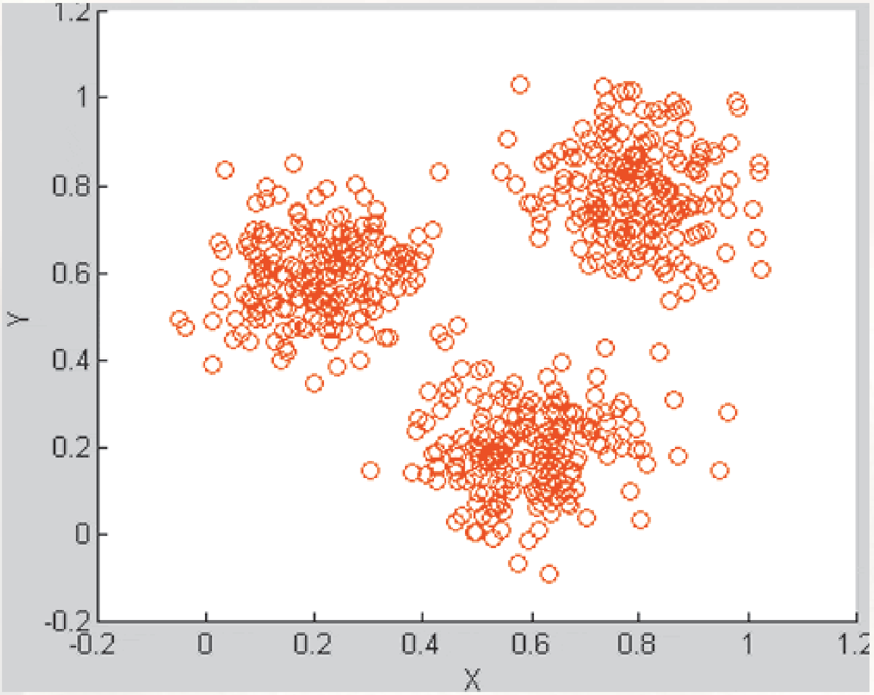
# Lection 12

## Unsupervised learning:

## Clustering and Association Rules

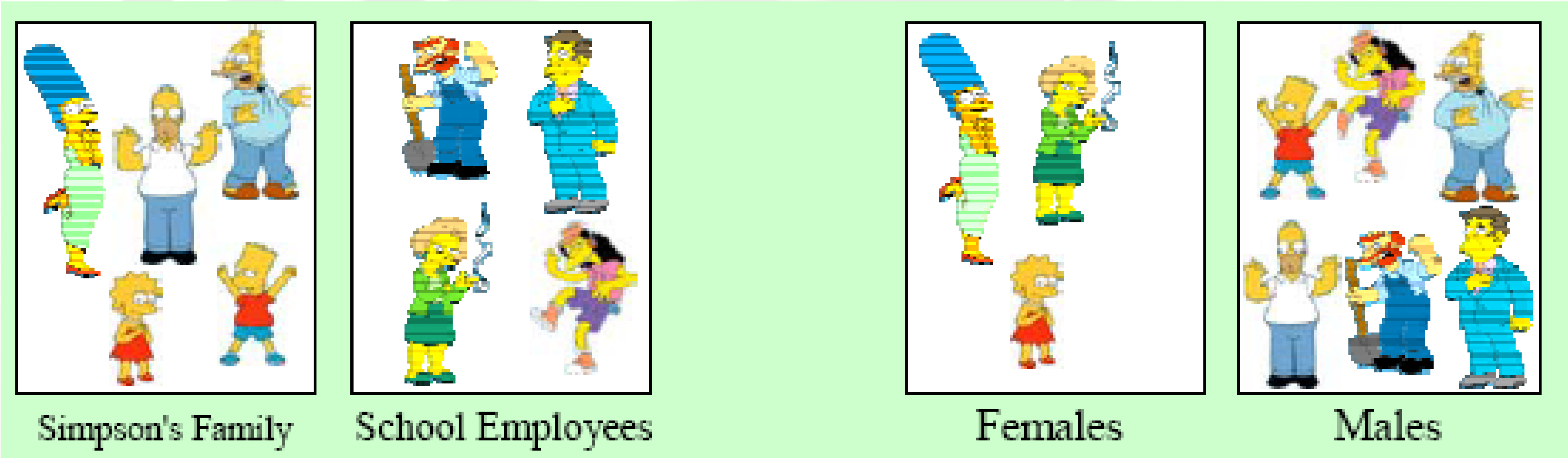
# Unsupervised Learning: Clustering

- Given:
  - Data Set D (training set)
  - Similarity/distance metric/information
- Find:
  - Partitioning of data
  - Groups of similar/close items



# Not a well-defined problem

What is a natural grouping among these objects?



# Similarity?

- Groups of similar customers
  - Similar demographics
  - Similar buying behavior
  - Similar health
- Similar products
  - Similar cost
  - Similar function
  - Similar store
  - …
- Similarity usually is domain/problem specific

# Distance Between Records

- *d*-dim vector space representation and distance metric

$r_1$: 57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0

$r_2$: 78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0

...

$r_N$: 18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Distance $(r_1,r_2)$ = ???

- Pairwise distances between points (no *d*-dim space)
  - Similarity/dissimilarity matrix (upper or lower diagonal)
  - Distance: 0 = near, ∞ = far
  - Similarity: 0 = far, ∞ = near

| -- | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | - | d | d | d | d | d | d | d | d | d |
| 2 | | - | d | d | d | d | d | d | d | d |
| 3 | | | - | d | d | d | d | d | d | d |
| 4 | | | | - | d | d | d | d | d | d |
| 5 | | | | | - | d | d | d | d | d |
| 6 | | | | | | - | d | d | d | d |
| 7 | | | | | | | - | d | d | d |
| 8 | | | | | | | | - | d | d |
| 9 | | | | | | | | | - | d |

# Properties of Distances: Metric Spaces

- A metric space is a set *S* with a global distance function *d.* For every two points *x, y* in *S*, the distance *d(x,y)* is a nonnegative real number.
- A metric space must also satisfy
  - d(x,y) = 0 iff *x* = *y*
  - d(x,y) = d(y,x) (symmetry)
  - d(x,y) + d(y,z) >= d(x,z) (triangle inequality)

# Minkowski Distance ($L_p$ Norm)

- Consider two records $x=(x_1,\ldots,x_d)$, $y=(y_1,\ldots,y_d)$:

$$d(x,y)=\sqrt[p]{|x_1-y_1|^p+|x_2-y_2|^p+\ldots+|x_d-y_d|^p}$$

Special cases:

- p=1: Manhattan distance

$$d(x,y)=|x_1-y_1|+|x_2-y_2|+\ldots+|x_p-y_p|$$

- p=2: Euclidean distance

$$d(x,y)=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_d-y_d)^2}$$

# Only Binary Variables

2x2 Table:

| | 0 | 1 | Sum |
|---|---|---|---|
| 0 | a | b | a+b |
| 1 | c | d | c+d |
| Sum | a+c | b+d | a+b+c+d |

- Simple matching coefficient: (symmetric) $$d(x, y) = \frac{b + c}{a + b + c + d}$$
- Jaccard coefficient: (asymmetric) $$d(x, y) = \frac{b + c}{b + c + d}$$

# Mixtures of Variables

- Weigh each variable differently
- Can take “importance” of variable into account (although usually hard to quantify in practice)

# Clustering: Informal Problem Definition

<u>Input:</u>

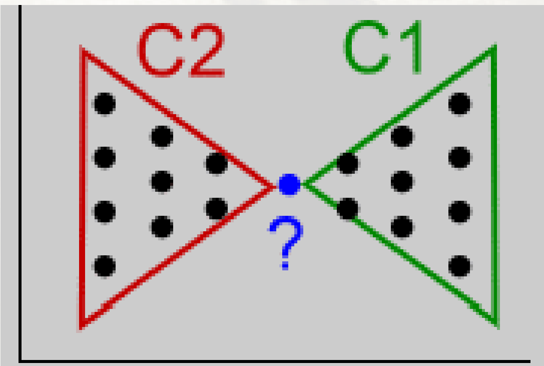
- A data set of *N* records each given as a *d*-dimensional data feature vector.

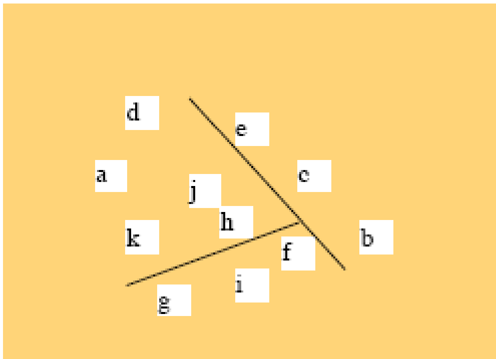
<u>Output:</u>

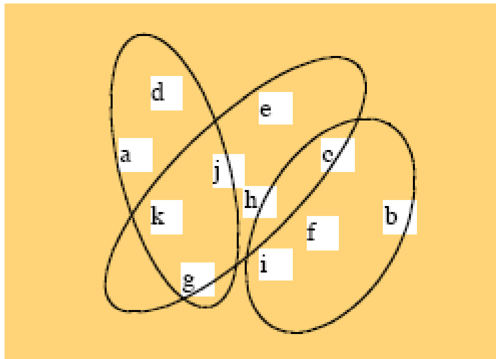
- Determine a natural, useful "partitioning" of the data set into a number of (k) clusters and noise such that we have:
  - High similarity of records within each cluster (intra-cluster similarity)
  - Low similarity of records between clusters (inter-cluster similarity)

# Types of Clustering

- Hard Clustering:
  - Each object is in one and only one cluster
- Soft Clustering:
  - Each object has a probability of being in each cluster



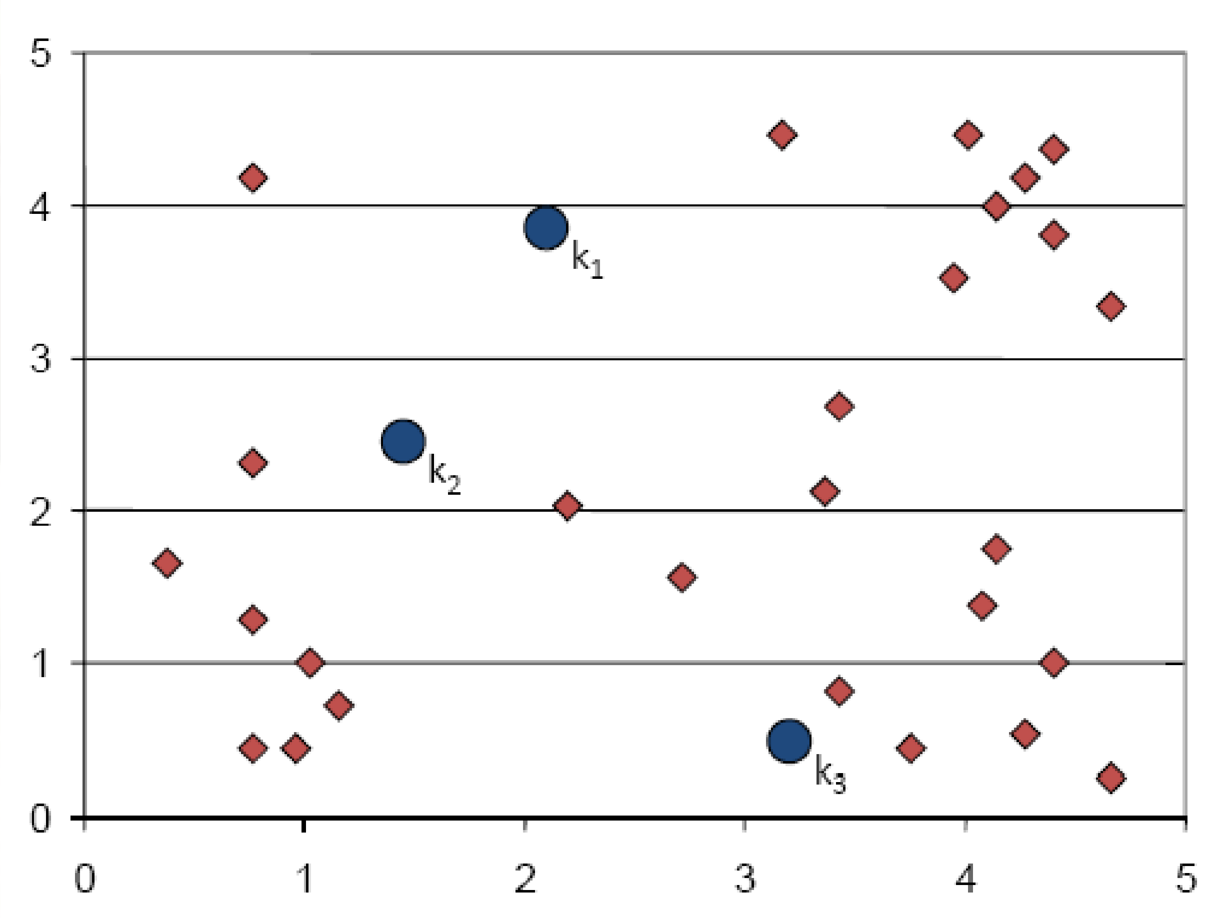
# Clustering Algorithms

- Partitioning-based clustering
  - K-means clustering
  - K-medoids clustering
  - EM (expectation maximization) clustering
- Hierarchical clustering
  - Divisive clustering (top down)
  - Agglomerative clustering (bottom up)
- Density-Based Methods
  - Regions of dense points separated by sparser regions of relatively low density

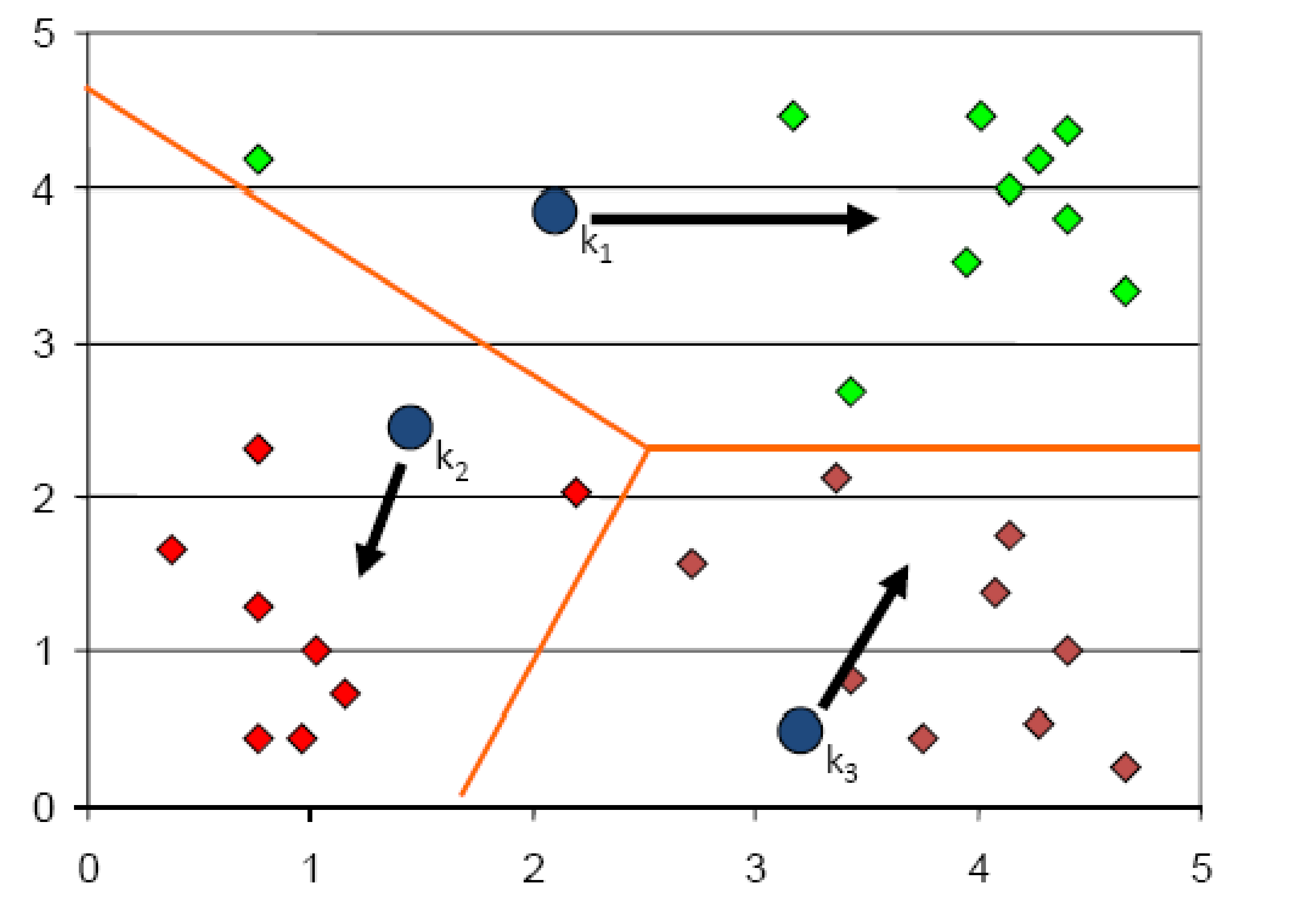
# K-Means

1. Decide on a value for *k.*
2. Initialize the *k cluster centers (randomly, if* necessary).
3. Decide the class memberships of the *N objects by* assigning them to the nearest cluster center.
4. Re-estimate the *k cluster centers, by assuming the* memberships found above are correct.
5. If none of the *N objects changed membership in* the last iteration, exit. Otherwise goto 3.

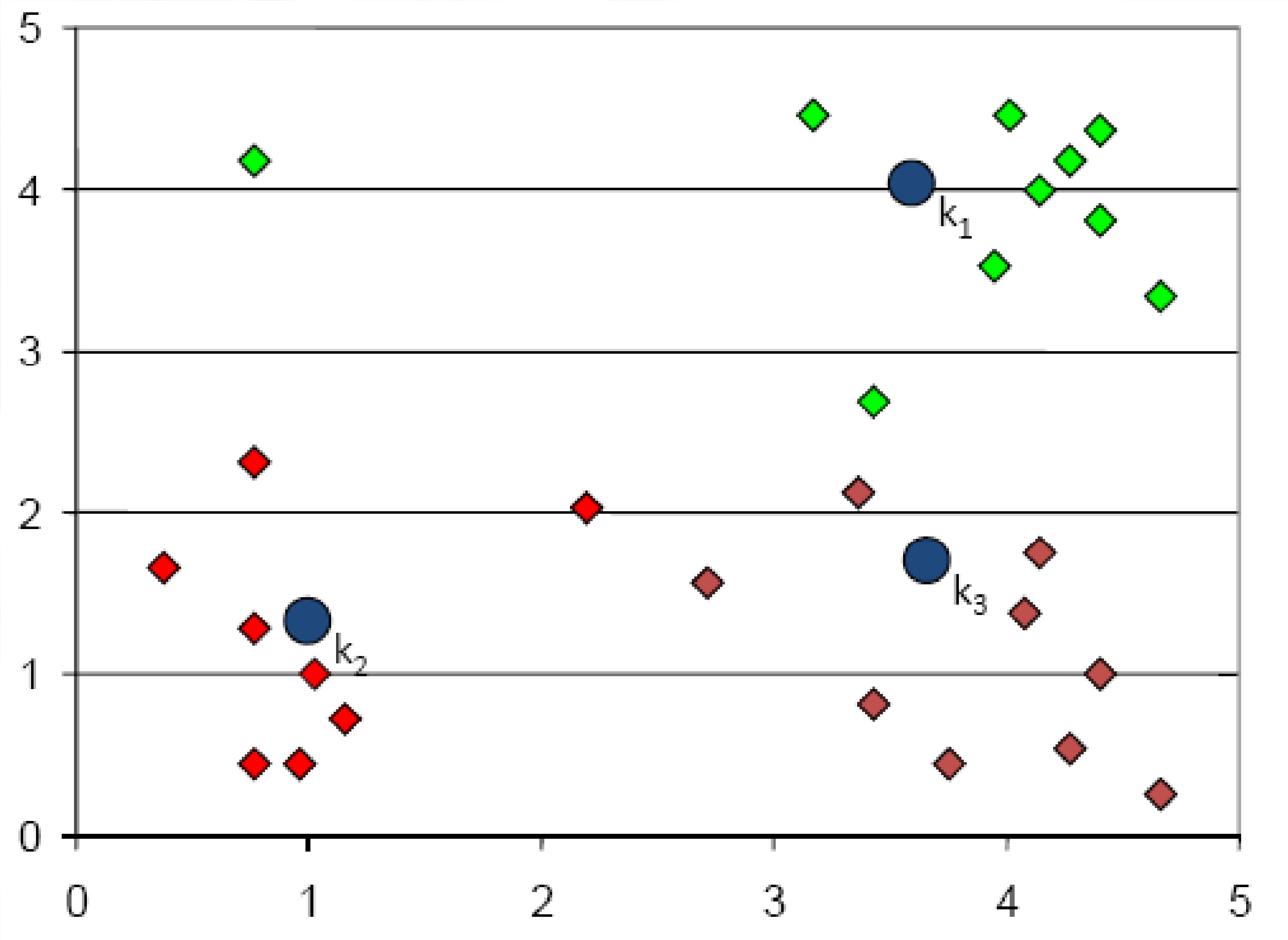
# K-Means: Step 1



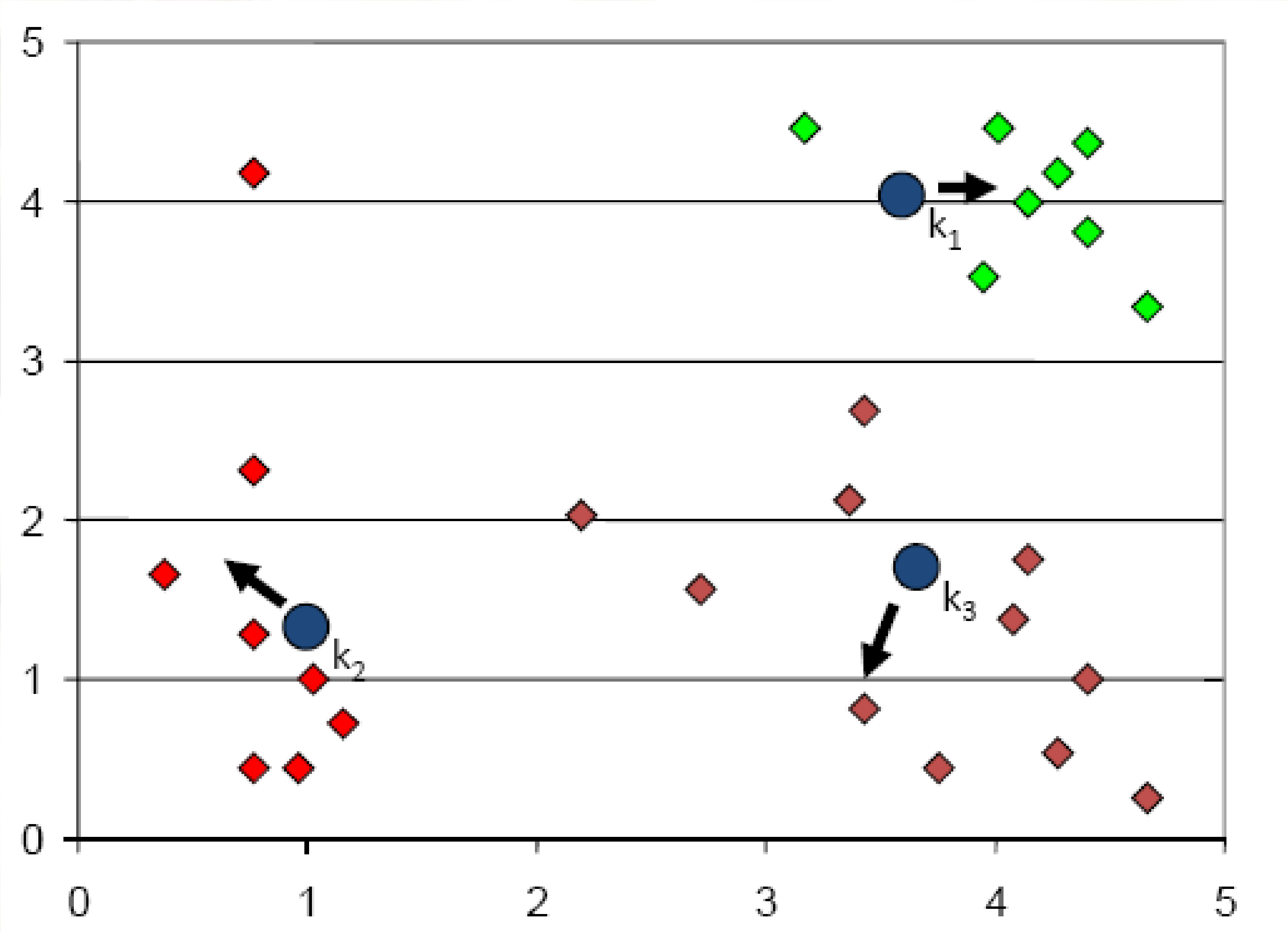
# K-Means: Step 2



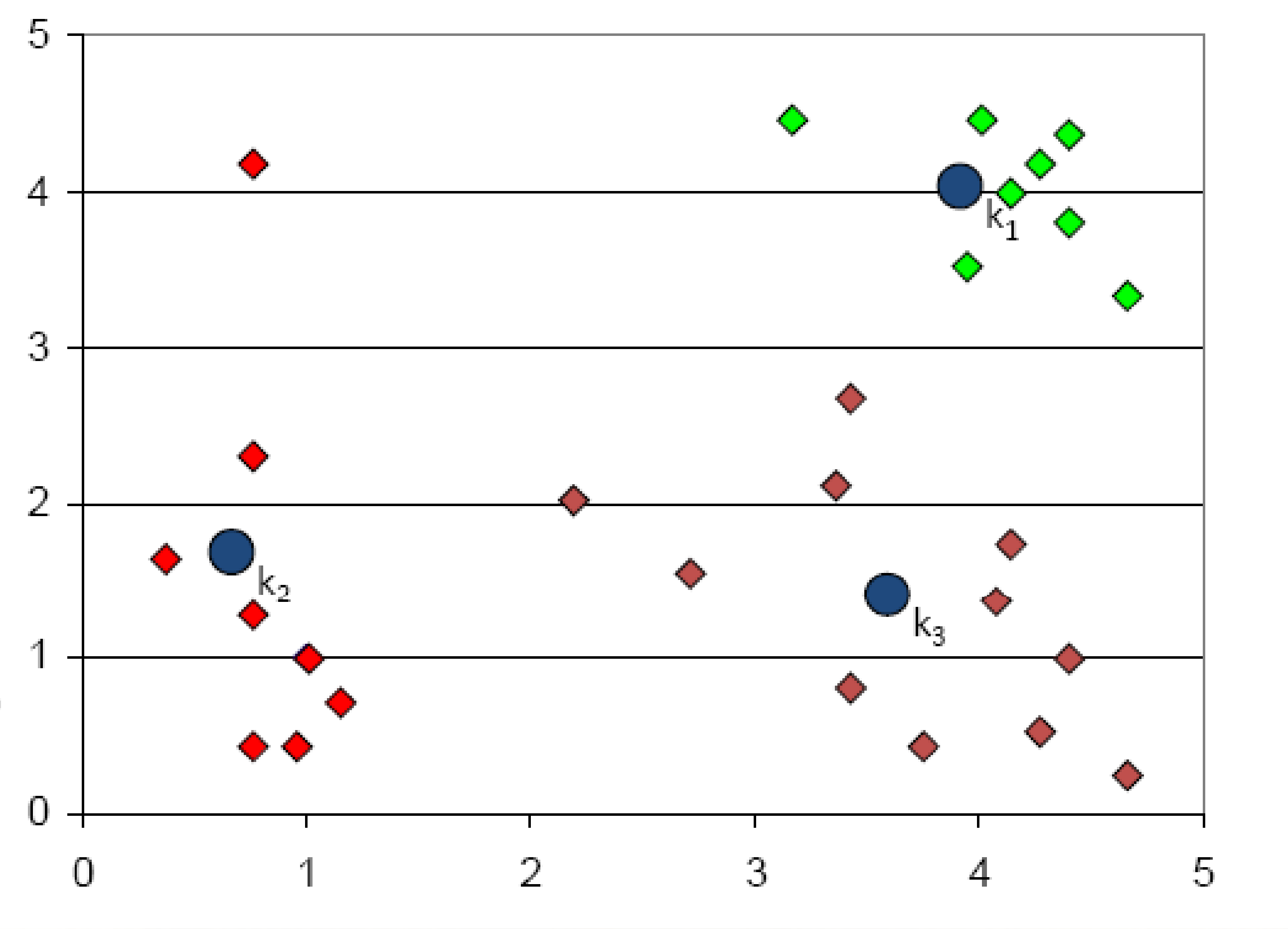
# K-Means: Step 3



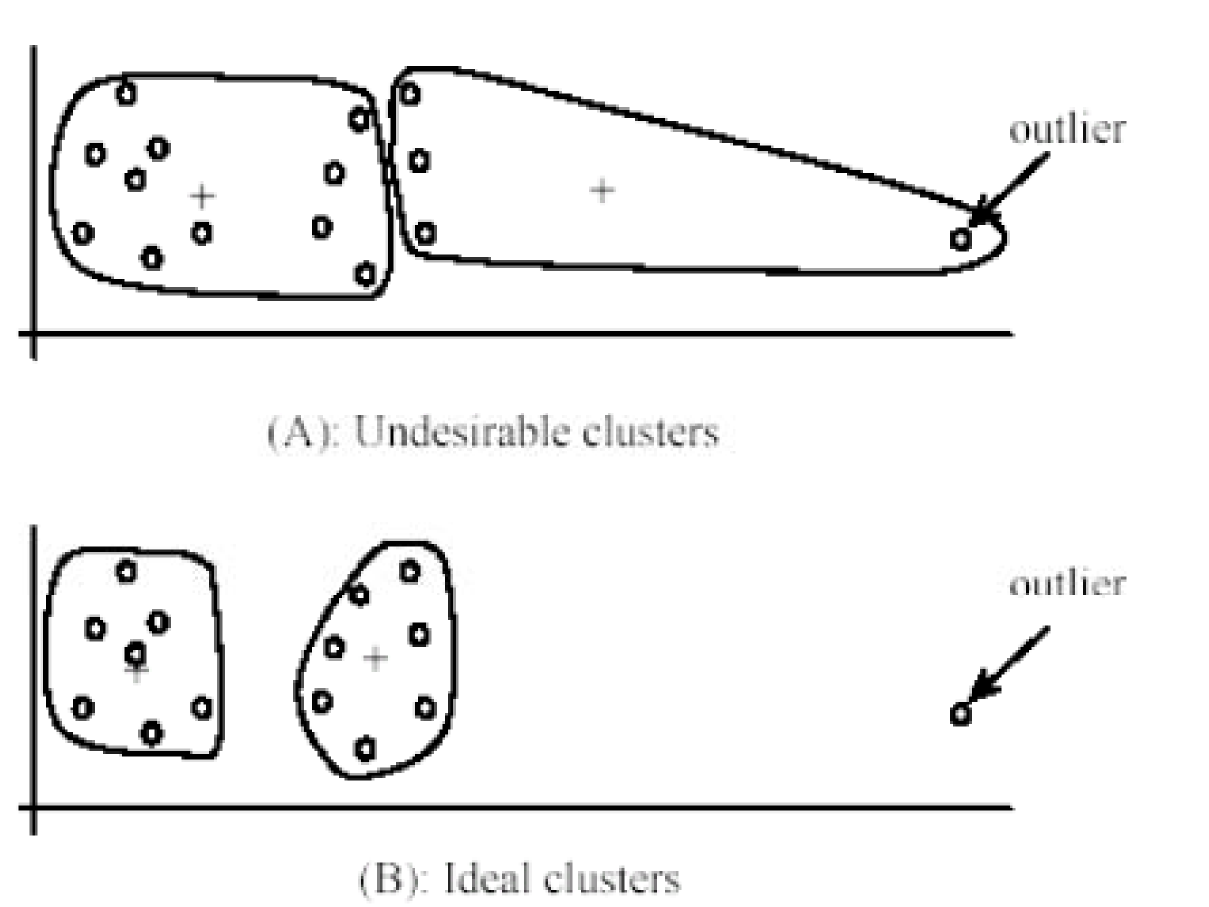
# K-Means: Step 4



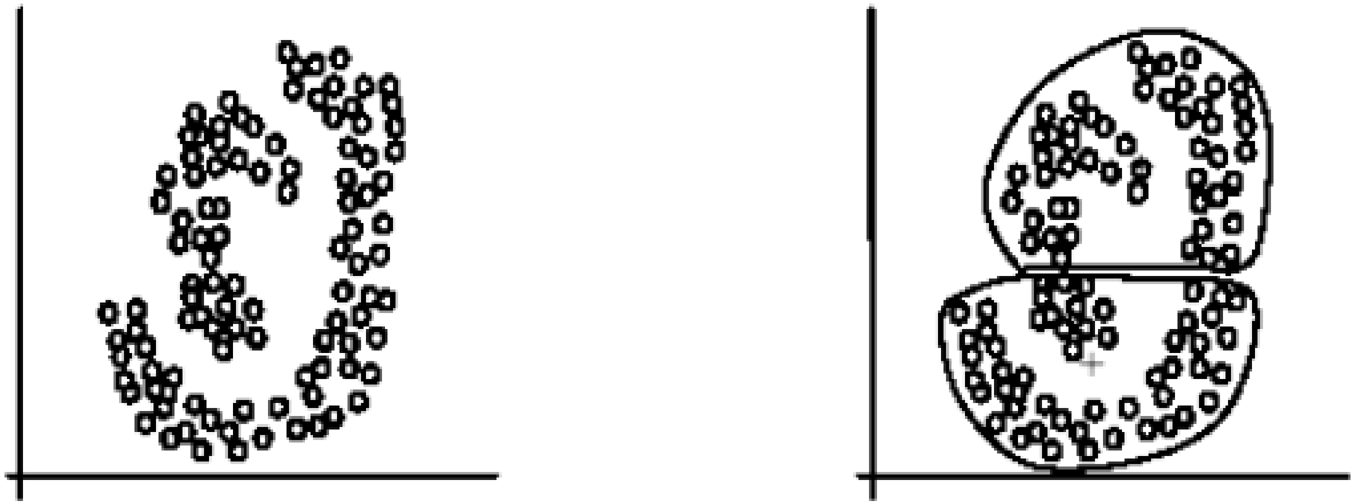
# K-Means: Step 5



# K-Means is sensitive to outliers



# K-Means and complex clusters



(A): Two natural clusters　　(B): *k*-means clusters

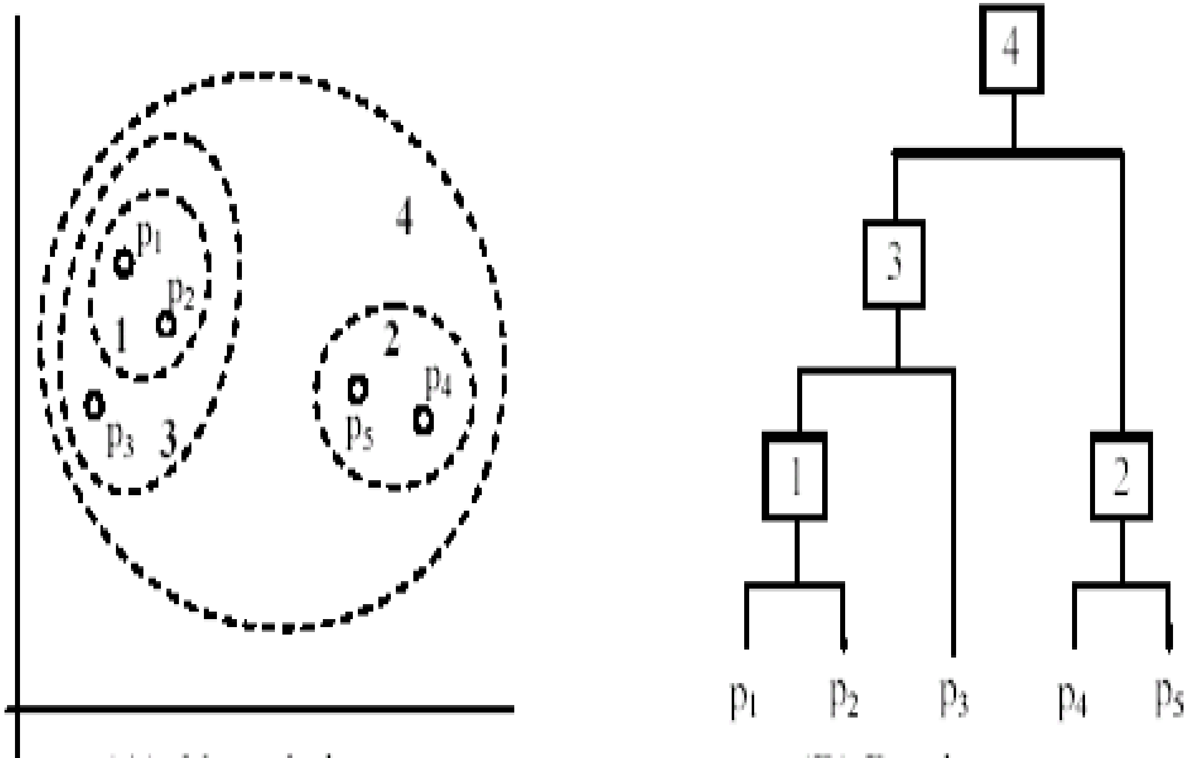
# K-Means: Summary

- Despite its weaknesses, *k-means is still the most popular* algorithm due to its simplicity and efficiency
- Other clustering algorithms have also their own weaknesses
  - No clear evidence that any other clustering algorithm performs better than *k-means in general*
- Some clustering algorithms may be more suitable for some specific types of dataset, or for some specific application problems, than the others
  - Comparing the performance of different clustering algorithms is a difficult task
- No one knows the correct clusters!

# Hierarchical clustering

- Hierarchical agglomerative (bottom-up) clustering builds the dendrogram from the bottom level
- The algorithm
  - At the beginning, each instance forms a cluster (also called a node)
  - Merge *the most similar (nearest) pair of clusters*
    - i.e., The pair of clusters that have *the least distance among all* the possible pairs
  - Continue the merging process
  - Stop when all the instances are merged into a single cluster (i.e., the *root cluster)*

# Example

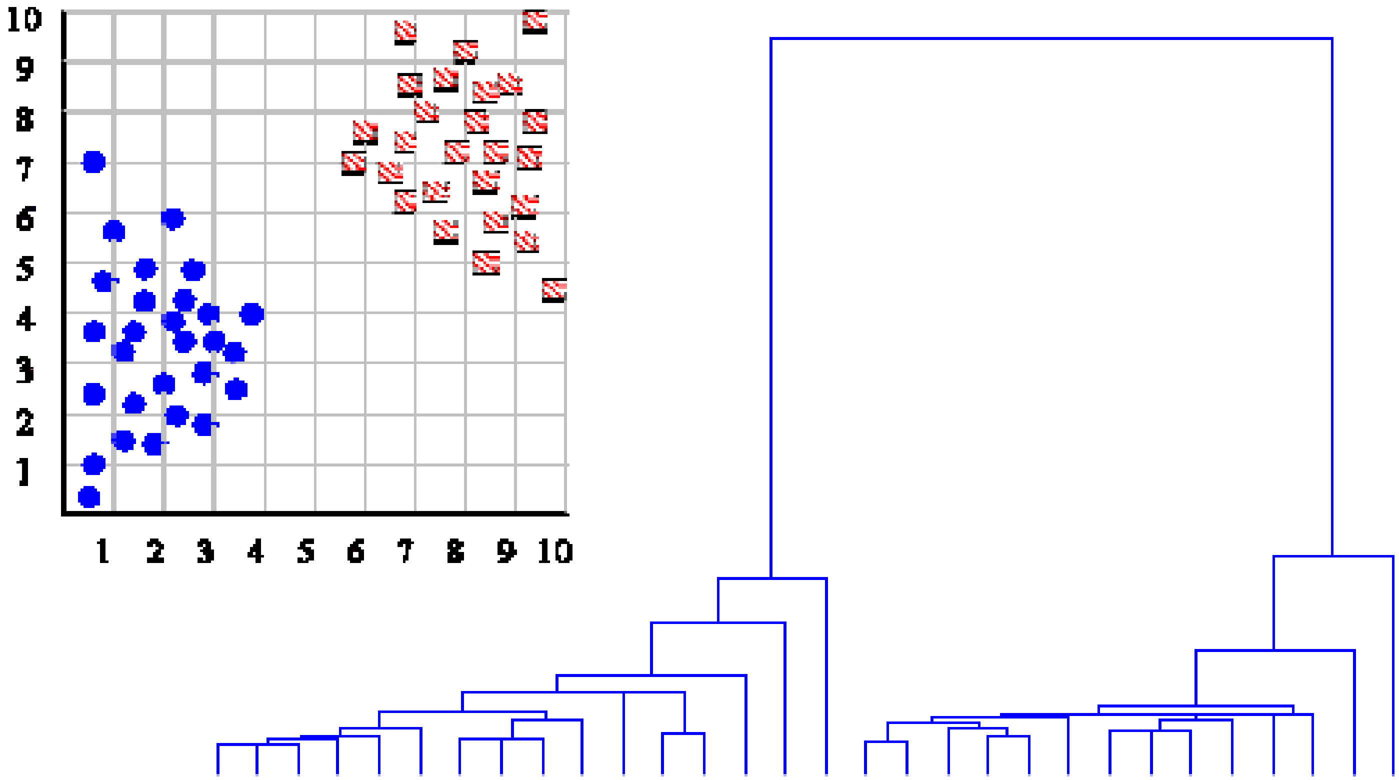



(A). Nested clusters
(Venn diagram)

(B) Dendrogram

# Determining the number of clusters

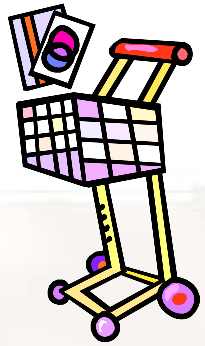
2 highly separated subtrees => 2 clusters



# Hierarchical clustering: summary

- No need to specify the number of clusters in advance.
- Hierarchal nature maps nicely onto human intuition for some domains
- They do not scale well: time complexity of at least O($n^2$), *where n is the number of total objects.*
- Like any heuristic search algorithms, local optima are a problem.
- Interpretation of results is (very) subjective

# Market Basket Analysis



- Retail – each customer purchases different set of products, different quantities, different times
- MBA uses this information to:
  - Identify who customers are (not by name)
  - Understand why they make certain purchases
  - Gain insight about its merchandise (products):
    - Fast and slow movers
    - Products which are purchased together
    - Products which might benefit from promotion
  - Take action:
    - Store layouts
    - Which products to put on specials, promote, coupons…
- Combining all of this with a customer loyalty card it becomes even more valuable

# Nappies and beer

http://www.daedalus.es/en/data-mining/nappies-and-beer/

# How Good is an Association Rule?

| Customer | Items Purchased |
|---|---|
| 1 | OJ, soda |
| 2 | Milk, OJ, window cleaner |
| 3 | OJ, detergent |
| 4 | OJ, detergent, soda |
| 5 | Window cleaner, soda |

← POS Transactions

Co-occurrence of Products

| | OJ | Window cleaner | Milk | Soda | Detergent |
|---|---|---|---|---|---|
| OJ | 4 | 1 | 1 | 2 | 2 |
| Window cleaner | 1 | 2 | 1 | 1 | 0 |
| Milk | 1 | 1 | 1 | 0 | 0 |
| Soda | 2 | 1 | 0 | 3 | 1 |
| Detergent | 2 | 0 | 0 | 1 | 2 |

# How Good is an Association Rule?

| | OJ | Window cleaner | Milk | Soda | Detergent |
|---|---|---|---|---|---|
| OJ | 4 | 1 | 1 | 2 | 2 |
| Window cleaner | 1 | 2 | 1 | 1 | 0 |
| Milk | 1 | 1 | 1 | 0 | 0 |
| Soda | 2 | 1 | 0 | 3 | 1 |
| Detergent | 2 | 0 | 0 | 1 | 2 |

Simple patterns:

1. OJ and soda are more likely purchased together than other two items
2. Detergent is never purchased with milk or window cleaner
3. Milk is never purchased with soda or detergent

But, what about 3 (or more) items combinations?

# Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

| *TID* | *Items* |
|---|---|
| **1** | **Bread, Milk** |
| **2** | **Bread, Diaper, Beer, Eggs** |
| **3** | **Milk, Diaper, Beer, Coke** |
| **4** | **Bread, Milk, Diaper, Beer** |
| **5** | **Bread, Milk, Diaper, Coke** |

Example of Association Rules

{Diaper} $\rightarrow$ {Beer},
{Milk, Bread} $\rightarrow$ {Eggs,Coke},
{Beer, Bread} $\rightarrow$ {Milk},

Implication means co-occurrence, not causality!

# Definition: Frequent Itemset

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

| *TID* | *Items* |
|---|---|
| 1 | Bread, Milk |
| 2 | Bread, Diaper, Beer, Eggs |
| 3 | Milk, Diaper, Beer, Coke |
| 4 | Bread, Milk, Diaper, Beer |
| 5 | Bread, Milk, Diaper, Coke |

# Definition: Association Rule

- Association Rule
  - An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
  - Example:
    $\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}$

- Rule Evaluation Metrics
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

| *TID* | *Items* |
|---|---|
| **1** | **Bread, Milk** |
| **2** | **Bread, Diaper, Beer, Eggs** |
| **3** | **Milk, Diaper, Beer, Coke** |
| **4** | **Bread, Milk, Diaper, Beer** |
| **5** | **Bread, Milk, Diaper, Coke** |

Example:

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

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

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

# Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

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

  $\Rightarrow$ Computationally prohibitive!

# Mining Association Rules

| TID | Items |
|---|---|
| 1 | Bread, Milk |
| 2 | Bread, Diaper, Beer, Eggs |
| 3 | Milk, Diaper, Beer, Coke |
| 4 | Bread, Milk, Diaper, Beer |
| 5 | Bread, Milk, Diaper, Coke |

## Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

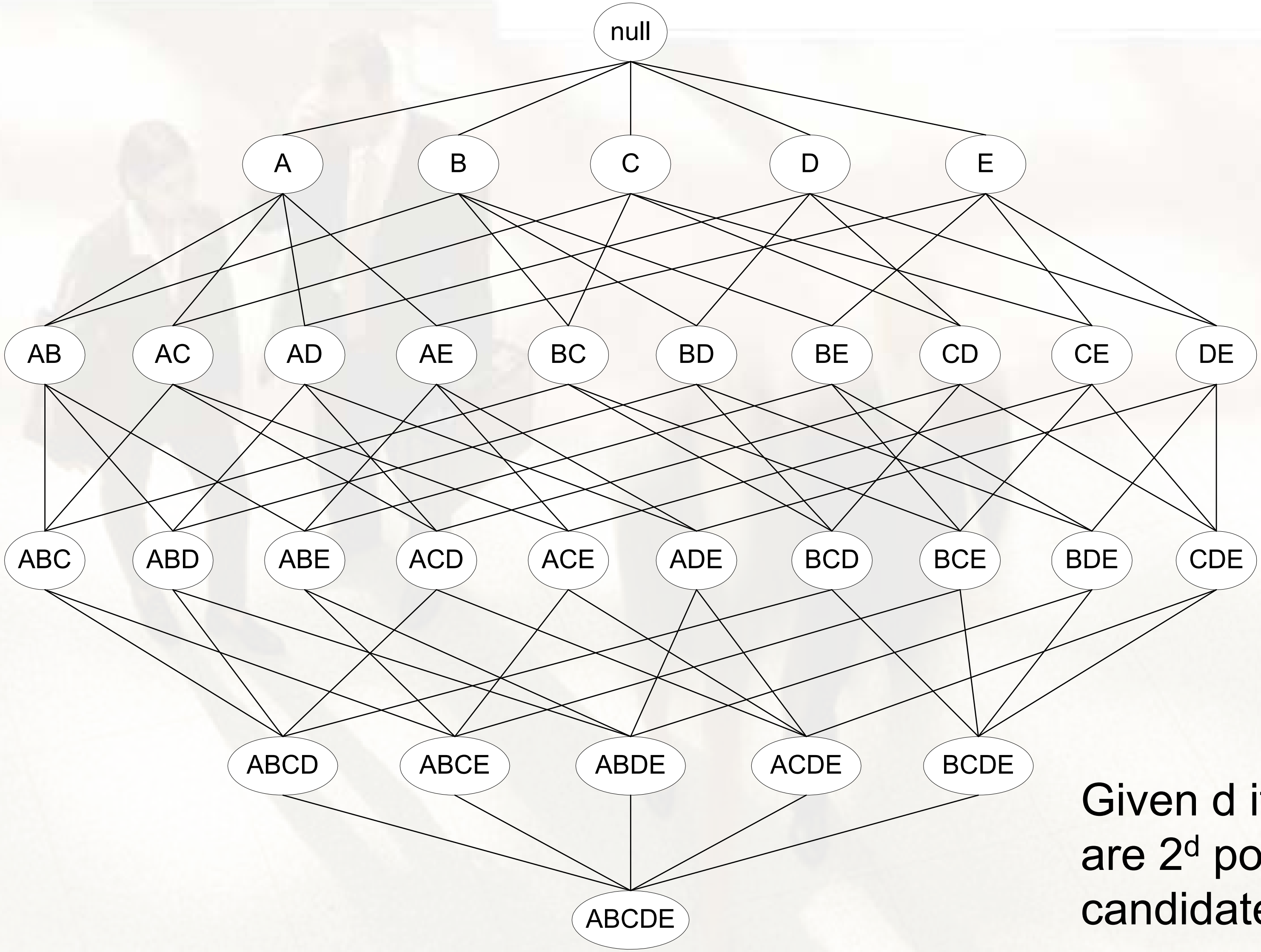
## Observations:

- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

# Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support ≥ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

# Frequent Itemset Generation




Given d items, there are $2^d$ possible candidate itemsets

# Reducing Number of Candidates

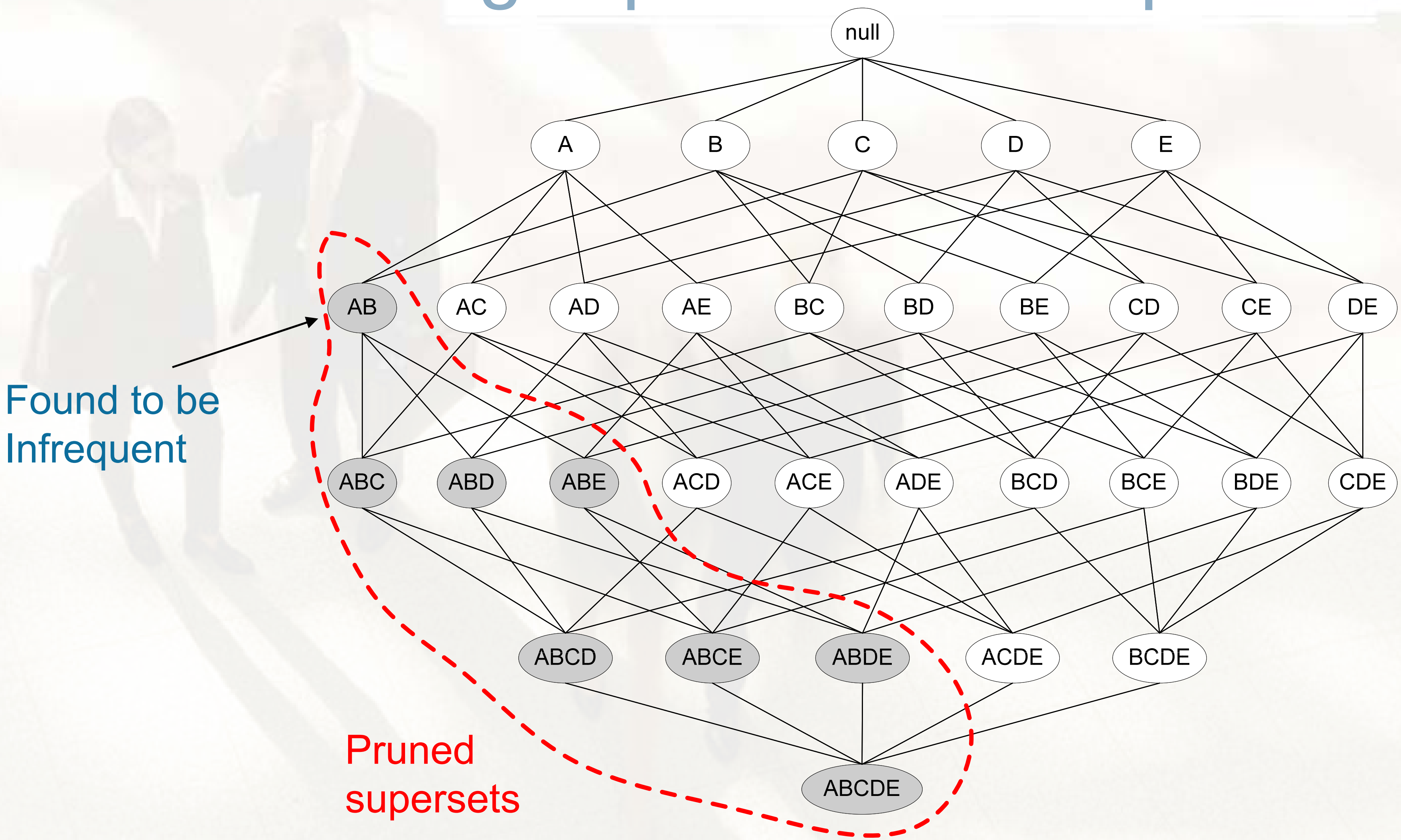
- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent

- Apriori principle holds due to the following property of the support measure:

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

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

# Illustrating Apriori Principle



# Apriori Algorithm

- Method:
  - Let k=1
  - Generate frequent itemsets of length 1
  - Repeat until no new frequent itemsets are identified
    - Generate length (k+1) candidate itemsets from length k frequent itemsets
    - Prune candidate itemsets containing subsets of length k that are infrequent
    - Count the support of each candidate by scanning the DB
    - Eliminate candidates that are infrequent, leaving only those that are frequent

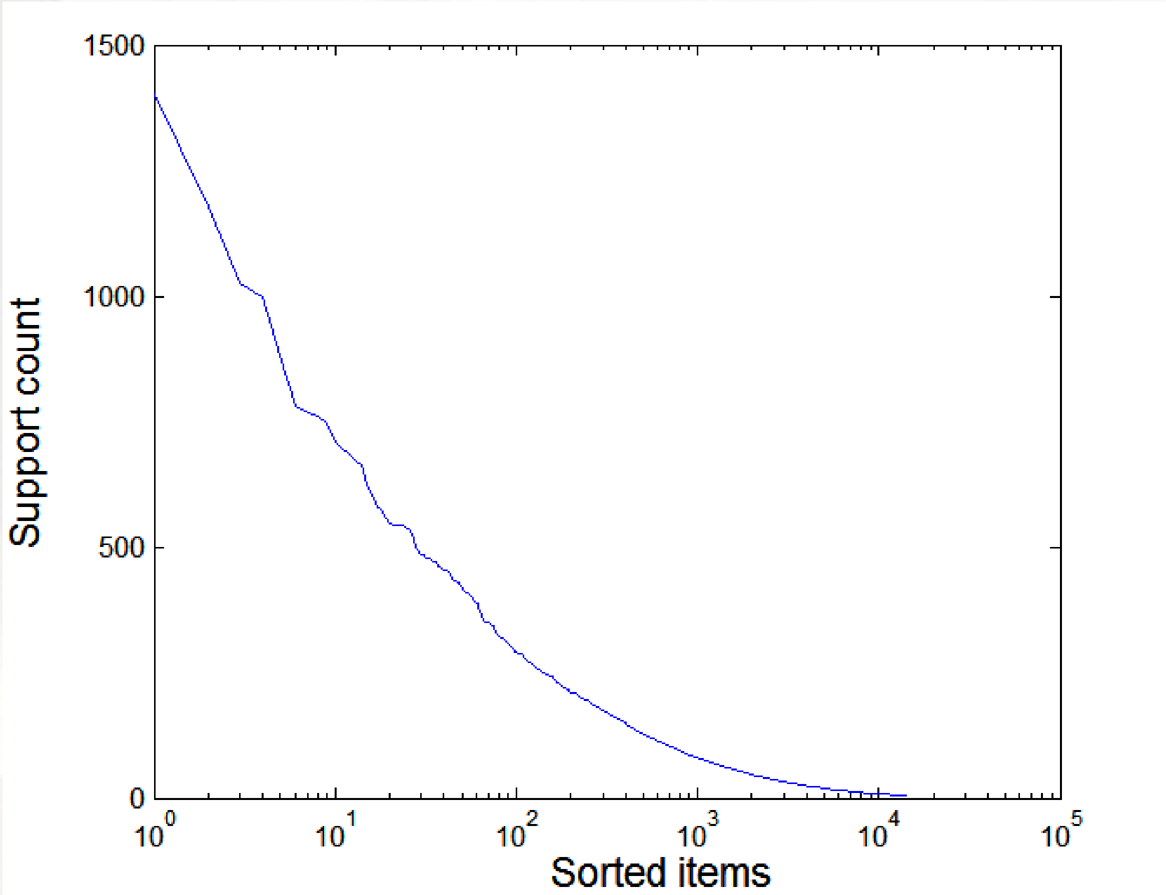
# Factors Affecting Complexity

- Choice of minimum support threshold
  - lowering support threshold results in more frequent itemsets
  - this may increase number of candidates and max length of frequent itemsets
- Dimensionality (number of items) of the data set
  - more space is needed to store support count of each item
  - if number of frequent items also increases, both computation and I/O costs may also increase
- Size of database
  - since Apriori makes multiple passes, run time of algorithm may increase with number of transactions
- Average transaction width
  - transaction width increases with denser data sets
  - This may increase max length of frequent itemsets and traversals of hash tree (number of subsets in a transaction increases with its width)

# Effect of Support Distribution

- Many real data sets have skewed support distribution

Support distribution of a retail data set



# Effect of Support Distribution

- How to set the appropriate *minsup* threshold?
  - If *minsup* is set too high, we could miss itemsets involving interesting rare items (e.g., expensive products)
  - If *minsup* is set too low, it is computationally expensive and the number of itemsets is very large
- Using a single minimum support threshold may not be effective