IS 733 Lesson 10

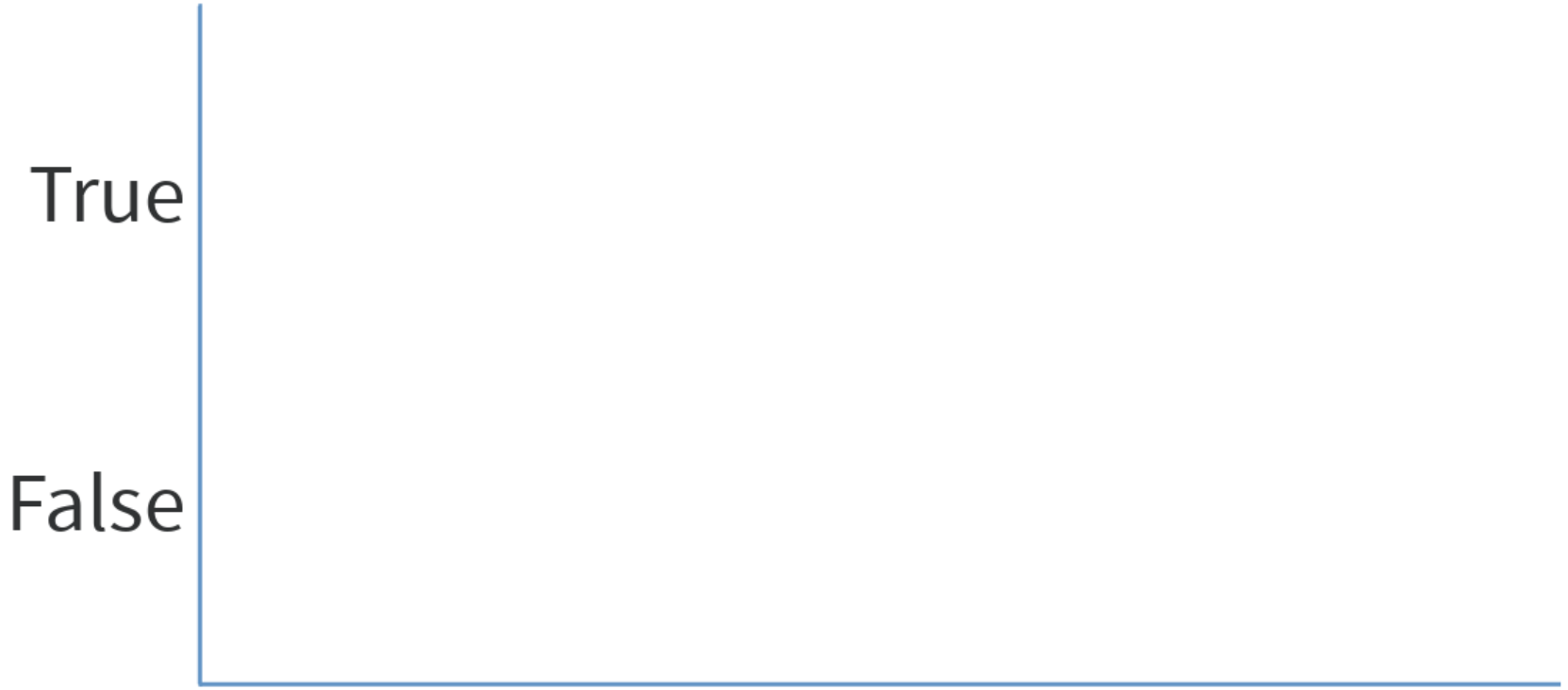
Unsupervised Learning:
Association Rule Learning and Clustering

Slides based on those from Data Mining by I. H. Witten, E. Frank, M. A. Hall and C. J. Pal, Data Mining: Concepts and Techniques by Han et al., and Vandana Janeja, James Foulds

Announcements

- Reminder: Homework 4 will be due on next week.

A separate-and-conquer algorithm can be used to efficiently perform association rule mining



Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

To efficiently perform association rule mining, there are two stages: generating the item sets with the specified _____ , and from each item set determining the rules that have the specified _____ .

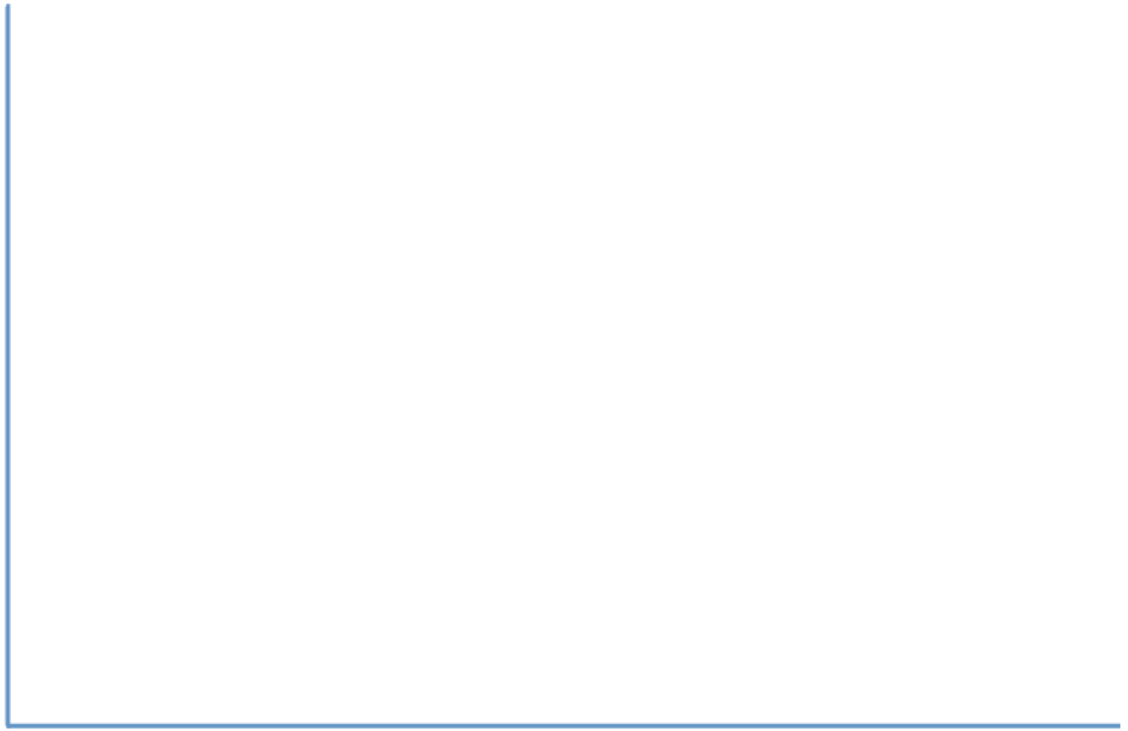
- Minimum accuracy, minimum coverage.
- Minimum coverage, minimum accuracy
- Maximum accuracy, maximum coverage
- Maximum coverage, maximum accuracy

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Which clustering method does not require the number of clusters to be specified in advance?

K-means

Agglomerative clustering



Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Agglomerative clustering is a _____ approach to hierarchical clustering.

Top-down

Bottom-up

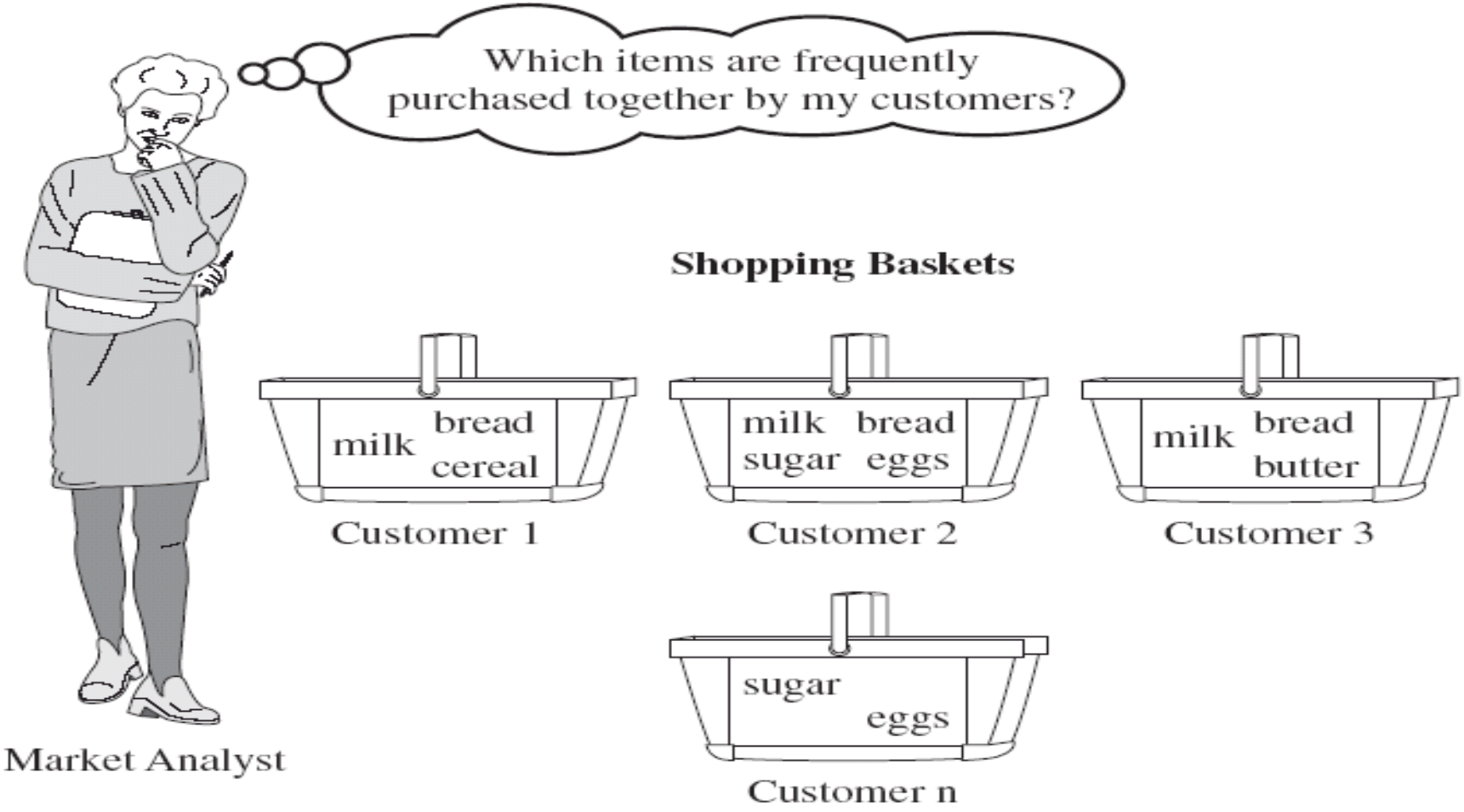
Start the presentation to see live content. Still no live content? Install the app or get help at **PollEv.com/app**

Learning outcomes

By the end of the lesson, you should be able to:

- **Describe** the steps of the Apriori algorithm for association rule mining
- **Implement** the **K-means** algorithm
- **Explain** the steps of the main **hierarchical clustering** algorithms
- **Make appropriate algorithmic choices** for clustering

Frequent Pattern Analysis and Association Rule Mining

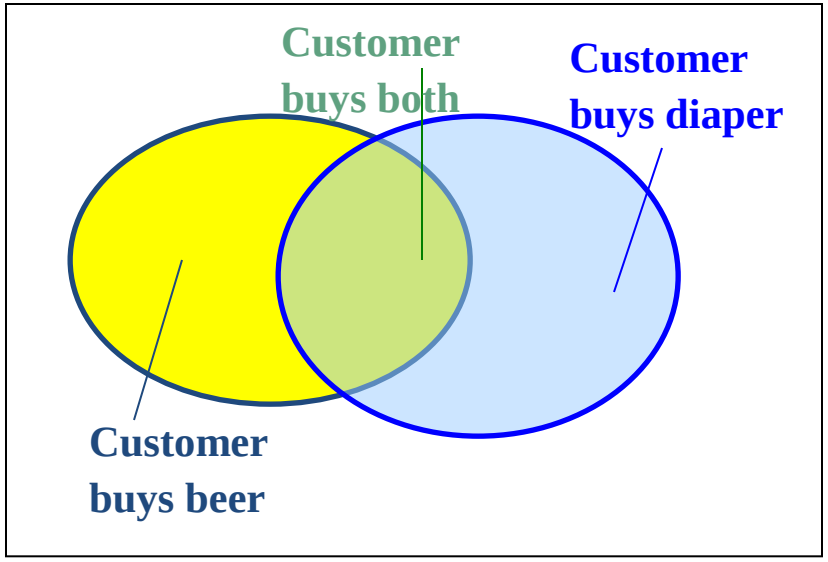


Frequent Pattern Analysis and Association Rule Mining

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Basic Concepts: Frequent Patterns and Association Rules

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F



- Itemset X = $\{x_1, \ldots, x_k\}$
  - In this case {A,B,C,D,E,F}
- Find all the rules *X* → *Y* with minimum support and confidence
  - support, *s*, probability that a transaction contains both X ∪ Y. *(AKA **coverage**)*
  - confidence, *c*, conditional probability that a transaction having X also contains *Y* *(AKA **accuracy**)*

Association measures

- Support ($X \rightarrow Y$):

$$P(X,Y) = \frac{\#\{\text{customers who bought } X \text{ and } Y\}}{\#\{\text{customers}\}}$$

- Confidence ($X \rightarrow Y$):

$$P(Y \mid X) = \frac{P(X,Y)}{P(X)}$$

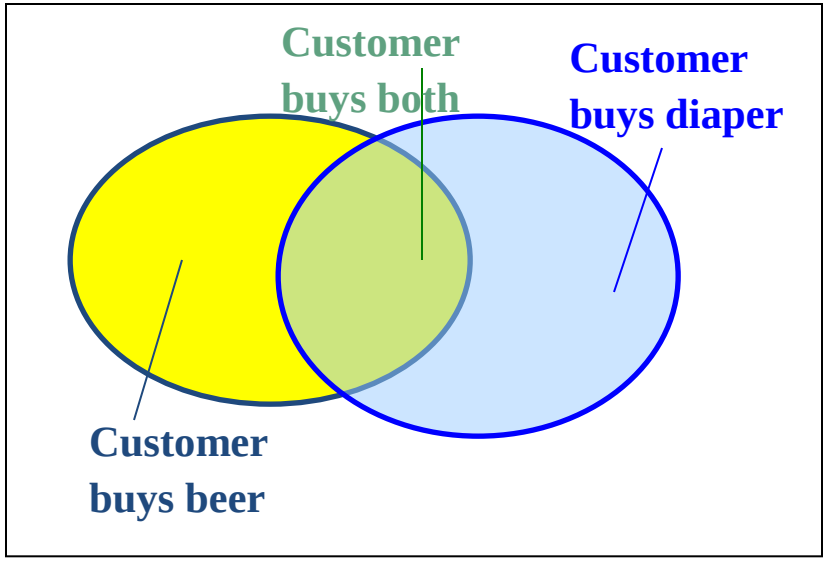
$$= \frac{\#\{\text{customers who bought } X \text{ and } Y\}}{\#\{\text{customers who bought } X\}}$$

- Lift ($X \rightarrow Y$):

$$= \frac{P(X,Y)}{P(X)P(Y)} = \frac{P(Y \mid X)}{P(Y)}$$

Basic Concepts: Frequent Patterns and Association Rules

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F



Let sup_{min} *= 50%,* $conf_{min}$ *= 50%*
Freq. Pat.: {A:3, B:3, D:4, E:3, AD:3}

Association rules:

$A \rightarrow D$ (60%, 100%)

$D \rightarrow A$ (60%, 75%)

Significance of Association measures

- *Confidence*
  - Conditional probability
  - Value should be close to 1
  - Strength of the rule, rule holds enough confidence
- *Support*
  - Statistical significance of the rule
  - Strong confidence value but # of such customers is small, rule is worthless
- Minimum support and confidence are set by the user/entity
  - Rules with higher support and confidence are searched for in the database

Association Rules

- In general, X and Y can be a sets of items
  - Basket analysis
  - E.g., customers buying hot dogs and buns are more likely to buy mustard and catsup
- Association rule mining
  - Given database of customer purchases
  - Find all association rules with high support and confidence
  - Apriori algorithm [Agrawal et al., 1996]

Mining association rules

- Naïve method for finding association rules:
  - Use separate-and-conquer method
  - Treat every possible combination of attribute values as a separate class
- Two problems:
  - Computational complexity
  - Resulting number of rules (which would have to be pruned on the basis of support and confidence)
- It turns out that we can look for association rules with high support and accuracy directly

Item sets: the basis for finding rules

- ***Item*:** one test/attribute-value pair
- ***Item set*:** all items occurring in a rule
- **Support:** number (or %) of instances which contain the association rule's item set
  - The same as the number of instances covered by *all* tests in the rule (LHS and RHS!)

- Goal: find only rules that exceed pre-defined support
  - Do it by finding all item sets with the given minimum support and generating rules from them!

Weather data

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Item sets for weather data

One-item sets	Two-item sets	Three-item sets	Four-item sets
Outlook = Sunny (5)	Outlook = Sunny Temperature = Hot (2)	Outlook = Sunny Temperature = Hot Humidity = High (2)	Outlook = Sunny Temperature = Hot Humidity = High Play = No (2)
Temperature = Cool (4)	Outlook = Sunny Humidity = High (3)	Outlook = Sunny Humidity = High Windy = False (2)	Outlook = Rainy Temperature = Mild Windy = False Play = Yes (2)
…	…	…	…

- Total number of item sets with a minimum support of at least two instances: 12 one-item sets, 47 two-item sets, 39 three-item sets, 6 four-item sets and 0 five-item sets

Generating rules from an item set

- Once all item sets with the required minimum support have been generated, we can turn them into rules
- Example 3-item set with a support of 4 instances:

```
Humidity = Normal, Windy = False, Play = Yes (4)
```

- Seven (2^N-1) potential rules:

```
If Humidity = Normal and Windy = False then Play = Yes    4/4
If Humidity = Normal and Play = Yes then Windy = False    4/6
If Windy = False and Play = Yes then Humidity = Normal    4/6
If Humidity = Normal then Windy = False and Play = Yes    4/7
If Windy = False then Humidity = Normal and Play = Yes    4/8
If Play = Yes then Humidity = Normal and Windy = False    4/9
If True then Humidity = Normal and Windy = False
   and Play = Yes                                         4/12
```

Rules for weather data

- All rules with support > 1 and confidence = 100%:

	Association rule		Sup.	Conf.
1	Humidity=Normal Windy=False	⇒ Play=Yes	4	100%
2	Temperature=Cool	⇒ Humidity=Normal	4	100%
3	Outlook=Overcast	⇒ Play=Yes	4	100%
4	Temperature=Cold Play=Yes	⇒ Humidity=Normal	3	100%
	...	...	...	...
58	Outlook=Sunny Temperature=Hot	⇒ Humidity=High	2	100%

- In total:
  - 3 rules with support four
  - 5 with support three
  - 50 with support two

Example rules from the same item set

- Item set:

```
Temperature = Cool, Humidity = Normal, Windy = False, Play = Yes (2)
```

- Resulting rules (all with 100% confidence):

```
Temperature = Cool, Windy = False ⇒ Humidity = Normal, Play = Yes
Temperature = Cool, Windy = False, Humidity = Normal ⇒ Play = Yes
Temperature = Cool, Windy = False, Play = Yes ⇒ Humidity = Normal
```

- We can establish their confidence due to the following "frequent" item sets:

```
Temperature = Cool, Windy = False                      (2)
Temperature = Cool, Humidity = Normal, Windy = False   (2)
Temperature = Cool, Windy = False, Play = Yes          (2)
```

Generating item sets efficiently

- How can we efficiently find all frequent item sets?
- Finding one-item sets easy
- Idea: use one-item sets to generate two-item sets, two-item sets to generate three-item sets, …
  - If (A B) is a frequent item set, then (A) and (B) have to be frequent item sets as well!
  - In general: if X is a frequent *k*-item set, then all (*k*-1)-item subsets of X are also frequent
  - Compute *k*-item sets by merging (*k*-1)-item sets

Apriori: A Candidate Generation & Test Approach

- **Apriori pruning principle**: If there is any itemset which is infrequent, its superset should not be generated/tested! *(Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)*

- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

Example

- Given: five frequent three-item sets

  (A B C), (A B D), (A C D), (A C E), (B C D)

- Lexicographically ordered!
- Candidate four-item sets:

  (A B C D) **OK because of (A C D) (B C D)**

  (A C D E) **Not OK because of (C D E)**

- To establish that these item sets are really frequent, we need to perform a final check by counting instances
- For fast look-up, the (k –1)-item sets are stored in a hash table

Algorithm for finding item sets

```
Set k to 1
Find all k-item sets with sufficient coverage and store them in hash table #1
While some k-item sets with sufficient coverage have been found
    Increment k
    Find all pairs of (k–1)-item sets in hash table #(k–1) that differ only in
    their last item
    Create a k-item set for each pair by combining the two (k–1)-item sets
    that are paired
    Remove all k-item sets containing any (k–1)-item sets that are not in the
    #(k–1)hash table
    Scan the data and remove all remaining k-item sets that do not have
    sufficient coverage
    Store the remaining k-item sets and their coverage in hash table #k,
    sorting items in lexical order
```

Generating rules efficiently

- We are looking for all high-confidence rules
  - Support of antecedent can be obtained from item set hash table
  - But: brute-force method is (2^N-1) for an N-item set
- Better way: building (c + 1)-consequent rules from c-consequent ones
  - Observation: (c + 1)-consequent rule can only hold if all corresponding c-consequent rules also hold
- Resulting algorithm similar to procedure for large item sets

Example

- 1-consequent rules:

```
If Outlook = Sunny and Windy = False and Play = No
   then Humidity = High (2/2)
```

```
If Humidity = High and Windy = False and Play = No
   then Outlook = Sunny (2/2)
```

- Corresponding 2-consequent rule:

```
If Windy = False and Play = No
   then Outlook = Sunny and Humidity = High (2/2)
```

- Final check of antecedent against item set hash table is required to check that rule is actually sufficiently accurate

Algorithm for finding association rules

Set *n* to 1

Find all sufficiently accurate n-consequent rules for the *k*-item set and store them in hash table #1, computing accuracy using the hash tables found for item sets

While some sufficiently accurate *n*-consequent rules have been found

- Increment *n*
- Find all pairs of (*n*–1)-consequent rules in hash table #(*n*–1) whose consequents differ only in their last item
- Create an *n*-consequent rule for each pair by combining the two (*n*–1)-consequent rules that are paired
- Remove all *n*-consequent rules that are insufficiently accurate, computing accuracy using the hash tables found for item sets
- Store the remaining *n*-consequent rules and their accuracy in hash table #*k*, sorting items for each consequent in lexical order

Apriori algorithm for association rules

- **Representation**
  - Association rules
- **Objective function for training**
  - Coverage and accuracy of rules
- **Search algorithm**
  - Construct frequent item sets of increasing size, and of sufficient coverage, based on smaller item sets. From each item set, construct rules of increasing consequent size, and of sufficient accuracy, based on smaller sets of consequents.

What is Clustering?

- Cluster: A collection of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis (or *clustering*, *data segmentation, …*)
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- Unsupervised learning: no predefined classes (i.e., *learning by observations* vs. learning by examples: supervised)
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

Clustering

- Clustering techniques apply when there is no class to be predicted: they perform unsupervised learning
- Aim: divide instances into “natural” groups
- Clusters can be:
  - disjoint vs. overlapping
  - deterministic vs. probabilistic
  - flat vs. hierarchical
- We will look at a classic clustering algorithm called *k-means*
- *k-means* clusters are disjoint, deterministic, and flat

Clustering for Data Understanding and Applications

- Biology: taxonomy of living things: kingdom, phylum, class, order, family, genus and species
- Information retrieval: document clustering
- Land use: Identification of areas of similar land use in an earth observation database
- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- City-planning: Identifying groups of houses according to their house type, value, and geographical location
- Earth-quake studies: Observed earthquake epicenters should be clustered along continent faults
- Climate: understanding earth climate, find patterns of atmospheric and ocean
- Economic Science: market research

Clustering as a Preprocessing Tool (Utility)

- Summarization:
  - Preprocessing for regression, PCA, classification, and association analysis
- Compression:
  - Image processing: vector quantization
- Finding K-nearest Neighbors
  - Localizing search to one or a small number of clusters
- Outlier detection
  - Outliers are often viewed as those “far away” from any cluster

Partitioning Algorithms: Basic Concept

- Partitioning method: Partitioning a database ***D*** of ***n*** objects into a set of ***k*** clusters, such that the sum of squared distances is minimized (where c_i is the centroid or medoid of cluster C_i)

$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} \| p - c_i \|^2$$

- Given *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
  - *k-means* (MacQueen'67, Lloyd'57/'82): Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

The *k*-means algorithm

- **Step 1:** Choose *k* random cluster centers
- **Step 2:** Assign each instance to its closest cluster center based on Euclidean distance
- **Step 3:** Recompute cluster centers by computing the average (aka *centroid*) of the instances pertaining to each cluster
- **Step 4:** If cluster centers have moved, go back to Step 2

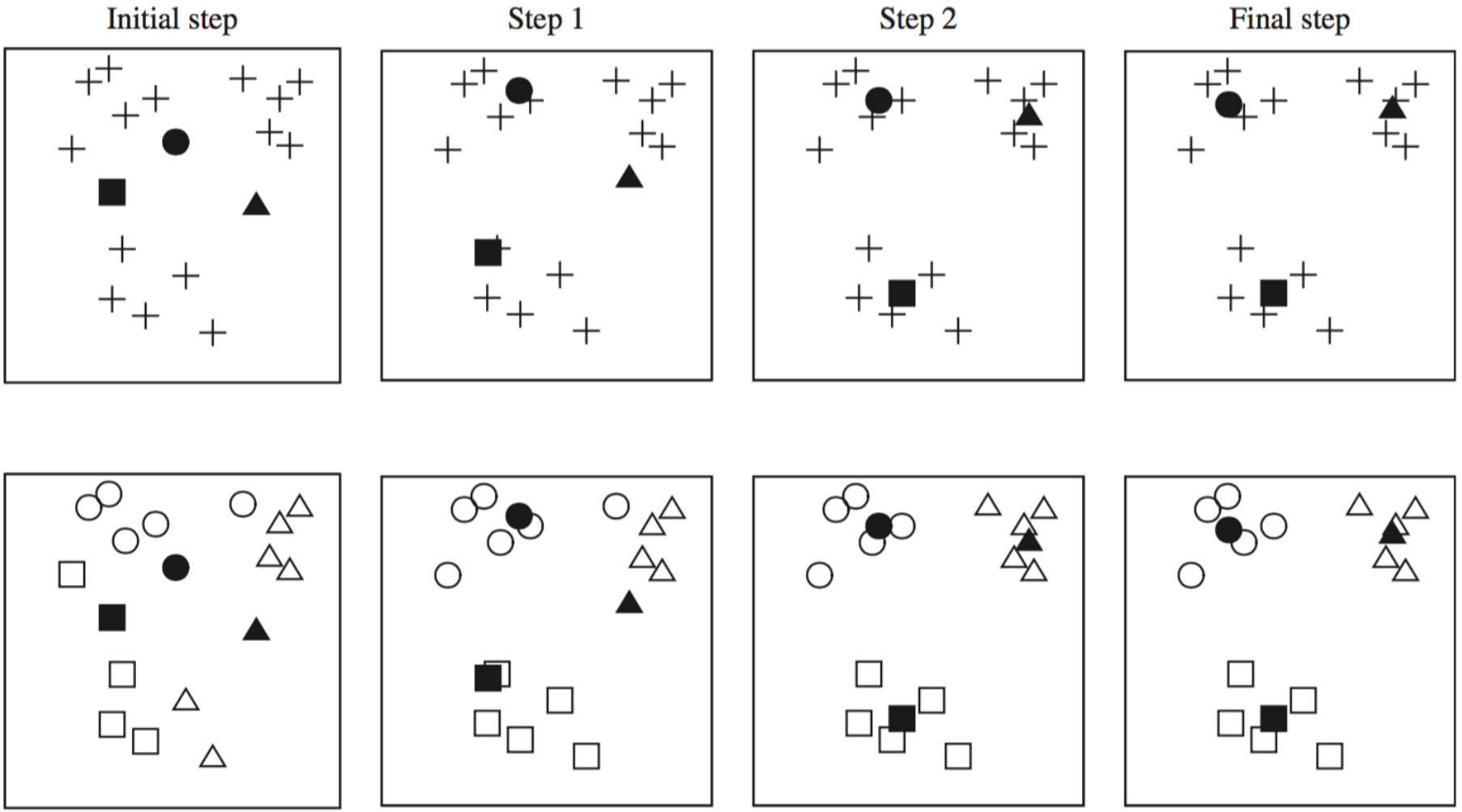
The *k*-means algorithm

- Equivalent termination criterion: stop when assignment of instances to cluster centers has not changed

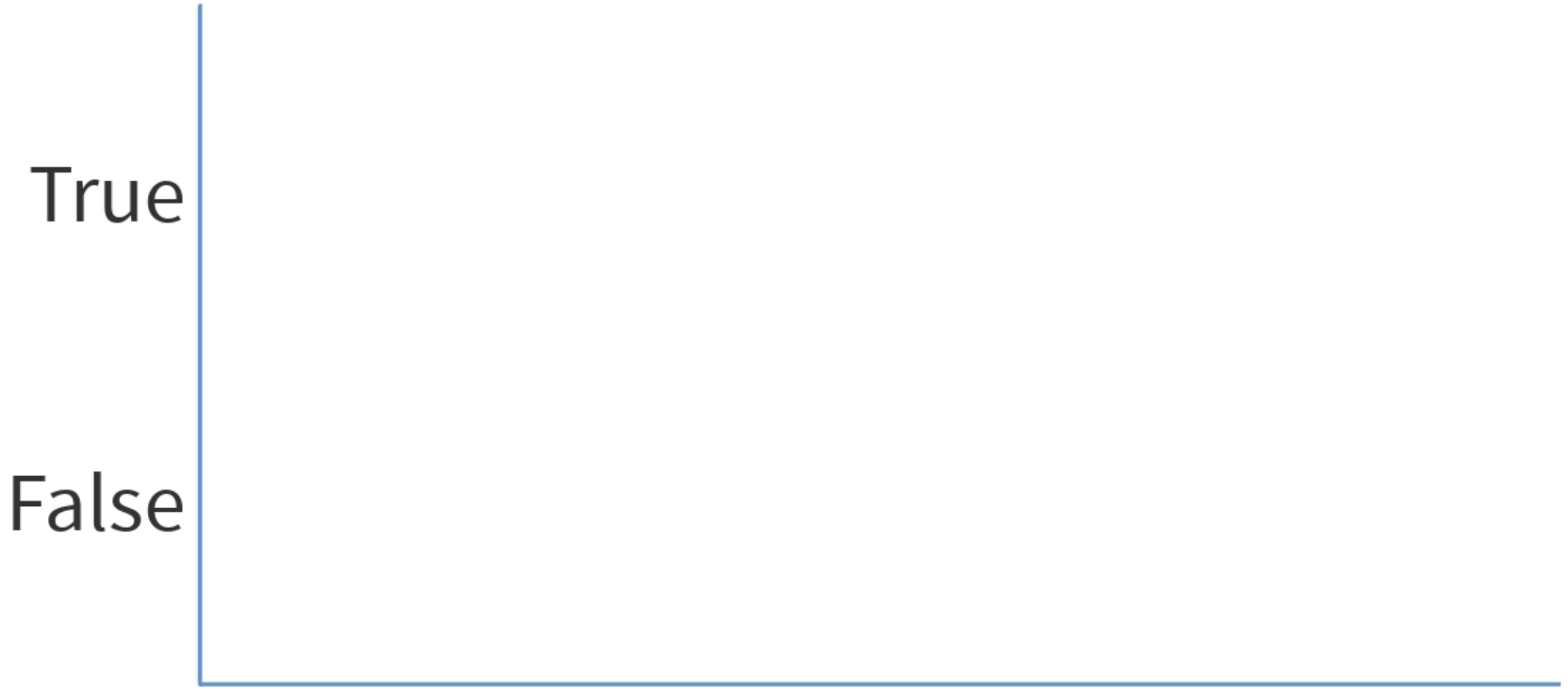
- This algorithm minimizes the squared Euclidean distance of the instances from their corresponding cluster centers:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} \| p - c_i \|^2$$

The *k*-means algorithm: example



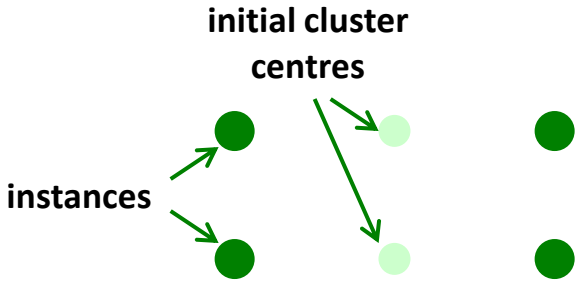
Depending on the random initialization of K-means, it may converge to a different solution.



Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

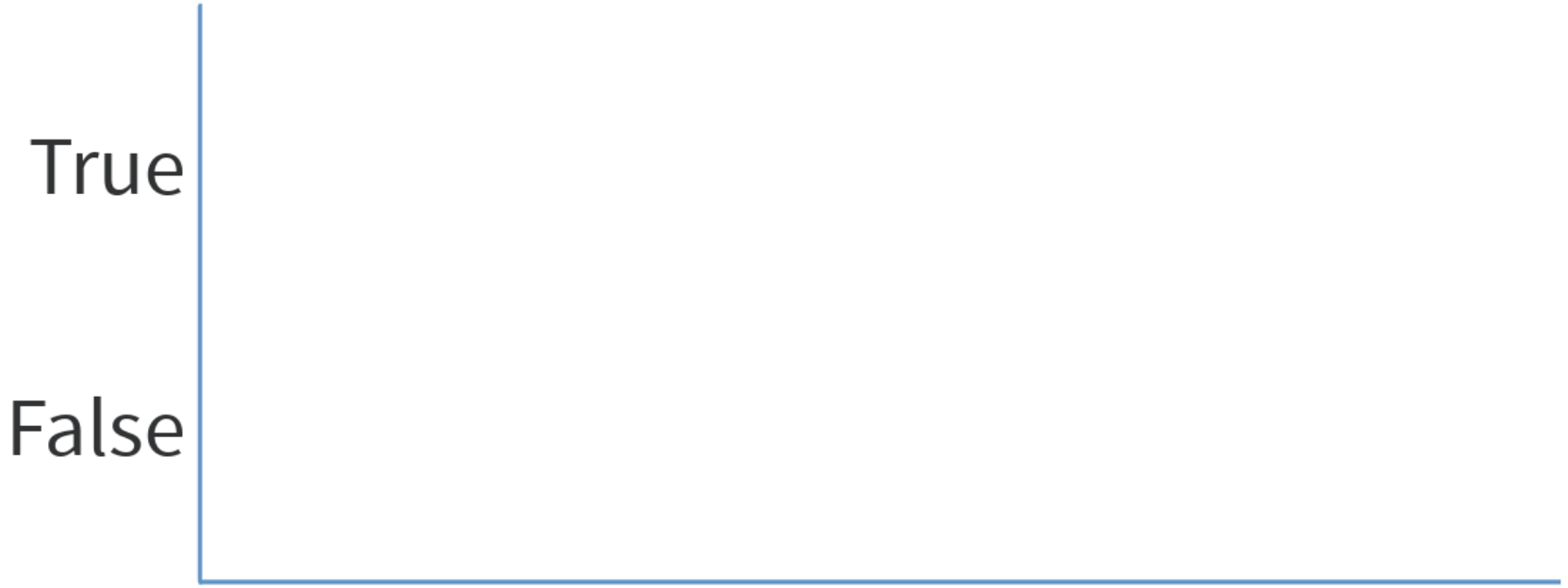
Discussion

- Result can vary significantly
  - based on initial choice of seeds
- Can get trapped in local minimum
  - Example:



- To increase chance of finding global optimum: restart with different random seeds

The objective function that K-means aims to minimize, the sum of the squared distances between instances and their cluster centers, can go up in some iterations.



Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Discussion

- K-means minimizes squared distance to cluster centers

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} \| p - c_i \|^2$$

  - Both steps improve the squared distance objective function
  - Therefore, in each iteration, the objective function improves (until we terminate)

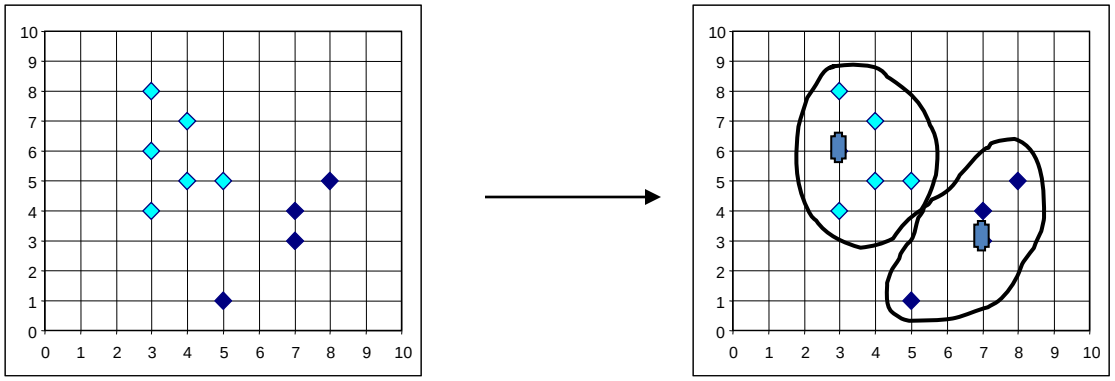
- Can be applied recursively with $k = 2$ (bisecting k-means)
  - This produces a hierarchical clustering

Comments on the *K-Means* Method

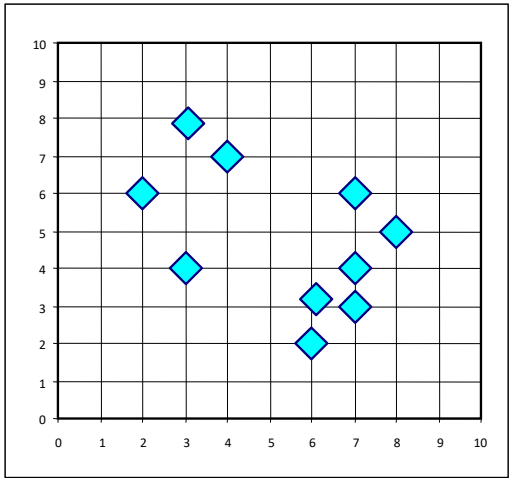
- Strength: *Efficient*: *O*(*tkn*), where *n* is # objects, *k* is # clusters, and *t* is # iterations. Normally, *k*, *t* << *n*.
    - Comparing: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$
- Comment: Often terminates at a *local optimal*.
- Weakness
  - Applicable only to objects in a continuous n-dimensional space
    - Using the k-modes method for categorical data
    - In comparison, k-medoids can be applied to a wide range of data
  - Need to specify *k,* the *number* of clusters, in advance (there are ways to automatically determine the best k (see Hastie et al., 2009)
  - Sensitive to noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

What Is the Problem of the K-Means Method?

- The k-means algorithm is sensitive to outliers !
  - Since an object with an extremely large value may substantially distort the distribution of the data
- K-Medoids: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster

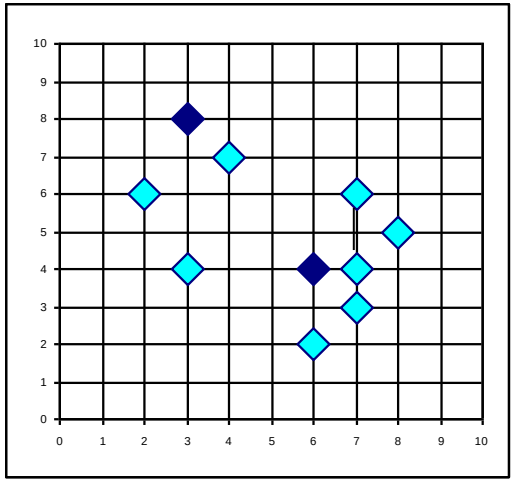


PAM: A Typical K-Medoids Algorithm



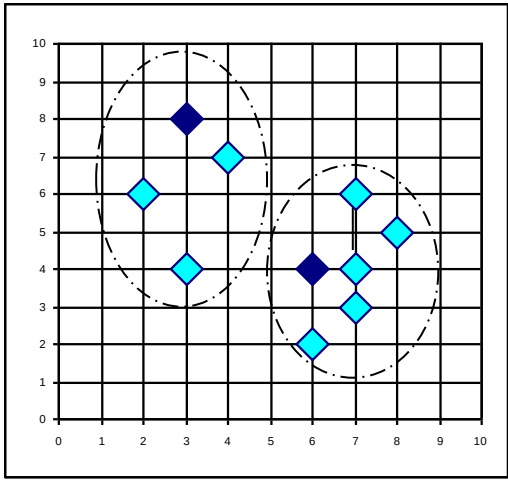
K=2

Arbitrary choose k object as initial medoids

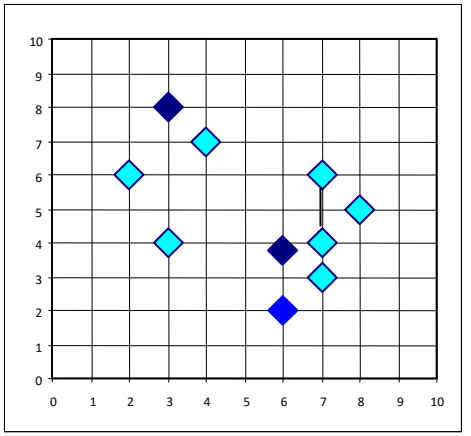


Assign each remaining object to nearest medoids

Total Cost = 20

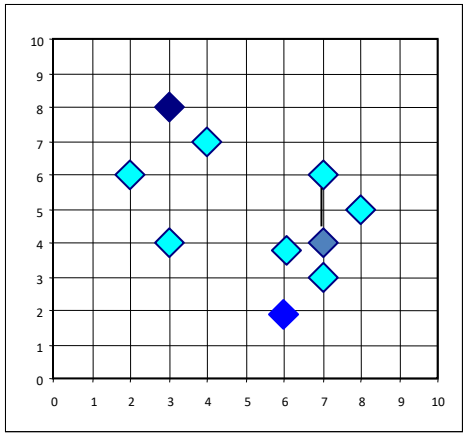


Randomly select a nonmedoid object, O_{random}



Compute total cost of swapping

Total Cost = 26



Swapping O and O_{random}

If quality is improved.

Do loop

Until no change

The K-Medoid Clustering Method

- *K-Medoids* Clustering: Find *representative* objects (medoids) in clusters
  - *PAM* (Partitioning Around Medoids, Kaufmann & Rousseeuw 1987)
    - Starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
    - *PAM* works effectively for small data sets, but does not scale well for large data sets (due to the computational complexity)
- Efficiency improvement on PAM
  - *CLARA* (Kaufmann & Rousseeuw, 1990): PAM on samples
  - *CLARANS* (Ng & Han, 1994): Randomized re-sampling

Choosing the number of clusters

- Big question in practice: what is the right number of clusters, i.e., what is the right value for *k*?

- Cannot simply optimize squared distance on training data to choose k
  - Squared distance decreases monotonically with increasing values of *k*

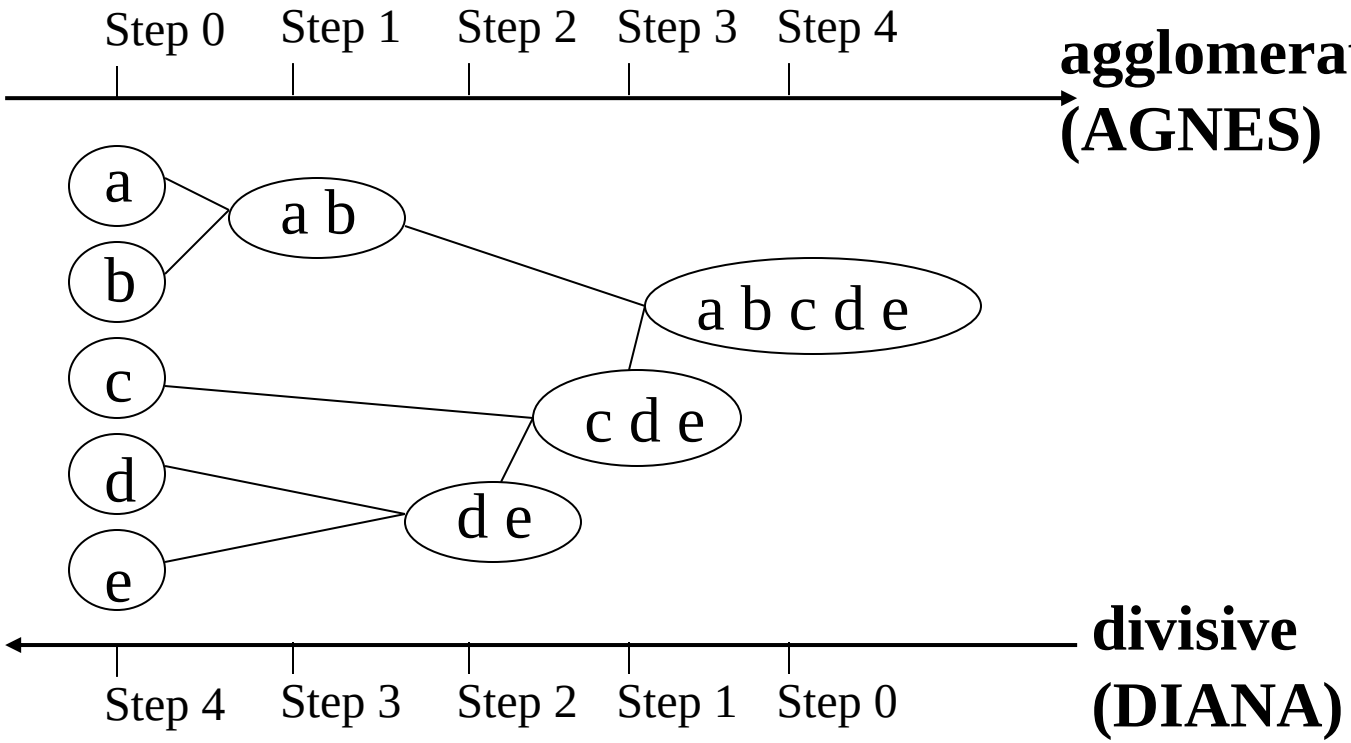
- Need some measure that balances distance with complexity of the model, e.g., based on the minimum description length (MDL) principle
  - information required to store data centroid, and the location of each instance with respect to this centroid

Hierarchical clustering

- Bisecting *k*-means performs hierarchical clustering in a top-down manner
- Standard hierarchical clustering performs clustering in a bottom-up manner; it performs *agglomerative* clustering:
  - First, make each instance in the dataset into a trivial mini-cluster
  - Then, find the two closest clusters and merge them; repeat
  - Clustering stops when all clusters have been merged into a single cluster

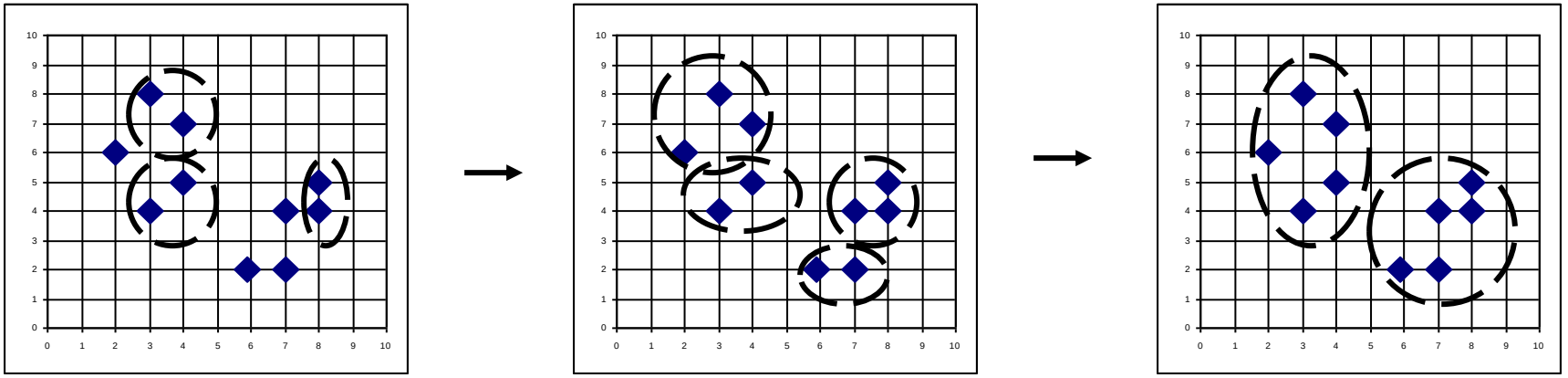
Hierarchical Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition



AGNES (Agglomerative Nesting)

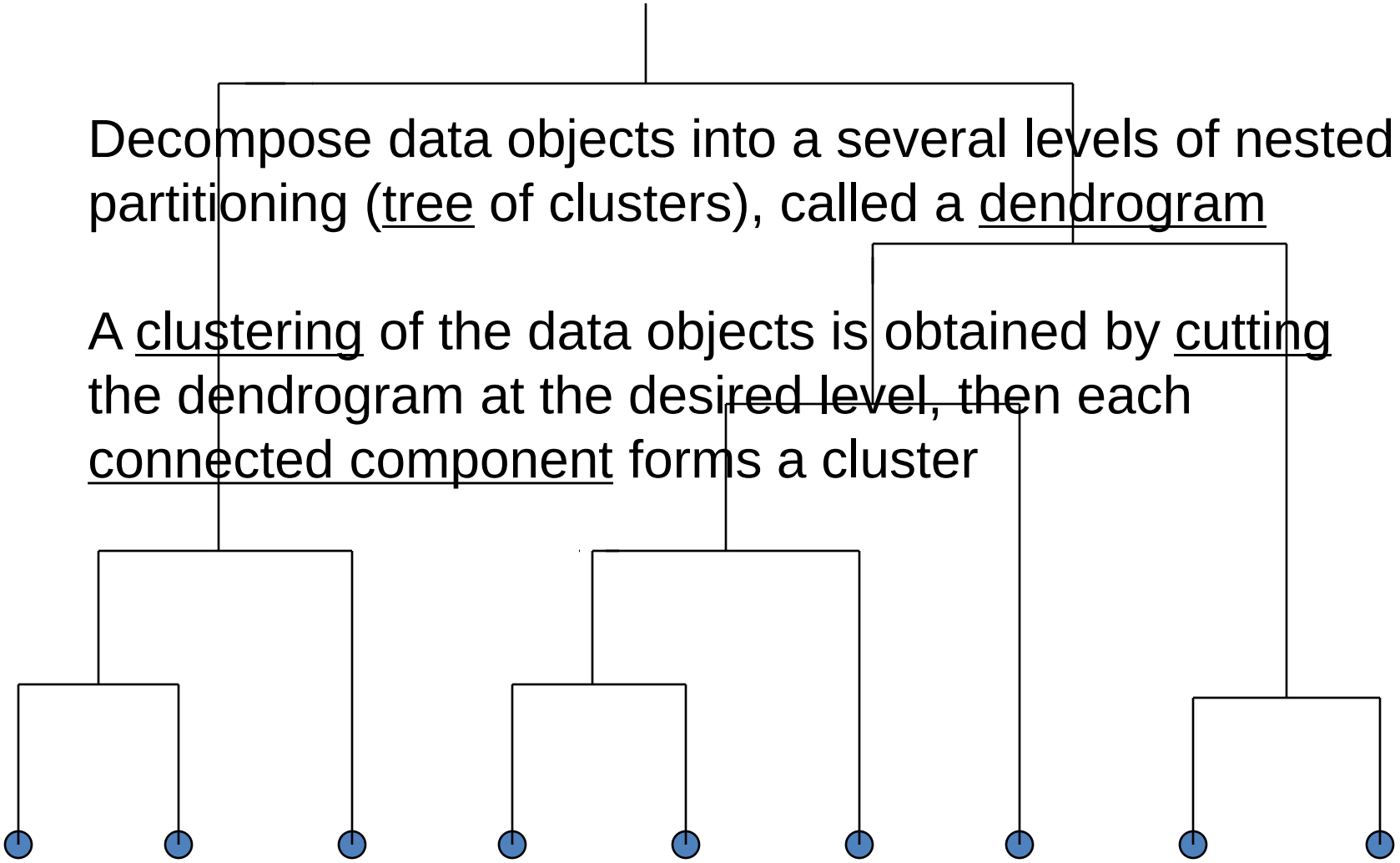
- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical packages, e.g., Splus
- Use the **single-link** method and the dissimilarity matrix
- Merge nodes that have the least dissimilarity
- Go on in a non-descending fashion
- Eventually all nodes belong to the same cluster



Dendrogram: Shows How Clusters are Merged

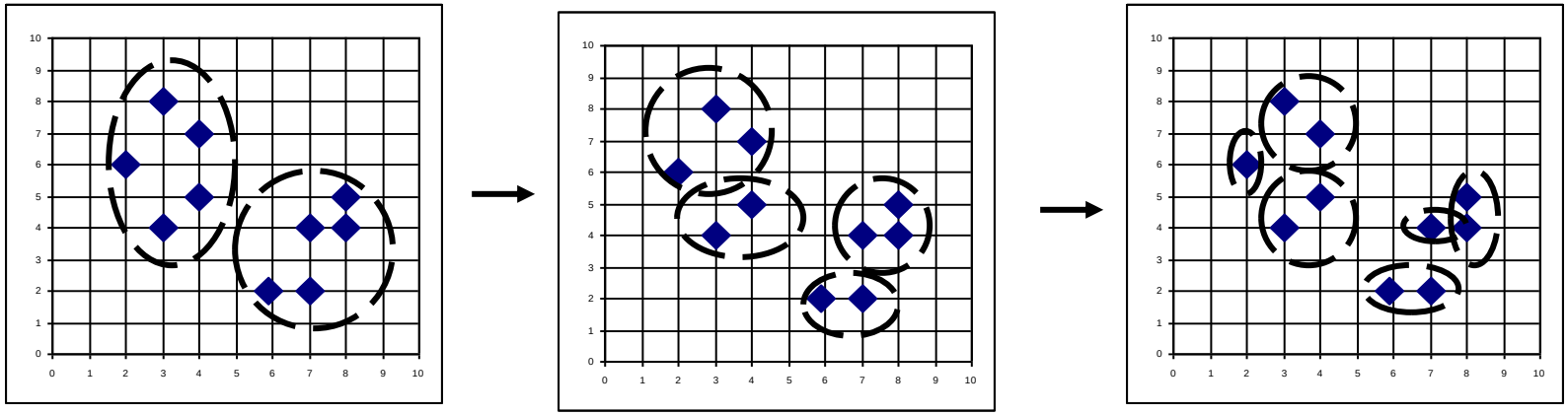
Decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram

A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster

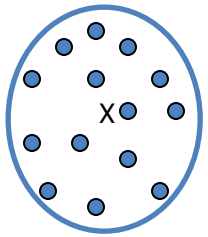


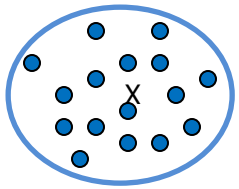
DIANA (Divisive Analysis)

- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical analysis packages, e.g., Splus
- Inverse order of AGNES
- Eventually each node forms a cluster on its own



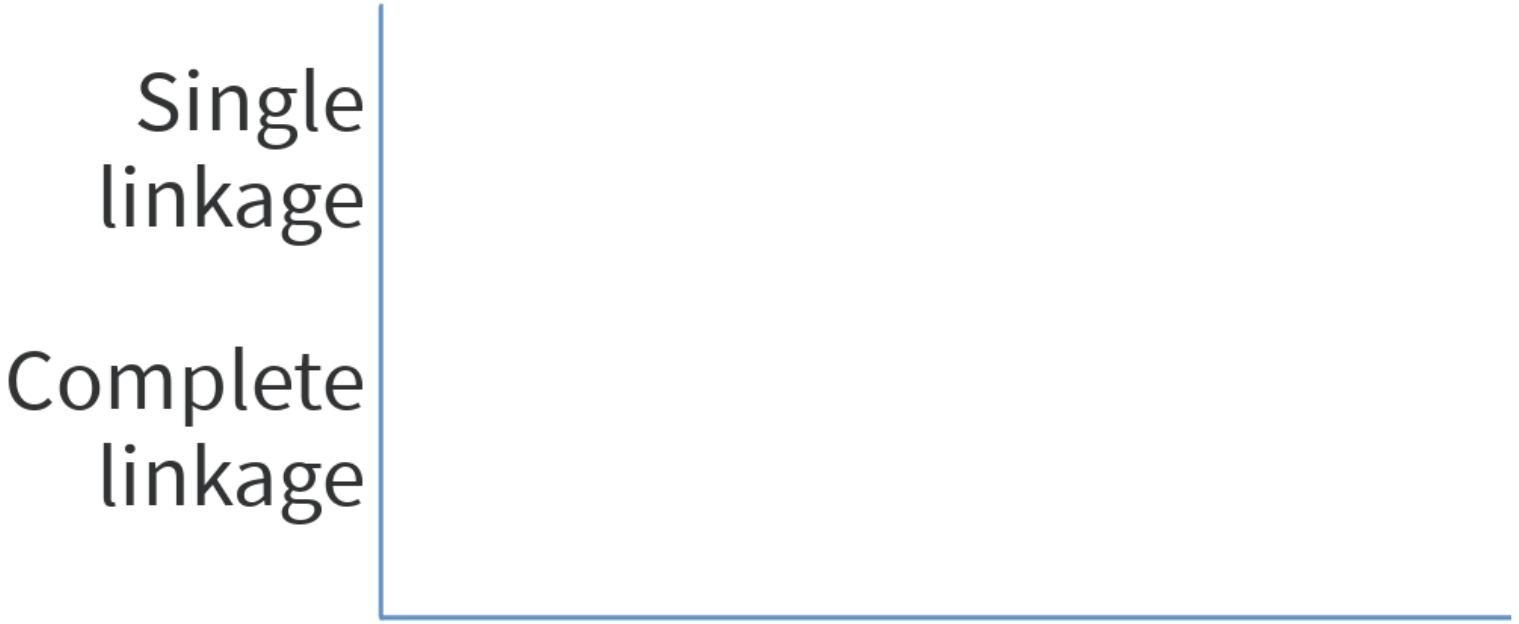
Distance between Clusters





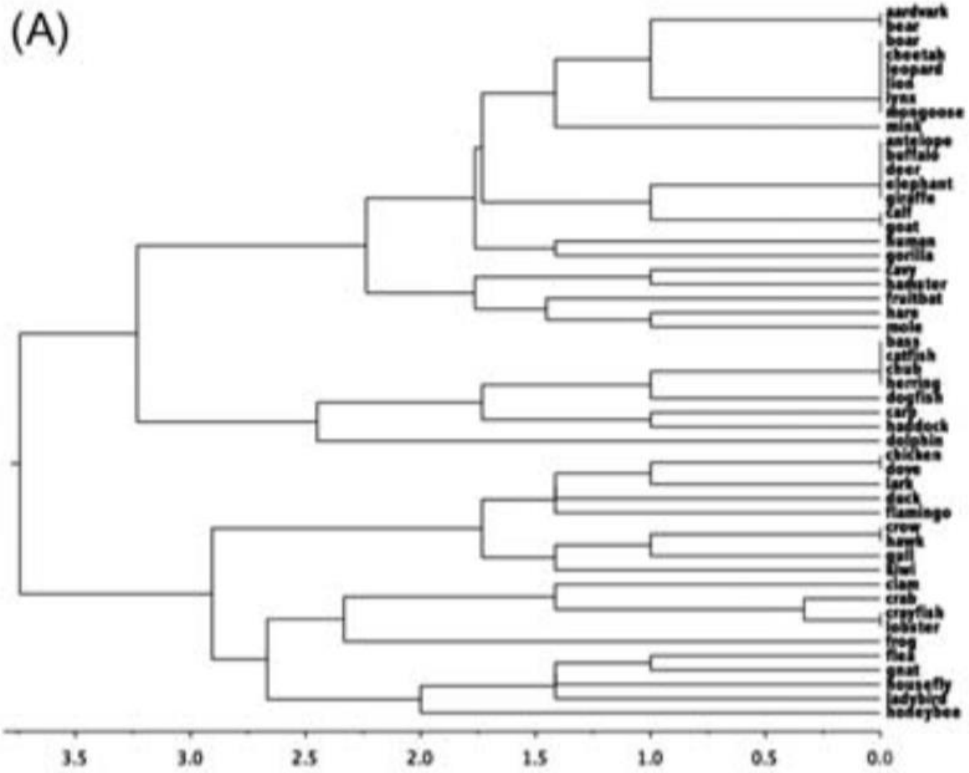
- **Single linkage:** smallest distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = min(t_{ip}, t_{jq})$
- **Complete linkage:** largest distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = max(t_{ip}, t_{jq})$
- **Average linkage:** avg distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = avg(t_{ip}, t_{jq})$
- **Centroid linkage:** distance between the centroids of two clusters, i.e., $dist(K_i, K_j) = dist(C_i, C_j)$
- **Medoid linkage:** distance between the medoids of two clusters, i.e., $dist(K_i, K_j) = dist(M_i, M_j)$
  - Medoid: a chosen, centrally located object in the cluster

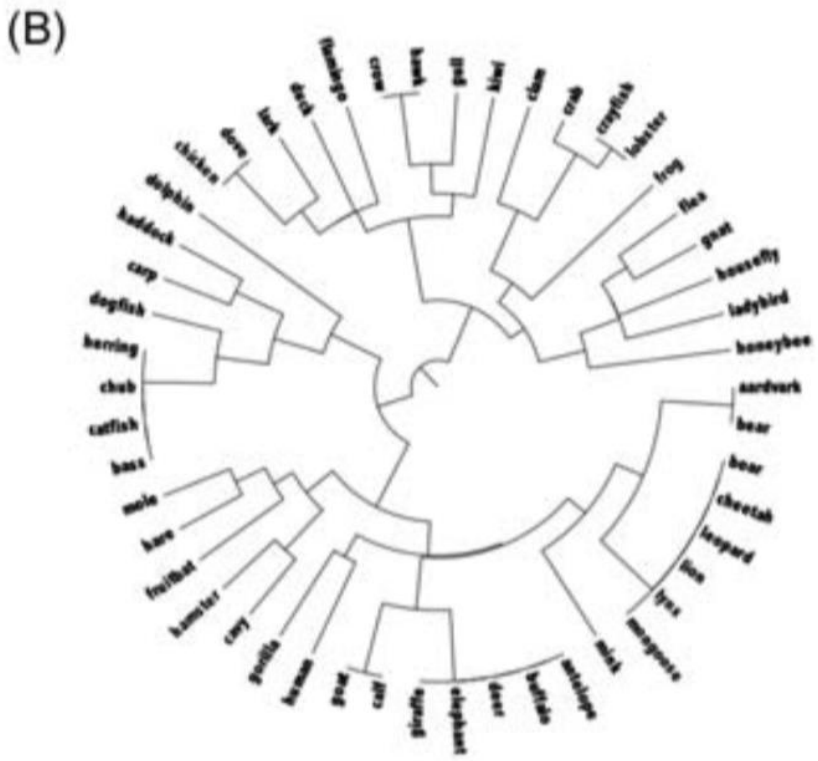
The "diameter" of a cluster is the greatest distance between any two of its members. Which method is best at keeping the diameter of the clusters small?



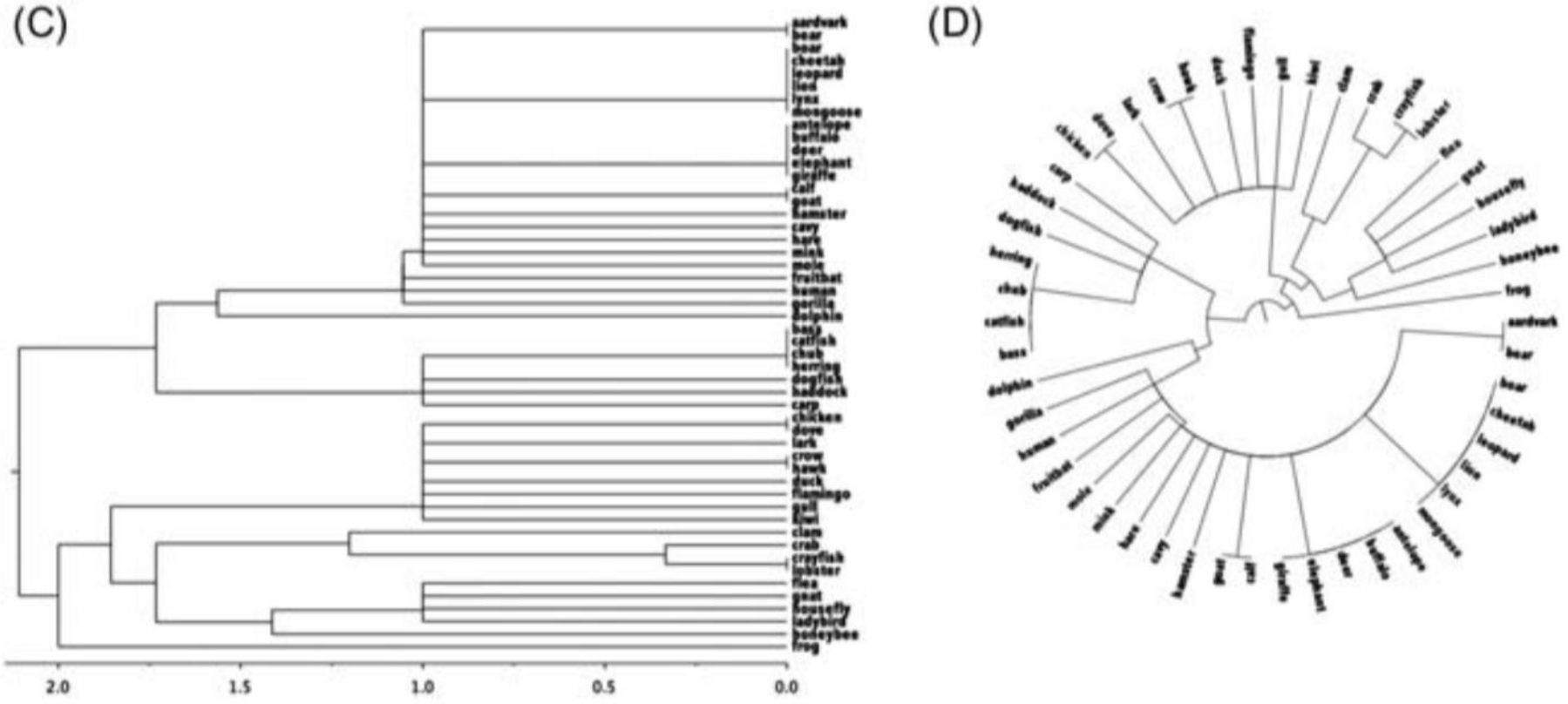
Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Example: complete linkage





Example: single linkage



Think-Pair-Share: Clustering Facebook Users

- Suppose you would like to cluster 10,000 users of the Facebook social network, who all attend the same university, into meaningful groups.

  You have access to everything Facebook stores, including friendship links, posts, “likes,” etc.

- How will you go about this?
  - Features and data pre-processing?
  - Algorithms, methods, and types of clustering?