DATA MINING

**Frequent Item Dataset**

**Market-Basket Data** – A general many-to-many mapping between two kinds of things, where the one (the baskets) is a set of the other (the items). The technology focuses on common events, not rare events.

Example:

|  |  |
| --- | --- |
| TID | Items |
| 1 | Bread, Milk |
| 2 | Bread, Diaper, Beer Eggs |
| 3 | Milk, Diaper, Beer, Coke |
| 4 | Bread, Milk, Diaper, Beer |
| 5 | Bread, Milk, Diaper, Coke |

Examples of frequent itemsets s(1) >= 3

{Bread} = 4

{Milk} = 4

{Diaper} = 4

{Beer} = 3

{Diaper, Beer} = 3

{Milk, Bread} = 3

Items = products; basket = sets of products

**Definition**:

**Itemset**: a collection of one or more items

**Support**: count (frequency of occurrence of an itemset), fraction (fraction of transaction that contain an itemset)

**Frequent itemset**: an itemset whose support is greater than or equal to a minsup threshold

**Mining frequent itemsets task**

Input: a set of transactions T, over a set of items I

Output: all itemsets with items in I having support >= minsup threshold

Problem parameters:

N = |T|: number of transactions

d = |I|: number of distinct items

w: max width of transactions

number of possible itemsets? M= 2d

**A Naïve algorithm**

**Brute-force approach**, each itemset is a candidate:

Consider each itemset is a lattice, and count the support of each candidate by scanning he data

Time complexity – O(NMw), Space complexity – O(M)

**Or**:

Scan the data, and for each transaction generate all possible itemsets. Keep a count for each itemset in the data

Time complexity – O(N2w), Space complexity – O(M)

**Computation Model**

Typically, data is kept in flat files rather than in database system.

* Stored on disk
* Stored basket-by-basket
* Expand baskets into pairs

**Computational Model 2**

The true cost of mining disk-resident data is usually the number of disk I/O’s

In practice, association-rule algorithms read the data in passes – all baskets read in turn

Thus, we measure the cost by the number of passes an algorithm takes

**Main-memory bottleneck**

For many frequent-itemsets algorithms, main memory is a critical resource

* As we read baskets, we need to count something, for example, occurrences of pairs.
* The number of different things we can count is limited by main memory.
* Swapping counts in/out is a disaster.

**The Apriori principle**

* If an itemset is frequent, then all of its subsets must also be frequent
* If an itemset is not frequent, then all of its supersets cannot be frequent
* The support of an itemset never exceeds the support of its subsets
* This is known as the anti-monotone property

**The Apriori algorithm**

* Ck = candidate itemsets of size k
* Lk = frequent itemsets of size k
* k = 1, C1 = all items
* While Ck not empty
  + Scan the database to find which itemsets in Ck are frequent and put them into Lk
  + Use Lk to generate a collection of candidate itemsets Ck+1 of size k+1
  + k++

**Candidate generation**

* Basic principle:
  + An itemset of size k+1 is a candidate to be frequent only if all its subsets of size k are known to be frequent
* Main idea:
  + Construct a candidate of size k+1 by combining frequent itemsets of size k
    - If k = 1, take all the pairs of frequent items
    - If k > 1, join pairs of itemsets that differ by just one item
    - For each generated candidate itemset, ensure that all subsets of size k are frequent