SLIQ: Supervised Learning in Quest

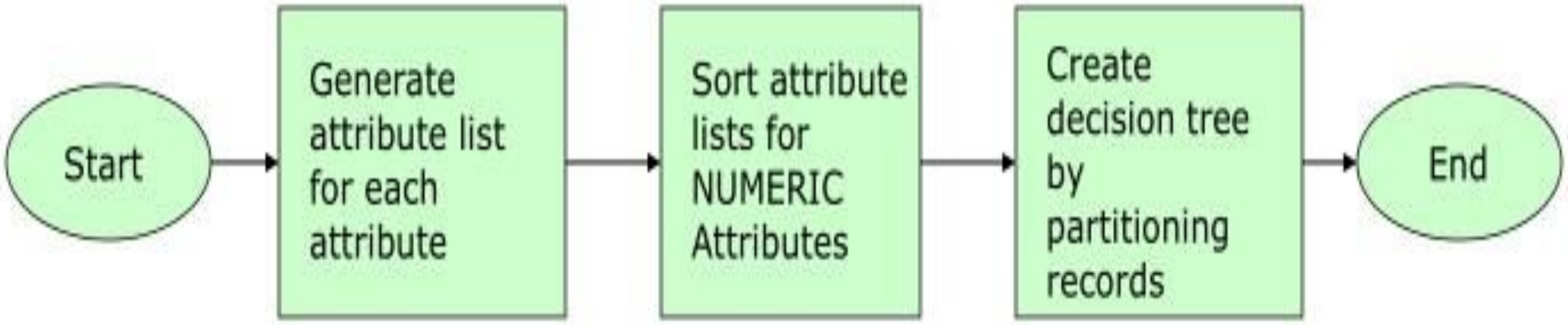
SLIQ

- SLIQ is a decision tree classifier that can handle both numerical and categorical attributes
- Uses a pre-sorting technique in the tree growing phase
- Suitable for classification of large disk-resident datasets.

Issues

- There are two major, critical performance, issues in the tree-growth phase:
  - How to find split points
  - How to partition the data
- The well-known decision tree classifiers:
  - Grow trees depth-first
  - Repeatedly sort the data at every node
- SLIQ:
  - Replace this repeated sorting with one-time sort
  - Use Breadth-first Search
  - Use a new data structure call class-list
  - Class-list must remain memory resident at all time

SLIQ Methodology



SLIQ – Data Structure

- Attribute List
  - An entry in attribute list has two fields
    - Attribute Value
    - Index into the class list
- Class List
  - An entry in class list also has two fields.
    - Class label
    - Reference to the leaf node of decision tree
  - The ith entry of the class list corresponds to the ith example in the training data

rid	age	salary	marital	car
1	30	60	single	sports
2	25	20	single	mini
3	40	80	married	van
4	45	100	single	luxury
5	60	150	married	luxury
6	35	120	single	sports
7	50	70	married	van
8	55	90	single	sports
9	65	30	married	mini
10	70	200	single	luxury

SLIQ - Attribute Lists

rid	age
1	30
2	25
3	40
4	45
5	60
6	35
7	50
8	55
9	65
10	70

rid	salary
1	60
2	20
3	80
4	100
5	150
6	120
7	70
8	90
9	30
10	200

rid	marital
1	single
2	single
3	married
4	single
5	married
6	single
7	married
8	single
9	married
10	single

These are projections on (rid, attribute).

SLIQ - Sort Numeric, Group Categorical

rid	age
2	25
1	30
6	35
3	40
4	45
7	50
8	55
5	60
9	65
10	70

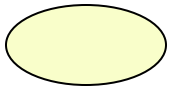
rid	salary
2	20
9	30
1	60
7	70
3	80
8	90
4	100
6	120
5	150
10	200

rid	marital
3	married
5	married
7	married
9	married
1	single
2	single
4	single
6	single
8	single
10	single

SLIQ - Class List

rid	car	LEAF
1	sports	N1
2	mini	N1
3	van	N1
4	luxury	N1
5	luxury	N1
6	sports	N1
7	van	N1
8	sports	N1
9	mini	N1
10	luxury	N1

N1



SLIQ - Histograms

rid	age
2	25
1	30
6	35
3	40
4	45
7	50
8	55
5	60
9	65
10	70

rid	car	LEAF
1	sports	N1
2	mini	N1
3	van	N1
4	luxury	N1
5	luxury	N1
6	sports	N1
7	van	N1
8	sports	N1
9	mini	N1
10	luxury	N1

N1

	sports	mini	van	luxury
L	0	0	0	0
R	3	2	2	3

age≤25 ?

	sports	mini	van	luxury
L				
R				

age≤30 ?

	sports	mini	van	luxury
L				
R				

...

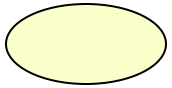
- Evaluate each split, using GINI or Entropy.
- L values represents the distributions for examples that satisfies the test
- The R values represents examples that do not satisfies the test

SLIQ - Histograms

rid	age
2	25
1	30
6	35
3	40
4	45
7	50
8	55
5	60
9	65
10	70

rid	car	LEAF
1	sports	N1
2	mini	N1
3	van	N1
4	luxury	N1
5	luxury	N1
6	sports	N1
7	van	N1
8	sports	N1
9	mini	N1
10	luxury	N1

N1



	sports	mini	van	luxury
L	0	0	0	0
R	3	2	2	3

age≤25

	sports	mini	van	luxury
L	0	1	0	0
R	3	1	2	3

age≤30

	sports	mini	van	luxury
L	1	1	0	0
R	2	1	2	3

Evaluate each split, using GINI or Entropy.

...

SLIQ - Histograms

rid	salary
2	20
9	30
1	60
7	70
3	80
8	90
4	100
6	120
5	150
10	200

rid	car	LEAF
1	sports	N1
2	mini	N1
3	van	N1
4	luxury	N1
5	luxury	N1
6	sports	N1
7	van	N1
8	sports	N1
9	mini	N1
10	luxury	N1

N1

	sports	mini	van	luxury
L	0	0	0	0
R	3	2	2	3

salary≤20

	sports	mini	van	luxury
L	0	1	0	0
R	3	1	2	3

salary≤30

	sports	mini	van	luxury
L	0	2	0	0
R	3	0	2	3

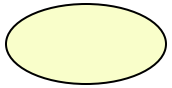
...

Evaluate each split, using GINI or Entropy.

SLIQ - Histograms

rid	marital
3	married
5	married
7	married
9	married
1	single
2	single
4	single
6	single
8	single
10	single

rid	car	LEAF
1	sports	N1
2	mini	N1
3	van	N1
4	luxury	N1
5	luxury	N1
6	sports	N1
7	van	N1
8	sports	N1
9	mini	N1
10	luxury	N1

N1 

Married

	sports	mini	van	luxury
Yes	0	1	2	1
No	3	1	0	2

Single

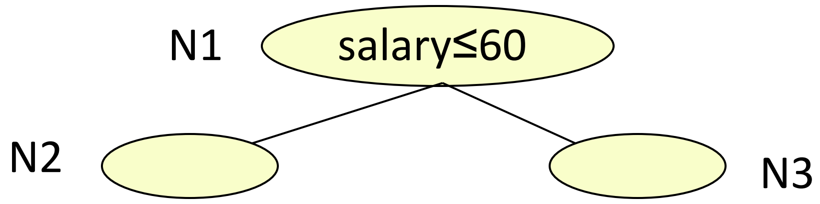
	sports	mini	van	luxury
Yes	3	1	0	2
No	0	1	2	1

Evaluate each split, using GINI or Entropy.

SLIQ - Perform best split and Update Class List

rid	salary
2	20
9	30
1	60
7	70
3	80
8	90
4	100
6	120
5	150
10	200

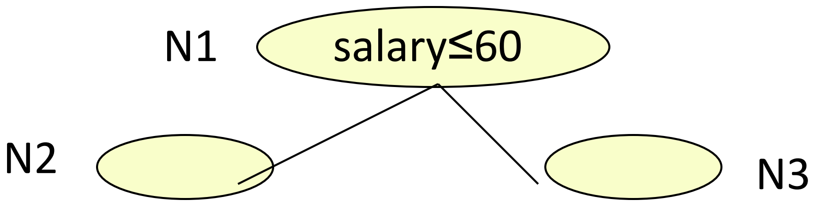
rid	car	LEAF
1	sports	N1
2	mini	N1
3	van	N1
4	luxury	N1
5	luxury	N1
6	sports	N1
7	van	N1
8	sports	N1
9	mini	N1
10	luxury	N1



SLIQ - Perform best split and Update Class List

rid	salary
2	20
9	30
1	60
7	70
3	80
8	90
4	100
6	120
5	150
10	200

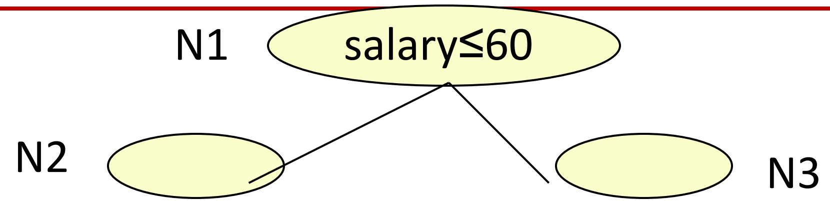
rid	car	LEAF
1	sports	N2
2	mini	N2
3	van	N3
4	luxury	N3
5	luxury	N3
6	sports	N3
7	van	N3
8	sports	N3
9	mini	N2
10	luxury	N3



SLIQ - Histograms

rid	age
2	25
1	30
6	35
3	40
4	45
7	50
8	55
5	60
9	65
10	70

rid	car	LEAF
1	sports	N2
2	mini	N2
3	van	N3
4	luxury	N3
5	luxury	N3
6	sports	N3
7	van	N3
8	sports	N3
9	mini	N2
10	luxury	N3



N2

	sports	mini	van	luxury
L	0	0	0	0
R	1	2	0	0

N3

	sports	mini	van	luxury
L	0	0	0	0
R	2	0	2	3

age≤25 ?

N2

	sports	mini	van	luxury
L				
R				

N3

	sports	mini	van	luxury
L				
R				

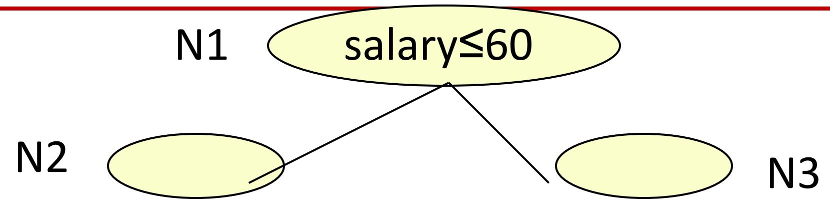
...

Evaluate each split, using GINI or Entropy.

SLIQ - Histograms

rid	age
2	25
1	30
6	35
3	40
4	45
7	50
8	55
5	60
9	65
10	70

rid	car	LEAF
1	sports	N2
2	mini	N2
3	van	N3
4	luxury	N3
5	luxury	N3
6	sports	N3
7	van	N3
8	sports	N3
9	mini	N2
10	luxury	N3



N1

	sports	mini	van	luxury
L	0	0	0	0
R	1	2	0	0

N2

	sports	mini	van	luxury
L	0	0	0	0
R	2	0	2	3

. . . .

age≤25

N1

	sports	mini	van	luxury
L	0	1	0	0
R	1	1	0	0

N2

	sports	mini	van	luxury
L	0	0	0	0
R	2	0	2	3

...

Evaluate each split, using GINI or Entropy.

SLQ - Pseudocode

- **Split evaluation:**

```
EvaluateSplits()
    for each numeric attribute A do
      for each value v in the attribute list do
        find the corresponding entry in the class list, and
        hence the corresponding class and the leaf node Ni
        update the class histogram in leaf Ni
        compute splitting score for test (A ≤ v) for Ni

    for each categorical attribute A do
      for each leaf of the tree do
        find subset of A with best split
```

SLIQ - Pseudocode

- Updating the ***class list***

```
UpdateLabels()
    for each split leaf Ni do
      Let A be the split attribute for Ni.
      for each (rid,v) in the attribute list for A do
        find the corresponding entry in the class list e (using
      the rid)
        if the leaf referenced by e is Ni then
              find the new leaf Nj to which (rid,v) belongs
            (by applying the splitting test)
              update the leaf pointer for e to Nj
```

SLIQ - bottleneck

- Class-list must remain memory resident at all time!
  - Although not a big problem with today's memories, still there might be cases where this is a bottleneck.

SPRINT: A Scalable Parallel Classifier for Data Mining

Outline

- Why do we need scaling?
  - Data doesn't fit in RAM
  - Numeric attributes require repeated sorting
  - Noisy datasets lead to very large trees
  - Large datasets fundamentally different from smaller ones
    - Can't store the entire dataset
    - Underlying phenomenon changes over time

SPRINT - Overview

- SPRINT stands for <u>S</u>calable <u>P</u>a<u>R</u>allelizable <u>IN</u>duction of decision <u>T</u>rees
- A fast, scalable classifier, requires one scan over the data per level of the induced tree
- Use pre-sorting method
- No memory restriction
- Easily parallelized
  - Allow many processors to work together to build a single consistent model
  - The parallel version is also scalable

Serial Algorithm

- A decision tree classifier is built in two phases
  - Growth phase
  - Prune phase
- In the growth phase, the tree is built by recursively partitioning the data until each partition is either “pure” or sufficiently small.
- The tree growth phase is computationally much more expensive than pruning, since the data is scanned multiple times in this part of the computation.

```
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 S1 and S2;
    Partition(S1);
    Partition(S2);

Initial call: Partition(TrainingData)
```

Serial Algorithm

- Growth Phase: Key Issues
  - To find split points that define node tests.
  - Having chosen a split point, how to partition the data
- SPRINT addresses the above two issues differently from previous algorithms
- It has no restriction on the size of input and yet is a fast algorithm.

SPRINT - Data Structure

- Data Structures
  - Attribute lists
  - Histograms
- SPRINT creates an attribute list for each attribute
- Entries are called attribute records which contains
  - Attribute value
  - Class label
  - Index of the record
- Initial list for Continuous attributes are sorted by attributes value once when first created

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

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

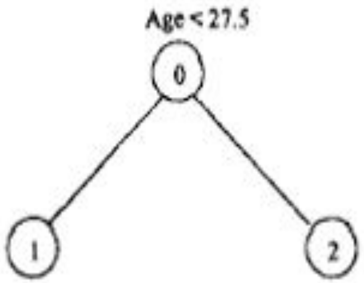
Data Structure - Attribute lists

- If entire data does not fit in the memory, attribute lists are maintained in the memory
- 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
- When a list is partitioned, the order of the records in the list is preserved

Attribute lists for node 0

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

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



Attribute lists for node 1

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

Car Type	Class	Tid
family	High	0
sports	High	1
family	High	5

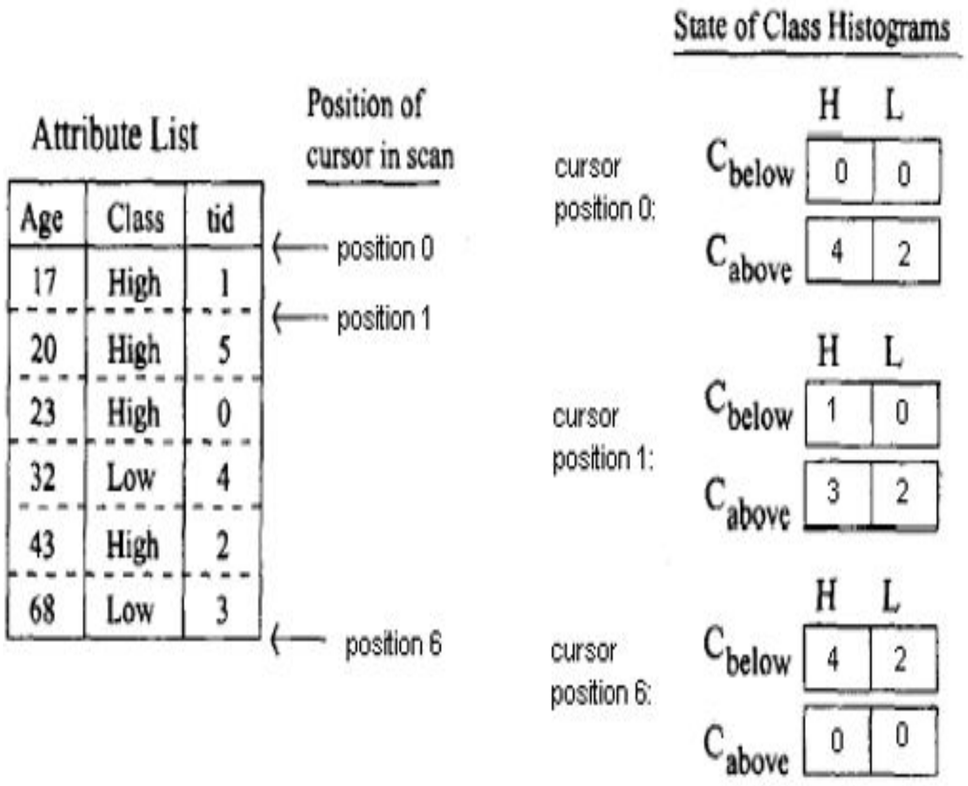
Attribute lists for node 2

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

Car Type	Class	Tid
sports	High	2
family	Low	3
truck	Low	4

Data Structure – Histograms

- For continuous attributes, two histograms are associated with each decision-tree node that is under consideration for splitting
- These histograms, denoted as C_{above} and C_{below}, are used to capture the class distribution of the of the attribute records at a given node.
- 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



Data Structure – Histograms

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

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

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**
- **In SPRINT, gini index is used to determine split point**

$$Gini_{split} = \frac{n1}{n} gini(S1) + \frac{n2}{n} gini(S2)$$

$$Example:$$

$$Ginisplit3 = \frac{3}{6} gini(S1) + \frac{3}{6} gini(S2)$$

$$gini(S1) = 1 - [(\frac{3}{3})^2 + (\frac{0}{3})^2] = 0$$

$$gini(S2) = 1 - [(\frac{1}{3})^2 + (\frac{2}{3})^2] = 0.44$$

$$Ginisplit3 = 0.22$$

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

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

Cursor Position 3

Finding split points

- 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)

$$Gini_{split0} = 0.44$$

$$Gini_{split1} = 0.40$$

$$Gini_{split2} = 0.33$$

$$Gini_{split3} = 0.22$$

$$Gini_{split4} = 0.41$$

$$Gini_{split5} = 0.26$$

$$Gini_{split6} = 0.44$$

Finding split points

- **Example – Split point for the categorical attributes**

	H	L
Family	2	1
Sports	2	0
Truck	0	1

Example

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

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

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

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

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

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

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

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

Data placement and Workload Balancing

- 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
- Each processors then generates its own attribute-list partition in parallel.

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

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
- Each processor scans its list to find its best split
- Processors communicate to determine the best split

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

Finding split points - Categorical attributes

- Each processor builds the count matrix
- A coordinator collect all the count matrices
- Sum up all counts and find the best split

Performing the Splits

- Almost identical to the serial version
- Except the processor needs <rids, child> information from other processors
- After getting information about all rids from other processors, it can build a hash table and partition the attribute lists