Relevance of Counting in Data Mining Tasks

Osmar R. Zaïane

Department of Computing Science,
University of Alberta,
Edmonton AB, Canada
zaiane@cs.ualberta.ca

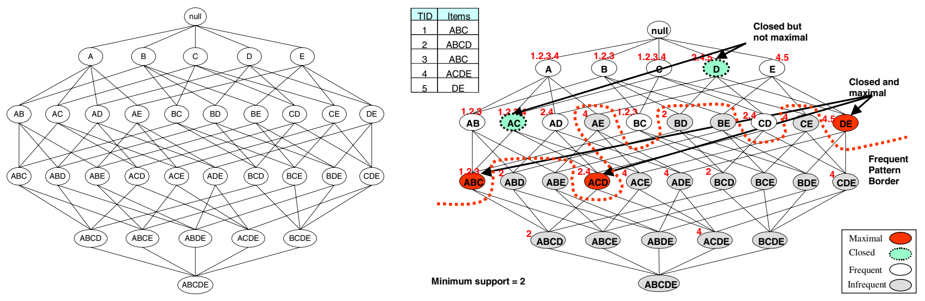
Abstract. In many languages, the English word "computer" is often literally translated to "the counting machine." Counting is apparently the most elementary operation that a computer can do, and thus it should be trivial to a computer to count. This, however, is a misconception. The apparently simple operation of enumeration and counting is actually computationally hard. It is also one of the most important elementary operation for many data mining tasks. We show how capital counting is for a variety of data mining applications and how this complex task can be achieved with acceptable efficiency.

1 Introduction

Counting is an elementary computer operation. Of course for humans counting has its limitations, but for computers one might think it is a trivial task. All computer programs entail counting in one form or another. From business management programs to programs for scientific research, counting is omnipresent in the implementations of these programs. However, counting can be very complex. In particular, the scalability issue with counting can be of a major concern.

For example, consider an alphabet of 5 letters {a, b, c, d, e}. The number of all possible subsets is 32 (i.e $2^5 = 32$). All these combinations are shown in Figure 1. These 5 letters could be the 5 unique products that a specialized store sells. The combinations illustrated in Figure 1 are the possible transactions that potential customers have when visiting this store. To study the relationships between products bought together, one might take existing real transactions and for each transaction, check the combinations, and for each of them traverse the graph in Figure 1 to increment the respective counters. Let us take the example now of a more realistic department store with 10,000 different products. The graph in Figure 1 would explode to $2^{10,000}$ nodes. Traversing the graph efficiently to find and increment the exact counter is complex, but even keeping the complete graph resident in main memory is phenomenal.

To make things more complicated, let us reduce the alphabet to only 4 {A, C, G, T} and now allow the items to repeat in a combination. In a very long sequence of these 4 items, counting the existing different combinations (or subsequences) of different lengths in this sequence is almost unthinkable. This is



TID	Items
1	ABC
2	ABCD
3	ABC
4	ACDE
5	DE

Fig. 1. Left:Search Space with an alphabet of 5 items. Right: Example database and the frequent pattern border at minimum support of 2

a common problem in genomics where the 4 letter alphabet is Adenine, Cytosine, Guanine, Thymine and the sequences of DNA are hundreds of thousands of elements long. In proteomics, studying the proteins, the alphabet is of 20 amino-acids making counting a more daunting task.

2 Enumeration and Counting in Data Mining

Here are some typical examples of data mining applications where counting is important and at the same time can be overwhelming.

2.1 Frequent Itemset Mining

Enumerating and counting frequent itemsets is the first and most important phase in the process of mining for association rules. The illustrative example given above is an example of market basket analysis, the typical application for association rules [1], However, the counting of itemsets is also an intricate part of many other data mining tasks and applications such as classification [8, 13] and clustering [4].

2.2 Event Sequence Analysis and Sequential Patterns

When items, events or measurements are chronologically ordered, the time element becomes relevant and should be taken into account. There are many variations of these sequences depending upon the nature of the elements in the sequence. If they are nominal symbols from a given alphabet, the sequence is known as a temporal event sequence [12] such as the events on a power or telecommunication grid or simple click-streams on a web site; if they are continuous valued elements, the sequence is known as time series [10] such as stock market feeds or meteorological data. Detecting important subsequences or building predictive models from these data require sophisticated counting.

2.3 Frequent Subsequence Analysis

In bioinformatics, identifying and counting significant sub-sequences in a set of very long sequences is important for the understanding of protein functions, the identification of transcriptor factors and even the reconstruction of genome phylogeny [11]. Given the particularly large sequences and close to infinite search space, erudite methods for counting sub-sequences were devised [7].

2.4 Contrast Sets

Contrast sets [2] are used to describe the fundamental differences between groups. Simply put, they are conjunctions of items that differ meaningfully in their distributions across groups. This again entails counting. Another variation of these are emerging patterns [5].

3 Top-Down Versus Bottom-Up Enumeration

Looking at Figure 1, it is obvious that when the alphabet is large, enumerating all nodes is onerous if not infeasible. The clever idea used in [1] and later in many other publications is based on the *apriori* property or observation. There is no need to visit a node and all its descendents if one if its ancestors is not frequent. The goal is to find the frequent pattern border (Figure 1, Right) above which itemsets are frequent and below which itemsets are irrelevant. Nevertheless, this approach may yield too many useless enumerations for nodes that are doomed to be irrelevant. This is particularly the case for datasets with long patterns (i.e. the frequent pattern border is deep in the graph). The bottom-up approach, starting from the long patterns and going toward the empty set searching for the frequent pattern border is also interesting with very clever heuristics for pruning. It is however burdensome if this border is high in the graph (i.e. the relevant patterns are short). Many attempts at reducing the enumeration of frequent patterns were done by concentrating on non redundant itemsets such as closed [9] and maximal patterns [3], but the main strategies remain the same: either top-down or bottom-up.

4 Leap Approach

There are two issues in frequent itemset mining: relevant itemset enumeration and counting the exact frequencies. Discovering all frequent patterns from the non-redundant sets such as the maximal patterns, does not necessarily mean we get de facto their exact counts. Moreover, every superfluous enumeration and counting of an itemset doomed infrequent is definitely time-consuming and useless in the final result. The idea of a leap traversal of the search space is, rather than systematically traverse the graph top-down or bottom-up, to cunningly jump from one node to the other avoiding as much as possible the superfluous nodes doomed irrelevant. The leap traversal searches for the frequent pattern border by identifying maximal patterns and collecting enough information in the process to generate exact counts for all frequent patterns at the end. The idea is that maximal patterns, which subsume all frequent patterns, are frequent sub-transactions. The process starts by marking some interesting nodes that appear as complete sub-transactions of frequent unique items. The leap is made from those marked nodes identified as non maximals and the jump goes to the node resulting from intersecting the marked nodes that are not maximals [6]. The jumps are often many levels ahead, avoiding many irrelevant nodes. On average, the leap traversal generates and tests more than one order of magnitude less candidates than other traversal strategies and produces the exact same results at the end of the process. This approach can mine millions of transactions with hundreds of thousand items on a small desktop in a reasonable time (i.e. few seconds). However, while linear scalability is achieved, with larger and larger real application datasets, physical limits are quickly reached. Current hardware and state-of-the-art algorithms can not cope realistically. More clever ideas are needed for real world enumeration and counting problems.

5 Conclusion

Mining for frequent itemsets is a canonical task, fundamental for many data mining applications. It is used to generate association rules, produce contrast sets, count frequent subsequences in event sequences and time series, estimate probabilities in a belief network, and even create a rule-based classification model or cluster data. It is the primary operation for data analysis. Yet, it is still an open problem how to achieve this counting efficiently. Many algorithms have been reported in the literature, some original and others extensions of existing techniques. While we showed some effective approaches for this task, the problem of how to improve on the existing methods remains a challenging puzzle for the future.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases*, pages 487–499, Santiago, Chile, September 1994.
2. S. Bay and M. Pazzani. Detecting changes in categorical data: Mining contrast sets. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 302–306, San Diego, USA, 1999.
3. R. J. Bayardo. Efficiently mining long patterns from databases. In *ACM SIGMOD*, 1998.
4. F. Beil, M. Ester, and X. Xu. Frequent term-based text clustering. In *ACM Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD'02)*, Edmonton, Canada, 2002.
5. G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *ACM SIGKDD International Conference on Knowledge Discovery and Data MIning*, pages 43–52, San Diego, USA, 1999.

6. M. El-Hajj and O. R. Zaïane. Cofi approach for mining frequent itemsets revisited. In *9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD-04)*, pages 70–75, Paris, France, June 2004.
7. D. Gusfield. *Algorithms on Strings, Trees, and sequences: Computer Science and Coputational Biology*. Cambridge University Press, 1997.
8. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *4th Intl. Conf. on Knowledge Discovery and Data Mining (KDD'98)*, pages 80–86, New York City, NY, August 1998.
9. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *International Conference on Database Theory (ICDT)*, pages pp 398–416, January 1999.
10. A. Weigend and N. Gershenfeld. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, 1993.
11. X. Wu, X. Wan, G. Wu, D. Xu, and G.-H. Lin. Whole genome phylogeny construction via complete composition vectors. Technical Report TR05-06, Department of Computing Science, University of Alberta, January 2005.
12. Q. Yang, H. Wang, and W. Zhang. Web-log mining for quantitative temporal-event prediction. *IEEE Computational Intelligence Bulletin*, 1(1):10–18, December 2002.
13. O. R. Zaïane and M.-L. Antonie. Classifying text documents by associating terms with text categ ories. In *Proc. of the Thirteenth Australasian Database Conference (A DC'02)*, pages 215–222, Melbourne, Australia, January 2002.