Taller de Sistemas Multiagentes

Association Rules

Agenda

- Introduction
- Concept
- Examples
- Algorithms
- Temporal Association Rules
- Software
  - WEKA
  - ARTool
- Conclusions

Introduction

- Data Mining is the process of finding interesting **trends** or **patterns** in large datasets in order to guide future decisions.
- Data Mining is the discovery of knowledge and useful information from the large amounts of data stored in databases.
- Related to **exploratory data analysis** (area of statistics) and **knowledge discovery** (area in artificial intelligence, machine learning).
- Data Mining is characterized by having **VERY LARGE** datasets.

Some real problems

- Associations between products bought in a store
  - milk
  - beers and diapers
- Deciding on product discounts
- Figuring out how to keep customer at lowest cost to company
- Personal recommendation page at Amazon
- Stock market trends

Association Rules: describing association relationships among the attributes in the set of relevant data.

Association Rules

A **market basket** is a collection of items bought by a single customer in a single customer transaction

Problem: Identify sets of items that are purchased together

Attempt to identify rules of the form

{pen} ➔ {ink}

Meaning: If a pen is bought in a transaction, it is likely that ink will also be bought

Association Rules

General form: L➔R, where L and R are sets of items

Support: The support for L➔R is the percentage of transactions containing all items from L and from R

Confidence: The confidence for L➔R are the percentage of transactions that contain R, from among the transactions that contain L

L is the antecedent of the rule and R its consequent

Understanding the measures

- If a rule has low support then it might have arisen by chance. There is not enough evidence to draw a conclusion.
- If a rule L➔R has low confidence then it is likely that there is no relationship between buying L and buying R

Note: L➔R and R➔L will always have the same support, but may have different confidence

Goal

Find association rules with high support and high confidence. Support and confidence values are specified by the user.

Remember: Finding such a rule does not mean that there **must** be a relationship between the left and right sides. A person must evaluate such rules by hand.

Example: {milk} ➔ {bread}

Formal statement of the problem

$I = \{ i_1 , i_2 , \ldots , i_m \}$ is a set of items

D is a set of transactions T

Each transaction T is a set of items (subset of I)

TID is a unique identifier that is associated with each transaction

The problem is to generate all *association rules* that have *support* and *confidence* greater than the user-specified minimum support and minimum confidence

Problem decomposition

The problem can be decomposed into two subproblems:

1. Find all sets of items (*itemsets*) that have support (number of transactions) greater than the minimum support (*large itemsets*).
2. Use the *large itemsets* to generate the desired rules.

   For each *large itemset l*, find all non-empty subsets, and for each subset *a* generate a rule a ==> (l-a) if its confidence is greater than the minimum confidence.

Notation

k-itemset: itemset containining k items

L_k: set of frequent (large) k-itemsets. Each member has to fields: i) itemset; ii) support count

C_k : set of candidate k-itemsets (potencially frequent. Each member has to fields: i) itemset; ii) support count

General Algorithm

1. In the first pass, the support of each individual item is counted, and the *large* ones are determined
2. In each subsequent pass, the *large* itemsets determined in the previous pass is used to generate new itemsets called *candidate* itemsets.
3. The support of each *candidate* itemset is counted, and the *large* ones are determined.
4. This process continues until no new *large* itemsets are found.

AIS Algorithm

Candidate itemsets are generated and counted on-the-fly as the database is scanned.

1. For each transaction, it is determined which of the *large* itemsets of the previous pass are contained in this transaction.
2. New *candidate* itemsets are generated by extending these *large* itemsets with other items in this transaction.

The disadvantage is that this results in unnecessarily generating and counting too many *candidate* itemsets that turn out to be *small.*

Example

Database

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

L_1

Itemset	Support
{1}	2
{2}	3
{3}	3
{5}	3

C_2

Itemset	Support
{1 3}*	2
{1 4}	1
{3 4}	1
{2 3}*	2
{2 5}*	3
{3 5}*	2
{1 2}	1
{1 5}	1

C_3

Itemset	Support
{1 3 4}	1
{2 3 5}*	2
{1 3 5}	1

SETM Algorithm

Candidate itemsets are generated on-the-fly as the database is scanned, but counted at the end of the pass.

1. New *candidate* itemsets are generated the same way as in AIS algorithm, but the TID of the generating transaction is saved with the *candidate* itemset in a sequential structure.
2. At the end of the pass, the support count of *candidate* itemsets is determined by aggregating this sequential structure

It has the same disadvantage of the AIS algorithm.

Another disadvantage is that for each *candidate* itemset, there are as many entries as its support value

Example

Database

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

L_1

Itemset	Support
{1}	2
{2}	3
{3}	3
{5}	3

C_2

Itemset	TID
{1 3}	100
{1 4}	100
{3 4}	100
{2 3}	200
{2 5}	200
{3 5}	200
{1 2}	300
{1 3}	300
{1 5}	300
{2 3}	300
{2 5}	300
{3 5}	300
{2 5}	400

C_3

Itemset	TID
{1 3 4}	100
{2 3 5}	200
{1 3 5}	300
{2 3 5}	300

Apriori Algorithm

Candidate itemsets are generated using only the *large* itemsets of the previous pass without considering the transactions in the database.

1. The *large* itemset of the previous pass is joined with itself to generate all itemsets whose size is higher by 1.
2. Each generated itemset, that has a subset which is not *large*, is deleted. The remaining itemsets are the *candidate* ones.

Example

Database

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

L_1

Itemset	Support
{1}	2
{2}	3
{3}	3
{5}	3

C_2

Itemset	Support
{1 2}	1
{1 3}*	2
{1 5}	1
{2 3}*	2
{2 5}*	3
{3 5}*	2

C_3

Itemset	Support
{2 3 5}*	2

{1 2 3}
{1 3 5}
{2 3 5}

Apriori TID

The database is not used at all for counting the support of *candidate* itemsets after the first pass.

1. The *candidate* itemsets are generated the same way as in Apriori algorithm.
2. Another set C' is generated of which each member has the TID of each transaction and the large itemsets present in this transaction. This set is used to count the support of each *candidate* itemset.

The advantage is that the number of entries in C' may be smaller than the number of transactions in the database, especially in the later passes.

Example

Database

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

L_1

Itemset	Support
{1}	2
{2}	3
{3}	3
{5}	3

C_2

Itemset	Support
{1 2}	1
{1 3}*	2
{1 5}	1
{2 3}*	2
{2 5}*	3
{3 5}*	2

C'_3

200	{2 3 5}
300	{2 3 5}

C'_2

100	{1 3}
200	{2 3}, {2 5}, {3 5}
300	{1 2}, {1 3}, {1 5}, {2 3}, {2 5}, {3 5}
400	{2 5}

C_3

Itemset	Support
{2 3 5}*	2

Apriori Hybrid

Performance Analysis demonstrated that:

- Apriori does better than AprioriTid in the earlier passes.
- AprioriTid does better than Apriori in the later passes.

Hence, a hybrid algorithm can be designed that uses Apriori in the initial passes and switches to AprioriTid when it expects that the set C' will fit in memory.

Apriori Algorithm

Apriori

- Given values *s* and *c*, find all association rules with support >= *s* and confidence >= *c*.
- Can be done in 2 steps.
- **Step 1:** Find **frequent itemsets**, i.e., sets of items that appear with support >= s
- **Step 2:** For each frequent itemset *F*, try all ways to partition *F* to L and R and check if the rule generated will have confidence >= c.

Finding Frequent Itemsets

The A Priori Property: Every subset of a frequent itemset must also be a frequent itemset.

This means that we can create frequent itemsets iteratively, by taking frequent itemsets of size *n*, and extending them to frequent itemsets of size *n+1*.

Finding Frequent Itemsets (2)

Freq = {}
scan all transactions once and add to Freq the items that
    have support > *s*
k = 1
repeat
    foreach l_k in Freq with *k* items
        generate all itemsets l_{k+1} with *k+1* items,
            such that l_k is contained in l_{k+1}
        scan all transactions once and add to Freq the
            k+1-itemsets that have support > *s*
        k++
until no new frequent itemsets are found

Example Purchase Table

transid	item
111	pen
111	ink
111	diary
111	soap
112	pen
112	ink
112	diary

transid	item
113	pen
113	diary
114	pen
114	ink
114	soap
114	tissues

Example

Run the algorithm with support = 0.7

- Level 1: Find frequent itemsets:

  {pen}, {ink}, {diary}

- Level 2: Check each of:

  {pen, ink}, {pen, diary}, ~~{pen, soap}~~,
  ~~{pen, tissues}~~, {ink, diary}, ~~{ink, soap}~~,
  ~~{ink, tissues}~~, ~~{diary, soap}~~, ~~{diary, tissues}~~

  The frequent itemsets are:

  {pen, ink}, {pen, diary}

Example

- Level 3: Check each of:

~~{pen, ink, diary}~~, ~~{pen, ink, soap}~~,

~~{pen, ink, tissues}~~, ~~{pen, diary, soap}~~,

~~{pen, diary, tissues}~~

No frequent itemsets!

To summarize, the frequent itemsets were:

{pen}, {ink}, {diary}

{pen, ink}, {pen, diary}

Refinements

- Note that we checked if {pen, tissues} is a frequent itemset even though we already knew that {tissues} is not a frequent itemset. **Refinement:** Try to extend frequent itemsets in a fashion that will ensure that all their subsets are frequent itemsets

- Scanning the transactions can be expensive if the table does not fit in main memory. **Refinement:** Find rules over a sample of the database that fits in memory

Deriving Association Rules

foreach frequent itemset *I*
    foreach partition of *I* to two sets L, R
        generate a candidate rule L→R
foreach transaction T in the database
    foreach candidate rule L→R
        if L in T then
            lnum(L→R)++
            if R in T then rnum(L→R)++
return all rules L→R with
    rnum(L→R)/lnum(L→R) > *c*

Several Refinements...

- Improved algorithms for the discovery of large itemsets
- Improved algorithms for the discovery of generalized and quantitative association rules
- New measures for other types of data (different from basket data)
- Temporal considerations

Temporal Association Rules

- In the calculus of support the denominator represents the total number of transactions in a time period when the involved items may have not existed. For example, a supermarket may start selling new products.
- Some itemsets may become obsolete, and thus, some rules may become outdated.

Solution

Consider the lifetime of an item, this being the period between the first and the last time the items appears in transactions in the database.

The basic idea is to limit the search for frequent itemsets to the lifetime of the itemset's members.

Each rule has an associated time frame, corresponding to the lifetime of the items participating in the rule.

Definitions

The lifespan (lifetime) of an item A_i is given by an interval $[t_1, t_2]$ with $t_1 \leq t_2$

The lifespan of a k-itemset X with k>1 is [t, t'] where

t=max{t_1 / $[t_1,t_2]$ is the lifespan of an item A_i in X}

t'=min{t_2 / $[t_1,t_2]$ is the lifespan of an item A_i in X}

Determining frequent itemsets

Compute the ratio between the number of transactions that contain the itemset and the number of transactions in the database such that their valid time is included in the itemset's lifespan.

Temporal Support: the amplitude of the lifespan of an itemset

Temporal Association Rule

A temporal association rule is an expression of the form X➔Y [t1, t2], where [t1, t2] is a time frame corresponding to the lifespan of $X \cup Y$ expressed in a granularity determined by the user.

A temporal association rule has three factors associated with it: support, temporal support and confidence

(support and confidence have to be redefined in this model!!)

Cyclic Association Rules

Association rules that exist in certain time intervals and thus display cyclic variations over time.

Example: If we compute association rules over monthly sales data, we may observe seasonal variation where certain rules are true at approximately the same month each year. Similarly, association rules can also display regular hourly, daily, weekly, etc. Variation that is cyclical in nature.

[Ozden98] Approach

It is assumed that transactional data is timestamped and that time intervals are specified by te user to divide the data into disjoint segments.

Cyclical Association Rule: if the ruel has the minimum confidence and support at regular time intervals

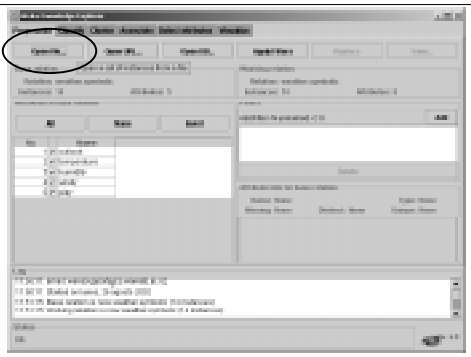
Mining Association Rules Problem: the generation of all the cycles of an association rule

Software: WEKA

Machine Learning Software in Java

http://www.cs.waikato.ac.nz/~ml/weka/

WEKA

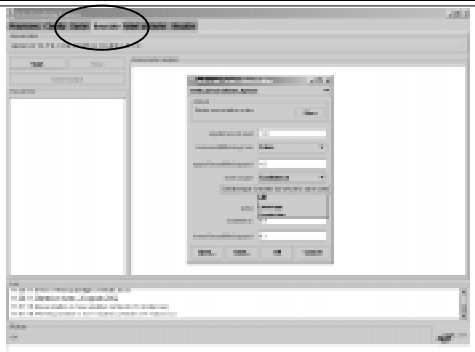


WEKA: Input file

```
@relation weather.symbolic
@attribute outlook {sunny, overcast, rainy}
@attribute temperature {hot, mild, cool}
@attribute humidity {high, normal}
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,hot,high,FALSE,no
sunny,hot,high,TRUE,no
overcast,hot,high,FALSE,yes
rainy,mild,high,FALSE,yes
rainy,cool,normal,FALSE,yes
rainy,cool,normal,TRUE,no
overcast,cool,normal,TRUE,yes
sunny,mild,high,FALSE,no
sunny,cool,normal,FALSE,yes
rainy,mild,normal,FALSE,yes
sunny,mild,normal,TRUE,yes
overcast,mild,high,TRUE,yes
overcast,hot,normal,FALSE,yes
rainy,mild,high,TRUE,no
```

WEKA



WEKA

WEKA: Generated Rules

Minimum support: 0.05
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12
Size of set of large itemsets L(2): 47
Size of set of large itemsets L(3): 39
Size of set of large itemsets L(4): 6

Best rules found:
1. humidity=normal windy=FALSE 4 ==> play=yes 4 conf:(1)
2. temperature=cool 4 ==> humidity=normal 4 conf:(1)
3. outlook=overcast 4 ==> play=yes 4 conf:(1)
4. temperature=cool play=yes 3 ==> humidity=normal 3 conf:(1)
5. outlook=rainy windy=FALSE 3 ==> play=yes 3 conf:(1)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3 conf:(1)
7. outlook=sunny humidity=high 3 ==> play=no 3 conf:(1)
8. outlook=sunny play=no 3 ==> humidity=high 3 conf:(1)
9. temperature=cool windy=FALSE 2 ==> humidity=normal play=yes 2 conf:(1)
10. temperature=cool humidity=normal windy=FALSE 2 ==> play=yes 2 conf:(1)

Software: ARTool

ARtool represents a collection of algorithms and tools for the mining of association rules in binary databases.

http://www.cs.umb.edu/~laur/ARTool

ARTool

Different algorithms for generating frequent itemsets:

- Apriori
- Closure
- ClosureOpt
- FPGrowth

Different algorithms for rules generation:

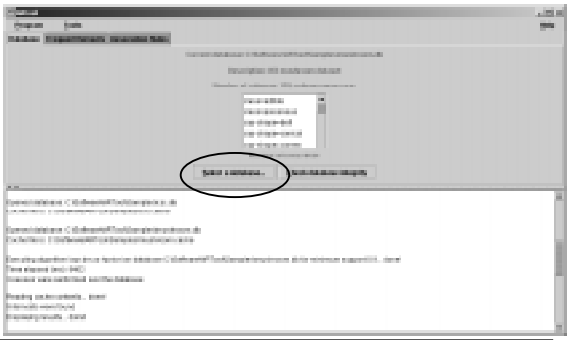
- Apriori Rules
- Cover Rules
- Cover Rules Opt.

ARTool: Rules Generation

The user can indicate:

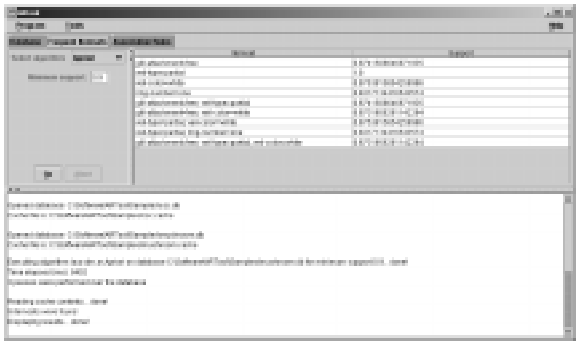
- Attributes the antecedent must contain
- Attributes the consequent must contain
- Items to ignore
- Maximum antecedent size
- Minimum consequent size

ARTool

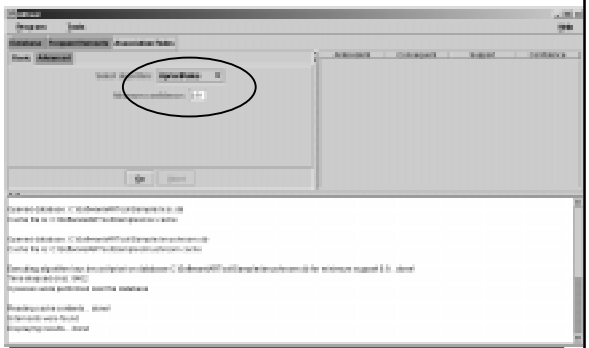


ARTool

ARTool



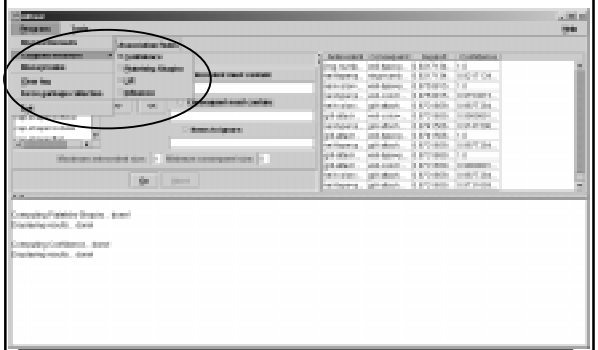
ARTool



ARTool

ARTool

ARTool



Bibliography

1. R. Agrawal, T. Imielinski, A. Swami: "Mining Associations between Sets of Items in Massive Databases", Proc. of the ACM-SIGMOD 1993 Int'l Conference on Management of Data, Washington D.C., May 1993, 207-216.

2. R. Agrawal, R. Srikant: "Fast Algorithms for Mining Association Rules", Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, Sept. 1994. Expanded version available as IBM Research Report RJ9839, June 1994

3. Yingjiu Li, Peng Ning, X. Sean Wang, Sushil Jajodia, ``Discovering Calendar-based Temporal Association Rules,'' (Full version) in Proceedings of the 8th International Symposium on Temporal Representation and Reasoning (TIME 01), pages 111-118, Italy, June 2001.