Data Mining
Cluster Analysis: Advanced Concepts and Algorithms

Lecture Notes for Chapter 8

Introduction to Data Mining, 2nd Edition

by

Tan, Steinbach, Karpatne, Kumar

Outline

- Prototype-based
  - Fuzzy c-means
  - Mixture Model Clustering
  - Self-Organizing Maps
- Density-based
  - Grid-based clustering
  - Subspace clustering
- Graph-based
  - Chameleon
  - Jarvis-Patrick
  - Shared Nearest Neighbor (SNN)
- Characteristics of Clustering Algorithms

Hard (Crisp) vs Soft (Fuzzy) Clustering

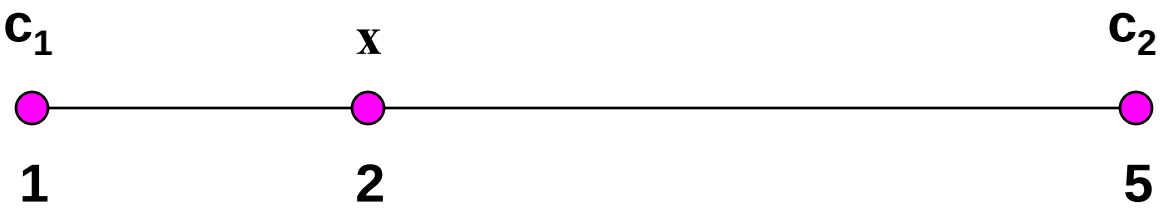
- Hard (Crisp) vs. Soft (Fuzzy) clustering
  - For soft clustering allow point to belong to more than one cluster
  - For K-means, generalize objective function

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{m} w_{ij}\, dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2 \qquad \sum_{j=1}^{k} w_{ij} = 1$$

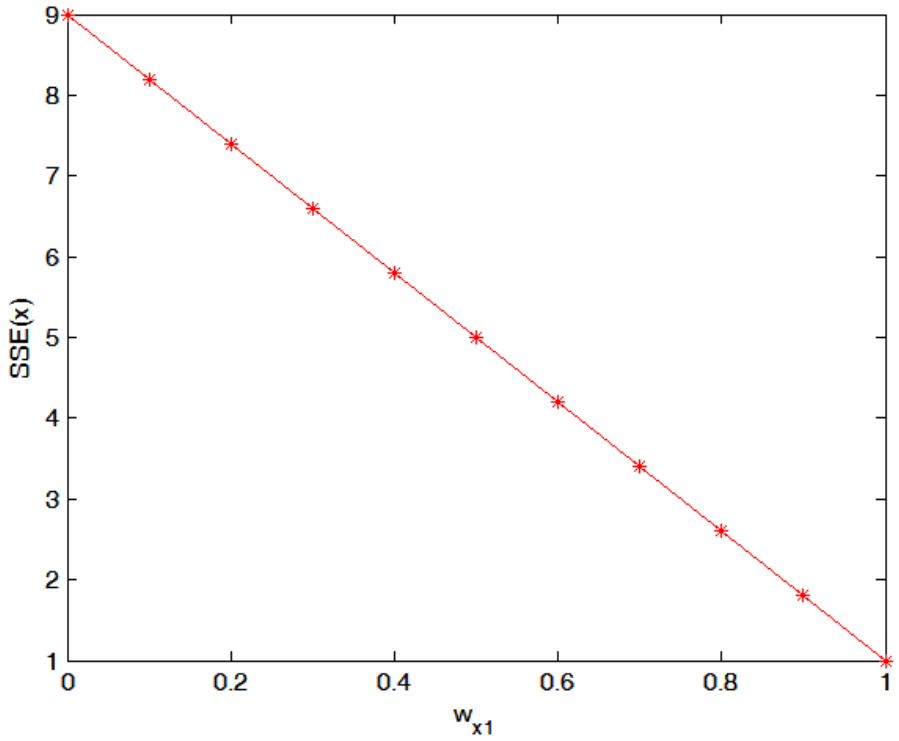
w_{ij} : weight with which object x_i belongs to cluster $\boldsymbol{c_j}$

  - To minimize SSE, repeat the following steps:
    - Fix $\boldsymbol{c_j}$ and determine w_{ij} (cluster assignment)
    - Fix w_{ij} and recompute $\boldsymbol{c_j}$
  - Hard clustering: $w_{ij} \in \{0,1\}$

Soft (Fuzzy) Clustering: Estimating Weights



$$SSE(x) = w_{x1}(2-1)^2 + w_{x2}(5-2)^2$$
$$= w_{x1} + 9w_{x2}$$



SSE(x) is minimized when $w_{x1} = 1$, $w_{x2} = 0$

Fuzzy C-means

- Objective function

p: fuzzifier (p > 1)

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{m} w_{ij}^{p} \, dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2 \qquad \sum_{j=1}^{k} w_{ij} = 1$$

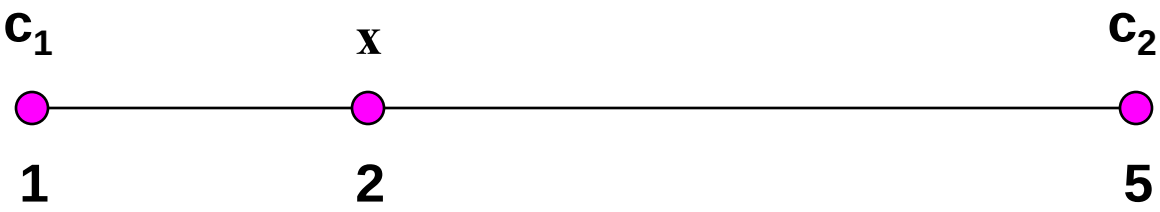
  - w_{ij}: weight with which object $\boldsymbol{x}_i$ belongs to cluster $\boldsymbol{c_j}$
  - p: a power for the weight not a superscript and controls how "fuzzy" the clustering is

  - To minimize objective function, repeat the following:
    - Fix $\boldsymbol{c_j}$ and determine w_{ij}
    - Fix w_{ij} and recompute $\boldsymbol{c}$

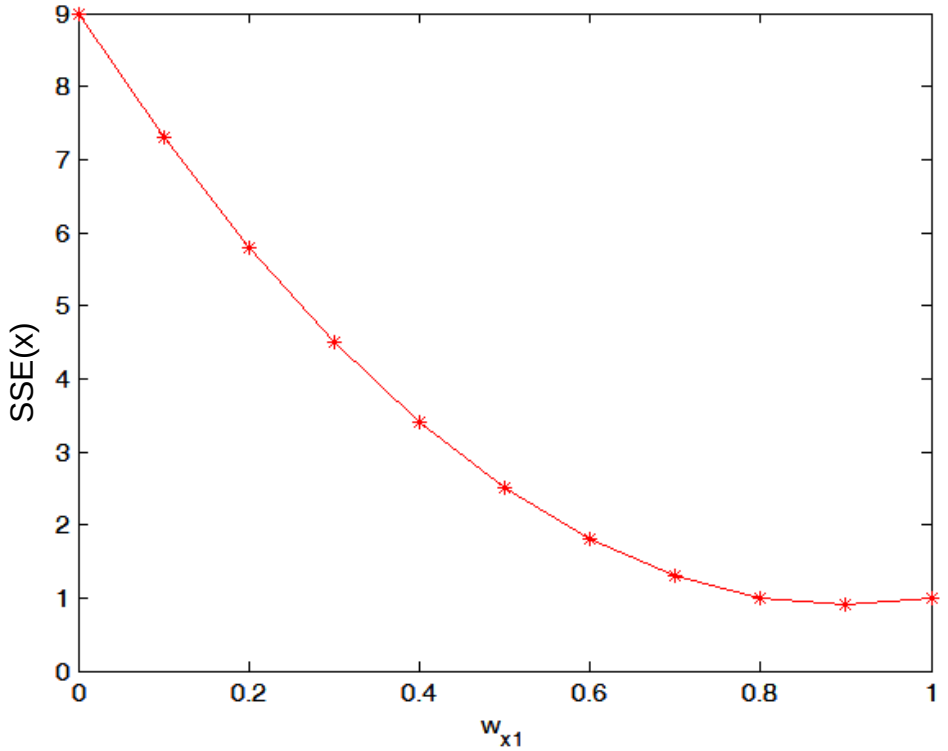
  - Fuzzy c-means clustering: $w_{ij} \in [0,1]$

Bezdek, James C. *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers, 1981.

Fuzzy C-means



$$SSE(x) = w_{x1}^2(2-1)^2 + w_{x2}^2(5-2)^2$$
$$= w_{x1}^2 + 9w_{x2}^2$$



SSE(x) is minimized when w_{x1} = 0.9, w_{x2} = 0.1

Fuzzy C-means

- Objective function:

p: fuzzifier (p > 1)

$$SSE = \sum_{j=1}^{k}\sum_{i=1}^{m} w_{ij}^{p}\, dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2 \qquad \sum_{j=1}^{k} w_{ij} = 1$$

- Initialization: choose the weights w_{ij} randomly
- Repeat:
  - Update centroids:

$$\boldsymbol{c}_j = \sum_{i=1}^{m} w_{ij}\boldsymbol{x}_i \Big/ \sum_{i=1}^{m} w_{ij}$$

  - Update weights:

$$w_{ij} = (1/dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2)^{\frac{1}{p-1}} \Big/ \sum_{j=1}^{k} (1/dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2)^{\frac{1}{p-1}}$$

Algorithm

Step-1: Randomly initialize the membership matrix using this equation,

$$\sum_{j=1}^{c} \mu_j(x_i) = 1 \qquad i = 1, 2 \ldots k$$

Step-2: Calculate the Centroid using equation,

$$Cj = \frac{\sum_i [\mu_j(x_i)]^m x_i}{\sum_i [\mu_j(x_i)]^m}$$

Step-3: Calculate dissimilarly between the data points and Centroid using the Euclidean distance.

$$D_i = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Step-4: Update the New membership matrix using the equation,

$$\mu_j(x_i) = \frac{[\frac{1}{d_{ji}}]^{1/m-1}}{\sum_{k=1}^{c} [\frac{1}{d_{ki}}]^{1/m-1}}$$

Here **m** is a fuzzification parameter.
The range **m** is always [1.25, 2]

Step -5: Go back to Step 2, unless the centroids are not changing.

Worked out Example

- ***Input:*** Number of Objects = 6 Number of clusters = 2

X	Y	C1	C2
1	6	0.8	0.2
2	5	0.9	0.1
3	8	0.7	0.3
4	4	0.3	0.7
5	7	0.5	0.5
6	9	0.2	0.8

Step-1: Initialize the membership matrix.

Step-2: Find the constraint using the equation

$$Cj = \left[\frac{\sum_i [\mu_j(x_i)]^m x_i}{\sum_i [\mu_j(x_i)]^m}, \frac{\sum_i [\mu_j(y_i)]^m y_i}{\sum_i [\mu_j(y_i)]^m}\right]$$

$$C1 = \left[\frac{1*0.8^2 + 2*0.9^2 + 3*0.7^2 + 4*0.3^2 + 5*0.5^2 + 6*0.2^2}{0.8^2 + 0.9^2 + 0.7^2 + 0.3^2 + 0.5^2 + 0.2^2}, \frac{6*0.8^2 + 5*0.9^2 + 8*0.7^2 + 4*0.3^2 + 7*0.5^2 + 9*0.2^2}{0.8^2 + 0.9^2 + 0.7^2 + 0.3^2 + 0.5^2 + 0.2^2}\right]$$

$$C1 = \frac{5.58}{2.32}, \frac{14.28}{2.32}$$

$$C1 = (2.4, 6.1)$$

$$C2 = \left[\frac{1*0.2^2 + 2*0.1^2 + 3*0.3^2 + 4*0.7^2 + 5*0.5^2 + 6*0.8^2}{0.2^2 + 0.1^2 + 0.3^2 + 0.7^2 + 0.5^2 + 0.8^2}, \right.$$

$$\left. \frac{6*0.2^2 + 5*0.1^2 + 8*0.3^2 + 4*0.7^2 + 7*0.5^2 + 9*0.8^2}{0.2^2 + 0.1^2 + 0.3^2 + 0.7^2 + 0.5^2 + 0.8^2}\right]$$

$$C2 = \frac{7.38}{1.52}, \frac{10.48}{1.52}$$

$$C_2 = (4.8, 6.8)$$

Step- 3 : Find Distance

$$D_i = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Centroid 1 :

$(1,6)(2.4,6.1) = \sqrt{(1.4)^2 + (0.1)^2} = \sqrt{1.96 + 0.01} = \sqrt{1.97} = 1.40$

$(2,5)(2.4,6.1) = \sqrt{0.16 + 1.21} = \sqrt{1.37} = 1.17$

$(3,8)(2.4,6.1) = \sqrt{0.36 + 3.61} = \sqrt{3.97} = 1.99$

$(4,4)(2.4,6.1) = \sqrt{2.56 + 4.41} = \sqrt{6.97} = 2.64$

$(5,7)(2.4,6.1) = \sqrt{6.76 + 0.81} = \sqrt{7.57} = 2.75$

$(6,9)(2.4,6.1) = \sqrt{12.96 + 8.41} = \sqrt{21.37} = 4.62$

Cluster 1		Cluster 2	
Datapoint	**Distance**	**Datapoint**	**Distance**
(1,6)	1.40	(1,6)	3.88
(2,5)	1.17	(2,5)	3.32
(3,8)	1.99	(3,8)	2.16
(4,4)	2.64	(4,4)	2.91
(5,7)	2.75	(5,7)	0.28
(6,9)	4.62	(6,9)	2.50

Centroid 2:

$(1,6)(4.8,6.8) = \sqrt{14.44 + 0.64} = \sqrt{15.08} = 3.88$

$(2,5)(4.8,6.8) = \sqrt{7.84 + 3.24} = \sqrt{11.08} = 3.32$

$(3,8)(4.8,6.8) = \sqrt{3.24 + 1.44} = \sqrt{4.68} = 2.16$

$(4,4)(4.8,6.8) = \sqrt{0.64 + 7.84} = \sqrt{8.48} = 2.91$

$(5,7)(4.8,6.8) = \sqrt{0.04 + 0.04} = \sqrt{0.08} = 0.28$

$(6,9)(4.8,6.8) = \sqrt{1.44 + 4.84} = \sqrt{6.28} = 2.50$

Step-4 : Update the membership value

$$\mu_j(x_i) = \frac{[\frac{1}{d_{ji}}]^{1/m-1}}{\sum_{k=1}^{c}[\frac{1}{d_{ki}}]^{1/m-1}}$$

here m = 2, i – first data point, j - first cluster

Cluster 1

$\mu_{11} = (1 / d_{11})^{1/2-1} / (1 / d_{11})^{1/2-1} + (1 / d_{21})^{1/2-1}$

$= (1/1.40)^{1} / (1/1.40)^{1} + (1/3.88)^{1} = 0.71 / 0.71+0.25$

$= 0.71 / 0.96 = 0.7$

$\mu_{12} = (1 / d_{12}) / (1 / d_{12}) + (1 / d_{22})$

$\Rightarrow 1/1.17 / 1/1.17 + 1/3.32 = 0.56 / 0.56+0.30$

$= 0.56 / 0.86 = 0.6$

$\mu_{13} = (1 / d_{13}) / (1 / d_{13}) + (1 / d_{23})$

$\Rightarrow 1/1.99 / 1/1.99 + 1/2.16 = 0.50 / 0.50+0.46$

$= 0.50 / 0.96 = 0.5$

$\mu_{14} = (1 / d_{14}) / (1 / d_{14}) + (1 / d_{24})$

$\Rightarrow 1/2.64 / 1/2.64 + 1/2.91 = 0.37 / 0.37+0.34$

$= 0.37 / 0.71 = 0.5$

$\mu_{15} = (1 / d_{15}) / (1 / d_{15}) + (1 / d_{25})$

$\Rightarrow 1/2.75 / 1/2.75 + 1/0.28 = 0.36 / 0.36+3.57$

$= 0.36 / 3.93 = 0.1$

$\mu_{16} = (1 / d_{16}) / (1 / d_{16}) + (1 / d_{26})$

$\Rightarrow 1/4.62 / 1/4.62 + 1/2.50 = 0.21 / 0.21+0.4$

$= 0.21 / 0.61 = 0.3$

Cluster 2

$\mu_{21} = (1 / d_{21}) / (1 / d_{12}) + (1 / d_{21})$

$\Rightarrow 1/3.88 / 1/1.40 + 1/3.88 = 0.25 / 0.71+0.25$

$= 0.25 / 0.96 = 0.3$

$\mu_{22} = (1 / d_{22}) / (1 / d_{12}) + (1 / d_{22})$

$\Rightarrow 1/3.32 / 1/1.17 + 1/3.32 = 0.30 / 0.56+0.30$

$= 0.30 / 0.86 = 0.4$

$\mu_{23} = (1 / d_{23}) / (1 / d_{13}) + (1 / d_{23})$

$\Rightarrow 1/2.16 / 1/1.99 + 1/2.16 = 0.46 / 0.50+0.46$

$= 0.46 / 0.96 = 0.5$

$\mu_{24} = (1 / d_{24}) / (1 / d_{14}) + (1 / d_{24})$

$\Rightarrow 1/2.19 / 1/2.64 + 1/2.19 = 0.34 / 0.37+0.34$

$= 0.34 / 0.71 = 0.5$

$\mu_{25} = (1 / d_{25}) / (1 / d_{15}) + (1 / d_{25})$

$\Rightarrow 1/0.28 / 1/2.75 + 1/0.28 = 3.57 / 0.36+3.57$

$= 3.57 / 3.93 = 0.9$

$\mu_{26} = (1 / d_{26}) / (1 / d_{16}) + (1 / d_{26})$

$= 0.4 / 0.21+0.4 = 0.4/0.61$

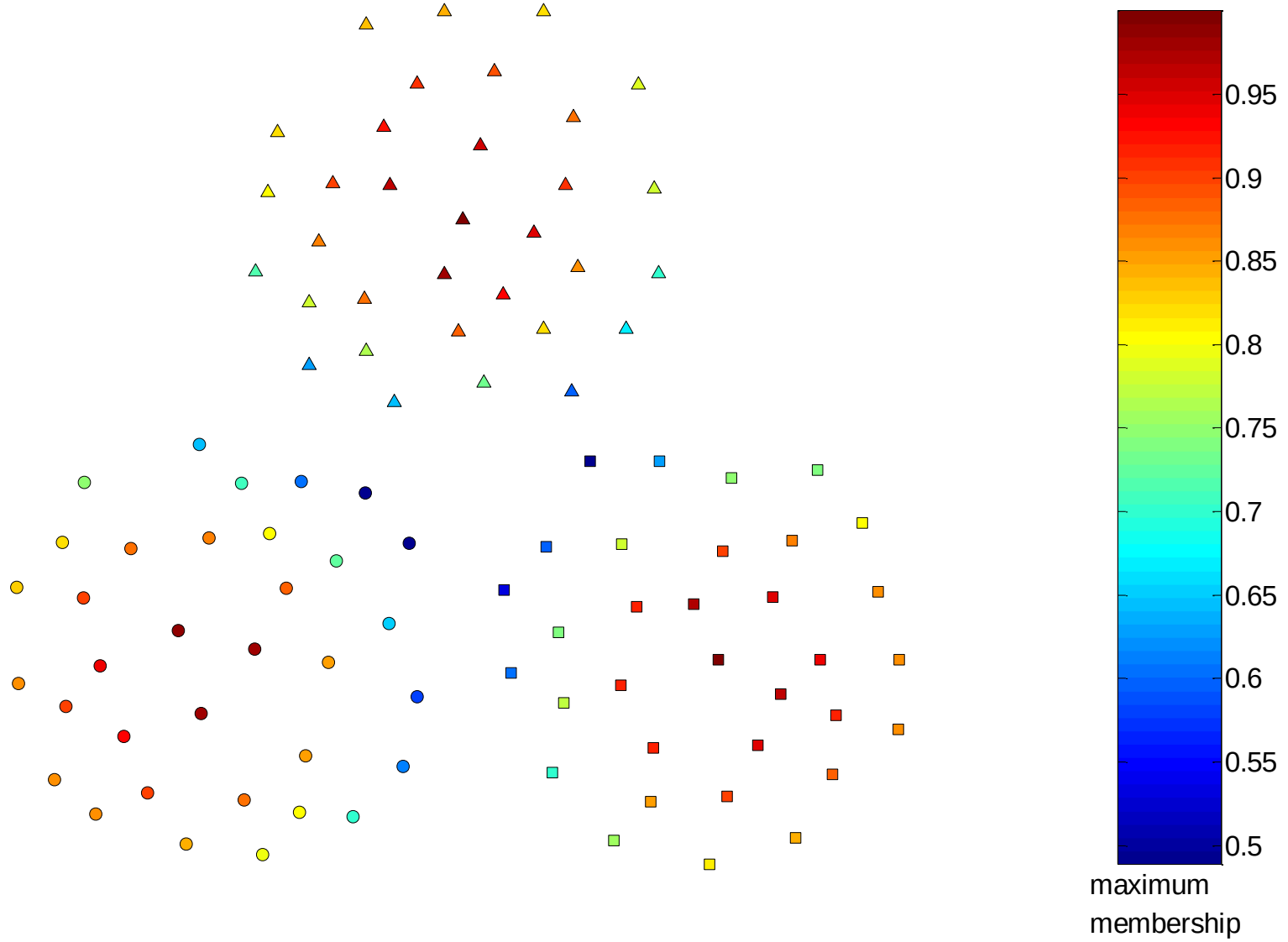
$= 0.7$

- Now the New Membership value is

X	Y	C1	C2
1	6	0.7	0.3
2	5	0.6	0.4
3	8	0.5	0.5
4	4	0.5	0.5
5	7	0.1	0.9
6	9	0.3	0.7

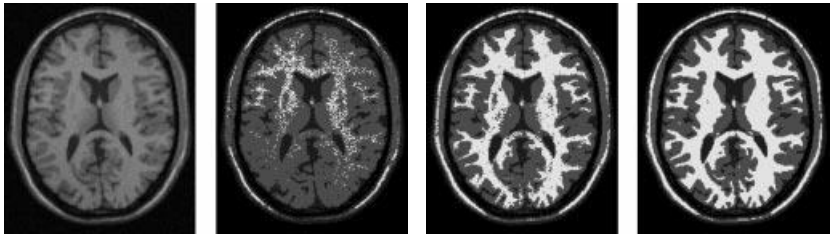
Step 5 : Now continue this process until get the same centroids.

Fuzzy K-means Applied to Sample Data



An Example Application: Image Segmentation

- Modified versions of fuzzy c-means have been used for image segmentation
  - Especially fMRI images (functional magnetic resonance images)
- References
  - Gong, Maoguo, Yan Liang, Jiao Shi, Wenping Ma, and Jingjing Ma. "Fuzzy c-means clustering with local information and kernel metric for image segmentation." *Image Processing, IEEE Transactions on* 22, no. 2 (2013): 573-584.



From left to right: original images, fuzzy c-means, EM, BCFCM

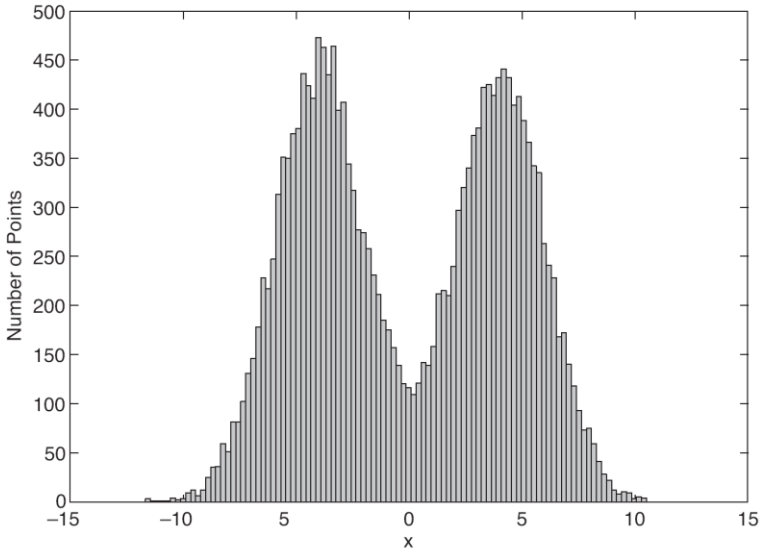
  - Ahmed, Mohamed N., Sameh M. Yamany, Nevin Mohamed, Aly A. Farag, and Thomas Moriarty. "A modified fuzzy c-means algorithm for bias field estimation and segmentation of MRI data." *Medical Imaging, IEEE Transactions on* 21, no. 3 (2002): 193-199.

Hard (Crisp) vs Soft (Probabilistic) Clustering

- Idea is to model the set of data points as arising from a mixture of distributions
  - Typically, normal (Gaussian) distribution is used
  - But other distributions have been very profitably used

- Clusters are found by estimating the parameters of the statistical distributions
  - Can use a k-means like algorithm, called the Expectation-Maximization (EM) algorithm, to estimate these parameters
    - Actually, k-means is a special case of this approach
  - Provides a compact representation of clusters
  - The probabilities with which point belongs to each cluster provide a functionality similar to fuzzy clustering.

Probabilistic Clustering: Example



- Informal example: consider modeling the points that generate the following histogram.
- Looks like a combination of two normal (Gaussian) distributions
- Suppose we can estimate the mean and standard deviation of each normal distribution.
  - This completely describes the two clusters
  - We can compute the probabilities with which each point belongs to each cluster
  - Can assign each point to the cluster (distribution) for which it is most probable.

$$prob(x_i|\Theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Probabilistic Clustering: EM Algorithm

Initialize the parameters

Repeat

For each point, compute its probability under each distribution

Using these probabilities, update the parameters of each distribution

Until there is no change

- Very similar to K-means
- Consists of assignment and update steps
- Can use random initialization
  - Problem of local minima
- For normal distributions, typically use K-means to initialize
- If using normal distributions, can find elliptical as well as spherical shapes.

Probabilistic Clustering: Updating Centroids

Update formula for weights assuming an estimate for statistical parameters

$$\boldsymbol{c}_j = \sum_{i=1}^{m} \boldsymbol{x}_i p(C_j \mid \boldsymbol{x}_i) \Big/ \sum_{i=1}^{m} p(C_j \mid \boldsymbol{x}_i)$$

$\boldsymbol{x_i}$ *is a data point*
C_j *is a cluster*
$\boldsymbol{c_j}$ *is a centroid*

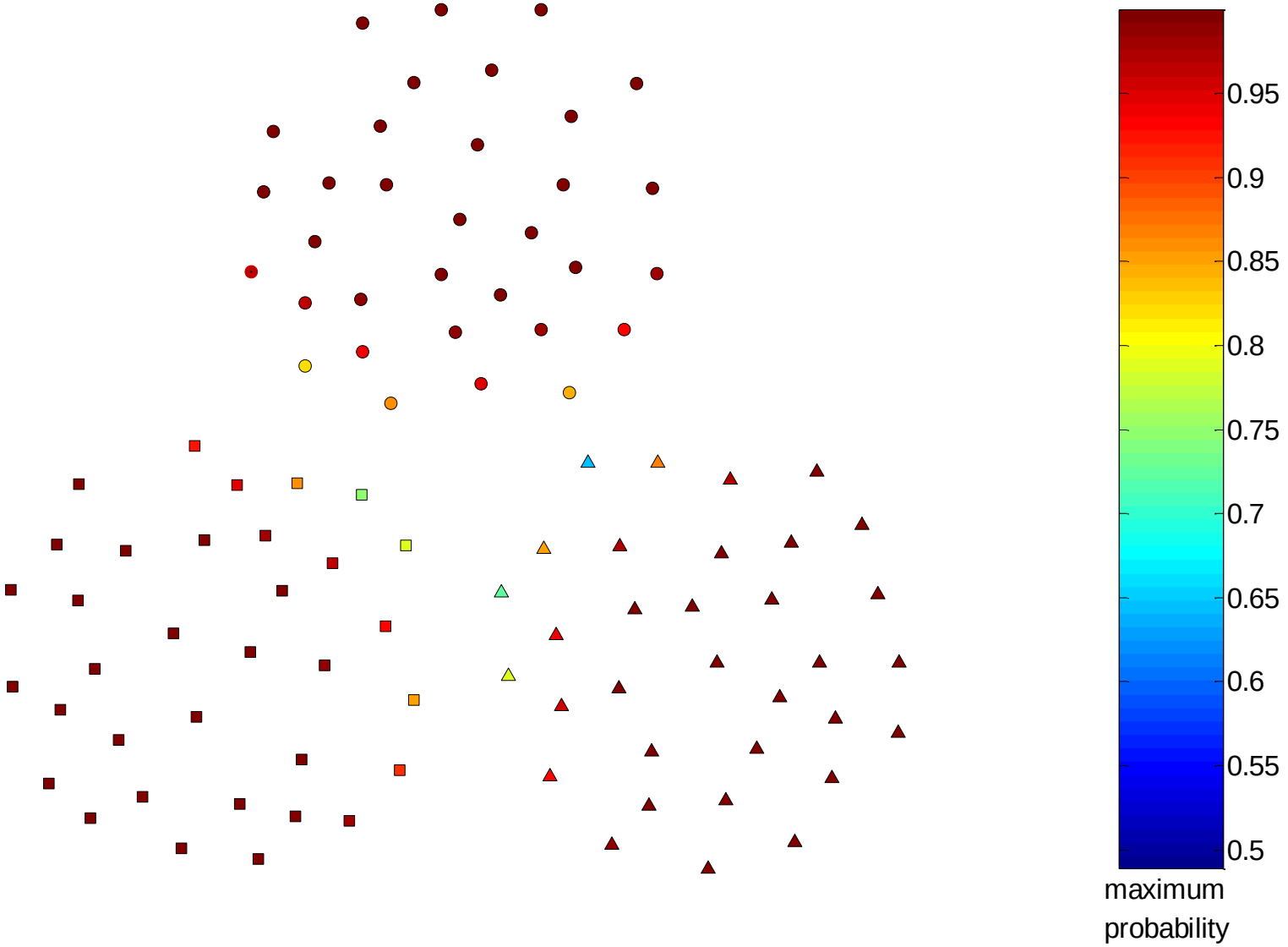
- Very similar to the fuzzy k-means formula
  - Weights are probabilities
  - Weights are not raised to a power
  - Probabilities calculated using Bayes rule: $p(C_j|\boldsymbol{x_i}) = \frac{p(\boldsymbol{x_i}|C_j)p(C_j)}{\sum_{l=1}^{k} p(\boldsymbol{x_i}|C_l)p(C_l)}$
- Need to assign weights to each cluster
  - Weights may not be equal
  - Similar to prior probabilities
  - Can be estimated: $p(C_j) = \frac{1}{m}\sum_{i=1}^{m} p(C_j \mid \boldsymbol{x}_i)$

More Detailed EM Algorithm

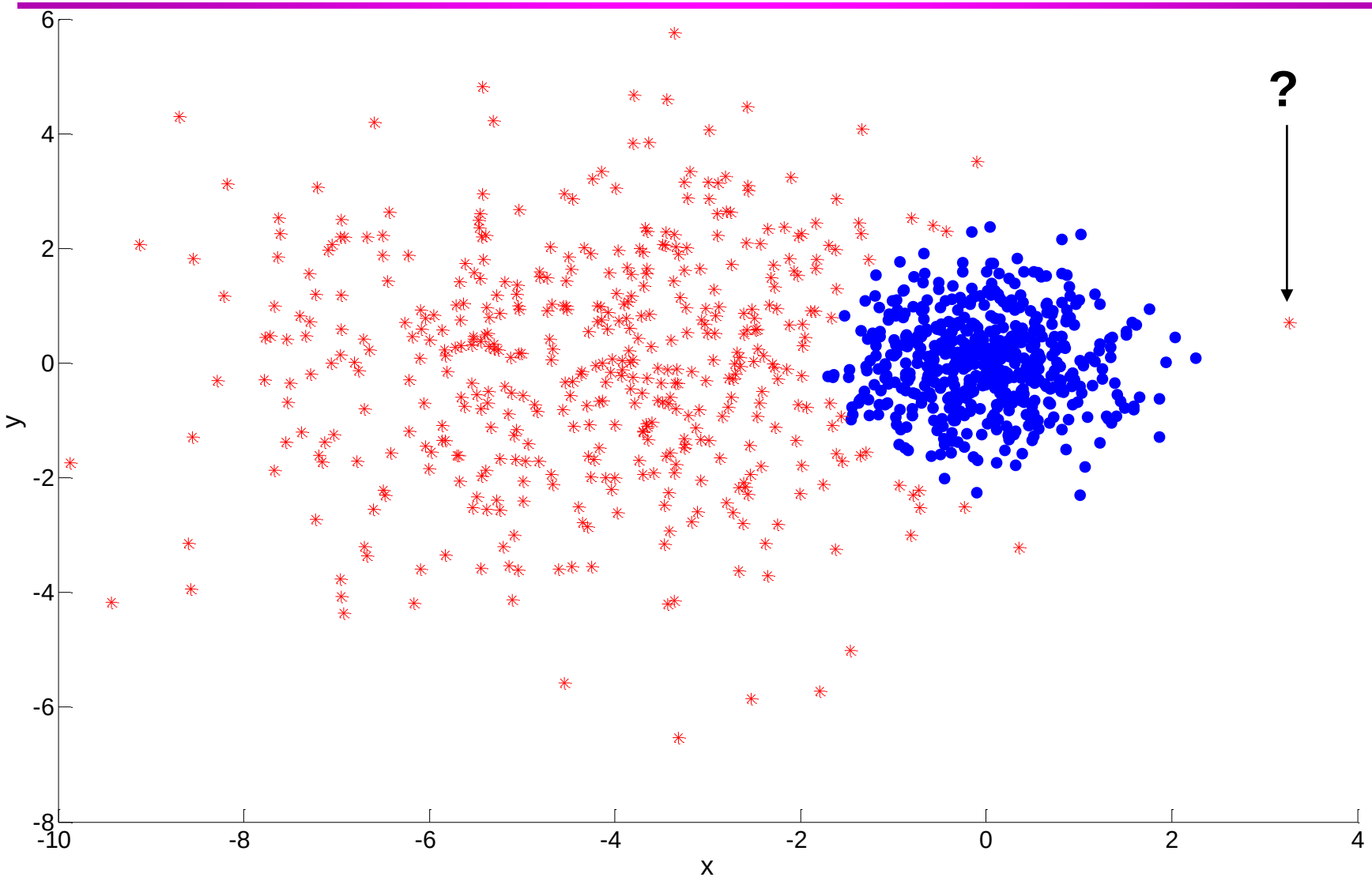
Algorithm 9.2 EM algorithm.

1: Select an initial set of model parameters.
(As with K-means, this can be done randomly or in a variety of ways.)
2: **repeat**
3: **Expectation Step** For each object, calculate the probability that each object belongs to each distribution, i.e., calculate $prob(distribution\ j|\mathbf{x}_i, \Theta)$.
4: **Maximization Step** Given the probabilities from the expectation step, find the new estimates of the parameters that maximize the expected likelihood.
5: **until** The parameters do not change.
(Alternatively, stop if the change in the parameters is below a specified threshold.)

Probabilistic Clustering Applied to Sample Data



Probabilistic Clustering: Dense and Sparse Clusters



Problems with EM

- Convergence can be slow
- Only guarantees finding local maxima
- Makes some significant statistical assumptions
- Number of parameters for Gaussian distribution grows as $O(d^2)$, d the number of dimensions
  - Parameters associated with covariance matrix
  - K-means only estimates cluster means, which grow as $O(d)$

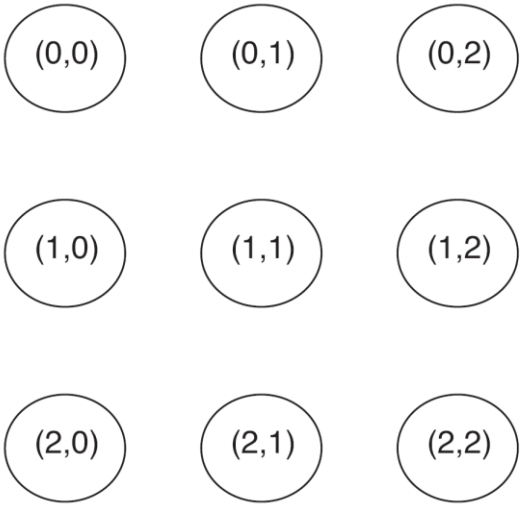
Alternatives to EM

- Method of moments / Spectral methods
  - ICML 2014 workshop bibliography

    https://sites.google.com/site/momentsicml2014/bibliography
- Markov chain Monte Carlo (MCMC)
- Other approaches

SOM: Self-Organizing Maps

- Self-organizing maps (SOM)
  - Centroid based clustering scheme
  - Like K-means, a fixed number of clusters are specified
  - However, the spatial relationship of clusters is also specified, typically as a grid
  - Points are considered one by one
  - Each point is assigned to the closest centroid
  - Other centroids are updated based on their nearness to the closest centroid



Kohonen, Teuvo, and Self-Organizing Maps. "Springer series in information sciences." *Self-organizing maps* 30 (1995).

SOM: Self-Organizing Maps

Algorithm 9.3 Basic SOM Algorithm.

1: Initialize the centroids.
2: **repeat**
3: Select the next object.
4: Determine the closest centroid to the object.
5: Update this centroid and the centroids that are close, i.e., in a specified neighborhood.
6: **until** The centroids don't change much or a threshold is exceeded.
7: Assign each object to its closest centroid and return the centroids and clusters.

- Updates are weighted by distance
  - Centroids farther away are affected less
- The impact of the updates decreases with each time
  - At some point the centroids will not change much

More details for algorithm

1 Initialize M weight vectors. Set topological neighborhood parameters and learning rate, $\alpha(< 1)$.

2 For step $k = 1, 2, \ldots,$ do steps a - d by cycling through training set until weight vectors converge

a Set input vector $\mathbf{x} = \mathbf{s}^{(q)}$, one of the training vectors.

b Compute for each cluster unit $j = 1, \ldots, M$ the Euclidean distance

$$d_j = \sum_{i \neq 1}^{N} (x_i - w_{ij}(k))^2.$$

c Find the index j' such that $d_{j'}$ is a minimum.

d For all cluster units j within the specified neighborhoods of j', update the weight vectors

$$w_{ij}(k+1) = w_{ij}(k) + \alpha \left[x_i - w_{ij}(k) \right], \quad i = 1, \ldots, N.$$

e May reduce the learning rate.

f May reduce the radii that define the topological neighborhoods.

Learning Rate

- The above updating rule moves the weight vectors for the winning neuron and those in its neighborhood towards the input vector. The amount of change is proportional to . In the extreme limit of α = 1, all these weight vectors are set to the input vector.
- During training, the learning rate can be decreased linearly, that is

$$\alpha(k) = \frac{\alpha(1)}{k}$$

where k = 1, 2, . . . is the iteration counter

- Geometric decrease of :

α(k + 1) = α f(k) where 0 < f < 1, also works.

Example

- We now consider an example where 4 input vectors

$$\mathbf{s}^{(1)} = [\ 1 \quad 1 \quad 0 \quad 0\ ] \qquad \mathbf{s}^{(2)} = [\ 0 \quad 0 \quad 0 \quad 1\ ]$$
$$\mathbf{s}^{(3)} = [\ 1 \quad 0 \quad 0 \quad 0\ ] \qquad \mathbf{s}^{(4)} = [\ 0 \quad 0 \quad 1 \quad 1\ ]$$

are to be grouped into 2 clusters. It is clear that N = 4, M = 2 and Q = 4.

- Suppose the initial learning rate is (1) = 0.6, we use a geometric schedule with f = 0.5 for decreasing α.

- We assume that the initial weight matrix is given by

$$\mathbf{W} = \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.5 & 0.7 \\ 0.9 & 0.3 \end{bmatrix}$$

$$
\begin{aligned}
d_1 &= \| \begin{bmatrix} 0.2 & 0.6 & 0.5 & 0.9 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \|^2 \\
&= \| \begin{bmatrix} -0.8 & -0.4 & 0.5 & 0.9 \end{bmatrix} \|^2 = 1.86.
\end{aligned}
$$

and

$$
\begin{aligned}
d_2 &= \| \begin{bmatrix} 0.8 & 0.4 & 0.7 & 0.3 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \|^2 \\
&= \| \begin{bmatrix} -0.2 & -0.6 & 0.7 & 0.3 \end{bmatrix} \|^2 = 0.98.
\end{aligned}
$$

Since d_2 is the smallest, the winning neuron is $j' = 2$. So we update the second weight vector

$$
\begin{aligned}
\mathbf{W}_{\cdot 2} &= \begin{bmatrix} 0.8 & 0.4 & 0.7 & 0.3 \end{bmatrix} + \\
&\quad \alpha(1) \left(\begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0.8 & 0.4 & 0.7 & 0.3 \end{bmatrix} \right) \\
&= \begin{bmatrix} 0.92 & 0.76 & 0.28 & 0.12 \end{bmatrix}.
\end{aligned}
$$

The weight matrix is now given by

$$
\mathbf{W} = \begin{bmatrix} 0.2 & 0.92 \\ 0.6 & 0.76 \\ 0.5 & 0.28 \\ 0.9 & 0.12 \end{bmatrix}.
$$

Then we present $\mathbf{x} = \mathbf{s}^{(2)}$. we find

$$\begin{aligned} d_1 &= \| \begin{bmatrix} 0.2 & 0.6 & 0.5 & 0.9 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \|^2 \\ &= \| \begin{bmatrix} 0.2 & 0.6 & 0.5 & -0.1 \end{bmatrix} \|^2 = 0.66. \end{aligned}$$

and

$$\begin{aligned} d_2 &= \| \begin{bmatrix} 0.92 & 0.76 & 0.28 & 0.12 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \|^2 \\ &= \| \begin{bmatrix} 0.92 & 0.76 & 0.28 & -0.88 \end{bmatrix} \|^2 = 2.2768. \end{aligned}$$

Since d_1 is the smallest, the winning neuron is $j' = 1$. So we update the first weight vector

$$\begin{aligned} \mathbf{W}_{\cdot 1} &= \begin{bmatrix} 0.2 & 0.6 & 0.5 & 0.9 \end{bmatrix} + \\ &\quad \alpha(1) \left(\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.2 & 0.6 & 0.5 & 0.9 \end{bmatrix} \right) \\ &= \begin{bmatrix} 0.92 & 0.76 & 0.28 & 0.12 \end{bmatrix}. \end{aligned}$$

The weight matrix is then given by

$$\mathbf{W} = \begin{bmatrix} 0.08 & 0.92 \\ 0.24 & 0.76 \\ 0.20 & 0.28 \\ 0.96 & 0.12 \end{bmatrix}.$$

We continue this process for the remaining two training vectors in the data set to obtain the weight matrix

$$\mathbf{W} = \begin{bmatrix} 0.032 & 0.968 \\ 0.096 & 0.304 \\ 0.680 & 0.112 \\ 0.984 & 0.048 \end{bmatrix}$$

after the first epoch. Next we reduce the learning rate by a multiplicative factor $f = 0.5$ and repeat another round through the training set.

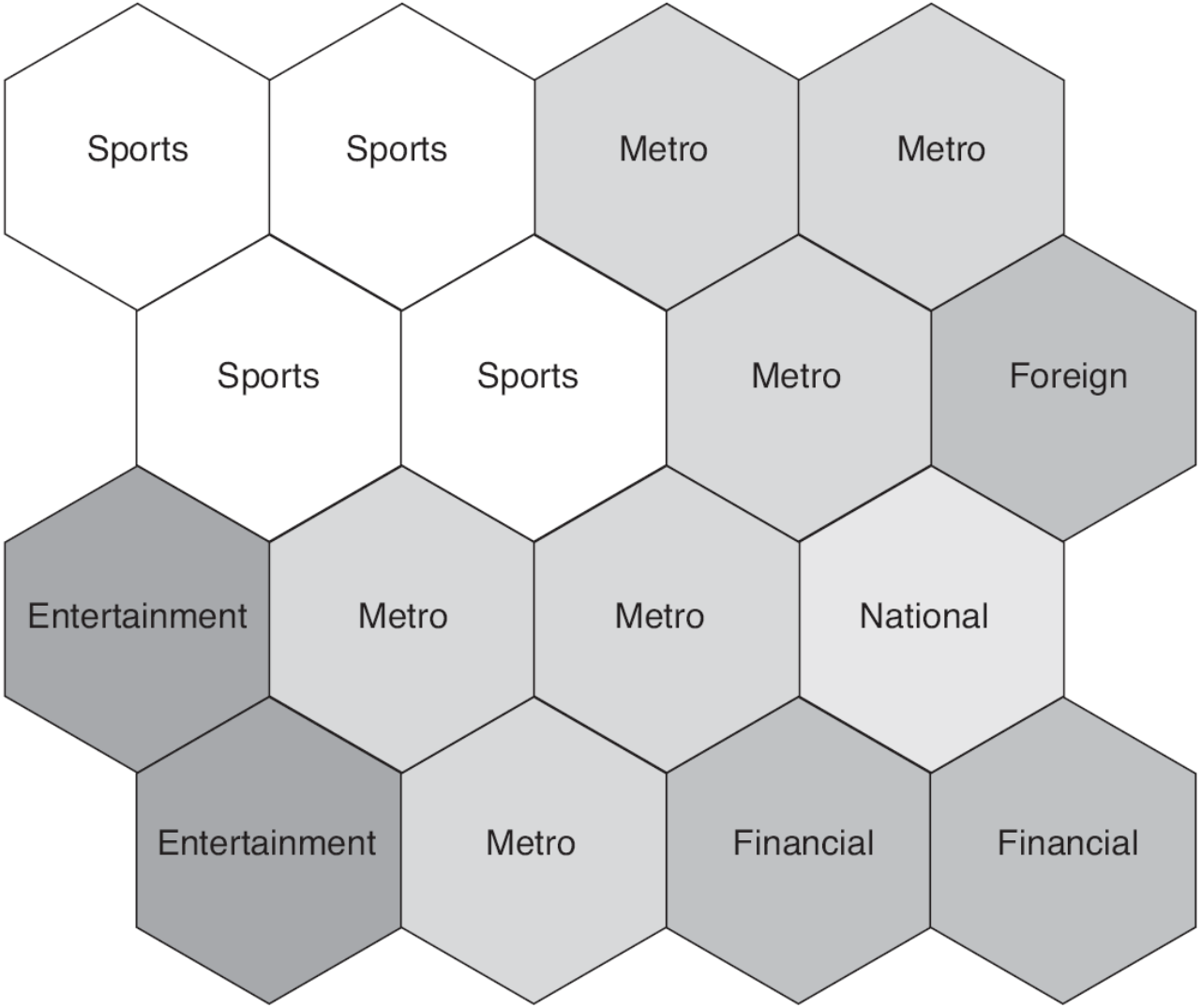
After 100 iterations, the learning rate decreases to 0.006, and the weight matrix becomes

$$\mathbf{W} = \begin{bmatrix} 1.61e-4 & 0.99984 \\ 2.0e-16 & 0.49376 \\ 0.50616 & 5.65e-4 \\ 0.99992 & 2.42e-4 \end{bmatrix}$$

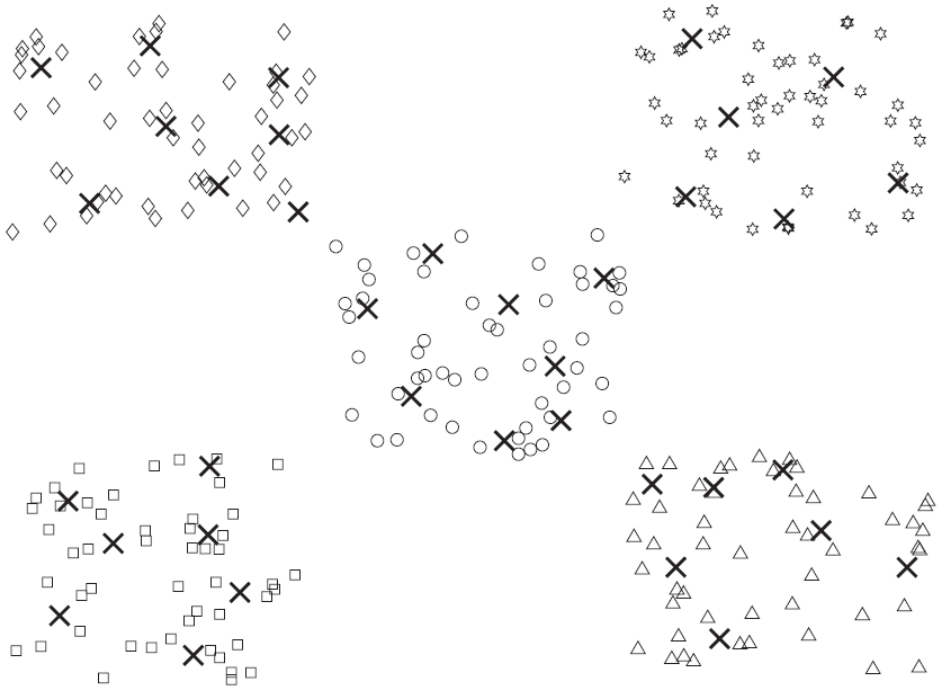
SOM: Self-Organizing Maps

- SOM can be viewed as a type of dimensionality reduction
- If a two-dimensional grid is used, the results can be visualized

SOM Clusters of LA Times Document Data



Another SOM Example: 2D Points



(a) Distribution of SOM reference vectors (**X**'s) for a two-dimensional point set.

diamond	diamond	diamond	hexagon	hexagon	hexagon
diamond	diamond	diamond	circle	hexagon	hexagon
diamond	diamond	circle	circle	circle	hexagon
square	square	circle	circle	triangle	triangle
square	square	circle	circle	triangle	triangle
square	square	square	triangle	triangle	triangle

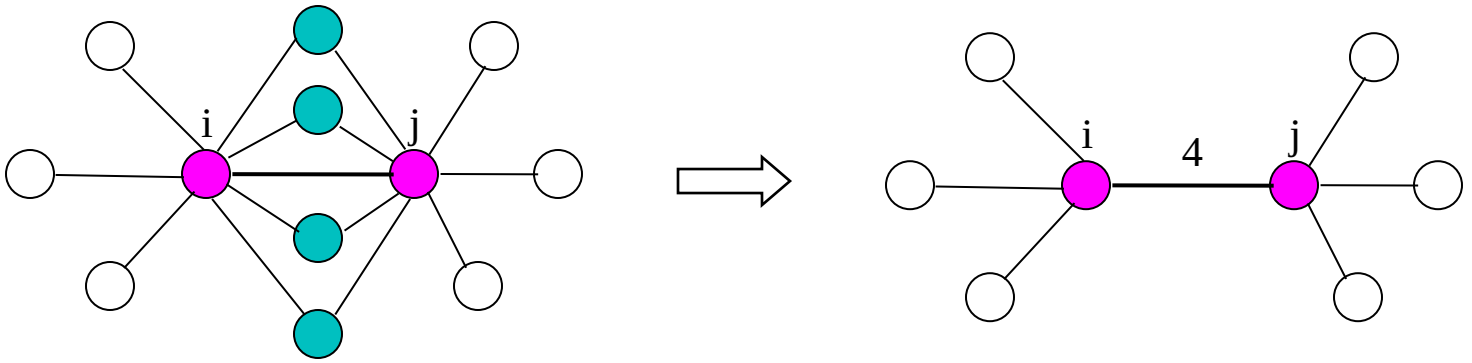
(b) Classes of the SOM centroids.

Issues with SOM

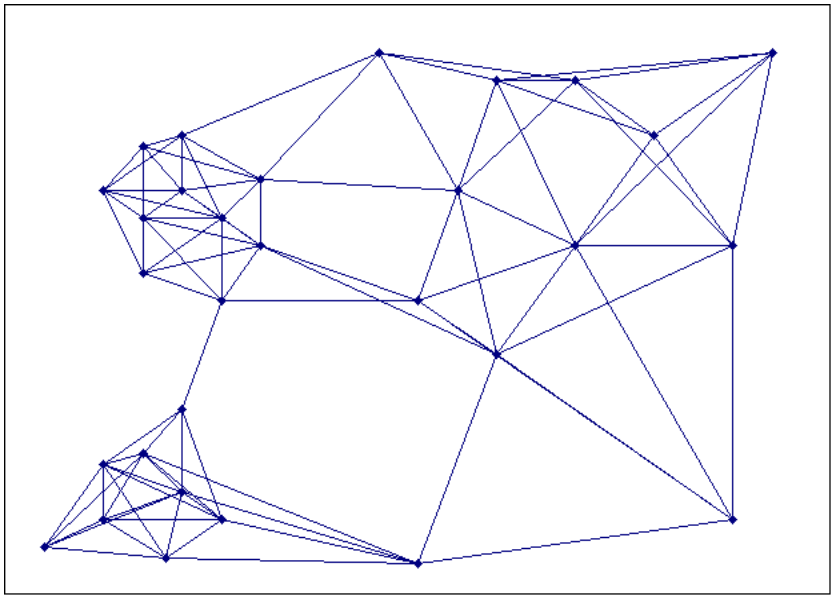
- Computational complexity
- Locally optimal solution
- Grid is somewhat arbitrary

Graph-Based Clustering: SNN Approach

Shared Nearest Neighbor (SNN) graph: the weight of an edge is the number of shared neighbors between vertices given that the vertices are connected

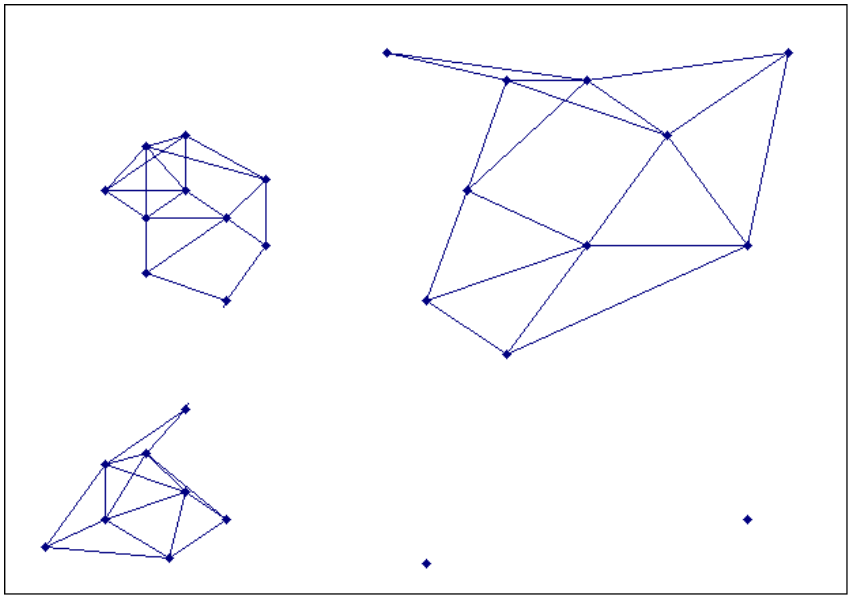


Creating the SNN Graph



Sparse Graph

Link weights are similarities between neighboring points



Shared Near Neighbor Graph

Link weights are number of Shared Nearest Neighbors

SNN Density-Based Clustering

- Combines:
  - Graph based clustering (similarity definition based on number of shared nearest neighbors)
  - Density based clustering (DBSCAN-like approach)

- SNN density measures whether a point is surrounded by similar points (with respect to its nearest neighbors)

SNN Clustering Algorithm

1. **Compute the similarity matrix**
   This corresponds to a similarity graph with data points for nodes and edges whose weights are the similarities between data points

2. **Sparsify the similarity matrix by keeping only the *k* most similar neighbors**
   This corresponds to only keeping the *k* strongest links of the similarity graph

3. **Construct the shared nearest neighbor graph from the sparsified similarity matrix.**
   At this point, we could apply a similarity threshold and find the connected components to obtain the clusters (Jarvis-Patrick algorithm)

4. **Find the SNN density of each Point.**
   Using a user specified parameters, *Eps*, find the number points that have an SNN similarity of *Eps* or greater to each point. This is the SNN density of the point

SNN Clustering Algorithm …

5. **Find the core points**
   Using a user specified parameter, *MinPts*, find the core points, i.e., all points that have an SNN density greater than *MinPts*

6. **Form clusters from the core points**
   If two core points are within a "radius", *Eps*, of each other they are placed in the same cluster

7. **Discard all noise points**
   All non-core points that are not within a "radius" of *Eps* of a core point are discarded

8. **Assign all non-noise, non-core points to clusters**
   This can be done by assigning such points to the nearest core point

   (Note that steps 4-8 are DBSCAN)

Example

Consider, a dataset of eight geographical locations where individuals live. The purpose is to divide these individuals into groups based on their geographical locations, as determined by the Global Positioning System.

This chart shows a simple dataset of individuals' geographic data. Assume that all the data collected about these eight individuals was collected at a specific point in time.

Individual ID	GPS – Geographical Longitude	GPS – Geographical Latitude
1	2	10
2	2	5
3	8	4
4	5	8
5	7	5
6	6	4
7	1	2
8	4	9

Example

Similarity Matrix

	Individual #1	Individual #2	Individual #3	Individual #4	Individual #5	Individual #6	Individual #7	Individual #8
Individual #1	0	5	6	3.6	7.07	7.21	8.062	2.23
Individual #2		0	6.8	4.24	5	4.12	3.16	4.47
Individual #3			0	5	1.41	1.41	7.28	6.4
Individual #4				0	3.31	4.12	7.21	1.41
Individual #5					0	1.41	6.7	5
Individual #6						0	5.38	5.38
Individual #7							0	7.61
Individual #8								0

The similarity of these data elements is calculated as a Euclidean distance. Individuals with similarity values closer to 0 have greater similarity. Half the matrix is not filled because the matrix is *symmetric*.

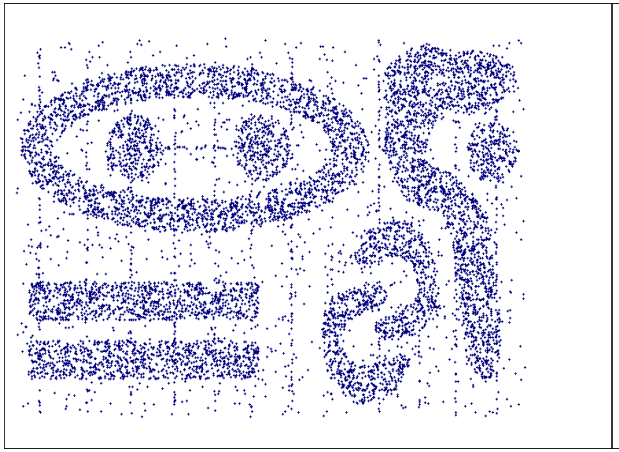
- Moving the focus to Individual 3, you find that the similarity between Individual 3 and Individual 2 & 1 is larger than the threshold value 4.5. Thus you assign Individual 3 to a new cluster containing one item: C3 = {Individual 3}.
- Moving to Individual 4, you calculate how similar Individual 4 is to Individuals 1, 2, and 3. The nearest (most similar) to Individual 4 happens to be Individual 1. The similarity between 4 and 1 is approximately 3.6, which is less than the threshold value 4.5.
- Individual 4 joins Individual 1 in Cluster C1.

- Next is to examine Individual 5 and calculate how similar it is to Individuals 1, 2, 3, and 4. The item nearest in distance (most similar) to Individual 5 is Individual 3. The similarity is $\sqrt{2}$, which is less than the threshold value of 4.5. Thus Individual 5 joins C3.
- When you examine Individual 6 and calculate how similar it is to Individuals 1, 2, 3, 4, and 5, you discover that Individual 3 is nearest (most similar) to Individual 6. Thus Individual 6 joins C3.
- When you examine Individual 7 and calculate how similar it is to Individuals 1, 2, 3, 4, 5, and 6, you find that the nearest (most similar) item to Individual 7 is Individual 2. Thus Individual 7 joins C2.
- When you examine Individual 8, and calculate its similarity to Individuals 1, 2, 3, 4, and 5, you find that the nearest (most similar) item to Individual 8 is Individual 4. Thus Individual 8 joins C1.

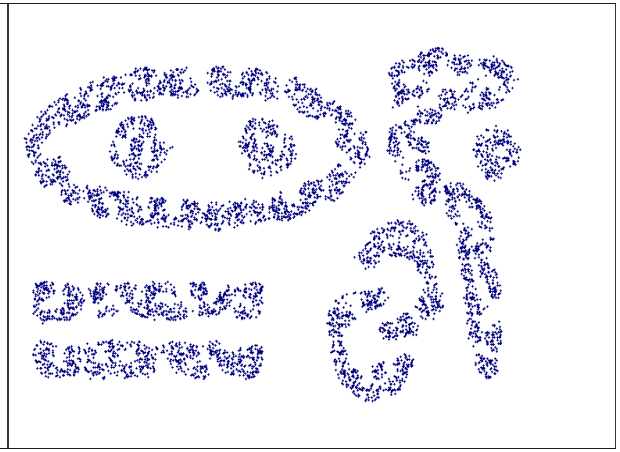
- The clusters constructed so far, containing items most similar to each other, are

C1 = {Individual 1, Individual 4, Individual 8}
C2 = {Individual 2, Individual 7}
C3 = {Individual 3, Individual 5, Individual 6}

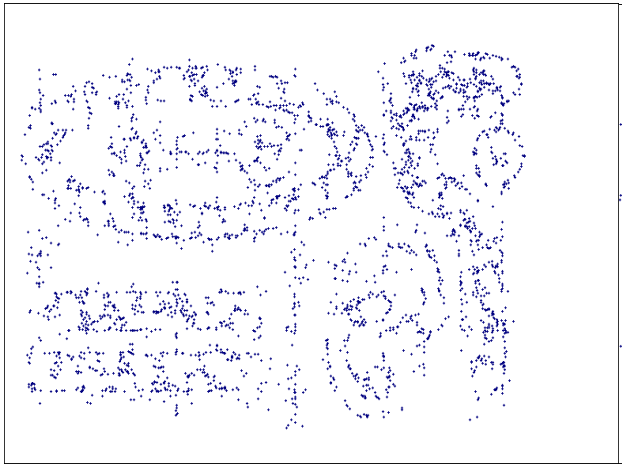
SNN Density



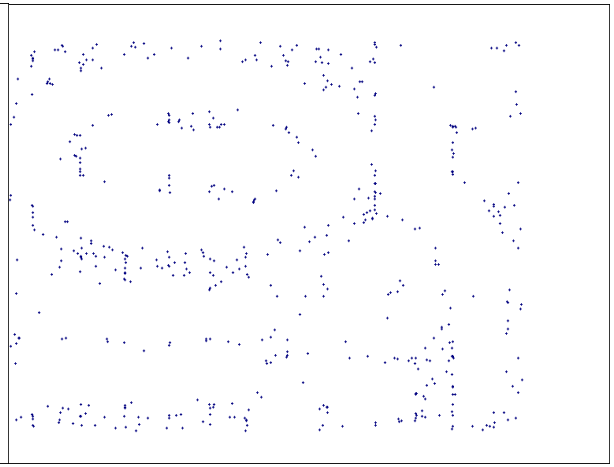
a) All Points



b) High SNN Density

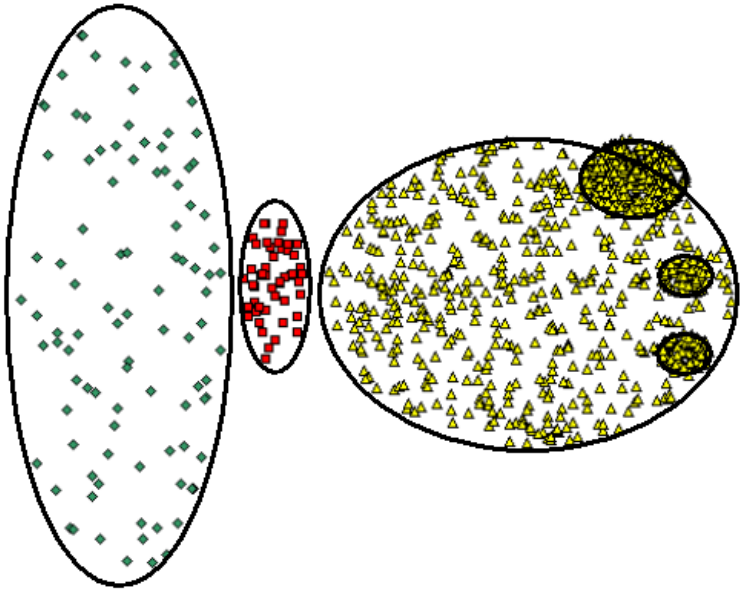


c) Medium SNN Density

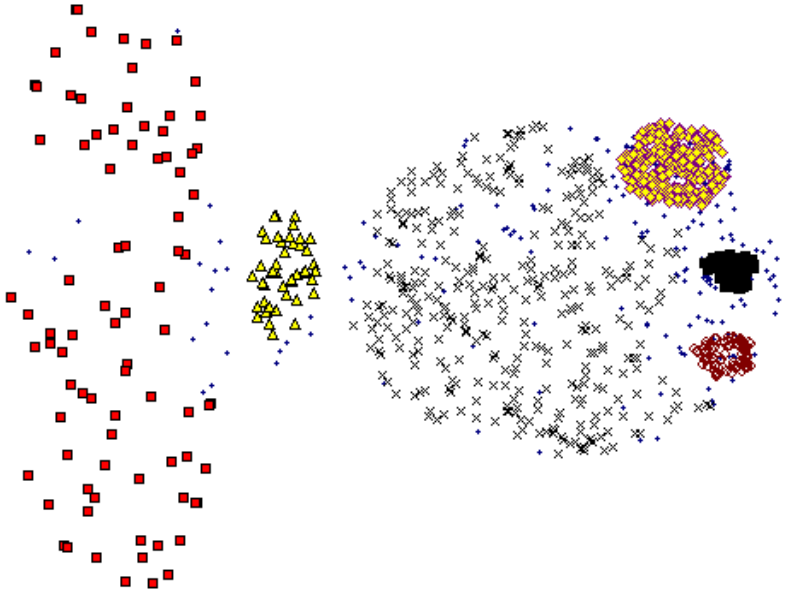


d) Low SNN Density

SNN Clustering Can Handle Differing Densities

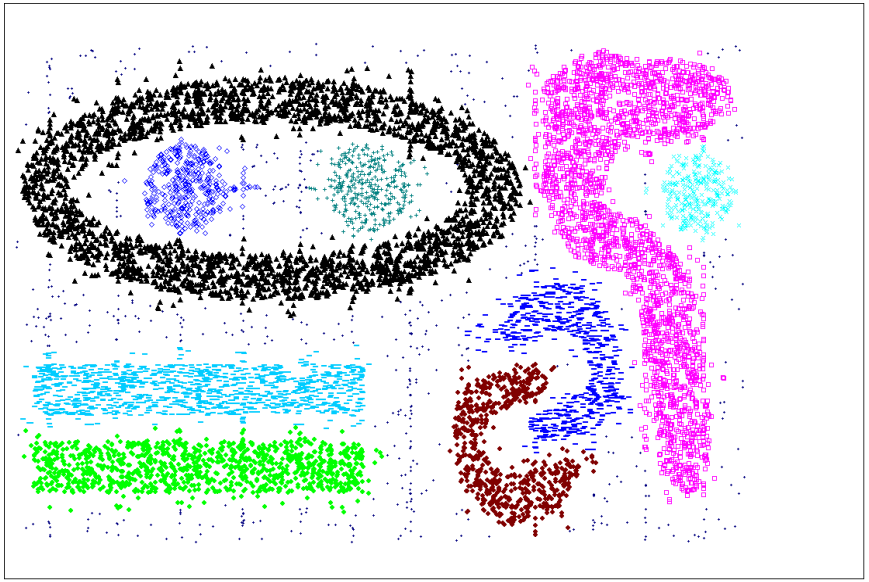


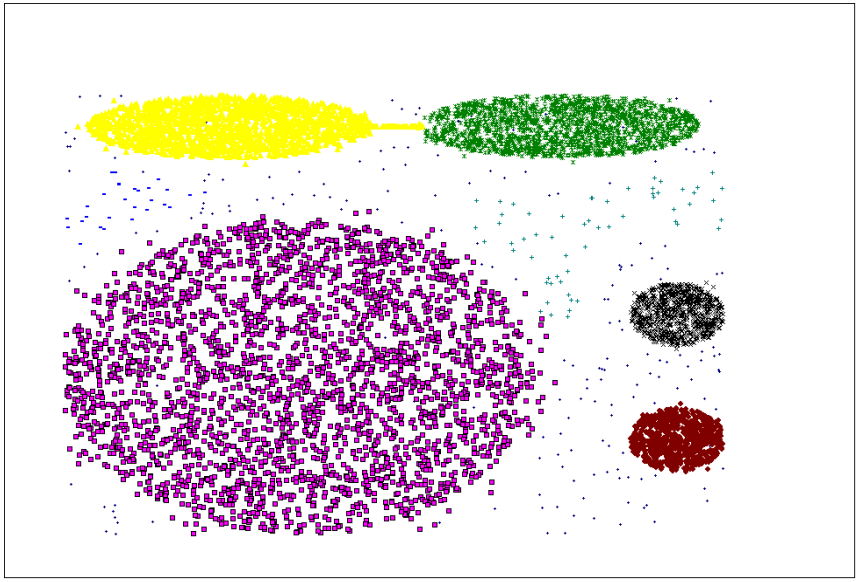
Original Points



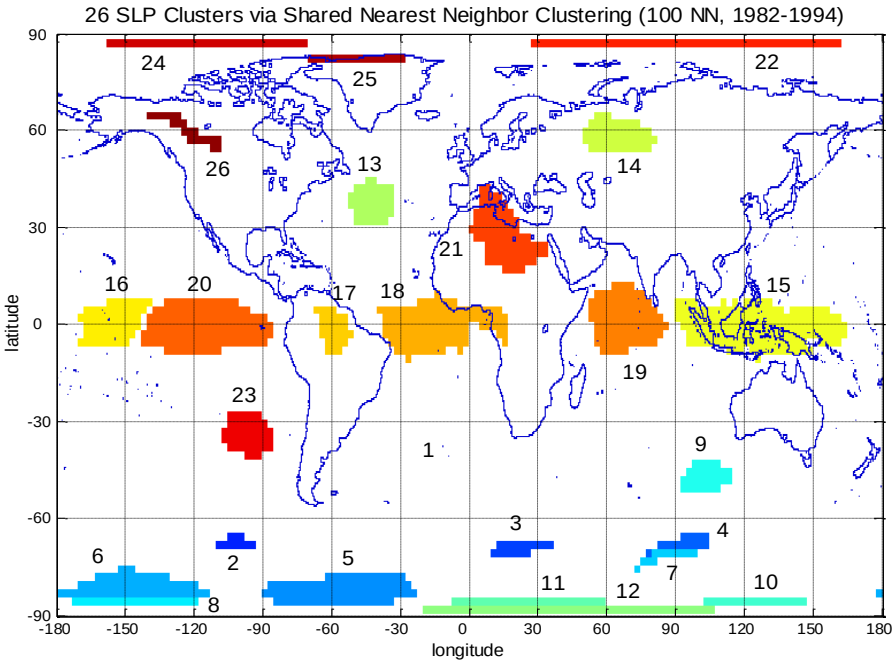
SNN Clustering

SNN Clustering Can Handle Other Difficult Situations

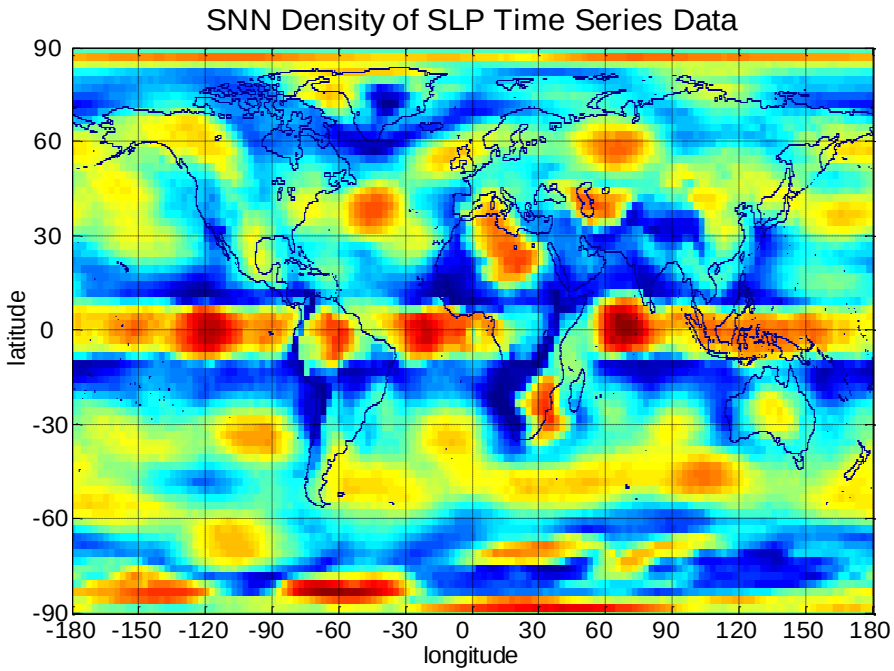




Finding Clusters of Time Series In Spatio-Temporal Data



SNN Clusters of SLP.



SNN Density of Points on the Globe.

Limitations of SNN Clustering

- Does not cluster all the points
- Complexity of SNN Clustering is high
  - O(n * time to find numbers of neighbor within *Eps*)
  - In worst case, this is $O(n^2)$
  - For lower dimensions, there are more efficient ways to find the nearest neighbors
    - R* Tree
    - k-d Trees
- Parameteriziation is not easy

Characteristics of Data, Clusters, and Clustering Algorithms

- A cluster analysis is affected by characteristics of
  - Data
  - Clusters
  - Clustering algorithms

- Looking at these characteristics gives us a number of dimensions that you can use to describe clustering algorithms and the results that they produce

Characteristics of Data

- High dimensionality
- Size of data set
- Sparsity of attribute values
- Noise and Outliers
- Types of attributes and type of data sets
- Differences in attribute scales
- Properties of the data space
  - Can you define a meaningful centroid

Characteristics of Clusters

- Data distribution
- Shape
- Differing sizes
- Differing densities
- Poor separation
- Relationship of clusters
- Types of clusters
  - Center-based, contiguity-based, density-based
- Subspace clusters

Characteristics of Clustering Algorithms

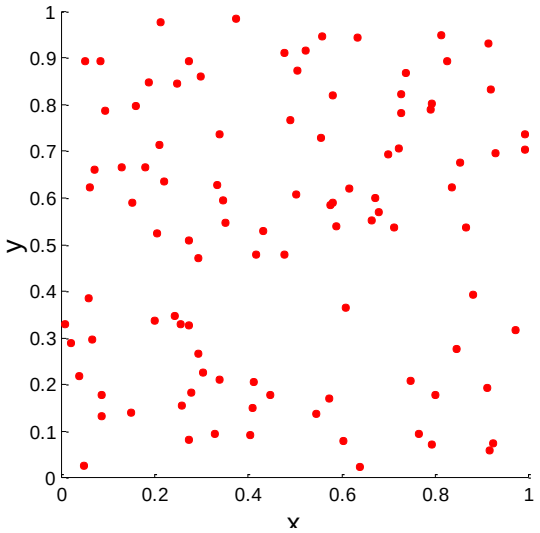
- Order dependence
- Non-determinism
- Parameter selection
- Scalability
- Underlying model
- Optimization based approach

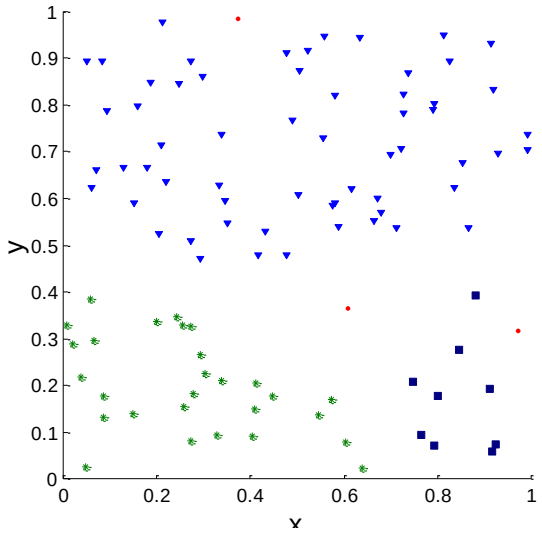
Cluster Validity

- For supervised classification we have a variety of measures to evaluate how good our model is
  - Accuracy, precision, recall
- For cluster analysis, the analogous question is how to evaluate the "goodness" of the resulting clusters?
- But "clusters are in the eye of the beholder"!
- Then why do we want to evaluate them?
  - To avoid finding patterns in noise
  - To compare clustering algorithms
  - To compare two sets of clusters
  - To compare two clusters

Clusters found in Random Data

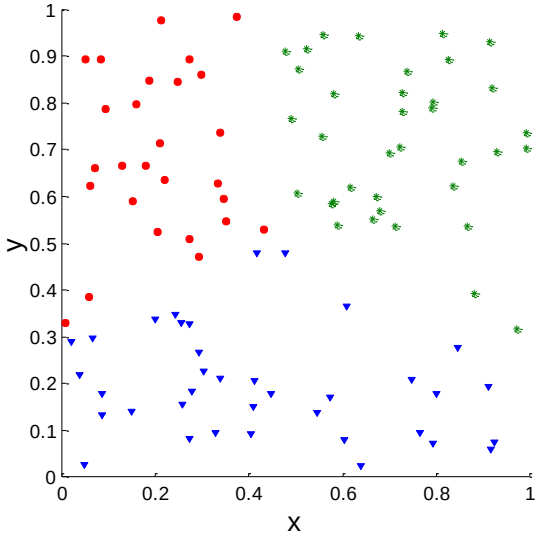
Random Points

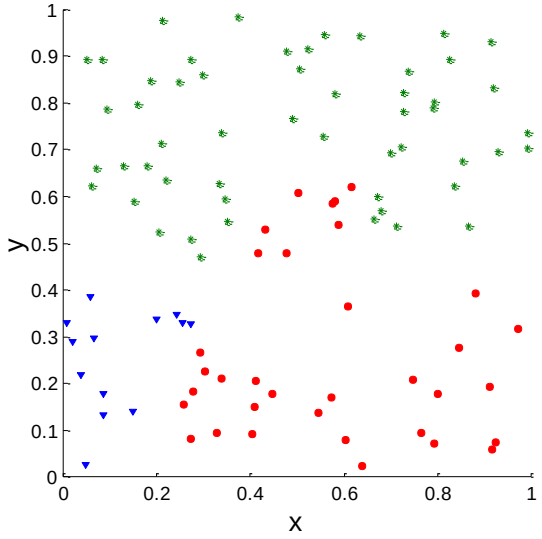




DBSCAN

K-means





Complete Link

Different Aspects of Cluster Validation

1. Determining the clustering tendency of a set of data, i.e., distinguishing whether non-random structure actually exists in the data.
2. Comparing the results of a cluster analysis to externally known results, e.g., to externally given class labels.
3. Evaluating how well the results of a cluster analysis fit the data *without* reference to external information.

   - Use only the data

4. Comparing the results of two different sets of cluster analyses to determine which is better.
5. Determining the 'correct' number of clusters.

For 2, 3, and 4, we can further distinguish whether we want to evaluate the entire clustering or just individual clusters.

Measures of Cluster Validity

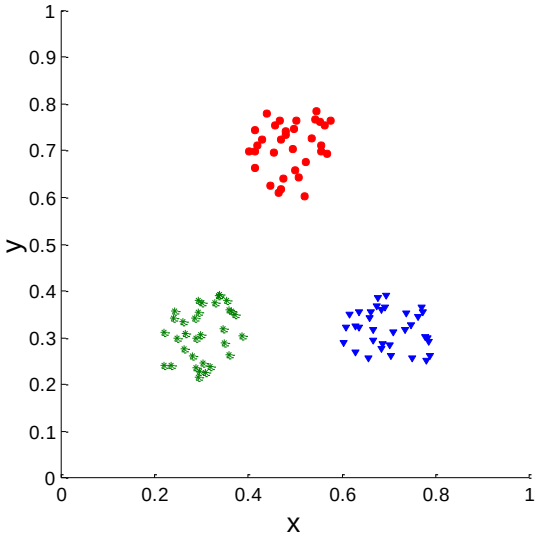
- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types.
  - External Index: Used to measure the extent to which cluster labels match externally supplied class labels.
    - Entropy
  - Internal Index: Used to measure the goodness of a clustering structure *without* respect to external information.
    - Sum of Squared Error (SSE)
  - Relative Index: Used to compare two different clusterings or clusters.
    - Often an external or internal index is used for this function, e.g., SSE or entropy
- Sometimes these are referred to as criteria instead of indices
  - However, sometimes criterion is the general strategy and index is the numerical measure that implements the criterion.

Measuring Cluster Validity Via Correlation

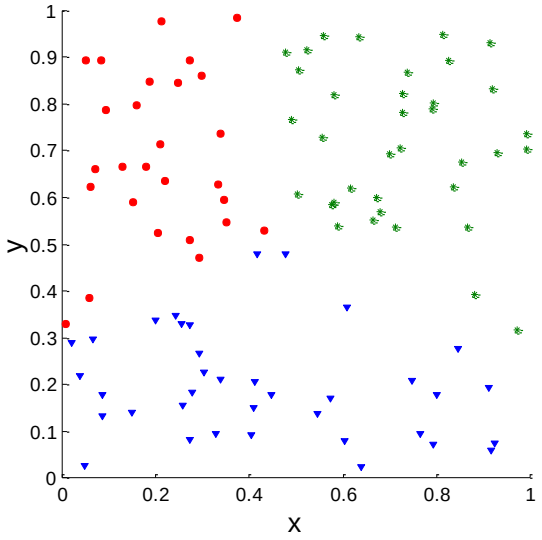
- Two matrices
  - Proximity Matrix
  - Ideal Similarity Matrix
    - One row and one column for each data point
    - An entry is 1 if the associated pair of points belong to the same cluster
    - An entry is 0 if the associated pair of points belongs to different clusters
- Compute the correlation between the two matrices
  - Since the matrices are symmetric, only the correlation between n(n-1) / 2 entries needs to be calculated.
- High correlation indicates that points that belong to the same cluster are close to each other.
- Not a good measure for some density or contiguity based clusters.

Measuring Cluster Validity Via Correlation

- Correlation of ideal similarity and proximity matrices for the K-means clusterings of the following two data sets.



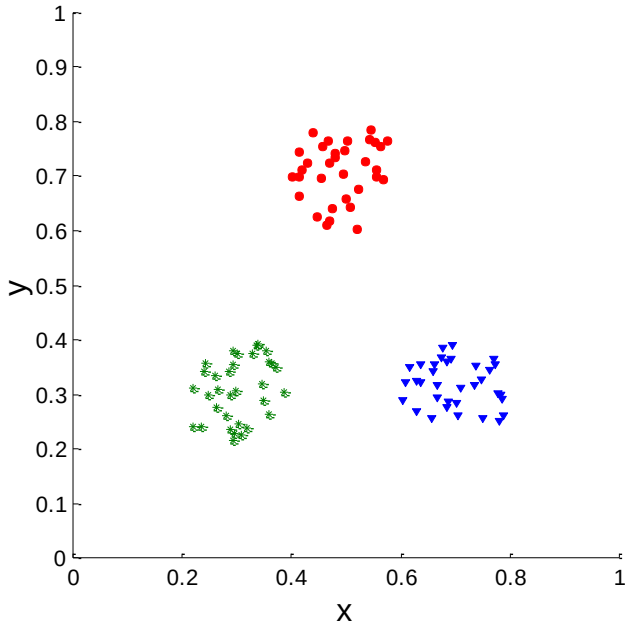
Corr = -0.9235

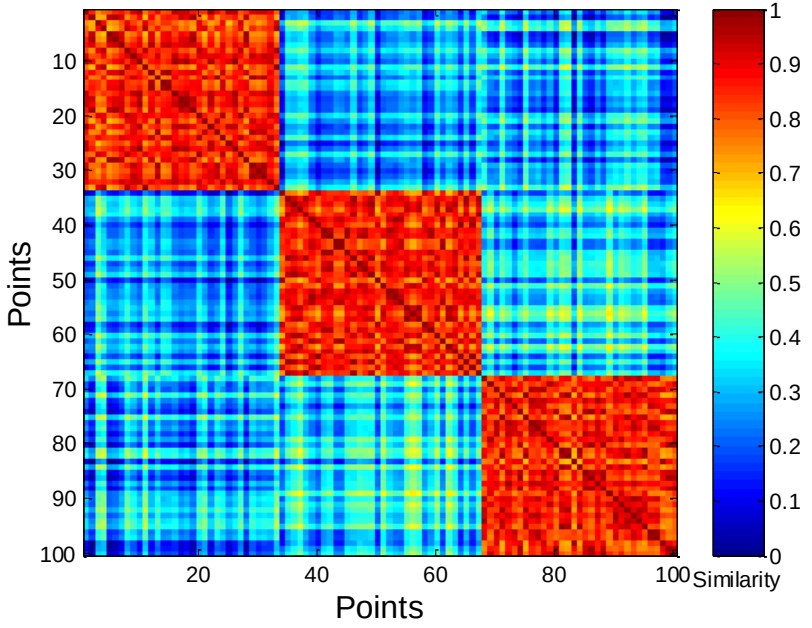


Corr = -0.5810

Using Similarity Matrix for Cluster Validation

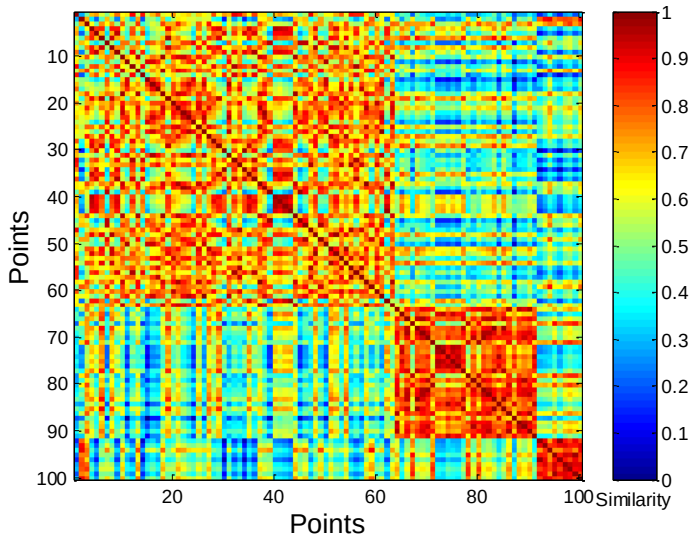
- Order the similarity matrix with respect to cluster labels and inspect visually.

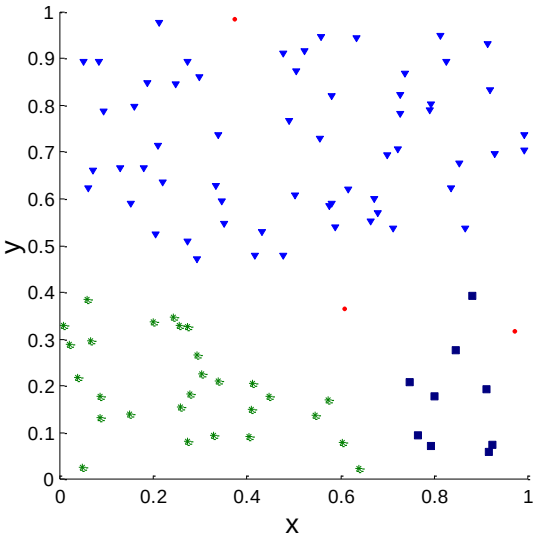




Using Similarity Matrix for Cluster Validation

- Clusters in random data are not so crisp

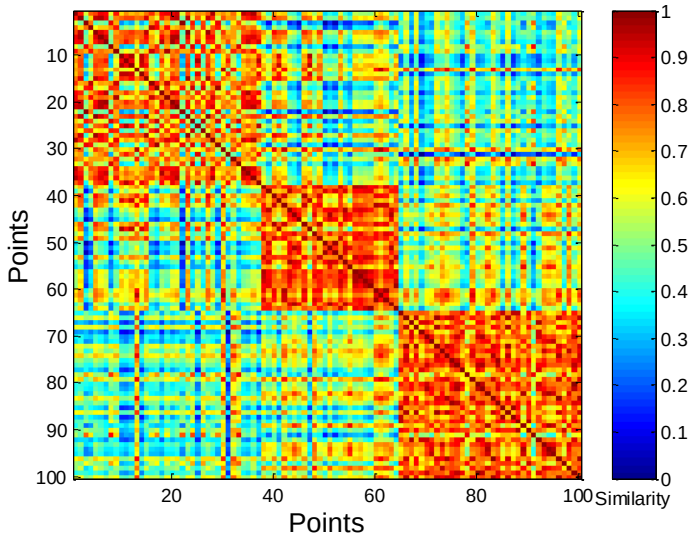


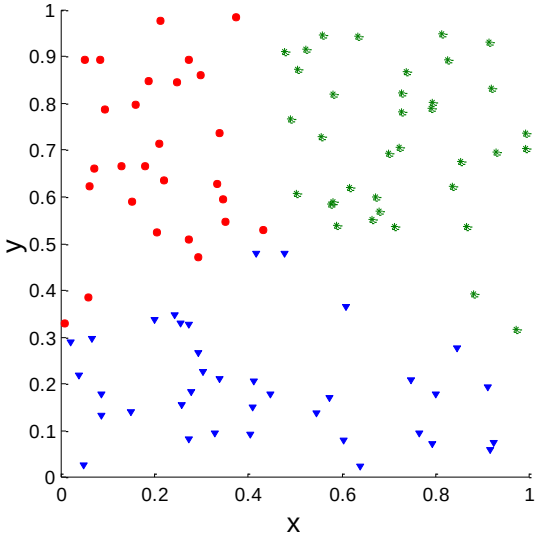


DBSCAN

Using Similarity Matrix for Cluster Validation

- Clusters in random data are not so crisp

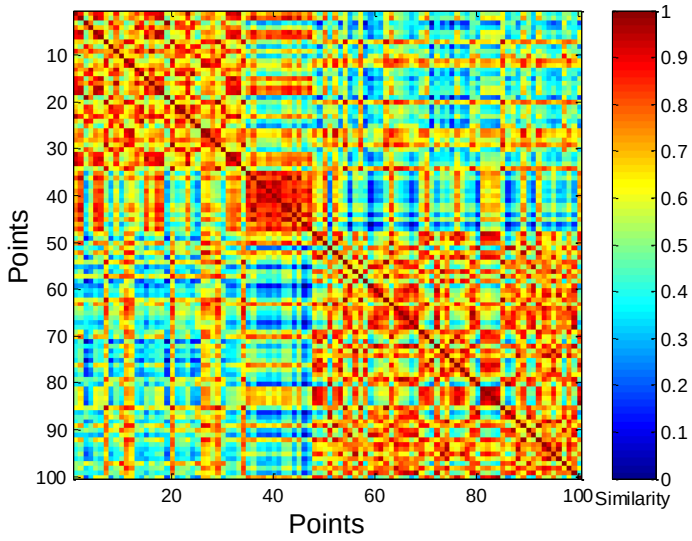


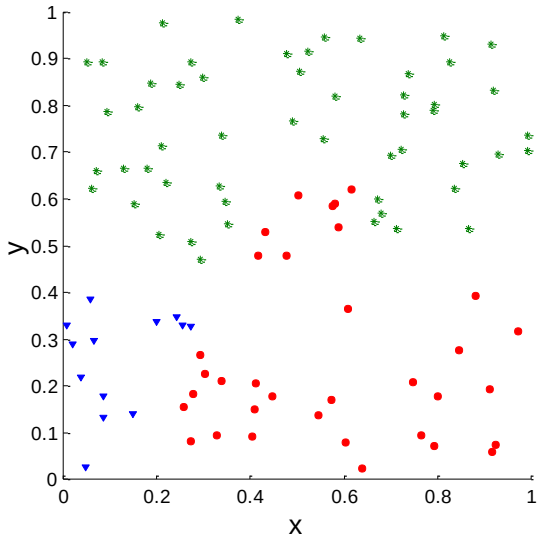


K-means

Using Similarity Matrix for Cluster Validation

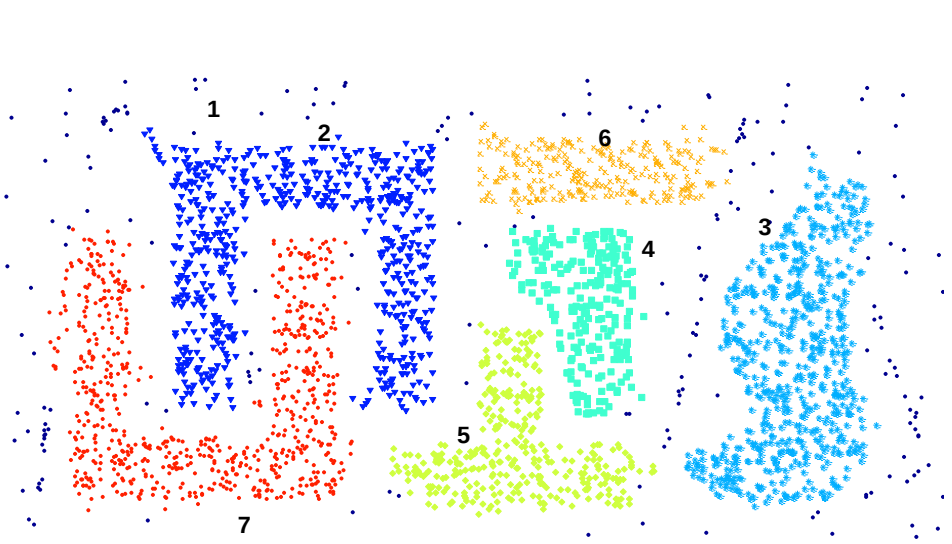
- Clusters in random data are not so crisp

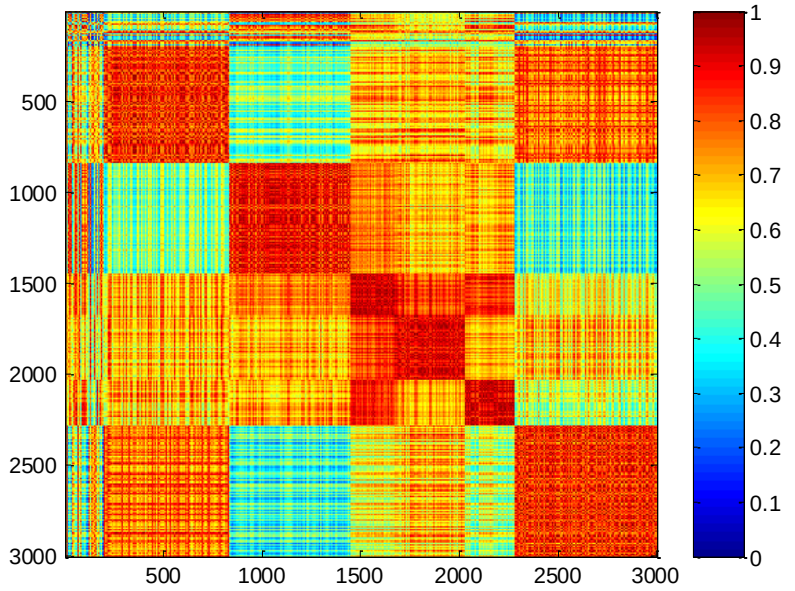




Complete Link

Using Similarity Matrix for Cluster Validation

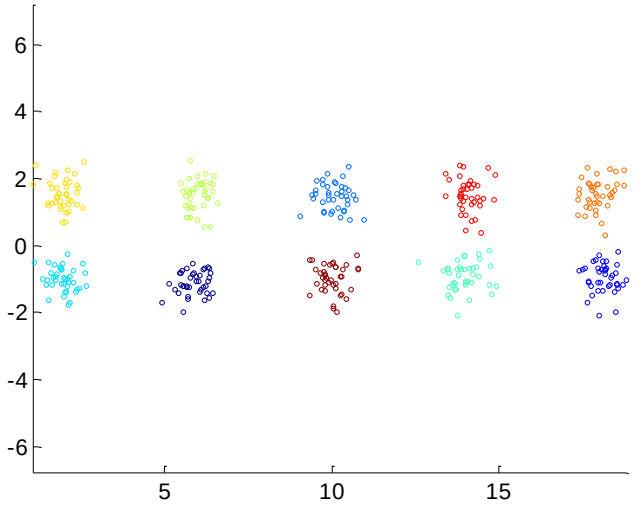


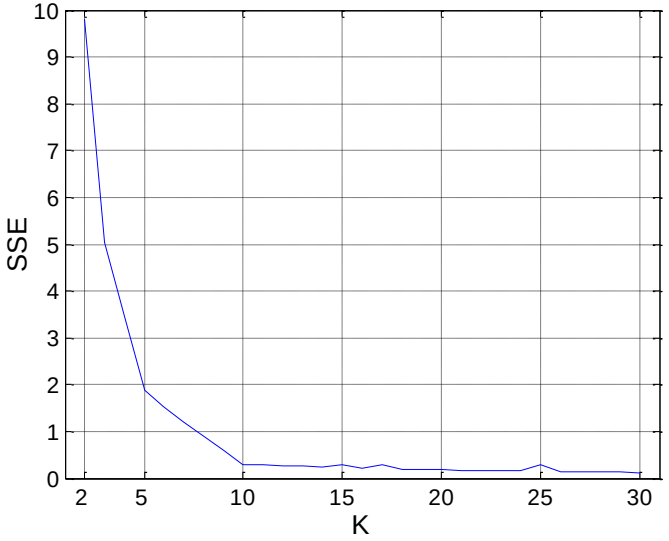


DBSCAN

Internal Measures: SSE

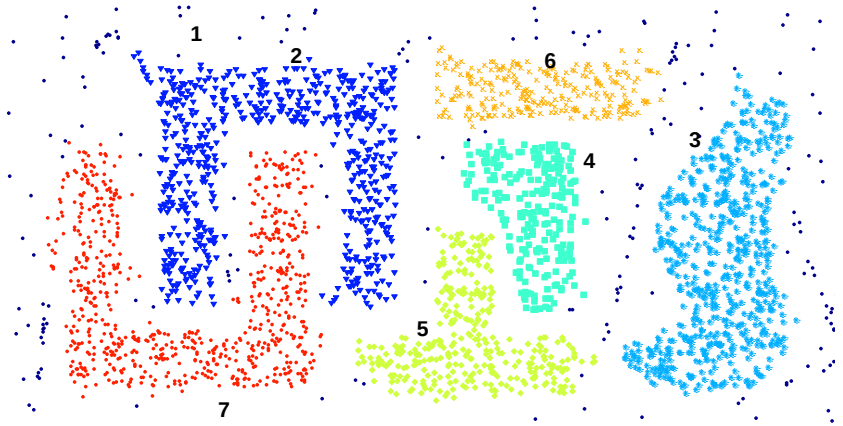
- Clusters in more complicated figures aren't well separated
- Internal Index: Used to measure the goodness of a clustering structure without respect to external information
  - SSE
- SSE is good for comparing two clusterings or two clusters (average SSE).
- Can also be used to estimate the number of clusters

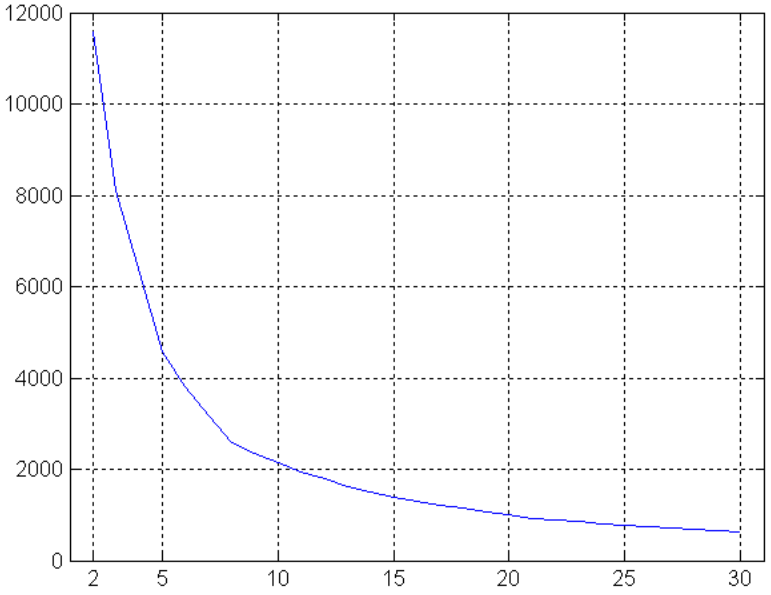




Internal Measures: SSE

- SSE curve for a more complicated data set





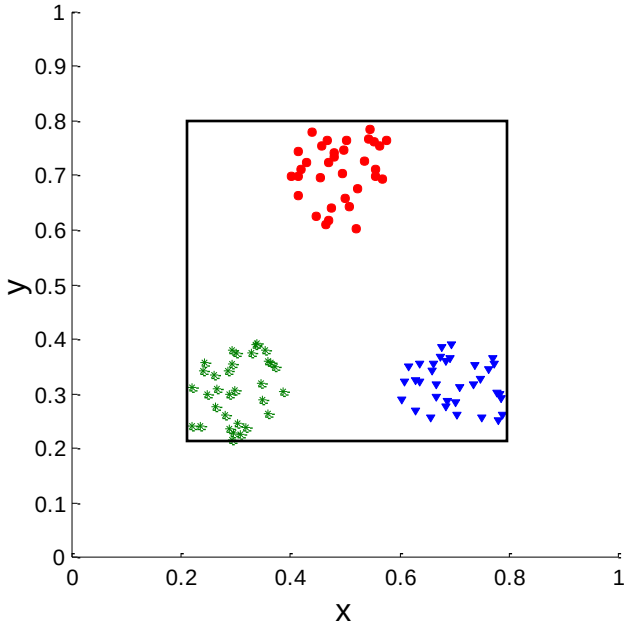
SSE of clusters found using K-means

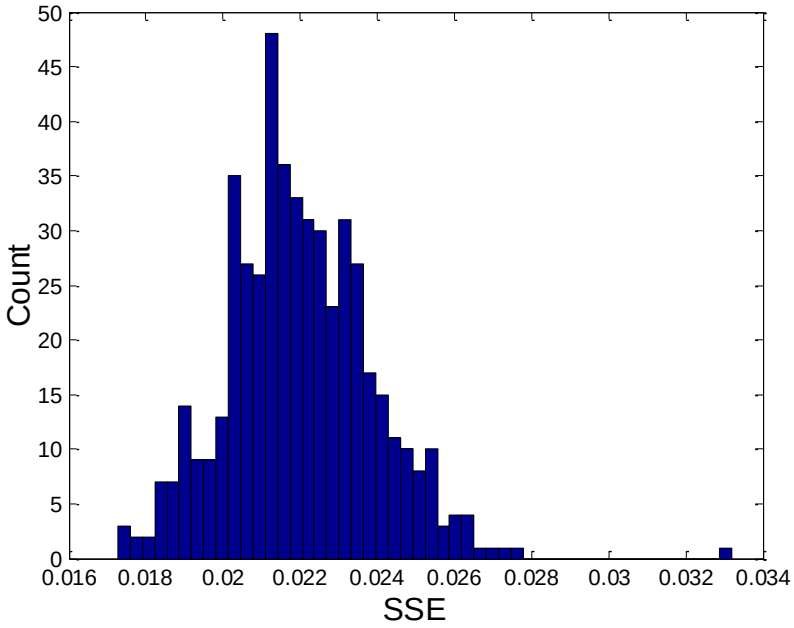
Framework for Cluster Validity

- Need a framework to interpret any measure.
  - For example, if our measure of evaluation has the value, 10, is that good, fair, or poor?
- Statistics provide a framework for cluster validity
  - The more “atypical” a clustering result is, the more likely it represents valid structure in the data
  - Can compare the values of an index that result from random data or clusterings to those of a clustering result.
    - If the value of the index is unlikely, then the cluster results are valid
  - These approaches are more complicated and harder to understand.
- For comparing the results of two different sets of cluster analyses, a framework is less necessary.
  - However, there is the question of whether the difference between two index values is significant

Statistical Framework for SSE

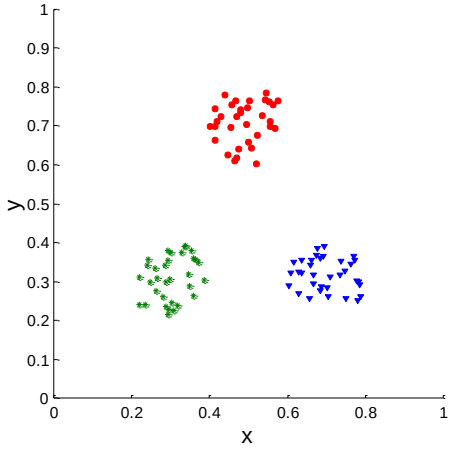
- Example
  - Compare SSE of 0.005 against three clusters in random data
  - Histogram shows SSE of three clusters in 500 sets of random data points of size 100 distributed over the range 0.2 – 0.8 for x and y values

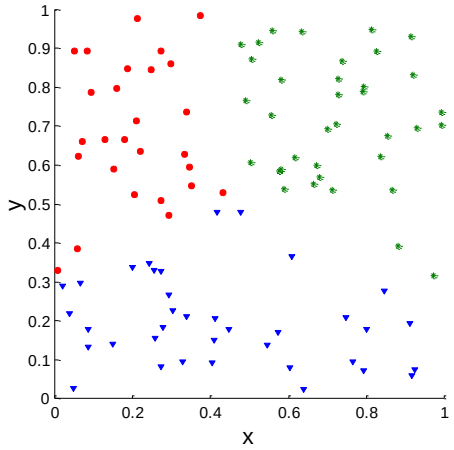


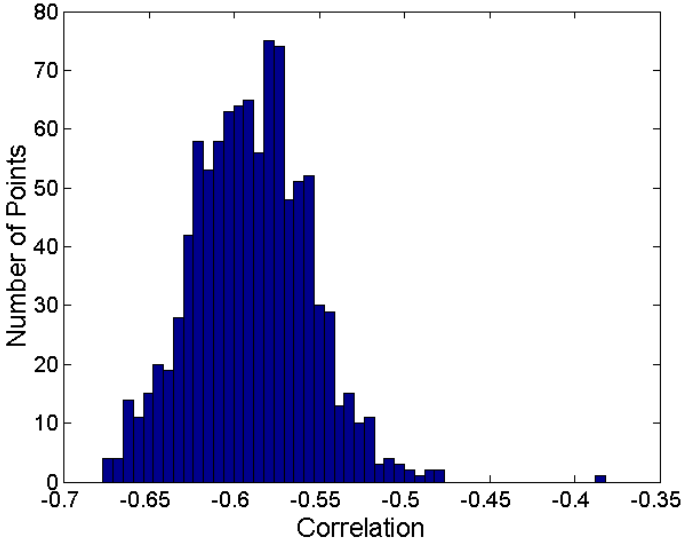


Statistical Framework for Correlation

- Correlation of ideal similarity and proximity matrices for the K-means clusterings of the following two data sets.







Corr = -0.9235

Corr = -0.5810

Internal Measures: Cohesion and Separation

- Cluster Cohesion: Measures how closely related are objects in a cluster
  - Example: SSE
- Cluster Separation: Measure how distinct or well-separated a cluster is from other clusters
- Example: Squared Error
  - Cohesion is measured by the within cluster sum of squares (SSE)

$$SSE = WSS = \sum_{i} \sum_{x \in C_i} (x - m_i)^2$$

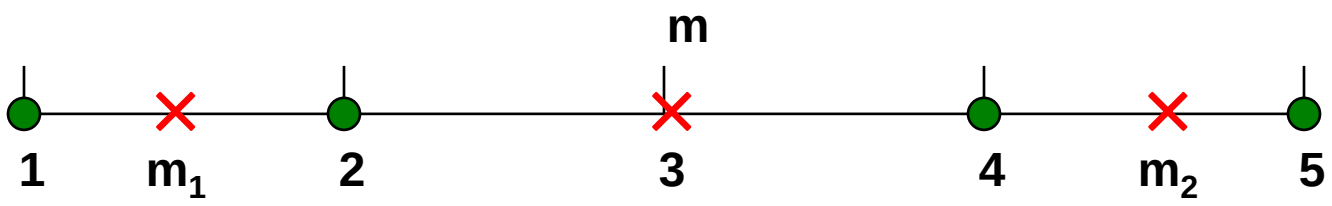
  - Separation is measured by the between cluster sum of squares

$$BSS = \sum_{i} |C_i| (m - m_i)^2$$

    - Where $|C_i|$ is the size of cluster i

Internal Measures: Cohesion and Separation

- Example: SSE
  - BSS + WSS = constant



K=1 cluster:

$$SSE = WSS = (1-3)^2 + (2-3)^2 + (4-3)^2 + (5-3)^2 = 10$$

$$BSS = 4 \times (3-3)^2 = 0$$

$$Total = 10 + 0 = 10$$

K=2 clusters:

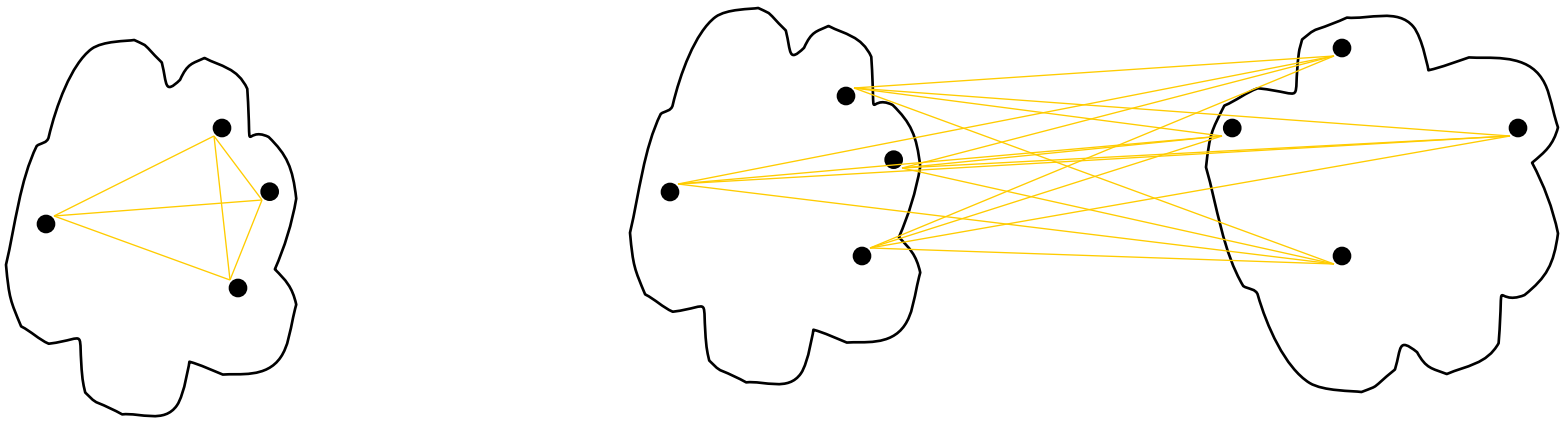
$$SSE = WSS = (1-1.5)^2 + (2-1.5)^2 + (4-4.5)^2 + (5-4.5)^2 = 1$$

$$BSS = 2 \times (3-1.5)^2 + 2 \times (4.5-3)^2 = 9$$

$$Total = 1 + 9 = 10$$

Internal Measures: Cohesion and Separation

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



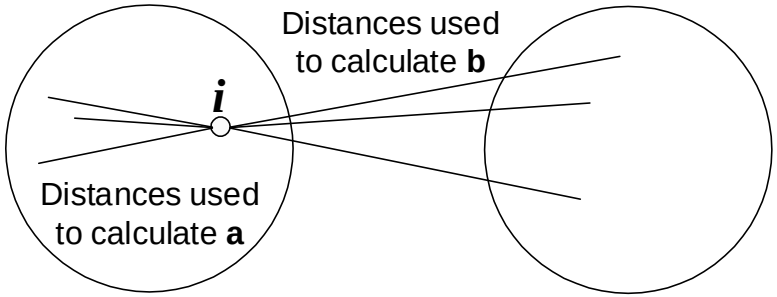
cohesion separation

Internal Measures: Silhouette Coefficient

- Silhouette coefficient combines ideas of both cohesion and separation, but for individual points, as well as clusters and clusterings
- For an individual point, ***i***
  - Calculate ***a*** = average distance of ***i*** to the points in its cluster
  - Calculate ***b*** = min (average distance of ***i*** to points in another cluster)
  - The silhouette coefficient for a point is then given by

    s = (b – a) / max(a,b)

  - Typically between 0 and 1.
  - The closer to 1 the better.



- Can calculate the average silhouette coefficient for a cluster or a clustering

External Measures of Cluster Validity: Entropy and Purity

Table 5.9. K-means Clustering Results for LA Document Data Set

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Entropy	Purity
1	3	5	40	506	96	27	1.2270	0.7474
2	4	7	280	29	39	2	1.1472	0.7756
3	1	1	1	7	4	671	0.1813	0.9796
4	10	162	3	119	73	2	1.7487	0.4390
5	331	22	5	70	13	23	1.3976	0.7134
6	5	358	12	212	48	13	1.5523	0.5525
Total	354	555	341	943	273	738	1.1450	0.7203

entropy For each cluster, the class distribution of the data is calculated first, i.e., for cluster j we compute p_{ij}, the 'probability' that a member of cluster j belongs to class i as follows: $p_{ij} = m_{ij}/m_j$, where m_j is the number of values in cluster j and m_{ij} is the number of values of class i in cluster j. Then using this class distribution, the entropy of each cluster j is calculated using the standard formula $e_j = \sum_{i=1}^{L} p_{ij} \log_2 p_{ij}$, where the L is the number of classes. The total entropy for a set of clusters is calculated as the sum of the entropies of each cluster weighted by the size of each cluster, i.e., $e = \sum_{i=1}^{K} \frac{m_i}{m} e_j$, where m_j is the size of cluster j, K is the number of clusters, and m is the total number of data points.

purity Using the terminology derived for entropy, the purity of cluster j, is given by $purity_j = \max p_{ij}$ and the overall purity of a clustering by $purity = \sum_{i=1}^{K} \frac{m_i}{m} purity_j$.

Final Comment on Cluster Validity

"The validation of clustering structures is the most difficult and frustrating part of cluster analysis.

Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage."

Algorithms for Clustering Data, Jain and Dubes