CS559 – Unsupervised Learning

Lecture 3

Fall 2021

In Suk Jang

Outline

- Intro to Unsupervised Learning
- Clustering
    - K-means Clustering
    - Hierarchical Clustering
- Dimensionality reduction.
    - Principal Component Analysis

Recap From Last Lecture

- Machine Learning Overview
  - Preprocessing – EDA, Feature Engineering, and Imputation…
  - Modeling – Training, Validating, and Testing
    - Cross Validation
  - Supervised Learning vs. Unsupervised Learning

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
- Two methods:
  - Principal Components Analysis (PCA) – 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 subjets 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.

Cluster Analysis

We have been concerned with building models that **perform predictions**:

- Regression systems that attempts to predict a numeric outputs
- Classifiers that attempts to predict class membership.

In contrast, *cluster analysis* is an unsupervised task that:

- Does **<u>not</u>** 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.

Clustering 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.

K-Means Clustering

With the K-means clustering algorithm, we aim to split up our observations into a predetermined number of clusters.

- The number of clusters *K* must be specified in advance.
- These cluster memberships are distinct and non-overlapping.

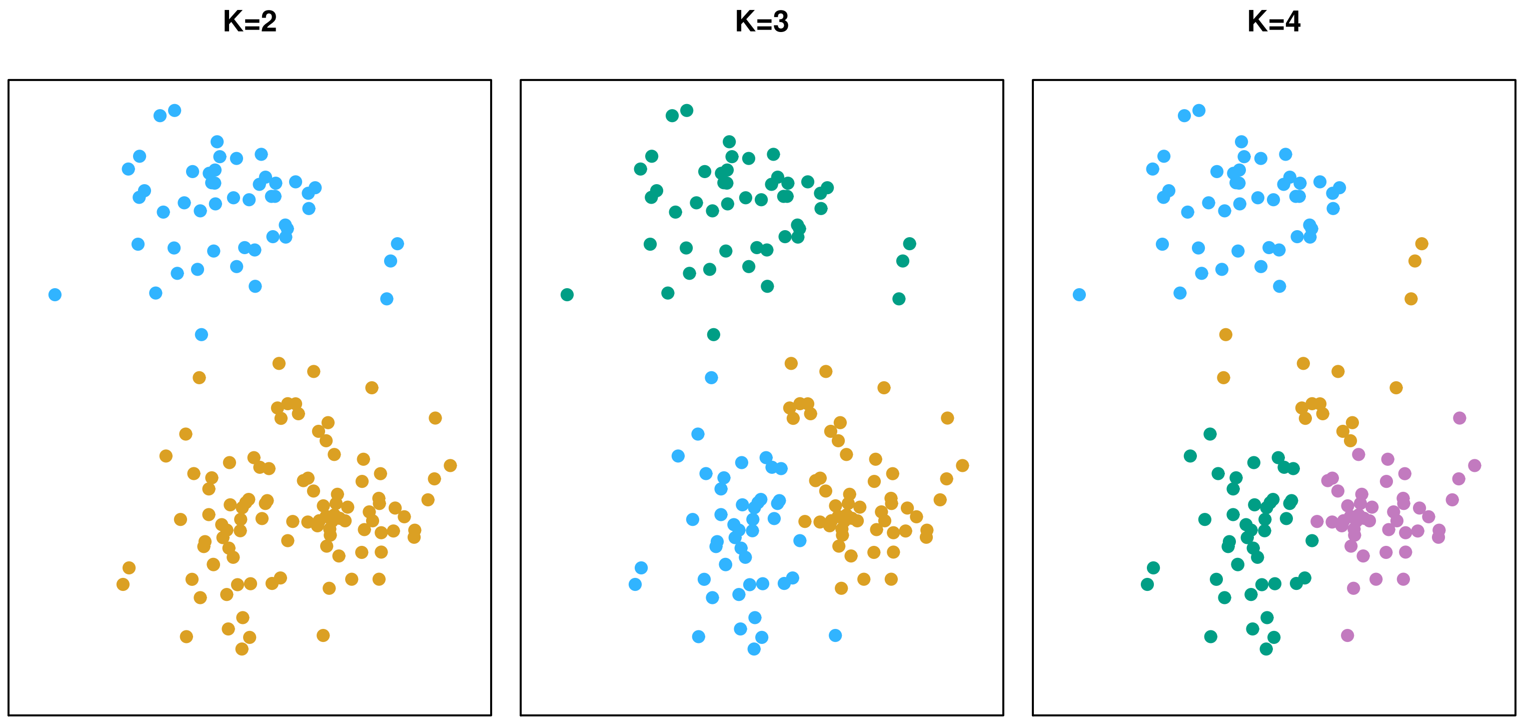
The data points in each of the clusters are determined to be mostly similar to a specific centroid value:

- 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 observation 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:

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

The number of variables/features in our dataset.

The total number of observations in cluster *k*.

Indices of observations in cluster C_k

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:

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

K-Means Clustering

To find the global minimum of the above function is very difficult.

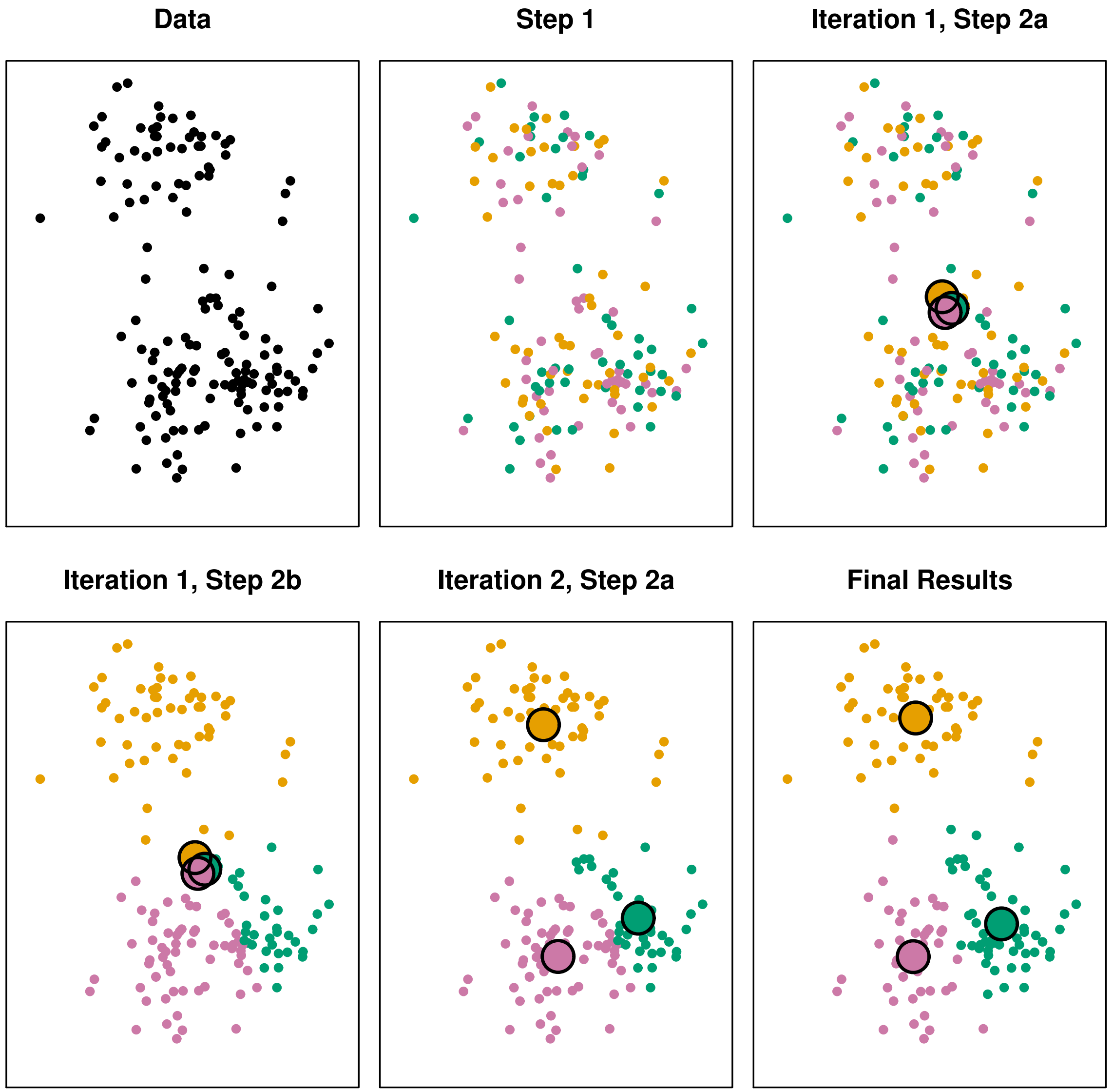
In practice, most K-means packages perform the following greedy algorithm, also known as Lloyd algorithm in the computer science circle:

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:
   1) For each of the K clusters, compute the cluster's new centroid.
   2) 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):



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***.

K-Means Clustering

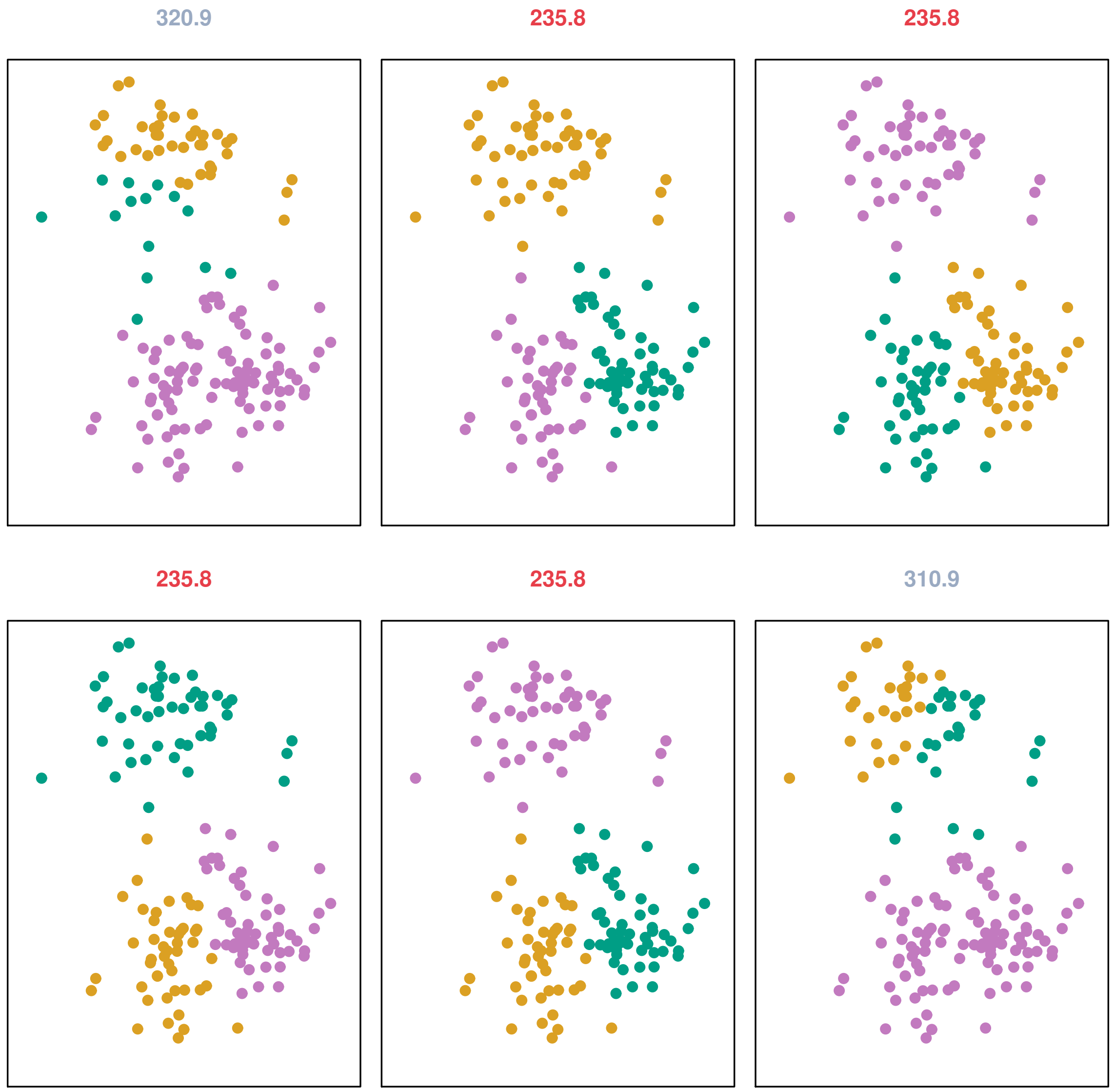
The Kmeans++ (2007) improves the random seeding of the original KMeans.

- 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 trends to keep a far distance from the existing ones.

The default initialization scheme of Scikit-Learn’s KMeans uses KMeans++.

K-Means Clustering

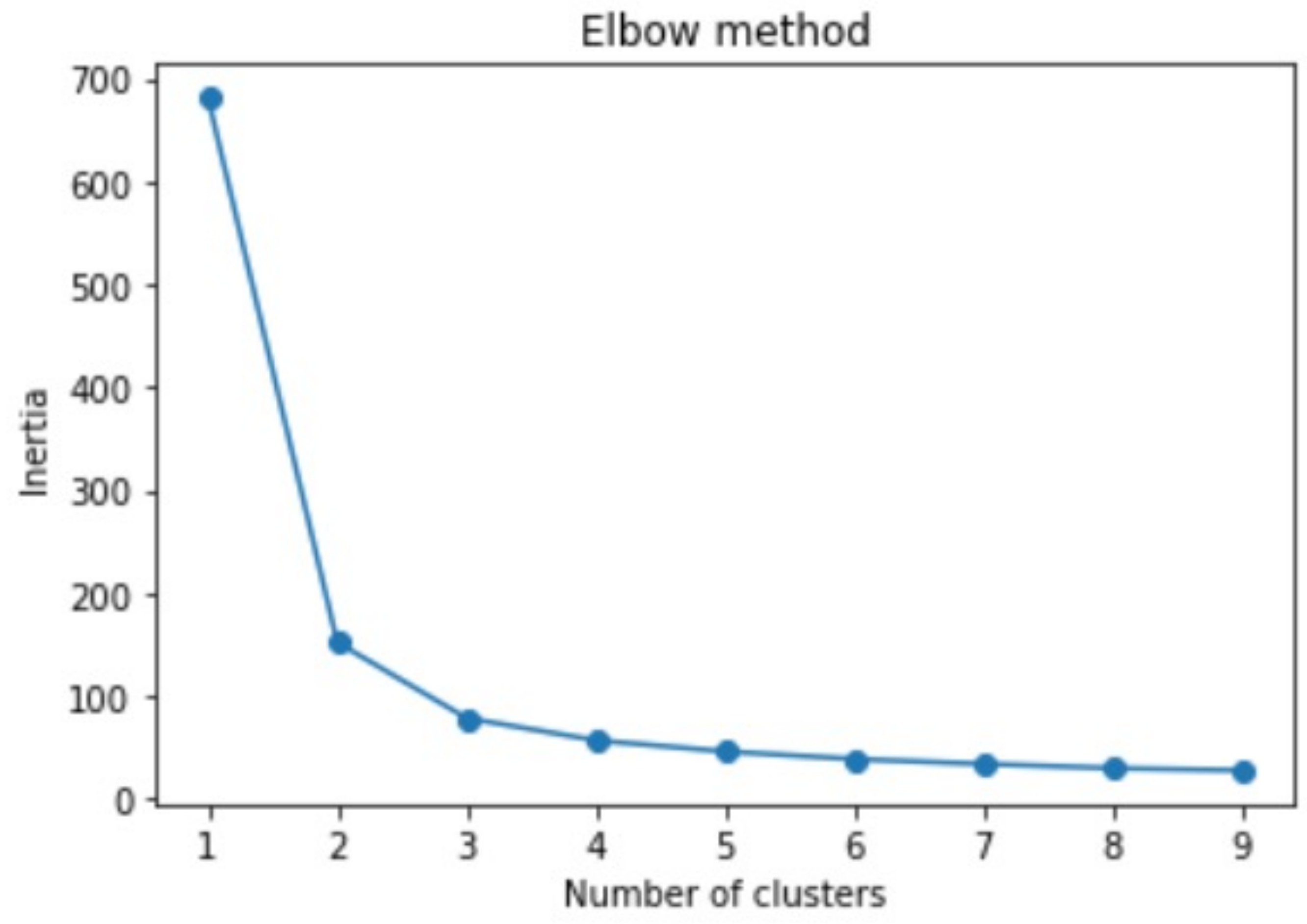
The algorithm will stop when it has found the least/best (local optimum) value.



K-Means Clustering

How do we determine the K value?

- We use the elbow method



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 observations.
  - At the highest level, all of the data points from a single cluster.

Hierarchical Clustering (Agglomerative Clustering).

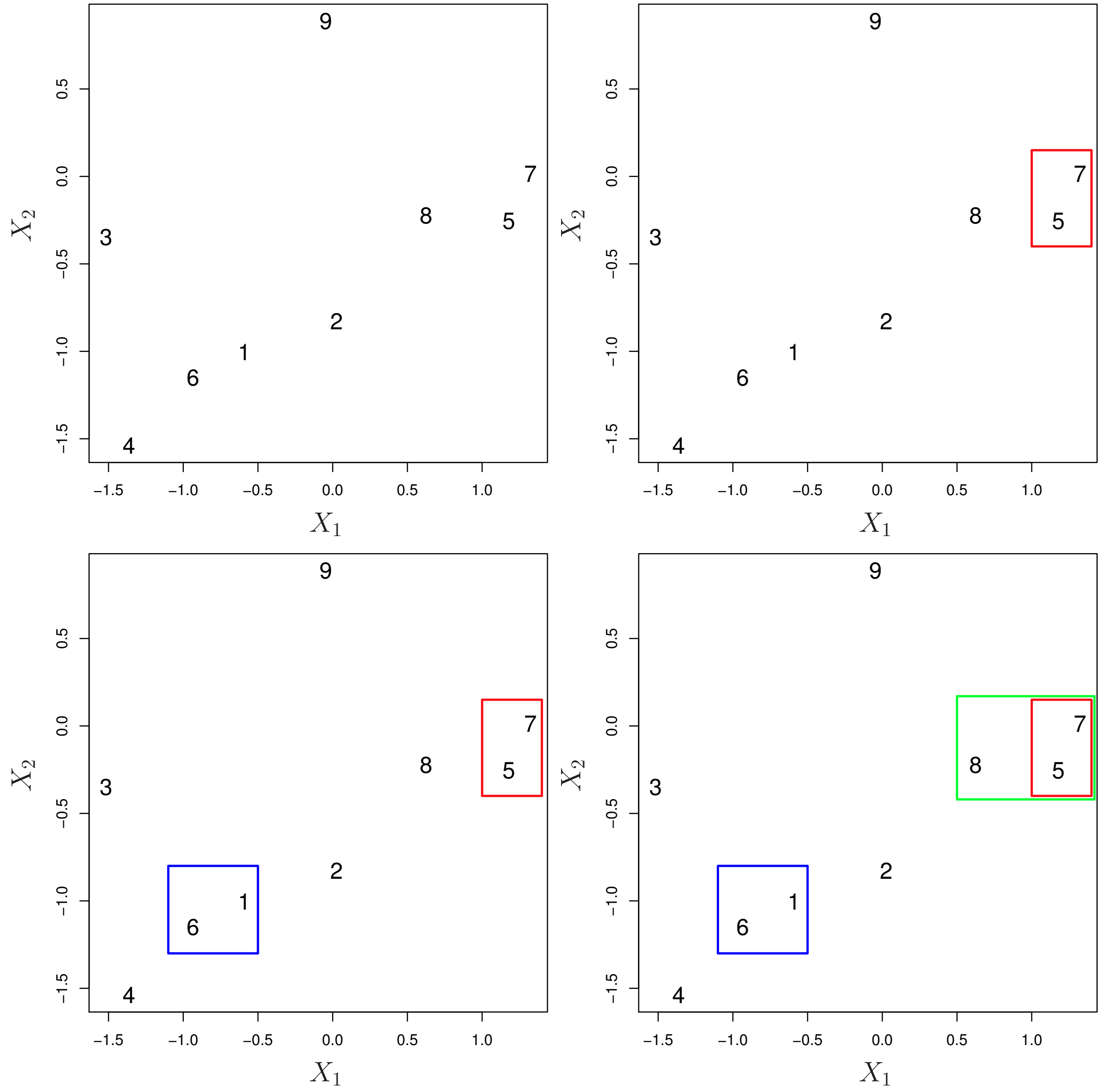
Two strategies for hierarchical clustering: bottom-up and top-down.

The approach can be summarized as:

- Start with each point in its own cluster.
- Identify the two clusters which are most similar and merge them.
- Repeat step 2.
- Ends when all data points are in a single cluster.

In the next few slides we show how to build a hierarchy in a bottom-up fashion.

Hierarchical Clustering (Agglomerative Clustering).



Hierarchical Clustering

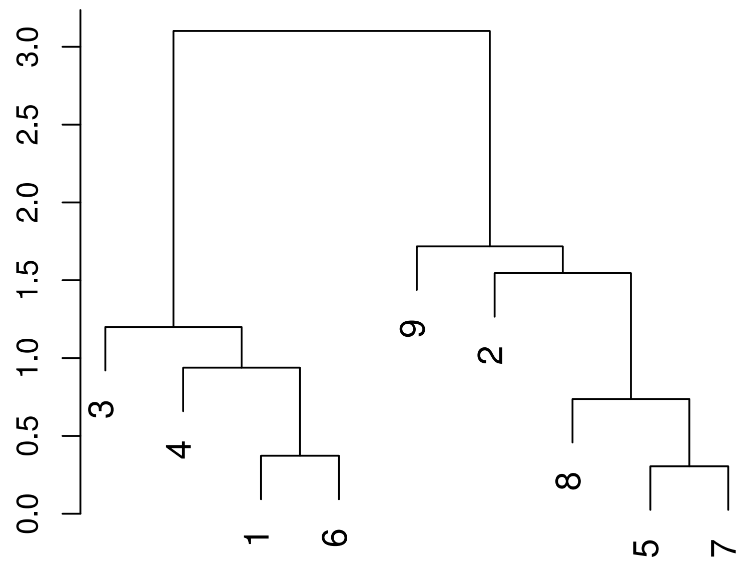
There are some interpretative advantages to visualizing the <u>dendrogram</u> created from hierarchical clustering:

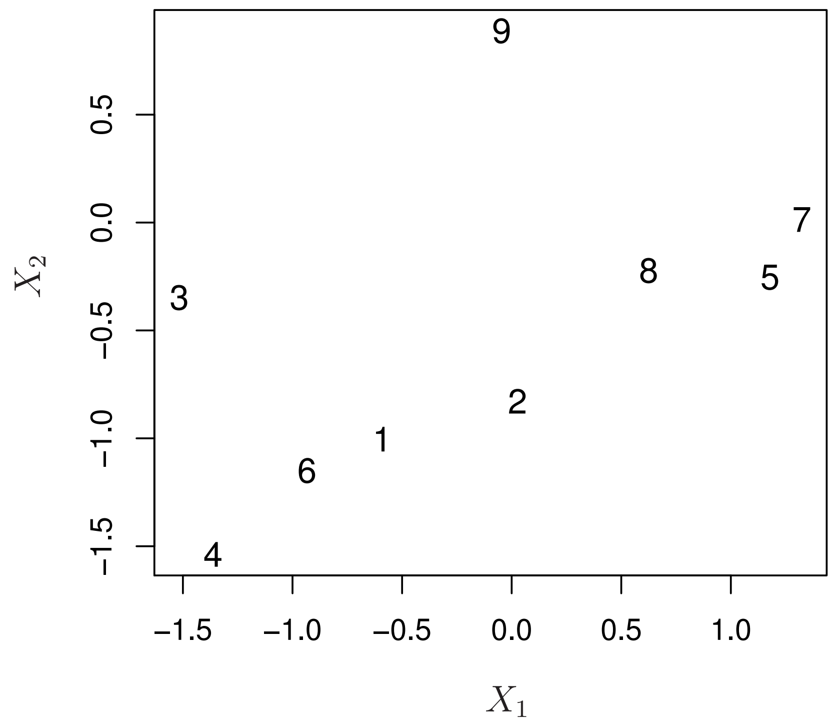
- The lower down in the dendrogram a cluster fusion occurs, the more similar the fused clusters are to each other.
- The higher up in the dendrogram a fusion occurs, the more dissimilar the fused groups are to each other.

In general, for any two observations we can inspect the <u>dendrogram</u> 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.

Hierarchical Clustering (Agglomerative Clustering).





Hierarchical Clustering Algorithm

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.

For $i = n, (n-1), \dots, 2$:

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.

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?

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.

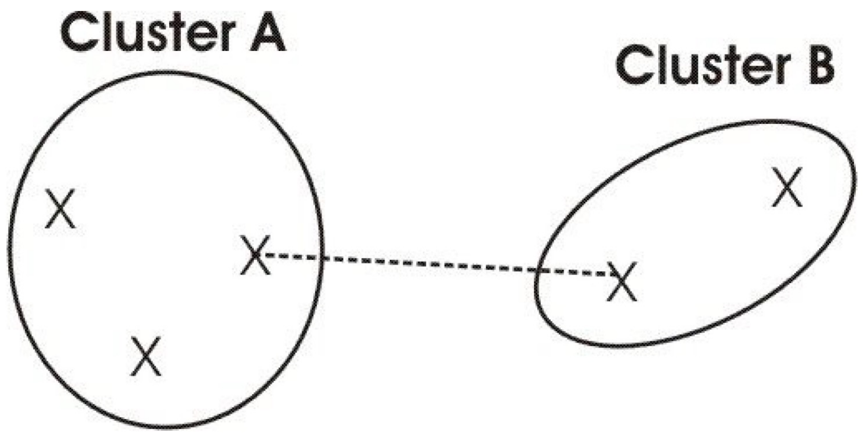
Hierarchical Clustering Algorithm: Linkage

- *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.

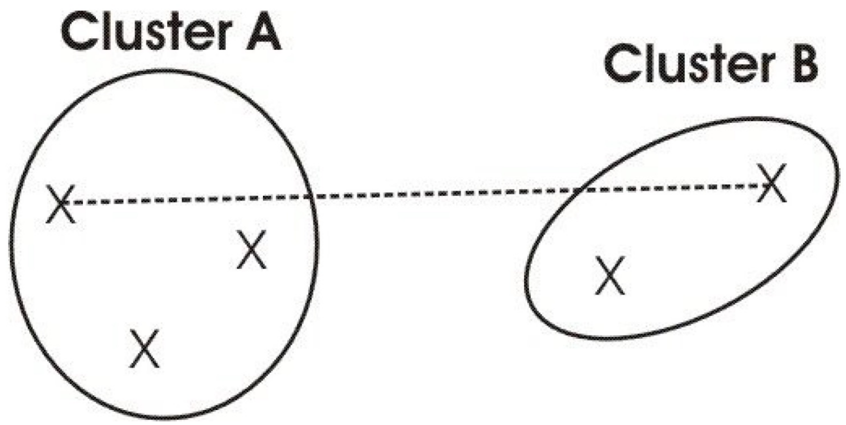
- *Ward's Linkage*: Minimum variance method (for Euclidean distance).
  - Minimize the variance of the clusters being merged.

Hierarchical Clustering Algorithm: Linkage

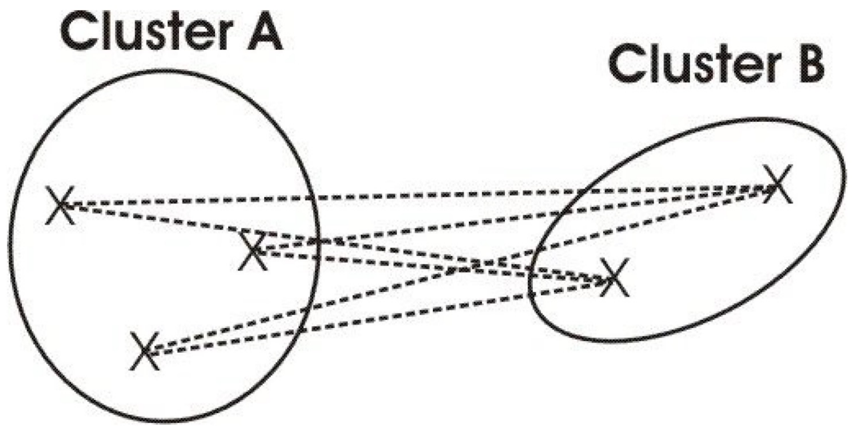
Single Linkage



Complete Linkage



Average Linkage



- *Complete Linkage*: Maximal inter-cluster dissimilarity.
- *Single Linkage*: Minimal inter-cluster dissimilarity.
- *Average Linkage*: Mean inter-cluster dissimilarity.

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.

Principal Component Analysis (PCA) – Dimensional Reduction

PCA - Multicollinearity

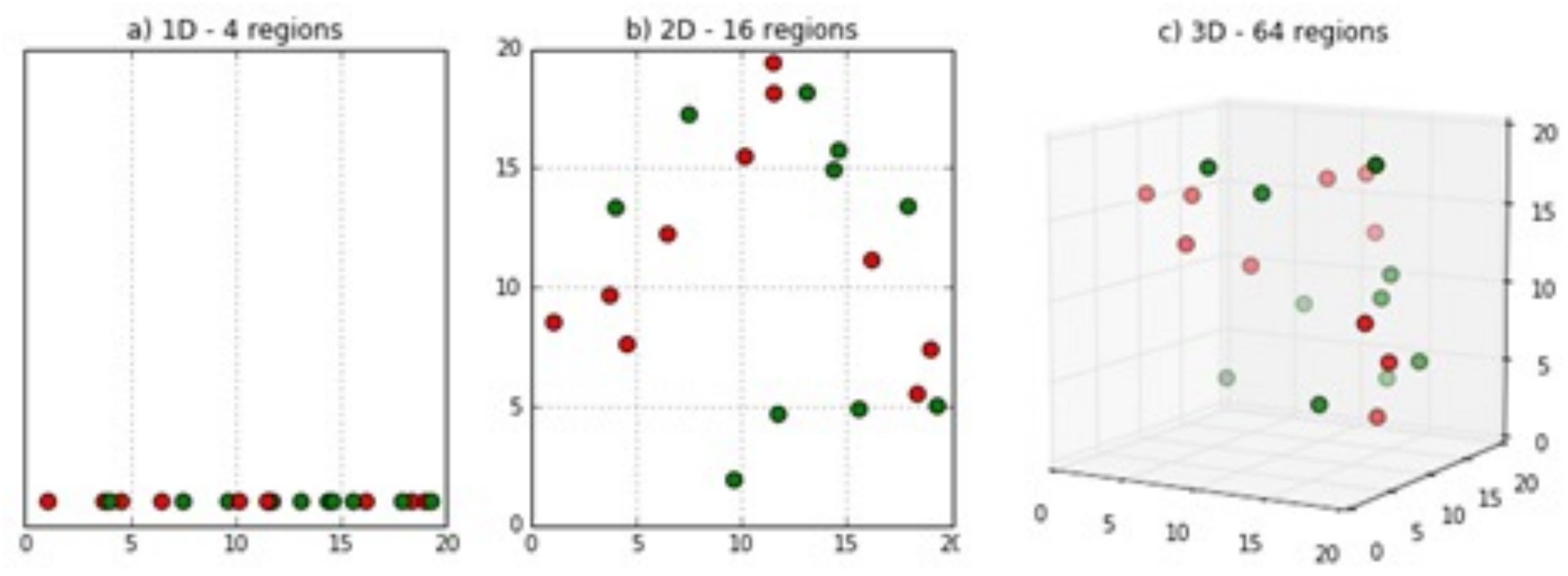
Multicollinearity is a phenomenon in which two or more predictor variables in a multiple regression model are *highly correlated*, meaning that one can be predicted from the others through linear formulae with *a substantial degree of accuracy*.

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*.
- The standard errors of the affected coefficients tend to be *large*.

PCA - Multicollinearity

- 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 **Supportive Vector Machine** takes advantage of **the curse of dimensionality**.



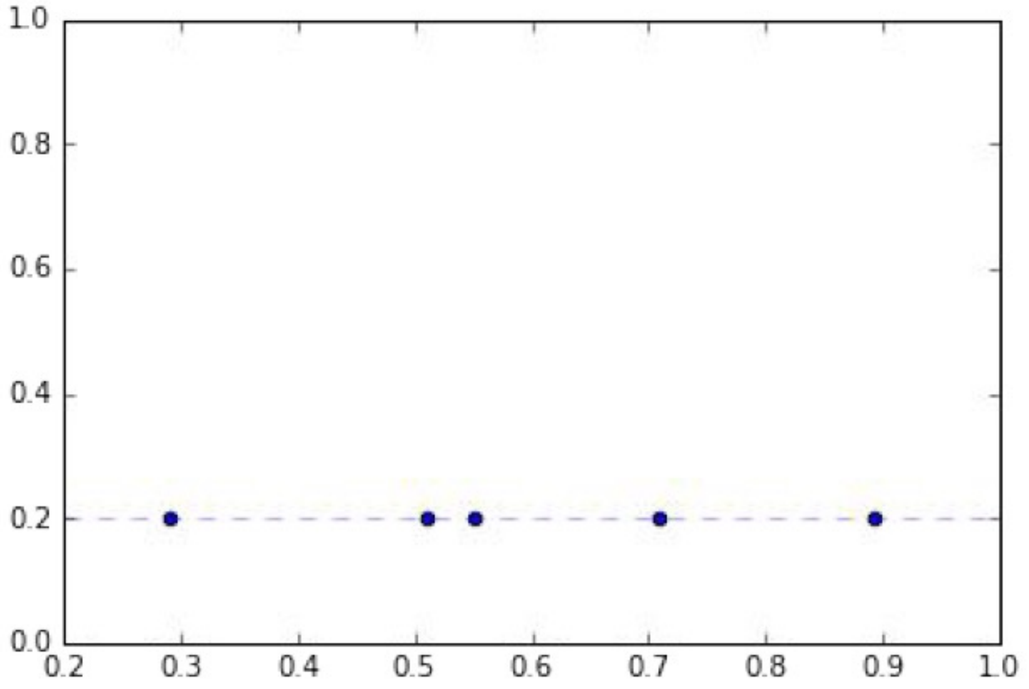
PCA

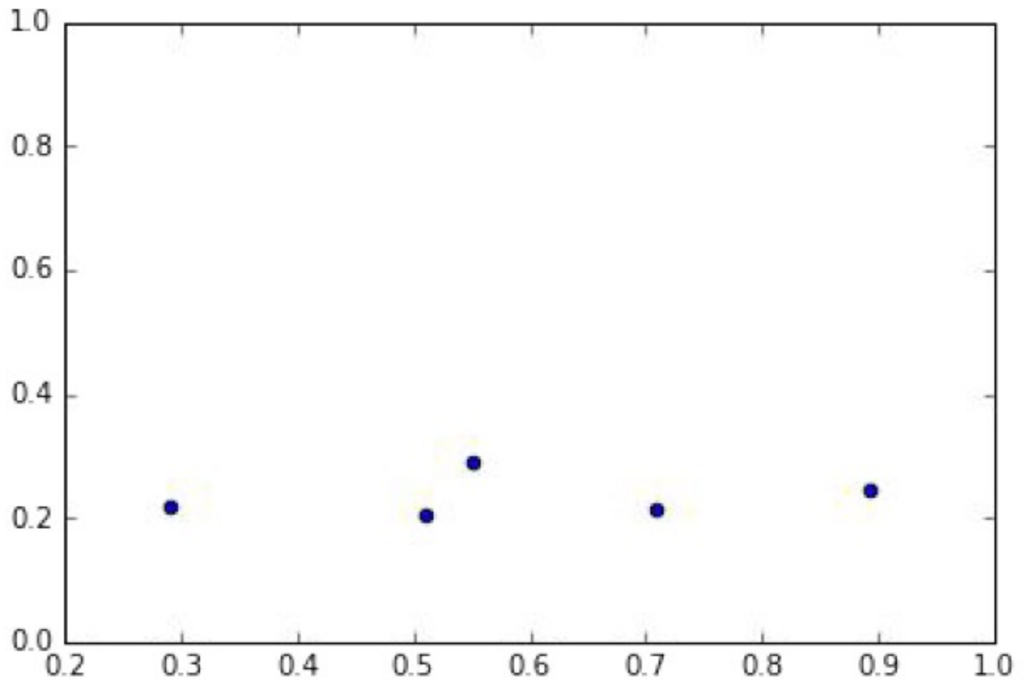
Principal component analysis (PCA) is a tool that finds a sequence of linear combinations of the variables to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called **principal components**.

Ideal input variables:

- Linearly uncorrelated
- Low-dimensional in the feature space

PCA – Geometric Motivation

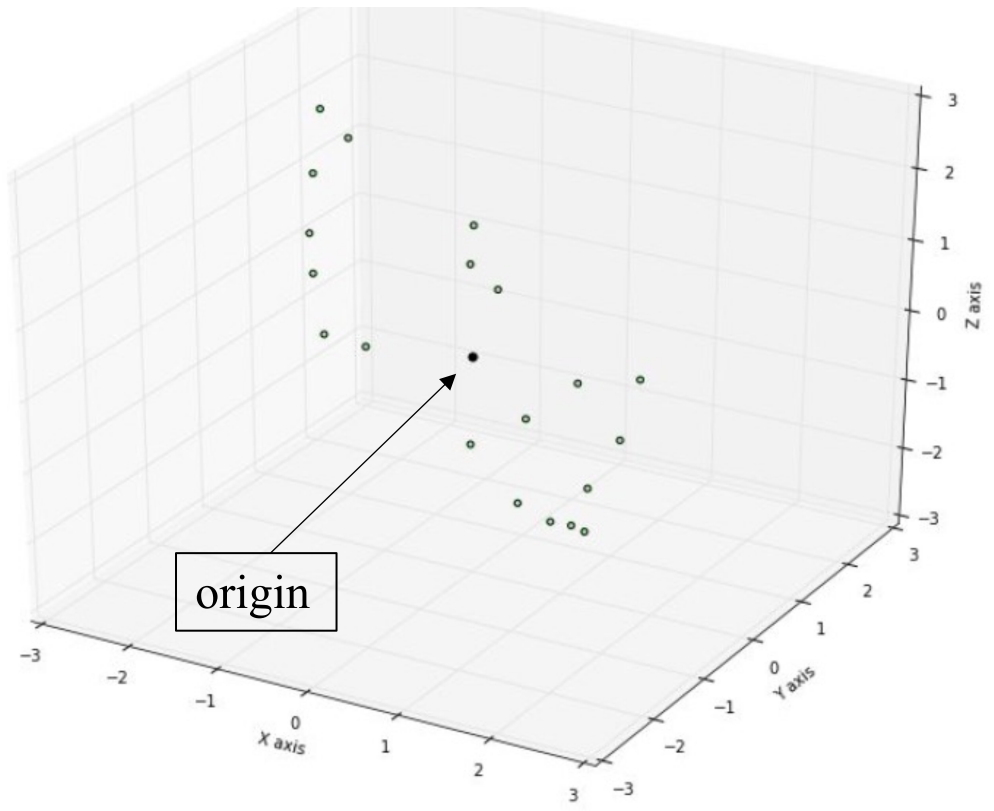


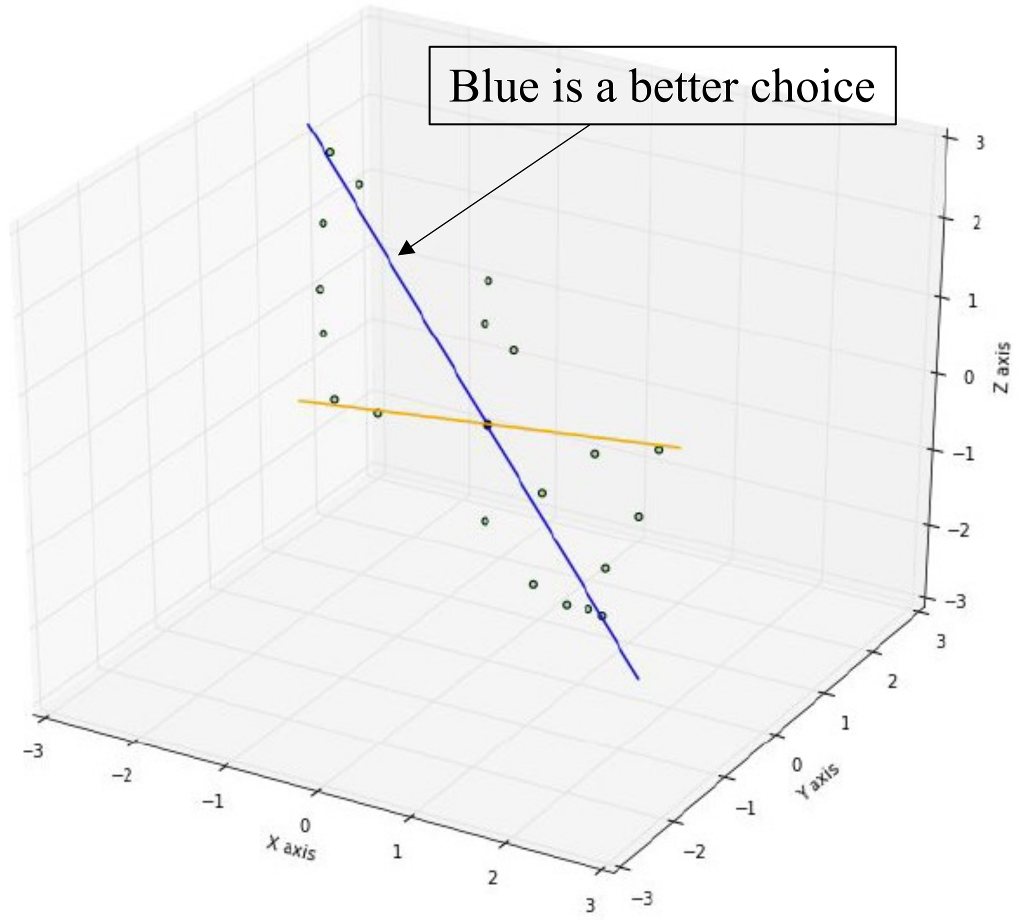


Left: We always do not need all the features. The *y* component of all the points are the same, it provides **NO** additional information.
Right: *y* values are restricted in a much smaller region than *x* values. This suggests that *x* component might provide more information.

PCA – Geometric Motivation

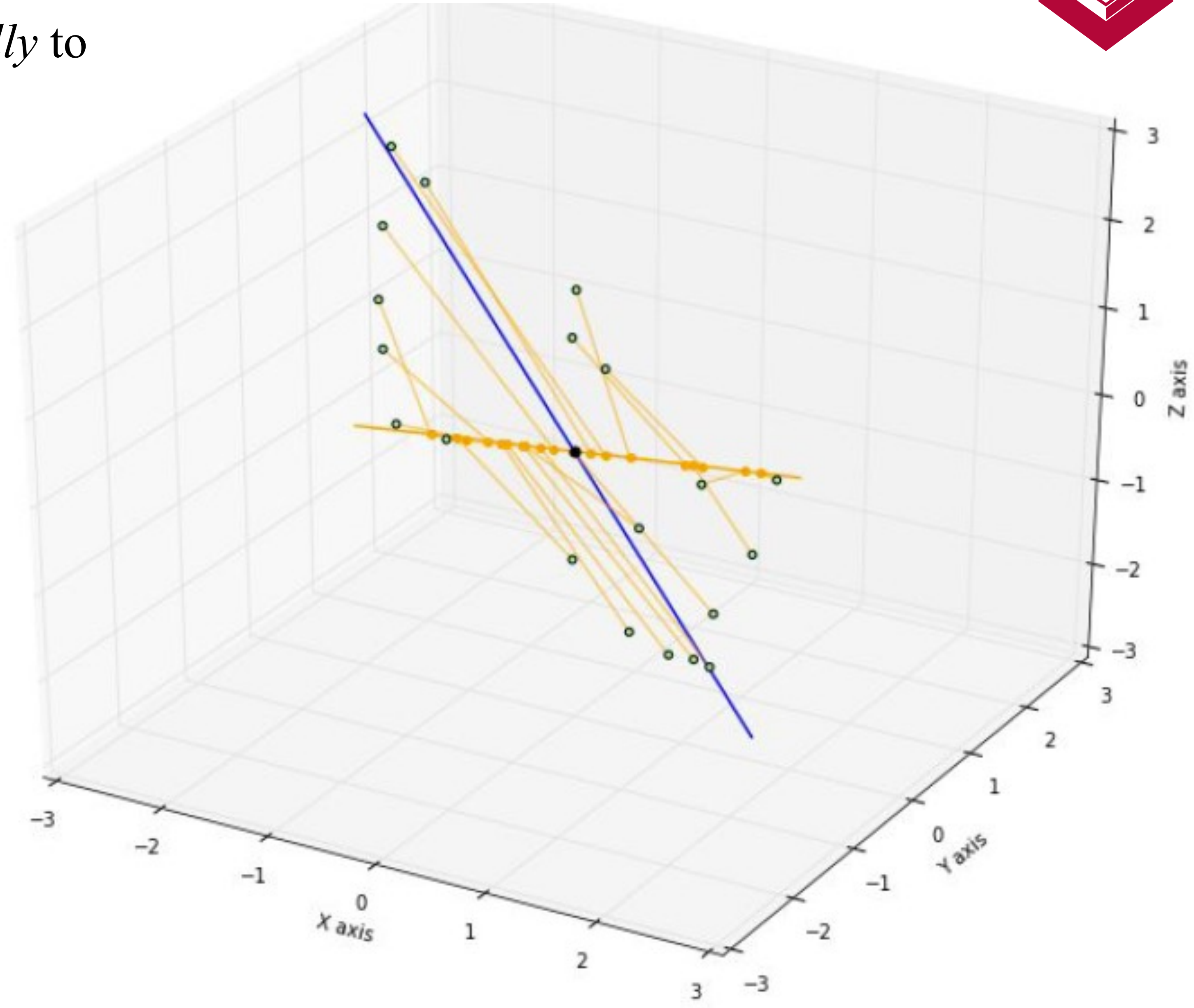
- Left: Consider a set of 20 points in a three dimensional space. In such a scenario, we have: 20 observations and 3 features.
- Right: We compare the **importance** of each direction. Note that the chosen directions in the example are *not parallel* to any coordinate axis.





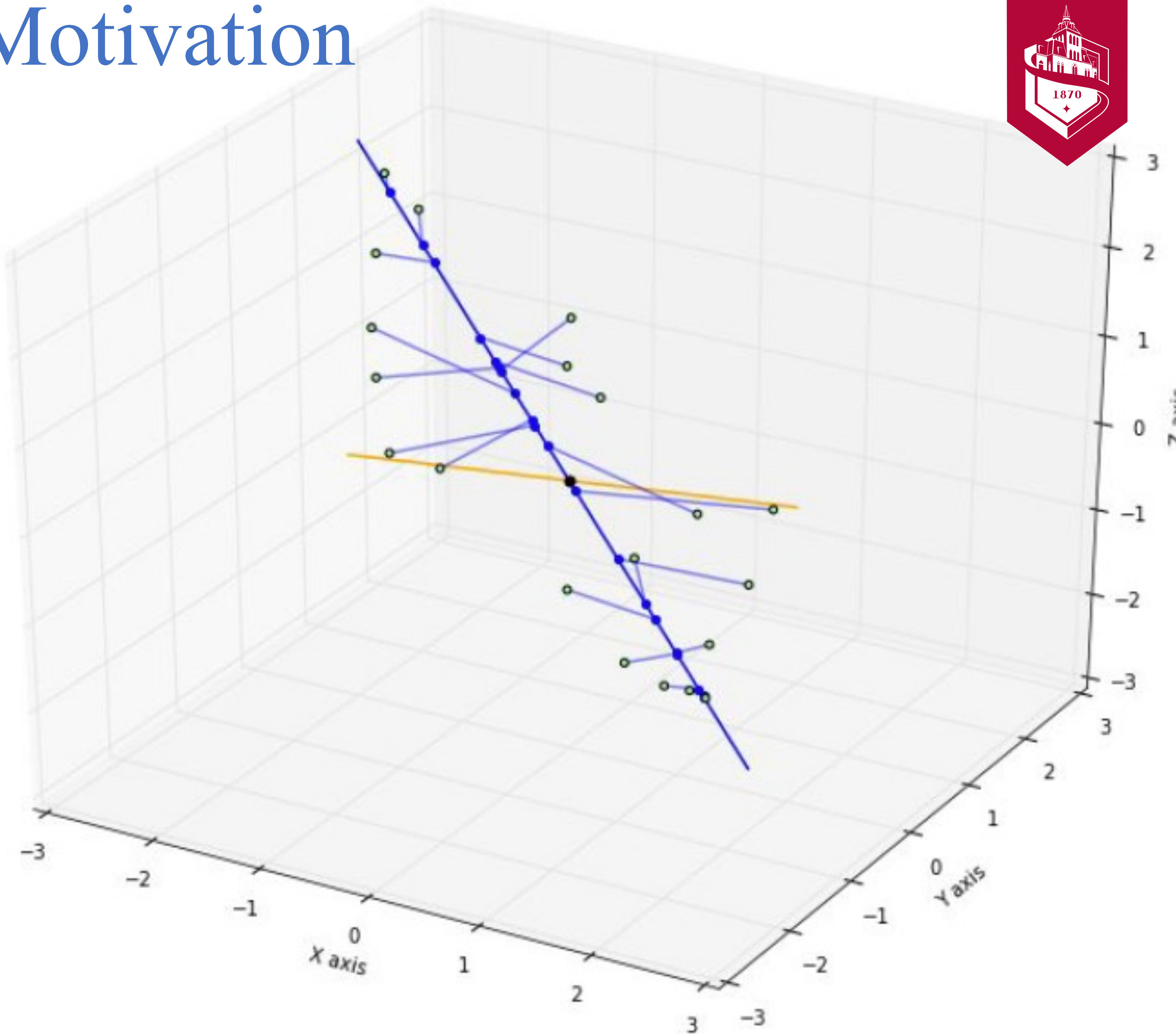
PCA – Geometric Motivation

- We project each observation *orthogonally* to the "orange" direction.



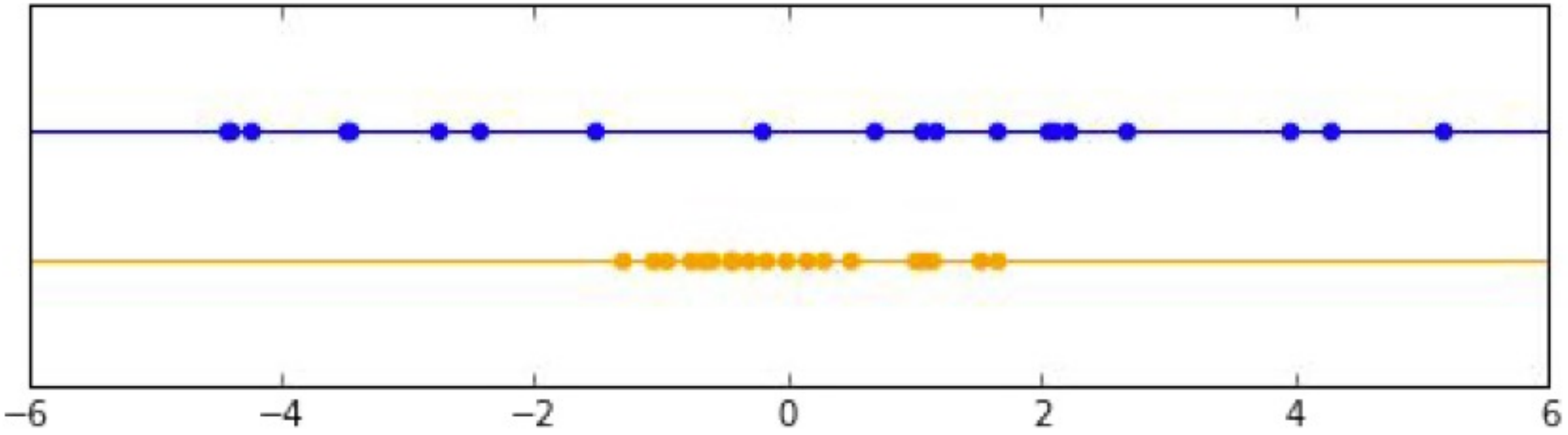
PCA – Geometric Motivation

- We project each observation *orthogonally* to the "blue" direction.



PCA – Geometric Motivation

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



PCA – Geometric Motivation

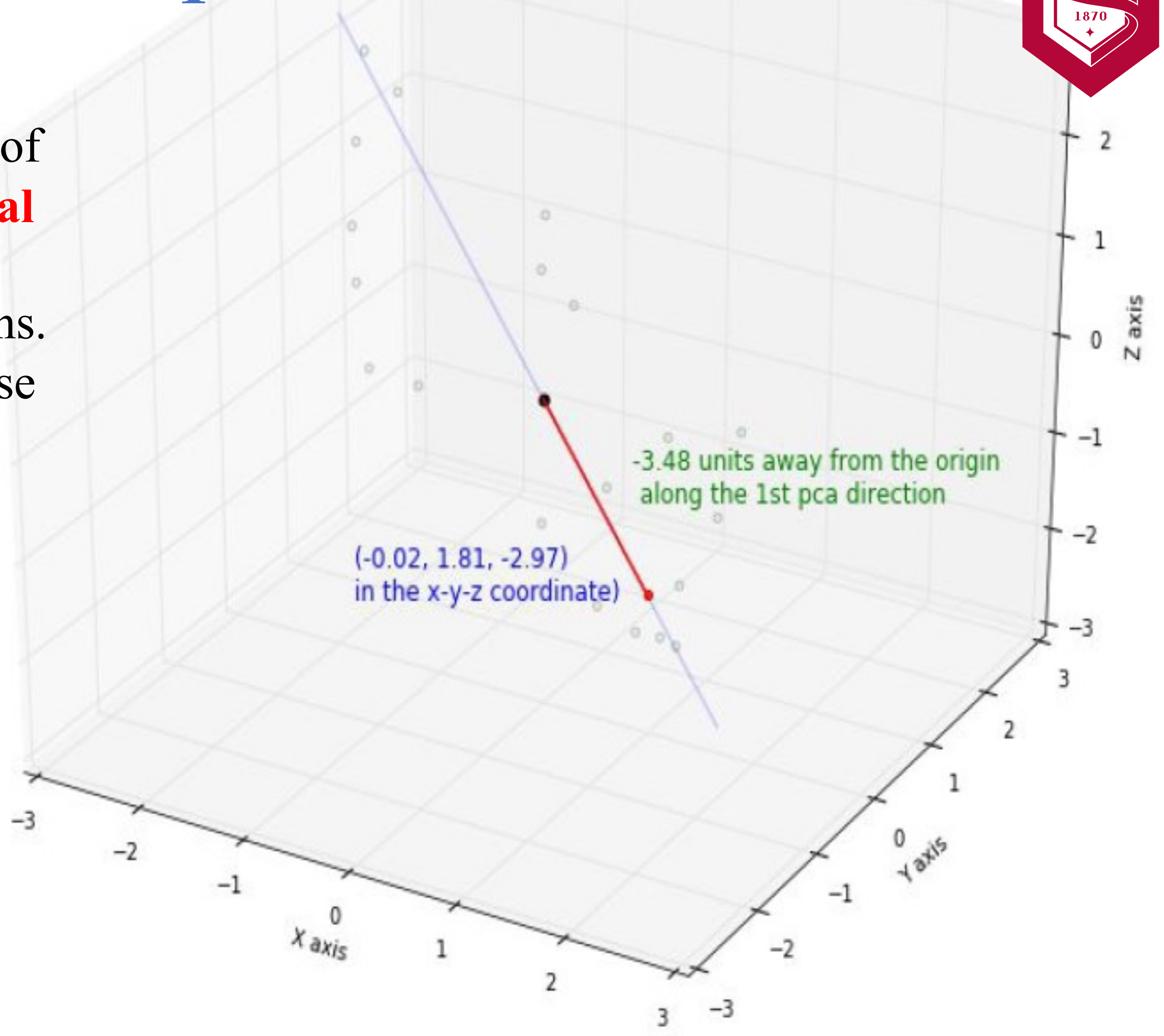
- The "blue" direction above is actually the first loading vector, which means:
  - it is the direction into which the projection of the observations is more widely spread than the projection into any other direction.
  - being a direction (vector), it has as many entries as the number of the features.

- The statements above characterize the principal direction. To find the principal direction we need to apply the technique of linear algebra.

- 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 done by orthogonal linear projection.
  - There are in general N (the number of samples) components for **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.

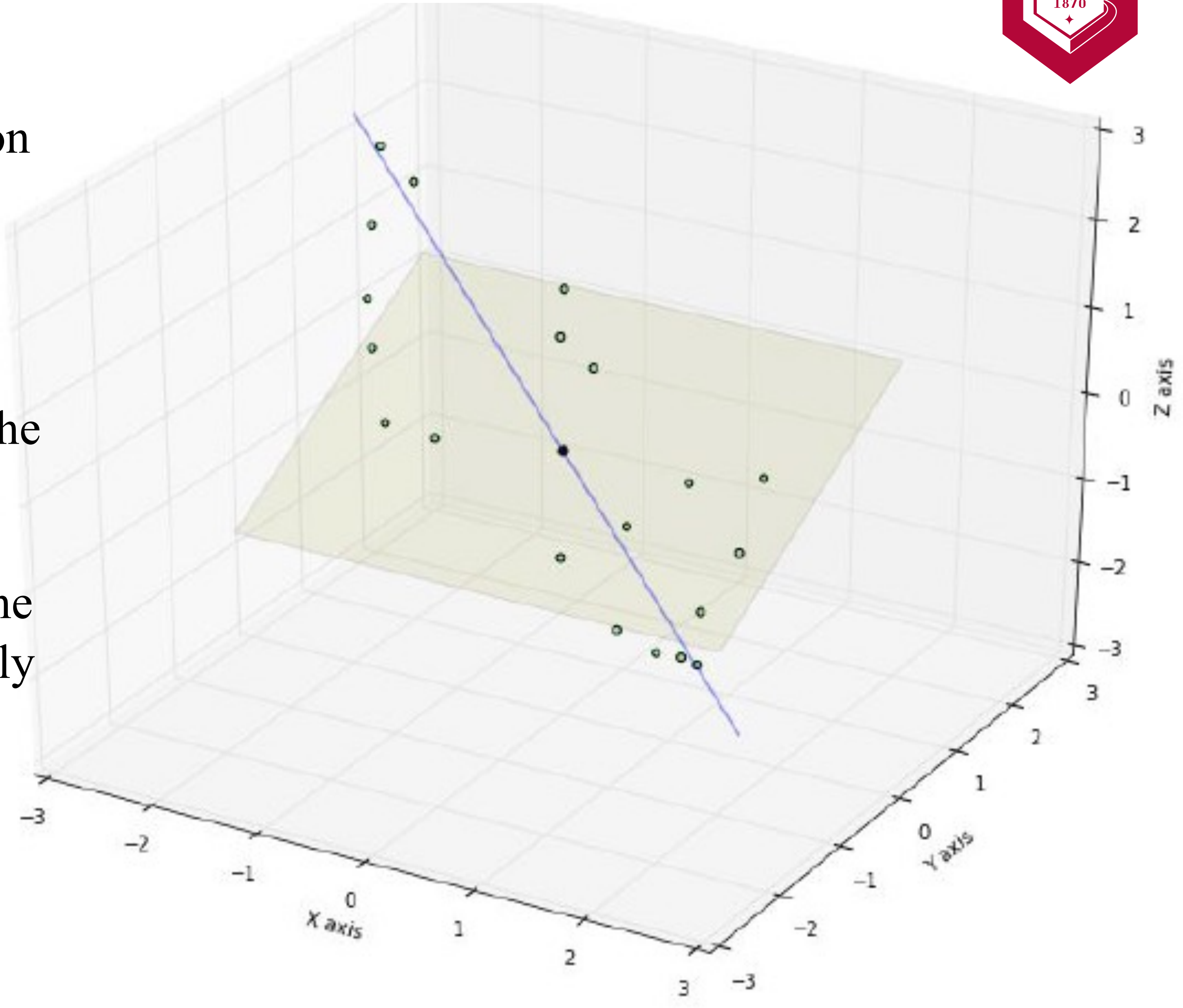
PCA – First Principal Component

- The first loading vector gives us a vector of length 20. This vector is the **first principal component**.
- We need 20 records for all the observations.
- We do not use the *xyz*-coordinates – we use **one coordinate**, the first principal component.
- The *red projection* can describe a certain length away from the origin along the principal direction and is a vector in the original *xyz*-coordinates.



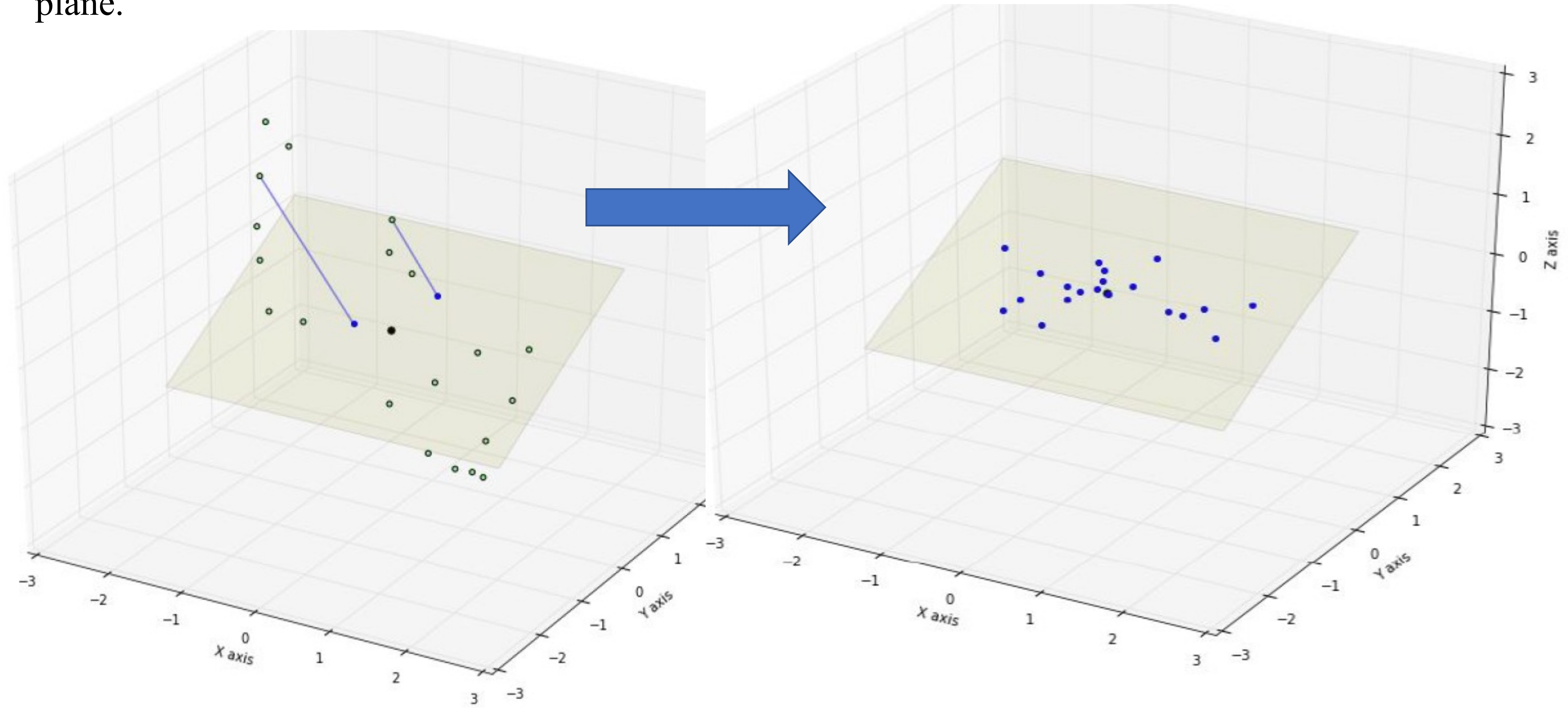
PCA – Second Principal Component

- The information stored is about the variation of the points across the whole sample set.
- Not all directions are born equal.
- The first principal component provides the most information but most likely not all.
- We remove the data information stored in the first principal component.
- Then find the new direction (orthogonal to the original principal direction) on which the projection of the observations is most widely spread.
- We consider the 2-D plane that is perpendicular to the first loading vector.



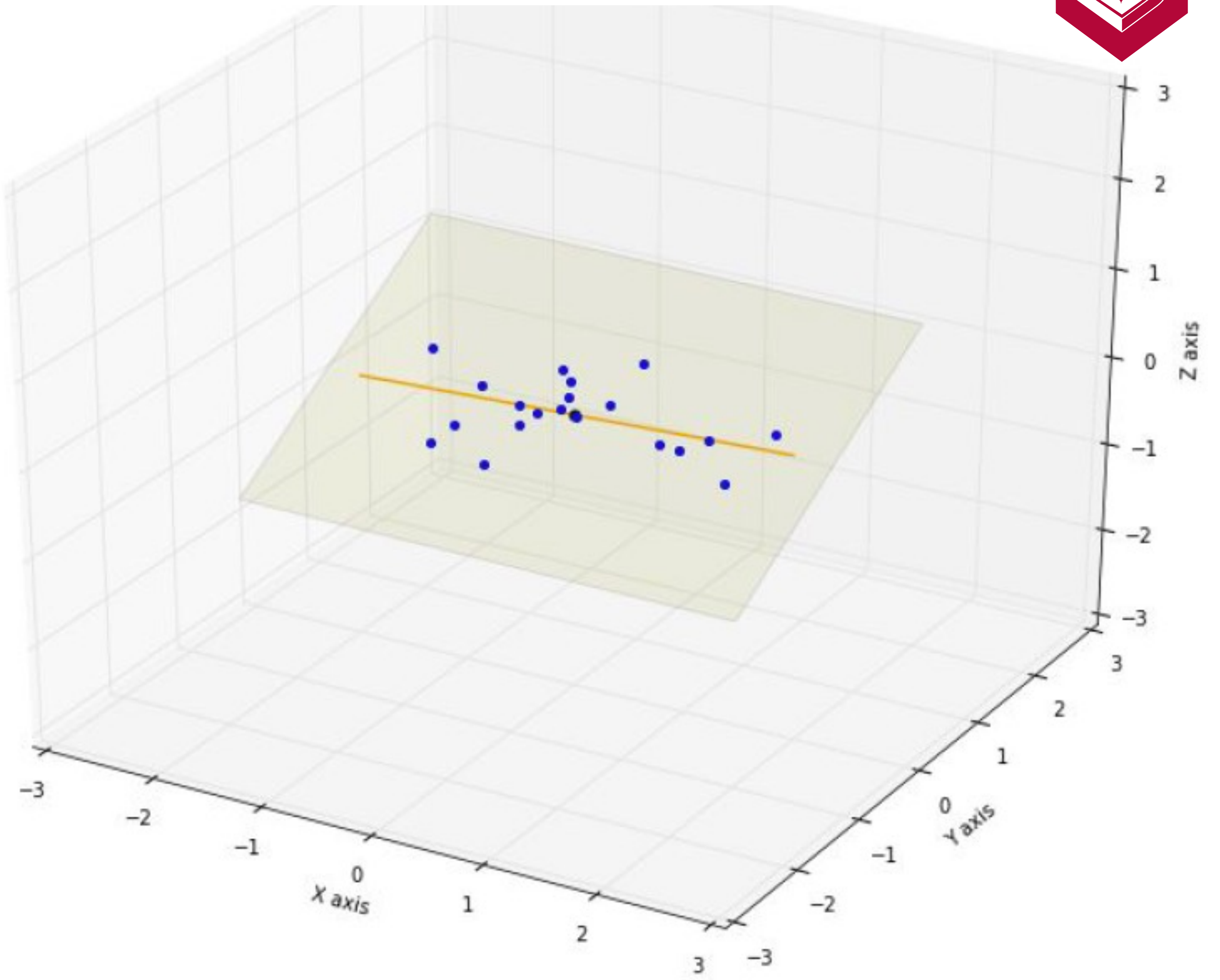
PCA – Second Principal Component

- We can remove the effect of the first principal component by first projecting the observations to th plane.



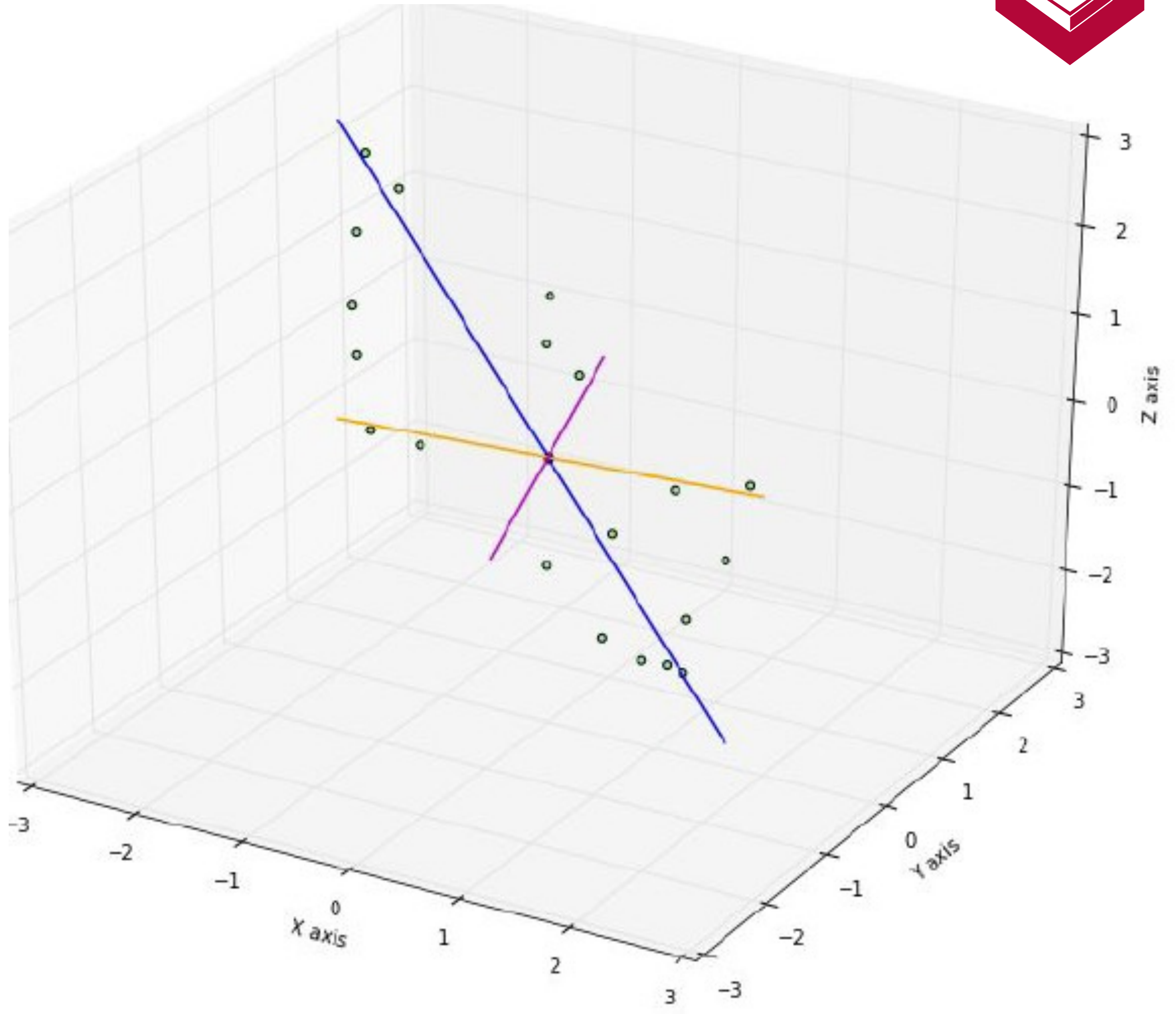
PCA – Second Principal Component

- We find the direction on which the projection is most widely spread.
- Since all the provided 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**.



PCA – Second Principal Component

- This process can be continued to retain more and more information from the raw data. However, we remove one dimension each time when we remove the information recorded in a principal direction.
- We cannot 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.



PCA – The Mathematical Formulation

- The first (very important) step is to centralize the raw data. Assume that our data X is an n by p matrix. 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 $\hat{u}$ and the projection is $X\hat{u}^T$.
- We need to find the direction on which the the projection of the data is most widely spread.

$$\boxed{\phi = \max Var(X\hat{u}^T)\ \ s.t. \left|\left|\hat{u}\right|\right| = 1}$$

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

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

which is called the first principal component.

PCA – The Mathematical Formulation

- Once the first k-1 principal component 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).

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

- With this matrix we solve the optimization problem again:

$$\boxed{\phi_k = \max Var(X_k\hat{u}^T)\ \ s.t. ||\hat{u}|| = 1}$$

- Again the solution ϕ_k is the k_{th} loading vector and the projection on this direction is called the k_{th} principal component.

$$\boxed{Z_k = X_k\phi_k^T}$$

PCA – The Properties

- 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: $Var(Z_1) \geq Var(Z_2) \geq \cdots \geq Var(Z_p)$
- 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 variance of the data along the principal directions (eigenvectors) are the corresponding positive eigenvalues.

PCA – Geometrical Meaning

- Geometrically we can imagine that the original data set sits inside $\mathbb{R}^f$ as a high dimensional scatterplot.
- The selection of the top p principal directions establishes a linear projection $\mathbb{R}^f \to \mathbb{R}^p$ into a lower dimensional space.
- There are many orthogonal linear projections from $\mathbb{R}^f$ to $\mathbb{R}^p$. But PCA is special 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 noise of the data.
- 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.
- Nonlinear projections into curved objects instead of linear spaces is called **manifold** learning as non-linear smooth objects are called manifolds in geometry.

Highlights from This Lecture

- Unsupervised Learning
    - How it is different from supervised learning?
        - Data Structure – unlabeled data
        - Different goal – no prediction
    - What are we trying learn in unsupervised learning?
        - Learn more about data structure, meanings of data itself
- K-mean Clustering
    - Easy to implement but has uncertainties of making decisions on cluster numbers.
    - Elbow method
- Hierarchical Clustering
    - Options of making clusters (e.g., distance, linkage)
    - Similar to decision trees
- PCA
    - Linear Combination
    - Dimension Reduction
    - Often used in pre-processing