Mining algorithm for weighted FP-tree frequent item sets based on two-dimensional table

Yuanyuan Li, Shaohong Yin*

School of Computer Science and Technology, Tianjin Polytechnic University, Tianjin, 300387, China
yinsh@tjpu.edu.cn

Abstract. FP-growth algorithm is a classic algorithm of mining frequent item sets, but there exist certain disadvantages for mining the weighted frequent item sets. Based on the weighted downward closure property of the weighted model, this paper proposed that the weighted support was recorded in the two-dimensional table. This method has saved the process to search for the first conditional pattern base by traversing the weighted FP-tree, improved the efficiency of the weighted frequent item sets generated.

1. Introduction

Association rule mining is one of the important research topics in data mining. It aims to explore large transaction databases and reveals the implicit relationships among data attributes. The concept of association rule was first proposed by Agrawal et al. in 1993. Through the measurement framework of support and confidence, the association rule mining is simplified to the discovery of frequent item sets. The Apriori algorithm[1] was proposed by Agrawal et al. in 1994, but it needs to scan the database multiple times and generate a large number of candidate sets. Therefore, in 2000, Han et al. proposed the FP-growth algorithm[2] based on FP-tree data structure for the deficiencies of the Apriori algorithm. The algorithm only needs to scan the database twice to avoid generating a large number of candidate sets.

The above two classic algorithms treat every transaction equally, but in real life, the importance of every transaction is different and unevenly distributed. In response to this problem, scholars proposed a weighted association rule mining algorithm, but the original weighted model[3] broke the weighted downward closure property, making the mining algorithm more complicated and time consuming. In 2003, Tao et al. proposed a weighted model[4] that preserved “weighted downward closure property”. The efficiency of the algorithm is improved. Many scholars have conducted in-depth research on this model and proposed many weighted models.

In this paper, Chen Wen's weighted FP-tree model[5] proposed in 2012 is used to study the FP-growth algorithm. Based on this weighted model, method of introducing a two-dimensional table to record weighted support is introduced. We propose a new weighted FP-tree frequent item sets mining algorithm based on two-dimensional table.

2. Weighted model

The project set I = {i1, i2, ..., ik} is a set of k different items. The transaction database D={T1, T2,..., Tn}, where every transaction Ti(i = 1, 2, ..., n) contains a unique transaction identifier TID and a subset of I.
Definition 1 Let each item \( i(j=1,2,\ldots,k) \) in item set \( I=\{i_1,i_2,\ldots,i_k\} \) have a weight \( W(j) \), \( 0 \leq W(j) \leq 1 \). The project item set \( X \) also has a corresponding weight, which is recorded as \( W_X \). The weight of the transaction \( t \) in the transaction database \( D \) is denoted by \( W_T \). The weighted support for association rule \( A \Rightarrow B \) is denoted as \( \text{sup}(A \cup B) = \left( \frac{\sum_{t \in D} W_T}{\sum_{t \in D} W_T} \right) \).

Definition 2 Let the minimum weighted support be \( W_{\text{min sup}} \). If the item set \( X \) is a frequent item set, the weighted support is not less than the minimum weighted support, i.e. \( W_{\text{sup}}(X) \geq W_{\text{min sup}} \).

Theorem 1 If item sets \( X \) and \( Y \) are respectively a subset of \( I \), i.e. \( X \subseteq I, Y \subseteq I \) and \( X \subseteq Y \), then \( W_{\text{sup}}(X) \geq W_{\text{sup}}(Y) \).

Proof: Let the transaction data set containing \( X, Y \) be \( T_X, T_Y \), because \( X \subseteq Y \), for \( \forall t \in T_Y \), there must be \( t \in T_X \), so \( \sum_{t \in T_X} W_T \geq \sum_{t \in T_Y} W_T \), \( W_{\text{sup}}(X) \geq W_{\text{sup}}(Y) \) can be obtained.

Theorem 2 (weighted downward closure property) When \( X \subseteq I, Y \subseteq I \) and \( X \subseteq Y \) and \( X \) and \( Y \) have the same prefix, if \( X \) is not frequent, \( Y \) must be infrequent.

Proof: From theorem 1, we know that \( W_{\text{sup}}(X) \geq W_{\text{sup}}(Y) \). If \( X \) is not frequent, we can know \( W_{\text{sup}}(X) < W_{\text{min sup}} \) by definition 2. So \( W_{\text{sup}}(Y) < W_{\text{min sup}} \), i.e. \( Y \) is not frequent.

The weighted model proposed here satisfies the weighted downward closure property, ensuring that the subset of frequent item sets is also frequent, and the parent-set of infrequent item sets is also infrequent, so that frequent item sets can be merged and infrequent item sets can be pruned. Applying the weighted model to the weighted association mining algorithm, we can optimize the algorithm process to mine the weighted frequent item sets.

3. Weighted FP-tree

Taking the transaction database of Table 1 as an example, the frequent pattern mining algorithm based on weighted FP-tree includes the following four steps.

3.1. Scan the database and sort by project support in descending order

Scanning the transaction database of Table 1, we get the project set \( I = \{ A, B, C, D, E, F, G, I, J, L, M, N, O, P \} \). We arrange the items in the project set in descending order of support. When the two projects have the same support, they are arranged in alphabetical order. This arrangement ensures that the sorting results are consistent each time. The ordering of the items and their weights are shown in Table 2.

| Transaction | Transaction set |
|-------------|-----------------|
| T1          | A,B,C,E,F,O     |
| T2          | A,C,G           |
| T3          | E,I             |
| T4          | A,C,D,E,G       |
| T5          | A,C,E,G,L       |
| T6          | E,J             |
| T7          | A,B,C,E,F,P     |
| T8          | A,C,D           |
| T9          | A,C,E,G,M       |
| T10         | A,C,E,G,N       |
Table 2. Project name and weight.

| Project name | Weight |
|--------------|--------|
| A            | 3      |
| C            | 4      |
| E            | 2      |
| G            | 3      |
| B            | 7      |
| D            | 4      |
| F            | 5      |
| I            | 1      |
| J            | 1      |
| L            | 5      |
| M            | 2      |
| N            | 2      |
| O            | 2      |
| P            | 8      |

3.2. Scan the database to normalize the transaction weights

Scanning transaction database calculates the weight of each transaction, i.e., $WT(t) = \sum_{ij} W_{ij} / |t|$, for example, $t_1 = \{A, B, C, E, F, O\}$, $WT(t_1) = (3 + 7 + 4 + 2 + 5 + 2) / 5 = 4.6$. For the same reason, the weight of each transaction is shown in the third column of Table 3. In order to solve the problem that the weight is greater than 1, the transaction weight is normalized, which is also beneficial to the improvement of the algorithm efficiency. The sum of the transaction weights is $W(T) = \sum_{i \in T} WT(t_i)$, and the transaction weights are normalized to calculate the weight ratio of each transaction, for example, $t_1 = \{A, B, C, E, F, O\}$, $WT(t_1) = 4.6$, $W(T) = 31.63333$, $WT(t_1) / W(T) = 0.1454$. Similarly, the normalized transaction weights are shown in the fourth column of Table 3. Also, $WT(t_1) / W(T) + WT(t_2) / W(T) + ... + WT(t_i) / W(T) = 1$.

Table 3. Transaction weight table.

| Transaction | Item set         | Transaction weight | the normalized transaction weight |
|-------------|------------------|--------------------|-----------------------------------|
| T1          | A,B,C,E,F,O      | 4.60000            | 0.1454                            |
| T2          | A,C,G            | 3.33333            | 0.1054                            |
| T3          | E,I              | 1.50000            | 0.0474                            |
| T4          | A,C,D,E,G        | 3.20000            | 0.1012                            |
| T5          | A,C,E,G,L        | 3.40000            | 0.1075                            |
| T6          | E,J              | 1.50000            | 0.0474                            |
| T7          | A,B,C,E,F,P      | 4.83333            | 0.1528                            |
| T8          | A,C,D            | 3.66667            | 0.1159                            |
| T9          | A,C,E,G,M        | 2.80000            | 0.0885                            |
| T10         | A,C,E,G,N        | 2.80000            | 0.0885                            |

The sum of the transaction weights 31.63333 1.0000

3.3. Build the weighted FP-tree

The weighted FP-tree is defined as follows:

(1) Each node of the FP-tree consists of four domains, which are the node name (i.e. name), the sum of the normalized weights of the transaction set of the node (i.e. count), the parent node pointer (i.e. parent), and the child node pointer (i.e. children).
The structure of the item header table is composed of node-name, node-link, and weight. Node-name is the project name of the item header table; node-link points to the first node with the same value as the node-name field; weight records the sum of the weights after the normalization of the transaction set of the project. For example, node-name = 'G', WI ('G') = 0.1054 + 0.1012 + 0.1075 + 0.0885 + 0.0885 = 0.4911.

The following is a weighted FP-tree construction algorithm:

Input: transaction database D, minimum weighted support Wminsup
Output: Weighted FP-tree

1. We need to scan the transaction database, calculate the support count for each item, and build the item header table in descending order of support count.
2. We create a root node (i.e., root) with the initial value set to "Null". For each transaction T in the transaction database, the following operations are performed: deleting items that do not satisfy the minimum weighted support degree, frequent items are sorted in descending order of support degree, and the sorted transaction T is [i, I], where i is the first element of the transaction and I is a collection of remaining elements. Recursively call updataFPtree([i, I], root). The method is performed as follows: First, look up i in the child node of root. If the same node as i is found, the count field of the Node records the weight of the normalized transaction set of the Node; If the same node as i is not found, we create a new node and set its count field value to the normalized weight of the transaction set of Node. The new node is linked to its parent node root, and is linked to the same node through the node chain structure. If I is not empty, recursively call updataFPtree([i, I], root).

Let Wminsup = 0.2, the constructed weighted FP-tree is shown in Figure 1.

### Figure 1. Weighted FP-tree.

#### 3.4. Mining weighted FP-tree

According to theorem 2, the item set with the same prefix has weighted downward closure property, so we start from the second item of the head table to generate the condition pattern base of each item, and construct the weighted condition FP-tree according to the condition pattern base. Table 4 shows the corresponding weighted condition FP-tree and frequent mode.

| Project name | Weighted condition FP-tree | frequent mode |
|--------------|---------------------------|---------------|
| C            | <A:0.9052>                | AC:0.9052     |
| E            | <A:0.5034, C:0.5034>      | AE:0.5034, CE:0.5034 |
| G            | <A:0.4911, C:0.4911, E:0.3929> | AG:0.4911, CG:0.4911, EG:0.3929, ACG:0.4911, AEG:0.3929, ACEG:0.3929 |

---

4
4. Implementation of weighted FP-growth algorithm based on two-dimensional table

In the traditional FP-growth algorithm or the weighted FP-growth algorithm, it is necessary to traverse the conditional pattern base twice. In this paper, the two-dimensional table is used to record the weighted support of each item's two-two combination. Thus it eliminates the first traversal of the algorithm for the conditional pattern basis. The specific algorithm is described as follows:

(1) When traversing the transaction database for the first time, the frequent item set and the transaction data set with the infrequent items removed are obtained. We create an nxn two-dimensional table (i.e. dataframe) by using the transaction data set with the infrequent items removed ( n is the number of frequent items, and the value in the dataframe is initialized to 0), which ensures the consistency of the item header table and the two-dimensional list. This eliminates the need to delete infrequent items in the two-dimensional list, where the dataframe is used to record the weighted support of each frequent item's two-two combination. For example, there is a transaction "A, C, G", the weighted support of (A, C), (A, G), (C, G) in the two-dimensional list is added to the transaction weight normalized by the transaction. Among them, the combinations (A, C) and (C, A) are two different combinations.

(2) Recursively call the weighted condition FP-tree under the condition of \( \alpha \), without obtaining the weighted support by traversing the condition pattern base, which can be obtained from the two-dimensional table. We select the row values and column values associated with \( E \) in the two-dimensional table, and add the corresponding row values and column values to obtain the weighted support of the two-two combination of \( E \) and other frequent items under the condition of \( \alpha \).

(3) Traversing the conditional pattern base, we create a new weighted support two-dimensional table while creating a weighted FP subtree.

The improved algorithm is described using the transaction database of Table 1 mentioned above:

When we traverse the transaction database for the first time, we create a two-dimensional table (i.e. dataframe (nxn)) with an initial value of all 0. In this example, \( n = 7 \), we traverse the transaction data set with infrequent items removed and update the normalized transaction weights for each transaction. The corresponding normalized transaction weights are obtained, and the weighted support degree of each of the frequent items in the transaction is obtained. Then we get the weighted support for the two-two combination of each frequent item in the transaction. For the transaction \( T_1 = \{ A, B, C, E, F, O \} \), all two-two combinations of \( T_1 \) can be obtained: \{A, B\}, \{A, C\}, \{A, E\}, \{A, F\}, \{A, O\}, \{B, C\}, \{B, E\}, \{B, F\}, \{B, O\}, \{C, E\}, \{C, F\}, \{C, O\}, \{E, F\}, \{E, O\}, \{F, O\}. The weighted support for each two-two combination is added to the normalized transaction weight of \( T_1 \). The remaining transaction sets are processed in the same way, and then the normalized transaction weights corresponding to the transaction set in which each combination is located are added as the weighted support of the combination. For example, \{A, B\} in the transaction sets \( T_1 \) and \( T_7 \), the weighted support of \{A, B\} is the sum of the normalized transaction weight of \( T_1 \) and the normalized transaction weight of \( T_7 \). For example, \{A, B\} = 0.1454 + 0.1528 = 0.2982. The resulting two-dimensional table is shown in Table 5.
Next, the establishment of the weighted condition FP subtree is performed to find the weighted support under the condition of $\alpha$.

Since the weighting model used in the algorithm has weighted downward closure property, in order to improve the efficiency of the algorithm, the algorithm in this paper no longer generates the conditional pattern base from the bottom up. Instead, starting from the second item of the item header table, the condition pattern base is generated for each node in the item header table in turn according to the count value of each item, and the corresponding weighted condition FP subtree is constructed by the condition pattern base. Since $C$ is the second item of the item header table, $\alpha$ first grows to $\{C\}$, and the row-column value associated with $C$ is obtained from the two-dimensional table with weighted support: $\{C, A\} = 0.9052 + 0$, $\{C, E\} = 0.6839 + 0$, $\{C, G\} = 0.4911 + 0$, $\{C, B\} = 0.2982 + 0$, $\{C, D\} = 0.2171$, $\{C, F\} = 0.2982 + 0$. Therefore, the weighted support under the condition of $\alpha$ is $L_C = \{A: 0.9052, E: 0.6839, G: 0.4911, B: 0.2982, D: 0.2171, F: 0.2982\}$.

When the growth is $\{E\}$, the row values and column values associated with $E$ are obtained, and the weighted support table under the condition of $\alpha$ is $L_E = \{C: 0.6839, G: 0.3857, B: 0.2982, D: 0.1012, 0.2982\}$.

In the same way, we get a weighted support table for all items to grow to $\alpha$.

5. Algorithm analysis and experimental results

The weighting model used in the algorithm proposed in this paper is more efficient than the traditional weighting model by normalizing the transaction weights. At the same time, the weighted downward closure property of the weighted model saves a part of the mining time, which improves the efficiency of the algorithm. By storing data with a data structure such as FP-tree, there is no need to repeatedly generate candidate sets, which reduces the number of scans of the database and saves I/O overhead. In addition, the process of finding long frequent item sets is transformed into recursively finding some short frequent item sets, and then we connect the suffixes to reduce the search overhead.

The weighted FP-tree frequent item set mining algorithm based on two-dimensional table eliminates the first traversal of the conditional pattern base when generating the weighted conditional FP subtree by generating the two-dimensional table to record the weighted support of the frequent item two-two combination. In terms of time complexity, it is assumed that the database has $m$ items satisfying the minimum weighted support, the number of transactions is $p$, and each transaction contains $n$ items satisfying the minimum weighted support. The experiment ignores the external environmental factors, and the time complexity of the weighted FP-tree algorithm is $mxpxn$. The time complexity of the improved algorithm is $p \times n \times (n - 1) / 2$. It can be seen that when $n > m$, the time complexity of the improved algorithm is smaller, and the shorter the transaction length, the better the performance of the algorithm.

The experimental performance comparison is carried out for the improved algorithm and the weighted FP-tree algorithm (the improved algorithm is recorded as DWFP algorithm and the weighted FP-tree algorithm is recorded as WFP algorithm). The experimental results are shown in Figure 2. The experimental environment: Xiaomi notebook Pro, Windows 10 operating system, memory 8GB, development environment is Pycharm2018. The transaction set is generated 100000 by Python code, and each transaction has a maximum length of 10.
Figure 2. Experimental result.

In the experiment, the minimum weighted support is set to a minimum of 0.1 and the maximum is set to 0.7. As the weighted support increases, the execution time of the two algorithms decreases. Under the same minimum weighted support, the improved DWFP algorithm is faster than the WFP algorithm.

6. Conclusion
In this paper, the weighted FP-tree frequent item set mining algorithm is improved. The two-dimensional list is used to record the weighted support of the frequent items, which eliminates the first traversal of the conditional pattern base when generating the weighted FP subtree and reduces frequent item set mining time. When the weighted support changes, there is no need to scan the database again. We only need to update the two-dimensional table to get a new frequent item set. Compared with the weighted FP-tree algorithm, the performance of the improved algorithm is better.

References
[1] Agrawal, Rakesh, Srikant, et al. (1994) Fast algorithms for mining association rules. In: International Conference on Very Large Data Bases. Santiago, Chile. pp. 487-499.
[2] Han J, Pei J, Yin Y. (2000) Mining frequent patterns without candidate generation. In: ACM SIGMOD International Conference on Management of Data. Dallas, TX, United states. pp. 1-12.
[3] C.H. Cai, W.C. Ada, W.C. Fu, C.H. Cheng, et al. (1998) Mining association rules with weighted items. In: Proceedings of the International Database Engineering and Application Symposium. Cardiff, UK. pp. 68-77.
[4] F. Tao, F. Murtagh, M. Farid. (2003) Weighted Association Rule Mining Using Weighted Support and Significance Framework. In: 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03. Washington, DC, United states. pp. 661-666.
[5] Wen Chen. (2012) Mining Algorithm for Weighted Frequent Pattern Based on Fp Tree. Computer Engineering, 38(6):63-65.
[6] Zhaopeng Pan, Peiyu Liu, Jing Yi. (2018) An Improved FP-Tree Algorithm for Mining Maximal Frequent Patterns. In: 2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA). Changsha, China. pp. 309-312.
[7] T. Saha, et al. (2018) Association Rules to Analyze Hospital Resources with Mortality Rates. In: 2018 5th International Conference on Business and Industrial Research (ICBIR). Bangkok, Thailand. pp. 51-56.
[8] Yan Shi, Yan Fu. (2006) Algorithm for Frequent Pattern Mining Based on FP Reference Tree/List. Computer Science, 33(6):206-209.

[9] K. Sun, F. Bai. (2008) Mining Weighted Association Rules without Preassigned Weights. IEEE Transactions on Knowledge and Data Engineering, 20(4):489-495.

[10] Yan Wang, Haiyan Xue, Lingling Li, et al. (2010) An improved algorithm for mining weighted frequent itemsets. Computer Engineering and Applications, 46(23):135-137+197.

[11] Zhigang Zhao, Jun Wan, Fang Wang. (2014) A Probability-Weighted association rules mining algorithm based on vector. Computer Engineering and Science, 36(2):354-358.

[12] Haitao Hao, Yuanyuan Ma. (2016) Research on e-commerce commodity recommendation system-based on mining algorithm of weighted association rules. Modern Electronic Technology, 39(15):133-136.

[13] Shuai Yue, Shaohong Yin. (2018) Study on frequent patterns mining based on sorted FP-Tree and two-dimensional table. Journal of Harbin University of Commerce(Natural Science Edition), 34(6):692-697.

[14] Hongguang Xiao, Guoqun Deng, Wen Tan, et al. (2018) A Weighted Association Rules Mining Algorithm Based on Matrix Compression. Measurement and Control Technology, 37(3):10-13.