Online Association Rule Mining

By Christian Hidber
Department of Electrical Engineering and Computer Science, UC Berkeley

Background

- Mining for association rules is a form of data mining
- An example:
  - 65% of all customers who buy pasta and tomato sauce also buy parmesan cheese and red wine
- Useful for customer segmentation, cross-marketing, catalog design and product placement
- Online aggregation

Association Rule

- An association rule is an expression X =>Y where X and Y are disjoint itemsets
  - The *confidence* of this rule is the fraction of all transactions containing X that also contain Y
  - The *support* of this rule is the support of X U Y.
- A support must >= a user-specified support threshold
- A confidence must >= a user-specified confidence threshold

What's the problem?

- Finding association rules is a very CPU and memory intensive task
- All traditional algorithms operate offline
- User does not know the appropriate thresholds in advance

Traditional algorithms

- Apriori
- Partition
- Dynamic Itemset Counting

- User fixed support threshold in advance
- No feedback to user
- May need more than two scans

- OLAP-style
  - Support threshold s specified before the precomputation of the large itemsets
  - Large Itemset computation remains offline
  - Only rules with support >= s can be generated

Carma

- Continuous feedback
  - continuously produces association rules, while the list of purchases is scanned
- user controllable
  - During the first scan the user is free to change the support and confidence thresholds "on the fly"
- deterministic and accurate results
  - guarantees that it produces all association rules after at most 2 scans and for each rule its precise support and confidence value

Carma Algorithm: Phase I

Figure 1

- Count(v): number of occurrence of itemset v since v was inserted
- First Trans(v): index of the transaction at which v was inserted
- maxMissd(v): upper bound on the number of occurrences of v before v was inserted
- Minsupport: count(v)/i lower bound
- Maxsupport: (maxMissed(v) + count(v))/i upper bound

Carma Algorithm: Phase I

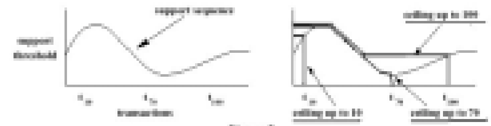


Figure 2

- Support lattice: a superset of all large itemsets
- Support sequence: a sequence of support threshold σ = (σ1, σ2,…..) σi denotes support threshold for the i-th transaction
- ⌈ σ ⌉i denotes the least monotone decreasing sequence. It is called ceiling of σ up to i. A sharp lower bound relative to which V is a support lattice.
- Avgi(σ)=1/i*Σ σ j (where 1<= j <= I) It's the running average of σ up to i.

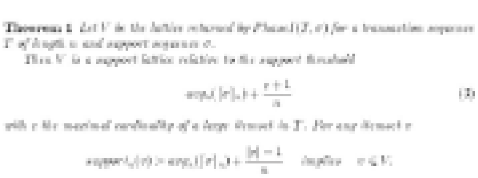
Carma Algorithm: Phase I

Figure 3

Carma Algorithm: Phase I

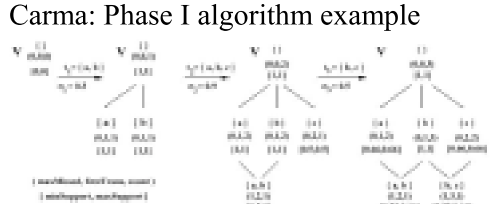
- Heart of the algorithm: maxMissed(v)
- maxMissied(v) <=maxMissed(w)+count(w) - 1
  - Support i (w) >= support i (v) for all subsets w of v and w in t i
- maxMissied(v) <=⌊ (i-1)avg i-1(⌈ σ ⌉i-1) ⌋+ ⎸v ⎹-1
  - Support (i-1) (v) <= avg i-1(⌈ σ ⌉i-1) + (⎸v ⎹-1)/(i-1)
- maxMissied(v) is defined as
  Min {⌊ (i-1)avg i-1(⌈ σ ⌉i-1) ⌋+ ⎸v ⎹-1,
  maxMissed(w)+count(w) - 1 ⎸ w ⊂ v }.
- maxMissed(v)<= i –1 for the current transaction index.

Carma: theorem 1



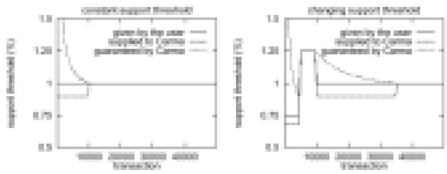
- The term (c+1)/n is desirable
- The term avg i(⌈ σ ⌉i) is a sharp lower bound relative to which V is a support lattice
- Support guarantee may not match the threshold specified by the user, but this guarantees *converges* to the user-specified threshold if the user keeps it constant for a large number of transactions

Carma: Phase I algorithm example



- Count(v): number of occurrence of itemset v since v was inserted
- First Trans(v): index of the transaction at which v was inserted
- maxMissd(v): upper bound on the number of occurrences of v before v was inserted
- Minsupport: count(v)/i lower bound
- Maxsupport: (maxMissed(v) + count(v))/i upper bound
- maxMissied(v) = Min {⌊ (i-1)avg i-1(⌈ σ ⌉i-1) ⌋+ ⎸v ⎹-1,
  maxMissed(w)+count(w) - 1 ⎸ w ⊂ v }.

Carma: Changing support thresholds



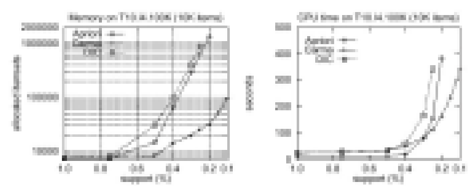
- To improve the speed of convergence
  - Run phase I with a lower threshold of s*0.9 instead of s.
  - Increase the threshold from s*0.9 to s, as the guaranteed threshold reaches s.

Carma: phase II

Carma Implementation

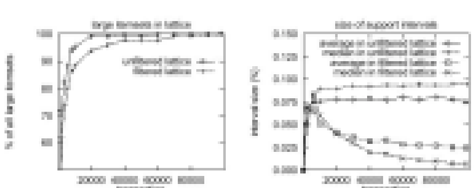
- Dataset with 100k transactions of an average size of 10 items chosen from 10k items and an average large itemset size of 4
- All itemsets are stored in a single hastable
  - Itemsets as keys: quickly access any subset

Performance:Carma, Apriori and DIC



- At thresholds 0.25% and below Carma outperform Apriori and DIC
  - Less number of scans
  - Smaller lattice maintained by Carma

Support Intervals



- Phase I maintains a superset of the large itemsets but not necessarily for the full transaction sequence
- Size of the support intervals (given by minsupport and maxsupport)
  - Average size 0.042% at threshold of 0.1% while 50% of all itemsets with an interval size below 0.004%

Conclusion

- Carma-compute large itemsets online
- Continuously produces large itemsets along with a shrinking support interval for each itemset
- Allow user to change the support threshold anytime during the first scan and always completes in at most 2 scan
- Carma's itemset lattice quickly approximates a superset of all large itemset while the sizes of the corresponding support intervals shrink rapidly
- Second scan is not needed when shrinking support intervals suffice so phase I can be used continuously