Clustering

Yanyan Lan

lanyanyan@ict.ac.cn

Review

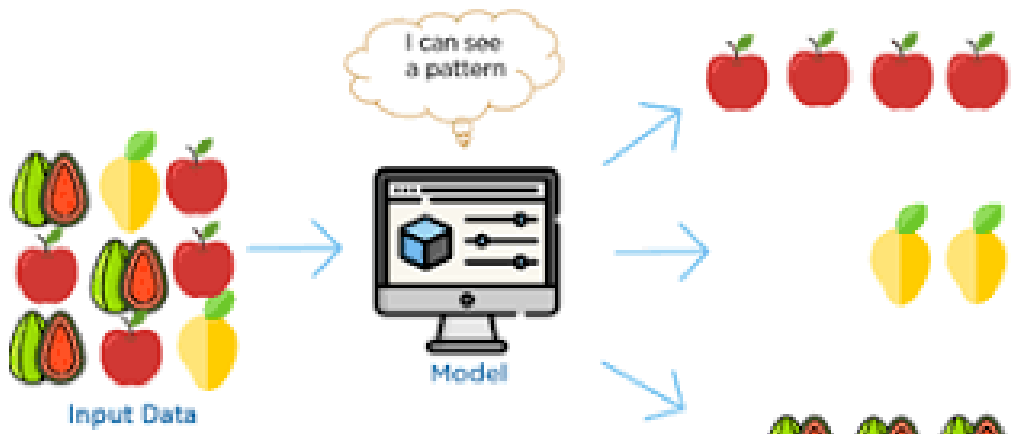
- Introduction (2 lessons)
- Model Selection (2 lessons)
- Supervised Learning (12 lessons):
  - Linear Model(2 lessons)
  - Naive Bayes (2 lessons)
  - Decision Tree (2 lessons)
  - Support Vector Machine (2 lessons)
  - Neural Network (4 lessons)
- Ensemble Learning (4 lessons)
  - Boosting
  - Bagging , Random Forest

Supervised vs. Unsupervised Learning

- Supervised learning: given $\{x^i, y^i\}_{i=1}^{\mathrm{N}}$, Learn $\hat{y} = f(x, y)$
  - classification: y is categorical
  - regression: y is continuous
  - ranking: y is ordinal
- Unsupervised learning: given $\{x^i\}_{i=1}^{N}$, learn $\widehat{y} = f(x; w)$
  - Density estimation: y is density
  - Clustering: y is clusters
  - Dimensionality reduction/visualization: y is lower-dimensional representations of x

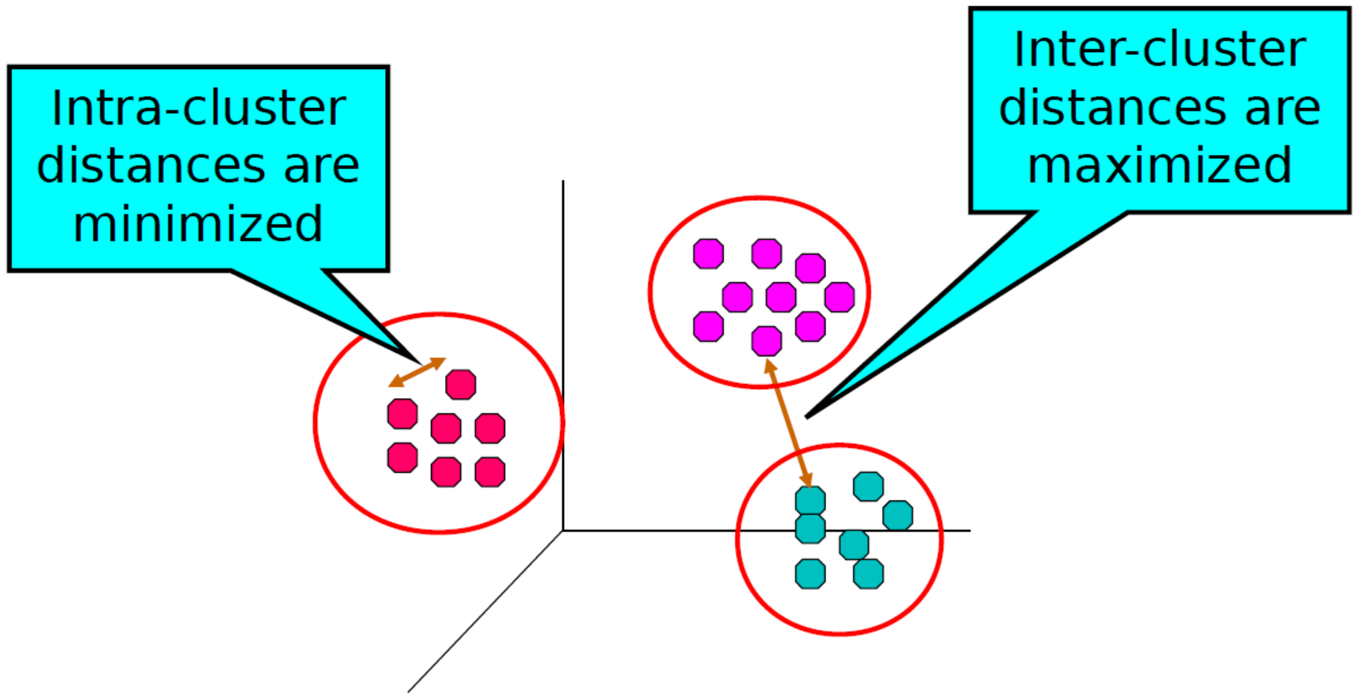
Why do Unsupervised Learning?

- Raw data cheap. Labeled data expensive.
- Save memory/computation.
- Reduce noise in high-dimensional data.
- Useful in exploratory data analysis.
- Often a pre-processing step for supervised learning.



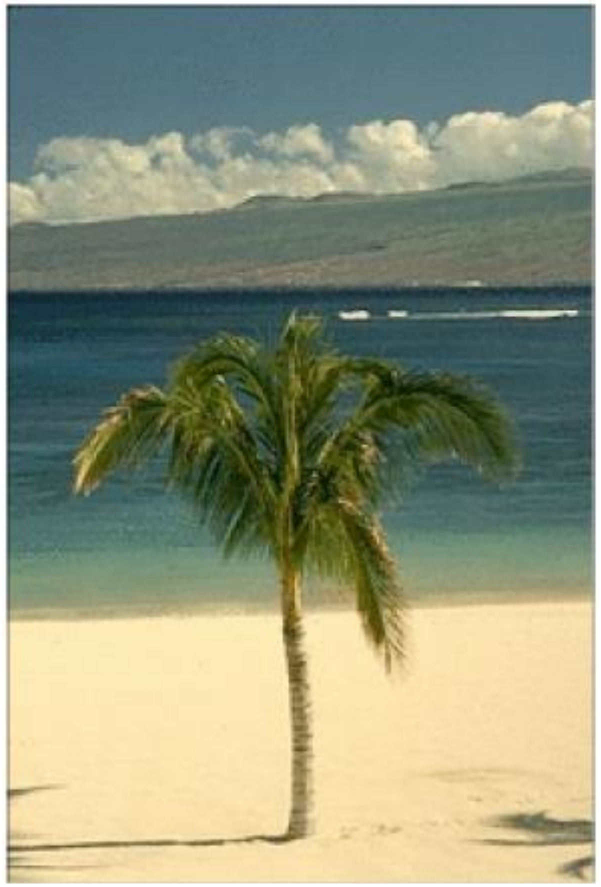
Clustering: Problem Formulation

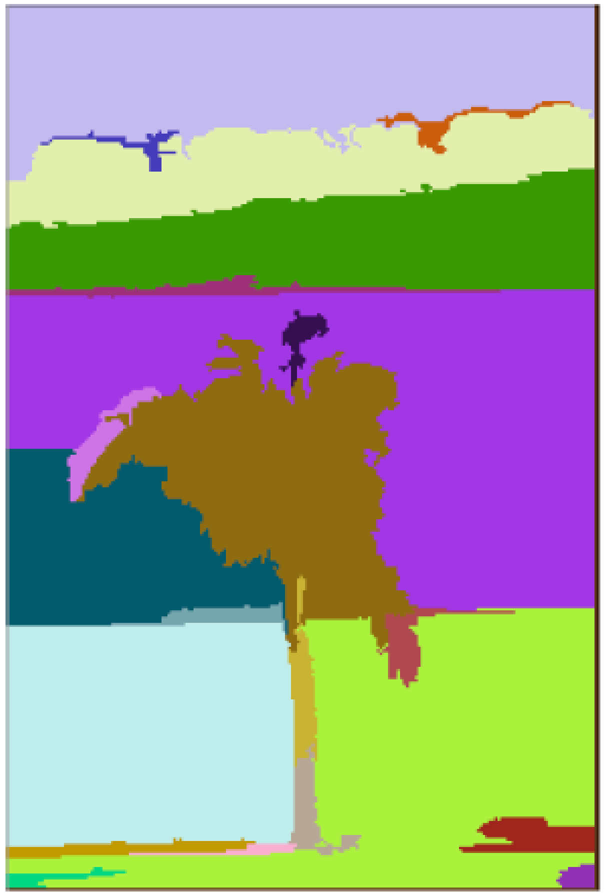
- Discover groups such that samples within a group are more similar to each other than samples across groups.


Intra-cluster distances are minimized
Inter-cluster distances are maximized

Applications of Clustering

Image Segmentation

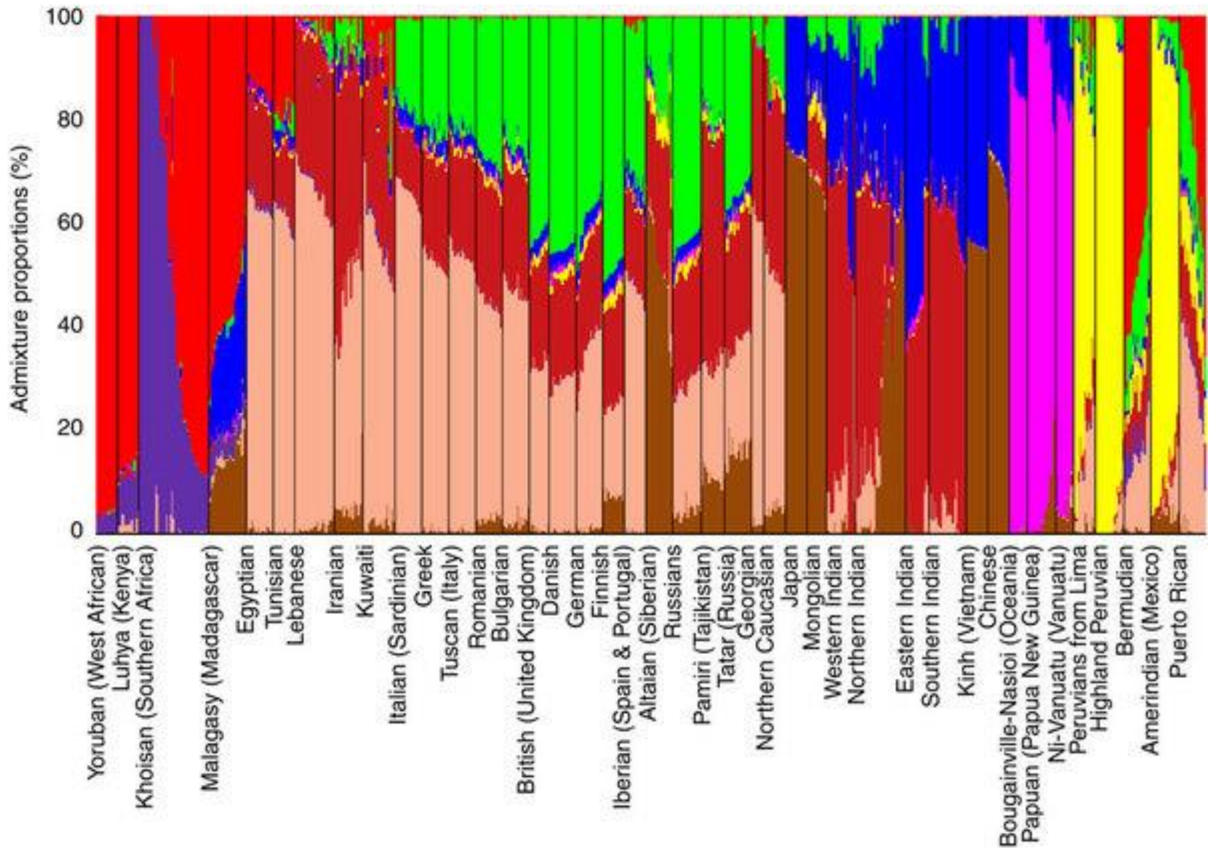


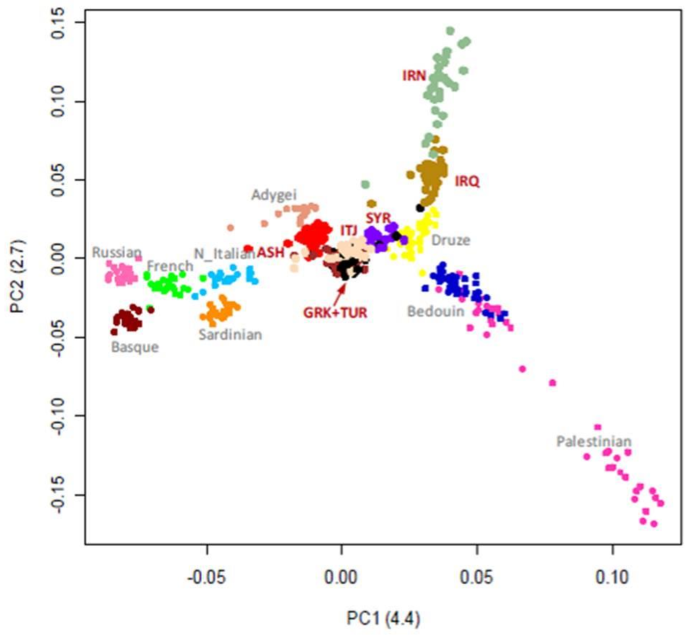


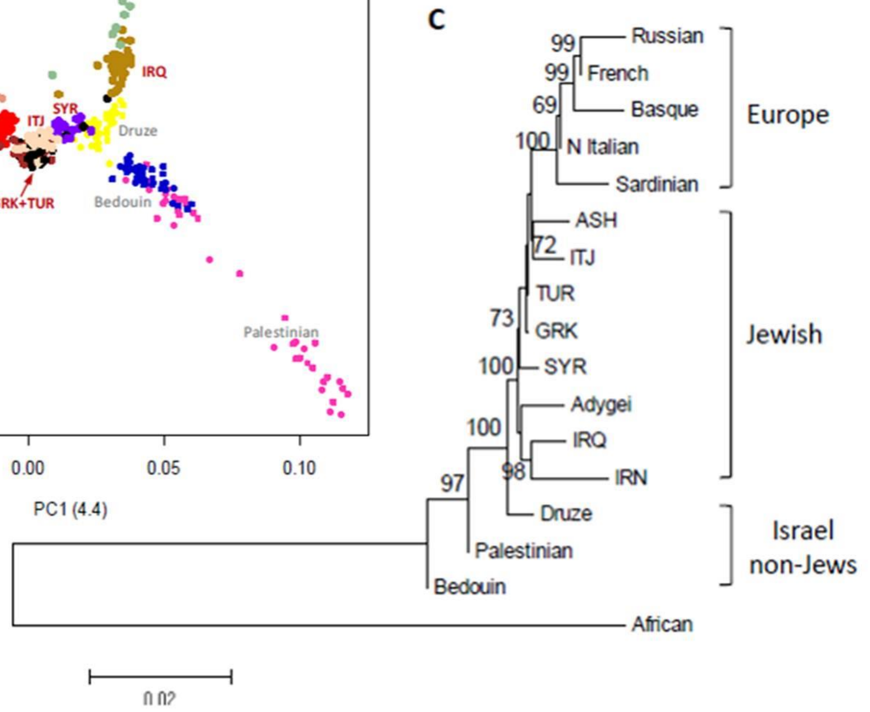
http://people.cs.uchicago.edu/ pff/segment

Applications of Clustering

Human Population



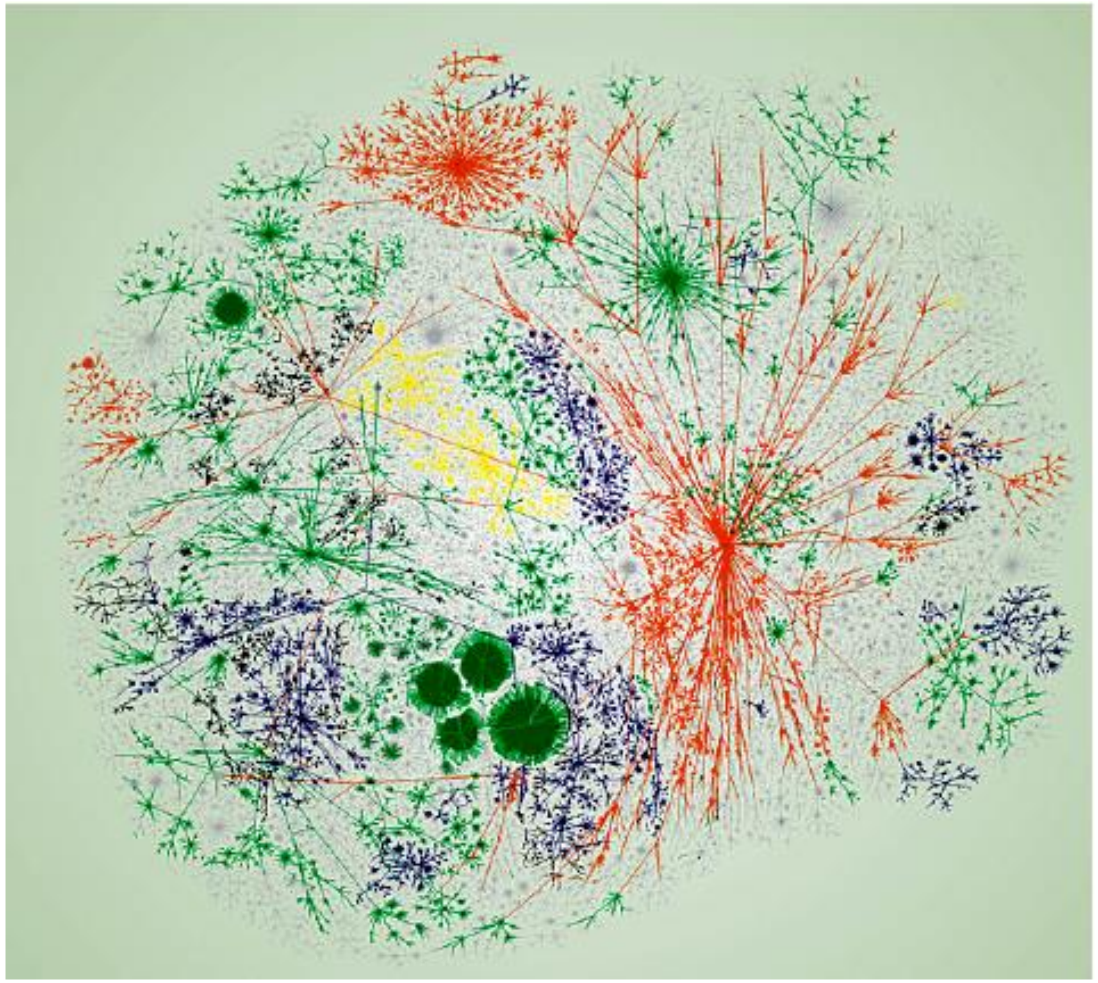




Eran Elhaik et al. Nature

Applications of Clustering

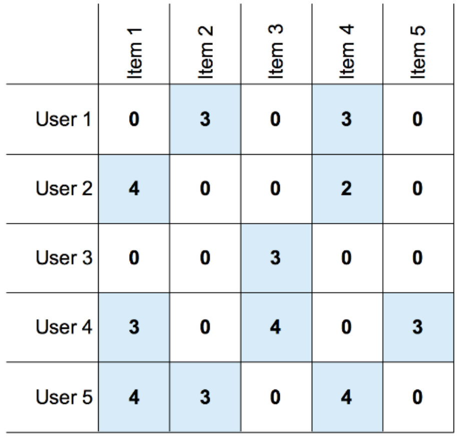
Clustering Graphs



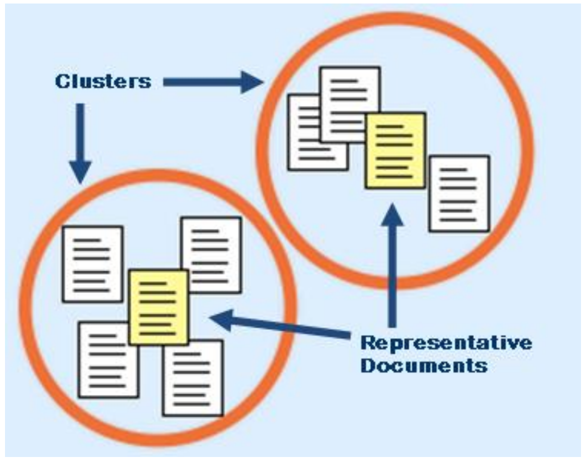
Newman, 2008

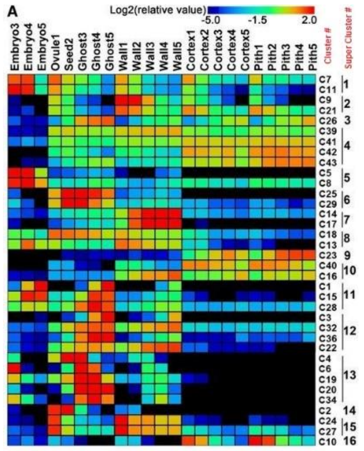
Other Applications

- Cluster customers based on their purchase histories
- Cluster products based on the sets of customers who purchased them
- Cluster documents based on similar words or shingles
- Cluster DNA sequences based on edit distance



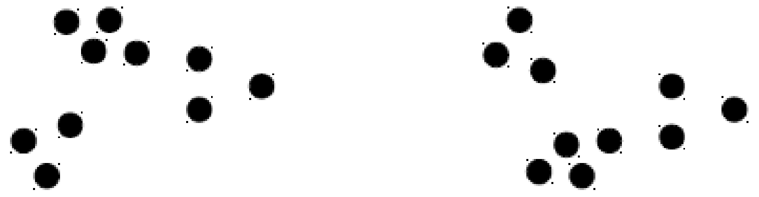
	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	0	3	0	3	0
User 2	4	0	0	2	0
User 3	0	0	3	0	0
User 4	3	0	4	0	3
User 5	4	3	0	4	0





Evaluation Metrics

Notion of a Cluster can be Ambiguous



How many clusters?

Evaluation of Clustering

- Performance Evaluation of Clustering: Validity index
- Evaluation metrics:
  - Reference model (external index)

  ✓compare with reference
  - Non-reference model (internal index)

  ✓measure distance of inner-class and inter-class

Reference Model

- Dataset : $D = \{x_1, x_2, \dots x_m\}$
- Clusters of clustering: $C = \{C_1, C_2, \dots C_m\}$
- Clusters of reference model: $C^* = \{C_1, C_2, \dots C_m\}$
- λ and λ^* the clusters' label of C and C^* respectively
- Sample pair: $(x_i, y_i), i \leq i < j \leq m$

m(m-1)/2		reference	
		same	not
clustering	same	a	b
	not	c	d

a ↑, consistence ↑
b ↑, consistence ↓
c ↑, consistence ↓
d ↑, consistence ↑

- Naturally, we can define external index by a, b, c, d

External Index

- Jaccard coefficient, JC

$$JC = \frac{a}{a+b+c} \qquad JC \in [0,1], \quad JC \uparrow, consistence \uparrow$$

- Fowlkes and Mallows index, FMI

$$FMI = \sqrt{\frac{a}{a+b}\frac{a}{a+c}} \qquad FMI \in [0,1], \quad FMI \uparrow, consistence \uparrow$$

- Rand Index, RI

$$RI = \frac{2(a+d)}{m(m-1)} \qquad RI \in [0,1], \quad RI \uparrow, consistence \uparrow$$

Non-reference model

- Only having result of clustering, how can we evaluate it ?
  - Intra-cluster similarity: larger is better
  - Inter-cluster similarity: smaller is better

Non-reference model

- Intra-cluster similarity
  - Average distance

$$avg(C) = \frac{2}{|C|(|C|-1)} \sum_{1\le i\le j\le |C|} dist(x_i, x_j)$$

  - Maximal distance

$$diam(c) = \max_{1\le i\le j\le |C|} 1 \le i \le j \le |C|$$

- Inter-cluster similarity
  - Minimal distance

$$d_{min}(c_i, c_j) = \min_{x_i\in C_i, x_j\in C_j} dist(\mu_i, \mu_j)$$

  - Distance of centers

$$d_{cen}(c_i, c_j) = dist(\mu_i, \mu_j), where\ \mu_i = \frac{1}{|C_i|} \sum\nolimits_{x_i\in C_i} x_i$$

Clustering Methods

- K-Means
- Hierarchical Clustering
- Gaussian Mixture Models
- Density Based Methods

K-means

K-means: Basic Idea

- Given a sample set $X = \{x^{(i)}\}^N$
- Each cluster is associated with a centroid (or prototype) μ_k
- Each point is assigned to the cluster with the closest centroid
  - '**Closeness**' is measured by Euclidean distance
  - Assignment of data x_i to a cluster k represented by Responsibilities $r_{ik} \in \{0,1\}$ with $\sum_{k=1}^{K} r_{ik} = 1$
- Number of clusters, K, must be specified

K-means: minimizing the loss function

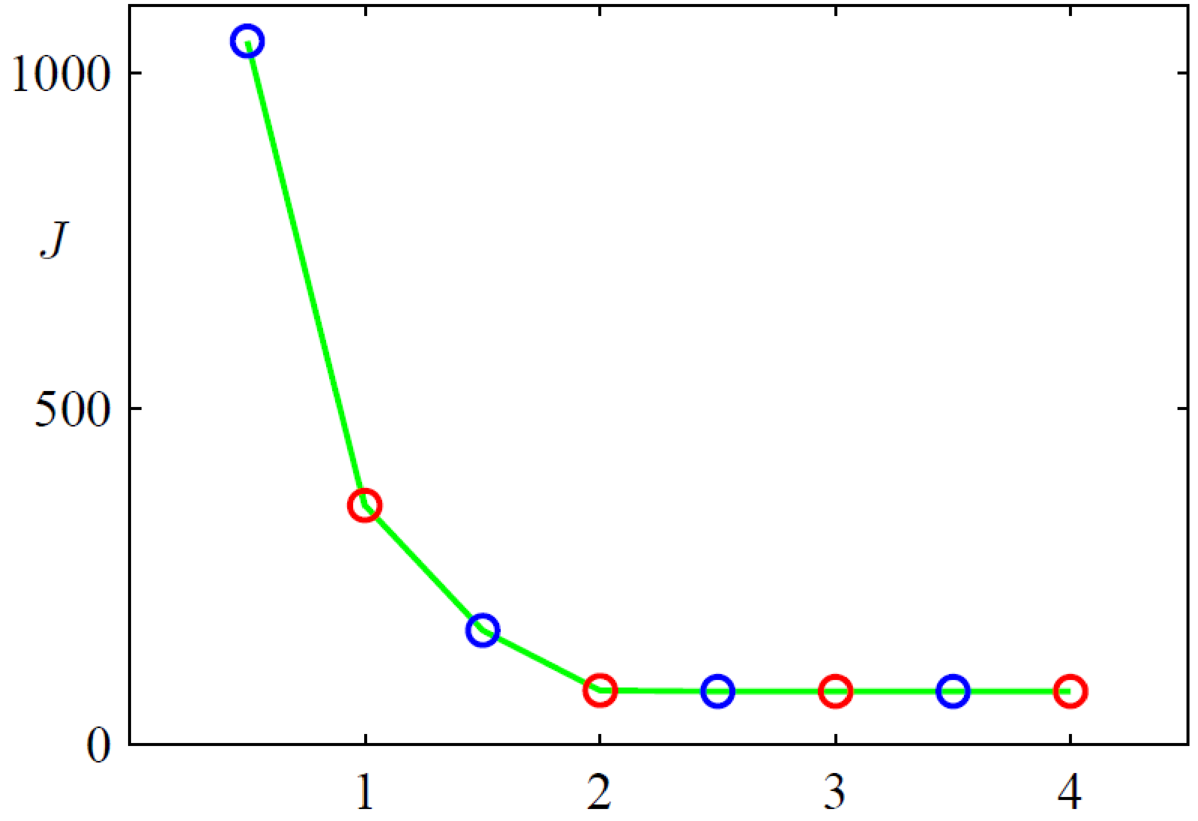
- Loss function $J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} ||x_i - \mu_k||^2$, Sum of Squared Error (SSE)
- How do we minimize J w.r.t (r_{ik},u_k)?
- Chicken and egg problem
    - If prototypes known, can assign responsibilities
    - If responsibilities known, can compute prototypes
- We use an iterative procedure

K-means Algorithms

- E-step: fix μ_k, minimize J w.r.t. r_{ik}
  - Assign each data point to its nearest prototype
- M-step: Fix r_{ik} ,minimize J w.r.t. μ_k
  - Set each prototype to the mean of the points in that cluster

    i.e., $\mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}$
- This procedure is guaranteed to converge

Loss function J after each iteration



Convergence of K-means

- k-means is exactly coordinate descent on the reconstruction error J.
- J monotonically decreases, and the value of J converges, so do the clustering results.
- It is possible for k-means to oscillate between a few different clusterings, but this almost never happens in practice.
- J is non-convex, so coordinate descent on J cannot guaranteed to converge to global minimum. One common thing to do is running k-means many times and pick the best one.

Example: Watermelon Dataset

How do you cluster watermelons according to the dataset?

id	density	sugar content	id	density	sugar content
1	.697	.460	16	.593	.042
2	.774	.376	17	.719	.103
3	.634	.264	18	.359	.188
4	.608	.318	19	.339	.241
5	.556	.215	20	.282	.257
6	.403	.237	21	.748	.232
7	.481	.149	22	.714	.346
8	.437	.211	23	.483	.312
9	.666	.091	24	.478	.437
10	.243	.267	25	.525	.369
11	.245	.057	26	.751	.489
12	.343	.099	27	.532	.472
13	.639	.161	28	.473	.376
14	.657	.198	29	.725	.445
15	.360	.370	30	.446	.459

Example: Watermelon Dataset

- k = 3, three clusters C_1, C_2, C_3
- Use x_6, x_{12}, x_{24} as initial centroid vector
- $\mu_1 = (0.403; 0.237)$, $\mu_2 = (0.343; 0.099)$, $\mu_3 = (0.478; 0.437)$
- For x_1, the distance between μ_1, μ_2, μ_3 is 0.369, 0.506, 0.220. Hence x_1 belongs to the cluster C_3

id	density	sugar content	id	density	sugar content
1	.697	.460	16	.593	.042
2	.774	.376	17	.719	.103
3	.634	.264	18	.359	.188
4	.608	.318	19	.339	.241
5	.556	.215	20	.282	.257
6	.403	.237	21	.748	.232
7	.481	.149	22	.714	.346
8	.437	.211	23	.483	.312
9	.666	.091	24	.478	.437
10	.243	.267	25	.525	.369
11	.245	.057	26	.751	.489
12	.343	.099	27	.532	.472
13	.639	.161	28	.473	.376
14	.657	.198	29	.725	.445
15	.360	.370	30	.446	.459

Example: Watermelon Dataset

- After first iteration:

$C_1 = \{x_3, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{13}, x_{14}, x_{17},$

$x_{18}, x_{19}, x_{20}, x_{23}\}$

$C_2 = \{x_{11}, x_{12}, x_{16}\}$

$C_3 = \{x_1, x_2, x_4, x_{15}, x_{21}, x_{22}, x_{24}, x_{25}, x_{26},$

$x_{27},\ x_{28}, x_{29}, x_{30}\}$

- New centroid vector

$$\mu_1' = \frac{1}{|C_1|} \sum_{x \in C_1} x_i = (0.493; 0.207)$$

$$\mu_2' = \frac{1}{|C_2|} \sum_{x \in C_2} x_i = (0.394; 0.066)$$

$$\mu_3' = \frac{1}{|C_3|} \sum_{x \in C_3} x_i = (0.602; 0.396)$$

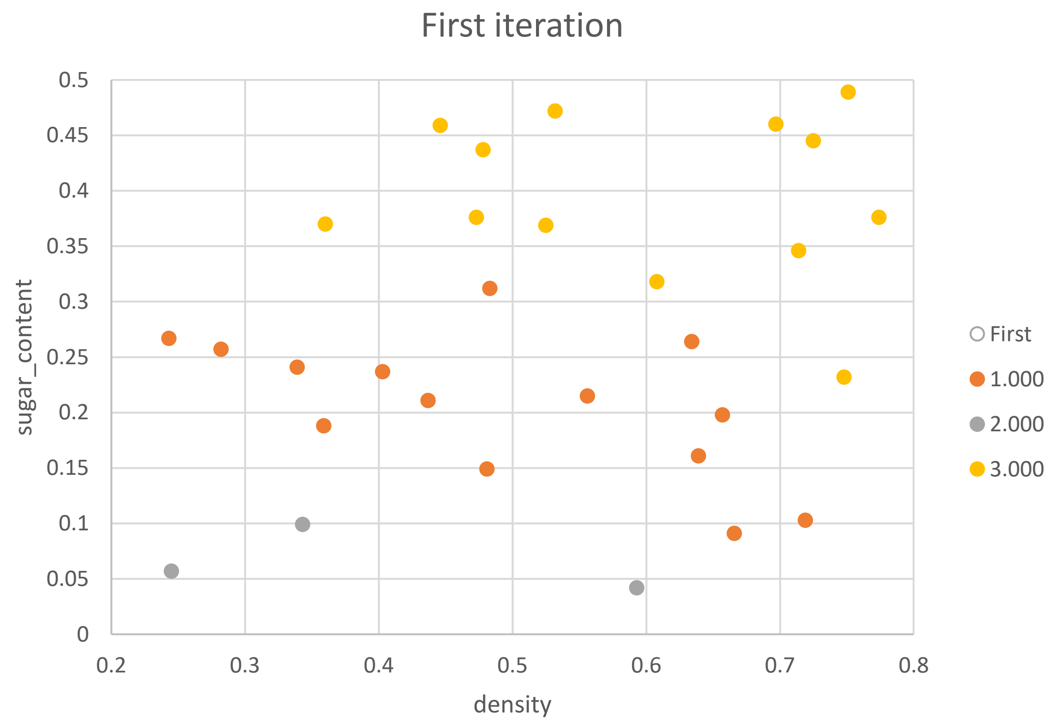
id	density	sugar content	id	density	sugar content
1	.697	.460	16	.593	.042
2	.774	.376	17	.719	.103
3	.634	.264	18	.359	.188
4	.608	.318	19	.339	.241
5	.556	.215	20	.282	.257
6	.403	.237	21	.748	.232
7	.481	.149	22	.714	.346
8	.437	.211	23	.483	.312
9	.666	.091	24	.478	.437
10	.243	.267	25	.525	.369
11	.245	.057	26	.751	.489
12	.343	.099	27	.532	.472
13	.639	.161	28	.473	.376
14	.657	.198	29	.725	.445
15	.360	.370	30	.446	.459

Example: Watermelon Dataset

- Repeat aforementioned process until it converges

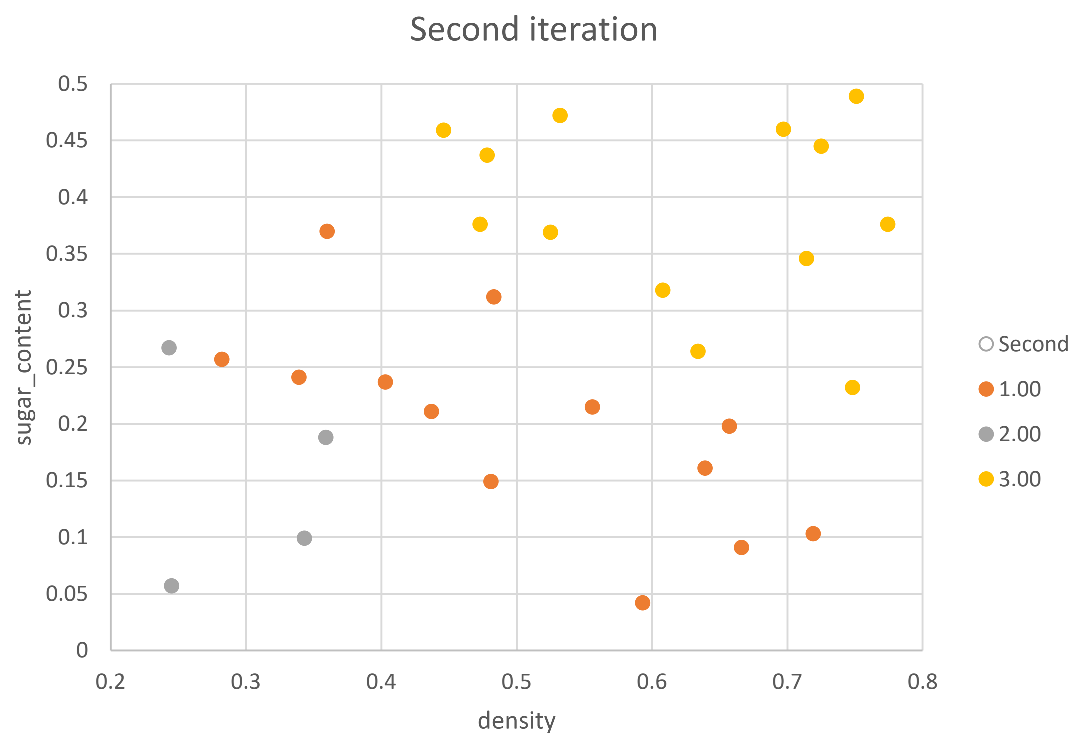
id	density	sugar content	id	density	sugar content
1	.697	.460	16	.593	.042
2	.774	.376	17	.719	.103
3	.634	.264	18	.359	.188
4	.608	.318	19	.339	.241
5	.556	.215	20	.282	.257
6	.403	.237	21	.748	.232
7	.481	.149	22	.714	.346
8	.437	.211	23	.483	.312
9	.666	.091	24	.478	.437
10	.243	.267	25	.525	.369
11	.245	.057	26	.751	.489
12	.343	.099	27	.532	.472
13	.639	.161	28	.473	.376
14	.657	.198	29	.725	.445
15	.360	.370	30	.446	.459

Example: Watermelon Dataset



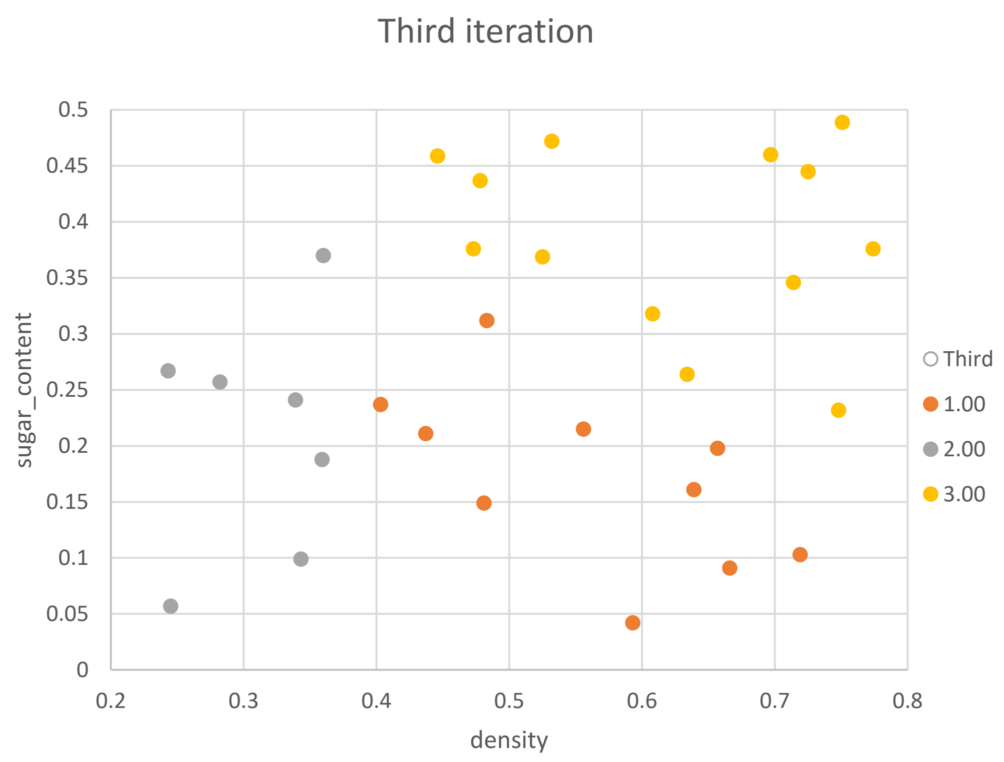
id	density	sugar content	id	density	sugar content
1	.697	.460	16	.593	.042
2	.774	.376	17	.719	.103
3	.634	.264	18	.359	.188
4	.608	.318	19	.339	.241
5	.556	.215	20	.282	.257
6	.403	.237	21	.748	.232
7	.481	.149	22	.714	.346
8	.437	.211	23	.483	.312
9	.666	.091	24	.478	.437
10	.243	.267	25	.525	.369
11	.245	.057	26	.751	.489
12	.343	.099	27	.532	.472
13	.639	.161	28	.473	.376
14	.657	.198	29	.725	.445
15	.360	.370	30	.446	.459

Example: Watermelon Dataset



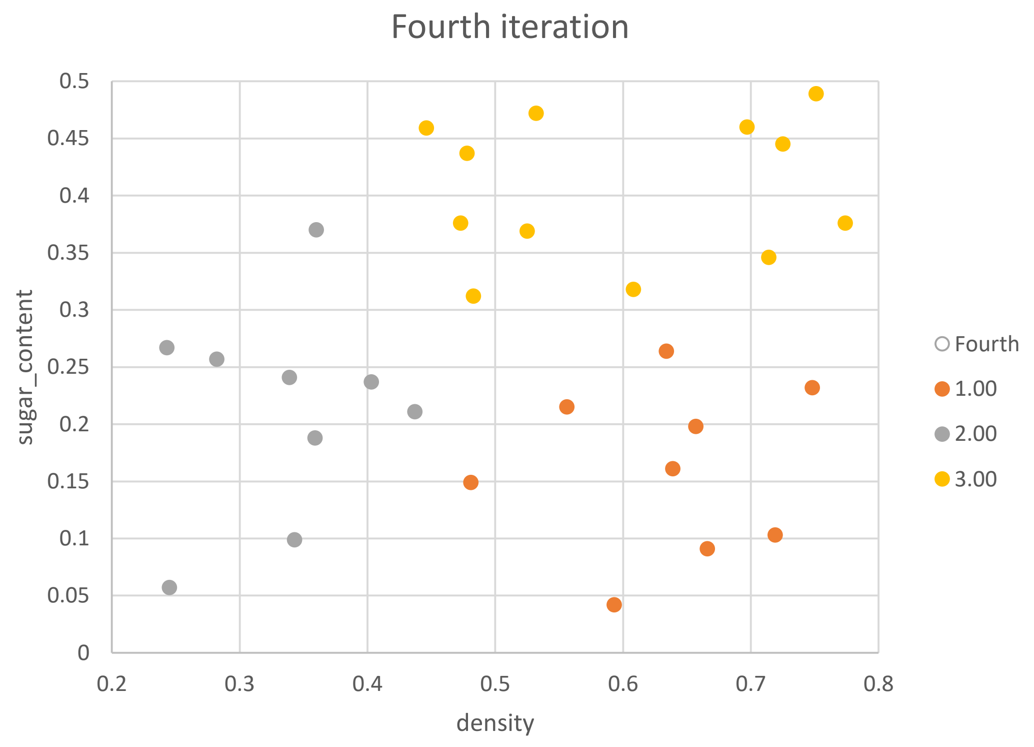
id	density	sugar content	id	density	sugar content
1	.697	.460	16	.593	.042
2	.774	.376	17	.719	.103
3	.634	.264	18	.359	.188
4	.608	.318	19	.339	.241
5	.556	.215	20	.282	.257
6	.403	.237	21	.748	.232
7	.481	.149	22	.714	.346
8	.437	.211	23	.483	.312
9	.666	.091	24	.478	.437
10	.243	.267	25	.525	.369
11	.245	.057	26	.751	.489
12	.343	.099	27	.532	.472
13	.639	.161	28	.473	.376
14	.657	.198	29	.725	.445
15	.360	.370	30	.446	.459

Example: Watermelon Dataset



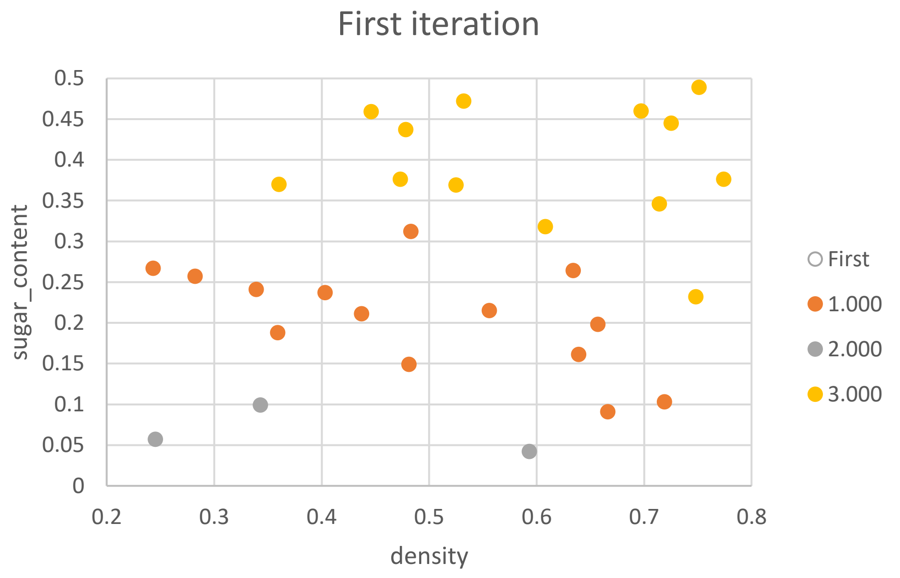
id	density	sugar content	id	density	sugar content
1	.697	.460	16	.593	.042
2	.774	.376	17	.719	.103
3	.634	.264	18	.359	.188
4	.608	.318	19	.339	.241
5	.556	.215	20	.282	.257
6	.403	.237	21	.748	.232
7	.481	.149	22	.714	.346
8	.437	.211	23	.483	.312
9	.666	.091	24	.478	.437
10	.243	.267	25	.525	.369
11	.245	.057	26	.751	.489
12	.343	.099	27	.532	.472
13	.639	.161	28	.473	.376
14	.657	.198	29	.725	.445
15	.360	.370	30	.446	.459

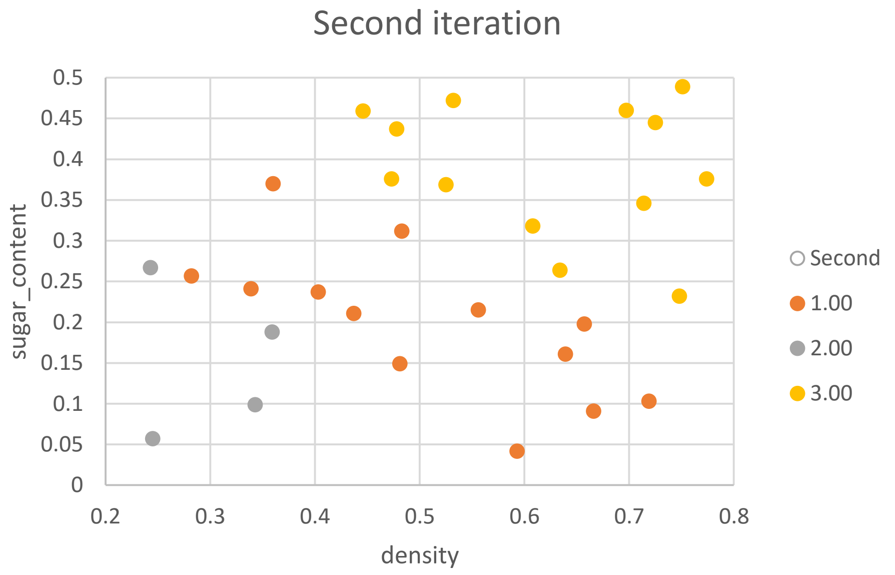
Example: Watermelon Dataset

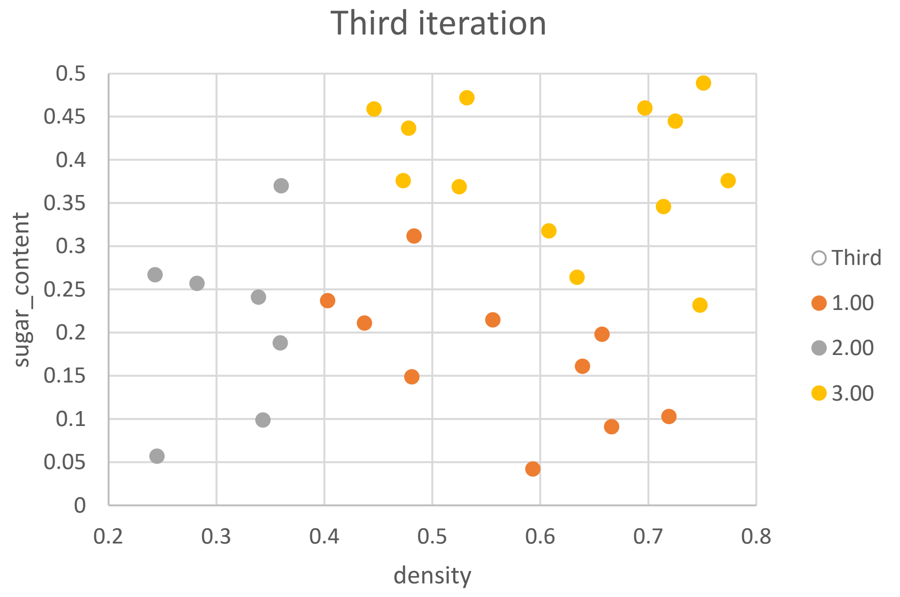


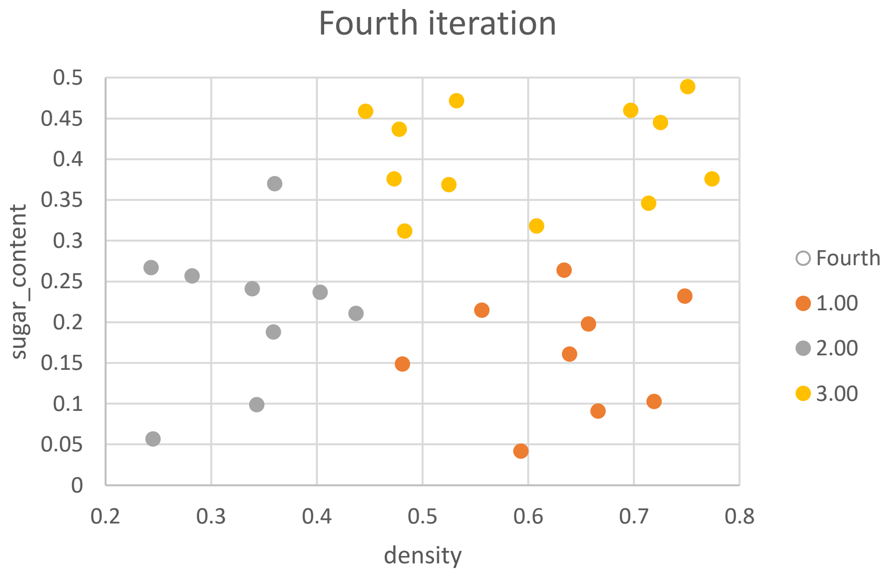
id	density	sugar content	id	density	sugar content
1	.697	.460	16	.593	.042
2	.774	.376	17	.719	.103
3	.634	.264	18	.359	.188
4	.608	.318	19	.339	.241
5	.556	.215	20	.282	.257
6	.403	.237	21	.748	.232
7	.481	.149	22	.714	.346
8	.437	.211	23	.483	.312
9	.666	.091	24	.478	.437
10	.243	.267	25	.525	.369
11	.245	.057	26	.751	.489
12	.343	.099	27	.532	.472
13	.639	.161	28	.473	.376
14	.657	.198	29	.725	.445
15	.360	.370	30	.446	.459

Example: Watermelon Dataset





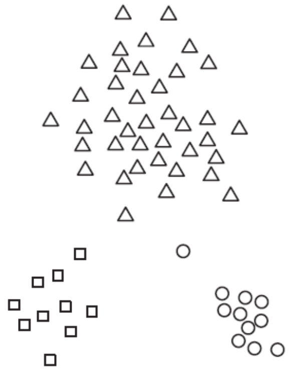


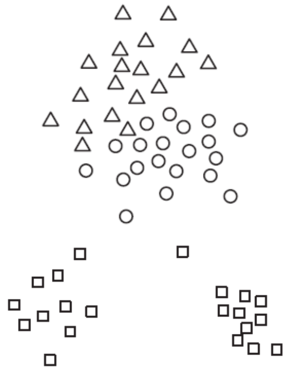


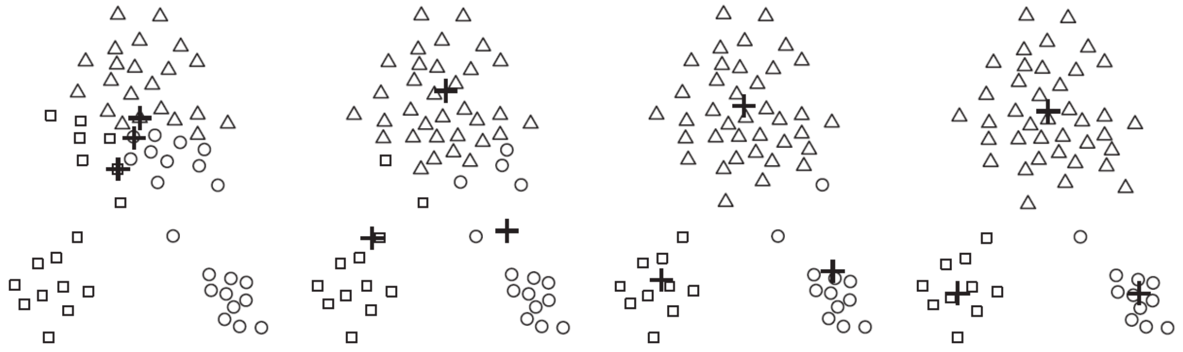
How do we initialize K-means?

- If there are K ‘real’ clusters then the chance of selecting one centroid from each cluster is small.
- Some heuristics
  - Randomly pick K data points as prototypes
  - Pick prototype i+1 to be the farthest from prototypes{1,2….i}

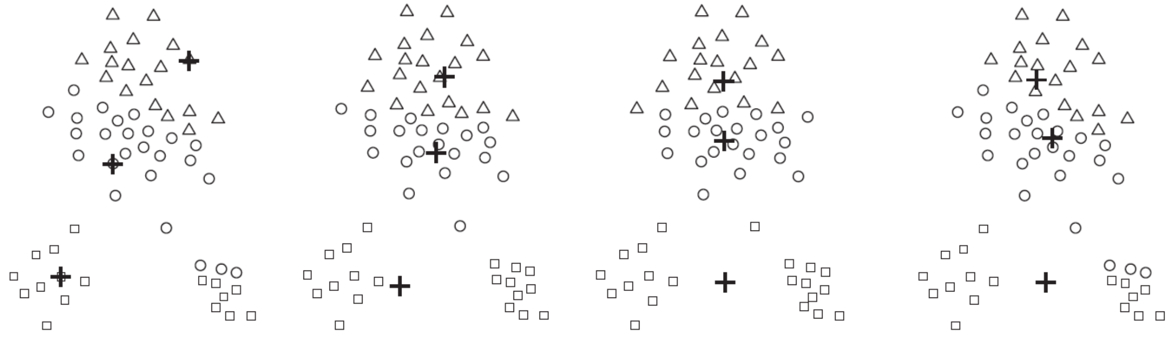
Effect of Initial Points


(a) Optimal clustering.


(b) Suboptimal clustering.


(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

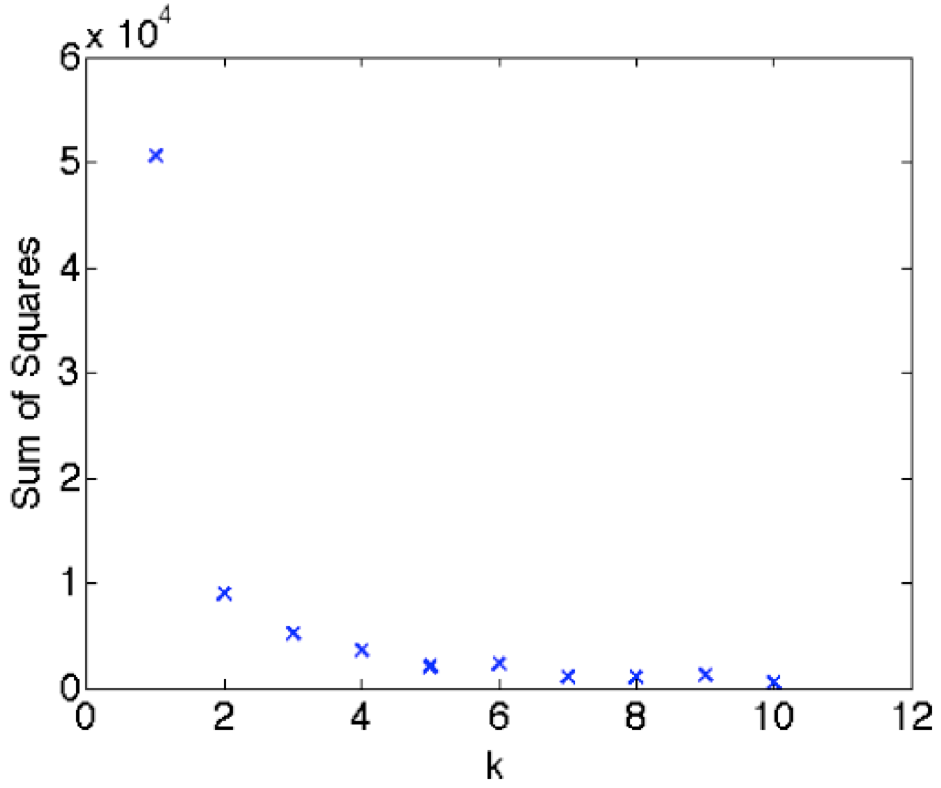
Good Clustering


(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

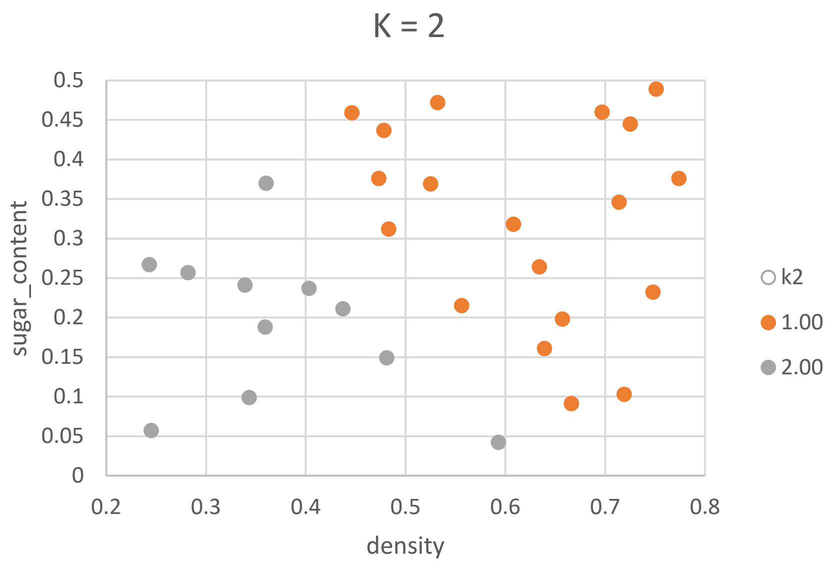
Poor Clustering

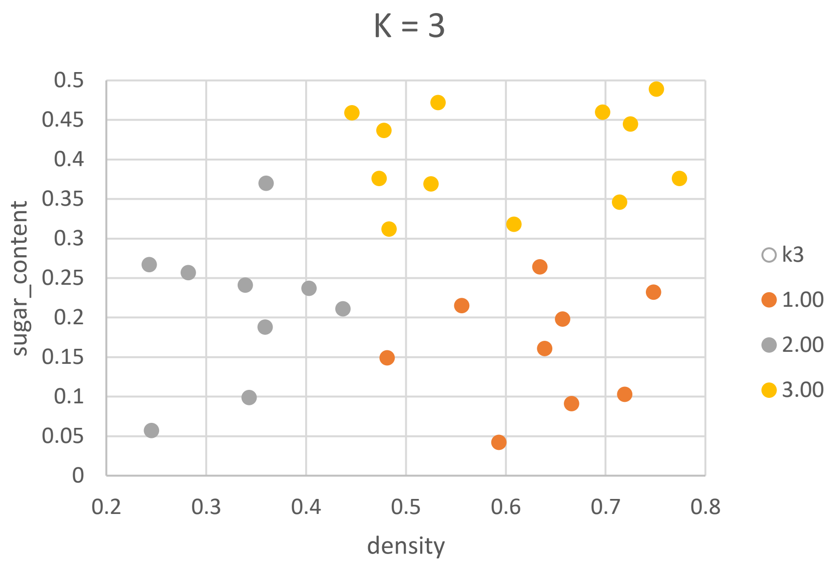
How to choose K ?

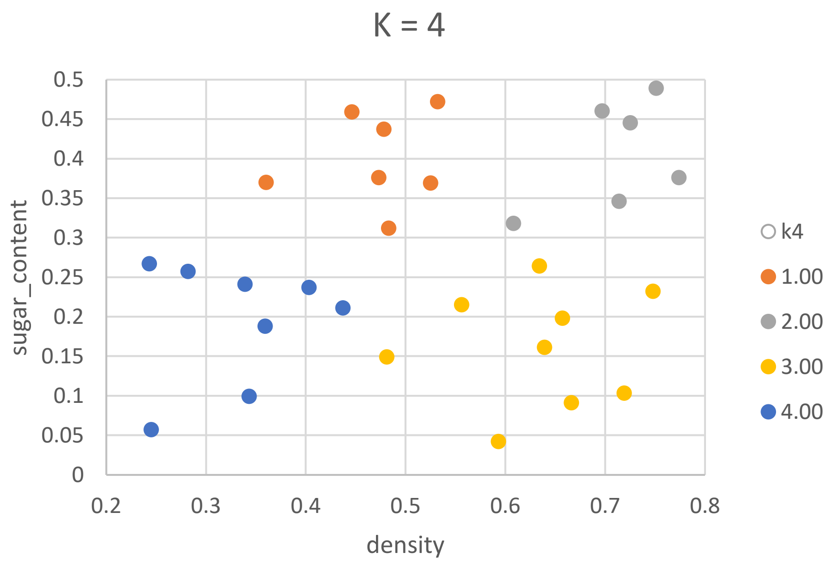
- The loss function J generally decreases with K .

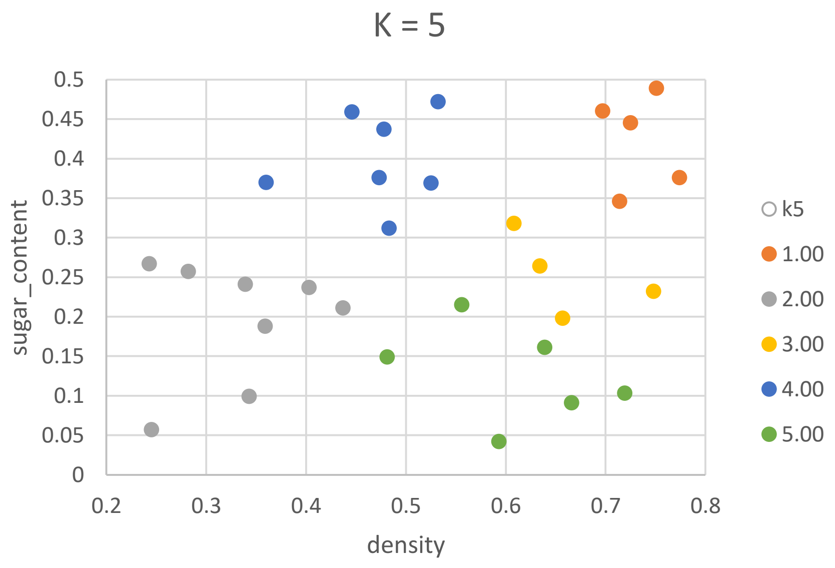


Example: Different K of Watermelon Dataset









How to choose K ?

- Gap statistic
- Cross-validation: Partition data into two sets. Estimate prototypes on one and use these to compute the loss function on the other.
- Stability of clusters: Measure the change in the clusters obtained by resampling or splitting the data.
- Non-parametric approach: Place a prior on K .

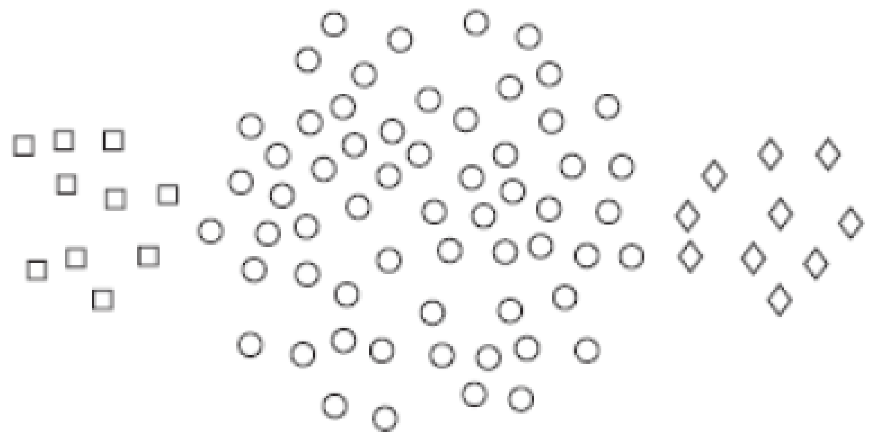
Pre-processing and Post-processing

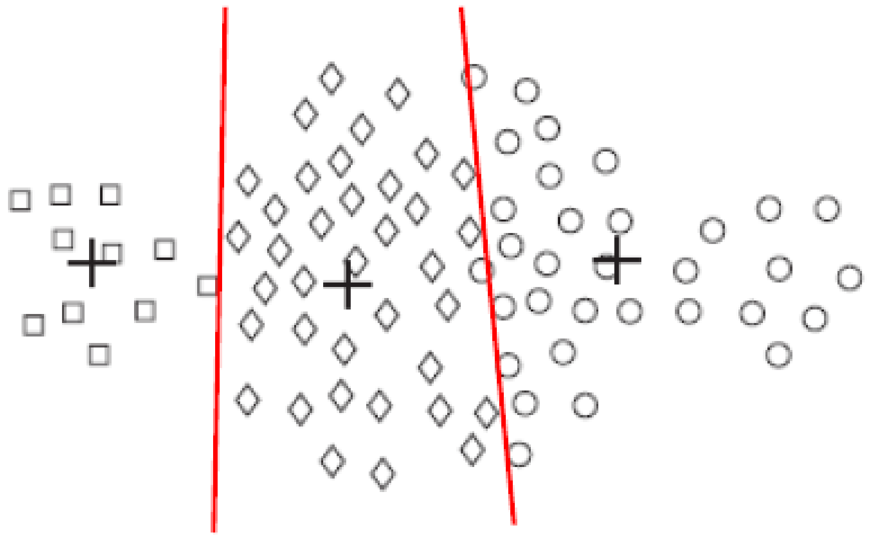
- Pre-processing
  - **Normalize** the data (e.g., scale to unit standard deviation)
  - **Eliminate outliers**
- Post-processing
  - **Eliminate** small clusters that may represent outliers
  - **Split** ‘loose’ clusters, i.e., clusters with relatively high SSE
  - **Merge** clusters that are ‘close’ and that have relatively low SSE

Limitations of K-means

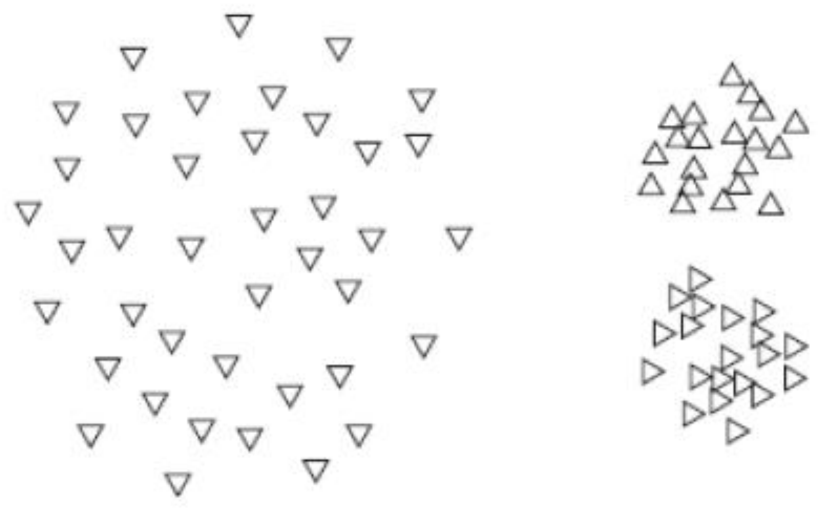
- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-Spherical Shapes

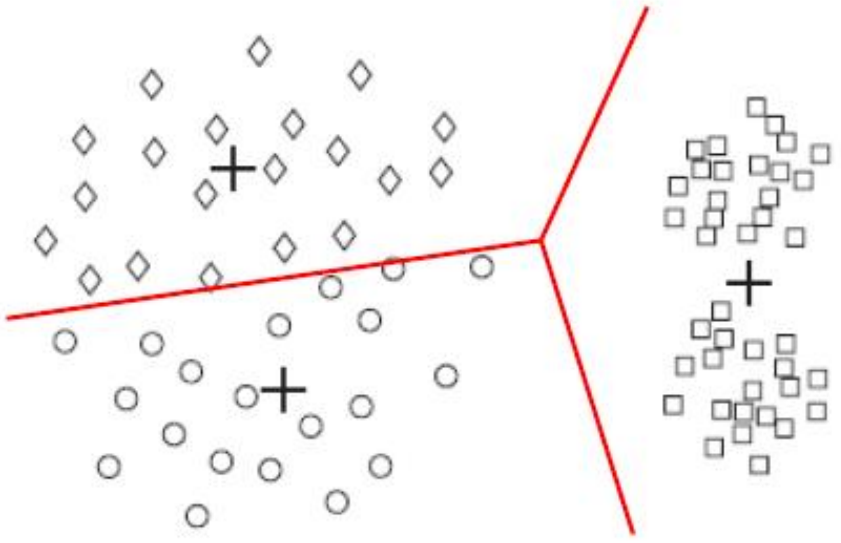
Limitations of K-means: Differing Sizes


(a) Original points.

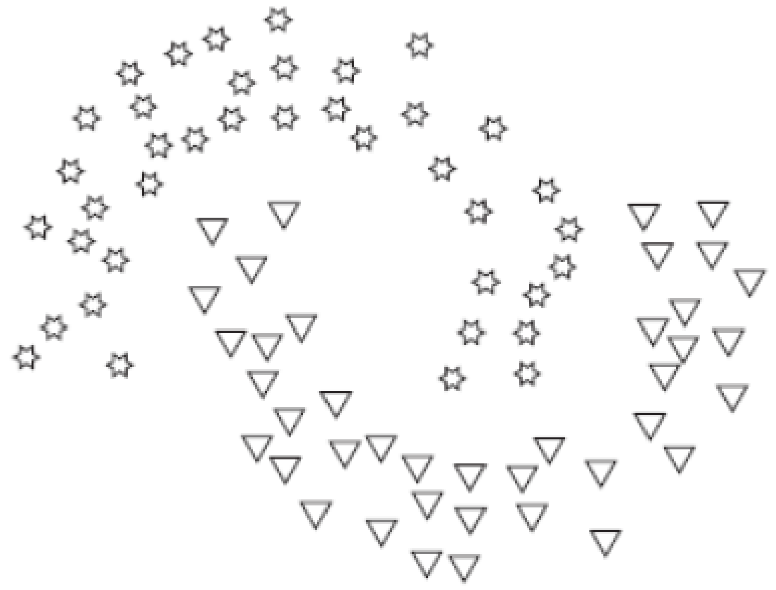

(b) Three K-means clusters.

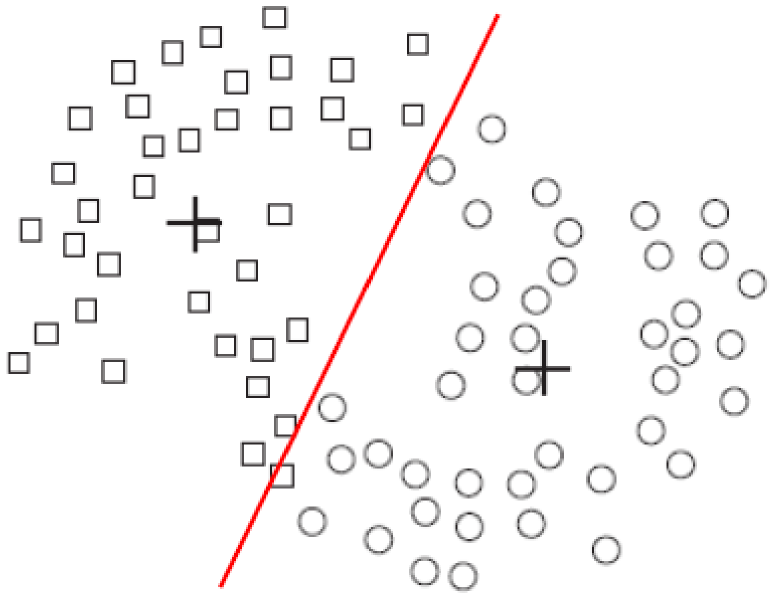
Limitations of K-means: Differing Densities


(a) Original points.


(b) Three K-means clusters.

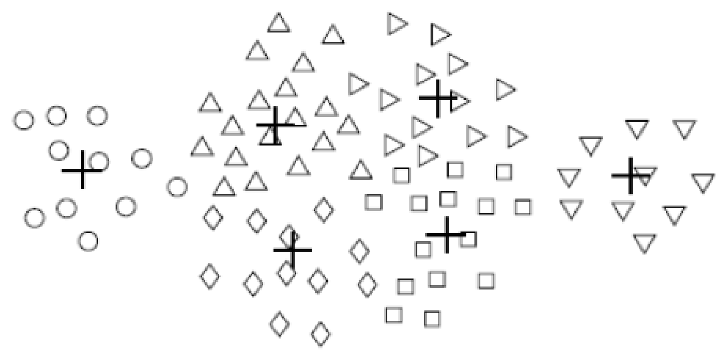
Limitations of K-means: Non-globular shapes

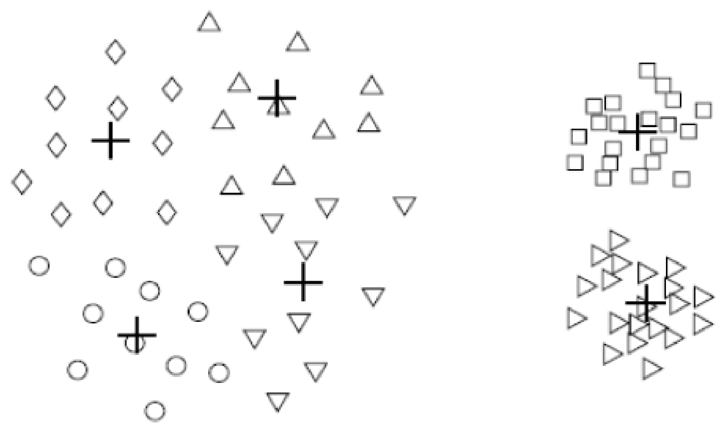

(a) Original points.

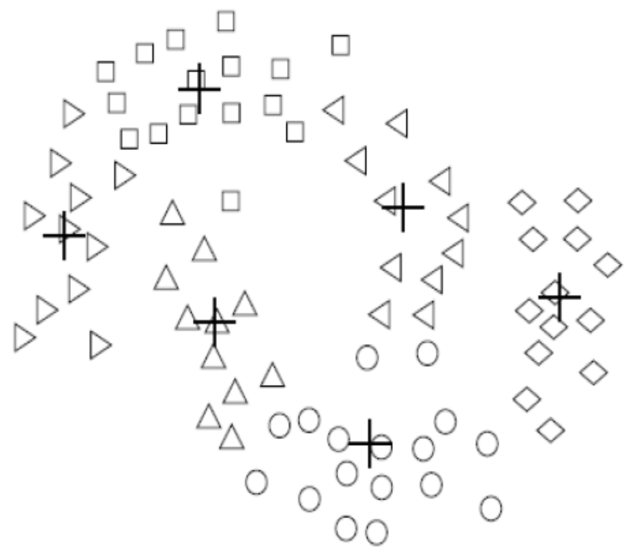

(b) Two K-means clusters.

Overcoming K-means Limitations

- Use a larger number of clusters
- Several clusters represent a true cluster
- Or use density based method


(a) Unequal sizes.

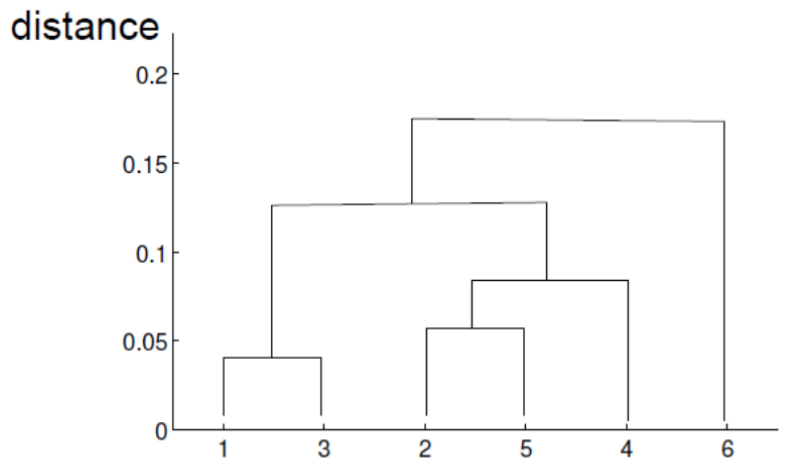

(b) Unequal densities.

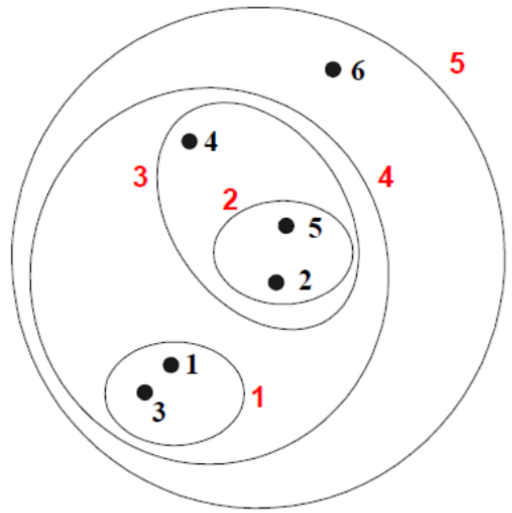

(c) Non-spherical shapes.

Hierarchical Clustering

Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree.
- Can be visualized as a **dendrogram**
  - A tree like diagram that records the sequences of merges or splits



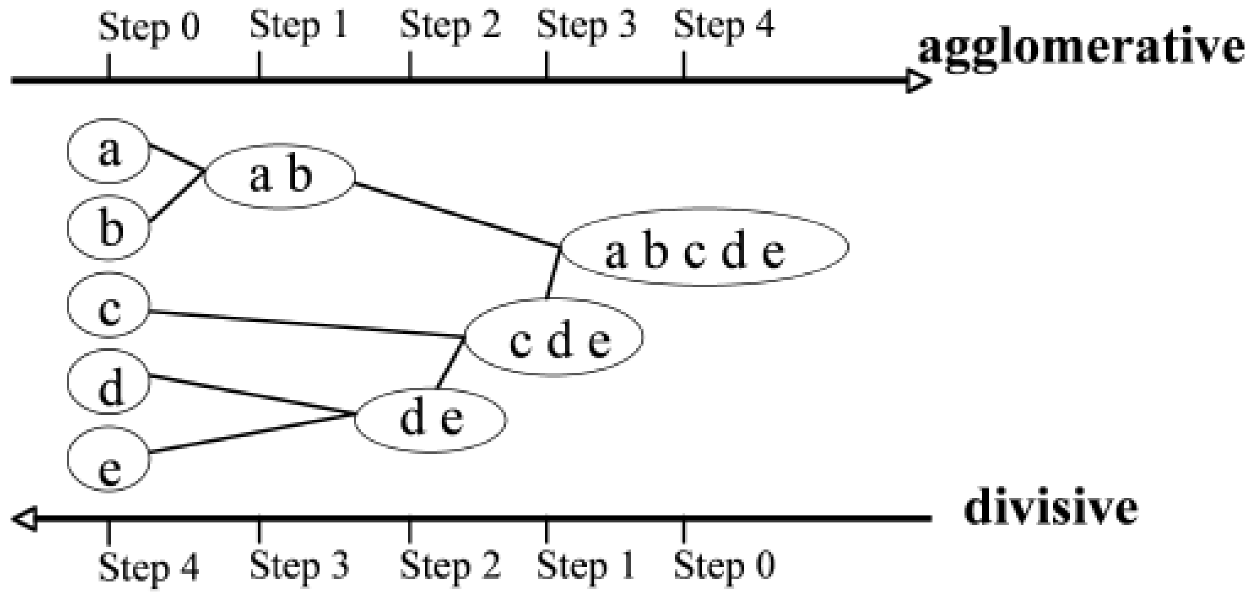


Strength of Hierarchical Clustering

- You do not have to assume any particular number of clusters
    - Any desired number of clusters can be obtained by ‘cutting’ the dendogram at the proper level
- They may correspond to meaningful taxonomies
    - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)

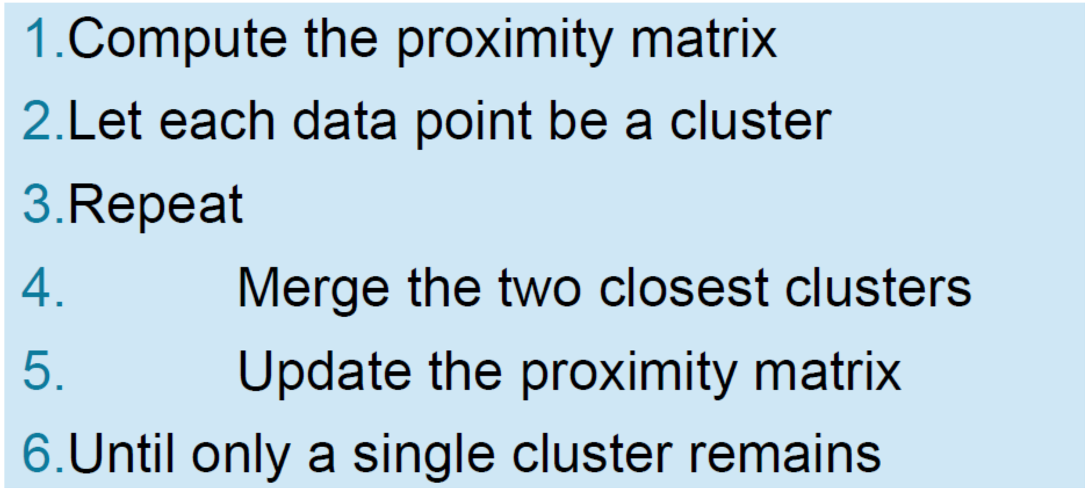
Hierarchical Clustering

- **Bottom-up (agglomerative):** Recursively merge two groups with the smallest between-cluster similarity.
- **Top-down (divisive):** Recursively split a least-coherent (e.g. largest diameter) cluster.
- Users can then choose a cut through the hierarchy to represent the most natural division into clusters (e.g. where intergroup similarity exceeds some threshold).



Agglomerative Clustering Algorithm

- More popular hierarchical clustering technique
- Basic algorithm is straightforward

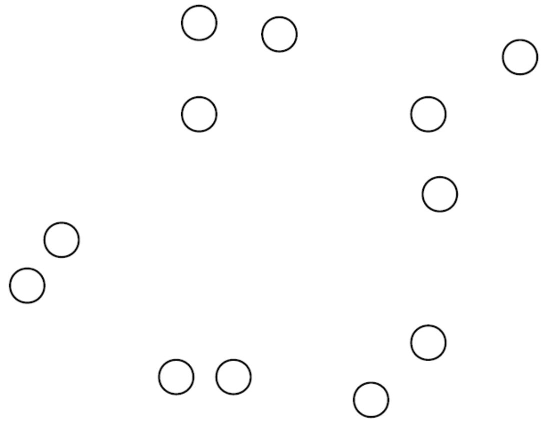


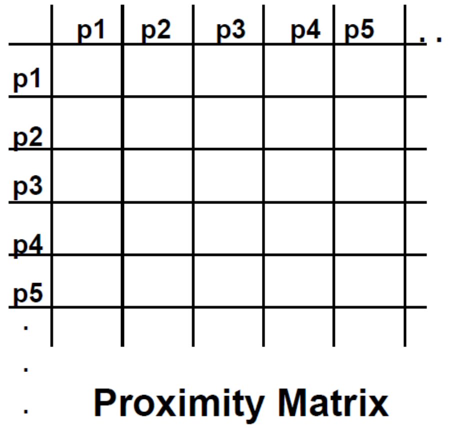
```
1. Compute the proximity matrix
2. Let each data point be a cluster
3. Repeat
4.         Merge the two closest clusters
5.         Update the proximity matrix
6. Until only a single cluster remains
```

- Key operation is the computation of the proximity of two clusters → Different approaches to **defining the distance between clusters** distinguish the different algorithms

Starting Situation

- Start with clusters of individual points and a proximity matrix

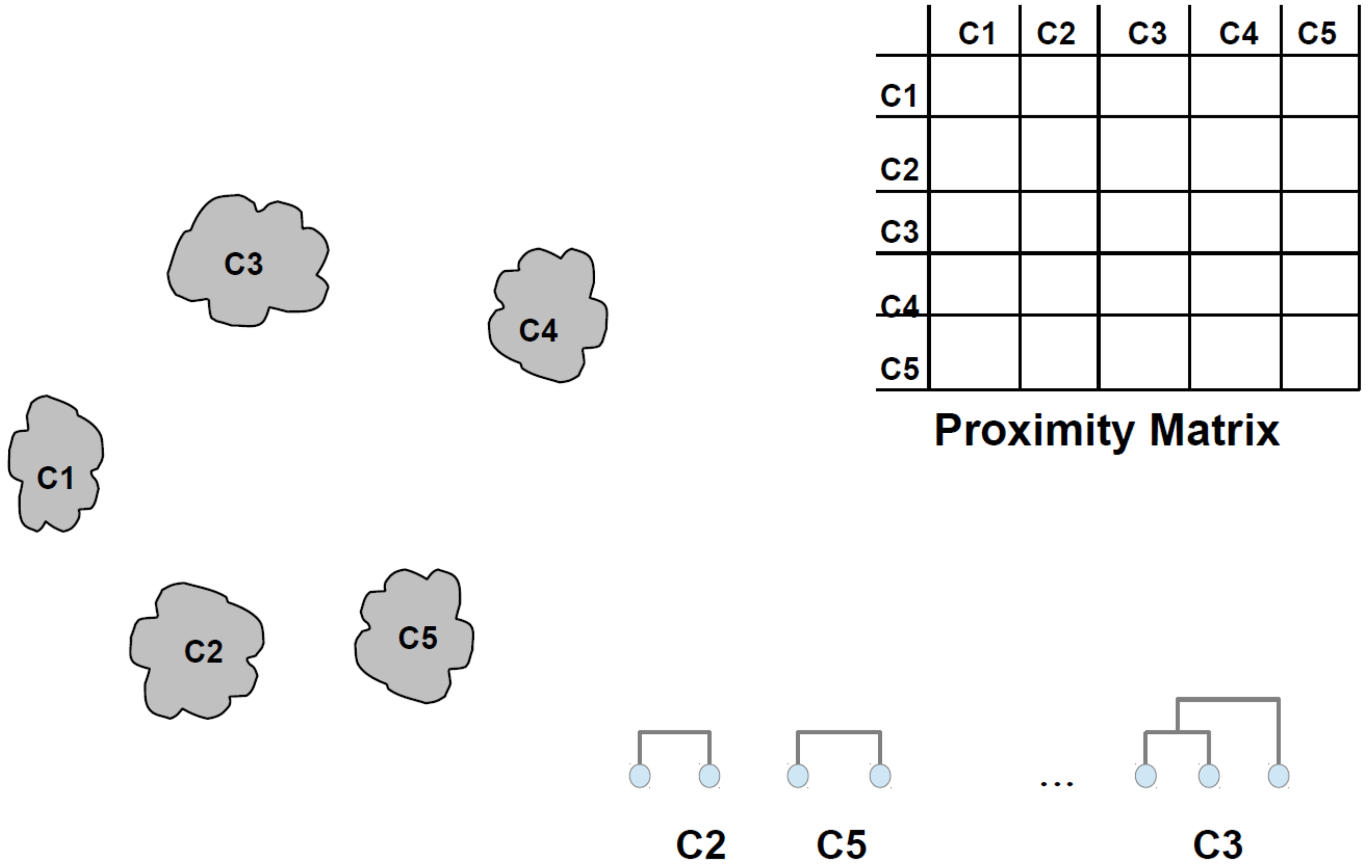




Proximity Matrix

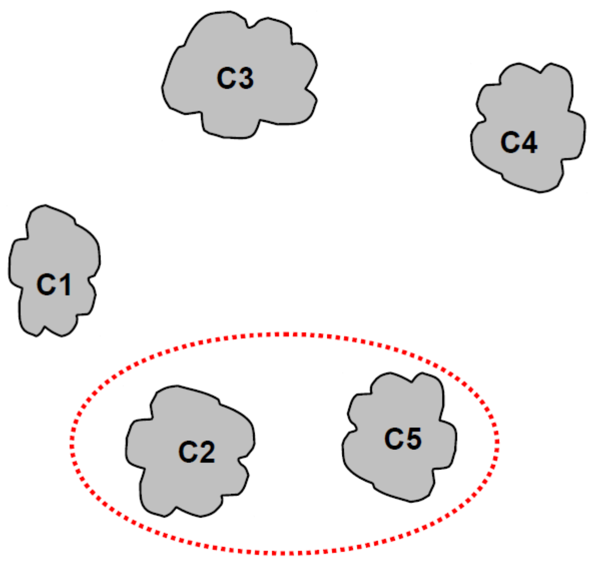
Intermediate Situation

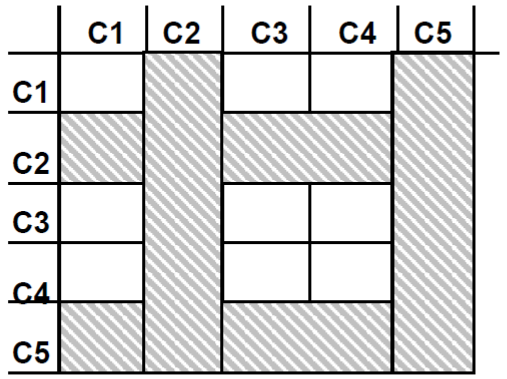
- After some merging steps, we have some clusters



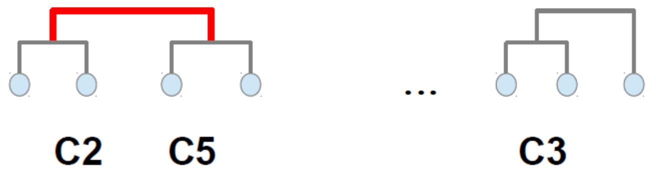
Intermediate Situation

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



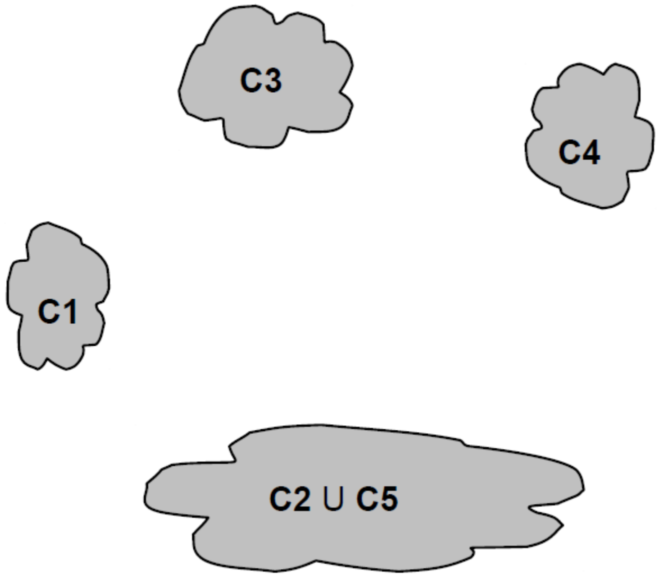


Proximity Matrix



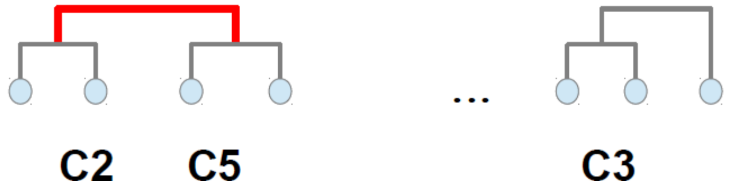
After Merging

- The question is “How do we update the proximity matrix?”



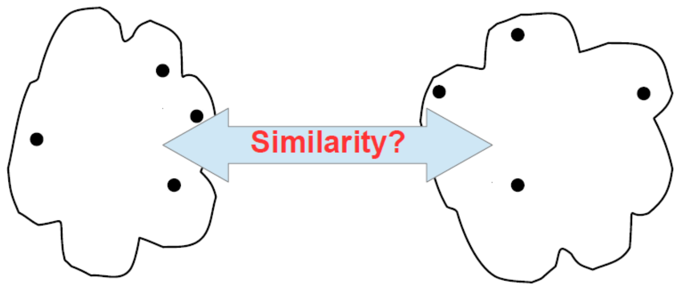
	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

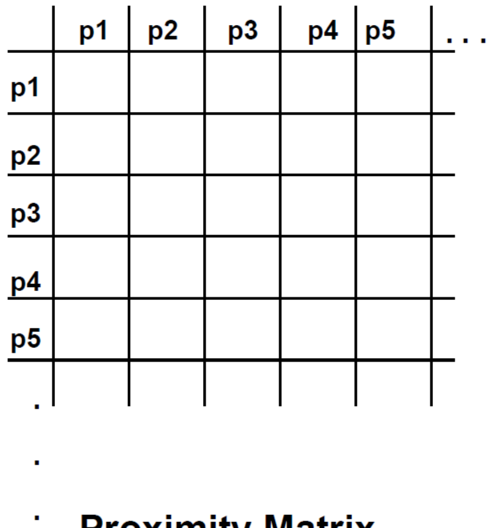
Proximity Matrix



How to Define Inter-Cluster Similarity

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward’s Method uses squared error

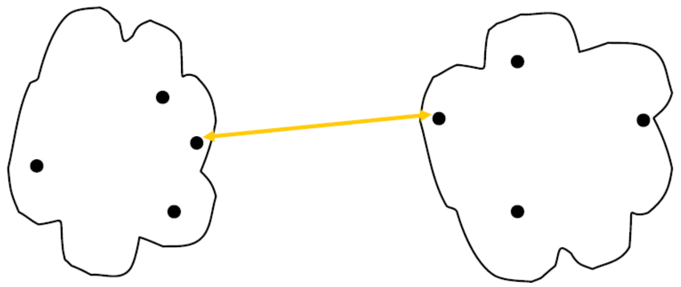


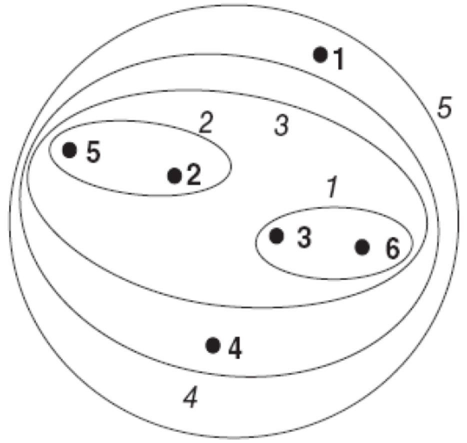


Proximity Matrix

How to Define Inter-Cluster Similarity

- MIN (Single link)
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error



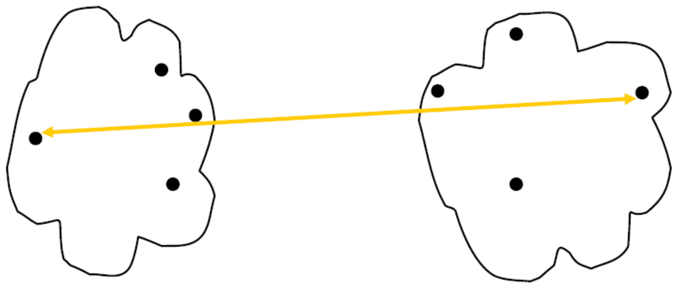


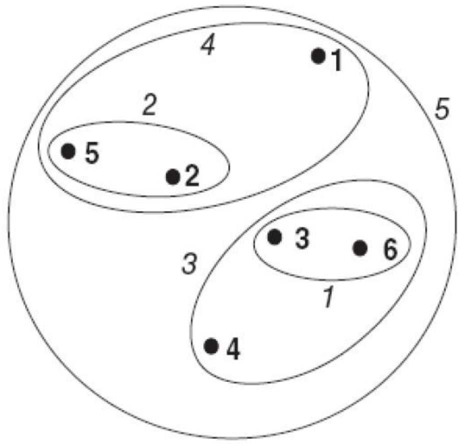
Advantage: Non-spherical, non-convex clusters

Problem: Chaining

How to Define Inter-Cluster Similarity

- MIN
- MAX (Complete link)
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error





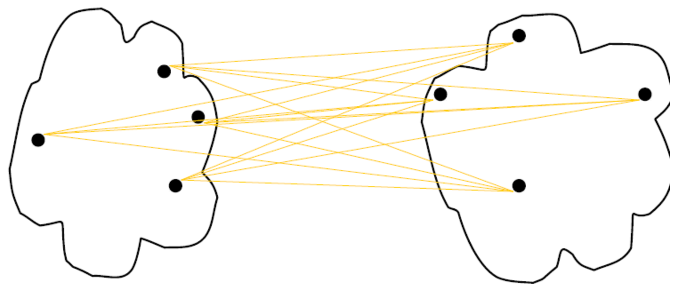
Advantage: more robust against noise (no chaining)

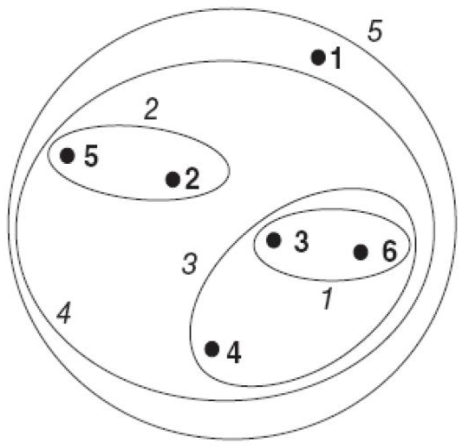
Problem: Tends to break large clusters,

Biased towards globular/round clusters

How to Define Inter-Cluster Similarity

- MIN
- MAX
- Group Average (Average link)
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward’s Method uses squared error



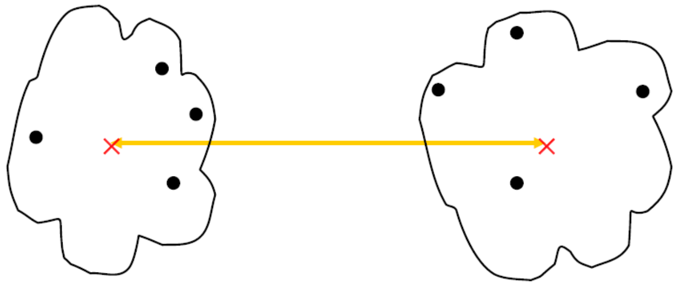


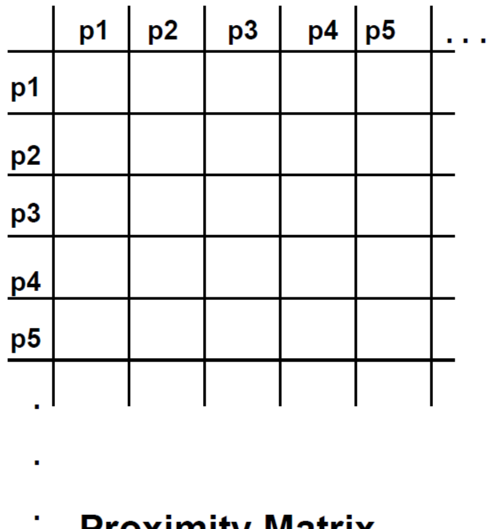
Compromise between Single and Complete Link

How to Define Inter-Cluster Similarity

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

Problem: Inversion (Two clusters that are merged may be more similar than the pair of clusters that were merged in a previous step)



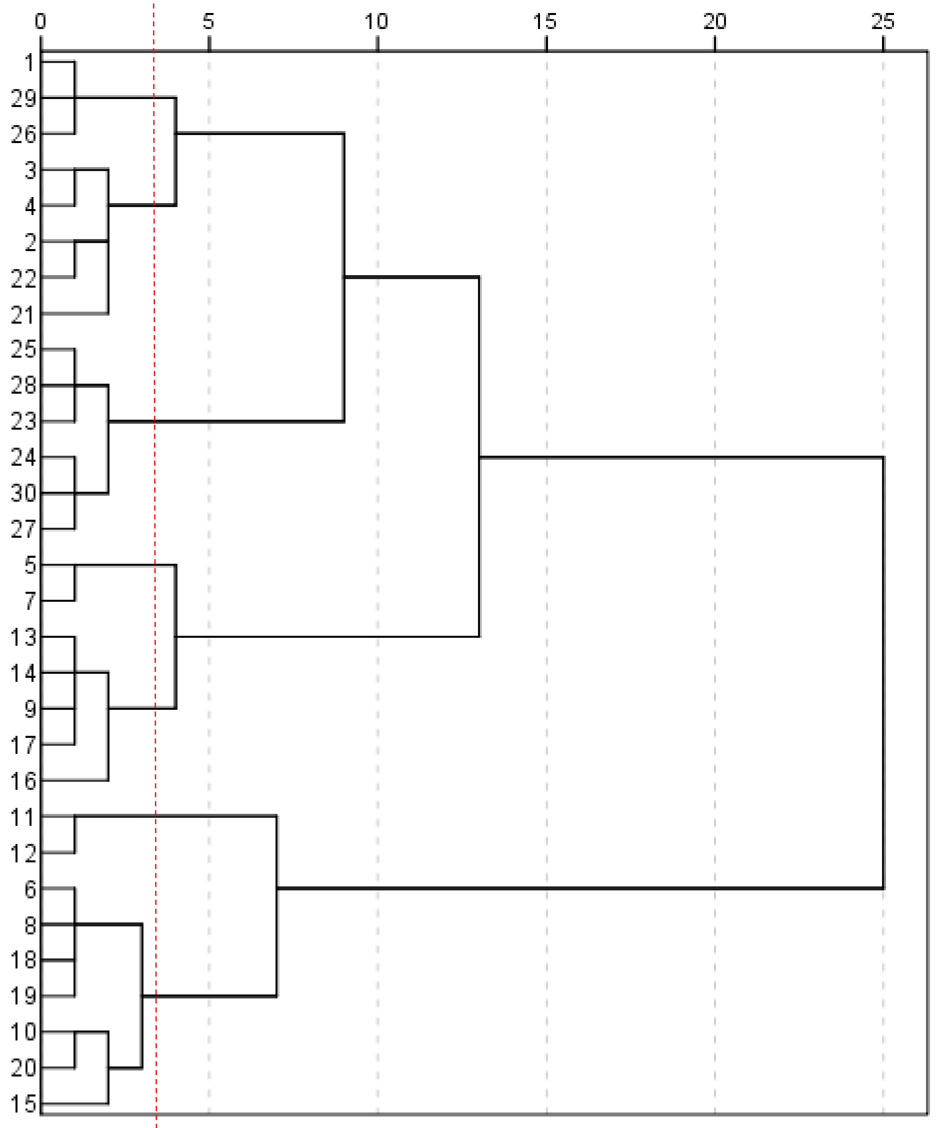


Proximity Matrix

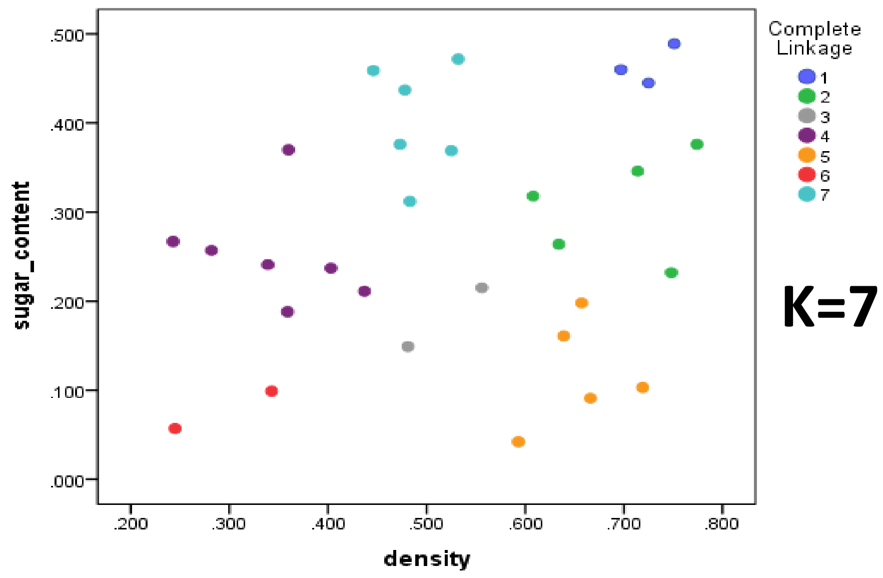
Example

- Hierarchical clustering with Watermelon Dataset
  - Bottom-up + d_max

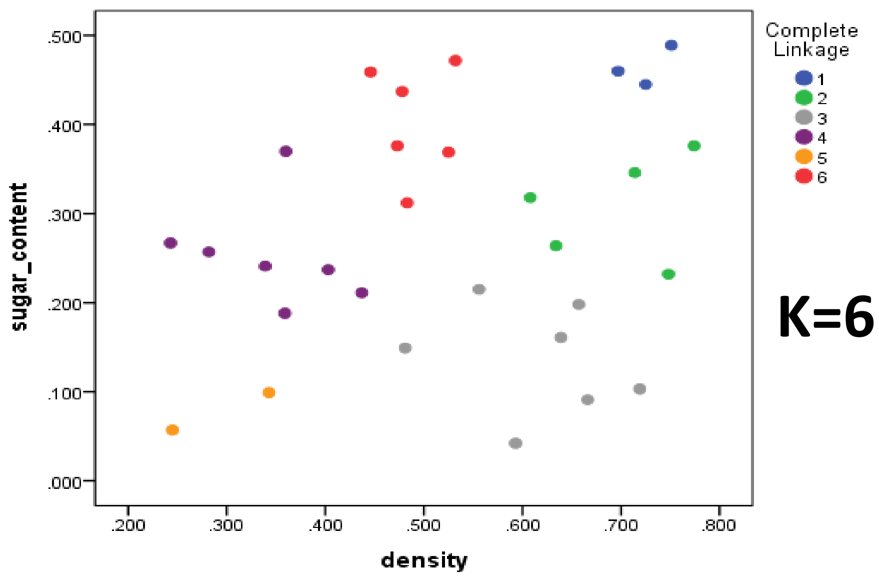
id	density	sugar content	id	density	sugar content
1	.697	.460	16	.593	.042
2	.774	.376	17	.719	.103
3	.634	.264	18	.359	.188
4	.608	.318	19	.339	.241
5	.556	.215	20	.282	.257
6	.403	.237	21	.748	.232
7	.481	.149	22	.714	.346
8	.437	.211	23	.483	.312
9	.666	.091	24	.478	.437
10	.243	.267	25	.525	.369
11	.245	.057	26	.751	.489
12	.343	.099	27	.532	.472
13	.639	.161	28	.473	.376
14	.657	.198	29	.725	.445
15	.360	.370	30	.446	.459



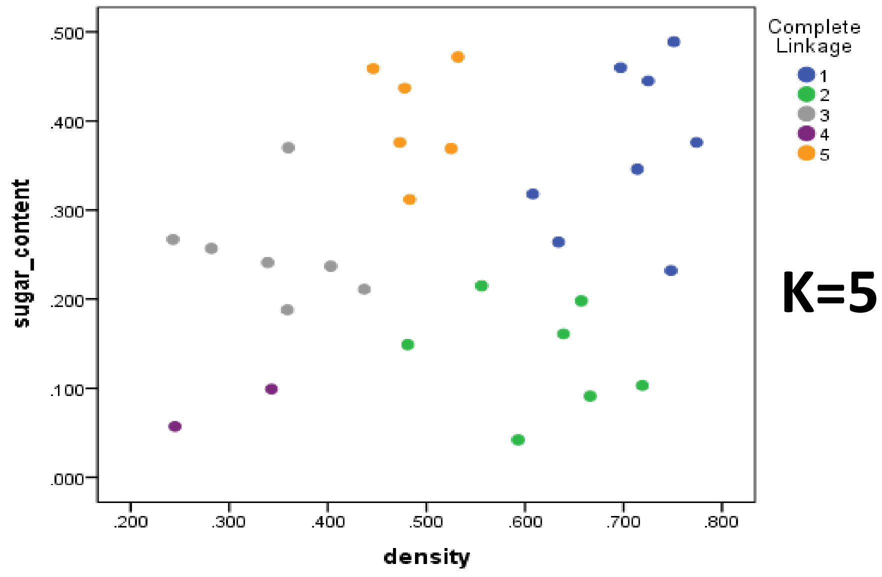
Example



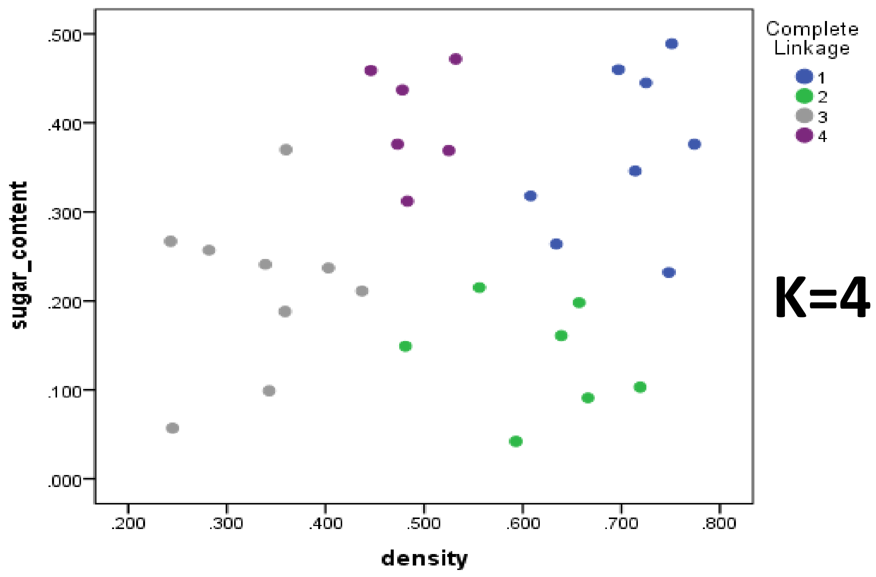
K=7



K=6



K=5



K=4

Limitations of Hierarchical Clustering

- **Greedy:** Once a decision is made to combine two clusters, it cannot be undone
- **No global objective function** is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Chaining, breaking large clusters

Gaussian Mixture Models

Gaussian Mixture Models

- Multivariate Normal Distribution

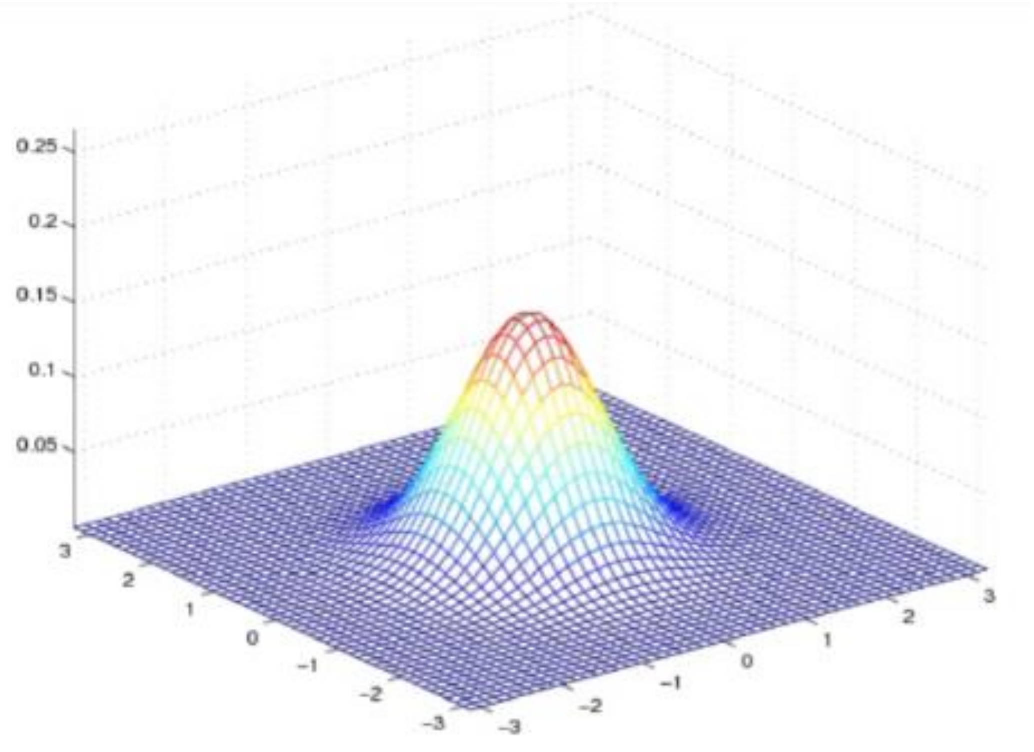


Figure 6: Multivariate Normal Distribution

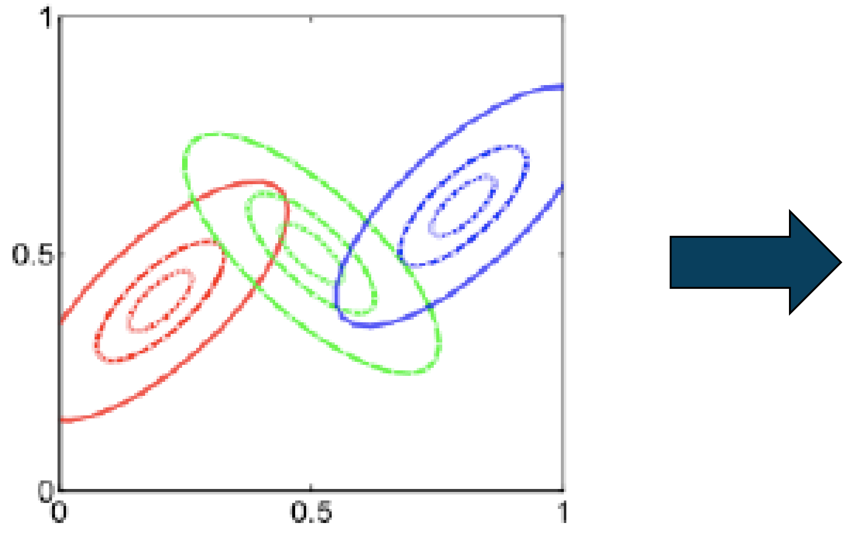
$$x \sim \mathcal{N}(\mu, \Sigma)$$

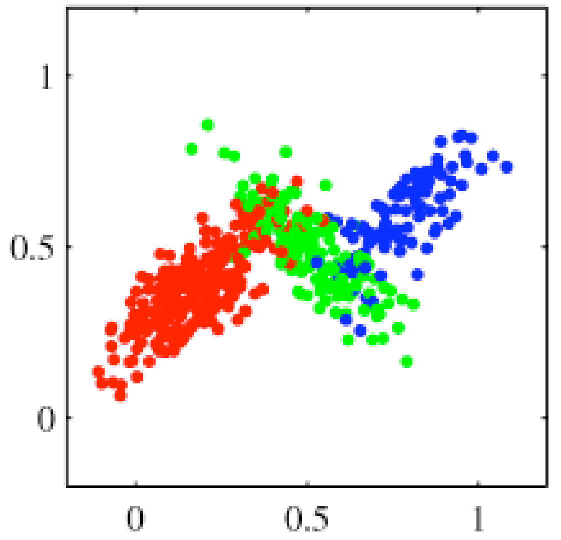
$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)}$$

Gaussian Mixture Model

- Probabilistic story: Each cluster is associated with a Gaussian distribution, To generate data, randomly choose a cluster k with probability π_k and sample from its distribution
- Likelihood

$$P(\boldsymbol{x}) = \prod_i^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad where \sum_{k=1}^{K} \pi_k = 1, \quad 0 \le \pi_k \le 1$$





Gaussian Mixture Model with Latent Variable Introduced

- Each $\boldsymbol{x}$ is associated with a K-dimensional latent variable $\boldsymbol{z} = (z_1, \dots z_K)$, having a one-hot representation

$$P(z_k = 1) = \pi_k$$

$$P\,(x_i \,|\, \boldsymbol{z}) = \prod_{k=1}^{K} \mathcal{N}(x_i \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k} \qquad P\,(x_i \,|\, z_k = 1) = \mathcal{N}(x_i \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$P(x_i) = \sum_{\boldsymbol{z}} P(x_i|\boldsymbol{z})P(\boldsymbol{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Responsibility

- Conditional probability of $\boldsymbol{z}$ given $\boldsymbol{x}$

$$\gamma(z_{ik}) = p(z_{ik} = 1|x_i) = \frac{P(z_{ik} = 1)p(x_i|z_{ik} = 1)}{\sum_{k=1}^{K}(z_{ik} = 1)p(x_i|z_{ik} = 1)}$$

$$= \frac{\pi_k \mathcal{N}(x_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

- $\gamma(z_{\mathrm{i}k})$ can be viewed as the responsibility that component k takes for "explaining" the observation x_i

Optimization

- Maximum likelihood Estimation (MLE)

$$\log P(\boldsymbol{x}|\theta) = \sum_{i=1}^{\mathrm{N}} \log\{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\}$$

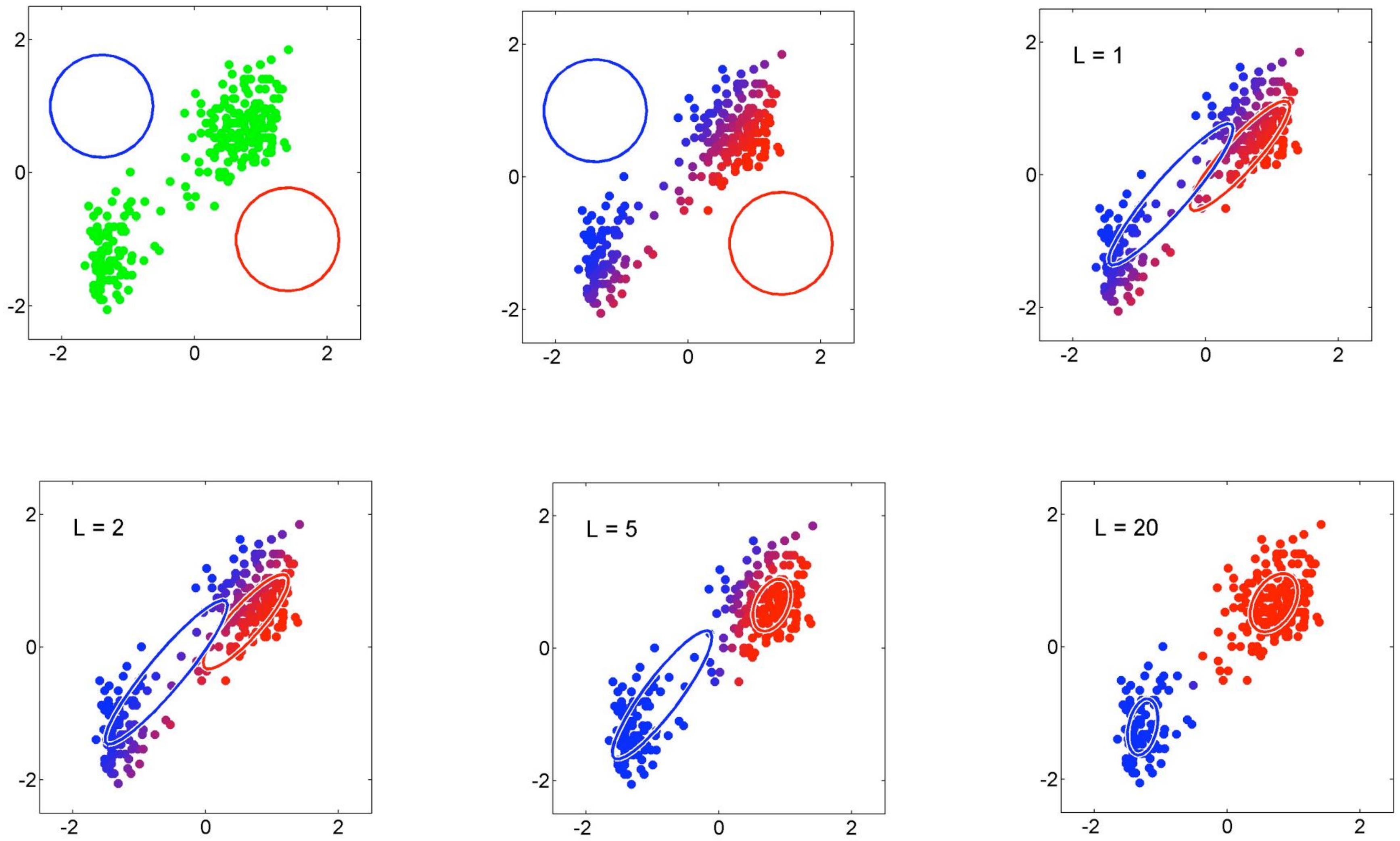
- Optimization: Expectation-Maximization (EM)
  - E-step: estimate responsibility

$$\gamma(z_{ik}) \triangleq E(z_{ik}) = \frac{\pi_k N(x_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_k N(x_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

  - M-step: re-estimate parameters

$$\pi_k = \frac{\sum_i \gamma(z_{ik})}{N}, \ \mu_k = \frac{\sum_i \gamma(z_{ik}) x_i}{\sum_i \gamma(z_{ik})}, \ \Sigma_k = \frac{\sum_i \gamma(z_{ik})(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_i \gamma(z_{ik})}$$

Example



Example

- GMM with Watermelon Dataset
  - Init:

    $\pi_1 = \pi_2 = \pi_3 = 1/3$

    $\mu_1 = x_6, \mu_2 = x_{22}, \mu_3 = x_{27}$

    $\Sigma_1 = \Sigma_2 = \Sigma_3 = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$
  - Iter 1:

    $\gamma_{11} = 0.219, \gamma_{12} = 0.404, \gamma_{13} = 0.377$

    $\pi'_1 = 0.361, \pi'_2 = 0.323, \pi'_3 = 0.316$

    $\mu'_1 = (0.419; 0.251), \mu'_2 = (0.571; 0.281),$

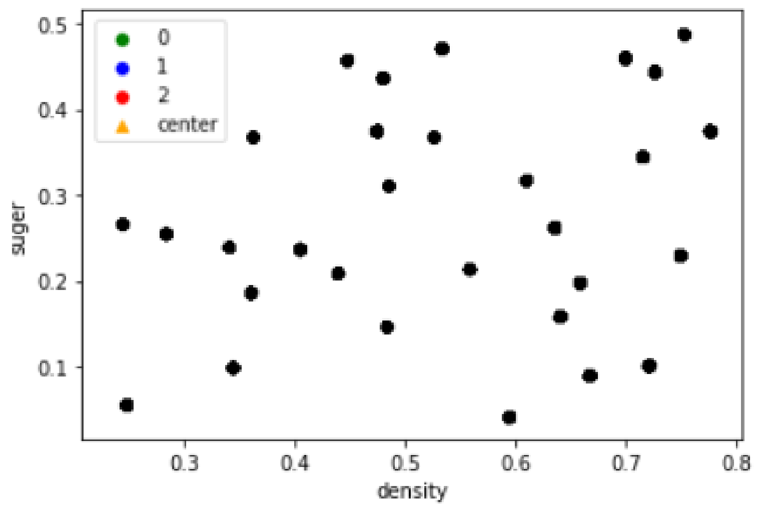
    $\mu'_3 = (0.534; 0.295)$

    $\Sigma'_1 = \begin{pmatrix} 0.025 & 0.004 \\ 0.004 & 0.016 \end{pmatrix}, \Sigma'_2 = \begin{pmatrix} 0.023 & 0.004 \\ 0.004 & 0.017 \end{pmatrix},$

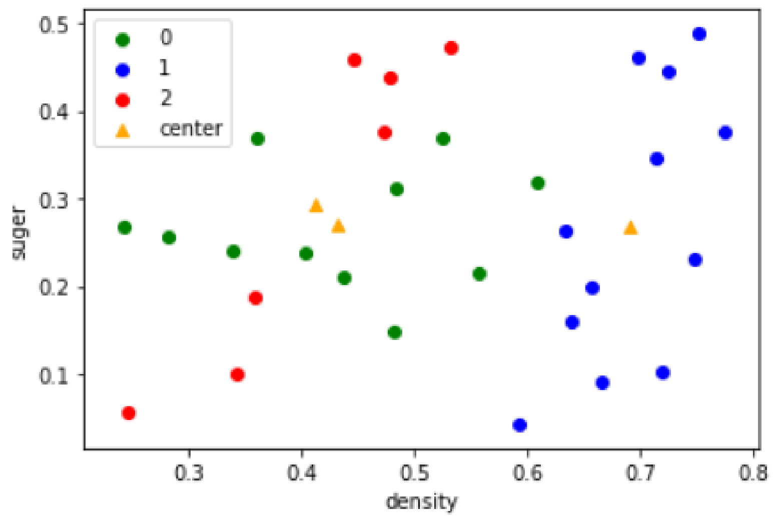
    $\Sigma'_3 = \begin{pmatrix} 0.034 & 0.005 \\ 0.005 & 0.016 \end{pmatrix}$
  - Iter 2 …

id	density	sugar content	id	density	sugar content
1	.697	.460	16	.593	.042
2	.774	.376	17	.719	.103
3	.634	.264	18	.359	.188
4	.608	.318	19	.339	.241
5	.556	.215	20	.282	.257
6	.403	.237	21	.748	.232
7	.481	.149	22	.714	.346
8	.437	.211	23	.483	.312
9	.666	.091	24	.478	.437
10	.243	.267	25	.525	.369
11	.245	.057	26	.751	.489
12	.343	.099	27	.532	.472
13	.639	.161	28	.473	.376
14	.657	.198	29	.725	.445
15	.360	.370	30	.446	.459

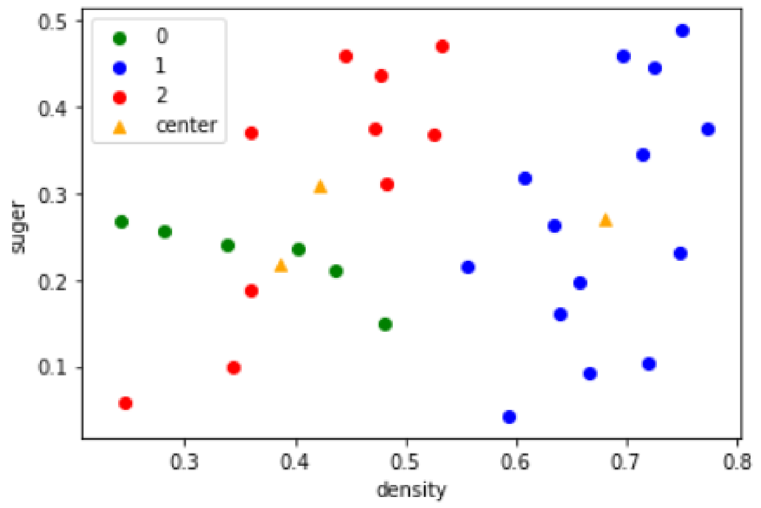
Example



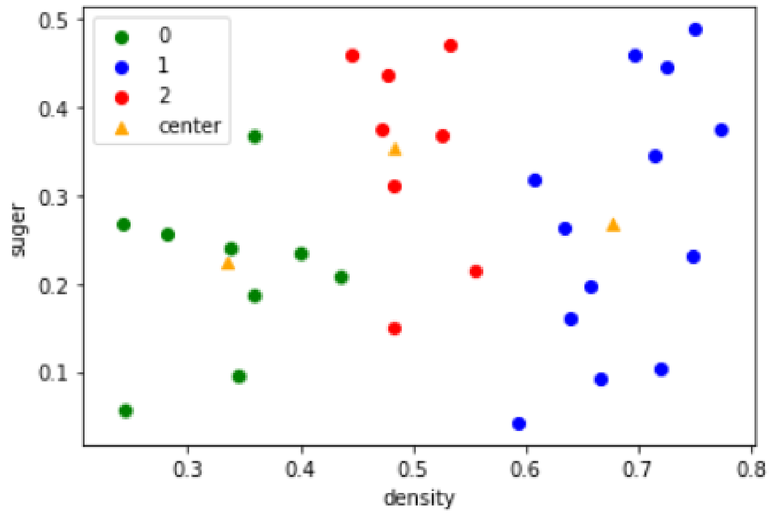
I=0



I=50



I=100

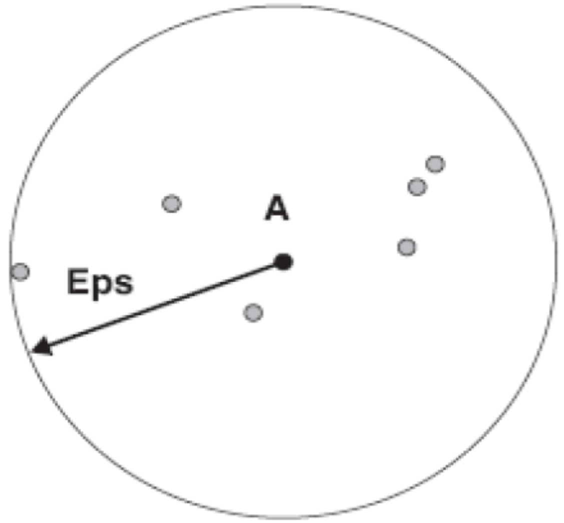


I=200

Density-based Clustering

DBSCAN

- DBSCAN (density-based spatial clustering of application with Noise)[1]
- **Density** = number of points within a specified radius (ε)

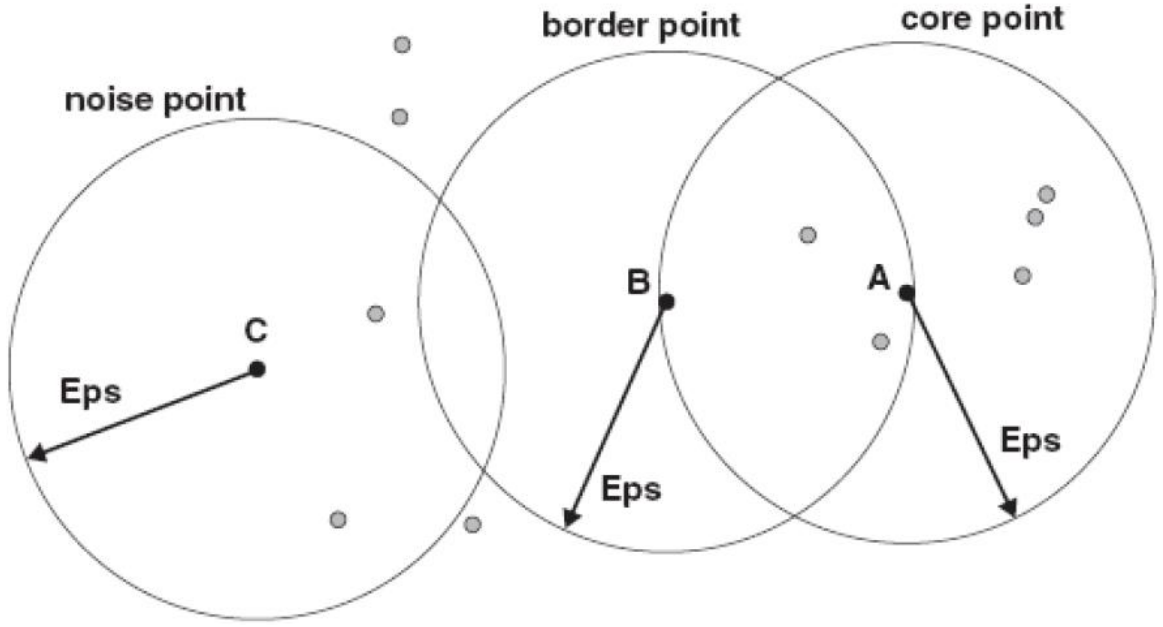


Density = 7 points

1Ester et al. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD). 1996.

Preliminary

- **Core point**: it has more than a specified number of points (MinPts) within ε
- **Border point**: it has fewer than MinPts within ε, but is in the neighborhood of a core point
- **Noise point (Outliers)**: any point that is not a core point or a border point.



Preliminary

- Point *q* is density-reachable from *p*: all points on the path connecting them must be core points
  - If p is a core point, then it forms a cluster together with all points that are reachable from it

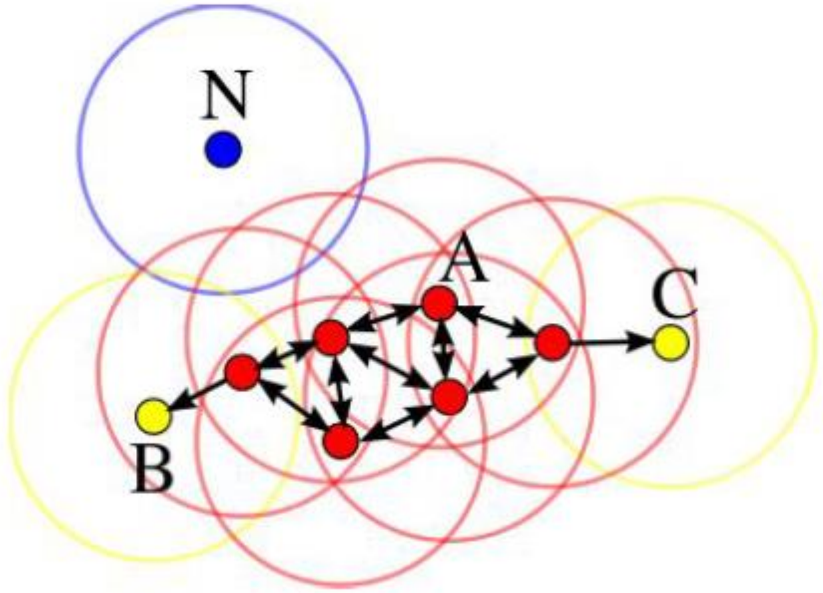
- Two points *p* and *q* are density-connected if there is a point *o* such that both *p* and *q* are reachable from *o*

- A cluster satisfies two properties:
  - All points within the cluster are mutually density-connected;
  - If a point is density-reachable from any point of the cluster, it is part of the cluster as well

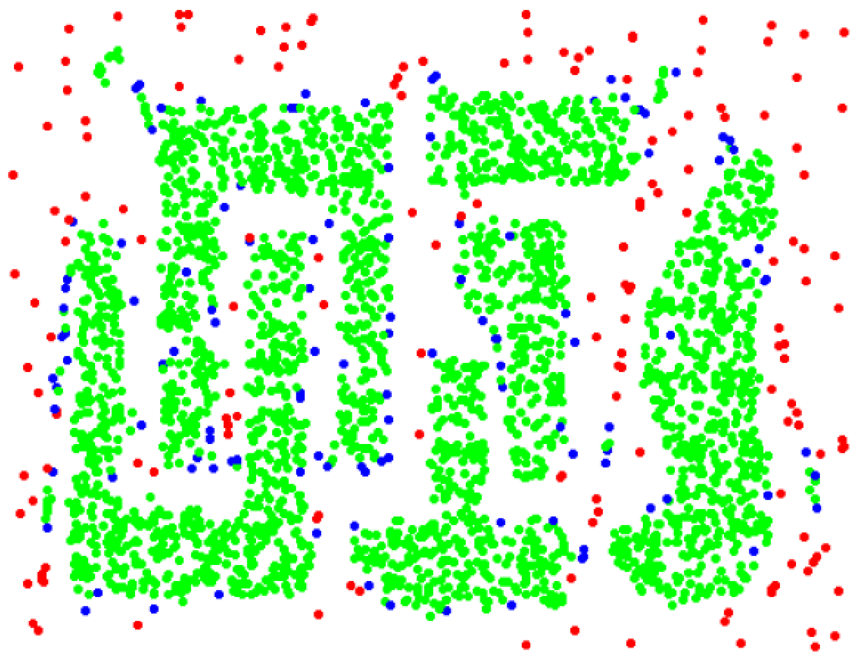
DBSCAN Algorithm

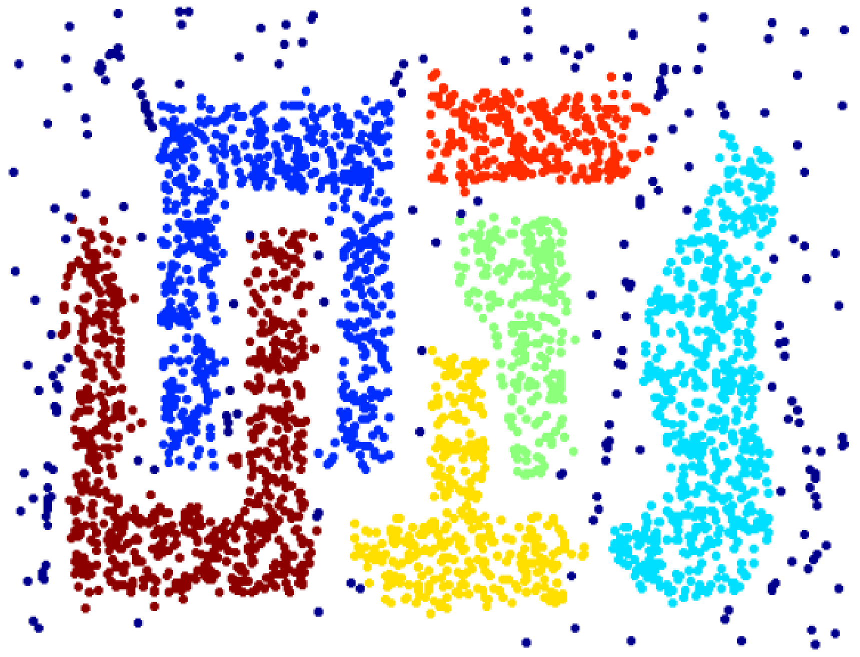
```
DBSCAN(D, eps, MinPts)
  C = 0
  for each unvisited point P in dataset D
    mark P as visited
    NeighborPts = regionQuery(P, eps)
    if sizeof(NeighborPts) < MinPts
      mark P as NOISE
    else
      C = next cluster
      expandCluster(P, NeighborPts, C, eps, MinPts)

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



DBSCAN Case


Point types: core, border and noise


Clusters