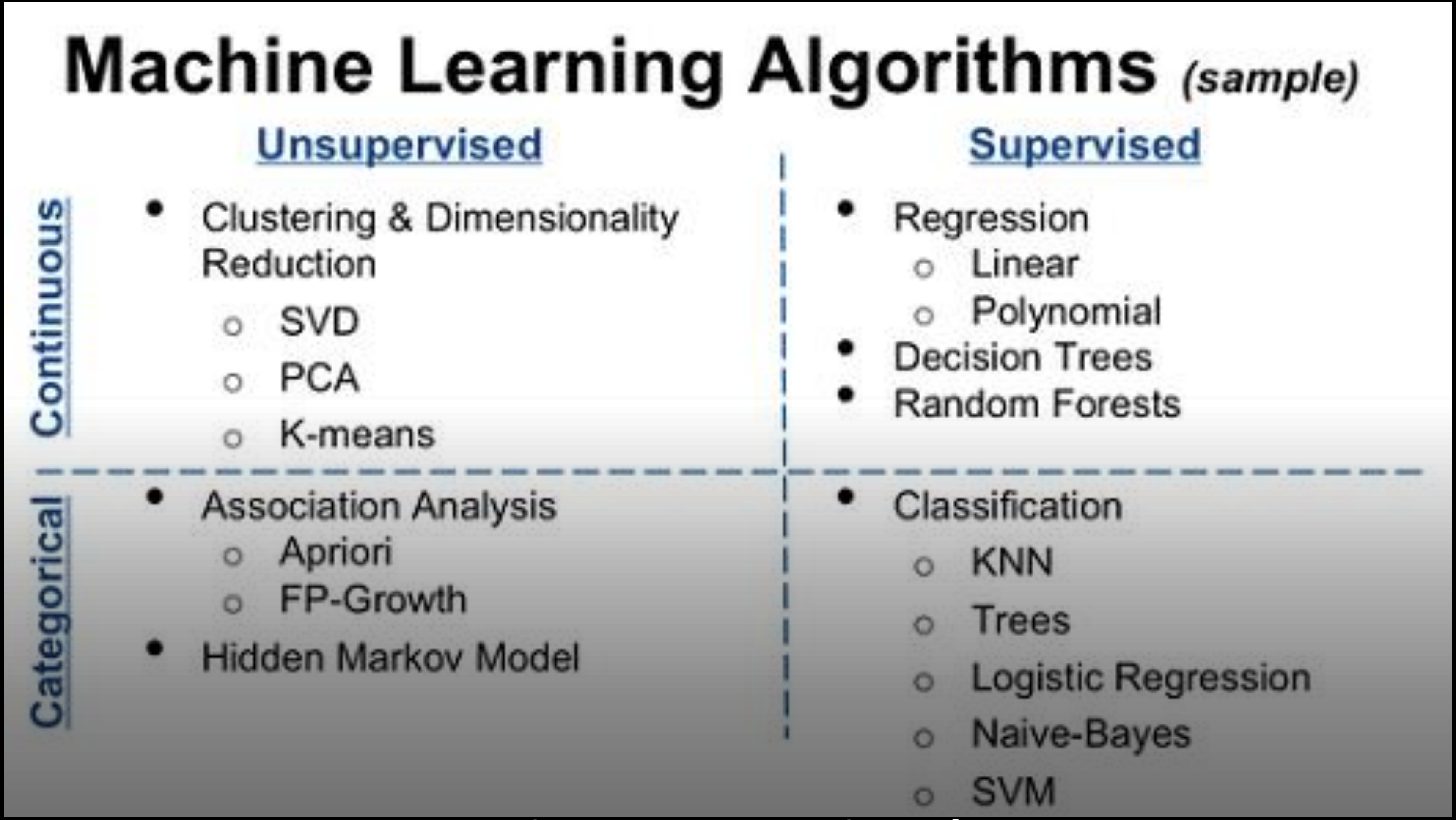
INTRODUCTION TO MACHINE LEARNING

DATA 602
Lecture 8

Dr. Tony Diana
tonydian@umbc.edu

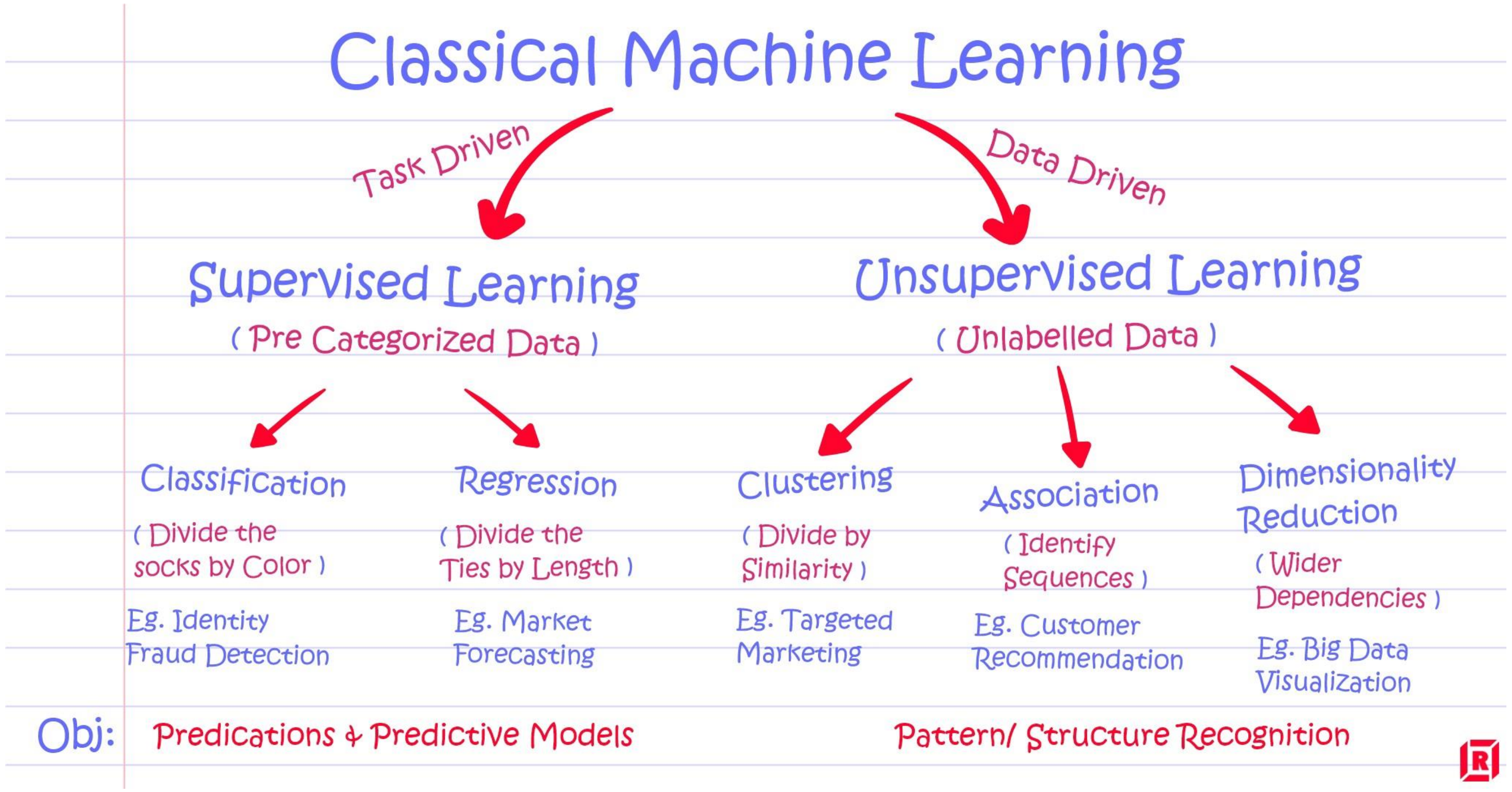


Machine Learning Algorithms *(sample)*

	Unsupervised	Supervised
Continuous	• Clustering & Dimensionality Reduction o SVD o PCA o K-means	• Regression o Linear o Polynomial • Decision Trees • Random Forests
Categorical	• Association Analysis o Apriori o FP-Growth • Hidden Markov Model	• Classification o KNN o Trees o Logistic Regression o Naive-Bayes o SVM

Unsupervised Models

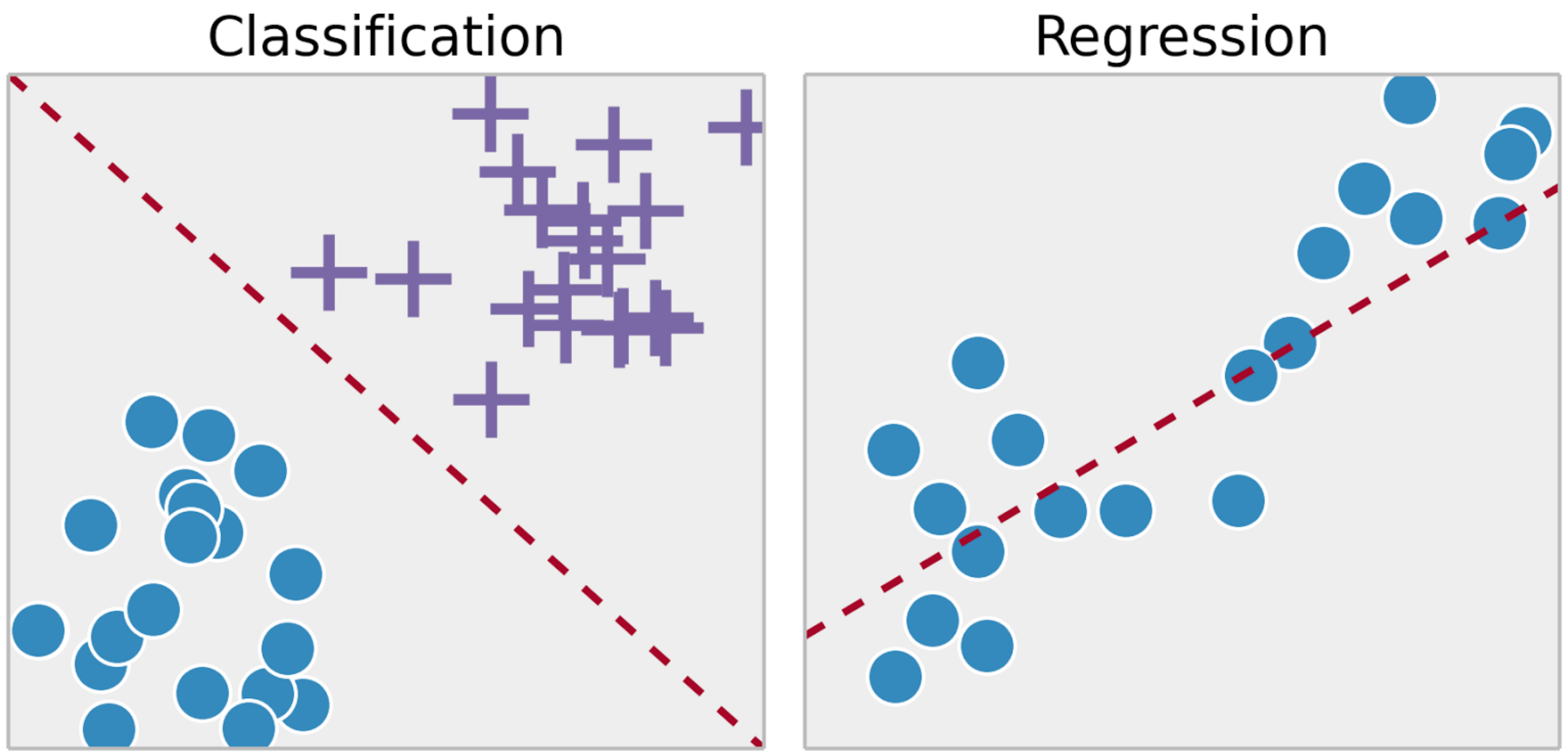
Supervised v. Unsupervised Learning Methods (Cont.)



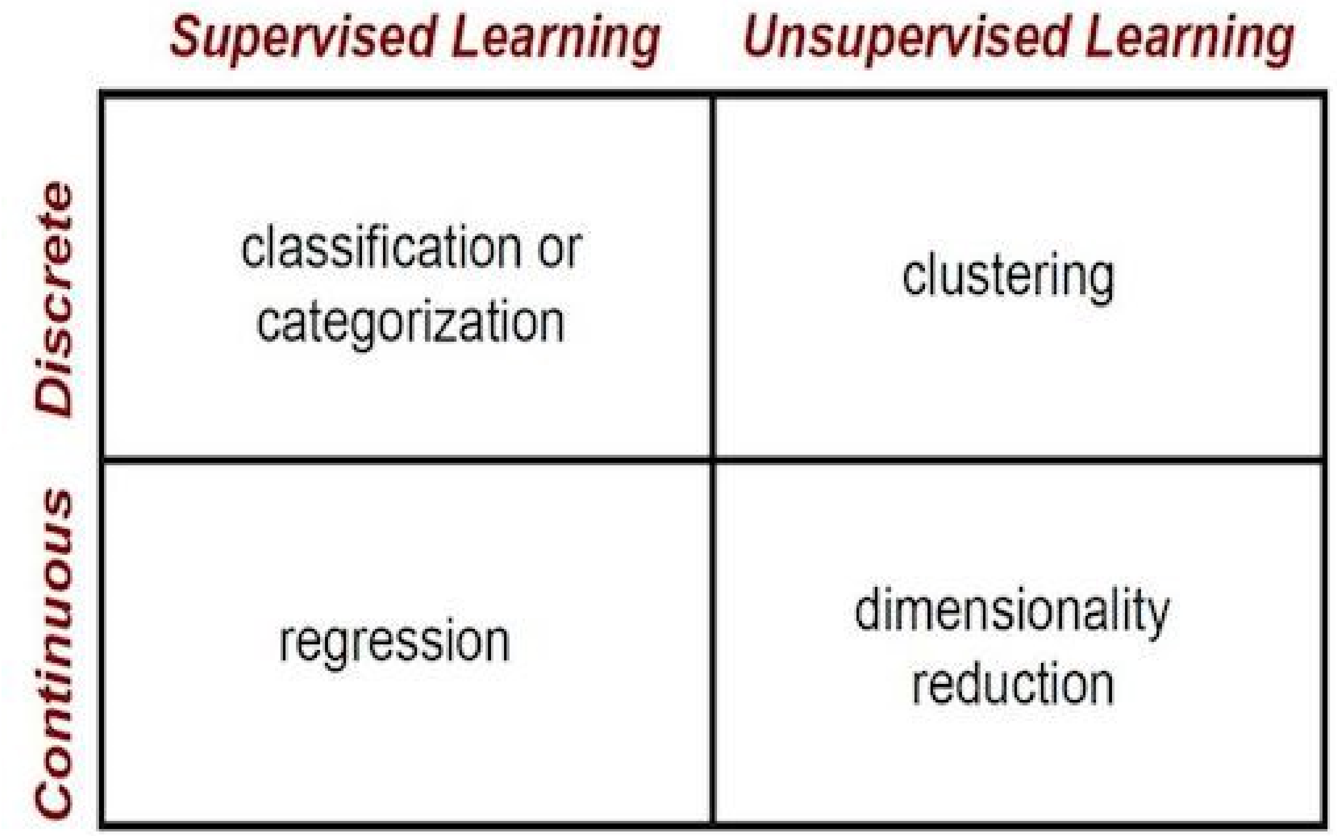
Supervised v. Unsupervised Learning

- In **supervised learning**, we have prior knowledge of what the output values for our samples should be
  - The goal of **supervised learning** is **to learn a function** that, given a sample of data and desired outputs, **best approximates** the **observable relationship between input and output** in the data
- In **unsupervised learning**, we do not have labeled outputs
  - There is no predefined category of labels available for a target variable
  - Unsupervised models can help **reduce dimensions** (i.e., Principal Component Analysis)
  - Unsupervised models can also help **infer the natural structure existing** within a set of data points by creating a category or label based on patterns available in data
  - There are three types of algorithms:
    - **Clustering**
    - **Dimensionality Reduction**
    - **Association**

Supervised Learning: Difference between Classification and Regression



Supervised v. Unsupervised Learning by Type of Data

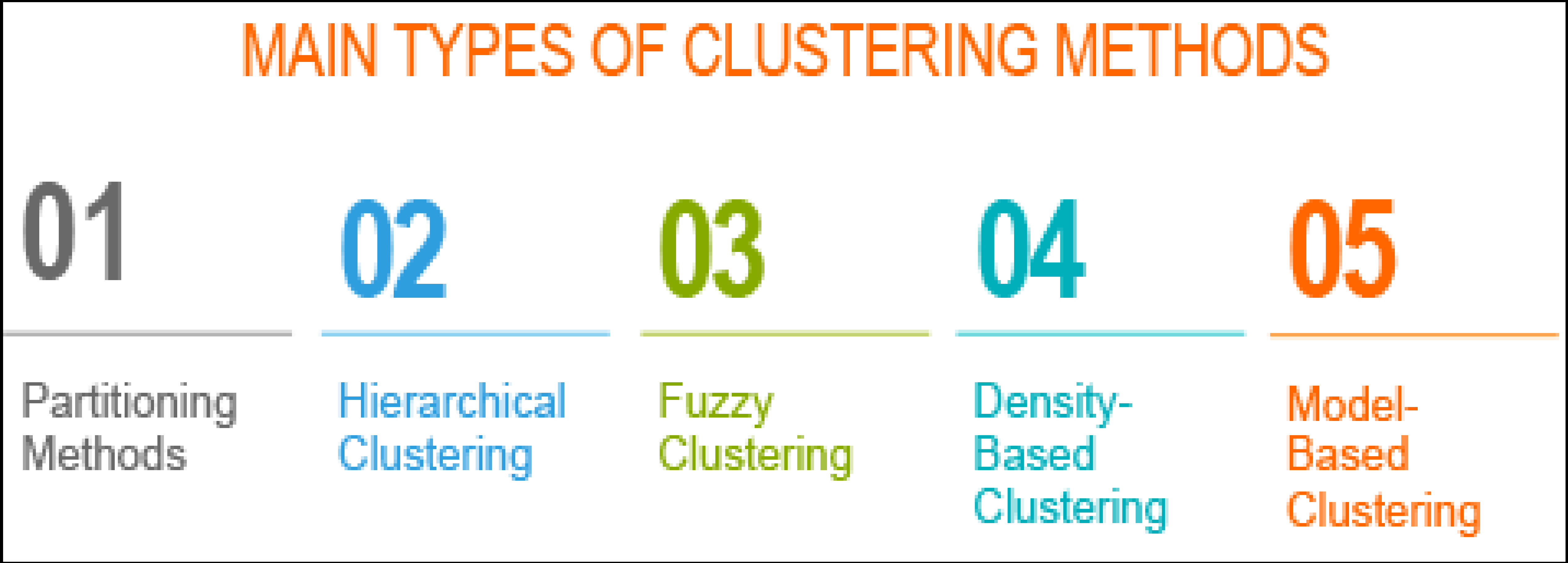


	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Examples of Unsupervised Learning

Examples of where unsupervised learning methods might be useful:

- An advertising platform segments the U.S. population into smaller groups with similar demographics and purchasing habits so that advertisers can reach their ‘target market’ with relevant ads
- Airbnb groups its housing listings into ‘neighborhoods’ so that users can navigate listings more easily
- Looking at the patterns of transactions, a bank can determine fraud through the identification of out-of-bound clusters



MAIN TYPES OF CLUSTERING METHODS

01 Partitioning Methods

02 Hierarchical Clustering

03 Fuzzy Clustering

04 Density-Based Clustering

05 Model-Based Clustering

Clustering Models

Clustering

- **Clustering** is an **unsupervised learning problem**
- **Clustering analysis** can be traced to the areas of anthropology and psychology in the 1930s
- The key objective is **to identify distinct groups** called **clusters** based on some notion of similarity within a given dataset
- **Clustering** is a Machine Learning technique that involves the grouping of data points
- Given a set of data points, we can use a clustering algorithm to classify each data point into a specific group
- In theory, data points in the same group should have similar properties and/or features, while data points in different groups should have highly dissimilar properties and/or features
- Most popular clustering techniques are **K-Means** (divisive) and **hierarchical** (agglomerative)

Types of Clustering

One of the primary uses of clustering algorithms is data exploration as we try to understand data

- **Partitioning Clustering**: It organizes the objects of a dataset into many groups or clusters. The specification of the number of clusters or k is the start of the process (i.e. K-Means)
- **Fuzzy Clustering**: Objects can belong to several clusters with varied membership degrees. Such portioning is also called 'soft' partitioning. In **Fuzzy K-Means clustering**, data points are allocated to many clusters with different degrees of membership
- **Hierarchical Clustering**: In this type of clustering, data is organized in a hierarchical manner such that clusters have sub-clusters. Hierarchical clustering can be **agglomerative** or **divisive**
- **Density-Based Clustering**: a cluster in a data space is a contiguous region of high point density, separated from other such clusters by contiguous regions of low point density (i.e. DBSCAN, HDBSCAN, OPTICS)
- **Model-Based Clustering**: Clusters are the outcomes of a model which tries to optimize the fit between the data and the model
    - The data are viewed as coming from a mixture of probability distributions, each of which represents a different cluster
    - The clusters to be derived from cluster analysis are distributions over the data space, which can be represented using probability density functions. The samples come from a mixture of K normal (Gaussian) distributions

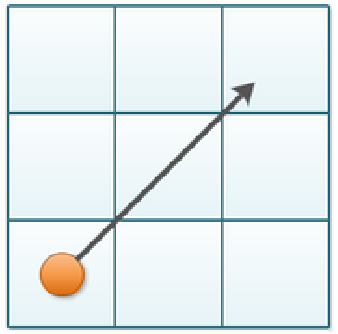
UMBC

What are the Different Types of Distance?

The distance from a centroid to data points determine the size of the cluster. Among the most widely used:

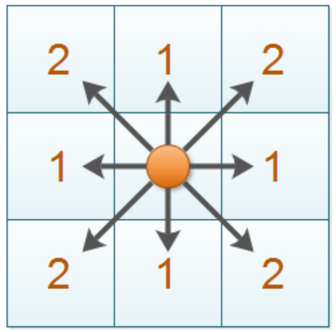
- **Euclidean Distance**
  - It is the straight-line distance between two points in Euclidean space
- **Manhattan Distance**
  - The Manhattan distance between two vectors (or points) *a* and *b* is defined as $\sum_i |a_i - b_i|$ over the dimensions of the vectors
- **Chebyshev Distance**
  - It is defined on a vector space where the distance between two vectors is the greatest of their differences along any coordinate dimension

Euclidean Distance



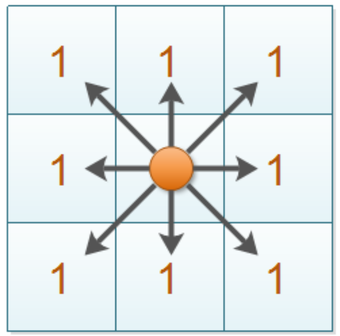
$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Manhattan Distance



$$|x_1 - x_2| + |y_1 - y_2|$$

Chebyshev Distance



$$\max(|x_1 - x_2|, |y_1 - y_2|)$$

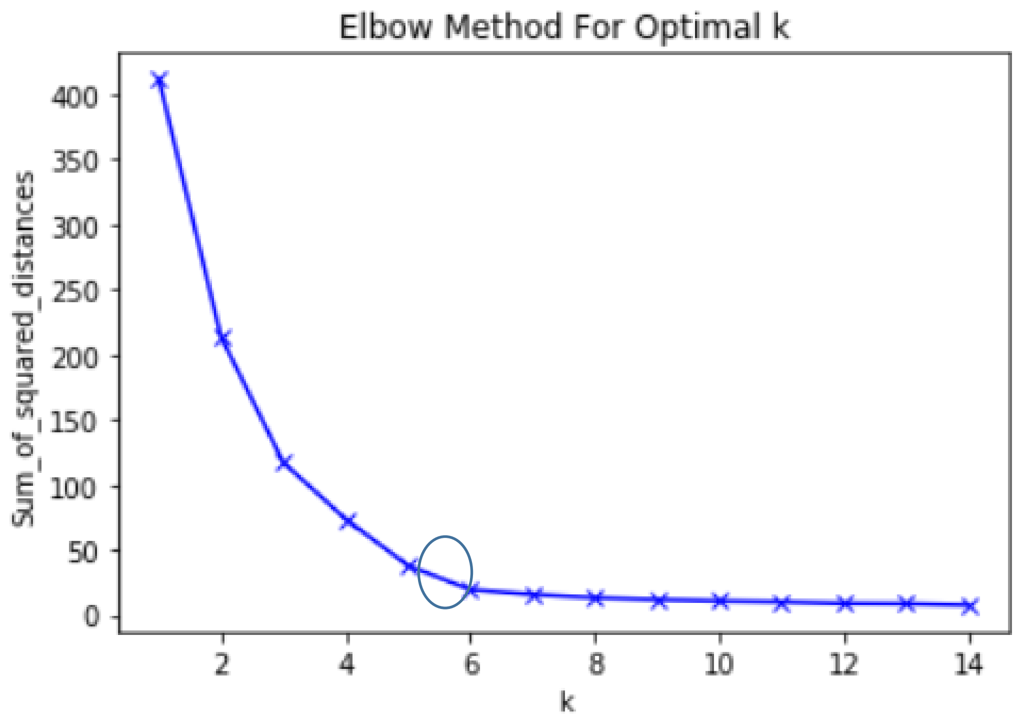
What are the Objectives of K-Means?

- To group similar data points together
- To discover underlying patterns
- To achieve these objectives, **K-Means** looks for a fixed number (*k*) of clusters in a dataset
  - A cluster refers to a **collection of data points aggregated together because of certain similarities**

Methodology

- Determine a **target number *k***, which refers to the **number of centroids** you need in the dataset
  - A ***centroid*** is the imaginary or real location representing the center of the cluster
  - Every data point is allocated to each of the clusters through **reducing the in-cluster sum of squares**
- Allocate every data point to the nearest cluster, while keeping the centroids as small as possible
- The ***'means'*** in **'K-Means'** refers to **averaging of the data** (finding the centroid)
- To process the learning data, the **K-Means** algorithm starts with a first group of randomly selected centroids-- which are used as the beginning points for every cluster-- and then performs iterative (repetitive) calculations to optimize the positions of the centroids
- It stops creating and optimizing clusters when either:
  - The **centroids have stabilized** — there is no change in their values because the clustering has been successful
  - The **defined number of iterations has been achieved**

Selecting k with the Elbow Method



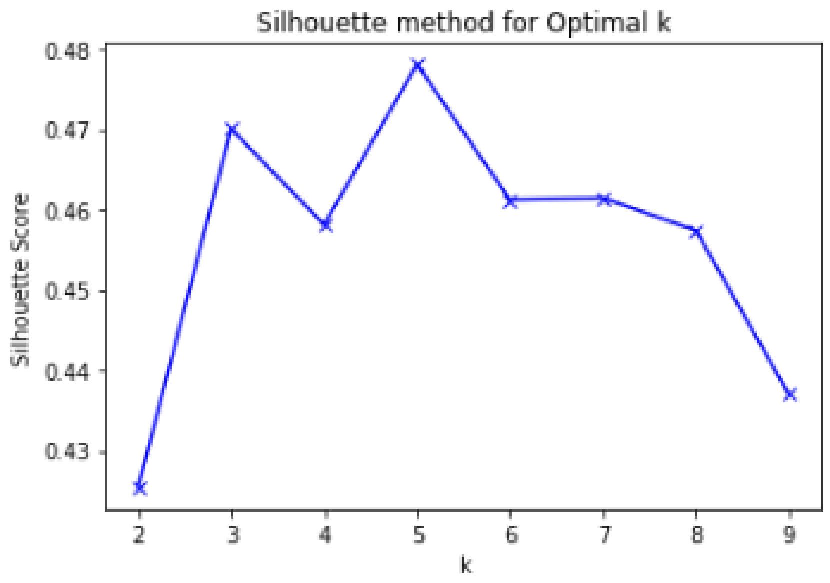
- The ’**elbow’ method** is used to determine the optimal number of clusters in k-means clustering
- The ’**elbow’ method** plots the value of the cost function produced by different values of k
- As k increases, the sum of squared distance tends to zero
- The basic idea behind this method is that it plots the various values of cost when changing **k**
- As the value of **k** increases, there will be fewer elements in the cluster
- If the line chart looks like an arm, then the ‘**elbow**’ (the point of inflection on the **curve**) is the best value of k
- The ‘arm’ can be either up or down, but if there is a strong inflection point, it is a good indication that the underlying model fits best at that point

Selecting k with the Silhouette Score

- The **Silhouette Score** is a measure of goodness of fit that helps decide the number of clusters to be formulated from the data
- The **Silhouette Score = (x - y)/ max(x, y)**

where **y** is the mean intra-cluster distance (the mean distance to the other instances in the same cluster), **x** is the mean nearest-cluster distance (the mean distance to the instances of the next closest cluster)

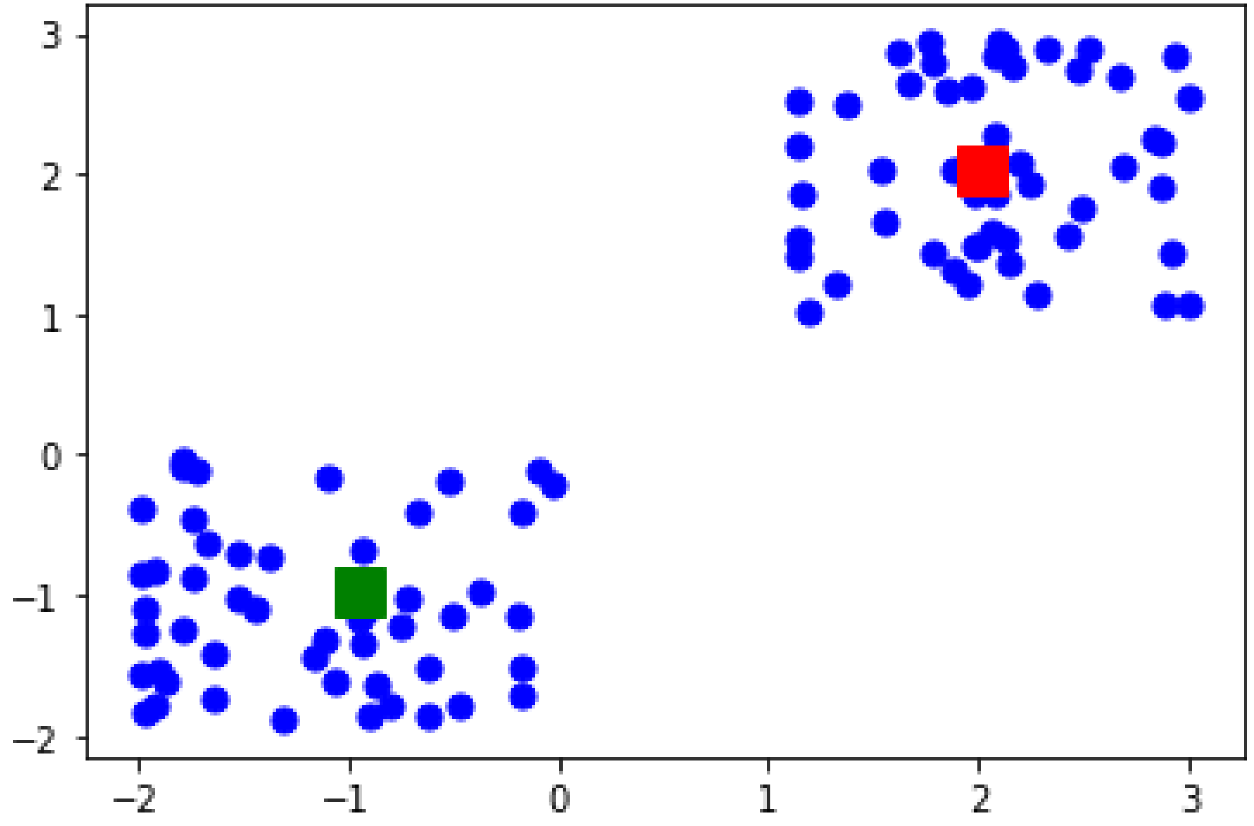
- The coefficient varies between -1 and 1
  - A value close to 1 implies that the instance is close to its cluster and is a part of the right cluster
  - A value close to -1 means that the value is assigned to the wrong cluster



The **silhouette** graph shows that k = 5 is a better choice than k = 3

Example of K-Means Programming

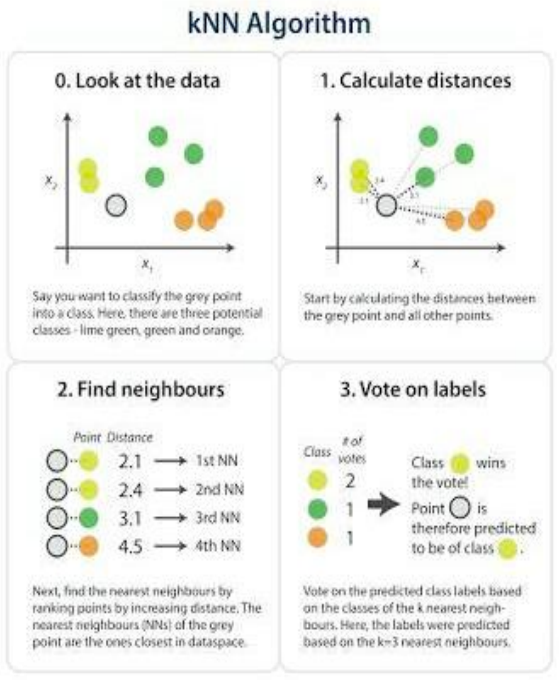
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
%matplotlib inline
# Generate Random Data
X= -2 * np.random.rand(100,2)
X1 = 1 + 2 * np.random.rand(50,2)
X[50:100, :] = X1
plt.scatter(X[ : , 0], X[ :, 1], s = 50, c = 'b')
plt.show()
from sklearn.cluster import KMeans
Kmean = KMeans(n_clusters=2)
Kmean.fit(X)
# Find the Centroid
Kmean.cluster_centers_
plt.scatter(X[ : , 0], X[ : , 1], s =50, c='b')
plt.scatter(-0.94665068, -0.97138368, s=200, c='g',
marker='s')
plt.scatter(2.01559419, 2.02597093, s=200, c='r',
marker='s')
plt.show()
# Test the Algorithm
Kmean.labels_
sample_test=np.array([-3.0,-3.0])
second_test=sample_test.reshape(1, -1)
Kmean.predict(second_test)
```



UMBC

K-Nearest Neighbor (KNN)

- **KNN** is a **non-parametric** algorithm
- K refers to ‘n_neighbors’ in the KNN function
    - It does **not make any underlying assumptions about the distribution of data**
    - KNN algorithm can also be used for regression problems
        - Regression problems will be using **averages of nearest neighbors** rather than **voting** from nearest neighbors
    - In KNN algorithm output is a **class membership**
    - An object is assigned a class which is most common among its **K-nearest neighbors**
        - K is the **number of neighbors** and it is intuitively always a **positive integer**
    - KNN stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions)



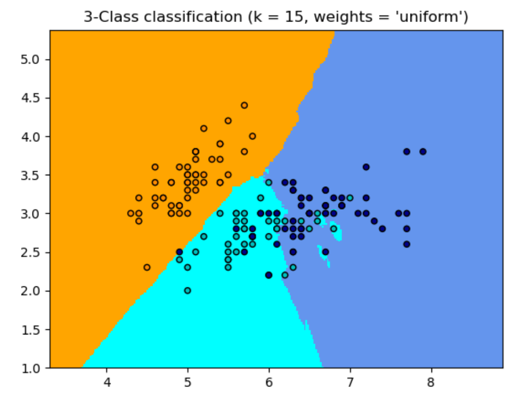
Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

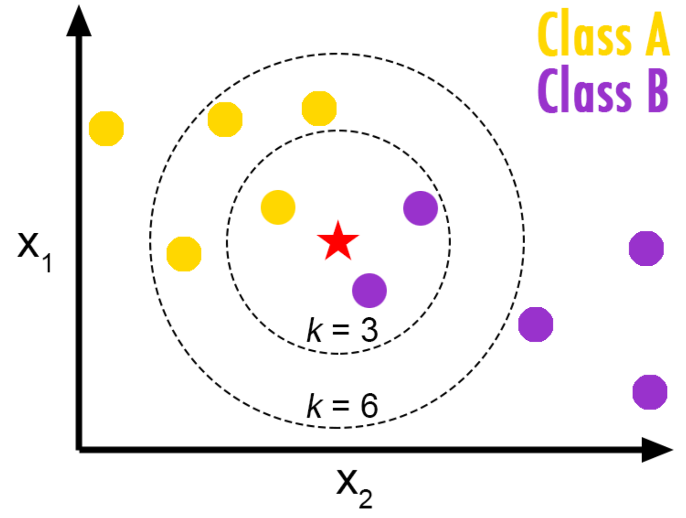
Start by calculating the distances between the grey point and all other points.

Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

Class ● wins the vote! Point ○ is therefore predicted to be of class ●.

Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.





KNN: Applications | Pros and Cons

Applications

- In **text analysis**, it is used in searching for semantically similar documents
    - Finding documents containing similar topics
- Setting up a **Recommender System**
    - If you know what users like, then you can recommend *similar* items for them
- **KNN** has also been used for **economic forecasting**, **data compression**, and **genetics**

Pros

- **No assumptions** about data
- **Simple algorithm** to explain and understand/interpret
- Overall **good accuracy**
- **Versatile** — useful for classification or regression

Cons

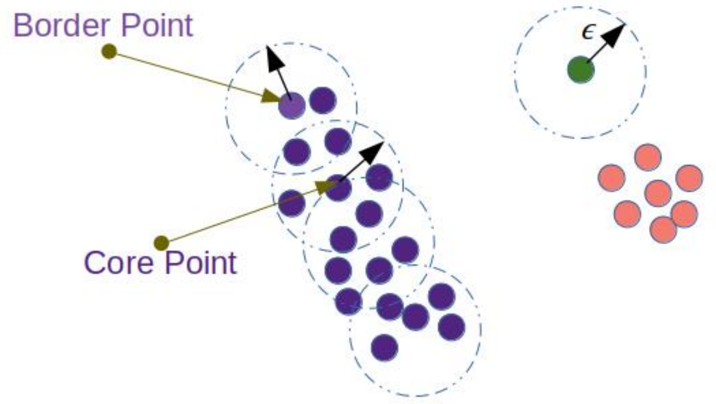
- **Computationally expensive** — because the algorithm stores all the training data
- **High memory requirement**
- **Prediction stage** might be **slow** with large datasets
- May be **sensitive to features not relevant** in the classification and to the **scale** of the data

What is Density-Based Spatial Clustering of Applications with Noise?

- DBSCAN **groups** together points that are **close to each other** based on a **distance measurement** (usually Euclidean distance) and a **minimum number of points**
- The main concept of DBSCAN algorithm is **to locate regions of high density** that are **separated from one another** by **regions of low density**
- It also marks as **outliers** the points that are in **low-density regions**
- **DBSCAN** requires two parameters:
    - **Epsilon (ε)**: specifies how close points should be to each other to be considered a part of a cluster. It means that, if the distance between two points is **lower or equal to this value** (eps), these points are considered **neighbors**
        - Density at a point P is the number of points within a circle of radius epsilon *(ϵ)* from point P
        - The eps should be chosen based on the **distance** of the dataset (we can use a k-distance graph to find it), but in general small epsilon values are preferable
    - **minPoints**: the minimum number of points to form a dense region. For example, if we set the minPoints parameter as 5, then we need at least 5 points to form a dense region
        - The minimum value for the minPoints must be 3, but the larger the data set, the larger the minPoints value that should be chosen
        - For each point in the cluster, the circle with radius eps contains at least minimum number of points
- The DBSCAN algorithm should be used to find **associations** and **structures** in data, which can be hard to find manually, but can be relevant and useful to find patterns and predict trends

What are the Benefits of Density-Based Spatial Clustering of Applications with Noise?

- It requires minimum domain knowledge
- It can discover clusters of arbitrary shape
- It works well with large samples of data

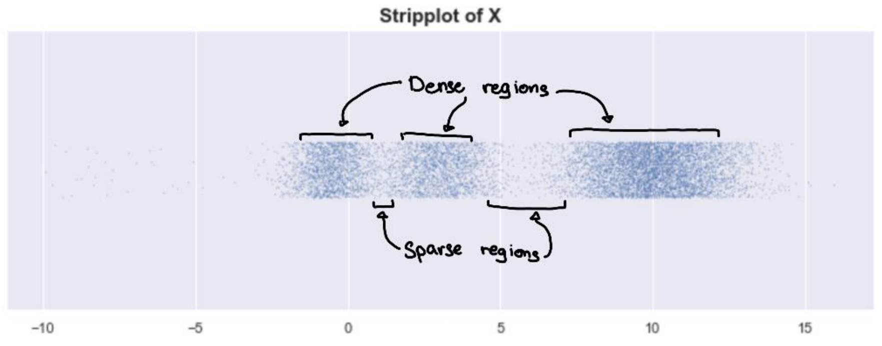


$$N_{Eps}(p) = \{q \in D \text{ such that } dist(p,q) \leq \epsilon\}$$

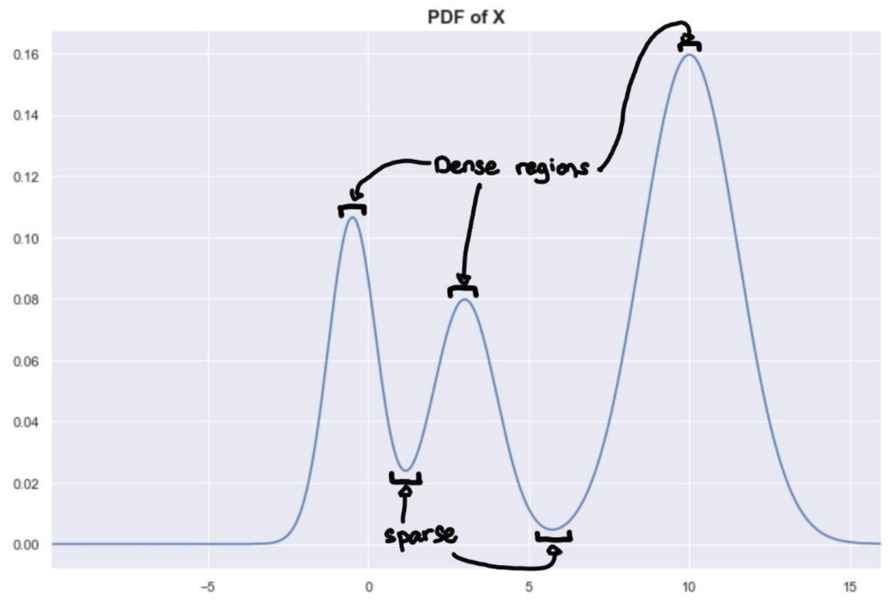
ϵ = 1 unit, MinPts = 7

HDBSCAN

- **Hierarchical DBSCAN** is a clustering algorithm developed by Campello, Moulavi, and Sander (2013, 2015)
- It extends **DBSCAN** by converting it into a hierarchical clustering algorithm, and then using a technique to extract a flat clustering based in the stability of clusters
- Rather than looking for clusters with a particular shape, it looks for regions of the data that are denser than the surrounding space
- It is a non-parametric method that looks for a cluster hierarchy shaped by the multivariate modes of the underlying distribution



- Identify the clusters (3 in this case)
- Look at the underlying distribution through the probability density function of X
- The peaks are the densest regions and the troughs are the sparsest regions
- Highly dense regions are separated by sparse regions
- Highly probable regions are separated by improbable regions
- The probability density function serves to determine the clusters



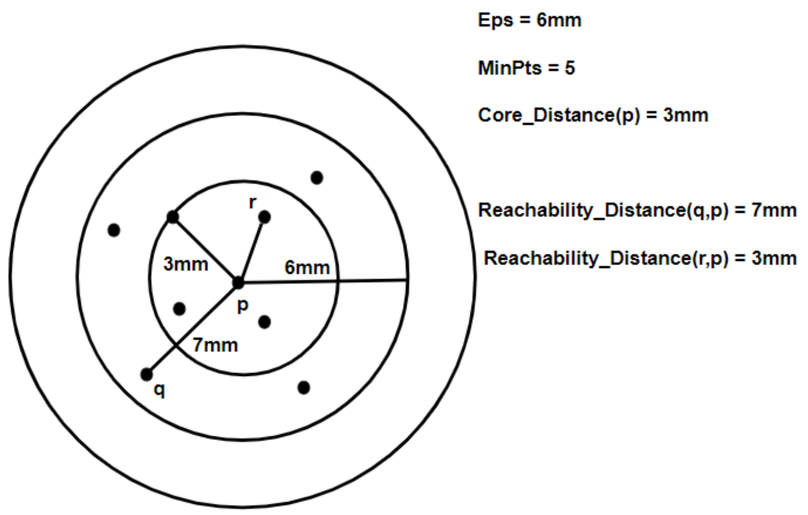
OPTICS ***(Ordering Points to Identify the Clustering Structure)***

- The more difficult parameter for **DBSCAN** is the radius. In some cases, the parameter will not be obvious, or you might need multiple values. That is when **OPTICS** comes into play
- **OPTICS** addresses one of **DBSCAN**'s major weaknesses: **the problem of detecting meaningful clusters in data of varying density**

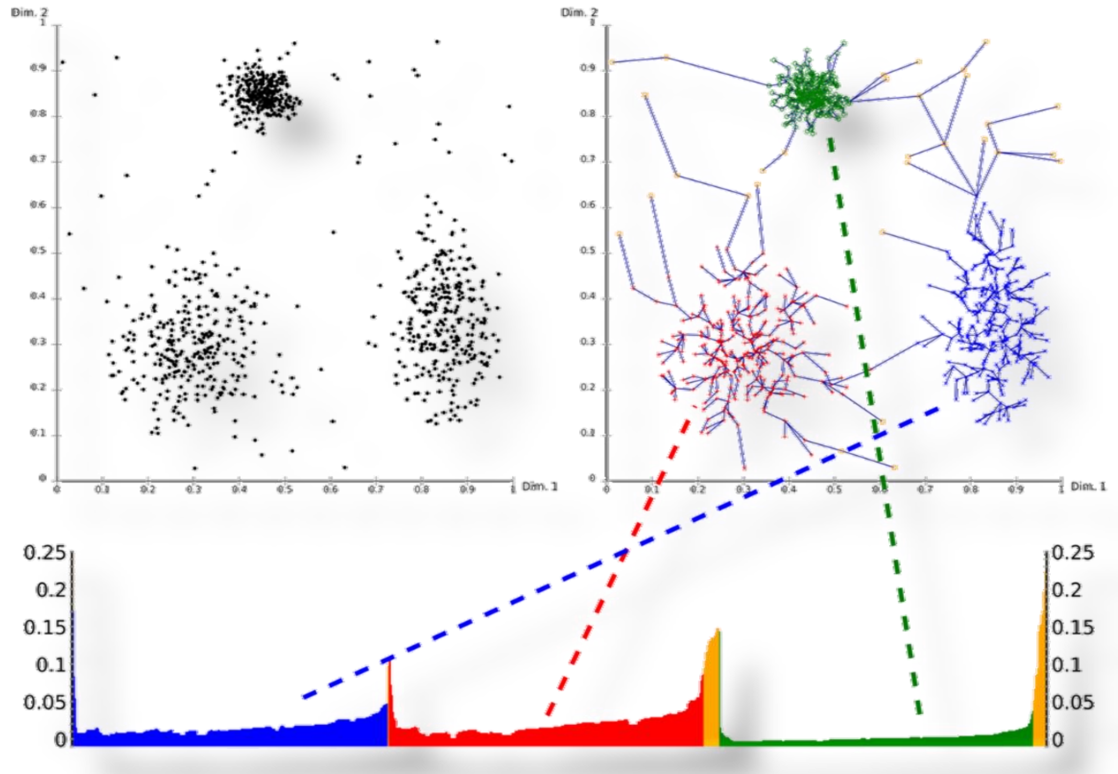
- The points of the database are (linearly) ordered such that spatially closest points become neighbors in the ordering
- A special distance is stored for each point that represents the density that must be accepted for a cluster so that both points belong to the same cluster
- Instead of fixing **MinPts** and the **radius**, we only deal with **MinPts**, and plot the **radius** at which an object would be considered dense by **DBSCAN**
- The **OPTICS** clustering technique requires more memory than **DBSCAN**
- **OPTICS** maintains a priority queue (Min Heap) to determine the next data point which is closest to the point currently being processed in terms of **Reachability Distance**
- **OPTICS** requires more computational power because the nearest neighbor queries are more complicated than radius queries in DBSCAN
- **OPTICS** does not segregate the given data into clusters: It merely produces a **Reachability Distance** plot and it is upon the interpretation of the analyst to cluster the points accordingly
- Ankers, Breuning, Kriegel, and Sanders presented the OPTICS model in 1999

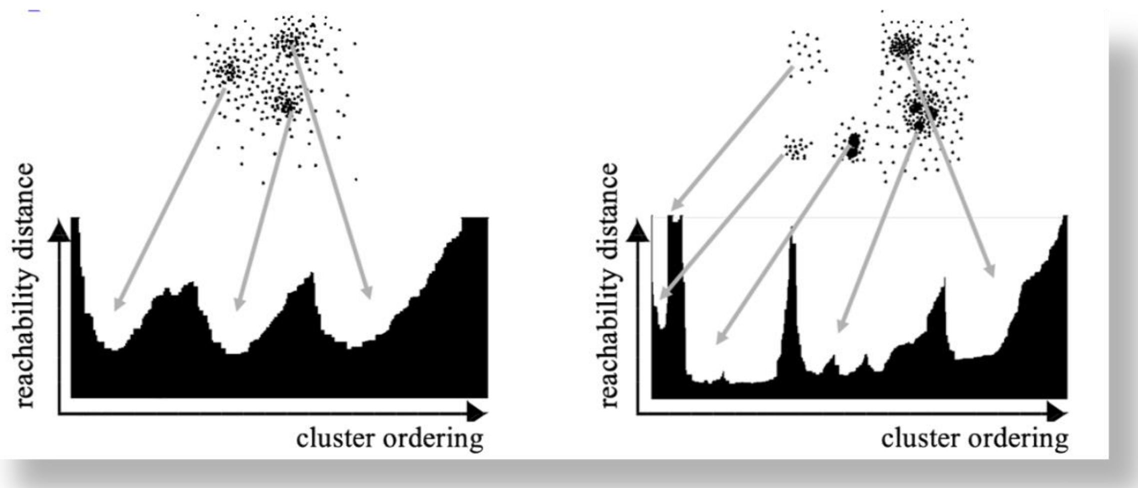
OPTICS : Key Concepts

- **Core Point**. A point *p* is a **core point** if at least **MinPts** points are found within its ***ε*-neighborhood**
- **Core Distance.** It is the minimum value of radius required to classify a given point as a **core point**
- If a given point is not a core point, then its core distance is undefined
- **Reachability Distance.** It is defined with respect to another data point **q**
    - The reachability distance between a point **p and q** is the **maximum** of the **core distance of p** and the **distance between p and q**
    - Note that the reachability distance is not defined if q is not a core point


Eps = 6mm
MinPts = 5
Core_Distance(p) = 3mm
Reachability_Distance(q,p) = 7mm
Reachability_Distance(r,p) = 3mm
r
3mm
6mm
p
7mm
q

OPTICS: Reachability Plot





- Using a ***reachability-plot*** (a special kind of dendrogram), the hierarchical structure of the clusters is a 2D plot, **with the ordering of the points as processed by OPTICS on the x-axis** and **the reachability distance on the y-axis**
- Since the **points belonging to a cluster have a low reachability distance to their nearest neighbor**, the clusters show up as valleys in the reachability plot. **The deeper the valley, the denser the cluster**

Agglomerative Clustering

- **What is Agglomerative Clustering?**
- **Agglomerative Clustering** is one of the most common hierarchical clustering techniques
- The clustering technique assumes
  - Each data point is **similar** enough to the other data points
  - Data can be aggregated into 1 cluster
- Bottom-up algorithms like **agglomerative** clustering treat each data as a **singleton cluster** at the outset and then successively **agglomerates pairs of clusters** until all clusters have been merged into a **single cluster** that contains all data
- The top-down approach is called **divisive** clustering

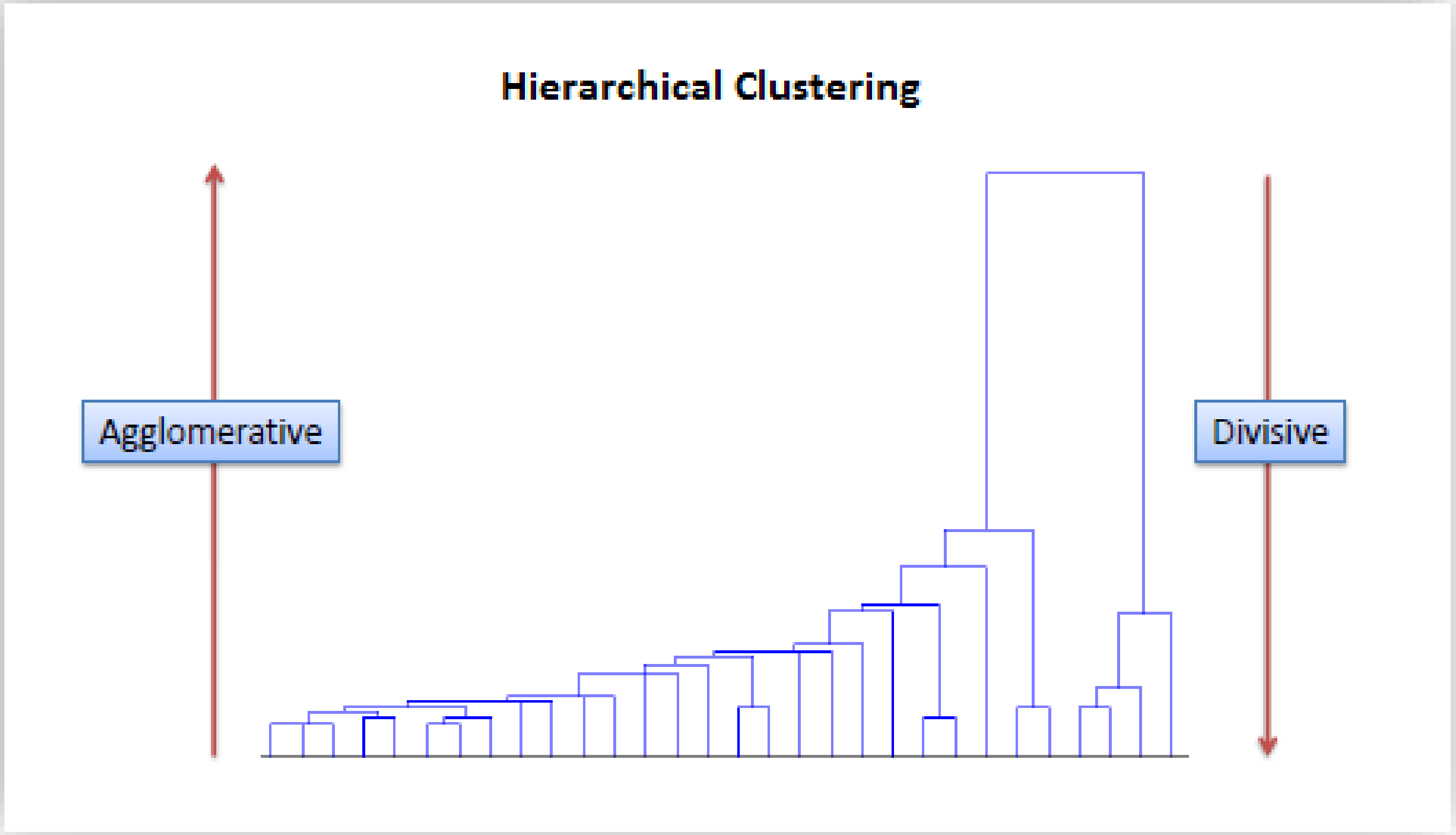


Illustration of Dendrogram

Dimensionality Reduction

What is Principal Component Analysis?

- **Principal Component Analysis** or **PCA** is a **dimensionality-reduction method** that transforms a large set of variables into a smaller one that still contains most of the information of the large set
- Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity
- The objectives of **PC**A is to **reduce the number of variables** of a data set, while **preserving as much information as possible**
- **PCA** finds the **direction of maximum variance** in **high-dimensional data** such that most of the information is retained, and projects it onto a smaller dimensional subspace

Principal Component Analysis (Cont.)

Methodology

- **Standardize**
  - The aim of this step is to **standardize the range of the continuous initial variables** so that each one of them contributes equally to the analysis
  - This can be done by **subtracting the mean and dividing by the standard deviation** for each value of each variable
- **Covariance Matrix Computation**
  - The aim of this step is **to understand how the variables of the input data set are varying from the mean** with respect to each other, or in other words, to see if there is any relationship between them
  - The sign of the covariance matters:
    - If **positive**, then the two variables increase or decrease together (**correlated**)
    - If **negative**, then one increases when the other decreases (**inversely correlated**)
- **Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components**
  - Principal components are **new variables constructed as linear combinations or mixtures of the initial variables**
  - These combinations are done in such a way that the new variables (i.e. principal components) are uncorrelated and most of the information within the initial variables is squeezed or compressed into the first components
    - **Principal components** represent the **directions of the data** that explain a **maximum amount of variance**

Principal Component Analysis (Cont.)

How are Principal Components Constructed?

- The first principal component accounts for the **largest possible variance** in the data set
- The second principal component is calculated in the same way, with the condition that it is *uncorrelated* with (i.e. perpendicular to) the first principal component and that it accounts for the next highest variance
- This continues until a total of *p principal components* have been calculated, equal to the original number of variables

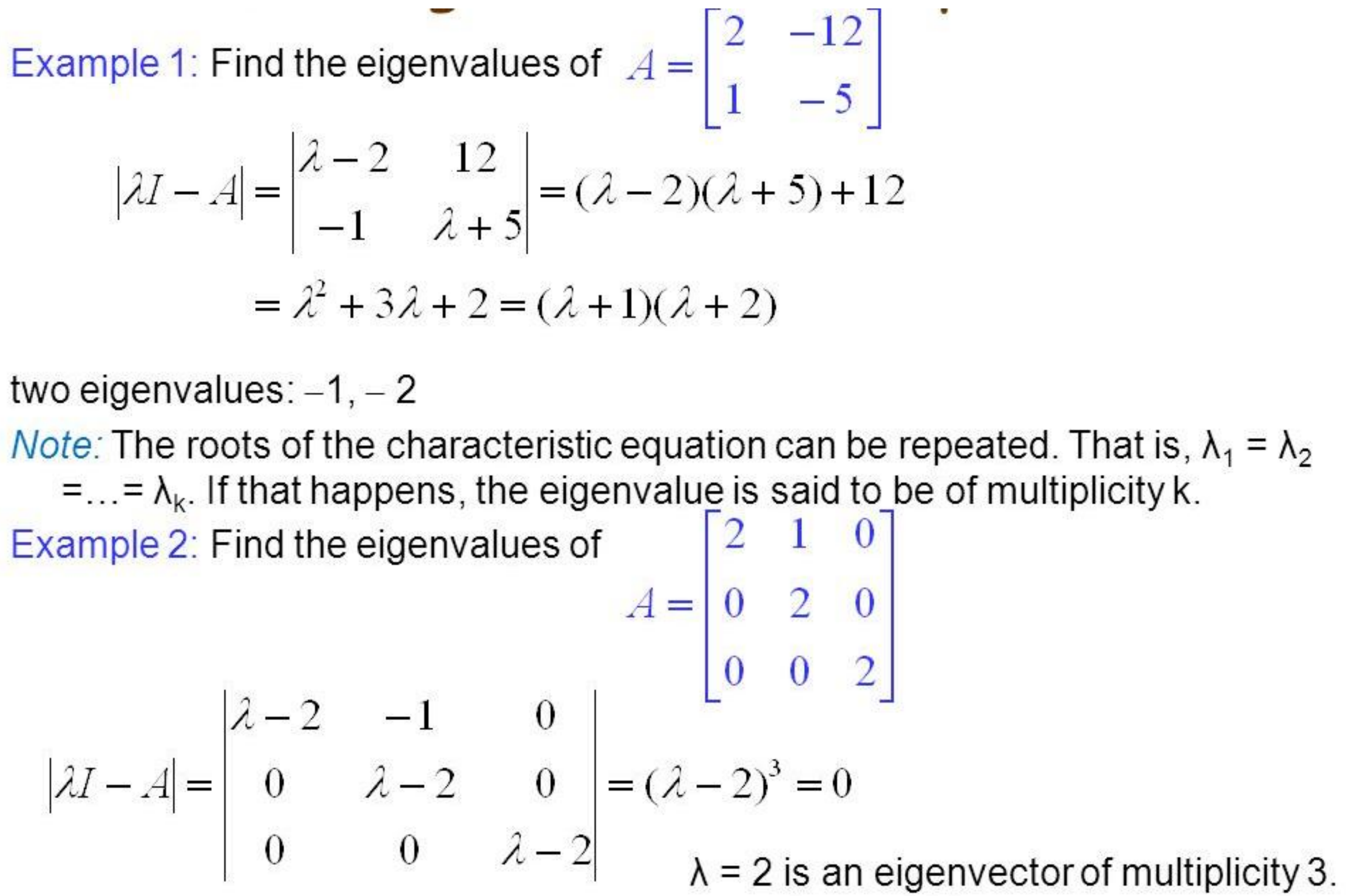
What about Eigenvalues?

- Every eigenvector has an eigenvalue.
- Their number is equal to the number of dimensions of the data
  - For example, for a 3-dimensional data set, there are 3 variables, therefore there are 3 eigenvectors with 3 corresponding eigenvalues
  - The **eigenvectors of the covariance matrix** are the **directions of the axes** where there is the most **variance** (most information)
  - We call the **eigenvectors** the **principal components**
  - **Eigenvalues** are simply the **coefficients attached to eigenvectors** that give the **amount of variance carried in each principal component**
  - By **ranking eigenvectors** in order of their **eigenvalues**, highest to lowest, you obtain the **principal components in order of significance**

What about Eigenvalues?

- If we rank the eigenvalues in descending order, we get $\lambda_1 > \lambda_2$
- As a result, the eigenvector that corresponds to the first principal component (PC1) is v_1 and the one that corresponds to the second component (PC2) is v_2
- After identifying the principal components, we **divide the eigenvalue of each component by the sum of eigenvalues to determine the percentage of variance** (information) accounted for by each component
- The **feature vector** is simply a matrix that has the eigenvectors of the components as columns
- This is the first step towards dimensionality reduction because, if we choose to keep only ***p*** eigenvectors (components) out of ***n***, the final data set will have only ***p*** dimensions

Eigenvalue Computation



Example 1: Find the eigenvalues of $A = \begin{bmatrix} 2 & -12 \\ 1 & -5 \end{bmatrix}$

$$|\lambda I - A| = \begin{vmatrix} \lambda - 2 & 12 \\ -1 & \lambda + 5 \end{vmatrix} = (\lambda - 2)(\lambda + 5) + 12$$

$$= \lambda^2 + 3\lambda + 2 = (\lambda + 1)(\lambda + 2)$$

two eigenvalues: –1, – 2

Note: The roots of the characteristic equation can be repeated. That is, $\lambda_1 = \lambda_2 = \ldots = \lambda_k$. If that happens, the eigenvalue is said to be of multiplicity k.

Example 2: Find the eigenvalues of $A = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$

$$|\lambda I - A| = \begin{vmatrix} \lambda - 2 & -1 & 0 \\ 0 & \lambda - 2 & 0 \\ 0 & 0 & \lambda - 2 \end{vmatrix} = (\lambda - 2)^3 = 0$$

$\lambda = 2$ is an eigenvector of multiplicity 3.

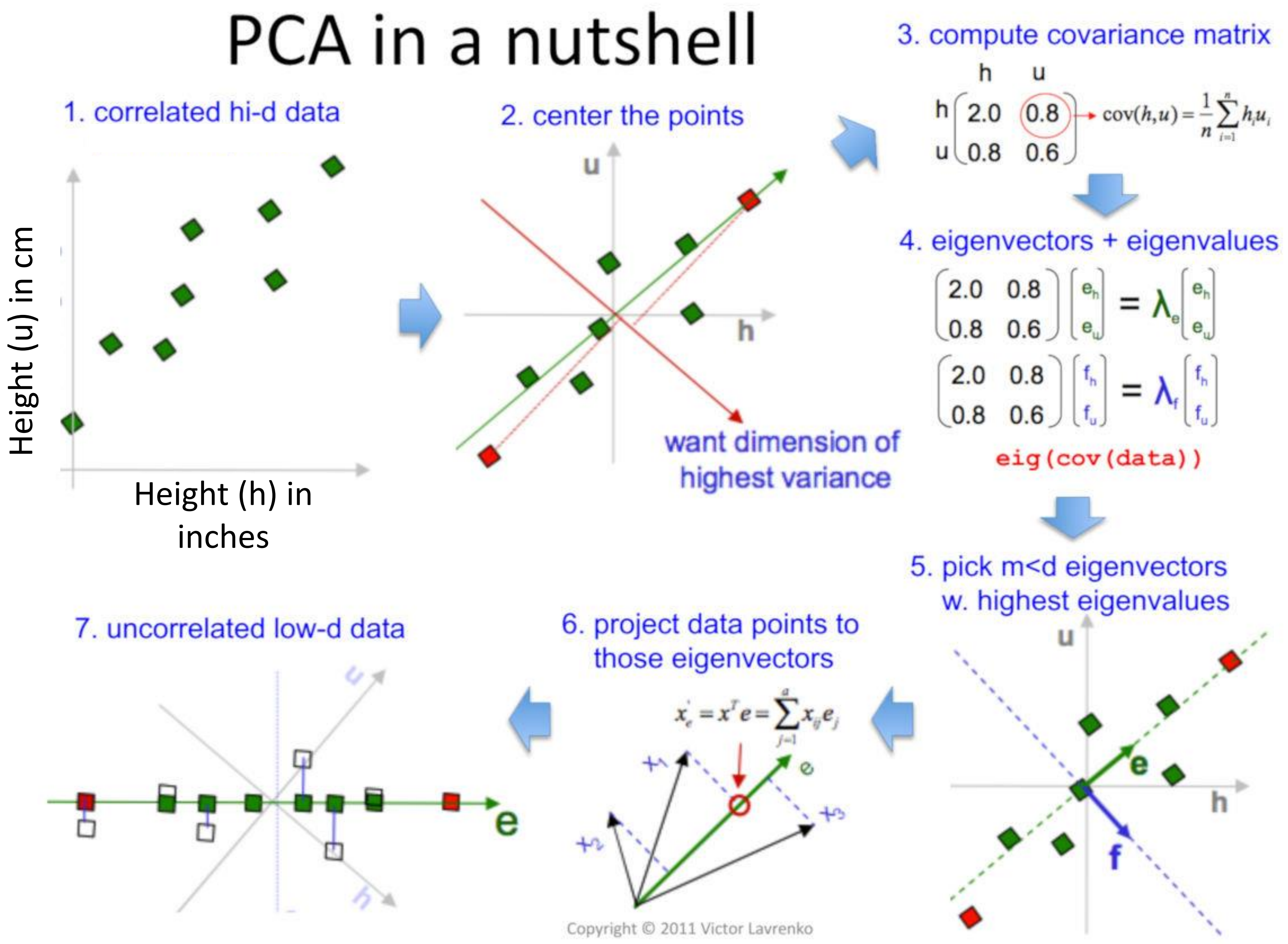
Principal Component Analysis (Cont.)

Recasting Data along the Principal Component Axis

- We use the **feature vector** using the **eigenvectors of the covariance matrix** to reorient the data from the original axes to the ones represented by the principal components (hence the name **Principal Components Analysis**)
- This can be done by multiplying the transpose of the original data set by the transpose of the feature vector

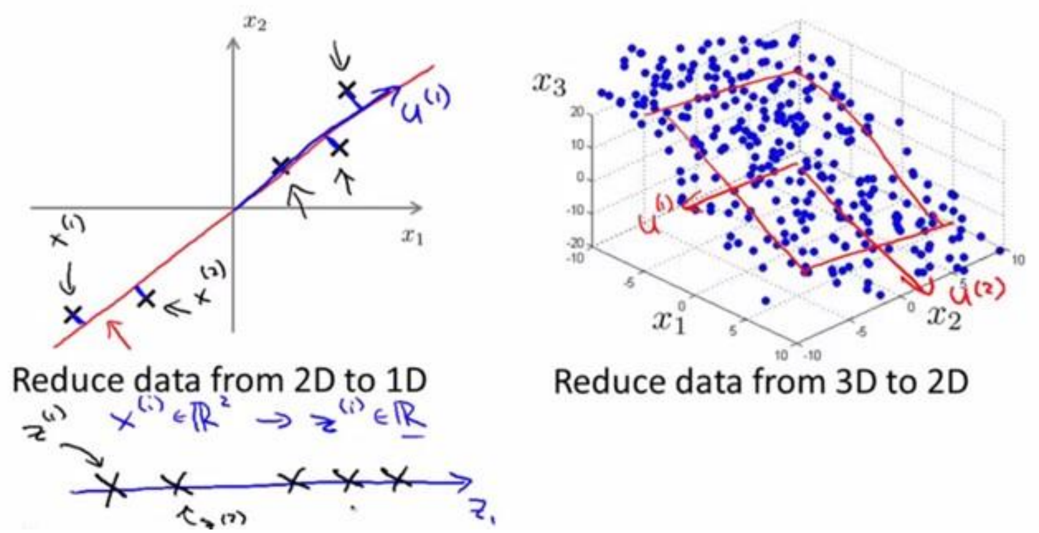
$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

Principal Component Analysis

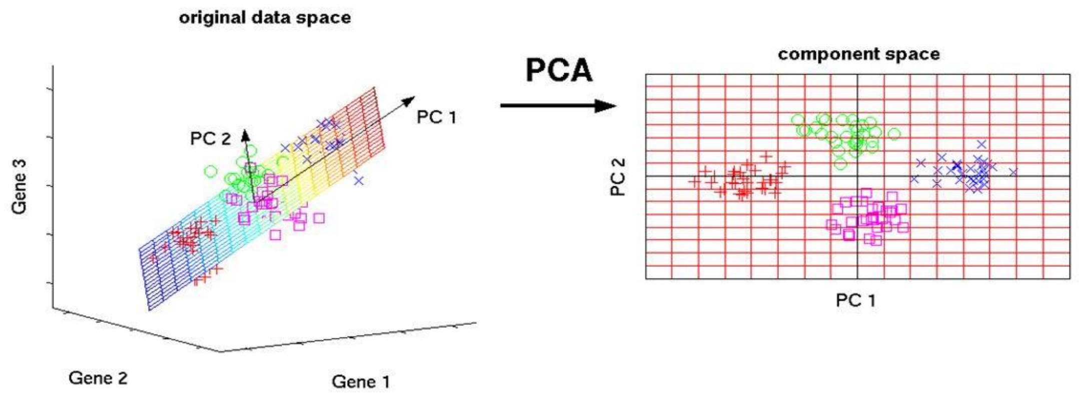

PCA in a nutshell
1. correlated hi-d data
Height (u) in cm
Height (h) in inches
2. center the points
u
h
want dimension of highest variance
3. compute covariance matrix
h u
h 2.0 0.8
u 0.8 0.6
$\mathrm{cov}(h,u) = \frac{1}{n}\sum_{i=1}^{n} h_i u_i$
4. eigenvectors + eigenvalues
$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} e_h \\ e_u \end{pmatrix} = \lambda_e \begin{pmatrix} e_h \\ e_u \end{pmatrix}$
$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} f_h \\ f_u \end{pmatrix} = \lambda_f \begin{pmatrix} f_h \\ f_u \end{pmatrix}$
eig(cov(data))
5. pick m<d eigenvectors w. highest eigenvalues
u
h
e
f
6. project data points to those eigenvectors
$x'_e = x^T e = \sum_{j=1}^{d} x_{ij} e_j$
e
7. uncorrelated low-d data
e
u
h
Copyright © 2011 Victor Lavrenko

Principal Component Analysis

Principal Component Analysis (PCA) algorithm


x_2
x_1
$u^{(1)}$
$x^{(1)}$
$x^{(2)}$
x_3
x_1
x_2
$u^{(1)}$
$u^{(2)}$
Reduce data from 2D to 1D
$z^{(1)}$
$x^{(i)} \in \mathbb{R}^2 \rightarrow z^{(i)} \in \mathbb{R}$
$z^{(2)}$
z_1
Reduce data from 3D to 2D

PCA finds new variables which are linear combinations of the original variables such that in the new space, the data has fewer dimensions.



Think of a data set consisting of points in 3D on the surface of a flat plate held up at an angle. In the original x, y, z axes you need 3 dimensions to represent the data, but with the correct linear transformation, you only need 2.

Quotes: Shlomo Argamon, from http://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues
Image: http://phdthesis-bioinformatics-maxplanckinstitute-molecularplantphys.matthias-scholz.de/

T-Distributed Stochastic Neighbor Embedding

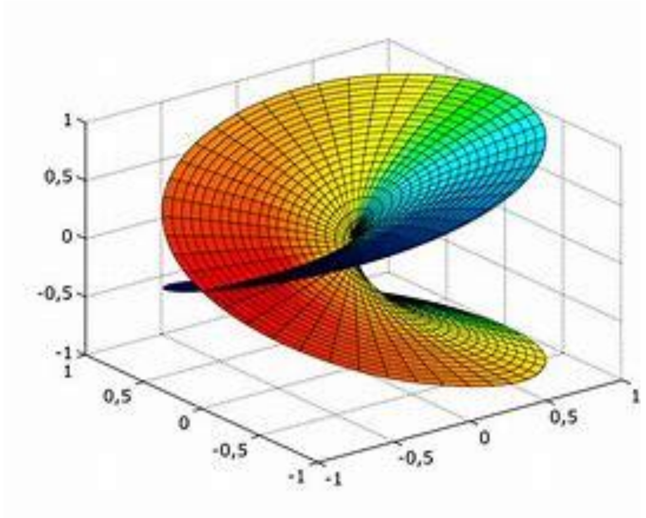
- **T-Distributed Stochastic Neighbor Embedding (t-SNE)** is another technique for dimensionality reduction
- It is particularly well suited for the visualization of high-dimensional datasets
- Contrary to PCA, it is not a mathematical technique but a probabilistic one
- *"t-Distributed stochastic neighbor embedding (t-SNE)* ***minimizes the divergence between two distributions****: a distribution that measures* ***pairwise similarities of the input objects*** *and a distribution that measures* ***pairwise similarities of the corresponding low-dimensional points*** *in the embedding"*

  van der Maaten and Hinton (2008)
- It is extensively applied in image processing, NLP, genomic data and speech processing

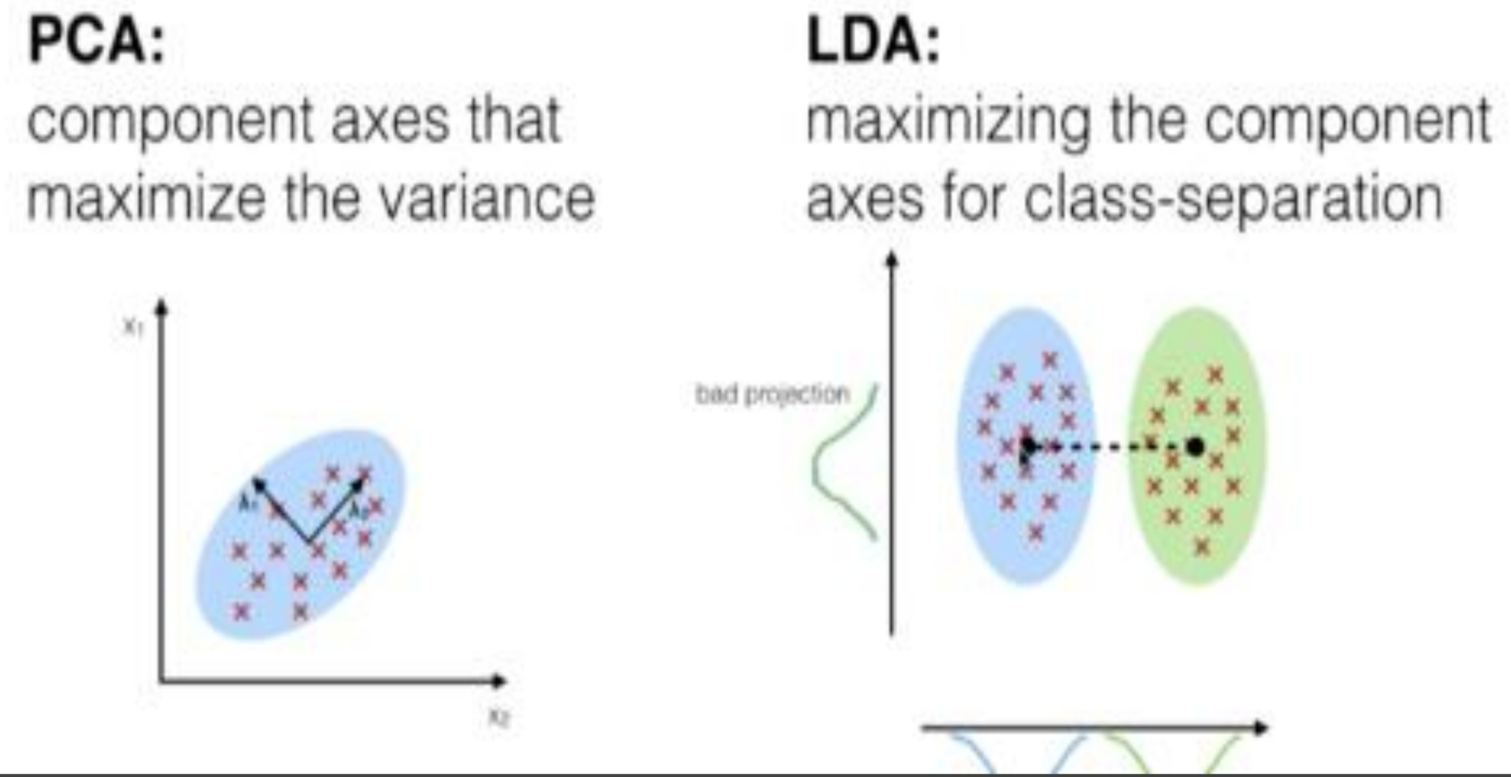
- **T-Distributed stochastic neighbor embedding** (**t-SNE**) minimizes the divergence between two distributions:
  - A distribution that measures pairwise similarities of the input objects and
  - A distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding
- **T-SNE** maps the multi-dimensional data to a lower dimensional space and attempts to find patterns in the data by identifying observed clusters based on similarity of data points with multiple features
- After this process, the input features are no longer identifiable
- It is not possible to make any inference based only on the output of **t-SNE**
- It is mainly a data exploration and visualization technique

Uniform Manifold Approximation and Projection

- **Uniform Manifold Approximation and Projection (UMAP)** is a dimension reduction technique that can be used for visualization similarly to **t-SNE**, but also for general non-linear dimension reduction
- The algorithm is founded on three assumptions about the data
  - The data is uniformly distributed on Riemannian manifold
  - The Riemannian metric is locally constant (or can be approximated as such)
  - The manifold is locally connected
- Riemannian geometry includes **the branch of non-Euclidean geometry** that replaces the postulate of Euclidean geometry with the postulate that, in a plane, **every pair of distinct lines intersects**
- **UMAP** uses **exponential probability distribution in high dimensions** but **not necessarily Euclidean distances** like **t-SNE**
- Any distance can be plugged in
- In addition, the probabilities are **not normalized**



Riemannian Manifold
Sources: McInnes and Healy, 2018


PCA:
component axes that maximize the variance
LDA:
maximizing the component axes for class-separation
x1
x2
bad projection

Linear Discriminant Analysis

Introduction

- **Linear Discriminant Analysis (LDA)** is a dimensionality reduction technique
- **Linear Discriminant Analysis** is a **supervised algorithm** that takes into the account the labelled data while **carrying out dimensionality reduction method**
- **Principal Component Analysis** is a statistical procedure that converts a set of possibly correlated variables into a set of **linearly uncorrelated features called principal components**
  - It essentially drops the least important variables while retaining the valuable ones by finding principal component axes along which the variance of data is high
- With **LDA**, we intend to maximize the separation between classes by
  - **Maximizing the distance between the centroids of the classes** and, at the same time,
  - **Minimizing the within-class variance to create non-overlapping clusters**
- **Logistic regression** is a classification algorithm traditionally limited to **only two-class classification problems**
- If you have more than two classes, then **Linear Discriminant Analysis** is the preferred linear classification technique

LDA and the Limitations of Logistic Regression

- **Logistic regression** is a simple and powerful linear classification algorithm. However, it also has limitations that are addressed by **LDA**
  - **Two-Class Problems**. **Logistic regression** is intended for two-class or binary classification problems. It can be extended for multi-class classification. However, it is not often used for that purpose
  - **Unstable With Well Separated Classes**. **Logistic regression** can become unstable when the classes are well separated
  - **Unstable With Few Examples**. **Logistic regression** can become unstable when there are few examples from which to estimate the parameters

- **Linear Discriminant Analysis** does address each of these points and is the choice method for **multi-class classification problems**
- Even with binary-classification problems, it is a good idea to try both logistic regression and linear discriminant analysis and compare the outcomes

Linear Discriminant Analysis (Cont.)

Steps for LDA Modeling:

- Compute d-dimensional mean vectors for different classes from the dataset, where d is the dimension of feature space
- Compute in-between class and with-in class scatter matrices
- Compute eigenvectors and corresponding eigenvalues for the scatter matrices
- Choose k eigenvectors corresponding to top k eigen values to form a transformation matrix of dimension d x k
- Transform the d-dimensional feature space X to k-dimensional feature space X_lda via the transformation matrix

- **LDA** can match and even outperform **logistic regression** in a variety of settings
- **LDA** is often used in machine learning for **dimensionality reduction.** However, it can also be effectively used for **classification**
- The technique will work better if the input variables come from an **approximately normal distribution**
- The less this approximation holds, the poorer the performance will be

Linear Discriminant Analysis (Cont.)

- **LDA** makes some simplifying assumptions about your data:
  - The **data is Gaussian**, that each variable is shaped like a bell curve when plotted
  - **Each attribute has the same variance**
  - **Values of each variable vary around the mean by the same amount on average**
- With these assumptions, the **LDA** model estimates the mean and variance from the data for each class
- It is easy to think about this in the **univariate (single input variable) case with two classes**
- The **mean** ($\boldsymbol{\mu}$) value of each input (x) for each class (k) can be estimated in the normal way by dividing the sum of values by the total number of values, that is, **$\boldsymbol{\mu}_k$ = 1/nk * sum(x)**

where $\boldsymbol{\mu}_k$ is the mean value of x for the class k, nk is the number of instances with class k

- The **variance** is calculated across all classes as the average squared difference of each value from the mean such that **$\boldsymbol{\sigma}^2$ = 1 / (n - K) * sum((x – $\boldsymbol{\mu}$) ^{2})**

where $\boldsymbol{\sigma}^2$ is the variance across all inputs (x), n is the number of instances, K is the number of classes and $\boldsymbol{\mu}$ is the mean for input x

Making Predictions with LDA Model

- **LDA** makes predictions by estimating the probability that a new set of inputs belongs to each class. The class that gets the highest probability is the output class and a prediction is made
- The model uses **Bayes Theorem** to estimate the probabilities. **Bayes Theorem** can be used to estimate the probability of the output class (k) given the input (x) using the probability of each class and the probability of the data belonging to each class:

$$\mathbf{P(Y = x|X = x) = (PI_k * f_k(x)) / sum(PI_l * f_l(x))}$$

where PI_k refers to the base probability of each class (k) observed in your training data (e.g. 0.5 for a 50-50 split in a two-class problem). In **Bayes Theorem** this is called the prior probability

$$\mathbf{PI_k = nk/n}$$

- The f(x) above is the estimated probability of x belonging to the class
- A **Gaussian distribution function** is used for f(x). Plugging the Gaussian into the above equation and simplifying we end up with the equation below. This is called a **discriminate function** and the class is calculated as having the **largest value will be the output classification** (y):

$$\mathbf{D_k(x) = x * (\mu_k/\sigma^2) - (\mu_k^2/(2* \sigma^2)) + \ln(PI_k)}$$

where $D_k(x)$ is the discriminate function for class k given input x, the μ_k, σ^2 and PI_k are all estimated from your data

Data Preparation for LDA Models

- **Classification Problems**. The output variable is categorical. **LDA** supports both binary and multi-class classification
- **Gaussian Distribution**. The standard implementation of the model assumes a Gaussian distribution of the input variables. Consider reviewing the univariate distributions of each attribute and using transforms to make them more Gaussian-looking (e.g. log and root for exponential distributions and Box-Cox for skewed distributions)
- **Remove Outliers**. Consider removing outliers from your data. These can skew the basic statistics used to separate classes in LDA such the mean and the standard deviation
- **Same Variance. LDA** assumes that each input variable has the same variance
    - It is a good idea to standardize the data before using **LDA** so that it has a mean of 0 and a standard deviation of 1

Linear Discriminant Analysis (Cont.)

LDA Extensions

- There are many extensions and variations to the method. Some popular extensions include:

  - **Quadratic Discriminant Analysis (QDA)**: Each class uses its own estimate of variance (or covariance when there are multiple input variables)
  - **Flexible Discriminant Analysis (FDA)**: Where non-linear combinations of inputs is used such as splines
  - **Regularized Discriminant Analysis (RDA)**: Introduces regularization into the estimate of the variance (actually, it is covariance), moderating the influence of different variables on LDA

- The original development was called the **Linear Discriminant** or **Fisher's Discriminant Analysis**
- The multi-class version was referred to **Multiple Discriminant Analysis**. These are all simply referred to as **Linear Discriminant Analysis** nowadays