SEMINARARBEIT

Using PCA on EEG Data to Distinguish Sleep Stages

ausgeführt am

Institut für
Analysis und Scientific Computing
TU Wien

unter der Anleitung von

Assistant Prof.
Dipl.-Ing. Dipl.-Ing. Dr.techn. Andreas Körner, BSc

durch

Ida Hönigmann
Matrikelnummer: 12002348

Wien, am 07. Februar 2024

Contents

1 Introduction

This work aims to combine Principal Component Analysis (PCA), a data analysis tool, with electroencephalogram (EEG) recordings of sleeping persons. The main focus lies in giving an understanding of PCA as well as demonstrating an use case.

1.1 Data Analysis Tool

PCA is a data analysis tool that aims to find the best change of basis to apply to a given set of data points. Therefore it does not change the data points, but simply orients new axis through the data. These new axis are mathematically optimal in some sense and can be ordered. The ordering is done by a value representing the amount of information of the data along this axis.

When the data is high dimensional, PCA can be used to get few axis that sill describe most of the information in the dataset. Thus PCA is often used as a dimension reduction technique.

1.2 EEG Data and Sleep Stages

An EEG records the variations in potential in the brain. A bipolar EEG measures the potential between two electrodes placed on the head. In comparison a unipolar measures the potential between a electrode on the head and one on some other part of the body. In this work we are only concerned with bipolar EEG measurements.

Ganong [Gan97, chapter 11] describes typical patterns observed in EEG data of a sleeping person. He describes the EEG patterns associated with rapid eye movement (REM) sleep and non-REM (NREM) sleep. NREM sleep is further partitioned into three stages, termed Stage 1 (S1) to Stage 3 (S3)[1].

The EEG data of these stages is characterized by

S1: low amplitude, high frequency

S2: appearance of sleep spindles (bursts of higher amplitude, lower frequency waves)

S3: high amplitude, low frequency

[1] some authors use four stages

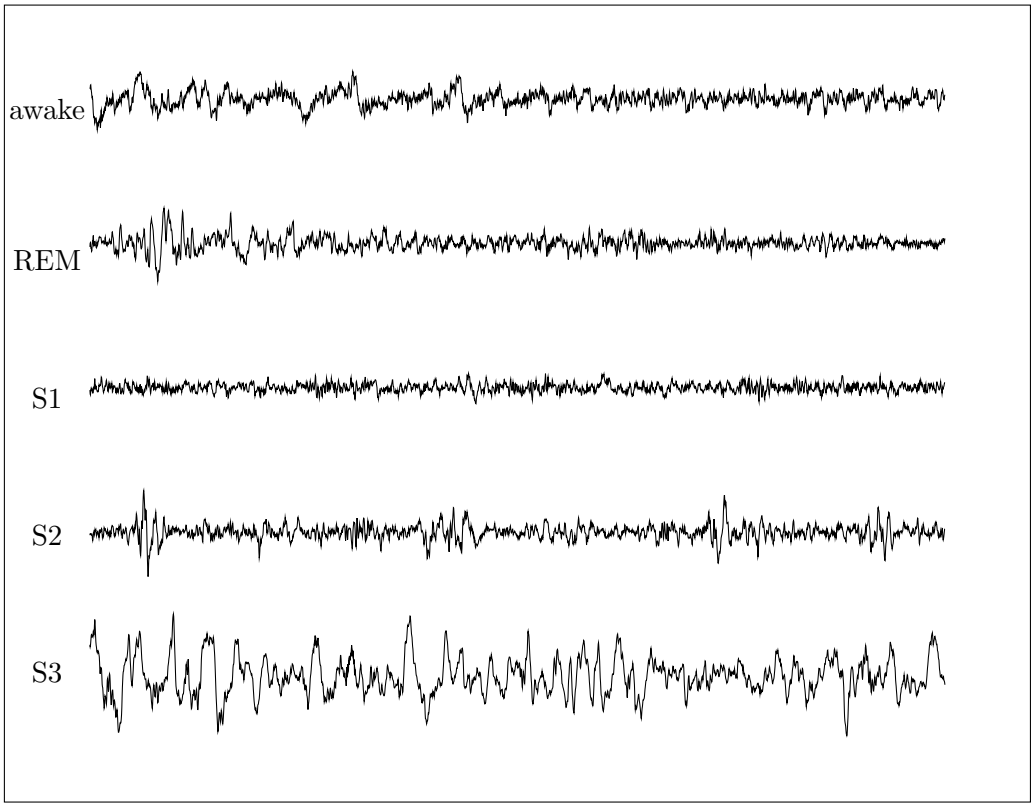


Figure 1.1: Short data segments of the the different sleep stages.

In REM sleep the EEG data is that of high frequency and low amplitude patterns, resembling the data observed in alert humans.

Example EEG data of these different sleep stages can be seen in Figure 1.1.

2 Study of Literature

As shown in this chapter, the utilization of PCA to analyze EEG data has been used with success.

2.1 Principal Component Analysis

A substantial body of scientific research has been devoted to exploring PCA. The foundation of this method was laid by Pearson [Pea01] and Hotelling [Hot33].

An introduction to PCA, as well as a good overview on how to derive the formula used to compute the principal components is given by Shlens [Shl14]. Recent applications and variants of PCA are explored by Jolliffe et. al. [JC16].

Shlens discusses the limitations of PCA, as well as examples in which PCA fails, such as the requirement of linearly dependent data. Tenenbaum proposes a non-linear method to combat this problem[TDSL00].

Generally speaking the variables must not have third or higher order dependencies[1] between them. In some cases it is possible to reduce a problem with higher order dependencies to a second order one by applying a non-linear transformation beforehand. This method is called kernel PCA[SSM97].

Another method for dealing with this problem is Independent Component Analysis (ICA) which is discussed by Naik et. al.[NK11].

2.2 Using PCA on EEG Data

The given problem of distinguishing sleep stages given some EEG data has been investigated by use of PCA, as well as neural networks. Some of these works are summarized below.

A review of different methods in the preprocessing, feature extraction and classification is given by Boostani et. al.[BKN17]. They find that using a random forest classifier[Bre01] and entropy of wavelet coefficients[CMP94] as feature gives the best results.

Tăuţan et. al.[TRdFI21] compare different methods of dimensionality reduction on EEG data, such as PCA, factor analysis and autoencoders. They conclude that the use of PCA and factor analysis improves the accuracy of the model.

[1] e.g. $\mathbb{E}[x_i x_j x_k] \neq 0$ for some i, j, k assuming mean-free variables

Putilov[Put15] used PCA to find boundaries between Stage 1, Stage 2 and Stage 3. Changes in the first two principal components were related to changes between the Stage 1 and Stage 2, while changes in the fourth principal component exhibited a change in sign at the boundary of Stage 2 and Stage 3. This suggests that changes between Stage 1 and Stage 2 are easier to detect that ones between Stage 2 and Stage 3.

Metzner et. al.[MST+23] try to rediscover the different human-defined sleep stages. They find that using PCA on the results makes clusters apparent. These clusters could then be used as a basis for a redefinition of sleep stages.

The PhysioNet/Computing in Cardiology Challange 2018[GMwHL+18] was a competition using a similar dataset. The goal was to identify arousal during sleep from EEG, EOG, EMG, ECG and SaO2 data given. The winning paper[HPPB18] of this competition describes the use of a dense recurrent convolutional neural network (DRCNN) comprised of multiple dense convolutional layers, a bidirectional long-short term memory layer and a softmax output layer.

3 Mathematical Basics

We define mathematical notation, which will be used in Section 4 to define PCA.

3.1 Covariance

Assume we have two sets of n observations of variables with mean 0. Let us call the first list of observations $\mathbf{a} = (a_1, ..., a_n)$ and the second $\mathbf{b} = (b_1, ..., b_n)$.

Definition 1 (Covariance)**.** *Let us define the covariance of* $\mathbf{a} \in \mathbb{R}^n$ *and* $\mathbf{b} \in \mathbb{R}^n$ *as*

$$\sigma_{\mathbf{ab}} := \frac{1}{n} \sum_{i=1}^{n} a_i b_i = \frac{1}{n} \mathbf{a} \cdot \mathbf{b}^T.$$

From the definition it is obvious that the covariance is symmetric, $\sigma_{\mathbf{ab}} = \sigma_{\mathbf{ba}}$. In the special case $\mathbf{a} = \mathbf{b}$ the covariance $\sigma_{\mathbf{aa}}$ is called *variance* $\sigma^2_{\mathbf{a}}$.

Definition 2 (Covariance Matrix)**.** *Generalizing to* m *variables* $\mathbf{X} = (\mathbf{x_1}, ..., \mathbf{x_m})$*, each having been observed* n *times, gives us the covariance matrix.*

$$\mathbf{C_X} := \begin{pmatrix} \sigma_{\mathbf{x_1 x_1}} & \cdots & \sigma_{\mathbf{x_1 x_m}} \\ \vdots & \ddots & \vdots \\ \sigma_{\mathbf{x_m x_1}} & \cdots & \sigma_{\mathbf{x_m x_m}} \end{pmatrix} = \frac{1}{n} \mathbf{X} \mathbf{X}^T$$

The covariance matrix is a symmetric $m \times m$ matrix.

3.2 Diagonalizable Matrix

Definition 3 (Diagonalizable Matrix)**.** *A square matrix* $\mathbf{A}$ *is called diagonalizable, if there exists an invertable matrix* $\mathbf{P}$ *and a diagonal matrix* $\mathbf{D}$ *such that* $\mathbf{A} = \mathbf{P D P}^{-1}$.

Definition 4 (Symmetric matrix)**.** *A square matrix* $\mathbf{A}$ *is called symmetric, if* $\mathbf{A}^T = \mathbf{A}$.

Theorem 1. *Every symmetric matrix is diagonalizable.*

This is the main theorem we need in order to derive PCA. The proof of this theorem requires some preparation, which we will do now.

Definition 5 (Eigenvalues and Eigenvectors)**.** *Let* $\mathbf{A}$ *be a real* $m \times m$ *matrix.* $\lambda \in \mathbb{C}$ *is called a eigenvalue with eigenvector* $\mathbf{v} \in \mathbb{C}^m \setminus \{\mathbf{0}\}$ *if*

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}. \tag{3.1}$$

Lemma 1. *Every square* $m \times m$ *matrix has* m *(not necessarily unique) eigenvalues.*

Proof. We can rewrite equation 3.1 as

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{v} = \mathbf{0}$$

This allows us to interpret $(\mathbf{A} - \lambda\mathbf{I})$ as a function, which takes vectors $\mathbf{v} \in \mathbb{C}^m$. For λ to be a eigenvalue of $\mathbf{A}$ with eigenvector $\mathbf{v}$ it has to satisfy $\mathbf{v} \in \ker(\mathbf{A} - \lambda\mathbf{I})$ and $\mathbf{v} \neq 0$. From this we gather that all λ with $\ker(\mathbf{A} - \lambda\mathbf{I}) \neq \{\mathbf{0}\}$ are eigenvalues. We know this holds if and only if $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$. The determinant is a polynomial of degree m which can be expressed in the form $(\lambda - \lambda_1)...(\lambda - \lambda_m)$ with $\lambda_1, ..., \lambda_m \in \mathbb{C}$. These $\lambda_1, ..., \lambda_m$ are the m eigenvalues we wanted to find. □

Lemma 2. *A symmetric matrix has real eigenvalues.*

Proof. Let $\bar{\cdot}$ denote the complex conjugate. Define a complex dot product

$$(\mathbf{u}, \mathbf{v}) := \sum_{i=1}^{m} u_i \bar{v_i}$$

This dot product has the following properties for all $\mathbf{A} \in \mathbb{C}^{m\times m}, \mathbf{u}, \mathbf{v} \in \mathbb{C}^m, \lambda \in \mathbb{C}$

- $(\mathbf{A}\mathbf{u}, \mathbf{v}) = (\mathbf{u}, \mathbf{A}^T\mathbf{v})$,
- $(\lambda\mathbf{u}, \mathbf{v}) = \lambda(\mathbf{u}, \mathbf{v})$,
- $(\mathbf{u}, \lambda\mathbf{v}) = \bar{\lambda}(\mathbf{u}, \mathbf{v})$
- $(\mathbf{u}, \mathbf{u}) = 0 \iff \mathbf{u} = 0$

Let $\mathbf{A}$ be a symmetric matrix with eigenvalue $\lambda \in \mathbb{C}$.

For all $\mathbf{u} \in \mathbb{C}^m$ we have

$$\lambda(\mathbf{u}, \mathbf{u}) = (\lambda\mathbf{u}, \mathbf{u}) = (\mathbf{A}\mathbf{u}, \mathbf{u}) = (\mathbf{u}, \mathbf{A}^T\mathbf{u}) = (\mathbf{u}, \mathbf{A}\mathbf{u}) = (\mathbf{u}, \lambda\mathbf{u}) = \bar{\lambda}(\mathbf{u}, \mathbf{u}).$$

For $\mathbf{u} \neq \mathbf{0}$ we get $\lambda = \bar{\lambda}$ and thus $\lambda \in \mathbb{R}$. □

Are the corresponding eigenvectors real? From the proof of lemma 1 we know that the eigenvector $\mathbf{v}$ of eigenvalue λ is in $\ker(\mathbf{A} - \lambda\mathbf{I})$. Both the matrix $\mathbf{A}$ and λ are real, so $\mathbf{v}$ must be in $\mathbb{R}^m$ as well.

Lemma 3. *The eigenvectors of a symmetric matrix with distinct eigenvalues are orthogonal.*

Proof. Let λ_1, λ_2 be two distinct eigenvalues with eigenvectors $\mathbf{v}_1, \mathbf{v}_2$ of the matrix $\mathbf{A}$.

$$\lambda_1 \mathbf{v}_1 \cdot \mathbf{v}_2 = (\lambda_1 \mathbf{v}_1)^T \mathbf{v}_2 = (\mathbf{A}\mathbf{v}_1)^T \mathbf{v}_2 = \mathbf{v}_1^T \mathbf{A}^T \mathbf{v}_2 = \\ \mathbf{v}_1^T \mathbf{A}\mathbf{v}_2 = \mathbf{v}_1^T (\lambda_2 \mathbf{v}_2) = \lambda_2 \mathbf{v}_1 \cdot \mathbf{v}_2$$

This shows $(\lambda_1 - \lambda_2)\mathbf{v}_1 \cdot \mathbf{v}_2 = 0$ and as λ_1 and λ_2 are distinct, $\mathbf{v}_1$ and $\mathbf{v}_2$ must be orthogonal. □

What if the eigenvalues of the matrix are not distinct? In the proof of lemma 1 we showed that every $\mathbf{v} \in \ker(\mathbf{A} - \lambda\mathbf{I}) \backslash \{\mathbf{0}\}$ is a eigenvector. If and only if $(\lambda - \lambda_i)$ appears $k \geq 2$ times in the determinant of $(\mathbf{A} - \lambda\mathbf{I})$ then $\mathbf{A}$ has a non unique eigenvalue λ_i. As $\dim(\ker(\mathbf{A} - \lambda_i\mathbf{I})) = k$ we can choose orthogonal eigenvectors.

Now we have everything we need to prove theorem 1.

Proof of Theorem 1. Let $\mathbf{A} \in \mathbb{R}^{m \times m}$ be a symmetric matrix. From lemma 1 we know that eigenvalues $\lambda_1, ..., \lambda_m$ with corresponding eigenvectors $\mathbf{v}_1, ..., \mathbf{v}_m$ exist.

Define the following matrices

$$\mathbf{D} := \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_m \end{pmatrix} \qquad \mathbf{V} := \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_m \end{pmatrix}$$

The definition of eigenvalues and eigenvectors gives us

$$\mathbf{A}\mathbf{V} = \begin{pmatrix} \mathbf{A}\mathbf{v}_1 & \cdots & \mathbf{A}\mathbf{v}_m \end{pmatrix} = \begin{pmatrix} \lambda_1 \mathbf{v}_1 & \cdots & \lambda_m \mathbf{v}_m \end{pmatrix} = \mathbf{V}\mathbf{D}. \tag{3.2}$$

From lemma 3 we know that the eigenvectors, and therefore the columns of $\mathbf{V}$, are orthogonal. It follows that $rank(\mathbf{V}) = m$ which gives us the existence of $\mathbf{V}^{-1}$.

Rearranging equation 3.2 now gives us $\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1}$ which is what we wanted to show.

This shows that $\mathbf{A}$ is diagonalizable. □

Lemma 4. *If the columns of matrix* $\mathbf{A}$ *are orthonormal, then* $\mathbf{A}^{-1} = \mathbf{A}^T$.

Proof. Let $(\mathbf{a}_i)_{i=1,\dots,m}$ be the columns of the matrix. The columns are orthogonal and normed, therefore

$$\forall i, j : \mathbf{a}_i^T \mathbf{a}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \implies \mathbf{A}^T \mathbf{A} = \mathbf{I}$$

This shows $\mathbf{A}^{-1} = \mathbf{A}^T$. □

4 Principal Component Analysis

Combining the concepts in section 3 we derive the ideas and implementation of PCA.

Assume we have gathered observations of different variables as part of an experiment. If we have n variables, each having been observed m times, we can create a $m \times n$ matrix of this data. The goal is to get more insight and find underlying patterns in the collected data. For $n = 2$ we could try to plot the data, with the first variable as the x-axis and the second as the y axis. An exemplary plot of some data can be seen in figure 4.1.

For larger values of n this gets increasingly difficult[1]. PCA tries to solve this problem by transforming the data in such a way that the most interesting features are in the first few axis of the transformed m dimensional space. This makes it easy to look at a low dimension representation of the data, without loosing much information.

An example of PCA being applied to the data from figure 4.1 can be seen in figure 4.2. In the top figure the normed data and the direction of the new axis (called principal components) in relation to the two original axis are shown. The bottom figure depicts the transformed data. One can see that the variance is maximal in the first principal component.

Now we derive how to compute PCA. First we formulate a goal and define some assumptions.

We assume that the most interesting features are those that have a large variance[2]. Our goal is to find a transformation into new coordinates such that:

- the variance in the each axis is as large as possible.
- the axis are all orthogonal to each other.
- the axis are sorted (descending) by the variance in the axis.

From this we gather that another assumption is, that the axis are orthogonal. Lastly we are only concerned with linear dependent features in the data. Some example cases in which PCA fails are shown in figure 4.3.

[1] For higher dimensionality we have to use some projection. Depending on the chosen projection the interpretation changes making it difficult to interpret the resulting image.

[2] This assumption can be false. For data where the noise has a larger variance than the feature we are trying to observe, PCA fails because this assumption is not met.

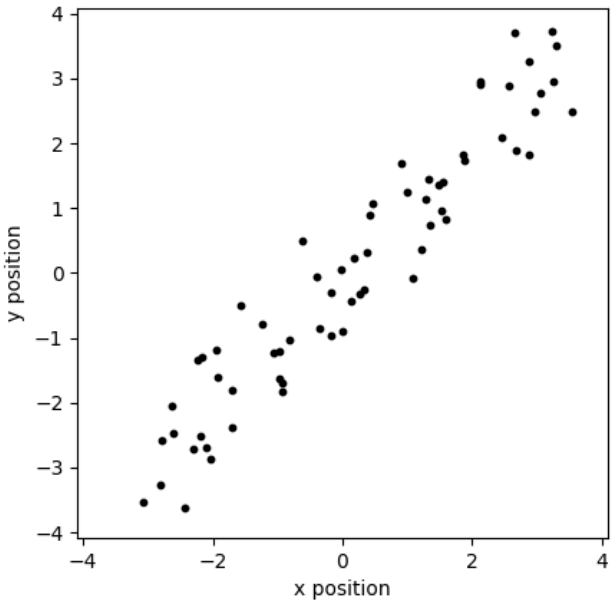


Figure 4.1: Randomly generated sample data. The data lies along a line with slope 1 and has mean $\mathbf{0}$.

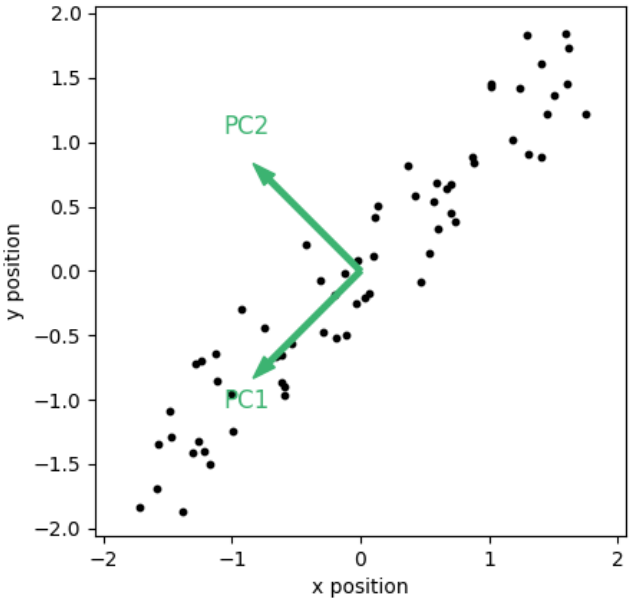
One way to achieve the goal is as follows:

1. Find the direction which maximizes the variance.
2. Save this direction as the next axis.
3. Determine the subspace that is orthogonal to all axis we found so far.
4. If the subspace is non-trivial start at the first step again.
5. If the subspace is trivial we have found all axis.

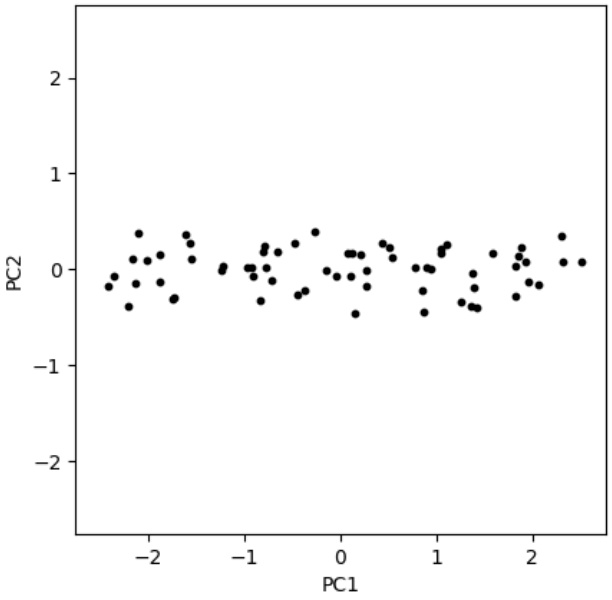
While this algorithm shows us what conceptually has to be done, we do not know how to compute the axis yet. We will now investigate this problem using the mathematical concepts from section 3. This will lead us to an algorithm in which all axis can be computed simultaneously.

Let $\mathbf{X} \in \mathbb{R}^{m \times n}$ be the data matrix. We want to find some orthonormal matrix $\mathbf{P}$ such that $\mathbf{Y} := \mathbf{PX}$ has a diagonal covariance matrix $\mathbf{C_Y}$.

$$\mathbf{C_Y} = \frac{1}{n}\mathbf{YY}^T = \frac{1}{n}(\mathbf{PX})(\mathbf{PX})^T = \frac{1}{n}\mathbf{PXX}^T\mathbf{P}^T = \\ = \mathbf{P}(\frac{1}{n}\mathbf{XX}^T)\mathbf{P}^T = \mathbf{PC_X}\mathbf{P}^T$$

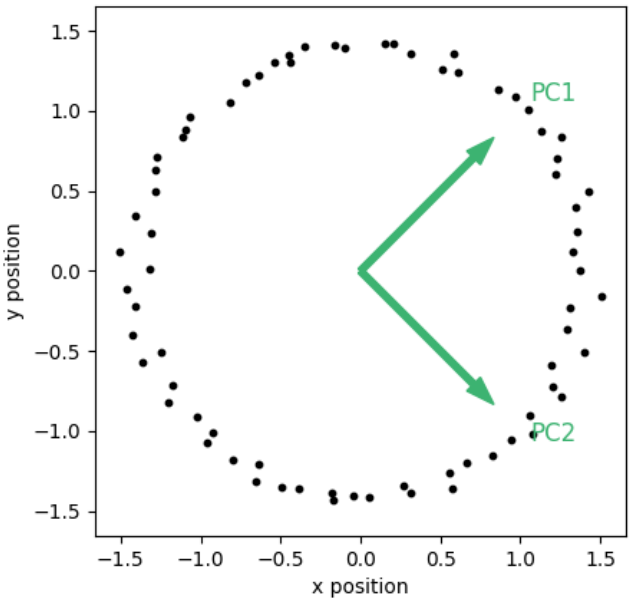


(a) Normed data (mean is zero and variance is one) and direction of the two principal components (PC1, PC2) in relation to the x and y position.

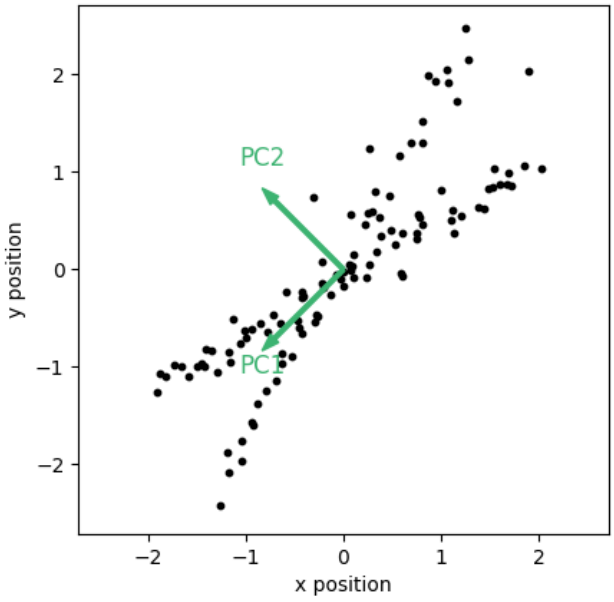


(b) The data after being transformed by PCA. The variance along the first principal component (PC1) axis is maximal, therefore the data is spread out most along this axis.

Figure 4.2: Example application of PCA.



(a) Clearly the relationship in this figure is non-linear. PCA can not describe circular dependencies, as shown in this data.



(b) The two main axis along which the data is aligned are not orthogonal to each other. PCA always outputs orthogonal principal components, therefore it fails in this example.

Figure 4.3: Examples in which some of the assumptions of PCA are not valid. The results are sub-optimal.

The covariance matrix $\mathbf{C_X}$ is symmetric and therefore has a decomposition into an orthogonal matrix of eigenvectors $\mathbf{V}$ and a diagonal matrix of eigenvalues $\mathbf{D}$. We choose $\mathbf{P} = \mathbf{V}^T$. From lemma 4 it follows that $\mathbf{V}^{-1} = \mathbf{V}^T$.

$$\begin{aligned}\mathbf{PC_XP}^T = \mathbf{P}(\mathbf{VDV}^{-1})\mathbf{P}^T = \mathbf{P}(\mathbf{VDV}^T)\mathbf{P}^T = \\ \mathbf{P}(\mathbf{P}^T\mathbf{DP})\mathbf{P}^T = (\mathbf{PP}^T)\mathbf{D}(\mathbf{PP}^T) = \\ (\mathbf{PP}^{-1})\mathbf{D}(\mathbf{PP}^{-1}) = \mathbf{D}\end{aligned}$$

In summary $\mathbf{Y}$ has a diagonal covariance matrix if we choose $\mathbf{Y} = \mathbf{V}^T\mathbf{X}$, where $\mathbf{V}$ is the matrix of eigenvectors of $\mathbf{C_X}$. The eigenvectors are the principal components and the eigenvalues are the variance in each new axis.

As pseudo code we get the program from algorithm 1 for calculating the PCA.

Algorithm 1 Principal Component Analysis

Require: matrix $X \in \mathbb{R}^{m \times n}$
Normalize each row in the matrix X
Calculate the covariance matrix C_X
Calculate the eigenvalues and eigenvectors of C_X
Sort the eigenvalues
Return sorted eigenvalues and corresponding eigenvectors

What happens if we skip the step in which we normalize each row in the matrix? A big variance is interpreted by the PCA algorithm as much information, thus the variance of the variables have an impact on how "important" the variable is deemed. As we do not want to prioritize certain variables we avoid this behavior by normalizing the data beforehand.

5 Methodology

For the main algorithm of this work from section 6 a few other methods will be used. As they are not the focus of this work we only give a short overview.

5.1 Fourier Transformation

Many applications are concerned with cyclic temporal data. Examples are sound waves or EEG data. This data can be represented as a function of amplitude over time. Most of the time we are not interested in the amplitude at a specific point in time, as a temporal shift would represent a very similar information. Such a shift is demonstrated in figure 5.1.

As the function of sound waves or EEG data is cyclic we might be interested in a decomposition into simple cyclic function, such as sin. We can then analyze the most prominent frequencies and their respective amplitudes. This gives a representation of the data, that can be easier to interpret.

We describe one approach to achieve this representation as a sum of sin functions, called Fast Fourier Transformation (FFT), in algorithm 2.

Figure 5.2 shows an example input and output of the FFT algorithm.

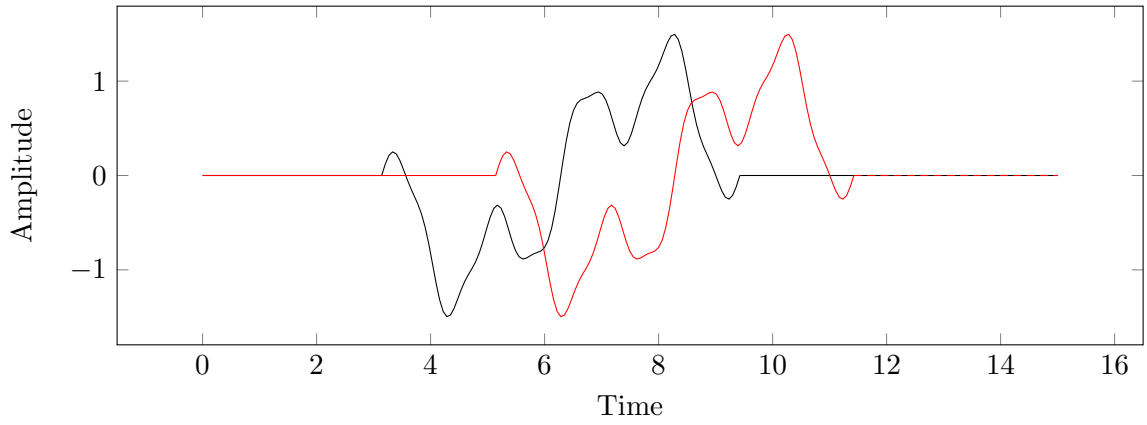


Figure 5.1: Two signals that only differ by a temporal shift.

Algorithm 2 FFT

function FFT(x, N, s)
  if $N = 1$ **then**
    $X_0 \leftarrow x_0$
  else
    $X_{0,\dots,N/2-1} \leftarrow FFT(x, N/2, 2s)$
    $X_{N/2,\dots,N-1} \leftarrow FFT(x + s, N/2, 2s)$
    for $k = 0$ to $N/2 - 1$ **do**
      $p \leftarrow X_k$
      $q \leftarrow exp(-2\pi i/Nk)X_{k+N/2}$
      $X_k \leftarrow p + q$
      $X_{k+N/2} \leftarrow p - q$
    end for
  end if
  return $X_{0,\dots,N-1}$
end function

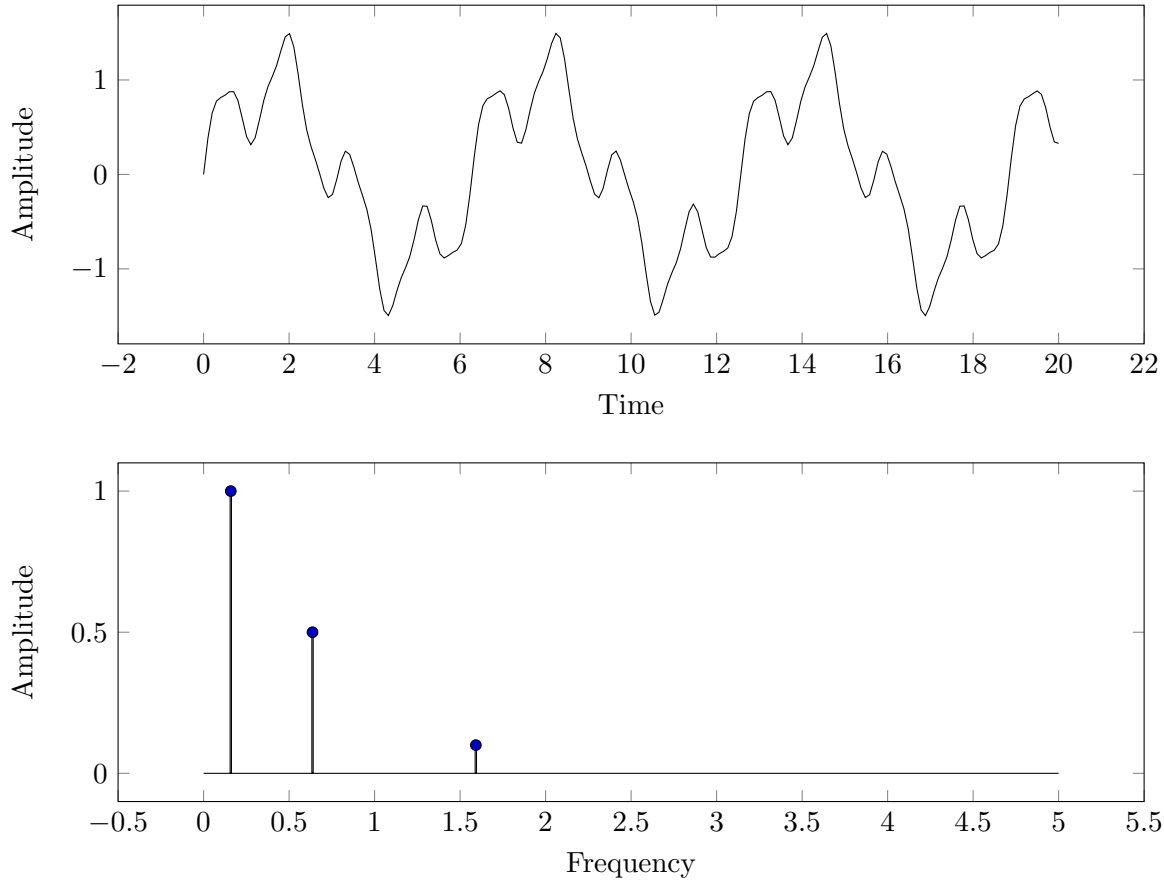


Figure 5.2: A signal composed of different sin functions and the corresponding FFT output.

5.2 Classification

In classification the objective is to find assignments between data points and categories. In our context we are interested in finding an assignment which closely matches some already categorized data. One simple approach to this problem is the k-nearest-neighbors algorithm.

For data that can be represented in $\mathbb{R}^m$ and l categories the pseudo code is shown in algorithm 3.

Algorithm 3 k Nearest Neighbors

Require: data points $(p_i)_{i\in\mathbb{N}} \in \mathbb{R}^m$, categories $(c_i)_{i\in\mathbb{N}} \in \{0, 1, ..., l\}$, point $x \in \mathbb{R}^m$, $k \in \mathbb{N}$
Calculate the distance between each point p_i and x
Take the k data points with the smallest distance to x
return the category that most of the k points are assigned

Figure 5.3 shows a graphical representation of the algorithm for points in $\mathbb{R}^2$ and $k = 10$.

This algorithm is slow for big datasets as for each point the distance to x has to be calculated. One possibility to reduce calculation time is to partition the space into smaller chunks. Then the loop only has to be over data points lying in the same or close chunks as the point we are interested in.

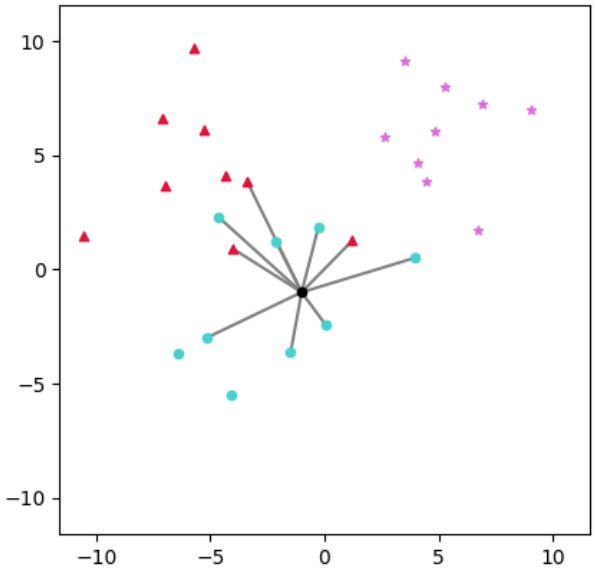


Figure 5.3: Data points in three categories are given. The black point is classified as a blue, circular point by the k-nearest-neighbors algorithm for $k = 10$.

6 Data and Algorithm

This chapter describes the CAP Sleep Database used and the algorithm implemented. The algorithm uses FFT, PCA and k-nearest-neighbor to estimate the sleep stage from a recorded EEG signal.

6.1 Data

We use the CAP Sleep Database[TPS+01][GAG+] which provides 16 recordings of patients without pathology. Two of these recordings have corrupted files and one did not record the data points used for this analysis and thus could not be analyzed. Table 6.1 shows some meta data of the recordings.

The given dataset describes multiple channels of EEG data recorded during sleep. We are interested in the channels recording activity in the central area, as these are the two channels most of the recordings share. Figure 6.1 shows where the sensors were placed. Other channels recorded activity in the frontal area, but are not used as mixing different channels poses problems in comparability between recordings.

The data was recorded in different frequencies. To reduce data size and standardize all channels we resample to 200 Hz.

Very low and very high frequencies often stem from other sources, such as the measurement equipment, breathing of the patients or measuring inaccuracies. To get rid of these noises we apply a low pass filter of 30 Hz and a high pass filter of 0.5 Hz.

File Name	Age	Gender	Length of Recording	Channel Name	Sampling Frequency
n1	37	F	577 minutes	C4-A1	512 Hz
n2	34	M	735 minutes	C4-A1	512 Hz
n3	35	F	551 minutes	C4-A1	512 Hz
n4	25	F	596 minutes	C4-A1	200 Hz
n5	35	F	524 minutes	C4-A1	512 Hz
n6	31	M	527 minutes	C3-A2	128 Hz
n7	31	M	492 minutes	C3-A2	128 Hz
n8	42	F	501 minutes	C3-A2	200 Hz
n9	31	M	532 minutes	C3-A2	128 Hz
n10	23	M	490 minutes	C4-A1	512 Hz
n11	28	F	527 minutes	C4-A1	512 Hz
n12	29	M	495 minutes	C3-A2	100 Hz
n16	41	F	513 minutes	C4-A1	100 Hz

Table 6.1: Meta data of the 13 recordings from the CAP Sleep Database.

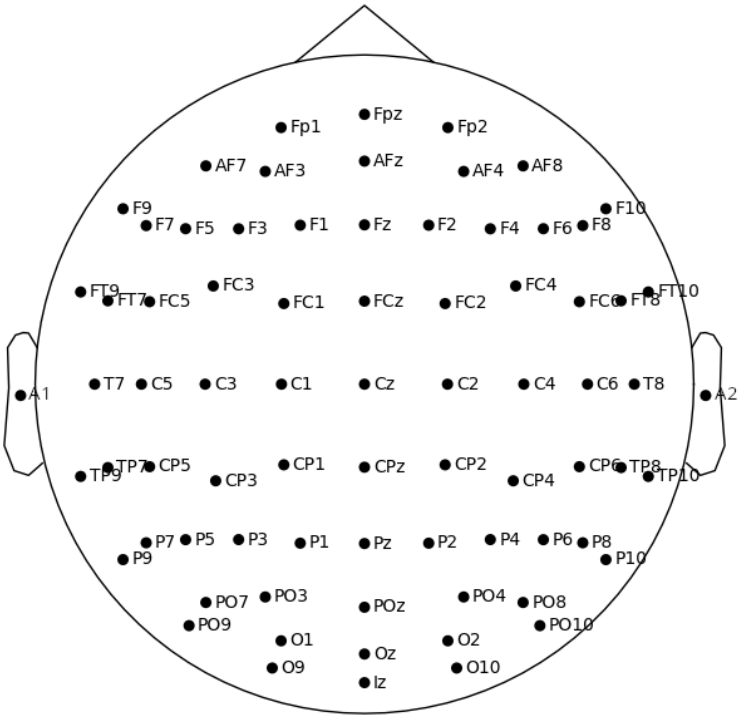


Figure 6.1: Scheme of sensor placement in recordings of EEG data.

6.2 Algorithm

We now describe the general structure of the algorithm from reading the data files to the clustering.

The EEG data is stored in European data format (EDF) files. They provide a standard for exchanging and storing medical time-series data.

First we split the recorded data into 30 second sections. For each of these sections we get the data as a function of amplitude over time. An example segment is shown in the upper figure of 6.2. We apply a Fourier transformation as described in section 5.1 and get a function of amplitude over frequency. This gives a frequency based representation of the data. As the sleep stages are characterized by different frequencies and amplitudes, as described in section 1.2, this will help us distinguishing them. The output of such an FFT can be seen in the lower figure of 6.2. The pseudo code for this processing is described in algorithm 4.

Algorithm 4 EEG data processing

```
for each patient do
    Read EEG data from file
    Apply a 30 Hz low pass and a 0.5 Hz high pass filter
    Resample with frequency 200 Hz
    Get one of the channels C4-A1 or C3-A2 (in this order)
    Section out the part of data where the patient is asleep
    for each 30 second segment of the data do
        Do a Fourier transformation (FFT)
    end for
    Save all the outputs from the FFT to the patients file
end for
```

We now have high dimensional data points, as each 30 second segment is represented by the amplitudes for each of the distinct frequencies output by the FFT. Before we can start searching for cluster in the data we want to find a lower dimensional representation. This is were PCA can be used. From the output of algorithm 5 we get principal components, which reveal the frequencies were the most variance is shown. Under our assumptions this variation is caused by the different sleep stages each recording goes through. We show a visual representation of the data in the first three principal components in figure 6.3.

If we have a new recording and want to know the sleep stages we have to follow similar steps. First the recording has to be split into 30 second segments. These have to be transformed by a FFT. The output can be converted to the PCA basis by multiplying with the matrix of principal components. Lastly we can get a estimate of the sleep stage by using the k-nearest-neighbor algorithm. The pseudo code is shown in algorithm 6.

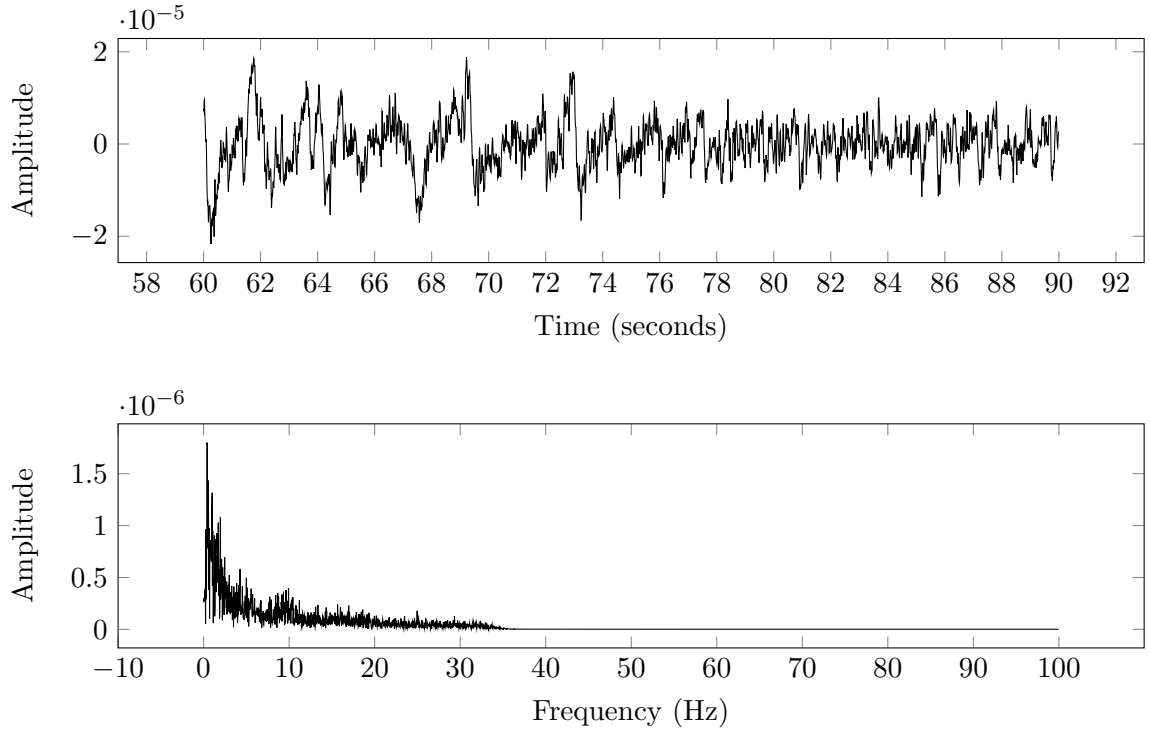


Figure 6.2: A 30 second segment of the data. The top shows the recorded signal of the EEG. The bottom shows the output of the Fourier transformation.

Algorithm 5 Apply PCA to the EEG data

for each patient **do**
    Read FFT output from patients file
    Read the sleep stages from another file
end for
Concatenate FFT data of all patients
Concatenate sleep stages of all patients
Do PCA on the combined FFT data
Transform data according to the principal components from the PCA
Visualize the data in the first three axis, with the color corresponding to the sleep stage

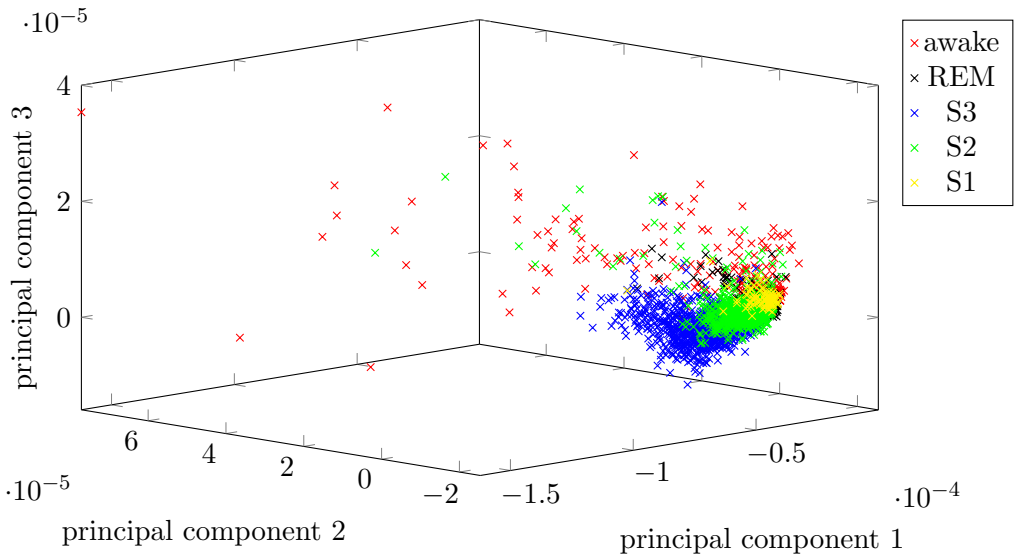


Figure 6.3: Three dimensional representation of the data. Each 30 second segment is represented by a dot in the color corresponding to the sleep stage this segment is assigned to.

Algorithm 6 Get estimate for sleep stage

Require: EEG recording of sleep
  for each 30s segment **do**
    Apply FFT to the segment
    Multiply FFT output by principal component matrix
    Reduce dimensions
    Apply k-nearest-neighbor
    Save the result of k-nearest-neighbor
  end for
  return results of k-nearest-neighbor

7 Results

In this chapter we will analyze with which accuracy the algorithm from chapter 6 classifies sleep stages. We will have a look into which frequencies differ most between the stages and whether the principal components of the PCA use the same frequencies.

7.1 Compression of Data

From section 4 we know that it is possible to use PCA to reduce dimensions. We want to know how much information is still present in the compressed data. This can be calculated by testing how many dimensions it takes to retain a certain amount of variance in the data.

To do this we find the first index for which the sum of eigenvalues up to this index from the PCA is bigger than a certain percentage of the sum of all eigenvalues. The output for a few percentages are shown in table 7.1. As only few dimensions are needed to preserve 90% of variance, or in other words information, the PCA can successfully be used as a way to reduce dimensions.

percentage of variance	number of dimensions needed
50%	1 out of 3000
80%	4 out of 3000
90%	24 out of 3000
95%	55 out of 3000

Table 7.1: Percentage of variance retained after reducing the dimension.

7.2 Testing the Accuracy

To test the accuracy of the algorithm from section 6.2 we divide the dataset into two partitions, one for training the PCA and one for validating the results. The training data contains 10 of the 13 patients and the validation data contains the other 3 patients.

First the training data is used to calculate the principal components of the PCA. Then the validation data is transformed according to the principal components. Both the training and validation data are reduced to 24 dimensions, as we know that means only about 10% of variance is lost. Lastly we use the k-nearest-neighbor algorithm to estimate the sleep stage of the validation data and compare it to the true classification. This is algorithm 6.

When randomly guessing the sleep stage we expect an accuracy of 20%, as there are five different stages. For better comparison of the accuracy achieved, we run an algorithm, that does not use PCA, but otherwise follow the same logic. In comparison to algorithm 6 the algorithm 7 does not reduce dimensions, as there is no clear way of doing this without using PCA.

Algorithm 7 Get estimate for sleep stage without PCA

Require: EEG recording of sleep
  for each 30s segment **do**
    Do FFT on the segment
    Do k nearest neighbor
    Save the result of k nearest neighbor
  end for
  return results of k nearest neighbor

We test different values for k of the k-nearest-neighbor algorithm. Table 7.2 gives a overview of the accuracy achieved with and without PCA and different values for k.

Using $k = 10$ yields the best outcome in this limited validation set. Out of TODO 30 second segments the algorithm estimated the correct one TODO times, which corresponds to a success rate of 68.37%. Table 7.3 shows the outcome in more detail.

k	accuracy with PCA	accuracy without PCA
1	59.67%	50.80%
5	65.57%	56.13%
10	68.37%	57.73%
15	67.74%	56.60%
20	68.28%	57.42%
25	68.28%	56.39%
30	68.18%	56.89%
35	68.28%	56.73%

Table 7.2: Overview of accuracy achieved with different values for k of k-nearest-neighbor.

	S3	S2	S1	REM	awake
S3	467	35	1	1	28
S2	339	1026	18	101	38
S1	0	2	4	2	21
REM	4	258	64	577	44
awake	3	33	8	11	102

	S3	S2	S1	REM	awake
S3	290	5	1	0	9
S2	508	803	4	11	29
S1	0	2	16	1	26
REM	13	518	66	626	74
awake	2	26	8	54	95

Table 7.3: The left table shows output for the algorithm using PCA, while the right table is from the algorithm that does not use PCA. Both tables show the number of sleep stages assigned to each stage, with the true value corresponding to the column and the estimate from the algorithm corresponding to the row. For both tables $k = 20$.

7.3 Rhythm Analysis

There are four rhythms observed in the EEG[Gan97, chapter 11]. These rhythms differ in the frequency and are characterized as

- Alpha: 8 - 12 Hz
- Beta: 18 - 30 Hz
- Theta: 4 - 7 Hz
- Delta: $<$ 4 Hz

It is easy to analyze the presence of these rhythms when looking at the fourier transformed data, as shown in figure 7.1.

We can now see if the output of the PCA gives importance to these rhythms. In figure 7.2 the sum of the absolute values of the first 24 principal components is shown. We chose 24 as from table 7.1 we gather that 90% of the variance is preserved. The most emphasis is given on the values from the theta rhythm. This was expected as the amplitudes in the FFT output are the highest in this region and we did not normalize the data before doing PCA.

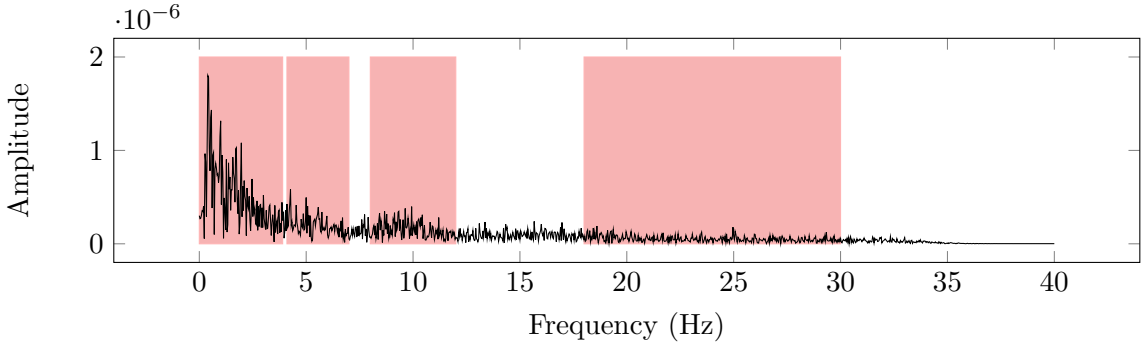


Figure 7.1: Fourier transformed data with rhythms highlighted.

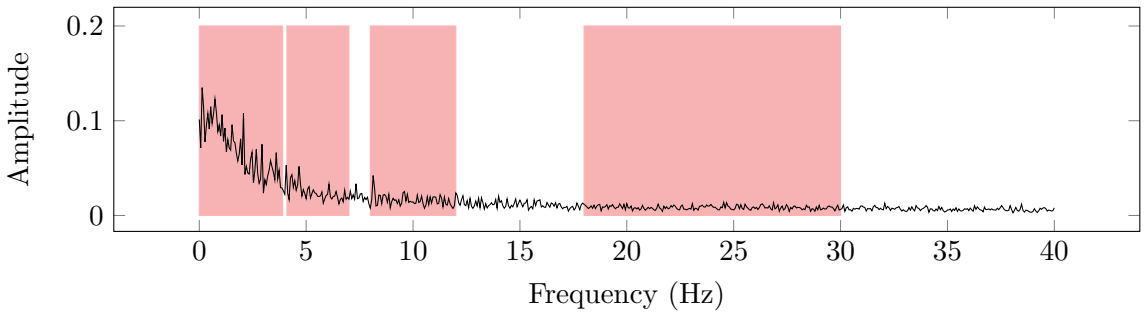


Figure 7.2: Sum of the absolute values of the first 24 principal components.

8 Discussion

8.1 Limitations of the Algorithm

With an accuracy of 68.37% the algorithm proposed gives better results than a comparative one not using PCA. From table 7.3 it is apparent that the algorithm can not easily distinguish stages S2 and S3. Furthermore stage S1 is underrepresented in the estimation given by the algorithm.

From the results of chapter 7 it is clear, that the algorithm proposed is does not have enough accuracy to be used in a medical setting. The reasons for this could be multitude. For one there is only limited training data. Using a bigger sample of sleep recordings could improve the results achieved. Secondly a different classification algorithm could be tested. From figure 6.3 it looks like a linear separation could be possible between the sleep stages S3, S2 and S1. Awake and REM on the other hand appear more difficult so separate from the rest.

8.2 Future Work

We propose some improvements for possible future work. One way to get more data is to use more than one channel per sleep recording. There are two options, which we propose. Either all available channels are treated as separate sleep recordings. Comparing different channels might be difficult, but all channels share similar aspects in the different sleep stages. The disadvantage is that the data points would no longer be independent of each other, which reduces the data quality.

The other option is to construct a higher dimensional input from multiple channels recorded from the same sleep recording. This requires all recordings to have all these channels, as the input dimension of PCA must be fixed. One work around could be to replace the missing data with a fixed value, such as zero.

One could also put more thought into the input of the PCA. Compacting the data by calculating the presence of the signal in each of the rhythms from section 7.3 could be an option.

The data points depend on each other, as not every change from one sleep stage to another is equally likely. For example typically the recording start with the patient still awake and then proceeding to sleep stages S1, S2 and S3. The REM stage often only appears later. The information of the last sleep stage could be used as input in estimating the next sleep stage.

Lastly the classification algorithm used could be changed. Using k-nearest-neighbors works best if there are about equally many data points in each category. This is not the case in the CAP Sleep Database, as not all sleep stages appear equally often.

List of Figures

List of Tables

List of Algorithms

Bibliography

[BKN17] Reza Boostani, Foroozan Karimzadeh, and Mohammad Nami. A comparative review on sleep stage classification methods in patients and healthy individuals. *Computer Methods and Programs in Biomedicine*, 140:77 – 91, 2017. Cited by: 212.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[CMP94] Charles K Chui, Laura Montefusco, and Luigia Puccio. *Wavelets: theory, algorithms, and applications*, volume 5. Academic press, 1994.

[GAG+] Goldberger, L. Amaral A., L. Glass, J. Hausdorff, P. C. Ivanov, R. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, and H. E. Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals.

[Gan97] William F. Ganong. *Review of medical physiology*. Appleton & Lange, Stamford, Conn, 18. ed edition, 1997.

[GMwHL+18] Mohammad M Ghassemi, Benjamin E Moody, Li wei H Lehman, Christopher Song, Qiao Li, Haoqi Sun, Roger G Mark, M Brandon Westover, and Gari D Clifford. You snooze, you win: the physionet/computing in cardiology challenge 2018. *2018 Computing in Cardiology Conference (CinC)*, pages 1–4, 2018.

[Hot33] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[HPPB18] Matthew Howe-Patterson, Bahareh Pourbabaee, and Frederic Benard. Automated detection of sleep arousals from polysomnography data using a dense convolutional neural network. In *2018 Computing in Cardiology Conference (CinC)*, volume 45, pages 1–4. IEEE, 2018.

[JC16] I. T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. *Royal Society*, 374(2065), 2016.

[MST+23] Claus Metzner, Achim Schilling, Maximilian Traxdorf, Holger Schulze, Konstantin Tziridis, and Patrick Krauss. Extracting continuous sleep depth from eeg data without machine learning. *Neurobiology of Sleep and Circadian Rhythms*, 14, 2023. All Open Access, Gold Open Access, Green Open Access.

[NK11] Ganesh R Naik and Dinesh K Kumar. An overview of independent component analysis and its applications. *Informatica*, 35(1), 2011.

[Pea01] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[Put15] Arcady A. Putilov. Principal component analysis of the eeg spectrum can provide yes-or-no criteria for demarcation of boundaries between nrem sleep stages. *Sleep Science*, 8(1):16–23, 2015.

[Shl14] Jonathon Shlens. A tutorial on principal component analysis. 2014.

[SSM97] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer, 1997.

[TDSL00] J.B. Tenenbaum, V. De Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319 – 2323, 2000. Cited by: 10812.

[TPS+01] M G Terzano, L Parrino, A Sherieri, R Chervin, S Chokroverty, C Guilleminault, M Hirshkowitz, M Mahowald, H Moldofsky, A Rosa, R Thomas, and A Walters. Atlas, rules, and recording techniques for the scoring of cyclic alternating pattern (cap) in human sleep., 2001.

[TRdFI21] Alexandra-Maria Tăuţan, Alessandro C. Rossi, Ruben de Francisco, and Bogdan Ionescu. Dimensionality reduction for eeg-based sleep stage detection: comparison of autoencoders, principal component analysis and factor analysis. *Biomedical Engineering / Biomedizinische Technik*, 66(2):125–136, 2021.