PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Introduction to Machine Learning

MLEARN 510A – Lesson 10

Logistics

- Today is our last class
- Please submit your assignment (including any outstanding ones) by next Tuesday
- Please fill End of Course Survey

Recap of Lesson 9

- Introduction to Time Series
- Time Series Decomposition
- Time Series Forecasting
- ETS Models
- ARIMA Models
- Model Evaluation

Outline for Lesson 10

- Frequent Itemset Mining
  - Frequent Itemsets
  - Apriori Algorithm
  - FP Tree
  - Maximal and Closed Frequent Itemsets
  - Measures of interestingness

- Hands-on Lab

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Frequent Itemset Mining

Motivation

- What do beers have in common with diaper sales?

- Walmart supposedly found out that there are certain times at which beer and diapers sell particularly well together
  - Friday evenings
  - Why would that be the case?

Market Basket Analysis

- “*As they say, never let truth get in the way of a good story*”
- This is more of an urban legend than a true data mining discovery
- There may be some benefit in mining data to study what items are bought together i.e., what items *frequently* occur together?
- **Market-Basket Model**
  - Defines many-many relationship between “items” and “baskets”
  - The frequent-itemsets problem is that of finding sets of items that appear in (are related to) many of the same baskets

Market Basket Analysis – Continued

TID	Beer	Milk	Bread	Diapers	Coke
T1	0	1	1	0	0
T2	1	0	0	0	0
T3	1	0	0	1	0
T4	0	1	1	0	0
T5	0	0	1	0	1

- We have items and baskets, sometimes called “transactions”
- Each basket consists of a set of items (an itemset), and usually we assume that the number of items in a basket is small – much smaller than the total number of items
- The number of baskets is usually assumed to be very large

Market Basket Analysis – Definitions

- $I = \{i_1, i_2, \ldots, i_n\}$ a set of literals, called **items**
- **Transaction** T: a set of items such that $T \subseteq I$
- **Dataset** D: a set of transactions
- A **transaction** T contains X, a set of items in I if $X \subseteq I$
- An **association rule** is an implication $X \Longrightarrow Y$
- The rule $X \Longrightarrow Y$ has **support** s in transaction set D if s% of transactions in D contain $X \cup Y$
- The rule $X \Longrightarrow Y$ holds in transaction set D with **confidence** c if c% of transaction in D that contain X also contain Y

Definitions – Continued

- **Itemset:**
  - A collection of one or more items. Example: {Milk, Bread, Diaper}
  - k-itemset. An itemset containing k items
- **Support count (σ)**
  - Frequency of an itemset
    - σ(Milk, Bread, Diaper) = 2
- **Support (s)**
  - Fraction of transactions containing an itemset
  - s(Milk, Bread, Diaper) = 2/5
- **Frequent Itemset**
  - One whose support is greater than or equal to a *minsup* threshold

TID	Items
T1	Bread, milk
T2	Bread, diaper, beer, eggs
T3	Milk, diaper, beer, coke
T4	Bread, milk, diaper, beer
T5	Bread, milk, diaper, coke

Applications of Frequent Itemsets

- Market – Basket Analysis in Retail
  - A retailer can learn what is commonly bought together

- Related Concepts
  - Let items be words, and baskets be documents (e.g., Web pages). A basket/document contains those items/words that are present in the document
  - Ignoring all the most common words, then we would hope to find among the frequent pairs some pairs of words that represent a joint concept

- Biomarkers
  - Let the items be of two types – biomarkers such as genes blood proteins, and diseases. Each basket is the set of data about a patient: their genome and blood-chemistry analysis, as well as their medical history of disease
  - A frequent itemset that consists of one disease and one or more biomarkers suggests a test for the disease.

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

Association Rules

Association Rules

- The process of discovering interesting relations between variables in large datasets
- Introduced by Agrawal, Imielinski and Swami in 1993
- Before dropping out of Stanford, Sergey Brin (Google's cofounder) published several papers on Association Rules

Association Rules Example

Example:

$$\{\text{Milk}, \text{Diaper}\} \rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

TID	Items
T1	Bread, milk
T2	Bread, diaper, beer, eggs
T3	Milk, diaper, beer, coke
T4	Bread, milk, diaper, beer
T5	Bread, milk, diaper, coke

More Discussion on Association Rules

- Why use support and confidence?
  - Rules with low support may occur simply by chance
  - A low support rule is not interesting from a business perspective
  - Confidence measures the reliability of the inference made by a rule.
    - For $X \Rightarrow Y$, the higher the confidence, the more likely it is for Y to be present in transactions containing X

- Association rules results should be interpreted with caution
  - They do not imply causality, which requires extra knowledge of your data
  - Instead, they simply imply a strong co-occurrence relationship between items

Interest of an Association Rule

- Confidence alone can be useful, provided the support for the left side of the rule is large

- Often more value to an association rule if it reflects a true relationship

- **Interest** of an association rule $X \Rightarrow Y$: the difference between its confidence and the fraction of baskets that contain Y
  - That is, if X has no influence on Y, then we would expect that the fraction of baskets including X that contain Y would be exactly the same as the fraction of all baskets that contain Y
  - Such a rule has interest 0

Quiz

- We have seen an example of an association rule with positive interest {Beer} => Diapers. That is, the fraction of diaper buyers who buy beer is significantly greater than the fraction of all customers that buy beer.

- Can you think of an association rule with negative interest?

Strong Association Rules

- We are often interested in finding strong association rules
  - *sup R* ≥ *minsup* and *conf R* ≥ *minconf*

Itemset $I = \{i_1, i_2, \ldots, i_n\}$

Find all rules $X \Rightarrow Y$ such that:

– $min\_support\ =\ 50\%$

– $min\_conf\ =\ 50\%$

TID	Items
T1	A, B, C
T2	A, C
T3	A, D
T4	B, E, F

$\boldsymbol{A \Rightarrow D}$ $(25\%, 33.3\%)$ ✗

$\boldsymbol{A \Rightarrow B}$ $(25\%, 33.3\%)$ ✗

$\boldsymbol{A \Rightarrow C}$ $(50\%, 66.7\%)$ ✓

$\boldsymbol{C \Rightarrow A}$ $(50\%, 100\%)$ ✓

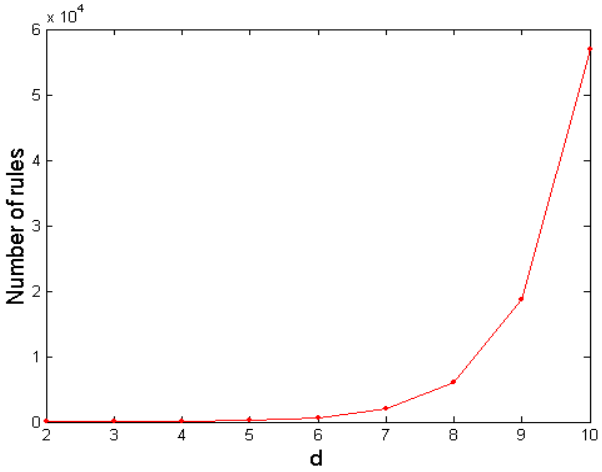
The Association Rule Mining Problem

- Given a set of transactions T, find all the rules having support ≥ *minsup* and confidence c, where *minsup* and *minconf* are the support and confidence thresholds, respectively

The Association Rule Mining Problem

- **Brute-force Approach**
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

- Computationally prohibitive!



- Given *d* items in *I:*
  - Total number of possible itemsets = 2^d
  - Total number of association rules

$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right]$$
$$= 3^d - 2^{d+1} + 1$$

Mining Association Rules

- Two step approach

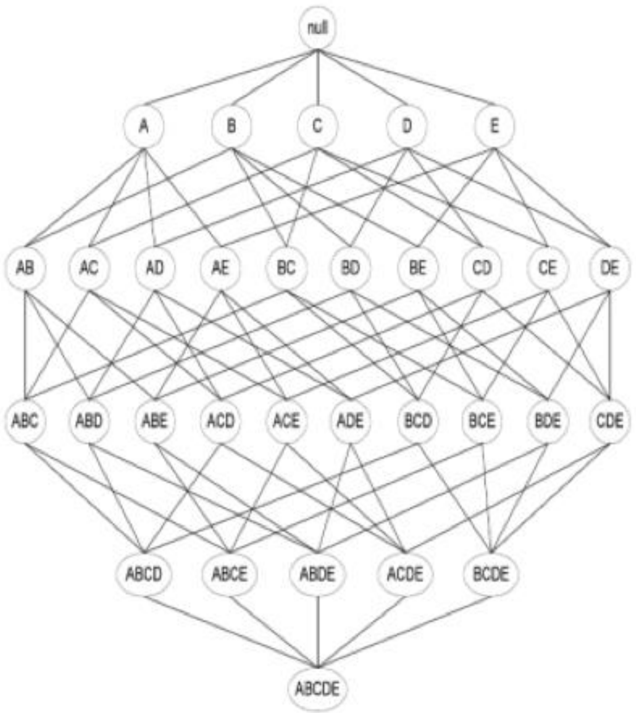
- **Step 1: Frequent Itemset Generation**
  - Generate all itemsets whose support ≥ *minsup*

- **Step 2: Rule Generation**
  - Generate high confidence rules from each frequent itemset

Frequent Itemset Generation

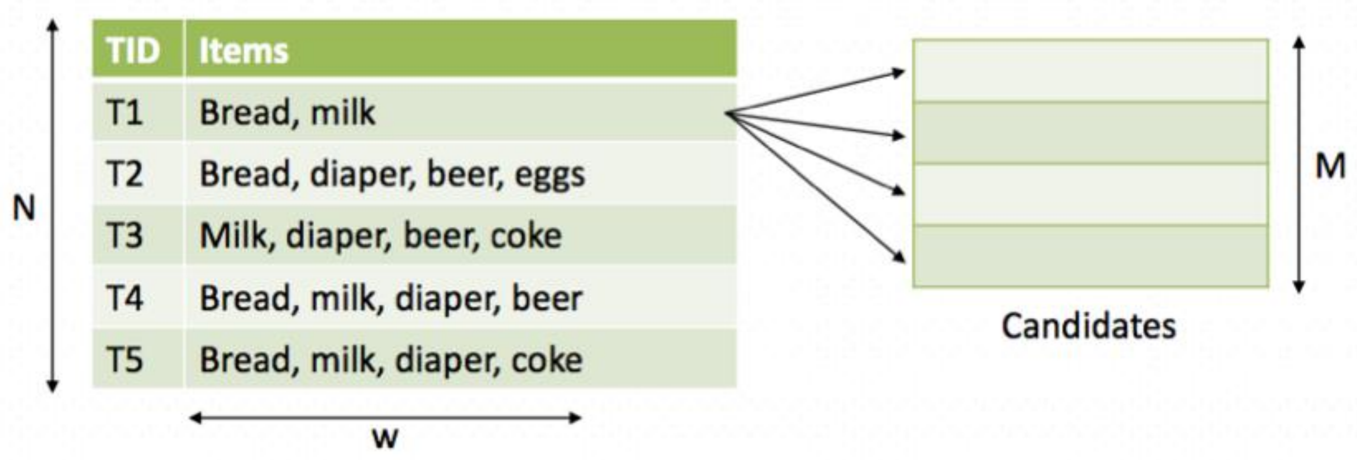
- For a set with *n* items, we have $2^n - 1$ possible itemsets

- Each of these is called a **candidate frequent itemset**



Itemset lattice

Frequent Itemset Generation



TID	Items
T1	Bread, milk
T2	Bread, diaper, beer, eggs
T3	Milk, diaper, beer, coke
T4	Bread, milk, diaper, beer
T5	Bread, milk, diaper, coke

- ➢ Compare each candidate against every transaction
- ➢ Complexity: *O(NMw)*

Frequent Itemset Generation

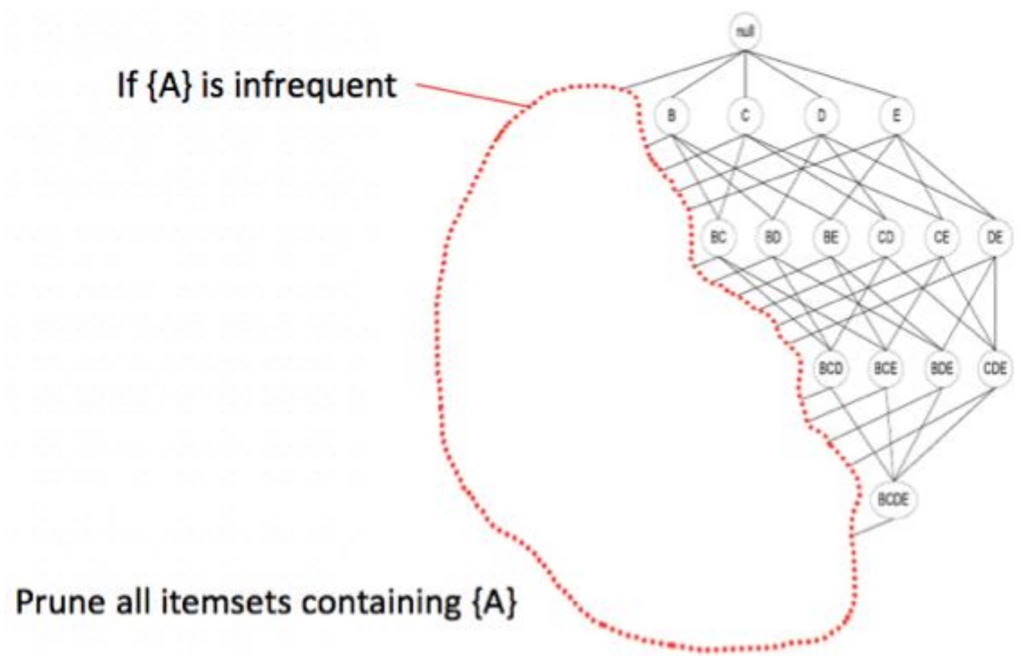
- Several ways to reduce the computational complexity:
  - Reduce the number of candidate itemsets (M) by pruning the itemset lattice
    - e.g., Apriori Algorithm
  - Reduce the number of comparisons by using advanced data structures to store the candidate itemsets or to compress the dataset
    - e.g., FP Growth

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Apriori Algorithm

Apriori Algorithm – Intuition

- **Intuition:** Apriori Algorithm reduces the number of candidate itemsets (M) by pruning the itemset lattice



Apriori Algorithm – Basic Principle

- **Apriori Principle**: If an itemset is frequent, then all of its subsets must also be frequent

- This principle holds true because of the following property of support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

- The support of an itemset never exceeds that of its subsets

Apriori Algorithm – Details

- If we identify an itemset as being infrequent, its subsets should not be generated/tested
  - If {Beer, Diaper, Oreos} is frequent, so is {Beer, Diaper}, since every transaction having the first also contains the latter

- Algorithm:
  - Iteratively generate candidates with length (k + 1) from the frequent itemsets with length k
  - Test candidates against dataset of transactions
  - Update k and repeat the process

Apriori Algorithm Illustrated

D

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	BE

$minsup_{count} = 2$

C_1

Itemset	Support
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

F_1

Itemset	Support
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A,B}
{A,C}
{A,E}
{B,C}
{B,E}
{C,E}

C_2

Itemset	Support
{A,B}	1
{A,C}	2
{A,E}	1
{B,C}	2
{B,E}	3
{C,E}	2

F_2

Itemset	Support
{A,C}	2
{B,C}	2
{B,E}	3
{C,E}	2

Interesting part

Apriori Algorithm Illustrated

D

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	BE

$minsup_{count} = 2$

F_1

Itemset	Support
{A}	2
{B}	3
{C}	3
{E}	3

F_2

Itemset	Support
{A,C}	2
{B,C}	2
{B,E}	3
{C,E}	2

C_3

Itemset	Support
{A,B,C}	1
{A,C,E}	1
{B,C,E}	2
{A,B,E}	1

F_3

Itemset	Support
{B,C,E}	2

C_4

Itemset	Support
{A,B,C,E}	1

$F_4 = \emptyset$

Done !

Recall the Apriori principle: *All subsets of a frequent subset must also be frequent*

Apriori Algorithm Illustrated

D

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	BE

$minsup_{count} = 2$

F_1

Itemset	Support
{A}	2
{B}	3
{C}	3
{E}	3

F_2

Itemset	Support
{A,C}	2
{B,C}	2
{B,E}	3
{C,E}	2

F_3

Itemset	Support
{B,C,E}	2

- We have just illustrated the $F_{k-1} \times F_1$ method
  - We combined frequent itemsets of size k − 1 with all frequent itemsets of size 1 to generate F_k

- But we can do better! → $F_{k-1} \times F_{k-1}$ method

F_{k-1} × F_{k-1} method

D

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	BE

$minsup_{count} = 2$

F_2

Itemset	Support
{A,C}	2
{B,C}	2
{B,E}	3
{C,E}	2

F_2

Itemset	Support
{A,C}	2
{B,C}	2
{B,E}	3
{C,E}	2

C_3

Itemset	Support
{B,C,E}	2

F_3

Itemset	Support
{B,C,E}	2

Done !

Merge pairs of frequent $(k-1)$-itemsets only if their first $k-2$ items are identical

Quiz

D

TID	Items
1	A, B,C
2	B,D
3	B,C
4	A,B,D
5	A,C
6	B,C
7	A,C
8	A,B,C,E
9	A,B,E

$minsup_{count} = 2$

F_1 ?

F_2 ?

F_3 ?

Generating Association Rules

- For each frequent itemset f, generate all nonempty subsets of f
- For every non-empty subsets s of f, output the rule $s \Rightarrow (f - s)$, if

$$\frac{support_{count}(f)}{support_{count}(s)} \geq minconf$$

- Since these rules are generated from frequent itemsets, each automatically satisfies the minimum support.

Generating Association Rules

Suppose $F_3 = \{A, B, E\}$. What association rules can be generated?

Subsets: $\{A,B\}, \{A,E\}, \{B,E\}, \{A\}, \{B\}, \{E\}$

$$\{A,B\} \Rightarrow \{E\}, conf = 2/4 = 0.5$$
$$\{A,E\} \Rightarrow \{B\}, conf = 2/2 = 1.0$$
$$\{B,E\} \Rightarrow \{A\}, conf = 2/2 = 1.0$$
$$\{A\} \Rightarrow \{B,E\}, conf = 2/6 = 0.33$$
$$\{B\} \Rightarrow \{A,E\}, conf = 2/7 = 0.29$$
$$\{E\} \Rightarrow \{A,B\}, conf = 2/2 = 1.0$$

TID	Items
1	A, B,C
2	B,D
3	B,C
4	A,B,D
5	A,C
6	B,C
7	A,C
8	A,B,C,E
9	A,B,E

Itemset	Support
{A}	6
{B}	7
{C}	6
{D}	2
{E}	2

Itemset	Support
{A,B}	4
{A,C}	4
{A,E}	2
{B,C}	4
{B,D}	2
{B,E}	2

Itemset	Support
{A,B,C}	2
{A,B,E}	2

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Frequent Pattern (FP) Growth

Motivation

- Using Apriori needs a generation of candidate itemsets. These itemsets may be large in number if the itemset in the database is huge

- Apriori needs multiple scans of the database to check the support of each itemset generated and this leads to high costs

- Can we do something better?

Frequent Pattern (FP) Growth

- This algorithm is an improvement to the Apriori method
- A frequent pattern is generated without the need for candidate generation
- FP growth algorithm represents the database in the form of a tree called a frequent pattern tree or FP tree. This tree structure will maintain the association between the itemsets

FP Tree Representation

- An FP-tree is a compressed representation of the input
- It is constructed by reading the dataset one transaction at a time and mapping each transaction onto a path in the FP-tree
- The more the paths overlap with one another, the greater the compression that can be achieved
- The root node represents null while the lower nodes represent the itemsets. The association of the nodes with the lower nodes, that is the itemsets with the other itemsets, are maintained while forming the tree

Constructing an FP Tree

- Consider the dataset to the left
  - Ten total transactions or itemsets
  - Five total items (a, b, c, d, 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}

Constructing an FP Tree

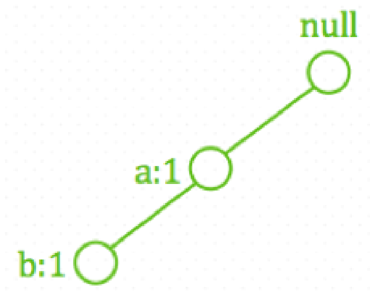
- **Step 1**: The dataset is scanned once to determine the support count of each item. Infrequent items are discarded, while the frequent items are sorted in decreasing support counts. For this dataset, a is the most frequent item, followed by b, c, d, and e

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}

Constructing an FP Tree

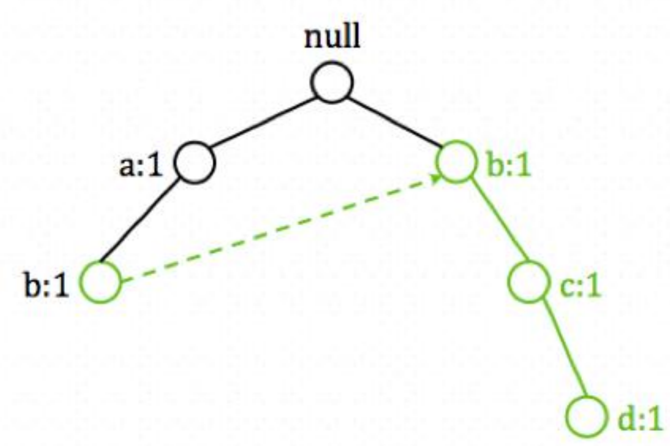
- **Step 2**: A second pass over the dataset is used to construct the FP-tree. After reading the first transaction, a, b , the nodes a and b are created. A path is then formed from null → *a* → *b* to encode the transaction. Every node along the path has a frequency count of 1

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}



Constructing an FP Tree

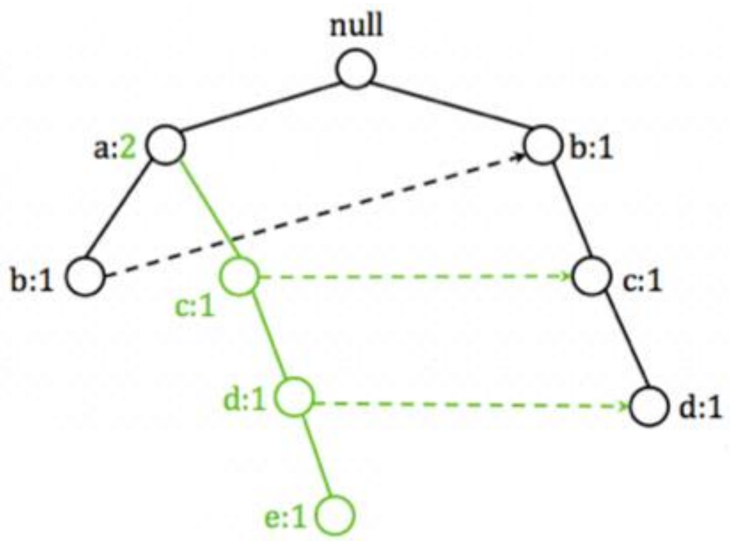
- **Step 3**: After reading the second transaction, b, c, d , a new set of nodes is created for items b, c, and d. A path is formed to represent the transaction by connecting nodes null → *b* → *c* → *d*.



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}

Constructing an FP Tree

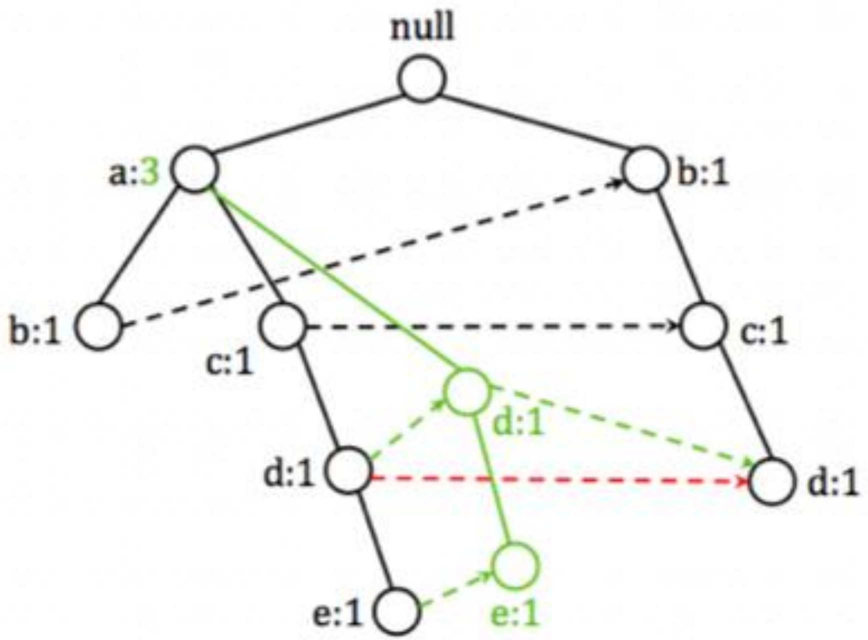
- **Step 4**: The third transaction, a, c, d, e shares a common prefix item with the first transaction (a), thus overlapping. Due to this, the frequency count for node *a* is incremented to 2



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}

Constructing an FP Tree

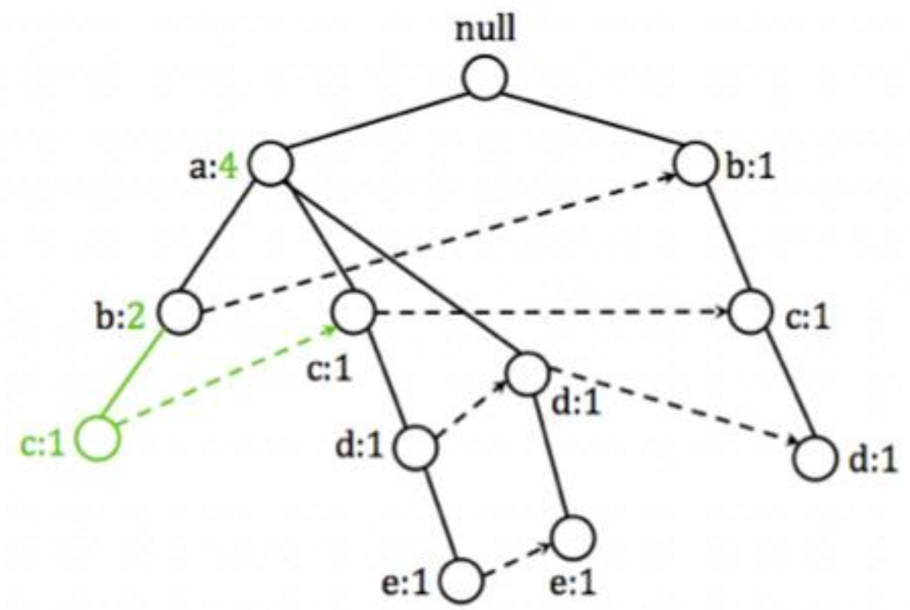
- Step 5:



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}

Constructing an FP Tree

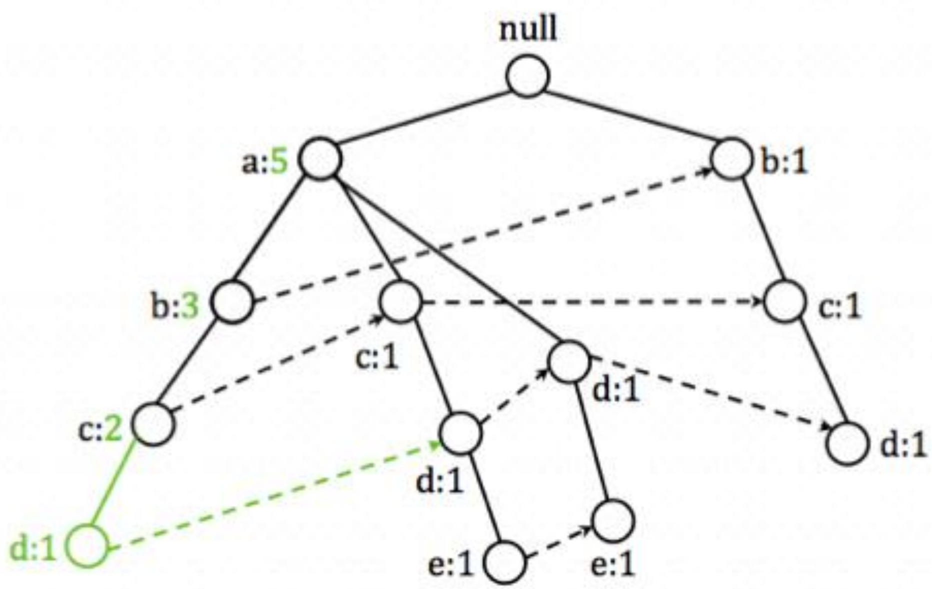
- Step 6:



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}

Constructing an FP Tree

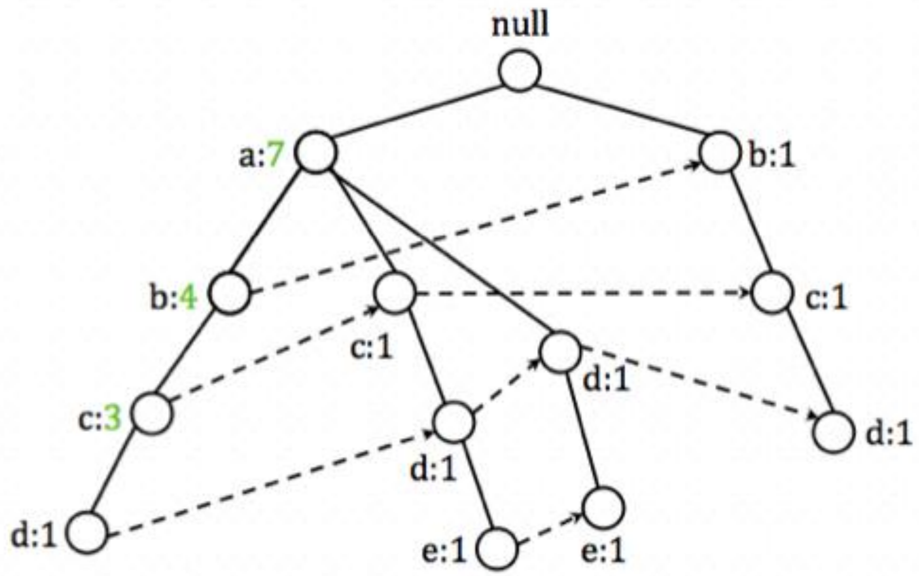
- Step 7:



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}

Constructing an FP Tree

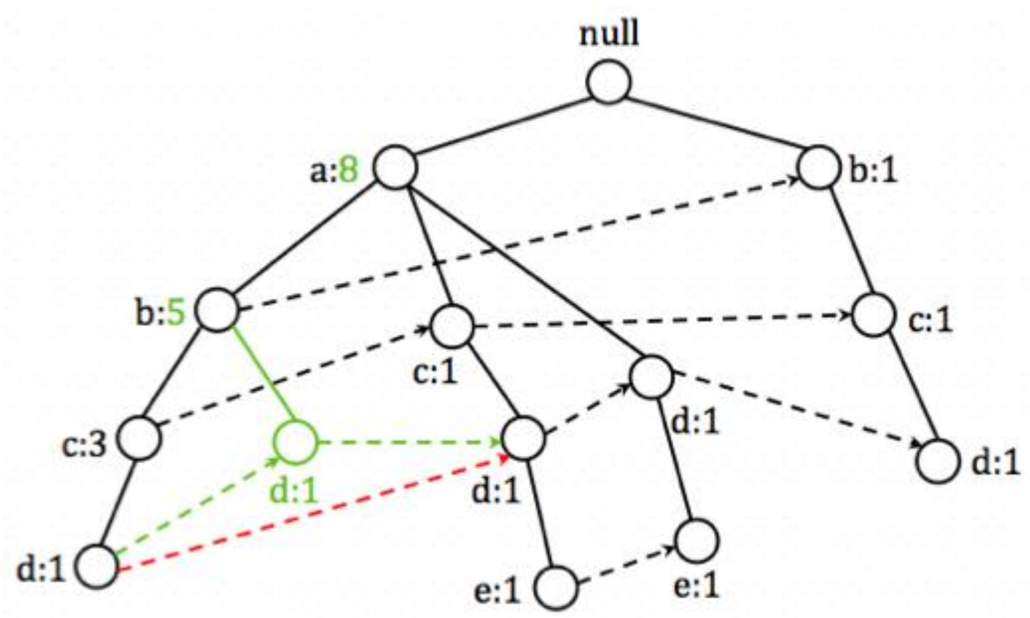
- Step 9:



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}

Constructing an FP Tree

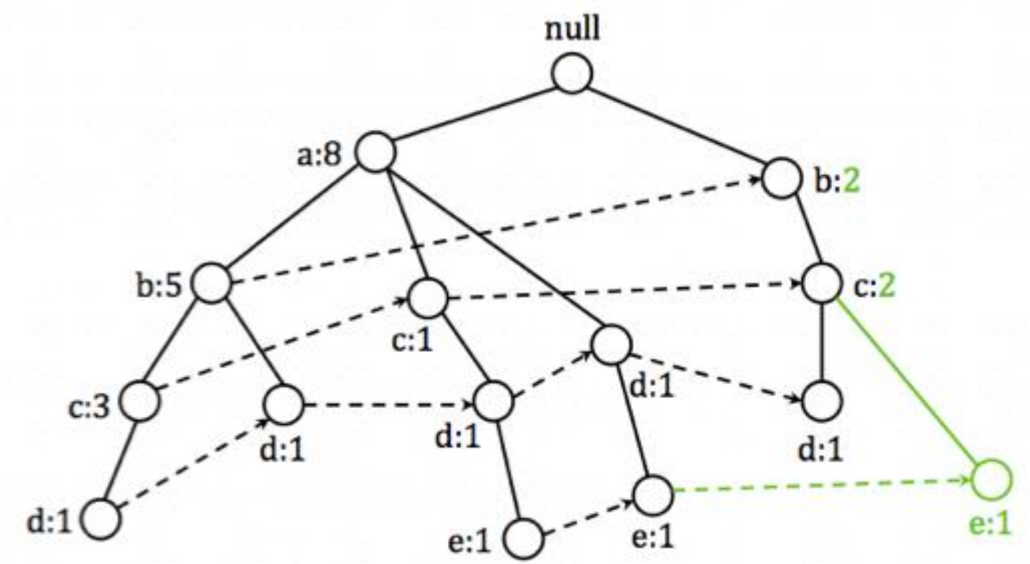
- **Step 10**:



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}

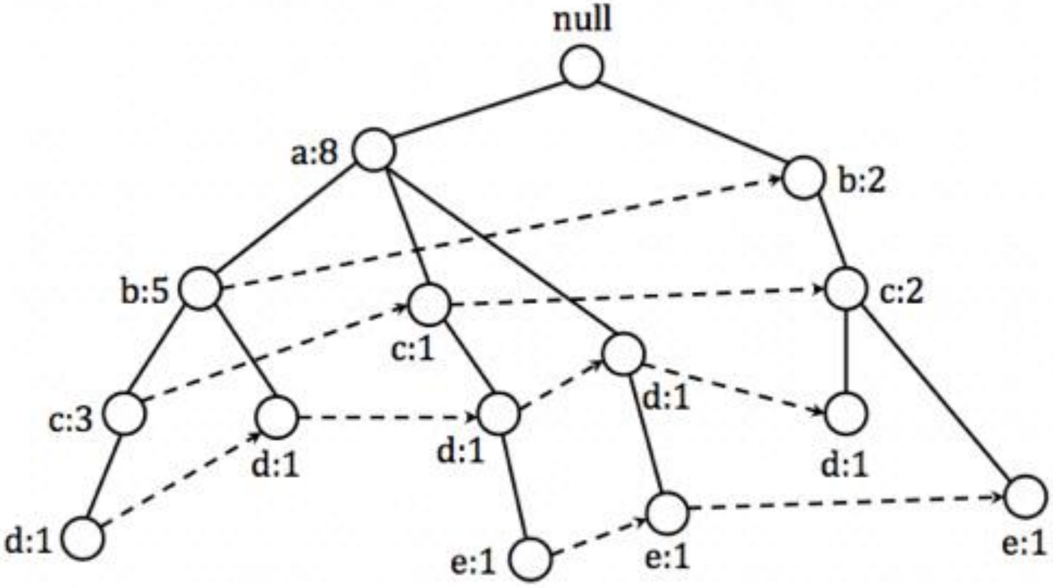
Constructing an FP Tree

- Step 11:



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}

The Final FP Tree



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}

Properties of an FP Tree

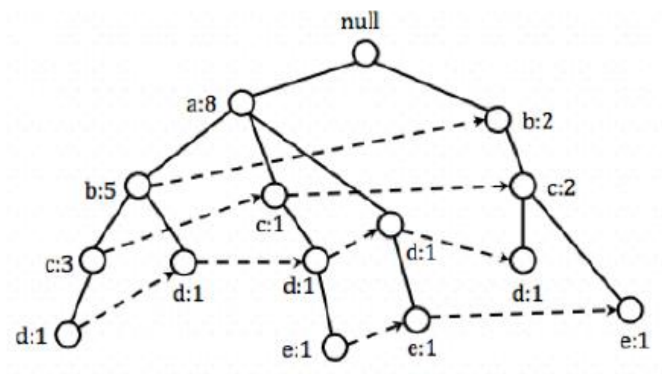
- An FP-tree is typically smaller than the size of the uncompressed data, because many transactions in market basket data often share a items in common
  - If all transactions have the same set of items, only one branch
  - If no transactions have common items, no compression

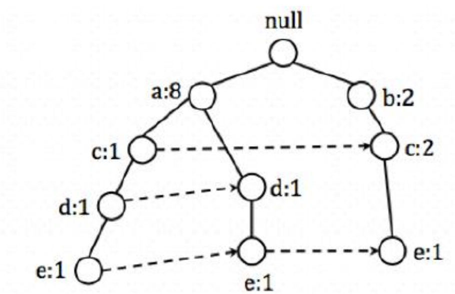
FP-Growth Frequent Itemset Generation

- FP-Growth is an algorithm that generates frequent itemsets from an FP-tree

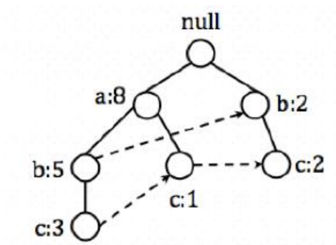
- The algorithm looks for frequent itemsets in a bottom-up fashion

Examples of FP-Growth Frequent Itemsets

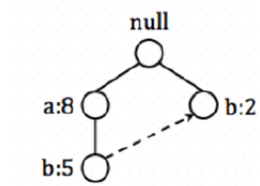




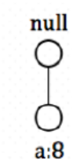
Paths containing node e



Paths containing node c



Paths containing node b



Paths containing node a

FP-Growth to Find Frequent Itemsets

- Gather all the paths containing the relevant node. These paths are called prefix paths

- From the prefix paths, the support count for the item is obtained by adding the support counts associated with the node

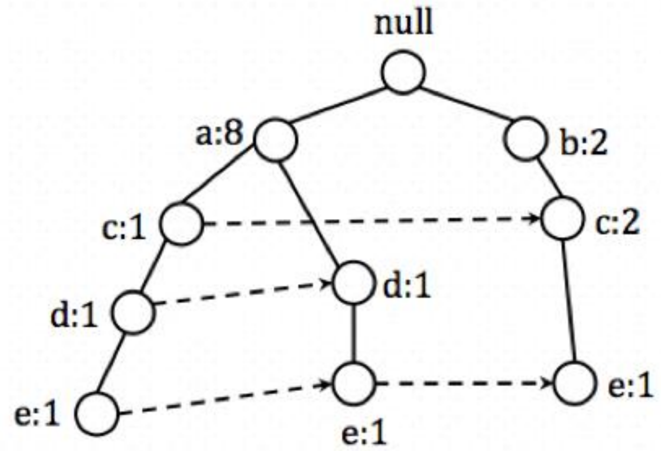
- If the item is frequent, the algorithm has to solve the subproblems of finding frequent itemsets that derive from this item. To do this, it first converts the prefix paths into a conditional FP-tree

Conditional FP Tree

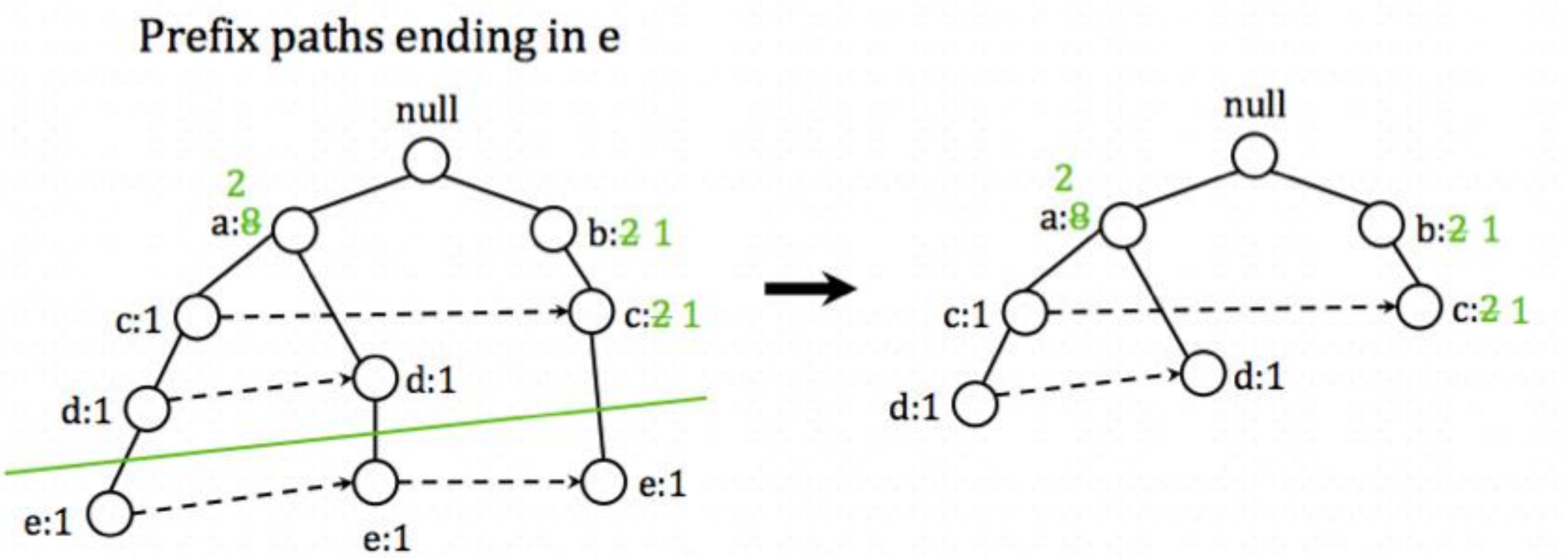
TID	Items
1	{a, b}
2	{b, c, d}
3	{a, c, d, e}
4	{a, d, e}
5	{a, b, c}
6	{a, b, c, d}
7	{a}
8	{a, b, c}
9	{a, b, d}
10	{b, c, e}

- For this example, let's set *minsup* = 2.

Prefix paths ending in e

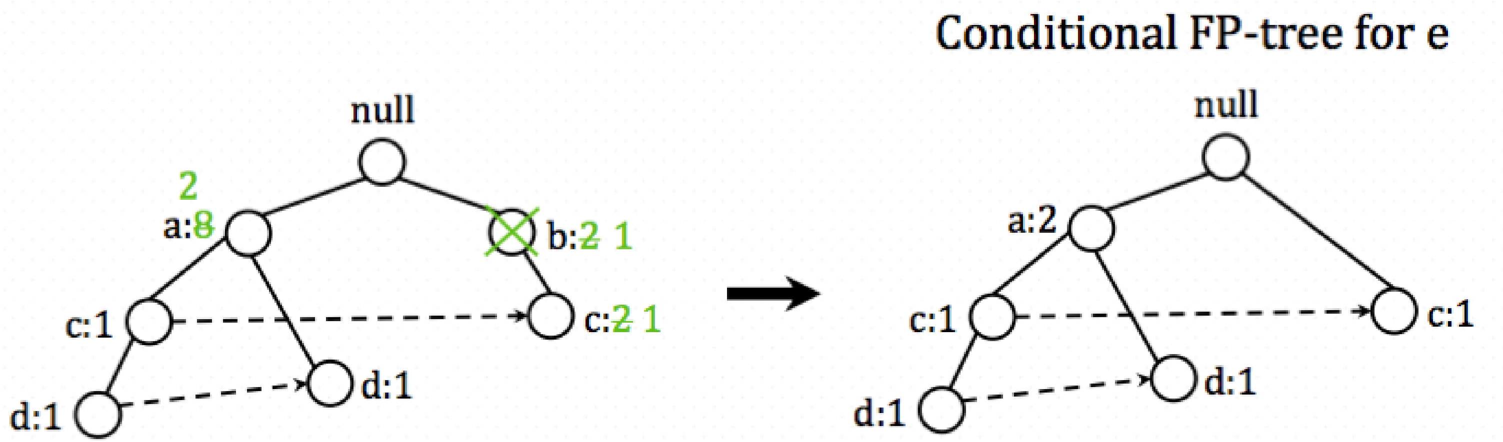


Conditional FP Tree



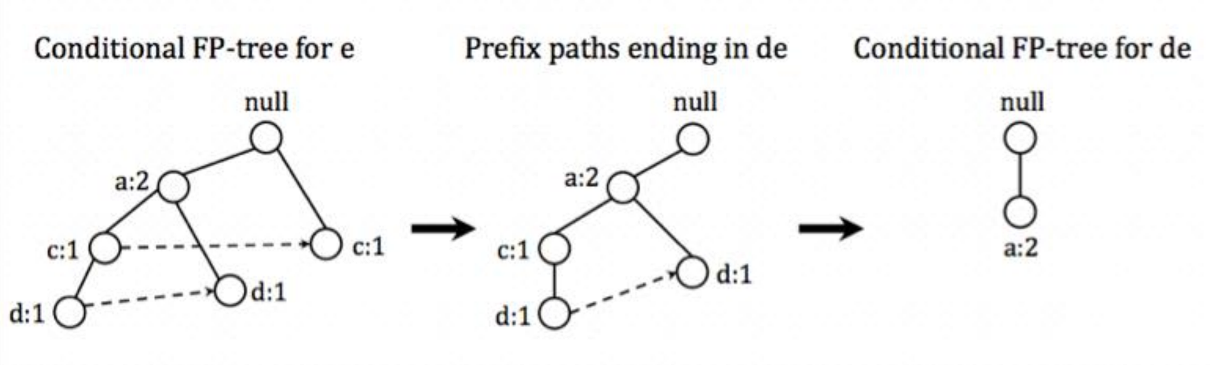
To obtain the conditional FP-tree for *e* from the prefix subtree ending in *e*, remove the nodes containing *e* (as this information is no longer needed).

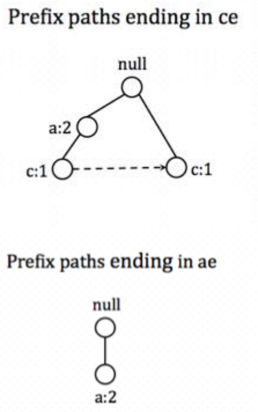
Conditional FP Tree



Now remove infrequent items (nodes) from the prefix paths. In this example, *b* has a support of 1 (note this really means *be* has a support of 1). That is, there is only 1 transaction containing *b* and *e*

Conditional FP Tree

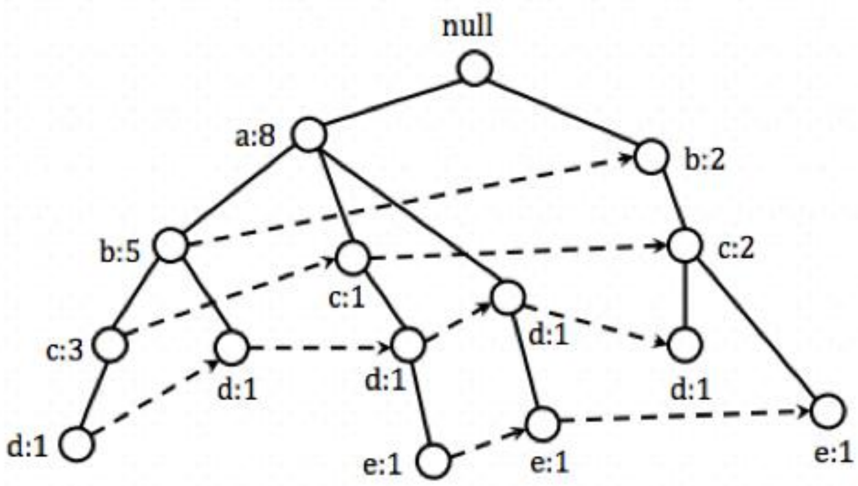




Use the conditional FP-tree for *e* to find frequent itemsets ending in *de*, *ce*, and *ae* (*be* is not considered as *b* is not in the conditional FP-tree for *e*). For each item, find the prefix paths from the conditional tree for *e*, generate the conditional FP-tree and continue recursively.

Conditional FP Tree

- Given the example tree below, FP-growth looks for frequent itemsets ending in *e* first, followed by *d*, *c*, *b*, and finally *a*



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}

Apriori vs. FP-Growth

- The biggest advantage found in FP-Growth is the fact that the algorithm only needs to read the file twice, as opposed to Apriori who reads it once for every iteration

Algorithm	Technique	Runtime	Memory Usage	Parallelizability
Apriori	Generate singletons, pairs, triplets, etc.	Candidate generation is extremely slow. Runtime increases exponentially depending on the number of different items.	Saves singletons, pairs, triplets, etc.	Candidate generation is very parallelizable
FP-Growth	Insert sorted items by frequency into a pattern tree	Runtime increases linearly, depending on the number of transactions and items	Stores a compact version of the database	Data are very inter dependent, each node needs the root

Apriori vs. FP-Growth

- FP-Growth beats Apriori by far. It has less memory usage and less runtime. FP-Growth is more scalable because of its linear running time

File	Apriori	FP-Growth
Simple Market Basket test file	3.66s	3.03s
"Real" test file (1 Mb)	8.87s	3.25s
"Real" test file (20 Mb)	34m	5.07s
Whole "real" test file (86 Mb)	4+ hours (Never finished, crashed)	8.82s

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

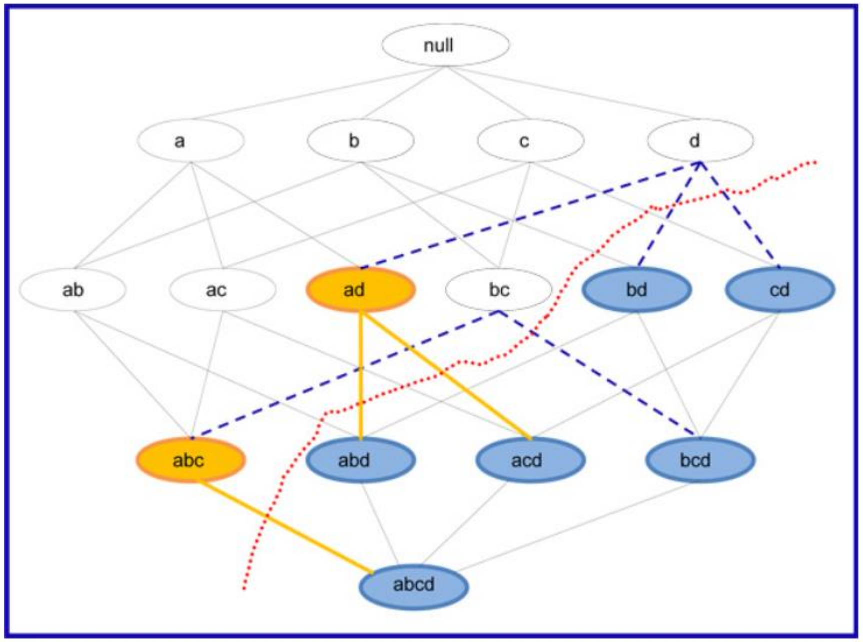
Maximal and Closed Frequent Itemsets

Motivation

- What happens when you have a large market basket data with over a hundred items?
- The number of frequent itemsets grows exponentially and this in turn creates an issue with storage
  - Need an alternative representation
- The **Maximal** and **Closed** Frequent Itemsets are two such representations that are subsets of the larger frequent itemsets

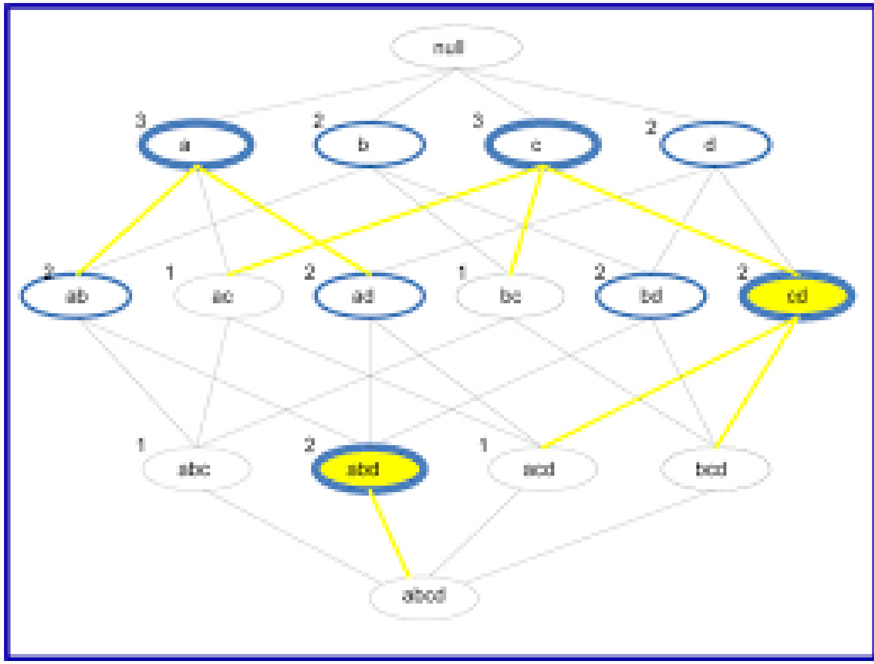
Maximal Frequent Itemset

- It is a frequent itemset for which none of its immediate supersets are frequent
- ***Identification***
  - Examine the frequent itemsets that appear at the border between the infrequent and frequent itemsets
  - Identify all of its immediate supersets
  - If none of the immediate supersets are frequent, the itemset is maximal frequent



Closed Frequent Itemset

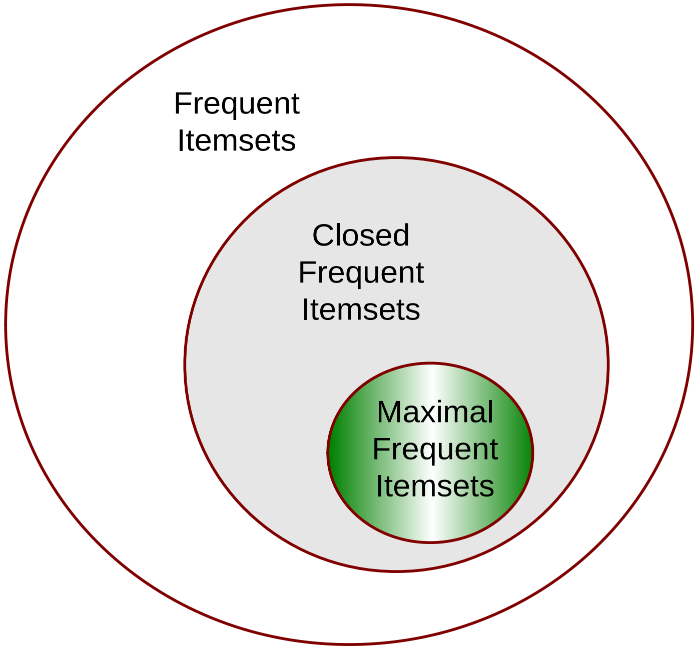
- An itemset is closed in a data set if there exists no superset that has the same support count as this original itemset
- ***Identification***
  - First identify all frequent itemsets
  - Check to see if there exists a superset that has the same support as the frequent itemset, if there is, the itemset is disqualified, otherwise, the itemset is closed



Relationship Between Frequent Itemset Representations

- Maximal frequent itemsets are a more compact representation because it is a subset of closed frequent itemsets

- Closed frequent itemsets are more widely used than maximal frequent itemset
  - When efficiency is more important than space, they provide us with the support of the subsets, so no additional pass is needed to find this information



PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Measures of Interestingness

Motivation

> A very important aspect of data mining research is the determination of how interesting a pattern is

> Researchers have created 9 measures of interestingness

- Conciseness
- Generality
- Reliability
- Peculiarity
- Diversity
- Novelty
- Surprisingness
- Utility
- Actionability

> More details at http://www2.cs.uregina.ca/~dbd/cs831/notes/objectiveMeasur[illegible]index.html

Lift of Association Rule

- Also defined as “Interest”

$$\text{lift}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X) \times \text{supp}(Y)}$$

- If the rule had a lift of 1, it would imply that the probability of occurrence of the antecedent and that of the consequent are independent of each other, thus no rule can be drawn involving those two events
- If the lift is > 1, it would imply those two occurrences are dependent on one another, and makes those rules potentially useful for predicting the consequent in future data sets
- If the lift is < 1, it would imply that presence of one item has negative effect on presence of other item and vice versa

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

Hands-on Lab

Final Thoughts

- Hopefully, you had fun learning to learn!

- Get your hands dirty with EDA and model development

- Follow AI/Engineering blogs to learn how production ML systems are built
  Amazon: https://www.amazon.science/blog
  Google: https://ai.googleblog.com/
  Uber: https://eng.uber.com/

- You can reach me through UW email or feel free to add me on Linked

ON-BRAND STATEMENT

FOR GENERAL USE

> What defines the students and faculty of the University of Washington? Above all, it's our belief in possibility and our unshakable optimism. It's a connection to others, both near and far. It's a hunger that pushes us to tackle challenges and pursue progress. It's the conviction that together we can create a world of good. And it's our determination to Be Boundless. Join the journey at **uw.edu**.