An Improved DFD Based on Attribute Partition Information Entropy

Liu Bohonga, Jiang Xinyuan

College of Computer Science and Technology, Chongqing University of Post & Telecommunications, Chongqing 400065, China.

liubh@cqupt.edu.cn, s160231032@stu.cqupt.edu.cn

Keywords: Functional Dependencies; Attribute Partition; Information Entropy.

Abstract: DFD is a depth-traversal functional dependencies discovery method, it does not consider association between nodes of power set lattice. We improved DFD by using attribute information entropy combined with DFD to reduce the repeated frequencies of traversals. Datasets of UCI are used to verify that the improved DFD runs faster than original DFD.

1. Introduction

Functional dependencies (FDs) is a key theory in relational database fields. Finding minimal non-trivial FDs attracted many scholars to study. TANE, FD_Mine, FUN, Fast_FD and FDEP proposed in decades. General complexity of FDs mining algorithm is $\Omega(2^m)$, where m is number of columns.

2. Related Concepts

Relational mode $R=\{X_1, X_2 \ldots X_n\}$, r is an instance consisting of $|r|$ tuples. For the tuple $t \in r$, denoted as $t[X]$.

FD: Academic degree $\rightarrow$ EDUY is a valid FD since $t_1[\text{Academic degree}]=t_2[\text{Academic degree}]$ and $t_1[\text{EDUY}]=t_2[\text{EDUY}]$. X, Y abbreviated as LHS and RHS in FD: $X \rightarrow Y$.

3. Algorithm Dfd

DFD is a FDs mining algorithm based on partition method, which recombines components from TANE and unique column combinations. UCC mining is a sub-problem of the minimal FD. For instance r, the unique column combinations must be able to uniquely determine a tuple. After DFD determines the unique attribute combination, it still traverses on the attribute power set of Figure 1 in a depth-first manner.

3.1 Attribute Partition Information Entropy

3.1.1 Theorem and Inference

For attribute sets $X, Y \subseteq R$, the partition information entropy are:

$$H(\pi_X) = -\sum_{i=1}^{n} p(\pi_i) \log_2 p(\pi_i) \tag{1}$$

This is the paragraph spacing that occurs when you use the [ENTER] key. According to (1), the information gain between attributes can be obtained:
\[ IG(\pi_Y|\pi_X) = H(\pi_X) - H(\pi_Y|\pi_X) \]

**Lemma:** \( \forall \) FD: \( X \rightarrow Y \) is valid, \( \forall \) FD: \( X \rightarrow Z \) is non-FD, then \( IG(\pi_Y|\pi_X) > IG(\pi_Z|\pi_X) \).

**Proof:** Since FD: \( X \rightarrow Y \) is valid, then

\[ IG(\pi_Y|\pi_X) = H(\pi_X) \] (2)

\[ IG(\pi_Z|\pi_X) = H(\pi_Z) - H(\pi_Z|\pi_X) \] (3)

\[ H(\pi_X, \pi_Z) = H(\pi_X) + H(\pi_Z|\pi_X) \geq \max\{H(\pi_X), H(\pi_Z)\} \] (4)

According to (2), (3), (4): Since LHS and RHS of FD are not independent of each other as a random variable, therefore \( IG(\pi_Y|\pi_X) > IG(\pi_Z|\pi_X) \).

we decide to introduce the information entropy sequence in the findLhs algorithm of DFD, we construct a set \( \{IG(X_u|X_i)| i \geq j \geq 1, n = |r|\} \) in descending order, LHS with a larger information entropy is preferentially picked for computation.

### 4. Algorithm Dfd Description

#### Algorithm Frame

**Algorithm 1 main algorithm**

**Input:** instance \( r \subseteq \text{dom}(X_1) \times \text{dom}(X_2) \times \ldots \times \text{dom}(X_n) \)

**Output:** minimal FDs set mindeps

1. foreach attribute \( X \in R : |\pi_X| = |r| \) then \( R \leftarrow R \setminus \{X\} \)
2. foreach attribute \( X' \) in \( R \setminus \{X\} : \text{add}\{X \rightarrow X'\} \text{ to minDepset} \)
3. foreach RHS \( \in R : \text{minDepset} \leftarrow \text{minDepset} \cup \{Y \rightarrow \text{RHS}'|Y \in \text{findlhs}(\text{RHS}, r)\} \)
4. return minDepsets

**Algorithm 2 compute attribute partition information entropy**

**Input:** Database instance \( R' \) after culling the unique attribute

**Output:** Attribute information gain sorting pair sequence

1. foreach attribute \( X'_i \in R' \)
2. foreach attribute \( X'_i \in R', i \neq j \)
3. \( igSeries \leftarrow \text{calculate } IG(X_i|X_j) \)
4. quickSort_MaxtoMin(IGseries)
5. return igSeries

**Algorithm 3 compute LHS**

**Input:** RHS, \( r', igSeries \)

**Output:** Minimal FDs set Mindeps

1. seeds\( \leftarrow R \setminus \max\{igSeries(X'_i|X'_j), X'_j\}\}
2. while !isEmpty(seeds) do
3. foreach node in seeds
4. computePar(node)
5. picknextNode()
6. seeds \( \leftarrow \text{nextSeeds()} \)
7. return minDeps

In Algorithm 3, algorithm preferentially selects an attribute pair with a larger value as the starting LHS in igSeries. Algorithm 4 and Algorithm 5 are the same as original DFD.
5. Experiment and Analysis

5.1 Experimental Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cols</th>
<th>Rows</th>
<th>Size(KB)</th>
<th>FDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>iris</td>
<td>5</td>
<td>150</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>balance</td>
<td>5</td>
<td>625</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>chess</td>
<td>7</td>
<td>28056</td>
<td>519</td>
<td>1</td>
</tr>
<tr>
<td>abalone</td>
<td>9</td>
<td>4177</td>
<td>187</td>
<td>137</td>
</tr>
<tr>
<td>nursery</td>
<td>9</td>
<td>12960</td>
<td>1024</td>
<td>1</td>
</tr>
<tr>
<td>Breast-cancer</td>
<td>11</td>
<td>699</td>
<td>20</td>
<td>46</td>
</tr>
<tr>
<td>bridges</td>
<td>13</td>
<td>108</td>
<td>6</td>
<td>142</td>
</tr>
<tr>
<td>adult</td>
<td>14</td>
<td>48842</td>
<td>3528</td>
<td>78</td>
</tr>
<tr>
<td>letter</td>
<td>17</td>
<td>20000</td>
<td>695</td>
<td>61</td>
</tr>
</tbody>
</table>

5.2 Algorithm Running Time Analysis

In case of a few potential FDs, DFD ran almost the same time as the improved DFD. However, when there are more potential FDs, for example the calculation of adult and letter is more efficient compared with original DFD.

In the original algorithm 4, for an LHS judged as a candidate Mindep node, LHS is a Mindep node when the set of unchecked is empty after LHS removed all subsets that can be pruned. However, when set of unchecked subsets is not empty, DFD will randomly pick an unchecked subsets of LHS as the next node, and the attribute information entropy sequence igSeries will help to pick the node that may has more potential FDs, thus improving efficiency to some degree.

Figure 2. Comparison of algorithm running time under different datasets.

5.3 Algorithm Memory Consumption Analysis

Since the original DFD has stored the partitions of all attributes when algorithm computes the igSeries sequence, the extra memory consumption is only stored in a float type array that only \((n^2-n)/2\times 4\) bit\(\ll\)1MB.
6. Conclusion

In this paper we presented an Improved DFD algorithm that improved original DFD’s random traversal strategy by computing the attribute partition information entropy sequence. This algorithm does not consider the threshold of the values of igSeries, sorted and computed all attributes. Therefore, combining the method of statistical learning methods with this algorithm is one of the feasible research directions in the future.

Acknowledgments

This work was financially supported by Chongqing Research Program of Basic science and Frontier Technology (No. cstc2016jcyjA0542) fund.

References