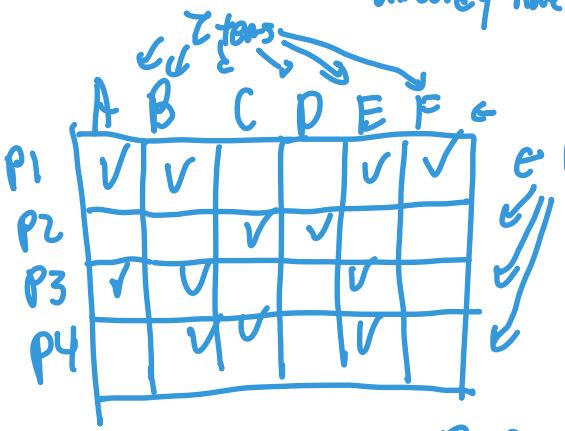
Market Basket Analysis

Provides Answer - How likely customer will buy product A if they already have products B, C, etc.



Idea → People who buy {A, B} likely buy {E} as well.

Definitions

Support: Number of baskets (rows) containing the set {u, v, ...}
(can also be a percent or fraction)

Support Threshold "s": sets {u, v, ...} that appear in ≥ s baskets are Frequent Itemsets

$$\text{Confidence}(I \to j) = \frac{\text{support}(I \cup j)}{\text{support}(I)} \to \frac{\text{baskets that satisfy } I \to j}{\text{baskets with } I}$$

Confidence Threshold "c": Sets {i_1, i_2 ...} where Confidence(I, j) ≥ c

Association Algebra

1. Find sets I that satisfy support threshold
2. From above result, calculate confidence and only keep the best

Logical Error: If {a, b, c} → {y} has ST ≥ s and CT ≥ c, then subset of the set also satisfies the conditions

Correct: There is NO SPEEDUP for Apriori :(

Proof: Let $s(I) = \text{support}(I)$, $c(I) = \text{confidence}(I)$, $c(I, y) = \text{confidence}(I \| y)$

Let $ST_0 = s(\{A,B,C\})$, $CT_0 = c(\{A,B,C\}) = \dfrac{s(\{A,B,C,Y\})}{s(\{A,B,C\})}$

$S_n < S_{n+1}$ because narrowing search space

Case 1: $\{A,B,C\}$ does not satisfy support threshold
$\{A,B\}$ MAY but... then case 2.

Case 2: $\{A,B,C\}$ does not satisfy confidence.

Prove: $c(\{A,B,C\}) \geq c(\{A,B\})$

AB Y

AB Y

AB Y

A B C

The "proof" is invalid as well.

Truth (rip): There is no real algo optimization for Apriori

2 types of relationships

$\{a,b\} \rightarrow \{a,b,c\}$: Not true bc $s(\{a,b,c\}) \leq s(\{a,b\})$

$\{a,b,c\} \rightarrow \{a,b\}$: Not true bc universal generalization is invalid

Say we had 

Then $c(\{A,B\}) \geq c(\{A,B,C\})$

Not $\{a,b\} \rightarrow$ Not $\{a,b,c\}$ or Not $\{a,b,c\} \rightarrow$ Not $\{a,b\}$ are contrapositive of the two cases reversed

Any of the $x \rightarrow$ Not y or Not $x \rightarrow y$ do not make logical sense

Support Algebra rule!

If $\{a,b\}$ does not satisfy support, then $\{a,b,c\}$ will not either.

Proof: Since $\{a,b\}$ is a subset of $\{a,b,c\}$, then there cannot be more counts of a set than a subset.

Apriori Algo

Key Idea: If $\{a,b,c\}$ satisfies support, then all subsets do as well

Take Frequent Items (Items that appear $\geq$ st times) $O(N)$

Take Frequent Pairs (Pairs that appear $\geq$ st times) $O(N^2)$

Take F Triples, F Quads, etc $O(N^3)$ why?

Notation: A_n = set of length n

Recursively Define Merge: saves "generating" time

Generating Triples is $O(N^3)$

Merge A_{n-1}, A_1

Merge candidate sets is $O(N_n)$

BUT! this is worst case. Normally we expect the A and B sets to be pruned so it will be much faster

Can further be optimized by repruning the Z

(say $\{n,n\}$ is bad, then $\{a,n\}$ $\{b,n\}$ should not be included)

Say we had candidate sets $Z_n = \{A_1B_1, A_1B_2, \text{etc}\}$

Then run support threshold, generate A_n :

Generate all candidate sets $O(2^n)$ but optimized.

Confidence :

No real rules.

Apriori

Run through support, calculate confidence, choose best ones.

Generally, $k=2$ has most memory (given reasonable support 1%)

PCY -> Park-Chen-Yu Algo

Observation: In pass 1 of Apriori, most memory is idle (we only store item cts)

Can we use idle memory to reduce memory in pass 2? (pairs)

Pass 1 of PCY: Maintain hashtable with as many buckets as fit in memory

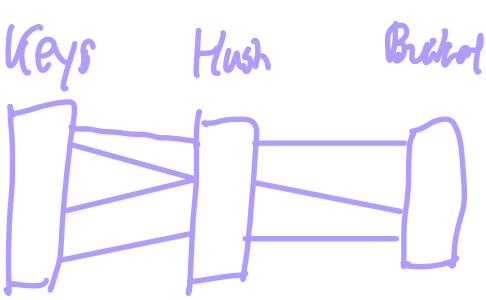
Keep a count for each bucket into which pairs of items are hashed

For each bucket, keep only the count ... not the actual pairs

pseudo code

```
for (each row)
    for(item in row)
      Freq[item]++
    for(p1 in row)
      for(p2 in row)
        Generate Hash
        bucket(hash)++
```

Hash bucket



A faster way to hash/save memory

If no bucket, all keys in memory

KEY OBSERVATION -> If a bucket contains a frequent pair, the bucket is surely frequent

However, a bucket without a FP is not always bad.

BUT: For a bucket with count < s, no pairs are frequent ☺

↓

Even if a pair has 2 frequent items, pairs in this bucket are eliminated as candidate pairs

Proof: Even if count(bucket(h)) all belong to {a, b}, count < s

Between Passes: Replace buckets by bitset (0/1)

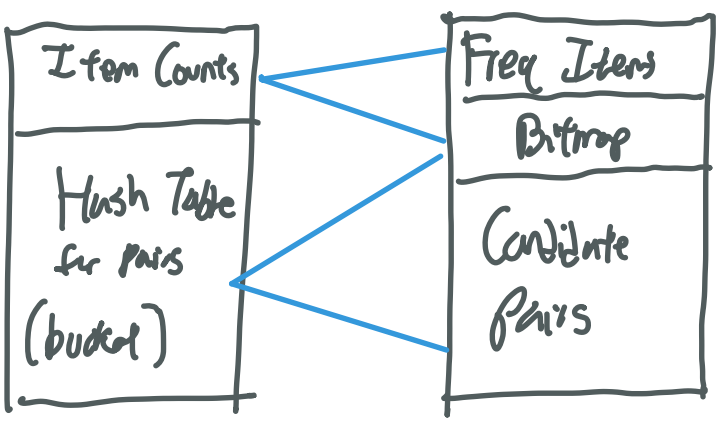
Memory reduction by 1/32 because 4-bytes -> 1 bit

Also list frequent items

Pass 2: Only count pairs in frequent buckets

Candidate Pair Rules {i, j}

1. i and j are frequent items
2. {i, j} hash bucket entry is 1 (frequent bucket)



PCY in Action

Say we had a veggie shop that sells apples, bananas, carrot, avacado, blueberries, celery

Table

	Apple	Avacado	Bananas	Blueberry	Carrot	Celery
1	X		X	X		
2		X	X			
3		X			X	
4					X	X
5			X	X	X	
6	X	X	X	X	X	X

Define Hash(a, b) -> "a[0]" + "a[1]" -> h("apple","banana") = "ab"

PCY Pass 1

Let support = 4

Apple	Avacado	Bananas	Blueberry	Carrot	Celery
2	3	4	3	4	2

= F items

Total Table

	AA	AB	AC	BB	BC	CC
written for Clarity 1	0	2	0	1	0	0
2	0	1	0	0	0	0
3	0	0	1	0	0	0
4	0	0	0	0	0	1
5	0	0	0	1	2	0
6	1	4	4	1	4	1
Total	1	7	5	3	6	2

← Hash table

Frequent Bkts

AB, AC, BC

↓

Exclude AA, BB, CC

PCY Pass 2

Convert Hash Table Buckets into Bitset

AA	AB	AC	BB	BC	CC
0	1	1	0	1	0

Find Candidate Pairs!

From Frequent Value List (Bananas, Carrot)

Generate Pairs ((Bananas, Carrot)) and Hash (BC)

Check bitset [BC] = 1 → Yes

Pruning complete.

Try Candidates (BC) → only 2 cases, not 4 ∴

Moral of the story:

1. PCY Narrows search space by a lot!
2. Candidate Pairs must be made up of 2 cand. values (rule 1)
3.