Classification Using Decision Trees

- **Partitioning based**: Divide search space into rectangular regions.
- Tuple is placed into a class based on the region within which it falls.
- DT approaches differ in how the tree is built: **DT Induction**
- Internal nodes associated with attribute and arcs with values for that attribute.
- Algorithms: ID3, C4.5, CART

Decision Tree

Given:
- \( D = \{ t_1, ..., t_n \} \) where \( t_i = (x_1, ..., x_n) \)
- Database schema contains \( (A_1, A_2, ..., A_n) \)
- Classes \( C = \{ C_1, ..., C_m \} \)

**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, \( C_j \)

DT Induction

Input:
\( D \) : Training data

Output:
\( T \) : Decision tree

DT Induction Algorithm:

1. Choose \( f \) 
2. Determine best splitting criterion
3. \( T \) Create root node and labeled with splitting attribute
4. Add arc to root node for each split predicate and label
5. for each arc do
6. \( U \) Database created by applying splitting predicate to \( D \)
7. if stopping condition reached for this path then
8. \( T' \) Create leaf node and label with appropriate class
9. else
10. \( T' = DT(U) \)
11. \( T \) Add \( T' \) to \( T \)
Comparing Decision Trees

DT Issues that Impact Performance
- Choosing Splitting Attributes
- Ordering of Splitting Attributes
- Split Points
- Tree Structure
- Stopping Criteria
- Training Data (size of)
- Pruning

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 $p_1, p_2, \ldots, p_n$ whose sum is 1, Entropy is defined as:

$$H(p_1, p_2, \ldots, p_n) = \sum_{i=1}^{n} (p_i \log(1/p_i))$$

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

Entropy

$\log (1/p)$

$H(p, 1-p)$
**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:

\[
Gain(D, S) = H(D) - \sum_{i=1}^{a} P(D_i)H(D_i)
\]

**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:

\[
GainRatio(D, S) = \frac{Gain(D, S)}{H(D)}
\]

  - Takes into account the cardinality of each split area

**ID3 Example (Output1)**

- Starting state entropy:
  \[
  \frac{4/15 \log(15/4)}{8/15 \log(15/8)} + \frac{3/15 \log(15/3)}{3} = 0.4384
  \]
- Gain using gender:
  - Female: \(\frac{3/9 \log(9/3)}{6/9 \log(9/6)} = 0.2764\)
  - Male: \(\frac{1/6 \log(6/1)}{2/6 \log(6/2)} = \frac{3/6 \log(6/3)}{3} = 0.4392\)
  - Weighted sum: \(9/15)(0.2764) + (6/15)(0.4392) = 0.34152\)
  - Gain: 0.4384 - 0.34152 = 0.09688
- Gain using height:
  \[
  0.4384 - (2/15)(0.301) = 0.3983
  \]
- Choose height as first splitting attribute

**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

**Classification Using Rules**

- Perform classification using If-Then rules
- **Classification Rule:** \(r = \langle a, c \rangle\)
  - Antecedent, Consequent
- May generate rules from other techniques (DT, NN) or generate directly.
- Algorithms: Gen, RX, IR, PRISM
Generating Rules from Decision Trees

Input: T //Decision Tree
Output: R //Rules

Algorithm:
//Illustrate simple approach to generating classification rules from a DT
R = ∅
for each path from root to a leaf in T do
  a = True
  for each internal node do
    a = a ∧ (label of parent node in path combined with label of incident arc)
    c = label of leaf node
    R = R ∪ {a–>c}

1R Algorithm

Input: D //Training data
R //Rules to consider for rules
C //Classes

Output: R //Rules
1R Algorithm:
//1R algorithm generates rules based on one attribute
R = ∅
for each A ∈ R do
  Rₐ = R
  for each possible value, v of A do
    //v may be a range rather than a specific value
    for each C ∈ C find count(v,C)
    //Here count is the number of occurrences of this class for this attribute
    let Cᵥ be the class with the largest count
    Rᵥ = Rₐ(A = v) → (class = Cᵥ)
  S(R) = number of tuples incorrectly classified by Rᵥ
  R = R ∪ {A–>Cᵥ} where S(Rₐ) is minimum

PRISM Algorithm

Algorithm 0.1
Input: D //Training data
C //Classes

Output: R //Rules
PRISM Algorithm:
//PRISM algorithm generates rules based on best attribute-value pairs
R = ∅
for each A ∈ C do
  repeat
    Y = ∅ //All instances of class Cᵥ will be systematically removed from T
    p = X //A bruteforce unit with empty left-hand side
    repeat
      for each attribute A value v pair found in T do
        calculate
        find x = v that maximizes this value
        p = p ∪ (A=x)
        Y = (tuples in T that satisfy p A = x)
        until there are no tuples in Y
      R = R ∪ {p–>Cᵥ}
      repeat
        repeat
          repeat
            until there are no tuples in Y which belong to Cᵥ
        R = R ∪ {p–>Cᵥ}
  until no more repeats are possible

Generating Rules Example

1R Example

Option | Attribute | Rules | Errors | Total Errors
---|---|---|---|---
1 | Gender | F → Medium | 3 | 0 | 6/15
   | | M → Tall | 3 | 0 | 3/6
2 | Height | (0,10) → Short | 3 | 0 | 1/15
   | | (10,15) → Tall | 3 | 0 | 1/15
   | | (15,20) → Medium | 3 | 1 | 0/15
   | | (20,∞) → Tall | 3 | 0 | 0/15

PRISM Example

Gender = F 0/9
Gender = M 3/6
If Gender = M and Height in range? then Class = Tall.
If Gender = M and Height > range? then Class = Tall.
Height <= 1.6 0/9
1.6 < Height <= 1.7 0/1
1.7 < Height <= 1.8 0/2
1.8 < Height <= 1.9 0/1
1.9 < Height <= 2.0 1/2
2.0 < Height 2/2
If Gender = M and Height > 2 then Class = Tall.
**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.

---

**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.

---

**Estimating Classifier Accuracy**

- **K-fold cross-validation:**
  - Partition known data $S$ 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