Data-driven Intelligent Systems

Lecture 8
Decision Trees and Classification

KNOWLEDGE
TECHNOLOGY

http://www.informatik.uni-hamburg.de/WTM/

Overview

- Advanced decision trees
  - Continuous attributes
  - Gain ratio
  - Missing values
  - Pruning
  - Rule extraction
- Limitations of decision trees

Advanced Decision Trees

- C4.5 - improvements from ID3
  - Handling both continuous and discrete attributes
  - Handling training data with missing attribute values
  - Handling attributes with differing costs
  - Pruning trees after creation
- C5 - Quinlan made further improvements (boosting)
  - Many commercial data mining packages use the C5 algorithm
- CART (Classification And Regression Trees, Breiman et al. 1984)
  - Similar to C4.5, boosting & bagging the data
  - Multivariate: tests linear combinations of variables

Decision Tree Algorithms – C4.5

- ***Recursive building*** tree phase:

```
1. Initialize root node of tree.
2. while a node N that can be split:
3.     for each attribute A, evaluate splits on A,
4.     use best split to split N.
```

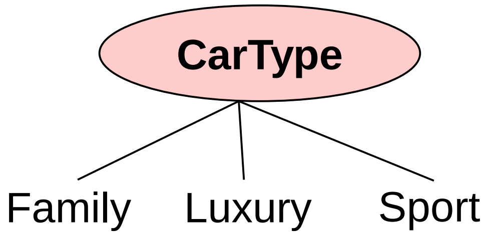
- Use ***entropy (information gain)*** to find best split

- Separate attribute lists maintained in each node of tree

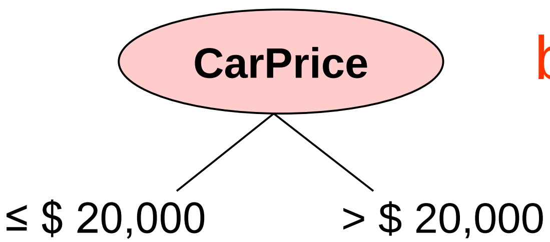
Overview

- Advanced decision trees
  - Continuous attributes
  - Gain ratio
  - Missing values
  - Pruning
  - Rule extraction
- Limitations of decision trees

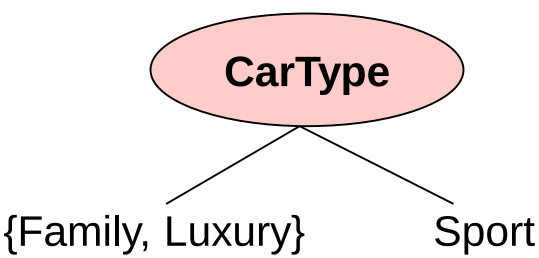
C4.5 – Possible Mechanisms for Tests



a. “standard” test on a ***discrete attribute***: one branch for each possible value of that attribute



b. If attribute Y has ***continuous numeric values***, binary test with outcomes $Y \leq Z$ and $Y > Z$ could be defined



c. possible values are allocated to various numbers of ***groups*** with one outcome/branch for each group

Continuous-valued Attributes – No Problem!

- Must determine the ***best split point*** for a continuous attribute
- Define ***binary test*** with outcomes $X \leq Z$ and $X > Z$, based on comparing the value of attribute against a ***threshold value*** Z
- ***Sort*** the training samples w.r.t. the values of the chosen attribute X
  - Number of these values is finite
  - Notation for sorted order: $\{v_1, v_2, \ldots, v_m\}$
- Examine ***all*** m-1 possible ***splits*** on X
  - Any threshold value between $\boldsymbol{v}_{\mathbf{i}}$ and $\boldsymbol{v}_{\mathbf{i+1}}$ has the same effect of dividing the cases into D1= $\{v_1, v_2, \ldots, v_i\}$ and D2=$\{v_{i+1}, v_{i+2}, \ldots, v_m\}$.
  - Representative threshold: ***midpoint*** of each interval: $(v_i + v_{i+1})/2$
  - C4.5 chooses, instead, the ***smaller*** value v_i of an interval $\{v_i, v_{i+1}\}$ – ensures that threshold values exist in the data
- Select ***optimal split***, i.e. with ***largest gain*** ratio

Example (1) Threshold Finding with Gain

Sometimes we have to find the threshold and the attribute

Database D

Attribute 1	Attribute 2	Attribute 3	Class
A	70	True	Class**1**
A	90	True	Class**2**
A	85	False	Class**2**
A	95	False	Class**2**
A	70	False	Class**1**
B	90	True	Class**1**
B	78	False	Class**1**
B	65	True	Class**1**
B	75	False	Class**1**
C	80	True	Class**2**
C	70	True	Class**2**
C	80	False	Class**1**
C	80	False	Class**1**
C	96	False	Class**1**

Attribute 2:

- After a sorting process, the set of values is: {65, 70, 75, 78, 80, 85, 90, 95, 96},
- … the set of potential threshold values *Z* is: {65, 70, 75, 78, 80, 85, 90, 95}.
- The optimal *Z* value is *Z*=80 (highest Inf. Gain)

9 are ≤ 80 … *7 are* Class**1** *2 are* Class**2**

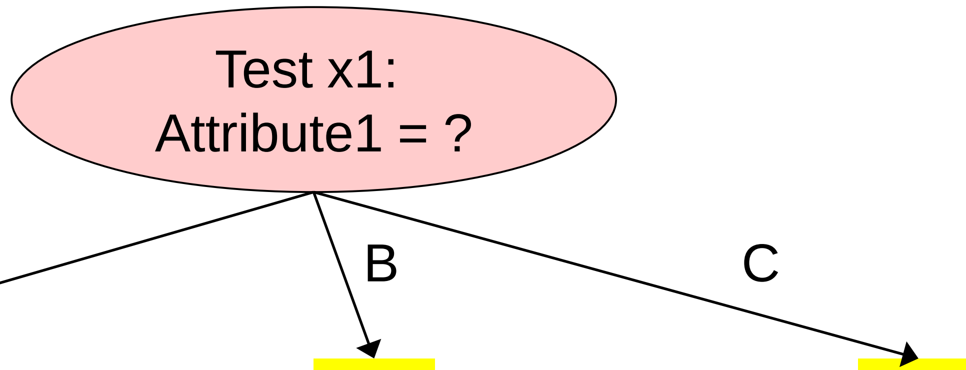
5 are > 80

- $\text{Info}_{Z=80}(D) = 9/14 \cdot (-\,7/9 \cdot \log_2(7/9) - 2/9 \cdot \log_2(2/9))$
  $+\ 5/14 \cdot (-\,2/5 \cdot \log_2(2/5) - 3/5 \cdot \log_2(3/5))$
  $=\ 0.837$ bits
- Gain(*Z=80*) = 0.940 – 0.837 = 0.103 bits

However, Attribute 1 gives the highest gain of 0.246 bits → this will be selected for first split

(are attributes with many values favored?)

Example (2) Initial Decision Tree



D1

Att. 2	Att. 3	Class
70	True	Class**1**
90	True	Class**2**
85	False	Class**2**
95	False	Class**2**
70	False	Class**1**

D2

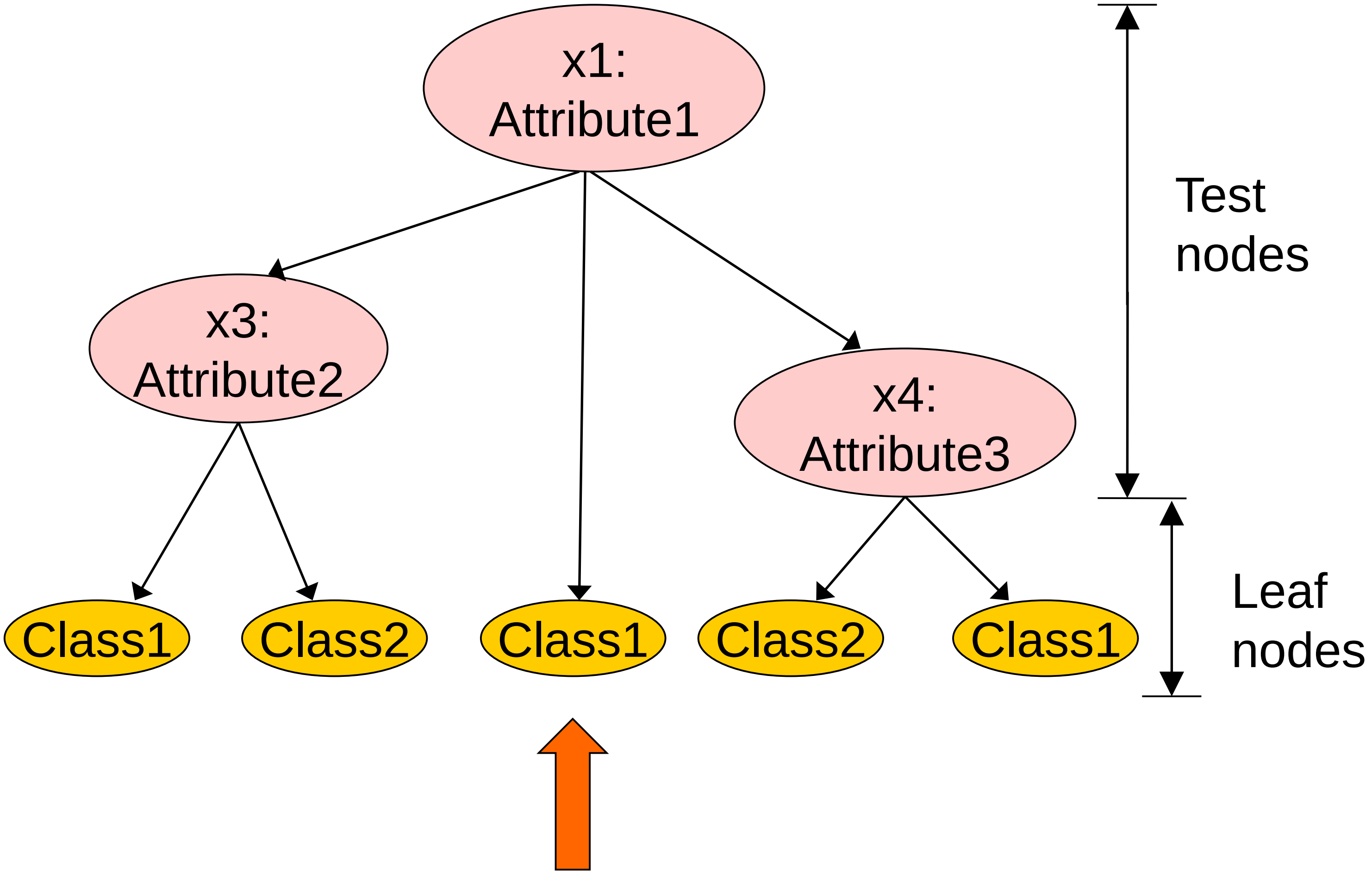
Att. 2	Att. 3	Class
90	True	Class**1**
78	False	Class**1**
65	True	Class**1**
75	False	Class**1**

D3

Att. 2	Att. 3	Class
80	True	Class**2**
70	True	Class**2**
80	False	Class**1**
80	False	Class**1**
96	False	Class**1**

Initial decision tree and subset cases
for a database ***D***

Example (3) Final Decision Tree



All of them are in CLASS1

Final Decision Tree as Pseudo Code

- Decision Tree – **Pseudo-code Example:**

```
If      Attribute1 = A
        Then
                If      Attribute2 <= 70
                        Then
                                Classification = CLASS1;
                        Else
                                Classification = CLASS2;
Elseif          Attribute1 = B
        Then
                                Classification = CLASS1;
Elseif          Attribute1 = C
        Then
                If      Attribute3 = True
                        Then
                                Classification = CLASS2;
        Else
                                Classification = CLASS1.
```

Overview

- Advanced decision trees
  - Continuous attributes
  - Gain ratio
  - Missing values
  - Pruning
  - Rule extraction
- Limitations of decision trees

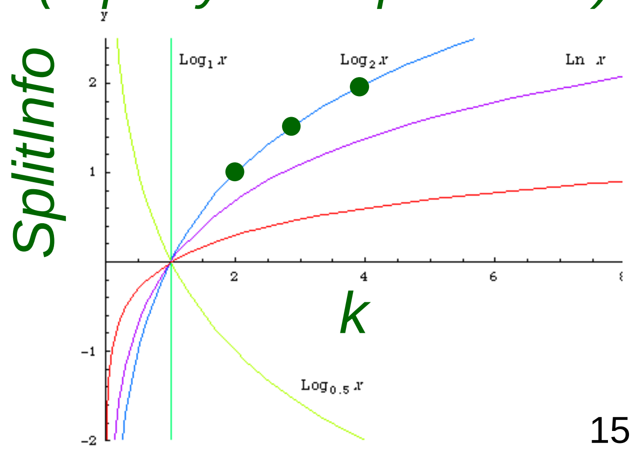
C4.5 Algorithm: Gain Ratio

- *Revision*: Measures we defined so far:
  - Entropy to classify a tuple in *D*: $$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$
  - Information needed (after using *A* to split *D* into *k* partitions) to classify *D*: $$Info_A(D) = \sum_{j=1}^{k} \frac{|D_j|}{|D|} \cdot Info(D_j)$$
  - Information gained for attribute *A*: $$Gain(A) = Info(D) - Info_A(D)$$
- Information gain (also: Gini impurity) is ***biased*** towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to ***normalize the information gain***

$$SplitInfo = -\sum_{j=1}^{k} \left(\frac{|D_j|}{|D|} \cdot \log_2 \left(\frac{|D_j|}{|D|} \right) \right)$$

$$GainRatio(A) = Gain(A) / SplitInfo(A)$$

(equally sized partitions)



Information Gain → Gain Ratio (prev. Example)

- Class "buys_computer =yes" (9x)
- Class "buys_computer =no" (5x)

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.94$$

$$Info_{age}(D) = \boxed{\frac{5}{14}I(2,3)} + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

age	yes_i	no_i	$I(yes_i, no_i)$
<=30	2	3	0,971
31…40	4	0	0
>40	3	2	0,971

"age <=30" has 5 out of 14 samples, with 2 "yes" and 3 "no"

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

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

$$SplitInfo(age) = -\sum_{j=1}^{3}\left(\frac{|D_j|}{|D|}\cdot\log_2\left(\frac{|D_j|}{|D|}\right)\right)$$

$$= -\frac{5}{14}\log_2(\frac{5}{14}) - \frac{4}{14}\log_2(\frac{4}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 1.577$$

$$GainRatio(age) = 0.246 / 1.557 = 0.156$$

Overview

- Advanced decision trees
  - Continuous attributes
  - Gain ratio
  - Missing values
  - Pruning
  - Rule extraction
- Limitations of decision trees

C4.5 Algorithm: Unknown Values

- New information gain criterion for split in attribute X:

  $$Gain(X) = F \cdot (Info(D) - Info_X(D))$$

  - *Factor F* = number of samples in database with known value for a given attribute X / total number of samples in a data set
  - *Factor F* here 13/14

Attribute 1	Attribute 2	Attribute 3	Class
A	70	True	Class**1**
A	90	True	Class**2**
A	85	False	Class**2**
A	95	False	Class**2**
A	70	False	Class**1**
?	90	True	Class**1**
B	78	False	Class**1**
B	65	True	Class**1**
B	75	False	Class**1**
C	80	True	Class**2**
C	70	True	Class**2**
C	80	False	Class**1**
C	80	False	Class**1**
C	96	False	Class**1**

C4.5 Algorithm: Unknown Values – Example (1)

13 remaining cases with values for Attribute1

$$\text{Info}(D) = -8/13\ \log_2 (8/13)\ - 5/13\ \log_2 (5/13) = \mathbf{0.961\ bits}$$

8 belong to CLASS1 *5 belong to CLASS2*

Test X_1 for the three values A, B, or C:

$$\text{Info}_{X1}(D) = \ 5/13\ (-2/5 \log_2 (2/5) – 3/5 \log_2 (3/5))$$
$$+ 3/13\ (-3/3 \log_2 (3/3) – \ 0/3 \log_2 (0/3))$$
$$+ 5/13\ (-3/5 \log_2 (3/5) –2/5 \log_2 (2/5))$$
$$= \ \mathbf{0.747\ bits}$$

$$\mathbf{Gain\ (X_1) = 13/14} \cdot (0.961 – 0.747) = \mathbf{0.199\ bits}$$

Factor F

Attribute 1	Attribute 2	Attribute 3	Class
A	70	True	Class**1**
A	90	True	Class**2**
A	85	False	Class**2**
A	95	False	Class**2**
A	70	False	Class**1**
~~?~~	~~90~~	~~True~~	~~Class**1**~~
B	78	False	Class**1**
B	65	True	Class**1**
B	75	False	Class**1**
C	80	True	Class**2**
C	70	True	Class**2**
C	80	False	Class**1**
C	80	False	Class**1**
C	96	False	Class**1**

C4.5 Algorithm: Unknown Values – Example (2)

Attribute 1	Attribute 2	Attribute 3	Class
A	70	True	Class**1**
A	90	True	Class**2**
A	85	False	Class**2**
A	95	False	Class**2**
A	70	False	Class**1**
?	90	True	Class**1**
B	78	False	Class**1**
B	65	True	Class**1**
B	75	False	Class**1**
C	80	True	Class**2**
C	70	True	Class**2**
C	80	False	Class**1**
C	80	False	Class**1**
C	96	False	Class**1**

- C4.5 assumes that samples with unknown values are ***distributed proportionally*** according to the ***relative frequency*** of known values

Distribution of samples into subsets with corresponding weight factors *w*

D1: Attribute1 = A

Att.2	Att.3	Class	*w*
70	True	Class**1**	1
90	True	Class**2**	1
85	False	Class**2**	1
95	False	Class**2**	1
70	False	Class**1**	1
90	***True***	***Class1***	***5/13***

D2: Attribute1 = B

Att.2	Att.3	Class	*w*
90	***True***	***Class1***	***3/13***
78	False	Class**1**	1
65	True	Class**1**	1
75	False	Class**1**	1

D3: Attribute1 = C

Att.2	Att.3	Class	*w*
80	True	Class**2**	1
70	True	Class**2**	1
80	False	Class**1**	1
80	False	Class**1**	1
96	False	Class**1**	1
90	***True***	***Class1***	***5/13***

C4.5 Algorithm: Generalizing Partitioning

- When a sample from D with **known value** is assigned to subset D_i, its probability belonging to ***D_i* is 1**, and in all other subsets is 0
- C4.5 associates with each sample (having ***missing value***) a ***weight w*** representing the ***probability*** that it belongs to each subset D_i :

  $$w_{\text{new}} = w_{\text{old}} \cdot P(D_i)$$

- Splitting set D using test X_1 on Attribute1: New weights w_i will be probabilities, here: 5/13, 3/13, and 5/13, since initial w_{old} is 1

  $|D_1|$ = 5+**5/13**, $|D_2|$ = 3+**3/13**, and $|D_3|$ = 5+**5/13**

- The decision tree **leaves** are defined with two new parameters: **($|D_i|$/*E*)**
- $|D_i|$ is the sum of the ***fractional samples*** that reach the leaf, and

  E is the ***number of samples*** belonging to classes other than nominated class
- (3.4 / 0.4) means:
  - 3.4 (or 3 + 5/13) fractional training samples reached leaf,
  - 0.4 (or 5/13) of which did not belong to the class of the leaf

Partitioning – Example

- Decision tree for the database *D* with missing values:

If **Attribute1 == A**
Then
If **Attribute2 <= 70**
Then
Classification = CLASS1 (2.0 / 0);
Else
Classification = CLASS2 (3.4 / 0.4);
Elseif **Attribute1 == B**
Then
Classification = CLASS1 (3.2 / 0);
Elseif **Attribute1 == C**
Then
If **Attribute3 = True**
Then
Classification = CLASS2 (2.4 / 0.4);
Else
Classification = CLASS1 (3.0 / 0).

($|Di|$/E):
$|Di|$ = sum of the fractional samples that reach the leaf,
E = number of samples that belong to classes other than the nominated class.

Overview

- Advanced decision trees
  - Continuous attributes
  - Gain ratio
  - Missing values
  - Pruning
  - Rule extraction
- Limitations of decision trees

Decision Tree Algorithms – Building and Pruning

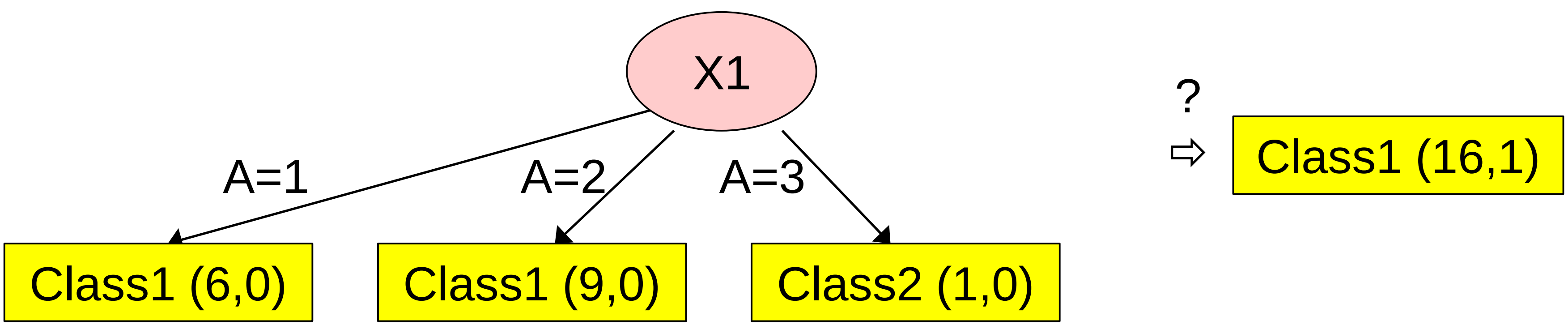
- ***Building phase***
  - Recursively split nodes using best splitting attribute for node.
- ***Pruning phase***
  - Smaller imperfect decision tree generally achieves better accuracy on test data.
  - Prune leaf nodes recursively to prevent over-fitting.

Avoid Overfitting in Classification

- The generated tree may overfit the training data:
  - Too ***many branches***, some may reflect anomalies due to noise or outliers
  - Result: poor accuracy for unseen samples
- Two approaches to avoid overfitting:
  - ***Prepruning***: Halt tree construction early—do not split a node if the goodness measure would then fall below a threshold
    - Difficult to choose an appropriate threshold
  - ***Postpruning***: Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a ***set of data different from the training data*** to decide which is the "best pruned tree"

Pruning a Decision Tree

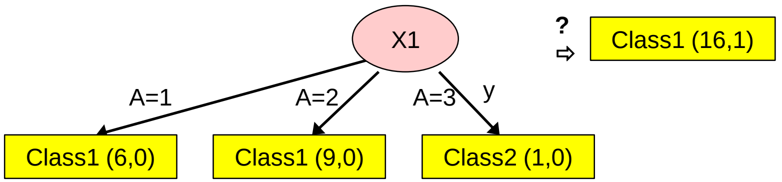
- ***Pruning***: Discarding one or more subtrees and replacing them with leaves
  - C4.5 follows a ***postpruning*** approach (pessimistic pruning)



Shall we replace this subtree with a single leaf node?

Pruning Decision Tree: **P**redicted **E**rror

$$PE = \sum_{i=1}^{nodes} n_i \cdot U_{25\%}$$



\# of samples in the node

upper limit on error rate (for the node):
from statistical tables for binomial distributions

- Using default confidence of 25%, **upper limits on the error rates** for all nodes are collected from statistical tables for binomial distributions:

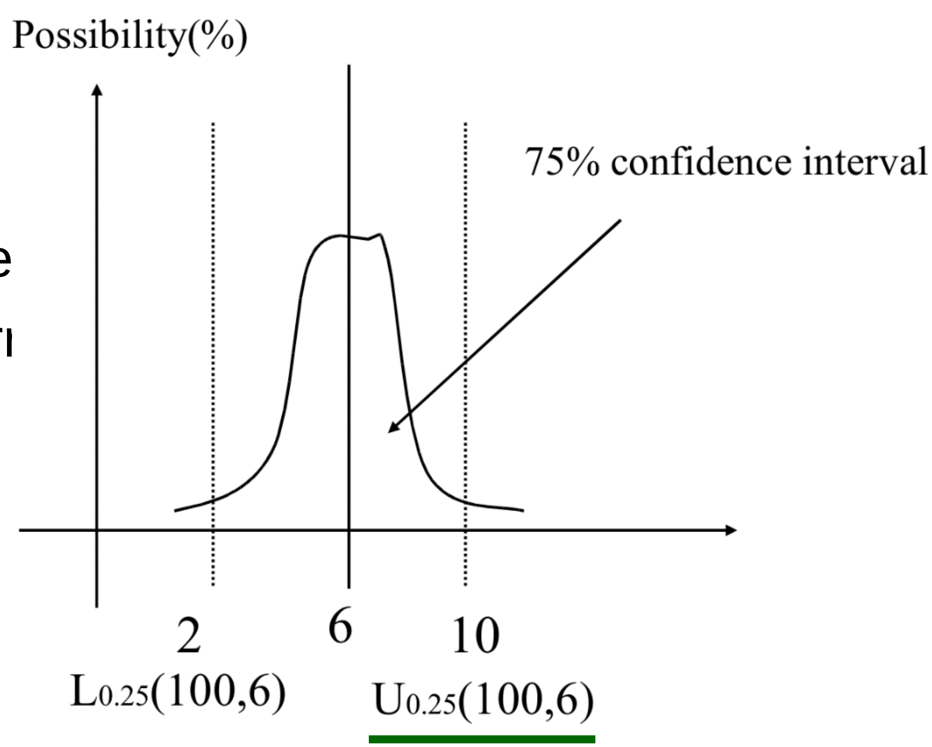
  Tree: $U_{25\%}(6,0) = 0.206$, $U_{25\%}(9,0) = 0.143$, $U_{25\%}(1,0) = 0.750$

  Node: $U_{25\%}(16,1) = 0.157$

- **Predicted errors** for the subtree and the replaced node are:
  - $PE_{tree} = 6 \cdot 0.206 + 9 \cdot 0.143 + 1 \cdot 0.750 = 3.257$
  - $PE_{node} = 16 \cdot 0.157 = 2.512$
  - Since $PE_{tree} > PE_{node}$, replace the subtree with the new leaf node.

$U_{CF}(|D_i|, E)$

- Consider classifying E examples incorrectly out of $|D_i|$ samples (like observing E events in $|D_i|$ trials in the binomial distribution)
- For a given confidence level CF, the upper limit on the error rate over the whole population is $U_{CF}(|D_i|, E)$ with CF% confidence.
- Example:
  - $U_{25\%}(100,6)$
  - 100 examples in a leaf
  - 6 examples misclassifie
  - How large is the true erı assuming a pessimistic estimate with a confidence of 25%?



Overview

- Advanced decision trees
  - Continuous attributes
  - Gain ratio
  - Missing values
  - Pruning
  - Rule extraction
- Limitations of decision trees

Extracting Decision Rules from Trees

- Rules are easier for humans to understand
- Represent the knowledge in the form of ***IF-THEN*** rules
  - One rule is created for each path from the root to a leaf.
  - Each attribute-value pair along a path forms a conjunction.
  - The leaf node holds the class prediction.

Examples:

```
IF age = "<=30" AND student = "no"
   THEN buys_computer = "no"
IF age = "<=30" AND student = "yes"
   THEN buys_computer = "yes"
IF age = "31…40"
   THEN buys_computer = "yes"
IF age = ">40" AND credit_rating = "excellent"
   THEN buys_computer = "yes"
IF age = ">40" AND credit_rating = "fair"
   THEN buys_computer = "no"
```

Rule Ordering (I/II)

More than one rule may by triggered:

- Order of presentation to expert to be determined
- Different decision trees may be considered
- Missing attributes allow different paths from the root node to a leaf node

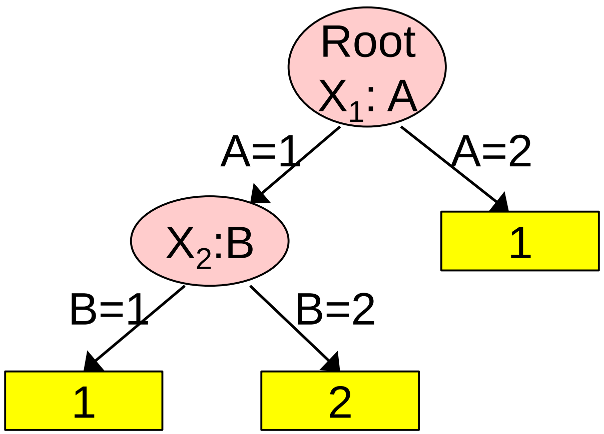
Rule Ordering (II/II)

Considering more than one rule, we need ***conflict resolution***

- Size ordering
  - assign the highest priority to the triggering rule that has the "toughest" requirement (i.e., with the *most attribute tests*)
- Class-based ordering
  - Rules for the most frequent class come first, or
  - Sort based on misclassification cost per class
- Rule-based ordering (decision list)
  - rules are organized into one long priority list, according to some measure of rule quality (e.g. accuracy, # attribute tests) or by experts

C4.5 Algorithm: Generating Decision Rules may not really simplify

Decision tree



Transformation

Paths into Rules

Decision rules

If A=1 and B=1 Then Class1
If A=1 and B=2 Then Class2
If A=2 Then Class1

Decision rules for database **D:**

Attribute 1	Attribute 2	Attribute 3	Class
A	70	True	Class**1**
A	90	True	Class**2**
A	85	False	Class**2**
A	95	False	Class**2**
A	70	False	Class**1**
?	90	True	Class**1**
B	78	False	Class**1**
B	65	True	Class**1**
B	75	False	Class**1**
C	80	True	Class**2**
C	70	True	Class**2**
C	80	False	Class**1**
C	80	False	Class**1**
C	96	False	Class**1**

If Attribute1 = A and Attribute2 <= 70
Then Classification = CLASS1 (2.0 / 0);

If Attribute1 = A and Attribute2 > 70
Then Classification = CLASS2 (3.4 / 0.4);

If Attribute1 = B
Then Classification = CLASS1 (3.2 / 0);

If Attribute1 = C and Attribute3 = True
Then Classification = CLASS2 (2.4 / 0.4);

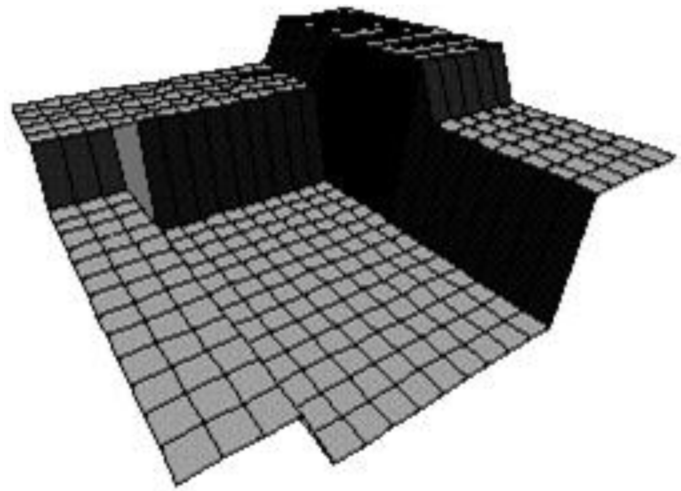
If Attribute1 = C and Attribute3 = False
Then Classification = CLASS1 (3.0 / 0).

Bottom example is for previous partitioning data set (14 samples).

Overview

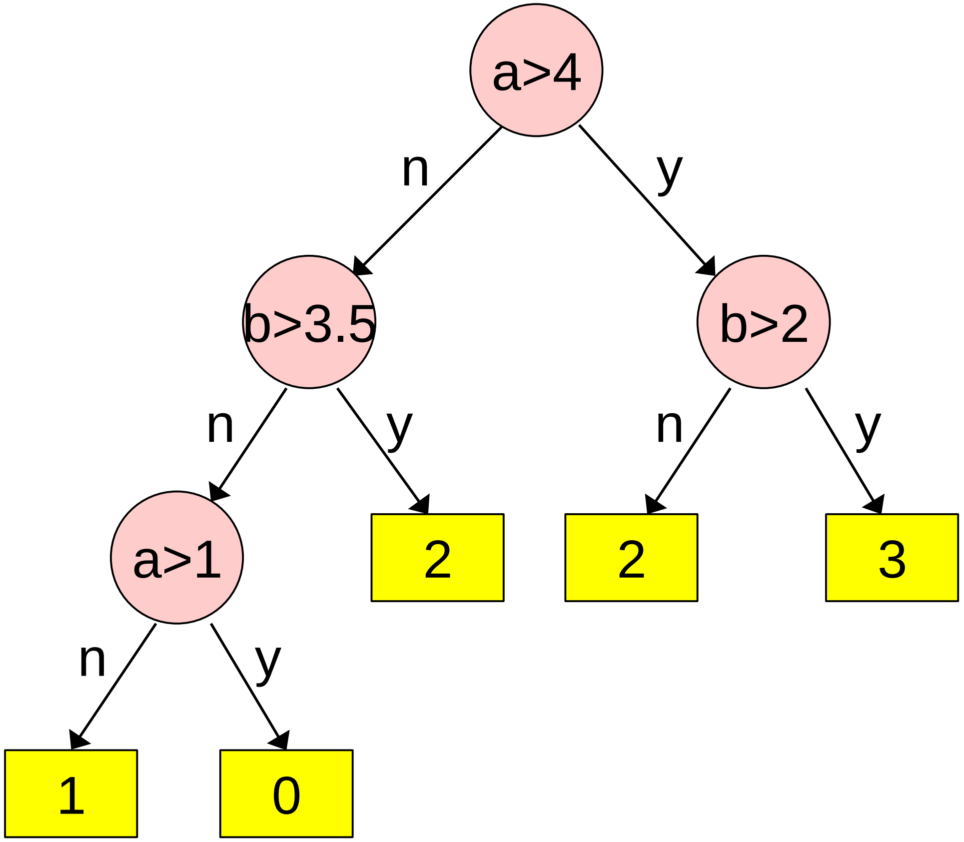
- Advanced decision trees
  - Continuous attributes
  - Gain ratio
  - Missing values
  - Pruning
  - Rule extraction
- Limitations of decision trees

Limitations of Decision Trees and Decision Rules (1)

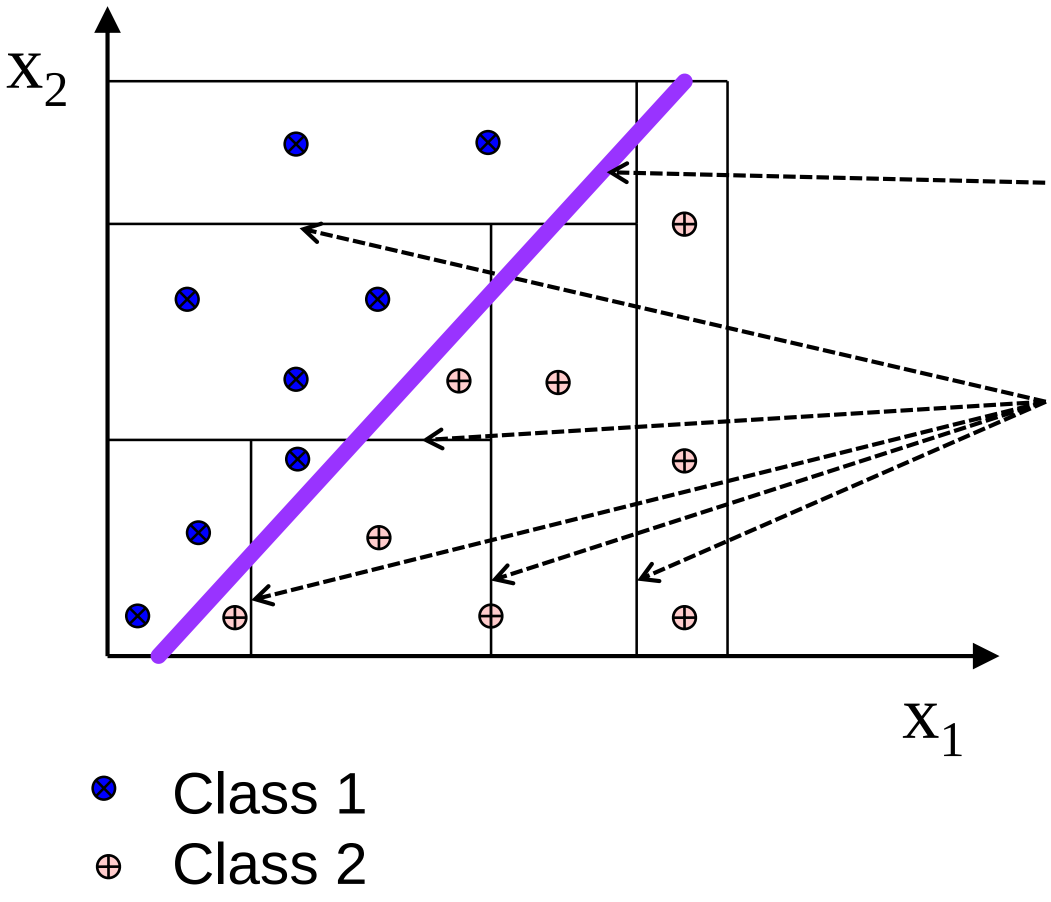


Example:

- 2D samples are classified using a third dimension for classes
- Problematic: classification function is much ***more complex*** with ***related attributes***



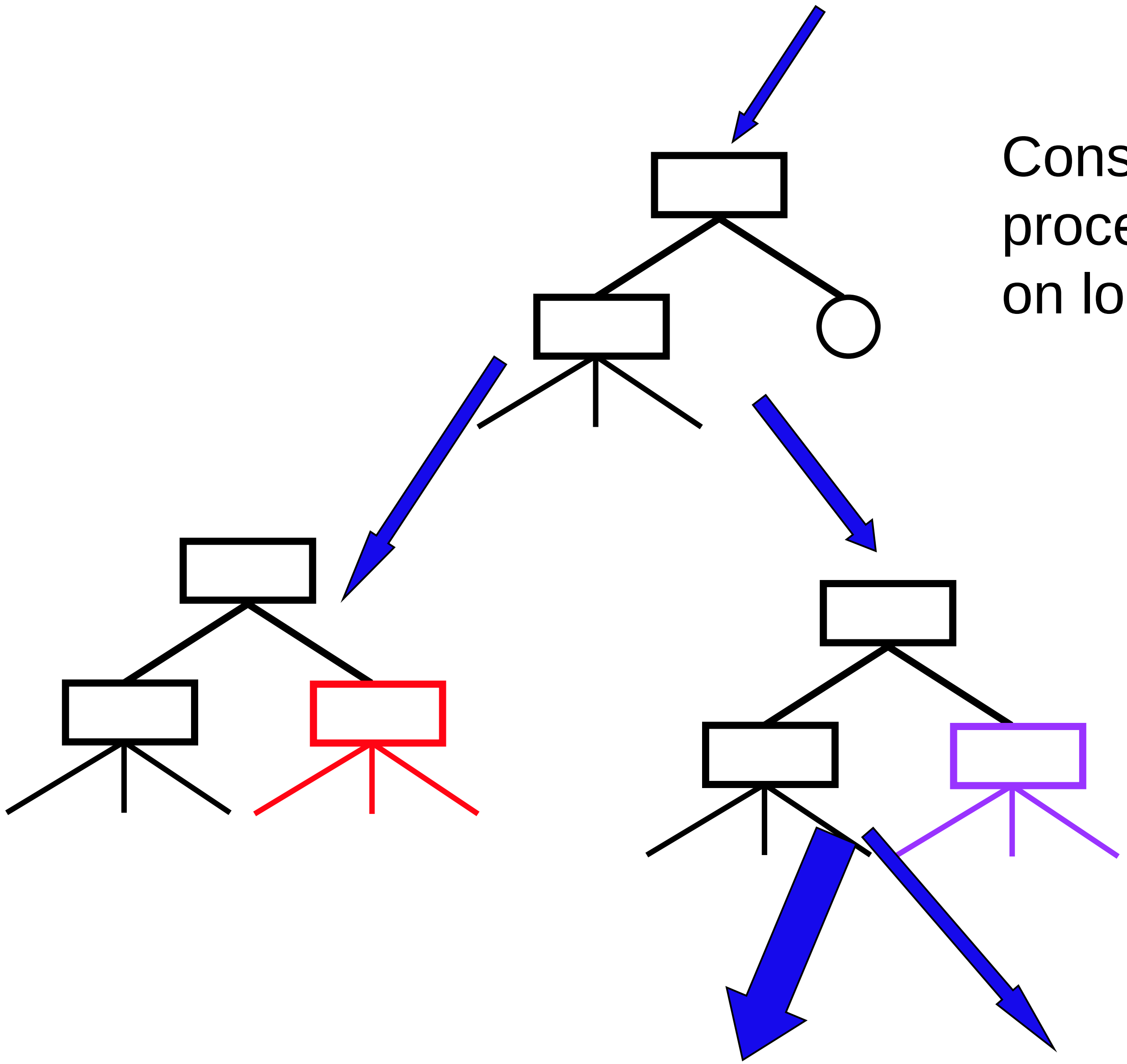
Limitations of Decision Trees and Decision Rules (2)



Classification through a linear combination of attributes

Rules as orthogonal hyperplanes

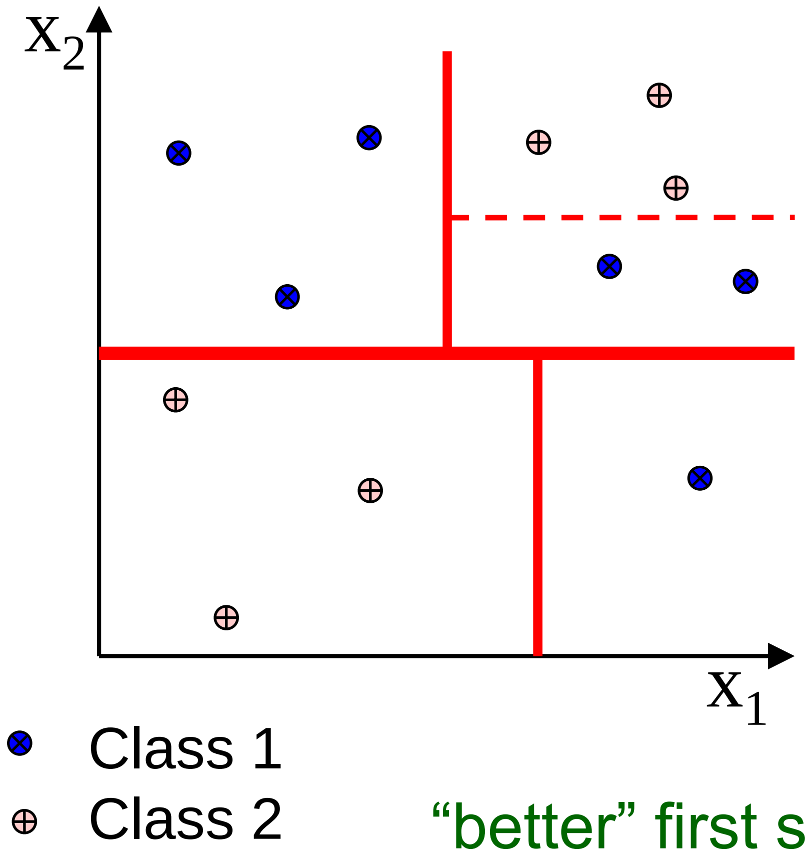
Limitations of Decision Trees and Decision Rules (3)

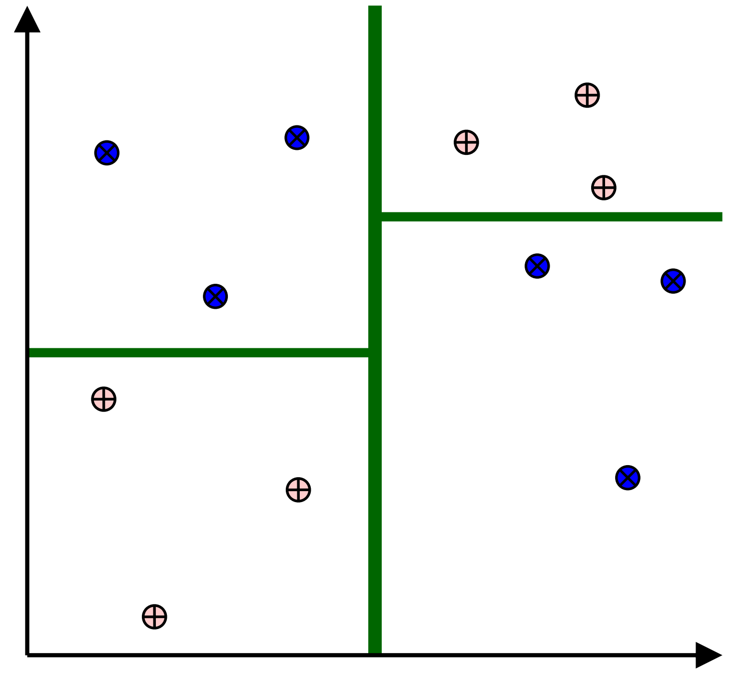


Construction / search process depends only on local information

Limitations of Decision Trees and Decision Rules (3)

- ***Greedy: current best*** split does not consider future splits





"better" first split (global view) not found by information gain

Limitations of Decision Trees and Decision Rules (4)

- Let a given class be supported, if ***any k out of n*** conditions are met.
- To represent this classifier with rules, it would be necessary to define $\binom{n}{k}$ regions only for one class

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}.$$

- **Example**: Medical diagnostic:
  - If 4 out of 11 symptoms support diagnosis of a given disease, then the corresponding classifier will generate 330 regions in 11-dimensional space for positive diagnosis only.

⇨ corresponds to 330 decision rules.

Limitations of Decision Trees and Decision Rules: Further Ideas

- Introducing new attributes, rather than removing old ones, can avoid sometimes-intensive fragmentation of the n-dimensional space:

Model: $(A1 \vee A2 \vee A3) \wedge (A4 \vee A5 \vee A6) \wedge (A7 \vee A8 \vee A9) \rightarrow$ **C1**

Solution 1:

$A1 \wedge A4 \wedge A7 \rightarrow C1$

$A1 \wedge A5 \wedge A7 \rightarrow C1$

$A1 \wedge A6 \wedge A7 \rightarrow C1$

…

→ 27 combinations

Solution 2: Introduce **new derived attributes**:

$B1 = A1 \vee A2 \vee A3$

$B2 = A4 \vee A5 \vee A6$

$B3 = A7 \vee A8 \vee A9$

→ $\mathbf{B1 \wedge B2 \wedge B3 \rightarrow C1}$

Enhancements to Basic Decision Tree Induction (Summary I/II)

- Allow for ***continuous-valued attributes***
  - Partition a continuous attribute into a discrete set of intervals
- Handle ***missing attribute values***
  - Assign probability to each of the possible values
- Pruning
  - ***Avoid overfitting*** using separate pruning data set
- Challenges:
  - ***Attribute construction*** of new attributes based on existing ones that are sparsely represented
    - Reduces fragmentation, repetition, replication
  - ***Incremental learning*** of decision trees

Decision Trees (Summary II/II)

- Advantages
  - Automatically create tree representations from data
  - Trees can be converted to rules, can discover “new” rules
  - Identify most discriminating attribute first
    - Using Information Gain (Ratio) or Gini Impurity
  - Tree can handle discrete, continuous, mixed, and missing attributes
- Disadvantages
  - Trees can become large and difficult to understand
  - Can produce counter-intuitive rules
  - Examines attributes individually, but not inter-attribute relationships
  - Future splits not known when splitting
    → not globally optimal tree
  - Tree induction rules not directly related to training objective, i.e. minimizing classification error