Research on the Structure of Computer Interaction Hash Algorithm for Mining

Zhao zhu

Hunan Communication Polytechnic, Changsha, Hunan, 410004

Keywords: data mining data; association rule mining; frequent pattern mining; interactive mining

Abstract. The association rule mining is one of the main technical data mining, the existing association rule mining algorithms based on support - confidence framework, there are many issues of double counting and traversing the database when the user adjusts the threshold. The paper maintains support for the associated threshold rule changes the problem, association rule mining algorithm interaction HIUA, the algorithm improves the pruning process of the original IUA algorithm, and improve the operating efficiency of the algorithm by Hash structure. UCI datasets experimental results and their actual financial data set showed that: in the process of support threshold changes in HIUA algorithm further use of existing mining results, effectively improve the efficiency of mining association rules.

Introduction

Association rule mining is an effective means of data mining, but in a different threshold of support and confidence threshold, dig out the pattern and number of rules often differ, in practical applications, often need to constantly adjust the threshold to get the desired size rule set, a process that is required to interact with the program. In view of this situation, there has been excavated scholars use information, including the proposed IUA, LIUA, QIUA, IIUA etc. for interactive mining algorithm and improved algorithm.

Association rule mining has been carried out in the practical application of marketing, insurance and other industries and achieves better results. In recent years, with the development of data mining technology and data warehouse technology, there are many scholars association rule mining in the stock forecasting, financial analysis and the application of research and have achieved some results. In this paper, the main consideration when association rules support threshold changes when the update problem, a structure based on interactive Hash mining algorithm HIUA, in order to improve the original IUA algorithm pruning process, it is desirable to improve the efficiency of the algorithm by Hash structure. And apply the improved algorithm HIUA UCI datasets and their actual financial data set.

Problem Description

Most association rule mining algorithms are based support - confidence framework, basic methods include algorithm based on Apriori as the representative of a candidate with the test pattern generation algorithms and to FP-Growth-based model represented growth. IUA algorithm is proposed for interactive mining, which is based on Apriori algorithm and using the excavated information to improve mining efficiency. The basic idea of Apriori, FP-Growth, and IUA algorithm is described below.

Apriori algorithm

Apriori algorithm first established support-confidence framework, its basic approach is to generate all the length of the frequent pattern set by iteration.

(1). Apriori algorithm Apriori property (Property 1), namely frequent pattern of all sub model is frequent, namely the anti-monotony of frequent patterns.

(2). Let length k frequent pattern set, the candidate pattern set respectively L_k, C_k, the Apriori algorithm first generates L_1, then iterate: if L_{k-1} is not empty, C_k is generated by splicing L_{k-1} in a

specified pattern, and L_k generated by shearing C_k.

FP-Growth algorithm

Because the candidate mode algorithm will generate a lot of candidate pattern set, and the need for frequent scanning the database, resulting in lower dense datasets or support threshold algorithm based on performance degradation, so the officer proposed pattern-based growth FP- Growth algorithm [1]. The algorithm main contents are:

(1). The introduction of an extended prefix tree FP-tree structure to save the data set information, construct FP-tree simply scan data sets twice, and how often each path of the root to leaf nodes in decline, making the tree more compact, and conducive pattern generation algorithm of FP-tree of the split.

(2). we present a frequent pattern based on FP-tree generation algorithm. It is a divide and conquer algorithm that mode suffixes for FP-tree split by mode segments, namely the condition FP-tree, recursively repeat this operation for each split, and by extension in the recursive and recursive splicing process is terminated When generating frequent patterns and stitching on the combination of a single path.

HUA algorithm description

Let DS for the data set, SptThr to support threshold, CfdThr for the confidence threshold, Ptni of length i of the pattern, Ptn_{ki} to Ptn_k in paragraph i, C_i, L_i are the length of i candidate pattern set frequent pattern set , LOrg original support for the next set of frequent patterns, LNew frequent pattern set by the new support at the difference between L_1 and $LOrg_1$ generated, Rules for the rule set, Hash Table of Hash table, Base as the base, then HIUA pseudo-code:

(1) HashFunc

input：Ptn_k

output：value of Hash is HashCode

HashFunc(Ptn_k) {

P=1；HashCode＝Ptn_{k1}；

for i=2 to k {P＊=Base；HashCode+= Ptn_{ki}＊P； }

Return HashCode% TableSize； }

(2) GetSupportCnt

input：Ptn_k

output： SuppotCnt of Ptn_k

GetSupportCnt（Ptn_k）{if HashFunc（Ptn_k）is not existed in HashTable {Scan DS to get SupporCnt of Ptn_k HashTable.add（HashFunc（Ptn_k）, SupporCnt）； }

Return HashTable［HashFunc（Ptn_k）］； }

(3) IUA_Gen

input：k，LOrg，LNew

output：L^3_k

IUA_Gen（k，LOrg，LNew）{for（i=1 to k）{ foreach Ptn_i in $LOrg_i$ {foreach Ptn_{k-i} in $LNew_{k-i}$ { C_k. Add（Concat（Ptn_i，Ptn_{k-i}））； } } } return select Ptn_ks from C_k where each Ptn_{k-1} in Ptn_k exist in L_{k-1}； }

(4) HIUA

input：LOrg，DS，SptThr

output：new Rules

HIUA（LOrg，DS，SptThr，CfdThr）{if（SptThr went up）{

L=select Ptns from L where Ptn.Support＞=Spt-Thr； }

if（SptThr went down）{L_1=Select Ptn_1 from DS where GetSupportCntBy-Hash(Ptn_1) ＞＝SptThr；$LNew_1=L_1-LOrg_1$；

for（k=2；L_{k-1} is not null;k++）L_k=IUA_GenL_k（k，Lorg，LNew，DS，SptThr）；
L=Union（L_1，L_2，…，L_k）；｝
If （SptThr changed）Rules = GenerateRules（L，CfdThr）；
If （CfdThr changed）Rules =select Rules from Rules where Rule.Cfd＞＝CfdThr；
Return Rules；｝

Algorithm Analysis

Hash collision occurs when the value of paper, list treatment, ie the same mapping Hash values stored in the first node points to Hash-Table of the list. For example: Let presence mode {1,2,6}, {1,2,9}, {1,2,12}, {4,9,11}, corresponding support count as 10,15,20,20 , Hash value of 1,4,7,4, the mode is sequentially added to the above Hash table, resulting Hash table in Figure 1.

Find and maintenance method GetSupportCnt step Hash table is as follows: First, look for the calculation model Hash values and determine whether this Hash value corresponding to the list this mode; if not present, scan data set to give its support count, and then the model and support count added in pairs into the corresponding list; and finally returned Hash tables corresponding results.

Hash tables introduced so that the program can save interaction obtained support count, and can be approximated O (1) time complexity directly read from memory, thus avoiding redundant data sets and data sets larger scanning when database I / O overhead, reducing the running time of the program.

Since HIUA saved during the execution of the algorithm obtained support count, further use of the results that have been excavated, and the use of fast-access support structures Hash count, compared with IUA reduced data set scanning overhead and to further improve the efficiency.

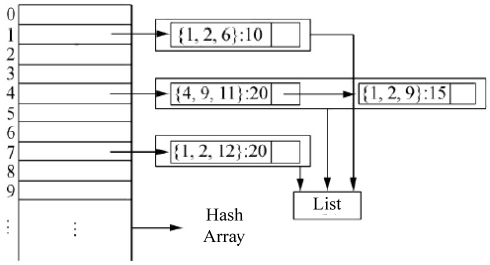


Fig 1.Hash table structure

Performance Analysis

This article compares Apriori, IUA and HIUA run time performance analysis, the above algorithms by C # (. Net 4.0) achieved operating environment for i5 processor, 2G memory, Win7 system.

Since in the case of a confidence threshold change or support threshold increase does not require frequent updating pattern set, IUA and HIUA clearly faster speeds, so this support declined only relatively frequent pattern set of different algorithms to calculate the running time of the case [2]. Where in the data set 1-5 from the UCI standard data, data sets, 6 and 7 for the enterprise actual financial data.

(1). Data collection for the retail, a total of 88,000 records, each 1-50 Transaction ID range. Hash function radix 13, Hash table length of 100,000, the average probability of collision Hash value is 0%. Comparative efficiency of the algorithm shown in Figure 2, where X coordinate support threshold, from .05 to 0.03 in steps of 0.02, Y coordinates for the run time calculation frequent patterns set.

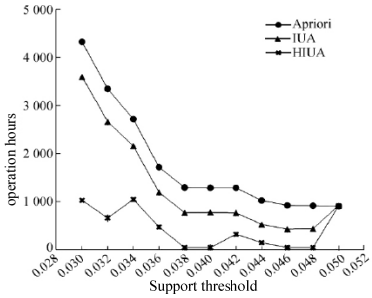


Fig 2.algorithm efficiency comparison on retail data set

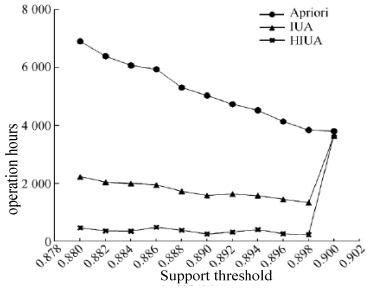


Fig 3 .performance comparison data set on the chess algorithm

(2). Data collection for the chess, a total of 3,200 records, each 37 transaction ID. Hash function radix 13, Hash table length of 100,000, the average probability of collision Hash value is 1.06%. Comparative efficiency of the algorithm shown in Figure 3, where X coordinate support threshold, from 0.9 to 0.88 in steps of 0.002, Y coordinates for the run time calculation frequent patterns set.

(3). Data collection for the accidents, a total of 5,000 records, each 20-40 Transaction ID range. Hash function radix 13, Hash table length of 100,000, the average probability of collision Hash value of 0.34%. Comparative efficiency of the algorithm shown in Figure 4, where X coordinate support threshold, from 0.8 to 0.7, in steps of 0.01, Y coordinates for the run time calculation frequent patterns set.

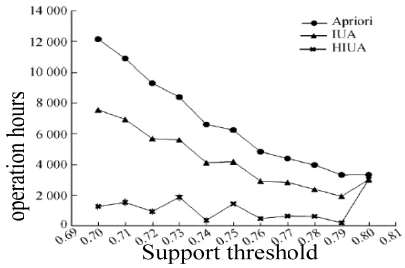


Fig 4.Performance comparison of data on accidents Set Algorithm

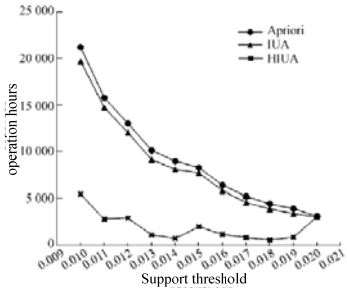
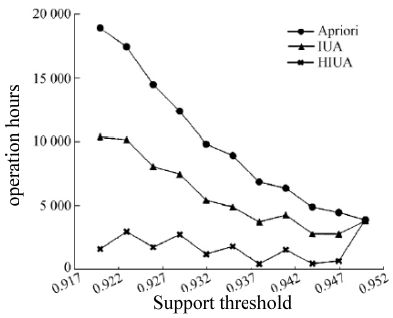


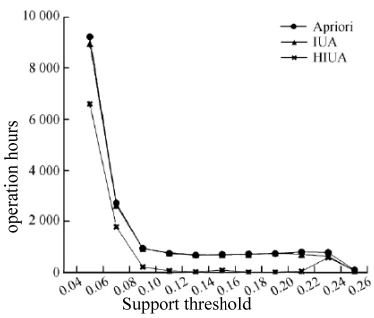
Fig 5.performance comparison data set algorithm on kosarak

(4). Data collection for the kosarak, total 10,000 records, each transaction ID 1-100 range. Hash function radix 13, Hash table length of 100,000, with an average probability of collision Hash value was 3.68%. Comparative efficiency of the algorithm shown in Figure 5, wherein the X coordinates

Labeled support threshold, from .02 to .01, in steps of 0.001, Y coordinates of running time calculation frequent patterns set.

(5) The data set is pumsb, a total of 5,000 records, each transaction ID 1-100 range. Hash function radix 13, Hash table length of 100,000, with an average probability of collision Hash value was 3.68%. Comparative efficiency of the algorithm shown in Fig 6, where X coordinate support threshold, from .95 to 0.92 in steps of 0.003, Y coordinates for the run time calculation frequent patterns set.





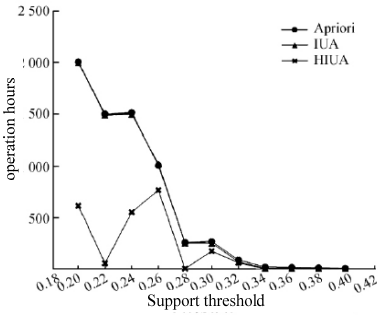


Fig 6. Pumsb dataset algorithm performance comparison

Fig 7.Fin_ratio dataset algorithm performance comparison

Fig 8.Performance comparison data set algorithm on Fin forecast

(6) The data collection for the retail industry index 73 per share, profitability, solvency category, management, growth capacity, cash flow, capital structure and other seven categories of 14

indicators in four quarterly property (fin_ratio), a total of 620 records. All records in the dataset will be clustered into three categories, and each property into six discrete intervals [3]. Comparative efficiency of the algorithm shown in Fig 7, where X coordinate support threshold, from 0.25 to 0.05 in steps of 0.02, Y coordinates to calculate the frequent pattern set running time can be further concentrated extracts from mode regulation correlating the analysis of financial indicators.

(7) The data set for a TV sales business four years a total of 35 months of cash, bank deposits, fixed assets, profit for the year, current assets and other financial indicators 27 record (fin forecast), the data set is converted to a record of each financial indicators data for three consecutive months, then each property into six discrete intervals [4]. Rules algorithm efficiency comparison shown in Fig 8, where X coordinate support threshold, from 0.4 to 0.2, in steps of 0.02, Y coordinates to calculate the frequent pattern set running time can be further concentrated extract patterns from the constraints were in line with forecast predictive analysis of financial indicators.

You can see from the above comparison, IUA first execution time approximately IUA, along with running constantly using the saved support count, HIUA efficiency has improved significantly compared with IUA. In addition, when the support threshold is reduced, local frequent pattern set in little change is possible, then IUA and HIUA candidate frequent pattern set Cik will be smaller, and they can be found in the part of the pattern in HIUA Hash table, so as shown in the above comparison, there will be some IUA, HIUA when support threshold fall, the running time but will decline.

Conclusion

In this paper, the problem associated with the interactive update rules were discussed and research, we propose a Hash-based interactive structure mining algorithms HIUA, to deal with the same database and minimum support association rule changes when the update in question. HIUA algorithm pruning original IUA algorithm has been improved, and quick access to support count by using the Hash structure, reducing the overhead of scanning the data set, thereby increasing the efficiency of the algorithm. Focus on the experimental data sets in UCI and corporate financial data, the results showed that: The algorithm uses the existing association rule mining results improved mining efficiency, the algorithm has better performance.

Reference

[1] XU YUE．Reliable representations for association rules[J]．Data & Knowledge Engineering,2011,70(6)：555-575

[2] Muata K，Bryson O.A context-aware data mining process model based framework for supporting evaluation of data mining results [J]. Expert Systems with Applications, 2012, 39(1): 1156-1164.

[3] LIN Xiaoyong．Share-inherit: A novel approach for mining frequent patterns[C]// The Proceedings of 8th World Congress on Intelligent Control and Automation. Jinan, China: IEEE Press, 2013: 2712-2717.

[4] Aaron C．Association mining [J]．ACM Computing Surveys, 2012, 38(2):1-42.