Difference between binary and itemset representations in frequent itemset mining algorithms

---

Approach of investigation

Firstly, we tried searching for papers that compared between the two dataset representations but could not find a paper that directly addressed the difference between the two representations. Then, we started searching for implementations of frequent itemset mining algorithms of both the binary and itemset representations, but did not find many algorithms that were implemented using both the representations. Finally, we analysed the two implementations using the two representations of the Apriori algorithm and came up with the advantages and disadvantages of the two representations, mainly in terms of run-time and space considerations.

Binary representation

The dataset's rows correspond to transactions and columns correspond to items. The transactions are represented as binary vectors where a 0 refers to an absence and a 1 refers to a presence of an item in the transaction.

Advantages of binary representation

1. **Efficient array operations:** The representation allows the use of efficient array operations such as summing for support calculation, intersection between sets for candidate generation, and so on.

2. **Efficiency in dense datasets:** Run-time performance will improve compared to an itemset representation as the dataset becomes more dense because the use of vectorized computations will be more effective.

3. **Reduced traversal of items:** Due to the use of array operations like slicing and indexing, there is reduced necessity of traversal of items.

Disadvantages of binary representation

1. **Space inefficiency:** If the dataset is sparse, i.e. not a lot of items are present in most transactions, but there are a lot of items, then there will be a lot of 0s in the dataset. And, it will be costly to store large and sparse datasets. Moreover, this is a problem because the gains from vectorized computations may not be enough to outweigh the disadvantages of space inefficiency if compared to the itemset representation.

2. **Extra large representation:** Since the representation has one column for each item, if there are a large number of items, the program will have to process all the information of all the columns in each of its operations. This may lead to stack overflows for some operations.

3. **Added complexity:** Preprocessing is required to represent the transactions as a vector of 0s and 1s and this might introduce errors or inefficiencies.

Itemset representation

The dataset can be represented as a list of sets where each set corresponds to a transaction containing the names of the items involved in the transaction.

Advantages of itemset representation

1. **Space efficiency for sparse datasets:** We only need to store information for the exact number of items present in each transaction. This leads to a great advantage over the binary representation especially when the dataset is sparse.

2. **Candidate generation:** Efficient libraries like Python's itertools can be used to easily generate candidates.

3. **Natural Representation:** Less preprocessing required compared to binary representation as the itemset representation is similar to real-world transaction data.

Disadvantages of itemset representation

1. **Several traversals over items:** Traversals over items to calculate support and generate candidates increase the run-time of implementations using this representation. For dense datasets, i.e. each transaction having many items, the problem is exacerbated.

Conclusion

From our high-level analysis, we can get a sense that itemset representation may be better suited for sparse and/or large datasets because of the space efficiency, and binary representation may be better for dense and small datasets because of the efficient vectorized operations and reduced traversal of items.

Reference

1. https://github.com/ymoch/apyori/tree/master
2. https://github.com/rasbt/mlxtend/tree/master
3. https://borgelt.net/