Tree Based Methods (Ensemble Schemes)

Machine Learning Spring 2018

Feb 26 2018

Kasthuri Kannan

kasthuri.kannan@nyumc.org

Overview

- Decision Trees
    - Overview
    - Splitting nodes
    - Limitations
- Bagging/Bootstrap Aggregating and Boosting
    - How bagging reduces variance?
    - Boosting
- Random Forests
    - Overview
    - Why RF works?
- Cancer Genomics Application

Classification

- Given a collection of records (training set)
  - Each record contains a set of attributes, one of the attributes is the class/label

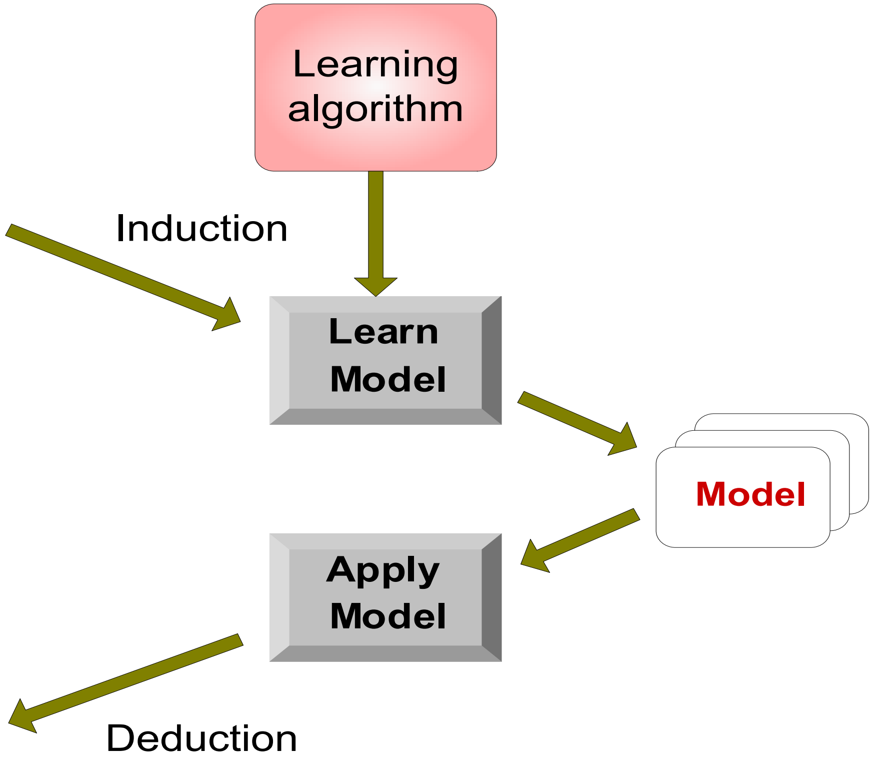
- Find a model for class attribute as a function of the values of other attributes.

- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A test set is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

Classification Illustration

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Courtesy: www.cs.kent.edu/~jin/DM07/

Classification Examples

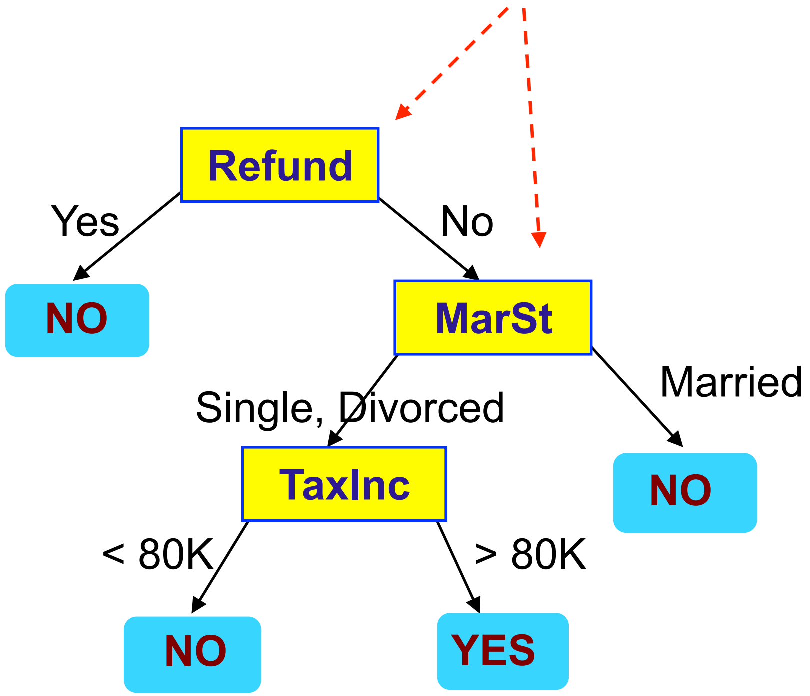
- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent
- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil
- Categorizing news stories as finance, weather, entertainment, sports, etc.
- Several algorithms – Decision trees, Support Vector Machines, Rule-based Methods etc.

Decision Tree (Example)

		categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**	
1	Yes	Single	125K	**No**	
2	No	Married	100K	**No**	
3	No	Single	70K	**No**	
4	Yes	Married	120K	**No**	
5	No	Divorced	95K	**Yes**	
6	No	Married	60K	**No**	
7	Yes	Divorced	220K	**No**	
8	No	Single	85K	**Yes**	
9	No	Married	75K	**No**	
10	No	Single	90K	**Yes**	

Training Data

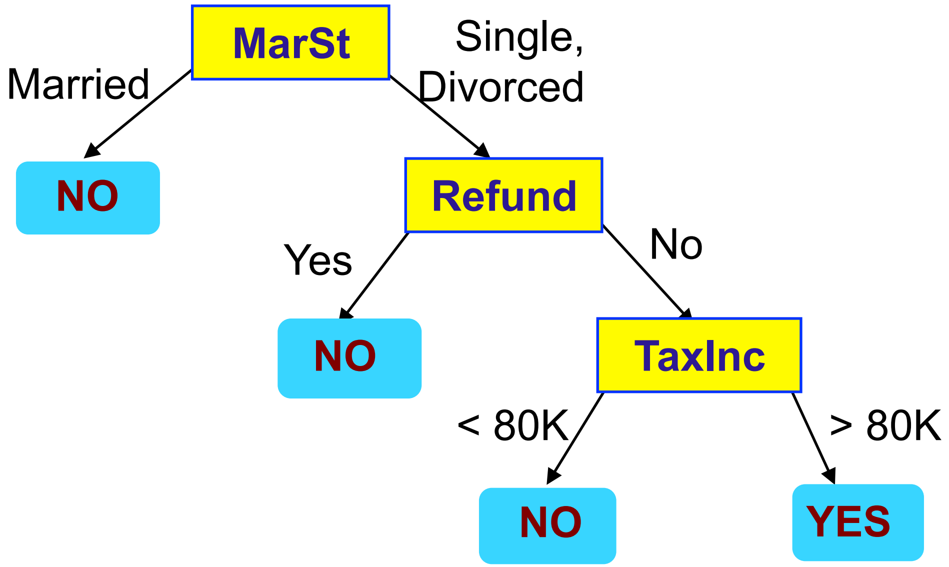
Splitting Attributes



Model: Decision Tree

Decision Tree (Another Example)

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**



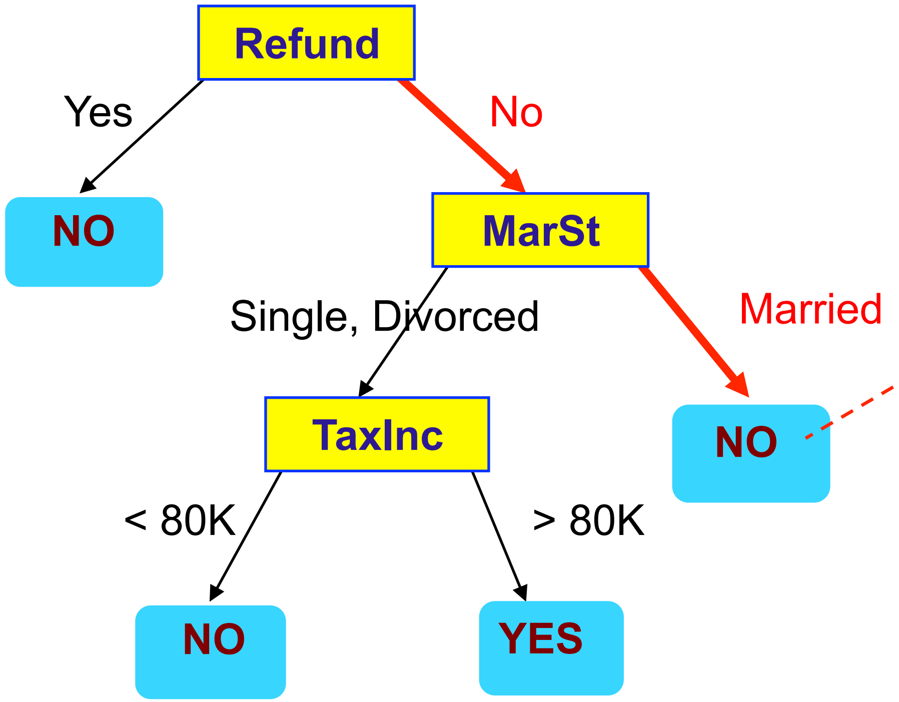
There could be more than one tree that fits the same data!

Applying the model

Once the decision tree is built, the model can be used to test an unclassified data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

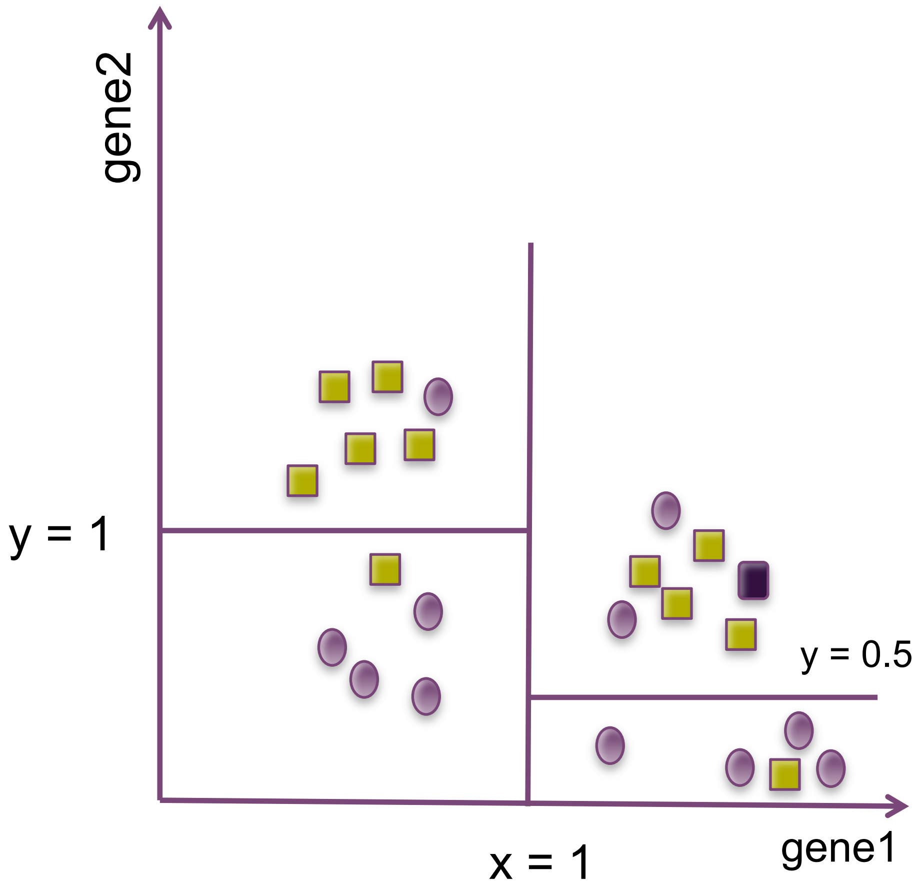


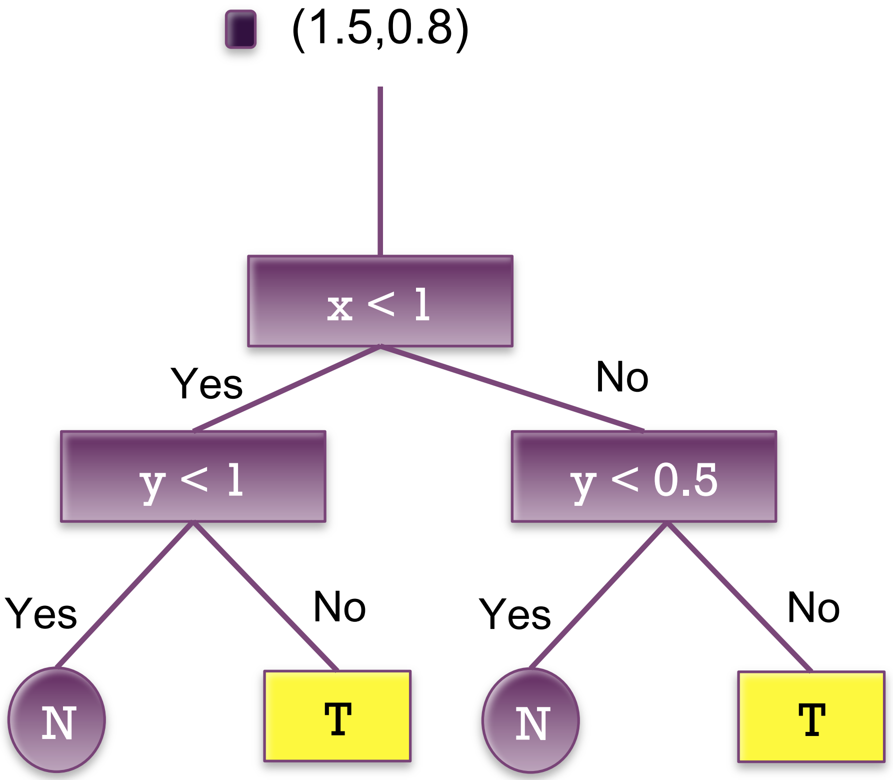
Assign Cheat to “No”

Tumor Classification (Example)

🟨 Tumor samples/patients given by expression in (gene1,gene2)

🟣 Normal samples/patients given by expression in (gene1,gene2)





Decision Tree Algorithms

- Many Algorithms:
  - Hunt' s Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ,SPRINT

Hunt's Algorithm (General Idea)

Let D_t be the set of training records that reach a node t

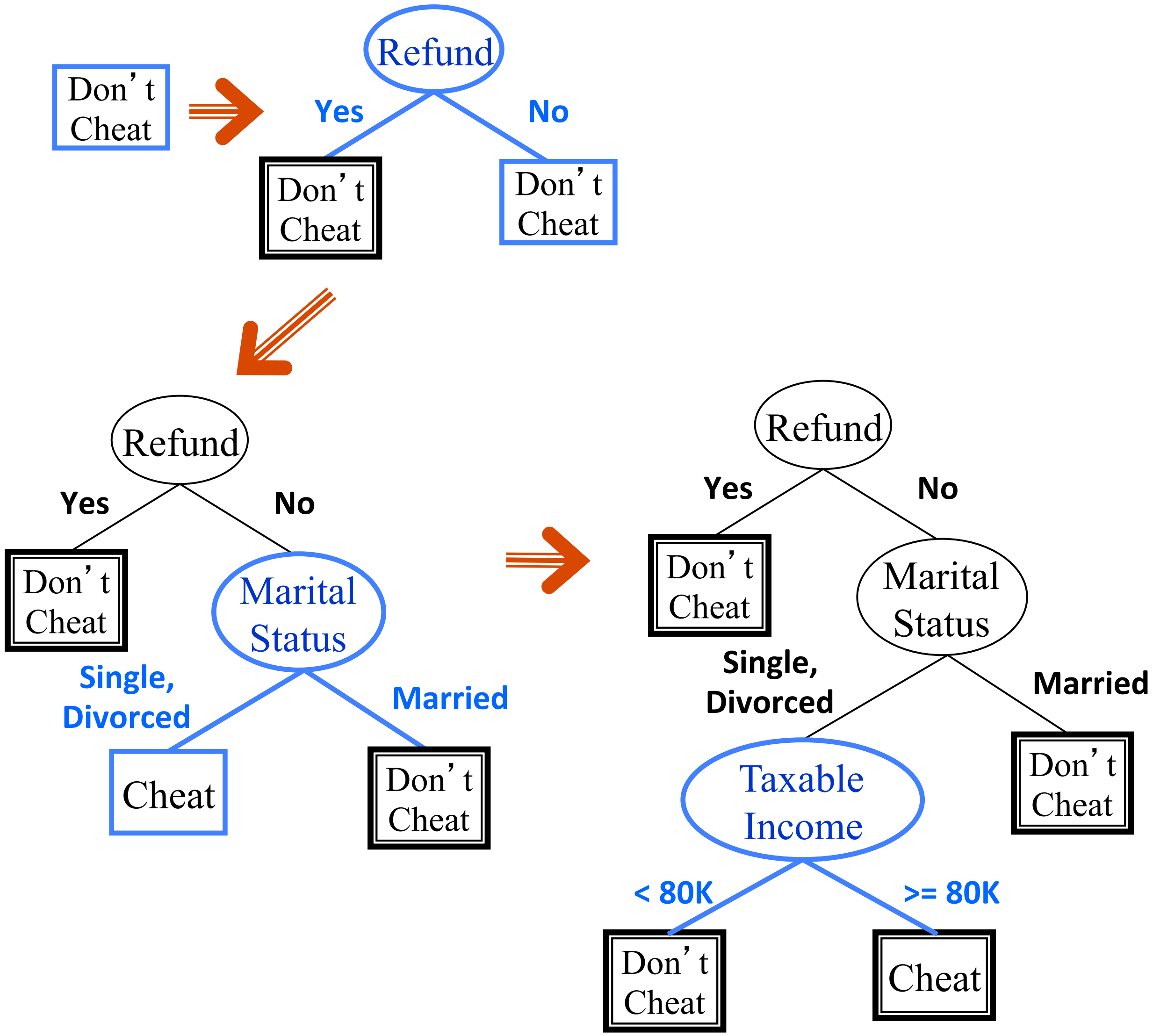
General Procedure:

If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t

If D_t is an empty set, then t is a leaf node labeled by the default class, y_d

If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

Hunt’s Algorithm (Illustration)



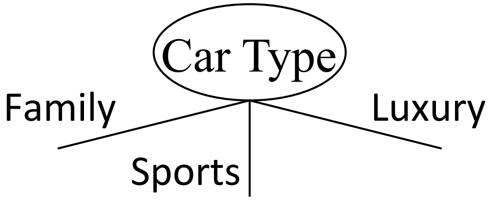
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Tree Induction

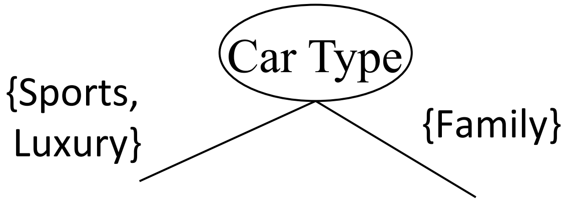
- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
- Main questions
  - Determine how to split the records
    - Binary or multi-way split?
  - How to determine the best split?
  - Determine when to stop splitting

Test Condition (Splitting Based on Nominal/Ordinal Attributes)

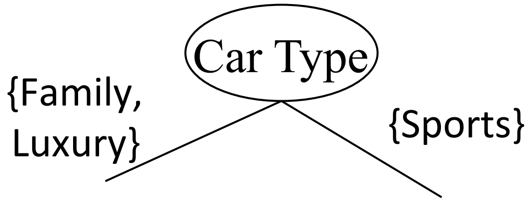
Multi-way split: Use as many partitions as distinct values.



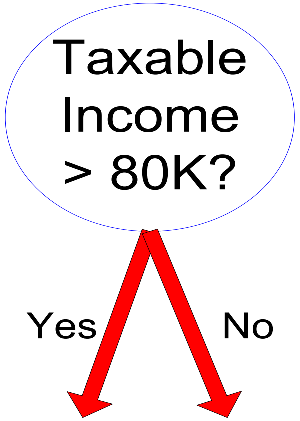
Binary split: Divides values into two subsets.
Need to find optimal partitioning.



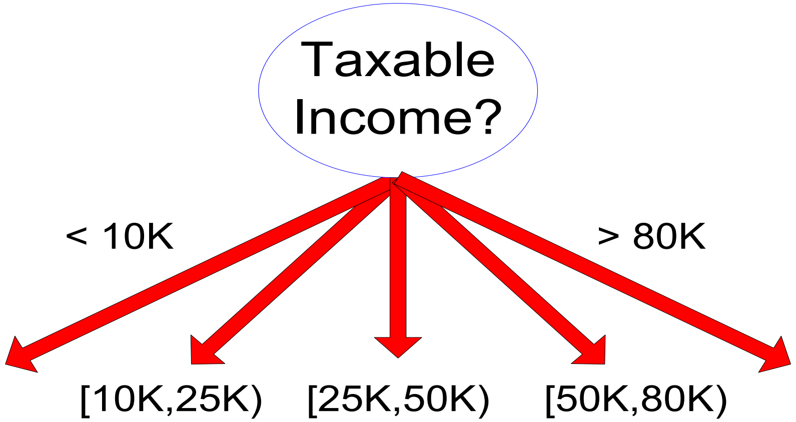
OR



Test Condition (Splitting Based on Continuous Attributes)



(i) Binary split



(ii) Multi-way split

Binary vs. Multi-way split – which is the best?

Seven-class, 4-feature classification problem

Apple = (green AND medium) OR (red AND medium) = (Medium AND NOT yellow)

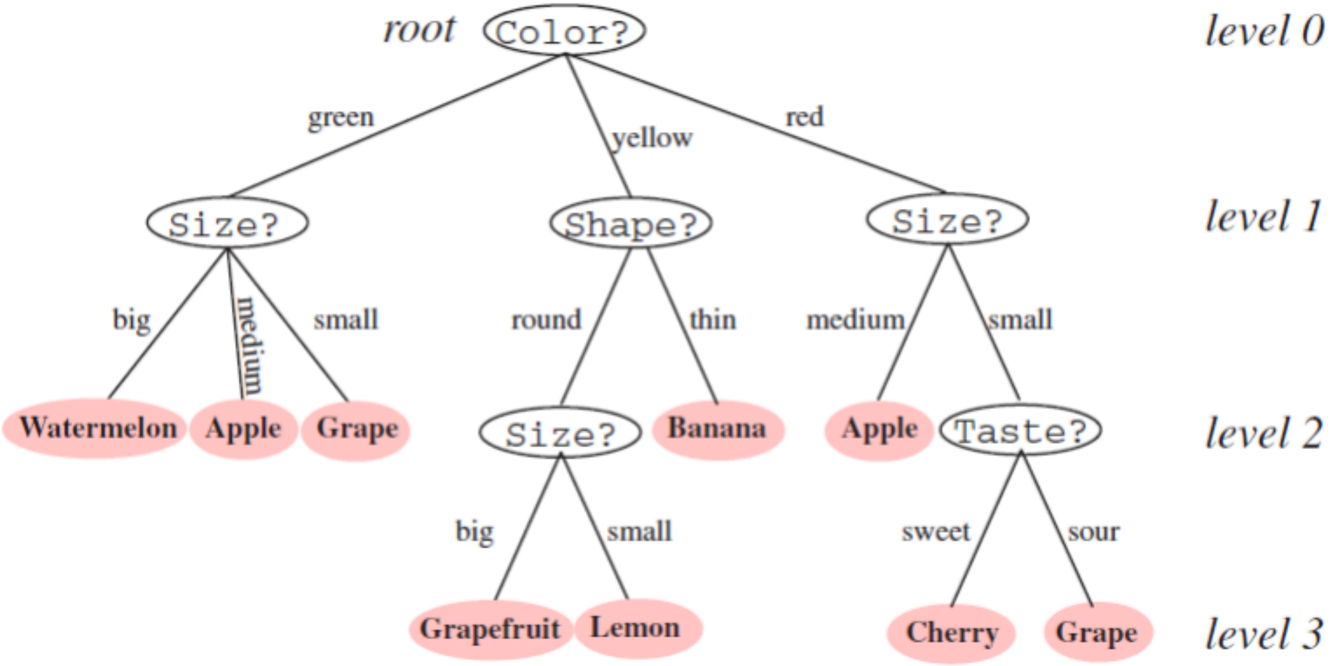


Figure 8.1: Classification in a basic decision tree proceeds from top to bottom. The questions asked at each node concern a particular property of the pattern, and the downward links correspond to the possible values. Successive nodes are visited until a terminal or leaf node is reached, where the category label is read. Note that the same question, Size?, appears in different places in the tree, and that different questions can have different numbers of branches. Moreover, different leaf nodes, shown in pink, can be labeled by the same category (e.g., **Apple**).

http://www.cse.msu.edu/~cse802/DecisionTrees.pdf

Binary vs. Multi-way split – which is the best?

Seven-class, 4-feature classification problem

Apple = (green AND medium) OR (red AND medium) = (Medium AND NOT yellow)

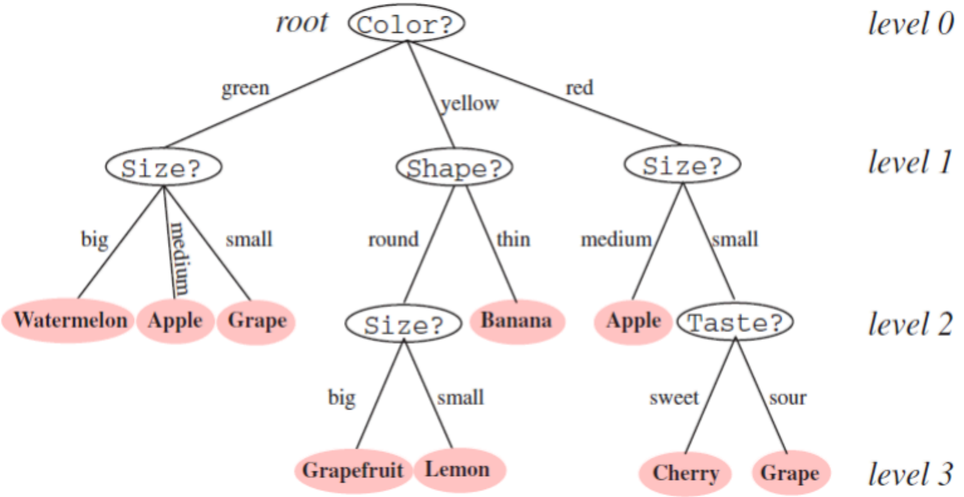


Figure 8.1: Classification in a basic decision tree proceeds from top to bottom. The questions asked at each node concern a particular property of the pattern, and the downward links correspond to the possible values. Successive nodes are visited until a terminal or leaf node is reached, where the category label is read. Note that the same question, Size?, appears in different places in the tree, and that different questions can have different numbers of branches. Moreover, different leaf nodes, shown in pink, can be labeled by the same category (e.g., **Apple**).

Binary tree: every decision can be represented using just binary outcome; tree of Fig 8.1 can be equivalently written as

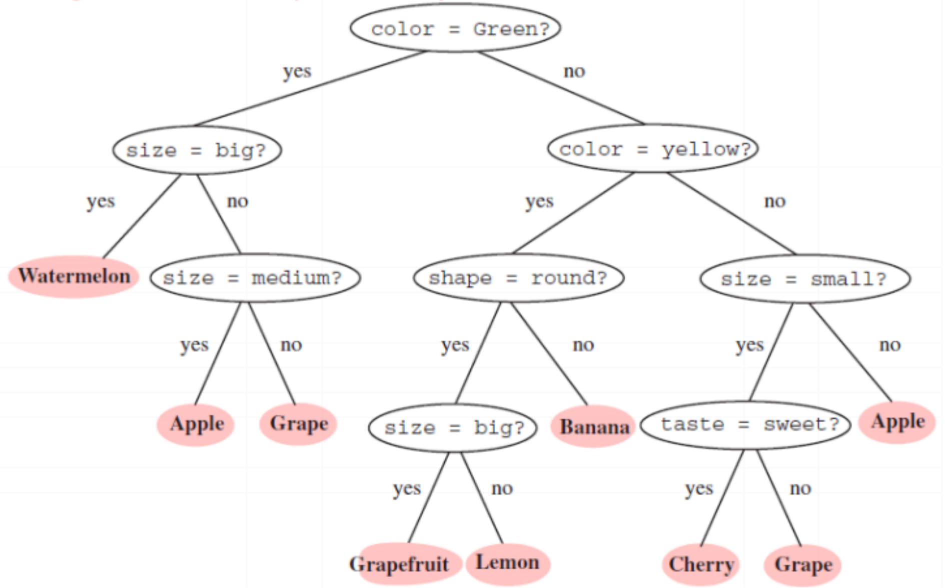
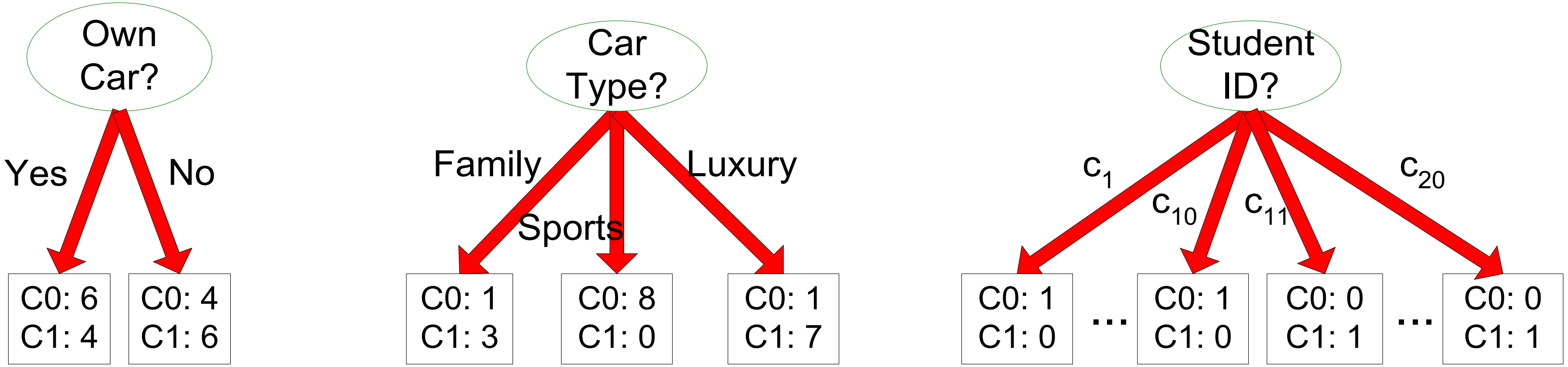


Figure 8.2: A tree with arbitrary branching factor at different nodes can always be represented by a functionally equivalent binary tree, i.e., one having branching factor $B = 2$ throughout. By convention the "yes" branch is on the left, the "no" branch on the right. This binary tree contains the same information and implements the same classification as that in Fig. 8.1.

http://www.cse.msu.edu/~cse802/DecisionTrees.pdf

Determining Best Split

Before Splitting: 10 records of class 0,
10 records of class 1



Which attribute is the best for splitting?

Determining Best Split

- Greedy approach:
  - Nodes with homogeneous class distribution are preferred

- Need a measure of node impurity:

C0: 5
C1: 5

Non-homogeneous,

High degree of impurity

C0: 9
C1: 1

Homogeneous,

Low degree of impurity

Measures of Node Impurity

- Gini Index

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

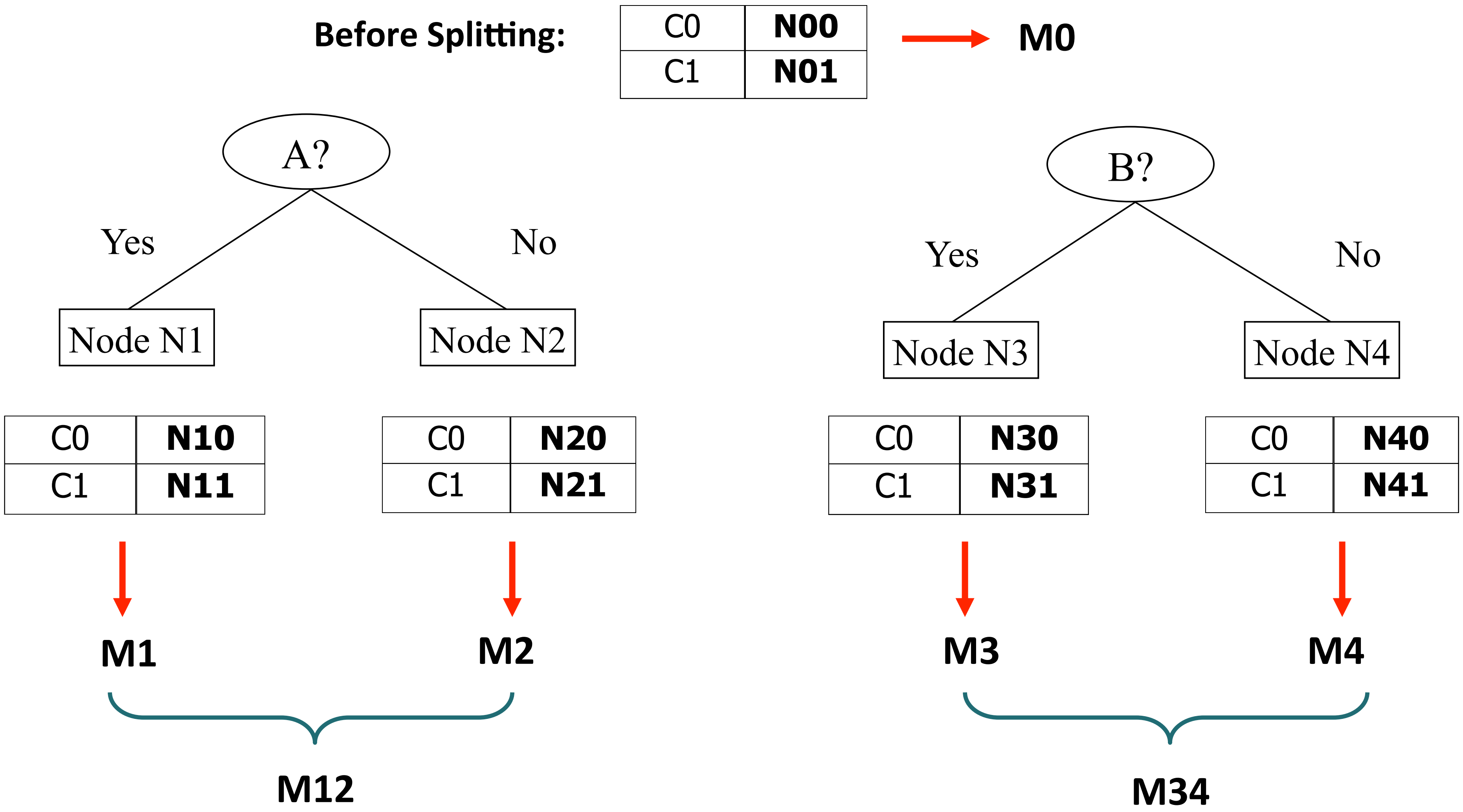
- Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

- Misclassification error

$$Error(t) = 1 - \max_i P(i \mid t)$$

Computing Measures of Node Impurity



Before Splitting:

C0	**N00**
C1	**N01**

→ **M0**

C0	**N10**
C1	**N11**

C0	**N20**
C1	**N21**

C0	**N30**
C1	**N31**

C0	**N40**
C1	**N41**

Gain = M0 – M12 vs M0 – M34

Computing Measures of Node Impurity (Gini Index)

- Gini Index for a given node t :

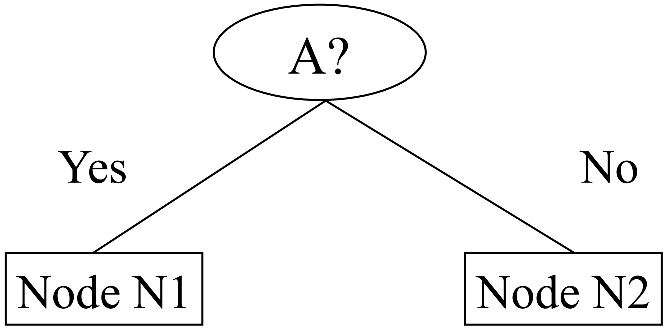
$$GINI(t) = 1 - \sum_{j} [p(j \mid t)]^2$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t)

  - Maximum (1 - 1/n_c) when records are equally distributed among all classes, implying least interesting information

  - Minimum (0.0) when all records belong to one class, implying most interesting information

Example (Gini Index of a Node)



C0	**0**
C1	**6**

↓

M1

C0	**1**
C1	**5**

↓

M2

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

M1

C0	**0**
C1	**6**

P(C0) = 0/6 = 0 P(C1) = 6/6 = 1

Gini = 1 – P(C0)^2 – P(C1)^2 = 1 – 0 – 1 = 0

M2

C0	**1**
C1	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – (1/6)^2 – (5/6)^2 = 0.278

Splitting based on Gini Index

- Used in CART, SLIQ, SPRINT

- When a node *p* is split into *k* partitions (children), the quality of split is computed as,

$$Gini_{split} = Gini(Parent) - \frac{n_L}{n_{parent}} Gini(Left) - \frac{n_R}{n_{parent}} Gini(Right)$$

where, n_L = number of records at the left child node,

n_R = number of records at the right child node

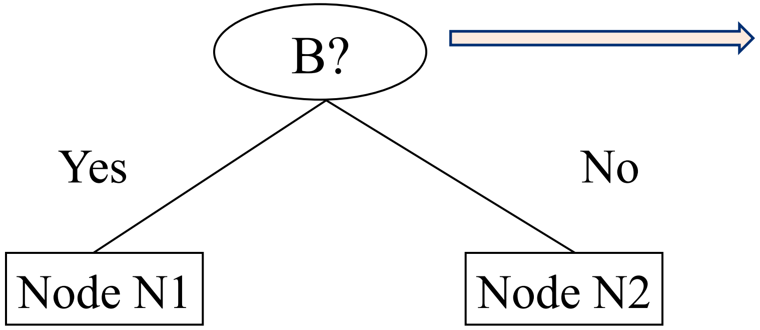
- Split on the attribute that maximizes $Gini_{split}$

Splitting Based on Gini Index

- Splits into two partitions

$$Gini_{split} = Gini(Parent) - \frac{n_L}{n_{parent}} Gini(Left) - \frac{n_R}{n_{parent}} Gini(Right)$$

- Effect of Weighing partitions:
  - Larger and purer partitions are sought for.



	Parent
C1	**6**
C2	**6**
Gini = 0.500	

Gini(N1)
= 1 – $(5/6)^2$ – $(2/6)^2$
= 0.194

Gini(N2)
= 1 – $(1/6)^2$ – $(4/6)^2$
= 0.528

	N1	**N2**
C1	**5**	**1**
C2	**2**	**4**
Gini=0.333		

Gini(Children)
= 7/12 * 0.194 +
5/12 * 0.528
= 0.333

$Gini_{split(B)}$ = 0.5-0.3 = 0.2

Measures of Node Impurity (Entropy)

- Entropy at a given node t:

$$Entropy(t) = -\sum_{j} p(j\,|\,t)\log p(j\,|\,t)$$

  - (NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

- Measures homogeneity of a node.
  - Maximum (log n_c) when records are equally distributed among all classes implying least information
  - Minimum (0.0) when all records belong to one class, implying most information
- Entropy based computations are similar to the GINI index computations

Example (Entropy)

C1	**0**
C2	**6**

$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$

$\text{Entropy} = -\,0 \log 0 - 1 \log 1 = -\,0 - 0 = 0$

C1	**1**
C2	**5**

$P(C1) = 1/6 \qquad P(C2) = 5/6$

$\text{Entropy} = -\,(1/6) \log_2 (1/6) - (5/6) \log_2 (1/6) = 0.65$

C1	**2**
C2	**4**

$P(C1) = 2/6 \qquad P(C2) = 4/6$

$\text{Entropy} = -\,(2/6) \log_2 (2/6) - (4/6) \log_2 (4/6) = 0.92$

Splitting Based on Entropy: Information Gain

- Information Gain:

$$\boxed{GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)}$$

Parent Node, *p* is split into *k* partitions;

n_i is number of records in partition *I*

- Measures reduction in entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
    - Used in ID3 and C4.5
    - Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Measures of Node Impurity (Misclassification Error)

- Classification error at a node t :

$$\boxed{Error(t) = 1 - \max_{i} P(i \mid t)}$$

- Measures misclassification error made by a node.
  - Maximum (1 - 1/n_c) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information
- Misclassification based computations are similar to the GINI index and entropy computations

$$MC_{split} = MC(Parent) - \frac{n_L}{n_{parent}} MC(Left) - \frac{n_R}{n_{parent}} MC(Right)$$

Example (Misclassification Error)

$$\boxed{Error(t) = 1 - \max_{i} P(i \mid t)}$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

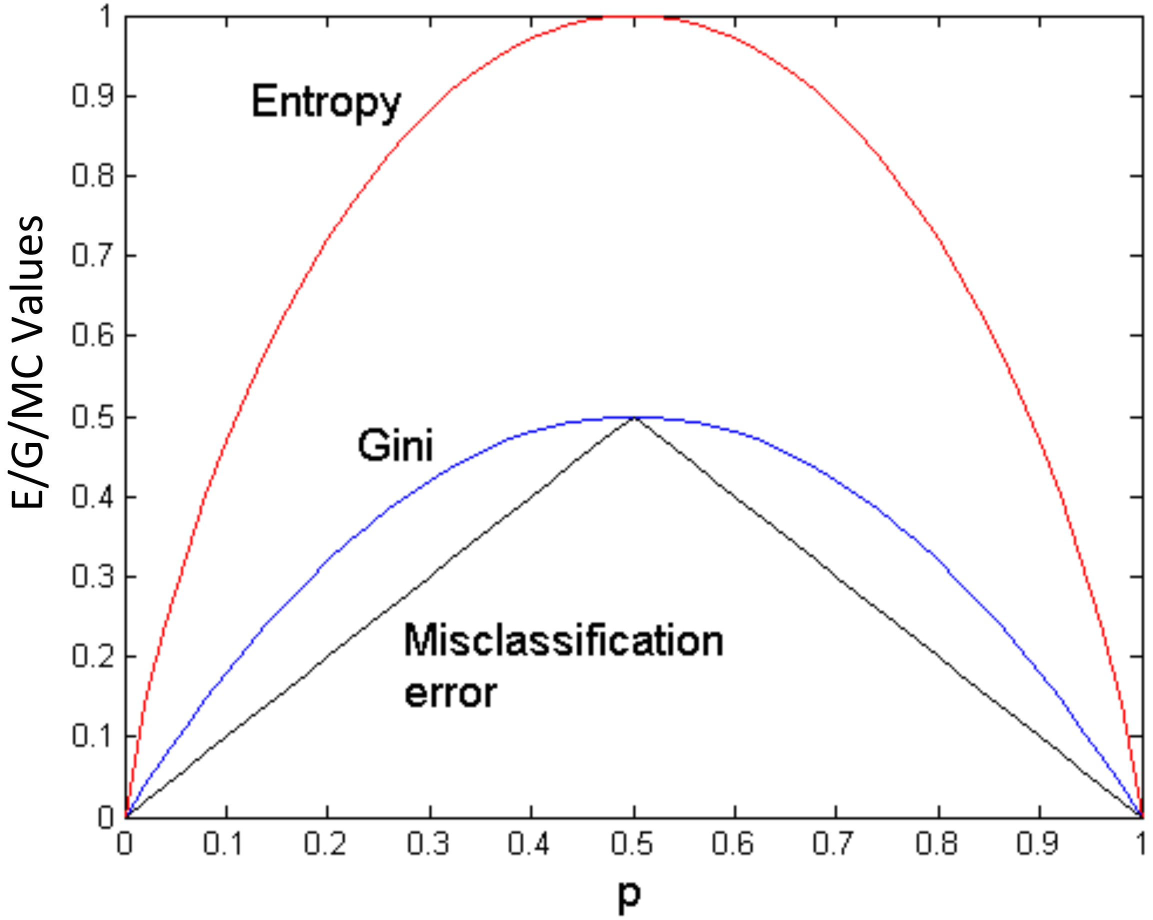
C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison (Splitting Criteria)

Two-class problem

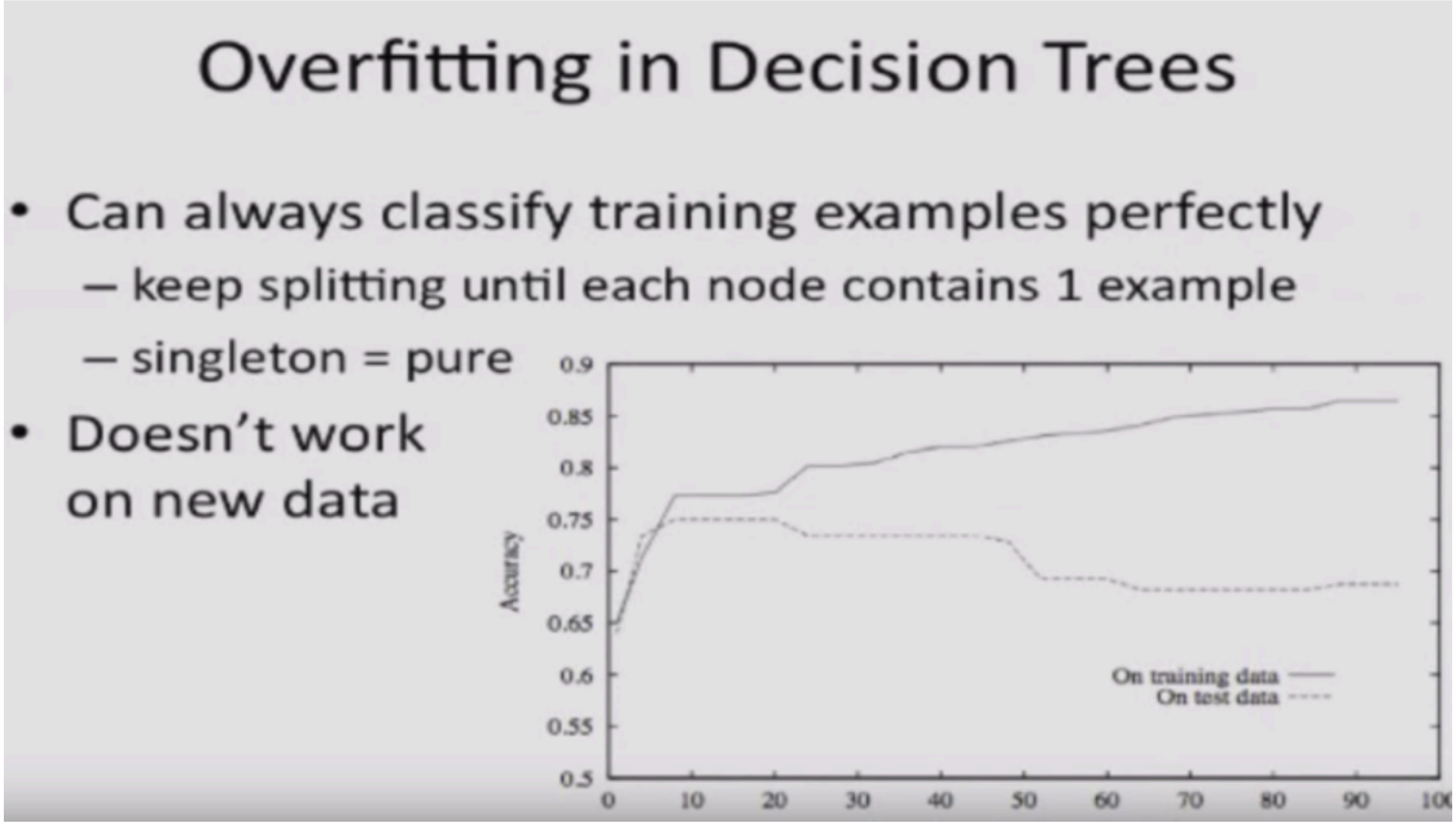


Summary

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
- Questions
  - **Determine how to split the records**
    - **Binary or multi-way split? Multi-way is same as binary split.**
  - **How to determine the best splitting features? Gini/Entropy/Misclassification**
  - **When to stop splitting?**

When to stop splitting?

- A maximal node depth is reached
    - If splitting is stopped too early, error on training data is not sufficiently low and performance on the test data will suffer (underfitting)
- The leaf nodes doesn’t have any impurity
    - If we continue to grow the tree fully until each leaf node corresponds to the lowest impurity, then the data have typically been overfit; in the limit, each leaf node has only one pattern!



Overfitting in Decision Trees

- Can always classify training examples perfectly
  - keep splitting until each node contains 1 example
  - singleton = pure
- Doesn’t work on new data

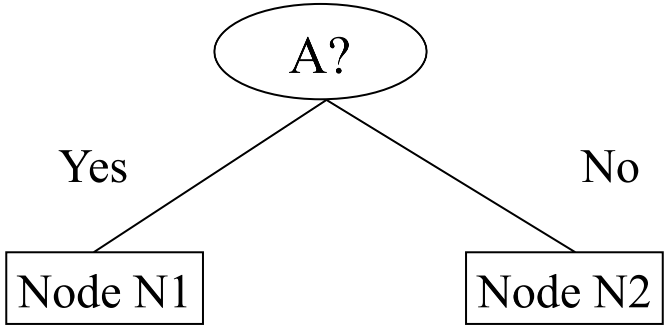
# of nodes

Tom Mitchell, 1997

When to stop splitting?

- Stop splitting when the best candidate split at a node reduces the impurity by less than the preset amount (threshold)
    - How to set the threshold?
- Splitting a note does not lead to an information gain
    - Depends on the metric used for information gain

Comparison (Gini Vs. Misclassification)



	Parent
C1	**7**
C2	**3**
Gini = 0.42	

MC(Parent) = 1- max(7/10,3/10)
= 1-7/10 = 3/10

Gini(N1)
= 1 – $(3/3)^2$ – $(0/3)^2$
= 0

	N1	**N2**
C1	**3**	**4**
C2	**0**	**3**
Gini = 0.34		

Gini(N2)
= 1 – $(4/7)^2$ – $(3/7)^2$
= 0.489

Gini(Children)
= 3/10 * 0
+ 7/10 * 0.489
= 0.342

$\text{Gini}_{\text{split(A)}}$ = 0.42 – 0.34 = 0.08

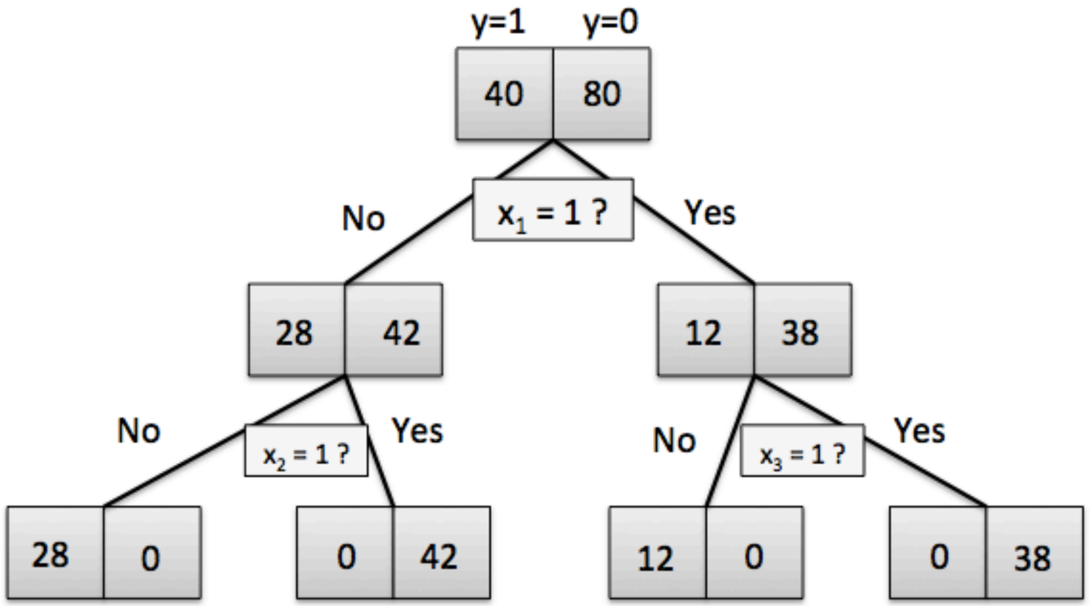
$$MC_{split} = MC(Parent) - \frac{n_L}{n_{parent}} MC(Left) - \frac{n_R}{n_{parent}} MC(Right)$$

MC(N1) = 1- max(3/3,0/3) = 1-1 = 0
MC(N2) = 1-max(4/7,3/7) = 1-4/7 = 3/7

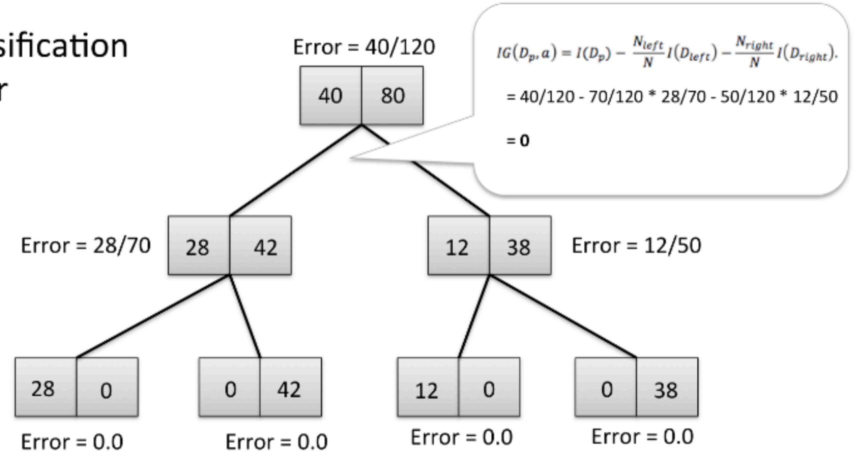
$\text{Mc}_{\text{split(A)}}$ = 3/10 - (3/10)0-(7/10)(3/7) = 0

Comparison (Entropy Vs. Misclassification)

Original Tree



Classification Error

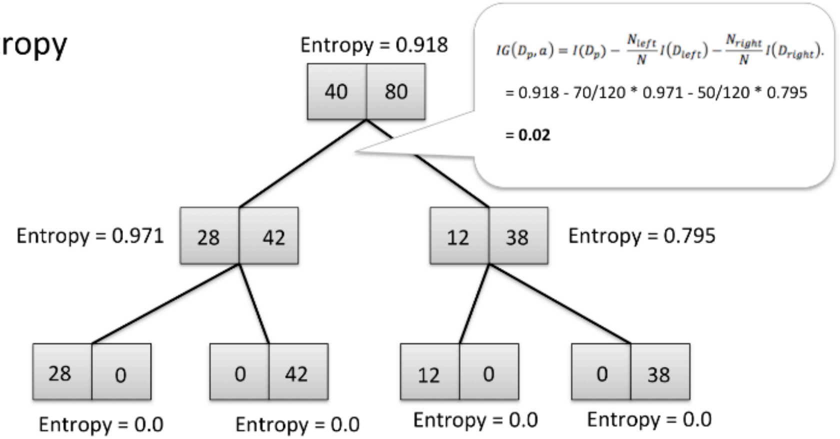


$IG(D_p, a) = I(D_p) - \frac{N_{left}}{N} I(D_{left}) - \frac{N_{right}}{N} I(D_{right}).$

= 40/120 - 70/120 * 28/70 - 50/120 * 12/50

= 0

Entropy



$IG(D_p, a) = I(D_p) - \frac{N_{left}}{N} I(D_{left}) - \frac{N_{right}}{N} I(D_{right}).$

= 0.918 - 70/120 * 0.971 - 50/120 * 0.795

= 0.02

https://sebastianraschka.com/faq/docs/decisiontree-error-vs-entropy.html

Advantages & Disadvantages

- **Inexpensive to construct if classifying features are given or known**

- **Extremely fast at classifying unknown records**

- **Easy to interpret for small-sized trees**

- **Accuracy is comparable to other classification techniques for many simple data sets**

- **Underfitting and Overfitting**

- **Costs of classification high as tree construction is combinatorial in features**

- **Sensitive to data - high variance – less accurate prediction**

When to stop splitting?

- Use validation/cross-validation techniques
    - Continue splitting until error on validation set is minimum
    - Cross-validation relies on several independently chosen subsets
- Cross-validation methods
    - Hold-out method (50-70 % Training, 50-30 % testing/validation)
        - Prone to selection bias
    - K-fold cross-validation
    - Leave one out cross-validation (lots of computation time)
    - Bootstrap methods

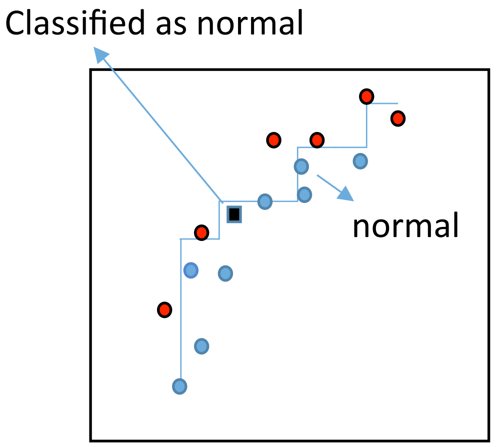
Bootstrap Aggregation

- Key variance reduction technique

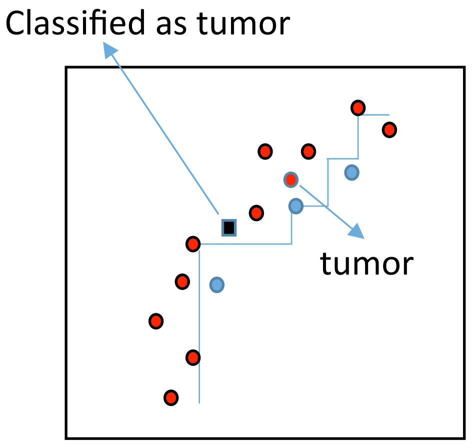
- If each classifier has a high variance the aggregated classifier has a smaller variance than each single classifier

- The bagging classifier is like an approximation of the true average computed by replacing the probability distribution with bootstrap approximation

Variance



Classifier 1



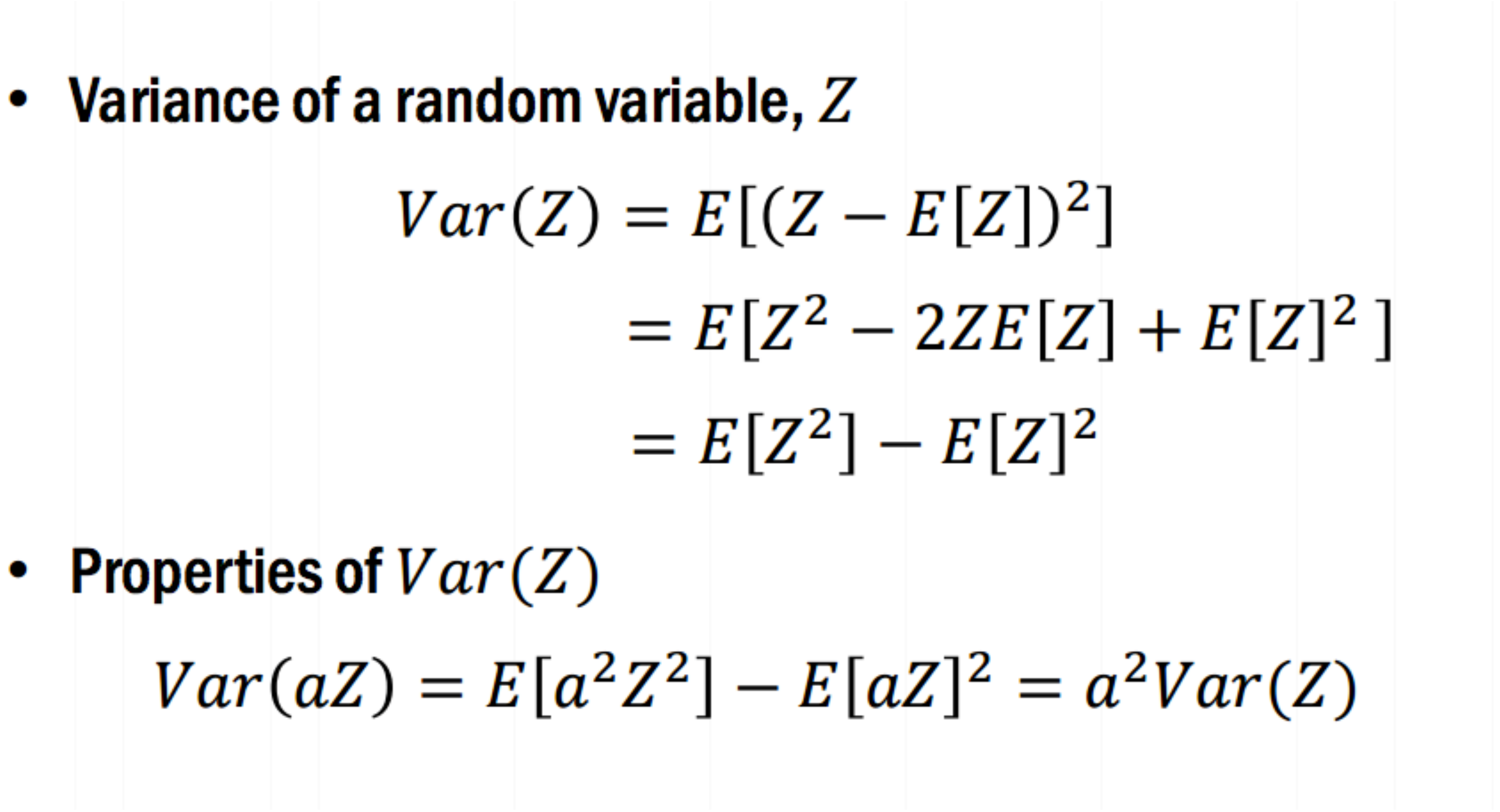
Classifier 2

Small change in the data could result in a radically different tree structure

Large variance in classification

Variance reduces by aggregating/averaging

Why Bagging Works?



- **Variance of a random variable, Z**

$$Var(Z) = E[(Z - E[Z])^2]$$
$$= E[Z^2 - 2ZE[Z] + E[Z]^2]$$
$$= E[Z^2] - E[Z]^2$$

- **Properties of $Var(Z)$**

$$Var(aZ) = E[a^2Z^2] - E[aZ]^2 = a^2Var(Z)$$

Why Bagging Works?

$$(x_1^1, y_1^1), (x_2^1, y_2^1), \cdots, (x_k^1, y_k^1) \quad \leftrightarrow (x, y_1)$$

$$(x_1^2, y_1^2), (x_2^2, y_2^2), \cdots, (x_k^2, y_k^2) \quad \leftrightarrow (x, y_2)$$

$$\vdots$$

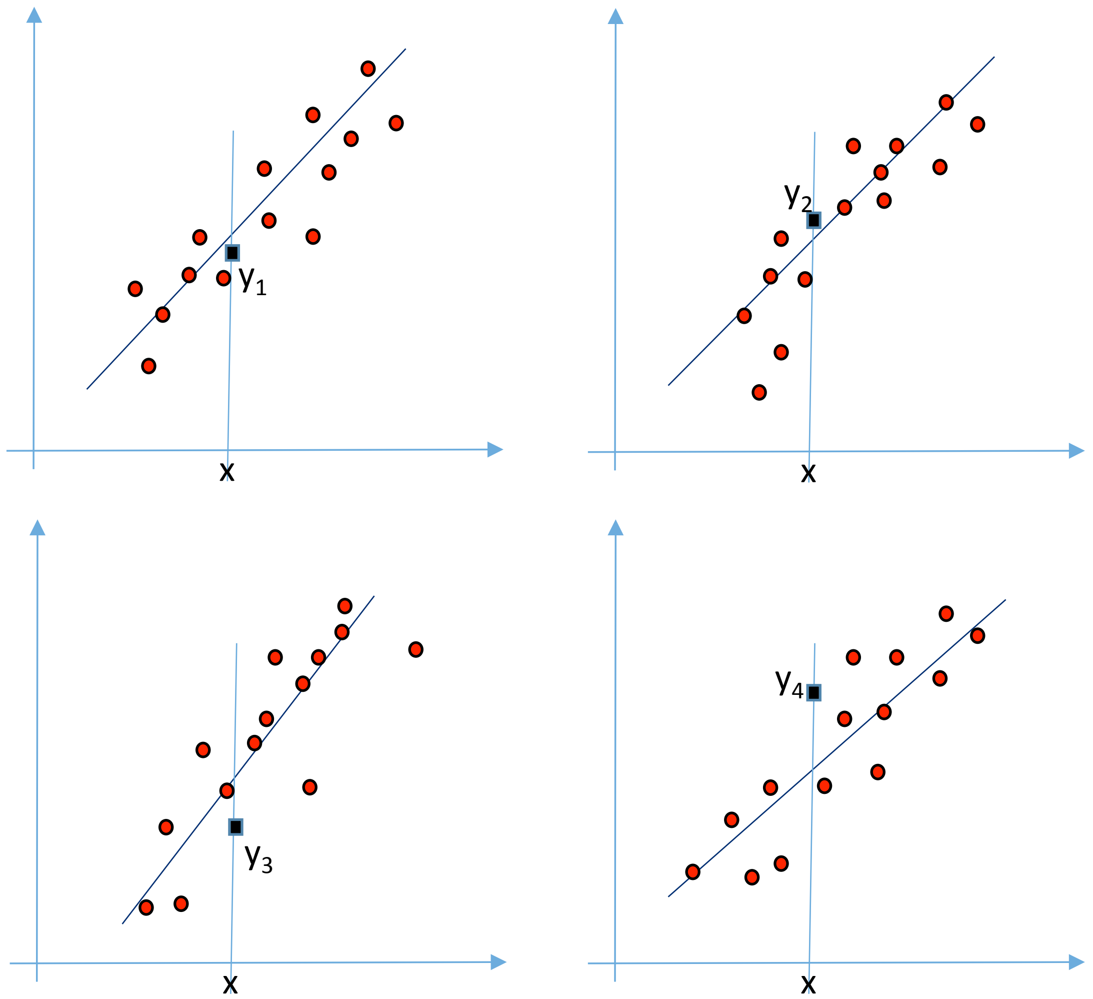
$$(x_1^m, y_1^m), (x_2^m, y_2^m), \cdots, (x_k^m, y_k^m) \leftrightarrow (x, y_m)$$

$$E[y_i] = y \quad Var[y] = E[(y_i - E[y_i])^2] = E[(y_i - y)^2]$$

$$Z = \frac{1}{m} \sum y_i$$

$$E[Z] = \frac{1}{m} \sum E[y_i] = \frac{1}{m}(mE[y]) = E[y]$$

$$Var[Z] = Var[\frac{1}{m} \sum y_i] = \frac{1}{m^2} Var[\sum y_i]$$



If y_i's are **uncorrelated**

$$\frac{1}{m^2} Var[\sum y_i] = \frac{1}{m^2} \sum Var[y_i]$$

$$\frac{1}{m^2} Var[\sum y_i] = \frac{1}{m^2} \sum Var[y_i] = \frac{1}{m^2} \sum (E[(y_i^2 - E[y_i])^2] = \frac{1}{m^2} \sum (E[(y_i^2 - y)^2] = \frac{1}{m^2} \sum Var[y] = \frac{1}{m} Var[y]$$

Why bagging works?

$$E[Z] = E[y] = y$$
$$Var[Z] = \frac{1}{m} Var[y]$$

- As m increases, variance is reduced and the aggregated prediction is closer to the true value.

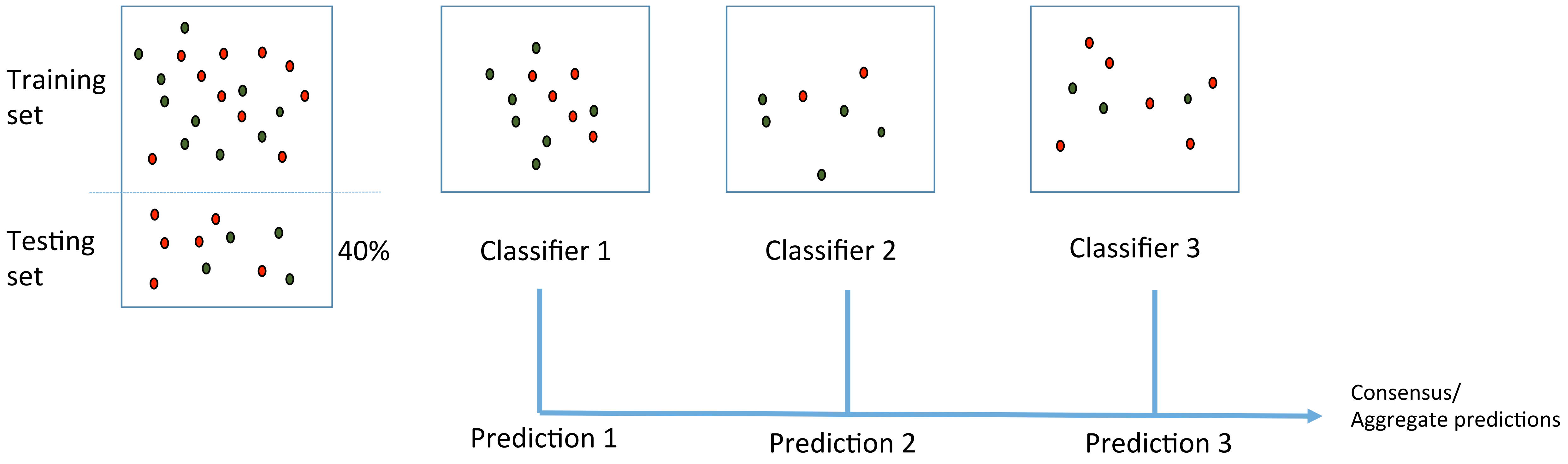
- Unfortunately, we DON'T have several sets of samples!!!

- What should we do?

- Of course, make a uniform sampling from given sample set, with replacement and use each sample set as a bootstrap sample set!!!

Ensemble Classifiers

- Subsample with replacement
- Uniformly subsample



Bagging (Error Curves)

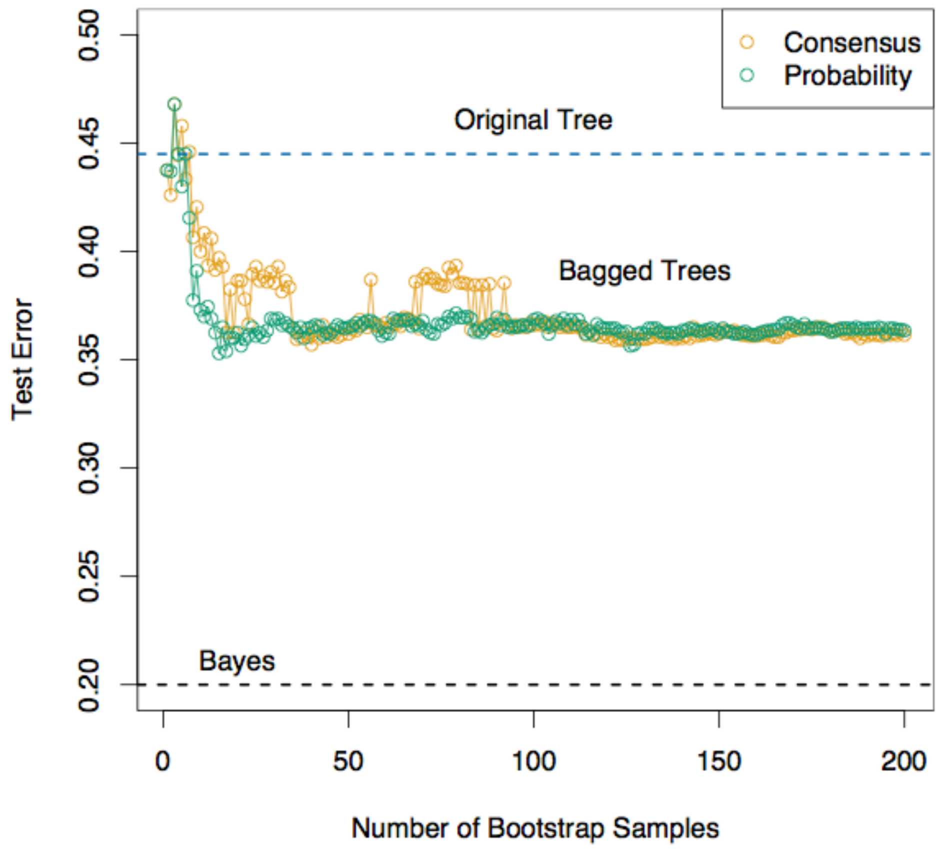


FIGURE 8.10. *Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The orange points correspond to the consensus vote, while the green points average the probabilities.*

web1.sph.emory.edu/users/tyu8/740/Lecture%2011%20forest.pptx

Boosting

- Powerful technique for combining multiple “base” classifiers to form a committee whose performance can be significantly better than any of the base classifiers

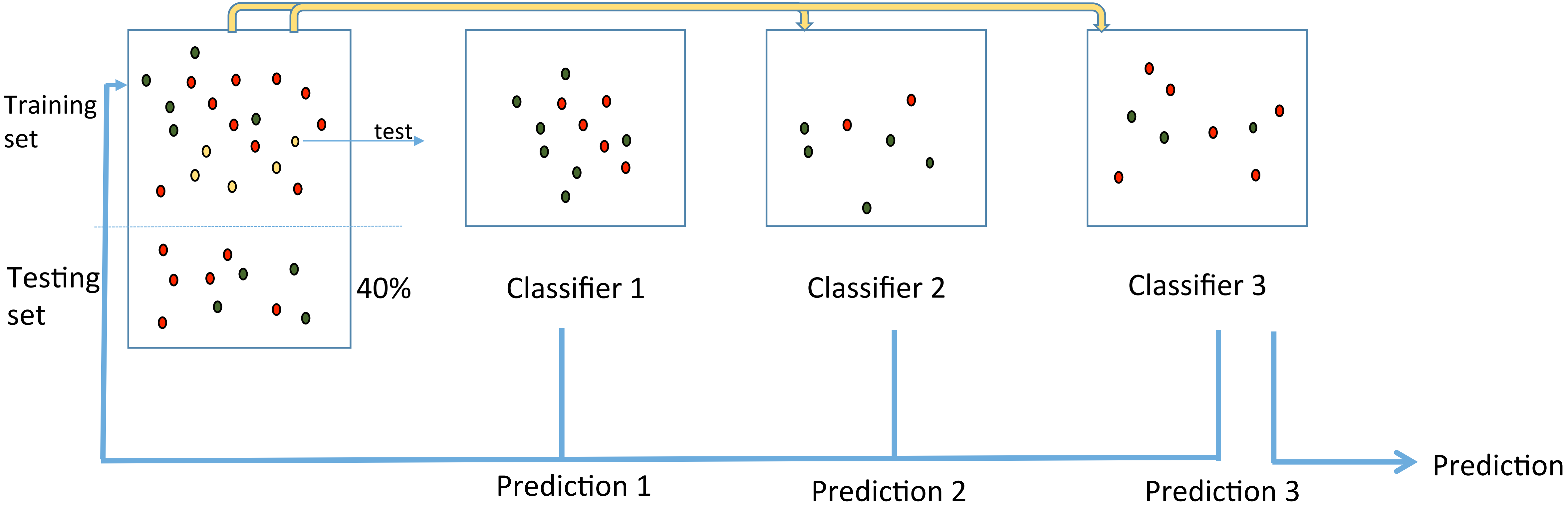
- AdaBoost (adaptive boosting) can give good results even if base classifier performance is only slightly better than random

- Hence base classifiers are known as weak learners

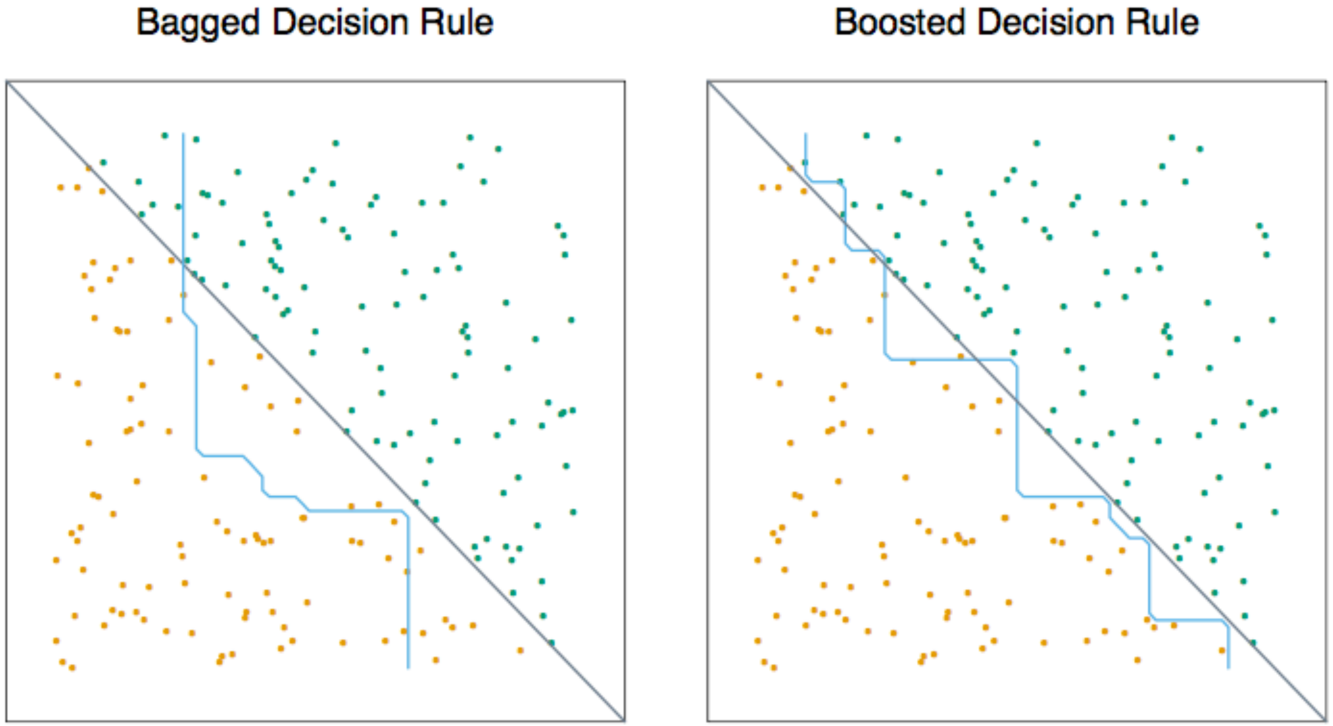
- Designed for classification, can be used for regression as well

Boosting

- Uniformly subsample with replacement
- Test on the training data and weight appropriately according to error
- Sequentially subsample with weights based on misclassification



Bagging vs. Boosting Comparison



Bagged Decision Rule

Boosted Decision Rule

FIGURE 8.12. *Data with two features and two classes, separated by a linear boundary. (Left panel:) Decision boundary estimated from bagging the decision rule from a single split, axis-oriented classifier. (Right panel:) Decision boundary from boosting the decision rule of the same classifier. The test error rates are* 0.166, *and* 0.065, *respectively. Boosting is described in Chapter 10.*

web1.sph.emory.edu/users/tyu8/740/Lecture%2011%20forest.pptx

Random Forest

- Bagging can be seen as a method to reduce variance of an estimated prediction function. It mostly helps high-variance classifiers.

- Comparatively, boosting build weak classifiers one-by-one, allowing the collection to evolve to the right direction.

- Random forest is a substantial modification to bagging

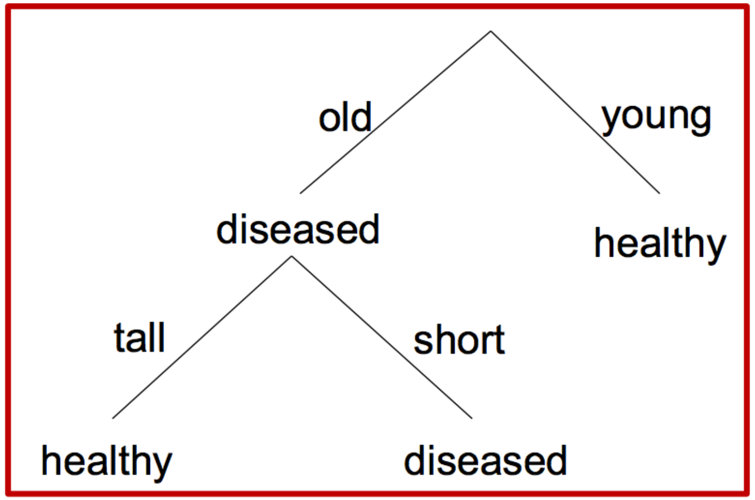
- Build a collection of **de-correlated** trees.

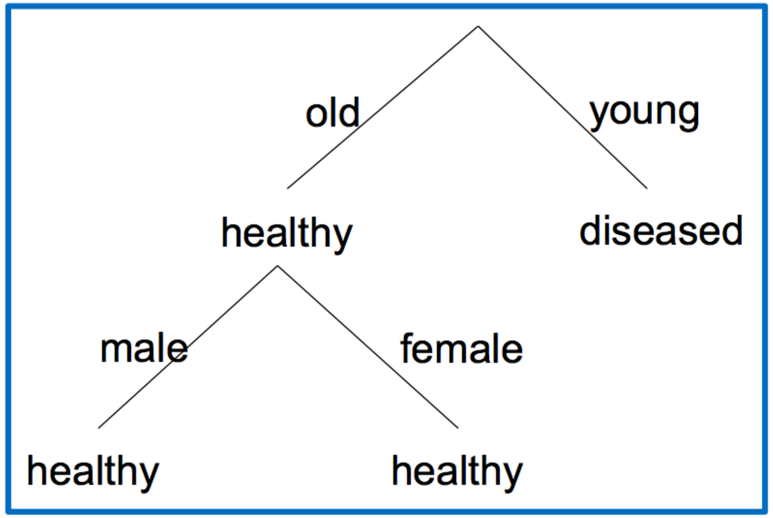
- Similar performance to boosting

- Simpler to train and tune compared to boosting

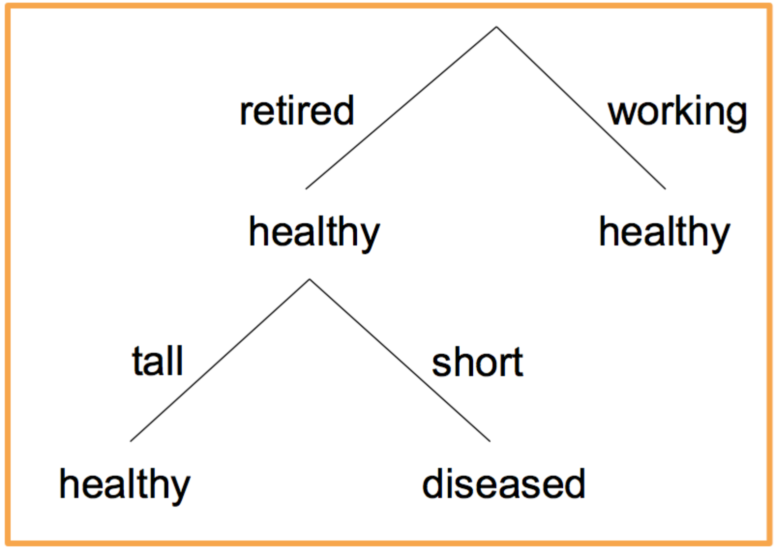
Random Forest (Intuition)

Tree 1





Tree 3



New sample:

old, retired, male, short

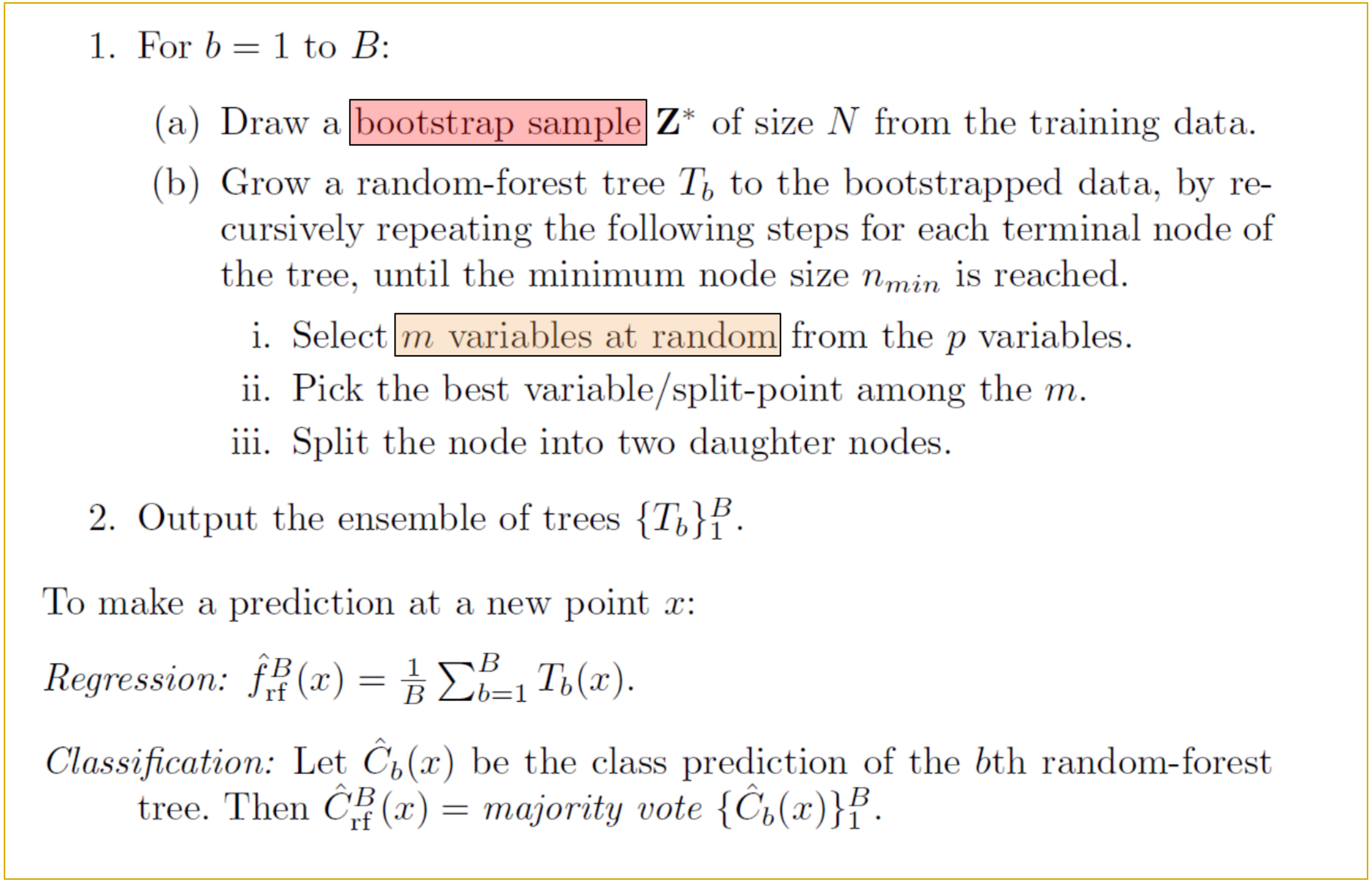
Tree predictions:

diseased, healthy, diseased

Majority rule:

diseased

Random Forest (Algorithm)



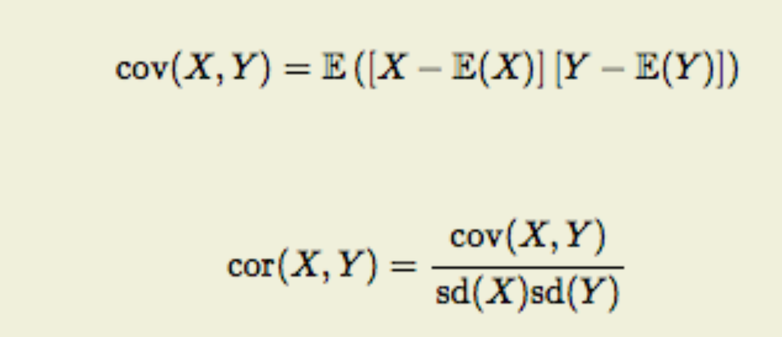
```
1. For $b = 1$ to $B$:
    (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size $N$ from the training data.
    (b) Grow a random-forest tree $T_b$ to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{min}$ is reached.
        i. Select $m$ variables at random from the $p$ variables.
        ii. Pick the best variable/split-point among the $m$.
        iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.
```

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Why Random Forest Works?



$$\mathrm{cov}(X,Y) = \mathbb{E}\left([X-\mathbb{E}(X)]\,[Y-\mathbb{E}(Y)]\right)$$

$$\mathrm{cor}(X,Y) = \frac{\mathrm{cov}(X,Y)}{\mathrm{sd}(X)\mathrm{sd}(Y)}$$

$$Var\left(\frac{1}{B}\sum_{i=1}^{B} T_i(c)\right) = \frac{1}{B^2}\sum_{i=1}^{B}\sum_{j=1}^{B} Cov(T_i(x),T_j(x))$$

i=j

$$= \frac{1}{B^2}\sum_{i=1}^{B}\left(\sum_{j\neq i}^{B} Cov(T_i(x),T_j(x)) + Var(T_i(x))\right)$$

$$= \frac{1}{B^2}\sum_{i=1}^{B}\left(\overbrace{(B-1)}\overbrace{\sigma^2\cdot\rho} + \sigma^2\right)$$

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

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

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

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

De-correlation gives better accuracy

Decreaes, if ρ decreases, i.e., if m decreases

Decreases, if number of trees B increases (irrespective of ρ)

Bagging vs. Boosting vs. Random Forest Comparison

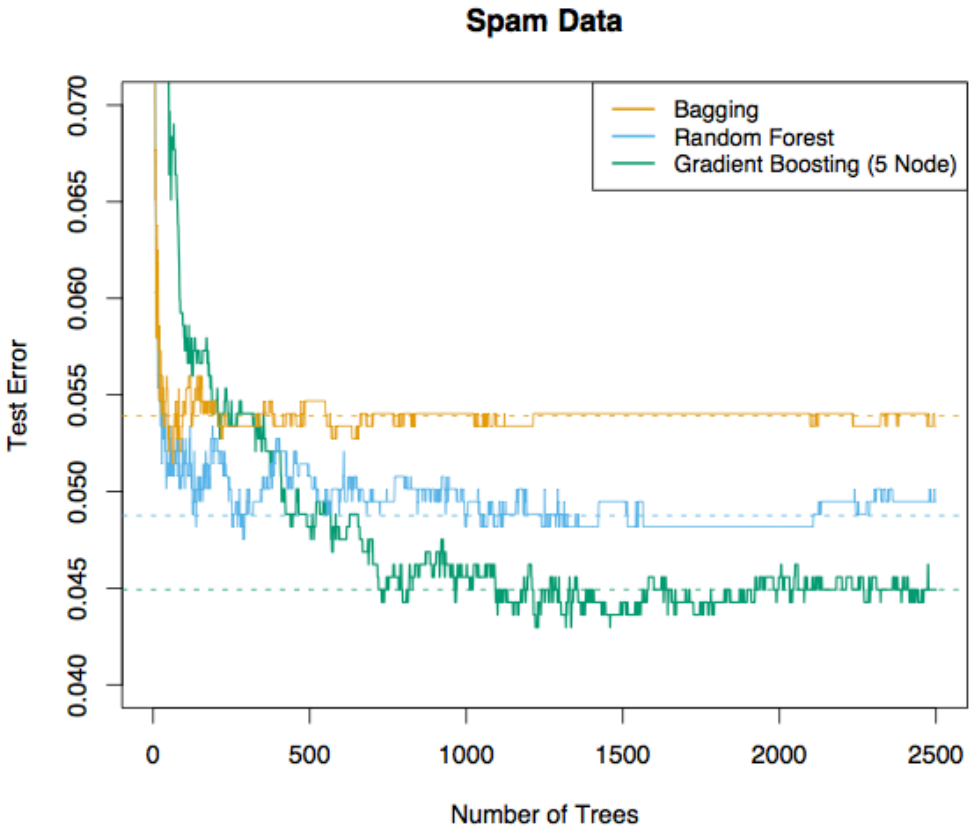
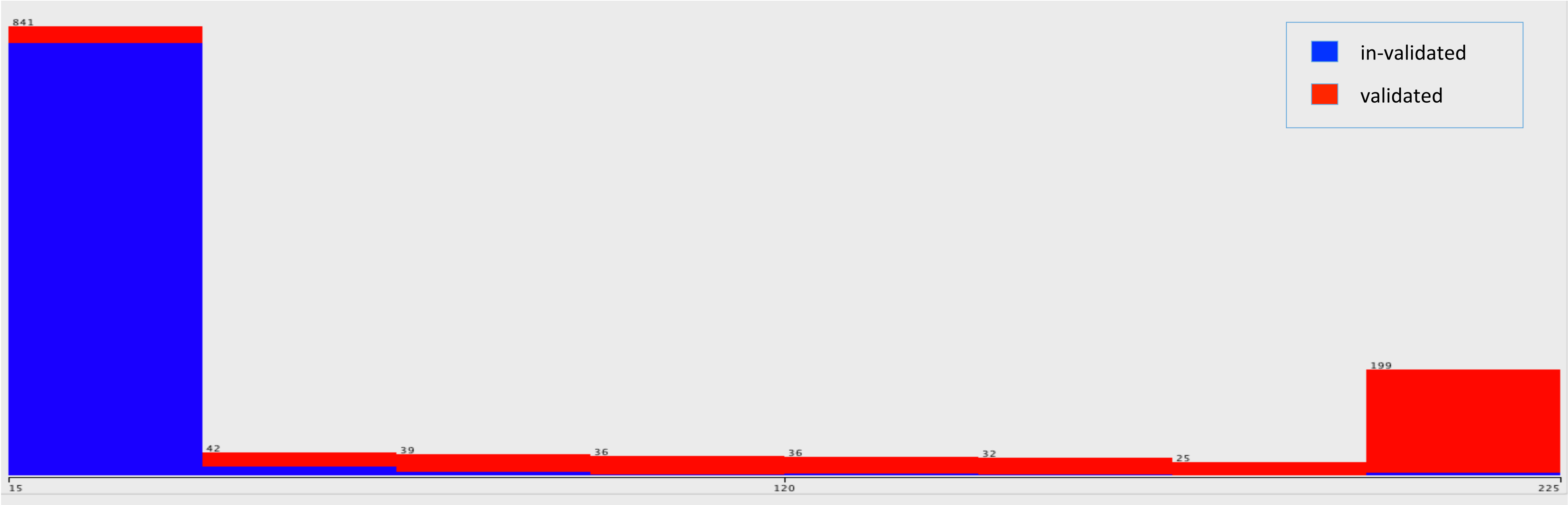


FIGURE 15.1. *Bagging, random forest, and gradient boosting, applied to the spam data. For boosting,* 5*-node trees were used, and the number of trees were chosen by* 10*-fold cross-validation (*2500 *trees). Each "step" in the figure corresponds to a change in a single misclassification (in a test set of* 1536*).*

Elements of Statistical Learning (2nd Ed.) c Hastie, Tibshirani & Friedman 2009 Chap 15

Cancer Genomics Applications I: Somatic Score



Distribution of somatic scores

- Few somatic mutations below 15 (max 255).
    - 47 is the validation necessity
    - Taken as a minimum threshold

Cancer Genomics Applications II: Determining key mutations

- How do we determine key mutations?

- Look into evolution for answers

- "Nothing in Biology Makes Sense Except in the Light of Evolution”

- Dobzhansky

Somatic Mutations

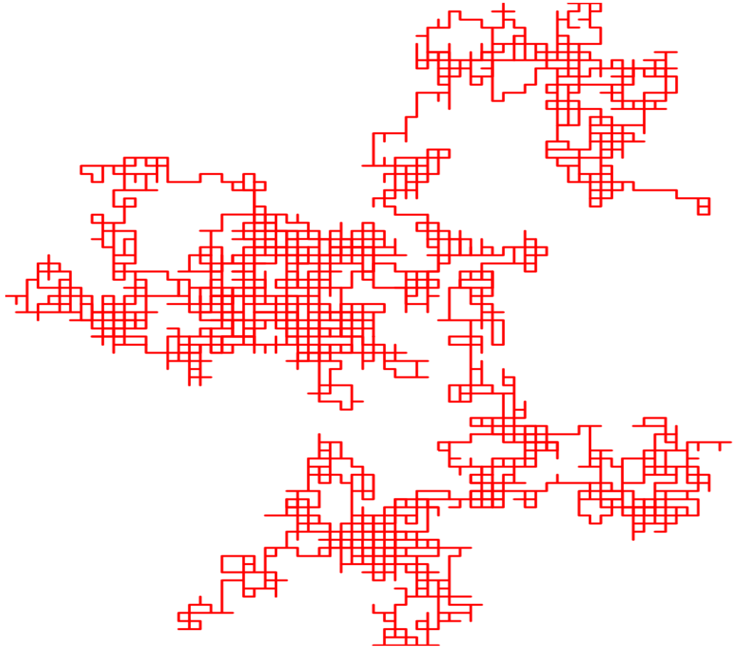
- All cancers arise because of somatically acquired changes
- Not all somatic variations will result in cancer – the ones that result are called “drivers”. “Passengers” are passive.
- Idea behind ‘driver’ and ‘passenger’ mutations is based on evolution
- Certain mutations are casually **selected** in the tumor micro-environment that would lead to increased survival/reproduction.
- Selection - What does this mean ??*?#$%$%*&???!!!!

Non-selection

- Understanding selection requires understanding non-selection.

- Non-selection – essentially stochastic process.
  - e.g. Random walk/Brownian motion etc.

- **Does not mean uncertainty**
  - if uncertainty means the probability of observing an event is not 1. e.g., Drunkard's walk.



- Expectation is zero at any instance of time.

- Genetic drift
  - Change in the allele frequency due to randomness.

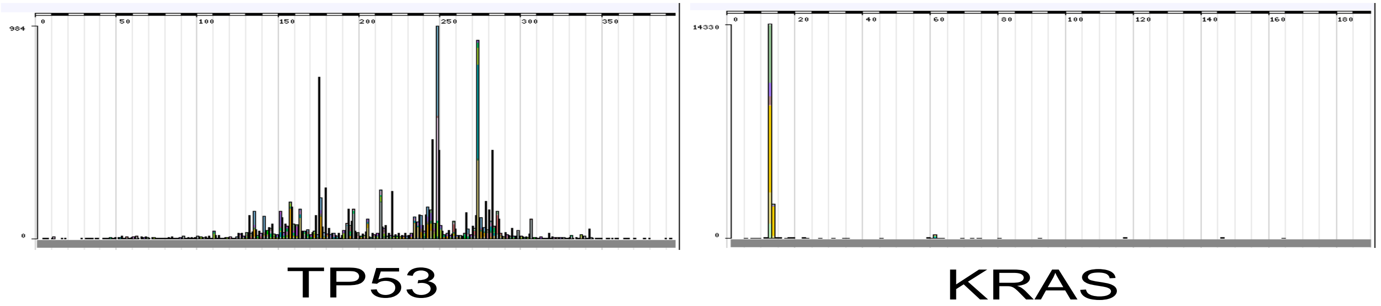
- Driver mutations are not outcomes of genetic drift, but due to **selection.**

Natural Selection and Cancer

- Natural selection is a ***non-random**** process by which phenotypes become fixated in a population as a function of differential reproduction/survival

- Cells in pre-malignant and malignant state (tumors) evolve by selecting genes that favors tumor genesis

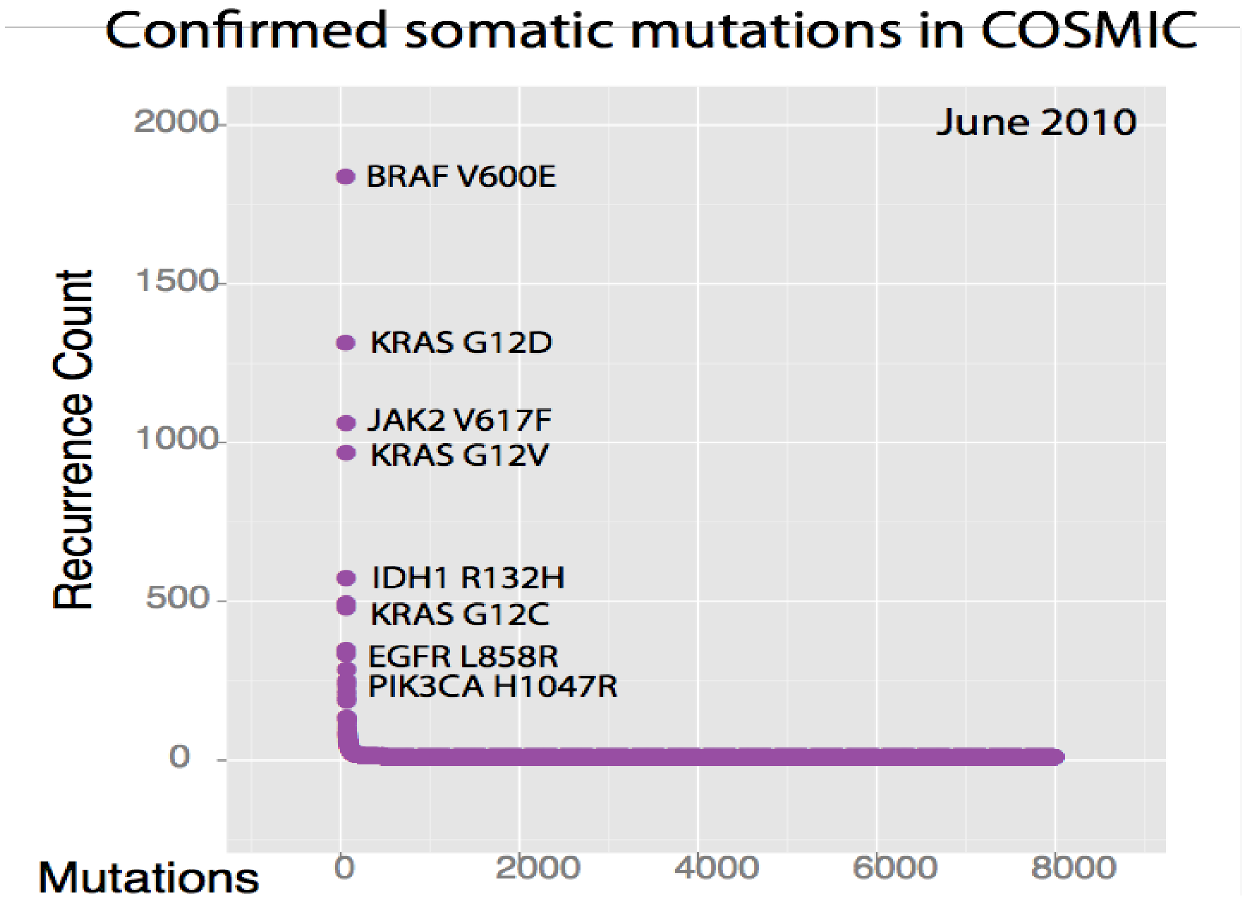
Driver Mutations



* As opposed to genetic drift, NS is deterministic and has direction.

Mutational Frequency

- Mutational frequency a defining criteria?



CHASM

- Methods independent of frequency of occurrence are required to determine driver genes.
- Cancer-specific High-throughput Annotation of Somatic Mutations (CHASM)* is a machine learning algorithm designed to identify driver mutations.
- Based on random forest classifier

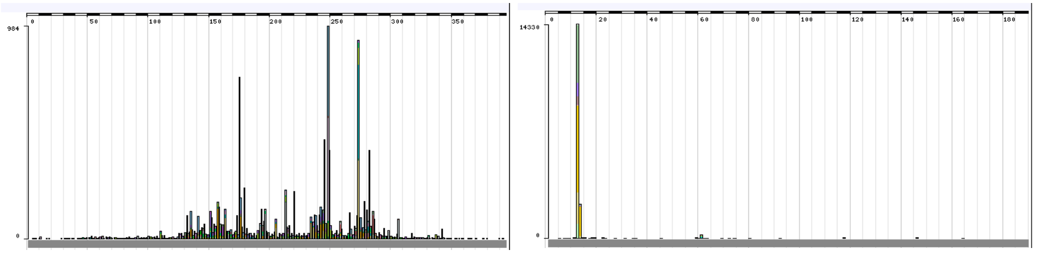
* Bozic I et al., PNAS, 2010

Natural Selection and Cancer

- Natural selection is a ***non-random**** process by which phenotypes become fixated in a population as a function of differential reproduction/survival

- Cells in pre-malignant and malignant state (tumors) evolve by selecting genes that favors tumor genesis

Driver Mutations

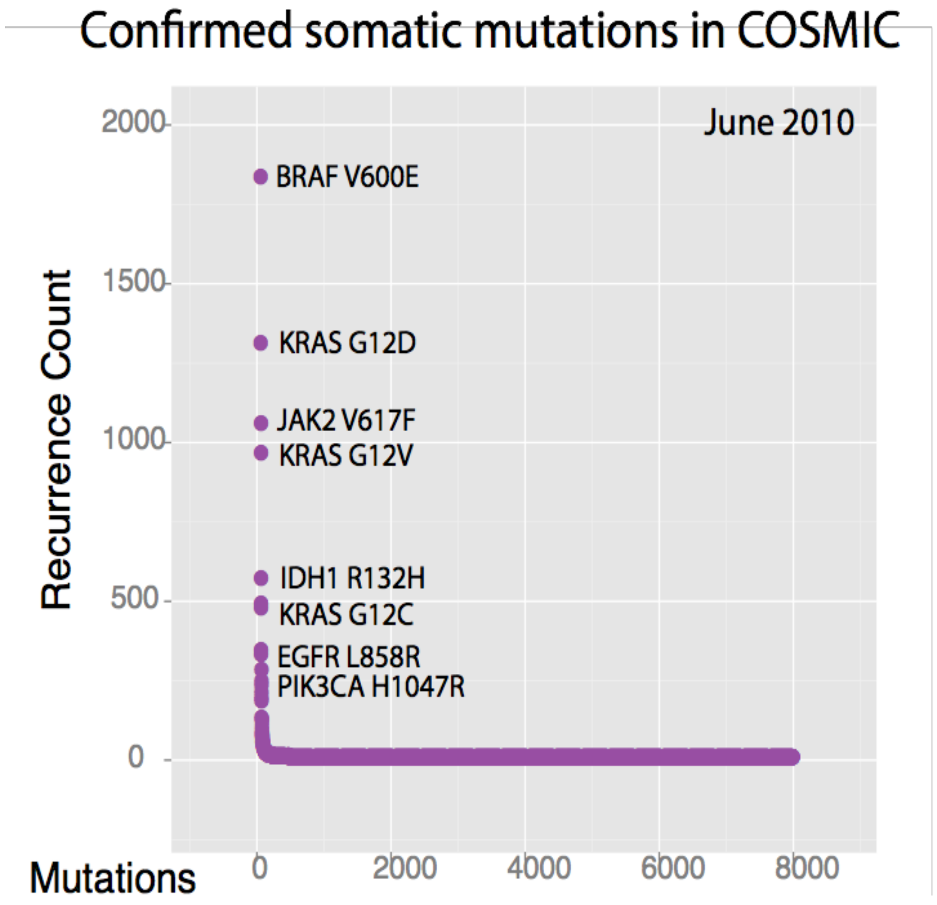


TP53 KRAS

* As opposed to genetic drift, NS is deterministic and has direction.

Mutational Frequency

- Mutational frequency a defining criteria?



CHASM

- Methods independent of frequency of occurrence are required to determine driver genes.

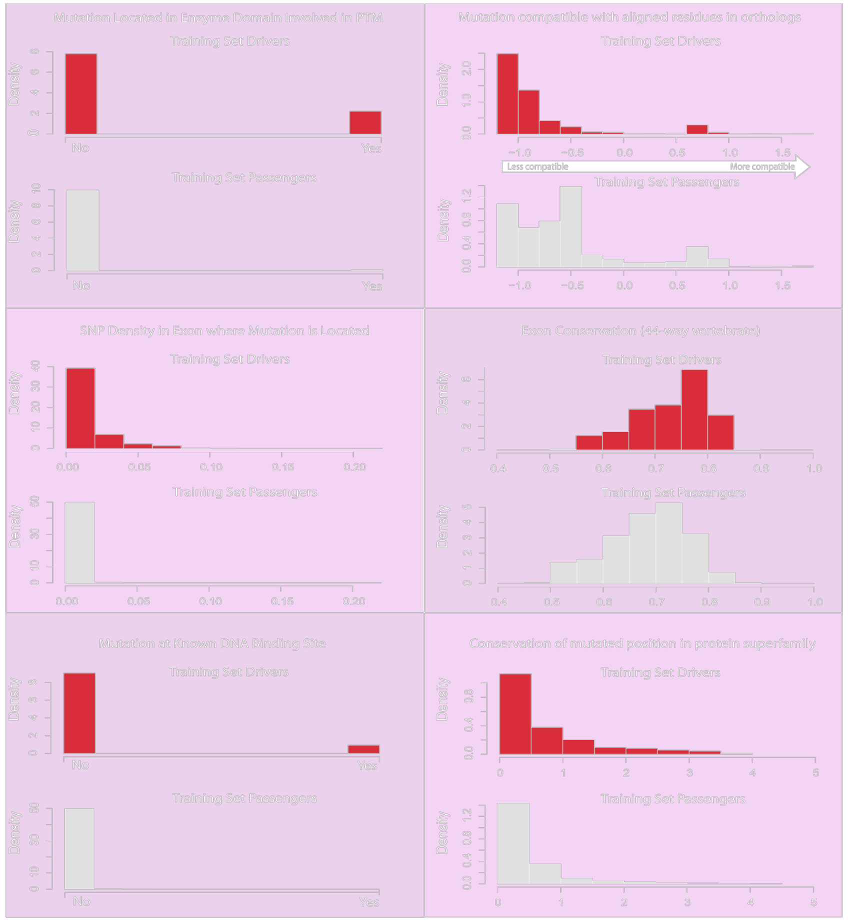
- Cancer-specific High-throughput Annotation of Somatic Mutations (CHASM)* is a machine learning algorithm designed to identify driver mutations.

- Based on random forest classifier

* Bozic I et al., PNAS, 2010

CHASM Features Set

- For each mutation, 56 default features like,



PTM enzyme

Ortholog compatible amino acid

SNP density

Exon conservation

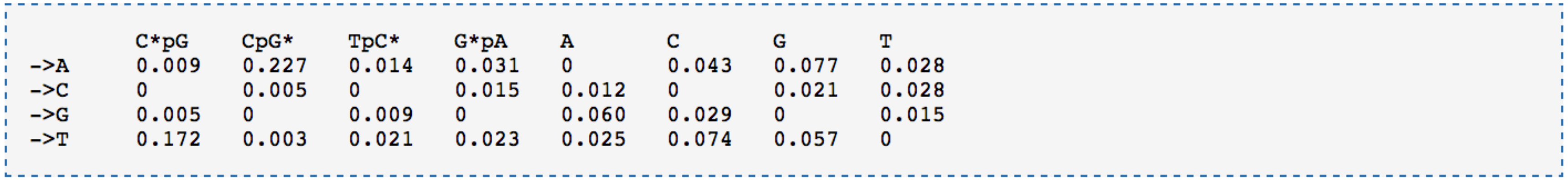
DNA binding

Superfamily conservation

CHASM Overview

- CHASM has a curated set of 2488 driver genes taken from COSMIC and other cancer datasets

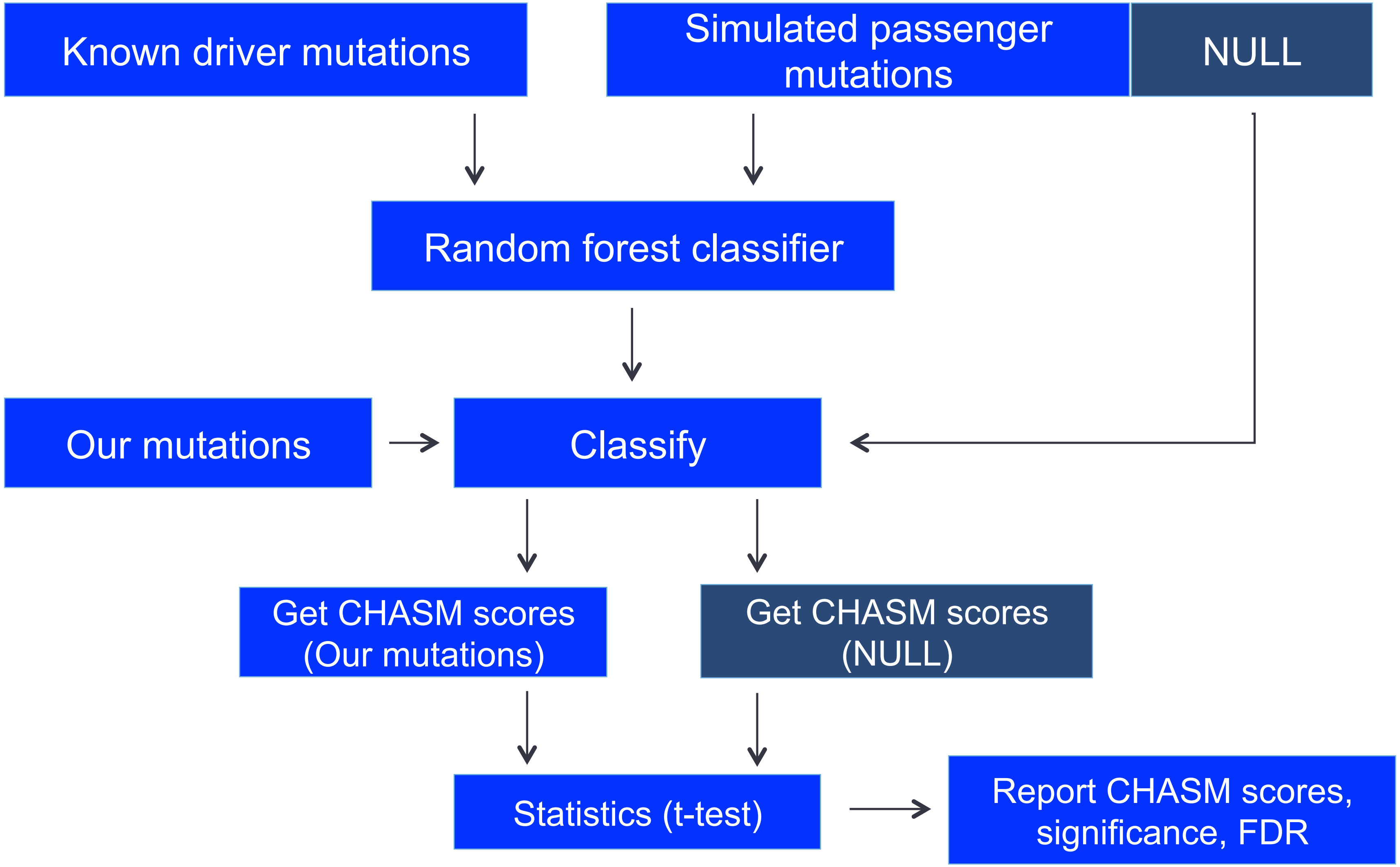
- CHASM simulates mutations based on given contexts and calls them passenger mutations. These are mutations that occur by random chance alone



```
        C*pG    CpG*    TpC*    G*pA    A       C       G       T
->A     0.009   0.227   0.014   0.031   0       0.043   0.077   0.028
->C     0       0.005   0       0.015   0.012   0       0.021   0.028
->G     0.005   0       0.009   0       0.060   0.029   0       0.015
->T     0.172   0.003   0.021   0.023   0.025   0.074   0.057   0
```

- A portion of these passenger mutations are isolated to form a null distribution

CHASM Overview



Known driver mutations

Simulated passenger mutations

NULL

Random forest classifier

Our mutations

Classify

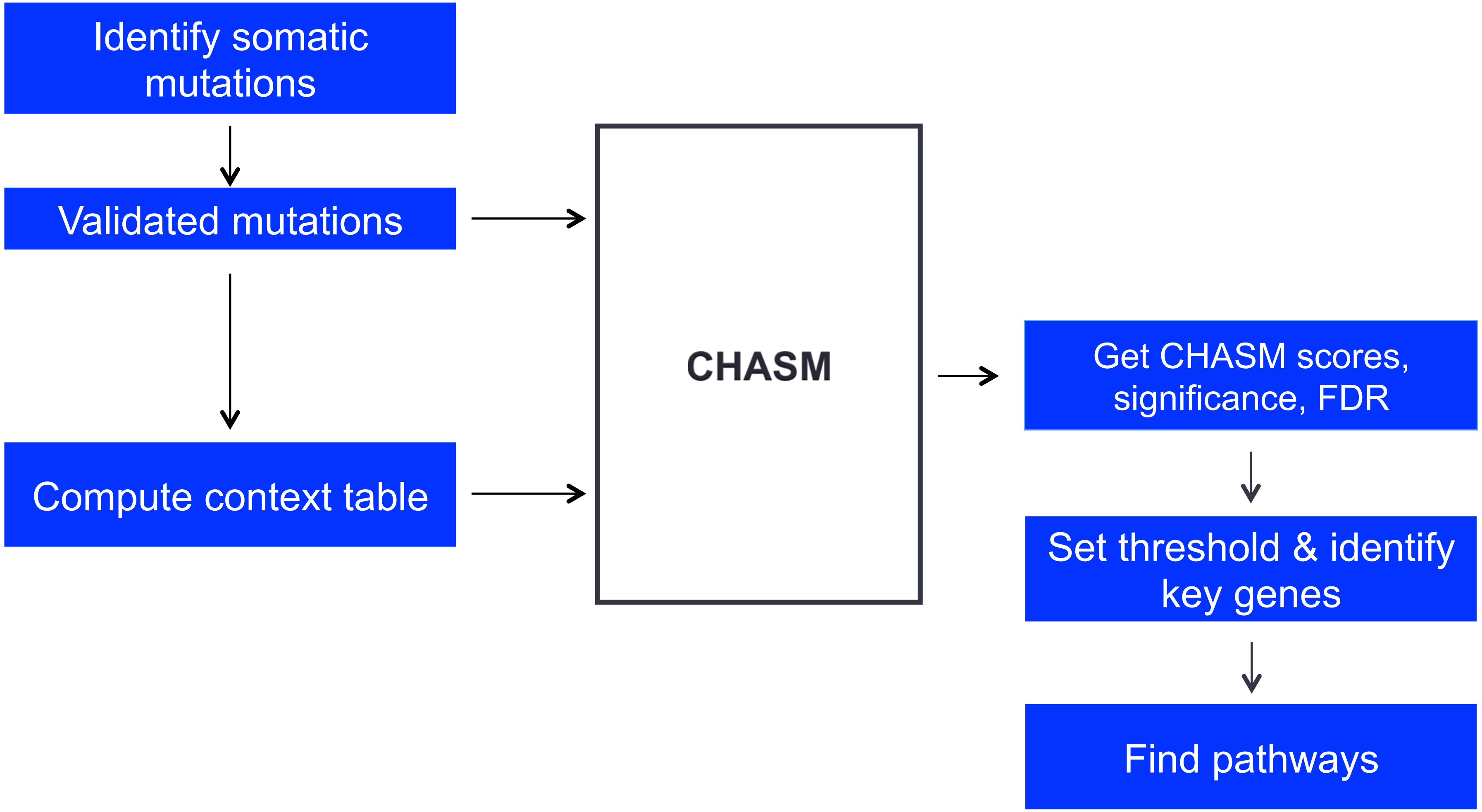
Get CHASM scores (Our mutations)

Get CHASM scores (NULL)

Statistics (t-test)

Report CHASM scores, significance, FDR

CHASM Pipeline



Identify somatic mutations

Validated mutations

Compute context table

CHASM

Get CHASM scores, significance, FDR

Set threshold & identify key genes

Find pathways

Summary

- Decision Trees
  - Overview
  - Splitting nodes
  - Limitations

- Bagging/Bootstrap Aggregating and Boosting
  - How bagging reduces variance?
  - Boosting

- Random Forests
  - Overview
  - Why RF works?

- Cancer Genomics Application