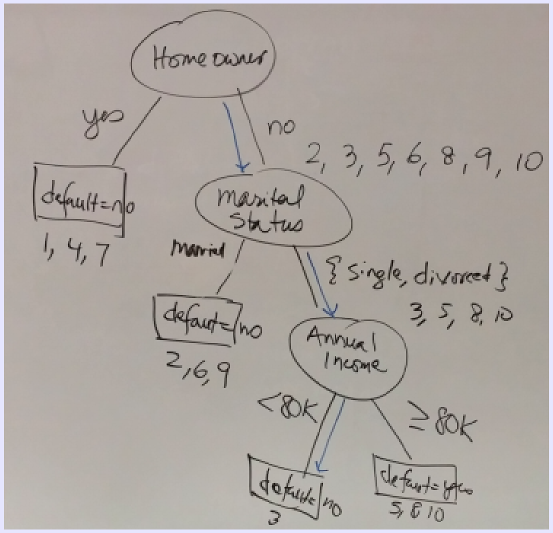
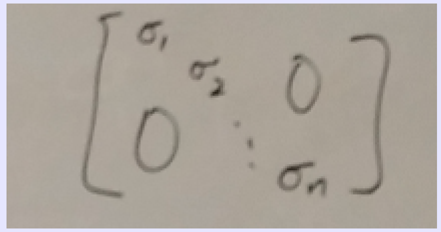
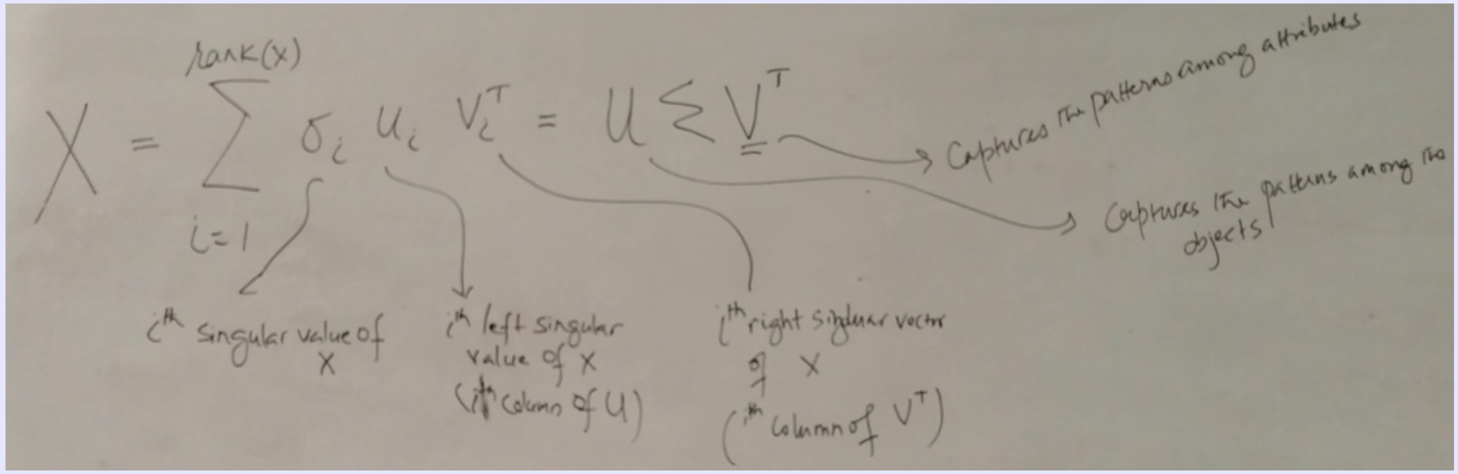
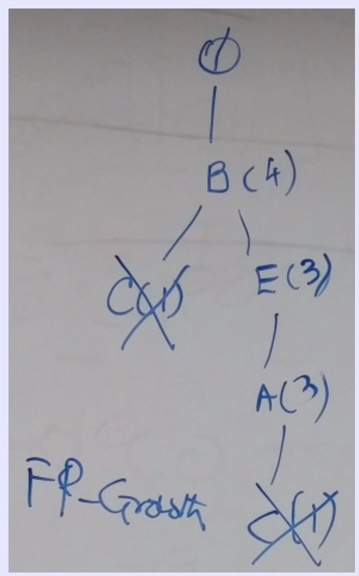
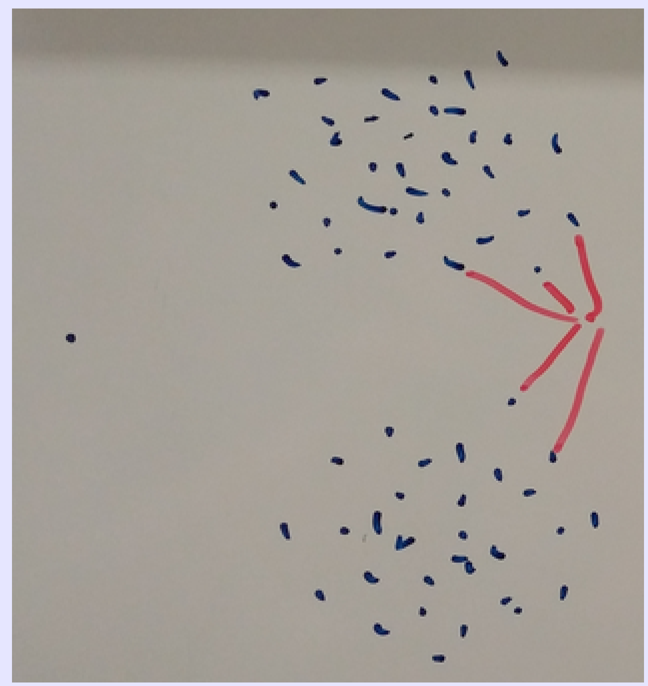










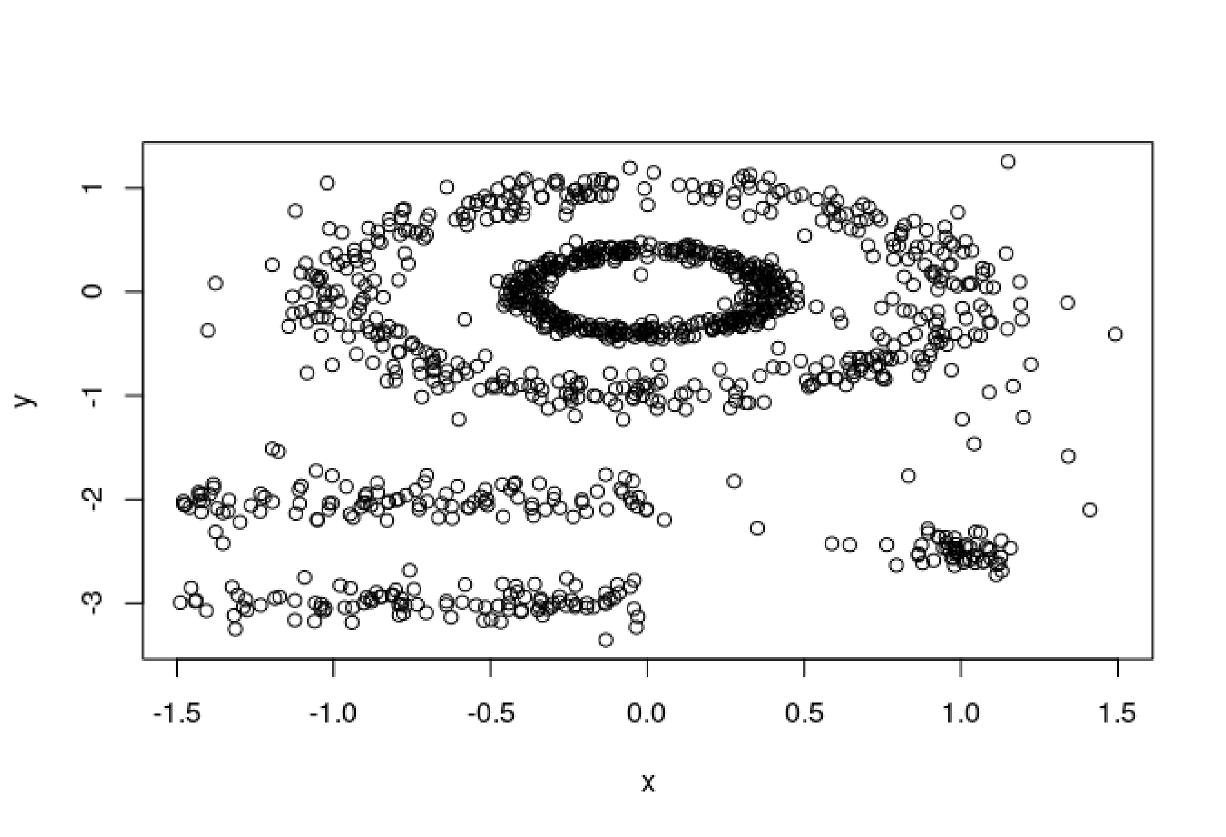
CS 422: Data Mining

Vijay K. Gurbani, Ph.D.,
Illinois Institute of Technology

Clustering I

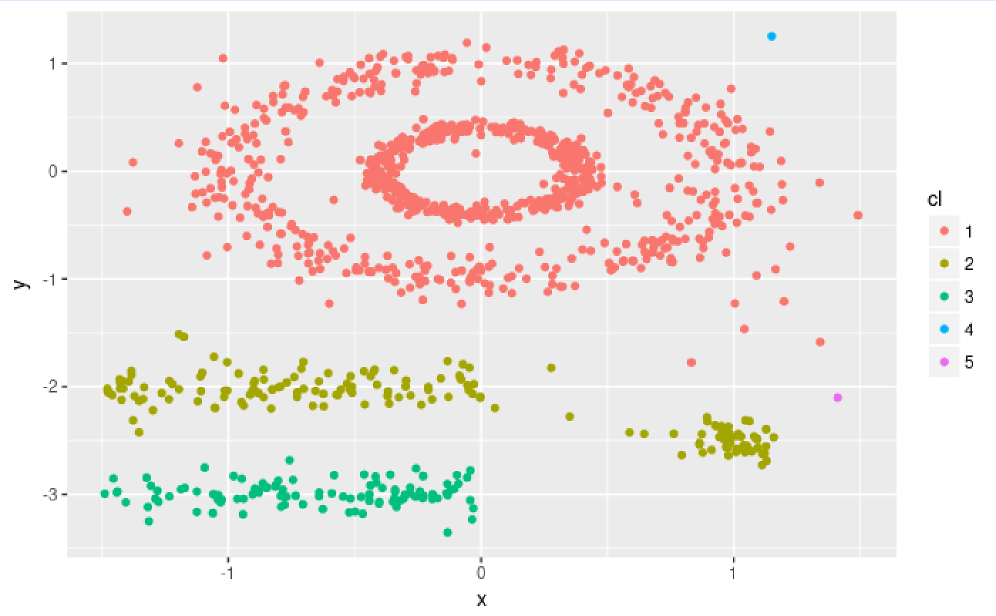
Density-based clustering: dbscan

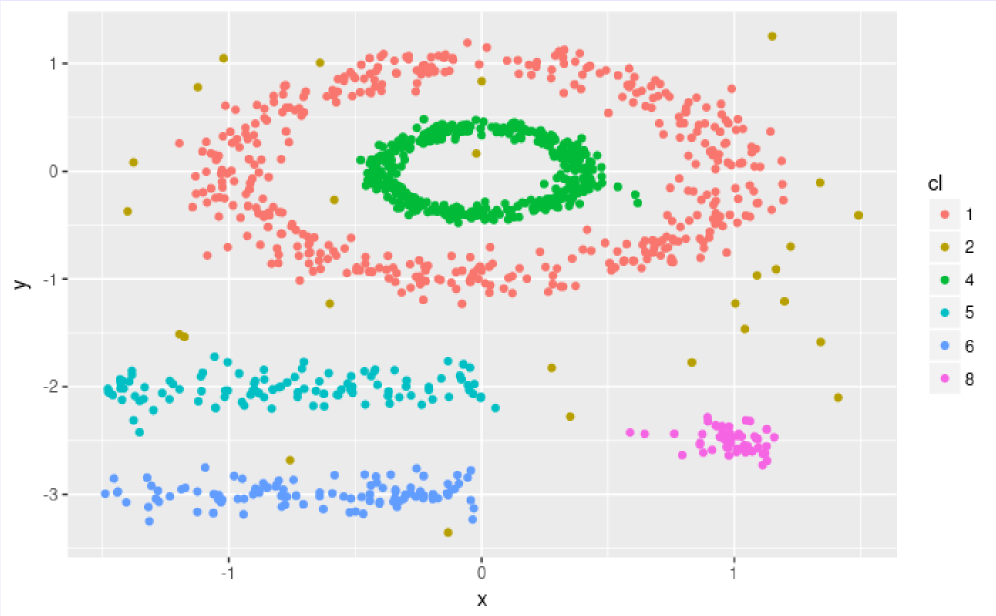
- K-means and hierarchical clustering do not gracefully handle non-globular clusters as shown on the left.
  - They will **find** clusters, but the resulting clusters may not be what we need.



Density-based clustering: Miscellaneous

- Hierarchical clustering do on globular data:





- Observations:
  - Hierarchical clustering is not able to eliminate what would be called "noise" points in DBSCAN. So these become part of a cluster.
  - With 5 clusters hierarchical clustering is unable to discern the two nested clusters. It considers them as one.

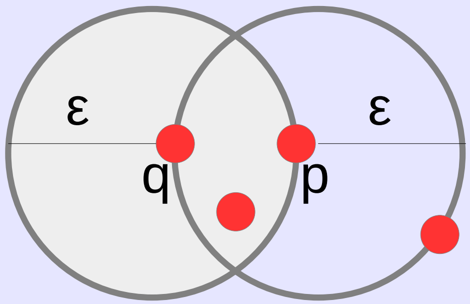
Code:
hierarchical-clustering-globular-data.Rmd

Density-based clustering: dbscan

- DBSCAN is a density-based clustering algorithms parameterized by:
  - A radius (*eps*, ε, calculated empirically), or a neighbourhood;
  - Number of neighbouring points (*MinPts*, specified by user).

Density-based clustering: dbscan

- ε-Neighbourhood: Objects within a radius ε from a source object: $N_\epsilon(p) : \{q \mid d(p,q) \leq \epsilon\}$
- **Density**: If ε-neighbourhood of a source point (object) contains at least MinPts other points (object), then the source point is in a “high-density” area.



- Density of *p* is high (MinPts = 4)
- Density of *q* is low (MinPts = 3)

Density-based clustering: dbscan

- DBSCAN divides all points in:
  - Core point: a point that has at least *MinPts* within an ε.
  - Border point: a point that is not a core point, but is in the neighbourhood of a core point.
  - Noise point: Points that are neither core or border points.

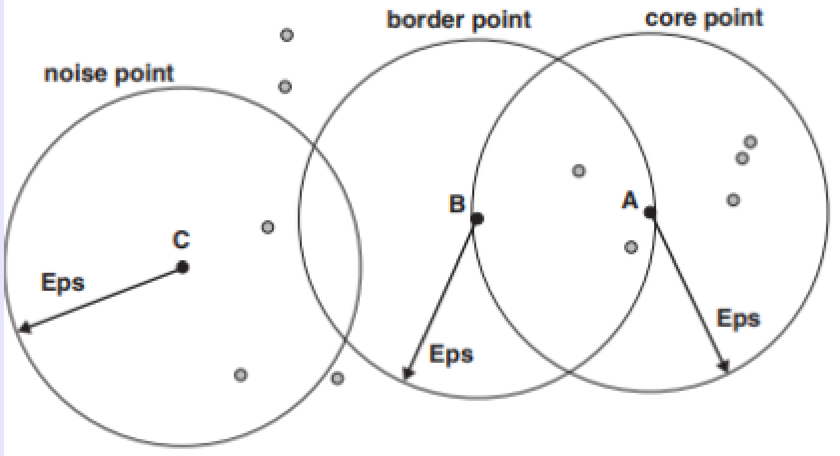


Figure 8.21. Core, border, and noise points.

Density-based clustering: dbscan

Schubert et al. (2017), “DBSCAN Revisited, Revisited: Why and how you should (still) use DBSCAN,” ACM Transactions of Database systems 42(3), 2017.

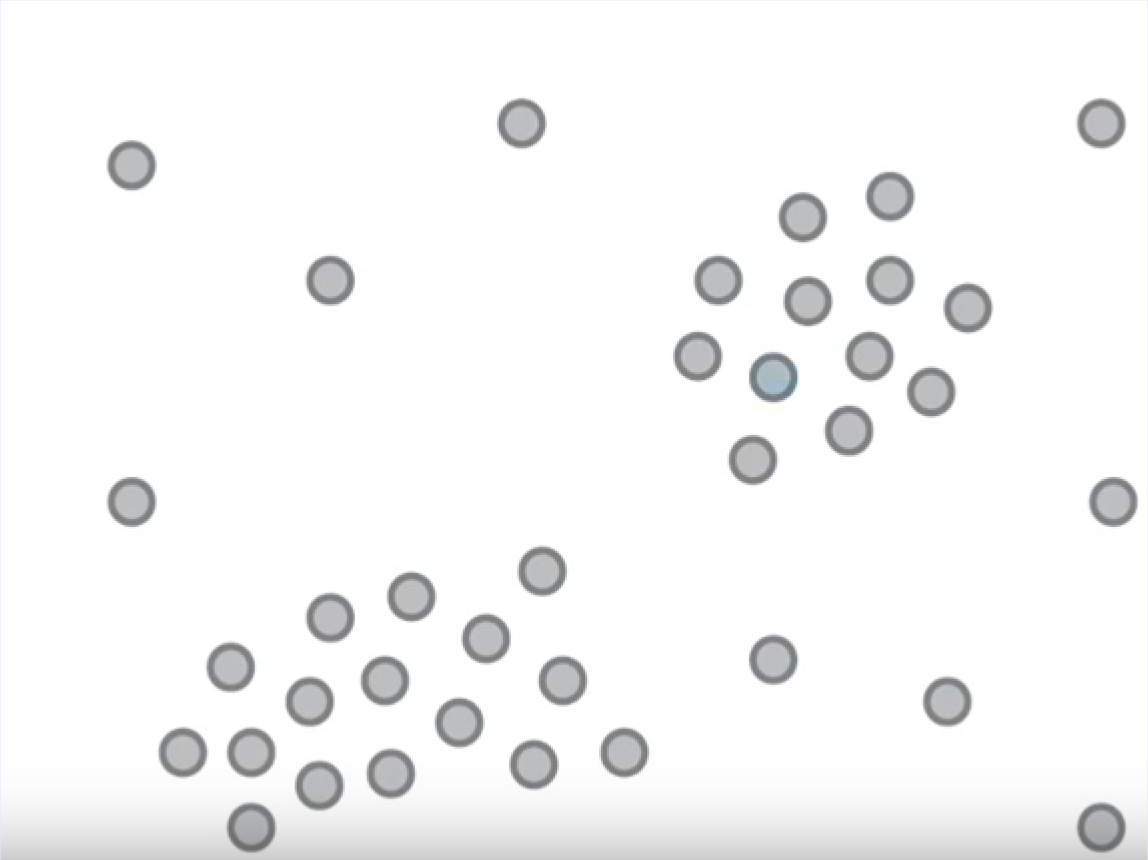
```
DBSCAN(DB, distFunc, eps, minPts) {
   C = 0                                                  /* Cluster counter */
   for each point P in database DB {
      if label(P) ≠ undefined then continue               /* Previously processed in inner loop */
      Neighbors N = RangeQuery(DB, distFunc, P, eps)      /* Find neighbors */
      if |N| < minPts then {                              /* Density check */
         label(P) = Noise                                 /* Label as Noise */
         continue
      }
      C = C + 1                                           /* next cluster label */
      label(P) = C                                        /* Label initial point */
      Seed set S = N \ {P}                                /* Neighbors to expand */
      for each point Q in S {                             /* Process every seed point */
         if label(Q) = Noise then label(Q) = C            /* Change Noise to border point */
         if label(Q) ≠ undefined then continue            /* Previously processed */
         label(Q) = C                                     /* Label neighbor */
         Neighbors N = RangeQuery(DB, distFunc, Q, eps)   /* Find neighbors */
         if |N| ≥ minPts then {                           /* Density check */
            S = S ∪ N                                     /* Add new neighbors to seed set */
         }
      }
   }
}
```

```
RangeQuery(DB, distFunc, Q, eps) {
   Neighbors = empty list
   for each point P in database DB {                      /* Scan all points in the database */
      if distFunc(Q, P) ≤ eps then {                      /* Compute distance and check epsilon */
         Neighbors = Neighbors ∪ {P}                      /* Add to result */
      }
   }
   return Neighbors
}
```

Density-based clustering: dbscan

Algorithm 8.4 DBSCAN algorithm.

1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within *Eps* of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.

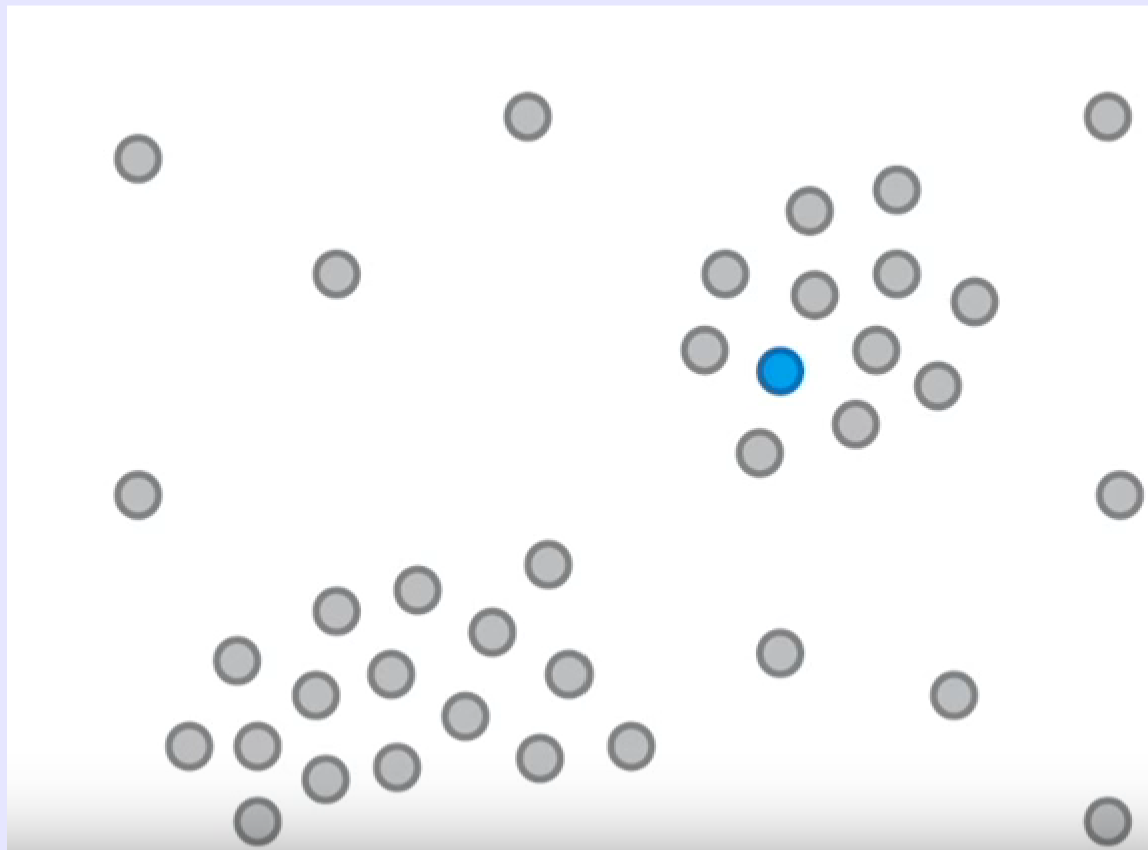


Screenshots from https://www.youtube.com/watch?v=5E097ZLE9Sg

Density-based clustering: dbscan

Algorithm 8.4 DBSCAN algorithm.

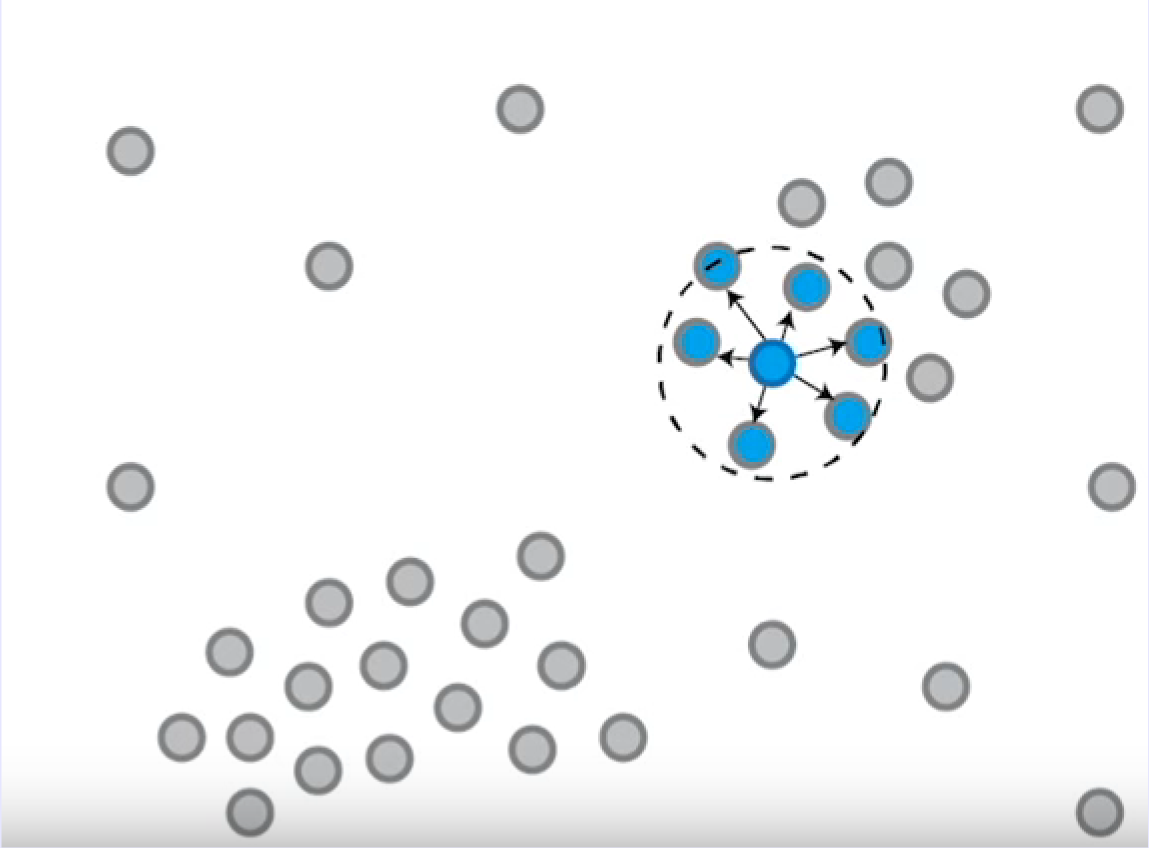
1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within *Eps* of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.



Density-based clustering: dbscan

Algorithm 8.4 DBSCAN algorithm.

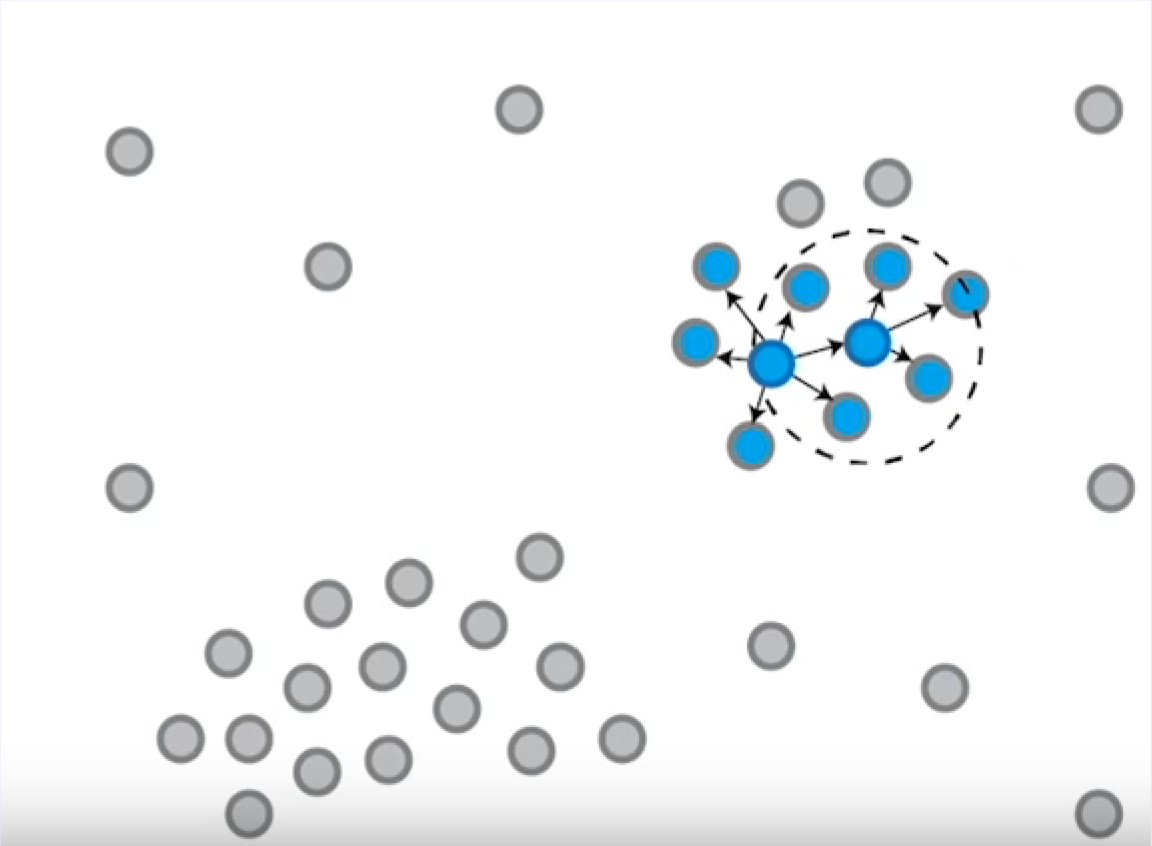
1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within *Eps* of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.



Density-based clustering: dbscan

Algorithm 8.4 DBSCAN algorithm.

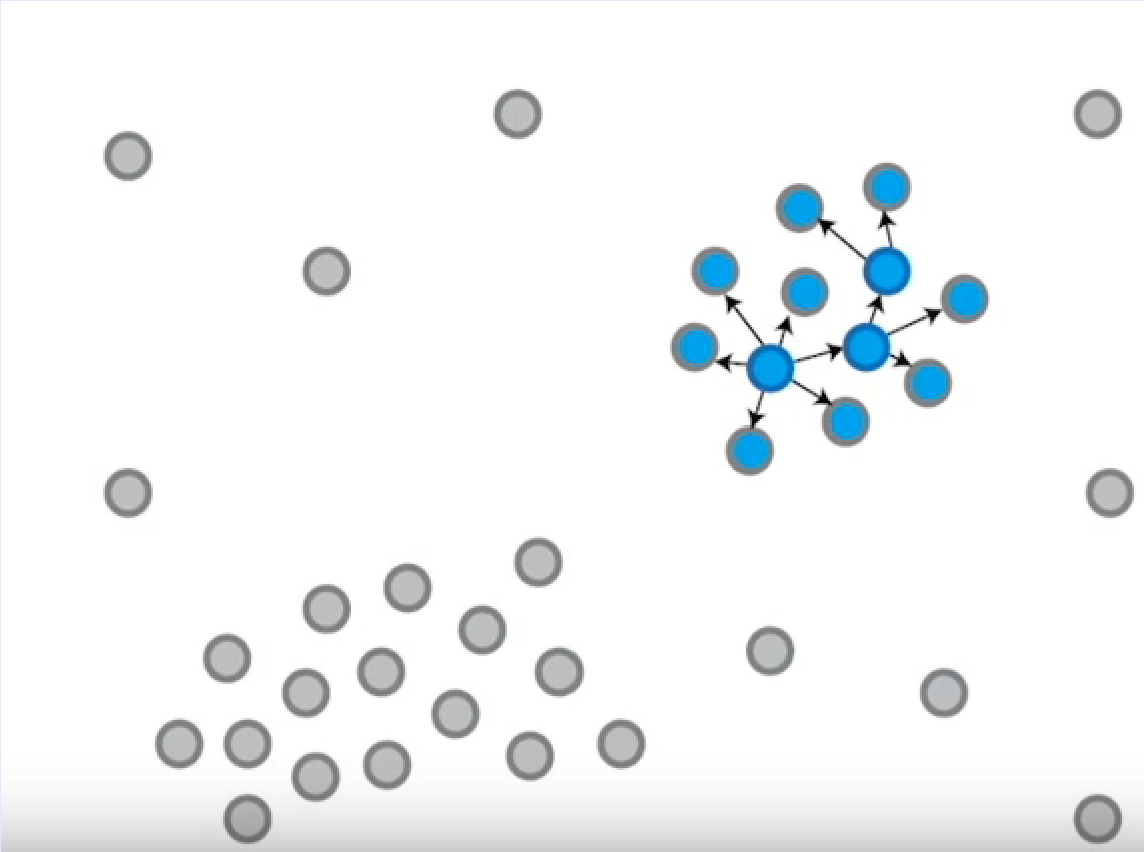
1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within *Eps* of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.



Density-based clustering: dbscan

Algorithm 8.4 DBSCAN algorithm.

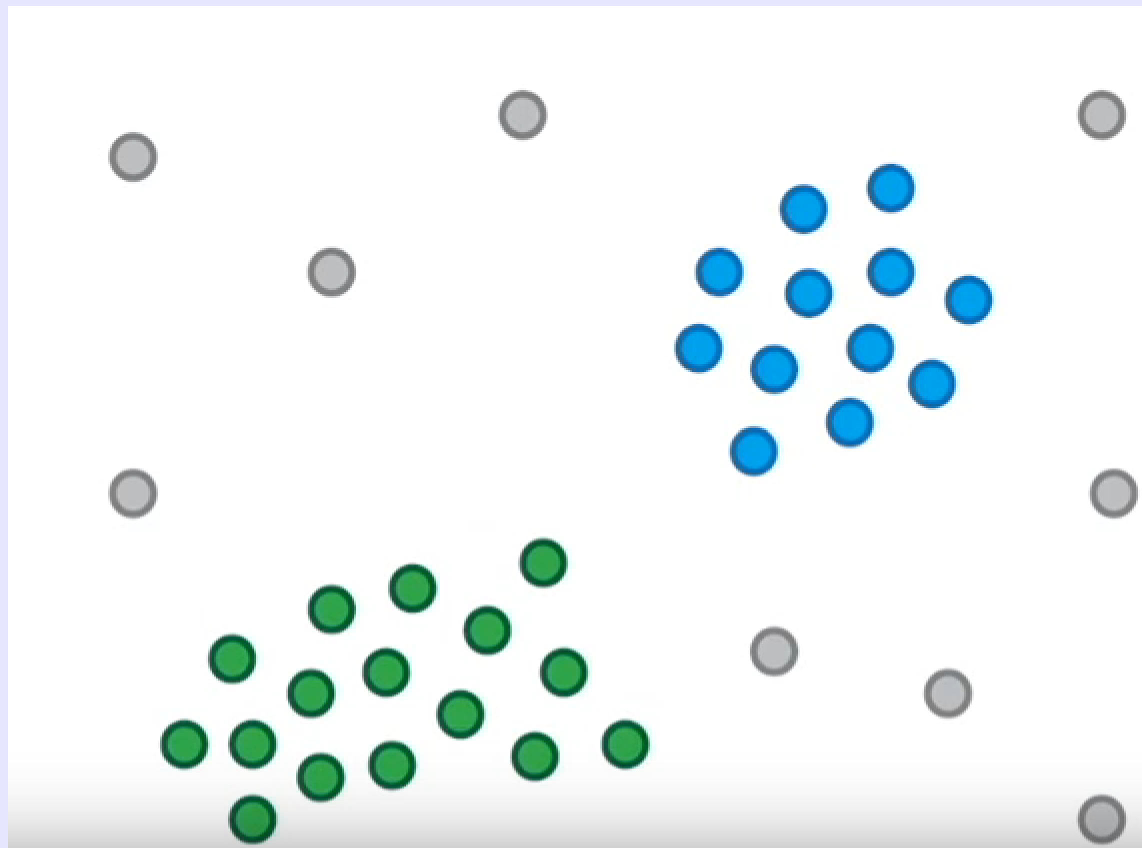
1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within *Eps* of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.



Density-based clustering: dbscan

Algorithm 8.4 DBSCAN algorithm.

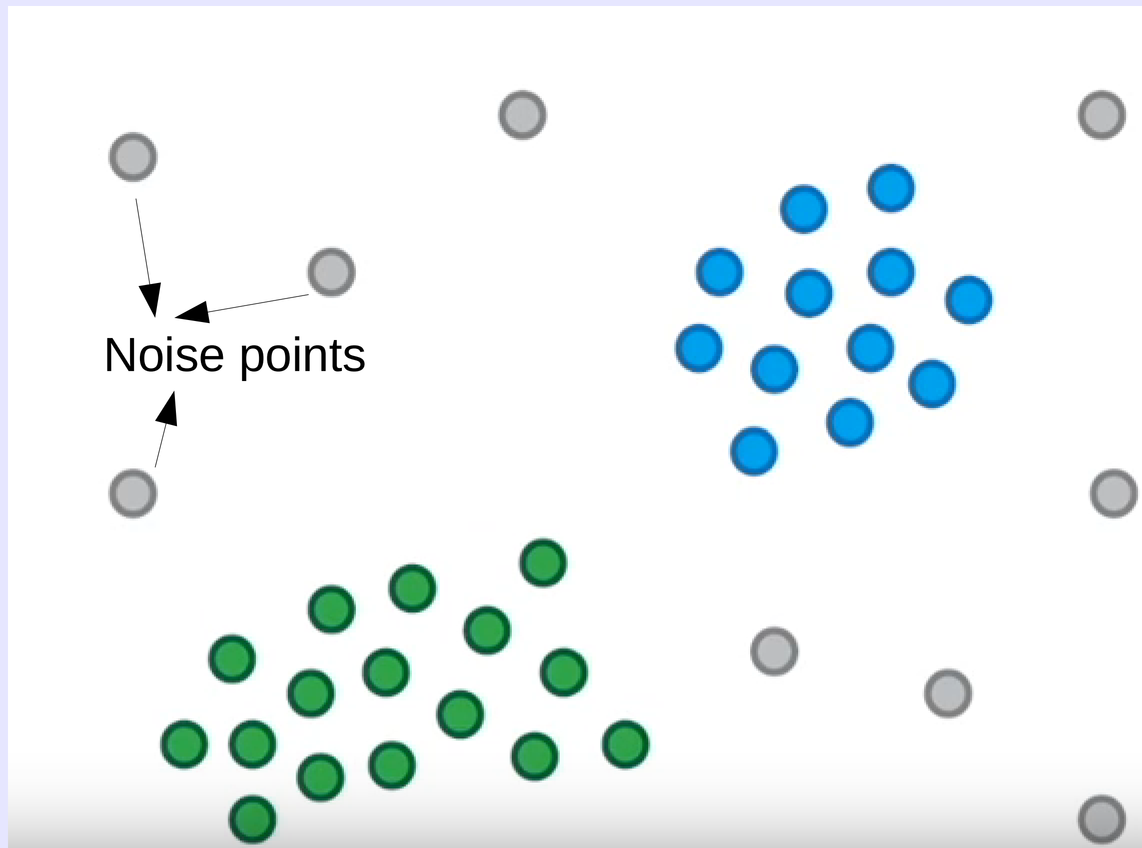
1. Label all points as core, border, or noise points.
2. Eliminate noise points.
3. Put an edge between all core points that are within *Eps* of each other.
4. Make each group of connected core points into a separate cluster.
5. Assign each border point to one of the clusters of its associated core points.



Density-based clustering: dbscan

Algorithm 8.4 DBSCAN algorithm.

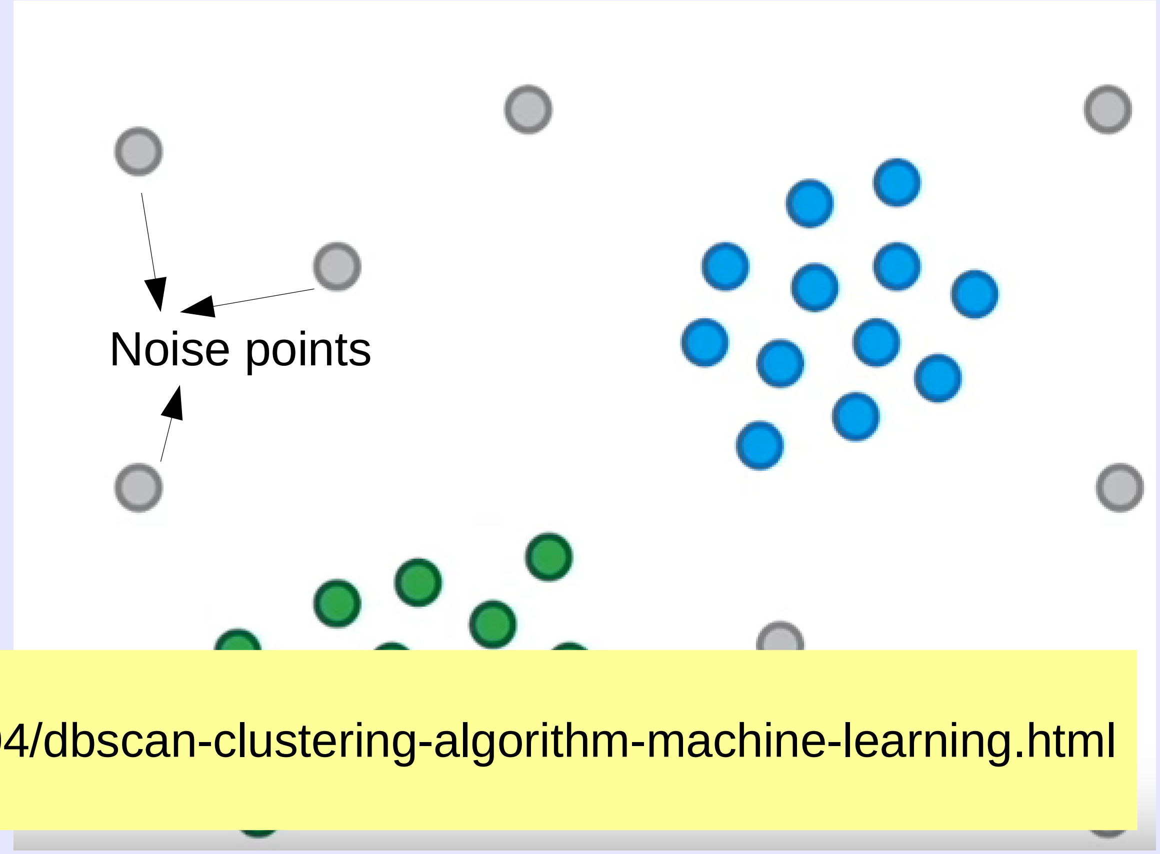
1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within *Eps* of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.



Density-based clustering: dbscan

Algorithm 8.4 DBSCAN algorithm.

1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within *Eps* of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.



Resources:
1. https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html
2. Code: dbscan-clust.r

Density-based clustering: dbscan

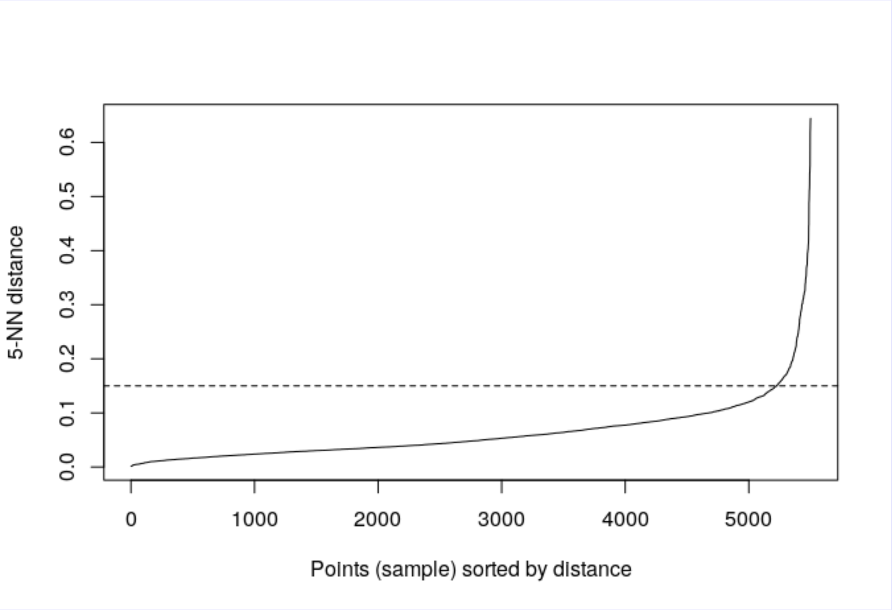
- Complexity:
  - Space: $O(m)$, where m is number of points.
  - Time:
    - *O(m * time to find points in eps-neighbourhood),* in worst case $O(m^2)$.
    - In low-dimension spaces using kd-tree data structure, $O(m \log m)$.

Density-based clustering: Practical issues in dbscan

- How do choose *MinPts* (or define the neighbourhood)?
  - If neighbourhood is too small, then a sparse cluster may be erroneously labeled as noise.
  - If the neighbourhood is too large, then dense clusters may be merged together, and small clusters may be labeled as noise.
  - Original dbscan used *minPts* = 4; suffices for 2 dimensions.
    - For > 2 dimensions: *MinPts* = 2*dimensions (Sander et al. 1998, “Density based clustering in spatial databases: The algorithm GDBSCAN and its applications”, Journal of Data Mining and Knowledge Discovery, 2(2), June 1998).
  - Schubert et al. 2017 suggests large *MinPts* for large and noisy datasets.
  - Schubert et al. 2017: Clusters too large: decrease ε; too much noise: increase ε.
  - Generally domain expertise (or magic) required to choose the right *k*.

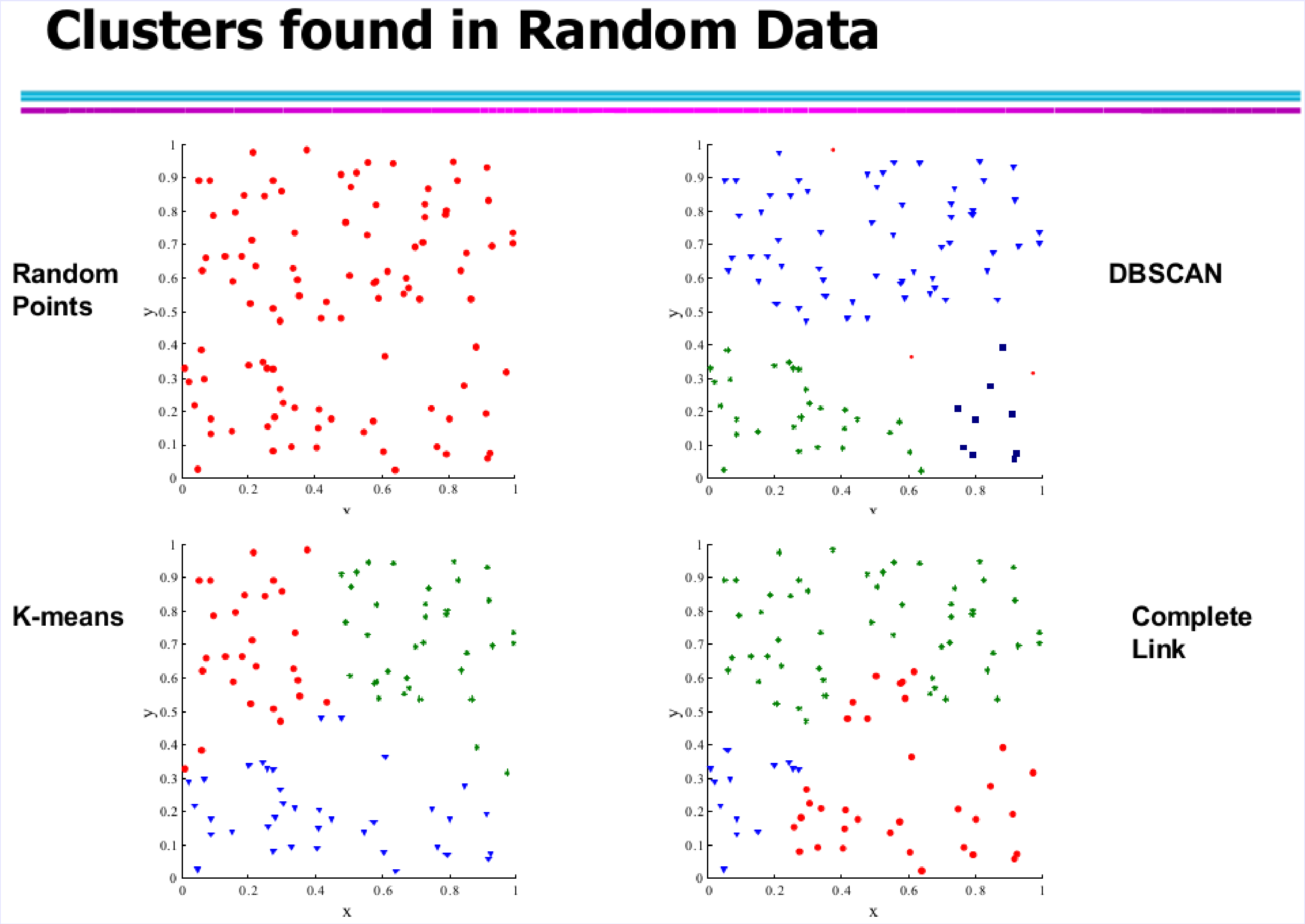
Density-based clustering: Practical issues in dbscan

- How to choose *eps*?
  - Calculate the average of the distances of every point to its *k* nearest neighbours.
    - *k-dist* small for core points and border points in a cluster.
    - *k-dist* large for noise points.
    - Plot the *k-distances* in increasing order, and look for the *knee* to get the value of *eps* at that *k*.



Cluster validation

- Clusters are in the eye of the beholder.



Cluster validation

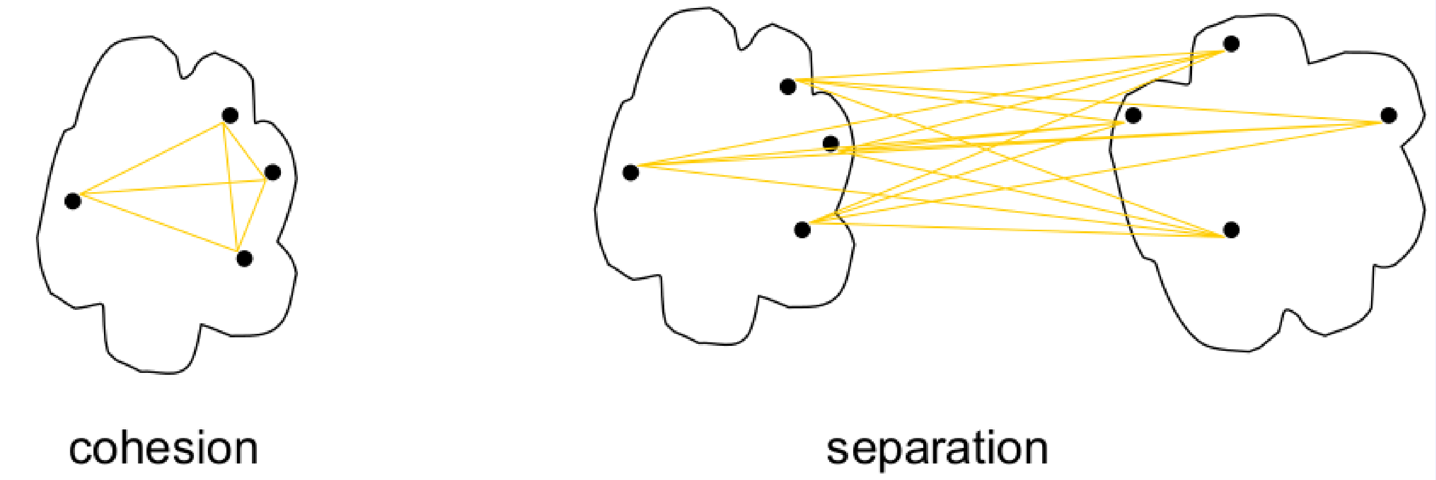
- Validation is the process of evaluating the goodness of clustering algorithm results.
- Why validate?
  - Avoid finding random patterns in data;
  - Compare two clusters (how does k-means compare when k = 2 versus k = 3?);
  - Compare two clustering algorithms (e.g., how does single linkage compare against complete linkage?);
  - ...

Cluster validation

- Types of validations
  - Internal: Used to measure the goodness of a clustering structure without respect to external information.
    - Sum of Squared Error (SSE), for example.
  - External: Match cluster result with external results (class labels).
  - Relative: Used to compare two different clustering algorithms or clusters.

Cluster validation

- Types of validations
  - Internal: Used to measure the goodness of a clustering structure without respect to external information.
    - Two measures: cohesion (how close are objects in the same cluster, measured by within cluster SSE) and separation (how well separated are the clusters).



Cluster validation

- Measuring separation
  - Silhouette value: measures the degree of confidence in the clustering assignment of a particular observation, with well-clustered observations having values near 1 and poorly clustered observations having values near -1.
  - Silhouette width for the i^{th} point, S_i, is defined as:

    $S_i = (b_i - a_i)/\max(b_i, a_i)$

    - a_i is the average distance from the i^{th} point to all other points in the same cluster as the i^{th} point.
    - For the i^{th} point and any cluster not containing that point, calculate the i^{th} point's average distance to all the points in the given cluster. Find the minimum such value with respect to all the clusters; this value is b_i.

Cluster validation

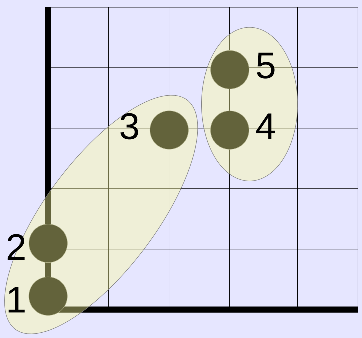
- The silhouette width, S_i, lies in the interval [-1,1] and should be maximized.
  - Large S_i: observations are well clustered.
  - Small S_i : observations lies between two clusters.
  - Negative S_i: observations probably placed in wrong cluster.

Sample code to get Silhouette widths:

```
library(cluster)
library(factoextra)
...
> df <- …
> k <- kmeans(df, centers=2)
> silhouette(k$cluster, dist(df))
     cluster neighbor sil_width
[1,]       1        2 0.7665041
[2,]       1        2 0.7190122
[3,]       2        1 0.6247713
[4,]       2        1 0.7451642
[5,]       2        1 0.7387961
attr(,"Ordered")
[1] FALSE
attr(,"call")
silhouette.default(x = k$cluster, dist = dist(df))
attr(,"class")
[1] "silhouette"
```

Cluster validation

- Silhouette width example:



Data: {(0,0), (0,1), (2,3)} → Cluster 1
{(3,3), (3,4)} → Cluster 2

```
> dist(df, method="manhattan",
+        diag = T, upper = T)
  1 2 3 4 5
1 0 1 5 6 7
2 1 0 4 5 6
3 5 4 0 1 2
4 6 5 1 0 1
5 7 6 2 1 0
```

Silhouette of observation 1: $S_1 = (b_1 – a_1)/\max(b_1, a_1)$

$a_1 = (1+5)/2 = 6/2 = 3$

$b_1 = (6+7)/2 = 13/2 = 6.5$

$S_1 = (6.5 – 3)/6.5 = 3.5/6.5 = 0.538$

Silhouette of observation 2: $S_2 = (b_2 – a_2)/\max(b_2, a_2)$

$a_2 = (1+4)/2 = 5/2 = 2.5$

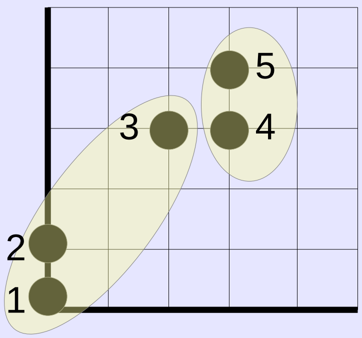
$b_2 = (5+6)/2 = 11/2 = 5.5$

$S_2 = (5.5 – 2.5)/5.5 = 3/5.5 = 0.545$

...

Cluster validation

- Silhouette width example:



Data: {(0,0), (0,1), (2,3)} → Cluster 1
{(3,3), (3,4)} → Cluster 2

```
> dist(df, method="manhattan",
+        diag = T, upper = T)
  1 2 3 4 5
1 0 1 5 6 7
2 1 0 4 5 6
3 5 4 0 1 2
4 6 5 1 0 1
5 7 6 2 1 0
```

Silhouette of observation 1: $S_1 = (b_1 - a_1)/\max(b_1, a_1)$

$a_1 = (1+5)/2 = 6/2 = 3$

$b_1 = (6+7)/2 = 13/2 = 6.5$

$S_1 = (6.5 - 3)/6.5 = 3.5/6.5 = 0.538$

Silhouette of observation 2: $S_2 = (b_2 - a_2)/\max(b_2, a_2)$

$a_2 = (1+4)/2 = 5/2 = 2.5$

$b_2 = (5+6)/2 = 11/2 = 5.5$

$S_2 = (5.5 - 2.5)/5.5 = 3/5.5 = 0.545$

...

Silhouette table

Point	Cluster	Neighbour	Sil. width
1	1	2	0.538
2	1	2	0.545
3	1	2	-0.667
4	2	1	0.750
5	2	1	0.800

Avg. Sil. width Cluster 1: 0.139
Avg. Sil. width Cluster 2: 0.775
Overall avg. Sil. Width: 0.457

Cluster validation

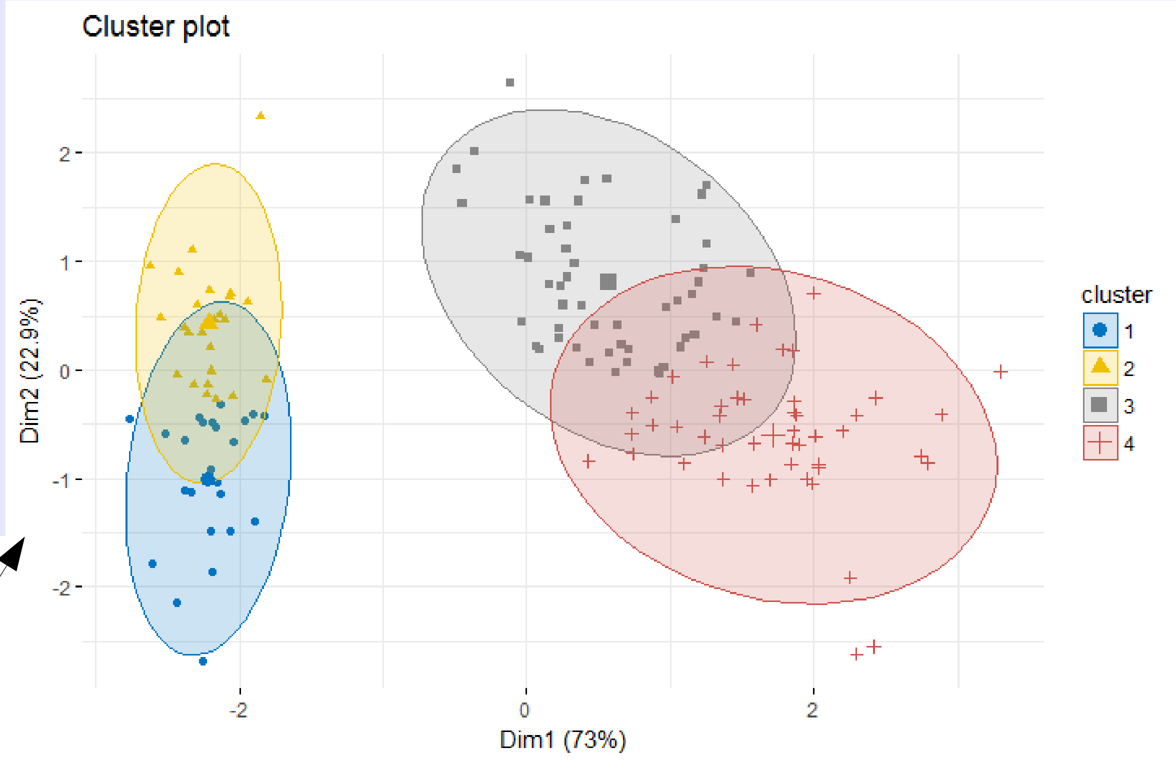
- The Dunn index is the ratio of the smallest distance between observations not in the same cluster (*min.separation*) to the largest intra-cluster distance *(maximum diameter)*

  D = min.separation/max.diameter

- Has a value between 0 and ∞ and should be maximized.
- Code on next page shows how to get the Dunn index.
- **Drawback:** high computational cost as the number of clusters and dimensionality of the data increase.

Cluster validation

- Code to get various cluster metrics:




```
> library(factoextra)
> library(fpc)
> data(iris)
> df <- scale(iris[, 1:4])
> k <- eclust(df, "kmeans", k=4, nstart=25, graph=T)
> fviz_cluster(k, geom = "point", ellipse.type = "norm",
+ palette = "jco", ggtheme = theme_minimal())
> stats <- cluster.stats(dist(df), k$cluster)
> names(stats)
 [1] "n"                  "cluster.number"     "cluster.size"        "min.cluster.size"
 [5] "noisen"             "diameter"           "average.distance"    "median.distance"
 [9] "separation"         "average.toother"    "separation.matrix"   "ave.between.matrix"
[13] "average.between"    "average.within"     "n.between"           "n.within"
[17] "max.diameter"       "min.separation"     "within.cluster.ss"   "clus.avg.silwidths"
[21] "avg.silwidth"       "g2"                 "g3"                  "pearsongamma"
[25] "dunn"               "dunn2"              "entropy"             "wb.ratio"
[29] "ch"                 "cwidegap"           "widestgap"           "sindex"
[33] "corrected.rand"     "vi"
> 
```