MACHINE INTELLIGENCE

Dr. N MEHALA
Department of Computer
Science and Engineering

MACHINE INTELLIGENCE

Module 4 [Unsupervised Learning]

Dr. N MEHALA

Department of Computer Science and Engineering

- Use a compressed representation of the database using an FP-tree
- FP-Growth: allows frequent item set discovery without candidate item set generation

Two step approach:

Step 1: Build a compact data structure called the FP-tree

- Built using 2 passes over the data-set.

Step 2: Extracts frequent item sets directly from the FP-tree

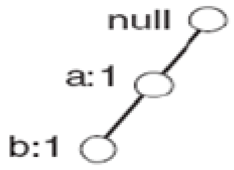
- Traversal through FP-Tree

- Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent item sets

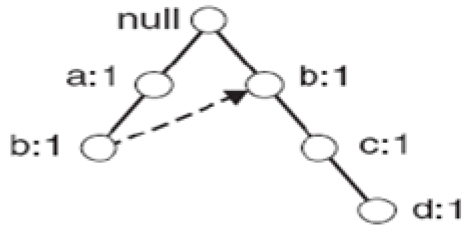
STEP1 : FP-Tree construction (Example)

Transaction Data Set

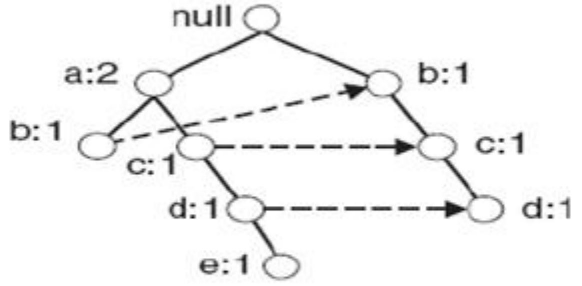
TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}



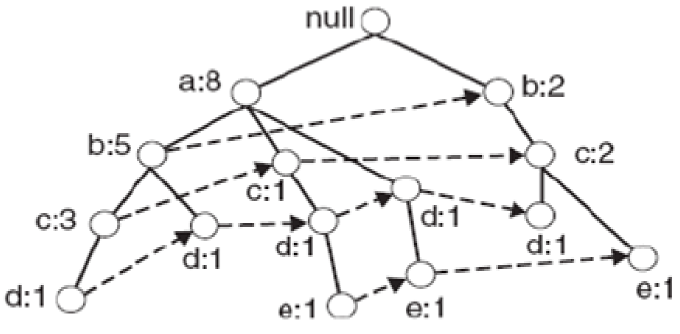
(i) After reading TID=1



(ii) After reading TID=2



(iii) After reading TID=3



(iv) After reading TID=10

- Pointers are maintained between nodes containing the same item, creating singly linked lists (dotted lines)
- The more paths that overlap, the higher the compression. FP-tree may fit in memory.
- Frequent itemsets are extracted from the FP-Tree.

FP Tree Zize

- The FP-Tree usually has a smaller size than the uncompressed data – typically many transactions share items (and hence prefixes).
  - *Best case scenario*: all transactions contain the same set of items.
    - 1 path in the FP-tree
  - *Worst case scenario*: every transaction has a unique set of items (no items in common)
    - Size of the FP-tree is *at least* as large as the original data.
    - Storage requirements for the FP-tree are higher – need to store the pointers between the nodes and the counters.
  - The size of the FP-tree depends on how the items are ordered
    - Ordering by decreasing support is typically used but it does not always lead to the smallest tree (it's a heuristic).

MACHINE INTELLIGENCE

Dr. N MEHALA
Department of Computer
Science and Engineering

MACHINE INTELLIGENCE

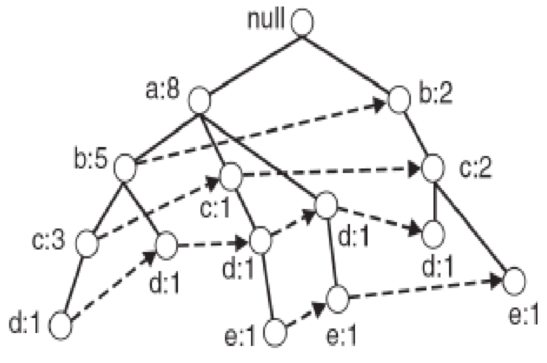
Module 4 [Unsupervised Learning]

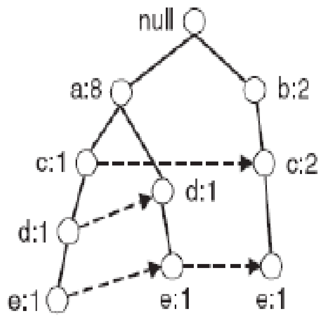
Dr. N MEHALA
Department of Computer Science and Engineering

STEP2: Frequent Item Generation

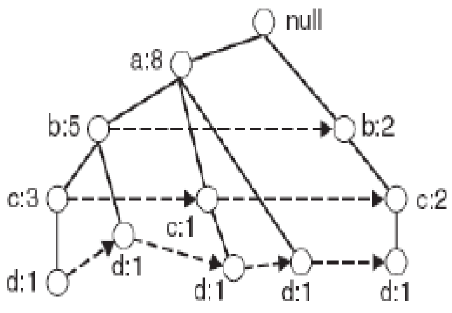
STEP2: Frequent Item Generation

- FP-Growth extracts frequent itemsets from the FP-tree.
- Bottom-up algorithm – from the leaves towards the root
  - Divide and conquer: first look for frequent itemsets ending in *e*, then *de*, etc. . . then *d*, then *cd*, etc. . .
- First, extract prefix path sub-trees ending in an item(set). (*hint*: use the linked lists)





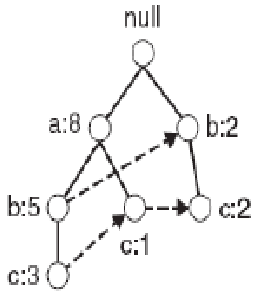
(a) Paths containing node e



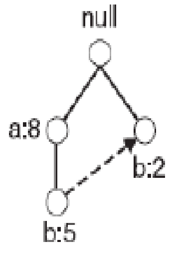
(b) Paths containing node d

↑ Complete FP-tree

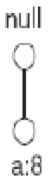
→ **Example:** prefix path sub-trees



(c) Paths containing node c



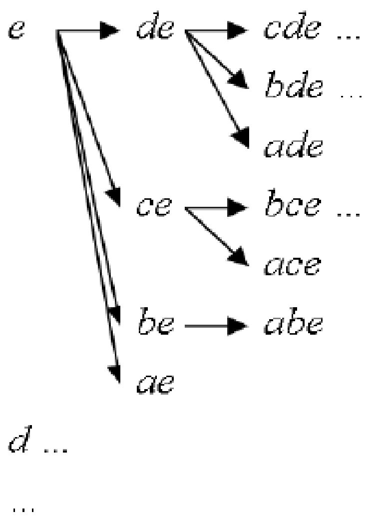
(d) Paths containing node b

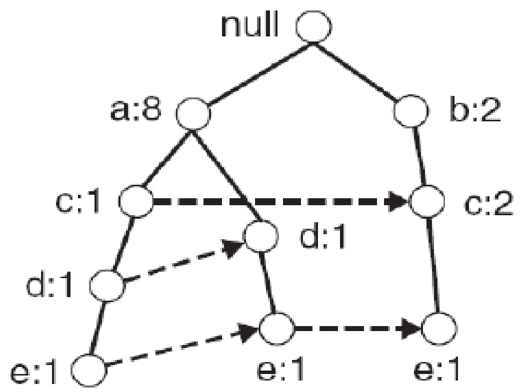


(e) Paths containing node a

STEP2: Frequent Item Generation

- Each prefix path sub-tree is processed recursively to extract the frequent itemsets. Solutions are then merged.
  - **E.g.** the *prefix path sub-tree* for e will be used to extract frequent itemsets ending in e, then in de, ce, be and ae, then in cde, bde, cde, etc.
  - Divide and conquer approach





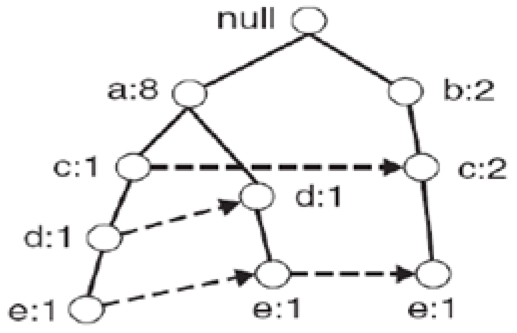
Prefix path sub-tree ending in e.

STEP2: Frequent Item Generation

Example

Let $minSup = 2$ and extract all frequent itemsets containing e.

- 1. Obtain the prefix path sub-tree for e:

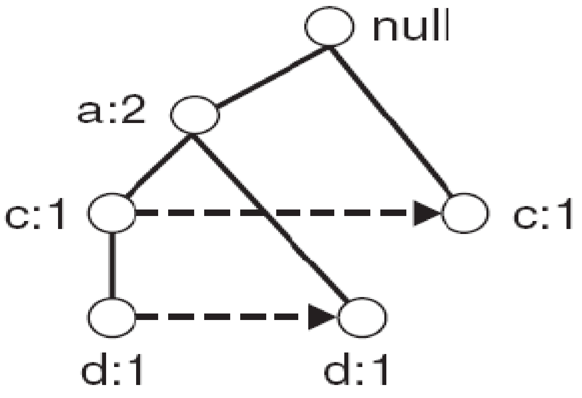


- 2. Check if e is a frequent item by adding the counts along the linked list (dotted line). If so, extract it.
  - Yes, count =3 so $\{e\}$ is extracted as a frequent itemset.
- 3. As e is frequent, find frequent itemsets ending in e. i.e. de, ce, be and ae.
  - i.e. decompose the problem recursively.
  - To do this, we must first to obtain the conditional FP-tree for e.

Conditional FP-Tree

- The FP-Tree that would be built if we only consider transactions containing a particular itemset (and then removing that itemset from all transactions).
- **Example**: FP-Tree conditional on *e*.

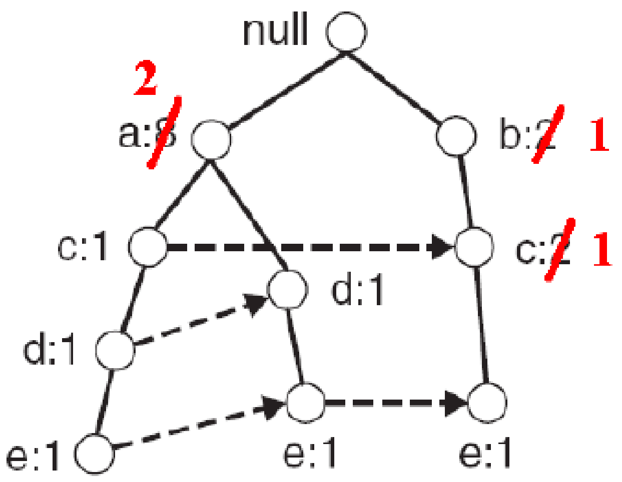
TID	Items
~~1~~	~~{a,b}~~
~~2~~	~~{b,c,d}~~
3	{a,c,d,~~e~~}
4	{a,d,~~e~~}
~~5~~	~~{a,b,c}~~
~~6~~	~~{a,b,c,d}~~
~~7~~	~~{a}~~
~~8~~	~~{a,b,c}~~
~~9~~	~~{a,b,d}~~
10	{b,c,~~e~~}



STEP2: Frequent Item Generation

To obtain the *conditional FP-tree* for e from the *prefix sub-tree* ending in e:

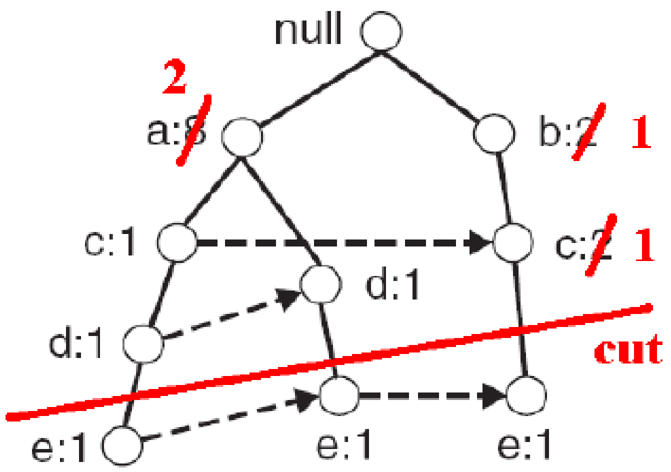
- Update the support counts along the prefix paths (from e) to reflect the number of transactions containing e.
  - b and c should be set to 1 and a to 2.

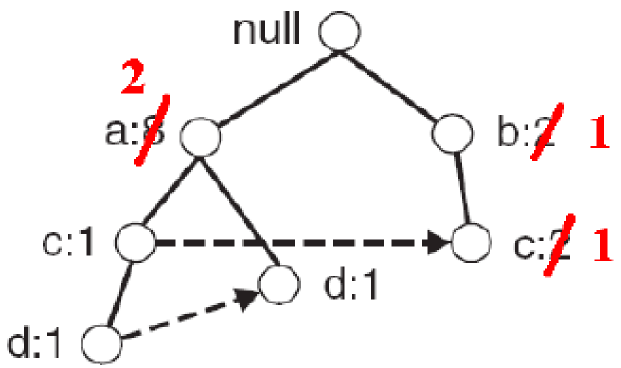


STEP2: Frequent Item Generation

To obtain the *conditional FP-tree* for *e* from the *prefix sub-tree* ending in *e*:

- Remove the nodes containing *e* – information about node *e* is no longer needed because of the previous step

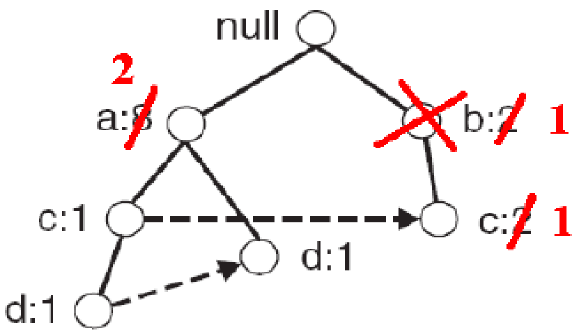


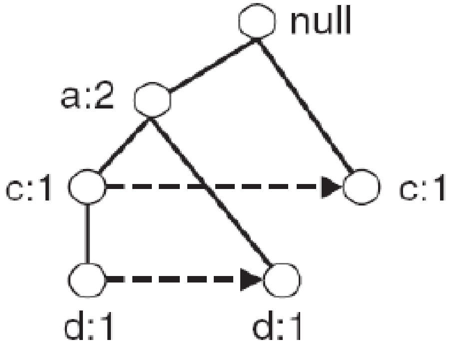


Conditional FP-Tree

To obtain the *conditional FP-tree* for *e* from the *prefix sub-tree* ending in *e*:

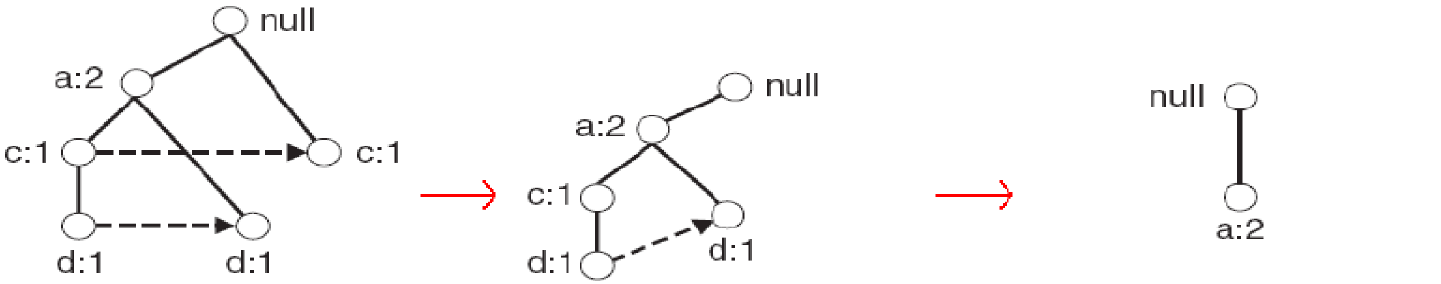
- Remove infrequent items (nodes) from the prefix paths
- **E.g.** *b* has a support of 1 (note this really means *be* has a support of 1). i.e. there is only 1 transaction containing *b and e* so *be* is infrequent – can remove b.





Example (continued)

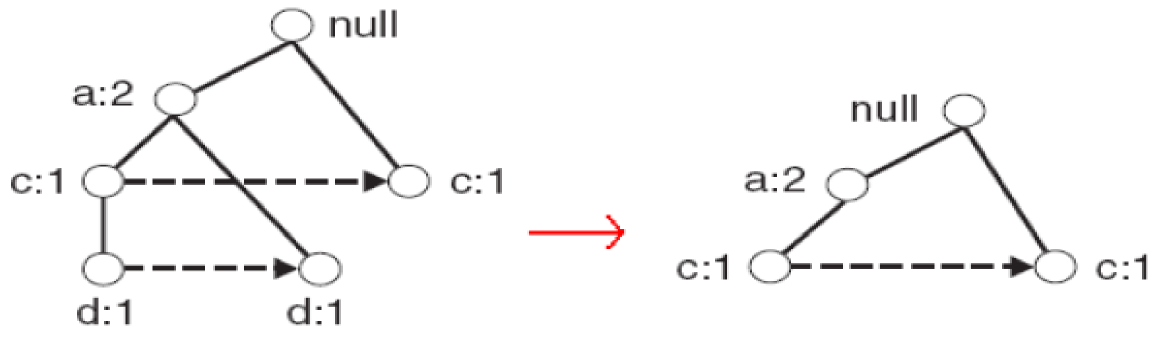
- 4. Use the the conditional FP-tree for e to find frequent itemsets ending in de, ce and ae
  - Note that be is not considered as b is not in the conditional FP-tree for e.
  - For each of them (e.g. de), find the prefix paths from the conditional tree for e, extract frequent itemsets, generate conditional FP-tree, etc... (recursive)
  - **Example:** $e \longrightarrow de \longrightarrow ade$ ($\{d, e\}$,$\{a, d, e\}$ are found to be frequent)



Conditional FP-tree for e   Prefix paths ending in de   Conditional FP-tree for de

Example (continued)

- 4. Use the the conditional FP-tree for e to find frequent itemsets ending in de, ce and ae
  - **Example:** $e \longrightarrow ce$ ($\{c, e\}$ is found to be frequent)



Conditional FP-tree for e

Prefix paths ending in ce

- etc... (ae, then do the whole thing for b,... etc)

STEP2: Frequent Item Generation

- Frequent itemsets found (ordered by suffix and order in which they are found):

Suffix	Frequent Itemsets
e	{e}, {d,e}, {a,d,e}, {c,e},{a,e}
d	{d}, {c,d}, {b,c,d}, {a,c,d}, {b,d}, {a,b,d}, {a,d}
c	{c}, {b,c}, {a,b,c}, {a,c}
b	{b}, {a,b}
a	{a}

Discussion

- Advantages of FP-Growth
  - only 2 passes over data-set
  - "compresses" data-set
  - no candidate generation
  - much faster than Apriori
- Disadvantages of FP-Growth
  - FP-Tree may not fit in memory!!
  - FP-Tree is expensive to build
    - Trade-off: takes time to build, but once it is built, frequent itemsets are read off easily.
    - Time is wasted (especially if support threshold is high), as the only pruning that can be done is on *single items*.
    - support can only be calculated once the entire data-set is added to the FP-Tree.

Summary

Resources

- http://www2.ift.ulaval.ca/~chaib/IFT-4102-7025/public_html/Fichiers/Machine_Learning_in_Action.pdf
- http://wwwusers.cs.umn.edu/~kumar/dmbook/.
- ftp://ftp.aw.com/cseng/authors/tan
- http://web.ccsu.edu/datamining/resources.html

THANK YOU

Dr. N MEHALA
Department of Computer Science and Engineering

mehala@pes.edu