# IFI 9000 Analytics Methods
# Unsupervised Learning

by **Houping Xiao**

Spring 2021

# Introduction

# Unsupervised Learning

- In the last lectures, we had problems where a set features $x_1, \cdots, x_p$ were given along with a response $y$, and the goal was constructing a model relating the features to the response
- In **unsupervised learning** we do not observe any responses (or there might not be any responses defined) and the observation only consists of the feature values $x_1, \cdots, x_p$
- Unsupervised learning can give us a better understanding of the data, and allow us the **visualize** them in an informative way
- They can also be applied as a **preprocessing tool** before a supervised learning scheme

# Unsupervised Learning

- Unsupervised learning is more of a subjective analysis compared to the supervised learning which has a clear goal
- Probably **grouping or clustering the data** is the most popular task done in unsupervised learning, e.g.:
  - grouping movies based on the rating and viewers
  - grouping patients by their gene measurements
- Often times collecting unsupervised data is easier than supervised data, since labeling the data often requires **human interaction**

# What is Principal Component Analysis (PCA)?

- You may hear about SVD, singular value decomposition
- In this lecture, we try to present a thorough view of this topic
- PCA uses the full set of features and **linearly weights** them to produce new variables that have **maximal variance** and are **mutually uncorrelated**
- While the new variables can always be used as a new set of features in supervised learning application, the PCA can also help with visualizing the data

# Principal Component Analysis

- The first principal component is a (normalized) weighting of the features which has the largest sample variance:

$$z_1 = \boldsymbol{x}^\top \boldsymbol{\phi}_1 = \phi_{1,1} x_1 + \phi_{2,1} x_2 + \cdots + \phi_{p,1} x_p$$

  we have the constraint $||\phi_1||^2 = \sum_{j=1}^{p} \phi_{j,1}^2 = 1$
- The vector $\phi_1$ is called the **loading vector** associated with the first principal component
- Next, we explain what we mean by sample variance and how to calculate it

# Principal Component Analysis

- Suppose that all the variables (features) $x_1, \cdots, x_p$ are centered so that they are zero-mean
- When we have $n$ data observations, we will have a matrix $\boldsymbol{X}$ each row of which is an observation
- Recall that to calculate the first PC we needed to maximize the sample variance of

$$z_1 = \boldsymbol{x}^\top \boldsymbol{\phi}_1 = \phi_{1,1} x_1 + \phi_{2,1} x_2 + \cdots + \phi_{p,1} x_p$$

- Since we have $n$ samples, for each we can define

$$z_{i,1} = \boldsymbol{x}_i^\top \boldsymbol{\phi}_1 = \phi_{1,1} x_{i,1} + \phi_{2,1} x_{i,2} + \cdots + \phi_{p,1} x_{i,p}$$

  for all samples $i = 1, \cdots, n$
- Since all variables are already zero-mean, the sample variance of $z_1$ can now be written as

$$SV(z_1) = \frac{1}{n} \sum_{i=1}^{n} z_{i,1}^2 = \frac{1}{n} ||\boldsymbol{X} \boldsymbol{\phi}_1||^2$$

# Calculation

- By definition, the loading vector associated with the first principal component can be obtained by solving the optimization problem:

$$\phi_1 = \arg\max_{\phi} \frac{1}{n}||\boldsymbol{X}\phi_1||^2, \quad s.t. \quad ||\phi||^2 = 1$$

- This optimization is equivalent to the following optimization

$$\phi_1 = \arg\max_{\phi} \frac{\phi^\top \boldsymbol{X}^\top \boldsymbol{X}\phi}{\phi^\top \phi}$$

- As will be detailed later this problem can be solved by eigen-decomposition of $\boldsymbol{X}^\top \boldsymbol{X}$

# More on Principal Components

- The process explained above can continue to generate next principal components, basically a variable $z_2$ can loading vector $\phi_2$ such that

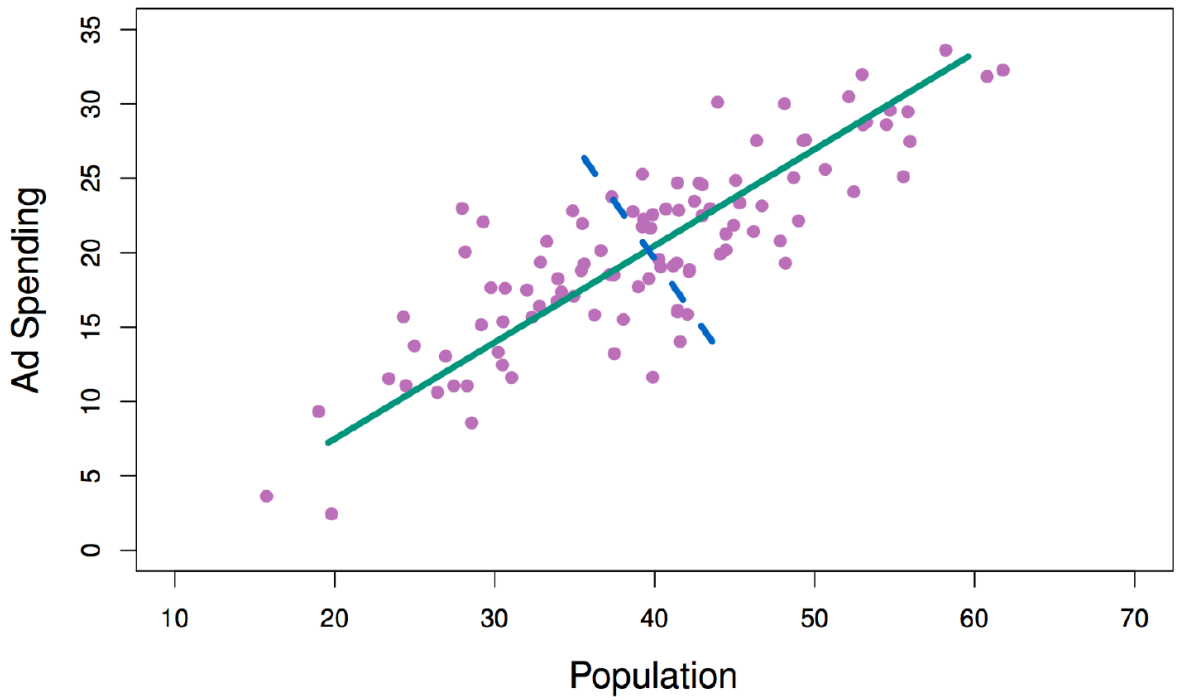
$$z_2 = \boldsymbol{x}^\top \boldsymbol{\phi}_2 = \phi_{1,2} x_1 + \phi_{2,2} x_2 + \cdots + \phi_{p,2} x_p$$

  has the largest possible sample variance while $||\phi_2||^2 = \sum_{j=1}^{p} \phi_{j,2}^2 = 1$ and $\phi_2 \top \phi_1 = 0$
- The condition $\phi_2 \top \phi_1 = 0$ warrants that $z_1$ and $z_2$ are uncorrelated since

$$\sum_{i=1}^{n} z_{i,1} z_{i,2} = \boldsymbol{\phi}_1^\top \left( \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top \right) \boldsymbol{\phi}_2 = \boldsymbol{\phi}_1^\top \boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{\phi}_2 = \lambda \boldsymbol{\phi}_2 \boldsymbol{\phi}_1 = 0$$

- We will later see the reason for the last two equalities
- This process may continue until we reach the maximum number of principal components with is $min(n-1, p)$

# Example




The population size (pop) and ad spending (ad) for 100 different cities are shown as purple circles. The green solid line indicates the first principal component direction ($\phi_1$), and the blue dashed line indicates the second principal component direction ($\phi_2$)

# Addressing the optimization problem

- For the first PCA, we had the optimization

$$\arg\max_{\phi} \phi^\top \boldsymbol{X}^\top \boldsymbol{X} \phi, \quad s.t. \quad \phi^\top \phi = 1$$

- The Langrangian takes the form

$$\mathcal{L}(\phi, \lambda) = -\phi^\top \boldsymbol{X}^\top \boldsymbol{X} \phi + \lambda(\phi^\top \phi - 1)$$

- Therefore the optimality happens when for some $\lambda$

$$\boldsymbol{X}^\top \boldsymbol{X} \phi = \lambda \phi$$

# Linking to the SVD of $\boldsymbol{X}$

- So far we talked about the principal components $z_j$, $j = 1, \cdots$, which are uncorrelated and have respectively the largest sample variations
- Associated with each $z_j$ we had a **loading vector** $\phi_j$ obtaining which required us to solve an optimization problem
- Projection of each data point $\boldsymbol{x}_i$ onto each $\phi_j$ is the point $(\boldsymbol{x}_i^\top \phi_j)\phi_j$ and $(\boldsymbol{x}_i^\top \phi_j)$ is called the principal component score
- As a result the vector $\boldsymbol{X}\phi_j$ is called the vector of principal component scores associated with $\phi_j$
- We will next link these quantities to the SVD of $\boldsymbol{X}$

# Linking to the SVD of $\boldsymbol{X}$

- The SVD of $\boldsymbol{X}$:

$$\boldsymbol{X} = \boldsymbol{U}_{n\times r}\boldsymbol{\Sigma}_{r\times r}\boldsymbol{V}^{\top}_{r\times p}$$

where

$$\boldsymbol{\Sigma}_{r\times r} = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}$$

- The singular vectors are orthonormal, i.e., $\boldsymbol{U}^{\top}\boldsymbol{U} = \boldsymbol{I}$ and $\boldsymbol{V}^{\top}\boldsymbol{V} = \boldsymbol{I}$
- If $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_r$ denote the columns of $\boldsymbol{U}$ and $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_r$ denote the columns of $\boldsymbol{V}$ then,

$$\boldsymbol{X} = \sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^{\top} + \sigma_2\boldsymbol{u}_2\boldsymbol{v}_2^{\top} + \cdots + \sigma_r\boldsymbol{u}_r\boldsymbol{v}_r^{\top}$$

# Linking to the SVD of $\boldsymbol{X}$

- Recall that to warrant the optimality we needed to have:

$$\boldsymbol{X}^\top \boldsymbol{X} \phi = \lambda \phi$$

or using the SVD

$$\boldsymbol{V} \boldsymbol{\Sigma}^2 \boldsymbol{V}^\top \phi = \lambda \phi$$

- It can be easily seen that setting $\boldsymbol{v}_j = \phi_j$ warrants all the optimality conditions
- So the loading vectors can easily be obtained as $\boldsymbol{v}_j = \phi_j$
- An the vector of principal component scores is obtained via

$$\boldsymbol{X} \phi_j = \left( \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^\top + \sigma_2 \boldsymbol{u}_2 \boldsymbol{v}_2^\top + \cdots + \sigma_r \boldsymbol{u}_r \boldsymbol{v}_r^\top \right) \boldsymbol{v}_j = \sigma_j \boldsymbol{u}_j$$

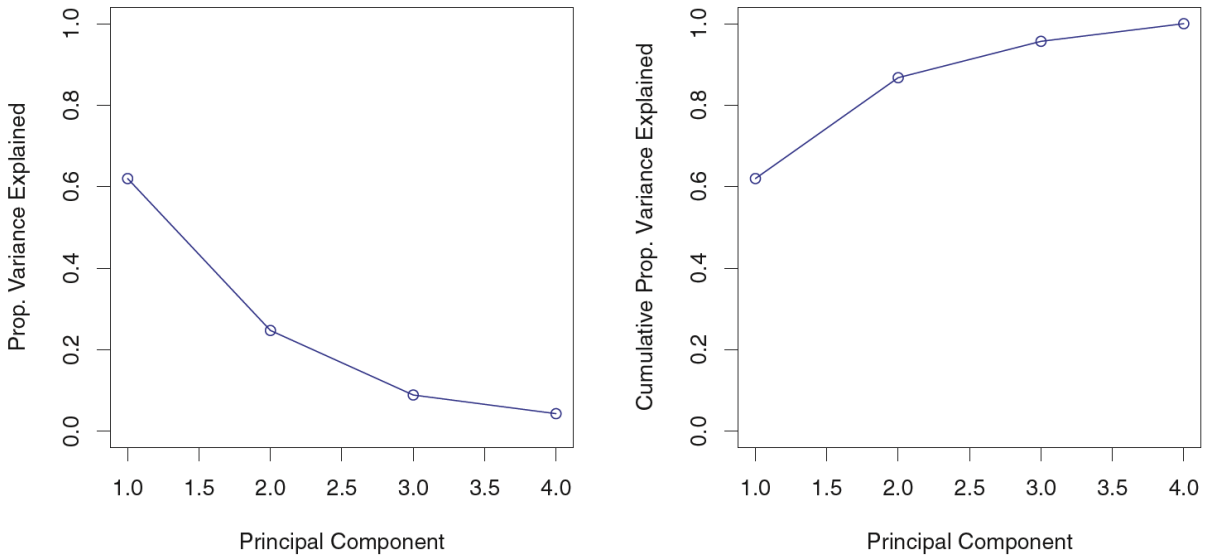
- So SVD would directly give us the loading vectors and the projection of the data onto them

# More on PCA

- The proportion of data explained by each principal component $z_j$ would be

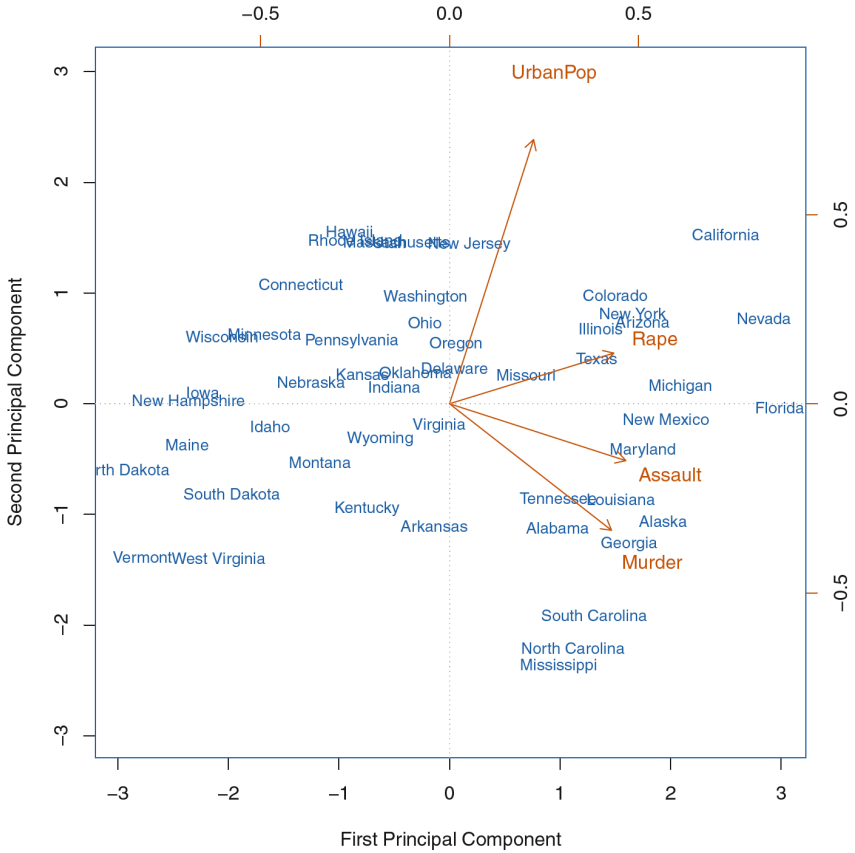
$$\frac{||\boldsymbol{X}\phi_j||^2}{||\boldsymbol{X}||_F^2} = \frac{\sigma_j^2}{\sigma_1^2 + \cdots + \sigma_r^2}$$

- How many principal components do we need?



# BiPlot Example

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

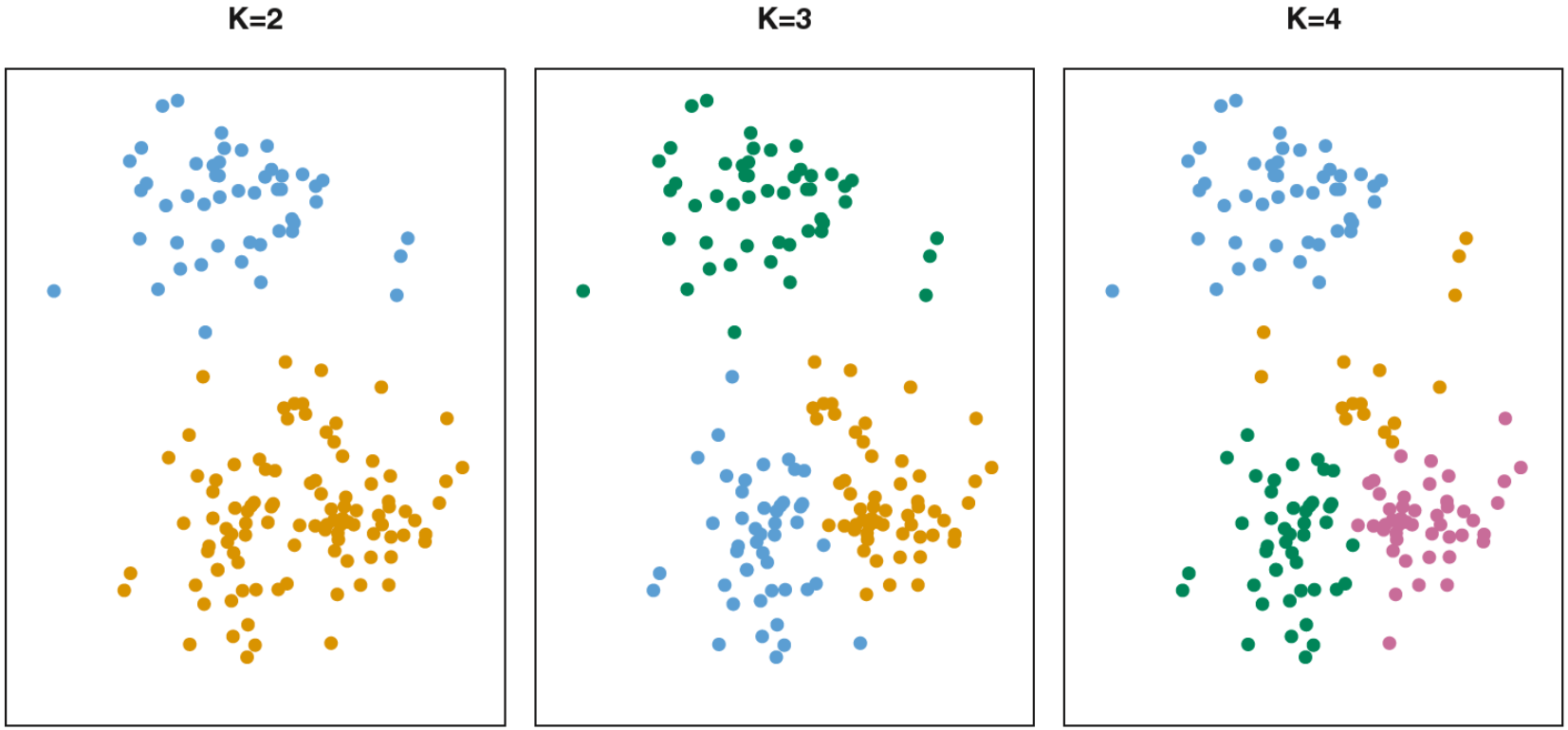


# Unsupervised Learning: Clustering

- PCA uses the full set of features and **linearly weights** them to produce new variables that have **maximal variance** and are **mutually uncorrelated**
- This lecture is devoted to clustering methods
- Intuitively, clustering deals with finding subgroups or partitions in the data which are similar
- An immediate observation, is we need a notion of **similarity** or **difference**
- In PCA, we found a low dimensional representation of the data, while in clustering we find similar subgroups

# K-Means Clustering

- In K-means clustering, we seek to partition the data into $K$ subgroups ($K$ **is pre-defined**) based on some measure of similarity



# K-Means Clustering

What do we mean by partitioning?

- Consider the samples $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$, we construct $K$ sets $C_k$ such that
  - $C_1 \cup C_2 \cup \cdots \cup C_K = \{1, \cdots, n\}$
  - $C_i \cap C_j = \emptyset, \forall i \neq j$
- After defining a distance measure, a good clustering would be the one for which the **within-cluster variance (WCV)** is minimized
- WCV would be able to generate actual numeric values for us

$$WCV(C_k) \to \mathbb{R}$$

- An optimal K-Means clustering pattern would hence be the minimizer of the following:

$$\min_{C_1, \cdots, C_K} \sum_{k=1}^{K} WCV(C_k)$$

# K-Means Clustering

- Usually the most standard distance measure used is the Euclidean distance, which yields

$$WCV(C_k) = \frac{1}{|C_k|} \sum_{i,j \in k} ||\boldsymbol{x}_i - \boldsymbol{x}_j||$$

  where $|C_k|$ denotes the number of elements in $C_k$
- Note that $\boldsymbol{x}_i \in \mathbb{R}^p$, and each observation has $p$ features associated with it
- For such distance the K-Means formal minimization is

$$\min_{C_1, \cdots, C_K} \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,j \in k} ||\boldsymbol{x}_i - \boldsymbol{x}_j||$$

- This problem is also NP-hard, but this would not prevent us from finding a local minimizer

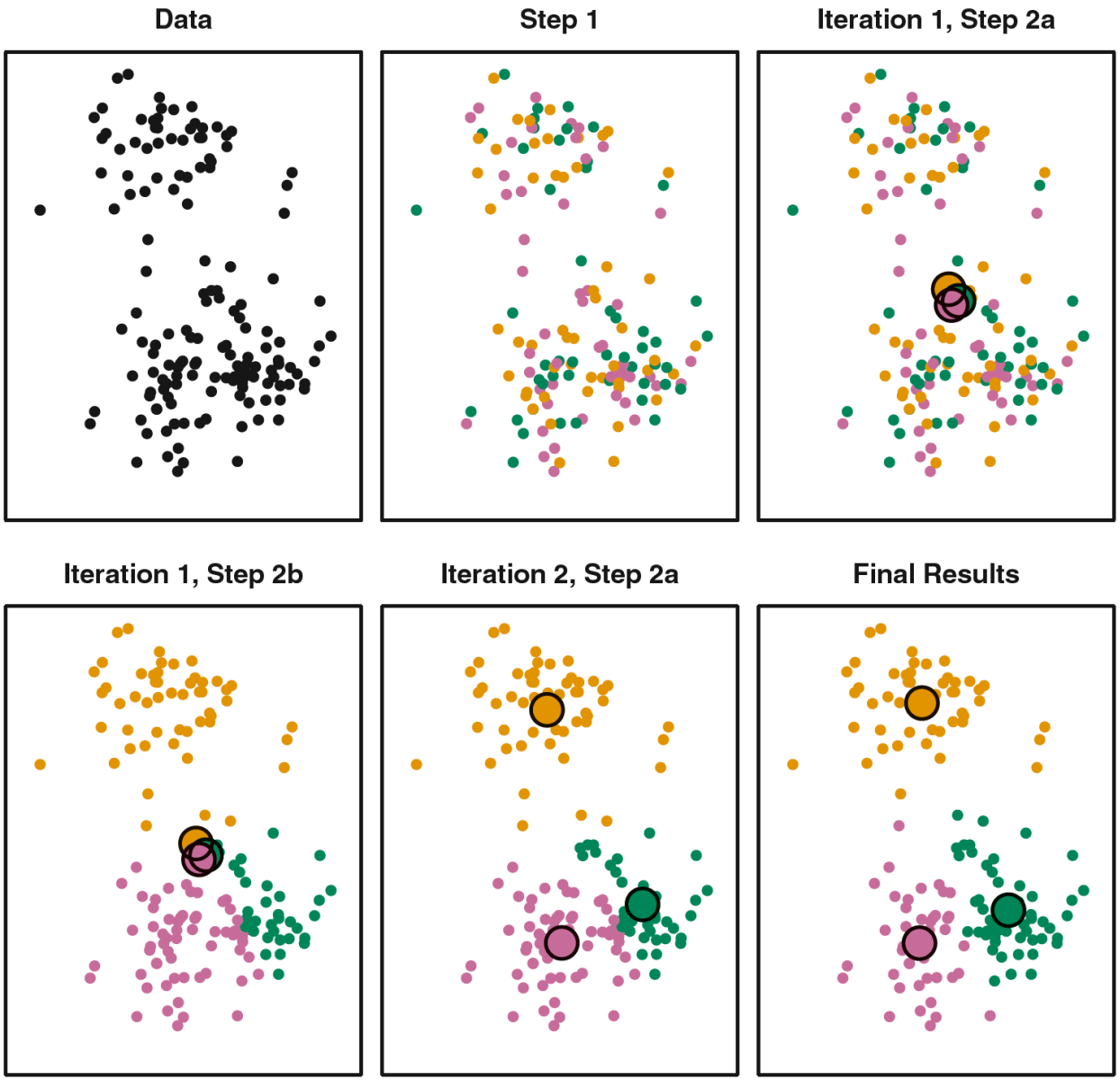
# K-Means Clustering: Local Minimization

- Here is a heuristic that would provide some local minimizers of the original problem

**Algorithm 10.1** *K-Means Clustering*

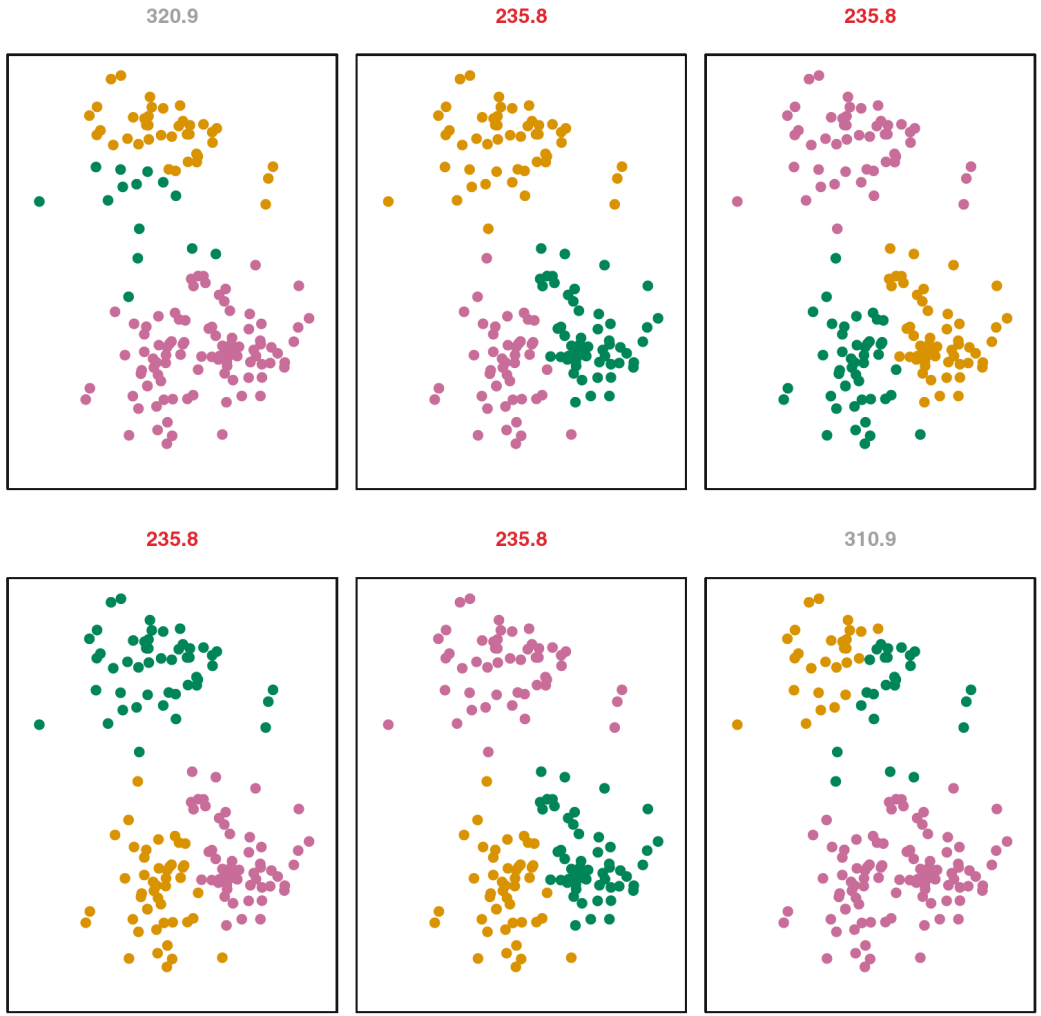
1. Randomly assign a number, from 1 to $K$, to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
   (a) For each of the $K$ clusters, compute the cluster *centroid*. The $k$th cluster centroid is the vector of the $p$ feature means for the observations in the $k$th cluster.
   (b) Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).

# K-Means Clustering: Example



# K-Means is Sensitive to Initialization

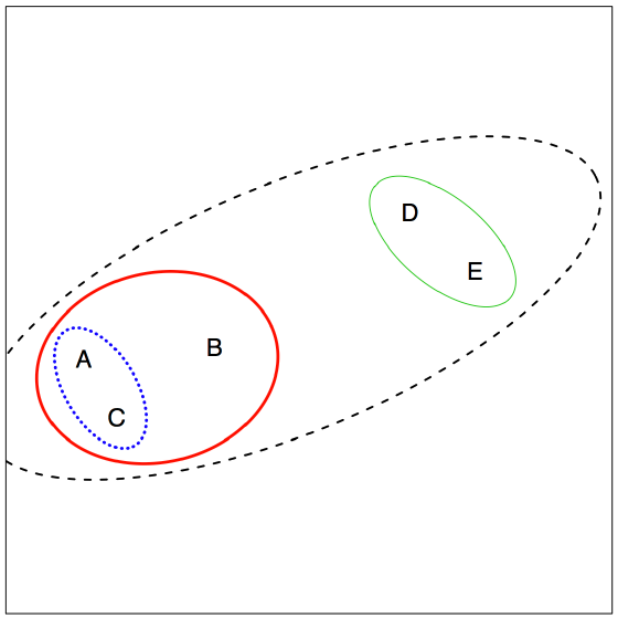
- It is often recommended to run K-Means multiple times with different random seeds and pick the one with the least cost

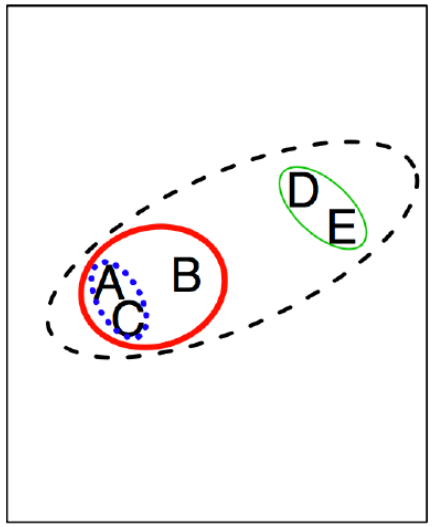


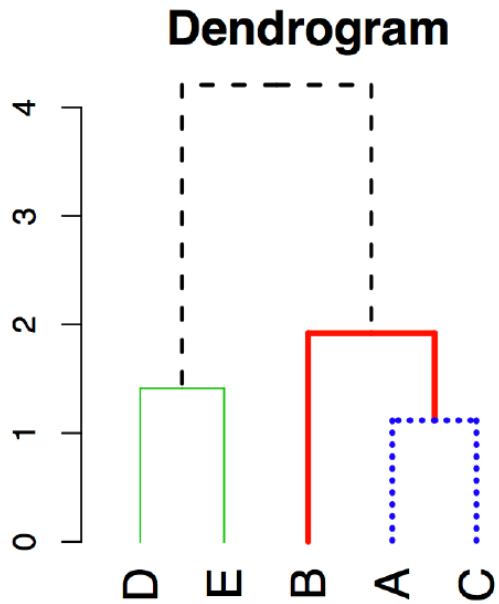
# Hierarchical Clustering

- A main drawback of K-means is the need to know $K$ in advance
- An alternative algorithm called **Hierarchical Clustering** can resolve this issue
- The output of a hierarchical clustering is a so called dendrogram, which is a tree-based illustration of the partitions from the very coarse to the very fine parititions

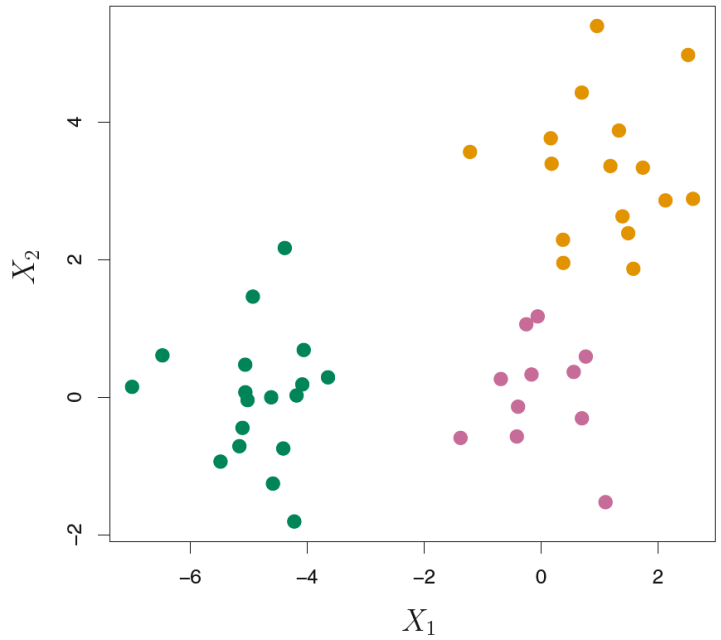
# Hierarchical Clustering: How Does It Work?

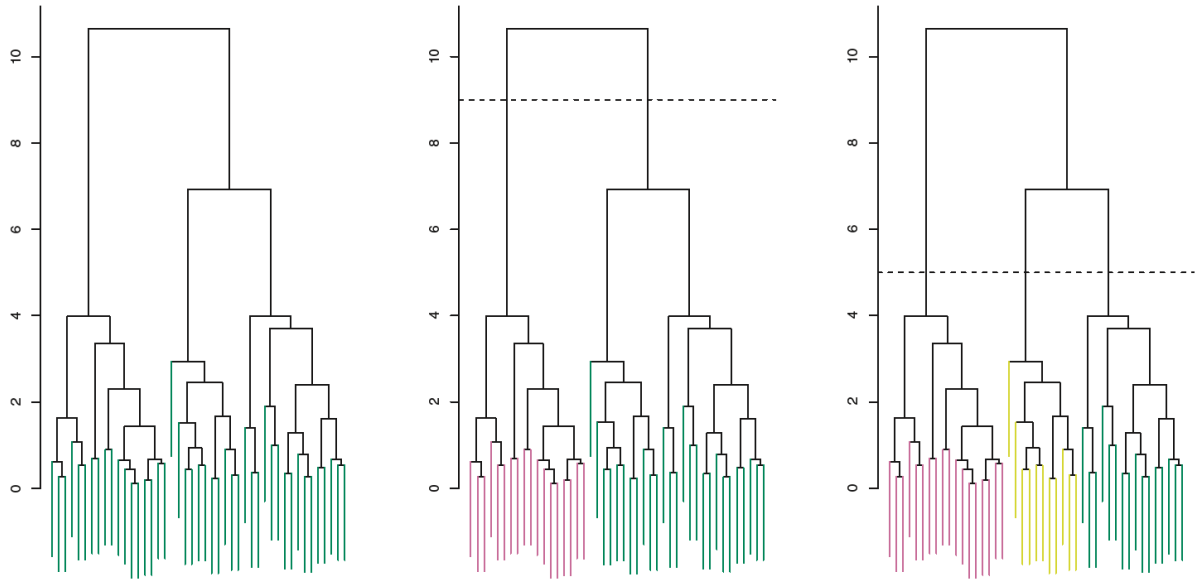


# Hierarchical Clustering: Easy Interpretation



# Hierarchical Clustering: Algorithm

**Algorithm 10.2** *Hierarchical Clustering*

1. Begin with $n$ observations and a measure (such as Euclidean distance) of all the $\binom{n}{2} = n(n-1)/2$ pairwise dissimilarities. Treat each observation as its own cluster.

2. For $i = n, n-1, \ldots, 2$:

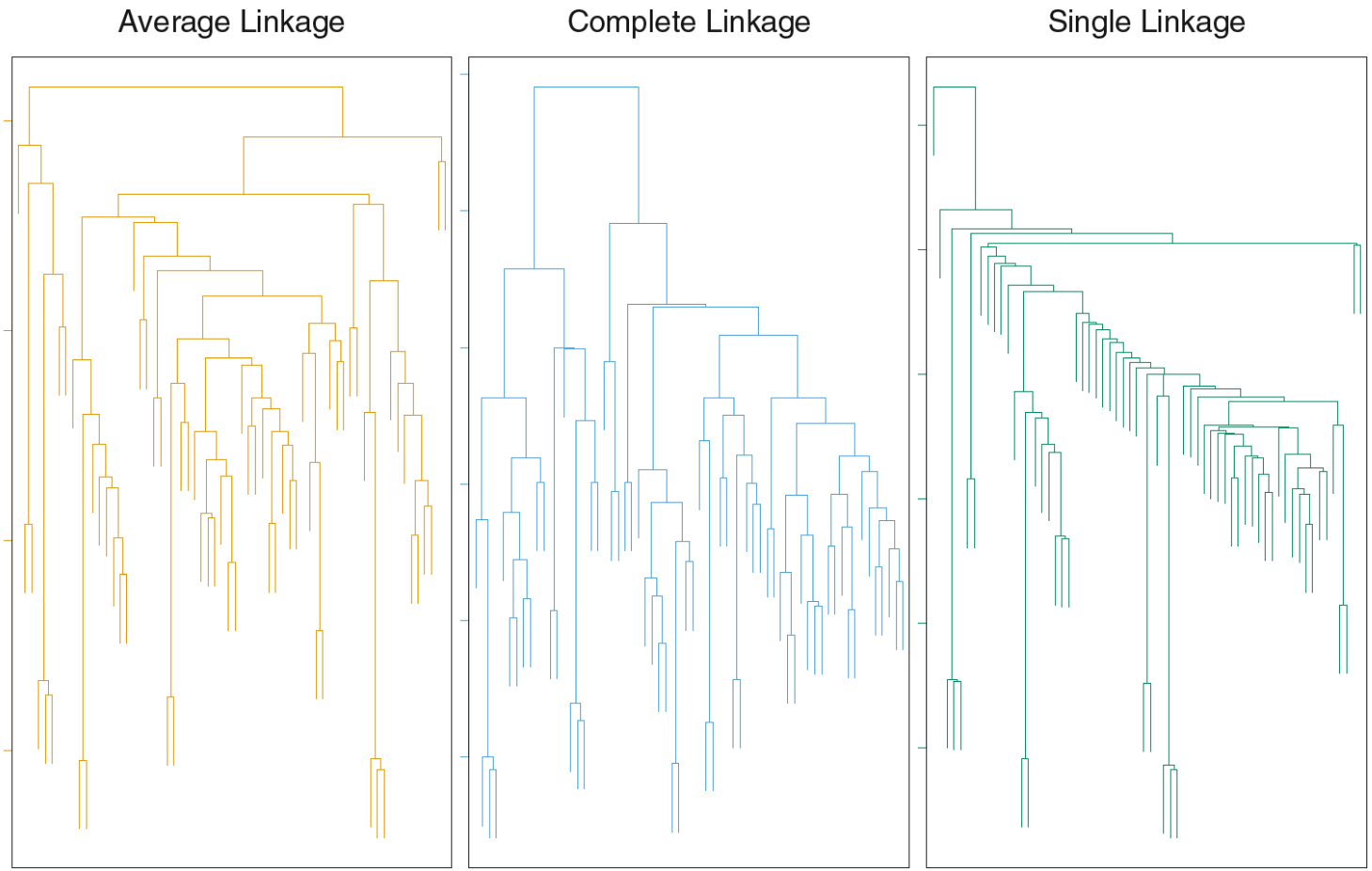
   (a) Examine all pairwise inter-cluster dissimilarities among the $i$ clusters and identify the pair of clusters that are least dissimilar (that is, most similar). Fuse these two clusters. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.

   (b) Compute the new pairwise inter-cluster dissimilarities among the $i-1$ remaining clusters.

# Hierarchical Clustering: How Do We Link Samples?

| *Linkage* | *Description* |
|---|---|
| Complete | Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities. |
| Single | Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time. |
| Average | Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities. |
| Centroid | Dissimilarity between the centroid for cluster A (a mean vector of length $p$) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*. |

# Hierarchical Clustering: Difference Between Linkage Schemes




Also the use of different distance measures (e.g., Euclidean vs correlation) can significantly change the results

# Take-Away Notes

- Standardizing the data can sometimes help (it however would change the results since the scales in different directions can be different)
- For K-Means the choice of $K$ and the distance is important
- For Hierarchical Clustering, the choice of linkage and distance are also of major importance
- How many clusters to chose? Really depends on the problem
- Unsupervised learning is intrinsically a harder problem than the supervise learning problem, mainly because of the lack of clear objective and performance evaluations

# The End