Story of Usefulness from Uselessness Part I

Minor in AI, IIT Ropar

15th April, 2025

Intelligent Backpacking

Imagine you are a data scientist tasked with reducing the number of features in a large dataset to improve model efficiency. This is analogous to packing for a trip with strict luggage weight limits: you must select the most important items (features) without losing critical information (comfort).

1. Background and Analogy

- **Scenario:** Like a traveler flying to Goa with a 15 kg luggage limit, you must choose only the most essential items from all your belongings. In data science, you must select the most informative features from a large dataset due to constraints (e.g., computational resources, overfitting risk).
- **Key Questions:**
  - How do you detect which items (features) are most important?
  - How much does your experience (model performance) deviate from the ideal (using all features)?

2. Problem Statement

- **Objective:** Identify the most important features in a dataset (matrix) to maximize information retained while minimizing redundancy and loss, akin to maximizing comfort with limited luggage.
- **Challenge:** Too many features make models inefficient and may cause overfitting; too few lose important information, reducing performance.

3. Methodology: Matrix Operations and Convergence

- **Matrix-Vector Multiplication:**
  - Repeatedly multiplying a vector by a matrix and normalizing the result leads to convergence towards a specific direction, regardless of the starting vector.
  - This direction is the **dominant eigenvector** of the matrix.
- **Eigenvectors and Eigenvalues:**
  - **Eigenvector:** A non-zero vector v that satisfies $Mv = \lambda v$, where M is a square matrix and λ is the eigenvalue. It represents a direction invariant under the transformation by M.
  - **Eigenvalue:** A scalar λ that scales the eigenvector during the transformation. It quantifies the importance of the eigenvector.
  - **Finding Eigenvalues:** Solve $\det(M - \lambda I) = 0$.
  - **Finding Eigenvectors:** For each eigenvalue λ, solve $(M - \lambda I)v = 0$.
  - For a square matrix, there are as many eigenvectors as dimensions; the one with the highest eigenvalue is most important.

4. Matrix Transformations & Eigenvectors

Consider matrix A:

$$A = \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix}$$

Vector Transformations

- For $V_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$:

$$AV_1 = \begin{bmatrix} 4 \\ 1 \end{bmatrix} \quad \text{(changes direction)}$$

- For eigenvector $V_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$:

$$AV_2 = 5 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{(same direction, scaled by 5)}$$

- For eigenvector $V_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$:

$$AV_3 = 3 \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad \text{(same direction, scaled by 3)}$$

Centered Data Example

Let Original data:

$$X = \begin{bmatrix} 5 & 2 \\ 3 & 1 \\ 4 & 3 \end{bmatrix}$$

The data points are rows i.e. (5,2) is a data point. For centering, perform X - mean(X), here, mean(X) is mean of all datapoints in X. After centering:

$$X_{\text{centered}} = \begin{bmatrix} 1 & 0 \\ -1 & -1 \\ 0 & 1 \end{bmatrix}$$

Covariance Matrix Calculation

$$C = \frac{1}{2} X_{\text{centered}}^T X_{\text{centered}} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

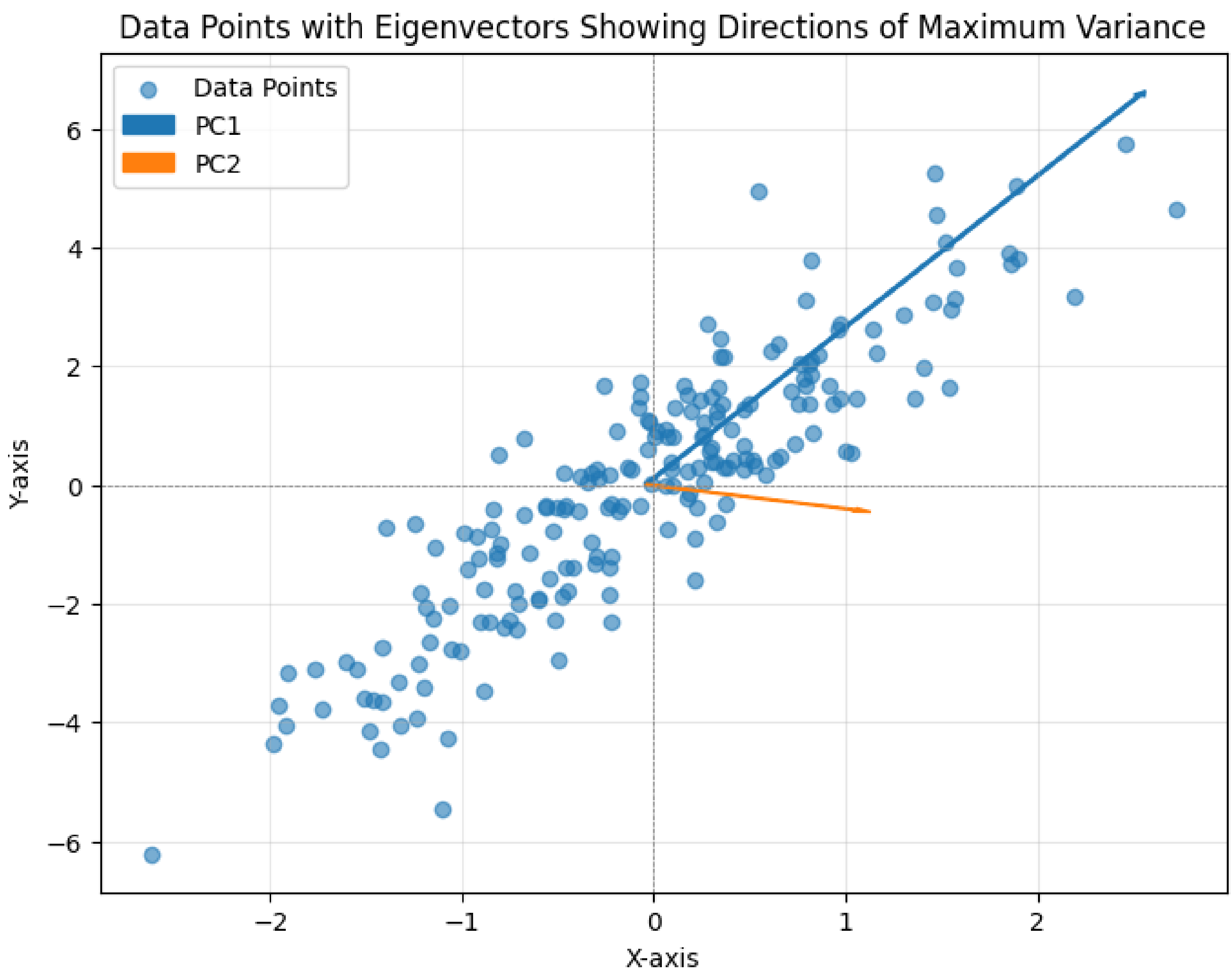
5. Application to Real-World Datasets

- **Feature Extraction via PCA:**
  (a) Compute covariance matrix $D^T D$
  (b) Find eigenvalues/eigenvectors
  (c) Select top K components

6. Visualizing Principal Components

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

np.random.seed(42)
x = np.random.normal(0, 1, 200)
y = 2 * x + np.random.normal(0, 1, 200)

data = np.column_stack((x, y))

# Perform PCA
pca = PCA(n_components=2)
pca.fit(data)
principal_components = pca.components_
mean = np.mean(data, axis=0)

# Scatter plot of data points
plt.figure(figsize=(8, 6))
plt.scatter(data[:, 0], data[:, 1], alpha=0.6, label='Data Points')

# Plot eigenvectors
for i, vector in enumerate(principal_components):
    start_point = mean
    end_point = mean + 3 * np.sqrt(pca.explained_variance_[i]) * vector
    plt.arrow(start_point[0], start_point[1],
```

```
            end_point[0] - start_point[0],
            end_point[1] - start_point[1],
            color=f'C{i}',
            width=0.02,
            label=f'PC{i+1}')

plt.title('Data Points with Eigenvectors Showing Directions of Maximum Variance')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.axhline(0, color='gray', linestyle='--', linewidth=0.5)
plt.axvline(0, color='gray', linestyle='--', linewidth=0.5)
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```



7. Insights and Key Themes

- **Convergence:** Matrix operations naturally converge to dominant eigenvectors

- **Feature Importance:** Eigenvalues quantify directional significance
- Enables efficient models without sacrificing critical information