Duale Hochschule Baden-Württemberg Mannheim

Hausarbeit

A Comprehensive Visualization of the Principal Component Analysis

Studiengang Wirtschaftsinformatik

Studienrichtung Data Science

Verfasser:	André Anan Gilbert, Jan Henrik Bertrand, Felix Noll, Marc Grün
Matrikelnummern:	3465546, 8556462, 9467152, 9603221
Kurs:	WWI 21 DSB
Modul:	Algorithm Visualization
Kursleiter:	Prof. Dr. Maximilian Scherer
Bearbeitungszeitraum:	20.11.2023 – 07.02.2024

Contents

List of Figures ii

Acronyms iii

Abstract iv

1 Introduction 1

2 Visualization 2
- 2.1 Rules of perception . . . 2
- 2.2 Model of human Perceptual Processing . . . 4
- 2.3 Computer Graphics . . . 5
  - 2.3.1 VFX and CGI . . . 6
  - 2.3.2 Manim . . . 7
  - 2.3.3 Challenges during implementation with Manim . . . 8

3 Principal Component Analysis 9
- 3.1 Motivation and Historical Background . . . 9
- 3.2 Mathematical Foundation . . . 10
  - 3.2.1 Covariance Matrix . . . 10
  - 3.2.2 Eigenvalues and Eigenvectors . . . 11
- 3.3 Principal Component Analysis Algorithm . . . 11
  - 3.3.1 Interpretation of the Results . . . 13
  - 3.3.2 Choosing k Principal Components . . . 13
- 3.4 Practical Considerations . . . 14
  - 3.4.1 Standardization . . . 14
  - 3.4.2 Dealing with Missing Data . . . 15
- 3.5 Comparison with other Techniques . . . 15
  - 3.5.1 Autoencoders . . . 15
  - 3.5.2 Singular Value Decomposition (SVD) . . . 17
  - 3.5.3 t-Distributed Stochastic Neighbor Embedding (t-SNE) . . . 19
  - 3.5.4 Locally Linear Embedding (LLE) . . . 20
  - 3.5.5 Kernel PCA . . . 21

4 Summary 22

Bibliography 24

List of Figures

2.1 In the Muller-Lyer illusion on the left, the horizontal line in the upper figure appears longer than the one below. On the right hand side, the rectangle appears distorted into a pincushion shape (Ware, 2019, p.13) 3
2.2 Some regions are easier to distinguish from each other than others (Ware, 2019, p.13) 3
2.3 A three-stage model of visual information processing (Ware, 2019, p.20). . . 4

Acronyms

CGI	Computer Generated Imagery
LLE	Locally Linear Embedding
PCA	Principal Component Analysis
SVD	Singular Value Decomposition
t-SNE	t-Distributed Stochastic Neighbor Embedding
VAE	Variational Autoencoder
UMAP	Uniform Manifold Approximation and Projection

Abstract

This thesis deals with the study of Principal Component Analysis (PCA) and the development and application of effective visualization methods. After diving into the computer graphics, the Python animation package called Manim is presented. Manim is especially strong at visualising complex problems whilst using a mathematical approach for coding.
PCA is a powerful statistical method used in dimension reduction and pattern recognition. Our goal is to explain the theoretical foundations, like Covariance, Eigenvectors and Eigenvalues to provide a deep understanding as to how and why the PCA works. After that a more practical approach is taken by providing solutions for some of the issues of PCA. At the end PCA is being compared with other dimensionality reduction techniques.

1 Introduction

The exploration and comprehension of complex datasets play a pivotal role in various fields, ranging from computer graphics and visualization to data analysis and interpretation. Principal Component Analysis (PCA) stands as a fundamental tool in the arsenal of techniques employed for unraveling the underlying structures within data. This paper embarks on a comprehensive journey into the world of PCA, diving into its mathematical foundations, algorithmic intricacies, and practical considerations. Moreover, it navigates through a comparative analysis with other prominent techniques such as autoencoders, Singular Value Decomposition (SVD), t-t-Distributed Stochastic Neighbor Embedding (t-SNE), and Locally Linear Embedding (LLE).

The exploration begins with a section on visualization, where the rules of perception, models of human perceptual processing, and the integration of computer graphics, particularly Visual Effects (VFX) and the Computer Graphics Interface (CGI) using the Manim Python package, are clarified.

Subsequently, the paper unfolds the historical background and motivation behind PCA, laying the groundwork for a profound understanding. A rigorous examination of the mathematical underpinnings, including the covariance matrix, eigenvalues, and eigenvectors, forms the crux of the discussion. The PCA algorithm is dissected, and its results are interpreted, providing insights into the selection of principal components and practical considerations, such as standardization, and handling missing data.

In an effort to contextualize PCA within the landscape of data analysis, the paper engages in a comparative study with alternative techniques. Autoencoders, SVD, t-SNE, LLE, and a nuanced evaluation of strengths and limitations are explored, offering a panoramic view of PCA's efficacy in various scenarios.

To enhance accessibility, a Python-based visualization using the Manim package has been crafted, presenting a video tutorial that explains the intricacies of PCA. This holistic approach ensures a thorough exploration of PCA, from theoretical foundations to practical applications, establishing its significance in the broader realm of data analysis and visualization.

2 Visualization

Visualizations play a central role in communicating complex information and concepts. Pictures and symbols often help us grasp the ideas behind even some of the most sophisticated methods or algorithms, especially when they are animated. This chapter is dedicated to exploring and presenting the fundamental aspects, literature sources, and applications in the context of visualizations, especially visualizations with the aim of explaining a certain topic.

The visualization of data and concepts is not merely a technique for illustration but also a powerful tool to enhance understanding of intricate relationships and share them with others. In this chapter we will delve into the different techniques of visualization and, more importantly, why and how we perceive these (educational) visualizations the best. We will not only provide an overview of visualization and perception fundamentals, but also conduct a literature review to shed light on current insights and developments in the fields.

2.1 Rules of perception

Thinking is not something that goes on entirely in ones head. Very little academic work is done with the eyes and ears closed, it is rather done through the interaction with different cognitive tools. These tools can range from something as simple as a pen and a paper to sophisticated data analisation tools like MATLAB or Tableau (Ware, 2019, p.2). As the latter is heavily based on humans understanding the interfaces and visualizations in order to being able to analyze the underlying data, we will use the insights that Ware (2019) shares, to apply them onto an educational animated video.

Sensory representations are techniques to convey information through the senses of a human being. As the visualization is trying to convey information through the eye, i.e. the visual sense, the basic rules for sensory representations also apply to visualizations (Carbon, 2014). An important property of a sensory representation is that it should be understood without training (Ware, 2019, p. 12). The perceiver should be able to identify the shapes and forms in the visualization, this does not mean that he immediately understands what these shapes and forms express, however our visual systems are built to perceive the shapes of 3D/2D surfaces. When trying to visualize something, one should therefore focus on using simple forms (like arrows, rectangles or circles) whenever possible in order to enable easy and intuitive understanding of these shapes.

Moreover, visualizations can be subjects of illusions (Carbon, 2014). These illusions can persist, even when the perceiver is aware of them (Ware, 2019, p.12).

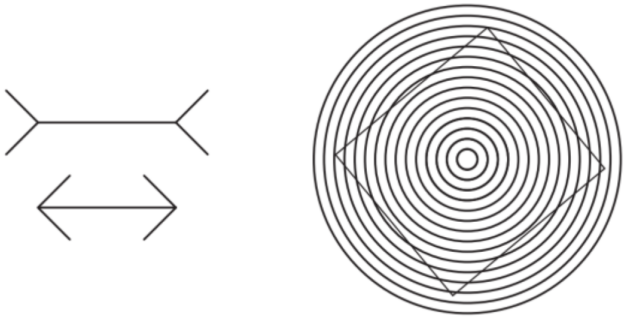

Figure 2.1: In the Muller-Lyer illusion on the left, the horizontal line in the upper figure appears longer than the one below. On the right hand side, the rectangle appears distorted into a pincushion shape (Ware, 2019, p.13)

Illusions as depicted in 2.1 are very likely to be misleading in diagrams (Carbon, 2014), but also in explanatory visualizations. Illusions should be avoided not only visually but also logically, to ensure the visualization is perceived in the way it was intended to. Logical illusions can be caused through careless visualizations, e.g. connecting two points with a straight line in a graph leads the perceiver to interpreting that there is also information in between the points, which is obviously a bad idea if there is in fact none (Ware, 2019, p.12).

Moreover, when displaying different groups of symbols that should depict different areas, the cross-cultural validity has to be taken into account (Gregory, 2015; Ware, 2019, p.13, p.4). Different data patterns should be perceived quickly and leave no room for misinterpretation.

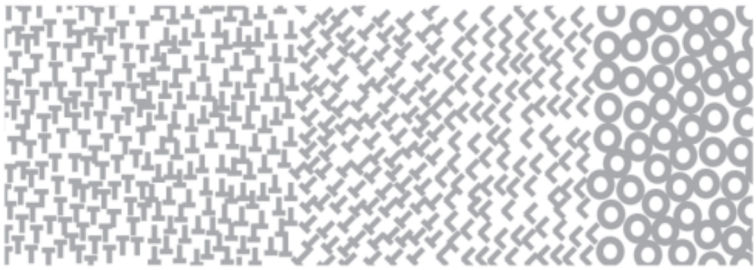

Figure 2.2: Some regions are easier to distinguish from each other than others (Ware, 2019, p.13)

Therefore, visualizations of different groups with (almost) similar symbols (like on the left hand side of figure 2.2) should be avoided. To create understandable visualizations like diagrams or models, one has to assure that important data elements and patterns can be perceived quickly and correctly by the observer (Ware, 2019, p.14). Another important guideline to follow whilst designing comprehensive visualizations are the standardization of graphical symbol systems across the video (Ware, 2019, p.16).

2.2 Model of human Perceptual Processing

Ware (2019) simplifies the model of human information processing, to provide a high-level understanding about how we understand what we see.

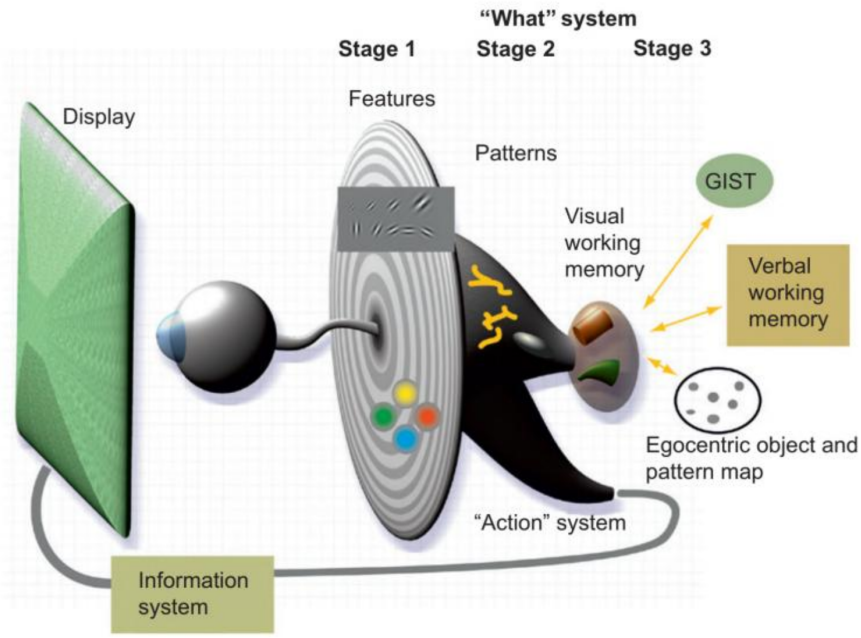

Figure 2.3: A three-stage model of visual information processing (Ware, 2019, p.20).

Stage one of the model is the extraction of low-level properties through parallel processing (Ware, 2019, p.20). In this first stage, billions of neurons in our eyes extract the features that they are fine-tuned on (e.g. edges and colours). This process is parallel and continues to be, independent from our choice of attention (Willis & Todorov, 2006). Subsequently, to enable a fast understanding of the information that we display it is crucial to make the information attainable by these fast computing neurons (e.g. use clear and common shapes, colors and movement patterns) (Ware, 2019, p.20/21).

In the second stage the perceived image is segmented into simple patterns or regions that share characteristics (e.g. same color, motion). This process happens slower than the stage on feature extraction and, also contrary to stage one, can be influenced by the brain of the observer. The attention is usually top-down, which makes the formation of objects a critical factor while explaining a sequential process for instance (Ware, 2019, p.21). The brain processes the recognized patterns in one of two ways: the action system or the object identification, which is the basis for the two-visual-systems theory by Milner and Goodale (2006). For the perception of information, the object identification part is usually more important.

The third stage of perception is much more influenced by what we are doing or what problems we are trying to solve, then by what we see. Moreover, our attention determines which information we presume as important. Attention is a fine-tuned process which tries to remember

what we have seen in the past and anticipate what will be important in the future (Ware, 2019, p.22). However, although the process of attention is highly complex and depends on a facet of factors inside of the human brain, Ware (2019) states that by clever design and following the foregoing guidelines and best practices one can influence the visualization to force an arbitrary piece of information into the center of attention.

To conclude, the comprehensiveness and effectiveness of the conveying of an information is heavily dependent on the design of the visualization. The human perception process as well as traps, illusions and best practices have to be taken into account by the designer, in order to ensure that the observer understands the visualization at hand. If this is done properly however, it is possible to "manipulate" the attention in a way that facilitates the understanding. This is the core challenge of educational visualization.

2.3 Computer Graphics

After we dived into the foundations of human perception and how to visualise information in a comprehensive and educational way, this chapter explores different ways of displaying/animating these visualizations on a computer. With the course of time many different tools for this purpose emerged. Firstly, we will introduce the general field of computer graphics. Following this, we will give a brief overview about what those tools are and discuss their role in the process of creating an educational video.

Computer graphics, more precise computer graphic animation started in the year 1964, when the graduate student Ivan Sutherland in his Technical Report No. 296 "Sketchpad: A Man-Machine Graphical Communication System" wrote "Sketchpad need not be restricted to engineering drawings. Since motion can be put into Sketchpad drawings, it might be exciting to try making cartoons" (Sutherland, 1964). Within the last 60 years this field came a long way from hand sketched cartoons to the stunning Computer Generated Imagery (CGI) effects used in todays Hollywood movies (Sito, 2013, p.2).

The field of computer graphics can be divided into several core elements including modeling, rendering, interaction, and animation (Alesandrini, 1987). Modeling involves the creation of a digital representation of an object which can then be displayed. Modeling proves to be especially useful when trying to visualise processes or things, that are too big/small/expensive etc. to film them with a normal camera.

Rendering is the process of generating an image from a model, which includes geometric transformation, visibility determination, and simulation of light. Through rendering the previously created models can be displayed realistically and in 3D. This makes models (almost) lifelike and enhances the comprehensiveness of an explanation because the observer sees the explained process similar to what it would be in real life.

Interaction is the use of input/output devices and tools to engage with and manipulate the graphical information on the display. This is especially useful when trying to create interactive educational content, however it is not relevant when creating a video and is therefore neglected in this paper.

Lastly, animation involves creating lifelike characters, natural phenomena, and their interactions (Alesandrini, 1987). This is arguably the most important part of computer graphics regarding explainability and comprehensiveness. With animations we can create movements and processes (almost) like they happen in the real world, manipulate their speed and zoom in on details. This enables observers to grasp even the most complex and theoretical ideas by creating an animated model that visualises the core concepts.

2.3.1 VFX and CGI

Visual effects (VFX) and Computer Generated Imagery (CGI) are two closely related concepts in the field of computer graphics. Visual effects refer to the process of manipulating existing live-action shots or create new imagery from scratch. CGI is a specific subset of VFX which refers to the creation of images, animations or visual effects using only computer graphics (Kumar, 2022).

VFX is about adding or enhancing visual elements that are not present in the original footage. The techniques go from compositing and motion tracking to green screen keying, matte painting and many more (Kumar, 2022) and will be explained in the following. Compositing is the technique of combining multiple images into one. The most popular example of compositing its combination with green screen keying. By filming an actor in front of a green (or blue) screen, the creators can replace the background with any image or video, even despite varying camera angles or lighting conditions. Motion tracking involves generating a 3D camera to help create 3D characters and objects that can be seamlessly integrated with the original footage (Jackson, 2016). Lastly, matte painting is a technique used to create or enhance backgrounds or environments that are difficult or impossible to capture during filming. It involves painting or digitally creating detailed backgrounds that can be seamlessly integrated into the footage, adding depth and realism to the scene (Pardeshi & Karbhari, 2019). To summarize, VFX can solve for realistic or fantasy elements in film and video production that would be difficult or impossible to capture filming with a real camera.

As mentioned before, the term CGI refers to a specific subset of VFX (Kumar, 2022). CGI is widely used in professional film and video production to generate both 2D and 3D models, illustrations and digital images. It allows artists to create lifelike or stylized visuals that can be seamlessly integrated into live-action footage or used to create entirely computer-generated scenes. Through the advancement of technology and the increasing availability of computing power, CGI artists are nowadays able to create CGI imagery in 2D and 3D with a level

of incredible detail and realism (Pandian et al., 2021). This enables the creation of visually immersive and believable environments without a real camera being involved. Subsequently, in these digital worlds the laws of nature can also be bent arbitrarily which, regarding educational videos, enables the explanation and visualization of abstract ideas with an unseen level of complexity and authenticity.

In summary, VFX encompasses a broader range of techniques used to manipulate or enhance visual elements in film and television, while CGI specifically refers to the creation of images and animations using computer graphics. CGI is a subset of VFX and is often used as a tool within the VFX process, it can however also be used independently and not only in combination with other VFX tools.

2.3.2 Manim

After learning about the basics of human perception, computer graphics and the concepts of VFX and CGI which enable computers to create lifelike visual representations of objects, this chapter highlights a tool that enables the creation of educational videos with mathematical correctness.

Manim is a Python library for creating mathematical animations, developed by Grant Sanderson (also known as 3Blue1Brown on Social Media). The original library has then split into the "traditional" manim and a community maintained version. Manim is a framework for the creation of animations using code, which enables mathematical control over paths of moving objects and more (ManimCommunity, 2024). Whilst this might be a challenge in the beginning, creating a video through code could confront non-technical users with unforeseen difficulties due to missing knowledge, once mastered the library lets the creator visualise mathematical transformations, calculations and much more. This is especially useful when creating educational videos about topics that are based on mathematical or physical concepts.

For our case, the Principal Component Analysis, Manim offers many different visualization opportunities like scatter plots or complex movements of items in the three-dimensional space to exemplify the transformation of data. As the animation is generated through mathematical expressions in the code, the data in the (conceptual) visualization behaves in an exact manner which assures there are no ambiguities for the observer to stumble upon.

These characteristics make Manim highly suited for an explanatory video on PCA. We design the video to enable the observer to gain a deep understanding of the Principal Component Analysis through comprehensive and exact modelling of the concepts and ideas behind the data transformation.

2.3.3 Challenges during implementation with Manim

During the creation of our educational video about the Principal Component Analysis with Manim we faced various challenges. By listing these challenges and presenting our workarounds, we aim to provide effective solutions for common problems during the implementation with Manim.

One of the biggest Challenges was the lack of code segmentation. Due to the fact that our animation shows how the PCA works step by step, a lot of modifications to the same object had to be done. This prevented us from splitting the code into many different scenes. Scene properties have to be defined on the highest level Scene, resulting in a very bloated scene with bad naming convention that consists of duplicate names with prefixes. To fix that, we tried using many scenes with the same code to have less bloated scene. This however made the transitions between the different scenes very challenging. We ended up calling functions that perform certain modifications to a scene given as argument. Essentially, the code was still somewhat monolithic since the functions could not be entirely independent of each other.

Another challenge we faced was the system of reference. To give a good intuition for how PCA works, we used a three-dimensional space for the animations. To make sure the viewer has the most telling point of view, the camera angle was a constant concern. This became even more challenging because we faced multiple reference systems, like the radial, degree-based or the cartesian coordinate system. When rotating the plot around the z-axis, to ensure the viewer can comprehend as much information as possible, the center of rotation remains fixed. However, when shifting the entire plot aside to reveal a formula describing our actions, the center of rotation remains stationary and does not move along with the plot. Therefore, the plot did not rotate around itself, it rotated around a empty point in space. In order to preserve as much information as possible the point of view has to be aligned with the direction of the third principal component. To convert from this vector (the third PC vector) to a position in the coordinate system, we implemented a function called calculate_view_pos.

A third challenge we had to overcome is the lack of functionality. While for other creators like Grant Sanderson (know on Youtube as 3Blue1Brown) Manim is second nature, it was new for us. By the lack of customizability of some of the functions we had to create some workarounds for our animation to work. For example the Axes class, which we used frequently, has an option to show its number scale. However, the number scale only goes towards positive infinity. This would result in a positive labeling even for negative parts of the axis and would be mathematically wrong. Therefore we had to find a workaround. After trying out different ideas we decided to use arrows as the axis of our coordinate system.

3 Principal Component Analysis

3.1 Motivation and Historical Background

In the age of Industry 4.0, where a large number of devices in manufacturing, smart homes or transportation gather immense amounts of data (Transforma Insights, Exploding Topics, 2023), using this data in an efficient way becomes one of the key challenges. Often, such data is to be used for predictive maintenance or performance analysis. Frequently, most of a dataset's variance can be retained using only a limited number of features. Finding efficient ways to compress the information into a lower-dimensional representation that still contains most or all of the original information is key to save cost on processing in downstream applications.

This motivates the need for methodology that can detect and disentangle covariance in a dataset. Covariance can be seen as a form of redundancy when it comes to dimensionality reduction, as by definition it shows that knowledge about one feature of an instance explains part of the variance (i.e. information) of another feature of that instance. PCA is such a methodology as it transforms the variance onto so-called principal components that are uncorrelated. That way we can push the maximum amount of information possible onto the first k number of principal components, where k is the number of dimensions our transformed dataset should have.

Principal Component Analysis (PCA) is a powerful statistical technique used for dimensionality reduction and feature extraction in data analysis. The fundamental idea behind PCA is to transform a set of correlated variables into a new set of uncorrelated variables, called principal components, which capture the maximum variance in the data. By identifying and retaining the most significant information in the dataset, PCA allows for the simplification of complex data structures while preserving essential patterns and relationships. This process not only helps in reducing computational complexity but also identifies the key features that contribute most to the overall variability in the data (Jolliffe, 2002, p. 1).

The roots of PCA trace back to the late 19th and early 20th centuries. Beltrami and Jordan independently developed the Singular Value Decomposition (SVD), a crucial component of PCA, in the 1870s. Pearson and Hotelling formalized PCA in 1901 and 1933, respectively (Brunton & Kutz, 2024, p. 23). Pearson's focus was on finding optimal lines and planes in multidimensional space, leading to the concept of principal components. Hotelling introduced the term "principal component" and framed PCA as a method to identify a smaller set of independent variables contributing maximally to the variance. Although PCA's early development saw limited activity, subsequent contributions by Girshick in the mid-20th century solidified

its theoretical foundations and practical applications. With the advent of computers, PCA's widespread adoption became feasible for larger datasets. Today, PCA is widely used in various fields, such as image processing, finance, and biology, offering insights into underlying patterns and structures within high-dimensional datasets (Jolliffe, 2002, pp. 6-9).

3.2 Mathematical Foundation

PCA relies on key linear algebra concepts to transform and extract meaningful information from datasets. The mathematical foundation of PCA is built upon the following fundamental elements:

- Covariance Matrix
- Eigenvalues and Eigenvectors

3.2.1 Covariance Matrix

The covariance matrix is a central component in PCA, capturing the relationships between different variables in a dataset. For a dataset with n variables, the covariance matrix (Σ) is a $n \times n$ symmetric matrix. The diagonal elements contain the variances of individual variables, while the off-diagonal elements represent the covariances between pairs of variables. In PCA, the covariance matrix serves as the basis for understanding the data's variability and relationships (Jolliffe, 2002, pp. 87-88). Mathematically, the covariance between two variables X_i and X_j is defined as:

$$\operatorname{cov}[X_i, X_j] = \mathrm{E}\left[(X_i - \mathrm{E}[X_i])(X_j - \mathrm{E}[X_j])\right]$$

Before performing the PCA, we center the data to extract the effects of the mean in later steps. Therefore, the expected value of each value is zero in our case and we can express the covariance as follows:

$$\operatorname{cov}[X_i, X_j] = \mathrm{E}[X_i * X_j]$$

The covariance matrix in PCA forms the base for comprehending the dataset's variability and inter-variable relationships.

Since we calculate the covariance only to find the *directions* of maximum variance, i.e. the absolute amount of covariance is irrelevant, the calculation of the covariance matrix for centered

data matrix $B = X - \bar{X}$ simplifies to the following for PCA:

$$\mathrm{Cov}(B) = B^T B$$

3.2.2 Eigenvalues and Eigenvectors

Eigenvalues and eigenvectors play a crucial role in PCA, providing a way to decompose the covariance matrix. By solving the characteristic equation of the covariance matrix (Σ), we obtain eigenvalues and corresponding eigenvectors. The eigenvalues represent the amount of variance associated with each eigenvector. The eigenvectors, in turn, indicate the directions in which the data varies the most (Jolliffe, 2002, pp. 86-87). The eigenvectors are orthonormal and form the basis for the principal components, which are linear combinations of the original variables capturing the maximum variance. The characteristic equation takes the form:

$$\det(\Sigma - \lambda \mathbf{I}) = 0$$

Where $\mathbf{I}$ is the identity matrix. Once the eigenvalues are obtained, the eigenvectors can be found by solving $(\Sigma - \lambda \mathbf{I})\mathbf{V} = 0$. The eigenvectors then serve as the foundation for constructing principal components ($\mathbf{PC}$), which are linear combinations of the original variables designed to capture the highest variance (Jolliffe, 2002, p. 5):

$$\mathbf{PC_k} = \mathbf{V_k'X}$$

In summary, the mathematical intricacies of PCA rely on foundational concepts, enabling a rigorous analysis of complex datasets.

3.3 Principal Component Analysis Algorithm

This chapter outlines the steps involved in the PCA algorithm, from computing the covariance matrix to reducing the dimensionality of the data. The algorithm typically consists of the following steps:

1. **Standardization:** Standardize each variable in the dataset. This ensures that all variables are on a similar scale (Jolliffe, 2002, p. 21).
2. **Covariance Matrix:** Compute the covariance matrix of the standardized data. The covariance matrix captures the relationships between variables (Jolliffe, 2002, p. 21).

3. **Eigenvalue Decomposition:** Calculate the eigenvalues and corresponding eigenvectors of the covariance matrix. These eigenvectors represent the principal components, and the eigenvalues quantify the variance along each component (Jolliffe, 2002, p. 86).
4. **Sort Eigenvalues:** Sort the eigenvalues in descending order to prioritize the principal components with the greatest variance (Jolliffe, 2002, p. 75).
5. **Select Principal Components:** Choose the top k eigenvectors based on the desired dimensionality (k) for the reduced feature space (Jolliffe, 2002, p. 112).

The i-th principal component is computed as a linear combination of the original variables:

$$PC_i = \mathbf{V}_i \cdot \mathbf{X}$$

where $\mathbf{V}_i$ is the i-th eigenvector and $\mathbf{X}$ is the standardized data matrix.

The final step in the PCA algorithm is dimensionality reduction. By selecting the top k principal components, the dataset is projected onto a subspace that retains the most significant information (Jolliffe, 2002, p. 112). This lower-dimensional representation is achieved through the transformation:

$$\text{Reduced Data} = (\mathbf{V}_k^T \cdot \mathbf{X}^T)^T$$

where $\mathbf{V}_k^T \in \mathbb{R}^{m \times k}$ is the matrix composed of the k selected eigenvectors and $\mathbf{X}^T \in \mathbb{R}^{m \times n}$ is the standardized data matrix.

In summary, PCA involves complex computations. Its core is the generation of the covariance matrix, which has a time complexity of $O(dn^2)$. Here, d represents the dimensionality of the data and n signifies the number of samples. Performing eigendecomposition on the covariance matrix is computationally expensive, with a cubic complexity of $O(d^3)$. In cases where d significantly surpasses n, it is beneficial to compute eigenvalues of $\frac{1}{n}X^TX$ - in practice $\frac{1}{n-1}X^TX$ is used to reduce the bias of the estimation (Shlens, 2014, p. 5). This strategy reduces the computational burden of the PCA from $O(d^2n+d^3)$ to $O(d^2n+n^3)$. Moreover, it is not always necessary to perform a complete eigendecomposition. Extracting only the top-k eigenvalues and corresponding eigenvectors offers a faster solution. SVD can also be used as an alternative, delivering a runtime of $O(mdk)$, matrix m * dimensions d * basis vectors k, with higher numerical precision (Fetaya et al., n.d., p. 20).

3.3.1 Interpretation of the Results

This section dives into the interpretation of the PCA results, providing a comprehensive understanding of the principal components. The principal components obtained from PCA represent the directions of maximum variance in the dataset. Each principal component is a linear combination of the original variables (Jolliffe, 2002, p. 14). The interpretation involves understanding the contribution of each principal component to the overall variability and identifying patterns in the data. In PCA "loadings" refer to the coefficients of the linear combination of the original variables used to construct the principal components. They represent the contribution of each original variable to the principal components. In scikit-learn, after applying PCA to a dataset, the "loadings" can be obtained from the components_ attribute of the PCA object, and they provide insights into the weights each original variable has in the principal components (scikit-learn, 2024a).

Eigenvalues associated with each principal component quantify the amount of variance captured by that component. Larger eigenvalues correspond to principal components that explain more variance in the data. The significance of eigenvalues helps selecting principal components that contribute significantly to the overall variability (Jolliffe, 2002, p. 75).

To determine the impact of individual principal components on the overall variance, we can analyze the explained variance. This metric represents the proportion of the total variance that can be attributed to each specific principal component. We can also calculate the cumulative explained variance, which represents the total variance explained by including a certain number of principal components (Jolliffe, 2002, p. 112). These metrics can be used to determine how many principal components should be retained for dimensionality reduction.

When performing PCA, it's crucial to determine the number of principal components to keep. A widely used approach is to examine the cumulative explained variance and select a threshold (such as 90% or 95%) that indicates the number of components that can sufficiently account for the variability in the data. Additionally, tools like scree plots, which display eigenvalues in descending order, can be helpful in determining the appropriate number of principal components (Jolliffe, 2002, p. 116).

3.3.2 Choosing k Principal Components

PCA not only helps in dimensionality reduction but can also provide graphical insights into the underlying structure of high-dimensional data. This section explores various methods for visualizing PCA results.

One of the primary objectives of PCA is to transform high-dimensional data into a lower-dimensional subspace for visualization. Scatter plots of observations based on the first two or three principal components provide a straightforward representation of the data. Even in

the presence of a few more dimensions, interactive computer graphics can help gain a visual impression of the data (Jolliffe, 2002, p. 80).

In addition to scatter plots, two other essential graphical tools for understanding PCA results are scree plots and cross-validation methods. A scree plot is a diagnostic tool used to assess the number of principal components to retain in a PCA analysis. It displays the eigenvalues of each principal component in decreasing order. The point where the eigenvalues start to level off indicates the number of principal components that capture most of the variability in the data. Including principal components beyond this point might not contribute significantly to the representation of the data (Jolliffe, 2002, p. 116).

Cross-validation is a statistical technique used to assess the performance and generalization of a model. In the context of PCA, cross-validation methods help evaluate the effectiveness of the dimensionality reduction and visualization. Techniques like k-fold cross-validation can provide insights into how well the PCA results generalize unseen data (Jolliffe, 2002, p. 123).

3.4 Practical Considerations

PCA is a powerful technique for dimensionality reduction and pattern recognition in high-dimensional datasets. However, successful implementation requires careful consideration of various practical aspects, particularly in the preprocessing of the data. This section explores essential considerations in preparing the data for PCA, emphasizing the significance of standardization and handling missing data.

3.4.1 Standardization

Standardization is a crucial process in performing PCA. This is because PCA is sensitive to the units of measurement used for each element of the input data. When the elements of the input data are of different types, such as lengths, weights, temperatures, or arbitrary scores on a scale, standardization becomes even more critical. If the input data is not standardized, the results of PCA can be heavily influenced by the magnitudes of the variances of individual variables. Variables with larger variances can dominate the principal components, leading to misinterpretations and an inadequate representation of the underlying structure in the data. By standardizing variables, PCA ensures that each variable contributes proportionally to the principal components, irrespective of its original scale. (Jolliffe, 2002, p. 22).

In summary, standardizing variables is a critical step to mitigate scale dependence in PCA. It enables a more accurate representation of the underlying structure in datasets with diverse types of measurements (Jolliffe, 2002, p. 24). Standardizing the data to have a mean ($\bar{x}$)

of 0 and a standard deviation (σ) of 1 ensures that each variable contributes equally to the PCA process (scikit-learn, 2024b).

3.4.2 Dealing with Missing Data

Addressing missing data is another critical consideration in the application of PCA. The presence of missing values can disrupt the computation of principal components and compromise the reliability of the results (Jolliffe, 2002, p. 363).

Missing data can create obstacles in performing PCA as it can affect the precise calculation of covariance and correlation matrices, which are crucial for this analysis. If missing values are ignored or unsuitable strategies are applied, it can result in biases and distortions in the principal components, leading to incorrect interpretations of the dataset's structure (Jolliffe, 2002, p. 363).

There are two main approaches to deal with missing data in a dataset: imputation and exclusion. Imputation techniques use observed data to estimate missing values, which can be done through methods like mean or median imputation. Exclusion strategies, on the other hand, involve removing observations or variables that have missing values. The decision on which approach to use depends on how much data is missing and the pattern of missingness in the dataset (Jolliffe, 2002, p. 363).

3.5 Comparison with other Techniques

In the realm of dimensionality reduction, Principal Component Analysis (PCA) stands as a widely employed technique, offering valuable insights into data representation. However, a comprehensive understanding of its strengths and limitations becomes imperative when evaluating its efficacy against alternative methods. This section delves into a comparative analysis between PCA and two prominent dimensionality reduction techniques, namely t-Distributed Stochastic Neighbor Embedding (t-SNE) and Locally Linear Embedding (LLE). By exploring the distinctive features of each method, we aim to provide a nuanced perspective on their respective applicability and performance. Furthermore, we extend our analysis explaining how Autoencoders emerged to tackle non-linear settings as well as going into more detail on Singular Value Decomposition (SVD) in this comparative study.

3.5.1 Autoencoders

Today, dimensionality reduction is often done using autoencoders (W. Wang et al., 2014; Y. Wang et al., 2016). A core difference between PCA and neural network based approaches

is that while PCA is a linear transformation, neural networks allow the introduction of non-linearities into the process. Another advantage of neural network based dimensionality reduction is that during training, we can encourage the model to let these mostly lower-dimensional latent representations of the input assume certain properties (more on that in Section 3.5.1 - Variants of Autoencoders).

Relationship between Autoencoders and PCA

Baldi and Hornik, 1989 showed that a single-layer encoder without a non-linear activation function converges to a global minimum where the resulting transformation is the same as the corresponding PCA transformation. Depending on the amount of dimensionality reduction enforced by the bottleneck layer of the autoencoder, this resulting transformation projects into a subspace of the PCA. This finding underlines how both the analytical PCA method and the iterative training process of an autoencoder (Kramer, 1991) come to the same result in finding the linear transformation that allows us to project our data into another space where as much information as possible is retained using the least possible[1] amount of dimensions, just as we would expect.

This interesting relationship however is not pure coincidence. The autoencoder was developed by Kramer, 1991 in the context of non-linear dimensionality reduction as a next step to PCA.

Variants of Autoencoders

Attempts to make the autoencoder framework robust against small perturbations of the input and in order to enable to generalize better when applied to unseen data, several regularized variants of the autoencoder have been proposed. Rifai et al., 2011 proposed penalizing the weights of the encoder using the squared Frobenius norm of the encoder's Jacobian and Vincent et al., 2008 add noise to the input and train the autoencoder to remove the noise in pursuit of more stable and robust latent representations of the input.

More recently, Variational Autoencoder (VAE) have been used to enforce certain characteristics of the lower-dimensional representation, i.e. the embedding of the input. As the VAE builds on a stochastic approach where the latent space is defined using a distribution, the introduction of priors becomes a tool for enforcing certain characteristics of the embedding. Higgins et al., 2017 proposed the disentangled variational autoencoder, also referred to as β-VAE. This variant of the autoencoder encourages the model to learn extremely low dimensional representations of the input in an image processing setting, which is often times referred to "data-generative" factors. For instance, with hand-written digits, these data-generative factors would be the type of number, the thickness of the line, the angle, the width and the

[1] The least possible amount using a linear transformation

tilt of the number. Disentangling these data-generative factors allows for a highly efficient low-dimensional representation of the input far beyond PCA and regularized autoencoders.

Today, the use of autoencoders goes far beyond dimensionality reduction. The autoencoder has become a universal framework for learning efficient and meaningful latent space representations across various domains such as Natural Language Processing, Computer Vision, Time-series Forecasting (Ronneberger et al., 2015; Salinas et al., 2020; Vaswani et al., 2017), where all of the mentioned domains use their own highly adopted versions of the broader encoder-decoder framework.

3.5.2 Singular Value Decomposition (SVD)

This section explores a more advanced mathematical solution for PCA by introducing SVD, which provides a comprehensive understanding of basis transformation. PCA and SVD are revealed to be closely related, often used interchangeably due to their deep connection.

SVD is a mathematical technique applied to an arbitrary matrix X with dimensions $n \times m$. The initial step involves constructing orthonormal eigenvectors $\{\mathbf{v}^1, \mathbf{v}^2, ..., \mathbf{v}^r\}$ from the eigenvalues $\{\lambda_1, \lambda_2, ..., \lambda_r\}$ of the matrix $X^T X$, a symmetric $m \times m$ matrix. Simultaneously, a diagonal matrix Σ is formed, with singular values σ_i representing the square roots of the eigenvalues and arranged in descending order. The vectors $\{\mathbf{u}^1, \mathbf{u}^2, ..., \mathbf{u}^r\}$ are then defined as $n \times 1$ vectors, each equal to $\sigma_i X \mathbf{v}^i$. Notably, these vectors possess orthonormality properties, as expressed by the Kronecker delta, and play a pivotal role in the scalar version of SVD: $X\mathbf{v}^i = \sigma_i \mathbf{u}^i$ (Shlens, 2014, p. 7).

Moving to the matrix representation, two matrices, V and U, are constructed, with their columns containing the orthonormal eigenvectors and singular value-related vectors, respectively. The SVD equation $XV = U\Sigma$ is established, and by multiplying both sides by the transpose of V, denoted as V^T, due to orthogonality, the final decomposition formulates as $X = U\Sigma V^T$ (Shlens, 2014, p. 7).

This decomposition reveals the matrix X as a product of an orthogonal matrix (U), a diagonal matrix with singular values (Σ), and another orthogonal matrix (V^T), providing profound insights into the inherent structure and characteristics of the original matrix (Shlens, 2014, p. 7).

Interpreting SVD

At its core, SVD provides a way to decompose a matrix into three simpler matrices, revealing the underlying structure and intrinsic features of the original data. The intuition behind SVD lies in the idea of expressing a matrix as a combination of elementary transformations.

Essentially, SVD breaks down a matrix into three components: a left singular matrix, a diagonal matrix of singular values, and a right singular matrix. The singular values capture the importance of each dimension or feature in the original data, enabling dimensionality reduction and noise reduction. The intuitive power of SVD lies in its ability to reveal the latent patterns and relationships within complex datasets, making it a fundamental tool in the realm of data analysis and numerical computations. SVD is a linear transformation that can be understood by envisioning a square. SVD avoids shearing and only allows predictable changes. The singular values in SVD represent the length and width of the transformed square. If one singular value is zero, it flattens the square. The larger singular value indicates the maximum "action" of the transformation. SVD helps uncover essential geometric changes and magnitudes associated with linear transformations (Gundersen, 2018).

SVD provides a detailed view of the transformed basis vectors. It introduces the concept of orthonormal sets, specifically $\{\mathbf{v}^1, \mathbf{v}^2, ..., \mathbf{v}^n\}$ and $\{\mathbf{u}^1, \mathbf{u}^2, ..., \mathbf{u}^n\}$, which encompasses all conceivable "inputs" and "outputs". This idea is formalized through the definition of $\mathbf{Z}$, where $\mathbf{Z} \equiv U^T\Sigma$. $\mathbf{Z}$ acts as a transformation from the transpose of X to X, capturing the essence of the entire basis transformation process (Shlens, 2014, pp. 7-8).

The row space interpretation is significant, where XV equates to $\Sigma U(XV)^T = (\Sigma U)^T$. This equality emphasizes that matrix V spans the row space of X and portrays the possible "inputs" into any arbitrary matrix. Consequently, the row space interpretation provides a profound understanding of the transformed basis vectors in the context of SVD, offering valuable insights into the potential variations in the input space of a given matrix (Shlens, 2014, pp. 7-8).

Relationship between SVD and PCA

The relationship between SVD and PCA is quite fascinating as PCA is a specific case of the more general SVD. If we introduce a new matrix Y of dimensions $n \times m$, and then perform PCA by calculating the SVD of Y, we will find a deep connection between SVD and PCA. The resulting matrix V from the SVD of Y contains the principal components of the original matrix X. It is important to note that V not only covers the row space of Y but also shows the column space of X. This connection highlights the versatility of SVD, emphasizing its role in capturing both the variation in the data (via PCA) and the potential transformations in the input space. This is achieved by diagonalizing the covariance matrix of X, allowing PCA to identify the primary axes along which our data varies. In summary, the powerful SVD decomposition allows us to represent any matrix X as the product of an orthogonal matrix, a diagonal matrix, and another orthogonal matrix, providing deep insights into basis transformations and establishing a vital link with the principles of PCA (Shlens, 2014, pp. 8-9).

3.5.3 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a dimensionality reduction technique created by van der Maaten and Hinton in 2008 and is widely used for visualizing high-dimensional data in two or three dimensions. The goal of t-SNE is to overcome the limitations of traditional methods like PCA by keeping local similarities in high-dimensional data while downplaying global structures. It does this by modeling the pairwise similarities between data points in both high and low-dimensional spaces using probability distributions, and utilizing the Student's t-distribution to measure similarities (van der Maaten & Hinton, 2008, p. 2583).

The algorithm is iterative and adjusts the positions of data points in the low-dimensional space to minimize the divergence between the probability distributions in the two spaces (van der Maaten & Hinton, 2008, p. 2587). Perplexity is an important parameter that affects the balance between local and global relationships during optimization (van der Maaten & Hinton, 2008, p. 2582). t-SNE is commonly used in data exploration, image analysis, natural language processing, and biology, providing valuable insights into complex datasets. However, users should be careful when interpreting the results as t-SNE is sensitive to hyperparameters and can cause distortions in representing distances between points in the original high-dimensional space (van der Maaten & Hinton, 2008, pp. 2589-2590).

Today, Uniform Manifold Approximation and Projections (UMAPs) has become increasingly popular in the field of visual embeddings, often supplanting t-SNE in many applications. UMAP is a dimension reduction technique rooted in manifold theory and topological data analysis. It leverages concepts from topology and category theory to construct a topological representation of high-dimensional data. By approximating the local manifold structure and creating fuzzy simplicial sets, UMAP builds a low-dimensional representation while minimizing the cross-entropy between the high-dimensional and low-dimensional topological representations. The algorithm starts by approximating the assumed manifold where the data lies, then constructs fuzzy simplicial sets to capture the topological structure. The process involves converting metric spaces into fuzzy simplicial sets using functors between categories, bridging the gap between incompatible local views of the data. The algorithm optimizes the layout of the low-dimensional representation based on the topological structure of the original data, enabling efficient dimension reduction while preserving essential data relationships (McInnes et al., 2020, pp. 3-12).

Relationship between t-SNE and PCA

Both t-SNE and PCA are widely used techniques in dimensionality reduction, but they have different approaches and objectives. PCA aims to identify the principal components, which are linear combinations of the original features, to preserve the variance in the data (Jolliffe, 2002, p. 1). On the other hand, t-SNE prioritizes the preservation of local similarities among the

data points in both high-dimensional and low-dimensional spaces (van der Maaten & Hinton, 2008, p. 2587). While PCA is effective for capturing linear relationships, t-SNE is known for its ability to capture non-linear structures, making it useful for revealing complex patterns and clusters in intricate datasets (van der Maaten & Hinton, 2008, p. 2580). In some cases, it is common to use PCA as a preprocessing step to reduce the dimensionality of the data before applying t-SNE. This approach helps to overcome some of the computational challenges associated with t-SNE and provides a more efficient way of capturing both global and local structures in high-dimensional datasets (van der Maaten & Hinton, 2008, p. 2589).

3.5.4 Locally Linear Embedding (LLE)

LLE is a technique used to reduce dimensionality, which was introduced by Roweis and Saul in 2000. LLE involves three main steps. Firstly, it creates a k-nearest neighbors (kNN) graph for all training points, forming a matrix that includes the k neighbors of each data point. In the second step, LLE determines weights for the linear reconstruction of each point based on its neighbors. It aims to maintain a linear combination of these neighbors. These weights are then used to embed each point into a lower-dimensional space, following the same linear combination principle. LLE's unique feature lies in using the same reconstruction weights in both the high-dimensional input space and the lower-dimensional embedding space (Ghojogh et al., 2020, p. 2).

The k-nearest neighbors are found using pairwise Euclidean distances between data points, and the linear reconstruction weights are obtained through an optimization process. The optimization problem ensures that the sum of weights for each point equals one, which prevents issues such as the explosion of weights. The final step involves embedding the data in the lower-dimensional space using the same weights obtained in the reconstruction step. This linear embedding is achieved through an optimization problem that minimizes the trace of a matrix involving the Laplacian of the weight matrix (Ghojogh et al., 2020, pp. 2-4).

LLE is useful for dealing with data that has several features. It has been used for feature fusion in cases where multiple sets of weights are obtained for different features. LLE's flexibility makes it a valuable tool for capturing complex relationships in high-dimensional data.

Relationship between LLE and PCA

LLE and PCA are two commonly used techniques in machine learning and data analysis to reduce the number of dimensions in a dataset. Although they have different underlying principles, there are certain connections between LLE and PCA. The primary focus of LLE is to preserve the local relationships within the data by reconstructing each data point as a linear combination of its neighbors. This emphasizes the local structures in the manifold (Ghojogh

et al., 2020, p. 1). On the other hand, PCA aims to identify the principal components which are linear combinations of the original features, capturing the directions of maximum variance in the data (Jolliffe, 2002, p. 1).

The connection between LLE and PCA becomes apparent when we consider LLE as a special case of kernel PCA. In this context, the inverse or negative sign of the matrix in LLE can be interpreted as its kernel. This connection highlights the relationship between linear and nonlinear techniques for dimensionality reduction and emphasizes the versatility of LLE in capturing intricate local structures in high-dimensional data (Ghojogh et al., 2020, pp. 7-8).

3.5.5 Kernel PCA

Kernel PCA is an advanced mathematical technique that expands the capabilities of traditional PCA. The primary purpose of Kernel PCA is to analyze complex data, extract essential features from it, and identify patterns and relationships among a set of data points in space (Schölkopf et al., 1997, p. 1).

In essence, Kernel PCA involves transforming the original data using a mathematical function called a "kernel". The primary objective of Kernel PCA is similar to standard PCA: to identify significant trends or variations in the data. However, Kernel PCA goes a step further by accommodating situations where the data is not linearly inseparable and projecting the data onto a higher dimensional space where it becomes linearly separable (Schölkopf et al., 1997, pp. 1-3).

The magic of Kernel PCA lies in its ability to compute dot products in the transformed space using kernel functions. This capability enables analysts to effectively comprehend and analyze complex datasets, particularly when dealing with nonlinear relationships in the data (Schölkopf et al., 1997, pp. 2-3). Kernel PCA can be applied in various domains such as image analysis, machine learning, and pattern recognition, where capturing intricate structures in the data is critical for meaningful analysis and interpretation (Schölkopf et al., 1997, p. 1). By leveraging kernel functions, Kernel PCA provides a powerful tool for dimensionality reduction and extracting valuable insights from complex datasets.

4 Summary

In conclusion, computer-aided visualisation is a pivotal tool for effectively communicating intricate information and concepts. We emphasized the importance of rules of perception, employing simple forms, and avoiding visual and logical illusions that might thwart a proper understanding. Cross-cultural considerations were highlighted to ensure universal understanding. The model of human perceptual processing illuminated the three-stage process, emphasizing the need for designing visualizations that align with the stages to enhance effectiveness.

The chapter also dived into the evolution of computer graphics, from the early days of Sketchpad to contemporary CGI effects. Computer graphics, encompassing modeling, rendering, interaction, and animation provide powerful t for visual storytelling. Animation, especially in the realm of VFX and CGI, allows for the creation of lifelike scenarios, breaking the bounds of reality to convey complex ideas with unparalleled authenticity.

The introduction of Manim, a Python library for mathematical animations, highlighted its significance in creating educational videos with mathematical precision. While initially challenging for non-technical users, Manim offers a unique advantage in visualizing mathematical concepts, ensuring the accuracy and comprehensibility of the presented information.

In essence, the art and science of visualization, coupled with advancements in computer graphics and animation tools like Manim, provide a robust framework for crafting educational content that captivates and fosters a deep understanding of complex subjects.

Shifting focus to Principal Component Analysis (PCA), in the realm of data analysis and pattern recognition, PCA emerges as a powerful tool for dimensionality reduction and feature extraction. Its historical roots trace back to the late 19th and early 20th century, with significant contributions from pioneers like Pearson and Hotelling.

PCA cleverly relies on the fact that the eigenvectors of the covariance matrix indicate orthogonal directions of maximum variance, thereby transforming an input data space with redundancies in the form of covariance into a space of disentangled uncorrelated variance. This enables to have dimensions that convey the maximum amount of information possible when using a linear method. We also mentioned how standardization and normalization is key when employing PCA in practice.

Comparisons with other techniques highlight PCA's strengths in computational efficiency and clear interpretation, acknowledging its limitations in handling nonlinear relationships. In the visual representation of PCA results, scatterplots, biplots, scree plots, and cross-validation methods play crucial roles in understanding the underlying structure of high-dimensional data.

In conclusion, PCA stands as a versatile and valuable technique, offering a comprehensive approach to dimensionality reduction and data analysis. Its integration of mathematical foundations, algorithmic steps, and practical considerations ensures its applicability in various domains, providing practitioners with a powerful tool for uncovering patterns and reducing complexity in their datasets. The synthesis of visualization principles and PCA methodology lays the groundwork for effective knowledge transfer, setting the stage for the subsequent exploration of PCA using Manim as a medium for educational content creation.

Bibliography

Alesandrini, K. L. (1987). Computer graphics in learning and instruction. In H. A. Houghton & D. M. Willows (Eds.), *The psychology of illustration: Volume 2: Instructional issues* (pp. 159–188). Springer US. https://doi.org/10.1007/978-1-4612-4706-7_6

Baldi, P., & Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, *2*(1), 53–58. https://doi.org/https://doi.org/10.1016/0893-6080(89)90014-2

Brunton, S. L., & Kutz, J. N. (2024). *Data Driven Science and Engineering* [Accessed: Jan 2, 2024]. http://databookuw.com/

Carbon, C.-C. (2014). Understanding human perception by human-made illusions. *Frontiers in human neuroscience*, *8*, 566.

Fetaya, E., Lucas, J., & Andrews, E. (n.d.). *Principal component analysis* [Accessed Jan 10, 2024]. https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec12_handout.pdf

Ghojogh, B., Ghodsi, A., Karray, F., & Crowley, M. (2020). Locally linear embedding and its variants: Tutorial and survey.

Girshick, M. A. (1939). On the Sampling Theory of Roots of Determinantal Equations. *The Annals of Mathematical Statistics*, *10*(3), 203–224. https://doi.org/10.1214/aoms/1177732180

Gregory, R. L. (2015). *Eye and brain: The psychology of seeing* (Vol. 80). Princeton university press.

Gundersen, G. (2018). *Singular value decomposition as simply as possible* [Accessed: 13th of January 2024]. https://gregorygundersen.com/blog/2018/12/10/svd/

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., & Lerchner, A. (2017). Beta-VAE: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*. https://openreview.net/forum?id=Sy2fzU9gl

Jackson, W. (Ed.). (2016). *Vfx fundamentals*. Apress. https://doi.org/10.1007/978-1-4842-2131-0

Jolliffe, I. T. (2002). *Principal component analysis* (2nd ed.) [ISBN: 978-0387954424]. Springer.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, *37*(2), 233–243. https://doi.org/https://doi.org/10.1002/aic.690370209

Kumar, A. (Ed.). (2022). *Beginning vfx with autodesk maya*. Apress. https://doi.org/10.1007/978-1-4842-7857-4

ManimCommunity. (2024). Manim: A community-maintained python framework for creating mathematical animations.

McInnes, L., Healy, J., & Melville, J. (2020). Umap: Uniform manifold approximation and projection for dimension reduction.

Milner, D., & Goodale, M. (2006). *The visual brain in action* (Vol. 27). OUP Oxford.

Pandian, A. P., Palanisamy, R., & Ntalianis, K. (Eds.). (2021). *Proceedings of international conference on intelligent computing, information and control systems*. Springer Singapore. https://doi.org/10.1007/978-981-15-8443-5

Pardeshi, A. S., & Karbhari, V. B. (2019). Recent trends in vfx (virtual effects) and sfx (special effects). *Int. J. Eng. Res. Technol*, *8*(07), 882–884.

Rifai, S., Muller, X., Glorot, X., Mesnil, G., Bengio, Y., & Vincent, P. (2011). Learning invariant features through local space contraction.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326. https://doi.org/10.1126/science.290.5500.2323

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, *36*(3), 1181–1191. https://doi.org/https://doi.org/10.1016/j.ijforecast.2019.07.001

Schölkopf, B., Smola, A., & Müller, K.-R. (1997). Kernel principal component analysis. In W. Gerstner, A. Germond, M. Hasler, & J.-D. Nicoud (Eds.), *Artificial neural networks — icann'97* (pp. 583–588). Springer Berlin Heidelberg.

scikit-learn. (2024a). *Sklearn.decomposition.pca* [Accessed: Jan 10, 2024]. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

scikit-learn. (2024b). *Sklearn.preprocessing.standardscaler* [Accessed: Jan 10, 2024]. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

Shlens, J. (2014). A tutorial on principal component analysis.

Sito, T. (2013). *Moving innovation: A history of computer animation*. MIT press.

Sutherland, I. E. (1964). Sketch pad a man-machine graphical communication system. *Proceedings of the SHARE design automation workshop*, 6–329.

Transforma Insights, Exploding Topics. (2023). Anzahl der mit dem internet der dinge (iot) verbundenen geräte weltweit von 2019 bis 2030 (in milliarden) [Accessed: 18th of January 2024]. *Statista GmbH*. https://de.statista.com/statistik/daten/studie/1420315/umfrage/anzahl-der-iot-geraete-weltweit/

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International Conference on Machine Learning*, 1096–1103. https://doi.org/10.1145/1390156.1390294

Wang, W., Huang, Y., Wang, Y., & Wang, L. (2014). Generalized autoencoder: A neural network framework for dimensionality reduction. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Wang, Y., Yao, H., & Zhao, S. (2016). Auto-encoder based dimensionality reduction [RoLoD: Robust Local Descriptors for Computer Vision 2014]. *Neurocomputing*, *184*, 232–242. https://doi.org/https://doi.org/10.1016/j.neucom.2015.08.104

Ware, C. (2019). *Information visualization: Perception for design*. Morgan Kaufmann.

Willis, J., & Todorov, A. (2006). First impressions: Making up your mind after a 100-ms exposure to a face. *Psychological science*, *17*(7), 592–598.