

## Ненадгледано учење – кластеризација података

### DBSCAN

Уопштено говорећи, све методе кластеризације користе исти приступ, тј. прво се израчунава сличност/различитост између објеката, а затим се те вредности користе за груписање тачака у групе (кластере). Овде ћемо се фокусирати на методу просторне кластеризације заснованој на густини са применама над подацима са шумом (DBSCAN).

DBSCAN алгоритам (скраћеница од *Density-Based Spatial Clustering of Application with Noise*) представља кластеровање засновано на густини, односно, на препознавању региона високе густине који су међусобно раздвојени регионима ниске густине. Густина се у овом случају дефинише као број тачака унутар одређеног полупречника ( $Eps$ ). Оно што је посебно добро у вези овог алгоритма је чињеница да алгоритам није ограничен обликом густинских кластера.

Алгоритам као улаз захтева два параметра – растојање  $Eps$  и минималан број тачака  $MinPts$  ( $\mu$ ).

#### Класификација тачака у односу на густину

Инстанце могу бити:

(1) **тачка у језгру** (engl. *core point*)

Тачка  $A$  је „тачка у језгру“ ако се унутар њене  $Eps$  околине налази бар  $\mu$  тачака (укључујући и саму тачку која се посматра). У питању су тачке унутар кластера.

(2) **тачка на граници** (engl. *border point*)

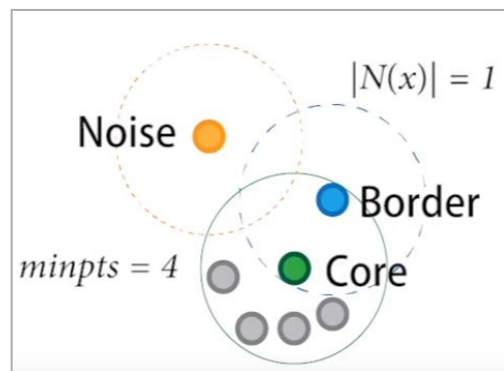
Тачка  $A$  је „тачка на граници“ ако има мање од  $\mu$  тачака унутар своје  $Eps$  околине, али је суседна са „тачком у језгру“, тј. она сама се налази у  $Eps$  околини неке тачке у језгру.

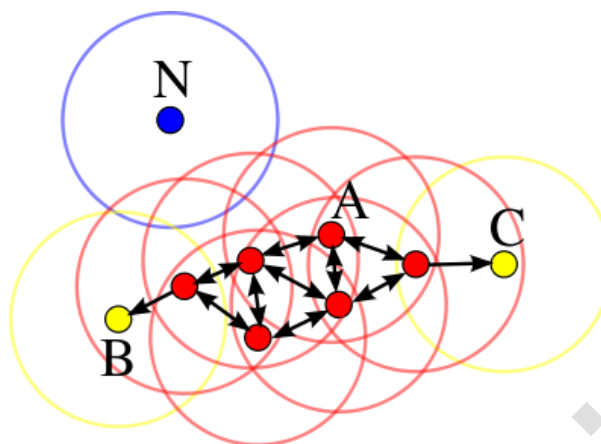
(3) **шум** (engl. *noise point*).

Тачка  $A$  је шум ако није ни тачка у језгру ни тачка на граници.

Илустрација класификације тачака у скупу је дата на сликама 1 и 2.

Слика 1.





**Слика 2.** Три врсте тачака са којима оперише алгоритам DBSCAN (за вредност  $minPts = 4$ ): црвеним кружићима означене су тачке језгра, жутим граничне тачке, а плавим тачке које чине шум.

### Алгоритам:

**Улаз:** Подаци из тренинг скупа  $Train$ , растојање  $Eps$  и  $minPts$ .

**Излаз:** Дисјунктни непразни подскупови  $C_1, C_1, \dots, C_k$

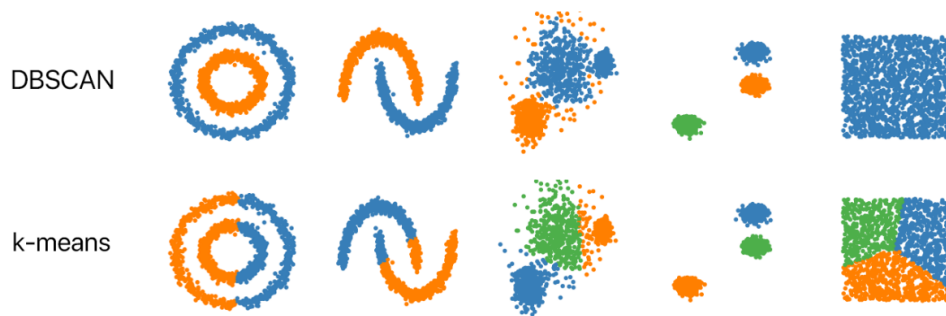
#### Кораци:

1. Формирај скуп  $C$  свих тачака које у својој  $Eps$  околини имају бар  $minPts$  тачака (core points).
2. Формирај скуп  $B$  свих тачака из  $Train \setminus C$  које у својој  $Eps$  околини имају бар једну тачку из  $C$  (border points).
3. Формирај граф  $G$  чији су чворови из  $C \cup B$ , а грана постоји између сваке две тачке које су на растојању највише  $Eps$ .
4. Врати компоненте повезаности графа  $G$  као решење.

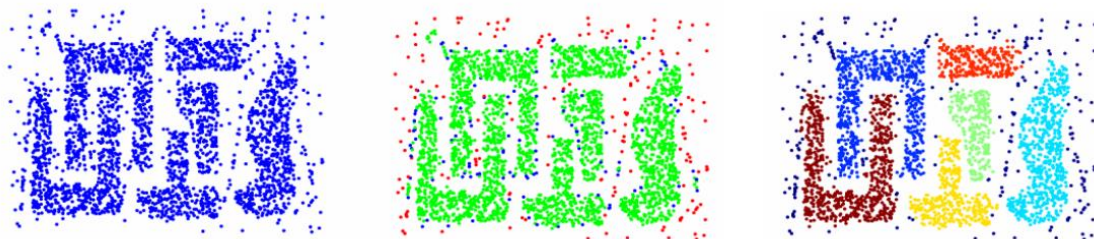
**Напомена:** уколико нека тачка на граници може да се придружи већем броју тачака у језгру, онда бирамо насумично којој тачки у језгру се придружује та тачка на граници.

**Анимација:** <https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html>

На слици 3 дат је пример извршавања алгоритма DBSCAN.



Слика 3. Пример извршавања алгоритма DBSCAN.

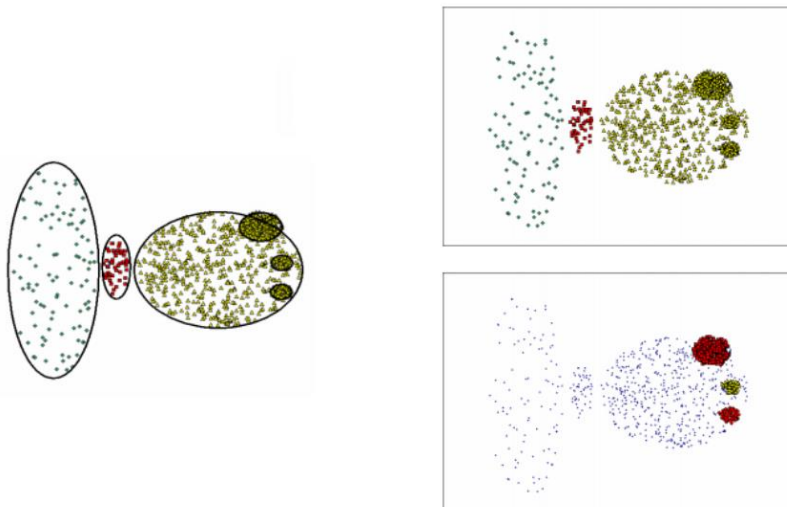


Слика 4. Пример извршавања алгоритма DBSCAN, пример 2. Лево су приказане оригиналне тачке, у средини класификација инстанци, а десно препознати кластери. Параметри алгоритма су  $Eps = 10$  и  $MinPts = 4$ . У средини видимо класификације тачака и то: зеленом су означене тачке у језгру, плавом тачке на граници и црвеном шум. На слици десно су различити кластери приказани различитим бојама.

### Карактеристике алгоритма

Предност алгоритма је свакако могућност препознавања кластера различитих облика и величина, као и отпорност на шум. *Када алгоритам лоше ради?*

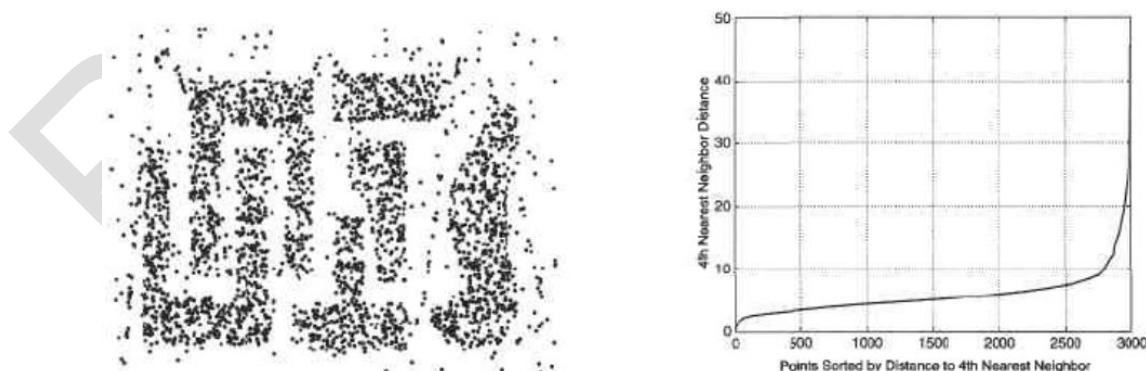
Проблем се јавља у ситуацијама када су кластери различитих густина. Тада се одабиром различитих параметара  $Eps$  и  $minPts$  могу погодније ухватити кластери једне густине, али не и кластери других густина, а такође и да у односу на један кластер који је врло густ неки други кластер (који је редак) може изгледати као шум.



Слика 5. Резултати алгоритма који нису задовољавајући.

### Одређивање вредности $Eps$ и $MinPts$

Иако не постоји конкретан и прецизан метод за одређивање ових параметара, може се применити једна експериментална техника. На слици 6 приказан је скуп података (лево) и график растојања до  $k$ -тог најближег суседа (конкретно, за  $k = 4$ ) (десно). Очекујемо да тачке у језгру имају четвртог најближег суседа релативно близу, а тачке које представљају шум имају четвртог најближег суседа далеко. Ако дати скуп тачака сортирамо по растојању од четвртог најближег суседа, онда можемо добити график приказан на датој слици десно. Приметимо да се на око 2800-тој тачки јавља скок у растојању до четвртог најближег суседа. Можемо рећи да су све тачке након те тачке (у сортираног низу) шум, па бирамо да поставимо вредност за  $Eps$  око вредности 10.



Слика 6. Скуп података (лево) и график  $k$ -растојања за дати скуп података (десно) за одређивање вредности  $Eps$  и  $minPts$ .

## DBSCAN ДЕМО:

[https://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_dbscan.html#sphx-glr-auto-examples-cluster-plot-dbscan-py](https://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html#sphx-glr-auto-examples-cluster-plot-dbscan-py)

## Литература:

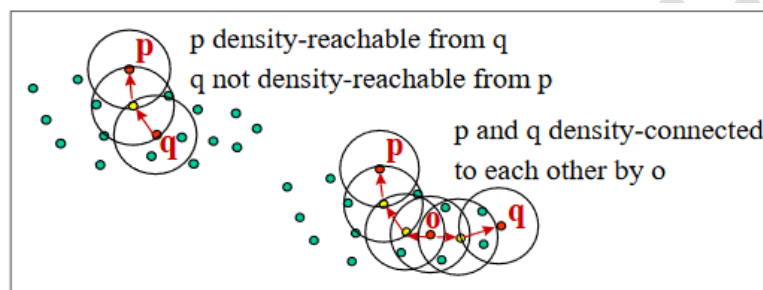
DBSCAN paper: <https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf>

<https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html>

## OPTICS алгоритам

**Ordering points to identify the clustering structure (OPTICS)** је алгоритам за тражење кластера на основу густине у просторним подацима. Основна идеја је слична као код алгоритма DBSCAN, али се овај алгоритам бави једним од главних недостатака DBSCAN алгоритма: детектовањем кластера у подацима који имају различиту густину.

Интересантно је на почетку видети на који начин се дефинише густински кластер. Наредне слике су скриншотови из [објављеног рада](#):



**Figure 2. Density-reachability and connectivity** introduced in the following. For a detailed presentation see [EKSX 96].

**Definition 1:** (directly density-reachable)

Object  $p$  is *directly density-reachable* from object  $q$  wrt.  $\epsilon$  and  $MinPts$  in a set of objects  $D$  if

- 1)  $p \in N_\epsilon(q)$  ( $N_\epsilon(q)$  is the subset of  $D$  contained in the  $\epsilon$ -neighborhood of  $q$ .)
- 2)  $Card(N_\epsilon(q)) \geq MinPts$  ( $Card(N)$  denotes the cardinality of the set  $N$ )

The condition  $Card(N_\epsilon(q)) \geq MinPts$  is called the “core object condition”. If this condition holds for an object  $p$ , then we call  $p$  a “core object”. Only from core objects, other objects can be directly density-reachable.

**Definition 2:** (density-reachable)

An object  $p$  is *density-reachable* from an object  $q$  wrt.  $\epsilon$  and  $MinPts$  in the set of objects  $D$  if there is a chain of objects  $p_1, \dots, p_n$   $p_1 = q, p_n = p$  such that  $p_i \in D$  and  $p_{i+1}$  is directly density-reachable from  $p_i$  wrt.  $\epsilon$  and  $MinPts$ .



**Definition 3:** (density-connected)

Object  $p$  is *density-connected* to object  $q$  wrt.  $\epsilon$  and  $MinPts$  in the set of objects  $D$  if there is an object  $o \in D$  such that both  $p$  and  $q$  are density-reachable from  $o$  wrt.  $\epsilon$  and  $MinPts$  in  $D$ .

Density-connectivity is a symmetric relation. Figure 2 illustrates the definitions on a sample database of 2-dimensional points from a vector space. Note that the above definitions only require a distance measure and will also apply to data from a metric space.

A density-based *cluster* is now defined as a set of density-connected objects which is maximal wrt. density-reachability and the *noise* is the set of objects not contained in any cluster.

**Definition 4:** (cluster and noise)

Let  $D$  be a set of objects. A *cluster*  $C$  wrt.  $\epsilon$  and  $MinPts$  in  $D$  is a non-empty subset of  $D$  satisfying the following conditions:

- 1) Maximality:  $\forall p, q \in D$ : if  $p \in C$  and  $q$  is density-reachable from  $p$  wrt.  $\epsilon$  and  $MinPts$ , then also  $q \in C$ .
- 2) Connectivity:  $\forall p, q \in C$ :  $p$  is density-connected to  $q$  wrt.  $\epsilon$  and  $MinPts$  in  $D$ .

Every object not contained in any cluster is *noise*.

Note that a cluster contains not only core objects but also objects that do not satisfy the core object condition. These objects

У наставку је у кратким цртама описана процедура како се долази до ових кластера, а за више детаља треба консултовати научни рад.

**Основна идеја**

Као и код DBSCAN алгоритма, и OPTICS као улаз очекује два параметра  $Eps$  ( $\epsilon$ ) и  $minPts$ . За разлику од DBSCAN алгоритма, OPTICS такође узима у обзир тачке које су део гушће збијеног кластера, па је свакој тачки додељена и **удаљеност језгра** (*core distance*) која описује удаљеност до  $minPts$ -те најближе тачке:

$$core\_dist_{\epsilon, MinPts}(p) = \begin{cases} UNDEFINED, & \text{if } |N_{\epsilon}| < MinPts \\ MinPts - th \text{ smallest distance in } N_{\epsilon}, & \text{otherwise} \end{cases}$$

Додатно се дефинише и: The **reachability-distance** of another point  $o$  from a point  $p$  is either the distance between  $o$  and  $p$ , or the core distance of  $p$ , whichever is bigger:

$$reachability\_dist_{\epsilon, MinPts}(o, p) = \begin{cases} UNDEFINED, & \text{if } |N_{\epsilon}| < MinPts \\ \max(core\_dist_{\epsilon, MinPts}(p), dist(p, o)), & \text{otherwise} \end{cases}$$

Ако су  $p$  и  $o$  најближи суседи, онда је подразумевамо да  $p$  и  $o$  припадају истом кластеру.

Као што се може приметити из дефиниција, ове две дистанце нису дефинисане у случају да кластер није довољно густ (за изабрано  $\epsilon$ ). За довољно велико  $\epsilon$ , ово се не дешава никад, али  $\epsilon$ -околина претражује велики број тачака, па је комплексност алгоритма  $O(n^2)$ . Зато је параметар  $\epsilon$  потребан да би се „одсекла“ густина оних кластера који више нису занимљиви, а самим тим и убрзао алгоритам.

*Our algorithm OPTICS creates an ordering of a database, additionally storing the core-distance and a suitable reachability-distance for each object. We will see that this information is sufficient to extract all density-based clusterings with respect to any distance  $\epsilon'$  which is smaller than the generating distance  $\epsilon$  from this order.*

## PSEUDOCODE

```
# DB je database.
function OPTICS(DB, eps, MinPts) is
  for each point p of DB do
    p.reachability-distance = UNDEFINED
  for each unprocessed point p of DB do
    N = getNeighbors(p, eps)
    mark p as processed
    output p to the ordered list
    if core-distance(p, eps, MinPts) != UNDEFINED then
      Seeds = empty priority queue
      update(N, p, Seeds, eps, MinPts)
      for each next q in Seeds do
        N' = getNeighbors(q, eps)
        mark q as processed
        output q to the ordered list
        if core-distance(q, eps, MinPts) != UNDEFINED do
          update(N', q, Seeds, eps, MinPts)
```

In update(), the priority queue Seeds is updated with the  $\epsilon$ -neighborhood of  $p$  and  $q$ , respectively:

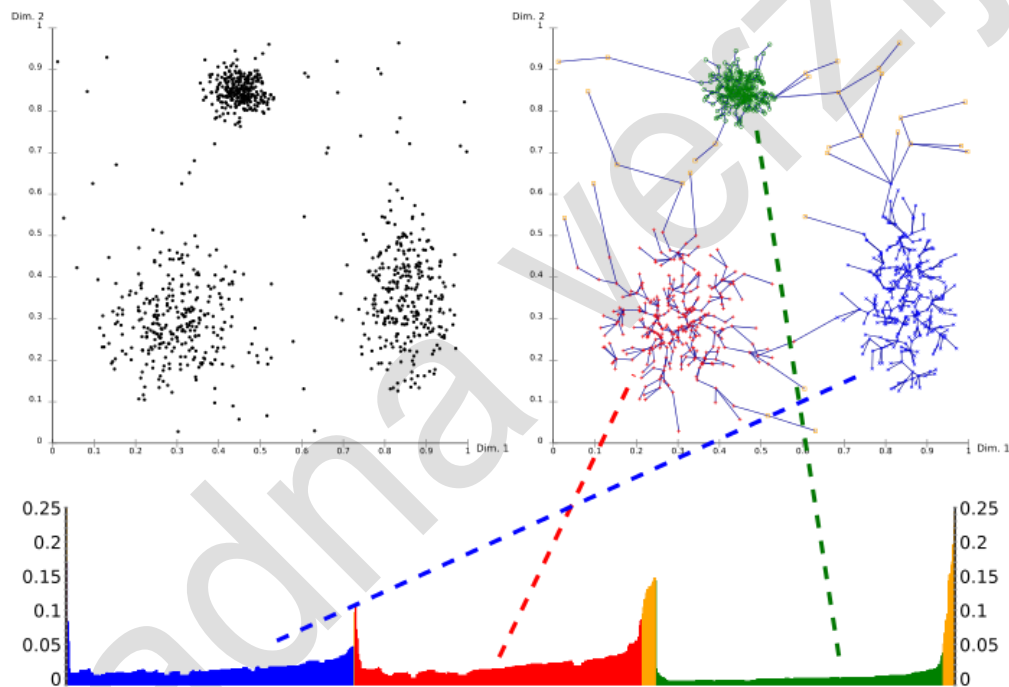
```
function update(N, p, Seeds, eps, MinPts) is
  coredist = core-distance(p, eps, MinPts)
  for each o in N
    if o is not processed then
      new-reach-dist = max(coredist, dist(p,o))
      if o.reachability-distance == UNDEFINED then // o is not in Seeds
        o.reachability-distance = new-reach-dist
        Seeds.insert(o, new-reach-dist)
      else // o in Seeds, check for improvement
        if new-reach-dist < o.reachability-distance then
          o.reachability-distance = new-reach-dist
          Seeds.move-up(o, new-reach-dist)
```



OPTICS hence outputs the points in a particular ordering, annotated with their smallest reachability distance (in the original algorithm, the core distance is also exported, but this is not required for further processing).

### Екстракција кластера

Using a *reachability-plot*, the hierarchical structure of the clusters can be obtained easily. It is a 2D plot, with the ordering of the points as processed by OPTICS on the x-axis and the reachability distance on the y-axis. Since points belonging to a cluster have a low reachability distance to their nearest neighbor, the clusters show up as valleys in the reachability plot. The deeper the valley, the denser the cluster.



The image above illustrates this concept. In its upper left area, a synthetic example data set is shown. The upper right part visualizes the spanning tree produced by OPTICS, and the lower part shows the reachability plot as computed by OPTICS. Colors in this plot are labels, and not computed by the algorithm; but it is well visible how the valleys in the plot correspond to the clusters in above data set. The yellow points in this image are considered noise, and no valley is found in their reachability plot. They are usually not assigned to clusters, except the omnipresent "all data" cluster in a hierarchical result.

Extracting clusters from this plot can be done manually by selecting a range on the x-axis after visual inspection, by selecting a threshold on the y-axis (the result is then similar to a DBSCAN clustering result with the same  $\epsilon$  and minPts parameters; here a value of 0.1 may yield good

results), or by different algorithms that try to detect the valleys by steepness, knee detection, or local maxima. Clusterings obtained this way usually are hierarchical, and cannot be achieved by a single DBSCAN run.

*Preuzeto sa Wikipedia/en*

### Литература:

- Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander (1999). *OPTICS: Ordering Points To Identify the Clustering Structure*. ACM SIGMOD international conference on Management of data. ACM Press. pp. 49–60. CiteSeerX 10.1.1.129.6542

### OPTICS Extensions

- OPTICS-OF is an outlier detection algorithm based on OPTICS. The main use is the extraction of outliers from an existing run of OPTICS at low cost compared to using a different outlier detection method. The better known version LOF is based on the same concepts.
- DeLi-Clu, Density-Link-Clustering combines ideas from single-linkage clustering and OPTICS, eliminating the  $\epsilon$  parameter and offering performance improvements over OPTICS.
- HiSC is a hierarchical subspace clustering (axis-parallel) method based on OPTICS.
- HiCO is a hierarchical correlation clustering algorithm based on OPTICS.
- DiSH is an improvement over HiSC that can find more complex hierarchies.
- FOPTICS is a faster implementation using random projections.
- HDBSCAN is based on a refinement of DBSCAN, excluding border-points from the clusters and thus following more strictly the basic definition of density-levels by Hartigan.