Scalable Decision Tree Methods

Decision Tree Construction

- Construct a tree in a top-down recursive divide-and-conquer manner
  - Which attribute is the best at the current node?
  - Create a node for each possible attribute value
  - Partition training data into descendant nodes
- Conditions for stopping recursion
  - All samples at a given node belong to the same class
  - No attribute remained for further partitioning
    - Majority voting is employed for classifying the leaf
  - There is no sample at the node

Decision Tree on Numeric Data

- A numeric attribute has a potentially infinite number of possible values
  - A branch per value does not work
- Finding one or multiple splitting points to partition a numeric attribute domain into a set of ranges, one branch a range

Example

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

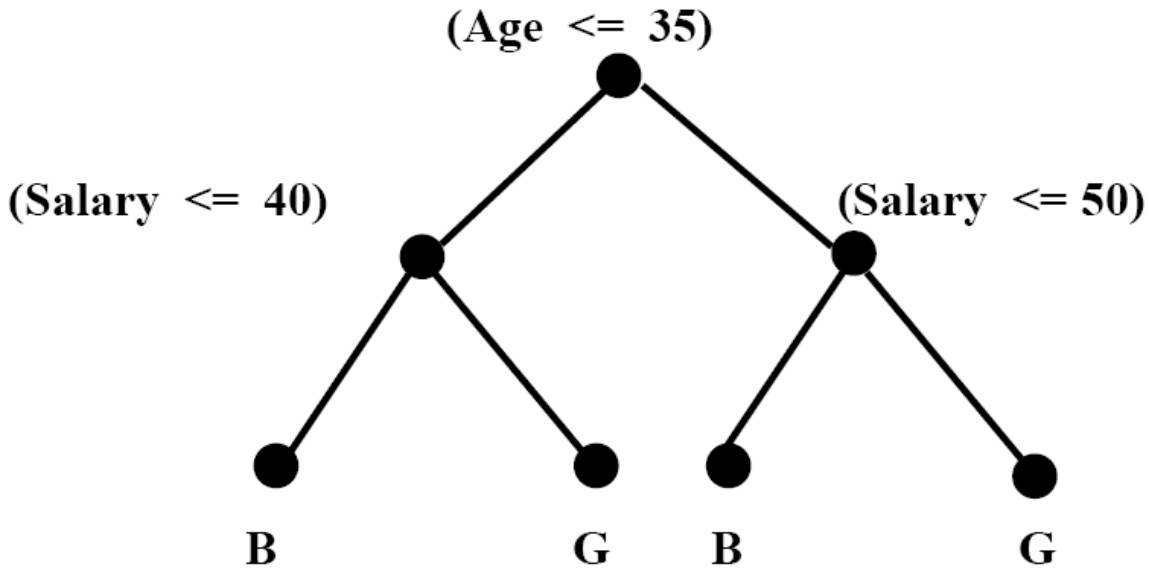


Figure from [MAR96]

Numeric/Categorical Attributes

Age	Car Type	Risk
23	family	High
17	sports	High
43	sports	High
68	family	Low
32	truck	Low
20	family	High

Generally, a categorical attribute can also be split into subsets

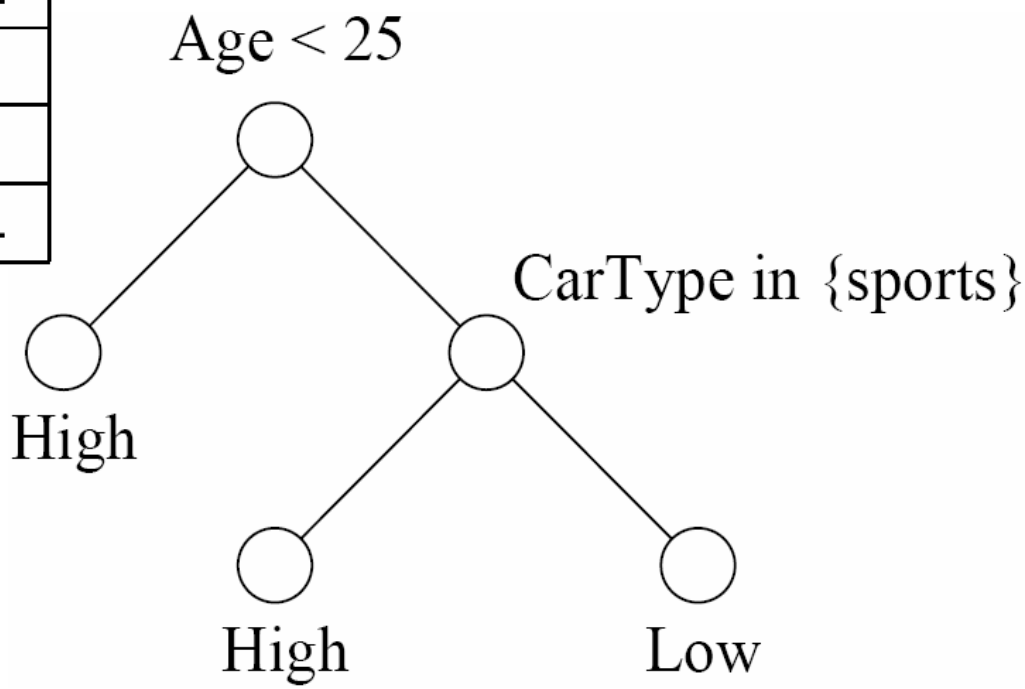


Figure from [GRG98]

Scalability Issues

- Two major operations in decision tree building
  - Evaluate splits for each attribute, select the best split – the most costly step
  - Create partitions using the best split – a simple application of the splitting criteria
- To find the best split in an attribute, we need to sort training examples on the attribute
  - Sorting needs to be conducted for each node
  - Sorting can be costly when the database cannot be held in main memory

Classification in Large Databases

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

SLIQ

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

Attribute Lists in SLIQ

Training data

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

Attribute lists

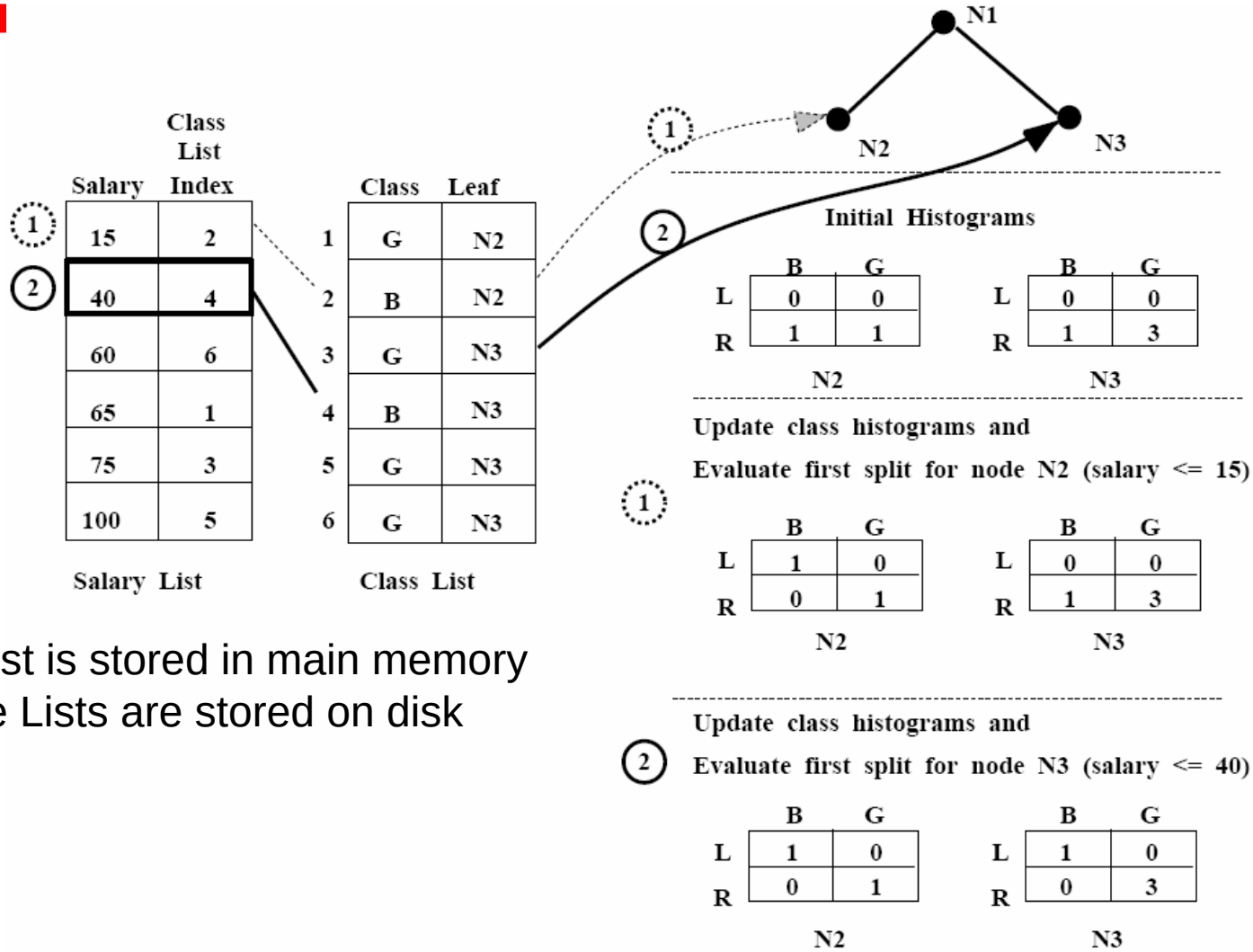
Age	Class list index
23	2
30	1
40	3
45	6
55	5
55	4

Salary	Class list index
15	2
40	4
60	6
65	1
75	3
100	5

Class list

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

Attribute and Class Lists



Class List is stored in main memory
Attribute Lists are stored on disk

Major Ideas in SLIQ

- Breadth-first growth of decision trees
  - All nodes at the same level are constructed at the same time
- One scan of an attribute list determines the split points for the attribute for all nodes at a level
  - An attribute list is sorted
  - When tuples of value A=v are read, the entropy/gini index of A<=v can be computed

Tree Pruning Using MDL

- Minimum Description Length (MDL): minimizing Cost(M, D)=Cost(D | M) + Cost(M)
  - Cost(D | M) is the cost in number of bits of encoding data D given a model M
  - Cost(M) is the cost of encoding the model M
- In the context of decision tree, Cost(D | M) is the the sum of all classification errors

Encoding a Decision Tree

- Encoding a node
  - Case 1: each node has either 0 or 2 children – 1 bit is needed
  - Case 2: each node has either 0 children, a left child, a right child, or both children – 2 bits are needed
- Encoding a split
  - A numeric attribute A<=v: the cost of encoding v
  - A categorical attribute $A \in S$: ln n_A, n_A is the number of nodes using this test

From SLIQ to SPRINT

- The class list in SLIQ must stay in memory
  - Bottleneck: the class list can be huge
- SPRINT: put class information in attribute lists
  - No class list anymore
- Parallelizing classification
  - Partition the attribute lists

Example of Attribute Lists

Training data

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

Attribute lists

Age	Class	rid
23	B	2
30	G	1
40	G	3
45	G	6
55	G	5
55	B	4

Salary	Class	rid
15	B	2
40	B	4
60	G	6
65	G	1
75	G	3
100	G	5

Integrating Decision Tree Growing and Pruning

- Growing then pruning a tree is costly
  - Waste time in constructing the part to be pruned
- Integrate growing and pruning
  - At each node, test the cost of growing sub-tree
  - Cost of encoding data records

$$C(S) = \sum_i n_i \log \frac{n}{n_i} + \frac{k-1}{2} \log \frac{n}{2} + \log \frac{\pi^{k/2}}{\Gamma(k/2)}$$

- Not grow any node certainly will be pruned

RainForest: A Generic Framework

- What is the bottleneck of scalability?
  - Computing the attribute-value, class label (AVC-group) for each node
- RainForest: separate quality and scalability designs, focus on scalability
  - A set of algorithms for fast AVC-group computation

BOAT

- Bootstrapped optimistic decision tree construction
  - Bootstrapping sampling: sample the data set uniformly with replacement
- Splitting points in samples can be close to global ones
  - Use bootstrapped samples to determine correct splitting points
- Incrementally update/locally reconstruct a tree
- Construct several levels by one scan

Summary

- Decision tree construction for numeric data
- Scalability issues for decision tree building on large databases
  - Finding the best split for each attribute for each node
- Scalable solutions
  - Using attribute lists (and class lists)
  - Using specific AVC routines to address scalability
  - Using sampling and statistic methods

To-Do-List

- Read the paper “Rainforest - a framework for fast decision tree construction of large datasets” and understand how the scalability of data mining methods can be isolated and addressed in the database context