Mining Multi-level Association Rules in Large Databases

IEEE Transactions on Knowledge and Data Engineering, 1999

Jiawei Han [1] Yongjian Fu [2]
Presentation by Ethan Eldridge [3]

[1]Simon Fraser University, British Columbia

[2]University of Missouri-Rolla, Missouri

[3]University of Vermont, Vermont

March 24, 2013

Overview

- What is MLAR?

Overview

- What is MLAR?
- Concepts behind the Method

Overview

- What is MLAR?
- Concepts behind the Method
- The Method For Mining Multi-Level Association Rules

Overview

- What is MLAR?
- Concepts behind the Method
- The Method For Mining Multi-Level Association Rules
- Pruning Rules for interestingness

Overview

- What is MLAR?
- Concepts behind the Method
- The Method For Mining Multi-Level Association Rules
- Pruning Rules for interestingness
- Variations and Enhancements

Overview

- What is MLAR?
- Concepts behind the Method
- The Method For Mining Multi-Level Association Rules
- Pruning Rules for interestingness
- Variations and Enhancements
- Conclusions and Future Work

Overview

- What is MLAR?
- Concepts behind the Method
- The Method For Mining Multi-Level Association Rules
- Pruning Rules for interestingness
- Variations and Enhancements
- Conclusions and Future Work
- Exam Questions

What is MLAR Overview

- MLAR stands for Multi-Level Association Rule
- Motivation for MLDM*
- Requirements for MLDM*

*MLDM: Multi Level Data Mining

Whats different?

What's different between each of these rules?

- **Rule A** : 70% of customers who bought diapers also bought beer
- **Rule B** : 45% of customers who bought clothe diapers also bought dark beer
- **Rule C** : 35% of customers who bought pampers also bought Samuel Adams

Whats different?

This process is called Drilling down

- **Rule A** applies at a *generic* higher level of abstraction **(product)**
- **Rule B** applies at a *more specific* level of abstraction **(category)**
- **Rule C** applies at the *lowest* level of abstraction **(brand)**

Why Drill Down?

- The information is more valuable
- Different levels of associations enable different strategies for marketing

Why Drill Down?

- Remove uninteresting rules
- toy $\implies$ milk is not interesting (coincidence)

In a Nutshell

We need to be able to create *interesting* and *valuable* rules
What are the pre-requisites for MLDM?

2 Things

To do MLDM we need 2 things:

1. Data at Multiple Levels of Abstraction
2. An efficient method for Multi-Level Rule Mining (This Papers work)

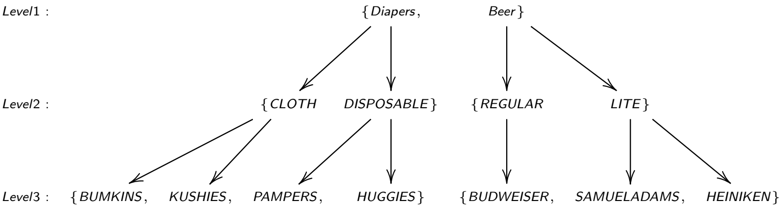
Data at Multiple Levels of Abstraction

We can find Data:

- Implicitly stored in a database
- Provided by Experts or Users
- Data Warehousing and OLAP (Online Analytics Processing)

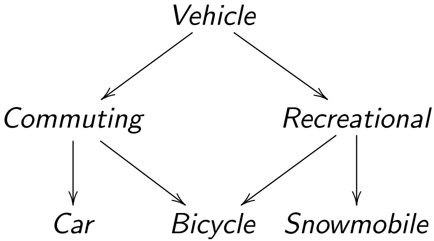
Data at Multiple Levels of Abstraction

Concept Taxonomies in Databases might look like:
Generalization to Specialization:
(is-a relationship)



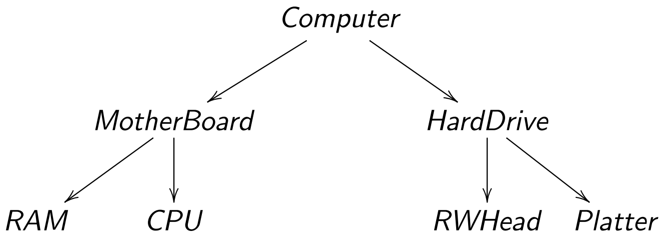
Data at Multiple Levels of Abstraction

Generalization to Specilization with Multiple Inheritance:



Data at Multiple Levels of Abstraction

Whole-Part Hierachies
(is-part-of, has-part)



What about Apriori?

What can we try to find rules for these multi-level datasets?

- We can apply Apriori to each level

What about Apriori?

What can we try to find rules for these multi-level datasets?

- We can apply Apriori to each level
- Problems?

What about Apriori?

Problems with Apriori

- Higher levels of abstraction have higher support
- Lower levels have lower support
- Optimum minimal support for all levels?
- Min. Support too high: not enough itemsets in low levels
- Min. Support too low: too many uninteresting rules

An efficient method for Multi-Level Rule Mining

Possible Solutions:

- Different minimal support at each level
- Different minimal confidence at each level
- Reduce minimal support as level increases

Progressive Deepening Method

The authors propose a Progressive Deepening method which

- Makes some assumptions about data
- Introduces work around for those who have issues with the assumption
- Is significantly different from other research

Main Assumption

Main Assumptions:

- Explore only descendants of frequent items
- If an item is non frequent at one level, none of it's descendants figure in future analysis

What are some problems with this?

The Problem and the Work Around

Will this eliminate possible interesting rules for itemsets whose ancestors were infrequent?

- Work around
  1. 2 Min. Support values. One absolute cutoff point (normal minisup), one for allowing frequent items to lower levels, called the *Level Passage Threshold LPH*
  2. The LPH can be adjusted by user to allow descendents of sub-frequent items

How is this different?

- Other research uses same minisup accross all levels
- Problems with this?

How is this different?

- Other research uses same minisup accross all levels
- Problems with this?
- As said before:
    - Min. Support too high: not enough itemsets in low levels
    - Min. Support too low: too many uninteresting rules

This Study...

This Study does a few things differently:

- Uses different minisup values at different levels of the hierachy
- Analyzes different optimization techniques
- Proposes extensions to best methods found
- Implements formal interestingness measures

Data Format and Definitions

Each database contains:

1. Item dataset containing item description { A_i,Description }
2. A transaction dataset T containing set of transactions { tid, $\{A_p \ldots A_q$ } } *

*tid is transaction identifier (key)

Data Format and Definitions

- A *pattern* or *itemset* A is one item A_i or a set of conjunctive items $A_i \wedge \ldots \wedge A_j$
- The support of a pattern is the number of transactions that contain A vs the total number of transactions, denoted $\sigma(A|S)$
- Confidence ϕ of a rule $A \Rightarrow B \in S$ is denoted
  $\phi(A \Rightarrow B) = \sigma(A \wedge B)/\sigma(A)$
  $\phi(A \Rightarrow B)$ is the conditional probability of B occuring given A has

Data Format and Definitions

- 2 Thresholds at each level:
  1. minisup (σ')
  2. miniconf (ϕ')

Data Format and Definitions

GID encoding:

- 112 means Level 1 item 1, level 2 item 1, level 3 item 2.
- For example Hood 1% Milk, might encode to 243 if Milk is the second category of the first level, 1% is the fourth type of milk of the second level, and Hood is the third brand possible brand for an item.

Frequency and Strong Rules

A pattern A is frequent in set S if:

- $\sigma(A) \geq \sigma'$ (Support of A is no less than minimum support for that level)

A rule $A \Rightarrow B$ in S is strong if:

- each ancestor of every item in A and B is frequent at its corresponding level
- $A \wedge B$ is frequent at the current level
- $\phi(A \Rightarrow B) \geq \phi'$ (The confidence of $A \Rightarrow B$ is no less than the minimum confidence at that level)

What's the point?

So why do we want Strong rules and frequent items?

What's the point?

- This ensures that the patterns examined at the lower levels arise from itemsets that have a high support at higher levels
- Strong rules help filter out 'uninteresting' rules

High Level View of the Algorithm

At level 1

1. get frequent itemsets
2. create filtered virtual table

For each other level

1. Generate candidates (Apriori) from frequent itemsets
2. for each transaction
    1. Get subsets and calculate support for generated candidates
    2. Pass to next level if they meet criteria
3. Union all found subsets that have met criteria

Repeat until desired level reached or empty frequent 1-itemset is generated

So how does it work?

Example query: Find multiple level strong association rules for purchase patterns related to *category*, *content*, and *brand*

- Retrieve relevant data from the database relations

id	category	content	brand	price
101	milk	2%	Hood	3.99
⋮	⋮	⋮	⋮	⋮

- Merge into generalized table with id's replaced with id set

gid	id	category	content	brand
112	{ 101,102,114 }	milk	2%	Hood
⋮	{ ..., ..., ... }	⋮	⋮	⋮

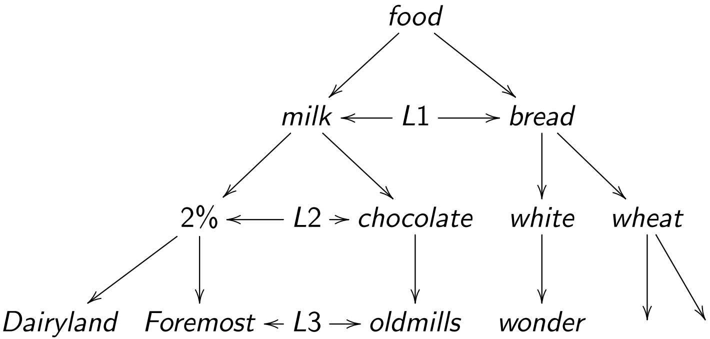
So how does it work?

Example query: Find multiple level strong association rules for purchase patterns related to *category*, *content*, and *brand*

- Find frequent patterns and strong rules at highest level. 1-item, 2-item,k-item itemsets may be discovered of the form {bread, vegetable, milk,}
- At the next level the process is repeated but the itemsets will be more specific ex: {2% milk, lettuce, white bread}
- Repeat previous 2 steps until all levels until no more frequent patterns

Taxonomy For Exercise

L1 = Level 1, L2 = Level 2, L3 = Level 3



Data Set For Exercise

Let's do an example to fully understand the algorithm!

Data Set For Exercise

Table 1: Sales Transaction Table

Trans id	Bar_code_set
351428	{ 17325, 92108, . . . }
653234	{23423, 56432, . . . }

Table 2: sales_item (Description) Relation

bar_code	category	brand	content	size	price
17325	milk	Foremost	2%	1 Gal	3.31
. . .	. . .	. . .	. . .	. . .	. . .

Table 3: Generalized sales_item Description Table

GID	Barcode_set	Category	Content	brand
112	{ 17325, 31414, 91265, . . .}	Milk	2%	Foremost
. . .	. . .	. . .	. . .	. . .

Preprocessing

Before running the algorithm we encode the data into the following table using the GID and Transaction Id's.

TID	GID encoded Items
T1	{ 111, 121, 211, 221 }
T2	{ 111, 211, 222, 323 }
T3	{ 112, 122, 222, 323 }
T4	{ 111, 121 }
T5	{ 111, 122, 211, 221, 413 }
T6	{ 211, 323, 524 }

Algorithm Step 1

Created Level 1 Item 1 Table with MiniSup = 4

Table T[1] (initial set)

TID	GID encoded Items
T1	{ 111, 121, 211, 221 }
T2	{ 111, 211, 222, 323 }
T3	{ 112, 122, 222, 323 }
T4	{ 111, 121 }
T5	{ 111, 122, 211, 221, 413 }
T6	{ 211, 323, 524 }

L(1,1) (Level 1 1-itemsets

{1**}	5
{2**}	5

We can see that 5 transactions support both1-itemsets

Algorithm Step 2

L(1,1)

itemset	support
{1**}	5
{2**}	5

L(1,2)

itemset	support
{1**, 2**}	4

Filtered T[2]

TID	Items
T1	{111, 121, 211, 221}
T2	{ 111, 211, 222 }
T3	{ 112, 122, 221 }
T4	{ 111, 121 }
T5	{ 111, 122, 211, 221 }
T6	{ 211 }

The new table is created by filtering the old with respect to L(1,1)

Algorithm Step 3

Level 2 MiniSup = 3

Filtered T[2]

TID	Items
T1	{111, 121, 211, 221}
T2	{ 111, 211, 222 }
T3	{ 112, 122, 221 }
T4	{ 111, 121 }
T5	{ 111, 122, 211, 221 }
T6	{ 211 }

L(2,1)

Itemset	Support
{ 11* }	5
{ 12* }	4
{ 21* }	4
{22* }	4

L(2,2)

Itemset	Support
{ 11* , 12* }	4
{ 11*, 21* }	3
{ 11*, 22* }	4
{ 12*, 22* }	3
{ 21*, 22* }	3

L(2,3)

Itemset	Support
{ 11*, 12*, 22* }	3
{ 11*, 21*, 22* }	3

Algorithm Step 3

Level 3 MiniSup = 3

Filtered T[2]

TID	Items
T1	{111, 121, 211, 221}
T2	{ 111, 211, 222 }
T3	{ 112, 122, 221 }
T4	{ 111, 121 }
T5	{ 111, 122, 211, 221 }
T6	{ 211 }

L(3,1)

Itemset	Support
{ 111 }	4
{ 211 }	4
{ 221 }	3

L(3,2)

Itemset	Support
{111, 211}	3

What counts as interesting?

The paper defines two filters for interesting rules

1. Removal of Redundant Rules
2. Removal of Unneccesary Rules

Redundant Rules

A rule is redundant if

- it can be derived or computed from a higher level rule *
- we assume a relatively uniform distribution

*every item is the same or a higher level item

Redundant Rules

Example:

- $R = milk \Rightarrow bread$ with $\sigma(R) = 12\%$, $\phi(R) = 85\%$
- $R' = chocolate \quad milk \Rightarrow bread$ with $\sigma(R') = 1\%$ and $\phi(R') = 84\%$

R' might not be interesting if only 8% of all milk is chocolate.

Redundant Rules

Formal Definition of Redundant Rule:

A rule $R, A_i \wedge A_2 \wedge \ldots A_n \Rightarrow B_1 \wedge B_2 \ldots B_m$ *is redundant if there is some rule* $R', A'_1 \wedge A'_2, \ldots A'_n \Rightarrow B'_1 \wedge B'_2 \ldots B'_m$ *where every item in* R *is a descendant or the same in* R' *and* $\phi(R) \in [exp(\phi(R) - \alpha, exp(\phi(R)) + \alpha]$ *where* $exp(\phi(R)) = (\sigma(B_1)/\sigma(B'_1)x \cdots x(\sigma(B_n)/\sigma(B'_n))x\phi(R')$ *and* α *is a user defined constant.*

In english: If the confidence of a rule falls within a certain range and its items are shared in other rules, it is Redundant.

Redundant Rules

Applying Redundant Rule reduction cuts Strong Rules by 40-70%

Unnecessary Rules

A rule is unncessary if

- it does not differ significanlty from a simpler rule

Unnecessary Rules

Example:

- 80% of Customers who buy milk also buy bread
- 80% of Customers who buy milk and butter also buy bread

The extra information doesn't really tell us anything new.

Unnecessary Rules

Formal Definition: A rule $R, A \wedge C \Rightarrow B$ is unnecessary if there is a rule $R', A \Rightarrow B$ and $\phi(R) \in [\phi(R') - \beta, \phi(R') + \beta]$ where β is a user defined constant. A, B, C are itemsets and C is not empty.

Unnecessary Rules

Applying Unnecessary rule reduction cuts Strong rules by 20-50%

List of Variations

Variations on ML_T2L1 (Original Algorithm)

1. ML_T1LA
2. ML_TML1
3. ML_T2LA

(There are also two extensions on T2LA and T2L1 but they are beyond the scope of this presentation)

ML_T1LA

Variations:

- Uses only one encoded table $T[1]$
- Computes support for all levels of hiearchy with one scan

Pros? Cons?

ML_T1LA

Pros:

- Avoids generation of new transaction table
- Total number of scans = k for largest k-itemset

ML_T1LA

Cons:

- Scanning $T[1]$ scans infrequent items which is wasteful
- Large space for all subsets, page swapping possibly needed

ML_TML1

Varation:

- Instead of just T[1] and T[2]. We have T[1],T[2],... t[max l + 1]
- Generate each T[l] in parallel.

ML_TML1

Pros:

- Newly filtered tables will cut down amount of processing at each level if some data is frequent

ML_TML1

Cons:

- Small number of items filtered out at processing each level. Will make the algorithm run for a long time

ML_T2LA

Variation:

- Uses T[1] and T[2] (like original) but optimizes like ML_T1LA
- Calculates support with one scan of T[2]
- Scans k-1 times per generation of all k-itemsets

ML_T2LA

Pros:

- Scans T[2] k-1 times total for generation of all k-itemsets.
- Potentially efficient due to filtering of T[1]

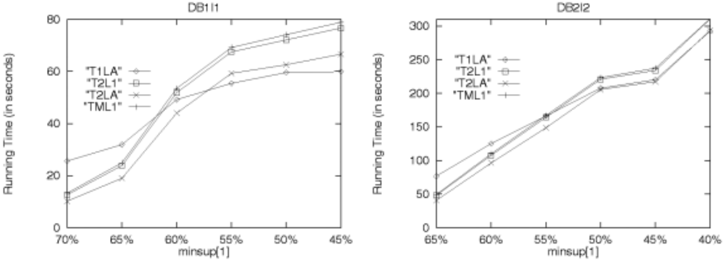
ML_T2LA

Cons:

- The efficiency of this algorithm depends on the data being able to be filtered

Typo In Paper

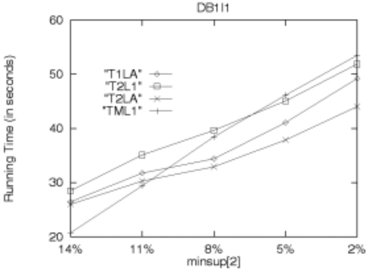
According to the book, ML_T1LA, is the best or second best. However I believe to be a typo, the charts indicate that ML_T2LA is the best most of the time.



Peformance with respect to top-level minimum support.

Typo In Paper

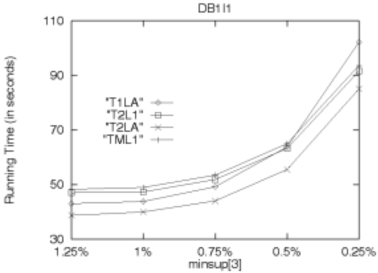
According to the book, ML_T1LA, is the best or second best. However I believe to be a typo, the charts indicate that ML_T2LA is the best most of the time.



Minimum support at level 2.

Typo In Paper

According to the book, ML_T1LA, is the best or second best. However I believe to be a typo, the charts indicate that ML_T2LA is the best most of the time.



Minimum support at level 3.

Conclusions

The authors

- Extended association rules from single to multi level
- A top-down progressive deepening technique was developed for finding such rules
- Filtering of uninteresting rules was formally defined in two ways.

Future work

The future may hold

- Developing efficient algorithms for mining multi-level sequential patterns
- Mining multiple level correlations in databases
- Cross level associations
- More interestingness measures of rules

Question 1

- What is a major drawback to multiple level data mining using the same minimal support at all levels of a concept hierarchy?
- Large Support exists at higher levels of abstraction, and smaller support at lower levels. To find strong rules in the deeper levels we must relax support at higher levels, which can result in uninteresting rules at higher levels. It is hard to determine an optimal minimal support for all levels.

Question 2

- What are the 2 pre-requisites to performing multiple-level association rule mining?
- To explore multiple level association rule mining one needs to provide
  1. Data at multiple levels of abstraction
  2. Efficient methods for multiple level rule mining

Question 3

- Give an example of a multiple level association rule
- At a high level in the hierarchy one may have a general rule like 80% of people who buy cereal also buy milk, at a lower level the rule becomes more specific like 25% of people who buy cheerios buy almond milk from silk.