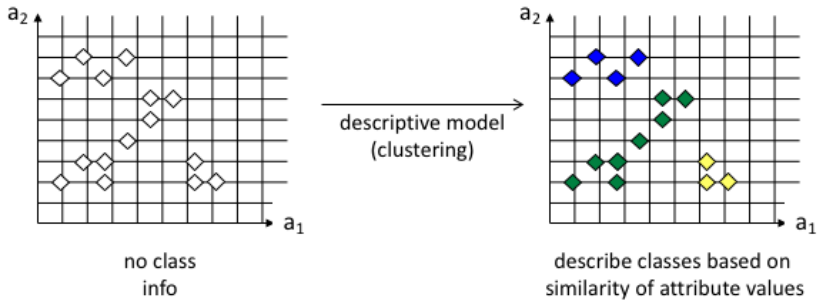
3. CLUSTERING

Clustering and Classification

Given a dataset of *objects* described by *attributes*, build a model that assigns objects to a *class (or label)*



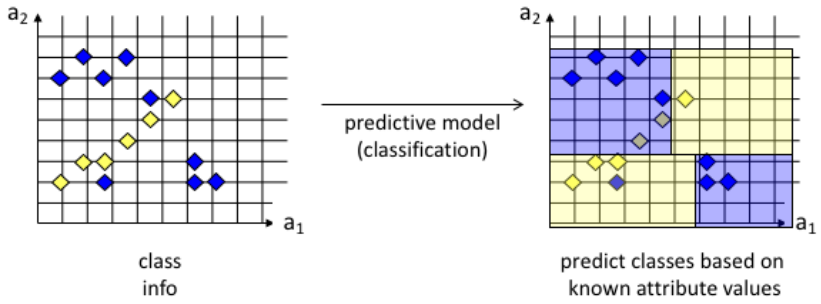
For inferring global models of data collections there exist two types of approaches: descriptive and predictive modeling. We illustrate the difference among them by an example.

We assume that a set of data items (or objects) with two attributes a1 and a2 is given. Assume the global model we are interested in is a classification (or as often said labeling) of the data items.

In descriptive modeling we just know the data items, as indicated by the points in the 2-dimensional grid. A descriptive modeling technique, such as clustering, produces classes, which are not known in advance. For doing this it relies on some criteria that specify when two data items probably belong to the same class. Such a criteria is usually based on a similarity measure.

Clustering and Classification

Given a dataset of *objects* described by *attributes*, build a model that assigns objects to a *class*



class info

predict classes based on known attribute values

A predictive modeling technique, such as classification, starts from a given classification (or labeling) of data items. Using that classification of the dataset the classification method infers conditions on the properties of the data objects, that allow to predict the membership to a specific class. For example, the prediction could be based on a partitioning of the attribute values along each dimension, as shown in the figure on the right. There, first attribute a1 is partitioned into two intervals, and for each of the intervals a different partitioning of the attribute a2 is used to determine the regions corresponding to classes. Misclassifications may occur as seen in the example.

Clustering

Model: partition a set of objects into clusters

- Objects that belong to the same cluster are similar
- Objects that belong to different clusters are dissimilar

Clustering belongs to *unsupervised* machine learning

- Objects do not have class information

Both clustering and classification aim at partitioning a dataset into subsets that have similar characteristics. Different to classification, clustering does not assume any prior knowledge, which are the classes/clusters to be searched for. There exist no class label attributes, that would tell which classes exist. Thus clustering serves in particular for exploratory data analysis with little or no prior knowledge.

One important application of clustering is information retrieval. The basic problem of information retrieval, i.e., finding a set of documents matching a query, can be interpreted as a clustering problem, where the goal is to find two clusters of documents, namely the cluster of relevant ones and the cluster of non-relevant ones. With the tf-idf similarity metrics in fact the tf-measure served to measure intra-cluster similarity for the two document clusters, whereas the idf-measure served to measure inter-cluster dissimilarity of the document clusters.

Usage of Clustering

Information retrieval

- C_1: relevant to query, C_2, C_3: not relevant

Web content classification

- C_1: sport, C_2: politics, C_3: economics ...

Customer behaviour analysis

- C_1: diapers-beer buyers, C_2: Fri afternoon buyers ...

Image/video processing

- C_1: image with faces, C_2: image with animals, …

Clustering finds wide usage in many applications, where the different types of classes of objects (the labels) are not known in advance. The standard process is to first perform a clustering, and then inspecting the result. Inspection of the result may be performed manually (e.g. a user inspects some images in image clusters, and recognizes that in one cluster all, or most, images contain faces), or features of the objects in the cluster are extracted (e.g. frequent keywords if the objects are documents) and a user inspects the result. For example, if the keywords of documents in a cluster would turn around sports, the user would decide the cluster is on sports and would label it as sports.

Clustering

Problem

Given a database D with n data items described by d attributes

Find

Partition of D into k clusters

such that

Intra-cluster similarity is high

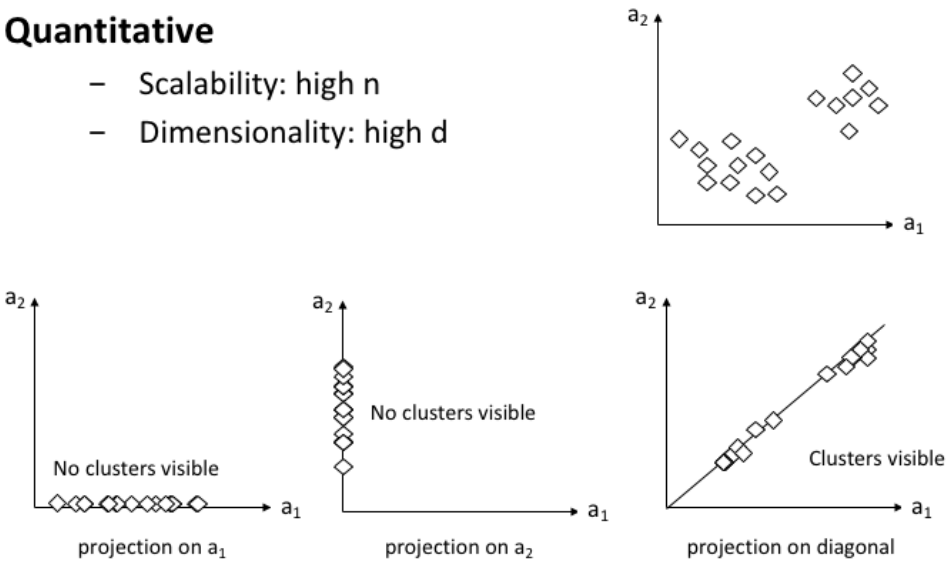
Inter-cluster similarity is low

In its simplest formulation the clustering problem can be described in a way analogous to the vector space retrieval model. Given a database of data items that are represented by d-dimensional vectors (feature vectors), partition the database into k clusters. Not all elements need to be assigned to a cluster, then we consider those elements as noise in the data collection. Frequently used similarity measures include Euclidean distance and Manhattan distance.

Characteristics of Clustering Methods

Quantitative

- Scalability: high n
- Dimensionality: high d



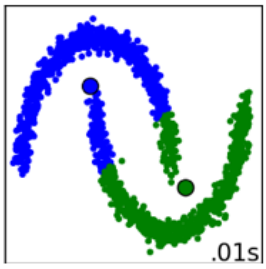
Clearly, clustering methods have to work efficiently for large datasets. Another scalability problem clustering methods have to deal with is dimensionality: in general high-dimensional data (large d) is too sparsely distributed in its high-dimensional space in order to identify meaningful clusters. Therefore the data is often projected into a lower dimensional space. Doing so makes the clustering method sensitive to the choice of dimensions used for projection. The figure illustrates this problem for the case of 2 dimensions:

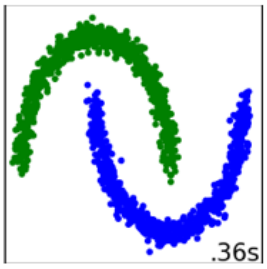
In the 2-dimensional representation we see clearly two clusters. However, if we project either on the a1 or a2 axes, there is no cluster structure visible. Only when we project on the diagonal we will also see the clusters in one dimension. Thus, when projecting into lower dimensions the choice of the subspaces used for projection is crucial. The number of choices for projection dimensions grows on the other hand combinatorial, which makes the problem computationally hard.

Characteristics of Clustering Methods

Qualitative

- Different types of attributes
  - numerical, categorical
- Type of shapes
  - spheres, hyperplanes ...





Qualitative criteria for assessing the performance of a clustering method concern the ability of dealing with continuous as well as categorical attributes, and the type of clusters that can be found. Many clustering methods can detect only very simple geometrical shapes, like spheres, hyperplanes etc.

Characteristics of Clustering Methods

Robustness

- Sensitivity to noise and outliers
- Sensitivity to the processing order

User interaction

- Incorporation of user constraints (e.g., number of clusters, maximal size of clusters)
- Interpretability and usability

Clustering methods can be sensitive both to noisy data and the order of how the records are processed. In both cases it would be undesirable to have a dependency of the clustering result on these aspects which are unrelated to the nature of data in question.

Finally, an important criterion is the ability of how well a clustering method can incorporate user requirements both in terms of information that is provided from the user to the clustering method (in terms of constraints), which can guide the clustering process, and in terms of what information is provided from the method to the user.

Partitioning Methods: Score Function

Model: Partitioning

Given database D of n objects, split D into k sets $C_1,\ldots,C_k$ such that $C_i \cap C_j = \emptyset$ for all $C_i \neq C_j$ and $\bigcup_i C_i = D$

Score function: find C_i that minimises cost function J

$$J = \frac{1}{n}\sum_{i=1}^{k}\sum_{x_j \in C_i} \left\|x_j - \mu_i\right\|^2,\ \mu_i = \frac{1}{|C_i|}\sum_{x_j \in C_i} x_j$$

μ_i = centroid of C_i

More formally partitioning methods can be described as methods that attempt to optimize a cost function for a set of clusters. We show here the formulation for k-Means.

The cost function minimizes the distances of the cluster elements from the cluster representation, which is the centroid of the cluster.

Partitioning Methods

Optimal algorithm: enumerate all partitions and pick the best (not practical)

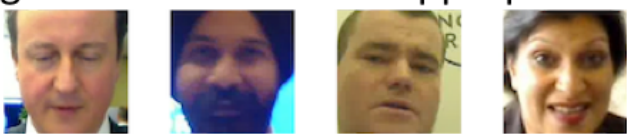
Heuristic algorithms:

- **K-means**: cluster is represented by the ***point*** whose *mean* distance with the objects in the cluster is minimal
- **K-medoids**: cluster is represented by the ***object*** whose *mean* distance with the objects in the cluster is minimal
- **K-medians**: cluster is represented by the ***point*** whose *median* distance with all the objects in the cluster is minimal

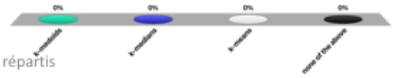
Since an exhaustive enumeration for finding the optimal partitioning is not practical various heuristic methods have been proposed. On class of algorithms are partitioning methods, that represent clusters be selected points and objects. The difference between using points and objects, is that points are not part of the dataset that is being clusters.

Partitioning methods are one of the most basic and widely used approaches to clustering. Partitioning methods attempt to partition the data set into a predetermined number k of clusters while maximizing the intra-cluster similarity.

Suppose we have a dataset of pictures and we want to cluster them. Which partitioning algorithm seems more appropriate?



A. k-medoids
B. k-medians
C. k-means
D. none of the above



K-means Algorithm

Initialise k random points as **cluster centers**

Assign each object to the "nearest" cluster center

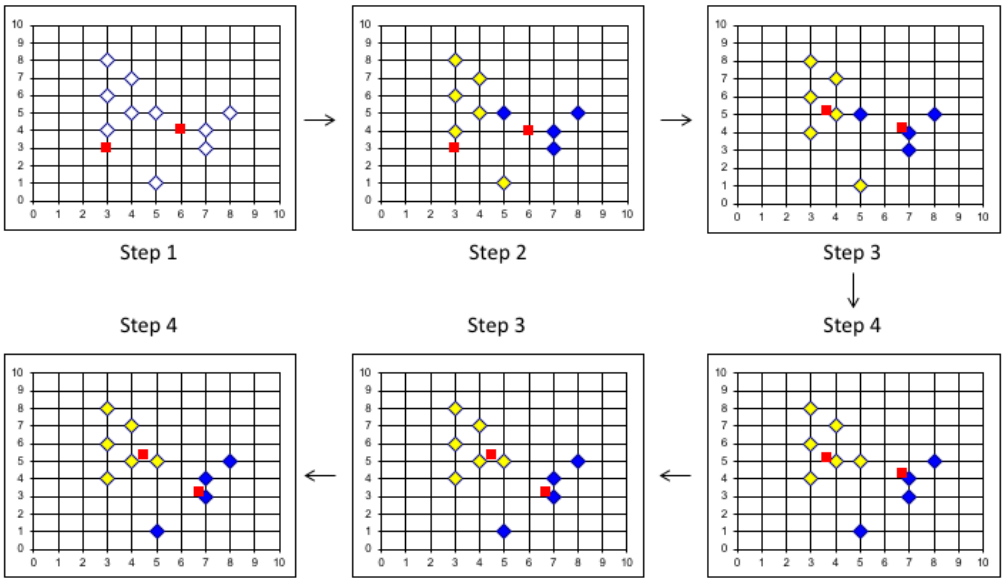
generates a partitioning $C_1,...,C_k$ of D

While partitioning changes

- for each cluster, calculate the centroid of the points and set it as new cluster center
- assign each point to the "nearest" cluster center

In the widely used k-means algorithm clusters are represented by the centroid of their elements. The algorithm starts from some initial partitioning of the data. Initial centroids can be k random poins in the space or k objects (Forgy initialisation). Then, it iteratively reassigns data points to the closests centroids till the algorithm converges. The distances are computed according to a given metrics, e.g. Euclidean distance.

K-means Algorithm: Example



Characteristics of K-means

Advantages

- Efficient: O(tkn)
- n is #objects, k is #clusters, t is #iterations (k,t << n)
- Converges fast

Shortcomings

- Often terminates at a local optimum
- Needs a distance function to compute the mean
- Needs to specify k in advance
- Does not handle noisy data and outliers
- Clusters only have convex shapes

We assess here the properties of k-means following the list of criteria for evaluating clustering methods that we have introduced earlier.

K-means for Categorical Attributes

Needs a distance function to compute the mean:

Matching coefficient for categorical attributes

- a and be are objects with d attributes

$$distance(a,b) = \frac{|\{i \mid a_i \neq b_i\}|}{d}$$

Example:

	domain	location	category
x_1 =	.com	US	sport
x_2 =	.com	CH	econ

distance(x_1, x_2) = 2 / 3

One of the short-comings of k-means is that it relies on the existence of a distance function. Thus, when the attributes are categorical such a distance function needs to be constructed as it is not naturally available. A simple approach to define a distance function in this setting is to count the number of attribute values that do not match (mismatches) and divide by the total number of attributes. Note that this function is a distance function, as it satisfies the properties of a distance function:

$d(x,y) \geq 0$

$d(x,y) = 0$ iff $x = y$

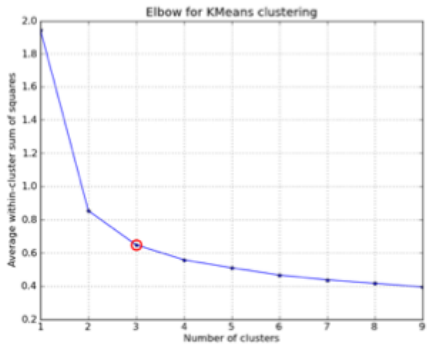
$d(x,y) = d(y,x)$

$d(x,z) \leq d(x,y) + d(y,z)$

Choosing Parameter k for K-means

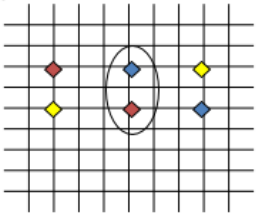
Needs to specify **model parameter** k in advance

- Execute for k = 1, 2, 3 ...
- Plot the score J against k
- Pick k such that J(k) ~ J(k+1)

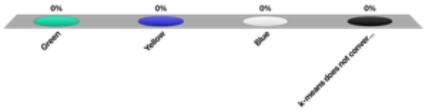


To find a suitable number of k we rely on the following intuition. The higher the value of k, the lower the within-cluster sum of squares. However, creating too many clusters is useless. Thus we want to find the smallest k after which the within-cluster sum of squares decreases “slowly”

What will be the color of the middle points after convergence (k=3)?



A. Green
B. Yellow
C. Blue
D. k-means does not converge

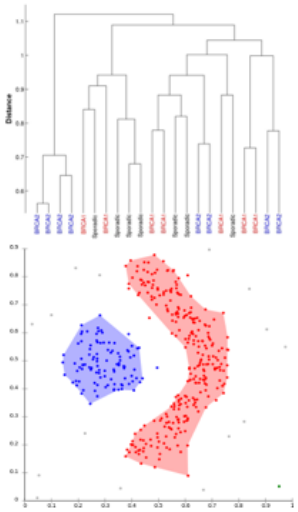


Advanced Clustering

Distance measures for mixed attributes

Other methods

- Density-based clustering
- Hierarchical clustering
- Online incremental clustering



There is a lot of existing work on clustering, including more complex distance functions for mixed (numerical + nominal) attributes and similarity functions that computes “how similar” are two objects (e.g., cosine similarity)

There exists also other methods beyond partitioning. Density-based methods are able to discover clusters of arbitrary shape by checking if the number of objects in the neighbourhood of a given objects is above a certain density threshold. Bottom-up hierarchical methods starts from a number of cluster equal to the number of objects (i.e., each object is a cluster) and then iteratively merge similar clusters. Online incremental methods create clusters in an incremental way, by placing each new incoming object into an existing cluster or into a new one.

Density-based Clustering - DBSCAN

Clustering based on a local, density-based criterion

Properties

- Discovers clusters of arbitrary shape
- Handles noise
- Clusters in one scan
- No predetermined number of clusters
- Model parameters: density parameters

We have identified a number of shortcomings of K-means. Density based clustering addresses a number of those, in particular the need to specify the number of clusters in advance, the possibility to discover non-convex clusters, handling local minima and handling of outliers. Of course, this will not come for free: on the one hand also density-based clustering requires specification of (other) model parameters, and as we will see, the are more costly.

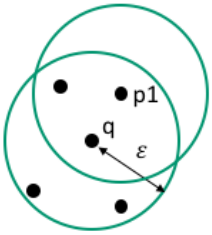
Basic Notions

We assume a distance metric d is given

ε- neighborhood: $N_\varepsilon(q) = \{p | d(p,q) < \varepsilon\}$

A point q is **a core point** if $|N_\varepsilon(q)| \geq \mu$

ε and μ are model parameters



q is a core point with $\mu = 5$
p1 is not a core point with $\mu = 5$

We first introduce the basic notions used in density-based clustering. For each point we can consider its epsilon-neighborhood. If we find within such a neighborhood a certain number of points, we consider such a point as being inside a cluster and call it a core point. The definition of core points also introduces the two model parameters the density-based clustering relies on, epsilon and mu.

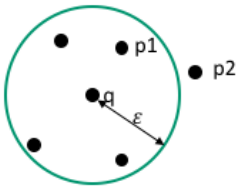
Basic Notions: Directly Density-Reachable

A point p is **directly density-reachable** from q if

$$p \in N_\varepsilon(q) \text{ and } |N_\varepsilon(q)| \geq \mu$$

A point that is directly density-reachable but not a core point is a **border point**

A point that is not directly density-reachable is an **outlier**

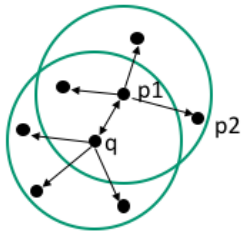


p1 directly density-reachable from q with $\mu = 5$
p2 not directly density-reachable from q with $\mu = 5$
q is not directly density-reachable from p1

Based on the notion of core points, we can next introduce the notion of directly-density reachable. A point is directly-density reachable if it is in the neighborhood of a core point. Since not every point in the neighborhood of a core point is a core point itself, we identify a second type of points, border points. They are directly-density –reachable, informally this means they are part of a cluster, but not core points. Thus they define the border of a cluster. In addition, certain points might not be directly-density reachable at all, these are considered as outliers.

Directly Density-Reachable: Properties

Direct density-reachability induces a **directed graph** on the points

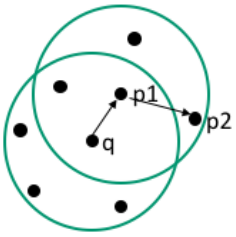


Direct-density reachable induces a directed graph structure on the points. Note that border points are connected with a simple directed link coming from a core point, whereas two core points are connected by bi-directional links.

Basic Notions: Density-Reachable

A point p is **density-reachable** from a point q with ε, μ if there is a chain of points p_1, …, p_n, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i

– Asymmetric relationship



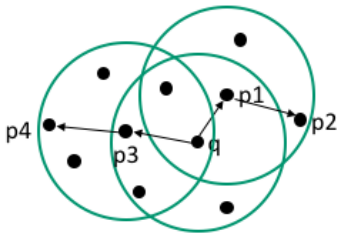
p2 density-reachable from q with $\mu = 5$
p1 and q not density-reachable from p2
q density-reachable from p1

Next we consider the transitive closure of the direct-density reachability, resulting in the notion of density-reachability. A point is density-reachable when it can be reached through a chain of direct density reachability links. Note that this relationship is asymmetric. A border point can be density reachable from a core point, but the inverse is not true, since the border point has no reverse link.

Basic Notions: Density-Connected

A point p is **density-connected** to a point q with ε, μ if there is a point r such that both, p and q are density-reachable from r with ε, μ

– Symmetric relationship

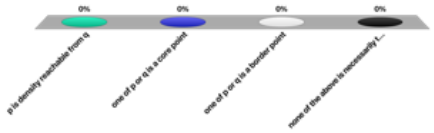


p2 and p4 density-connected with $\mu = 5$

To introduce a symmetric relationships that connects all points within the same cluster, we define the notion of density-connected. Two points are density connected if the can reached both from the some (core) point. This allows then also to connect different border points.

If p and q are density connected, then …

A. p is density reachable from q
B. one of p or q is a core point
C. one of p or q is a border point
D. none of the above is necessarily true



Clusters and Noise

Definition: a **cluster** C satisfies

- **Maximality**: if q in C is a core point, and p is density reachable from q, then also p is in C
- **Connectivity**: any two points in C must be density connected

Properties

- Connectivity implies that a cluster contains at least one core point
- The set of clusters is unique
- Clusters are not necessarily disjoint

Now we have all the notions in order to define what a cluster is in the context of density-based clustering. The basic idea is that density-connected points are considered to belong to the same cluster and that points not density-connected to any other point are considered as noise. To make the definition precise we can state the two properties of maximality and connectivity. With this two properties one can prove that the clusters are uniquely defined,

DBSCAN Algorithm: Initialization

Construct a directed graph G using direct density-reachability

Initialize

- V_{core} = set of core points
- P = set of all points
- set of clusters C = {}

We have now introduced the prerequisites to describe the DBSCAN algorithm for density-based clustering. The algorithm starts by first computing the direct density-reachability graph. This requires to scan the whole database and to compute the neighborhoods of each point. Without any further optimizations this requires each point with every other point, thus a cost that is square in the size of the database. Spatial indexing could be used to reduce this cost. Then three variables are initialized, the set of all core points, the set of all points and the set of clusters found, which initially is empty.

DBSCAN Algorithm: Cluster Construction

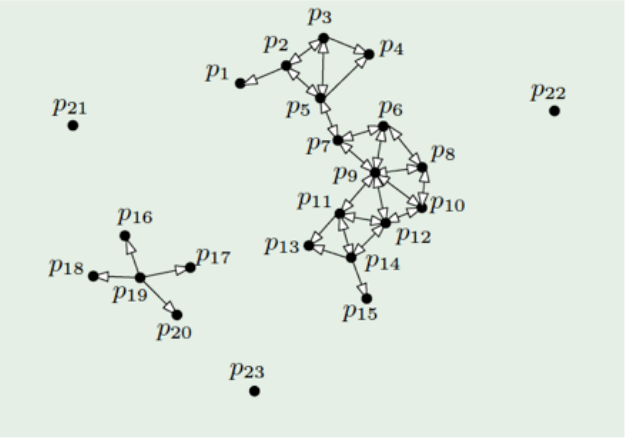
while V_{core} not empty

- select a point p from V_{core} and construct *S(p)*, the set of all points density-reachable from p: breadth-first search on G starting from p
- $C = C \cup \{S(p)\}$
- $P = P \setminus S(p)$
- $V_{core} = V_{core} \setminus S_{core}(p)$, where $S_{core}(p)$ = core points in *S(p)*

Mark remaining points in P as unclustered

For constructing clusters the algorithm packs any core point that has not yet been included in a cluster, and computes all points that are density-reachable from that bound. This set of points can be found be performing a breadth-first search in the directed graph G starting from p and this will return a uniquely defined set of points forming a cluster. After such a cluster has been found the three variables are V_core, C, and P are updated. The new cluster is added to the set of clusters C, the points that have been included in the cluster are removed from P and thus are known to be assigned to a cluster, and the core points contained in the new cluster are removed from V_core and can thus no longer be used as starting point to find a new cluster. Once V_core is empty, we know that no more new clusters can be found. Still, the set P might be non-empty. This means, that those points are outliers that do not belong to any clusters and they are marked as such. Note that the set P is not maintained to indicate the points that need still to be clustered (some points might belong to multiple clusters), but to check which points become never part of a cluster and are finally identified as noise.

Example



Source: Yufei Tao, Chinese University of Hong Kong

First cluster: choose p_2 and compute $S(p_2) = \{p_1,\ldots,p_{15}\}$

Then, $V_{core} = \{p_{19}\}$

Second cluster: $\{p_{16},\ldots,p_{20}\}$

$p_{21},\ldots,p_{23}$ are outliers

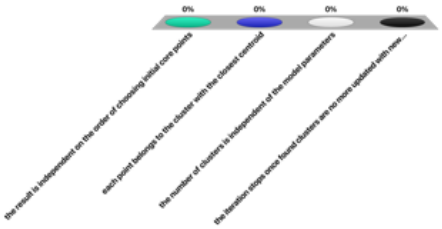
This example illustrates how DBSCAN works. Initially V_core would be {p2, p3, …, p19}. If we select p2 as first core point to construct a cluster we obtain the "big" cluster S(p2). Then the only remaining core point is p19, which gives the second cluster. The remaining points are outliers. The figure also illustrates the difference between core and border points. For example for the big cluster the border points are p1, p4 and p15.

In density-based clustering, which points can belong to multiple clusters?

A. Core points
B. Border points
C. Outliers
D. None

When executing DBSCAN …

A. the result is independent on the order of choosing initial core points
B. each point belongs to the cluster with the closest centroid
C. the number of clusters is independent of the model parameters
D. the iteration stops once found clusters are no more updated with new points



DBSCAN Complexity

Construction of directed graph $O(n^2)$

Construction of clusters $O(n^2)$

Even in the case of dimension d = 3 the worst case complexity is $\Omega(n^{\frac{4}{3}})$

The complexity of the DBSCAN algorithm is O(n^2) which in practice makes it expensive for larger datasets. It is an open problem in data mining finding better algorithms for density clustering, ideally algorithms with running time O(n log(n)^c) for some constant c. It turns out that even in dimension 3 this cannot be achieved, unless some theoretical problems are solved through breakthroughs. The problem can be reduced to the unit-spherical emptiness checking problem, for which today the best algorithm has complexity O(n^4/3 log^4/3 n)