An Improved DFD Based on Attribute Partition Information Entropy

Liu Bohong[a], **Jiang Xinyuan**[b]

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

[a]liubh@cqupt.edu.cn, [b]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 [2][3][4]. Finding minimal non-trivial FDs attracted many scholars to study. TANE [5], FD_Mine [6], FUN [7], Fast_FD [8] and FDEP[9] proposed in decades. General complexity of FDs mining algorithm is Ω (2^m) [10], where m is number of columns.

2. Related Concepts

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

FD: Academic degree →EDUY is a valid FD since t_1 [Academic degree] =t2 [Academic degree] and t1 [EDUY] =t2 [EDUY]. X, Y abbreviated as LHS and RHS in FD: X→Y.

3. Algorithm Dfd

DFD[11] is a FDs mining algorithm based on partition method, which recombines components from TANE and unique column combinations [12]. 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.

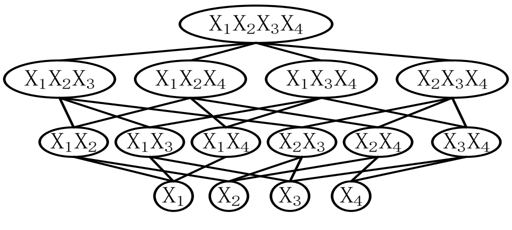

Figure 1. power set lattice R={X_1, X_2, X_3, X_4}.

3.1 Attribute Partition Information Entropy

3.1.1 Theorem and Inference

For attribute sets X, Y⊆ 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:

  DOI: 10.25236/icscbd.2018.011

$IG(\pi_Y|\pi_X)=H(\pi_Y)-H(\pi_Y|\pi_X)$

Lemma: ∀ FD: X→Y is valid, ∀ FD: X→Z is non-FD, then IG $(Y|\pi_X)$>IG$(\pi_Z|\pi_X)$.

Proof: Since FD: X→Y is valid, then

$$IG(\pi_Y|\pi_X) = H(\pi_X) \quad (2)$$

$$IG(\pi_Z|\pi_X) = H(\pi_Z)-H(\pi_Z|\pi_X) \quad (3)$$

$$H(\pi_X, \pi_Z) = H(\pi_X)+H(\pi_Z|\pi_X)\geq \max\{H(\pi_X),H(\pi_Z)\} \quad (4)$$

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 decides to introduce the information entropy sequence in the findlhs algorithm of DFD, we construct a set $\{IG(X_u|\ X_i)|\ n\geq i>j\geq 1, n=|r|\}$ 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⊆ dom(X_1) ×dom(X_2)×…×dom(X_n)
Output:minimal FDs set mindeps
foreach attribute X∈R :if $|\pi_X|$=|r| then R ← R\ {X}
foreach attribute X’ in R\{X} :add{X→X’} to minDepset
foreach RHS∈R: minDepset ←minDepset∪{Y → RHS’|Y ∈ findlhs(RHS, r)}
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
foreach attribute X'_i∈R’
    foreach attribute X'_i∈R’,i≠j
        igSeries ← calculate IG($X_i|X_j$)
quickSort_Maxtomin(IGseries)
return igSeries

Algorithm 3 compute LHS
Input: RHS,r’,igSeries
Output:Minimal FDs set Mindeps
seeds←R\ max{igSeries ($X'_j|X'_i$), X'_j)}
while !isEmpty(seeds) do
    foreach node in seeds
        computePar(node)
picknextNode()
seeds ←nextSeeds()
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 1.Dataset.

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

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.

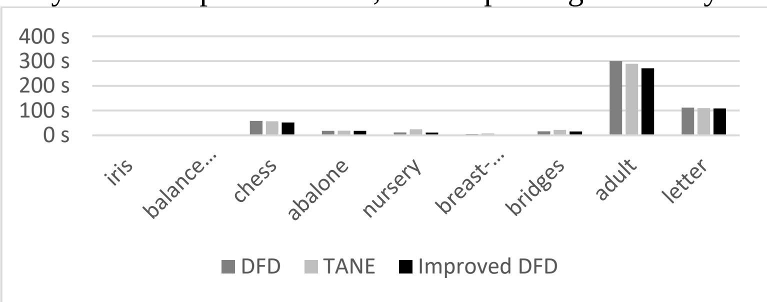

Figure 2. Comparison of algorithm running time under different datasets.

5.3 Algorithm Memory Consumption Analysis

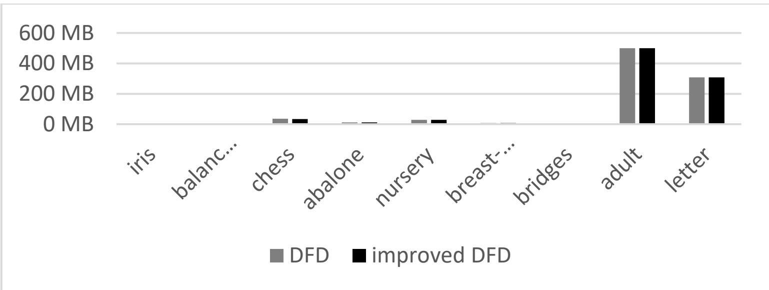

Figure 3. Comparison of DFD and improved DFD memory consumption under different datasets.

Since the original DFD has stored the partitions of all attributes when algorithm computes the igSeris sequence, the extra memory consumption is only stored in a float type array that only $(n^{2-}n)/2\times4$ bit≪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

[1] Ding Xiaoou, Wang Hongzhi et al.Relationship between multiple properties of data quality [J].Journal of Software,2016,27(7):1626-1644.

[2] E.F.Codd.A relational model of data for large shared data banks [J].Commun.ACM,1970,13(6):377-387.

[3] P. Bohannon, W. Fan, and F. Geerts. Conditional functional dependencies for data cleaning[C]. Proceedings of the International Conference on Data Engineering (ICDE),2007:746-755.

[4] Zhong Ping, Li Zhanhuai, Chen Qun.Function dependence detection method in relational data[J].Chinese Journal of Computers,2017,40(1):207-222.

[5] Nikita Bobrov1, George Chernishev,et al.An Evaluation of TANE Algorithm for Functional Dependency Detection[J].MEDI.2017:208-222.

[6] H. Yao, H. J. Hamilton, and C. J. Butz. FD Mine: discovering functional dependencies in a database using equivalences[C]. In Proceedings of the International Conference on Data Mining (ICDM).2002:729-732.

[7] N. Novelli and R. Cicchetti. FUN: An efficient algorithm for mining functional and embedded dependencies[C]. In Proceedings of the International Conference on Database Theory (ICDT).2001, 189-203.

[8] C. Wyss, C. Giannella, and E. Robertson.FastFDs:A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances extended abstract[C]. In Proceedings of the International Conference of Data Warehousing and Knowledge Discovery (DaWaK),2001 :101-110.

[9] P.A.Flach,I.Savnik.Database dependency discovery: a machine learning approach[J]. AI Communications.1999,12(3):139-160.

[10] HeiKKi Mannila,Kari-Jouko Raiha.On the complexity of inferring functional dependencies[J]. Discrete Appl.Math,1992,40:237-243.

[11] Ziawasch Abidjan, Patrick Schulze, Felix Naumann.DFD:Efficient Functional Dependency Discovery[C].Proc-eedings of International Conference on Information and Knowldege Management(CIKM),2014:949-958.

[12] Z.Abedjan and F.Naumann.Advancing the discovery of unique column combinations.In CIKM,2011:1565-1570.

[13] Y.Huhtala,J.Karkkainen,P.Porkka,H.Toivonen.TANE:An efficient algorithm for discovering functional and approximate dependencies.The Computer Journal,1999:100-111.