CS8803 / CS4365 Big Data Systems and Analytics

Lecture 7

# Data Analytics (II)
# Supervised Learning

Ling Liu
Professor
College of Computing
Georgia Institute of Technology

Adapted from Slides by Mahout team members, Mohamed Eltabakh@WPI

Georgia Institute of Technology | Georgia Tech College of Computing

## Outline

- Last Lecture
  - Data Analytics @ Apache Mahout
  - Unsupervised Learning
    - Clustering
    - Collaborative Filter (User-Based)
    - Association Rule Mining (Frequent Pattern Mining)
- This week's Lectures
  - Data Analytics (II) Supervised Learning →Classification
    - Decision Trees
    - Neural Networks
    - Ensemble

Georgia Institute of Technology | Georgia Tech College of Computing

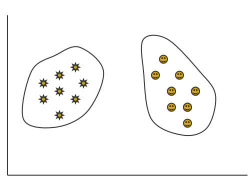
## Data Analytics: Learning

- Learning systems are complex and may have many parameters.
- It is impractical and often impossible to encode all the knowledge a system needs.
- Different learning goals on the same type of data or the same learning goal on different types of data may require very different parameters.
- Instead of trying to hard code all the knowledge, it makes sense to learn it.



## Learning from Observations

- Unsupervised Learning – No classes are given. The idea is to find patterns in the data. This generally involves clustering.



- Reinforcement Learning – learn from feedback after a decision is made.

## Learning from Observations

- Supervised Learning – learn a function from a set of training examples which are preclassified feature vectors.

| feature vector | class |
|---|---|
| (square, red) | I |
| (square, blue) | I |
| (circle, red) | II |
| (circle blue) | II |
| (triangle, red) | I |
| (triange, green) | I |
| (ellipse, blue) | II |
| (ellipse, red) | II |

Given a previously unseen feature vector, what is the rule that tells us if it is in class I or class II?

| | |
|---|---|
| (circle, green) | ? |
| (triangle, blue) | ? |



## Classification

- **Classification** is the process of supervised learning
  - Learning is **supervised** as the classes to be learned are predetermined.
  - Learning is accomplished by using a training set of pre-classified data.
- **The Classification Problem**:
  - Given a set of example records, each consists of
    - A set of attributes + A class label
  - Build an accurate model for each class label based on the set of attributes
  - Use the model to classify future data for which the class labels are unknown

## Classification Techniques

- Two Phase Approach:
  1. **Training Phase:** Create specific model by evaluating training data (or using domain experts' knowledge).
  2. **Prediction Phase:** Apply model developed to new data.
- Classes must be predefined

| feature vector | class |
| --- | --- |
| (square, red) | I |
| (square, blue) | I |
| (circle, red) | II |
| (circle blue) | II |
| (triangle, red) | I |
| (triange, green) | I |
| (ellipse, blue) | II |
| (ellipse, red) | II |

Given a previously unseen feature vector, what is the rule that tells us if it is in class I or class II?

(circle, green) ?
(triangle, blue) ?

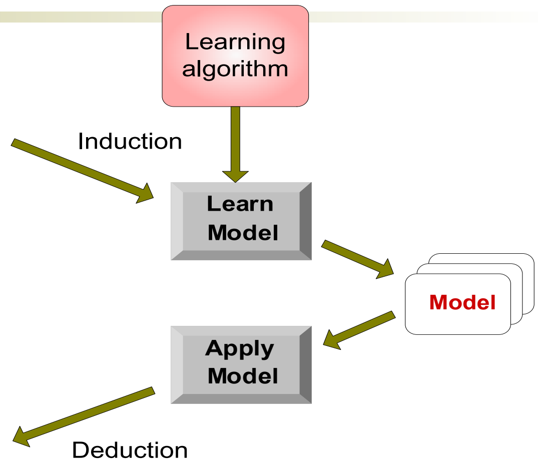
## Classification: Definition

- Given a collection of records (*training set* )
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for the 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.

# Illustrating Classification Task

| 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

# Classification Methods

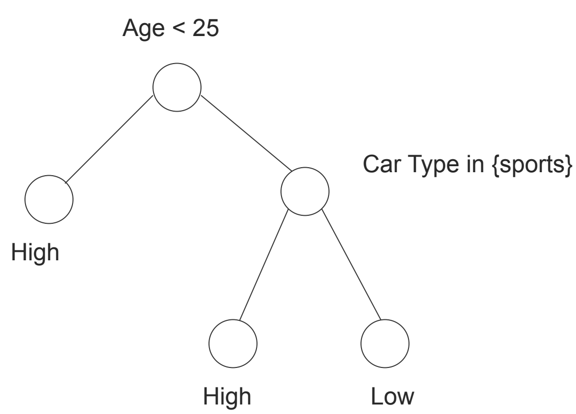
- **Decision tree induction**
  - ID3, C4.5
- **Neural Networks**
  - **CNN**
- **Bayesian Classification**
  - Bayesian Belief Networks
- **Support Vector Machine (SVM)**
  - Regression

## A Training set

| Age | Car Type | Risk |
|---|---|---|
| 23 | Family | High |
| 17 | Sports | High |
| 43 | Sports | High |
| 68 | Family | Low |
| 32 | Truck | Low |
| 20 | Family | High |



## A Decision Tree

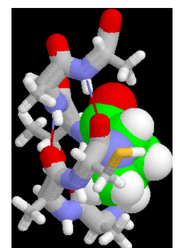


# Examples of Classification Task

- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent

[NEWSFACTOR NETWORK]

- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil



- Categorizing news stories as finance, weather, entertainment, sports, etc

Georgia Institute of Technology — Georgia Tech College of Computing

# Supervised Learning: Classification

- **Inductive Learning**
  - Decision trees
  - Neural networks
  - Ensembles
  - Bayesian decision making
  - Regression and SVM

14

Georgia Institute of Technology — Georgia Tech College of Computing

# An Example from Quinlan's ID3

- Data: Computer purchase data
- Task: Predict whether a person will buy computer or not
- Performance measure: accuracy

Classification goal

Attributes used for classification

| age | income | student | credit_rating | buys_computer |
|---|---|---|---|---|
| <=30 | high | no | fair | no |
| <=30 | high | no | excellent | no |
| 31…40 | high | no | fair | yes |
| >40 | medium | no | fair | yes |
| >40 | low | yes | fair | yes |
| >40 | low | yes | excellent | no |
| 31…40 | low | yes | excellent | yes |
| <=30 | medium | no | fair | no |
| <=30 | low | yes | fair | yes |
| >40 | medium | yes | fair | yes |
| <=30 | medium | yes | excellent | yes |
| 31…40 | medium | no | excellent | yes |
| 31…40 | high | yes | fair | yes |
| >40 | medium | no | excellent | no |

Yes 9/14

No 5/14



# Simple Statistic based Learning: An Example

- Data: Computer purchase data
- Task: Predict whether a person will buy a computer or not
- Performance measure: accuracy

Simple Statistic-based learning:

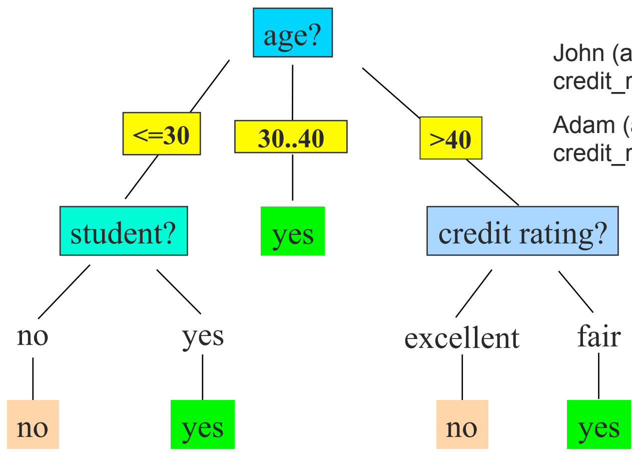
classify all future computer purchase predictions (test data) to the majority class (i.e., Yes):

Accuracy = 9/14 ~ 64%

- Can we do better than 64% with model based learning?

## A Decision Tree Classification for *"buys_computer"*

Training Data: **Age, Income, Student, CreditRating**, ClassLabel




John (age=27, student=no, credit_rating=fair)

Adam (age=35, student=no, credit_rating=fair)

## Decision Trees

- Theory is well-understood.
- Often used in pattern recognition problems.
- Has the nice property:
  - you can easily understand the decision rules resulting from DT induction based learning.

## Decision Overview

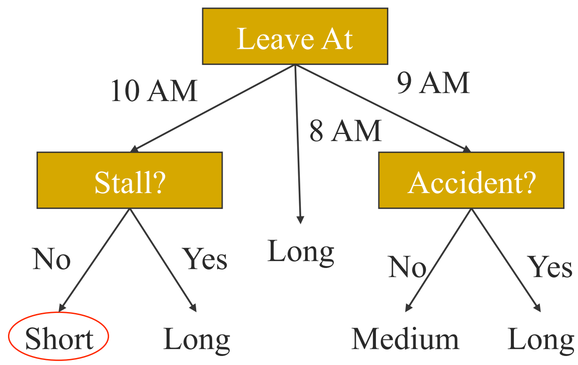
- What is a Decision Tree
- Sample Decision Trees
- How to Construct a Decision Tree
- Problems with Decision Trees
- Research in Decision Trees
- Summary



## What is a Decision Tree?

- An *inductive learning task*
  - Use particular facts to make more generalized conclusions
  - Binary classification v.s. Multi-Label classification
- A predictive model based on a branching series of Boolean tests
  - These smaller Boolean tests are less complex than a one-stage classifier

# Predicting Commute Time




Unseen data testing:

If we leave at 10 AM and there are no cars stalled on the road, what will our commute time be?

# Inductive Learning

- In this decision tree, we made a series of Boolean decisions and followed the corresponding branch
  - Did we leave at 10 AM?
  - Did a car stall on the road?
  - Is there an accident on the road?

- By answering each of these yes/no questions, we then came to a conclusion on how long our commute might take

## Decision Trees as Rules

- We did not have to represent this tree graphically
- We could have represented as a set of rules. However, this may be much harder to read…

## Decision Tree as a Rule Set

```
if hour == 8am
    commute time = long
else if hour == 9am
    if accident == yes
        commute time = long
    else
        commute time = medium
else if hour == 10am
    if stall == yes
        commute time = long
    else
        commute time = short
```

Notice that

- Not all attributes have to be used in each path of the decision.
- Some attributes may not even appear in the tree.

# How to Create a Decision Tree

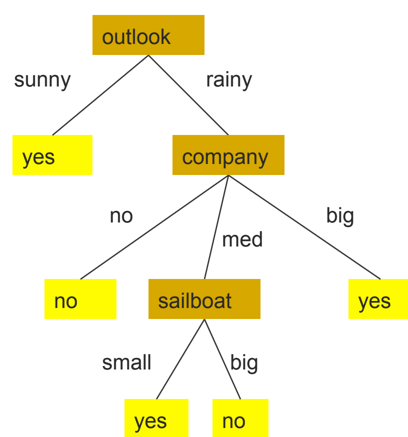
- We first make a list of attributes that we can measure
  - These attributes (for now) must be discrete
- We then choose a *target attribute* that we want to predict
- Then create an *experience table* that lists what we have seen in the past (training set)

# An Example Training Data Set and Decision Tree

Target attribute

| # | Attribute | | | Class |
|---|---|---|---|---|
| | Outlook | Company | Sailboat | Sail? |
| 1 | sunny | big | small | yes |
| 2 | sunny | med | small | yes |
| 3 | sunny | med | big | yes |
| 4 | sunny | no | small | yes |
| 5 | sunny | big | big | yes |
| 6 | rainy | no | small | no |
| 7 | rainy | med | small | yes |
| 8 | rainy | big | big | yes |
| 9 | rainy | no | big | no |
| 10 | rainy | med | big | no |

Experience Table

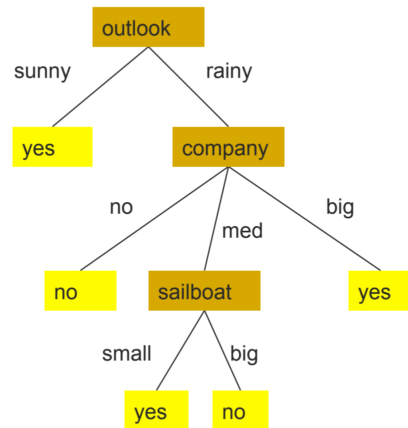


# Classification

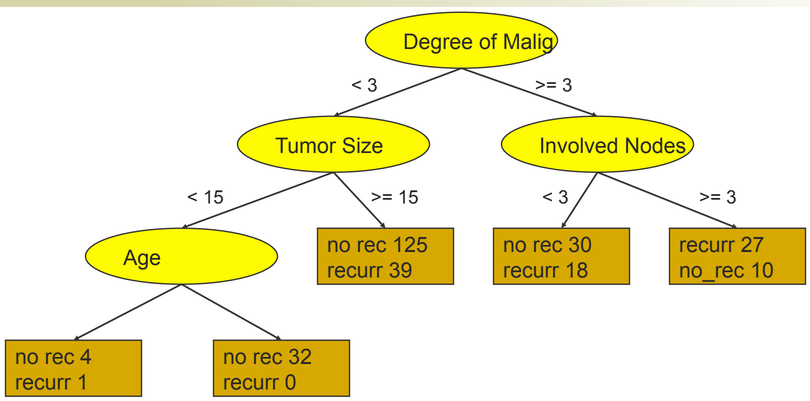
| # | Attribute | | | Class |
|---|---|---|---|---|
| | Outlook | Company | Sailboat | Sail? |
| 1 | sunny | no | big | ? |
| 2 | rainy | med | big | ? |

How about the class label for the following

| | | | | |
|---|---|---|---|---|
| 3 | *Sunny* | *small* | *big* | *?* |
| 4 | *Rainy* | *small* | *big* | *?* |



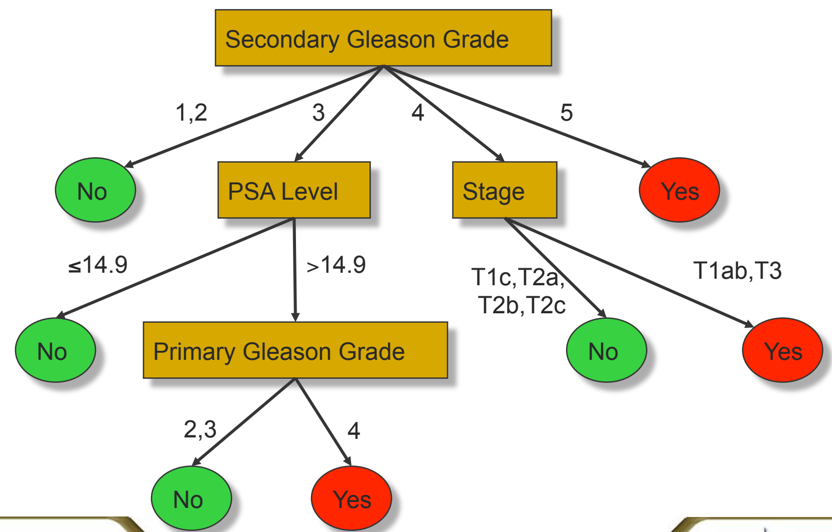

# Breast Cancer Recurrence




Tree induced by Assistant Professional

Interesting: Accuracy of this tree compared to medical specialists

# Prostate cancer recurrence



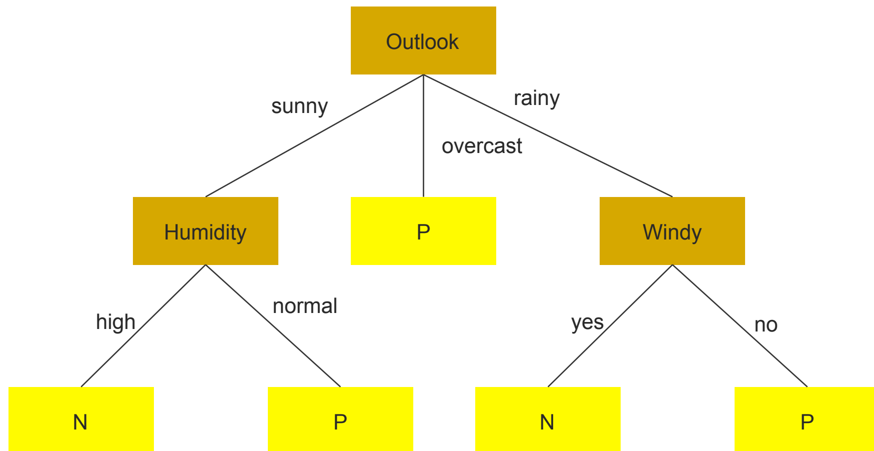

Georgia Institute of Technology
Georgia Tech College of Computing

# Another Example

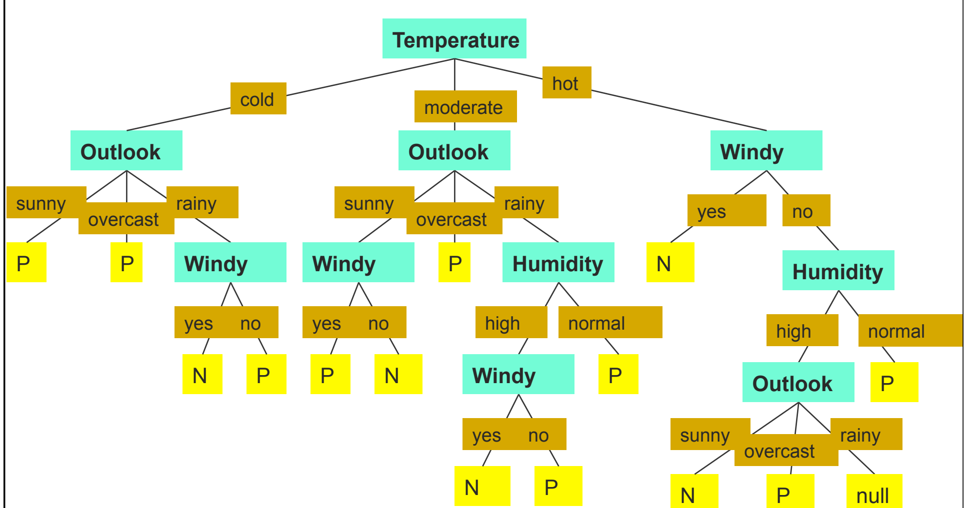
| # | Attribute | | | | Class |
|---|---|---|---|---|---|
| | Outlook | Temperature | Humidity | Windy | Play |
| 1 | sunny | hot | high | no | N |
| 2 | sunny | hot | high | yes | N |
| 3 | overcast | hot | high | no | P |
| 4 | rainy | moderate | high | no | P |
| 5 | rainy | cold | normal | no | P |
| 6 | rainy | cold | normal | yes | N |
| 7 | overcast | cold | normal | yes | P |
| 8 | sunny | moderate | high | no | N |
| 9 | sunny | cold | normal | no | P |
| 10 | rainy | moderate | normal | no | P |
| 11 | sunny | moderate | normal | yes | P |
| 12 | overcast | moderate | high | yes | P |
| 13 | overcast | hot | normal | no | P |
| 14 | rainy | moderate | high | yes | N |

Georgia Institute of Technology
Georgia Tech College of Computing

# Simple Tree




# Complicated Tree



## Attribute Selection Criteria

- Main principle
  - Select attribute which partitions the learning set into subsets as "pure" as possible
- Various measures of purity
  - Information-theoretic
  - Gini index
  - $X^2$
  - ReliefF
  - ...
- Various improvements
  - probability estimates
  - normalization
  - binarization
  - subsetting



## Induction of Decision Trees

- Data Set (Learning Set)
  - Each example = Attributes + Class
- Induced description = Decision tree

- TDIDT: Top Down Induction of Decision Trees
- Recursive Partitioning

## TDIDT Algorithm

- Also known as ID3 (Quinlan)
- To construct decision tree T from learning set S:
  - **If** all examples in S belong to some class C **Then** make leaf labeled C
  - **Otherwise**
    - select the "most informative" attribute A
    - partition S according to A's values
    - recursively construct subtrees T1, T2, ..., for the subsets of S



## Decision Tree Induction Algorithms

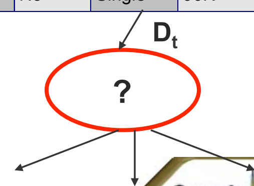
- Hunt's Algorithm (one of the earliest)
- CART
- ID3 (Quinlan 79)
- CART (Brieman et al. 84)
- Assistant (Cestnik et al. 87)
- C4.5 (Quinlan 93)
- See5 (Quinlan 97)
- SLIQ/SPRINT

Georgia Institute of Technology Georgia Tech College of Computing

# General Structure of Hunt's Algorithm

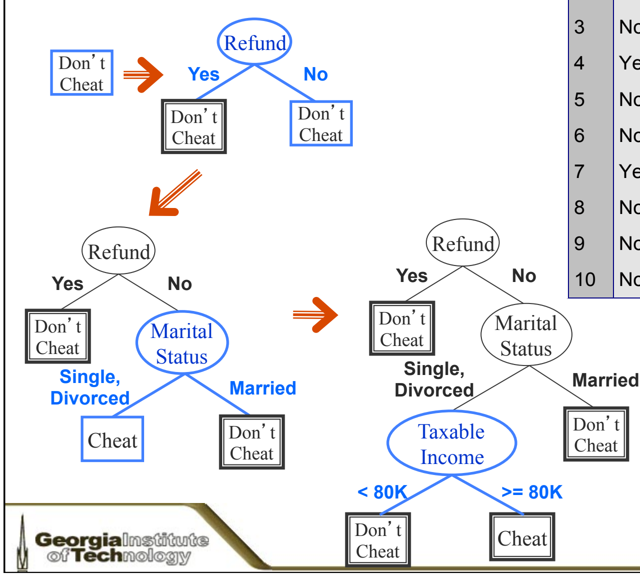
- Let $D_t$ be the set of training records that reach a node t
- General Procedure:
  - If $D_t$ contains records that belong the same class $y_t$, then t is a leaf node labeled as $y_t$
  - If $D_t$ is an empty set, then t is a leaf node labeled by the default class, $y_d$
  - If $D_t$ contains records 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.

| *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 |




# Hunt's Algorithm

| *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.
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting



## Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split (best attribute to use)?
  - Determine when to stop splitting

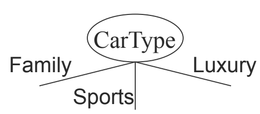
# How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

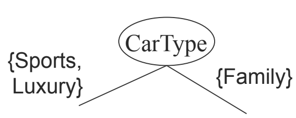
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

# Splitting Based on Nominal Attributes

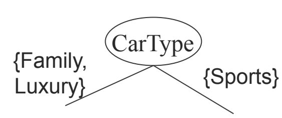
- Multi-way split: Use as many partitions as distinct values.




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

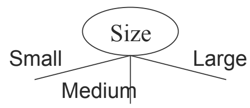



OR

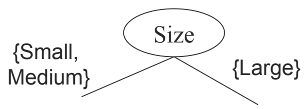


# Splitting Based on Ordinal Attributes

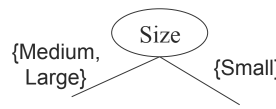
- Multi-way split: Use as many partitions as distinct values.



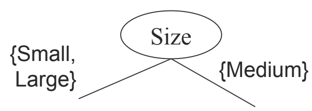

- Binary split: Divides values into two subsets.
  Need to find optimal partitioning.




OR



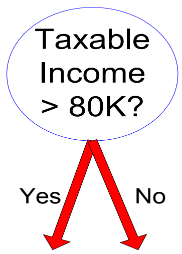

- What about this split?



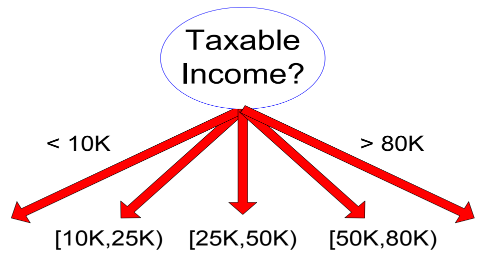

# Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more compute intensive

# Splitting Based on Continuous Attributes




(i) Binary split




(ii) Multi-way split

# Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

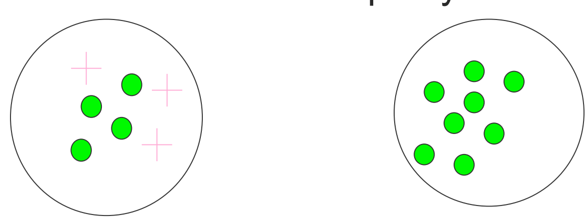
## Criterion for attribute selection

- Which is the best attribute?
- The one which will result in the smallest tree
  - Heuristic: choose the attribute that produces the "purest" nodes
- Need a good measure of **purity**!
  - Maximal when?
  - Minimal when?



## How to determine the Best Split

- Greedy approach:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity:



**Non-homogeneous,**

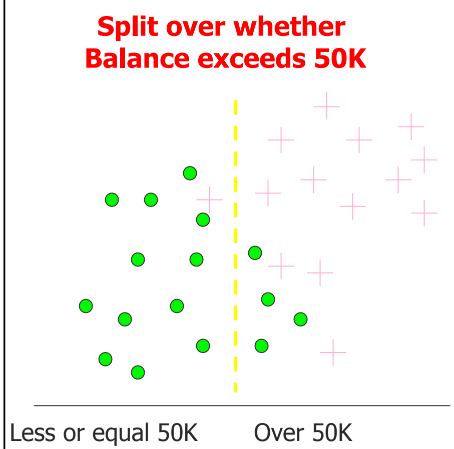
**High degree of impurity**

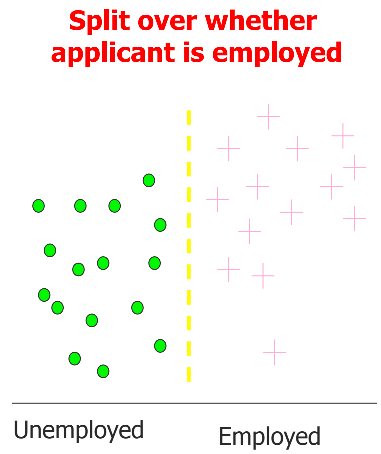
**Homogeneous,**

**Low degree of impurity**

# Information Gain

Which test is more informative?



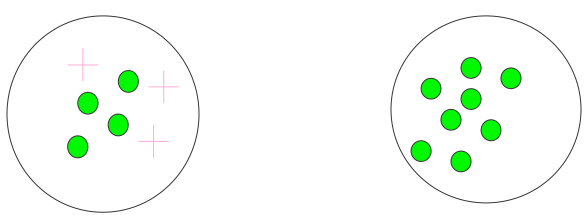






# Information Gain

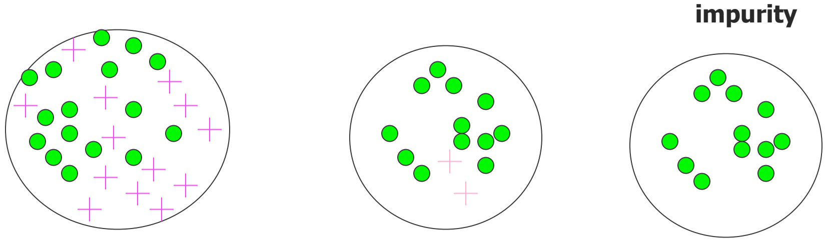
**Impurity/Entropy** (informal)

- Measures the level of **impurity** in a group of examples



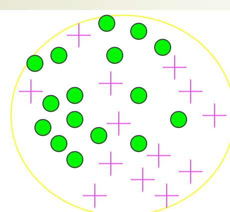
# Impurity

**Very impure group** **Less impure** **Minimum impurity**





# Entropy: a common way to measure impurity



- Entropy = $\sum_i - p_i \log_2 p_i$

  $p_i$ is the probability of class i
  Compute it as the proportion of class i in the set.

- Entropy comes from information theory. The higher the entropy the more the information content.

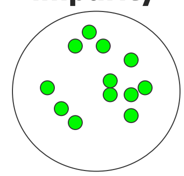
  What does that mean for learning from examples?

# Binary-Class Cases:

- What is the entropy of a group in which all examples belong to the same class?
  - entropy = $-1 \log_2 1 = 0$

not a good training set for learning

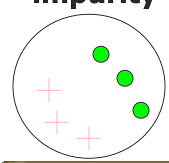
**Minimum impurity**



- What is the entropy of a group with 50% in either class?
  - entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

good training set for learning

**Maximum impurity**





# Information Gain

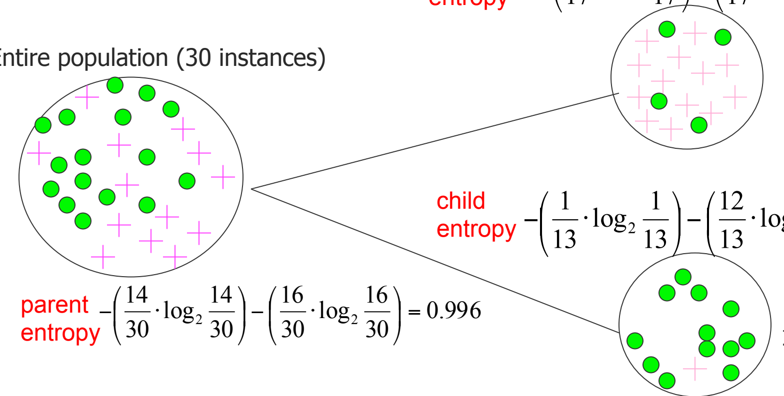
- We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.
- Information gain tells us how important a given attribute of the feature vectors is.
- We will use it to decide the ordering of attributes in the nodes of a decision tree.

## Calculating Information Gain

**Information Gain** = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$

Entire population (30 instances)



17 instances

child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$

13 instances

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

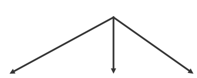
**Information Gain= 0.996 - 0.615 = 0.38**



## Entropy-Based Automatic Decision Tree Construction

| |
|---|
| Training Set S<br>x1=(f11,f12,…f1m)<br>x2=(f21,f22, f2m)<br>.<br>.<br>xn=(fn1,f22, f2m) |

Node 1
What feature
should be used?

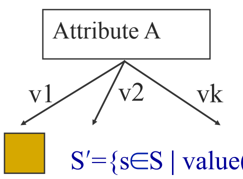


What values?

Quinlan suggested information gain in his ID3 system and later the gain ratio, both based on entropy.

## Using Information Gain to Construct a Decision Tree

Full Training Set S




Choose the attribute A with highest information gain for the full training set at the root of the tree.

Construct child nodes for each value of A. Each has an associated subset of vectors in which A has a particular value.

Set S ′

S′={s∈S | value(A)=v1}

repeat recursively till when?

Information gain has the disadvantage that it prefers attributes with large number of values that split the data into small, pure subsets.
Quinlan's gain ratio did some normalization to improve this.

57

## Information-Theoretic Approach

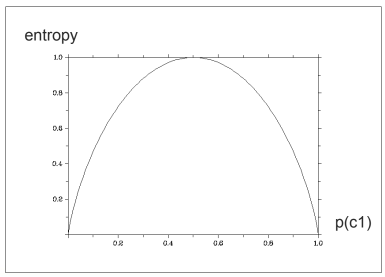
- To classify an object, a certain information is needed
  - I, information
- After we have learned the value of attribute A, we only need some remaining amount of information to classify the object
  - Ires, residual information
- Gain
  - Gain(A) = I – Ires(A)
- The most 'informative' attribute is the one that minimizes Ires, *i.e.*, maximizes Gain

# Entropy

- The average amount of information *I* needed to classify an object is given by the entropy measure

$$I = -\sum_{c} p(c) \log_2 p(c)$$

- For a two-class problem:




# Residual Information

- After applying attribute *A*, *S* is partitioned into subsets according to values *v* of *A*
- *Ires* is equal to weighted sum of the amounts of information for the subsets

$$I_{res} = -\sum_{v} p(v) \sum_{c} p(c|v) \log_2 p(c|v)$$

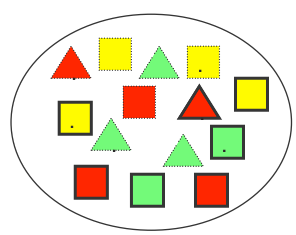
# Triangles and Squares

| # | Color | Outline | Dot | Shape |
|---|---|---|---|---|
| | Attribute | | | |
| 1 | green | dashed | no | triange |
| 2 | green | dashed | yes | triange |
| 3 | yellow | dashed | no | square |
| 4 | red | dashed | no | square |
| 5 | red | solid | no | square |
| 6 | red | solid | yes | triange |
| 7 | green | solid | no | square |
| 8 | green | dashed | no | triange |
| 9 | yellow | solid | yes | square |
| 10 | red | solid | no | square |
| 11 | green | solid | yes | square |
| 12 | yellow | dashed | yes | square |
| 13 | yellow | solid | no | square |
| 14 | red | dashed | yes | triange |

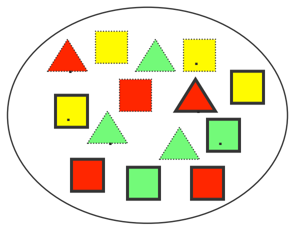
# Triangles and Squares

| # | Color | Outline | Dot | Shape |
|---|---|---|---|---|
| | Attribute | | | |
| 1 | green | dashed | no | triange |
| 2 | green | dashed | yes | triange |
| 3 | yellow | dashed | no | square |
| 4 | red | dashed | no | square |
| 5 | red | solid | no | square |
| 6 | red | solid | yes | triange |
| 7 | green | solid | no | square |
| 8 | green | dashed | no | triange |
| 9 | yellow | solid | yes | square |
| 10 | red | solid | no | square |
| 11 | green | solid | yes | square |
| 12 | yellow | dashed | yes | square |
| 13 | yellow | solid | no | square |
| 14 | red | dashed | yes | triange |

Data Set:
A set of classified objects



# Entropy



- 5 triangles
- 9 squares
- class probabilities

$$p(\square) = \frac{9}{14}$$

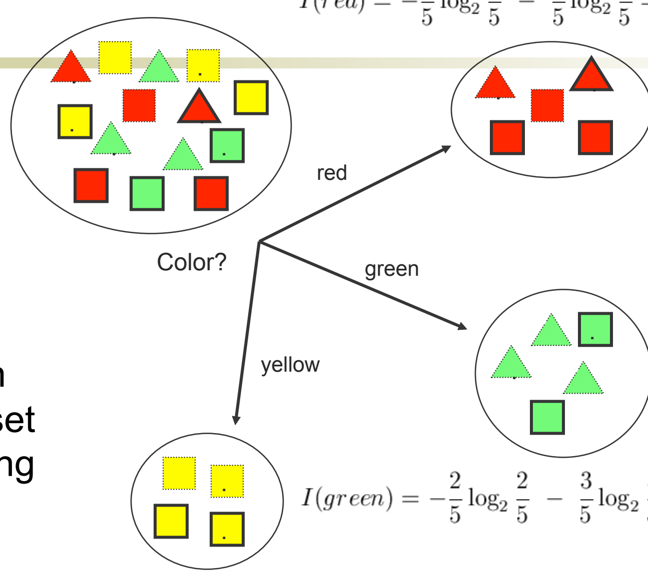
$$p(\triangle) = \frac{5}{14}$$

- entropy

$$I = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940 \text{ bits}$$

---

$$I(red) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971 \text{ bits}$$

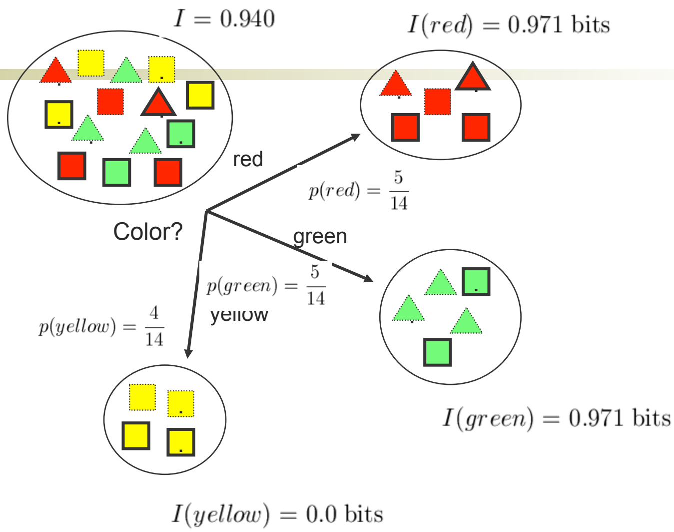



Entropy reduction by data set partitioning

$$I(green) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971 \text{ bits}$$

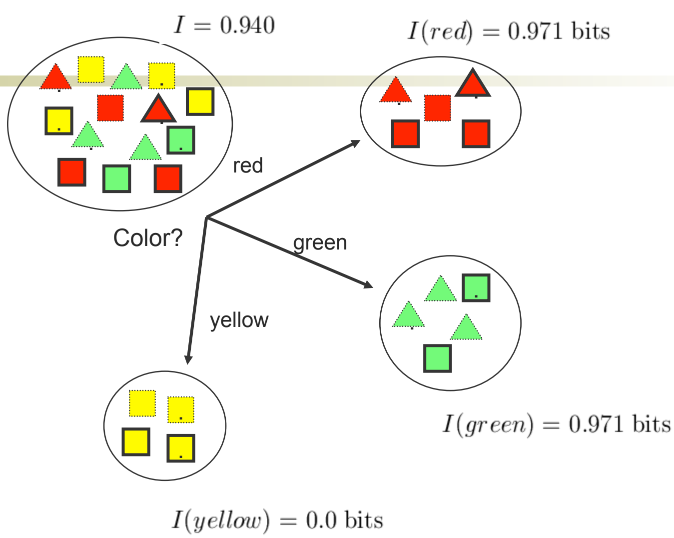
$$I(yellow) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0.0 \text{ bits}$$

## Entropija vrednosti atributa




$$I_{res}(\text{Color}) = \sum p(v)I(v) = \frac{5}{14}0.971 + \frac{5}{14}0.971 + \frac{4}{14}0.0 = 0.694\ bits$$

## Information Gain

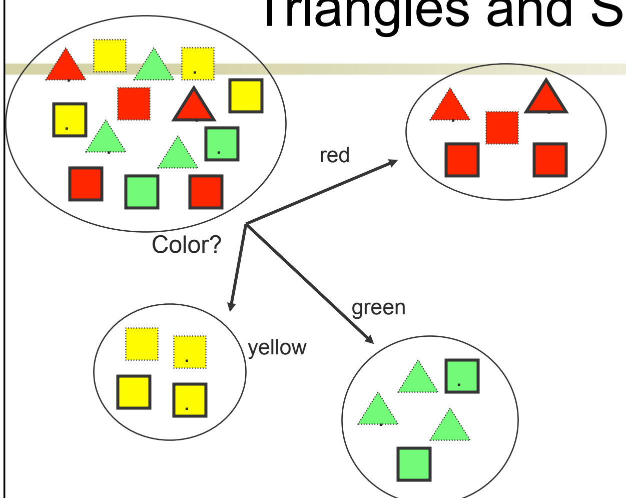



$$Gain(\text{Color}) = I - I_{res}(\text{Color}) = 0.940 - 0.694 = 0.246\ bits$$

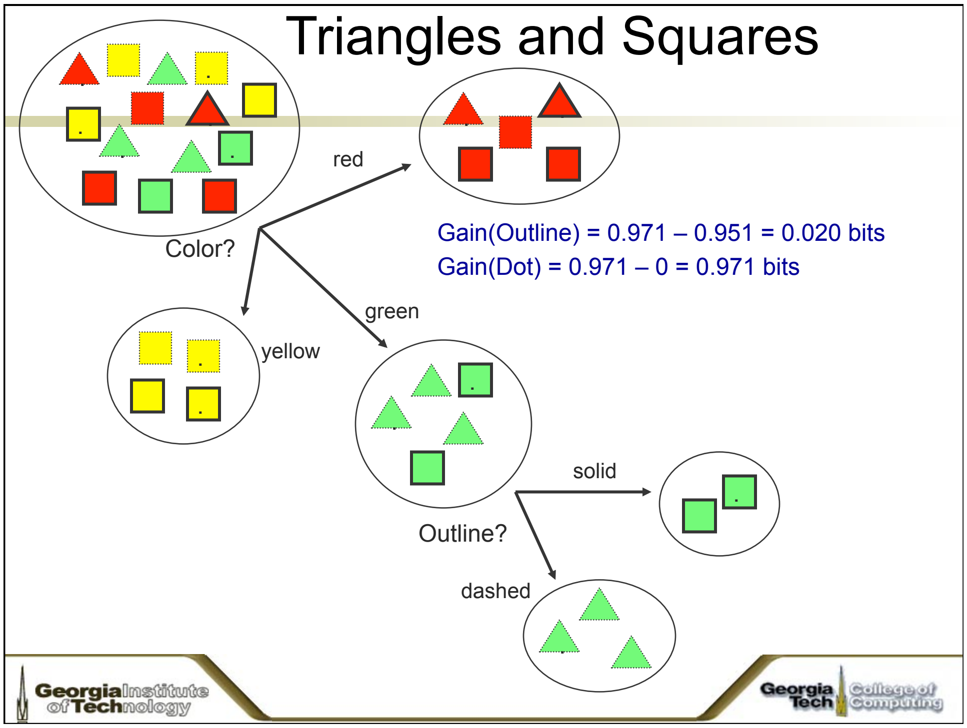
# Information Gain of The Attribute

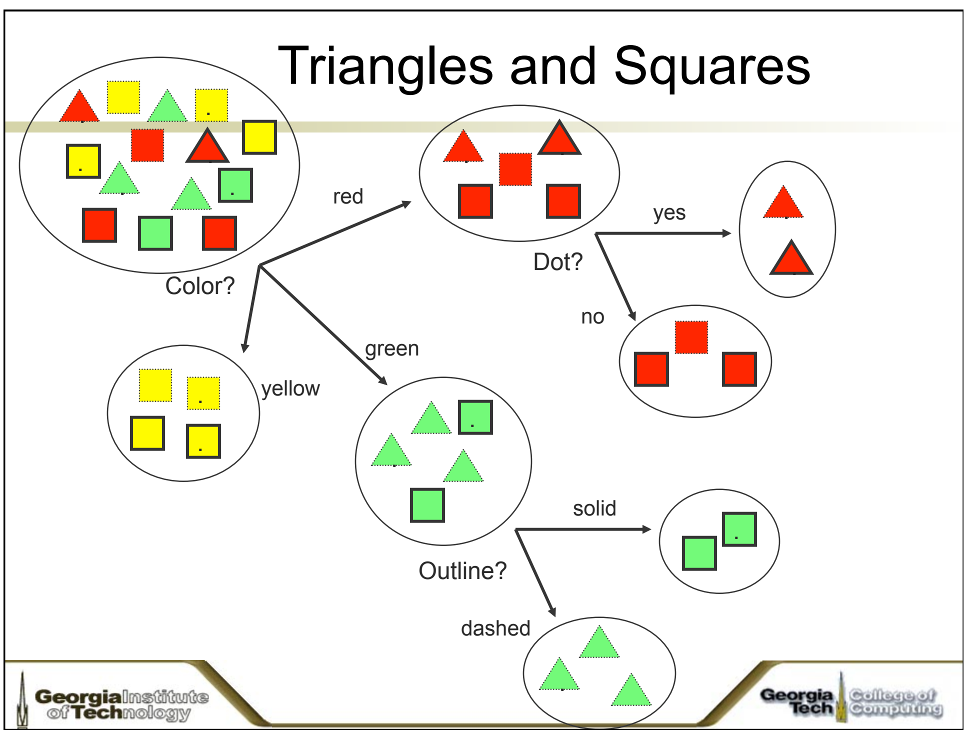
- Attributes
  - Gain(Color) = 0.246
  - Gain(Outline) = 0.151
  - Gain(Dot) = 0.048
- Heuristics:
  - attribute with the highest gain is chosen
- This heuristics is local (local minimization of impurity)

# Triangles and Squares (cont)

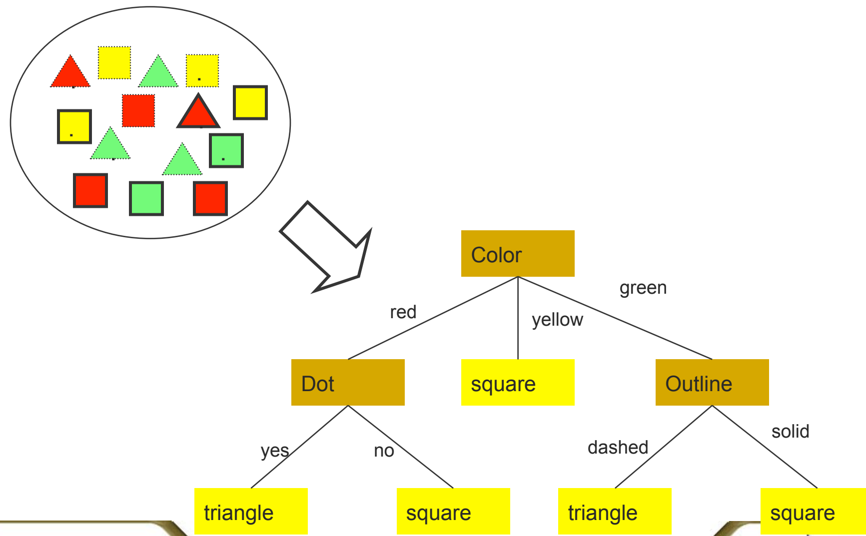



Gain(Outline) = 0.971 – 0 = 0.971 bits
Gain(Dot) = 0.971 – 0.951 = 0.020 bits


Triangles and Squares
red
Color?
Gain(Outline) = 0.971 – 0.951 = 0.020 bits
Gain(Dot) = 0.971 – 0 = 0.971 bits
green
yellow
solid
Outline?
dashed
Georgia Institute of Technology
Georgia Tech College of Computing


Triangles and Squares
red
Color?
yes
Dot?
no
green
yellow
solid
Outline?
dashed
Georgia Institute of Technology
Georgia Tech College of Computing

# Decision Tree






# A Defect of *Ires*

- *Ires* favors attributes with many values
- Such attribute splits S to many subsets, and if these are small, they will tend to be pure anyway
- One way to rectify this defect is through a corrected measure of **information gain ratio**.

Georgia Institute of Technology
Georgia Tech College of Computing

# Information Gain Ratio

- I(A) is amount of information needed to determine the value of an attribute A

$$I(A) = -\sum_{v} p(v)\log_2(p(v))$$

- Information gain ratio

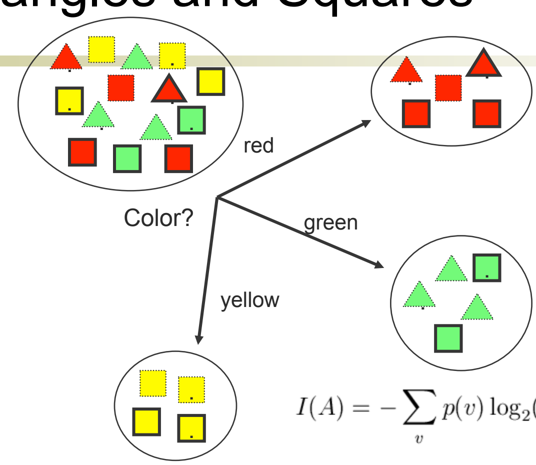
$$GainRatio(A) = \frac{Gain(A)}{I(A)} = \frac{I - I_{res}(A)}{I(A)}$$

Georgia Institute of Technology

Georgia Tech College of Computing

# Triangles and Squares

Information Gain Ratio




$$I(A) = -\sum_{v} p(v)\log_2(p(v))$$

$$I(\text{Color}) = -\frac{5}{14}\log_2\frac{5}{14} - \frac{5}{14}\log_2\frac{5}{14} - \frac{4}{14}\log_2\frac{4}{14} = 1.58 \text{ bits}$$

$$GainRatio(\text{Color}) = \frac{Gain(\text{Color})}{I(\text{Color})} = \frac{0.940 - 0.694}{1.58} = 0.156$$

Georgia Institute of Technology

Georgia Tech College of Computing

## Information Gain and Information Gain Ratio

| A | \|v(A)\| | Gain(A) | GainRatio(A) |
|---|---|---|---|
| Color | 3 | 0.247 | 0.156 |
| Outline | 2 | 0.152 | 0.152 |
| Dot | 2 | 0.048 | 0.049 |

## Measures of Node Impurity

- Entropy
- Gini Index
- Misclassification error

# Gini Index

- Another sensible measure of impurity (i and j are classes)

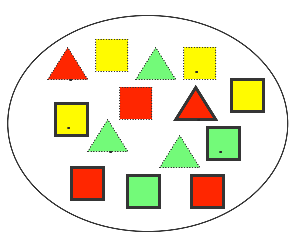
$$Gini = \sum_{i \neq j} p(i)p(j)$$

- After applying attribute A, the resulting Gini index is

$$Gini(A) = \sum_{v} p(v) \sum_{i \neq j} p(i|v)p(j|v)$$

- Gini can be interpreted as **expected error rate**

# Gini Index



$$p(\square) = \frac{9}{14}$$

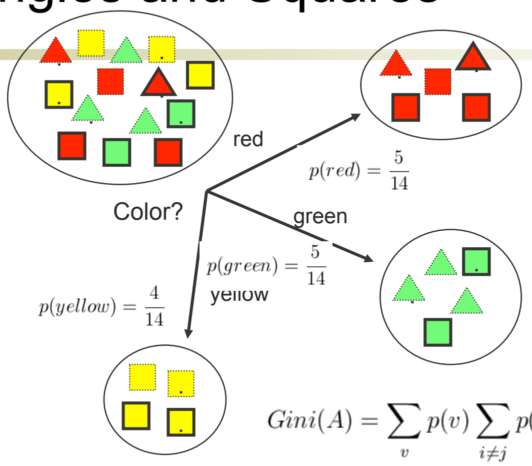
$$p(\triangle) = \frac{5}{14}$$

$$Gini = \sum_{i \neq j} p(i)p(j)$$

$$Gini = \frac{9}{14} \times \frac{5}{14} = 0.230$$

## Triangles and Squares

### Gini Index for Color




$$Gini(A) = \sum_v p(v) \sum_{i \neq j} p(i|v)p(j|v)$$

$$Gini(\text{Color}) = \frac{5}{14} \times (\frac{3}{5} \times \frac{2}{5}) + \frac{5}{14} \times (\frac{2}{5} \times \frac{3}{5}) + \frac{4}{14} \times (\frac{4}{4} \times \frac{0}{4}) = 0.171$$

## Gain of Gini Index

$$Gini = \frac{9}{14} \times \frac{5}{14} = 0.230$$

$$Gini(\text{Color}) = \frac{5}{14} \times (\frac{3}{5} \times \frac{2}{5}) + \frac{5}{14} \times (\frac{2}{5} \times \frac{3}{5}) + \frac{4}{14} \times (\frac{4}{4} \times \frac{0}{4}) = 0.171$$

$$\boxed{GiniGain(\text{Color}) = 0.230 - 0.171 = 0.058}$$

## Three Impurity Measures

| A | Gain(A) | GainRatio(A) | GiniGain(A) |
|---|---|---|---|
| Color | 0.247 | 0.156 | 0.058 |
| Outline | 0.152 | 0.152 | 0.046 |
| Dot | 0.048 | 0.049 | 0.015 |

(1) These impurity measures assess the effect of a single attribute
(2) The criterion "most informative" that they define is local (and "myopic")
(3) It does not reliably predict the effect of several attributes applied jointly



## Algorithm for decision tree learning

- Basic algorithm (a greedy **divide-and-conquer** algorithm)
  - Assume attributes are categorical (continuous attributes can be handled too)
  - Tree is constructed in a top-down recursive manner
  - At start, all the training examples are at the root
  - Examples are partitioned recursively based on selected attributes
  - Attributes are selected on the basis of an impurity function (e.g., information gain)
- Conditions for stopping partitioning
  - All examples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority class is the leaf
  - There are no examples left

## Choose an attribute to partition data

- The *key* to building a decision tree - which attribute to choose in order to branch.
- The objective is to reduce impurity or uncertainty in data as much as possible.
  - A subset of data is pure if all instances belong to the same class.
- The *heuristic* in C4.5 is to choose the attribute with the maximum Information Gain or Gain Ratio based on information theory.



## Decision tree: Highlight

- Decision tree learning is one of the most widely used techniques for classification
  - Its classification accuracy is competitive with other methods, and
  - it is very efficient
- The classification model is a tree, called decision tree.
- C4.5 by Ross Quinlan is perhaps the best known system.

# Example: C4.5

- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - Needs out-of-core sorting.

- You can download the software from: http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz

# Popular Decision Tree Packages

- ID3 (ID4, ID5, …) [Quinlan]
  - research code with many variations introduced to test new ideas
- CART: Classification and Regression Trees [Breiman]
  - best known package to people outside machine learning
  - 1st chapter of CART book is a good introduction to basic issues
- C4.5 (C5.0) [Quinlan]
  - most popular package in machine learning community
  - both decision trees and rules
- IND (INDuce) [Buntine]
  - decision trees for Bayesians (good at generating probabilities)
  - available from NASA Ames for use in U.S.

## Scaling Up

- ID2, C4.5, etc. assume that data fits into the main memory (ok for up to hundreds of thousands of examples)
- SPRINT, SLIQ: multiple sequential scans of data (ok for up to millions of examples)
- VFDT at most for one sequential scan of data (ok for up to billions of examples)



## Why Decision Tree Model?

- Relatively fast compared to other classification models
- Obtain similar and sometimes better accuracy compared to other models
- Simple and easy to understand
- Can be converted into simple and easy to understand classification rules

## When to Use Decision Trees

- Regression does not work
- Model intelligibility is important
- Problem does not depend on many features
  - Modest subset of features contains relevant info
  - not vision
- Speed of learning is important
- Linear combinations of features not critical
- Medium to large training sets

Georgia Institute of Technology

Georgia Tech College of Computing

## Decision Trees: Summary

- Representation=decision trees
- Bias=preference for small decision trees
- Search algorithm=
- Heuristic function=information gain or information content or others
- Overfitting and pruning
- Advantage is simplicity and easy conversion to rules.

Georgia Institute of Technology

90

Georgia Tech College of Computing

## Current Research

- Increasing representational power to include M-of-N splits, non-axis-parallel splits, perceptron-like splits, …
- Handling real-valued attributes better
- Using DTs to explain other models such as neural nets
- Incorporating background knowledge
- TDIDT on really large datasets
  - >> $10^6$ training cases
  - >> $10^3$ attributes
- Better feature selection
- Unequal attribute costs
- Decision trees optimized for metrics other than accuracy



## Questions

## Classification and regression

- Decision tree induction
- Bayesian Classification
- Bayesian Belief Networks
- Regression
- Support Vector Machine (SVM)



## What is Bayesian Classification?

- Bayesian classifiers are statistical classifiers
- For each new sample they provide a probability that the sample belongs to a class (for all classes)
- Example:
  - sample John (age=27, income=high, student=no, credit_rating=fair)
  - P(John, buys_computer=yes) = 20%
  - P(John, buys_computer=no) = 80%

## Naive Bayesian Classifier Example

naive hypothesis:
**attributes are conditionally independent:**

play tennis?

| Outlook | Temperature | Humidity | Windy | Class |
|---|---|---|---|---|
| sunny | hot | high | false | N |
| sunny | hot | high | true | N |
| overcast | hot | high | false | P |
| rain | mild | high | false | P |
| rain | cool | normal | false | P |
| rain | cool | normal | true | N |
| overcast | cool | normal | true | P |
| sunny | mild | high | false | N |
| sunny | cool | normal | false | P |
| rain | mild | normal | false | P |
| sunny | mild | normal | true | P |
| overcast | mild | high | true | P |
| overcast | hot | normal | false | P |
| rain | mild | high | true | N |



## Naive Bayesian Classifier Example

| Outlook | Temperature | Humidity | Windy | Class |
|---|---|---|---|---|
| overcast | hot | high | false | P |
| rain | mild | high | false | P |
| rain | cool | normal | false | P |
| overcast | cool | normal | true | P |
| sunny | cool | normal | false | P |
| rain | mild | normal | false | P |
| sunny | mild | normal | true | P |
| overcast | mild | high | true | P |
| overcast | hot | normal | false | P |

9

P = 9/14

| Outlook | Temperature | Humidity | Windy | Class |
|---|---|---|---|---|
| sunny | hot | high | false | N |
| sunny | hot | high | true | N |
| rain | cool | normal | true | N |
| sunny | mild | high | false | N |
| rain | mild | high | true | N |

5

N = 5/14

## Naive Bayesian Classifier Example

- Given the training set, we compute the probabilities:

| Outlook | P | N | Humidity | P | N |
|---|---|---|---|---|---|
| sunny | 2/9 | 3/5 | high | 3/9 | 4/5 |
| overcast | 4/9 | 0 | normal | 6/9 | 1/5 |
| rain | 3/9 | 2/5 | | | |
| Tempreature | | | Windy | | |
| hot | 2/9 | 2/5 | true | 3/9 | 3/5 |
| mild | 4/9 | 2/5 | false | 6/9 | 2/5 |
| cool | 3/9 | 1/5 | | | |

- We also have the probabilities
  - P = 9/14
  - N = 5/14

## Naive Bayesian Classifier: An Example

- To classify a new sample X:
  - outlook = sunny, temperature = cool, humidity = high, windy = false
- Prob(P|X) = Prob(P)*Prob(sunny|P)*Prob(cool|P)* Prob(high|P)*Prob(false|P) = 9/14*2/9*3/9*3/9*6/9 = 0.01
- Prob(N|X) = Prob(N)*Prob(sunny|N)*Prob(cool|N)* Prob(high|N)*Prob(false|N) = 5/14*3/5*1/5*4/5*2/5 = 0.013
- Therefore X takes class label N

## Naive Bayesian Classifier: Example 2

- Second example X = <rain, hot, high, false>
- P(X|p)·P(p) =
  P(rain|p)·P(hot|p)·P(high|p)·P(false|p)·P(p) =
  3/9·2/9·3/9·6/9·9/14 = 0.010582
- P(X|n)·P(n) =
  P(rain|n)·P(hot|n)·P(high|n)·P(false|n)·P(n) =
  2/5·2/5·4/5·2/5·5/14 = 0.018286
- Sample X is classified in class N (don't play)

## Naive Bayesian Classifier

- The classification problem is formalized using a-posteriori probabilities:
- P(C|X) = prob. that the sample tuple $X=<x_1,\ldots,x_k>$ is of class C.
  - E.g. P(class=N | outlook=sunny,windy=true,…)
- Assign to sample X the class label C such that P(C|X) is maximal
- Naïve assumption: attribute independence

$$P(x_1,\ldots,x_k|C) \sim P(x_1|C)\cdot\ldots\cdot P(x_k|C)$$

# The independence hypothesis…

- Inherent Problems
  - … makes computation possible
  - … yields optimal classifiers when satisfied
  - … but is seldom satisfied in practice, as attributes (variables) are often correlated.
- Attempts to overcome this limitation:
  - Bayesian networks, which combine Bayesian reasoning with causal relationships between attributes
  - Decision trees, which reason on one attribute at a time, considering most important attributes first