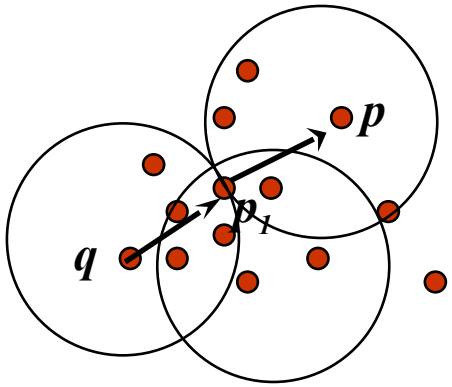
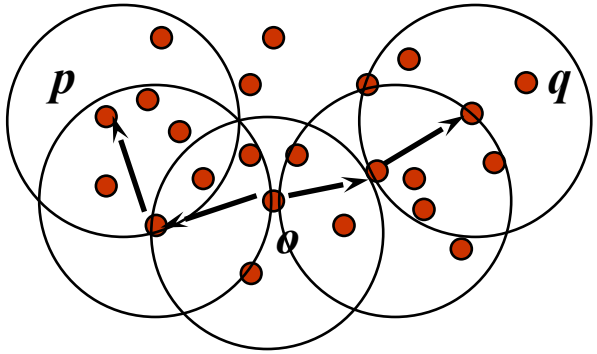
Cluster Analysis II

Density-based Clustering

- Clustering based on density (local cluster criterion), such as density-connected points
- A cluster is a maximal set of ***density-connected*** points
- Basic idea: clusters are dense regions in the data space, separated by regions of lower density
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - **One scan**
  - Need density parameters such as termination criteria





Density-based Approaches

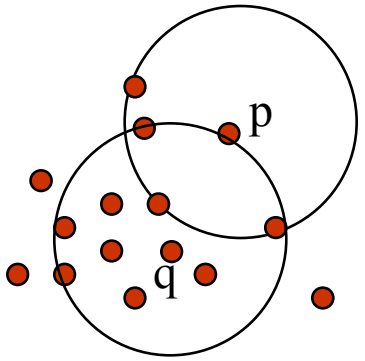
- DBSCAN (KDD'96)
  - The first density based clustering
- OPTICS
  - Density based cluster-ordering
- DENCLUE
  - A general density-based description of cluster and clustering

Density-based Clustering: Basic Concepts

- Two parameters:
  - ***Eps***: Maximum radius of the neighborhood
  - ***MinPts***: Minimum number of points in an Eps-neighborhood (N_{Eps}) of that point

- **Directly density-reachable**: A point *p* is directly density-reachable from a point *q* w.r.t. *Eps*, *MinPts* if
  - *p* belongs to $N_{Eps}(q)$
  - core point condition:

$$|N_{Eps}(q)| \geq MinPts$$

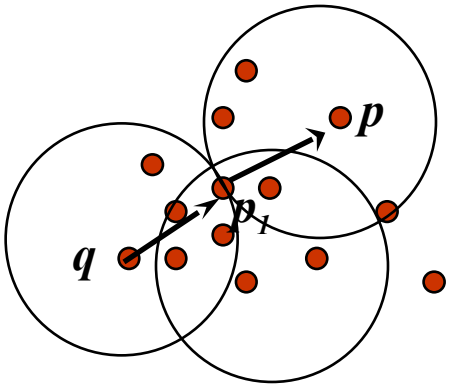


MinPts = 5

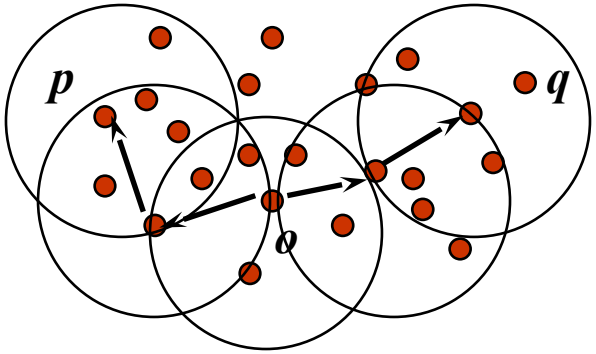
Eps = 1 cm

Density-reachable and Density-connected

- Density-reachable:
  - A point *p* is **density-reachable** from a point *q* w.r.t. *Eps* & *MinPts*, if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i

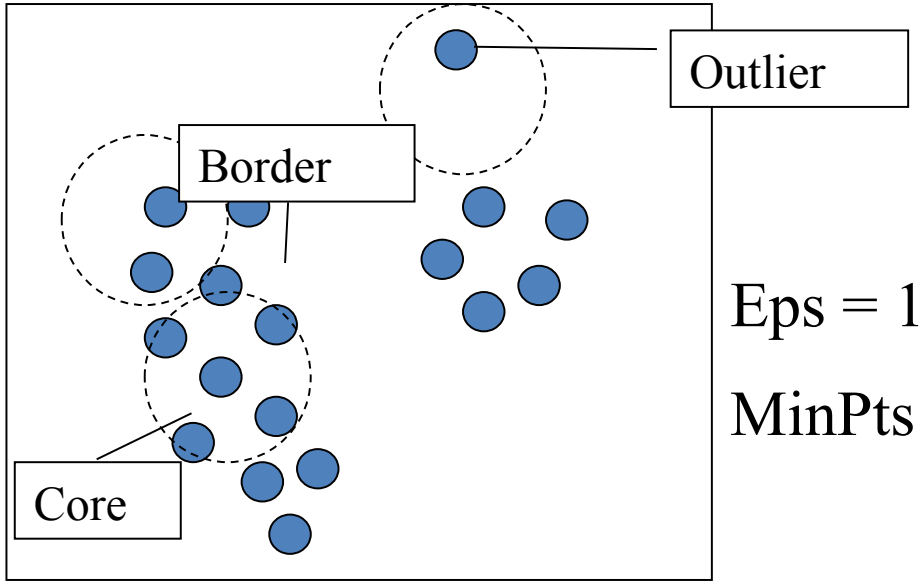


- Density-connected
  - A point *p* is **density-connected** to a point *q* w.r.t. *Eps* & *MinPts*, if there is a point *o* such that both, *p* and *q* are density-reachable from *o* w.r.t. *Eps* and *MinPts*



DBSCAN: Density-based Spatial Clustering of Applications with Noise

- Relies on a *density-based* notion of cluster: A *cluster* is defined as a **maximal** set of ***density-connected*** points
- Discovers clusters of arbitrary shape in spatial databases with noise



- A point is a **core point** if it has more than a specified number of points (*MinPts*) within *Eps*—These are points that are at the interior of a cluster.
- A **border point** has fewer than *MinPts* within Eps, but is in the neighborhood of a core point.
- A **noise/outlier point** is any point that is not a core point nor a border point.

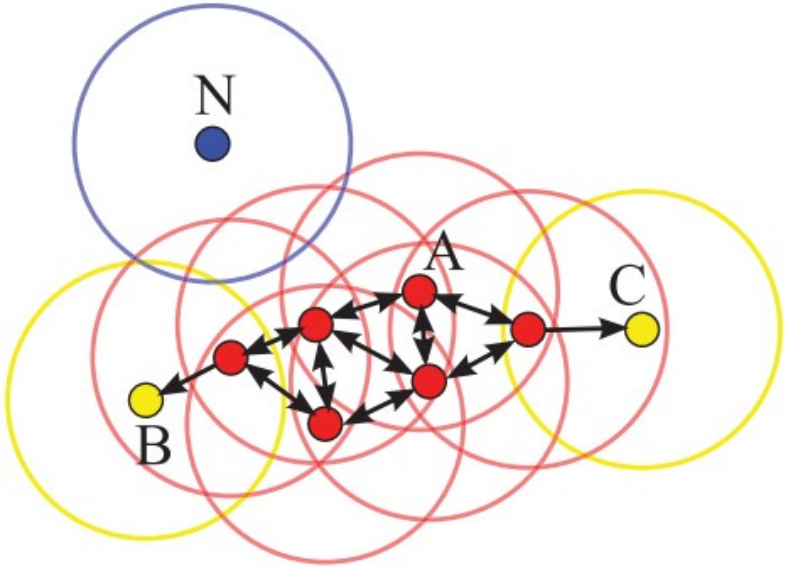
Formal Description of Cluster

Given a dataset *D*, *Eps*, *MinPts*

- a cluster *C* is a subset of objects satisfying 2 criteria:
  - Connected: objects are density-connected
  - Maximal: if *p* is in *C* and *q* is *density-reachable* from *p* (*p* is a core object), then *q* is in *C*

DBSCAN: The Algorithm

1. Compute neighbors of each point and identify **core** points
2. Join neighboring core points into clusters
3. For each non-core point do
   - Add to a neighboring core point if *d* < *eps*
   - Otherwise add to noise




```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

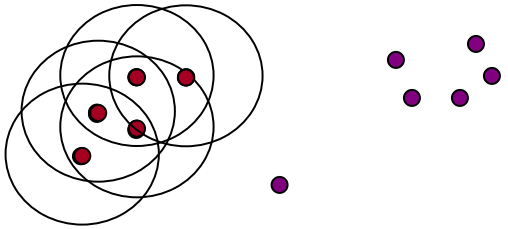
DBSCAN: The Algorithm

1. Arbitrary select a point *p*
2. Retrieve all points **density-reachable** from *p* w.r.t *Eps* and *MinPts*
   - If *p* is a core point, a cluster is formed.
   - If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database.
3. Continue the process until all of the points processed.

✓ Only one scan is needed.

DBSCAN Algorithm: Example

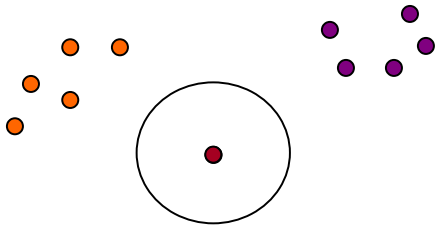
- Parameter
  - ε = 2 cm
  - *MinPts* = 3



```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

DBSCAN Algorithm: Example

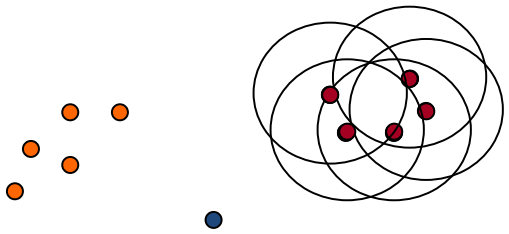
- Parameter
  - ε = 2 cm
  - *MinPts* = 3



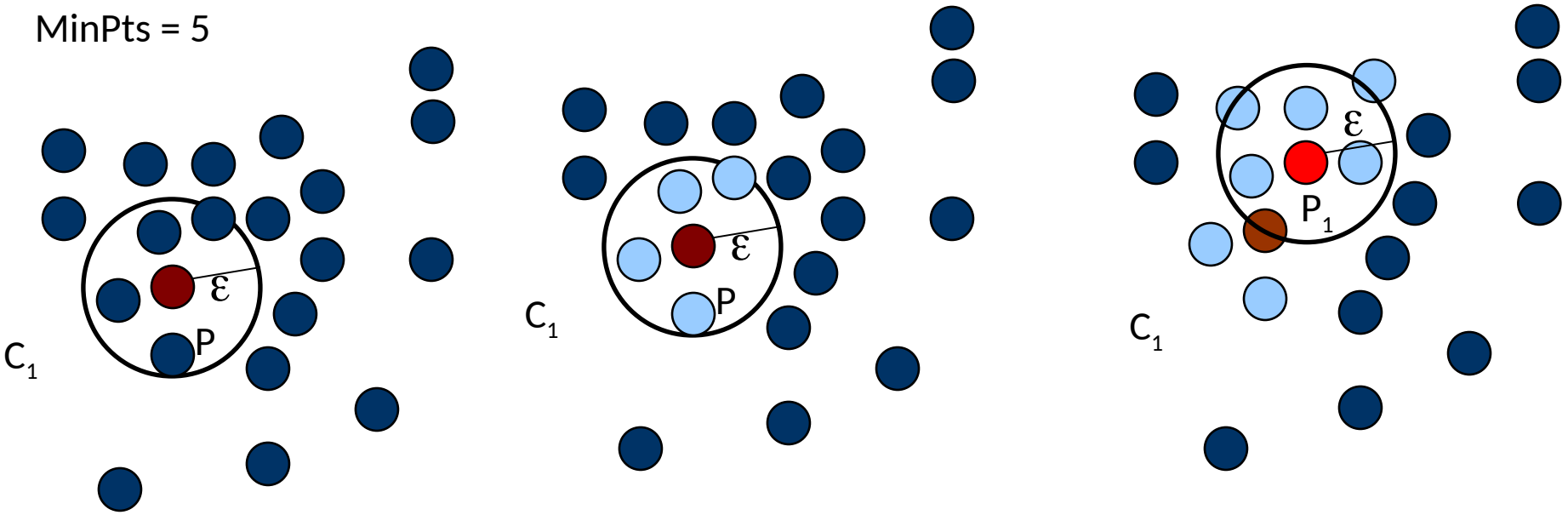
for each $o \in D$ **do**
  if o is not yet classified **then**
    if o is a core-object **then**
      collect all objects density-reachable from o
      and assign them to a new cluster.
    else
      assign o to NOISE

DBSCAN Algorithm: Example

- Parameter
  - ε = 2 cm
  - *MinPts* = 3

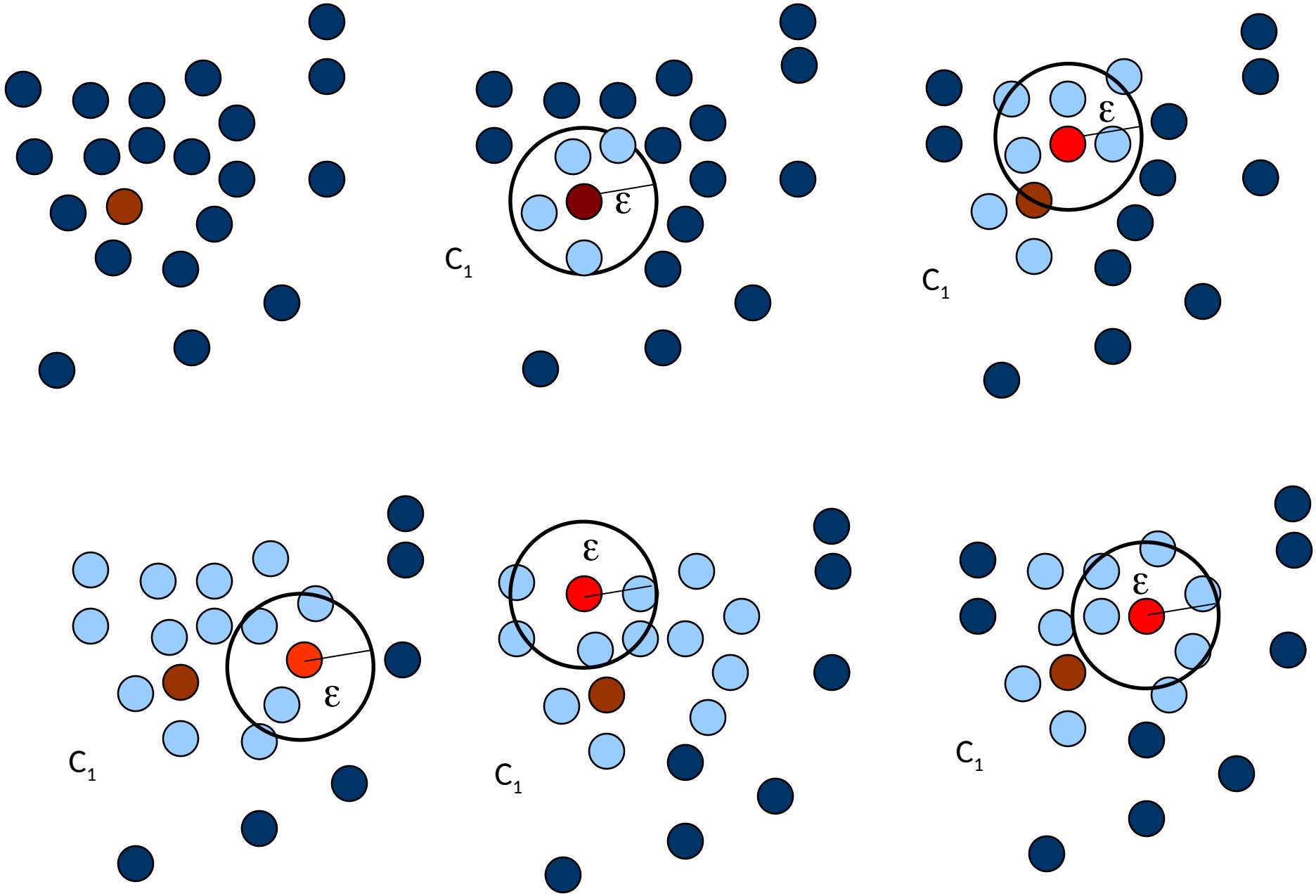


for each $o \in D$ **do**
  if o is not yet classified **then**
    if o is a core-object **then**
      collect all objects density-reachable from o
      and assign them to a new cluster.
    else
      assign o to NOISE



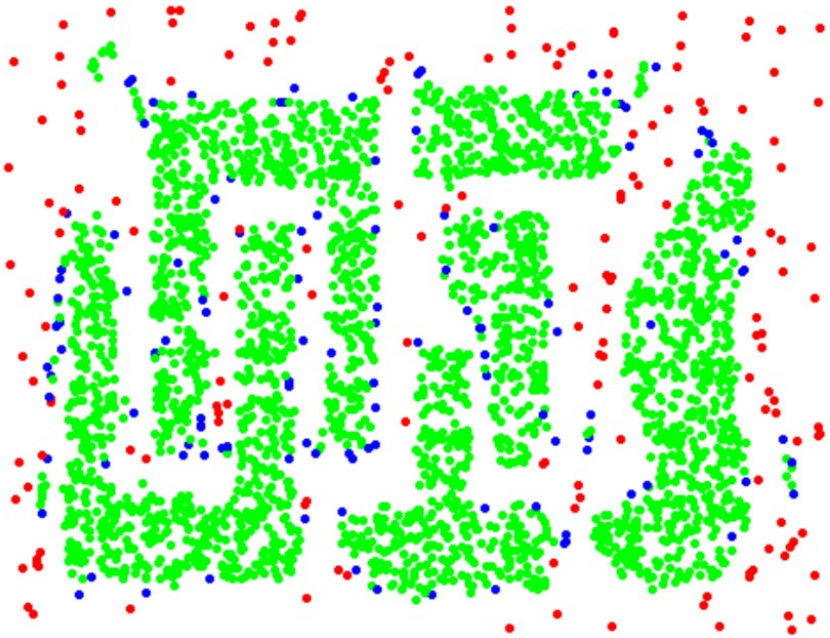
1. Check the ε-neighborhood of p;
2. If p has less than MinPts neighbors then mark p as outlier and continue with the next object
3. Otherwise mark p as processed and put all the neighbors in cluster C

1. Check the unprocessed objects in C
2. If no core object, return C
3. Otherwise, randomly pick up one core object p_1, mark p_1 as processed, and put all unprocessed neighbors of p_1 in cluster C


ε
C1
ε
C1
ε
C1
ε
C1
ε
C1

Example

Original Points



Point types: core, border and outliers

ε = 10, MinPts = 4

DBSCAN: Sensitive to Parameters

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

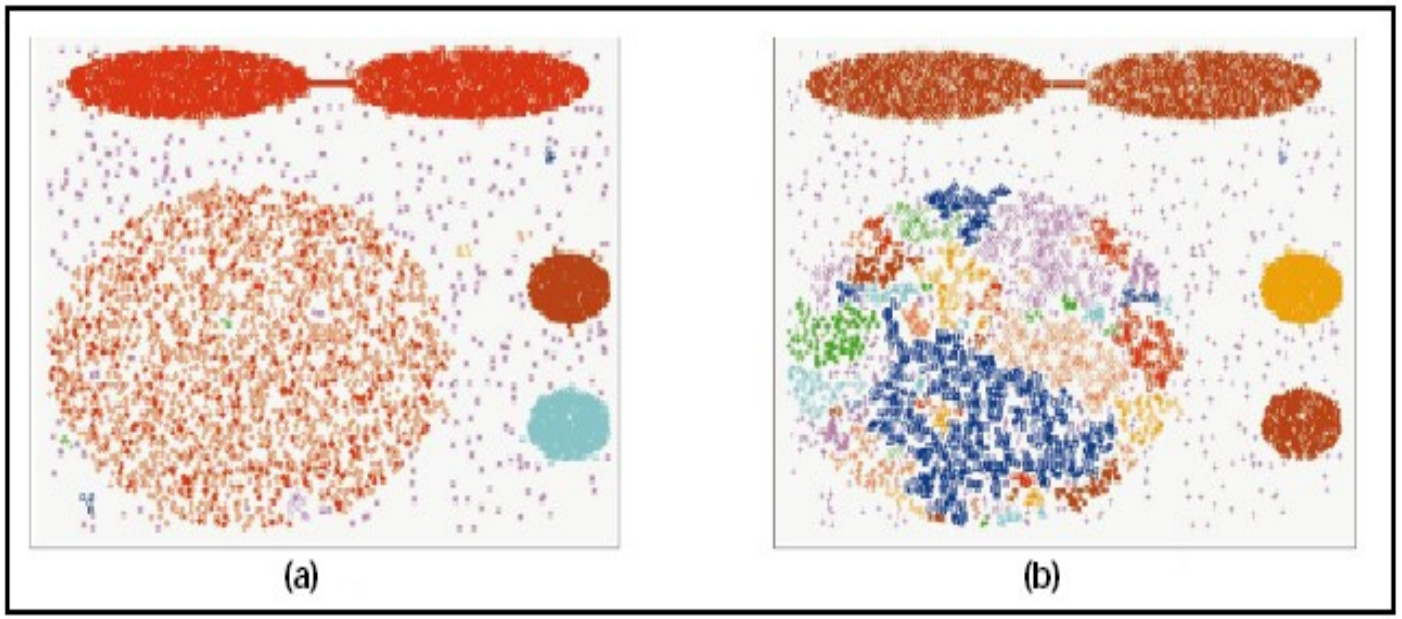
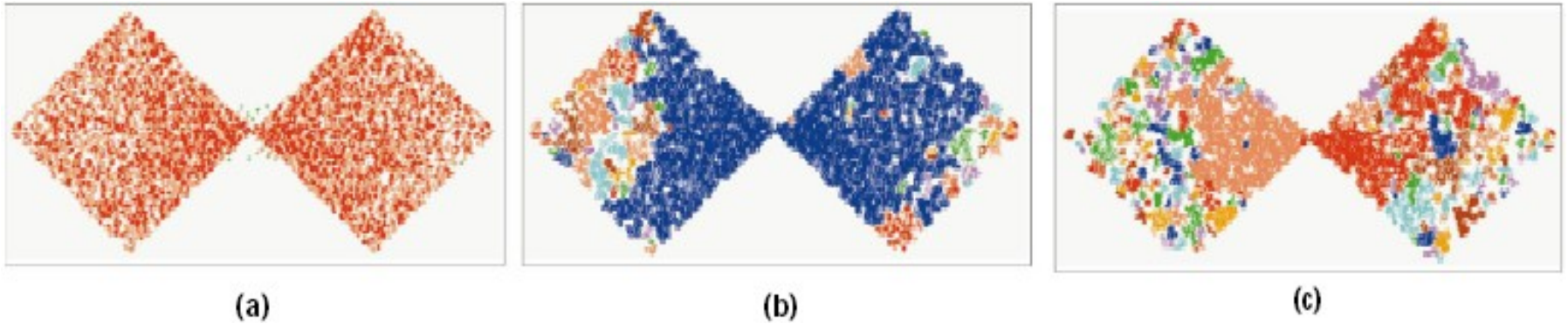


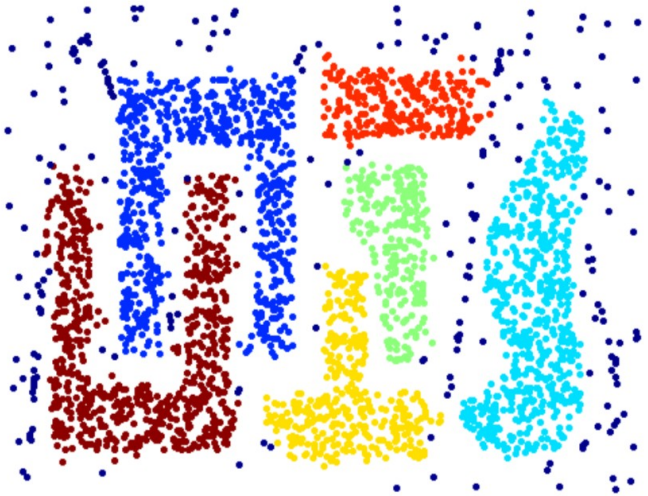
Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



DBSCANE” Pros & Cons

- Pros:
  - Resistant to Noise
  - Can handle clusters of different shapes and sizes

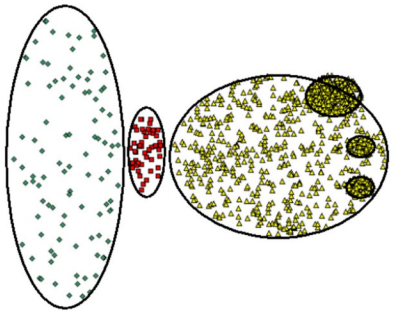
Original Points



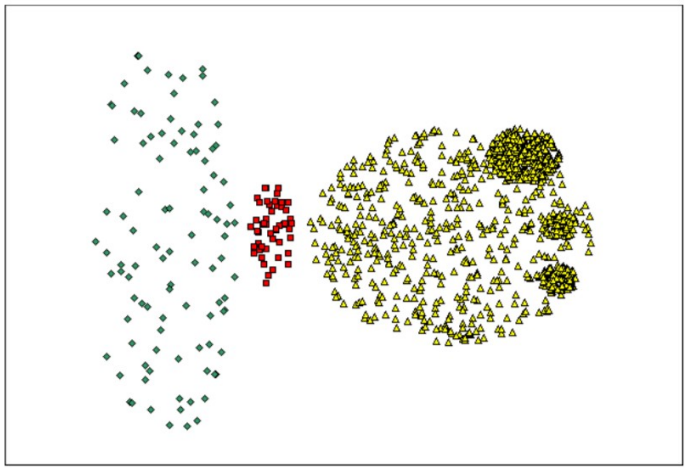
Clusters

DBSCANE” Pros & Cons

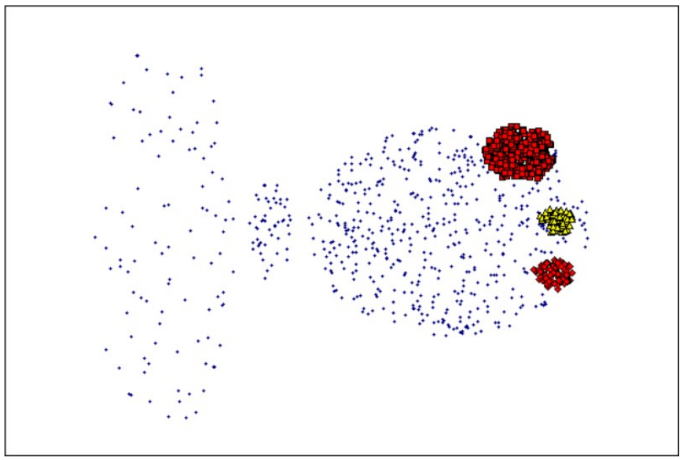
- Cons:
  - Cannot handle varying densities
  - Sensitive to parameters
    - Hard to determine the correct set of parameters



Original Points



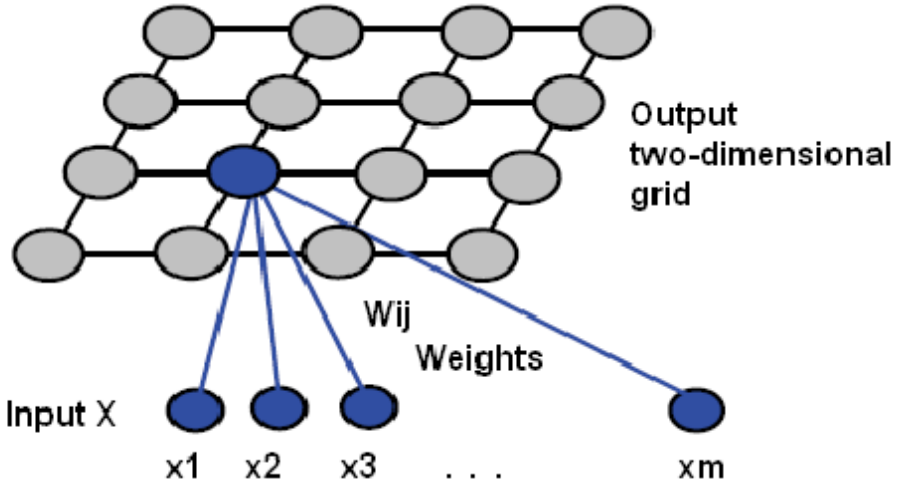
(MinPts=4, Eps=9.92).



(MinPts=4, Eps=9.75)

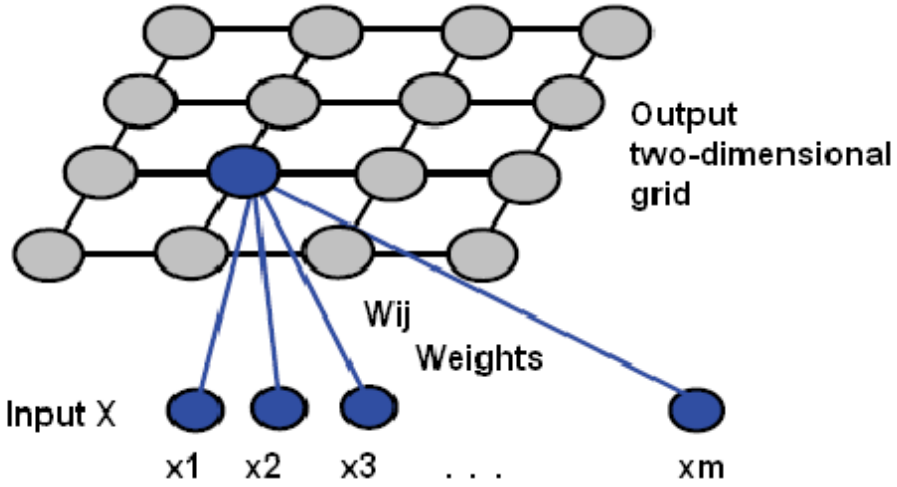
Kohonen Networks for Clustering

- Uses self organizing map (SOM) which is a type of neural network
- SOM converts a complex high-dimensional input signal into a simpler low-dimensional discrete map.
- SOMs structure output nodes into clusters of nodes
- Nodes with closer proximity are more similar



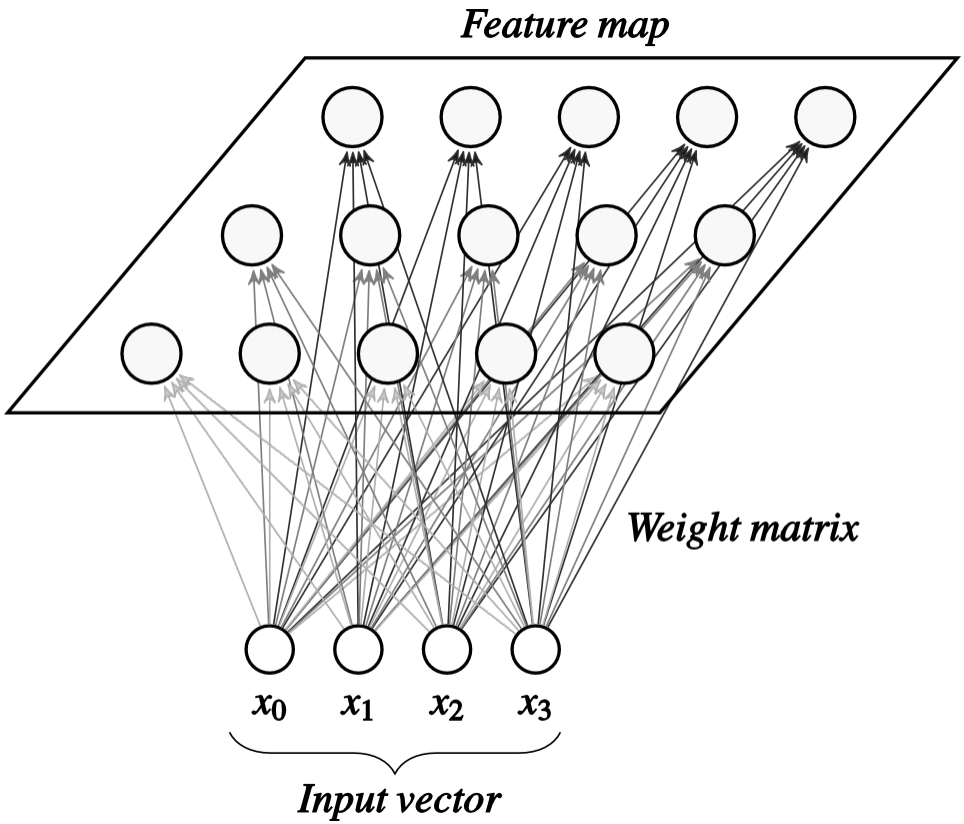
Competitive Learning

- Output nodes compete among themselves to be the winning node (or neuron)
- Neurons become tuned to various input patterns
- SOMs are feed forward and completely connected
- Each connection has a weight associated with it
- Adjusting weights is the key for the learning mechanism



SOM Intuition

- SOMs have no hidden layer
  - Data from the input layer are passed along directly to the output layer
- The output layer is represented in the form of a lattice.
- For a given instance, a particular field value is forwarded from a particular input node to every output node
- A scoring function is then used to designate the winning node



Kohonen Networks

- SOMs exhibit Kohonen learning (Kohonen, 89)
- Given
  - input vector $x_n = x_{n1}, x_{n2}, \ldots x_{n3}$
  - weights vector $w_j = w_{1j}, w_{2j}, \ldots w_{3j}$
- In Kohonen learning, the nodes in the **neighborhood** of the winning node adjust their weights using a linear combination of the input vector and the current weight vector

$$w_{ij,new} = w_{ij.current} + \eta(x_{ni} - w_{ij,current})$$

- The weights are initialized as random values
- Parameters include the learning rate η and the neighborhood size R
- Both η and R should start out large and decrease as the algorithm progresses

Kohonen Network Algorithm

- For each input vector do:
  - ***Competition***: For each output node j, calculate the value $D(w_j, x_n)$ of the scoring function (e.g. Euclidean Distance) and find the winning node that minimizes $D(w_j, x_n)$ over all output nodes
  - ***Cooperation***: Identify all output nodes *j* within the neighborhood of *j* defined by the neighborhood size *R*. For these nodes, do the following for all input fields:
  - ***Adaptation***: Adjust the weights:

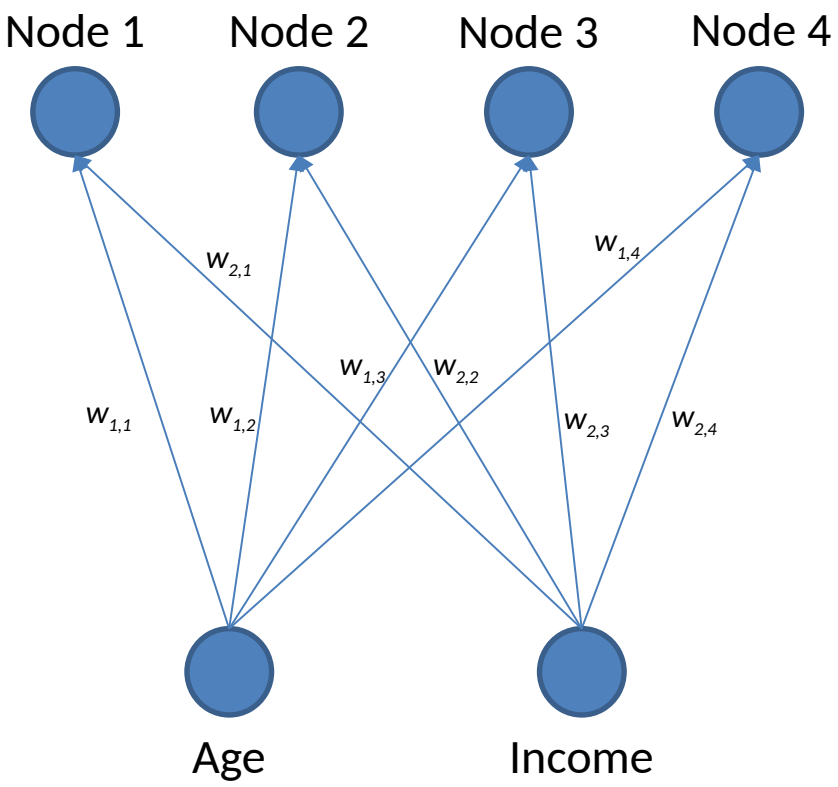
$$w_{ij,new} = w_{ij.current} + \eta(x_{ni} - w_{ij,current})$$

  - Adjust the learning rate and neighborhood size, as needed
  - Stop when the termination criteria are met (e.g. maximum number of iterations)

Kohonen Network Example

$\eta = 0.5$

$R = 0$



Input Data

Age (1)	Income (2)
0.8	0.8
0.8	0.1
0.1	0.8
0.1	0.2

Weights

	Age (1)	Income (2)
Node1	0.9	0.8
Node 2	0.9	0.2
Node 3	0.1	0.8
Node 4	0.1	0.2

Competition:

$x_1 = (0.8, 0.8)$

$D(w_1, x_1) = 0.1$

$D(w_2, x_1) = 0.61$

$D(w_3, x_1) = 0.70$

$D(w_4, x_1) = 0.92$

Adaptation:

$w_{ij,new} = w_{ij.current} + \eta(x_{ni} - w_{ij,current})$

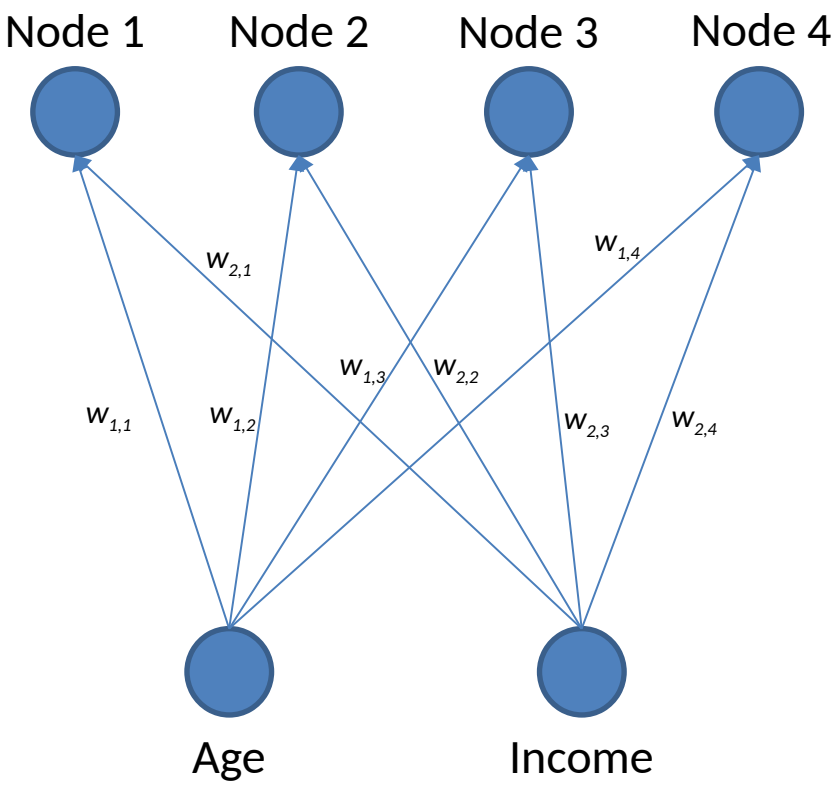
For age: $0.9 + 0.5(0.8-0.9) = 0.85$

For Income: $0.8 + 0.5(0.8-0.8) = 0.8$

Kohonen Network Example

$\eta = 0.5$

$R = 0$



Input Data

Age (1)	Income (2)
0.8	0.8
0.8	0.1
0.2	0.9
0.1	0.1

Weights

	Age (1)	Income (2)
Node1	0.85	0.8
Node 2	0.9	0.2
Node 3	0.1	0.8
Node 4	0.1	0.2

Competition:

$x_2 = (0.8, 0.1)$

$D(w_1, x_2) = 0.71$

$D(w_2, x_2) = 0.14$

$D(w_3, x_2) = 0.99$

$D(w_4, x_2) = 0.78$

Adaptation:

$w_{ij,new} = w_{ij.current} + \eta(x_{ni} - w_{ij,current})$

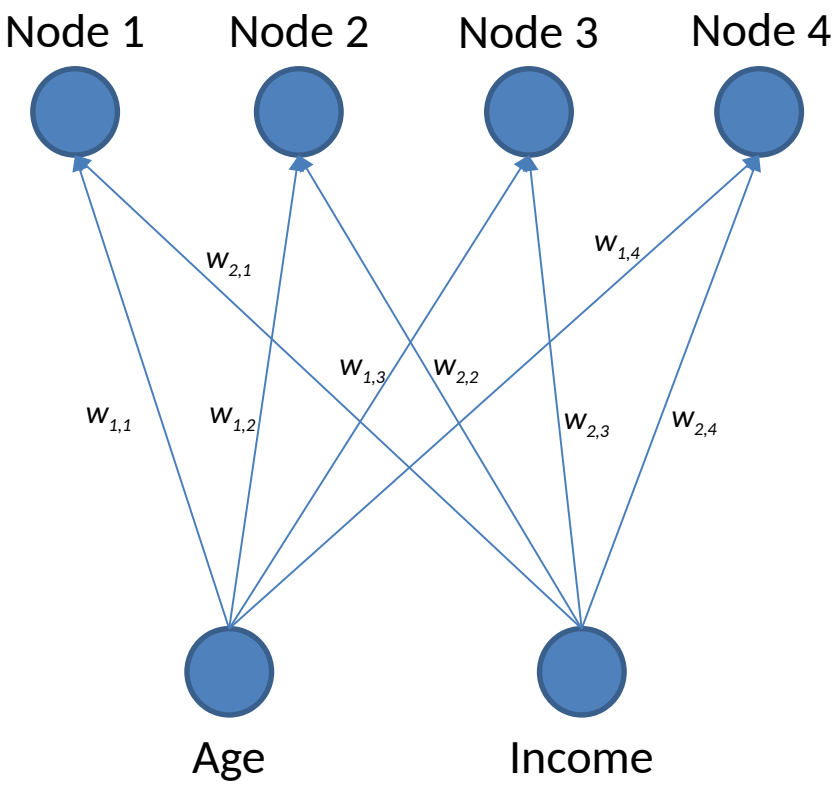
For age: 0.9 + 0.5(0.8-0.9) = 0.85

For Income: 0.2 + 0.5(0.1-0.2) = 0.15

Kohonen Network Example

$\eta = 0.5$

$R = 0$



Input Data

Age (1)	Income (2)
0.8	0.8
0.8	0.1
0.2	0.9
0.1	0.1

Weights

	Age (1)	Income (2)
Node1	0.85	0.8
Node 2	0.85	0.15
Node 3	0.1	0.8
Node 4	0.1	0.2

Competition:

$x_3 = (0.2, 0.9)$

$D(w_1, x_3) = 0.66$

$D(w_2, x_3) = 0.99$

$D(w_3, x_3) = 0.14$

$D(w_4, x_3) = 0.71$

Adaptation:

$w_{ij,new} = w_{ij.current} + \eta(x_{ni} - w_{ij,current})$

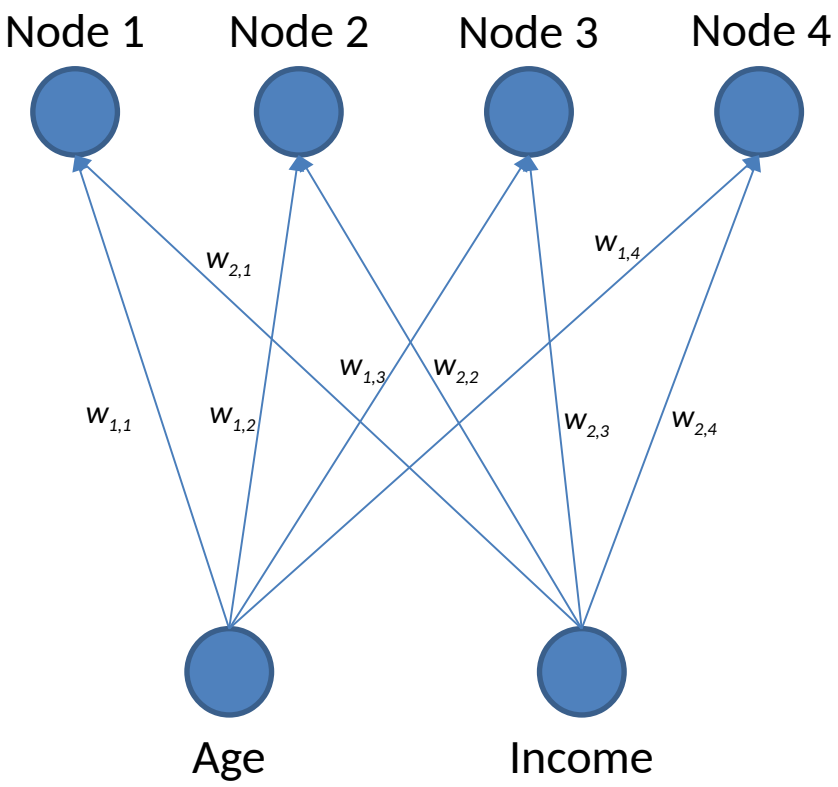
For age: 0.1 + 0.5(0.2-0.1) = 0.15

For Income: 0.8 + 0.5(0.9-0.2) = 0.85

Kohonen Network Example

$\eta = 0.5$

$R = 0$



Input Data

Age (1)	Income (2)
0.8	0.8
0.8	0.1
0.2	0.9
0.1	0.1

Weights

	Age (1)	Income (2)
Node1	0.85	0.8
Node 2	0.85	0.15
Node 3	0.15	0.85
Node 4	0.1	0.2

Competition:

$x_4 = (0.1, 0.1)$

$D(w_1, x_4) = 1.03$

$D(w_2, x_4) = 0.75$

$D(w_3, x_4) = 0.75$

$D(w_4, x_4) = 0.1$

Adaptation:

$w_{ij,new} = w_{ij.current} + \eta(x_{ni} - w_{ij,current})$

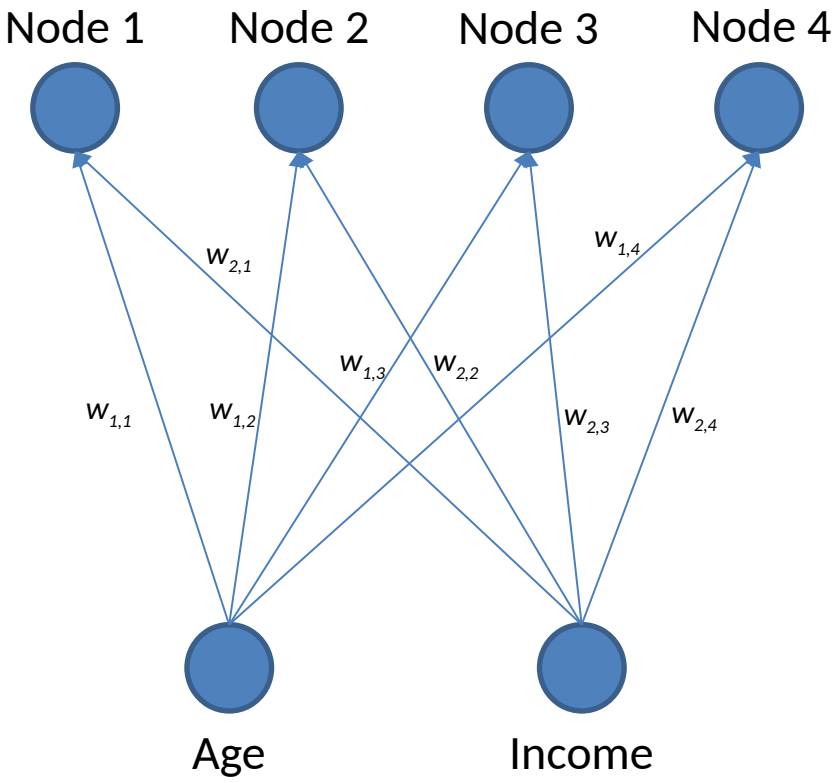
For age: $0.1 + 0.5(0.1-0.1) = 0.1$

For Income: $0.8 + 0.5(0.1-0.2) = 0.15$

Kohonen Network Example

$\eta = 0.5$

$R = 0$



Input Data

Age (1)	Income (2)
0.8	0.8
0.8	0.1
0.2	0.9
0.1	0.1

Weights

	Age (1)	Income (2)
Node1	0.85	0.8
Node 2	0.85	0.15
Node 3	0.15	0.85
Node 4	0.1	0.15

	Associated With	Description
Cluster 1	Node1	Older person with high income
Cluster 2	Node 2	Older person with low income
Cluster 3	Node 3	Younger person with high income
Cluster 4	Node 4	Younger person with low income

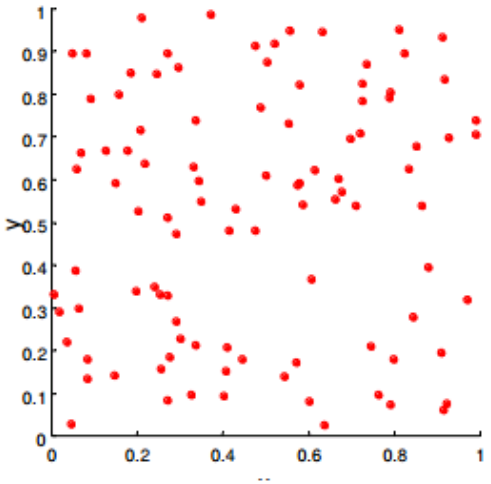
Evaluation of Clustering

- Assessing Cluster Tendency
  - Assess whether a non-random structure exists in the data
- Measuring Cluster Validity
  - How good are the resulting clusters?
- Determining the number of clusters
  - Estimate this number before applying clustering

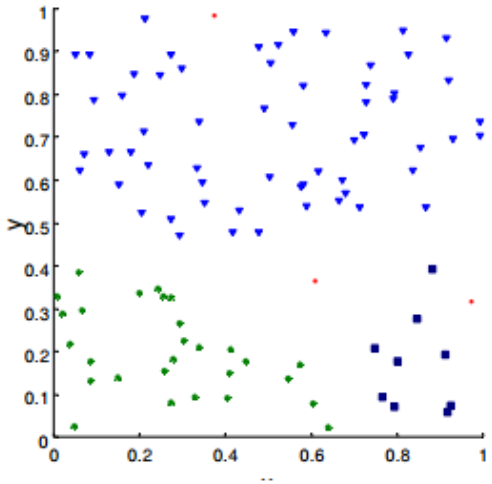
Assessing Clustering Tendency

- What happens when we cluster a completely random dataset?

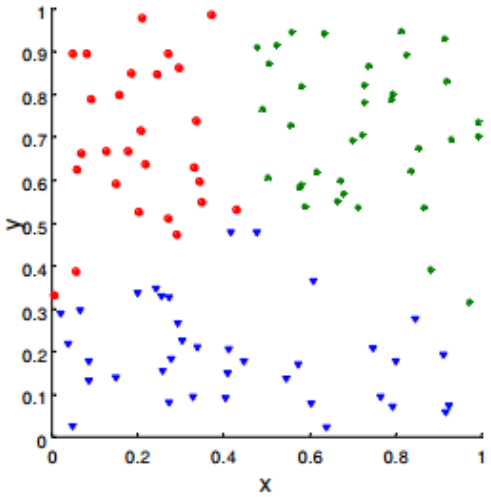
Random Points



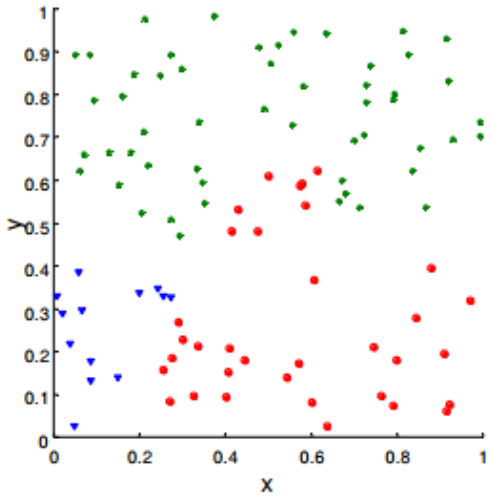
DBSCAN



K-means



Complete Link



Hopkins Statistic

- Given a dataset D regarded as a sample of a random variable o, determine how far away o is from being uniformly distributed in the data space
- Sample *n* points, p_1, …, p_n, uniformly from D. For each p_i, find its nearest neighbor in D: $x_i = min\{dist\ (p_i, v)\}$ where *v* in D
- Sample *n* points, q_1, …, q_n, uniformly from D. For each q_i, find its nearest neighbor in D – $\{q_i\}$: $y_i = min\{dist\ (q_i, v)\}$ where *v* in D and v ≠ q_i
- Calculate the Hopkins Statistic: $$H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i}$$

- If D is uniformly distributed, ∑ x_i and ∑ y_i will be close to each other and H is close to 0.5.
- H > 0.75 indicates a clustering tendency at the 90% confidence level

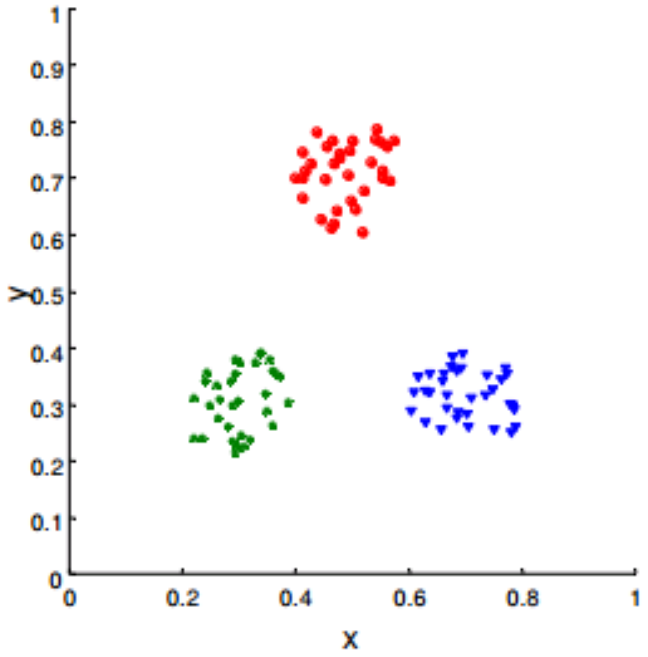
Measuring Cluster Quality

- Evaluate the goodness of a clustering by considering the following criteria:
  - how well the clusters are separated (distance between cluster centers)
  - how compact the clusters (distance between points within clusters)

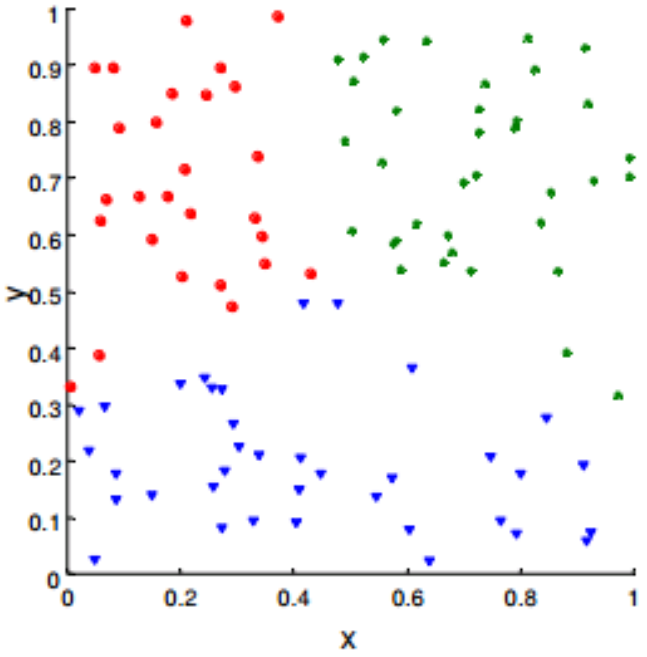
Using Correlation

- Two matrices:
  - Proximity Matrix
  - Incidence Matrix
    - One row and one column for each data point
    - An entry is 1 if the associated pair of points belong to the same cluster
    - An entry is 0 if the associated pair of points belongs to different clusters
- Compute the correlation between the two matrices

Measuring Cluster Validity Via Correlation



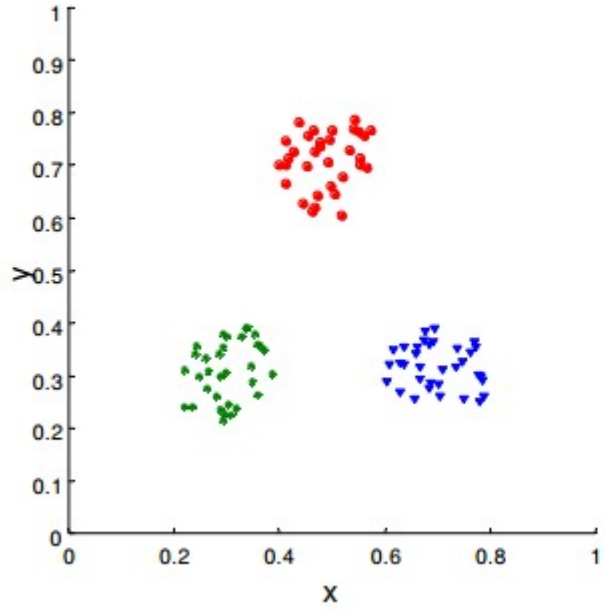
Corr = -0.9235

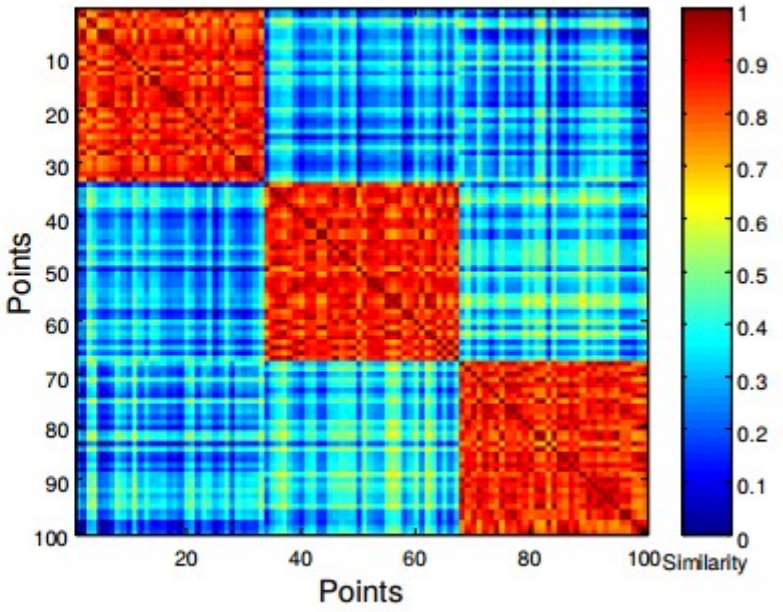


Corr = -0.5810

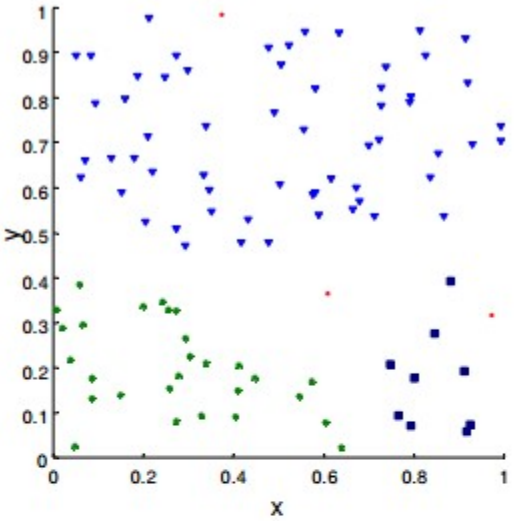
Use Similarity Matrix

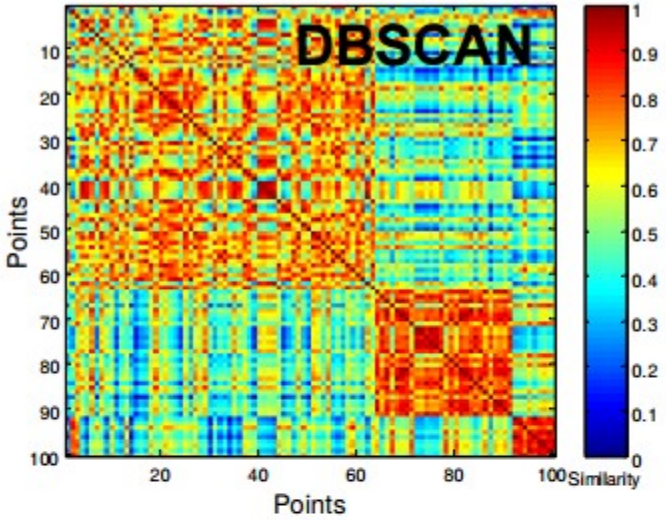
- Order the similarity matrix with respect to cluster labels (visual inspection)

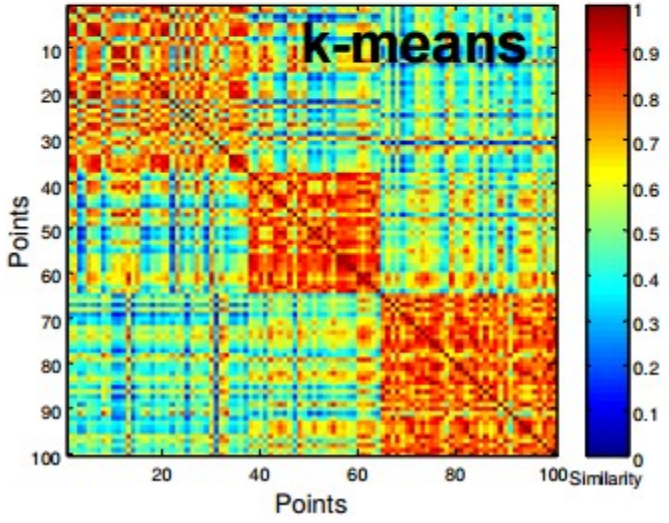


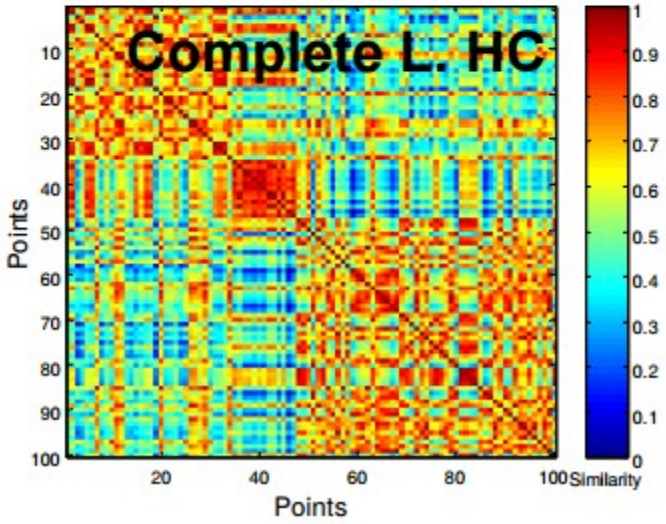


Similarity Matrix for Random Data









Sum of Squares (SSE)

- SSE is good for comparing two clusters (average SSE)

$$SSE = \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{m_i}\|^2$$

- Can also be used to estimate the number of clusters

Cohesion and Separation

- Cluster Cohesion: Measures how closely related objects are in a cluster:
  - Within cluster sum of squares:

$$WSS = \sum_i \sum_{x \in C_i} \|x - m_i\|^2$$

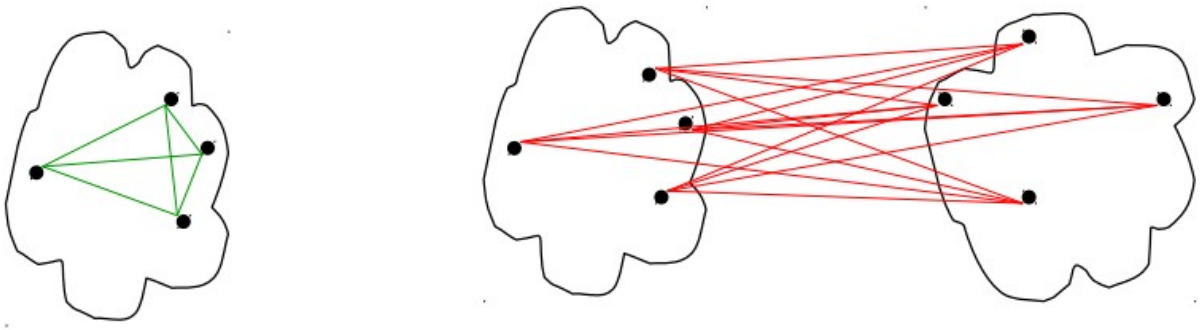
- Cluster Separation: Measure how distinct or well separated a cluster is from other clusters
  - Between cluster sum of squares

$$BSS = \sum_i |C_i| \|m - m_i\|^2$$

- Total sum of squares $TSS = WSS + BSS$

Proximity Graph for Cohesion and Separation

- A proximity graph-based approach can also be used for cohesion and separation.
  - Cohesion is the sum of the weights of all links within a cluster
  - Separation is the sum of the weights between nodes in the cluster and nodes outside the cluster



cohesion

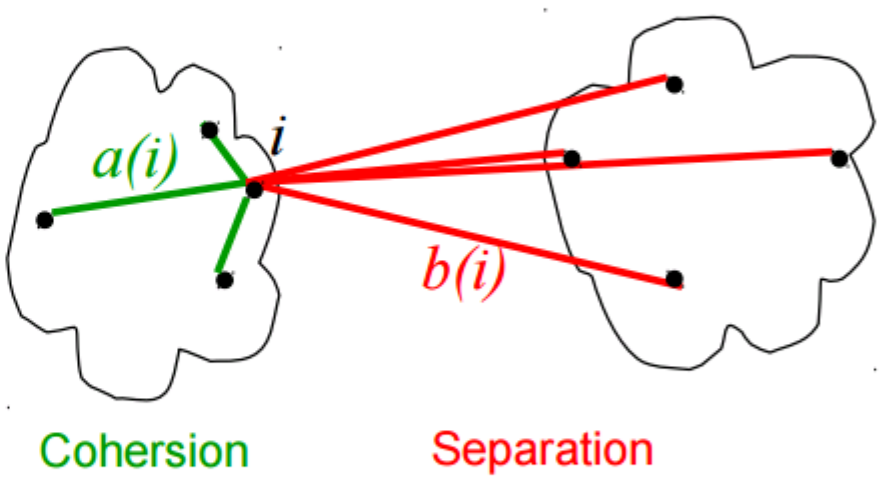
separation

Silhouette Coefficient

- Silhouette Coefficient combine ideas of both cohesion and separation, but for individual points. For an individual point i:
  - Calculate $a(i)$ = average dissimilarity of i to all other points in its cluster
  - Calculate $b(i)$ = lowest average dissimilarity of i to any other)

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

$$-1 \leq s(i) \leq 1$$



  - The closer to 1 the better.

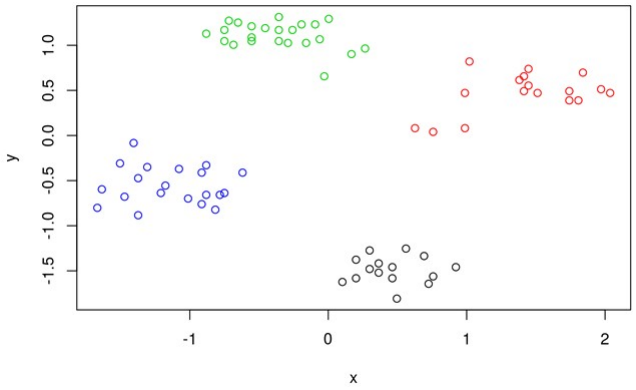
- Can calculate the Average Silhouette width for a cluster or a clustering

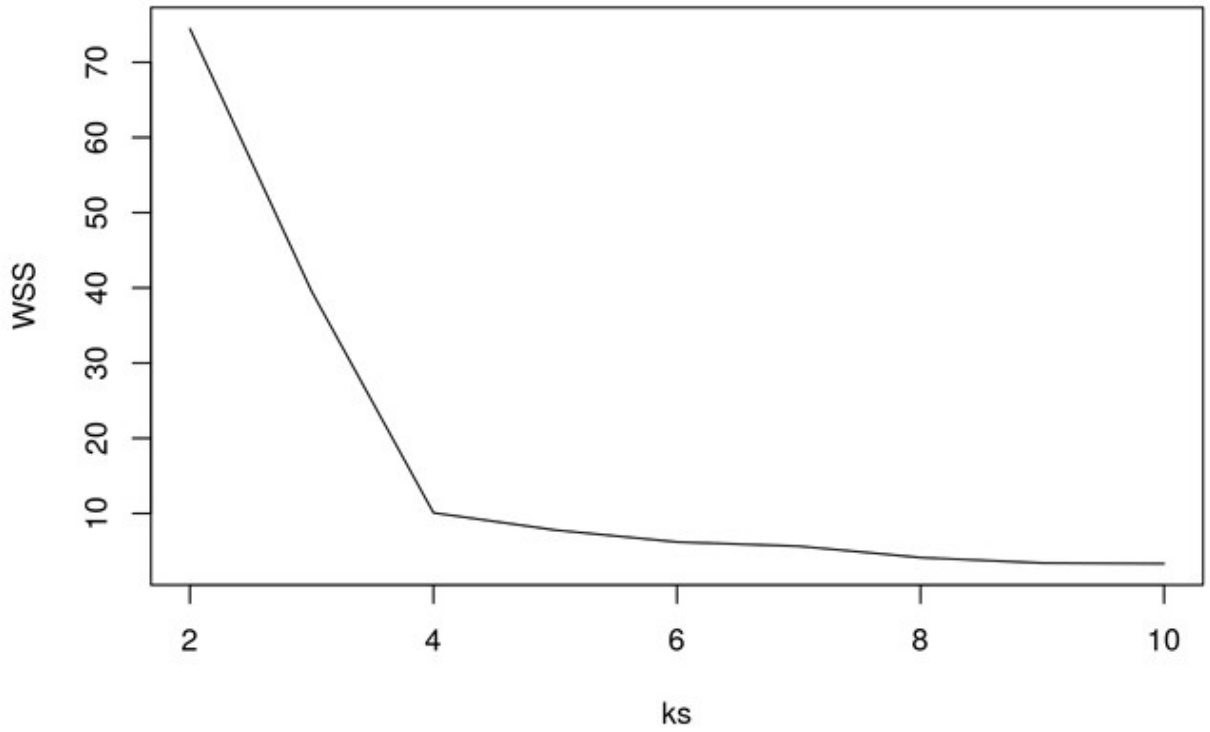
Silhouette Plot

Determine the Number of Clusters

- Empirical method
  - # of clusters ≈√n/2 for a dataset of n points
- Elbow method
  - Use the turning point in the curve of sum of within cluster variance with respect to the # of clusters
- Cross validation method
  - Divide a given data set into *m* parts
  - Use *m* – 1 parts to obtain a clustering model
  - Use the remaining part to test the quality of the clustering
    - E.g., For each point in the test set, find the closest centroid, and use the sum of squared distance between all points in the test set and the closest centroids to measure how well the model fits the test set
  - For any m > 0, repeat it *m* times, compare the overall quality measure with respect to different *m's*, and find # of clusters that fits the data the best

Elbow Method Using SSE





Elbow Method using Average Sillouette