NYC DATA SCIENCE ACADEMY

Python Machine Learning Class 9: Unsupervised Learning

NYC 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 infer the properties directly 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*, **PCA**, 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 distinct subsets known as “clusters”.

Unsupervised Learning

- Unlabeled data is easier to obtain than labeled data.
- No specific prediction goals, therefore more subjective.
- We are usually interested in discovering the hidden pattern of the data.
- Examples:
  - Groups of online shoppers characterized by their browsing and purchase histories.
  - Movies grouped by the comments given by movie viewers.

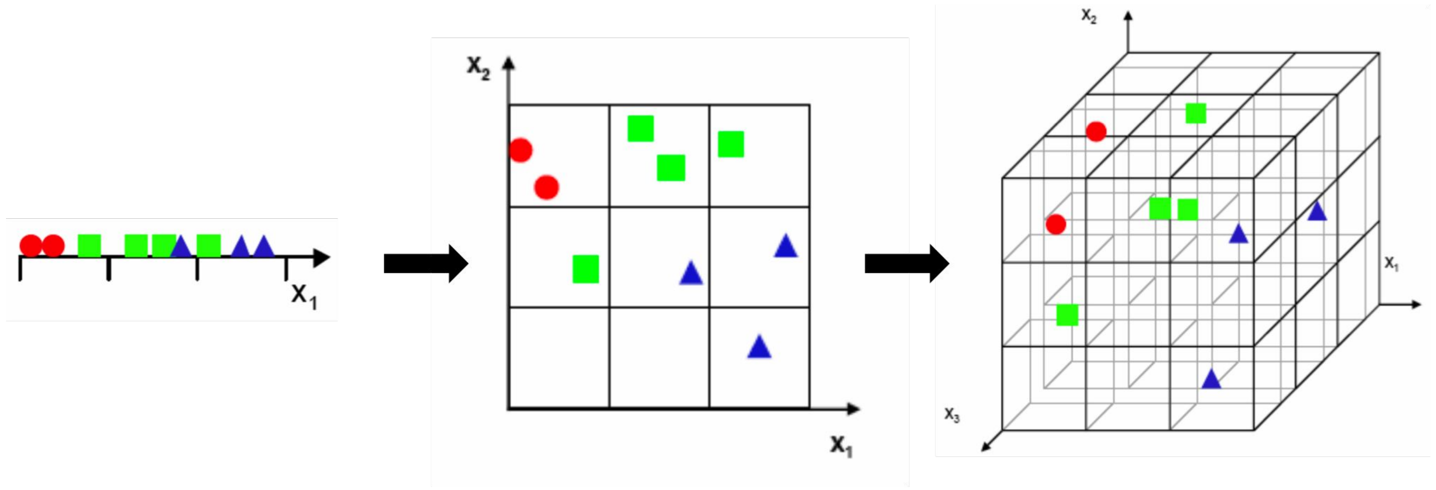
Outlines

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 predicted from the others through linear formulae with a substantial degree of accuracy.
- Issues:
  - The regression coefficients of highly correlated variables might be inaccurate (high model variance).
  - The estimate of one variable's impact on the dependent variable *Y* while controlling for the others tends to be less precise.
  - The nearly collinear variables contain similar information about the dependent variable, which may lead to overfitting (why?).
  - 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 one another.
- Sparsity becomes exponentially worse as the dimensionality of the data increases.
- The model **SVM** takes advantage of the **curse of dimensionality**.



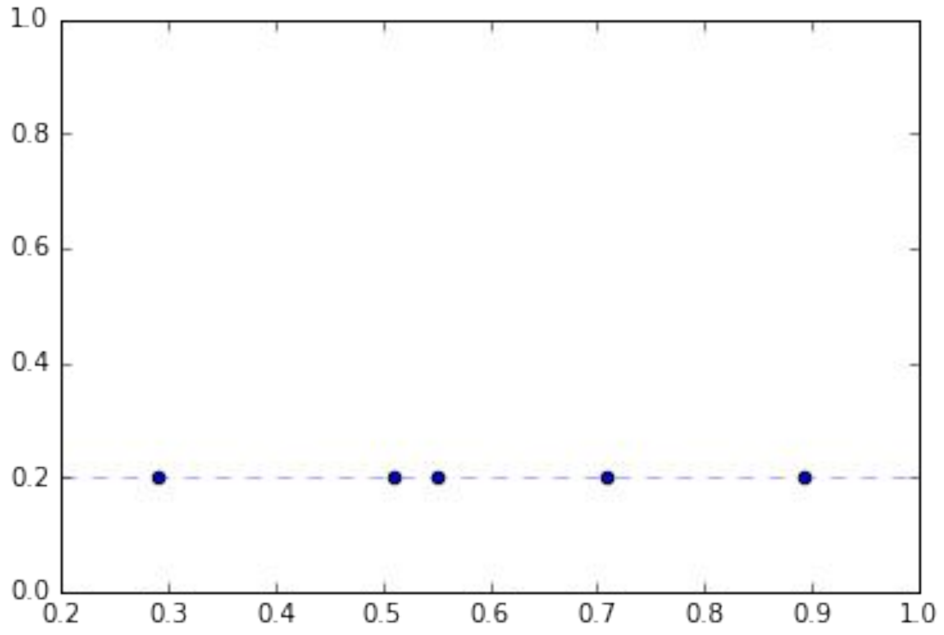
Principal Component Analysis

- Ideal input variables:
  - linearly uncorrelated,
  - low-dimensional in the feature space.
- ***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

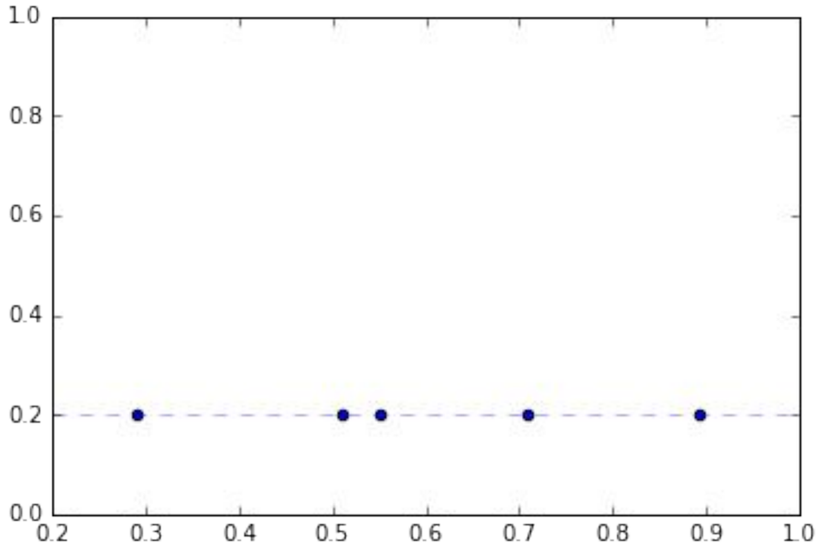
Motivation: Toy Example

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



Motivation: Toy Example

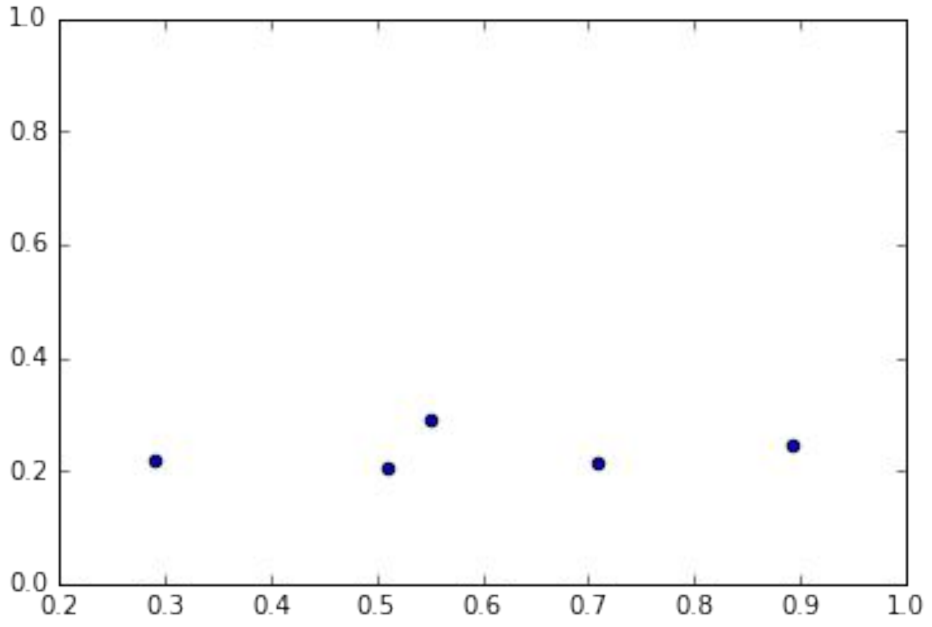
- Each observation’s coordinates have two components. Do we really need two?



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

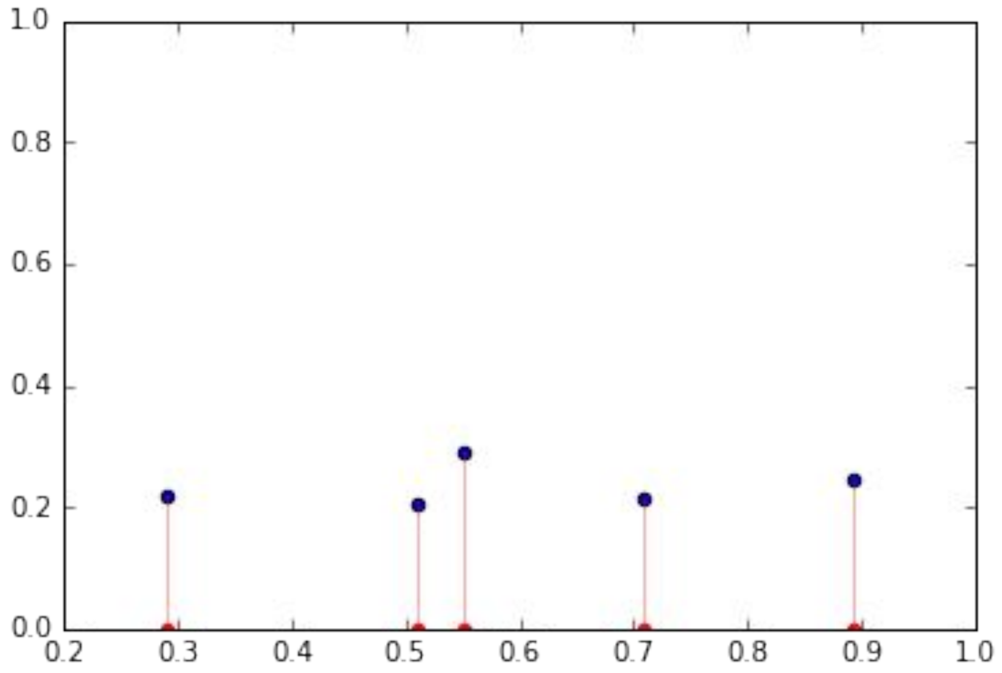
Motivation: Toy Example

- 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?



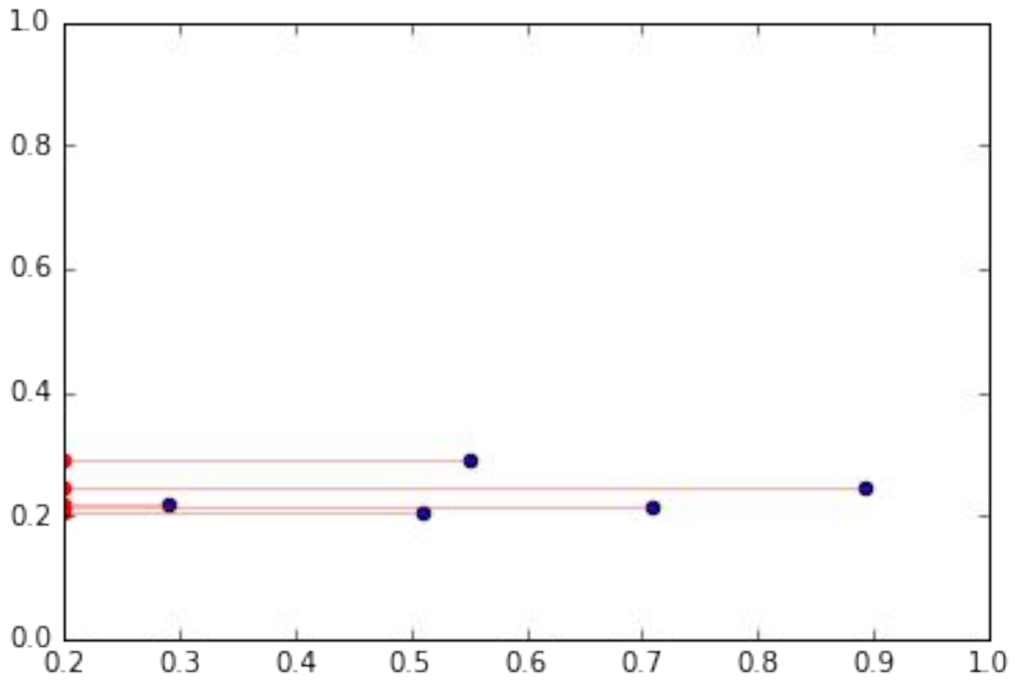
Motivation: Toy Example

- One way to compare is to project the observations to the two coordinate axes.
  - For x axis, the projected points spread in a range around 0.3 to 0.9.



Motivation: Toy Example

- In contrast, y values are 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

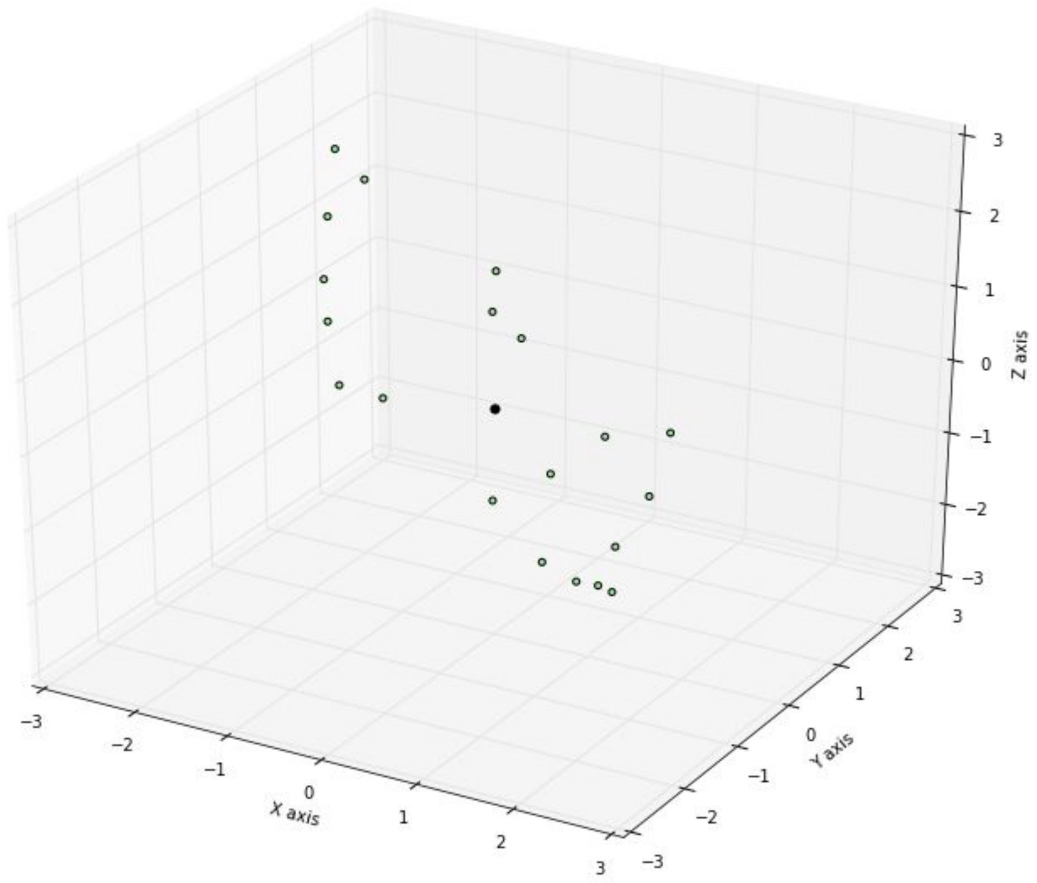
- This motivates the idea of **PCA**. We are searching for, among all the possible directions in the feature space, the direction along which the projection of the observations are most widely spread.
- We emphasize that the motivation discussed above can be a little misleading. Even in the two dimensional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

The First Loading Vector

- Consider a set of 20 points in a three dimensional space. In such a scenario, we have:
  - 20 observations.
  - 3 features.

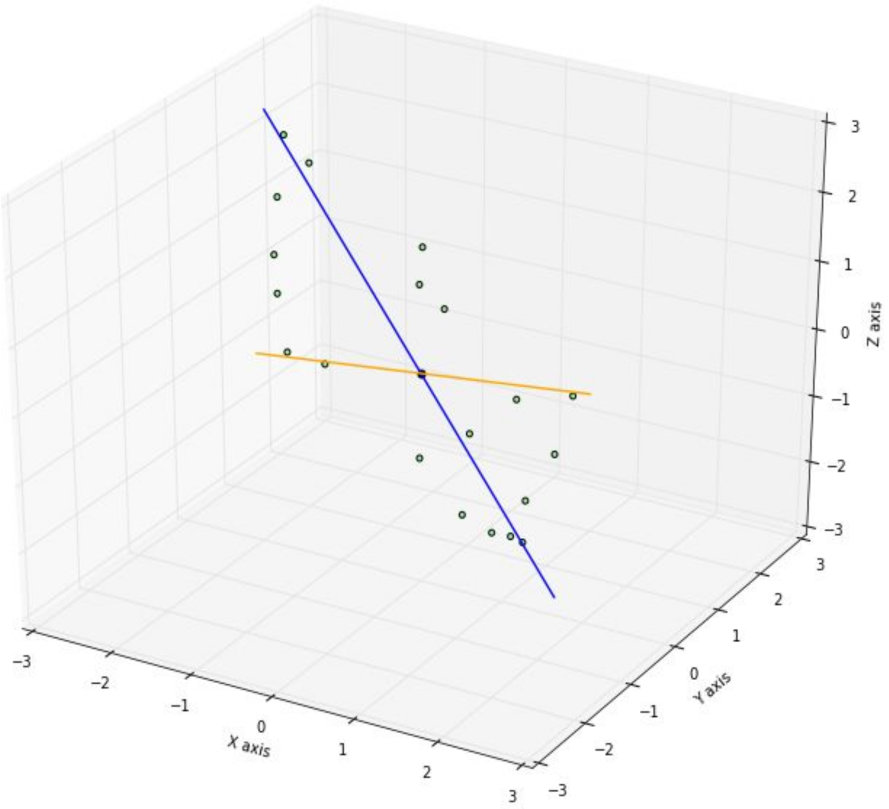
The First Loading Vector

- We visualize the data below. The **black dot** in the middle represents the origin.



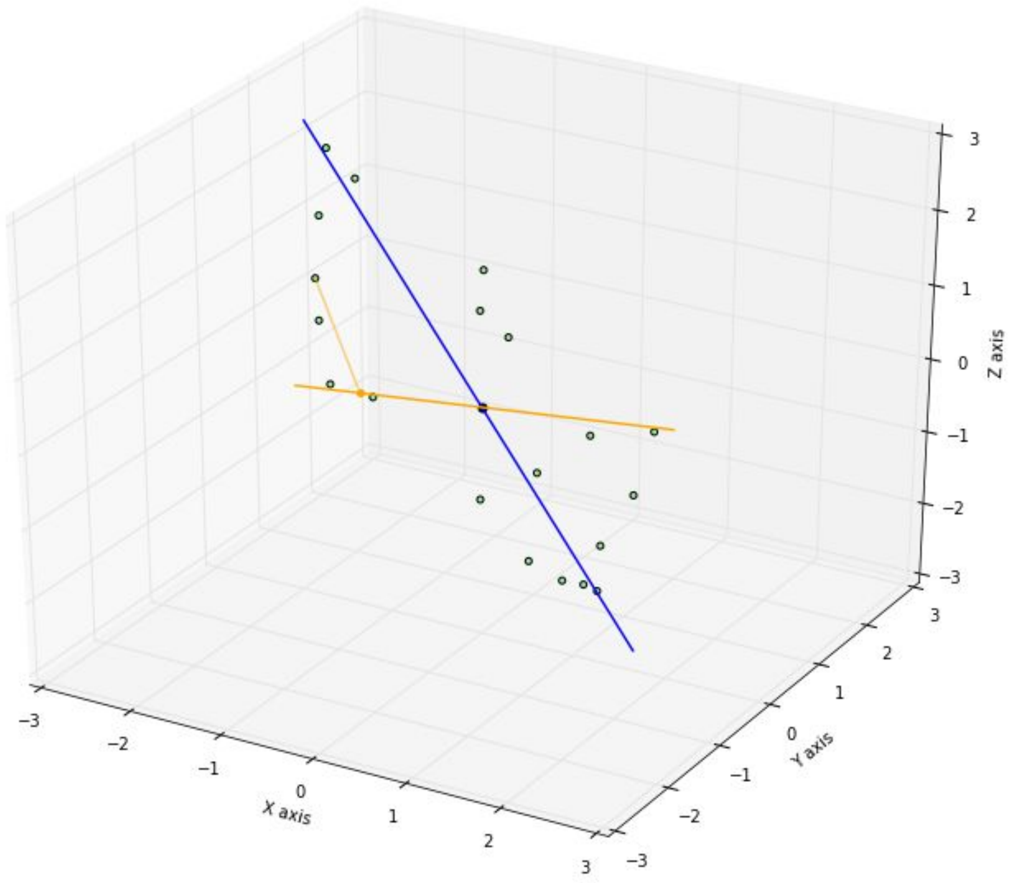
The First Loading Vector

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



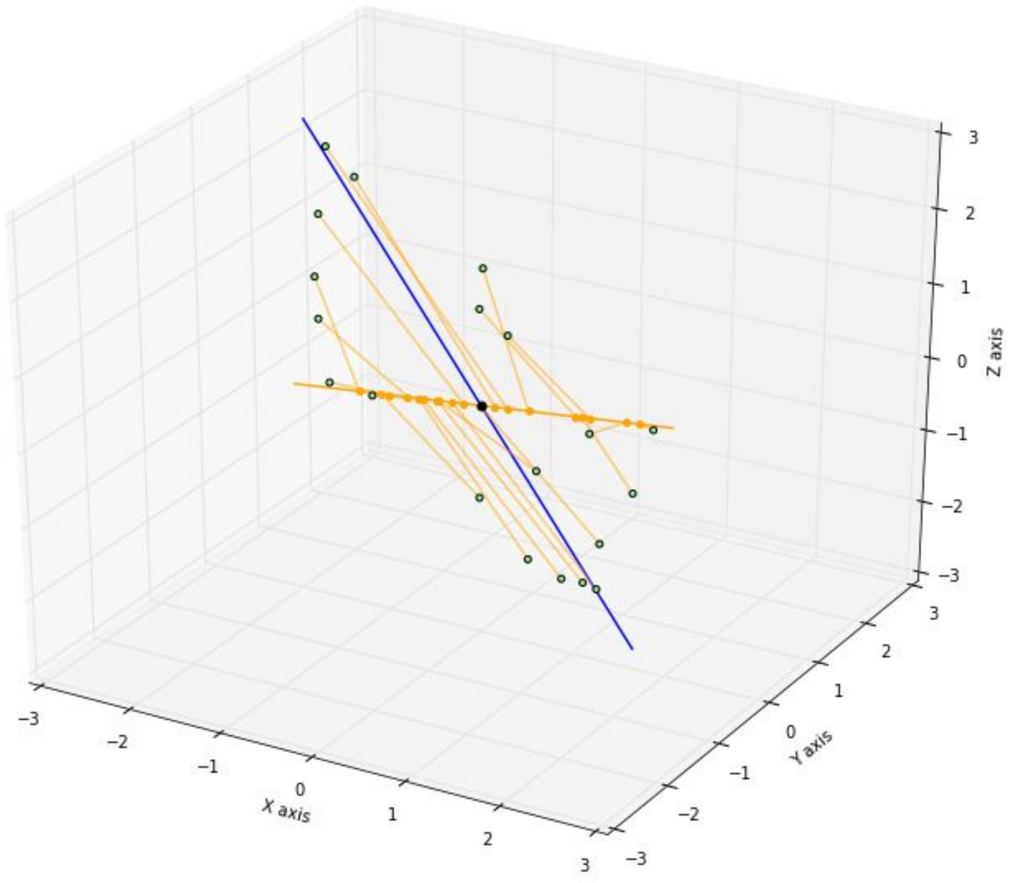
The First Loading Vector

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



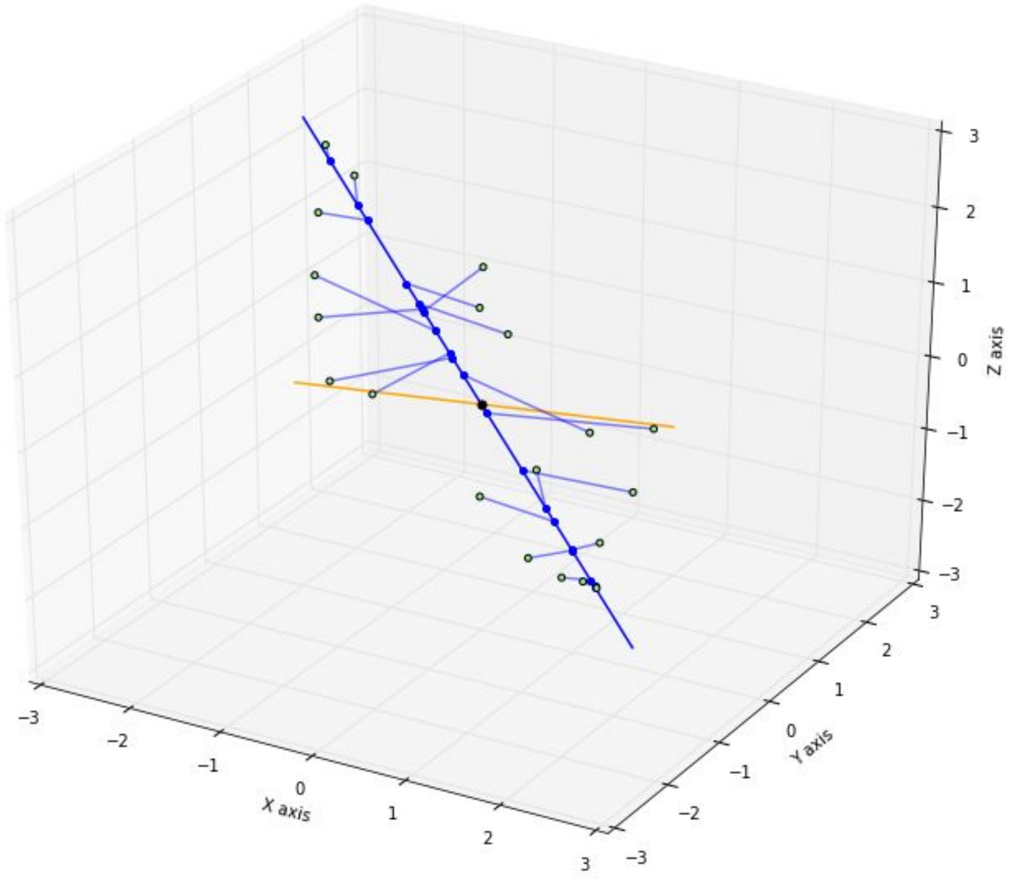
The First Loading Vector

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



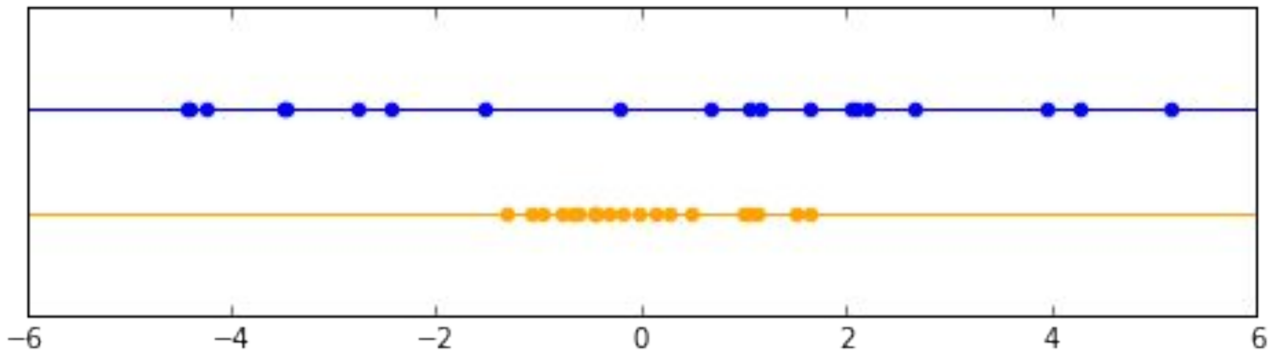
The First Loading Vector

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



The First Loading Vector

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



The First Loading Vector

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

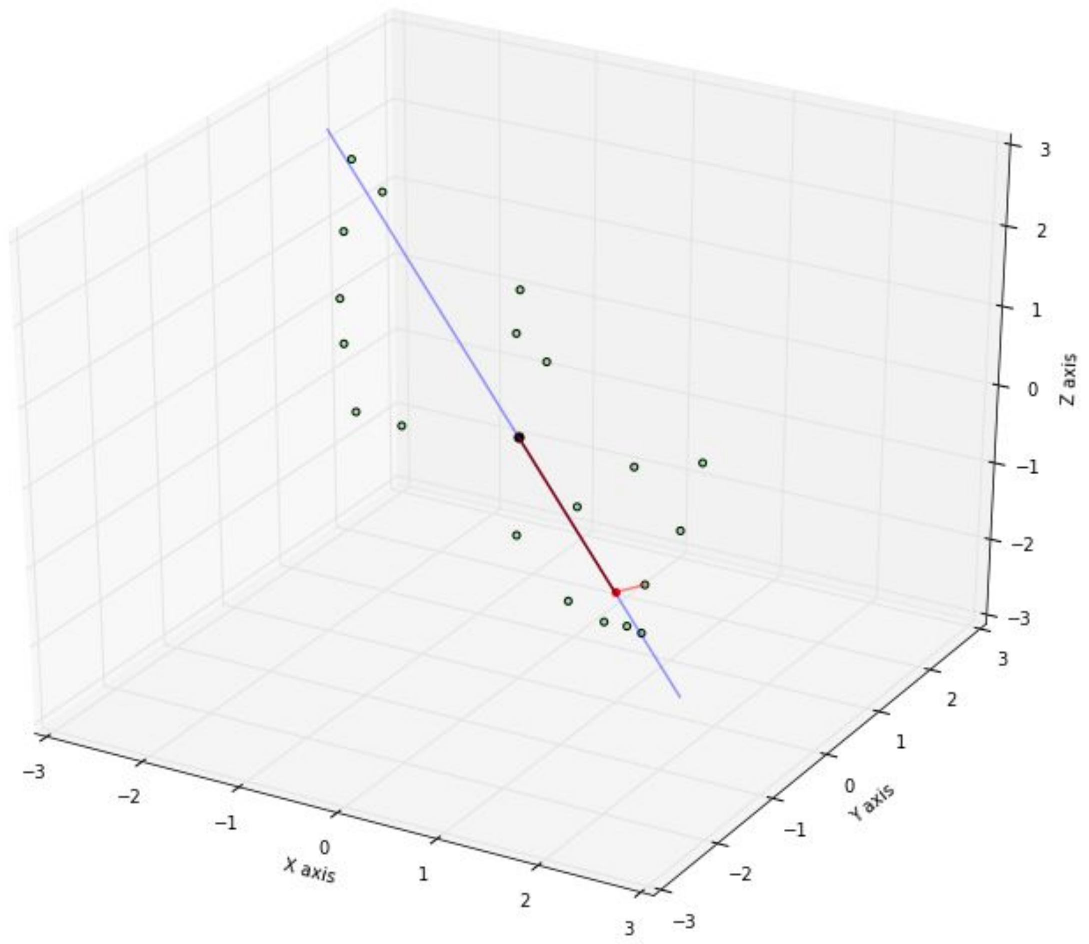
- The statements above characterize the **principal direction**. To find the **principal direction** we need to apply the technique of linear algebra, which we will discuss briefly. However, if you don't care about math, Python will find it for you.

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 **orthogonal linear projection**.
  - There are in general N (the number of samples) components for a **principal component**.
  - There are in general p (the number of features) components for a **principal direction** (the loading vector)
  - The **principal components** live in the space of samples, while the **principal directions** live in the space of features.

The First Principal Component

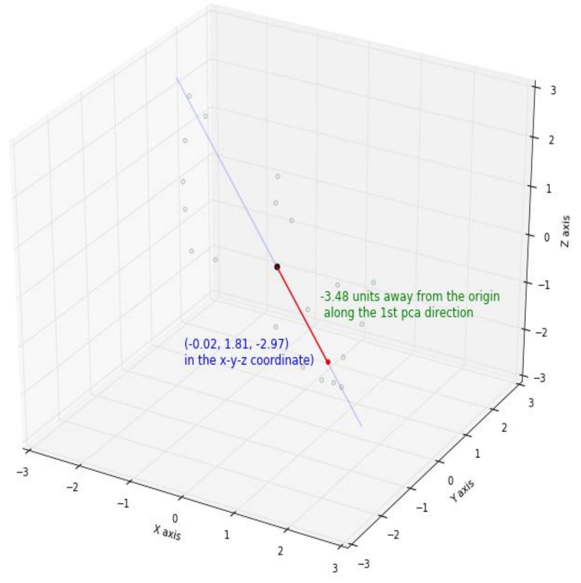


The First Principal Component

- 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 single coordinate is supposed to provide the most information.

The First Principal Component

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

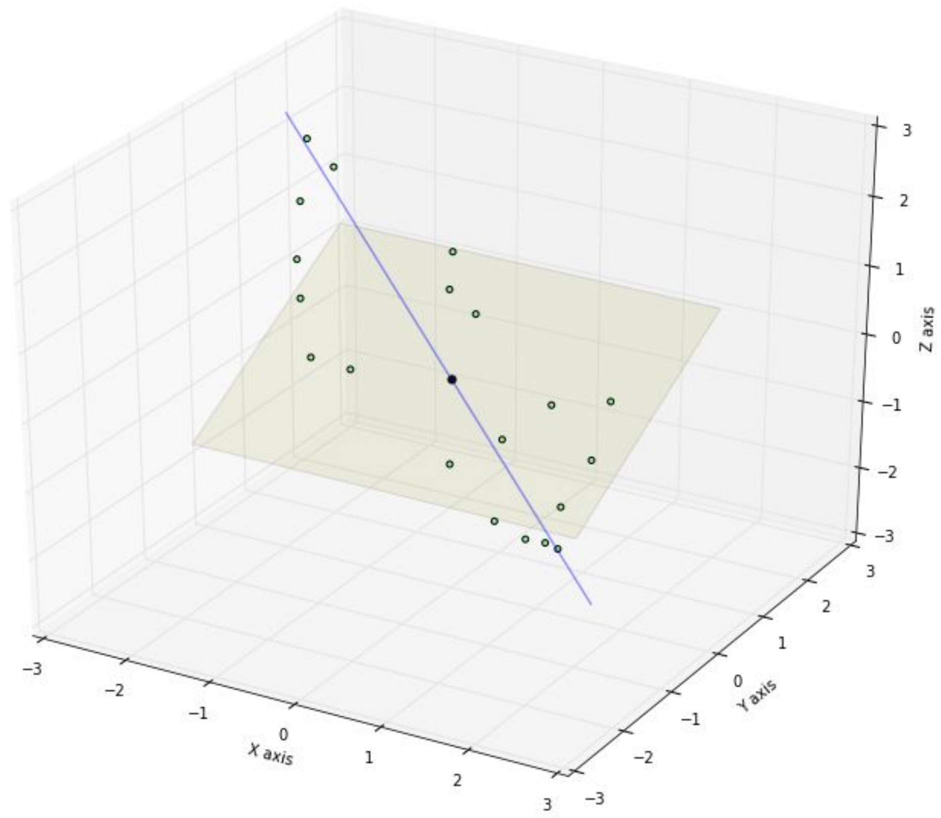


The Second Principal Component

- ❖ The information stored in a data set is about the variation of the points across the whole sample set
- ❖ Not all the directions are born equal!
- ❖ The first principal component provides the most information, but most likely it is NOT all. To find the next most significant direction, we need to:
  - ➢ First, remove the data information stored in the first principal component.
  - ➢ Then we again find the new direction (orthogonal to the original principal direction) on which the projection of the observations is most widely spread.

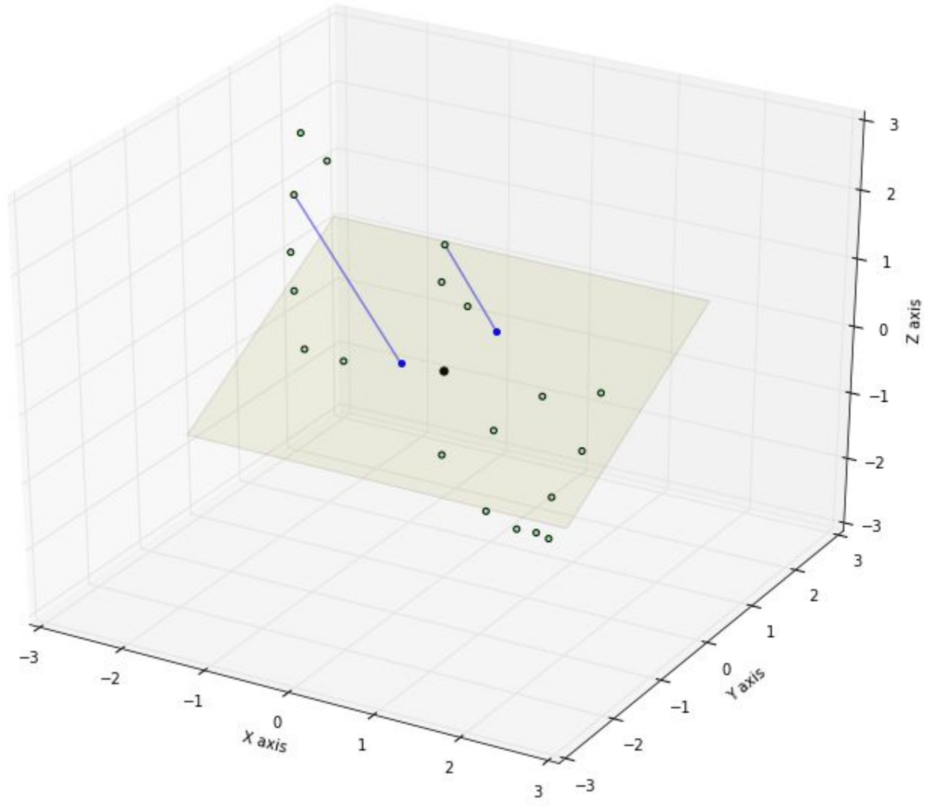
The Second Principal Component

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



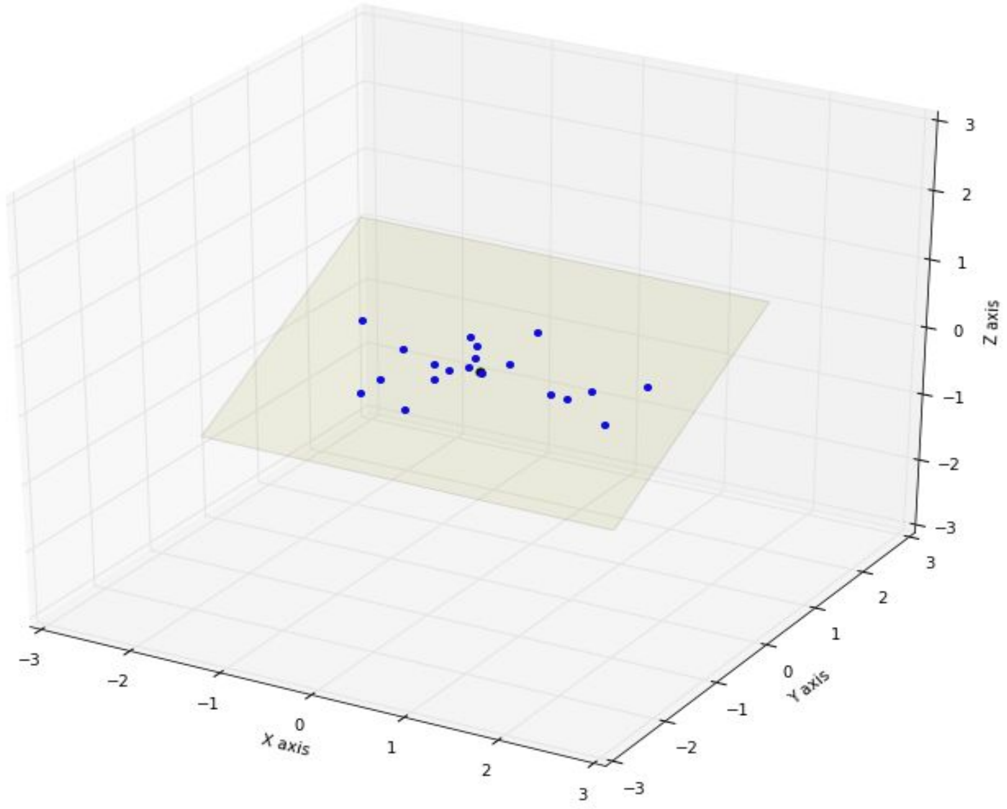
The Second Principal Component

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



The Second Principal Component

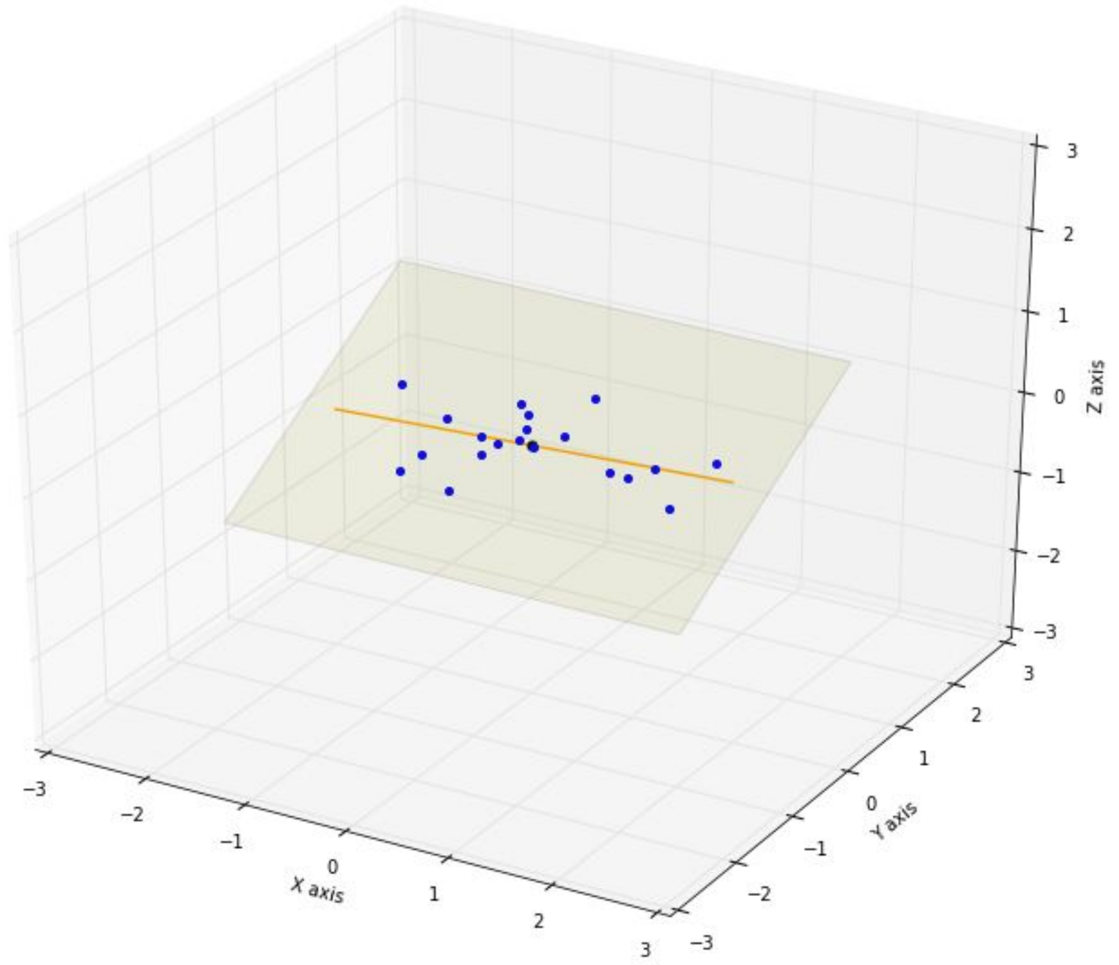
- We can remove the effect of the first principal component by projecting the observations to this plane. The blue points below are the images of the projection:



The Second Principal Component

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

The Second Principal Component

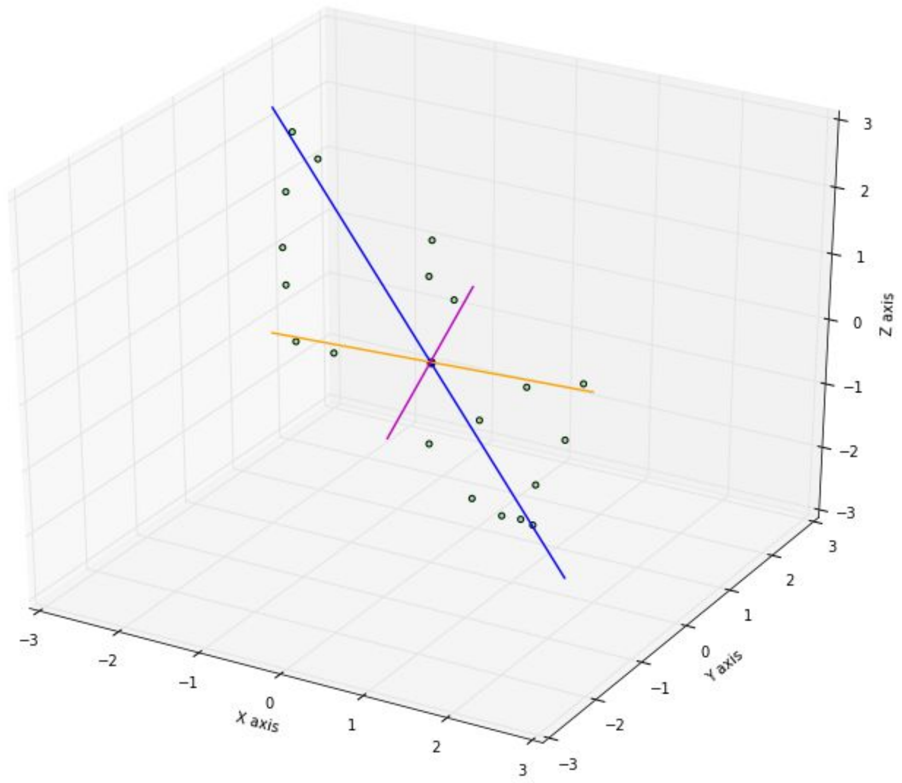


The Second Principal Component

- **Remark** Clearly this process can be continued to retain more and more information from the raw data. However, as we can see from the process above, each time we remove the information recorded in a principal direction, we remove one dimension. So we can't have more number of the principal components than the number of the original features we have.
- This induction process always terminates within a finite step.

The Second Principal Component

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



Outlines

- Introduction 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 formulae. 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 raw data. We may then assume our data X is an n by p matrix (that means we have a data set of n observations and p features). The average of each feature 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**

- ➢ How many components does $\hat{u}$ have?
- ➢ How many components does $X\hat{u}^{\mathrm{T}}$ have?
- ➢ Why do these dimensions make sense? Or not?

The Mathematical Formulation

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

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

- The solution to the optimization problem above is the **first loading vector (first principal direction)**, denoted by ϕ_1The projection of our data X on the first loading vector is

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

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

The Mathematical Formulation

- Once the first k-1 principal components have been found, the next one (if there is one) can be found inductively.
  - We first remove the information stored in the first k-1 components from X (X_k denotes the resulting matrix).

$$\mathbf{X}_k = \mathbf{X} - \sum_{i=1}^{k-1} \mathbf{X}\phi_i^T \phi_i$$

  - 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 often be hard. In the setting of **PCA**, this is relatively easy. The **principal directions** (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 to be smaller among the two, so there are p of them).
- The variance of each principal component decreases:

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

- 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.
- The variances of the data along the **principal directions** (**eigenvectors**) are the corresponding **positive** eigenvalues.

Geometric Meaning of PCA

- Geometrically we can imagine that the original data set sits inside $\mathbf{R}^f$, as a high dimensional scatterplot.
- The selection of the top p principal directions establishes a linear projection $\mathbf{R}^f \longrightarrow \mathbf{R}^p$ into a lower dimensional space.
- There are many orthogonal linear projections from $\mathbf{R}^f$ to

$\mathbf{R}^p$. But PCA is special is that it collapses directions in which the variances are small and preserve those whose variances are larger.

- Those directions which get collapsed are interpreted as noises of the data.

Manifold Learning

- Even though the apparent dimension of the data is f dimensional, PCA hypothesizes that the true dimension of the data lies in a p dimensional linear space.
- In this sense, PCA is a de-noising process revealing the true nature of the data.

Why do we stop at linear objects?

Nonlinear projections into curved objects (e.g. spheres) instead of linear spaces has been a hot research area in machine/deep learning.

- This is known as **manifold** learning, as nonlinear smooth objects are called manifolds in geometry.

Hands-on Session

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

Outlines

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 label.
  - Does aim to uncover **underlying structure** of the data and see what pattern exists 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 smaller subgroups that can further segment current subgroups?
  - Are there any outlying observations?
- Notice that these questions are largely exploratory in nature.

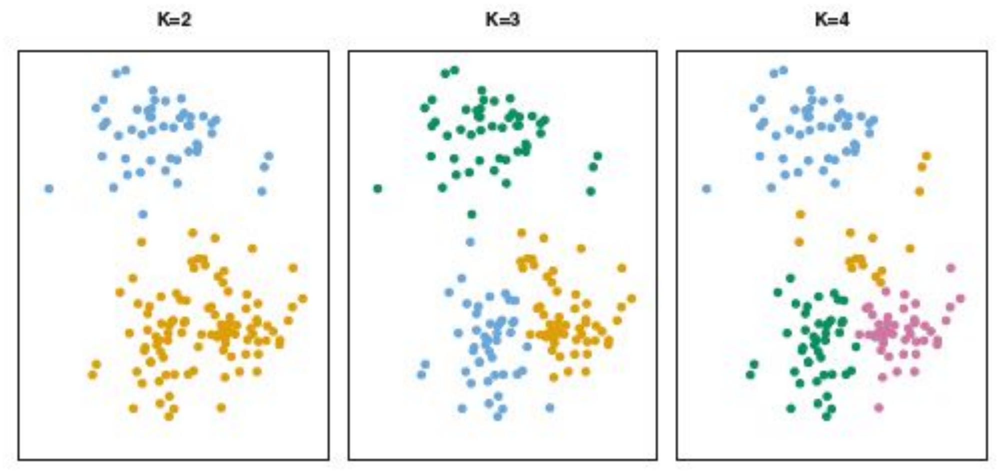
Outlines

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 cluster memberships are distinct and non-overlapping.
- The data points in each of the clusters are determined to be mostly similar to a specific centroid value:
  - The centroid of a cluster represents the average of the observations within a given cluster; it is a single theoretical center that represents the prototypical member that exists within the given cluster.
  - Each observation will be assigned to exactly one of the K clusters depending on where the observation falls in the feature space relative to the cluster centroid locations.

K-Means Clustering

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



- The color labels the cluster to which it has been assigned. Note that the cluster coloring is arbitrary since there is no absolute ordering of the clusters.

K-Means Clustering

- Now the main question: what technique does k-means algorithm use to create these clusters?
- Suppose we use the Euclidean distance. 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/features in our dataset.

K-Means Clustering

- Since the within-cluster variation is a quantitative gauge of the amount by which the observations in a specific cluster differ from one another, we want to minimize the sum of 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 would like to partition the observations into K clusters such that the total within-cluster variation aggregated across all K clusters is as small as possible; the optimization problem for K-means is as follows:

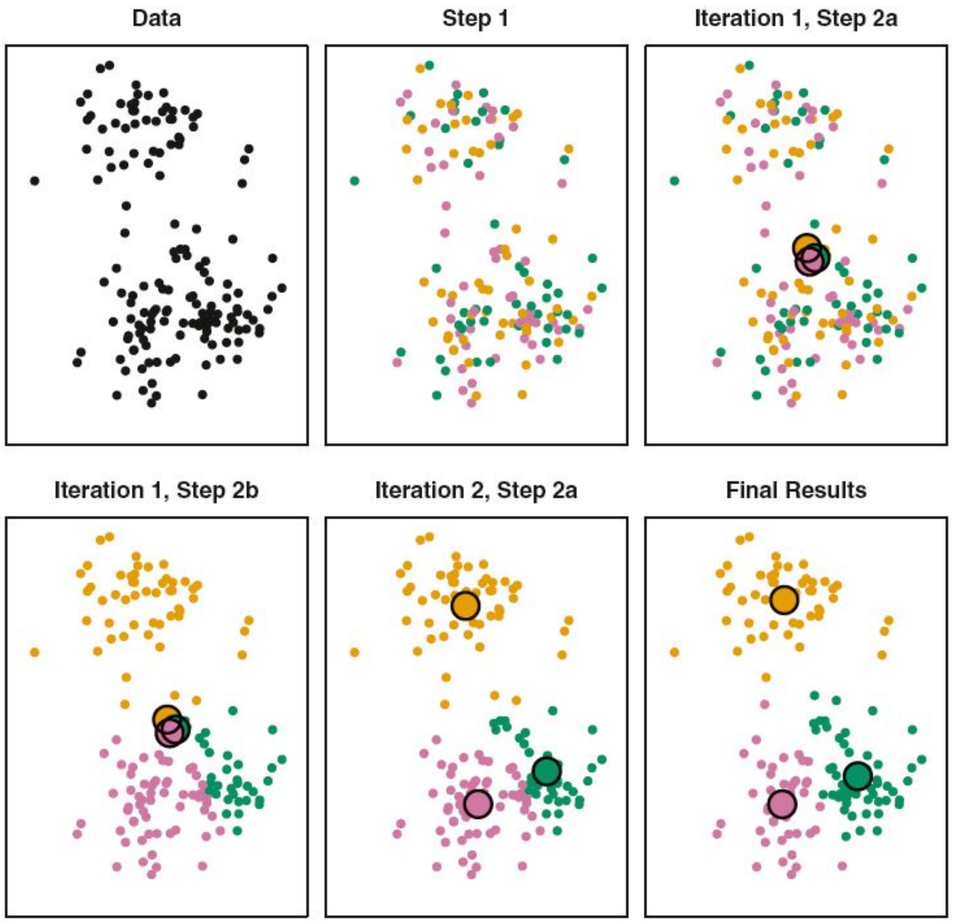
$$\min_{C_1,\ldots,C_K}\left\{\sum_{k=1}^{K}\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2\right\}$$

K-Means Clustering

- To find the global minimum of the above function is very difficult (why?).
- In practice, most kmeans packages perform the following greedy algorithm, also known as **Lloyd** algorithm in the computer science circle:
  1. Randomly assign an integer label, 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 new centroid.
     - ii. Assign each observation to the cluster whose centroid is closest (closest is measured using Euclidean distance).

K-Means Clustering

❖ The algorithm in action (cited from [ISLR)](ISLR):



K-Means Clustering

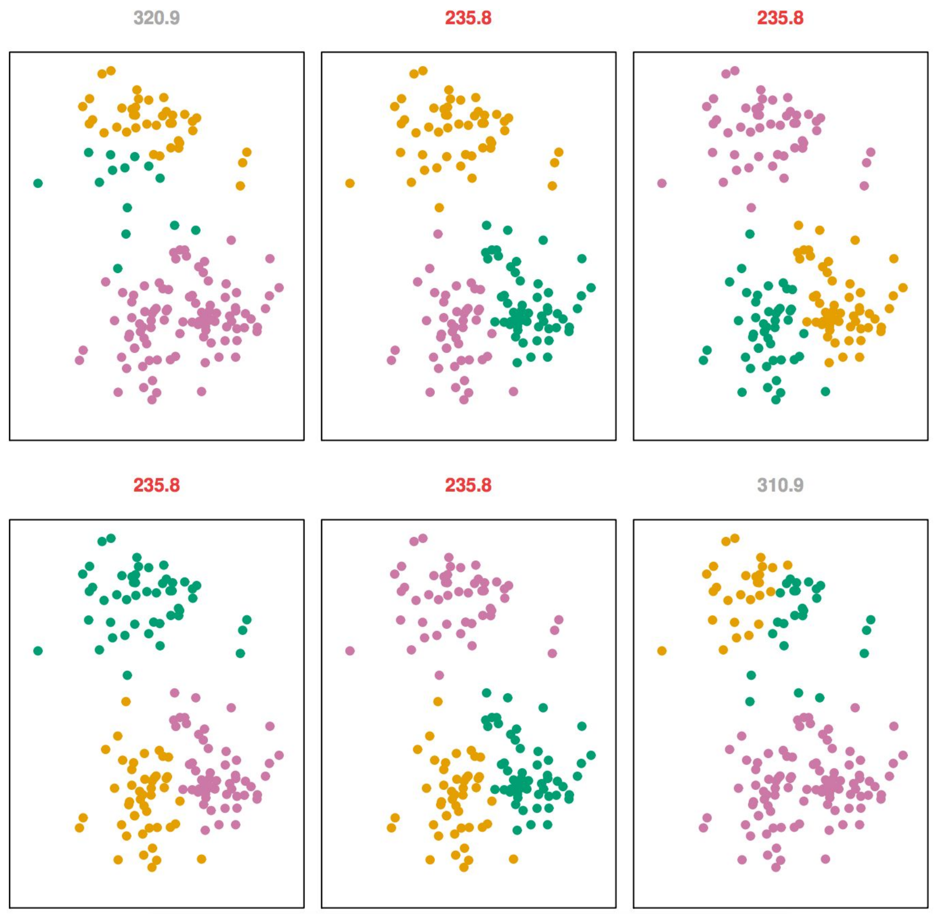
- The K-means procedure always converges:
  - If you run the algorithm from a fixed initial assignment, it will reach a stable endpoint where the clustering solution will no longer change through the iterations.
- Unfortunately, the guaranteed convergence is to a local minimum.
  - Thus, if we begin the K-means algorithm with a different initial configuration, it is possible that convergence will find different centroids and therefore ultimately assigning different cluster memberships.
- What can 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.

Initial Seeding on KMeans

- The KMeans++ (2007) improves the random seeding of the original KMeans
  - The initialization step runs inductively.
  - Firstly, pick a data point randomly as the first centroid
  - Suppose that k of the seed centroid have been chosen, compute for each data point x the distance D(x) to the closest centroid among these k seed centroids. Select the (k+1)-th centroid randomly, according to a probability distribution with probability proportional to D(x)^2.
  - In each inductive step, the newly found seed centroid tends to keep a far distance from the existing ones.
- The default initialization scheme of Scikit-Learn's KMeans uses kmeans++.

K-Means Clustering

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



(cited from [ISLR)]()

Hands-on Session

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

Outlines

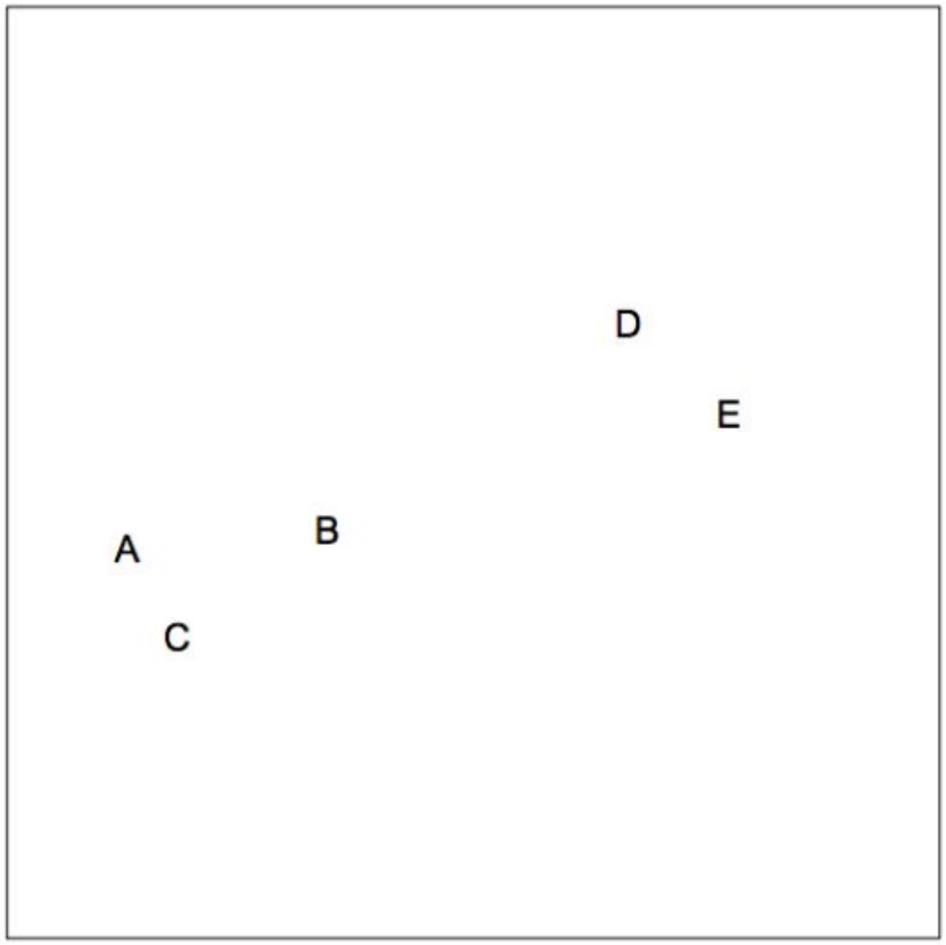
Hierarchical Clustering (Agglomerative Clustering)

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

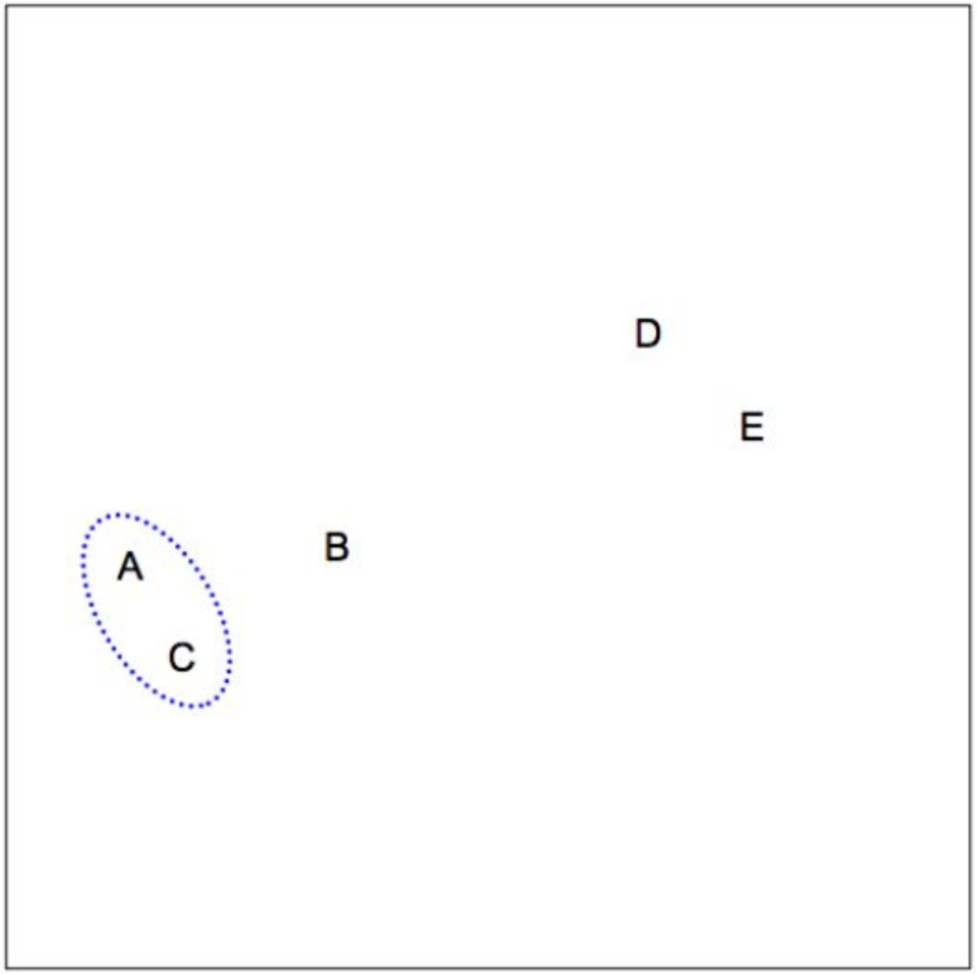
Hierarchical Clustering

- Two strategies for hierarchical clustering: *bottom-up* and *top-down*.
- 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 two clusters which are most similar and merge them.
  3. Repeat step 2.
  4. Ends when all data points are in a single cluster.

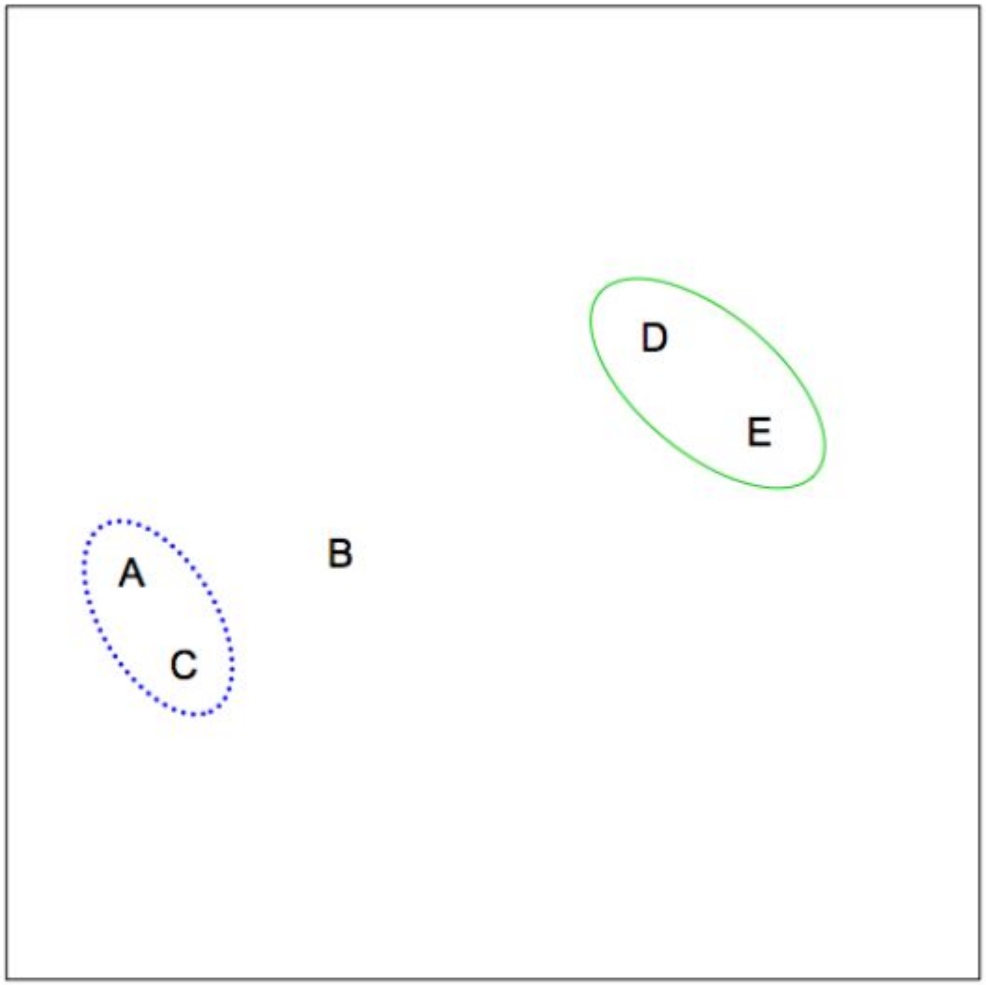
Hierarchical Clustering: “bottom-up” fashion



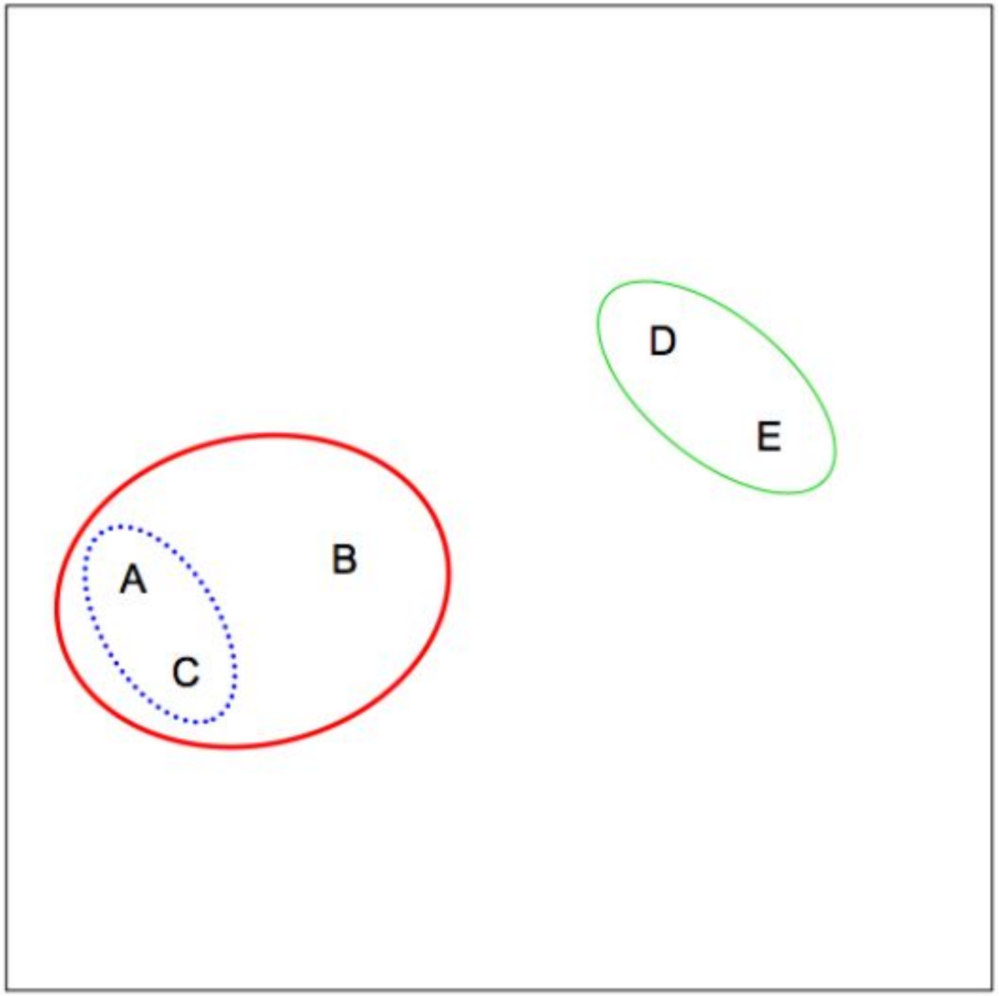
Hierarchical Clustering: “bottom-up” fashion



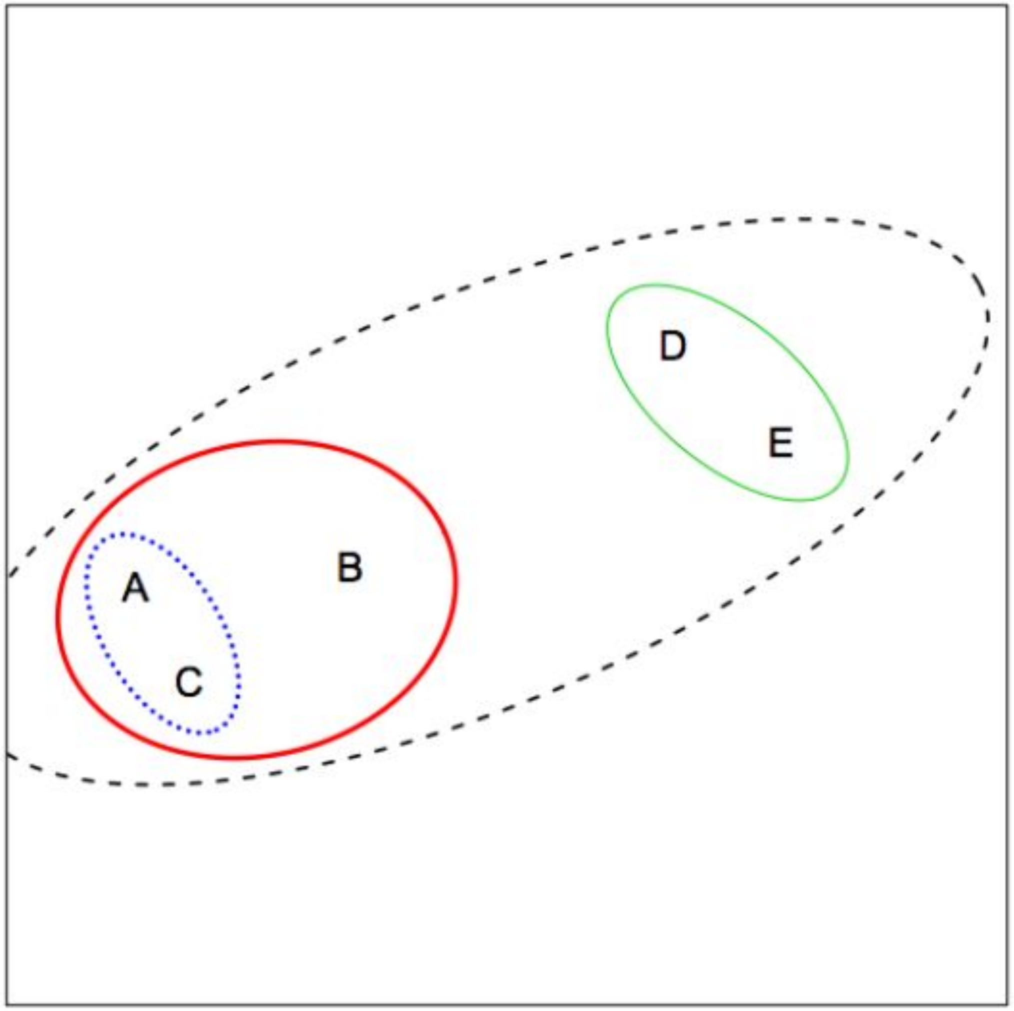
Hierarchical Clustering: “bottom-up” fashion



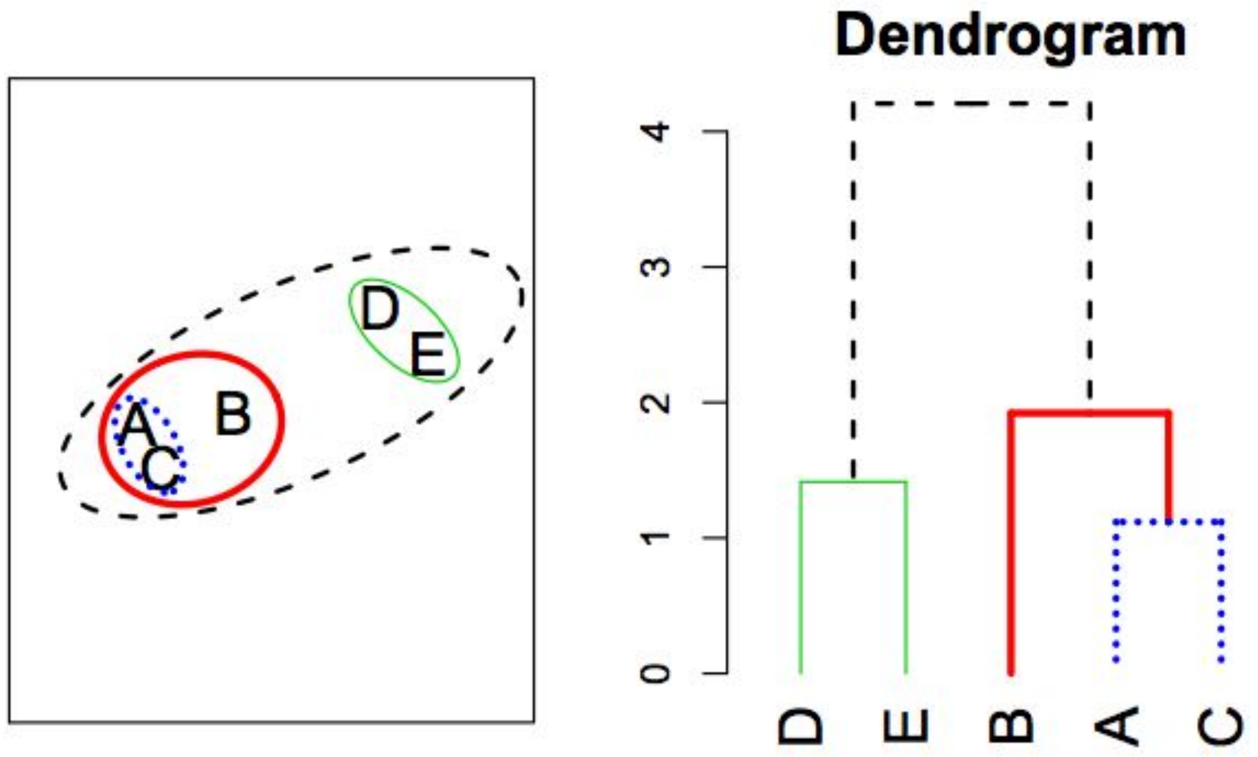
Hierarchical Clustering: “bottom-up” fashion



Hierarchical Clustering: “bottom-up” fashion



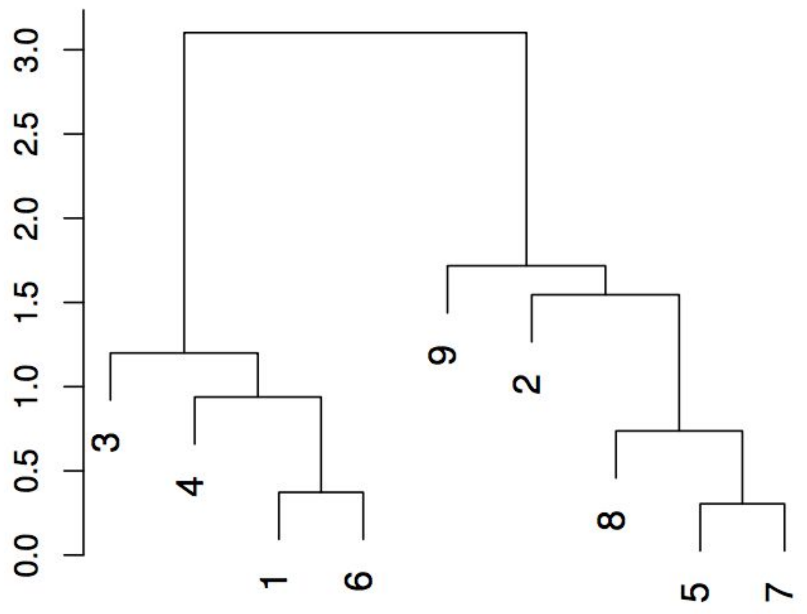
Hierarchical Clustering: “bottom-up” fashion

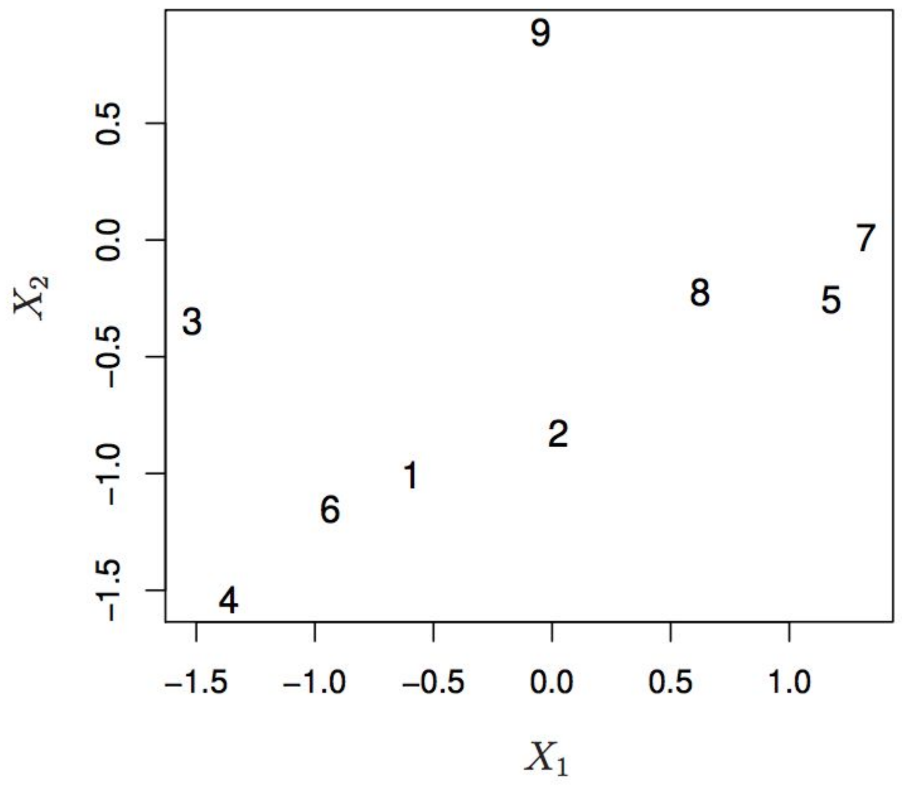


Interpreting the Dendrogram

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

Interpreting the Dendrogram: Visually





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 in step a, 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 a priori, 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 many cases, it is sufficient to use the Euclidean distance.
- A linkage is a measure of the dissimilarity between two 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 two groups of points?

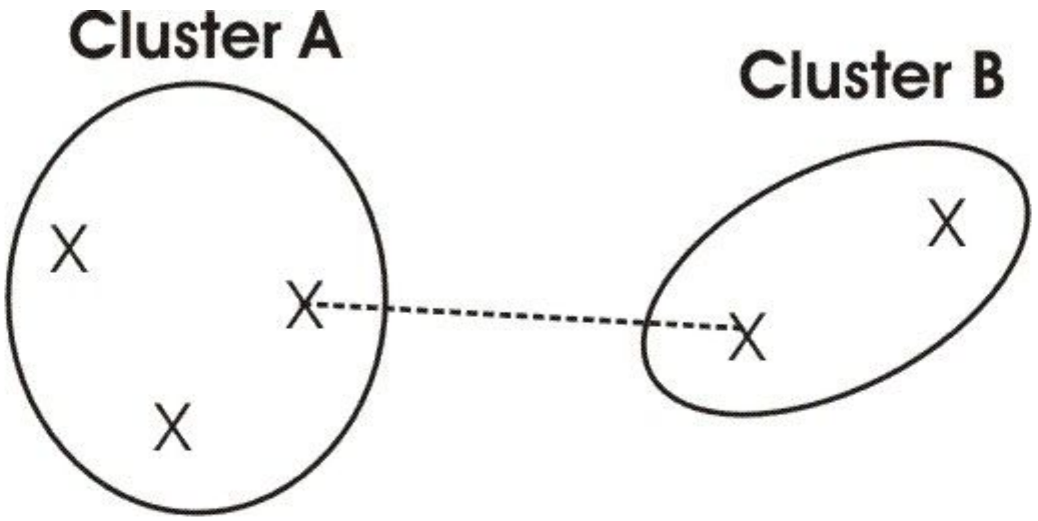
The Hierarchical Clustering Algorithm: Linkage

- 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 members of A and of B as the overall inter-cluster dissimilarity.
  - *Single Linkage*: Minimal inter-cluster dissimilarity.
    - Record the smallest of the dissimilarities listed between members of A and of B as the overall inter-cluster dissimilarity.

The Hierarchical Clustering Algorithm: Linkage

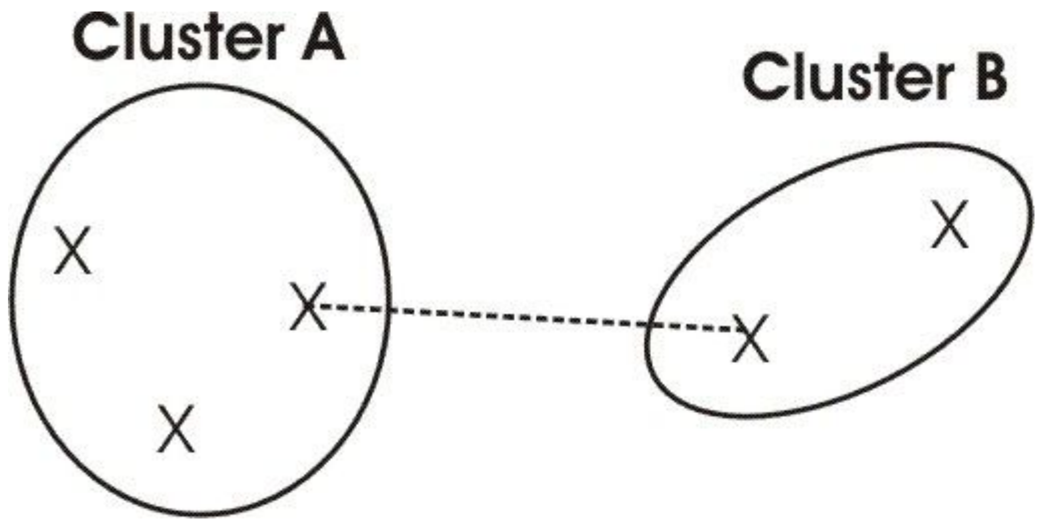
- *Average Linkage*: Mean inter-cluster dissimilarity.
  - Record the average of the dissimilarities listed between the members of A and of B as the overall inter-cluster dissimilarity.
- *Ward’s Linkage*: Minimum variance method (for Euclidean distance).
  - Minimize the variance of the clusters being merged.

The Hierarchical Clustering Algorithm: Linkage

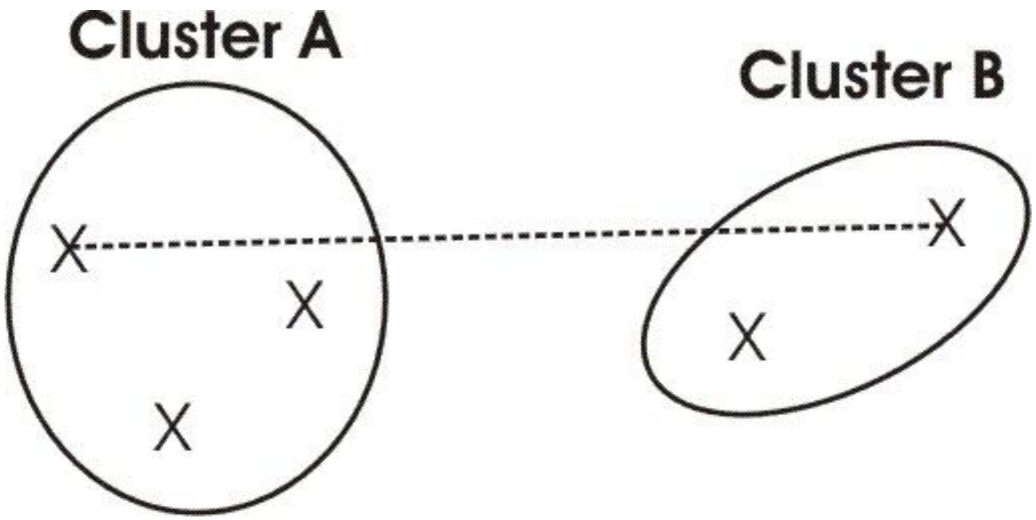


The Hierarchical Clustering Algorithm: Linkage

Single Linkage

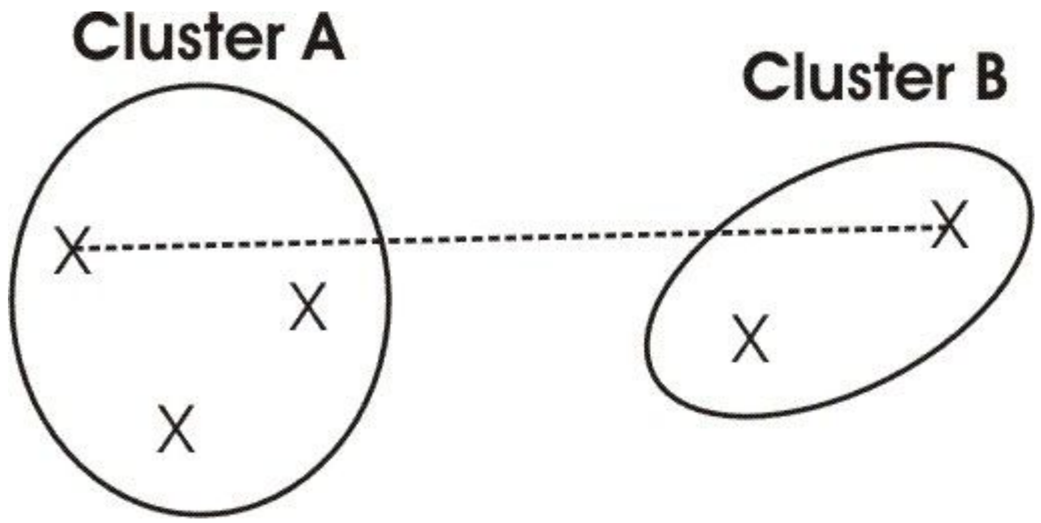


The Hierarchical Clustering Algorithm: Linkage

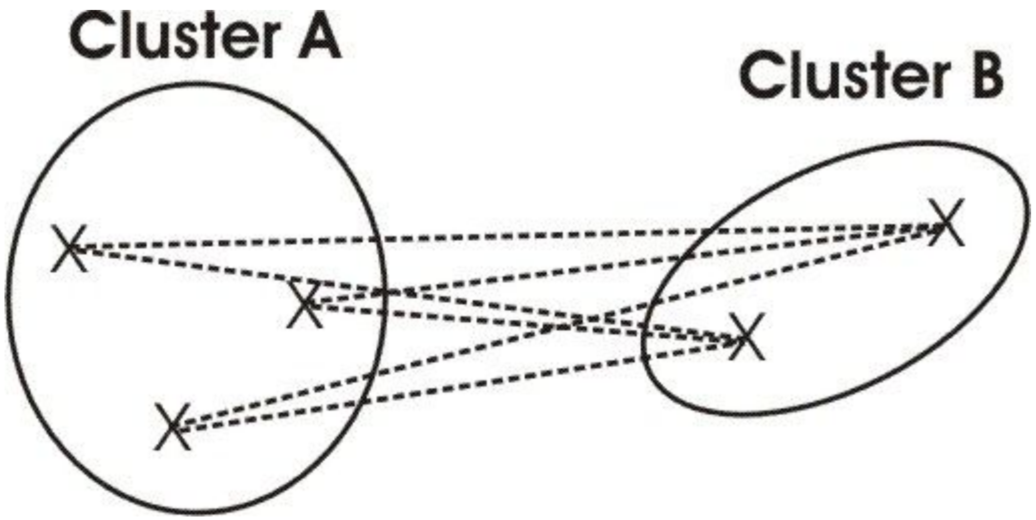


The Hierarchical Clustering Algorithm: Linkage

Complete Linkage

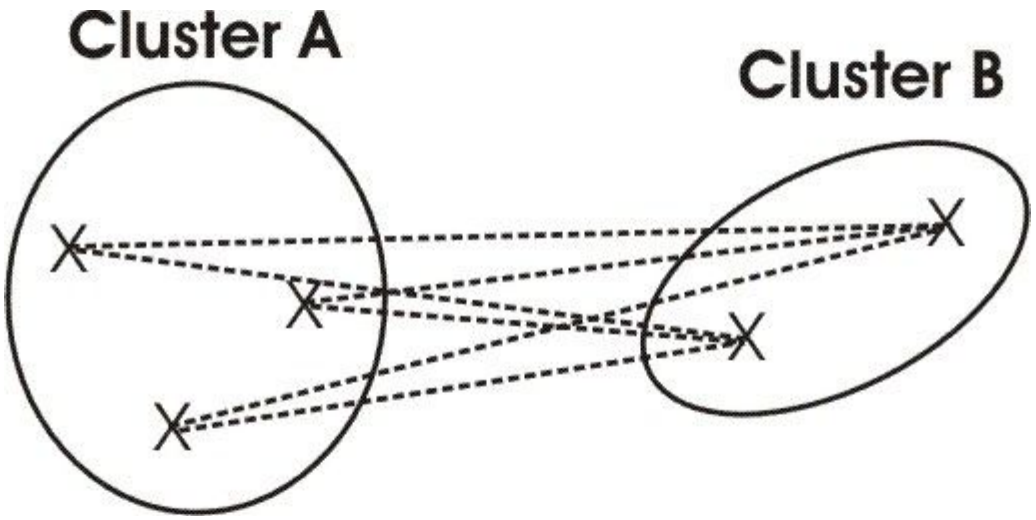


The Hierarchical Clustering Algorithm: Linkage



The Hierarchical Clustering Algorithm: Linkage

Average Linkage



The Hierarchical Clustering Algorithm: Linkage

- *Complete linkage* is sensitive to outliers, yet it tends to identify clusters that are compact, somewhat spherical objects with relatively similar 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 the observations in the same cluster 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.