Pattern Mining

Knowledge Discovery and Data Mining 2

Roman Kern

ISDS, TU Graz

2019-05-02

Outline

Introduction to Pattern Mining

What & Why

Pattern Mining Intro

Definition of KDD

KDD is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable **patterns** in data

$\rightarrow$ But, until now we did not directly investigate into mining of these patterns.

Pattern Mining Example

Famous example for pattern mining

- Many supermarkets/grocery (chains) do have loyalty cards
- $\rightarrow$ gives the store the ability to analyse the purchase behaviour (and increase sales, for marketing, etc.)
- Benefits
  - Which items have been bought together
  - Hidden relationships: probability of a purchase an item given another item
  - ... or temporal patterns

Pattern Mining Example

Transactions in a grocery

- All purchases are organised in transactions
- Each transaction contains a list of purchased items
- For example:

Id	Items
0	Bread, Milk
1	Bread, Diapers, Beer, Eggs
2	Milk, Diapers, Beer, Coke
3	Bread, Milk, Diapers, Beer
4	Bread, Milk, Diapers, Coke

Pattern Mining Example

Ways of analysing the transaction data

- Frequent itemset (which items are often bought together) $\rightarrow$ **association analysis**
- Frequent associations (relations between items (if/then)) $\rightarrow$ **association rule mining**

Pattern Mining Example

Result

A grocery store chain in the Midwest of the US found that on Thursdays men tend to buy beer if they bought diapers as well.

Association Rule

$\{diapers \rightarrow beer\}$

Note #1: The grocery store could make use of this knowledge to place diapers and beer close to each other and to make sure none of these product has a discount on Thursdays.
Note #2: Not clear whether this is actually true.

Formal description: items and transactions

- $\mathcal{I}$ - the set of **items** (e.g. the grocery goods - inventory)
- $\mathcal{D} = \{t_1, t_2, ..., t_n\}$ - the **transactions**, where each transaction consists of a set of items ($t_x \subseteq \mathcal{I}$)

Note #1: One could see each item as feature and thus each transaction as feature vector with zeros for items not contained in the transaction.
Note #2: This organisation of transaction is sometimes referred to as horizontal data format

Pattern Mining Basics

Formal description: itemset and association rule

- $X = \{I_1, I_2, ..., I_n\}$ - the **itemset**, consisting of a set of items
- *Frequent itemset is an itemset that found often in the transactions*
- $X \rightarrow Y$ - an **association rule**, where X and Y are itemssets (premise $\rightarrow$ conclusion)
- *Frequent association rule is an association rule that is true for many transactions*

Pattern Mining Basics

Formal description: support (itemset)

- How frequent is a frequent itemset?
- $\rightarrow$ What is the probability of finding the itemset in the transactions, which is called the **support**
- $\text{Support}(X) = P(X) = P(I_1, I_2, ..., I_n)$
- $\text{Support}(X) = \frac{|\{t \in \mathcal{D} | X \subseteq t\}|}{|D|}$
- The relative frequency of the itemset (relative number of transactions containing all items from the itemset)

Note: Some algorithms take the total number of transactions instead of the relative frequency

Formal description: support (association rule)

- How frequent is a frequent association rule?
- $\rightarrow$ What is the probability of finding the association rule to be true in the transactions, again called the **support**
- $\text{Support}(X \rightarrow Y) = P(X \cup Y) = P(I_1^X, I_2^X, ..., I_n^X, I_1^Y, I_2^Y, ..., I_m^Y)$
- $\text{Support}(X) = \frac{|\{t \in \mathcal{D} | X \cup Y \subseteq t\}|}{|D|}$
- ... equals to the joint itemset

Formal description: confidence

- How relevant is a (frequent) association rule?
- $\rightarrow$ What is the probability of the conclusion given the premise, which is called the **confidence**
- $\text{Conf}(X \rightarrow Y) = P(I_1^Y, I_2^Y, ..., I_m^Y | I_1^X, I_2^X, ..., I_n^X)$
- $\text{Conf}(X \rightarrow Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)}$
- $\text{Conf}(X \rightarrow Y) = \frac{|\{t \in \mathcal{D} | X \cup Y \subseteq t\}|}{|\{t \in \mathcal{D} | X \subseteq t\}|}$
- The confidence reflects how often one finds Y in transactions, which contain X

Pattern Mining Basics

Association Rule Mining Goal

Find all frequent, relevant association rules, i.e. all rules with a sufficient support and sufficient confidence

What is it useful for?

- Market basket analysis, query completion, click stream analysis, genome analysis, ...
- Preprocessing for other data mining tasks

Pattern Mining Basics

Simple Solution (for association analysis)

1. Pick out any combination of items
2. Test for sufficient support
3. Goto 1 (until all combinations have been tried)

Not the optimal solution $\rightarrow$ two improvements are presented next, the Aprioi Algorithm and the FP-Growth Algorithm

The Apriori Algorithm

... and the apriori principle

Apriori Algorithm

Apriori algorithm

- Simple and efficient algorithm to find frequent itemsets and (optionally) find association rules

Input

- The transactions
- The (minimal) support threshold
- The (minimal) confidence threshold
  (just needed for association rules)

R. Agrawal, H. Road, and S. Jose, "Mining Association Rules between Sets of Items in Large Databases," ACM SIGMOD Record. Vol. 22. No. 2. ACM, 1993.

Apriori Algorithm

Apriori principle

- If an itemset is **frequent**, then all of the **subsets** are frequent as well
- If an itemset is **infrequent**, then all of the **supersets** are infrequent as well

Note: The actual goal is to reduce the number of support calculations

Apriori Algorithm

Apriori basic approach for frequent itemsets

1. Create frequent itemsets of size $k = 1$ (where k is the size of the itemset)
2. Extends the itemset for size $k + 1$, following the apriori principle (generate superset candidates only for frequent itemsets)
3. Only test those itemsets for sufficient support that are theoretically possible
4. Goto 2 (until there are no more candidates)

Apriori Algorithm

Apriori basic approach for frequent association rules

1. Take the frequent itemsets (of size $k \geq 2$)
2. Create frequent association rules of size $k' = 1$ (where k' is the size of the conclusion set), by picking each $I_i \subset X$ and test for sufficient confidence (i.e. split the itemset into premise and conclusion)
3. Extends the conclusion for size $k' + 1$, following the apriori principle (and create new association rule candidates)
4. Only test those association rules for sufficient confidence that are theoretically possible
5. Goto 2 (until there are no more candidates)

The FP-Growth Algorithm

... more efficient than Apriori

FP-Growth Algorithm

Overview

- (Usually) faster alternative to Apriori for finding frequent itemsets
  - Needs two scans through the transaction database
- Main idea: transform the data set into an efficient data structure, the FP Tree
  1. Build the FP-Tree
  2. Mine the FP-Tree for frequent itemsets
     - ★ ... thereby creating (conditional) FP-Tree on-the-fly

FP-Growth Algorithm

The FP-Tree

- Contains all information from the transactions
- Allows efficient retrieval of frequent itemsets
- Consists of a tree
  - Each node represents an item
  - Each node has a count, # of transactions from the root to the node
- Consists of a linked list
  - For each frequent item there is a head with count
  - Links to all nodes with the same item

FP-Growth Algorithm

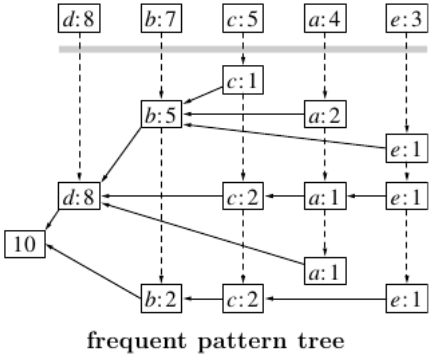
Building the FP-Tree

1. Scan through all transactions and count each item, $I_x \rightarrow count_x$
2. Scan through all transactions and
   1. Sort the items (descending) by count
   2. Throw away infrequent items, $count_x < minSupport$
   3. Iterate over the items and match against tree (starting with the root node)
      - ★ if the current item is found as child to the current node, increase its count
      - ★ if the current item is not found, create a new child (branch)

FP-Growth Algorithm

Simple Example Database

(1)	(4)
a d f	d b
a c d e	d b c
b d	d b a
b c d	d b a
b c	d b e
a b d	d c
b d e	d c a e
b c e g	d a
c d f	b c
a b d	b c e



frequent pattern tree

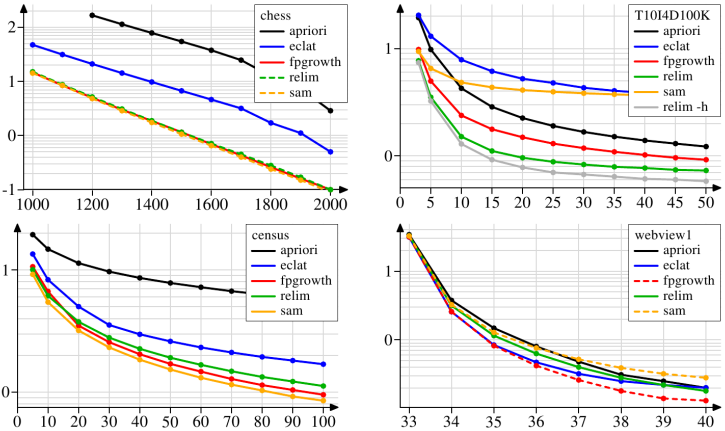
Given a set of transactions (1), sort the items by frequency in the data set and prune infrequent items (4), create the FP-Tree with the tree itself (root on the left) a linked list for each of the items (head on the top)

FP-Growth Algorithm

Mine the FP-Tree for frequent itemsets

1. Start with single item itemsets
2. Recursively mine the FP-Tree
   - Starting by the items and traverse to the root
   - Test each “parent” items for minimal support
     - ★ Build a new (conditional) FP-Tree (new recursion)

FP-Growth Algorithm



Comparison of Apriori, FP-Growth and others on four data sets with varying support counts (x-axis), measured in runtime (y-axis).

Sequence Pattern Mining

Mining patterns in temporal data

Sequence Pattern Mining

Initial definition

Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items, and given a user-specified min_support threshold, sequential pattern mining is to find all frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than min_support.

- R. Agrawal and R. Srikant, "Mining Sequential Patterns," Proc. 1995 Int'l Conf. Data Eng. (ICDE '95), pp. 3-14, Mar. 1995.

Note #1: This definition can then be expanded to include all sorts of constraints.
Note #2: Each element in a sequence is an itemset.

Sequence Pattern Mining

Applications & Use-Cases

- Shopping cart analysis
  - e.g. common shopping sequences
- DNA sequence analysis
- (Web)Log-file analysis
  - Common access patterns
- Derive features for subsequent analysis

Transaction database vs. sequence database

ID	Itemset
1	a, b, c
2	b, c
3	a, e, f, g
4	c, f, g

ID	Sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

Sequence Pattern Mining

Subsequence & Super-sequence

- A sequence is an ordered list of elements, denoted $< x_1\, x_2\, \ldots\, x_k >$
- Given two sequences $\alpha =< a_1\, a_2\, \ldots\, a_n >$ and $\beta =< b_1\, b_2 \ldots\, b_m >$
- α is called a subsequence of β, denoted as $\alpha \sqsubseteq \beta$
  - if there exist integers $1 \leq j_1 < j_2 < \cdots < j_n \leq m$ such that $a_1 \subseteq b_{j_1}, a_2 \subseteq b_{j_2}, \ldots, a_n \subseteq b_{j_n}$
- β is a super-sequence of α
  - e.g. $\alpha =< (ab), d >$ and $\beta =< (abc), (de) >$

Note #1: A database transaction (single sequence) is defined to contain a sequence, if that sequence is a subsequence.
Note #2: Frequently found subsequences (within a transaction database) are called sequential patterns.

Sequence Pattern Mining

- Initial approaches to sequence pattern mining relied on the a-priory principle
  - If a sequence S is not frequent, then none of the super-sequences of S is frequent
- Generalized Sequential Pattern Mining (GSP) is one example of such algorithms
- Disadvantages:
  - Requires multiple scans within the database
  - Potentially many candidates

FreeSpan - Algorithm

FreeSpan - Frequent Pattern-Projected Sequential Pattern Mining

- Input: minimal support, sequence database
- Output: all sequences with a frequency equal or higher than the minimal support
- Main idea: Recursively reduce (project) the database instead of scanning the whole database multiple times
- Projection
  - Reduced version of a database consisting just a set of sequences
  - Thus each sequence in the projected database contains the sequence
- Keep the sequences sorted
  - All sequences are kept in an fixed order
  - Starting with the most frequent to the least frequent

FreeSpan - Algorithm

1. Compute all items that pass the minimal support test, sort them by frequency $\{x_1, \ldots, x_n\}$
2. For each x_i of the items build a set of items $\{x_1, \ldots, x_i\}$
   - Project the database given the set of items
   - Mine the projected database containing x_i for frequent sequences of length 2
     - ★ Recursively apply the algorithm on the projected database to find frequent sequences of $length + 1$

Alternatives & Remarks

... and remarks

Remarks

Problems of frequent association rules

- Rules with high confidence, but also high support
  - e.g. use significance test
- Confidence is relative to the frequency of the premise
  - Hard to compare multiple association rules
- Redundant association rules
- Many frequent itemsets (often even more than transactions)

Remarks

How to deal with numeric attributes?

- Binning (discretization) on predefined ranges
- Distribution-based binning
- Extend algorithm to allow numeric attributes, e.g. Approximate Association Rule Mining

Frequent closed pattern mining

- A pattern is called closed, if there is no superset that satisfies the minimal support
- Produces a smaller number of patterns
- Similar maximum pattern mining

Further Algorithms

- Frequent tree mining
- Frequent graph mining
- Constraint based mining
- Approximate pattern mining
  - e.g. picking a single representative for a group of patterns
  - ... by invoking a clustering algorithm
- Mining cyclic or periodic patterns

J. Han, H. Cheng, D. Xin, and X. Yan, “Frequent pattern mining: current status and future directions,” Data Min. Knowl. Discov., vol. 15, no. 1, pp. 55–86, Jan. 2007.

Connection to ML

Connection to other tasks in machine learning

- Frequent pattern based classification
  - Mine the relation between features/items and the label
  - ... useful for feature selection/engineering
- Frequent pattern based clustering
  - Clustering in high dimensions is a challenge, e.g. text
  - ... use frequent patterns as a dimensionality reduction

Tools

... for pattern mining

Connection to ML

Java Tools

- SPMF - http://www.philippe-fournier-viger.com/spmf/

The End

Christian Borgelt Slides: http://www.borgelt.net/teach/fpm/slides.html (516 slides)
http://www.is.informatik.uni-duisburg.de/courses/im_ss09/folien/MiningSequentialPatterns.ppt