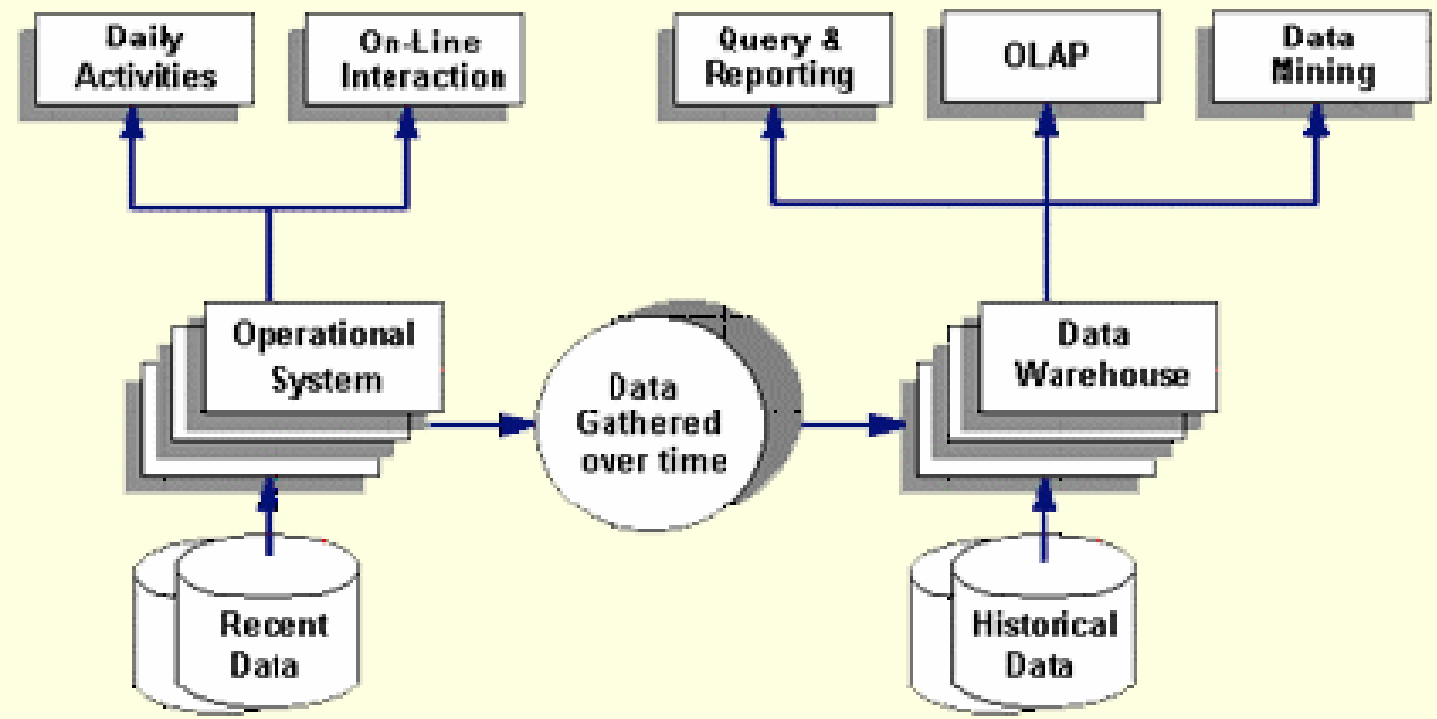
Decision Tree

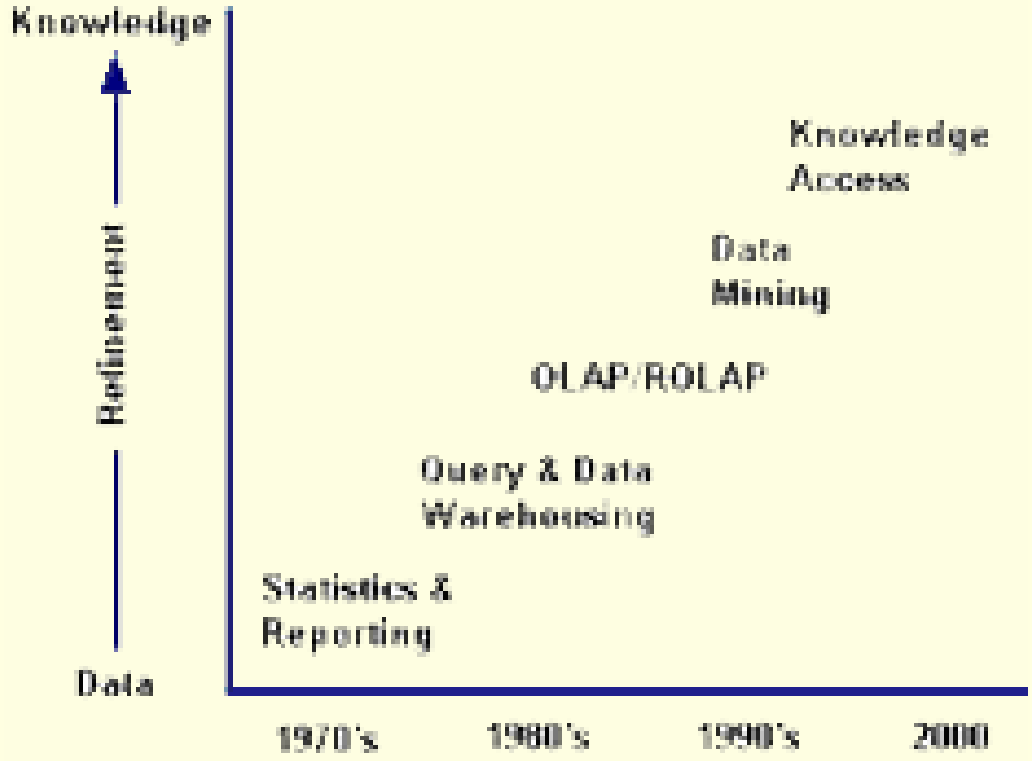
ID3, SLIQ, SPRINT.

~ Kaushik Malakar (9925219)

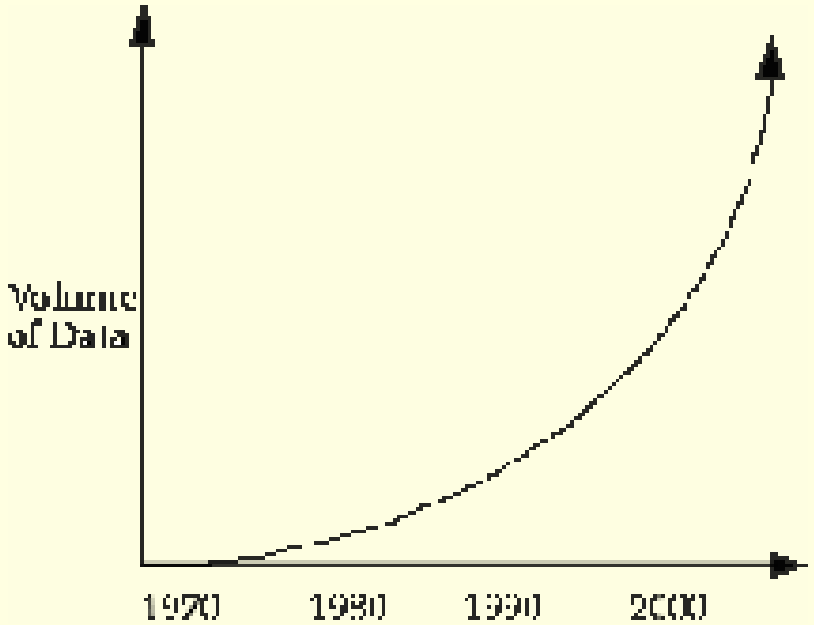
Data Mining


Daily Activities
On-Line Interaction
Query & Reporting
OLAP
Data Mining
Operational System
Data Gathered over time
Data Warehouse
Recent Data
Historical Data

Data Mining


Knowledge
Refinement
Data
Knowledge Access
Data Mining
OLAP/ROLAP
Query & Data Warehousing
Statistics & Reporting
1970's
1980's
1990's
2000

Data Mining



The Growing Base of Data

Data Mining

- ***Data mining*** *refers to "using a variety of techniques to identify nuggets of information or decision-making knowledge in bodies of data, and extracting these in such a way that they can be put to use. The data is often voluminous, but as it stands of low value as no direct use can be made of it; it is the hidden information in the data that is useful".*

- **Stages/Processes identified:**
  - Selection
  - Transformation
  - Interpretation and evaluation
  - Preprocessing
  - Data mining

- **Functions:**
  - Classification
  - Sequential/Temporal patterns
  - Associations
  - Clustering/Segmentation

- **Techniques:**
  - Cluster Analysis
  - Induction
    - decision trees
    - rule induction
  - Neural networks
  - On-line Analytical processing
  - Data Visualization

Introduction to Classification

- **An important Data Mining Problem**
- **Input:** a database of training records
  - Class label attributes
  - Predictor Attributes
- **Goal**
  - to build a concise model of the distribution of class label in terms of predictor attributes
- **Applications**
  - scientific experiments, medical diagnosis, fraud detection, etc.

Introduction

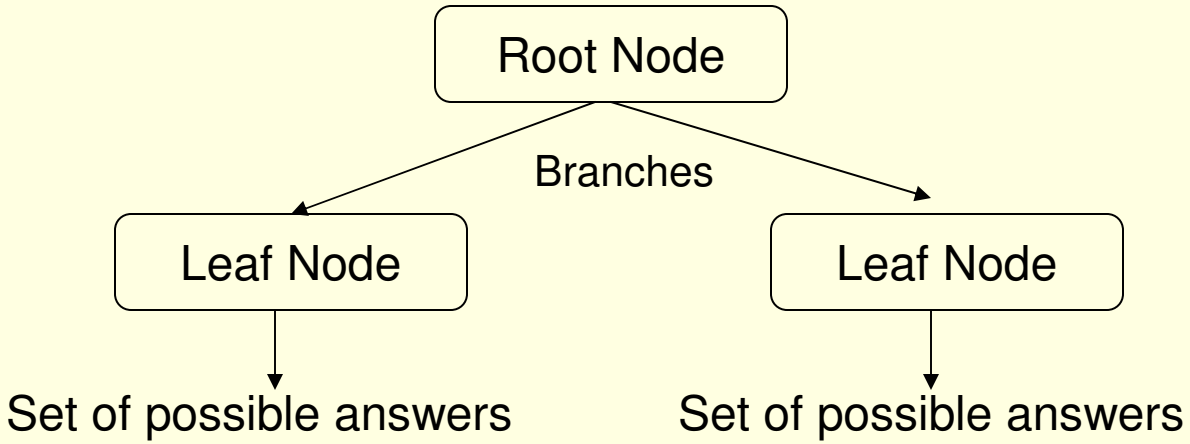
- Decision tree is the process of representing the induced knowledge by selecting the best attribute to obtain the compact tree with the high predictive accuracy.
- **Appropriate Problems for Decision Tree Learning**
  Decision tree learning is generally best suited to problems with the following characteristics:
  - Instances are represented by **attribute-value pairs**.
  - The target function has **discrete output values**.
  - Disjunctive descriptions may be required.
  - The training data may contain errors.
  - The training data may contain missing attribute values.

Introduction

- Decision tree has
  - A set of internal nodes which are decisions, where each node tests the value of an attribute and branches on all possible values.
  - A set of leaves which are labels, where each leaf gives a class value.
- Diagram Representation
  - Each non-leaf node is connected to a test that splits its set of possible answers into subsets corresponding to different test results.
  - Each branch carries a particular test result's subset to another node.
  - Each node is connected to a set of possible answers.

Learned functions are either represented by a decision tree or re-represented as sets of if-then rules to improve readability.

Introduction

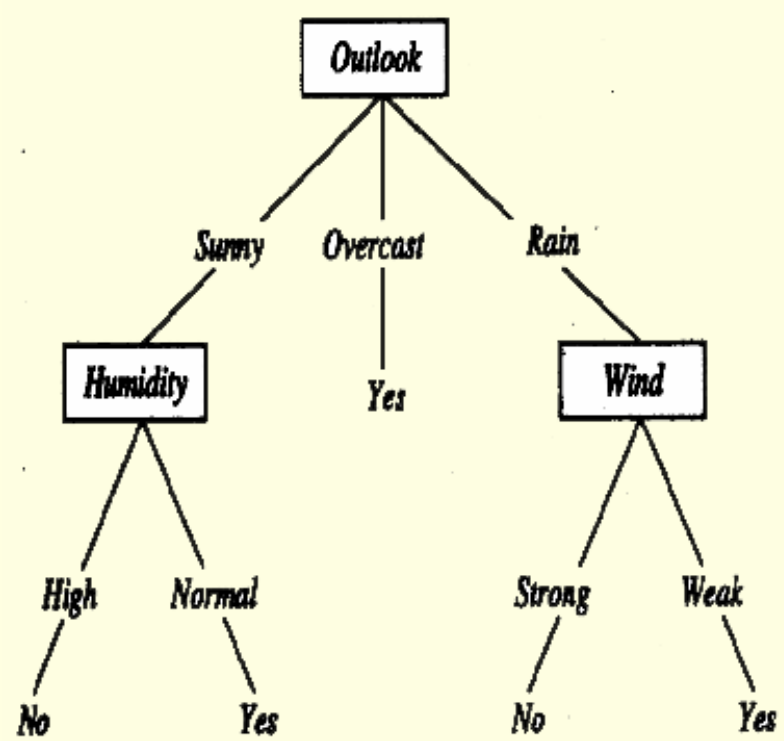


Given *m* attributes, a decision tree may have a maximum height of *m*.

Stages :

1. Growing of decision tree.
2. Pruning process.
3. Generation of rules.

Introduction



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Why Decision Tree Induction

- Compared to a neural network or a Bayesian classifier, a decision tree is easily interpreted/comprehended by humans
- While training neural networks can take large amounts of time and thousands of iterations, inducing decision trees is efficient and is thus suitable for large training sets
- Decision tree generation algorithms do not require additional information besides that already contained in the training data
- Decision trees display good classification accuracy compared to other techniques
- Decision tree induction can use SQL queries for accessing databases

Tree Quality Measures

- **Accuracy**
- **Complexity**
  - Tree size
  - Number of leaf nodes
- **Computational speed**
- **Scalability:** Classifying data sets with millions of examples and hundreds of attributes with reasonable speed

Entropy (ID3)

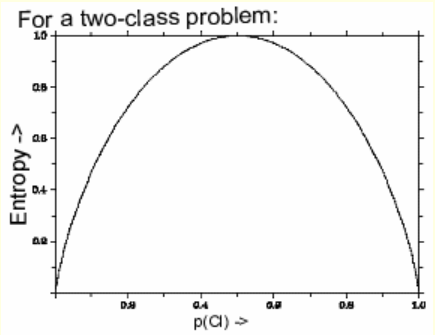
- **Probability**

$$P_b = \text{Probability an instance on a branch } b \text{ is positive}$$

$$= \frac{\text{number of positive instances on branch}}{\text{total number of instances on branch}} = \frac{n_{bc}}{n_b}$$

- **Entropy**

$$\text{Entropy} = \sum_c -\left(\frac{n_{bc}}{n_b}\right)\log_2\left(\frac{n_{bc}}{n_b}\right)$$



- **Average Entropy**

$$\text{Average Entropy} = \sum_b \left(\frac{n_b}{n_t}\right) \times \left[\sum_c -\left(\frac{n_{bc}}{n_b}\right)\log_2\left(\frac{n_{bc}}{n_b}\right)\right]$$

Split Selection

- **Information gain / Gain ratio (ID3/C4.5)**
  - All attributes are assumed to be categorical
  - Can be modified for continuous-valued attributes
- **Gini index (IBM IntelligentMiner)**
  - All attributes are assumed continuous-valued
  - Assume there exist several possible split values for each attribute
  - May need other tools, such as clustering, to get the possible split values
  - Can be modified for categorical attributes

Split Selection

- *T* – Training set; *S* – any set of cases
- *freq*(C_i, *S*) – the number of cases that belong to class C_i
- |*S*| -- the number of cases in set *S*
- Information of set S is defined:

$$info(S) = -\sum_{j=1}^{k} \frac{freq(C_j, S)}{|S|} \times \log_2 \frac{freq(C_j, S)}{|S|}$$

- consider a similar measurement after *T* has been partitioned in accordance with the *n* outcomes of a test *X*. The expected information requirement can be found as the weighted sum over the subsets, as

$$info_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times info(T_i)$$

- The quantity

measures the information that is gained by partitioning *T* in accordance with the test *X*. The *gain criterion*, then, selects a test to maximize this information gain.

$$gain(X) = info(T) - info_X(T)$$

Split Selection

- Gain criterion has a serious deficiency – it has a strong bias in favor of tests with many outcomes.
- We have

$$\text{split info}(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times \log_2 \frac{|T_i|}{|T|}$$

  This represents the potential information generated by dividing T into n subsets, whereas the information gain measures the information relevant to classification that arises from the same division.
- Then,

$$Gain\ ratio(X) = gain(X)/split\ info(X)$$

  expresses the proportion of information generated by the split that is useful, i.e., that appears helpful for classification.

Split Selection

- If a data set *T* contains examples from *n* classes, gini index, *gini*(*T*) is defined as

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

where p_j is the relative frequency of class *j* in *T*.

- If a data set *T* is split into two subsets T_1 and T_2 with sizes N_1 and N_2 respectively, the *gini* index of the split data contains examples from *n* classes, the *gini* index *gini*(*T*) is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute provides the smallest $gini_{split}(T)$ is chosen to split the node (*need to enumerate all possible splitting points for each attribute*).

ID3: Decision Tree Learning Algorithm

- **ID3(Examples, Target, Attributes)**
- Create a root node
- If all Examples have the same Target value, give the root that label
- Else if Attributes is empty label, the root according to the most common value
- Else begin
  - Calculate the information gain for each attribute, according to the average entropy formula
  - Select the attribute, **A**, with the lowest average entropy (highest information gain) and make this the attribute tested at the root
  - For each possible value, **v**, of this attribute
    - Add a new branch below the root, corresponding to **A** = **v**
    - Let Examples (**v**) be those examples with **A** = **v**
    - If Examples (**v**) is empty, make the new branch a leaf node labeled with the most common value among Examples
    - Else let the new branch be the tree created by **ID3**(Examples(**v**), Target, Attributes - {**A**})
- end
- Return root

ID3: Limitations

- **Some of the limitations of ID3 are:**
  - Sometimes decision trees conform to their training set too tightly (over fitted data).
  - Operate on discrete-valued functions and does not deal with continuous-valued functions.
  - It does not handle situations in which attributes do not have specified values.
  - Not computationally efficient.
  - Doesn't deal with scalability issues.
  - Requires Memory resident Data.

SLIQ (Supervised Learning In Quest)

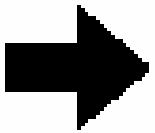
- No memory-resident data required.
- Handle numerical and categorical attributes.
- Pre-sorting technique is used in tree growing phase. This is integrated with a breadth-first tree growing phase.
- Gini Index is used as the splitting index.
- **Data Structure used**:
  - A separate list, called ***attribute list,*** for each attribute of the training data is created.
  - A separate list, called ***class list,*** is created for the class labels attached to the examples.

SLIQ

Pre-Sorting and creation of Attribute & Class List

TRAINING DATA

Age	Salary	Class
30	65	G
23	15	B
40	75	G
55	40	B
55	100	G
45	60	G



AFTER PRE-SORTING

Age List

Age	Class List Index
23	2
30	1
40	3
45	6
55	5
55	4

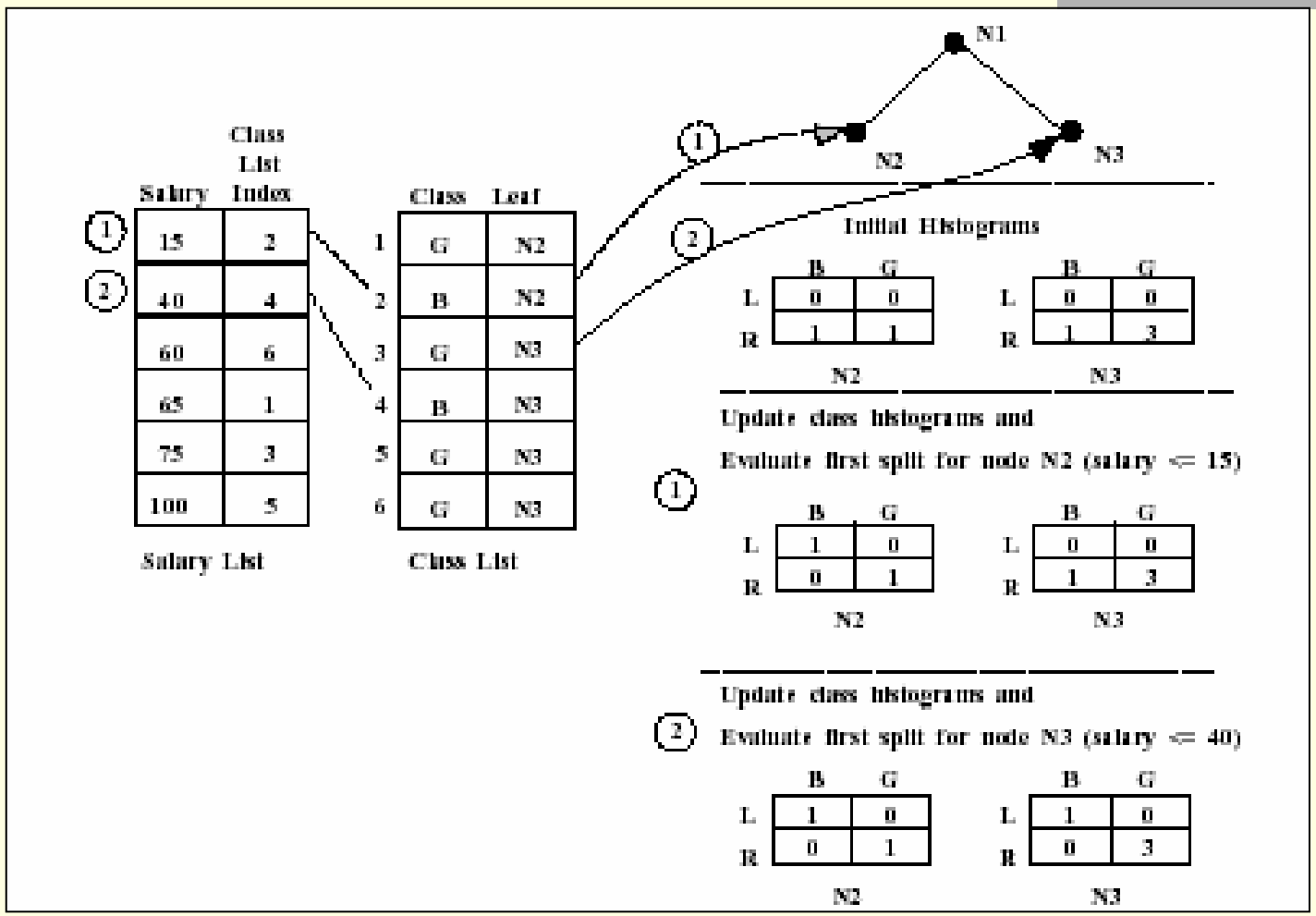
Salary List

Salary	Class List Index
15	2
40	4
60	6
65	1
75	3
100	5

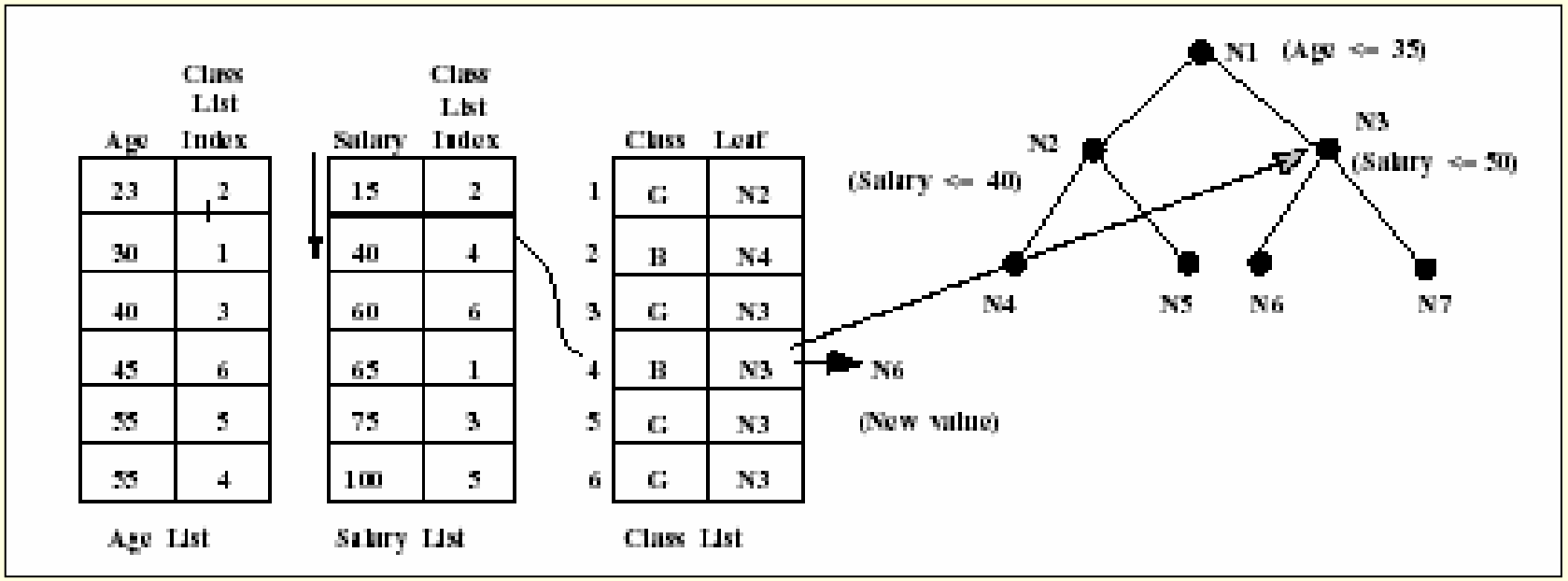
Class List

	Class	Leaf
1	G	N1
2	B	N1
3	G	N1
4	B	N1
5	G	N1
6	G	N1

SLIQ : Evaluating Splits



SLIQ: Class List Updating



SLIQ: Limitation

- The splits for categorical attribute a are of the form AЄS', where S'⊂S and S is the set of possible values of attribute A. The evaluation of all the subsets of S can be prohibitively expensive, especially if the *cardinality of S is large.*
- SLIQ uses a hybrid approach to overcome this issue. If the cardinality of S is less than a threshold, MAXSETSIZE, then all of the subsets of S are evaluated. Otherwise, a greedy algorithm is used to obtain the desired subset.
- It does not handle situations in which attributes do not have specified values or missing values.
- Requires some in-memory data structure that grows with the no. of input.

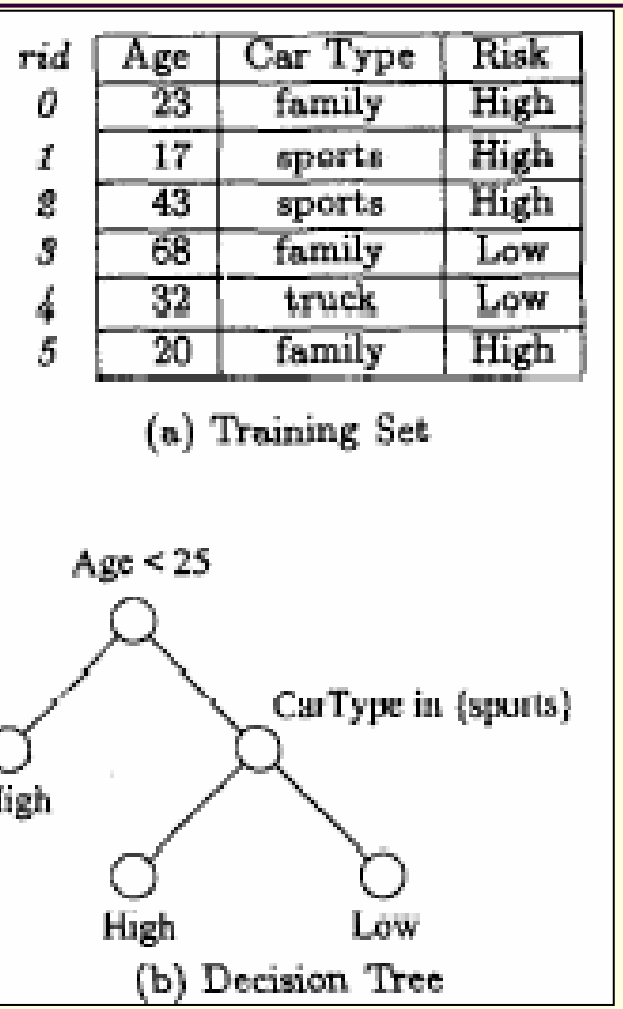
SPRINT: Scalable PaRallelizable INnduction of decision Trees

- **SPRINT: a new decision-tree-base algorithm**
  - Removes all of memory restrictions.
  - Fast and scalable.
  - Uses Gini Index.
- **Growth phase**
  - The tree is built by recursively partitioning the data until each partition is either “pure”.
  - Consider only binary splits.
- **The critical performance**
  - How to find split point.
  - How to partition the data.
- **SPRINT**
  - Shares with SLIQ the advantage of one-time sort.
  - Uses different data structures.

SPRINT: Data Structure

- **Attribute list**
  - SPRINT initially creates an attribute list for each attribute.
  - Entries in these lists called attribute records.
  - An attribute record consist of an attribute value, a class label, and the index of record (rid).
  - Continuous attributes are sorted when first created.
- **Histograms**
  - Continuous attributes
    - Uses C_{above} and C_{below} to capture the class distribution
  - Categorical attributes
    - Uses count matrix

SPRINT : Attribute List



rid	Age	Car Type	Risk
0	23	family	High
1	17	sports	High
2	43	sports	High
3	68	family	Low
4	32	truck	Low
5	20	family	High

(a) Training Set

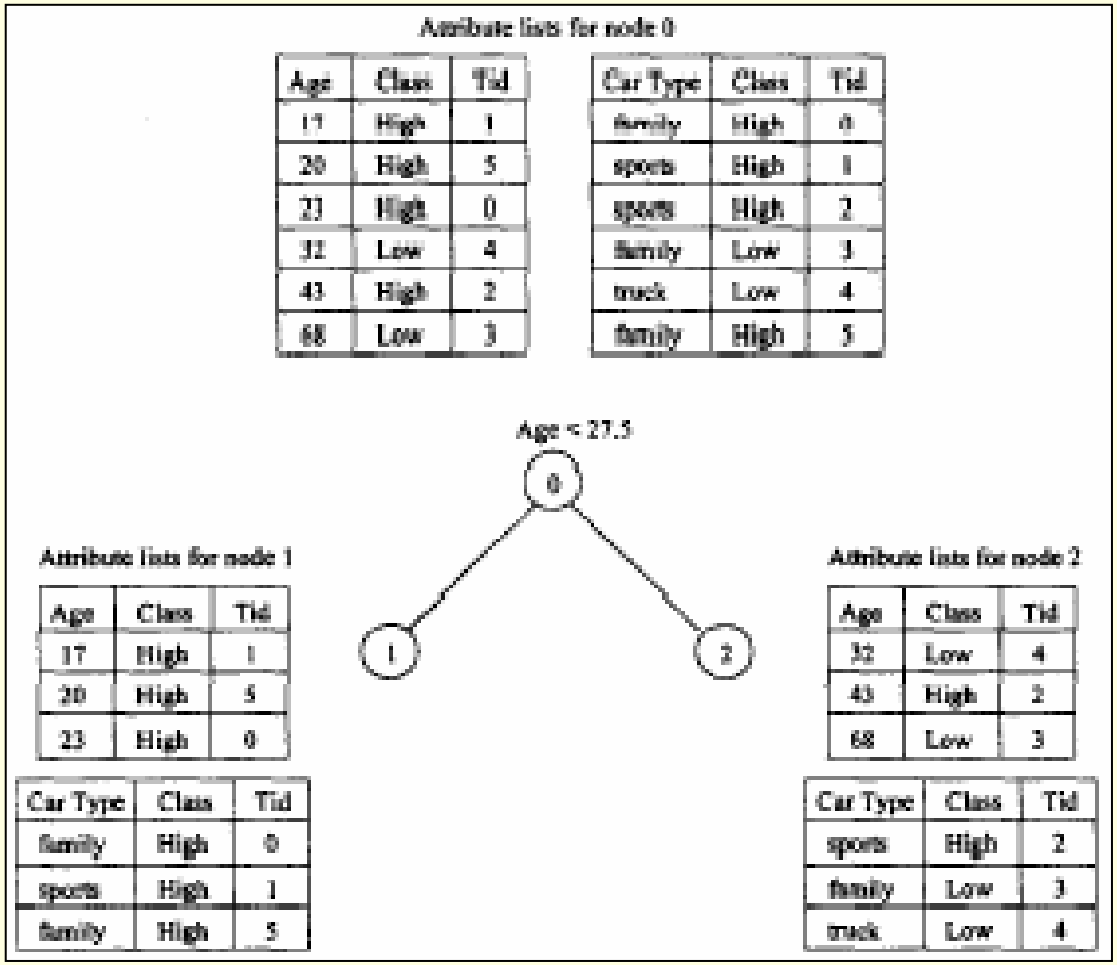
(b) Decision Tree

Age	Class	rid
17	High	1
20	High	5
23	High	0
32	Low	4
43	High	2
68	Low	3

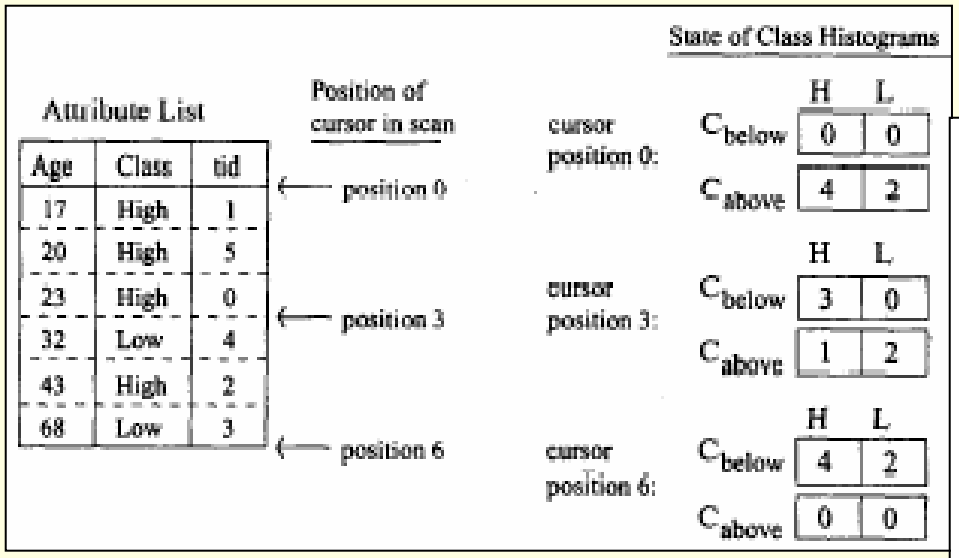
Car Type	Class	rid
family	High	0
sports	High	1
sports	High	2
family	Low	3
truck	Low	4
family	High	5

Example of attribute lists

SPRINT: Splitting a node attribute



SPRINT: Finding Split Point



Attribute List

Age	Class	tid
17	High	1
20	High	5
23	High	0
32	Low	4
43	High	2
68	Low	3

Position of cursor in scan

← position 0
← position 3
← position 6

State of Class Histograms

cursor position 0:

	H	L
C_{below}	0	0
C_{above}	4	2

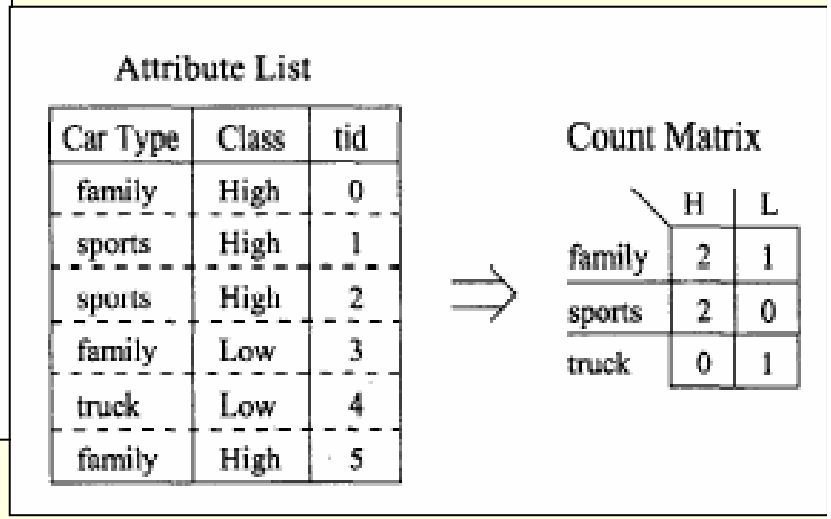
cursor position 3:

	H	L
C_{below}	3	0
C_{above}	1	2

cursor position 6:

	H	L
C_{below}	4	2
C_{above}	0	0

Evaluating Continuous Split point



Attribute List

Car Type	Class	tid
family	High	0
sports	High	1
sports	High	2
family	Low	3
truck	Low	4
family	High	5

⇒

Count Matrix

	H	L
family	2	1
sports	2	0
truck	0	1

Evaluating Categorical split point

SPRINT : Parallelizing

- **Data Placement and workload Balancing**
  - Distributing the training-set
  - Each processor generates its own attribute-list
  - Categorical attributes require no further processing
  - Continuous attribute lists must now be sorted
- **Finding split points**
  - Continuous attributes
    - C_{above} and C_{below} must be initialized to reflect the fact that there are sections on other processors.
  - Categorical attributes
    - must be exchange “local”count-metrix to get “gobal” count-metrix
- **Performing the Splits**
  - must be exchange *rid*

SPRINT : Parallelizing

Parallel Data Placement :

Processor 0

Age	Class	rid
17	High	1
20	High	5
23	High	0

Car Type	Class	rid
family	High	0
sports	High	1
sports	High	2

Processor 1

Age	Class	rid
32	Low	4
43	High	2
68	Low	3

Car Type	Class	rid
family	Low	3
truck	Low	4
family	High	5

Some Published Methods

- PUBLIC (VLDB'98 — Rastogi & Shim)
  - integrates tree splitting and tree pruning: stop growing the tree earlier
- RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - separates the scalability aspects from the criteria that determine the quality of the tree
  - builds an AVC-list (attribute, value, class label)
- BOAT
- CLOUDS

Other Issues

- **Allow for continuous-valued attributes**
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- **Handle missing attribute values**
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- **Attribute (feature) construction**
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication
- **Integration of data warehousing techniques**
- **Different data access methods**
- **Bias in split selection**

References

- http://www.cs.cornell.edu
- http://www.cs.uregina.ca
- http://www.risc.uni-linz.ac.at
- http://www.cs.brandeis.edu
- http://mycroft.ncsa.uiuc.edu
- Jiawei Han and Micheline Kamber. Data Mining-Concepts and technique. Morgan Kaufmann publication.
- Michael j. A. Berry and Gordon S. Linoff. Mastering Data Mining. Wiley publication.
- J. Ross Quinlan. Introduction of decision tree. Machine Learning, 1:81-106, 1986.
- J. Ross Quinlan. C4.5: programs for Machine Learning. Morgan Kaufmann, 1993.

References

- Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Database mining: A performance per-spective. IEEE Transactions on Knowledge and Data Engineering, 5(6):914-925, December 1993
- J. Gehrke, R. Ramakrishnan, V. Ganti. “RainForest – A Framework for Fast Decision Tree Construction of Large Datasets”.
- Blaž Zupan, Ivan Bratko “Induction of Decision Trees”.
- John C Shafer, Rakesh Agarwal, and Manish Mehta. SPRINT: A scalable parallel classifier for data mining. Research report, IBM Almaden Research Center, San Jose, California, 1996. Available from http://www.almaden.ibm.com/cs/quest.
- Manish Mehta, Rakesh Agarwal and Jorma Rissanen. SLIQ: A fast Scalable Classifier for Data Mining. Research report, IBM Almaden Research Center, San Jose, California

Thank You .