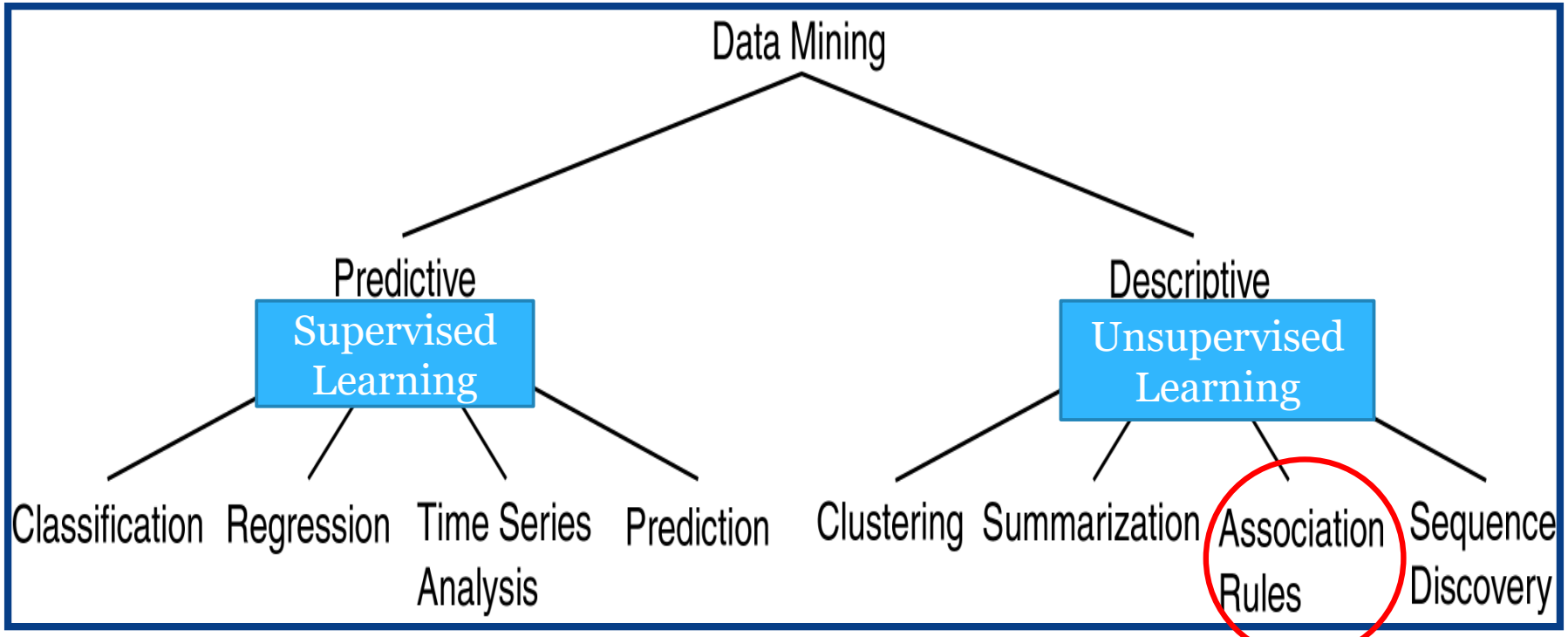
Unsupervised Learning: Association Analysis with Apriori Algorithm

Dr. Yuzana Win (Nagasaki University, Japan)
Lecturer
Department of Computer Engineering and Information Technology

Lecture Objectives

- To introduce
  - What is Association Rules Analysis?
  - Association analysis with Apriori algorithm
  - How does the Apriori algorithm work?
  - Why Apriori algorithm is used for?

Data Mining Methods and Models



Basket Data (called Transaction Data)

Transaction data => each record represents a transaction between a customer and a shop

CEIT
Think smart, be different.

Association Rules Analysis

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

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

Example of Association Rules

{Diaper} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {Milk},

more item frequently => to increase profit

The process of identifying an associates between products => association rule mining

Discovering Rules

- According to the transaction data, we find the useful rule such that

  If a basket contains bread and milk, then it also contains diaper

Any such rule has two associated measures:

1. *confidence* – when the `if' part is true, how often is the `then' also true? This is the same as *accuracy*.
2. *coverage* or *support* – how much of the database contains the `if' part?

Definition: Association Rule

- **Association Rule**
  - An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
  - Example:
    {Milk, Diaper} $\rightarrow$ {Beer}

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

- **Rule Evaluation Metrics**
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

Example:

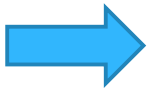
$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold



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

Algorithm to Generate Association Rule

Input:

D //Database of transactions
I //Items
L //Large itemsets
s //Support
α //Confidence

Output:

R //Association Rules satisfying s and α

ARGen Algorithm:

$R = \emptyset$;
for each $l \in L$ **do**
  for each $x \subset l$ **such that** $x \neq \emptyset$ **and** $x \neq l$ **do**
    if $\frac{support(l)}{support(x)} \geq \alpha$ **then**
      $R = R \cup \{x \Rightarrow (l - x)\}$;

Association Rule Mining Task

1. Find Large Itemsets

1. Generate rules from frequent itemsets and find all rules having
   - support ≥ *minsup* threshold
   - confidence ≥ *minconf* threshold

Mining Association Rules

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

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

Observations:

- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence

Association Rules Algorithm

- Apriori Algorithm
- FP-growth Algorithm
- ECLAT

Apriori Algorithm

- Apriori is simple, fast and very good at finding interesting rules of a specific kind in baskets or other transaction data
- A candidate generation-and-test Approach [Jiawei Han , 2011]
- Given a frequent itemset, its subset must be frequent

Apriori Algorithm

- Finding rules in two stages

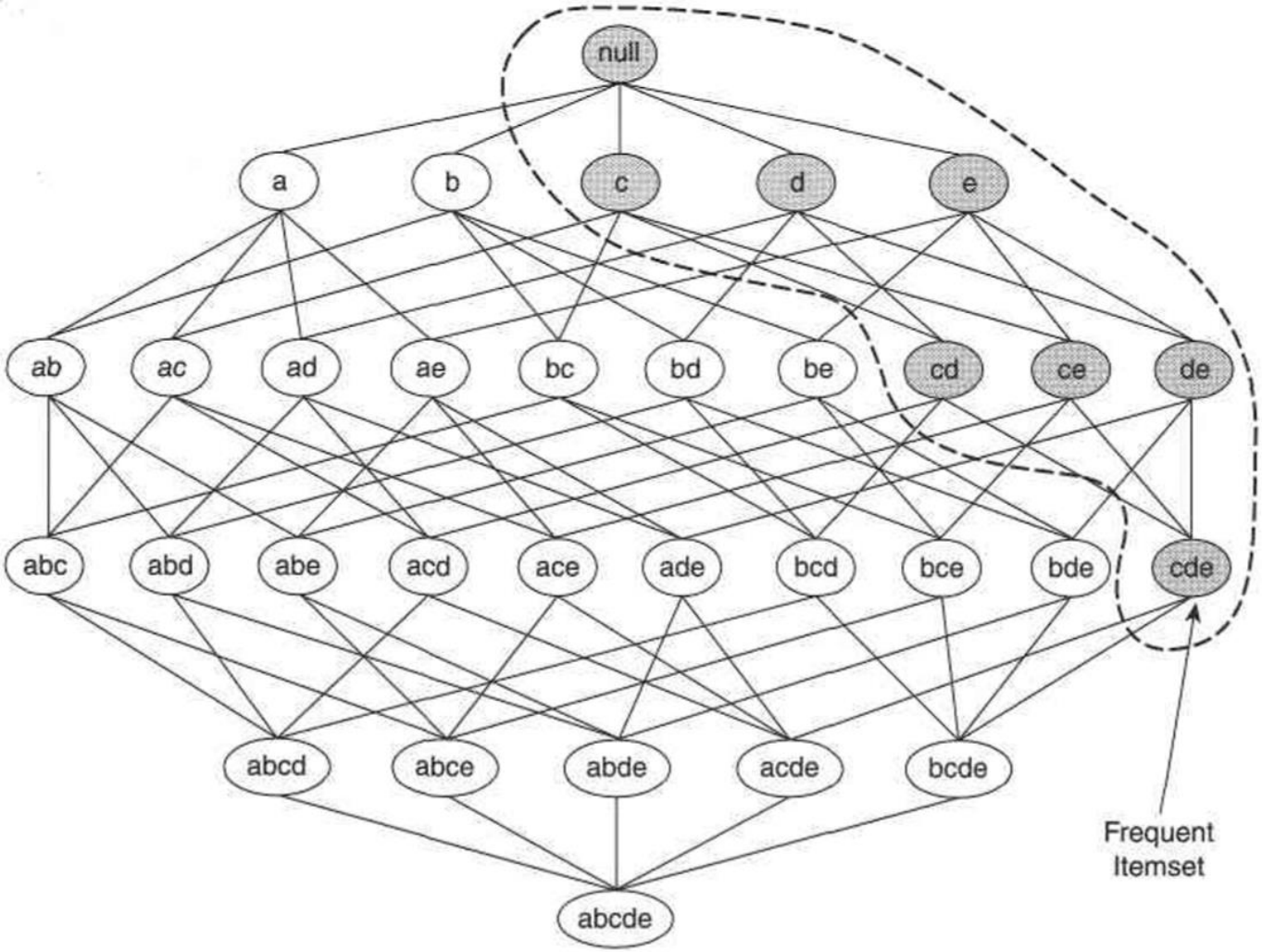
(1) Find all itemsets with a specified minimal support

(2) Use these itemsets to help generate

Apriori Algorithm

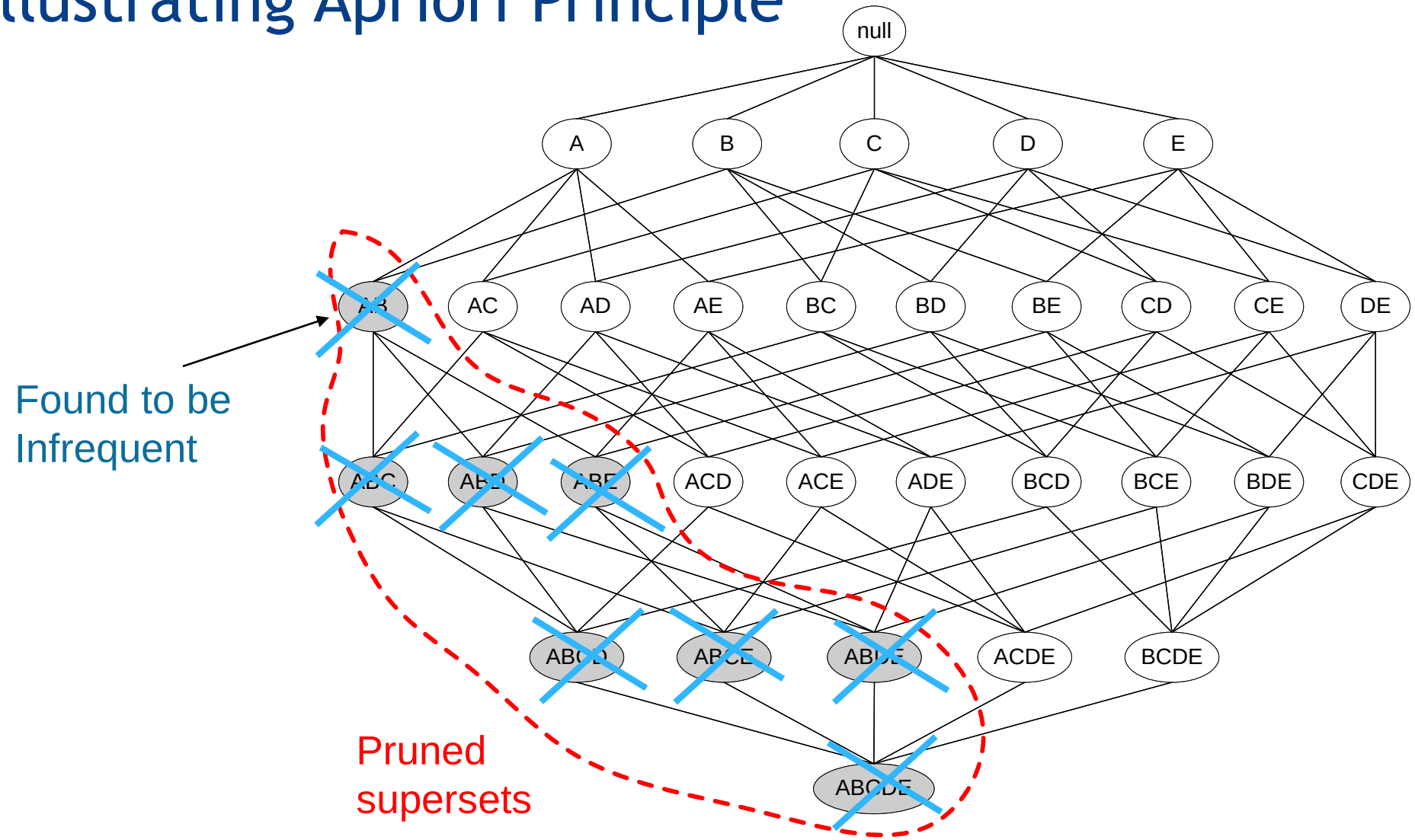
1. C_1 = Itemsets of size one in I;
2. Determine all large itemsets of size 1, L_1;
3. i = 1;
4. Repeat
5. i = i + 1;
6. C_i = Apriori-Gen(L_{i-1});
7. Count C_i to determine L_i;
8. until no more large itemsets found;

Illustrating Apriori Principle



An illustration of the Apriori principle. If {c,d,e} is frequent => all subsets of this itemset are frequent

Illustrating Apriori Principle



An illustration of support-based pruning. If {a,b} is infrequent => then all supersets of {a,b} are infrequent

Illustrating Apriori Principle

Items (1-itemsets)

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items remove because of low support

Pairs (2-itemsets)

Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

Minimum Support = 3

If every subset is considered,
$^{6}C_{1} + {}^{6}C_{2} + {}^{6}C_{3} = 41$
With support-based pruning,
$6 + 6 + 1 = 13$

Candidate Generation and Pruning

- Candidate Generation:
  - generates new candidate k-itemsets based on the frequent (k-1) itemsets

Items

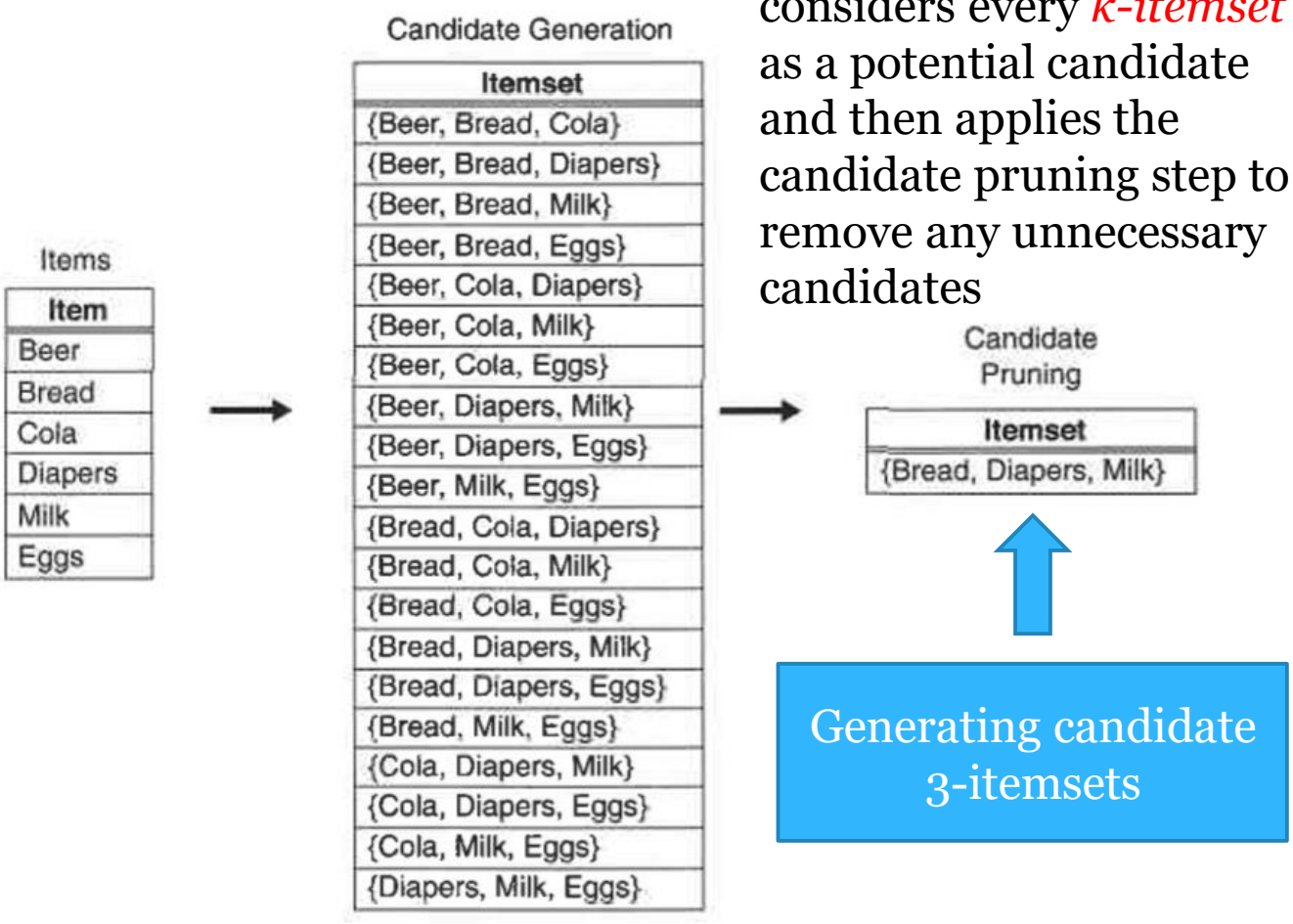
Item
Beer
Bread
Cola
Diapers
Milk
Eggs

Candidate Generation

Itemset
{Beer, Bread, Cola}
{Beer, Bread, Diapers}
{Beer, Bread, Milk}
{Beer, Bread, Eggs}
{Beer, Cola, Diapers}
{Beer, Cola, Milk}
{Beer, Cola, Eggs}
{Beer, Diapers, Milk}
{Beer, Diapers, Eggs}
{Beer, Milk, Eggs}
{Bread, Cola, Diapers}
{Bread, Cola, Milk}
{Bread, Cola, Eggs}
{Bread, Diapers, Milk}
{Bread, Diapers, Eggs}
{Bread, Milk, Eggs}
{Cola, Diapers, Milk}
{Cola, Diapers, Eggs}
{Cola, Milk, Eggs}
{Diapers, Milk, Eggs}

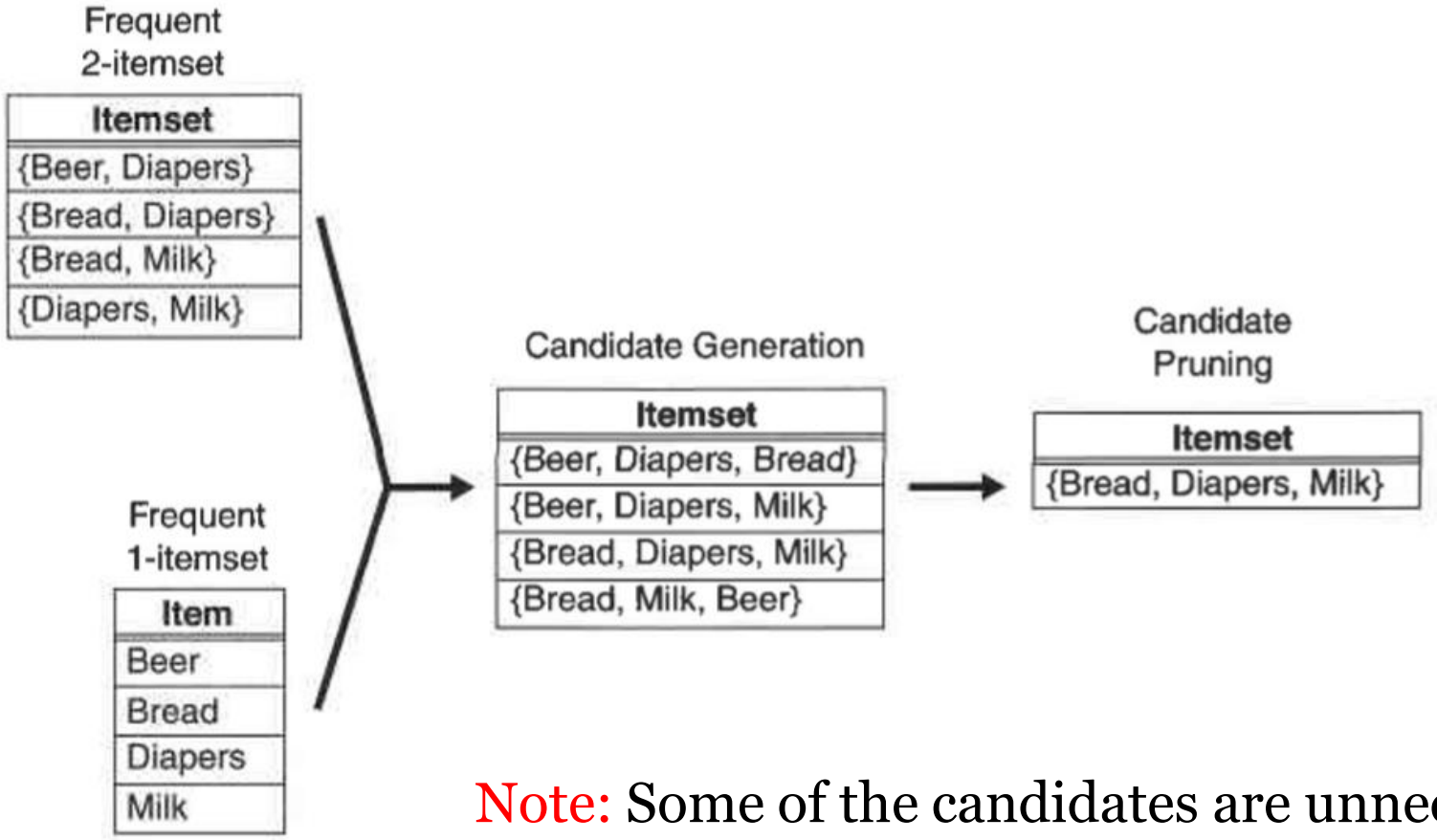
Candidate Generation and Pruning

- Candidate Pruning:
  - Eliminates some of the candidate k-itemsets using the support-based pruning strategy



Candidate Generation and Pruning

- Generaing and pruning candidate *k*-itemsets by merging a frequent (k-1) itemset with a frequent item.



Note: Some of the candidates are unnecessary because their subsets are infrequent

Advantages and Disadvantages

- ***Advantages:***
  - Uses large itemset property
  - Easily parallelized
  - Easy to implement

- ***Disadvantages:***
  - Assumes transaction database is memory resident
  - Requires up to *m* database scans

CEIT
Think smart, be different.

Implementing Apriori Algorithm with Python

- ***Import the libraries and download dataset***

  https://drive.google.com/file/d/1y5DYn0dGoSbC22xowBq2d4po6h1JxcTQ/view?usp=sharing

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from apyori import apriori
```

```
store_data = pd.read_csv('D:\\Datasets\\store_data.csv')
```

```
store_data.head()
```

	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
0	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Implementing Apriori Algorithm with Python

```
store_data = pd.read_csv('D:\\Datasets\\store_data.csv', header=None)
```

```
store_data.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
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	frozen smoothie	s
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

Implementing Apriori Algorithm with Python

```
association_rules = apriori(records, min_support=0.0045, min_confidence=0.2, min_
lift=3, min_length=2)
association_results = list(association_rules)
```

```
print(len(association_rules))
```

```
print(association_rules[0])
```

```
RelationRecord(items=frozenset({'light cream', 'chicken'}), support=0.00453272896
9470737, ordered_statistics[OrderedStatistic(items_base=frozenset({'light cream
'}), items_add=frozenset({'chicken'}), confidence=0.290598290598290 57, lift=4.843
95061728395)])
```

Implementing Apriori Algorithm with Python

```
for item in association_rules:

    # first index of the inner list
    # Contains base item and add item
    pair = item[0]
    items = [x for x in pair]
    print("Rule: " + items[0] + " -> " + items[1])

    #second index of the inner list
    print("Support: " + str(item[1]))

    #third index of the list located at 0th
    #of the third index of the inner list

    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("====================================")
```

Implementing Apriori Algorithm with Python

```
Rule: light cream -> chicken
Support: 0.004532728969470737
Confidence: 0.29059829059829057
Lift: 4.84395061728395
=====================================
Rule: mushroom cream sauce -> escalope
Support: 0.005732568990801126
Confidence: 0.3006993006993007
Lift: 3.790832696715049
=====================================
Rule: escalope -> pasta
Support: 0.005865884548726837
Confidence: 0.3728813559322034
Lift: 4.700811850163794
=====================================
Rule: ground beef -> herb & pepper
Support: 0.015997866951073192
Confidence: 0.3234501347708895
Lift: 3.2919938411349285
=====================================
```

Implementing Apriori Algorithm with Python

- ***According to the experimental results,***
  - Lift of 3.79 shows that
  - the escalope is 3.79 more likely to be bought by the customers buy mushroom cream and sauce

References

- Jiawei Han and Micheline Kamber. *Data Mining - Concepts and Techniques*. MorganKaufmann Publishers, 2001
- Introduction to Data Mining, Tan, Steinbach, Kumar, 2004
- https://stackabuse.com/association-rule-mining-via-apriori-algorithm-in-python/

Next Week Lecture

- Unsupervised Learning: Association Analysis with FP-Growth Algorithm

Thank You