Association Analysis

Josep Curto Díaz, Adjunct Professor[a]

[a] IE Business School, Madrid, 28006, Spain, jcurto@faculty.ie.edu

This version was compiled on April 16, 2019

This technical note introduces what is Association Analysis and its benefits and limitations.

association analysis | machine learning | recommender systems | customer analytics | r

The problem. Many companies accumulate large quantities of customer purchasing data from their day-to-day operations. This dataset represents an opportunity to understand customer purchasing preferences.

- **Problem**: we don't know which products are purchased by customers, which products are purchased jointly and which regularities can be found between products and customers.
- **Goals**:
  - We want to understand customer purchasing preferences.
- **Why?** Perform specific actions to improve our services and products (such as placement, recommendations, etc.)

A more general problem: Recommender Systems. Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user.

"Item" is the general term used to denote what the system recommends to users. A RS normally focuses on a specific type of item (e.g., CDs, or news) and accordingly its design, its graphical user interface, and the core recommendation technique used to generate the recommendations are all customized to provide useful and effective suggestions for that specific type of item.

RSs are primarily directed towards individuals who lack sufficient personal experience or competence to evaluate the potentially overwhelming number of alternatives.

- Netflix: 2/3 of the movies watched are recommended .
- Google News: recommendations generate 38% more click-through.
- Amazon: 35% sales are generated from recommendations.
- Choicestream: 28% of the people would buy more music if they found what they liked.

There are two types of recommender systems:

- **Personalized**: Different users receive different suggestions
- **Non-Personalized**: Easy to create based on simple rules. Example, top-read articles.

The Recommender Problem. The **goal** is:

- Estimate a utility function that automatically predicts how a user will like an item.

Based on:

- Past behaviour
- Relations to other users
- Product attributes
- Item similarity
- Context

There are several techniques:

- **Collaborative Filtering**: Recommend items based only on the users past behavior
- User-based: Find similar users to me and recommend what they liked
- Item-based: Find similar items to those that I have previously liked
- **Content-based**: Recommend based on item features
- **Personalized Learning to Rank**: Treat recommendation as a ranking problem
- **Demographic**: Recommend based on user features
- **Social recommendations** (trust-based)
- **Hybrid**: Combine any of the above

We will focus on one of the **Collaborative Filtering** techniques: **Association Analysis**. But there are many others (such as Factorization Machines).

What is Association Analysis.

- It is a technique that helps to detect and analyse the relationships in registered transactions of individuals, groups and objects.
- One the most known analysis is the **market basket analysis** aiming to understand the relationship between acquired products.
- We will use the **Apriori Algorithm**. But there are many others such as eclat or opus.

Main Concepts. The main concepts are:

- **Association rules**, described by lhs => rhs:
- **Lhs** refers to the left element in the rule and it is the acronym of *left hand side*. It is an itemset.
- **Rhs** refers to the right element in the rule and it is the acronym of *right hand side*. It is an itemset.
- **Support**: This says how popular an itemset (or an association) is. Measured by the proportion of transactions in which an itemset (association) appears. That means "Number of transactions with both lhs and rhs" divided by "Total number of transactions".

$$support(lhs => rhs) = p(lhs \cap rhs)$$

- **Confidence**: This says how likely itemset rhs is purchased when itemset lhs is purchased, expressed as {lhs -> rhs}. This is measured by the proportion of transactions with item lhs, in which item rhs also appears. That means "Number of transactions with both lhs and rhs" divided by "Total number of transactions with lhs".

$$confidence(lhs => rhs) = \frac{p(lhs \cap rhs)}{p(lhs)}$$

- **Expected Confidence**: refers to how likely itemset rhs is purchased. That means “Number of transactions with rhs” divided by “Total number of transactions”.

$$expectedconfidence(rhs) = p(rhs)$$

- **Lift** This says how likely item Y is purchased when item X is purchased, while controlling the expected confidence for itemset rhs is. That means “confidence(lhs => rhs)” divided by “expectedconfidence(rhs)”.

$$lift(lhs => rhs) = \frac{p(lhs \cap rhs)}{p(lhs)p(rhs)}$$

The best predictor is *lift*. We should start our analysis with this parameter. - If $lift < 1$, lhs presence doesn’t imply rhs presence. - If $lift = 1$, lhs and rhs are independent. - If $lift > 1$, lhs presence increases the probability of rhs presence in the transaction.

Another useful metric is **conviction**: It measures the frequency at which the rule makes an incorrect prediction. Additional interest measures can be calculated for existing sets of itemsets or rules. Definitions and equations (including conviction) can be found in Hahsler (2015).

How to apply Association Analysis.

- **[BU/DU]** Determine whether Association Analysis fits in our business
- **[DP]** Identification and understanding of sources and metadata
- **[DP]** Extract, clean and load data
- **[DP]** Purchasing Transaction identification
- **[M]** Choose the right algorithm (Apriori, Eclat, FP-growth, etc.)
- **[M]** Choose starting values for **lift**, **support** and **confidence**
- **[M]** Adjust **lift**, **support** and **confidence**
- **[E]** Analyze results and adjust parameters
- **[D]** Present and explain the results/embed in business process

Benefits.

- Understand purchasing patterns. Example: Who is buying what? Which products are the best ones by region, channel, account,…? Which is the profile?
- Open the door to other analysis such as Frequent items analysis: understanding the combination of products more popular.
- Helps to unlock Propension Analysis. Example: Who will buy what? Which is the profile of those customers?
- We can create recommendation based on transactional historical data. It is a recommender system (collaborative-filtering).

Use Cases.

- **Online**: automated recommendation system, discounts, promotion, placement, cross-selling, etc.
- **Offine**: product placement, supplier management, store design, product catalog design, bundles, discounts, promotions, etc.
- **Customer Retention**: it can be used to create compelling arguments (using, for example, product bundles) to retain customers.
- **Other**: Web usage mining, Intrusion detection, Continuous production, Bioinformatics, etc.

Association Analysis Algorithms.

Apriori Algorithm is only one of available algorithms for association analysis. Let’s discuss about other options. In general, mining association rules has a two-step approach:

- **Frequent Itemset Generation**: Generate all itemsets whose support greater than minsup 1.
- **Rule Generation**: Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

Frequent itemset generation may be computationally expensive. How we can generate all itemsets? One initial approach is **Brute-force approach**:

- Each itemset in the lattice is a candidate frequent itemset – Count the support of each candidate by scanning the database – Match each transaction against every candidate – Complexity ~ O(NMw) => Expensive since M = 2d !!!

We need better approaches what we called *Frequent Itemset Generation Strategies*:

- Reduce the **number of candidates** (M):
- Complete search: M=2d – Use pruning techniques to reduce M
- Reduce the **number of transactions** (N)
- Reduce size of N as the size of itemset increases – Use a subsample of N transactions
- Reduce the **number of comparisons** (NM) – Use efficient data structures to store the candidates or transactions – No need to match every candidate against every transaction

Therefore, we have different algorithms.

Apriori

It is based on **Apriori principle**: *If an itemset is frequent, then all of its subsets must also be frequent*. Apriori principle holds due to the following property of the support measure (support of an itemset never exceeds the support of its subsets):

$$X, Y : (X \subseteq Y) \Rightarrow supp(X) \geq supp(Y)$$

This is known as the anti-monotone property of support. The method is:

– Let k=1 – Generate frequent itemsets of length 1 – Repeat until no new frequent itemsets are identified - Generate length (k+1) candidate itemsets from length k frequent itemsets - Prune candidate itemsets containing subsets of length k that are infrequent - Count the support of each candidate by scanning the DB - Eliminate candidates that are infrequent, leaving only those that are frequent - Reducing Number of Comparisons (based on candidate counting): - Scan the database of transactions to determine the support of each candidate itemset – To reduce the number of comparisons, store the candidates in a hash structure - Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets - Alternative Search Methods: Traversal of Itemset Lattice - General-to-specific vs Specific-to-general – Breadth-first vs Depth-first - Bottlenecks of Apriori.Candidate generation can result in huge candidate sets: - 10^4 frequent 1-itemset will generate 10^7 candidate 2- itemsets - To discover a frequent pattern of size 100, e.g., $a_1, a_2, \ldots, a_{100}$, one needs to generate $2^{100} \sim 10^{30}$ candidates. - Multiple scans of database: Needs $n+1$ scans, n is the length of the longest pattern

ECLAT:

While the purpose is the same, the method is different (based on for each item, store a list of transaction ids (tids): Vertical data layout TID-list.):

- Determine support of any k-itemset by intersecting tid- lists of two of its (k-1) subsets.
- 3 raversal approaches: top-down, bottom-up and hybrid
- Advantage: very fast support counting
- Disadvantage: intermediate tid-lists may become too large for memory

FP-growth:
Another method:

- Use a compressed representation of the database using an FP-tree
- Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent itemsets
- Benefits of the FP-tree Structure: Performance study shows FP-growth is an order of magnitude faster than Apriori, and is also faster than tree-projection
- Reasoning:
- No candidate generation, no candidate test – Use compact data structure – Eliminate repeated database scan – Basic operation is counting and FP-tree building

In sumary, the **Complexity of Association Mining** refers to:

- **Choice of minimum support threshold** – lowering support threshold results in more frequent itemsets – this may increase number of candidates and max length of frequent itemsets
- **Dimensionality (number of items) of the data set** – more space is needed to store support count of each item – if number of frequent items also increases, both computation and I/O costs may also increase
- **Size of database** – since Apriori makes multiple passes, run time of algorithm may increase with number of transactions
- **Average transaction width** – transaction width increases with denser data sets – This may increase max length of frequent itemsets and traversals of hash tree (number of subsets in a transaction increases with its width)

Association Analysis with R. We have several R libraries related to Association Analysis:

- Arules: A R extension package for mining association rules and frequent itemsets with R. It provides an easy to use and flexible platform for experiments and research.
- ArulesViz: Add-on for arules to visualize association rules.
- ArulesSequences: Add-on for arules to handle and mine frequent sequences.
- ArulesNBMiner: Mining NB-Frequent Itemsets and NB-Precise Rules.
- ArulesCBA: implements the CBA algorithm described in Liu, et al. 1998. It creates classifiers based on association rules and can then use those classifiers to classify incoming datasets.

The apriori principle can reduce the number of itemsets we need to examine. Put simply, the apriori principle states that if an itemset is infrequent, then all its subsets must also be infrequent.

- Computationally Expensive. Even though the apriori algorithm reduces the number of candidate itemsets to consider, this number could still be huge when store inventories are large or when the support threshold is low.
- Spurious Associations. Analysis of large inventories would involve more itemset configurations, and the support threshold might have to be lowered to detect certain associations. However, lowering the support threshold might also increase the number of spurious associations detected.

Interesting packages.

- RSarules: Random Sampling Association Rules from a Transaction Dataset
- arc: Association Rule Classification
- rCBA: CBA Classifier for R
- sbrl: Scalable Bayesian Rule Lists Model
- opusminer: OPUS Miner Algorithm for Filtered Top-k Association Discovery

References.

- Fast Algorithms for Mining Association Rules
- Association Analysis
- Arules Package
- Apriori Algorithm. Anita Wasilewska
- Algorithmic Features of Eclat
- Association Mining with R
- Frequent Itemset Mining Dataset Repository