Frequent Sets Mining in Data Stream Environments

Xuan Hong Dang
Nanyang Technological University, Singapore

Wee-Keong Ng
Nanyang Technological University, Singapore

Kok-Leong Ong
Deakin University, Australia

Vincent Lee
Monash University, Australia

INTRODUCTION

In recent years, data streams have emerged as a new data type that has attracted much attention from the data mining community. They arise naturally in a number of applications (Brian et al., 2002), including financial service (stock ticker, financial monitoring), sensor networks (earth sensing satellites, astronomic observations), web tracking and personalization (web-click streams). These stream applications share three distinguishing characteristics that limit the applicability of most traditional mining algorithms (Minos et al., 2002; Pedro and Geoff, 2001): (1) the continuous arrival rate of the stream is high and unpredictable; (2) the volume of data is unbounded, making it impractical to store the entire content of the stream; (3) in terms of practical applicability, stream mining results are often expected to be closely approximated the exact results as well as to be available at any time. Consequently, the main challenge in mining data streams is to develop effective algorithms that support the processing of stream data in one-pass manner (preferably on-line) whilst operating under system resources limitations (e.g., memory space, CPU cycles or bandwidth).

This chapter discusses the above challenge in the context of finding frequent sets from transactional data streams. The problems will be presented and some effective methods, both from deterministic and probabilistic approaches, are reviewed in details. The trade-offs between memory space and accuracy of mining results are also discussed. Furthermore, the problems will be considered in three fundamental mining models for stream environments: landmark window, forgetful window and sliding window models.

BACKGROUND

Generally, the problem of finding frequent sets from a data stream can be stated as follows. Let $I = \{a_1, a_2,... a_m\}$ be a set of items (or objects) and let X be an itemset such as $X \subseteq I$. Given a transactional data stream, DS, which is a sequence of incoming transactions, $(t_1, t_2,...,t_n)$, where $t_i \subseteq I$, the frequency of X is defined as the number of transactions in DS that contain X and its support is defined as the ratio between its frequency and the number of transactions in the stream. Then, the problem of mining frequent sets from data stream DS is to find all itemsets whose support is greater than a given threshold $s \in (0,1)$ called minimum support.

However, due to the unlimited size of streaming data, all transactions appearing in the stream DS are not always of equal importance. Rather, their usefulness is dependent upon applications. For example, some applications focus only on a fixed portion (most recently arriving) of the stream, while other applications consider all transactions seen so far; yet the transaction weight (or importance) is reduced gradually with time. This gives rise to different mining models: Landmark window model, Forgetful window model and Sliding window model. In the following section, most typical algorithms classified in these models will be discussed in details.

MAIN FOCUS

This section presents and discusses some typical algorithms addressing the problem of finding frequent sets from data streams. The mining model is first described, following it are the algorithm analysis and discussion.

Landmark Window Model

In this model, frequent sets are computed from a set of contiguous transactions in the stream identified between a specific transaction in the past, called landmark, and the current transaction. Accordingly, one endpoint (landmark) is fixed, while the other gradually increases with the arrival of new transactions. This model is often suitable for off-line data streams where transactions usually arrive in batches (Gurmeet and Rajeev, 2002), for example, with warehouse applications where new batches of records are updated at regular time intervals. However, there are also many other online streaming applications employing this model (Johannes et al., 2001). For example, in a telecom transaction stream, the scope of the stream is captured on a daily basic. Accordingly, a set of landmarks is given to the mining system and the results are computed from the current point back to the immediately-preceding landmark. We discuss some typical algorithms in this mining model.

Lossy Counting (Gurmeet and Rajeev, 2002) is one of the first algorithms to find all frequent sets from entire transactional streams. It is a deterministic approach since the mining results are guaranteed within some error. Given an error threshold ε and a minimum support threshold s, Lossy Counting guarantees that all itemsets whose true frequency exceeds sN are included in the mining results (where N is the number of transactions seen so far in the stream). Furthermore, the estimated frequency for each output itemset is guaranteed to be no more than its true frequency by an amount of εN. In Lossy Counting, the stream is divided into buckets, each has the same size of $w = 1/\varepsilon$ transactions and labeled with bucket ids, starting from 1. At transaction N, the bucket id is $b = \lceil N/w \rceil$. During the mining process, Lossy Counting maintains a synopsis S which is a set of entries (X, f, Δ), where X is an itemset, f is its estimated frequency, and Δ is the maximal possible error in f. For each X found in the stream, its frequency will be incremented by 1 if it is found in S; otherwise a new entry is created with the value $(X, f, b - 1)$, where $b = \lceil N/w \rceil$. At each bucket boundary (i.e., $N \equiv 0 \bmod w$), S is pruned by deleting all itemsets that have $f + \Delta \leq b$. The frequent itemsets are those having $f \geq (s - \varepsilon)N$.

To see how Lossy Counting approximates frequency errors, it is important to note that the stream is divided into equally sized buckets, and each itemset X will have its frequency tracked only if it appears frequently enough; i.e., on average, at least once on every bucket sized $1/\varepsilon$. Accordingly, the deletion rule ensures that whenever $f \leq b - \Delta$, X will be deleted. Note that, $(b - \Delta)$ is the number of buckets since the last time X occurs frequently enough to be counted. This value is always no more than N/w which is the number of buckets so far in the stream. Consequently, the pruning condition always makes sure that $f \leq b - \Delta \leq N/w$ or $f \leq \varepsilon N$. This is also the maximal lost on counting frequency of every itemset.

It is important to note that when Lossy Counting is applied to mining frequent itemsets, it needs to mine the stream in batch mode in order to avoid explicitly enumerating all subsets of every transaction. Consequently, this approach may limit its applicability in some online streaming environments (Johannes et al., 2001). Recently, this drawback has been addressed by EStream algorithm (Xuan Hong et al., 2006). In this algorithm, the error on the mining results is strictly guaranteed within a given threshold whilst the data stream is processed online. The basic idea of EStream is still based on the bucketing technique. However, it has been observed that when the transaction flow is processed online and the downward closure property is employed to find frequent sets, a longer itemset is usually being delayed counting until all its subsets are found potentially frequent. Accordingly, there will be a bigger error margin on frequent sets of larger sizes. Therefore, in EStream, the error (also frequency) of each itemset is identified precisely based on their length. This algorithm has been theoretically proven and guarantees that: (i) there is no error on frequency counting of 1-itemsets; (ii) for 2-itemsets, the maximal error on frequency counting is no more than εN; (iii) and for itemsets of length $k > 2$, the error is no more than $2\varepsilon N$ (where k is the maximal length of frequent itemsets).

The advantage of a deterministic approach is that it is able to produce all truly frequent itemsets and limit the maximal error on frequency counting. However, in order to provide this guarantee, a large amount of