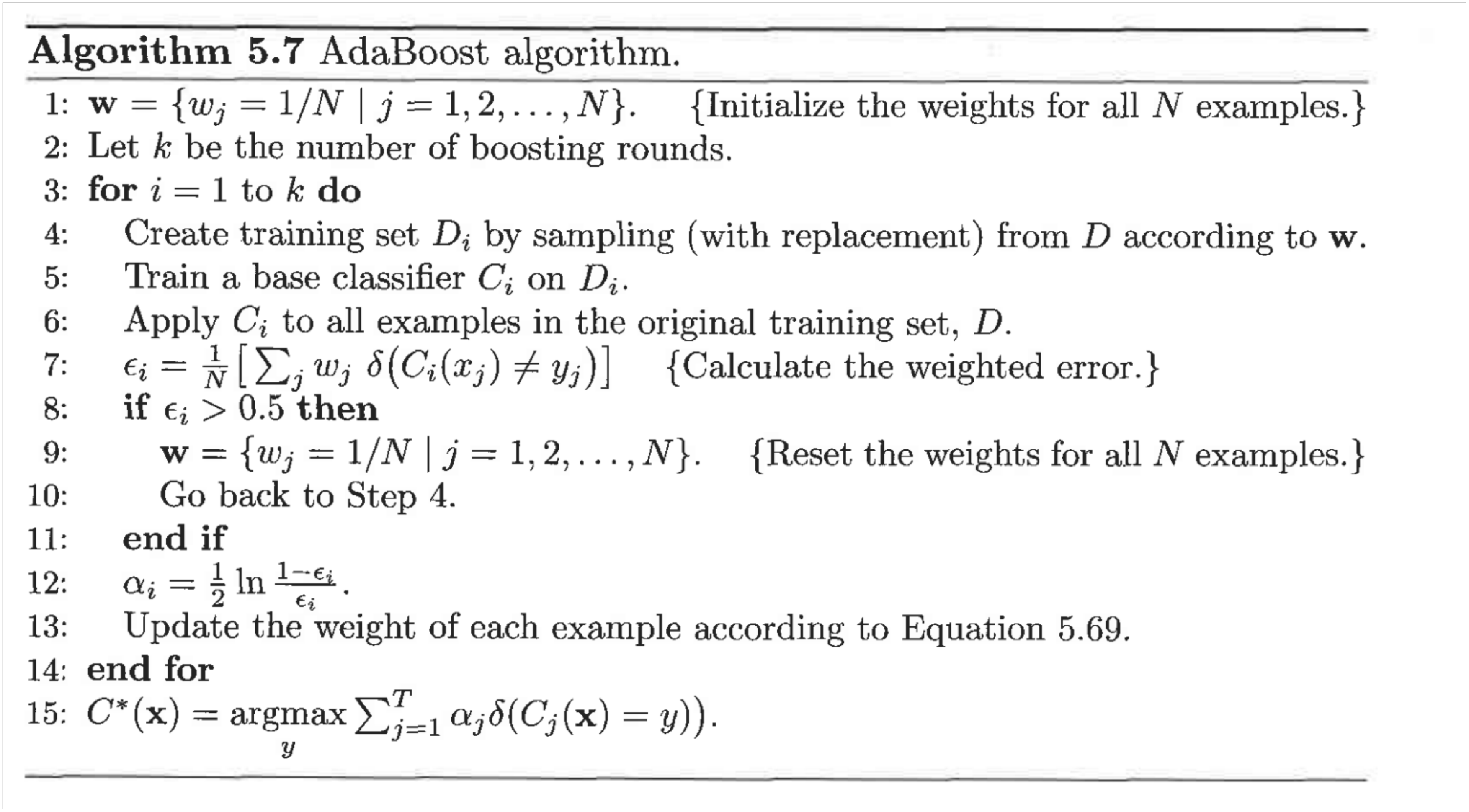
Boosting

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

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

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

AdaBoost Algorithm



Algorithm 5.7 AdaBoost algorithm.

```
1: $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \ldots, N\}$.   {Initialize the weights for all $N$ examples.}
2: Let $k$ be the number of boosting rounds.
3: for $i = 1$ to $k$ do
4:     Create training set $D_i$ by sampling (with replacement) from $D$ according to $\mathbf{w}$.
5:     Train a base classifier $C_i$ on $D_i$.
6:     Apply $C_i$ to all examples in the original training set, $D$.
7:     $\epsilon_i = \frac{1}{N}\left[\sum_j w_j\ \delta\big(C_i(x_j) \neq y_j\big)\right]$   {Calculate the weighted error.}
8:     if $\epsilon_i > 0.5$ then
9:         $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \ldots, N\}$.   {Reset the weights for all $N$ examples.}
10:        Go back to Step 4.
11:    end if
12:    $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$.
13:    Update the weight of each example according to Equation 5.69.
14: end for
15: $C^*(\mathbf{x}) = \underset{y}{\operatorname{argmax}} \sum_{j=1}^{T} \alpha_j \delta(C_j(\mathbf{x}) = y)\big)$.
```

Example: AdaBoost

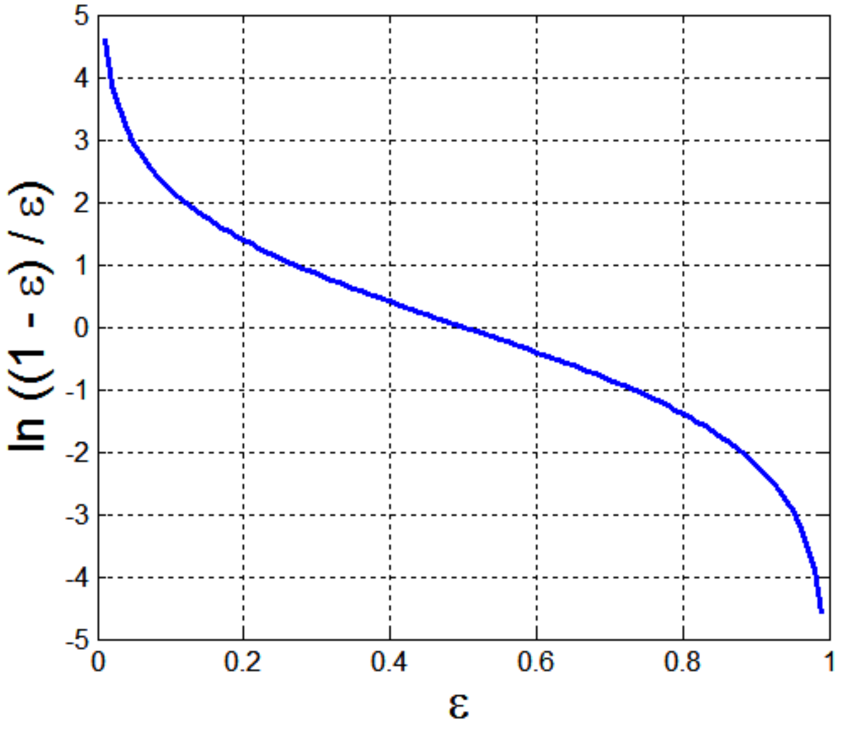
- Base classifiers: $C_1, C_2, \ldots, C_K$

- Error rate:

$$\varepsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

- Importance of a classifier:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$



Example: AdaBoost

- Weight update:

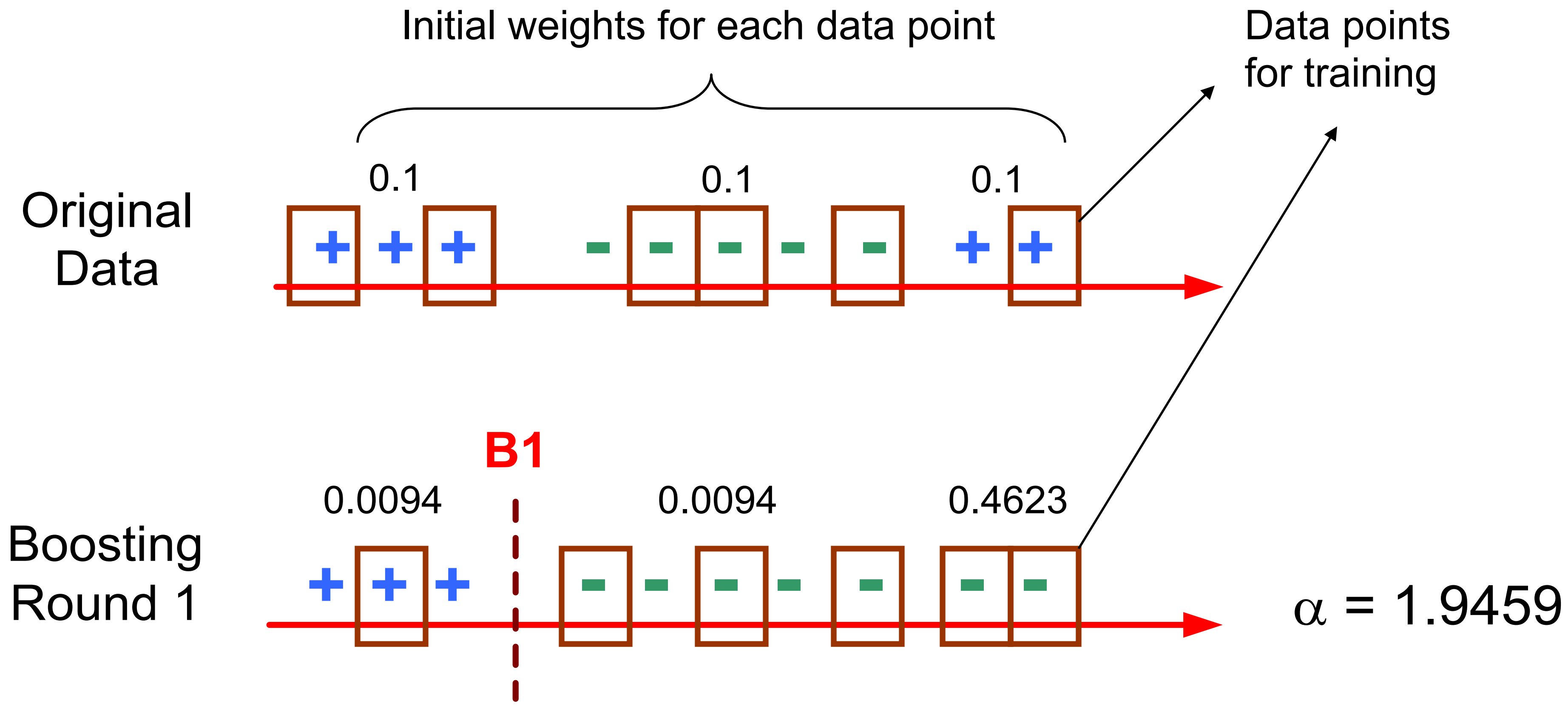
$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

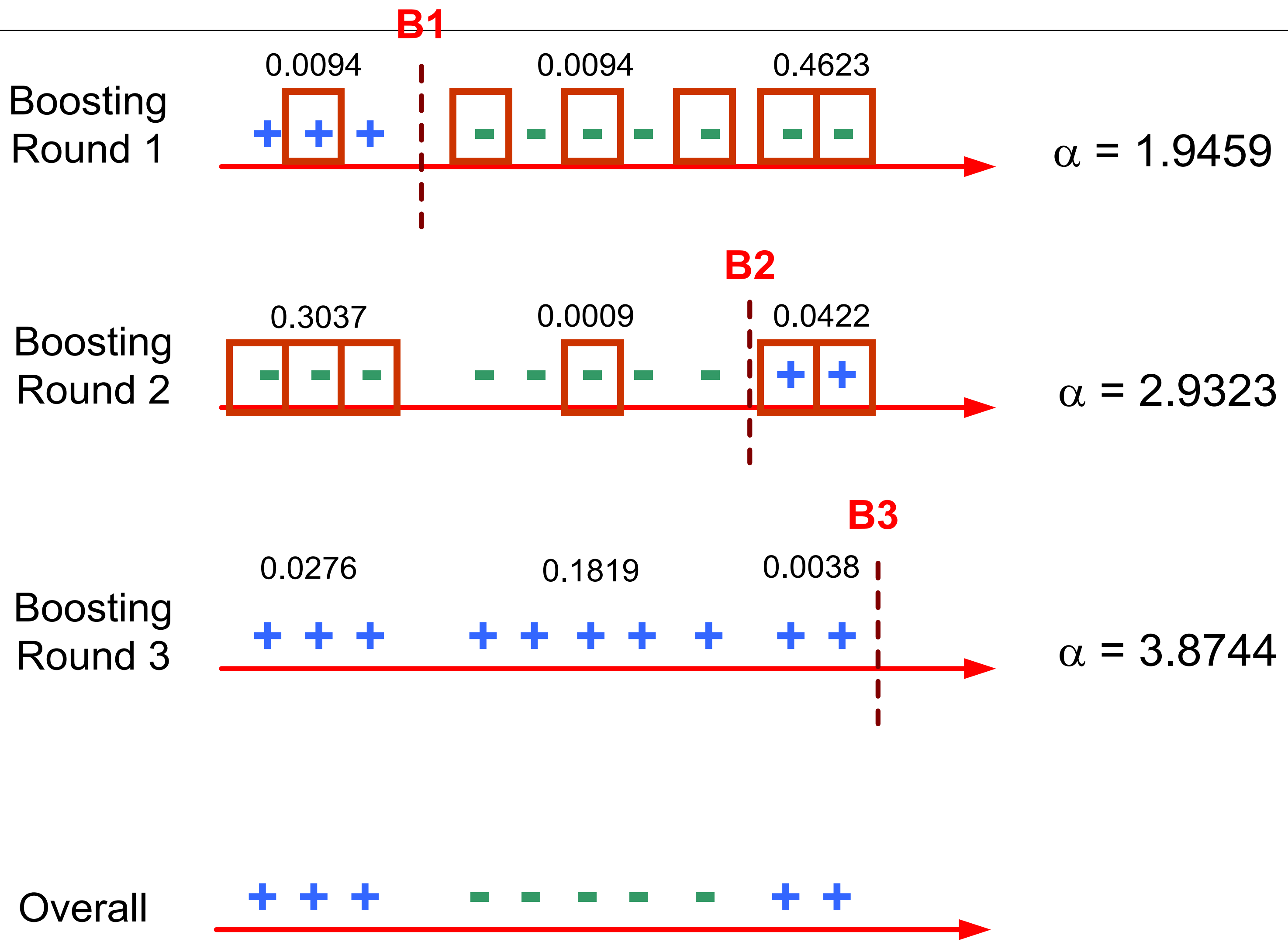
- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to 1/n and the resampling procedure is repeated
- Classification:

$$C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta\left(C_j(x) = y\right)$$

Illustrating AdaBoost



Illustrating AdaBoost



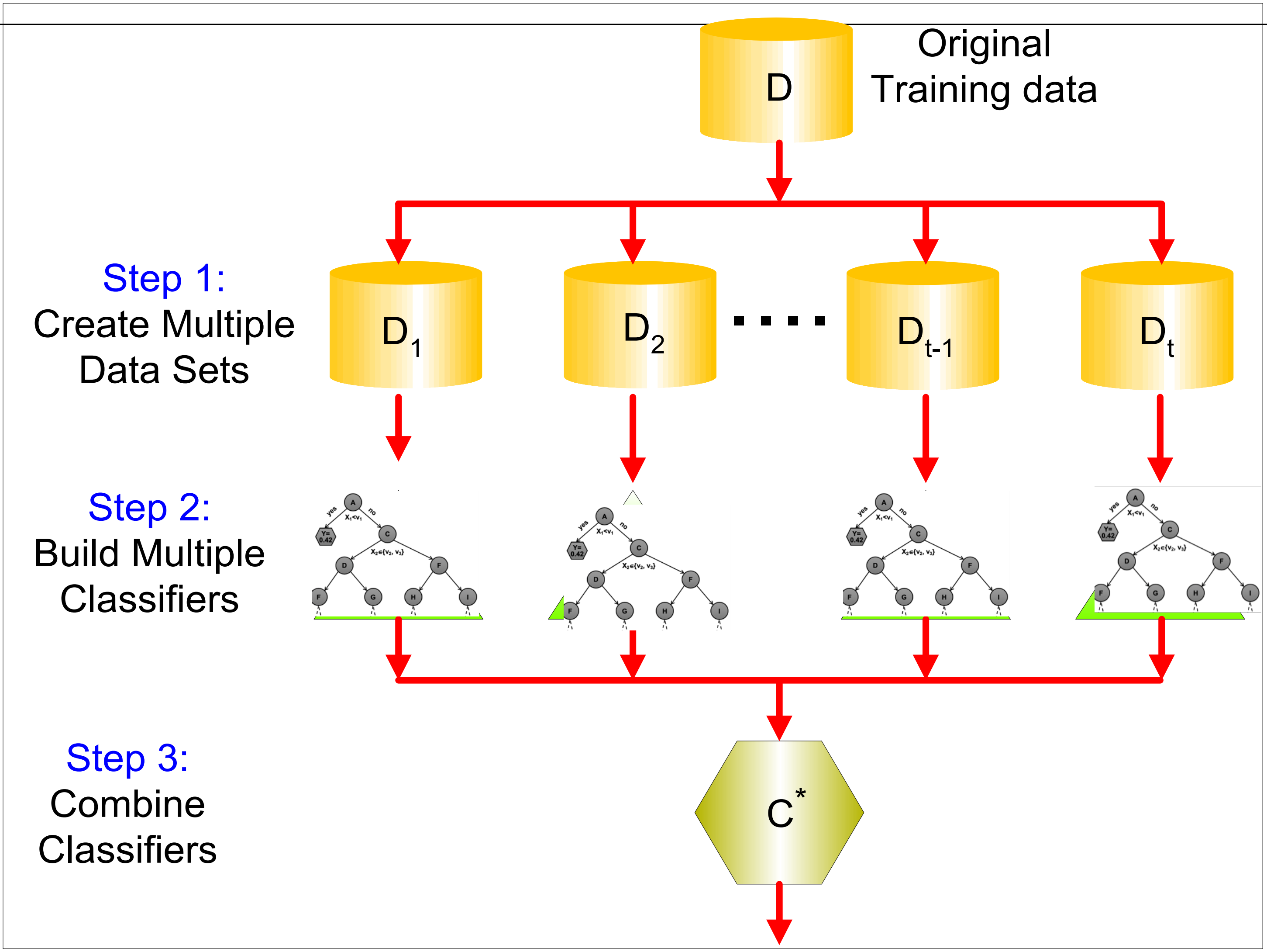
Random Forest

- Random forest is a class of ensemble methods specifically designed for decision tree classifiers
- It combines the predictions made by multiple decision trees
- each tree is generated based on the values of an independent set of random vectors
- The random vectors are generated from a fixed probability distribution
- Theoretically proven that

$$\text{Generalization error} \leq \frac{\overline{\rho}(1-s^2)}{s^2},$$

- Where,

Random Forest



Random Forests

As in bagging, we build a number of decision trees on bootstrapped training samples each time a split in a tree is considered, a random sample of *F* features is chosen as split candidates from the full set of m features.

Note that if $m = p$, then this is bagging.

Random Forests Algorithm

```
For b = 1 to B:
    (a) Draw a bootstrap sample D∗ of size $N$ from the training data.
        (b) Grow a random-forest tree 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 $F$ features at random from the $m$ variables.
            ii. Pick the best variable/split-point among the $F$.
            iii. Split the node into two child nodes.
Output the ensemble of trees.

To make a prediction at a new point $x$ we do:
        For regression: average the results
        For classification: majority vote
```

Random Forests Tuning

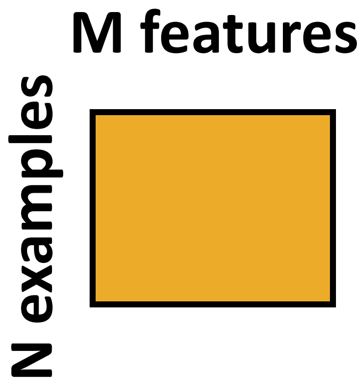
The inventors make the following recommendations:

- ► For classification, the default value for F is $\sqrt{m}$ and the minimum node size is one.
- ► For regression, the default value for m is $m/3$ and the minimum node size is five.

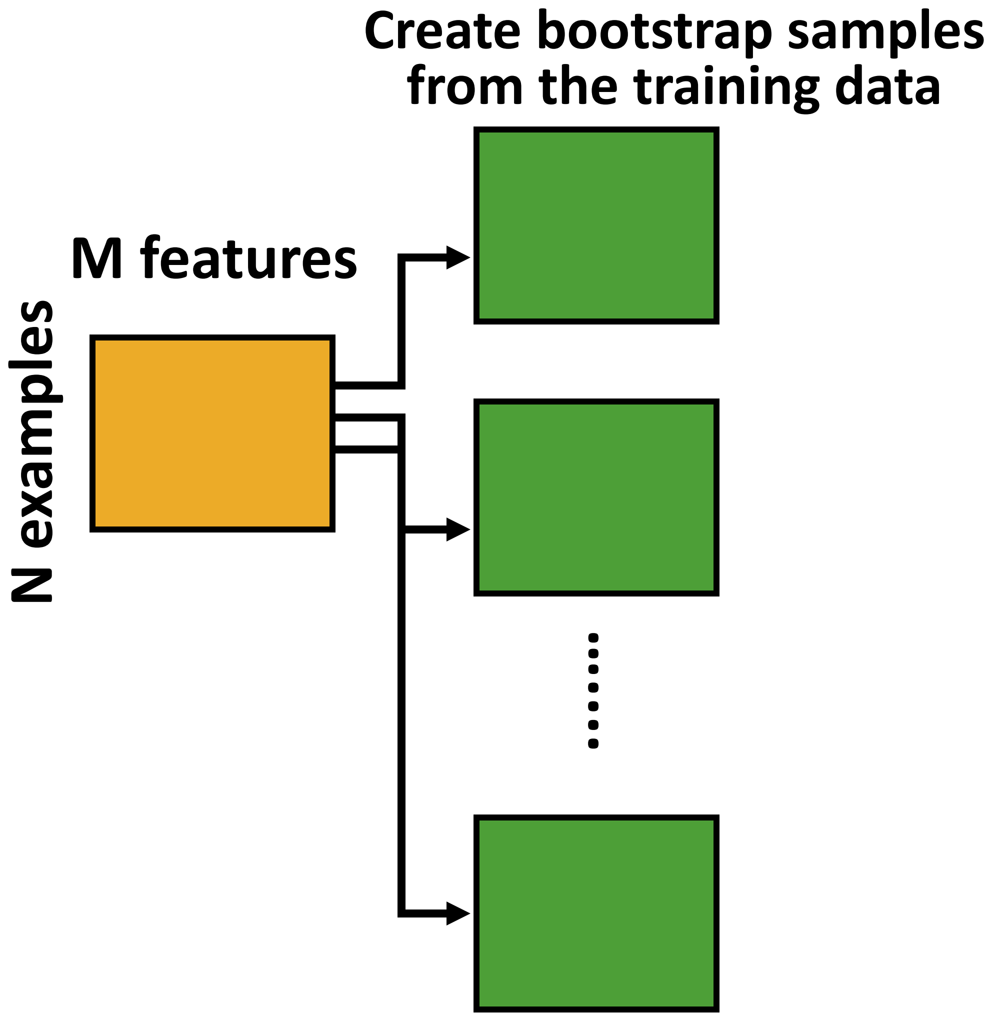
In practice the best values for these parameters will depend on the problem, and they should be treated as tuning parameters.

Random Forest Classifier

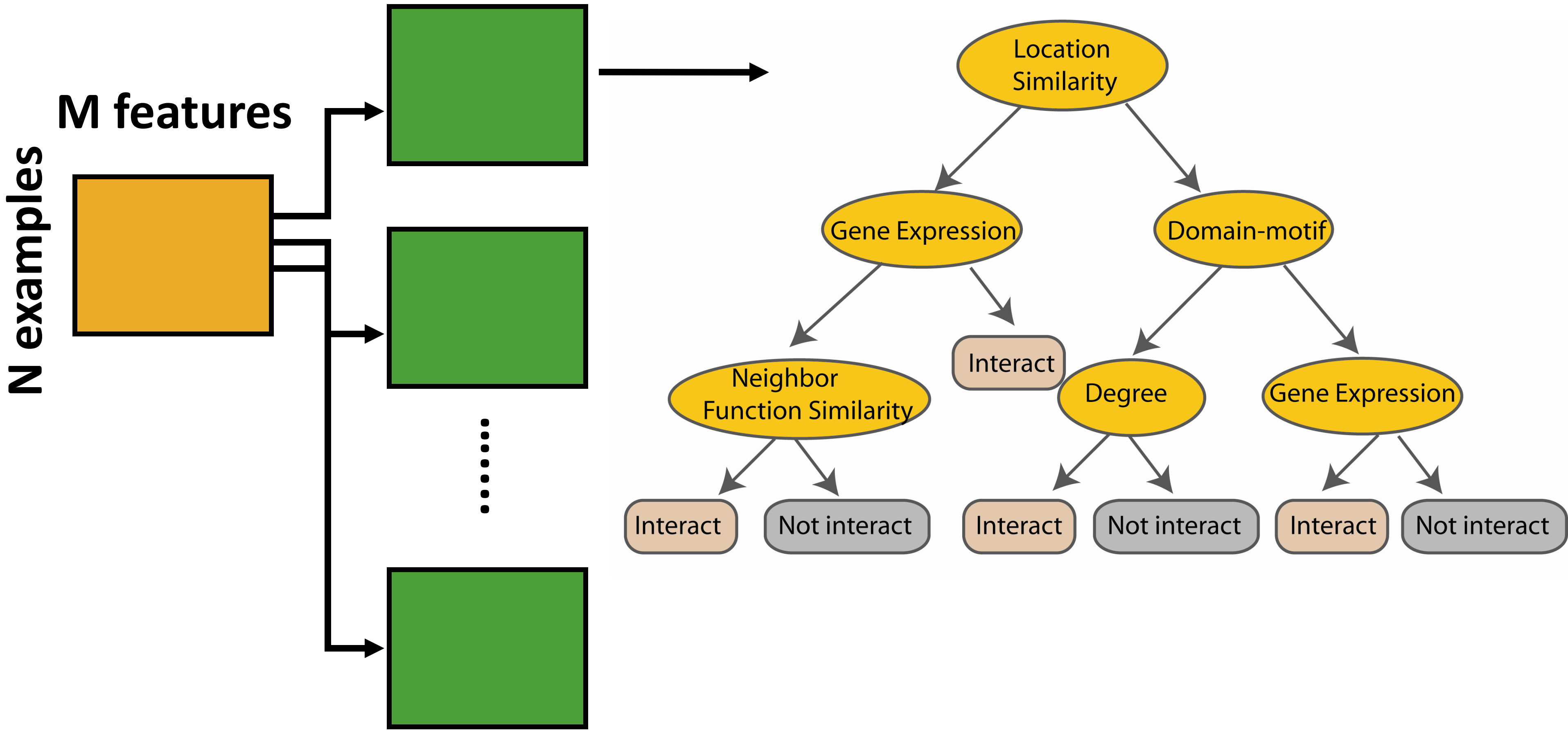
Training Data



Random Forest Classifier

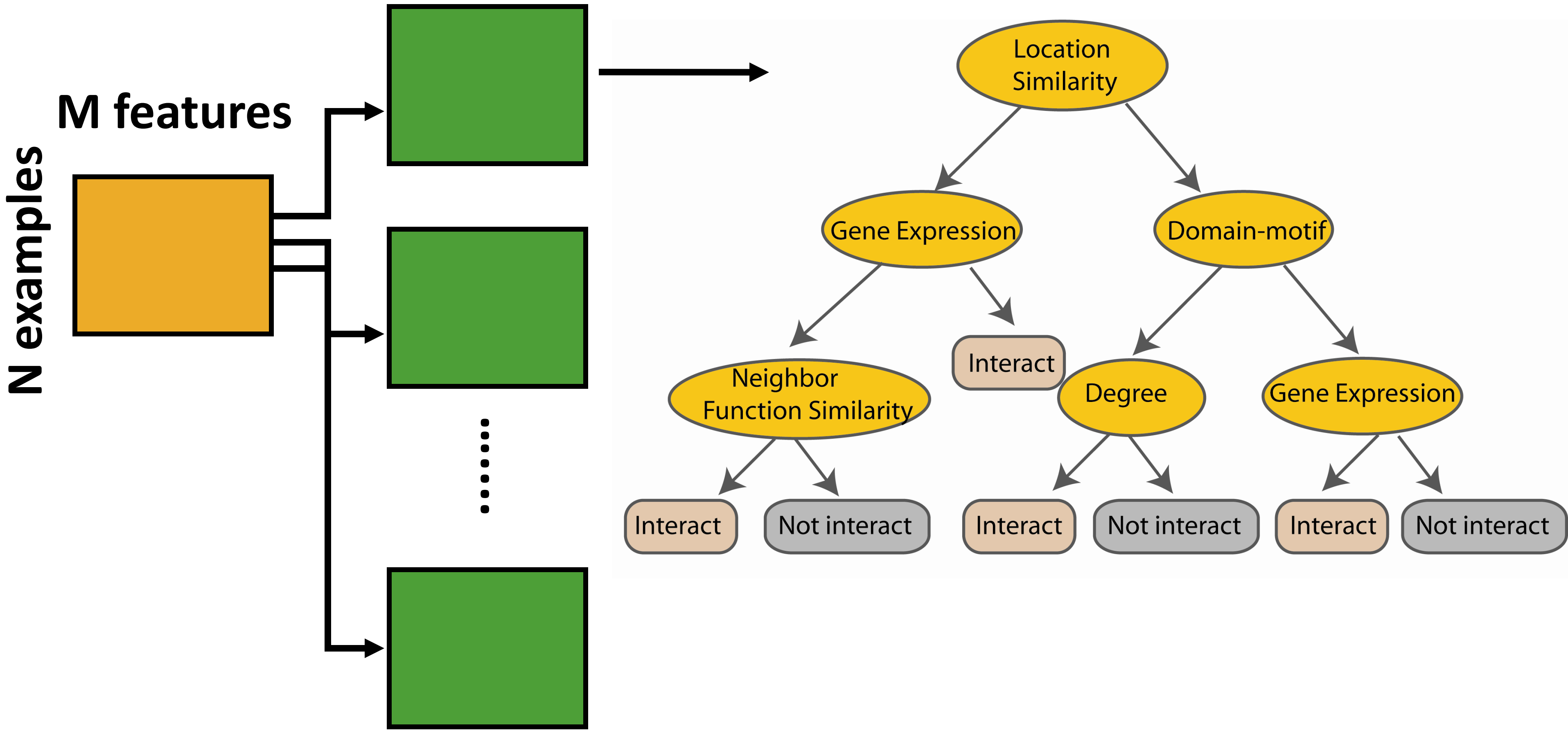


Random Forest Classifier

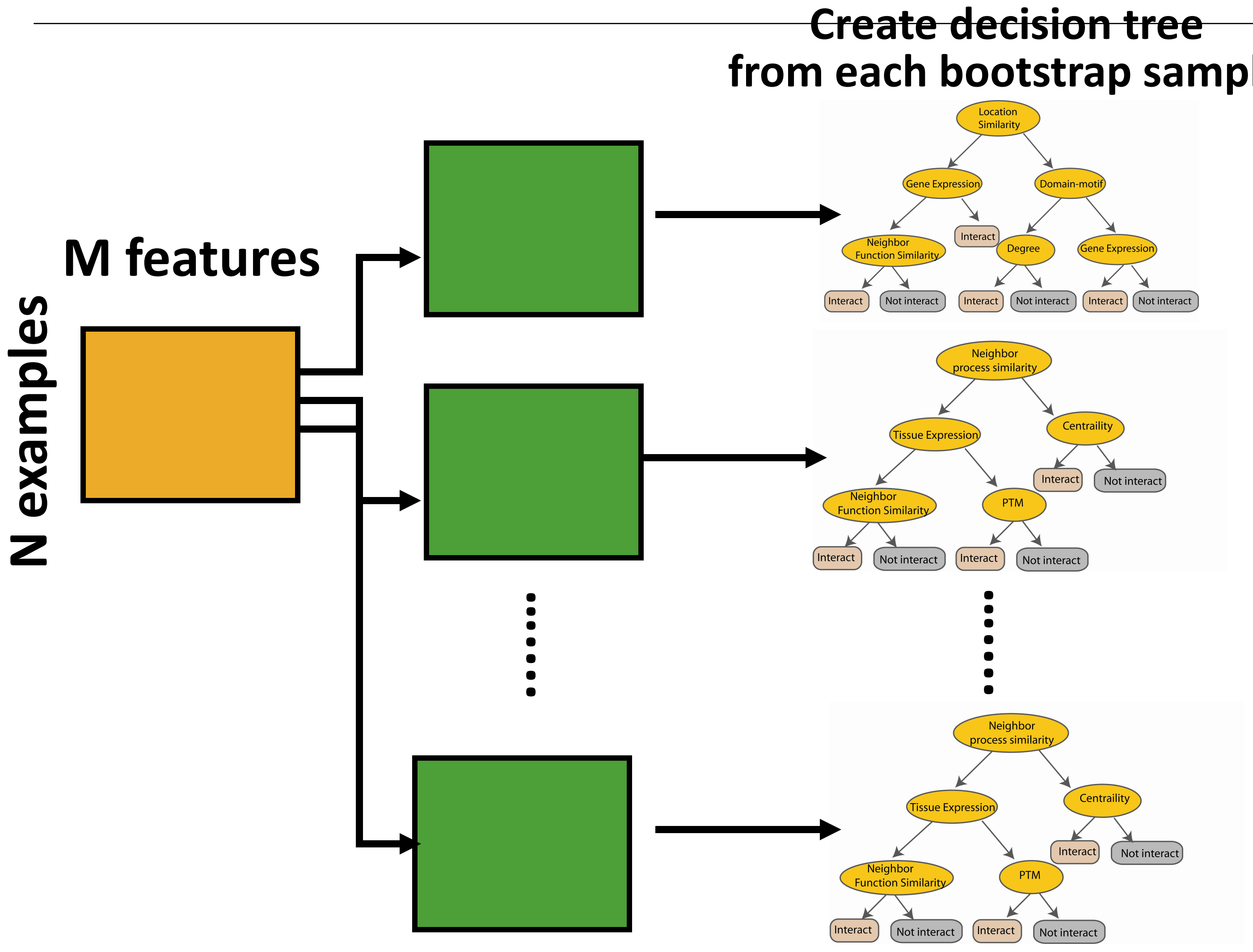


Random Forest Classifier

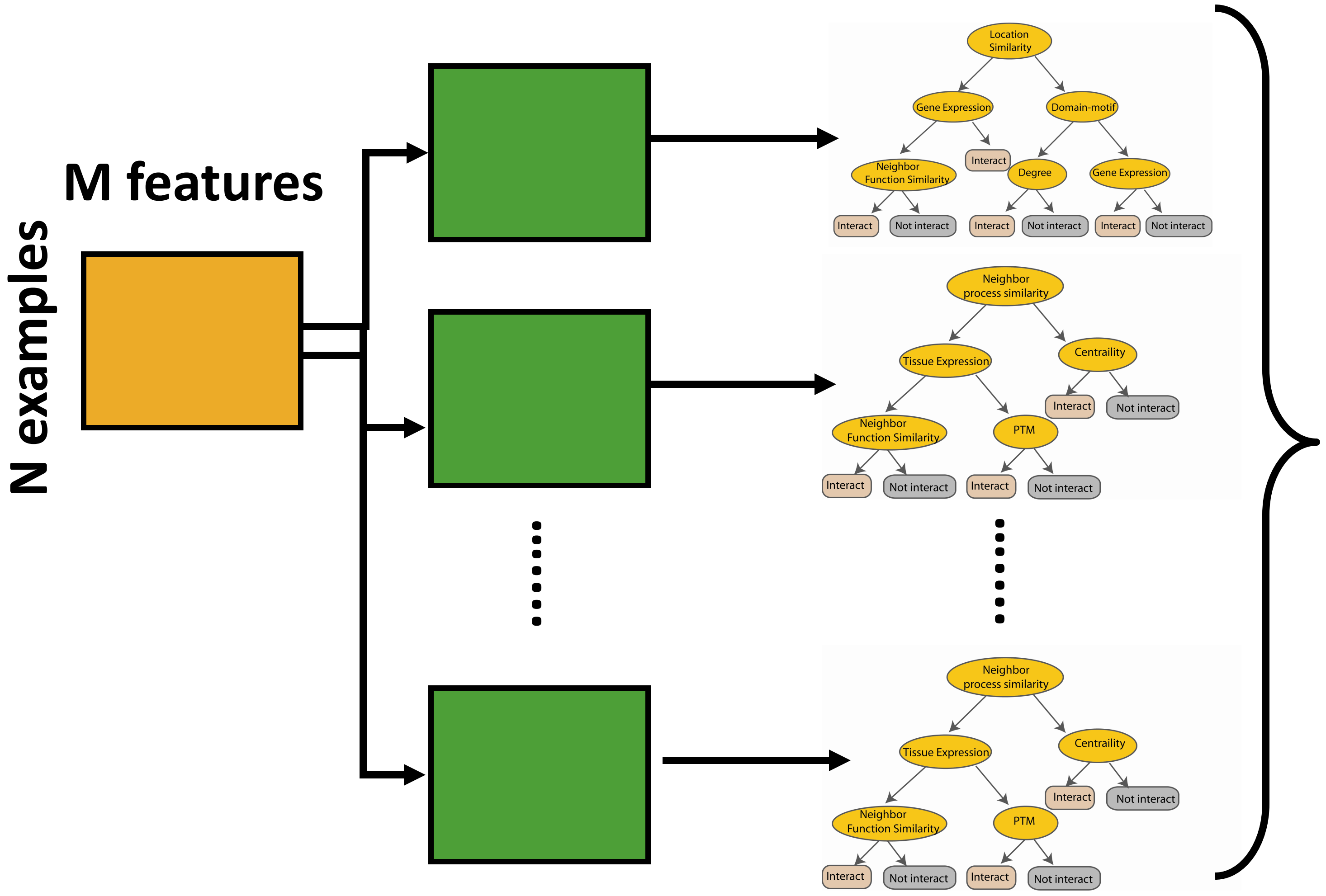
At each node in choosing the split feature choose only among *F<M* features



Random Forest Classifier



Random Forest Classifier



Take he majority vote

Characterizing the accuracy of RF

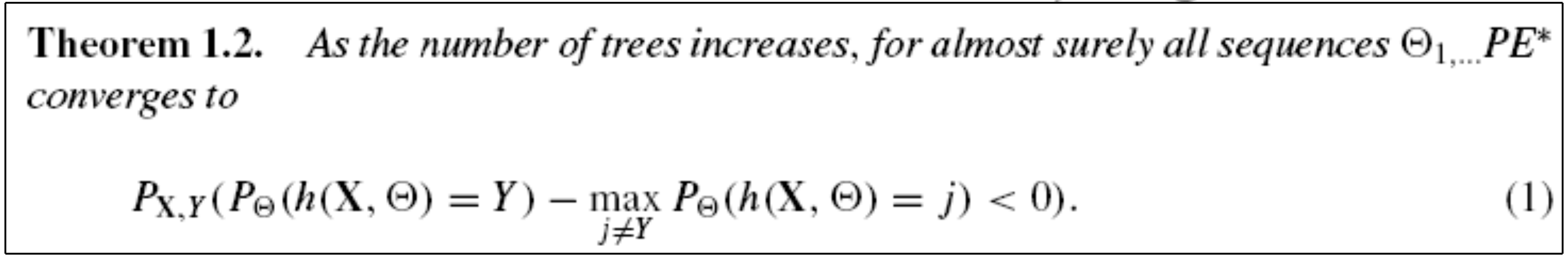
- **Margin function**:

$$mg(\mathbf{X}, Y) = av_k I(h_k(\mathbf{X}) = Y) - \max_{j \neq Y} av_k I(h_k(\mathbf{X}) = j)$$

which measures the extent to which the average number of votes at **X**,*Y* for the *right class* exceeds the average vote for *any other class*. The larger the margin, the more confidence in the classification.

- **Generalization error**:

$$PE^* = P_{\mathbf{X},Y}(mg(\mathbf{X}, Y) < 0)$$



Theorem 1.2. *As the number of trees increases, for almost surely all sequences $\Theta_{1,\ldots}$ PE^* converges to*

$$P_{\mathbf{X},Y}(P_\Theta(h(\mathbf{X}, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(\mathbf{X}, \Theta) = j) < 0). \quad (1)$$

Characterizing… (Cont.)

- Margin function for a random forest:

$$mr(\mathbf{X}, Y) = P_\Theta(h(\mathbf{X}, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(\mathbf{X}, \Theta) = j)$$

strength of the set of classifiers $\{h(\mathbf{x}, \Theta)\}$

$$s = E_{\mathbf{X},Y}\, mr(\mathbf{X}, Y)$$

suppose $\bar{\rho}$ s the mean value of **correlation**

Theorem 2.3. *An upper bound for the generalization error is given by*

$$PE^* \leq \bar{\rho}(1 - s^2)/s^2.$$

Definition 2.4. The c/s2 ratio for a random forest is defined as

$$c/s2 = \bar{\rho}/s^2.$$

the smaller,

the better

Literature

- Chapter 5 from the Tan et. al. Textbook.