

Measuring the *Support* of an Itemset

- An itemset A is only interesting if it occurs in a significant number of transactions
 - In other words, we want |T(A)| to be "large enough"
 Notation: we use #(A) to represent |T(A)|: in other wo
 - Notation: we use #(A) to represent |T(A)|; in other words, it's the number of transactions that include all items of itemset A
- The support of itemset A is defined as #(A) / #(Ø)
 #(Ø) is just the total number of all transactions
- We say itemset A is supported if support(A) > s₀ where s₀ is a constant that the user gets to choose
 - s_0 is typically a small fraction of a percent
 - "A *is supported*" is another way of saying that the items of A appear *together* in a significant number of transactions

Goal: Find all Supported Itemsets

Outline of algorithm:

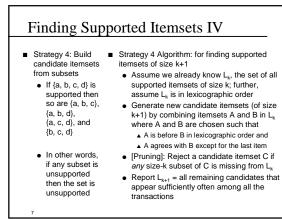
- Choose a set of candidate itemsets
- Run through all the transactions and count how many times each candidate itemset appears
- Itemsets that appear sufficiently often are reported
- This algorithm should work as long as our set of candidate itemsets is not too large
- Strategy 1: Check all possible itemsets
 - For 1000 items (not unusually large), there are 2¹⁰⁰⁰ subsets
 - $2^{1000} = (2^{10})^{100} \approx (10^3)^{100} = 10^{300}$
 - We can't possibly check this many itemsets

Finding Supported Itemsets II

- Strategy 2: Check only the itemsets that actually occur
 - A single transaction might include, say, 40 items
 - We can generate candidate itemsets by looking at subsets of these 40 items; we can do this for each transaction
 - Number of subsets (for one typical transaction) = 2^{40} = $(2^{10})^4 \approx (10^3)^4$ = 10^{12} = 1 trillion
 - · We can't check this many itemsets either

Finding Supported Itemsets III

- Strategy 3: Build candidate itemsets by adding one item at a time
 - Note that itemset {a, b, c, d} is supported only if itemset {a, b, c} is supported
 - ▲ In other words, once we find an unsupported itemset, adding an additional item will only make it less supported
 - Algorithm for finding supported itemsets of size k+1:
 Assume we already know L_k, the set of all supported itemsets of size k
 - ▲ Generate new candidate itemsets (of size k+1) by looking for transactions that contain some A∈ L_k and then adding one more item to A from that transaction
 - ▲ Run through all the transactions and count how many times each candidate itemset appears
 - \blacktriangle Report L_{k+1} = all candidates that appear sufficiently often



Example: Strategy 4

- Let L₃ = { {a, b, c}, {a, b, d}, {a, c, d}, {a, c, e}, {b, c, d} }
- Generated candidates (before pruning):
 - {a, b, c, d}
 - {a, c, d, e}
 - Candidates after pruning:
- {a, b, c, d}
- Note that {a, c, d, e} was pruned because {a, d, e} is missing from L₃
 - {c, d, e} is also missing from L₃

Association Rules

- An association rule has the form A→B where A and B are itemsets
- Each association rule has a confidence factor
 This indicates how often
 - the rule appears to have "worked" in the dataset The confidence factor for
 - A→B is defined as ▲ #(A∪B) / #(A)
 - ▲ In other words: Of all the times that A appears in transactions, what fraction also includes the items of B

Example: ■ {bread, milk} → {eggs}

- The rule is a way of expressing the idea that people who buy bread and
- milk are likely to also buy eggs Example confidence factor:
- (the number of transactions involving bread, milk, and eggs) divided by (the number of transactions involving just
- bread and milk)
 #({bread, milk, eggs}) /
 #({bread, milk})

Using Association Rules

- For a rule A→B, A is the *antecedent* and B is the *consequent*
- By finding association rules, we can answer useful questions
 Find all rules with Coke as a consequent
 - ▲ What can done to boost Coke sales?
 - Find all rules with bagels in the antecedent
 What products might be affected if bagels are discontinued?
 - Find all rules with sausage in the antecedent and mustard as the consequent
 What should be placed near sausage to encourage mustard
 - sales?

Confidence vs. Support

- A rule with a high confidence factor is not necessarily useful
 Example: Suppose there is exactly one transaction that includes both Poptarts and lobster and that transaction also includes pizza
 - The confidence factor for {lobster, Poptarts} → {pizza} is #({Poptarts, lobster, pizza}) / #({Poptarts, lobster}) = 1
 - This rule has high confidence, but low support
- The support for a rule A→B is defined as support(A→B) = support(A ∪ B) = #(A ∪ B) / #(∅)

Suppose we already know All supported itemsets The value of support(C) for

Reporting the Useful Association Rules

- The value of support(C) for each supported itemset C
- Observe that if C is a supported itemset and C=A∪B then
 - A→B is an association rule that is supported,
 - A is supported (so we
 - know support(A)), and
 - the confidence factor for A→B is given by support(C) / support(A)
- fractions of a percent) • 0.3 {bread} • 0.25 {eggs}
 - 0.2 {milk}
 - 0.15 {bread, milk}
 - 0.10 {eggs, milk}
 - 0.08 {bread, eggs}
 0.05 {bread, eggs, milk}
- Find the association rules involving all of bread, eggs, and
 - milk and determine the confidence factor for each rule