Apriori algorithm

Seminar of Popular Algorithms in Data Mining and Machine Learning, TKK

Presentation 12.3.2008
Lauri Lahti
Association rules

- Techniques for data mining and knowledge discovery in databases

Five important algorithms in the development of association rules (Yilmaz et al., 2003):
  - AIS algorithm 1993
  - SETM algorithm 1995
  - Apriori, AprioriTid and AprioriHybrid 1994
Apriori algorithm

- Developed by Agrawal and Srikant 1994
- Innovative way to find association rules on large scale, allowing implication outcomes that consist of more than one item
- Based on minimum support threshold (already used in AIS algorithm)
- Three versions:
  - Apriori (basic version) faster in first iterations
  - AprioriTid faster in later iterations
  - AprioriHybrid can change from Apriori to AprioriTid after first iterations
Limitations of Apriori algorithm

- Needs several iterations of the data
- Uses a uniform minimum support threshold
- Difficulties to find rarely occurring events
- Alternative methods (other than Apriori) can address this by using a non-uniform minimum support threshold
- Some competing alternative approaches focus on partition and sampling
Phases of knowledge discovery

1) data selection
2) data cleansing
3) data enrichment (integration with additional resources)
4) data transformation or encoding
5) data mining
6) reporting and display (visualization) of the discovered knowledge

(Elmasri and Navathe, 2000)
Application of data mining

- Data mining can typically be used with transactional databases (for ex. in shopping cart analysis)
- Aim can be to build association rules about the shopping events
- Based on **item sets**, such as
  - \{milk, cocoa powder\} 2-itemset
  - \{milk, corn flakes, bread\} 3-itemset
Association rules

- Items that occur often together can be associated to each other
- These together occurring items form a frequent itemset
- Conclusions based on the frequent itemsets form association rules
- For ex. \{milk, cocoa powder\} can bring a rule \textit{cocoa powder} $\rightarrow$ \textit{milk}
Sets of database

- Transactional database $D$
- All products an itemset $I = \{i_1, i_2, \ldots, i_m\}$
- Unique shopping event $T \subseteq I$
- $T$ contains itemset $X$ iff $X \subseteq T$
- Based on itemsets $X$ and $Y$ an association rule can be $X \Rightarrow Y$
- It is required that $X \subset I$, $Y \subset I$ and $X \cap Y = \emptyset$
Properties of rules

- Types of item values: boolean, quantitative, categorical
- Dimensions of rules
  - 1D: $\text{buys(cocoa powder)} \Rightarrow \text{buys(milk)}$
  - 3D: $\text{age(X,’under 12’) \land gender(X,’male’) \Rightarrow buys(X,’comic book’)}$
- Latter one is an example of a profile association rule
- Intradimension rules, interdimension rules, hybrid-dimension rules (Han and Kamber, 2001)
- Concept hierarchies and multilevel association rules
Quality of rules

• Interestingness problem (Liu et al., 1999):
  – some generated rules can be self-evident
  – some marginal events can dominate
  – interesting events can be rarely occurring
• Need to estimate how interesting the rules are
• Subjective and objective measures
Subjective measures

- Often based on earlier user experiences and beliefs
- Unexpectedness: rules are interesting if they are unknown or contradict the existing knowledge (or expectations).
- Actionability: rules are interesting if users can get advantage by using them
- Weak and strong beliefs
Objective measures

• Based on threshold values controlled by the user
• Some typical measures (Han and Kamber, 2001):
  – simplicity
  – support (utility)
  – confidence (certainty)
Simplicity

• Focus on generating simple association rules
• Length of rule can be limited by user-defined threshold
• With smaller itemsets the interpretation of rules is more intuitive
• Unfortunately this can increase the amount of rules too much
• Quantitative values can be quantized (for ex. age groups)
Simplicity, example

- One association rule that holds association between cocoa powder and milk
  \[ \text{buys(cocoa powder)} \rightarrow \text{buys(bread,milk,salt)} \]
- More simple and intuitive might be
  \[ \text{buys(cocoa powder)} \rightarrow \text{buys(milk)} \]
Support (utility)

• Usefulness of a rule can be measured with a minimum support threshold.
• This parameter lets to measure how many events have such itemsets that match both sides of the implication in the association rule.
• Rules for events whose itemsets do not match both sides sufficiently often (defined by a threshold value) can be excluded.
Support (utility) (2)

- Database D consists of events $T_1, T_2, \ldots, T_m$, that is $D = \{T_1, T_2, \ldots, T_m\}$
- Let there be an itemset $X$ that is a subregion of event $T_k$, that is $X \subseteq T_k$
- The support can be defined as
  $$| \{T_k \in D \mid X \subseteq T_k\} |$$
  $$\text{sup}(X) = \frac{\text{-----------------------------}}{\text{|D|}}$$
- This relation compares number of events containing itemset $X$ to number of all events in database
Support (utility), example

• Let’s assume $D = \{(1,2,3), (2,3,4), (1,2,4), (1,2,5), (1,3,5)\}$
• The support for itemset $(1,2)$ is
  \[
  \left| \{T_k \in D \mid X \subseteq T_k\} \right| \sup((1,2)) = \frac{3}{5} = \frac{\left| \{T_k \in D \mid X \subseteq T_k\} \right|}{|D|}
  \]
• That is: relation of number of events containing itemset $(1,2)$ to number of all events in database
Confidence (certainty)

• Certainty of a rule can be measured with a threshold for confidence
• This parameter lets to measure how often an event’s itemset that matches the left side of the implication in the association rule also matches for the right side
• Rules for events whose itemsets do not match sufficiently often the right side while matching the left (defined by a threshold value) can be excluded
Confidence (certainty) (2)

• Database D consists of events $T_1, T_2, \ldots, T_m$, that is: $D = \{T_1, T_2, \ldots, T_m\}$
• Let there be a rule $X_a \rightarrow X_b$ so that itemsets $X_a$ and $X_b$ are subregions of event $T_k$, that is: $X_a \subseteq T_k \land X_b \subseteq T_k$
• Also let $X_a \cap X_b = \emptyset$
• The confidence can be defined as
  $$\text{conf}(X_a, X_b) = \frac{\text{sup}(X_a \cup X_b)}{\text{sup}(X_a)}$$
• This relation compares number of events containing both itemsets $X_a$ and $X_b$ to number of events containing an itemset $X_a$
Confidence (certainty), example

• Let’s assume $D = \{(1,2,3), (2,3,4), (1,2,4), (1,2,5), (1,3,5)\}$
• The confidence for rule $1 \Rightarrow 2$

\[
\text{conf}((1,2)) = \frac{\text{sup}(1 \cup 2)}{\text{sup}(1)} = \frac{3/5}{4/5} = \frac{3}{4}
\]
• That is: relation of number of events containing both itemsets $X_a$ and $X_b$ to number of events containing an itemset $X_a$
Support and confidence

- If confidence gets a value of 100% the rule is an **exact rule**
- Even if confidence reaches high values the rule is not useful unless the support value is high as well
- Rules that have both high confidence and support are called **strong rules**
- Some competing alternative approaches (other than Apriori) can generate useful rules even with low support values
Generating association rules

- Usually consists of two subproblems (Han and Kamber, 2001):
  1) Finding frequent itemsets whose occurrences exceed a predefined minimum support threshold
  2) Deriving association rules from those frequent itemsets (with the constrains of minimum confidence threshold)
- These two subproblems are solved iteratively until new rules no more emerge
- The second subproblem is quite straightforward and most of the research focus is on the first subproblem
Use of Apriori algorithm

- Initial information: transactional database D and user-defined numeric minimum support threshold $min_sup$
- Algorithm uses knowledge from previous iteration phase to produce frequent itemsets
- This is reflected in the Latin origin of the name that means "from what comes before"
Creating frequent sets

- Let’s define:
  - \( C_k \) as a candidate itemset of size \( k \)
  - \( L_k \) as a frequent itemset of size \( k \)

- Main steps of iteration are:
  1) Find frequent set \( L_{k-1} \)
  2) Join step: \( C_k \) is generated by joining \( L_{k-1} \) with itself (cartesian product \( L_{k-1} \times L_{k-1} \))
  3) Prune step (apriori property): Any \((k - 1)\) size itemset that is not frequent cannot be a subset of a frequent \( k \) size itemset, hence should be removed
  4) Frequent set \( L_k \) has been achieved
Creating frequent sets (2)

- Algorithm uses breadth-first search and a hash tree structure to make candidate itemsets efficiently.
- Then occurrence frequency for each candidate itemset is counted.
- Those candidate itemsets that have higher frequency than minimum support threshold are qualified to be frequent itemsets.
Apriori algorithm in pseudocode

$L_1 = \{\text{frequent items}\}$;  
for (k = 2; $L_{k-1} \neq \emptyset$; k++) do begin  
    $C_k = \text{candidates generated from } L_{k-1}$ (that is: cartesian product $L_{k-1} \times L_{k-1}$ and eliminating any $k-1$ size itemset that is not frequent);  
    for each transaction $t$ in database do  
        increment the count of all candidates in $C_k$ that are contained in $t$  
    $L_k = \text{candidates in } C_k \text{ with } min\_sup$  
end  
return $\bigcup_k L_k$;  

(www.cs.sunysb.edu/~cse634/lecture_notes/07apriori.pdf)
Apriori algorithm in pseudocode (2)

Apriori Pseudocode

\[ \text{Apriori}(T, \varepsilon) \]

\[ L_1 \leftarrow \{ \text{large 1-itemsets that appear in more than } \varepsilon \text{ transactions} \} \]
\[ k \leftarrow 2 \]

\[ \text{while } L_{k-1} \neq \emptyset \]
\[ C_k \leftarrow \text{Generate}(L_{k-1}) \]
\[ \text{for transactions } t \in T \]
\[ C_t \leftarrow \text{Subset}(C_k, t) \]
\[ \text{for candidates } c \in C_t \]
\[ \text{count}[c] \leftarrow \text{count}[c] + 1 \]
\[ L_k \leftarrow \{ c \in C_k | \text{count}[c] \geq \varepsilon \} \]
\[ k \leftarrow k + 1 \]

\[ \text{return } \bigcup L_k \]

← Join step and prune step (Wikipedia)