# Principle Component Analysis

**Sachin Tripathi**

IIT(ISM), Dhanbad

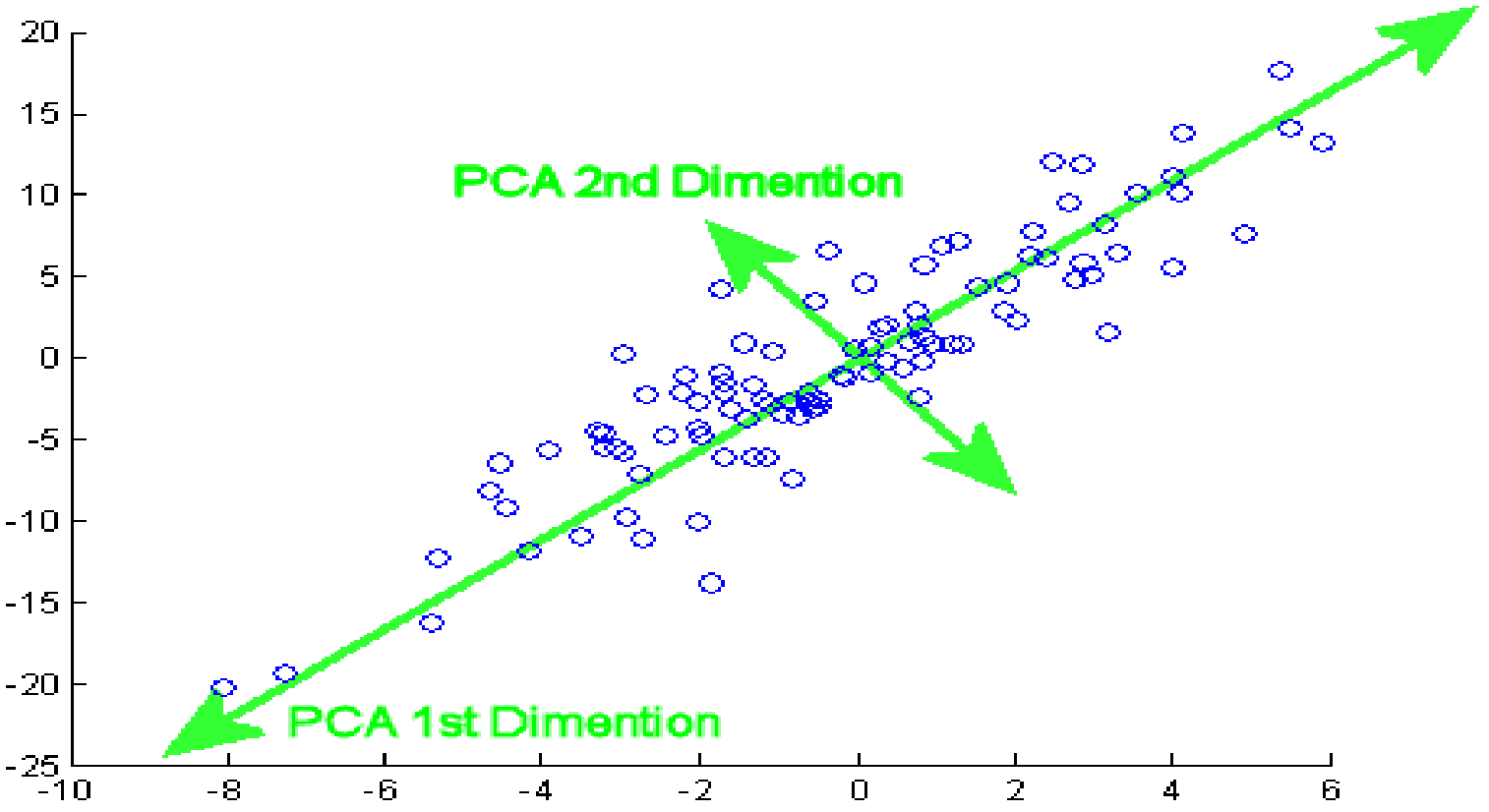
# What is PCA ?

- PCA (Principal Component Analysis) is a dimensionality reduction technique used in data analysis and machine learning. It helps you to reduce the number of features in a dataset while keeping the most important information.

- It changes your original features into new features these new features don’t overlap with each other

- PCA is commonly used for data preprocessing

# Basic Idea


PCA 2nd Dimention
PCA 1st Dimention
20
15
10
5
0
-5
-10
-15
-20
-25
-10
-8
-6
-4
-2
0
2
4
6

- It helps to remove redundancy, improve computational efficiency and make data easier to visualize and analyze especially when dealing with high-dimensional data.
- It takes data with features(Columns) and find new axes (Principal Components) along which data varies the most.
- These new axes are uncorrelated and are sorted by importance (how much variance they capture)
- PCA allows to compress data in to fewer features without losing much information.

# How PCA Works

- PCA uses linear algebra to transform data into new features called principal components.
- It finds these by calculating eigenvectors (directions) and eigenvalues (importance) from the covariance matrix.
- PCA selects the top components with the highest eigenvalues and projects the data onto them simplify the dataset.

➢ **Note:** It prioritizes the directions where the data varies the most because more variation = more useful information.

# Steps Used

- Standardize the Data
- Calculate Covariance Matrix
- Find the Principal Components
- Pick the Top Directions & Transform Data

# Why standardize features in PCA

- Real datasets often have features in different units or scales. Example: height in cm (100–200), weight in kg (30–100)
- Variance is influenced by scale → larger-scale features dominate PCA axes if not standardized.

# Standardize the Data

- Different features may have different units and scales like salary vs. age. To compare them fairly PCA first standardizes the data by making each feature have:
  - A mean of 0
  - A standard deviation of 1

$$Z = \frac{X-\mu}{\sigma}$$ i.e. $$x_{ij}^{scaled} = \frac{x_{ij} - \bar{x}_j}{s_j}$$

where:

- $\mu$ is the mean of independent features $\mu=\{\mu1,\mu2,\cdots,\mu m\}$
- $\sigma$ is the standard deviation of independent features $\sigma=\{\sigma1,\sigma2,\cdots,\sigma m\}$

# Proof for Standardization

$$\text{Mean of } X_{\text{centered}} = \frac{(x_1 - \bar{X}) + (x_2 - \bar{X}) + \ldots + (x_n - \bar{X})}{n}$$

$$= \frac{(x_1 + x_2 + \ldots + x_n) - n\bar{X}}{n}$$

$$= \frac{n\bar{X} - n\bar{X}}{n} = 0$$

- **Original SD formula:**

$$SD(X) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

- After standardization:

$$Z_i = \frac{X_i - \bar{X}}{SD(X)}$$

Compute the SD of $Z$:

$$SD(Z) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(Z_i - \bar{Z})^2}$$

- But $\bar{Z} = 0$ (because we subtracted the mean)

$$SD(Z) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{X_i - \bar{X}}{SD(X)}\right)^2} = \frac{1}{SD(X)}\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2} = \frac{SD(X)}{SD(X)} = 1$$

So after dividing by SD, the new feature has **standard deviation = 1.**

# Calculate Covariance Matrix

- Next PCA calculates the covariance matrix to see how features relate to each other whether they increase or decrease together. The covariance between two features $x_1$ and $x_2$ is:

$$cov(x1, x2) = \frac{\sum_{i=1}^{n}(x1_i - \bar{x1})(x2_i - \bar{x2})}{n-1}$$

Where:

- $\bar{x}_1 \, and \, \bar{x}_2$ are the mean values of features $x_1 \, and \, x_2$
- $n$ is the number of data points

The value of covariance can be positive, negative or zeros.

# Find the Principal Components

- PCA identifies **new axes** where the data spreads out the most:

  **1st Principal Component (PC1):** The direction of maximum variance (most spread).

  **2nd Principal Component (PC2):** The next best direction, *perpendicular to PC1* and so on.
- These directions come from the **eigenvectors** of the covariance matrix and their importance is measured by **eigenvalues**.
- For a square matrix A an **eigenvector** X (a non-zero vector) and its corresponding **eigenvalue** λ satisfy:

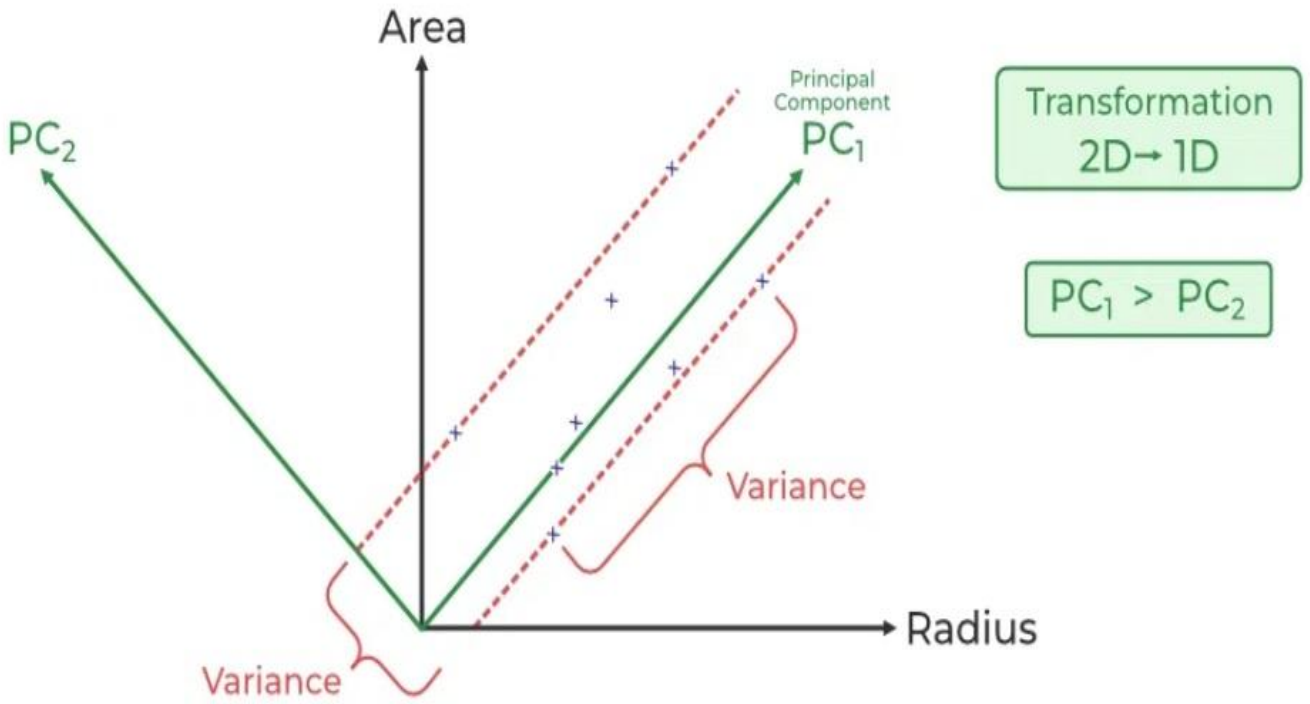
$$AX=\lambda X$$

- This means: When *A* acts on X it only stretches or shrinks X by the scalar λ.
- The direction of X remains unchanged hence eigenvectors define "stable directions" of A.
- Eigenvalues help rank these directions by importance.

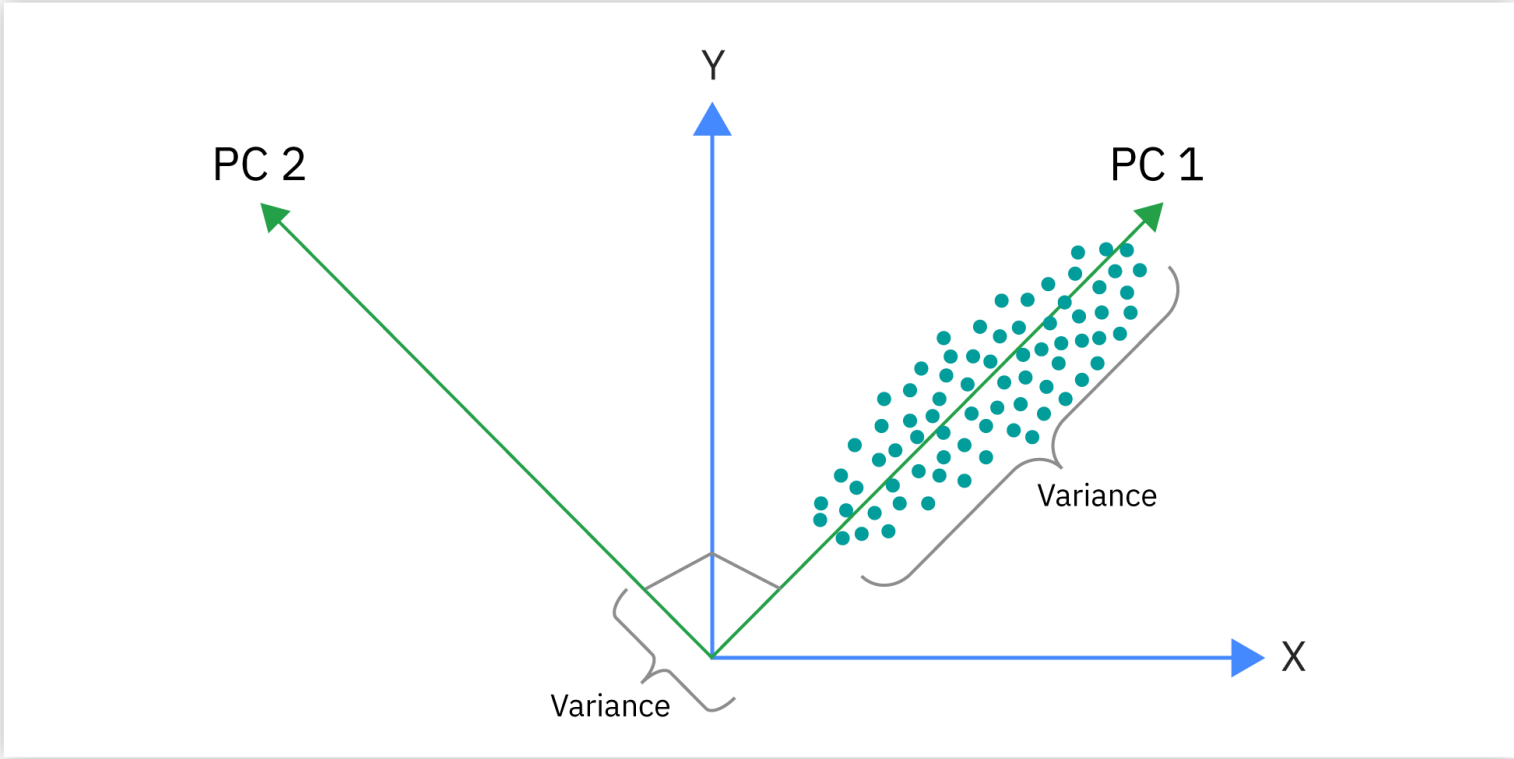
# Pick Top Directions & Transform Data

- After calculating the eigenvalues and eigenvectors PCA ranks them by the amount of information they capture. We then:
  - Select the top k components hat capture most of the variance like 95%.
  - Transform the original dataset by projecting it onto these top components.

- This means we reduce the number of features (dimensions) while keeping the important patterns in the data.




Transform this 2D dataset into a 1D representation while preserving as much variance as possible.


Y
PC 2
PC 1
Variance
X
Variance

- In the image given in previous slide the original dataset has two features "Radius" and "Area" represented by the black axes.
- PCA identifies two new directions: **$PC_1$** and **$PC_2$** which are the **principal components**.

- These new axes are rotated versions of the original ones. **$PC_1$** captures the maximum variance in the data meaning it holds the most information while **$PC_2$** captures the remaining variance and is perpendicular to $PC_1$.

- The spread of data is much wider along $PC_1$ than along $PC_2$. This is why $PC_1$ is chosen for dimensionality reduction. By projecting the data points (blue crosses) onto $PC_1$ we effectively **transform the 2D data into 1D and** retain most of the important structure and patterns.

# Example Illustration

Suppose we have 2D data with two features, $X$ and $Y$:

| Point | X | Y |
|---|---|---|
| A | 2 | 3 |
| B | 3 | 5 |
| C | 4 | 2 |
| D | 5 | 4 |
| E | 6 | 5 |

We can **plot these points** on a 2D graph.

PCA requires **centering the data** (subtract the mean of each feature):

- Mean of $X$: $(2+3+4+5+6)/5 = 4$
- Mean of $Y$: $(3+5+2+4+5)/5 = 3.8$

Subtract means from each value:

| Point | X-Mean | Y-Mean |
|---|---|---|
| A | -2 | -0.8 |
| B | -1 | 1.2 |
| C | 0 | -1.8 |
| D | 1 | 0.2 |
| E | 2 | 1.2 |

This **shifts the data to be centered at the origin (0,0).**

Covariance measures how variables vary together:

$$\mathrm{Cov}(X,X) = \frac{\sum(X_i - \bar{X})^2}{n-1}, \quad \mathrm{Cov}(X,Y) = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$$

For our data:

$$\text{Cov Matrix} = \begin{bmatrix} 2.5 & 1.25 \\ 1.25 & 1.7 \end{bmatrix}$$

## Compute Eigenvalues and Eigenvectors

- Solve characteristic equation:

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

- Eigenvalues found (λ1 > λ2):

$$\lambda_1 = 3.0, \quad \lambda_2 = 1.2$$

- Corresponding eigenvectors (normalized):

$$\text{PC1} = \begin{bmatrix} 0.78 \\ 0.62 \end{bmatrix}, \quad \text{PC2} = \begin{bmatrix} -0.62 \\ 0.78 \end{bmatrix}$$

**PC1** → direction of maximum variance

**PC2** → perpendicular to PC1, next highest variance

## Project data onto PCs

$$\text{PC scores} = \text{Centered data} \cdot \text{Eigenvectors}$$

Example for A:

$$\text{PC1}_A = (-2)(0.78) + (-0.8)(0.62) = -2.06$$

$$\text{PC2}_A = (-2)(-0.62) + (-0.8)(0.78) = 0.62$$

Do the same for all points → get **PC1 and PC2 coordinates**.

**Interpretation**

**PC1 axis** → most important direction in dataset

**PC2 axis** → second most important, orthogonal to PC1

You can reduce **2D → 1D** by keeping only PC1 if needed

# Advantages of PCA

- **Multicollinearity Handling:** Creates new, uncorrelated variables to address issues when original features are highly correlated.
- **Noise Reduction:** Eliminates components with low variance enhance data clarity.
- **Data Compression:** Represents data with fewer components reduce storage needs and speeding up processing.
- **Outlier Detection:** Identifies unusual data points by showing which ones deviate significantly in the reduced space.

# Disadvantages of PCA

- Interpretation Challenges: The new components are combinations of original variables which can be hard to explain.
- Data Scaling Sensitivity: Requires proper scaling of data before application or results may be misleading.
- Information Loss: Reducing dimensions may lose some important information if too few components are kept.
- Assumption of Linearity: Works best when relationships between variables are linear and may struggle with non-linear data.
- Computational Complexity: Can be slow and resource-intensive on very large datasets.

- Risk of Overfitting: Using too many components or working with a small dataset might lead to models that don't generalize well.

# T-distributed Stochastic Neighbor Embedding (t-SNE)

- Non linear dimensionality reduction technique used for visualizing high-dimensional data in a lower-dimensional space mainly in 2D or 3D. Unlike linear methods such as Principal Component Analysis (PCA), t-SNE focus on preserving the local structure and pattern of the data.

  - Dimensionality reduction is a process that simplifies complex dataset by combining similar or correlated features. It helps in improving analysis and computational efficiency.

- t-SNE works by looking at the similarity between data points in the high-dimensional space. The similarity is computed as a conditional probability. It calculates how likely it is that one data point would be near another.

# Advantages of t-SNE

- Great for Visualization: t-SNE is particularly used to convert complex high-dimensional data into 2D or 3D for visualization making patterns and clusters easy to observe.
- Preserve Local Structure: Unlike linear techniques like PCA t-SNE focus on maintaining the local relationships between data points meaning similar data points remain close in the lower-dimensional space.
- Non-Linear Capability: It captures non-linear dependencies in the data which makes it suitable for complex datasets where linear methods fail.
- Cluster Separation: Helps in clearly visualizing clusters and class separability in datasets like MNIST making it easier for interpretation and exploration.

# Disadvantages of t-SNE

- Computationally Intensive: t-SNE is slower and more computationally expensive compared to linear methods especially on large datasets.
- Non-deterministic Output: The output can vary with each run due to its randomness unless a fixed random_state is used.
- Not Scalable for Large Datasets: It struggles with very large datasets (e.g., millions of points) unless optimized or approximated versions are used.
- Not Good for Downstream Tasks: t-SNE is mainly for visualization and is not suitable for dimensionality reduction when feeding data into other ML algorithms.

- No Global Structure Preservation: It may distort global distances and structures in the data focusing more on preserving local neighborhoods.

# Thank You