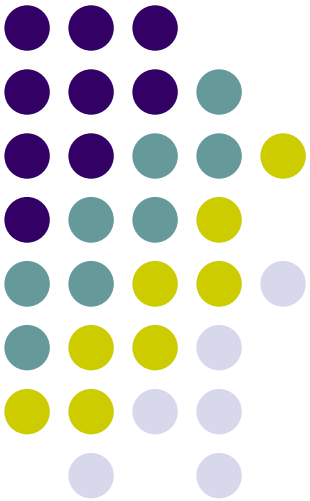
Association Rules

J. Savoy
Université de Neuchâtel

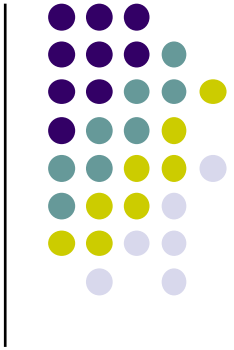


Ian H. Witten, Eibe Frank, M.A: Hall: Data Mining. Practical Machine Learning Tools and Techniques. Morgan Kaufmann

A. Rajaraman, D. Ullman: Mining of Massive Datasets. Cambridge University Press

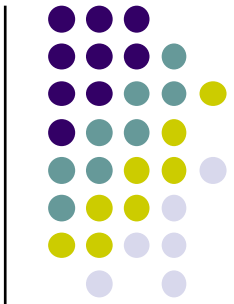
M. Bramer: Principles of Data Mining. Springer, 2007

Overview



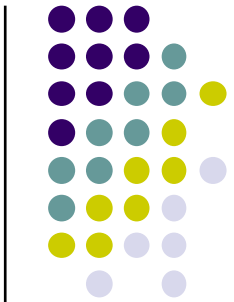
- **Market-Basket Model**
- Association Rules
- Applications
- Measures
- Finding the Best *N* Frequent Itemsets
- APriori Algorithm
- Improvements to Apriori

The Market-Basket Model



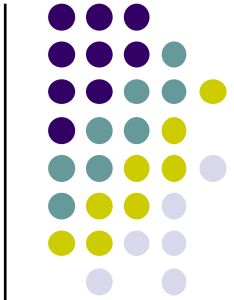
- A large set of *items*, e.g., objects sold in a supermarket.
- A large set of *baskets*, each of which is a small set of the items, e.g., the things one customer buys on one day.
- Customer Analysis
  - Market Basket Analysis uses the information about what a customer purchases to give us insight into who they are and why they make certain purchases.
- Product Analysis
  - Market basket Analysis gives us insight into the merchandise by telling us which products tend to be purchased together and which are most amenable to purchase.

The Market-Basket Model



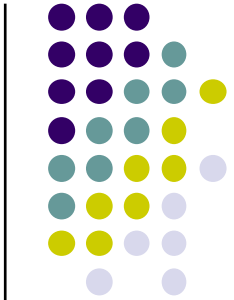
- Where does the data come from?
  - Credit card transactions, loyalty cards, discount coupons, customer complaint calls, plus (public) lifestyle studies
- Due to the evolution
  - simple invoice
  - barcode (the real revolution, + product id)
  - barcode & CumulusCard (+ customer id)
- Target marketing
  - Find clusters of “model” customers who share the same characteristics: interest, income level, spending habits, etc.
  - Determine customer purchasing patterns over time

The Market-Basket Model



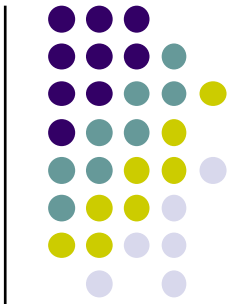
- Cross-market analysis
  - Associations/co-relations between product sales & prediction based on such association
- Ethics
  - medical products, GPS information
- Customer profiling
  - What types of customers buy what products (clustering or classification)
- Customer requirement analysis
  - Identifying the best products for different customers
  - Predict what factors will attract new customers

Overview



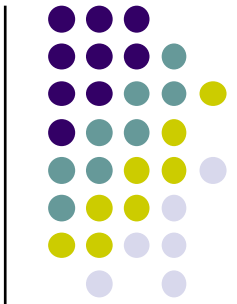
- Market-Basket Model
- **Association Rules**
- Applications
- Measures
- Finding the Best *N* Frequent Itemsets
- APriori Algorithm
- Improvements to Apriori

Association Rule



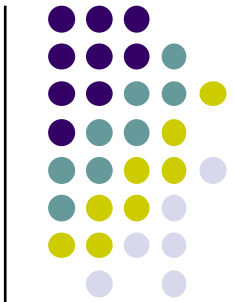
- Association rules…
  - … can predict any attribute and combinations of attributes
  - … are not intended to be used together as a set
  - Example: Amazon.com
- Association rules are not decision rules (that are more specific)
  - An association rule may include many attributes in its right-hand side.
  - An association rule may not use the class attribute
  - Or even, it may use it in the left-hand side

Association Rule



- Problem: huge number of possible associations
- Output needs to be restricted to show only the most predictive associations
  → only those that are pertinent
  (with high *support* and high *confidence*)
- Logical implication ≠ causality
  (here we are not at the logical level)

Association Rule



- Interpretation is not obvious:
  If windy = *false* and play = *no*
  then outlook = *sunny* and humidity = *high*
- is *not* the same as
  If windy = *false* and play = *no* then outlook = *sunny*
  If windy = *false* and play = *no* then humidity = *high*
- It means that the following also holds:
  If windy = *false* and play = *no* and humidity = *high*
  then outlook = *sunny*

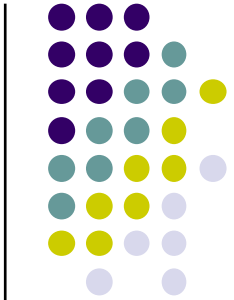
An Example

Customer	Product
1	Orange Juice (OJ), Soda
2	Milk, Orange Juice, Window cleaner
3	Orange Juice, Detergent
4	Orange Juice, Detergent, Soda
5	Window cleaner, Soda

The occurrence data

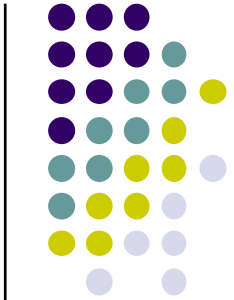
Co-occurrences	OJ	Window cleaner	Milk	Soda	Detergent
OJ	4	1	1	2	2
Window cleaner	1	2	1	1	0
Milk	1	1	1	0	0
Soda	2	1	0	3	1
Detergent	2	0	0	1	2

An Example



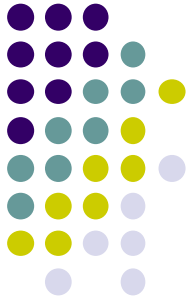
- The co-occurrence table contains some simple patterns
  - Orange juice and soda are more likely to be purchased together than any other two items
  - Detergent is never purchased with window cleaner or milk
  - Milk is never purchased with soda or detergent
- We focus on binary information
  (item bought or not in conjunction with another one)
- These simple observations are examples of associations and may suggest a formal rule like:
  - IF a customer purchases soda
    THEN the customer also purchases orange juice

An Example



- How good are the rules?
- How can we measure this goodness?
- Question:
  the items appearing in the rule can be find in only a few or in *many baskets* (transactions)?
- In the data, two of five transactions (2 baskets with both items over 5 baskets) include both soda and OJ.
  These two transactions *support* the rule
  IF soda THEN orange juice.
  The support for the rule is two out of five (2/5) or 40%
- Support is also called *coverage* (WEKA).

An Example

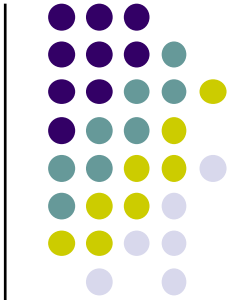


- But the soda (antecedent of the rule) may appear in many other baskets… Thus the confidence for the rule is lower. In our data, 2 transactions contain soda and orange juice, and 3 baskets contains soda.
  There is a high degree of *confidence* in the rule.
  In fact 2 transactions over 3 that contains soda contains orange juice.

  So the rule
  "IF soda THEN orange juice" has a confidence of 66.7%.

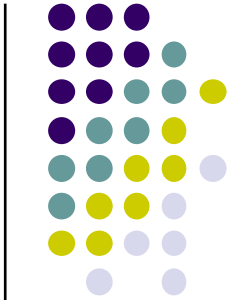
- Confidence is also called *accuracy* (WEKA)

An Example



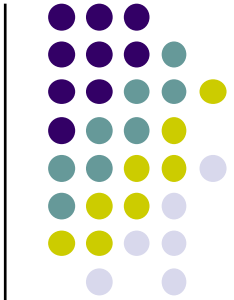
- Confidence and support
- A rule must have some minimum user-specified confidence
  - 1 & 2 → 3 has a 90% *confidence* if when a customer bought 1 and 2, in 90% of the cases, the customer also bought 3.
    (don't expect having a 100% confidence, at least for interesting baskets)
- A rule must have some minimum user-specified *support*
  - 1 & 2 → 3 should hold in some minimum percentage of transactions to have value (be valid for only 1% of the case in usually not really useful).

An Example



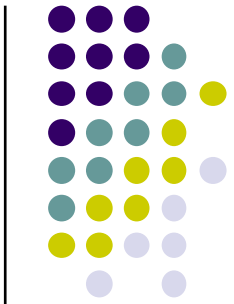
- Find all rules that have “Diet Coke” as a *result*. These rules may help plan what the store should do to boost the sales of Diet Coke.
- Find all rules that have “Yogurt” in the *condition*. These rules may help determine what products may be impacted if the store discontinues selling “Yogurt”.
- Find all rules that have “Brats” in the *condition* and “mustard” in the *result*. These rules may help in determining the additional items that have to be sold together to make it highly likely that mustard will also be sold.
- Find the best *k* rules that have “Yogurt” in the *result*.

Overview



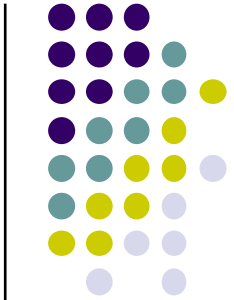
- Market-Basket Model
- Association Rules
- **Applications**
- Measures
- Finding the Best *N* Frequent Itemsets
- APriori Algorithm
- Improvements to Apriori

Applications



- Real market baskets: chain stores keep terabytes of information about what customers buy together. (starting to talk about Big Data)
  - Tells how typical customers navigate stores, verify if they buy tempting items (according to their position).
  - Suggests tie-in “tricks,” e.g., run sale on diapers and raise the price of beer.
- High support needed, or no $$’s .

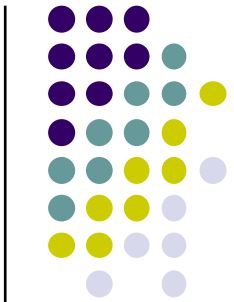
Applications



- Other examples…

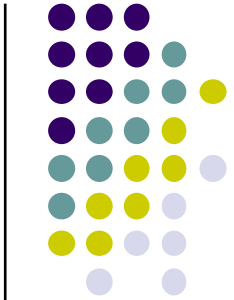
- “Baskets” = documents; “items” = words in those documents.
  - Lets us find words that appear together unusually frequently, i.e., linked concepts.
- “Baskets” = sentences, “items” = documents containing those sentences.
  - Items that appear together too often could represent plagiarism.

Applications



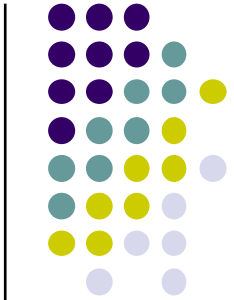
- “Baskets” = Web pages; “items” = linked pages.
  - Pairs of pages with many common references may be about the same topic.
- “Baskets” = Web pages *p* ;
  “items” = pages that link to *p* .
  - Pages with many of the same links may be mirrors or about the same topic.
- Useful for e-business (reputation)!

Important Point



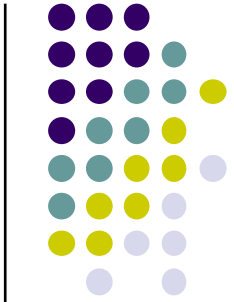
- “Market-Basket” is an abstraction that models any many-many relationship between two concepts: “items” and “baskets.”
  - Items need not be “contained” in baskets.
- The only difference is that we count co-occurrences of items related to a basket, not vice-versa.
- WalMart sells 100,000 items and can store billions of baskets. (Fewer items in Switzerland!)
- The Web has over 100,000,000 words and billions of pages.

Overview



- Market-Basket Model
- Association Rules
- Applications
- **Measures**
- Finding the Best *N* Frequent Itemsets
- APriori Algorithm
- Improvements to Apriori

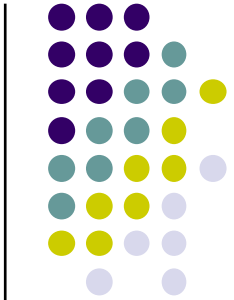
Measures



More formally
IF *left* THEN *right*

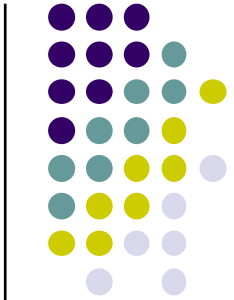
- N_{left}: Number of transactions matching the *left*
  N_{right}: Number of transactions matching the *right*
  N_{both}: Number of transactions matching both *left* & *right*
  N_{total}: Number of instances
- Support (coverage) = N_{both} / N_{total}
- Confidence (accuracy) = N_{both} / N_{left}
  (predictive accuracy, reliability).
- Completeness = N_{both} / N_{right}
- All values between 0 and 1 (or 100%)

Support



- Simplest question: find sets of items that appear "frequently" in the baskets.
- *Support* for itemset *I* = the number of baskets containing all items in *I*.
- Given a support *threshold s*, sets of items that appear in $\geq$ *s* baskets are called *frequent itemsets*.

A Simple Example



TransID	Items
1	{1, 2, 3}
2	{1, 3}
3	{1, 4}
4	{2, 5, 6}

For rule 1 → 3:

Support = support({1,3}) = 2/4 = 50%

Confidence (1→3) =

supp({1,3}) / supp({1}) = 2/3 = 66%

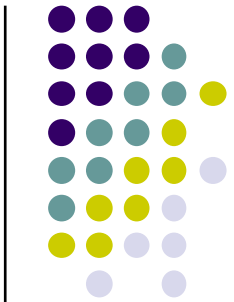
Confidence (3→1) =

supp({1,3}) / supp({3}) = 2/2 = 100%

FreqItem	Support
{1}	75%
{2}	50%
{3}	50%
{4}	25%

FreqItem 2 items	Support
{1, 2}	25%
{1, 3}	50%
{1, 4}	25%
{2, 3}	25%

Example



- Items = {milk, coke, pepsi, beer, juice}.
- Support = 3 baskets.

B_1 = {m, c, b} B_2 = {m, p, j}

B_3 = {m, b} B_4 = {c, j}

B_5 = {m, p, b} B_6 = {m, c, b, j}

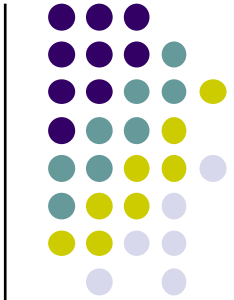
B_7 = {c, b, j} B_8 = {b, c}

- Frequent itemsets:

{m}	{c}	{b}	{j}	{m,b}	{c,b}	{j,c}
(5)	(5)	(5)	(4)	(4)	(4)	(3)

- Are all rules pertinent? Produce by chance only?

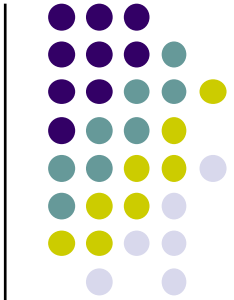
Other Measures



The Piatetsky-Shapiro Criteria

- Criteria 1
  The measure should be zero if $N_{both} = (N_{left} \times N_{right}) / N_{total}$
  Interestigeness should be zero if the antecedent and the consequent are statistically independent
- Criteria 2
  The measure should increase monotonically with N_{both}
- Criteria 3
  The measure should decrease monotonically with each of N_{left} and N_{right}

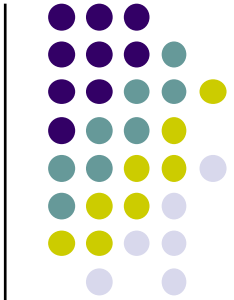
Measures



Explanation (*ceteris paribus*)

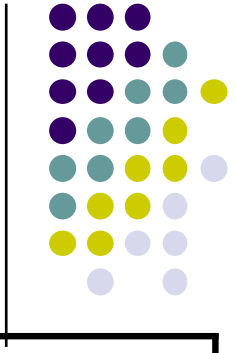
- Criteria 2
  The more the rule predicts, the more interesting it is
- Criteria 3
  The more instances that match the left-hand side (the right-hand side) of a rule, the less interesting it is
- Criteria 1
  If the antecedent and the consequent are statistically independent, a random rule will have no value
- We can define the RI Measure as
  $RI = N_{both} - ((N_{left} \times N_{right})/N_{total})$

Overview



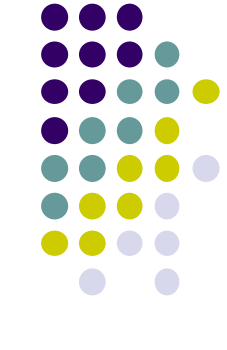
- Market-Basket Model
- Association Rules
- Applications
- Measures
- **Finding the Best *N* Frequent Itemsets**
- APriori Algorithm
- Improvements to Apriori

Example: Weather Data



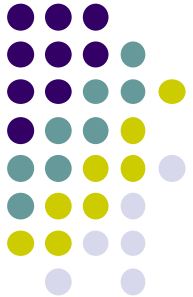
Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	mild	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Example

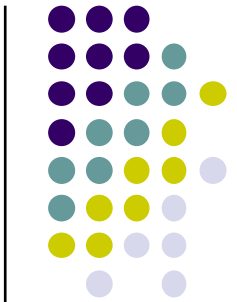


One-item sets	Two-item sets	Three-item sets	Four-item sets
Outlook = *Sunny* (5)	Outlook = *Sunny* Temperature = *Hot* (2)	Outlook = *Sunny* Temperature = *Hot* Humidity = *High* (2)	Outlook = *Sunny* Temperature = *Hot* Humidity = *High* Play = *No* (2)
Temperature = *Cool* (3)	Outlook = *Sunny* Humidity = *High* (3)	Outlook = *Sunny* Humidity = *High* Windy = *False* (2)	Outlook = Rainy Temperature = *Mild* Windy = *False* Play = *Yes* (2)

Example



- In total: 12 one-item sets
  47 two-item sets,
  39 three-item sets,
  6 four-item sets and
  0 five-item sets (with minimum $N_{both} \geq 2$)
- Once all item sets with minimum support have been generated, we can turn them into rules
- Example (N=3):
  Humidity = *Normal*, Windy = *False*, Play = *Yes* (4)
- 2^N-1 potential rules
  With my example: 7 possible rules



Example

- Having a frequent itemset is the first task.
- Defining the rule is the second.
  In the Weather data, we have seven potential rules:

If Humidity = *Normal* and Windy = *False* **then** Play = *Yes* (4/4)

If Humidity = *Normal* and Play = *Yes* **then** Windy = *False* (4/6)

If Windy = *False* and Play = *Yes* **then** Humidity = *Normal* (4/6)

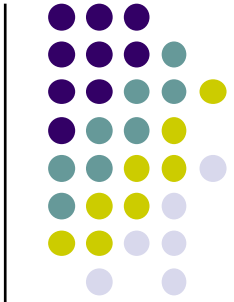
If Humidity = *Normal* **then** Windy = *False* and Play = *Yes* (4/7)

If Windy = *False* **then** Humidity = *Normal* and Play = *Yes* (4/8)

If Play = *Yes* **then** Humidity = *Normal* and Windy = *False* (4/9)

If True **then** Humidity = *Normal* and Windy = *False*
and Play = *Yes* (4/12)

Finding the Best *N* Rules

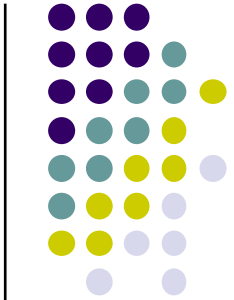


- In general: Find the best N rules.
  Admit that the value of *N* is around 20 to 50
- Define what we may by "best":
  we can use the J-measure (see Appendix)
- Which rules: all?
  Conjunction "attribute = *value*" but the same attribute cannot appear in both the *left* and *right*-hand side
- Too many rules to test …
  Limit the number of attributes in the *left* / *right*-hand side and thus defining the search space
- Define a search strategy to generate all the rules and maintain a table of the best *N* rules

Finding the Best *N* Rules

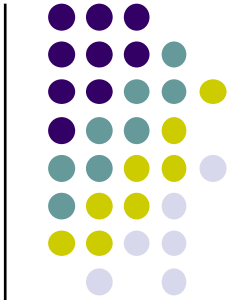
- With our measure, we need to define our search strategy
  Generate all potential rules?
- Imagine we have only 10 attributes (a_1, a_2, …, a_{10}) with only three values (1, 2, 3).
- Limiting the search to one term in the RHS (30 possibilities).
  - Limiting the LHS to one term (27 possibilities)
    IF $a_1 = 1$ THEN $a_2 = 3$
    IF $a_1 = 2$ THEN $a_2 = 3$
    IF $a_1 = 3$ THEN $a_2 = 3$
    IF $a_3 = 1$ THEN $a_2 = 3$
    IF $a_3 = 2$ THEN $a_2 = 3$
    → generating 27 x 30 rules and keep the best *N* in a table

Finding the Best *N* Rules



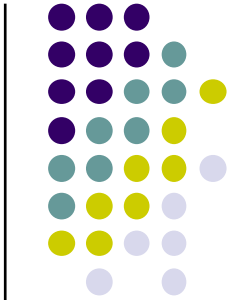
- Continuation
  - Limiting the LHS to one term (27 possibilities)
    → generating 27 x 30 rules and keep the best *N* in a table
  - Limiting the LHS to two terms (27 x 24 possibilities)
    IF $a_1 = 1$ and $a_3 = 1$ THEN $a_2 = 3$
    IF $a_1 = 1$ and $a_3 = 2$ THEN $a_2 = 3$
    …
  - Limiting the LHS to three terms
    IF $a_1 = 1$ and $a_3 = 1$ and $a_4 = 1$ THEN $a_2 = 3$
    …
  - up to rules having 9 attributes
    7,864,290 rules to generate and evaluate
- Can we do a better job?

Finding the Best *N* Rules



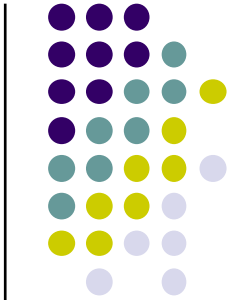
- Try to find the best (say) 20 rules of order one
  and expand only those
- The best 20 rules of order two are used to continue the search in order to find the "best" order three rules.
- This is a beam search (heuristics)
  No guarantee to have the best N rules
- How to solve this problem?

Generating Itemsets



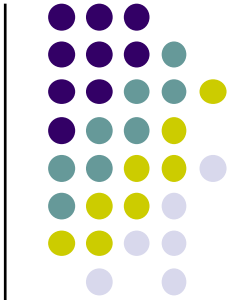
- If we have a pattern like ($L \rightarrow R$)
  and if we note $L \cup R$ the set of all items appearing in the rule (L and R are disjoint).
  We are interested of rules having a $card[L \cup R] \geq 2$
- If we have 8 transactions
  L matches 4 transactions (namely 2, 4, 6 and 7)
  $L \cup R$ matches 3 transaction (namely 2, 4 and 7)
- $support(L) = count(L) / 8 = 4 / 8$
  $support(L \cup R) = count(L \cup R) / 8 = 3 / 8$
  The support is the proportion of the database to which the item in L and the item in R occur together.

Generating Itemsets



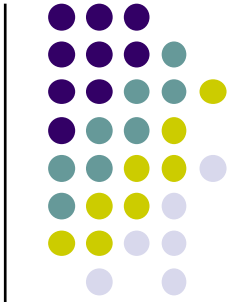
- The predictive accuracy of the rule L → R is measured by the confidence, the proportion of transactions for which the rule is satisfied
- confidence(L → R) = count(L ∪ R) / count(L) = 3 / 4
  or
  confidence(L → R) = support(L ∪ R) / support(L)
- Ideally, every transaction matched by L would also be matched by L ∪ R (confidence will be 1).
- Usually, we ignore any rule for which the support is below a *minsup* (1%) or rules having a confidence below a *minconf* threshold (e.g., 70%).

Overview



- Market-Basket Model
- Association Rules
- Applications
- Measures
- Finding the Best *N* Frequent Itemsets
- **APriori Algorithm**
- Improvements to Apriori

The Problem



- The typical question, we want to address:
  "find all association rules
  with support $\geq s$ and confidence $\geq c$"
  - "support" of an association rule is the support of the set of items it mentions.
- Hard part: finding the high-support (*frequent*) itemsets.
  - Checking the confidence of association rules involving those sets is relatively easy
  - Do it within a reasonable delay

Frequent Itemsets

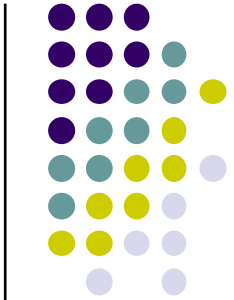
- *Supported itemset* is any itemset for which the value of support is greater than or equal *minsup*
  (can also be called *frequent itemset* or *large itemset*)
- Objective: Generate all supported itemset
  Naïve approach:
  1. Generate all supported itemsets (L ∪ R) with cardinality at least two.
  2. For each such itemset, generate all the possible rules with at least one item on each side and retain those for which confidence ≥ *minconf*.
- We achieve our goal, but …

Generating Frequent Itemsets

- To obtain all supported itemsets having a cardinality ≥ 2, we need to generate the number of subsets of S = (L U R), the set of all items. If we have *m* items, the size of S = 2^m. From this number we may of course remove the set composed on a single item (*m*) and the empty set.

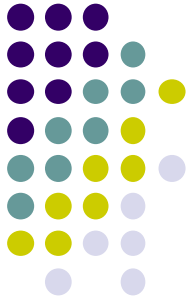
  Thus we have to generate 2^m - *m* - 1 sets!
- If *m* = 20, we have to generate 2^{20} - 20 - 1 = 1,048,555
  If *m* = 100, we obtain 2^{100} - 100 - 1 ≈ 10^{30}.
- This is unrealistic …

The Idea



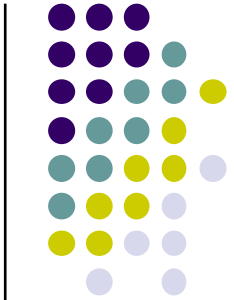
- Apriori algorithm (Agrawal & Srikant 1994)
- How can we efficiently find all frequent itemsets?
- Finding one-item sets (that is relatively easy)
- Idea: use one-item sets to generate two-item sets, two-item sets to generate three-item sets, …
- If (A B) is supported, then (A) and (B) are also supported
- This is the *download closure property* of itemsets
- In general: if X is frequent *k*-item set, then all (*k*-1) item subsets of X are also frequent
- → Compute *k*-item set by merging (*k*-1) item sets

The Idea



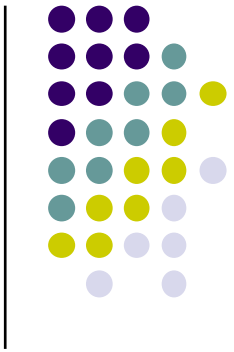
- If we write by L_k the set containing all the supported itemsets with cardinality *k*,
- If $L_k = \emptyset$ (the empty set) then L_{k+1}, L_{k+2}, etc. must also be empty
  If any supported itemsets of cardinality *k*+1 or larger exists, they will have subsets of cardinality *k*.
- The idea is therefore to generate the supported itemsets in *ascending* order of cardinality, i.e., all those with one element first, then all those with two elements, …

Apriori



- Thus we need a method to go from L_{k-1} to L_k
- We will use the set L_{k-1} to form the set C_k, the candidate itemsets of cardinality *k*. In C_k, we must have all supported itemsets of cardinality *k* and of course some other itemsets.
- Next we need to form L_k, a subset of C_k either by checking the content of L_{k-1} or by checking against the transactions in the database. Of course, only itemsets having a support value > *minsup* are considered.

Apriori



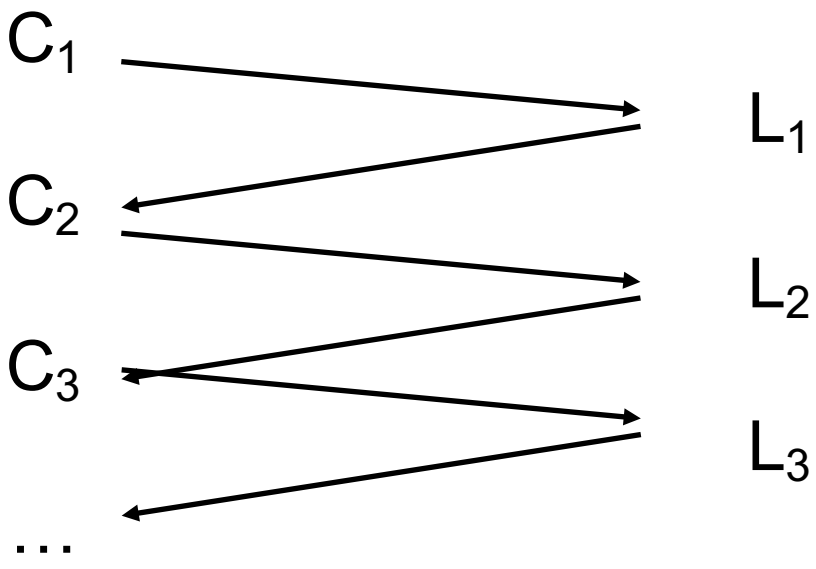
- Apriori algorithm

```
Create L1 = set of supported itemsets of
      cardinality 1;
set k ← 2;
while (Lk ≠ Ø) {
    create Ck from Lk-1;
    prune all the itemsets in Ck that are
       not supported to create Lk;
    k ← k + 1;
    }
```

- The set of all supported itemsets is $L_1 \cup L_2 \cup \ldots \cup L_k$

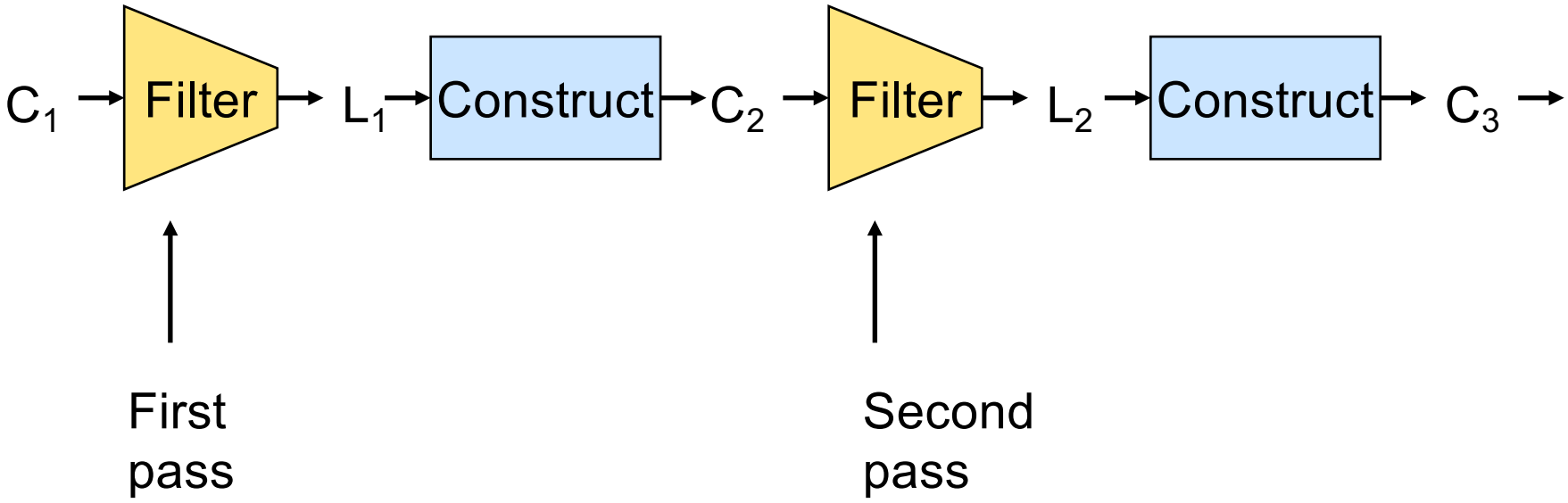
APriori

- The Apriori algorithm works as follows
- Candidate Itemset Supported Itemset

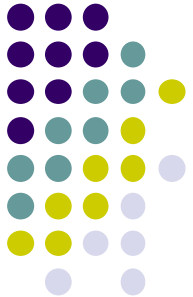


APriori

Or using another graphical view

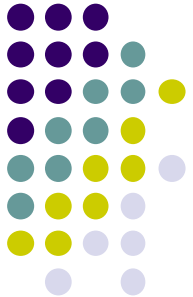


Example



- We have a database of 100 items and a large number of transactions
- We start by building C_1 and make a pass through the database to establish their support count and to define L_1 (we found 8 members {a}, {b}, {c}, {d}, {e}, {f}, {g}, {h})
  No rule can be generated.
- We generate C_2 from L_1 (join step) by having all the itemsets with two members drawn from the 8 items {a} to {h}. (We do not need to consider the 92 other items such as {b,w} because {w} is not supported)

Example

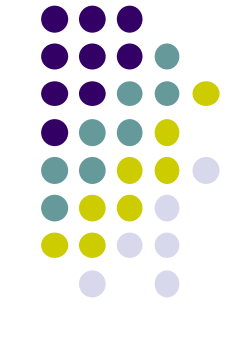


- We found 28 possible itemsets of cardinality 2

  {a,b}, {a,c}, {a,d}, {a,e}, {a,f}, {a,g}, {a,h},
  {b,c}, {b,d}, {b,e}, {b,f}, {b,g}, {b,h},
  {c,d}, {c,e}, {c,f}, {c,g}, {c,h},
  {d,e}, {d,f}, {d,g}, {d,h},
  {e,f}, {e,g}, {e,h},
  {f,g}, {f,h},
  {g,h}

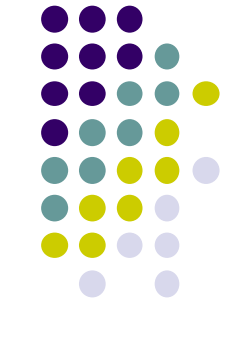
- We represent each itemset in a standard sorted form (lexicographic) to include only once, for example, the set {d,g} and {g,d}

Example



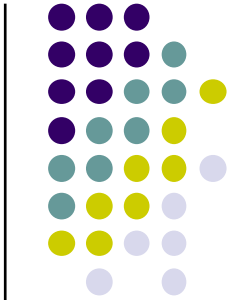
- We make a pass through the database to find the support counts of those itemsets.
- L_2 is formed by {a,c}, {a,d}, {a,h}, {c,g}, {c,h}, {g,h}
- We form C_3 composed of {a,c,d}, {a,c,h}, {a,d,h}, {c,g,h} but we can find that the set {a,c,d}, and {a,d,h} must be removed. Why? Because {c,d} and {d,h} do not appear in L_2. Thus C_3 is formed only by {a,c,h}, {c,g,h}
- We make a pass through the database to find the support counts of those itemsets and we found L_3 = {{a,c,h}, {c,g,h}}
- We try to form C_4 but we found no member for this set.

Example



- Finally, we have found L_1 (8) U L_2 (6) U L_3 (2) supported itemsets.
  The final result will be 8 itemsets = $L_2 \cup L_3$
- We made three passes through the database
- The support counts was done for 100 + 28 + 2 = 130 sets, a huge improvement over the 10^{30} possible itemsets
- One of the main problem is to verify the C_2 candidates which is composed of $m(m\text{-}1)/2$ sets

Apriori



- Create C_k from L_k

  Compare each member of L_{k-1}, say A, with every other member, say B, in turn. If the first k-2 items in A and B (i.e. all but the rightmost elements of the two itemsets) are identical, place A U B into C_k

- Prune step

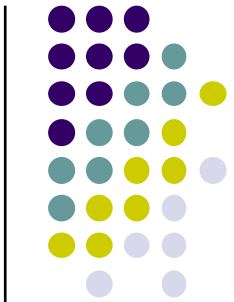
  For each member c of C_k in turn {
  Examine all subsets of c with k-1 elements
  Delete c from C_k if any of the subsets is not a member of L_{k-1}
  }

Generating Rules

- From a supported itemsets {c,d,e},
  we can form different rules such as

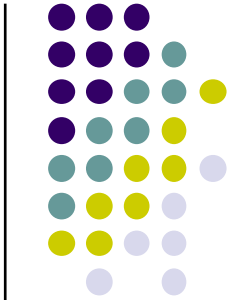
Rule L → R	count(L∪R)	count(L)	conf(L→R)
de → c	3	3	1.0
ce → d	3	4	0.75
cd → e	3	4	0.75
e → cd	3	4	0.75
d → ce	3	4	0.75
c → de	3	7	0.43

Generating Rules



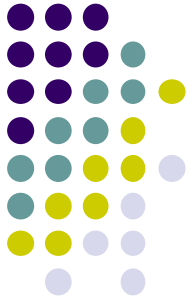
- Transferring members of a supported itemset from the left-hand side of a rule to the right-hand side cannot increase the value of rule confidence
- R1: Confidence ($A \rightarrow B \cup C$) = support(S) / support(A)
  R2: Confidence ($A \cup B \rightarrow C$) = support(S) / support($A \cup B$)
  The proportion of transactions supporting A is clearly greater or equal than those supporting ($A \cup B$)
  Thus support(A) ≥ support($A \cup B$)
  Thus confidence(R1) ≤ confidence(R2)

Generating Rules



- From this point,
  if the confidence of a rule ≥ *minconf*, we can call the itemset of its right-hand side *confident*.
  If not, we will use the term *unconfident*.
- Any superset of an unconfident right-hand itemset is unconfident.
- Any (non-empty) subset of a confident right-hand itemset is confident.
- Reuse the Apriori strategy to generate the most pertinent rules, by generating R itemsets of increasing cardinality

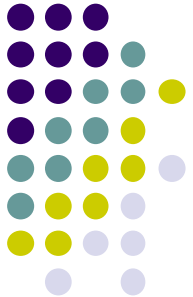
Rules Generations



- Given the transactions T, min support s_0 and min confidence c_0
- Set R = ∅

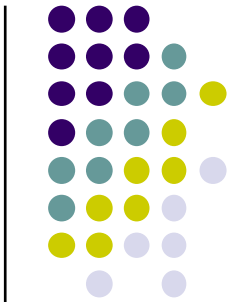
```
M = FrequentItems(T, s0, c0)
For each m ∈ M {        # each frequent itemset
 For each h ⊆ m and b ⊆ m such that h∩b=∅ {
   if (supp(h∪b)/supp(b)≥ c0)
      then R ← R ∪ {if h then b}
   }
}
return R
```

Quality of the Rules



- After selecting rules according to the support and confidence measures, we may want to *rank the rules* according to their importance (and explain the rules).
- The lift of a rule $L \rightarrow R$
  measures how many more times the items in L and R occur together in transactions than would be expected it if the itemsets L and R were statistically independent
- lift($L \rightarrow R$) = count($L \cup R$)/ [count(L) x support(R)]
  If L and R are independent, we can expect
  count(L) x support(R) transactions in which both itemsets will appear

Quality of the Rules



- The lift of a rule L → R

  lift(L → R) = support(L ∪ R)/ [support(L) x support(R)]

  and we can see that

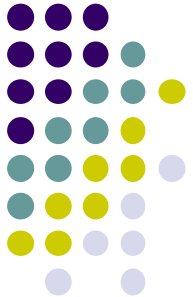
  lift(L → R) = lift(R → L)

- Rules with lift value > 1 are the interesting rules, indicating that transactions contains L tend also to contain R

- Example: lift(L→R) = count(L ∪ R)/ [count(L)xsupport(R)]

  = 30 / [100 x 0.2] = 30 / 20 = 1.5

Quality of the Rules

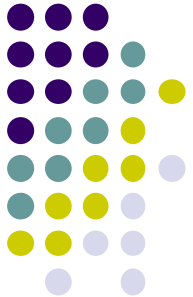


- Example: our database contains 2,000 transactions

count(L)	count(R)	count(L ∪ R)
220	250	190

- support(L → R) = 190 / 2,000 = 0.095
- confidence(L → R) = 190 / 220 = 0.864
- lift(L → R) = 190 / [220 x (250/2,000)] = 6.91
  So purchasing the items in L makes it 0.864 / 0.125 = 6.91 times more likely that the items in R are also purchased.

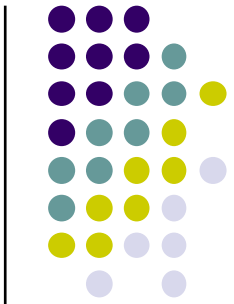
Quality of the Rules



- The leverage is another measure of interestingness.
- Leverage($L \rightarrow R$) =

  support($L \cup R$) - support(L) x support(R)
- Improvement over a pure chance association

Quality of the Rules

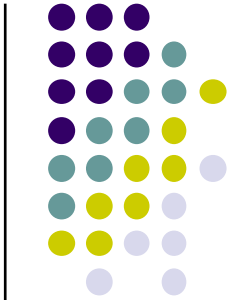


- Example: our database contains 100,000 transactions

count(L)	count(R)	count(L ∪ R)
8,000	9,000	7,000

- support(L→R) = 7,000 / 100,000 = 0.07
- confidence(L→R) = 7,000 / 8,000 = 0.875
- lift(L→R) = 7,000 / [8,000 x (9,000/100,000)] = 9.72
- leverage(L→R) = 7,000/100,000
  -[(8,000/100,000) x (9,000/100,000)] = 0.0628

Quality of the Rules



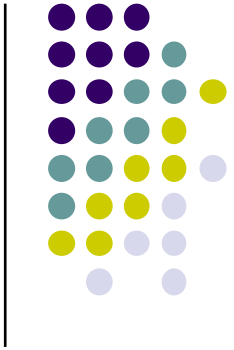
This rule

- applies to 7% of the transactions in the database (support)
- is satisfied for 87.5% of the transactions that include the items in L (confidence)
- and this is 9.72 times more frequent than would be expected by chance (lift)
- The improvement in support compared with chance is 0.063 (or 6.3 transactions per 100) (leverage)

Relationships Among Measures

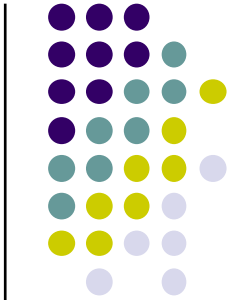
- Rules with high support and confidence may be useful even if they are not “interesting.”
  - We don’t care if buying bread *causes* people to buy milk, or whether simply a lot of people buy both bread and milk.
- But *high interest* (lift, leverage) suggests a cause that might be worth investigating.

Overview



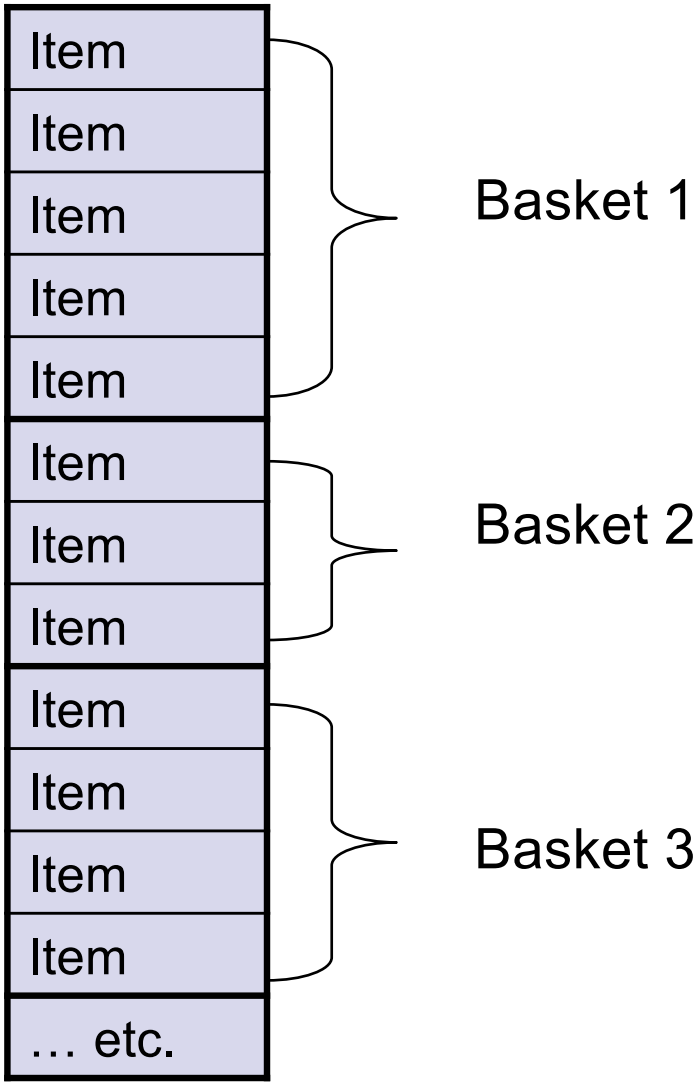
- Market-Basket Model
- Association Rules
- Applications
- Measures
- Finding the Best *N* Frequent Itemsets
- APriori Algorithm
- **Improvements to Apriori**

Computation Model

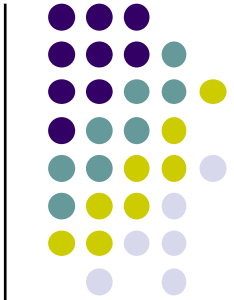


- Typically, data is kept in a “flat file” rather than a database system.
  - Stored on disk.
  - Stored basket-by-basket.
  - Expand baskets into pairs, triples, etc. as you read baskets.

File Organization

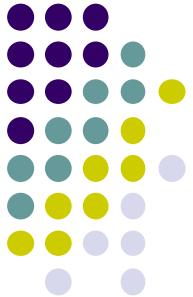


Computation Model



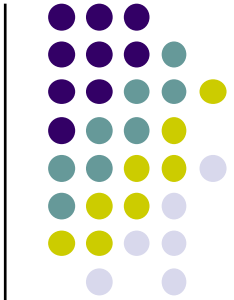
- The true cost of mining disk-resident data is usually the number of disk I/O's.
  why?
- In practice, association-rule algorithms read the data in *passes* --- all baskets read in turn.
- Thus, we measure the cost by the number of passes an algorithm takes.

Main-Memory Bottleneck



- For many frequent-itemset algorithms, main memory is the critical resource.
  - As we read baskets, we need to count something, e.g., occurrences of pairs.
  - The number of different things we can count is limited by main memory.
  - Swapping counts in/out is a disaster.
- The hardest problem often turns out to be finding the frequent pairs.
- We'll concentrate on how to do that, then discuss extensions to finding frequent triples, etc.

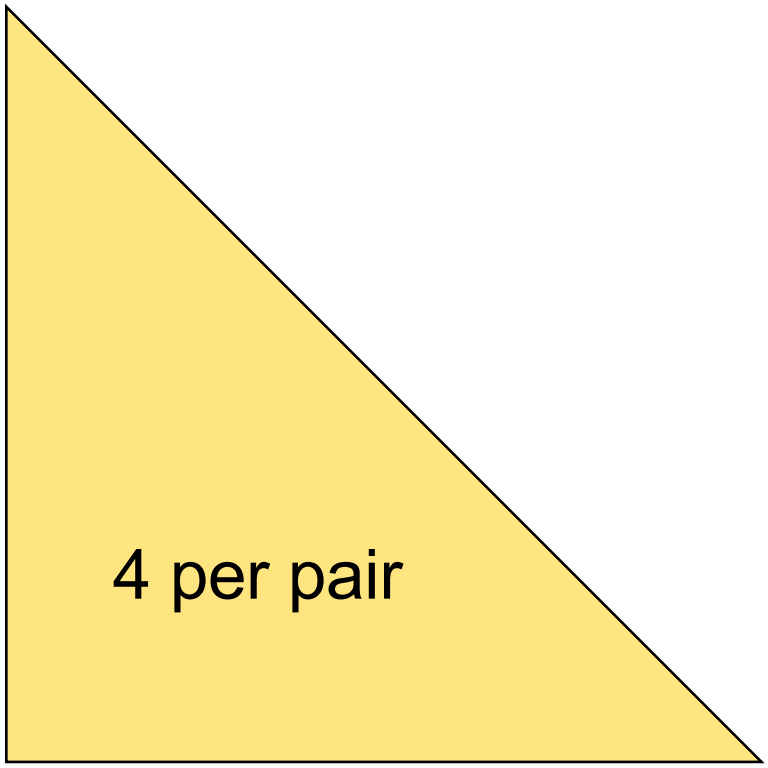
Naïve Algorithm

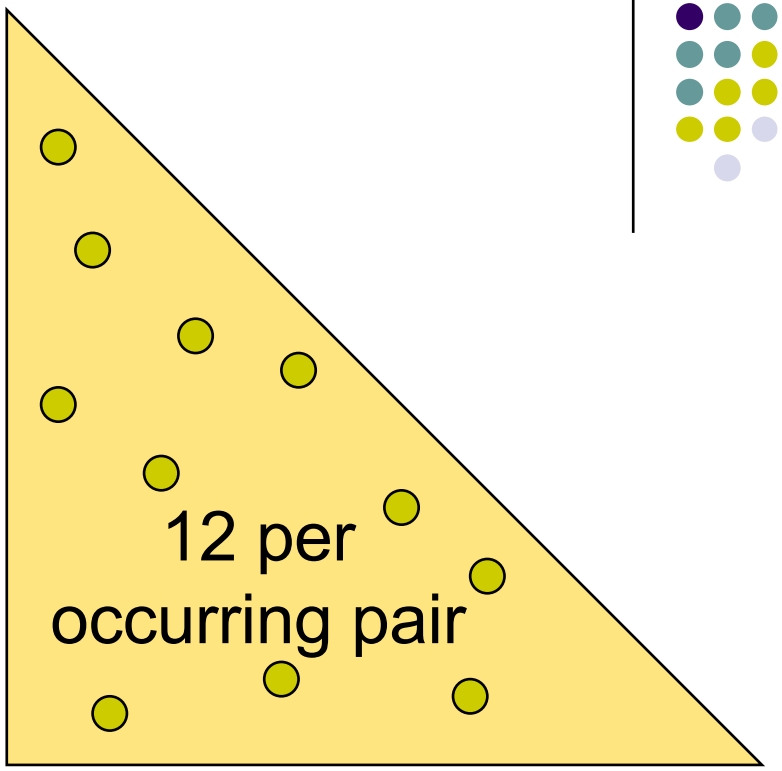


- Read file once, counting in main memory the occurrences of each pair.
  - Expand each basket of *n* items into its *n* (*n* -1)/2 pairs.
- Fails if (#items)^2 exceeds main memory.
  - Remember: #items can be 100K (Wal-Mart) or 10T (Web pages).

Details of Main-Memory Counting

- Two suggested approaches:
  1. Count all item pairs, using a triangular matrix.
  2. Keep a table of triples [*i*, *j*, *c*] = the count of the pair of items {*i*,*j* } is *c*.
- (1) requires only (say) 4 bytes/pair.
- (2) requires 12 bytes, but only for those pairs with count > 0.
- Which approach do you prefer?
  (1) is simple to implement (sort the items in the set)
  (2) little bit more complex, reduce the memory usage but does it worth?

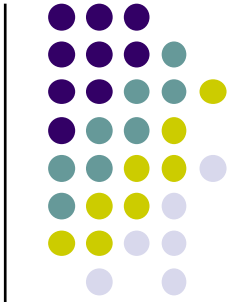




Method (1)

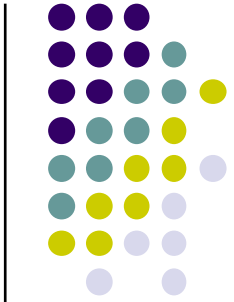
Method (2)

Details of Approach #1



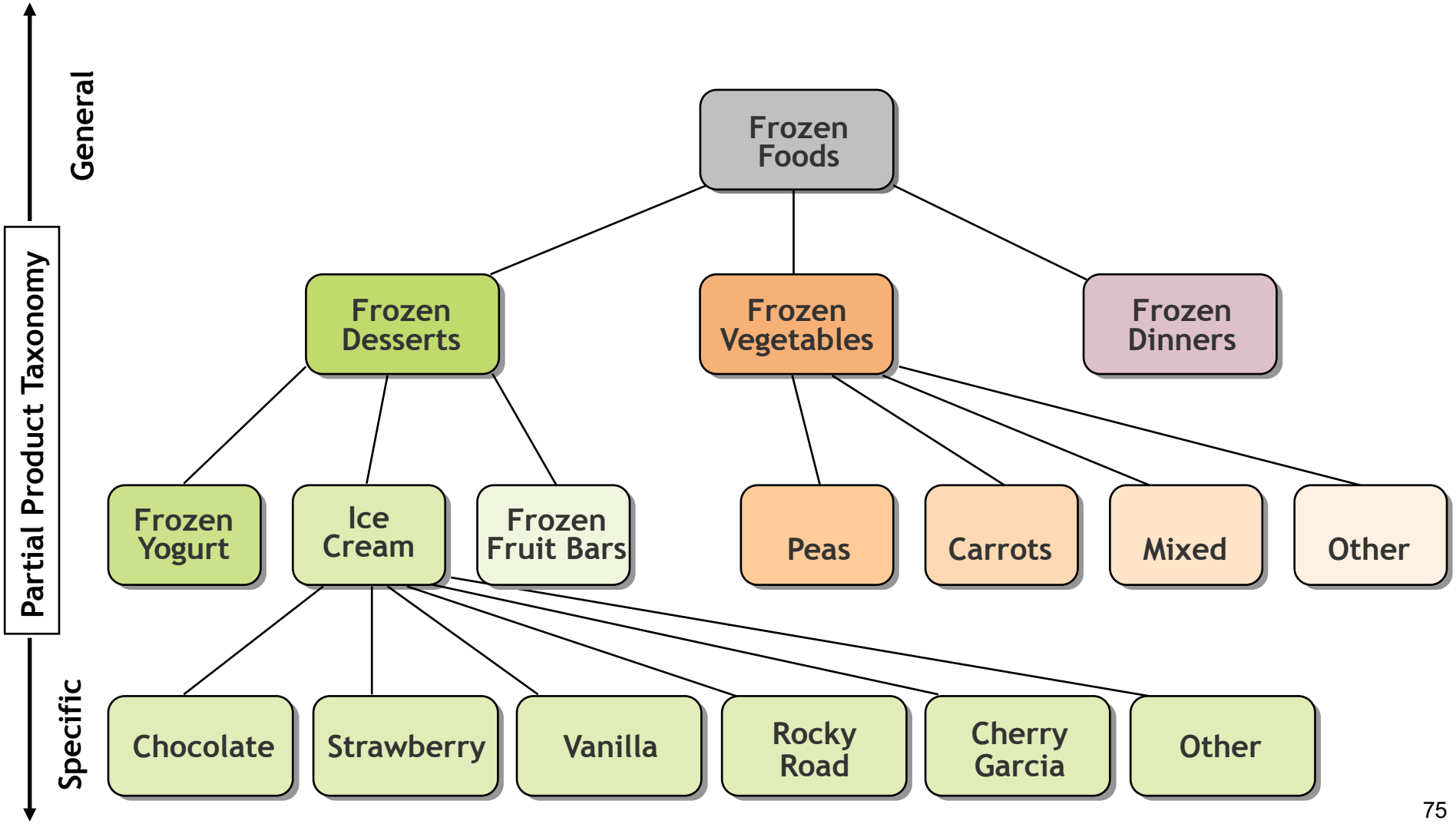
- Number items 1, 2, …, m
- Keep pairs in the order {1,2}, {1,3}, …, {1,m}, {2,3}, {2, 4}, …, {2,m}, {3,4}, …, {3,m}, …, {m -1, m}.
- Find pair {i, j} at the position (with m=100, starting at 1)
  $(i-1)\cdot(m-i/2)+j-i$
  e.g., {2,3} → $1\cdot(100-2/2)+3-2 = 99+1 = 100$
  e.g., {1,3} → $0\cdot(100-1/2)+3-1 = 0+2 = 2$
  e.g., {3,4} → $2\cdot(100-3/2)+4-3 = 197+1 = 198$
- Total number of pairs $m(m-1)/2$;
  Total bytes about $2m^2$

Details of Approach #2

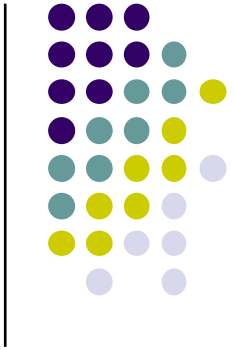


- You need a hash table, with *i* and *j* as the key, to locate (*i, j, c*) triples efficiently.
  - Typically, the cost of the hash structure can be neglected.
- Total bytes used is about 12*p*, where *p* is the number of pairs that actually occur.
  - Beats triangular matrix if at most 1/3 of possible pairs actually occur.
- Example: *m*=100, number of pairs = 4,950, ***p***=20%
  Approach#1: $4 \cdot 4{,}950 = 19{,}800$ bytes
  Approach#2: $12 \cdot 990 = 11{,}880$ bytes

Choosing the right set of items

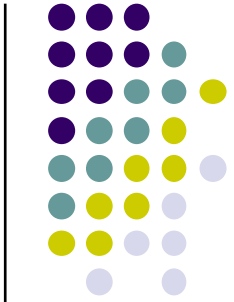


Conclusion



- Apriori method makes one pass through the data for each different size item set
- Other possibility: generate (*k*+2)item sets just after (*k*+1)item sets have been generated
- Result: more (*k*+2)item sets than necessary will be considered but less passes through the data
- Makes sense if data too large for main memory
- Practical issue: generating a certain number of rules (e.g. by incrementally reducing min. support)
- Confidence is not necessarily the best measure
  - Example: milk occurs in almost every supermarket transaction
- Other measures have been suggested (e.g. lift)

J-Measure



- From a rule IF (Y=*y*) THEN X=*x*
  we can measure the information content of the rule (in bits of information) by J(x;Y=*y*)
- Prob[y] = probability that the left-hand side (antecedent) of the rule will occur
- Prob[x] = probability that the right-hand side (consequent) of the rule will be satisfied
- Prob[x|y] = probability that the right-hand side will be satisfied if we know that the left-hand side is satisfied
- We can estimate as Prob[y] = N_{left} / N_{total}
  Prob[x] = N_{right} / N_{total}
  Prob[x|y] = N_{both} / N_{left}

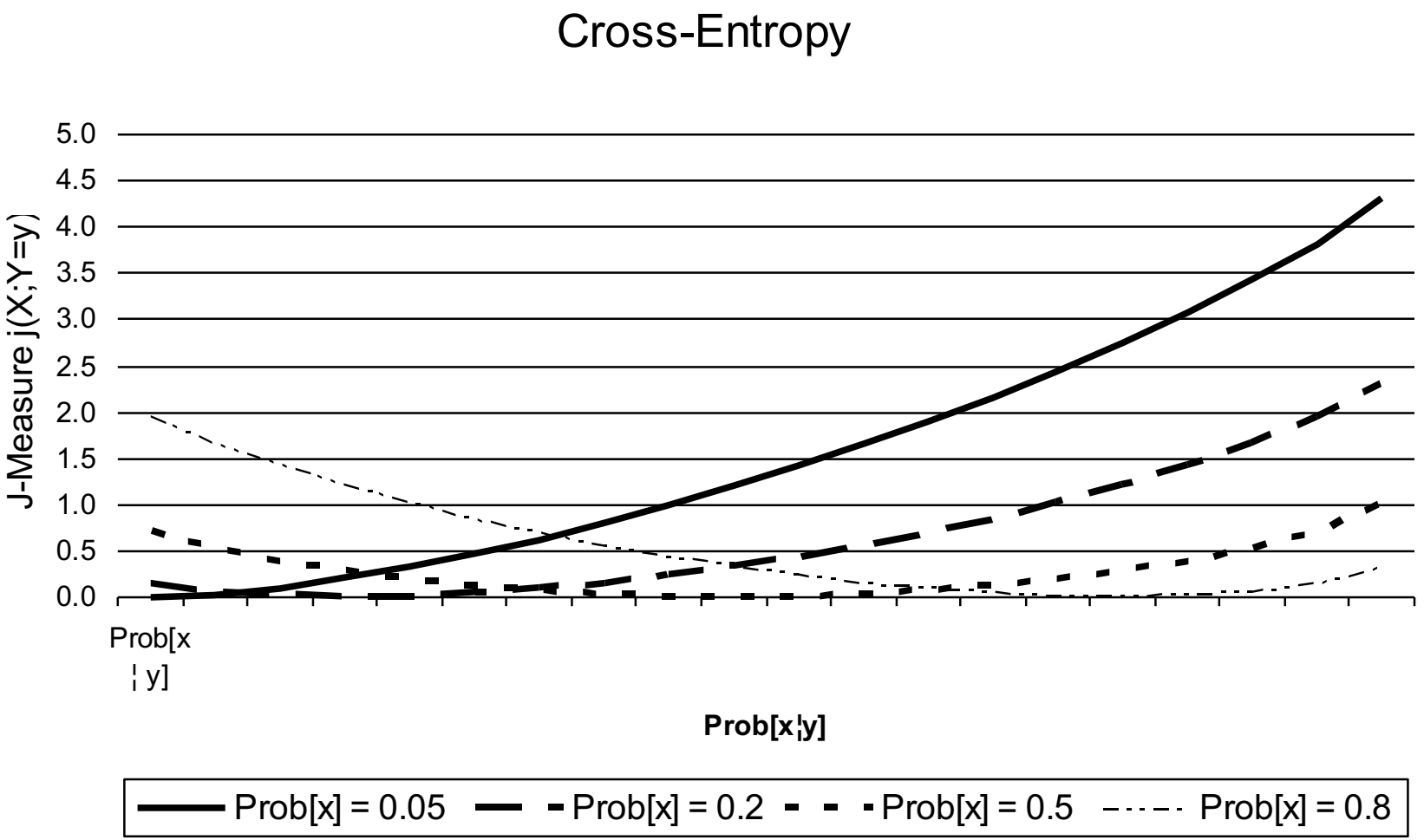
J-Measure

- From a rule IF Y=*y* THEN X=*x*
  the information content of the rule (in bits of information) is defined by J(x;Y=*y*) (Smyth & Goodman)
  of the value of the cross-entropy

$$J[X;Y=y] = Prob[x|y] \cdot \log_2 \left(\frac{Prob[x|y]}{Prob[x]} \right)$$
$$+ (1 - Prob[x|y]) \cdot \log_2 \left(\frac{1-Prob[x|y]}{1-Prob[x]} \right)$$

J-Measure

Graphically, we have



J-Measure

- We can also defined a max value for the J-measure for any specializing a given rule (adding further terms to the left-hand side) which is:

$$J_{max} = Prob[y] \cdot \max\left(Prob[x|y] \cdot \frac{1}{Prob[x]}, (1 - Prob[x|y]) \cdot \frac{1}{1 - Prob[x]}\right)$$

- Thus if a rule as a J-measure of 0.35 and the value of J_{max} is also 0.35, you cannot have benefit by adding further terms to the left-hand side