PCA

PCA

Principal Component Analysis is a statistical technique used for dimensionality reduction, which transforms a large set of correlated variables into a smaller set of uncorelated variables called principal components.

Why to use PCA?

→ Reduces the number of features (dimensions) in a dataset, making analysis and visualization easier.

→ Removes redundancy by transforming correlated variables into uncorrelated principal components.

→ Helps in identifying patterns, compressing data, and denoising.

How did PCA came to this game?

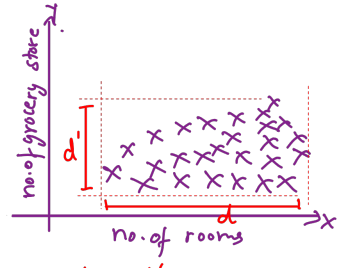
Suppose, you have this data, and you are asked to remove one of the column. Which one to remove?

room in house	grocery store nearby	price
3	4	720
2	2	300
8	1	1.7

→ offcourse, the price of the house would depend more on the total room in the house. And feature selection would remove this by measuring variance.

If you notice the graph, the variance of room in house (x-axis) is greater than the variance of y-axis.

Therefore, Total groceries store nearby column would be removed due to low variance.



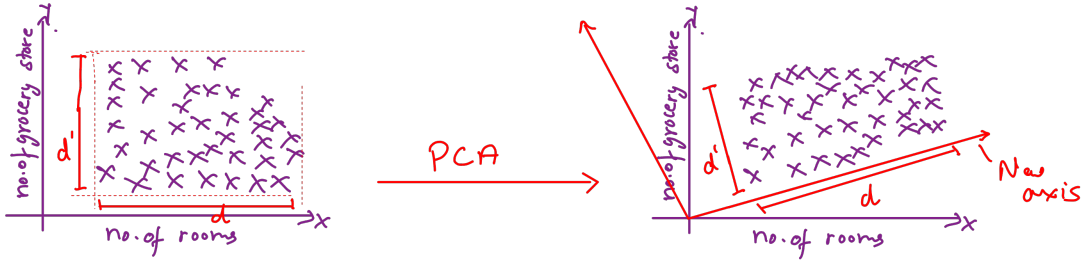
$d > d'$

But what if instead of no. of grocery store col, you are given no. of bathroom?

Since, now both of the cols, 'rooms' and 'no. of bathroom' are strongly related to price, we cannot eliminate both of the colums. Therefore, feature selection is confused on which col to remove, and dosen't do it's work.

Therefore, to solve this problem, PCA comes in the game.

PCA simply rotates the axis in a way that it will be able to compare the new variance and decide which one to eliminate.



Cant really tell which one is greater from "d and d'"

$d > d'$

PCA views the data from a different perspective (axis) and then eliminates/choose based on the variance.

Our Main goal is to Maximize the Variance.

Steps of PCA

1. **Standardize the Data**
- Adjust the scale of each feature so they contribute equally to the analysis. This typically means transforming the data so each feature has a mean of zero and a standard deviation of one. This step prevents features with larger scales from dominating the results.

2. **Compute the Covariance Matrix**
- Calculate the covariance matrix to understand how variables in the dataset relate to each other. The covariance matrix reveals the direction in which the data varies the most and highlights correlations between features.

3. **Calculate Eigenvalues and Eigenvectors**
- Compute the eigenvalues and eigenvectors of the covariance matrix. Eigenvectors determine the directions (principal components) in which the data varies, while eigenvalues indicate the magnitude of variance in those directions.

4. **Select Principal Components**
- Rank the principal components by their corresponding eigenvalues (from highest to lowest). Decide how many principal components to retain based on how much total variance you want to preserve (commonly using a scree plot or a set variance threshold, such as 95%).

5. **Project the Data onto the Principal Components**
- Transform the original dataset by projecting it onto the selected principal components. This step creates a new dataset with reduced dimensions, capturing most of the original variance but with fewer variables.

Limitations of PCA

1. Linearity Assumption

PCA assumes that the relationships between variables are linear. It cannot effectively capture or represent nonlinear relationships in the data. If your data has complex, nonlinear patterns, PCA may not provide meaningful results.

2. Interpretability

The principal components generated by PCA are linear combinations of the original features. This can make them difficult to interpret, especially when many variables are involved. Understanding what each principal component represents in real-world terms can be challenging.

3. Sensitivity to Feature Scaling

PCA is sensitive to the scale of the features. Variables with larger variances or different units can dominate the principal components. It is essential to standardize or normalize the data before applying PCA to ensure that all features contribute equally.

4. Sensitivity to Outliers

PCA can be significantly affected by outliers. Outliers can distort the direction of the principal components, leading to misleading results. Proper outlier detection and handling are important before using PCA.

5. Handling of Missing Data

PCA does not handle missing data inherently. Datasets with missing values must be preprocessed, either by removing incomplete records or imputing missing values, before applying PCA.

6. Assumption of Feature Correlation

PCA is most effective when features are correlated. If the features are uncorrelated, PCA may not achieve significant dimensionality reduction or reveal meaningful structure in the data.

7. Limited Usefulness for Specific Tasks

PCA reduces dimensionality by maximizing variance, not by optimizing for specific tasks such as classification or regression. The principal components may not always be the most relevant features for predictive modeling.

8. Poor Performance with Complex Population Structures

In fields like genetics, PCA can struggle with datasets that have complex population structures or relatedness (such as many distant relatives). In such cases, more advanced models may be required.

9. Assumption of Gaussian Distributions

PCA works best when the data follows a Gaussian (normal) distribution. For data that is highly skewed or non-Gaussian, the principal components may not capture the underlying structure effectively.

Eigen Vectors and Eigen Values

Eigen Vectors: An Eigenvector of a matrix A is non-zero vector v that, when multiplied by A, only gets scaled (not rotated or changed in direction). The direction stays the same.

Eigen Values: An Eigenvalue (λ) is the scaling factor that tells you how much the eigenvector gets streched or shrunk.

Mathematical Definition:

$$A \cdot v = \lambda v$$

- A: Matrix
- v: eigen vector
- λ: Scalar quantity
- v: eigen vector

Properties

1. Sum of Eigenvalues: The sum of all the eigenvalues of a matrix is equal to its trace (the sum of the diagonal elements of the matrix). This holds true regardless of whether the matrix is square or not.
2. Product of Eigenvalues: The product of all the eigenvalues of a matrix is equal to its determinant. This also holds for square matrices.
3. Eigenvectors corresponding to different eigenvalues are orthogonal: If a matrix A is symmetric (i.e., A = A^T), the eigenvectors corresponding to distinct eigenvalues are orthogonal to each other.
4. Eigenvalue of a Identity Matrix: For an identity matrix, the eigenvalues are all 1, regardless of the dimension of the matrix.
5. Eigenvalue of a Scalar Multiple: If B is a matrix obtained by multiplying a scalar c to a matrix A (i.e., B = cA), then the eigenvalues of B are just the eigenvalues of A each multiplied by c.
6. Eigenvalues of a Diagonal Matrix: For a diagonal matrix, the eigenvalues are the diagonal elements themselves.
7. Eigenvalues of a Transposed Matrix: The eigenvalues of a matrix and its transpose are the same.

A brief revision of Matrices

1. Diagonal Matrix
A diagonal matrix is a type of square matrix where the entries outside the main diagonal are all zero; the main diagonal is from the top left to the bottom right of the square matrix.

Powers: The nth power of a diagonal matrix (where n is a non-negative integer) can be obtained by raising each diagonal element to the power of n.

a. Eigenvalues: The eigenvalues of a diagonal matrix are just the values on the diagonal. The corresponding eigenvectors are the standard basis vectors.

b. Multiplication by a Vector: When a diagonal matrix multiplies a vector, it scales each component of the vector by the corresponding element on the diagonal.

c. Matrix Multiplication: The product of two diagonal matrices is just the diagonal matrix with the corresponding elements on the diagonals multiplied.

2. Orthogonal Matrix
An orthogonal matrix is a square matrix whose columns and rows are orthogonal unit vectors (i.e., orthonormal vectors), meaning that they are all of unit length and are at right angles to each other.
Perfect rotation, no scaling or shearing.

Inverse Equals Transpose: The transpose of an orthogonal matrix equals its inverse, i.e., A^T = A^(-1). This property makes calculations with orthogonal matrices computationally efficient.

3. Symmetric Matrix
A symmetric matrix is a type of square matrix that is equal to its own transpose. In other words, if you swap its rows with columns, you get the same matrix.

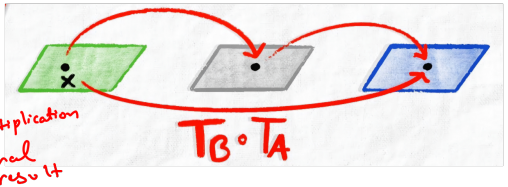
Real Eigenvalues: The eigenvalues of a real symmetric matrix are always real, not complex.

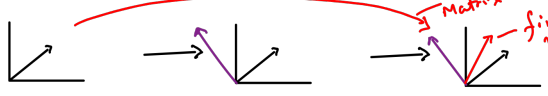
Orthogonal Eigenvectors: For a real symmetric matrix, the eigenvectors corresponding to different eigenvalues are always orthogonal to each other. If the eigenvalues are distinct, you can even choose an orthonormal basis of eigenvectors.

Matrix Composition

Matrix Composition refers to the process of combining two or more matrix transformations into a single transformation by performing matrix multiplication.

Meaning, if there are two matrix A and B, we first apply B transformation on the vector and then A to the updated vector.





Matrix Decomposition

Matrix Decomposition, also known as matrix factorization, is the process of breaking down a matrix into a product of simpler or canonical matrices. or in simple terms, breaking down a complex matrix into simpler matrices.

Types of Matrix Decomposition.

LU Decomposition
QR Decomposition
Eigenvalue Decomposition
SVD Decomposition
etc...

Eigen Decomposition

The eigen decomposition of a matrix A is given by the equation:

$$A = V \Lambda V^{-1}$$

where

- V is a matrix whose columns are the eigenvectors of A
- Λ (lambda) is a diagonal matrix, whose entries are eigenvalues of A
- V^{-1} is the inverse of V
- A is a square matrix, "A must be diagonizable".

Assuming:

Square matrix: Eigen decomposition is only defined for square matrices

Diagonalizibility: For a n*n matrix it should have n linearly independent eigen vectors.

Eigen Decomposition of Symmetric Matrix.

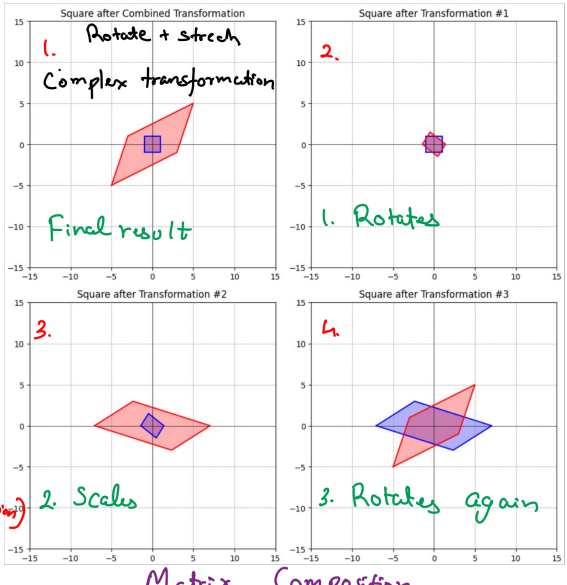
The formula stays the same

$$A = V \Lambda V^{-1}$$ — Spectral decomposition

But since A is a symmetric matrix, we call it as spectral decomposition

$A = V \Lambda V^{-1}$

- V: Orthogonal Matrix
- Λ: Diagonal Matrix
- V^{-1}: Orthogonal matrix



Matrix Composition.

Graph 2, 3 and 4 are transformation results when those matrices are applied on a vector.

Ex: there are three matrices A, B, C

Step①:

A transformation is applied on vector and graph 2 is the result

Step②:

Another matrix B is applied (transformation) on graph ②'s vector which results in graph 3

Step③

Another matrix C is applied and results in graph ④.

And the final result is graph ① (Matrix Composition, final result).

This is a complex transformation (combination of different transformation) (Rotation + Scaling)

Therefore, in the formula:

$A = V \Lambda V^{-1}$

- V: Rotates
- Λ: Scaling
- V^{-1}: de-rotates

Advantages of Eigen Decomposition

Simplifies Computation: Makes matrix operations like computing powers, exponentials, and inverses much easier, especially when matrix is diagonizable.

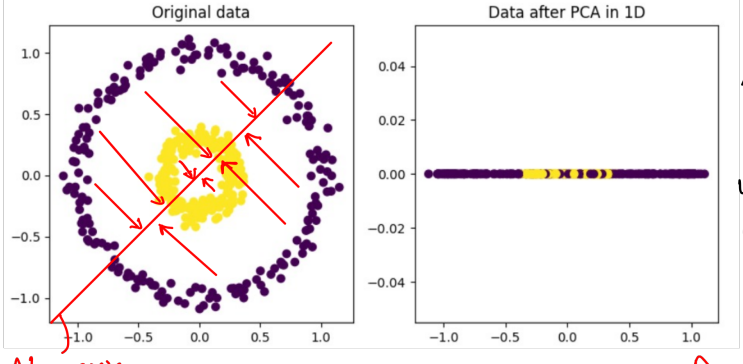
Dimensionality Reduction: Helps in PCA.

Kernel SVM (PCA + SVM)

Kernel SVM is an extension of standard PCA designed to handle non-linear relationship in a data.

Why use it?

- Standard PCA can only capture linear relationships. Kernel PCA can uncover complex non-linear structures and patterns in data.
- It is especially useful for datasets where classes or patterns are not linearly seperable in original feature space.



New axis
(Data will be dropped on this to convert to 1D)

This is after applying PCA. We can notice that since the data is non linear, when the points are dropped to new resultant axis, the data is mixed, and we cannot differentiate between classes

Now let's apply Kernel PCA

Kernel PCA

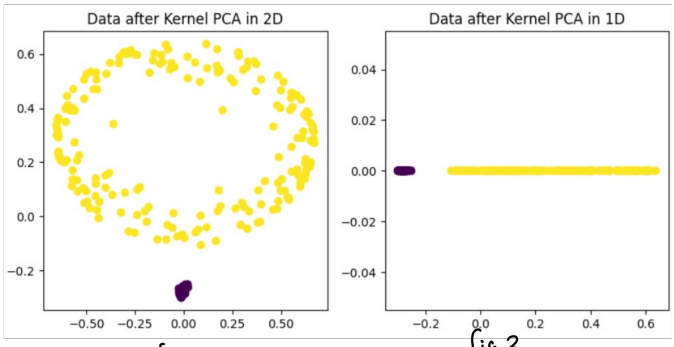


fig 1.

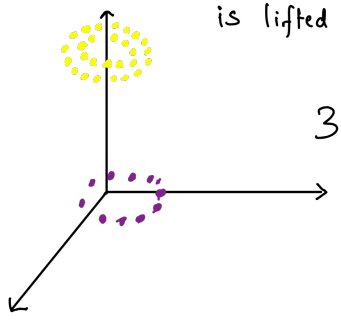
fig 2

To achieve this, A Mathematical function called as Kernel is applied on the 2D Data. This kernel function converts the 2D data to 3D. Now, once we have a 3D data, the yellow data is lifted to 2-axis.

When we apply Kernel PCA on the same data, it will give us different output everytime. Because Kernel PCA does not have a concept of finding the best fit.
We can solve this by applying a classification algo with this, and check accuracy to find best.

This is for scikit learn function.

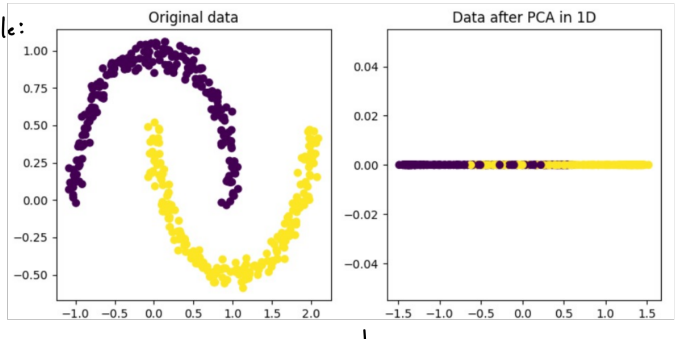
dont need this if we use scipy

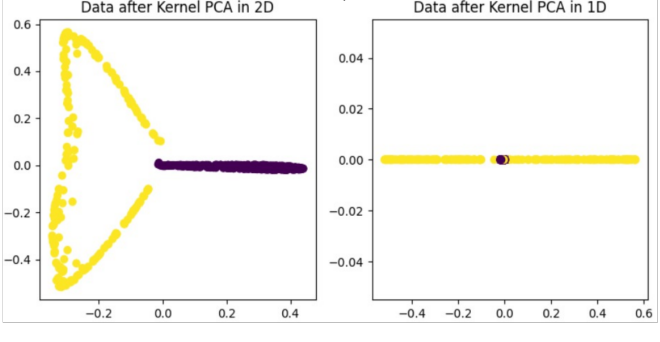


3D data

And again PCA is applied to find the best axis with most variance and we get fig 2.

Another Example:





(Review the code on github to see the 3D result)