McMaster University ENGINEERING

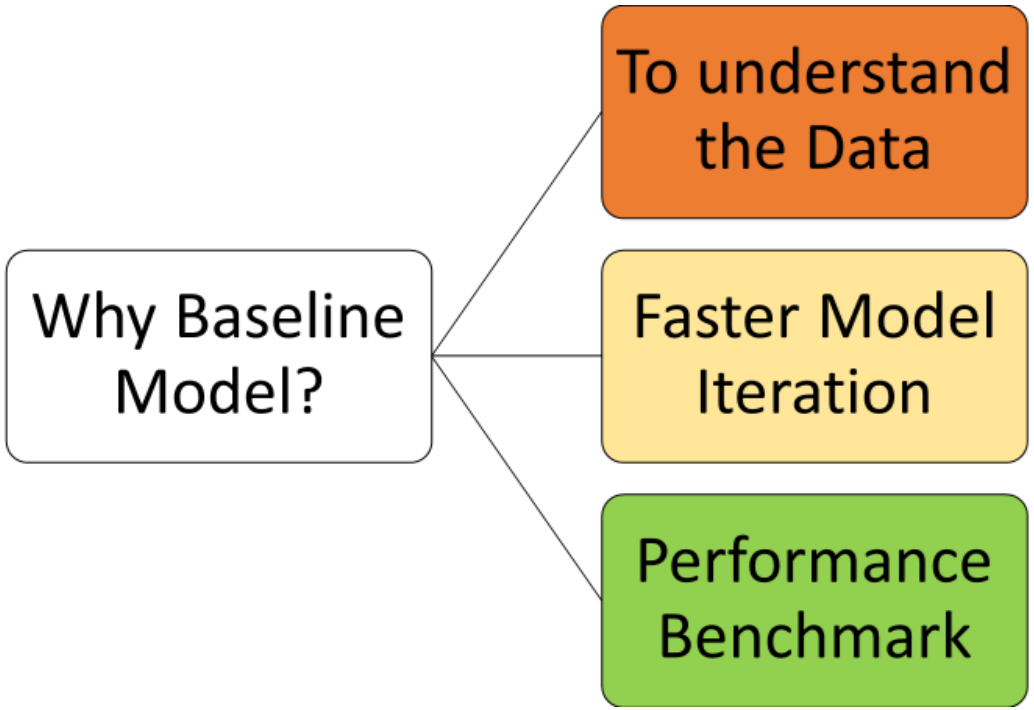
W BOOTH SCHOOL OF ENGINEERING PRACTICE AND TECHNOLOGY

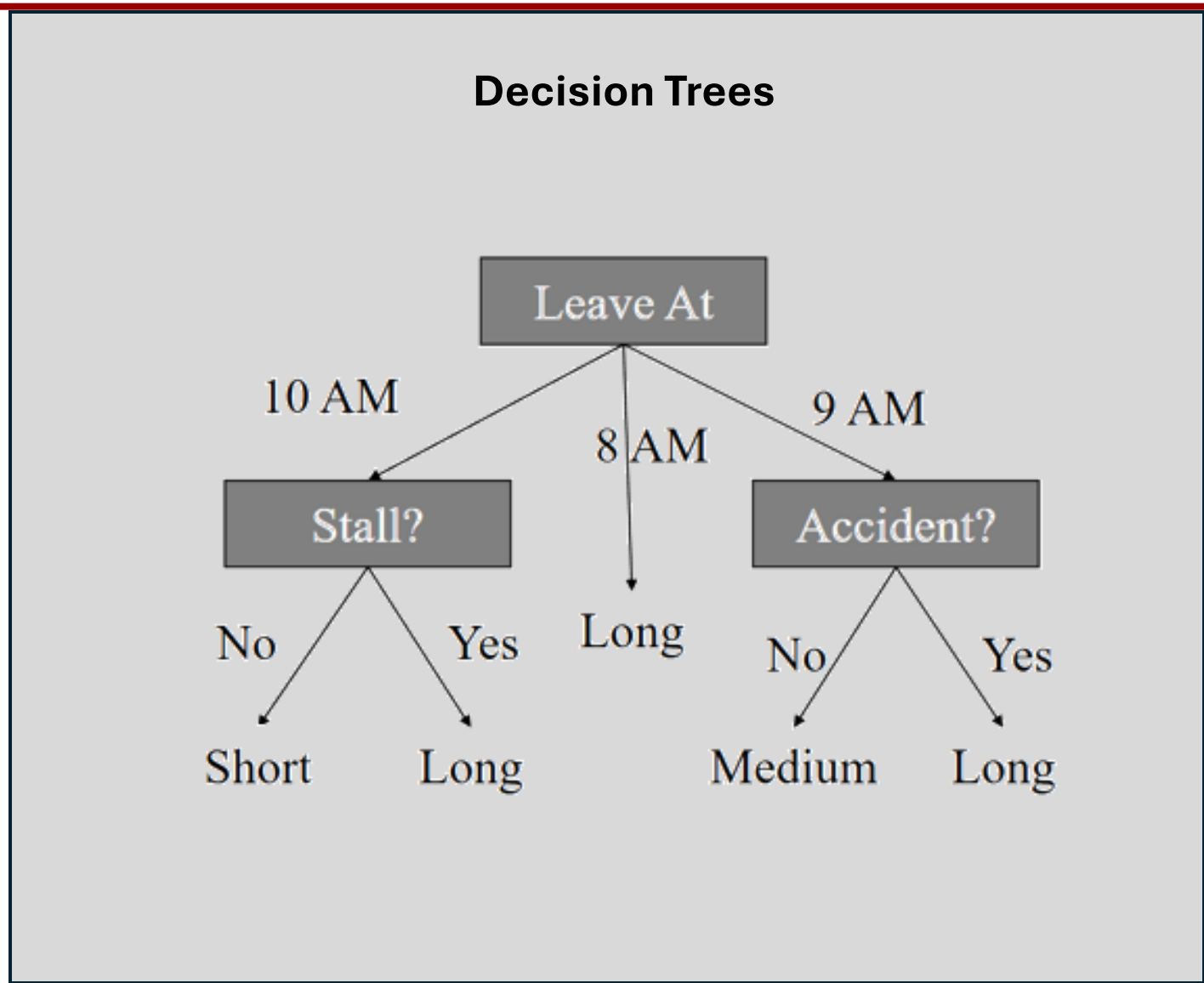
SEP 785: Machine Learning

Lecture 4: Dimensional Reduction and k-Nearest Neighbor

Instructor: Dr. Dalia Mahmoud, PhD
(Mechanical Engineering, McMaster University)
Email: mahmoudd@mcmaster.ca

Recap


Why Baseline Model?
To understand the Data
Faster Model Iteration
Performance Benchmark


Decision Trees
Leave At
10 AM
8 AM
9 AM
Stall?
Long
Accident?
No
Yes
No
Yes
Short
Long
Medium
Long

Intended Learning Outcomes

- **Explain the Importance of Dimensionality Reduction** in Machine Learning and its impact on model performance.
- **Perform Principal Component Analysis (PCA)** and interpret its results.
- **Conduct Linear Discriminant Analysis (LDA)** for feature extraction and classification.
- **Compare and Contrast PCA and LDA**, highlighting their applications and differences.
- **Implement and Apply the k-Nearest Neighbors (K-NN) Algorithm** to real-world datasets.

Contents

Dimensional Reduction

- As the number of features or dimensions in a **dataset increases**, the amount of data required to obtain a statistically significant result increases exponentially.
- This can lead to issues such as
  - overfitting,
  - increased computation time,
  - and reduced accuracy of machine learning models
- This is known as the **curse of dimensionality** problems that arise while working with high-dimensional data.
- **Dimensionality reduction** is a type of feature extraction technique that aims to reduce the number of **input features** while retaining as much of the **original information** as **possible**.

Dimensional Reduction Techniques

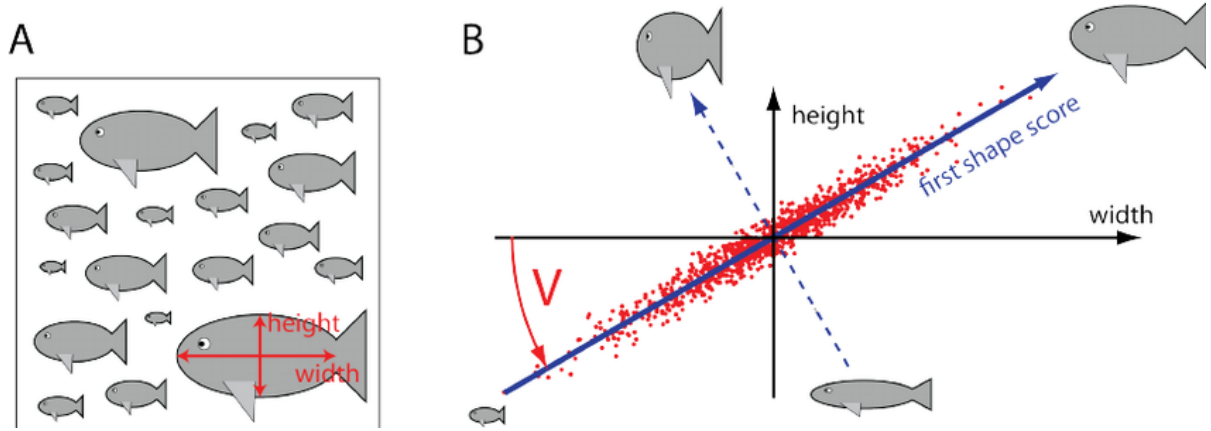
Some Popular Dimensional Reduction Techniques:

- Principal component analysis (PCA)
- Linear discriminant analysis(LDA)
- T-distributed stochastic neighbor embedding (tSNE)
- Independent component analysis (ICA)
- Uniform manifold approximation and projection (UMAP)

Principle Component Analysis

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

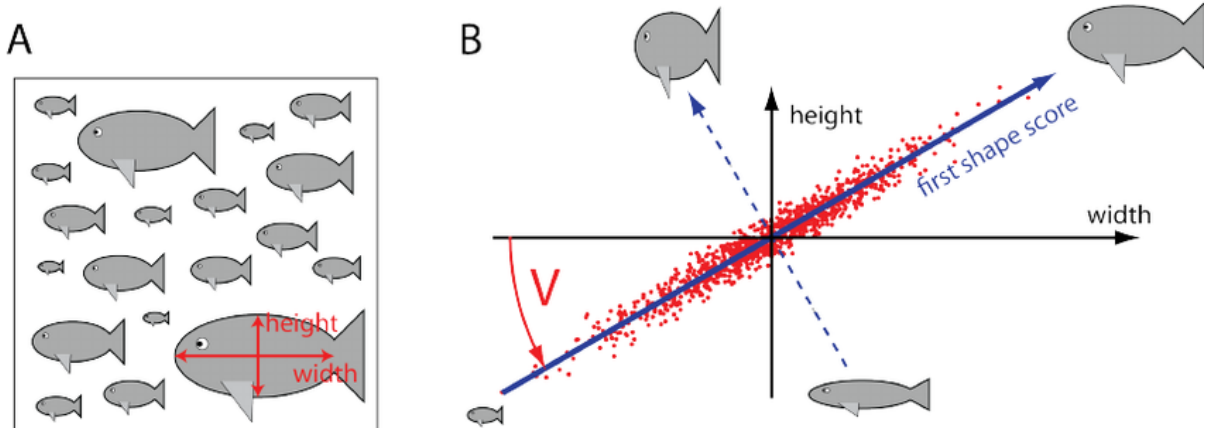
- We can describe the shape of a fish with two variables: **height** and **width**.
- However, these two variables have a **strong correlation**.
- Given the **height**, we can probably **estimate** the **width**; and vice versa.
- Thus, we may say that the shape of a fish can be described with a **single component**.



Principle Component Analysis

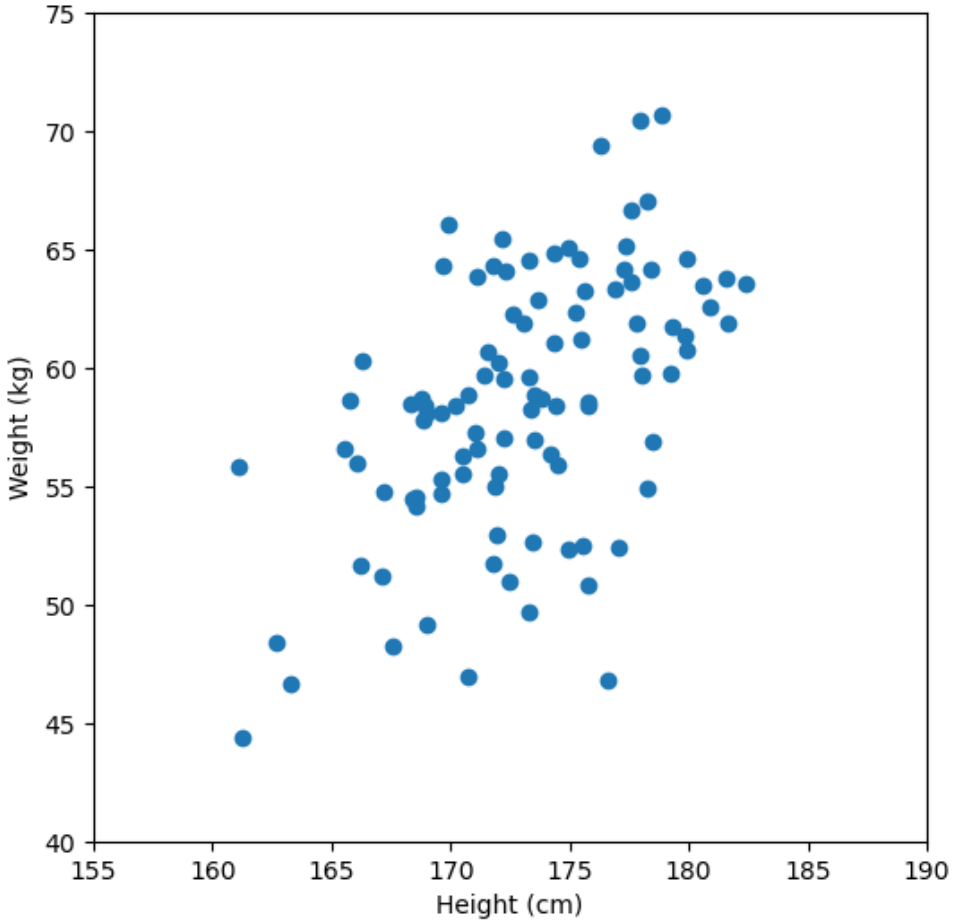
McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- This doesn't mean that we simply **ignore** either height or width.
- Instead, we transform our two original variables into two orthogonal (independent) components that give a complete alternative description.
- The first component (blue line) will **explain most** of the **variation** in the data.
- The second component (dotted line) will explain the **remaining variation**.
- Note that both components are derived from both **height** and **width**.



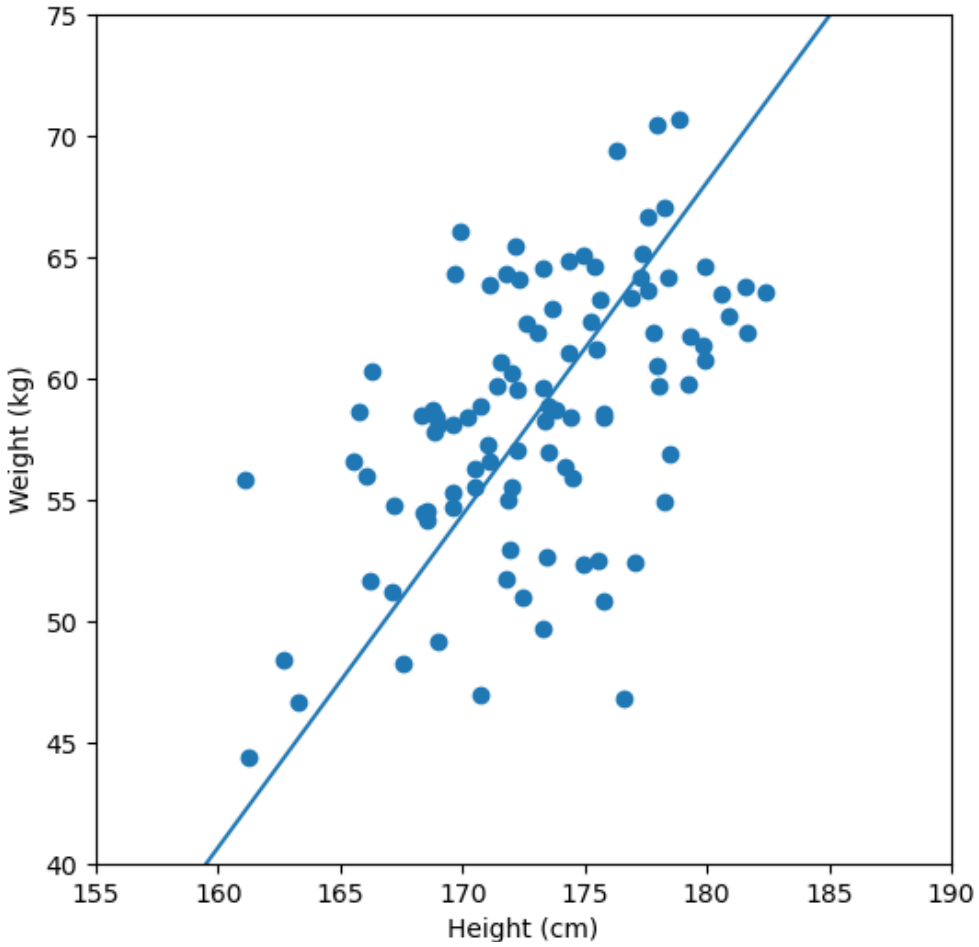
PCA Example 2D

- Let's say we have some data of 50 people's height and weight.



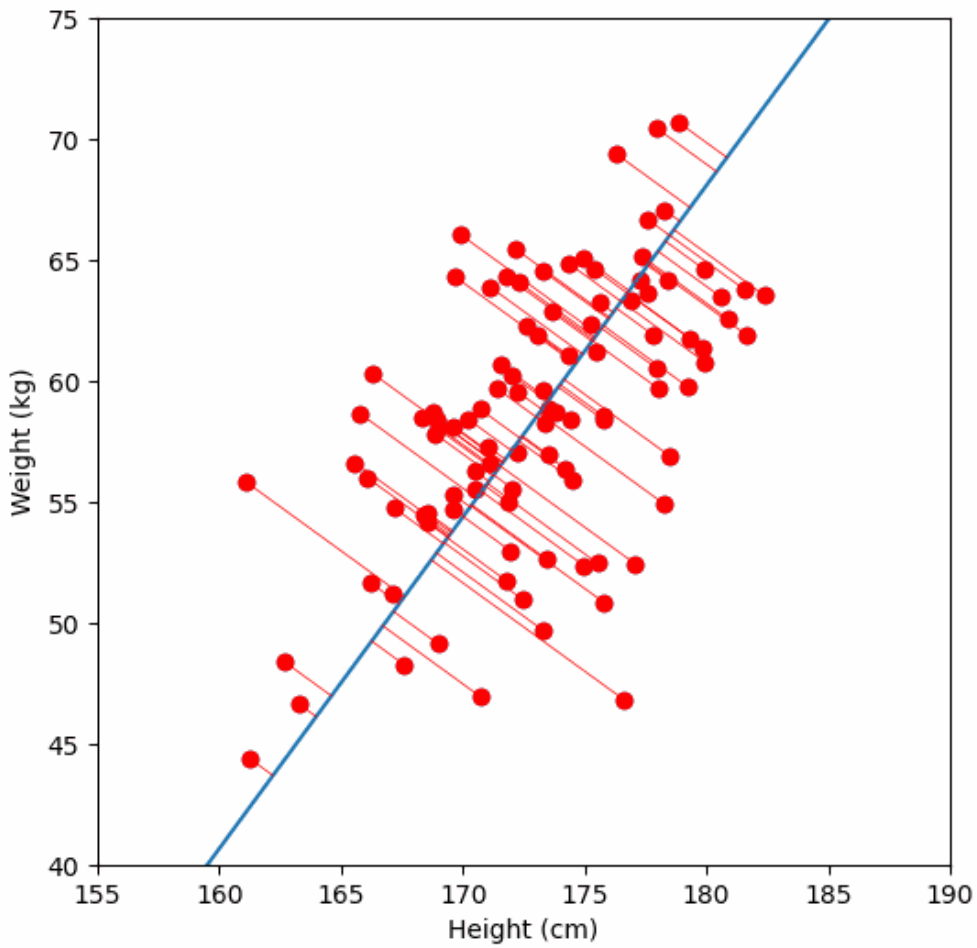
PCA Example 2D

- In order to perform a PCA, we first find the axis of greatest variation.
- Which is basically the line of best fit.
- This line is called the first principal component.



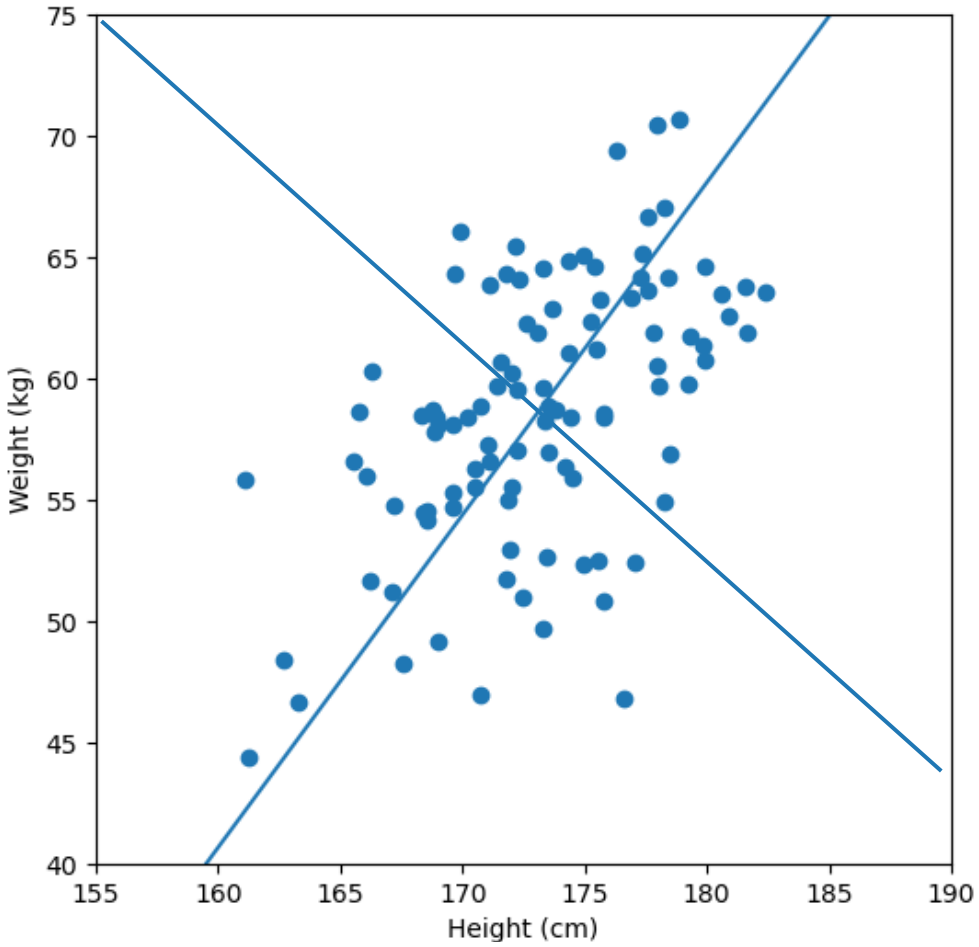
PCA Example 2D

- Finally, we "project" our data points onto first principal component.
- Which means, imagine the line of best fit is a ruler.
- Each data points new value is how far along the ruler it is.
- Because our original data has 2 dimensions, there is also a second principal component, which is perpendicular (or "orthogonal") to the first principal component.



PCA Example 2D

- In higher dimensions, things get difficult the visualize, but in practice the idea is the same: we get a bunch of principal components, with the first one being the line through our data in the direction of most of the variance,
- and each subsequent line being orthogonal to the previous, but in the direction of the next most variance.



Steps To Apply PCA in large datasets

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

1-Standardization

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.

f1	f2	f3	f4
1	2	3	4
5	5	6	7
1	4	2	3
5	3	2	1
8	1	2	2

$$x_{new} = \frac{x - \mu}{\sigma}$$

	f1	f2	f3	f4
μ =	??	??	??	??
σ =				

https://medium.com/analytics-vidhya/understanding-principle-component-analysis-pca-step-by-step-e7a4bb4031d9

Steps To Apply PCA in large datasets

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

2- Covariance Matrix Computations

Because sometimes, variables are highly correlated in such a way that they contain redundant information. So, in order to identify these correlations, we compute the covariance matrix.

f1	f2	f3	f4
-1	-0.63246	0	0.26062
0.33333	1.26491	1.73205	1.56374
-1	0.63246	-0.57735	-0.17375
0.33333	0	-0.57735	-1.04249
1.33333	-1.26491	-0.57735	-0.60812

	f1	f2	f3	f4
f1	var(f1)	cov(f1,f2)	cov(f1,f3)	cov(f1,f4)
f2	cov(f2,f1)	var(f2)	cov(f2,f3)	cov(f2,f4)
f3	cov(f3,f1)	cov(f3,f2)	var(f3)	cov(f3,f4)
f4	cov(f4,f1)	cov(f4,f2)	cov(f4,f3)	var(f4)

	f1	f2	f3	f4
f1	??	??	??	??
f2	??	??	??	??
f3	??	??	??	??
f4	??	??	??	??

For Population

$$Cov(x,y) = \frac{\sum (x_i - \bar{x}) * (y_i - \bar{y})}{N}$$

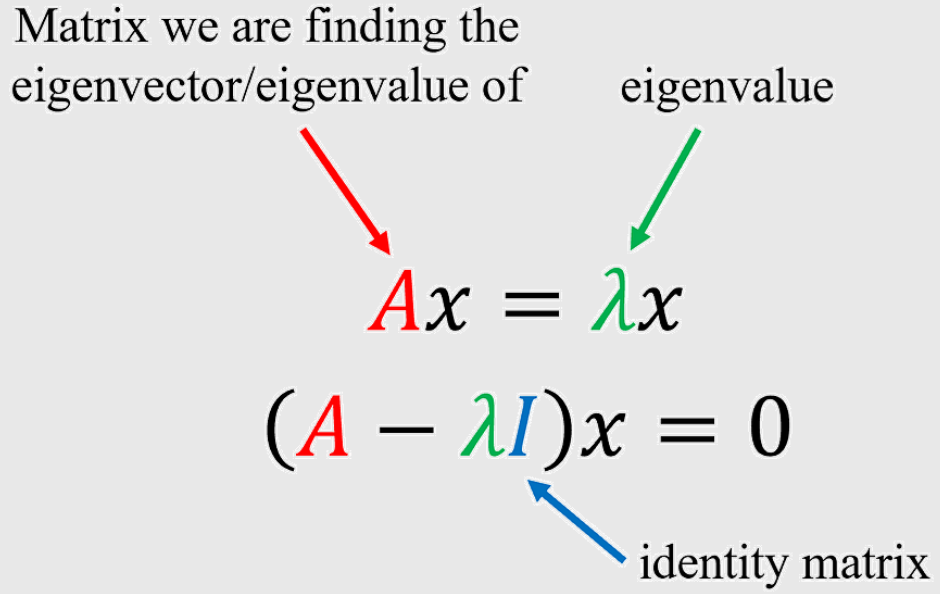
For Sample

$$Cov(x,y) = \frac{\sum (x_i - \bar{x}) * (y_i - \bar{y})}{(N-1)}$$

Eigenvalues and Eigenvectors

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- One of the most important problems in Linear algebra
- In linear algebra, an **eigenvector** is a special vector that, when multiplied by a matrix, doesn't change direction—only its length might change. The amount by which it is stretched or shrunk is called the **eigenvalue**.
- A is an nxn matrix
- X is a non zero vector
- λ is a scalar could be 0



https://youtu.be/PFDu9oVAE-g?si=dy0Y_1E6j1FAthTu

Steps To Apply PCA in large datasets

3-Compute the eigenvectors and eigenvalues

Let A be a square matrix (in our case the covariance matrix), v a vector and λ a scalar that satisfies Av = λv, then λ is called eigenvalue associated with eigenvector v of A.

Av - λv = 0 , (A-λI)v=0

Since we have already know v is a non- zero vector, only way this equation can be equal to zero, if

Det (A-λI)=0

	f1	f2	f3	f4
f1	0.8 - λ	-0.25298	0.03849	-0.14479
f2	-0.25298	0.8- λ	0.51121	0.4945
f3	0.03849	0.51121	0.8 - λ	0.75236
f4	-0.14479	0.4945	0.75236	0.8 - λ

A-λI = 0

$$\det(A-\lambda I) = 0$$

$$\lambda = 2.51579324 , 1.0652885 , 0.39388704 , 0.02503121$$

Steps To Apply PCA in large datasets

- Solving the (A-λI)v = 0 equation for v vector with different λ values:

$$\begin{pmatrix} 0.800000-\lambda & -(0.252982) & 0.038490 & -(0.144791) \\ -(0.252982) & 0.800000-\lambda & 0.511208 & 0.494498 \\ 0.038490 & 0.511208 & 0.800000-\lambda & 0.752355 \\ -(0.144791) & 0.494498 & 0.752355 & 0.800000-\lambda \end{pmatrix} \times \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = 0$$

- For λ = *2.51579324, solving the above equation using Cramer's rule, the values for v vector are*

v1 = 0.16195986

v2 = -0.52404813

v3 = -0.58589647

v4 = -0.59654663

- Going by the same approach, we can calculate the eigen vectors for the other eigen values. We can from a matrix using the eigen vectors.

e1	e2	e3	e4
0.161960	-0.917059	-0.307071	0.196162
-0.524048	0.206922	-0.817319	0.120610
-0.585896	-0.320539	0.188250	-0.720099
-0.596547	-0.115935	0.449733	0.654547

Steps To Apply PCA in large datasets

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

4. Sort eigenvalues and their corresponding eigenvectors: Since eigenvalues are already sorted in this case so no need to sort them again.

5. Pick k eigenvalues and form a matrix of eigenvectors:

If we choose the top 2 eigenvectors, the matrix will look like this:

6. Transform the original matrix:

Feature matrix * top k eigenvectors = Transformed Data

e1	e2	e3	e4
0.161960	-0.917059	-0.307071	0.196162
-0.524048	0.206922	-0.817319	0.120610
-0.585896	-0.320539	0.188250	-0.720099
-0.596547	-0.115935	0.449733	0.654547

e1	e2
0.161960	-0.917059
-0.524048	0.206922
-0.585896	-0.320539
-0.596547	-0.115935

Top 2 eigenvectors(4*2 matrix)

f1	f2	f3	f4
-1.000000	-0.632456	0.000000	0.260623
0.333333	1.264911	1.732051	1.563740
-1.000000	0.632456	-0.577350	-0.173749
0.333333	0.000000	-0.577350	-1.042493
1.333333	-1.264911	-0.577350	-0.608121

(5,4)

*

e1	e2
0.161960	-0.917059
-0.524048	0.206922
-0.585896	-0.320539
-0.596547	-0.115935

(4,2)

=

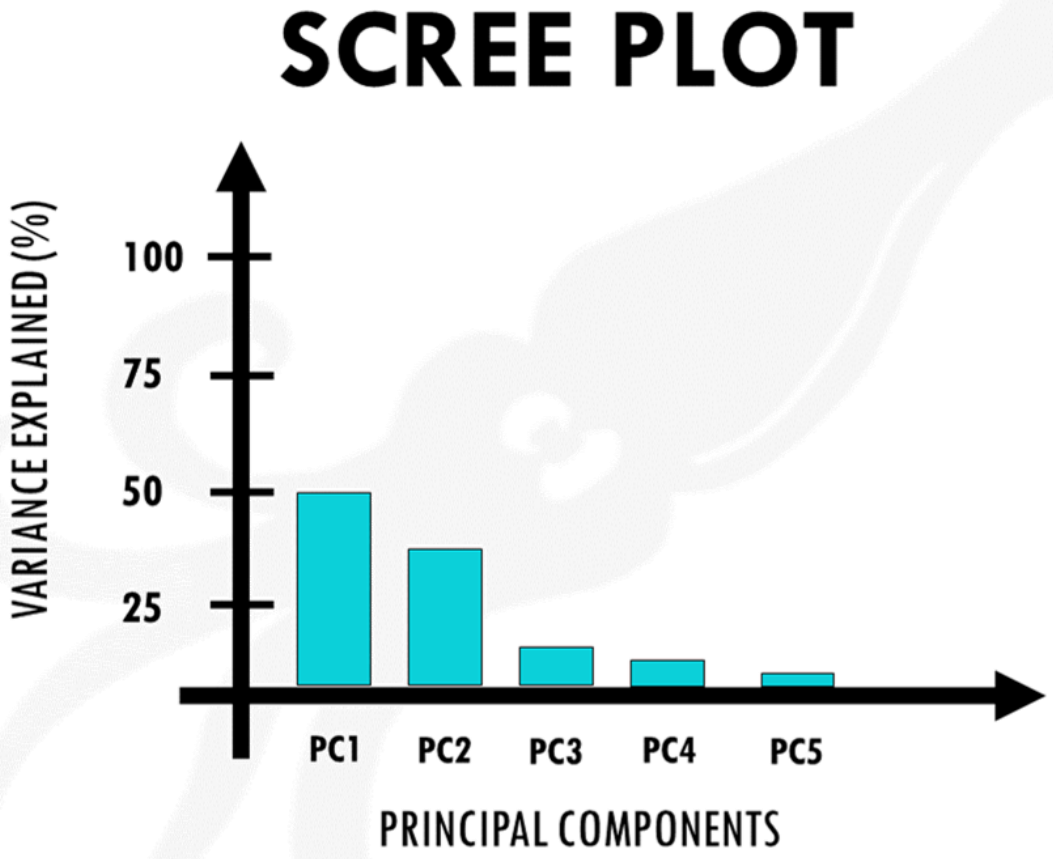
nf1	nf2
0.014003	0.755975
-2.556534	-0.780432
-0.051480	1.253135
1.014150	0.000239
1.579861	-1.228917

(5,2)

Scree Plot

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- A scree plot tells you how much variance of the dataset (basically how much information) is explained with each principal component.

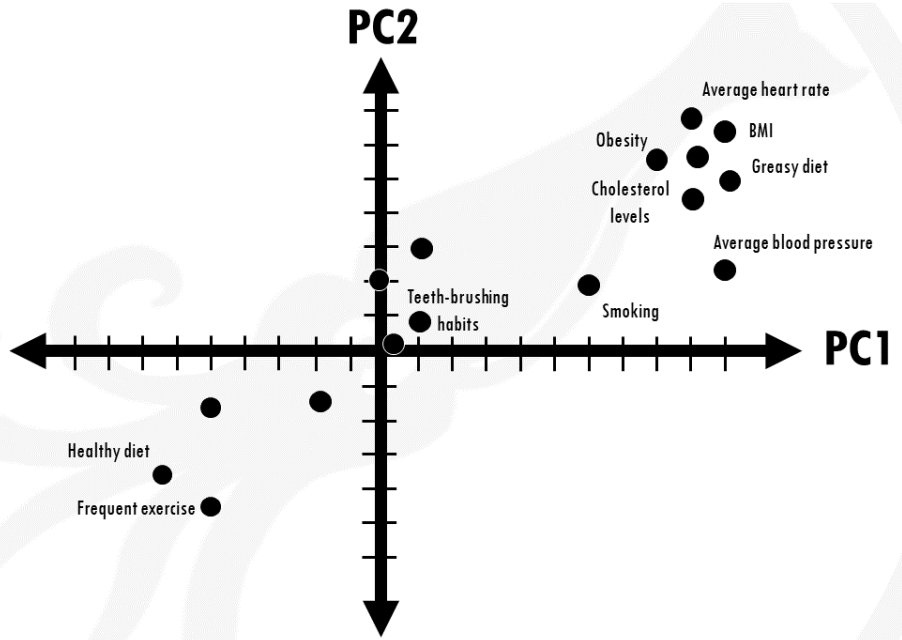


https://biostatsquid.com/pca-simply-explained/

Principal Components Loadings

- Each variable gets a loading, or weight, for each principal component, which tells you how much it contributes to that PC.

	PC1	PC2	PC3	PC4	PC5
Height	-1	3	-1	4	4
Average heart rate	9	7	5	-4	-4
BMI	10	6.5	2	2	5
Cholesterol levels	9	5	-4	3	2
Average cigarettes/day	7	2	5	5	-4
Greasy diet	10	5	-6	2	5
Frequent exercise	-5	-6	8	1	9
Eye colour	0.1	0.3	0.1	0.3	0.3
Teeth-brushing habits	0.2	0.2	0.2	0.2	0.2



Drawbacks of PCA

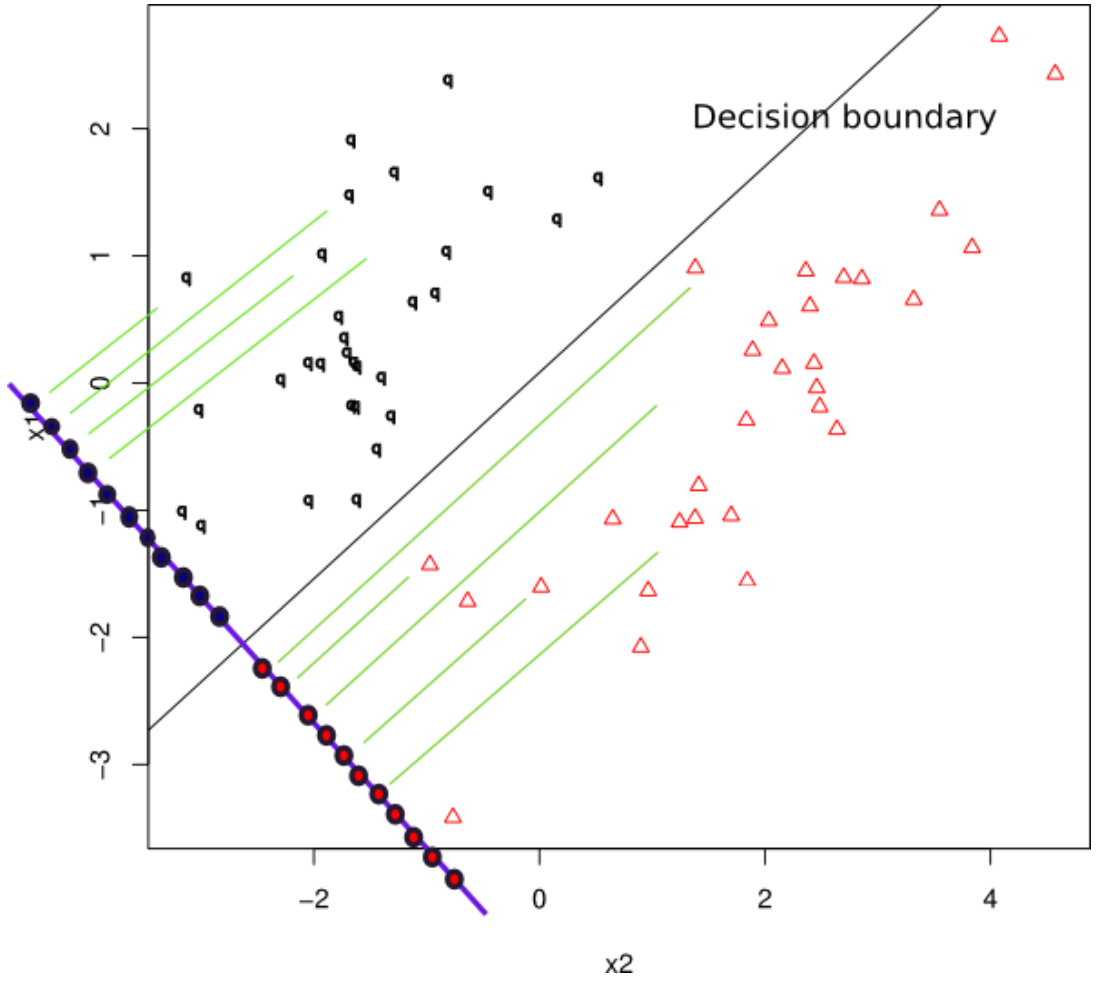
- PCA works only if the observed variables are **linearly correlated**. If there's no correlation, PCA will fail to capture adequate variance with fewer components.
- PCA is lossy. Information is **lost** when we discard insignificant components.
- **Scaling** of **variables** can yield different results. Hence, scaling that you use should be documented. Scaling should not be adjusted to match prior knowledge of data.
- Since each principal components is a linear combination of the original features, **visualizations** are **not easy** to **interpret** or relate to **original features.**

Contents

Linear Discriminant Analysis

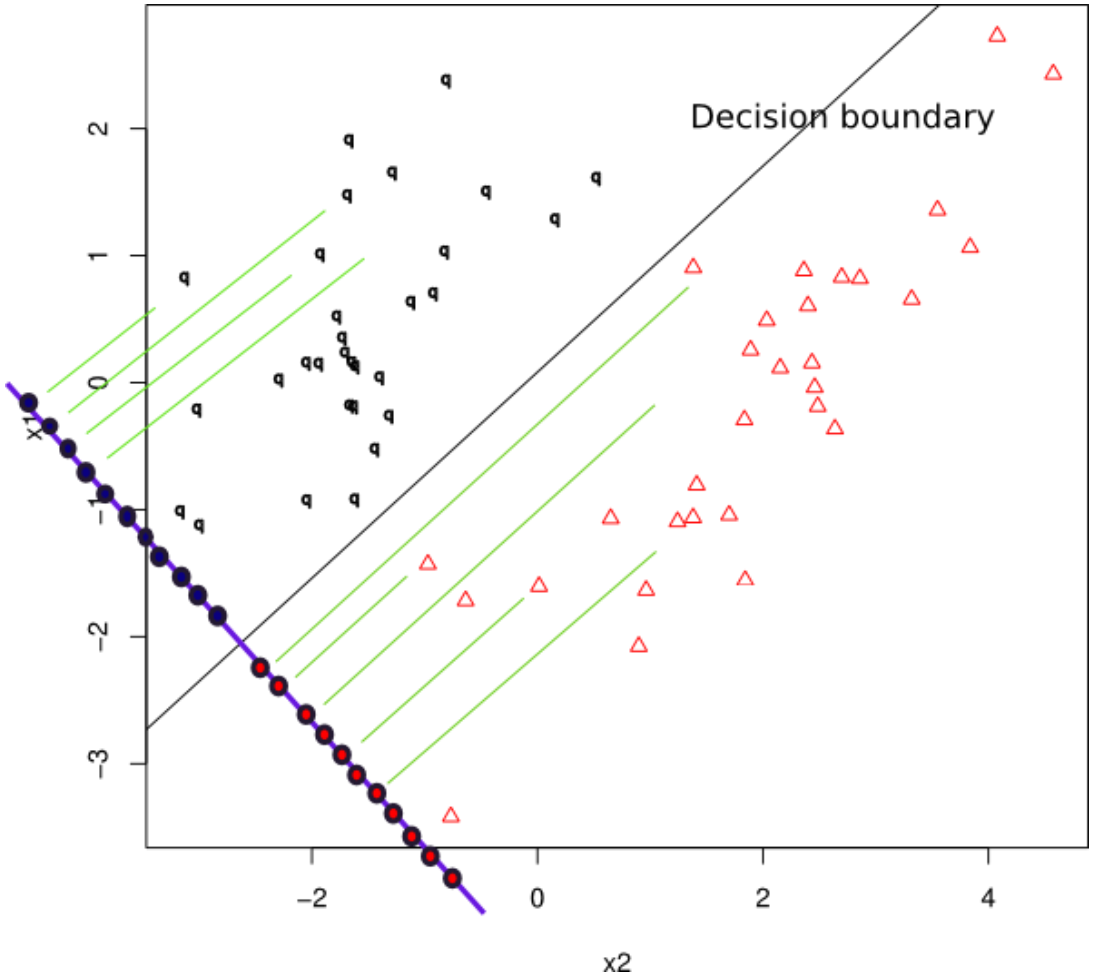
McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- Is a dimensionality reduction technique primarily utilized in supervised classification problems.
- It facilitates the modeling of distinctions between groups, effectively separating two or more classes.
- LDA operates by projecting features from a higher-dimensional space into a lower-dimensional one.



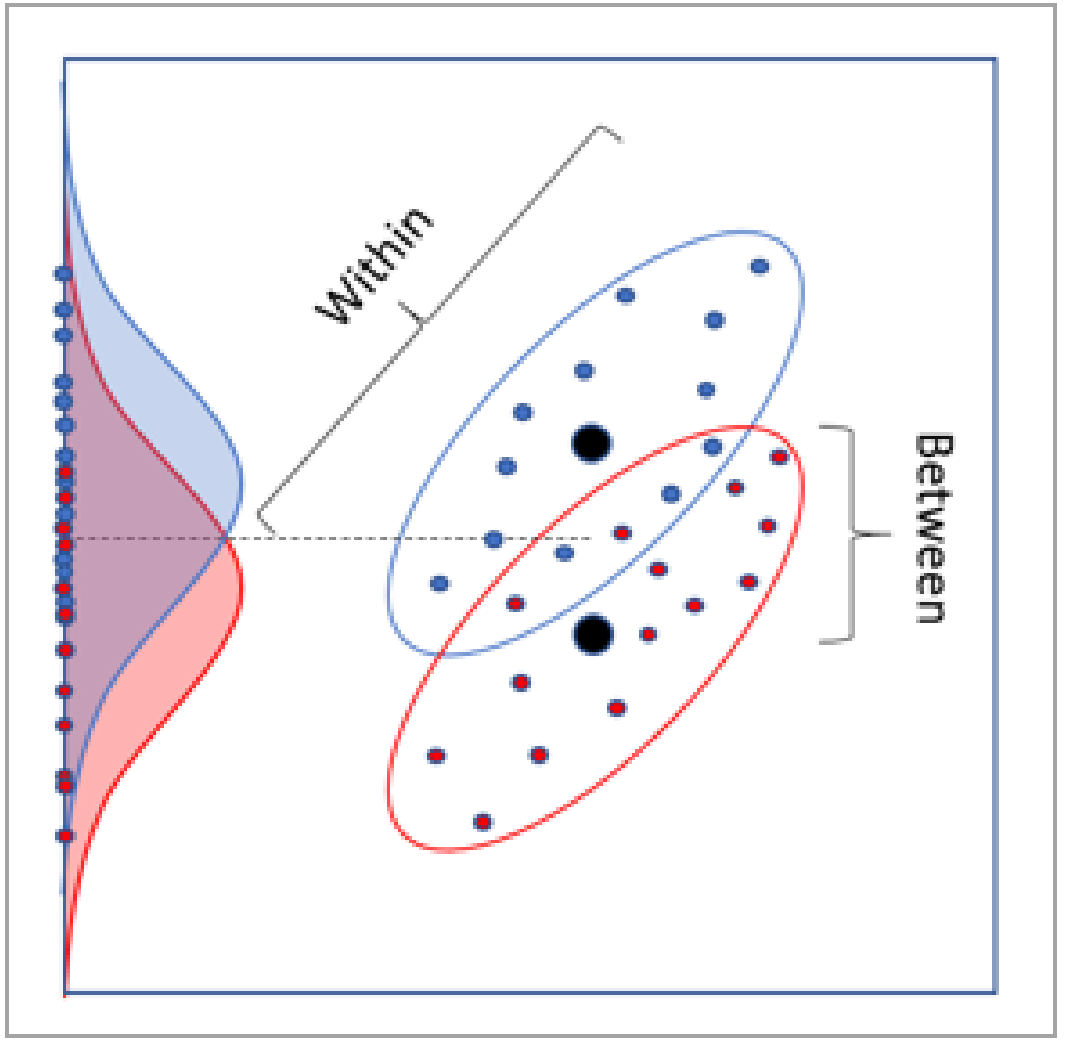
Assumptions of LDA

- LDA assumes that the data has a **Gaussian distribution** and that the **covariance matrices** of the different classes are **equal**.
- It also assumes that the data is **linearly separable**, meaning that a linear decision boundary can accurately classify the different classes.



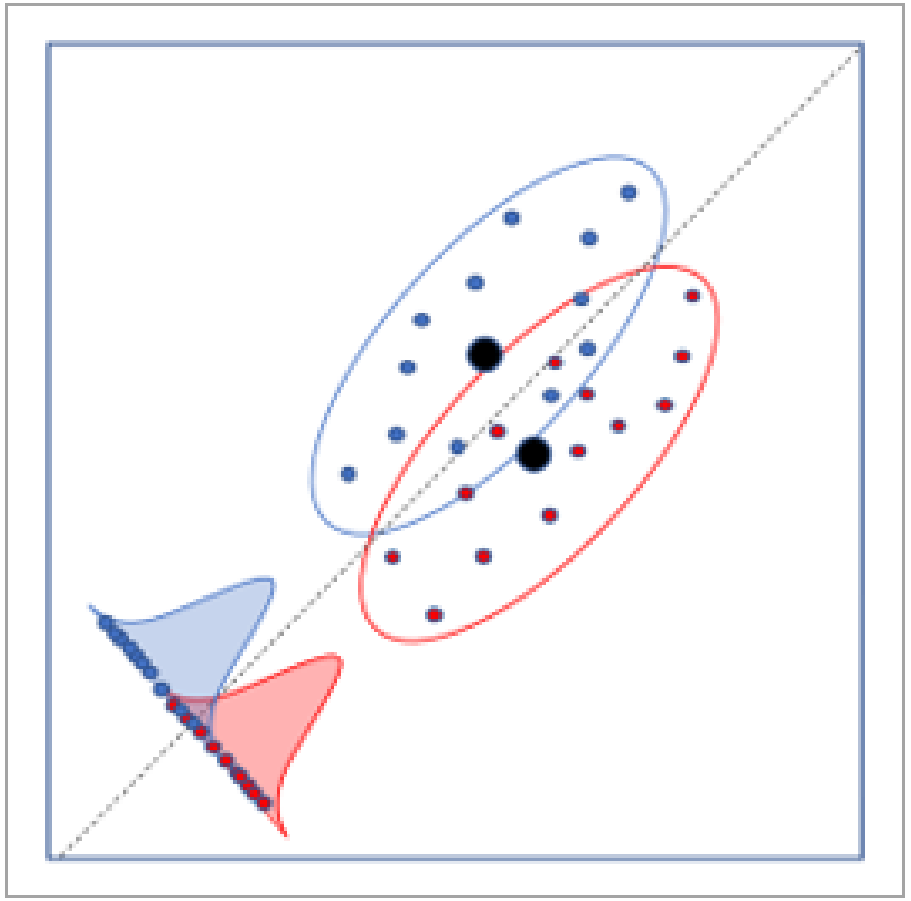
Basic Idea of LDA

- Consider the two datasets plotted in this figure.
- The x-axis and y-axis are the two continuous analysis fields, and the points are colored red or blue based on their category.
- The red and blue distributions are the distributions of the categories when projected to the y-axis.
- There is some separability in the distributions of the classes, but they have large overlap and are difficult to separate.
- A similar lack of separation occurs by projecting to the x-axis.



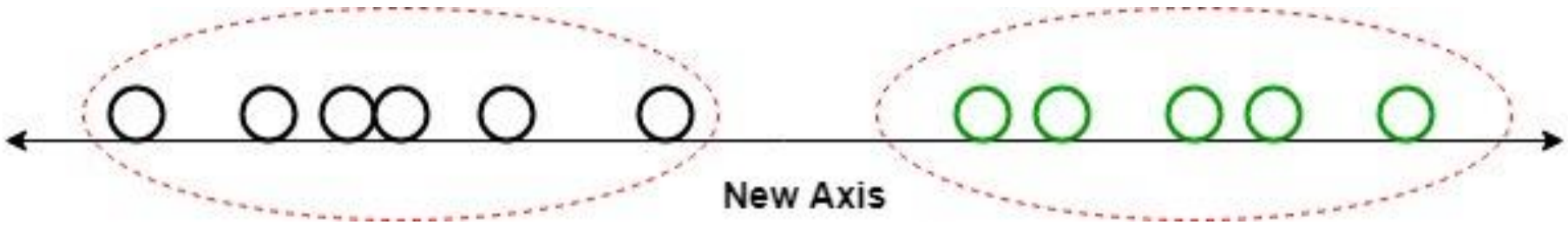
Basic Idea of LDA

- This figure shows the optimal axis rotation determined by LDA.
- This rotation results in the largest separation between the distributions of the categories, allowing the highest rate of classification of the category.
- Here, Linear Discriminant Analysis uses both axes (X and Y) to create a new axis and projects data onto a new axis in a way to **maximize the separation** of the two categories and hence, reduces the 2D graph into a 1D graph.



LDA Criteria

- Two criteria are used by LDA to create a new axis:
  - Maximize the distance between the means of the two classes.
  - Minimize the variation within each class.



Steps of LDA

1. Compute the class means of dependent variable

$$\mu_1 = \frac{1}{N_1} \sum_{x \in \omega_1} x$$

2. Derive the ~~covariance/~~scatter/coherence matrix of the class variable

$$S_1 = \sum_{x \in \omega_1} \frac{(x - \mu_1)(x - \mu_1)^T}{\text{N-1}}$$

3. Compute the within class-scatter matrix

$$S_w = S_1 + S_2$$

4. Compute the between class scatter matrix

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

5. Compute the Eigen values and eigen vectors from the within class and between class scatter matrix.

$$S_W^{-1} S_B w = \lambda w$$

Steps of LDA

6. Sort the values of eigne values and select the top k values.

7. Find the eigne vectors corresponds to the top K eigne vectors.

$$\left(S_W^{-1}S_B - \lambda I\right)\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = 0$$

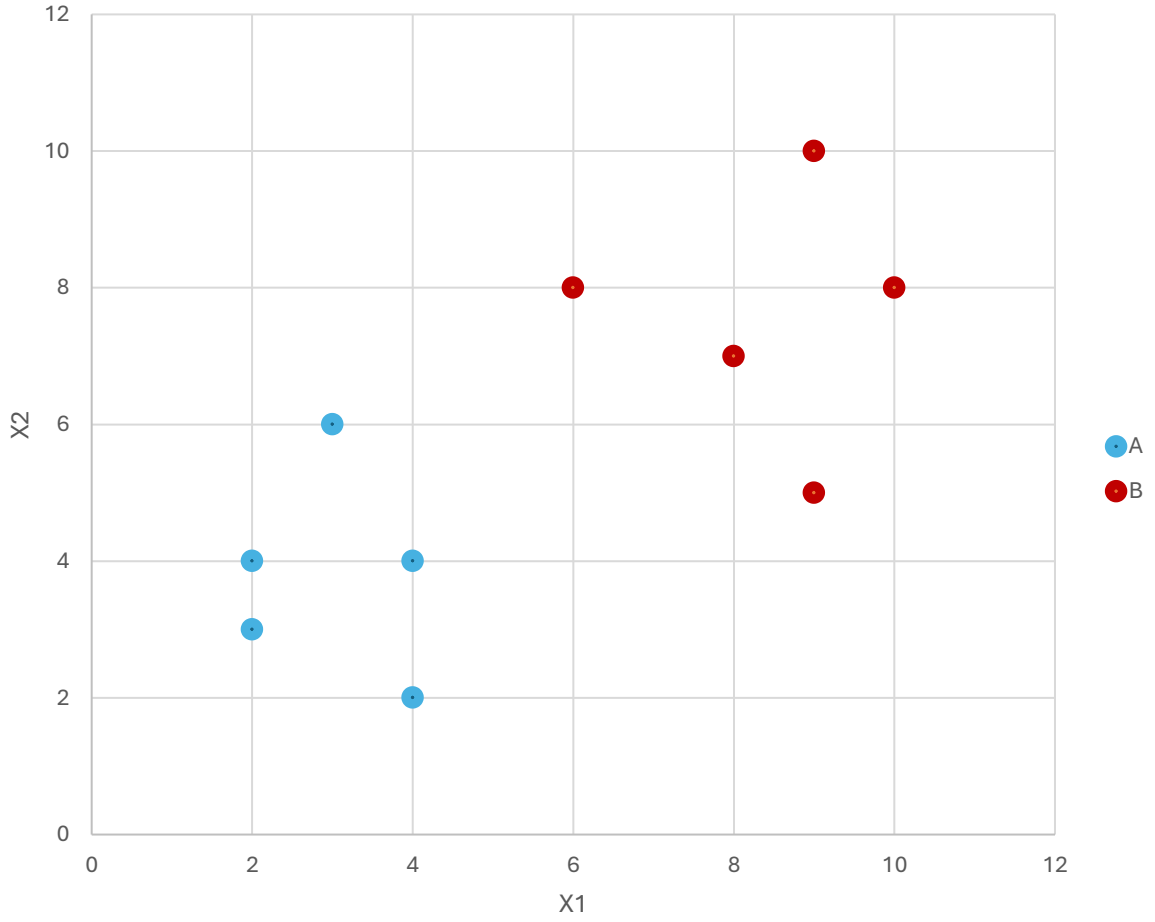
8.Obtain the LDA by taking the dot product of eigne vectors and original data.

LDA Solved Example

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Compute the Linear Discriminant projection for the following dataset

X1	X2	Class
4	2	A
2	4	A
2	3	A
3	6	A
4	4	A
9	10	B
6	8	B
9	5	B
8	7	B
10	8	B



LDA Solved Example

1. Compute the class means of dependent variable

X1	X2	Class
4	2	A
2	4	A
2	3	A
3	6	A
4	4	A
9	10	B
6	8	B
9	5	B
8	7	B
10	8	B

$$\mu_1 = \frac{1}{N_1}\sum_{x\in\omega_1} x = \frac{1}{5}\left[\begin{pmatrix}4\\2\end{pmatrix}+\begin{pmatrix}2\\4\end{pmatrix}+\begin{pmatrix}2\\3\end{pmatrix}+\begin{pmatrix}3\\6\end{pmatrix}+\begin{pmatrix}4\\4\end{pmatrix}\right] = \boxed{??}$$

$$\mu_2 = \frac{1}{N_2}\sum_{x\in\omega_2} x = \frac{1}{5}\left[\begin{pmatrix}9\\10\end{pmatrix}+\begin{pmatrix}6\\8\end{pmatrix}+\begin{pmatrix}9\\5\end{pmatrix}+\begin{pmatrix}8\\7\end{pmatrix}+\begin{pmatrix}10\\8\end{pmatrix}\right] = \boxed{??}$$

LDA Solved Example

2. Derive the covariance matrix of the class variable (class A)

X1	X2	Class
4	2	A
2	4	A
2	3	A
3	6	A
4	4	A
9	10	B
6	8	B
9	5	B
8	7	B
10	8	B

$$S_1 = \sum_{x \in \omega_1} \frac{(x-\mu_1)(x-\mu_1)^T}{\text{N-1}} = \left[\begin{pmatrix}4\\2\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]\left[\begin{pmatrix}4\\2\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]^T + \left[\begin{pmatrix}2\\4\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]\left[\begin{pmatrix}2\\4\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]^T$$

$$+ \left[\begin{pmatrix}2\\3\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]\left[\begin{pmatrix}2\\3\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]^T + \left[\begin{pmatrix}3\\6\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]\left[\begin{pmatrix}3\\6\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]^T + \left[\begin{pmatrix}4\\4\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]\left[\begin{pmatrix}4\\4\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]^T \Big/ \text{N-1}$$

$$= \begin{pmatrix} ?? \end{pmatrix}$$

LDA Solved Example

2. Derive the covariance matrix of the class variable (class B)

X1	X2	Class
4	2	A
2	4	A
2	3	A
3	6	A
4	4	A
9	10	B
6	8	B
9	5	B
8	7	B
10	8	B

$$S_2 = \sum_{x \in \omega_2} \frac{(x-\mu_2)(x-\mu_2)^T}{\text{N-1}} = \left[\begin{pmatrix}9\\10\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]\left[\begin{pmatrix}9\\10\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^T + \left[\begin{pmatrix}6\\8\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]\left[\begin{pmatrix}6\\8\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^T$$

$$+\left[\begin{pmatrix}9\\5\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]\left[\begin{pmatrix}9\\5\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^T + \left[\begin{pmatrix}8\\7\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]\left[\begin{pmatrix}8\\7\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^T + \left[\begin{pmatrix}10\\8\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]\left[\begin{pmatrix}10\\8\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^T \Big/ \text{N-1}$$

$$= \begin{pmatrix} ?? \end{pmatrix}$$

LDA Solved Example

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

3. Compute the within class-scatter matrix

$$S_w = S_1 + S_2 = \begin{pmatrix} 1 & -0.25 \\ -0.25 & 2.2 \end{pmatrix} + \begin{pmatrix} 2.3 & -0.05 \\ -0.05 & 3.3 \end{pmatrix}$$

$$= \begin{pmatrix} ?? \end{pmatrix}$$

4. Compute the between class scatter matrix

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

$$= \left[\begin{pmatrix} 3 \\ 3.8 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix} \right] \left[\begin{pmatrix} 3 \\ 3.8 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix} \right]^T$$

$$= \begin{pmatrix} -5.4 \\ -3.8 \end{pmatrix} \begin{pmatrix} -5.4 & -3.8 \end{pmatrix}$$

$$= \begin{pmatrix} ?? \end{pmatrix}$$

LDA Solved Example

5-Compute the Eigen values.

$$S_W^{-1}S_B w = \lambda w$$

$$\Rightarrow \left|S_W^{-1}S_B - \lambda I\right| = 0$$

$$\Rightarrow \left|\begin{pmatrix} 3.3 & -0.3 \\ -0.3 & 5.5 \end{pmatrix}^{-1}\begin{pmatrix} 29.16 & 20.52 \\ 20.52 & 14.44 \end{pmatrix} - \lambda\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right| = 0$$

$$\Rightarrow \left|\begin{pmatrix} 0.3045 & 0.0166 \\ 0.0166 & 0.1827 \end{pmatrix}\begin{pmatrix} 29.16 & 20.52 \\ 20.52 & 14.44 \end{pmatrix} - \lambda\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right| = 0$$

$$\Rightarrow \begin{vmatrix} 9.2213-\lambda & 6.489 \\ 4.2339 & 2.9794-\lambda \end{vmatrix}$$

$$= (9.2213-\lambda)(2.9794-\lambda) - 6.489 \times 4.2339 = 0$$

$$\Rightarrow \lambda^2 - 12.2007\lambda = 0 \Rightarrow \lambda(\lambda - 12.2007) = 0$$

$$\Rightarrow$$??

LDA Solved Example

5- Compute the Eigen Vectors

$$\left(S_W^{-1}S_B - \lambda I\right)\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = 0$$

6- Sort the values of eigne values and select the top k values.

$$w_1 = \begin{pmatrix} ?? \end{pmatrix}$$

$$w_2 = \begin{pmatrix} ?? \end{pmatrix} = w^*$$

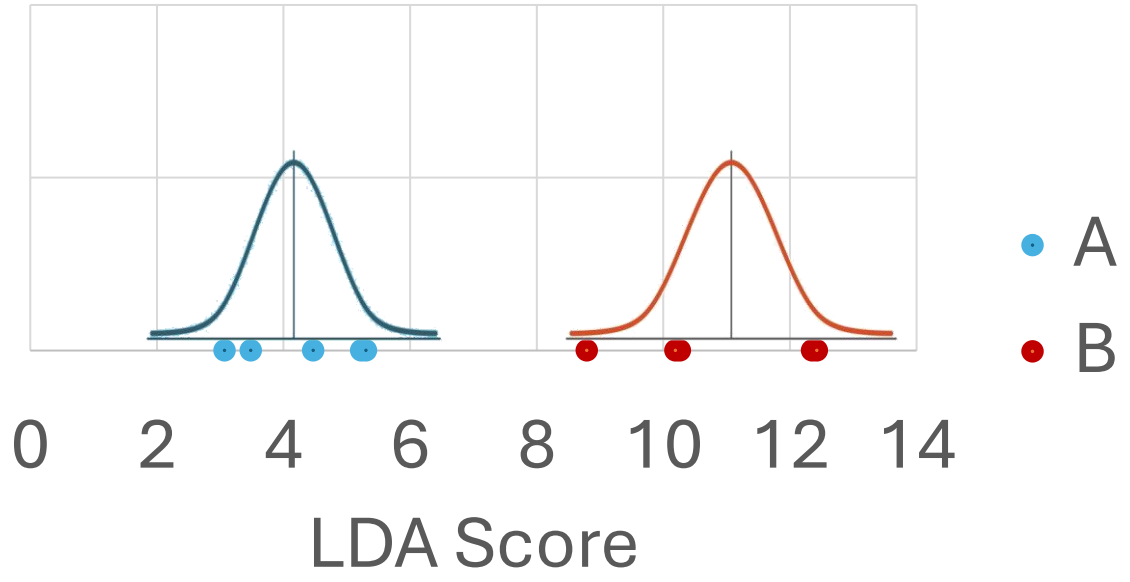
LDA Solved Example

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

8- Obtain the LDA by taking the dot product of eigne vectors and original data.

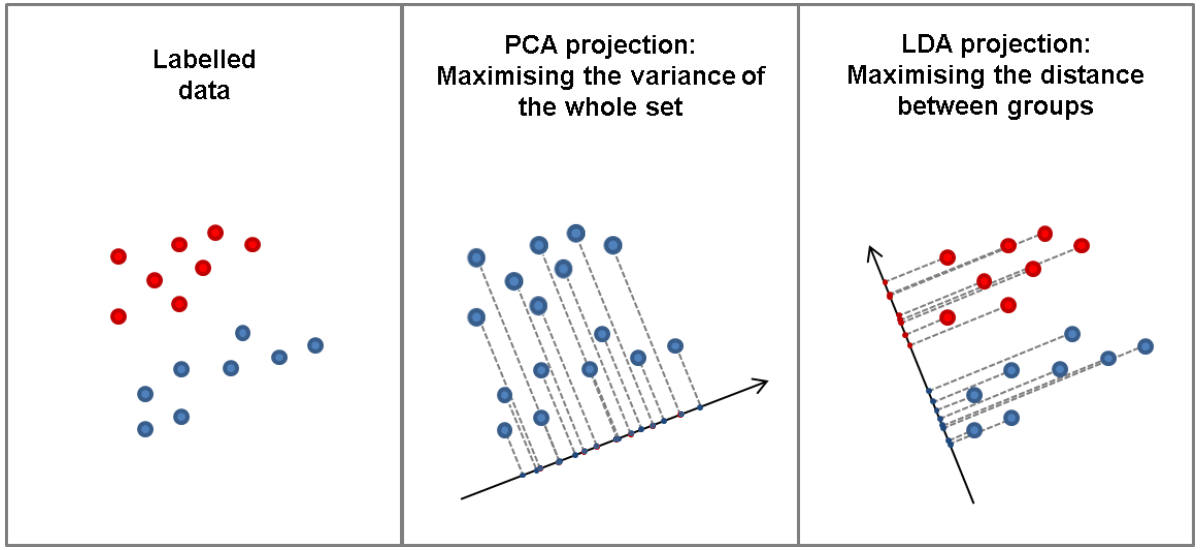
X1	X2	Class	1st LD
4	2	A	??
2	4	A	??
2	3	A	??
3	6	A	??
4	4	A	??
9	10	B	??
6	8	B	??
9	5	B	??
8	7	B	??
10	8	B	??

$$w_2 = \begin{pmatrix} 0.9088 \\ 0.4173 \end{pmatrix} = w^*$$



PCA Vs LDA

- **PCA**: Reduces dimensionality by maximizing **variance**. It is **unsupervised** and works with the data as a whole, without considering class labels.
- **LDA**: Reduces dimensionality by maximizing **class separability**. It is **supervised** and uses class labels to find directions that separate the classes.



PCA

- **Dimensionality Reduction**: Use PCA when you need to **reduce the number of features** while retaining as much variance (information) as possible.
- **Data Visualization**: Ideal for **visualizing high-dimensional data** by projecting it to 2D or 3D for exploratory analysis.
- **Noise Reduction**: Helps to **remove noise** and less important features that have low variance.
- **Improving Computational Efficiency**: Speeds up machine learning models by reducing the number of input features.
- **Unsupervised Learning**: Use when class labels are not available or class separation is not the goal.
- **Feature Compression**: Retains important patterns while compressing the feature set for efficient modeling.
- **Identifying Structure**: Reveals the **underlying structure** and relationships in the data, useful for pattern discovery.

LDA

- Labeled Data for Classification: Works with data where you have class labels for each sample.
- Maximizing Class Separation: Ideal when the goal is to separate classes as effectively as possible.
- Dimensionality Reduction with Class Information: Reduces dimensions while preserving the ability to distinguish between classes.
- Normally Distributed Features: Assumes data from each class follows a Gaussian distribution.
- Simple and Fast Classifier: Suitable for small to medium datasets with fewer features.
- Multiclass Classification: Effective for problems with more than two classes.
- Handles Multicollinearity: Combines correlated features efficiently.

PCA Vs LDA

Aspect	PCA	LDA
Purpose	Dimensionality reduction by capturing variance	Dimensionality reduction by maximizing class separation
Type	Unsupervised (no class labels needed)	Supervised (requires class labels)
Focus	Retains maximum variance in data	Maximizes separation between different classes
Data Requirement	Can be used with unlabeled data	Requires labeled data
Assumptions	Assumes linear relationships and high variance	Assumes classes follow Gaussian distributions
Use Cases	Visualization, noise reduction, feature compression	Classification, improving class separability
Dimensionality Reduction Goa	Preserve as much variance as possible	Maintain ability to distinguish between classes

Quiz

https://forms.office.com/r/8aCKveBbvV

Contents

Introduction

- Nearest neighbor algorithms are among the "simplest" supervised machine learning algorithms.
- Well studied in the field of pattern recognition.
- Used as a predictive performance benchmark when you are trying to develop more sophisticated models.
- K-NN is also called a lazy learning algorithm.

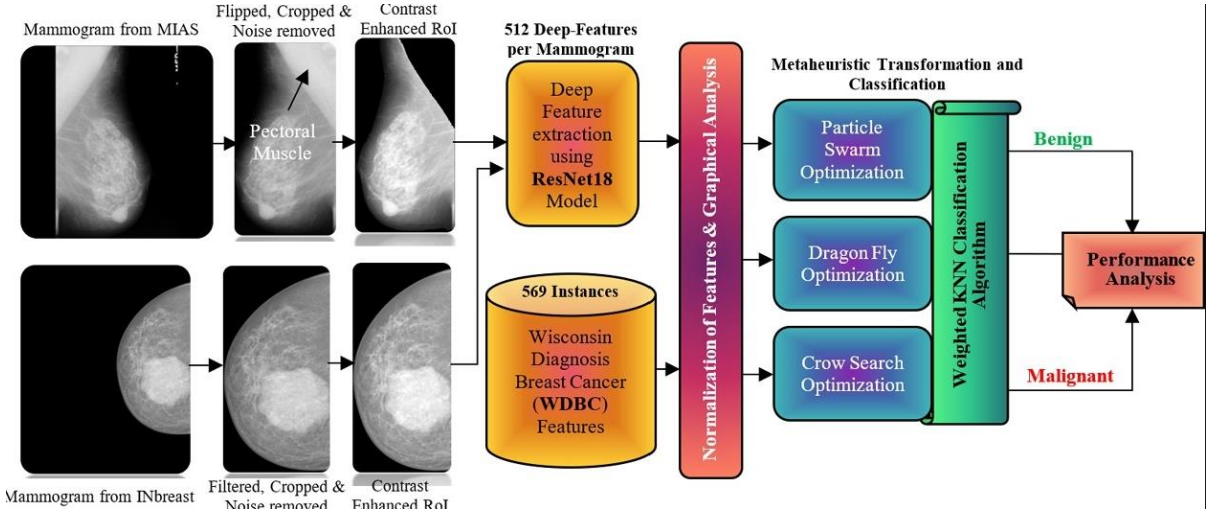
Applications

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Face Recognition

https://github.com/hanifizzudinrahman/Face-Recognition-using-Dlib-and-KNN-Classifier

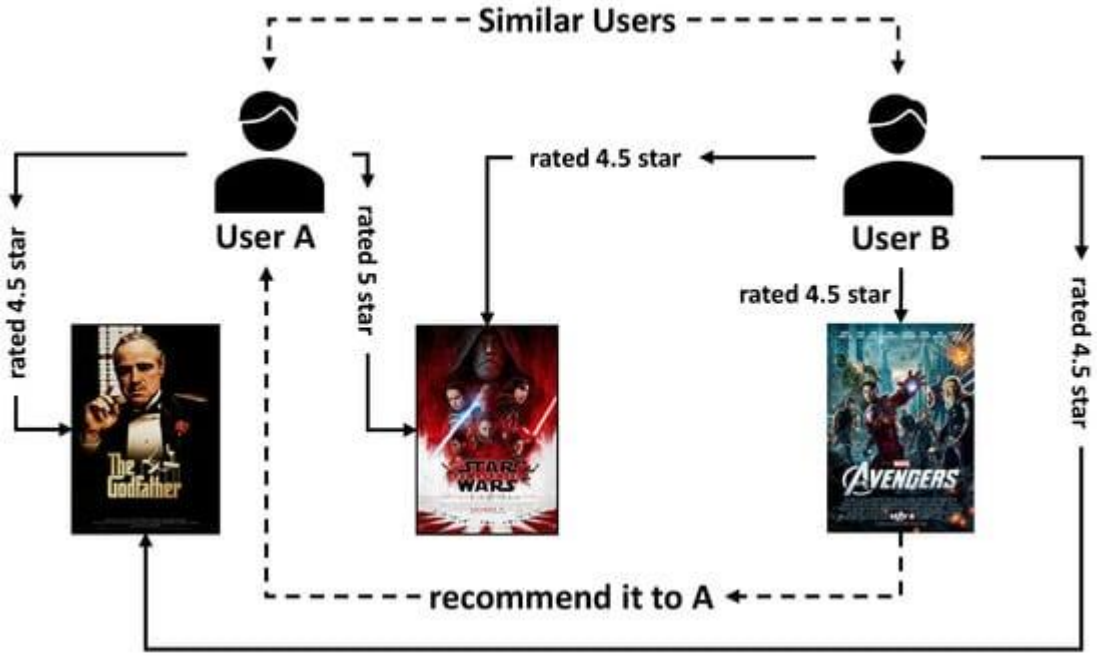
Medical image classification



https://doi.org/10.1016/j.irbm.2022.100749

Applications

Recommender



Applied Computing and Informatics (2016) 12, 90–108
Saudi Computer Society, King Saud University
Applied Computing and Informatics
(http://computer.org.sa)
www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

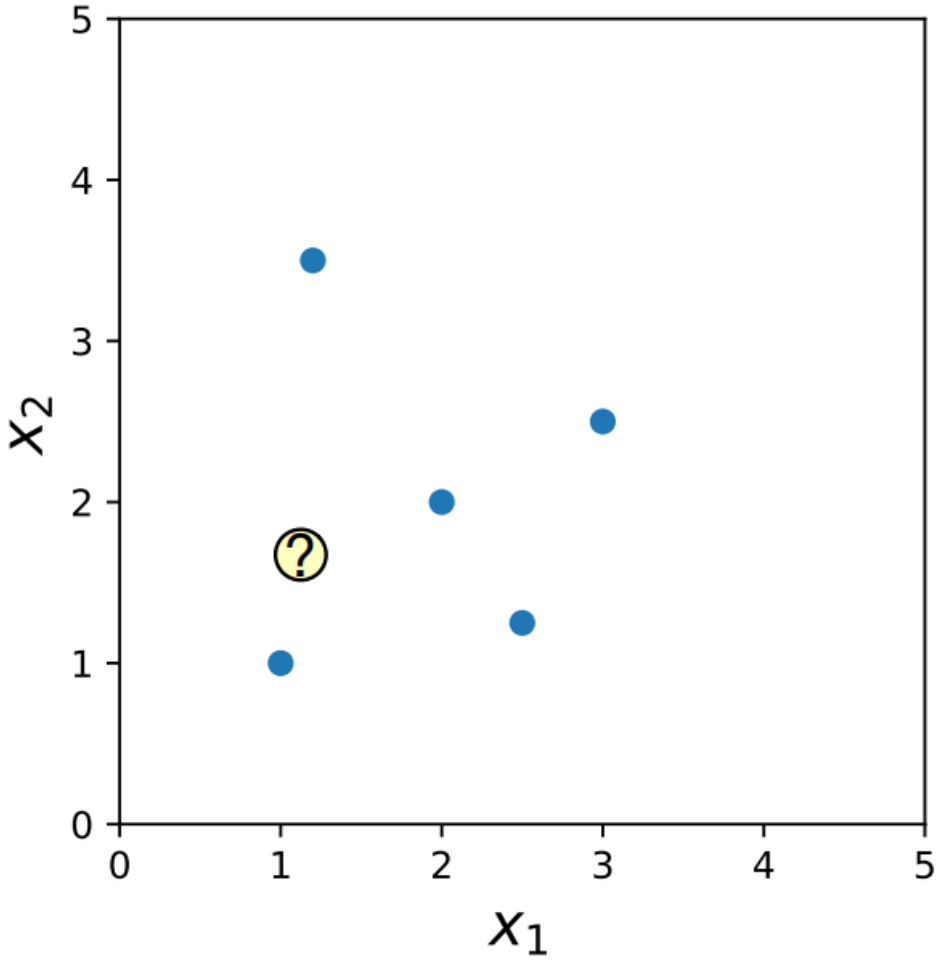
Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method

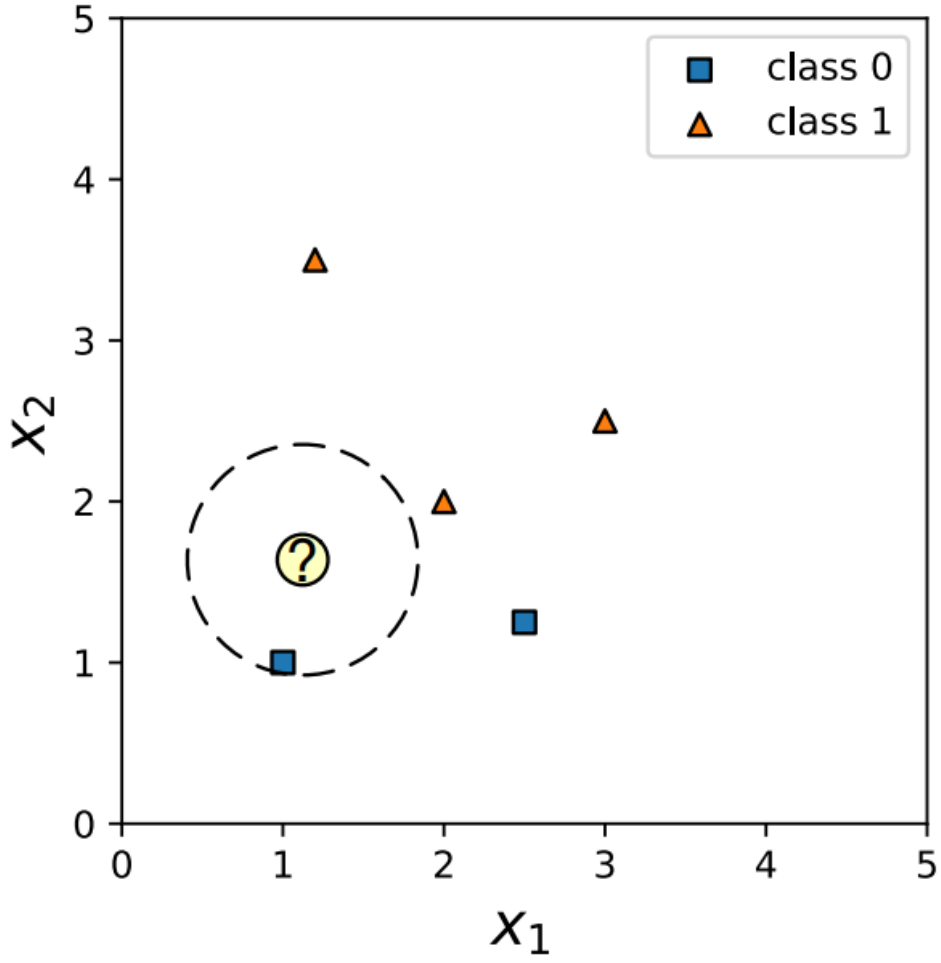
D.A. Adeniyi, Z. Wei, Y. Yongquan *

The major problem of many on-line web sites is the presentation of many choices to the client at a time; this usually results to strenuous and time consuming task in finding the right product or information on the site. In this work, we present a study of automatic web usage data mining and recommendation system based on current user behavior through his/her click stream data on the newly developed Really Simple Syndication (RSS) reader website, in order to provide relevant information to the individual without explicitly asking for it. **The K-Nearest-Neighbor (KNN) classification** method has been trained to be used on-line and in Real-Time to identify clients/visitors click stream data, matching it to a particular user group and recommend a tailored browsing option that meet the need of the specific user at a particular time. [...]

https://doi.org/10.3390/bdcc7020106

Basic Idea: Nearest Neighbor





Nearest Neighbor Algorithm

Given: $\langle \mathbf{x}^{[i]}, y^{[i]} \rangle \in \mathcal{D} \quad (|\mathcal{D}| = n)$

$\langle \mathbf{x}^{[q]}, ??? \rangle$

Predict: $f(\mathbf{x}^{[q]})$

Algorithm:

closest_point := None

closest_distance := ∞

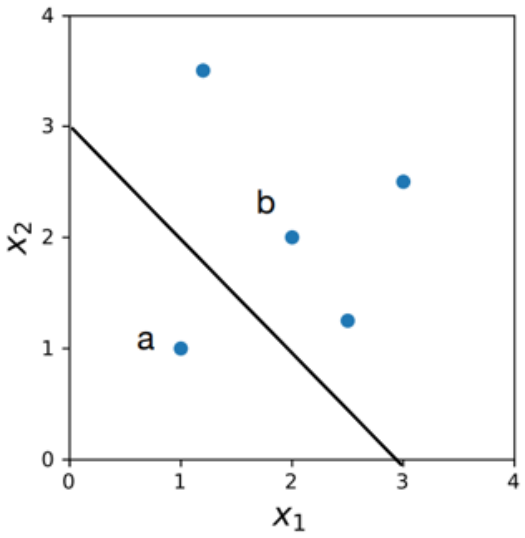
- for $i = 1, \ldots, n$:
  - current_distance := $d(\mathbf{x}^{[i]}, \mathbf{x}^{[q]})$
  - if current_distance < closest_distance:
    - closest_distance := current_distance
    - closest_point := $\mathbf{x}^{[i]}$
- return f(closest_point)

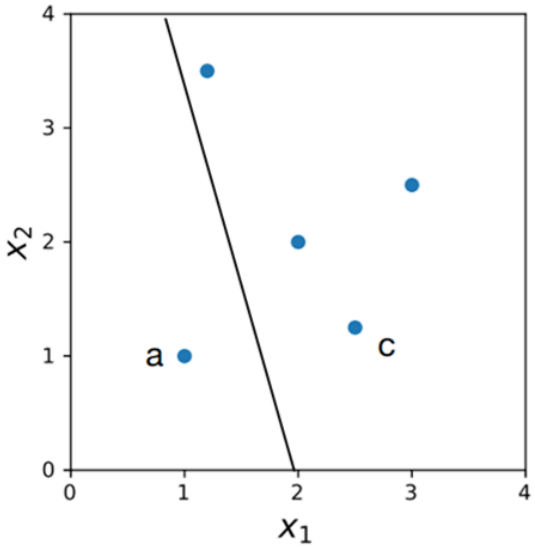
query point

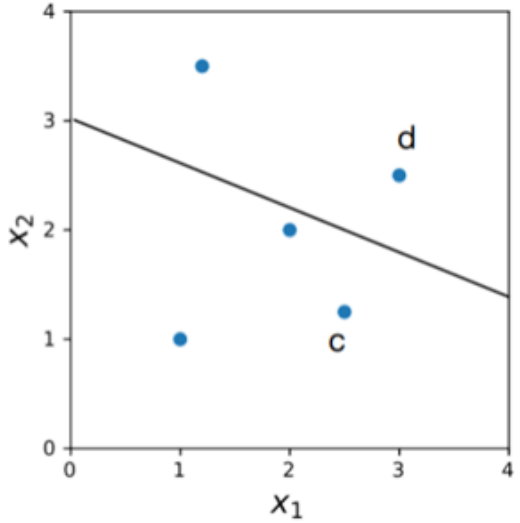
Would return the closest distance, point and label

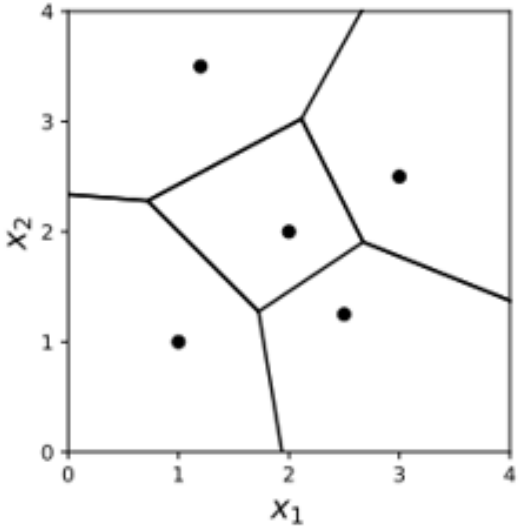
Nearest Neighbor Decision boundary:

- The boundary between regions of input space assigned to different categories.
- What boundary line can be used to separate between a and b.



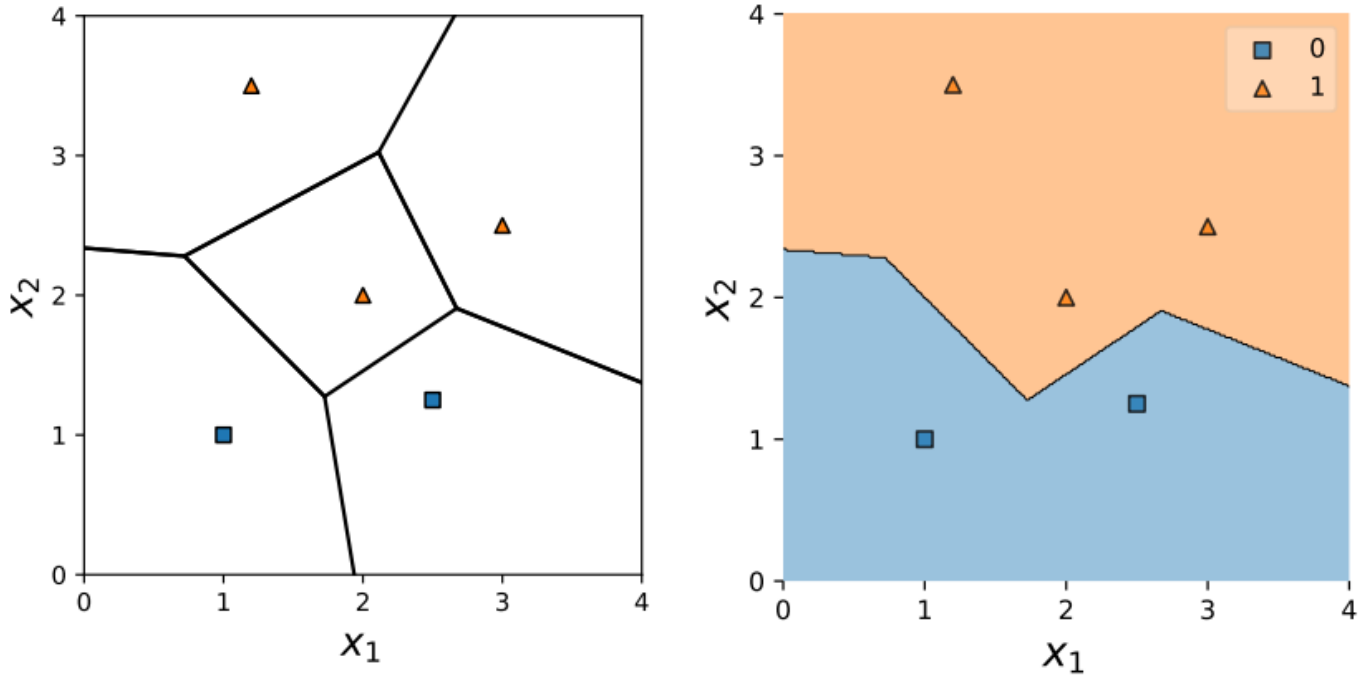




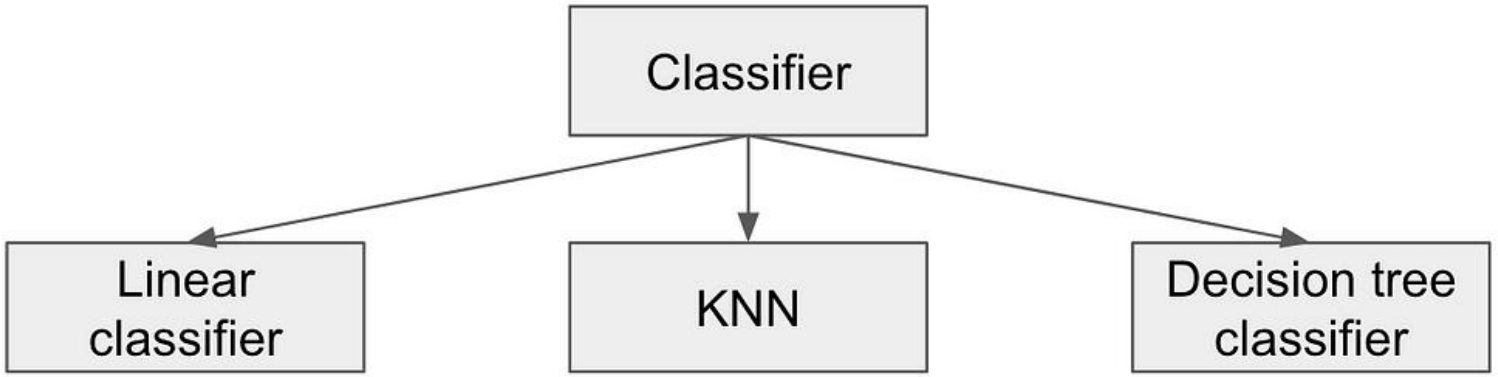


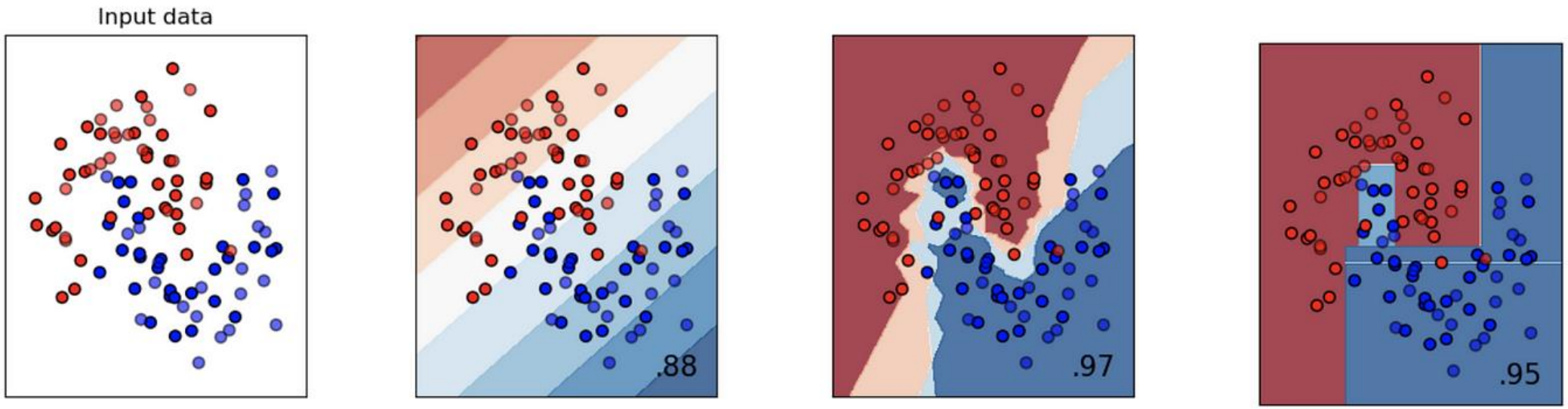
Nearest Neighbor Decision boundary:

- Decision boundaries: Voronoi diagram visualization
  - show how input space divided into classes
  - each line segment is equidistant between two points of opposite classes



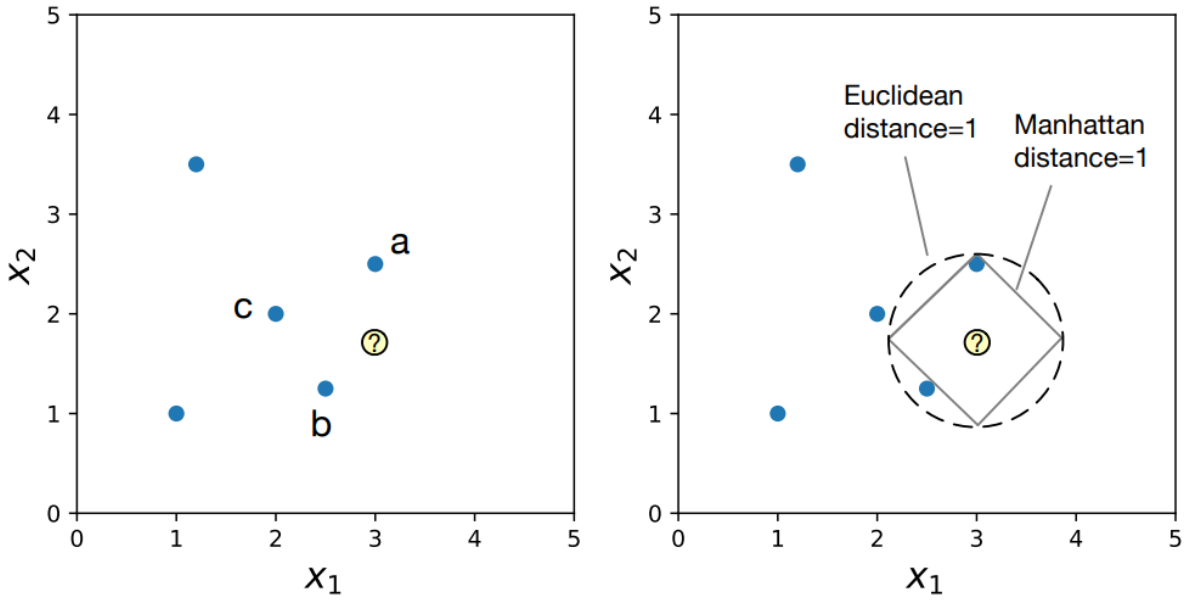
Decision boundary:

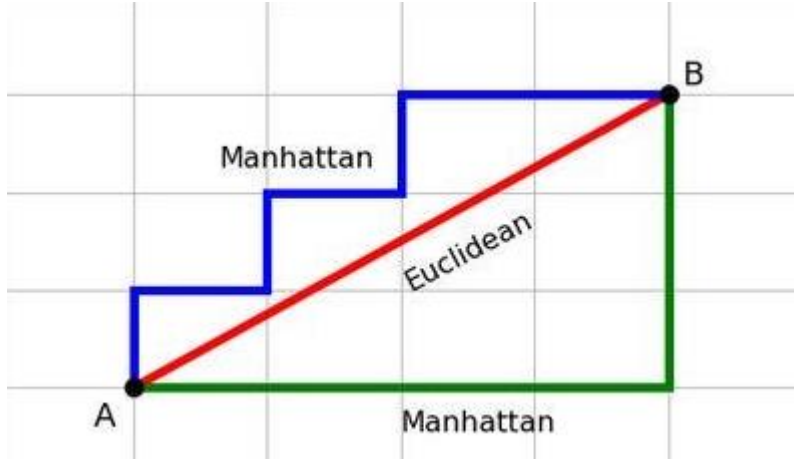




Distance Measurement in Nearest Neighbor

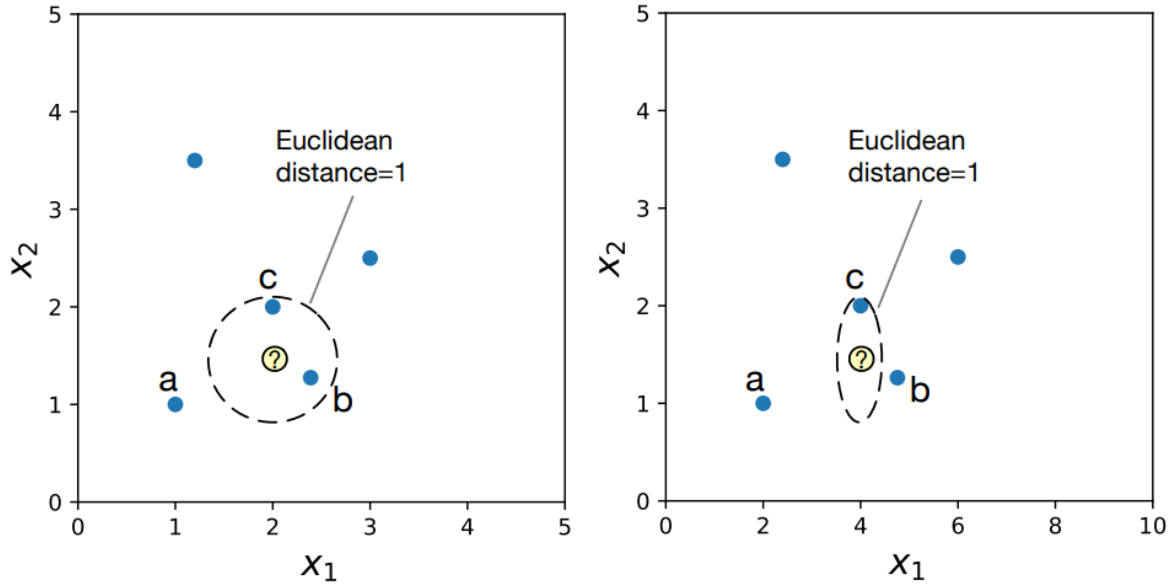
- KNN classifies a data point based on the majority class of its **k nearest neighbors**.
- The choice of **distance metric** determines how "closeness" is measured.





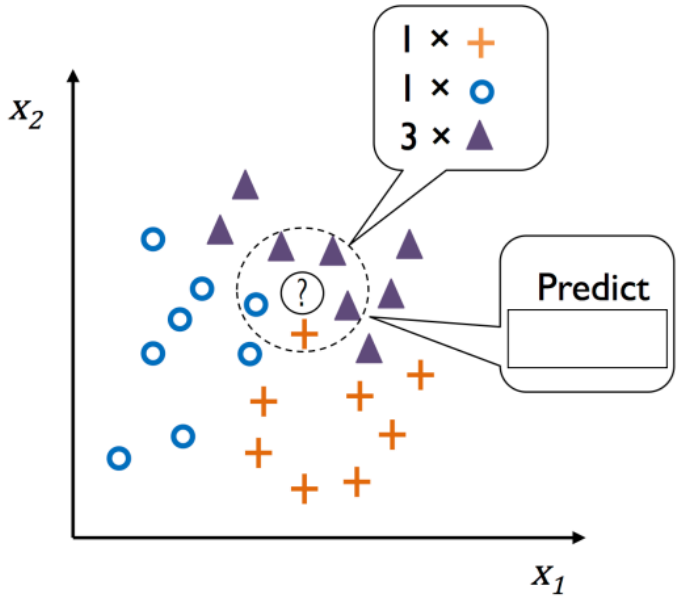
Scaling in Nearest Neighbor

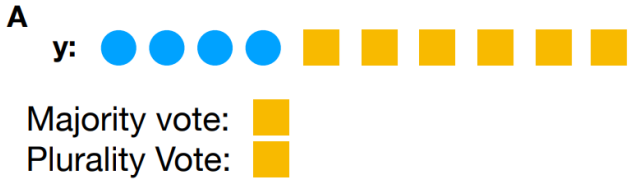
- k-NN relies on distance metrics (e.g., Euclidean distance).
- Features with larger magnitudes dominate the distance calculation, leading to biased results.



KNN- More than one Neighbor

- 1-NN is a special case of k-NN
- When k=1, k-NN becomes Nearest Neighbor (1-NN).
- Using multiple neighbors (k>1) reduces sensitivity to outliers and random variations.
- 1-NN: Low bias, high variance (prone to overfitting).
- Higher k: Increased bias, lower variance (smoother decision boundary).





KNN- Algorithm

Variant A

$\mathcal{D}_k := \{\}$

while $|\mathcal{D}_k| <$ k:

- `closest_distance` $:= \infty$
- for $i = 1, ..., n, \quad \forall i \notin \mathcal{D}_k$:
  - `current_distance` $:= d(\mathbf{x}^{[i]}, \mathbf{x}^{[q]})$
  - if `current_distance` $<$ `closest_distance`:
    - `closest_distance` $:=$ `current_distance`
    - `closest_point` $:= \mathbf{x}^{[i]}$
- add `closest_point` to $\mathcal{D}_k$

Variant B

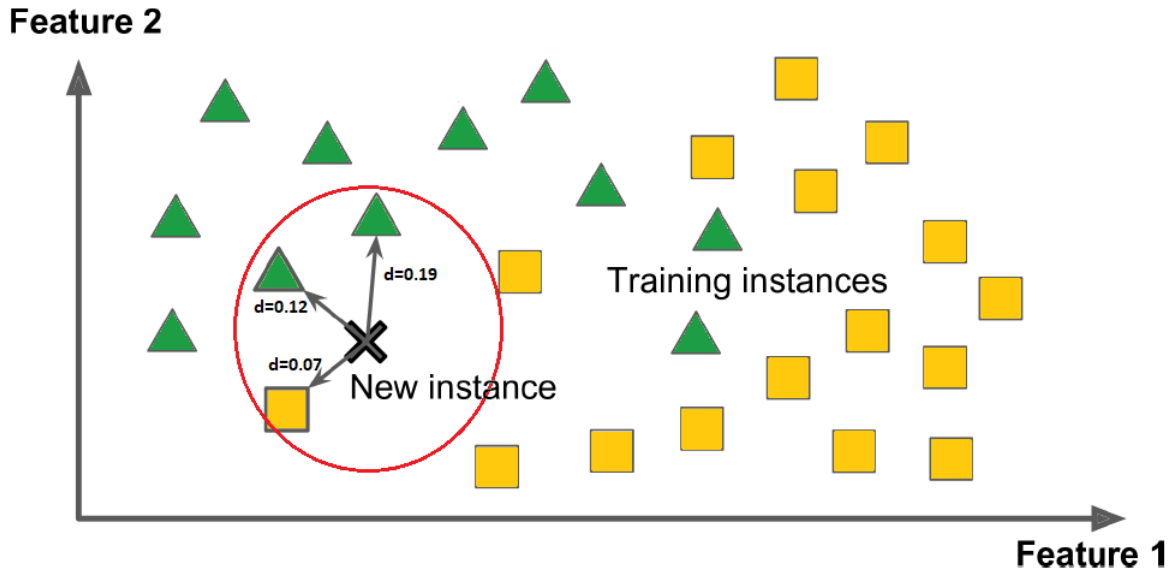
$\mathcal{D}_k := \mathcal{D}$

while $|\mathcal{D}_k| > k$:

- `largest_distance` $:= 0$
- for $i = 1, ..., n \quad \forall i \in \mathcal{D}_k$:
  - `current_distance` $:= d(\mathbf{x}^{[i]}, \mathbf{x}^{[q]})$
  - if `current_distance` $>$ `largest_distance`:
    - `largest_distance` $:=$ `current_distance`
    - `farthest_point` $:= \mathbf{x}^{[i]}$
- remove `farthest_point` from $\mathcal{D}_k$

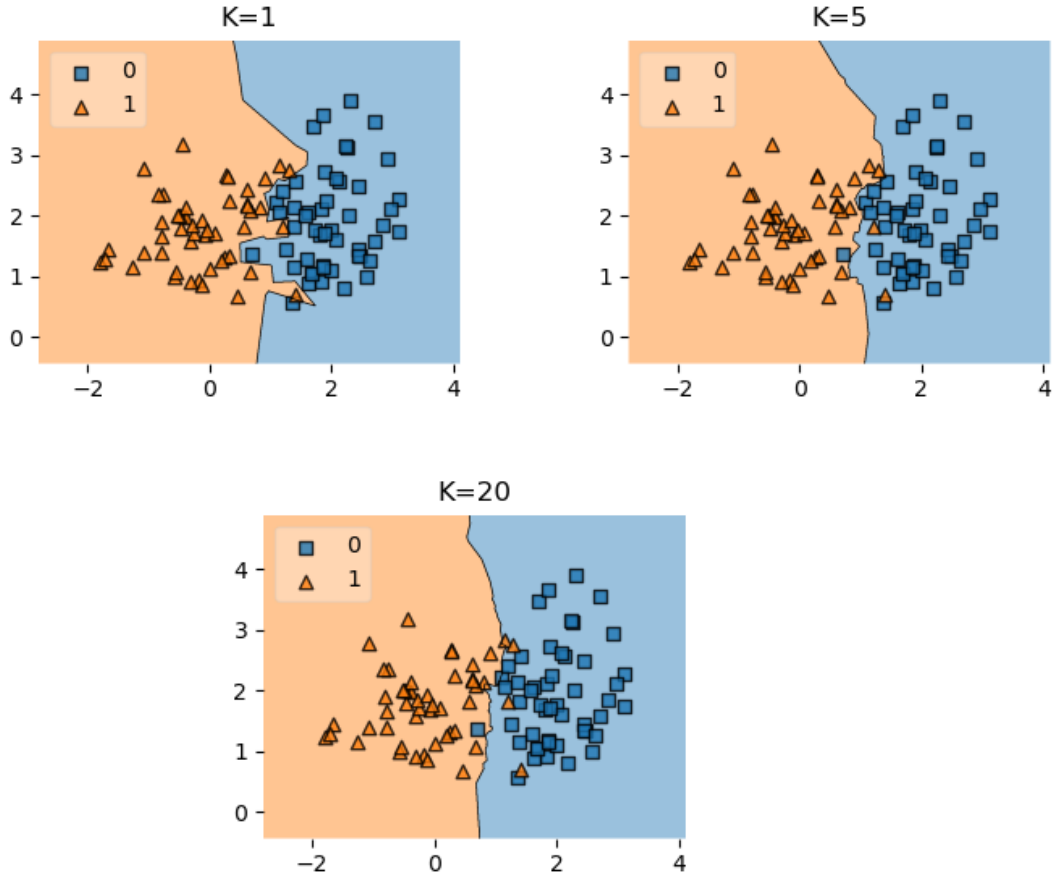
Weighted k-NN

- In weighted KNN, the nearest K points are assigned a weight.
- Then we give more weight to the points which are nearby and less weight to the points which are farther away.
- The simple function which is used is the inverse distance function (1 / Euclidian distance) which implies that as the distance increases weight decreases and as the distance decreases, weight increases.



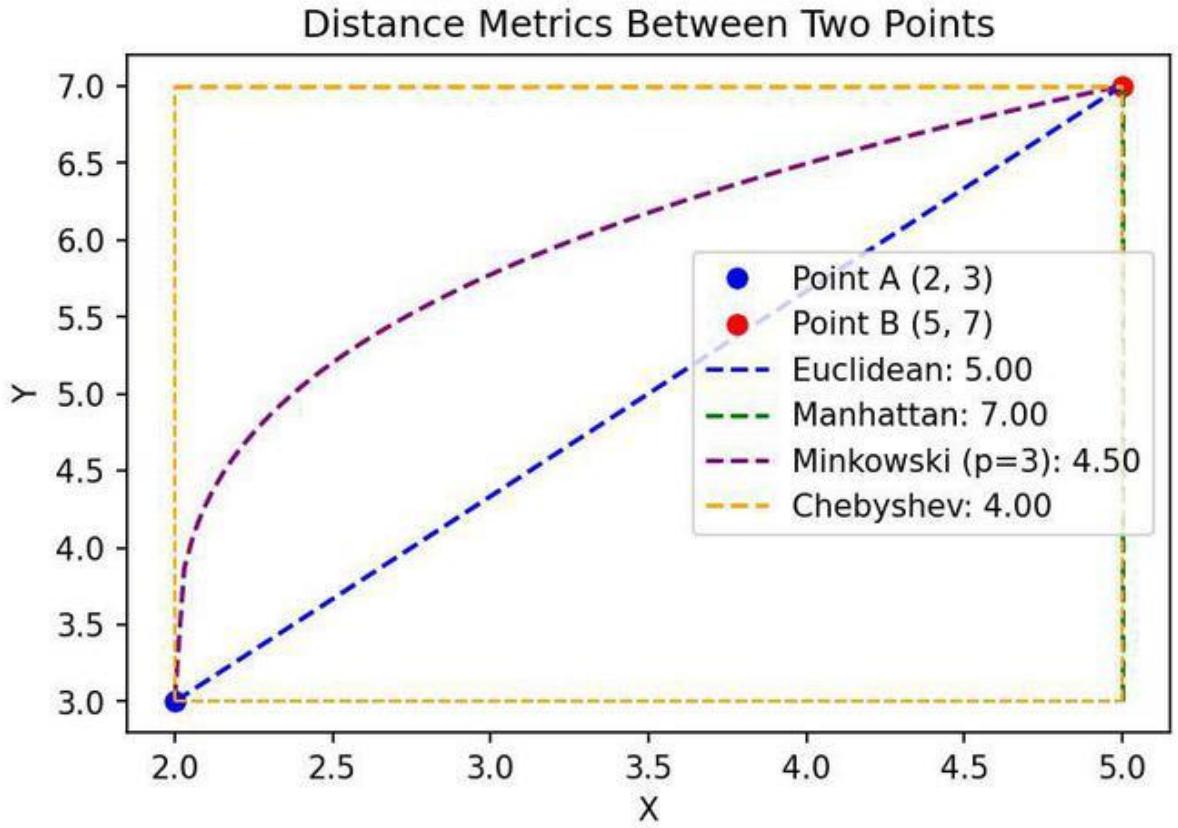
Hyperparameters: Value of K

- Determines how many nearest neighbors influence classification/regression.
- Small k → More flexible, can overfit (high variance).
- Large k → Smoother decision boundary, less overfitting, but may underfit (high bias).
- Common choices: Odd values (to avoid ties), typically tested in range k=3 to k=15.



Hyperparameters: Distance Metrics

- Euclidean Distance
- Manhattan Distance
- Minkowski Distance
- Chebyshev Distance
- Cosine Similarity



Hyperparameters: Distance Metrics

Distance Metric	When to Use	Use Case Scenario
Euclidean Distance	- Continuous numerical data. - When the data is well-scaled.	- Predicting house prices based on square footage and number of bedrooms. - Image recognition where pixel values are continuous features.
Manhattan Distance	- Data with features on a grid (e.g., city streets). - When data is less sensitive to outliers.	- Delivery routing for trucks following city grids. - Robot navigation through a grid with restricted movement (i.e., only vertical or horizontal). - Infrastructure planning and transportation networks.
Minkowski Distance	- When you need a flexible metric that can represent different distances. - When you want to tune the parameter 'p' for customization.	- Analyzing weather data like temperature, humidity, and wind speed to predict likelihood of rain. - Choosing between Euclidean or Manhattan depending on the problem's spatial relationship.[1][5]
Chebyshev Distance	- When the maximum difference between coordinates is important. - When features represent movements along a grid with equal importance.	- In a board game, measuring the maximum number of moves a piece can make in any direction. - Robot movement where diagonal and straight moves are equally important (e.g., chess, checkers).
Cosine Similarity	- When the direction of the vectors is more important than their magnitude.	- Text analysis, image retrieval, and recommendation systems. [Note: Cosine similarity is not a distance metric but a similarity measure, yet it is often used in similar contexts.]

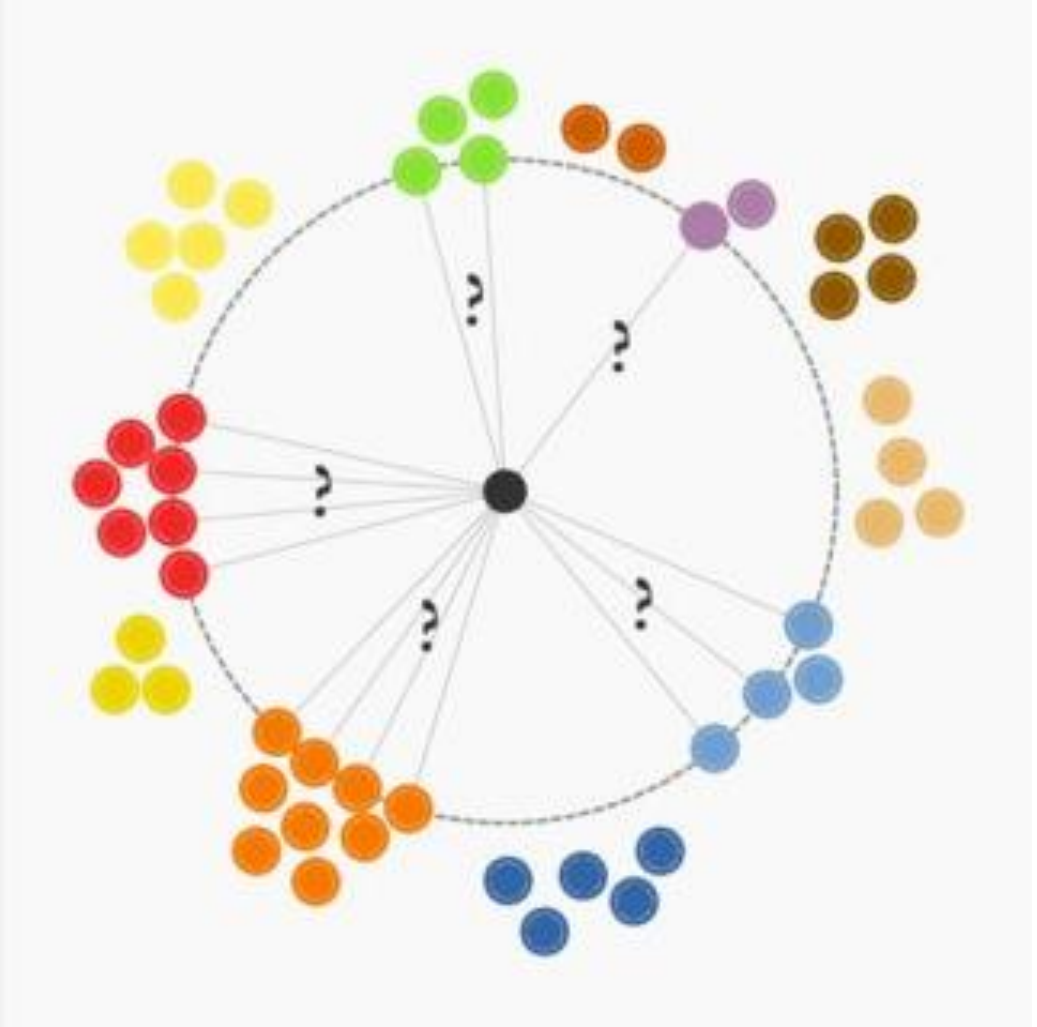
https://www.geeksforgeeks.org/how-to-choose-the-right-distance-metric-in-knn/

Hyperparameters: Weighting Scheme

- Defines how neighbors contribute to the prediction.
- Uniform: All neighbors have equal weight (default).
- Distance based: Closer points get higher weight (e.g., w=1/d)
- Custom Weights: User-defined weighting function.

Curse of Dimensionality

- Affects all learners but especially bad for nearest-neighbour.
- k-NN usually works well when the number of dimensions is small but things fall apart quickly as goes up.
- If there are many irrelevant attributes, k-NN is hopelessly confused because all of them contribute to finding similarity between examples.
- With enough irrelevant attributes the accidental similarity swamps out meaningful similarity and k-NN is no better than random guessing.



Benefits and Challenges of k-NN

- Easy to understand, interpret.
- Simple hyperparameter controlling the fundamental tradeoff.
- Can learn very complex functions given enough data.
- Lazy learning: Takes no time to fit

- Can be potentially be VERY slow during prediction time, especially when the training set is very large.
- Often not that great test accuracy compared to the modern approaches.
- It does not work well on datasets with many features or where most feature values are 0 most of the time (sparse datasets).