1/2019

Efficient algorithm : Given T, generate all possible frequent itemsets.

Apriori - too many repetitions - bottom up approach.
fp growth - 2 scans.

Last step of Apriori is $C_{n+1} = \phi$.

- Upward closure - if an itemset is infrequent then it's & immediate supersets are infrequent.
- Downward closure - exact opp of upward closure.

No of scans in Apriori : $|C_2| + |C_3| + \cdots + |C_n|$
Time complexity of apriori.

- How is fp-growth better?
  2 d scan is to generate fp-tree.
- Sequence pattern mining in Apriori - Generalized Sequence pattern mined.

Apriori is adjustable in SPM domain.
fp growth is not suitable for GSP.

fpgrowth was developed by Han + Team (2010)
fpgrowth is not for sequence pattern mining.

- fpgrowth for sequence pattern - PrefixSpan.
  SPM used in web patterns. Generic to specific websites.

← fpgrowth, does it require only 2 overall scans?
Tree is referred as the projection of the database.
Projection is to support efficient counting.

Apriori given by Rakesh Agarwal.

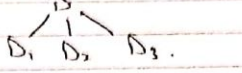
DIC was about parallel counting.

Where do more countings occur in fp tree?
I_1, I_2 might occur in many paths. So the subset merging requires smaller scans (not the entire DB). So it suffers from repeated scans.
Subset merging : looking at sequences where I_1 is present.

Both Apriori & fp-growth are time complex.
DIC is space complex.
fpgrowth is a good balance b/n Apriori and DIC.

Prefix Search (Maximal) ; A-close. (Closed FI)

- Apriori is good for sparse dataset. That means when there are many infrequent itemsets.
- Can we try improving the efficiently of Apriori?
  Apriori is a balance b/n space & time complexity.
- Partitioning approach of Apriori:



if |D| = 15 (minsup = 3)
|D1| = 5 = |D2| = |D3|.

local mining approach is done (local-Apriori).
To arrive at the global count we need merging of D_1, D_2 & D_3. (Union all three).

4/1/2019

In partitioning, something that is locally infrequent might be globally frequent.
Do union.

Hash pruning approach → in early levels for counting. Not in the C_2 level.

→ Why not in higher levels?
Occurance of the itemset with higher length will come down.

1,2,5
2,4
2,3
1,2,4
1,3
2,3
1,3
1,2,3,5
1,2,3

Minsup = 2.

if a b c d are there, then

element	
a	1
b	2
c	3
d	4

→ order

$L_1 = \{1,2,3,4,5\}$

$C_2 = L_1 \times L_2$

C_2 / L_2 freq, C_2 infreq

instead of going thru the whole dataset, we look at the hash table. In this way Apriori with hashing is better.

$H(x,y) = (\text{order of } x \times 10 + \text{order of } y) \% 7$

Buckets →	0	1	2	3	4	5	6
Contents →	(1,4) (3,5)	(1,5) (2,3) (2,3) (2,3)	(2,3) (2,4)	(2,4) (2,5)	(1,2) (1,3) (1,2) (1,3) (1,2) (1,3)		
	2	2	4	2	2	4	4

for (1,4) $H(1,4) = 10 \times 1 + 4 = 14\%7 = 0$

Repeat for all. Collision happens only for infrequent.
C_2 is better.
In case of collision, revise the hash function

- Transaction Reduction:-

If L_2 (1,2) is infrequent, then remove the transaction from the Database.

1,3,7	MS=3	1 (5)
2,3,7		2 (7)
1,2,3		3 (7)
2,3		4 (3)
2,3,4,5		
2,3		$L_1 \neq C_1$
1,2,3,4,6		
2,3,6,6		
1,3		
1		

If an itemset is infrequent in the earlier levels remove it from the later levels.
5, 7, 6 are infrequent. Remove them from the Dataset.
So now dB is,

1,3	
2,3	
1,2,3	Atleast one character is being thrown out.
2,3	
2,3,4	(Time complexity)
2,3	The order might remain same.
1,2,3,4	
2,3,4	Apriori's complexity is exponential
1,3	
1	

11/1/2017

Transaction reduction is also level based.
Transaction reduction is not efficient as an algo, but implementation wise it is efficient.

ECLAT → fi's
AClose → cfi
Dinus → mfi

* ECLAT → Equivalence Class Lattice Traversal

Apriori is BF traversal.
ECLAT → vertical representation of DB → focus on item
Horizontal focuses on Transactions.

$I_1 \to \{T_1, T_4, T_5, T_7, T_8, T_9\}$
$I_2 \to \{T_1, T_2, T_3, T_4, T_6, T_8, T_9\}$ (MS = 2)
$I_3 \to \{T_2, T_5, T_6, T_7, T_8, T_9\}$
$I_4 \to \{T_2, T_4\}$.
$I_5 \to \{T_1, T_8\}$

'∩' in order (intersection in order)

✓ $I_1 I_2 = \{T_1, T_4, T_8, T_9\}$.
✓ $I_1 I_3 = \{T_5, T_7, T_8, T_9\}$.
/ $I_1 I_4 = \{T_4\}$.
✓ $I_1 I_5 = \{T_1, T_8\}$
✓ $I_2 I_3 = \{T_2, T_6, T_8, T_9\}$.
✓ $I_2 I_4 = \{T_2, T_4\}$.
✓ $I_2 I_5 = \{T_1, T_8\}$.
/ $I_3 I_4 = 0$.
/ $I_3 I_5 = \{T_8\}$.
/ $I_4 I_5 = 0$

→ C_2 in terms of Apriori.

ECLAT is better than Apriori and fp growth on counting.
No need to go to the dataset at all.

→ $I_1 I_2 I_3$

$I_1 I_2 I_3 = \{T_8, T_9\}$
$I_1 I_2 I_5 = \{T_1, T_8\}$

(Last (maybe) ... follow up ...)

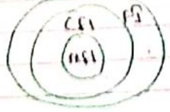
CHARM, MAFIA | Read about them

- What was the need for CFI, MFI and what?
Rare is not infrequent entirely. Sensitive & high utility itemsets are other types of itemsets.
Hiding sensitive itemset → Privacy preservation data mining.

14/1/2017

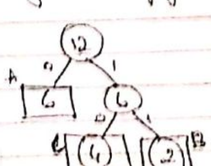
A-close → Apriori-close.

- why do we need CFI, MFI & LFI?



The application is not happy with so many FI's. So shrink it down.

Closed is from closure. The result should belong to the universe of Discourse. Here FI is the universe of discourse. CFI is lossless, MFI is lossy.
Lossless → from CFI can we reconstruct FI? Yes.
gzip, pkzip, RLE (Run Length Encoding)
Huffman trace was taught.
~~AAAAAAB~~ ABAAAABCCCC.
A → 6 B → 2 C → 4.
descending order of frequencies 6, 4, 2 (12)



D → text

for A → 0 ; B → 11 ; C → 10 → code table.
The letter with more frequency gets the ~~more~~ lesser code length.
Biology + Data Mining = Bioinformatics.

Decoding should be unambiguous. (Prefix free encoding)

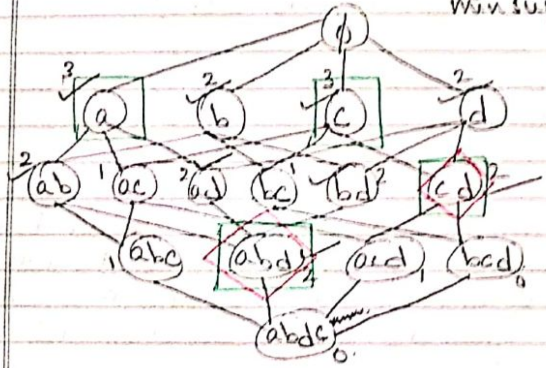
CFI, MFI find applications in compression.

Video and image compression can be lossy.

CFS = A(S) AC(S).

1/2019

min support = 2.



✓ → fi's. (9)

□ → cfi's (4)

◇ → mfi (2)

Cfi is lossless, mfi is lossy.

are we getting the entire fi from cfi's.

MFI is more about |FI| (cardinality)

CFI is a tighter constraint on MFI.

MFI also comments on the subsets.

We will not be able to say the subset count from MFI. Confidence calculation won't be possible.

With CFI we can tell the support count of subsets.

x → If there is no superset of x with the same support count as x (or less support) → CFI.

Superset (x) < SC(x)

x → ∄ any superset of x which is frequent → MFI.

- Difference between closed itemset and closed frequent itemsets.

* A-"Close"-Algorithm :-

5 items (A, C, T, D, W).

→ all the transactions that contain

t(x) itemset x. ∃ x ⊆ I.

i(y). y ⊆ T

1	A C T W
2	C D W
3	A C T W
4	A C D W
5	A C D T W
6	C D T

t(C) = {1, 2, 3, 4, 5, 6}.

i(C) = undefined.

i(T₁, T₂) = intersection of 1 & 2 = C, W.

i(t(C)) = C

t(C, D) = {2, 4, 5, 6}.

i(t(C, D)) :

t(A) = {1, 3, 4, 5}.

i(t(A)) = {A, C, W}

i(t(ACW)) = {A, C, W}.

} i(t(A)) ≠ A unlike the examples before this. it is equal to a greater set

So, i(t(A, C, W)) = {A, C, W}.

So t(x) then i(t(x)). If i(t(x)) ≠ x but is equal to y, So i(t(y)). Here it has to stop.

1 level we have elements gaining.

(ii) $i(t(x)) = x$.

(iii) $i(t(x)) = y$; $x \subset y$ → elements gained
$i(t(y)) = y$

- **Generators :-**

Identify those elements that act as generators.

x is gen(y) if $x^+ = y$.
(generator of y ; closure of x yields y)

In the example above, A is a generator of ACW

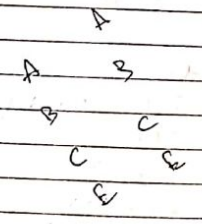
In CFP we are interested about minimal generators
There is no proper subset of x generates y.

What all generates ACW.
ACW and A.
We cannot have more than 1 minimal generator.

1/2019

A-Close Trace.

1	A C D	
2	B E F	ms = 3
3	A B C E	
4	B E	
5	A B C E	



$C_1 = \{A\ B\ C\ D\ E\}$.
$L_1 = \{A, B, C, E\}$.
$C_2 = \{\underset{2}{AB}, \underset{3}{AC}, \underset{2}{AE}, \underset{3}{BC}, \underset{4}{BE}, \underset{3}{CE}\}$

With CFI we reconstruct the FI's but with Closed Itemset we can traceback to the transactions.

Communique

$L_2 = \{\underset{3}{AC}, \underset{3}{BC}, \underset{4}{BE}, \underset{3}{CE}\}$.
Is any item having the same support as it's subset. Here $AC \subseteq A$ and $BE \subseteq B$. Throw them away.
$L_2 = \{\underset{3}{BC}, \underset{3}{CE}\}$

$C_3 = \phi$

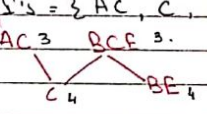
Since we left out AC & BE we must prove that AC can be reached from some other minimal set.

(x) Generators	t(x)	i(t(x))
A	1, 3, 5	AC
B	2, 3, 4, 5	BE
C	1, 2, 3, 5	C
E	2, 3, 4, 5	× BE
BC	2, 3, 5	BCE
CF	2, 3, 5	× BCE

We are able to reach them here

Whatever is left out is able to be reached from $i(t(x))$. Reaching out in mathematical terms is closure (x^+) $x^+ = y$

Remove the duplicates in order.
CFI's = {AC, C, BCE, BE}.



→ with this as a start pt we should be able to trace back all the FI's.

By not retaining the higher cardinality sets, we are saving space. The cardinality of the Generators should be minimum.
One CFI alone cannot give all the FI's. Go for the CFI with the larger count.

bottom up.

* Pincer Search. ↑↓ "Dataset is given"

support = 20%, relative support = 15
∴ min sup = 3

{1, 2, 3, 4, 5, 6, 7, 8, 9} 9 symbols.
Move in both bottom up and topdown.
Maximal frequent set ← MFS = ∅
Maximal frequent candidate set. MFCS = {1 2 3 4 5 6 7 8 9} (Maximal frequent candidate set).

S_i → infrequent at i^{th} level.

$L_1 = \{1, 2, 3, 4, 5, 6, 7, 8\}$.
$S_1 = \{1, 9\}$
We keep record of infrequent to see that they are not part of MFS.
Keep resizing MFCS.
Trimming down MFCS is top down approach.
MFCS and S_i are helping to bring in the top down flavour to the trace.

Updating MFCS:
$S_1 = \{1, 9\}$ MFCS = {1 2 3 4 5 6 7 8 9}
for {1} in S_1 :- {2, 3, 4, 5, 6, 7, 8, 9} = MFCS
MFCS \ S_1 → minus in set theory.
for {9} in S_1 :- {2, 3, 4, 5, 9, 7, 8} = MFCS.
MFCS \ S_1.

$C_2 = L_1 \bowtie L_1$

$L_2 = \{23, 24, 35, 37, 56, 57, 67\}$

1/1/2019

$S_2 = \{25, 26, 27, 28, 34, 36, 38, 45, 46, 47, 48, 58, 68, 78\}$
Exhaust S_i

smt18@iiitdm.ac.in

Communique

25 are infrequent together. '26' is occurring together only here.

(25) S_2 → MFCS = {345678, 234678}

trim down only if they are occurring together

(26) S_2 → MFCS = {345678, 34678, 23478} 34678 is already in

So do not consider '34678'

∴ updated MFCS = {23478, 345678}

(27) S_2 → MFCS = {3478, 2348, 345678}. (3478 is a subset of 345678)

MFCS = {2348, 345678}

(28) S_2 → MFCS = {348, 234, 345678} (348 subset of 2348)

MFCS = {234, 345678}

(34) S_2 → MFCS = {23, 24, 45678, 35678}.

MFS is a lossy reconstruction of FS. We do not have the sup count. So, we cannot have the confidence.

continue for the other S_2's

(23) (S_2) → MFCS = {23, 24, 357, 5678, 8}

C_3 = {234, 357, 567}

Topdown will stop if S_i is nullset. There is no more trimming down of MFCS.

Here S_3 is a null set.

LAB-2.

Implement A close and Pincer Search.
Two more algos. MAFIA & CHARM.

28/1/2019

- Difference b/n BigData and DataMining.
~~How well my~~ how well the storage systems can handle the distributed data. Incremental learning (with added data, the system must be intelligent enough to relearn the patterns). Storage issues according to scaling up of data must be addressed. Scaling up is done across networks.

Online DB ≡ Incremental learning.

Bucket Brigade → Genetic Algorithms
↓
evolutionary computation.
↓
evolve for the good.

GA is one evolutionary computation strategy.
Tabu search, PSO, Ant colony are others.
↓
particle swarm optimization.

Reinforcement learning thrives on penality & awards.
Most evolutionary computation algos work on Initial population.
All the evolutionary computation methods are basically for optimization (on the computers front).

Bother for optimum after seeing if the solution is feasible or not.

David Goldberg → GA book (1989)
James Holland → inventor of GA.

- Enumerative search → point by point search. Also called try and error. (but it is random)
- Operations Research → mathematical optimization methods.
- Derivative based → Gradient ascent or descent. Hill climbing is also another word. A slope should always exist. But real life data is not always continuous. Also, the data can be multimodal, the hill climbing gets stuck in local optimum.

In the search for optimal value, the path towards optima is also important.

Where will GA fit in? → Classification.
If we are not able to give unique class label in Dtree, it is non feasible.

Convergence → how quickly will I reach the leaf no

Efficiency vs efficacy.

28/1/2019

- Why EVOLUTIONARY?
Can my overall objective keep bettering. GA, Tabu Search etc are search algos that search for optimal.

Fetness of a solution:- how good is a solution

iteration = Generation.
As we do more generations, the fetness should improve.