ATLANTIS PRESS
Advances in Computer Science Research, volume 83
8th International Conference on Social Network, Communication and Education (SNCE 2018)

Association Rule Mining Algorithm of Transposed matrix Based on Python Language

Shaoyun Song[1, a]

[1] School of Mathematics and Information Technology, Yuxi Normal University, Yunnan, China

[a]ssy@yxnu.net

Keywords: Transposed matrix; Association rules; Data mining; Python language; NumPy

Abstract. Apriori and its improved algorithms can be generally classified into two kinds: SQL-based and on memory-based. In order to improve association rule mining efficiency, after analyzing the efficiency bottlenecks in some algorithms of the second class, an improved efficient algorithm for Python language is proposed. Two matrixes are introduced into the algorithm: one is used to map database and the other to store frequent 2-itemsets related information. Through the operation of two matrixes, its time complexity and space complexity decrease significantly. The experiment indicates that the method has better performance.

Introduction

1993 R.Agrawal et. Proposed the basic concepts of association rule mining, in 1994 proposed the Apriori algorithm, which is based on two stages frequent sets recursive algorithm. Apriori algorithm has the following disadvantages: First, a large set of candidate and there is a lot frequent itemsets, take up a lot of memory space; second, repeatedly scanning the database, take up a lot of CPU time, so that the speed increases greatly reduced along with the data. J.Park [8] proposed DHP algorithm of using hash generate candidate itemsets, it is Apriori algorithm for direct improvement; Savasere etc. [4] The Partition algorithm effectively reduces the memory requirements while increasing parallelism, H. Atoivonen [9] proposed Sampling algorithm for scanning results before analyzing the information obtained through the analysis of samples some of the rules may be established in the database, the use of sampling to verify the rest of the earlier results, reducing the I / O costs and greatly reducing the number of database scans. FP-tree algorithm [10] will find long frequent patterns problem into a recursive find some short mode, and then connect the suffix. The algorithm does not generate candidate itemsets, nor tested candidate itemsets, and only twice scanning database. Algorithms for mining long and short frequent patterns are very effective and can scale. However, the database is large, the structure of FP-tree-based memory is unrealistic. BitMatrix algorithm is only the matrix as the database data storage form, there is no direct calculation using matrix properties, the algorithm is still based on Apriori algorithm ideas, the implementation process is still to produce a large number of candidate itemsets and needs at all levels in order to obtain frequent item sets..

The Analysis of Current Matrix Algorithm

References [1] proposed algorithm based on matrix for mining association rules, Apriori algorithm will prune the matrix linking, this method only needs to scan the database once, and no candidate set, you can get frequent set, thereby greatly improve the efficiency of the algorithm in generating association rules, the use of the basic properties of probability theory, but also greatly reduces the amount of computation. However, the algorithm generates frequent sets K-term is still used Apriori algorithm, reducing the efficiency of the implementation.

References [1] proposed an algorithm, construct Boolean m×n matrix with m and n tasks projects, the algorithm need to construct a row vector of two-dimensional whose elements are "1". For getting the number of support with the two-dimensional line vector and the two candidates sets each entry consisting of vector inner product operation. Compared with the support resulting in frequent two sets;

frequent 2 sets 3 sets connected to generate candidate before the frequent two sets once cut, the use of inferences drawn deleted cannot become frequent three sets of elements of the project, which will reduce the generation candidate three sets scale. And so on, in the first k-itemsets seeking the support when the k-dimensional vector construct items with each candidate set consisting of k vector inner product operation, while connecting to generate k + I term candidate sets, the first item on the set of k was cut. The algorithm requires a lot of inner product operation, and the need to compare with the support resulting in frequent sets, consumes a lot of CPU time and memory space.

References [3]. First, Scan the database to generate the corresponding initial matrix; Secondly, Pruning the matrix use the resulting frequent item set size k; Then directly generate k frequent item sets, multiplied by the matrix vector sum k item set to be generated to get the support. Obtained in the algorithm does improve the efficiency of k frequent itemsets, but the matrix can also be more efficient pruning.

References [4] algorithm introduces two matrixes, a matrix for mapping databases, one for storing 2-frequent item sets related information. The algorithm tailoring transaction matrix, improves the efficiency of the algorithm. But in seeking frequent sets K-entry is required when a mapping of K items in the matrix column vector inner product operation to do, reducing the efficiency of the algorithm.

Basic Concepts

Association Rules

Association rules: assume $I=\{I_1,I_2,\cdots,I_n\}$ is a set of n different items. Transactional database is $D=\{T_1,T_2,\cdots,T_m\}$. Among $T_i=\{I_{i1},I_{i2},\cdots,I_{ik}\}$ and $I_{ij}\in I$ .The implication $X\Rightarrow Y$ is association rule. Among $X,Y\subset I$ and $X\cap Y=\Phi$.

Item set support: Supported number is the number of transactions X include in D,denoted $Count(X)$. Assume $|D|$ is the total number of transactions in D. Then $Sup(X)=Count(X)/|D|$ called item set support of X .

The support number and Metrics of association rules: Number of transactions called association rules support numbers of support item set $X\cup Y$ in database D.Denoted $Count(X\Rightarrow Y)$.The $Count(X\Rightarrow Y)/|D|$ called rule Support Metrics, Denoted $Sup(X\Rightarrow Y)$.

Items set: Contains the item for the k-th set is called k-item set.

K-item set: meet the minimum support k-itemsets

Strong association rules: Members meet minimum support threshold and minimum confidence threshold rules.

Association rule mining is to discover meet a given minimum support and minimum confidence of all the conditions of the implication that the strong association rules.

Bit Matrix

Let a transaction database with m records of transaction and different n items, for building a m+1 rows and n+1 columns of a matrix $M_{(m+1)\times(n+1)}$. And each element of the matrix is initialized to 0. Scan the transaction database and calculate transaction matrix M ,

Among $M_{ij}=\begin{cases}0, & I_j\in T_i\\ 1, & I_j\notin T_i\end{cases}, i=1,2,\cdots,m;\ \ j=1,2,\cdots,n$

After the transaction matrix M of calculating called transaction bit-matrix. The last column of the matrix $M_{i(n+1)}$ is the number of i-th line with "1", which means that the length of the transaction T_j. That T_j contains the number of items. The last line of the matrix $M_{(m+1)j}$ is the number of J-th column with "1". I_i Expressed support for the project number.

Algorithm Based on Transposing Matrix (ABTM)

Algorithm Design

T Let the transaction database are: $D=\{T_1,T_2,\cdots,T_n\}$,Item set as follows: $I=\{I_1,I_2,\cdots,I_m\}$, Minimum support is min-sup, the minimum confidence is min-conf.

Algorithm is as follows:

Construction transaction bit-matrix and get 1-item frequent sets.

Created transaction bit-matrix M of n+1 rows and m+1 columns by the $D=\{T_1,T_2,\cdots,T_n\}$ and $I=\{I_1,I_2,\cdots,I_m\}$. The last line of the matrix where the value of the item is the support number of columns of greater than min-sup where all the items on the formation of 1-Item frequent sets . When the value of the last column of the length of the transaction, which includes the number of entries.

Prune transaction matrix M

The last line of the M value is smaller than the min-sup column will be deleted and deleting the last column value is less than k (k> = 2) rows, then recalculation of M and last column of the last row of values.

Calculate 2-item frequent sets using the transfer matrix

$L_{m\times m}=M^T_{m\times n}\times M_{n\times m}$, The upper triangular matrix are the support value of frequent 2-items, the values on the main diagonal are the support number of 1-Item frequent set. According to an upper triangular matrix where the value is greater than min-sup ranks portfolio available 2-items frequent sets.

Prune 2-frequent item sets, generate frequent k-item sets (k> 2).

Definition 1: If k-itemsets there is an $j\in X$ in $X=\{i_1,i_2,\cdots,i_k\}$, among $|j|$ is the number of frequent itemsets of L_k contains j ,and $|j|<k$. In the frequent k-itemsets generated (K +1)-item set, you can prune the data.

Proof: Assume X is k +1 frequent item sets .Then K+1 is a subset of K in L_k and in the generated k subset of k+1, $j\in X$ of each item appears k times. Given any $j\in X$ has $|j|>k$. Therefore, even if such $|j|<k$. It will no longer produce K+1-itemsets.

According to prune of the above properties on the k-item frequent sets: First, the matrix $L_{m\times m}$ is processed. We first import the NumPy module into the Python using "import NumPy as np". The diagonal and lower triangular matrix element is set to 0, you can use the model NumPy of Python function np.triu(L,1) to achieve, if the value is greater than min-sup of the upper triangular matrix element then set to 1, otherwise set is 0, you can use NumPy function np.int64 (L> = minsup) implementation. Second, calculate the matrix in each column and row numbers of 1, can be used with NumPy function A=sum(L)+sum(L.T) implementation.

Delete the columns which value is less than k in L, using NumPy function np.delete(L,n,axis=1) to achieve, where n is the column number to be deleted. According matrix L of a combination of the line can get k-frequent sets. Their support number obtained from columns inner product of bit-matrix according to the where transactions.

These algorithms do not need to generate a lot of candidate set. It has high time efficiency and space efficiency in mining frequent sets; the following examples are given and analyzed.

The above algorithm does not need to generate a large number of candidate sets, mining frequent sets of time efficiency and space efficiency is higher, the following examples are given and analyzed. Data from many years of the school Academic Affairs Department's academic performance data, have been processed to obtain part of the data for calculation. Set the score equal to 60 in the bit matrix to 1 and the score less than 60 points to 0. The various subjects said: high number Ⅰ-A, high number Ⅱ-B, computer network-C, data structure-D, C language-E, routing and exchange-F. As shown in Table 1.

Table 1 student scores in some subjects

Student ID	Name	A	B	C	D	E	F
2010093101	Zhan Sa	64	88	64	69	53	87
2010093102	Liu Ma	68	79	85	89	92	74
2010093103	Li Ming	77	66	87	92	92	79
2010093108	Wan Han	61	62	84	83	83	88
2010093109	Song Er	80	85	84	86	82	72
……	……	……	……	……	……	……	……

Practical Examples of Research

A transaction database D, where there are five transactions, T1={A,B,C}, T2={A,C,D}, T3={A,B,E,F}, T4={A,B,C,D,F}, T5={B,C,D,E}. Item I={A,B,C,D,E,F}. Minimum support is 30%, which is the minimum number of support 30% ×10=3. The following sets using Python frequent mining.

$$M = \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & E & F & \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 5 \\ 0 & 1 & 1 & 0 & 1 & 1 & 4 \\ 0 & 1 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 1 & 1 & 0 & 3 \\ 1 & 0 & 1 & 0 & 1 & 1 & 4 \\ 0 & 1 & 1 & 0 & 1 & 1 & 4 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 4 \\ 0 & 0 & 1 & 1 & 1 & 0 & 3 \\ 0 & 0 & 0 & 0 & 1 & 1 & 2 \\ 2 & 5 & 7 & 4 & 8 & 6 & \end{bmatrix} \end{array}$$

Fig. 1 Initialization bit-matrix

Construction transaction bit-matrix and get 1- item frequent sets

The algorithm is designed and constructed 11 ×7 bits-matrix in accordance with the above transaction, item-set of column is to sort by the dictionary, row are transaction records. Shown in Figure 1. Using Python to achieve the following functions:

M=np.row_stack((m,sum(m))) to achieve the value of sum of the last line. Where n is the number of the last row.

M =np.column_stack(m, sum(m.T') to achieve the final value of a sum. Where m is the last number of the column.. In this case n = 11, m = 7.

By the value of last line compared with the minimum support to obtain 1-item frequent sets.

$L_1 = \{B, C, D, E, F\}$,

Prune transaction matrix M, seeking 2-item frequent sets

According to the last line of transaction bit-matrix M, since the first column (Item A) values less than min-sup, so deleting the first column, and the last one in line 7 of the value is less than 2, so deleting the 7th line, that things T7 . Recalculate the last row and the last column to get things trimmed matrix M. Shown in Figure 2.

$$M = \begin{array}{c} \begin{array}{cccccc} B & C & D & E & F & \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 4 \\ 1 & 1 & 0 & 1 & 1 & 4 \\ 1 & 1 & 0 & 0 & 0 & 2 \\ 0 & 1 & 1 & 1 & 0 & 3 \\ 0 & 1 & 0 & 1 & 1 & 3 \\ 1 & 1 & 0 & 1 & 1 & 4 \\ 1 & 0 & 1 & 1 & 1 & 4 \\ 0 & 1 & 1 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 & 1 & 2 \\ 5 & 6 & 4 & 8 & 6 & \end{bmatrix} \end{array}$$

Fig. 2 Prune transaction matrix

Using the transfer matrix calculation of 2-item frequent sets

According to $L_{m\times m} = M^{T}_{m\times n} \times M_{n\times m}$ Obtain 2-item frequently matrix (need not the last row and the last one), the matrix values in the upper triangle is composed of 2-frequent item set support number, the value on the main diagonal pm 1-item frequent set support number. According to an upper triangular matrix where the value is greater than min-sup ranks portfolio available 2-items frequent sets. Shown in Figure 3.

$$L_k = \begin{bmatrix} 5 & 3 & 2 & 4 & 4 \\ 3 & 6 & 2 & 5 & 3 \\ 2 & 2 & 4 & 4 & 2 \\ 4 & 5 & 4 & 8 & 6 \\ 4 & 3 & 2 & 6 & 6 \end{bmatrix} \quad L_k = \begin{array}{c} \begin{array}{ccccc} B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{array}{c} B \\ C \\ D \\ E \\ F \end{array}$$

Fig. 3 get the upper triangular matrix of bits

$$L_k = \begin{array}{c} \begin{array}{cccc} \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{array}{c} \mathrm{B} \\ \mathrm{C} \\ \mathrm{D} \\ \mathrm{E} \end{array}$$

Fig. 4 k-item frequently set

According to 2-frequent matrix of following operation to get the upper triangular bit-matrix.

$L_k = triu(triu(L_k) - tril(L_k))$, $L_k(L_k < \min\sup) = 0$, $L_k(L_k >= \min\sup) = 1$

Prune 2-frequent item sets, generate frequent k-item sets (k> 2)

By NumPy function np.sum(Lk)+np.sum(Lk.T) computer Lk, the number of each item 1, the calculated value obtained in this example as following:

{3,3,1,4,3}, where 1<k, so deleting items where the row and column D, to obtain a new matrix on the k-item frequently set, Shown in Figure 4.

The matrix of Figure 4 k-1 inner product line, the calculation result is equal to k-1 rows and columns of the project and project portfolio can get frequent k item set.

This column 3-item frequent sets as follows: $L_3 = \{BEF, CEF\}$, In the absence of k is equal to k-row after row plot, so operation ends here.

Performance comparison and experimental results

Performance Comparison

The proposed algorithm of transposed matrix only needs to scan the database once, then you can get 1-frequent item sets, multiplied by the matrix transpose can obtain 2-frequent item sets, without going through the calculation of candidate sets, reducing the database read time. Through to 2-iterm frequent sets prune and inner product operation can get frequent k-item set. ABM algorithm is an efficient algorithm for mining association rules, In comparison, ABTM is a matrix mapping database, rather than creating a single item bit vector, although both algorithms database mapped into memory space occupied by the same, but the matrix storage can be achieved on the scale of the effective prune, as the item set dimension growth, ABTM algorithm matrix of rows and columns to delete useless, making the

memory space to be released gradually, while the length of the vector involved in computing the shorter the amount of computation is smaller. ABTM algorithm can save storage airborne.

Experimental results and analysis

Algorithm using NumPy of Python language, data were Mashroom data sets, in order to compare the different degrees of support for the time spent, the experimental machine's CPU is the Inter Pentium3.0GHz and memory of 2GB. Give its run-time for Apriori algorithm, matrix algorithms and ABTM algorithms, testing for different support shown in Figure 5.

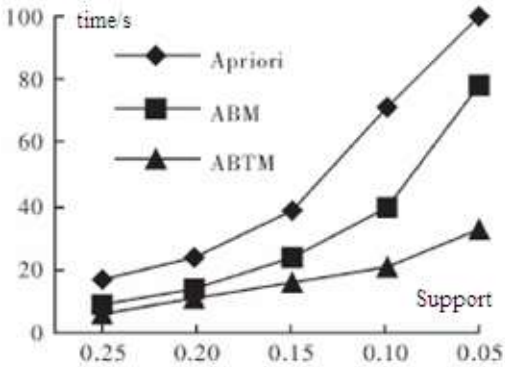

Fig. 5 Different run-time support algorithms

As ABM and ABTM algorithm generate the candidate set differs depending on the speed and size of the generated number, and the number of attributes directly affects the generation of the candidate set, set the support is fixed to 0.15, the number of attributes for different respectively, statistics algorithms used time, drawing in Figure 6.

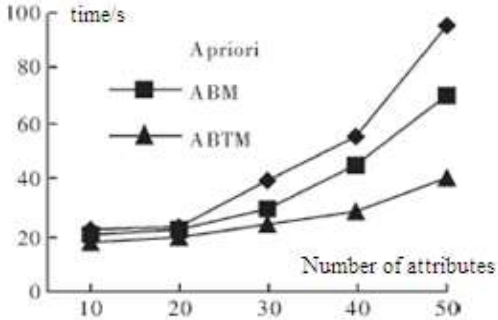

Fig. 6 the number of attributes on the algorithm

Figure 6 show ABTM algorithm significantly less than the time required to run Apriori algorithm and ABM algorithms. Figure 5, with the support set smaller, Apriori algorithm and the total amount of computation ABM algorithm was charge with support linear growth, the required time is greatly increased, and as ABTM algorithm to scan the database only once, reducing the time to read the database and frequent k-item set production through extension set and the vector inner product operation to solve, no other operations connected with the support reduced the time required for the algorithm does not increase significantly. Figure 6, along with the increase in the number of attributes, Apriori algorithm and the ABM algorithm running time rising rapidly, while ABTM was extended inner product vector collection of items and not because of an increase in the number of attributes being greater impact.

In the case of smaller support, matrix transpose algorithm increasingly obvious advantages, as compared with the Apriori algorithm, the running time to obtain a greater degree of reduction in the increased number itemsets, while scanning the database only once, is to save costs. Experimental results show that with the increase of the transaction record, Apriori algorithm will spend a large overhead to scan multiple databases, and the introduction of a transpose matrix algorithm only needs to scan the database once, by contrast, it can be seen the advantage of the transposed matrix algorithm.

Conclusions

The proposed algorithm introduces two matrices, respectively, from the time and space to optimize the efficiency of the algorithm is significantly improved, is an easy and feasible better algorithms. Algorithm can also be done in the following two further optimization:

Since the matrix structure of things changed after the play car 1 - frequent item sets generated, it is possible to construct two support matrix mapping matrix once before clipping.

Since there may be some items is not 1 - frequent item sets, these items no longer appear in the higher-level frequent item set. Therefore, two support matrix structures are not necessary to consider these items.

Typical frequent itemset mining algorithm Apriori algorithm and correlation matrix analysis, we propose a new transposed matrix-based frequent itemset mining algorithms, and a new algorithm to do a detailed analysis and description. Theoretical and experimental results show that the algorithm performance arm Apriori algorithm and the matrix algorithm superior.

References

[1] Song Shaoyun; Zhang Baohua: The New Fast Algorithm Based on Transposed Matrix for Frequent Sets Mining of Association Rule[D], Thesis. Applied Mechanics and Materials Vols. 457-458 (2014) pp 992-997

[2] Tang Ping. Association rules algorithm based on matrix and Research and Improvement of Apriori algorithm [D]. Thesis. Southwest Jiaotong University. 2009-10-01

[3] Ding Yanhui; WANG Hongguo; clever; Gu Jianjun A matrix-based new algorithm for mining association rules [J]. Computer Science .2006-04-25

[4] Cao Fenghua Improved two matrices based association rule mining algorithm Electronic Technology [J] .2012-05-15

[5] Wang Bosheng; LIU Han-bing; Jin books and; Mali Yan-based matrix association rule mining algorithm. Microcomputer Information [J]. 2007-05-25

[6] Wang Juanqin; Li Shuqin based association rule mining algorithm Matrix Research and Improvement. Computer Measurement & Control [J]. 2011-09-25

[7] Lu Tao Xia; LIU Pei-yu A strong association rules based on matrix generation algorithm. Computer Application Research [J]. 2011-4-20

[8] Park JS, Chen MS, Yu P S. An effective hash-based algorithm for mining association rules [C]. Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California: ACM Press, 1995: 175 a 186.

[9] Tovionen. H.Sampling large databases for association rules [C]. Proc of 22th VLDB Conference, Bombay, India, 1996: 1 a 12.