FUNDAMENTALS OF MACHINE LEARNING IN DATA SCIENCE

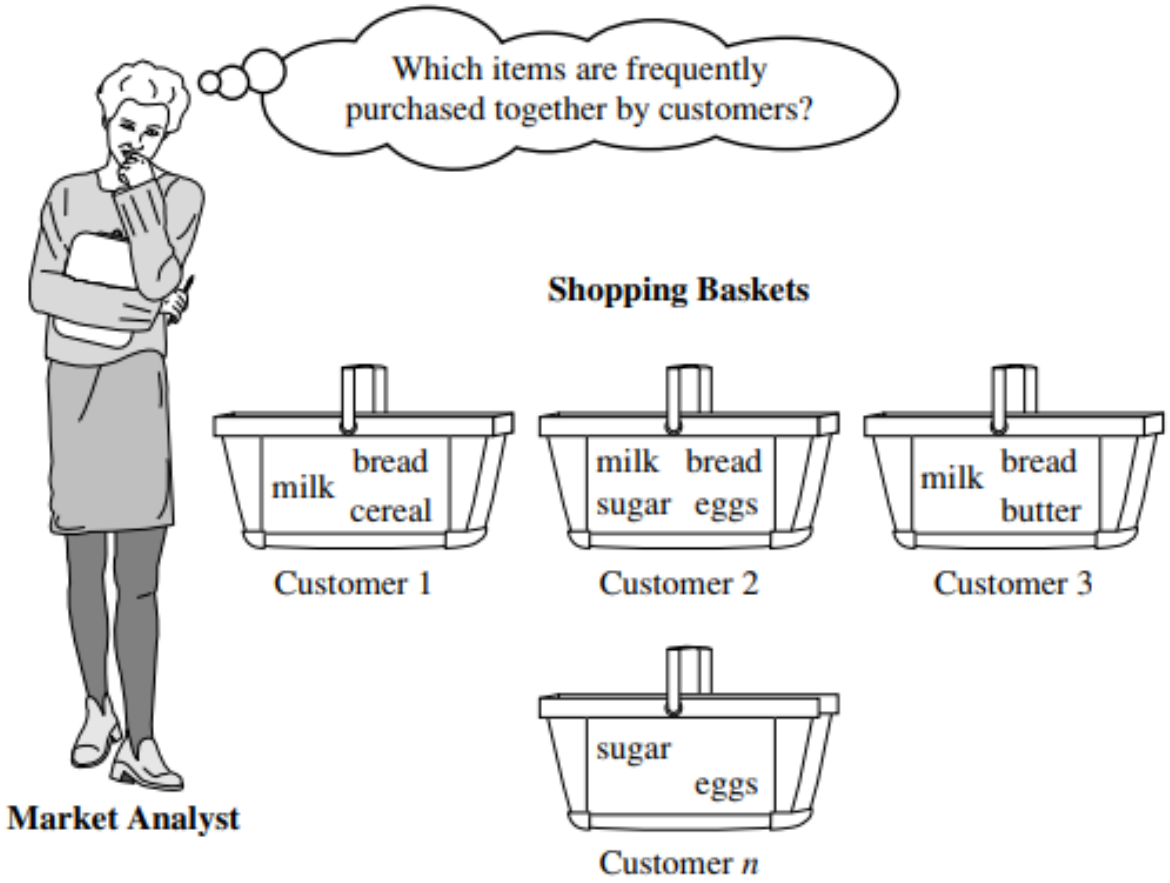
CSIS 3290

FREQUENT PATTERN MINING

(MARKET BASKET ANALYSIS)

FATEMEH AHMADI

Market Basket Analysis



Market basket analysis.

Market Basket Analysis

- Frequent itemset mining leads to the discovery of associations and correlations among items in large transactional or relational data sets. With massive amounts of data continuously being collected and stored, many industries are becoming interested in mining such patterns from their databases.
- The discovery of interesting correlation relationships among huge amounts of business transaction records can help in many business decision-making processes such as catalog design, cross-marketing, and customer shopping behavior analysis.
- A typical example of **frequent itemset mining** is **market basket analysis**. This process analyzes customer buying habits by finding associations between the different items that customers place in their "shopping baskets".
- The discovery of these associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers. For instance, if customers are buying milk, how likely are they to also buy bread (and what kind of bread) on the same trip to the supermarket?
- This information can lead to increased sales by helping retailers do selective marketing and plan their shelf space.

If we think of the universe as the set of items available at the store, then each item has a Boolean variable representing the presence or absence of that item. Each basket can then be represented by a Boolean vector of values assigned to these variables. The Boolean vectors can be analyzed for buying patterns that reflect items that are frequently *associated* or purchased together. These patterns can be represented in the form of **association rules**. For example, the information that customers who purchase computers also tend to buy antivirus software at the same time is represented in the following association rule:

$$computer \Rightarrow antivirus\_software\ [support = 2\%, confidence = 60\%]. \qquad (6.1)$$

Rule **support** and **confidence** are two measures of rule interestingness. They respectively reflect the usefulness and certainty of discovered rules. A support of 2% for Rule (6.1) means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together. A confidence of 60% means that 60% of the customers who purchased a computer also bought the software. Typically, association rules are considered interesting if they satisfy both a **minimum support threshold** and a **minimum confidence threshold**. These thresholds can be a set by users or domain experts. Additional analysis can be performed to discover interesting statistical correlations between associated items.

$$support(A \Rightarrow B) = P(A \cup B) \tag{6.2}$$

$$confidence(A \Rightarrow B) = P(B|A). \tag{6.3}$$

Rules that satisfy both a minimum support threshold (*min_sup*) and a minimum confidence threshold (*min_conf*) are called **strong**. By convention, we write support and confidence values so as to occur between 0% and 100%, rather than 0 to 1.0.

A set of items is referred to as an **itemset**.[2] An itemset that contains k items is a ***k*-itemset**. The set {*computer*, *antivirus_software*} is a 2-itemset. The **occurrence frequency of an itemset** is the number of transactions that contain the itemset. This is also known, simply, as the **frequency**, **support count**, or **count** of the itemset. Note that the itemset support defined in Eq. (6.2) is sometimes referred to as *relative support*, whereas the occurrence frequency is called the **absolute support**. If the relative support of an itemset I satisfies a prespecified **minimum support threshold** (i.e., the absolute support of I satisfies the corresponding **minimum support count threshold**), then I is a **frequent** itemset.[3] The set of frequent k-itemsets is commonly denoted by L_k.[4]

From Eq. (6.3), we have

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} = \frac{support\_count(A \cup B)}{support\_count(A)}. \tag{6.4}$$

Apriori Algorithm: Finding Frequent Itemsets by Confined Candidate Generation

- **Apriori** is a seminal algorithm proposed by *R. Agrawal* and *R. Srikant* in 1994 for mining frequent itemsets for Boolean association rules. The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties, as we shall see later.
- Apriori employs an iterative approach known as a level-wise search, where k-itemsets are used to explore (k + 1)-itemsets. First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support.
- The resulting set is denoted by L1. Next, L1 is used to find L2, the set of frequent 2-itemsets, which is used to find L3, and so on, until no more frequent k-itemsets can be found. The finding of each Lk requires one full scan of the database. To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the **Apriori property** is used to reduce the search space.

Apriori Algorithm: Finding Frequent Itemsets by Confined Candidate Generation

1. **The join step**: To find L_k, a set of **candidate** k-itemsets is generated by joining L_{k-1} with itself. This set of candidates is denoted C_k. Let l_1 and l_2 be itemsets in L_{k-1}. The notation $l_i[j]$ refers to the jth item in l_i (e.g., $l_1[k-2]$ refers to the second to the last item in l_1). For efficient implementation, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. For the $(k-1)$-itemset, l_i, this means that the items are sorted such that $l_i[1] < l_i[2] < \cdots < l_i[k-1]$. The join, $L_{k-1} \bowtie L_{k-1}$, is performed, where members of L_{k-1} are joinable if their first $(k-2)$ items are in common. That is, members l_1 and l_2 of L_{k-1} are joined if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \cdots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$. The condition $l_1[k-1] < l_2[k-1]$ simply ensures that no duplicates are generated. The resulting itemset formed by joining l_1 and l_2 is $\{l_1[1], l_1[2], \ldots, l_1[k-2], l_1[k-1], l_2[k-1]\}$.

Apriori Algorithm: Finding Frequent Itemsets by Confined Candidate Generation

2- **The prune step**: Ck is a superset of Lk , that is, its members may or may not be frequent, but all of the frequent k-itemsets are included in Ck . A database scan to determine the count of each candidate in Ck would result in the determination of Lk (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to Lk). Ck , however, can be huge, and so this could involve heavy computation. To reduce the size of Ck , the Apriori property is used as follows. Any (k − 1)-itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any (k − 1)-subset of a candidate k-itemset is not in Lk−1, then the candidate cannot be frequent either and so can be removed from Ck . This subset testing can be done quickly by maintaining a hash tree of all frequent itemsets.

Apriori

Transaction _ Id	List of Item_Ids
T100	I2, I1, I5
T200	I2, I4
T300	I2, I3
T400	I2, I1, I4
T500	I1, I3
T600	I2, I3
T700	I1,I3
T800	I2, I1, I3, I5
T900	I2,I1, I3

Minimum Support count:2

Compare with minimum support count

C1

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L1

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Apriori

Itemset	Sup.Count
{I1,I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

C2

→ **Compare with minimum support count**

Itemset	Sup.Count
{I1,I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

L2

{I1,I2,I3}: {I1, I2}, {I1, I3}, {I2, I3}**ε** L2 → in C3

{I2,I3,I5}: {I2, I3}, {I2, I5}, **{I3, I5} ε** L2 → not in C3

{I1,I2,I3,I5}: {I1,I2,I3}, {I1,I2, I5}, **{I2, I3,I5} ε** L3 → not in C4 → **Finish**

Itemset	Sup.Count
{I1,I2,I3}	2
{I1,I2,I5}	2

C3

→ **Compare with minimum support count**

Itemset	Sup.Count
{I1,I2,I3}	2
{I1,I2,I5}	2

L3

Association Rules

Confidence (A → B) = P(B|A) = Support(A ∪ B) / Support(A)

1: {I1, I2}→ {I5} Confidence: 2/4 = 50%
2: {I1, I5}→ {I2} Confidence : 2/2 = 100%
3: {I2, I5}→ {I1} Confidence : 2/2 = 100%
4: {I1}→ {I2, I5} Confidence : 2/6 = 33%
5: {I2}→ {I1, I5} Confidence : 2/7 = 29%
6: {I5}→ {I1, I2} Confidence : 2/2 = 100%

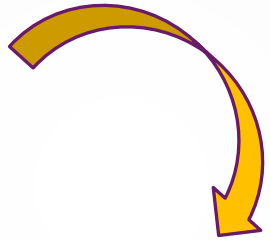
Minimum Confidence Threshold:70%

Strong Association Rules

2: {I1, I5}→ {I2} Confidence : 2/2 = 100%
3: {I2, I5}→ {I1} Confidence : 2/2 = 100%
6: {I5}→ {I1, I2} Confidence : 2/2 = 100%

Eclat (Equivalence Class Transform)

Transaction _ Id	List of Item_Ids
T100	I2, I1, I5
T200	I2, I4
T300	I2, I3
T400	I2, I1, I4
T500	I1, I3
T600	I2, I3
T700	I1,I3
T800	I2, I1, I3, I5
T900	I2,I1, I3



Itemset	TID_Set
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

1

Eclat (Equivalence Class Transform)

Itemset	TID_Set
{I1,I2}	{T100, T400, T800, T900}
{I1,I3}	{T500, T700, T800, T900}
{I1,I4}	{T400}
{I1,I5}	{T100, T800}
{I2,I3}	{T300, T600, T800, T900}
{I2,I4}	{T200, T400}
{I2,I5}	{T100, T800}
{I3,I4}	{ }
{I3,I5}	{T800}
{I4,I5}	{ }

2: The 2-itemsets

Minimum Support count:2

Itemset	TID_Set
{I1,I2,I3}	{T800, T900}
{I1,I2,I5}	{T100, T800}

3: The 3-itemsetd

From Association Analysis to Correlation Analysis

	C_Game	C_Game	$\sum_{row}$
V_Game	4000	3500	7500
V_Game	2000	500	2500
$\sum_{col}$	6000	4000	10000

Buying (C_Game) → Buying (V_Game) {*Confidence: 66%, Support: 40%*}

Correlation Analysis using *Lift*

Buying (C_Game) → Buying (V_Game) {*Confidence: 66%, Support: 40%*}

	C_Game	$\overline{\text{C\_Game}}$	$\sum_{row}$
V_Game	4000	3500	7500
$\overline{\text{V\_Game}}$	2000	500	2500
$\sum_{col}$	6000	4000	10000

Contingency Table

$$Lift(A,B) = \frac{P(A \cup B)}{P(A)p(B)} = \frac{(\frac{4000}{10000})}{(\frac{6000}{10000}) * (\frac{7500}{10000})} = 0.89$$

(Confidence (A → B) / Support (B)) = P(B|A) / P(B)

Because the result of *Lift* is less than one, so there is a <u>negative correlation</u> between the occurrence of *C_Game* and *V_Game* which could not be discovered with *Support and Confidence* values.

Correlation Analysis using χ^2

	C_Game	$\overline{\text{C\_Game}}$	$\sum_{row}$
V_Game	4000	3500	7500
$\overline{\text{V\_Game}}$	2000	500	2500
$\sum_{col}$	6000	4000	10000

Contingency Table

	C_Game	$\overline{\text{C\_Game}}$	$\sum_{row}$
V_Game	4000 (4500)	3500 (3000)	7500
$\overline{\text{V\_Game}}$	2000 (1500)	500 (1000)	2500
$\sum_{col}$	6000	4000	10000

$$\text{Expected}_{4000} = \frac{6000 * 7500}{10000} = 4500$$

$$\text{Expected}_{3500} = \frac{4000 * 7500}{10000} = 3000$$

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected} = \frac{(4000-4500)^2}{4500} + \frac{(3500-3000)^2}{3000} + \frac{(2000-1500)^2}{1500} + \frac{(500-1000)^2}{1000} = 555.6$$

Result of χ^2 is greater than ten and in the slot *C-Game* and *V_Game* the expected value is greater than the observed value so *C-Game* and *V_Game* are negatively correlated.

Other Correlation Measures

1 $$All\_Confidence(A,B) = \frac{Sup(A \cup B)}{Min\{Sup(A), Sup(B)\}} = \text{Min}\{P(A|B), P(B|A)\}$$

2 $$Max\_Confidence(A,B) = Max\{P(A|B), P(B|A)\}$$

3 $$Kulczynski(A,B) = (1/2)(P(A|B) + P(B|A))$$

or *Harmonized Lift*

4 $$Cosine(A,B) = \frac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \frac{Sup(A \cup B)}{\sqrt{Sup(A) \times Sup(B)}} = \sqrt{P(A|B) \times P(B|A)}$$

The results of All measures is between zero and one. If it is close to one, A and B are positively correlated and if the result id less than 0.5 they are negatively correlated.

Reference

Data Mining, Concepts and Techniques, Jiawei Han, Micheline Kamber, Jian Pei. MK. Chapter 6.