

# Naïve Bayes Classifier 복습

- Given an input vector  $\mathbf{x} = (x_1, x_2, \dots, x_n)$
- Computes  $p(C_k|\mathbf{x})$  for each of  $K$  possible classes  $C_k$

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)}{p(\mathbf{x})} p(C_k)$$

Bayes' rule

$$\propto p(\mathbf{x}|C_k)p(C_k)$$

Uniform prior assumption

$$p(\mathbf{x}|C_k) = p(x_1, x_2, \dots, x_n|C_k)$$

$$= \prod_{i=1}^n p(x_i|C_k)$$

**Conditional independence**  
assumption

# Pattern Mining

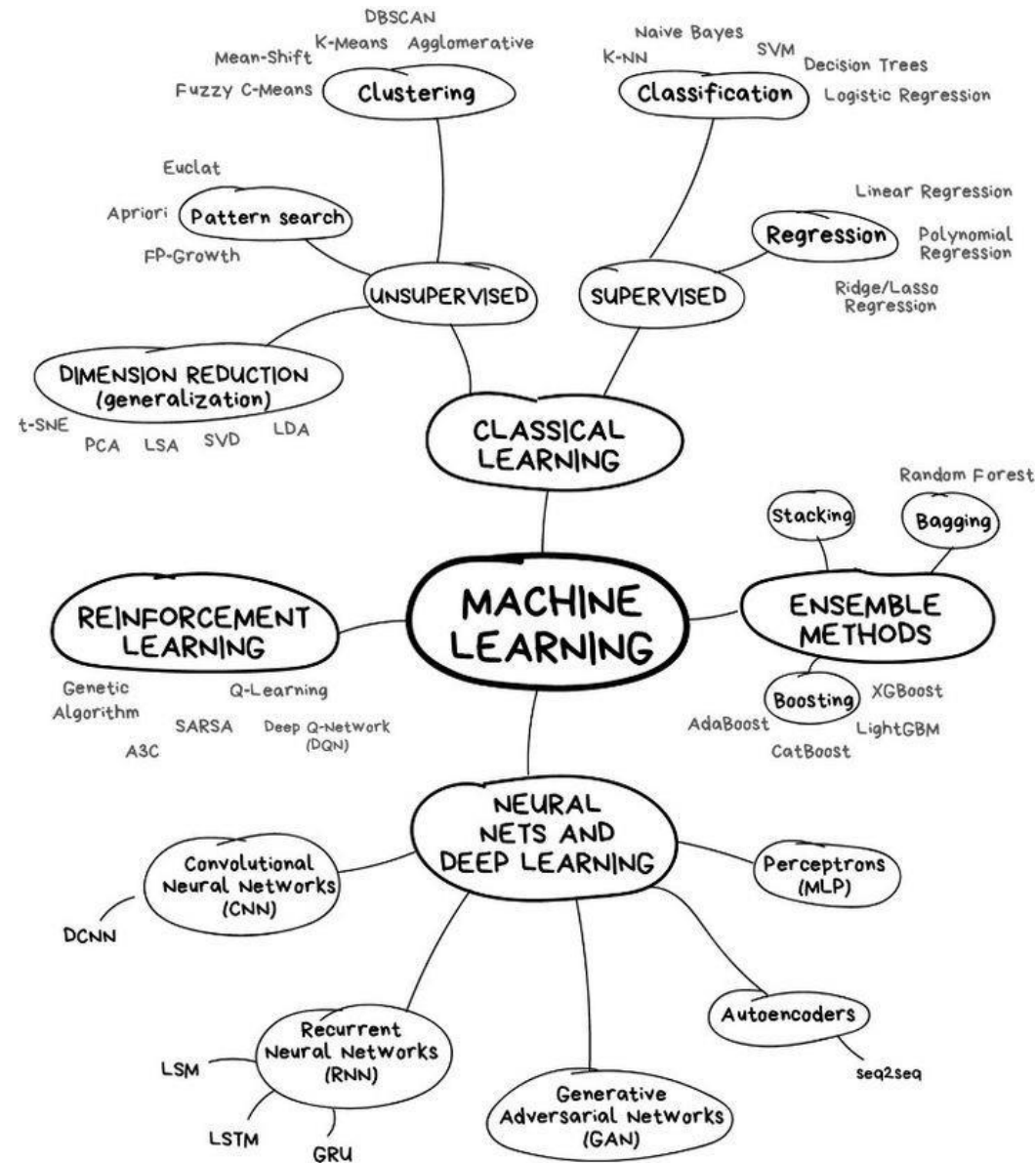


한양대학교 ERICA  
소프트웨어융합대학  
COLLEGE OF COMPUTING

인공지능학과  
Department of  
Artificial Intelligence

정 우 환 (whjung@hanyang.ac.kr)

Fall 2021



# Frequent Pattern Mining

- **Association rules**
  - **Apriori**
  - FP-tree
- Sequential patterns
  - Apriori
  - PrefixSpan

# Association Rules




- 데이터 상호간의 연관 규칙을 찾아내는 기술
- '{라면, 우유}->{커피}'
  - 라면과 우유를 산 사람은 커피도 같이 산다
  - 지지도 (support)
    - 전체 소비자 중에서 그 규칙을 구성하는 물품을 구매한 소비자의 비율
    - 50% - 4명 중 **라면**, **우유**, **커피**를 구매한 사람은 2명
  - 신뢰도 (confidence)
    - 규칙의 왼쪽에 있는 물품을 산 소비자 중에서 오른쪽에 있는 물품들을 산 소비자의 비율
    - 66.7% - **라면**과 **우유**를 산 사람들은 3명인데 그 중에서 **커피**를 산 사람은 2명

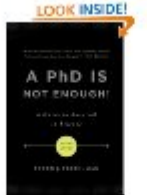
# 사용 사례

- 고객들의 물품 구매 패턴을 분석한 결과에 기반하여
  - 연관 물품 쿠폰이나 할인 행사 제공
  - 온라인 서점에서 다른 구매자들이 구매한 책 정보를 함께 제공

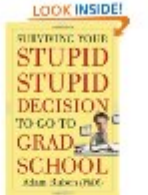
## Customers Who Bought This Item Also Bought




Graduate Admissions Essays, Fourth ...  
Donald Asher  
★★★★☆ (9)  
Paperback  
\$14.95



A PhD Is Not Enough!: A Guide to Survival in ...  
> Peter J. Feibelman  
★★★★☆ (58)  
Paperback  
\$10.17



Surviving Your Stupid, Stupid Decision to Go ...  
> Adam Ruben  
★★★★☆ (42)  
Paperback  
\$9.60



Get Into Graduate School  
Kaplan  
★★★★★ (2)  
Paperback  
\$13.33

<예: 아마존(amazon.com)의 상품 추천>

# 사용 사례

- 고객들의 물품 구매 패턴을 분석한 결과에 기반하여
  - Cross-selling (교차 판매) – 서로 다른 카테고리 상품을 추천하여 판매
    - 자동차 보험과 생명보험을 함께 판매하는 온라인 보험회사에서 10 억짜리 생명보험을 가입한 사람에게 자동차보험 대물배상 3억원 추천
    - 11번가에서 디지털카메라를 구매할 때에 명품지갑 추천
  - 백화점의 Package 구매 상품 조합 결정, 물건 진열 순서 결정 등
  - 효과적인 상품 카탈로그 디자인



# Association Rule Problem

- Given:
  - A database of customer transactions
  - Each transaction is a set of items
- Find all rules  $X \rightarrow Y$  that correlate the presence of one set of items  $X$  with another set of items  $Y$ 
  - e.g.) 98% of people who purchase diapers and baby food also buy beers.
  - Any number of items in  $X$  and  $Y$  of a rule
  - Possible to specify constraints on rules (e.g., find only rules involving expensive imported products)



# Support and Confidence

- $X \rightarrow Y$  [support, confidence]

$$\text{지지도(support)} = \frac{\text{\# of transactions containing all the items in } X \cup Y}{\text{total \# of transactions in the database}}$$

$$\text{신뢰도(confidence)} = \frac{\text{\# of transactions that contain both } X \text{ and } Y}{\text{\# of transactions containing } X}$$

- For minimum support (최소 지지도) = 50%, minimum confidence (최소 신뢰도) = 50%
  - B  $\Rightarrow$  C with 50% support and 66% confidence

TID	Items
10	a, c, d
20	b, c, e
30	a, b, c, e
40	b, e

# Association Rule Problem

- 문제를 풀기 위하여 2 개의 스텝으로 나누어 처리함
  - 스텝 1: Find all (frequent) itemsets that have minimum support
    - Most expensive phase
    - Lots of research
  - 스텝 2: Use the frequent itemsets to generate the desired rules
    - Generation is straight forward

# Association Rules 찾는 방법

TID	Items
10	a, c, d, f
20	b, c, e
30	a, b, c, e,
40	b, e
50	a, f

최소지지도=40%  
최소신뢰도 = 100%

## • 스텝 1

- 최소지지도 를 만족하는 frequent itemset들을 모두 찾음

Itemset	Sup	Itemset	Sup
a	3	a,c	2
b	3	a,f	2
c	3	b,c	2
e	3	b,e	3
f	2	c,e	2

Itemset	Sup
b,c,e	2

## • 스텝 2

- 모든 frequent itemset 으로부터 룰 생성
- {b,c,e} 에서 아래 룰들을 다 만든 후에 신뢰도를 체크함
  - {b}->{c,e} (X)
  - {c}->{b,e}
  - {e}->{b,c}
  - {b,c}->{e} (O)
  - {b,e}->{c}
  - {c,e}->{b} (O)

# Naïve Counting of All Itemsets

Itemsets & Counts

Transactions

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



Itemset	Count
A	1
C	1
D	1
A,C	1
A,D	1
C,D	1
A,C,D	1

# Naïve Counting of All Itemsets

Itemsets & Counts

Transactions

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



Itemset	Count
A	1
C	2
D	1
A,C	1
A,D	1
C,D	1
A,C,D	1
B	1
E	1
B,C	1
B,E	1
C,E	1
B,C,E	1

# Naïve Counting of All Itemsets

Transactions

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



Itemsets & Counts

Itemset	Count	Itemset	Count
A	2	A,B	1
C	3	A,E	1
D	1	A,B,C	1
A,C	2	A,B,E	1
A,D	1	A,B,C,E	1
C,D	1		
A,C,D	1		
B	2		
E	2		
B,C	2		
B,E	2		
C,E	2		
B,C,E	2		

# Naïve Counting of All Itemsets

Transactions

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




Itemsets & Counts

Itemset	Count	Itemset	Count
A	2	A,B	1
C	3	A,E	1
D	1	A,B,C	1
A,C	2	A,B,E	1
A,D	1	A,B,C,E	1
C,D	1		
A,C,D	1		
B	3		
E	3		
B,C	2		
B,E	3		
C,E	2		
B,C,E	2		

# Naïve Counting of All Itemsets

Transactions		Frequent itemsets	
TID	Items	Itemset	Count
10	A,C,D	A	2
20	B,C,E	C	3
30	A,B,C,E	D	1
40	B,E	A,C	2
		A,D	1
		C,D	1
		A,C,D	1
		B	3
		E	3
		B,C	2
		B,E	3
		C,E	2
		B,C,E	2
		A,B	1
		A,E	1
		A,B,C	1
		A,B,E	1
		A,B,C,E	1

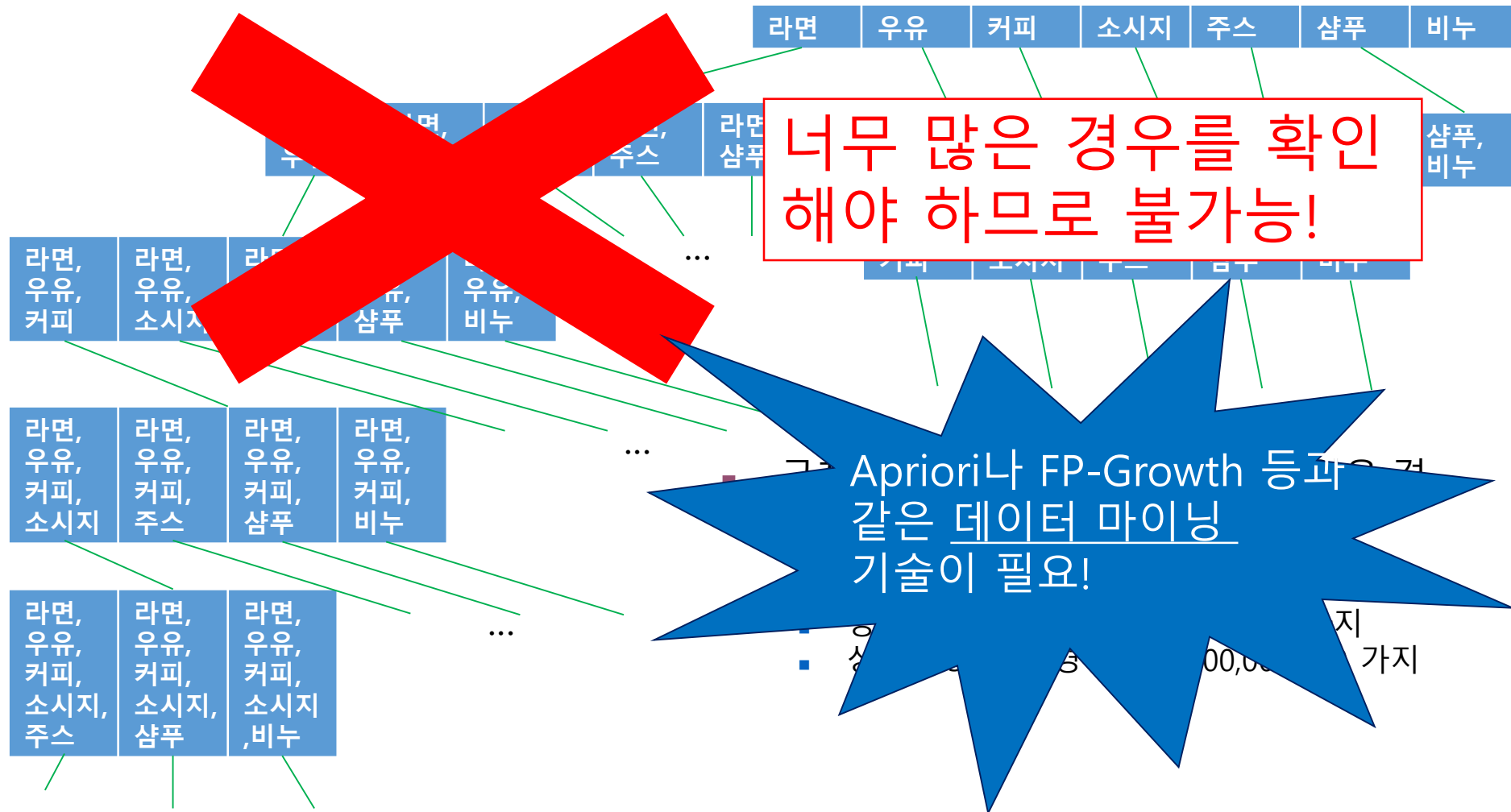
Sup<sub>min</sub>=2



We may need  $2^n$  itemset entries for counts !



# Checking All Combinations?



# Can we do better?

- 모든 상품들의 부분집합을 다 count하면 exponential 한 개수의 부분집합을 count 하게 됨
- 모든 부분집합을 count 안 하는 방법이 있을까?
- Key Observation
  - Every subset of a frequent item set is also frequent item set.
  - If {beer, diaper, nuts} is frequent, {beer, diaper} must be frequent.
- If there is any item set which is infrequent, its superset will not be generated!
  - A powerful candidate set pruning technique.

# Apriori: A Candidate Generation-and-Test Approach

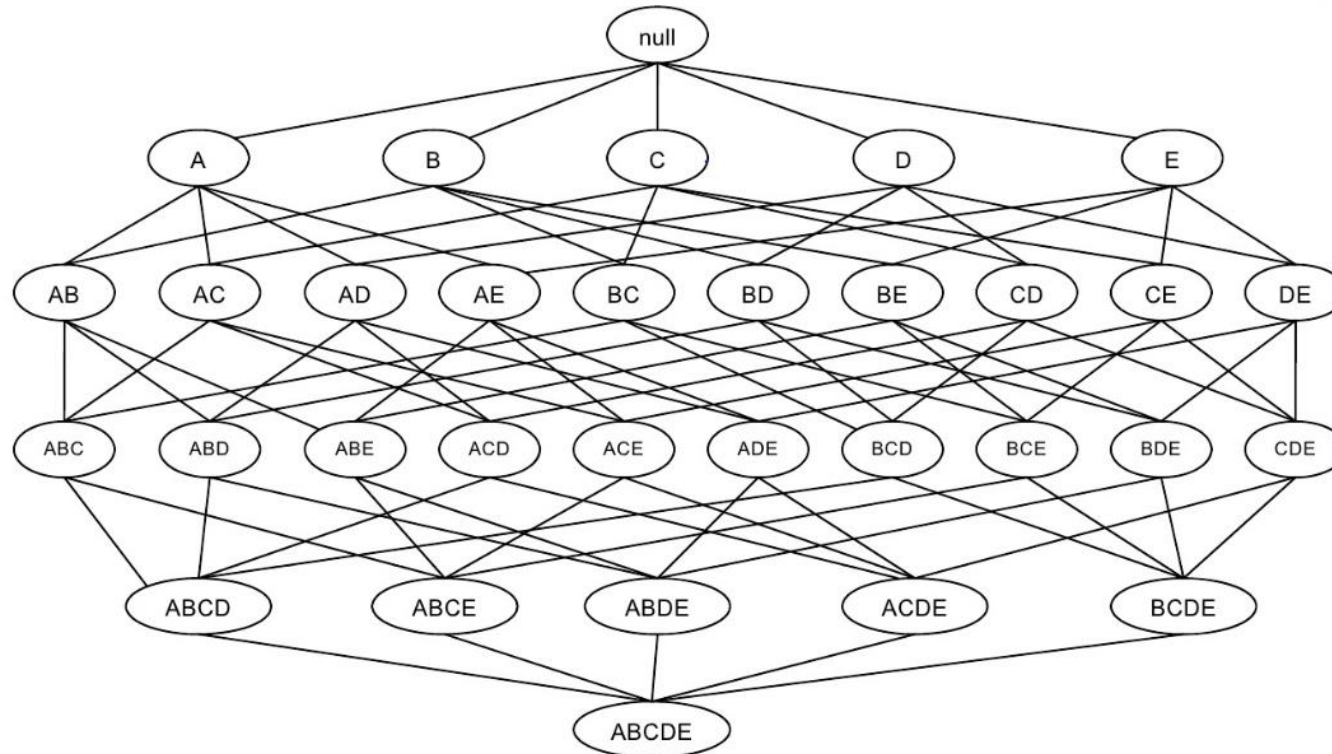
- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length  $(k+1)$  candidate itemsets from length  $k$  frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

# Scalable Methods for Mining Frequent Patterns

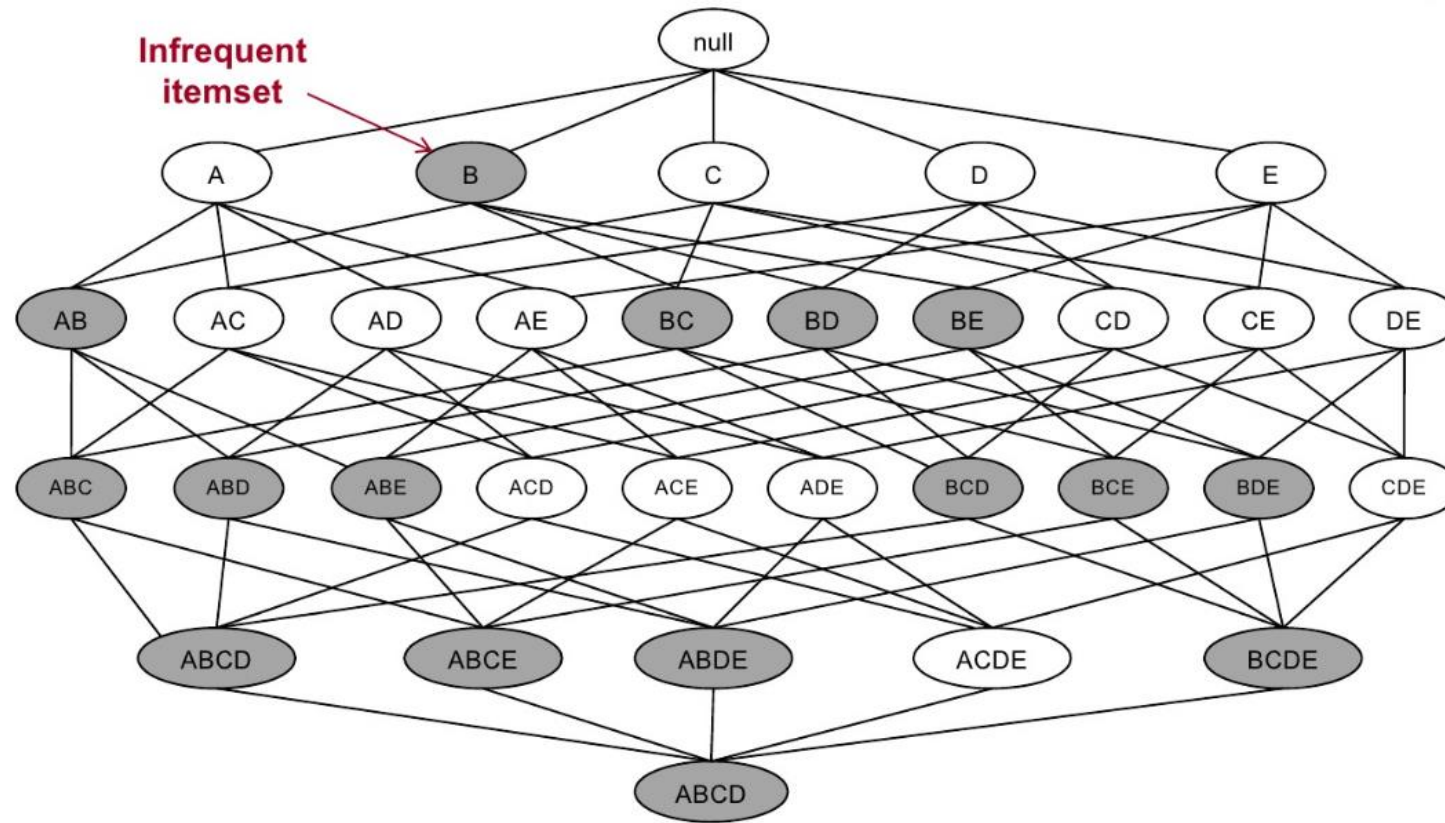
- The downward closure property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Scalable mining methods: Three major approaches
  - Apriori (Agrawal & Srikant@VLDB'94)
  - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
  - Vertical data format approach (Charm—Zaki & Hsiao @SDM'02)

# Naïve Counting

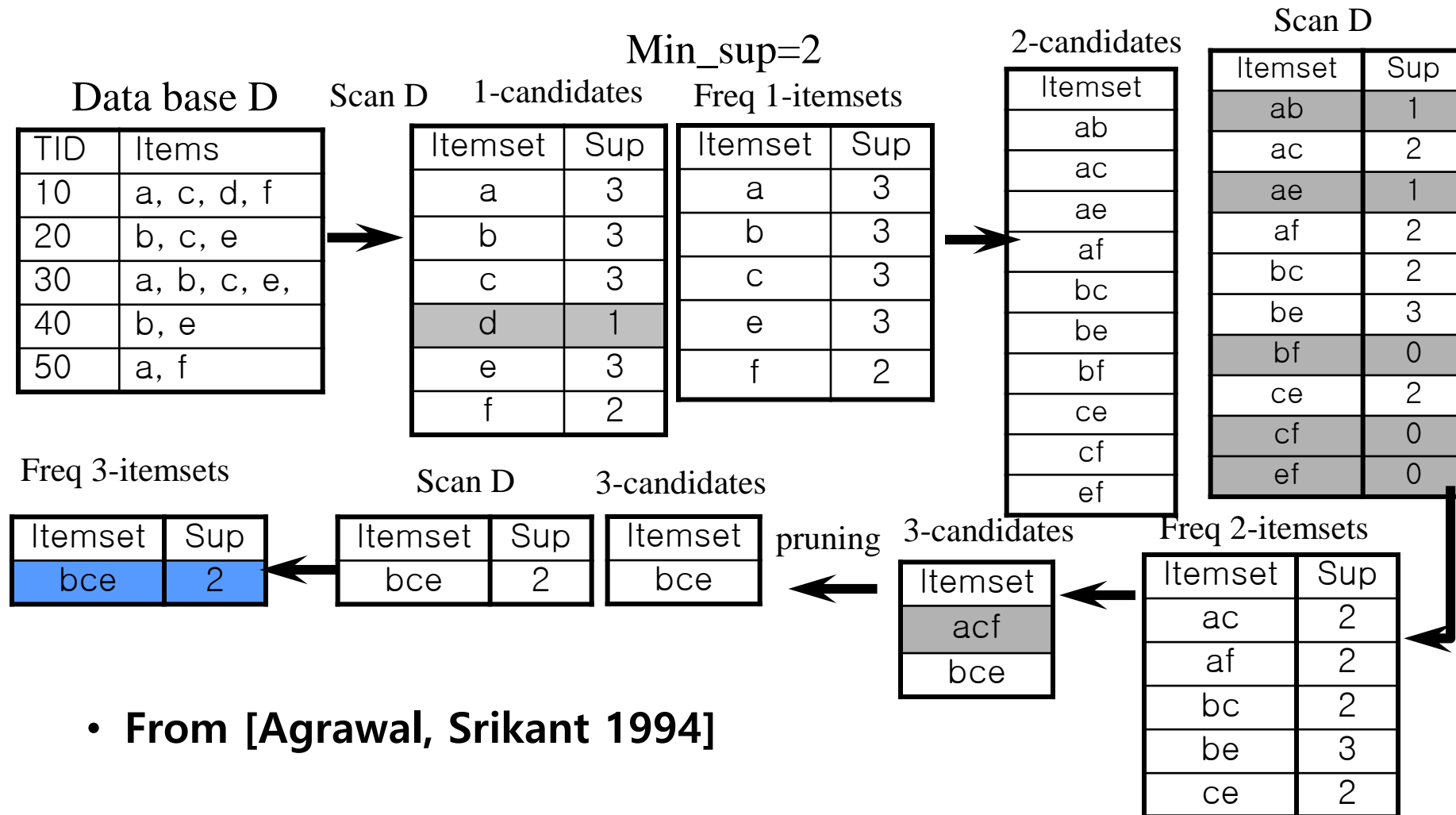
- Given  $d$  items, there are  $2^d$  itemsets



# Candidate Itemset Generation by Apriori



# An Apriori Example



- From [Agrawal, Srikant 1994]

# The Apriori Algorithm

Procedure Apriori(DB)

// $C_k$ : Candidate itemset of size  $k$

// $F_k$  : frequent itemset of size  $k$

$F_1 = \{\text{frequent items}\};$

for ( $k = 1; F_k \neq \emptyset; k++$ ) do

$C_{k+1} = \text{candidate-generate}(F_k);$

    for each transaction  $t \in \text{DB}$

        for each transaction  $c \in C_{k+1}$

            if  $c$  is contained in  $t$

$c.\text{count}++;$

$F_{k+1} = \{c \in C_{k+1} \mid c.\text{count}/|\text{DB}| \geq \text{min\_support}\};$

return  $\cup_k F_k;$



# The Apriori Algorithm

Function candidate-generate( $F_k$ )

// $C_k$ : Candidate itemset of size  $k$

// $F_k$ : frequent itemset of size  $k$

$C_{k+1} = \emptyset$ ;

for all  $f_1, f_2 \in F_k$  s.t.  $f_1 = \{i_1, i_2, \dots, i_{k-1}, i_k\}$ ,  $f_2 = \{i_1, i_2, \dots, i_{k-1}, i'_k\}$  and  $i_k < i'_k$  do

$c = \{i_1, i_2, \dots, i_{k-1}, i_k, i'_k\}$

$C_{k+1} = C_{k+1} \cup \{c\}$

    for each  $k$ -sized subset  $s$  of  $c$

        if  $s$  is not contained in  $F_k$

            delete  $c$  from  $C_{k+1}$

            break;

return  $C_{k+1}$ ;

# Discovering Rules

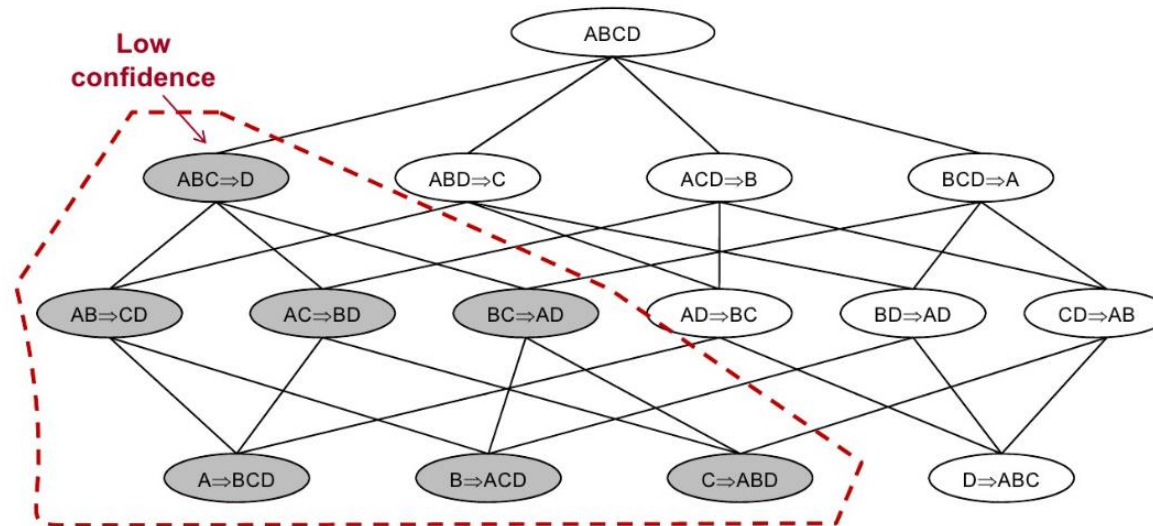
Naïve Algorithm:

```
for each frequent itemset f do  
  for each subset c of f do  
    if (support(f)/support(f-c)  $\geq$  minconf) then  
      output the rule (f-c)  $\rightarrow$  c,  
        with confidence = support(f)/support(f-c)  
        and support = support(f)
```

- Given a frequent itemset f
  - Find all non-empty subsets c in f
    - s.t. the rule (f-c)  $\rightarrow$  c satisfies the minimum support
  - Output the rule (f-c)  $\rightarrow$  c
- Let f = {A,B,C}
  - Candidate itemsets are {A}, {B}, {C}, {A,B}, {A,C}, {B,C}
  - {B,C}  $\rightarrow$  {A}, {A,C}  $\rightarrow$  {B}, {A,B}  $\rightarrow$  {C}, {C}  $\rightarrow$  {A,B}, {B}  $\rightarrow$  {A,C}, {A}  $\rightarrow$  {B,C}

# Can we do better?

- The confidence of rules generated from the same itemset have the anti-monotonicity property
- Let  $f = \{A, B, C, D\}$ 
  - $\text{Confidence}(\{A, B, C\} \rightarrow \{D\}) \geq \text{Confidence}(\{A, B\} \rightarrow \{C, D\})$
  - $\geq \text{Confidence}(\{A\} \rightarrow \{B, C, D\})$



# Discovering Rules

- Consider the rule  $(f-c) \rightarrow c$
- Now, if  $c_1$  is a subset of  $c$ 
  - $f-c_1$  is a superset of  $C$ 
    - $\text{support}(f-c_1) \leq \text{support}(f-c)$
    - $\text{support}(f)/\text{support}(f-c_1) \geq \text{support}(f)/\text{support}(f-c)$
    - $\text{conf}((f-c_1) \rightarrow c_1) \geq \text{conf}((f-c) \rightarrow c)$
- So, if a consequent  $c$  generates a valid rule, so do all subsets of  $c$
- Can use the apriori candidate generation algorithm to limit number of possible rules tested.
- Consider a frequent itemset ABCDE
  - If  $ACDE \rightarrow B$  and  $ABCE \rightarrow D$  are the only one-consequent rules with minimum confidence, then  $ACE \rightarrow BD$  is the only other rule that needs to be tested.