High Performance Data Mining

Chapter 4: Association Rules

Vipin Kumar
Army High Performance Computing Research Center
Department of Computer Science
University of Minnesota

http://www.cs.umn.edu/~kumar
Chapter 4: Algorithms for Association Rules Discovery

Outline

- **Serial Association Rule Discovery**
  - Definition and Complexity.
  - Apriori Algorithm.

- **Parallel Algorithms**
  - Need
  - Count Distribution, Data Distribution
  - Intelligent Data Distribution, Hybrid Distribution
  - Experimental Results
## Association Rule Discovery: Support and Confidence

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Beer, Diaper, Bread, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Beer, Coke, Diaper, Milk</td>
</tr>
<tr>
<td>4</td>
<td>Beer, Bread, Diaper, Milk</td>
</tr>
<tr>
<td>5</td>
<td>Coke, Bread, Diaper, Milk</td>
</tr>
</tbody>
</table>

**Association Rule:** \( X \Rightarrow_{s,\alpha} y \)

**Support:** \( s = \frac{\sigma(X \cup y)}{|T|} (s = P(X, y)) \)

**Confidence:** \( \alpha = \frac{\sigma(X \cup y)}{\sigma(X)} (\alpha = P(y \mid X)) \)

**Example:**
\[
\{\text{Diaper, Milk}\} \Rightarrow_{s,\alpha} \text{Beer}
\]
\[
s = \frac{\sigma(\text{Diaper, Milk, Beer})}{\text{Total Number of Transactions}} = \frac{2}{5} = 0.4
\]
\[
\alpha = \frac{\sigma(\text{Diaper, Milk, Beer})}{\sigma(\text{Diaper, Milk})} = 0.66
\]
Handling Exponential Complexity

- Given $n$ transactions and $m$ different items:
  - number of possible association rules: $O(m2^{m-1})$
  - computation complexity: $O(nm2^m)$

- Systematic search for all patterns, based on support constraint [Agarwal & Srikant]:
  - If $\{A,B\}$ has support at least $\alpha$, then both $A$ and $B$ have support at least $\alpha$.
  - If either $A$ or $B$ has support less than $\alpha$, then $\{A,B\}$ has support less than $\alpha$.
  - Use patterns of $n-1$ items to find patterns of $n$ items.
Apriori Principle

- Collect single item counts. Find large items.
- Find candidate pairs, count them => large pairs of items.
- Find candidate triplets, count them => large triplets of items, And so on...
- Guiding Principle: Every subset of a frequent itemset has to be frequent.
  - Used for pruning many candidates.
Illustrating Apriori Principle

<table>
<thead>
<tr>
<th>Item</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread</td>
<td>4</td>
</tr>
<tr>
<td>Coke</td>
<td>2</td>
</tr>
<tr>
<td>Milk</td>
<td>4</td>
</tr>
<tr>
<td>Beer</td>
<td>3</td>
</tr>
<tr>
<td>Diaper</td>
<td>4</td>
</tr>
<tr>
<td>Eggs</td>
<td>1</td>
</tr>
</tbody>
</table>

Items (1-itemsets)

If every subset is considered,
\[ ^6C_1 + ^6C_2 + ^6C_3 = 41 \]

With support-based pruning,
\[ 6 + 6 + 2 = 14 \]

Minimum Support = 3

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Bread,Milk}</td>
<td>3</td>
</tr>
<tr>
<td>{Bread,Beer}</td>
<td>2</td>
</tr>
<tr>
<td>{Bread,Diaper}</td>
<td>3</td>
</tr>
<tr>
<td>{Milk,Beer}</td>
<td>2</td>
</tr>
<tr>
<td>{Milk,Diaper}</td>
<td>3</td>
</tr>
<tr>
<td>{Beer,Diaper}</td>
<td>3</td>
</tr>
</tbody>
</table>

Pairs (2-itemsets)

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Bread,Milk,Diaper}</td>
<td>3</td>
</tr>
<tr>
<td>{Milk,Diaper,Beer}</td>
<td>2</td>
</tr>
</tbody>
</table>
Apriori Algorithm

\[ F_1 = \{ \text{frequent 1-item sets} \} ; \]
\[ k = 2 ; \]
\[ \text{while}( F_{k-1} \text{ is not empty} ) \{ \]
\[ \quad C_k = \text{Apriori}_\text{generate}( F_{k-1} ) ; \]
\[ \quad \text{for all transactions } t \text{ in } T \{ \]
\[ \quad \quad \text{Subset}( C_k , t ) ; \]
\[ \quad \} \]
\[ \quad F_k = \{ c \text{ in } C_k \text{ s.t. } c.\text{count } \geq \text{minimum_support} \} ; \]
\[ \} \]
\[ \text{Answer} = \text{union of all sets } F_k ; \]
Association Rule Discovery: Apriori_generate

Apriori_generate( F(k-1) ) {
    join F_{k-1} with F_{k-1} such that,
    \( c_1 = (i_1, i_2, \ldots, i_{k-1}) \) and \( c_2 = (j_1, j_2, \ldots, j_{k-1}) \) join together if
    \( i_p = j_p \) for \( 1 \leq p \leq k-1 \),
    and then new candidate, \( c \), has a form
    \( c = (i_1, i_2, \ldots, i_{k-1}, j_{k-1}) \).
    \( c \) is then added to a hash-tree structure.
}
Counting Candidates

- Frequent Itemsets are found by counting candidates.
- Simple way:
  - Search for each candidate in each transaction. 
    Expensive!!!
Association Rule Discovery: Hash tree for fast access.

Hash Function

1,4,7
2,5,8
3,6,9

Candidate Hash Tree

1 4 5
2 3 4
3 5 6
4 5 7
5 6 7
6 8 9
7 8 9
8 9 1
9 1 2
2 3 4
4 5 7
5 6 7
6 8 9
1 2 5
4 5 8
1 3 6
3 4 5
3 5 7
3 6 7
1 5 9
3 6 8
Association Rule Discovery: Subset Operation

Hash Function
1,4,7
2,5,8
3,6,9

1 + 2 3 5 6
2 + 3 5 6
3 + 5 6

1 5 9
1 4 5
1 3 6
3 4 5
3 6 7
3 6 8
2 3 4
5 6 7
1 2 4
4 5 7
4 5 8
1 5 9
6 8 9

© R. Grossman, C. Kamath, V. Kumar
Data Mining for Scientific and Engineering Applications
Association Rule Discovery: Subset Operation (contd.)

1,4,7
2,5,8
3,6,9

Hash Function

1 + 2 3 5 6
1 5 9
1 4 5
1 3 6
3 4 5
3 6 7
3 6 8
3 5 6
3 5 7
3 5 8
6 8 9

1 2 4
1 2 5
1 2 + 5 6
1 3 + 5 6
1 5 + 6

1 4 5
1 3 6
1 5 9

1 2 4
1 2 5
1 5 9

1 2 + 4 5 7
1 2 + 4 5 8
1 5 9

Data Mining for Scientific and Engineering Applications
Parallel Formulation of Association Rules

- **Need:**
  - Huge Transaction Datasets (10s of TB)
  - Large Number of Candidates.

- **Data Distribution:**
  - Partition the Transaction Database, or
  - Partition the Candidates, or
  - Both
Parallel Association Rules: Count Distribution (CD)

- Each Processor has complete candidate hash tree.
- Each Processor updates its hash tree with local data.
- Each Processor participates in global reduction to get global counts of candidates in the hash tree.
- Multiple database scans are required if the hash tree is too big to fit in the memory.
CD: Illustration

Global Reduction of Counts
Parallel Association Rules: Data Distribution (DD)

- Candidate set is partitioned among the processors.
- Once local data has been partitioned, it is broadcast to all other processors.
- High Communication Cost due to data movement.
- Redundant work due to multiple traversals of the hash trees.
DD: Illustration

All-to-All Broadcast of Candidates
Parallel Association Rules: Intelligent Data Distribution (IDD)

- Data Distribution using point-to-point communication.
- Intelligent partitioning of candidate sets.
  - Partitioning based on the first item of candidates.
  - Bitmap to keep track of local candidate items.
- Pruning at the root of candidate hash tree using the bitmap.
- Suitable for single data source such as database server.
- With smaller candidate set, load balancing is difficult.
IDD: Illustration

All-to-All Broadcast of Candidates

Data Shift

<table>
<thead>
<tr>
<th>P0</th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/p</td>
<td>N/p</td>
<td>N/p</td>
</tr>
<tr>
<td>Remote Data</td>
<td>Remote Data</td>
<td>Remote Data</td>
</tr>
<tr>
<td>bitmask</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2,3</td>
<td>5</td>
</tr>
<tr>
<td>Count</td>
<td>Count</td>
<td>Count</td>
</tr>
<tr>
<td>{1,2}</td>
<td>{2,3}</td>
<td>{5,8}</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>17</td>
</tr>
<tr>
<td>{1,3}</td>
<td>{3,4}</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
Filtering Transactions in IDD

bitmask 1,3,5

1 2 3 5 6 transaction

1 + 2 3 5 6

2 + 3 5 6

Skipped!

1 2 4

4 5 7

1 2 5

4 5 8

1 5 9

3 4 5

3 5 6

3 5 7

3 6 7

3 6 8

2 3 4

5 6 7

1,3,5

1 2 3 5 6 transaction

bitmask
Parallel Association Rules: Hybrid Distribution (HD)

- Candidate set is partitioned into G groups to just fit in main memory
  - *Ensures Good load balance with smaller candidate set.*

- Logical processor mesh G x P/G is formed.

- Perform IDD along the column processors
  - *Data movement among processors is minimized.*

- Perform CD along the row processors
  - *Smaller number of processors is global reduction operation.*
HD: Illustration

P/G Processes per Group

All-to-All Broadcast of Candidates

IDD along Columns

CD along Rows

N/(P/G)

N/P C₀

N/P C₁

N/P C₂

G Groups of Processors
Parallel Association Rules: Experimental Setup

- 128-processor Cray T3D
  - 150 MHz DEC Alpha (EV4)
  - 64 MB of main memory per processor
  - 3-D torus interconnection network with peak unidirectional bandwidth of 150 MB/sec.

- MPI used for communications.

- Synthetic data set: avg transaction size 15 and 1000 distinct items.

- For larger data sets, multiple read of transactions in blocks of 1000.

- HD switch to CD after 90.7% of the total computation is done.
Parallel Association Rules: Scaleup Results (100K, 0.25%)
Parallel Association Rules: Sizeup Results (np=16, 0.25%)
Parallel Association Rules: Response Time (np=16,50K)

- CD
- HD
- IDD
- simple hybrid
Parallel Association Rules: Response Time (np=64,50K)
## Parallel Association Rules: Minimum Support Reachable

<table>
<thead>
<tr>
<th>Number of Processors</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Successful Down to</td>
<td>0.25</td>
<td>0.2</td>
<td>0.15</td>
<td>0.1</td>
<td>0.06</td>
<td>0.04</td>
<td>0.03</td>
</tr>
<tr>
<td>Ran out of memory at</td>
<td>0.2</td>
<td>0.15</td>
<td>0.1</td>
<td>0.06</td>
<td>0.04</td>
<td>0.03</td>
<td>0.02</td>
</tr>
</tbody>
</table>
Parallel Association Rules: Processor Configuration in HD

64 Processors and 0.04 minimum support

<table>
<thead>
<tr>
<th>Pass</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td>8 x 8</td>
<td>64 x 1</td>
<td>4 x 16</td>
<td>2 x 32</td>
<td>2 x 32</td>
<td>2 x 32</td>
<td>2 x 32</td>
</tr>
<tr>
<td># of Candidates</td>
<td>351 K</td>
<td>4348 K</td>
<td>115 K</td>
<td>76 K</td>
<td>56 K</td>
<td>34 K</td>
<td>16 K</td>
</tr>
</tbody>
</table>
Parallel Association Rules: Summary of Experiments

- HD shows the same linear speedup and sizeup behavior as that of CD.
- HD Exploits Total Aggregate Main Memory, while CD does not.
- IDD has much better scaleup behavior than DD