Introduction

Classification is an important data mining problem. Input: database of training records
Each record has a class label and predictor attributes. The resulting model assigns class labels to testing records.

Decision tree construction algorithms:
- Easily assimilated by humans
- Can be constructed fast
- Highly accurate

Outline

1. Introduction
2. Problem Definition
3. Related Work
4. RainForest Framework
   - Family of Algorithms
5. Experiments
6. Conclusion

Proposed approaches to deal with large datasets:
- Discretize ordered attributes
- Sampling at each node of the classification tree
- Assume dataset fits in main memory
- Quality?
Introduction (cont’d)

RainForest framework scales up the existing decision tree construction algorithms.

Data access algorithms scale with the size of database, adapt to available main memory, and are not restricted to a specific classification algorithm.

RainForest applied to existing algorithms, results in a scalable version of the algorithm without modifying the result of the algorithm.

Problem Definition

RainForest framework scales up the existing decision tree construction algorithms.

Data access algorithms scale with the size of database, adapt to available main memory, and are not restricted to a specific classification algorithm.

Problem Definition (cont’d)

For each internal node $n$

- Splitting attribute
- Set of predicates
- Combined information of splitting attribute and splitting predicates: splitting criteria on $n$, denoted as $\text{crit}(n)$.

Related Work

The literature survey shows that almost all the previous approaches do not scale to large datasets.

Sprint [SAM96] works for large databases.
- Builds classification trees with binary split
- Uses gini index to decide on splitting criteria
- Uses Minimal Description Length pruning (no test sample is needed)

Related Work (cont’d)

- To decide on splitting attribute at a tree node \( n \), Sprint needs to access \( F(n) \) for each ordered attribute in sorted order.
- Creates an attribute list for each attribute.

### RainForest Framework

Most decision tree algorithms (C4.5, CART, CHAID, FACT, ID3, SLIQ, Sprint, and Quest)

- Consider every attribute individually
- Need the distribution of class labels for each distinct value of an attribute to decide on the splitting criteria.

---

**AVC-set (Attribute Value Classlabel)** of a predictor attribute \( a \) at a node \( n \) is the projection of \( F(n) \) onto \( a \) and the class label whereby counts of individual class labels are aggregated.

**AVC-group** of a node \( n \) is the set of all AVC-sets at \( n \).
RainForest Framework (cont’d)

Depending on the amount of main memory available, three cases may happen:

1. The AVC-Group of the root node fits in main memory.
2. Each individual AVC-set of the root node fits in main memory, but not the AVC-Group of the root.
3. None of individual AVC-sets of the root node fits in main memory.

Proposed algorithms RF-Write, RF-Read, and RF-Hybrid deal with case 1, and RF-Vertical deals with case 2.

In RainForest family of algorithms, the following steps are carried for each node \( n \):

1. AVC-group construction
2. Choose splitting attribute and predicate
3. Partition \( F(n) \) across the children nodes

• First the database is scanned to build the AVC-group of the root node \( r \).
• Then the AVC-group is passed to the \( CL \) (classification algorithm being scaled by RainForest) to compute \( crit(r) \).
• The children are allocated & another scan is made over the database to partition the database across children of the root node \( r \).
• RF-Write is applied to each partition recursively

At each level of the tree, families belonging to that level, are read twice and written once.
RF-Read

- Database is scanned to build the AVC-group of the root node \( r \).
- Then the AVC-group is passed to \( CL \) to compute \( crit(r) \).
- Children are allocated

Suppose there is enough memory to hold the AVC-groups of all children
- Another scan is made over the database to build the AVC-groups of children simultaneously.
- Continue the same process until a level \( L \) where not all AVC-groups of new nodes \( N \) fit in memory
- Divide \( N \) into \( g_L \) groups such that the AVC-group of each fits in memory
- Process each group individually: \( g_L \) scans over the database is needed

Increasing number of database scans as the decision tree gets deeper

RF-Hybrid

- Starts with RF-Read.
- Continue the same process until a level \( L \) where not all AVC-groups of new nodes \( N \) fit in memory
- Now RF-Hybrid switches to RF-Write but tries to use the available memory efficiently
- Choose \( M \subset N \) the AVC-groups of which are constructed \textit{while} writing the partitions in \( N \).

Each \( n \in M \) has a cost and benefit

<table>
<thead>
<tr>
<th>cost</th>
<th>The size of its AVC-group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benefit</td>
<td>The size of ( F(n) )</td>
</tr>
</tbody>
</table>

A modified greedy approximation of \textit{Knapsack} problem is applied to choose \( M \)

RF-Vertical

The AVC-group of the root node \( r \) dose not fit in memory but each individual AVC-set of \( r \) does.

- \( P_{\text{large}} = \{a_1, \ldots, a_v\} \) such that each individual AVC-set fits in memory but \textit{not} two AVC-sets
- \( P_{\text{small}} = \{a_{v+1}, \ldots, a_m\} \) & \( c \) denote the rest of the attributes, & class label respectively

- At a node \( n \) the AVC-sets of \( P_{\text{small}} \) are built in memory
- Meanwhile, the projection of \( F(n) \) onto \( P_{\text{large}} \) and class label \( c \) are written to a temporary file \( Z_n \). \( Z_n \) has the schema \(< a_1, \ldots, a_v, c > \)
- Then \( Z_n \) is scanned \( v \) times to build the AVC-sets of \( P_{\text{large}} \)

AVC-Group Size Estimation

The AVC-group of node \( n \) is estimated to the same as that of its parent \( p \) except for the size of the splitting attribute \( a \)
Experiments

The RainForest generic schema allows the instantiation of all existing classification tree algorithms without modifying the resulting tree.

The quality is an orthogonal issue and the experiments are focused on decision tree construction time.

A synthetic data generator (referred to as Generator) introduced by [AIS93]

Functions 1 and 7 from Generator are used. Function 1 generates small decision trees whereas Function 7 generates large ones.

Experiments were performed on a Pentium Pro with 200 MHz processor running Solaris X86 version 2.5.1 with 128 MB of main memory.

Algorithms were written in C++ and compiled using gcc version 2.7.2.1 with -O3 compilation option.


<table>
<thead>
<tr>
<th>Predictor Attribute</th>
<th>Distribution</th>
<th>Maximum number of entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salary</td>
<td>$U(20000, 150000)$</td>
<td>130001</td>
</tr>
<tr>
<td>Commission</td>
<td>if salary &gt; 75k, then commission = 0 else $U(10000,75000)$</td>
<td>65001</td>
</tr>
<tr>
<td>Age</td>
<td>$U(20,80)$</td>
<td>61</td>
</tr>
<tr>
<td>Education Level</td>
<td>$U(0,4)$</td>
<td>5</td>
</tr>
<tr>
<td>Car</td>
<td>$U(1,20)$</td>
<td>20</td>
</tr>
<tr>
<td>ZipCode</td>
<td>Uniformly chosen from nine zipcodes</td>
<td>9</td>
</tr>
<tr>
<td>House Value</td>
<td>$U(0.5 \cdot k \cdot 100000, 1.5 \cdot k \cdot 100000)$ where $k$ depends on ZipCode</td>
<td>1350001</td>
</tr>
<tr>
<td>Home Years</td>
<td>$U(1,30)$</td>
<td>30</td>
</tr>
<tr>
<td>Loan</td>
<td>$U(0,500000)$</td>
<td>500001</td>
</tr>
<tr>
<td>Overall size of the AVC-group of the root</td>
<td>2045129</td>
<td></td>
</tr>
</tbody>
</table>

17 MB main memory is needed to hold the AVC-group of the root node

The number of AVC-set entries fitting in memory is called buffer size

RF-Write needs a buffer size of at least 2.1 million entries; RF-Vertical needs 1.3 million ones.

Buffer size for RF-Write and RF-Hybrid: 2.5 million entries
Buffer size for RF-Vertical: 1.8 million entries
Experiments (cont’d)

Buffer size for RF-Write & RF-Hybrid: 2.5 million entries
Buffer size for RF-Vertical: 1.8 million entries
Size of input dataset: 2,000,000 records

Experiments (cont’d)

RainForest is a comprehensive approach to scaling decision tree algorithms.
The key observation is that splitting criteria can be computed using AVC-Group.
Given enough memory, RainForest algorithms outperform Sprint, the fastest scalable state-of-the-art classification algorithm.
Thanks! 😊
Questions?