Lecture 4 – Pattern and Association Mining 1

Data Mining, Spring 2016

Anders Hartzen, andershh@itu.dk

IT UNIVERSITY OF COPENHAGEN

Some slides adapted from Hector Martinez of University of Malta and Julian Togelius of New York University (both former ITU).

Overview of today's lecture

- Follow Up from last week
- Quick Math Primer on sets
- Pattern Mining Basics
- Apriori Algorithm
- Pattern Mining Challenges
- Pattern evaluation

Follow up from last week

- Can ID3 handle missing data?
  - Original/standard version of ID3 assumes no missing data
    - Or at least assumes that placeholder value has been inserted (e.g. “unknown” or 0.0)
- However researchers have looked into adapting ID3
  - *Handling Missing Value in Decision Tree Algorithm* - http://research.ijcaonline.org/volume70/number13/pxc3888063.pdf
  - *A method of processing unknown attribute values by ID3* - http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=227661&url=http%3A%2F%2Fieeexplore.ieee.org%2Fstamp%2Fstamp.jsp%3Farnumber%3D227661

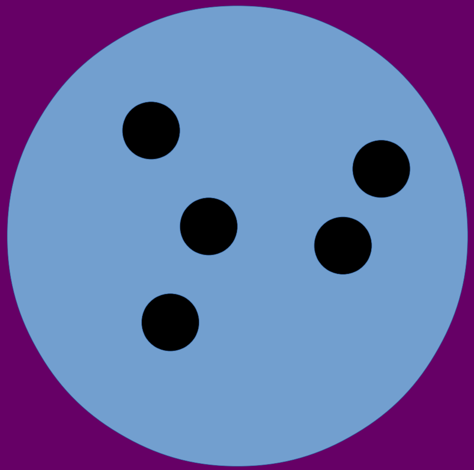
Follow up from last week

- The C4.5 algorithm (successor to ID3) can handle missing values
  - *A comparative study of decision tree ID3 and C4.5* http://saiconference.com/Downloads/SpecialIssueNo10/Paper_3-A_comparative_study_of_decision_tree_ID3_and_C4.5.pdf
  -

Math Primer on Sets

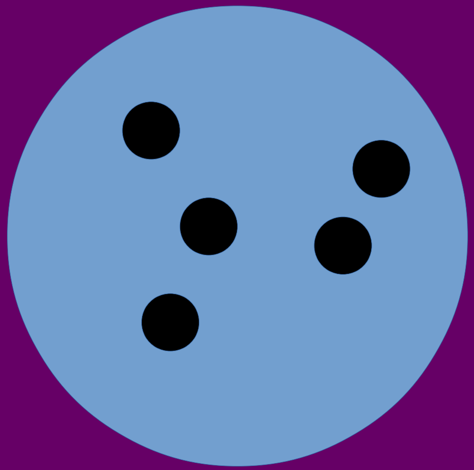
Math Primer Sets

- Set
  - Collection of distinct objects
  - Mathematician Georg Cantor: *A set is a gathering together into a whole of definite, distinct objects of our perception [Anschauung] or of our thought—which are called elements of the set.* (1895)



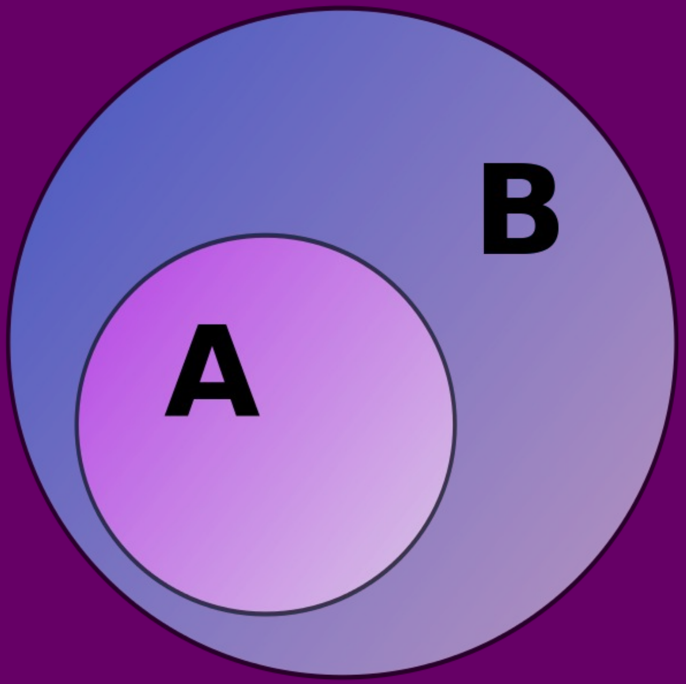
Math Primer Sets

- Set
  - Example
    - A = {1,2,7,8,12}
  - Cardinality (size)
    - |A| = 5



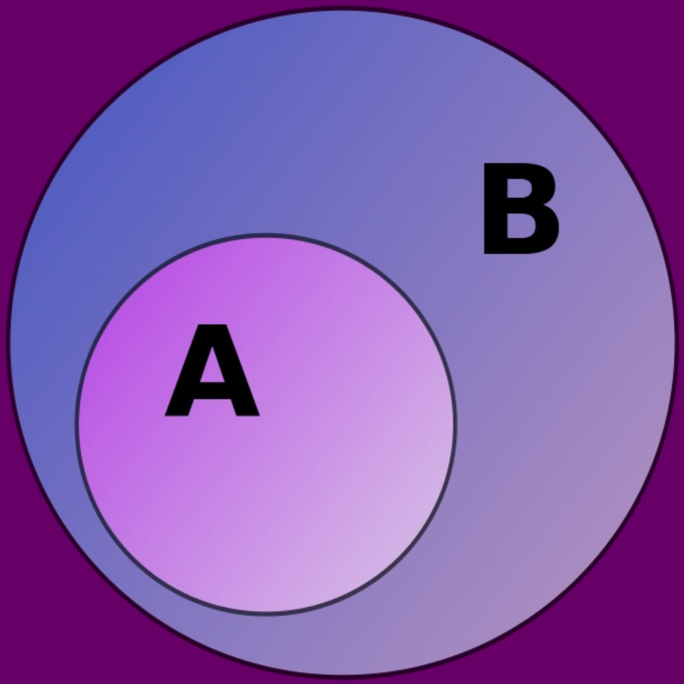
Math Primer Sets – Subset/Superset

- Subset
  - If every element of set A is also an element of set B, then A is a *subset* of B
  - Notation: $A \subseteq B$
    - “A is contained in B”
- Superset
  - If set B contains every element of set A, then B is a *superset* of A
  - Notation: $B \supseteq A$
    - “B contains A”



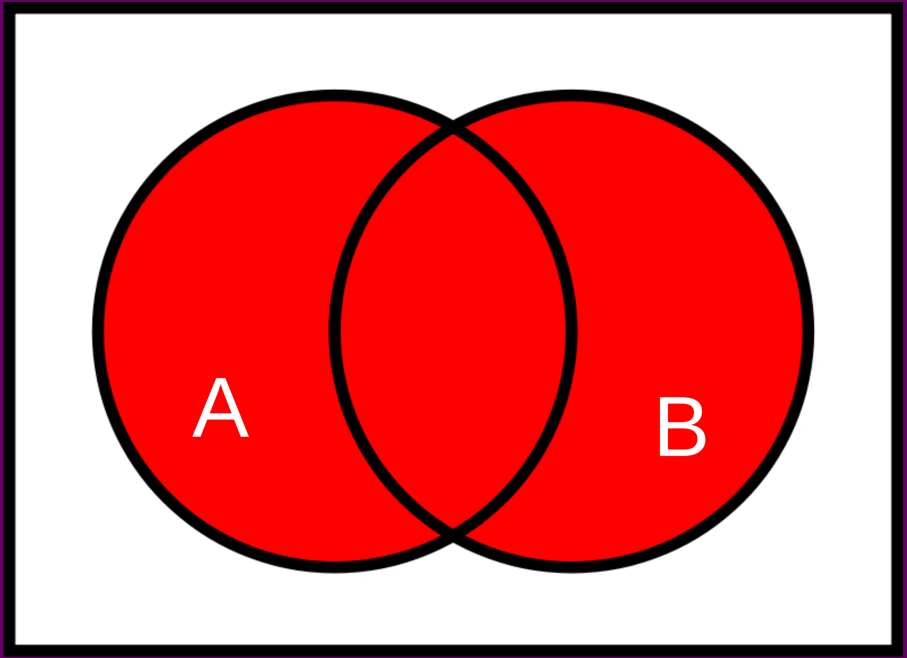
Math Primer Sets – Proper Subset/Superset

- A is a a *proper subset* of B if B contains all elements of A, but is not equal to A
  - i.e. B has other elements than set A's elements
- B is a *proper superset* if B contains all elements of A, but is not equal to A
- Notation: $A \subset B$ and $B \supset A$
  - Different notations used by different authors!



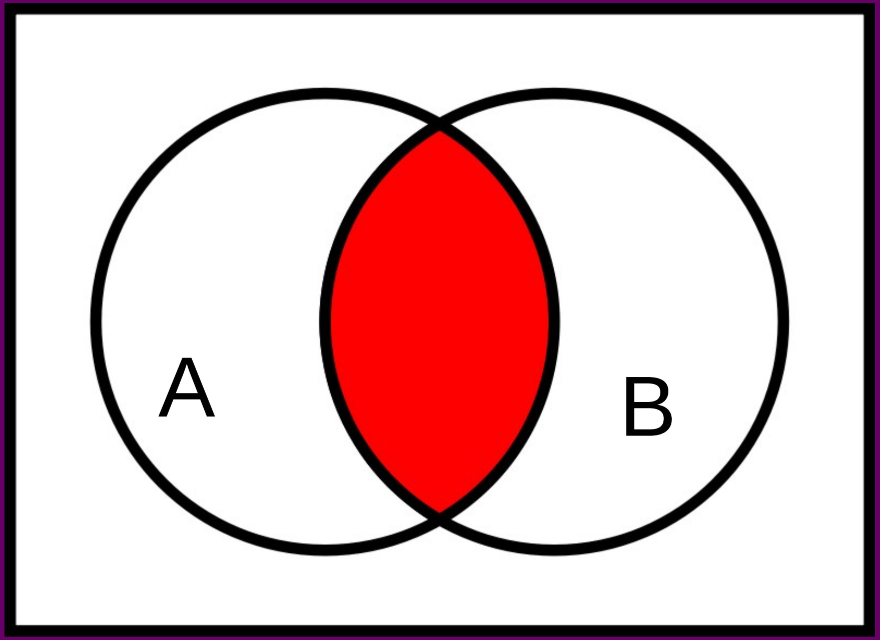
Math Primer Sets – Union

- “Adding” two sets together
- Union of set A and B is the set of elements part of A **or** B
- Notation: $A \cup B$
- Examples:
  - $\{1, 2\} \cup \{1, 2\} = \{1, 2\}$
  - $\{1, 2\} \cup \{2, 3\} = \{1, 2, 3\}$
  - $\{1, 2, 3\} \cup \{3, 4, 5\} = \{1, 2, 3, 4, 5\}$



Math Primer Sets – Intersection

- What two sets have in common
- Intersection of set A and B is the set of elements part of A **and** B
- Notation: $A \cap B$
- Examples:
  - $\{1, 2\} \cap \{1, 2\} = \{1, 2\}$
  - $\{1, 2\} \cap \{2, 3\} = \{2\}$

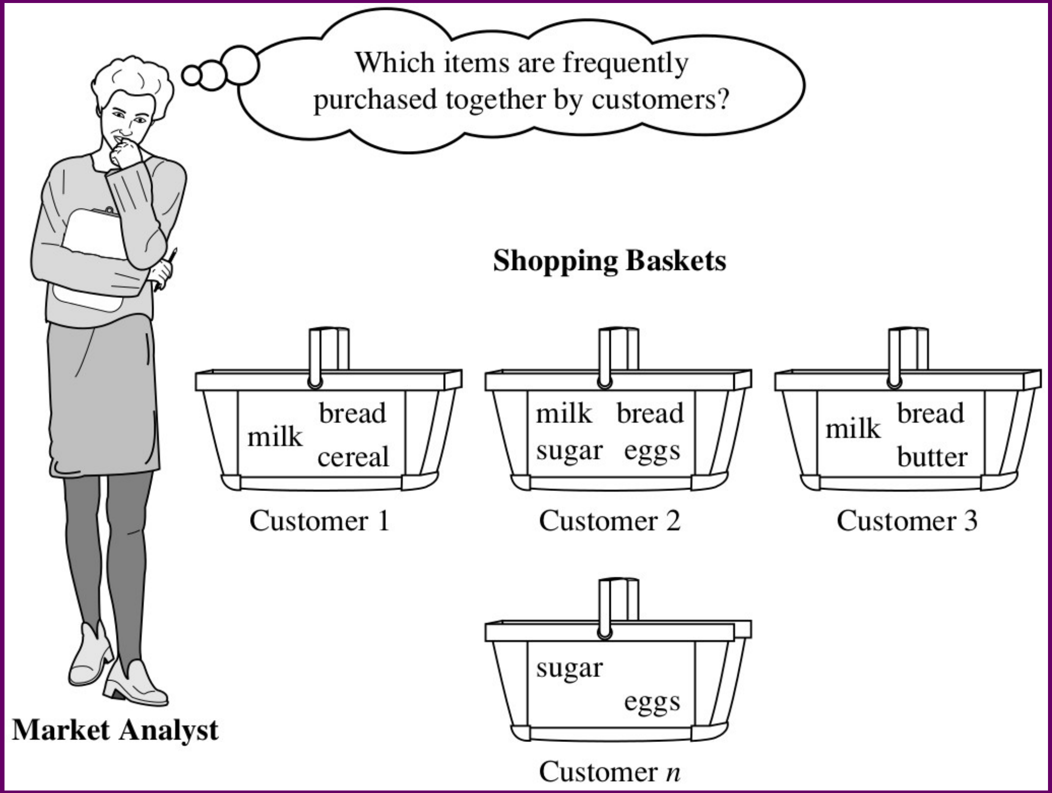


Pattern Mining Basics

Introduction

- Frequent Pattern Mining
  - Searching for recurring relationships in a data set
- Frequent Pattern
  - Patterns (e.g. set of items, subsequence) that appear frequently in a dataset
- Frequent Itemset
  - A set of items (e.g. milk and bread) that appear frequently together
  - Focus of today!
- Frequent Sequential Pattern
  - A subsequence such as first buying item a then item b that appears frequently
  - Focus of later lecture

Example: Market Basket Analysis



Example: World of Warcraft group class composition

Motivation and Applications

- Motivation: Finding inherent regularities in data
  - What products were often purchases together?
  - What types of classes usually form groups in World of Warcraft?
  - What are the subsequent purchases after buying a computer?
  - What kinds of DNA are sensitive to this new pharmaceutical drug?
- Applications
  - Market basket analysis, cross-marketing, catalog design, sale campaign analysis, web log (click stream) analysis, DNA sequence analysis

Association Rules

- Is a way representing frequent patterns
- Notation and example:

  $A \Rightarrow B$ [support = x%, confidence = y%]

  computer ⇒ antivirus software [support = 2%, confidence = 60%]
- Support and confidence two measures of how interesting the found rule is
  - Support measures the percentage of tuples in the data set which had both A and B (i.e. $A \cup B$)
    - Also known as the relative support
  - Confidence measures the percentage of tuples having A that also has B

Association Rules

- Minimum support threshold and minimum confidence threshold
  - Threshold values between 0% and 100% used to define when an association rule is interesting
  - Association rules that satisfy both are called **strong**

Itemset

- A set of items (e.g. contents of a shopping basket) is known as an itemset
- An itemset with k items is known as a *k-itemset*
  - e.g. the itemset {milk, bread, flour} is a 3-itemset
- The number of times an itemset appears in the data is known as *the occurrence frequency of an itemset*
  - Other names include: frequency, support count, count or *absolute support*

Support count, support and confidence

- Support count
  - Number of times an itemset appears in the data
  - Aka *absolute support*
- Support
  - Percentage of tuples in the data set which had both itemset A and itemset B
  - Aka *relative support*
- Confidence
  - Percentage of tuples having itemset A that also has itemset B

Support count, support and confidence

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

P(B|A) also known as the conditional probability, reads as “the probability of B under the condition A” → https://en.wikipedia.org/wiki/Conditional_probability

Support/Confidence example

- Find the support and confidence for these association rules:
  - Bread ⇒ Milk
  - Newspaper ⇒ Tomato
- Data (supermarket transaction)
  - Bread, milk, butter, newspaper
  - Bread, milk
  - Butter, newspaper, asparagus
  - Bread, milk, tortellini, batteries
  - Tortellini, asparagus, mozzarella
  - Bread, milk, butter
  - Newspaper, asparagus, cigarettes, tomato
  - Newspaper, tomato
  - Asparagus, batteries, cigarettes
  - Bread, milk

Support/Confidence example

- Find the support and confidence for these association rules:
  - Bread ⇒ Milk [support = 50 %, confidence = 100%]
  - Newspaper ⇒ Tomato [support = 20%, confidence = 50%]
- Data (supermarket transaction)
  - Bread, milk, butter, newspaper
  - Bread, milk
  - Butter, newspaper, asparagus
  - Bread, milk, tortellini, batteries
  - Tortellini, asparagus, mozzarella
  - Bread, milk, butter
  - Newspaper, asparagus, cigarettes, tomato
  - Newspaper, tomato
  - Asparagus, batteries, cigarettes
  - Bread, milk

The two-step association rule mining process

- Because of the close connection between confidence, support and support count, we can easily derive the confidence of an association rule (e.g. $A \Rightarrow B$) based on the support/ support counts
  - In other words once the support/support counts for A, B and B U A have been found we can generate the different association rules and check whether they are strong
- Therefore the association rule mining process can be viewed as a two-step process
  - Find all frequent itemsets
  - Generate strong association rules based on the found frequent itemsets

The two-step association rule mining process

- Problem: Often a huge number of itemsets are generated during mining
  - Especially if minimum support is low
- This is because if an itemset is frequent, then each of its subsets is frequent as well
- Example: Consider the frequent 100-itemset {a1, a2, a3 … a99}
  - It can be divided into many subsets like
    - {a1}, {a2}, {a3} …, {a99}
    - {a1,a2}, {a1,a3} … {a99, a100}
    - And so on
  - Total number of subsets = $2^{100} – 1 = 1.27 \times 10^{30}$ - too many to store in memory
- Solution: Only mine closed frequent itemsets and maximal frequent itemsets!

Closed/Maximal Itemset

- An itemset X is closed if there exists no proper super-itemset Y that has the same support count as X
  - i.e. there is no itemset Y where $Y \supset X$ and also has the same support count as X
  - i.e. there is no itemset Y which has all the same elements as X (and at least one other item, which is not part of X) and also has the same support count as X
- An item X is a maximal itemset if there exists no frequent super-itemset Y where $Y \supset X$
- Closed itemsets are a lossless compression of frequent patterns
  - Reduces the number of generated itemsets to consider

Closed example

- Are the following itemsets closed?
  - {Bread, Milk}
  - {Bread, Butter}

- Data (supermarket transaction)
  - Bread, milk, butter, newspaper
  - Bread, milk
  - Butter, newspaper, asparagus
  - Bread, milk, tortellini, batteries
  - Tortellini, asparagus, mozzarella
  - Bread, milk, butter
  - Newspaper, asparagus, cigarettes, tomato
  - Newspaper, tomato
  - Asparagus, batteries, cigarettes
  - Bread, milk

Closed example

- Are the following itemsets closed?
  - {Bread, Milk}
    - Yes!
  - {Bread, Butter}
    - No!

- Data (supermarket transaction)
  - Bread, milk, butter, newspaper
  - Bread, milk
  - Butter, newspaper, asparagus
  - Bread, milk, tortellini, batteries
  - Tortellini, asparagus, mozzarella
  - Bread, milk, butter
  - Newspaper, asparagus, cigarettes, tomato
  - Newspaper, tomato
  - Asparagus, batteries, cigarettes
  - Bread, milk

Basics Overview of Association Rules and Frequent Pattern Mining

- Itemset X = {X1, X2, … ,Xk}
- Find all the rules $X \rightarrow Y$ with minimum support and minimum confidence
  - Support, s, percentage of tuples that contain $X \cup Y$
  - Confidence, c, percentage of tuples having X that also contains Y
- Closed Itemset
  - If there is no proper superset of X with same support
- Maximal Itemset
  - If there is no proper frequent superset of X with same support

Apriori Algorithm

Apriori

- Utilizes the concept of closed item sets and the fact that if an itemset is frequent, then each of its subsets is frequent as well
  - **Apriori Property:** *All non-empty subsets of a frequent itemset must also be frequent*
  - If {bread, milk, newspaper} is frequent, then so are the subsets {bread, milk}, {milk, newspaper}, {bread, newspaper} etc.
- Apriori property used to reduce the number of itemsets to consider when finding frequent itemsets
- Apriori output: The frequent itemsets in data set

Apriori

- **Apriori pruning principle**: If there is **any** itemset, which is infrequent, its superset should not be generated/tested!
- Algorithm (informally)
  - Initially scan database once to find frequent 1-itemsets
  - Generate length k itemsets from length (k-1) frequent itemsets
  - Test the candidates against database to see if they are frequent
  - Terminate when no frequent or candidate set can be generated
- Candidate itemsets are generated using a two-step process
  - The join step
  - The prune step

Apriori – The Join Step

- Used to generate candidate itemsets of length k when k >= 2, denoted C_k
- Main Ingredient: List of frequent itemsets of length k-1 from previous iteration of apriori algorithm, L_{k-1}
- New candidate itemsets generated by joining L_{k-1} with itself

Apriori – The Join Step

- Suppose that the items in each itemset in L_{k-1} are ordered
- Members of L_{k-1} are only joined with each other if their first k-2 elements are the same,
  - i.e. all elements apart from the last one must be the same
- Simple example (k=4)
  - L_3 = {abc, abd, acd, ace, bcd}
  - abc and abd are joined together to form abcd
  - acd and ace are joined together to form acde

Apriori – The Prune Step

- From the join step we get a set of candidate itemsets C_k
- In the prune step we utilize the apriori property to remove candidate itemsets we already can rule out as being infrequent
  - Done to limit the amount of candidates for which to compute support for
- If a subset of any candidate itemset in C_k is infrequent, then the itemset as whole will be infrequent as well
- Check each k-1 subset against L_{k-1} – if it is not there then it is infrequent

Apriori – The Prune Step

- Simple example continued (k=4)
  - L_3 = {abc, abd, acd, ace, bcd}
  - Join step gave us two candidate itemsets
    - C1 = abcd
    - C2 = acde
  - C1's k-1 subsets: abc and bcd
    - Since both are in L_3 C1 is not pruned
  - C2's k-1 subsets: acd and cde
    - Since cde is not in L_3, C2 is pruned

Apriori – Step for Step example 0/10

Minimum support = 2

Database TDB

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

Apriori – Step for Step example 1/9

Database TDB

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

Apriori – Step for Step example 2/9

Database TDB

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

C_1

1st scan →

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

Apriori – Step for Step example 3/9

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

Apriori – Step for Step example 4/9

Database TDB

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

1st 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}

Apriori – Step for Step example 5/9

Database TDB

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

1st 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

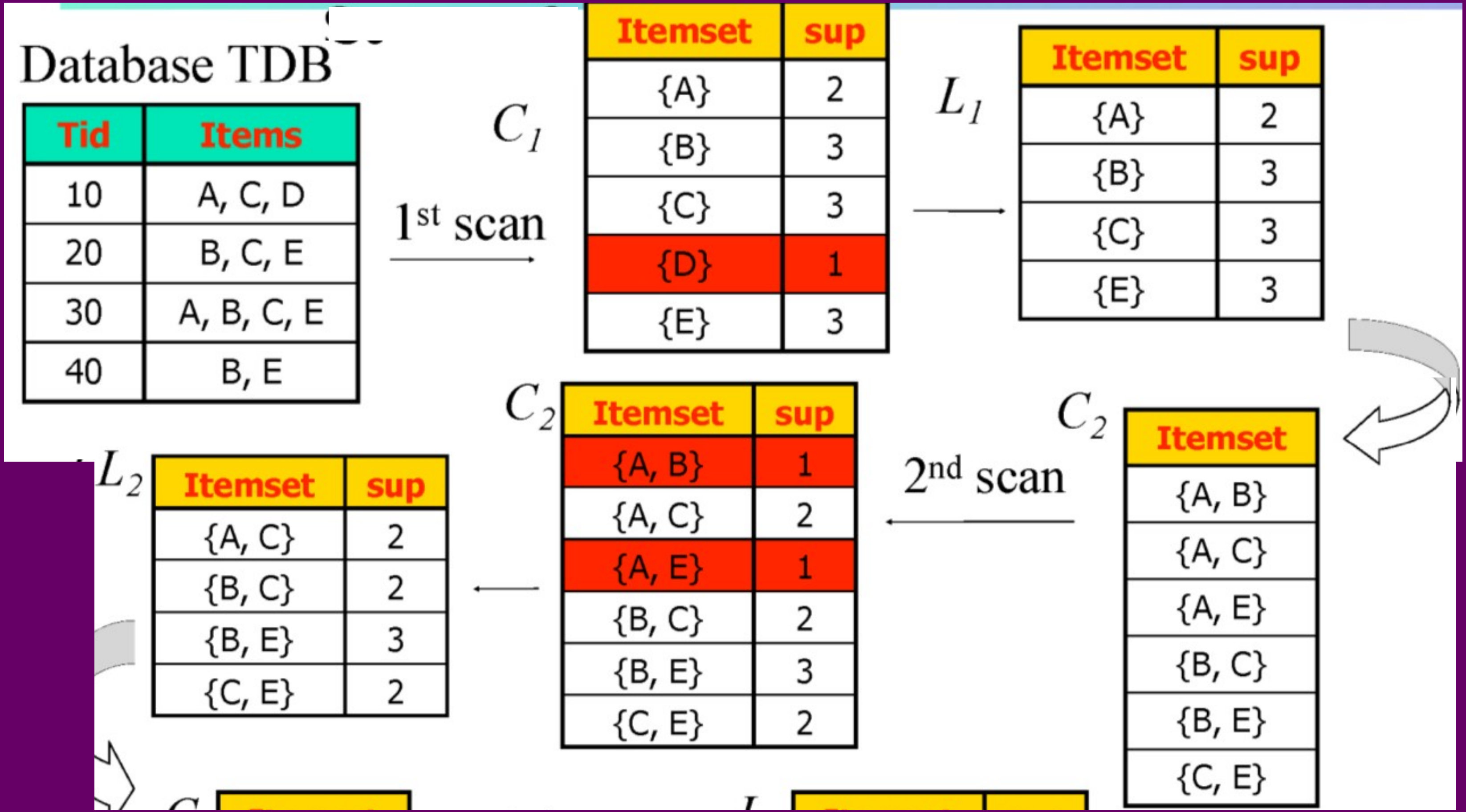
Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan

C_2

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

Apriori – Step for Step example 6/9



Database TDB

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

1st 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}

2nd 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

Apriori – Step for Step example 7/9

C_2

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

2nd 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}

Apriori – Step for Step example 8/9

C_2

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

2nd 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}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

Apriori – Step for Step example 9/9

$Sup_{min} = 2$

Database TDB

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

1st 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}

2nd 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}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

Apriori – Pseudocode

Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- min_sup, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

L_1 = find_frequent_1-itemsets(D);
for $(k = 2; L_{k-1} \neq \phi; k{+}{+})$ {
    C_k = apriori_gen(L_{k-1});
    for each transaction $t \in D$ { // scan D for counts
        C_t = subset(C_k, t); // get the subsets of t that are candidates
        for each candidate $c \in C_t$
            c.count++;
    }
    $L_k = \{c \in C_k | c.count \geq min\_sup\}$
}
return $L = \cup_k L_k$;

procedure apriori_gen(L_{k-1}:frequent $(k-1)$-itemsets)
    for each itemset $l_1 \in L_{k-1}$
        for each itemset $l_2 \in L_{k-1}$
            if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2])$
                $\wedge ... \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ **then** {
                $c = l_1 \bowtie l_2$; // join step: generate candidates
                if has_infrequent_subset(c, L_{k-1}) **then**
                    delete c; // prune step: remove unfruitful candidate
                else add c **to** C_k;
            }
    return C_k;

procedure has_infrequent_subset(c: candidate k-itemset;
        L_{k-1}: frequent $(k-1)$-itemsets); // use prior knowledge
    for each $(k-1)$-subset s **of** c
        if $s \notin L_{k-1}$ **then**
            return TRUE;
    return FALSE;

Generating Association Rules from Frequent Itemsets

- For each frequent itemset l, generate all nonempty subsets of l.
- For every nonempty subset s of l, output the rule "$s \Rightarrow (l - s)$" if $\frac{support\_count(l)}{support\_count(s)} \geq min\_conf$, where min_conf is the minimum confidence threshold.

Pattern Mining Challenges

Pattern Mining Challenges

- Multiple scans of database
  - Each is costly
- Potentially large number of candidates
- Tedious workload to count support for candidates
  - Possible bottleneck generating and testing candidates

Improving Apriori

- Reducing the size of the data that we mine for frequent patterns
  - Sampling can be used to create smaller-size dataset to represent full data set
  - Smaller sized data set is created by randomly selecting data points in original data set
  - Tradeoff between efficiency and accuracy
  - May lose some of the global frequent itemsets
- Pruning search space by utilizing closed/max itemsets (see 6.2.6)
  - Item merging, sub-itemset pruning, Item skipping

Improving Apriori

- Reducing the size of the candidate k-sets
  - Using a hash-based technique to map k-sets into buckets
- Reducing the number of transactions/tuples scanned in future iterations
  - Remove any transaction/tuple that does not have any frequent k-itemsets (won't have any k+1 itemsets)
- Reducing the number of needed database scans
  - Use a partitioning scheme to divide data set into manageable partitions.
    - One scan to make the partitions.
    - Then use partitions to find candidate itemsets (a global frequent itemset must be locally frequent in one of the partitions).
    - Second scan of database then done to find actual support for candidate sets.

Improving Apriori

- Reducing the size of the data that we mine for frequent patterns
  - Sampling can be used to create smaller-size dataset to represent full data set
  - Smaller sized data set is created by randomly selecting data points in original data set
  - Tradeoff between efficiency and accuracy
  - May lose some of the global frequent itemsets
- Pruning search space by utilizing closed/max itemsets (see 6.2.6)
  - Item merging, sub-itemset pruning, Item skipping

Vertical Data Format

- So far we have been dealing with transactional data in the TID-itemset format i.e. *{TID : Itemset}*
  - TID = transaction ID
  - Also known as the horizontal data format
- Alternatively we could present the data vertically instead, i.e. *{Itemset: TID}*
- Is the basis for the Eclat pattern mining algorithm
- Mining done by intersecting the TID_sets of every pair of frequent itemsets

Vertical Data Format

The Vertical Data Format of the Transaction Data Set *D* of Table 6.1

itemset	*TID_set*
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

→

2-Itemsets in Vertical Data Format

itemset	*TID_set*
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3, I5}	{T800}

Vertical Data Format

2-Itemsets in Vertical Data Format

itemset	*TID_set*
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3, I5}	{T800}

3-Itemsets in Vertical Data Format

itemset	*TID_set*
{I1, I2, I3}	{T800, T900}
{I1, I2, I5}	{T100, T800}

Pattern Evaluation

Pattern Evaluation

- Minimum support and confidence helps us eliminate uninteresting association rules
- But even association rules that passes our minimum support and confidence may still be uninteresting
  - Especially true if using low threshold values for support and confidence
- We therefore need objective “interesting” measures based on the statistics of the data to further help eliminate uninteresting rules
- Pattern evaluation bottleneck for successful usage of association mining

Adding Correlation into the mix

- Therefore we augment our association rules to also include a correlation measure

$$A \Rightarrow B \; [support, confidence, correlation].$$

- Correlation helps us determine if the occurrence of itemset A is independent of the occurrence of itemset B, or if A and B are in fact dependent on each other and thus correlated
- Many different correlation measures available to use

Lift

$$lift(A, B) = \frac{P(A \cup B)}{P(A)P(B)}.$$

- Equivalent to Confidence(A → B) / support(B)
- Result > 1
  - Positive correlation i.e. the occurrence of one implies the occurrence of the other
- Result = 1
  - A and B are independent of each other
- Result < 1
  - Negative correlation i.e. the occurrence of one likely leads to the absence of the other

Other Correlation Measures

- Chi-square test (Chapter 3.3.2)
- All_Confidence
- Max_Confidence
- Kulczynski
- Cosine measure

- All_Confidence, Max_Confidence, Kulczynski and the Cosine measures are only influenced by the relative relationship of support values of A, B and A U B
  - i.e. not influenced by the total number of transactions in the dataset
  - Results range from 0 to 1, the higher the value, the closer the relationship between A and B

The problem of null

- The Lift and Chi-Square measures run into problems when the data contains *null-transactions,* because its values are influenced by the total number of transactions and hence null-transactions
  - Null-transaction: A transaction that does not contain any of the itemsets undergoing correlation analysis
- Null-invariance
  - A correlation measure is said to be null-invariant if its value is free from the influence of null-transactions
  - The All_Confidence, Max_Confidence, Kulczynski and Cosine measures are all null-invariant
- Book recommends: Kulczynski used in conjunction with imbalance ratio (IR)
  - IR measures imbalance between two itemsets

Today's Lab

- Implement Apriori
- We all meet in 4A58 to see if we can fit in
  - If not, we also use 4A54

Thanks for listening!

How did I do? Send questions or feedback to andershh@itu.dk