Solutions

Exercise 2: Apriori, Confidence, Itemsets and Association Rules

Exercise 2-1: Combinatoric explosion

(a) A database contains transactions over the following items: “apples”, “bananas”, and “cherries”. How many different combinations of these items can exist (i.e., how many different transactions could possibly occur in the database)?

(We do not distinguish whether a transaction contains a fruit once or several times, e.g., if someone bought one apple or several apples would just result in the transaction containing “apples”.)

Suggested solution:

A transaction can either contain apples or not. We have 2 possibilities here.

Each of these possibilities can either contain bananas or not. That is, for each of the 2 previous possibilities, we have 2 possibilities. Therefore we have four overall.

Each of these four possibilities can either contain cranberries or not. Eight possibilities.

(b) The database now also contains the items “dates”, “eggplants”, “figs”, and “guavas”. How many possible transactions do we have now?

Suggested solution:

It becomes clear that sketching a tree is not convenient anymore. Maybe some have already noted that we have powers of 2.

TID	A	B	C	D	E	F	G
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1
2	0	0	0	0	0	1	0
3	0	0	0	0	0	1	1
4	0	0	0	0	1	0	0
5	0	0	0	0	1	0	1
6	0	0	0	0	1	1	0
7	0	0	0	0	1	1	1
8	0	0	0	1	0	0	0
$\vdots$							
16	0	0	1	0	0	0	0
$\vdots$							
127	1	1	1	1	1	1	1

(c) How many combinations (possible different transactions) do we have with n items?

Suggested solution:

All possible combinations are the powerset over the set of items, where each item can be either in or out. This property (in or out) can be represented as a binary code, i.e., each element of the powerset can be uniquely mapped to exactly one number in binary representation, and each number x, $0 \leq x < 2^n$, can be uniquely mapped to exactly one element of the powerset.

So we have overall 2^n possible combinations (i.e., different transactions), where n is the number of items.

We say, the number of possibilities grows exponentially. And this growth rate is quite fast. For $n = 10$ we have 1024, for $n = 20$ we have 1,048,576, for $n = 30$ we have 1,073,741,824.

(d) How many transactions with exactly two items (i.e., 2-itemsets) can we have when the database contains 3 items? When it contains 5 items? How many k-itemsets do we have when the database contains n items?

Suggested solution:

Use the database with 3 items as example: we can have two of the three elements A, B, C: $\{A, B\}$, $\{A, C\}$, $\{B, C\}$.

To answer the question with complete enumeration of all possibilities for 5 elements already becomes tiresome, so we will derive the answer from the general solution.

This question is equivalent to the problem "drawing without replacement", i.e., one has a collection of n elements and draws k of them sequentially without putting an element back, once it has been drawn. Let us first assume, we care for the order of drawing, i.e., we distinguish $\{A, B\}$ from $\{B, A\}$. Then we have n possibilities to draw the first element, $n - 1$ possibilities to draw the second, and so on until we have $n - k + 1$ possibilities to draw the k-th element. Altogether:

$$
\begin{aligned}
n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot (n-k+1) &= \prod_{i=0}^{k-1} (n-i) && (1) \\
&= \frac{n!}{(n-k)!} && (2)
\end{aligned}
$$

Now we do actually not care for the order, i.e., we do *not* distinguish $\{A, B\}$ from $\{B, A\}$. Therefore we have to divide the result by the number of possible orderings. A set of k elements can be ordered in $k!$ different ways. Intuition: recursive explanation – each of the k elements can be placed as first, each of the remaining $k - 1$ elements as second etc., i.e., we have $k \cdot (k-1) \cdot (k-2) \cdot \ldots \cdot 1 = k!$ possibilities.

The number of k-itemsets out of n different items is therefore the expression from Eq. 2, divided by $k!$:

$$\frac{n!}{k!(n-k)!}$$

This is also written with the expression

$$\binom{n}{k}$$

and called the binomial coefficient.

$$\binom{5}{2} = \frac{5 \cdot 4 \cdot 3}{3 \cdot 2} = 10$$

Exercise 2-2: Itemsets and Association Rules

Given a set of transactions T according to the following table:

Set of transactions T

Transaction ID	items in basket
1	{Milk, Beer, Diapers}
2	{Bread, Butter, Milk}
3	{Milk, Diapers, Cookies }
4	{Bread, Butter, Cookies}
5	{Beer, Cookies, Diapers}
6	{Milk, Diapers, Bread, Butter}
7	{Bread, Butter, Diapers}
8	{Beer, Diapers}
9	{Milk, Diapers, Bread, Butter}
10	{Beer, Cookies}

(a) What are the support and the confidence of {Milk} $\Rightarrow$ {Diapers}?

Suggested solution:
Support is 4.
Confidence is $\frac{4}{5} = 80\%$.

(b) What are the support and the confidence of {Diapers} $\Rightarrow$ {Milk}?

Suggested solution:
Support is 4. (Surprise?)
Confidence is $\frac{4}{7} \approx 57\%$.

(c) What is the maximum number of size-3 itemsets that can be derived from this data set?

Suggested solution:
First we need to know the number of items. It helps to sort them alphabetically:
{Beer, Bread, Butter, Cookies, Diapers, Milk}
To choose any 3 of 6, the mathematical term is

$$\begin{aligned}\binom{6}{3} &= \frac{6!}{3! \cdot (6-3)!} \\ &= \frac{6 \cdot 5 \cdot 4}{3 \cdot 2} \\ &= \frac{2 \cdot 5 \cdot 2}{1} \\ &= 20\end{aligned}$$

(d) What is the maximum number of association rules that can be extracted from this dataset (including rules, that have zero support)?

Suggested solution:
From six items, we can generate association rules by having 1 or 2 or ...or 5 items in the antecedent and include all or some of the remaining items in the consequent (we exclude the case of an empty consequent, hence we subtract 1 from the number of elements in the powerset of the remaining items).
Mathematically:

$$\binom{6}{1} \cdot (2^5 - 1) + \binom{6}{2} \cdot (2^4 - 1) + \binom{6}{3} \cdot (2^3 - 1) + \binom{6}{4} \cdot (2^2 - 1) + \binom{6}{5} \cdot (2^1 - 1),$$

that is for d items:

$$\sum_{i=1}^{d-1} \binom{d}{i} \cdot (2^{d-i} - 1).$$

We can see that the number grows superexponentially.
The actual number is therefore:

$$\sum_{i=1}^{5} \binom{6}{i} \cdot (2^{6-i} - 1) = 602.$$

Note that the association rules have to consist of frequent itemsets in both, antecedent and consequent. Thus all possible candidates should be checked for their frequency in a database.
The number of candidates we have computed for 6 items in the database should be checked over all transactions.
With 10 transactions, you check over 10 transaction, with 100 transactions, you check over 100 transactions.
This is linear.
With 8 instead of 6 items, we have a lot more candidates. More than ten times as many things to check?

$$\sum_{i=1}^{7} \binom{8}{i} \cdot (2^{8-i} - 1) = 6050$$

(e) What is the maximum size of frequent itemsets that can be extracted (assuming $\sigma > 0$)?

Suggested solution:
The maximum frequent itemset occuring in the database has size 4. We can therefore not find any larger itemset with support > 0.

(f) Find an itemset (of size 2 or larger) that has the largest support.

Suggested solution:
$s(\{Bread, Butter\}) = 5$

(g) Find a pair of items, a and b, such that the rules $\{a\} \Rightarrow \{b\}$ and $\{b\} \Rightarrow \{a\}$ have the same confidence.

Suggested solution:

$$\begin{aligned}
\text{conf}(\{Bread\} \Rightarrow \{Butter\}) &= \frac{s(\{Bread, Butter\})}{s(\{Bread\})} \\
&= \frac{5}{5} \\
&= 1 \\
\text{conf}(\{Butter\} \Rightarrow \{Bread\}) &= \frac{s(\{Bread, Butter\})}{s(\{Butter\})} \\
&= \frac{5}{5} \\
&= 1
\end{aligned}$$

Exercise 2-3: Apriori candidate generation

Given the frequent 3-itemsets:

$$\{1,2,3\},\{1,2,4\},\{1,2,5\},\{1,3,5\},\{2,3,4\},\{2,3,5\},\{2,3,6\},\{2,5,6\},\{3,4,5\},\ \{3,5,6\}$$

List all candidate 4-itemsets following the Apriori joining and pruning procedure.

Suggested solution:

joining:

$$\begin{aligned}
\{1,2,3\}+\{1,2,4\} &\rightarrow \{1,2,3,4\}\\
\{1,2,3\}+\{1,2,5\} &\rightarrow \{1,2,3,5\}\\
\{1,2,4\}+\{1,2,5\} &\rightarrow \{1,2,4,5\}\\
\{2,3,4\}+\{2,3,5\} &\rightarrow \{2,3,4,5\}\\
\{2,3,4\}+\{2,3,6\} &\rightarrow \{2,3,4,6\}\\
\{2,3,5\}+\{2,3,6\} &\rightarrow \{2,3,5,6\}\\
\{1,3,5\} &\text{no joining partner}\\
\{2,5,6\} &\text{no joining partner}\\
\{3,4,5\} &\text{no joining partner}\\
\{3,5,6\} &\text{no joining partner}
\end{aligned}$$

pruning:

$$\begin{aligned}
\{1,2,3,4\} &\text{cannot be frequent as } \{1,3,4\} \text{ is not frequent}\\
\{1,2,4,5\} &\text{cannot be frequent as } \{1,4,5\} \text{ is not frequent}\\
&\text{alternative: } \{2,4,5\} \text{ is not frequent}\\
\{2,3,4,5\} &\text{cannot be frequent as } \{2,4,5\} \text{ is not frequent}\\
\{2,3,4,6\} &\text{cannot be frequent as } \{3,4,6\},\ \text{is not frequent}\\
&\text{alternative: } \{2,4,6\} \text{ is not frequent}
\end{aligned}$$

$$C_4 = \{\{1,2,3,5\},\{2,3,5,6\}\}$$

Exercise 2-4: The monotonicity of confidence

Theorem 2.1 in the Lecture states:

Given:

— itemset X

— $Y \subset X, Y \neq \emptyset$

If $\text{conf}(Y \Rightarrow (X \setminus Y)) < c$, then $\forall Y' \subset Y$:

$$\text{conf}(Y' \Rightarrow (X \setminus Y')) < c.$$

(a) Prove the theorem.

> **Suggested solution:**
> Consider the following two rules:
> $$Y' \Rightarrow X \setminus Y'$$
> and
> $$Y \Rightarrow X \setminus Y$$
> where $Y' \subset Y$.
> The confidence of the rules are: $\frac{s(X)}{s(Y')}$ and $\frac{s(X)}{s(Y)}$, respectively.
> Since $Y' \subset Y$, we have: $s(Y') \geq s(Y)$.
> Therefore the former rule cannot have higher confidence than the latter rule.

(b) Sketch an algorithm (pseudo code) that generates all association rules with support σ or above and a minimum confidence of c, provided the set F of all frequent itemsets (w.r.t. σ) with their support, efficiently using the pruning power of the given theorem.

> **Suggested solution:**
>
> AssociationRules(F, c):
> - foreach $Z \in F, |Z| \geq 2$ do:
>   - $A \leftarrow \{X | X \subset Z, X \neq \emptyset\}$
>   - while $A \neq \emptyset$ do:
>   - $X \leftarrow$ maximal element in A
>   - $A \leftarrow A \setminus \{X\}$
>   - $c_{\text{tmp}} \leftarrow s(Z)/s(X)$
>   - if $c_{\text{tmp}} \geq c$ then
>   - print $X \Rightarrow (Z \setminus X), s(Z), c_{\text{tmp}}$
>   - else
>   - $A \leftarrow A \setminus \{W | W \subset X\}$
>   - end if
>   - end while
> - end foreach

Exercise 2-5: Tools

(a) Install python packages: `scikit-learn`, `numpy`, `matplotlib`, `metrics`, and `linear-model` from `scikit-learn`, then load `diabetes` dataset from `sklearn`.

Suggested solution:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler

# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)

# Use only one feature
diabetes_X = diabetes_X[:, 2].reshape(-1, 1)
```

(b) Reserve a randomly chosen 80% of the data for training and the remaining for test using `sklearn.model_selection.train_test_split`, then assign data as x and target as y and investigate the shapes of the data.

Suggested solution:

```
# split data to train and test
diabetes_X_train,diabetes_X_test,diabetes_y_train,
  diabetes_y_test= train_test_split(diabetes_X,diabetes_y,
  test_size=0.2, random_state=0)

#shape of data
print(diabetes_X_train.shape, diabetes_X_test.shape )
print(diabetes_y_train. shape , diabetes_y_test. shape )
```

(c) Normalize data using `StandardScaler` from `sklearn.preprocessing`.

Suggested solution:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
diabetes_X_train = scaler.fit_transform(diabetes_X_train)
diabetes_X_test = scaler.transform(diabetes_X_test)
```

(d) Fit a linear regression model to the training set and make prediction.

Suggested solution:

```
# Train the model using the training sets
regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)
```

(e) Evaluate mean squared error (MSE) of the fitted model on the test set.

Suggested solution:

```
print(f"MSE:{mean_squared_error(diabetes_y_test,diabetes_y_pred):
.3f}")
```

(f) Plot the fitted model as a line and print its intercept and slope.

Suggested solution:

```
# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test, color="black")
plt.plot(diabetes_X_test,diabetes_y_pred,color="blue", linewidth=3)
plt.show()

# Slope and intercept
print(f"intercept:{regr.intercept_}")
print(f"slope:{regr.coef_}")
```

(g) Comment on the outcome. Could the model fit to data accurately enough?