Chapter 4

FREQUENT PATTERN MINING IN DATA STREAMS

Ruoming Jin

Computer Science Department
Kent State University
jin@cs.kent.edu

Gagan Agrawal

Department of Computer Science and Engineering
The Ohio State University
agrawal@cse.ohio-state.edu

Abstract

Frequent pattern mining is a core data mining operation and has been extensively studied over the last decade. Recently, mining frequent patterns over data streams have attracted a lot of research interests. Compared with other streaming queries, frequent pattern mining poses great challenges due to high memory and computational costs, and accuracy requirement of the mining results.

In this chapter, we overview the state-of-art techniques to mine frequent patterns over data streams. We also introduce a new approach for this problem, which makes two major contributions. First, this one pass algorithm for frequent itemset mining has deterministic bounds on the accuracy, and does not require any out-of-core summary structure. Second, because the one pass algorithm does not produce any false negatives, it can be easily extended to a two pass accurate algorithm. The two pass algorithm is very memory efficient.

1. Introduction

Frequent pattern mining focuses on discovering frequently occurring patterns from different types of datasets, including unstructured ones, such as transaction and text datasets, semi-structured ones, such as XML datasets, and structured ones, such as graph datasets. The patterns can be itemsets, sequences, subtrees, or subgraphs, etc., depending on the mining tasks and targeting datasets. Frequent patterns can not only effectively summarize the underlying datasets,
providing key sights into the data, but also serve as the basic tool for many other data mining tasks, including association rule mining, classification, clustering, and change detection among others [21, 37, 20, 24].

Many efficient frequent pattern algorithms have been developed in the last decade [1, 17, 18, 35, 26, 33, 36]. These algorithms typically require datasets to be stored in persistent storage and involve two or more passes over the dataset. Recently, there has been much interest in data arriving in the form of continuous and infinite data streams. In a streaming environment, a mining algorithm must take only a single pass over the data [4]. Such algorithms can only guarantee an approximate result.

Compared with other stream processing tasks, the unique challenges in discovering frequent patterns are in three-fold. First, frequent pattern mining needs to search a space with an exponential number of patterns. The cardinality of the answering set itself which contains all frequent patterns can be very large too. In particular, it can cost much more space to generate an approximate answering set for frequent patterns in a streaming environment. Therefore, the mining algorithm needs to be very memory-efficient. Second, frequent pattern mining relies on the down-closure property to prune infrequent patterns and generate the frequent ones. This process (even without the streaming constraint) is very compute-intensive. Consequently, keeping up the pace with high-speed data streams can be very hard for a frequent pattern-mining task. Given these challenges, a more important issue is the quality of the approximate mining results. The more accurate results usually require more memory and computations. What should be the acceptable mining results to a data miner? To deal with this problem, a mining algorithm needs to provide users the flexibility to control the accuracy of the final mining results.

In the last several years, several new mining algorithms have been proposed to find frequent patterns over data streams. In the next chapter, we will overview these new algorithms.

2. **Overview**

2.1 **Frequent Pattern Mining: Problem Definition**

Let the dataset \(D\) be a collection of objects, i.e. \(D = \{o_1, o_2, \cdots, o_{|D|}\}\). Let \(P\) be the set of all possible (interesting) patterns occurring in \(D\), \(g\) be the counting function \(g : P \times O \rightarrow N\), where \(O\) is the set of objects, and \(N\) is the set of nonnegative integers. Given parameters \(p \in P\), and \(o \in O\), \(g(p, o)\) returns the number of times \(p\) occurs in \(o\). The support of a pattern \(p \in P\) in the dataset \(D\) is defined as

\[
\text{supp}(p) = \sum_{j=0}^{j=|D|} I(g(p, o_j))
\]

where \(I\) is an indicator function.