INFSCI 2915: Machine Learning

Unsupervised Learning: Dimensionality Reduction & Clustering

Mai Abdelhakim

School of Computing and Information

610 IS Building

Spring 2018

This Unit

- Unsupervised Learning
  - Dimensionality Reduction
    - PCA
    - t-SNE
  - Clustering
    - K-Means
    - Hierarchical (Agglomerative)
    - DBSCAN

Supervised vs. Unsupervised Learning

- Supervised Learning: both X (features) and Y (labels) are known
- Unsupervised Learning: only X (features) available

X Sample		Y Target Value (Label)	
	x_1	Apple	y_1
	x_2	Lemon	y_2
	x_3	Apple	y_3
	x_4	Orange	y_4

Supervised Learning

X Sample	
	x_1
	x_2
	x_3
	x_4

Unsupervised Learning

Unsupervised Learning

- With unsupervised learning:
  - Is there an efficient way to **visualize** the data?
  - **Can we find subgroups** among the observations?
    - Are there interesting patterns?
- **Hard to assess the performance**
  - How to do that without ground truth labels?
  - More challenging, and subjective

Dimensionality Reduction

- Why needed?
  - **Preprocessing for unsupervised learning**: reduce overfitting, less complex
  - Can also be used for data **visualization**
    - Observations with p feature, if we want to **visualize** the observations
      - We can see pair-plots: p(p-1)/2 plots! ➔ difficult to visualize
      - Instead, **find low dimensional representation that captures as much information as possible**
- Unsupervised Dimensionality Reduction approaches: No labels used (covered)
  - Ex. PCA, t-SNE (mainly visualization)
- Supervised Dimensionality Reduction Approaches (Not covered):
  - Ex. LDA

Principle Component Analysis (PCA)

- Principle Components: **Smaller number of representative features** that **explain most of the variability in the original data**

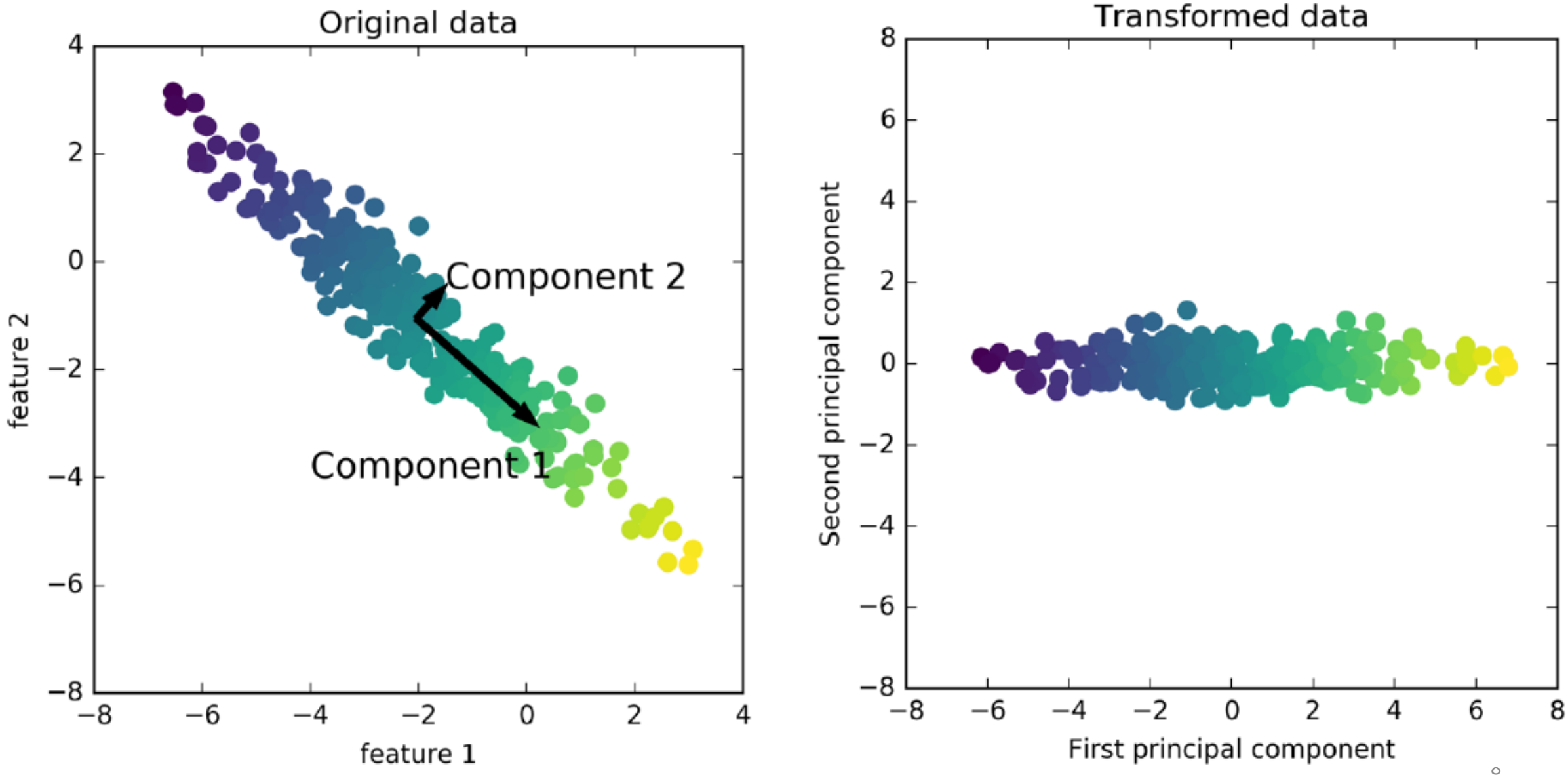
- Used as preprocessing for supervised learning and for visualization
  - After we get the principle components, we can use them instead of the original features for supervised learning.

PCA

- PCA produces a low-dimensional representation of a dataset
  - It finds a sequence **of linear combinations of the original features** that have **maximal variance**, and are **mutually uncorrelated**

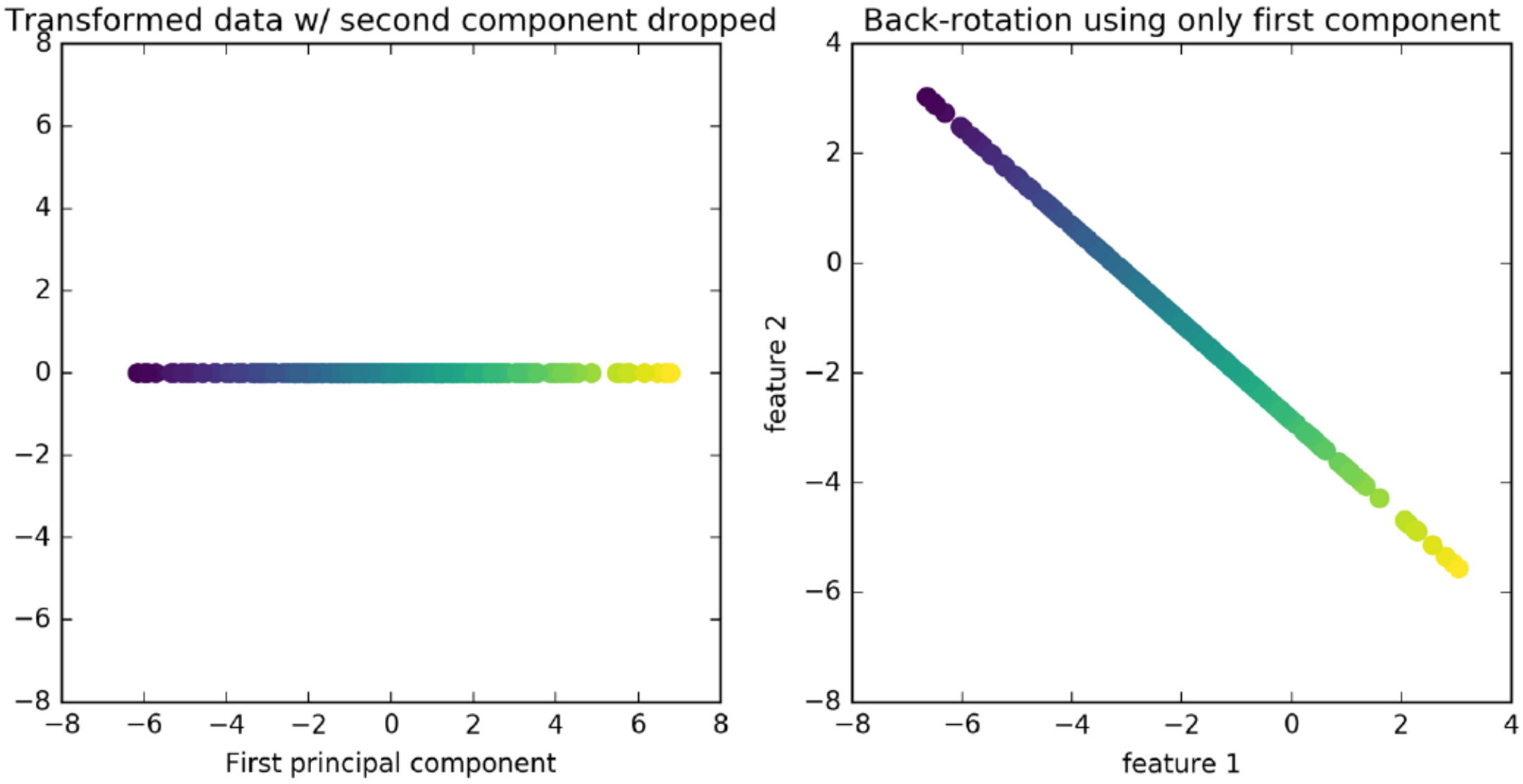
Example: p=2,

The principle component loading vectors are the direction in the feature space where the data varies the most



Muller et al., Introduction to Machine Learning with Python

Example: p=2 … cont..



Muller et al., Introduction to Machine Learning with Python

PCA

- If original features are X_1, X_2,…;Xp , then **the first principle component** (Z_1) is the normalized **linear combination of features** that has the **largest variance:**

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \ldots + \phi_{p1}X_p$$

Coefficients $\phi_{11}, .. \phi_{p1}$ are called the loading of the principle component Z_1

- Normalized means: $\sum_{j=1}^{p} \phi_{j1}^2 = 1.$
  - Without this normalization the variance can be large due to $\phi's$ and not due to data (X)
- The **first principle component loading vector**

$$\phi_1 = (\phi_{11}, \phi_{21}, .. \phi_{p1})^T$$

- The second principle component (Z_2) has the next **maximal variance** out of **all linear combinations** of (X_1,..X_p) that are **uncorrelated** with Z_1

  - The second principle component loading vector

$$\phi_2 = (\phi_{12}, \phi_{22}, .. \phi_{p2})^T$$

- An so on, we can have up to M principle components (vectors), $M \leq p$

Example: Breast Cancer Data Set

Muller et al., Introduction to Machine Learning with Python

- Not very easy to visualize histogram of 30 features in cancer dataset
- We can use PCA and only 2 features
- The data is well separated with only 2 principle components!
  - Simple linear classifier would do well

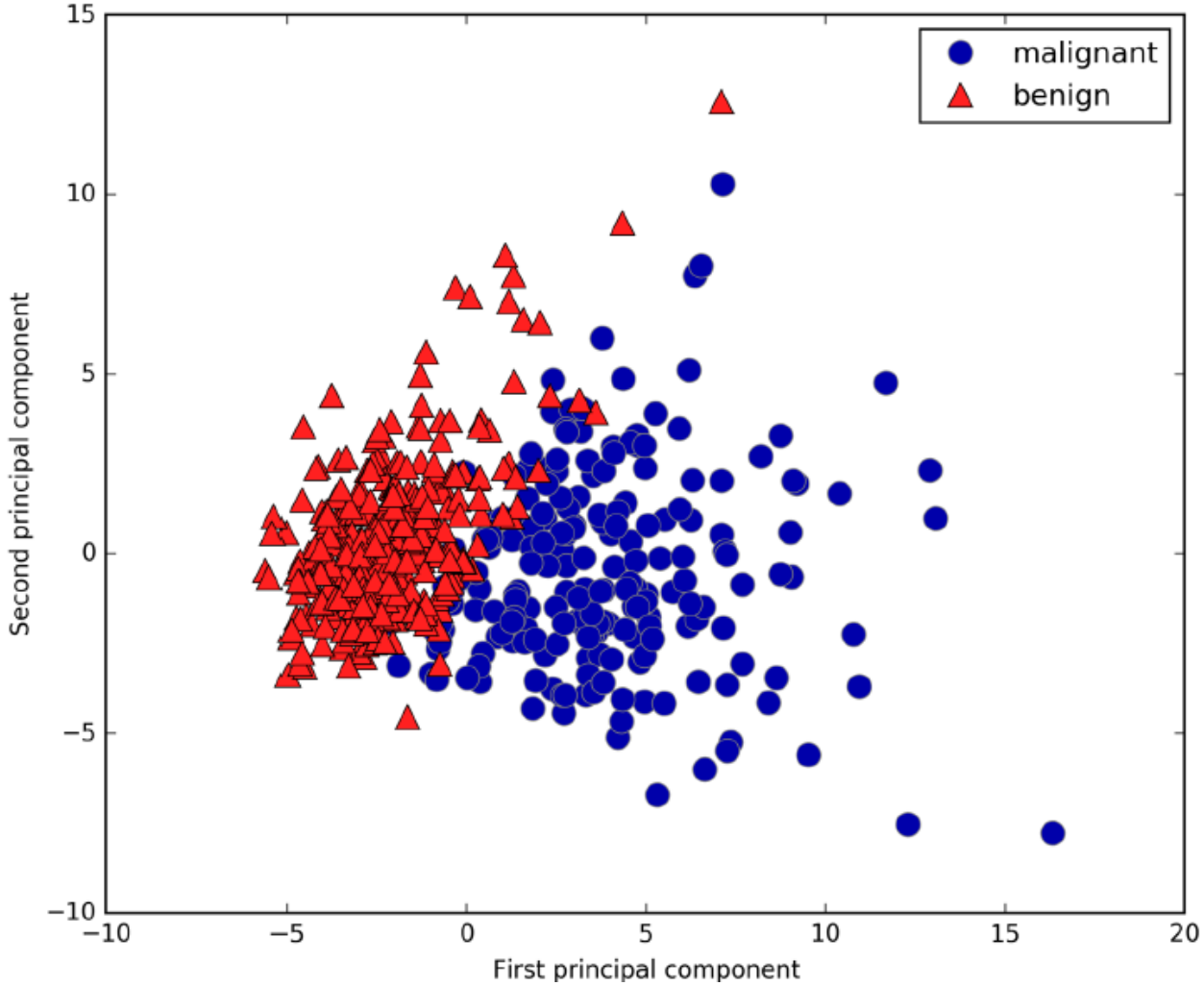
$$\phi_{11}, .. \phi_{p1}$$

```
PCA components:
[[ 0.219  0.104  0.228  0.221  0.143  0.239  0.258  0.261  0.138  0.064
   0.206  0.017  0.211  0.203  0.015  0.17   0.154  0.183  0.042  0.103
   0.228  0.104  0.237  0.225  0.128  0.21   0.229  0.251  0.123  0.132]
```

$$\phi_{12}, .. \phi_{p2}$$

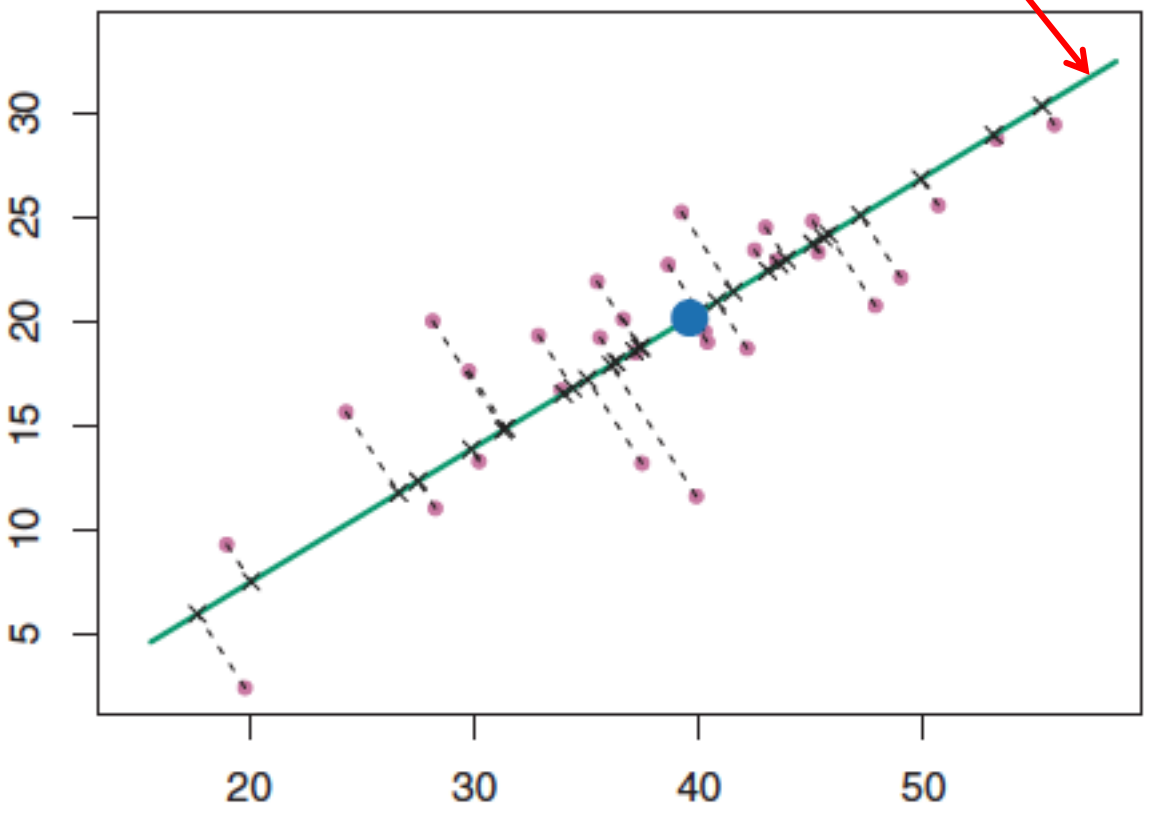
```
[-0.234 -0.06  -0.215 -0.231  0.186  0.152  0.06  -0.035  0.19   0.367
 -0.106  0.09  -0.089 -0.152  0.204  0.233  0.197  0.13   0.184  0.28
 -0.22  -0.045 -0.2   -0.219  0.172  0.144  0.098 -0.008  0.142  0.275]]
```

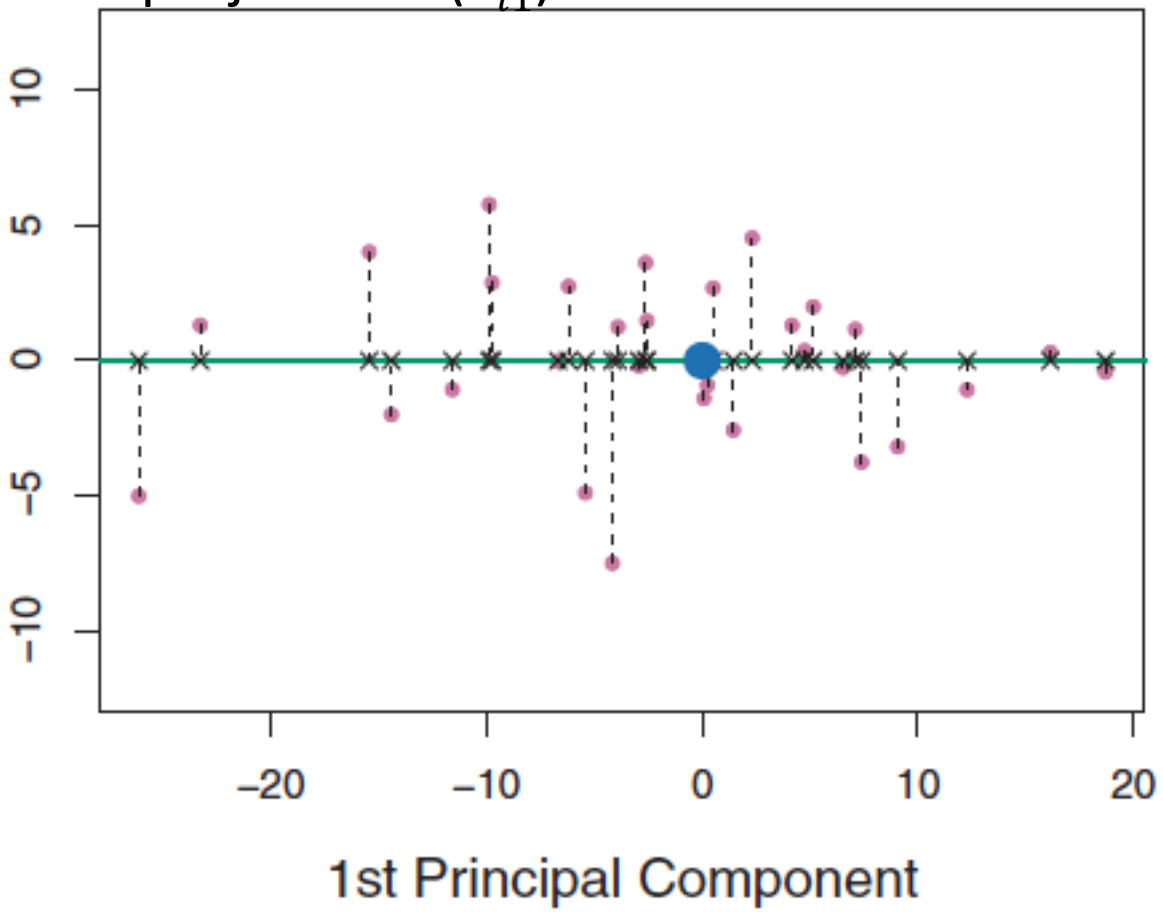
Elements in the 2 vectors are loadings of the 2 principle components on the 30 features



Direction of the first principle component, represented by:
$\phi_1 = (\phi_{11}, \phi_{21}, .. \phi_{p1})^T$

Each sample is represented in the new low dimensional features space through projections (z_{i1})





Problem Formulation for First Principle Component

- **In PCA, X is generally normalized to zero mean and unit variance**
  - **PCA is sensitive to feature scaling**

- **Principle components also have zero mean**

- Finding the first principle component: **find loadings the maximizes the variance of Z_1**
  - This can be formulated as the following optimization

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{maximize}} \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j1}x_{ij}\right)^2 \text{ subject to } \sum_{j=1}^{p}\phi_{j1}^2 = 1$$

  - Similarly, problem can be formulated to get second principle component
  - Solved by eigen-decomposition of X:

- The **principle components** are the **eigenvectors** corresponding to the **largest eigenvalues** of the pxp **covariance matrix Σ of features X**

- Then we transform each observation to M dimensional space
- Matrix form:

$$[z_{i1}, z_{i2} \dots z_{iM}] = [x_{i1}, x_{i2} \dots x_{ip}]\ \mathrm{W}$$

  - Column j of the W matric is loading vector $\phi_j = (\phi_{1j}, \phi_{2j}, .. \phi_{pj})^T$
    - X is n x p matrix
    - W is p x M matrix: M eigenvectors (principle components) in direction of largest variance
    - Z is n x M matrix

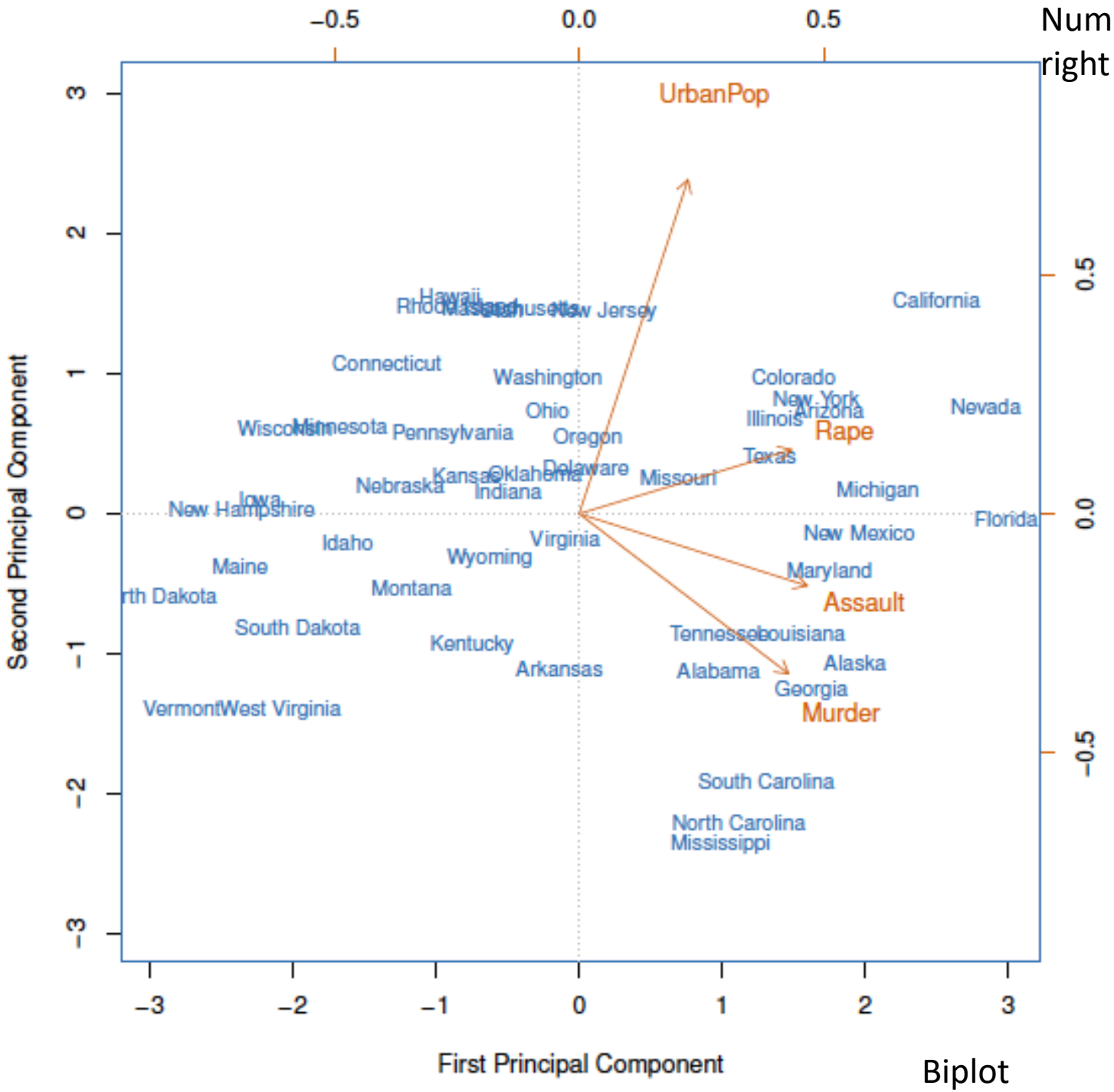
p-dimensional feature space ➔ M dimensional space

Example: USAarrests data

- For each of the **50 states** in the United States, the set contains:
  - The **number of arrests** per 100,000 residents for each of **three crimes**: Assault, Murder, Rape.
  - It also contains **UrbanPop** feature (the percent of the population in each state living in urban areas).

- Thus, n=50, p=4 (Assault, Murder, Rape, UrbanPop)

- Apply PCA, get two principle components (PC1 PC2) with and without scaling



Biplot

Numbers in table represent loadings/weights (also in top and right axes)

	PC1	PC2
Murder	0.5358995	-0.4181809
Assault	0.5831836	-0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

Equal weights on all crimes, less weigh on population

High weight on population

PC1: roughly measures overall rate of serious crimes, and PC2 roughly measures population (level of urbanization)

Proportion of Variance Explained

- How much of the data is contained in the first few principle components?
  - We select just few principle components to represent the data
  - How much of the variance in the data in contained in the first few principle components

- We need to know the proportion of variance explained (PVE) by each principle component?

Proportion of Variance Explained

- Total variance in the data (all p features) is

$$\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$$

  - Note data is centered around zero mean so we took out the mean from the equation

- The **variance explained by a principle component Z_m** is:

$$\mathrm{Var}(Z_m) = \frac{1}{n} \sum_{i=1}^{n} z_{im}^2$$

- You can show **that <u>if we take all possible principle components (all eigenvectors)</u> then the total variance is the same (all variance is explained)**

If $p < n$, then maximum $M = p$ $\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{m=1}^{M} \mathrm{Var}(Z_m)$

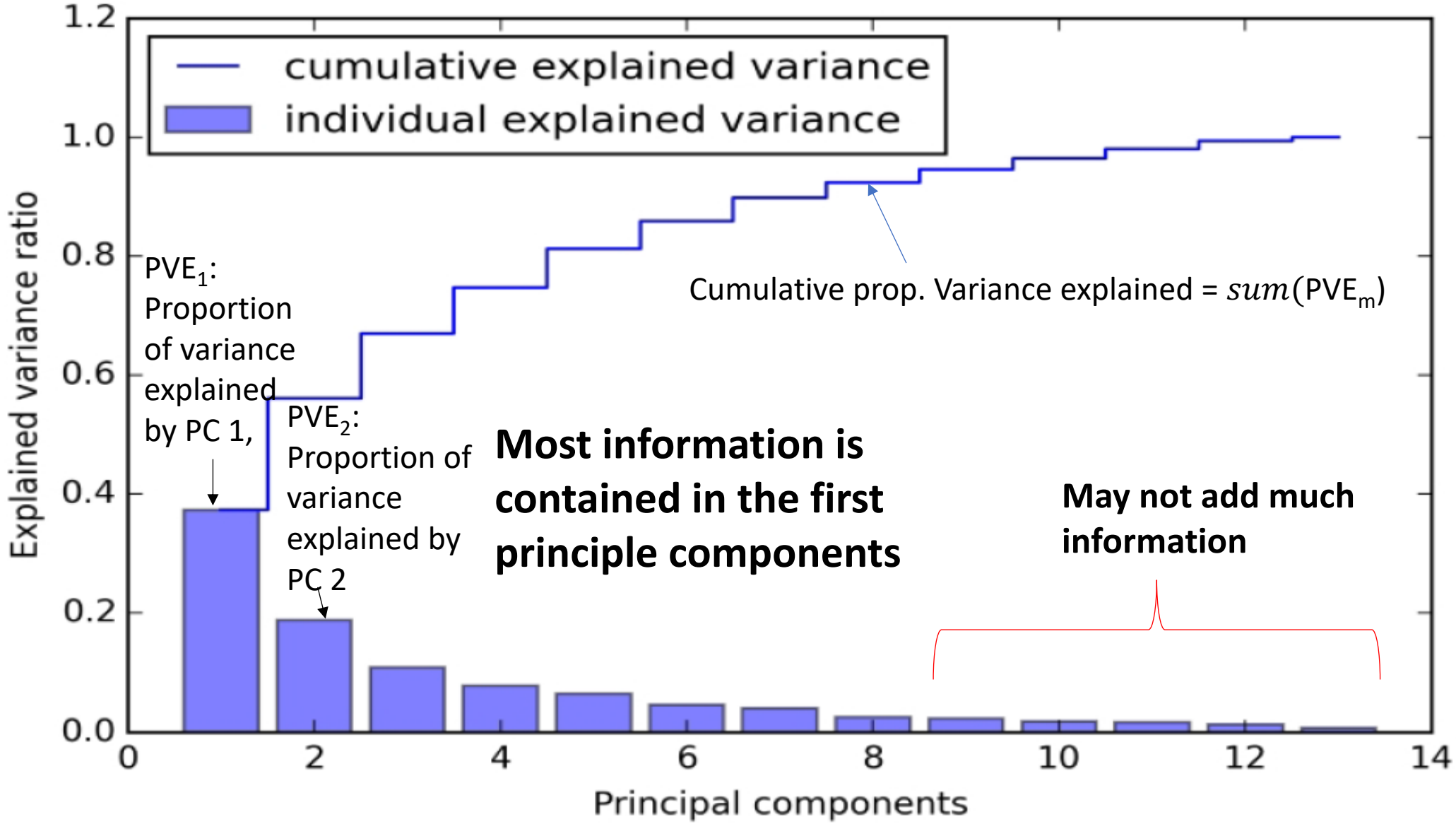
Proportion of Variance Explained

- However, we want to reduce the dimension so we only take few principle components
- Then, the **proportion of variance explained (PVE)** of the mth principle component is **the ratio between variance of Z_m to the total variance of X**
  - Positive quantity between 0 and 1

$$\text{PVE}_m = \frac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2}$$

- Sum over all PVE's is equal to 1 (total variance)

Typical Pattern for PVE



Choice of the Number of Principle Components

- One way to choose the number of principle components is to find number of components after which only slight information is gained
  - from previous graph: 6 components explained more than 60% of the variance, which could be sufficient for some applications.
- **PCA** can be used to reduce the dimension for **supervised learning** methods,
  - in this case **cross-validation** can be used to choose the number of principle components
- The number of PC to use depends on the application and dataset

Python

from sklearn.decomposition import PCA

- Define PCA components using scaled training data:
  N_components=2 # Define the number of principle components
  Data_pca = **PCA**(n_components=N_components).**fit(X_train_scaled)**

- Transform data into the defined principal components
  X_train_pca = **pca.transform**(X_train_scaled)
  X_test_pca = **pca.transform**(X_test_scaled)

- Get variance explained by each of the PCA components
  print('Explained variance,' , Data_pca.**explained_variance_ratio_**)

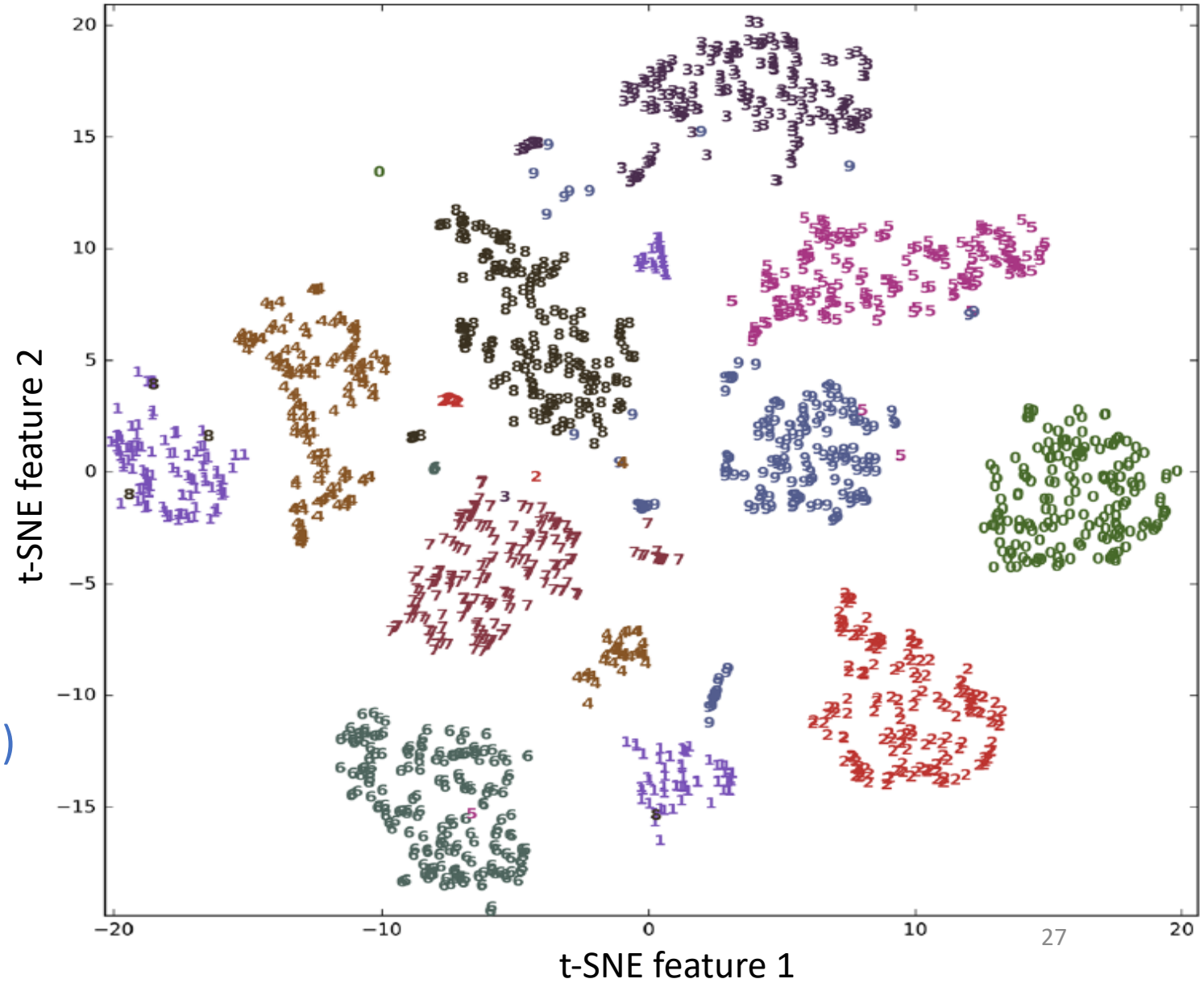
- More: http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

Manifold Learning with t-SNE

- t-Distributed Stochastic Neighbor Embedding (**t-SNE**): **non-linear** dimensionality reduction technique,
- Used for data **visualization**
  - Typically used to generate 2 new features (visualized in 2D plots)
- Rarely used for supervised learning
- **The transformation depends on how points are in the original feature space**
  - Tries to make points that are close in the original feature space closer, and points that are far apart in the original feature space farther apart in the new space.
    - (Uses joint probabilities and Kullback-Leibler divergence)

Example: t-SNE Applied to Handwritten Digit Dataset

- Each observation is colored by its class (0-9)
- # Original feature is 64 (8x8), gray scale values of pixels
- All classes are clearly separated using 2 derived features of t-SNE

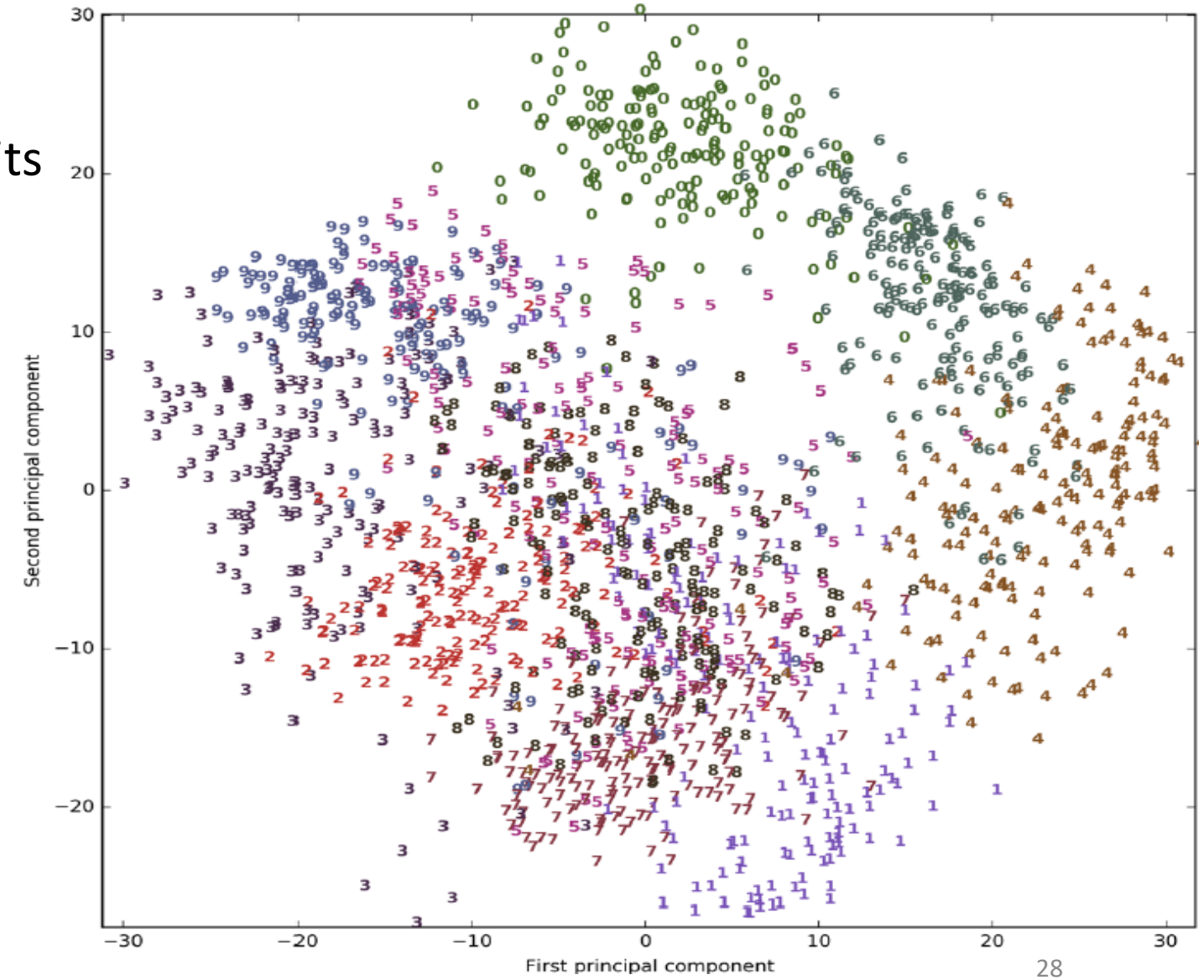



```
From sklearn.manifold import TSNE
Tsne_data=TSNE().fit_transform(digits.data)
```

http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

Example: PCA Applied to Handwritten Digit Dataset

Using two principle components on digits data, classes are not well-separated

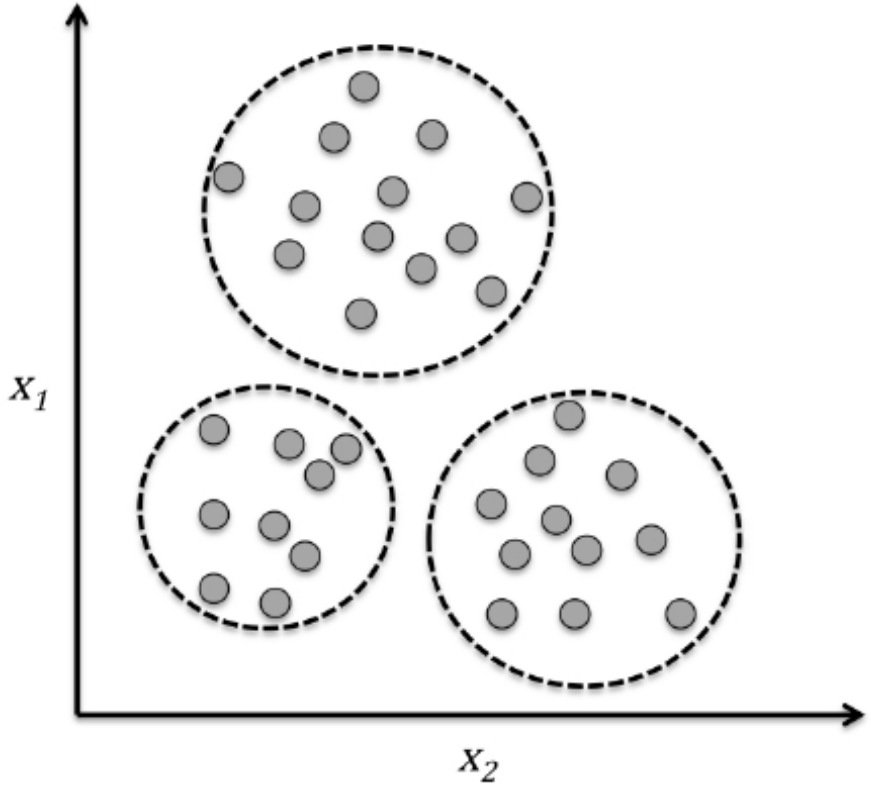


Clustering

- Find **distinct groups** in the data
  - Observation **within a group** are quite **similar**
  - Observations in **different groups** are quite different from each other

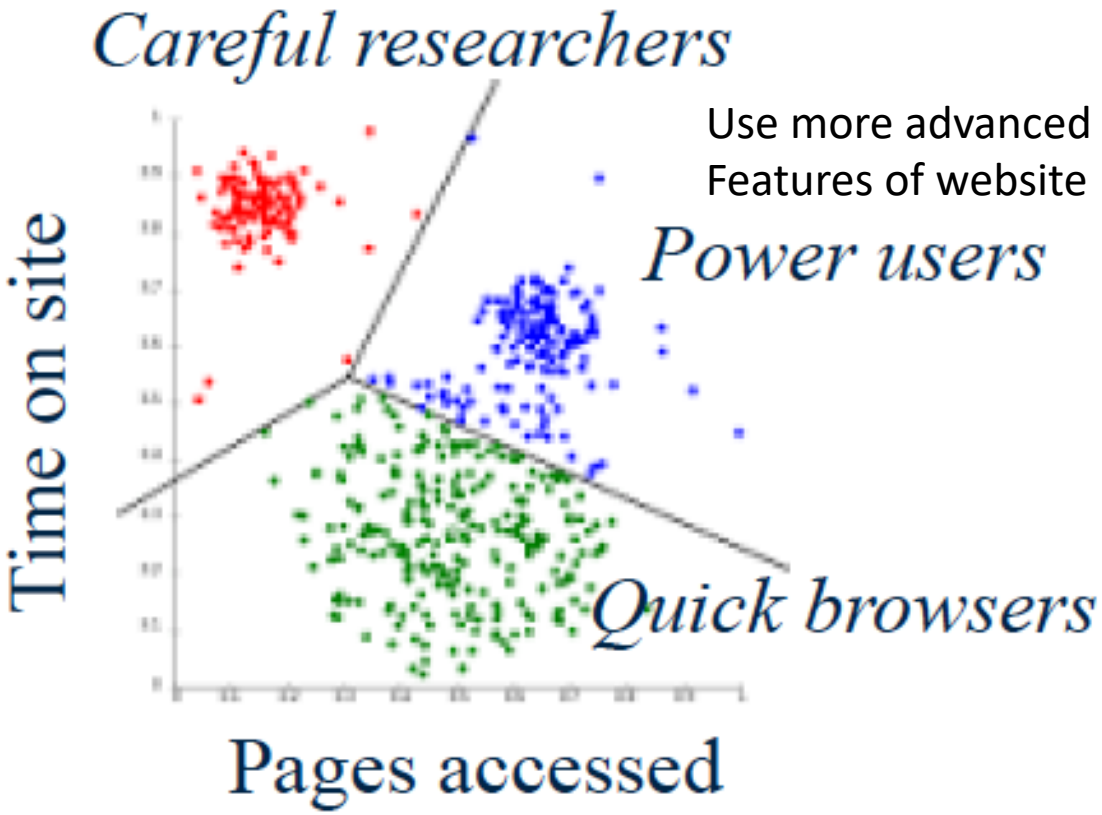
- Similar or Different are based on the domain/application and data being studied

- For n observations, what is the minimum and maximum number of clusters?



Examples:

- Market Segmentation: Identify subgroups of people who are more receptive to particular advertisement to likely to purchase a particular product
  - Features could be: household income, occupation, …
- Clustering could also be performed based on browsing activity
  - **Tailor website for each group**

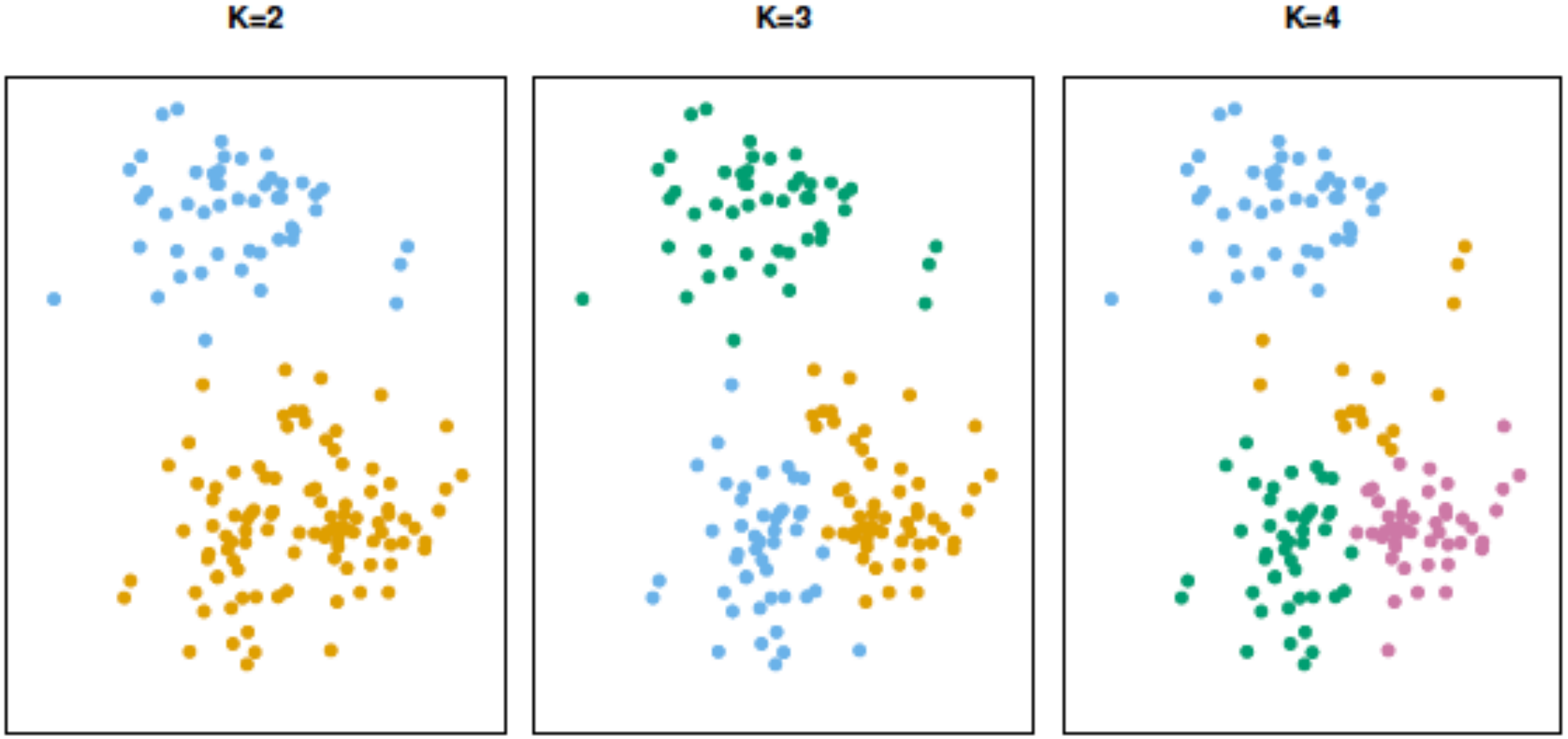


Well-known Algorithms

- Algorithms:
  - K means
  - Agglomerative Clustering
  - DBSCAN: Density Based Spatial Clustering of Applications with Noise

K-Means

- Partition the observations into **<u>pre-specified</u>** number of clusters (equal to K)
- Data is partitioned into K clusters, each observation is assigned to one of these clusters
- Figure: k=2,3,4 each cluster has different color



- Idea: find clusters where the **within-cluster variations** is as small as possible
  - Within-cluster variation (WCV) is the amount by which the observations within a cluster differ from each other
- Suppose we have K clusters represented by set: $C_1, C_2, \ldots, C_k$
  - Each set contains the indices of observations within the cluster
- Within-cluster variation of C_k is

Squared Euclidean distance between two samples

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \overbrace{\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2}$$

$|C_k|$Number of observations in cluster C_k

i *and* i' *are two indicies of two observations in cluster* C_k , each has p features

- Within-cluster variation can also be measured by how far observations are form their **cluster centroids**

Squared Euclidean distance between sample and cluster centroid

$$\sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2 = \sum_{i \in C_k} \overbrace{\|x_i - \mu_k\|^2}$$

- μ_k: is centroid of a cluster k; it's a vector with jth element $\bar{x}_{kj} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{ij}$

K-Means

- Optimization problem: finding clustering that **minimize the within-cluster variation**

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2 \right\}$$

- Solution is difficult

- A simplified algorithm can find a local optimum solution

K-means Clustering Algorithm

Step 1: Random Initialization:

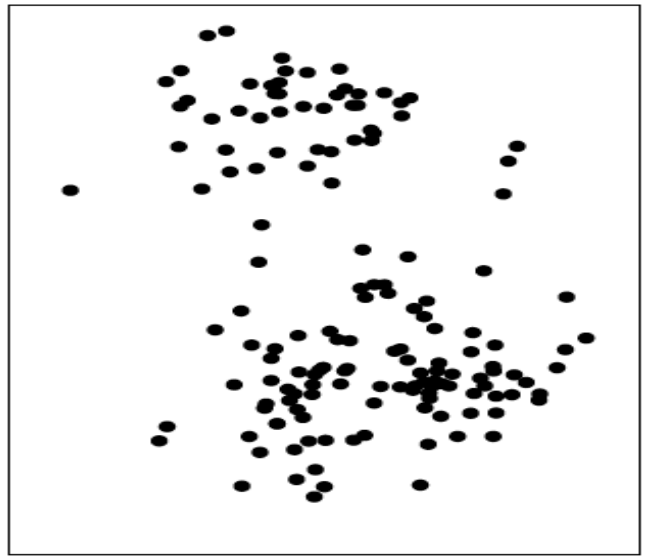
- Randomly assign each observation to a cluster (1,2,… K)
- (Other implementation: randomly select centroids from training data)

Step 2: Iterate until cluster assignment stops changing:

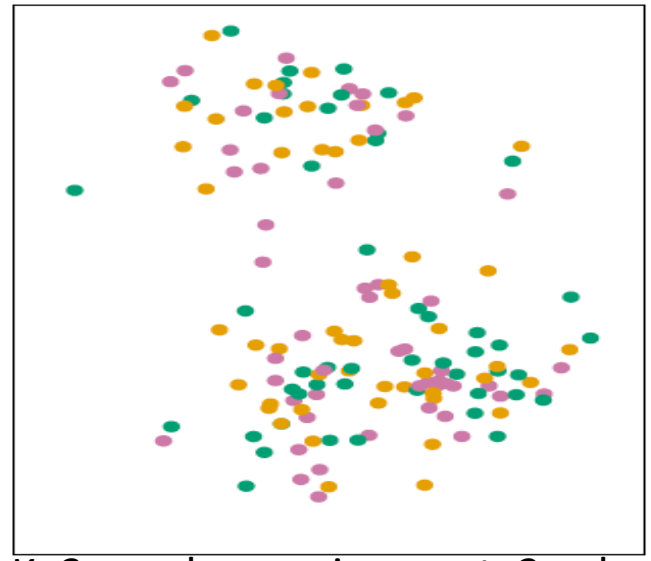
a) Compute the cluster center (centroid) == mean of the observations assigned to that cluster.

b) Assign each observation to the cluster with closest centroid

Data

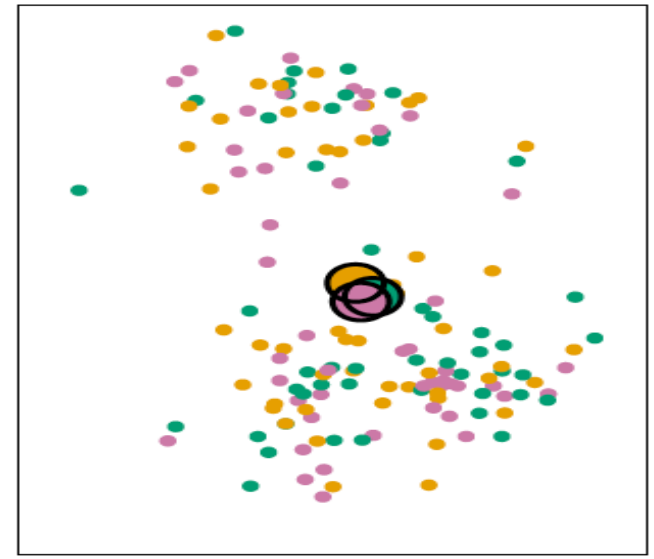


Step 1



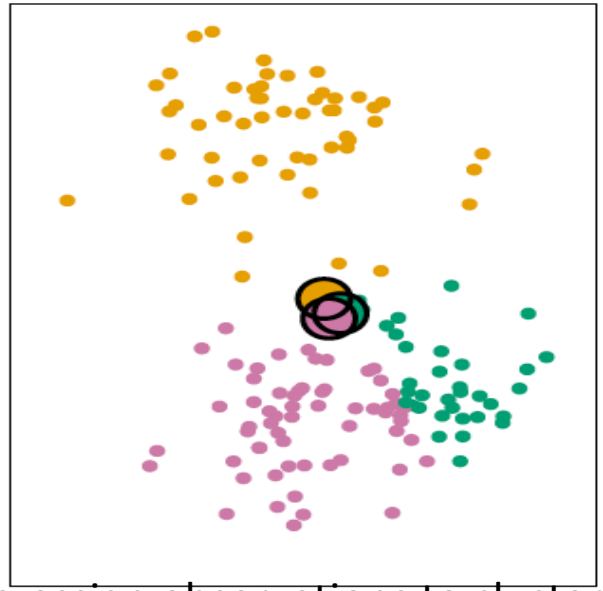
K=3, random assignment, 3 colors

Iteration 1, Step 2a



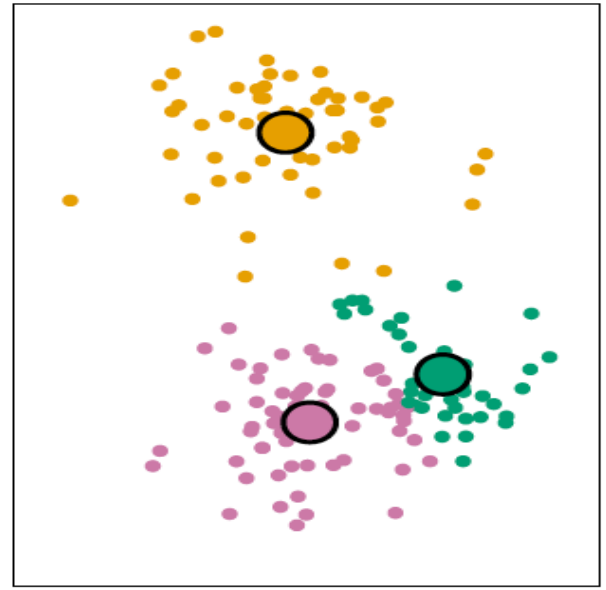
Calculate centroid of each cluster

Iteration 1, Step 2b



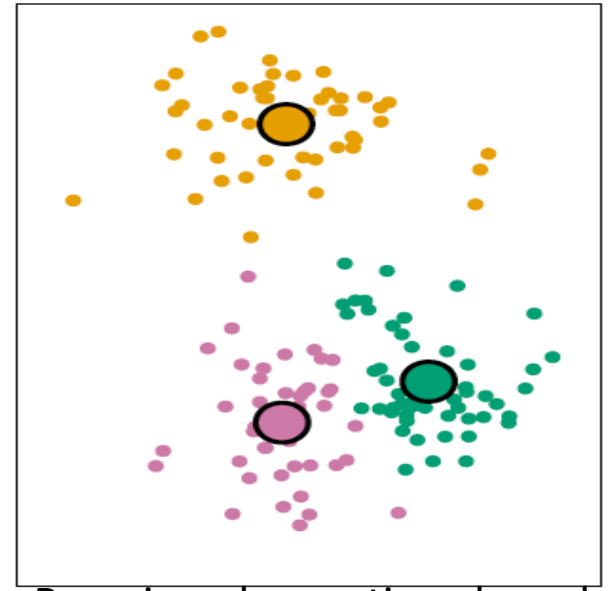
Re-assign observations to clusters based on closest centroid

Iteration 2, Step 2a



Recalculate centroids

Final Results



Reassign observations based on new centroids

K-Means, Initialization

- The algorithm may converge to a **local minimum**
- The result **depends on the initial random assignment**
  - Run algorithm multiple times with different initial configuration, then select solution with **smallest within-cluster variations**

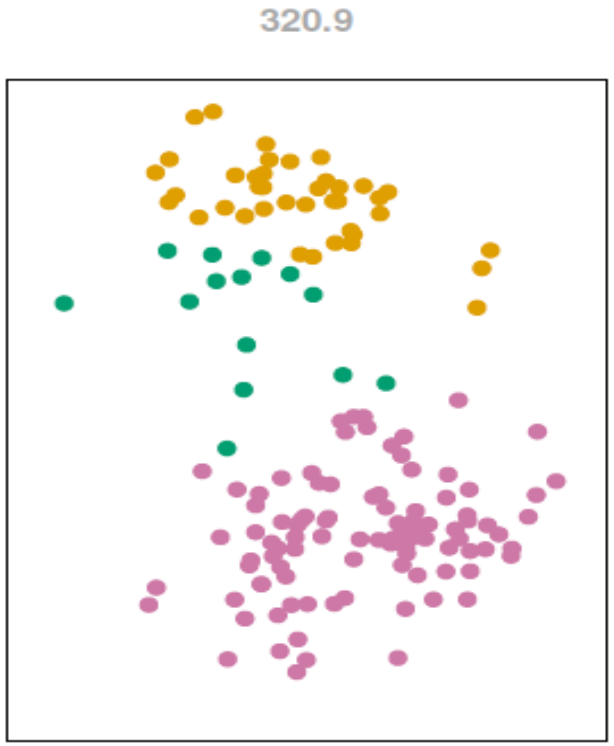
Example: Different Initial Setting Results in Different Clustering

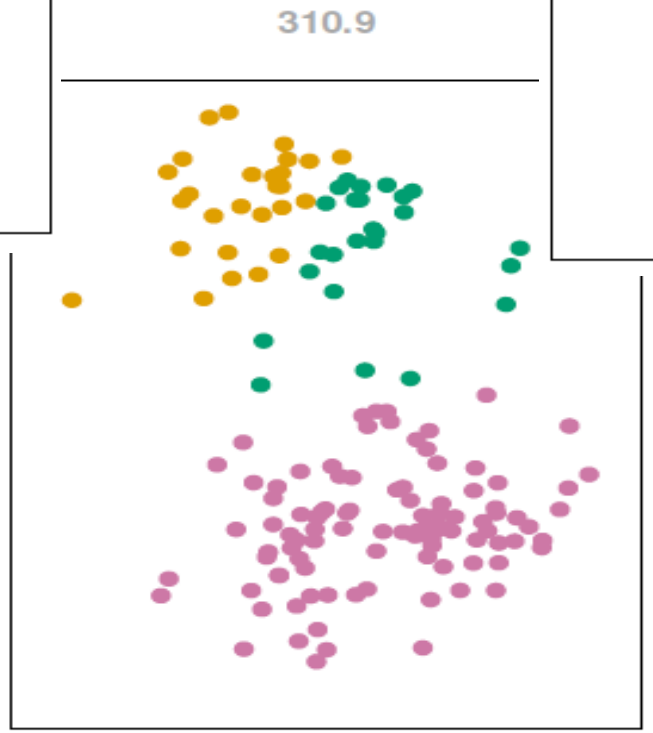
K=3, each cluster is represented by different color

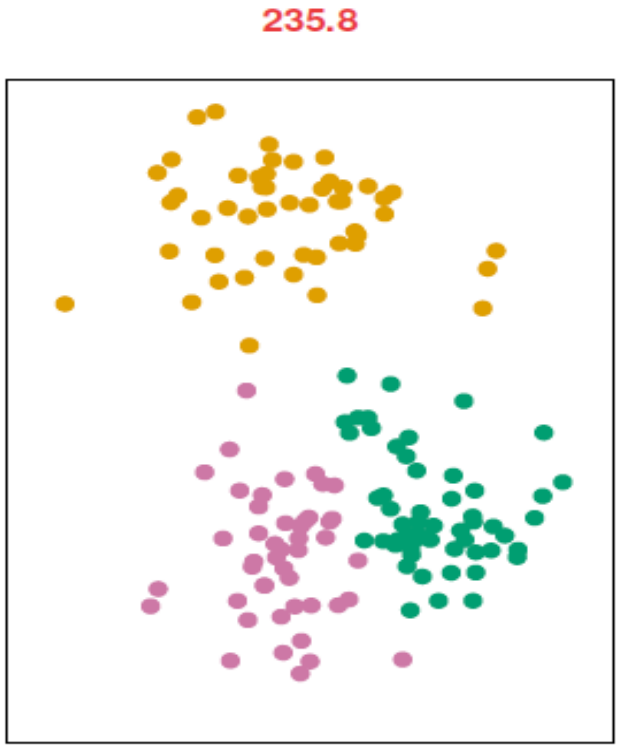
Each figure represents an output of K-Means clustering with **different random assignment** of observations at the beginning

Different local optima are obtained

Best solution results in objective (lowest within-cluster variation) of 235.8







Variants of K-Means

- **K-means ++:** Chooses the **initial centroids as far as possible** from each other

- **Soft K-means:** Helps when there are outliers
  - Typical K-means is a hard clustering, where each sample is assigned to one cluster
  - **Soft clustering uses probabilities of membership of each sample to one of the clusters**
  - Example: Hard cluserting, with K=3 clusters, a sample (i) in 2nd cluster can be represented by a weight vector $w_i = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$,

    In soft clustering, weight vector contains probabilities that depend on how close/far the sample is from the centroids $w_i = \begin{bmatrix} 0.1 \\ 0.85 \\ 0.05 \end{bmatrix}$,

    (Centeroids can be updated as: $\mu_k = \frac{\sum_{i=1}^{n} w_{ik}^{m} x_i}{\sum_{i=1}^{n} w_{ik}^{m}}$, m is parameter to control the fuzziness)

K-means in Python

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

- **Import and define class**
  from sklearn.cluster import Kmeans
  kmeans = KMeans(n_clusters=NumberOfClusters, random_state=0).fit(X)
  - To use Kmeans++: set *init='k-means++'*
- Use .predict to predict cluster of an observation
- Attributes:
  - **cluster_centers_**
  - **labels_:** has cluster assignment of each point
  - **inertia_**: is the sum of squared distances of samples to their closest cluster center.

Agglomerative (Hierarchical) Clustering

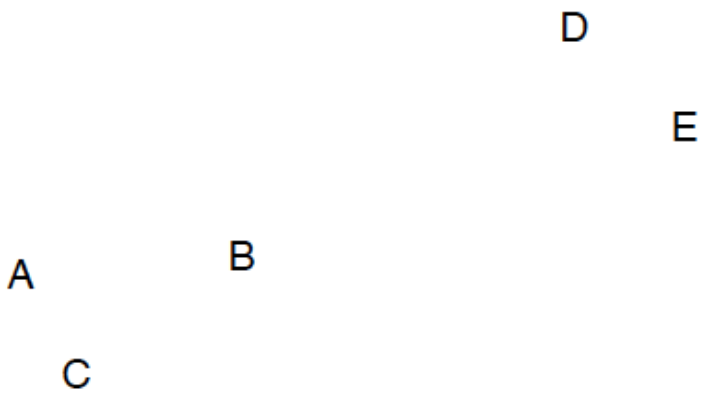
- Disadvantage of K-means: need to specify number of clusters
- We may not know how many clusters in advance
- Hierarchical Clustering: Produce a **dendrogram**, which is a tree-based representation of data that shows the **clustering obtained for each possible number of clusters**
- **Agglomerative clustering is common type of hierarchical clustering**
  - **bottom-up: tree is built starting from leaves**

Example: Hierarchical Clustering with n=5 observations

Calculate all n(n-1)/2 pair-wise dissimilarities,
and merge the least dissimilar (most similar)

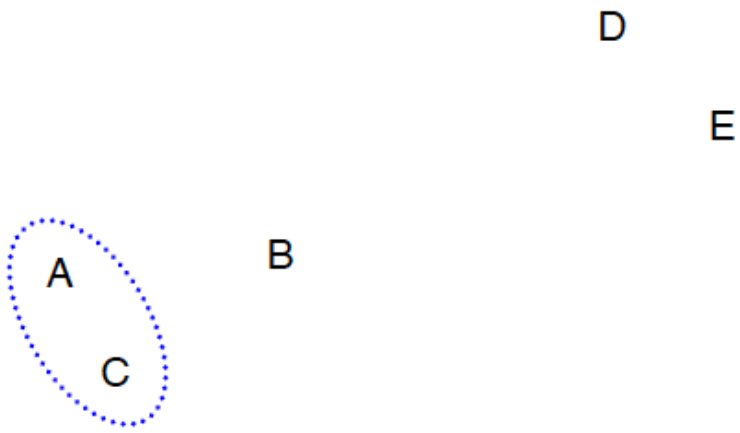
(1) n = 5 clusters

Start with each point in its own cluster



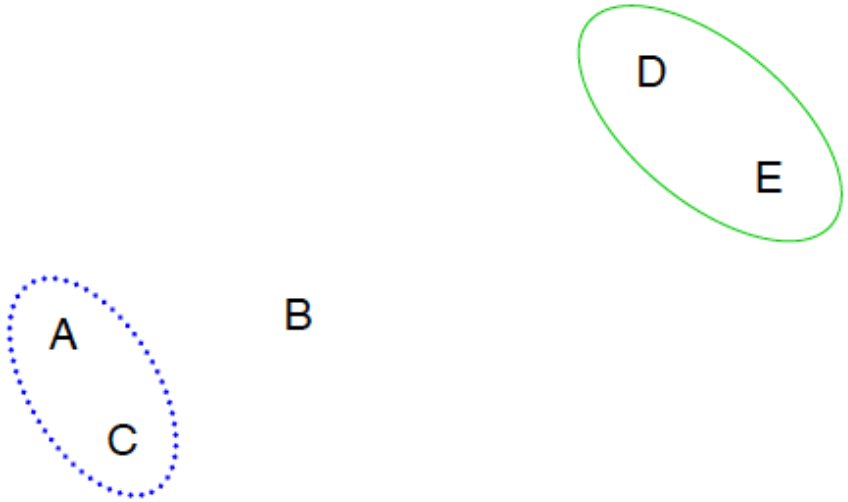
(2) n-1 = 4 clusters

Identify two closest clusters and merge them



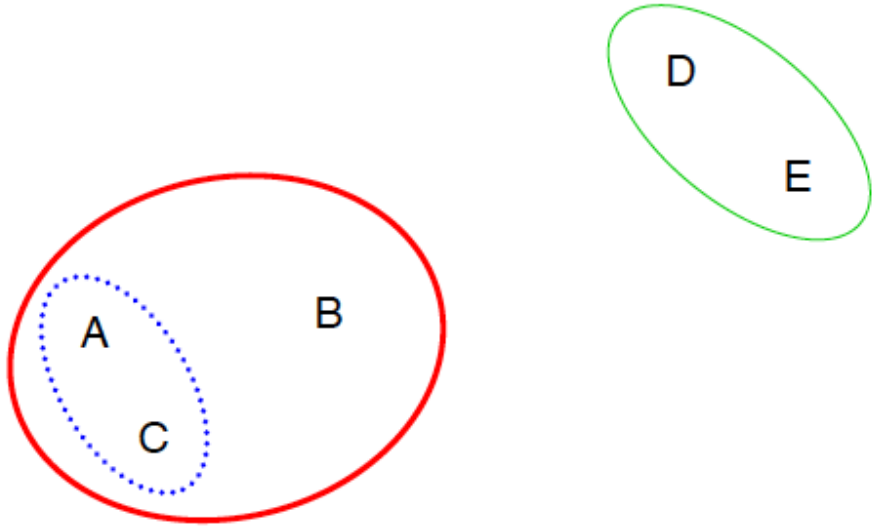
(3) n-2 = 3 clusters

Identify two closest clusters and merge them



(4) n-3 = 2 clusters

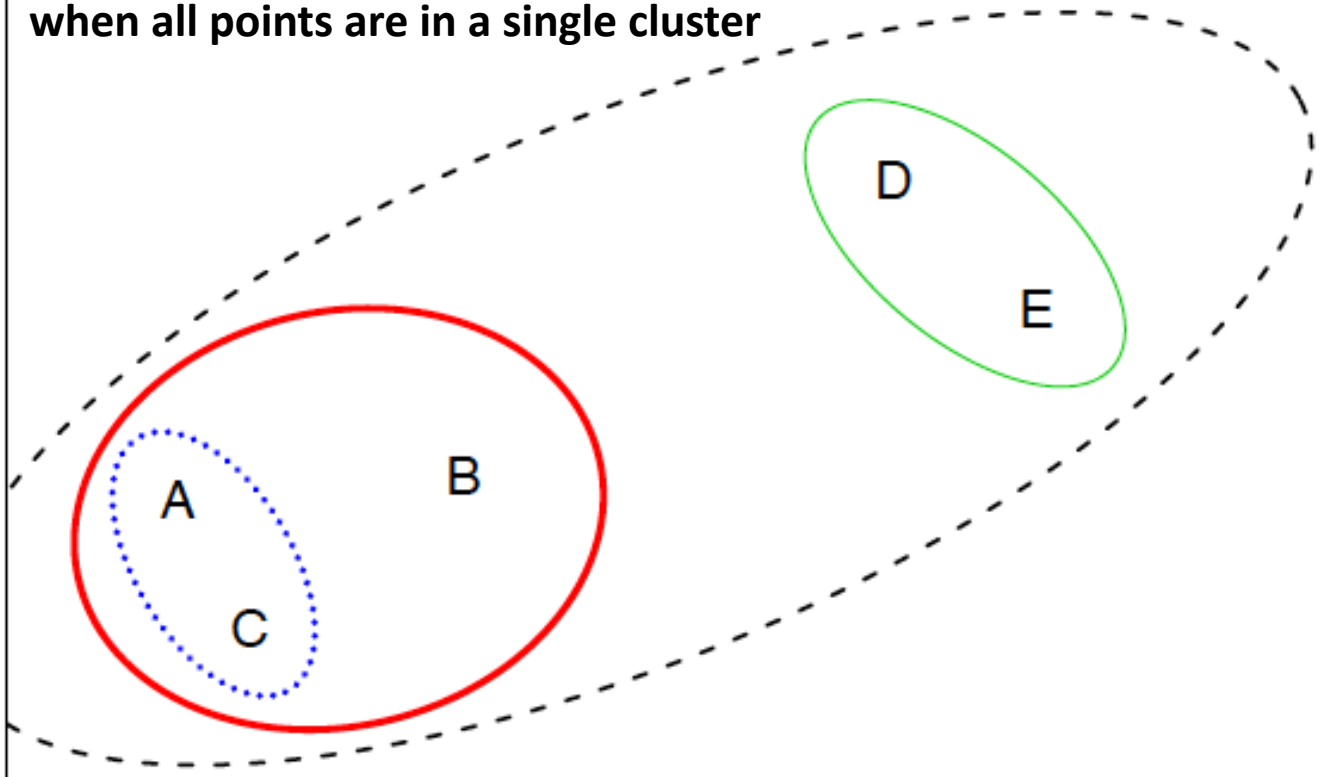
Identify two closest clusters and merge them



- Hierarchical clustering provides information about clustering obtained with different number of clusters
- Number of cluster ranges from: n to 1

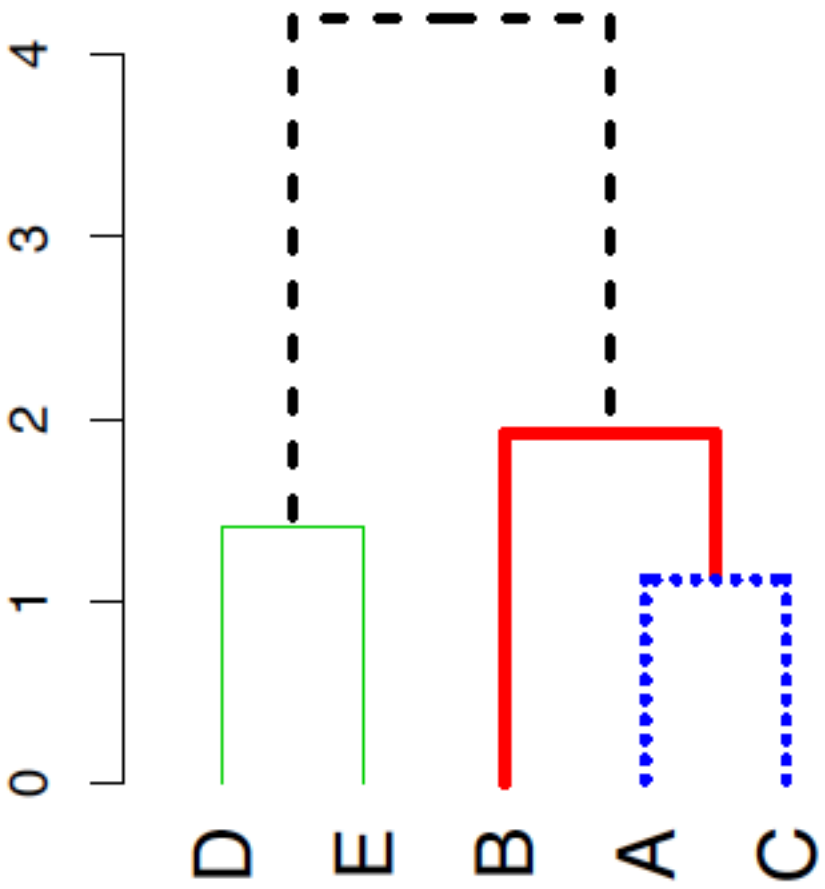
(5) n-4 = 1 cluster

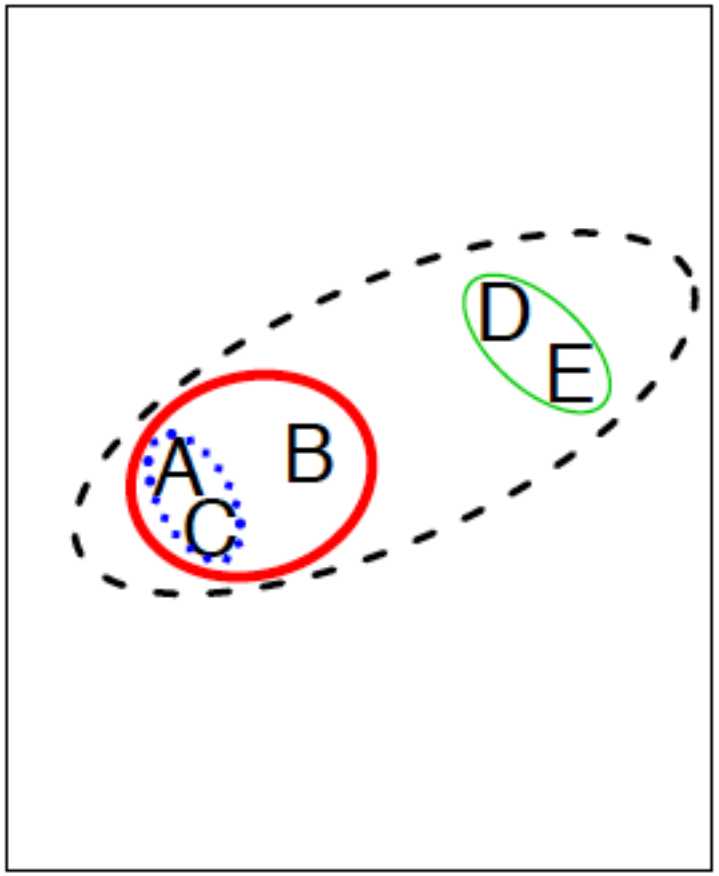
Identify two closest clusters and merge them, end when all points are in a single cluster



Hierarchical Clustering: Dendrogram

- Dendrogram: read from bottom to up
- Dissimilarity can be Euclidian distance, correlation,..
- **Height of a branch** represents **how far, i.e., <u>dissimilar</u>,** the merged clusters are
  - Observations that fuse at bottom of the tree are more similar to those merged at the top



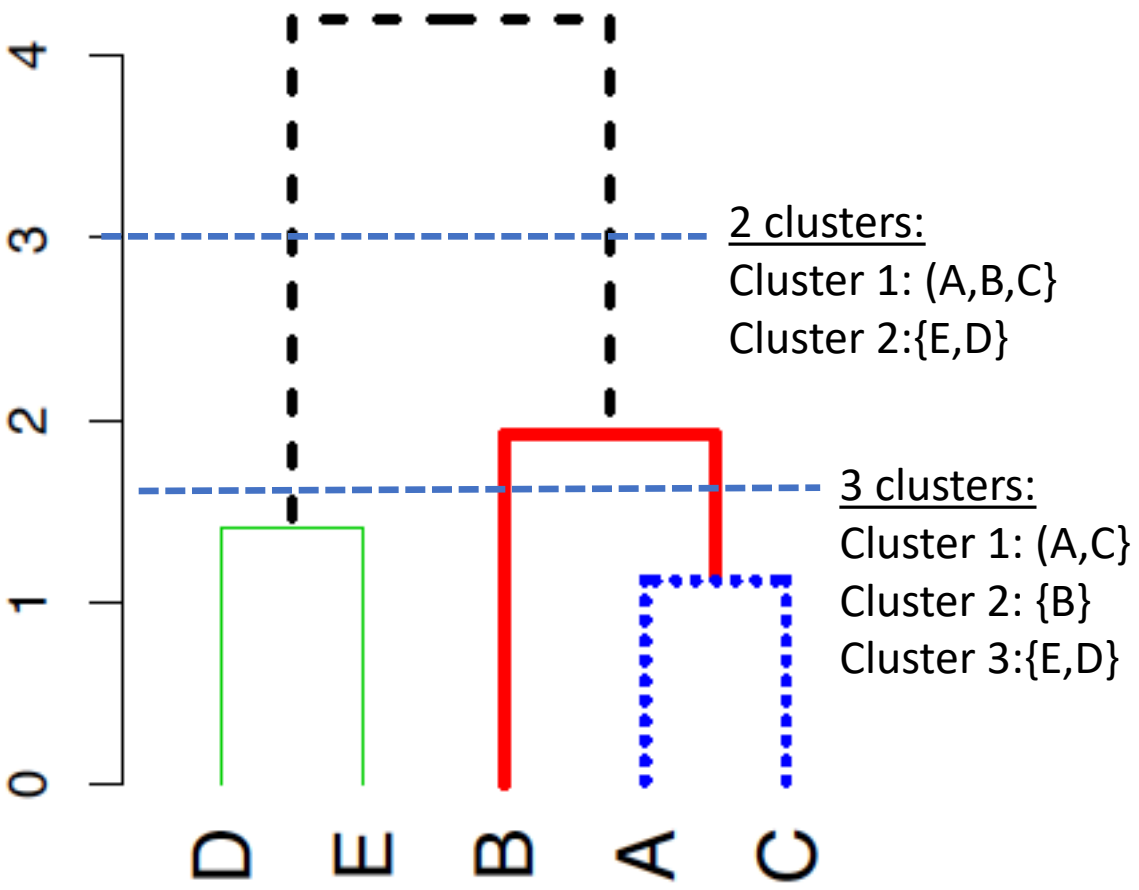


Dendrogram

Using the Dendrogram

- Cut dendrogram at a certain height to get clusters

- In previous example:
  Cut at height 3 ➔ get 2 clusters
  Cut at height 1.6 ➔ get 3 clusters



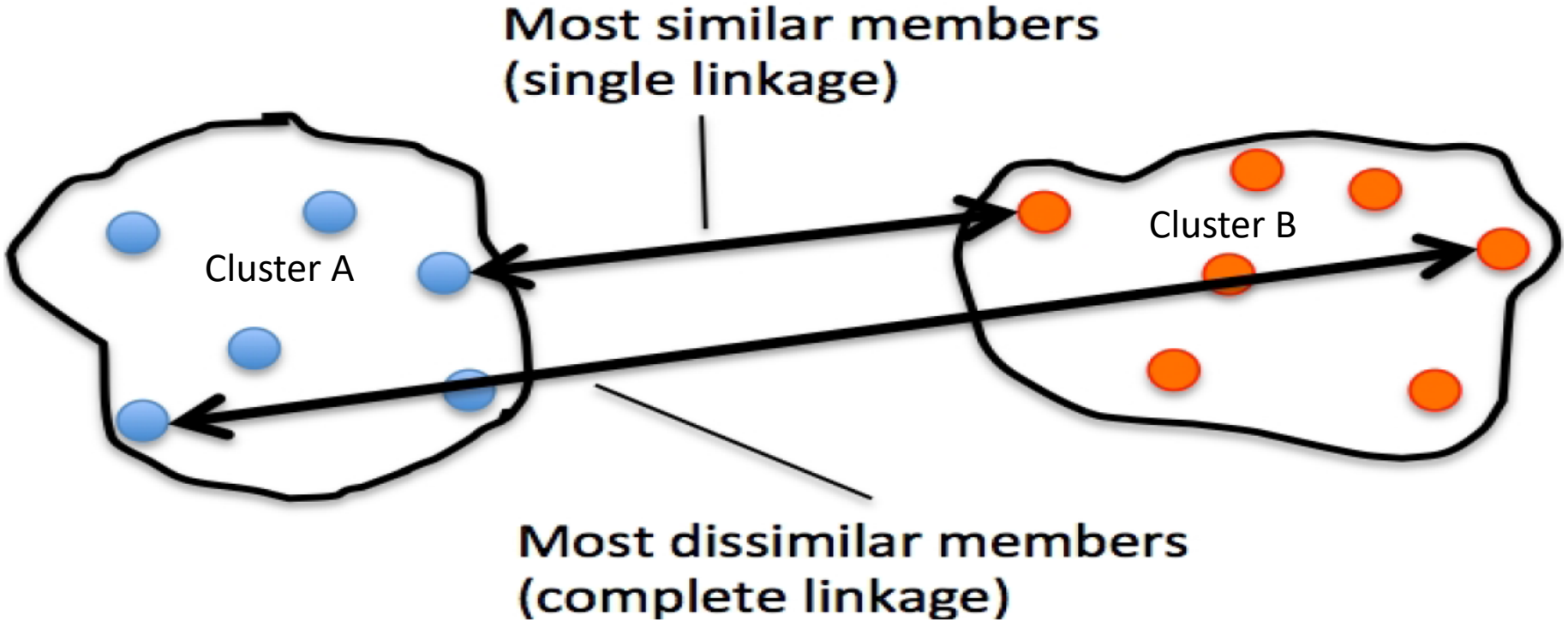
How to Measure Dissimilarity Between Clusters?

- We need to compute the dissimilarity
  - To know which clusters to be merged and at what height
  - **Merge clusters** that are **least dissimilar (most similar)**

- **Linkage**: defines the **dissimilarity between two groups/clusters**
- **Types of linkage are:**
  - **Complete**
  - **Single**
  - **Average**
  - **Centroid**
  - **Ward**
    - Ward merges clusters that lead to the minimum increase of the total within-cluster variation.

Types of Linkage

<u>Single linkage</u> (minimal inter-cluster dissimilarity): compute all pairwise dissimilarity between samples in cluster A and B, and record **smallest** of them (This will be the dissimilarity between cluster A and Cluster B)



<u>Complete linkage</u> (maximal inter-cluster dissimilarity): compute all pairwise dissimilarity between samples in cluster A and B, and **largest** of them will be the dissimilarity between the two groups

Types of Linkage … Cont.

- **Average linkage:** compute all pairwise dissimilarity between samples in cluster A and B, and record **average** of these dissimilarities

- **Centroid linkage:** compute the dissimilarity between centroid for cluster A and cluster B.

Dendrogram of Hierarchical Clustering depends on the type of linkage used

Example: Single linkage and Complete linkage

DISSIMILARITY between observations	Observation 1	Observation 2	Observation 3	Observation 4	Observation 5
Observation 1	0				
Observation 2	0.1	0			
Observation 3	0.5	0.25	0		
Observation 4	0.8	0.2	0.15	0	
Observation 5	0.3	0.75	0.7	0.4	0

Single linkage:
Step 1: G12=Group (1,2), dissimilarity 0.1
Step 2: G34= Group (3,4), dissimilarity 0.15
Step 3: G1234=Group (G12, G34), dissimilarity 0.2
Step 5: Group (G1234 , 5) dissimilarity 0.3

Complete linkage:
Step 1: G12=Group (1,2)
Step 2: G34=Group (3,4)
Step 3: Group (G34, 5), dissimilarity 0.7
Step 4: Group all, max height of dendrogram 0.8

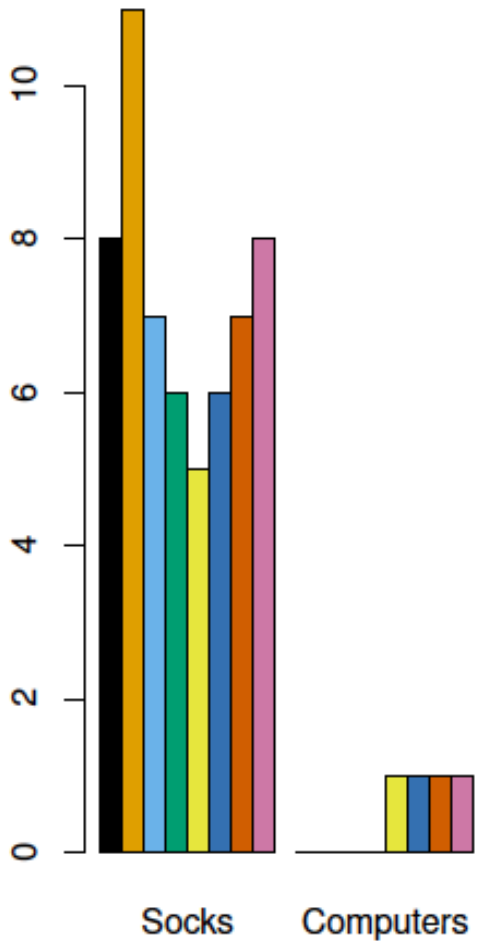
Scaling

- When features have different units, scaling helps
- When features are of same units
  - Scaling ensures that **features are given equal importance**
  - However, this depends on the application – in some cases not scaling is better
    - Choose most interpretable solution

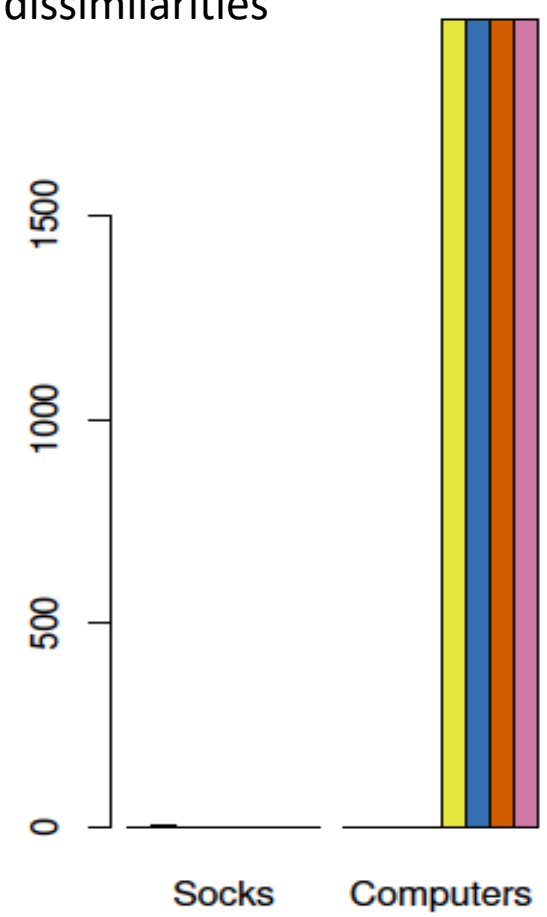
Example on Scaling

- Consider a retailer that sells two items: socks and computers, 8 online shoppers
- Assume Euclidean Distance is used to compute dissimilarity

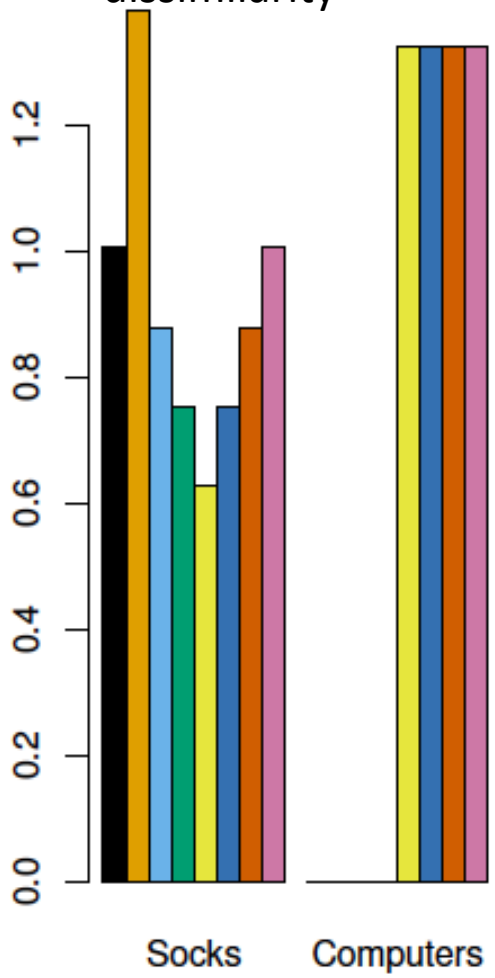
If we use **quantity**, socks will have higher weight in dissimilarity calculations



If we use **dollar** sales, computers have higher weight in calculating dissimilarities



After **scaling**, socks and computers have equal weight in calculating dissimilarity



DBSCAN

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

- Reference: Introduction to Machine Learning with Python, Chapter 3

- DBSCAN can **capture clusters of complex shapes and without identifying number of clusters a priori**
  - Check notebook file for illustration, where we apply clustering algorithms to a toy data set (two moons)

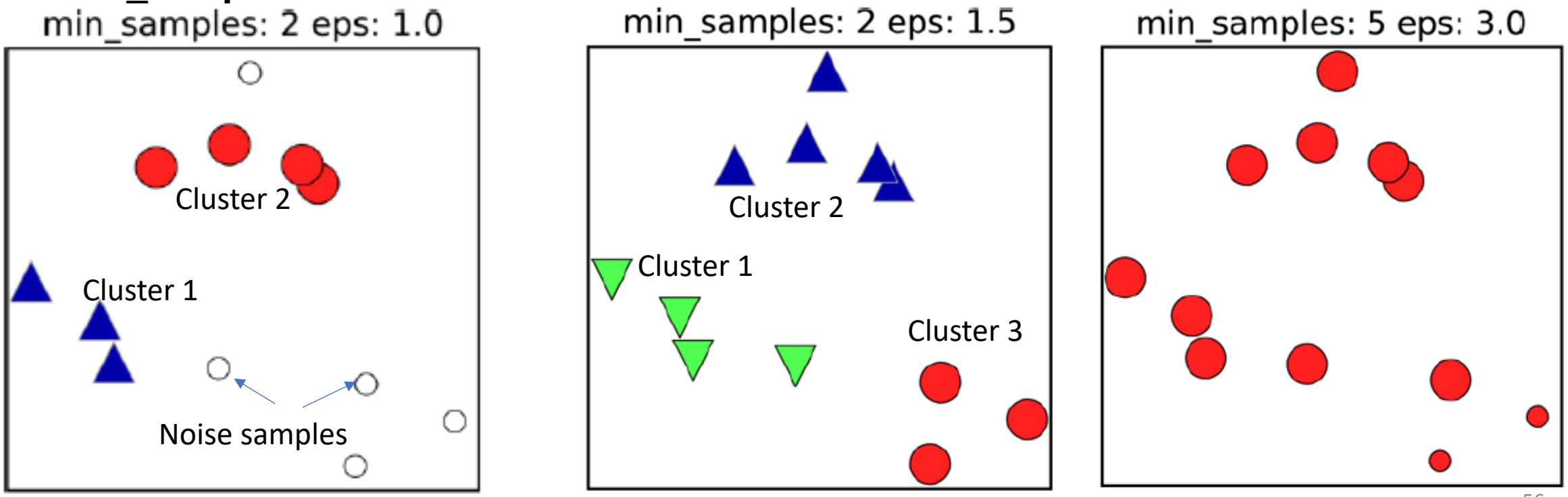
- **Idea: clusters form dense regions of data followed by regions that are relatively empty**

- Two parameters that DBSCAN depends on: **min_samples**, **eps**
  - Min_samples is minimum number of samples around a core cluster point that are within a distance eps

- **Identify points** that are in “crowded”/”**dense**” regions in the feature space
  - These observations are called **core points**
  - Starts with an arbitrary point, check if it is a **core point**
    - If there are “**min_samples**” number of observations within a distance “**eps**” form that point, then this is identified as a core point
      - Core points within a distance eps are in the same cluster
  - A cluster grows until there is not more core samples within distance “eps”, then new points are visited to create other clusters.

- **Noise**: observation points that do not belong to any cluster
  - Have less than **min_samples** within **eps**
- **Result of clustering depends on the parameters min_samples and eps**
  - **When eps increase ➔ less number of clusters**
  - **min_samples** deteremines the smallest class size

Large eps➔ all samples in one cluster



Practical Issues: Parameter Setting

- What are the best parameters to use in a clustering algorithm?
  - How many clusters to choose if K-means is used?
  - What type of dissimilarity & linkage to use in hierarchical clustering?
  - What is eps and min_samples for DBSCAN?
- Scale features or not?
- How to evaluate?

Evaluation of Clustering

- When labels are known (typically not the case), results of clustering can be compared:
  - Example: adjusted_rand_score, Normalized Mutual Information: value of maximum 1 means perfect clustering

- There is **no single agreed-upon performance metric** for unsupervised learning

- Some metrics measure compactness of clustering (ex. silhouette score)

- The **only way to know whether clustering worked well is to analyze it manually**