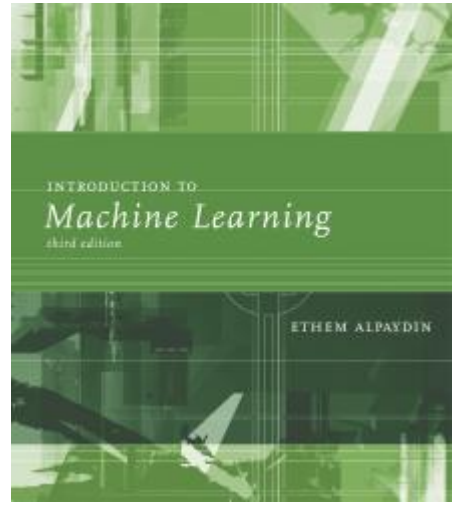


Clustering

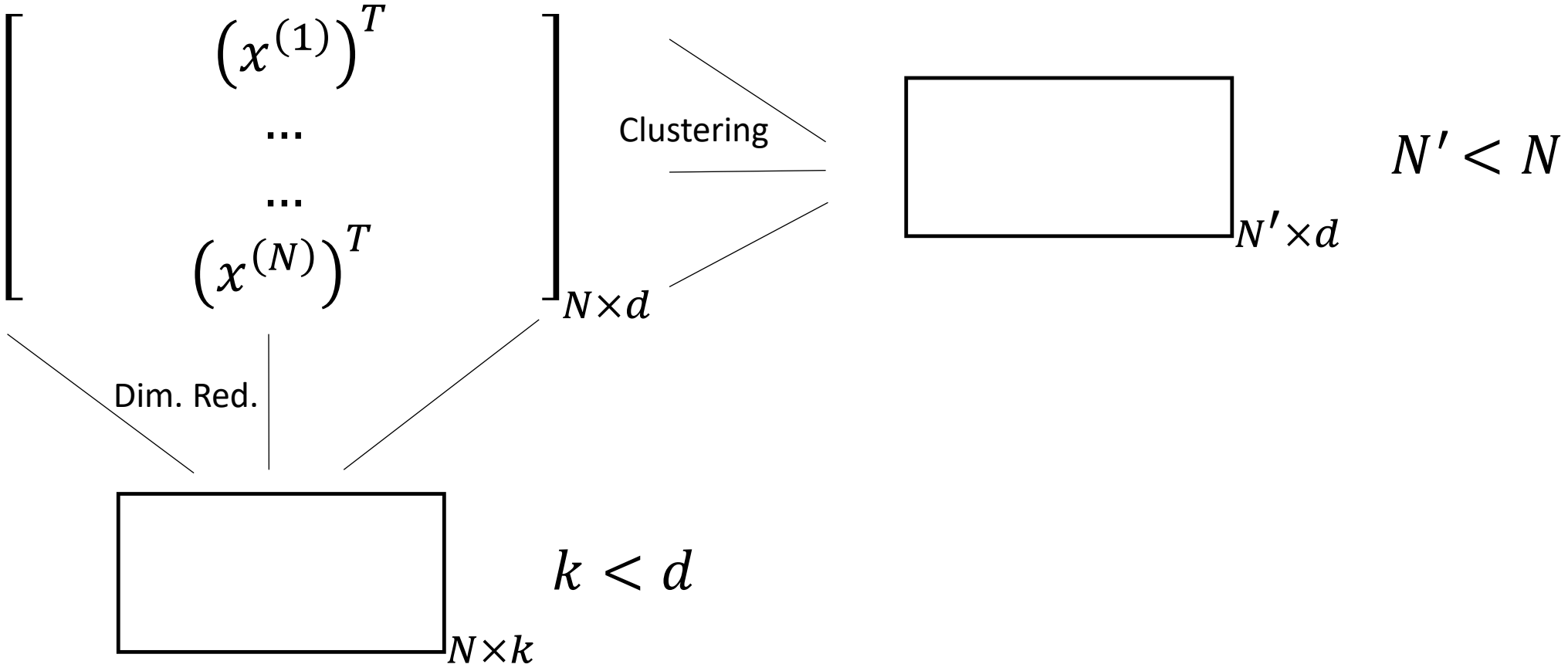
Lecture notes by Ethem Alpaydın
Introduction to Machine Learning (Boğaziçi Üniversitesi)

Lecture notes by Kevyn Collins-Thompson
Applied Machine Learning (Coursera)

Unsupervised Learning

- Unsupervised learning involves tasks that operate on datasets without labeled responses or target values.
- Instead, the goal is to capture interesting structure or information.

Dimensionality Reduction vs Clustering



Unsupervised Learning

Dimensionality Reduction

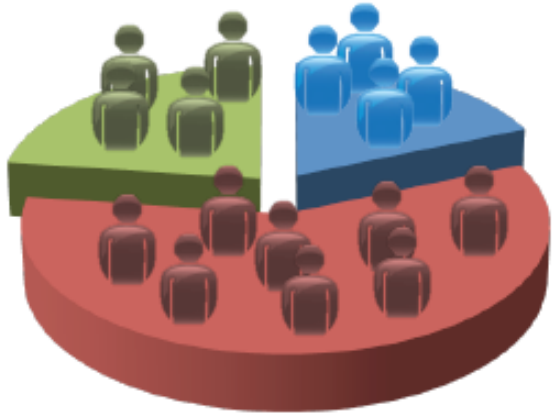
- Reduce the dimensionality of inputs
- Useful if there are correlation between input dimensionalities
- Find correlation between features and group features
- Reduce time and space complexity
- Easier to explain model
- Data visualization
- Speed up learning algorithm

Clustering

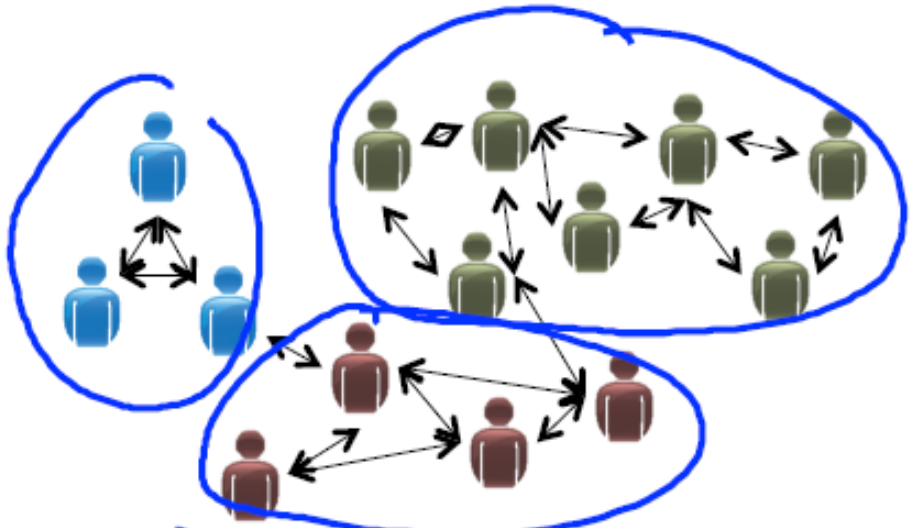
- Group all N points into k clusters
- Reduce all N points into k cluster centers
- Useful if there are groups of data in d dimensional space
- Find similarities between instances and group instances

Clustering

Applications of clustering



→ Market segmentation



→ Social network analysis

Semiparametric Density Estimation

- Parametric: Assume a single model for $p(\boldsymbol{x} \mid C_i)$
- Semiparametric: $p(\boldsymbol{x} \mid C_i)$ is a mixture of densities

  Multiple possible explanations/prototypes:

  Different handwriting styles, accents in speech
- Nonparametric: No model; data speaks for itself

Mixture Densities

$$p(\mathbf{x}) = \sum_{i=1}^{k} p(\mathbf{x} \mid G_i) P(G_i)$$

where G_i the components/groups/clusters,

$P(G_i)$ mixture proportions (priors),

$p(\boldsymbol{x} \mid G_i)$ component densities

Gaussian mixture where $p(\boldsymbol{x} \mid G_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ parameters $\Phi = \{P(G_i), \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{k}$

unlabeled sample $X=\{\boldsymbol{x}^t\}_t$ (unsupervised learning)

Classes vs. Clusters

Classes

- Supervised: X = $\{\boldsymbol{x}^t, \boldsymbol{r}^t\}_t$
- Classes C_i i=1,...,K

$$p(\mathbf{x}) = \sum_{i=1}^{K} p(\mathbf{x} \mid C_i) P(C_i)$$

where $p(\boldsymbol{x} \mid C_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$

- $\Phi = \{P(C_i), \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{K}$

$$\hat{P}(C_i) = \frac{\sum_t r_i^t}{N} \quad \mathbf{m}_i = \frac{\sum_t r_i^t \mathbf{x}^t}{\sum_t r_i^t}$$

$$\mathbf{S}_i = \frac{\sum_t r_i^t (\mathbf{x}^t - \mathbf{m}_i)(\mathbf{x}^t - \mathbf{m}_i)^T}{\sum_t r_i^t}$$

Clusters

- Unsupervised : X = $\{\boldsymbol{x}^t\}_t$
- Clusters G_i i=1,...,k

$$p(\mathbf{x}) = \sum_{i=1}^{k} p(\mathbf{x} \mid G_i) P(G_i)$$

where $p(\boldsymbol{x} \mid G_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$

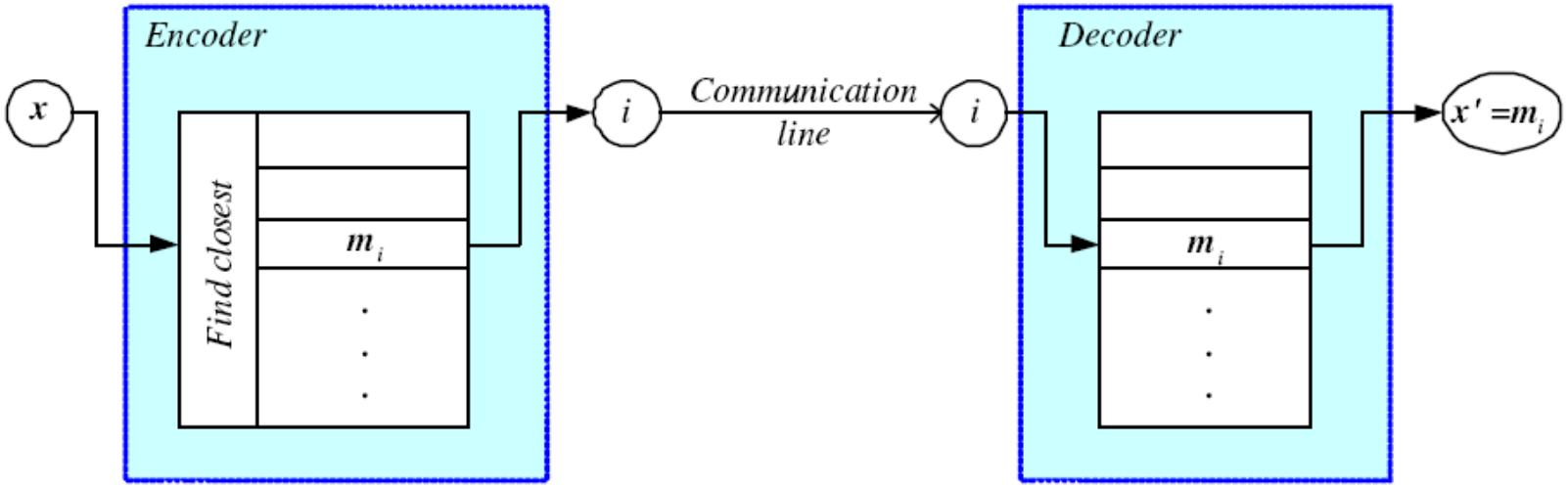
- $\Phi = \{P(G_i), \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{k}$
- *Label* $\boldsymbol{r}^t$ *?*

k-Means Clustering

- Find *k* reference vectors (prototypes/codebook vectors/codewords) which best represent data
- Reference vectors, $\boldsymbol{m}_j$, j =1,...,k
- Use nearest (most similar) reference:

$$\left\|\mathbf{x}^t - \mathbf{m}_i\right\| = \min_j \left\|\mathbf{x}^t - \mathbf{m}_j\right\|$$

Encoding / Decoding



k-Means Clustering

- How we can calculate $\boldsymbol{m}_i$
- Reconstruction error

$$E\left(\{\mathbf{m}_i\}_{i=1}^{k} | \mathcal{X}\right) = \sum_t \sum_i b_i^t \left\| \mathbf{x}^t - \mathbf{m}_i \right\|$$

$$b_i^t = \begin{cases} 1 & \text{if } \left\| \mathbf{x}^t - \mathbf{m}_i \right\| = \min_j \left\| \mathbf{x}^t - \mathbf{m}_j \right\| \\ 0 & \text{otherwise} \end{cases}$$

k-Means Clustering

Initialize $\boldsymbol{m}_i, i = 1, \ldots, k$, for example, to k random $\boldsymbol{x}^t$

Repeat

For all $\boldsymbol{x}^t \in \mathcal{X}$

$$b_i^t \leftarrow \begin{cases} 1 & \text{if } \|\boldsymbol{x}^t - \boldsymbol{m}_i\| = \min_j \|\boldsymbol{x}^t - \boldsymbol{m}_j\| \\ 0 & \text{otherwise} \end{cases}$$

For all $\boldsymbol{m}_i, i = 1, \ldots, k$

$$\boldsymbol{m}_i \leftarrow \sum_t b_i^t \boldsymbol{x}^t / \sum_t b_i^t$$

Until $\boldsymbol{m}_i$ converge

k-Means Clustering

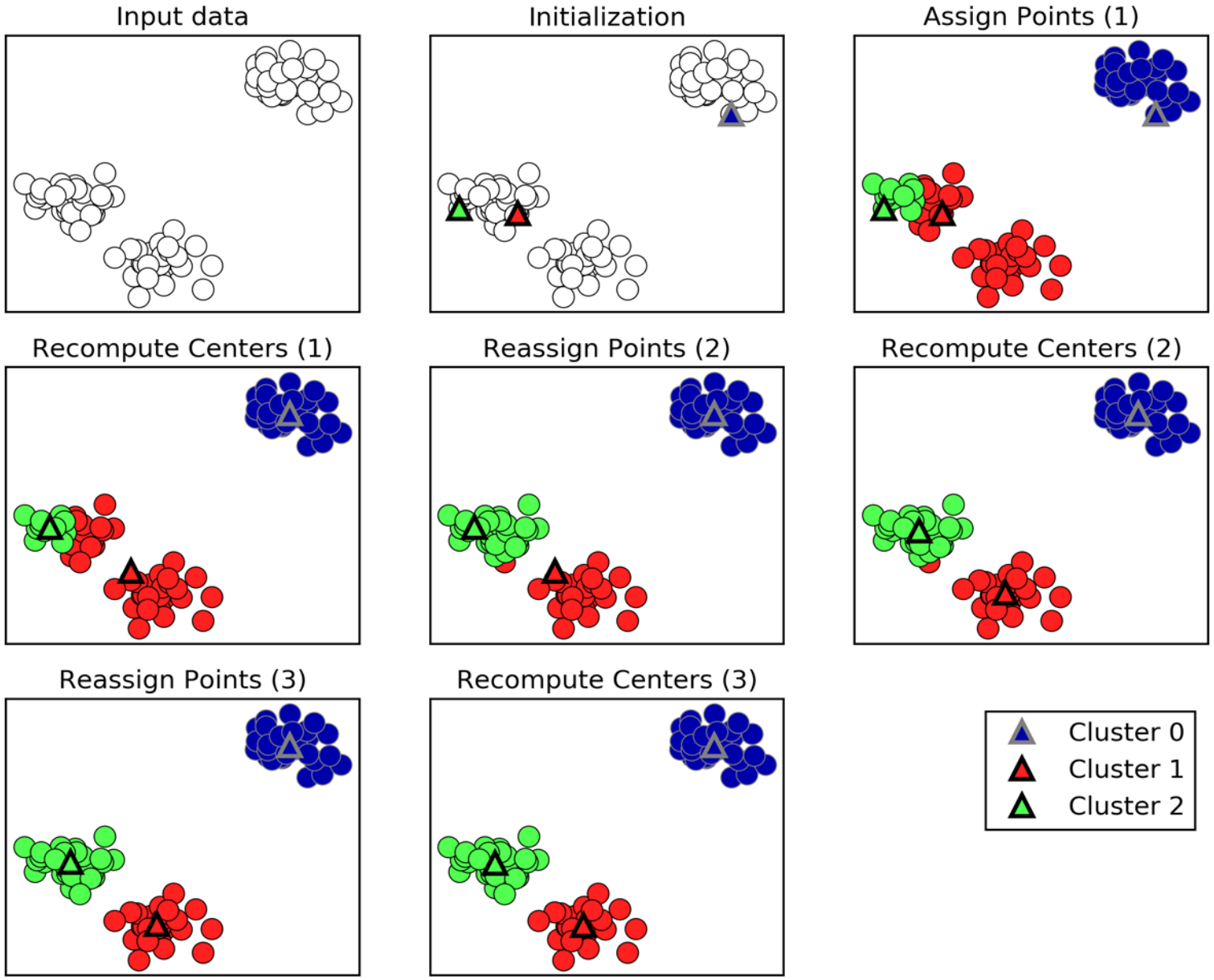
The k-means algorithm

Initialization Pick the number of clusters *k* you want to find. Then pick *k* *random* points to serve as an initial guess for the cluster centers.

Step A Assign each data point to the nearest cluster center.

Step B Update each cluster center by replacing it with the mean of all points assigned to that cluster (in step A).

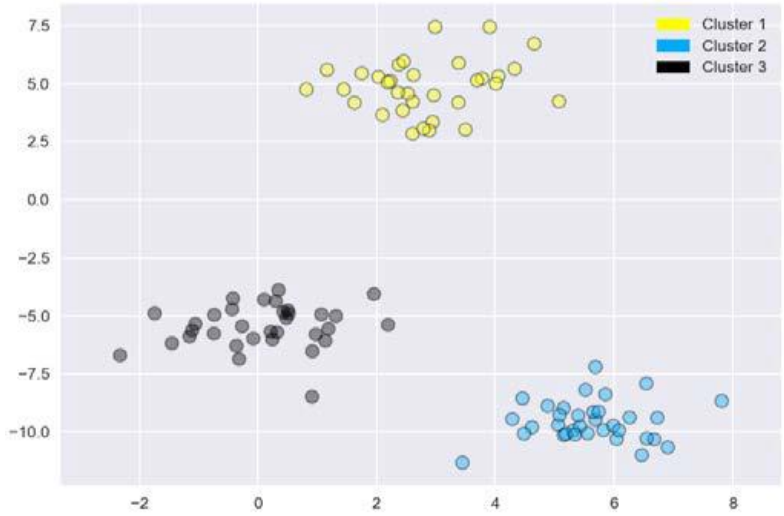
Repeat steps A and B until the centers converge to a stable solution.



Input data and three steps of the k-means algorithm

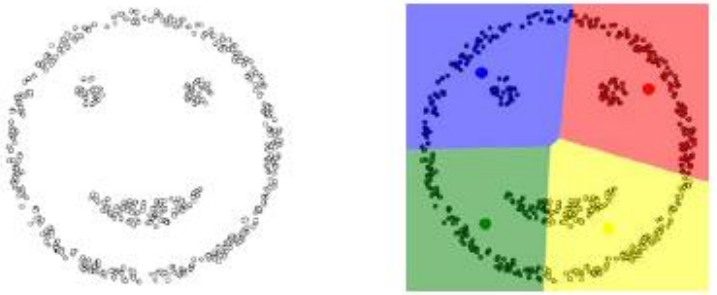
k-Means Clustering in Scikit-Learn

```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from adspy_shared_utilities import plot_labelled_scatter

X, y = make_blobs(random_state = 10)

kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)

plot_labelled_scatter(X, kmeans.labels_, ['Cluster 1', 'Cluster 2', 'Cluster 3'])
```



Limitations of k-means

- Works well for simple clusters that are same size, well-separated, globular shapes.
- Does not do well with irregular, complex clusters.
- Variants of k-means like k-medoids can work with categorical features.



K-means typically performs poorly with data having complex, irregular clusters.

Hierarchical Clustering

- Only use similarities of instances, without any other requirement on the data;
- The aim is to find groups such that instances in a group are more similar to each other than instances in different groups.
- This is the approach taken by *hierarchical clustering*.

Hierarchical Clustering

- This needs the use of a similarity, or equivalently a distance, measure defined between instances
- Generally Euclidean distance is used, where one has to make sure that all attributes have the same scale.
- This is a special case of the *Minkowksi distance* with $p = 2$:

$$d_m\left(\mathbf{x}^r, \mathbf{x}^s\right) = \left[\sum_{j=1}^{d}\left(x_j^r - x_j^s\right)^p\right]^{1/p}$$

Hierarchical Clustering

- *City-block distance* is easier to calculate:

$$d_{cb}\left(\mathbf{x}^r, \mathbf{x}^s\right) = \sum_{j=1}^{d} \left| x_j^r - x_j^s \right|$$

Agglomerative vs Divisive Clustering

- An *agglomerative clustering* algorithm starts with *N* groups, each initially containing one training instance, merging similar groups to form larger groups, until there is a single one.
- A *divisive clustering* algorithm goes in the other direction, starting with a single group and dividing large groups into smaller groups, until each group contains a single instance.

Agglomerative Clustering

- At each iteration of an agglomerative algorithm, we choose the two closest groups to merge

Agglomerative Clustering

- Distance between two groups G_i and G_j:
  - Single-link:

$$d(G_i, G_j) = \min_{\mathbf{x}^r \in G_i, \mathbf{x}^s \in G_j} d(\mathbf{x}^r, \mathbf{x}^s)$$

  - Complete-link:

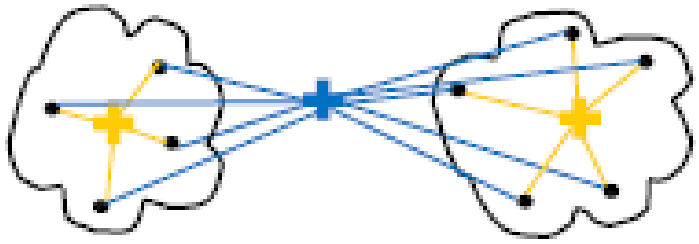
$$d(G_i, G_j) = \max_{\mathbf{x}^r \in G_i, \mathbf{x}^s \in G_j} d(\mathbf{x}^r, \mathbf{x}^s)$$

  - Average-link, centroid

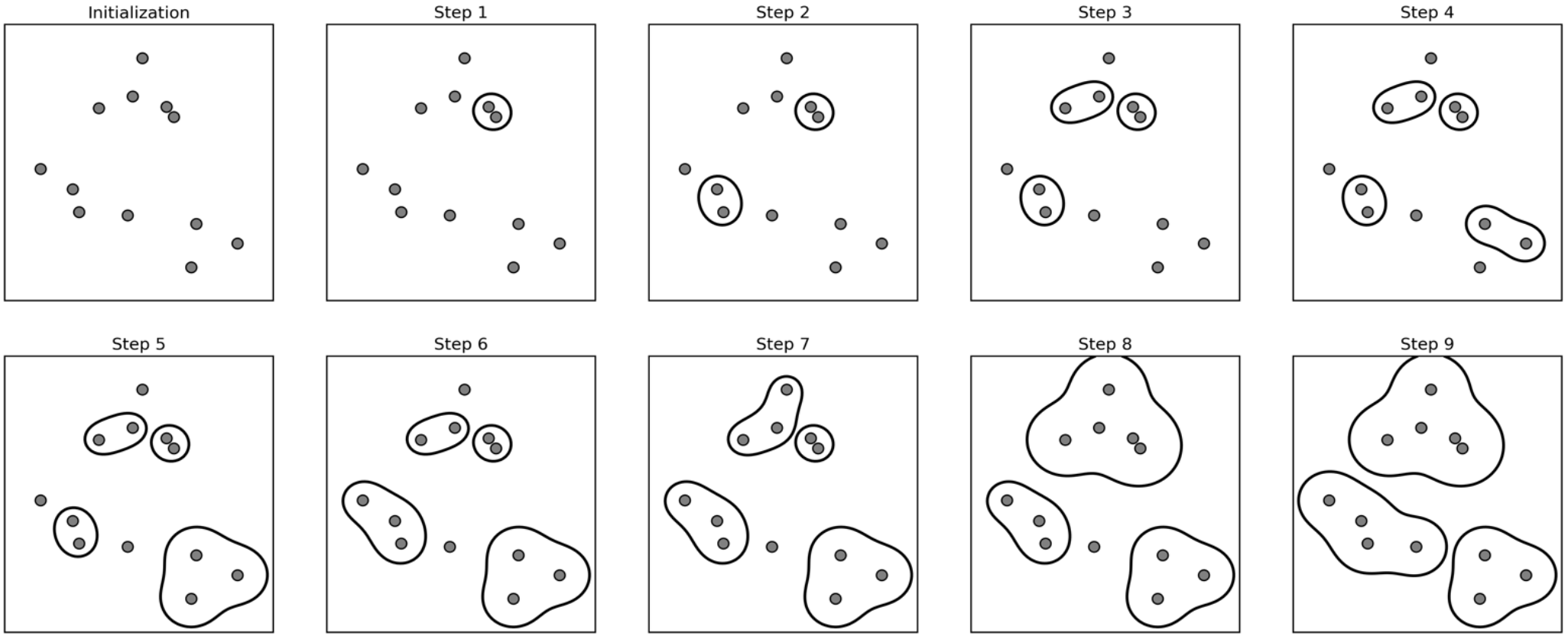
$$d(G_i, G_j) = \text{ave}_{\mathbf{x}^r \in G_i, \mathbf{x}^s \in G_j} d(\mathbf{x}^r, \mathbf{x}^s)$$

Agglomerative Clustering

- Distance between two groups G_i and G_j:
  - Ward's method: picks the two clusters to merge such that the variance within all clusters increases the least.
  - This often leads to clusters that are relatively equally sized.

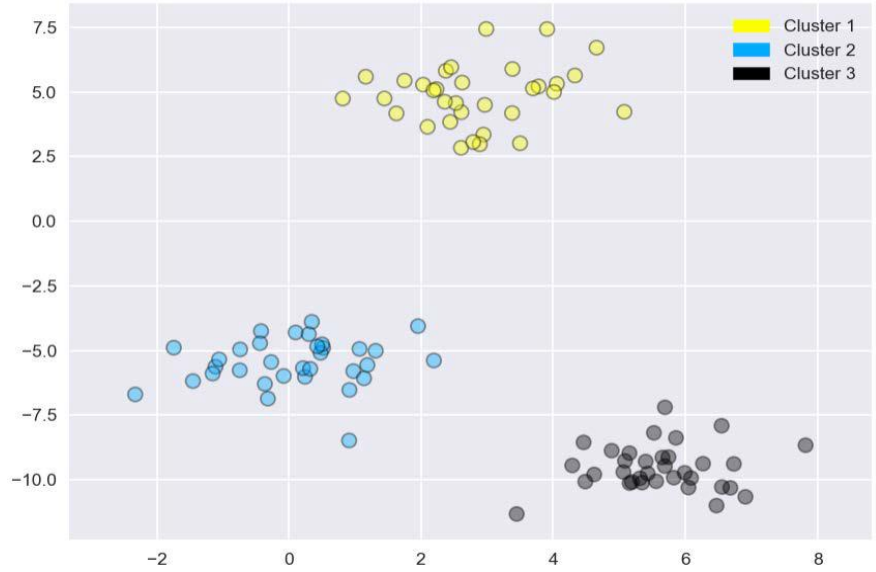


Agglomerative Clustering



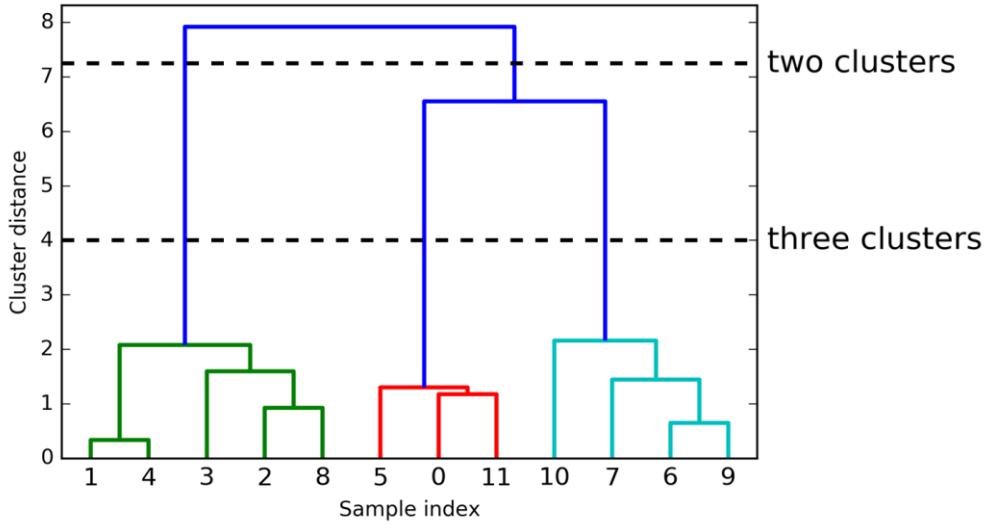
Agglomerative Clustering in Scikit-Learn

```
from sklearn.datasets import make_blobs
from sklearn.cluster import AgglomerativeClustering
from adspy_shared_utilities import plot_labelled_scatter

X, y = make_blobs(random_state = 10)

cls = AgglomerativeClustering(n_clusters = 3)
cls_assignment = cls.fit_predict(X)

X, y = make_blobs(random_state = 10)
plot_labelled_scatter(X, cls_assignment,
        ['Cluster 1', 'Cluster 2', 'Cluster 3'])
```



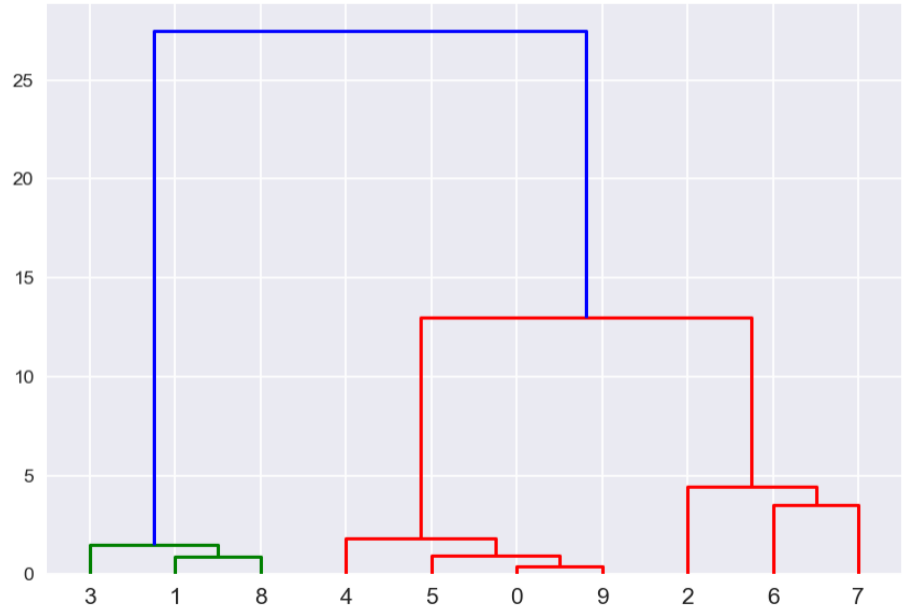
Dendrogram

- Once an agglomerative method is run, the result is generally drawn as a hierarchical structure known as the *dendrogram*.
- This is a tree where leaves correspond to instances, which are grouped in the order in which they are merged.



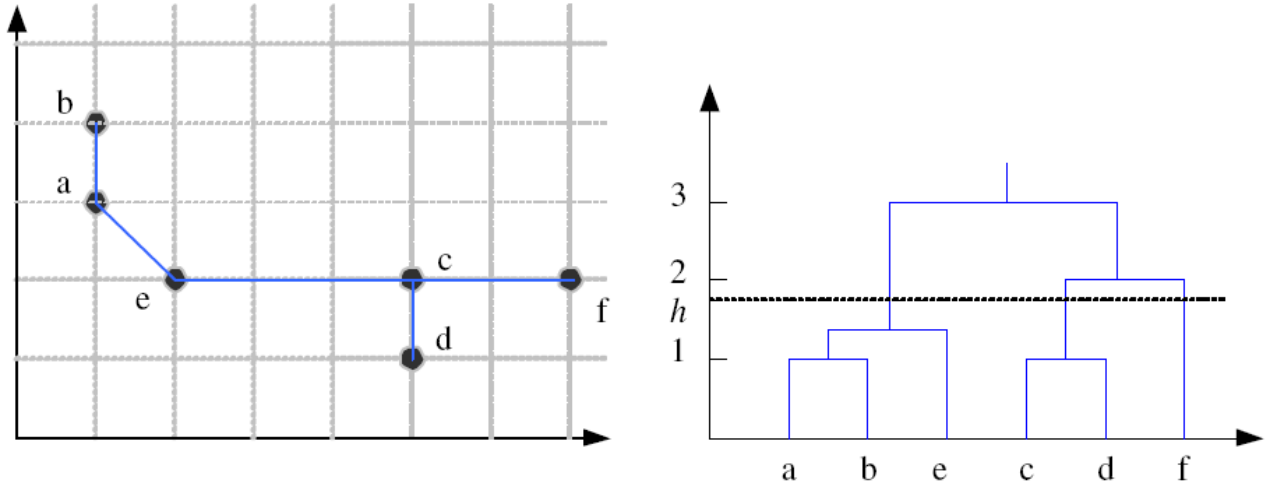
Dendrogram in Scikit-Learn

```
from scipy.cluster.hierarchy import ward, dendrogram
from sklearn.datasets import make_blobs
from sklearn.cluster import AgglomerativeClustering

X, y = make_blobs(random_state = 10, n_samples = 10)

plt.figure()
dendrogram(ward(X))
plt.show()
```



Example: Single-Link Clustering

- A two-dimensional dataset and the dendrogram showing the result of single-link clustering is shown.
- Note that leaves of the tree are ordered so that no branches cross.
- The tree is then intersected at a desired value of *h* to get the clusters.



Example

- Dissimilarities, based on the Jaccard index
- Choose one of the most popular ones, called the *maximum*, or *complete linkage*, method: the dissimilarity between the merged pair and the others will be the maximum of the pair of dissimilarities in each case.

Example

- Dissimilarities, based on the Jaccard index

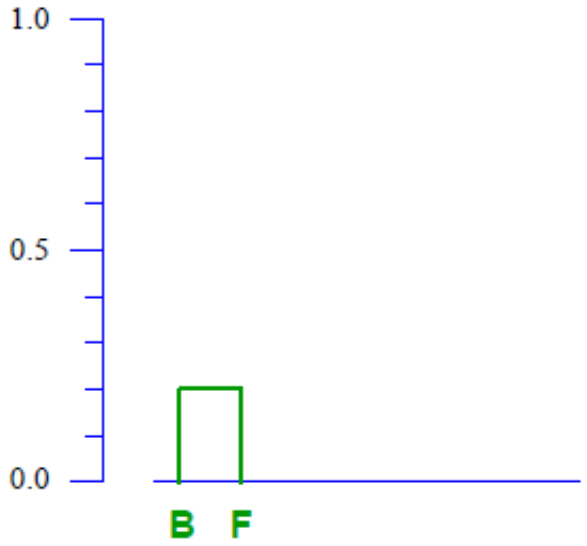
samples	A	B	C	D	E	F	G
A	0	0.5000	0.4286	1.0000	0.2500	0.6250	0.3750
B	0.5000	0	0.7143	0.8333	0.6667	0.2000	0.7778
C	0.4286	0.7143	0	1.0000	0.4286	0.6667	0.3333
D	1.0000	0.8333	1.0000	0	1.0000	0.8000	0.8571
E	0.2500	0.6667	0.4286	1.0000	0	0.7778	0.3750
F	0.6250	0.2000	0.6667	0.8000	0.7778	0	0.7500
G	0.3750	0.7778	0.3333	0.8571	0.3750	0.7500	0

Example

- The first step in the hierarchical clustering process is to look for the pair of samples that are the most similar, that is are the closest in the sense of having the lowest dissimilarity – this is the pair B and F, with dissimilarity equal to 0.2000

samples	A	B	C	D	E	F	G
A	0	0.5000	0.4286	1.0000	0.2500	0.6250	0.3750
B	0.5000	0	0.7143	0.8333	0.6667	0.2000	0.7778
C	0.4286	0.7143	0	1.0000	0.4286	0.6667	0.3333
D	1.0000	0.8333	1.0000	0	1.0000	0.8000	0.8571
E	0.2500	0.6667	0.4286	1.0000	0	0.7778	0.3750
F	0.6250	0.2000	0.6667	0.8000	0.7778	0	0.7500
G	0.3750	0.7778	0.3333	0.8571	0.3750	0.7500	0

Example



samples	A	B	C	D	E	F	G
A	0	0.5000	0.4286	1.0000	0.2500	0.6250	0.3750
B	0.5000	0	0.7143	0.8333	0.6667	0.2000	0.7778
C	0.4286	0.7143	0	1.0000	0.4286	0.6667	0.3333
D	1.0000	0.8333	1.0000	0	1.0000	0.8000	0.8571
E	0.2500	0.6667	0.4286	1.0000	0	0.7778	0.3750
F	0.6250	0.2000	0.6667	0.8000	0.7778	0	0.7500
G	0.3750	0.7778	0.3333	0.8571	0.3750	0.7500	0

How to calculated the dissimilarity between the merged pair (B,F) and the other samples

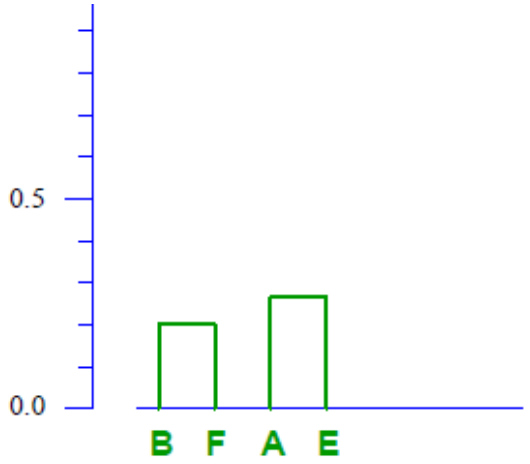
samples	A	(B,F)	C	D	E	G
A	0	0.6250	0.4286	1.0000	0.2500	0.3750
(B,F)	0.6250	0	0.7143	0.8333	0.7778	0.7778
C	0.4286	0.7143	0	1.0000	0.4286	0.3333
D	1.0000	0.8333	1.0000	0	1.0000	0.8571
E	0.2500	0.7778	0.4286	1.0000	0	0.3750
G	0.3750	0.7778	0.3333	0.8571	0.3750	0

Example

samples	A	B	C	D	E	F	G
A	0	0.5000	0.4286	1.0000	0.2500	0.6250	0.3750
B	0.5000	0	0.7143	0.8333	0.6667	0.2000	0.7778
C	0.4286	0.7143	0	1.0000	0.4286	0.6667	0.3333
D	1.0000	0.8333	1.0000	0	1.0000	0.8000	0.8571
E	0.2500	0.6667	0.4286	1.0000	0	0.7778	0.3750
F	0.6250	0.2000	0.6667	0.8000	0.7778	0	0.7500
G	0.3750	0.7778	0.3333	0.8571	0.3750	0.7500	0

samples	A	(B,F)	C	D	E	G
A	0	0.6250	0.4286	1.0000	0.2500	0.3750
(B,F)	0.6250	0	0.7143	0.8333	0.7778	0.7778
C	0.4286	0.7143	0	1.0000	0.4286	0.3333
D	1.0000	0.8333	1.0000	0	1.0000	0.8571
E	0.2500	0.7778	0.4286	1.0000	0	0.3750
G	0.3750	0.7778	0.3333	0.8571	0.3750	0

find the smallest dissimilarity: A and E



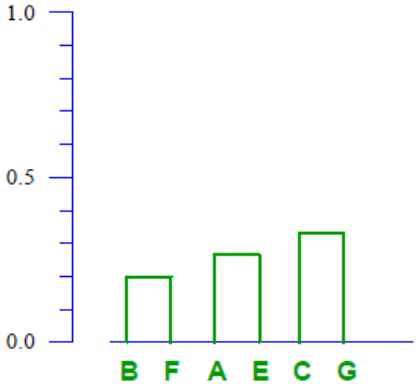
samples	(A,E)	(B,F)	C	D	G
(A,E)	0	0.7778	0.4286	1.0000	0.3750
(B,F)	0.7778	0	0.7143	0.8333	0.7778
C	0.4286	0.7143	0	1.0000	0.3333
D	1.0000	0.8333	1.0000	0	0.8571
G	0.3750	0.7778	0.3333	0.8571	0

Example

samples	A	B	C	D	E	F	G
A	0	0.5000	0.4286	1.0000	0.2500	0.6250	0.3750
B	0.5000	0	0.7143	0.8333	0.6667	0.2000	0.7778
C	0.4286	0.7143	0	1.0000	0.4286	0.6667	0.3333
D	1.0000	0.8333	1.0000	0	1.0000	0.8000	0.8571
E	0.2500	0.6667	0.4286	1.0000	0	0.7778	0.3750
F	0.6250	0.2000	0.6667	0.8000	0.7778	0	0.7500
G	0.3750	0.7778	0.3333	0.8571	0.3750	0.7500	0

samples	(A,E)	(B,F)	C	D	G
(A,E)	0	0.7778	0.4286	1.0000	0.3750
(B,F)	0.7778	0	0.7143	0.8333	0.7778
C	0.4286	0.7143	0	1.0000	0.3333
D	1.0000	0.8333	1.0000	0	0.8571
G	0.3750	0.7778	0.3333	0.8571	0

find the smallest dissimilarity: C and G

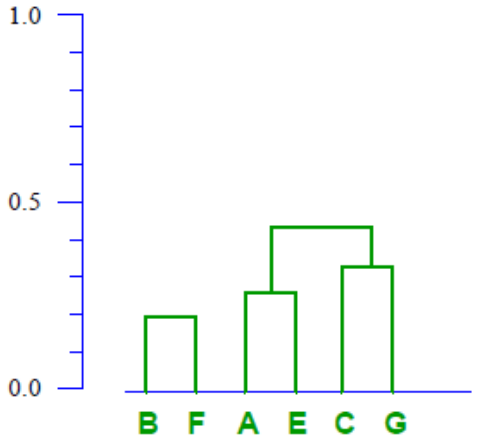


samples	(A,E)	(B,F)	(C,G)	D
(A,E)	0	0.7778	0.4286	1.0000
(B,F)	0.7778	0	0.7778	0.8333
(C,G)	0.4286	0.7778	0	1.0000
D	1.0000	0.8333	1.0000	0

Example

samples	A	B	C	D	E	F	G
A	0	0.5000	0.4286	1.0000	0.2500	0.6250	0.3750
B	0.5000	0	0.7143	0.8333	0.6667	0.2000	0.7778
C	0.4286	0.7143	0	1.0000	0.4286	0.6667	0.3333
D	1.0000	0.8333	1.0000	0	1.0000	0.8000	0.8571
E	0.2500	0.6667	0.4286	1.0000	0	0.7778	0.3750
F	0.6250	0.2000	0.6667	0.8000	0.7778	0	0.7500
G	0.3750	0.7778	0.3333	0.8571	0.3750	0.7500	0

samples	(A,E)	(B,F)	(C,G)	D
(A,E)	0	0.7778	0.4286	1.0000
(B,F)	0.7778	0	0.7778	0.8333
(C,G)	0.4286	0.7778	0	1.0000
D	1.0000	0.8333	1.0000	0



find the smallest dissimilarity: (A,E) and (C,G)

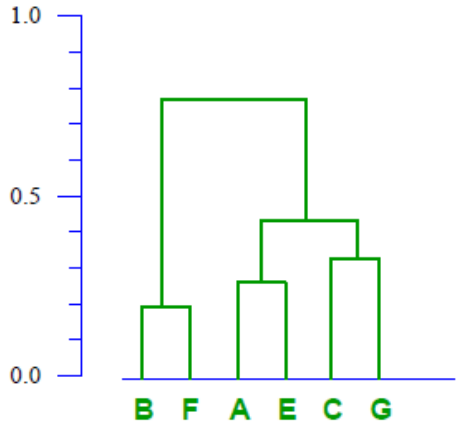
samples	(A,E,C,G)	(B,F)	D
(A,E,C,G)	0	0.7778	1.0000
(B,F)	0.7778	0	0.8333
D	1.0000	0.8333	0

Example

samples	A	B	C	D	E	F	G
A	0	0.5000	0.4286	1.0000	0.2500	0.6250	0.3750
B	0.5000	0	0.7143	0.8333	0.6667	0.2000	0.7778
C	0.4286	0.7143	0	1.0000	0.4286	0.6667	0.3333
D	1.0000	0.8333	1.0000	0	1.0000	0.8000	0.8571
E	0.2500	0.6667	0.4286	1.0000	0	0.7778	0.3750
F	0.6250	0.2000	0.6667	0.8000	0.7778	0	0.7500
G	0.3750	0.7778	0.3333	0.8571	0.3750	0.7500	0

samples	(A,E,C,G)	(B,F)	D
(A,E,C,G)	0	0.7778	1.0000
(B,F)	0.7778	0	0.8333
D	1.0000	0.8333	0

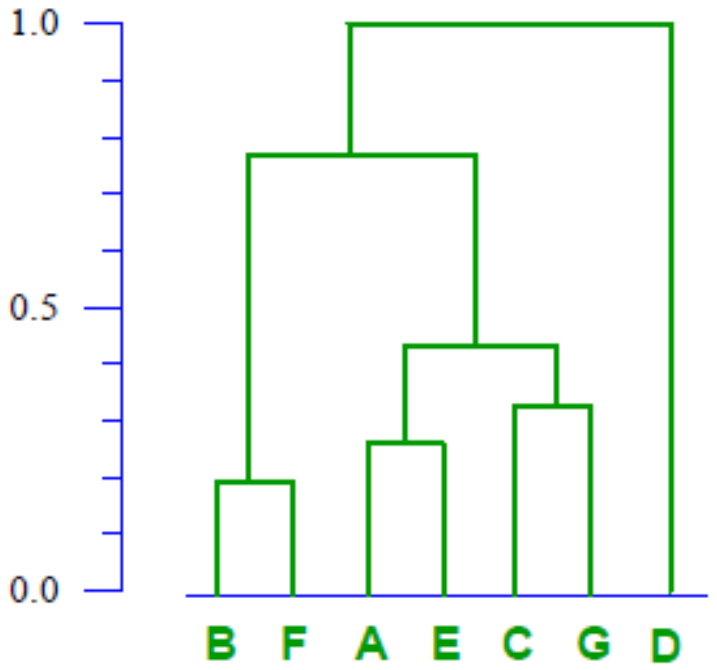
find the smallest dissimilarity: (A,E,C,G) and (B,F)



samples	(A,E,C,G,B,F)	D
(A,E,C,G,B,F)	0	1.0000
D	1.0000	0

Example

samples	(A,E,C,G,B,F)	D
(A,E,C,G,B,F)	0	1.0000
D	1.0000	0



Choosing k

- In some applications such as color quantization, *k* is defined by the application.
- Plotting the data in two dimensions using PCA may be used in uncovering the structure of data and the number of clusters in the data.

Choosing k

- Depending on what type of clustering method we use, we can plot the reconstruction error or log likelihood as a function of *k* and look for the “elbow.”
- After a large enough *k*, the algorithm will start dividing groups, in which case there will not be a large decrease in the reconstruction error or large increase in the log likelihood.
- Similarly, in hierarchical clustering, by looking at the differences between levels in the tree, we can decide on a good split.