2022년 가을학기

Association Mining

Lecture
13

Association Mining

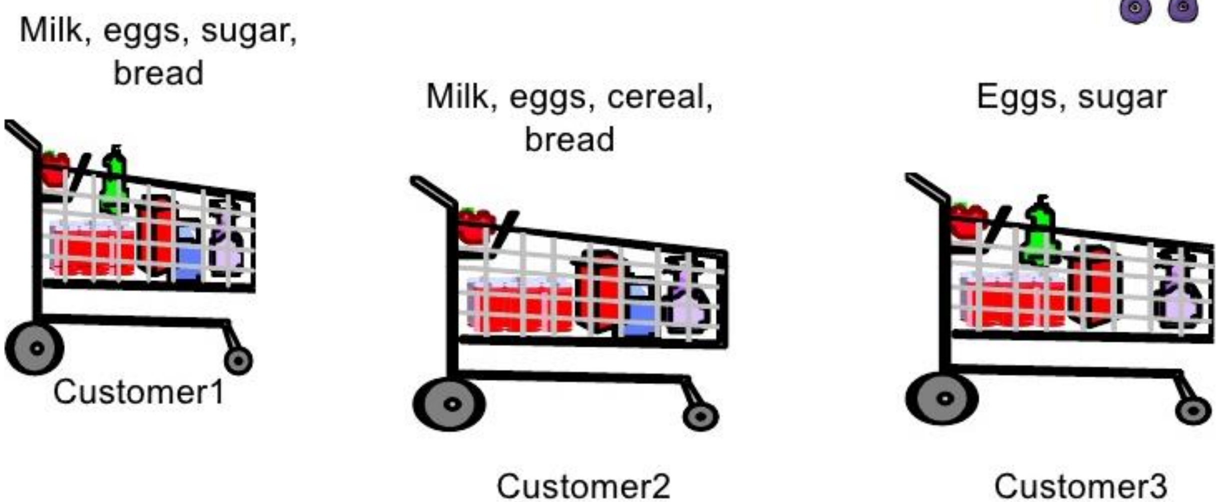
CONTENTS

What is Apriori Algorithm?

What is Apriori Algorithm?

❖ Ideas come from the market basket analysis

- Shopping example



- Questions
  - Given a list of products, can we predict what a customer will buy next?

- Goal
  - Find associations between different items that customers place in their basket

What is Apriori Algorithm?

Association rules

- Itemset
  - Groups of one or more items that appears in the data with some regularity
    - {bread, peanut butter, jelly}

- Association rules
  - If/then statements that help uncover associations between unrelated data in a dataset
    - {peanut butter, jelly} -> {bread}

  - Specify patterns found in the relationships/associations among items in the itemsets

What is Apriori Algorithm?

❖ Applications

- Searching for interesting and frequently occurring patterns of DNA and protein sequences in cancer data.
- Finding patterns of purchases or medical claims that occur in combination with fraudulent credit card or insurance use.
- Identifying combinations of behavior that precede customers dropping their cellular phone service or upgrading their cable television package.

What is Apriori Algorithm?

❖ Measuring rule interest – support and confidence

- Association rules are determined by two statistical measures
- Support
  - Measures how frequently the itemset occurs in the data

$$Support(X) = count(X)/N$$

- Confidence
  - Measurement of its predictive power or accuracy

$$Confidence(X \rightarrow Y) = support(X, Y)/support(X)$$

What is Apriori Algorithm?

❖ **Support**

- This says how popular item is in the dataset

$$\text{Support}\{🍎\} = \frac{4}{8}$$

Transaction 1	🍎 🍺 🍚 🍗
Transaction 2	🍎 🍺 🍚
Transaction 3	🍎 🍺
Transaction 4	🍎 🍐
Transaction 5	🍼 🍺 🍚 🍗
Transaction 6	🍼 🍺 🍚
Transaction 7	🍼 🍺
Transaction 8	🍼 🍐

What is Apriori Algorithm?

- **Confidence**
  - This says how likely the item Y is purchased when item X is purchased

$$\text{Confidence}\{🍎 \rightarrow 🍺\} = \frac{\text{Support}\{🍎, 🍺\}}{\text{Support}\{🍎\}}$$

$$= 3/8 * 8/4 = 75\%$$

B

How Apriori Works

How Apriori Works

❖ Example

- Given sample transactional data

Transaction ID	Onion	Potato	Burger	Milk	Beer
T1	1	1	1	0	0
T2	0	1	1	1	0
T3	0	0	0	1	1
T4	1	1	0	1	0
T5	1	1	1	0	1
T6	1	1	1	1	0

Transactional data

Assume:
minsup = 40%
minconf = 70%

How Apriori Works

Example

- Step 1: Create a frequency table of all the items that occur in all transactions

Item	Frequency (No. of transactions)
Onion	4
Potato	5
Burger	4
Milk	4
Beer	2

Support(Onion) = 4/6 = 66.6%
Support(Potato) = 5/6 = 83.3%
Support(Burger) = 4/6 = 66.6%
Support(Milk) = 4/6 = 66.6%
Support(Beer) = 2/6 = 33.3%

How Apriori Works

❖ Example

- Step 2: Determine elements for which the support is greater than or equal to the threshold support

1-Frequent Itemset

Item	Frequency (No. of transactions)
Onion	4
Potato	5
Burger	4
Milk	4
~~Beer~~	~~2~~

How Apriori Works

❖ Example

- Step 3: The next step is to make all the possible pairs of the significant items keeping in mind that the order doesn’t matter

Itemset	Frequency (No. of transactions)
Onion, Potato	4
Onion, Burger	3
Onion, Milk	2
Potato, Burger	4
Potato, Milk	3
Burger, Milk	2

How Apriori Works

❖ Example

- Step 4: Remove the itemsets that has less support than minimum support

Itemset	Frequency (No. of transactions)
Onion, Potato	4
Onion, Burger	3
~~Onion, Milk~~	~~2~~
Potato, Burger	4
Potato, Milk	3
~~Burger, Milk~~	~~2~~

Support(Onion, Potato) = 4/6 = 66.6%
Support(Onion, Burger) = 3/6 = 50%
Support(Onion, Milk) = 2/6 = 33.3%
Support(Potato, Burger) = 4/6 = 66.6%
Support(Potato, Milk) = 3/6 = 50%
Support(Burger, Milk) = 2/6 = 33.3%

How Apriori Works

❖ Example

- Step 4: Remove the itemsets that has less support than minimum support

2-Frequent Itemset

Itemset	Frequency (No. of transactions)
Onion, Potato	4
Onion, Burger	3
Potato, Burger	4
Potato, Milk	3

How Apriori Works

❖ Example

- Step 5: Now let's say we would like to look for a set of three items that are purchased together. We will use the itemsets found in step 4 and create a set of 3 items

3-Frequent Itemset

Itemset	Frequency (No. of transactions)
Onion, Potato, Burger	3
~~Potato, Burger, Milk~~	~~2~~

Support(Onion, Potato, Burger) = 3/6 = 50%
Support(Potato, Burger, Milk) = 2/6 = 33%

How Apriori Works

❖ Example

- Step 6: We can’t make any more frequent itemsets, so we stop here

1-Frequent Items

Item	Frequency (No. of transactions)
Onion	4
Potato	5
Burger	4
Milk	4

2-Frequent Items

Itemset	Frequency (No. of transactions)
Onion, Potato	4
Onion, Burger	3
Potato, Burger	4
Potato, Milk	3

3-Frequent Items

Itemset	Frequency (No. of transactions)
Onion, Potato, Burger	3

How Apriori Works

Example

- Step 7: Now we have our frequent itemset. From here we should create the association rules and calculate the *confidence* of each rules.

- In this example let's use the {Onion, Potato, Burger} as an example

Itemset	Frequency (No. of transactions)
Onion, Potato, Burger	3

Association rules	
Onion, Potato => Burger	Onion => Potato, Burger
Onion, Burger => Potato	Burger => Onion, Potato
Burger, Potato => Onion	Potato => Onion, Burger

How Apriori Works

❖ Example

- Step 8: Calculate confidence score for all rules

Association rules
Onion, Potato => Burger
Onion, Burger => Potato
Burger, Potato => Onion
Onion => Potato, Burger
Burger => Onion, Potato
Potato => Onion, Burger

Confidence(Onion, Potato => Burger) = 3/4 = 75%
Confidence(Onion, Burger => Potato) = 3/3 = 100%
Confidence(Burger, Potato => Onion) = 3/4 = 75%
Confidence(Onion => Potato,Burger) = 3/4 = 75%
Confidence(Burger => Onion, Potato) = 3/4 = 75%
Confidence(Potato => Onion, Burger) = 3/5 = 60%

How Apriori Works

❖ Example

- The final association rules from the {Onion, Potato, Burger} will be shown in following table.

Association rules
Onion, Potato => Burger
Onion, Burger => Potato
Burger, Potato => Onion
Onion => Potato, Burger
Burger => Onion, Potato
~~Potato => Onion, Burger~~

How Apriori Works

❖ Apriori algorithm in Python

- pip install mlxtend
- pip install pandas

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

dataset = [['Onion', 'Potato', 'Burger'],
            ['Potato', 'Burger', 'Milk'],
            ['Milk', 'Beer'],
            ['Potato', 'Milk'],
            ['Onion', 'Potato', 'Burger', 'Beer'],
            ['Onion', 'Potato', 'Burger', 'Milk']
          ]
```

How Apriori Works

❖ Apriori algorithm in Python

- Transaction encoding

```
encode = TransactionEncoder()

encoded_array = encode.fit(dataset).transform(dataset)
encoded_array
```

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

How Apriori Works

❖ Apriori algorithm in Python

- Transforming into data frame

```
dataframe = pd.DataFrame(encoded_array, columns=encode.columns_)
dataframe
```

	Beer	Burger	Milk	Onion	Potato
0	False	True	False	True	True
1	False	True	True	False	True
2	True	False	True	False	False
3	False	False	True	False	True
4	True	True	False	True	True
5	False	True	True	True	True

How Apriori Works

❖ Apriori algorithm in Python

- Train with Apriori algorithm (min_support = 0.4)

```
frequent_itemsets = apriori(dataframe, min_support=0.4, use_colnames=True)
frequent_itemsets
```

	support	itemsets
0	0.666667	(Burger)
1	0.666667	(Milk)
2	0.500000	(Onion)
3	0.833333	(Potato)
4	0.500000	(Burger, Onion)
5	0.666667	(Potato, Burger)
6	0.500000	(Milk, Potato)
7	0.500000	(Potato, Onion)
8	0.500000	(Potato, Burger, Onion)

How Apriori Works

❖ Apriori algorithm in Python

- Creating association rules

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

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Burger)	(Onion)	0.666667	0.500000	0.500000	0.75	1.5	0.166667	2.000000
1	(Onion)	(Burger)	0.500000	0.666667	0.500000	1.00	1.5	0.166667	inf
2	(Potato)	(Burger)	0.833333	0.666667	0.666667	0.80	1.2	0.111111	1.666667
3	(Burger)	(Potato)	0.666667	0.833333	0.666667	1.00	1.2	0.111111	inf
4	(Milk)	(Potato)	0.666667	0.833333	0.500000	0.75	0.9	-0.055556	0.666667
5	(Onion)	(Potato)	0.500000	0.833333	0.500000	1.00	1.2	0.083333	inf
6	(Potato, Burger)	(Onion)	0.666667	0.500000	0.500000	0.75	1.5	0.166667	2.000000
7	(Potato, Onion)	(Burger)	0.500000	0.666667	0.500000	1.00	1.5	0.166667	inf
8	(Burger, Onion)	(Potato)	0.500000	0.833333	0.500000	1.00	1.2	0.083333	inf
9	(Burger)	(Potato, Onion)	0.666667	0.500000	0.500000	0.75	1.5	0.166667	2.000000
10	(Onion)	(Potato, Burger)	0.500000	0.666667	0.500000	1.00	1.5	0.166667	inf

C

Text Mining Practice

Text Mining Practice

❖ Dataset

- Market_Basket_Optimisation.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	shrimp	almonds	avocado	vegetable	green gra	whole we	yams	cottage ch	energy dri	tomato ju	low fat yo	green tea	honey	salad	mineral w	salmon	antioxyda	frozen sm	spinach	olive oil
2	burgers	meatballs	eggs																	
3	chutney																			
4	turkey	avocado																		
5	mineral w	milk	energy ba	whole wh	green tea															
6	low fat yogurt																			
7	whole wh	french fries																		
8	soup	light crear	shallot																	
9	frozen veg	spaghetti	green tea																	
10	french fries																			
11	eggs	pet food																		
12	cookies																			
13	turkey	burgers	mineral w	eggs	cooking oil															
14	spaghetti	champagn	cookies																	
15	mineral w	salmon																		
16	mineral water																			
17	shrimp	chocolate	chicken	honey	oil	cooking oi	low fat yogurt													
18	turkey	eggs																		
19	turkey	fresh tuna	tomatoes	spaghetti	mineral w	black tea	salmon	eggs	chicken	extra dark chocolate										
20	meatballs	milk	honey	french frie	protein bar															
21	red wine	shrimp	pasta	pepper	eggs	chocolate	shampoo													
22	rice	sparkling water																		
23	spaghetti	mineral w	ham	body spra	pancakes	green tea														
24	burgers	grated che	shrimp	pasta	avocado	honey	white win	toothpaste												
25	eggs																			
26	parmesan	spaghetti	soup	avocado	milk	fresh bread														
27	ground be	spaghetti	mineral w	milk	energy ba	black tea	salmon	frozen sm	escalope											
28	sparkling water																			
29	mineral w	eggs	chicken	chocolate	french fries															
30	frozen veg	spaghetti	yams	mineral water																
31	herb & pe	tomato sa	light crear	magazines																
32	mineral w	chocolate	avocado	eggs																
33	turkey	french frie	strawberries																	
34	frozen veg	strong che	chocolate																	

Text Mining Practice

❖ Loading dataset

```
import pandas as pd
import seaborn as sns
import numpy as np
from pandas import DataFrame
import matplotlib.pyplot as plt
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

basket = pd.read_csv("D:₩Market_Basket_Optimisation.csv", header = None)
basket.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antioxydant juice	froz smootl
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N:
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N:
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N:
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N:

Text Mining Practice

❖ Converting the data frame into a list of lists

```
records = []
for i in range (0, 7501):
    records.append([str(basket.values[i,j]) for j in range(0, 20)])
```

❖ Encoding and transforming back to data frame

```
encode = TransactionEncoder()
encoded_array = encode.fit(records).transform(records)

data_frame = pd.DataFrame(encoded_array, columns = encode.columns_)
data_frame
```

	asparagus	almonds	antioxydant juice	asparagus	avocado	babies food	bacon	barbecue sauce	black tea	blueberries	...	turkey	vegetables mix	water spray	white wine	whole weat flour
0	False	True	True	False	True	False	False	False	False	False	...	False	True	False	False	True
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False
3	False	False	False	False	True	False	False	False	False	False	...	True	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False
5	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False

Text Mining Practice

❖ Drop missing values

```
basket_clean = data_frame.drop(['nan'], axis = 1)
basket_clean
```

❖ Train with Apriori

```
frequent_itemsets = apriori(basket_clean, min_support = 0.04, use_colnames = True)
frequent_itemsets.head()
```

	support	itemsets
0	0.087188	(burgers)
1	0.081056	(cake)
2	0.046794	(champagne)
3	0.059992	(chicken)
4	0.163845	(chocolate)

Text Mining Practice

❖ Creating rules

```
pattern_rules = association_rules(frequent_itemsets, metric = 'lift', min_threshold = 1)
pattern_rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(chocolate)	(mineral water)	0.163845	0.238368	0.052660	0.321400	1.348332	0.013604	1.122357
1	(mineral water)	(chocolate)	0.238368	0.163845	0.052660	0.220917	1.348332	0.013604	1.073256
2	(eggs)	(mineral water)	0.179709	0.238368	0.050927	0.283383	1.188845	0.008090	1.062815
3	(mineral water)	(eggs)	0.238368	0.179709	0.050927	0.213647	1.188845	0.008090	1.043158
4	(ground beef)	(mineral water)	0.098254	0.238368	0.040928	0.416554	1.747522	0.017507	1.305401
5	(mineral water)	(ground beef)	0.238368	0.098254	0.040928	0.171700	1.747522	0.017507	1.088672
6	(milk)	(mineral water)	0.129583	0.238368	0.047994	0.370370	1.553774	0.017105	1.209650
7	(mineral water)	(milk)	0.238368	0.129583	0.047994	0.201342	1.553774	0.017105	1.089850
8	(spaghetti)	(mineral water)	0.174110	0.238368	0.059725	0.343032	1.439085	0.018223	1.159314
9	(mineral water)	(spaghetti)	0.238368	0.174110	0.059725	0.250559	1.439085	0.018223	1.102008

Quiz for Lecture 13

- Submit your source code for the following task:
  1. Try all source code in the lecture

- Submission: source code, result screenshots and result explanation

감사합니다!