APPLICATIONS OF LINEAR ALGEBRA IN DATA ANALYSIS AND CRYPTOGRAPHY

PREPARED BY	TOPICS COVERED
M YAMMAN	Vector Spaces
M IRFAN	Lattices and Vector Spaces
MAHEEN ALI	Classical & Post-Quantum Cryptography
AMNA ASIF	Principal Component Analysis (PCA)
MARIA QASIM	PCA Implementation

ABOUT THE PROJECT

This report explores the fundamental concepts and applications of Linear Algebra. It covers vector spaces, lattices, and their role in modern cryptography, including post-quantum cryptographic techniques. The report also discusses Principal Component Analysis (PCA), presenting its theory, implementation, and practical applications in data analysis. Overall, the study highlights the importance of linear algebra in data science, machine learning, and secure communication systems.

INTRODUCTION

This project aims to study the fundamental concepts of linear algebra and demonstrate their practical relevance through applications in data analysis and cryptography. By exploring both theoretical foundations and real-world implementations, the project highlights how linear algebra serves as a powerful tool in solving complex modern problems.

VECTOR SPACES

Vector Space:

A **vector space** V is a collection of objects (called vectors) where you can:

1. **Add two vectors** together → get another vector in V
2. **Multiply a vector by a number** (scalar) → get another vector in V

These operations must follow certain rules (like addition being commutative: v + w = w + v).

Simple Example: 2D Plane ($\mathbb{R}^2$)

Think of vectors as arrows on graph paper.

Vector v = [3, 2] means: go 3 steps right, 2 steps up **Vector w** = [1, 4] means: go 1 step right, 4 steps up

Addition:

- v + w = [3, 2] + [1, 4] = [4, 6]
- Still a vector on the plane

Scalar Multiplication:

- 2v = 2 × [3, 2] = [6, 4]

- Still a vector on the plane!

Since both operations keep us in the same space, $\mathbb{R}^2$ is a vector space.

Basis:

A **basis** is a minimal set of vectors that can create any other vector in the space through addition and scaling.

Example in $\mathbb{R}^2$:

Standard basis: $\mathbf{e_1} = [1, 0]$ and $\mathbf{e_2} = [0, 1]$

Any 2D vector can be made from these:

- $[5, 3] = 5\times[1, 0] + 3\times[0, 1] = 5e_1 + 3e_2$
- $[7, -2] = 7\times[1, 0] + (-2)\times[0, 1] = 7e_1 - 2e_2$

Another valid basis: $\mathbf{v_1} = [1, 1]$ and $\mathbf{v_2} = [1, -1]$

Same vectors, different basis:

- $[5, 3] = 4\times[1, 1] + 1\times[1, -1] = 4v_1 + v_2$
- Check: $4[1,1] + 1[1,-1] = [4,4] + [1,-1] = [5, 3]$

Dimension:

Dimension = number of vectors in a basis

Examples:

- $\mathbb{R}^2$ (2D plane): dimension = 2 (needs 2 basis vectors)
- $\mathbb{R}^3$ (3D space): dimension = 3 (needs 3 basis vectors)
- A line through origin: dimension = 1 (needs 1 basis vector)

Subspaces:

A **subspace** is a smaller vector space inside a larger one.

Example: Line through origin in $\mathbb{R}^2$

Consider all vectors of the form: $t\times[2, 1]$ where t is any number

This is a subspace because:

- Contains zero vector: [0, 0] (when $t = 0$)
- Closed under addition: $t_1[2,1] + t_2[2,1] = (t_1+t_2)[2,1]$
- Closed under scaling: $c\times(t[2,1]) = (ct)[2,1]$

Visual: All points on the line $y = x/2$

This 1D subspace lives inside the 2D space $\mathbb{R}^2$.

Importance of Vector Spaces:

Provides Structure for Data

Enables Dimensionality Reduction

Allows Change of Basis

Supports Linear Transformations

Foundation for Modern Cryptography

Core of Machine Learning

Lattices and Vector Spaces

What is a Lattice?

A lattice is a special subset of a vector space formed by taking integer linear combinations of a set of linearly independent vectors.

If b1, b2, …, bk are vectors in R^n, then the lattice generated by them is:

L = { z1 b1 + z2 b2 + … + zk bk | zi ∈ Z }

3D Lattice Model:

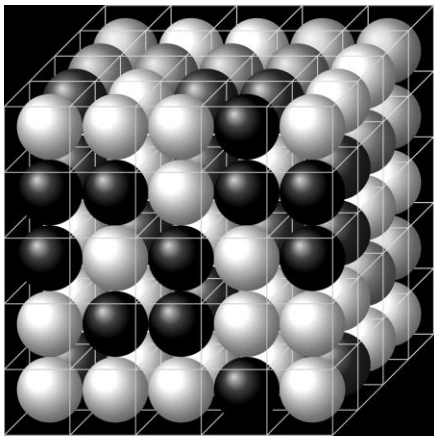


Image source address

Difference between Vector Space and Lattice:

- Vector space allows real number scalars
- Lattice allows only integer scalars
- Vector spaces are continuous
- Lattices are discrete

Lattices as Integer Combinations of Vectors

Example:

Let b1 = (1, 0) and b2 = (0, 1).

All integer combinations form a grid of points on the plane.

L = {(z1, z2) | z1, z2 ∈ Z}

This shows that a lattice exists inside a vector space but is not itself a vector space.

Role of Basis in Lattices

A lattice basis is a set of linearly independent vectors whose integer combinations generate the entire lattice.

Important facts:

- A lattice can have multiple bases.
- Some bases are short and nearly orthogonal.
- Some bases are long and skewed.

Basis choice affects computational difficulty. Problems like the Shortest Vector Problem (SVP) become very hard with a bad basis.

Lattices in High-Dimensional Vector Spaces

In cryptography, lattices are defined in very high dimensions (hundreds or thousands).

Higher dimension means:

- More complexity
- Harder computations
- Stronger security

Connection Between Lattices and Vector Spaces

Lattices are subsets of vector spaces.

Vector space concepts used in lattices:

- Linear independence
- Basis and dimension
- Distance and norms
- Orthogonality

Thus, lattice theory is deeply rooted in linear algebra.

Importance in Post-Quantum Cryptography

Lattice-based cryptography relies on hard mathematical problems in high-dimensional vector spaces.

Reasons for use:

- Resistant to quantum attacks
- Based on well-studied linear algebra concepts
- Efficient and secure

Examples:

NTRU

NTRU (lattice-based post-quantum cryptographic algorithm) is one of the earliest lattice-based cryptographic schemes.

It uses polynomial lattices and relies on the difficulty of solving lattice problems.

NTRU is known for being fast and efficient, even on low-power devices.

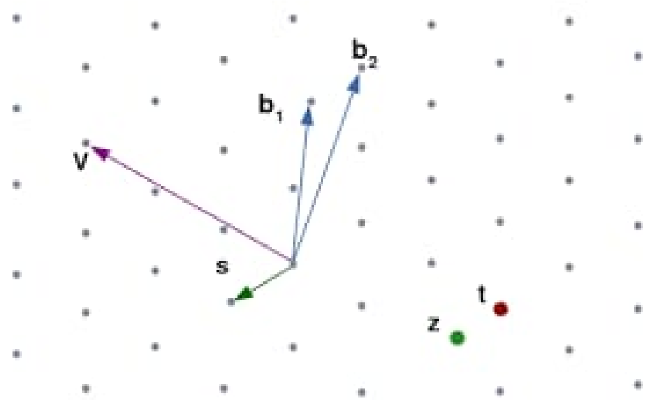


Image source address

Research source address

CRYSTALS-Kyber

CRYSTALS-Kyber is a lattice-based encryption algorithm selected by NIST(National Institute of Standards and Technology) for post-quantum security.

It is based on hard problems in high-dimensional lattices.

Kyber is mainly used for secure key exchange resistant to quantum attacks.

Image source address

Research source address

CRYSTALS Dilithium

CRYSTALS-Dilithium is a lattice-based digital signature scheme.

It uses vector spaces and lattice hardness for security against quantum computers.

Dilithium is efficient and widely recommended for post-quantum digital signatures.

Research source address

Classical & Post Quantum Cryptography

Classical Cryptography:

Defination:

Classical cryptography uses mathematical problems that are hard for normal computers, like factoring large numbers (RSA) or solving discrete logarithms (ECC). Quantum computers can solve these problems much faster using Shor's algorithm, making RSA and ECC insecure. Even symmetric encryption like AES is weakened by Grover's algorithm, which reduces brute-force complexity.

Key Idea:

RSA → Integer Factorization

ECC → Discrete Logarithm

Quantum computers break these because they use superposition and entanglement to compute in parallel.

Post-Quantum Cryptography:

Defination:

Post-Quantum Cryptography (PQC) is a set of cryptographic algorithms designed to resist attacks from quantum computers. These algorithms are based on problems that remain hard even for quantum computers, such as lattice problems, hash-based schemes, and code-based systems.

Key Families:

Lattice-based (Kyber, Dilithium, Falcon)

Hash-based (XMSS, SPHINCS+)

Code-based (McEliece)

Lattice-Based Cryptography:

Defination:

Lattice-based cryptography uses mathematical structures called lattices—grids of points in multi-dimensional space. Security relies on hard problems like Shortest Vector Problem (SVP) and Learning With Errors (LWE), which involve solving noisy linear equations.

Connection with Linear Algebra:

A lattice is a set of points in space formed by integer linear combinations of basis vectors.

If you have basis vectors v1 , v2 , . . . , vn , then any lattice point is a1v1 + a2v2 + . . . + anvn , where ai are integers.

This is pure linear algebra:

Basis vectors → span a space.

Integer coefficients → restrict to discrete points (grid-like structure).

In cryptography, we use high-dimensional lattices (hundreds or thousands of dimensions) because finding short or closest vectors in such lattices is extremely hard.

Learning With Errors (LWE) and Linear Algebra

LWE is based on solving a system of linear equations with noise:

$b = A . s + e \pmod{q}$

Where:

A : a public matrix (size $n \times m$)

s : secret vector (unknown)

e : small error vector (noise)

q : modulus (large prime)

Without error e, this is a simple linear algebra problem:

$b = A . s$

which can be solved easily using Gaussian elimination or matrix inversion.

Adding noise e makes the problem hard because the equations are no longer exact—similar to solving an inconsistent system.

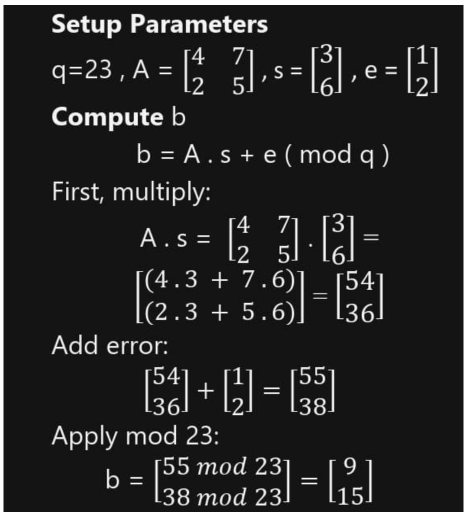
Why This Is Hard

In high dimensions (e.g., $n=512$), even approximating the solution is computationally infeasible.

Quantum computers do not have an efficient algorithm for LWE (unlike RSA/ECC).

Step-by-Step Example of LWE Encryption

take a tiny example (real systems use huge numbers):



Setup Parameters

$q=23$, $A = \begin{bmatrix} 4 & 7 \\ 2 & 5 \end{bmatrix}$, $s = \begin{bmatrix} 3 \\ 6 \end{bmatrix}$, $e = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

Compute b

$$b = A . s + e \; (\bmod q)$$

First, multiply:

$$A . s = \begin{bmatrix} 4 & 7 \\ 2 & 5 \end{bmatrix} . \begin{bmatrix} 3 \\ 6 \end{bmatrix} = \begin{bmatrix} (4.3 + 7.6) \\ (2.3 + 5.6) \end{bmatrix} = \begin{bmatrix} 54 \\ 36 \end{bmatrix}$$

Add error:

$$\begin{bmatrix} 54 \\ 36 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 55 \\ 38 \end{bmatrix}$$

Apply mod 23:

$$b = \begin{bmatrix} 55 \; mod \; 23 \\ 38 \; mod \; 23 \end{bmatrix} = \begin{bmatrix} 9 \\ 15 \end{bmatrix}$$

Public Key: A , b

Secret Key: s

Anyone trying to recover s from A , b faces a noisy linear system—hard in high dimensions.

Why It's Secure

Solving for s is equivalent to solving the Learning With Errors problem, which is as hard as worst-case lattice problems like SVP.

No known quantum algorithm can solve this efficiently.

References

Wikipedia: Lattice-based Cryptography

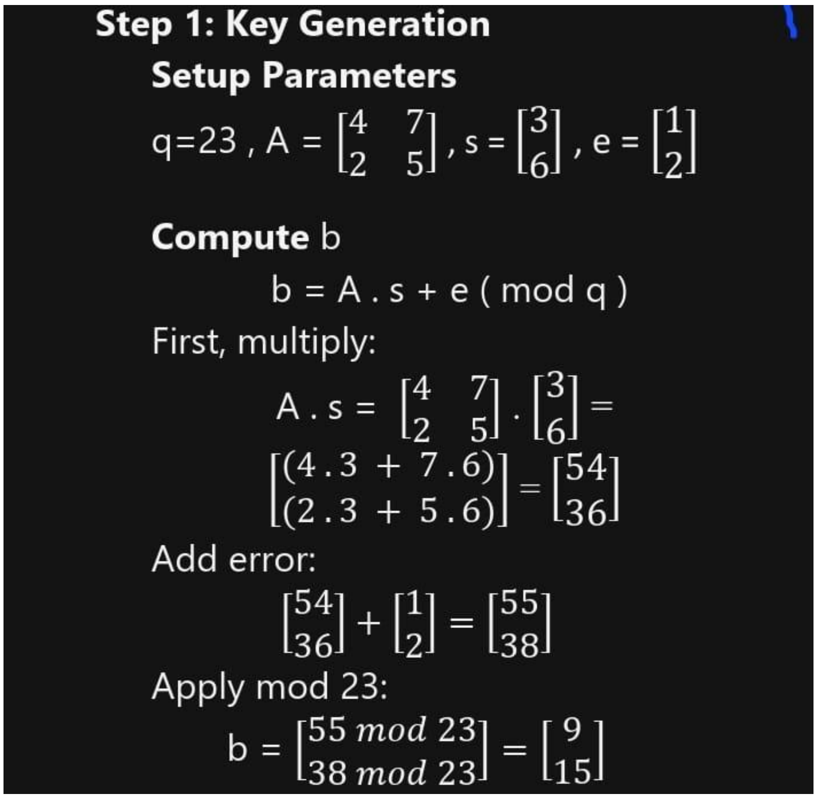
Wikipedia: Learning With Errors

IACR Beginner's Guide

ArXiv Tutorial on Lattice Crypto

Example:

Step 1: Key Generation



Step 1: Key Generation

Setup Parameters

$q=23$, $A = \begin{bmatrix} 4 & 7 \\ 2 & 5 \end{bmatrix}$, $s = \begin{bmatrix} 3 \\ 6 \end{bmatrix}$, $e = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

Compute b

$$b = A . s + e \ (\bmod q)$$

First, multiply:

$$A . s = \begin{bmatrix} 4 & 7 \\ 2 & 5 \end{bmatrix} . \begin{bmatrix} 3 \\ 6 \end{bmatrix} =$$

$$\begin{bmatrix} (4 . 3 + 7 . 6) \\ (2 . 3 + 5 . 6) \end{bmatrix} = \begin{bmatrix} 54 \\ 36 \end{bmatrix}$$

Add error:

$$\begin{bmatrix} 54 \\ 36 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 55 \\ 38 \end{bmatrix}$$

Apply mod 23:

$$b = \begin{bmatrix} 55 \ mod \ 23 \\ 38 \ mod \ 23 \end{bmatrix} = \begin{bmatrix} 9 \\ 15 \end{bmatrix}$$

Public Key: A , b

Secret Key: s

Step 2: Encryption of bit m

Choose random vector x

Compute:

u = AT . x , v = bT . x + q/2 . m

This means:

u is the transpose of A multiplied by x.

v adds the inner product of b and x plus half of q times the message bit m.

If m=1 , then q/2 . m = 11.5 (approx for q=23).

Step 3: Decryption

Compute:

v - sT . u

If the result is close to 0 , then m=0.

If it’s close to q/2 , then m=1.

This works because the noise is small compared to q/2, so the difference reveals the bit.

Referennces:

- Wikipedia: Post-Quantum Cryptography
- Cisco Blog on Quantum Threats
- NIST PQC Project
- Wikipedia: PQC
- Wikipedia: Lattice-based Cryptography
- IACR Beginner’s Guide
- ArXiv Tutorial on Lattice Crypto

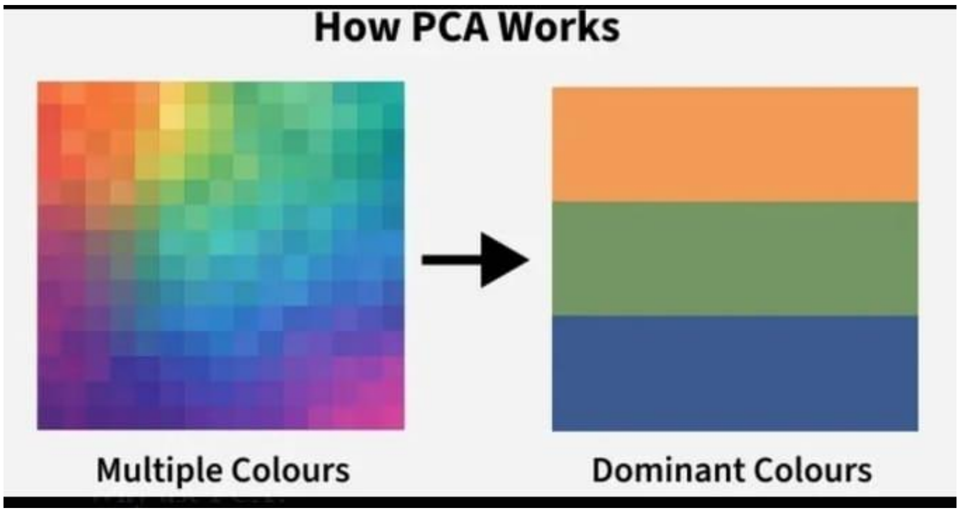
Principal Component Analysis

Introduction: What is PCA?

PCA (Principal Component Analysis) is a dimensionality reduction technique and helps us to reduce the number of features in a dataset while keeping the most important information. It changes complex datasets by transforming correlated features into a smaller set of uncorrelated components.

In simple terms, PCA transforms the original variables into a new set of uncorrelated variables, called principal components (PCs).

It helps us to remove redundancy, improve computational efficiency and make data easier to visualize and analyze.



Why use PCA?

- Reduce computational cost for high-dimensional data.
- Remove noise or redundant features.
- Enable visualization of high-dimensional data in 2D or 3D.

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

How PCA Constructs the Principal Components:

PCA produces the same number of principal components as the original variables.

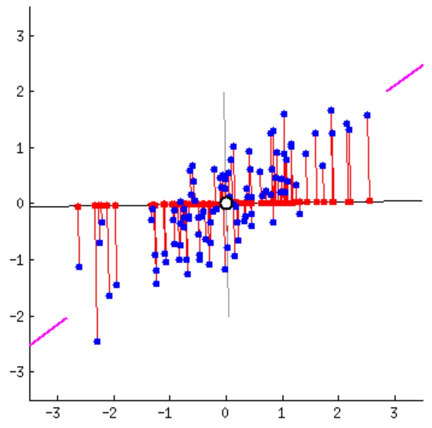
The first principal component (PC1) is the line through the origin along which the projected data points are most spread out in the scatter plot.

Graphically, PC1 appears as the direction that best fits the overall trend of the data cloud.

The second principal component (PC2) is drawn perpendicular to PC1 and captures the next highest variance.

In the scatter plot, projections of points onto PC2 show less spread than PC1.

This process continues until all principal components are obtained.



How Principal Component Analysis (PCA) Works :

Step 1: Standardization (Centering the Data)

Makes all variables comparable in scale.

Prevents variables with large ranges from dominating PCA.

Done by subtracting the mean and dividing by the standard deviation.

Result: all variables have mean = 0 and equal importance.

$$z = \frac{value - mean}{standard\ deviation}$$

Step 2: Covariance Matrix

- Shows how variables vary together.
- Helps detect relationships and redundancy between variables.

- Diagonal entries = variances of variables.
- Off-diagonal entries = covariances:
- Positive → variables increase together
- Negative → one increases, other decreases

For example, for a 3-dimensional data set with 3 variables x, y, and z, the covariance matrix is a 3×3 data matrix of this form :

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

Step 3: Eigenvectors and Eigenvalues

- Eigenvectors → directions of maximum data spread (principal components).
- Eigenvalues → amount of variance in each direction.
- Number of eigenvectors = number of variables.
- Eigenvectors are ranked by largest eigenvalues first.
- First eigenvector = most important principal component (PC1).
- Principal Component Analysis Example:

Let's suppose that our data set is 2-dimensional with 2 variables x,y and that the eigenvectors and eigenvalues of the covariance matrix are as follows:

$$v1 = \begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix} \qquad \lambda_1 = 1.284028$$

$$v2 = \begin{bmatrix} -0.7351785 \\ 0.6778736 \end{bmatrix} \qquad \lambda_2 = 0.0490832$$

- Ranking the eigenvalues in descending order (λ1>λ2) identifies the corresponding eigenvectors as the first and second principal components (PC1 and PC2). The proportion of variance explained by each component is obtained by dividing its eigenvalue by the sum of all eigenvalues; in this example, PC1 explains 96% of the variance, while PC2 explains 4%.

Step 4: Feature Selection (Dimensionality Reduction)

- Select only the most important principal components.
- Form a feature vector using selected eigenvectors.
- Discard components with small eigenvalues (low information).
- Reduces dimensions while keeping maximum information.

Step 5: Data Projection

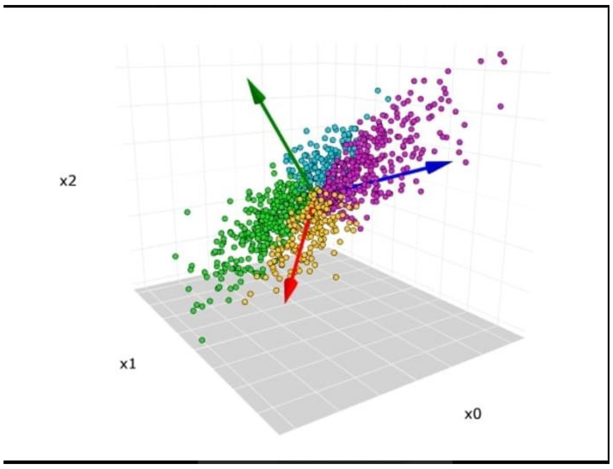
- Original data is projected onto the selected principal components.
- Data is re-expressed in terms of new uncorrelated axes.
- Final result is a lower-dimensional dataset with minimal information loss.

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

Applications of Principal Component Analysis:

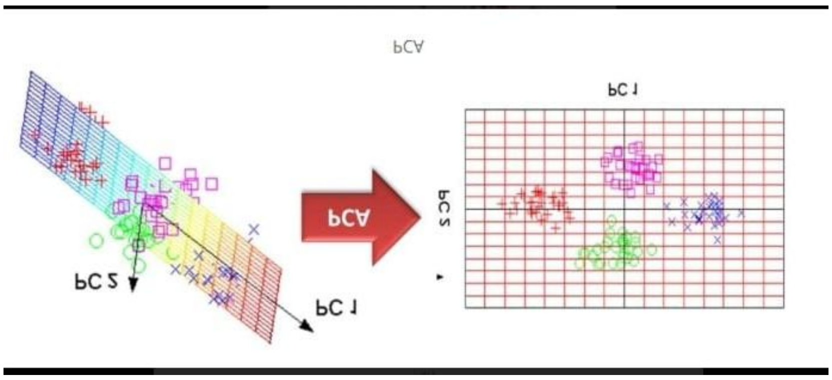
Data Visualization

Reduces high-dimensional data to 2D or 3D for easier visualization of patterns and clusters.



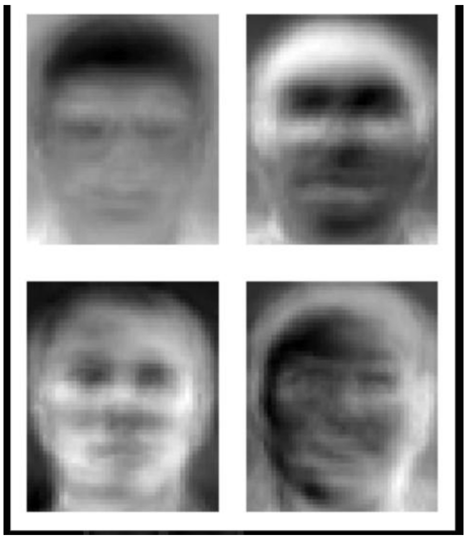
Machine Learning Preprocessing

Lowers dimensionality to speed up training and improve model performance.



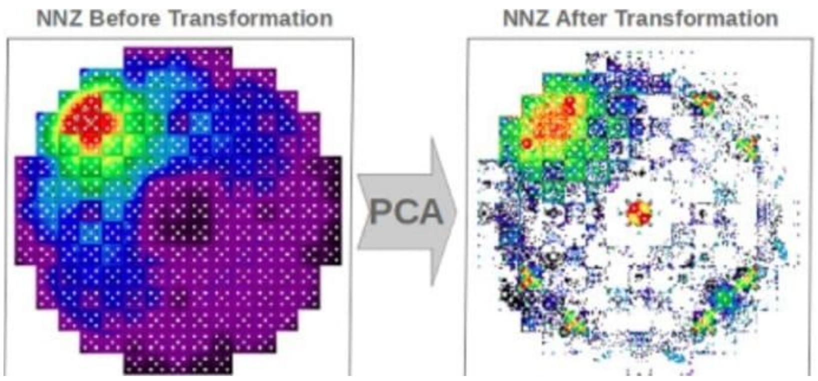
Feature Extraction

Converts correlated features into fewer uncorrelated components.



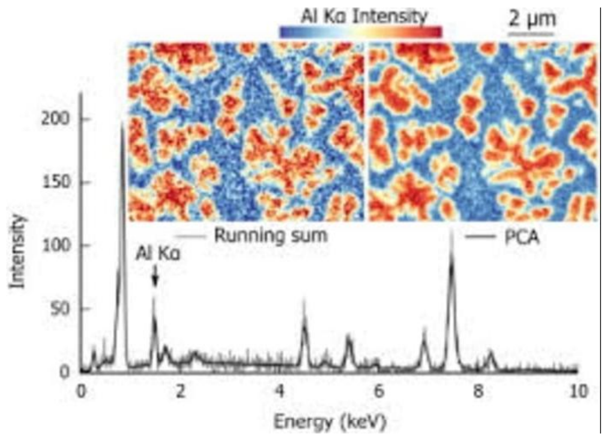
Data Compression

Stores data using fewer dimensions while preserving most information.



Noise Reduction

Removes low-variance components to reduce noise in data.



References :

Jolliffe, I.T., Principal Component Analysis, Springer, 2002.

Strang, G., Linear Algebra and Its Applications, 5th Edition, Cengage, 2016

https://www.geeksforgeeks.org/data-analysis/principal-component-analysis-pca/

https://builtin.com/data-science/step-step-explanation-principal-component-analysis

PCA IMPLEMENTATION THROUGH PYTHON

OVERVIEW

In this section, we apply **Principal Component Analysis (PCA)** to a dataset of **movie ratings** from **1000 users** across **6 genres**: Action, Comedy, Drama, Romance, Sci-Fi, and Horror. The goal is to reduce the **6-dimensional data** (representing ratings for different genres) into **2 dimensions**, making it easier to visualize and interpret users' preferences.

DATASET OVERVIEW

we are working with a **simulated movie ratings dataset** for **1000 users.** Each user provides ratings for **6 different movie genres**:

- **Action**
- **Comedy**
- **Drama**
- **Romance**
- **Sci-Fi**
- **Horror**

The ratings are on a scale from **1 (dislike)** to **5 (love)**, representing how much a user enjoys each genre. The dataset has **1000 rows** (representing users) and **6 columns** (representing movie genres).

Why PCA?

The primary reason for using PCA in this project is to simplify the dataset while maintaining the **important information** about users' preferences. PCA allows us to reduce the number of dimensions (from 6 genres) to just 2 principal components, making it easier to visualize and analyze patterns in user preferences.

CODE:

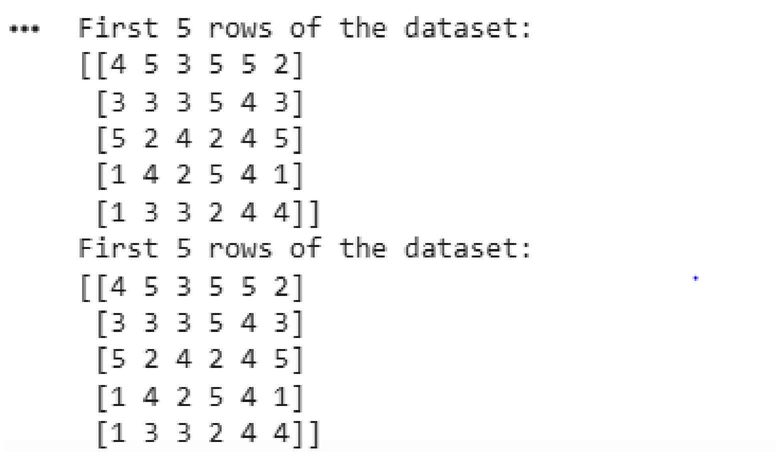
```
import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans


# Simulating ratings from 1000 users for 6 genres (Action, Comedy, Drama,
Romance, Sci-Fi, Horror)

np.random.seed(42)

n_users = 1000

n_genres = 6
```

```
data = np.random.randint(1, 6, size=(n_users, n_genres))

# Displaying the first 5 rows of the dataset
print("First 5 rows of the dataset:")
print(data[:5])

# Standardizing the data to ensure equal contribution from all genres
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

# Applying PCA to reduce the data to 2 dimensions
pca = PCA(n_components=2)
data_pca = pca.fit_transform(data_scaled)

# Displaying the first 5 rows of the PCA-transformed data
print("First 5 rows of the PCA-transformed data:")
print(data_pca[:5])

# Checking how much variance each principal component explains
print("Explained Variance Ratio:", pca.explained_variance_ratio_)

# Plotting the reduced 2D data (users in 2D space)
plt.figure(figsize=(8, 6))
plt.scatter(data_pca[:, 0], data_pca[:, 1], c='blue', edgecolor='k', s=100)
plt.title('PCA of Movie Genre Ratings (6D to 2D)', fontsize=14)
plt.xlabel('Principal Component 1 (PC1)', fontsize=12)
plt.ylabel('Principal Component 2 (PC2)', fontsize=12)
plt.grid(True)

plt.show()


# Clustering users into 3 groups based on their preferences

kmeans = KMeans(n_clusters=3, random_state=42)

clusters = kmeans.fit_predict(data_pca)


# Plotting the clusters

plt.figure(figsize=(8, 6))

plt.scatter(data_pca[:, 0], data_pca[:, 1], c=clusters, cmap='viridis', edgecolor='k',
s=100)

plt.title('Clustering of Users Based on Movie Genre Preferences', fontsize=14)

plt.xlabel('Principal Component 1 (PC1)', fontsize=12)

plt.ylabel('Principal Component 2 (PC2)', fontsize=12)

plt.grid(True)

plt.show()


# Checking which genres contribute most to each principal component

print("Principal Components (loadings):", pca.components_)
```

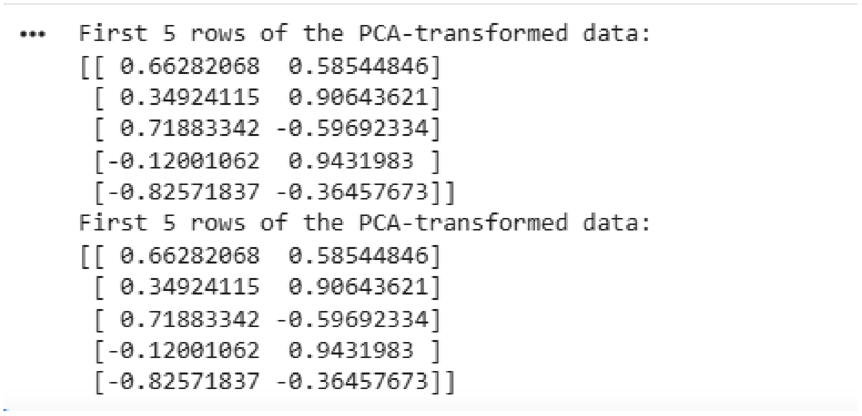
It covers the following steps:

1. Simulating the dataset
2. Standardizing the data
3. Applying PCA
4. Visualizing the results
5. Clustering users based on their preferences

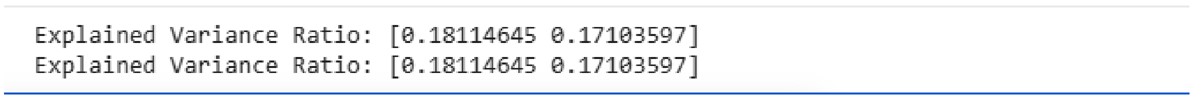
Output



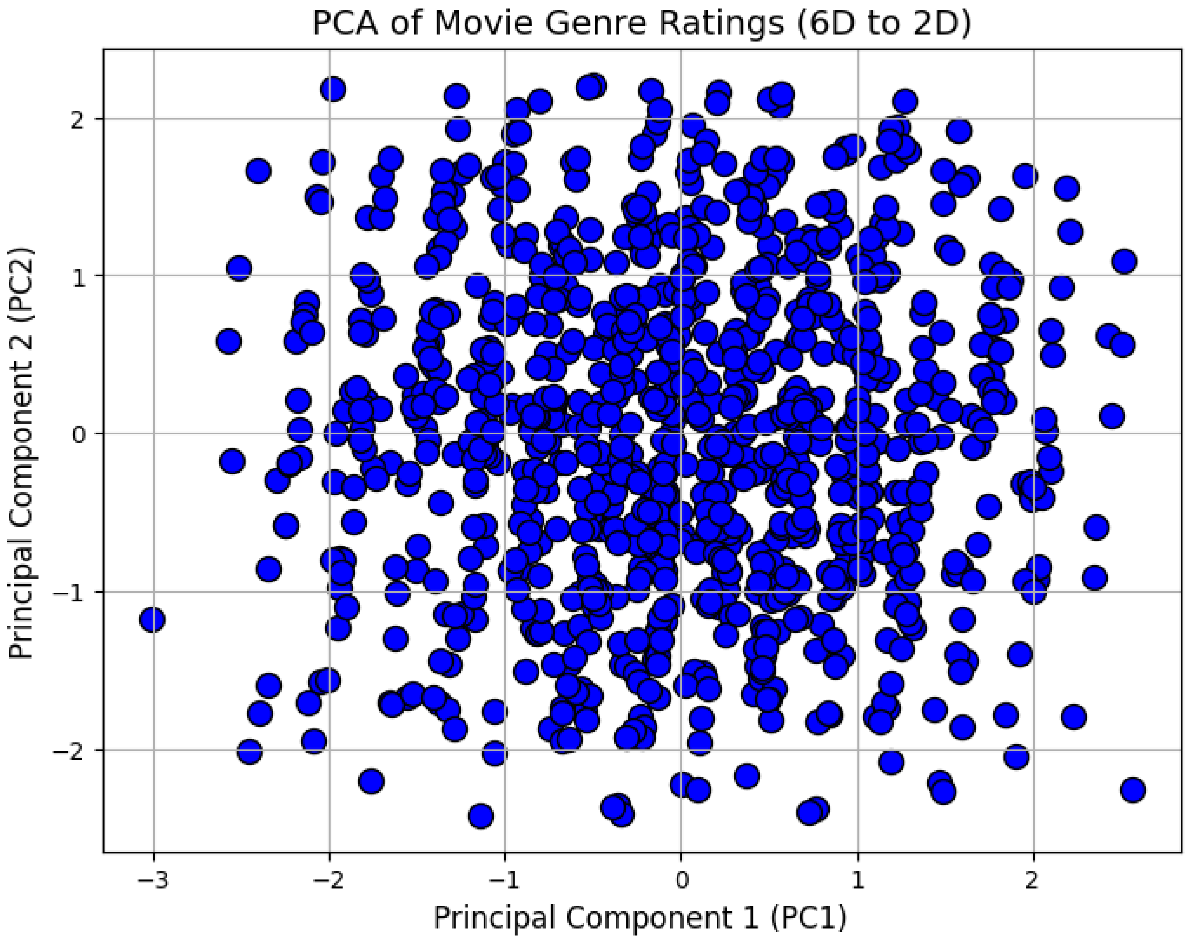
```
•••  First 5 rows of the dataset:
     [[4 5 3 5 5 2]
      [3 3 3 5 4 3]
      [5 2 4 2 4 5]
      [1 4 2 5 4 1]
      [1 3 3 2 4 4]]
     First 5 rows of the dataset:
     [[4 5 3 5 5 2]
      [3 3 3 5 4 3]
      [5 2 4 2 4 5]
      [1 4 2 5 4 1]
      [1 3 3 2 4 4]]
```

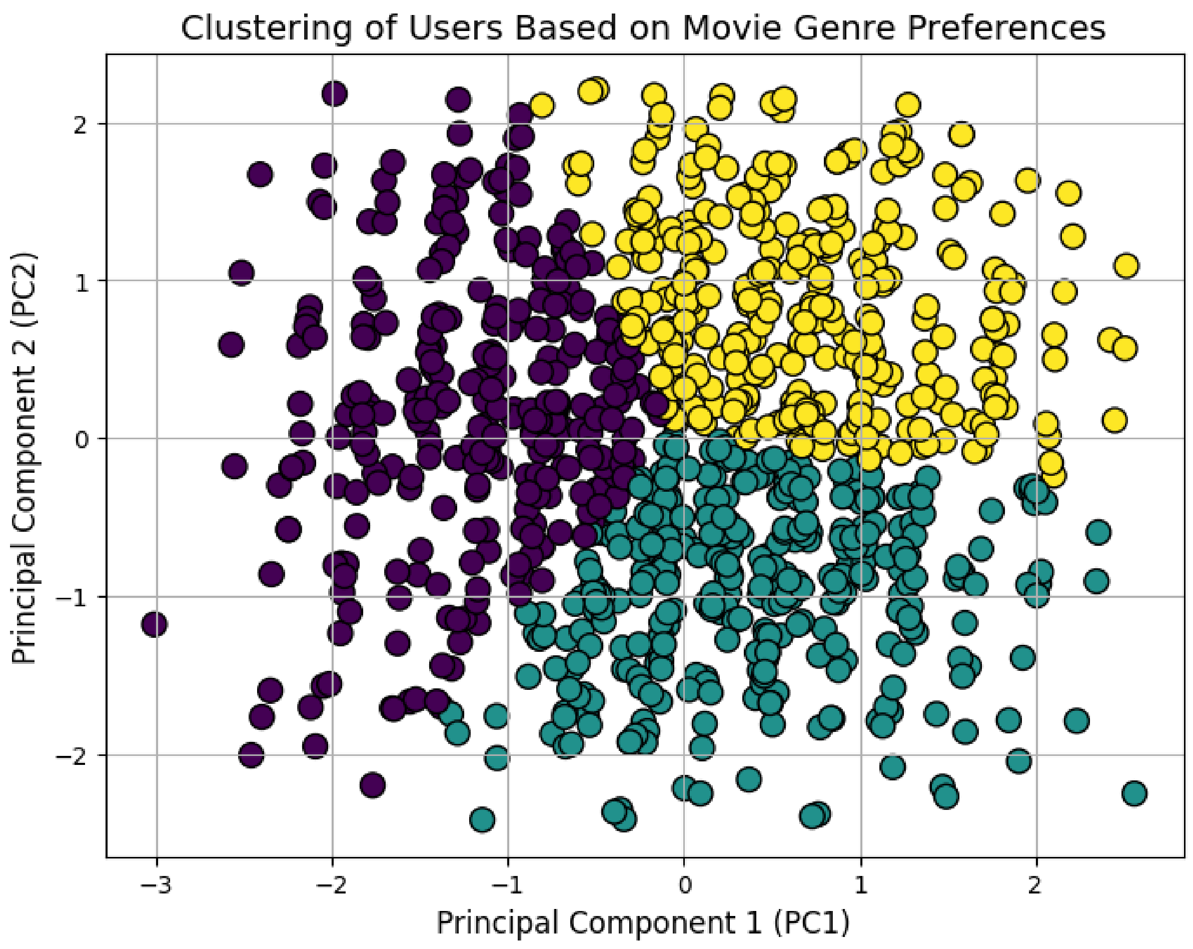


```
•••  First 5 rows of the PCA-transformed data:
     [[ 0.66282068  0.58544846]
      [ 0.34924115  0.90643621]
      [ 0.71883342 -0.59692334]
      [-0.12001062  0.9431983 ]
      [-0.82571837 -0.36457673]]
     First 5 rows of the PCA-transformed data:
     [[ 0.66282068  0.58544846]
      [ 0.34924115  0.90643621]
      [ 0.71883342 -0.59692334]
      [-0.12001062  0.9431983 ]
      [-0.82571837 -0.36457673]]
```



```
Explained Variance Ratio: [0.18114645 0.17103597]
Explained Variance Ratio: [0.18114645 0.17103597]
```


PCA of Movie Genre Ratings (6D to 2D)
Principal Component 1 (PC1)
Principal Component 2 (PC2)
−3
−2
−1
0
1
2
−2
−1
0
1
2



```
Principal Components (loadings): [[ 0.5509195  -0.51470543 -0.08226661  0.04836852  0.48986976 -0.42718419]
 [-0.00678156 -0.07877042 -0.64658631  0.71179518 -0.02602668  0.26142992]]
```

Interpretation

1. Data Overview:

The dataset consists of 1000 users with ratings for 6 movie genres. Each row represents a user, and each column represents the rating for a particular genre. The ratings are on a scale of 1 to 5, where 1 indicates a dislike and 5 indicates a love.

2. Data Standardization:

Before applying PCA, the data was standardized to ensure that all genres contribute equally to the analysis. Without standardization, genres with wider rating ranges could dominate the results, making the PCA ineffective.

3. PCA Application:

We applied PCA to reduce the dataset from 6 dimensions (one for each genre) to 2 principal components (PC1 and PC2). The reduction helps to capture the main patterns in the data while simplifying it. This makes it easier to analyze and visualize the users' preferences.

4. Explained Variance:

The explained variance ratio shows how much of the original data's variability is captured by the principal components:

- PC1 explains 18.1% of the variance in the dataset.
- PC2 explains 17.1% of the variance.
- Together, PC1 and PC2 explain 35.2% of the variance.

While this doesn't capture all the data's complexity, it is often sufficient for visualization and exploratory analysis. The remaining variance might be captured by additional principal components, but these are less important for initial analysis.

5. Data Visualization:

The scatter plot shows how the users are distributed in the 2D PCA space. This is helpful for identifying clusters of users with similar preferences. Users who are closer together in the plot share similar tastes, while users who are farther apart have different preferences.

6. Clustering Users:

Using KMeans clustering, we grouped the users into 3 clusters based on their genre ratings. This helps identify distinct groups of users who prefer similar genres. These groups could be used for movie recommendations or other personalized services.

7. Principal Components Loadings:

The loadings show us which genres contribute most to the principal components:

- PC1 is influenced by Action, Comedy, and Drama.
- PC2 is influenced by Sci-Fi and Romance, with some contribution from Action.
- These loadings give us insights into how the principal components reflect user preferences across different genres.

5. Explanation of PCA Results

1. PCA Dimensionality Reduction:

PCA was used to reduce the dataset from 6 dimensions to 2 principal components. This reduction makes it easier to visualize the data and helps identify patterns in users' preferences.

2. Explained Variance:

The explained variance ratio indicates how much of the original data’s information is retained by the first two principal components. In our case, 35.2% of the variance is explained by PC1 and PC2, which is good enough for initial exploration and visualization.

3. Clustering Users:

The clustering step allows us to group users into 3 distinct preferences. The clusters help identify users who share similar genre tastes and could benefit from personalized movie recommendations.

Conclusion

This project demonstrates the importance of linear algebra as a foundational tool in modern applications. By studying vector spaces, lattices, and Principal Component Analysis (PCA), the report highlights how theoretical concepts are directly applied in data analysis and cryptography. The discussion of post-quantum cryptography further emphasizes the relevance of linear algebra in ensuring secure communication for the future. Overall, the project shows that linear algebra plays a vital role in solving real-world problems in data science, machine learning, and cybersecurity.

References :

- Image source address
- Image source address
- Research source address
- Image source address
- Research source address
- Research source address
- Wikipedia: Lattice-based Cryptography
- Wikipedia: Learning With Errors
- IACR Beginner's Guide
- ArXiv Tutorial on Lattice Crypto
- Wikipedia: Post-Quantum Cryptography
- Cisco Blog on Quantum Threats
- NIST PQC Project
- Wikipedia: PQC
- Wikipedia: Lattice-based Cryptography
- IACR Beginner's Guide
- ArXiv Tutorial on Lattice Crypto
- Jolliffe, I.T., Principal Component Analysis, Springer, 2002.
- Strang, G., Linear Algebra and Its Applications, 5th Edition, Cengage, 2016
- https://www.geeksforgeeks.org/data-analysis/principal-component-analysis-pca/
- https://builtin.com/data-science/step-step-explanation-principal-component-analysis