Advanced clustering

Jan Sedmidubský Masaryk University

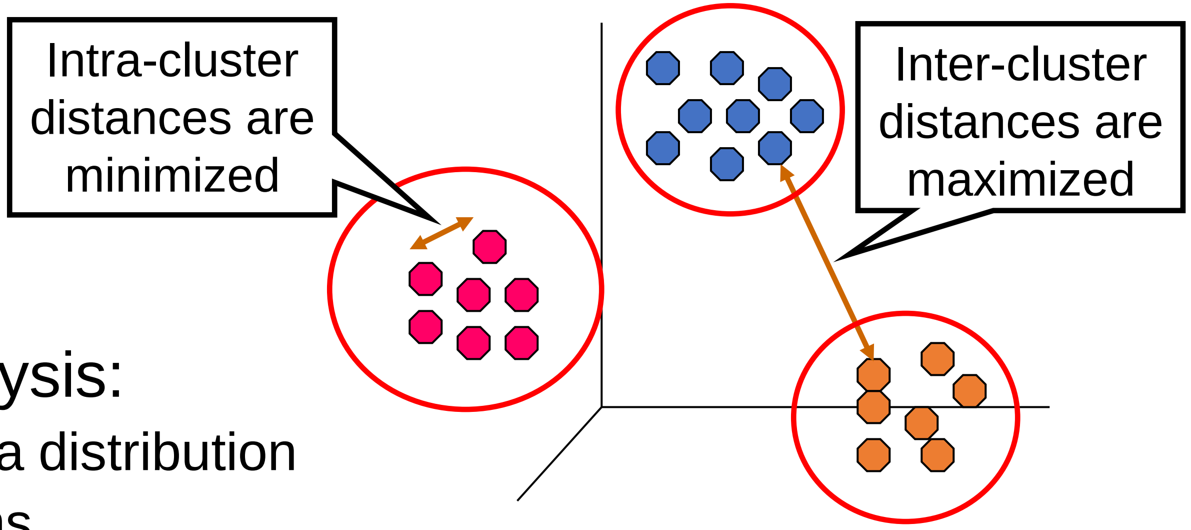
sedmidubsky@mail.muni.cz

Outline

- Basics of clustering
- Clustering algorithms:
  - k-means
  - Agglomerative clustering
  - DBSCAN
  - Chameleon
  - Jarvis-Patrick clustering
  - SNN Density-based Clustering

What is clustering?

- Cluster analysis (clustering, segmentation, quantization, …)
  - Given a set of data objects, partition them into a set of groups (i.e., clusters) such that the objects are:
    - Similar (or related) to one another within the same group (i.e., cluster) and
    - Dissimilar (or unrelated) to the objects in other groups (i.e., clusters)
  - Unsupervised learning (i.e., no predefined classes), in contrast to classification

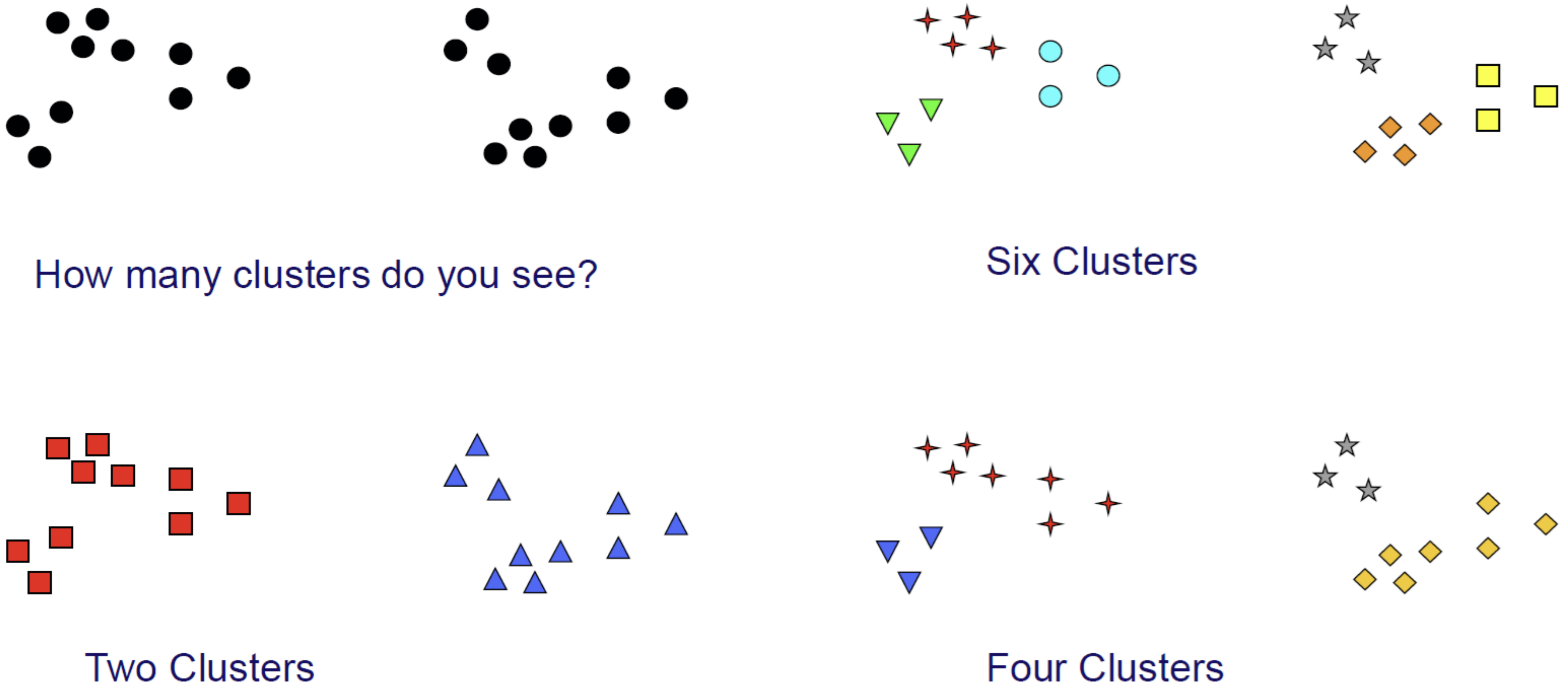


- Clustering:
  - Core task of data mining
  - Typical ways to use/apply cluster analysis:
    - As a stand-alone tool to get insight into data distribution
    - As a preprocessing step for other algorithms

Applications

- Generating a compact summary of data for classification, pattern discovery, data indexing, outlier detection, etc.
  - Outliers are objects “far away” from any cluster
- Data compression and reduction
  - Image processing – vector quantization
- Analysis of multimedia, biological, or social-network data
  - Clustering images or video/audio clips, gene/protein sequences, etc.
- Collaborative filtering, recommendation systems, customer segment.
  - Find like-minded users or similar products
- Dynamic trend detection
  - Clustering stream data and detecting trends and patterns

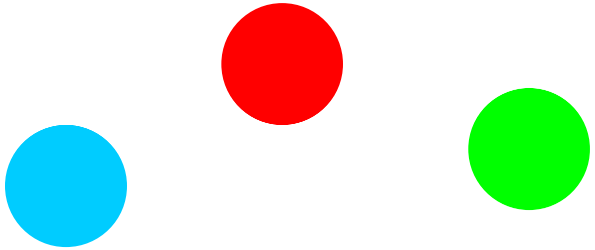
The notion of a cluster is ambiguous



- The usefulness of a clustering depends on the goal of the analysis

Cluster types

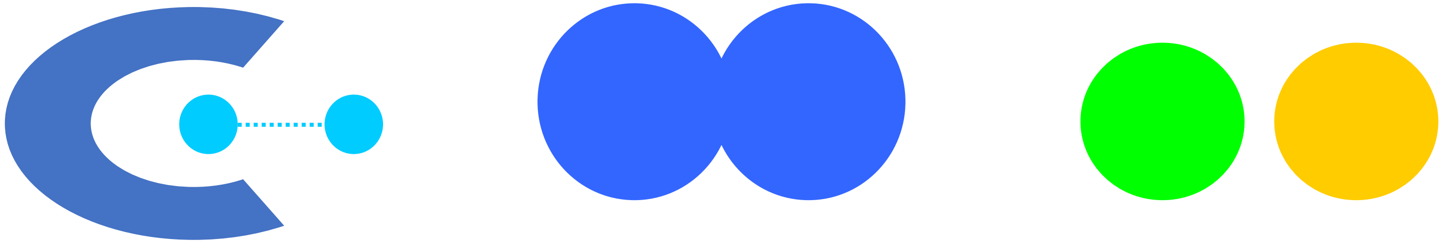
- Well-separated – any object in a cluster is closer (more similar) to every other object in the cluster than to any object outside the cluster



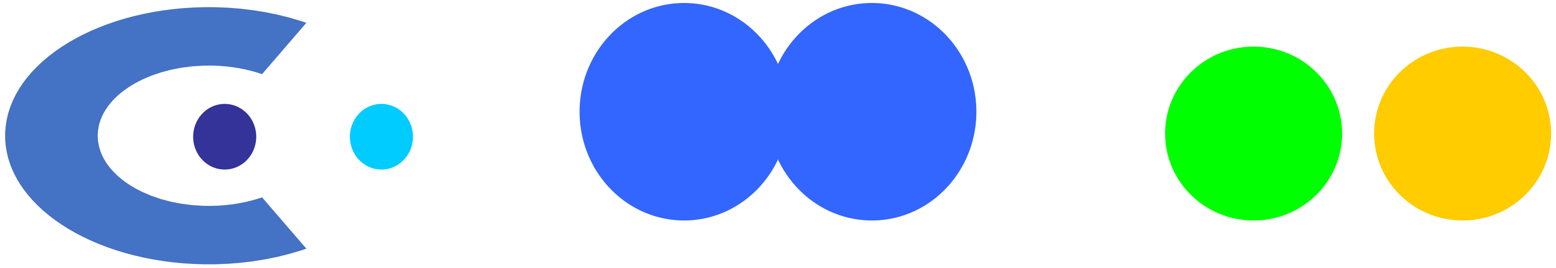
- Prototype-based – an object in a cluster is closer to the center of the cluster than to the center of any other cluster
  - Center – often a centroid (the average of all the objects in the cluster), or a medoid (the most "representative" object of a cluster)

Cluster types

- Contiguous cluster (nearest neighbor) – any object in a cluster is closer to one or more other objects in the cluster than to any object in a different cluster



- Density-based – a cluster is a dense region of objects, which is separated by low-density regions, from other regions of high density
  - Used when clusters are irregular or intertwined, and in the presence of noise and outliers



Clustering methodologies

- Distance-based methods
  - Partitioning algorithms – k-means, BFR
  - Hierarchical algorithms – agglomerative vs. divisive methods
- Density-based methods
  - Data space is explored at a high-level of granularity and then post-processed to put together dense regions into an arbitrary shape
- Graph-based methods
  - Construct a graph of datapoints and form clusters based on edge connectivity
- Grid-based methods
  - Divide data space into a grid-like structure and perform clustering on grid cells
- Probabilistic methods
  - Modeling data from a generative process (e.g., mixture of Gaussians) and estimating the generative probability of the underlying data points

Similarity/dissimilarity

- A proximity (similarity, or dissimilarity) measure
  - Numerical measure of how similar/different two datapoints are
    - Dissimilarity ~ inverse of similarity
  - Usually quantified by a distance function → dissimilarity
    - Minimum dissimilarity is 0 (i.e., completely similar)
    - Range [0, 1] or [0, ∞), i.e., similarity decreases with an increasing distance

- There are many similarity measures for different applications
  - Commonly used, e.g., Euclidean, Cosine, or Manhattan (city block) distances
  - Selection depends on data characteristics and a target application
    - Data characteristics – e.g., dimensionality (sparseness), distribution

Properties

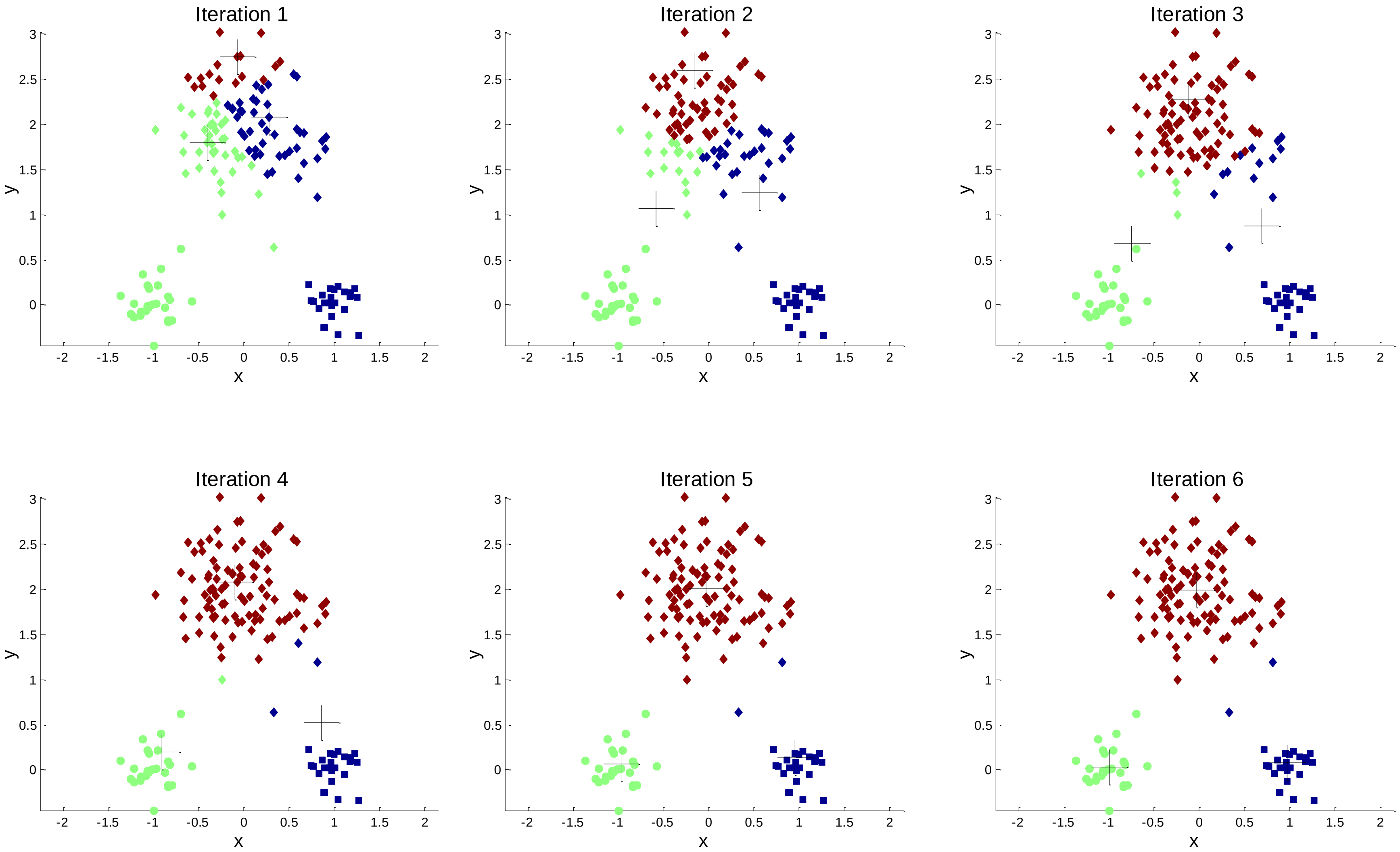
- Considerations:
  - Partitioning criteria – single level vs. hierarchical partitioning (e.g., grouping topical terms)
  - Separation of clusters – exclusive (e.g., one customer belongs to only one region) vs. non-exclusive (e.g., one document may belong to more classes)
  - Similarity of clusters – distance-based (e.g., Euclidean distance) vs. connectivity-based (e.g., density or contiguity)
- Requirements/challenges of clustering algorithms:
  - Abilities to deal with arbitrary shapes of clusters or noisy data
  - Scalability – in terms of dataset size, data dimensionality, different data types (e.g., numerical, multimedia, text), incremental/stream clustering, or sensitivity to input order of data objects
  - Constraint-based clustering – in terms of user-given preferences or constraints, domain knowledge, user queries

Distance-based methods

- Representatives – k-means, BFR (Bradley-Fayyad-Reina), CURE
  - Details in other courses, e.g., PA212
- k-means:
  - Number of clusters, k, must be specified in advance
  - Each cluster is associated with a centroid (center object) – centroid/medoid
  - Each object is assigned to the cluster with the closest centroid
  - Convergence criterion – minimizing the SSE (Sum of Squared Error) function
    - $SSE = \sum_{i=1}^{k} \sum_{x \in C_i} dist^2(m_i, x)$
    - x is a data point in cluster C_i and m_i is the centroid (mean) for cluster C_i

```
select k points as initial centroids
repeat
  form k clusters by assigning each point to its closest centroid
  re-compute the centroids (i.e., mean point) of each cluster
until convergence criterion is satisfied
```

k-means example

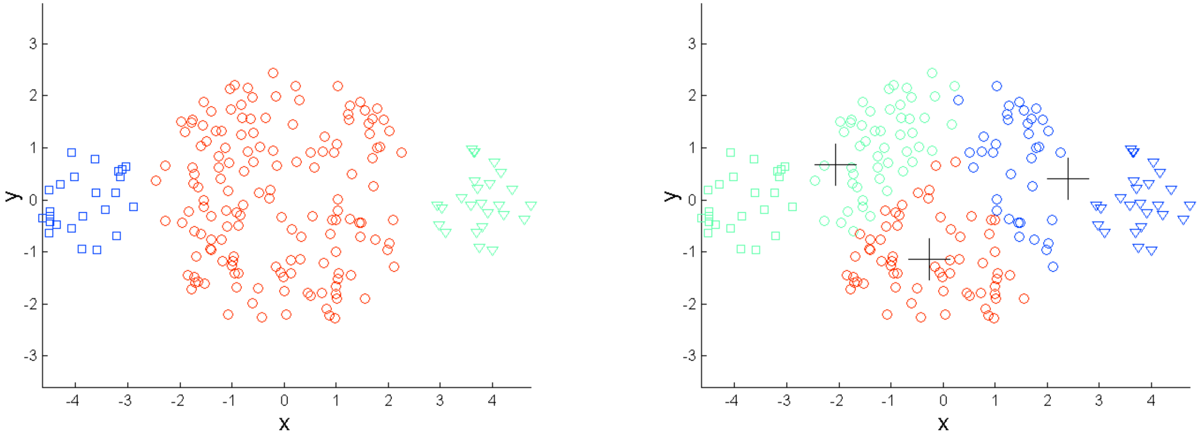


k-means

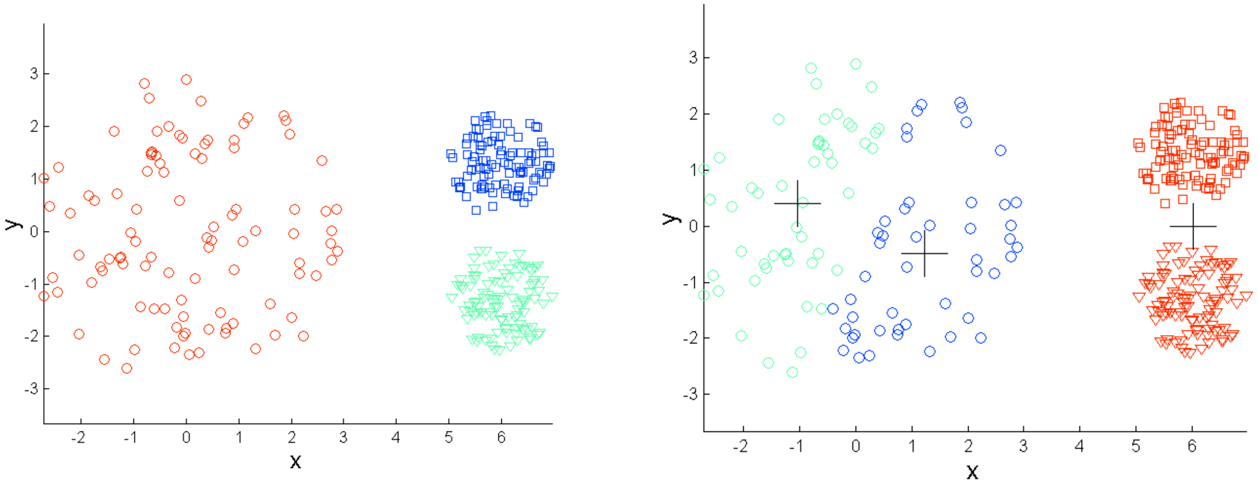
- Complexity:
  - $O(t \cdot k \cdot n)$, where n is # of objects, k is # of clusters, and t is # of iterations
  - Typically: $t, k \ll n$ → a quite efficient method
- Limitations:
  - Need to specify k in advance
    - Initialization can be important to find high-quality clusters
  - Problems when clusters are of different sizes or densities
  - Not suitable to discover clusters with non-convex (non-globular) shapes
  - Sensitive to noisy data and outliers

k-means – limitations

- Different sizes



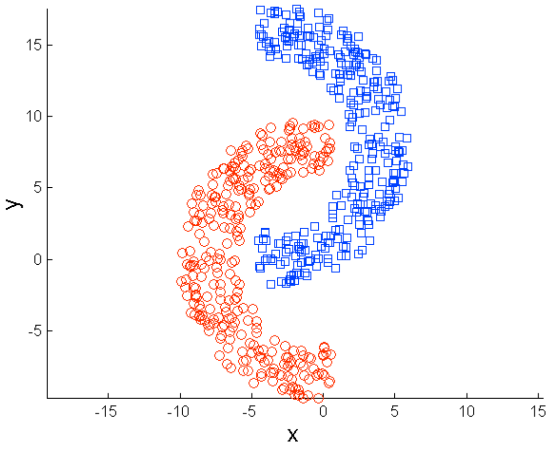
- Different density



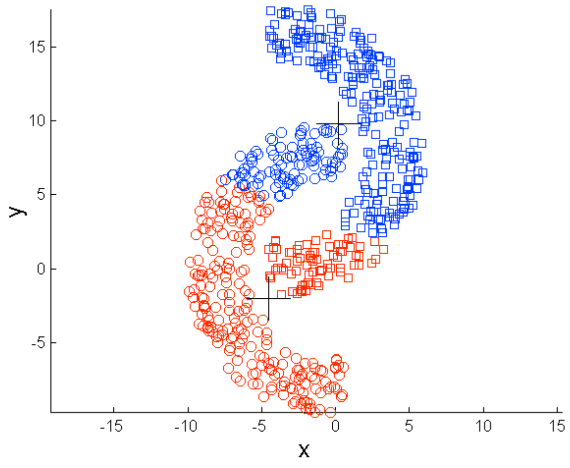
Original points　　　　k-means (3 clusters)

k-means – limitations

- Non-globular shapes



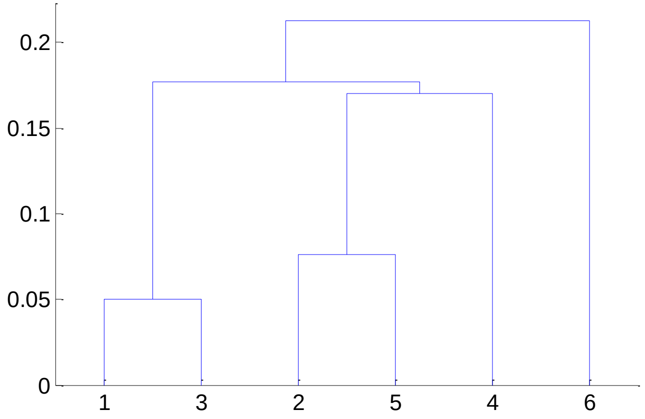
Original points

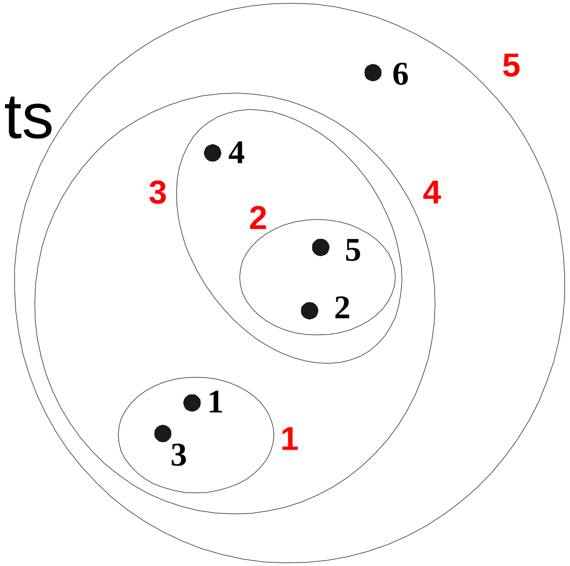


k-means (2 clusters)

Hierarchical clustering

- Produces a set of nested clusters organized as a hierarchical tree
  - Agglomerative approach:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or *k* clusters) left
  - Divisive approach:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a single point (or there are *k* clusters)
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges/splits



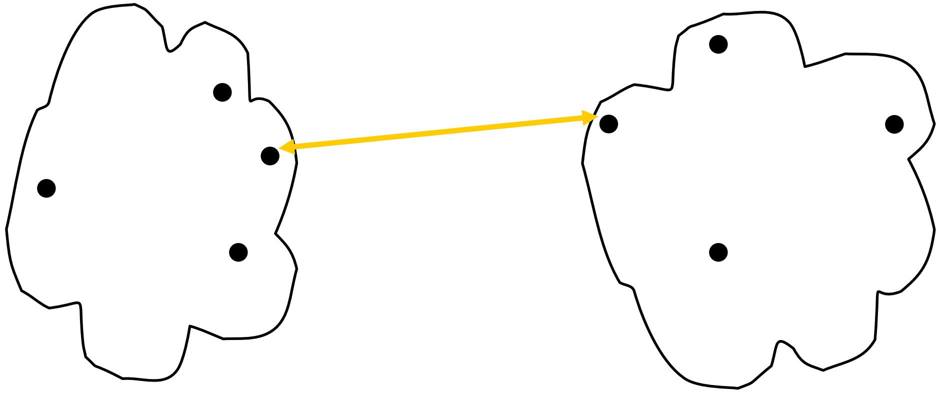


Hierarchical clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by “cutting” the dendrogram at the proper level

- Key operation is the computation of the proximity of two clusters
  - Different approaches to define the distance between clusters
    - MIN
    - MAX
    - Group average
    - Distance between centroids

Hierarchical clustering

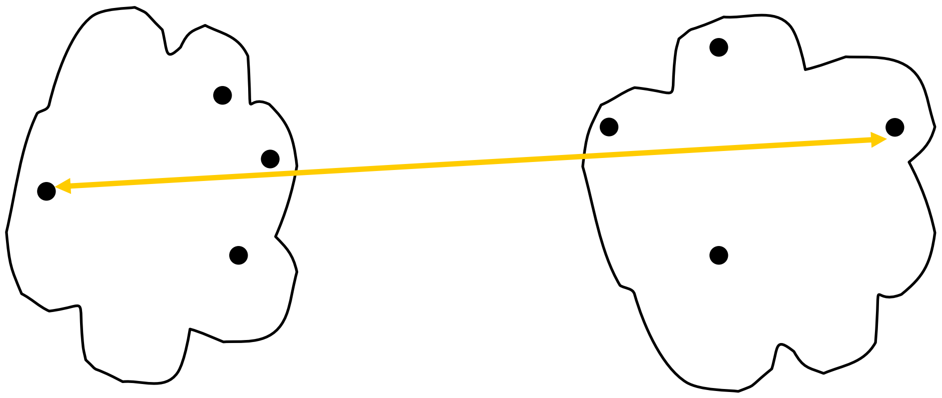


- Proximity of clusters based on:
  - **MIN**
  - MAX
  - Group average
  - Distance between centroids

- MIN:
  - Strengths – can handle non-elliptical shapes
  - Limitations – sensitive to noise

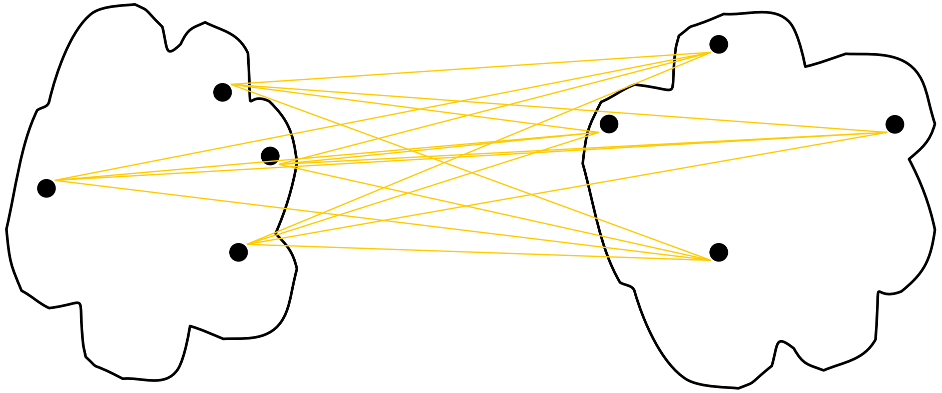
Hierarchical clustering

- Proximity of clusters based on:
  - MIN
  - **MAX**
  - Group average
  - Distance between centroids



- MAX:
  - Strengths – less susceptible to noise/outliers
  - Limitations – tends to break large clusters + biased towards globular clusters

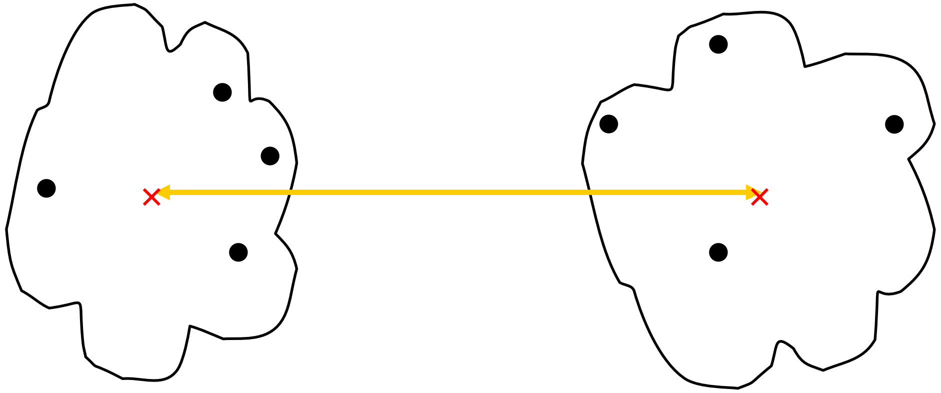
Hierarchical clustering



- Proximity of clusters based on:
  - MIN
  - MAX
  - **Group average**
  - Distance between centroids

- Group average:
  - Strengths – less susceptible to noise/outliers
  - Limitations – biased towards globular clusters

Hierarchical clustering

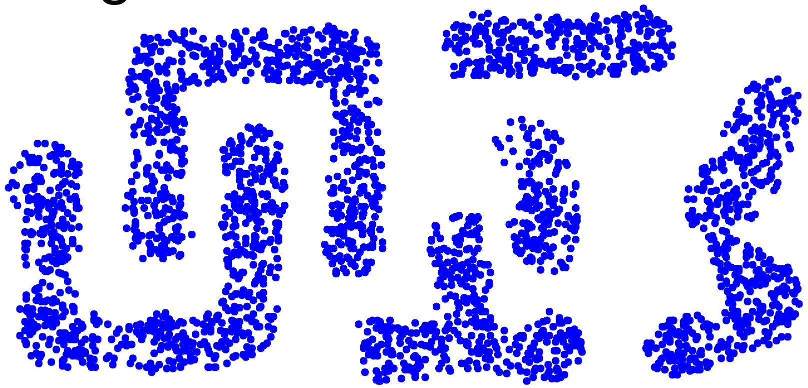


- Proximity of clusters based on:
  - MIN
  - MAX
  - Group average
  - **Distance between centroids**

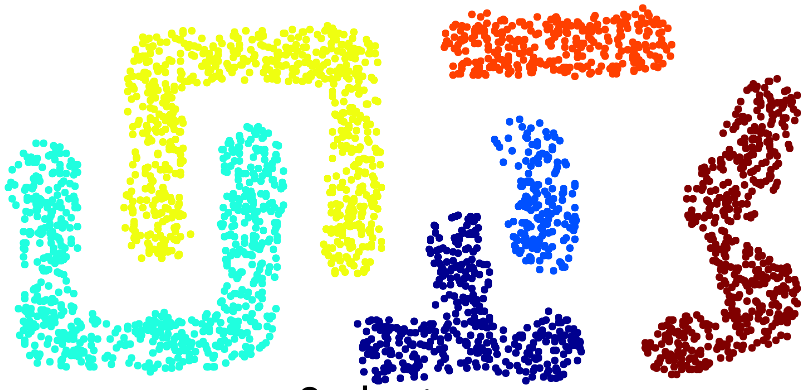
- Distance between centroids:
  - Strengths – very fast, useful for compact, spherical clusters
  - Limitations – not robust to elongated or non-spherical clusters

Hierarchical clustering – MIN

- MIN:
  - Strengths – can handle non-elliptical shapes

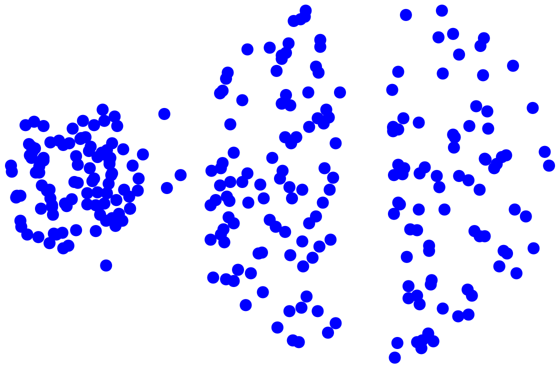


Original points

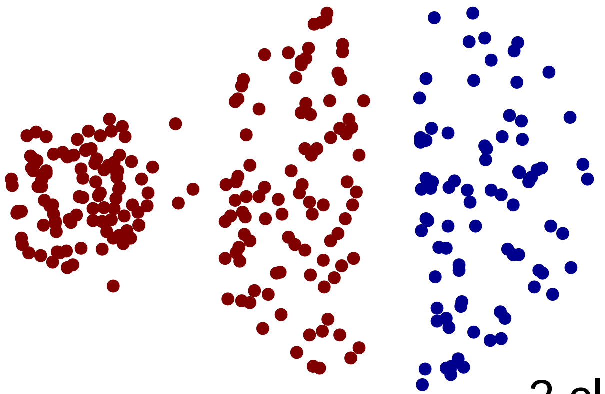


6 clusters

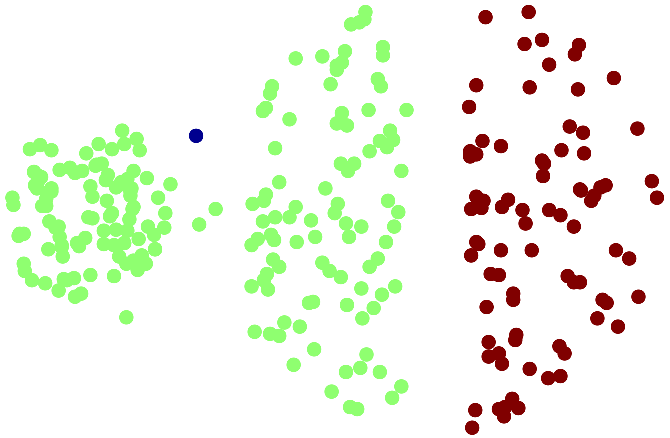
  - Limitations – sensitive to noise



Original points



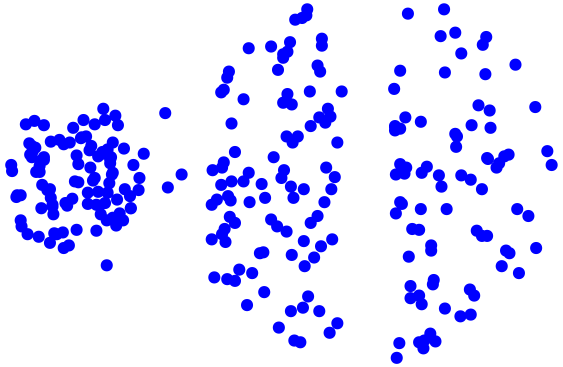
2 clusters



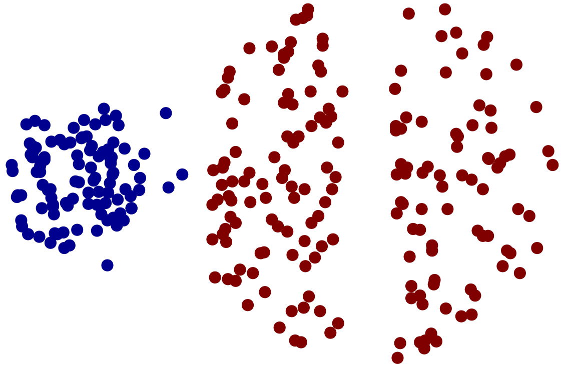
3 clusters

Hierarchical clustering – MAX

- MAX:
  - Strengths – less susceptible to noise

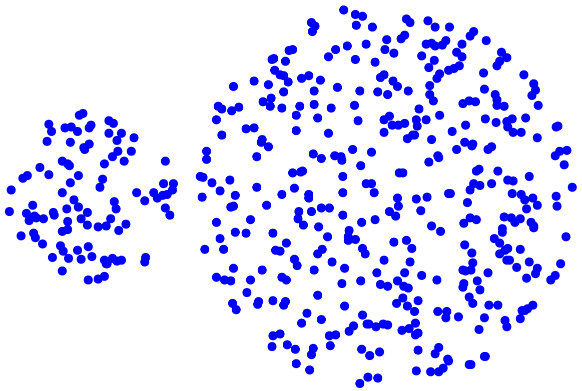


Original points

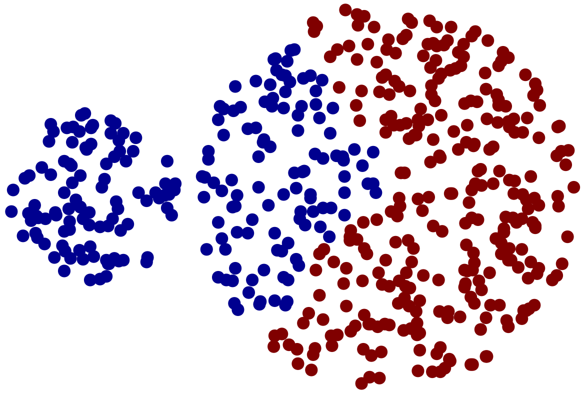


2 clusters

  - Limitations – tends to break large clusters + biased towards globular clusters



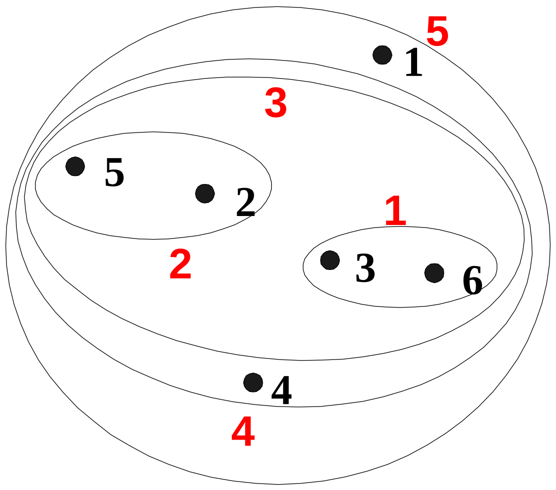
Original points



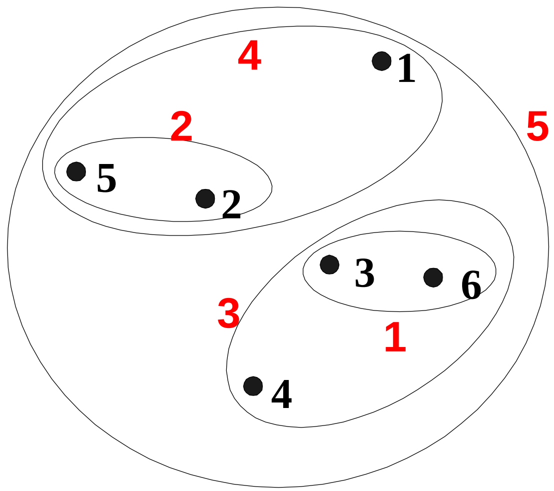
2 clusters

Hierarchical clustering – Group average

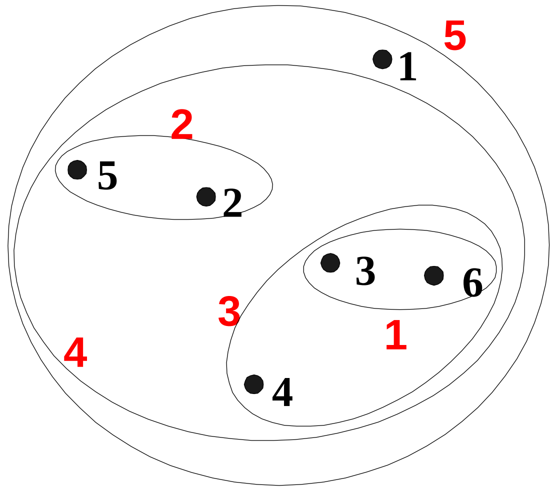
- Group average:
  - Strengths – less susceptible to noise/outliers
  - Limitations – biased towards globular clusters
- Comparison:



MIN



MAX



Group average

Hierarchical clustering

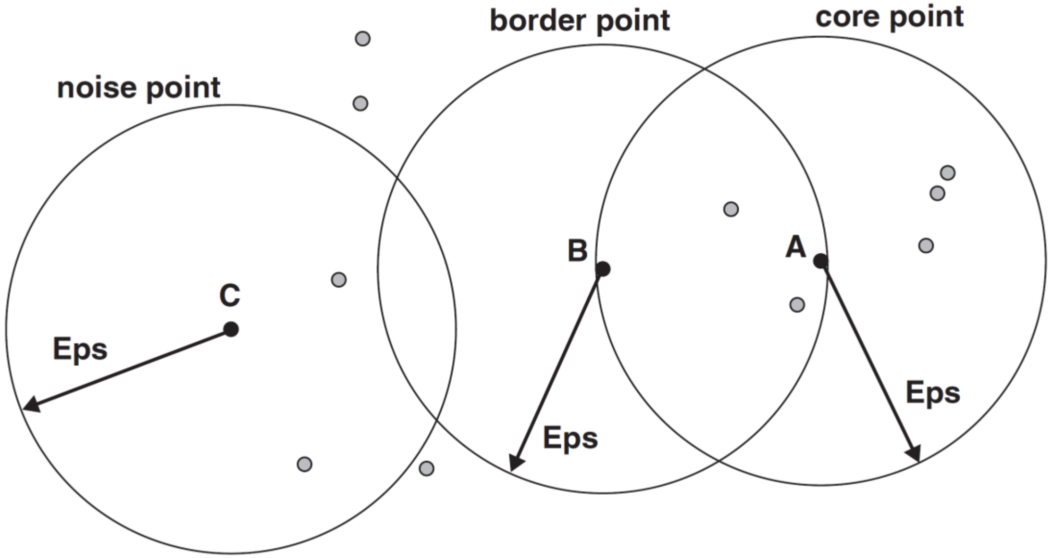
- Complexity:
  - Space: $O(n^2)$ ~ proximity matrix, where n is # of objects
  - Time: $O(n^3)$ in many cases
    - n steps and at each step the proximity matrix (n^2) must be updated
    - Can be reduced to $O(n^2 \cdot \log(n))$ with some cleverness
- Limitations:
  - Once a decision is made to combine two clusters, it cannot be undone
  - No global objective function is directly minimized
  - Different schemes have problems with one or more of the following:
    - Sensitivity to noise
    - Difficulty handling clusters of different sizes and non-globular shapes
    - Breaking large clusters

Density based clustering

- Clusters are regions of high density that are separated from one another by regions of low density

DBSCAN

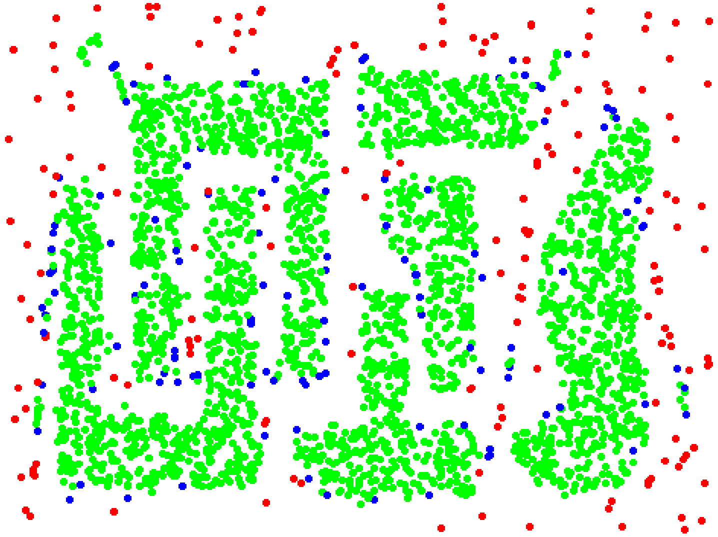
- DBSCAN – a density-based algorithm
  - Density – number of points within a specified radius (*Eps*)
  - A point is a core point if it has at least a specified number of points (*MinPts*) within distance *Eps*
    - These are the points inside a cluster (counting the point itself)
  - A border point is not a core point, but is in the neighborhood of a core point
  - A noise point is any point that is not a core point or a border point



MinPts = 7

DBSCAN

Original points

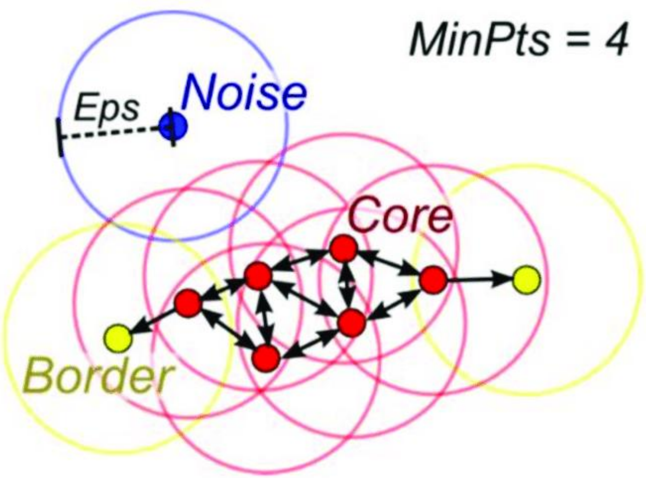


Point types: core, border and noise

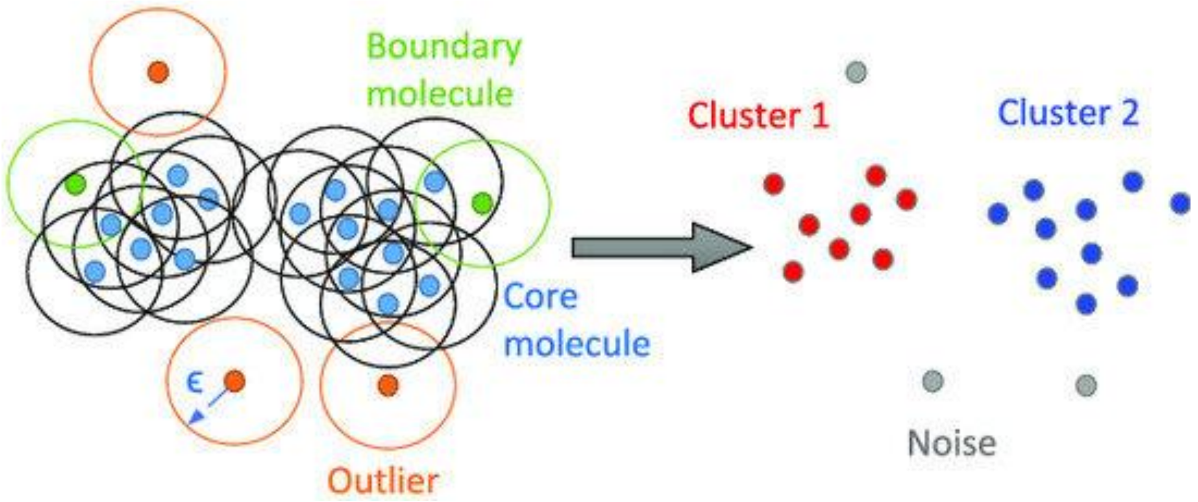
Eps = 10, $MinPts$ = 4

DBSCAN

- Algorithm – form clusters using core points, and assign border points to one of its neighboring clusters
  1) label all points as core, border, or noise points
  2) put an edge between all core points within a distance *Eps* of each other
  3) make each group of connected core points into a separate cluster
  4) assign each border point to one of the clusters of its associated core points
  5) noise points become outliers



Source: https://www.researchgate.net/figure/Illustration-of-DBSCAN-clustering-algorithm-Source-42_fig2_351565115

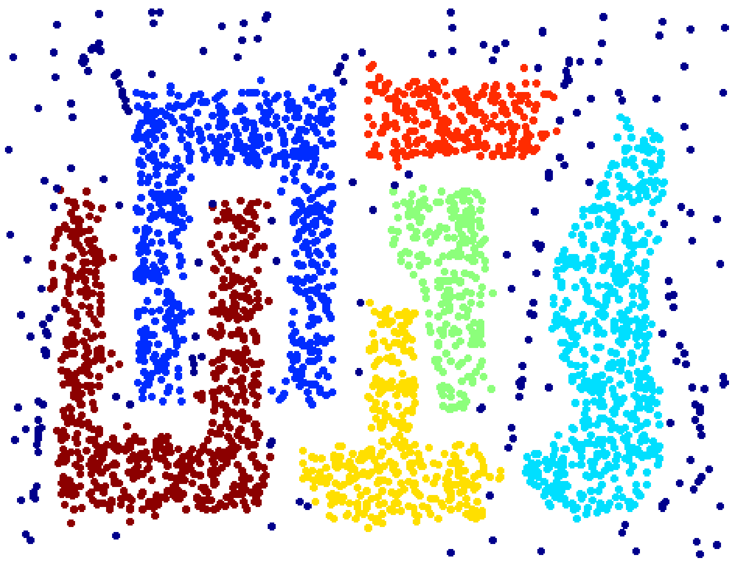


Source: https://www.researchgate.net/figure/Illustration-de-la-methode-de-clustering-DBSCAN_fig44_361040063

DBSCAN

- Strengths:
  - Can handle clusters of different shapes and sizes
  - Resistant to noise

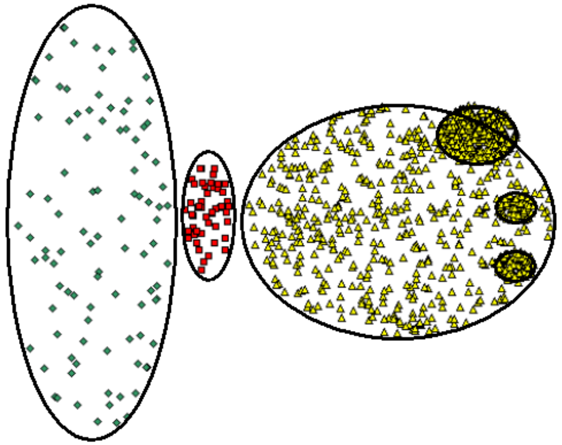
Original points



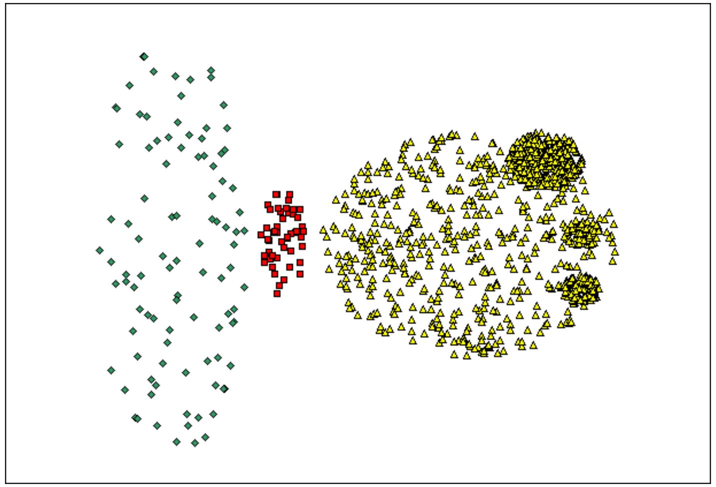
Clusters (dark blue points indicate noise)

DBSCAN

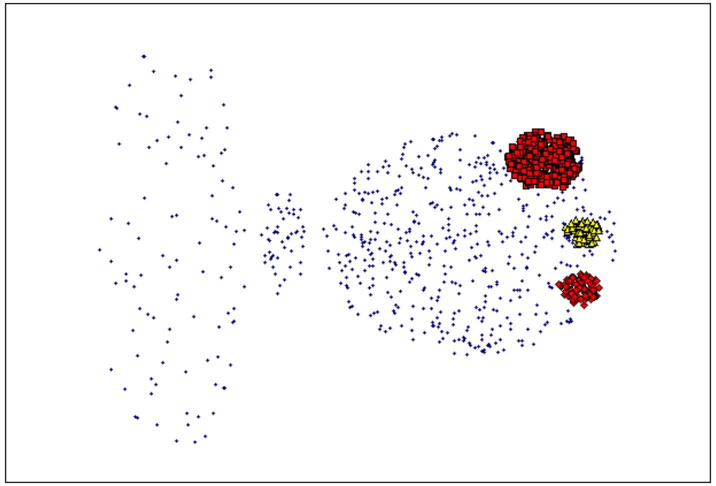
- Does not work well for
  - Varying densities
  - High-dimensional data



Original points



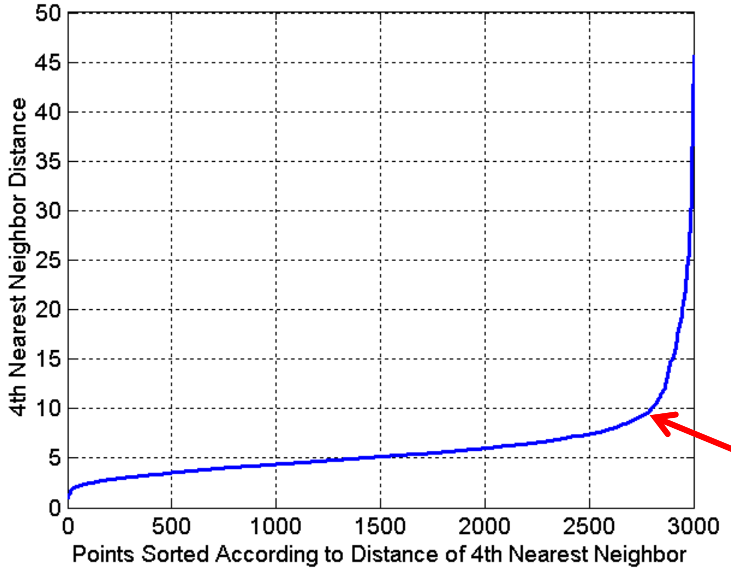
MinPts=4, *Eps*=9.92



MinPts=4, *Eps*=9.75

DBSCAN

- Determining *MinPts* and *Eps*:
  - *MinPts* – depends on data complexity (e.g., dimensionality, noise, dataset size)
    - Usually set between 4–20
  - *Eps* – depends on data density
    - Points in a cluster should have their *k*-th nearest neighbor at close distance (*k* = *MinPts*)
    - Noise points should have the *k*-th nearest neighbor at farther distance
    - So, plot sorted distances of every point to its *k*-th nearest neighbor



Suitable *Eps* (~9.0) at this point

DBSCAN versus k-means

- k-means has a prototype-based notion of a cluster; DBSCAN uses a density-based notion
- k-means can find clusters that are not well-separated; DBSCAN will merge clusters that touch
- DBSCAN handles clusters of different shapes and sizes; k-means prefers globular clusters
- DBSCAN can handle noise and outliers; k-means performs poorly in the presence of outliers
- k-means can only be applied to data for which a centroid is meaningful; DBSCAN requires a meaningful definition of density
- DBSCAN makes no distribution assumptions; k-means is really assuming spherical Gaussian distributions

DBSCAN versus k-means

- DBSCAN works poorly on high-dimensional data; k-means works well for some types of high-dimensional data
- Because of random initialization, the clusters found by k-means can vary from one run to another; DBSCAN always produces the same clusters
- DBSCAN automatically determines the number of clusters; k-means does not
- k-means has only one parameter; DBSCAN has two parameters

Graph-based clustering

- Graph-based clustering requires to construct a proximity graph
  - Each datapoint is a node
  - Each edge between nodes has a weight which is the proximity between points
    - Weight between points p and q based on the inverse of the distance (i.e., $1/dist(p,q)$)

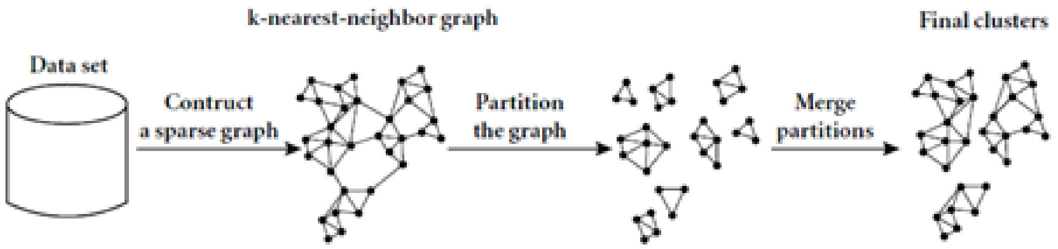
- Graph-based clustering:
  - Advantages (compared to DBSCAN) – can handle varying density of clusters and are less sensitive to parameter settings
  - Disadvantages – limited scalability – graph construction can be very expensive
    - "Sparsification" can drastically improve the scalability by reducing the number of edges in a graph while preserving its essential structure

Chameleon algorithm

- Sparsification of the proximity graph
  - A *k*-nearest neighbors (*k*-NN) graph:
    - Capturing the relationship between a point and its *k*-nearest neighbors
    - Each datapoint is a node
    - Each node is connected to its *k* most similar neighbors forming *k* edges per node
  - A symmetric *k*-NN graph:
    - There is an edge between two nodes if they are among each other's *k*-nearest neighbors
    - Substantially reduces the number of edges compared to a *k*-NN graph → preferred variant
  - *k*-NN graphs reduce the number of edges from $O(n^2)$ to $O(n)$

  - Advantages
    - Drastically reduces computational cost (99% of entries in the proximity matrix can be elimin.)
    - Preserves cluster structure by maintaining strong intra-cluster connectivity while breaking the connections to less similar points
      - This reduces the impact of noise and outliers and sharpens the distinction between clusters

Chameleon algorithm



- Preprocessing step:
  - Construct a (symmetric) *k*-NN graph
    - To capture the relationship between a point and its *k*-nearest neighbors (i.e., compute and sparsify the proximity matrix)
- Phase 1:
  - Partition the sparse *k*-NN graph into small sub-clusters of well-connected vertices (using some multilevel graph partitioning algorithm)
    - Each such sub-cluster should contain mostly points from one “true” cluster, i.e., be a sub-cluster of a “real” cluster
- Phase 2:
  - Use hierarchical agglomerative clustering to dynamically merge sub-clusters
    - Combine sub-clusters if they maintain strong connectivity and similar densities
    - Quantified by two properties: Relative Interconnectivity (RI) and Relative Closeness (RC)
    - Select the pair of clusters C_i and C_j that maximizes $RI(C_i, C_j) \cdot RC(C_i, C_j)^{\alpha}$, where α is a user-defined parameter balancing the importance between RI and RC

Chameleon – merging properties

- Relative Interconnectivity – two clusters are combined if the points in the resulting cluster are almost as strongly connected as points in each of the original clusters
  - Quantified by absolute interconnectivity of two clusters normalized by the internal connectivity of the clusters:

$$RI = \frac{EC(C_i, C_j)}{\frac{1}{2}(EC(C_i) + EC(C_j))}$$

    - $EC(C_i, C_j)$ – sum of edge weights (of *k*-NN graph) that interconnect clusters C_i and C_j
    - $EC(C_i)$ – minimum sum of the cut edges if we bisect cluster C_i (i.e., when the graph is divided into two roughly equal parts)

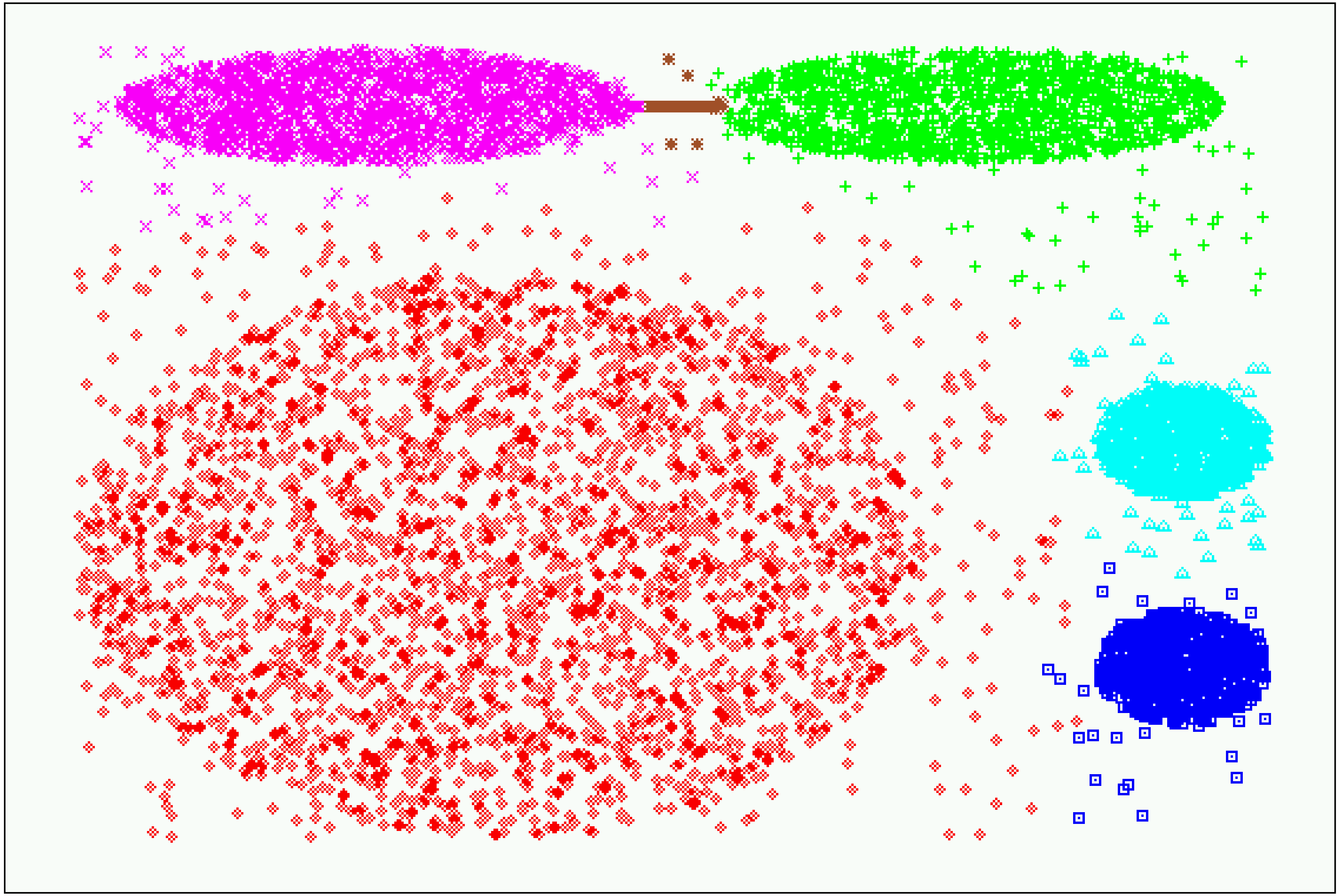
Chameleon – merging properties

- Relative Closeness – two clusters are combined only if the points in the resulting cluster are almost as close to each other as in each of the original clusters
  - Quantified by absolute closeness of two clusters normalized by the internal closeness of the clusters:

$$RC = \frac{\bar{S}_{EC}\,(C_i, C_j)}{\frac{|C_i|}{|C_i| + |C_j|}\bar{S}_{EC}(C_i) + \frac{|C_j|}{|C_i| + |C_j|}\bar{S}_{EC}(C_j))}$$

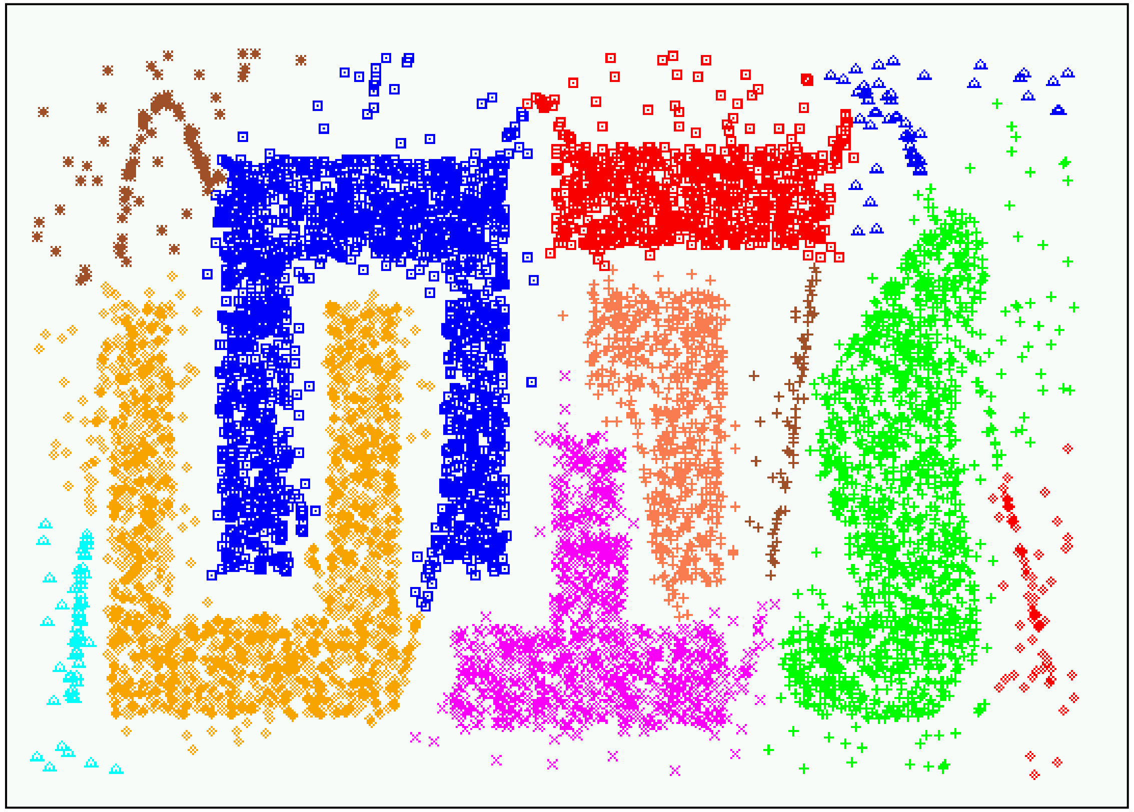
    - $\bar{S}_{EC}(C_i, C_j)$ – average weight of the edges (of *k*-NN graph) that connect clusters C_i and C_j
    - $\bar{S}_{EC}(C_i)$ – average weight of the edges if we bisect cluster C_i
    - $|C_i|$ – size of cluster C_i

Experimental comparison

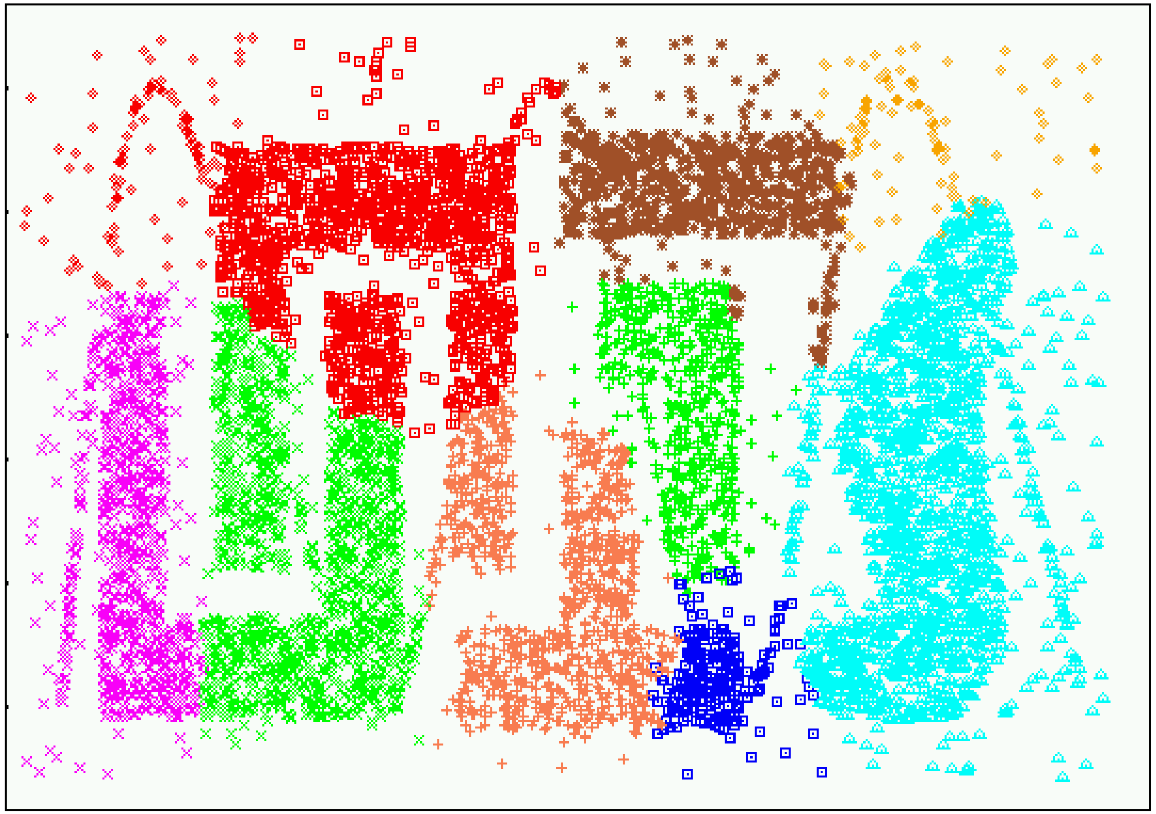


Experimental comparison

- Comparison to CURE (Clustering Using REpresentatives) – PA212
  - Compared to k-means, CURE is more robust to outliers and able to identify clusters having non-spherical shapes and size variances



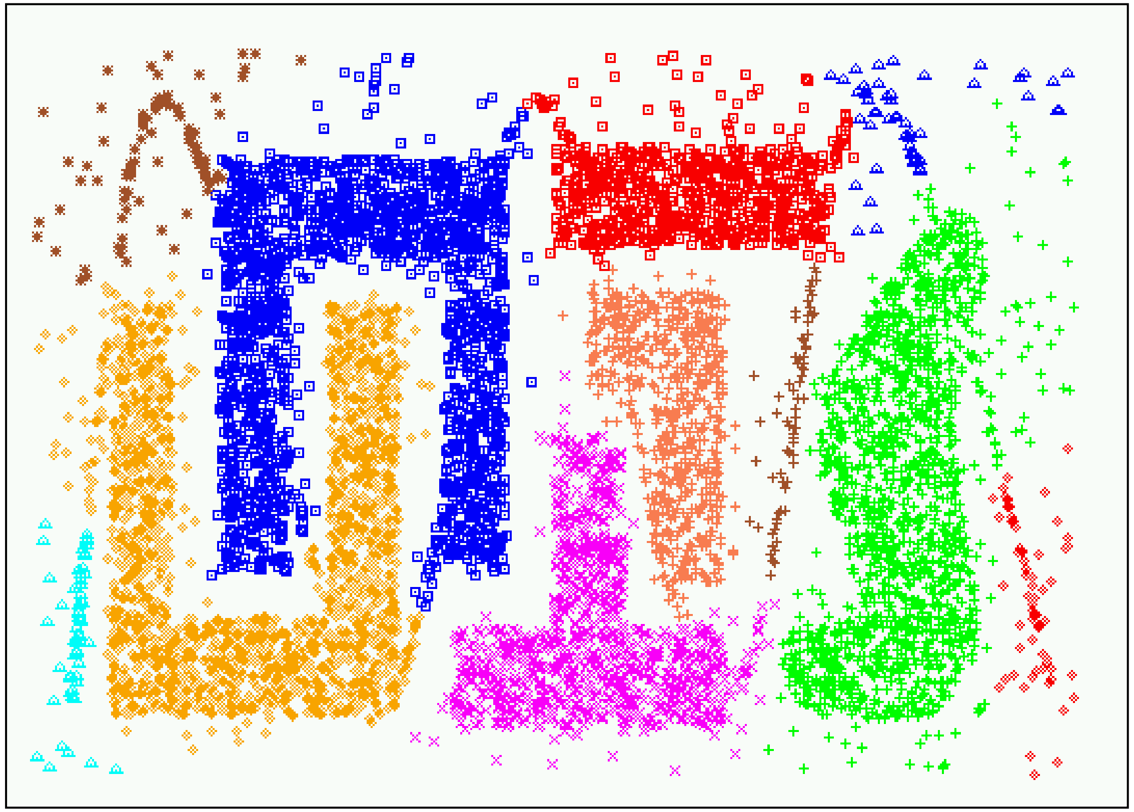
Chameleon



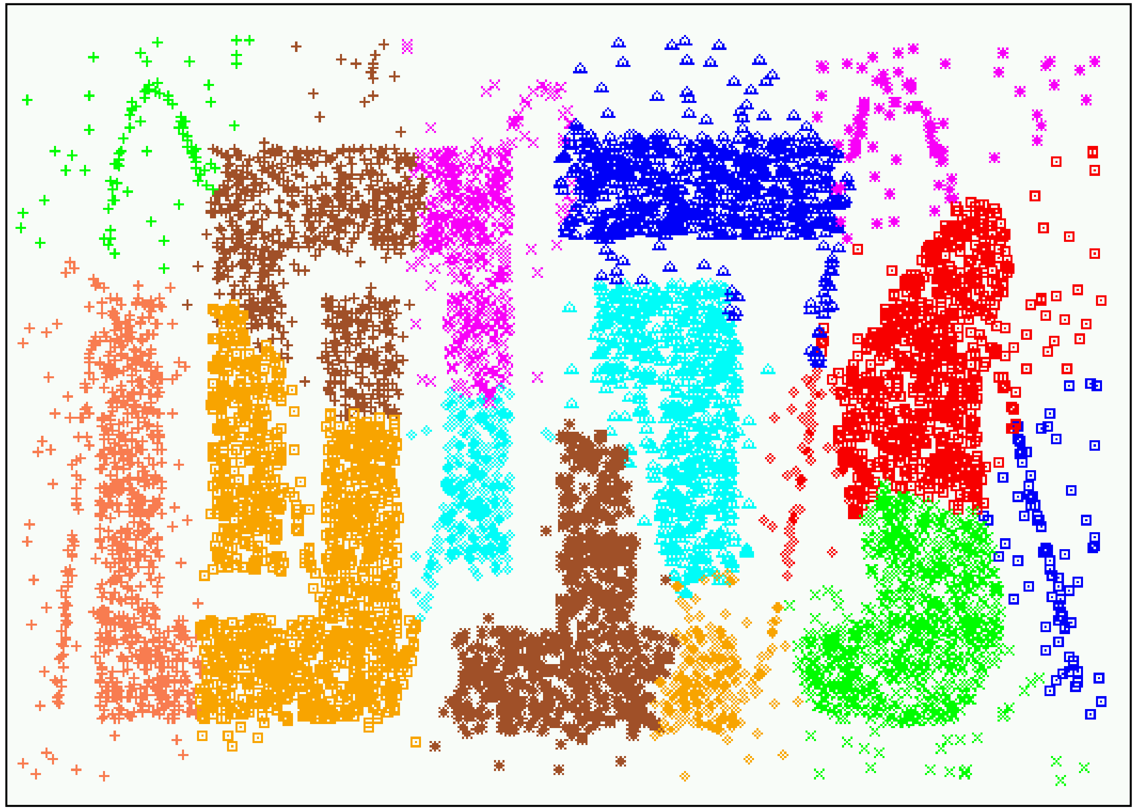
CURE (10 clusters)

Experimental comparison

- Comparison to CURE



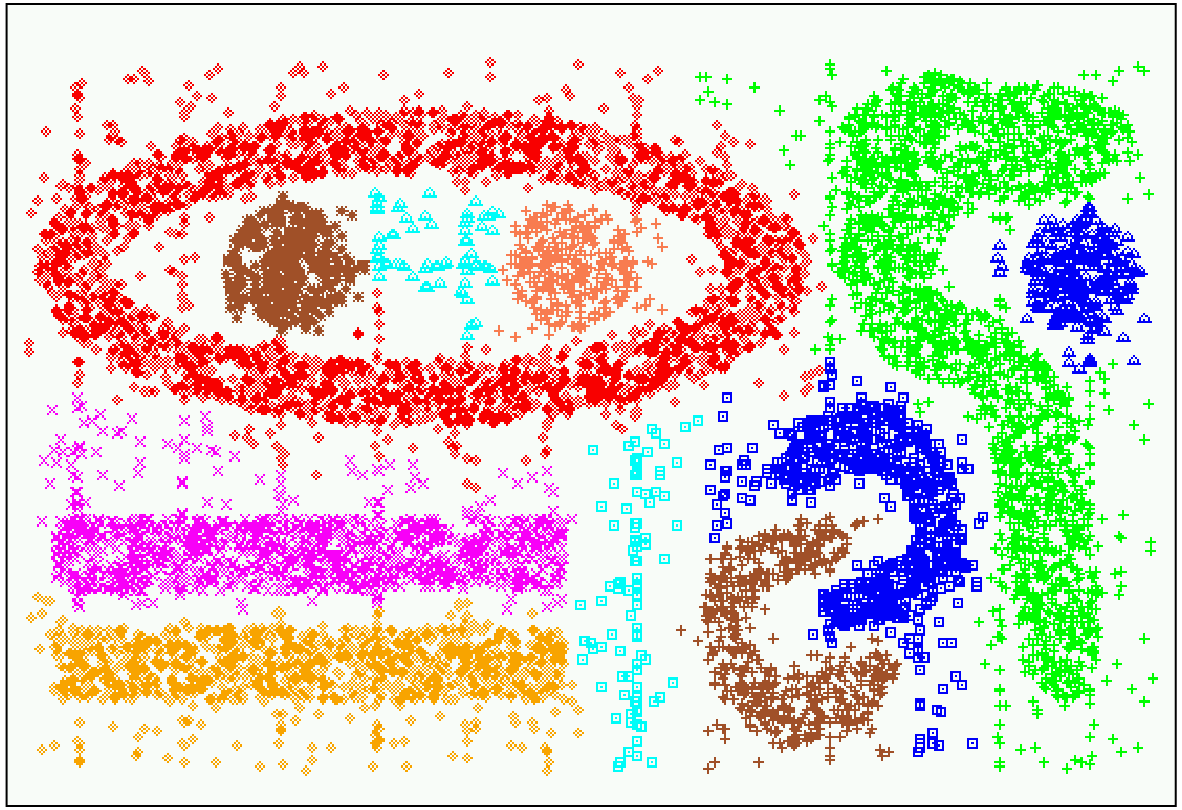
Chameleon



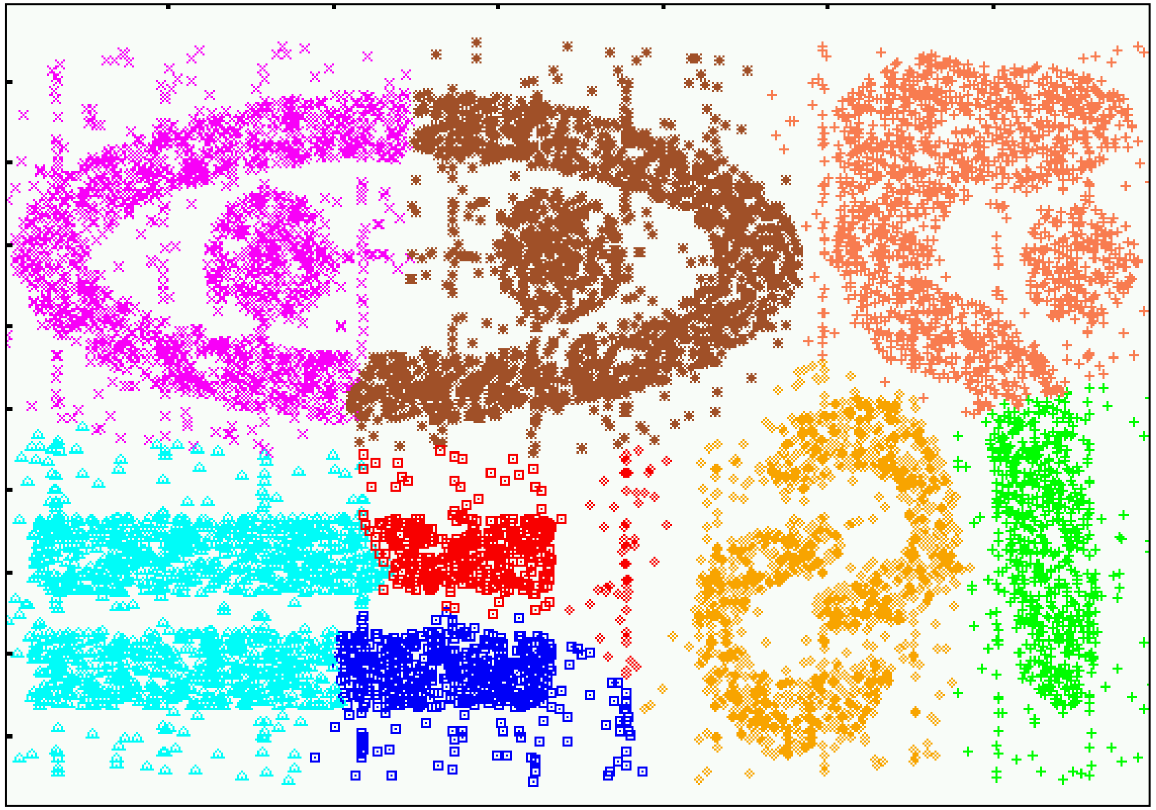
CURE (15 clusters)

Experimental comparison

- Comparison to CURE



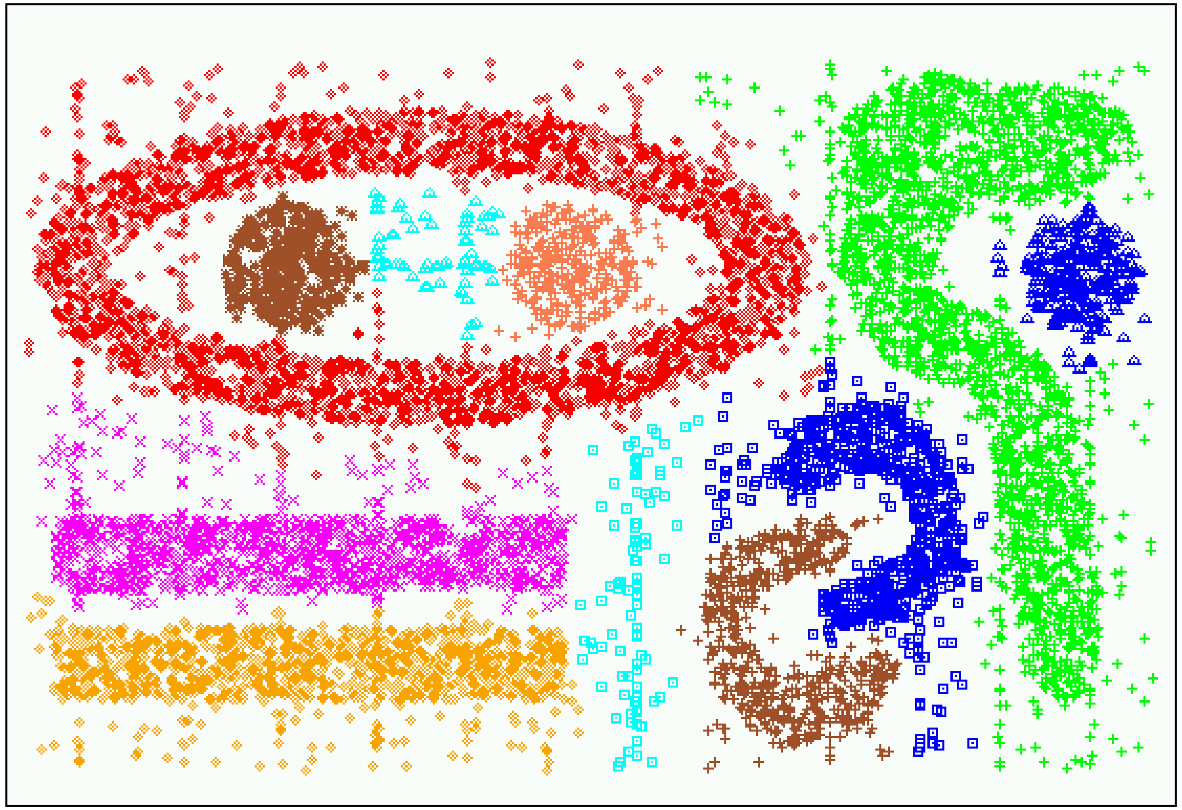
Chameleon



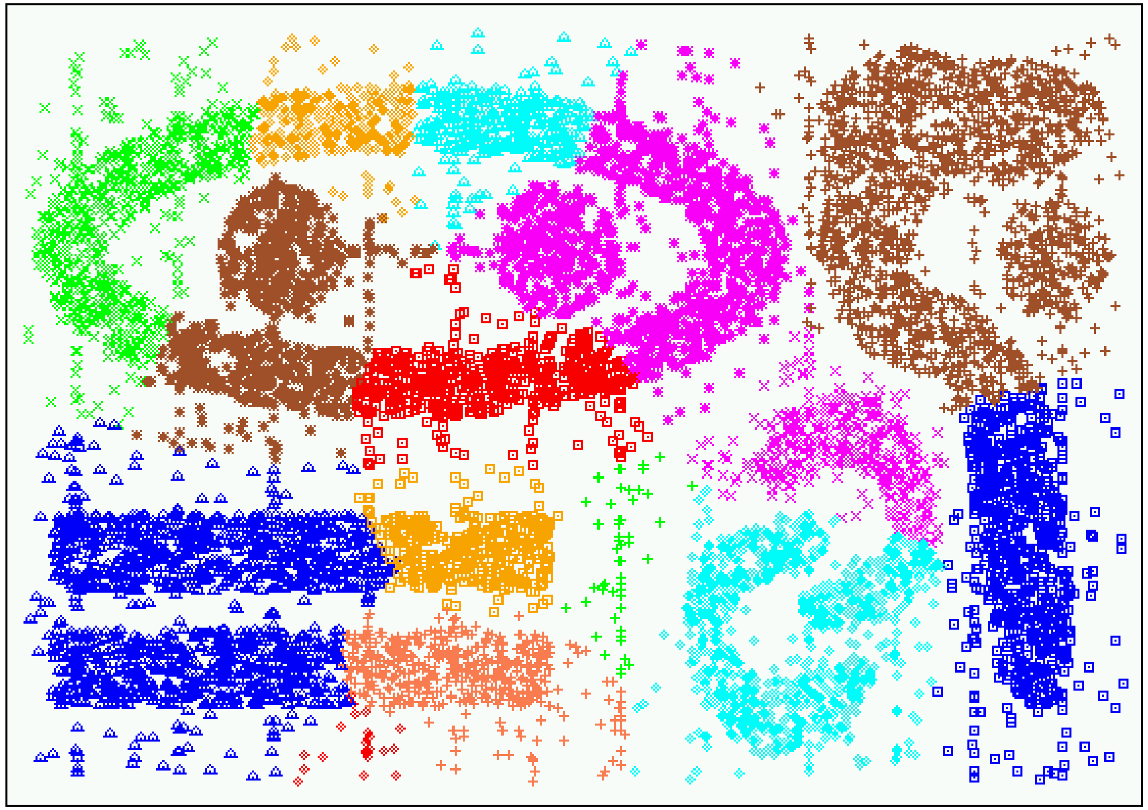
CURE (9 clusters)

Experimental comparison

- Comparison to CURE



Chameleon



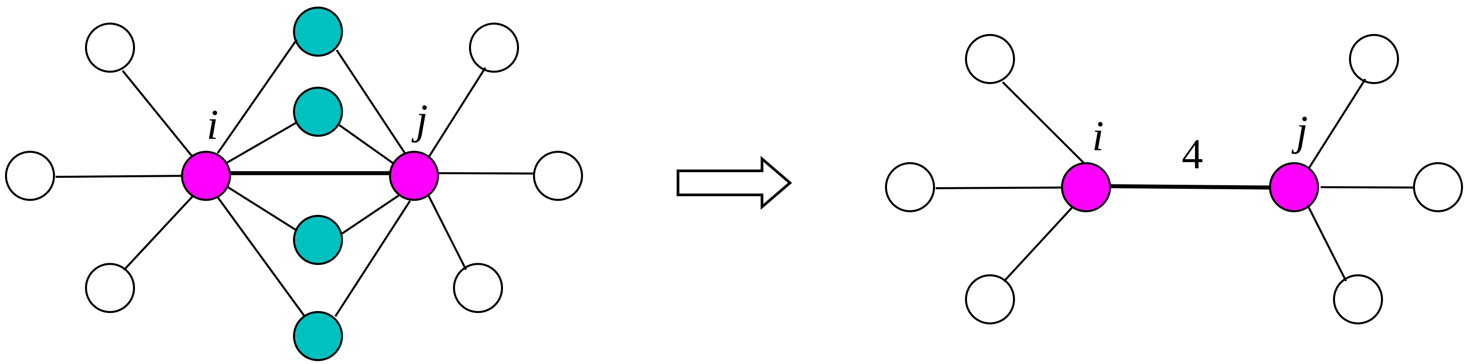
CURE (15 clusters)

Chameleon – properties

- Existing merging schemes in hierarchical clustering algorithms are static in nature, e.g.:
  - MIN – merges two clusters based on their closeness
  - Group average – merges two clusters based on their average connectivity
- Chameleon uses a dynamic model that adapts to the characteristics of the data by finding natural clusters (based on RI and RC properties)

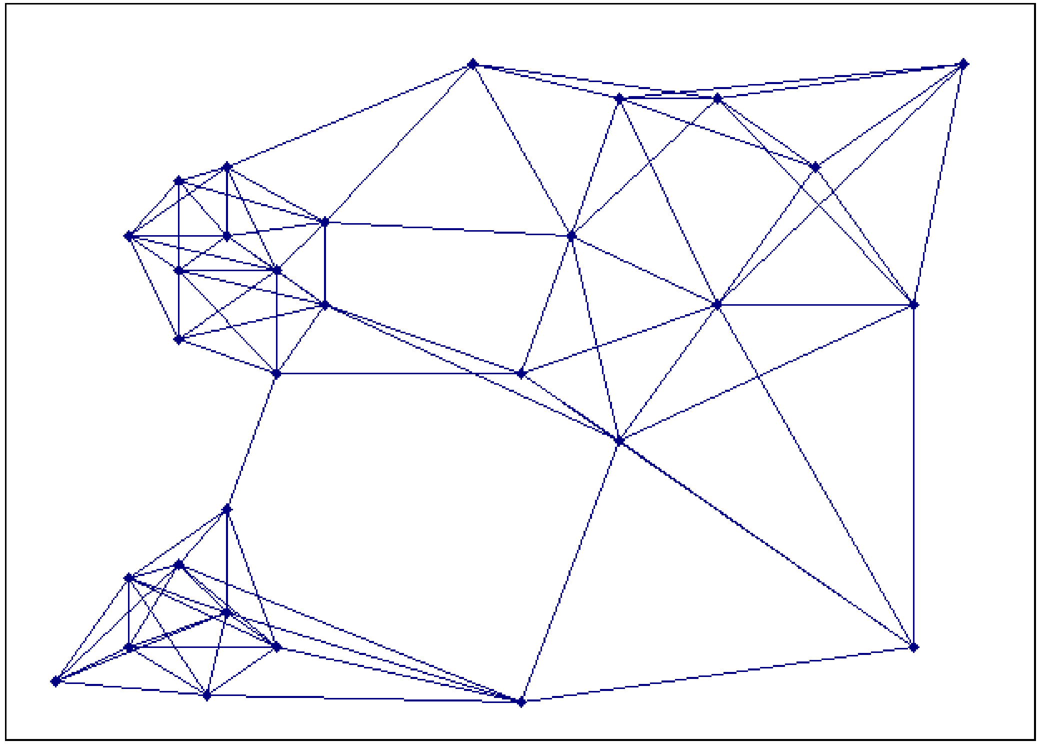
Shared nearest neighbor graph

- Shared Nearest Neighbor (SNN) graph
  - Extension of the *k*-NN graph
  - Given that the vertices are connected, the weight of an edge is the number of shared nearest neighbors between vertices
    - Idea – if two points are similar to many of the same points, then they are likely similar to one another, even if a direct measurement of similarity does not indicate this

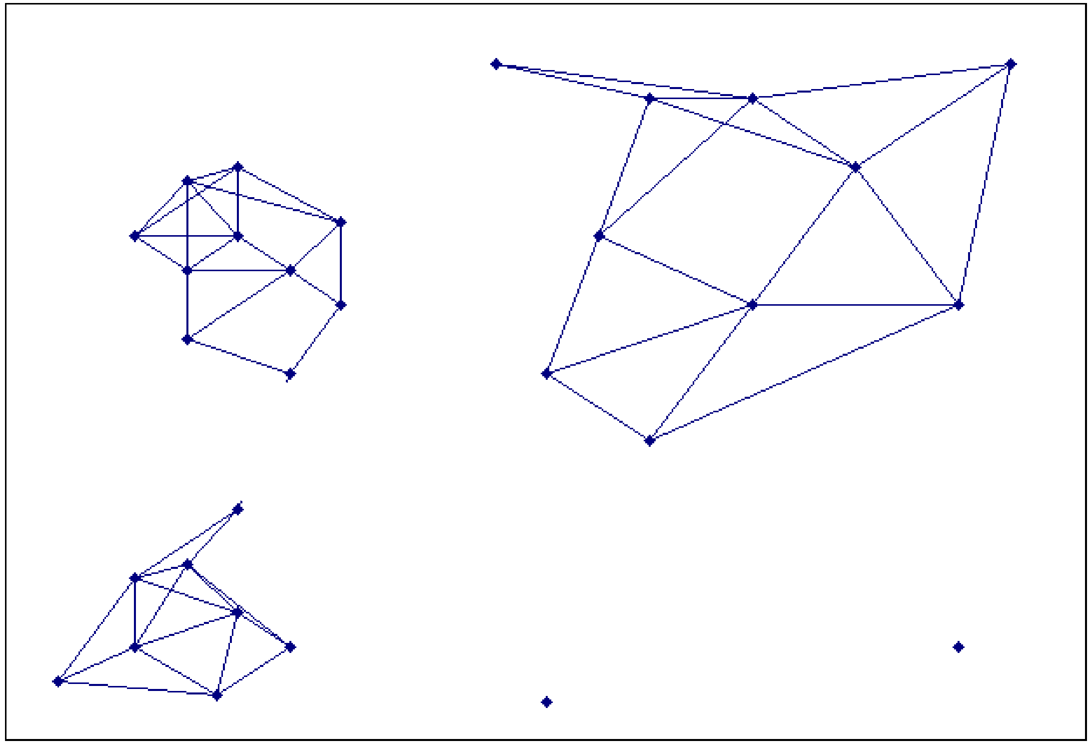


SNN graph illustration

- Sparse graph – link weights are similarities between neighboring points
- SNN graph – link weights are numbers of shared nearest neighbors



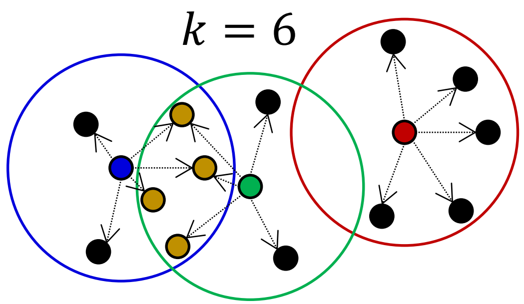
Sparse graph



SNN graph

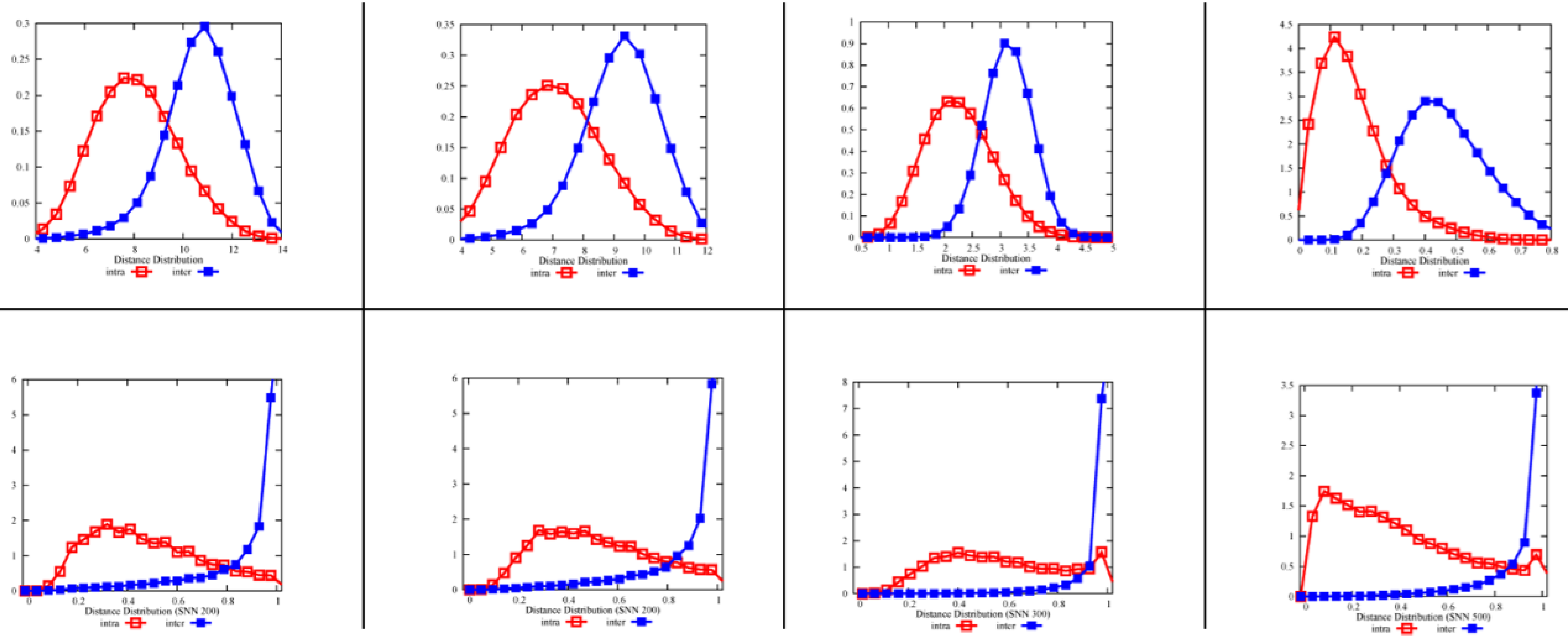
SNN graph illustration

- Intra-cluster distances
- Inter-cluster distances



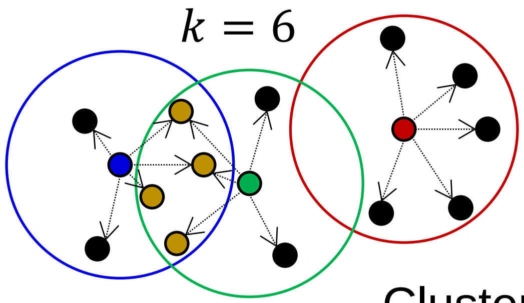
$$SNNsim_k(p,q) = |NN(p,k) \cap NN(q,k)|$$
$$SNNsim_6(\bullet,\bullet) = 4$$
$$SNNsim_6(\bullet,\bullet) = 0$$

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Euclidean distance				
SNN distance				



Jarvis-Patrick algorithm

- Steps:
  1) Construct the SNN graph
  2) SNN-based clustering
     - Initially, each datapoint is its own cluster
     - A pair of points p and q is put in the same cluster if
       - p and q share at least *T* neighbors (user-defined threshold *T*), i.e., $SNNsim_k(p, q) \geq T$, and
       - p and q are in each others *k*-nearest neighbor list
         - E.g., we might choose a nearest neighbor list of size $k = 20$ and put points in the same cluster if they share more than $T = 10$ near neighbors
     - If a point does not share enough neighbors with any other point, it is considered an outlier

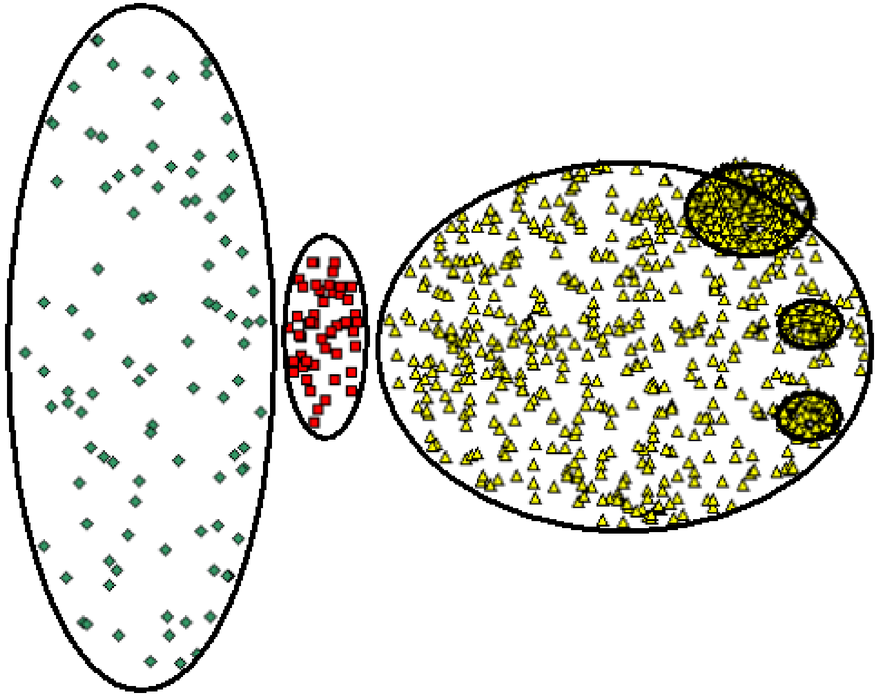


$$SNNsim_k(p, q) = |NN(p, k) \cap NN(q, k)|$$
$$SNNsim_6(\bullet, \bullet) = 4$$
$$SNNsim_6(\bullet, \bullet) = 0$$

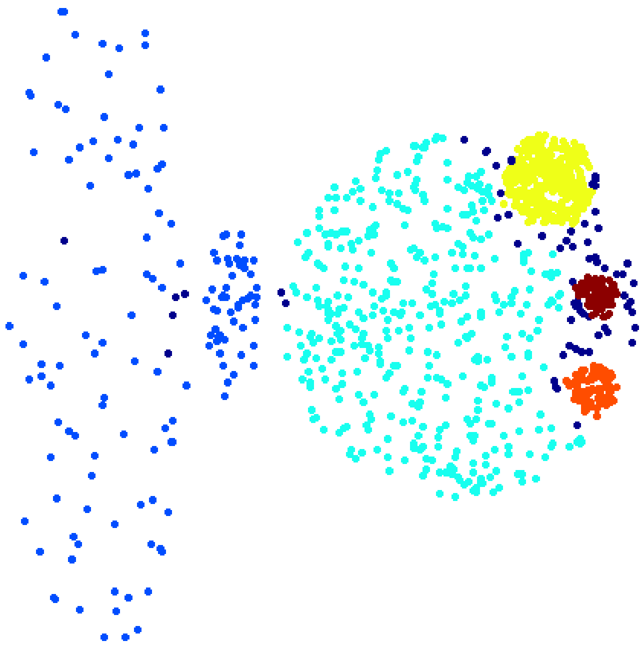
Clusters ● and ● merged for $k = 8$ but not for $k = 6$

Experimental evaluation

- When Jarvis-Patrick works reasonably well



Original points

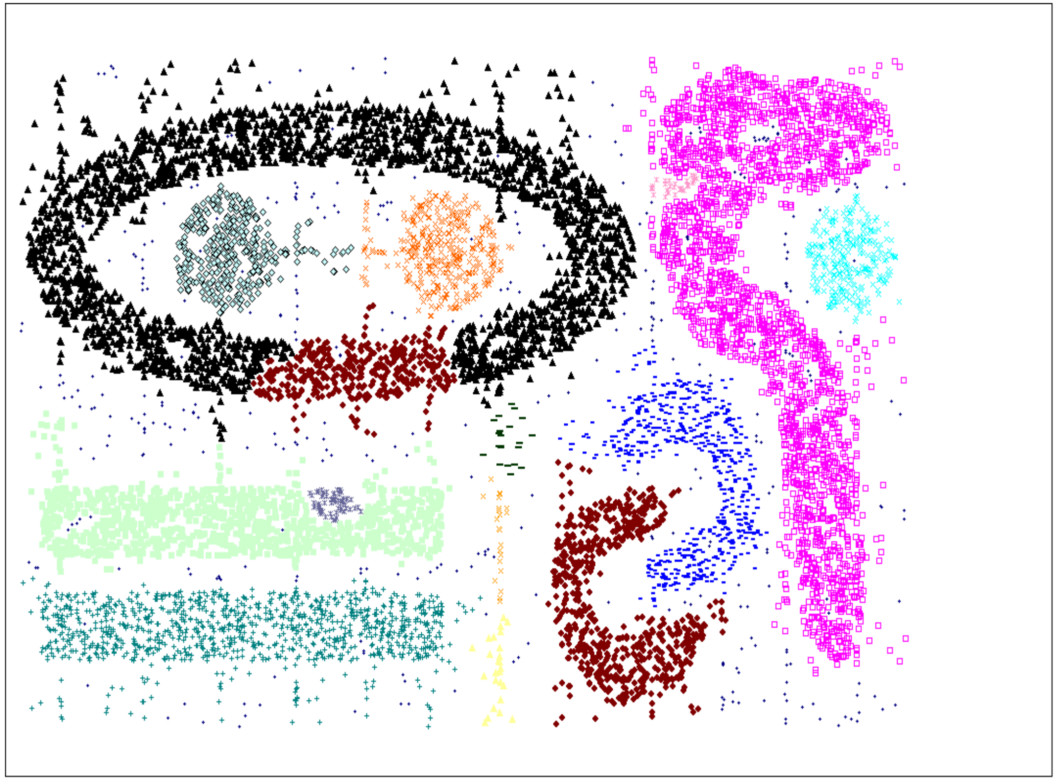


Jarvis-Patrick clustering

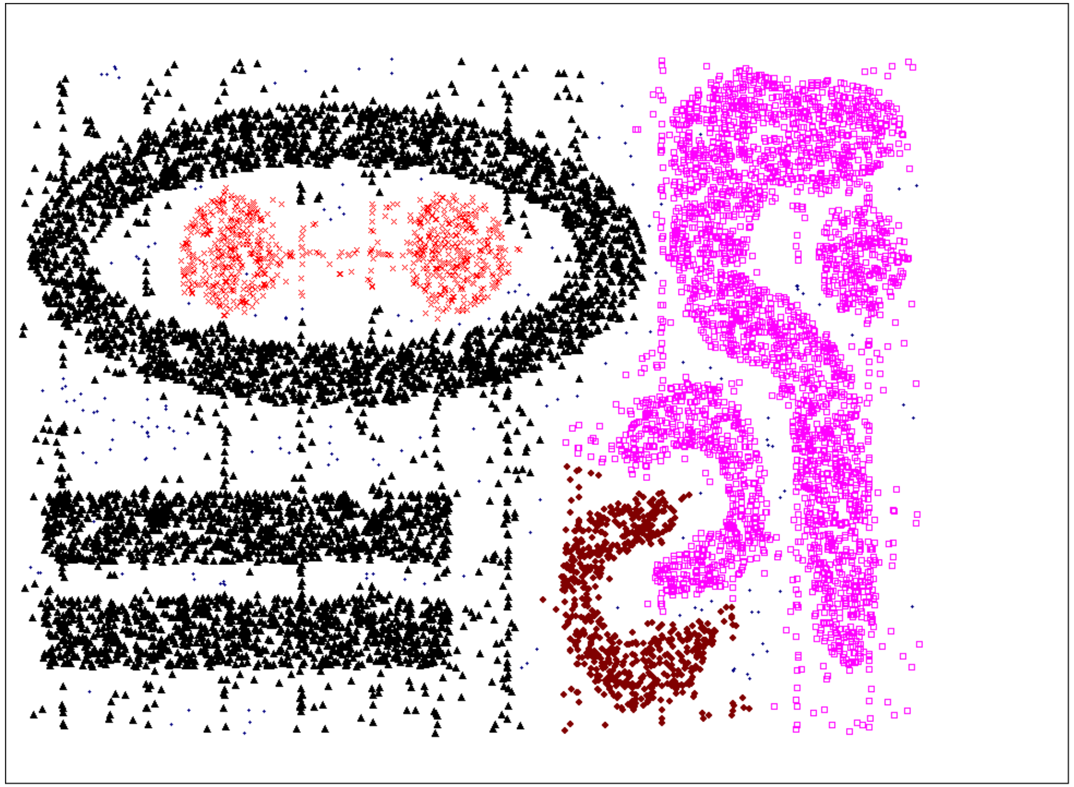
6 shared neighbors out of 20

Experimental evaluation

- When Jarvis-Patrick does not work well
  - Jarvis-Patrick clustering is too brittle



Smallest threshold (T) that does not merge clusters



Threshold $T - 1$

SNN Density-based Clustering

- SNN Density-based Clustering algorithm (SNN-DBSCAN)
  - Combines:
    - SNN graph (similarity definition based on the number of shared nearest neighbors)
    - Density based clustering (DBSCAN-like approach)
  - Advantages:
    - Improve clustering quality of DBSCAN, especially for arbitrarily shaped clusters and varying densities

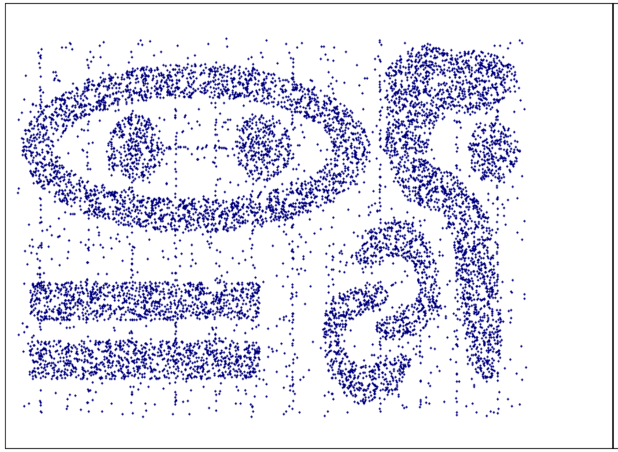
SNN Density-based Clustering

DBSCAN parameters: $MinPts$ and Eps

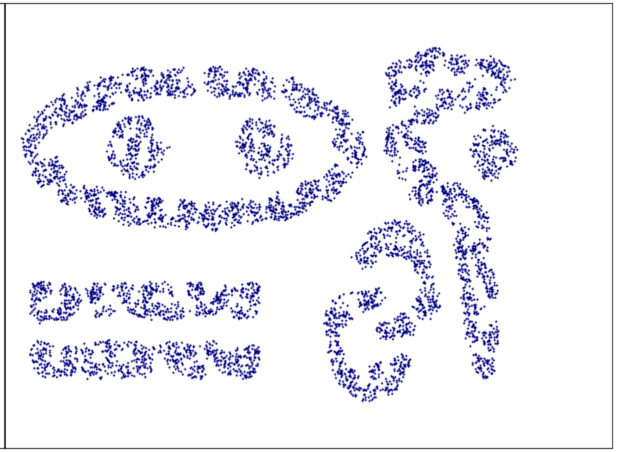
- Steps:
  1) Construct the SNN graph
  2) Compute the SNN density $SNNdens_k(p, T)$ of each point p
     - $SNNdens_k(p, T) = |\{q \mid SNNsim_k(p, q) \geq T\}|$ ~ # of neighbors with $\geq T$ shared neighbors
  3) Find the core points
     - A core point is a high-density point p such that $SNNdens_k(p, T) \geq MinPts$
  4) Form clusters from the core points
     - Two core points p and q are connected if $SNNsim_k(p, q) \geq T$ ($T \sim Eps$)
  5) Connect border points to the clusters
     - Non-core point p is connected to the cluster with core point q if $SNNsim_k(p, q) \geq T$ ($T \sim Eps$)
  6) The rest of points (noise points) remain outliers

- Points 2–6 correspond to DBSCAN

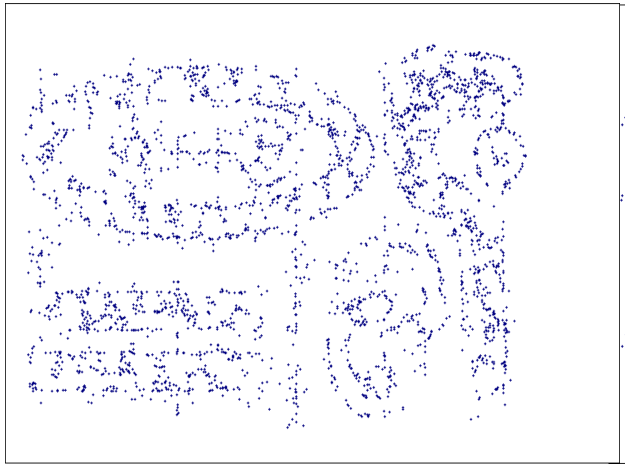
Experimental evaluation



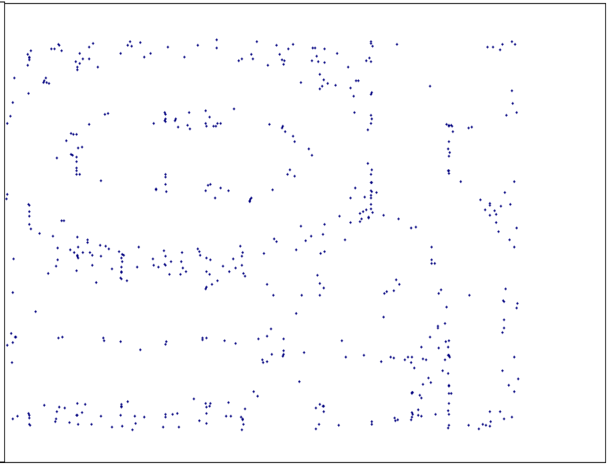
(a) All points



(b) Core points (~high SNN density)

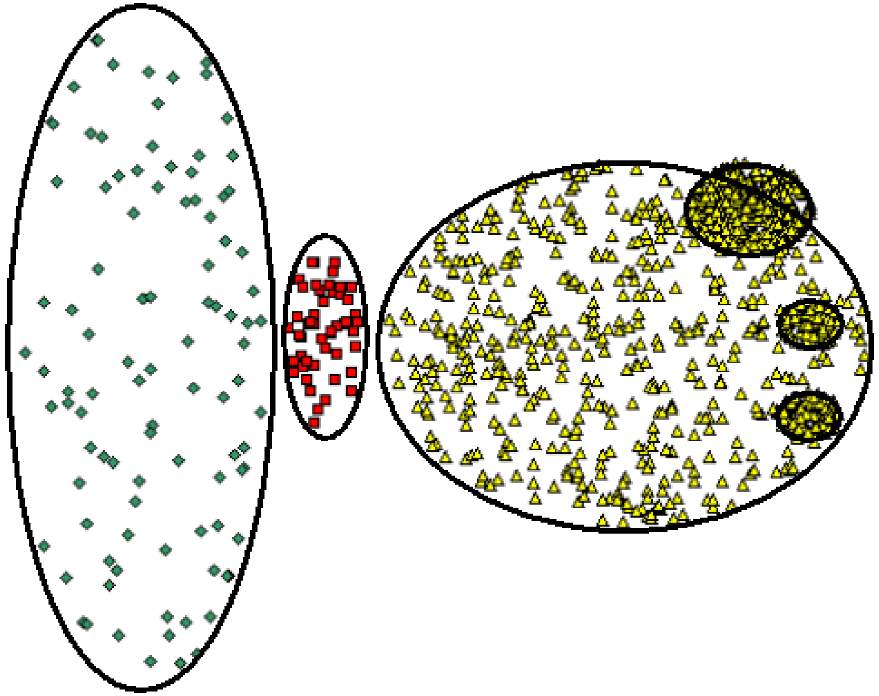


(c) Border points (~medium SNN density)

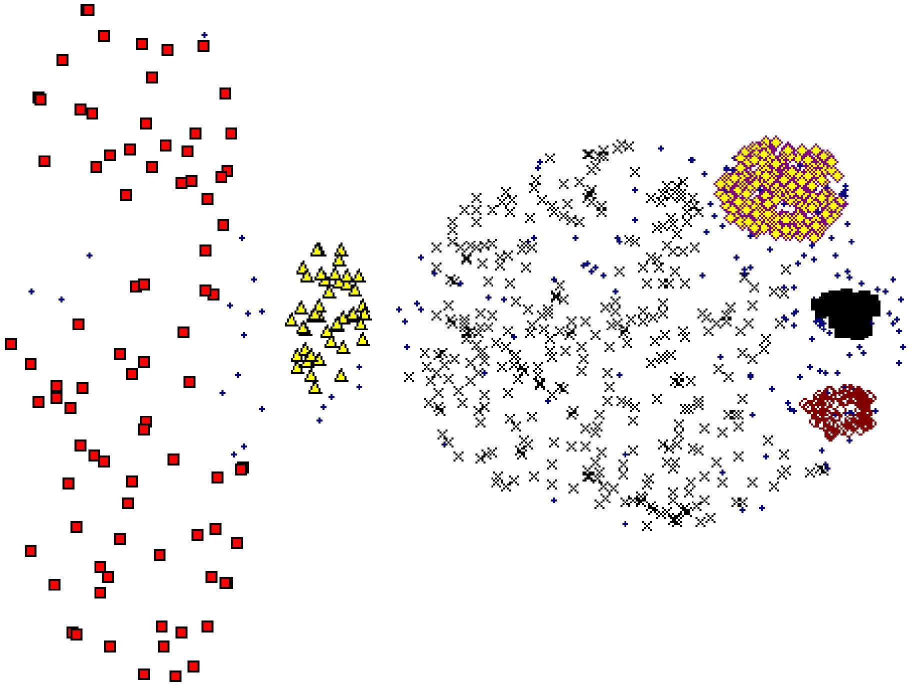


(d) Noise points (~low SNN density)

Experimental evaluation



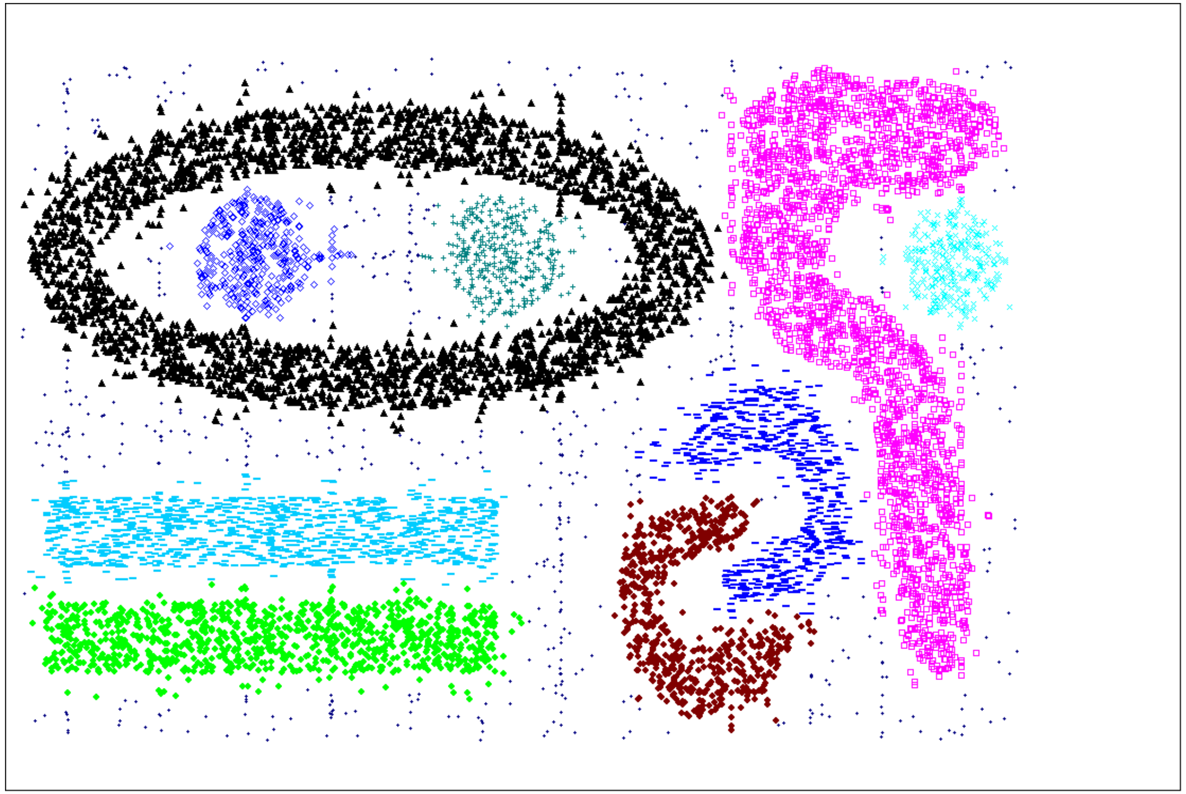
Original points

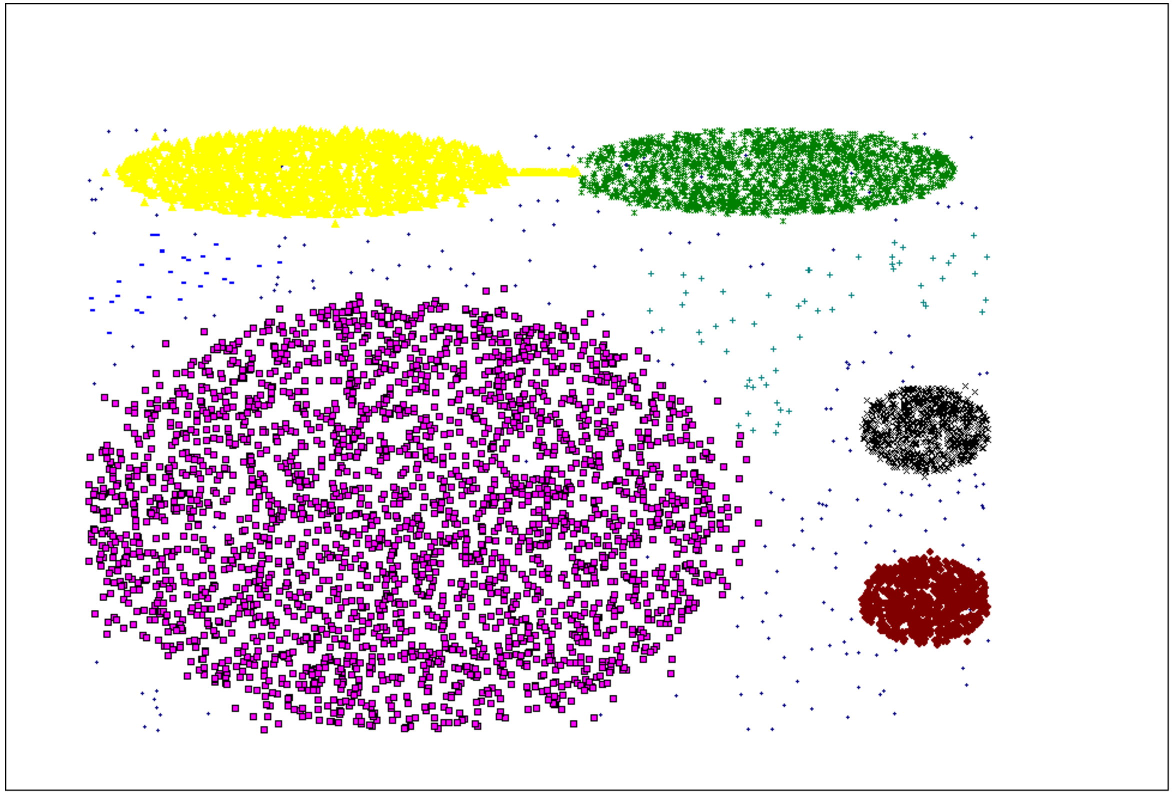


SNN Density-based Clustering

Experimental evaluation

- SNN Density-based Clustering can handle other difficult situations





SNN Density-based Clustering

- Limitations – complexity is high:
  - Time: $O(n^2)$ in the worst case, where n is # of objects
    - $O(n \cdot$ time to find numbers of neighbor within threshold $T \sim Eps)$
    - There are more efficient ways to find the nearest neighbors:
      - R* Tree or k-d Trees for lower dimensions
      - M-Tree, LMI, FAISS for high-dimensional data
- Parameterization is not easy

Sources

- Introduction to Data Mining, University of Minnesota:
  - https://www-users.cse.umn.edu/~kumar001/dmbook/firsted.php
- Machine Learning Bits (Cluster Analysis), University of Dortmund:
  - https://dm.cs.tu-dortmund.de/mlbits/cluster-intro/