CSCI 4022 Fall 2021
Representing Itemsets

Example: Suppose our **items** are {milk, coke, pepsi, beer, juice} and we want a support threshold of $s = 3$ baskets.

$B_1 =$	{m,c,b}	$B_5 =$	{m,p,j}
$B_2 =$	{m,b}	$B_6 =$	{c,j}
$B_3 =$	{m,p,b}	$B_7 =$	{m,c,b,j}
$B_4 =$	{c,b,j}	$B_8 =$	{b,c}

What are all of the frequent itemsets?

CSCI 4022 Fall 2021
Representing Itemsets

Example: Suppose our **items** are {milk, coke, pepsi, beer, juice} and we want a support threshold of $s = 3$ baskets.

$B_1 =$	{m,c,b}	$B_5 =$	{m,p,j}
$B_2 =$	{m,b}	$B_6 =$	{c,j}
$B_3 =$	{m,p,b}	$B_7 =$	{m,c,b,j}
$B_4 =$	{c,b,j}	$B_8 =$	{b,c}

What are all of the frequent itemsets?

Solution:

1. Size 1: {{m},{c},{b},{j}}. Not {~~p~~}. 4
2. Size 2: 10 possibilities! (Since that's ~~5~~ 4 choose 2). In this case, {m,b},{c,j} and {c,b} are both frequent. 6 → $\binom{4}{2} = \frac{4!}{2} = 6$
3. Size 3: None... could we have known this already?

if mcb had 3 baskets... mc, mb, cb had 3 or more

Association Rules

Definition: An *association rule* is an *if-then* rule about the contents of baskets. We denote

$$\underline{\{i_1, i_2, \dots i_k\} \to i_j}$$

to represent "if a basket contains all of $i_1, i_2, \dots i_k$, it is *likely* to contain j as well." item j given already have all of 1, 2, 3, 4, . . . , k

In practice, there will of course be lots of rules, so we want to find the most significant ones. This requires a notion of *confidence.*

It also turns out that some rules, like $X \to milk$ will inevitably have high confidence for many itemsets X simply because milk is popular. Having many high-confidence associations isn't necessarily actionable, so we also want a measure of *interest*.

Association Rules

Definition: The *confidence* of the association rule $I \to J$ is the ratio of the support for $I \cup \{j\}$ to the support for I.

baskets w/ I and J

$$conf(I \to J) = \frac{support(I \cup \{j\})}{support(I)}$$

P(J | I is there)

baskets w/ I

P(A|B): "Prob. of A given B"; P(A) after we gain of knowledge of B.

$$P(A|B) = \frac{P(\text{both}) \to P(A \cap B)}{P(B)} = \frac{P(\text{both sets})}{P(I)}$$

Association Rules

Definition: The *confidence* of the association rule $I \to J$ is the ratio of the support for $I \cup \{j\}$ to the support for I.

recall:

$P(A|B) = P(A)$ if A & B are independent

$$conf(I \to J) = \frac{support(I \cup \{j\})}{support(I)}$$

$\approx P(J|I)$

Definition: The *interest* of the association rule $I \to J$ is the difference between its confidence and the fraction of baskets that contain j

$$interest(I \to J) = conf(I \to J) - P(j)$$

$P(J|I) - P(J)$

Association Rules

Definition: The *confidence* of the association rule $I \to J$ is the ratio of the support for $I \cup \{j\}$ to the support for I.

$$conf(I \to J) = \frac{support(I \cup \{j\})}{support(I)}$$

This is a lot like *conditional probability*. Recall that $P(A|B) = \frac{P(both)}{P(B)} = \frac{P(A \cap B)}{P(B)}$. Then $support(I \cup \{j\})$ is the count of all the baskets that have *both* all of I and j as the numerator: so it's really like an intersection of those two sets! $conf(I \to J)$ behaves like probability of J **given** I.

Definition: The *interest* of the association rule $I \to J$ is the difference between its confidence and the fraction of baskets that contain j

$$interest(I \to J) = conf(I \to J) - P(j)$$

Association Rules

Definition: The *confidence* of the association rule $I \to J$ is the ratio of the support for $I \cup \{j\}$ to the support for I.

$$conf(I \to J) = \frac{support(I \cup \{j\})}{support(I)}$$

This is a lot like *conditional probability*. Recall that $P(A|B) = \frac{P(both)}{P(B)} = \frac{P(A \cap B)}{P(B)}$. Then $support(I \cup \{j\})$ is the count of all the baskets that have *both* all of I and j as the numerator: so it's really like an intersection of those two sets! $conf(I \to J)$ behaves like probability of J **given** I.

Definition: The *interest* of the association rule $I \to J$ is the difference between its confidence and the fraction of baskets that contain j

$$interest(I \to J) = conf(I \to J) - P(j)$$

In the probability sense, this is a bit like $P(J|I) - P(J)$

Frequent Itemsets

Example: We can't escape Walmart. Consider the association rule {m,b}→ c. What are the confidence and interest?

$B_1 =$ {m,c,b} $B_5 =$ {m,p,j}
$B_2 =$ {m,b} $B_6 =$ {c,j}
$B_3 =$ {m,p,b} $B_7 =$ {m,c,b,j}
$B_4 =$ {c,b,j} $B_8 =$ {b,c}

$$\frac{|\{m,b\} \cup \{c\}|}{|\{m,b\}|} = 2/4$$

$$2/4 - \frac{|\{c\}|}{total} = 2/4 - 5/8 = -.125$$

FEBRUARY 4, 2013

Police Arrest Florida Man For Drunken Joyride On Motorized Scooter At Walmart

Frequent Itemsets

Example: We can't escape Walmart. Consider the association rule {m,b}$\rightarrow$ c. What are the confidence and interest?

B_1=	{m,c,b}	B_5=	{m,p,j}
B_2=	{m,b}	B_6=	{c,j}
B_3=	{m,p,b}	B_7=	{m,c,b,j}
B_4=	{c,b,j}	B_8=	{b,c}

Solution:

1. Support {m,b}=4; Support {m,b,c}=2. $conf(\{m,b\} \rightarrow c) = \frac{2}{4}$
2. Interest: $interest(\{m,b\} \rightarrow c) = \frac{2}{4} - \frac{5}{8} = -0.125$

FEBRUARY 4, 2013

Police Arrest Florida Man For Drunken Joyride On Motorized Scooter At Walmart

Frequent Itemsets

Example: We can't escape Walmart. Consider the association rule {m,b}$\rightarrow$ c. What are the confidence and interest?

B_1=	{m,c,b}	B_5=	{m,p,j}
B_2=	{m,b}	B_6=	{c,j}
B_3=	{m,p,b}	B_7=	{m,c,b,j}
B_4=	{c,b,j}	B_8=	{b,c}

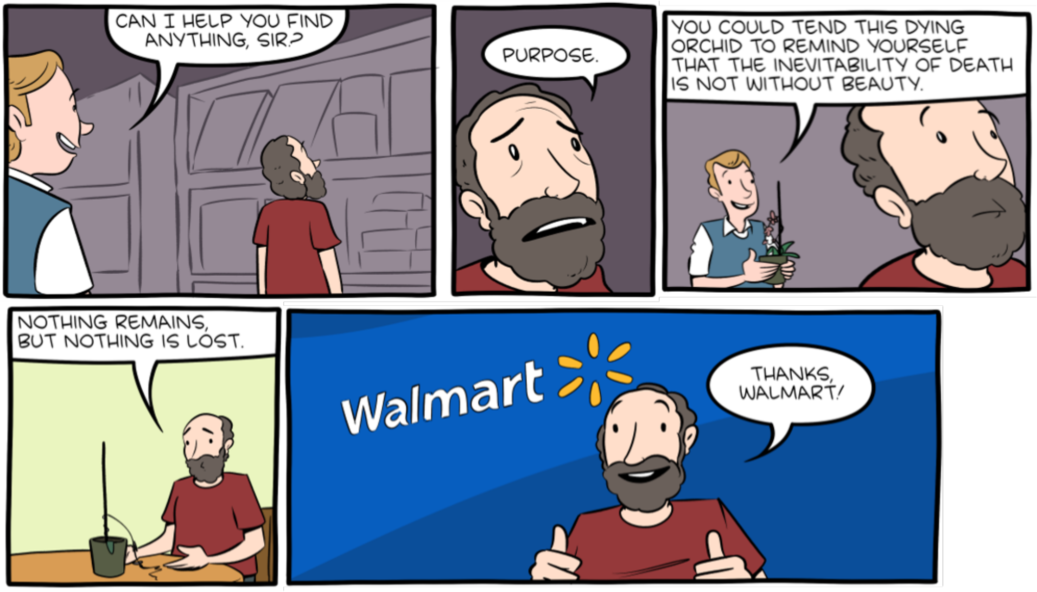

FEBRUARY 4, 2013

Police Arrest Florida Man For Drunken Joyride On Motorized Scooter At Walmart

Solution:

1. Support {m,b}=4; Support {m,b,c}=2. $conf(\{m,b\} \rightarrow c) = \frac{2}{4}$
2. Interest: $interest(\{m,b\} \rightarrow c) = \frac{2}{4} - \frac{5}{8} = -0.125$

Coke is pretty popular, since it's in 5/8 baskets. So the rule that half of the milk+beer baskets also have coke is very uninteresting!


CAN I HELP YOU FIND ANYTHING, SIR?
PURPOSE.
YOU COULD TEND THIS DYING ORCHID TO REMIND YOURSELF THAT THE INEVITABILITY OF DEATH IS NOT WITHOUT BEAUTY.
NOTHING REMAINS, BUT NOTHING IS LOST.
Walmart
THANKS, WALMART!

Itemsets: Algorithmic Base

Goal: find all association rules with support $\geq s$ and confidence $\geq c$ (support of all items on the left-hand side, that is)

Note that if the set $J = \{i_1, i_2, i_3 \ldots j\}$ is frequent (support $\geq s$), then it must necessarily be the case that the smaller set $I = J - \{j\} = \{i_1, i_2, \ldots, i_k\}$ is at least as frequent.

Gives a sense of how we might mine association rules using a top-down approach, by considering subsets of frequent itemsets.

Association Rules: Top down

Suppose we have an assoc. rule $I \rightarrow j$ with support s, and high confidence c. Then $I \cup j$ has support of at least cs because

$$conf(I \rightarrow j) = \frac{support(I \cup \{j\})}{support(I)} \Leftrightarrow c = \frac{support(I \cup \{j\})}{s}$$

So we could...

9000 100

1. Find all itemsets with support at least cs (Set 1) error things here
2. Find all itemsets with support at least s (Set 2, which will be a subset of Set 1 since $s \geq cs$)
3. Loop: For each itemset J of Set 1...
   - 3.1 Consider the $support(J) = s_2$ (we would have previously computed this)
   - 3.2 For each element $j \in J$, remove j and compute $support(J - \{j\}) = s_1$
   - 3.3 If $s_1/s_2 \geq c$ then $J - \{j\} \rightarrow j$ is an acceptable association rule.

Association Rules

Example: We can't escape Walmart. For $s = 3$ and confidence of $c = .75$, what are the association rules?

$B_1 =$	{m,c,b}	$B_2 =$	{m,p,b}	$B_3 =$	{m,p,j}	$B_4 =$	{m,c,b,j}
$B_5 =$	{m,c,b,n}	$B_6 =$	{c,b,j}	$B_7 =$	{c,j}	$B_8 =$	{b,c}

$3 \cdot .75 \rightarrow$ everything of support $\geq$.

$|mbc| \rightarrow 3$

$|mbc|$ 3 numerator

$-m$ $-b$ $-c$

$|bc| = 5$ $|mc| = 3$ $|mb| = 4$

denom

Association Rules

Example: We can't escape Walmart. For $s = 3$ and confidence of $c = .75$, what are the association rules?

$B_1 =$	{m,c,b}	$B_2 =$	{m,p,b}	$B_3 =$	{m,p,j}	$B_4 =$	{m,c,b,j}
$B_5 =$	{m,c,b,n}	$B_6 =$	{c,b,j}	$B_7 =$	{c,j}	$B_8 =$	{b,c}

Solution:

1. Item sets: {{m,b},{b,c},{c,j}, {c,m}, and {m,c,b}}

Association Rules

Example: We can't escape Walmart. For $s = 3$ and confidence of $c = .75$, what are the association rules?

$B_1 =$	{m,c,b}	$B_2 =$	{m,p,b}	$B_3 =$	{m,p,j}	$B_4 =$	{m,c,b,j}
$B_5 =$	{m,c,b,n}	$B_6 =$	{c,b,j}	$B_7 =$	{c,j}	$B_8 =$	{b,c}

Solution:

1. Item sets: {{m,b},{b,c},{c,j}, {c,m}, and {m,c,b}}
2. Candidates: (a subset):

From {m,b}:	$m \to b$ $\|\{b,m\}\| = 4; \|\{m\}\| = 5;$ $c = 4/5$	$b \to m$ $\|\{b,m\}\| = 4; \|\{b\}\| = 6;$ $c = 4/6$	
From {m,c,b}:	$\{m,c\} \to b$ $\|\{m,c,b\}\| = 3; \|\{m,c\}\| = 3;$ $c = 1$ $\{b,c\} \to m$ $\dots$	$\{m,b\} \to c$ $\|\{m,c,b\}\| = 3; \|\{m,b\}\| = 4;$ $c = 3/4$ $\{c\} \to \{b,m\}$ $\dots$	...and more.

3. Remove rules that don't pass the c threshold.

Market Basket: The Bottleneck

So we can find rules once we have frequent item counts… but how do we get those? First, we need to talk about how to represent market-basket data.

basket 1: { . . . },
bask. 2 { - - - }
basket 3 { . - - }

Some assumptions:

- The data is stored in a file basket-by-basket
- Many more baskets than items per basket

Generally, the size of the file of baskets is so large that it does not all fit in main memory
Example: A file of market-basket data might begin: {23, 456, 1001}{3, 18, 92, 145}{… and so on…

Obviously, smaller files could be stored as CSV or similar tabular formats, and

May be a preliminary step of data processing to encode names of items as numbers (e.g., through a bar code, or hash table)

Worst Case Complexity

$\binom{10000}{2}$ $\Theta(10{,}000^2)$

$\binom{10000}{3}$ $\Theta(10{,}000^3)$

We want to generate subsets (pairs, for example) of items in a basket and see how frequent they are.

As the size of the subsets to generate increases, time required also grows:

$\frac{n^k}{k!}$ is roughly the time to generate subsets of size k from a basket with n items. That's maybe a lot! But we're saved by a couple nice facts, plus algorithms:

1. Often, we only really care about small frequent itemsets, so k never grows beyond 2 or 3. (Is a size 10 frequent itemset operationally useful?)

2. If k is large, then we can usually eliminate many items in each basket as impossible candidates, so n decreases as k increases.

Memory Requirements

All frequent itemset algorithms require us to maintain different counts as we make a pass over the data (e.g., how many baskets does a pair appear in, to compute its support)

Need to be able to store the counts all in main memory

(otherwise, when we update some of the counts, it will thrash and run horribly slowly)

Example: Suppose we have a computer with 8 GB of main memory. How many different kinds of items can we store counts of pairs of?

Memory Requirements

All frequent itemset algorithms require us to maintain different counts as we make a pass over the data (e.g., how many baskets does a pair appear in, to compute its support)

Need to be able to store the counts all in main memory

(otherwise, when we update some of the counts, it will thrash and run horribly slowly)

Example: Suppose we have a computer with 8 GB of main memory. How many different kinds of items can we store counts of pairs of? **Solution:**

- n items, means $C(n,2) \approx n^2/2$ possible pairs.
- Say we use 4-byte integers, we have $2n^2$ B used.
- At $8GB$, and 2^{30} B/GB, we have

$$2n^2 \leq 8 \cdot 2^{30} \implies n^2 < 2^{32} \implies n < 2^{16} = 65536.$$

Memory Format

Note that that's just to save a single item count for all *pairs* of items. Not to load the basket data! And it's larger for $C(n,3)$ to count frequent triples.

Suppose we wanted to score the entire $n \times n$ matrix for pair counts. Thats likely pretty inefficient:

U_{ij} = count of $|\{i,j\}|$

1. "A customer who bought X also bought Y" is **symmetric**, so we could store only half the matrix.
2. If the items are strings like "bread," how do we map this to an elements of the matrix?

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & \dots & u_{1n} \\ \mathbf{0} & u_{22} & u_{2,3} & \dots & u_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & 0 & u_{n,n} \end{bmatrix}$$

We may not even know beforehand what all elements we have in the space are. If our data set is purchases in 2018 at Safeway, there may be new products we aren't aware of, or products with zero sales.

Array vs. Matrix

1. "A customer who bought X also bought Y" is **symmetric**, so we could store only half the matrix.

We can avoid storing all those zeros! Instead:

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & \dots & u_{1n} \\ 0 & u_{22} & u_{2,3} & \dots & u_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & 0 & \end{bmatrix}$$

1. Only store the upper-triangular elements in a 1D **triangular array**
2. the diagonal elements represent when X was bought with X, so that yields no useful information (beyond raw count of X), so we skip those.

items 2,1 item 3,1 items n,1

Our goal: an array a that has entries:

$$\begin{bmatrix} . & a[1] & a[2] & \dots & \dots & a[n-1] \\ & . & a[n] & a[n+1] & & a[n-1+n-2] \\ & & & & \dots & \dots \end{bmatrix}$$

items 3,2 items n,2

Triangular Indexing

Our goal: a function that gives us the indices of the array a. This has input of {row, col} and entries like:

i	j	a
1	2	1
1	3	2
1	n	n-2
2	3	n-1
2	n	n-2+(n-2)
3	4	2n-3
3	n	n-2+(n-2)+(n-3)

$$\begin{bmatrix} . & a[1] & a[2] & \dots & \dots & a[n-1] \\ & . & a[n] & a[n+1] & & a[n-1+n-2] \\ & & & . & \dots & \dots \end{bmatrix} \begin{matrix} \to n-1 \\ \to n-2 \\ \to n-3 \end{matrix}$$

what do I do with (2, 4)

$(n-1) +$

The function:

$$a[k] = \boxed{(i-1)\left(n - \frac{i}{2}\right)} + j - i$$

$i=1, j=2$

$(1-1)\left(n - \frac{1}{2}\right) + 2 - 1 = 1$

as j inc, i constant we +1

will store item counts for the pair i, j, where $1 \le i < j \le n$.

0-indexed Triangular Array

The prior formulas was 1-indexed for both a and i, j.

1. To zero-index both a, add a -1 at the end of this.
2. To zero-index i and j, replace i, j with $i + 1, j + 1$.

np.triu_indices

$i > j$

Doing both results in:

$$a[k] = (i)\left(n - \frac{i+1}{2}\right) + j - i - 1$$

will store item counts for the pair i, j, where $0 \leq i < j \leq n$.

what if you had $a[k]$, what are i, j?

Array vs. Matrix

tri-array: $n^2/2$ $\binom{n}{2} - n$

1. "A customer who bought X also bought Y" is **symmetric**, so we could store only half the matrix.

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & \dots & u_{1n} \\ \mathbf{0} & u_{22} & u_{2,3} & \dots & u_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & 0 & \end{bmatrix}$$

\$7

We could alternatively use **triples**,

item 1 ; item 2 17 times

[1, 2, 17]

1. Store counts as triples $[i, j, c]$ where c is the count of $\{i, j\}$, $j > i$.
2. A hash table with i and j as a double search key can quickly determine whether or not there is already a triple for a given $\{i, j\}$ pair, which we could them increment as we scan data.

worst case:

$\binom{n}{2} \cdot 3$

all i, j

Array vs. Matrix

1. "A customer who bought X also bought Y" is **symmetric**, so we could store only half the matrix.

We could alternatively use **triples**,

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & \dots & u_{1n} \\ \mathbf{0} & u_{22} & u_{2,3} & \dots & u_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & 0 & \end{bmatrix}$$

1. Store counts as triples $[i, j, c]$ where c is the count of $\{i, j\}$, $j > i$.

(i, j, k, c)

2. A hash table with i and j as a double search key can quickly determine whether or not there is already a triple for a given $\{i, j\}$ pair, which we could them increment as we scan data.

Benefit: don't have to store infor on pairs with 0 counts
Cost: Now storing 3 integers instead of just one for each count, plus overhead of hash table
Result: in general, the triangular matrix is a little better if at least 1/3 of the $C(n, 2)$ possible pairs will actually appear in some basket.

Array vs. Matrix

2. If the items are strings like "bread," how do we map this to an elements of the triangular array/matrix?

- It's more space-efficient to represent items as consecutive integers $1, 2, \ldots n$ where $n :=$ $\#$ of distinct items.
- Can use a hash table to translate items as they appear in a file to integers.

Array vs. Matrix

2. If the items are strings like "bread," how do we map this to an elements of the triangular array/matrix?

- It's more space-efficient to represent items as consecutive integers $1, 2, \ldots n$ where $n :=$ $\#$ of distinct items.
- Can use a hash table to translate items as they appear in a file to integers. Note that these hash values aren't going to be necessarily $1, 2, \ldots n$. The hash table is there as a lookup device to map strings to elements in the triangular array!

row [317, Strings "tasty greens", "bread", hash 317, item # 17, 52]

Array vs. Matrix

2. If the items are strings like "bread," how do we map this to an elements of the triangular array/matrix?

- It's more space-efficient to represent items as consecutive integers $1, 2, \ldots n$ where $n :=$ $\#$ of distinct items.
- Can use a hash table to translate items as they appear in a file to integers.

- So the algorithm will read a data file, and as we hash items...
  1. If it (string) is already in the hash table, obtain its integer code and increment its count (in hash table or in tri array.)
  2. If it's not in the hash table, assign it the next available number - keeping a running count of how many distinct elements we've seen - and then enter the item and its code in the hash table.
  3. Result: table with each row being indexed by a hash value, holding an item or items with that hash value, their names, and their item numbers.

Monotonicity

We now need to also ask how to compute frequent pairs in a time-efficient manner.
Fact: if an itemset I is frequent, then so is eery subset of I.

Proof:

Monotonicity

We now need to also ask how to compute frequent pairs in a time-efficient manner.
Fact: if an itemset I is frequent, then so is eery subset of I.

Proof: By *contradiction*

- Suppose not, or there exists frequent itemset I with threshold s and itemset J a subset of I that is not frequent.
- Then the support of J is less than s, or fewer than s baskets contain J.
- But any basket that contains I contains J, since J is a subset. So $support(J) > support(I) > s$...
- which is a contradiction!

Monotonicity

Fact: if an itemset I is frequent, then so is eery subset of I.

Definition: Given a support threshold s, a frequent itemset I is *maximal* if no superset of I is also frequent.

So we if we list only *maximal* itemsets, we'll know...

1. All subsets of a maximal itemset must also be frequent
2. No set that is *not* a subset of some maximal itemset can possible be frequent

In other words, the set of maximal frequent itemsets is the most concise - or minimal - way to represent all frequent items.

Maximal Sets

Example: We can't escape Walmart. For $s = 3$, what itemsets are maximal?

$B_1 =$	{m,c,b}	$B_2 =$	{m,p,b}	$B_3 =$	{m,p,j}	$B_4 =$	{m,c,b,j}
$B_5 =$	{m,c,b,n}	$B_6 =$	{c,b,j}	$B_7 =$	{c,j}	$B_8 =$	{b,c}

Maximal Sets

Example: We can't escape Walmart. For $s = 3$, what itemsets are maximal?

$B_1 =$	{m,c,b}	$B_2 =$	{m,p,b}	$B_3 =$	{m,p,j}	$B_4 =$	{m,c,b,j}
$B_5 =$	{m,c,b,n}	$B_6 =$	{c,b,j}	$B_7 =$	{c,j}	$B_8 =$	{b,c}

Solution:

1. Item sets: {{b},{c},{j},{m}, {m,b},{b,c},{c,j}, {c,m}, and {m,c,b}}

Maximal Sets

Example: We can't escape Walmart. For $s = 3$, what itemsets are maximal?

$B_1=$	{m,c,b}	$B_2=$	{m,p,b}	$B_3=$	{m,p,j}	$B_4=$	{m,c,b,j}
$B_5=$	{m,c,b,n}	$B_6=$	{c,b,j}	$B_7=$	{c,j}	$B_8=$	{b,c}

Solution:

1. Item sets: {{b},{c},{j},{m}, {m,b},{b,c},{c,j}, {c,m}, and {m,c,b}}
2. {m,c,b} and {c,j} are the maximal itemsets.

What about bigger counts?

We talked a lot about counting pairs, but what about triples, quadruples, or larger frequent itemsets?

1. In practice, we pick support thresholds to be high enough that we do not have too many frequent itemsets - this makes it easier to have actionable information.
2. **Monotonicity** is important! If there is a frequent triple, it must contain 3 frequent pairs.
   - And then any frequent quadruple must contain 4 frequent triples and $\binom{4}{2} = 6$ frequent pairs.
   - ... and so on
   - As a result, we expect to find more frequent pairs than triples, more triples than quadruples, and so forth.

What about bigger counts?

We talked a lot about counting pairs, but what about triples, quadruples, or larger frequent itemsets?

1. In practice, we pick support thresholds to be high enough that we do not have too many frequent itemsets - this makes it easier to have actionable information.
2. **Monotonicity** is important! If there is a frequent triple, it must contain 3 frequent pairs.
   - And then any frequent quadruple must contain 4 frequent triples and $\binom{4}{2} = 6$ frequent pairs.
   - ... and so on
   - As a result, we expect to find more frequent pairs than triples, more triples than quadruples, and so forth.

Example: We don't have to search for, allocate memory for, or even consider the frequent itemset $\{m, c, b\}$ unless we've already observed **all** of $\{m, c\}$, $\{m, b\}$, and $\{c, b\}$

What about bigger counts?

Example: We don't have to search for, allocate memory for, or even consider the frequent itemset $\{m, c, b\}$ unless we've already observed **all** of $\{m, c\}$, $\{m, b\}$, and $\{c, b\}$

...and that's a really good thing!

There are many more *candidate* triples than pairs.

Example: for $n = 10$ items, how many pairs are there? How many triples?

What about bigger counts?

Example: We don't have to search for, allocate memory for, or even consider the frequent itemset $\{m, c, b\}$ unless we've already observed **all** of $\{m, c\}$, $\{m, b\}$, and $\{c, b\}$

...and that's a really good thing!

There are many more *candidate* triples than pairs.

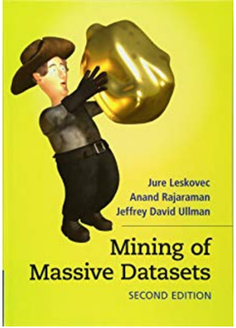
Example: for $n = 10$ items, how many pairs are there? How many triples? **Solution:** $C(10, 2) = 45$ pairs, but $C(10, 3) = 120$ triples. The order is $\mathcal{O}(10^k)$ for itemsets of size k!

at least up to $k = n/2$...

Acknowledgments

Next time: the A Prior algorithm, to glue it all together.

Some material is adapted/adopted from Mining of Massive Data Sets, by Jure Leskovec, Anand Rajaraman, Jeff Ullman (Stanford University) `http://www.mmds.org`



Special thanks to Tony Wong for sharing his original adaptation and adoption of slide material.