Engineering Applications of Artificial Intelligence 65 (2017) 119–136

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

# Efficient frequent itemsets mining through sampling and information granulation

Zhongjie Zhang [a], Witold Pedrycz [b], Jian Huang [a,*]

[a] *College of Mechatronics Engineering and Automation, National University of Defense Technology, Changsha, 410073, China*
[b] *Department of Electrical and Computer Engineering, University of Alberta, Edmonton, T6R2G7, AB, Canada*

A R T I C L E   I N F O

*Keywords:*
Frequent itemsets mining
Sampling
Information granulation

A B S T R A C T

In this study, we propose an algorithm forming high quality approximate frequent itemsets from those datasets with a large scale of transactions. The results produced by the algorithm with high probability contain all frequent itemsets, no itemset with support much lower than the minimum support is included, and supports obtained by the algorithm are close to the real values. To avoid an over-estimated sample size and a significant computing overhead, the task of reducing data is decomposed into three subproblems, and sampling and information granulation are used to solve them one by one. Firstly, the algorithm obtains rough support of every item by sampling and removes those infrequent items, so the data are simplified. Then, another sample is taken from the simplified data, and is clustered into some information granules. After data reduction, these granules obtained in this way are mined by the improved Apriori. A tight guarantee for the quality of final results is provided. The performance of the approach is quantified through a series of experiments.

## 1. Introduction

The key objective of frequent itemsets (FIs) mining is to form itemsets whose supports are higher than a given minimum support threshold (Agrawal et al., 1993). They are fundamental to association rule (AR) mining and can be applied to a number of areas. Many studies have been devoted to this topic. Agrawal et al. proposed an algorithm called Apriori Agrawal et al. (1994), which mines itemsets with different lengths in different loops and has many improved versions (Agrawal et al., 1994). To avoid repeatedly scanning in Apriori based algorithms, some algorithms change the form of dataset before mining, and there are two main techniques, the tree based method and the bitmap based way. Tree based algorithms, whose representative is FP-growth (Han et al., 2004), transform dataset to a tree structure and scan the tree rather than the original data (Sucahyo and Gopalan, 2004; Deng et al., 2012; Pyun et al., 2014; Alavi and Hashemi, 2015; Deng and Lv, 2015; Deng, 2016; Vo et al., 2016). Bitmap based algorithms, like BitTableFI (Dong and Han, 2007), transform data into a binary matrix or binary numbers, and use logical or matrix operations to replace scanning datasets (Song et al., 2008; Vo et al., 2012; Mohamed et al., 2011). Moreover, some algorithms use heuristic method to directly obtain ARs without mining FIs (Yan et al., 2005, 2009; Sarath and Ravi, 2013).

However, for those tree based and bitmap based algorithms, building and operating the tree or matrix require a lot of memory de-allocation, and for those heuristic based algorithms, evaluating particles or chromosomes needs to scan the whole data. When the data size is huge, the speed of them cannot be ensured.

Therefore, many sampling and information granulation (Bargiela and Pedrycz, 2012; Hu et al., 2015) based algorithms are proposed (Toivonen et al., 1996; Parthasarathy, 2002; Scheffer and Wrobel, 2002; Chen et al., 2002; Zhang et al., 2003; Brönnimann et al., 2003; Li and Gopalan, 2004; Jia and Lu, 2005; Jia and Gao, 2005; Chuang et al., 2005; Hwang and Kim, 2006; Hu and Yu, 2006; Zhao et al., 2006; Chuang et al., 2008; Chakaravarthy et al., 2009; Mahafzah et al., 2009; Pietracaprina et al., 2010; Chandra and Bhaskar, 2011; Chen et al., 2011; Riondato and Upfal, 2014, 2015; Zhang et al., 2015), which reduce data by taking sample of the dataset or compressing the dataset to information granules. The difficulty of them is the tradeoff between runtime and the quality of results. Most methods either call for too much time or give no tight guarantee for the quality of results.

In this study, a new algorithm called SG (Sampling and Granulation) is proposed, which has the good tradeoff between runtime and the quality of final results when dealing with datasets with large scales of transactions. SG not only runs fast but also strictly controls the errors in

* Corresponding author.
*E-mail addresses:* zjiezhang@hotmail.com (Z. Zhang), wpedrycz@ualberta.ca (W. Pedrycz), nudtjHuang@hotmail.com (J. Huang).

http://dx.doi.org/10.1016/j.engappai.2017.07.016
Received 7 August 2016; Received in revised form 18 May 2017; Accepted 17 July 2017
0952-1976/© 2017 Elsevier Ltd. All rights reserved.

a given range with high probability. Being different from other sampling and information granulation based algorithms, the fast speed and high accuracy of SG are caused by the following advantages and innovations:

(a) To avoid the over-estimated sample size, the task of reducing data is split into 3 subproblems, which are reducing the number of items, reducing the scale of transactions and realizing information granulation.
(b) To control the error, the normal approximation to the binomial distribution and a probability inequality called Union bound are joined to study the deviation of supports.
(c) To speed up the following mining, the bitmap technique is also taken to fast obtain the FIs.

The paper is organized as follows. Section 2 introduces some studies about FIs mining through sampling and information granulation, Section 3 covers some definitions and knowledge used in the paper, Section 4 describes the SG in detail, Section 5 shows some results of experiments on SG and gives some discussions, and Section 6 contains conclusions.

## 2. Related studies

Sampling and information granulation based FI mining algorithms can be classified into three categories, algorithm without any guarantee, algorithm with loose guarantee and algorithm with tight guarantee.

Algorithms without any guarantee have high risk to obtain wrong frequent itemsets. Parthasarathy (2002) keeps changing sample size until a criterion is satisfied. Similar works are done by Chen et al. (2002), Brönnimann et al. (2003) and Chuang et al. (2005). Hwang et al. improve Chen's work (Hwang and Kim, 2006) and Hu et al. join the weighted sampling in the progressive sampling (Hu and Yu, 2006). Chuang et al. also design a progressive sampling algorithm based on the probability distribution of the itemsets' supports (Chuang et al., 2008). Chandra et al. also make up with an algorithm to sample dataset based on the supports of items, where Hub-Averaging technique is applied (Chandra and Bhaskar, 2011). Chen et al. take locality sensitive hashing to cluster the initial sample of dataset and remove the outliers of sample (Chen et al., 2011). Zhang et al. propose an algorithm to compress transactions into information granules (Zhang et al., 2015).

Algorithms offering loose guarantee ensure that the deviation of a random support is limited in a given range with high probability, which is not good enough in some datasets with a large number of items. Toivonen builds an algorithm by building candidates and checking the candidates (Toivonen et al., 1996), where sample size is set through Chernoff bound. Zhang et al. (2003) set the sample bound based on central limit theory. A similar study has been reported by Li (Li and Gopalan, 2004). Jia et al. show that even the sample size is low, the accuracy of results can increase by integrating the results based on many individual samples (Jia and Lu, 2005). They also use a progressive sampling method to do this job (Jia and Gao, 2005), which considers the Hoeffding bound to ensure the accuracy of the estimated error of result. Furthermore, Zhao et al. use hybrid bounds to control the sample size (Zhao et al., 2006).

Algorithms with tight guarantee can extract all FIs with high probability, and no itemset with support much lower than the threshold is extracted. However, they either over estimate the sample size or cost too much time and memory. Tobias Scheffer et al. propose an algorithm called GSS (Scheffer and Wrobel, 2002), which mines the approximate $k$ most FIs by progressive sampling. In GSS, itemsets performing poorly are removed one by one as the scale of sample increases. However, the algorithm should store all the possible itemsets firstly. If the scale of items is large, it is unfeasible. Chakaravarthy et al. analysis the smallest size of the sample which can ensure the tight guarantee (Chakaravarthy et al., 2009), which is liner with the longest length of transactions. The algorithm has to scan the whole dataset at least once to get this important parameter. However, when the number of transactions is extremely large, even scanning the whole dataset once is unfeasible. Pietracaprina et al. propose an algorithm to mine the approximate top-$k$ FIs based on Chernoff and Union bound by progressive sampling (Pietracaprina et al., 2010). However, the maximum length of FIs should be fixed. Matteo Riondato et al. set the smallest sample size to ensure the tight guarantee of the results, which is based on VC-dimension (Riondato and Upfal, 2014), but it has to scan the dataset at least once, which is unfeasible when the size of dataset is extremely large. Matteo Riondato et al. improved their algorithm by progressive sampling, and the stop condition is based on Rademacher Averages (Riondato and Upfal, 2015).

## 3. Preliminaries

### 3.1. The $(\varepsilon, \delta)$-approximate FI

Given a dataset $D$ with some transactions $\{t_1 t_2 \dots t_m\}$, $I = \{a_1, a_2 \dots a_{|I|}\}$ is the universe set of items appearing in $D$. For $\forall t_i \in D$, $t_i \subseteq I$. An itemset, denoted by $x$, is a subset of $I$. The support of $x$, denoted by $f_D(x)$, is the percent of the transactions which contain $x$ in $D$. Given a minimum support $\theta$, the set of FIs of $D$ under $\theta$ is $FI_D = \{x | f_D(x) \geq \theta\}$, and $FI_S$ is the set of FIs mined from a sample $S$.

**Definition 1.** Given $S$, a sample of a dataset $D$, if $FI_S$ satisfies

(a) for every $x \in FI_S$, $|f_S(x) - f_D(x)| \leq \varepsilon/2$,
(b) for every $f_D(x) \geq \theta$, $x \in FI_S$,
(c) and for every $f_D(x) < \theta - \varepsilon$, $x \notin FI_S$,

$FI_S$ is the set of $\varepsilon$-approximate FIs of $D$ under $\theta$. If $FI_S$ has at least $1-\delta$ probability to be the set of $\varepsilon$-approximate FIs, $FI_S$ is the set of $(\varepsilon, \delta)$-approximate FIs, where $(\varepsilon, \delta)$ is set by the user depending on the demand of accuracy (Riondato and Upfal, 2015). The aim of our algorithm is to produce $(\varepsilon, \delta)$-approximate FIs in an efficient manner.

### 3.2. The union bound

Given two events $e_A$ and $e_B$, if they satisfy $\Pr(e_A) \geq 1 - p_A$ and $\Pr(e_B) \geq 1 - p_B$ at the same time, because $\Pr\left(e_A \cup e_B\right) = \Pr\left(e_A\right) + \Pr\left(e_B\right) - \Pr\left(e_A e_B\right)$ and $\Pr\left(e_A e_B\right) = \Pr\left(e_A\right) + \Pr\left(e_B\right) - \Pr\left(e_A \cup e_B\right)$, $\Pr\left(e_A e_B\right) \geq 1 - p_A - p_B$. This probability inequality is called Union bound. Moreover, if $\Pr(e_{A1}) \geq 1 - p_{A1}$, $\Pr(e_{A2}) \geq 1 - p_{A2}$ ... and $\Pr(e_{An}) \geq 1 - p_{An}$, we can get that

$$\Pr\left(e_{A1} e_{A2} \dots e_{An}\right) \geq 1 - \sum_{i=1}^{n} p_{Ai}. \tag{1}$$

### 3.3. The agglomerative hierarchical clustering

Agglomerative hierarchical clustering is a bottom-up clustering algorithm, where each element starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. The algorithm runs through two kinds of distances, the distance between two elements and the distance between two clusters. When clustering, the algorithm keeps merging two clusters with the shortest distance until the stopping condition has been satisfied (Akbari et al., 2015).

## 4. The sampling and granulation in SG

### 4.1. The outline of SG

Fig. 1 visualizes the essence of SG. The algorithm reduces the original dataset step by step. Firstly, items with low probability to be FIs are removed, which simplifies the problem. Then, SG takes another sample $S_2$, whose size is set fast based on the remained items and their supports in $S_1$. Furthermore, the algorithm granulates $S_2$, where those same and similar transactions are compressed in a granule, so the scale of data can be further reduced. SG ensures the final results be the $(\varepsilon, \delta)$-approximate FIs, and the detailed steps are shown as follows.