DEFINIZIONE DI DENSITÀ

Per poter definire cluster basati sulla densità bisogna definire come si intenda densità:
Abbiamo due parametri:

1) ε: Massimo raggio per cui definire il vicinato
2) MinPoints: Minimo numero di punti per far si che il vicinato sia denso.

CORE OBJECT

È un oggetto che entro il raggio ε ha un vicinato che comprende almeno MinPoints.

DIRECTLY DENSITY REACHABLE

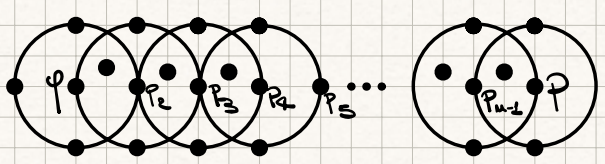
Un oggetto p è definito directly density-reachable da un altro oggetto q se:

1) $p \in \varepsilon$-vicinato(q)
2) q è un CORE OBJECT

DENSITY REACHABLE

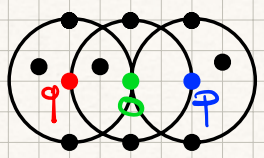
Un oggetto p è definito density-reachable da un altro oggetto q se esiste una catena di oggetti $p_1 \dots p_m$ con $p_1 = q$ e $p_m = p$ tale per cui p_{m+1} è directly density-reachable da p_i.

Esempio: $\varepsilon = 1$ e MinPoints $= 3$

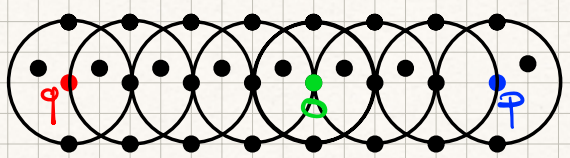


DENSITY-CONNECTED

Un oggetto p è density-connected a un altro oggetto q se esiste un oggetto O tale per cui sia q che p sono density-reachable da O.



$\varepsilon = 1$ e MinPoints = 3



DBSCAN

L'acronimo sta per Density-Based Spatial Clustering of Application with Noise.

L'idea alla base è semplice, un cluster è definito come un massimo insieme di density-connected objects.

Ogni qual volta trova un oggetto che ha nel ε-vicinato più di minPoints un nuovo cluster viene generato con quel core-object. Poi iterativamente DBSCAN colleziona gli oggetti directly density-reachable dal core-object trovato. Questo può far venire la necessità di fare il merge di qualche density-reachable cluster.

Stop-Condition: Non ci sono nuovi punti da aggiungere a un cluster.

DEFINIZIONE DI CLUSTER

Un sottoinsieme C in D è un cluster se:

1) $\forall o_1, o_2 \in C$: o_1 e o_2 sono density-connected
2) Non esiste nessun oggetto $o \in C$ e $o' \in (D-C)$ tale per cui o e o' sono density-connected

COMPLESSITÀ : $O(n^2)$

PARAMETRI:

1) ε
2) Min Points
3) D

ALGORITMO

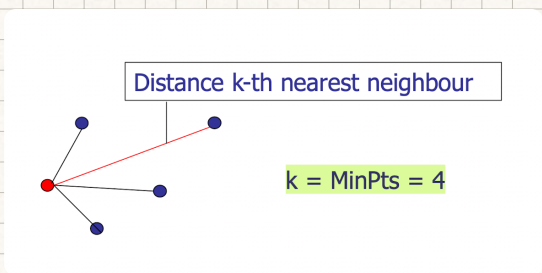
```
DBSCAN(ε, minPoints)
{
  while(Non ho visitato tutti gli oggetti)
  {
    p = oggetto Random non visitato.
    p = visitato
    if(ε-vicinato(p).size() ≥ minPoints)
    {
      C = nuovo Cluster
      C.aggiungi(p)
      N = ε-vicinato(p)
      for(∀p' ∈ N)
      {
        if(p' non è stato visitato)
        {
          p' = visitato
          if(ε-vicinato(p').size() ≥ minPoints)
            N.add(ε-vicinato(p'))
          if(p' non appartiene a un cluster)
            C.aggiungi(p')
        }
      }
      return C
    }
    else { p' = rumore }
  }
}
```

CONTRO

1) Molto sensibile ai parametri ε e MinPoints. Se il dataset presenta densità diverse nello spazio DBSCAN potrebbe perdersi qualche cluster.

Come determinare ε e MinPoints?

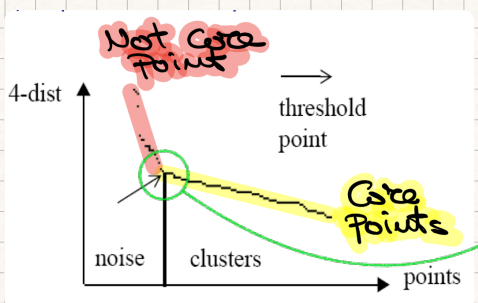
Usando un approccio euristico:
Possiamo settare MinPoints usando la distanza fra il punto preso in considerazione e il suo K-NN. Quindi diverso per ogni punto in D.
MinPoints sarà uguale al numero di punti all'interno del raggio dato da $\varepsilon = dist(p, K\text{-}NN)$, ovvero K.



Passi metodo euristico

1) Ordiniamo i punti in ordine decrescente di $dist(p, K\text{-}NN)$

2) $\forall p \in D \quad \varepsilon = dist(p, K\text{-}NN_p)$, MinPoints = K.
Così tutti i punti con K-dist minore saranno CORE-OBJECT



3) Settiamo i nostri parametri con quelli inerenti al primo punto in cui la K-dist non cambia più drasticamente.

PROBLEMA

Usare parametri generali per definire la densità, può non essere ottimale perché non tutto il DB ha la stessa densità.

SOLUZIONE 1: HIERARCHICAL CLUSTER CON SINGLE-LINK

Questa soluzione ha però il difetto che due cluster separati dai pochi punti più vicini possono non essere propriamente separati. Il risultato è un dendrogram difficile da capire.

SOLUZIONE 2: DENSITY-BASED PARTITION ALGORITHM

Il problema è la difficoltà di settare i parametri adeguatamente.

SOLUZIONE BUONA:

Usare un algoritmo che produce un ordine speciale dei punti del dataset, basato sui livelli di densità presenti fra i vari potenziali cluster. Che è quello che fa OPTICS