Association Rules and Sequential Patterns

Road Map

- ❑ Frequent itemsets and rules
- ❑ Apriori algorithm
- ❑ FP-Growth
- ❑ Data formats
- ❑ Class association rules
- ❑ Sequential patterns. GSP algorithm

Objectives

- Association rule learning was introduced in the article "Mining Association Rules Between Sets of Items in Large Databases" by Agrawal, Imielinski and Swami (see references), article presented at the 1993 SIGMOD Conference (ACM SIGMOD means ACM Special Interest Group on Management of Data).
- One of the best known applications is finding relationships (rules) between products as recorded by POS systems in supermarkets.
- For example, the statement that 85% of baskets that contain bread contain also mineral water is a rule with bread as antecedent and mineral water as consequent.

Objectives

- The original article (Agrawal et al.) lists some examples of the expected results:
  - Find all rules that have “Diet Coke” as consequent,
  - Find all rules that have “bagels” in the antecedent,
  - Find all rules that have “sausage” in the antecedent and “mustard” in the consequent,
  - Find all the rules relating items located on shelves A and B in the store,
  - Find the “best” k rules (considering rule support) that have “bagels” in the consequent

Frequent itemsets and rules

Frequent itemsets and rules:

- ❑ Items and transactions
- ❑ Association rules
- ❑ Goals for mining transactions

Items and transactions

- Let I = $\{i_1, i_2, \ldots, i_n\}$ be a set of **items**. For example, items may be all products sold in a supermarket or all words contained in some documents.
- A **transaction t** is a set of items, with $t \subseteq I$. Examples of transactions are market baskets containing products or documents containing words.
- A **transaction dataset** (or database) T is a set of transactions, $T = \{t_1, t_2, \ldots, t_m\}$. Each transaction may contain a different number of items and the dataset may be stored in a SGBD managed database or in a text file.
- An **itemset** S is a subset of I. If $v = |S|$ is the number of items in S (or the cardinal of S), then S is called a **v-itemset**.

Items and transactions

- The **support** of an itemset X, sup(X), is equal to the number (or proportion) of transactions in T containing X. The support may be given as an absolute value (number of transactions) or proportion or percent (proportion of transactions).
- In many cases we want to find itemsets with a support greater or equal with a given value (percent) s. Such an itemset is called **frequent itemset** and, in the market basket example, it contains items that can be found together in ***many*** baskets (where the measure of 'many' is s).
- These frequent itemsets are the source of all 'powerful' association rules.

Example 1

Let us consider a dataset containing market basket transactions:

- I = {laptop, mouse, tablet, hard-drive, monitor, keyboard, DVD-drive, CD-drive, flash-memory, . . .}
- T = {t1, t2, …, tm}
- t_1 = {laptop, mouse, tablet}
- t_2 = {hard-drive, monitor, laptop, keyboard, DVD-drive}
- . . .
- t_m = {keyboard, mouse, tablet)

- S = {keyboard, mouse, monitor, laptop} is a 4-itemset.

Example 2

If items are words and transactions are documents, where each document is considered a bag of words, then we can have:

- T = {Doc1, Doc2, …, Doc6}
- Doc1 = {rule, tree, classification}
- Doc2 = {relation, tuple, join, algebra, recommendation}
- Doc3 = {variable, loop, procedure, rule}
- Doc4 = {clustering, rule, tree, recommendation}
- Doc5 = {join, relation, selection, projection, classification}
- Doc6 = {rule, tree, recommendation}

Example 2

In that case:

- sup({rule, tree}) = 3 or 50% or 0.5
- sup({relation, join}) = 2 or 33.33% or 1/3
- If the threshold is s = 50% (or 0.5) then {rule, tree} is frequent and {relation, join} is not.

Frequent itemsets and rules

Frequent itemsets and rules:

- ❑ Items and transactions
- ❑ Association rules
- ❑ Goals for mining transactions

Association rules

- If X and Y are two itemsets, an **association rule** is an implication of the form $X \rightarrow Y$ where $X \cap Y = \varnothing$.
- For each association rule we can compute the **support of the rule** and the **confidence of the rule**. If **m** is the number of transactions in T then:
  - $\sup(X \rightarrow Y) = \sup(X \cup Y) / m$
  - $\mathrm{conf}(X \rightarrow Y) = \sup(X \cup Y) / \sup(X)$

where the support of an itemset is given as absolute value (number of transactions).

Association rules

- The **support** of a rule $X \rightarrow Y$ is given by the proportion of transactions in T containing both X and Y.
- The **confidence** of a rule $X \rightarrow Y$ is given by the proportion of transactions containing Y in the set of transactions containing X (the set of transactions containing $X \cup Y$ is included in the set of transactions containing X).
- If the **support** of a rule **is high** it describes a relationship between itemsets that are found together in many transactions (itemsets in many baskets or word sets in many documents).
- If the **confidence** of a rule $X \rightarrow Y$ **is high** then if a transaction contains X then with a high probability (given by the confidence of the rule) it also contains Y

Example 3

Consider the set of six transactions in Example 2:

- Doc1 = {rule, tree, classification}
- Doc2 = {relation, tuple, join, algebra, recommendation}
- Doc3 = {variable, loop, procedure, rule}
- Doc4 = {clustering, rule, tree, recommendation}
- Doc5 = {join, relation, selection, projection, classification}
- Doc6 = {rule, tree, recommendation}

Example 3

- With a minimum support of 50% we find that {rule, tree} is a frequent itemset. The two rules with the same minimum support are:
  - rule → tree

  with sup = 50% and conf = 3 / 4 = 75% and
  - tree → rule

  with sup = 50% and conf = 3 / 3 = 100%
- If the minimum confidence required is 80% then only the second rule is kept, the first being considered not enough powerful.

Frequent itemsets and rules

Frequent itemsets and rules:

- ❑ Items and transactions
- ❑ Association rules
- ❑ <u>Goals for mining transactions</u>

Goals for mining transactions

- The previous paragraph states that if the support of a rule X → Y is high, then X and Y can be found together in many transactions.
- Consequently, both itemsets are part of a frequent itemset (considering the same minimum support), or in other terms, each such rule can be determined starting from a frequent itemset and dividing it in two disjoint parts: X and Y.

Goals for mining transactions

- That means that the process of finding all the rules given the minimum support and the minimum confidence has three steps:
  - **Step 1.** Find all frequent itemsets containing at least two items, considering the given minimum support ***minsup***.
  - **Step 2.** For each frequent itemset U found in step 1 list all splits (X, Y) with $X \cap Y = \emptyset$ and $X \cup Y = U$. Each split generates a rule $X \rightarrow Y$.
- Step 3. Compute the confidence of each rule. Keep only the rules with confidence at least ***minconf***.

Goal 1

- **Find frequent itemsets**. Frequent itemsets can be used not only to find rules but also for marketing purposes.
- As an example, in a supermarket, frequent itemsets helps marketers to place items in an effort to control the way customers walk through the store:
- Items that are sold together are placed for example in distant corners of the store such that customers must go from one product to another possibly putting other products in the basket on the way.

Goal 2

- **Find association rules**. Such a rule tells that people that buy some items X buy some other items Y with a high probability. Association rules may be used for marketing purposes.
- A well known example in the domain literature is the rule:

Diapers → Beer

Diapers → Beer

In [Whitehorn 06] this example is described as follows:

- “Some time ago, Wal-Mart decided to combine the data from its loyalty card system with that from its point of sale systems.
- The former provided Wal-Mart with demographic data about its customers, the latter told it where, when and what those customers bought.
- Once combined, the data was mined extensively and many correlations appeared.
- Some of these were obvious; people who buy gin are also likely to buy tonic. They often also buy lemons.
- However, one correlation stood out like a sore thumb because it was so unexpected.

Diapers → Beer

On Friday afternoons, young American males who buy diapers (nappies) also have a predisposition to buy beer.

- No one had predicted that result, so no one would ever have even asked the question in the first place.
- Hence, this is an excellent example of the difference between data mining and querying."

- This example is only a Data Mining myth (as Daniel Power described in http://www.dssresources.com/newsletters/66.php) but was for many years a very used example because the psychological impact.

Goal 3

In [Ullman 03-09] is listed also a third goal for mining transactions:

- **Goal 3: Find causalities**. In the case of the rule Diapers → Beer a natural question is if the left part of the rule (buying diapers) causes the right part (buy also beer).
- Causal rules can be used in marketing purposes: a low price of diapers will attract diaper buyers and an increase of the beer price will grow the overall sales numbers.

Algorithms

- There are many algorithms for finding frequent itemsets and consequently the association rules in a dataset.
- All these algorithms are developed for huge volumes of data, meaning that the dataset is too large to be read and processed in the main memory.
- For that reason minimizing the number of times the data are read from the disk become a key feature of each algorithm.

Road Map

- Frequent itemsets and rules
- Apriori algorithm
  - Apriori principle
  - Apriori algorithm
- FP-Growth
- Data formats
- Class association rules
- Sequential patterns. GSP algorithm

Apriori algorithm

- This algorithm is introduced in 1994 in [Agrawal, Srikant 94], at the VLDB conference (VLDB = International Conference on Very Large Databases).
- It is based on the Apriori principle (called also monotonicity or downward closure property).

Apriori principle

The Apriori principle states that any subset of a frequent itemset is also a frequent itemset.

Example 4: If {1, 2, 3, 4} is a frequent itemset then all its four subsets with 3 values are also frequent: {1, 2, 3}, {1, 2, 4}, {1, 3, 4} and (2, 3, 4}.

- ❑ Consequently each frequent v-itemset is the reunion of v (v-1)-itemsets.
- ❑ That means we can determine the frequent itemsets with dimension v examining only the set of all frequent itemsets with dimension (v-1).

Apriori principle

- It means that an algorithm for finding frequent itemsets must:
  1. find first frequent 1-itemsets (frequent items)
  2. find frequent 2-itemsets considering all pairs of frequent itemsets found in step 1
  3. Find frequent 3-itemsets considering all triplets with each subset in the frequent pairs set found in step 2
  4. . . . and so on.
- It is a level-wise approach where each step requires a full scan of the dataset (residing on disk).
- A diagram is presented in the next slide where C_i is the set of candidates for frequent i-itemsets and L_i is the actual set of frequent i-itemsets.
- C_1 is the set of all itemsets found in transactions (a subset of I) and may be obtained either by a reunion of all transactions in T or by considering C_1 = I (in that case some items may have a zero support)

Using Apriori Principle

$$\begin{tikzcd}
C_1 \arrow[r] & L_1 \arrow[dl] \\
C_2 \arrow[r] & L_2 \arrow[dl] \\
C_3 \arrow[r] & L_3 \arrow[dl] \\
{} & {} \arrow[dl] \\
C_k \arrow[r] & L_k
\end{tikzcd}$$

The process stops in two cases:

- No candidate from C_k has the support at least minsup (L_k is empty; this case is included in case 2 below)
- There is no (k+1)-itemset with all k-subsets in L_k (meaning that C_{k+1} is empty)

Apriori Algorithm

The algorithm is described in [Agrawal, Srikant 94] and uses the level-wise approach described before:

- A first scan of the dataset leads to the L_1 (the set of frequent items). For each transaction **t** in T and for each item **a** in t the count of **a** is increased (a.count++). At the end of the scan L_1 will contain all items with a count at least minsup (given as absolute value).
- For k=2, 3, … the process continues by generating the set of candidates C_k and then counting the support of each candidate by a full scan of the dataset.
- Process ends when L_k is empty.

Apriori Algorithm

```
Algorithm Apriori (T)
$L_1$ = scan (T);
for (k = 2; $L_{k-1} \neq \emptyset$; k++)  do
      $C_k \leftarrow$ apriori-gen($L_{k-1}$);
      for each transaction $t \in T$ do
         for each candidate $c \in C_k$ do
               if c is contained in t then
                  c.count++;
         end
      end
      $L_k \leftarrow \{c \in C_k \mid c.count \geq minsup\}$
end
return $L = \cup L_k$;
```

Candidate generation

- Candidate generation is also described in the original algorithm as having two steps: the join step and the prune step. The first builds a larger set of candidates and the last removes some of them proved impossible to be frequent.
- In the **join step** each candidate is obtained from two different frequent (k-1)-itemsets containing (k-2) identical items:
- $C_k = \{ \{i_1, \ldots, i_{k-1}, i'_{k-1}\} \mid \exists p=\{i_1, \ldots, i_{k-1}\} \in L_{k-1}$, $\exists q=\{i1, \ldots, i'k-1\} \in L_{k-1}$, $i_{k-1} < i'_{k-1}\}$

Candidate generation

- The **prune step** removes those candidates containing a (k-1)-itemset that is not in Lk-1 (if a subset of an itemset is not frequent the whole itemset cannot be frequent, as a consequence of the Apriori principle):
- The **join step** is described in [Agrawal, Srikant 94] as a SQL query (see Figure 3). Each candidate in C_k comes from two frequent itemsets in L_{k-1} that differ by a single item

Join

The join step

```
insert into $C_k$
    select $p.item_1$, $p.item_2$, ..., $p.item_{k-1}$, $q.item_{k-1}$
    from $L_{k-1}$ p, $L_{k-1}$ q
    where $p.item_1 = q.item_1$, . . ., $p.item_{k-2} = q.item_{k-2}$,
            $p.item_{k-1} < q.item_{k-1}$;
```

Join and prune

The prune step

```
foreach c in $C_k$
   foreach (k-1)-subset q of c do
        if ($s \notin L_{k-1}$) then
              remove c from $C_k$;
   end
end
```

Example 5

Consider again the set of six transactions in Example 2:

- Doc1 = {rule, tree, classification}
- Doc2 = {relation, tuple, join, algebra, recommendation}
- Doc3 = {variable, loop, procedure, rule}
- Doc4 = {clustering, rule, tree, recommendation}
- Doc5 = {join, relation, selection, projection, classification}
- Doc6 = {rule, tree, recommendation}

and a minimum support of 50% (minsup=3).

Step 1

- At the first scan of the transaction dataset T the support for each item is computed:

Rule	**4**	**recommendation**	**3**
Tree	**3**	**variable**	**1**
classification	**2**	**loop**	**1**
relation	**2**	**procedure**	**1**
Tuple	**1**	**clustering**	**1**
Join	**2**	**selection1**	**1**
algebra	**1**	**projection**	**1**

With a minsup=3, L1 = { {rule}, {tree}, {recommendation} }.

Step 2

Considering:

rule < tree < recommendation

- From the join C2 = { {rule, tree}, {rule, recommendation}, {tree, recommendation} }.
- The prune step does not modify C2.
- The second scan of the transaction dataset leads to the following pair support values:

Step 2

- **Step 2**

{rule, tree}	3
{rule, recommendation}	2
{tree, recommendation}	2

- The only frequent pair is {rule, tree}. L2 = { {rule, tree} }.
- **Step 3**. Because L2 has a single element, C3 = ∅, so L3 = ∅, and the process stops. L = L1 ∪ L2 = { {rule}, {tree}, {recommendation}, {rule, tree} }. If we consider only maximal itemsets, L = { {recommendation}, {rule, tree} }.

Example 6

Consider the transaction dataset {(1, 2, 3, 5), (2, 3, 4), (3, 4, 5)} and the minsup s = 50% (or s = 3/2; because s must be an integer $\Rightarrow$ s = 2)

- C_1 = {1, 2, 3, 4, 5}
  L_1 = {2, 3, 4, 5}
- C_2 = {(2, 3), (2, 4), (2, 5), (3, 4), (3, 5), (4, 5) }
  L_2 = {(2, 3), (3, 4), (3, 5)}
- After the join step C_3 = {(3, 4, 5)} - obtained by joining (3, 4) and (3, 5).
- In the prune step (3, 4, 5) is removed because its subset (4, 5) is not in L_2

Example 6

- After the prune step = ∅, so L3 = ∅, and the process stops. L = L1 ∪ L2 = {(2), (3), (4), (5), (2, 3), (3, 4), (3, 5)} or as maximal itemsets L = L2.
- The rules generated from itemsets are:

  2 → 3, 3 → 2, 3 → 4, 4 → 3, 3 → 5, 5 → 3.
- The support of these rules is at least 50%.
- Considering a minconf equal to 80% only 3 rules have a confidence greater or equal with minconf: 2 → 3, 4 → 3, 5 → 3 (with conf = 2/2 = 100%).
- The rules having 3 in the antecedent have a confidence of 67% (because sup(3) = 3 and sup(2, 3) = sup(3, 4) = sup(3, 5) = 2).

Apriori summary

- Apriori algorithm performs a level-wise search.
- If the largest frequent itemset has **w** items, the algorithm must perform **w** passes over the data (most probably stored on disk) for finding all frequent itemsets.
- In many implementations the maximum number of passes may be specified, leading to frequent itemsets with the same maximum dimension.

Road Map

- ❑Frequent itemsets and rules
- ❑Apriori algorithm
- ❑FP-Growth
- ❑Data formats
- ❑Class association rules
- ❑Sequential patterns. GSP algorithm

FP-Growth

FP-Growth algorithm performs frequent itemsets discovery without candidate generation. It has a 2 steps approach:

- ❑**Step 1**: Build FP-tree. Requires only 2 passes over the dataset.
- ❑**Step 2**: Extracts frequent itemsets from the FP-tree.

Build the FP-tree

- At the first pass over the data frequent items are discovered.
- All other items (infrequent items) are discarded: transactions will contain only frequent items.
- Also, these items are ordered by their support in decreasing order.

Build the FP-tree

At the second pass over the data the FP-tree is built:

- FP-Growth considers an ordered set of item (frequent items in decreasing order).
- Each transaction is written with items in that order.
- The algorithm reads a transaction at a time and adds a new branch to the tree, branch containing as nodes the transaction items.

Build the FP-tree

Pass 2 - cont.:

- Each node has a counter.
- If two transactions have the same prefix the two branches overlap on the nodes of the common prefix and the counter of those nodes are incremented.
- Also, nodes with the same item are linked by orthogonal paths.

Example 7

- Let’s consider the following transaction set and minsup=3

TID	Items
1	a, b, c
2	a, b, c, d
3	a, b, f
4	a, b, d
5	c, d, e

Example 7

- Item counts. Only a, b, c, d are frequent:

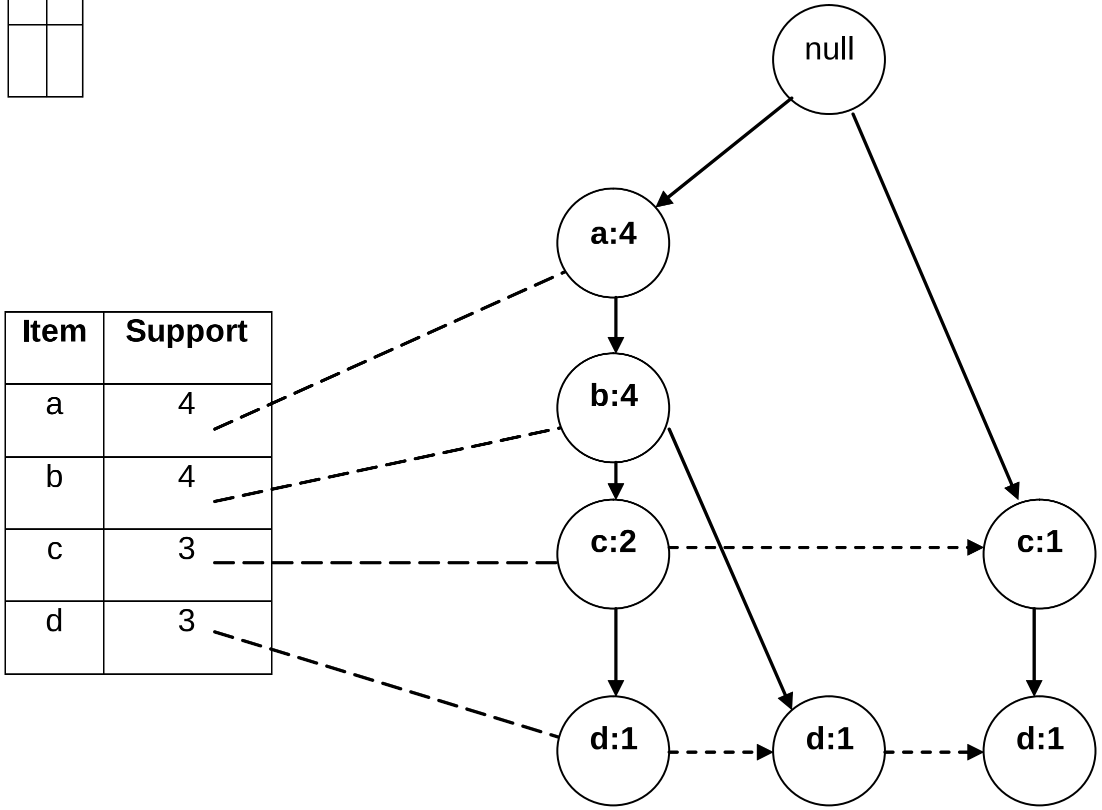
a	4
b	4
c	3
d	3
e	1
f	1

Example 7

- After discarding nonfrequent items and ordering descending using the item support the transaction dataset is the following. Note that the order is a, b, c, d.

TID	Items
1	a, b, c
2	a, b, c, d
3	a, b
4	a, b, d
5	c, d

Example 7



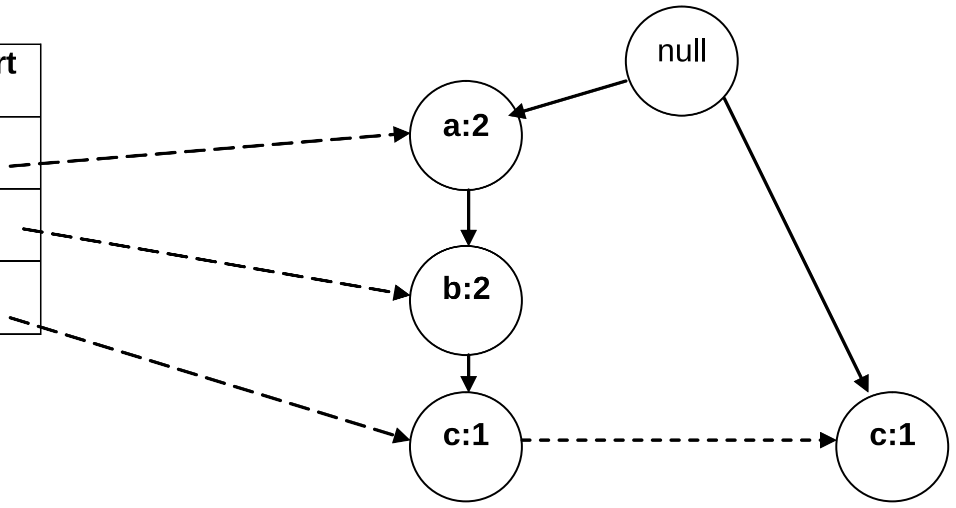
Item	Support
a	4
b	4
c	3
d	3

Extract frequent itemsets

- After building the FP-tree the algorithm starts to build partial trees (called conditional FP-trees) ending with a given item (a suffix).
- The item is not present in the tree but all frequent itemsets generated from that conditional tree will contain that item.
- In building the conditional FP-tree, non-frequent items are skipped (but the branch remains if there are still nodes on it).
- In the previous example trees ending with d, c, b and a may be built.
- For each tree the dotted path is used for starting points and the algorithm goes up propagating the counters (with sums where two branches go to the same node).

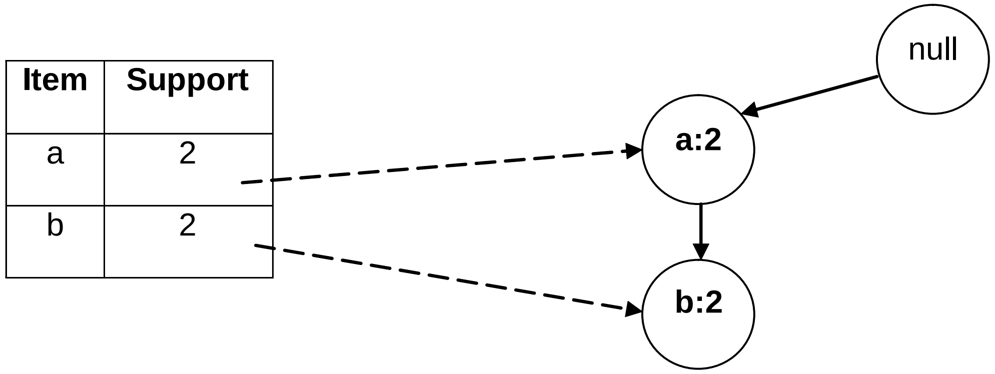
d conditional FP-tree

Item	Support
a	2
b	2
c	2



c conditional FP-tree

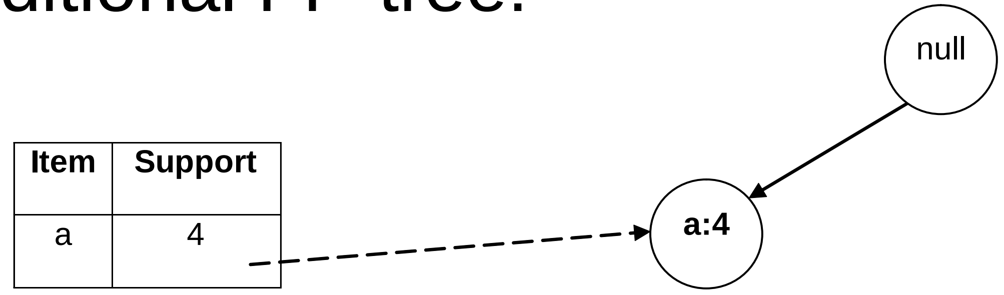
- Because all items have a support below minsup, no itemset containing d is frequent.
- The same situation is for the c conditional FP-tree:



Item	Support
a	2
b	2

b conditional FP-tree

- b conditional FP-tree:



Item	Support
a	4

- Because the support of a is above minsup {a, b} is a frequent itemset.
- In fact, {a, b} is the only frequent itemset with more than one item produced by the algorithm (there are no other frequent itemsets in the given dataset for minsup=3).

Results

- If there are more than one item with support above or equal minsup in a conditional FP-tree then the algorithm is run again against the conditional FP-tree to find itemsets with more than two items.
- For example, if the minsup=2 then from the c conditional FP-tree the algorithm will produce {a, c} and {b, c}. Then the same procedure may be run against this tree for suffix **bc**, obtaining {a, b, c}. Also from the d conditional FP-tree first {c, d}, {b, d} and {a, d} are obtained, and then, for the suffix **bd**, {a, b, d} is obtained.
- Suffix **cd** leads to unfrequent items in the FP-tree and suffix **ad** produces {a, d}, already obtained.

Road Map

- ❑Frequent itemsets and rules
- ❑Apriori algorithm
- ❑FP-Growth
- ❑Data formats
- ❑Class association rules
- ❑Sequential patterns. GSP algorithm

Data formats

Table format

- In this case a dataset is stored in a two columns table:

  Transactions(Transaction-ID, Item) or

  T(TID, Item)

where all the lines of a transaction have the same TID and the primary key contains both columns (so T does not contain duplicate rows).

Data formats

Text file format

- In that case the dataset is a textfile containing a transaction per line. Each line may contain a transaction ID (TID) as the first element or this TID may be missing, the line number being a virtual TID.

Example 8:

```
10 12 34 67 78 45 89 23 67 90  → line 1
789 12 45 678 34 56 32         → line 2
. . . . . . . .
```

- Also in this case any software package must either have a native textfile input option or must contain a conversion module from text to the needed format

Data formats

Custom format

- Many data mining packages use a custom format for the input data.
- An example is the ARFF format used by Weka, presented below. Weka means Waikato Environment for Knowledge Analysis) is a popular open source suite of machine learning software developed at the University of Waikato, New Zealand.
- ARFF stands for Atribute-Relation File Format. An .arff file is an ASCII file containing a table (called also relation). The file has two parts:
  - A ***Header*** part containing the relation name, the list of attributes and their types.
  - A ***Data*** part containing the row values of the relation, comma separated.

ARFF example

```
% 1. Title: Iris Plants Database
%
% 2. Sources:
%    (a) Creator: R.A. Fisher
%    (b) Donor: Michael Marshall MARSHALL%PLU@io.arc.nasa.gov)
%    (c) Date: July, 1988
%
@RELATION iris
@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}

@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

Road Map

- ❑Frequent itemsets and rules
- ❑Apriori algorithm
- ❑FP-Growth
- ❑Data formats
- ❑Class association rules
- ❑Sequential patterns. GSP algorithm

Class association rules (CARs)

- As for clasical association rules, the model for CARs considers a set of items I = {i1, i2, …, in} and a set of transactions T = {t1, t2, …, tm}. The difference is that each transaction is labeled with a class label c, where $c \in C$, with C containing all class labels and $C \cap I = \emptyset$.
- A class association rule is a construction with the following syntax:

$$X \to y$$

  where $X \subseteq I$ and $y \in C$.
- The definition of the **support** and **confidence** for a class association rule is the same with the case of association rules.

Example 10

- Consider the set of six transactions in Example 2, now labeled with class labels from C = {database, datamining, programming}:

Doc1	**{rule, tree, classification}**	**datamining**
Doc2	**{relation, tuple, join, algebra, recommendation}**	**database**
Doc3	**{variable, loop, procedure, rule}**	**programming**
Doc4	**{clustering, rule, tree, recommendation}**	**datamining**
Doc5	**{join, relation, selection, projection, classification}**	**database**
Doc6	**{rule, tree, recommendation}**	**datamining**

Example 10

Then the CARs:

rule → datamining;

recommendation → database

has:

- ❑sup(rule → datamining) = 3/6 = 50%,
- ❑conf(rule → datamining) = 3/4 = 75%.
- ❑sup(recommendation → database) = 1/6 ≅ 17%,
- ❑conf(recommendation → database) = 1/3 ≅ 33%

For a minsup=50% and a minconf=50% the first rule stands and the second is rejected.

Mining CARs

- Algorithm for mining CARs using a modified Apriori algorithm (see [Liu 11]):
- At the first pass over the algorithm computes F1 where

F1= { the set of CARs with a single item on the left side verifying a given minsup and minconf}.

- At step k, C_{k-1} is built from F_{k-1} and then, passing through the data and counting for each member of C_{k-1} the support and the confidence, F_k is determined.
- Candidate generation is almost the same as for association rules with the only difference that in the join step only CARs with the same class in the right side are joined.

Candidates generation

```
$C_k$ = ∅ // starts with an empty set
forall f1, f2 ∈ Fk-1 // for each pair of frequent CAR
    f1 = {i1, … , $i_{k-2}$, $i_{k-1}$}  → y // only last item
    f2 = {i1, … , $i_{k-2}$, $i'_{k-1}$} → y // is different
    $i_{k-1}$ < $i'_{k-1}$ do                  // and same class

    c  = {i1, …, $i_{k-1}$, $i'_{k-1}$} → y;    // join step
    $C_k$ = $C_k$ ∪ {c}; // add new candidate

    for each (k-1)-subset s of {i1, …, $i_{k-1}$, $i'_{k-1}$}  do
        if (s → y ∉ Fk-1) then
            $C_k$ = $C_k$ - {c};              // prune step
  endfor
endfor
```

Road Map

- ❑Frequent itemsets and rules
- ❑Apriori algorithm
- ❑FP-Growth
- ❑Data formats
- ❑Class association rules
- ❑Sequential patterns. GSP algorithm

Sequential patterns model

- **Itemset**: a set of n distinct items

$$I = \{i_1, i_2, \ldots, i_n\}$$

- **Event**: a non-empty collection of items; we can assume that items are in a given (e.b. lexicographic) order: $(i_1, i_2 \ldots i_k)$
- **Sequence** : an ordered list of events: $< e_1 \ e_2 \ \ldots \ e_m >$
- **Length** of a sequence: the number of items in the sequence

Example: <AM, CDE, AE> has length 7

Sequential patterns model

- **Size** of a sequence: the number of itemsets in the sequence

  Example: <AM, CDE, AE> has size 3
- **K-sequence** : sequence with k items, or with length k

  Example: <B, AC> is a 3-sequence
- **Subsequence** and **supersequence**: $<e_1\ e_2\ \ldots e_u>$ is a subsequence of or included in $<f_1\ f_2\ \ldots f_v>$ (and the last is a supersequence of the first sequence or contains that sequence) if there are some integers $1 \leq j_1 < j_2 < \ldots < j_{u-1} < j_u \leq v$ such that $e_1 \subseteq f_{j1}$ and $e_2 \subseteq f_{j2}$ and … and $e_u \subseteq f_{ju}$.

Sequential patterns model

- **Sequence database** X: a set of sequences
- **Frequent sequence** (or **sequential pattern**): a sequence included in more than **s** members of the sequence database X;
- **s** is the user-specified minimum support.
- The number of sequences from X containing a given sequence is called the **support** of that sequence.
- So, a frequent sequence is a sequence with a support at least **s** where **s** is the **minsup** specified by the user.

Example 11

- <A, BC> is a subsequence of <**A**B, E, A**BC**D>
- <AB, C> is not a subsequence of <ABC>
- Consider a minsup=50% and the following sequence database:

Sequence ID	**Sequence**
1	**<A, B, C>**
2	**<AB, C, AD>**
3	**<ABC, BCE>**
4	**<AD, BC, AE>**
5	**<B, E>**

Example 11

- The frequent sequences (support at least 50% means 3 of 5) are:

1-sequences	**<A>, <B>, <C>, <E>**
2-sequences	**<A, B>, <A, C>, <B, C>, <B, E>**

- There is no 3-sequence (or upper) with support at least 50%.

Algorithms

- Apriori
- GSP (Generalized Sequential Pattern)
- FreeSpan (Frequent pattern-projected Sequential pattern mining)
- PrefixSpan (Prefix-projected Sequential pattern mining)
- SPADE (Sequential PAttern Discovery using Equivalence classes)

GSP Algorithm

- Similar with Apriori:

```
Algorithm GSP(I, X, minsup)
$C_1$ = I //  initial n candidates
$L_1$ = {<{f}>| f$\in$ $C_1$, f.count/n $\geq$ minsup}; // first pass over X
for (k = 2; $L_{k-1} \neq \emptyset$; k++) do // loop until $L_{k-1}$ is empty
$C_k$ = candidate-generation($L_{k-1}$);
foreach s $\in$ X do //
foreach c $\in$ $C_k$ do
if c is-contained-in s then
c.count++;
endfor
    endfor
$L_k$ = {c $\in$ $C_k$ | c.count/n $\geq$ minsup}
endfor
return $\cup_k$ $F_k$;
```

GSP Algorithm

- Candidate generation is made in a join and prune manner.
- At the join step two sequences f_1 and f_2 from L_{k-1} are joined if removing the first item from f_1 and the last item from f_2 the result is the same.
- The joined sequence is obtained by adding the last item of f_2 to f_1, with the same status (separate element or part of the last element of f_1).

Example 12

- <AB, CD, E> join with <B, CD, EF>:

  <AB, CD, EF>

- <AB, CD, E> join with <B, CD, E, F>:

  <AB, CD, E, F>

Summary

This third course presented:

- ❑What are frequent itemsets and rules and their relationship
- ❑Apriori and FP-growth algorithms for discovering frequent itemsets.
- ❑Data formats for discovering frequent itemsets
- ❑What are class association rules and how can be mined
- ❑An introduction to sequential patterns and the GSP algorithm

❑ Next week: Supervised learning – part 1.

References

- [Agrawal, Imielinski, Swami 93] R. Agrawal; T. Imielinski; A. Swami: Mining Association Rules Between Sets of Items in Large Databases", SIGMOD Conference 1993: 207-216, (http://rakesh.agrawal-family.com/papers/sigmod93assoc.pdf)
- [Agrawal, Srikant 94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pages 487-499, Santiago, Chile, September 1994 (http://rakesh.agrawal-family.com/papers/vldb94apriori.pdf)
- [Srikant, Agrawal 96] R. Srikant, R. Agrawal: "Mining Sequential Patterns: Generalizations and Performance Improvements", to appear in Proc. of the Fifth Int'l Conference on Extending Database Technology (EDBT), Avignon, France, March 1996, (http://rakesh.agrawal-family.com/papers/edbt96seq.pdf)
- [Liu 11] Bing Liu, 2011. Web Data Mining, Exploring Hyperlinks, Contents, and Usage Data, Second Edition, Springer, chapter 2.
- [Ullman 03-09] Jeffrey Ullman, Data Mining Lecture Notes, 2003-2009, web page: http://infolab.stanford.edu/~ullman/mining/mining.html

References

- [Wikipedia] Wikipedia, the free encyclopedia, en.wikipedia.org
- [Whitehorn 06] Mark Whitehorn, The parable of the beer and diapers, web page: http://www.theregister.co.uk/2006/08/15/beer_diapers/
- [Silverstein et al. 00] Silverstein, C., Brin, S., Motwani, R., Ullman, J. D. 2000. Scalable techniques for mining causal structures. Data Mining Knowl. Discov. 4, 2–3, 163–192., www.vldb.org/conf/1998/p594.pdf
- [Verhein 08] Florian Verhein, Frequent Pattern Growth (FP-Growth) Algorithm, An Introduction, 2008, http://www.florian.verhein.com/teaching/2008-01-09/fp-growth-presentation_v1%20(handout).pdf
- [Pietracaprina, Zandolin 03] Andrea Pietracaprina and Dario Zandolin: Mining Frequent Itemsets using Patricia Tries,
- [Zhao, Bhowmick 03] Qiankun Zhao, Sourav S. Bhowmick, Sequential Pattern Mining: A Survey, Technical Report, CAIS, Nanyang Technological University, Singapore, No. 2003118 , 2003, (http://cs.nju.edu.cn/zhouzh/zhouzh.files/course/dm/reading/reading04/zhao_techrep03.pdf)