CS699
Lecture 8
Association Rule Mining

What Is Frequent Pattern Analysis?

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Why Is Freq. Pattern Mining Important?

- Freq. pattern: An intrinsic and important property of datasets
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: discriminative, frequent pattern analysis
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient
  - Broad applications

Association Rule Mining

- Two step process
- First, mine all frequent patterns
  A frequent pattern is also called a frequent itemset or a large itemset
- Second, mine strong rules from frequent itemsets

Basic Concepts: Frequent Itemsets

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Beer, Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- itemset: A set of one or more items
- k-itemset: a set of *k* items
- 1-itemset: {beer}, {nuts}, {diaper}, {coffee}, …
- 2-itemset: {beer, nuts}, {beer, coffee}, {eggs, milk}, …
- 3-itemset: {beer, nuts, diaper}, {nuts, coffee, eggs}, …

Basic Concepts: Frequent Itemsets

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Beer, Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- (*absolute*) *support*, or, *support count* of X: Frequency or the number of transactions that contain itemset X
- (*relative*) *support*, *s*, is the fraction of transactions that contain X (i.e., the probability that a transaction contains X)
- When we say "support" it could mean either. So, interpret it in the context.
- Support of {beer}: 4 (count), 0.8 (4 out of 5), or 80%
- Support of {coffee, diaper}: 2, 0.4 (2 out of 5), or 40%

Basic Concepts: Frequent Itemsets

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Beer, Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- An itemset X is *frequent* if X's support is no less than a predefined minimum support threshold, *minsup*.
- If *minsup* = 0.6 or 60%
  - {beer} is frequent, or is a frequent itemset, or is a large itemset, or is a frequent pattern
  - {coffee, diaper} is not frequent, or is not a frequent itemset/pattern

Basic Concepts: Association Rules

- A rule R1 = $X \rightarrow Y$

- Support of R1:

  s(R1) = *support*($X \cup Y$),

  or probability that a transaction contains $X \cup Y$

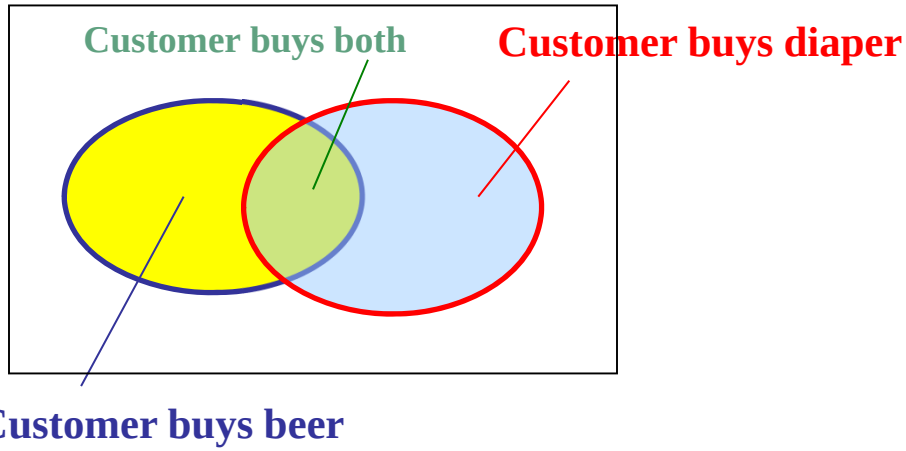
- Confidence of R1:

  c(R1) = *support*($X \cup Y$) / *support*(X),

  or conditional probability that a transaction having X also contains Y

Basic Concepts: Association Rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Beer, Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- R1 = {diaper} → {beer}

  s(R1) = support({diaper, beer}) = 3 (count), 0.6 or 60%

  3 transactions (or 60% of all transactions) contain both diaper and beer.

  c(R1) = support({diaper, beer}) / support({diaper})

  = 3 / 4 = 0.75 or 75%

  Among those who purchased diaper, 70% of them also purchased beer.

Basic Concepts: Association Rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Beer, Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- Find all rules $X \rightarrow Y$ with support ≥ minimum support and confidence ≥ minimum confidence. They are called *strong* rules.

- Let *minsup* = 40%, *minconf* = 60%
- Then,
  Beer → *Eggs* (support = 40%, confidence = 50%), is not a strong rule
  Eggs → *Beer* (support = 40%, confidence = 67%), is a strong rule

The Downward Closure Property and Scalable Mining Methods

- Apriori property of frequent itemsets
  - Any nonempty 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)

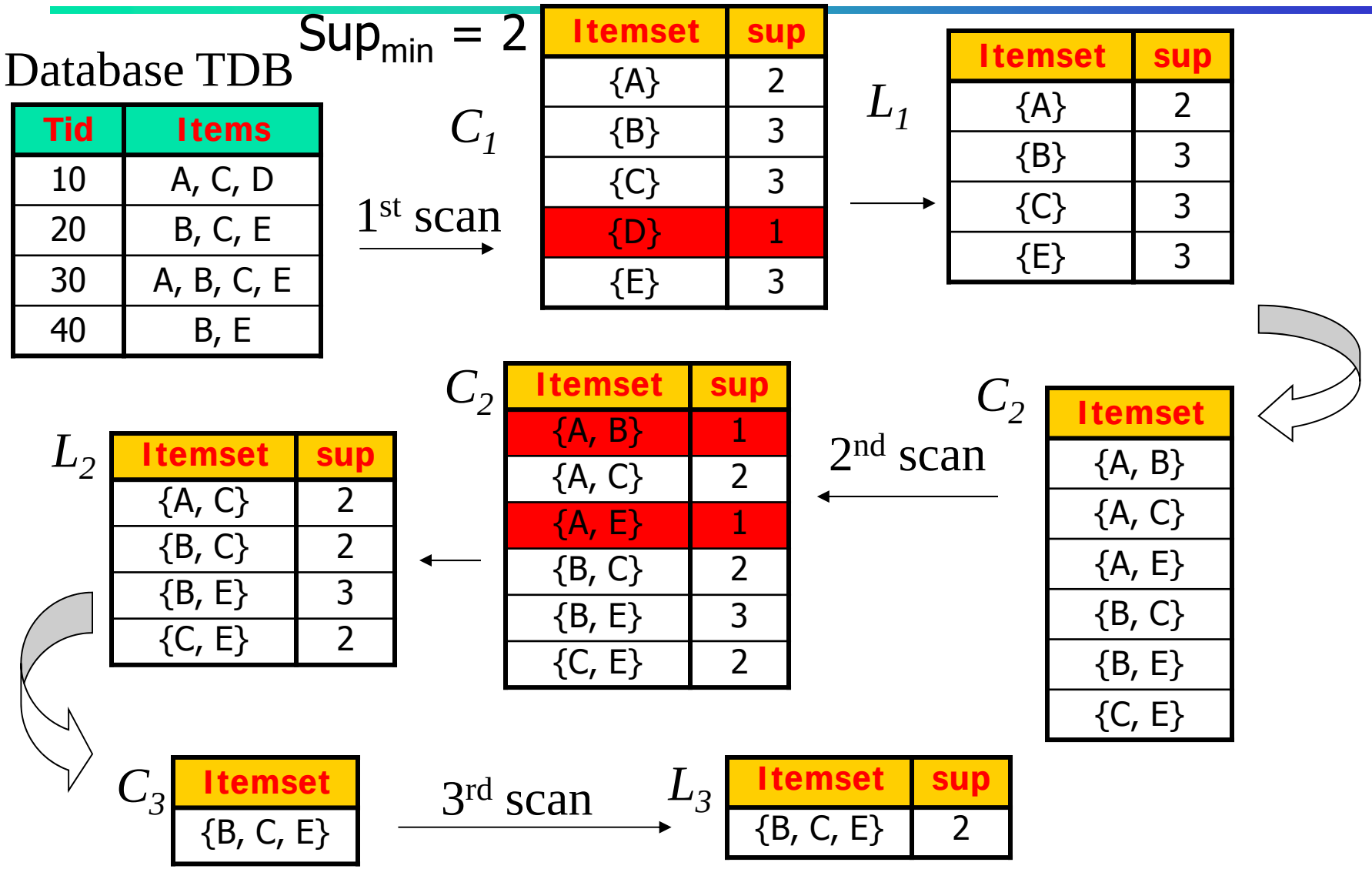
Apriori: A Candidate Generation & Test Approach

- Pruning using Apriori property: If an itemset has a subset which is infrequent, then the itemset should not be tested!

  ("test" means: to determine whether an itemset is frequent or not)

- Algorithm (simplified):
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Prune candidate itemsets using Apriori property
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated
- For simplicity, we will not discuss how to prune candidate itemsets.

The Apriori Algorithm - An Example



The Apriori Algorithm (outline)

1. Scan DB and find candidate 1-itemsets $\rightarrow C_1$
2. Mine frequent 1-itemsets from $C_1 \rightarrow L_1$
3. Generate candidate 2-itemsets from $L_1 \rightarrow C_2$ (w/o count)
4. Scan DB and count $\rightarrow C_2$ (with count)
5. Mine frequent 2-itemsets from $C_2 \rightarrow L_2$
6. k = 3
7. Generate candidate k-itemsets from $L_{k-1} \rightarrow C_k$ (w/o count)
8. Scan DB and count $\rightarrow C_k$ (with count)
9. Mine frequent k-itemsets from $C_k \rightarrow L_k$
10. k = k + 1, and Go To Step 7

***. Stop when C_k is empty or L_k is empty

Implementation of Apriori

- How to generate C_{k+1} from L_k
  - Join two k-itemsets to generate (k+1)-itmesets
  - When joining two k-itemsets, we join only if the first k-1 items are identical.
- Example: Generate C_4 from L_3.
  - L_3={*abc, abd, acd, ace, bcd*}
    - Generate *abcd* from *abc* and *abd*
    - Generate *acde* from *acd* and *ace*
  - C_4 = {*abcd*}

When joining two 3-itemsets, we join only if the first 2 items are identical.

Another Example

Dataset (min. support = 30% or three transactions)

Customer	Items
C1	beer, bread, chip, egg
C2	beer, bread, chip, egg, popcorn,
C3	bread, chip, egg
C4	beer, bread, chip, egg, milk, popcorn
C5	beer, bread, milk
C6	beer, bread, egg
C7	bread, chip, milk
C8	bread, butter, chip, egg, milk
C9	butter, chip, egg

- Items are sorted.
- Pruning is not shown in the subsequent steps.

Candidate 1-itemsets

C_1

Itemset	Support count
{beer}	5
{bread}	8
{butter}	2
{chip}	7
{egg}	7
{milk}	4
{popcorn}	2

Frequent 1-itemsets

L_1

Itemset	Support count
{beer}	5
{bread}	8
{chip}	7
{egg}	7
{milk}	4

Candidate 2-itemsets

C_2

Itemset
{beer, bread}
{beer, chip}
{beer, egg}
{beer, milk}
{bread, chip}
{bread, egg}
{bread, milk}
{chip, egg}
{chip, milk}
{egg, milk}

Candidate 2-itemsets with Counts

C_2 (scan the database and count the supports)

Itemset	Support count
{beer, bread}	5
{beer, chip}	3
{beer, egg}	4
{beer, milk}	2
{bread, chip}	6
{bread, egg}	6
{bread, milk}	4
{chip, egg}	6
{chip, milk}	3
{egg, milk}	2

Frequent 2-itemsets

L_2

Itemset	Support count
{beer, bread}	5
{beer, chip}	3
{beer, egg}	4
{bread, chip}	6
{bread, egg}	6
{bread, milk}	4
{chip, egg}	6
{chip, milk}	3

Candidate 3-itemsets

C_3

Itemset
{beer, bread, chip}
{beer. bread, egg}
{beer, chip, egg}
{bread, chip, egg}
{bread, chip, milk}
{bread, egg, milk}
{chip, egg, milk}

Two frequent 2-itemsets are joined only if the first items are identical.

We join {beer, bread} and {beer, chip} to generate {beer, bread, chip}.

But, we do not join {beer, chip} and {chip, egg}.

Candidate 3-itemsets with Supports

C_3 (scan the database and count supports)

Itemset	Support count
{beer, bread, chip}	3
{beer. bread, egg}	4
{beer, chip, egg}	3
{bread, chip, egg}	5
{bread, chip, milk}	3
{bread, egg, milk}	2
{chip, egg, milk}	2

Frequent 3-itemsets

L_3

Itemset	Support count
{beer, bread, chip}	3
{beer. bread, egg}	4
{beer, chip, egg}	3
{bread, chip, egg}	5
{bread, chip, milk}	3

Candidate 4-itemsets

C_4

Itemsets
{beer, bread, chip, egg}
{bread, chip, egg, milk}

Again, we join two frequent 3-itemsets only if the first *two* items are identical.

{beer, bread, chip} JOIN {beer, bread, egg} generates {beer, bread, chip, egg}
{bread, chip, egg} JOIN {bread, chip, milk} generates {bread, chip, egg, milk}

Candidate 4-itemsets after Pruning

C_4 (scan the database and count the supports)

Itemsets	Support count
{beer, bread, chip, egg}	3

.

Frequent 4-itemsets

L_4

Itemsets	Support count
{beer, bread, chip, egg}	3

All Frequent Itemsets

Frequent itemsets $L = L_1 \cup L_2 \cup L_3 \cup L_4$

L = {{beer}, {bread}, {chip}, {egg}, {milk}, {beer, bread},
{beer, chip}, {beer, egg}, {bread, chip}, {bread, egg},
{bread, milk}, {chip, egg}, {chip, milk}, {beer, bread, chip},
{beer. bread, egg}, {beer, chip, egg}, {bread, chip, egg},
{bread, chip, milk}, {beer, bread, chip, egg}}

Mining Strong Rules

- For each frequent itemset, identify all nonempty proper subsets:
- Example: from {beer, bread, egg}
- All nonempty proper subsets are:

  {beer}, {bread}, {egg}, {beer, bread}, {beer, egg}, {bread, egg}
- For each subset, we form a rule:

  R1: {beer} $\Rightarrow$ {bread, egg}

  R2: {bread} $\Rightarrow$ {beer, egg}

  R3: {egg} $\Rightarrow$ {beer, bread}

  R4: {beer, bread} $\Rightarrow$ {egg}

  R5: {beer, egg} $\Rightarrow$ {bread}

  R6: {bread, egg} $\Rightarrow$ {beer}

Mining Strong Rules

- Compute the confidences:

confidence = sup(all items) / sup(antecedent)

conf(R1) = (sup({ beer, bread, egg})) / sup({beer}) = 4/5 = 80%

conf(R2) = (sup({ beer, bread, egg})) / sup({bread})= 4/8 = 50%

conf(R3) = (sup({ beer, bread, egg})) / sup({egg}) = 4/7 = 57.1%

conf(R4) = (sup({ beer, bread, egg})) / sup({beer, bread}) = 4/5 = 80%

conf(R5) = (sup({ beer, bread, egg})) / sup({beer, egg}) = 4/4 = 100%

conf(R6) = (sup({ beer, bread, egg})) / sup({bread, egg}) = 4/6 = 66.7%

Mining Strong Rules

- Choose the rules whose confidences satisfy minimum confidence.

- If *min_conf* = 80%, R1, R4, and R5 are strong rules.
- If *min_conf* = 60%, R1, R4, R5, and R6 are strong rules.

Exercise

- Mine all frequent itemsets from the following dataset. Assume that the minimum support is 30% (or 3 transactions).
- Then, mine all strong rules from the first frequent 3-itemset (when 3-itemsets are sorted by the items). Assume that the minimum confidence is 80%.

TID	Items
100	2,4,5,6
200	1,4,5,7
300	2,4,5
400	1,2,4,5,6,7
500	1,2,6
600	1,2,5,7
700	2,4,6
800	2,3,4,5,6
900	3,4,5,6

References

- Han, J., Kamber, M., Pei, J., “Data mining: concepts and techniques,” 3rd Ed., Morgan Kaufmann, 2012
- http://www.cs.illinois.edu/~hanj/bk3/