**Classification Techniques (2)**

*Classification Using Decision Trees*
- A partitioning based technique
  - Divides the search space into rectangular regions
- Each tuple is placed into a class based on the region within which it falls
- Internal nodes associated with attribute and arcs with values for that attribute
- DT approaches differ in how the tree is built
- Algorithms: Hunt’s, ID3, C4.5, CART

*Decision Tree*

Given:
- $D = \{t_1, \ldots, t_n\}$ where $t \in \{t_1, \ldots, t_n\}$
- Database schema contains $(A_1, A_2, \ldots, A_h)$
- Classes $C = \{C_1, \ldots, C_m\}$

Decision or Classification Tree is a tree associated with $D$ such that:
- Each internal node is labeled with attribute, $A_i$
- Each arc is labeled with predicate which can be applied to attribute at parent
- Each leaf node is labeled with a class, $C_j$

*Example of a Decision Tree*

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>MarSt</th>
<th>TaxInc</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>&gt; 80K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>&gt; 80K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>&lt; 80K</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>&gt; 80K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>&lt; 80K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>&lt; 80K</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>&lt; 80K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>&lt; 80K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>&lt; 80K</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>&gt; 80K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Another Example of Decision Tree**

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>MarSt</th>
<th>TaxInc</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>&gt; 80K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>&gt; 80K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>&lt; 80K</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>&gt; 80K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>&lt; 80K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>&lt; 80K</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>&lt; 80K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>&lt; 80K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>&lt; 80K</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>&gt; 80K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

There could be more than one tree that fits the same data!
**Decision Tree Classification Task**

<table>
<thead>
<tr>
<th>Tid</th>
<th>Class</th>
<th>Attrib3</th>
<th>Attrib2</th>
<th>Attrib1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Apply Model to Test Data**

Start from the root of tree.

- **Refund**: No
  - **Marital Status**: Single, Divorced
    - **Taxable Income**: < 80K
      - **Refund**: No
    - **Taxable Income**: > 80K
      - **Refund**: Yes

- **Refund**: Yes
  - **Marital Status**: Married
    - **Taxable Income**: < 80K
      - **Refund**: No
    - **Taxable Income**: > 80K
      - **Refund**: Yes

**Apply Model to Test Data**

- **Refund**: No
  - **Marital Status**: Single, Divorced
    - **Taxable Income**: < 80K
      - **Refund**: No
    - **Taxable Income**: > 80K
      - **Refund**: Yes

- **Refund**: Yes
  - **Marital Status**: Married
    - **Taxable Income**: < 80K
      - **Refund**: No
    - **Taxable Income**: > 80K
      - **Refund**: Yes
Apply Model to Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Single</td>
<td>&gt; 80K</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Married</td>
<td>&lt; 80K</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>Divorced</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Single</td>
<td>&gt; 80K</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Married</td>
<td>&lt; 80K</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>Divorced</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Decision Tree Classification Task

Decision 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

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 to 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, then use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

Hunt’s Algorithm

Decision 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
DT Split Areas

Gender

M

F

1.0

2.5

Height

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.

  - CarType
    - Family, Luxury
    - Sports

  - Size
    - Small, Medium, Large

  OR

  - CarType
    - {Sports, Luxury} {Family, Luxury}
    - {Sports, Luxury} {Family}

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.

  - (Small, Medium) OR (Medium, Large)

  - Size
    - Small, Medium, Large

  OR

  - (Small, Large) OR (Medium, Large)

  - Size
    - Small, Medium, Large

  OR

  - (Small, Large) OR (Medium)

  - Size
    - Small, Medium

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 ≥ v)
    - considers all possible splits and finds the best cut
    - can be more compute intensive

Splitting Based on Continuous Attributes

- **Binary split**: (A < v) or (A ≥ v)

  - Yes
  - No

  - Taxable Income
    - > 80K
    - < 10K
    - [10K, 25K]
    - [25K, 50K]
    - [50K, 80K]

  (i) Binary split

  (ii) Multi-way split
Comparing Decision Trees

Balanced

Deep

DT Induction Issues that affect Performance

- Choosing Splitting Attributes
- Ordering of Splitting Attributes
- Split Points
- Tree Structure
- Stopping Criteria
- Training Data (size of)
- Pruning

How to determine the Best Split

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

Which test condition is the best?

How to Find the Best Split

Before Splitting: M0

Gain = M0 – M12 vs M0 – M34

Measure of Impurity: GINI

- Gini Index for a given node \( t \):

\[
GINI(t) = 1 - \sum_{j} \left[ p(j | t) \right]^2
\]

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

- Maximum (1 - \( 1/n \)) 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

Before Splitting:

Gain = M0 – M12 vs M0 – M34
Examples for computing GINI

\[ \text{GINI}(i) = 1 - \sum_j (p(j|i))^2 \]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(C_1)</td>
<td>0</td>
<td>(P(C_1) = \frac{0}{6} = 0)</td>
</tr>
<tr>
<td>(C_2)</td>
<td>6</td>
<td>(P(C_2) = \frac{6}{6} = 1)</td>
</tr>
<tr>
<td>(C_1)</td>
<td>1</td>
<td>(P(C_1) = \frac{1}{6})</td>
</tr>
<tr>
<td>(C_2)</td>
<td>5</td>
<td>(P(C_2) = \frac{5}{6})</td>
</tr>
</tbody>
</table>

\[ \text{Gini} = 1 - P(C_1) - P(C_2) = 1 - 0 - 1 = 0 \]

\[ \sum_{i=1}^{k} \frac{n_i \cdot \text{GINI}(i)}{n} \]

\[ \text{GINI}_{\text{split}} = 1 - \left( \frac{1}{6} \right)^2 - \left( \frac{5}{6} \right)^2 = 0.278 \]

\[ \text{GINI}_{\text{split}} = 1 - \left( \frac{2}{6} \right)^2 - \left( \frac{4}{6} \right)^2 = 0.444 \]

Splitting Based on GINI

- Used in CART
- When a node is split into \(k\) partitions (children), the quality of split is computed as:

\[ \text{GINI}_{\text{split}} = 1 - \sum_{i=1}^{k} \frac{n_i \cdot \text{GINI}(i)}{n} \]

where, \(n_i\) = number of records at child \(i\), \(n\) = number of records at node \(p\).

Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for.

\[ \text{Gini} = 1 - \left( \frac{3}{6} \right)^2 - \left( \frac{2}{6} \right)^2 = 0.194 \]

\[ \text{Gini} = 1 - \left( \frac{2}{6} \right)^2 - \left( \frac{4}{6} \right)^2 = 0.528 \]

Continuous Attributes: Computing GINI Index

- Use Binary Decisions based on one value
- Several choices for the splitting value
  - Number of possible splitting values
  - Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, \(A < v\) and \(A \geq v\)
- Simple method to choose best \(v\)
  - For each \(v\), scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient
  - Repetition of work.

Categorical Attributes: Computing GINI Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Continuous Attributes: Computing GINI Index

- For efficient computation for each attribute,
  - Sort the attribute on values
  - Linearly scan values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index
Information

Decision Tree Induction is often based on Information Theory.

```

```

DT Induction

- When all the marbles in the bowl are mixed up, little information is given.
- When the marbles in the bowl are all from one class and those in the other two classes are on either side, more information is given.

*Use this approach with DT Induction!*

Information/Entropy

Given probabilities $p_1, p_2, ..., p_s$ whose sum is 1, 

**Entropy** is defined as:

$$H(p_1, p_2, ..., p_s) = \sum_{i=1}^{s} (p_i \log(1/p_i))$$

- Entropy measures the amount of randomness or surprise or uncertainty.
- Goal in classification
  - no surprise
  - entropy $= 0$

Entropy

```

```

ID3

- Creates a decision tree using information theory concepts and tries to reduce the expected number of comparisons.
- ID3 chooses to split on an attribute that gives the highest information gain:

$$Gain(D, S) = H(D) - \sum_{i=1}^{s} P(D_i)H(D_i)$$

Height Example Data

```

```

```
ID3 Example (Output1)

- Starting state entropy:
  \[ \frac{4}{15} \log(15/4) + \frac{8}{15} \log(15/8) + \frac{3}{15} \log(15/3) = 0.4384 \]
- Gain using gender:
  - Female: \( \frac{3}{9} \log(9/3) + \frac{6}{9} \log(9/6) = 0.2764 \)
  - Male: \( \frac{1}{6} \log(6/1) + \frac{2}{6} \log(6/2) + \frac{3}{6} \log(6/3) = 0.4392 \)
  - Weighted sum: \( \frac{9}{15} \times 0.2764 + \frac{6}{15} \times 0.4392 = 0.34152 \)
- Gain using height:
  \[ 0.4384 - \left( \frac{2}{15} \times 0.301 \right) = 0.3983 \]
- Choose height as first splitting attribute

C4.5 Algorithm

- ID3 favors attributes with large number of divisions (is vulnerable to overfitting)
- Improved version of ID3:
  - Missing Data
  - Continuous Data
  - Pruning
  - Rules
  - GainRatio:
    \[ \text{GainRatio}(D,S) = \frac{\text{Gain}(D,S)}{\text{Gain}(D,S) - \text{Gain}(D,S)} \]
    - Takes into account the cardinality of each split area

CART: Classification and Regression Trees

- Creates a Binary Tree
- Uses entropy to choose the best splitting attribute and point
- Formula to choose split point, s, for node t:
  \[ \Phi(s/t) = 2P_LP_R \sum_{j=1}^{m} | P(C_j | t_L) - P(C_j | t_R) | \]
- \( P_L, P_R \) probability that a tuple in the training set will be on the left or right side of the tree.

CART Example

- At the start, there are six choices for split point (right branch on equality):
  - \( \Phi(\text{Gender}) = 2(6/15)(9/15)(2/15 + 4/15 + 3/15) = 0.224 \)
  - \( \Phi(1.6) = 0 \)
  - \( \Phi(1.7) = 2(2/15)(13/15)(0 + 8/15 + 3/15) = 0.169 \)
  - \( \Phi(1.8) = 2(5/15)(10/15)(4/15 + 6/15 + 3/15) = 0.385 \)
  - \( \Phi(1.9) = 2(9/15)(6/15)(4/15 + 2/15 + 3/15) = 0.256 \)
  - \( \Phi(2.0) = 2(12/15)(3/15)(4/15 + 8/15 + 3/15) = 0.32 \)
- Split at 1.8

Decision Tree Based Classification

- Advantages:
  - Inexpensive to construct
  - 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

Decision Boundary

- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time
Oblique Decision Trees

- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive

Tree Replication

- Same subtree appears in multiple branches

Classification Using Rules

- Perform classification using If-Then rules
- Classification Rule: \( r = <a, c> \)
  - Antecedent, Consequent
- May generate rules from other techniques (DT, NN) or generate directly.
- Algorithms: Gen, RX, 1R, PRISM

Generating Rules from Decision Trees

Input:
- \( T \) //Decision Tree
Output:
- \( R \) //Rules
Gen Algorithm:
- Illustrate simple approach to generating classification rules from a DT
  - \( B = 2 \)
  - for each path from root to a leaf in \( T \) do
    - if \( a = \text{true} \) for each internal node do
      - \( n = \# \text{leafs} + \# \text{null leafs} + \text{number of attribute tests} \)
    - \( c = \text{label of leaf} \) value
    - \( B = R(j) \leq a \times n \)

Generating Rules Example

<table>
<thead>
<tr>
<th>Rule</th>
<th>Antecedent</th>
<th>Consequent</th>
</tr>
</thead>
<tbody>
<tr>
<td>( &lt; \text{Height} \leq 180, \text{Hair} )</td>
<td>( \text{Height} &gt; 180 )</td>
<td>( \text{Height} \leq 180 )</td>
</tr>
</tbody>
</table>

1R Algorithm

Input:
- \( D \) //Training data
- \( A \) //Attributes to consider for rules
- \( C \) //Classes
Output:
- \( R \) //Rules
1R Algorithm:
- 1R algorithm generates rules based on one attribute
  - \( B = \# \)
  - for each \( A \in A \)
    - for each possible value, \( v \), of \( A \) do
      - \( R_v = \# \text{cases} \) such that \( v \in \text{cases} \)
      - \( R_v = \text{number of cases correctly classified by } R_v \) when
      - \( B = \# \text{cases} \) where \( B \) is minimum.
**1R Example**

<table>
<thead>
<tr>
<th>Option</th>
<th>Attribute</th>
<th>Rules</th>
<th>Errors</th>
<th>Total Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Gender</td>
<td>F → Medium 4/9</td>
<td>6/15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M → Tall</td>
<td>4/6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Height</td>
<td>(0,1,0) → Short 0/2</td>
<td>1/15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1,6,1)</td>
<td>Short 0/2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1,7,8)</td>
<td>Medium 0/3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1,8,1,0)</td>
<td>Medium 0/4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1,9,2,0)</td>
<td>Medium 1/2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(2,0,x)</td>
<td>Tall 0/2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PRISM Algorithm**

```
Algorithm PRISM
Input: D //Training data
C //Classes
R //Rules
PRISM Algorithm: //PRISM algorithm generates rules based on frequent attribute-value pairs

1. Set R = 0
2. For each C_i \in C do
   a. Find C_i \rightarrow x \subset T such that: |C_i \setminus x| = min
   b. For each attribute A such that a \leftarrow x, add rule:
      \begin{align*}
      \text{If } & A \leq x & \text{ then } & C_i \rightarrow x & \rightarrow \text{ rule } \\
      F & = \text{ rules in } R \text{ that match } A \leftarrow x & & \text{ rule } \\
      C & = \text{ Class } C_i & & \text{ rule }
      \end{align*}
   c. until there are no more rules in C
```

**Metrics for Performance Evaluation**

- **Class=Yes**
  - a (TP)
  - b (FN)
- **Class=No**
  - c (FP)
  - d (TN)

**Most widely-used metric:**

\[
\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}
\]

**Decision Tree vs. Rules**

- Tree has an implied order in which splitting is performed.
- Tree is created based on looking at all classes.
- Rules have no ordering of predicates.
- Only need to look at one class to generate its rules.

**PRISM Example**

```
Gender = F 0/9
Gender = M 3/6

If Gender = M then Class = Tall.
If Gender = M and Height in range 1 then Class = Tall.

1.0 \leq \text{Height} \leq 1.5 0/1
1.6 \leq \text{Height} < 1.7 0/1
1.7 \leq \text{Height} < 1.8 0/1
1.8 \leq \text{Height} < 1.9 0/1
1.9 \leq \text{Height} < 2.0 1/2
2.0 \leq \text{Height} 2/2
```

**Limitation of Accuracy**

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10
- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9%
  - Accuracy is misleading because model does not detect any class 1 example
### Estimating Classifier Accuracy

**IDEA:** Randomly select sampled partitions of the training data to estimate accuracy

- **Holdout method:**
  - Partition known data into two independent sets
  - Training set (usually 2/3 of data)
  - Test set (remaining 1/3)
  - Estimate of the accuracy of classifier is pessimistic

- **Random sub-sampling:**
  - Repeat the holdout method k times;
  - Overall accuracy estimate is taken as the average estimates obtained by the process.

- **K-fold cross-validation:**
  - Partition known data $S_i$ into $k$ mutually exclusive subsets (or “folds”) $S_1, S_2, \ldots, S_k$ of approximately equal size;
  - Use each $S_i$ as a test set
  - Accuracy estimate is the overall number of correct classifications divided by the total number of samples in the initial data

- **Leave-one-out:**
  - K-fold cross-validation with $k$ set to $|S|$.

### Increasing Classifier Accuracy

**Bagging:**
- Each classifier "votes";
- Winner class wins classification.

**Boosting:**
- Each classifier "votes";
- Votes are combined based on weights obtained by the estimates of each classifier’s accuracy;
- Winner class wins classification.

### Is Accuracy enough to judge a Classifier?

In practice, there are also other considerations

- Speed
- Robustness (influence of noisy data)
- Scalability (number of I/O operations)
- Interpretability of classification output