Hackathon

where quantum computing meets information theory

Quantum Principal Component Analysis

Srijita Dutta

June 17, 2025

ISIT JUNE 22-27 ANN ARBOR MICHIGAN

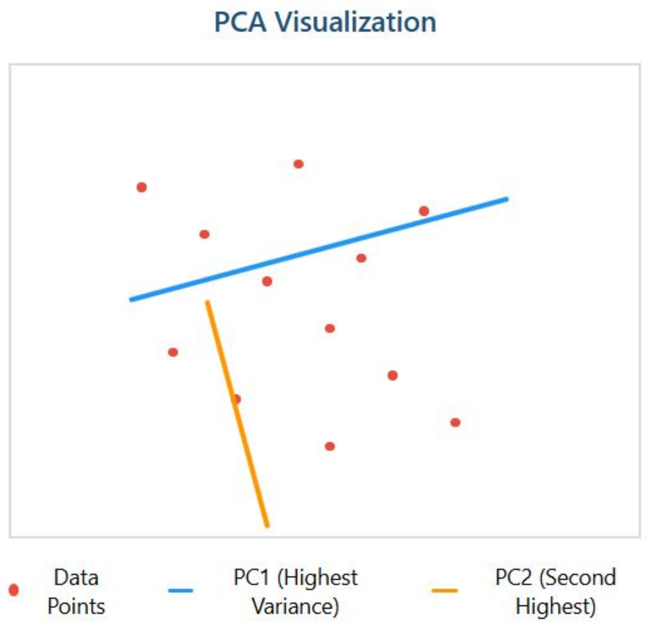
Understanding Classical Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique that identifies the most informative directions in data by finding principal components that capture maximum variance.

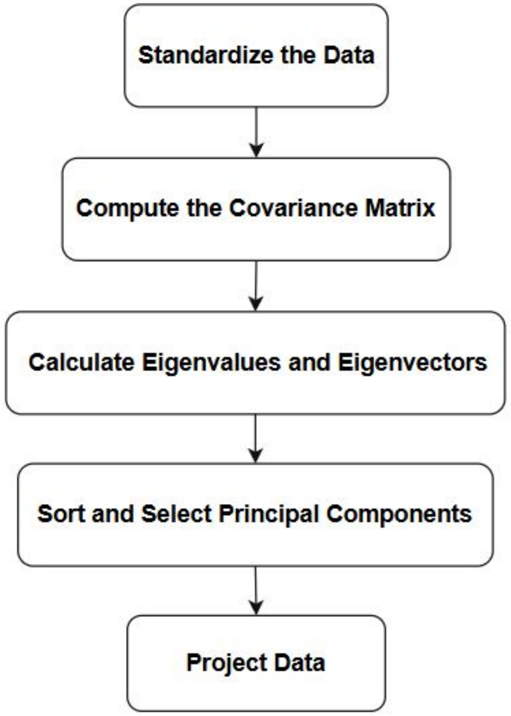
- Transforms data using **linear algebra** operations
- Calculates **eigenvectors (directions) and eigenvalues (importance)**
- Uses **covariance matrix** to identify relationships
- Prioritizes directions with **highest variance**
- Reduces computational complexity while preserving information

Primary Applications

Data preprocessing for machine learning algorithms, visualization of high-dimensional data, noise reduction, and feature extraction.



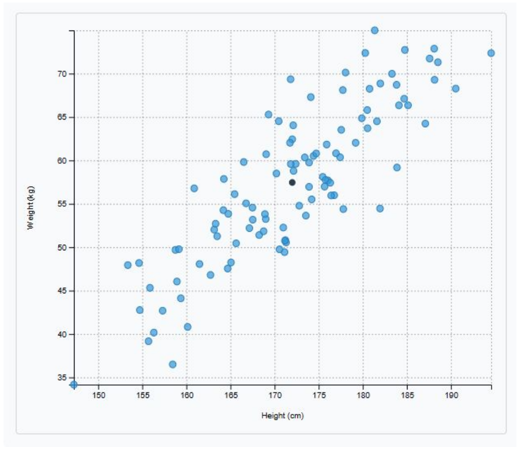
Performing PCA


Standardize the Data
Compute the Covariance Matrix
Calculate Eigenvalues and Eigenvectors
Sort and Select Principal Components
Project Data

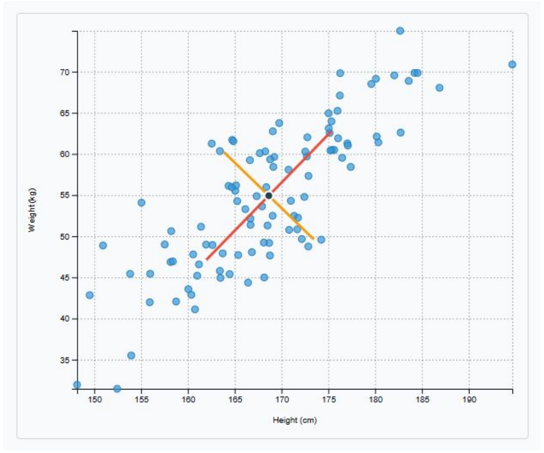
Example:
Let's consider a dataset with two features: **height and weight.**

PCA might reveal that most variance in the data lies along a **diagonal line**, indicating a strong correlation between height and weight.

By projecting the data onto this line, we **reduce dimensionality** while retaining essential information.



Original Data



Data with PCA

How Principal Component Analysis Works

Step 1: 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

$$Z = \frac{X-\mu}{\sigma}$$

where,
μ is the mean of independent features
σ is the standard deviation of independent features

Step 2: Calculate Covariance Matrix
PCA calculates the covariance matrix to see how features relate to each other whether they increase or decrease together. The covariance between two features x1 and x2:

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

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

How Principal Component Analysis Works

Step 3: 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.

How Principal Component Analysis Works

Step 4: Pick the 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 that 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.

Let’s visualize!

https://setosa.io/ev/principal-component-analysis/

For an easy mathematical proof see: Shlens, Jonathon. "A Tutorial on Principal Component Analysis." arXiv preprint arXiv:1404.1100 (2014). https://arxiv.org/abs/1404.1100.

Quantum Principal Component Analysis (QPCA)

Core Concept

QPCA leverages quantum phase estimation to extract eigenvalues and eigenvectors of a density matrix (derived from classical data's covariance matrix) with potential exponential speedup over classical PCA for high-dimensional data analysis.

1. **Construct Density Matrix:**

$$\rho = \frac{X^T X}{tr(X^T X)}$$

2. **Exponentiate ρ:**

$$\rho : density \;\; matrix \rightarrow e^{-i\rho t} : Unitary \;\; matrix$$

3. **Perform Quantum Phase Estimation to obtain eigenvectors *X_i* and eigenvalues *r_i***

$$QPE(e^{-i\rho t}, |\psi\rangle|0\rangle) = \sum_i \psi_i |X_i\rangle|r_i\rangle$$

S. Lloyd, M. Mohseni, and P. Rebentrost, “Quantum principal component analysis,” *Nature Physics*, vol. 10, pp. 631–633, 2014.

Density Matrix Exponentiation: Building Blocks for QPCA

1. The Problem

Need to implement:

$$e^{-i\rho t}$$

For QPE circuit

2. The Challenge

Cannot directly exponentiate ρ without knowing its eigenvectors and eigenvalues

3. The Solution

Lloyd's insight: Use multiple copies and SWAP operator to simulate $e^{-i\rho t}$

$$\rho\otimes = \rho \otimes \rho \otimes \ldots \otimes \rho$$

$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$\boxed{\rho} \qquad \boxed{\rho} \qquad \boxed{\rho}$$

Multiple Copies Required

$$e^{-i\rho t}|\phi\rangle = ?$$

$$\downarrow$$

If $|\phi\rangle$ is eigenvector:

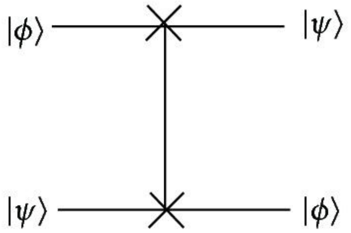
$$e^{-i\rho t}|\phi_j\rangle = e^{-i\lambda_j t}|\phi_j\rangle$$

But we don't know λ_j or $|\phi_j\rangle$!

THE SWAP TECHNIQUE

For small time δ:

$e^{-i\rho\delta} \approx I - i\rho\delta$



BASIC OPERATION

Basic swap:

$tr_p(S\,\rho\otimes\sigma) = \sigma\rho$

Partial swap for small δ:

$e^{-iS\delta}|\psi\rangle|\rho\rangle \approx |\psi\rangle|\rho\rangle - i\delta|\rho\rangle|\psi\rangle$

After tracing out:

$tr_p(e^{-iS\delta}\rho\otimes\sigma e^{iS\delta})$

$\approx e^{-i\rho\delta}\sigma e^{i\rho\delta}$

SCALING UP FOR QPE

For j-th qubit in QPE:

$Need\ U^{\{2^j\}} = e^{-i\rho t\cdot 2^j}$

Requires 2^j copies of ρ

Total for n bits precision:

$2^0 + 2^1 + ... + 2^{n-1} = 2^n - 1$ copies

The key insight: We can simulate $e^{-i\rho t}$ using multiple copies of ρ and controlled-SWAP operations without knowing ρ's eigenvectors!

Classical PCA vs. QPCA

Advantages Over Classical PCA:

- **Exponential speedup:** O(log N) vs $O(N^3)$ for classical PCA
- **Memory efficiency:** Quantum superposition stores exponentially more data
- **Parallel processing:** Quantum parallelism for eigenvalue computation
- **High-dimensional data:** Natural handling of large feature spaces

Real-World Applications:

- Finance: Modeling market correlations and volatility.
- Genomics: Analyzing genetic data for disease markers.
- Drug Discovery: Screening molecular structures for therapeutic potential.

Useful resources:

1. Towards Pricing Financial Derivatives with an IBM Quantum Computer: https://github.com/amartinfer/QPCA/tree/master

2. Towards An End-To-End Approach For Quantum Principal Component Analysis: https://github.com/Eagle-quantum/QuPCA/tree/main