Classification Techniques (2)

Overview

Previous Lecture

- Classification Problem
- Classification based on Regression
- Distance-based Classification (KNN)

This Lecture

- Classification using Decision Trees
- Classification using Rules
- Quality of Classifiers

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 $t_i = \langle t_{i1}, \ldots, t_{ih} \rangle$
- Database schema contains $\{A_1, A_2, \ldots, A_h\}$
- Classes $C = \{C_1, \ldots, C_m\}$

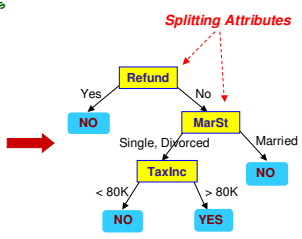
Decision or *Classification Tree* is a tree associated with D such that

- 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

	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

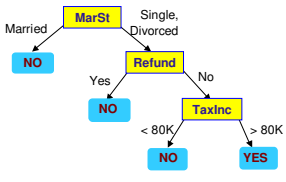
Training Data



Model: Decision Tree

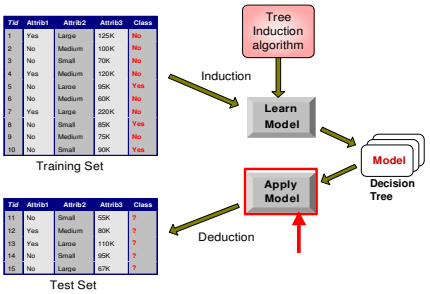
Another Example of Decision Tree

	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

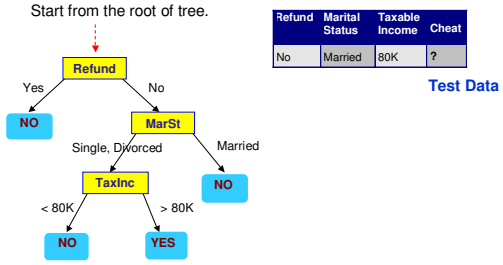


There could be more than one tree that fits the same data!

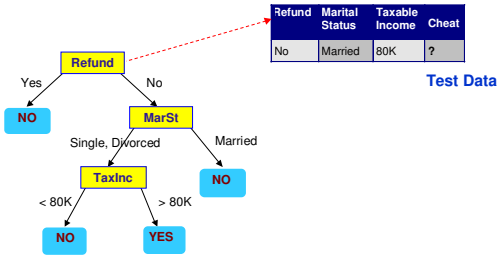
Decision Tree Classification Task



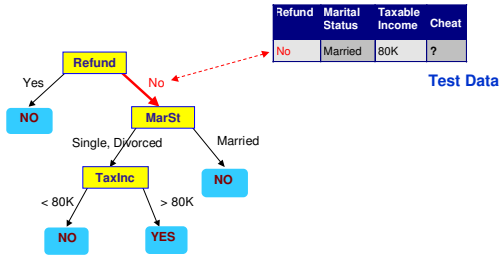
Apply Model to Test Data



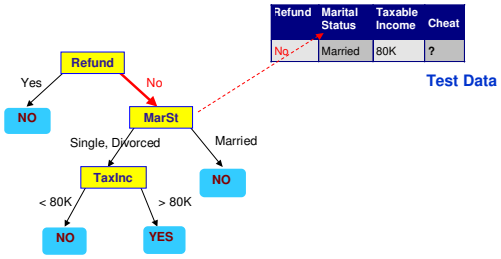
Apply Model to Test Data



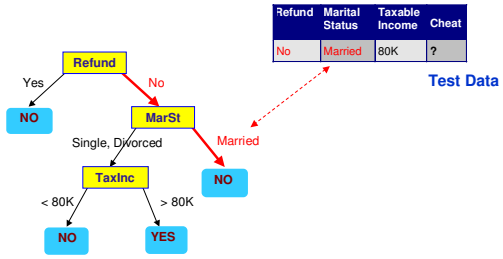
Apply Model to Test Data



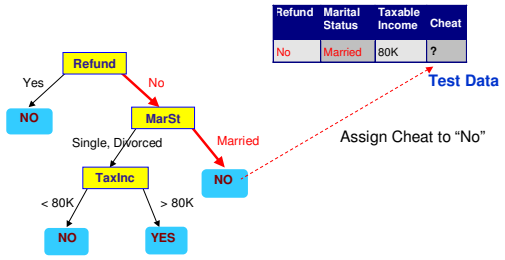
Apply Model to Test Data



Apply Model to Test Data

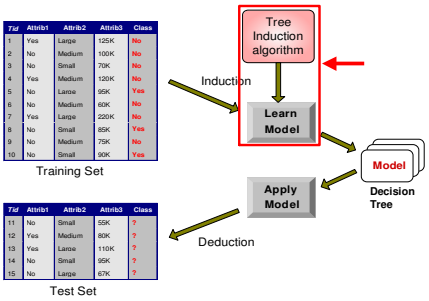


Apply Model to Test Data



Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

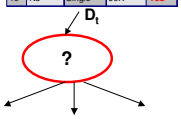
Decision Tree Classification Task



General Structure of Hunt's Algorithm

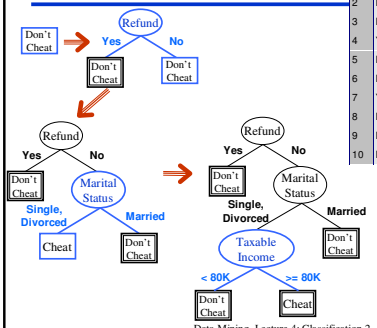
- Let D_t be the set of training records that reach a node t
- General Procedure:
  - If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t
  - If D_t is an empty set, then t is a leaf node labeled by the default class, y_d
  - If D_t contains records that belong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



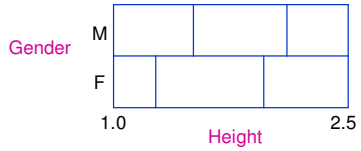
Decision Tree Induction

```
Input:
   D     //Training data
Output:
   T     //Decision Tree
DTBuild Algorithm:
         //Simplistic algorithm to illustrate naive approach to building DT
   T = ∅;
   Determine best splitting criterion;
   T = Create root node node and label with splitting attribute;
   T = Add arc to root node for each split predicate and label;
   for each arc do
      D = Database created by applying splitting predicate to D;
      if stopping point reached for this path then
         T' = Create leaf node and label with appropriate class;
      else
         T' = DTBuild(D);
      T = Add T' to arc;
```


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



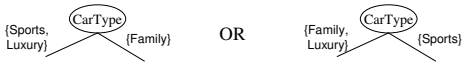
How to Specify Test Condition?

- 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

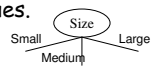
- Multi-way split: Use as many partitions as distinct values.

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

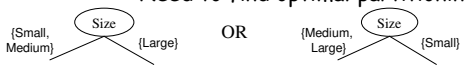


Splitting Based on Ordinal Attributes

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



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

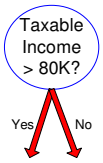


- What about this split?

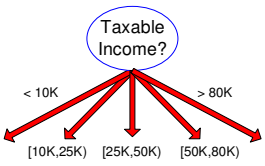
Splitting Based on Continuous Attributes

- 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 $(A \geq v)$
    - considers all possible splits and finds the best cut
    - can be more compute intensive

Splitting Based on Continuous Attributes

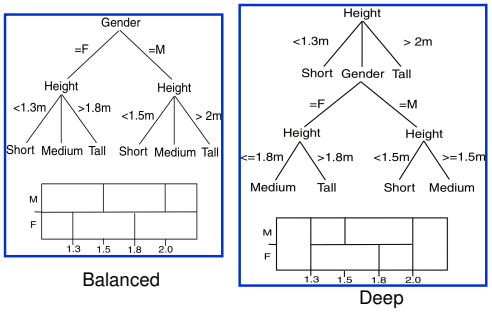


(i) Binary split



(ii) Multi-way split

Comparing Decision Trees

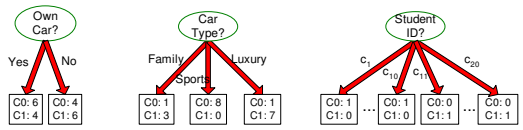


DT Induction Issues that affect Performance

- 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 determine the 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

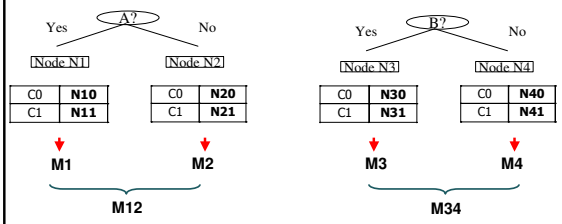
Homogeneous,
Low degree of impurity

How to Find the Best Split

Before Splitting:

C0	N00
C1	N01

→ **M0**



Gain = M0 – M12 vs M0 – M34

Measure of Impurity: GINI

- **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

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Examples for computing GINI

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

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

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1
Gini = 1 – P(C1)² – P(C2)² = 1 – 0 – 1 = 0

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

P(C1) = 1/6 P(C2) = 5/6
Gini = 1 – (1/6)² – (5/6)² = 0.278

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

P(C1) = 2/6 P(C2) = 4/6
Gini = 1 – (2/6)² – (4/6)² = 0.444

Splitting Based on GINI

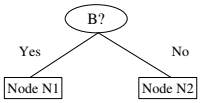
- Used in CART
- When a node p is split into k partitions (children), the quality of split is computed as:

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

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.



	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

Categorical Attributes: Computing GINI Index

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

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split (find best partition of values)

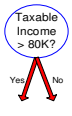
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

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 ≥ 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.

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



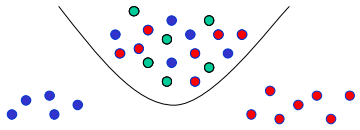
Continuous Attributes: Computing GINI Index

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

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Taxable Income										
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72	80	87	92	97	110	122	172	230
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >
Yes	0 3	0 3	0 3	0 3	1 2	2 1	3 0	3 0	3 0	3 0	3 0
No	0 7	1 6	2 5	3 4	3 4	3 4	3 4	4 3	5 2	6 1	7 0
Gini	0.420	0.400	0.375	0.343	0.417	0.400	0.300	0.343	0.375	0.400	0.420

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 !

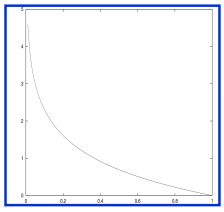
Information/Entropy

Given probabilities p_1, p_2, .., p_s whose sum is 1, *Entropy* is defined as:

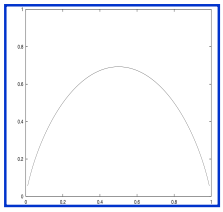
$$H(p_1, p_2, ..., p_s) = \sum_{i=1}^{s} (p_i log(1/p_i))$$

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

Entropy



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:

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

Height Example Data

Name	Gender	Height	Output1	Output2
Kristina	F	1.60	Short	Medium
Jim	M	2.02	Tall	Medium
Maggie	F	1.90	Medium	Tall
Martha	F	1.88	Medium	Tall
Stephanie	F	1.71	Short	Medium
Bob	M	1.85	Medium	Medium
Kathy	F	1.60	Short	Medium
Dave	M	1.72	Short	Medium
Worth	M	2.12	Tall	Tall
Steven	M	2.10	Tall	Tall
Debbie	F	1.78	Medium	Medium
Todd	M	1.95	Medium	Medium
Kim	F	1.89	Medium	Tall
Amy	F	1.81	Medium	Medium
Wynette	F	1.75	Medium	Medium

ID3 Example (Output1)

- Starting state entropy:
  4/15 log(15/4) + 8/15 log(15/8) + 3/15 log(15/3) = 0.4384
- Gain using gender:
  - Female: 3/9 log(9/3) + 6/9 log(9/6) = 0.2764
  - Male: 1/6 (log 6/1) + 2/6 log(6/2) + 3/6 log(6/3) = 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

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:

$$GainRatio(D,S) = \frac{Gain(D,S)}{H(\frac{|D_1|}{|D|},...,\frac{|D_s|}{|D|})}$$

  - 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:

$$\Phi(s/t) = 2P_L P_R \sum_{j=1}^{m} | P(C_j \mid t_L) - P(C_j \mid t_R) |$$

- P_L, P_R probability that a tuple in the training set will be on the left or right side of the tree.

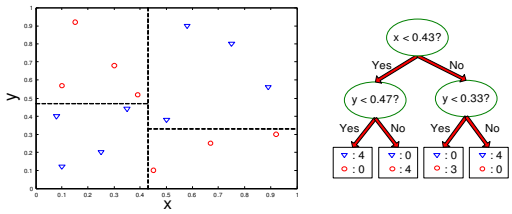
CART Example

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

Decision Tree Based Classification

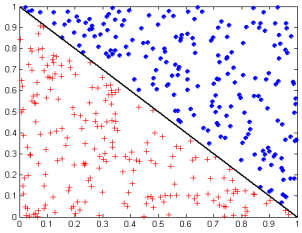
- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Accuracy is comparable to other classification techniques for many simple data sets

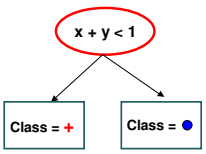
Decision Boundary



- **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**

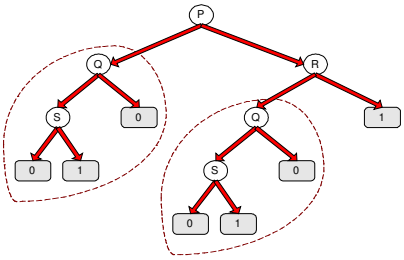
Oblique Decision Trees





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

Tree Replication



- Same subtree appears in multiple branches

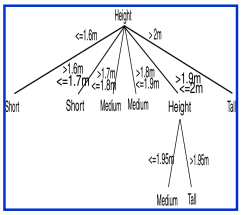
Classification Using Rules

- 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
  $R = \emptyset$
  for each path from root to a leaf in T do
    $a = True$
    for each internal node do
      $a = a \wedge (label\ of\ parent\ node\ in\ path\ combined\ with label\ of\ incident\ arc)$
      $c = label\ of\ leaf\ node$
    $R = R \cup r =< a, c >$

Generating Rules Example



$\{\langle (Height \leq 1.6m), Short \rangle,$
$\langle (Height > 1.6m) \wedge (Height \leq 1.7m), Short \rangle,$
$\langle (Height > 1.7m) \wedge (Height \leq 1.8m), Medium \rangle,$
$\langle (Height > 1.8m) \wedge (Height \leq 1.9m), Medium \rangle,$
$\langle (Height > 1.9m) \wedge (Height \leq 2m) \wedge (Height \leq 1.95m), Medium \rangle,$
$\langle (Height > 1.9m) \wedge (Height \leq 2m) \wedge (Height > 1.95m), Tall \rangle,$
$\langle (Height > 2m), Tall \rangle\}$

1R Algorithm

Input:
  D *//Training data*
  R *//Attributes to consider for rules*
  C *//Classes*
Output:
  R *//Rules*
1R Algorithm:
  //1R algorithm generates rules based on one attribute
  $R = \emptyset$;
  for each $A \in R$ **do**
    $R_A = \emptyset$;
    for each possible value, v, *of* A **do**
      //v may be a range rather than a specific value
      for each $C_j \in C$ *find* $count(C_j)$;
        // Here count is the number of occurences of this class for this attribute
      let C_m *be the class with the largest count;*
      $R_A = R_A \cup ((A = v) \rightarrow (class = C_m))$;
    $ERR_A =$ *number of tuples incorrectly classified by* R_A;
  $R = R_a$ *where* ERR_A *is minimum;*

1R Example

Option	Attribute	Rules	Errors	Total Errors
1	Gender	F → Medium	3/9	6/15
		M → Tall	3/6	
2	Height	(0,1.6] → Short	0/2	1/15
		(1.6,1.7] → Short	0/2	
		(1.7,1.8] → Medium	0/3	
		(1.8,1.9] → Medium	0/4	
		(1.9,2.0] → Medium	1/2	
		(2.0,∞) → Tall	0/2	

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
$R = \emptyset$;
for each $C_j \in C$ **do**
repeat
$T = D$; // All instances of class C_j will be systematically removed from T
$p = true$; // Create new rule with empty left hand side
$r = (\ If\ p\ then\ C_j)$;
repeat
for each attribute A value v pair found in T **do**
calculate $(\frac{|\{tuples \in T\ with\ A=v\} \wedge p \wedge \{C_j\}|}{|\{tuples \in T\ with\ A=v\} \wedge p|})$;
find $A = v$ **that maximizes this value;**
$p = p \wedge (A = v)$;
$T = \{tuples\ in\ T\ that\ satisfy\ A = v\}$;
until all tuples in T belong to C_j;
$D = D - T$;
$R = R \cup r$;
until there are no tuples in D which belong to C_j;

PRISM Example

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

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

$Height <= 1.6$ **0/0**
$1.6 < Height <= 1.7$ **0/1**
$1.7 < Height <= 1.8$ **0/0**
$1.8 < Height <= 1.9$ **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.

Metrics for Performance Evaluation…

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- Most widely-used metric:

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

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.

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 weighs 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