

Classification and Prediction (II)

Liaquat Majeed Sheikh

liaquat.majeed@nu.edu.pk

**National University of Computer
and Emerging Sciences**

Practical Issues with Decision Trees

- a) how deep to grow the decision tree
- b) handling continuous attributes
- c) choosing an appropriate attribute selection measure
- d) handling training data with missing attributes
- e) handling attributes with different costs
- f) improving computational efficiency

- The basic ID3 algorithm met so far can be extended
- to handle the above points. The result was Quinlan's "C4.5"
- (Quinlan, 1993). (Quinlan invented ID3).

Terminating Conditions

- All tuples for a given node belong to the same class OR
- There are no remaining attributes for partitioning testing, OR
 - Majority voting is employed for classifying the leaf
 - The most frequent class among the tuples at this node is chosen as the class of this node
- There are no tuples left
 - The most frequent class among the tuples at the parent node is chosen as the class of this node

what criterion is to be used to determine the correct final tree size?

- Use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.
- Use all the available data for training, but apply a statistical test to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set. For example, Quinlan uses a chi-square test to estimate whether further expanding a node is likely to improve performance over the entire instance distribution, or only on the current sample of training data.
- Use an explicit measure of the complexity for encoding the training examples and the decision tree, halting growth of the tree when this encoding size is minimized. This approach, based on a heuristic called the Minimum Description Length principle.

Alternative Measures

- Gain ratio: penalize attributes like date by incorporating split information

$$\textit{SplitInformation} (S, A) \equiv - \sum_{i=1}^c \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

- Split information is sensitive to how broadly and uniformly the attribute splits the data.

$$\textit{GainRatio} (S, A) \equiv \frac{\textit{Gain} (S, A)}{\textit{SplitInformation} (S, A)}$$

- Gain ratio can be undefined or very large
 - Only test attributes with average gain

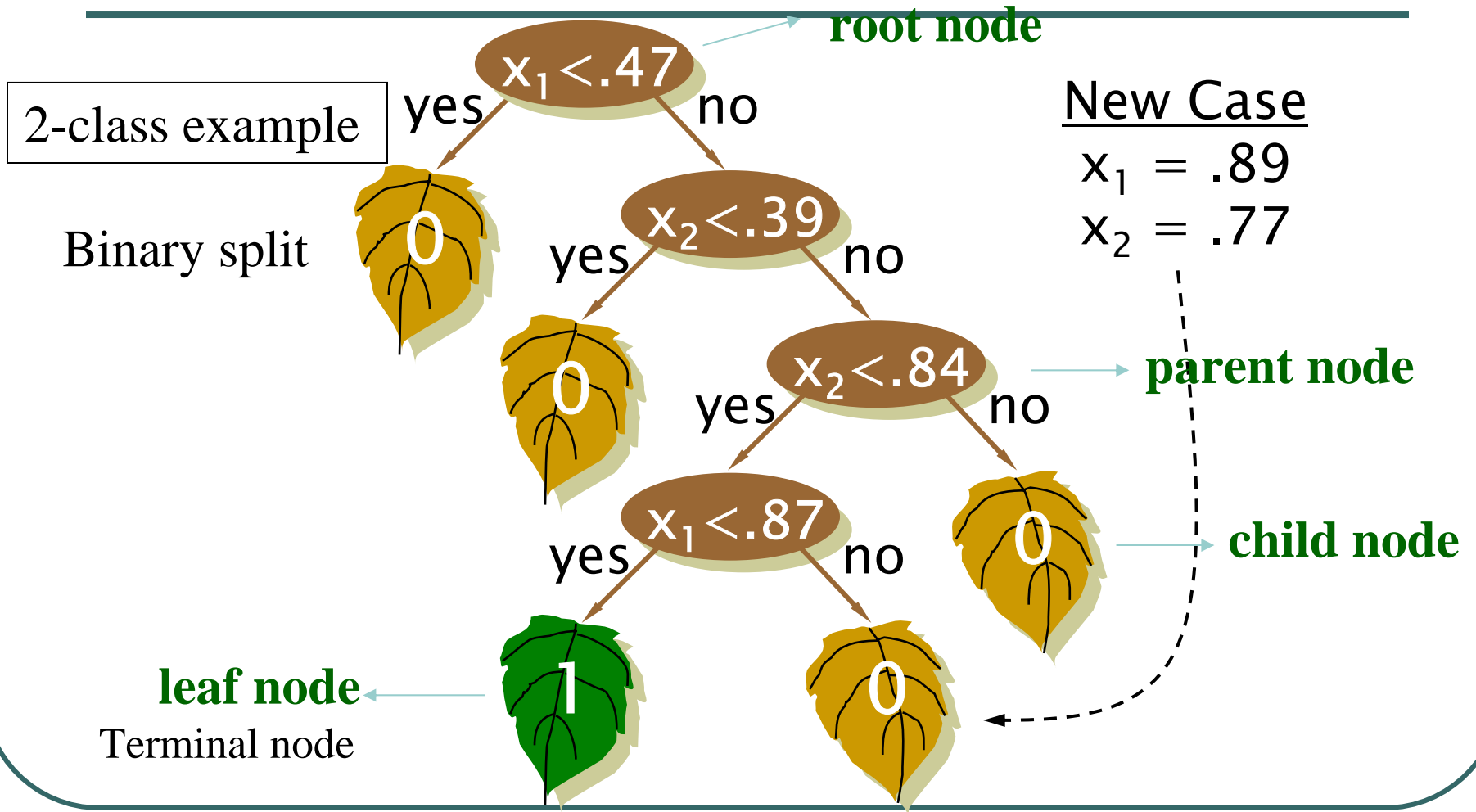
Tree Induction

- Greedy strategy
 - Choose to split records based on an attribute that optimizes splitting criterion.
- Two phases at each node
 - Split determining phase:
 - How to split a given attribute
 - Which attribute to split on? Use splitting criterion.
 - Splitting phase:
 - Split the records into children.

Splitting based on continuous attributes

- Different ways of handling:
 - Static: Based on Apriori Discretization to form a categorical attribute
 - May not be desirable in many situations
 - Dynamic: Make decisions as algorithm proceeds
 - Complex but more powerful and flexible in approximating true dependency

Tree?



Splitting based on continuous attributes (cont.)

- Dynamic Decisions
 - From binary decision based on the one value
 - Two partitions: $A < v$ and $A \geq v$
 - Find ranges of values and use each range to partition
 - Ranges can be found by a simple minded bucketing or by more intelligent clustering techniques.
 - Example: salary [0, 15k], [15k, 60k], [60k – 100k]
 - Find a linear combination of multiple variables, and make binary decisions or range-decisions based on it

Splitting Based on Gini - Index

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

Note: $p(j|t)$ is the relative frequency of class j at node t

- Measure impurity of a 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 belongs to one class implying least interesting information.

C1	0
C2	6
Gini = 0.00	

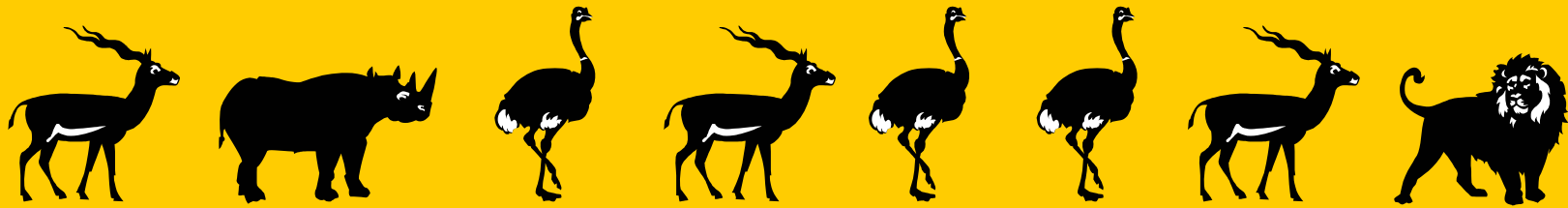
C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Diversity and Gini Index

high diversity, low purity



$$G = 1 - (3/8)^2 - (3/8)^2 - (1/8)^2 - (1/8)^2 = .69$$

low diversity, high purity



$$G = 1 - (6/7)^2 - (1/7)^2 = .24$$

CART(classification and regression tree)

- Invented by Breiman, Friedman, Olshen, and Stone (1984)
- If “response” is continuous, CART does regression! Otherwise, classification!
- Binary split
- Two necessary conditions for CART
 - Split rule=Gini index
 - Pruning=Cost-complexity
- Original CART is sold by Salford systems (<http://www.salford-systems.com>)

Splitting Based on Gini Index

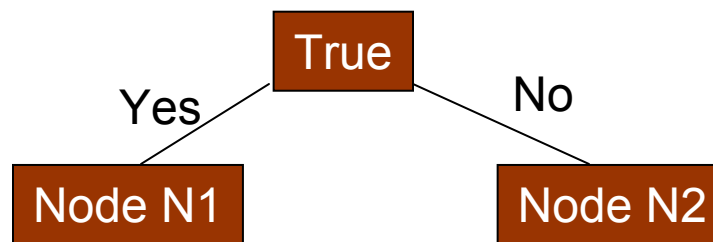
- Used in CART, SLIQ and SPRINT
- Splitting Criterion: Minimize Gini Index of the split.
- When a node p is split into k partitions (children), the quality of split is computed as:

$$GINI_{split} = \sum_i^k \frac{n_i}{n} Gini(i)$$

where n_i = number of children at child i
 n = number of records at node p

Binary Attributes: Computing Gini Index

- Splits into two partitions
- Effect of Weighing partitions
 - Large or pure partitions are sought for



	N1	N2
C1	0	4
C2	6	0
Gini = 0.00		

	N1	N2
C1	3	4
C2	3	0
Gini = 0.300		

	N1	N2
C1	4	2
C2	4	0
Gini = 0.400		

	N1	N2
C1	6	2
C2	2	0
Gini = 0.300		

Categorical Attributes: Computing Gini Index

- For each distinct value, gather counts for each class in the set.
- 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 attribute: Computing Gini index

- Use binary decision 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 $A \geq v$
- Simple method to choose best v
 - For each v , scan the data base to gather count matrix and compute its Gini index
 - Computationally inefficient! Repetition of work

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

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...

Cheat	No		No		No		Yes		Yes		Yes		No		No		No		No			
	Taxable Income																					
	60		70		75		85		90		95		100		120		125		220			
Sorted Values →	55		65		72		80		87		92		97		110		122		172		230	
Split Pos →	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>
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		<u>0.300</u>		0.343		0.375		0.400		0.420	

Classification in Large Databases

- Why decision tree induction on large data sets?
 - Relatively faster learning speed (than other classification methods)
 - Convertible to simple and easy to understand classification rules
 - Can use SQL queries for database accesses
- What about the training data not in memory?
- Scalability: build classifiers for large data sets with many attributes in a reasonable speed.

SLIQ (Supervised Learning In Quest)

- Assumption: the training data set cannot be held in memory
 - Bottleneck: determining the best split of each attribute
 - Have to sort examples by attributes repeatedly
- Presorted attribute lists and class list
- Breadth-first growth of decision tree

SLIQ (Supervised Learning In Quest)

- It is a scalable algorithm which uses a pre-sorting technique , integrated with a breadth-first tree growing strategy for the classification of the disk resident data.
- The basic idea is to avoid repeated sorting , by maintaining an attribute list for each attribute .
- It builds a decision tree in a breadth-first manner and avoids repeated sorting at every step. The choice of the splitting criterion depends on the domain of the attribute being numeric or categorical .

Major Steps of determining the split

- Calculating Splitting index for each value of each attribute .
- Determining the best split.
- Partitioning the data set of the current node and allocate to each child node .

Computation of Splitting Index

- Gini index is used as goodness measure .
- From attribute lists ,class histograms are computed .
- One pass over the attribute list is required for determining the splitting point of an attribute. It will be clear from an example .

Example (for numerical attributes)

- For numeric attributes, the candidate split point is taken to be the mid point of two consecutive distinct values .
- Each tuple of the attribute list is read one by one and class histograms are updated accordingly . Since the attribute list is sorted , the histogram for the next tuple can be updated using the previous values.
- Gini index values for all attribute lists can be computed in one pass and the best split point can be found .

Example of Gini split index:

Salary	Class
75	G
40	B
100	G
60	G



	<=35		<=50		<=70		<=80		<=120	
	N2	N3	N2	N3	N2	N3	N2	N3	N2	N3
G	0	3	0	3	1	2	2	1	3	0
B	0	1	1	0	1	0	1	0	1	0
	0.375		<u>0</u>		0.25		0.333		0.375	

$$gini(T) = 1 - \sum P_j^2$$

<=50

$$Gini(N2) = 1 - (0/1)^2 - (1/1)^2 = 0$$

$$Gini(N3) = 1 - (3/3)^2 - (0/1)^2 = 0$$

$$Gini(split) = 1/4 * 0 + 3/4 * 0 = 0$$

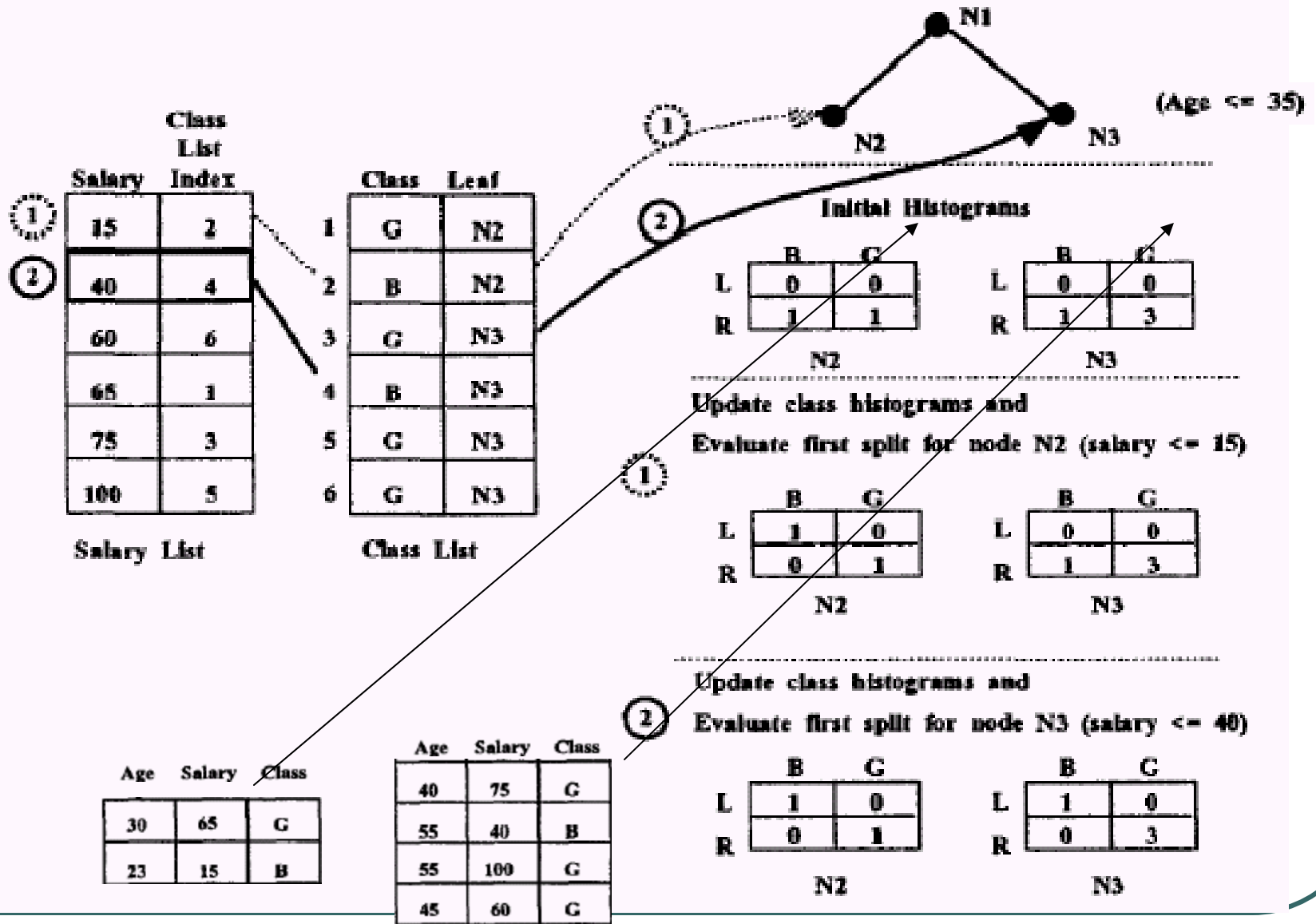
<=70

$$Gini(N2) = 1 - (1/2)^2 - (1/2)^2 = 1/2$$

$$Gini(N3) = 1 - (2/2)^2 - (0/2)^2 = 0$$

$$Gini(split) = 2/4 * 1/2 + 2/4 * 0 = 1/4 = 0.25$$

Example of Evaluating Splits:



Example of Updating Class List

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

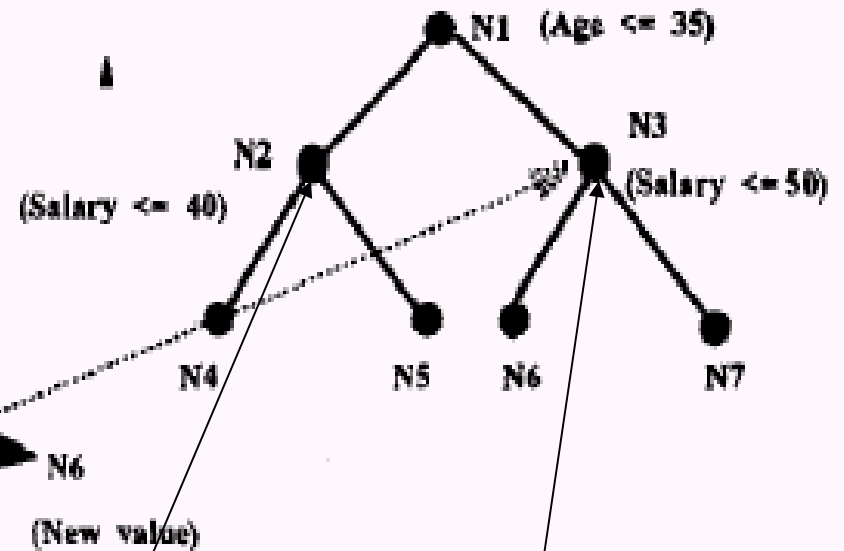
Age List

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

Salary List

Class List	
1	G N2
2	B N4
3	G N3
4	B N3
5	G N3
6	G N3

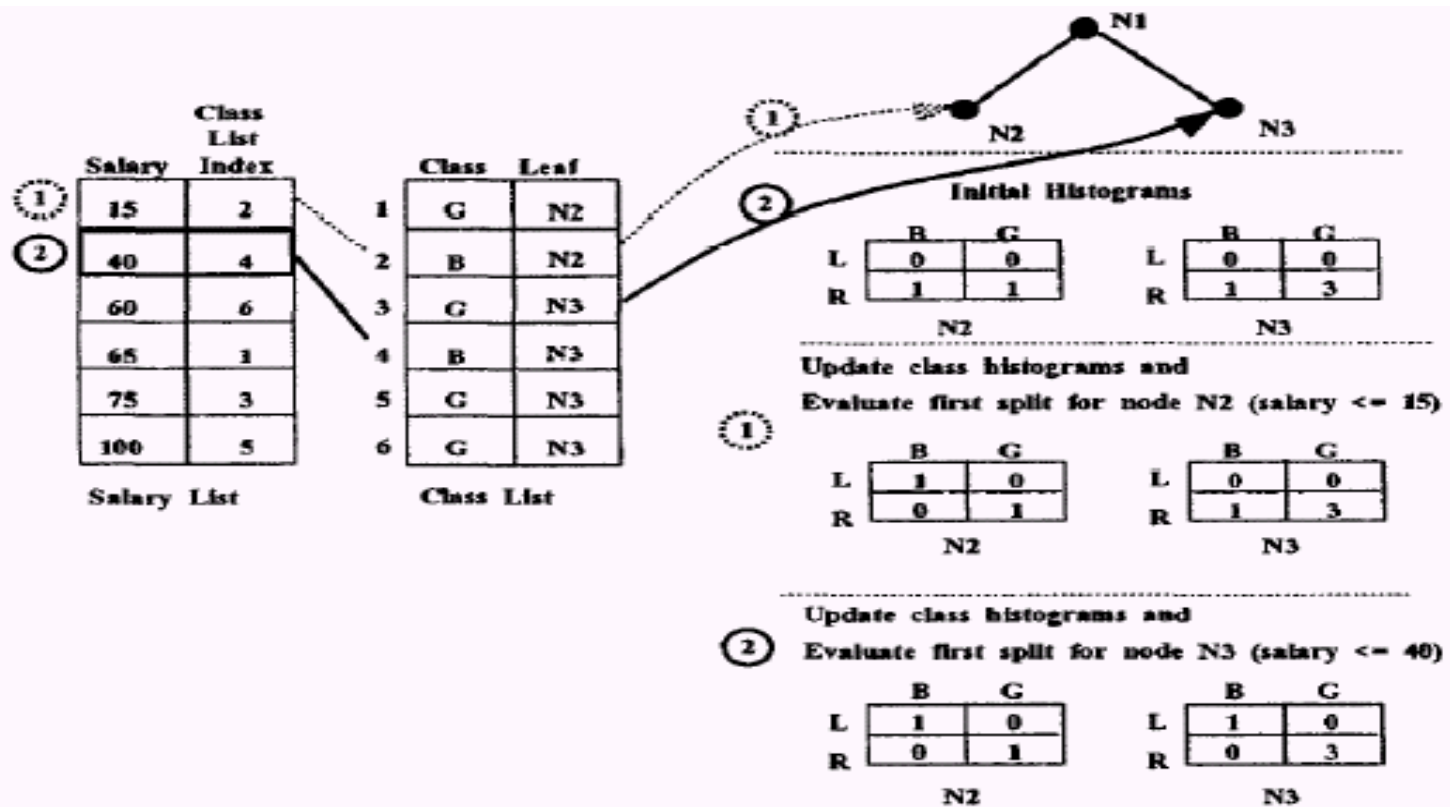
Class List



Age	Salary	Class
30	65	G
23	15	B

Age	Salary	Class
40	75	G
55	40	B
55	100	G
45	60	G

Example



SPRINT Serial Algorithm

- A decision tree classifier is built in two phases
 - Growth phase
 - Prune phase
- Growth phase is computationally much more expensive than pruning

```
Partition(Data  $S$ )  
  if (all points in  $S$  are of the same class) then  
    return;  
  for each attribute  $A$  do  
    evaluate splits on attribute  $A$ ;  
  Use best split found to partition  $S$  into  $S_1$  and  $S_2$ ;  
  Partition( $S_1$ );  
  Partition( $S_2$ );
```

Initial call: **Partition(TrainingData)**

Figure 2: General Tree-growth Algorithm

Serial Algorithm: Growth Phase

- **Key Issue**
 - To find split points that define node tests.
 - Having chosen a split point, how to partition the data
- **Data Structures**
 - Attribute lists
 - Histograms

Serial Algorithm: Data Structure (Attribute lists)

- SPRINT creates an attribute list for each attribute
- Entries are called attribute records which contains
 - Attribute value
 - Class label
 - Index of the record

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

Figure 3: Example of attribute lists

Serial Algorithm: Data Structure (Attribute lists)

- The initial lists are associated with the root
- As the tree is grown, the attribute lists belonging to each node are partitioned and associated with the children

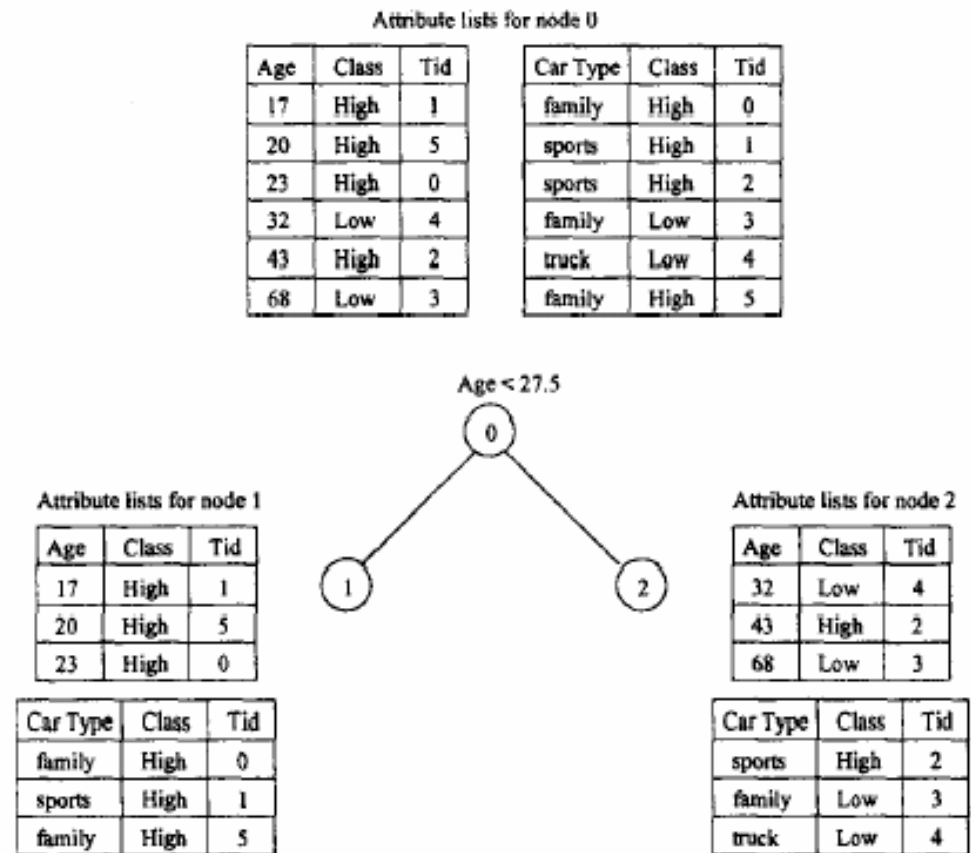


Figure 4: Splitting a node's attribute lists

Serial Algorithm: Data Structure (Histograms)

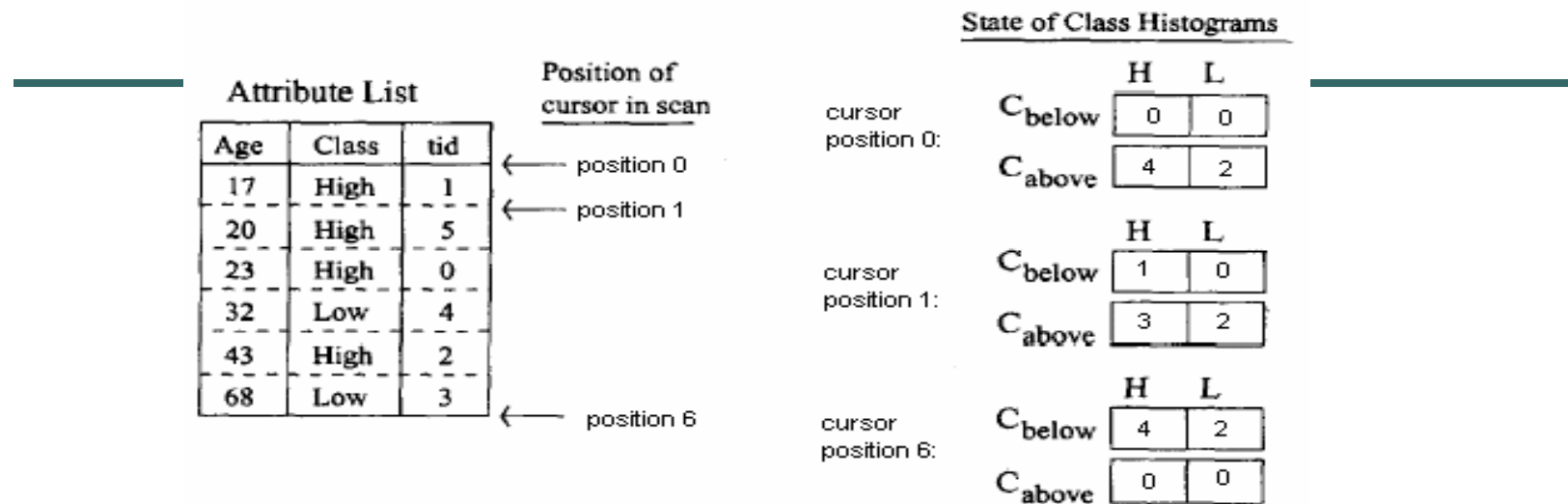


Figure 5: Evaluating continuous split points

- For continuous attributes, two histograms are associated with each decision-tree node. These histograms, denoted as C_{above} and C_{below}
 - C_{below} : maintains this distribution for attribute records that already been processed
 - C_{above} : maintains this distribution for attribute records that have not been processed

Serial Algorithm: Data Structure (Histograms)

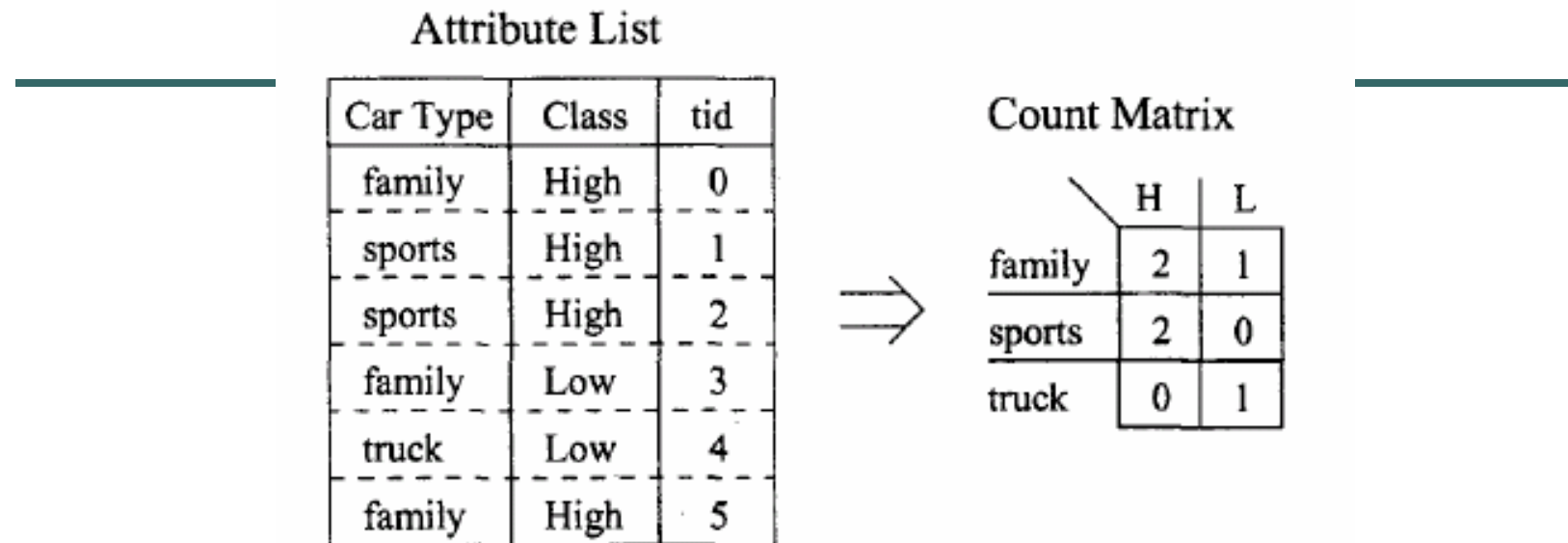


Figure 6: Evaluating categorical split points

- For categorical attributes, one histogram associated with a node. However, only one histogram is needed and called count matrix

Serial Algorithm: Finding split points

- While growing the tree, the goal at each node is to determine the split point that “best” divides the training records belonging to the leaf
- Gini index is used to evaluate the split point
- $Gini(S) = 1 - \sum p_j^2$
 - Where p_j is the relative of class j in S
- $Gini_{split}(S) = n_1/n(S_1) + n_2/n(S_2)$

Serial Algorithm: Example

Split point for the continuous attributes

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

Cursor
Position 3:

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

$$Gini_{\text{split}} = \frac{n_1}{n} gini(S_1) + \frac{n_2}{n} gini(S_2)$$

Example :

$$Gini_{\text{split } 3} = \frac{3}{6} gini(S_1) + \frac{3}{6} gini(S_2)$$

$$gini(S_1) = 1 - \left[\left(\frac{1}{1} \right)^2 + \left(\frac{0}{1} \right)^2 \right] = 0$$

$$gini(S_2) = 1 - \left[\left(\frac{1}{3} \right)^2 + \left(\frac{2}{3} \right)^2 \right] = 0.44$$

$$Gini_{\text{split } 3} = 0.22$$

Serial Algorithm: Example

Split point for the continuous attributes

	Position 0		Position 1		Position 2		Position 3		Position 4		Position 5		Position 6	
	B	A	B	A	B	A	B	A	B	A	B	A	B	A
H	0	4	1	3	2	2	3	1	3	1	4	0	4	0
L	0	2	0	2	0	2	0	2	1	1	1	1	2	0
Gini	0	.44	0	.48	0	.5	0	.44	.375	.5	.32	0	.44	0
Gini Split	.44		.4		.33		<u>.22</u>		.41		.26		.44	

Serial Algorithm: Example

Split point for the continuous attributes

$$\text{Gini}_{\text{split0}} = 0.44$$

$$\text{Gini}_{\text{split1}} = 0.40$$

$$\text{Gini}_{\text{split2}} = 0.33$$

$$\text{Gini}_{\text{split3}} = 0.22$$

$$\text{Gini}_{\text{split4}} = 0.41$$

$$\text{Gini}_{\text{split5}} = 0.26$$

$$\text{Gini}_{\text{split6}} = 0.44$$

- After finding all the gini indexes we choose the lowest as the split point
- Therefore, we split at position 3 where the candidate split point is the mid-point between age 23 and 32 (i.e. Age < 27.5)

Serial Algorithm: Example

Split point for the categorical attributes

	H	L
Family	2	1
Sports	2	0
Truck	0	1

Example

$$gini_{split(sport)} = \frac{2}{6} gini(S_1) + \frac{4}{6} gini(S_2)$$

$$gini(S_1) = 1 - \left(\frac{2}{2}\right)^2 - \left(\frac{0}{2}\right)^2 = 0$$

$$gini(S_2) = 1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2 = 0.5$$

$$gini_{split(sport)} = 0.33$$

$$gini_{split(family)} = 0.44$$

$$gini_{split(truck)} = 0.266$$

Serial Algorithm: Performing the split

- Once the best split point has been found for a node, we then execute the split by creating child nodes and dividing the attribute records between them
- For the rest of the attribute lists (i.e. CarType) we need to retrieve the information by using rids

Serial Algorithm: Comparison with SLIQ

- SLIQ does not have separate sets of attribute lists for each node
- Advantage
 - Do not have to rewrite these lists during a split
 - Reassignment of records is easy
- Disadvantage
 - It must stay in memory all the time which limits the amount of data that can be classified by SLIQ

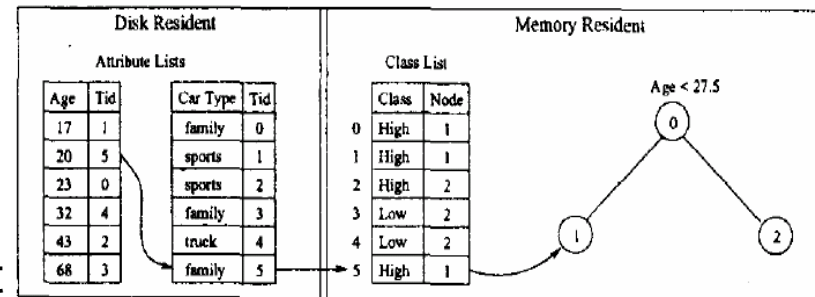


Figure 7: Attribute and Class lists in SLIQ

SPRINT was not to outperform SLIQ. Rather, the purpose of the algorithm is to develop an accurate classifier for datasets, and to develop a classifier efficiently. Furthermore, SPRINT is designed to be easily parallelizable, thus the workload can be shared among N processor

SPRINT: Parallelizing Classification

- SPRINT was specifically designed to remove any dependence on data structures that are either centralized or memory-resident
- These algorithms all based on a shared-nothing parallel environment where each of N processor has private memory and disks. The processor are connected by a communication network and can communicate only by passing message

Parallelizing Classification

Data placement and Workload Balancing

- The main data structures are
 - attribute lists
 - class histograms
- SPRINT achieves uniform data placement and workload balancing by distributing the attribute lists evenly over N processor
- Each processor to work on only 1/N of the total data

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

Figure 8: Parallel Data Placement

Parallelizing Classification

Finding split points (Continuous attributes)

- In a parallel environment, each processor has a separate contiguous section of a “global” attribute list
 - C_{below} : must initially reflect the class distribution of all sections of an attribute-list assigned to processors of lower rank
 - C_{above} : must initially reflect the class distribution of the local section as well as all sections assigned to processor of higher rank

Parallelizing Classification

Example: Split point for the continuous attributes

Processor 0

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

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

Processor 1

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

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

Parallelizing Classification:

Finding split points (Categorical attributes)

- Each processor built a counter matrix for a leaf
- Exchange these matrices to get the “global” counts
- Sums the local matrices to get the global count matrices

Parallelizing Classification: Performing the Splits

- Splitting the attribute lists for each leaf is identical to the serial algorithm
- The only difference is that before building the probe structure, we will need to collect rids from all the processors.
- Exchange the rids
- Each processor constructs a probe-structure with all the rids and using it to split the leaf's remaining attribute lists

Parallelizing Classification: Parallelizing SLIQ

- Two approaches for parallelizing SLIQ
 - The class list is replicated in memory of every processor (SLIQ/R)
 - The class list is distributed such that each processor's memory holds only a portion of the entire list (SLIQ/D)
- Disadvantage
 - SLIQ/R – the size of the training set is limited by the memory size of a single processor
 - SLIQ/D – High communication cost

Performance Evaluation

- The primary metric for evaluating classifier performance
 - Classification accuracy
 - Classification time
 - Size of the decision tree
- The accuracy and tree size characteristics of SPRINT are identical to SLIQ
- Focus only on the classification time

Some Approximate Algorithms

- CLOUDS (Classification of Large or Out-of-Core Data Sets) . It is a kind of approximate version of the SPRINT method .
- BOAT (Bootstrap Optimistic Algorithm for Tree Construction) .It is based on Sampling .
- CHAID
- CART (Classification and regression Tree)