Cse634

DATA MINING

SHORT MID 2 REVIEW

Professor Anita Wasilewska
Computer Science Department
Stony Brook University

Association and Genetic Algorithms

- Describe the Apriori Algorithm and Association Analysis
- Describe **all types** of Association Rules and **methods** of obtaining them
- Discuss types of Association Analysis applications
- Describe classification by Association and compare it with the classification by Decision Trees or Neural Network
- Discuss types of Classification by Association applications

Association and Genetic Algorithms

- Describe principles of Genetic Algorithms
- Give examples of chromosomes encoding
- Describe **GA** operators and parameters
- Describe the role of fitness function
- Describe **GA** Reproduction Cycle
- Discuss types of **GA** applications
- Compare classification by **GA** with NN and DT classifications

The Apriori Algorithm

- Apriori Algorithm

  The algorithm iteratively **finds** frequent itemsets with cardinality from 1 to *k* (k-itemset)

Key Concepts:

- Frequent Itemsets
- Apriori Property

- As the next step in the **Apriori Process**

  we use the frequent itemsets to **generate**

  association rules

The Apriori Algorithm: Basics

Key Concepts:

Frequent Itemsets

- The sets of item which has minimum support (denoted by L_i for i^{th}-Itemset)

- Apriori Property
- Any subset of frequent itemset must be frequent

- Join Operation
- To find L_k , a set of candidate k-itemsets is generated by **joining** L_{k-1} with itself.

The Apriori Algorithm : Pseudo code

- Join Step: C_k is generated by **joining** L_{k-1} with itself
- Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset

- Pseudo-code:

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

```
L_1 = {frequent items};
for (k = 1; L_k !=∅; k++) do begin
    C_k+1 = candidates generated from L_k;
   for each transaction t in database do
            increment the count of all candidates in C_k+1
         that are contained in t
   L_k+1  = candidates in C_k+1 with min_support
   end
return ∪_k L_k;
```

Methods to Improve Apriori's Efficiency

- Hash-based itemset counting:
- A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent.

- Transaction reduction:
- A transaction that does not contain any frequent k-itemset is useless in subsequent scans

- Partitioning:
- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB.

Methods to Improve Apriori's Efficiency

- Sampling:
- mining on a subset of given data,
- lower support threshold
- add a method to determine the completeness

- Dynamic itemset counting:
- add new candidate itemsets **only** when all of their subsets are estimated to be frequent

Mining Frequent Patterns Without Candidate Generation

- **Compress** a large database into a compact,
- Frequent-Pattern tree (FP-tree) structure
  - highly condensed, but complete for frequent pattern mining
  - avoid costly database scans
- **Develop** an efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - **Avoid** candidate generation: sub-database test only!

Why Frequent Pattern Growth Method?

- **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

Association Analysis: Basic Concepts

- **Given: a** database of transactions, where each transaction is a list of items
- **Find:** all rules that associate the presence of one set of items with that of another set of items
- **Example**

  98% of people who purchase tires and auto accessories also get automotive services done

Appriori Process: Rules Generation

- **Appriori Algorithm stops** after the **First Step**
- **Second Step** in the **Appriori Proces** (item-sets generation AND rules generation) is the **rules generation:**
- We calculate, from the frequent item-sets a set of the **strong rules**
- **Strong rules:** rules with at least minimum support (low) and minimum confidence (high)
- **Apriori Process** is then **finished** .

Apriori Process
Rules Generation

- The Apriori Process problem is:
- How do we form the association rules

  A =>B from the frequent item sets?

- **Remember:** A, B are disjoint subsets of the set **I** of items in general, and of the set 2-frequent, 3-frequent item sets ..... etc, ... as generated by the Apriori Algorithm

Association Rules

- Rule **general** form:

  "Body $\rightarrow$ Head [support, confidence]"

  Rule **Predicate** form:

  buys(x, "diapers") $\rightarrow$ buys(x, "beer")

  [0.5%, 60%]

  major(x, "CS") ^ takes(x, "DB") $\rightarrow$ grade(x, "A")
  [1%, 75%]

  Rule **Attribute** form:

  Diapers $\rightarrow$ beer [1%, 75%]

Apriori Process

- Given a data base D of TRANSACTIONS
- **Goal:** Find Association Rules
- We follow the **steps**
- **STEP 1**: **Count** occurrences of items in **D**
- **STEP2:** Fix Minimum support (usually **low**)
- **STEP 3:** Calculate all frequent k-item sets
- **STEP 4: STOP** when **there is no more frequent item** sets
- This is the **END** of Apriori Algorithm **phase**

Apriori Process

- **Rules Generation phase**
- **STEP 5:** Fix the **minimum** confidence (usually **high**)
- **STEP 6:** Generate **strong rules** (support >min support and confidence> min confidence)
- **END** of rules generation phase
- **END** of the **Apriori Process**

Generate requent i-item Sets

- **How to** generate **all** frequent i-item sets
- FIRST: Calculate frequent 1-item sets
- SECOND: use the frequent (i-1)-item sets to **generate** all **i-**item set **candidates**
- THIRD : use **Apriori Principle** to **PRUNE** the candidates set
- FOUR: **Evaluate** the count of the **pruned set**

  and **list** the frequent i-item sets

FIVE: **repeat** the procedure and **STOP** when there is no more frequent item sets

Mining Association Rules in Large Databases

- Mining single-dimensional association rules from transactional databases
- Mining multi-dimensional association rules from transactional databases and data bases (warehouse)
- Mining multilevel association rules from transactional databases

Single and Multidimensional Rules

- **Single-dimensional rules**:
- buys(X, "milk") $\Rightarrow$ buys(X, "bread")
- **Multi-dimensional rules**: Involve 2 or more dimensions or predicates
  - **Inter-dimension association rules (*no repeated predicates*)**
    - age(X,"19-25") $\wedge$ occupation(X,"student") $\Rightarrow$ buys(X,"coke")

Multi-Dimensional Association Rules

Hybrid-dimension association rules (repeated predicates)

- age(X,"19-25") $\wedge$ buys(X, "popcorn") $\Rightarrow$ buys(X, "coke")

- **Categorical (qualitative) Attributes**
  – finite number of possible values, no ordering among values
- **Quantitative Attributes**
  – numeric, implicit ordering among values

Mining Multi-Dimensional Association

- Categorical Attributes:
- finite number of possible values, no ordering among values

- Quantitative Attributes:
- Numeric, implicit ordering among values

- Discretization, clustering:
- Numeric values are replaced by ranges or names

Mining Multi-Level Associations

- A top_down, progressive deepening approach:
  - **First find high-level strong rules**:

    milk → bread [20%, 60%]
  - **Then find their lower-level "weaker" rules**:

    2% milk → wheat bread [6%, 50%]
- **Variations at mining multiple-level association rules**.
  - **Level-crossed association rules:**

    *2% milk → **Wonder** wheat bread*
  - **Association rules with multiple, alternative hierarchies:**

    *2% milk → **Wonder** bread*

Problem: Classification by Association

1. Use TRAIN data to find the set of classification rules using the Apriori Algorithm
   2. **Test** the rules with the TEST Data
   Use 2 different testing Method of your choice and compare the results

TRAIN DATA

Record	A1	A2	C
1	1	1	1
2	0	0	0
3	0	1	0
4	0	0	0
5	1	1	1
6	1	1	0
7	0	0	0
8	1	0	1

Transactional Data and Support calculations

	I1 (A1 =0)	I2(A1 = 1)	I3(A2 = 0)	I4(A2= 1)	I5(C=0)	I6(C=1)
1		+		+		+
2	+		+		+	
3	+			+	+	
4	+		+		+	
5		+		+		+
6		+		+	+	
7	+		+		+	
8		+	+			+
Count	4	4	4	4	5	3

Let the minimum support count = 3

L1:

Item sçet	Support Count
I1	4
I2	4
I3	4
I4	4
I5	5
I6	3

Candidate two item sets :

Item Set	Support Count
1,2	0
1,3	3
1,4	1
1,5	4
1,6	0
2,3	1
2,4	3
2,5	1
2,6	0
3,4	3
3,5	1
3,6	2
4,5	2
4,6	0

Classification by Association

Frequent 2 item set :

Item Set	Support Count
1,3	3
1,5	4
2,4	3
2,6	3
3,5	3

Classification by Association

Candidate 3 item set :

Item Set	Support Count
1,3,5	3
2,4,6	1

Classification by Association

Frequent 3 item Set :

Item set	Support Count
1,3,5	3

L = {(1,5),(2,6),(3,5),(1,3,5)}

This is the set used to find the classification rules by association

Write Rules in **PREDICATE form**

Don’t forget to FIX and calculate Confidence and Support!

Testing :

Record	A1	A2	Test Data Class	Rules assigned class	Correctly classified
1	1	1	1	1	Yes
2	1	0	0	?	No
3	0	0	1	0	No
4	1	0	0	0	Yes

Predictive accuracy = 2/4 * 100 = 50 %