A Multimodal Tutorial for Principal Component Analysis

Cristiana Vlădăreanu

A Multimodal Tutorial for Principal Component Analysis

by

Cristiana Vlădăreanu

submitted in partial fulfillment of the requirements for the degree of
Master of Science
in Applied Physics
at Delft University of Technology,
to be defended on Friday June 19, 2026 at 10:00.

Supervisor: Ilinca Rențea
Delft University of Technology

TU Delft

THIS THESIS IS CONFIDENTIAL AND CANNOT BE MADE PUBLIC.

Publication date:	May 4, 2026
Cover:	The cover image is a template demonstration background.
Copyright:	© 2026 Student Name

Optional custom colophon text below the publication table.

Made with MyST and Typst using the JBOSS Typst template.

Abstract

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique used in Machine Learning and Statistics. It is an unsupervised method that helps reduce the complexity of data by transforming it into a lower-dimensional space, making patterns and relationships within the data easier to understand, while preserving the most relevant information.

Keywords: Principal Component Analysis, PCA, Machine Learning, ML, Education, Multimodal, Modalities, Teaching, Learning, Computer Science, CS

Contents

This page is intentionally left blank.

1. Introduction

Machine Learning (ML) is a field of study that focuses on designing algorithms that learn patterns directly from data. Rather than relying on explicitly programmed rules, machine learning systems use data to automatically identify relationships, detect structure, and make predictions or decisions.

At the core of machine learning lies **data**.

A dataset consists of multiple objects (also called data points or samples). To use an object in a machine learning algorithm, we must represent it numerically. But how do we represent a real-world object so that a mathematical function can process it?

We do this by measuring relevant properties of the object. These measurements are called *features*. All features describing an object are combined into a vector, called a *feature vector*.

Each feature represents one measurable *dimension* of the data. If a dataset contains:

- 1 feature arrow.r the data is one-dimensional
- 2 features arrow.r the data is two-dimensional
- 3 features arrow.r the data is three-dimensional
- n features arrow.r the data is n-dimensional

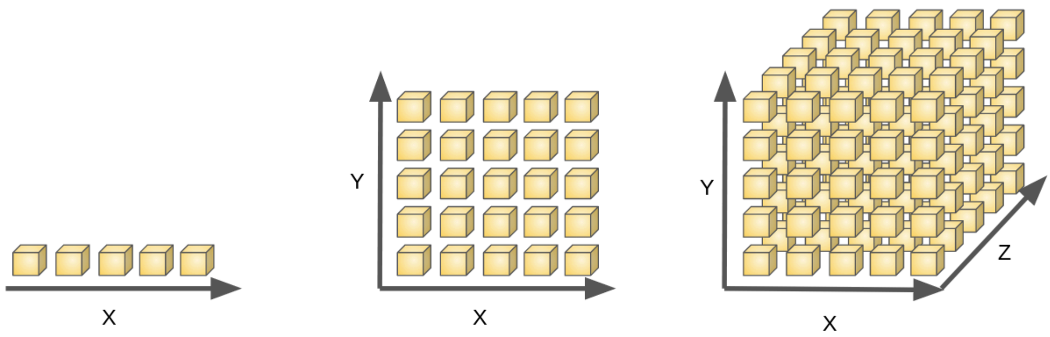

Figure 1.1: One-dimensional, two-dimensional and three-dimensional data. Source: Gleeson (2017)

1.1 The Curse of Dimensionality

As the number of features (dimensions) in a dataset increases, the data becomes harder to work with and understand.

In high-dimensional spaces:

- Data is difficult to visualize;
- The volume of the space increases so fast that the available data becomes sparse;
- Many features may be redundant or uninformative;
- ML models often require exponentially more data to generalize well.

These issues are known as the *curse of dimensionality*.

1.2 Enter: PCA

To address the curse of dimensionality, we need a way to reduce the number of features while keeping the most important information in the data.

Principal Component Analysis (PCA) achieves this by constructing a smaller set of new features that summarize the original data, allowing us to represent the same information in fewer dimensions without significant loss of information.

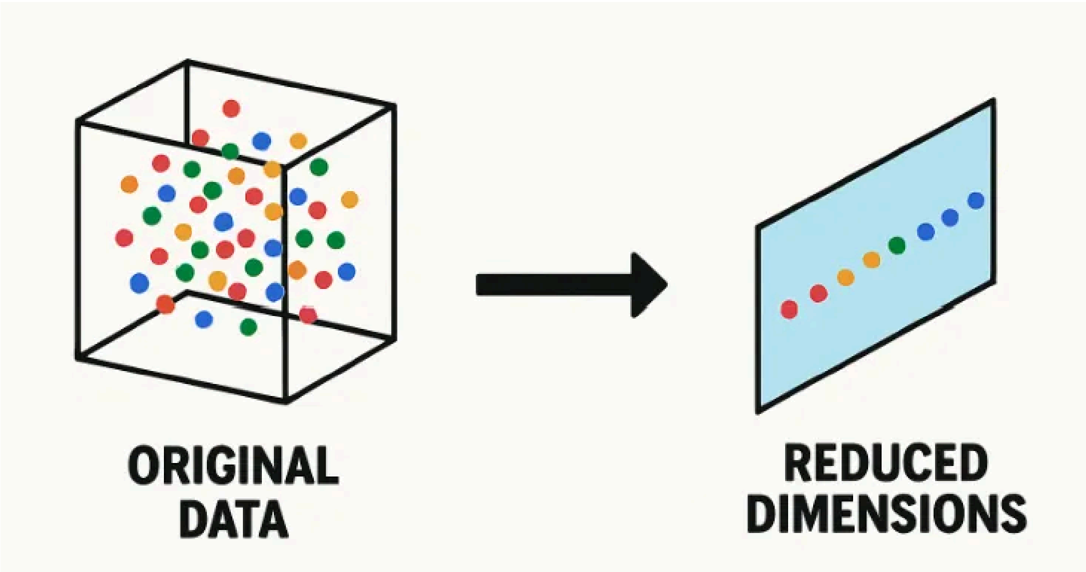

Figure 1.2: Dimensionality reduction with PCA. Source: Vutukuri (2025)

Before we can understand how PCA works, we first need to review some essential mathematical concepts from linear algebra and statistics. These foundations are covered in the next chapter.

2. Mathematical Foundations

To understand how PCA works, we first need to review a few key mathematical concepts. This theory section highlights concepts from linear algebra and statistics that form the building blocks for describing and manipulating data.

2.1 Linear Algebra

Linear algebra is the language for representing and transforming data in Machine Learning. Since datasets are represented as vectors and matrices, understanding how these objects behave is essential for understanding PCA.

2.2 Statistics

Statistics provides the tools to describe and understand data. Since PCA is based on identifying patterns in how data varies, we need basic statistical concepts to measure and interpret this variation.

We begin this mathematical review section by revisiting the key concepts from linear algebra.

2.3 Linear Algebra

2.3.1 Vectors

2.3.1.1 What is a vector?

A vector can be understood in two complementary ways: as an ordered list of numbers or as a geometric object in space.

In ML, a **feature vector** is an ordered list of values, where each entry represents a feature of the object:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{2.1}$$

Each coordinate x_i represents the value of the object for one specific feature.

Geometrically, the same vector can be interpreted as a point (or arrow) in an n-dimensional space, where each axis corresponds to one feature. In this space, the feature vector describes the object's position based on its feature values.

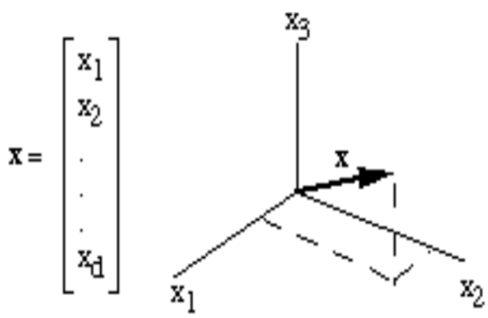

Figure 2.3: Algebraic vs. geometric representation of a vector. Source: Duda (1997)

2.3.1.2 Dot product

The **dot product** is an operation that takes two vectors and returns a single number (a scalar). It measures how much two vectors point in the same direction.

For two vectors:

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \quad and \quad \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{2.2}$$

the dot product is defined as:

$$\boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=1}^{n} a_i b_i \tag{2.3}$$

or explicitly:

$$\boldsymbol{a} \cdot \boldsymbol{b} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \tag{2.4}$$

2.3.1.3 Orthogonality

Two vectors are **orthogonal** if their dot product is equal to zero. This means they are perpendicular to each other in geometric space.

For two vectors:

$$\boldsymbol{a} \cdot \boldsymbol{b} = 0 \quad \Rightarrow \quad \boldsymbol{a} \perp \boldsymbol{b} \tag{2.5}$$

2.3.1.4 Projection

Projection describes how much of one vector lies in the direction of another vector. Geometrically, it can be seen as the "shadow" of a vector when it is projected onto another vector.

There are two related forms of projection: **scalar projection** and **vector projection**. The scalar projection is derived from the vector projection and represents only the *length* of the projection. The vector projection, on the other hand, gives the *full projected vector*, including both direction and magnitude.

The scalar projection of **a** onto **b** is:

$$s = \mid \boldsymbol{a} \mid \cos(\theta) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\mid \boldsymbol{b} \mid} \tag{2.6}$$

The vector projection of **a** onto **b** is:

$$proj_{\boldsymbol{b}}(\boldsymbol{a}) = \mid \boldsymbol{a} \mid \cos(\theta) \frac{\boldsymbol{b}}{\mid \boldsymbol{b} \mid} = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\mid \boldsymbol{b} \mid^2}\boldsymbol{b} \tag{2.7}$$

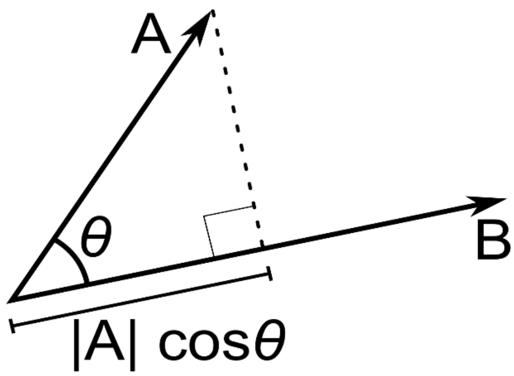

Figure 2.4: Projection of vector **a** onto vector **b**. Source: Wikipedia (2025)

2.3.2 Matrices and Transformations

2.3.2.1 What is a linear transformation?

A **transformation** (or function) T takes an input vector $\boldsymbol{x}$ and assigns it to an output vector $T(\boldsymbol{x})$.

A transformation is **linear** if it satisfies two algebraic rules for all vectors $\boldsymbol{u}, \boldsymbol{v}$ and any scalar c:

1. **Preserves addition:** $T(\boldsymbol{u} + \boldsymbol{v}) = T(\boldsymbol{u}) + T(\boldsymbol{v})$
2. **Preserves scalar multiplication:** $T(c\boldsymbol{u}) = cT(\boldsymbol{u})$

Geometrically, a transformation is linear if it keeps grid lines parallel and evenly spaced, and leaves the origin $(0, 0)$ fixed in place. It essentially keeps the space "flat" without curving it.

Watch the video below to understand the intuition behind linear transformations. Source: {cite} `3blue1brown2016 —

2.3.2.2 What is a matrix transformation?

A matrix transformation is a specific type of linear transformation that can be represented using matrix multiplication. It turns out that *every* linear transformation from an n-dimensional space to an m-dimensional space can be perfectly described by an $m \times n$ matrix A, such that:

$$T(\boldsymbol{x}) = A\boldsymbol{x} \tag{2.8}$$

This means every input vector x is transformed by multiplying it with a matrix A.

Geometrically, to understand what a specific matrix does to a dataset, you only need to ask one question: **Where do the basis vectors go?** Once you know where the basis vectors go, every other vector is determined automatically, since it can be written as a linear combination of the standard basis vectors.

> **What are basis vectors again?**
>
> In a standard 2D Cartesian coordinate system, every vector can be described as a combination of two special unit vectors, called the **basis vectors**:
>
> - $\hat{\imath}$ (i-hat): The vector pointing 1 unit right along the x-axis, $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$.
> - $\hat{\jmath}$ (j-hat): The vector pointing 1 unit up along the y-axis, $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

Watch the video below to see how matrix columns determine where the basis vectors go, and how all other vectors follow from linear combinations of those basis vectors. Source: {cite}`3blue1brown2016 —

2.3.2.3 Common Matrix Transformations

2.4 Statistics

2.4.1 Basic Statistical Theory

2.4.2 The Covariance Matrix

3. PCA Step-by-Step

PCA tutorial

4. Conclusion

In this study we investigated whether and how scientific publishing is feasible through the use of Jupyter Book. Using a starterkit template repository which can be easily accessed and used, a head start is provided. The manual provides detailed information to make more technical elements accessible. Hence, we conclude that Jupyter Book provides a feasible way of publishing scientific content - at least for bachelor and master thesis.

Bibliography

Duda, Richard O. 1997. "Pattern Recognition for HCI." Princeton University Archive,. https://www.cs.princeton.edu/courses/archive/fall08/cos436/Duda/PR_model/f_vecs.htm.

Gleeson, Peter. 2017. "Escaping the Curse of Dimensionality." FreeCodeCamp,. https://www.freecodecamp.org/news/the-curse-of-dimensionality-how-we-can-save-big-data-from-itself-d9fa0f872335/.

Vutukuri, Kiran. 2025. "Principal Component Analysis (PCA) & Dimensionality Reduction." Medium,. https://medium.com/@kiranvutukuri/27-principal-component-analysis-pca-dimensionality-reduction-b7ed1b724a02.

Wikipedia. 2025. "Scalar Projection." Wikipedia, The Free Encyclopedia,. https://en.wikipedia.org/w/index.php?title=Scalar_projection&oldid=1280429209.