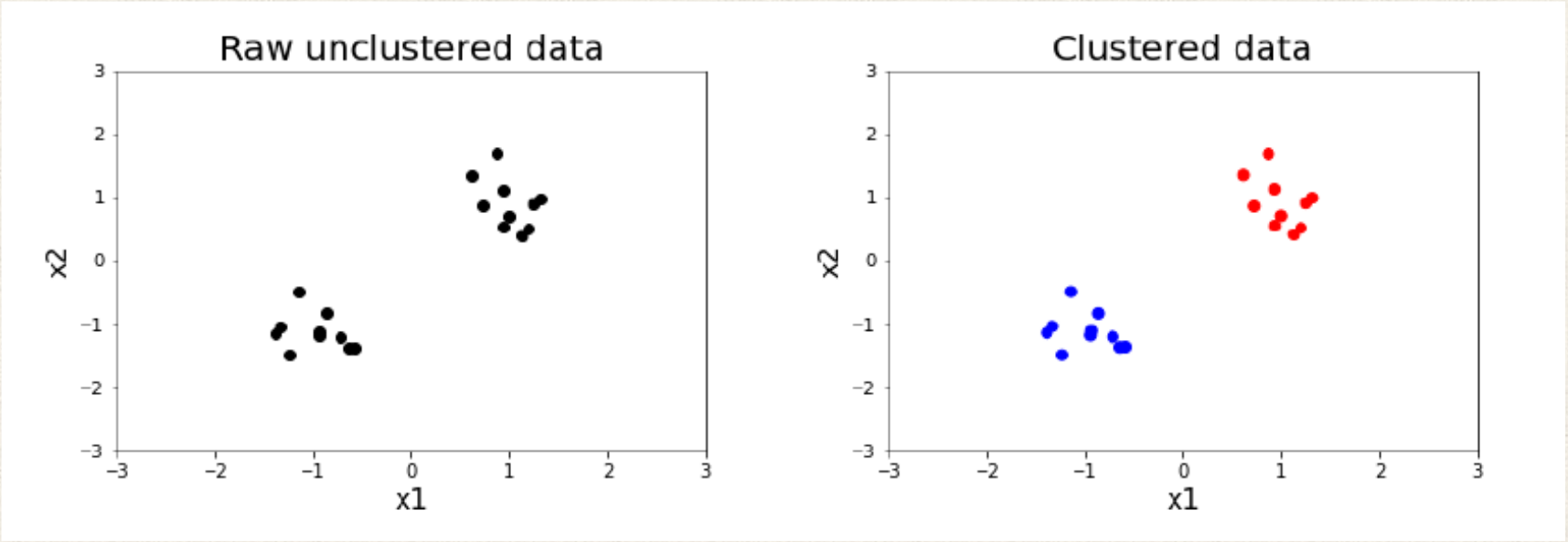
Unsupervised Learning

- What is clustering?
- K-means method for clustering
- Density based clustering
- Hierarchical Clustering
- Principal Components Analysis (PCA)

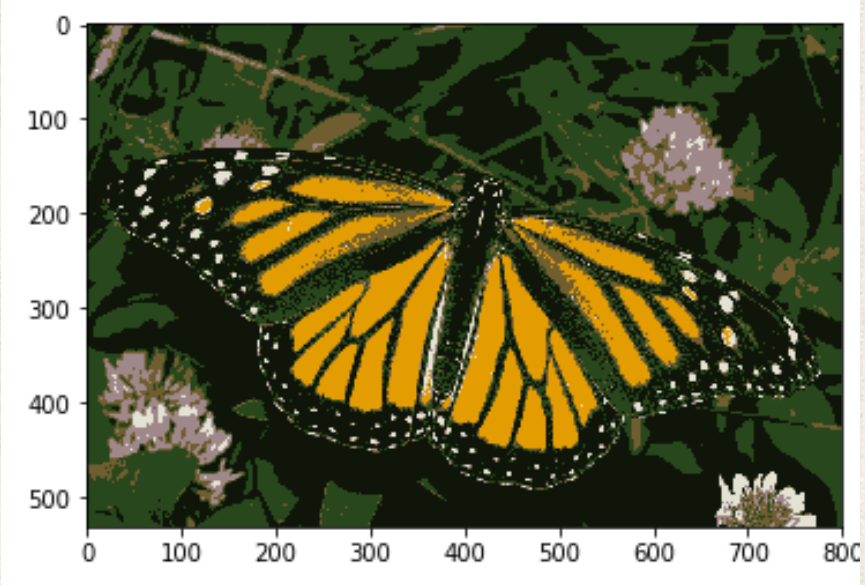
Clustering



Clustering

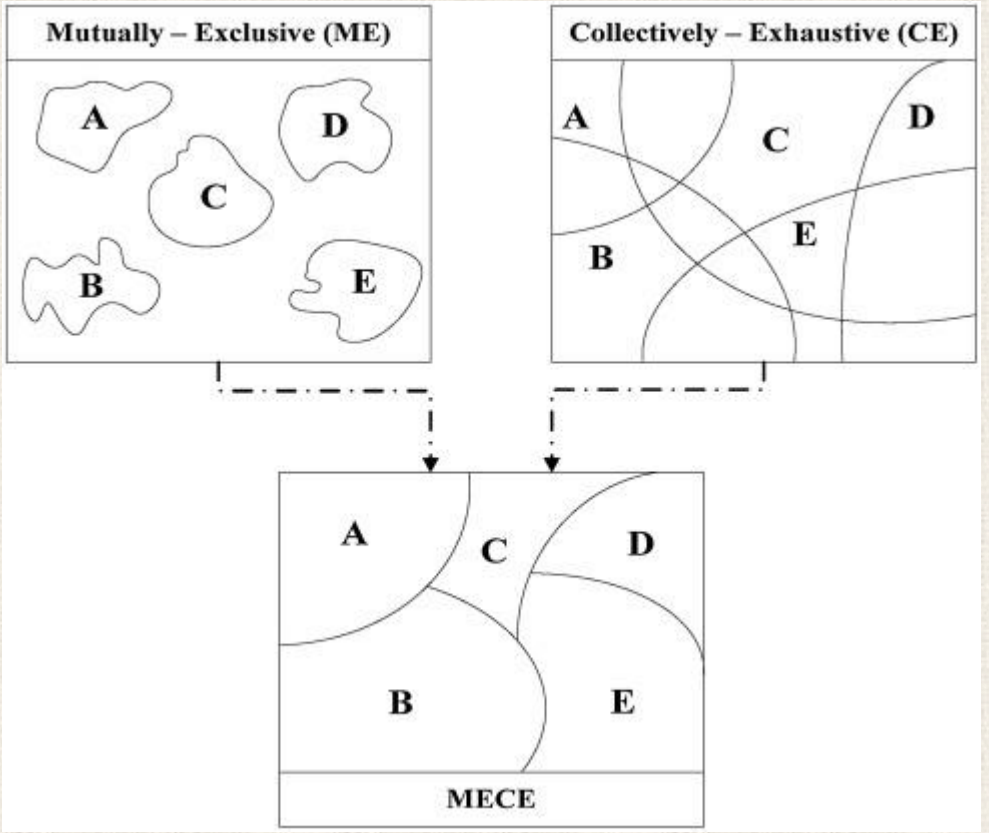
- In the problem of clustering, we are given a dataset comprised only of input features ***without*** class labels.
- ***Unsupervised*** Machine Learning Problem.
- ***Clustering*** is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).
- ***Clustering*** is most often used in ***exploratory data visualization***.
- ***Clustering*** is also used for outlier detection.

Clustering (image segmentation)



Cluster Assignment

- Let X denote the set of N data points $\boldsymbol{x}_i \in \mathcal{R}^d$.
- A ***cluster assignment*** is a partition $C_1, \ldots, C_K \subseteq X$ such that the sets C_k are disjoint and $X = C_1 \cup \cdots \cup C_K$ (***Mutually Exclusive*** *and* ***Exhaustive***).
- A data point $\boldsymbol{x} \in X$ is said to belong to cluster k if it is in C_k.



Cluster Assignment

- What are the desired properties of a good clustering assignment:
  - High ***intra-cluster*** similarity: points within a given cluster are very similar
  - Low ***inter-cluster*** similarity: points in different clusters are not very similar
- How is similarity defined?
  - The most used definition of similarity is using distance: two points in $\mathcal{R}^d$ are similar if their $L2$ distance is small, and dissimilar otherwise.
  - Other kinds of distance definitions can also be used (e.g., those used in KNN).

Position (Centroid) Based Clustering

- Each cluster C_k is represented by a single point $\boldsymbol{c}_k \in \mathcal{R}^d$ (*centroid*) in the input space and choose a cluster assignment such that the total distance of each point to its assigned centroid is minimized. That is:

$$\underset{\{C_k\},\{\boldsymbol{c}_k\}: X = C_1 \cup \cdots \cup C_K}{argmin} \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in \boldsymbol{C_k}} \|\boldsymbol{x} - \boldsymbol{c}_k\|^2$$

- This is a ***NP hard*** problem. Hence solving it in large scale exactly is intractable.

(it is suspected that there are no polynomial time algorithms for NP-hard problems)

K-means Clustering

We can produce a simple ***suboptimal*** algorithm to compute a candidate solution base on the following two thoughts: (K-means clustering)

- If we know the centroids $\boldsymbol{c}_k$, to choose the cluster assignment $C_1, \dots, C_K$ that minimizes the sum of squared distances to the centroids, we simply assign each data points $\boldsymbol{x}$ to the cluster represented by its closest centroid. That is, we assign $\boldsymbol{x}$ to :

$$\min_k \|\boldsymbol{x} - \boldsymbol{c}_k\|^2$$

- If we already have a cluster assignment $C_1, \dots, C_K$, we can choose the centroid of each set as the *mean* of all the data points in that set:

$$\boldsymbol{c}_k = \frac{1}{|C_k|} \sum_{\boldsymbol{x} \in C_k} \boldsymbol{x}$$

K-means Clustering

The Lloyd's Algorithm:

Initialize $\boldsymbol{c}_k, k = 1, \dots, K$

Do

Update $C_1, \dots, C_K$ given the $\boldsymbol{c}_k$ by assigning each $\boldsymbol{x} \in X$ to the cluster represented by its nearest centroid.

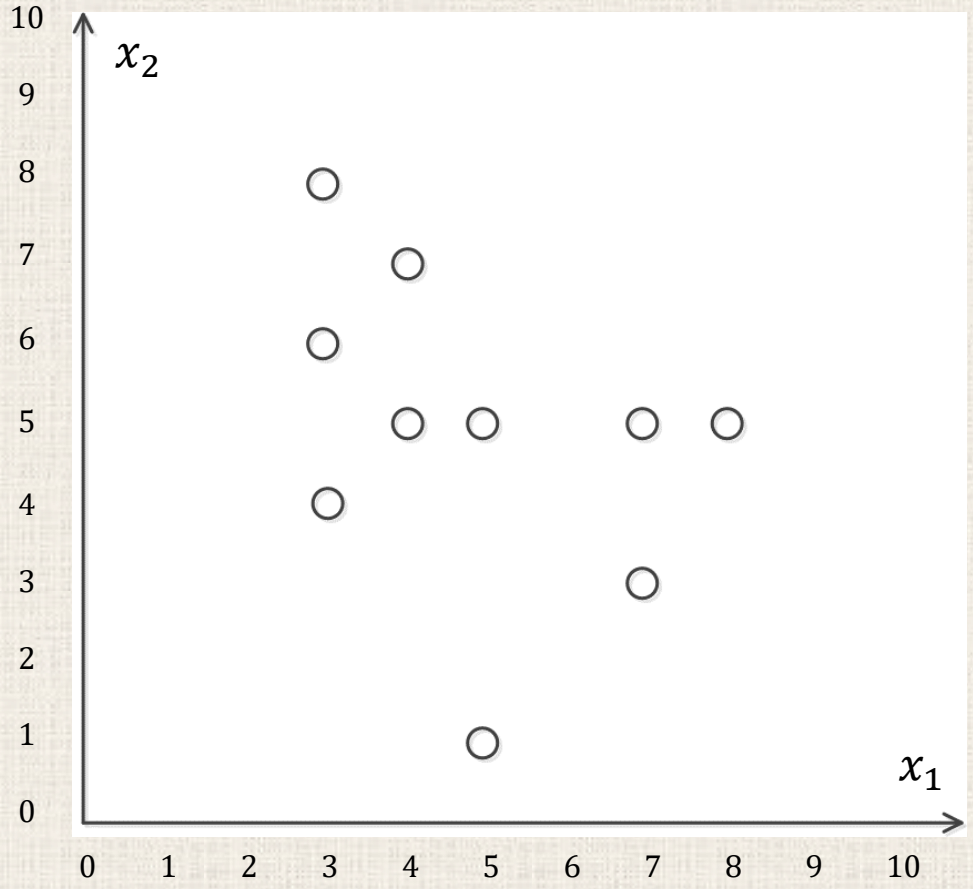
Update centroids $\boldsymbol{c}_k$ of given $C_1, \dots, C_K$ by $\quad \boldsymbol{c}_k = \frac{1}{|C_k|} \sum_{\boldsymbol{x} \in C_k} \boldsymbol{x}$

Until K-means objective converges

Return $C_1, \dots, C_K$

K-Means Clustering Example

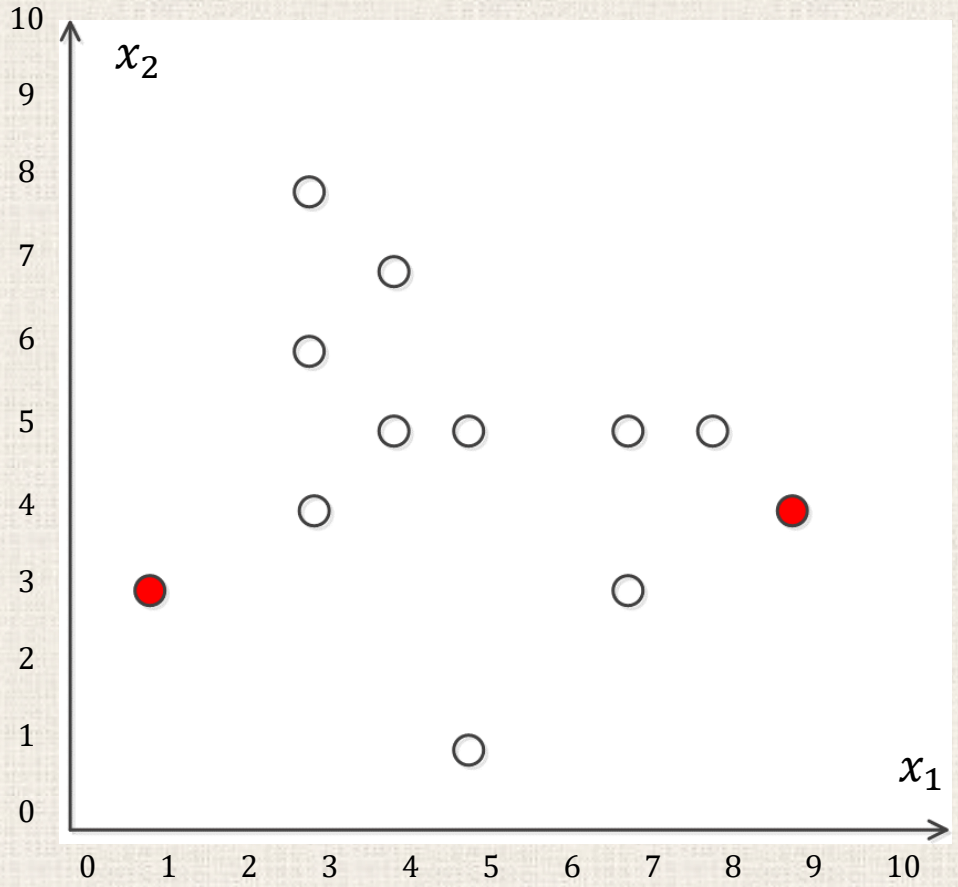
#	x_1	x_2
1	3	8
2	4	7
3	3	6
4	4	5
5	5	5
6	7	5
7	8	5
8	3	4
9	7	3
10	5	1



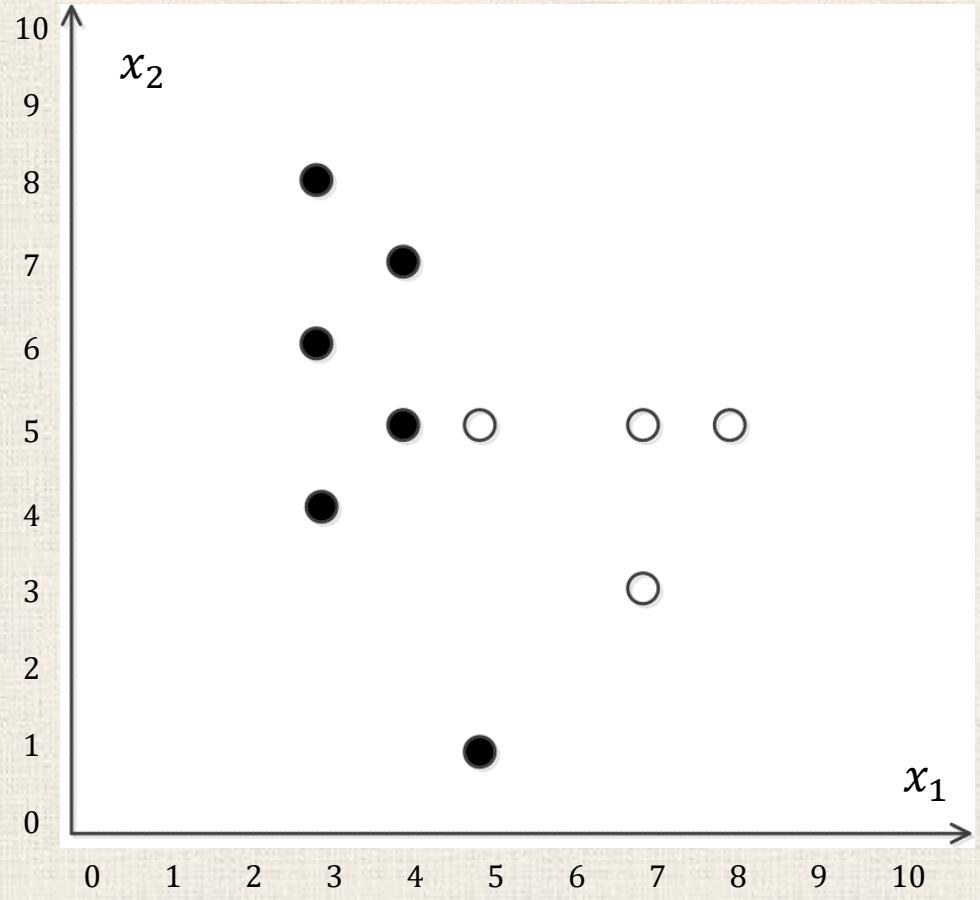
Let $K = 2$, the initial cluster centroids are randomly chosen as: $c_1 = (1,3),\ c_2 = (9,4)$

Let's assign the points to the clusters represented by these two centroids:

#	x_1	x_2	d_1	d_2	cluster
1	3	8	5.39	7.21	1
2	4	7	5	5.83	1
3	3	6	3.61	6.32	1
4	4	5	3.61	5.10	1
5	5	5	4.24	4.12	2
6	7	5	6.32	2.24	2
7	8	5	7.28	1.41	2
8	3	4	2.24	6	1
9	7	3	6	2.24	2
10	5	1	4.24	5	1

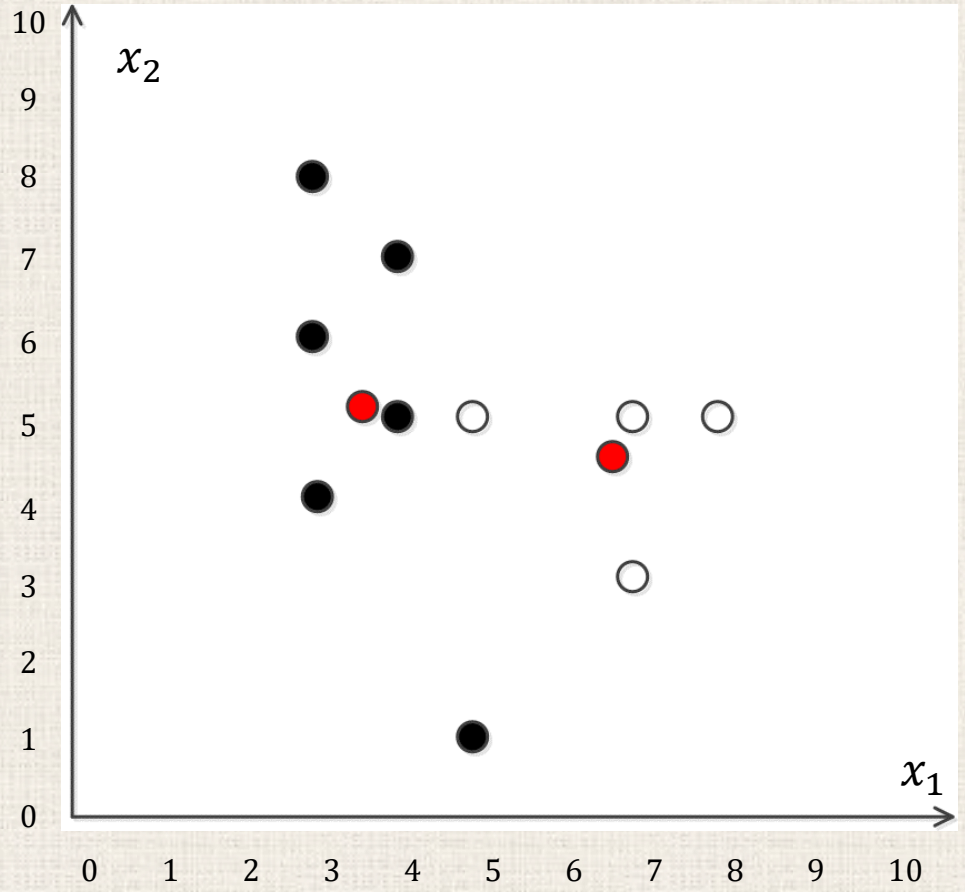


Clustering results after first round:



New centroids of the clusters can be calculated as:

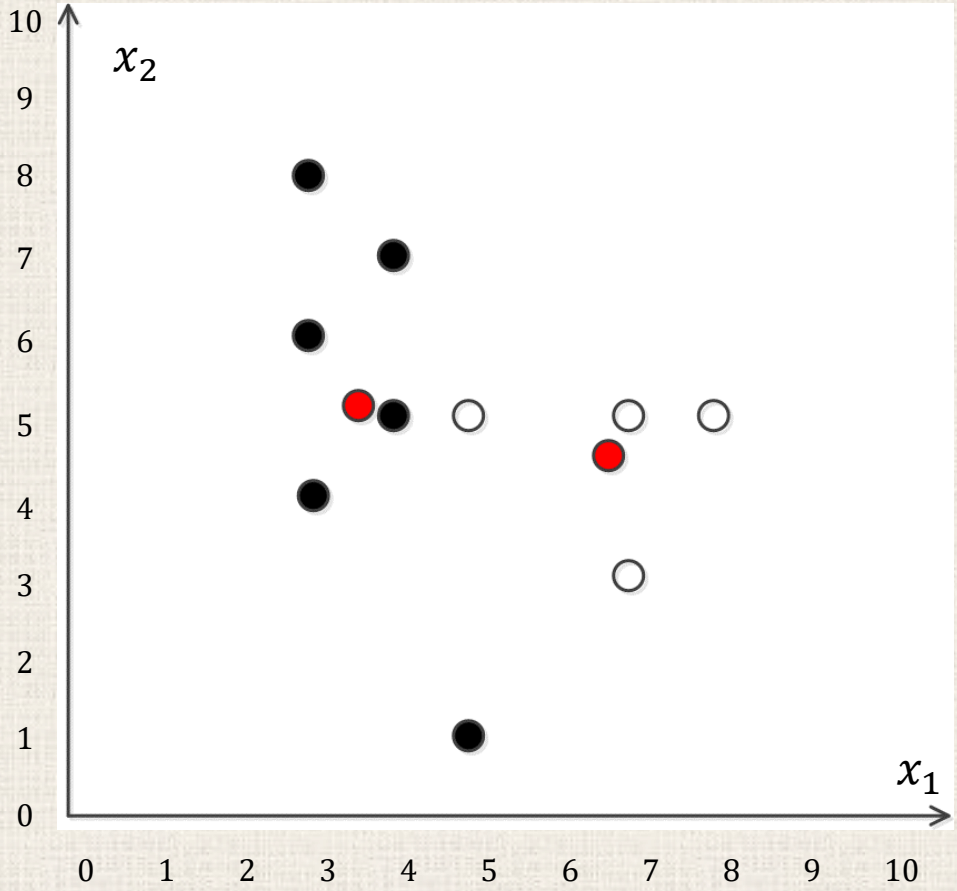
$$c_1 = (3.67,5.17), \qquad c_2 = (6.75,4.5)$$



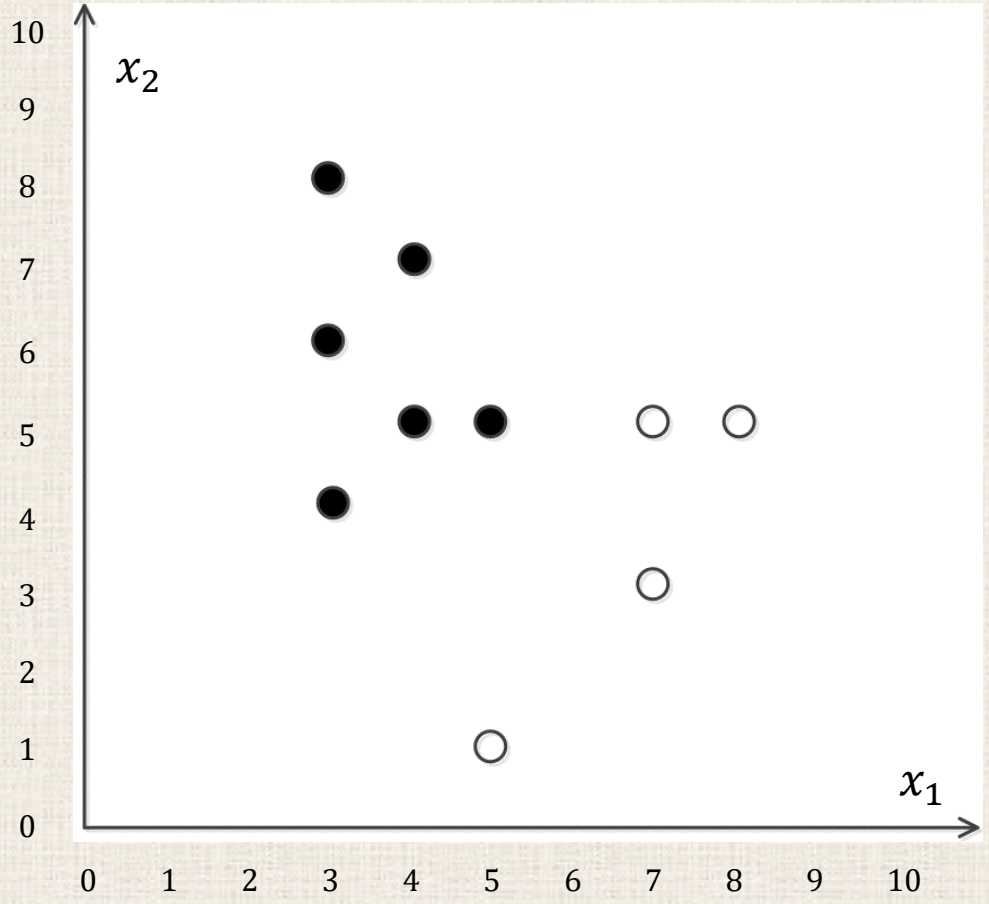
New centroids of the clusters can be calculated as: $c_1 = (3.67,5.17),\ c_2 = (6.75,4.5)$

Let's assign the points to the clusters:

#	x_1	x_2	d_1	d_2	class
1	3	8	2.91	5.13	1
2	4	7	1.86	3.72	1
3	3	6	1.07	4.04	1
4	4	5	0.37	2.80	1
5	5	5	1.34	1.82	1
6	7	5	3.33	0.56	2
7	8	5	4.33	1.35	2
8	3	4	1.35	3.78	1
9	7	3	3.97	1.52	2
10	5	1	4.38	3.91	2

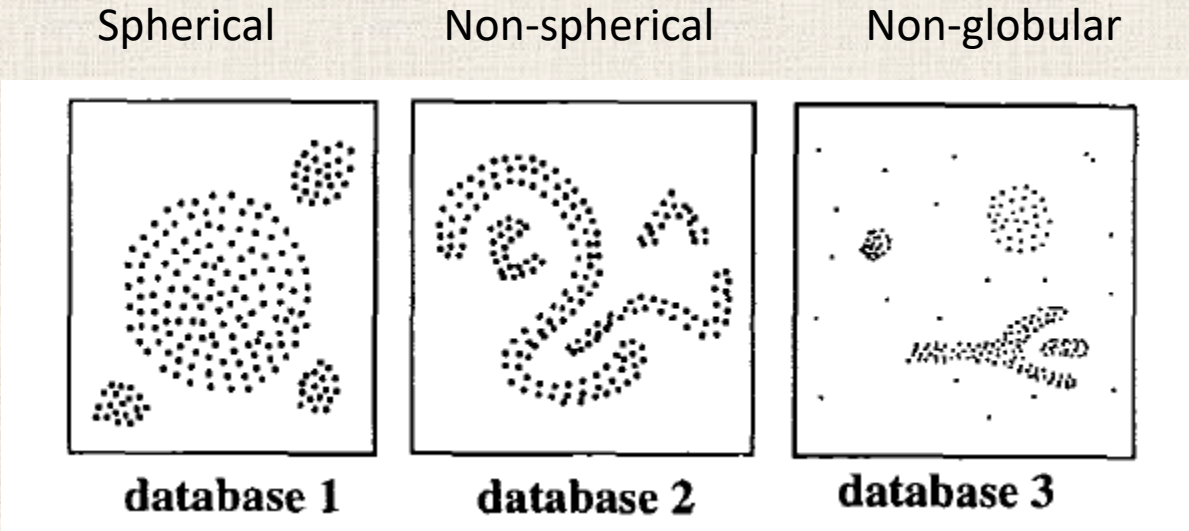


Clustering results after second round:



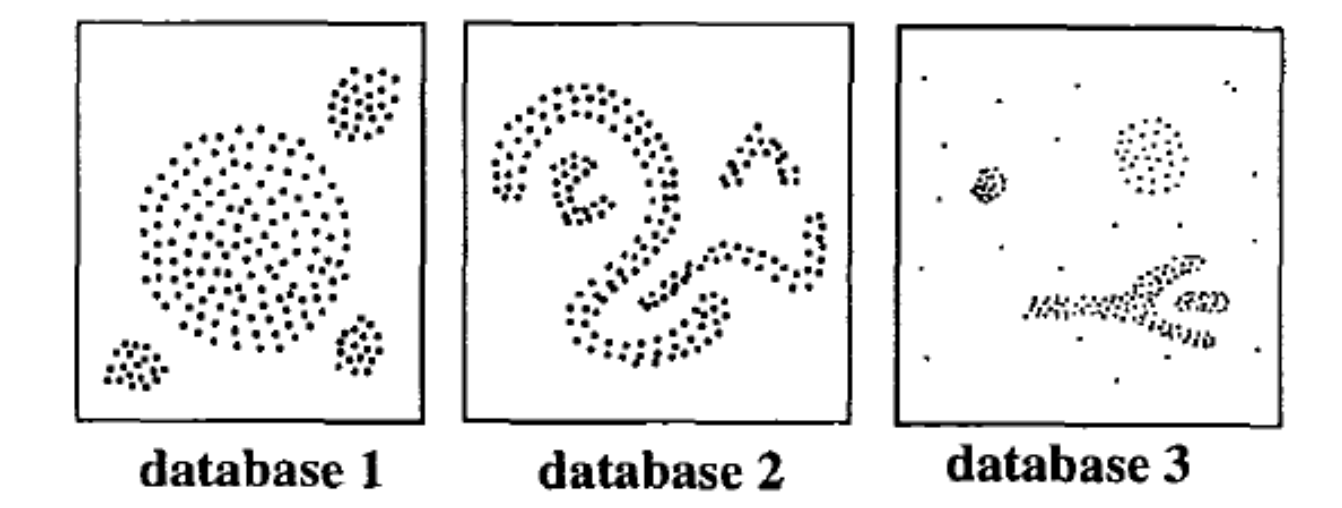
Limits of K means clustering

- K means clustering method is effective on spherical clusters but not very effective on non-spherical clusters.
- K means clustering method is sensitive to initial conditions and outliers. (detecting outliers is important for clustering)

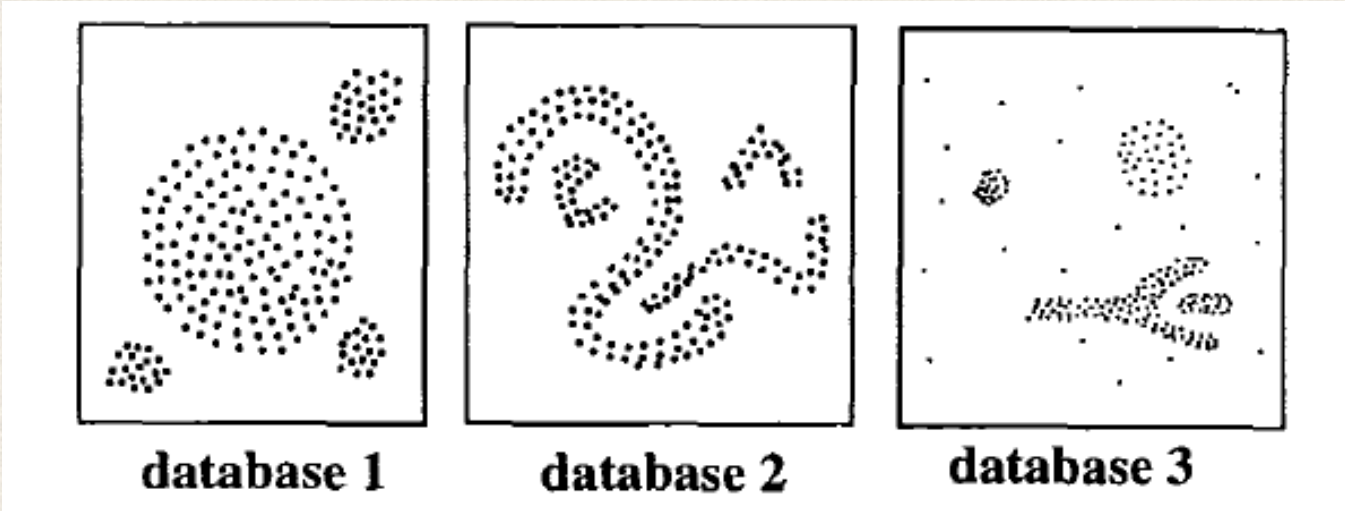


Density-based clustering

- Density based clustering algorithms assume that clusters are ***dense regions in data space separated by regions of lower density.***
- A dense cluster is a region which is "***density connected***", i.e., the density of points in that region is greater than a minimum.



- A dense cluster is a region which is "***density connected***", i.e., the density of points in that region is greater than a minimum.
- Since these algorithms expand clusters based on dense connectivity, ***they can find clusters of arbitrary shapes***.
- DBSCAN is an example of density-based clustering algorithm

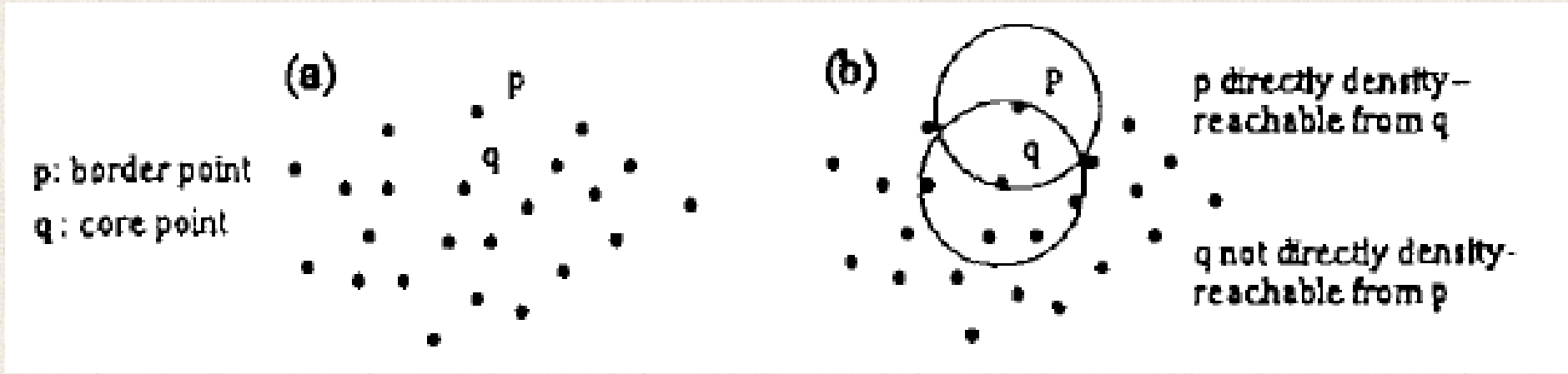


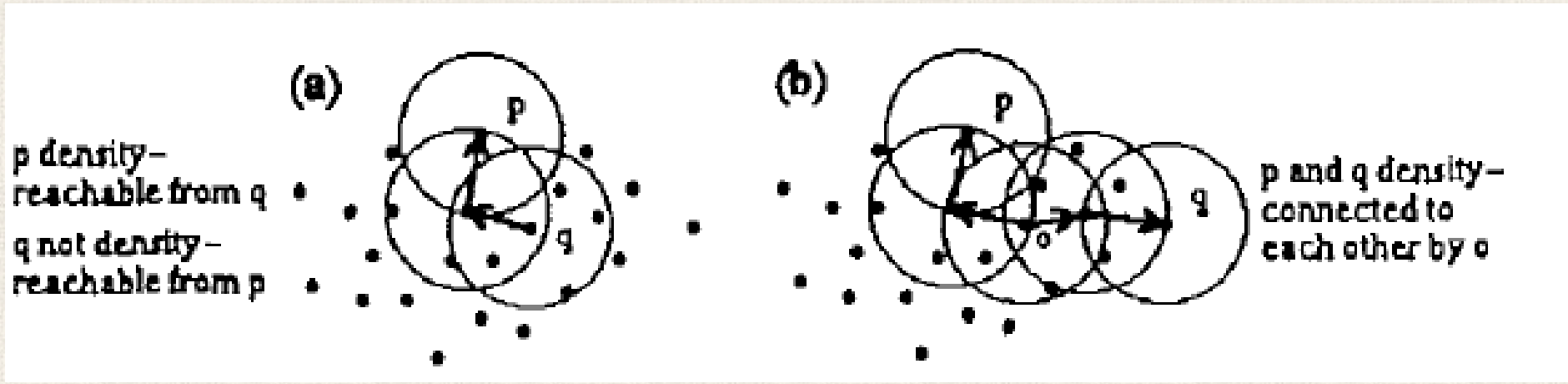
DBSCAN: Density Based Spatial Clustering of Applications with Noise

- Published by M. Easter et. al. in KDD'96. (international conference on knowledge discovery and data mining)
- The algorithm finds dense areas and expends these recursively to find dense arbitrarily shaped clusters
- Two main parameters of the algorithm are ϵ and $MinPts$
- ϵ defines ***radius of the neighborhood region*** and $MinPts$ defines the ***minimum number of points*** that should be contained within that neighborhood. The combination of $\epsilon \; and \; minPoints$ defines "***density***"
- Since it has a concept of ***noise***, *it works well even with noisy data sets*.

DBSCAN Algorithm

- ***Epsilon Neighborhood*** (N_ϵ): set of all points within a distance ϵ.
- ***Core point***: A point that has at least $MinPts$ (including itself) points within its N_ϵ
- ***Directly Density Reachable (DDR)***: a point p is directly density reachable from a point q if q is a core point and $p \in N_\epsilon(q)$.
- ***Border points:*** Points that are DDR but not a core point.
- ***Density Reachable (DR)***: a point p is DR from a core point q if there is a chain of core points that link these two points.
- ***Density-connected***: a point p is density-connected to a point q if there exist a point o such that both $p\ and\ q$ are ***density reachable*** from o


(a)
p
q
p: border point
q : core point
(b)
p
q
p directly density–reachable from q
q not directly density–reachable from p


(a)
p
q
p density–reachable from q
q not density–reachable from p
(b)
p
o
q
p and q density–connected to each other by o

DBSCAN Algorithm

- ***Cluster:*** Let D be a dataset of points. A cluster C w.r.t $\epsilon \; and \; MinPts$ is a non-empty subset of D satisfying the following conditions:
  - All points within the cluster are mutually density-connected.
  - If a point is density-reachable from some point of the cluster, it is part of the cluster as well
- ***Noise points***: Let $C_1, C_2, \ldots, C_k$ be the clusters of the given dataset D. Then the set of points in D that do not belong to any cluster are called noise.

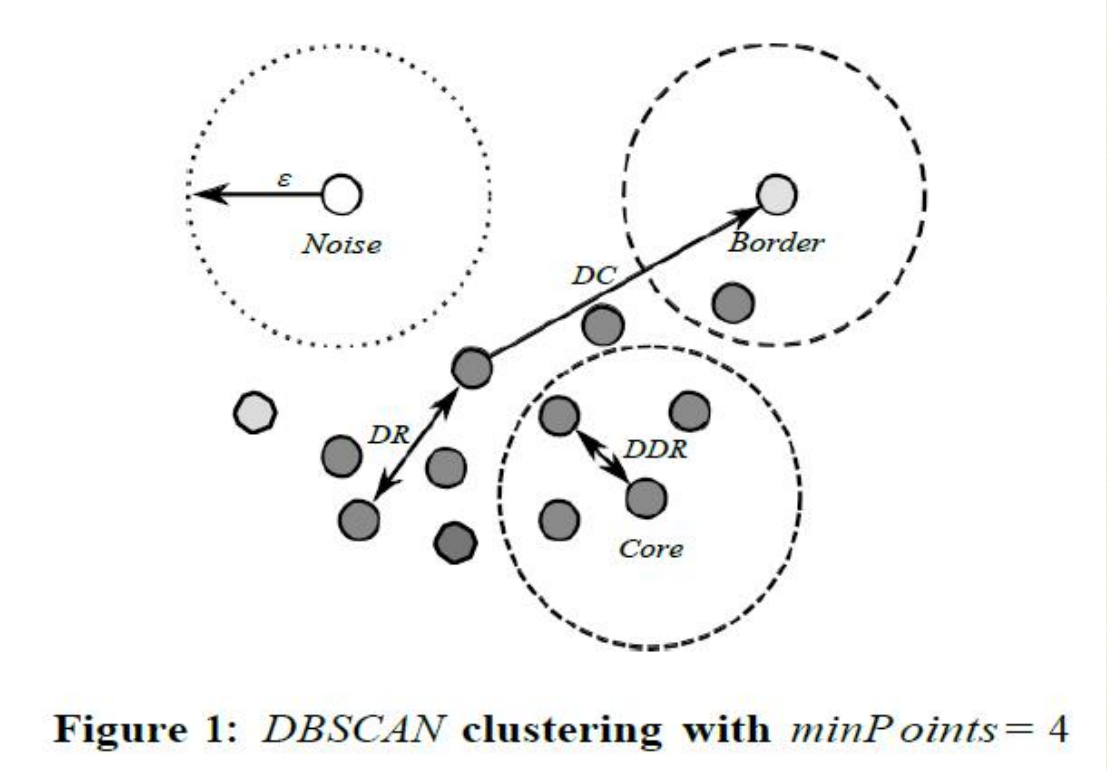


Figure 1: *DBSCAN* **clustering with** $minPoints = 4$

DBSCAN Algorithm

Input: a set of data points $\boldsymbol{X}$, $\epsilon, MinPts$
Output: a set of clusters

Begin
  For each unvisited point $\boldsymbol{x} \in \boldsymbol{X}$ **do**
    mark $\boldsymbol{x}$ as visited;
    $N_x \leftarrow GetNeighborhood(\epsilon, \boldsymbol{x})$;
    if $|N_x| < MinPts$ **then**
      mark x as noise;
    else
      $C = next\ cluster$;
      $C \leftarrow \{\boldsymbol{x}\}$;
      for each point $\boldsymbol{x}' \in N_x$ **do**
        $N_x \leftarrow N_x \backslash \boldsymbol{x}'$
        if $\boldsymbol{x}'$ is not visited **then**
          mark $\boldsymbol{x}'$ as visited
          $N_{x'} \leftarrow GetNeighborhood(\epsilon, \boldsymbol{x}')$;
          if $|N_{x'}| > MinPts$ **then** $N_x \leftarrow N_x \cup N_{x'}$
        if $\boldsymbol{x}'$ is not yet member of any cluster **then**
          $C \leftarrow C \cup \{\boldsymbol{x}'\}$
End

Visualizing DBSCAN clustering

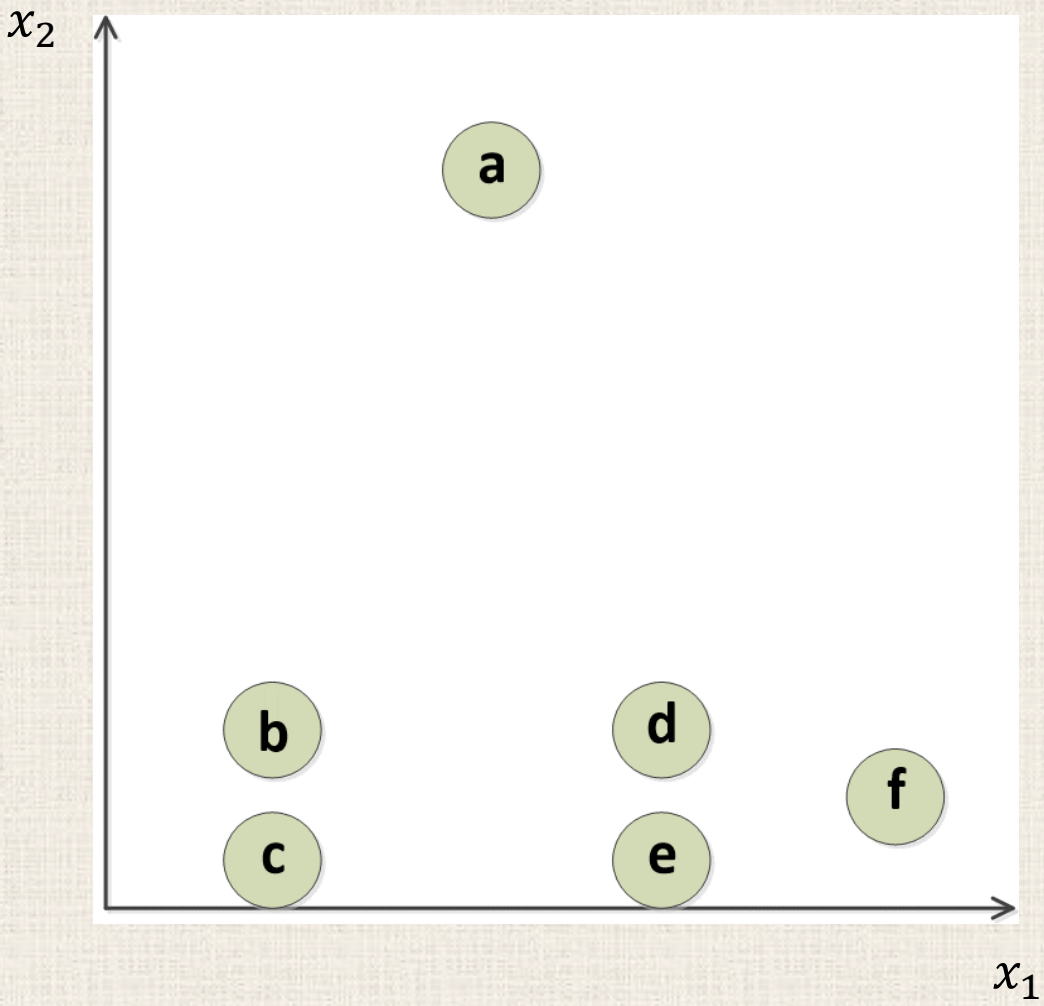
https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

Hierarchical Clustering

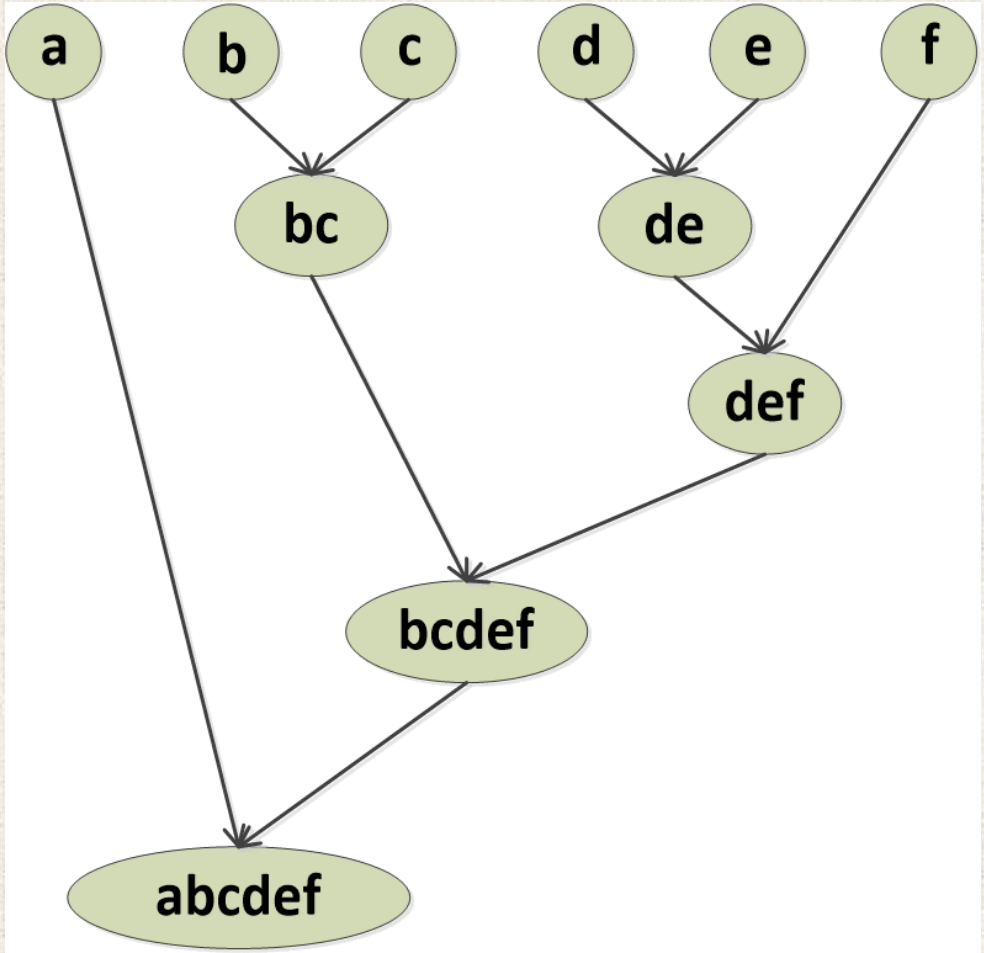
- ***Hierarchical clustering*** methods seek to build a ***hierarchy of clusters***
- Strategies for hierarchical clustering generally fall into two types:
  - ***Agglomerative ("bottom-up")*** approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
  - ***Divisive ("top-down")*** approach: all observations start in one cluster, and split are performed recursively as one moves down the hierarchy.
- The results of hierarchical clustering are usually presented in a ***dendrogram***

Example of a Hierarchical Agglomerative Clustering

Raw data:



Dendrogram:



Cluster Dissimilarity

- In order to determine which clusters should be combined, a measure of dissimilarity between sets of observations is required. This is achieved by using distance between pairs of observations and a ***linkage*** criterion which specifies the dissimilarity of sets as a function of the pairwise distances of observations in the sets.
- ***Distances:*** between a pair of observations.
- ***Linkage criterion***: determines the distance between two sets (A, B) of observations. Commonly used ones include the following:
  - ***Single-linkage:*** $min\{d(a,b): a \in A, b \in B\}$
  - ***Complete-linkage:*** $max\{d(a,b): a \in A, b \in B\}$
  - ***Unweighted average linkage*** (UPGMA): $\frac{1}{|A||B|}\sum_{a\in A}\sum_{b\in B} d(a,b)$

Single-linkage Hierarchical Clustering Algorithm

Input: A set of N data points $\boldsymbol{x}_i, i = 1, \dots, N$

Begin

1. Calculate the proximity matrix $D_{N\times N} = [d(\boldsymbol{x}_i, \boldsymbol{x}_j]$, assign $C_i = \{\boldsymbol{x}_i\}, i = 1, \dots, N$, with level $L(0) = 0$, and sequence number $m = 0$.

2. Find the least dissimilar pair of clusters in current clusters, say $pair(r, s)$, according to
$$d(r, s) = min\{d(i, j)\}$$
where the minimum is over all pairs of clusters in the current clustering.

3. $m = m + 1$, Merge clusters $r\ and\ s$ into a single cluster to form the new cluster with sequence number m. Set the level of this cluster as $L(m) = d(r, s)$

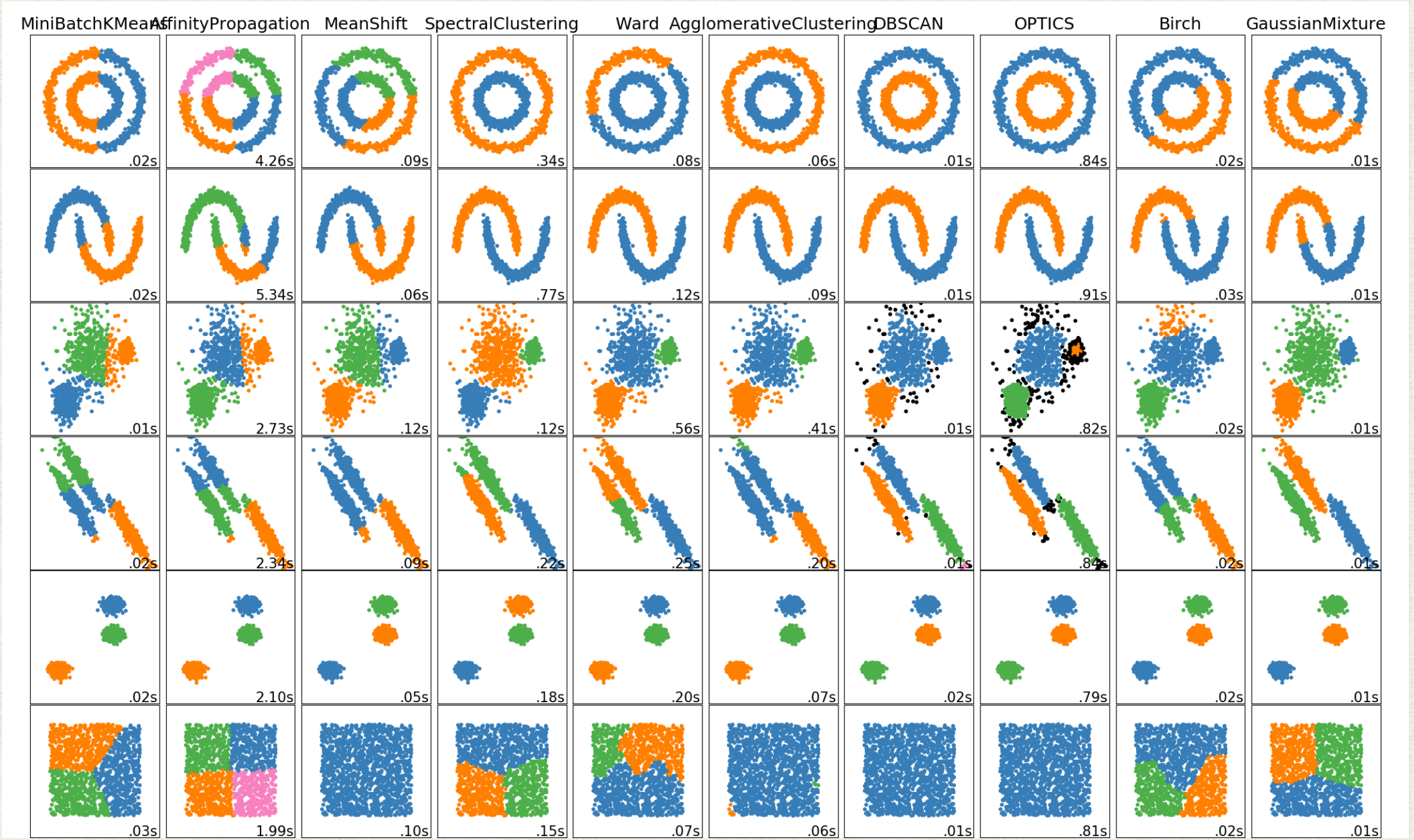
4. Update the proximity matrix D, by deleting the rows and columns corresponding to cluster $r\ and\ s$ and adding a row and column corresponding to the newly formed cluster. The proximity between the new cluster, denoted as (r, s) and older cluster (k) is defined as:
$$d\big((k), (r, s)\big) = min\{d(k, s), d(k, r)\}$$

If all data points are in one cluster

Stop

Else

Go to step2



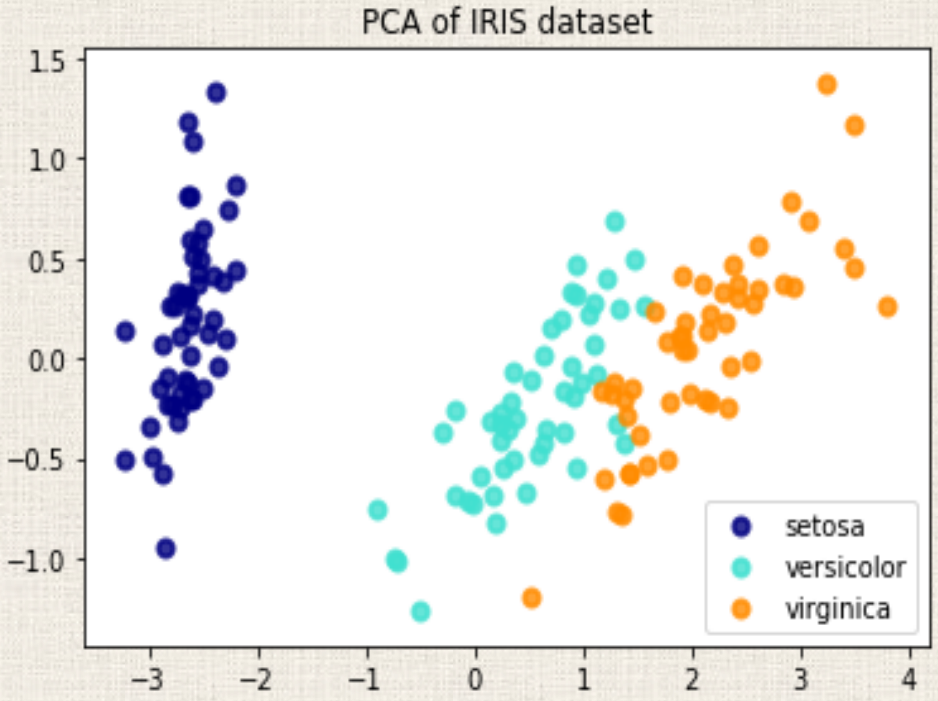
https://scikit-learn.org/stable/modules/clustering.html

Dimensionality Reduction In Machine Learning

- In machine learning, the data we have are often very high-dimensional.
- Feature mapping can make the dimensionality of the data even higher. High dimensionality has some benefits.
- In many cases, we might want to work with a low-dimensional representation
  - Data visualization (if we can get it down to 2 or 3 dimensions). E.g., for exploratory data analysis
  - Reduce computational load
  - Avoid curse of dimensions

Data set visualization using PCA

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	**5.1**	**3.5**	**1.4**	**0.2**	**0**
1	**4.9**	**3**	**1.4**	**0.2**	**0**
2	**4.7**	**3.2**	**1.3**	**0.2**	**0**
3	**4.6**	**3.1**	**1.5**	**0.2**	**0**
4	**5**	**3.6**	**1.4**	**0.2**	**0**
...					
50	**7**	**3.2**	**4.7**	**1.4**	**1**
51	**6.4**	**3.2**	**4.5**	**1.5**	**1**
52	**6.9**	**3.1**	**4.9**	**1.5**	**1**
53	**5.5**	**2.3**	**4**	**1.3**	**1**
54	**6.5**	**2.8**	**4.6**	**1.5**	**1**
...					
144	**6.7**	**3.3**	**5.7**	**2.5**	**2**
145	**6.7**	**3**	**5.2**	**2.3**	**2**
146	**6.3**	**2.5**	**5**	**1.9**	**2**
147	**6.5**	**3**	**5.2**	**2**	**2**
148	**6.2**	**3.4**	**5.4**	**2.3**	**2**
149	**5.9**	**3**	**5.1**	**1.8**	**2**



Dimensionality Reduction In Machine Learning

- How can we reduce the dimensionality of data?
  - Randomly choose a subset of features to keep
  - Keep those few features that have the most variability (measured by the variances of the features)
  - Principal Component Analysis
  - Feature extraction (Feature engineering)

#	feature#1	feature#2	feature#3	feature#4	feature#5	class
1	5.1	3.5	1.4	0.2	100	1
2	4.8	3	1.4	0.3	100	1
3	5.1	3.8	1.6	0.2	100	1
4	4.6	3.2	1.4	0.2	100	1
5	5.3	3.7	1.5	0.2	100	1
6	5	3.3	1.4	0.2	100	1
7	7	3.2	4.7	1.4	100	2
8	6.4	3.2	4.5	1.5	100	2
9	6.9	3.1	4.9	1.5	100	2
10	5.5	2.3	4	1.3	100	2
11	6.5	2.8	4.6	1.5	100	2
12	5.7	2.8	4.5	1.3	100	2
13	6.3	3.3	6	2.5	100	3
14	5.8	2.7	5.1	1.9	100	3
15	7.1	3	5.9	2.1	100	3
16	6.3	2.9	5.6	1.8	100	3
17	6.5	3	5.8	2.2	100	3
18	7.6	3	6.6	2.1	100	3

Which feature should be dropped?

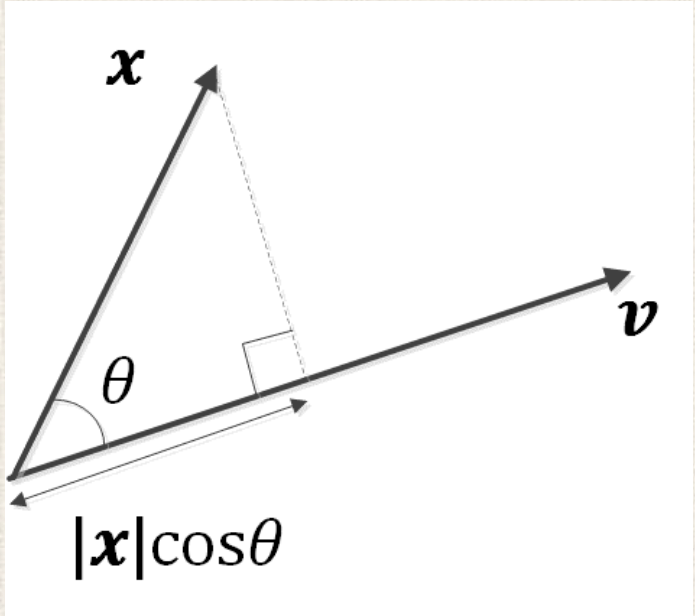
Principal Component Analysis (PCA)

- Given a matrix of data points in high-dimensional space, we want to project these data points into low-dimensional space (e.g., 3D) with ***as much information from the original data set reserved as possible***.
- We want to find one or more orthogonal directions ($\boldsymbol{v_i}$) (basis in low-dimensional space) that ***capture the largest amount of variance in the data***.

- ***Projection:*** if $\boldsymbol{v} \in \mathcal{R}^d$ is a unit vector, i.e., $\|\boldsymbol{v}\| = 1$, then the projection of a vector $\boldsymbol{x} \in \mathcal{R}^d$ onto $\boldsymbol{v}$ is given by

$$\boldsymbol{x}^T\boldsymbol{v} = \|\boldsymbol{x}\|\|\boldsymbol{v}\| \cos\theta = \|\boldsymbol{x}\| \cos\theta$$

Where θ is the angle between the vectors.



- Let $\boldsymbol{X} \in \mathcal{R}^{n \times d}$ be our matrix of data. Where each row is a d-dimensional data point.

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nd} \end{bmatrix}$$

$\boldsymbol{x}_i = [x_{i1} \quad x_{i2} \quad \dots \quad x_{id}]$ is the ith data point in the data set.

- We assume that the data points have zero mean. If that is not the case, we can make it so by subtracting the average of all the rows $\overline{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$, from each row (standardization)

- Since $\boldsymbol{X}$ is zero-mean, the sample variance of the datapoints' projections onto a unit vector $\boldsymbol{v}$ is given by:

$$\frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{x}_i^T\boldsymbol{v}\right)^2 = \frac{1}{n}\|\boldsymbol{X}\boldsymbol{v}\|^2 = \frac{1}{n}(\boldsymbol{X}\boldsymbol{v})^T(\boldsymbol{X}\boldsymbol{v}) = \frac{1}{n}\boldsymbol{v}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v}$$

Where the vector $\boldsymbol{v}$ is constrained to have norm of 1 (unit vector, basis vector)

The first loading vector:

- We then define the first ***loading vector*** $\boldsymbol{v}_1$ (the first ***basis***) as the solution to the following constrained optimization problem:

$$\max_{\boldsymbol{v}} \boldsymbol{v}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{v} \quad s.t.\ \boldsymbol{v}^T \boldsymbol{v} = 1$$

Notice that we have discard the positive constant factor $\frac{1}{n}$ which does not affect the optimal value of $\boldsymbol{v}$.

- To solve this problem, we use the ***Lagrangian*** function to reduce this constrained optimization problem into an unconstrained one:

$$\mathcal{L}(v) = \boldsymbol{v}^T \boldsymbol{X}^T \boldsymbol{X}\boldsymbol{v} - \lambda(\boldsymbol{v}^T \boldsymbol{v} - 1)$$

  - First order necessary conditions for optima imply that:

$$\nabla \mathcal{L}(v) = 2\boldsymbol{X}^T \boldsymbol{X}\boldsymbol{v} - 2\lambda \boldsymbol{v} = 0 \Rightarrow \boldsymbol{X}^T \boldsymbol{X}\boldsymbol{v} = \lambda \boldsymbol{v}$$

  This means that the optimal solution ($\boldsymbol{v_1}$) is an eigenvector of $\boldsymbol{X}^T \boldsymbol{X}$ with eigenvalue λ.

  - And, the optimal value of the objective function is

$$\lambda = \lambda_{max}(\boldsymbol{X}^T \boldsymbol{X})$$

  Which is achieved when $\boldsymbol{v_1}$ is a unit eigenvector of $\boldsymbol{X}^T \boldsymbol{X}$ corresponding to its largest eigenvalue. Where, $\boldsymbol{X}^T \boldsymbol{X}$ is the ***covariance matrix***.

Finding more loading vectors (basis):

- We want the subsequent directions found to also be direction of high variance and to be orthogonal to the existing directions.
- Assuming that we have already found loading vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots . \boldsymbol{v}_{k-1}$, we define the k^{th} loading vector $\boldsymbol{v}_k$ as the solution of the following constrained optimization problem:

$$\max_{\boldsymbol{v}} \boldsymbol{v}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{v} \quad s.t.\ \boldsymbol{v}^T \boldsymbol{v} = 1, \boldsymbol{v}^T \boldsymbol{v}_i = 0, i = 1, \ldots, k-1$$

- It can be shown that $\boldsymbol{v}_k$ is a unit eigenvector of $\boldsymbol{X}^T \boldsymbol{X}$ corresponding to its k^{th} largest eigenvalue.
- The loading vectors (basis) are orthogonal eigenvectors of $\boldsymbol{X}^T \boldsymbol{X}$ (covariance matrix). In other words, they are right-singular vectors of $\boldsymbol{X}$. So, they can all be found simultaneously by computing the ***SVD*** of $\boldsymbol{X}$.

Projecting onto the PCA coordinate system:

- Once we have computed the loading vectors (basis), we can use them as a new coordinate system
- The k^{th} ***principal component*** of a data point $\boldsymbol{x}_i$ is defined as the projection of $\boldsymbol{x}_i$ onto the k^{th} loading vector $\boldsymbol{v}_k$, that is,

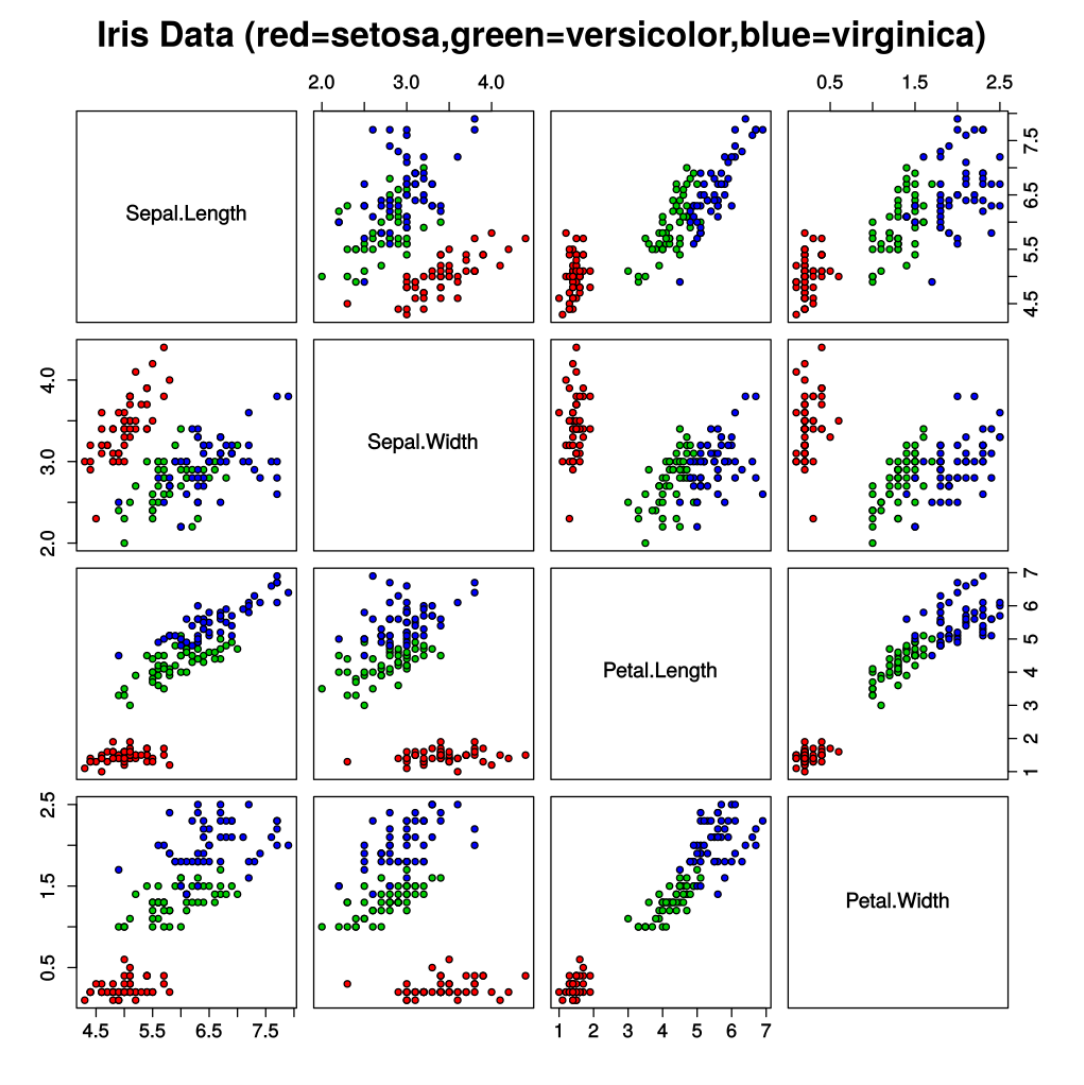
$$z_{ik} = \boldsymbol{x}_i^T \boldsymbol{v}_k; \quad \boldsymbol{z}_i = [z_{i1} \quad \dots \quad z_{ik}]$$

- We can compute all the principal components of all the data points at once using the following:

$$\boldsymbol{Z} = \boldsymbol{XV}$$

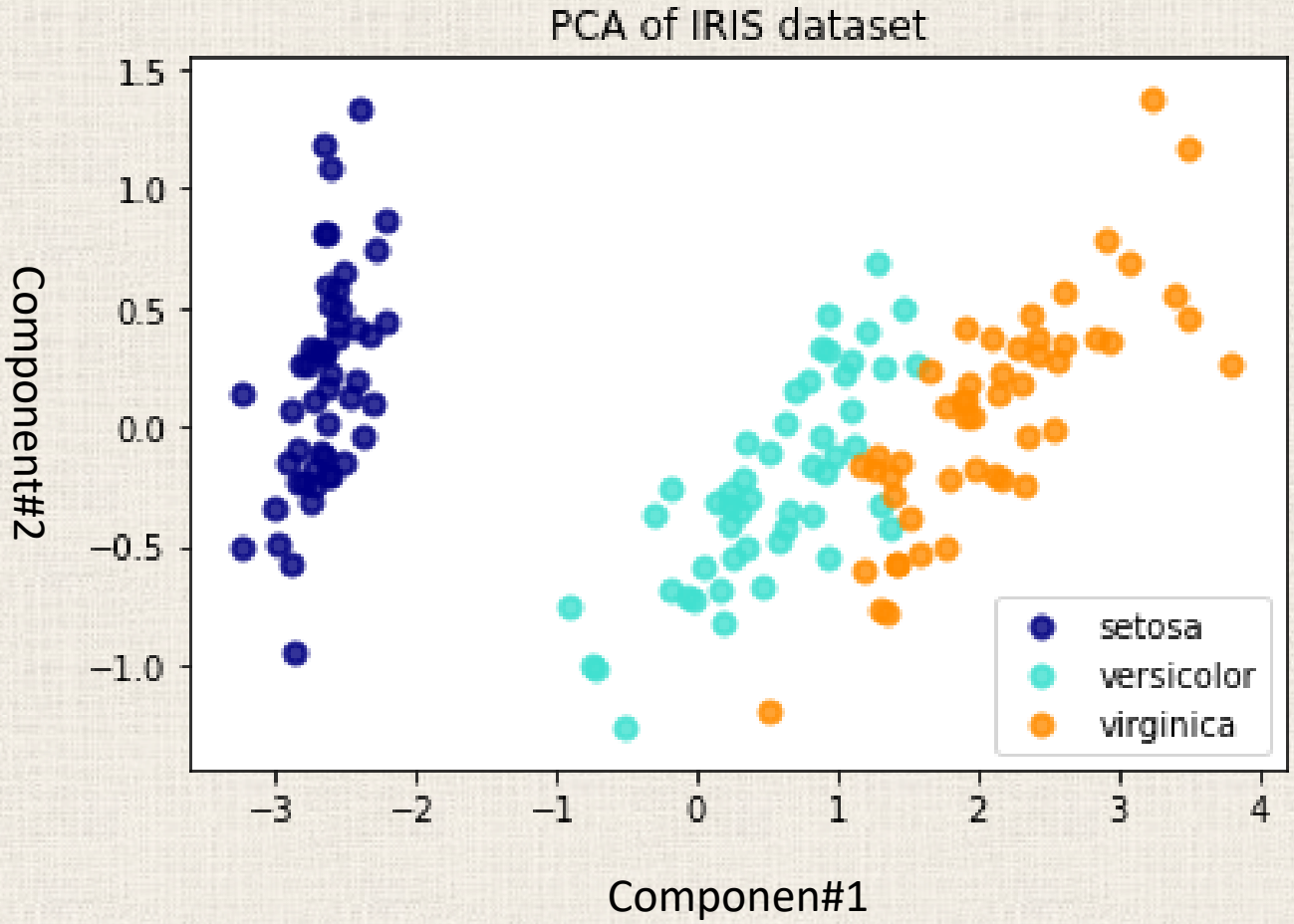
Where, matrix $\boldsymbol{Z}_{n\times k}$ has $\boldsymbol{z}_i = [z_1 \quad \dots \quad z_k]$ as its i^{th} row and matrix $\boldsymbol{V}_{d\times k}$ has the loading vectors $\boldsymbol{v_1}, \boldsymbol{v_2}, \dots, \boldsymbol{v_k}$ as its columns

Visualize the iris data set (PCA not used)



	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	**5.1**	**3.5**	**1.4**	**0.2**	**0**
1	**4.9**	**3**	**1.4**	**0.2**	**0**
2	**4.7**	**3.2**	**1.3**	**0.2**	**0**
3	**4.6**	**3.1**	**1.5**	**0.2**	**0**
4	**5**	**3.6**	**1.4**	**0.2**	**0**
...					
50	**7**	**3.2**	**4.7**	**1.4**	**1**
51	**6.4**	**3.2**	**4.5**	**1.5**	**1**
52	**6.9**	**3.1**	**4.9**	**1.5**	**1**
53	**5.5**	**2.3**	**4**	**1.3**	**1**
54	**6.5**	**2.8**	**4.6**	**1.5**	**1**
...					
144	**6.7**	**3.3**	**5.7**	**2.5**	**2**
145	**6.7**	**3**	**5.2**	**2.3**	**2**
146	**6.3**	**2.5**	**5**	**1.9**	**2**
147	**6.5**	**3**	**5.2**	**2**	**2**
148	**6.2**	**3.4**	**5.4**	**2.3**	**2**
149	**5.9**	**3**	**5.1**	**1.8**	**2**

Visualize the iris data set using PCA



https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_vs_lda.html#sphx-glr-auto-examples-decomposition-plot-pca-vs-lda-py