COMP5331 - Knowledge Discovery in Databases

Taught by Prof. Raymond Wong

Notes by Aaron Wang

Contents

1 September 2nd, 2019

1.1 Association Rule Mining

Suppose we have the following dataset:

Customer	Shopping List		
Raymond	Apple	Coke	Coffee
David	Diaper	Coke	
Emily	Milk	Biscuit	
Derek	Coke	Milk	

Definition 1.1. The things on the RHS are **items**.

Definition 1.2. For each customer we have their **history** or **transaction**.

We want to find some **associations** between items. An example of an interesting association might be:

Example 1.3 (Example of an interesting association)
Diapers and Beers are usually bought together.

This association could have different reasons, e.g. people buy both diapers and beer after work usually.

1.2 Applications of Association Rule Mining

Here are some examples of where association rule mining might be used:

- Supermarket - For recommendation
- Web Mining - Google for their autocomplete
- Medical Analysis - Diagnosis from the patient's attributes or finding key attributes linked to illnesses (diabetes and obesity)
- Bioinformatics - Patterns in genomes
- Network Analysis - Associating IP and DoS, e.g. seeing if your packet goes through
- Programming Patern Finding - e.g. linking segmentation faults and users

1.3 Problem Definition

Consider the following dataset (TID = Transaction ID):

- TID: t_1, Items: A, D
- TID: t_2, Items: A, B, D, E

- TID: t_3, Items: B, C
- TID: t_4, Items: A, B, C, D, E
- TID: t_5, Items: B, C, E

In table form this would be:

TID	A	B	C	D	E
t_1	1	0	0	1	0
t_2	1	1	0	1	1
t_3	0	1	1	0	0
t_4	1	1	1	1	1
t_5	0	1	1	0	1

Definition 1.4. A **single item** is a single item (duh).

Example 1.5 (Examples of Single Items)
A, B, C, D, or E

Definition 1.6. An **itemset** is a set of items (again, duh).

Example 1.7 (Examples of Itemsets)
$\{B, C\}, \{A, B, C\}, \{B, C, D\}, \{A\}$

Definition 1.8. An **n-itemset** is a set of n items.

Example 1.9 (Examples of n-itemsets)
From Example 1.7, we have the following:

- $\{B, C\}$ is a 2-itemset
- $\{A, B, C\}$ and $\{B, C, D\}$ are 3-itemsets
- $\{A\}$ is a 1-itemset

Definition 1.10. The **support** (or **frequency**) of an item or an itemset is the number of times it appears in the dataset.

Example 1.11 (Examples of the Support of Items and Itemsets)
From Example 1.7, we have the following:

- The support of A is 3: $A \in t_1, t_2, t_4$
- The support of B is 4: $B \in t_2, t_3, t_4, t_5$
- The support of $\{B, C\}$ is 3: $\{B, C\} \subseteq t_3, t_4, t_5$
- The support of $\{A, B, C\}$ is 3: $\{A, B, C\} \subseteq t_4$

As such, we might try to classify large itemsets or **frequent itemsets** as itemsets with support greater than a threshold, e.g. 3.

Definition 1.12. An n-frequent itemset is an itemset with support n.

Example 1.13
$\{B, C\}$ is a 3-frequent itemset of size 2.

Definition 1.14. An **association rule** is a association between an item/itemset and another.

Definition 1.15. The **support** of an association rule is the number of transaction with both the LHS and RHS of the association rule.

Definition 1.16. The **confidence** of an association rule is the support of the association rule divided by the number of transaction with the LHS of the rule.

Example 1.17
$\{B, C\} \to E$ is an example of an association rule. It has a support of 3 (t_3, t_4, t_5) and a confidence of $\frac{2}{3}$ (it's true for t_3 and t_4 but not t_5).

In essence, we want to find association rules with:

- Support greater than a threshold e.g. (≥ 3)
- Confidence greater than a threshold e.g. $(\geq 50\%)$

We can do split this into two steps:

1. Find all "large" itemsets (e.g. itemsets with support ≥ 3)
2. Find all "interesting" association rules after Step 1:
    - From all "large" itemsets, find the association rules with confidence $\geq 50\%$.
    - This can be done by taking every pair of elements from Step 1, X and Y, where $X \subset Y$, and checking if $\frac{\text{supp}(Y)}{\text{supp}(X)} \geq 50\%$.
    - If yes, then generate the rule: "$X \to Y - X$"

Homework

Show that the support of the association rule is still large. This can be easily seen, as X is large, and $Y - X \subseteq Y$, making it large from 2.4

2 September 4th, 2019

2.1 NP-Completeness of Finding Large Itemsets

The problem of finding all "large" itemsets is equivelent to finding all "large" J-itemsets for each positive integer J. (for our case this is finding J-itemsets with support ≥ 3).

However, this problem is NP-Complete, as it is equivelent to solving the **Balanced Complete bipartite Subgraph**, a known NP-Complete problem.

Definition 2.1. The **Balanced Complete Bipartitie Subgraph** is as follows:

- Instance: Given a bipartite graph $G = (V, E)$ and positive integer $K \leq |V|$
- Question: Are there two disjoint subsets $V_1, V_2 \subseteq V$ such that $|V_1| = |V_2| = K$ and such that, for each $u \in V_1$ and each $v \in V_2$, $(u, v) \in E$.

Figure 1: $V_1 = \{A, B, C\}, V_2 = \{E, F, G\}$ for the left, no such V_1, V_2 exists for the right

The reduction from the graph problem to the itemset problem is as follows:

- For each vertex in V_1, create a transaction
- For each vertex in V_2, create a item
- For each edge (u, v), create a purchase of item v in transaction u
- We have $f = K$ and $J = K$

Remark 2.2 — In other words, solving the graph problem is equivelent tothe itemset problem, i.e. is there a K-frequent itemset of size K. Since this is a restriction of the itemset problem, if we can solve the itemset problem, we can solve the Balanced Complete Bipartite Subgraph problem. As such, the itemset problem is also NP-Complete, since BCBS is NP-Complete.

Remark 2.3 — NP-Complete means that there is no polynomial time algorithm to solve it (unless P=NP).

2.2 Algorithm Aprior

This algorithm starts with "large" 1-itemsets, and iteratively builds "large" itemsets with bigger sizes (1-itemsets→ 2-itemset→ ...). It can be described as follows:

- Start with $L_1 = \{\{A\}, \{B\}, \{C\}, \{D\}, \{E\}\}$, i.e. the large 1-itemsets.
- From L_1, generate candidates for large 2-itemsets, C_2.
- From C_2, check and keep the large 2-itemsets, L_2.
- Repeat

To generate candidates for "large" n-itemsets from L_{n-1}, we can take note of the following properties:

Theorem 2.4

If an itemset S is large, then any proper subset of S must be large.

Proof. Note that the proper subsets of S are a relaxed version of S. Since we are relaxing the constraint, this property must be true. □

Theorem 2.5

If an itemset S is NOT large, then any proper superset of S must NOT be large.

Proof. Similarly, since the proper supersets are restricted versions of S, if S is not large, then any proper supersets of S must not be large, since it's even more restricted. □

Example 2.6

$\{B, C, E\}$ is large, thus $\{B, C\}, \{C, E\}\{C\}$ are all large (not exahstive).

Using these properties, we can split the generation step into two steps: Suppose we know that the itemset B, C and the itemset B, E are large (i.e. in L_2)