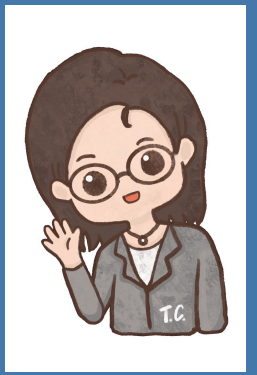
MTH 208 Exploratory Data Analysis

Lesson 07: Clustering and Dimensionality Reduction

Ying-Ju Tessa Chen, PhD

Associate Professor
Department of Mathematics
University of Dayton

@ying-ju
ying-ju
ychen4@udayton.edu



Learning Objectives

- Brief Introduction to Machine Learning
- Introduction to clustering
- What is dimensionality and why does it matter?
- Techniques like PCA (Principal Component Analysis)

Brief Introduction to Machine Learning

Machine learning is a subset of artificial intelligence (AI) that focuses on the development of systems that can learn from and make decisions based on data. Unlike traditional programming, where humans explicitly code the rules and logic for the computer to follow, machine learning enables computers to automatically learn and improve from experience without being explicitly programmed.

This learning process begins with feeding the machine learning algorithms data, which can be historical data or real-time inputs. These algorithms then analyze the data to find patterns or underlying structures. Based on these findings, the system can make predictions, decisions, or recommendations. Over time, as the system is exposed to more data, its performance improves, ideally becoming more accurate and efficient at its tasks.

Machine learning is widely used across various industries for a range of applications, including but not limited to:

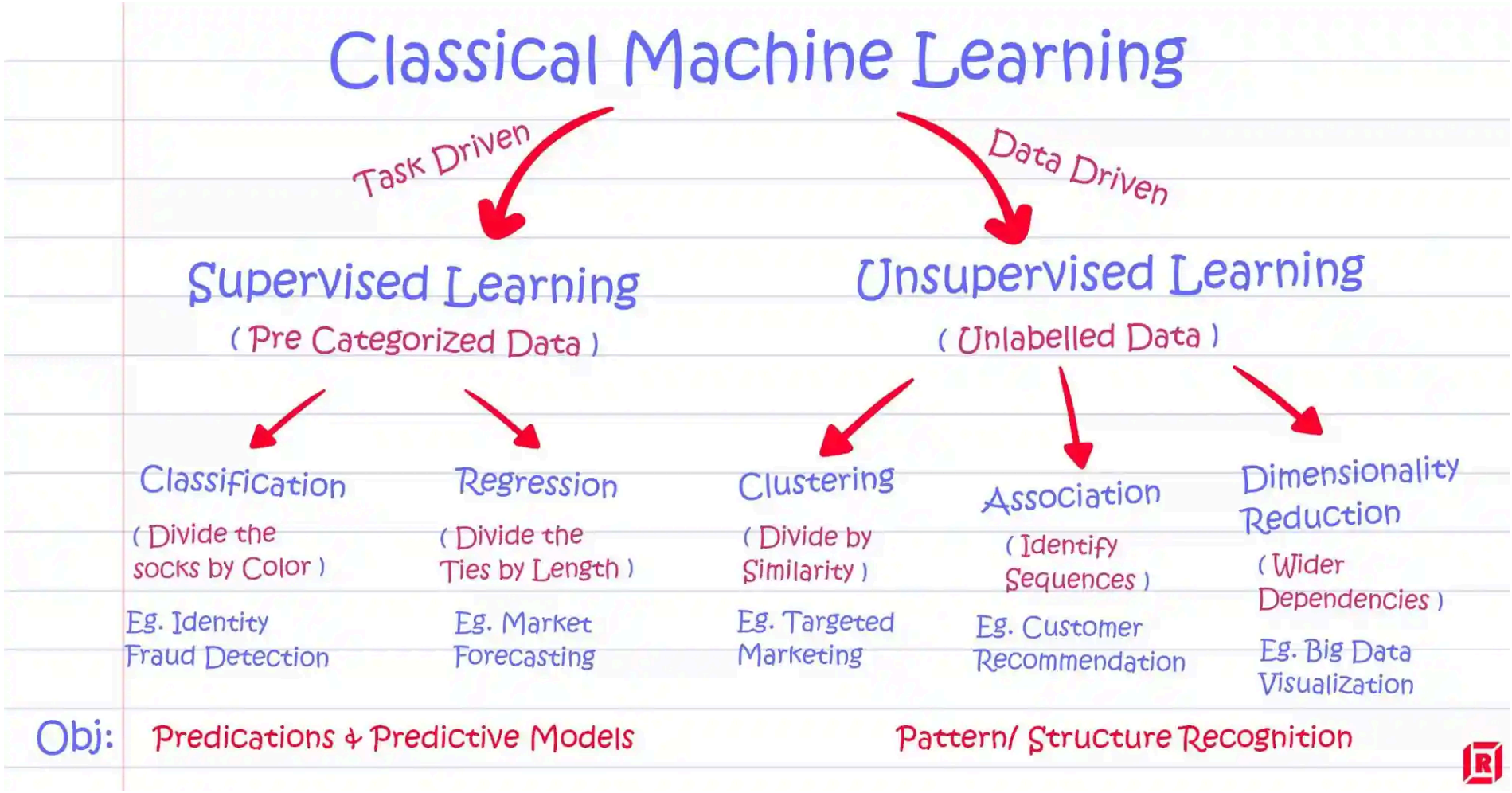
- Predictive analytics, such as forecasting sales or stock prices Image and speech recognition, as seen in facial recognition systems or virtual assistants like Siri and Alexa
- Natural language processing, which powers chatbots and translation services
- Anomaly detection, which helps identify fraudulent transactions in banking and finance
- Personalized recommendations, like those used by Netflix or Amazon to suggest content or products to users

Brief Introduction to Machine Learning (Continued)

There are several types of machine learning, including supervised learning, where the algorithm learns from labeled training data; unsupervised learning, where the algorithm learns from unlabeled data to find hidden patterns or intrinsic structures; and reinforcement learning, where an agent learns to make decisions by taking actions in an environment to achieve some goals.

Machine learning's ability to extract insights from data and automate decision-making processes makes it a crucial component of modern data analysis and artificial intelligence strategies.

Supervised vs. Unsupervised Learning



Credit: This figure is from https://www.labellerr.com/.

Clustering

Clustering in machine learning is a type of unsupervised learning technique that involves grouping a set of objects in such a way that objects in the same group, or cluster, are more similar to each other than to those in other clusters. The goal of clustering is to discover the inherent groupings in the data, such as grouping customers by purchasing behavior, documents by topics, or observations by their feature similarities.

Unlike supervised learning, where each data point is labeled with a predefined class, clustering algorithms must discover the patterns and groupings in the data on their own. This makes clustering particularly useful for exploratory data analysis (EDA), where the aim is to uncover unknown relationships in the data.

- Given set of rows
- Divide them into subsets of "similar" rows
- How to measure similarity?
- How to evaluate quality of results?

Clustering (Continued)

- **Given a set of points:** One should note that preprocessing steps such as normalization, handling missing values, and feature selection or extraction might be necessary to prepare the data for effective clustering.
- **Divide them into subsets of similar points:** The goal of this step is to maximize the similarity within clusters and minimize the similarity between different clusters. This can be achieved using various algorithms, each with its own method for forming these subsets.
- **How to measure similarity?** The choice of similarity (or distance) measure can significantly affect the results of clustering. Common measures include Euclidean distance, Manhattan distance, cosine similarity, and Jaccard similarity. The choice of measure might depend on the nature of our data and the specific requirements of the clustering algorithm.

Clustering (Continued)

- **How to evaluate the quality of results:** There are several methods to evaluate the quality of clustering results, including but not limited to:
  - `Silhouette Score:` Measures how similar an object is to its own cluster compared to other clusters. The silhouette score ranges from -1 to 1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.
  - `Davies-Bouldin Index:` The lower the Davies-Bouldin index, the better the clustering quality, as it implies a higher separation between clusters.
  - `Calinski-Harabasz Index:` Also known as the Variance Ratio Criterion, this index is higher for models with better defined clusters.

Additionally, it's important to note that the ultimate evaluation should also consider the domain-specific goals of the clustering. For example, in customer segmentation, beyond quantitative metrics, the clusters should offer actionable insights.

Clustering (Continued)

There are several algorithms for performing clustering, each with its own mechanisms and suitability for different types of data and applications. Some of the most widely used clustering algorithms include:

K-Means Clustering: This is one of the simplest and most popular clustering algorithms. It aims to partition the data into K predefined distinct non-overlapping clusters. It does so by minimizing the variance within each cluster. The algorithm iterates between assigning data points to the nearest cluster center and then moving each cluster center to the mean of its assigned points until convergence.

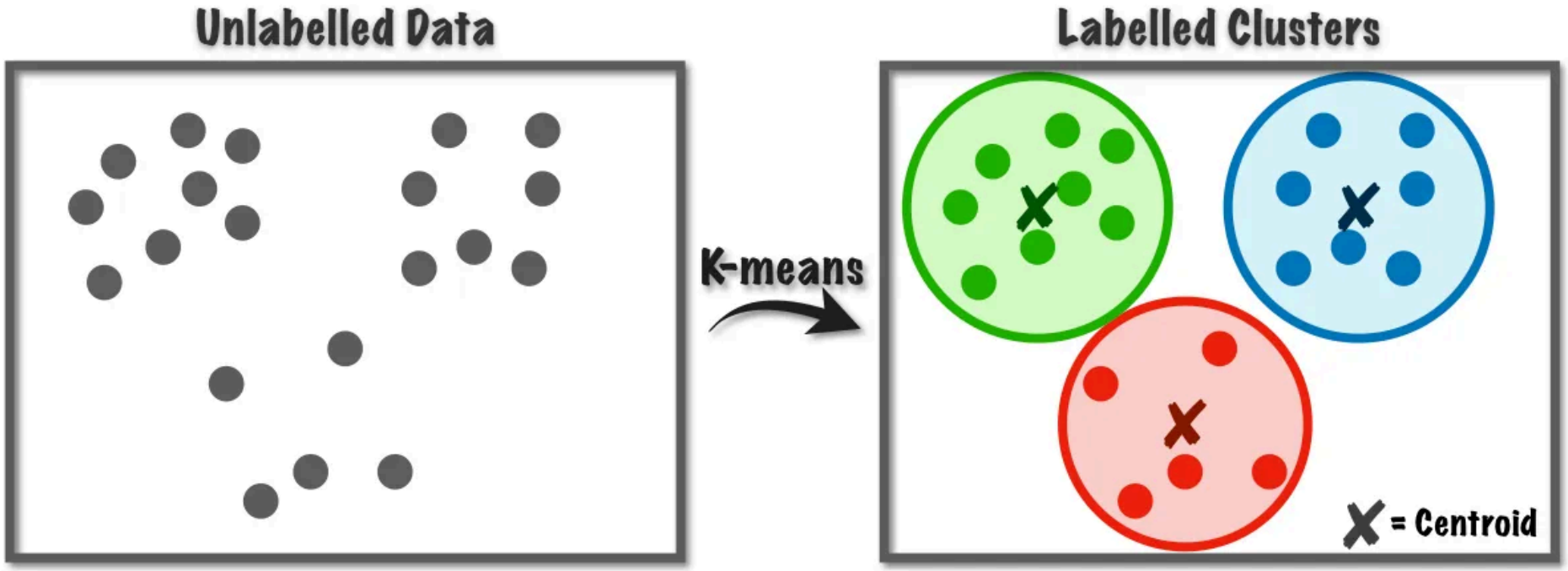
Hierarchical Clustering: This algorithm builds a hierarchy of clusters either agglomeratively (merging smaller clusters into larger ones) or divisively (splitting clusters). The result is a tree-like structure called a dendrogram, which shows the arrangement of the clusters and their proximity.

Mixture Models (Gaussian Mixture Models, GMM): A probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. GMMs use the expectation-maximization algorithm to infer the parameters of the Gaussian distributions. One of the benefits of mixture models, and GMM in particular, is the capability to model complex clusters that can be elliptical in shape, unlike K-means which assumes spherical clusters.

Clustering is applied in various domains, including market segmentation, social network analysis, search result grouping, medical imaging, and anomaly detection, among others. It helps in identifying the structure of data without prior knowledge of the number of clusters to be formed, providing valuable insights into the nature and grouping of the data.

K-means clustering

K-Means is a popular clustering algorithm that is used to partition data into K distinct, non-overlapping subsets or clusters. The goal is to minimize the variance within each cluster, making the data points in the same cluster as similar as possible.



Credit: This figure is from towardsdatascience.com.

K-means clustering (Continued)

How K-means works

- `Initialization:` Start by selecting K points in your dataset as the initial cluster centers. These can be chosen randomly or based on some heuristic.
- `Assignment Step:` Go through each data point (x_i) in your dataset, and assign it to the closest cluster center. The "closeness" is usually determined by the Euclidean distance between the data point and the cluster center.
- `Update Step:` Once all data points have been assigned to clusters, recalculate the position of each cluster center. This is done by taking the mean of all points assigned to that cluster, essentially finding the new center of gravity for each cluster.
- `Repeat:` Continue the assignment and update steps iteratively until the cluster centers no longer change significantly—this is referred to as convergence.
- `Result:` We end up with K clusters of data, each centered around a mean value, which can then be analyzed to uncover patterns or insights within each cluster.

Find an example here: http://shabal.in/visuals/kmeans/6.html.

K-means clustering (Continued)

Practical Applications in EDA

In the context of EDA, K-Means Clustering offers a powerful tool for:

- `Segmentation:` Identifying distinct groups within your data, such as customer segments based on purchasing behavior.
- `Pattern Recognition:` Uncovering patterns or trends within your dataset that might not be immediately apparent.
- `Anomaly Detection:` Identifying outliers or anomalies that do not fit well into any cluster could indicate errors or unusual behavior worth investigating further.
- `Feature Reduction:` Sometimes used in conjunction with other dimensionality reduction techniques to simplify datasets and make them more manageable for analysis.

K-means clustering (Continued)

K-means performs in different scenarios:

- **Clusters are Spherical:** K-means works best when the clusters are roughly spherical. Since the algorithm uses Euclidean distance to assign points to clusters, spherical clusters ensure that this distance metric effectively captures the notion of similarity. Non-spherical clusters might lead to less intuitive groupings.
- **Clusters are Well-Separated:** K-means tends to perform well when the clusters are well-separated from each other. This separation helps in clearly distinguishing between clusters, making the assignment of data points to their nearest center more accurate.
- **Clusters are of Similar Volumes:** The algorithm is more effective when the clusters have similar volumes. If clusters vary widely in volume, K-means might struggle, as it might incorrectly assign points from a smaller cluster to a larger one due to the way distances are computed.
- **Clusters Have Similar Numbers of Points:** K-means can yield better results when clusters have a similar number of points. If there's a significant imbalance in cluster sizes, the algorithm may favor creating clusters around larger groups of points, potentially overlooking smaller but meaningful clusters.

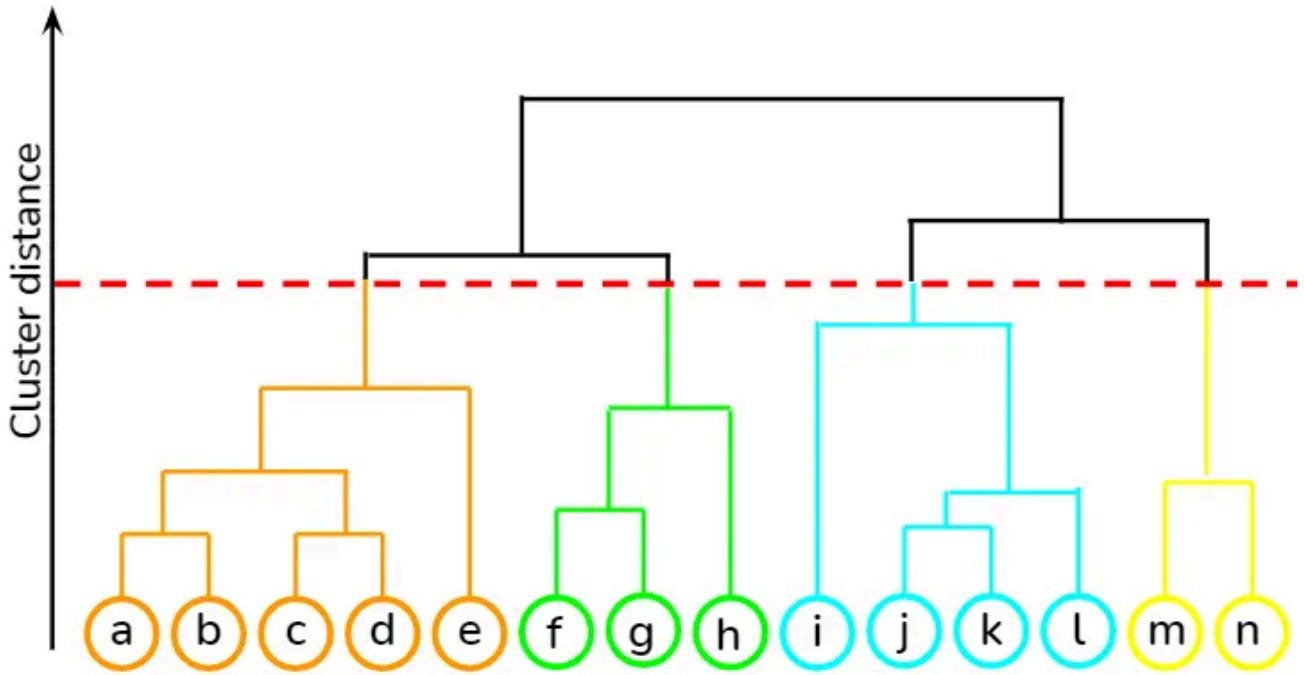
K-means clustering (Continued)

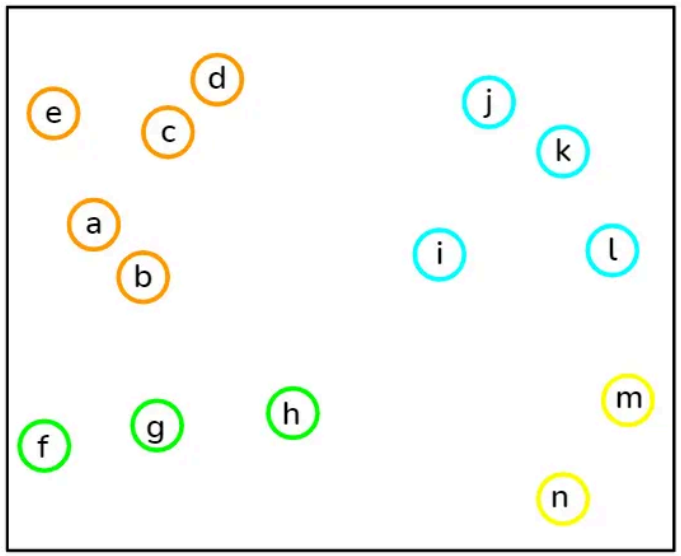
Iris Data Example

```
# Load the iris dataset
data(iris)

# Select only the numerical features for clustering
iris_features <- iris[,1:4]

# Perform K-Means clustering with 3 clusters
# (as we know there are 3 species)
set.seed(123) # Setting seed for reproducibility
kmeans_result <- kmeans(iris_features,
                        centers = 3,
                        nstart = 20)

# kmeans_result$cluster contains the cluster assignment for each flower
# Let's visualize the results with a plot for the first two dimensions

plot(iris_features[,1], iris_features[,2],
     col = kmeans_result$cluster,
     xlab = 'Sepal Length', ylab = 'Sepal Width',
     pch = 19, main = 'K-Means Clustering of Iris Data')
points(kmeans_result$centers[,1], kmeans_result$centers[,2],
       col = 1:3, pch = 8, cex = 2)
```

Hierarchical clustering

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Unlike K-Means clustering, which requires specifying the number of clusters beforehand, hierarchical clustering doesn't require that, making it particularly useful in exploratory data analysis when the number of clusters isn't known in advance.





Credit: These figures are from towardsdatascience.com

Hierarchical clustering (Continued)

How Hierarchical Clustering Works:

- `Start with Individual Points:` Initially, treat each data point as a single cluster. Thus, if there are N data points, you start with N clusters.
- `Find Closest Clusters and Merge:` At each step, identify the two clusters that are closest together and merge them into a single cluster. "Closeness" can be defined in several ways, such as the shortest distance between points in two clusters (single linkage), the longest distance (complete linkage), or the average distance (average linkage).
- `Repeat:` Repeat the process of finding and merging the closest clusters until all points are merged into a single cluster.
- `Resulting Dendrogram:` The result of hierarchical clustering can be represented as a dendrogram - a tree-like diagram that shows the arrangement of the clusters produced by the successive mergings.

Hierarchical clustering (Continued)

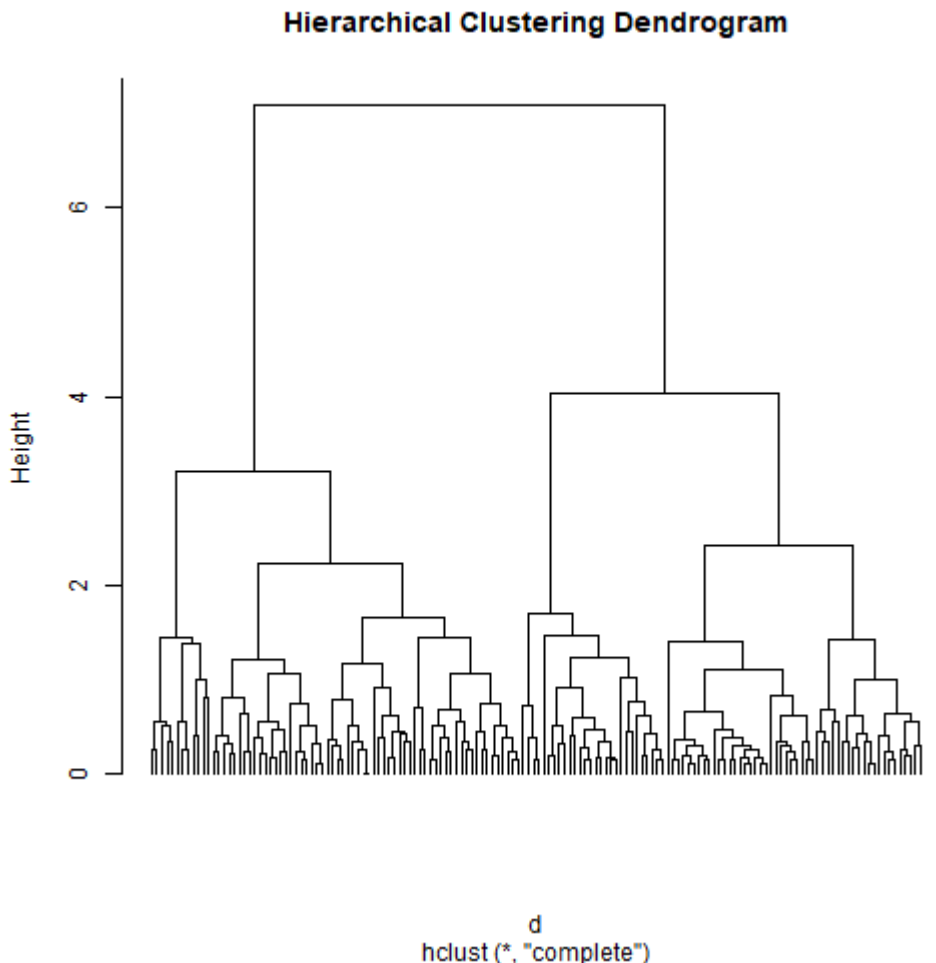
Scenarios Where Hierarchical Clustering is Useful

- **Unknown Number of Clusters:** Hierarchical clustering is ideal when the number of clusters is not known a priori, as it doesn’t require specifying the number of clusters at the start.
- **Non-spherical Clusters:** It can handle non-spherical clusters better than K-Means since it builds clusters based on distances without assuming a particular shape.
- **Discovering Hierarchies:** It is useful in cases where the data naturally forms hierarchical structures and you're interested in uncovering these hierarchies, such as in taxonomy creation, organizational structures, or product categorizations.

Hierarchical clustering (Continued)

Iris Data Example

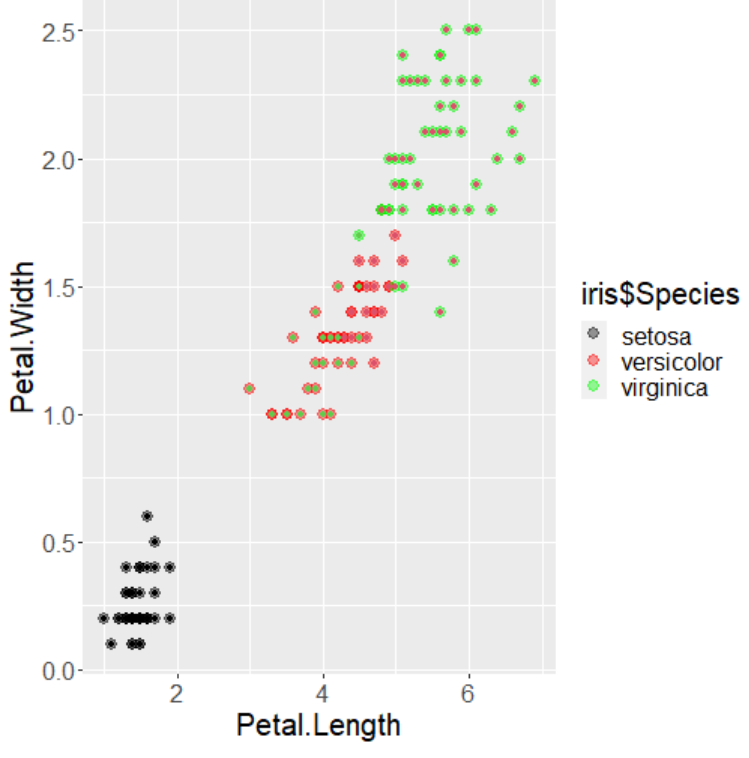
```
# Load the iris dataset
data(iris)
iris_features <- iris[,1:4]

# Perform Hierarchical Clustering
## Compute the distance matrix
d <- dist(iris_features)
# Apply hierarchical clustering
hc <- hclust(d, method = "complete")

# Plot the dendrogram
plot(hc, labels = FALSE, hang = -1,
     main = "Hierarchical Clustering Dendrogram")
```



Hierarchical clustering (Continued)

```
#  cut off the tree at the desired number of clusters
clusterCut <- cutree(hc, 3)

# compare it with the original species
table(clusterCut, iris$Species)

ggplot(iris,
       aes(Petal.Length, Petal.Width,
           color = iris$Species)) +
  geom_point(alpha = 0.4, size = 3.5) +
  geom_point(col = clusterCut) +
  scale_color_manual(
    values = c('black', 'red', 'green')) +
  theme(text=element_text(size=20))
```

```
##
## clusterCut setosa versicolor virginica
##          1     50          0         0
##          2      0         23        49
##          3      0         27         1
```



Hierarchical clustering (Continued)

Unique Advantages of Hierarchical Clustering for EDA:

- **Interpretable Dendrogram:** Hierarchical clustering produces a dendrogram, a tree-like diagram that shows the arrangement of the clusters produced by the hierarchical clustering. This can be particularly useful for understanding the data structure, exploring how clusters are related, and determining the appropriate number of clusters by visually inspecting the dendrogram.
- **No Need to Specify Number of Clusters Ahead of Time:** Unlike K-means, which requires you to specify the number of clusters at the beginning, hierarchical clustering doesn't require this. The hierarchical nature allows us to explore different levels of granularity from the dendrogram, making it flexible for exploratory analysis where we might not know the best number of clusters beforehand.
- **Finding Non-Linearly Separable Structures:** Hierarchical clustering can identify clusters with non-linear boundaries, making it suitable for complex datasets where the true structure isn't globular and might not be well-captured by K-means.

Hierarchical clustering (Continued)

Practical Applications in EDA where Hierarchical Clustering is Particularly Useful:

- **Gene Expression Analysis:** In bioinformatics, hierarchical clustering is used to analyze gene expression data, grouping genes with similar expression patterns. The dendrogram can help biologists understand functional relationships among genes.
- **Customer Segmentation:** While K-means is also used for customer segmentation, hierarchical clustering can offer more nuanced insights into customer behavior patterns. The dendrogram provides a visual representation of customer groups at various levels of similarity, which can be particularly insightful for tailoring marketing strategies.
- **Document Clustering:** Hierarchical clustering is well-suited for grouping documents with similar topics in natural language processing tasks. The hierarchy can reveal topic subcategories at different levels, useful for organizing large collections of text documents.
- **Market Research:** Identifying segments within a market based on consumer preferences or behaviors. The hierarchical structure allows businesses to understand the relationships between different market segments at a granular level.

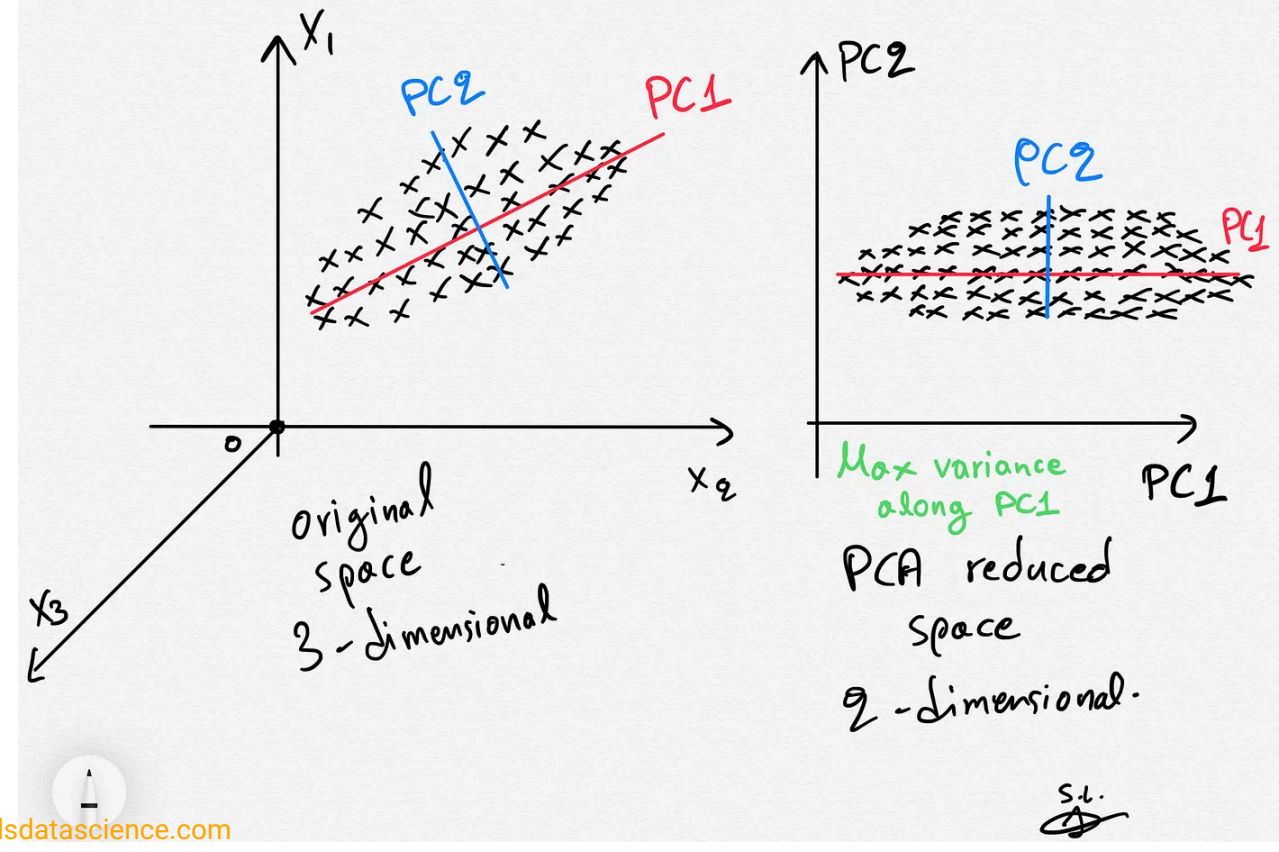
What is dimensionality and why does it matter?

Definition of Dimensionality: Refers to the number of variables (features) in a dataset. High-dimensional data can pose challenges for analysis due to the curse of dimensionality.

Curse of Dimensionality: A phenomenon where the feature space becomes increasingly sparse with the addition of more dimensions, making it difficult to analyze data effectively due to increased computational complexity and the risk of overfitting.

Principal component analysis (PCA)

PCA transforms the original variables into a new set of variables, the principal components (PCs), which are orthogonal (uncorrelated), ensuring that the first few retain most of the variation present in the original dataset. This makes PCA exceptionally useful for reducing the dimensionality of data, improving visualization, and even assisting in the preprocessing steps for other machine learning algorithms.



Credit: This figure is from towardsdatascience.com

Principal component analysis (Continued)

Steps of PCA:

- `Standardization:` Often, the first step involves standardizing the data so that each feature contributes equally to the analysis.
- `Covariance Matrix Computation:` Calculates how changes in one variable are associated with changes in another variable, helping identify the directions in which the data varies the most.
- `Eigenvalue and Eigenvector Calculation:` From the covariance matrix, PCA finds the eigenvectors (directions of maximum variance) and their corresponding eigenvalues (magnitude of the variance in those directions).
- `Sort Principal Components:` The eigenvectors are sorted by decreasing eigenvalues to prioritize the principal components that capture the most variance.
- `Projection:` The original data is projected onto the principal components, resulting in a transformed dataset with reduced dimensions.

Principal component analysis (Continued)

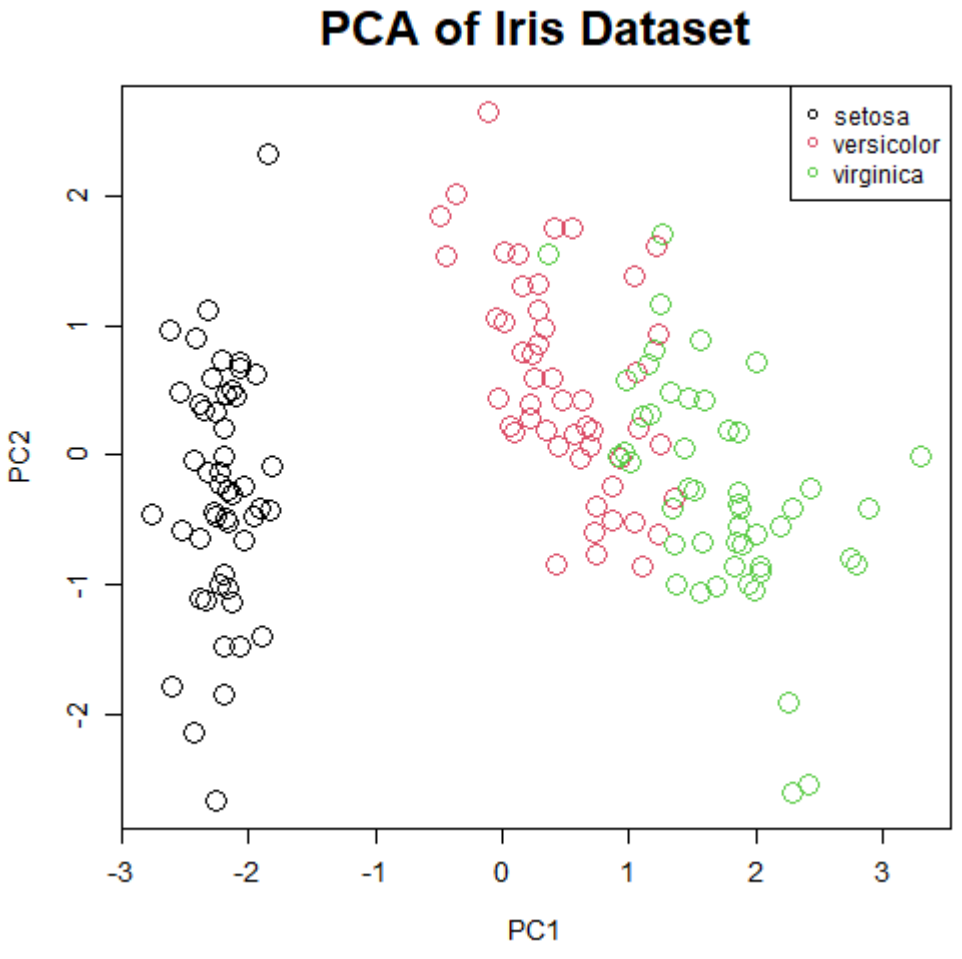
Practical Applications of PCA in EDA:

- **Data Visualization:** For high-dimensional data, PCA is often used to reduce dimensions to two or three principal components, making it possible to visualize complex datasets in 2D or 3D plots.
- **Feature Reduction:** In machine learning, too many variables can lead to model overfitting. PCA helps by reducing the number of features while retaining the essential information, improving model accuracy and computational efficiency.
- **Noise Reduction:** By keeping only the principal components with significant variance and discarding those with noise, PCA can enhance the signal-to-noise ratio in the data.
- **Discovering Patterns:** The projection of data into principal components can reveal hidden patterns that were not obvious in the original high-dimensional space.

Principal component analysis (Continued)

Example in R using Iris Data for PCA:

```
# Load the iris dataset
data(iris)

# Standardizing the data
## Excludes the species column
iris_std <- scale(iris[, -5])

# Perform PCA
pca_result <- prcomp(iris_std,
                     center = TRUE,
                     scale. = TRUE)

# Plot the first two principal components
plot(pca_result$x[,1:2],
     col = iris$Species,
     xlab = "PC1", ylab = "PC2",
     main = "PCA of Iris Dataset")
legend("topright",
       legend = unique(iris$Species),
       col = 1:length(unique(iris$Species)),
       pch = 1)
```



Multidimensional scaling (MDS)

MDS is a technique used to analyze similarity or dissimilarity data. It aims to position each object in a lower-dimensional space such that the between-object distances reflect the given similarities or dissimilarities as closely as possible. MDS is versatile, applicable to various types of data, including those without explicit spatial coordinates.

Reference: R-bloggers

Multidimensional scaling (Continued)

Steps of MDS:

- `Distance Matrix Computation:` Calculate a matrix representing the distances or dissimilarities between every pair of objects in the dataset.
- `Dimensionality Reduction:` Use the distance matrix to project the data into a lower-dimensional space, typically through optimization techniques that seek to preserve the original distances as much as possible.
- `Visualization:` The lower-dimensional representation can be visualized, helping to interpret the relationships and structures within the data.

Practical Applications of MDS in EDA:

- **Market Research:** Visualizing consumer preferences to understand market segments based on perceived similarities or differences in products.
- **Social Network Analysis:** Mapping the relationships within social networks to uncover communities or clusters.
- **Psychology and Cognitive Science:** Analyzing similarity judgments about concepts or stimuli to understand underlying cognitive or perceptual spaces.

Isomap

Isomap is a nonlinear dimensionality reduction method designed to uncover the underlying structure of data that lies on a curved manifold within the higher-dimensional space. By maintaining geodesic distances between all pairs of points, Isomap seeks to preserve the global geometry of the data, making it suitable for datasets where the relationships are not linear.

Reference: Sklearn — Isomap

Isomap (Continued)

Steps of Isomap:

- `Neighborhood Graph Construction:` Identify the nearest neighbors for each point, either based on a fixed number of neighbors (k-nearest) or within a fixed radius. This step constructs a graph reflecting the local neighborhood of each point.
- `Geodesic Distance Calculation:` Estimate the geodesic distances between all pairs of points using the constructed graph, typically by computing the shortest paths.
- `Multidimensional Scaling (MDS):` Apply classical MDS to the matrix of geodesic distances, projecting the data into a lower-dimensional space while trying to preserve the manifold's global geometric properties.

Practical Applications of Isomap in EDA:

- **Image Processing:** Unraveling the structure in image data where images lie on a manifold within a high-dimensional space. Speech Recognition: Understanding the complex structure of audio data for better feature extraction.
- **Biological Data Analysis:** Exploring the genetic or protein expression data where the intrinsic data structure is complex and nonlinear.

References

The lectures of this course are based on the ideas from the following references.

- Exploratory Data Analysis by John W. Tukey
- A Course in Exploratory Data Analysis by Jim Albert
- The Visual Display of Quantitative Information by Edward R. Tufte
- Data Science for Business: what you need to know about data mining and data-analytic thinking by Foster Provost and Tom Fawcett
- Storytelling with Data: A Data Visualization Guide for Business Professionals by Cole Nussbaumer Knaflic