NYC DATA SCIENCE ACADEMY

Python Machine Learning Class 5: Unsupervised Learning

Data Science Bootcamp

Outlines

- Intro to Unsupervised Learning
  - Principal Component Analysis
    - Motivation
    - The Mathematical Formulation
  - Clustering
    - K-means Clustering
    - Hierarchical Clustering

Supervised Learning Recap

- Task: to predict the values of one or more response variables *Y* from a given set of predictor variables *X*.
- Prediction are based on the training data of previously solved cases.
- Performance can be estimated by some loss function (for example, *RSS* in regression or *OOB error* in bootstrap aggregating), using training-test splitting or cross-validation.
- Regression:
  - simple/multiple linear regression, regression trees, etc.
- Classification:
  - logistic regression, discriminant analysis, naive bayes, support vector machines, classification trees, etc.

Unsupervised Learning

- Only a set of *N* observations with *p* features, *no response variables*.
- Goal: to directly infer the properties without knowing the "correct" answers or the error for each observation.
- "Learning without a teacher" - No direct measure of success.
  - We discuss two methods:
    - *principal components analysis*, which is often used for data visualization or data preprocessing for supervised learning.
    - *clustering*, a broad class of methods for grouping or segmenting a collection of objects into subsets or "clusters".

Unsupervised Learning

- Unlabeled data is easier to obtain than labeled data.
- No specific prediction goals, therefore more subjective.
- We are usually not interested in predictions.
- Examples:
  - Groups of online shoppers characterized by their browsing and purchase histories.
  - Movies grouped by the comments given by movie viewers.

Outlines

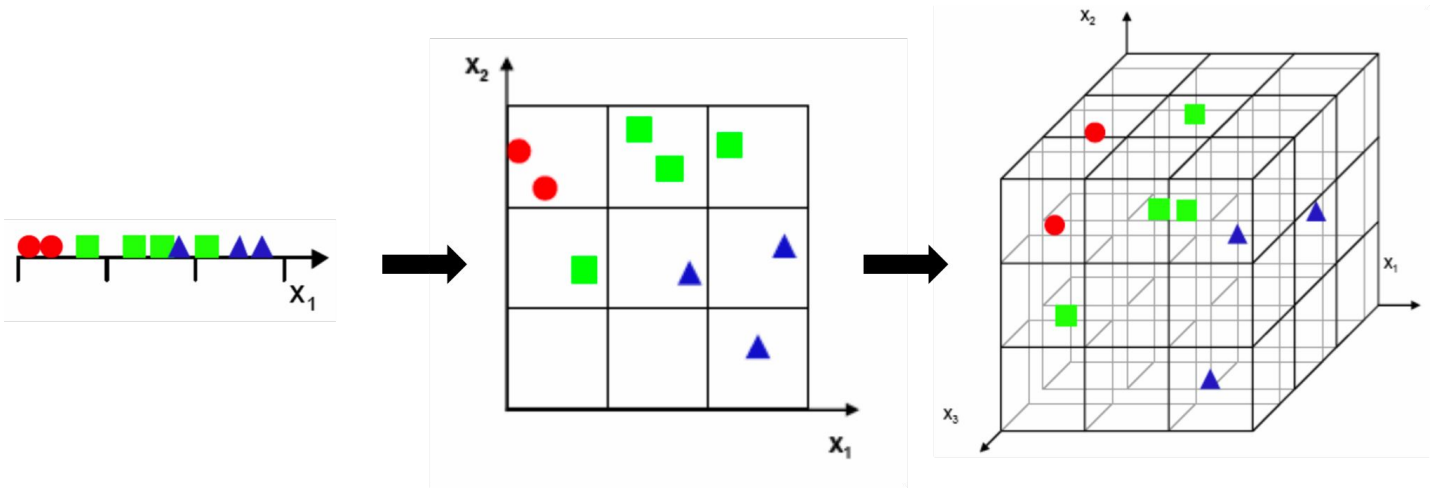
- Intro to Unsupervised Learning
- **Principal Component Analysis**
  - Motivation
  - The Mathematical Formulation
- Clustering
  - K-means Clustering
  - Hierarchical Clustering

Multicollinearity

- *Multicollinearity* is a phenomenon in which two or more predictor variables in a multiple regression model are highly correlated, meaning that one can be linearly predicted from the others with a substantial degree of accuracy.
- Issues:
  - The coefficients for highly correlated variables might be inaccurate.
  - The estimate of one variable's impact on the dependent variable *Y* while controlling for the others tends to be less precise.
  - The collinear variables contain the same information about the dependent variable, which may lead to overfitting.
  - The standard errors of the affected coefficients tend to be large.

The Curse of Dimensionality

- Given a number of observations, additional dimensions spread the points out further and further from each other.
- Sparsity becomes exponentially worse as the dimensionality of the data increases.



Principal Component Analysis

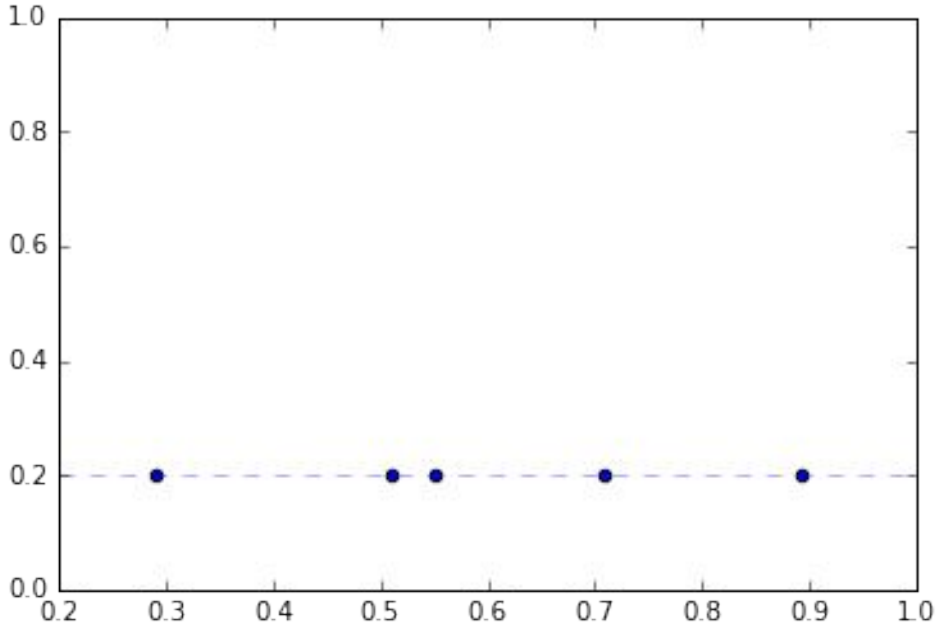
- Ideal input variables:
  - linearly uncorrelated
  - feature space has low number of dimensions
- *Principal component analysis* (PCA) is a tool that finds a sequence of linear combinations of the variables to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called *principal components*.

Outlines

- Intro to Unsupervised Learning
- Principal Component Analysis
  - **Motivation**
    - The Mathematical Formulation
- Clustering
  - K-means Clustering
  - Hierarchical Clustering

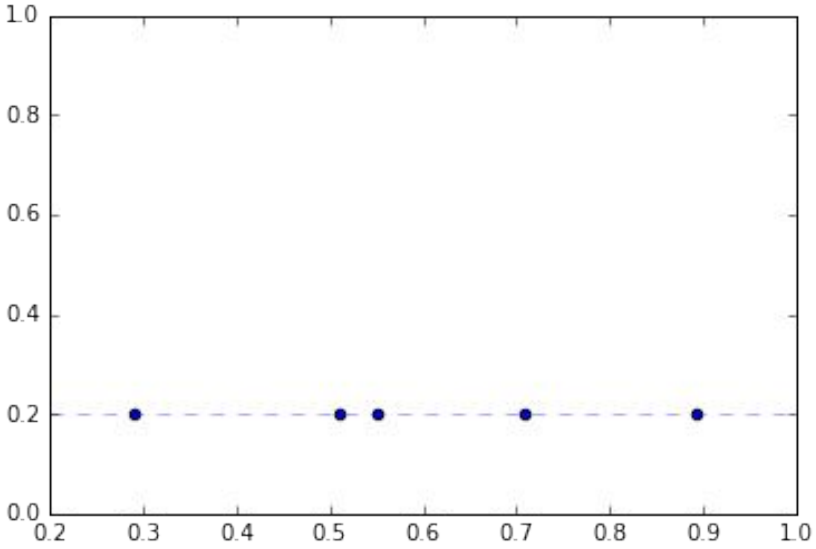
Motivation

- In some cases, it is obvious that we don't need all the component. For example:



Motivation

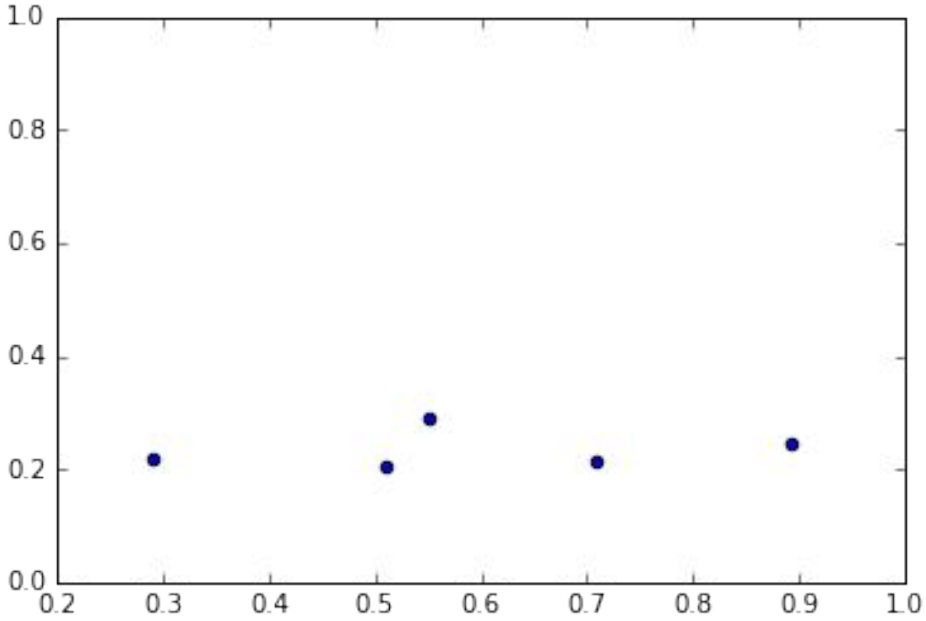
- Each observation has two components in its coordinate. Do we really need two?



- We don't. The y component of all the points are the same, it provides **NO** information.

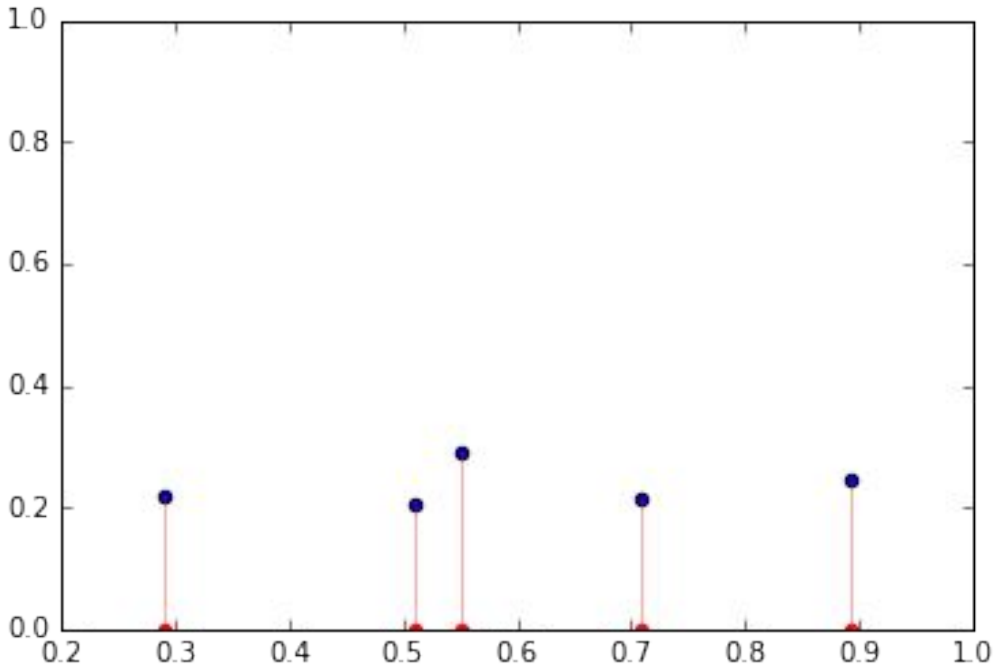
Motivation

- Consider the next example. None of the components is constant, but if we need to pick only one component among the two axes, how do we decide?



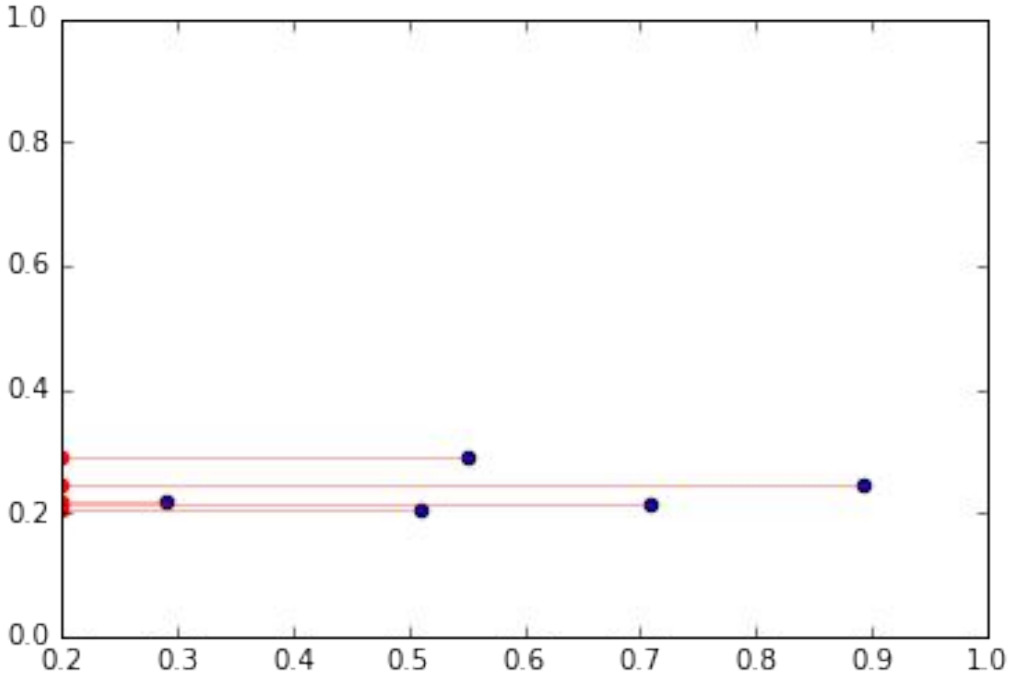
Motivation

- One way to compare is to project the observations to the two axes.
  - For x axis, the projection spreads in a range around 0.3 to 0.9.



Motivation

- In contrast, y is restricted in a much smaller region. This suggests that x component might provide more information, because all the y components are "about the same".



Motivation

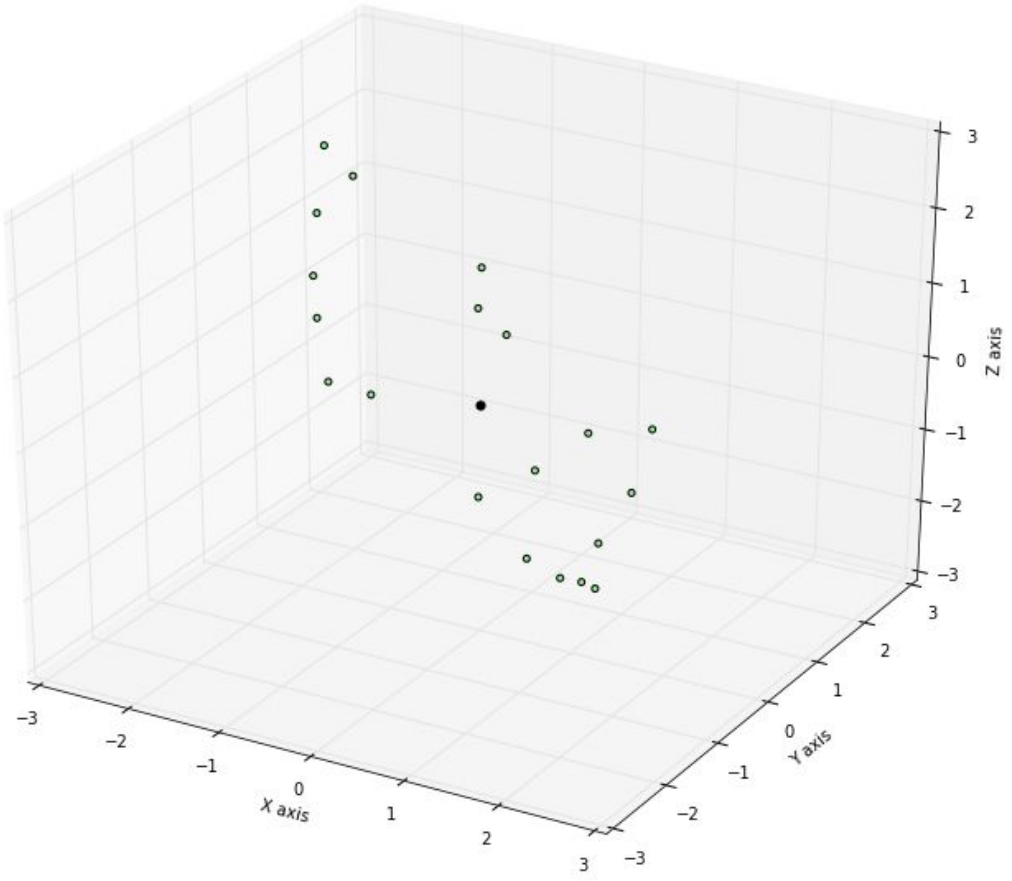
- This motivates the idea of PCA. We are searching for, among all the possible directions, the direction along which the projection of the observations are most widely spread.
- We emphasize that the motivation above can be a little misleading. Even in the two dimensional space, we can have **infinitely many** directions, not just along the two axes.
- Below we illustrate how PCA works in a three dimensional space. The process can be easily generalized into any higher dimensional space.

The First Loading Vector

- Consider a set of 20 points in a three dimensional space. This often represents a case when we have:
  - 20 observations.
  - 3 features.

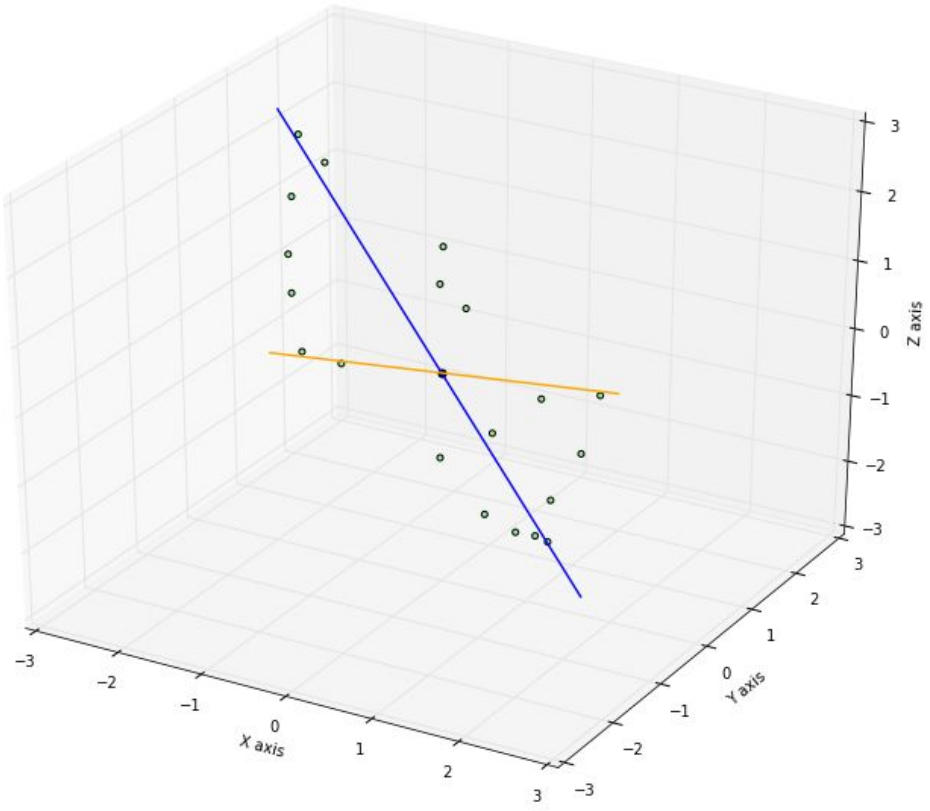
The First Loading Vector

- We visualize the data below. The black point is the origin.



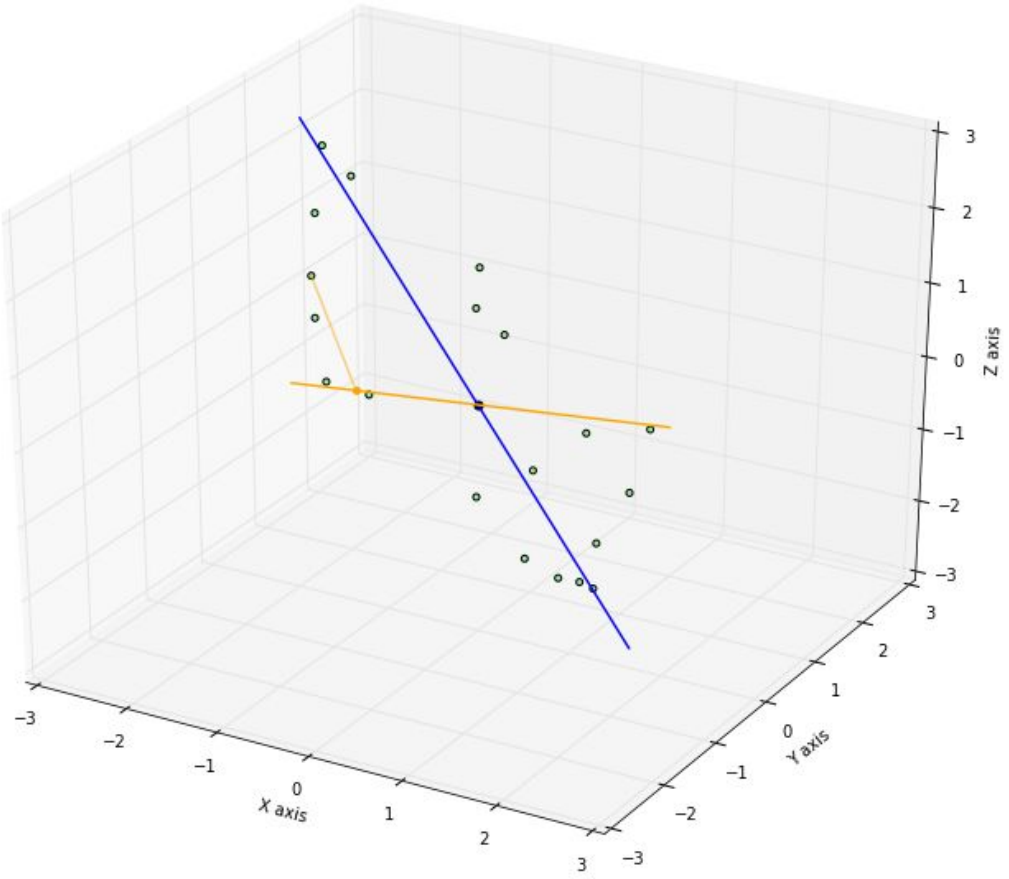
The First Loading Vector

- Below we visualize how we compare the importance of each direction. Note that the directions in the example are not along any axis.



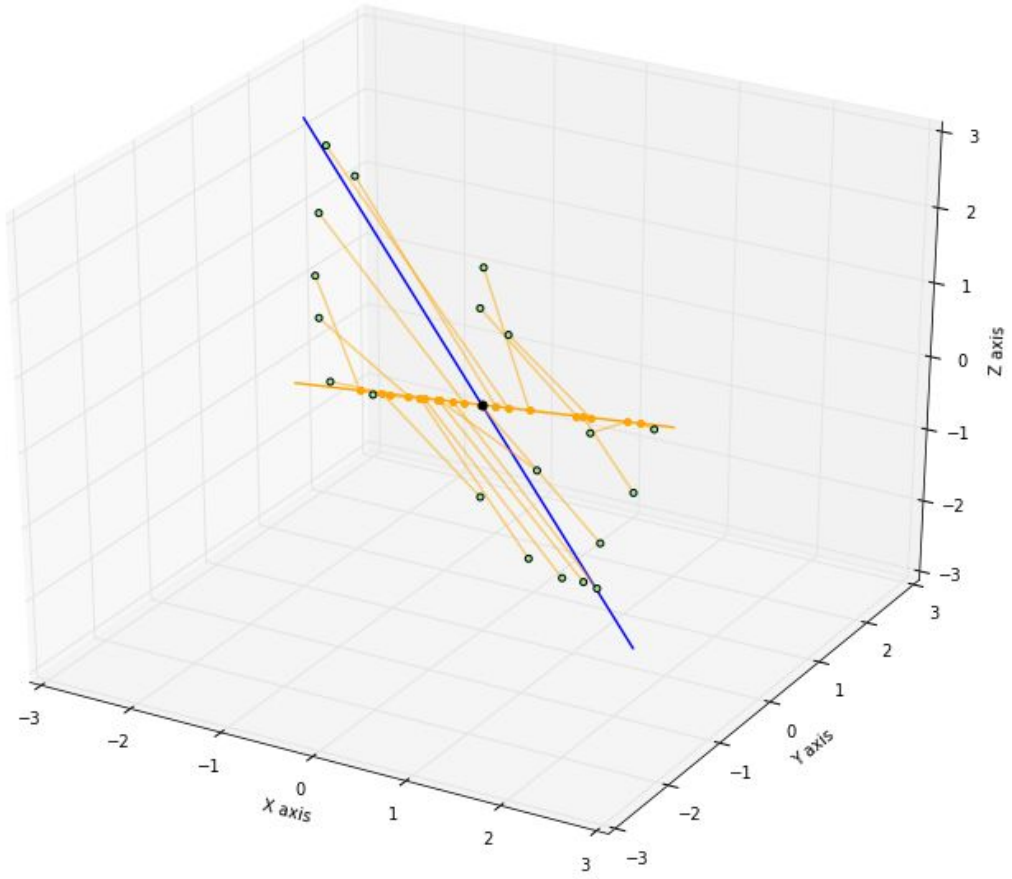
The First Loading Vector

- We project each observation to the "orange" direction:



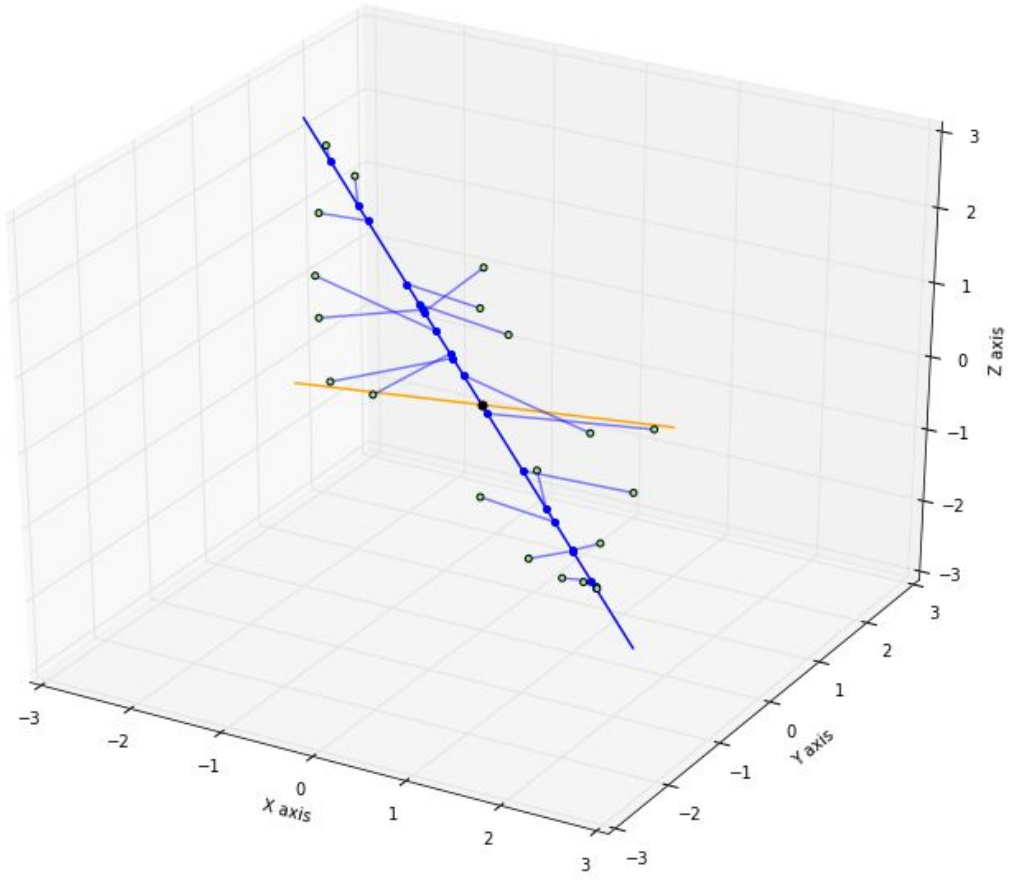
The First Loading Vector

- We project each observation to the "orange" direction:



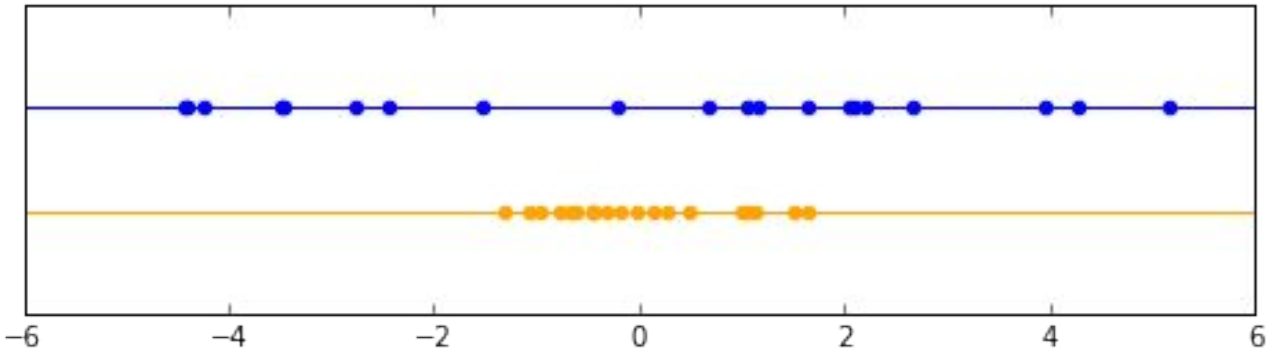
The First Loading Vector

- Project to the "blue" direction in the same way:



The First Loading Vector

- The projection of the observations on the "blue" direction is more widely spread than the one on the "orange" direction.



The First Loading Vector

- The "blue" direction above is actually the **first loading vector**, which means:
  - it is the direction on which the projection of the observations is more widely spread than the projection on any other direction.
  - being a direction (vector), it has as many components as the number of the features.

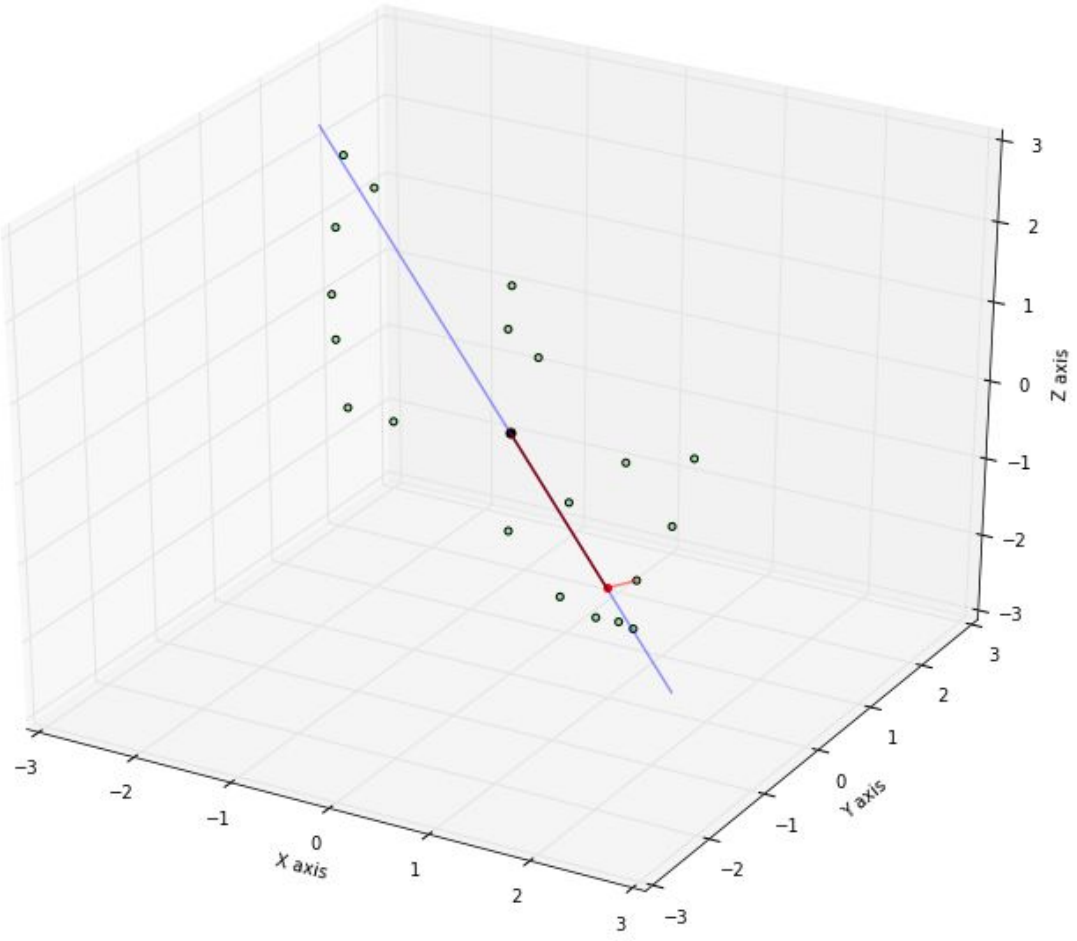
- The statements above characterize the principal component direction. To find the principal component direction we need to apply linear algebra, which we will discuss later. However, if you don't care about math, Python will find it for you.

The First Principal Component

- With the first loading vector (heuristically the most important one), we want to keep, for all the observations, only the information recorded in this direction.

  - This is again done by **projection**.

The First Principal Component

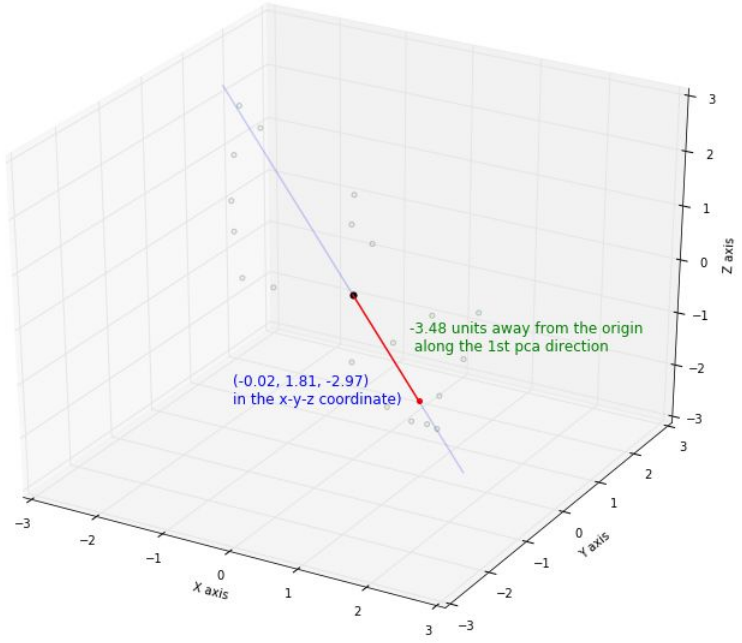


The First Principal Component

- In this particular example, projecting 20 observations to the first loading vector gives us a vector of length 20. This vector is the first principal component.
- **Remark**
  - We have 20 observations originally, so we still need 20 records for all the observations.
  - We no longer use the x-y-z coordinates to record the information for each observation, with the first principal component we use **one** coordinate only -- but this coordinate is supposed to provide the most information.

The First Principal Component

- We also remark here a little side effect that the projection (red part) can be described in two ways:
  - a certain length away from the origin along the PCA direction.
  - a vector in the original x-y-z coordinate.

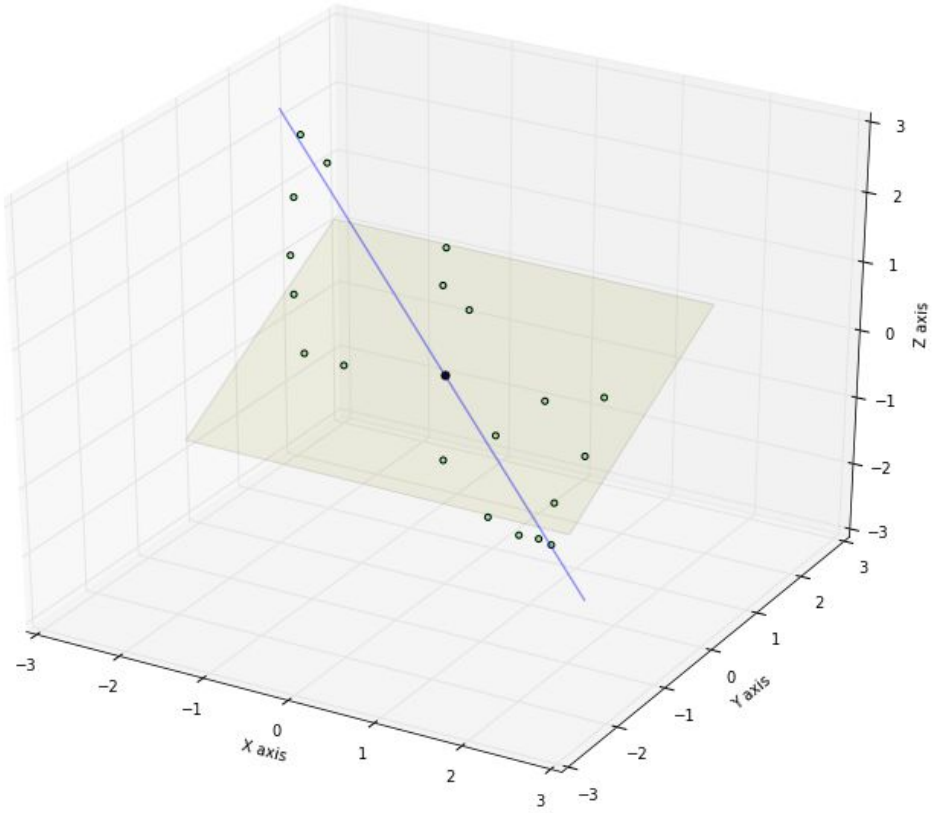


The Second Principal Component

- The first principal component provides the most information, but most likely NOT all. To find the next most important direction, we need to:
  - First, remove the information recorded in the first principal component.
  - Then we again find the direction on which the projection of the observations most widely spread.

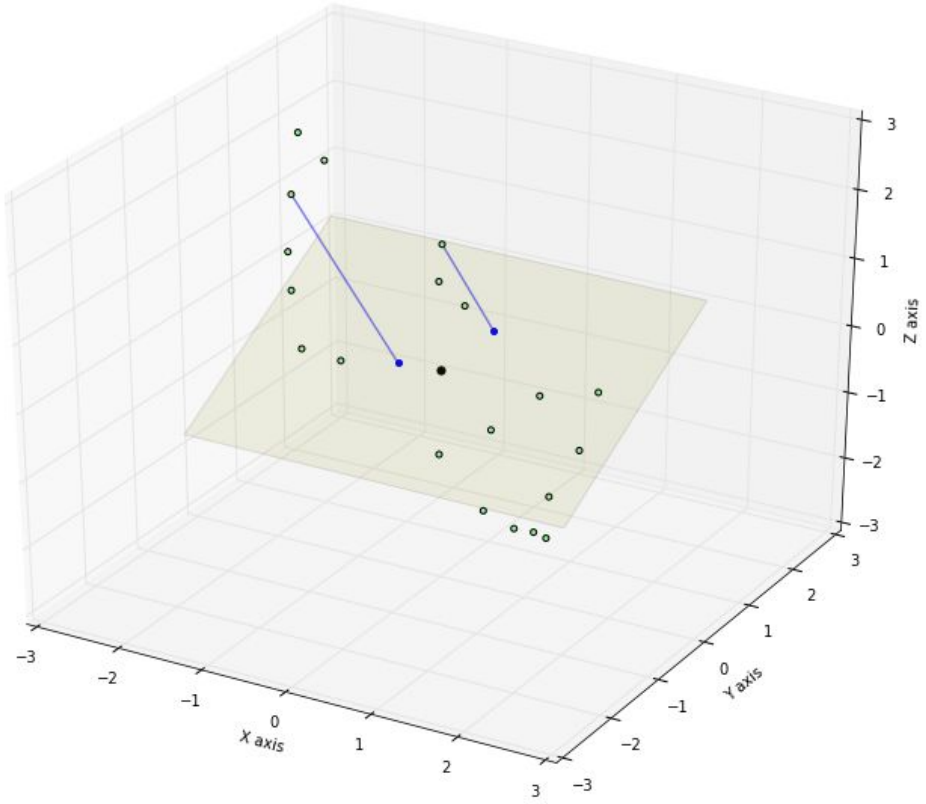
The Second Principal Component

- Consider the plane that is perpendicular to the first loading vector.



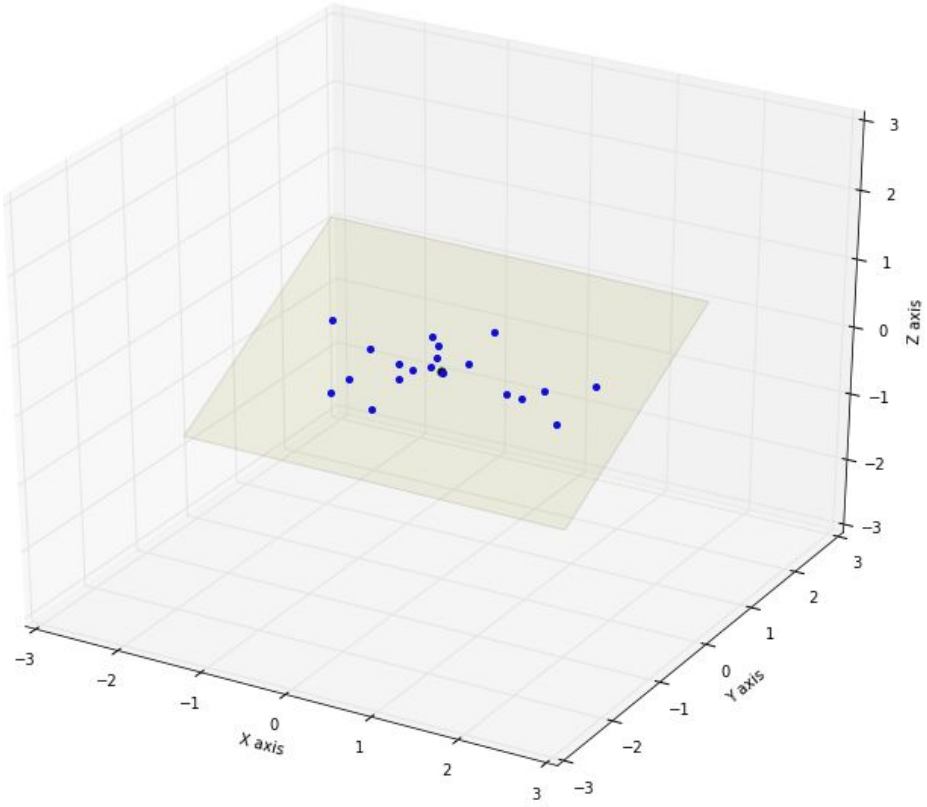
The Second Principal Component

- We can remove the effect of the first principal component by projecting the observations to this plane:



The Second Principal Component

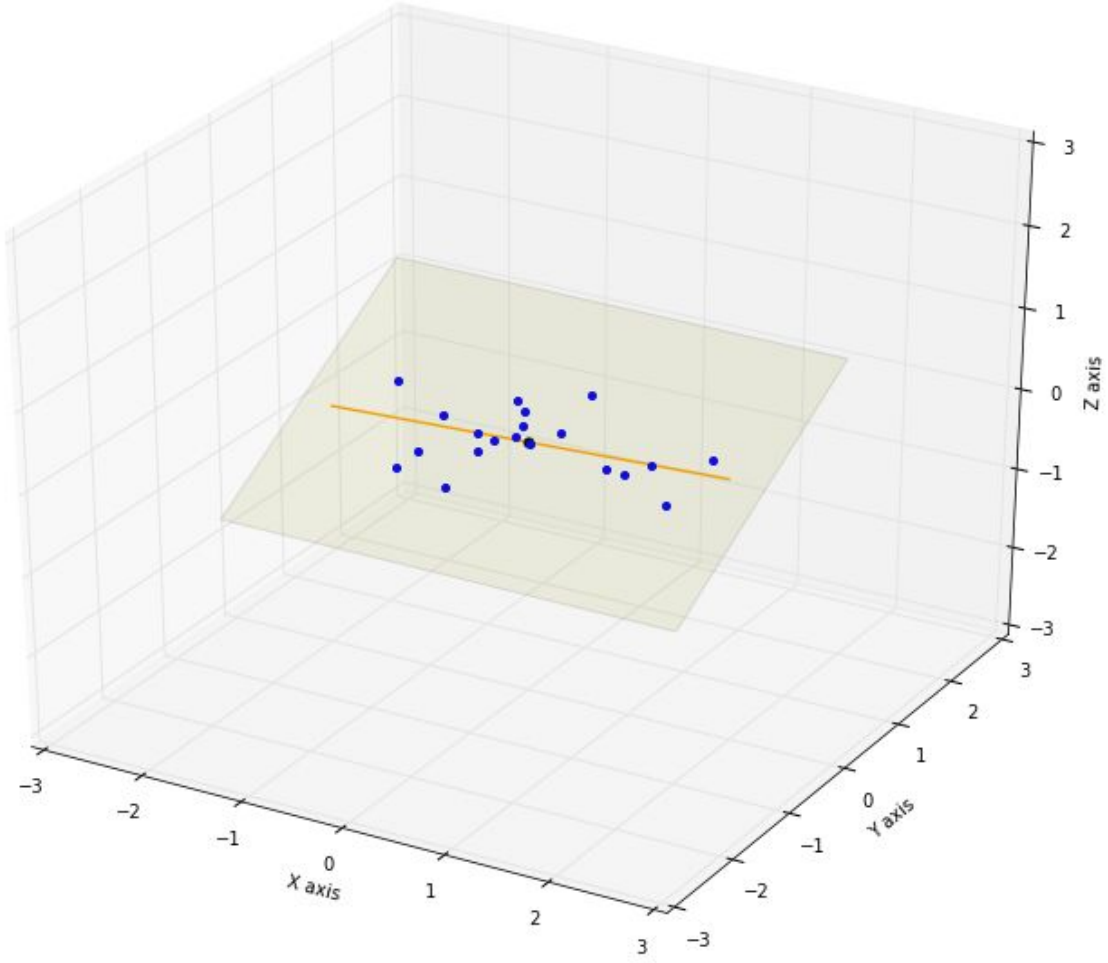
- We can remove the effect of the first principal component by projecting the observations to this plane:



The Second Principal Component

- Apply the same trick in linear algebra (which we didn't discuss), we can find the direction on which the projections most widely spread.
  - Since all the observations are now **in the plane**, the direction we find would be automatically **in the plane** and is perpendicular to the first loading vector.
  - This is the **second loading vector**. The projection of the observations to this direction is the **second principal component**.

The Second Principal Component

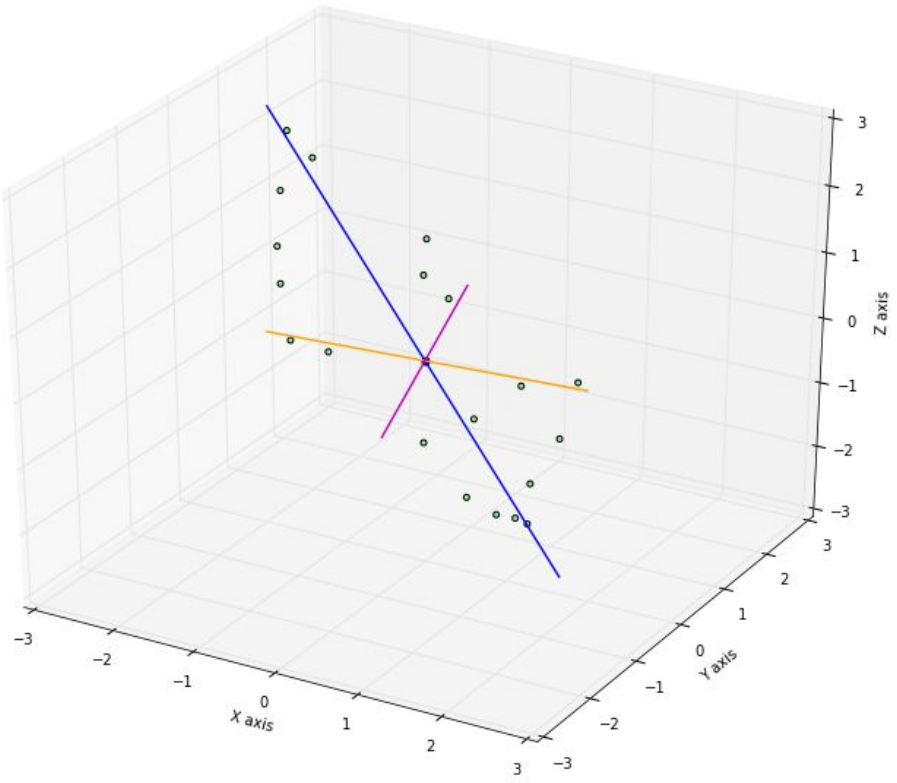


The Second Principal Component

- **Remark** Clearly this process can be continue to retain more and more information. However, as we can see from the process above, each time we remove the information recorded in a principal component direction, we remove one dimension. So we can't have more principal components than the amount of original features we have.

The Second Principal Component

- Below we visualize, for the example above, the observations and all its loading vectors:



Outlines

- Intro to Unsupervised Learning
- Principal Component Analysis
  - Motivation
  - **The Mathematical Formulation**
- Clustering
  - K-means Clustering
  - Hierarchical Clustering

The Mathematical Formulation

- In this section we identify all the elements we saw in the visualization with the mathematical formula. It is a good transition between the visualization and the Python code. To be compatible with the notation in Python, vectors are row vectors below.
- The first (very important) step we need to do is to centralize the data. We may then assume our data is X an n by p matrix (that means we have a data set of n observations and p features). The average of each column is 0.
- We then project the data into any possible direction. A direction is represented by a unit vector $\hat{u}$ in linear algebra, and the projection is

$$X\hat{u}^{\mathrm{T}}$$

The Mathematical Formulation

- **Question**
  - What is the dimension of $\hat{u}$?
  - What is the dimensions of $X\hat{u}^{\mathrm{T}}$?
  - Why do these dimension make sense? Or not?

The Mathematical Formulation

- As we discussed, we need to find the direction on which the projection of the data **most widely spread**. In the mathematical formulation, it can be stated as:

$$\text{maximize } \text{Var}(X\hat{u}^{\text{T}})$$
$$\text{subject to } \|\hat{u}\| = 1$$

- The solution to the optimization problem above is the **first loading vector**, denoted by ϕ_1. The projection of our data X on the first loading vector is

$$Z_1 = X\phi_1^{\text{T}}$$

which is called the **first principal component**.

The Mathematical Formulation

- Once the first k-1 principal components are found, the next one (if there is one) can be found inductively.
  - We first remove the information about the first k-1 components from X (X_k denotes the resulted matrix).

$$X_k = X - \sum_{i=1}^{k-1} X\phi_i\phi_i^{\mathrm{T}}$$

  - With this matrix we solve the optimization problem again:

$$\text{maximize } \mathrm{Var}(X_k\hat{u}^{\mathrm{T}})$$
$$\text{subject to } \|\hat{u}\| = 1$$

The Mathematical Formulation

- Again the solution ϕ_k is the **k_{th} loading vector** and the projection on this direction,

$$Z_k = X_k \phi_k^T$$

  is called the **k_{th} principal component**.

- **Note:** Solving an optimization problem can be hard. In the setting of PCA, this is relatively easy. The principal loading vectors are (essentially) the eigenvectors of the covariance matrix of the data, arranged in the descending order of the eigenvalues they correspond to.

The Properties of Principal Components

- There are most `min(n, p)` principal components (but we often assume p of them).
- The variances of each principal components decreases:

$$\text{Var}(Z_1) \geq \text{Var}(Z_2) \geq \ldots \geq \text{Var}(Z_p)$$

- The principal components $Z_1, Z_2, \ldots, Z_p$ are mutually uncorrelated.
- The principal loading vectors $\phi_1, \phi_2, \ldots, \phi_p$ are normalized and mutually perpendicular.

Hands-on Session

- Please go to the **"PCA in Scikit Learn"** in the lecture code.

Outlines

- Intro to Unsupervised Learning
- Principal Component Analysis
  - Motivation
  - The Mathematical Formulation
- **Clustering**
  - K-means Clustering
  - Hierarchical Clustering

What Is Cluster Analysis?

- Up to this point, for the most part we've been concerned with building models that perform predictions:
  - *Regression* systems that attempt to predict a numeric output.
  - *Classifiers* that attempt to predict class membership.
- In contrast, cluster analysis is an unsupervised task that:
  - Does not aim to specifically predict a numeric output or class membership.
  - Does aim to uncover underlying structure of the data and see what patterns exist in the data.
    - We aim to group together observations that are similar while separating observations that are dissimilar.

What Is Cluster Analysis?

- Cluster analysis attempts to explore possible subpopulations that exist within your data.
- Typical questions that cluster analysis attempts to answer are:
  - Approximately how many subgroups exist in the data?
  - Approximately what are the sizes of the subgroups in the data?
  - What commonalities exist among members in similar subgroups?
  - Are there deeper subgroups that can further segment current subgroups?
  - Are there any outlying observations?
- Notice that these questions are largely exploratory in nature.

Categories of Cluster Analysis

- Prototype-based
  - Example: K-means clustering
- Hierarchical-based
  - Hierarchical clustering
- Density-based
  - Density-based Spatial Clustering of Applications with Noise (DBSCAN)
    - Not covered in this course

Prototype-based Clustering

- Each cluster is represented by a prototype which may be:
  - Centroid - average of similar points with continuous features
  - Medoid - most representative or frequently occurring point for categorical features
- K-means is an example
  - K-means uses centroids

Outlines

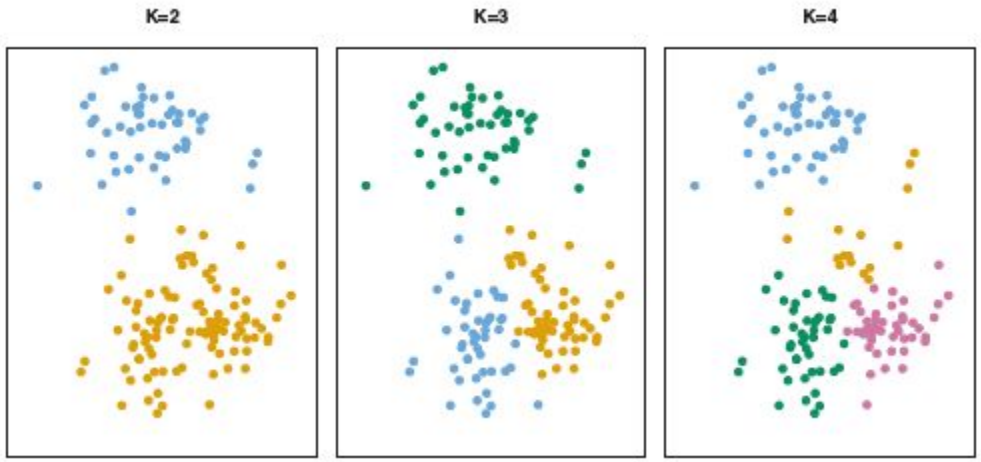
- Intro to Unsupervised Learning
- Principal Component Analysis
  - Motivation
  - The Mathematical Formulation
- Clustering
  - **K-means Clustering**
  - Hierarchical Clustering

K-Means Clustering

- With the K-means clustering algorithm, we aim to split up our observations into a predetermined number of clusters.
  - You must specify the number of clusters K in advance.
  - These clusters will be distinct and non-overlapping.
- The points in each of the clusters are determined to be similar to a specific centroid value:
  - The centroid of a cluster represents the average observation of a given cluster; it is a single theoretical observation that represents the prototypical member that exists within the cluster.
  - Each observation will be assigned to exactly one of the K clusters depending on where the observation falls in space relative to the cluster centroid locations.

K-Means Clustering

- A simulated data set with 150 observations in 2-dimensional space.



- The color indicates the cluster to which it was assigned. Note that the cluster coloring is arbitrary since there is no ordering of the clusters.

K-Means Clustering

- Now the main question: what technique does k-means algorithm use to create these clusters?
- Suppose we define the concept of distance using Euclidean measurement. Then the within-cluster variation is defined as:

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

- Here:
  - $|C_k|$ denotes the total number of observations in cluster k.
  - i and i' denote indices of observations in cluster C_k.
  - p is the number of variables/parameters in our dataset.

K-Means Clustering

- Since the within-cluster variation is a measure of the amount by which the observations in a specific cluster differ from one another, we want to minimize this quantity $W(C_k)$ over all clusters:

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

- In other words, we desire to partition the observations into K clusters such that the total within-cluster variation summed across all K clusters is as small as possible; the optimization problem for K-means is as follows:

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

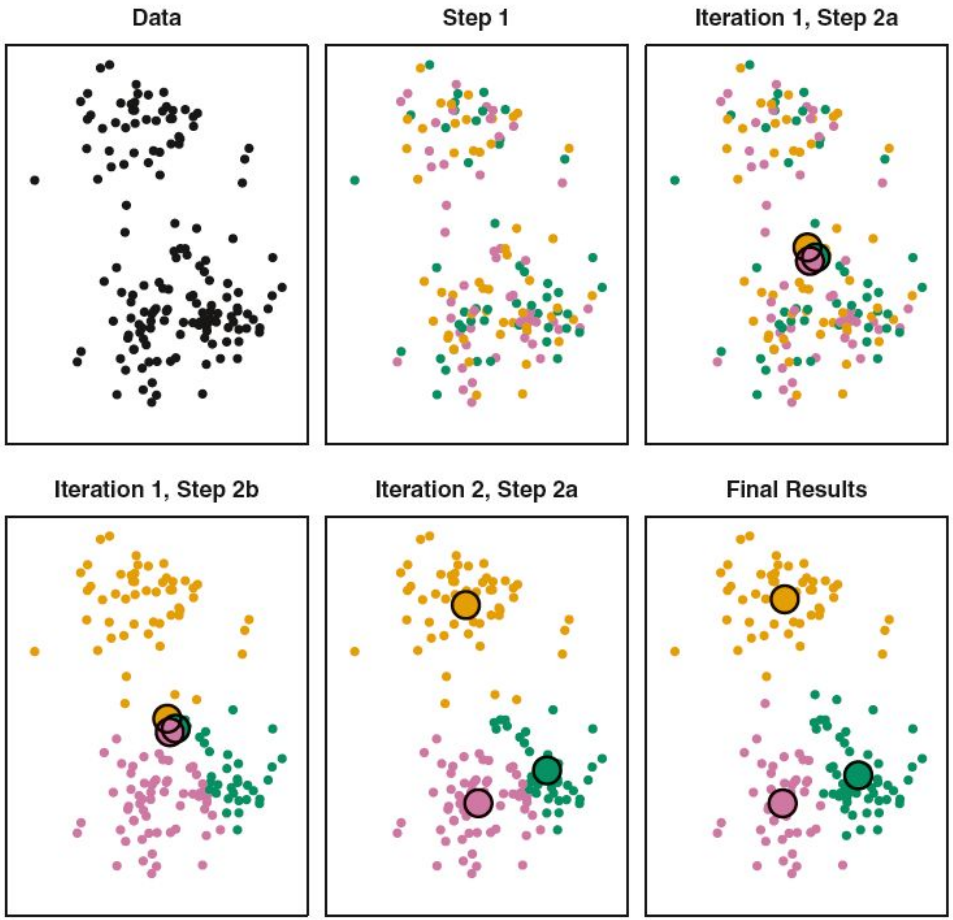
K-Means Clustering

- In practice, most packages perform the following algorithm:
  1. Randomly assign a number, from 1 to K (where K is the number of clusters), to each of the observations. These serve as initial cluster assignments for the observations.
  2. Iterate until the cluster assignments stop changing:
     i. For each of the K clusters, compute the cluster's centroid.
     ii. Assign each observation to the cluster whose centroid is closest (closest is defined using Euclidean distance).

Note: scikit-learn implements this algorithm with KMeans(init='random'). This is **not** the default init value

K-Means Clustering

- The algorithm in action (cited from ISLR):



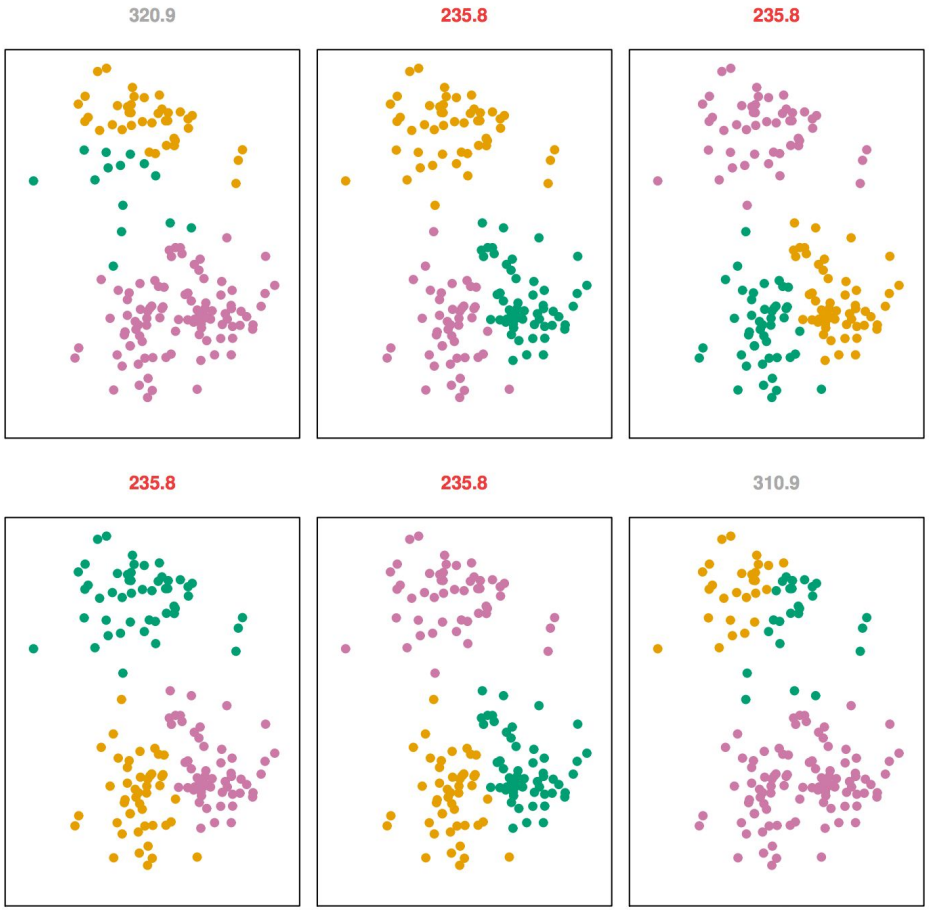
K-Means Clustering

- The K-means procedure always reaches convergence:
  - If you run the algorithm from a fixed beginning point, it will reach a stable endpoint where the clustering solution will no longer change.
- Unfortunately, the guaranteed convergence is to a local minimum.
  - Thus, if we begin the K-means algorithm with a different initial configuration, it is possible that convergence will find different centroids and therefore ultimately different cluster memberships.
- What do we do to get around this?
  - Run the K-means procedure several times and pick the clustering solution that yields the smallest aggregate within-cluster variance.

K-Means Clustering

- The algorithm will stop when it has found the least/best (local optimum) value.

(cited from ISLR)



K-Means++ Clustering

- Classic K-means can sometimes result in bad clustering or slow convergence
- K-means++ places the initial centroids far away from each other
  - This algorithm typically leads to better and more consistent results than the classic k-means

Note: scikit-learn defaults to K-means++ clustering with the KMeans() object

K-Means++ Clustering

The K-Means++ algorithm is an elaboration on the classic algo:

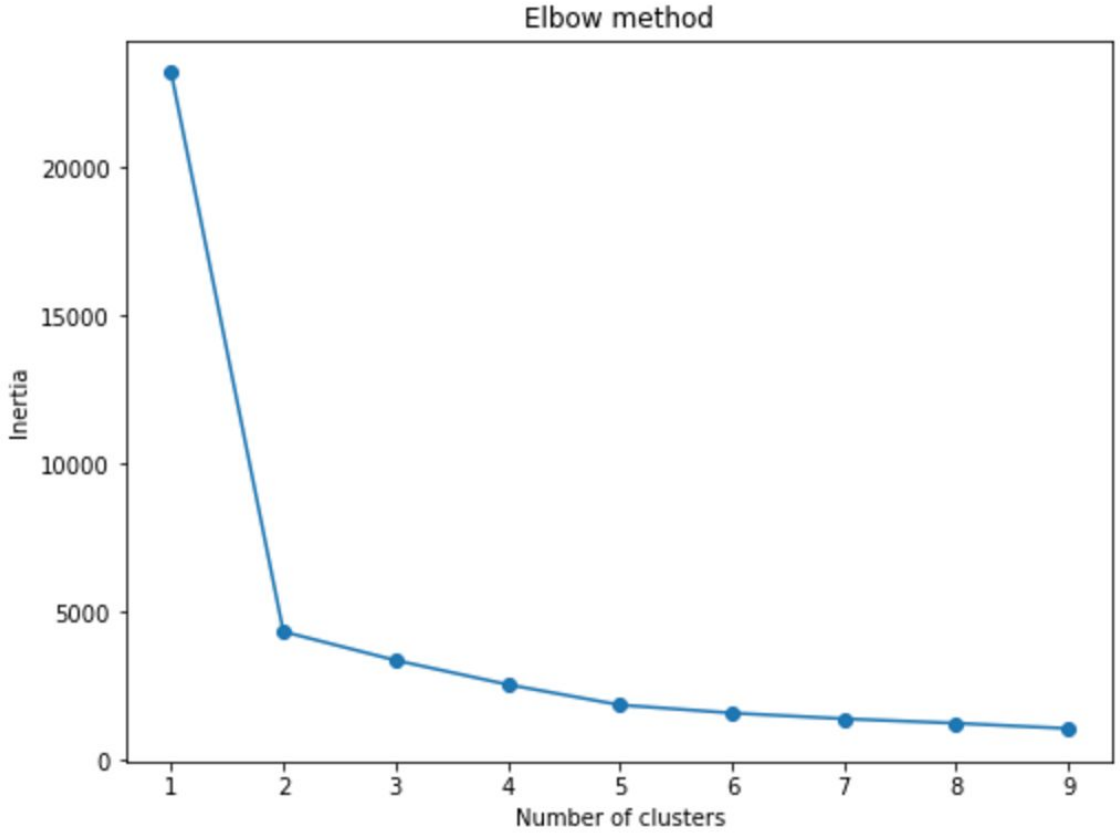
1. Initialize an empty set **M** to store k centroids
2. Randomly choose the first centroid $\boldsymbol{\mu}^{(j)}$ from input samples; assign to **M**
3. For each sample $\boldsymbol{x}^{(i)}$ not in **M**, find min squared distance $d(\boldsymbol{x}^{(i)}, \mathbf{M})^2$ to any of the centroids in **M**
4. To randomly select the next centroid $\boldsymbol{\mu}^{(p)}$, use a weighted probability distribution equal to

$$\frac{d(\mu^{(p)}, M)^2}{\sum_i d(x^{(i)}, M)}$$

1. Repeat steps 2 and 3 u…troids are chosen
2. Proceed with the classic k-means algorithm

Elbow method

- There is no way to measure the optimal number of clusters, but we can approximate with the elbow method.



Silhouette analysis

- **Silhouette analysis** is a second intrinsic metric to evaluate the quality of a clustering
- A **silhouette coefficient** is calculated for each sample
  - It compares the cluster cohesion to the cluster separation
- The silhouette coefficient is bounded in the range -1 to 1
  - 0 means that the cluster cohesion is equal to the cluster separation
  - An ideal clustering will give an average silhouette coefficient close to 1 and have few outliers that are closer to 0

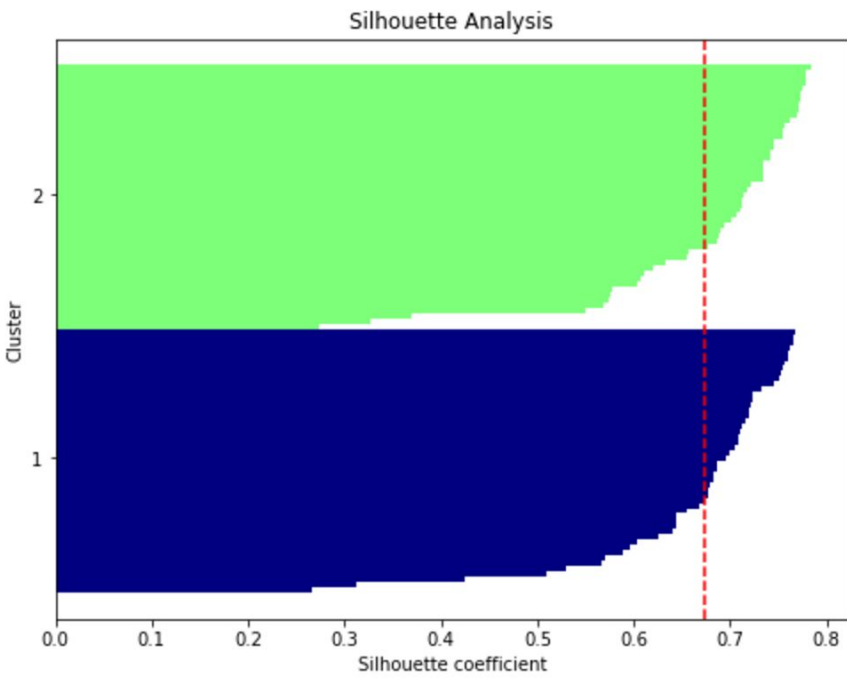
Silhouette Analysis - Formulation

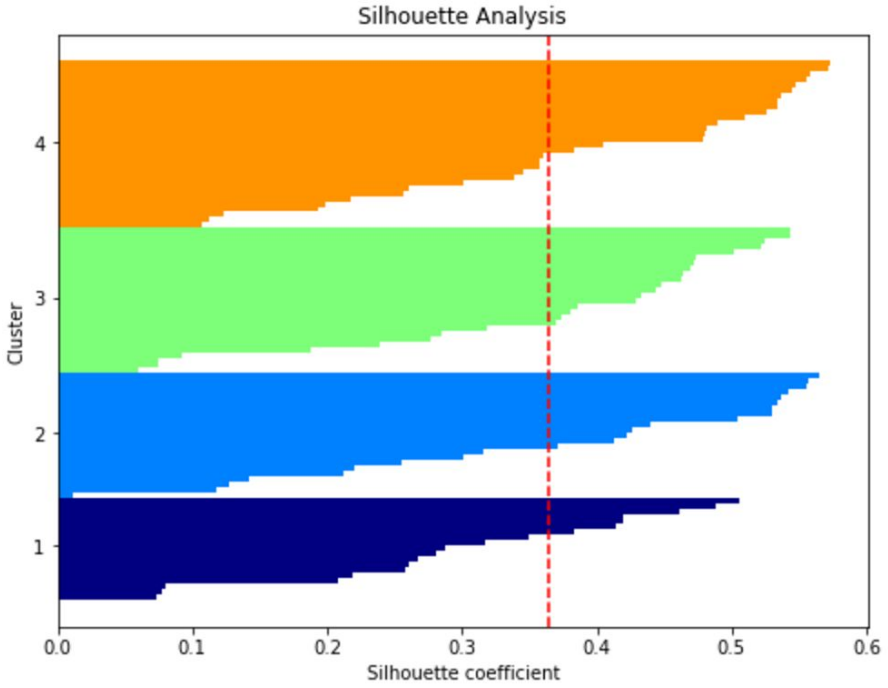
- The silhouette coefficient can be calculated in three steps:

1. Calculate cluster cohesion $a^{(i)}$ as the average distance between a sample $\boldsymbol{x}^{(i)}$ and all other points in the same cluster
2. Calculate the cluster separation $b^{(i)}$ from the next closest cluster as the average distance between the sample $\boldsymbol{x}^{(i)}$ and all the samples in the nearest cluster
3. Calculate the silhouette $s^{(i)}$ as the difference between cluster cohesion and separation divided by the greater of the two, as shown here:

$$s^{(i)} = \frac{b^{(i)} - a^{(i)}}{\max(b^{(i)}, a^{(i)})}$$

Silhouette Analysis - Good and Bad fit examples





Hands-on Session

- Please go to the **"K-means in Scikit Learn"** in the lecture code to meet the lovely panda!

Outlines

- Intro to Unsupervised Learning
- Principal Component Analysis
  - Motivation
  - The Mathematical Formulation
- Clustering
  - K-means Clustering
  - **Hierarchical Clustering**

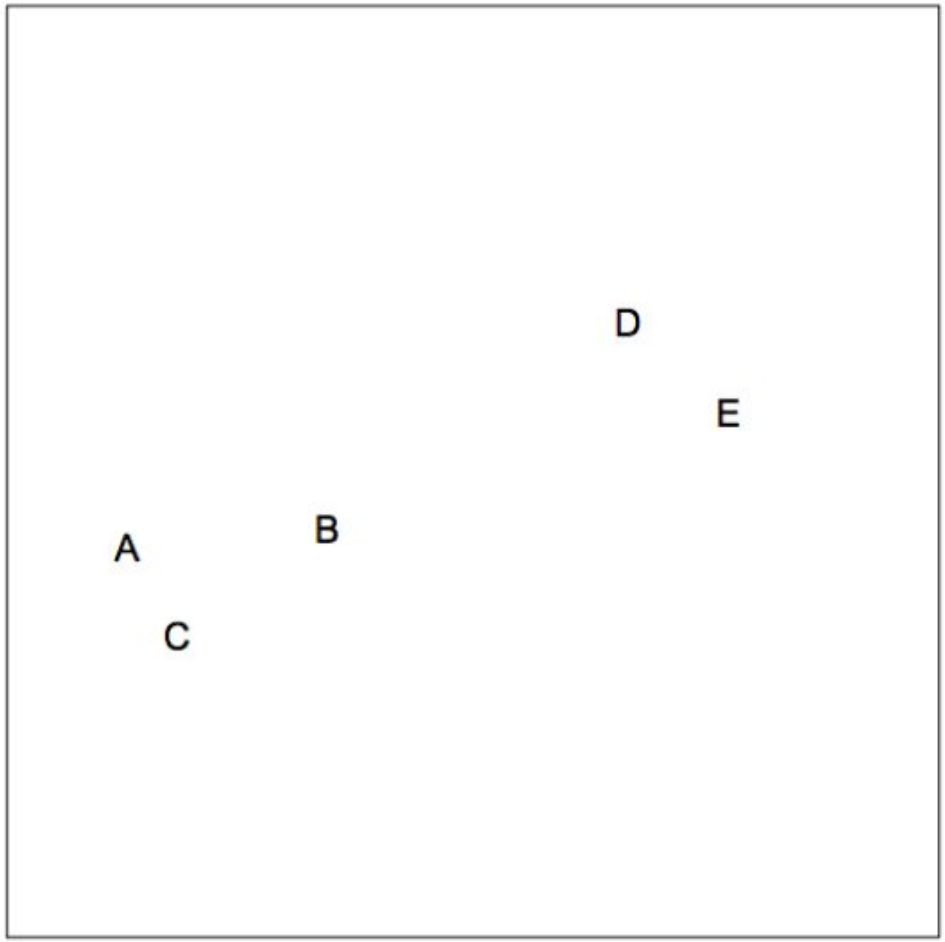
Hierarchical Clustering

- K-means clustering algorithms require: (1) the choice of the number of classes to be clustered; (2) a starting configuration assignment.
  - In most cases, this is hard to determine.
- *Hierarchical clustering* method is another clustering method which seeks to build a hierarchy of clusters.
  - It does not require to specify the number of clusters. Instead, it requires to measure the dissimilarity between groups.
  - The clusters at each level are created by merging clusters at the next lower level:
    - At the lowest level, each cluster contains a single observation.
    - At the highest level, all of the data is contained in a single cluster.

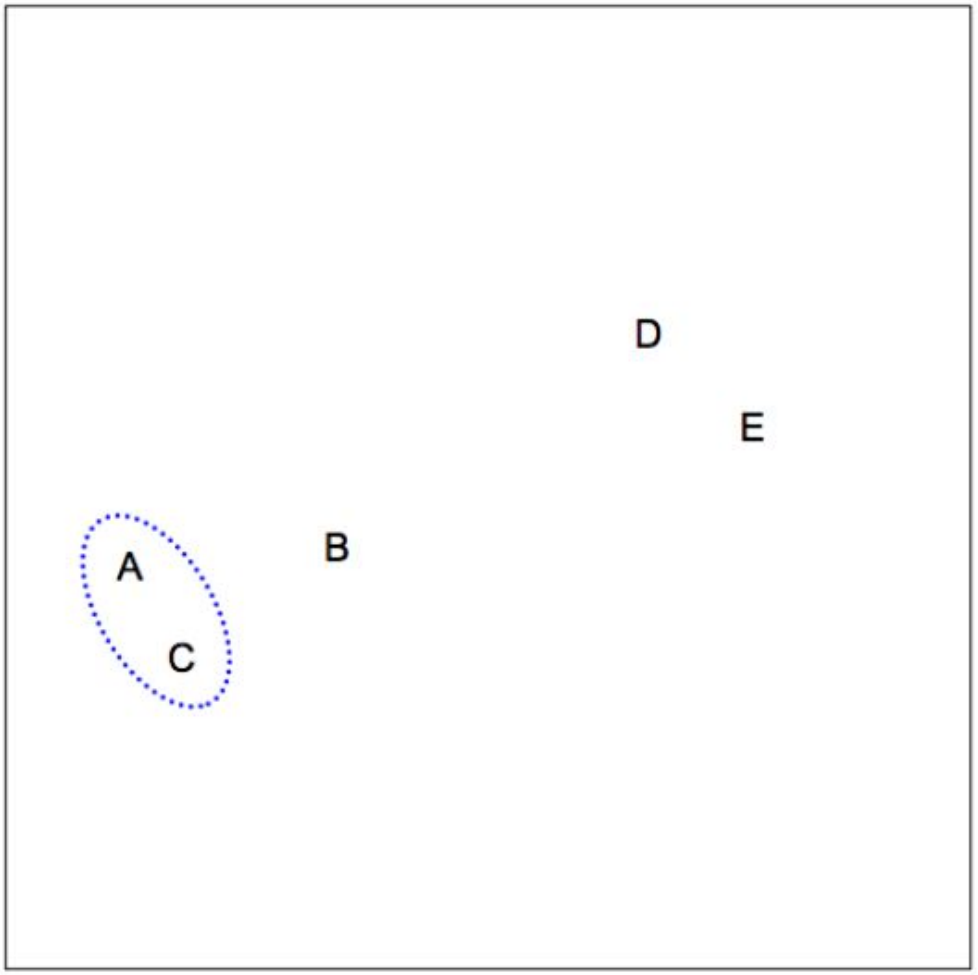
Hierarchical Clustering

- Two strategies for hierarchical clustering: *bottom-up* and *top-down*.
- Bottom-up is also known as **agglomerative clustering**, and top-down is also called **divisive clustering**.
- In the next few slides we show how to build a hierarchy in a bottom-up fashion. (cited from ISLR)
- The approach can be summarized as:
  1. Start with each point in its own cluster.
  2. Identify the closest two clusters and merge them.
  3. Repeat step 2.
  4. Ends when all points are in a single cluster.

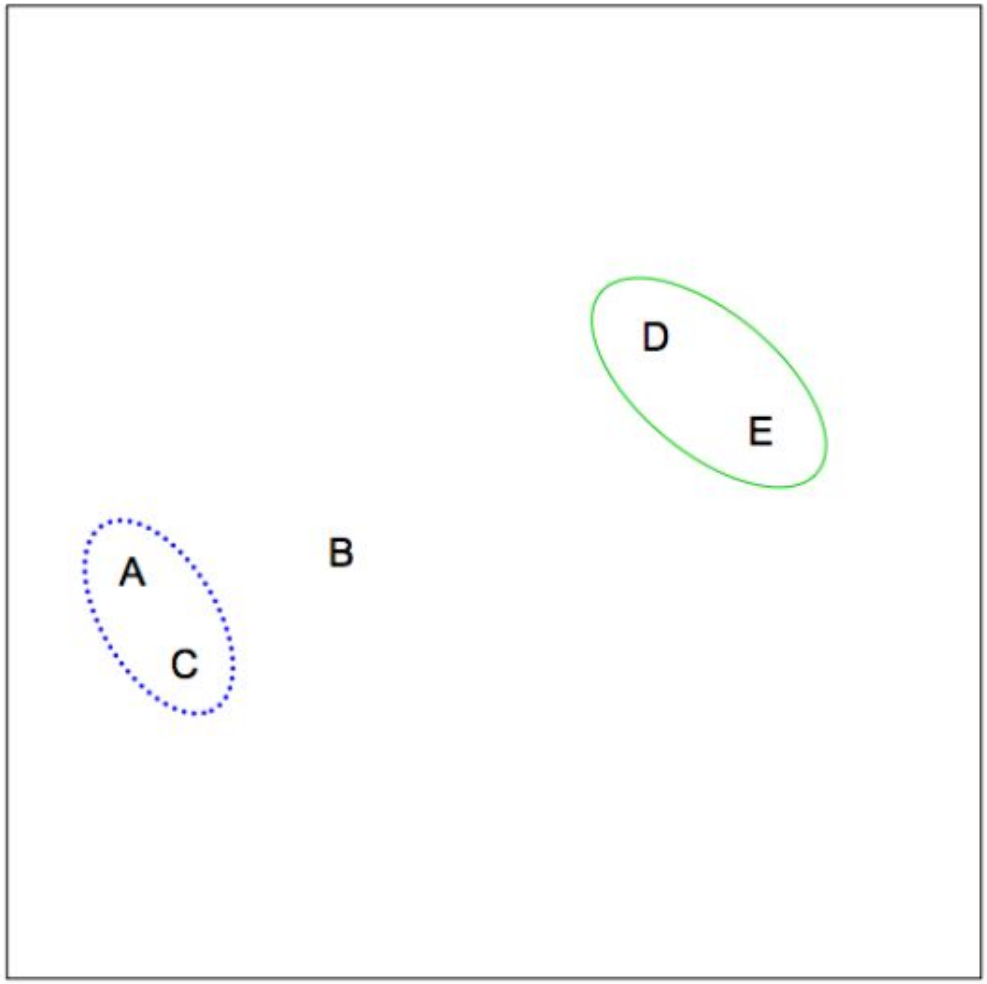
Hierarchical Clustering: “bottom-up” fashion



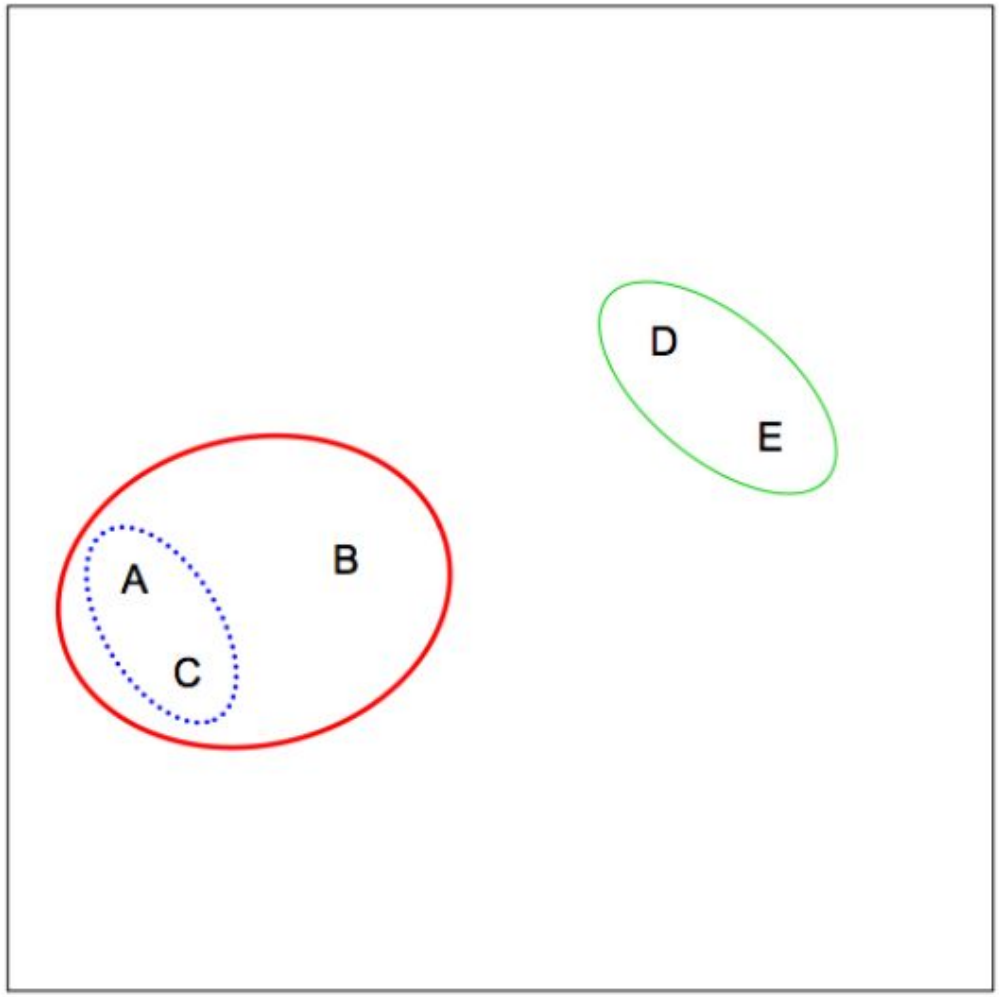
Hierarchical Clustering: “bottom-up” fashion



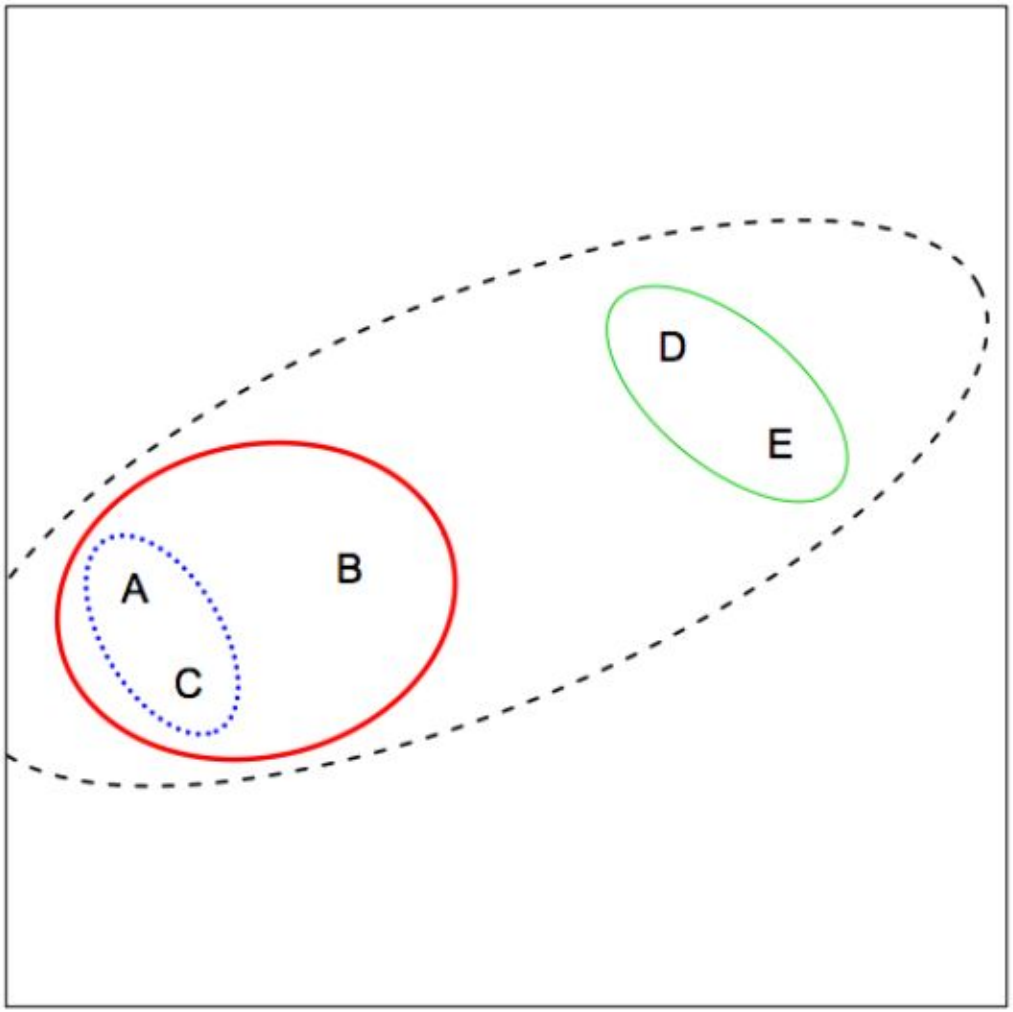
Hierarchical Clustering: “bottom-up” fashion



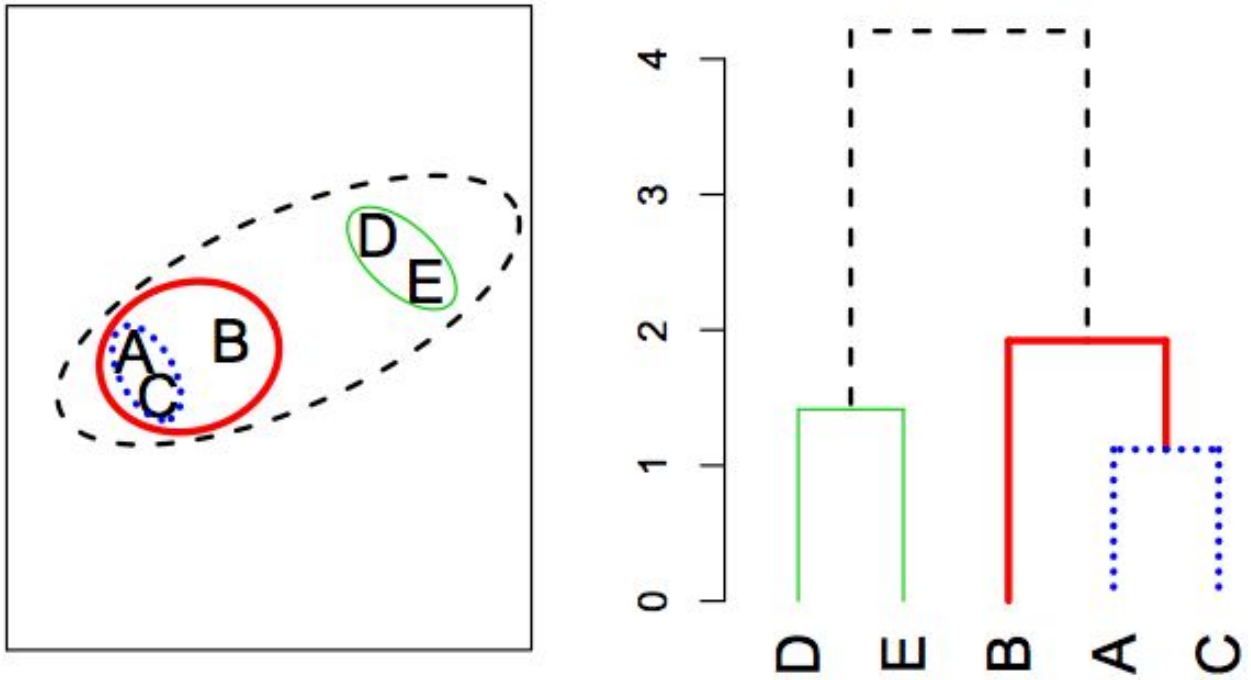
Hierarchical Clustering: “bottom-up” fashion



Hierarchical Clustering: “bottom-up” fashion



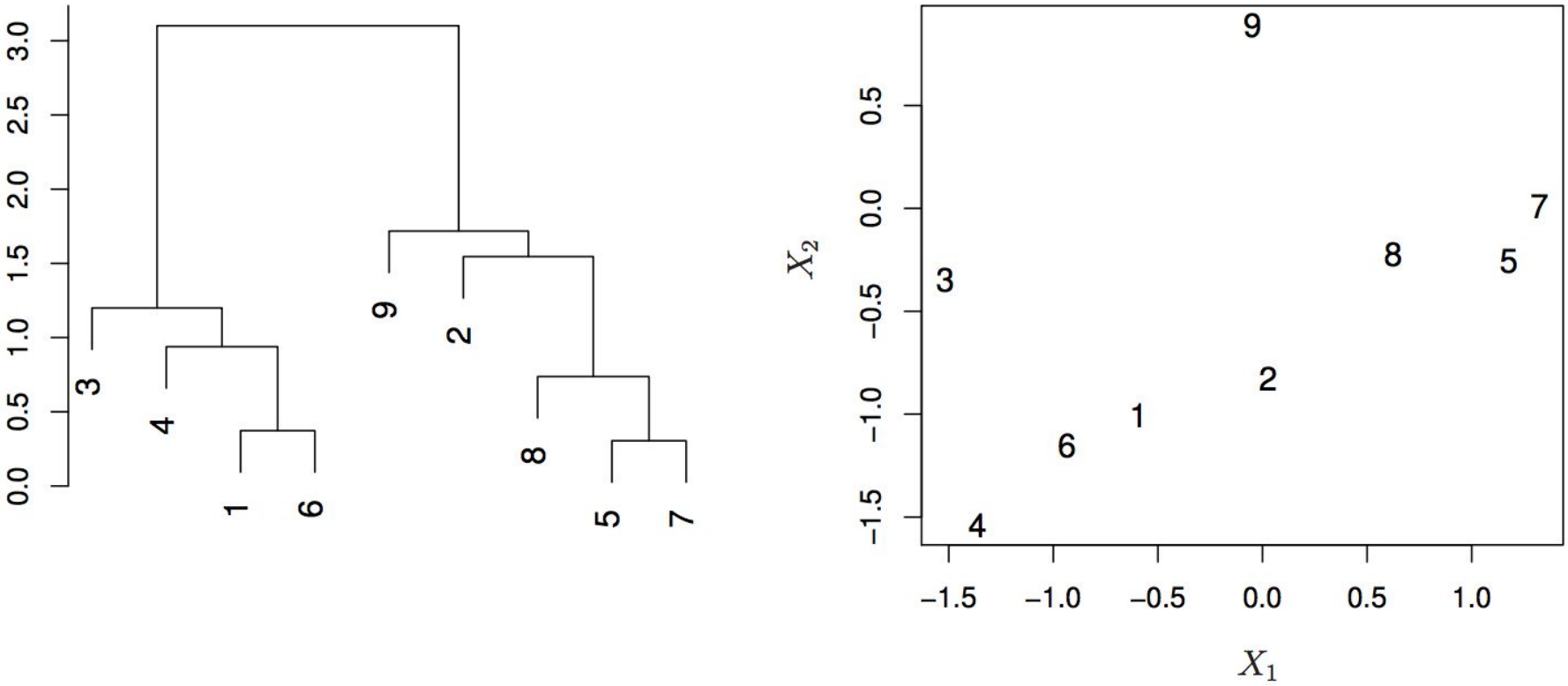
Hierarchical Clustering: “bottom-up” fashion



Interpreting the Dendrogram

- There are some interpretative advantages to visualizing the dendrogram created from hierarchical clustering:
  - The *lower down* in the dendrogram a fusion occurs, the more similar the groups of observations that have been fused are to each other.
  - The *higher up* in the dendrogram a fusion occurs, the more dissimilar the group of observations that have been fused are to each other.
- In general, for any two observations we can inspect the dendrogram and find the point at which the groups that contain those two observations are fused together to get an idea of their dissimilarity.
  - Be careful to consider groups of points in the fusions within the dendrograms, not just individual points.

Interpreting the Dendrogram: Visually



Image

The Hierarchical Clustering Algorithm

1. Begin with n observations and a distance measure of all pairwise dissimilarities. At this step, treat each of the n observations as their own clusters.
2. For $i = n, (n - 1), \ldots, 2$:
   a. Evaluate all pairwise inter-cluster dissimilarities among the i clusters and fuse together the pair of clusters that are the least dissimilar.
   b. Note the dissimilarity between the recently fused cluster pair and mark that as the associated height in the dendrogram.
   c. Repeat the process, calculating the new pairwise inter-cluster dissimilarities among the remaining $(i - 1)$ clusters.

The Hierarchical Clustering Algorithm

- While we do not need to specify K, in order to perform hierarchical clustering there are a few choices we need to make. Particularly:
  - A dissimilarity measure.
  - A linkage method.
- We're already familiar with the idea of choosing a dissimilarity measure with the choice of distance metric. In most cases, it is sufficient to use the Euclidean distance.
- A linkage is a measure of the dissimilarity between group of points. So far we only define the distance between two points, but what do we do when we want to assess the similarity among groups of points?

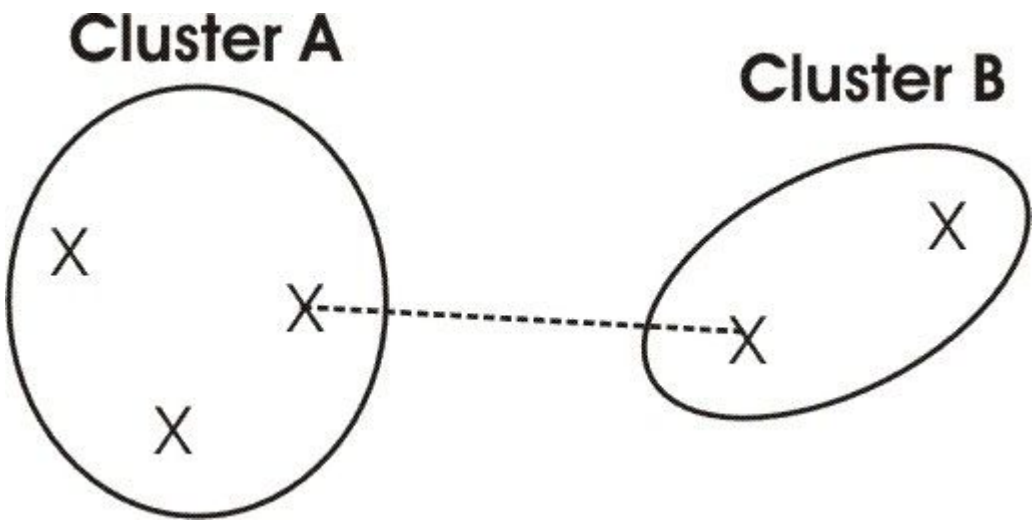
The Hierarchical Clustering Algorithm: Linkage

- The most common types of linkage are described below.
- First, compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B. Then:
  - *Complete Linkage*: Maximal inter-cluster dissimilarity.
    - Record the largest of the dissimilarities listed between A and B as the overall inter-cluster dissimilarity.
  - *Single Linkage*: Minimal inter-cluster dissimilarity.
    - Record the smallest of the dissimilarities listed between A and B as the overall inter-cluster dissimilarity.

The Hierarchical Clustering Algorithm: Linkage

- *Average Linkage*: Mean inter-cluster dissimilarity.
  - Record the average of the dissimilarities listed between A and B as the overall inter-cluster dissimilarity.
- *Ward's Linkage*: Minimum variance method.
  - minimizes the variance of the clusters being merged.
  - In other words, merges the two clusters that lead to the minimum increase of the total within-cluster SSE

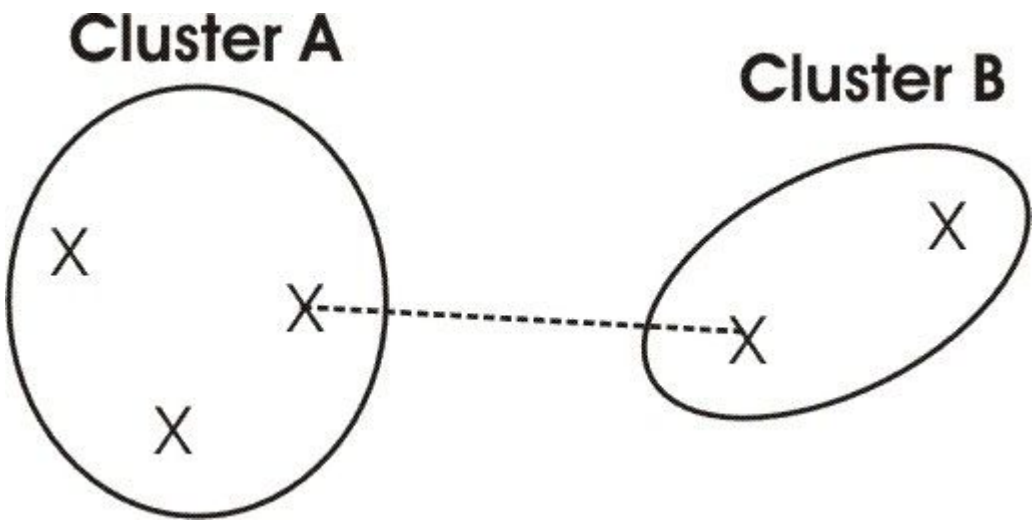
The Hierarchical Clustering Algorithm: Linkage



Image

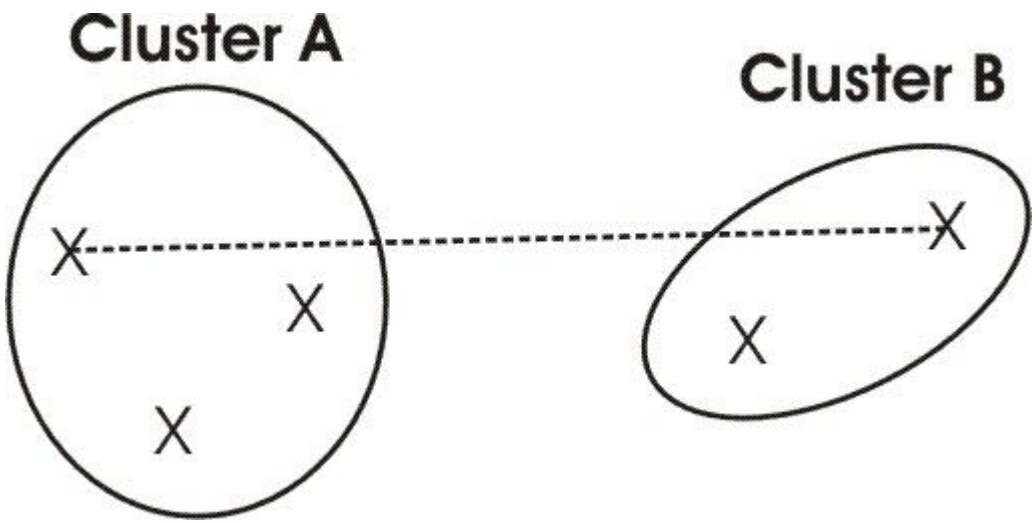
The Hierarchical Clustering Algorithm: Linkage

Single Linkage



Image

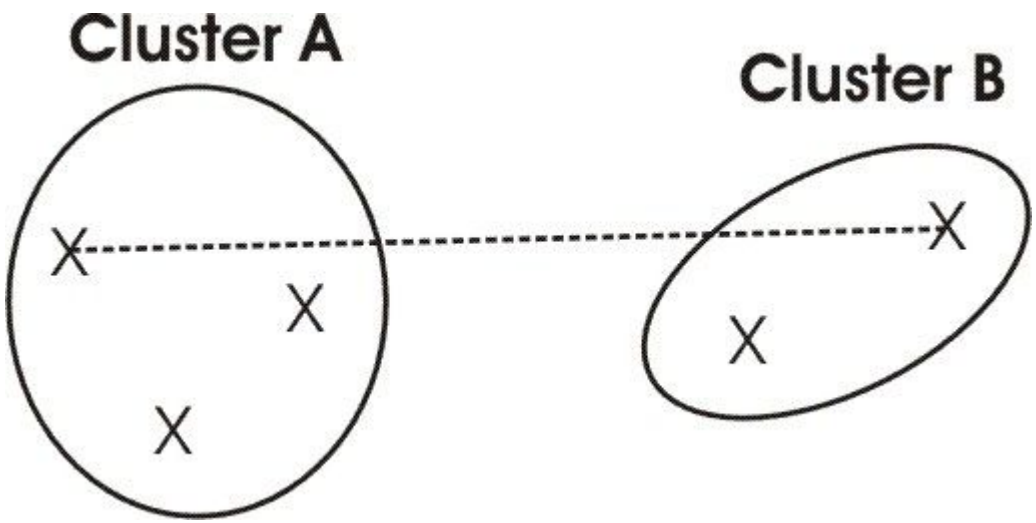
The Hierarchical Clustering Algorithm: Linkage



Image

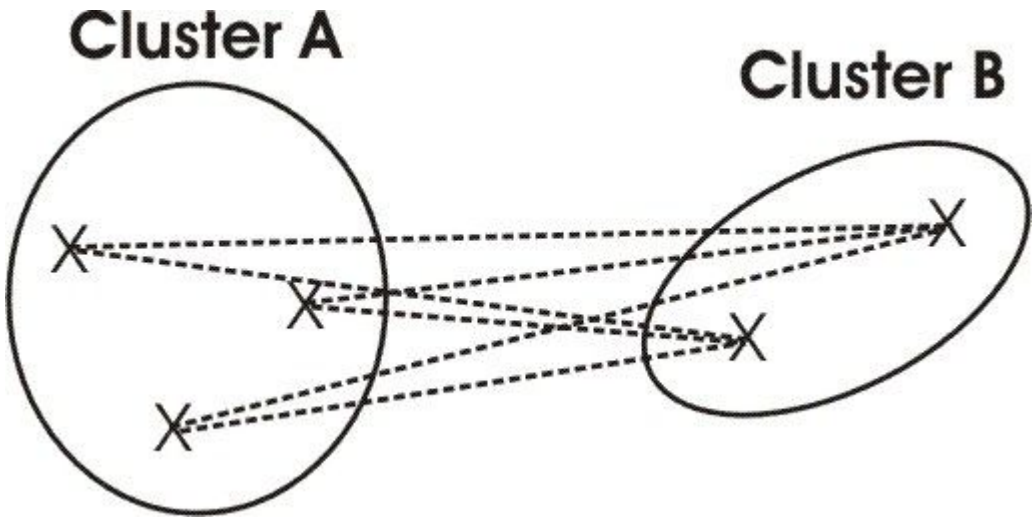
The Hierarchical Clustering Algorithm: Linkage

Complete Linkage



Image

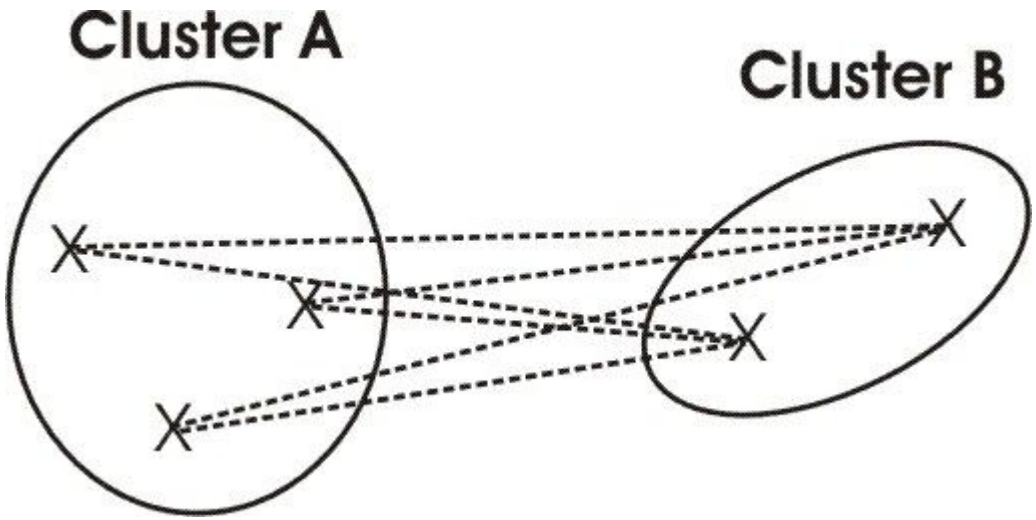
The Hierarchical Clustering Algorithm: Linkage



Image

The Hierarchical Clustering Algorithm: Linkage

Average Linkage



Image

The Hierarchical Clustering Algorithm: Linkage

- *Complete linkage* is sensitive to outliers, yet tends to identify clusters that are compact, somewhat spherical objects with relatively equivalent diameters.
- *Single linkage* is not as sensitive to outliers, yet tends to identify clusters that have a chaining effect; these clusters often fail to represent intuitive groupings among our data, and many pairs of observations might be quite distant from one another.
- *Average linkage* tends to strike a balance between the pros and cons of complete linkage and single linkage.

Hands-on Session

- Please go to the **"Hierarchical Clustering in Scikit Learn"** in the lecture code.