# Lesson B-2

## Frequent Pattern Mining

-frequent patterns, Apriori algorithms, mining association rules, and correlation rules

# Frequent Patterns

- Frequent pattern mining searches for recurring relationships in a given data set
- Frequent pattern
  - patterns (e.g., itemset, sequences, or structures) that appear frequently in a dataset.
    - milk and bread, that appear frequently together in a transaction
    - buying first a smartphone, then a phone case, and then a memory card, if it occurs frequently in a shopping history database

# Frequent Pattern Analysis

- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, recommender systems, DNA sequence analysis, etc.

# Basic Concepts of Frequent Patterns

- Itemset
- k-itemset
- the occurrence frequency of an itemset (frequency, support count or count of the itemset)
- frequent itemset
- closed frequent itemset
- maximal frequent itemset
- (relative) support
- (absolute) support
- minimum support threshold (count)
- association rules
- confidence
- minimum confidence threshold (count)
- strong rules

# Item, Itemset, k-itemset, a set of itemset, a transaction database

| TID | Items bought |
|---|---|
| 1 | A, B, C, E |
| 2 | A, C, D, E |
| 3 | B, C, E |
| 4 | A, C, D, E |
| 5 | C, D, E |
| 6 | A, D, E |

# The occurrence frequency of an itemset

| TID | Items bought |
|---|---|
| 1 | A, B, C, E |
| 2 | A, C, D, E |
| 3 | B, C, E |
| 4 | A, C, D, E |
| 5 | C, D, E |
| 6 | A, D, E |

{A}
{B}
{C}
{D}
{E}

{A,B}
{A,C}
{A,D}
{A,E}
{B,C}
{B,D}
{B,E}
{C,D}
{C,E}
{D,E}

{A,B,C}
{A,B,D}
{A,B,E}
{A,C,D}
{A,C,E}
{A,D,E}
{B,C,D}
{B,C,E}
{C,D,E}

{A,B,C,D}
{A,B,C,E}
{B,C,D,E}

# Minimum support count and frequent itemset

minimum support count = 3

| TID | Items bought |
|---|---|
| 1 | A, B, C, E |
| 2 | A, C, D, E |
| 3 | B, C, E |
| 4 | A, C, D, E |
| 5 | C, D, E |
| 6 | A, D, E |

{A} = 4
{B} = 2
{C} = 5
{D} = 4
{E} = 6

{A,B} = 1
{A,C} = 3
{A,D} = 3
{A,E} = 4
{B,C} = 2
{B,D} = 0
{B,E} = 2
{C,D} = 3
{C,E} = 5
{D,E} = 4

{A,B,C} = 1
{A,B,D} = 0
{A,B,E} = 1
{A,C,D} = 2
{A,C,E} = 3
{A,D,E} = 3
{B,C,D} = 0
{B,C,E} = 2
{C,D,E} = 3

{A,B,C,D} = 0
{A,B,C,E} = 1
{B,C,D,E} = 0

# Closed frequent itemset and maximal frequent itemset (max itemset)

Frequent itemset $X \in D$ is closed if it has no superset with the same frequency.
Frequent itemset $X \in D$ is maximal if it does not have any frequent supersets.

| TID | Items bought |
|---|---|
| 1 | A, B, C, E |
| 2 | A, C, D, E |
| 3 | B, C, E |
| 4 | A, C, D, E |
| 5 | C, D, E |
| 6 | A, D, E |

{A} = 4
~~{B} = 2~~
{C} = 5
{D} = 4
{E} = 6

~~{A,B} = 1~~
{A,C} = 3
{A,D} = 3
{A,E} = 4
~~{B,C} = 2~~
~~{B,D} = 0~~
~~{B,E} = 2~~
{C,D} = 3
{C,E} = 5
{D,E} = 4

~~{A,B,C} = 1~~
~~{A,B,D} = 0~~
~~{A,B,E} = 1~~
~~{A,C,D} = 2~~
{A,C,E} = 3
{A,D,E} = 3
~~{B,C,D} = 0~~
~~{B,C,E} = 2~~
{C,D,E} = 3

~~{A,B,C,D} = 0~~
~~{A,B,C,E} = 1~~
~~{B,C,D,E} = 0~~

# How to Generate Frequent Itemset?

- Let $I = \{I_1, I_2, ...., I_m\}$ be a set of items
- Let $D$, the task-relevant data, be a set of database transactions where each transaction $T$ is a set of items such that $T \subseteq I$
- Each transaction is associated with an identifier, called *TID*.
- Let $A$ be a set of items
- A transaction $T$ is said to contain $A$ if and only if $A \subseteq T$

# How to Generate Frequent Itemset?

- Suppose the items in $L_{k-1}$ are listed in an order
- **The join step**:
  - To find $L_k$, a set of candidate $k$-itemsets, $C_k$, is generated by joining $L_{k-1}$ with itself.
    - Let $l_1$ and $l_2$ be itemsets in $L_{k-1}$.
    - The resulting itemset formed by joining $l_1$ and $l_2$ is $l_1[1], l_1[2], \ldots, l_1[k-2], l_1[k-1], l_2[k-1]$
- **The prune step**:
  - Scan data set $D$ and compare candidate support count of $C_k$ with minimum support count.
  - Remove candidate itemsets that whose support count is less than minimum support count, resulting in $L_k$.

# Apriori Algorithm

- Initially, scan DB once to get frequent 1-itemset
- Generate length *(k+1)* candidate itemsets by joining length *k* frequent itemsets
- Prune length *(k+1)* candidate itemsets **with Apriori property**
  - **Apriori property: All nonempty subsets of a frequent itemset must also be frequent**
- Test the candidates against DB
- Terminate when no frequent or candidate set can be generated

**Algorithm: Apriori.** Find frequent itemsets using an iterative level-wise approach based on candidate generation.

**Input:**

- $D$, a database of transactions;
- $min\_sup$, the minimum support count threshold.

**Output:** $L$, frequent itemsets in $D$.

**Method:**

```
(1)     L1 = find_frequent_1-itemsets(D);
(2)     for (k = 2; L_{k-1} ≠ φ; k++) {
(3)        C_k = apriori_gen(L_{k-1});
(4)        for each transaction t ∈ D { // scan D for counts
(5)            C_t = subset(C_k, t); // get the subsets of t that are candidates
(6)            for each candidate c ∈ C_t
(7)                c.count++;
(8)        }
(9)        L_k = {c ∈ C_k | c.count ≥ min_sup}
(10)    }
(11)    return L = ∪_k L_k;

procedure apriori_gen(L_{k-1}:frequent (k − 1)-itemsets)
(1)     for each itemset l1 ∈ L_{k-1}
(2)        for each itemset l2 ∈ L_{k-1}
(3)            if (l1[1] = l2[1]) ∧ (l1[2] = l2[2])
                   ∧ ... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
(4)                c = l1 ⋈ l2; // join step: generate candidates
(5)                if has_infrequent_subset(c, L_{k-1}) then
(6)                    delete c; // prune step: remove unfruitful candidate
(7)                else add c to C_k;
(8)            }
(9)     return C_k;

procedure has_infrequent_subset(c: candidate k-itemset;
           L_{k-1}: frequent (k − 1)-itemsets); // use prior knowledge
(1)     for each (k − 1)-subset s of c
(2)        if s ∉ L_{k-1} then
(3)            return TRUE;
(4)     return FALSE;
```

Apriori algorithm for discovering frequent itemsets for mining Boolean association rules.

# Transactional Database

| ***TID*** | ***List of item_IDs*** |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

Transactional data for an AllElectronics branch.

Scan $D$ for count of each candidate

$C_1$

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

Compare candidate support count with minimum support count

$L_1$

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

Minimum support count = 2

Generate $C_2$ candidates from $L_1$

$C_2$

| Itemset |
|---|
| {I1, I2} |
| {I1, I3} |
| {I1, I4} |
| {I1, I5} |
| {I2, I3} |
| {I2, I4} |
| {I2, I5} |
| {I3, I4} |
| {I3, I5} |
| {I4, I5} |

Scan $D$ for count of each candidate

$C_2$

| 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 |

Compare candidate support count with minimum support count

$L_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |

Generate $C_3$ candidates from $L_2$

$C_3$

| Itemset |
|---|
| {I1, I2, I3} |
| {I1, I2, I5} |

Scan $D$ for count of each candidate

$C_3$

| Itemset | Sup. count |
|---|---|
| {I1, I2, I3} | 2 |
| {I1, I2, I5} | 2 |

Compare candidate support count with minimum support count

$L_3$

| Itemset | Sup. count |
|---|---|
| {I1, I2, I3} | 2 |
| {I1, I2, I5} | 2 |

Generation of candidate itemsets and frequent itemsets, where the minimum support count is 2.

# Rules

- Rule-based expert system
  - *If-then* rules
  - *X => Y*

# Association Rules

- In data mining, association rules are "if-then" statements, that help to show the probability of relationships between data items or attributes within large data sets.
  - Discover relations between attributes or data items within large dataset.

# Association Rules

- The rule *X => Y is called a* *<u>strong</u>* *association rule*
  - *when it satisfies a prespecified* ***minimum support threshold and*** *a prespecified* ***minimum confidence threshold.***

# Association Rules

- The association rule X => Y holds in the transaction set D with support s,
  - where s is the percentage of transactions in D that contain X U Y (i.e., the union of sets X and Y say, or, both X and Y).

# Support

- The rule *A*⇒ *B* holds in the transaction set *D* with *support **s***
  - *support*, ***s***, probability that a transaction contains *A* and *B*
  - *support (A⇒B) = support (A U B) = P (A ∪ B), range:[0,1]*

# Confidence

- The rule $A \Rightarrow B$ has **confidence *c*** in the transaction set *D*,
  - where *c* is the percentage of transactions in *D* containing *A* that also contain *Y*.
  - *confidence*, ***c,*** conditional probability that a transaction having *A* also contains *Y*
  - *confidence (A⇒ B) = P (B | A), range: [0,1]*
    - *confidence* $(A \Rightarrow B) = P(B|A) = P(A \cup B) / P(A)$

      $= support\ (A \cup B) / support(A)$

      $= support\_count(A \cup B) / support\_count(A)$

# Mining Association Rules

- The ***association rule mining*** can be viewed as a two-step process
  - **Find all frequent itemsets:**
    - By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, *min_sup*.
  - **Generate strong association rules from the frequent itemsets:**
    - By definition, these rules must satisfy minimum support and minimum confidence.

# Association Rules

- Generating Association Rules from Frequent Itemsets
  - *for each frequent itemset l, generate all nonempty subset of l*
  - *For every nonempty subset s of l,*

    *Output the rule "s $\Rightarrow$ (l – s)" If support_count(l) / support_count(s) $\geq$ min_confidence, where min_confidence is the minimum confidence threshold*

- Rules that satisfy both a minimum support threshold and a minimum confidence threshold are called strong

# Generating Association Rules from Frequent Itemsets

- Suppose the data contain the frequent itemset $l = \{I1, I2, I5\}$. What are the association rules that can be generated from $l$? If the minimum confidence threshold is 70%, then which rules are strong?

# Frequent Patterns -- Apriori Agorithm

```
dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
           ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
           ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]
```

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
te_ary
```

```
[2]: array([[False, False, False,  True, False,  True,  True,  True,  True,
         False,  True],
        [False, False,  True,  True, False,  True, False,  True,  True,
         False,  True],
        [ True, False, False,  True, False,  True,  True, False, False,
         False, False],
        [False,  True, False, False, False,  True,  True, False, False,
          True,  True],
        [False,  True, False,  True,  True,  True, False, False,  True,
         False, False]])
```

```
te.columns_
```

```
]: ['Apple',
 'Corn',
 'Dill',
 'Eggs',
 'Ice cream',
 'Kidney Beans',
 'Milk',
 'Nutmeg',
 'Onion',
 'Unicorn',
 'Yogurt']
```

```
te_ary.astype("int")
```

```
[3]: array([[0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1],
        [0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1],
        [1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0],
        [0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1],
        [0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0]])
```

```
te.columns_
```

```
['Apple',
 'Corn',
 'Dill',
 'Eggs',
 'Ice cream',
 'Kidney Beans',
 'Milk',
 'Nutmeg',
 'Onion',
 'Unicorn',
 'Yogurt']
```

```
df = pd.DataFrame(te_ary, columns=te.columns_)
df
```

| | Apple | Corn | Dill | Eggs | Ice cream | Kidney Beans | Milk | Nutmeg | Onion | Unicorn | Yogurt |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | False | False | False | True | False | True | True | True | True | False | True |
| 1 | False | False | True | True | False | True | False | True | True | False | True |
| 2 | True | False | False | True | False | True | True | False | False | False | False |
| 3 | False | True | False | False | False | True | True | False | False | True | True |
| 4 | False | True | False | True | True | True | False | False | True | False | False |

```
first4 = te_ary[:4]
te.inverse_transform(first4)
```

```
[['Eggs', 'Kidney Beans', 'Milk', 'Nutmeg', 'Onion', 'Yogurt'],
 ['Dill', 'Eggs', 'Kidney Beans', 'Nutmeg', 'Onion', 'Yogurt'],
 ['Apple', 'Eggs', 'Kidney Beans', 'Milk'],
 ['Corn', 'Kidney Beans', 'Milk', 'Unicorn', 'Yogurt']]
```

```python
from mlxtend.frequent_patterns import apriori

apriori(df, min_support=0.6)
```

| | support | itemsets |
|---|---|---|
| 0 | 0.8 | (3) |
| 1 | 1.0 | (5) |
| 2 | 0.6 | (6) |
| 3 | 0.6 | (8) |
| 4 | 0.6 | (10) |
| 5 | 0.8 | (3, 5) |
| 6 | 0.6 | (8, 3) |
| 7 | 0.6 | (5, 6) |
| 8 | 0.6 | (8, 5) |
| 9 | 0.6 | (10, 5) |
| 10 | 0.6 | (8, 3, 5) |

```python
apriori(df, min_support=0.6, use_colnames=True)
```

| | support | itemsets |
|---|---|---|
| 0 | 0.8 | (Eggs) |
| 1 | 1.0 | (Kidney Beans) |
| 2 | 0.6 | (Milk) |
| 3 | 0.6 | (Onion) |
| 4 | 0.6 | (Yogurt) |
| 5 | 0.8 | (Eggs, Kidney Beans) |
| 6 | 0.6 | (Eggs, Onion) |
| 7 | 0.6 | (Milk, Kidney Beans) |
| 8 | 0.6 | (Onion, Kidney Beans) |
| 9 | 0.6 | (Yogurt, Kidney Beans) |
| 10 | 0.6 | (Eggs, Onion, Kidney Beans) |

```python
frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
```

| | support | itemsets | length |
|---|---|---|---|
| 0 | 0.8 | (Eggs) | 1 |
| 1 | 1.0 | (Kidney Beans) | 1 |
| 2 | 0.6 | (Milk) | 1 |
| 3 | 0.6 | (Onion) | 1 |
| 4 | 0.6 | (Yogurt) | 1 |
| 5 | 0.8 | (Eggs, Kidney Beans) | 2 |
| 6 | 0.6 | (Eggs, Onion) | 2 |
| 7 | 0.6 | (Milk, Kidney Beans) | 2 |
| 8 | 0.6 | (Onion, Kidney Beans) | 2 |
| 9 | 0.6 | (Yogurt, Kidney Beans) | 2 |
| 10 | 0.6 | (Eggs, Onion, Kidney Beans) | 3 |

```python
frequent_itemsets[ (frequent_itemsets['length'] == 2) &
                   (frequent_itemsets['support'] >= 0.8) ]
```

| | support | itemsets | length |
|---|---|---|---|
| 5 | 0.8 | (Eggs, Kidney Beans) | 2 |

```python
frequent_itemsets[ (frequent_itemsets['length'] == 3) &
                   (frequent_itemsets['support'] >= 0.6) ]
```

| | support | itemsets | length |
|---|---|---|---|
| 10 | 0.6 | (Eggs, Onion, Kidney Beans) | 3 |

# Association Rules

```
from mlxtend.frequent_patterns import association_rules

association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
```

| | antecedents | consequents | antecedent support | consequent support | support | confidence |
|---|---|---|---|---|---|---|
| 0 | (Eggs) | (Kidney Beans) | 0.8 | 1.0 | 0.8 | 1.00 |
| 1 | (Kidney Beans) | (Eggs) | 1.0 | 0.8 | 0.8 | 0.80 |
| 2 | (Eggs) | (Onion) | 0.8 | 0.6 | 0.6 | 0.75 |
| 3 | (Onion) | (Eggs) | 0.6 | 0.8 | 0.6 | 1.00 |
| 4 | (Milk) | (Kidney Beans) | 0.6 | 1.0 | 0.6 | 1.00 |
| 5 | (Onion) | (Kidney Beans) | 0.6 | 1.0 | 0.6 | 1.00 |
| 6 | (Yogurt) | (Kidney Beans) | 0.6 | 1.0 | 0.6 | 1.00 |
| 7 | (Eggs, Onion) | (Kidney Beans) | 0.6 | 1.0 | 0.6 | 1.00 |
| 8 | (Eggs, Kidney Beans) | (Onion) | 0.8 | 0.6 | 0.6 | 0.75 |
| 9 | (Onion, Kidney Beans) | (Eggs) | 0.6 | 0.8 | 0.6 | 1.00 |
| 10 | (Eggs) | (Onion, Kidney Beans) | 0.8 | 0.6 | 0.6 | 0.75 |
| 11 | (Onion) | (Eggs, Kidney Beans) | 0.6 | 0.8 | 0.6 | 1.00 |

# Misleading “strong” association rule

- Analyzing transactions at *AllElectronics* with respect to the purchase of computer games and videos.
  - 10,000 transactions analyzed
  - 6000 of the customer transactions included computer games
  - 7500 included videos
  - 4000 included both computer games and videos
  - minimum support: 30%
  - minimum confidence: 60%
  - *buys(X, “computer games*” => *buys (X, “videos*”)
    - *support = 40%, confidence = 66%*

# From Association Analysis to Correlation Analysis

- A correlation measure can be used to augment the support–confidence framework for association rules
- *Correlation rules*
  - *A=>B* [*support, confidence*, *correlation (lift, leverage, conviction)]*
  - A correlation rule is measured not only by its support and confidence but also by the correlation between itemsets *A* and *B*

# Lift

- **Lift** is a simple correlation measure
- The occurrence of itemset A is independent of the occurrence of itemset B if *P(A U B) = P(A)P(B)*
  - otherwise, itemsets A and B are dependent and correlated as events.
- The Lift between the occurrence of A and B can be measured by computing
  - *lift (A=>B) = P(A U B) / P(A)P(B) = P(B|A)/P(B) = confidence(A=>B)/support(B), range: [0, $\infty$]*
    - *If the resulting value is greater than 1, then A and B are positively correlated,*
    - *If the resulting value is less than 1, then A and B are negatively correlated,*
    - *If the resulting value is equal to 1, then A and B are independent and there is no correlation between them.*

```python
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
rules
```

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
|---|---|---|---|---|---|---|---|---|---|
| 0 | (Eggs) | (Onion) | 0.8 | 0.6 | 0.6 | 0.75 | 1.25 | 0.12 | 1.6 |
| 1 | (Onion) | (Eggs) | 0.6 | 0.8 | 0.6 | 1.00 | 1.25 | 0.12 | inf |
| 2 | (Eggs, Kidney Beans) | (Onion) | 0.8 | 0.6 | 0.6 | 0.75 | 1.25 | 0.12 | 1.6 |
| 3 | (Onion, Kidney Beans) | (Eggs) | 0.6 | 0.8 | 0.6 | 1.00 | 1.25 | 0.12 | inf |
| 4 | (Eggs) | (Onion, Kidney Beans) | 0.8 | 0.6 | 0.6 | 0.75 | 1.25 | 0.12 | 1.6 |
| 5 | (Onion) | (Eggs, Kidney Beans) | 0.6 | 0.8 | 0.6 | 1.00 | 1.25 | 0.12 | inf |

# Leverage

- Leverage computes the difference between the observed frequency of A and B appearing together and the frequency that would be expected if A and B were independent.
  - *leverage (A=>B) = support(A=>B) – support(A) x support(B),*
  - *range: [-1, +1]*
  - leverage value of 0 indicates independence

```
rules = association_rules(frequent_itemsets, metric="leverage", min_threshold=0.12)
rules
```

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
|---|---|---|---|---|---|---|---|---|---|
| 0 | (Eggs) | (Onion) | 0.8 | 0.6 | 0.6 | 0.75 | 1.25 | 0.12 | 1.6 |
| 1 | (Onion) | (Eggs) | 0.6 | 0.8 | 0.6 | 1.00 | 1.25 | 0.12 | inf |
| 2 | (Eggs, Kidney Beans) | (Onion) | 0.8 | 0.6 | 0.6 | 0.75 | 1.25 | 0.12 | 1.6 |
| 3 | (Onion, Kidney Beans) | (Eggs) | 0.6 | 0.8 | 0.6 | 1.00 | 1.25 | 0.12 | inf |
| 4 | (Eggs) | (Onion, Kidney Beans) | 0.8 | 0.6 | 0.6 | 0.75 | 1.25 | 0.12 | 1.6 |
| 5 | (Onion) | (Eggs, Kidney Beans) | 0.6 | 0.8 | 0.6 | 1.00 | 1.25 | 0.12 | inf |

# *Conviction*

- A high conviction value means that the consequent is highly depending on the antecedent.
  - *confidence (A=>B) =  (1 – support(B)) / (1 –  confidence (A,B))*
  - *range: [0, ∞]*
  - For instance, in the case of a perfect confidence score, the denominator becomes 0 (due to 1 - 1) for which the conviction score is defined as 'inf’
  - if items are independent, the conviction is 1.

```python
rules = association_rules(frequent_itemsets, metric="conviction", min_threshold=2)
rules
```

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
|---|---|---|---|---|---|---|---|---|---|
| 0 | (Eggs) | (Kidney Beans) | 0.8 | 1.0 | 0.8 | 1.0 | 1.00 | 0.00 | inf |
| 1 | (Onion) | (Eggs) | 0.6 | 0.8 | 0.6 | 1.0 | 1.25 | 0.12 | inf |
| 2 | (Milk) | (Kidney Beans) | 0.6 | 1.0 | 0.6 | 1.0 | 1.00 | 0.00 | inf |
| 3 | (Onion) | (Kidney Beans) | 0.6 | 1.0 | 0.6 | 1.0 | 1.00 | 0.00 | inf |
| 4 | (Yogurt) | (Kidney Beans) | 0.6 | 1.0 | 0.6 | 1.0 | 1.00 | 0.00 | inf |
| 5 | (Eggs, Onion) | (Kidney Beans) | 0.6 | 1.0 | 0.6 | 1.0 | 1.00 | 0.00 | inf |
| 6 | (Onion, Kidney Beans) | (Eggs) | 0.6 | 0.8 | 0.6 | 1.0 | 1.25 | 0.12 | inf |
| 7 | (Onion) | (Eggs, Kidney Beans) | 0.6 | 0.8 | 0.6 | 1.0 | 1.25 | 0.12 | inf |

```python
# at least 2 antecedents

rules["antecedent_len"] = rules["antecedents"].apply(lambda x: len(x))
rules
```

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | antecedent_len |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | (Eggs) | (Kidney Beans) | 0.8 | 1.0 | 0.8 | 1.0 | 1.00 | 0.00 | inf | 1 |
| 1 | (Onion) | (Eggs) | 0.6 | 0.8 | 0.6 | 1.0 | 1.25 | 0.12 | inf | 1 |
| 2 | (Milk) | (Kidney Beans) | 0.6 | 1.0 | 0.6 | 1.0 | 1.00 | 0.00 | inf | 1 |
| 3 | (Onion) | (Kidney Beans) | 0.6 | 1.0 | 0.6 | 1.0 | 1.00 | 0.00 | inf | 1 |
| 4 | (Yogurt) | (Kidney Beans) | 0.6 | 1.0 | 0.6 | 1.0 | 1.00 | 0.00 | inf | 1 |
| 5 | (Eggs, Onion) | (Kidney Beans) | 0.6 | 1.0 | 0.6 | 1.0 | 1.00 | 0.00 | inf | 2 |
| 6 | (Onion, Kidney Beans) | (Eggs) | 0.6 | 0.8 | 0.6 | 1.0 | 1.25 | 0.12 | inf | 2 |
| 7 | (Onion) | (Eggs, Kidney Beans) | 0.6 | 0.8 | 0.6 | 1.0 | 1.25 | 0.12 | inf | 1 |

```python
# a confidence > 0.75
# a lift score > 1.2
# a leverage > 0.1
# a conviction > 2.0

rules[ (rules['antecedent_len'] >= 2) &
       (rules['confidence'] > 0.75) &
       (rules['lift'] > 1.2) &
       (rules['leverage'] > 0.1) &
       (rules['conviction'] > 2.0) ]
```

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | antecedent_len |
|---|---|---|---|---|---|---|---|---|---|---|
| 6 | (Onion, Kidney Beans) | (Eggs) | 0.6 | 0.8 | 0.6 | 1.0 | 1.25 | 0.12 | inf | 2 |

# Measures of Correlation

- *"Buy walnuts $\Rightarrow$ buy milk* [1%, 80%]" is misleading if 85% of customers buy milk
- Support and confidence are not good to indicate correlations
- Over 20 interestingness measures have been proposed (see Tan, Kumar, Sritastava @KDD'02)
- Which are good ones?

| symbol | measure | range | formula |
|---|---|---|---|
| $\phi$ | $\phi$-coefficient | -1...1 | $\frac{P(A,B)-P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$ |
| $Q$ | Yule's Q | -1 ... 1 | $\frac{P(A,B)P(\overline{A},\overline{B})-P(A,\overline{B})P(\overline{A},B)}{P(A,B)P(\overline{A},\overline{B})+P(A,\overline{B})P(\overline{A},B)}$ |
| $Y$ | Yule's Y | -1 ... 1 | $\frac{\sqrt{P(A,B)P(\overline{A},\overline{B})}-\sqrt{P(A,\overline{B})P(\overline{A},B)}}{\sqrt{P(A,B)P(\overline{A},\overline{B})}+\sqrt{P(A,\overline{B})P(\overline{A},B)}}$ |
| $k$ | Cohen's | -1 ... 1 | $\frac{P(A,B)+P(\overline{A},\overline{B})-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A)P(B)-P(\overline{A})P(\overline{B})}$ |
| $PS$ | Piatetsky-Shapiro's | -0.25 ... 0.25 | $P(A,B)-P(A)P(B)$ |
| $F$ | Certainty factor | -1 ... 1 | $\max(\frac{P(B\|A)-P(B)}{1-P(B)}, \frac{P(A\|B)-P(A)}{1-P(A)})$ |
| $AV$ | added value | -0.5 ... 1 | $\max(P(B\|A)-P(B), P(A\|B)-P(A))$ |
| $K$ | Klosgen's Q | -0.33 ... 0.38 | $\sqrt{P(A,B)}\max(P(B\|A)-P(B), P(A\|B)-P(A))$ |
| $g$ | Goodman-kruskal's | 0 ... 1 | $\frac{\Sigma_j \max_k P(A_j,B_k)+\Sigma_k \max_j P(A_j,B_k)-\max_j P(A_j)-\max_k P(B_k)}{2-\max_j P(A_j)-max_k P(B_k)}$ |
| $M$ | Mutual Information | 0 ... 1 | $\frac{\Sigma_i \Sigma_j P(A_i,B_j)\log\frac{P(A_i,B_j)}{P(A_i)P(B_J)}}{\min(-\Sigma_i P(A_i)\log P(A_i)\log P(A_i), -\Sigma_i P(B_i)\log P(B_i)\log P(B_i))}$ |
| $J$ | J-Measure | 0 ... 1 | $\max(P(A,B)\log(\frac{P(B\|A)}{P(B)})+P(A\overline{B})\log(\frac{P(\overline{B}\|A)}{P(\overline{B})}))$ $P(A,B)\log(\frac{P(A\|B)}{P(A)})+P(\overline{A}B)\log(\frac{P(\overline{A}\|B)}{P(\overline{A})})$ |
| $G$ | Gini index | 0 ... 1 | $\max(P(A)[P(B\|A)^2+P(\overline{B}\|A)^2]+P(\overline{A}[P(B\|\overline{A})^2+P(\overline{B}\|\overline{A})^2]-P(B)^2-P(\overline{B})^2,$ $P(B)[P(A\|B)^2+P(\overline{A}\|B)^2]+P(\overline{B}[P(A\|\overline{B})^2+P(\overline{A}\|\overline{B})^2]-P(A)^2-P(\overline{A})^2)$ |
| $s$ | support | 0 ... 1 | $P(A,B)$ |
| $c$ | confidence | 0 ... 1 | $max(P(B\|A), P(A\|B))$ |
| $L$ | Laplace | 0 ... 1 | $\max(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2})$ |
| $IS$ | Cosine | 0 ... 1 | $\frac{P(A,B)}{\sqrt{P(A)P(B)}}$ |
| $\gamma$ | coherence(Jaccard) | 0 ... 1 | $\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$ |
| $\alpha$ | all_confidence | 0 ... 1 | $\frac{P(A,B)}{\max(P(A),P(B))}$ |
| $o$ | odds ratio | 0 ... $\infty$ | $\frac{P(A,B)P(\overline{A},\overline{B})}{P(\overline{A},B)P(A,\overline{B})}$ |
| $V$ | Conviction | 0.5 ... $\infty$ | $\max(\frac{P(A)P(\overline{B})}{P(A\overline{B})}, \frac{P(B)P(\overline{A})}{P(B\overline{A})})$ |
| $\lambda$ | lift | 0 ... $\infty$ | $\frac{P(A,B)}{P(A)P(B)}$ |
| $S$ | Collective strength | 0 ... $\infty$ | $\frac{P(A,B)+P(\overline{AB})}{P(A)P(B)+P(\overline{A})P(\overline{B})} \times \frac{1-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A,B)-P(\overline{AB})}$ |
| $\chi^2$ | $\chi^2$ | 0 ... $\infty$ | $\Sigma_i \frac{(P(A_i)-E_i)^2}{E_i}$ |