# Семинарска работа по предметот

# Интелигентни системи

## Тема: DBSCAN

## алгоритам за кластерирање

Изработил: Бојан Давитков, 171004

# 1.Вовед

## 1.1.Што е  кластерирање?

Кластерирање(анг. clustering) е метод на групирање на објекти така што објектите кои ќе се наоѓааат во една група(кластер) ќе бидат повеќе слични едни на други отколку објектите кои не се наоѓаат во тој кластер.Бидејќи принципот за “кластер” се дефинира во зависност од проблемот и не е нешто што подлежи на стриктни принципи и правила, постојат многу различни видови и методи на кластерирање, кои во зависност од меѓусебните сличности, се поделени во неколку групи.

## 1.2.Видови на кластерирање

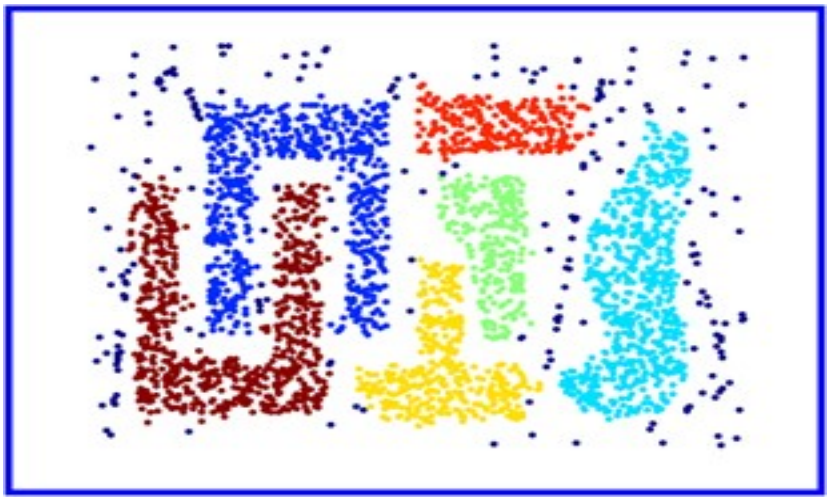
Едни од најчесто користените модели на кластерирање се:

- Конективни модели-во овој случај,кластерите се моделирани според дистанцата(конекцијата) меѓу објектите. Оваа фамилија на методи се разликува според начинот на пресметување на дистанците меѓу објектите.
  Пр:Хиерархиско кластерирање.
- Центроидни модели-кај центроидно-базирано кластерирање, секој кластер е репрезентиран од еден централен вектор или член,кој ненужно мора да биде реален физички член на податочното множество Пр:k-means метод на кластерирање.
- Дистрибуциони модели-Ова е еден од моделите кој највеќе се базира на принципите на статистиката.Алгоритмите кои градат вакви модели ги класифицираат објектите врз основа на нивната припадност кон некоја дефинирана распределба. Пр:EM алгоритмот, кој дава добри резултати на податоци со Гаусова распределба.
- Кластерирање на основа на густина-модел кој ги класифицира податоците врз основа на нивната близина(распореденост). Впрочем,областите кои имаат погуста дистрибуција на објекти е поверојатно да припаѓаат во кластер, додека пак податоците кои лежат во области со помала густина сметаме дека се гранични точки или точки на шум. Пр:DBSCAN алгоритмот.

# 2.Основи на DBSCAN алгоритмот

DBSCAN(**Density-based spatial clustering of applications with noise)** е алгоритам за кластерирање на податоци кој работи на модел базиран на густината на распоредот на податоците во просторот.Се работи за најрепрезентабилен алгоритам во споменатата класа на кластерирачки алгоритми,кој е предложен од страна на Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu во 1996 година.Овој алгоритам може да препознава и создава кластери во нестандардни форми,што во некои случаи би можело да биде пожелно бидејќи не секогаш податоците кои не се меѓусебно сродни не се линеарно или квадратно сепарабилни,или пак,сферични кластери нема да направат доволно добра класификација.

На следната слика,можат да се забележат кластерите во нестандардна форма.



## 2.1. Параметри на DBSCAN

DBSCAN е алгоритам кој на влез прима 2 параметри,а тоа се:

- minPts-минималниот број на точки кој сакаме да се наоѓа во густ регион за тој регион да можеме да го сметаме за посебен кластер,но во некои случаи оваа вредност може да биде и помала,што ќе зависи од поставеноста на податоците и другиот параметар на алгоритамот-eps.
- eps-колкава треба да биде максималната дистанца(растојание) меѓу две точки за да ги сметаме за соседни,а со тоа и како дел од кластер.Eps-соседството на една точка е дадено и со следната формула

$$N_{Eps}(p) = \{q \in D \mid dist(p,q) \leq Eps\}.$$

## 2.2.Поставување на параметрите

Од двата параметри кои ги побарува алгоритамот,*minPts* е веројатно оној кој е полесно да се подеси во однос на податочното множество.Обично се препорачува параметарот да биде двапати поголем од димензионалноста на податоците,т.е. **minPts=2*Dimensionality.** Но постојат случаи каде ова нема да доведе до добра сепарација на податоците,па така во големодимензионални податоци и податоци со многу шум е препорачливо зголемување на *minPts* од почетната формула.Другиот параметар,*eps* е оној кој е потешко да се подеси. Во идеални услови, *eps* би требало да е што е можно помало,но исто така зависи и од одбраната функција на растојание.Обично се препорачува eps да биде поврзан со дистанцата на најблиски соседи(nearest-neighbor distance),така што за н-димензионално податочно множество би се одбрала дистанцата до **2*n** или **2*n-1** најблизок сосед како најповолна естимација за *eps*.Треба да се напомене дека овој параметар е особено сензитивен,така што и мала промена во неговата вредност би довела до сосема различно кластерирање.Мала вредност на eps би ги отфрлила повеќето точки како аутлаери, додека пак голема вредност на овој параметар би довела до спојување на кластери со што повеќето точки би припаѓале во еден кластер.

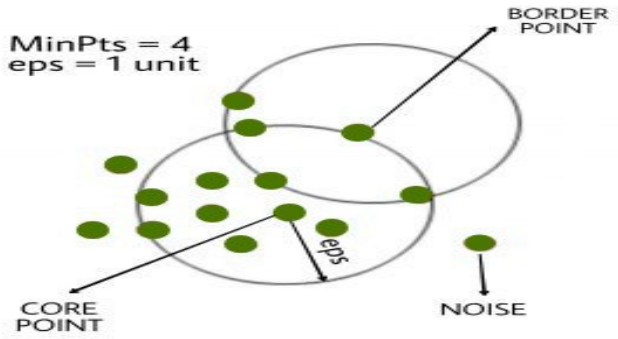
# 3.Точки во DBSCAN

## 3.1.Видови на точки

Во склоп на овој алгоритам се среќаваме со 3 различни видови на точки,а тие се:

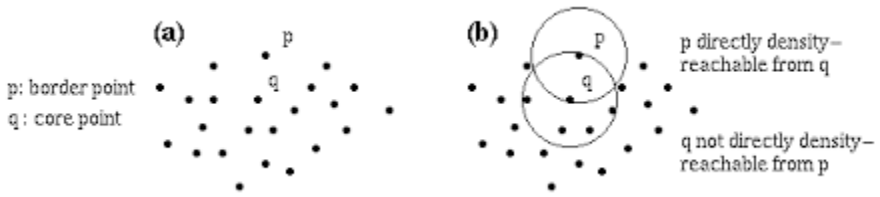
- Јадрена точка-секоја точка која е јадрена во своето *eps*-соседство соддржи повеќе или еднаков број на точки со претходно дефинираниот параметар *minPts*.
- Гранична точка-секоја точка која во своето *eps*-соседство соддржи помалку точки од *minPts* и се наоѓа во eps-соседството на јадрена точка.
- Аутлаер-сите останати точки кои не се јадрени или гранични(не припаѓаат на кластер во податочното множество),или формално кажано

$$noise = \{p \in D \mid \forall\, i: p \notin Ci\ \}$$

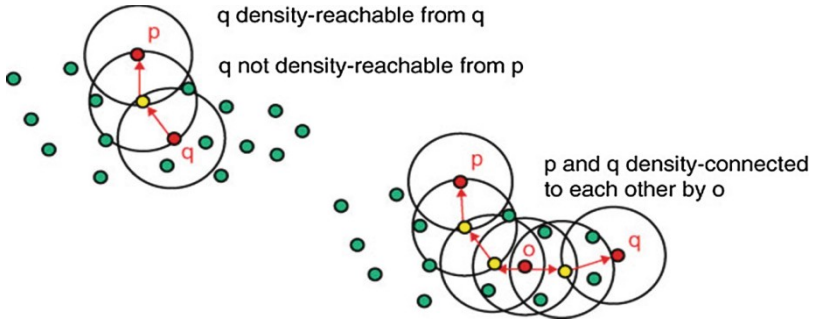



## 3.2.Поврзаност на точките

Бидејќи Eps-соседството на гранична точка има помалку точки од *Eps*-соседството на јадрена точка,*minPts* треба да има релативно мала вредност за да ги опфати сите точки што припаѓаат на еден кластер.Но,поради постоењето на шум во податоците,оваа вредност нема да биде доволно репрезентативна за секој кластер. Затоа,за секоја точка **p** во кластерот **C** постои точка **q** во истиот кластер така што **p** се наоѓа во *Eps*-соседството на **q** кое содржи поголем број на точки од *minPts*.Ако се исполнети овие услови,тогаш **p** е директно-густински достиглива со **q**.Оваа метрика во глобала е симетрична за две јадрени точки,но не мора да е случај ако имаме една јадрена и една гранична точка.




Во друг случај,**точка p е густински достиглива од q,ако за низа од точки $p_1 \ldots p_n$,каде што $p_1=p$ и $p_n=q$,каде $p_{i+1}$ е директно-густински достиглива со $p_i$**,што имплицира дека низата се состои целосно од јадрени точки(освен **q**).Ова е екстензија на директната густинска достигливост и важи истата симетрија само за јадрени точки.Но,условот често не важи токму од импликацијата точките во низата да бидат јадрени точки,така да теоремата паѓа во вода.Но во кластер **C** мора да постои точка од која две точки се густински достигливи.Затоа дефинираме **густинска поврзаност**.Имено,две точки p и q се густински поврзани ако постои точка o од која двете се густински достигливи.




За разлика од претходните релации,оваа релација е симетрична за било кои точки во еден кластер.Со дефинираните поими кои ги означуваат релациите меѓу точките во еден кластер,се назира значењето и дефиницијата на самиот кластер.За множество од податоци **D** непразното подмножество од податоци **C** е кластер ако ги исполнува следните услови

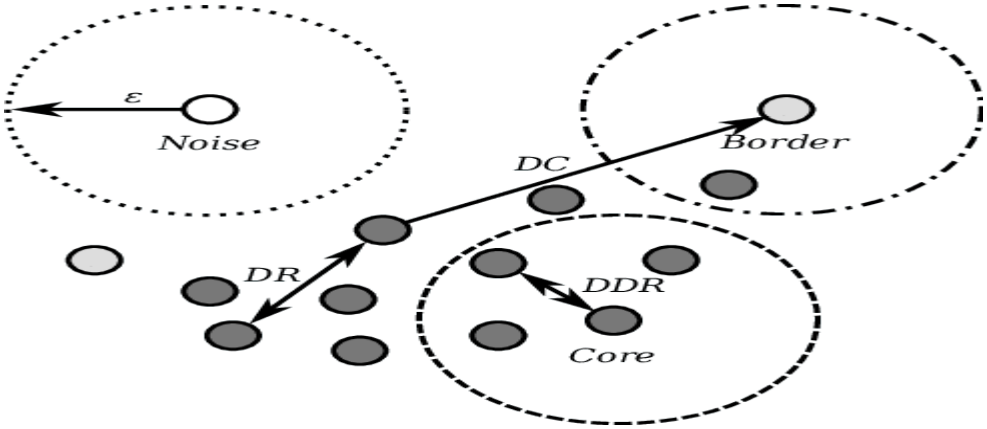
**1) ∀ p, q: if p ∈ C and q is density-reachable from p then q ∈ C. (Maximality)**
**2) ∀ p, q ∈ C: p is density-connected to q (Connectivity)**

Од дефиницијава следи дека поради условите на релациите меѓу точките, секој кластер **C** мора да содржи барем *minPts* број на точки и мора да соддржи јадрена точка во чие Eps-соседство има барем *minPts* број на точки.

## 4.Итерација на DBSCAN алгоритамот

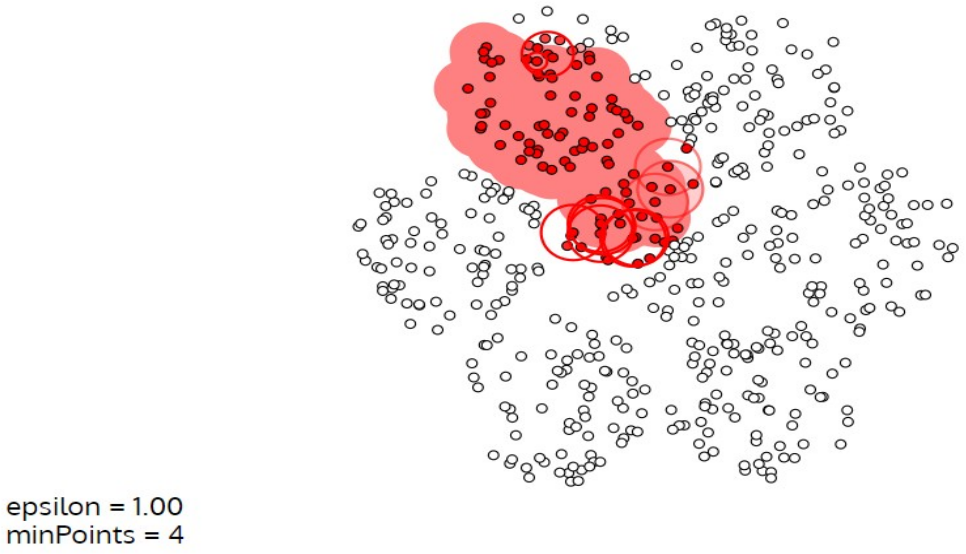
DBSCAN е алгоритам кој строго ги почитува принципите опишани во претходната точка кои важат и за секој друг алгоритам базиран на густина.За да не мора да ги пресметува вредностите на *eps* и *minPts* однапред за секој кластер,што и не е баш едноставна задача, тој користи глобални вредности за параметрите кои важат за сите кластери,што во некои случаи,кога се работи со кластери кои имаат големи разлики во густината, може да предизвика големи проблеми во прецизноста на алгоритмот.Вообичаено,параметрите се поставуваат според кластерот кој има најмала густина на точки во податочното множество.Исто така,алгоритамот не прави естимација на густината измеѓу точките, туку дефинира дека сите точки кои се наоѓаат во *Eps*-соседството на една јадрена точка се дел од истиот кластер се со цел да се заштеди време на дополнителни пресметки.

Најпрво, DBSCAN го дели податочното множество на n димензии од кои тоа е составено. Следно,тој избира случајна точка од множеството и ја означува за посетена точка.Потоа се повикува *rangeQuery* функцијата која го враќа бројот на точки кои се наоѓаат во *Eps*-соседството на таа точка.Треба да се напомене дека DBSCAN во една итерација линеарно ги проверува сите точки дали се лабелирани за посетени,и доколку се,ги прескокнува,но самото пребарување во секоја итерација го чини O(n) време.Доколку бројот на точки избројан со *rangeQuery* функцијата е помал од *minPts* параметарот,тогаш точката е лабелирана како Noise,а во обратен случај таа точка е јадрена и е поставена во нов кластер кој заедно точката и бројот на точките во нејзиното *Eps*-соседство се предаваат како аргументи на *expandCluster* функцијта.На сликата можеме да видиме како се лабелирани ако *minPts* праметарот е поставен на 4.




Потоа,во *expandCluster* функцијата итеративно се проверуваат сите точки кои сме ги испратиле како *NeighborPts* на јадрената точка.Доколку тие биле посетени и не се членови на ниеден кластер,тогаш се придодаваат кон соседната јадрена точка P во кластерот C како нејзини гранични точки.Доколку пак,тие не биле посетени и бројот на точки во нивното *Eps*-соседство е помал од *minPts*(односно биле лабелирани како Noise),се придодаваат кон кластерот C како гранични точки бидејќи во нивното *Eps*-соседство имаат барем една точка(јадрената точка).Имаме и случај каде што бројот на точки во *Eps*-соседството може да биде поголем или еднаков на дефинираниот параметар *minPts*,тогаш таа точка го исполнува условот за јадрена точка и се лабелира како таква.Кога DBSCAN целосно ќе заврши со итерацијата за една точка,преминува на друга сè додека не ги лабелира сите точки во податочното множество.Прифатено правило е дека доколку точка може да биде

гранична точка на повеќе кластери,се додава на првиот од кој е откриена.Итерацијата на точките во *Eps*-соседството на една точка,исто така,се извршува само еднаш за секоја точка и комплексноста на таква итерација е $\mathbf{O(nQ+\sum i*r_i)}$ за индексирана низа од податоци и $\mathbf{O( C+nQ+\sum i*r_i)}$,каде **Q** е комплексноста на *rangeQuery* функцијата, $\mathbf{r_i}$ е комплесноста на нејзиниот резултат,додека пак **C** е времето потребно за индексирање на податоците.На крај,можеме да видиме како изгледа кластерирањето на DBSCAN со параметри *minPts*=4 и *eps*=1.




# 5.Работа на DBSCAN со географски податоци

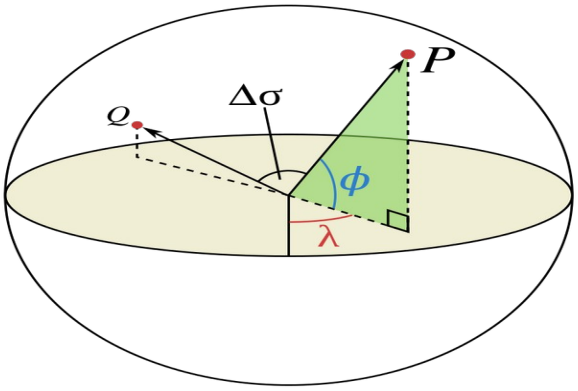
## 5.1.Избор на функција на растојание

Во согласност со неговата успешност на кластерирање на податоци по густина, многу често овој алгоритам се користи во кластерирање на географски точки. Како и многу други алгоритам,и DBSCAN  во најголем број случаи се ослонува на Евклидовото растојание при наоѓање на оддалеченост меѓу две точки.Но,Земјата не е ху-координатен систем,така да оваа формула е неприменлива во случај кога се работи за географска должина и широчина.При таква поставеност,DBSCAN најчесто се ослонува на дистанцата на голем круг.За точка 1 со географска должина $\lambda_s$ и широчина $\varphi_s$ и должина $\lambda_f$ и широчина $\varphi_f$ на точка 2 соодветно,при што се пресметани апсолутните разлики на географските должини $\Delta\lambda$,односно широчини $\Delta\varphi$,аголот меѓу двете точки $\Delta\sigma$ ќе го пресметаме на следниот начин:

$$\Delta\sigma=\arccos(\sin(\varphi1)*\sin(\varphi2)+\cos(\varphi1)*\cos(\varphi2)*\cos(\Delta\lambda))$$

со што го добивме централниот агол меѓу двете точки.За да го добиеме растојанието меѓу нив(или поинаку речено должината на лакот) **d**,ако централниот агол $\Delta\sigma$ е даден во радијани,е со следната формула: $d=r*\Delta\sigma$,каде што е радиусот на сферата(Земјата).

Во прилог е и графичкиот приказ на овие пресметки.Но,оваа формула има недостатоци кај компјутерските системи кои имаат мала прецизност на децимали(Пр: работат со float вредности),имаат проблем со пресметување на централниот агол кога растојанието меѓу двете точки е многу мало.




За подобрување на точноста на пресметување,многу системи ја користат **haversine formula**,која ги решава поголемиот дел од овие проблеми.

$$\Delta\widehat{\sigma} = 2\arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_s\cos\phi_f\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

при што **hav(θ) = sin²(θ /2).**

Со користењето на оваа врска меѓу хаверсин и синусот,оваа равенка е лесно применлива во денешните системи.Но,и оваа формула има своја негативна страна,а таа е кога двете точки за кои бараме должина на лак се дијаметрално спротивни(се наоѓаат на спротивните страни на Земјиниот дијаметар и се познати како антиподални точки).Имено,тогаш погоре споментата функција дава лоши резултати при заокружување на вредноста на централниот агол.Резултати со најмала можност на грешка во било кои случаи дава формулата на Тадеус Винсенти,која се базира на претпоставка дека Земјата има форма на сфероид (плосната сфера);наместо на совршена сфера,што е случај со претходните равенства.

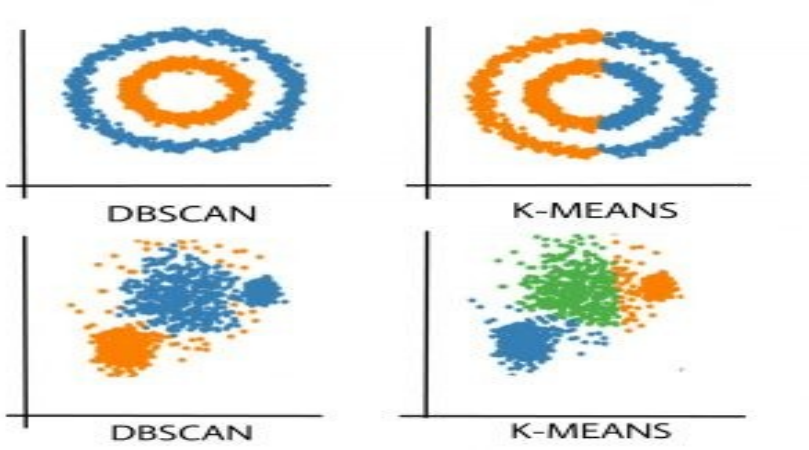
$$\Delta\widehat{\sigma} = \arctan\left(\frac{\sqrt{(\cos\phi_f\sin\Delta\lambda)^2 + (\cos\phi_s\sin\phi_f - \sin\phi_s\cos\phi_f\cos\Delta\lambda)^2}}{\sin\phi_s\sin\phi_f + \cos\phi_s\cos\phi_f\cos\Delta\lambda}\right);$$

# 6.DBSCAN vs k-means алгоритам

DBSCAN и k-means алгоритмите се едни од најупотребуваните алгоритми за кластерирање на податоци.Секој од нив има свои предности и недостатоци и секој од нив е повеќе употреблив од другиот,во зависност од типот на проблемот.Онаму каде DBSCAN е подобар од k-means е во неговата применливост,додека пак адут на k-means е брзината со ги обработува податоците,иако не е многу брз алгоритам

## 6.1Предности на DBSCAN во однос на k-means

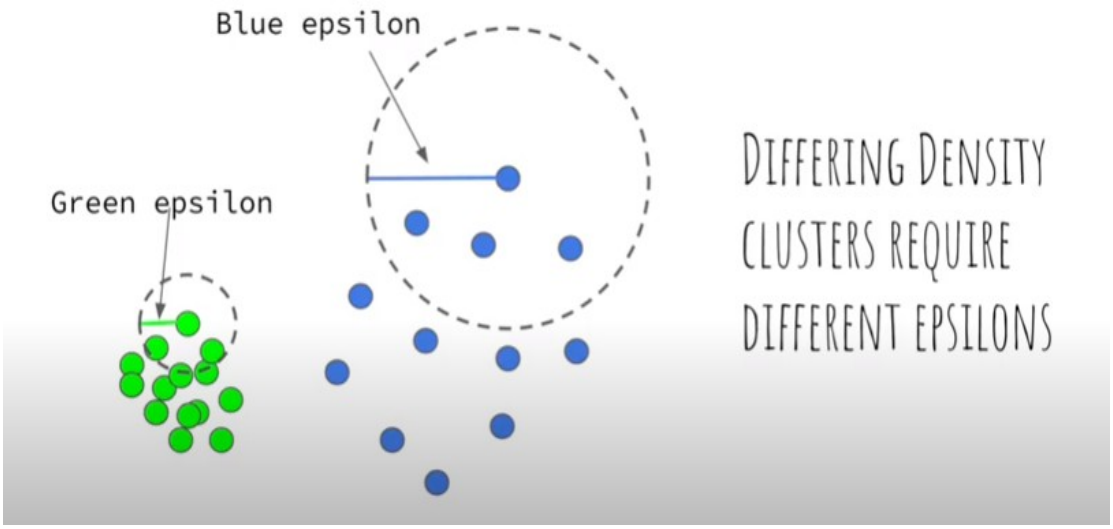
- ***Форма на кластери***-k-means може да формира само кластери во сферичен облик, иако некогаш тоа е далеку од оптималното решение.DBSCAN пак, секогаш ги бара најгустите региони на распространетост на точките.




- ***Број на кластери***-самото име на k-means(k) ни прецизира дека однапред треба да го специфицираме бројот на кластери,што во некои случаи е неприменливо. Од друга страна,иако кај DBSCAN комбинацијата на *eps* и *minPts* имплицитно го одредува бројот на кластери,не е секогаш клучна при донесувањето на одлуката.
- ***Сензитивност на аутлаери***-Аутлаерите се голем проблем кај k-means од причина што влијаат на пресметката на центроидите и мора да припаѓаат во кластер. DBSCAN успешно ги детектира аутлаерите и ги остава надвор од кластерите и тие како такви,можат да послужат за понатамошна анализа.

## 6.2.Недостатоци на DBSCAN во однос на k-means

- ***Кластери со различна густина***- ако во непосредна близина имаме кластери кои имаат многу голема разлика во густина на точките,ќе треба повторно да се постават параметрите eps и minPts,во спротивно,ќе добиеме многу лоши резултати.Пример за таква ситуација може да се забележи на сликата во продолжение.
- ***Брзина***-кај мали податочни множества предноста на k-means е незначителна, но како што се зголемува големината и димензионалноста на множеството, DBSCAN заостанува зад својот конкурент.
- ***Примена***-k-means работи многу подобро и поефикасно ако сакаме да го искористиме за регресиона анализа со помош на метод на најамли квадрати за да најдеме меѓусебна поврзаност и зависност на податоците.



## 7.Псевдокод ( алгоритам на DBSCAN )

```
DBSCAN(D,eps,minPts):
  C=0
  for each unvisited point P in dataset D:
    mark P as visited
    NeighborPts=regionQuery(P,eps)
    if sizeof(NeighborPts)<minPts:
      mark P as noise
    else:
      C=next cluster
      expandCluster(P,NeighborPts,C,eps,minPts)

expandCluster(P,NeighborPts,C,eps,minPts):
  add P in cluster C
  for each point P` in NeighborPts:
    if P` is not visited:
      mark P` as visited
      NeighborPts`=regionQuery(P`,eps)
      if sizeof(NeighborPts`)>=minPts:
        NeighborPts=NeighborPts joined with NeighborPts`
    else if P is not yet member of any cluster:
      add P` to cluster C

regionQuery(P,eps):
  return all points within P eps-neighborhood(including P)
```

# 8.Користена литература


1. “DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN”-Erich Schubert,Jörg Sander,Martin Ester,Hans Peter Kriegel,Xiaowei Xu

Линк: https://dl.acm.org/doi/10.1145/3068335

2.” A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”- Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu

Линк: http://www2.cs.uh.edu/~ceick/7363/Papers/dbscan.pdf

3. **“**Great-circle distance”-Wikipedia

Линк: https://en.wikipedia.org/wiki/Great-circle_distance

4. “DBSCAN: What is it? When to Use it? How to use it.”-Evan Lutins

Линк: https://medium.com/@elutins/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818

5.” Visualizing DBSCAN Clustering”-Naftali Harris

Линк: https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

6.”DBSCAN”-Wikipedia

Линк: https://en.wikipedia.org/wiki/DBSCAN

7.” The 5 Clustering Algorithms Data Scientists Need to Know”-George Seif

Линк: https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

8.” DBSCAN Clustering in ML | Density based clustering”

Линк: https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/

9.” Pseudocode of the DBSCAN algorithm“

Линк: https://www.researchgate.net/figure/Pseudocode-of-the-DBSCAN-algorithm_fig2_325059373

10.” Calculate distance, bearing and more between Latitude/Longitude points”

Линк: https://www.movable-type.co.uk/scripts/latlong.html