Data & Knowledge Engineering 89 (2014) 21–37

Contents lists available at ScienceDirect

Data & Knowledge Engineering

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

Editorial

Mining frequent itemsets in data streams within a time horizon

Luigi Troiano, Giacomo Scibelli

University of Sannio, Dept. of Engineering, Viale Traiano, 82100 Benevento, Italy

ARTICLE INFO

Article history:
Received 18 August 2011
Received in revised form 22 October 2013
Accepted 22 October 2013
Available online 27 December 2013

Keywords:
Data mining
Mining methods and algorithms
Frequent itemsets

ABSTRACT

In this paper, we present an algorithm for mining frequent itemsets in a stream of transactions within a limited time horizon. In contrast to other approaches that are presented in the literature, the proposed algorithm makes use of a test window that can discard non-frequent itemsets from a set of candidates. The efficiency of this approach relies on the property that the higher the support threshold is, the smaller the test window is. In addition to considering a sharp horizon, we consider a smooth window. Indeed, in many applications that are of practical interest, not all of the time slots have the same relevance, e.g., more recent slots can be more interesting than older slots. Smoothness can be determined in both qualitative and quantitative terms. A comparison to other algorithms is conducted. The experimental results prove that the proposed solution is faster than other approaches but has a slightly higher cost in terms of memory.

1. Introduction

When we search for interesting patterns in sets, sequences and relations of data, the identification of target itemsets, i.e., subsets of occurring items, represents the starting point of most of the analyses. As mined itemsets can reveal interesting and previously unknown facts [1–8], the learning of recurrent patterns in data streams is challenging because of the dynamic nature of the problem and the additional real-time constraints.

Data streams occur in many examples of practical interest, such as in sensor data generated by sensor networks, online transactions recorded by the retail chains, grocery sales data, visited pages and click-streams in web applications, calls and instant messages that are placed in telecommunication networks, daily weather/traffic records, stock market data, or performance measurements in network traffic management and monitoring. The mining of flowing data offers new challenges, because data streams are continuous and unbound, they require to be processed considering real-time constraints and data distributions which change over time. In the data streams, some of the patterns become more frequent, while other patterns tend to disappear. As motivating examples we outline the following two cases: the visited web pages and the rainfall data. In the first case, we are interested to analyze the web traffic generated by single visits in order to identify which pages are mostly visited over time. In this case, items are web pages or site sections, and transactions collect which pages/sections are visited by a unique visitor. Records have different sizes. Frequent itemsets represent the groups of pages/sections that are mostly visited within a limited time horizon. In the second case, we aim to analyze how rainfalls are changing by time over an area of interest. Here, items are the rainfall levels recorded over the time at different locations within a geographic area. Records have the same size. Frequent itemsets relate to groups of rainfall levels that are mostly occurring within a time period.

Both cases will be used as domains of application as part of our experimentation.

The time horizon provides a frame of interest in which to search for the frequent itemsets. The contribution of this paper is in presenting the Window Itemset Shift (WIS) algorithm, which provides an efficient solution to the search for frequent itemsets in a

E-mail addresses: troiano@unisannio.it (L. Troiano), scibelli@unisannio.it (G. Scibelli).

0169-023X/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.datak.2013.10.002

stream of transactions when a time horizon is given, specifically, when this horizon is smoothly defined by fuzzy sets or determined according to stream characteristics.

The simple iterated structure of a standard algorithm is inefficient, because the search begins from scratch at each iteration and does not account for any memory of the transaction stream. This case is the Apriori [1] algorithm, when it is re-iterated as new transactions are entering the frame of interest. Other approaches have been proposed in the literature. Although the literature contains some optimizations to Apriori, their logic is based on re-iterating the algorithm at each step.

However, frequent itemsets must occur in a restricted portion of time. If not, the support does not reach the threshold. With respect to this property, an initial idea is to consider the minimal window that makes it possible to discard those itemsets that are certainly not frequent. In addition, because most of the itemsets are held within the time frame of interest, a further enhancement is to maintain a memory of itemset candidates that might become frequent and to update this list as new transactions are entering and old transactions are exiting the time frame. As a consequence, the number of candidates to consider at each step is smaller than those processed by other approaches. Finally, the support counting is limited to the test window and it does not require a pass through the dataset. This approach, as shown by experimentation on datasets with different characteristics, allows the algorithm to perform better than the other algorithms proposed in the literature.

Because the time slots might have a different relevance within the time horizon considered, we can assign a degree of interest to each of them. This approximation leads to "fuzzify" the algorithm in such a way that the support count is affected by the different relevances assumed by the transactions as they move along the time frame. Although this feature does not provide any further enhancement to the performances, it expands the meaningfulness of the algorithm, enabling a higher expressivity. In addition, as explained in Section 4, this feature allows us to address with the unbounded windows with a limited capacity and improves the algorithm's scalability.

This paper is organized as follows: Section 2 introduces the definitions and the properties of frequent itemsets for the non-familiar reader, Section 3 provides a brief overview of the related research, Section 4 introduces the concepts that underlie the WIS algorithm, Section 5 describes the algorithm's logic and architecture, Section 6 compares WIS to other algorithms by experimentation on datasets that have different characteristics, and Section 7 outlines the conclusions.

2. Preliminaries

A group or set of items entailed by database records, e.g., the set of items that a customer collects in a market basket, is referred to as an *itemset*. More formally, let $X = \{i_1,i_2,\ldots,i_m\}$ be a set of m distinct literals called *items*. Let the dataset $\mathcal{D}$ be a collection of transactions over X. Each transaction (or record) $r \in \mathcal{D}$ contains a subset of items, i.e., $r \equiv \{i_i,i_j,\ldots,i_k\} \subseteq X$. The number of items in r provides its *length*. The transactions in $\mathcal{D}$ can entail different lengths. A subset of items $I \subseteq X$ is called an *itemset*.

The number of times that an itemset I occurs in the transactions is the *support count* of I, which is denoted as $supp(I)$. Frequent itemsets are defined with respect to the support threshold S; as a result an itemset I is frequent if $supp(I) \geq S$.

The itemset support count is related to the support count of its subsets and supersets. In fact, given two itemsets I and J such that $I \subset J$, the number of times that I occurs is at least the number of occurrences of J because the former is part of the latter. Therefore,

$$supp(I) \geq supp(J), \quad \forall I \subset J \subseteq X \tag{1}$$

In addition, it is useful to further classify an itemset X as being *maximal frequent* if X is frequent, but any $Y \supset X$ is not [9].

The problem of counting the number of distinct itemsets in a dataset, given an arbitrary support threshold, is NP-complete, and the problem of mining itemsets is NP-hard [10]. If $|X|$ is the cardinality of X, then the number of possible distinct itemsets is $2^{|X|}$. To reduce the combinatorial search space, most of the algorithms exploit the following two properties:

- *Downward closure*: all of the subsets of a frequent itemset are frequent[1]
- *Anti-monotonocity*: supersets of an infrequent itemset must be infrequent, too.

3. Related research

3.1. Mining of frequent itemsets

Traditionally, itemset mining is associated with the discovery of association rules [2,11] because they provide a subset of patterns within which to search. The problem of mining association rules in large databases can be divided into two subproblems: (i) seeking frequent itemsets; and (ii) discovering association rules among the found itemsets.

The first and most noticeable algorithm for mining frequent itemsets is known as *Apriori*, which was proposed independently by Agrawal and Srikant [3] and Mannila, Toivonen and Verkamo [12] and was later joined in [4]. Apriori is a levelwise, breadth-first, bottom-up algorithm, as outlined by Algorithm 1.

[1] The name of the property comes from the fact that the set of frequent itemsets is closed with respect to set inclusion.