NOTES 9

Association Rule — Unsupervised Learning

Acknowledgement: some contents are based on Pang-Ning Tan, Michael Steinbach and Vipin Kumar, Introduction to Data Mining, Addison-Wesley. Chapter 6: Association Analysis: Basic Concepts and Algorithms. Available at: http://www-users.cs.umn.edu/~kumar/dmbook/index.php

Association Rule

- Discover interesting correlation or association in large databases

- A rule of the form
  - X → Y: if X then Y
  - X: LHS(LEFT-HAND-SIDE) & Y: RHS (RIGHT-HAND-SIDE)
  - X: antecedent & Y: succedent
  - X and Y are disjoint

- Can be applied for Market basket analysis: associations between items in the shopping cart

An Example from Tan et al.

TID	Items
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

- Transactions type data

Terminology

- A Binary Representation

TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

- k-item set
  - Item - an attribute/value pair
  - k Item set – a combination of items contains k items
  - {Milk, Bread}: 2 item set

Support, Confidence

- Support count (σ)
  - σ(X): the number of occurrence of an item set (X) in the database
  - σ({Bread, Milk})=3

- Support (Coverage):
  - the proportion of transactions that contain both X and Y
  - $s(X \rightarrow Y) = \sigma(X \cup Y) / N$, where N is the total number of transactions in the database. $s(\{Bread, Milk\}) = \sigma(\{B, M\})/N = \frac{3}{5}$

- Confidence (Accuracy):
  - the proportion of the transactions that contains X which also contains Y
  - $c(X \rightarrow Y) = \sigma(X \cup Y) / \sigma(X)$

The Objective of Association Rule Mining

- Given a set of transactions database, find all rules satisfying:
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- An item set whose support is greater than or equal to a minsup threshold is called as Frequent Item set

- Want rules with high coverage/support

Rule Mining

STEP 1: Find all item sets that meet minimum coverage / Support

STEP 2: Find all rules that meet minimum accuracy / Confidence

STEP 3: Prune

Two ways to find Frequent Item Set

Apriori Algorithm

- Principle:
  - If an item set is frequent, then all of its subsets must also be frequent

- Procedure:
  - Create a list of candidates, one-item subsets of the feature space.
  - For each candidate count support in data.
  - Discard candidates with support less than t (predefined threshold).
  - For each length i = 2, . . .
    - Generate list of candidates of the length i. Join any two candidates from previous step having i − 2 elements common.
    - For each candidate, count support in data.
    - Discard candidates with support < t.
  - Until empty list of candidates.

An Illustrated Example from Tan et al. Min Support Count =3 ← Set threshold as 3 . 1 item set

Candidate Sets: Before Pruning

Item Set	Support Count
Beer	3
Bread	4
~~Coke~~	~~2~~ <3
Diaper	4
~~Eggs~~	~~1~~ <1
Milk	4

Frequent Sets: After Pruning

Item Set	Support Count
Beer	3
Bread	4
Diaper	4
Milk	4

An Illustrated Example from Tan et al. Min Support Count =3

2 item set

Candidate Sets:
Before Pruning

Item Set	Support Count
~~Beer, Bread~~	~~2~~ <3
Beer, Diaper	3
~~Beer, Milk~~	~~2~~ <3
Bread, Diaper	3
Bread, Milk	3
Diaper, Milk	3

Frequent Sets:
After Pruning

Item Set	Support Count
Beer, Diaper	3
Bread, Diaper	3
Bread, Milk	3
Diaper, Milk	3

An Illustrated Example from Tan et al. Min Support Count =3

3 item set

Candidate Sets:
Before Pruning

Item Set	Support Count
Bread, Diaper, Milk	2

Frequent Sets:
After Pruning

Item Set	Support Count

Property of Apriori

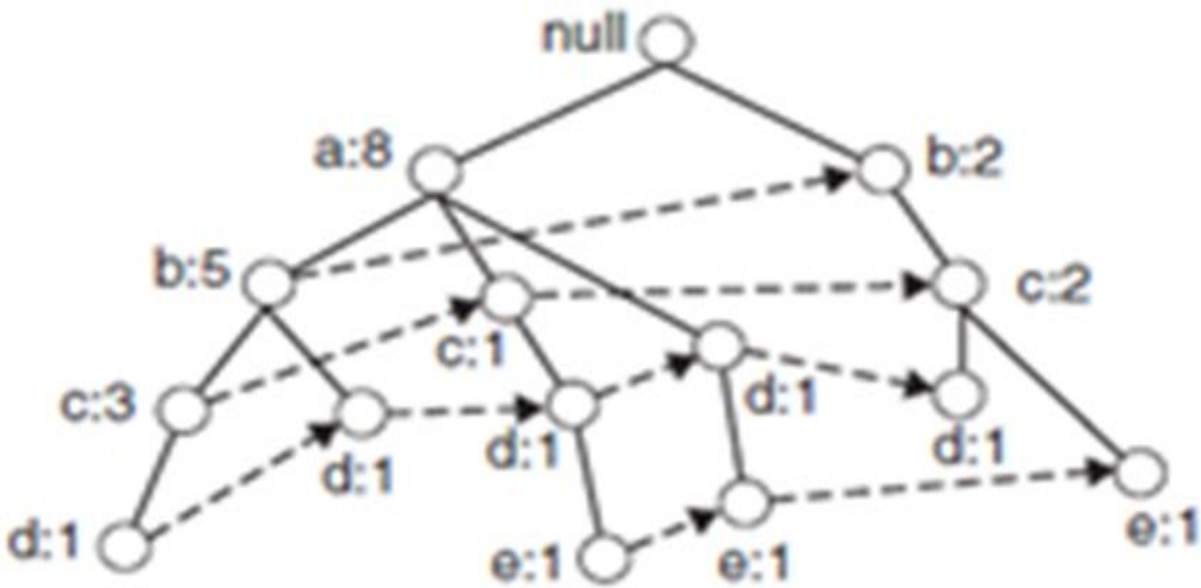
- Generate candidates items and test if they are frequent
  - If every subset is considered, there are 41 rules.
  - But here we only check 13.
- Apriori algorithm improves performance by using candidate item sets
  - Costly to generate large number of item sets
  - Support counting is expensive

2.
Frequent Pattern Tree (FP-tree)

- Encodes the data set using a compact data structure
- Extract frequent item sets from the FP-tree

Transaction
Data Set

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}



(iv) After reading TID=10

Figure 6.24. Construction of an FP-tree.

FP-tree Terminology

Set threshold = 2

support count

- Start with Root/Null node
- Each node has three fields
  - item name
  - count
  - node link
- Dotted lines connect the same item
- Remove infrequent 1-item (pruning)
- Different transactions have several items in common, their path can overlap.
- Items in transactions use the fixed order. Thus paths can overlap when transaction share the same items, or the same prefix.
  - sort items based on their support in decreasing order in each transaction.

End with e

e	3
de	2
ce	2
be	1
ae	2

cde	1
bde	0
ade	2

FP-tree Construction

Transaction Data Set

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}

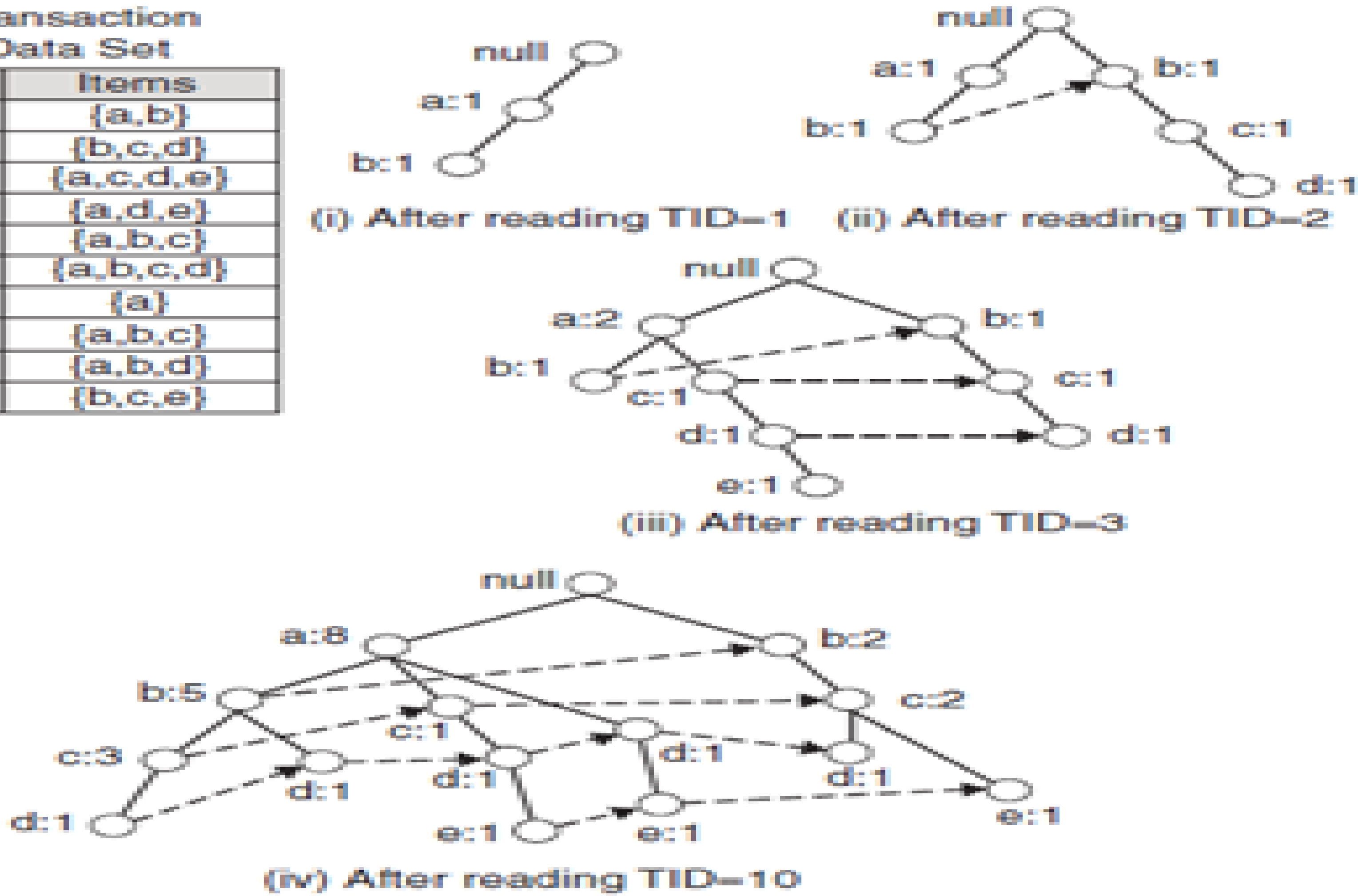


Figure 6.24. Construction of an FP-tree.

Frequent Item Set Generation

- Bottom-Up
  - first look up frequent item ending with e, then de…then d,…then c, b, a

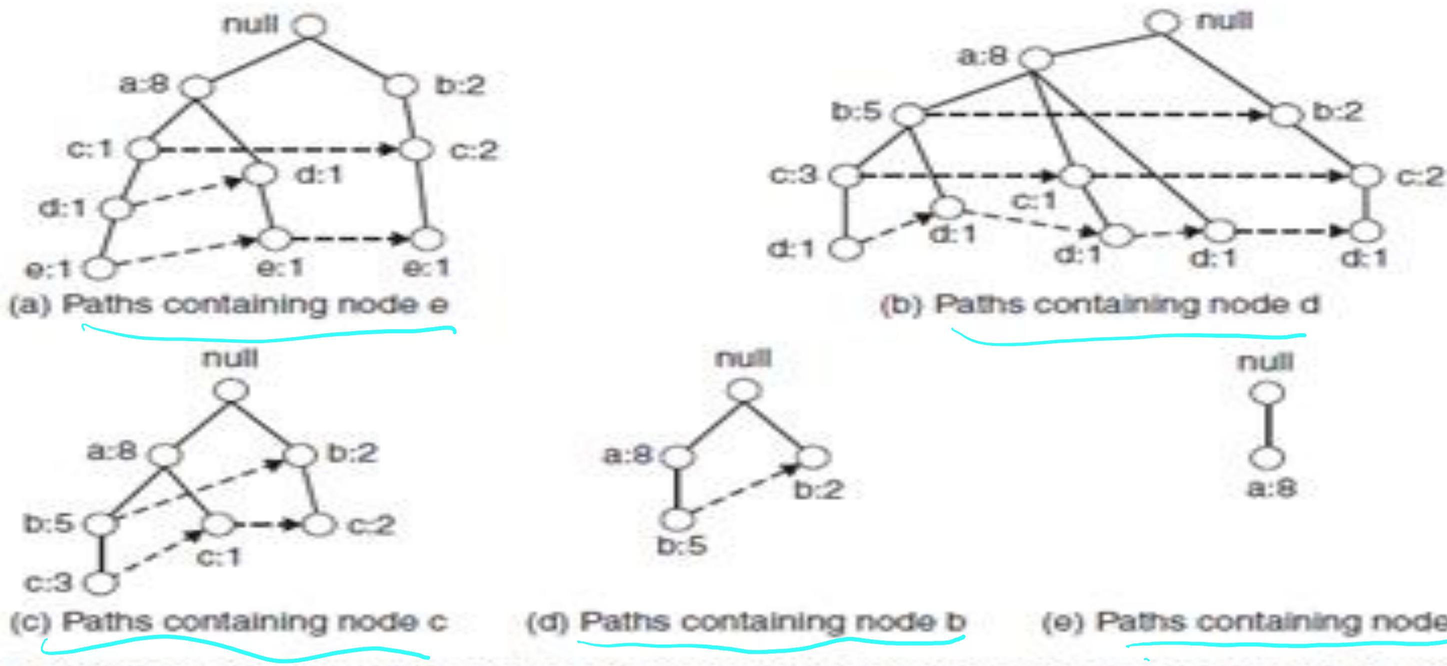


Figure 6.26. Decomposing the frequent itemset generation problem into multiple subproblems, where each subproblem involves finding frequent itemsets ending in e, d, c, b, and a.

Frequent Item Set Results

- e.g. if minsup=2

Table 6.6. The list of frequent itemsets ordered by their corresponding suffixes.

Suffix	Frequent Itemsets
e	{e}, {d,e}, {a,d,e}, {c,e},{a,e}
d	{d}, {c,d}, {b,c,d}, {a,c,d}, {b,d}, {a,b,d}, {a,d}
c	{c}, {b,c}, {a,b,c}, {a,c}
b	{b}, {a,b}
a	{a}

Discussion

- FP-tree is much faster than Apriori
- Once it is built, the frequent item sets can be read easily
- However, support can only be calculated after scanning the entire database.
- Time is wasted if the threshold of support is high

Rule Generation

- Given a frequent item set L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
  - If {A,B,C,D} is a frequent item set, candidate rules:

$ABC \rightarrow D$, $ABD \rightarrow C$, $ACD \rightarrow B$, $BCD \rightarrow A$,
$A \rightarrow BCD$, $B \rightarrow ACD$, $C \rightarrow ABD$, $D \rightarrow ABC$
$AB \rightarrow CD$, $AC \rightarrow BD$, $AD \rightarrow BC$, $BC \rightarrow AD$,
$BD \rightarrow AC$, $CD \rightarrow AB$,

e.g., L = {A,B,C,D}:

$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

$$\frac{\sigma(ABC \cup D)}{\sigma(ABC)} \geq \frac{\sigma(AB \cup CD)}{\sigma(AB)} \quad b/c \quad \sigma(ABC) < \sigma(AB)$$

Pattern Evaluation

- Association rule algorithms tend to produce too many rules
  - many of them are uninteresting or redundant
  - Redundant if $\{A,B,C\} \rightarrow \{D\}$ and $\{A,B\} \rightarrow \{D\}$ have same support & confidence

- Interestingness measures can be used to prune/rank the derived patterns

- In the original formulation of association rules, support & confidence are the only measures used

- However, confidence may be misleading

Contingency table for $X \rightarrow Y$

non-Y

	Y	$\bar{Y}$	
X	f_{11}	f_{10}	f_{1+}
non-X $\bar{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	$\|T\|$

f_{11}: support amount of X and Y
f_{10}: support amount of X and $\bar{Y}$
f_{01}: support amount of $\bar{X}$ and Y
f_{00}: support amount of $\bar{X}$and $\bar{Y}$

e.g.

	$Coffee$	$\overline{Coffee}$	
Tea	15	5	20
$\overline{Tea}$	75	5	80
	90	10	100

2f tea → coffee

$\frac{15}{20} = \frac{3}{4} < 90\%$

tea → coffee

negative

? Interpret

90%

Interest Factor

support

- I(X,Y)= s (X, Y)/ (s(X)* s(Y)) = $\dfrac{\frac{15}{1000}}{\frac{20}{1000}\cdot\frac{90}{1000}} < 1$

- $I(X,Y)=\begin{cases} 1, & dependent \\ > 1, & positive\ correlated \\ < 1, & negative\ correlated \end{cases}$

Association Rule Application

- Product associations

$$X \text{ and } Y \Rightarrow Z \ (85\%)$$

- User associations

- Collaborative filtering is based on correlation

- Association rules has broader application

Association Rule Example

- titanic.csv
- This data set provides the survival information of passengers on the titanic. There are 2201 observations on 4 variables.

```
> head(titanic)
  Class  Sex   Age Survived
1   3rd Male Child       No
2   3rd Male Child       No
3   3rd Male Child       No
4   3rd Male Child       No
5   3rd Male Child       No
6   3rd Male Child       No
```

RQ: Which factors may affect the survival rates of passengers.

```
library(arules)
rules <- apriori(titanic,parameter = list(minlen=2,
supp=0.1,conf=0.2))
```

threshold

```
> rules
set of 63 rules
> inspect(rules)
    lhs                rhs               support  confidence
lift
1   {Class=2nd}     => {Age=Adult}     0.1185825  0.9157895
0.9635051
2   {Class=1st}     => {Age=Adult}     0.1449341  0.9815385
1.0326798
3   {Sex=Female}    => {Survived=Yes} 0.1562926  0.7319149
2.2657450
4   {Survived=Yes} => {Sex=Female}    0.1562926  0.4838256
2.2657450
5   {Sex=Female}    => {Age=Adult}     0.1930940  0.9042553
0.9513700
......
```

Even though high, not sth we're looking for.

- Can determine the items shown in RHS or LHS, e.g. rules with the only item survival in RHS.

threshold

```
rules1 <- apriori(titanic,parameter = list(minlen=2,
supp=0.1,conf=0.2), appearance =
list(rhs=c("Survived=No",
"Survived=Yes"),default="lhs"))
```

only want survived in the right side rules.

```
> rules1
set of 16 rules
> inspect(rules1)
    lhs                rhs                support   confidence
lift
1   {Sex=Female} => {Survived=Yes} 0.1562926  0.7319149
2.2657450
2   {Class=3rd}  => {Survived=No}  0.2398910  0.7478754
1.1047474
3   {Sex=Male}   => {Survived=Yes} 0.1667424  0.2120162
0.6563257
4   {Age=Adult}  => {Survived=Yes} 0.2971377  0.3126195
0.9677574
5   {Class=Crew} => {Survived=No}  0.3057701  0.7604520
1.1233254
```

interest

<1 negative influence

- Find rules with LHS“Class=1st”, “Class=2nd”, and “Age=Child”, and no other items (default=”none”).

```
rules2 <- apriori(titanic,parameter = list(minlen=2,
supp=0.001,conf=0.02), appearance =
list(rhs=c("Survived=Yes"),lhs=c("Class=1st","Class=
2nd","Age=Child"),default="none"))
```

```
> inspect(rules2)
  lhs                rhs                support confidence
lift
1 {Age=Child}   => {Survived=Yes} 0.025897319  0.5229358
1.618821
2 {Class=2nd}   => {Survived=Yes} 0.053611995  0.4140351
1.281704
3 {Class=1st}   => {Survived=Yes} 0.092230804  0.6246154
1.933584
4 {Class=2nd,
   Age=Child}   => {Survived=Yes} 0.010904134  1.0000000
3.095640
5 {Class=1st,
   Age=Child}   => {Survived=Yes} 0.002726034  1.0000000
3.095640
```

Rules can be sorted based on the evaluation measures, e.g. confidence, support or lift.

```
rules.new <- sort(rules1, by="confidence")
inspect(rules.new)
```

When a rule is a super rule of another rule but the former one has the same or a lower lift, it is considered as redundant.

remove the redundant rules.

```
subset <- is.subset(rules.new, rules.new)
subset[lower.tri(subset, diag=T)] <- NA
redundant <- colSums(subset, na.rm=T) >= 1
which(redundant)
pruned.rules <- rules.new[!redundant]
```

The function apriori() can be used to identify frequent item sets, e.g. all 2-item set with the minimum support as 0.2.

```
items <- apriori(titanic,parameter = list(minlen=2,
maxlen=2,supp=0.2,target="frequent itemsets"))
```

```
> inspect(items)
   items              support
1  {Class=3rd,
    Survived=No}  0.2398910
2  {Class=3rd,
    Sex=Male}     0.2317129
3  {Class=3rd,
    Age=Adult}    0.2848705
4  {Age=Adult,
    Survived=Yes} 0.2971377
5  {Class=Crew,
    Survived=No}  0.3057701
6  {Class=Crew,
    Sex=Male}     0.3916402
7  {Class=Crew,
    Age=Adult}    0.4020900
8  {Sex=Male,
    Survived=No}  0.6197183
9  {Age=Adult,
    Survived=No}  0.6533394
10 {Sex=Male,
    Age=Adult}    0.7573830
```

Item set

~~Nothing to verify~~

No need Evaluation