SAMSUNG

Samsung Innovation Campus

Artificial Intelligence Course

Together for Tomorrow!
Enabling People
Education for Future Generations

Chapter 6.

Machine Learning 2
- Unsupervised Learning

Artificial Intelligence Course

Chapter Description

Chapter objectives

- ✔ Be able to apply necessary machine learning techniques according to the established analysis plan.
- ✔ Be able to apply various unsupervised learning techniques for data compression through reducing high-dimension data into low-dimension and data visualization by reducing dimensions.
- ✔ Be able to solve problems by applying various clustering techniques for subject segmentation.
- ✔ Be able to perform matrix decomposition, principal component analysis and applications.

Chapter contents

- ✔ Unit 1. Unsupervised Machine Learning Algorithm
- ✔ Unit 2. Hierarchical Clustering
- ✔ Unit 3. Non-Hierarchical Clustering
- ✔ Unit 4. Linear Factor Model for Dimensionality Reduction

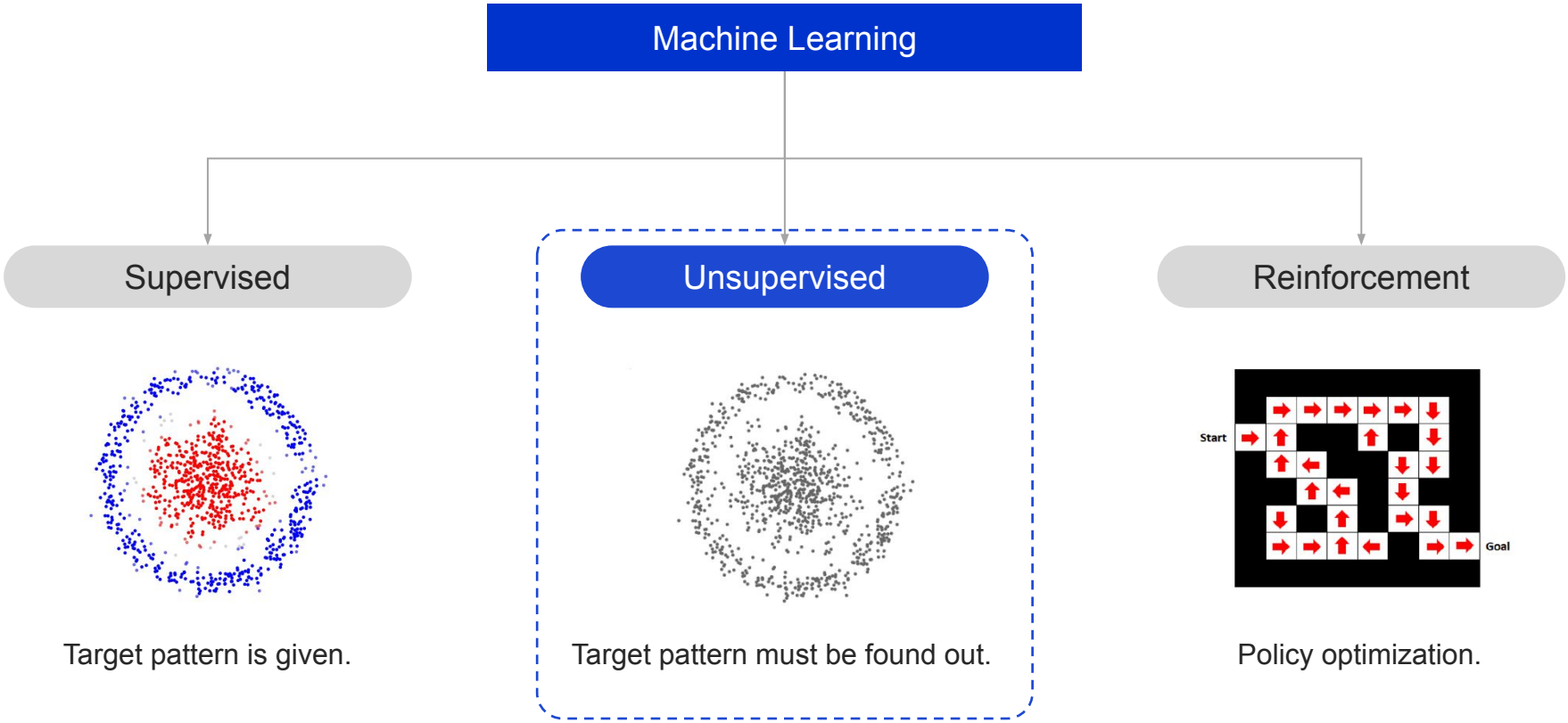
Unit 1.

Unsupervised Machine Learning Algorithm

1.1. The Concept of Unsupervised Learning

1.2. Clustering Analysis

Machine Learning Types



What is unsupervised learning?

The concept of unsupervised learning

- It is a machine learning technique that is performed only if a series of variable such as $x_1, x_2, x_3, \ldots x_p$ are given without any objective variables (or response variable Y) in the data set.
- Because there are no objective variables (or response variables) associated with explanatory variables, the objective of unsupervised learning is to discover a specific pattern or unknown knowledge from given data instead of making a prediction.

What is unsupervised learning?

Machine learning for unsupervised learning

- **Clustering** refers to making groups of similar objects or people according to the analysis purpose and algorithm
- **Association** refers to uncovering a co-occurrence relationship between specific items in the dataset
- **Dimensionality reduction** refers to summarizing a given variable set into a smaller number of variables for providing effective explanations.

Unit 1.

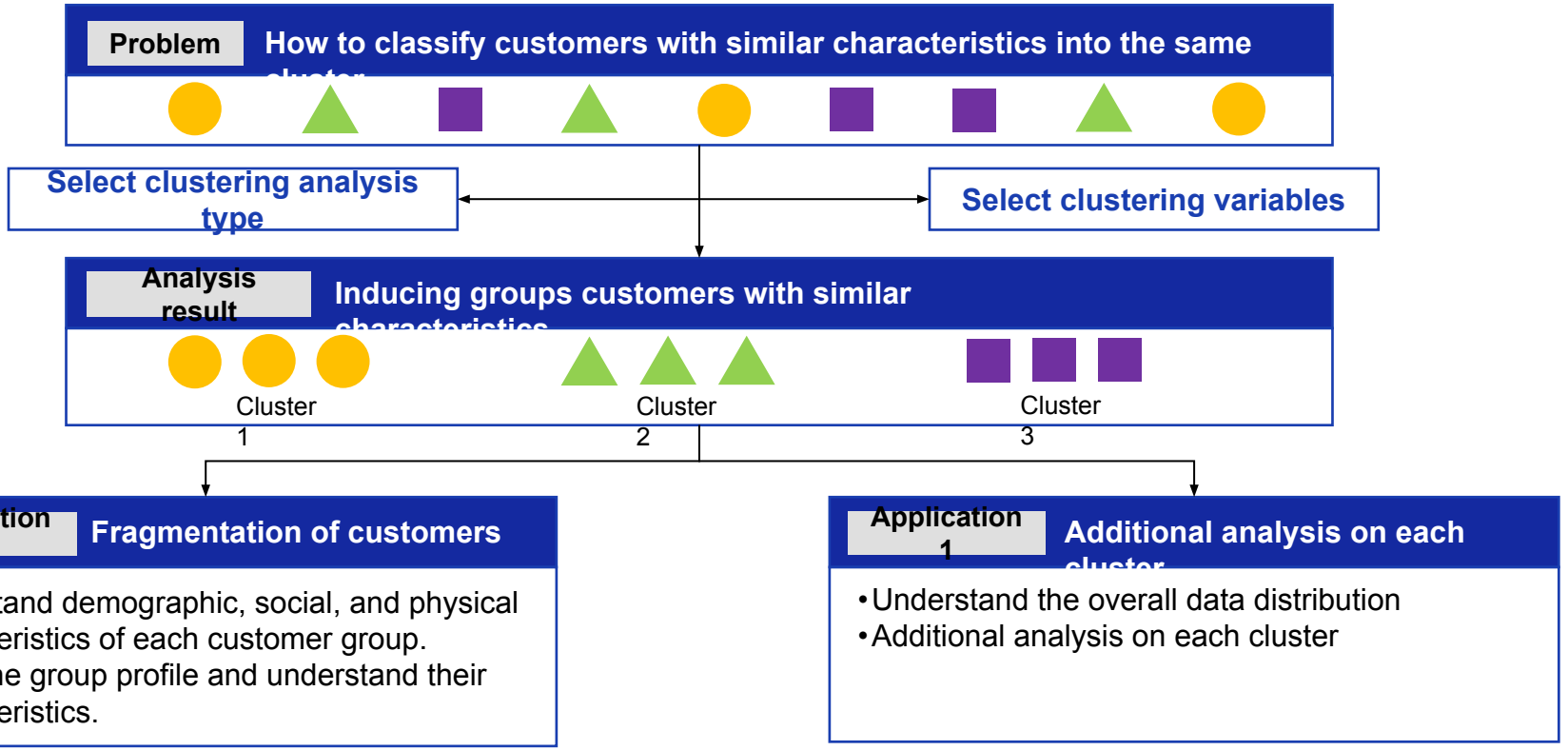
Unsupervised Machine Learning Algorithm

1.1. The Concept of Unsupervised Learning

1.2. Clustering Analysis

What is clustering analysis?

Clustering is a technique which divides unlabeled and uncategorized data into similar groups based on the given observed values.

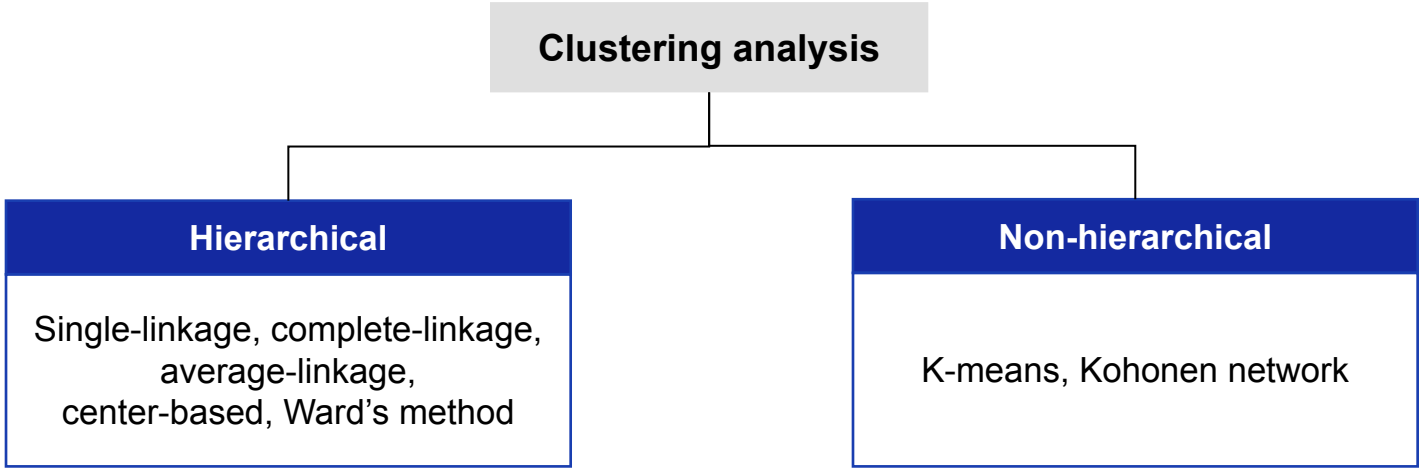


Clustering analysis

- Clustering technique is the most common type of unsupervised learning to group unlabeled data into similar observed values based on the suitable criteria.
- Because clustering deals with unknown groups within the data, analysts cannot instruct the computer program what to find.
- So, clustering is mainly used to know the attributes of different groups and give us insight into underlying patterns of different groups rather than for making a prediction.

Types of clustering analysis

- **Hierarchical clustering**: Divisive and agglomerative
- **Non-hierarchical clustering**: the number of clusters is determined in advance



Types of clustering analysis

- In general, clustering analysis is classified into hierarchical clustering, non- hierarchical clustering, or partition-based clustering.

Non-Hierarchical Clustering (Partition-based Clustering)	K-Means clustering
	K-Medoids clustering or PAM (Partitioning Around Method
	DBSCAN (Density Based Spatial Clustering of Application with Noise)
	Self Organizing Map
	Fuzzy clustering
Hierarchical Clustering	Agglomerative or Bottom-up clustering
	Divisive or Top-down clustering
Mixture Distribution Clustering	Gaussian Mixture Model

Application of clustering analysis

- Identifying the pattern of unknown data groups in financial data analysis
- Detecting any network intrusions (unauthorized penetration activities on a computer network)
- Object and facial recognition from digital images
- Clustering documents, music, or movies into different themes
- Customer segmentation in the marketing field

Unit 2.

Hierarchical Clustering

2.1. Hierarchical method

Hierarchical method

Hierarchical method is useful to sort clusters in the natural hierarchy structure. There are two hierarchical clustering methods, which are agglomerative and divisive clustering.

- Agglomerative clustering
  - Bottom-Up
  - Starts with n-clusters to progressively agglomerate similar clusters until one final cluster remains.
- Divisive clustering
  - Top-Down
  - Starts from a single cluster that includes all records and divides it into n-clusters (agglomerative clustering is a mainly used hierarchical method.)

How does **agglomerative clustering** work?

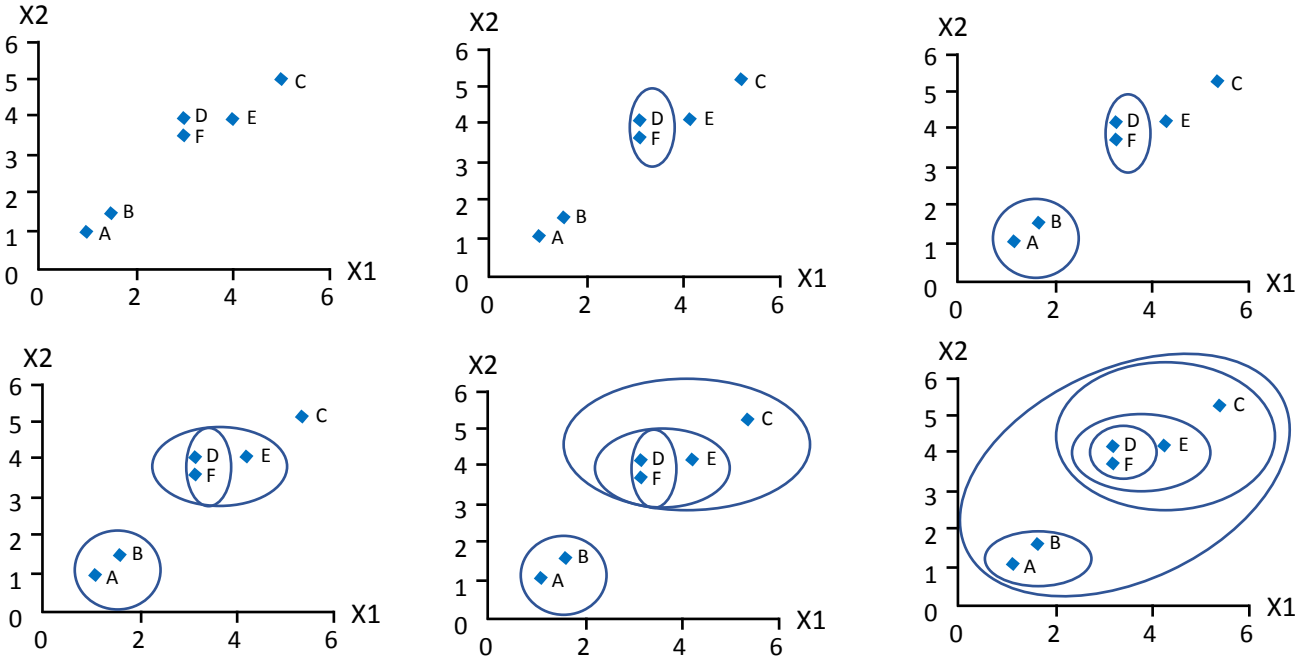
- Starts with single-record clusters
- Pairs of similar clusters are merged until all clusters have been merged into one big cluster containing all records.

What is **agglomerative clustering algorithm**?

- Starts with n-clusters
- Merges two closest record clusters as one cluster
- Two closest clusters are merged in every step. This refers to adding single records to the existing cluster or combining two existing clusters.

Agglomerative hierarchical clustering

- Start: Each object is an individual atomic cluster.
- Repetition: Repeated merging of two closest clusters
- Finish: Generates a single cluster



Hierarchical Clustering

About the hierarchical clustering:

- This is a unsupervised learning algorithm: there are only X variables.
- Also called “agglomerative clustering”.
- Nearest items are gathered to form ever growing clusters.
- Can be visualized resembling an inverted tree-like structure called “dendrogram”.

Purpose of the hierarchical clustering:

- Cluster the observations into a given number of clusters.

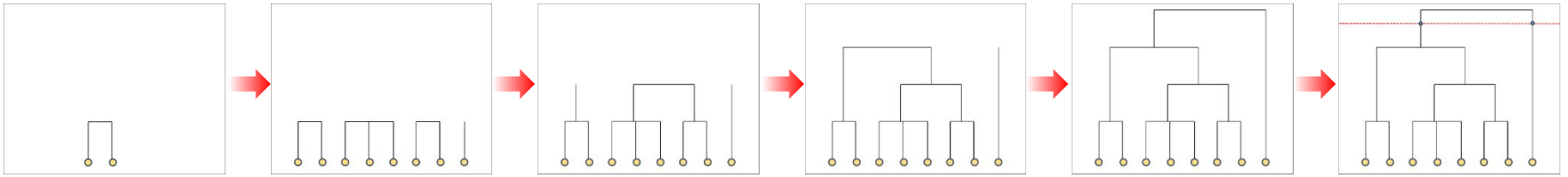
Pros:

- Intuitive interpretation of the results
- Dendrogram that can be further processed to produce a desired number of clusters.

Cons:

- A bit more time consuming than k-means.

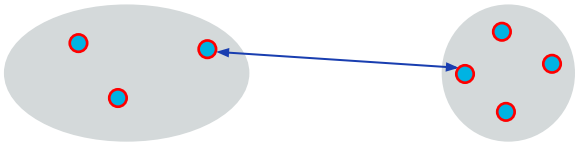
Hierarchical clustering algorithm:



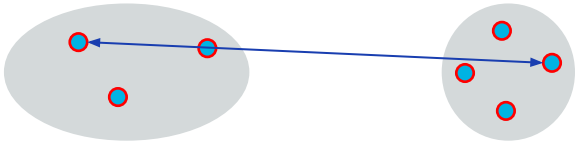
1) First, group together the nearest observations and get small clusters: (a) ~ (b).
2) Group the nearest clusters together to forma larger clusters: (c).
3) Repeat from the step 2) until all the observations are grouped together into a single cluster: (d) ~ (e).
4) Cut the dendrogram at an appropriate height such that we get the desired number of clusters: (f).

Hierarchical clustering algorithm:

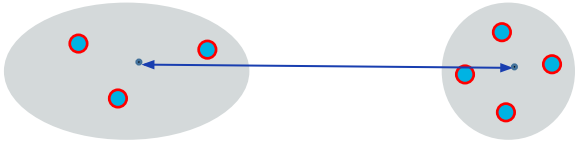
- How do we measure the inter-cluster distances?



Single linkage



Complete linkage



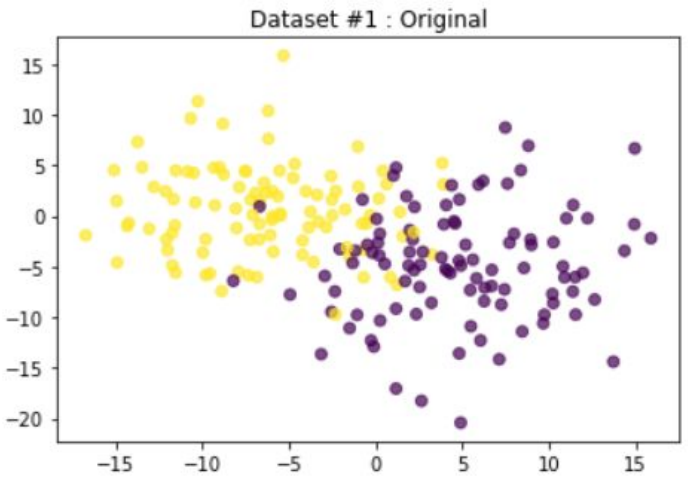
Average linkage

Apply agglomerative clustering and visualize:

Dataset #1

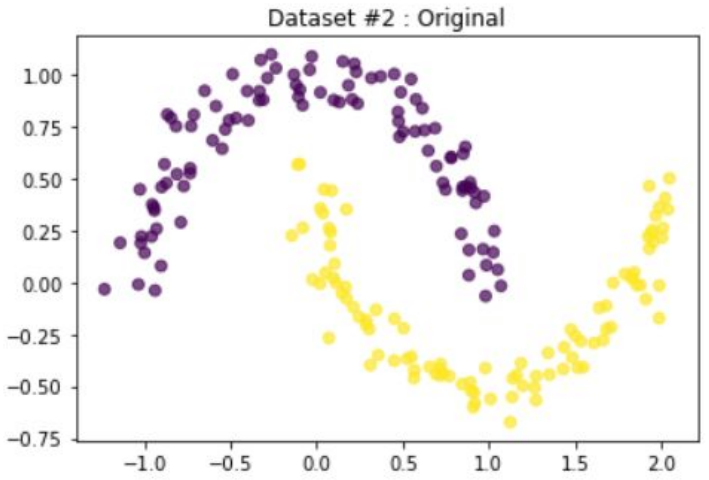
In [1]:
```
from sklearn.datasets import make_blobs, make_moons
import matplotlib.pyplot as plt
```

In [2]:
```
X1, label1 = make_blobs(n_samples=200, n_features=2, centers=2, cluster_std = 5, random_state=123)
plt.scatter(X1[:,0],X1[:,1], c= label1, alpha=0.7 )
plt.title('Dataset #1 : Original')
plt.show()
```



Dataset #2

In [3]:
```
X2, label2 = make_moons(n_samples=200, noise=0.08, random_state=123)
plt.scatter(X2[:,0],X2[:,1], c= label2, alpha=0.7 )
plt.title('Dataset #2 : Original')
plt.show()
```



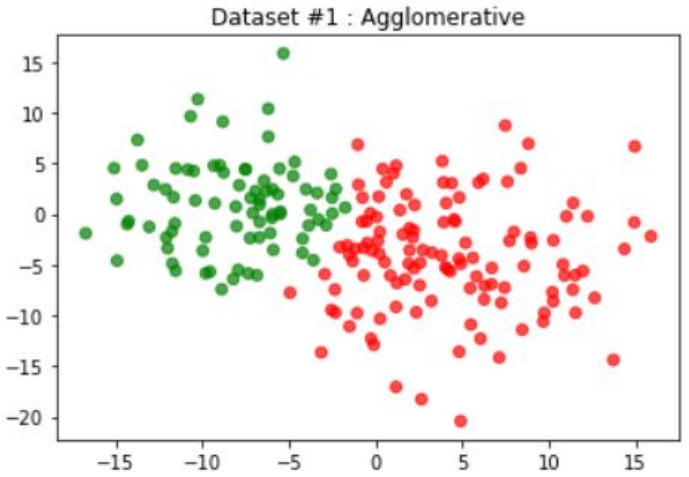
clusters

```
In [4]: from sklearn.cluster import AgglomerativeClustering

In [5]: import numpy as np
        import pandas as pd

In [6]: X1
```

```
Out[6]: array([[ 4.83455936e+00,  1.61209639e+00],
               [-7.19342110e+00,  1.57331116e+00],
               [-1.19919003e+01, -3.38784956e+00],
               [ 8.94965320e+00, -2.34628131e+00],
               [ 5.46630481e+00, -7.33334000e+00],
               [-9.82679362e-01,  2.87439331e+00],
               [ 4.82713114e+00, -1.35870919e+01],
               [-5.95639441e+00, -4.86198432e-02],
               [ 3.25723451e+00, -3.81242660e+00],
               [-1.49388489e+01,  1.46494770e+00],
               [-9.32588959e+00,  4.75039194e+00],
               [ 1.23752259e+00, -6.81998526e+00],
               [ 1.19728386e+01, -5.62837526e+00],
               [-2.58199902e-01, -1.23070271e+01],
               [ 1.49590341e+01,  6.65671714e+00],
               [ 2.25314845e-01, -3.91267708e+00],
               [-5.93898363e+00,  2.41971297e+00],
               [-4.66024882e+00,  5.12509843e+00],
               [-2.49261965e+00,  3.07334679e-02],
               [ 7.73413068e+00, -2.65986906e+00],
               [-5.12306455e-01, -7.42476287e-01],
               [ 1.57887480e+00, -2.01725863e+00],
               [-4.20565001e+00, -2.44769569e+00],
               [-4.00859899e+00,  2.42460873e+00],
               [ 3.91603180e+00,  3.10476808e+00],
               [-1.05722363e+00, -9.77742886e+00],
```

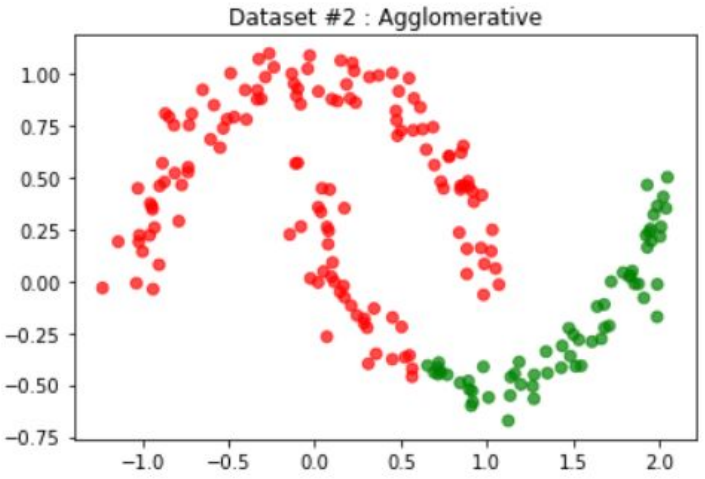
Dataset #1 and two clusters

In [7]:
```
agglo = AgglomerativeClustering(n_clusters=2)
agglo.fit(X1)
myColors = {0:'red',1:'green'}                                    # Define a color palette: 0~1.
plt.scatter(X1[:,0],X1[:,1], c= pd.Series(agglo.labels_).apply(lambda x: myColors[x]), alpha=0.7 )
plt.title('Dataset #1 : Agglomerative')
plt.show()
```



Dataset #2 and two clusters

In [8]:
```
agglo = AgglomerativeClustering(n_clusters=2)
agglo.fit(X2)
myColors = {0:'red',1:'green'}                                    # Define a color palette: 0~1.
plt.scatter(X2[:,0],X2[:,1], c= pd.Series(agglo.labels_).apply(lambda x: myColors[x]), alpha=0.7 )
plt.title('Dataset #2 : Agglomerative')
plt.show()
```

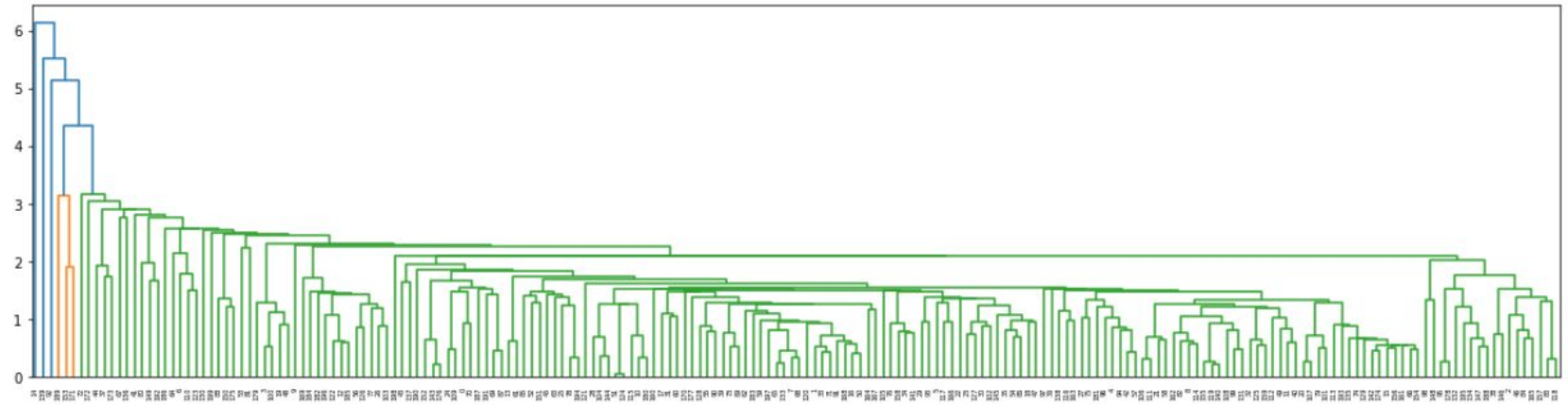


Apply hierarchical clustering and visualize:

- Dataset #1 and show dendrogram.
- Cluster hierarchically using single linkage

In [9]:
```
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
```

In [10]:
```
myLinkage = linkage(X1,method='single')
plt.figure(figsize=(20,5))
dendrogram(myLinkage)
plt.show()
```



Apply hierarchical clustering and visualize:

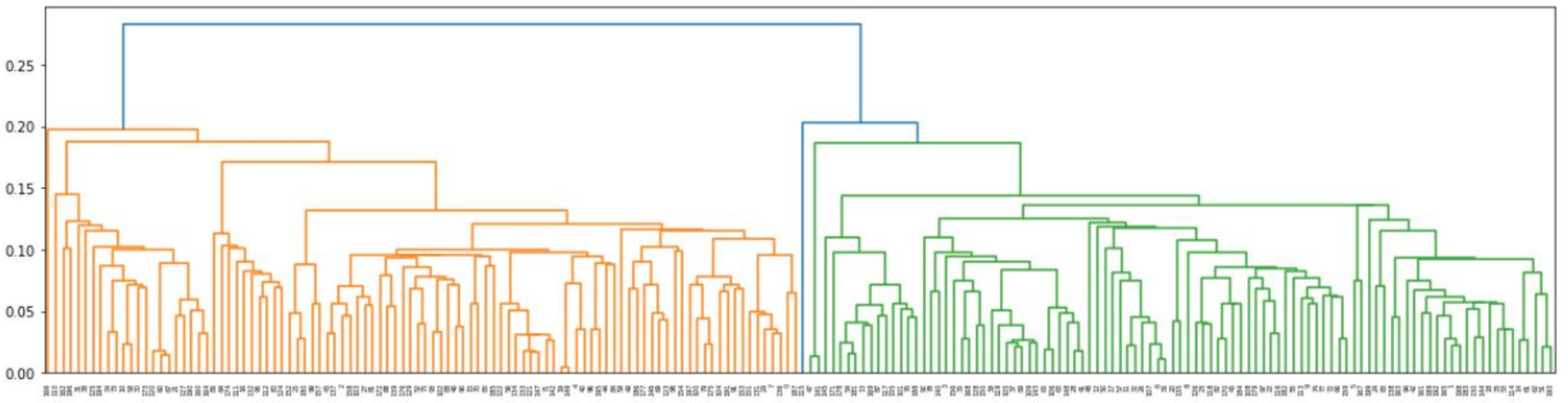
- Dataset #1 and clusters by cutting the dendrogram
- Cut at the height (distance) = 5 (change this value at will).

```
In [11]: labels = fcluster(myLinkage, 5, criterion='distance')
         pd.Series(labels).value_counts()

Out[11]: 1    197
         2      1
         3      1
         4      1
         dtype: int64
```

Apply hierarchical clustering and visualize:

- Dataset #2 and show dendrogram.
- Cluster hierarchically using single linkage.

In [12]:
```
myLinkage = linkage(X2,method='single')
plt.figure(figsize=(20,5))
dendrogram(myLinkage)
plt.show()
```



Apply hierarchical clustering and visualize:

- Dataset #2 and clusters by cutting the dendrogram
- Cut at the height (distance) = 0.23 (change this value at will).

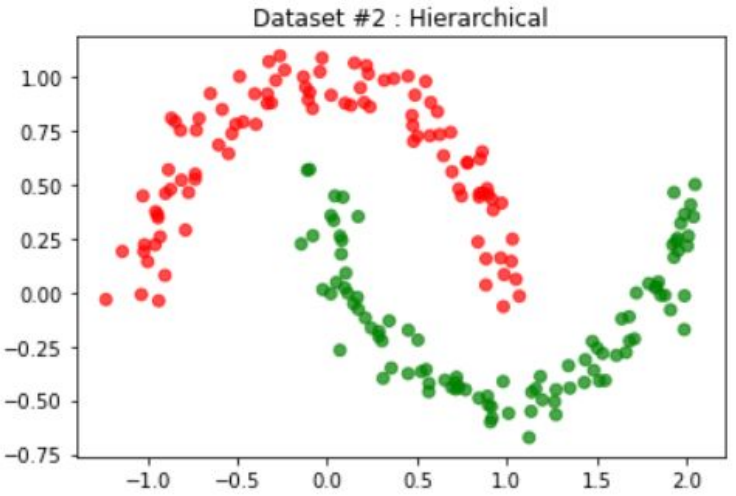
In [13]:
```
labels = fcluster(myLinkage, 0.23, criterion='distance')
pd.Series(labels).value_counts()
```

Out[13]:
```
1    100
2    100
dtype: int64
```

Apply hierarchical clustering and visualize:

- Define a color palette: 1~2

In [14]:
```
myColors = {1:'red',2:'green'}
plt.scatter(X2[:,0],X2[:,1], c= pd.Series(labels).apply(lambda x: myColors[x]), alpha=0.7 )
plt.title('Dataset #2 : Hierarchical')
plt.show()
```



Creating hierarchical clustering

- Use the agglomerative method to create hierarchical clustering.
- Import the iris data.

In [1]:
```
from sklearn.datasets import load_iris
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
%matplotlib inline
```

In [3]:
```
iris=load_iris()
```

Creating hierarchical clustering

In [4]:
```
iris_data=iris.data
iris_data_pd=pd.DataFrame(iris.data, columns=iris.feature_names)
iris_data_pd
```

Out[4]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

Creating hierarchical clustering

In [5]:
```
iris_data_pd.iloc[:,2:4]
```

Out[5]:

	petal length (cm)	petal width (cm)
0	1.4	0.2
1	1.4	0.2
2	1.3	0.2
3	1.5	0.2
4	1.4	0.2
...	...	...
145	5.2	2.3
146	5.0	1.9
147	5.2	2.0
148	5.4	2.3
149	5.1	1.8

150 rows × 2 columns

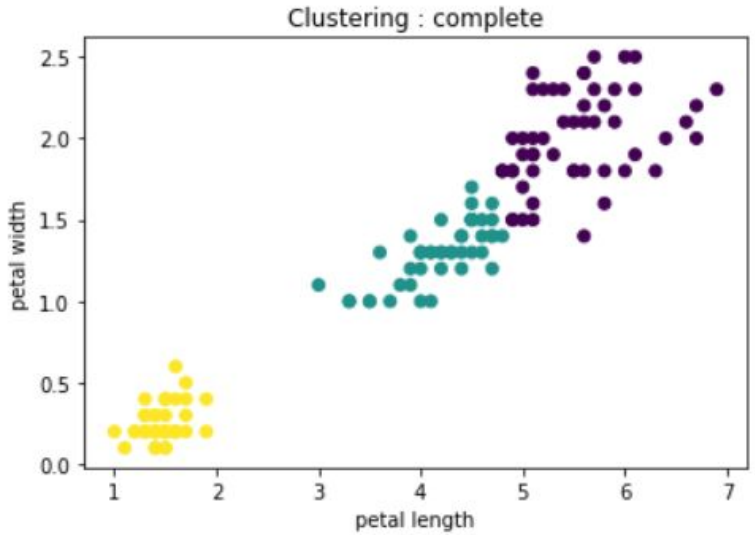
Creating hierarchical clustering

- To use 'petal length' and 'petal width' columns, import the sklearn agglomerative clustering, which is supported by the sklearn.clustering package.
- The distance measurement criteria between clustering can be set with the linkage parameter, which only supports ward, complete, and average. Check out how all of the three cases are used.

In [6]:
```
from sklearn.cluster import AgglomerativeClustering
likage=["complete", "average","ward"]
for idx, i in enumerate(likage):
    plt.figure(idx)
    hier=AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage=i)
    hier.fit(iris_data_pd.iloc[:,2:4])
    plt.scatter(iris_data_pd.iloc[:,2], iris_data_pd.iloc[:,3],c=hier.labels_)
    plt.title("Clustering : " + i)
    plt.xlabel('petal length')
    plt.ylabel('petal width')
    plt.show()
```

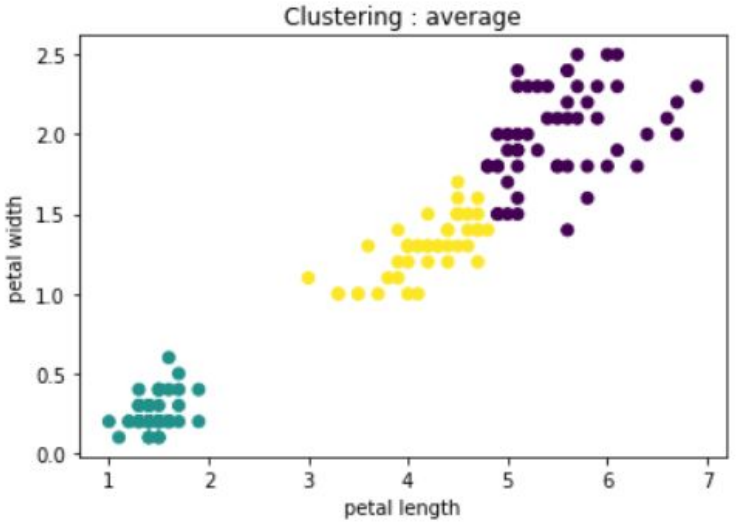
Creating hierarchical clustering

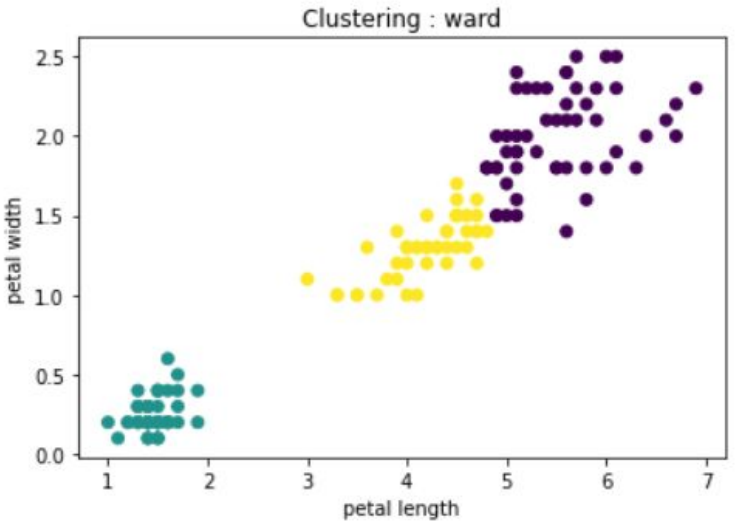
- To use ’petal length’ and ’petal width’ columns, import the sklearn agglomerative clustering, which is supported by the sklearn.clustering package.
- The distance measurement criteria between clustering can be set with the linkage parameter, which only supports ward, complete, and average. Check out how all of the three cases are used.



Creating hierarchical clustering

- To use 'petal length' and 'petal width' columns, import the sklearn agglomerative clustering, which is supported by the sklearn.clustering package.
- The distance measurement criteria between clustering can be set with the linkage parameter, which only supports ward, complete, and average. Check out how all of the three cases are used.





Creating hierarchical clustering

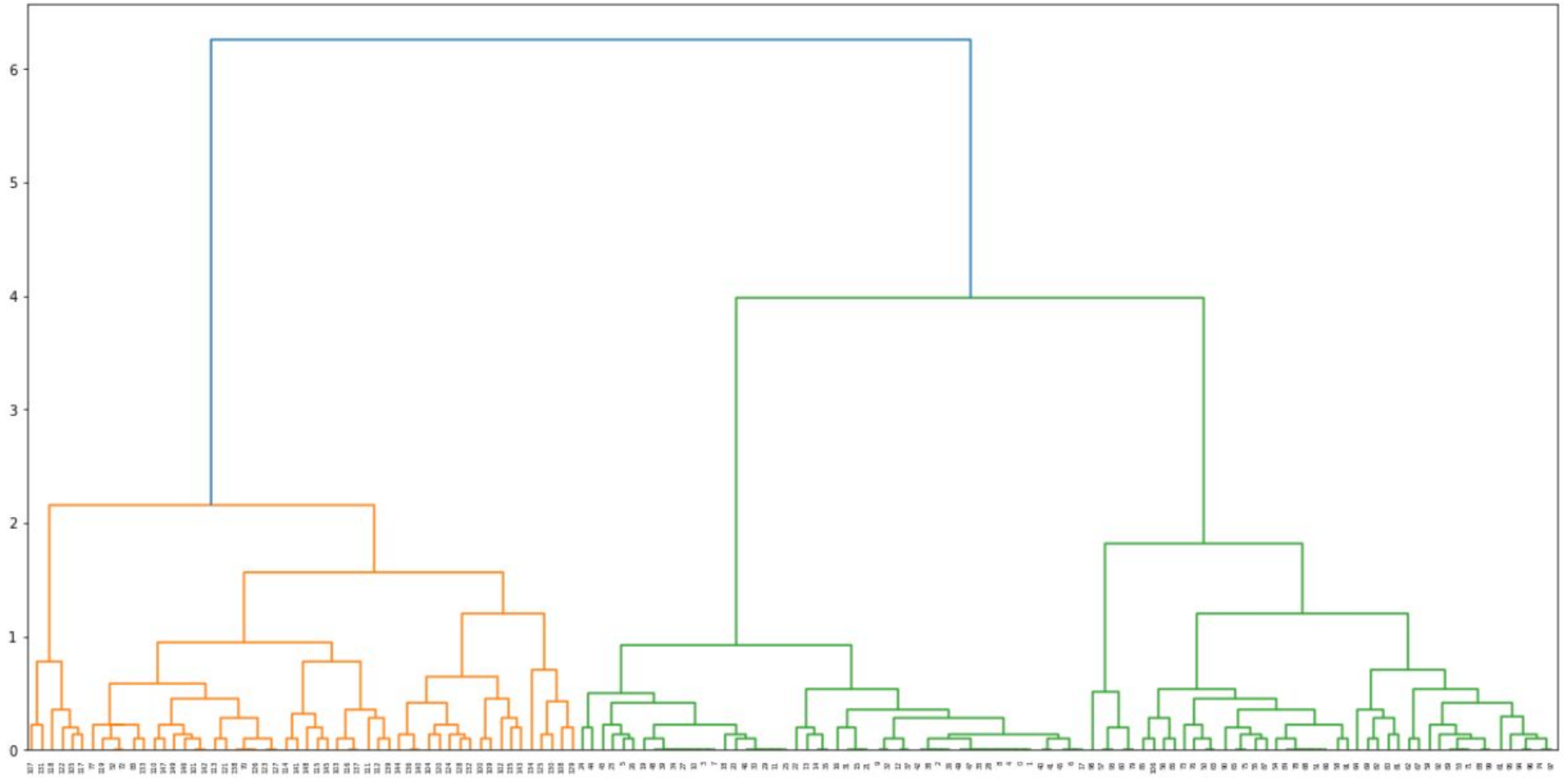
- Use the Scikit-learn to confirm the clustering result. You may want to check how the tree is created, but this package does not provide this function.
- So, use the Scipy package for clustering and drawing a tree.
- As provided previously, implement the complete linkage.

Creating hierarchical clustering

In [7]:
```
from scipy.cluster import hierarchy
```

In [8]:
```
hierar=hierarchy.linkage(iris_data_pd.iloc[:,2:4],'complete')
plt.figure(figsize=(20,10))
dn=hierarchy.dendrogram(hierar)
```

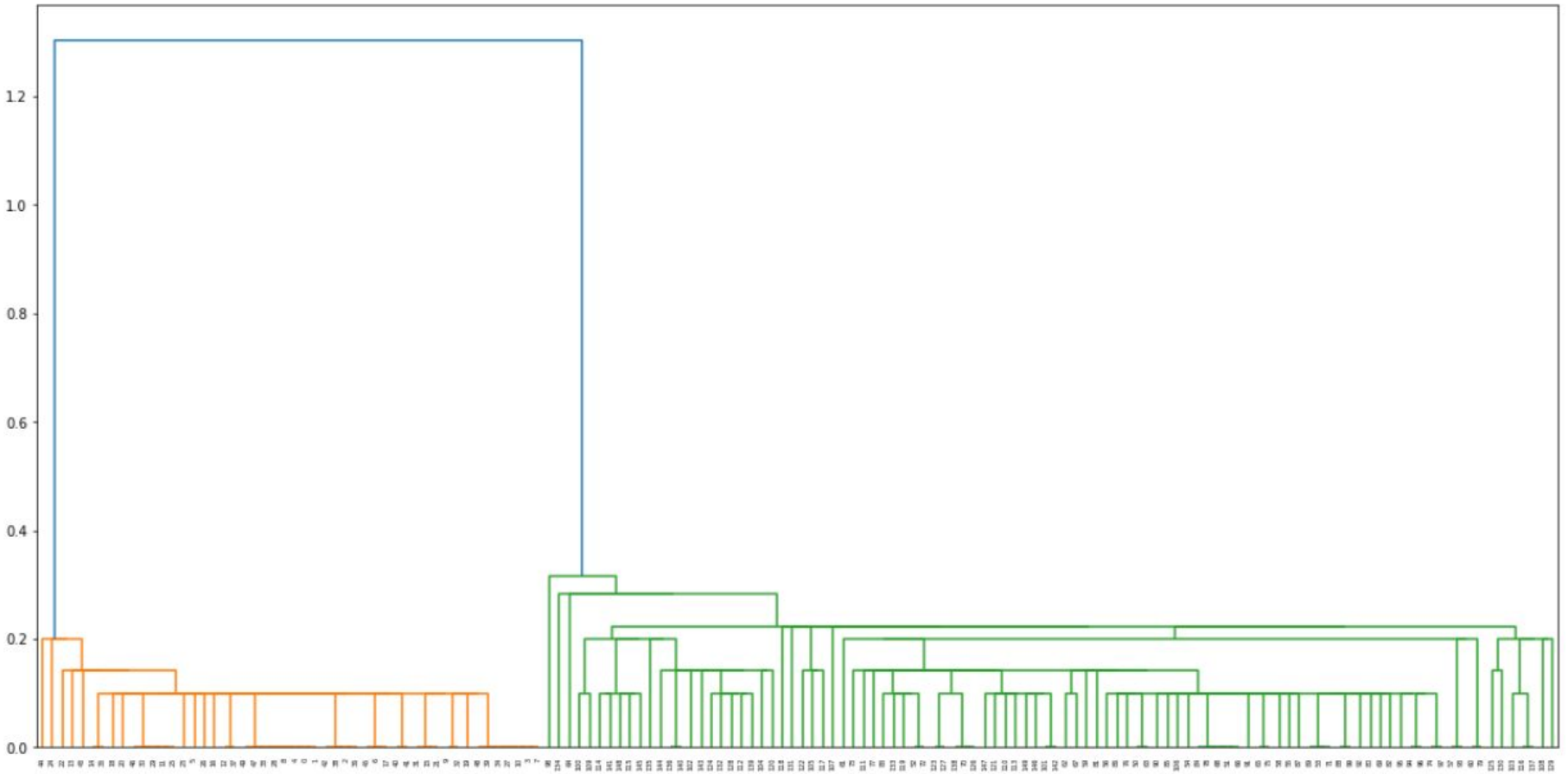
Creating hierarchical clustering



Creating hierarchical clustering

In [9]:
```
hierar=hierarchy.linkage(iris_data_pd.iloc[:,2:4],'single')
plt.figure(figsize=(20,10))
dn=hierarchy.dendrogram(hierar)
```

Creating hierarchical clustering



Creating hierarchical clustering

- Unlike complete linkage, **the single linkage creates hierarchical clustering from the closest elements**. It is highly different from K-means such that the single linkage creates a tree-shaped hierarchical cluster.
- It is important to select an appropriate linkage type to create a model that is suitable for the data condition.

Unit 3.

Non-hierarchical Clustering

3.1. K-means clustering

3.2. Other clustering methods

K-Means Clustering

About the k-means clustering:

- There is one ore more explanatory variables $X1$, $X2$, …, Xd
- There is no response variable. Hence, this is a unsupervised learning algorithm.

Purpose of the k-means clustering:

- Cluster the observations in k clusters.
- Find the centroids (cluster means) that characterize the clusters.

Pros:

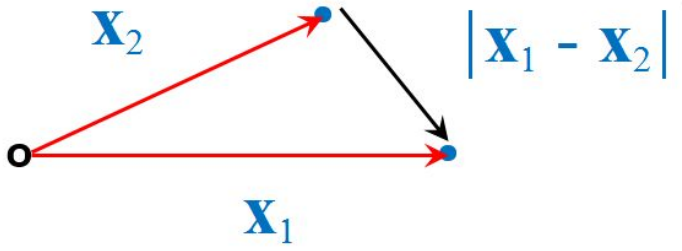
- Intuitive interpretation of the results.
- Quick and easy

Cons:

- Sensitive to the noise and outliers.
- Cluster boundaries can only be linear.

Distance metric:

- Euclidean distance: it is the modulus of the difference between two position vectors.



$$|X_1 - X_2| = \sqrt{(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2 + \cdots + (x_{1d} - x_{2d})^2}$$

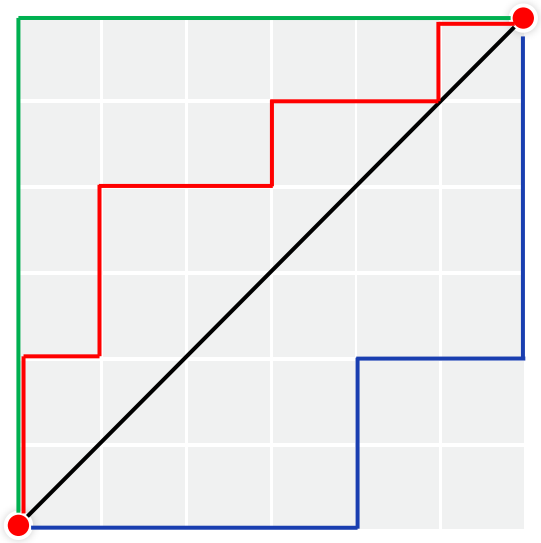
Distance metric:

- For numeric variables:

  Euclidean, Standardized, Mahalanobis, Chebyshev, Canberra, Manhattan, Minkowski, etc.

- For categorical variables:

  Jackard, etc.



Black: Euclidean / Colored: Manhattan.

Standard algorithm:

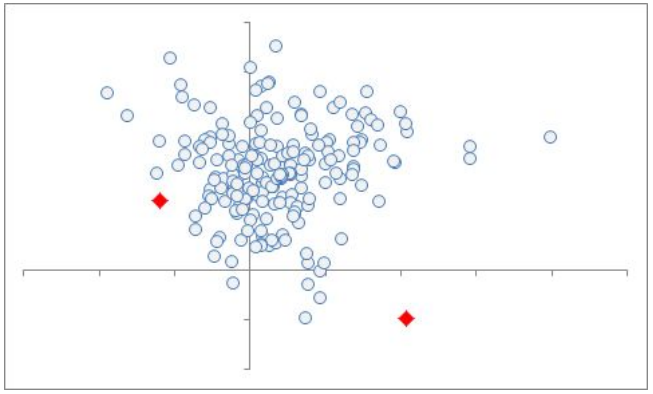
1) Let's suppose a dataset composed of n observations:

$x_1, x_2, \cdots, x_n$

- Each observation xi is a vector of d components (d = number of variables) .

2) Suppose that we target two (k=2) clusters, denoted as C_1 and C_2.

3) Two centroid positions are randomly initialized: μ_1 and μ_2.



Standard algorithm:

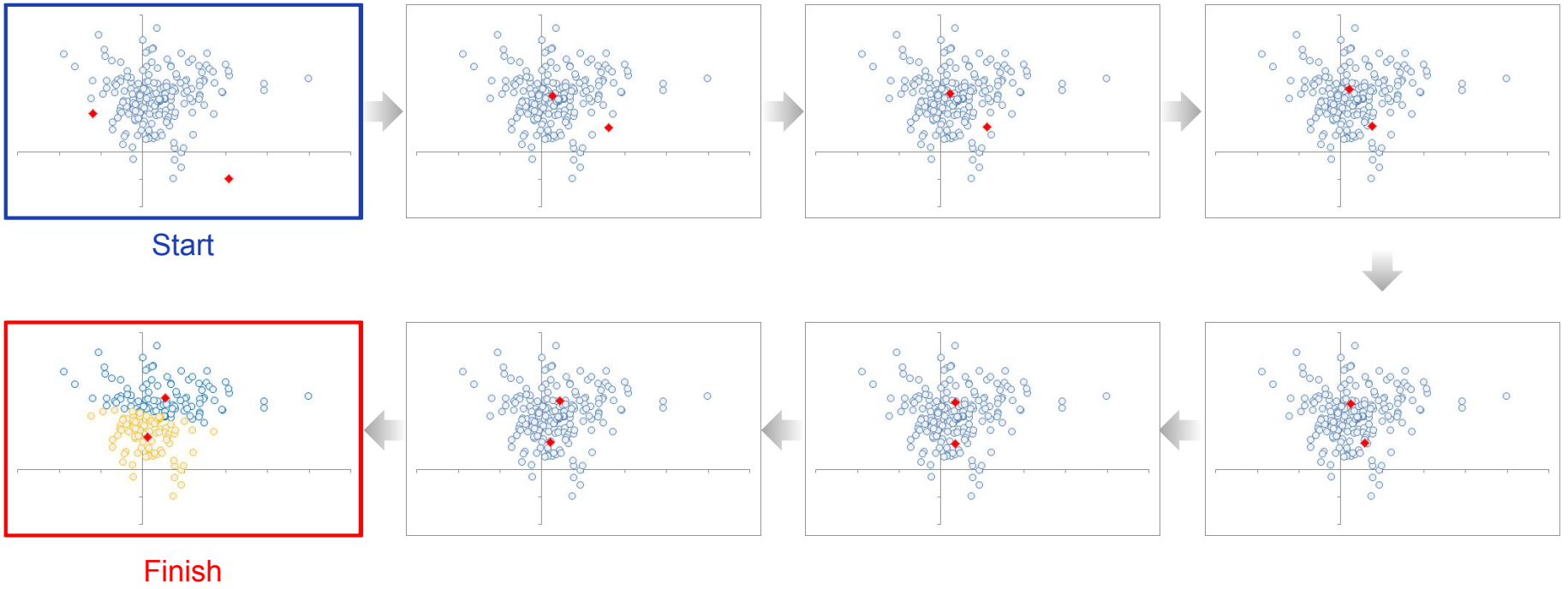
4) Assign the observations to the nearest centroids.

5) Update the centroid positions such that the sum of square distances between the observations and the nearest centroids gets smaller.

$$minimize \sum^{k} \sum |\boldsymbol{x} - \boldsymbol{\mu}_i|^2$$

6) Repeat from the step 4) until the centroids converge to their final positions.

Standard algorithm:

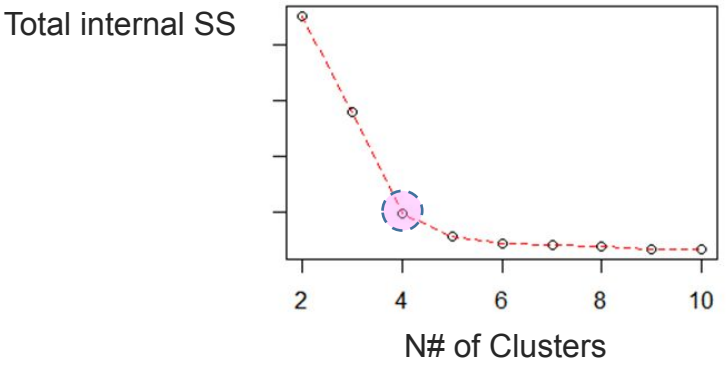


Standard algorithm:

- A question: how do we find the optimal number of clusters?
- Let's define the sum of square distances between the observations and the nearest centroids as:

$$Total\ internal\ sum\ of\ squares = \sum_{i=1}^{k} \sum_{x \in C_i} |\boldsymbol{x} - \boldsymbol{\mu}_i|^2$$

- Plot the "Total internal sum of squares" vs "N# of clusters" and find the inflexion point ← "Elbow".



Coding Exercise #0401

Follow practice steps on 'ex_0401.ipynb' file

Coding Exercise #0402

Follow practice steps on 'ex_0402.ipynb' file

Unit 3.

Non-hierarchical Clustering

3.1. K-means clustering

3.2. Other clustering methods

Other clustering methods

K-Medoids clustering

- It is the advanced version of the K-means clustering method, and it uses all types of similarity and dissimilarity measures. K-Medoids clustering is advantageous for noise or outlier processing because it designates the cluster center by using the actual data set values rather than random dots on plane coordinates.

DBSCAN (Density Based Spatial Clustering of Application with Noise)

- While the K-means clustering method calculates the average of K-clusters and distances between each data point for clustering, DBSCAN applies density to create a same group of data sets that are linked with constant density. It is a clustering method that is advantageous to identify noise and outlier.

Gaussian Mixture Model

- A probability-based clustering analysis that predicts the parameter by using the EM (Expectation Maximization) or MCMC (Markov Chain Monte Carlo) algorithms.

Density-based spatial clustering of applications with noise (DBSCAN)

About the DBSCAN:

- The most common algorithm in density clustering is Density-based Spatial Clustering of Applications with Noise (DBSCAN) Algorithm. Similar to K-means clustering, DBSCAN clustering is affected by data distribution.
- While K-means clustering creates clusters according to the distribution by using the standard data, the DBSCAN clustering is affected by the density of each data. In other words, it assumes that the data to be included in the same cluster would have high density.
- We are going to look at how the DBSCAN determines density.

About the DBSCAN:

- This is a unsupervised learning algorithm: there are only X variables.
- Developed in 1996.
- Allows density-based clustering.

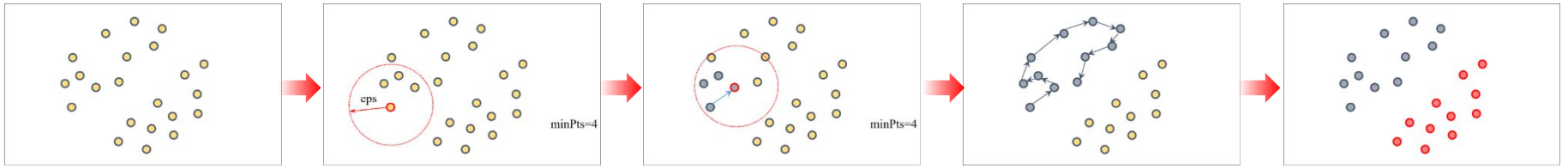
Pros:

- Can reveal structures that the k-means and hierarchical clustering cannot.

Cons:

- Hyperparameters eps and minPts must be specified.
- Hard to get stable clusters in case the dense regions overlap.

DBSCAN algorithm:



(a) (b) (c) (d) (e)

1) Suppose that the observations are distributed as in (a).

2) From a point, count the points within a radius eps and look around: (b).

3) If there are more than minPts points within the circle, include them in the same cluster: (b).

4) Then move on to the next point and repeat from the step 2) until all the points are exhausted: (c)~(d).

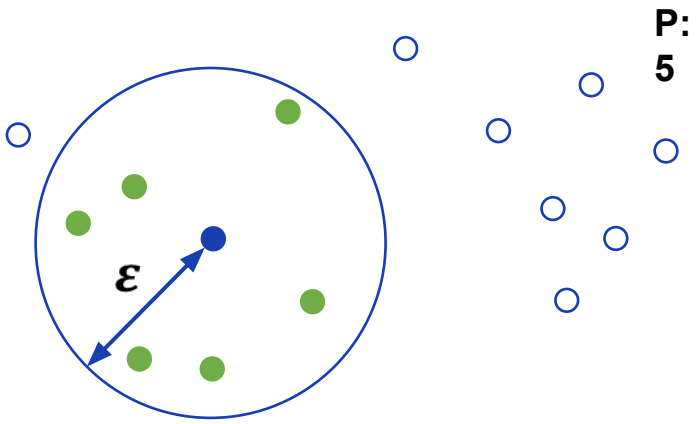
5) Move on to another point that has not been clustered yet. Repeat from the step 2): (e).

How to determine density for DBSCAN algorithm

- Check the following parameters to determine the density for DBSCAN algorithm.

1. ε as radius of the circle with the element in the center
2. P as the minimum number of elements that exist within the radius. The radius distance is measured with the Euclidian distance method.

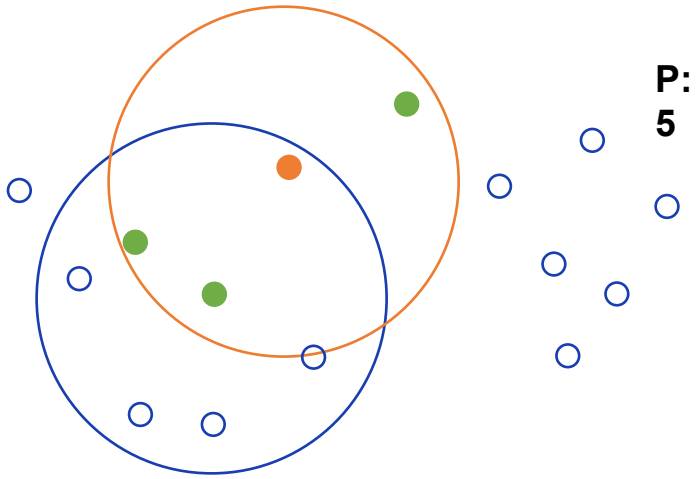
- Look at the following figure. First select a random point.



How to determine density for DBSCAN algorithm

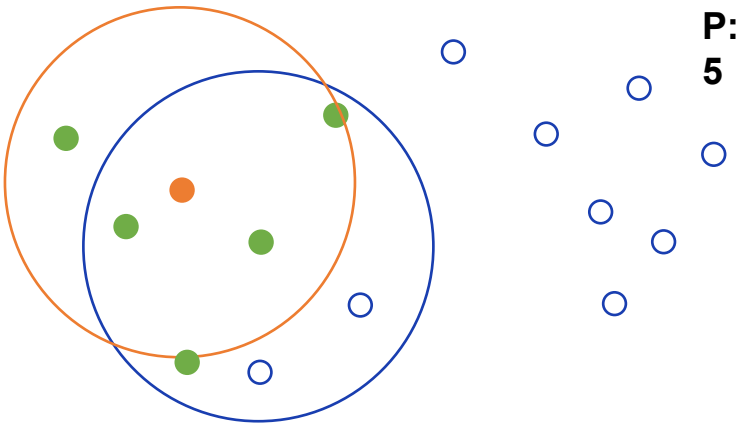
- The randomly selected point is designated at a red dot as shown in the figure in the previous slide.
- The circle is centered to the red dot, and has a radius of ε. Check if the number of elements inside the circle is the same as P or greater than P.
- If it is the same or greater than P, then the red dot becomes the core object and the circle is created as a cluster. If the value is smaller than P, then the red dot is defined as noise.
- In the figure, the elements in the red circle are six green dots. The P equals 5, so the minimum number of elements that should exist within the radius is satisfied.
- Thus, the rod dot becomes the core object and the red circle becomes the cluster. Now, move onto other elements. Select one of the elements in the circle other than the core object.
- Let’s look at the new dot in the next figure.

How to determine density for DBSCAN algorithm



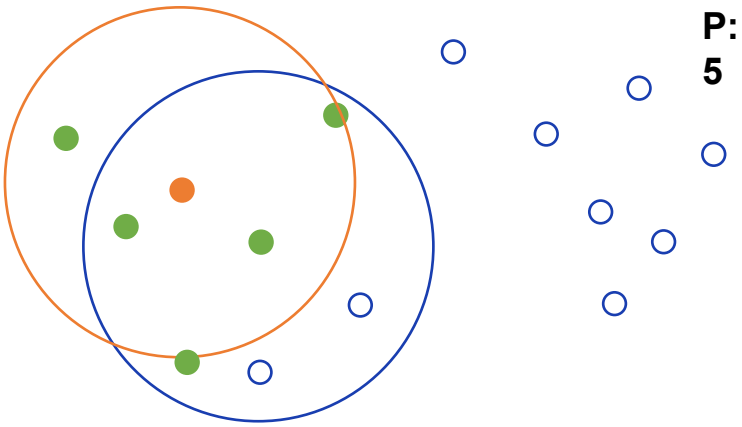
- Blue circle is the existing cluster, and a new dot is designated within the radius. When creating a circle with the red dot on the center, there are only three elements inside it.
- The P value is 5, so the number of elements inside the circle is less than P. So, this dot cannot become the core object. Instead, it is defined as noise.
- Move onto another dot within the radius.

How to determine density for DBSCAN algorithm



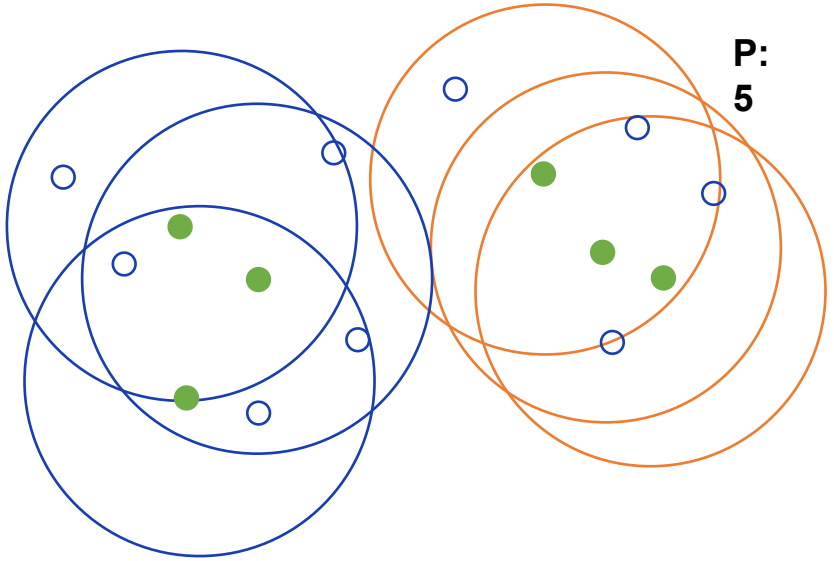
- Now, there are 5 elements in the red circle. It is the same as the P value, so this red dot is recognized as core object.
- It is possible to create a cluster, and make sure to consider if this element was included in the previous cluster as it is now classified as a new core object.
- If included in the previous cluster, do not create a new cluster but expand the existing cluster. If it is not relevant to the previous cluster, a new cluster will be created by using the red dot as a new core object.
- In the figure, the red dot is an element that is present in the blue circle which is the previous cluster. Because it is included in the previously created cluster, the previous cluster is expanded without making a new cluster.

How to determine density for DBSCAN algorithm



- This is the expanded cluster.
- The aim of the DBSCAN algorithm is to check every dot by repeating the same process over and over. After completing all processes, the data cluster looks like follows.

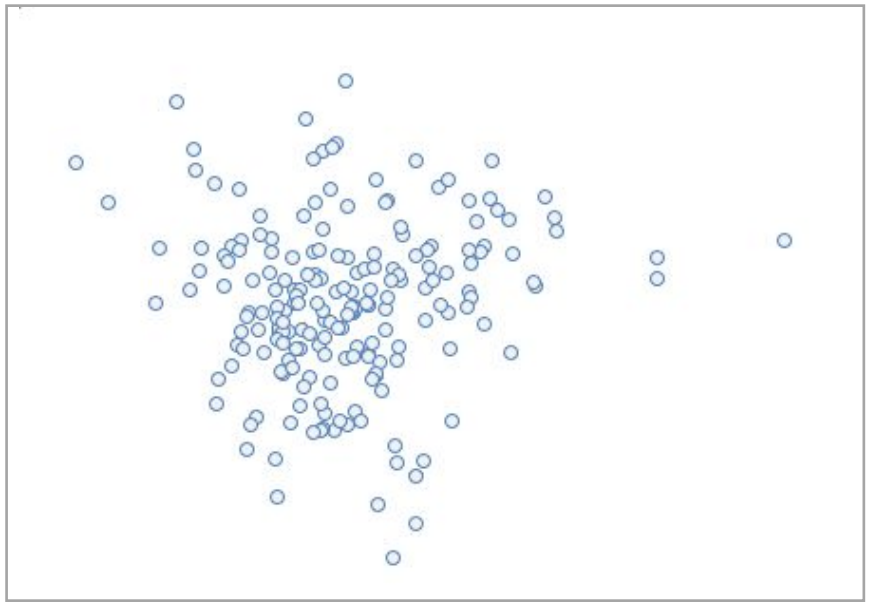
How to determine density for DBSCAN algorithm



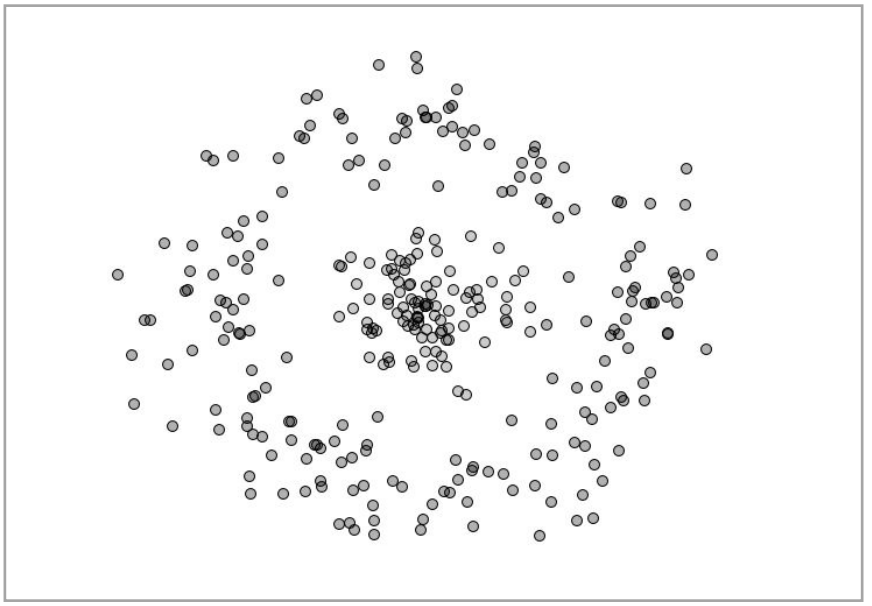
- The black dots are the core objects.
- Likewise, the DBSCAN takes only two parameters and uses them to create a cluster.
- Thus, no other parameters are required other than ε and P values. This is how the density is determined in the DBSCAN algorithm.

Clustering

Question: which clustering algorithms are suitable for the following cases?



(a)



(b)

Coding Exercise #0403

Follow practice steps on 'ex_0403.ipynb' file

Unit 4.

Linear Factor Model for Dimensionality Reduction

- 4.1. Principal Component Analysis
- 4.2. Applications of Principal Components

Principal Component Analysis

Purpose of the principal component analysis (PCA):

- Transform a set of correlated variables into another set of uncorrelated variables.
- Get a new set of orthogonal vectors (principal components or PCs).
- Order the new variables from the largest to the smallest variance.

Pros:

- Many data science applications: preprocessing, modeling, dimensional reduction, visualization, etc.

Cons:

- Difficult to interpret as PCs are obtained by linear combinations of the original features.

Terminology:

a) Loading:

- Normalized principal component (PC).
- There are as many loading vectors as the number of variables.

b) Variance σ^2 (or standard deviation σ):

- The principal components have associated variances.

c) Transformed scores:

- Raw scores (original observations) represented using the PCs as new coordinate axes.

Principal components:

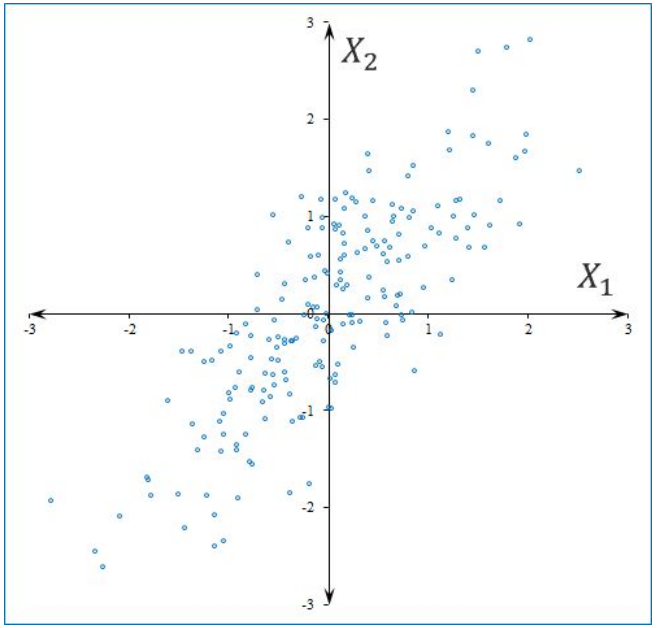
- Let's suppose that there are k variables or features $X1$, $X2$, …, Xk.
- Then, the $PC1$, $PC2$, …, PCk are linear combinations of the original features:

  $PCi=\alpha 1,\ iX1+\alpha 2,\ iX2+\ldots+\alpha k,\ iXk$
- Conversely, the original features can be expressed in terms of the PCs:

  $Xi = \beta 1,\ iPC1 + \beta 2,\ iPC2+\ldots+\beta k,\ iPCk$

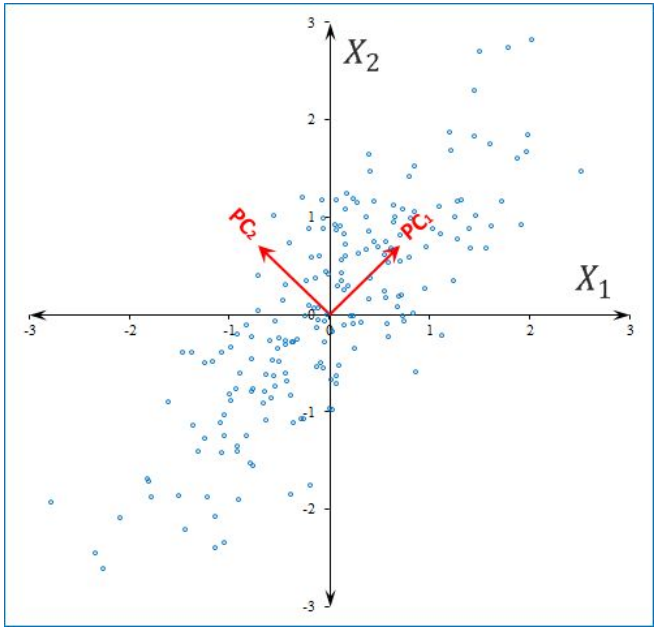
Principal component and variance:

- Suppose a dataset with two variables that can be conveniently visualized on a plane:



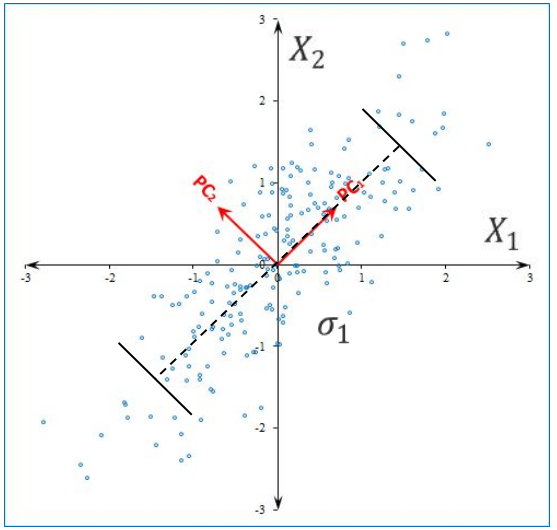
Principal component and variance:

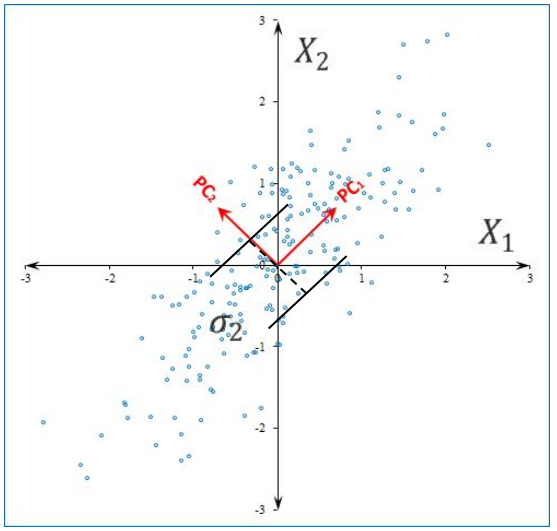
- We can find the $PC1$ and $PC2$ that are orthogonal to each other.



Principal component and variance:

- $PC1$ is the direction of largest variance, while $PC2$ is the direction of the next largest variance.

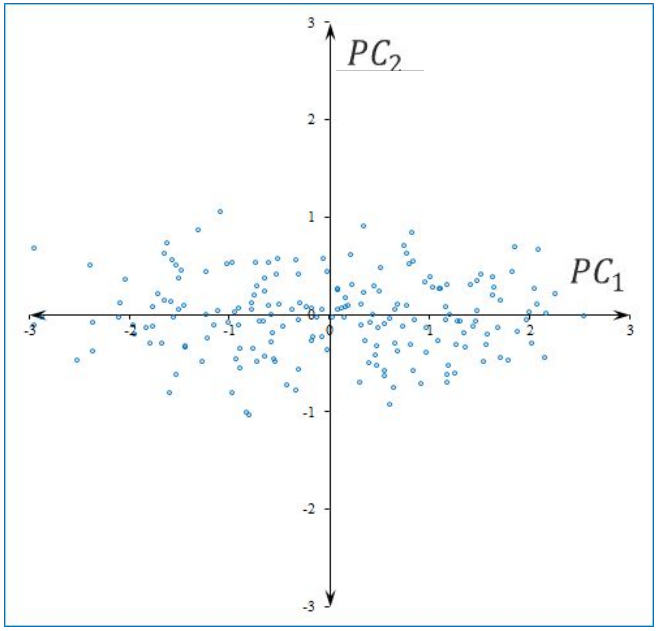




- So, we have $\sigma_1 > \sigma_2$.

Transformed scores:

- The observations can be represented using the $PC1$ and $PC2$ as new coordinate axes.



Cumulative variance:

- As the principal components can be regarded as independent variables, the total variance is:

$$\sigma_{total}^2 = \sigma_1^2 + \sigma_2^2 + \sigma_3^2 + \cdots$$

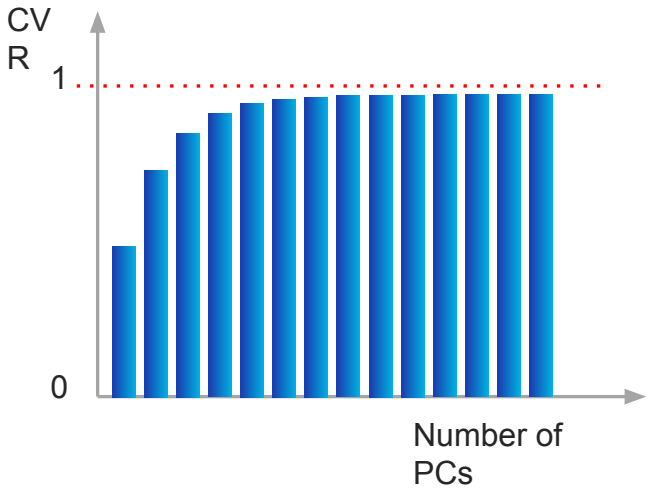
- So, we can calculate the cumulative variance ratios (CVRs):

$$CVR_1 = \frac{\sigma_1^2}{\sigma_{total}^2}$$

$$CVR_2 = \frac{\sigma_1^2 + \sigma_2^2}{\sigma_{total}^2}$$

$$CVR_3 = \frac{\sigma_1^2 + \sigma_2^2 + \sigma_3^2}{\sigma_{total}^2}$$

$$\vdots$$



Calculating the principal components:

- The PCs can also be obtained by eigenvalue decomposition (ED) of the covariance matrix.
- The PCs can be calculated by singular value decomposition (SVD) of the data matrix.
- If we standardize the variables, then instead of the covariance we would have the correlation.

Ex A covariance matrix and a correlation matrix.

$$\begin{bmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} & \sigma_{14} \\ \sigma_{21} & \sigma_2^2 & \sigma_{23} & \sigma_{24} \end{bmatrix}$$

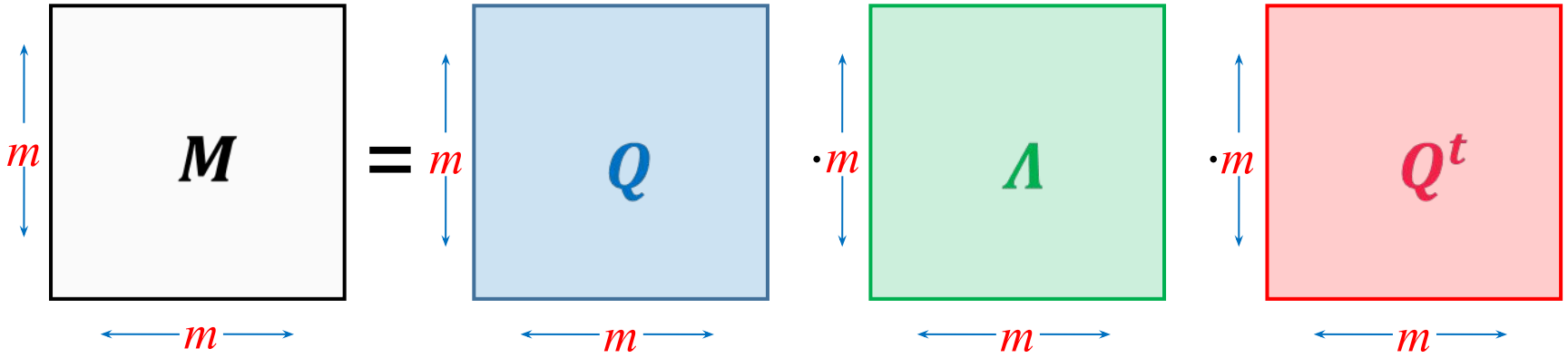
$$\begin{bmatrix} 1 & \rho_{12} & \rho_{13} & \rho_{14} \\ \rho_{21} & 1 & \rho_{23} & \rho_{24} \end{bmatrix}$$

“In the next few slides, let’s make a detour and review the ED and SVD in detail”

Matrix Decompositions

Eigenvalue decomposition (ED):

- A square matrix $\boldsymbol{M}$ is decomposed as $\boldsymbol{M}=\boldsymbol{Q}\,\boldsymbol{\Lambda}\,\boldsymbol{Q}^t$.
- All the matrices have the same size: $Size(\boldsymbol{M})=Size(\boldsymbol{Q})=Size(\boldsymbol{\Lambda})=m\times m$.



Eigenvalue decomposition (ED):

- A square matrix $\boldsymbol{M}$ is decomposed as $\boldsymbol{M}=\boldsymbol{Q}\,\boldsymbol{\Lambda}\,\boldsymbol{Q}^{t}$.
- Here, $\boldsymbol{\Lambda}$ is a diagonal matrix that contains the “eigenvalues”.

$$\boldsymbol{\Lambda}=\begin{bmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & & 0\\ \vdots & & \ddots & \vdots\\ 0 & 0 & \cdots & \lambda_m\end{bmatrix}$$

Eigenvalue decomposition (ED):

- A square matrix $\boldsymbol{M}$ is decomposed as $\boldsymbol{M}=\boldsymbol{Q}\,\boldsymbol{\Lambda}\,\boldsymbol{Q}^{t}$.
- The columns of $\boldsymbol{Q}$ are the so-called “eigenvectors”.

$$\boldsymbol{Q}=\begin{bmatrix}\uparrow & \cdots & \uparrow\\ \boldsymbol{q_1} & \cdots & \boldsymbol{q_m}\\ \downarrow & \cdots & \downarrow\end{bmatrix}$$

- Between an eigenvector and its eigenvalue we have the following relation:

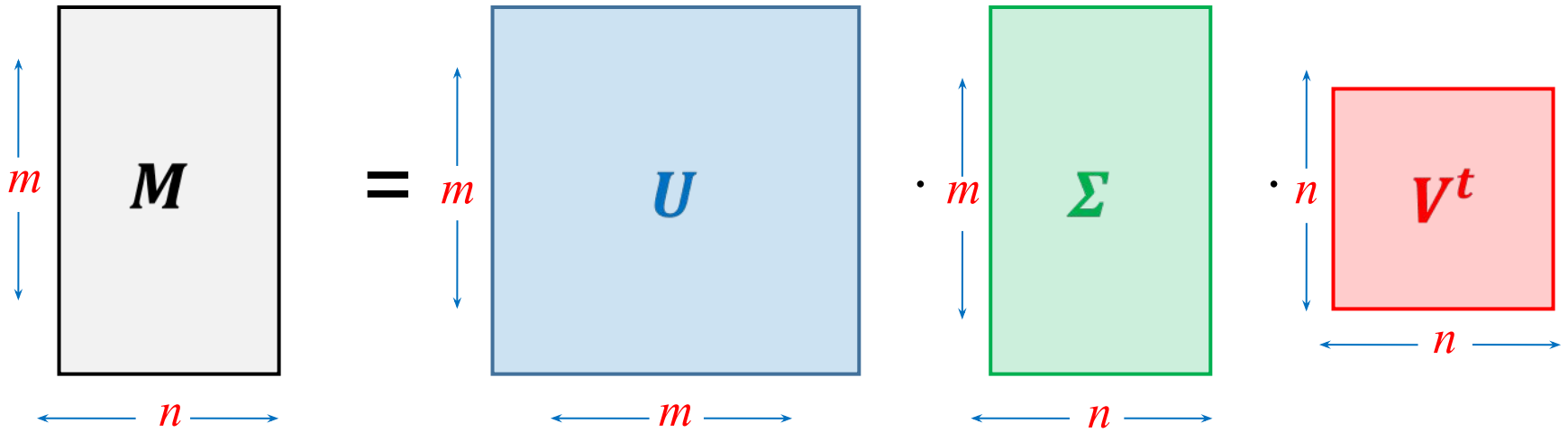
$$\boldsymbol{M}\boldsymbol{q_i}=\lambda_i\,\boldsymbol{q_i}$$

- Between any two eigenvectors, we have the following orthogonality condition:

$$\boldsymbol{q_i}\cdot\boldsymbol{q_j}=\delta_{ij}\quad\Leftrightarrow\quad\boldsymbol{Q}\boldsymbol{Q}^{\boldsymbol{t}}=\boldsymbol{Q}^{\boldsymbol{t}}\boldsymbol{Q}=\boldsymbol{I}$$

Singular value decomposition (SVD):

- A matrix M is decomposed as $M=U\Sigma V^t$.



Singular value decomposition (SVD):

- A matrix $\boldsymbol{M}$ is decomposed as $\boldsymbol{M}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^t$.
- Here, $\boldsymbol{\Sigma}$ contains the singular values as diagonal elements.
- These singular values are ordered from the largest to the smallest: $\sigma_1>\sigma_2>\ldots>\sigma_m$

$$\boldsymbol{\Sigma} = \begin{bmatrix} 0 & \sigma_2 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & \sigma_m & 0 \end{bmatrix}$$

Singular value decomposition (SVD):

- A matrix M is decomposed as $M=U\Sigma V^t$.
- The columns of U are the “left singular vectors”.
- The columns of V are the “right singular vectors”.

$$= \begin{bmatrix} \uparrow & \cdots & \uparrow \\ u_1 & \cdots & u_m \\ \downarrow & \cdots & \downarrow \end{bmatrix} \qquad V = \begin{bmatrix} \uparrow & \cdots & \uparrow \\ v_1 & \cdots & v_n \\ \downarrow & \cdots & \downarrow \end{bmatrix}$$

- Between a set of the left and right singular vectors and their singular value we have:

$$= M\, v_i = \sigma_i\, u_i$$

- Between any two singular vectors, we have the following orthogonality condition:

$$v_i \cdot v_j = \delta_{ij} \quad \Leftrightarrow \quad VV^t = V^tV = I$$
$$u_i \cdot u_j = \delta_{ij} \quad \Leftrightarrow \quad UU^t = U^tU = I$$

Relation between ED and SVD:

1) If X is a matrix composed of standardized scores, then we can apply SVD: $X=U\Sigma Vt$.

2) Using the decomposed form, we can express the covariance matrix as following:

$$= U\Sigma V^t \, V\Sigma^t U^t$$
$$= U\Sigma\Sigma^t U^t$$

} By orthogonality $V^tV = I$

- We can see that $\Sigma\Sigma^t$ is a diagonal matrix with elements equal to σi^2.
- Thus, $\Sigma\Sigma^t$ is the same as the eigenvalue matrix Λ.
- Also, we can see that the left singular matrix U is the same as the eigenvector matrix Q.

Relation between ED and SVD:

- Let’s summarize in the following table.

ED	SVD

Principal Component Analysis

Calculating the principal components:

- Finally, let us summarize the PCA in terms of the matrix decompositions.

	ED	SVD
Decomposition Target		

Unit 4.

Linear Factor Model for Dimensionality Reduction

4.1. Principal Component Analysis

4.2. Applications of Principal Components

Dimensional Reduction

About the dimensional reduction:

- The number of principal components (PCs) is equal to the number of variables, say k.
- The original variables X_i can be expressed in terms of the PCs:

$$X_i = \beta_{1,i}PC_1 + \beta_{2,i}PC_2 + \cdots + \beta_{k,i}PC_k$$

- The PCs are ordered by variance: $\sigma_1^2 > \sigma_2^2 > \sigma_3^2 > \cdots > \sigma_k^2$
- So we can reduce dimension starting from the last PC, ($q < k$):

$$X_i \approx \beta_{1,i}PC_1 + \beta_{2,i}PC_2 + \cdots + \beta_{q,i}PC_q$$ "Reduced dimension input"

Pros:

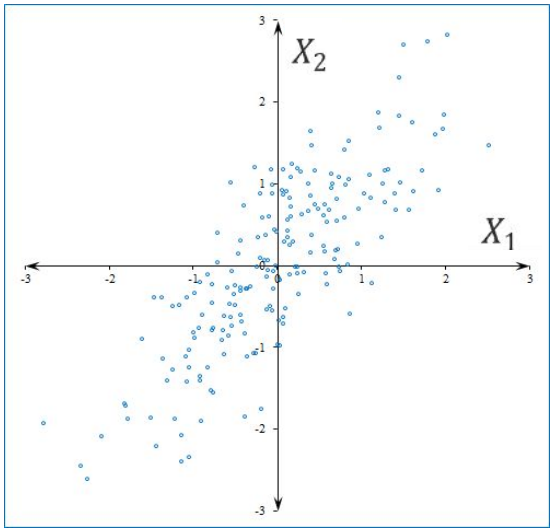
- We can simplify the data and reduce overfitting error.
- We get only the most salient features.

Cons:

- Loss of details.
- Difficult to interpret intuitively.

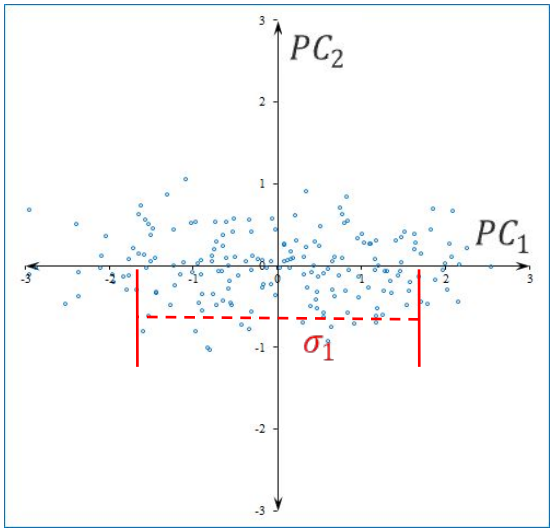
Dimensional reduction:

- Suppose a dataset with two variables that can be conveniently visualized on a plane:



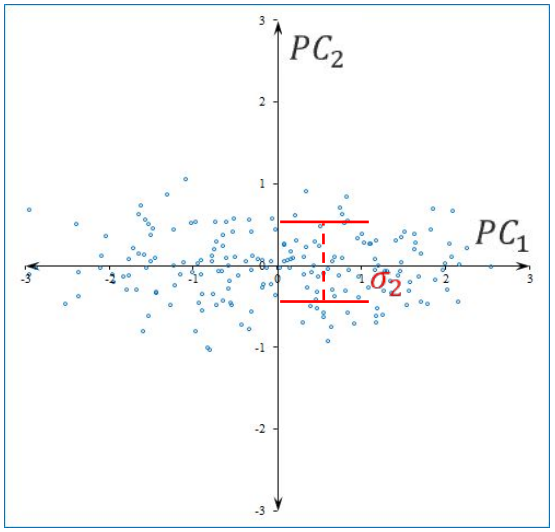
Dimensional reduction:

- The observations can be represented using the PC_1 and PC_2 as new coordinate axes:



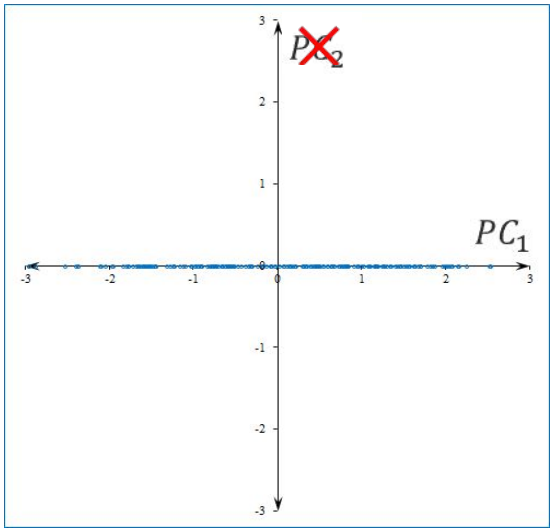
Dimensional reduction:

- The observations can be represented using the PC_1 and PC_2 as new coordinate axes:



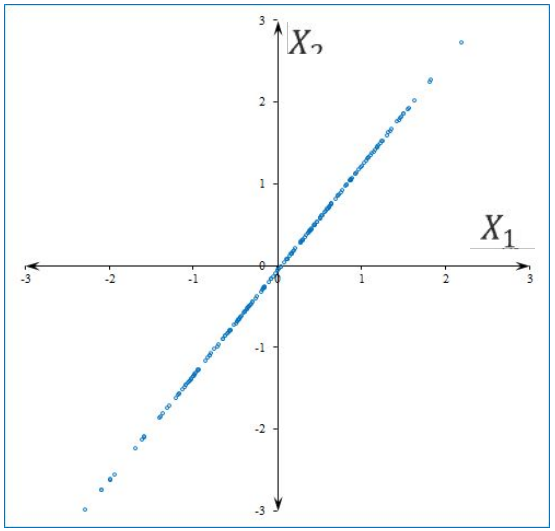
Dimensional reduction:

- We can eliminate the direction represented by PC_2 that corresponds to the smaller variance:



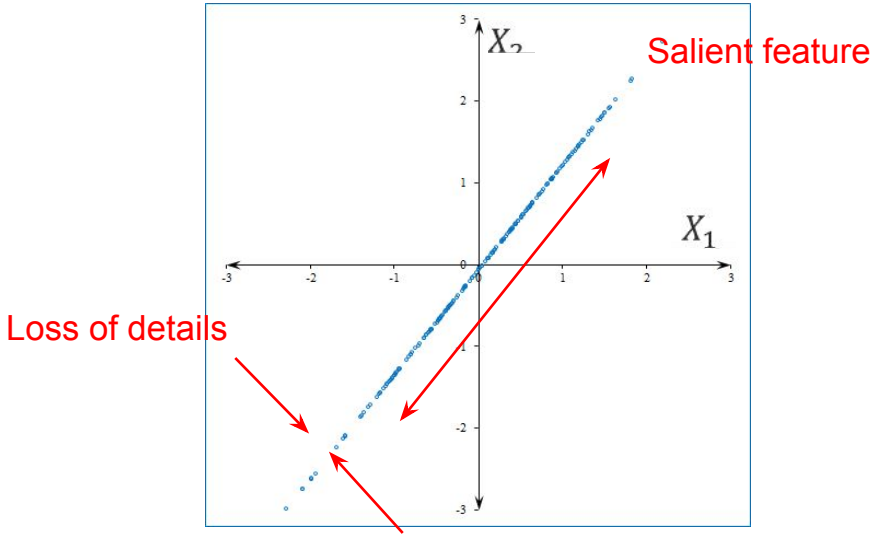
Dimensional reduction:

- Now, we can go back to the original coordinate system and show the “reduced dimensional input”:



Dimensional reduction:

- Now, we can go back to the original coordinate system and show the “reduced dimensional input”:



- We can notice that details have been lost leaving only the most salient feature.

Dimensional reduction:

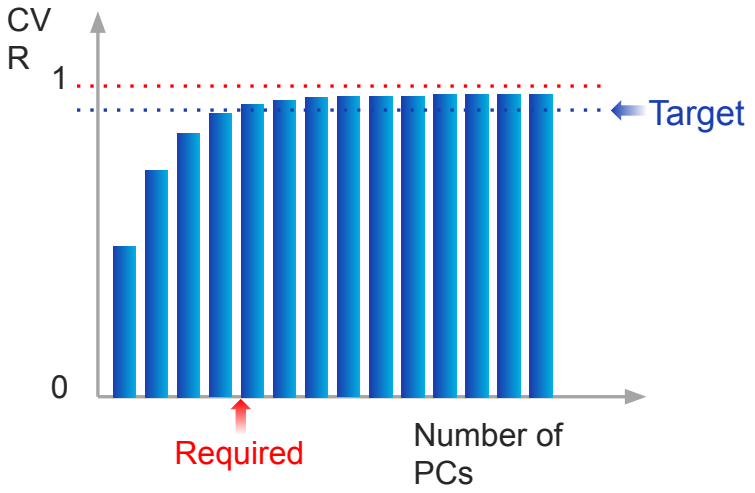
- The total variance is: $\sigma_{total}^2 = \sigma_1^2 + \sigma_2^2 + \sigma_3^2 + \cdots + \sigma_k^2$
- Then, we can calculate the cumulative variance ratios:

$$CVR_1 = \frac{\sigma_1^2}{\sigma_{total}^2}$$

$$CVR_2 = \frac{\sigma_1^2 + \sigma_2^2}{\sigma_{total}^2}$$

$$CVR_3 = \frac{\sigma_1^2 + \sigma_2^2 + \sigma_3^2}{\sigma_{total}^2}$$

$$\vdots$$

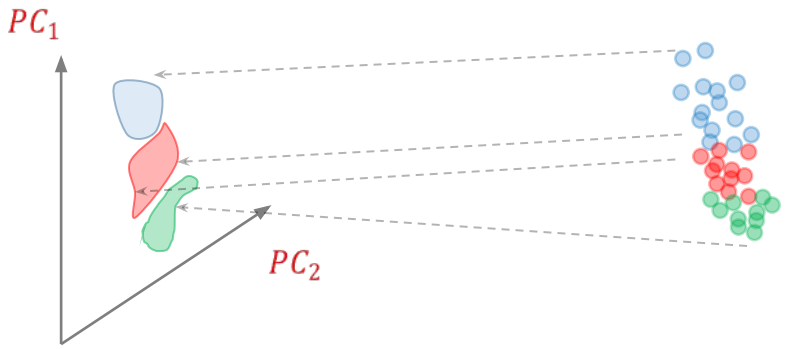


- We can set a target CVR and determine the required number of PCs.

High Dimension Visualization

High dimension visualization:

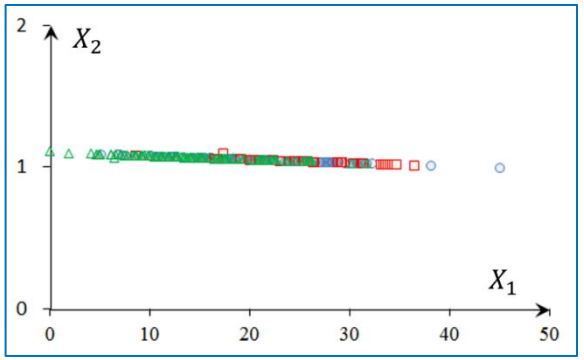
- PC_1 and PC_2 are the directions of the largest and the second largest variance.
- PC_1 and PC_2 define the most spread out plane on which to project the high dimensional coordinates.

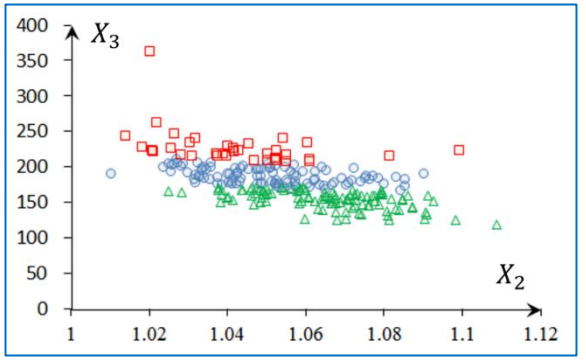


- Easy to implement: use only the first two components of the transformed scores.

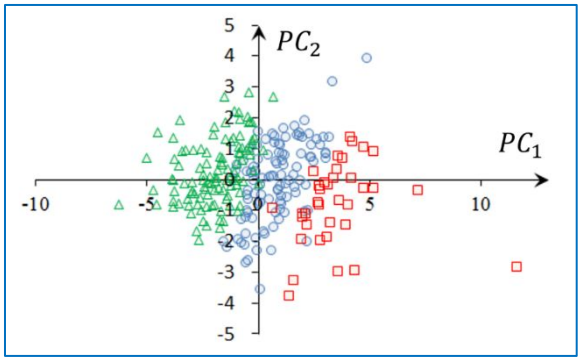
High dimension visualization:

Ex





Projected onto the original variable set.

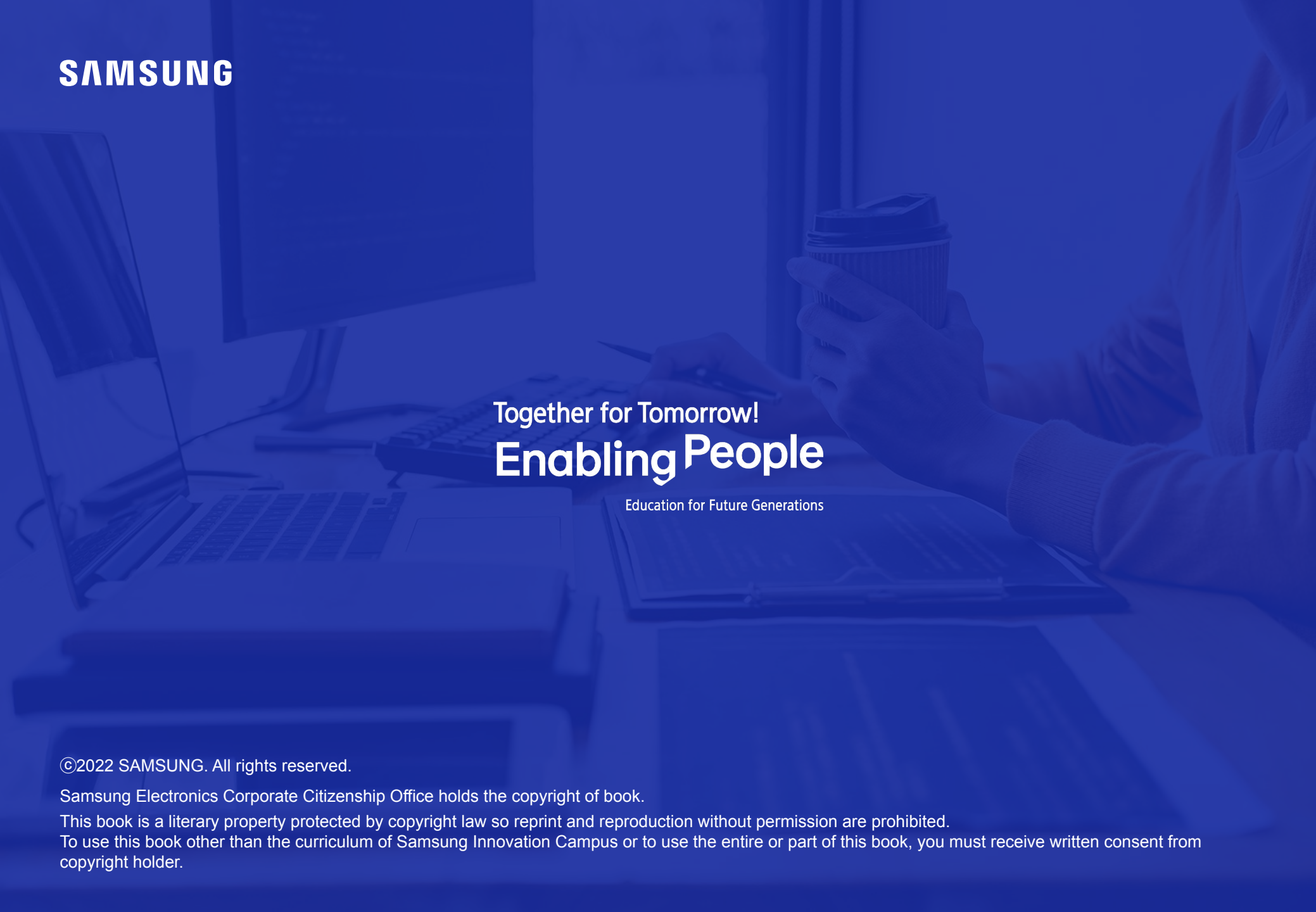


Projected onto the plane defined by PC_1 and PC_2.

Coding Exercise #0404

Follow practice steps on 'ex_0404.ipynb' file


End of
Document



ⓒ2022 SAMSUNG. All rights reserved.

Samsung Electronics Corporate Citizenship Office holds the copyright of book.

This book is a literary property protected by copyright law so reprint and reproduction without permission are prohibited.
To use this book other than the curriculum of Samsung Innovation Campus or to use the entire or part of this book, you must receive written consent from copyright holder.