Clustering & Association

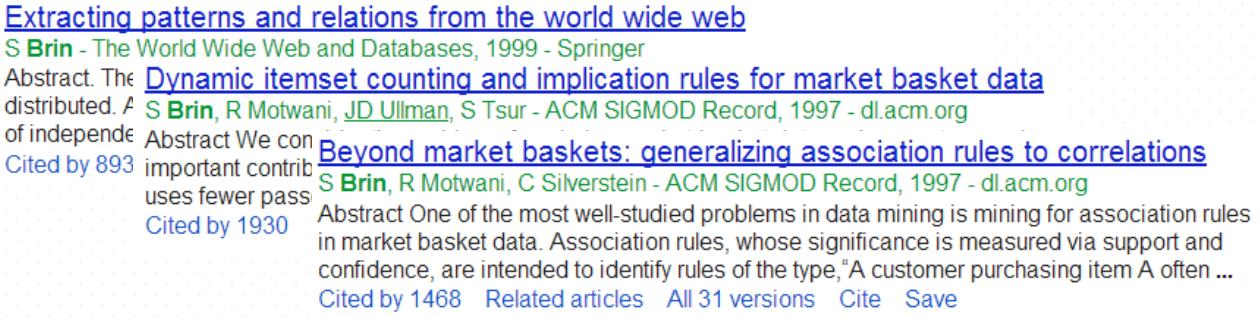
Follow me on LinkedIn for more:
Steve Nouri
https://www.linkedin.com/in/stevenouri/

Association Rules

The process of discovering interesting relations between variables in large datasets.

Association Rules

- Introduced by Agrawal, Imielinski and Swami in 1993
- Vastly studied over the years, with countless improvements proposed by many researchers
- Before dropping out of Stanford's graduate program, Sergey Brin (Google's co-founder) published several papers on Association Rules



Association Rules

1. Market basket analysis
2. The Apriori algorithm
3. FP-Growth

Market Basket Analysis

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

Market Basket Analysis

- Example:
  - 5% of transactions contain both these items
  - 30% of the transactions containing beer also contain diapers

- Beer ⇒ Diapers(0.05, 0.30)
  - 5% – **Support** of the rule
  - 30% – **Confidence** of the rule

Market Basket Analysis Applications

- Retail: *what sells with what*
- Marketing : *population segments, recommendations, etc.*
- Finance: *investment portfolios, "basket of stocks"*
- Biology: *genetics, microarrays, gene expressions*

Market Basket Analysis - Definitions

- $I = \{i_1, i_2, \dots, i_n\}$: 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 $X \subseteq T$
- An **association rule** is an implication of the form $X \Longrightarrow Y$, where $X, Y \subset I$
- 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

Market Basket Analysis – Definitions (cont.)

- **Itemset**:
  - A collection of one or more items
    - Example: $\{Milk, Bread, Diaper\}$
  - **k-itemset**
    - An itemset containing k items
- Support count ($\boldsymbol{\sigma}$)
  - Frequency of an itemset
    - $\sigma(\{Milk, Bread, Diaper\}) = 2$
- Support ($\boldsymbol{s}$)
  - Fraction of transactions containing an itemset
  - $s(\{Milk, Bread, Diaper\}) = \frac{2}{5}$
- **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

Association rules

- Ex.: $\{Milk, Diaper\} \Rightarrow \{Beer\}$

- Rule Evaluation Metrics
  - Support:

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

  - Confidence:

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

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

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 – Caution!

- 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

Example

- Itemset $I = \{i_1, i_2, \dots, i_n\}$
- 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

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

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

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

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

Strong Association Rules

- Most times, we will be interested in finding strong association rules
  - $sup(R) \geq minsup$ and $conf(R) \geq minconf$
- The problem of finding weak association rules can also have interesting applications
  - "What two items are almost never purchased together?"

The Association Rule Mining Problem

Definition. Given a set of transactions T, find all the rules having support $\geq minsup$ and confidence $\geq minconf$, where $minsup$ and $minconf$ are the corresponding support and confidence thresholds.

The Association Rule Mining Problem

- The brute-force approach
  - Compute the support and confidence for every possible rule
- Prohibitively expensive
  - Given an itemset with n items, there exist $\boldsymbol{R = 3^n + 2^{n+1} + 1}$ rules
- Suppose we had 6 items:

$$R = 3^6 + 2^{6+1} + 1 = 602$$

Mining Association Rules

- Two-step approach:

1. Frequent Itemset Generation
   - Generate all item sets whose support $\geq minsup$
     - Note that the support of a rule $X \Rightarrow Y$ depends only on the support of $X \cup Y$
     - All rules below have the same support:

$$\{Beer, Diapers\} \Rightarrow \{Milk\}$$
$$\{Diapers, Milk\} \Rightarrow \{Beer\}$$
$$\{Milk\} \Rightarrow \{Beer, Diapers\}$$
$$\{Beer\} \Rightarrow \{Milk, Diapers\}$$
$$\ldots$$

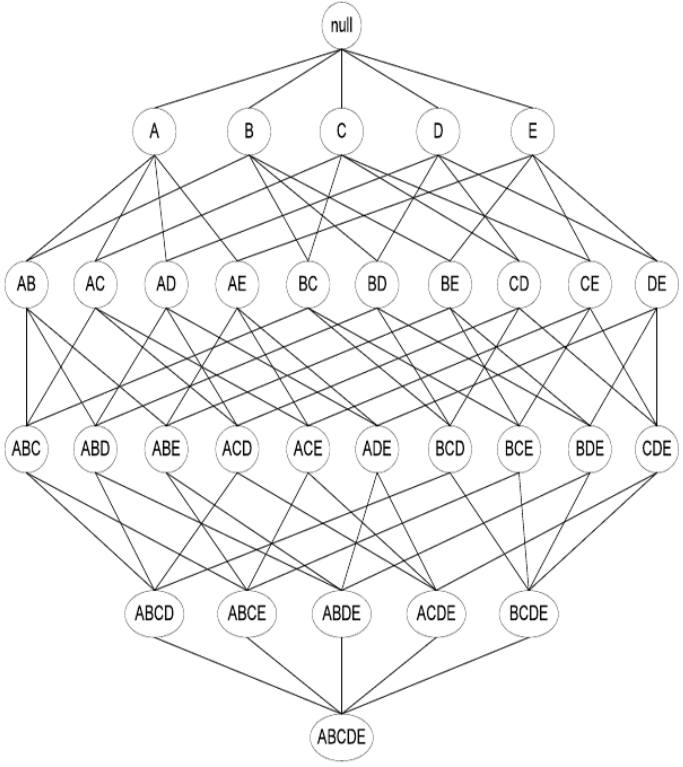
Mining Association Rules

- Two-step approach:

  1. Frequent Itemset Generation
     - Generate all item sets whose support $\geq$ *minsup*
  2. Rule Generation
     - Generate high confidence (strong) rules from each frequent itemset

- The computational complexity of (1) is higher.

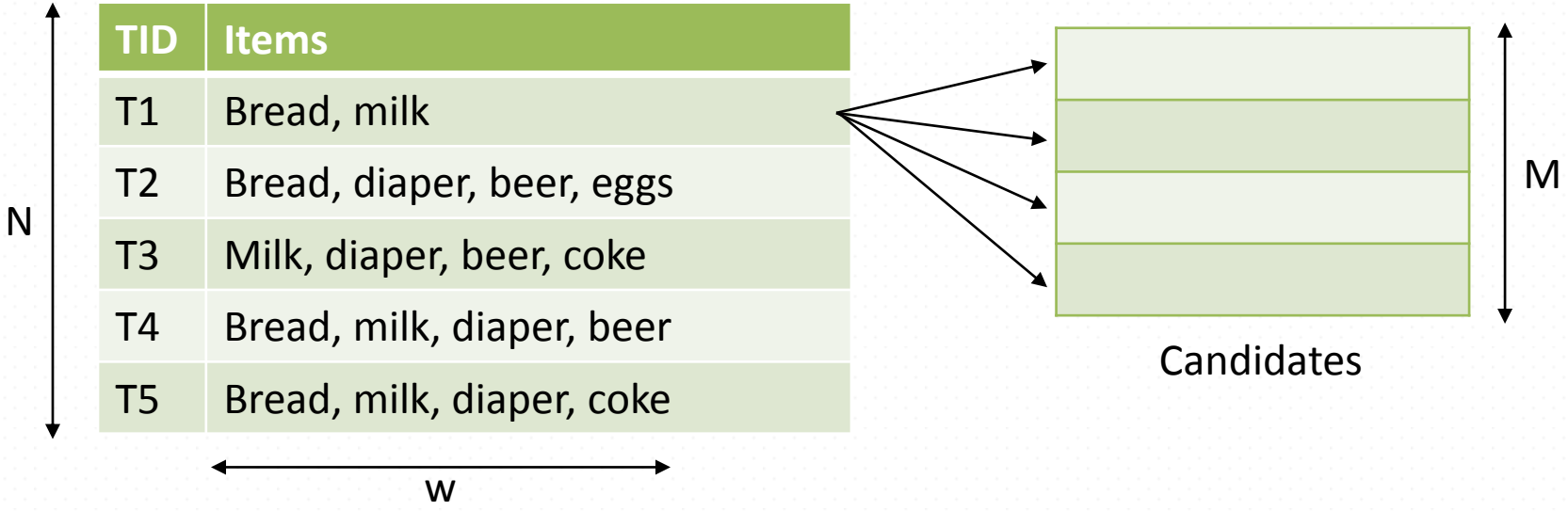
Frequent Itemset Generation



Itemset lattice

- For a set with n items:
  - 2^n-1 possible itemsets
- Each of these is called a *candidate* frequent itemset

Frequent Itemset Generation



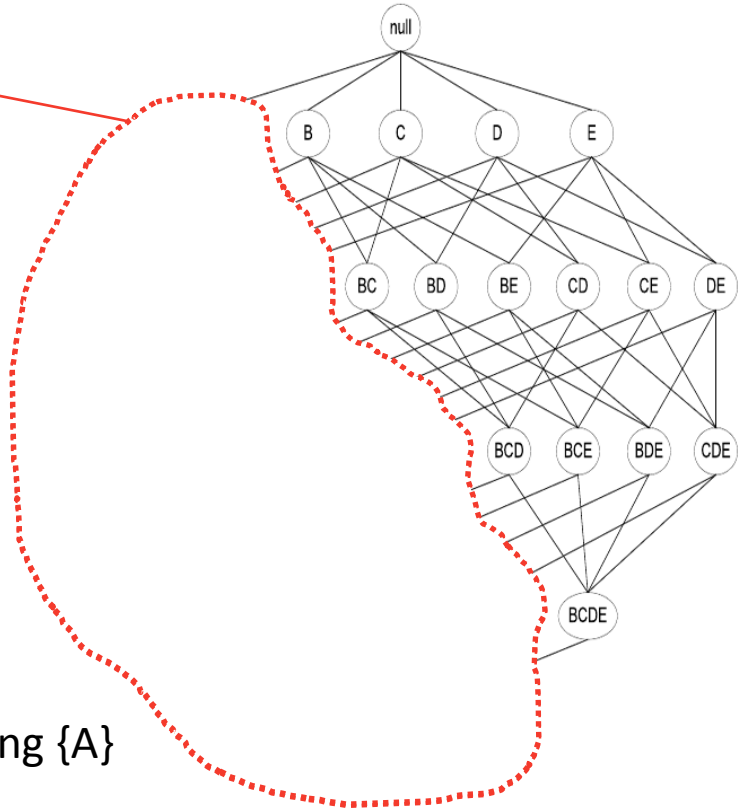
- Compare each candidate against every transaction
- Complexity: $O(NMw)$

Frequent Itemset Generation

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

Apriori Algorithm

If {A} is infrequent



Prune all itemsets containing {A}

Clustering & Association

Recall …

$$\{Milk, Diaper\} \Rightarrow \{Beer\}$$

- Rule Evaluation Metrics
  - Support:

$$s = \frac{2}{5} = 0.4$$

  - Confidence:

$$c = \frac{2}{3} = 0.67$$

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

The Association Rule Mining Problem

Definition. Given a set of transactions T, find all the rules having support $\geq minsup$ and confidence $\geq minconf$, where $minsup$ and $minconf$ are the corresponding support and confidence thresholds.

Mining Association Rules

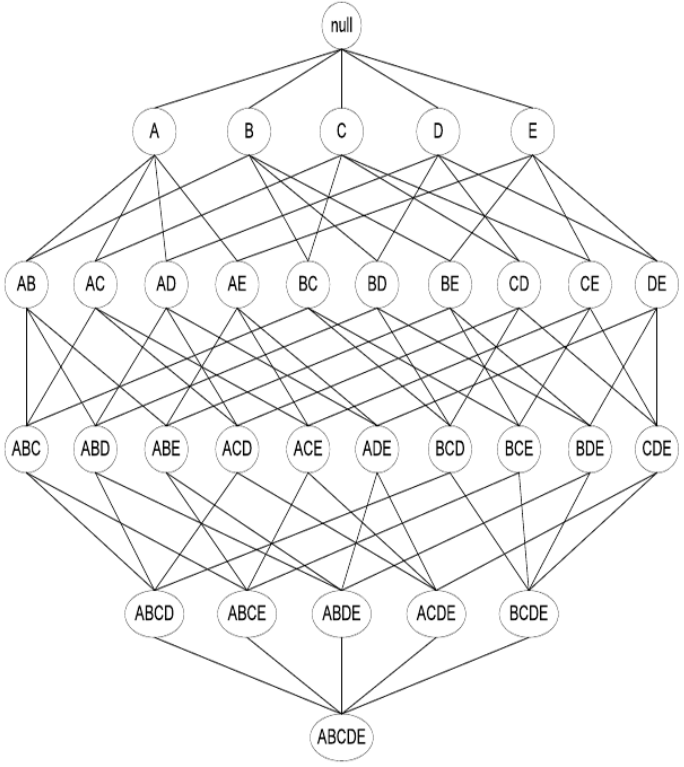
- Two-step approach:

1. Frequent Itemset Generation
   - Generate all item sets whose support $\geq$ *minsup*

Expensive

2. Rule Generation
   - Generate high confidence (strong) rules from each frequent itemset

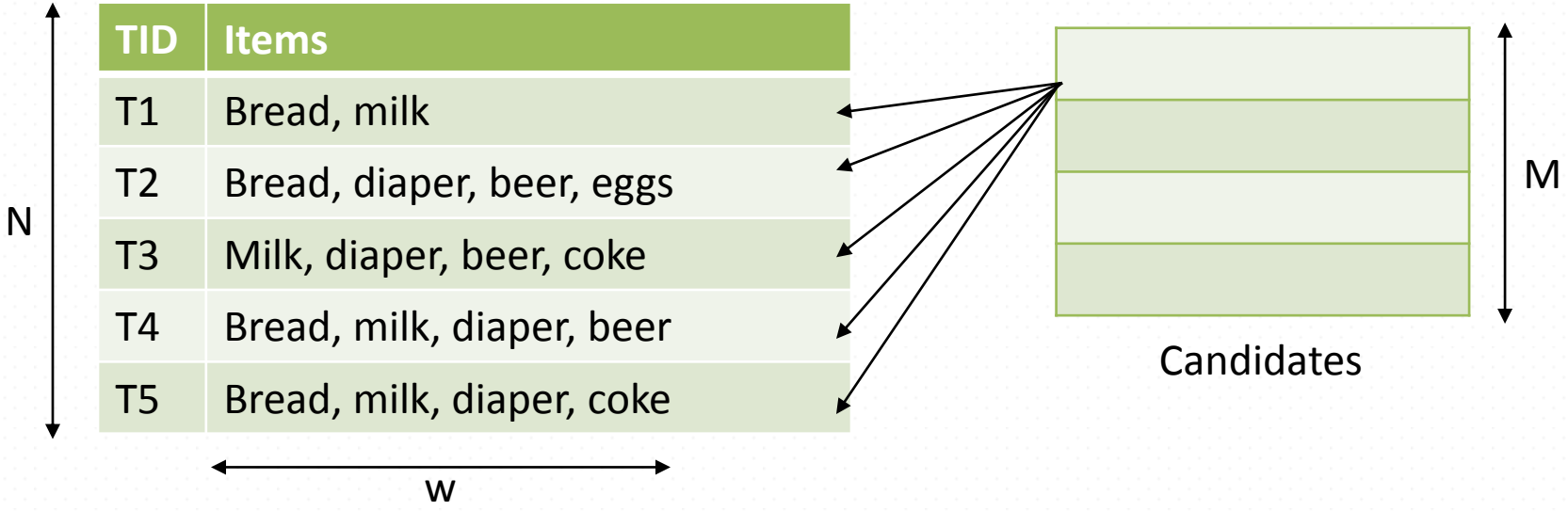
Frequent Itemset Generation



Itemset lattice

- For a set with n items:
  - 2^n-1 possible itemsets
- Each of these is called a *candidate* frequent itemset

Frequent Itemset Generation



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

Frequent Itemset Generation

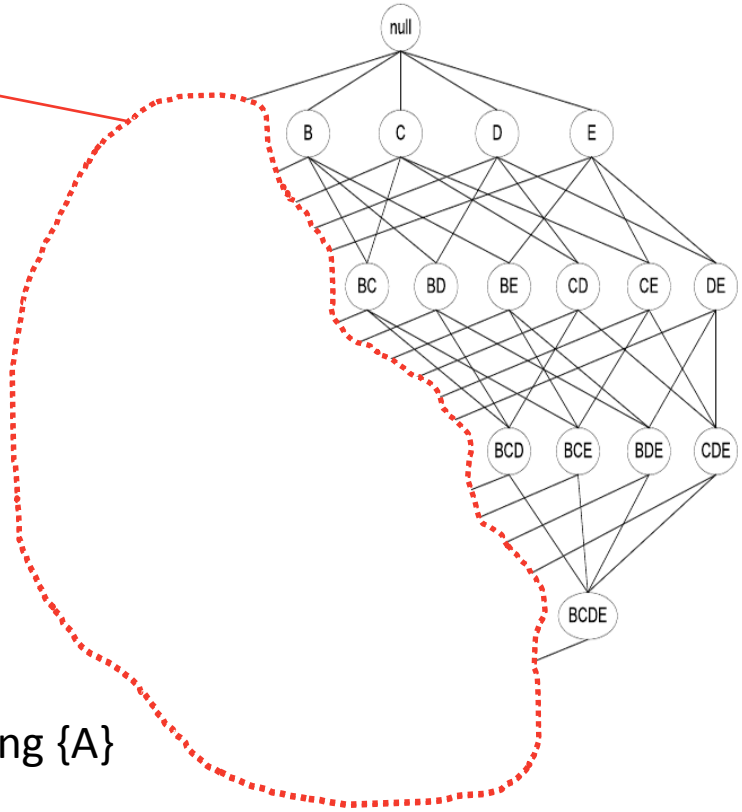
- Several ways to reduce the computational complexity:
  - **Reduce the number of candidate itemsets** (M) by pruning the itemset lattice → **Apriori Algorithm**
  - **Reduce the number of comparisons** by using advanced data structures to store the candidate itemsets or to compress the dataset → **FP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

Association Rules

1. Market basket analysis
2. The Apriori algorithm
3. FP-Growth

Apriori Algorithm

If {A} is infrequent



Prune all itemsets containing {A}

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:

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

  - The support of an itemset never exceeds that of its subsets
  - This is known as the **anti-monotone** property of support

Apriori Algorithm

- For example:
  - If $\{beer, diaper, oreos\}$ is frequent, so is $\{beer, diaper\}$ since every transaction having the first also contains the latter.
- Apriori pruning principle: If we identify an itemset as being infrequent, its subsets should not be generated/tested.
- Method:
  - 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

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

$minsup_{count} = 2$

- With that, we have generated all frequent itemsets of *D*!
- Can this be improved?

Apriori Algorithm Illustrated

- We have just illustrated the $\boldsymbol{F_{k-1} \times F_1}$ **method**
  - Where we combine the frequent itemsets of size $k-1$ with all frequent itemsets of size 1 to generate F_k
- But we can do better!
- $\boldsymbol{F_{k-1} \times F_{k-1}}$ **method**

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_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

Last caveat…

- After merging each F_{k-1} pairs, we must prune the result set!
- Suppose we have $F_3 = \{abc, abd, acd, ace, bcd\}$
  - Merging $F_3 \times F_3$ gives $\{abcd, acde\}$
  - But the subset $\{ade\}$ is not in F_3
  - So we prune $acde$
  - $C_4 = \{abcd\}$

Other important details

- How do we count supports of candidates? (chap 6.2.4)
- Computational complexity
  - Affected by *minsup*, number of items, number of transactions, transaction width, candidate generation, support counting

Another example – Store Transactions

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

F_1

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

F_2

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

F_3

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

$minsup_{count} = 2$

Mining Association Rules

- Two-step approach:

1. Frequent Itemset Generation
   - Generate all item sets whose support $\geq$ *minsup*
2. Rule Generation
   - Generate high confidence (strong) rules from each frequent itemset

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 – Example(cont.)

- 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

And now...

Lets see some Aprioring!

Mining Association Rules

Recall the two-step Approach:

1. **Frequent Itemset Generation** *Expensive*
   Generate all item sets whose support $\geq$ *minsup*

2. **Rule Generation**
   Generate high confidence (strong) rules from each frequent itemset

Frequent Itemset Generation

Several ways to reduce the computational complexity:

1. **Reduce the number of candidate itemsets** (M) by pruning the itemset lattice → **Apriori Algorithm**

2. **Reduce the number of comparisons** by using advanced data structures to store the candidate itemsets or to compress the dataset → **FP-Growth**

Key to FP-Growth

Efficiently encode the data using a compact data structure, from which frequent itemsets can be directly retrieved.

This method differs the "generate-and-test" paradigm employed by Apriori.

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.

Example of FP-Tree Construction

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}

- Consider the dataset to the left.
  - Ten total transactions or itemsets.
  - Five total items (a, b, c, d, and e).

Example of FP-Tree Construction

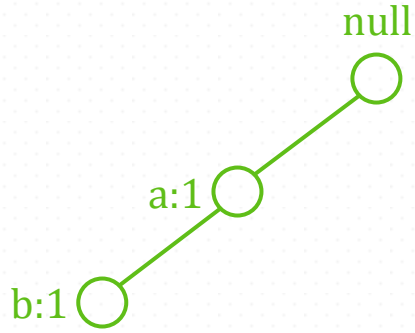
TID	Items
1	{a, b}
2	{b, c, d}
3	{a, c, d, e}
4	{a, d, e}
5	{a, b, c}
6	{a, b, c, d}
7	{a}
8	{a, b, c}
9	{a, b, d}
10	{b, c, e}

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*.

Example of FP-Tree Construction

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}

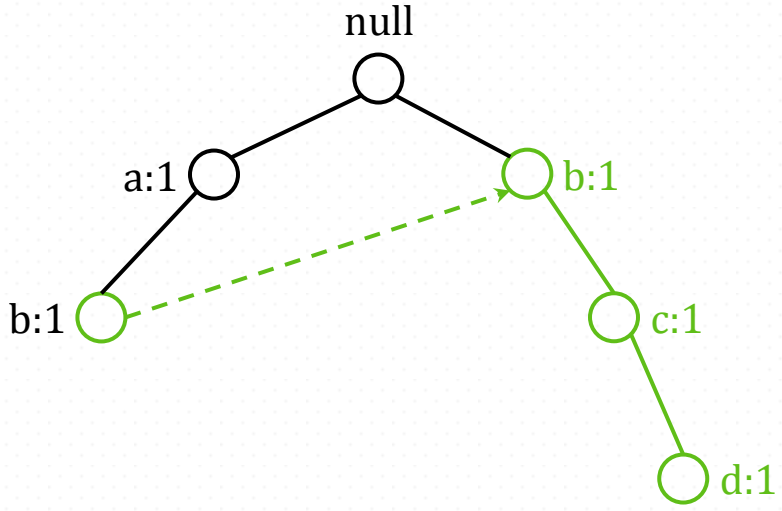


2

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 $\rightarrow a \rightarrow b$ to encode the transaction. Every node along the path has a frequency count of 1.

Example of FP-Tree Construction

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}

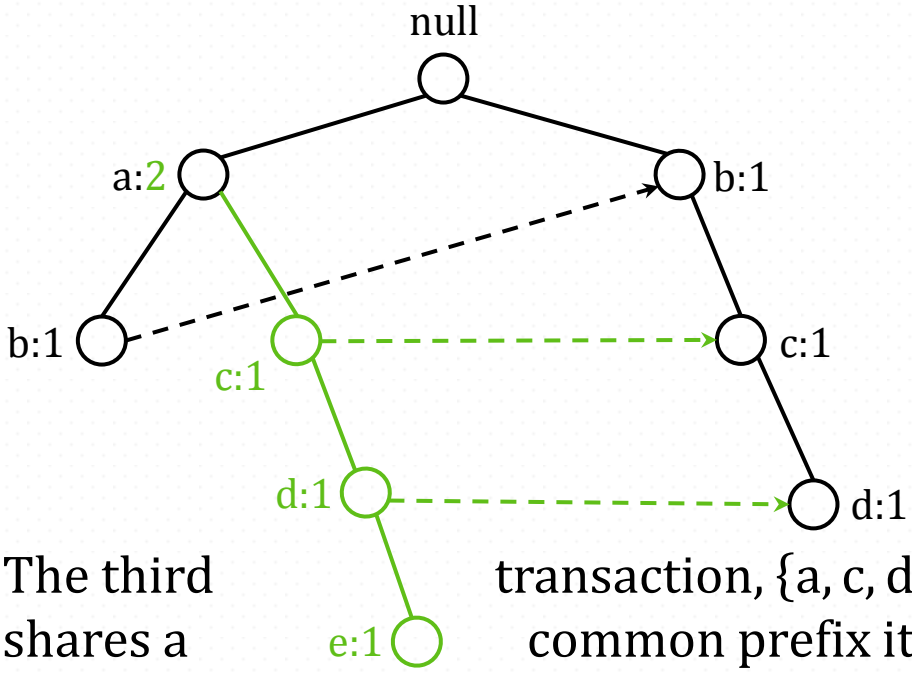


3

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 $\rightarrow b \rightarrow c \rightarrow d$.

Example of FP-Tree Construction

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}

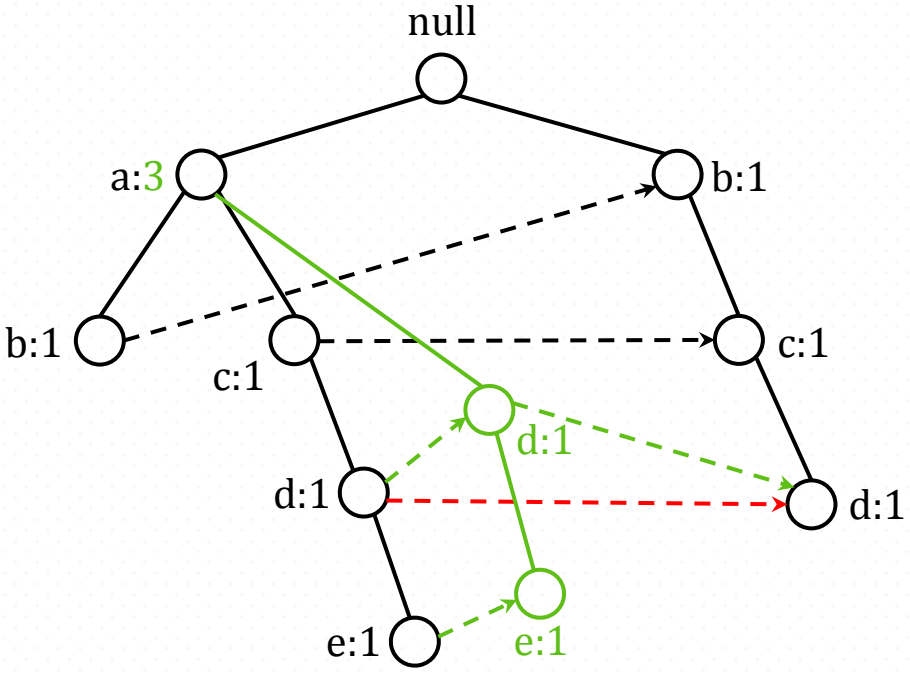


4

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.

Example of FP-Tree Construction

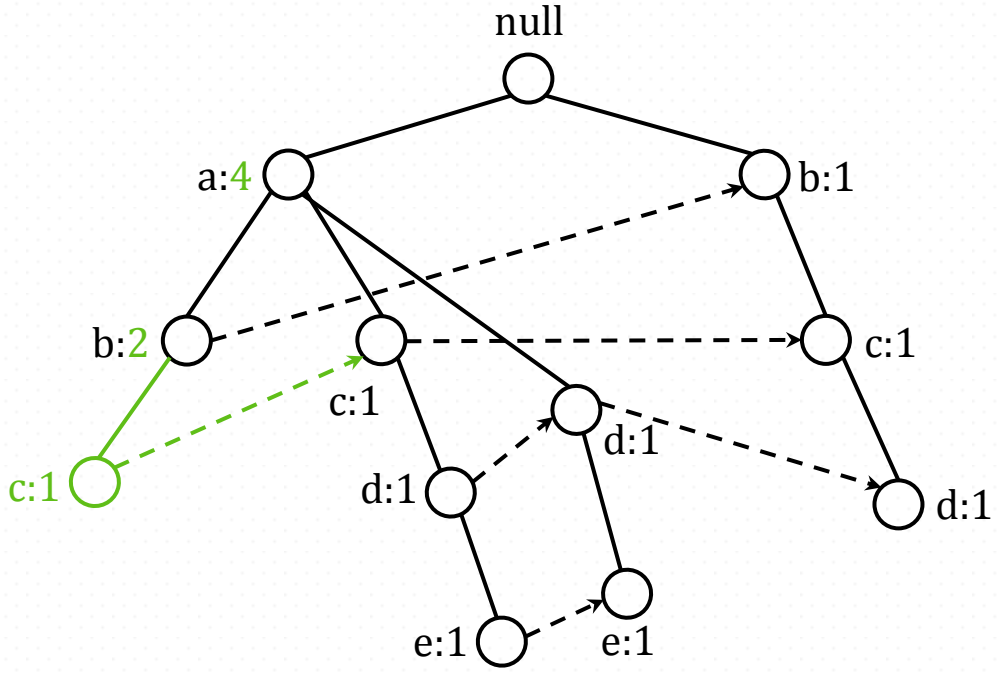
TID	Items
1	{a, b}
2	{b, c, d}
3	{a, c, d, e}
4	**{a, d, e}**
5	{a, b, c}
6	{a, b, c, d}
7	{a}
8	{a, b, c}
9	{a, b, d}
10	{b, c, e}



5 And the tree grows…

Example of FP-Tree Construction

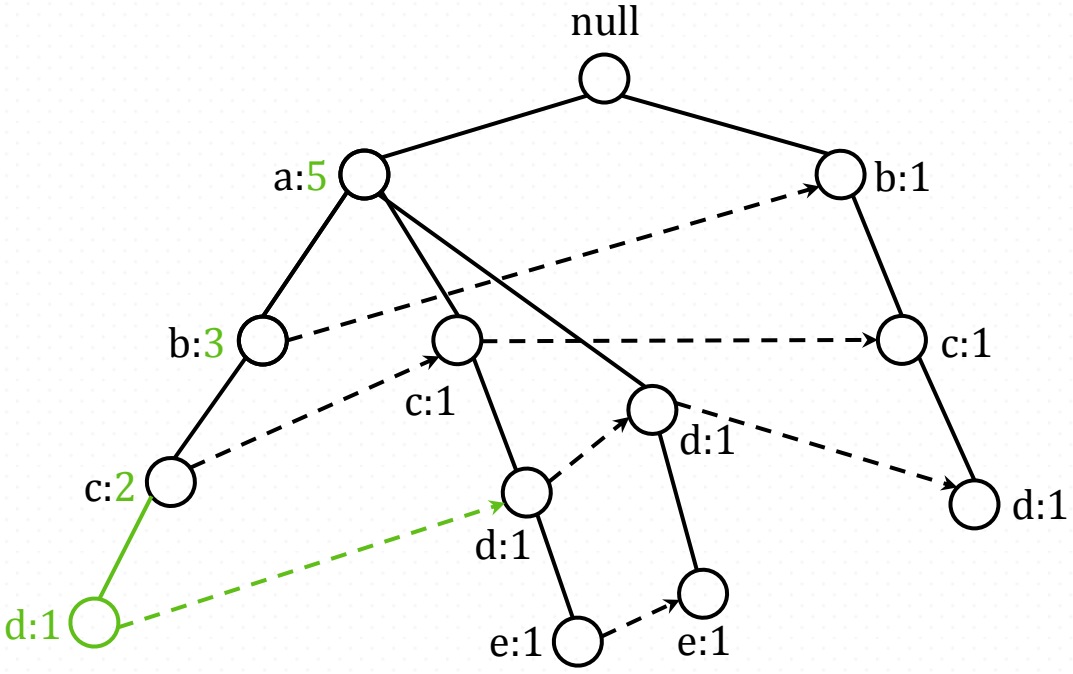
TID	Items
1	{a, b}
2	{b, c, d}
3	{a, c, d, e}
4	{a, d, e}
5	**{a, b, c}**
6	{a, b, c, d}
7	{a}
8	{a, b, c}
9	{a, b, d}
10	{b, c, e}



6 And grows…

Example of FP-Tree Construction

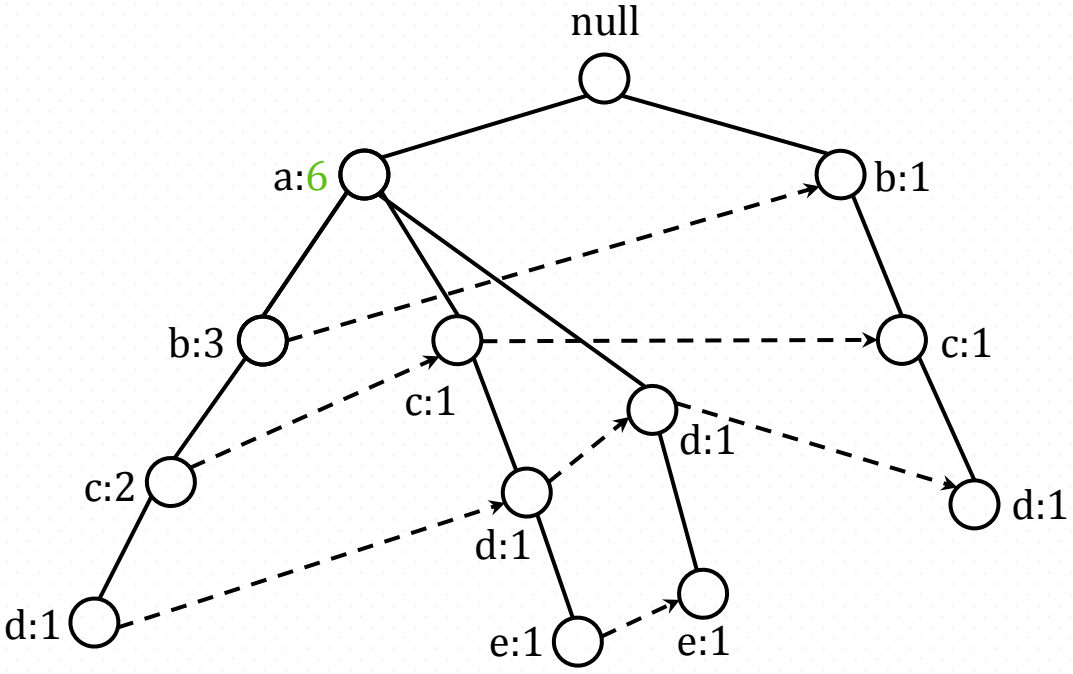
TID	Items
1	{a, b}
2	{b, c, d}
3	{a, c, d, e}
4	{a, d, e}
5	{a, b, c}
6	**{a, b, c, d}**
7	{a}
8	{a, b, c}
9	{a, b, d}
10	{b, c, e}



7 And grows…

Example of FP-Tree Construction

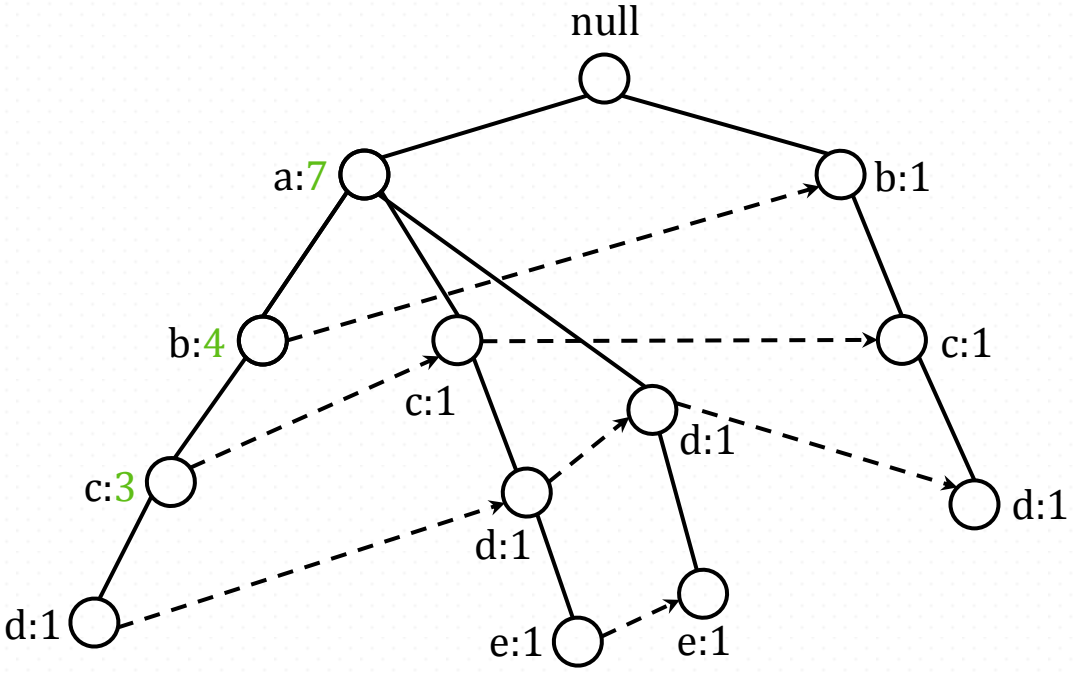
TID	Items
1	{a, b}
2	{b, c, d}
3	{a, c, d, e}
4	{a, d, e}
5	{a, b, c}
6	{a, b, c, d}
7	**{a}**
8	{a, b, c}
9	{a, b, d}
10	{b, c, e}



8 And grows…

Example of FP-Tree Construction

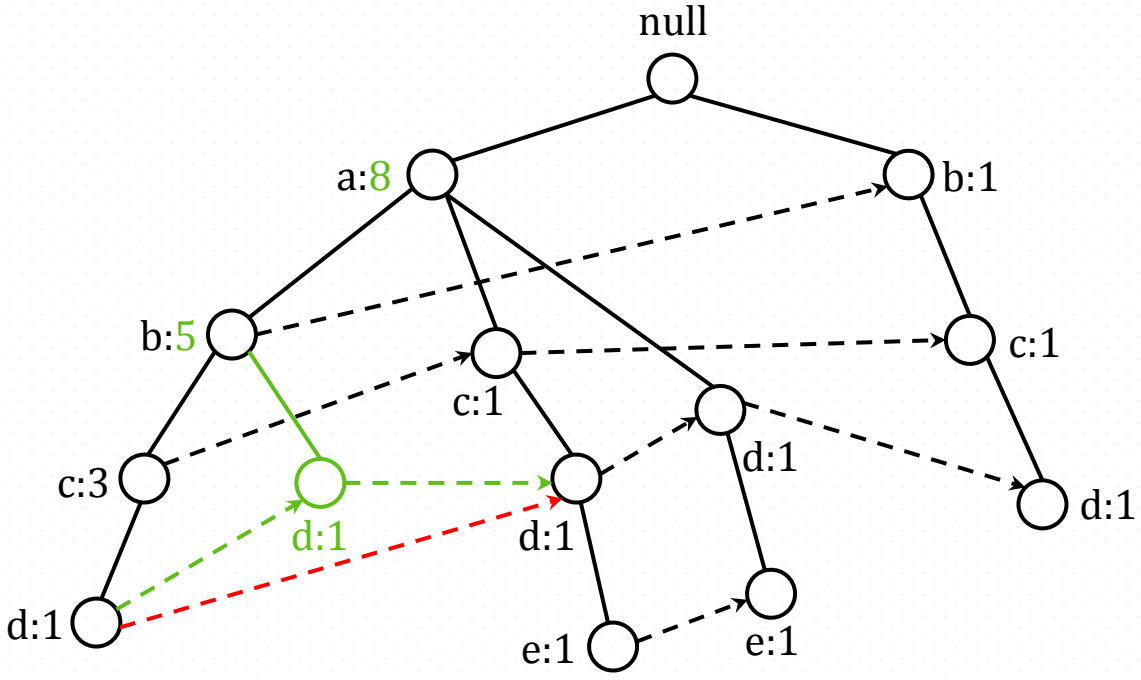
TID	Items
1	{a, b}
2	{b, c, d}
3	{a, c, d, e}
4	{a, d, e}
5	{a, b, c}
6	{a, b, c, d}
7	{a}
8	**{a, b, c}**
9	{a, b, d}
10	{b, c, e}



9 And grows…

Example of FP-Tree Construction

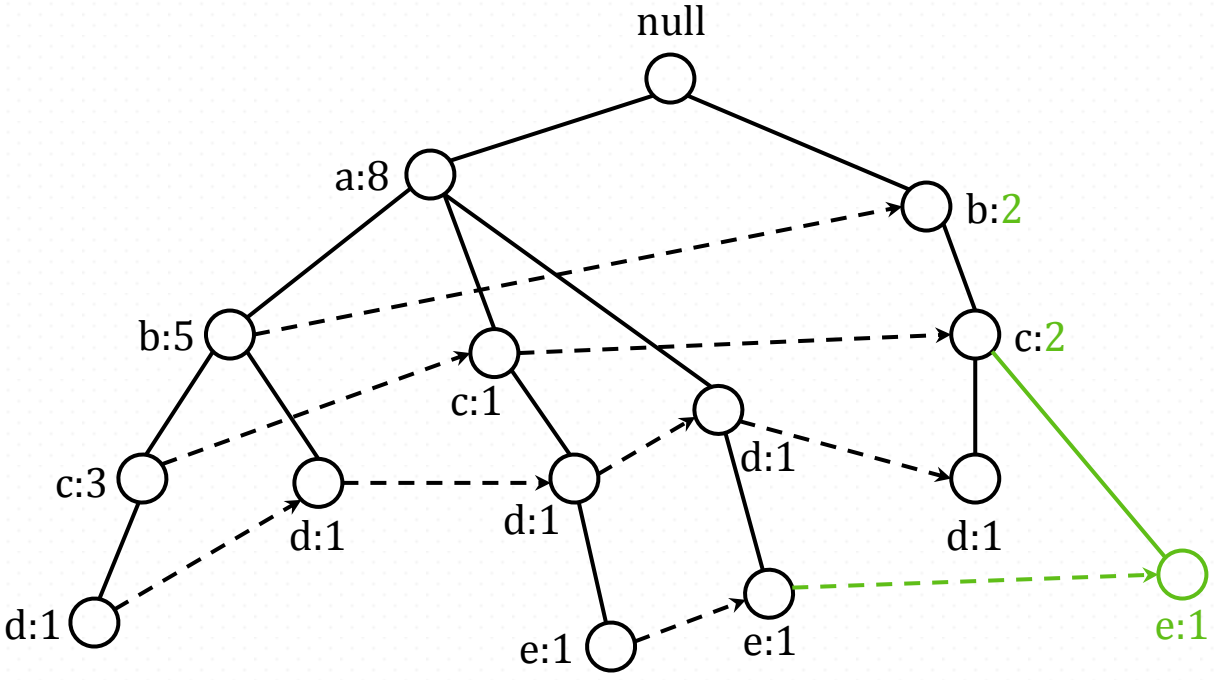
TID	Items
1	{a, b}
2	{b, c, d}
3	{a, c, d, e}
4	{a, d, e}
5	{a, b, c}
6	{a, b, c, d}
7	{a}
8	{a, b, c}
9	**{a, b, d}**
10	{b, c, e}



10 And grows…

Example of FP-Tree Construction

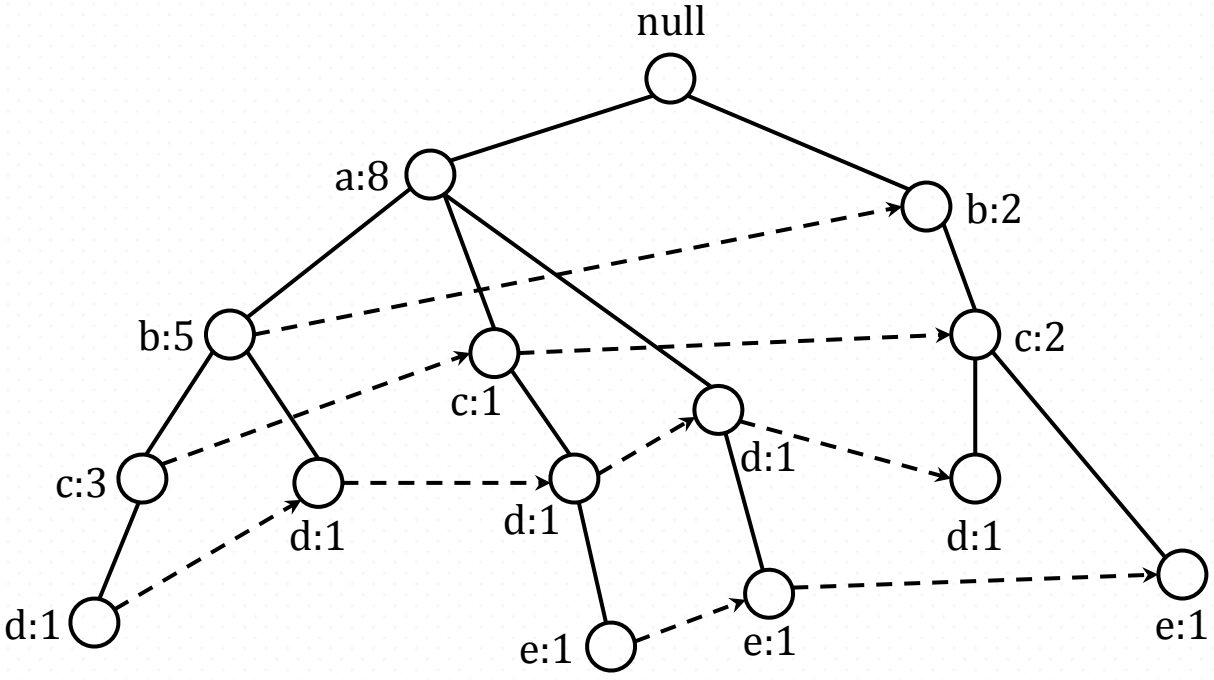
TID	Items
1	{a, b}
2	{b, c, d}
3	{a, c, d, e}
4	{a, d, e}
5	{a, b, c}
6	{a, b, c, d}
7	{a}
8	{a, b, c}
9	{a, b, d}
10	**{b, c, e}**



11 And finally stops.

Example of FP-Tree Construction

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

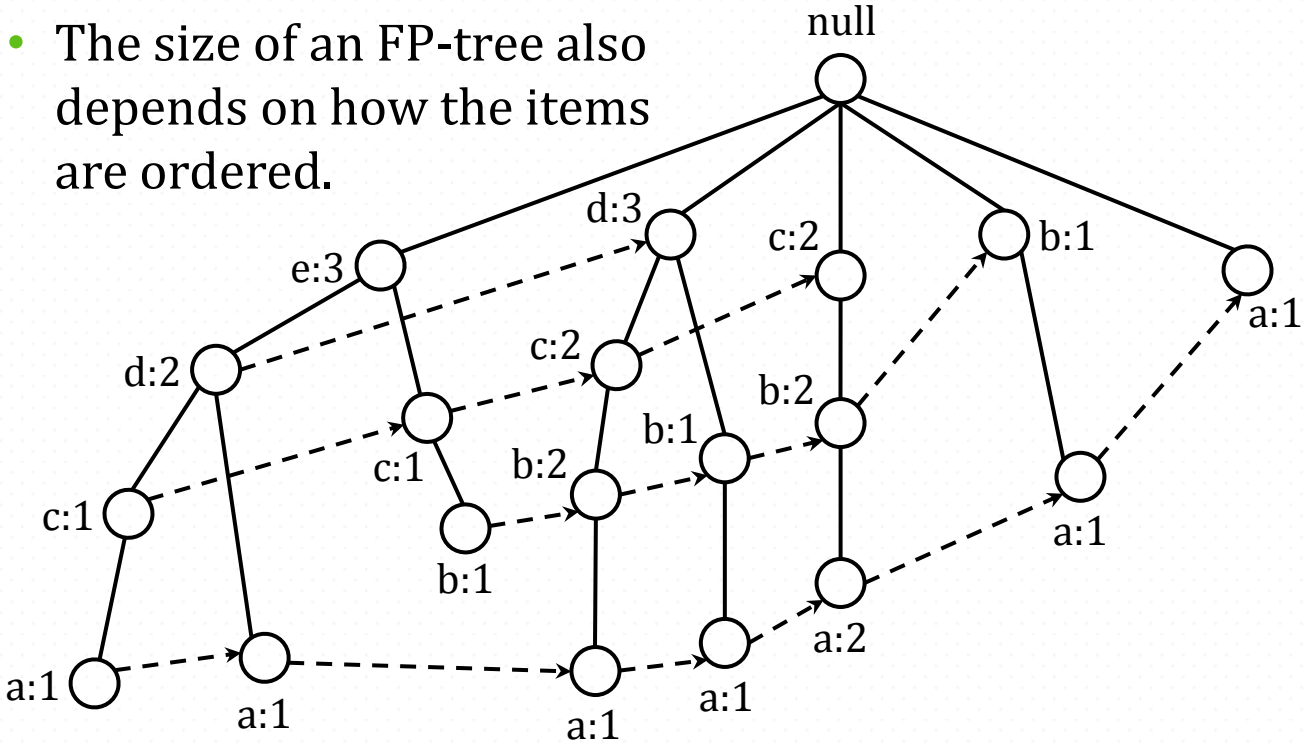
Notes on FP-Tree Construction

- 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.
- However, the physical storage requirement for the FP-tree is higher than the original data, because it requires additional space to store pointers between nodes and counters for each item.

Notes on FP-Tree Construction

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 size of an FP-tree also depends on how the items are ordered.

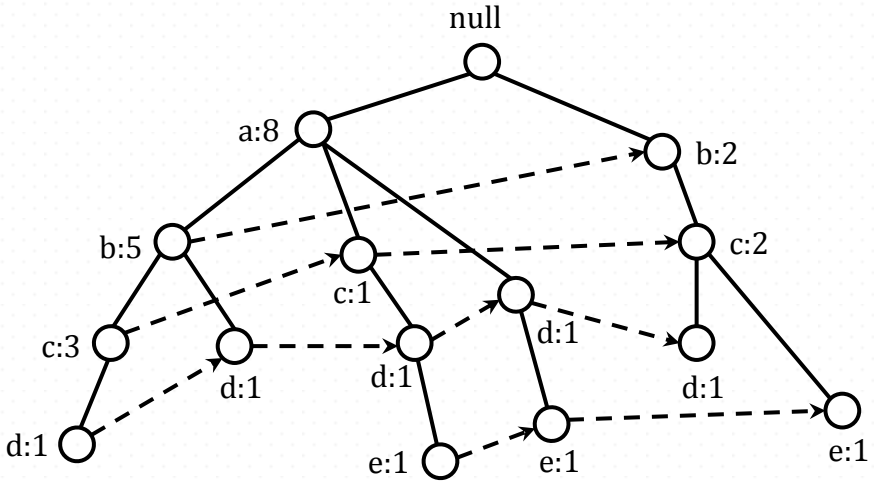


The Final FP-Tree
(Items sorted from lowest to highest support)

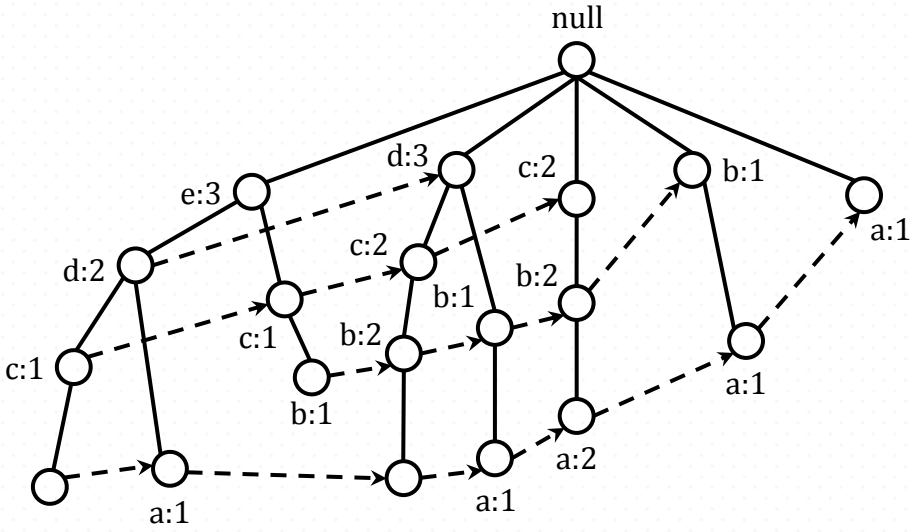
FP-Tree Comparison

The Final FP-Tree

Items sorted from highest to lowest support



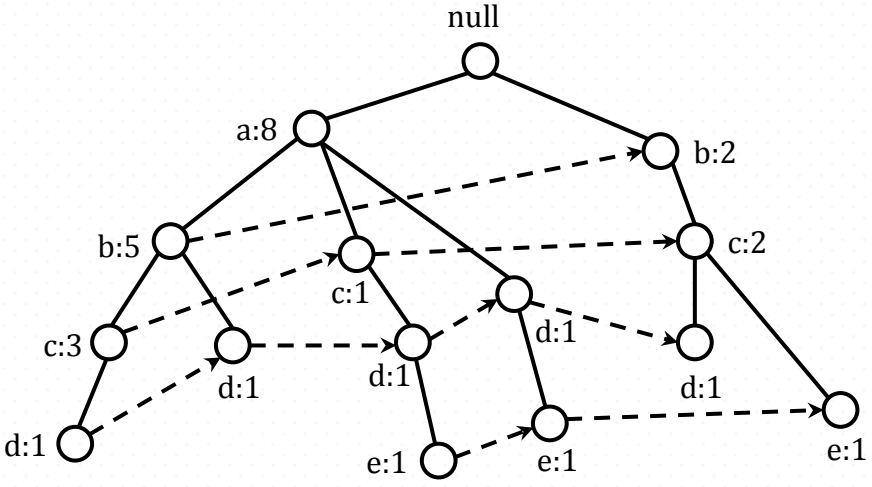
Items sorted from lowest to highest support

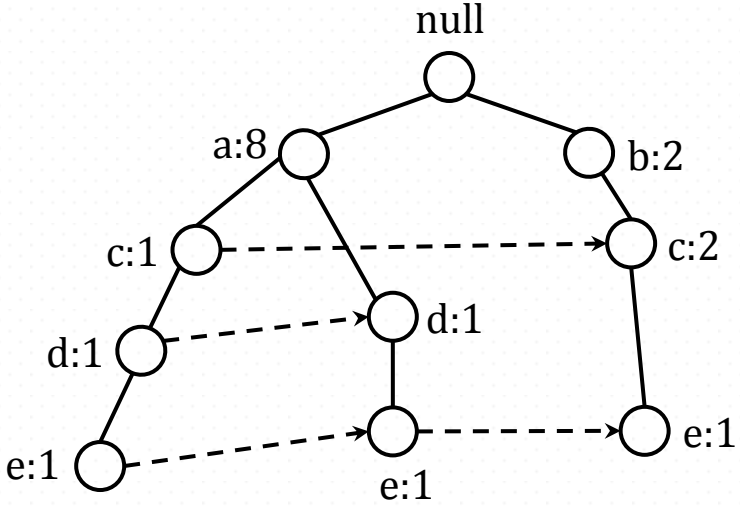


FP-Growth Frequent Itemset Generation

- FP-Growth is an algorithm that generates frequent itemsets from an FP-tree.
- The algorithm looks for frequent itemsets in a bottom-up fashion.

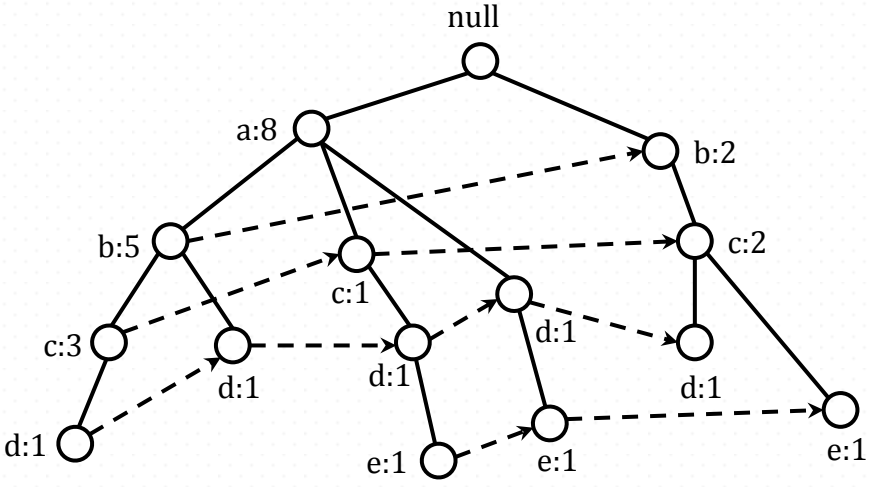
Example of FP-Tree Frequent Itemsets

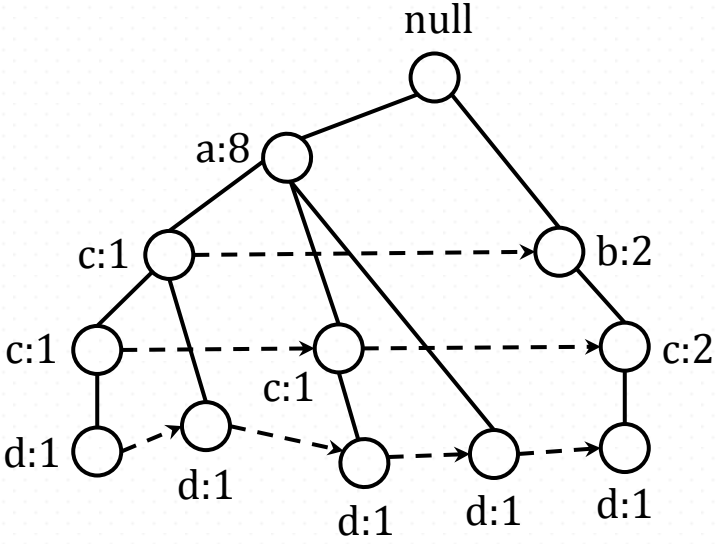




Paths containing node e

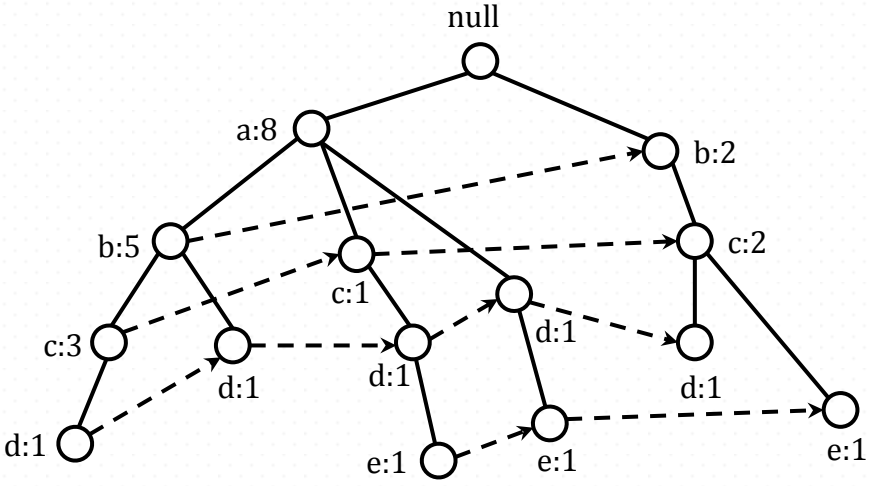
Example of FP-Tree Frequent Itemsets

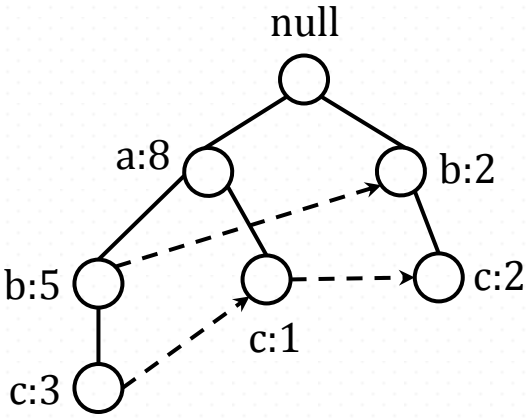




Paths containing node d

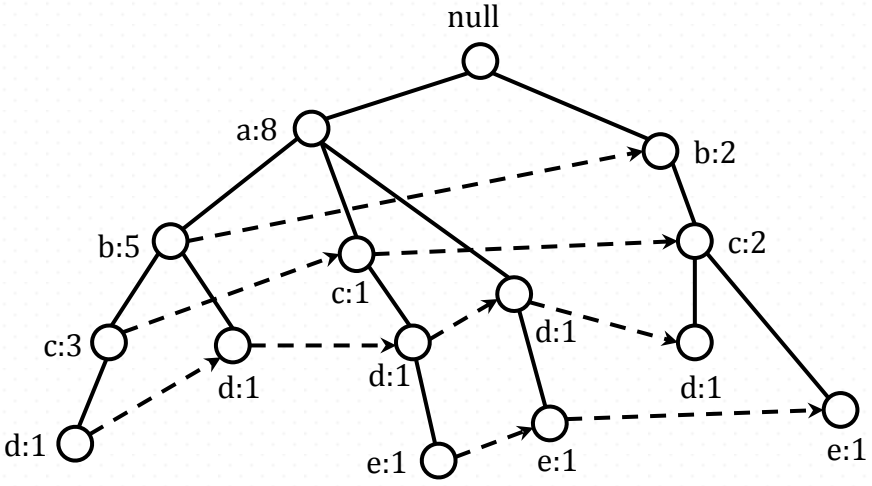
Example of FP-Tree Frequent Itemsets

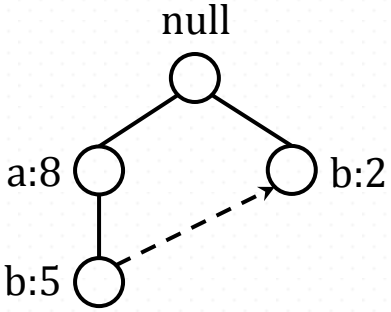




Paths containing node c

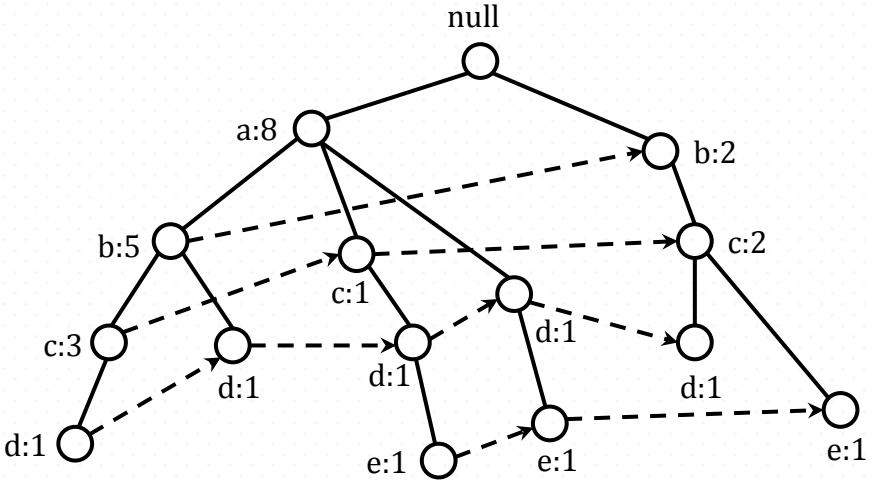
Example of FP-Tree Frequent Itemsets

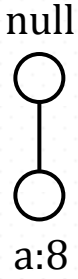




Paths containing node b

Example of FP-Tree Frequent Itemsets





Paths containing node a

FP-Growth to Find Frequent Itemsets

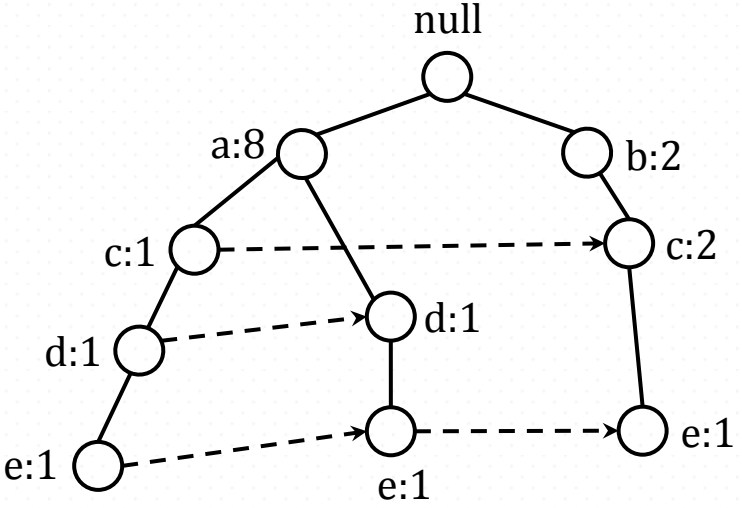
- Gather all the paths containing the relevant node. These paths are called **prefix paths**.
- From the prefix paths, the support count for the item is obtained by adding the support counts associated with the node.
- If the item is frequent, the algorithm has to solve the subproblems of finding frequent itemsets that derive from this item. To do this, it first converts the prefix paths into a **conditional FP-tree**.

Example of Conditional Trees

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 $minsup = 2$.

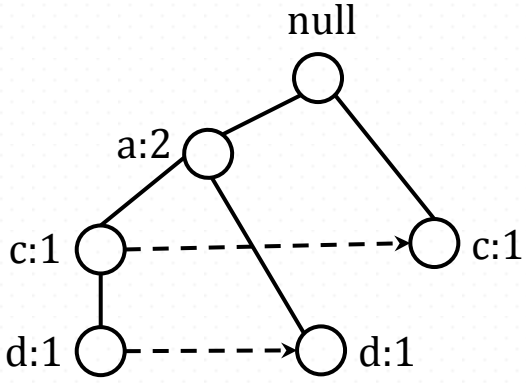
Prefix paths ending in e



Example of Conditional Trees

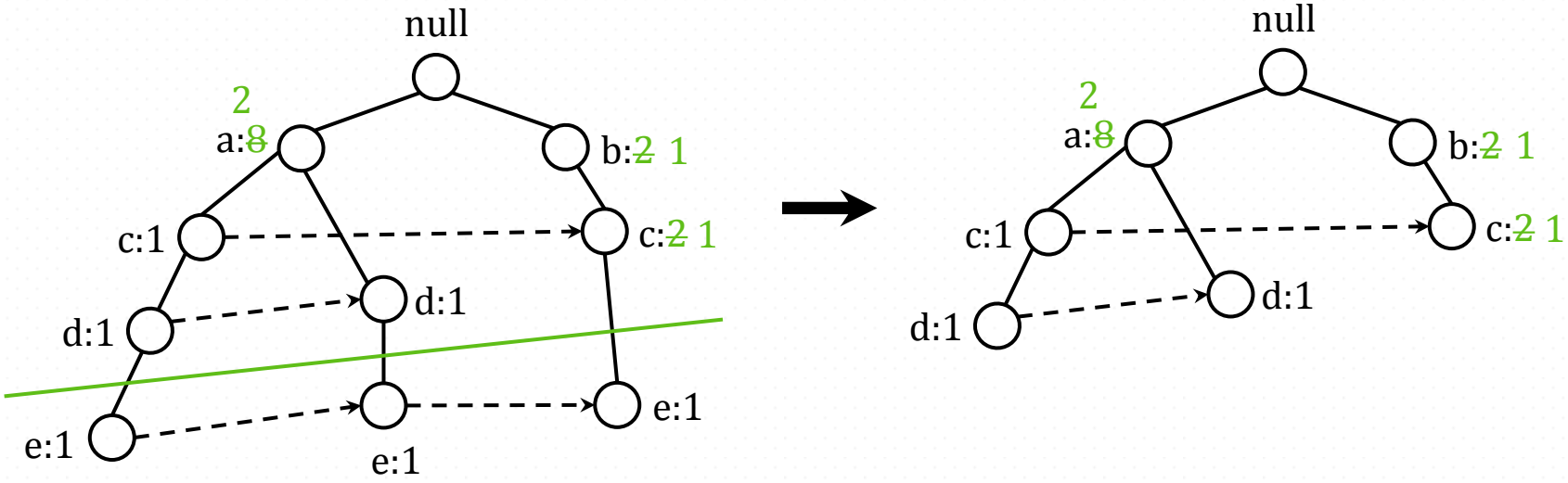
TID	Items
1	{a, b}
2	{b, c, d}
3	{a, c, d, ~~e~~}
4	{a, d, ~~e~~}
5	{a, b, c}
6	{a, b, c, d}
7	{a}
8	{a, b, c}
9	{a, b, d}
10	{b, c, ~~e~~}

Conditional FP-tree for e



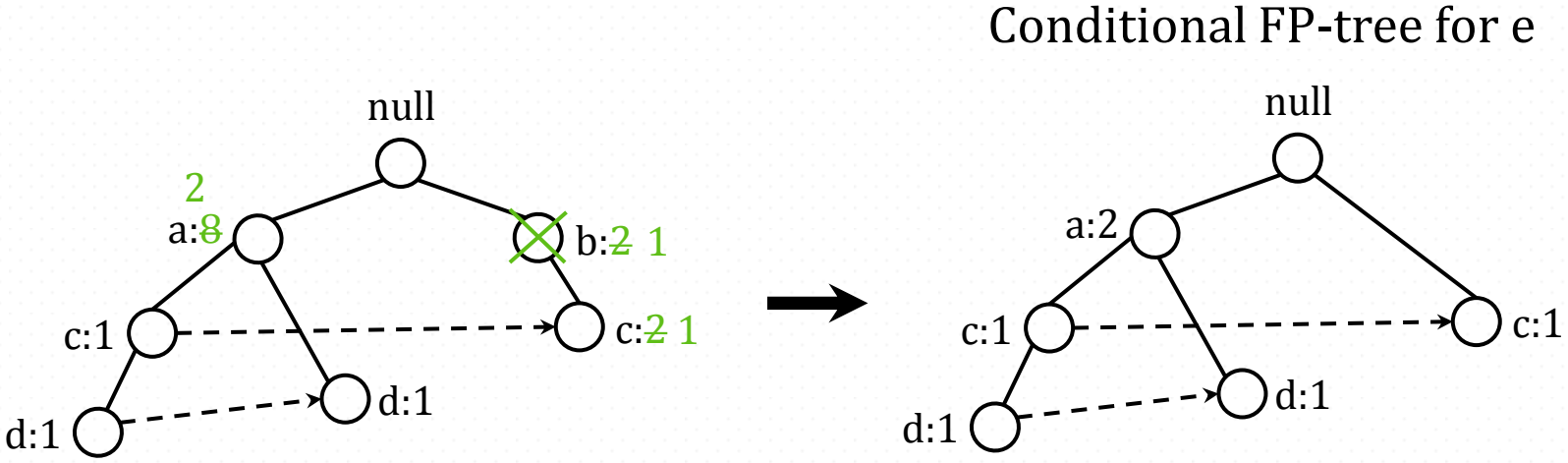
Example of Conditional Trees

Prefix paths ending in e



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

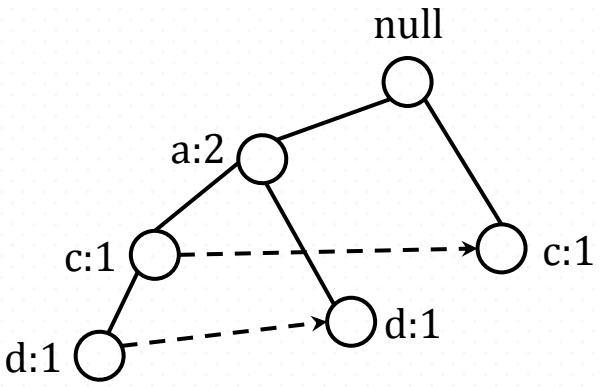
Example of Conditional Trees



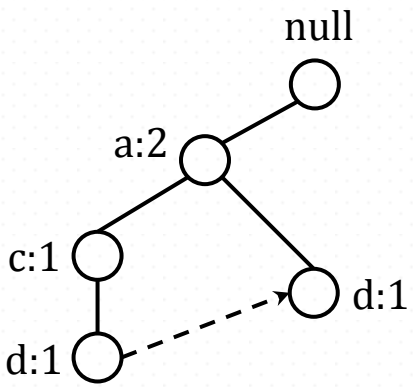
- Now remove infrequent items (nodes) from the prefix paths. In this example, *b* has a support of 1 (note this really means *be* has a support of 1). That is, there is only 1 transaction containing *b* and *e*.

Example of Conditional Trees

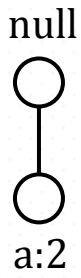
Conditional FP-tree for e



Prefix paths ending in de



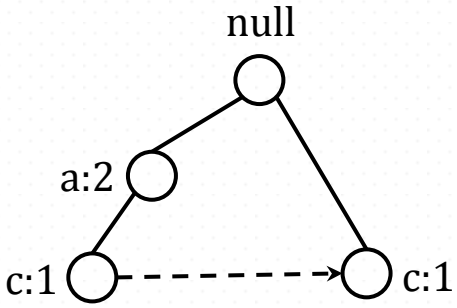
Conditional FP-tree for de



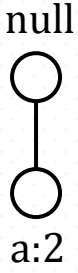
- Use the conditional FP-tree for *e* to find frequent itemsets ending in *de*, *ce*, and *ae* (*be* is not considered as *b* is not in the conditional FP-tree for *e*). For each item, find the prefix paths from the conditional tree for *e*, generate the conditional FP-tree, and continue recursively.

Example of Conditional Trees

Prefix paths ending in ce

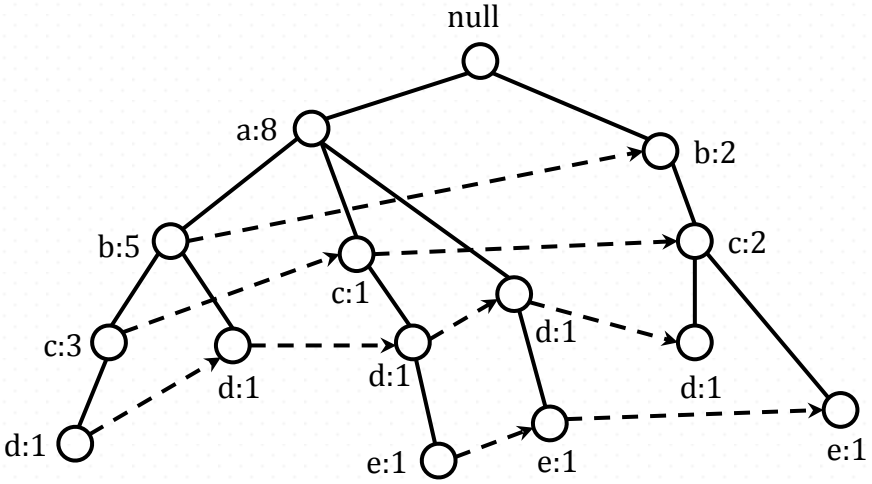


Prefix paths ending in ae



Example of FP-Tree Frequent Itemsets

- 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}

- Since every transaction is mapped onto a path in the FP-tree, we can derive the frequent itemsets ending with a particular item by examining only the paths containing that item's nodes.

So, we know how to generate association rules efficiently. Now, how do we decide if they are meaningful?

Evaluation of Association Patterns

- Sifting through the patterns to identify the most interesting ones is not a trivial task.
- Objectively, patterns that involve a set of mutually independent items or cover very few transactions are considered uninteresting.
- Subjectively, a pattern is uninteresting unless it reveals unexpected information about the data or provides useful knowledge.

Contingency Tables

2-way contingency table for variables A and B

	B	$\bar{B}$	
A	f_{11}	f_{10}	f_{1+}
$\bar{A}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

Beverage preferences among a group of 1000 people

	$Coffee$	$\overline{Coffee}$	
Tea	150	50	200
$\overline{Tea}$	650	150	800
	800	200	1000

Contingency Tables

Beverage preferences among
a group of 1000 people

	p	$\bar{p}$	
q	880	50	930
$\bar{q}$	50	20	70
	930	70	1000

	r	$\bar{r}$	
s	20	50	70
$\bar{s}$	50	880	930
	70	930	1000

Interest Factor

- High-confidence rules can sometimes be misleading because the confidence measure ignores the support of the itemset appearing in the rule consequent.
  - One way to address this problem is by applying a metric known as **lift**:

$$Lift = \frac{c(A \rightarrow B)}{s(B)}$$

  which computes the ratio between the rule's confidence and the support of the itemset in the rule consequent.

Interest Factor

- For binary variables, lift is equivalent to another objective measure called interest factor, which is defined as follows:

$$I(A,B) = \frac{s(A,B)}{s(A) \times s(B)} = \frac{N f_{11}}{f_{1+} + f_{+1}}$$

- Interest factor compares the frequency of a pattern against a baseline frequency computed under the statistical independence assumption.

Interest Factor

- The baseline frequency for a pair of mutually independent variables is

$$\frac{f_{11}}{N} = \frac{f_{1+}}{N} \times \frac{f_{+1}}{N}, \text{ or equivalently, } f_{11} = \frac{f_{1+} + f_{+1}}{N}$$

- This equation follows from the standard approach of using simply fractions as estimates of probabilities. The fraction f_{11}/N is an estimate for the joint probability $P(A, B)$, while f_{1+}/N and f_{+1}/N are the estimates for $P(A)$ and $P(B)$, respectively.

Interest Factor

We can interpret the measure as follows:

$$I(A,B)\begin{cases} = 1, & \text{if } A \text{ and } B \text{ are independent;} \\ > 1, & \text{if } A \text{ and } B \text{ are positively correlated;} \\ < 1, & \text{if } A \text{ and } B \text{ are negatively correlated.} \end{cases}$$

Correlation Analysis

- Correlation analysis is a statistical-based technique for analyzing relationships between a pair of variables.
- For continuous variables, correlation is defined using Pearson's correlation coefficient. For binary variables, correlation can be measured using the ϕ–coefficient, which is defined as

$$\phi = \frac{f_{11}f_{00} - f_{01}f_{10}}{\sqrt{f_{1+}f_{+1}f_{0+}f_{+0}}}$$

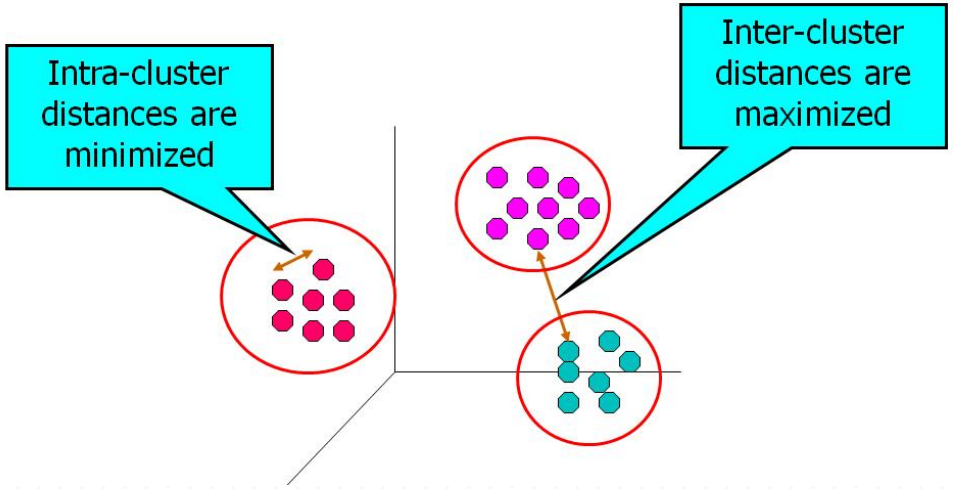
And now...

Lets see some FP-Growthing!

Clustering & Association

Clustering - Overview

- What is cluster analysis?
  - Grouping data objects based only on information found in the data describing these objects and their relationships
  - Maximize the similarity within objects in the same group
  - Maximize the difference between objects in different groups



Source: http://apandre.wordpress.com/visible-data/cluster-analysis/

Clustering - Overview

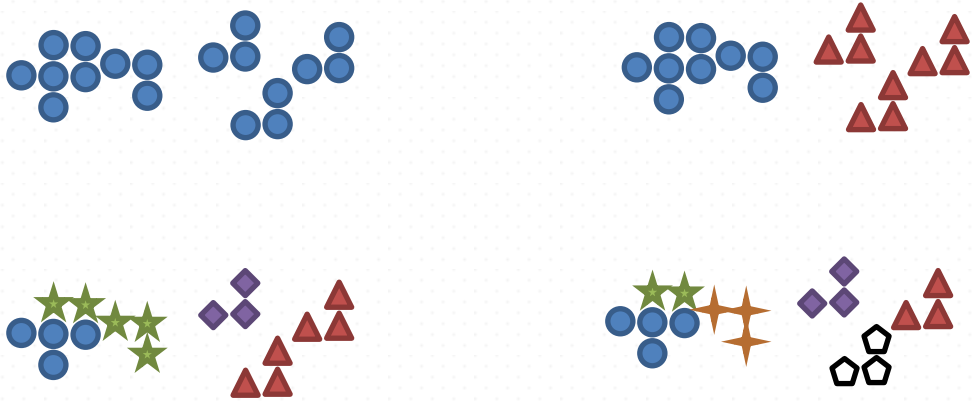
- What is cluster analysis?
  - **Similarity:**
    - Numerical measure of “alikeness” of two instances
    - Greater if more alike
    - Can be normalized to [0,1]
  - **Dissimilarity:**
    - How different two instances are
    - Lower for “alike” instances
  - These are usually expressed in terms of a distance function
  - Different weights can be assigned to different features
  - Hard to define what “similar enough” means. The answer is typically highly subjective.

Clustering - Overview

- What is cluster analysis?
  - Different from classification (supervised learning) in that the labels for each instance are derived only from the data
  - For that reason, cluster analysis is referred to as *unsupervised classification*

Clustering - Overview

- Not well defined at times:

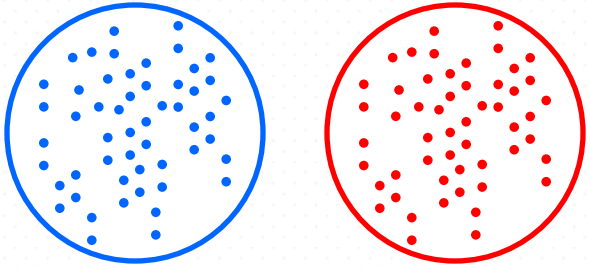


- How do we really know how many clusters should exist in the above example?

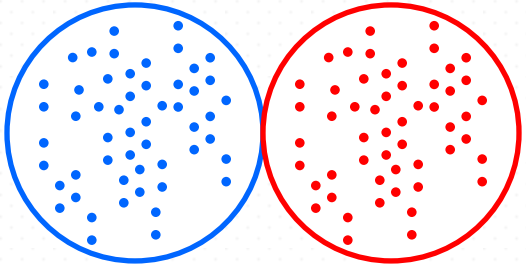
Clustering - Overview

- Different types of clusterings:
  - Hierarchical versus Partitional
  - Exclusive versus Overlapping versus Fuzzy
  - Complete versus Partial
- Different types of clusters:
  - Well separated
  - Prototype-Based
  - Density-Based
  - Shared-Property

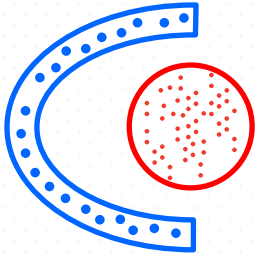
Clustering - Overview



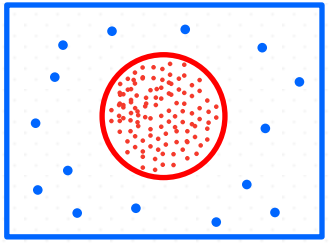
(a) Well separated clusters



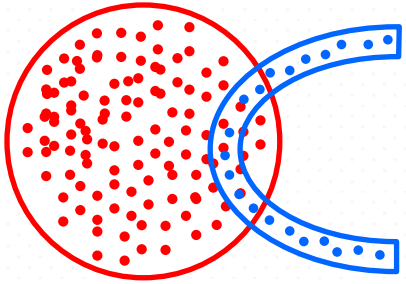
(b) Center-based clusters



(c) Contiguity-based clusters



(d) Density-based clusters



(e) Conceptual clusters

Clustering

1. K-means clustering
2. Hierarchical clustering
3. Evaluating clusters

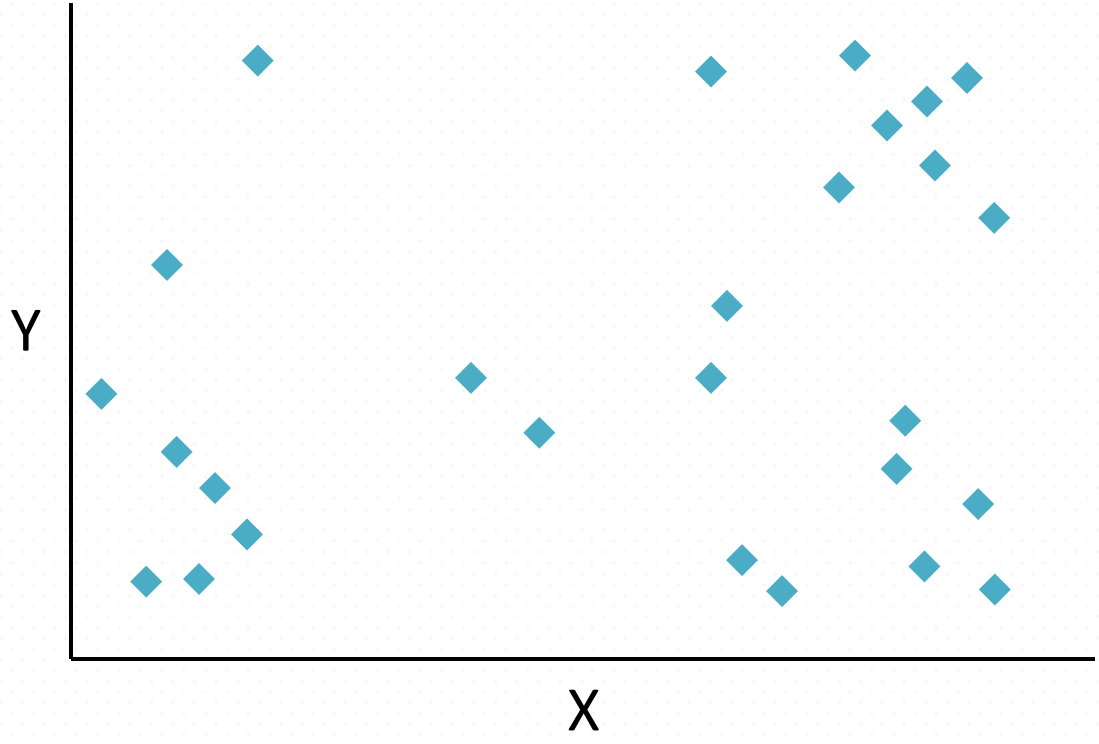
K-means clustering

- Works with numeric data only!
- Algorithm:

1. Pick a number *k* of random cluster centers
2. Assign every item to its nearest cluster center using a distance metric
3. Move each cluster center to the mean of its assigned items
4. Repeat 2-3 until convergence (change in cluster assignment less than a threshold)

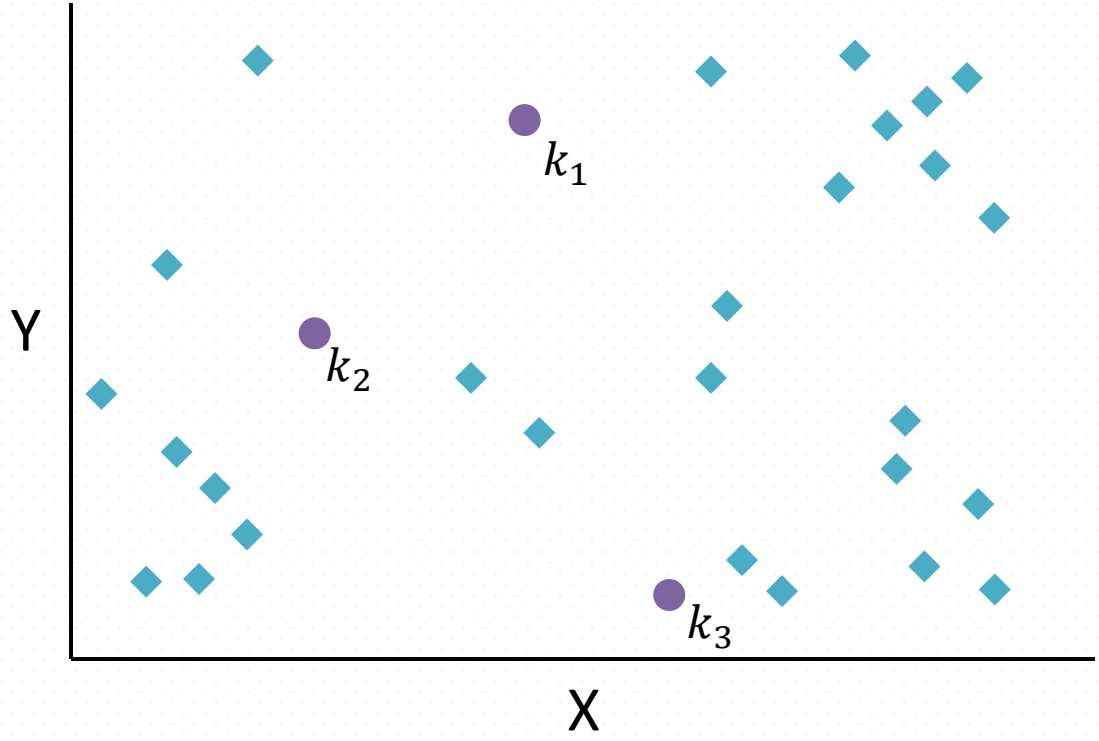
K-means clustering – Example

Suppose we wish to cluster these items



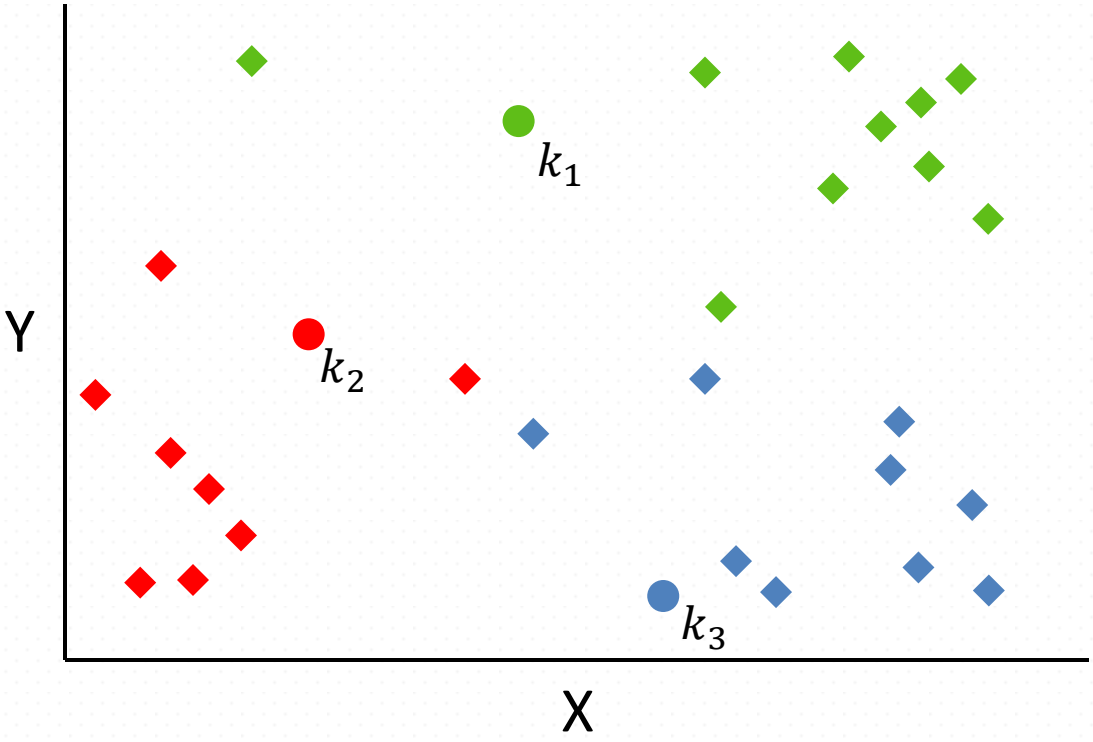
K-means clustering – Example

Pick 3 initial cluster centers at random



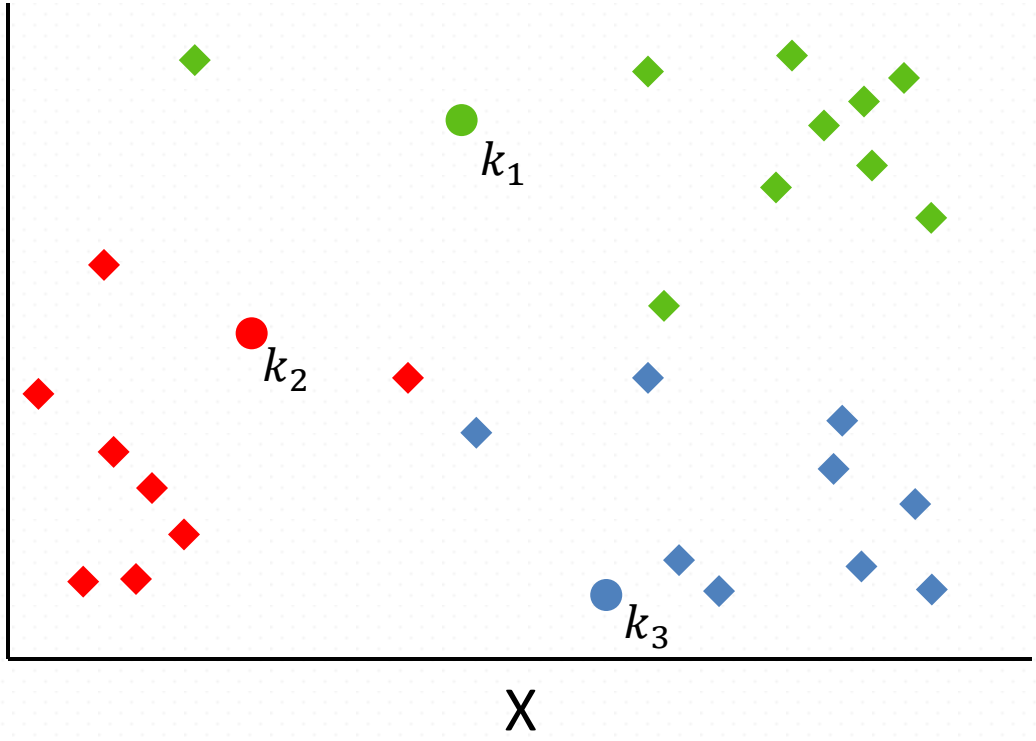
K-means clustering – Example

Assign each instance to the closest cluster center



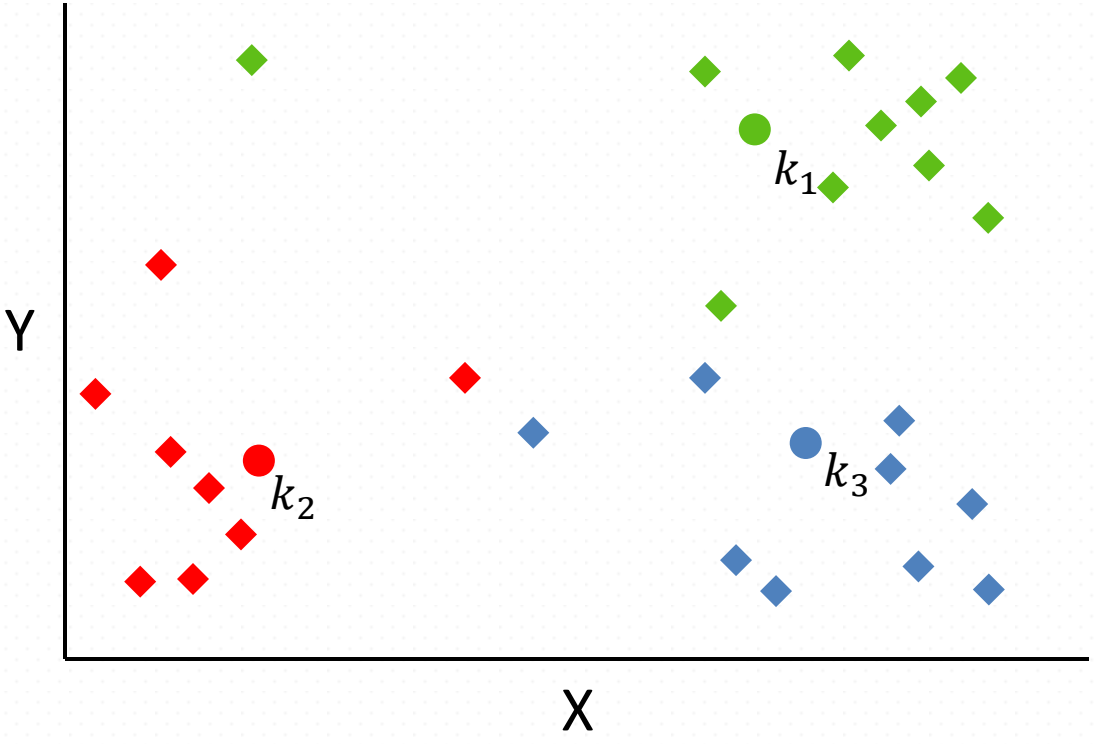
K-means clustering – Example

Move each cluster center to the mean of each cluster



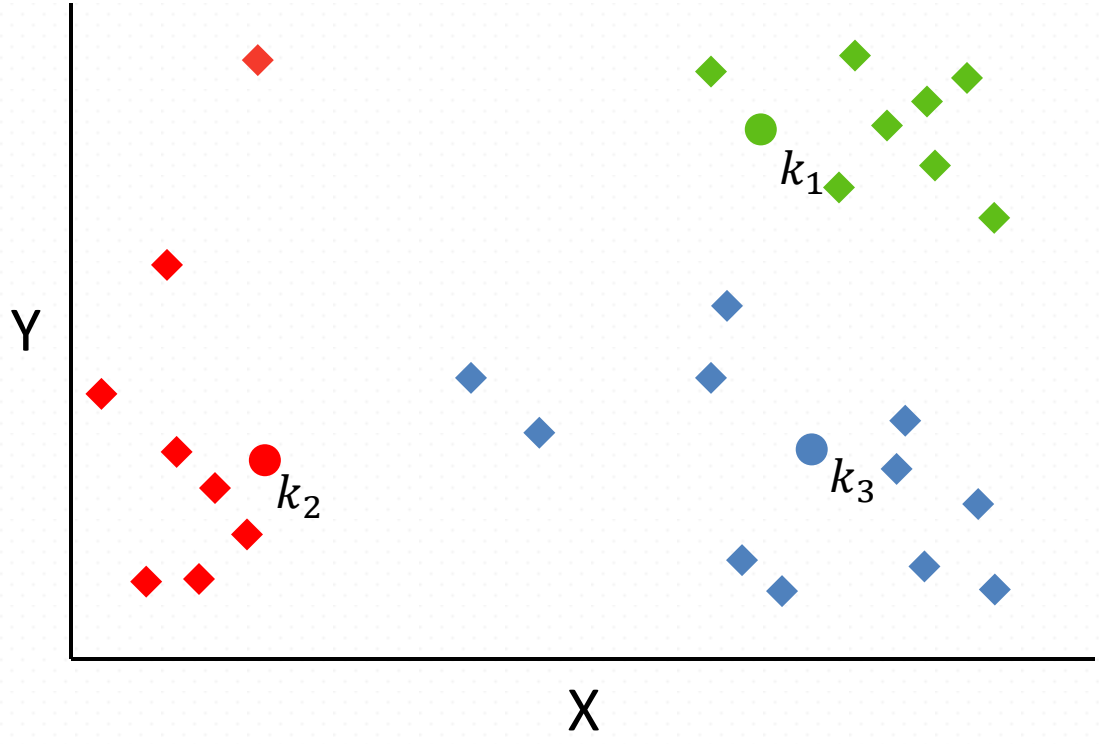
K-means clustering – Example

Reassign points closest to a different new cluster center



K-means clustering – Example

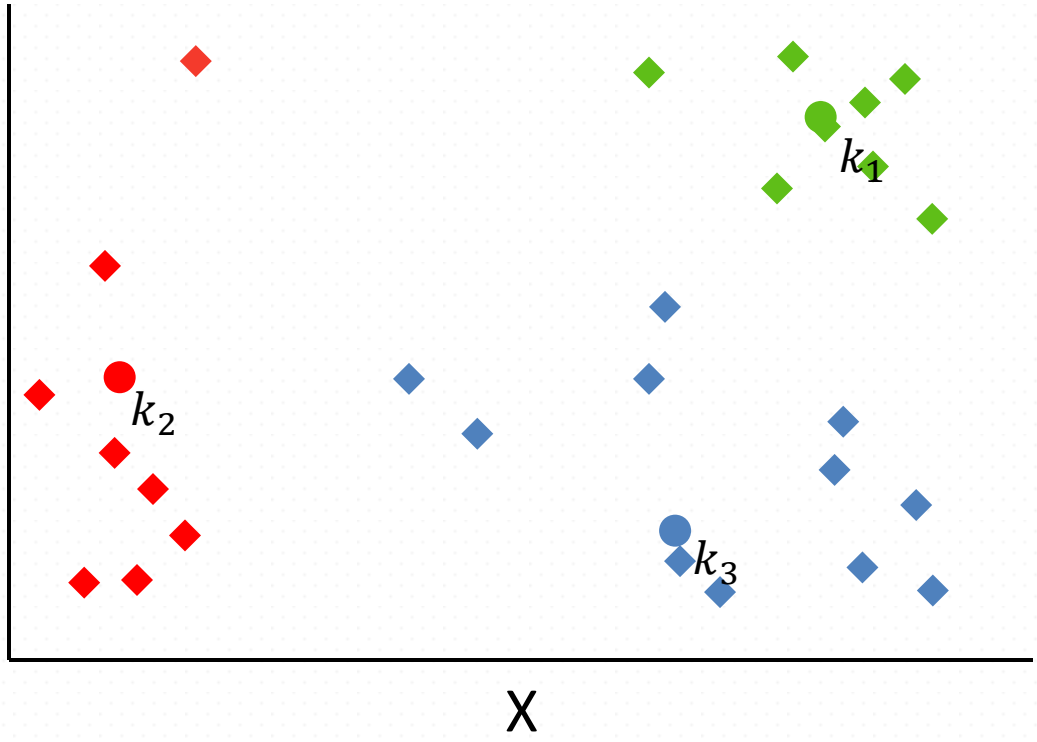
Recompute cluster means



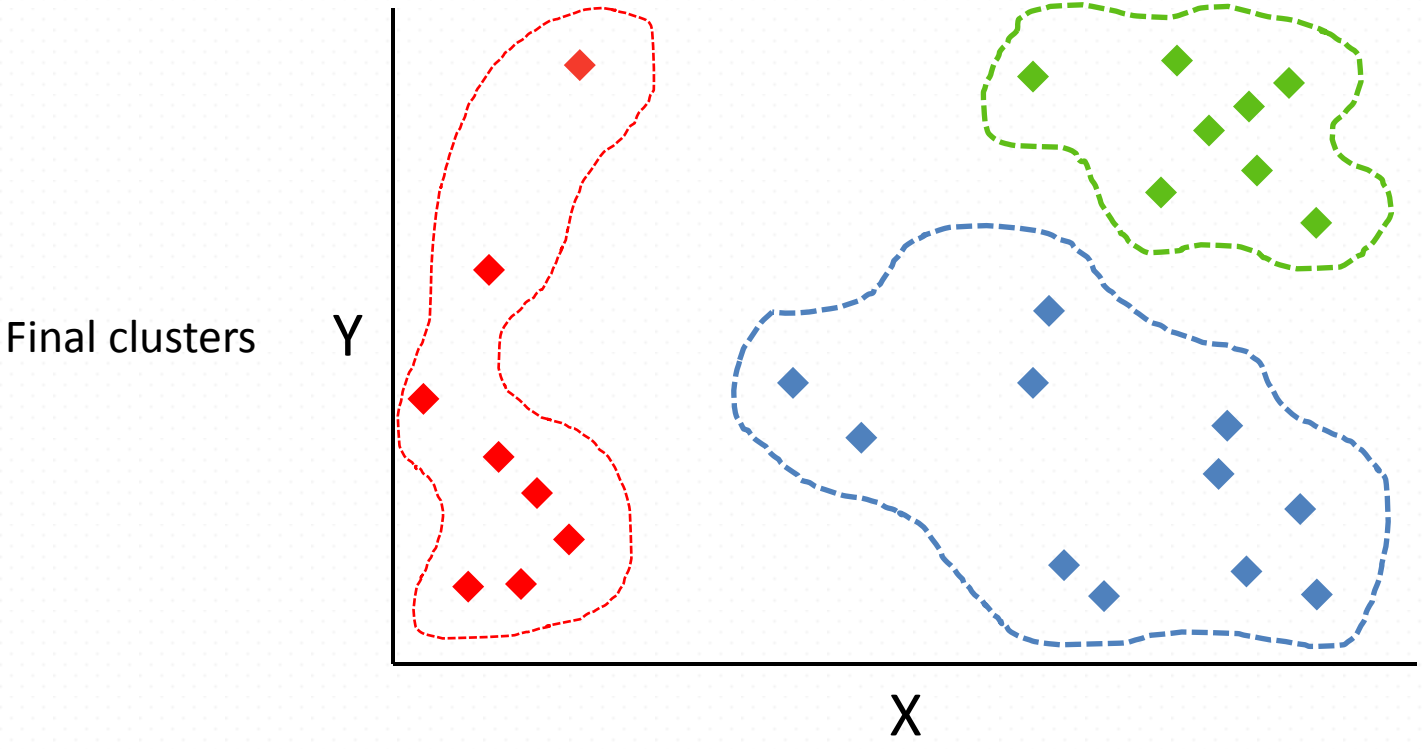
K-means clustering – Example

Reassign labels

No change. Done!



K-means clustering – Example



K-means – Evaluating clusters

- If Euclidian distance is utilized as the proximity measure, the quality of clusters can be evaluated by the sum of squared error (SSE)
  - Calculate the distance between each point and its closest centroid (error)
  - Sum the square of all errors

$$\text{SSE} = \sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i, x)^2$$

- It is important to choose values of k that minimize SSE

K-means – Discussion

- Results can vary significantly depending on initial choice of seeds
- How do you tackle this?
- How do you choose k?

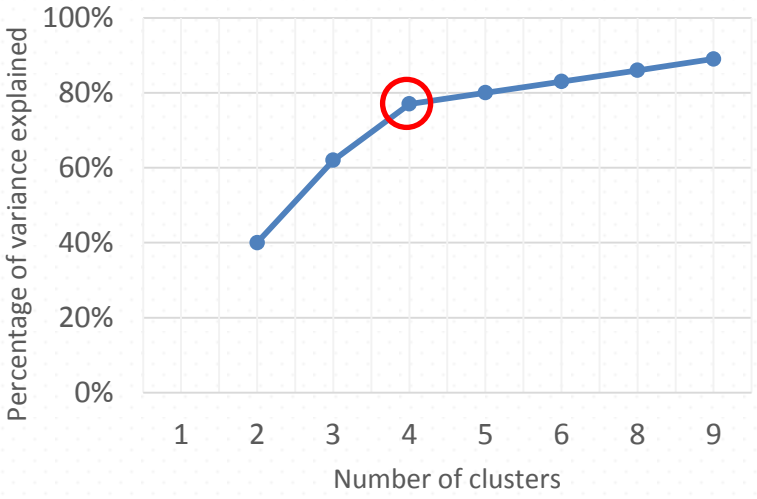
How do you choose k?

- Determining the number of clusters in a data set
  - The choice of k is often ambiguous and dependent on scale and distribution of your dataset
  - However, some generic methods for doing that do exist

- Rule of thumb:
  - $k \approx \sqrt{n/2}$, where n is the number of instances
  - This is a good starting point, but not a very reliable approach

How do you choose *k*?

- The *Elbow Method*:
  - Choose a number of clusters that covers most of the variance



How do you choose *k*?

- Other methods:
  - Information Criterion Approach
  - Silhouette method
  - Jump method
  - Gap statistic

K-means variations

- K-medoids – instead of mean, use medians of each cluster
  - Mean of 1, 3, 5, 7, 1009 is **205**
  - Median of 1, 3, 5, 7, 205 is **5**
  - Advantage: not affected by extreme values
- For large datasets, use sampling

K-means pros and cons

- **Pros:** very efficient (even if multiple runs are performed), can be used for a large variety of data types.
- **Cons:** Not suitable for all types of data, susceptible to initialization problems and outliers, restricted to data in which there is a notion of a center

And now...

Lets see some k-meaning!

Hierarchical vs. Partitional Clustering

- A distinction among different types of clusterings is whether the set of clusters is nested or unnested.
- A **partitional clustering** a simply a division of the set of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset.
- A **hierarchical clustering** is a set of nested clusters that are organized as a tree.

Why of Hierarchical Clustering?

1. It does not assume a particular value of k, as needed by k-means clustering.
2. The generated tree may correspond to a meaningful taxonomy.
3. Only a distance or "proximity" matrix is needed to compute the hierarchical clustering.

Hierarchical Clustering

Two types of algorithms:

1. **Agglomerative ("Bottom-up")**
   Start with the points as individual clusters and, at each step, merge the closest pair of clusters.

2. **Divisive ("Top-down")**
   Start with one, all-inclusive cluster and, at each step, split a cluster until only singleton clusters of individual points remain.

Basic Agglomerative Clustering

Basic agglomerative hierarchical clustering algorithm.

1: Compute the proximity matrix, if necessary.
2: **repeat**
3: Merge the closest two clusters.
4: Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
5: **until** Only one cluster remains.

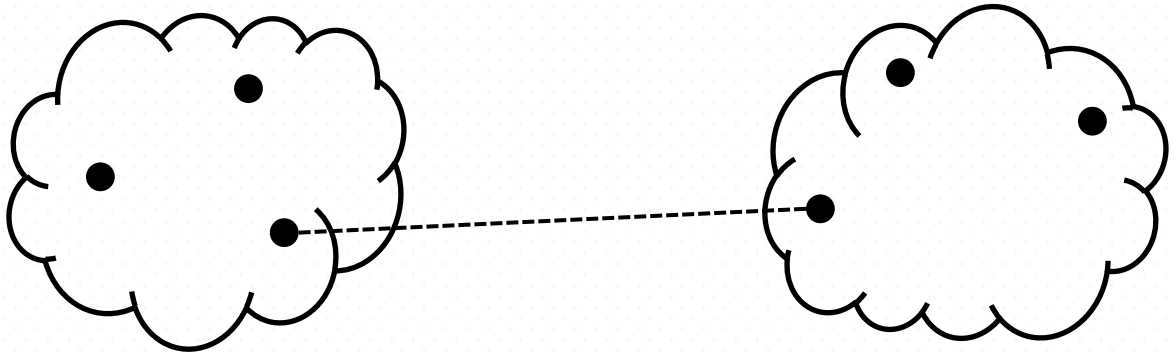
Defining Proximity Between Clusters

- The key operation of basic agglomerative clustering is the computation of the proximity between two clusters.
- The definition of cluster proximity differentiates the various agglomerative hierarchical techniques.
- MAX (single link), MIN (complete link), and group average are **graph-based proximities**.
- Ward's method is a **prototype-based proximity**.

MIN (Single Link) Proximity

Defines cluster proximity as the shortest distance between two points, x and y, that are in different clusters, A and B:

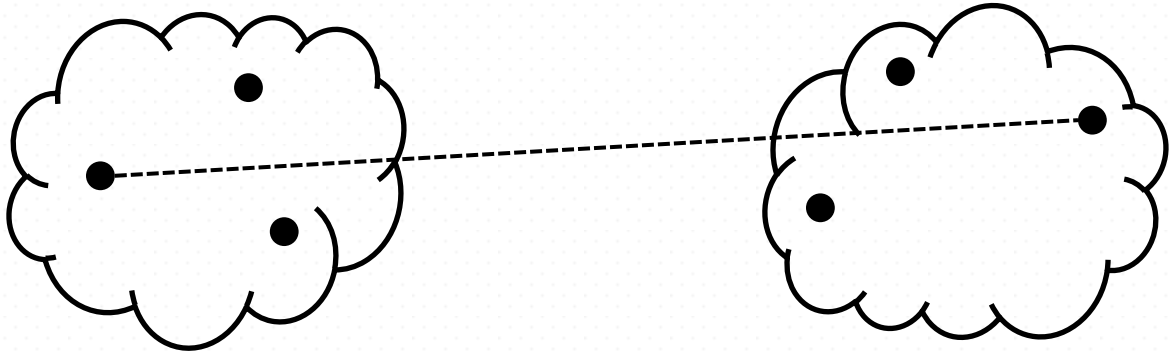
$$d(A, B) = \min_{x \in A, y \in B} d(x - y)$$



MAX (Complete Link) Proximity

Defines cluster proximity as the furthest distance between two points, x and y, that are in different clusters, A and B:

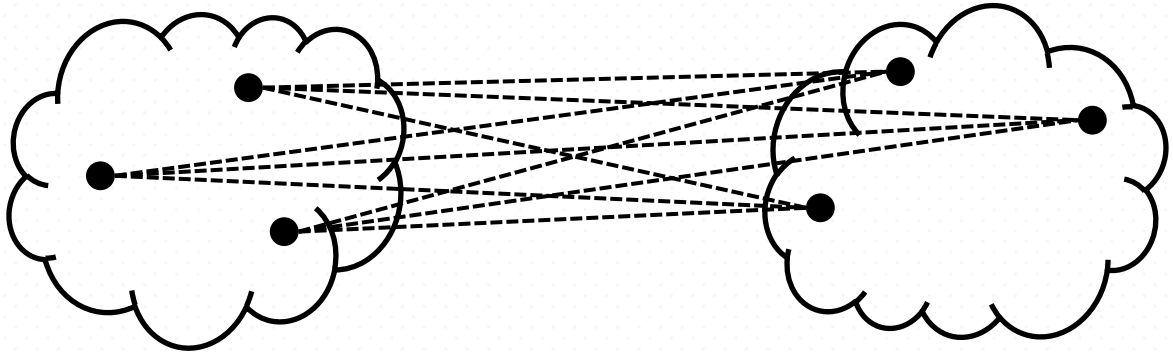
$$d(A, B) = \max_{x \in A, y \in B} d(x - y)$$



Group Average Proximity

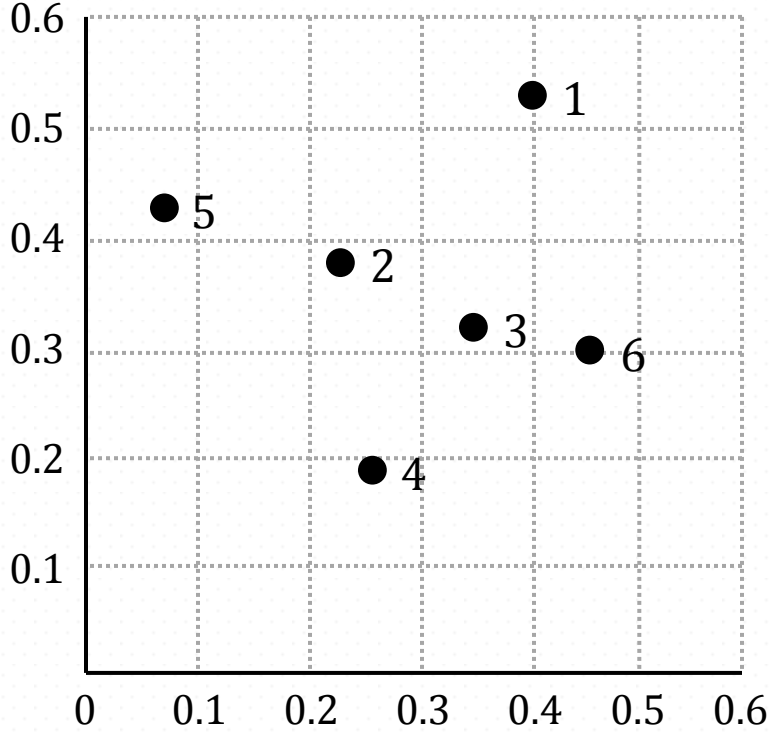
Defines cluster proximity as the average distance between two points, x and y, that are in different clusters, A and B (number of points in cluster j is n_j):

$$d(A,B) = \sum_{x \in A, y \in B} d(x-y)/n_A n_B$$



Example Data for Clustering

Set of 6 Two-Dimensional Points

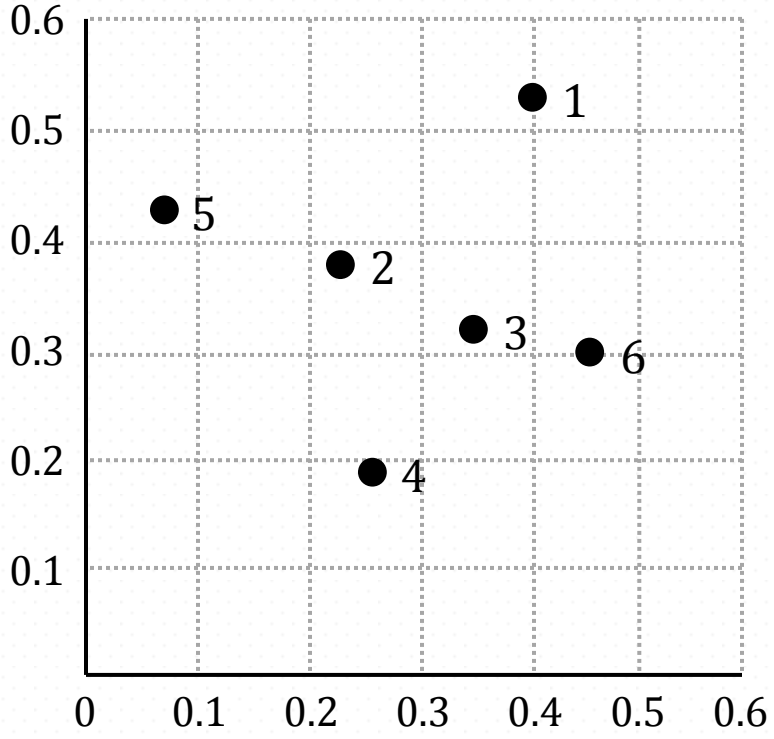


xy Coordinates of 6 Points

Point	x Coordinate	y Coordinate
p1	0.40	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.30

Example Data for Clustering

Set of 6 Two-Dimensional Points

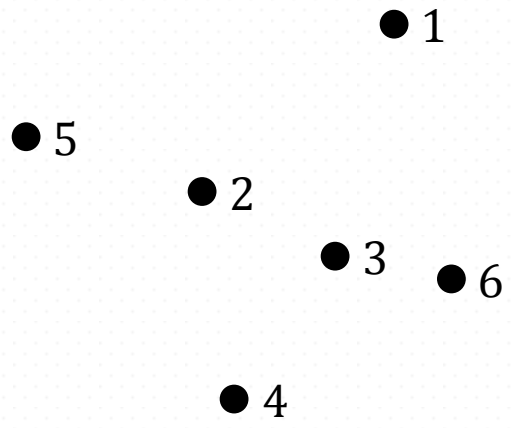


Euclidean Distance Matrix for 6 Points

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Example of Single Link Clustering

Nested Cluster Diagram

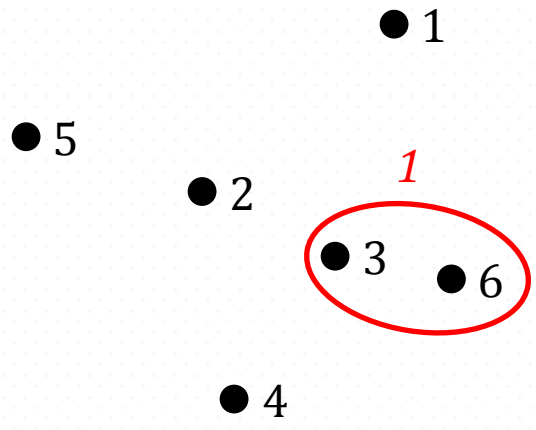


Single Link Distance Matrix

	1	2	3	4	5	6
1	0	0.24	0.22	0.37	0.34	0.23
2		0	0.15	0.20	0.14	0.25
3			0	0.15	0.28	0.11
4				0	0.29	0.22
5					0	0.39
6						0

Example of Single Link Clustering

Nested Cluster Diagram



Single Link Distance Matrix

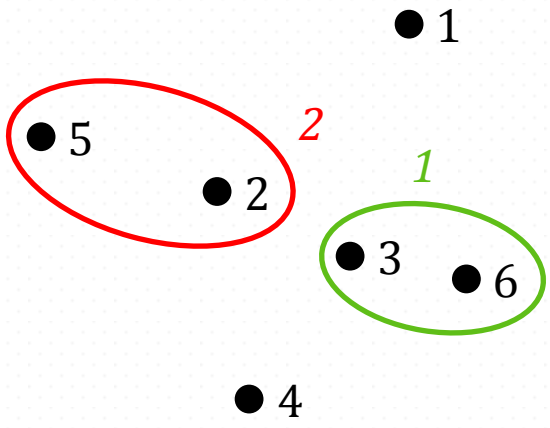
	1	2	3	4	5	6
1	0	0.24	*0.22*	0.37	0.34	0.23
2		0	*0.15*	0.20	0.14	0.25
3			0	*0.15*	*0.28*	0.11
4				0	0.29	0.22
5					0	0.39
6						0

1

Points 3 and 6 have the smallest single link proximity distance. Merge these points into one cluster and update the distances to this new cluster. For example, the distance from point 1 to this cluster is 0.22 (the distance to point 3).

Example of Single Link Clustering

Nested Cluster Diagram



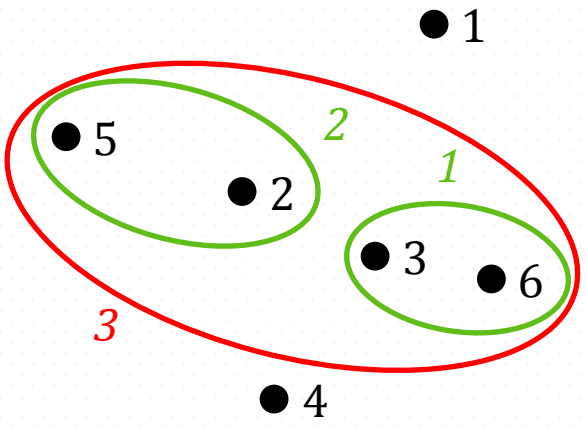
Single Link Distance Matrix

	1	2	4	5	3,6
1	0	*0.24*	0.37	0.34	0.22
2		0	*0.20*	0.14	0.15
4			0	0.29	0.15
5				0	0.28
3,6					0

2 Points 2 and5 have the smallest single link proximity distance. Merge these points into one cluster and update the distances to this new cluster.

Example of Single Link Clustering

Nested Cluster Diagram



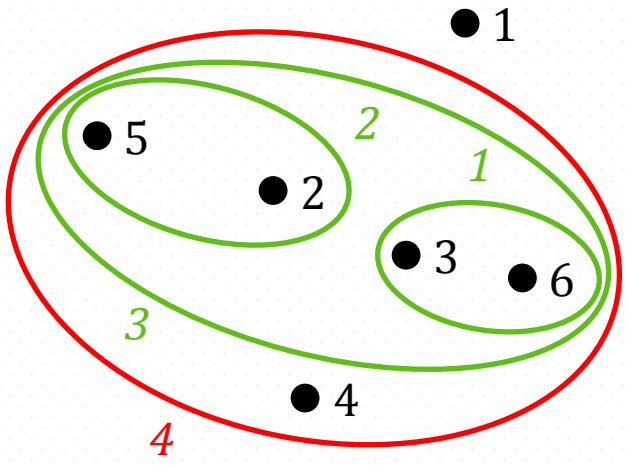
Single Link Distance Matrix

	1	4	2,5	3,6
1	0	0.37	0.24	*0.22*
4		0	0.20	*0.15*
2,5			0	0.15
3,6				0

3 And iterate…

Example of Single Link Clustering

Nested Cluster Diagram



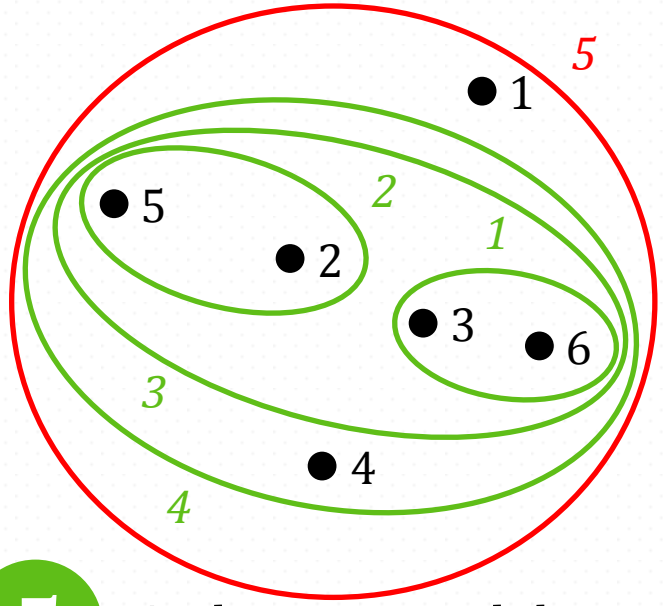
Single Link Distance Matrix

	1	4	2,5,3,6
1	0	0.37	*0.22*
4		0	0.15
2,5,3,6			0

4 And iterate…

Example of Single Link Clustering

Nested Cluster Diagram



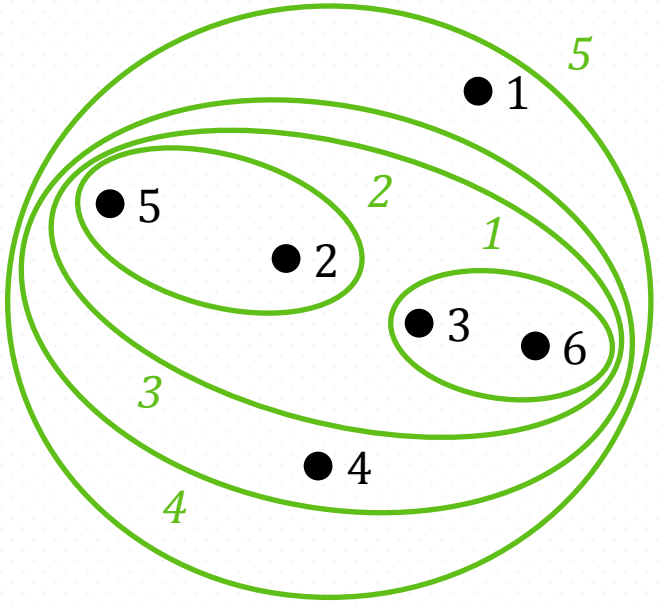
Single Link Distance Matrix

	1	4,2,5,3,6
1	0	0.22
2,5,3,6		0

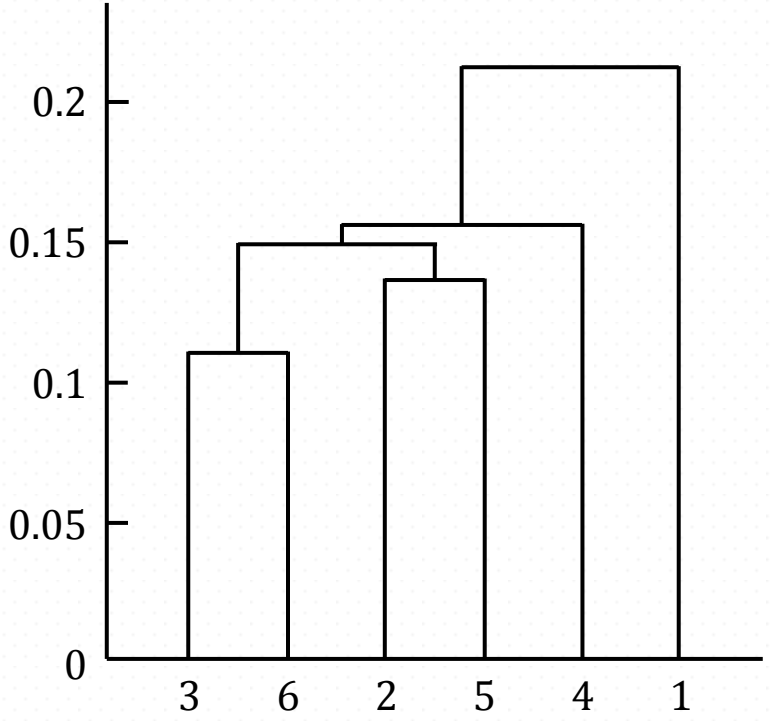
5 And iterate until there is one all-inclusive cluster.

Example of Single Link Clustering

Nested Cluster Diagram

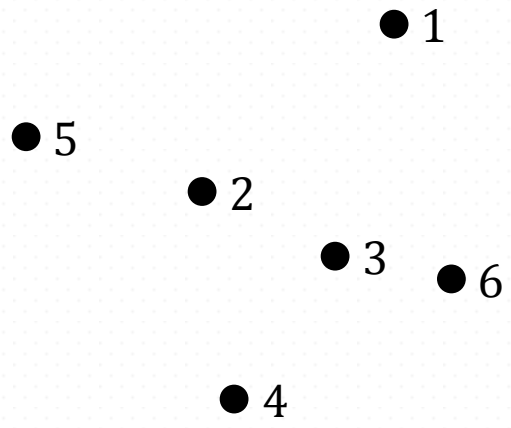


Hierarchical Tree Diagram



Example of Complete Link Clustering

Nested Cluster Diagram

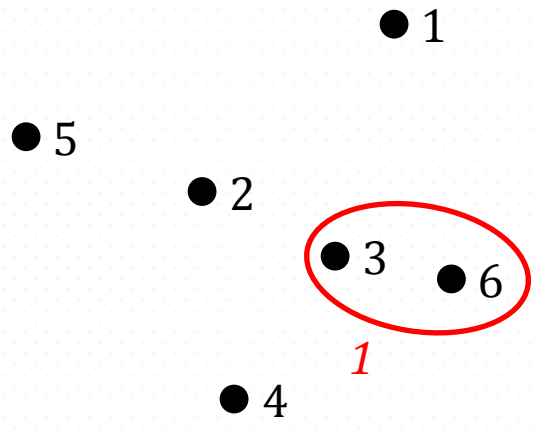


Single Link Distance Matrix

	1	2	3	4	5	6
1	0	0.24	0.22	0.37	0.34	0.23
2		0	0.15	0.20	0.14	0.25
3			0	0.15	0.28	0.11
4				0	0.29	0.22
5					0	0.39
6						0

Example of Complete Link Clustering

Nested Cluster Diagram



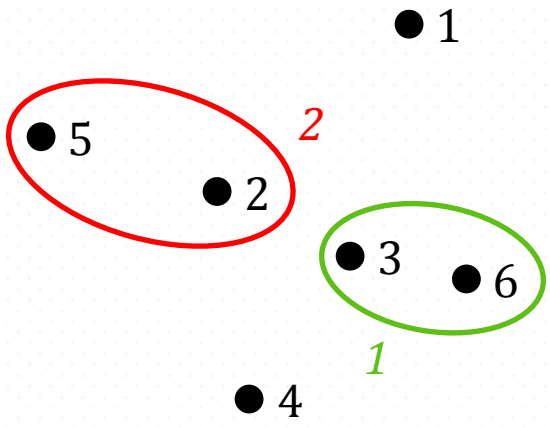
Complete Link Distance Matrix

	1	2	3	4	5	6
1	0	0.24	0.22	0.37	0.34	*0.23*
2		0	0.15	0.20	0.14	*0.25*
3			0	0.15	0.28	0.11
4				0	0.29	*0.22*
5					0	*0.39*
6						0

1 Points 3 and 6 have the smallest complete link proximity distance. Merge these points into one cluster and update the distances to this new cluster. For example, the distance from point 1 to this cluster is 0.23 (the distance to point 6).

Example of Complete Link Clustering

Nested Cluster Diagram



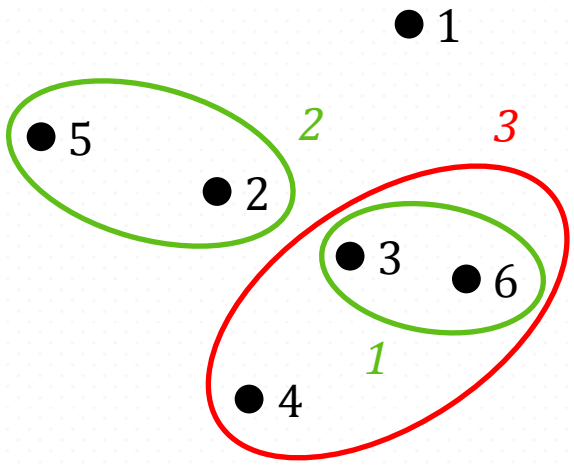
Complete Link Distance Matrix

	1	2	4	5	3,6
1	0	0.24	0.37	*0.34*	0.23
2		0	0.20	0.14	0.25
4			0	*0.29*	0.22
5				0	0.39
3,6					0

2 Points 2 and 5 have the smallest complete link proximity distance. Merge these points into one cluster and update the distances to this new cluster.

Example of Complete Link Clustering

Nested Cluster Diagram



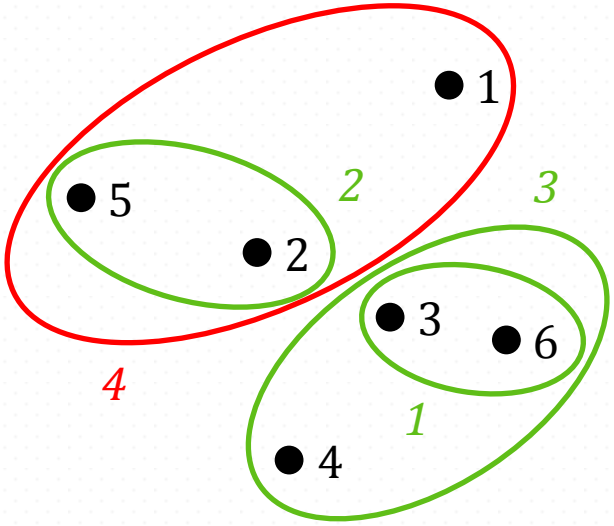
Complete Link Distance Matrix

	1	4	2,5	3,6
1	0	*0.37*	0.34	0.23
4		0	0.29	0.22
2,5			0	*0.39*
3,6				0

3 And iterate…

Example of Complete Link Clustering

Nested Cluster Diagram



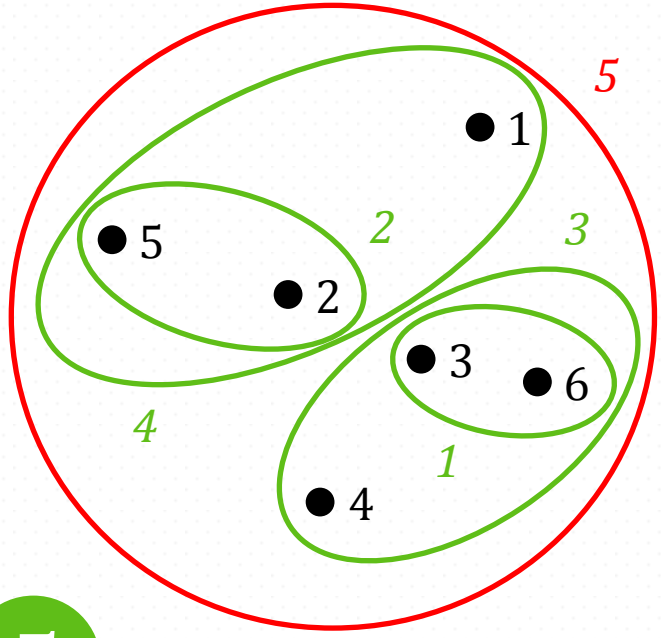
Complete Link Distance Matrix

	1	2,5	4,3,6
1	0	0.34	0.37
2,5		0	*0.39*
4,3,6			0

4 And iterate…

Example of Complete Link Clustering

Nested Cluster Diagram



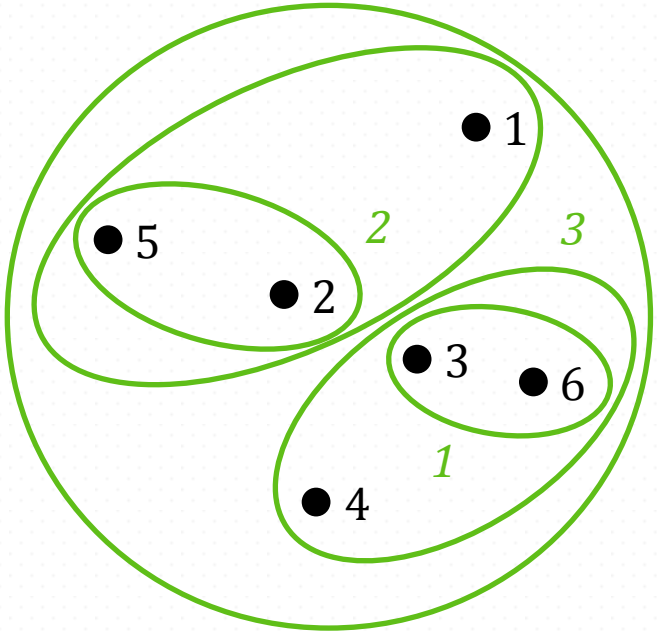
Complete Link Distance Matrix

	1,2,5	4,3,6
1,2,5	0	0.39
4,3,6		0

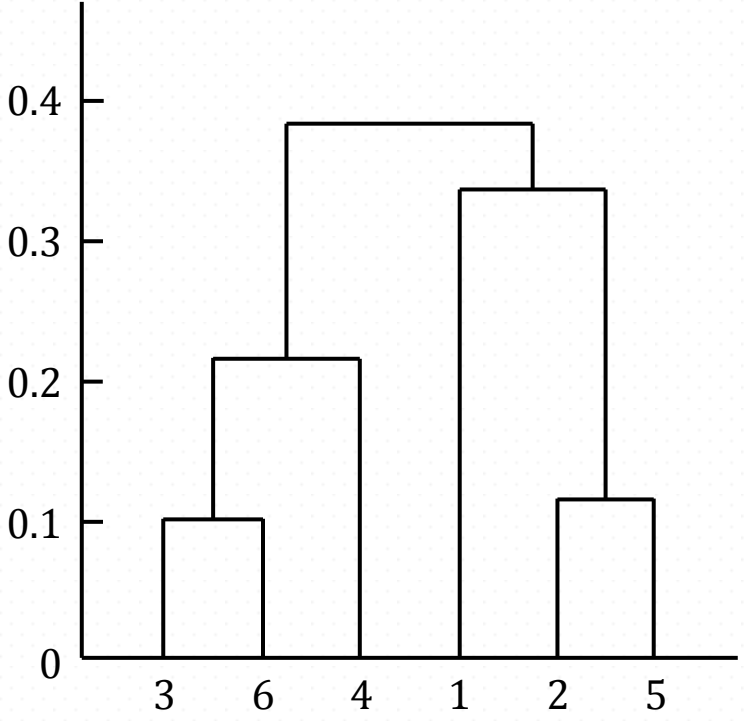
5 And iterate until there is one all-inclusive cluster.

Example of Complete Link Clustering

Nested Cluster Diagram

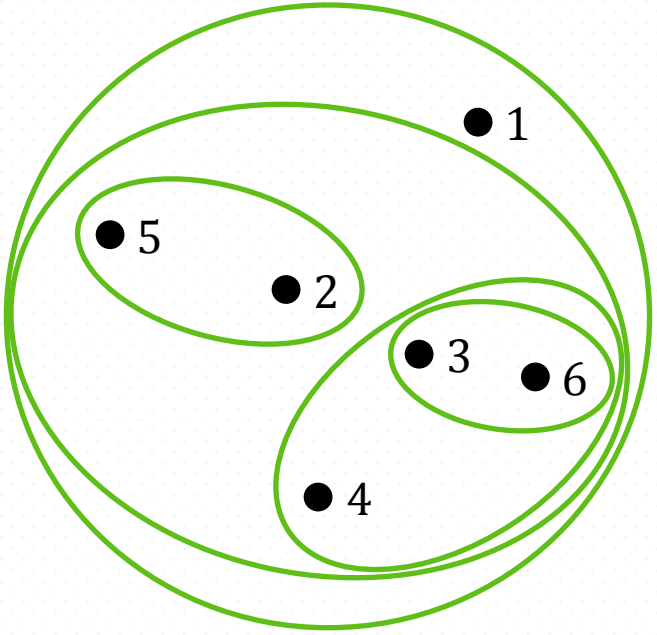


Hierarchical Tree Diagram

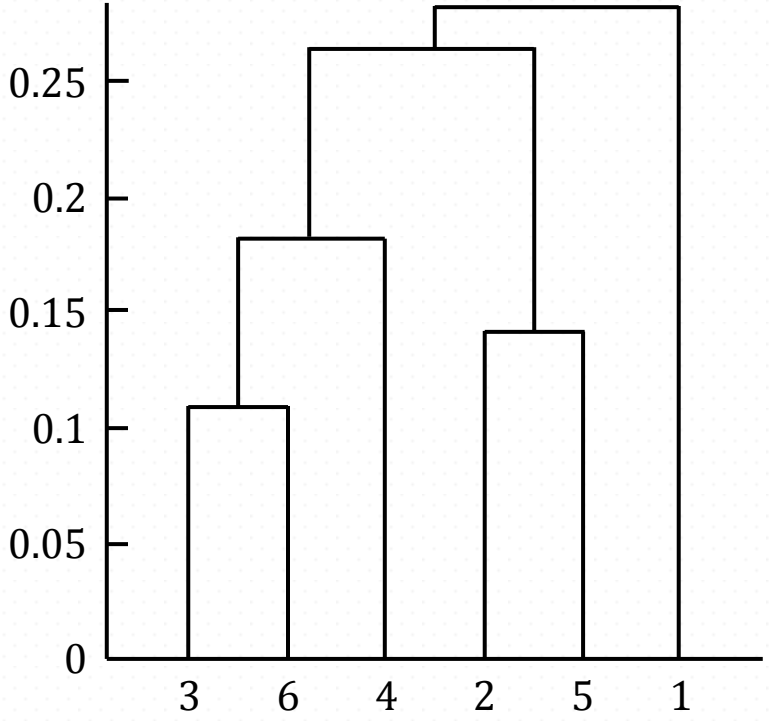


Example of Group Average Clustering

Nested Cluster Diagram



Hierarchical Tree Diagram



Discussion of Proximity Methods

- **Single link** is "chain-like" and good at handling non-elliptical shapes, but is sensitive to outliers.
- **Complete link** is less susceptible to noise and outliers, but can break large clusters and favors globular shapes.
- **Group average** is an intermediate approach between the single and complete link approaches.

Ward's Method

Assumes that a cluster is represented by its centroid, and measures the proximity between two clusters in terms of the increase in sum of the squared error (SSE) that results from merging the two clusters:

$$d(A, B) = SSE_{A \cup B} - SSE_A - SSE_B$$

where A and B are clusters.

Note that for hierarchical clustering, the SSE starts at 0.

Discussion of Hierarchical Clustering

- Useful if the underlying application has a taxonomy.
- Agglomerative hierarchical clustering algorithms are expensive in terms of their computational and storage requirements.
- Merges are final and cannot be undone at a later time, preventing global optimization and causing trouble for noisy, high-dimensional data.

And now...

Lets see some hierarchical clustering!

Cluster Similarity

Similarity is most often measured with the help of a *distance function*. The smaller the distance, the more similar the data objects (points).

A function $d: M \times M \rightarrow \mathbb{R}$ is a **distance** on M if it satisfies for all $\vec{x}, \vec{y}, \vec{z} \in M$ (where M is an arbitrary non-empty set and $\mathbb{R}$ is the set of real numbers):

$$(i)\ \ d(\vec{x}, \vec{y}) = 0 \ \leftrightarrow \ \vec{x} = \vec{y} \qquad \text{(coincidence axiom)}$$
$$(ii)\ \ d(\vec{x}, \vec{y}) = d(\vec{y}, \vec{x}) \qquad \text{(symmetry)}$$
$$(iii)\ \ d(\vec{x}, \vec{z}) \leq d(\vec{x}, \vec{y}) + d(\vec{y}, \vec{z}) \qquad \text{(triangle inequality)}$$

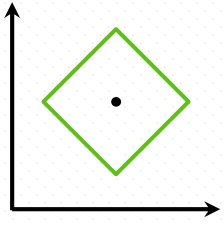
Distance Functions: Minkowski

The Minkowski family of distance functions is defined as:

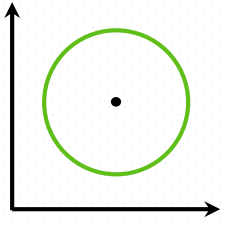
$$d_k(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{n} (x_i - y_i)^k \right)^{\frac{1}{k}}$$

where n is the number of features and k is a parameter.

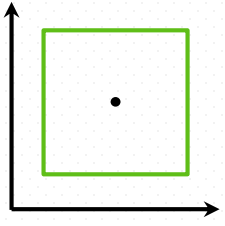
$k = 1$: Manhattan distance (or city block distance)



$k = 2$: Euclidean distance



$k \to \infty$: maximum distance (i. e. $d_\infty(\vec{x}, \vec{y}) = max_{i=1}^{n} |x_i - y_i|$)



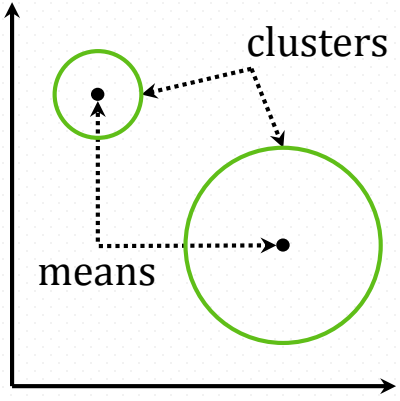
Distance Functions: Mahalanobis

Mahalanobis distance is defined as:

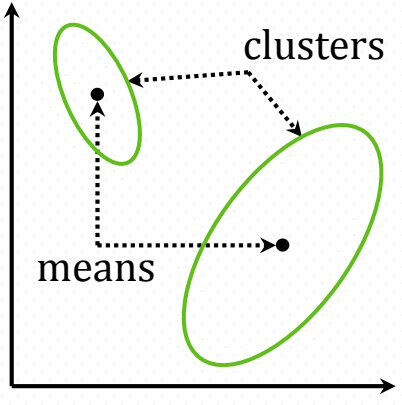
$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y})}$$

where Σ is the covariance matrix of $\vec{x}$ and $\vec{y}$.

Euclidean Distance



Mahalanobis Distance



Exclusive vs. Overlapping vs. Fuzzy

- An **exclusive clustering** assigns each object to a single cluster.
- An **overlapping clustering** or non-exclusive clustering is used to reflect the fact that an object can *simultaneously* belong to more than one group (class).
- A **fuzzy cluster** assigns objects to a cluster with a membership weight that is between 0 (absolutely doesn't belong) and 1 (absolutely belongs).

Clustering Using Mixture Models

- Often, it is convenient and effective to assume that data has been generate by a statistical process.
- To describe the data, we can find the statistical model (described by a distribution and a set of parameters for that distribution) that best fits the data.
- **Mixture models** use statistical distributions to model the data, each distribution corresponding to a cluster and the parameters of each distribution describing the corresponding cluster.

Mixture Models

- Mixture models view the data as a set of observations from a mixture of different probability distributions.
- The Idea:
  - Given several distributions, usually of the same type, but with different parameters, randomly select one of these distributions and generate an object from it.
  - Repeat the process m times, where m is the number of objects.

The EM Algorithm

EM algorithm

1: Select an initial set of model parameters.

2: **repeat**

3: **Expectation Step** For each object, calculate the probability that each object belongs to each distribution, i.e., calculate $prob(distribution\ j|x_i, \Theta)$.

4: **Maximization Step** Given the probabilities from the expectation step, find the new estimates of the parameters that maximize the expected likelihood.

5: **until** The parameters do not change (or are below a threshold).

Mixture Models: A Mathy Description

Assume K distributions and m objects, $X = \{x_1, \dots, x_m\}$. Let the j^{th} distribution have parameters θ_j, and let Θ be the set of all parameters, i.e. $\Theta = \{\theta_1, \dots, \theta_k\}$. The probability that the j^{th} distribution is chosen to generate an object is given by the weight w_j, $1 \leq j \leq K$, where these weights (probabilities) sum to one, i.e., $\sum_{j=1}^{K} w_j = 1$. Then the probability of an object x is given by:

$$prob(x_i|\Theta) = \sum_{j=1}^{K} w_j \underbrace{p_j(x_i|\theta_j)}_{\uparrow}$$

The probability of object x if it comes from the j^{th} distribution.

Mixture Models: A Mathy Description

If the objects are generated in an independent manner, then the probability of the entire set of objects is just the product of the probabilities of each individual x_i

$$prob(X|\Theta) = \prod_{i=1}^{m} \underbrace{prob(x_i|\Theta)}_{\uparrow} = \prod_{i=1}^{m} \sum_{j=1}^{K} w_j \underbrace{p_j(x_i|\theta_j)}_{\uparrow}$$

The probability of i^{th} object.

The probability of i^{th} object if it comes from the j^{th} distribution.

Gaussian Mixture: A Mathy Description

Consider a mixture model in terms of Gaussian distributions. The probability density function for a one-dimensional Gaussian distribution at a point x is:

$$prob(x_i|\Theta) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

The parameters of the Gaussian distribution are given by $\theta = (\mu, \sigma)$, where μ is the mean of the distribution and σ is the standard deviation.

MLE: A Mathy Description

Consider a set of m points that are generated from a one-dimensional Gaussian distribution:

$$prob(X|\Theta) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

$$\log prob(X|\Theta) = -\sum_{i=1}^{m} \frac{(x_i-\mu)^2}{2\sigma^2} - 0.5m \log 2\pi - m \log \sigma$$

One approach to estimate μ and σ is to choose the values of the parameters for which the data is most probable (most likely), known as the maximum likelihood principle (MLE).

MLE: A Mathy Description

The concept is called the maximum likelihood principle because, given a set of data, the probability of the data, regarded as a function for the parameters, is called a likelihood function.

$$likelihood(\Theta|X) = (\Theta|X) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

For practical reasons, the log likelihood is more commonly used. Note that the parameter values that maximize the log likelihood also maximize the likelihood.

$$\log\ likelihood(\Theta|X) = \ell(\Theta|X) = -\sum_{i=1}^{m} \frac{(x_i-\mu)^2}{2\sigma^2} - 0.5m\log 2\pi - m\log\sigma$$

Example of EM: Initialization

Begin the EM algorithm by making initial guesses for μ_1 and μ_2, say, $\mu_1 = -2$ and $\mu_2 = 3$. Thus, the initial parameters, $\theta = (\mu, \sigma)$, for the two distributions are, respectively, $\theta_1 = (-2,2)$ and $\theta_1 = (3,2)$. The set of parameters for the entire mixture model is $\Theta = \{\theta_1, \theta_2\}$.

Example of EM: Expectation Step

For the expectation step of EM, we want to compute the probability that a point came from a particular distribution. These values can be expressed by a straightforward application of Bayes' rule:

$$prob(distribution\ j|x_i,\theta) = \frac{0.5\ prob(x_i|\theta_j)}{0.5\ prob(x_i|\theta_1) + 0.5\ prob(x_i|\theta_2)}$$

where 0.5 is the probability (weight) of each distribution and j is 1 or 2.

Example of EM: Maximization Step

After computing the cluster membership probabilities for all the points, we compute new estimates for μ_1 and μ_2 in the maximization step of the EM algorithm.

$$\mu_j = \sum_{i=1}^{m} x_i \frac{prob(distribution\ j|x_i, \Theta)}{\sum_{i=1}^{m} prob(distribution\ j|x_i, \Theta)}$$

where j is a particular distribution and m is the number of objects.

Mixture Model Clustering Using EM

- The EM algorithm can be slow.
- Does not work well when clusters contain only a few data points or if the data points are nearly co-linear.
- The problem in estimating the number of clusters or choosing the exact form of the model to use.
- More general than *k*-means because they can use distributions of various types.
- Easy to characterize the clusters produced.

Clustering Evaluation

1. Determining the **clustering tendency** of a set of data, i.e., distinguishing whether non-random structure actually exists in the data.
2. Determining the correct number of clusters.
3. Evaluating how well the results of a cluster analysis fit the data *without* reference to external information.
4. Comparing the results of a cluster analysis to externally known results, such as externally provided class labels.
5. Comparing two sets of clusters to determine which is better.

Types of Cluster Evaluation

Unsupervised: Measures the goodness of a clustering structure without respect to external information.

Supervised: Measures the extent to which the clustering structure discovered by a clustering algorithm matches some external structure.

Relative: Compares different clusterings or clusters with a supervised or unsupervised evaluation measure that is used for the purpose of comparison.

Overall Cluster Validity

In general, we can consider expressing overall cluster validity for a set of K clusters as a weighted sum of the validity of individual clusters.

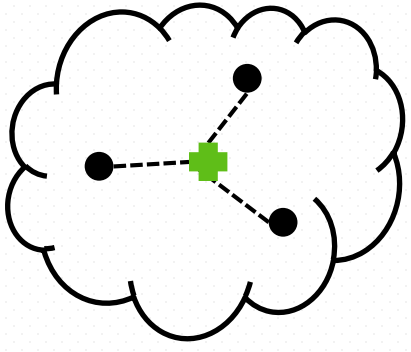
$$overall\ validity = \sum_{i=1}^{K} w_i\ validity(C_i)$$

The *validity* function can be cohesion (compactness, tightness), separation (isolation), or some combination.

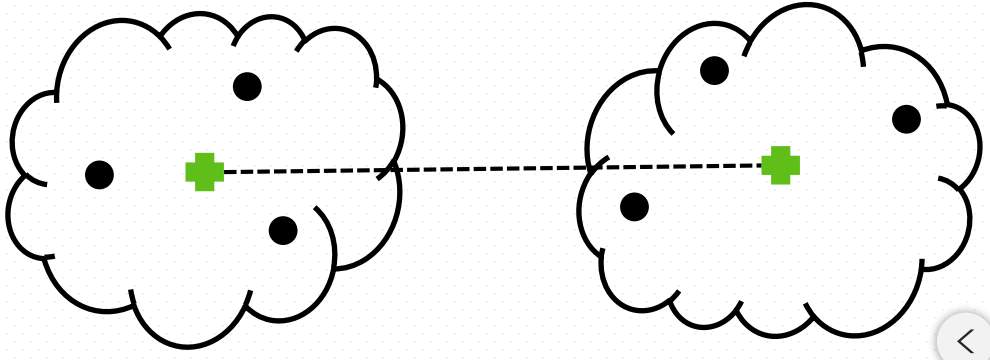
Prototype-Based Validity

- **Cohesion** can be defined as the sum of the proximities with respect to the prototype of the cluster.
- **Separation** can be measured by the proximity of the two cluster prototypes.

Cohesion



Separation



The Silhouette Coefficient

This method combines cohesion and separation:

1. For the i^{th} object, calculate its average distance to all other objects in its cluster, a_i.
2. For the i^{th} object and any cluster not containing the object, calculate the object's average distance to all the objects in the given cluster. Find the minimum such value with respect to all clusters, b_i.
3. For the i^{th} object, the silhouette coefficient is:

$$s_i = (b_i - a_i)/\max(a_i, b_i)$$

The Silhouette Coefficient

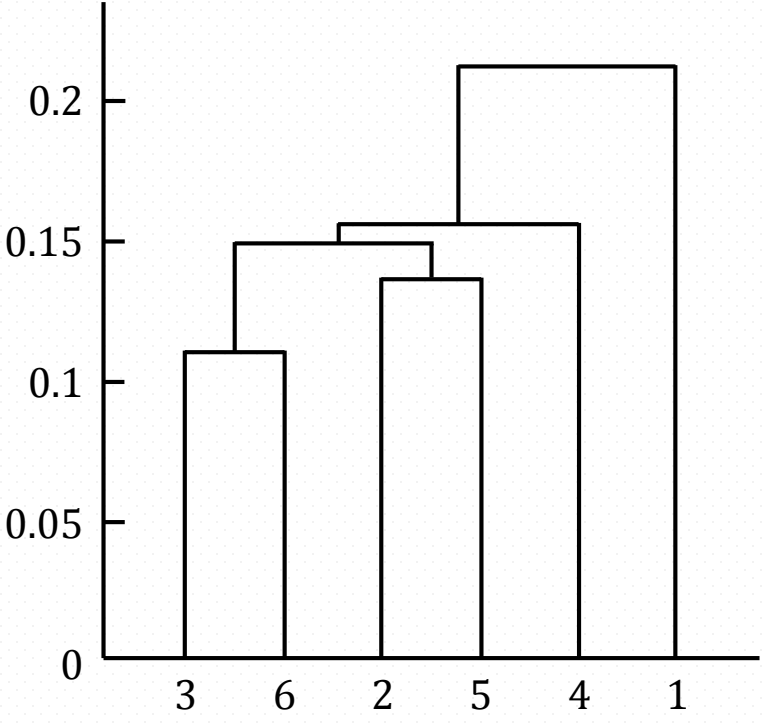
- The silhouette coefficient combines both cohesion and separation.
- The value can vary between -1 and 1.
  - A negative value is undesirable, because this corresponds to a case in which a_i, the average distance to points in the cluster, is greater than b_i, the minimum average distance to points in another cluster.
  - We want the silhouette coefficient to be positive $a_i < b_i$, and for a_i to be as close to 0 as possible.

Hierarchical Cluster Evaluation

- The **cophenetic distance** between two objects is the proximity at which an agglomerative hierarchical clustering technique puts the objects in the same cluster for the first time.
  - For example, if at some point in the agglomerative hierarchical clustering process, the smallest distance between the two clusters that are merged is 0.1, then all points in one cluster have a cophenetic distance of 0.1

Example of Cophenetic Distance
(Single Link Clustering)

Hierarchical Tree Diagram

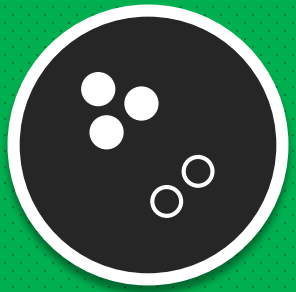


Cophenetic Distance Matrix

	p1	p2	p3	p4	p5	p6
p1	0	0.22	0.22	0.22	0.22	0.22
p2	0.22	0	0.15	0.15	0.14	0.15
p3	0.22	0.15	0	0.15	0.15	0.11
p4	0.22	0.15	0.15	0	0.15	0.15
p5	0.22	0.14	0.15	0.15	0	0.15
p6	0.22	0.15	0.11	0.15	0.15	0

Summarization of Clustering

- The greater the similarity (or homogenity) within a group and the greater the difference between groups, the better or more distinct the clustering.
- Clusters can be partitional (i.e., *k*-means) or hierarchical (i.e., agglomerative hierarchical).
- Clusterings can be *exclusive*, *overlapping*, or *fuzzy* (i.e., mixture models).
- Cluster evaluation assessed the validity of clusters.



Summarization of Clustering & Association

Summarization of Clustering & Association

- Association analysis and cluster analysis are both types of **unsupervised learning** (the process of trying to find hidden structure in *unlabeled* data).
- **Association analysis** is the process of discovering interesting relations between variables in large datasets.
- **Cluster analysis** is the process of grouping data objects based only on the description of objects and their relationships.