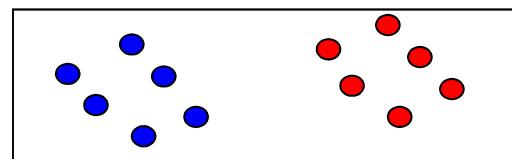
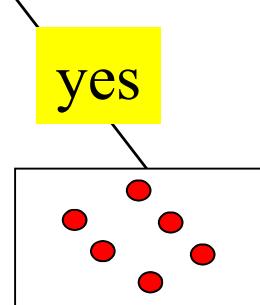
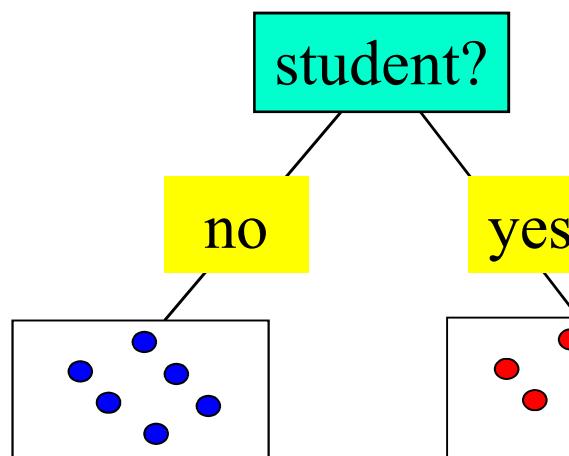
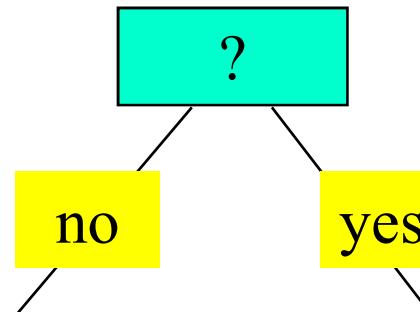


Test Attribute Selection

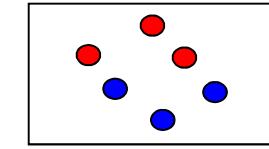
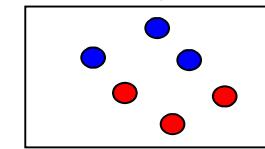
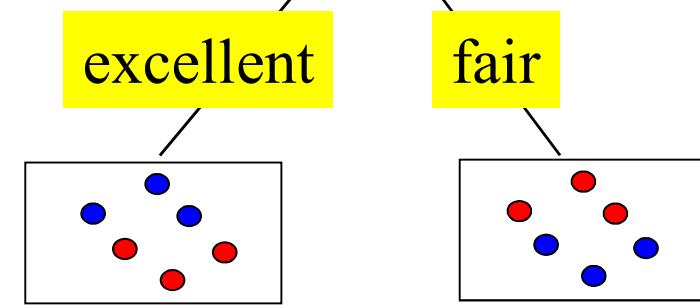
- Which is better as a test attribute?
 - Partitions a group into **more homogeneous** ones



Buy-computer: yes ●
Buy-computer: no ●

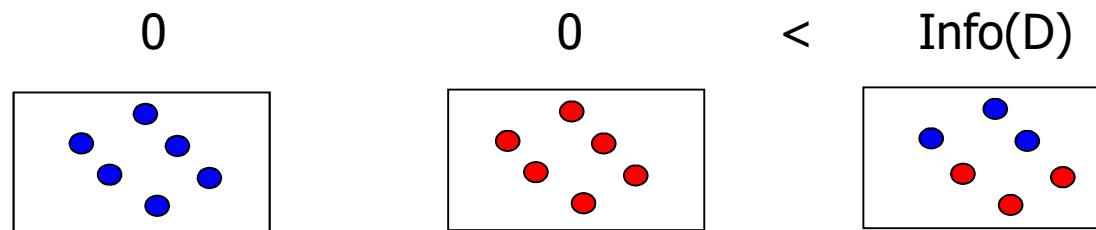


credit rating?



Attribute Selection Measure: Information Gain

- Select the test attribute having the **highest information gain**
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i , estimated by $|C_{i,D}|/|D|$
- **Entropy (expected information)** to classify a tuple in D:
 - $Info(D) = -\sum_{i=1}^m p_i \log_2(p_i)$
 - The more heterogeneous the higher

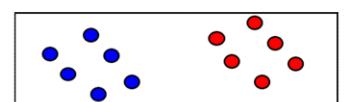


Attribute Selection Measure: Information Gain

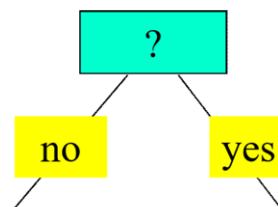
- Expected information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times Info(D_j)$$

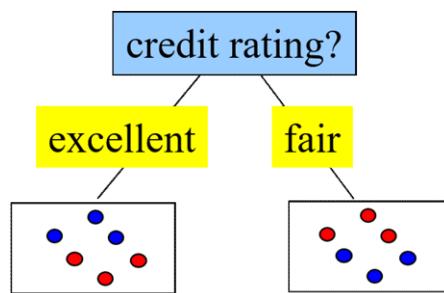
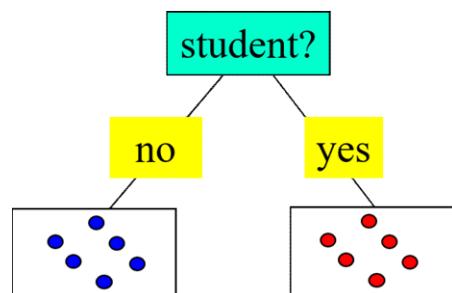
- Information gained by branching on attribute A



Buy-computer: yes •
Buy-computer: no •



$$Gain(A) = Info(D) - Info_A(D)$$



Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940$$

$$\begin{aligned} Info_{age}(D) &= \frac{5}{14} I(2,3) + \frac{4}{14} \\ &\quad + \frac{5}{14} I(3,2) = 0.694 \end{aligned}$$

age	p_i	n_i	$I(p_i, n_i)$
≤ 30	2	3	0.971
31...40	4	0	0
> 40	3	2	0.971

$\frac{5}{14} I(2,3)$ means "age ≤ 30 " has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

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

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

Similarly,

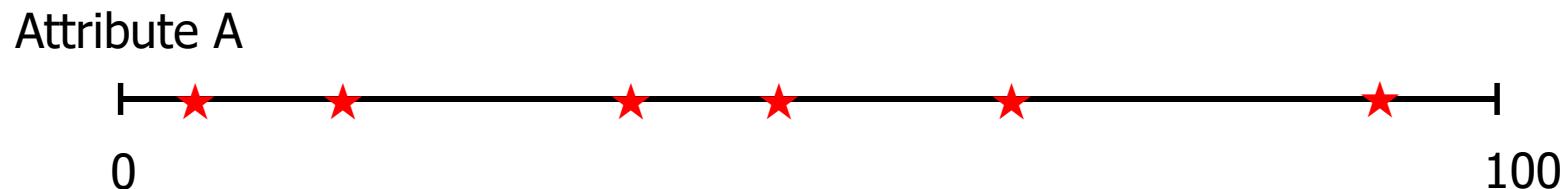
$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit_rating) = 0.048$$

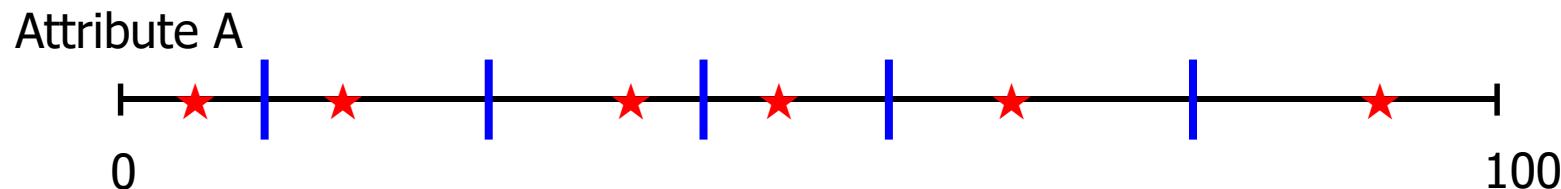
Computing Information-Gain for Continuous-Value Attributes

- Let attribute A be a continuous-valued attribute
- Make attribute A discrete by deciding the *split points* for A
 - Sort the existing values for A in increasing order
 - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
 - $(a_i + a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}



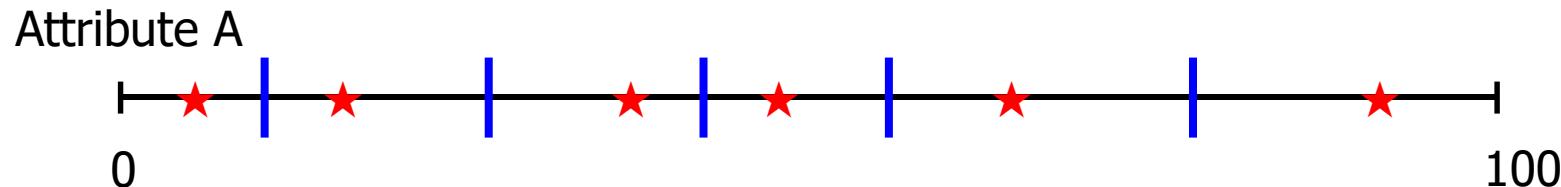
Computing Information-Gain for Continuous-Value Attributes

- Let attribute A be a continuous-valued attribute
- Make attribute A discrete by deciding the *split points* for A
 - Sort the existing values for A in increasing order
 - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
 - $(a_i + a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}



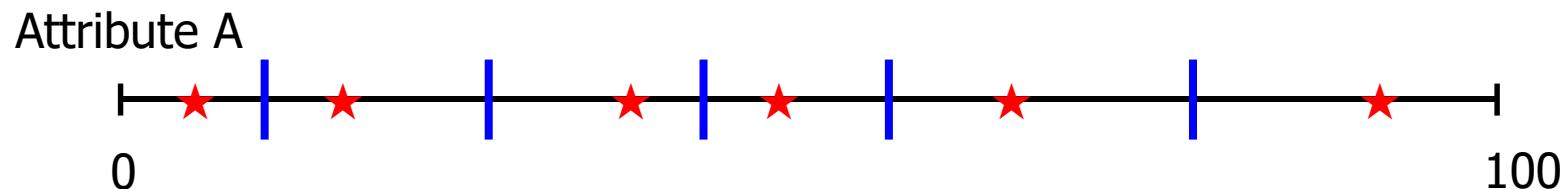
Computing Information-Gain for Continuous-Value Attributes

- Compute the entropy for each split point:
 - D_1 is the set of tuples in D satisfying $A \leq \text{split-point}$
 - D_2 is the set of tuples in D satisfying $A > \text{split-point}$
 - Determine the *best split point* for A
 - The point giving the *minimum entropy* (i.e., *maximum information gain*) for A is selected



Computing Information-Gain for Continuous-Value Attributes

- Question
 - While a binary partition is assumed currently, an **n-ary partition could be considered**
 - What is the problem with this case?



Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
 - If A has 5 values and B has 2 values,
 - A tends to have higher information gain than B
- C4.5 (a successor of ID3) uses **gain ratio** to overcome the problem (normalization to information gain)
 - $\text{GainRatio}(A) = \text{Gain}(A) / \text{SplitInfo}(A)$



Gain Ratio for Attribute Selection (C4.5)

- $\text{GainRatio}(A) = \text{Gain}(A) / \text{SplitInfo}_A(D)$

$$\text{SplitInfo}_A(D) = -\sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

- Example

- $\text{gain_ratio}(\text{income}) = 0.029/0.926 = 0.031$

$$\text{SplitInfo}_A(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 0.926$$

- The attribute with **the maximum gain ratio** is selected as the splitting attribute



Gini index (CART, IBM IntelligentMiner)

- If a data set D contains tuples from n classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^n p_j^2$$

where p_j is the relative probability of class j in D

- If a data set D is split on A into two subsets D_1 and D_2 , the gini index (impurity) $gini(D)$ is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- where D_1 is a set of tuples having some values for A while D_2 is a set of tuples having the other values for A



Gini index (CART, IBM IntelligentMiner)

- Reduction in impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute providing the largest reduction in impurity (i.e., the smallest $gini_A(D)$) is chosen to be the test attribute to split the node
 - *Binary partition: need to enumerate all the possible splitting points for each attribute*



Gini index (CART, IBM IntelligentMiner)

- Example: D has 9 tuples in buys_computer = "yes" and 5 tuples in buys_computer = "no"

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1 : {low, medium} and 4 in D_2 : {high}

$$\begin{aligned} gini_{income \in \{low, medium\}}(D) &= \left(\frac{10}{14}\right)Gini(D_1) + \left(\frac{4}{14}\right)Gini(D_1) \\ &= \frac{10}{14}(1 - (\frac{6}{10})^2 - (\frac{4}{10})^2) + \frac{4}{14}(1 - (\frac{1}{4})^2 - (\frac{3}{4})^2) \\ &= 0.450 \\ &= Gini_{income \in \{high\}}(D) \end{aligned}$$

- If $gini_{\{medium, high\}}$ is 0.30, it will be better since it is the lower



Comparing Attribute Selection Measures

- The three measures, in general, return good results but
 - Information gain:
 - biased towards multivalued attributes
 - Gain ratio:
 - tends to prefer unbalanced splits in which one partition is much smaller than the others
 - Gini index:
 - has difficulty when # of classes is large
 - tends to favor tests that result in equal-sized partitions and purity in both partitions



Overfitting

- Overfitting
 - An induced tree may overfit the training data
 - Extreme case
 - Every tuple has its own branch in the tree
- Problem
 - Too many branches, some may reflect anomalies due to noise or outliers
 - Poor accuracy for unseen samples



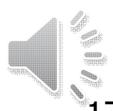
Overfitting and Tree Pruning

- Two approaches to avoid overfitting
 - Prepruning: Halt tree construction early
 - Do not split a node if this would result in the goodness measure falling below a threshold
 - Difficult to choose an appropriate threshold
 - Postpruning: Remove branches from a “fully grown” tree
 - Get a sequence of progressively pruned trees (see Fig. 6.6 in text)
 - Use a set of data different from the training data to decide which is the “best pruned tree”

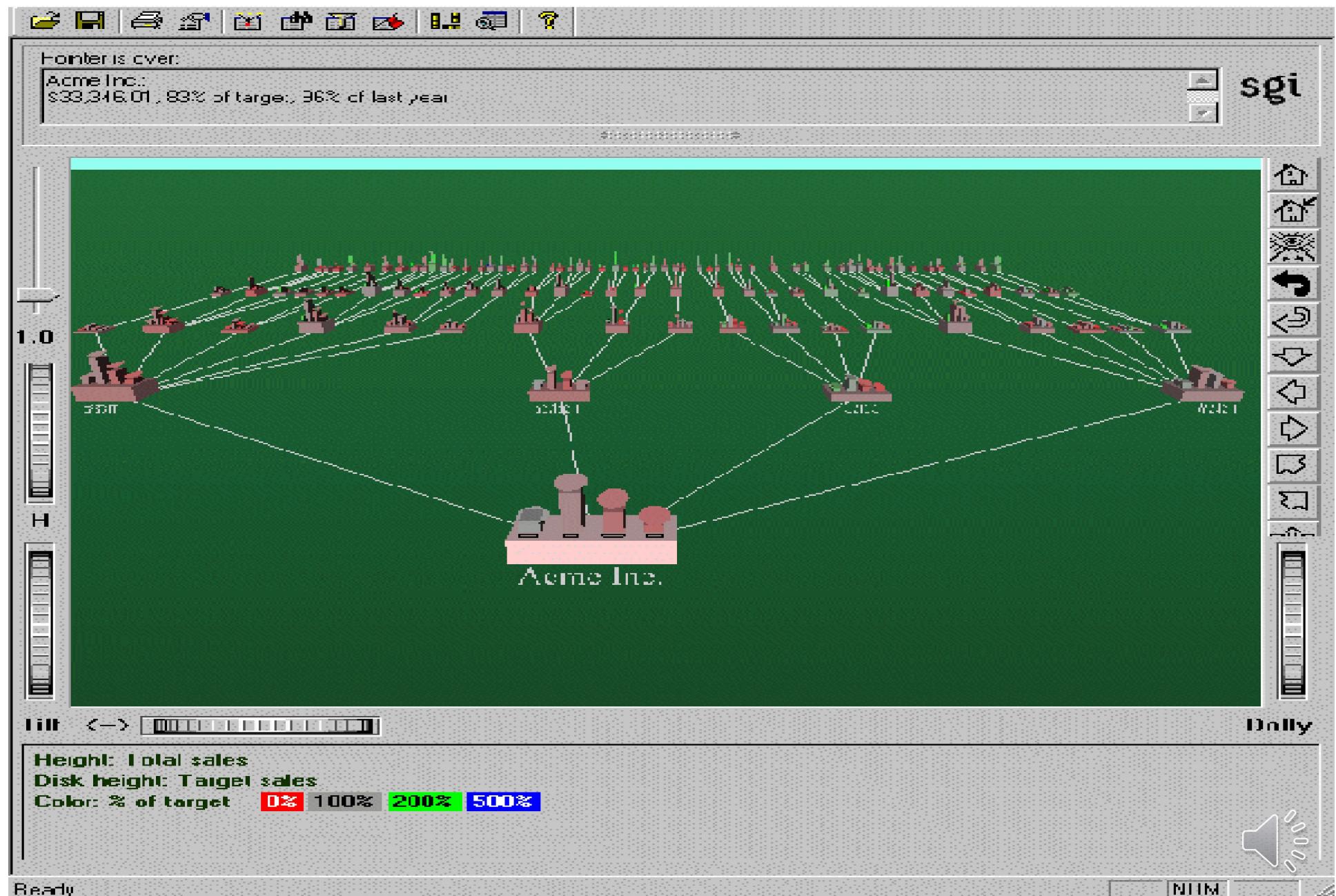


Classification in Large Databases

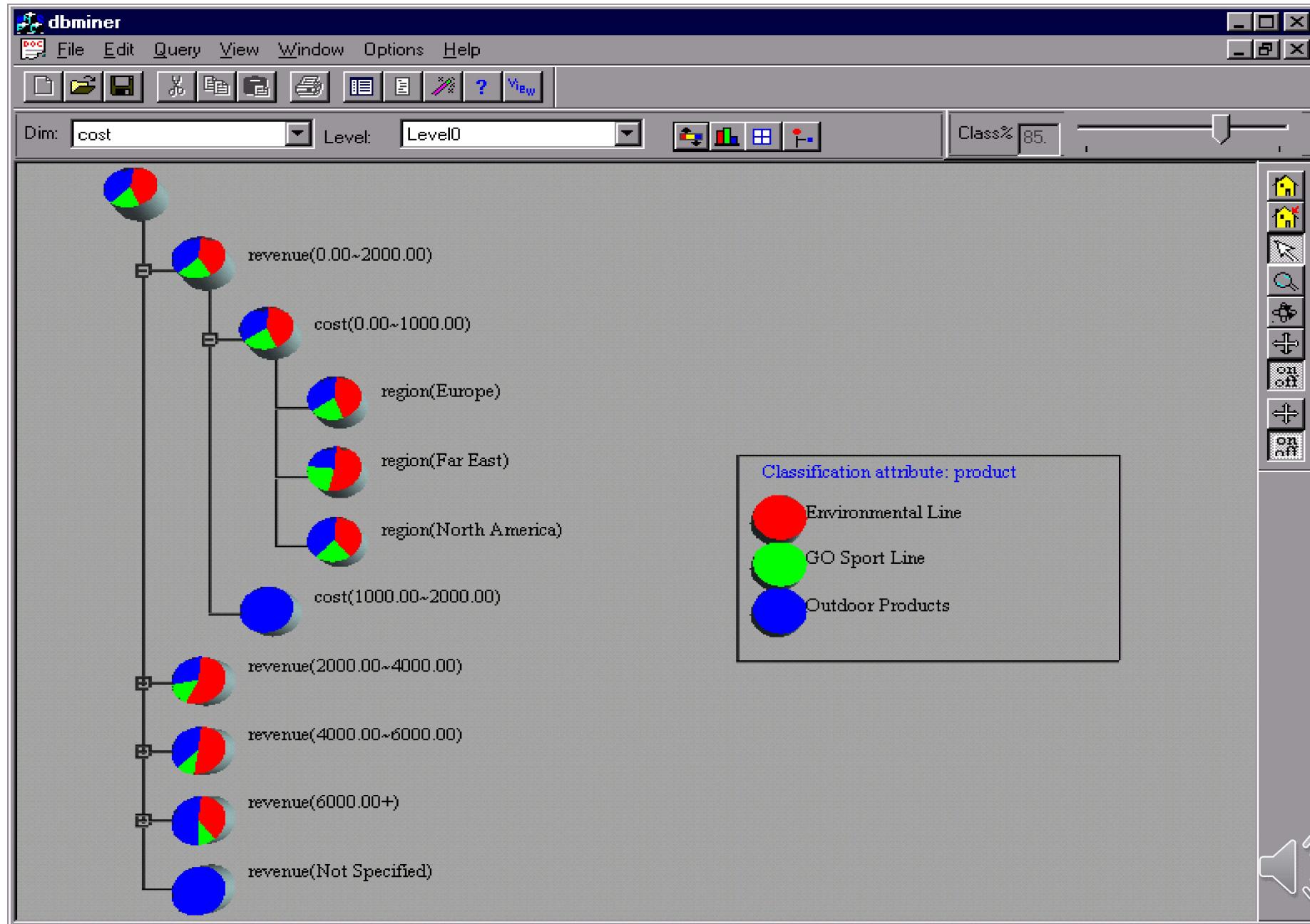
- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes **with reasonable speed**
- Why decision tree induction in data mining?
 - relatively faster learning speed (than other classification methods)
 - convertible to simple and easy to understand classification rules
 - can use SQL queries for accessing databases
 - comparable classification accuracy with other methods



Visualization of a Decision Tree in SGI/MineSet 3.0



Presentation of Classification Results



Chapter 6. Classification and Prediction

- What is classification? What is prediction?
 - Issues regarding classification and prediction
 - Classification by decision tree induction
 - Bayesian classification
 - Rule-based classification
 - Classification by back propagation
- 
- Support Vector Machines (SVM)
 - Associative classification
 - Lazy learners (or learning from your neighbors)
 - Other classification methods
 - Prediction
 - Accuracy and error measures
 - Ensemble methods
 - Model selection
 - Summary

Bayesian Classification: Why?

- A statistical classifier:
 - performs *probabilistic prediction*, i.e., predicts the membership probabilities for different classes
 - Foundation: Based on **Bayes' theorem**
- Performance:
 - A simple *naïve Bayesian classifier* has comparable performance with decision trees and neural network classifiers



Bayesian Classification: Why?

- Incremental:
 - Each training example can **incrementally increase/decrease the probability** that a hypothesis is correct
 - Prior knowledge can be combined with observed data, rather than training from the scratch
- Standard:
 - They can provide a standard of optimal decision making against which other methods can be measured



Bayesian Theorem: Basics

- Let \mathbf{X} be a data sample (*evidence*) whose class label is unknown
- Let H be a *hypothesis* that X belongs to a class C
- Classification
 - to determine $P(H|\mathbf{X})$, the probability that the hypothesis holds when the observed data sample \mathbf{X} is given



Bayesian Theorem: Basics

- $P(H)$ (*prior probability*)
 - The initial probability (independent of a specific \mathbf{X})
 - E.g., \mathbf{X} will buy computer, regardless of age, income, ...
- $P(\mathbf{X})$
 - The probability that sample data is observed
- $P(\mathbf{X}|H)$ (*posteriori probability*)
 - The probability of observing the sample \mathbf{X} , given that the hypothesis holds
 - E.g., Given that \mathbf{X} will buy computer, the prob. that \mathbf{X} is 31..40, medium income



Bayesian Theorem

- Conditional probability
 - $P(H|X) = P(H \cap X) / P(X)$
 - $P(X|H) = P(H \cap X) / P(H)$
 - $P(H \cap X) = P(H|X) * P(X) = P(X|H) * P(H)$
- Given training data \mathbf{X} , *posteriori probability of a hypothesis H*, $P(H|\mathbf{X})$, follows the Bayes theorem

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})}$$

