Association Rule Learning and its exercises

張家瑋 博士
副教授
國立臺中科技大學資訊工程系

關聯規則學習

Association Rule Learning

概念

- 在大型資料庫中發現項目間關聯的方法。
  - {牛奶, 麵包}→{可樂}：代表某人同時買了牛奶和麵包，就可能會買可樂。
- 該方法常使用於電子商務上，通常可為**促銷**、**產品推薦**等行銷活動的決策依據。

定義

- 商品的項目集合(itemset)，$I = \{ I_1, I_2, \ldots, I_m \}$ 。#Item
- 交易資料庫(Database)，$D = \{t_1, t_2, \ldots, t_n\}$ 。 #Transaction
- 關聯規則(Association Rule)， $X \rightarrow Y$

案例

TID	網球拍	網 球	運動鞋	羽毛球
1	1	1	1	0
2	1	1	0	0
3	1	0	0	0
4	1	0	1	0
5	0	1	1	1
6	1	1	0	0

- 顧客購買記錄的資料庫 *D*，包含 6 個 Transactions
- 項目集 *I* = {網球拍, 網球, 運動鞋, 羽毛球}

觀察關聯規則，網球拍 → 網球。

1. Transaction 1, 2, 3, 4, 6 包含網球拍。
2. Transaction 1, 2, 6 同時包含網球拍和網球。
3. 支持度 = 3/6 = 0.5，信心度 = 3/5 = 0.6。

- 若最小支持度為 0.5，最小信心度為 0.6。
- 關聯規則 "網球拍→網球" 是存在強關聯的。

- 1-itemset (4): {網球拍}, {網球}, {運動鞋}, {羽毛球}
- 2-itemset (6): {網球拍, 網球}, {網球拍, 運動鞋}, {網球拍, 羽毛球},
  {網球, 運動鞋}, {網球, 羽毛球}, {運動鞋, 羽毛球}
- 3-itemset (4): {網球拍, 網球, 運動鞋}, {網球拍, 網球, 羽毛球}, {網球拍, 運動鞋, 羽毛球}
  {網球, 運動鞋, 羽毛球}

Apriori

概念

- 逐層搜索的迭代方法。
- k-itemset 用於探索（k + 1） - itemset。
  1. 找出 frequent 1-itemset，L_1。L_1 用來找 frequent 2-itemset，L_2。而 L_2 用來找到 L_3。直到不能找到 k-itemset。
  2. 每找一個 L_k 需要掃描一次資料庫。為提高頻繁項集逐層產生的效率，Apriori 性質則可減少搜索。
- Apriori 性質：frequent itemset 的所有非空子集都必須是頻繁的。
  - 若某個 k-itemset 的 candidate 的 subsets 不在 (k-1)-itemset 時，這個 candidate 就可以直接刪除。

案例

TID	網球拍	網 球	運動鞋	羽毛球
1	1	1	1	0
2	1	1	0	0
3	1	0	0	0
4	1	0	1	0
5	0	1	1	1
6	1	1	0	0

- 顧客購買記錄的資料庫 *D*，包含 6 個 Transactions
- 項目集 *I* = {網球拍, 網球, 運動鞋, 羽毛球}

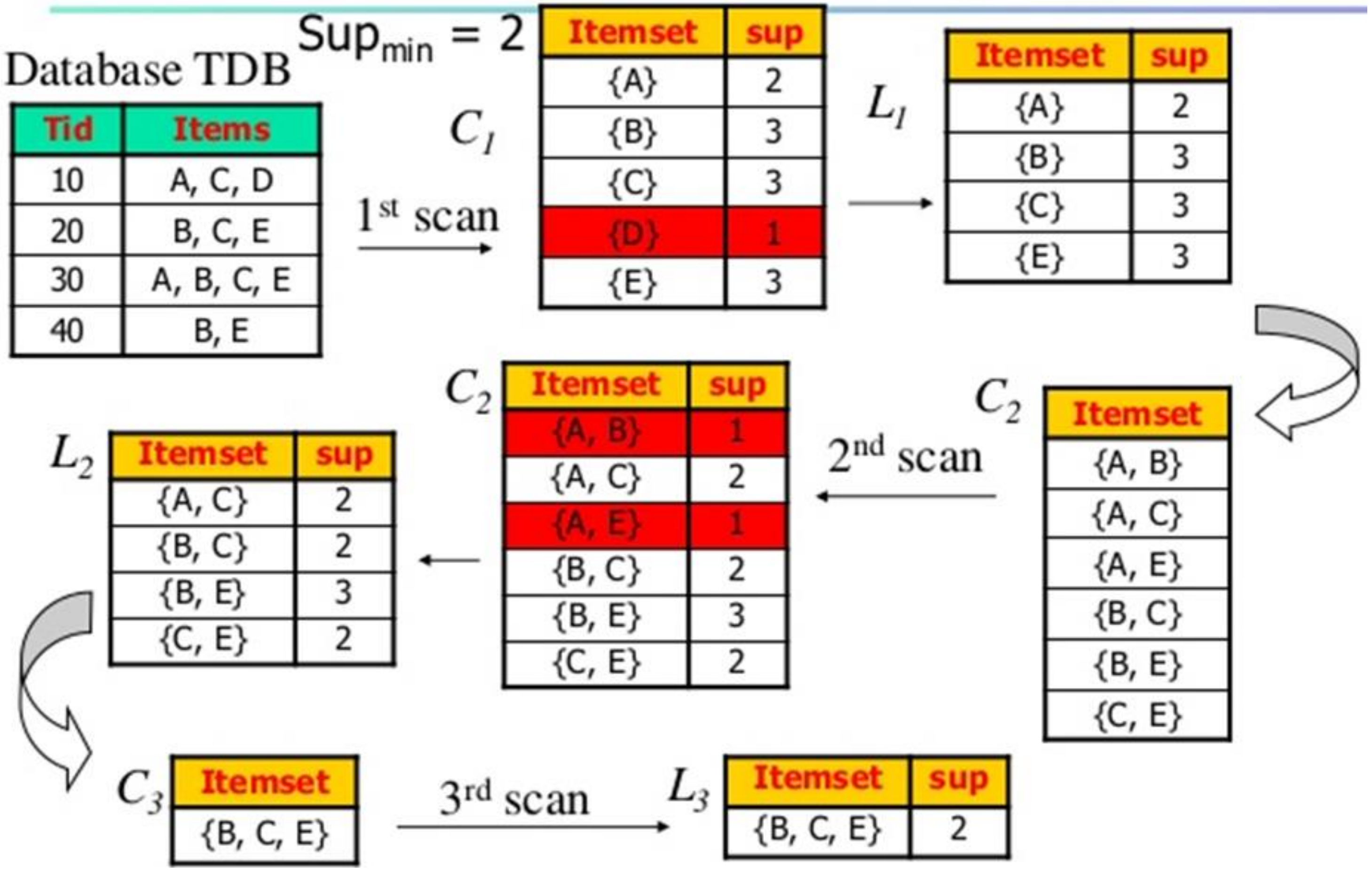
觀察關聯規則，網球拍 → 網球。

1. Transaction 1, 2, 3, 4, 6 包含網球拍。
2. Transaction 1, 2, 6 同時包含網球拍和網球。
3. 支持度 = 3/6 = 0.5，信心度 = 3/5 = 0.6。

- 若最小支持度為 0.5，最小信心度為 0.6。
- 關聯規則 "網球拍→網球" 是存在強關聯的。

- 1-itemset (4): {網球拍}, {網球}, {運動鞋}, ~~{羽毛球}~~
- 2-itemset (7): {網球拍, 網球}, {網球拍, 運動鞋}, ~~{網球拍, 羽毛球}~~,
  {網球, 運動鞋}, ~~{網球, 羽毛球}~~, ~~{運動鞋, 羽毛球}~~
- 3-itemset (4): {網球拍, 網球, 運動鞋}, ~~{網球拍, 網球, 羽毛球}~~, ~~{網球拍, 運動鞋, 羽毛球}~~
  ~~{網球, 運動鞋, 羽毛球}~~

方法



$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1^{st} scan

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2^{nd} scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3^{rd} scan

L_3

Itemset	sup
{B, C, E}	2

方法

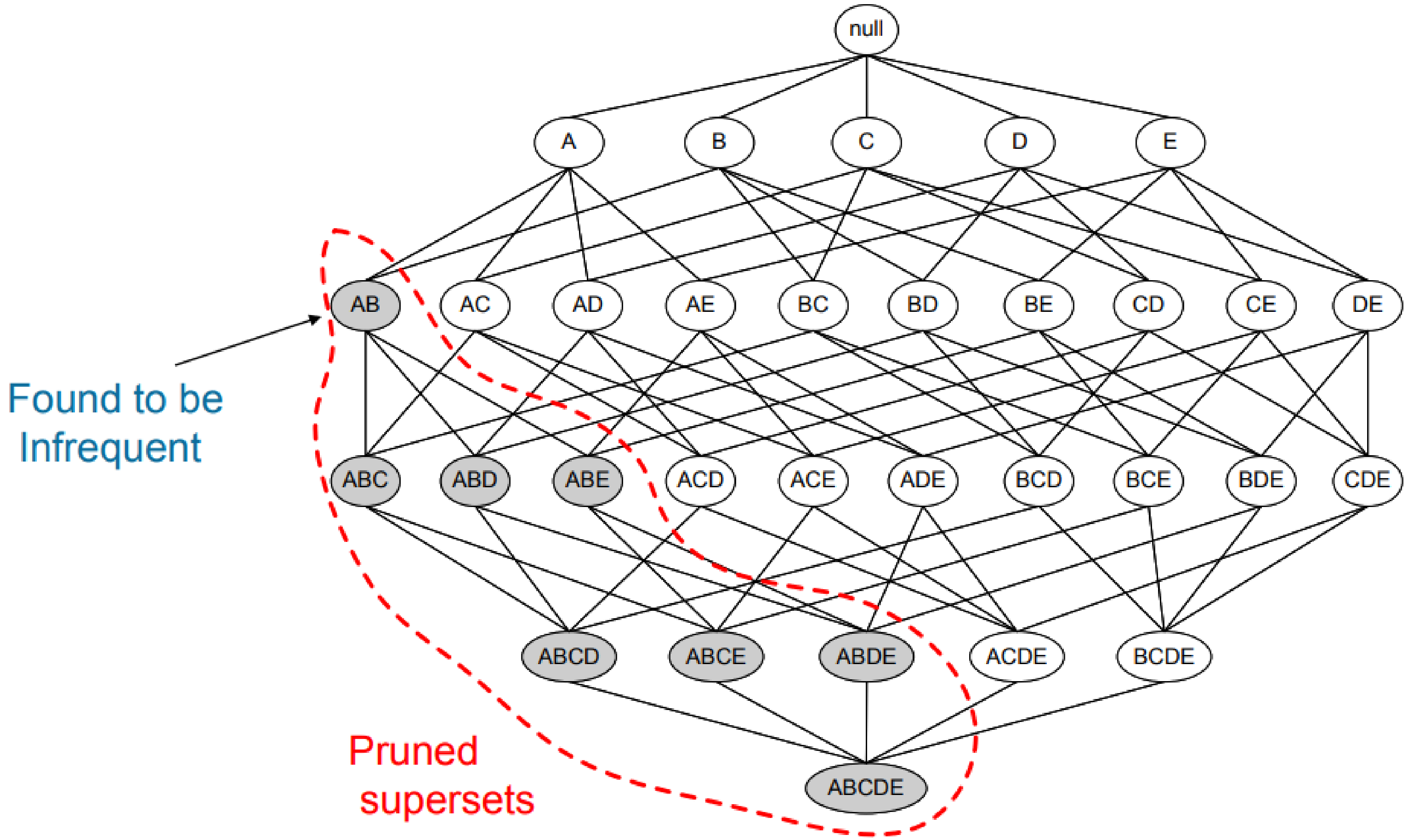
1. $C_3 = L_2$ 的組合
   - L_2 = {{A, C}, {B, C}, {B, E}, {C, E}}

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

     = {{A, B, C}, {A, C, E}, {B, C, E}}
2. 使用 Apriori 性質剪枝：某個 frequent itemset 的所有 subsets 必須是頻繁的，對 candidate itemset C_3，我們可以刪除其非頻繁的 subsets ：
   - {A, B, C} 的 2-itemset 是 {A, B}, {A, C}, {B, C}，其中 {A, B} 不是 L_2的元素，所以刪除;
   - {A, C, E} 的 2-itemset 是 {A, C}, {A, E}, {C, E}，其中 {A, E} 不是 L_2的元素，所以刪除;
   - {B, C, E} 的 2-itemset 是 {B, C}, {B, E}, {C, E}，所有 2-itemset 都是 L_2的元素，因此保留。
3. 剪枝後得到 C_3 ={{B, C, E}}

剪枝

Illustrating Apriori Principle



Thinking Time

重點

1. 在每一步產生 candidate itemset 時產生的組合過多，沒有排除不應該參與組合的元素。
2. 每次計算 itemset 的支持度時都對全部的 transactions 掃描一遍，造成龐大的I / O開銷。這種代價是隨著資料的增加而產生幾何級數的增長。

FP-Growth

概念

- 不用產生 candidate itemsets。
- 以樹(Tree)的結構儲存 frequent itemsets，即 frequent pattern tree (FP-tree)。
- 只要遞迴地探勘這棵樹。

FP-tree 建造方法

TID	Items bought
100	{a, c, d, f, g, i, m, p}
200	{a, b, c, f, i, m, o}
300	{b, f, h, j, o}
400	{b, c, k, s, p}
500	{a, c, e, f, l, m, n, p}

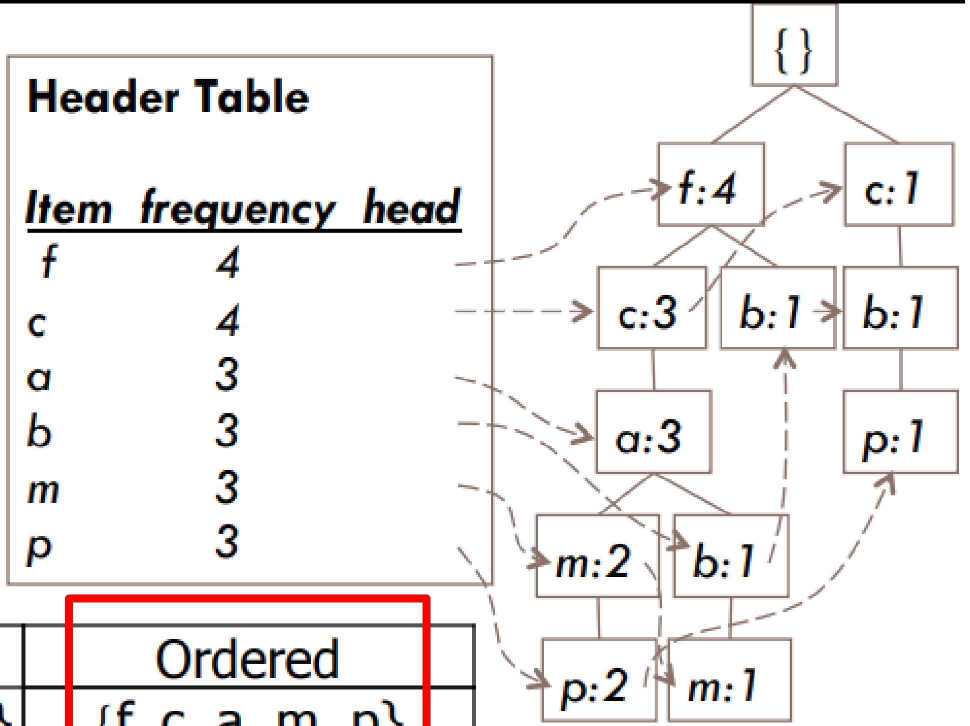
min_support = 3

a	3
b	3
c	4
f	4
m	3
p	3

f	4
c	4
a	3
b	3
m	3
p	3

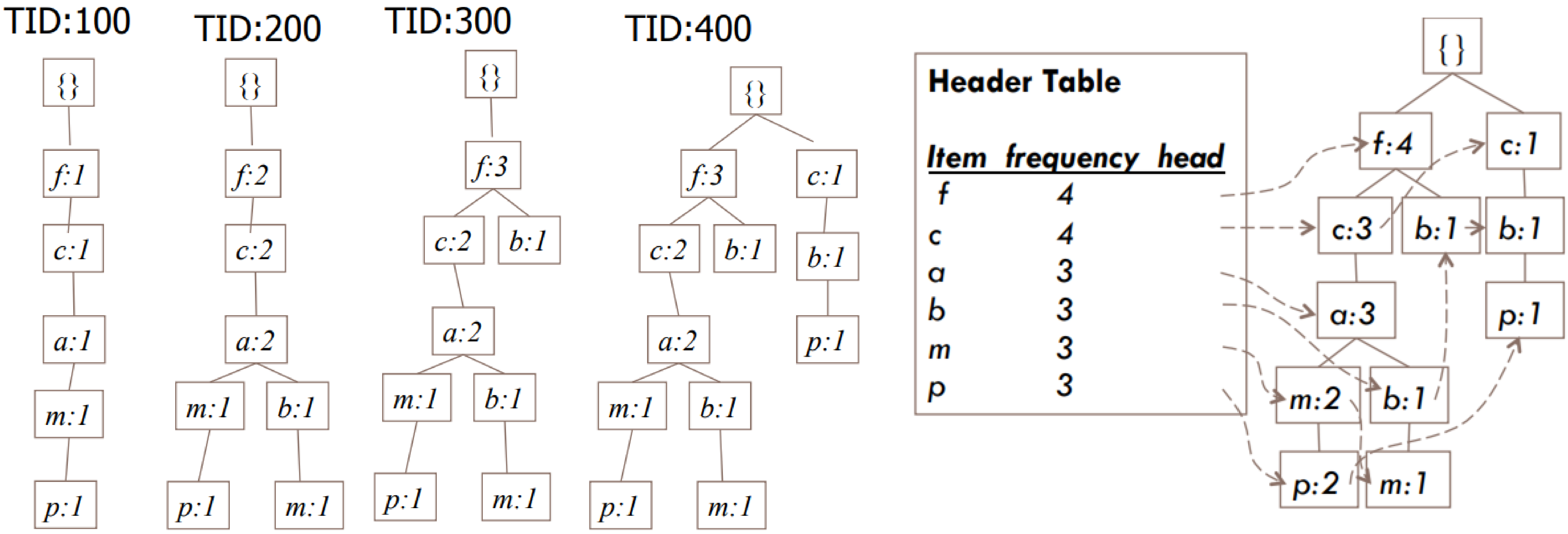
FP-tree 建造方法

f	4
c	4
a	3
b	3
m	3
p	3



TID	Items bought	Ordered
100	{a, c, d, f, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, i, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, c, e, f, l, m, n, p}	{f, c, a, m, p}

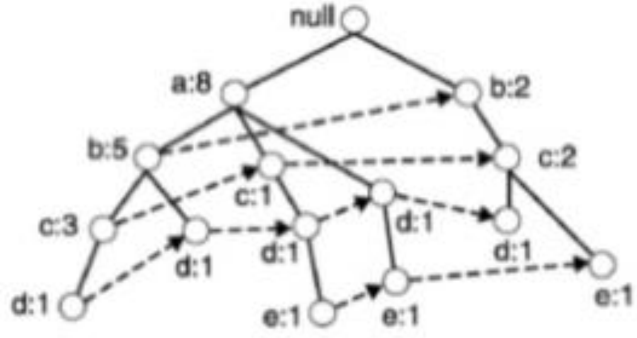
FP-tree 建造方法


TID:100
{}
f:1
c:1
a:1
m:1
p:1
TID:200
{}
f:2
c:2
a:2
m:1
b:1
p:1
m:1
TID:300
{}
f:3
c:2
b:1
a:2
m:1
b:1
p:1
m:1
TID:400
{}
f:3
c:1
c:2
b:1
b:1
a:2
p:1
m:1
b:1
p:1
m:1
Header Table
Item frequency head
f 4
c 4
a 3
b 3
m 3
p 3
{}
f:4
c:1
c:3
b:1
b:1
a:3
p:1
m:2
b:1
p:2
m:1

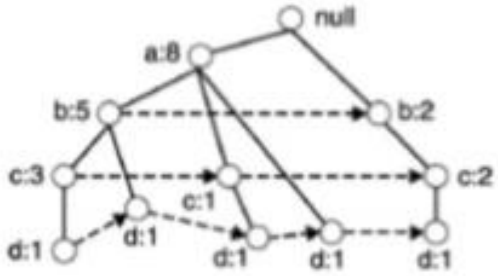
Mining Tree

- Bottom-Up 探索，依序檢視每個項目。
- 遞迴建子樹，找到所有 k-itemsets。

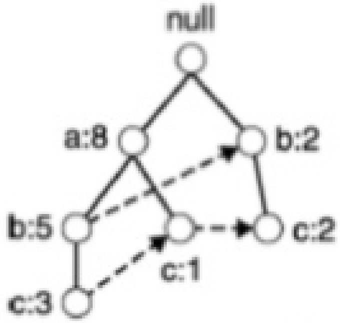
Mining Tree



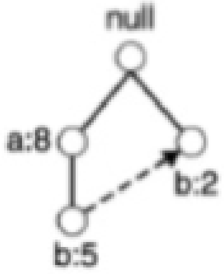
FP-tree



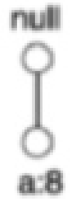
包含d節點
的子樹



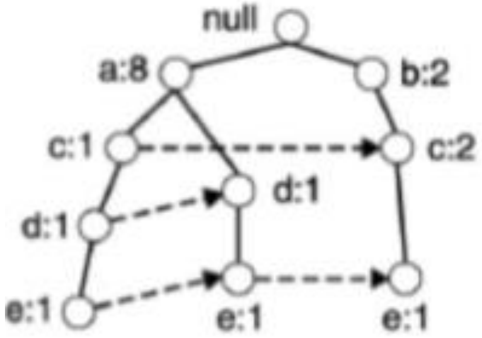
包含c節點
的子樹



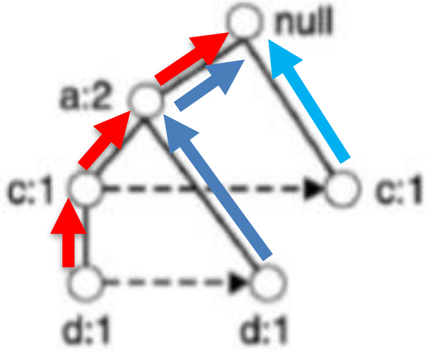
包含b節點
的子樹



包含a節點
的子樹



包含e節點
的子樹



{e}的條件FP-tree
<u>If *Sup{b} < 2*</u>

AI

Thinking Time

重點

- 避免多次掃描資料庫(for support)，節省了IO與運算成本。
- 不產生 candidate itemset。

Reference

- https://zh.wikipedia.org/wiki/%E5%85%B3%E8%81%94%E8%A7%84%E5%88%99%E5%AD%A6%E4%B9%A0
- https://www.slideshare.net/waynechung944/fp-growth-intro

Apriori

Case 1

Data Description

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

Online Retail Data Set

Download: Data Folder, Data Set Description

Abstract: This is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail.

Data Set Characteristics:	Multivariate, Sequential, Time-Series	**Number of Instances:**	541909	**Area:**	Business
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	8	**Date Donated**	2015-11-06
Associated Tasks:	Classification, Clustering	**Missing Values?**	N/A	**Number of Web Hits:**	338678

Source:

Dr Daqing Chen, Director: Public Analytics group. chend '@' lsbu.ac.uk, School of Engineering, London South Bank University, London SE1 0AA, UK.

Data Set Information:

This is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail.The company

Attribute Information:

InvoiceNo: Invoice number. Nominal, a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation.
StockCode: Product (item) code. Nominal, a 5-digit integral number uniquely assigned to each distinct product.
Description: Product (item) name. Nominal.
Quantity: The quantities of each product (item) per transaction. Numeric.
InvoiceDate: Invice Date and time. Numeric, the day and time when each transaction was generated.
UnitPrice: Unit price. Numeric, Product price per unit in sterling.
CustomerID: Customer number. Nominal, a 5-digit integral number uniquely assigned to each customer.
Country: Country name. Nominal, the name of the country where each customer resides.

https://archive.ics.uci.edu/ml/datasets/Online%20Retail

Data Description

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010/12/1 08:26	2.55	17850	United Kingdom
536365	71053	WHITE METAL LANTERN	6	2010/12/1 08:26	3.39	17850	United Kingdom
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010/12/1 08:26	2.75	17850	United Kingdom
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010/12/1 08:26	3.39	17850	United Kingdom
536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010/12/1 08:26	3.39	17850	United Kingdom
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010/12/1 08:26	7.65	17850	United Kingdom
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010/12/1 08:26	4.25	17850	United Kingdom
536366	22633	HAND WARMER UNION JACK	6	2010/12/1 08:28	1.85	17850	United Kingdom
536366	22632	HAND WARMER RED POLKA DOT	6	2010/12/1 08:28	1.85	17850	United Kingdom
536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	2010/12/1 08:34	1.69	13047	United Kingdom
536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	2010/12/1 08:34	2.1	13047	United Kingdom
536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	2010/12/1 08:34	2.1	13047	United Kingdom
536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	2010/12/1 08:34	3.75	13047	United Kingdom
536367	22310	IVORY KNITTED MUG COSY	6	2010/12/1 08:34	1.65	13047	United Kingdom
536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	6	2010/12/1 08:34	4.25	13047	United Kingdom
536367	22623	BOX OF VINTAGE JIGSAW BLOCKS	3	2010/12/1 08:34	4.95	13047	United Kingdom
536367	22622	BOX OF VINTAGE ALPHABET BLOCKS	2	2010/12/1 08:34	9.95	13047	United Kingdom
536367	21754	HOME BUILDING BLOCK WORD	3	2010/12/1 08:34	5.95	13047	United Kingdom
536367	21755	LOVE BUILDING BLOCK WORD	3	2010/12/1 08:34	5.95	13047	United Kingdom
536367	21777	RECIPE BOX WITH METAL HEART	4	2010/12/1 08:34	7.95	13047	United Kingdom
536367	48187	DOORMAT NEW ENGLAND	4	2010/12/1 08:34	7.95	13047	United Kingdom
536368	22960	JAM MAKING SET WITH JARS	6	2010/12/1 08:34	4.25	13047	United Kingdom
536368	22913	RED COAT RACK PARIS FASHION	3	2010/12/1 08:34	4.95	13047	United Kingdom
536368	22912	YELLOW COAT RACK PARIS FASHION	3	2010/12/1 08:34	4.95	13047	United Kingdom
536368	22914	BLUE COAT RACK PARIS FASHION	3	2010/12/1 08:34	4.95	13047	United Kingdom
536369	21756	BATH BUILDING BLOCK WORD	3	2010/12/1 08:35	5.95	13047	United Kingdom
536370	22728	ALARM CLOCK BAKELIKE PINK	24	2010/12/1 08:45	3.75	12583	France
536370	22727	ALARM CLOCK BAKELIKE RED	24	2010/12/1 08:45	3.75	12583	France
536370	22726	ALARM CLOCK BAKELIKE GREEN	12	2010/12/1 08:45	3.75	12583	France
536370	21724	PANDA AND BUNNIES STICKER SHEET	12	2010/12/1 08:45	0.85	12583	France
536370	21883	STARS GIFT TAPE	24	2010/12/1 08:45	0.65	12583	France
536370	10002	INFLATABLE POLITICAL GLOBE	48	2010/12/1 08:45	0.85	12583	France
536370	21791	VINTAGE HEADS AND TAILS CARD GAME	24	2010/12/1 08:45	1.25	12583	France

https://archive.ics.uci.edu/ml/datasets/Online%20Retail

Step 1 - Install mlxtend

pip install mlxtend xlrd

```
C:\Users\user>pip install mlxtend
Collecting mlxtend
  Downloading https://files.pythonhosted.org/packages/c0/ca/54fe0ae783ce81a467710d1c5fb41cfca0751
/mlxtend-0.16.0-py2.py3-none-any.whl (1.3MB)
    100% |████████████████████████████████| 1.3MB 754kB/s
Requirement already satisfied: scipy>=0.17 in c:\users\user\appdata\local\programs\python\python3
om mlxtend) (1.0.1)
Requirement already satisfied: setuptools in c:\users\user\appdata\local\programs\python\python36
m mlxtend) (39.0.1)
Requirement already satisfied: scikit-learn>=0.18 in c:\users\user\appdata\local\programs\python\
ges (from mlxtend) (0.19.1)
Requirement already satisfied: numpy>=1.10.4 in c:\users\user\appdata\local\programs\python\pytho
from mlxtend) (1.14.2)
Requirement already satisfied: matplotlib>=1.5.1 in c:\users\user\appdata\local\programs\python\p
es (from mlxtend) (2.2.2)
Requirement already satisfied: pandas>=0.17.1 in c:\users\user\appdata\local\programs\python\pyth
(from mlxtend) (0.22.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\user\appdata\local\programs\pytho
kages (from matplotlib>=1.5.1->mlxtend) (2.6.1)
Requirement already satisfied: six>=1.10 in c:\users\user\appdata\local\programs\python\python36\
 matplotlib>=1.5.1->mlxtend) (1.11.0)
Requirement already satisfied: cycler>=0.10 in c:\users\user\appdata\local\programs\python\python
rom matplotlib>=1.5.1->mlxtend) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\user\appdata\
ython36\lib\site-packages (from matplotlib>=1.5.1->mlxtend) (2.2.0)
Requirement already satisfied: pytz in c:\users\user\appdata\local\programs\python\python36\lib\s
lotlib>=1.5.1->mlxtend) (2018.3)
```

Step 2 – Import Libs & Data Preprocessing

```python
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

df=pd.read_excel('Online Retail.xlsx')
df.head()

df['Description'] = df['Description'].str.strip()
df.dropna(axis=0, subset=['InvoiceNo'], inplace=True)
df['InvoiceNo'] = df['InvoiceNo'].astype('str')
df = df[~df['InvoiceNo'].str.contains('C')]

basket = (df[df['Country'] =="France"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))
```

Step 2 – Import Libs & Data Preprocessing

Description InvoiceNo	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	12 PENCILS TALL TUBE POSY	12 PENCILS TALL TUBE RED RETROSPOT
536370	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
536852	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
536974	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
537065	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
537463	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
537468	24.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
537693	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
537897	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
537967	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
538008	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
538093	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
538196	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
539050	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
539113	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
539407	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
539435	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
539551	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
539607	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
539688	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
539727	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Step 3 – Data Preprocessing & Association Rule Learning

```python
def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1

basket_sets = basket.applymap(encode_units)
basket_sets.drop('POSTAGE', inplace=True, axis=1)

print(type(basket_sets))


frequent_itemsets = apriori(basket_sets, min_support=0.07, use_colnames=True)

rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules.head()

print(rules[ (rules['lift'] >= 6) & (rules['confidence'] >= 0.8) ])
```

Step 3 – Results

	antecedents \	consequents	antecedent support	\consequent support	support	confidence	lift	leverage	conviction
2	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.096939	0.094388	0.079082	0.815789	8.642959	0.069932	4.916181
3	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.094388	0.096939	0.079082	0.837838	8.642959	0.069932	5.568878
16	(SET/6 RED SPOTTY PAPER PLATES)	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.127551	0.132653	0.102041	0.800000	6.030769	0.085121	4.336735
18	(SET/6 RED SPOTTY PAPER PLATES)	(SET/6 RED SPOTTY PAPER CUPS)	0.127551	0.137755	0.122449	0.960000	6.968889	0.104878	21.556122
19	(SET/6 RED SPOTTY PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES)	0.137755	0.127551	0.122449	0.888889	6.968889	0.104878	7.852041
20	(SET/6 RED SPOTTY PAPER PLATES, SET/6 RED SPOT...	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.122449	0.132653	0.099490	0.812500	6.125000	0.083247	4.625850
21	(SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET...	(SET/6 RED SPOTTY PAPER CUPS)	0.102041	0.137755	0.099490	0.975000	7.077778	0.085433	34.489796
22	(SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...	(SET/6 RED SPOTTY PAPER PLATES)	0.102041	0.127551	0.099490	0.975000	7.644000	0.086474	34.897959

Apriori

Case 2

Generating Association Rules from Frequent Itemsets

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


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']]

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)

frequent_itemsets
```

Generating Association Rules from Frequent Itemsets

	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	(Kidney Beans, Eggs)
6	0.6	(Onion, Eggs)
7	0.6	(Milk, Kidney Beans)
8	0.6	(Onion, Kidney Beans)
9	0.6	(Kidney Beans, Yogurt)
10	0.6	(Onion, Kidney Beans, Eggs)

Generating Association Rules from Frequent Itemsets

```python
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	lift	leverage	conviction
0	(Kidney Beans)	(Eggs)	1.0	0.8	0.8	0.80	1.00	0.00	1.000000
1	(Eggs)	(Kidney Beans)	0.8	1.0	0.8	1.00	1.00	0.00	inf
2	(Onion)	(Eggs)	0.6	0.8	0.6	1.00	1.25	0.12	inf
3	(Eggs)	(Onion)	0.8	0.6	0.6	0.75	1.25	0.12	1.600000
4	(Milk)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00	inf
5	(Onion)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00	inf
6	(Yogurt)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00	inf
7	(Onion, Kidney Beans)	(Eggs)	0.6	0.8	0.6	1.00	1.25	0.12	inf
8	(Onion, Eggs)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00	inf
9	(Kidney Beans, Eggs)	(Onion)	0.8	0.6	0.6	0.75	1.25	0.12	1.600000
10	(Onion)	(Kidney Beans, Eggs)	0.6	0.8	0.6	1.00	1.25	0.12	inf
11	(Eggs)	(Onion, Kidney Beans)	0.8	0.6	0.6	0.75	1.25	0.12	1.600000

Other Tools

- PyFIM

http://www.borgelt.net/pyfim.html

```python
import fim

# 啤酒和尿布数据
tracts = [\
    ['牛奶','面包'],\
    ['面包','尿布','啤酒','鸡蛋'],\
    ['牛奶','尿布','啤酒','可乐'],\
    ['面包','牛奶','尿布','啤酒'],\
    ['面包','牛奶','尿布','可乐'],\
]

# 关联分析，设置支持度至少 60%，自信度至少 80%
r = fim.fpgrowth(tracts, zmin=2, supp=60, conf=80, target='r')
print(r)
```

得到结果：

```
[('尿布', ('啤酒',), 3)]
```

Reference

1. http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/
2. https://pbpython.com/market-basket-analysis.html
3. https://zhuanlan.zhihu.com/p/30600248

Thank you