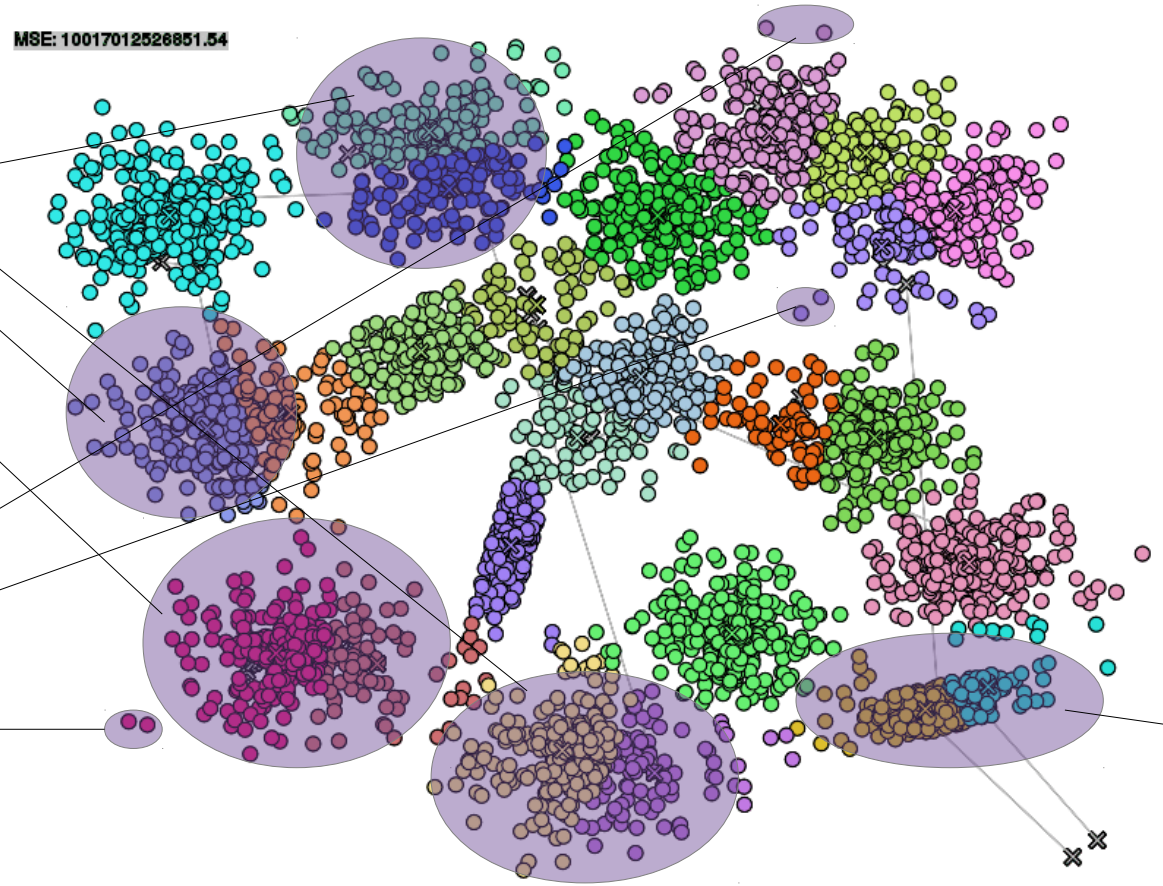
Problems of traditional methods

- Large data groups with arbitrary shape are not necessarily detected being in the same cluster.
- Choosing a proper number of clusters is usually impossible without any knowledge or efficient (possibly complicated and/or time-consuming) heuristics.
- Noise (distant isolated points) interfere with the clustering process.
- For example, consider the following illustration (S2 dataset clustered with RLS):



Clearly the decision to use 25 clusters was wrong: some visible segments are split, usually from their densest parts.

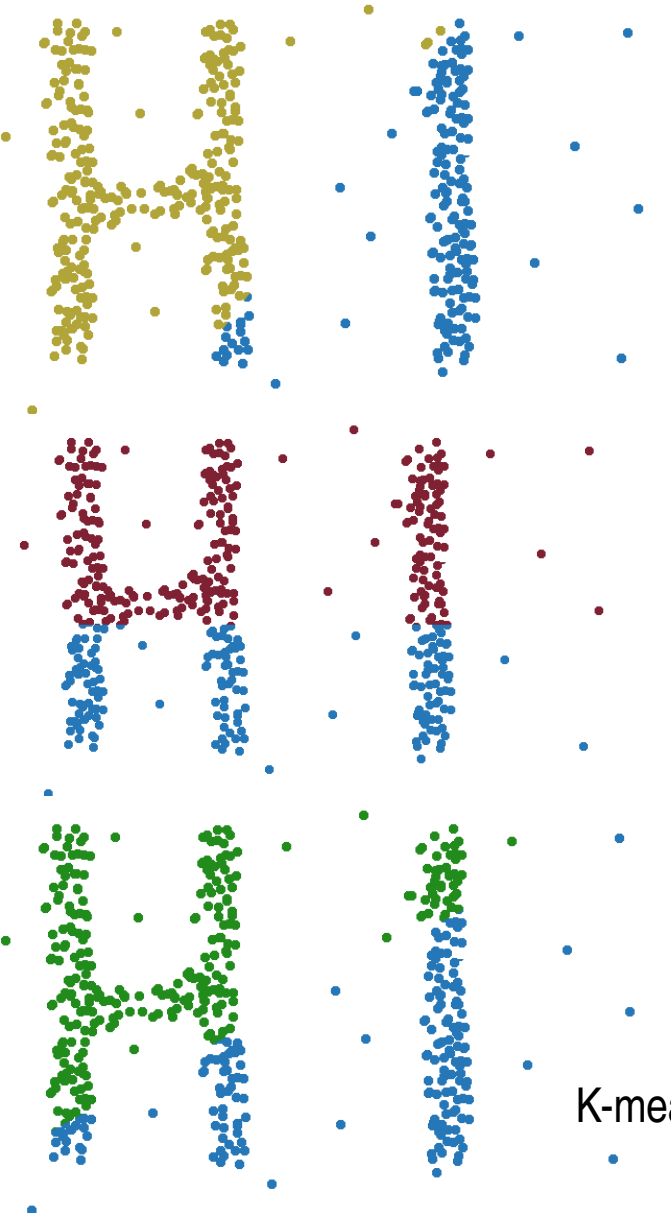
Isolated points which probably should be classified as noise rather than points belonging to any cluster.

Distance metrics in general (including Euclidean distance) are efficient in evaluating segments that span similarily in all dimensions. Segments with arbitrary shape, however, are usually detected with multiple clusters.

Pros of DBSCAN

- DBSCAN addresses some of these issues
  - Detection of noise
  - Number of clusters is automatically defined based on the density of data points
  - Detects arbitrary shapes (spherical, drawn-out, linear, elongated..)
- DBSCAN is well defined, not dependent on randomness
- According to M. Ester et al. outperforms CLARANS (which also detects some arbitrary shapes)

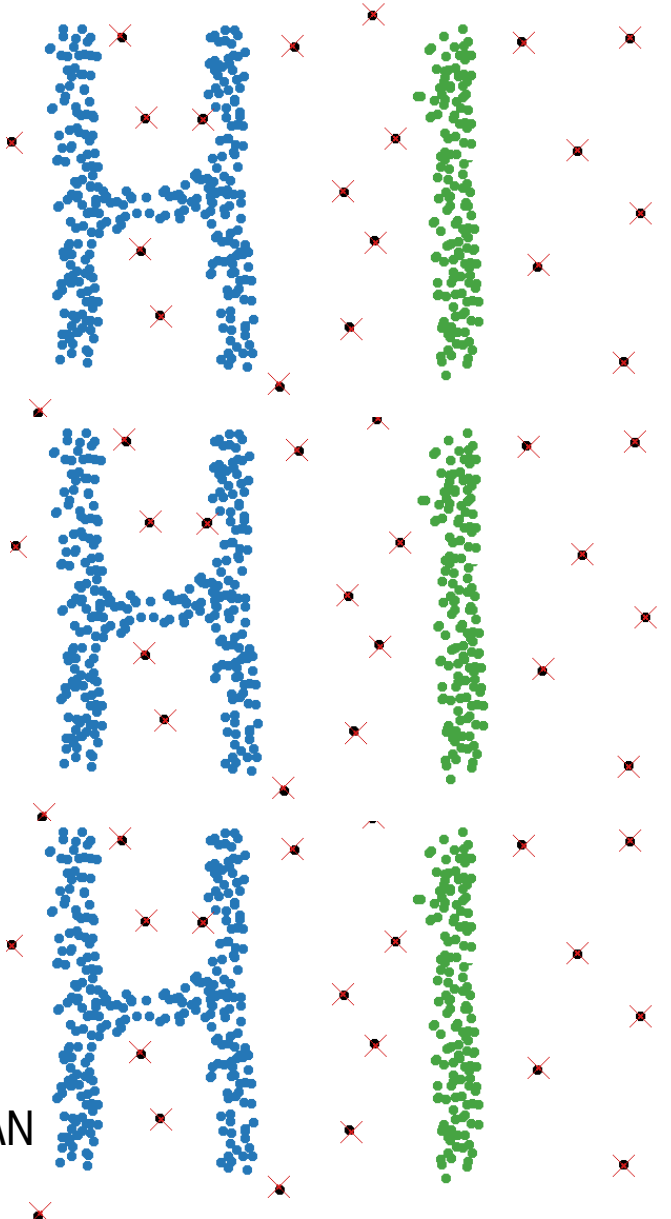
K-means vs. DBSCAN (visualized by an artificial example)



K-means

A priori knowledge has been used in order to set the number of clusters to two. Due to definitions of DBSCAN (density, density-reachability etc.) the output of DBSCAN stays the same on every run, but the output of K-means is heavily influenced by the initial clustering.

One could say that points detected by DBSCAN should be defined as a cluster of its own (resulting three clusters), but running K-means with three clusters on this example would probably introduce even worse results.



DBSCAN

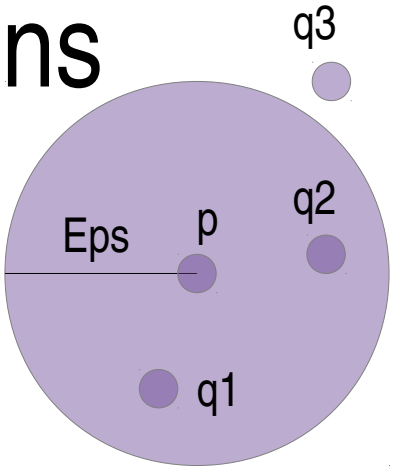
Cons of DBSCAN

- Clusters with varying inner densities can lead to problems
  - Too strict density criterion may lead to
    - Classifying points actually belonging to a cluster as noise instead
    - Splitted clusters
  - Too allowing density criterion may lead to
    - Visibly different segments classified as one cluster
    - Some noise classified as a part of a sparse cluster
- Measuring goodness of a clustering is challenging
  - Any measure emphasizing in any way minimal distances between points inside a cluster gives bad results on arbitrary shaped clusters, such as elongated clusters
  - In addition, traditional algorithms essentially attempt to minimize these intra-cluster distances, hence yielding better results for them

Definitions, definitions, definitions

- Let
  - D be group of data points
  - dist(a,b) be a distance metric between points a and b (i.e. Euclidean distance)
  - Eps and MinPts be parameters of DBSCAN
    - Eps stands for the radius used in density measures
    - MinPts stands for the minimum number of points required within the radius of Eps in order to consider a group of points “dense”

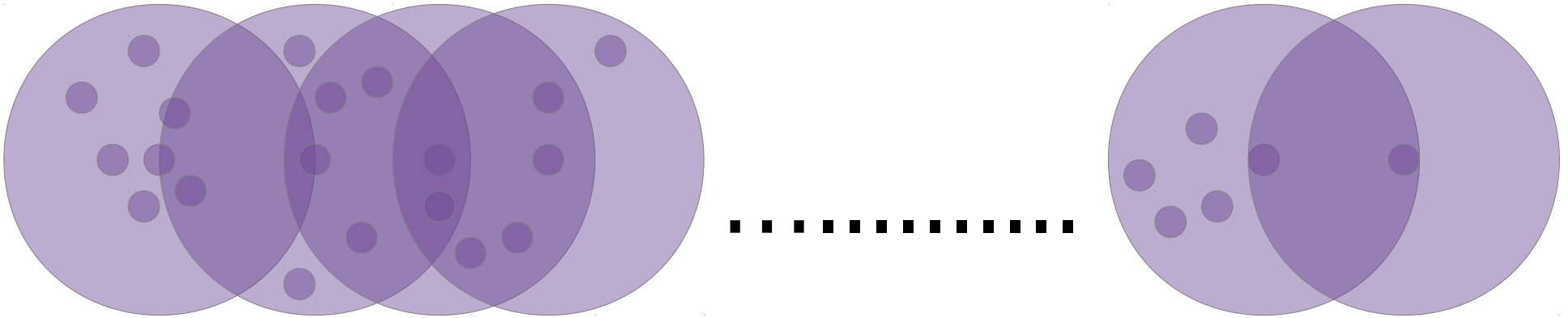
Definitions, definitions, definitions



- Eps-neighborhood of a point p
  - $N_{Eps}(p) = \{ q \in D \mid dist(p,q) \leq Eps \}$
  - In plain English: $N_{Eps}(p)$ is a group of data points having no more than distance of Eps to point p
  - Using N_{Eps} the so called core points of a cluster can be detected:
    - Each core point must have at least MinPts surrounding data points within the reach of Eps
      - Formally: $| N_{Eps}(p) | \geq MinPts$
    - For example: MinPts = 4 => a core point must, by definition, have at least four data points nearby (within the radius Eps)

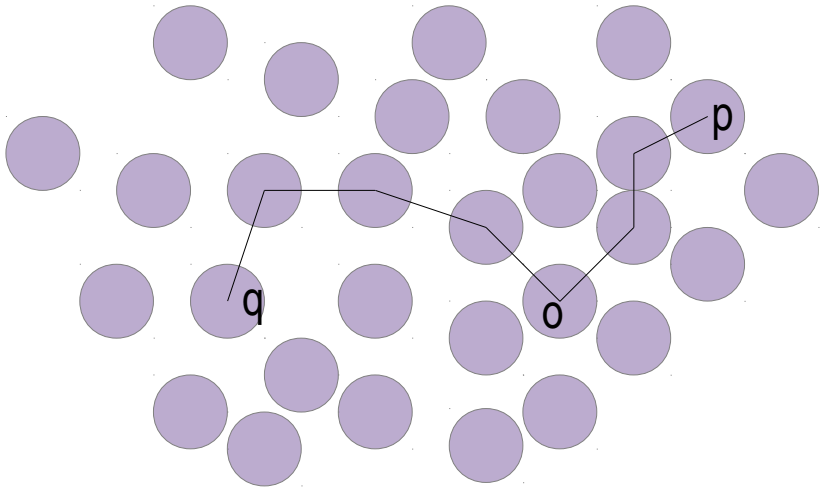
Definitions, definitions, definitions

- Point p is directly density-reachable from a point q if
  - $p \in N_{Eps}(q)$ (p is "near enough" from q: dist(p,q) ≤ Eps)
  - $| N_{Eps}(q) | \geq MinPts$ (q is a core point; it has "enough" surrounding points)
  - Note that p can also be a border point.
- Point p_n is density-reachable from a point p_1 if there is a series $p_1,...,p_n$ of points so that every point p_k is directly density-reachable from p_{k-1}
  - The general idea in plain English: a chain of directly density-reachable points means that the first and last point of the chain are connected in terms of density-reachability.
  - By definition, points $p_1,...,p_{n-1}$ are core points, but p_n can be a border point.



Definitions, definitions, definitions

- A point p is density-connected to a point q if there is a point o such that both p and q are density-reachable from o.
  - In other words, there are two “chains” (in the sense of density-reachability), one leading from o to p and another from o to q.



Definitions, definitions, definitions

- A cluster is a non-empty subset of D satisfying the following conditions:
  - $\forall$ p, q: if p $\in$ C and q is density-reachable from p then q $\in$ C (Maximality)
  - $\forall$ p, q $\in$ C: p is density-connected to q (Connectivity)
- The above conditions guarantee the uniqueness of clustering. A cluster can be found by using density-connectivity of any core point inside the cluster.
- Noise is set of points not belonging to any cluster C_i
  - Formally: noise = { p $\in$ D | $\forall i$: p $\notin C_i$ }

Finding convenient parameters

- It has been observed that MinPts = 4 is a good tradeoff between accuracy and time (M. Ester et al.).
- For Eps, this procedure is proposed by M. Ester et al.:
  - For all points p
    - q ← 4th closest neighbor of p
    - Array.add(dist(p,q))
  - Array.sort_descending()
  - Find a point having relatively “dramatic skew” in the graph drawn from Array and select that value as Eps.

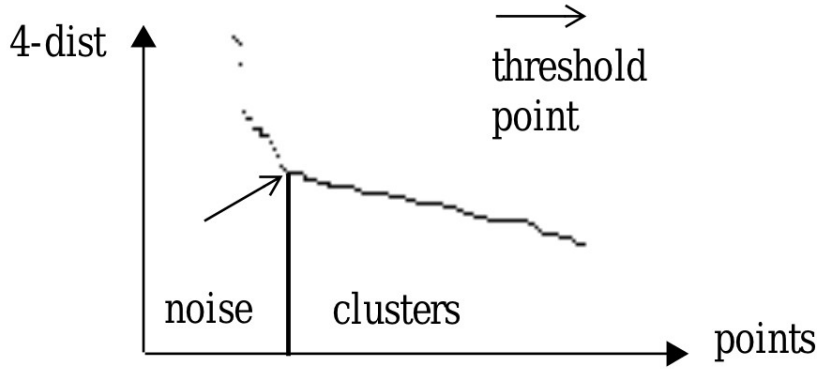


figure 4: sorted 4-dist graph for sample database 3

Time complexity

- As DBSCAN can be implemented as a series of breadth-first search (or depth-first), each point is considered an initial point for search once.
  - As earlier search results are remembered, each of the points are handled only once or twice.
  - However, searching directly density-reachable on each iteration of a search is time-demanding.
    - In brute force implementation each regional query takes O(n), because all distances must be considered => total complexity is O(n*n)
    - Optimizing regional query with efficient spatial search tree, such as R*-tree, reduces time complexity of regional query to O(log n) => total complexity is O(n*log n)

If there is still some time left..

- DBSCAN in action: http://vilikki.kapsi.fi/Opiskelu/KLU/dbscan_s2.avi
- Questions? Comments?
- Thank you for listening :)