Project "Topic: Eigenvalues and Eigenvectors in PCA; Derivatives in Neural Networks (NN)" IB3702 Mathematics for Machine Learning

Tobias Hungwe Harman Singh

15 november, 2025

Contents

1 Eigenvalues and Eigenvectors in Principal Component Analysis (PCA) 3
- 1.1 Introduction . . . 3
- 1.2 Preliminaries . . . 3
  - 1.2.1 Vectors . . . 3
  - 1.2.2 Linear transformations . . . 4
  - 1.2.3 Eigenvectors and Eigenvalues . . . 4
  - 1.2.4 Eigen-decomposition . . . 5
  - 1.2.5 Covariance matrix . . . 5
- 1.3 Methods . . . 6
  - 1.3.1 Step 1. Standardise the dataset . . . 6
  - 1.3.2 Step 2. Calculate the covariance matrix . . . 6
  - 1.3.3 Step 3. Perform eigen-decomposition on the covariance matrix . . . 6
  - 1.3.4 Step 4. Sort eigenvalues and select top k eigenvectors . . . 6
- 1.4 Numerical Examples . . . 7
  - 1.4.1 Methodology . . . 7
  - 1.4.2 Findings . . . 7

2 Derivatives in Neural Networks (NN) 8
- 2.1 Introduction . . . 8
- 2.2 Preliminaries . . . 9
- 2.3 Methods . . . 9
- 2.4 Numerical Examples . . . 9

3 Collaboration 9
- 3.1 Topic choice . . . 9
- 3.2 Code sharing . . . 9
- 3.3 Communication . . . 9

4 Reflection 9
- 4.1 Tobias Hungwe . . . 9
- 4.2 Harman Singh . . . 9

1 Eigenvalues and Eigenvectors in Principal Component Analysis (PCA)

1.1 Introduction

Machine learning (ML) relies heavily on mathematics. Luckily, ML is a rather new branch of computer science, which means that it has access to hundreds of years of advancements in the field of mathematics to base its core around, rather than having to discover new mathematical concepts. This means, usually, that the mathematics used in ML is rather straightforward. You build upon centuries worth of linear algebra and calculus to model, analyse and interpret data.

One technique to interpret data, and the topic of this report, is 'Principal Component Analysis' (PCA). PCA is a linear dimensionality reduction technique used in exploratory data analysis, with one main purpose — to reduce the dimensionality of a dataset. Here: to reduce the number of columns (variables) which in turn makes the underlying dataset easier to process by computers. It does this by transforming the data into a new set of variables, the principal components (PCs), which are uncorrelated and ordered by the amount of variance they capture from the original data. The first principal component captures the most variance, the second captures the second most, and so on. This transformation is achieved through the mathematical concepts of eigenvalues and eigenvectors.

The aim of this report is to explore a new field of mathematics and gain knowledge in it. We will connect the theoretical concepts of eigenvalues and eigenvectors from linear algebra to their practical use case in PCA.

1.2 Preliminaries

1.2.1 Vectors

You can visualise a column in a dataset as a vector in a high-dimensional space. Each row in the dataset corresponds to a component of the vector, so a dataset with n observations can be represented as a vector in an n-dimensional space.

x	y
0.41	0.36
0.24	0.09
0.77	0.66

$$\rightarrow \quad \mathbf{v_x} = \begin{bmatrix} 0.41 \\ 0.24 \\ 0.77 \end{bmatrix}, \quad \mathbf{v_y} = \begin{bmatrix} 0.36 \\ 0.09 \\ 0.66 \end{bmatrix}$$

Here, the columns `x` and `y` from the dataset are represented as vectors $\mathbf{v_x}$ and $\mathbf{v_y}$ in a 3-dimensional space.

1.2.2 Linear transformations

A linear transformation takes a vector as input and produces another vector as output, while maintaining the structure of the vector space. For example, a linear transformation is represented by a matrix A:

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$$

When this matrix transforms the vector $\mathbf{v} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, the result is:

$$A\mathbf{v} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

This transformation scales the first component of the vector by 2 and the second component by 3.

1.2.3 Eigenvectors and Eigenvalues

An eigenvector is a special type of vector where, when a linear transformation is applied to it, only the scale of the vector changes but not its direction. The eigenvalue is the factor by which the eigenvector scaled.

It is easier explained with a visualiation. In the figure below, the vector v retained the same direction after a linear transformation, unlike the vector w. This means that v is an eigenvector and the distance between v and Av is its eigenvalue

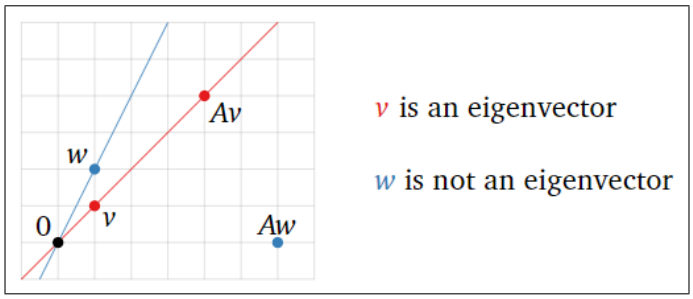

Figure 1: Visualization of eigenvectors and eigenvalues under a linear transformation [1].

The matrix and vectors used in the figure are:

$$A = \begin{bmatrix} 2 & 2 \\ -4 & 8 \end{bmatrix}, \quad v = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad w = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

If we calculate Av and Aw we can confirm that the vector v was simply scaled by a factor of 4, and that w changed direction.

$$Av = \begin{bmatrix} 2 & 2 \\ -4 & 8 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \end{bmatrix} = 4 \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$Aw = \begin{bmatrix} 2 & 2 \\ -4 & 8 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 0 \end{bmatrix}$$

1.2.4 Eigen-decomposition

Eigen-decomposition is a method used to break down a square matrix into its eigenvalues and eigenvectors. For a square matrix A, the eigen-decomposition can be expressed as:

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

where V is a matrix whose columns are the eigenvectors of A, and Λ is a diagonal matrix containing the corresponding eigenvalues.

We take the example matrix A from the previous subsection:

$$A = \begin{bmatrix} 2 & 2 \\ -4 & 8 \end{bmatrix}$$

For $\lambda = 4$

$$(A - 4I)v = 0 \implies \begin{bmatrix} -2 & 2 \\ -4 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0 \implies -2x + 2y = 0 \implies y = x$$

For $\lambda = 6$

$$(A - 6I)v = 0 \implies \begin{bmatrix} -4 & 2 \\ -4 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0 \implies -4x + 2y = 0 \implies y = 2x$$

So the eigenvectors corresponding to the eigenvalues $\lambda_1 = 4$ and $\lambda_2 = 6$ are:

$$v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad v_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

1.2.5 Covariance matrix

Previously we explained that PCA transforms data into a new set of variables, the principal components, which are uncorrelated and ordered by the amount of variance they capture from the original data. To find these principal components, PCA uses the eigen-decomposition of the covariance matrix (a measure of how variables in a dataset vary together).

The covariance matrix S is defined as:

$$S = \frac{1}{n-1}(X - \bar{X})^T(X - \bar{X})$$

where X is the data matrix and $\bar{X}$ is the mean vector.

1.3 Methods

1.3.1 Step 1. Standardise the dataset

Subtract the mean and scale by the standard deviation for each feature to make sure that each feature contributes equally. For this we use the z-score transformation formula, but adapted for each feature j:

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

with μ_j being the mean of feature j and σ_j being the standard deviation of feature j.

1.3.2 Step 2. Calculate the covariance matrix

The covariance matrix S is calculated as:

$$S = \frac{1}{n-1}Z^T Z$$

with Z being the standardised data matrix.

1.3.3 Step 3. Perform eigen-decomposition on the covariance matrix

We perform eigen-decomposition on the covariance matrix S:

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

with V being the matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues.

1.3.4 Step 4. Sort eigenvalues and select top k eigenvectors

$$\boxed{\lambda_1 \geq \lambda_2} \geq \lambda_3 \geq \cdots \geq \lambda_p$$

PCA is not a lossless algorithm, meaning that we do lose some information when we reduce the dimensionality of the data. However, by selecting the top k eigenvectors (principal components), we retain the directions that capture the most variance in the data, and this usually gives a pretty good approximation of the original data.

1.4 Numerical Examples

In the previous section we mentioned that PCA was not a lossless algorithm. With this we meant that when you remove columns from a dataset, you will end up with a smaller dataset, but also lose a part of it which could have been useful. To further explore this, we will perform PCA on a dataset and show the effects on model performance based on how many principal components we retain.

1.4.1 Methodology

We use the 'Wine Dataset' from the UC Irvine Machine Learning Repository [2]. The dataset itself used for the experiment does not matter, since the purpose of this little test is to demonstrate PCA in practice. We standardise the dataset, perform PCA, and then use a helper function to train a logistic regression model using the first k principal components. We evaluate the each of the models using 5-fold cross-validation and record the mean accuracy for each k.

The sklearn library in Python was implemented to standarise the data, perform PCA, train and evaluate the models [3]. Visualisations were made with matplotlib [4] and seaborn [5].

1.4.2 Findings

A visualisation of the findings can be found in the figure captioned "Model performance vs number of retained principal components". We plot the model performance (mean cross-validated accuracy) against the number of retained principal components.

This gives us a pretty nice overview of how the model performance changes as we retain more and more (or less and less) principal components.

The original dataset had 13 features, so when we retain all 13 principal components, we are essentially using the original dataset. As we reduce the number of PCs to just 10, we do not immediately see a significant drop in performance, rather the performance remains relatively stable. However, reducing the number of PCs further does show a significant drop in performance, until we only retain 1 PC which nets us a median accuracy of $0,84$.

This result makes sense if you understand the mathematics behind PCA. By retaining the top 10 PCs, we removed 3 dimensions that likely contained less variance, and thus less important data for the model.

The code used for this experiment is publicly available in the project repository hosted on GitHub at Harmxn02/M4ML-Project [6].

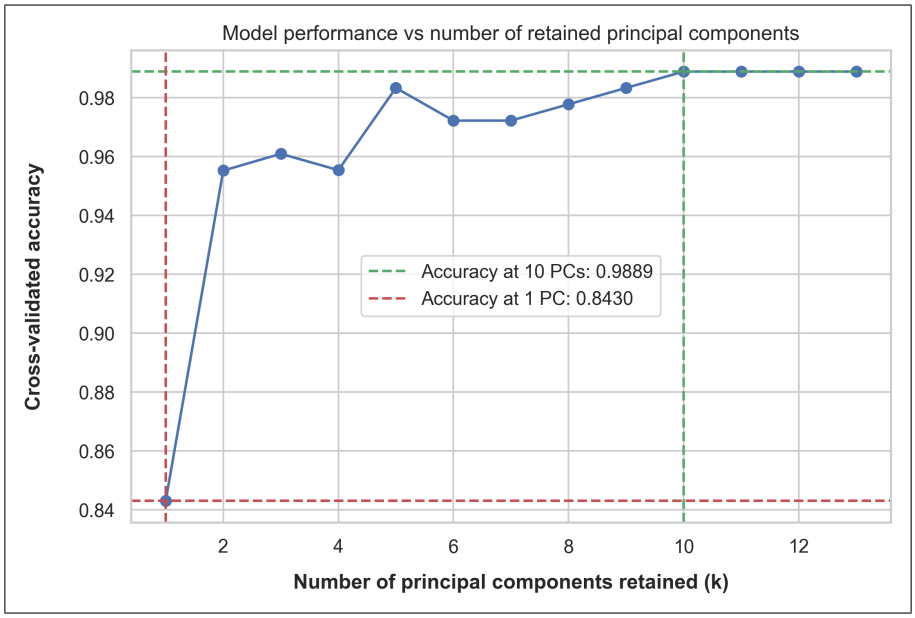

Figure 2: Model performance vs number of retained principal components.

2 Derivatives in Neural Networks (NN)

2.1 Introduction

Introduce the motivation and general context of the topic.

2.2 Preliminaries

Provide the mathematical background required to understand the topic

2.3 Methods

Explain how it works step-by-step

2.4 Numerical Examples

Provide a small illustrative example to demonstrate it in practice.

3 Collaboration

3.1 Topic choice

We went through the document with all possible topics, highlighted the ones we found interesting, and then discussed which 2 we liked the most for each of the two parts.

Initially we also had interest for "2.1.3 Topic 3: Hessian Matrix and Second-Order Optimization in Machine Learning", but we decided that Tobias handle the "Derivative in Neural Networks" topic since it was already covered in our bachelor's degree.

3.2 Code sharing

We used a shared GitHub repository to store all files for the project on. We used 3 LaTeX files, one for each of us, and a third that combines both parts and the collaboration/reflection parts.

3.3 Communication

Throughout the project we mainly communicated through Discord, before, during, and after the course meetings.

4 Reflection

4.1 Tobias Hungwe

...

4.2 Harman Singh

I had explored PCA already briefly before, but was not aware of the mathematics behind it. It was interesting to see that the what we learned in the course was directly connected to PCA. It's a technique that I have used in Python code before to do dimensionality reduction, and now after this project I understand better what the algorithm is actually doing. I "opened the black box", so to say.

References

[1] Libretexts. *8.5.3: Eigenvalues and Eigenvectors - Visualizations*. en. Aug. 2023. URL: https://math.libretexts.org/Courses/SUNY_Schenectady_County_Community_College/A_First_Journey_Through_Linear_Algebra/08%3A_Spectral_Theory/8.05%3A_Supplemental_Notes_-_More_on_Eigenvalues_and__Intro_to_Eigenspaces/8.5.03%3A_Eigenvalues_and_Eigenvectors_-_Visualizations.

[2] Stefan Aeberhard and M. Forina. *Wine*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5PC7J. 1992.

[3] Lars Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[4] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[5] Michael L. Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

[6] Harmxn. *GitHub - Harmxn02/M4ML-Project: Project material for the Mathematics for Machine Learning module*. en. URL: https://github.com/Harmxn02/M4ML-Project.