Tensor Decompositions of EEG Signals for Transfer Learning Applications

Emma Fallenius

Linda Karlsson

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6172
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

In this report, tensor decomposition methods of EEG signals have been evaluated for the purpose of transfer learning. The aim has been to address the *person-to-person Brain-Computer Interface (BCI) calibration problem* by transferring training data between sessions, which can shorten calibration times, extend the amount of training data, and enable using data from simulated environments in real world applications. For this, the datasets Alex MI (binary motor imagery) and SA Driving (drowsiness detection during simulated driving) have been analyzed. Tensor decompositions were performed unsupervised in two pipelines, with aim of capturing universal structures relevant to BCI tasks.

For the first pipeline, two decompositions (Canonical Polyadic and Tucker) were computed to compare similarity between sessions. From that, a subset of sessions were selected that during classification, were aimed to outperform random selection and training with the full training database. In the first pipeline, a new similarity measure was designed, which included *weighting* of the factor matrix in the mode of interest. We consider this a more representative measure of how similar two sessions are, compared to simply studying the unweighted factor matrices, which was done in previous literature. For the second pipeline, one tensor decomposition (Tucker) was used for feature extraction and similarity comparison between sessions. The aim was the same as for pipeline one, with the addition of investigating the properties of tensor decompositions as *features*. The results show that unsupervised tensor decompositions can extract structures of varying relevance to a classification problem but did not result in superior performance when used as features. With this knowledge, we propose extending tensor decompositions to supervised and/or nonlinear ones. Additionally, the proposed session selection methods showed potential in classification, but no significant conclusions could be drawn of their superiority compared to random selection or training with the full database. Additionally, the classifiers had a large variation in performance between sessions, making them far from applicable for a BCI in a real-world environment today.

Acknowledgments

We would like to say a big thank you to our supervisors Carolina Bergeling and Bo Bernhardsson. They have always encouraged our curiosity and reasoning when digging into the relatively unexplored field of tensors and EEG. Carolina, your ability to see the full picture of a problem and quickly understand our most unelaborated thoughts has been invaluable. Bo, even though you are a very busy man, you have constantly taken time to engage in our problems and involve us in the world of academics. We feel like both of you have believed in us from the very first email, which we consider a huge privilege.

To Frida Heskebeck, Martin Gemborn Nilsson and Pex Tufvesson, thank you for willingly including us in your EEG community and always being open to questions and great discussions. We wish you good luck with your super inspiring work.

Next, we want to thank our examiner Pontus Giselsson for valuable optimization inputs early on in the project. Another person that helped us getting started and probably saved us weeks of time was Anders Nilsson. Thank you for your patience and pedagogical computer support. Regarding computational resources, the support from SNIC is also gratefully acknowledged.

Furthermore, a great thanks to the department of Automatic Control at LTH for the warm welcome. The coffee breaks at 10:00 and 15:00 (set in stone) have been very, very appreciated. And, to our fellow students in the master's thesis room, working alongside you has lightened up many otherwise gray days.

Lastly, we want to thank our partners and family for their unconditional support throughout our five years at LTH.

Contents

Dictionary

Alex MI	Alex Motor Imagery (dataset)
ALS	Alternating Least Squares (algorithm)
BCI	Brain Computer Interface
core tensor	Component of the Tucker decomposition, translates how the factor matrices should be scaled and combined to obtain the approximated tensor
CP	Canonical Polyadic (decomposition)
DI	Drowsiness Index
EEG	Electroencephalogram
factor matrix	Component of a tensor decomposition, answering to a specific mode
feature	characteristic believed to be of importance for a classification task
HOOI	Higher Order Orthogonal Iteration (algorithm)
latent ~	non-observable ~, inferred through a mathematical model
mode	Specific dimension of a tensor
order	Dimension of a tensor
PSD	Power Spectral Density, a signal's power divided into frequency components
RT	Reaction Time
SA Driving	Sustained-Attention Driving task (dataset)
session	A set of trials, belonging to a subject
subject	A person whose EEG signals have been recorded
SVC	Support Vector Classifier
SVR	Support Vector Regressor
rbf	Radial basis function
target session	Previously never seen test session used to evaluate performance
TFR	Time-Frequency Representation, describes the power in a signal by time- and frequency components
trial	EEG recording when performing a task
tensor	Multidimensional array

1 Introduction

The purpose of this report was to address the person-to-person calibration problem in Brain-Computer Interfaces. This has been done by transferring training data between sessions of EEG data, with aim of selecting a subset of sessions similar to a target session. To compare similarity between sessions and for feature extraction, tensor decomposition methods have been applied. This chapter will further explain the background and motivate the relevance of this problem.

EEG Signals and Brain-Computer Interfaces

This first section serves as a brief introduction of the brain, EEG-signals and Brain-Computer Interfaces. The section is based on Chapter 2 **The Electroencephalogram - A Brief Background** in the book *Bioelectrical signal processing in cardiac and neurological applications* by Leif Sörnmo, 2005. For the interested reader, we refer to the book for a more detailed description.

The brain is a complex organ that controls an enormous amount of processes and activities in our body, such as thought, memory, motor skills, vision, breathing, hunger etcetera. All these processes and activities are interactions between billions of neurons (nerve cells), which send/receive chemical and electrical signals inside the brain and to/from the rest of the body via the spinal cord. Different parts of the brain are responsible for separate tasks. One example is the motor cortex, which is located on top of the head, and responsible of planning, control, and execution of voluntary movements. Another example is the visual cortex, located at the back of the head, which receives, integrates, and processes visual information.

The electrical activity of the brain can be measured with *electroencephalography* (EEG). During an EEG recording, electrodes are placed on top of the subject's scalp in a non-invasive manner. The recorded electrogram will be a representation of the macroscopic activity of the surface layer of the brain. Advantages with the technique, apart from being non-invasive, is its mobility and its millisecond-range temporal resolution. The big disadvantage is that it is hard to comprehend the signals and filter out a specific event occurring inside the brain, as the macroscopic behavior that is recorded represents a large complexity of processes.

During the EEG measurement, spatial behavior of the signal can be analyzed and mapped to areas of interest in the brain. A standardized system has been developed for the placement of scalp electrodes during EEG recordings. It is denoted the *10–20 system*, where the name refers to that the distance between adjacent electrodes usually is 10% or 20% of the total front–back or right–left distance of the skull. Depending on how many electrodes one wishes to use, there are different versions and extensions of the 10-20 system, called *montages*. An example of a 64-channel montage placement according to the modified[1] international 10-20 system can be seen in Figure 1.1.

[1] Including extensions/exclusions of electrodes compared to the original 10-20 system.

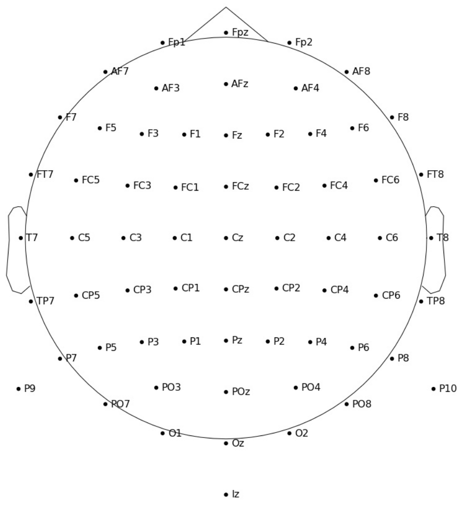


Figure 1.1 64-channel montage placement according to the modified international 10-20 system. Channel names are printed next to the electrodes.

Today, several types of *EEG caps* have been developed to simplify electrode placement and data gathering. The caps are designed with pre-determined montages and can be adapted to different head sizes. In Figure 1.2, an example of an EEG cap can be seen. This particular one is 24-channel EasyCap[2] of size 58.

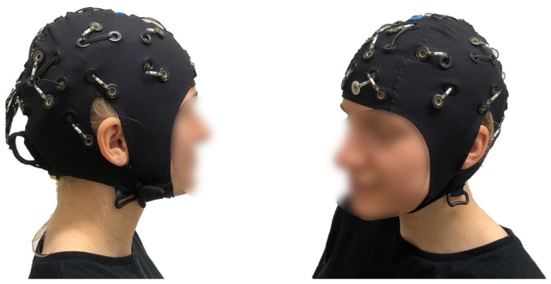


Figure 1.2 An EasyCap[2] EEG cap of size 58 from two different perspectives. The cap has a 24-channel montage, with the purpose of simplifying electrode placement in a consequent manner between subjects.

[2] https://www.easycap.de/

The measured electrical activity of all channels can be kept in the time domain, or transformed to the frequency/time-frequency domains. In the frequency domain, EEG signals have been sorted into different frequency bands which power increases during different states of brain activity. A description of the frequency bands can be seen in Table 1.1.

Table 1.1 Name, range and description of EEG frequency bands [Sörnmo, 2005].

Name	**Frequency Range**	**Description (increase in power)**
Delta	<4 Hz	Deep sleep.
Theta	4-7 Hz	Drowsiness and certain stages of sleep.
Alpha	8-13 Hz	Relaxation and eyes closing.
Beta	14-30 Hz	Awake stages, for example active focus and thinking.
Gamma	>30 Hz	Active information processing and cognitive incline.

An EEG recording can be used as a diagnostic tool, for example to detect epilepsy or sleeping disorders. Another application is *Brain-Computer Interfaces* (BCI). A BCI is a way for an external device to bypass the normal communication pathway between the brain and body, the spinal cord, and instead directly communicate with the brain. For a person with an injury/disease that affects the ability to for example move or talk, this can be a useful way to move a robotic limb or communicate with the outside world only by thinking.

A BCI is not only of interest for people with physical disabilities, but can be extended to areas of applicability in healthy subjects' every day lives. Controlling an external device only by thinking creates many possibilities, like turning on a light without moving, writing a text without typing or playing video games without controllers. Besides this, a person's mental state can be monitored, which for example could be a useful way to measure the drowsiness level during nighttime driving.

Worth keeping in mind when working with BCIs are the ethical, social, and legal challenges that follow from BCI integration in society. For example, accessing someone's thoughts could be considered a great question of privacy intrusion. This report will not handle the ethical, social or legal aspects of BCIs. Focus will only lie on the technicalities of EEG signals in transfer learning, applied in two specific BCI tasks. We encourage the reader to look into details[3] of the above mentioned challenges for a full perspective of what role BCIs can and/or should

[3] For example, see [Drew, 2019].

play in our future society.

EEG signals from a specific subject are typically recorded during a *session*. In each session, a task of interest is repeatedly performed, where every such iteration is called a *trial*. Trials can be denoted with specific labels, and classifiers can be trained to predict the label of new, unlabeled trials. These definitions will be further explained in Chapter 4 **Datasets**.

A key challenge when working with EEG data is the inter- and intra-variation between subjects. Two sessions of EEG recordings from two subjects who are performing the same task can be very different. Furthermore, not only do different subjects differ in EEG recordings, but separate session recordings from the same subject can also vary to an extensive amount. Historically, this has made it necessary to calibrate classifiers for each session individually, recognized as the *person-to-person BCI calibration problem*. This calibration process is not only time consuming, but also limits the amount of useful data during training of the classifier. Additionally, this constrains all classes to be performable during the calibration phase. For example, this means that the previously mentioned BCI application of measuring the drowsiness level during nighttime driving would be very dangerous to perform in reality. On top of that, not only does the inter- and intra-variation between subjects result in the need for individual calibration, but it also makes it a challenge to understand the properties of the EEG signals and hence, the brain itself.

Tensors

Tensors are multidimensional arrays that provide a useful way to store and work with high-dimensional data. By using tensors, no dimension needs to be excluded, and all spatial structures can be examined at the same time. EEG data can for example be formulated as the three dimensional structure (session × channel × frequency) that can be visualized as a cube. Similarly to matrices, tensors can be factorized or *decomposed*, and from the decomposition components, again be recreated. Instead of recreating the complete, original tensor, an approximation can be made by including fewer components of the decomposition. The purpose of this is usually to highlight structures of importance in the data and exclude noise. In Chapter 2 **Tensor Theory**, tensors and tensor decompositions will be described in detail.

Main Objective

Throughout this project, the main objective has been to evaluate tensor representations and decomposition methods for EEG signals in the field of *transfer learning*. In this report, transfer learning is defined as the action of transferring data between sessions. From this perspective, what is mainly of interest is how well tensors can capture and/or describe "universal" structures of EEG signals. By universal structures, we mean characteristics that are mutual between different subjects and sessions and prominent for a certain task (for instance moving a hand, or being drowsy during driving). By removing the session dependency of EEG signals, it would be possible to address the person-to-person BCI calibration problem. This would shorten the calibration time for every new session significantly, extend the amount of training data which could be used for a certain classification task, and make it possible to transfer data from simulated environments to real applications.

To accomplish this, classifiers/regressors have been trained only with transferred EEG data from other sessions than the *target session* (the previously never seen test session, used to evaluate performance). In this project, tensor representations of EEG signals have been applied in two different approaches; as a way of solely comparing similarity between sessions, or for feature extraction in combination with similarity comparison between sessions. The main questions have been the following:

- Can tensor decompositions, performed unsupervised, capture structures relevant and useful to a classification problem?
- Are the captured structures efficient for similarity comparison between sessions in a transfer learning setting?

Tools and Implementation

In this project, only publicly available open source datasets have been used, further described in Chapter 4 **Datasets**. All code has been written in Python, and can be found in a GitLab repository[4]. The main Python packages/libraries used were MNE [Gramfort et al., 2013] (for EEG data analysis and visualization), TensorLy [Kossaifi et al., 2019] (for tensor operations and decompositions) and scikit-learn [Pedregosa et al., 2011] (for machine learning classification/regression). Additionally, due to the extensive amount of CPU hours required to evaluate the different approaches, several Python scripts have been run on the computer cluster Kebnekaise, which is a part of Swedish National Infrastructure for Computing (SNIC).

[4] https://gitlab.control.lth.se/exjobb-bci/emmalinda

Report Outline

The report will start with relevant theory on tensors and machine learning in Chapters 2 **Tensor Theory** and 3 **Machine Learning Theory**. Thereafter, the two datasets Alex MI and SA Drivers, will be presented in Chapter 4 **Datasets**. Chapter 5 **Tensor Decompositions for Session Selection** will include the first pipeline; using tensor decompositions solely for session selection. Method, results and discussions will be described and analyzed, including our self-designed weighted similarity measure. Chapter 6 **Tensor Decompositions for Feature Extraction and Session Selection** will present the second pipeline in a similar manner, with the extension of using the decompositions for session selection *and* feature extraction. Lastly, a general discussion, future work, and conclusions will be summarized in Chapter 7 **Conclusions and Future Work**.

2

Tensor Theory

The following sections present tensors; their definition, operations, and two of the most common decomposition methods: the Canonical Polyadic (CP) decomposition and the Tucker decomposition. For an explanation of the mathematical operations, e.g. the outer product, see **Basic Operations** in the Appendix.

2.1 Definition and Operations

Tensors are multidimensional arrays. Their order answers to their number of dimensions, and every dimension is known as a *mode*. For example, vectors are tensors of order one, matrices are tensors of order two, and a third order tensor can be visualized as a cube, see Figure 2.1. Vectors will be denoted with bold lower case letters, matrices with bold upper case letters, and higher order tensors with bold, upper case, calligraphic letters as seen in Figure 2.1.

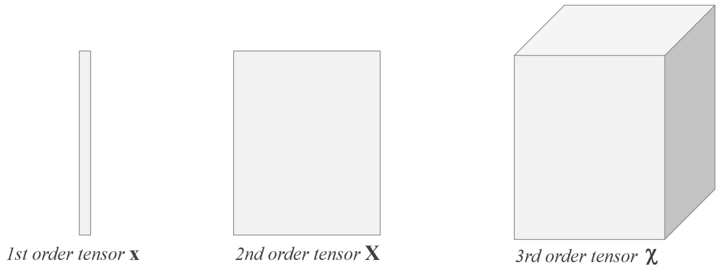

1st order tensor $\mathbf{x}$
2nd order tensor $\mathbf{X}$
3rd order tensor $\boldsymbol{\mathcal{X}}$

Figure 2.1 Visualization of a first, second, and third order tensor respectively.

More generally, a tensor $\boldsymbol{\mathcal{X}} \in \mathbf{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is said to be of order N and is by definition an element of the tensor product of N vector spaces. Each vector space $n = 1, \dots, N$ has a dimension of $I_n \times 1$. If the tensor in turn can be written as the outer product of N first order tensors (vectors), it is said to be of rank one:

$$\boldsymbol{\mathcal{X}} = \boldsymbol{a}^{(1)} \circ \boldsymbol{a}^{(2)} \circ \cdots \circ \boldsymbol{a}^{(N)} \tag{2.1}$$

where $\circ$ answers to the vector outer product. Expanding this thought, a tensor $\boldsymbol{\mathcal{X}} \in \mathbf{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is said to be of rank R^* if it can be expressed as a linear combination of R^* outer products of N first order tensors:

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R^*} \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} \tag{2.2}$$

Consequently, a non-zero vector is a tensor of rank one [Kolda and Bader, 2009]. The rank R^* is thus, the smallest integer such that (2.2) holds.

Similarly to matrices, higher order tensors can be multiplied, but the notation of the operations is less intuitive. Taking the matrix multiplication $\boldsymbol{AB}$ of two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, the number of columns in $\boldsymbol{A}$ must coincide with the number of rows in $\boldsymbol{B}$. If a matrix $\boldsymbol{A} \in \mathbf{R}^{J \times I_n}$ is to be multiplied with a tensor $\boldsymbol{\mathcal{X}} \in \mathbf{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the corresponding multiplication would evaluate the product along the mutual dimension I_n. To do so, one has to *unfold* the tensor; fixing all dimensions but one and concatenating the entries along this mode such that one obtains a matrix.

For example, a third order tensor $\boldsymbol{\mathcal{X}} \in \mathbf{R}^{I_1 \times I_2 \times I_3}$ can be unfolded in three ways, one for every dimension. Fixing the tensor in all directions but I_1, one can see the tensor as a $I_2 \times I_3$ matrix where every element corresponds to a vector, often referred to as a *fiber*, of length I_1. Figure 2.2 visualizes the fibers of a third order tensor:

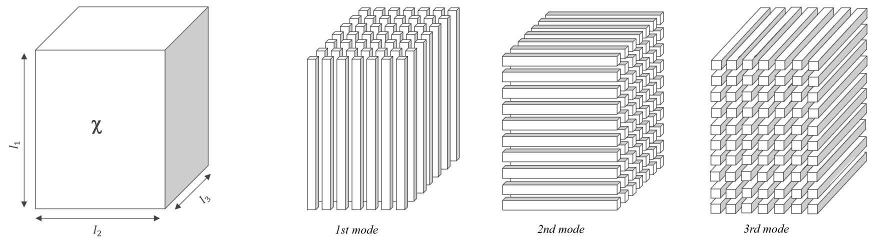


Figure 2.2 The fibers along the three modes of a third order tensor $\boldsymbol{\mathcal{X}} \in \mathbf{R}^{I_1 \times I_2 \times I_3}$.

Unfolding with respect to the first mode I_1, the resulting matrix has a dimension of $I_1 \times (I_2 I_3)$, where $I_2 I_3$ vectors of length I_1 have been rearranged as columns in a matrix. If unfolding along the second mode, I_1 and I_3 are fixed and the resulting tensor answers to a matrix of shape $I_2 \times (I_1 I_3)$, see Figure 2.3 for a visualization. Returning to a general tensor $\boldsymbol{\mathcal{X}}$ of order N, the n-mode unfolded tensor answers to a $I_n \times \Pi_{i=1,i\neq n}^{N} I_i$ matrix and is denoted $\boldsymbol{X}_{(n)}$ [Kolda and Bader, 2009]. Fixing a direction n is thus a way of enabling a multiplication between a matrix and a tensor. The n-mode product of a tensor is one method that utilizes this.

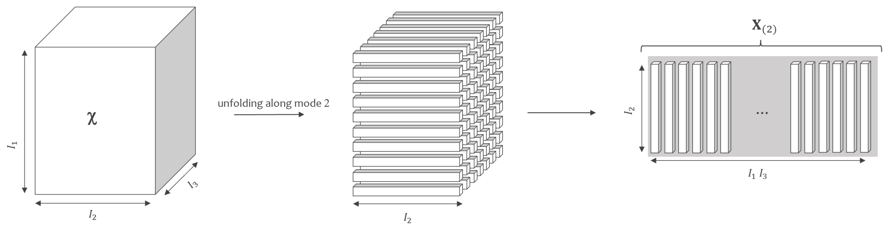


Figure 2.3 Unfolding along the second mode of a third order tensor $\boldsymbol{\mathcal{X}}$, resulting in the matrix $\boldsymbol{X}_{(2)}$ of shape $I_2 \times I_1 I_3$.

The n-mode product can be evaluated with respect to any tensors as long as the factors share an index. Taking a matrix $\boldsymbol{A} \in \mathbf{R}^{J \times I_n}$ and the tensor $\boldsymbol{\mathcal{X}} \in \mathbf{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the n-mode product answers to multiplying every n-mode fiber of the tensor with $\boldsymbol{A}$. The $\boldsymbol{i}$th element of the product becomes:

$$(\boldsymbol{\mathcal{X}} \times_n \boldsymbol{A})_{i_1 \,\dots\, i_{n-1} j i_{n+1} \,\dots\, i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \,\dots\, i_N} a_{j i_N} \tag{2.3}$$

and the resulting tensor $\boldsymbol{\mathcal{X}} \times_n \boldsymbol{A}$ is of size $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$. Unfolding this tensor along the nth mode, the resulting matrix $(\boldsymbol{\mathcal{X}} \times_n \boldsymbol{A})_{(n)}$ answers to the matrix multiplication for the corresponding mode $\boldsymbol{A}\boldsymbol{X}_{(n)}$ [Kolda and Bader, 2009]. Note that the n-mode product does not project the tensor $\boldsymbol{\mathcal{X}}$ to a lower dimensional space, but uses the unfolding to enable a multiplication along the desired dimension.

2.2 Tensor Decompositions

The Canonical Polyadic Decomposition

In accordance with the previous section, the rank R^* of a tensor is defined as the number of terms needed in a linear combination of rank-one tensors to describe the tensor exactly. These linear combinations answer to a *decomposition* of the original tensor, and is described by a linear combination of lower dimensional structures. In literature, the decomposition that describes the original tensor as a sum of R^* rank-one tensors is referred to as the *canonical polyadic* (CP) decomposition[1].

A CP decomposition of a tensor $\boldsymbol{\mathcal{X}} \in \mathbf{R}^{I_1 \times I_2 \times \cdots \times I_N}$ has the form:

$$\begin{aligned} \boldsymbol{\mathcal{X}} &= [\![\boldsymbol{A}^{(1)^*}, \boldsymbol{A}^{(2)^*}, \dots, \boldsymbol{A}^{(N)^*}]\!] \\ &= \sum_{r=1}^{R^*} \boldsymbol{a}_r^{(1)^*} \circ \boldsymbol{a}_r^{(2)^*} \circ \dots \circ \boldsymbol{a}_r^{(N)^*} \end{aligned} \tag{2.4}$$

and the $\boldsymbol{i}$th element answers to:

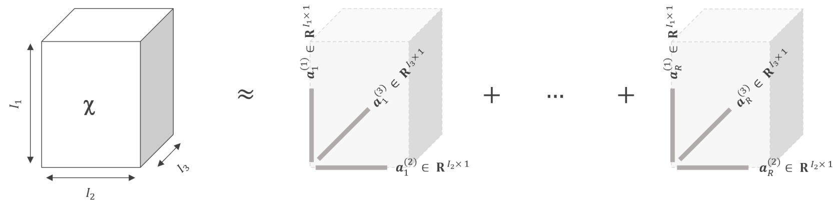
$$x_{i_1 \,\dots\, i_N} = \sum_{r=1}^{R^*} a_{i_1 r}^{(1)^*} a_{i_2 r}^{(2)^*} \cdots a_{i_N r}^{(N)^*} \tag{2.5}$$

where $\boldsymbol{a}_r^{(n)^*}$ in (2.4) denotes the rth column of the *factor matrix* $\boldsymbol{A}^{(n)^*} \in \mathbf{R}^{I_n \times R^*}$ answering to mode n, and $\circ$ the outer product [Kolda and Bader, 2009]. Note that there will be N factor matrices, each one corresponding to one of the N modes. Also note that * symbolizes the components of the exact decomposition and

[1] Also referred to as the CANDECOMP/PARAFAC decomposition [Kolda and Bader, 2009]

thus, does not answer to an approximation of $\boldsymbol{\mathcal{X}}$.

Expanding this argument, one can *approximate* the tensor by including fewer terms of one rank tensors. Figure 2.4 shows a third order tensor $\boldsymbol{\mathcal{X}}$ approximated by the R rank one tensors in the CP decomposition of order R. Note that the approximation should not be made by simply excluding the rank one tensors $R < r \leq R^*$ from the full decomposition of rank R^*. This, as there is no guarantee that the rank one tensors $R < r \leq R^*$ are of less importance than the $r \leq R$ ones. Instead, a new decomposition should be computed for each approximation, which could conclude in a completely different set of rank one tensors. [Kolda and Bader, 2009]



I_1
χ
I_3
I_2
$\approx$
$\boldsymbol{a}_1^{(1)} \in \mathbf{R}^{I_1 \times 1}$
$\boldsymbol{a}_1^{(3)} \in \mathbf{R}^{I_3 \times 1}$
$\boldsymbol{a}_1^{(2)} \in \mathbf{R}^{I_2 \times 1}$
$+ \quad \cdots \quad +$
$\boldsymbol{a}_R^{(1)} \in \mathbf{R}^{I_1 \times 1}$
$\boldsymbol{a}_R^{(3)} \in \mathbf{R}^{I_3 \times 1}$
$\boldsymbol{a}_R^{(2)} \in \mathbf{R}^{I_2 \times 1}$

Figure 2.4 A third order tensor $\boldsymbol{\mathcal{X}} \in \mathbf{R}^{I_1 \times I_2 \times I_3}$ approximated by the R rank one tensors in the CP decomposition of order R. The vectors $\boldsymbol{a}_r^{(n)}$ for $r = 1, \ldots, R$ correspond to the columns of the factor matrix $\boldsymbol{A}^{(n)}$ of the nth mode.

Computing a CP Decomposition

As mentioned, when the summation over r is taken for an R smaller than the rank R^* of the tensor, the decomposition becomes an approximation. For a fixed number of terms $R < R^*$, the computation is presented as an optimization problem; minimize the error between the true and approximated tensor. Here, the least squares formulation is often chosen as a loss function:

$$\begin{aligned}\mathcal{L}(\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, ..., \boldsymbol{A}^{(N)}) &= \|\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}\|^2 \\ &= \|\boldsymbol{\mathcal{X}} - [\![\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]\|^2 \\ &= \|\boldsymbol{\mathcal{X}} - \sum_{r=1}^{R} \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \ldots \circ \boldsymbol{a}_r^{(N)}\|^2 \end{aligned} \tag{2.6}$$

where the norm is the squared tensor norm of the error tensor, see **The Tensor Norm** in the Appendix. The optimization problem is then formulated as:

$$\min_{\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}} \mathcal{L}(\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, ..., \boldsymbol{A}^{(N)}) \tag{2.7}$$

Note that the decomposition is not unique; a factor $\boldsymbol{A}^{(n)}$ can equally be set to $\lambda \boldsymbol{A}^{(n)}$ as long as another factor matrix is scaled by the inverse $1/\lambda$. This is an acknowledged downside of the CP decomposition and will be discussed later.

Alternating Least Squares To compute factor matrices $\boldsymbol{A}^{(n)}$ for all $n = 1, \ldots, N$, one can iteratively solve the problem for every mode of the tensor by fixing the others as constant. Consequently, all factor matrices but one are fixed and the optimization is carried out for one matrix at the time. (2.6) can thus be formulated as a least squares problem with respect to matrices, for which the solution is given by the Moore-Penrose pseudoinverse (see Appendix):

$$\boldsymbol{A}^{(n)} = \mathbf{X}_{(n)} [(\boldsymbol{A}^{(N)} \odot \cdots \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \cdots \boldsymbol{A}^{(1)})^T]^\dagger \tag{2.8}$$

where $\mathbf{X}_{(n)}$ is the mode n unfolded tensor, and $\odot$ the Khatri-Rao product (see Appendix) [Kolda and Bader, 2009].

An issue with the alternating least squares (ALS) algorithm, is it not being guaranteed to converge to a stationary point. To account for this, one can apply gradient descent when updating the factor matrices (instead of the Moore-Penrose pseudoinverse). This has proven to be both more efficient and result in a better accuracy of the factor matrices [Acar et al., 2011][Lin et al., 2018]. Additionally, to find a unique solution, one needs to attack the scaling indeterminacy of the decomposition. One way of doing so is by benefiting one solution by subjecting the optimization problem to a constraint, discriminating a subset of the possible solutions. For example, one can restrict the optimization to benefit non-negative decompositions, or by adding a regularization term [Kolda and Bader, 2009].

The Tucker decomposition

Another commonly applied decomposition is the Tucker decomposition, which can be viewed as a higher order singular value decomposition (SVD). A tensor $\boldsymbol{\mathcal{X}} \in \mathbf{R}^{I_1 \times \cdots \times I_N}$ is then factorized by N factor matrices $\boldsymbol{A}^{(n)} \in \mathbf{R}^{I_n \times R_n}$ and a *core tensor* $\boldsymbol{\mathcal{G}} \in \mathbf{R}^{R_1 \times \cdots \times R_N}$ [Kolda, 2006]:

$$\begin{aligned}\boldsymbol{\mathcal{X}} &= \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{A}^{(1)} \times_2 \cdots \times_N \boldsymbol{A}^{(N)} \\ &= \sum_{r_1,\ldots,r_N}^{R_1,\ldots,R_N} g_{r_1 \ldots r_N} \boldsymbol{a}_{r_1}^{(1)} \circ \boldsymbol{a}_{r_2}^{(2)} \circ \cdots \circ \boldsymbol{a}_{r_N}^{(N)} \\ &= [\![\boldsymbol{\mathcal{G}}; \boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}]\!]\end{aligned} \tag{2.9}$$

where the $\boldsymbol{i}$th element of the resulting tensor is given by:

$$x_{i_1 \ldots i_N} = \sum_{r_1,\ldots,r_N}^{R_1,\ldots,R_N} g_{r_1 \ldots r_N} a_{i_1 r_1}^{(1)} a_{i_2 r_2}^{(2)} \cdots a_{i_N r_N}^{(N)} \tag{2.10}$$

Similarly to the singular value decomposition of a matrix, the factor matrix $\boldsymbol{A}^{(n)}$ can be viewed as the principal components of mode n, where the elements of the core tensor $\boldsymbol{\mathcal{G}}$ measures the interaction between components. A visualization of the Tucker decomposition for a third order tensor can be seen in Figure 2.5.

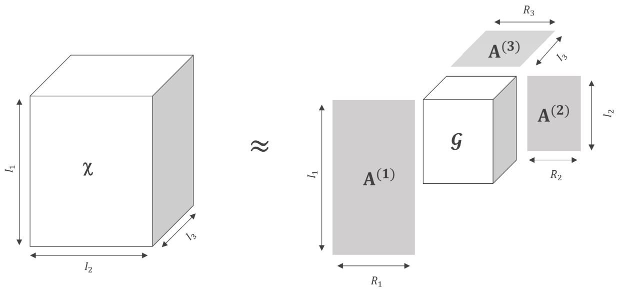


Figure 2.5 A third order tensor $\boldsymbol{\mathcal{X}} \in \mathbf{R}^{I_1 \times I_2 \times I_3}$ approximated by the Tucker decomposition of rank (R_1, R_2, R_3).

The shape of $\boldsymbol{\mathcal{G}}$ depends on the shape of the factor matrices, which illuminates the flexibility of the Tucker decomposition; the factor matrices $\boldsymbol{A}^{(n)} \in \mathbf{R}^{I_n \times R_n}$ must not have the same number of columns (which they must in the CP decomposition). Consequently, one can compress the core tensor differently in different modes. In the case of $R_n = I_n$ for all N factors, the core tensor will have the same shape as the original tensor $\boldsymbol{\mathcal{X}}$. The decomposition can then be seen as a change of basis, where the elements of $\boldsymbol{\mathcal{G}}$ answer to the scaling in a new space spanned by the factor matrices $\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}$. If $R_n < I_n$ for some mode(s) though, the core tensor will be smaller and answer to a lower dimensional representation of $\boldsymbol{\mathcal{X}}$ [Kolda and Bader, 2009].

Computing the Tucker Decomposition

Assuming one is to approximate a tensor $\boldsymbol{\mathcal{X}} \in \mathbf{R}^{I_1 \times \cdots \times I_N}$ by a Tucker decomposition, the optimization problem can be formulated as:

$$\min_{\boldsymbol{\mathcal{G}}, \boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}} \| \boldsymbol{\mathcal{X}} - [\![\boldsymbol{\mathcal{G}};\ \boldsymbol{A}^{(1)},\ \ldots, \boldsymbol{A}^{(N)}]\!] \|^2 \tag{2.11}$$

for the factor matrices $\boldsymbol{A}^{(n)} \in \mathbf{R}^{I_n \times R_n}$ and the core tensor $\boldsymbol{\mathcal{G}}$ with shape $(R_1 \times \cdots \times R_N)$. Remembering that the Tucker decomposition could be seen as a higher order SVD, one could define the factor matrices from the eigenvectors of the corresponding mode [Kolda and Bader, 2009]. The most common approaches today though, settle with computing an orthonormal basis since it is computationally cheaper. De Lathauwer, De Moor, and Vandewalle proposed in 2000 one such method; Higher Order Orthogonal Iteration [De Lathauwer et al., 2000], later referred to as HOOI. The optimization problem answers to the one presented in (2.11), but with an orthogonality constraint on the factor matrices:

$$\begin{aligned} &\min_{\boldsymbol{\mathcal{G}}, \boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}} \|\boldsymbol{\mathcal{X}} - [\![\boldsymbol{\mathcal{G}};\ \boldsymbol{A}^{(1)},\ \ldots, \boldsymbol{A}^{(N)}]\!]\|^2 \\ &\quad \text{subject to } \boldsymbol{\mathcal{G}} \in \mathbf{R}^{R_1 \times \cdots \times R_N} \\ &\qquad\qquad \boldsymbol{A}^{(n)} \in \mathbf{R}^{I_n \times R_n} \text{ columnwise orthogonal for all } n = 1, \ldots N \end{aligned} \tag{2.12}$$

where the loss function to minimize is:

$$\mathcal{L}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{G}}, \boldsymbol{A}^{(1)},\ \ldots, \boldsymbol{A}^{(N)}) = \|\boldsymbol{\mathcal{X}} - [\![\boldsymbol{\mathcal{G}};\ \boldsymbol{A}^{(1)},\ \ldots, \boldsymbol{A}^{(N)}]\!]\|^2 \tag{2.13}$$

Since the Tucker decomposition answers to the n-mode product between a core tensor $\boldsymbol{\mathcal{G}}$ and the factor matrices $\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}$, the norm in (2.12) can be written in vectorized form [Kolda and Bader, 2009]:

$$\|\text{vec}(\boldsymbol{\mathcal{X}}) - (\boldsymbol{A}^{(N)} \otimes \boldsymbol{A}^{(N-1)} \otimes \cdots \otimes \boldsymbol{A}^{(1)})\text{vec}(\boldsymbol{\mathcal{G}})\|^2 \tag{2.14}$$

where the vectorization of tensors are done with respect to the first mode (i.e. the columns). By definition, the norm will only equal zero if the vector answers to the zero vector. The optimal solution thereby answers to:

$$\text{vec}(\boldsymbol{\mathcal{G}}) = (\boldsymbol{A}^{(N)^T} \otimes \boldsymbol{A}^{(N-1)^T} \otimes \cdots \otimes \boldsymbol{A}^{(1)^T})\text{vec}(\boldsymbol{\mathcal{X}}) \tag{2.15}$$

where it has been used that the Kronecker product of (semi) orthogonal[2] matrices is (semi) orthogonal (see 8.1 *Basic Operations: The Kronecker Product* in the Appendix). Rewriting the expression in (2.15), the core tensor $\boldsymbol{\mathcal{G}}$ can be written as a Tucker decomposition of the tensor $\boldsymbol{\mathcal{X}}$ and the inverse factor matrices $\boldsymbol{A}^{(n)^T}$ [Kolda, 2006]:

$$\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{X}} \times_1 \boldsymbol{A}^{(1)^T} \times_2 \cdots \times_N \boldsymbol{A}^{(N)^T} \tag{2.16}$$

Using the expression above, the norm to minimize (see 2.13) can be simplified:

[2] A *semi* orthogonal matrix does not need to be a square matrix, but must only fulfill $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$ or $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}$.

$$
\begin{aligned}
&\|\boldsymbol{\mathcal{X}} - [\![\boldsymbol{\mathcal{G}}; \boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}]\!]\|^2 \\
&\quad = \|\boldsymbol{\mathcal{X}}\|^2 - 2\langle \boldsymbol{\mathcal{X}}, [\![\boldsymbol{\mathcal{G}}; \boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}]\!]\rangle + \|[\![\boldsymbol{\mathcal{G}}; \boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}]\!]\|^2 \\
&\quad = \|\boldsymbol{\mathcal{X}}\|^2 - 2\langle \boldsymbol{\mathcal{X}} \times_1 \boldsymbol{A}^{(1)^T} \times_2 \cdots \times_N \boldsymbol{A}^{(N)^T}, \boldsymbol{\mathcal{G}}\rangle \\
&\qquad\qquad + \langle \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{A}^{(1)} \times_2 \cdots \times_N \boldsymbol{A}^{(N)}, \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{A}^{(1)} \times_2 \cdots \times_N \boldsymbol{A}^{(N)}\rangle \\
&\quad = \|\boldsymbol{\mathcal{X}}\|^2 - 2\langle \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{G}}\rangle \\
&\qquad\qquad + \langle \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{A}^{(1)} \times_1 \boldsymbol{A}^{(1)^T} \times_2 \cdots \times_N \boldsymbol{A}^{(N)} \times_N \boldsymbol{A}^{(N)^T}, \boldsymbol{\mathcal{G}}\rangle \\
&\quad = \|\boldsymbol{\mathcal{X}}\|^2 - 2\langle \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{G}}\rangle \\
&\qquad\qquad + \langle \boldsymbol{\mathcal{G}} \times_1 (\boldsymbol{A}^{(1)^T} \boldsymbol{A}^{(1)}) \times_2 \cdots \times_N (\boldsymbol{A}^{(N)^T} \boldsymbol{A}^{(N)}), \boldsymbol{\mathcal{G}}\rangle \\
&\quad = \|\boldsymbol{\mathcal{X}}\|^2 - 2\langle \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{G}}\rangle + \langle \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{G}}\rangle \\
&\quad = \|\boldsymbol{\mathcal{X}}\|^2 - \|\boldsymbol{\mathcal{G}}\|^2 \\
&\quad = \|\boldsymbol{\mathcal{X}}\|^2 - \|\boldsymbol{\mathcal{X}} \times_1 \boldsymbol{A}^{(1)^T} \times_2 \cdots \times_N \boldsymbol{A}^{(N)^T}\|^2
\end{aligned}
\tag{2.17}
$$

where the fourth equality utilizes $\boldsymbol{\mathcal{X}} \times_n \boldsymbol{A} \times_n \boldsymbol{B} = \boldsymbol{\mathcal{X}} \times_n (\boldsymbol{BA})$, and the fifth equality that the factor matrices are (semi) orthogonal. Since the tensor $\boldsymbol{\mathcal{X}}$ is constant, the original problem in (2.12) can be reformulated as N subproblems, each maximizing the norm of the (optimal) core tensor $\boldsymbol{\mathcal{G}}$:

$$
\begin{aligned}
&\max_{\boldsymbol{A}^{(n)}} \|\boldsymbol{\mathcal{X}} \times_1 \boldsymbol{A}^{(1)^T} \times_2 \cdots \times_N \boldsymbol{A}^{(N)^T}\|^2 \\
&\quad \text{subject to } \boldsymbol{A}^{(n)} \in \mathbf{R}^{I_n \times R_n} \text{ columnwise orthogonal}
\end{aligned}
\tag{2.18}
$$

The solution can be addressed by using an alternating least squares approach. Assuming the nth factor matrix as constant, the expression within the norm can be written as:

$$
\|\boldsymbol{A}^{(n)^T} \boldsymbol{W}\|^2 \text{ with } \boldsymbol{W} = \boldsymbol{X}_{(n)}(\boldsymbol{A}^{(N)} \otimes \cdots \otimes \boldsymbol{A}^{(n-1)} \otimes \boldsymbol{A}^{(n+1)} \otimes \cdots \otimes \boldsymbol{A}^{(1)}) \tag{2.19}
$$

for which the optimal $\boldsymbol{A}^{(n)}$ can be computed as the R_n most significant left singular vectors of $\boldsymbol{W}$. Since the singular vectors by definition are orthogonal to each other, the orthogonality constraint on the factor matrices will be fulfilled.

There are two issues when computing the Tucker decomposition. Firstly, this ALS approach is not guaranteed to converge to a stationary point, but only to a solution where the loss in (2.13) ceases to decrease (as for the CP decomposition). Secondly, the Tucker decomposition itself is not unique [Kolda and Bader, 2009]. The algorithm is thereby likely to produce inconsistent results because of the various decompositions it can converge to, and its non-guarantee of converging to any of them. Similarly as for the CP-decomposition, the non-uniqueness can be addressed by restricting the loss function to a subset of solutions.

Non-negative Decompositions

A non-negative decomposition yields an approximated tensor with non-negative elements *only*. With the purpose of modeling positive data, these decompositions can be a desirable option. This is partly as they impose a constraint on the decomposition (shrinking the number of possible solutions), but also as they preserve the sign of data that (possibly after processing) is only positive. For naturally positive data, non-negative decompositions thereby allow for a more intuitive interpretation of the result [Kolda and Bader, 2009].

The restriction on preserving the positivity translates to computing non-negative factors of the decomposition. For the CP and Tucker decompositions, this translates to positive factor matrices, where the Tucker decomposition requires the core tensor to be non-negative as well. For the past twenty years, various approaches for computing non-negative CP and Tucker decompositions have been presented, where some are based on Lee and Seung's work on non-negative matrix factorization [Kolda, 2006]. The sections below account for two of those methods; non-negative CP decomposition, and non-negative Tucker decomposition. For a better understanding, Lee and Seung's conclusions on non-negative matrix factorization will firstly be presented.

Non-Negative Matrix Factorization A matrix $\boldsymbol{X}$ of shape $m \times n$ represents a second order tensor and can thus be factorized. Assuming the decomposition is defined from two factor matrices $\boldsymbol{A} \in \mathbf{R}^{m \times R}$ and $\boldsymbol{B} \in \mathbf{R}^{R \times n}$, an element of $\boldsymbol{X}$ answers to:

$$\begin{aligned} \boldsymbol{X} &= \boldsymbol{AB} \\ x_{ij} &= \sum_{r=1}^{R} a_{ir} b_{rj} \end{aligned} \tag{2.20}$$

which equals the element of a CP decomposition for the factor matrices $\boldsymbol{A}$ and $\boldsymbol{B}^T$, recall (2.5). A matrix factorization can thereby be seen as a second order CP decomposition. If one is to calculate a lower dimensional representation, the optimization can be formulated as a least squares problem:

$$\min_{\boldsymbol{A},\boldsymbol{B}} \|\boldsymbol{X} - \boldsymbol{AB}\|_F^2$$

To calculate a non-negative decomposition, Lee and Seung have added a constraint to this formulation, requiring all elements of the factor matrices to be non-negative:

$$\begin{aligned}&\min_{\boldsymbol{A},\boldsymbol{B}} \|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{B}\|_F^2\\ &\text{subject to } \boldsymbol{A} \geq 0,\ \boldsymbol{B} \geq 0\end{aligned} \tag{2.21}$$

Denoting the loss function $\mathcal{L}$ as the quadratic norm of the error, it is only convex if one studies one factor at the time [Lee and Seung, 1999]. Consequently, one can use an alternating approach with a gradient based search to compute the factor matrices. Calculating the gradient with respect to every factor, one obtains:

$$\begin{aligned}\nabla_{\boldsymbol{A}}\mathcal{L} &= -2\boldsymbol{X}\boldsymbol{B}^{\boldsymbol{T}} + 2\boldsymbol{A}\boldsymbol{B}\boldsymbol{B}^T\\ \nabla_{\boldsymbol{B}}\mathcal{L} &= -2\boldsymbol{A}^T\boldsymbol{X} + 2\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{B}\end{aligned} \tag{2.22}$$

Applying gradient descent, the update rule becomes [Lars-Christer Böiers, 2010]:

$$\begin{aligned}\boldsymbol{A} &\leftarrow \boldsymbol{A} - \eta_{\boldsymbol{A}}\nabla_{\boldsymbol{A}}\mathcal{L}\\ \boldsymbol{B} &\leftarrow \boldsymbol{B} - \eta_{\boldsymbol{B}}\nabla_{\boldsymbol{B}}\mathcal{L}\end{aligned} \tag{2.23}$$

To apply the constraint of positive factor matrices, Lee and Seung propose restricting the algorithm to return only non-negative updates. They do so by choosing the learning rates $\eta_{\boldsymbol{A}}$ and $\eta_{\boldsymbol{B}}$ such that there is no subtraction:

$$\begin{aligned}\eta_{\boldsymbol{A}} &= \boldsymbol{A} \oslash (2\boldsymbol{A}\boldsymbol{B}\boldsymbol{B}^T)\\ \eta_{\boldsymbol{B}} &= \boldsymbol{B} \oslash (2\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{B})\end{aligned} \tag{2.24}$$

where $\oslash$ denotes the element-wise division. Inserting the expression to (2.23) one obtains a multiplicative update rule:

$$\begin{aligned}\boldsymbol{A} &\leftarrow \boldsymbol{A} * (\boldsymbol{X}\boldsymbol{B}^T) \oslash (\boldsymbol{A}\boldsymbol{B}\boldsymbol{B}^T)\\ \boldsymbol{B} &\leftarrow \boldsymbol{B} * (\boldsymbol{A}^T\boldsymbol{X}) \oslash (\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{B})\end{aligned} \tag{2.25}$$

where $*$ is the element-wise multiplication, referred to as the Hadamard product (for details, see 8.1 **Hadamard Product** in the Appendix)). The positivity of the factor matrices will be preserved as long as $\boldsymbol{A}$ and $\boldsymbol{B}$ are initialized positive, and $\boldsymbol{X}$ itself is non-negative. In their work, Lee and Seung prove that this approach leads to convergence of the factor matrices (in terms of the gradients in (2.22) being zero) [Lee and Seung, 1999].

Non-Negative CP Decomposition Similarly to the original ALS approach, the non-negative CP decomposition can be derived by iteratively solving the factor matrices. The difference in the approach lies in the constraint on the factor matrices:

$$\min_{\boldsymbol{A}^{(1)},\,\dots,\,\boldsymbol{A}^{(N)}} \|\boldsymbol{\mathcal{X}} - [\![\boldsymbol{A}^{(1)},\dots,\boldsymbol{A}^{(N)}]\!]\|^2$$
$$\text{subject to } \boldsymbol{A}^{(n)} \geq 0 \quad \forall n = 1,\dots,N \tag{2.26}$$

To solve (2.26), one could form a loss function including the norm of the error, and penalty terms for the non-negativity constraints [Cichocki et al., 2009]. Another approach is by extending Lee and Seungs non-negative matrix factorization to general tensors of order N. The update rule in (2.25) then translates to:

$$\begin{aligned} \boldsymbol{A}^{(n)} &\leftarrow \boldsymbol{A}^{(n)} * \left(\boldsymbol{X}_{(n)}\boldsymbol{A}_{\odot}^{(-n)}\right) \oslash \left([\![\boldsymbol{A}^{(1)},\dots,\boldsymbol{A}^{(N)}]\!]_{(n)}\boldsymbol{A}_{\odot}^{(-n)}\right) \\ \boldsymbol{A}_{\odot}^{(-n)} &= \boldsymbol{A}^{(N)} \odot \cdots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \cdots \odot \boldsymbol{A}^{(1)} \end{aligned} \tag{2.27}$$

where one has computed the gradient of $\|\boldsymbol{\mathcal{X}} - [\![\boldsymbol{A}^{(1)},\dots,\boldsymbol{A}^{(N)}]\!]\|^2$ and applied gradient descent with a multiplicative update, setting the learning rates such that positivity is preserved [3] [Shashua and Hazan, 2005].

Non-Negative Tucker Decomposition Computing the non-negative Tucker decomposition, Kim and Choi utilize the properties of unfolding the decomposition. Recalling the Tucker decomposition, unfolding the tensor $\boldsymbol{\mathcal{X}} \in \mathbf{R}^{I_1\times\cdots\times I_N}$ along the nth mode answers to [Kolda and Bader, 2009]:

$$\begin{aligned} \boldsymbol{X}_{(n)} &\approx \left[\boldsymbol{\mathcal{G}} \times_1 \boldsymbol{A}^{(1)} \times_2 \cdots \times_N \boldsymbol{A}^{(N)}\right]_{(n)} \\ &= \boldsymbol{A}^{(n)}\boldsymbol{G}_{(n)}(\boldsymbol{A}^{(N)} \otimes \cdots \otimes \boldsymbol{A}^{(n+1)} \otimes \boldsymbol{A}^{(n-1)} \otimes \cdots \otimes \boldsymbol{A}^{(1)})^T \end{aligned} \tag{2.28}$$

which can be simplified to a matrix multiplication between two factors:

$$\begin{aligned} &\boldsymbol{X}_{(n)} \approx \boldsymbol{A}^{(n)}\boldsymbol{G}_{\boldsymbol{A}}^{(-n)} \\ &\text{where } \boldsymbol{G}_{\boldsymbol{A}}^{(-n)} = \boldsymbol{G}_{(n)}(\boldsymbol{A}^{(N)} \otimes \cdots \otimes \boldsymbol{A}^{(n+1)} \otimes \boldsymbol{A}^{(n-1)} \otimes \cdots \otimes \boldsymbol{A}^{(1)})^T \end{aligned} \tag{2.29}$$

allowing the decomposition to be read as a matrix factorization of $\boldsymbol{X}_{(n)}$. The decomposition can thus, be viewed as N sub-problems; the non-negative matrix factorization for every mode $n = 1,\dots,N$. By Lee and Seung, the least squares solution of the non-negative matrix factorization can be solved iteratively. For every mode n, the updating rule translates to:

[3] For a complete derivation, see [Shashua and Hazan, 2005]

$$\boldsymbol{A}^{(n)} \leftarrow \boldsymbol{A}^{(n)} * \left(\boldsymbol{X}_{(n)} \boldsymbol{G}_{\boldsymbol{A}}^{(-n)^T}\right) \oslash \left(\boldsymbol{A}^{(n)} \boldsymbol{G}_{\boldsymbol{A}}^{(-n)} \boldsymbol{G}_{\boldsymbol{A}}^{(-n)^T}\right) \tag{2.30}$$

Continuing with the second factor, $\mathbf{G}_{(n)}$ and not $\mathbf{G}_{\boldsymbol{A}}^{(-n)}$ is of interest. Therefore, the expression in (2.28) is rewritten as $\boldsymbol{X}_{(n)} \approx \boldsymbol{A}^{(n)} \boldsymbol{G}_{(n)} \boldsymbol{A}_{\otimes}^{(-n)^T}$. For the tensor $\boldsymbol{\mathcal{G}}$, the update rule becomes:

$$\boldsymbol{\mathcal{G}} \leftarrow \boldsymbol{\mathcal{G}} * \frac{\boldsymbol{\mathcal{X}} \times_1 \boldsymbol{A}^{(1)^T} \times_2 \cdots \times_N \boldsymbol{A}^{(N)^T}}{\boldsymbol{\mathcal{G}}_k \times_1 \boldsymbol{A}^{(1)^T} \boldsymbol{A}^{(1)} \times_2 \cdots \times_N \boldsymbol{A}^{(N)^T} \boldsymbol{A}^{(N)}} \tag{2.31}$$

which was derived in detail in [Kim and Choi, 2007], and can be found in **Non-negative Tucker Decomposition Derivation** Appendix.

As proven by [Lee and Seung, 1999], computing the factor matrices and the core iteratively from (2.30) and (2.31) leads to convergence.

Rank Selection of Decompositions

Computing a tensor decomposition for a rank R lower than the rank R^* of the tensor itself, one obtains (as mentioned) an approximation. A reasonable question is what rank R one should choose. To start with, the rank R^* of the tensor is often unknown, and calculating R^* is an NP-hard problem, and thereby not an option [Kolda, 2006]. Instead, one tends to focus on defining a suitable R only, assuming it will have lower rank than the original tensor. Considering the rank R is directly correlated to the shape of the factor matrices ($\boldsymbol{A}^{(n)} \in \boldsymbol{R}^{I_n \times R}$ for CP, and $\boldsymbol{A}^{(n)} \in \boldsymbol{R}^{I_n \times R_n}$ for Tucker), a small rank is beneficial in terms of dimensionality reduction. The ideal rank should thus be as low as possible, while still capturing prominent characteristics such that the decomposition suggests a good approximation. Based on work from [Phan and Cichocki, 2010], such a rank can be calculated for every mode by determining the extent of variation for each dimension. For a tensor $\boldsymbol{\mathcal{X}}$ of order N, a measure of the variation along mode n can be translated from the eigenvalues along that mode:

$$\boldsymbol{X}_{(n)} \boldsymbol{X}_{(n)}^T = \boldsymbol{U\Sigma V}^T \tag{2.32}$$

where $\boldsymbol{U\Sigma V}^T$ is the SVD of $\boldsymbol{X}_{(n)} \boldsymbol{X}_{(n)}^T$, and $\boldsymbol{X}_{(n)}$ answers to the original tensor $\boldsymbol{\mathcal{X}}$ unfolded along the nth mode. The diagonal matrix $\boldsymbol{\Sigma}$ thereby contains the squared eigenvalues of $\boldsymbol{X}_{(n)} \boldsymbol{X}_{(n)}^T$, assumed to represent the variation of mode n. Applying the elbow technique, the rank of the corresponding factor matrix $\boldsymbol{A}^{(n)}$ can be chosen such that a significance α of the variation is explained:

$$\underset{R_n}{\operatorname{argmin}} \frac{\sum_{i=1}^{R_n} \lambda_i}{\sum_{i=1}^{I_n} \lambda_i} > \alpha \tag{2.33}$$

where λ_i is the ith diagonal entry of $\mathbf{\Sigma}$, corresponding to the ith largest singular value. For example, choosing $\alpha = 0.95$ would yield a rank R_n that results in modeling 95% of the variability of mode n [Phan and Cichocki, 2010].

Applying the approach to the Tucker decomposition, one simply determines the rank of every factor matrix by (2.33). For the CP decomposition, one similarly calculates the rank of every mode by (2.33), but then has to settle for one of them. This, as all factor matrices must have the same number of columns in the CP decomposition.

3

Machine Learning Theory

The following chapter will serve as short description of the supervised machine learning classifiers and regressors used in this project. The descriptions will be based on the theory of classifiers (labeling discrete classes), but the methods can be generalized to continuous response variables (regression). What is of most relevance in this chapter is what techniques the different methods apply to predict and separate labels. This will later be relevant and discussed when put in relation to structures in EEG data, in combination with the tensor operations described in Chapter 2 **Tensor Theory**.

3.1 Support Vector Machines

The Support Vector Machine (SVM) is an extension of a Support Vector Classifier (SVC). An SVC is based on finding separating hyperplanes between classes. Given a 2-dimensional space, the separation answers to a line, and for the general case of an N-dimensional space, one has an $N-1$ dimensional hyperplane. Ideally, for a dataset with binary classes and separable data, data from one class would be located on one side of the hyperplane, and data from the other class on the other side. Since there often are several hyperplanes that could achieve this (for example by shifting the location of the hyperplane a tiny bit), the *maximal margin hyperplane* is usually selected. This is the hyperplane that is farthest from all training data. It can be found by computing the perpendicular distance for all training data points to the hyperplane, then selecting the one from each class that is closest to the hyperplane, and maximizing the corresponding distances. These data points are the *support vectors*, which are the only points that directly affect the location of the hyperplane. If these points are moved, the hyperplane will move as well. New data is classified based on what side of the hyperplane it is located [James et al., 2013].

Sometimes, no such hyperplane that separates all data into the two binary classes exists. Additionally, if it does exist, it might be overfitting the data as it only adapts to the support vectors, which can be outliers. Therefore, a soft margin classifier is often used, which allows some data points to be on the incorrect side of the margin/hyperplane. This is controlled by a tuning parameter C, which bounds the violation tolerance [James et al., 2013].

The SVM, in contrast to the SVC, applies a linear/nonlinear transformation to the training data, referred to as a *kernel* $K(\cdot,\cdot)$. The feature space of the training data can consequently be enlarged, and enhance different relationships and interactions between data points, for instance nonlinear ones. For the transformed training data, a hyperplane is estimated to separate labels, with hopes of more easily distinct different classes in this enlarged feature space. In the linear SVM (which consequently, is an SVC), the linear kernel in (3.1) is used. An example of another well established kernel is the radial basis function (rbf) kernel, see (3.2):

$$K(x_i, x_{i'}) = \sum_{n=1}^{N} x_{in} x_{i'n} \tag{3.1}$$

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{n=1}^{N} (x_{in} - x_{i'n})^2) \tag{3.2}$$

where x_i and $x_{i'}$ are two training observations, N is the number of features and γ is a positive constant. Using an rbf kernel instead of the standard linear one leads to a more flexible decision boundary. In particular, the rbf kernel enhances a local behavior, as only training observations nearby a test observation have an effect on the class label [James et al., 2013].

3.2 Random Forests

The Random Forest method utilizes a combination of several *Decision Trees*. A Decision Tree applies recursive binary splittings to find features and thresholds that divide the data into different classes, which is a nonlinear procedure. The aim is to choose the split that minimizes a loss function and thus, yields the best classification of the data [James et al., 2013].

A *Random Forest* uses the average of several Decision Trees. The Decision Trees are here only allowed to consider a random sample of m input features during each split, and then choose one of these features for the split. Typically, m is selected as the square root of the total number of features. With this technique, the variance is reduced as the trees are not as probable of being correlated (since they are ”forced” to use different sets of features), compared to for example bagging where a simple averaging of different trees is applied [James et al., 2013].

Random Forests have several different tuning parameters, which can be specified to affect regularization and overfitting. The depth of each tree, the number of trees, and requirements on each binary split can for example be selected. Good values for these parameters are usually highly dependent on the structure of the data [James et al., 2013].

4 Datasets

During this project, two different datasets were used: one small, binary (Alex Motor Imagery) and one larger with a continuous response variable (Sustained-Attention Driving Task Dataset). The datasets have contributed to a detailed and complex analysis of tensor decompositions of EEG signals in the field of transfer learning. They were both chosen as they illuminated different features, structures and possible applications of the result. In this chapter, the datasets will be described in detail.

Throughout the report, a *trial* (in literature sometimes also denoted *epoch*), refers to a continuous time series of an EEG signal that occurs around an event of interest. Every trial receives a *label* specifying the event, and has usually been extracted from a longer time interval during a *session*. The sessions are in turn related to a *subject*, which is the person who is performing the tasks and whose EEG signals are measured. For example, a trial in the binary dataset Alex Motor Imagery represents the 3 second EEG signals during an imagined hand or feet movement, which are the two possible labels. A trial of the Sustained-Attention Driving Task dataset answers to the 3 second EEG signals before reacting to a car drift. The label of the trial is computed from the subject's reaction time and translates to a continuous drowsiness index.

4.1 Alex Motor Imagery

Dataset Description

The first dataset is publicly available, and taken from the Mother Of All BCI Benchmarks (MOABB) science project. The dataset: Alex Motor Imagery (AlexMI) [Barachant and Chevallier, 2021][1], contained EEG signals of motor imagery, meaning a subject *imagined* motor movements. The data was collected from eight subjects, recorded at 512Hz using 16 wet electrodes placed according to the international 10-20 system. Every subject performed one session of 60 trials, answering to 20 trials of three classes; imagined movement of the right hand, imagined movement of the feet, and doing nothing. All trials were three seconds long and prior to every trial, the subjects were asked to move their right hand, feet, or do nothing, dependent on what they were to imagine during the trial [Barachant et al., n.d.] The placement of the electrodes and their names can be seen in Figure 4.1 below.

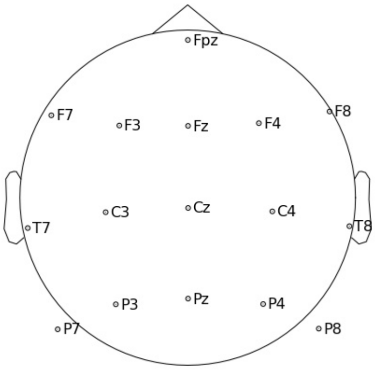


Figure 4.1 Electrode placement of the 16 electrodes during data collecting for the AlexMI dataset. Channel names are printed next to the electrodes.

Trial Extraction

For the classification, only the motor imagery classes were included; right hand and feet. The preprocessing used the MOABB-package in Python, where a *paradigm* is the pipeline defining how to extract trials from continuous EEG data. The MOABB project means to supply a standardized way of labeling EEG data and thus, its library has defined standard pipelines. For this task, the paradigm *MotorImagery()* was chosen with default settings. This resulted in the EEG data being divided into trials, where a trial was extracted from the raw EEG data using

[1] Instructions on how to download the data at
`http://moabb.neurotechx.com/docs/generated/moabb.datasets.AlexMI.html`

the event start and end times pre-denoted by the dataset creators; a three seconds time series of the subject imagining a motor movement. Additionally, the paradigm included a bandpass filter between 8 and 32Hz. The default frequency range 8-32Hz was concluded as reasonable since data characteristics for motor imagery often lies in the alpha- (8-13Hz) and beta- (14-30Hz) bands [*Energy Efficiency of Medical Devices and Healthcare Applications* 2020]. Table 4.1 below displays an overview of the dataset:

Table 4.1 Details about the AlexMI dataset used for the classification of right hand and feet respectively. Note that the labels were randomly mixed within each session, making it not possible for the subject to know what label to expect during each trial.

Subject	**Sessions**	**Trials**	**Label**
1	1	40	20×"right_hand" and 20×"feet"
⋮	⋮	⋮	⋮
8	1	40	20×"right_hand" and 20×"feet"
Total: 8	**Total: 8**	**Total: 320**	**Total: 160**×"right_hand" and **160**×"feet"

All trials were included, leading to 40 trials per session and subject where 20 trials answered to an imagined movement of the right hand and feet respectively. Consequently, there were 320 trials in total. For more detailed information on the dataset, see [Barachant and Chevallier, 2021].

The 3 first trials of the two classes (3 hand and 3 feet, so 6 trials in total) of a session are defined as the *pre-trial data* of that session. This is what will be used for session similarity comparisons. The advantages of representing a new target session just by its pre-trial data are that the data can be quickly collected, and that it will be computationally inexpensive to use during comparisons between sessions.

4.2 Sustained-Attention Driving Task Dataset

Dataset Description

The second dataset was, just as AlexMI, a publicly available EEG dataset, this one collected during a sustained-attention driving task [Cao et al., 2019]. In this report, the dataset will be denoted as the *SA Driving* dataset. The data had been collected from 27 subjects aged 22-28 years. Each subject participated in one to five 90 minute sessions, resulting in 62 sessions in total. The task of each session was for the subject to react to *deviation onsets* (the car starting to drift), during a simulated night driving session. Each deviation onset together with the participant's response onset and back-adjustment to the center of the original lane was defined as a *deviation event*. The *reaction time* (RT) during an event was defined as the time from the deviation onset to the response onset. The response onset was selected as the moment the subject started steering the wheel of the car to counteract the car drift. As the car was constantly moving at the speed 100 km/h in the simulation, there was no need for the subject to control the accelerator or breaks. This made it possible to view steering as the only response variable of the test. See Figure 4.2 for further explanations of the timeline of an event.

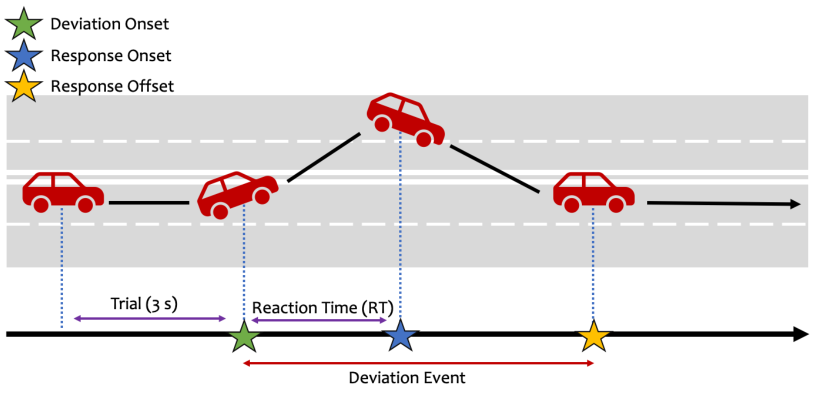


Figure 4.2 Timeline of an event during the Sustained-Attention Driving Task. The 3 seconds of EEG data are extracted before the deviation onset (beginning of car drift). The reaction time is measured as the time from the deviation onset to the response onset. The deviation onset together with the response onset and back-adjustment to the center of the original lane (ending at the response offset) is a deviation event.

A new deviation event was initialized 5-10 seconds after a finished one. During the session, 30-channel EEG signals were recorded from the participants using a wired EEG cap, with sampling frequency of 500 Hz. The electrodes were placed according to the modified international 10–20 system, see Figure 4.3:

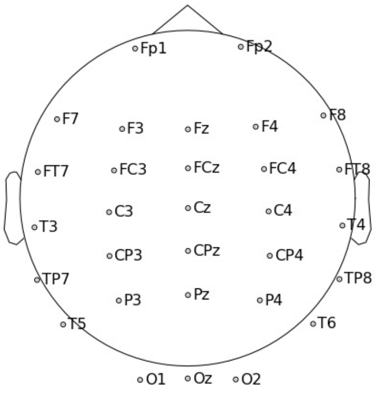


Figure 4.3 Electrode placement of the 30 electrodes during data collecting for the SA Driving dataset. Channel names are printed next to the electrodes.

The pre-processed version of the dataset from [Cao et al., 2019] was used, which had been prepared with bandpass filtering (1-50Hz) and artifact rejection. The artifact rejection included manually removal of evident eye-blink in EEG-signals and automatic removal of ocular and muscular artifacts.

Trial Extraction

3 second *trials* were extracted from the driving task EEG data set. Each trial contained 3 seconds of EEG data from right before a deviation onset (see Figure 4.2). Additionally, the RT for the corresponding deviation event was saved for each trial. For a session to be split into trials, it had to fulfill two requirements:

1. *The first 10 trials of a session were all below the non-drowsy limit.*
2. *The session contained at least 10 trials above the non-drowsy limit.*

where the non-drowsy limit was defined as $1.5\mu_0$, with μ_0 being the median RT of the first 10 trials. These 10 first trials of a session are defined as the *pre-trial data* of that session, which is what will be used for session similarity comparison (imitating the approach and definition in [Jeng et al., 2021]). Just as for AlexMI, there are several advantages with representing a new target session just by its pre-trial data. First of all, 10 trials of calibration data is quickly collected (less than three minutes) and computationally inexpensive to use for comparison between sessions. Additionally, in practice it would be more realistic and beneficial to collect only non-drowsy data for calibration of a new session, as drowsy trials would require exposing the subject to a dangerous traffic situation.

The trimmed dataset contained 19 177 trials from 42 sessions with 24 different subjects. The data for sessions 4 and 5 were identical, even though they were documented as belonging to different sessions and subjects. Both sessions were therefore excluded, as it was not possible to identify if the data belonged to subject 4 or 5. The final dataset contained 18455 trials from 40 sessions with 23 different subjects, see Table 4.2.

Table 4.2 Details about the final, trimmed dataset, containing 18455 trials from 40 sessions with 23 different subjects. μ_0 is the median RT of the first 10 trials for each session, which was used to filter out wanted sessions.

Subject	**Sessions**	**Trials**	$\boldsymbol{\mu_0}$ [s]
1	2	309, 533	1.078, 1.672
2	2	332, 344	0.686, 0.776
5	1	683	0.770
9	2	188, 254	0.920, 0.988
12	1	351	1.038
13	2	479, 269	0.586, 0.652
14	1	333	0.534
22	3	333, 509, 515	0.584, 0.800, 0.636
23	1	439	0.736
31	2	640, 566	1.188, 1.120
35	2	540, 555	0.718, 0.680
40	2	675, 632	0.860, 0.786
41	3	322, 527, 413	0.634, 0.500, 0.626
42	1	548	1.004
43	2	711, 564	0.602, 1.094
44	3	521, 662, 568	1.002, 0.568, 0.768
45	1	688	0.620
48	1	350	0.620
49	1	343	0.552
50	2	361, 334	0.786, 0.870
53	3	244, 510, 464	0.692, 0.696, 0.684
54	1	205	0.712
55	1	641	0.616
Total: 23	**Total: 40**	**Total: 18 455**	

Drowsiness Index

During the Sustained-Attention Driving Task, there were two problems with the raw response variable RT (intended to use for prediction). First of all, the distribution of RTs was unbalanced with low frequency of high values, see Figure 4.4. An unbalanced dataset results in less likeliness to predict minority classes, corresponding to high reaction times (drowsiness) in this case. A particular issue that contributed to this large spread in long RTs was that the subjects who had fallen asleep during the task were not woken up, resulting in occasional RTs longer than 100s.

The second problem was that RTs were session dependent, which becomes evident when looking at the variation in μ_0 for different sessions in Table 4.2. The drowsiness level for a specific subject could not be predicted by solely looking at the absolute value of a RT, but had to be individually adapted. For instance could 1.5 s be considered drowsy for one session and non-drowsy for another, depending on the subject's non-drowsy normal reaction time.

To account for these two problems, a *Drowsiness Index* (DI) was created, using the definition in [Wei et al., 2018]. The corresponding conversion from RT to DI for a trial m in a specific session can be seen in (4.1):

$$\mathrm{DI}_m = \max(0, \frac{1 - e^{-a(\mu_m - \mu_0)}}{1 + e^{-a(\mu_m - \mu_0)}}) \tag{4.1}$$

where μ_0 is the median RT of the first 10 trials in that session, μ_m is the RT of trial m and a is a constant set to 1 s^{-1}. The raw RT distribution and corresponding DI distribution can be seen in Figure 4.4.

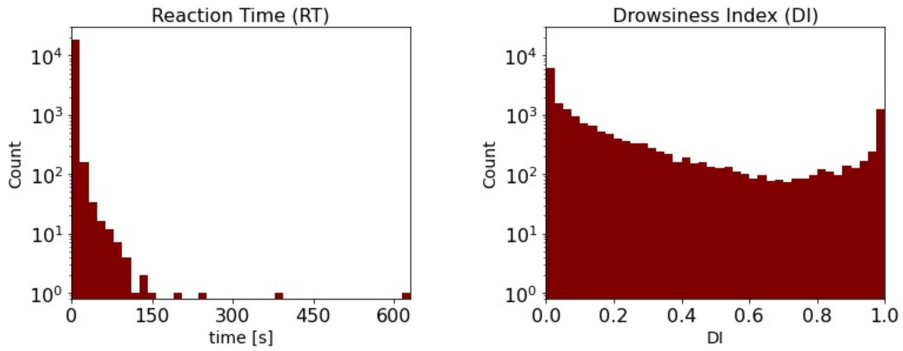


Figure 4.4 Raw RT distribution (left) and DI distribution after conversion of RTs according to 4.1 (right). Both distributions are plotted in log scale.

5

Tensor Decompositions for Session Selection

Throughout this project, the main objective has been to evaluate tensor decompositions of EEG signals in the field of transfer learning. The first pipeline, *Tensor Decompositions for Session Selection*, is described in this chapter. The approach is based on the assumption that a classifier/regressor will benefit from only training on sessions which have EEG data that is similar to the target session (which has been seen previously in literature, for example in [Jeng et al., 2021]). Here, tensor representations were thus only applied as a session selection step to measure similarity between different sessions. In Figure 5.1, this selection process is visualized. Additionally, two baseline methods (random and Riemann selection) were also created to validate the results.

In Section 5.1 **Background**, an overview of the implemented approach can be found, inspired by the article *Low-Dimensional Subject Representation-Based Transfer Learning* [Jeng et al., 2021]. Furthermore, the section includes explanations of how the factor matrices of the tensor decompositions can be interpreted, which later will be used for session selection. Thereafter, our extensions to the method used by Jeng et al. will be described in Section 5.2 **Our Contribution**. This will be followed by a detailed description of our approach in 5.3 **Method**, and the results combined with a discussion in 5.4 **Result and Discussion**.

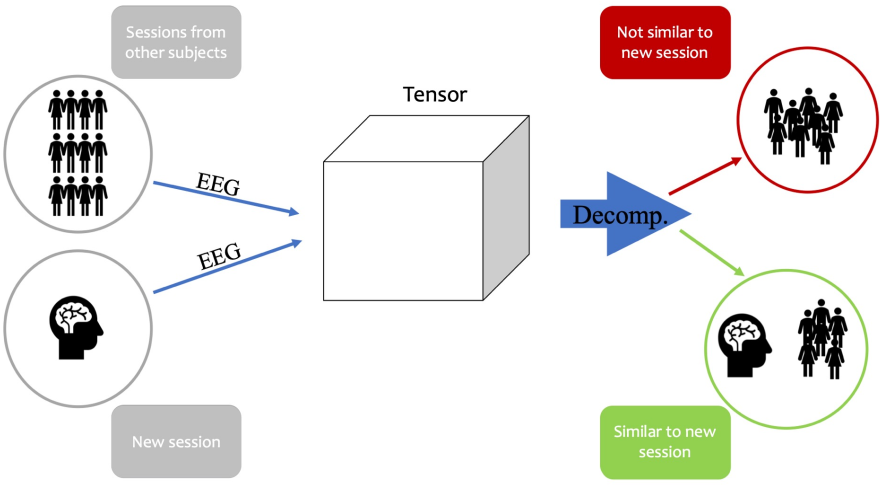


Figure 5.1 Session selection process using tensor decompositions. A tensor was created from a database of training sessions and a target session. A tensor decomposition was performed, and from it, the similarity of all training sessions and the target session was measured. The training sessions most similar to the target session were extracted and later used to train a classifier/regressor.

5.1 Background

In *Low-Dimensional Subject Representation-Based Transfer Learning* [Jeng et al., 2021], the SA Driving dataset was used to train regressors, which were to predict the drowsiness levels in a target session. Their approach was the following: firstly they converted each 3 second EEG trial to the frequency domain. The resulting power spectral density[1] (PSD) of a trial was used as the input features to the regressor. Thereafter, to evaluate the regressors, they used leave one session out (LOSO) cross-validation, meaning all sessions were viewed as the target session once. A subset of all available sessions were selected for training, with the aim of selecting the ones most similar to the target session. They performed a selection process by using a regularized CP decomposition[2] and studying the resulting factor matrix of the session mode, where each session was represented by a *latent vector*. The Pearson Correlation between latent vectors was used as the similarity measure. They found the tensor selection method could improve the prediction performance with 95% significance when choosing a subset of sessions, both

[1] The spectral energy distribution, that is, how much power each frequency component contains [Stoica and Moses, 2005].

[2] The regularization reduced the complexity of the model, see [Jeng et al., 2021] for details.

compared to random selection and a Riemannian selection method.

Inspired by the article, we have followed a similar approach but with several modifications. Before introducing these modifications, a detailed description of how tensor decompositions can be interpreted as a similarity measure will be given. The terms factor matrices, latent variables and latent vectors will play an important role in this, and will be explained next.

Factor Matrix Interpretations

For a tensor $\boldsymbol{\mathcal{X}}$ of order N, both the CP and Tucker decompositions yield N factor matrices (one for each mode). These factor matrices contain information of the variability in the corresponding mode.

CP Decomposition Consider a third order tensor, composed of the EEG data (session × channel × frequency). Performing a CP decomposition of rank R results in the three factor matrices $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \boldsymbol{A}^{(3)}$ of shape (# sessions × R), (# channels × R) and (# freqs × R) respectively. Together, they approximate the original tensor, $\hat{\boldsymbol{\mathcal{X}}} = [\![\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \boldsymbol{A}^{(3)}]\!]$, recall (2.4). Figure 5.2 visualizes this for the example with a third order tensor.

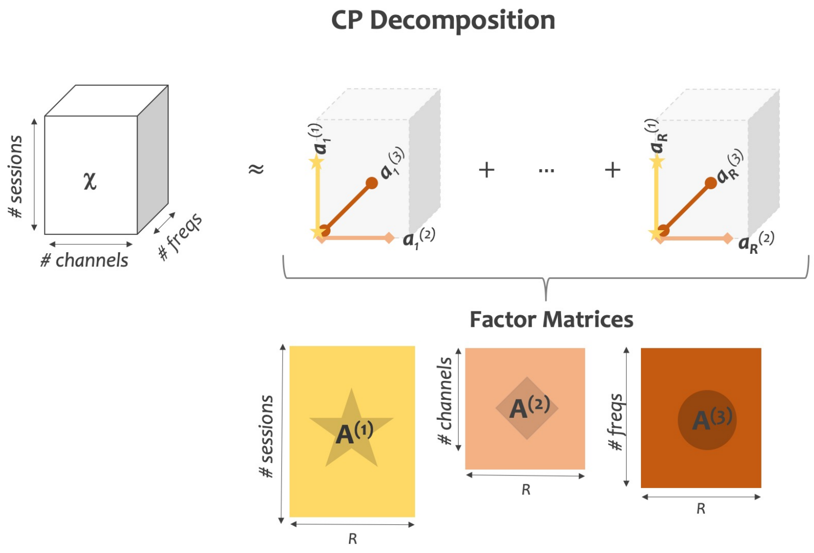


Figure 5.2 Visualization of the CP decomposition for a third order tensor (session × channel × frequency). The decomposition results in three factor matrices, $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \boldsymbol{A}^{(3)}$, each corresponding to one of the three modes. To approximate the original tensor $\boldsymbol{\mathcal{X}}$, R tensors of the same shape as $\boldsymbol{\mathcal{X}}$ are summed together, each tensor created from three rank one tensors (vectors), corresponding to the columns in the factor matrices.

Looking at a single column in one of the factor matrices, for example the first column $\mathbf{a}_1^{(1)}$ in $\boldsymbol{A}^{(1)}$, it can be seen that to approximate the original tensor, every element is multiplied with the first columns $\mathbf{a}_1^{(2)}$ and $\mathbf{a}_1^{(3)}$ of $\boldsymbol{A}^{(2)}$ and $\boldsymbol{A}^{(3)}$. This becomes evident from (2.4), and the definition of the outer product (see 8.1 **The Outer Product** in the Appendix). The elements in $\mathbf{a}_1^{(1)}$ can thus be considered *scaling* of how much of the structure described by the first column in the other factor matrices should be included. We define the elements of a column as different outputs of the same *latent variable* ($l_r^{(n)}$). Thus, a latent variable describes a coordinate in a vector space spanned by the corresponding columns in the other factor matrices. This vector space is the set $\{\mathbf{a}_r^{(k)}\}$, where $k \neq n$. For example, each session is therefore represented by a specific coordinate, a *latent variable value* $l_{r,s}^{(n)}$, in the vector space $\{\mathbf{a}_r^{(k)}\}$, where $k \neq 1$, corresponding to columns r in the channel and frequency factor matrices. The relation of the latent variable values can thus be interpreted as a description of how similar two sessions are in that specific vector space. See Figure 5.3a for the latent variable definition in the example.

Looking at a single row in one of the factor matrices, for example the first row in $\boldsymbol{A}^{(1)}$, it can be seen that every element corresponds to values of different latent variables $l_{r,1}^{(1)}$, where $r = 1, \ldots, R$. We define a row of latent variables as a *latent vector* $\mathbf{v}^{(n)}$. Each session is thereby represented by a latent vector $\mathbf{v}_s^{(n)}$, which is a specific set of R latent variable values, each value a coordinate in one of the R different vector spaces. The relation between latent vectors of different sessions can be interpreted as a description of how similar two sessions are. See Figure 5.3c for the latent vector definition in the example.

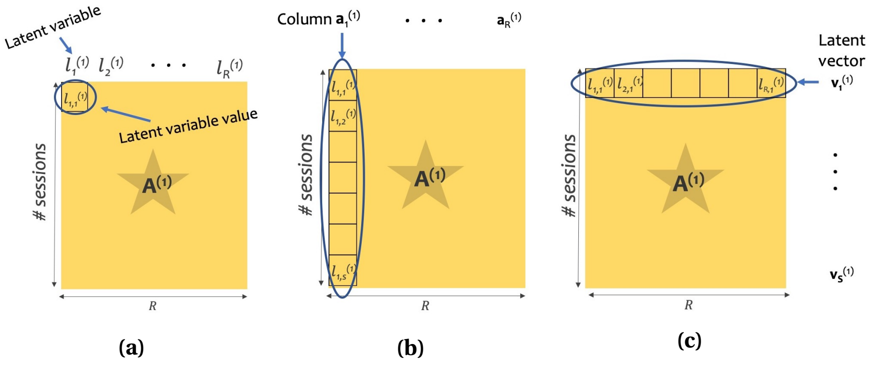


Figure 5.3 Definitions in a factor matrix: (a) *latent variable* $l_r^{(n)}$, *latent variable value* $l_{r,s}^{(n)}$, (b) column $\mathbf{a}_r^{(n)}$, and (c) *latent vector* $\mathbf{v}_s^{(n)}$. $\boldsymbol{A}^{(1)}$ corresponds to the factor matrix of the first mode in the example tensor (session × channel × frequency), which means that $n = 1$.

Tucker Decomposition In the Tucker decomposition, the factor matrices do not all have to be of the same rank. Consider again the previous example: a third order tensor composed of the EEG data (session × channel × frequency). Performing a Tucker decomposition of rank (R_1, R_2, R_3) results in three factor matrices $\boldsymbol{A}^{(1)}$, $\boldsymbol{A}^{(2)}$, $\boldsymbol{A}^{(3)}$ of shapes (# sessions × R_1), (# channels × R_2), and (# freqs × R_3), together with a core tensor $\boldsymbol{\mathcal{G}}$ of shape $(R_1 \times R_2 \times R_3)$.

For the Tucker decomposition, the same notation as in the CP case of *latent variables*, *latent variable values*, and *latent vectors* are used for the components of the factor matrices. The constraint that each latent variable is only multiplied with a single column in the other factor matrices is however removed (due to the core tensor). This allows for a more flexible combination of the latent variables. As a consequence, one can not consider the columns of the factor matrices to "belong together" in the same way as for the CP decomposition. This means that each latent variable can not be mapped to a specific vector space in a corresponding matter. Instead, all latent variables $l_{r_n}^{(n)}$ scale the same set $\{\mathbf{a}_{r_k}^{(k)}\}$, where $k \neq n$ and $r_k = 1...R_k$. The scaling and combinations of columns is further specified by the core tensor $\boldsymbol{\mathcal{G}}$.

5.2 Our Contribution

EEG Data Normalization

In *Low-Dimensional Subject Representation-Based Transfer Learning* [Jeng et al., 2021], the EEG-data for each 3 second trial was converted to its PSD. Thereafter, the trials were pre-processed by applying the logarithm and subtracting the median of each session's pre-trials (first 10 trials of a session). The last step created a relative adaption of the data for each session. The corresponding data contained positive and negative values around 0.

As the PSD of EEG-data is positive, we propose to retain the same sign of all data after the normalization. This makes it possible to restrict the tensor decompositions to non-negative ones, increasing the probability of finding unique solutions. Similarly to Jeng et al., we converted the EEG-data to its PSD, resulting in values around 10^{-15}. We then applied the logarithm, resulting in only negative values as the PSDs were positive but far less than 1. Thereafter, we *divided* the data with the median of the pre-trials instead of subtracting it. All data was now positive and contained values around 1. The last step was done as we found it beneficial to keep the relative normalization for each session. This made it possible to reduce the impact of data collection differences between sessions, for example electrode impedance. Note that this normalization transforms the data to the counter-intuitive form that values above 1 correspond to PSDs *lower* than the median pre-trial, and values below 1 correspond to PSDs *higher* than the median pre-trial.

Tensor Decompositions

In *Low-Dimensional Subject Representation-Based Transfer Learning* [Jeng et al., 2021], every session was represented by computing the mean of its pre-trials, resulting in a matrix of shape (channel × frequency). From these matrices, a third order tensor of shape (session × channel × frequency) was created. Jeng et al. created a regularized CP decomposition algorithm, which implemented gradient descent. They motivate the implementation of this algorithm as reducing the complexity of the decomposition.

As we applied a different normalization to the data, resulting in only positive values, we instead used only non-negative tensor decompositions. Additionally, we did not only consider the CP decomposition, but also studied the Tucker decomposition. The non-negative constraint was from empirical testing enough to find unique solutions in both cases; CP computed with ALS and Tucker computed with HOOI.

In *Low-Dimensional Subject Representation-Based Transfer Learning* [Jeng et al., 2021], a rank 10 CP decomposition was performed. For the Tucker decomposition, we applied a rank selection such that 95% of the variability in the original tensor $\mathcal{X}$ was explained by the factor matrices, see section 2.2 **Rank Selection of Decompositions**. For the CP decomposition, we performed the same rank investigation to find 95% of the variability in each mode, and then chose the smallest of the three resulting ranks as the mutual R. The 95% significance was chosen with the purpose of preserving the variability in the data, while still excluding noise.

Similarity Measure

In *Low-Dimensional Subject Representation-Based Transfer Learning* [Jeng et al., 2021], the session selection process was performed using a CP decomposition and studying the factor matrix of the session dimension. Each session was represented by its latent vector in that matrix, and the Pearson correlation between latent vectors was used as a similarity measure. This means that the intra-relation of latent variable values was considered of most importance when comparing sessions, and not the absolute difference between latent vectors. Consider the example of vector $\mathbf{v}_1 = [1,2,3]$. $\mathbf{v}_1$ will have higher correlation with $\mathbf{v}_2 = [10,11,12]$ than $\mathbf{v}_3 = [3,2,1]$, even though the absolute difference between $\mathbf{v}_1$ and $\mathbf{v}_3$ is smaller than for $\mathbf{v}_1$ and $\mathbf{v}_2$.

Another way to measure similarity and preserve the intra-session-relation is by using the Euclidean distance between *normalized* latent vectors. If all values in a latent vector sum up to 1, one can view the latent variables as capturing the session's *percentage of variability*. $\mathbf{v}_1$ would then become $[0.167, 0.333, 0.5]$, which can be interpreted as that 16.7% of the variability is described by latent variable l_1, 33.3% by latent variable l_2 and 50% by latent variable l_3.

What is excluded in this similarity measurement, as well as in the Pearson correlation measurement used by Jeng et al., is the fact that the latent vector spaces are of varying sizes. We therefore propose a new similarity measure, combining the *percentage of variability*-method and accounting for the weight of each latent vector space. This means not all latent variables are considered equally important, as was the case in Jeng et al., but their impact depends on how much of the data they describe. We consider this to be a more representative measurement of how similar two sessions are. Details on this similarity method for the two decompositions (CP and Tucker) will be described next.

CP Decomposition For the CP decomposition, the weighting of each latent vector space is relatively intuitive. Consider a CP decomposition of rank R of the third order tensor of EEG data (session × channel × frequency). As described in section 5.1 **Background**, the latent variables $l_r^{(n)}$ $r = 1...R$ in the factor matrix of the sessions-mode $(n = 1)$ represent values in the R vector spaces spanned by the sets $\{\mathbf{a}_r^{(k)}\}$, where $k \neq n$. As each of the R latent variables represent a value in a unique vector space, we calculate the size of the corresponding vector space and use as the weighting factor. The size should be proportional to how much of the data each vector space describe. When all vector spaces are non-negative, the Frobenius norm of the columns $\mathbf{a}_r^{(j)}$ can be considered a valid measurement for how much each column contributes to the inverse power approximated tensor. We choose to multiply the norms of all columns in a vector space, as this is most similar to how the recreation of the approximated tensor is made; by multiplicative operations between latent variables and columns. Another approach could be to add the norms together, or with some other combination of operations, create a weighting factor. Our multiplicative weighting factor becomes $\mathbf{w}^{(CP)} = \begin{bmatrix} w_1^{(CP)} & w_2^{(CP)} & \dots & w_R^{(CP)} \end{bmatrix}$, where each element $w_r^{(CP)}$ is computed according to (5.1):

$$w_r^{(CP)} = \prod_k^N ||\mathbf{a}_r^{(k)}||_F, \qquad k \neq n. \tag{5.1}$$

To include the information about the vector space sizes, the Hadamard product between each latent vector and the weighting factor $\mathbf{w}^{(CP)}$ is computed. Thereafter, each latent vector is normalized such that all values sum up to 1. Now, the *percentage of variability*-method can be used to interpret how much of the variability is described by each latent variable. The Euclidean distance between the weighted and normalized latent vectors of each session is viewed as the similarity measure. Similar sessions will hence have a shorter distance than non-similar ones.

Tucker Decomposition As the Tucker decomposition yields a more flexible combination of latent variables, the weighting factor needs to be calculated in a different way. Consider a Tucker decomposition of rank (R_1, R_2, R_3) of the third order tensor of EEG data (session × channel × frequency). The decomposition results in three factor matrices of shape (# sessions $\times R_1$), (# channels $\times R_2$) and (# freqs $\times R_3$), and a core tensor of shape $(R_1 \times R_2 \times R_3)$. To find a representative weight for the latent variables in the session dimension, it is reasonable to analyze what each latent variable is scaling. To do so, all factor matrices but that of the sessions dimension are multiplied with the core tensor. This creates a tensor of size ($R_1 \times$ # channels × # freqs). It now becomes evident that each latent variable in the session dimension is always combined with a specific slice $\mathbf{S}_{r_1}$ of this tensor, see Figure 5.4.

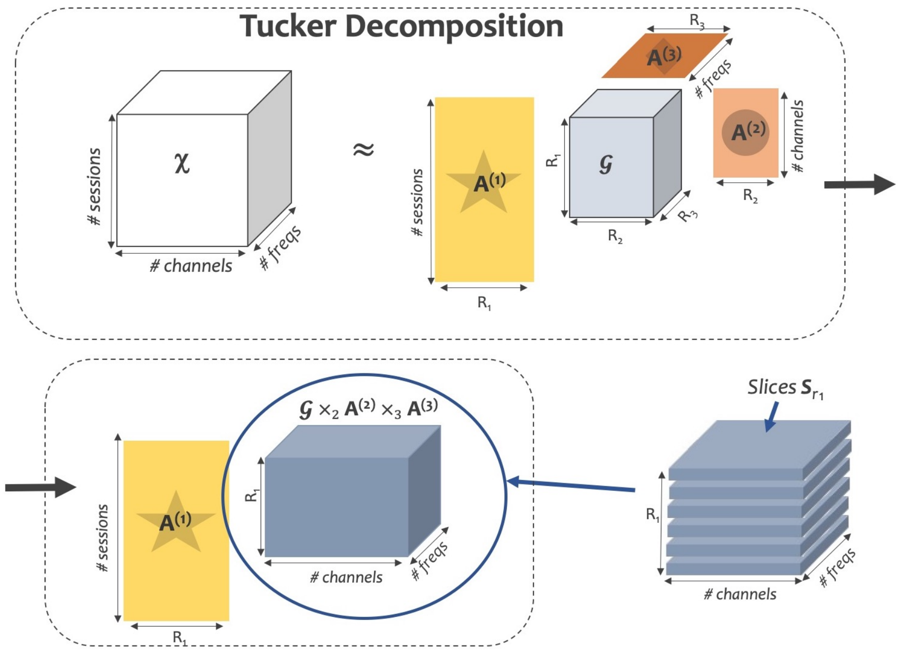


Figure 5.4 Each latent variable in the factor matrix of the session dimension $(\mathbf{A}^{(1)})$ is always combined with a specific slice S_{r_1} from $\boldsymbol{\mathcal{G}} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)}$ during approximation of the original tensor $\boldsymbol{\mathcal{X}}$. The size of the slices can therefore be considered a reasonable weighting factor for the latent variables in $\mathbf{A}^{(1)}$.

Similarly to the CP decomposition case, a weighting factor $\mathbf{w}^{(Tu)} =$

$\begin{bmatrix} w_1^{(Tu)} & w_2^{(Tu)} & \dots & w_{R_1}^{(Tu)} \end{bmatrix}$ can be created, where each element $w_{r_1}^{(Tu)}$ is computed according to (5.2). When all slices are non-negative, the Frobenius norm can be considered a valid measurement for how much of the inverse power of the data each slice describe.

$$w_{r_1}^{(Tu)} = ||\mathbf{S}_{r_1}||_F \tag{5.2}$$

To include the weighting information, the Hadamard product between each latent vector and the weighting factor $w_r^{(Tu)}$ is computed. Thereafter, each latent vector is normalized such that all values in it sum up to 1. The *percentage of variability*-method can now be used to interpret how much of the variability that is described by each latent variable. The Euclidean distances between sessions are viewed as the similarity measure, where similar sessions will have (as for the CP decomposition) a short distance.

5.3 Method

In the following section, a detailed description of this chapter's method can be found. The purpose is for the reader to understand the pipeline, the decisions made, and for reproducibility.

The section firstly handles the small, binary dataset Alex MI, for which accuracy for motor imagery classification is evaluated. Thereafter, the larger dataset SA Drivers will be used for regression analysis. The two datasets illuminate different aspects of the tensor decomposition methods. For Alex MI, focus will be on what structures the tensor decompositions capture and qualities of the similarity measures. This, as it is easier to visualize a small dataset, and that more preknowledge of important channel locations and frequency intervals exists for this binary classification problem. SA Drivers will instead mainly be used to evaluate what effect the tensor decomposition selection has on the performance of the regression. This, as a larger dataset enables a greater variation between sessions and thus, a more detailed analysis of the method's performance. Additionally, regressors can capture small improvements in a model better than binary classification, as the uncertainty of the prediction is more clearly included.

Apart from this, all preprocessing steps, as well as some data exploration in relation to the tensor decomposition methods, will be further looked into for both datasets.

Alex MI

Preprocessing For every trial, the EEG data in each of the 16 channels was transformed into its PSD using a multitaper method[3]. The PSD multitaper transformation resulted in 72 equally spaced bins of size 0.33 Hz between 8 to 32 Hz. As this high resolution was excessive and resulted in unnecessarily long training times later on, the amount of bins was reduced. This was done by applying a moving average filter to the data, using a window size of 3 together with a 3-step slide. The result was data of 25 equally spaced bins of size 1 Hz between 8-32 Hz.

The logarithm was thereafter applied to each trial, and normalization was performed by dividing all trials of a session with the median pre-trial from the corresponding session. This resulted in only positive values centered around 1. Note that this normalization transforms the data to the counter-intuitive form that values above 1 correspond to PSDs *lower* than the median pre-trial, and values below 1 correspond to PSDs *higher* than the median pre-trial. Each trial was now described by a normalized 16×25 matrix of the form (channel × frequency).

Data Exploration To get a better understanding if certain frequencies and channels were more important than others during this specific EEG classification problem, an ANOVA filter was applied to the training database of trials. This was done through leave one session out (LOSO), that is, all sessions were viewed as the target session once (and consequently, excluded from the training database). Using the ANOVA filter, the 10 features that separated the two classes right hand and feet most evidently were extracted. In total, 10×8 features were selected (10 features each for the 8 sessions). Every single feature represented a specific channel location and frequency bin. Statistics of the most frequently chosen frequency and channel combinations were studied. To visually inspect if the selected features differed between the two classes, the frequency ranges of greatest significance were plotted in a topographic map (topomap) together with all channels marked out (see Figures 5.5 and 5.6).

Session Selection: Baselines For baseline comparison, a *random selection* of sessions was applied. For each target session, this method randomly selected k different sessions from the database as the training data.

For comparison against one of today's state of the art methods, a *Riemann selection* method was created [Congedo et al., 2017]. For this method, six channel covariance matrices (of size 16×16) were estimated from the six pre-trials of a session, according to (8.10) in the Appendix. Then for each session, a mean covariance matrix was estimated from the corresponding pre-trial covariance matrices, using the Riemannian geometric mean, see (8.12). To select the ses-

[3] Estimation of the power spectrum using an average of multiple window functions applied to the signal [Thomson, 1982]

sion(s) most similar to the target session, the Riemannian distances (see 8.11)) between the target's and all other sessions' mean covariance matrices were computed. The shorter the distance, the more similar the sessions were assumed to be. The comparison between all sessions was displayed in a distance matrix.

Session Selection: Tensor Decomposition Methods Two different *tensor decomposition selection* methods were evaluated; non-negative CP decomposition and non-negative Tucker decomposition (see section 2.2 **Tensor Decompositions**). In both cases, each session was represented by the mean of its six pre-trials. The tensor was then formulated as (session × channel × frequency), resulting in the shape $(8 \times 16 \times 25)$. The rank of the two decompositions was selected to $\alpha = 0.95$, describing 95% of the variability in the data, see section 2.2 **Rank Selection of Decompositions**. The significance $\alpha = 0.95$ was chosen with the aim of excluding noise and yet, not remove any structure that could be of relevance for the classification. Assuming only a part of the decompositions' identified structures were of relevance, a smaller α should be enough in theory. Because of the unsupervised approach though, it was impossible to know what was excluded when decreasing α. To allow for a safety margin, it was thus chosen to 0.95.

Both decompositions resulted in three factor matrices; one in the session dimension, one in the channel dimension and one in the frequency dimension. The factor matrices of the CP decomposition were consequently, of shape $(8 \times R)$, $(16 \times R)$ and $(25 \times R)$. Correspondingly, the Tucker decomposition resulted in factor matrices individually adapted for each dimension, $(8 \times R_1)$, $(16 \times R_2)$ and $(25 \times R_3)$ respectively. In addition, the core tensor of the Tucker decomposition was of size $(R_1 \times R_2 \times R_3)$. The CP and Tucker similarity measures, proposed in section 5.2 **Our Contribution**, were used and evaluated. For each of the two tensor methods, the distance was pairwise computed between the latent vectors in the session dimension. The shorter the distance, the more similar the sessions were assumed to be. The comparison between all sessions was displayed in a distance matrix, one for each tensor decomposition method. Additionally, the channel and frequency factor matrices were studied and visually inspected to get a better idea of what structures the tensor decompositions captured.

Training and Evaluation For training and evaluation, LOSO cross-validation was applied. For each session, 1-7 other sessions were selected using one of the different session selection methods. To maintain consistent classification results, the selected sessions were sorted according to their subject number (see Table 4.1) . The selected session(s) where used to train a linear Support Vector Classifier (*linear SVC*), a Support Vector Classifier with an rbf kernel (*SVC rbf*) and a Random Forest Classifier (*ranfor*). Minimal tuning was performed to not favor any of the selection methods, details can be found in the project's GitLab repository[4]. For the random selection method, every selection, training and evaluation sequence was repeated 20 times and the average performance was calculated, as the selection was not unique. The average prediction accuracy was used to evaluate the LOSO cross-validation result.

SA Driving

Preprocessing The preprocessing steps for the SA Driving dataset were identical to that of Alex MI, apart from differences in dimensions and numbers. For every trial, the EEG data in each of the 30 channels were transformed into their power spectral densities (PSD) using a multitaper method. The PSD multitaper transformation resulted in 107 equally spaced bins of size 0.33 Hz between 1 to 35 Hz. The range 1 to 35 Hz was selected as a reasonable range, where lower frequencies than 8 Hz were included unlike the Alex MI case. This was as lower frequencies in EEG data are, in theory, enhanced during a subject's decrease in consciousness level (drowsiness), but are not as much of interest for motor imagery (see Table 1.1 in Chapter 1 **Introduction**).

As for Alex MI, the amount of bins was reduced by applying a moving average filter to the data, using the window size 3 together with a 3-step slide. The result was data of 35 equally spaced bins of size 1 Hz between 1-35 Hz. The logarithm was thereafter applied to each trial, and normalization was performed by dividing all trials of a session with the median pre-trial from that corresponding session. Again, this resulted in only positive values centered around 1. Each trial was now described by a 30×35 matrix of the form (channel × frequency).

Data Exploration To get a better understanding of if certain frequencies and channels were more important than others during this specific EEG regression problem, an ANOVA filter was applied to a subset of the training database of trials. Limits were set to filter out 20 drowsy and 20 non-drowsy trials. The limits were defined as reaction times (RTs) above $2.5\mu_0$ s and below $1.5\mu_0$ s for drowsy and non-drowsy respectively, where μ_0 is the median reaction time of the corresponding session's ten first trials. Seven of the 40 sessions did not contain 20 trials with RTs above $2.5\mu_0$ s and were thereby ignored. Similarly to Alex MI, LOSO was applied and the 10 features that separated the two classes drowsy and non-drowsy most evidently were extracted. In total, 10×33 features were selected (10 each for the 33 sessions). Every feature represented a specific channel location and frequency bin, and statistics of the most frequently chosen combinations was studied. To visually inspect if the selected features differed between the two classes, the frequency ranges that were of greatest significance were plotted in a topomap together with all channels marked out (see Figure 5.12).

[4] https://gitlab.control.lth.se/exjobb-bci/emmalinda

Session Selection For a target session, k sessions were chosen to train a classifier. Note that other sessions from the same subject as the target session were excluded during selection. At most, three sessions were recorded from the same subject, allowing the number of training sessions k to range between 1 and 37. During selection, only the ten pre-trials of every session were included.

For baseline comparison, a *random selection* of sessions was applied. For comparison against one of today's state of the art methods, a *Riemann selection* method was created.

Two tensor decomposition selection methods were evaluated; non-negative CP decomposition and non-negative Tucker decomposition. The tensor was formulated as (session × channel × frequency), resulting in a tensor of shape (40 × 30 × 35). The factor matrices of the CP decomposition of rank R were of sizes (40 × R), (30 × R) and (35 × R). Correspondingly, the (R_1, R_2, R_3) Tucker decomposition resulted in three latent factor matrices individually adapted for each dimension to sizes $(40 \times R_1)$, $(30 \times R_2)$ and $(35 \times R_3)$, and a core tensor of size $(R_1 \times R_2 \times R_3)$. The ranks of the decompositions were, using the same argument as for Alex MI, selected to $\alpha = 0.95$. For the CP method, the lowest rank of all dimensions was selected.

See *Session Selection*-specifications of Alex MI for more details, as the selection methods were identical.

Training and Evaluation For training and evaluation, LOSO cross-validation was applied. For each session, 1-37 other sessions were selected using one of the different session selection methods. To maintain consistent classification results, the selected sessions were sorted according to their subject/session number (see Table 4.2). The selected session(s) where used to train a linear Support Vector Regressor (*linear SVR*), a Support Vector Regressor with an rbf kernel (*SVR rbf*), and a Random Forest Regressor (*ranfor*). Minimal tuning was performed to not favor any of the selection methods and due to the computational load of tuning. All tuning details can be found in the Git repository. Before evaluating the performance, a causal moving average filter with window size 11 was applied to the true and predicted drowsiness indices (DIs). The window corresponds to approximately 2-3 minutes when including non-drowsy trials only, and a higher variation in length when including drowsy ones (dependent on the increase in reaction time). The smoothing was made as short-time fluctuations in reaction time were assumed to not be attributions of drowsiness. This was included and argued for in the article which defined the DI: [Wei et al., 2018]. The length of the window was selected according to the amount of pre-trial data, making the prediction performable directly after pre-trial data collection.

The evaluation was done for each subset of sessions by looking at the Pearson correlation and mean absolute error (MAE) of the predicted, smoothed DIs and the true, smoothed DIs of all sessions. For the random selection method, every selection, training and evaluation sequence was repeated 20 times and the average performance was calculated, as the selection was not unique.

5.4 Result and Discussion

Alex MI

Data Exploration After applying the ANOVA filters to the training database during LOSO, it became evident that two frequency ranges, each paired with two specific channels, were of most significance for the classification of imagery movement of right hand or feet. These were frequencies 9-12 Hz (alpha band) with channels C3 and C4, and frequencies 17-19 Hz (beta band) with channels Cz and C4. The two frequency bands appeared in 98 % of the 10 × 8 selected features (10 features each for the 8 sessions). The proportion of each freq-band/channels-pair were the following:

- Frequency range 9-12 Hz appeared in 84 % of the features, and was then in 75 % of the cases in channels C3 or C4.

- Frequency range 17-19 appeared in 14 % of the features, and was then in 91 % of the cases in channels Cz or C4.

The corresponding two frequency ranges for the two classes (averaged over all trials), plotted in topomaps with all channels marked out, can be seen below in Figures 5.5 and 5.6. Note that the normalization is such that low values correspond to higher PSD, and high values to lower PSD. Additionally, the absolute PSD difference between the two classes is displayed to emphasize where the classes differ the most for the specific frequencies.

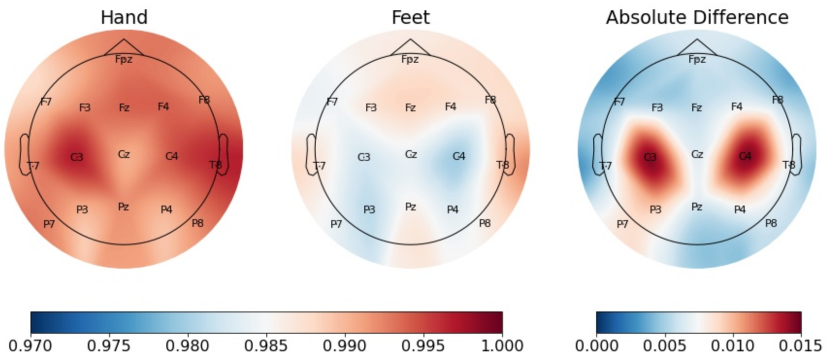


Figure 5.5 Logarithmic, normalized PSD for the frequency range 9-12 Hz, averaged over all trials from all sessions. The figure shows class right hand (left), class feet (middle) and the absolute difference between the two classes (right). Note that the normalization is such that low values correspond to higher PSD, and vice versa.

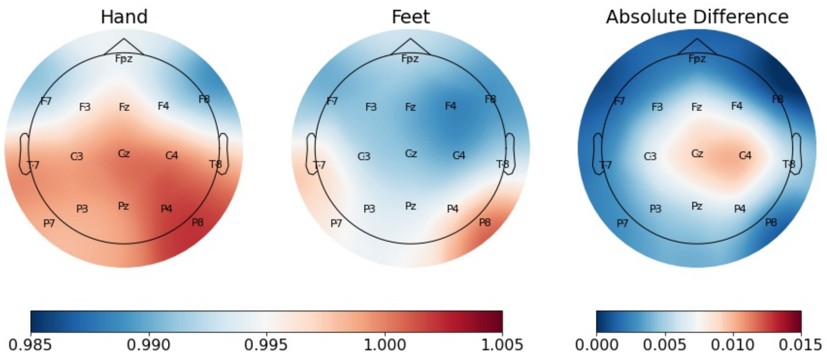


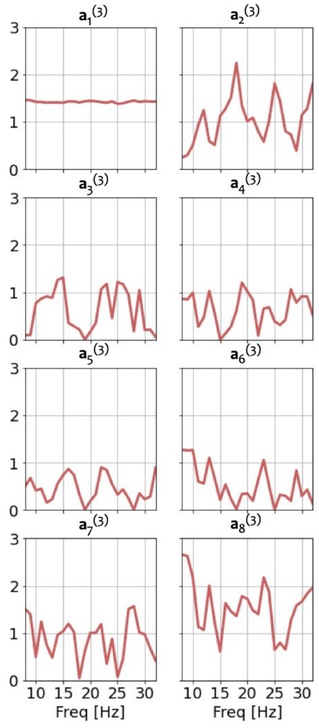
Figure 5.6 Logarithmic, normalized PSD for the frequency range 17-19 Hz, averaged over all trials from all sessions. The figure shows class right hand (left), class feet (middle) and the absolute difference between the two classes (right). Note that the normalization is such that low values correspond to higher PSD, and vice versa.

Looking at the class specific plots in Figure 5.5 and 5.6, it becomes evident that for both frequency ranges, the two classes right hand and feet are relatively well separated. As high values correspond to lower PSD, the EEG signal of imagined hand movement has less power than imagined feet movement in general. Additionally, looking at the absolute difference plot in Figure 5.5, it becomes clear that the channels C3 and C4 differ the most between the two classes in the frequency range 9-12 Hz. Correspondingly, in Figure 5.6 it can be seen that channels Cz and C4 differ the most between the two classes in the frequency range 17-19 Hz.

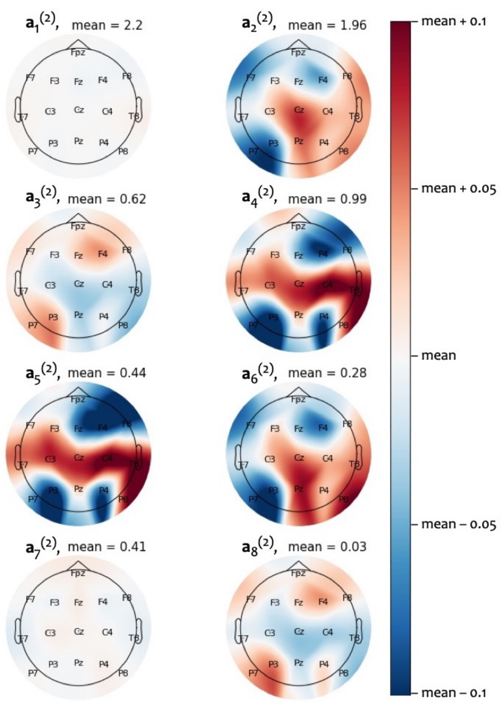
As expected, the most significant channels during this motor imagery classification problem were the ones closest to the brain's motor cortex (C3, Cz and C4). The fact that two narrow, specific frequency ranges (9-12 Hz and 17-19 Hz) within the larger expected interval of interest (8-32 Hz) would be of the greatest importance was less obvious beforehand.

CP Decomposition Selection Results For the chosen significance $\alpha = 0.95$, the rank of the CP decomposition was computed to $R = 8$. As explained earlier in 5.1 **Background**, the rank $R = 8$ CP decomposition of a third order tensor (session × channel × frequency) results in three factor matrices (one for each mode), with 8 columns each. For the factor matrix $\mathbf{A}^{(1)}$ (the session dimension), the latent vectors $\mathbf{v}_s^{(1)}$, with $s = 1, \dots, 8$, should be highlighted as each vector represents one specific session. For the factor matrices $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$ (the channel and frequency dimensions), the columns $\mathbf{a}_r^{(n)}$, where $n = 2, 3$ and $r = 1, \dots, 8$, were of more interest as they make up the 8 vector spaces that are to be scaled by the latent variables in the session dimension. The interpretation of these decomposition results will be presented and analyzed next.

A detailed study of the structures captured by the CP decomposition in the frequency and channel dimensions can be seen below. In Figure 5.7a, the columns of latent variable values $\mathbf{a}_r^{(3)}$ in the frequency dimension factor matrix $\mathbf{A}^{(3)}$ can be seen. In Figure 5.7b, the columns of latent variable values $\mathbf{a}_r^{(2)}$ of the channel dimension factor matrix $\mathbf{A}^{(2)}$ can be seen.



(a) Frequency dimension $\mathbf{A}^{(3)}$.



(b) Channel dimension $\mathbf{A}^{(2)}$.

Figure 5.7 The columns $\mathbf{a}_r^{(n)}$, where $n = 2, 3$ and $r = 1, \ldots, 8$, corresponding to the factor matrices $\mathbf{A}^{(3)}$ (left) and $\mathbf{A}^{(2)}$ (right) in the CP decomposition. The colors of the topomaps represent the deviation from the column's mean value (see colorbar).

Looking at Figure 5.7, it is clear that the columns of each factor matrix describe different structures of the data. As the latent variables in the session dimension can be viewed as *scaling* of the corresponding columns $\mathbf{a}_r^{(n)}$ in the other dimensions, high values in the figure should be interpreted as high variability of that specific characteristic (since the decomposition is non-negative). For example, the local frequency maximas at 12, 18, 25 and 32 Hz in $\mathbf{a}_2^{(3)}$ correspond to larger variability for those specific frequencies than those of the local minimas. This should be combined with the corresponding column $\mathbf{a}_2^{(2)}$ in the channel dimension, which in a similar way has the largest variability at the channels on top of the head (C3, Cz and C4) and to the back right (T8, Pz, P4 and P6). As the latent variable value $l_{2,s}^{(1)}$ of a session s scales the combination of these two columns $\{\mathbf{a}_2^{(2)}, \mathbf{a}_2^{(3)}\}$, the greatest impact of the value will be at those specific frequency and channel maximas.

Comparing the different vector spaces, columns $\{\mathbf{a}_1^{(2)}, \mathbf{a}_1^{(3)}\}$ describe the mean structure of the data, as they have high mean values and little internal variation between different channels/frequencies. The other columns in both dimensions had more internal variation. Notable is that vector space $\{\mathbf{a}_2^{(2)}, \mathbf{a}_2^{(3)}\}$ seem to signify the frequency range 17-19 Hz combined with the channels Cz and C4, which during the data exploration were of particular interest for this classification problem. The other informative combination, frequency range 9-12 Hz paired with channels C3 and C4, was not as clearly pronounced in a single vector space. Although, looking at the three vector spaces $\{\mathbf{a}_4^{(2)}, \mathbf{a}_4^{(3)}\}$, $\{\mathbf{a}_5^{(2)}, \mathbf{a}_5^{(3)}\}$ and $\{\mathbf{a}_6^{(2)}, \mathbf{a}_6^{(3)}\}$, they all seem to have less well pronounced peaks, but nonetheless peaks, at the frequency range 9-12 Hz, and they include the channels C3 and C4 in the channel dimension. In conclusion, it seems like channel-frequency-combinations of interest can be preserved, and even occasionally separated into a single/a few vector space(s), in the CP decomposition.

As seen in Figure 5.7, the mean value and variance of the vector spaces differed quite a bit. The proposed similarity measure (which includes weighting in sizes) thereby affected the selection. This compared to only considering the factor matrix of the session dimension, without taking the other factor matrices into account (as done in [Jeng et al., 2021]). The resulting weights, calculated from the norm of each space, can be seen in Table 5.1:

Table 5.1 The weights of the 8 different vector spaces from the CP decomposition, calculated from the norm of each space.

Vector Space	Weight
$\{\mathbf{a}_1^{(2)}, \mathbf{a}_1^{(3)}\}$	62.5
$\{\mathbf{a}_2^{(2)}, \mathbf{a}_2^{(3)}\}$	44.7
$\{\mathbf{a}_3^{(2)}, \mathbf{a}_3^{(3)}\}$	9.4
$\{\mathbf{a}_4^{(2)}, \mathbf{a}_4^{(3)}\}$	14.2
$\{\mathbf{a}_5^{(2)}, \mathbf{a}_5^{(3)}\}$	4.7
$\{\mathbf{a}_6^{(2)}, \mathbf{a}_6^{(3)}\}$	3.7
$\{\mathbf{a}_7^{(2)}, \mathbf{a}_7^{(3)}\}$	8.0
$\{\mathbf{a}_8^{(2)}, \mathbf{a}_8^{(3)}\}$	1.4

As seen in Table 5.1, there were large differences in weight of the 8 vector spaces.

The first vector space $\{\mathbf{a}_1^{(2)}, \mathbf{a}_1^{(3)}\}$ had clearly the largest weight (62.5). As mentioned above, this vector space represented the mean structure of the data. It is thus expected that this vector space should have great weight, as the data normalization in the original tensor resulted in positive data centered around 1. To reproduce the original tensor from the decomposition, an average is clearly beneficial to make up for this centering that differed from zero.

The weights of the other vector spaces varied between 1.4 and 44.7. We argue that our proposed similarity measure is hence more representative of the data, compared to solely looking at the latent factor matrix in the session dimension, assuming that each latent variable can be regarded as mutually important. This, as the vector spaces with larger weights contribute more to the approximation of the original tensor than those with small weights. Worth mentioning is that the weights of the CP decomposition are highly dependent on the normalization of data. This means that the weighting contribution will not have as large of an impact in the cases where the decomposition concludes in similar weights for all vector spaces.

An example of how the weighting factor affected the latent vector $\mathbf{v}_1^{(1)}$ (session 1) can be seen in Table 5.2:

Table 5.2 Latent vector $\mathbf{v}_1^{(1)}$ (session 1) before and after multiplied with weight and normalized to sum up to 1.0.

$l_r^{(n)}$	$l_1^{(1)}$	$l_2^{(1)}$	$l_3^{(1)}$	$l_4^{(1)}$	$l_5^{(1)}$	$l_6^{(1)}$	$l_7^{(1)}$	$l_8^{(1)}$	**Sum**
$\mathbf{v}_1^{(1)}$ w.o. weight	0.268	0.027	0.085	0.045	0.040	0.070	0.064	0.116	0.715
$\mathbf{v}_1^{(1)}$ w. weight	0.815	0.059	0.039	0.031	0.009	0.013	0.025	0.008	1.00

Looking at Table 5.2, the weighting factor enhanced some latent variable values and depressed others. Particularly large changes can be found in latent variables $l_1^{(1)}$, $l_6^{(1)}$ and $l_8^{(1)}$. Additionally, the normalization that made the latent vector sum up to 1.0, makes it possible to see how each latent variable value can be viewed as the session's percentage of variability.

The weighted latent vectors for all sessions from the CP decomposition can be seen in Table 5.3.

Table 5.3 Latent vectors $\mathbf{v}_s^{(1)}$, weighted and normalized (to sum up to 1.0), for all sessions s.

$\mathbf{v}_s^{(1)} \setminus l_r^{(n)}$	$l_1^{(1)}$	$l_2^{(1)}$	$l_3^{(1)}$	$l_4^{(1)}$	$l_5^{(1)}$	$l_6^{(1)}$	$l_7^{(1)}$	$l_8^{(1)}$
$\mathbf{v}_1^{(1)}$	0.816	0.059	0.039	0.031	0.009	0.013	0.025	0.008
$\mathbf{v}_2^{(1)}$	0.596	0.126	0.064	0.072	0.022	0.030	0.072	0.019
$\mathbf{v}_3^{(1)}$	0.869	0.043	0.028	0.025	0.011	0.005	0.013	0.006
$\mathbf{v}_4^{(1)}$	0.912	0.034	0.027	0.010	0.000	0.008	0.005	0.004
$\mathbf{v}_5^{(1)}$	0.699	0.096	0.052	0.051	0.014	0.024	0.049	0.016
$\mathbf{v}_6^{(1)}$	0.843	0.053	0.034	0.023	0.008	0.013	0.02	0.007
$\mathbf{v}_7^{(1)}$	0.849	0.047	0.035	0.027	0.011	0.008	0.016	0.007
$\mathbf{v}_8^{(1)}$	0.517	0.148	0.076	0.089	0.03	0.033	0.084	0.023

Studying Table 5.6, the greatest difference between latent vectors can be found in latent variable $l_1^{(1)}$. Since this vector space mainly represented the mean value of the data, it can be questioned if this latent variable/vector space should be included in the comparison between sessions or not. A difference in mean value could be an effect of difference in electrode impedance or general brain activity between sessions, which may or may not be of interest for a specific BCI problem. As the data normalization included individual adaption for each session, one can argue that the device differences during data collection was to a large extent compensated for already. Throughout this project, we have decided to keep the mean value component from the tensor decomposition when comparing the similarity between sessions, but we view it as an interesting future extension to exclude it.

Tucker Decomposition Selection Results For the chosen significance $\alpha = 0.95$, the rank of the Tucker decomposition was computed to $R_1 = 8$ in the session dimension, $R_2 = 14$ in the channel dimension and $R_3 = 20$ in the frequency dimension. Again, in the factor matrix $\mathbf{A}^{(1)}$ (the session dimension), the latent vectors $\mathbf{v}_s^{(1)}$ should be highlighted, as each vector represents one specific session. In the factor matrices $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$, the columns are of more interest as they make up the vector space that are to be combined and scaled by the latent variables in the session dimension and the core tensor.

A detailed study of the structures captured by the Tucker decomposition can be seen below. Figure 5.8 shows the columns of the channel factor matrix $\mathbf{A}^{(2)}$, and Figure 5.9 shows the columns of the frequency factor matrix $\mathbf{A}^{(3)}$.

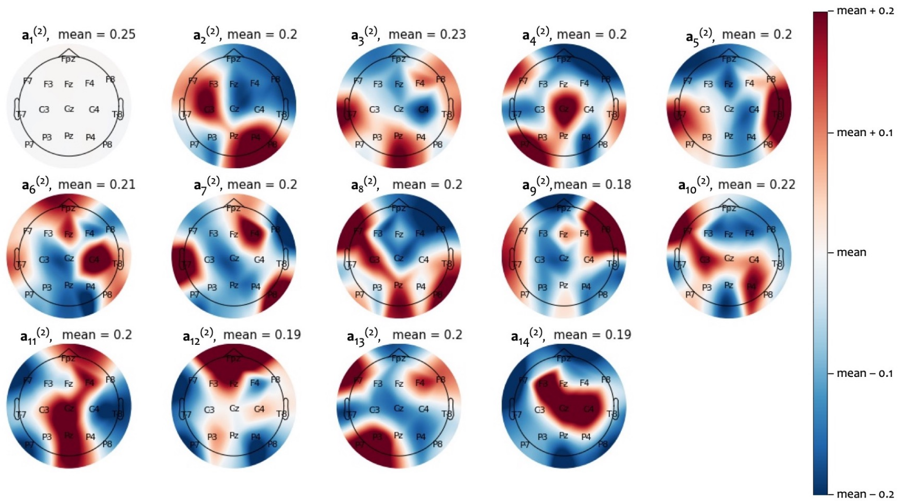


Figure 5.8 The columns $\mathbf{a}_{r_2}^{(2)}$ $(r_2 = 1, \ldots, 14)$ corresponding to the factor matrix $\mathbf{A}^{(2)}$ in the Tucker decomposition. The colors of the topomaps represent the deviation from the column's mean value (see colorbar).

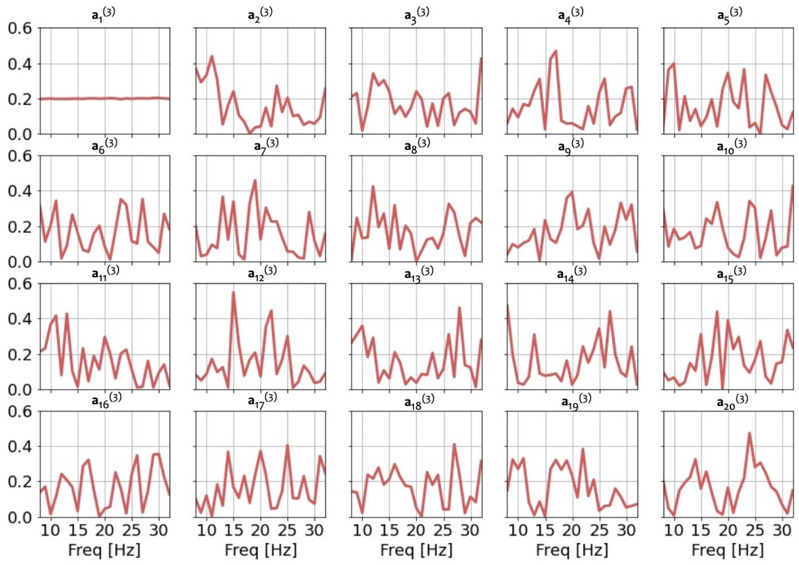


Figure 5.9 The columns $\mathbf{a}_{r_3}^{(3)}$ $(r_3 = 1, \ldots, 20)$ corresponding to the factor matrix $\mathbf{A}^{(3)}$ in the Tucker decomposition.

As for the CP decomposition, the columns in each latent factor matrix $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$ describe different structures in the data. The latent variables in the session dimension can again be viewed as *scaling* of the columns in $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$, but without the constraint of only being combined with a single column in each matrix (because of the core tensor).

Again, the mean structure of the data has been captured by columns $\mathbf{a}_1^{(2)}$ and $\mathbf{a}_1^{(3)}$. Apart from that, it is clear that all columns enhance different variability of the data in terms of channel location and frequency peaks. As a combination of all channels and frequency columns are used to recreate the data of each session, no evident channel-frequency pair can be viewed as "belonging together", in contrast to the CP decomposition. Still, channels and frequencies of more relevance to this classification problem, compared to the data exploration, can be identified (for example channel columns $\mathbf{a}_4^{(2)}$ and $\mathbf{a}_{14}^{(2)}$, and frequency columns $\mathbf{a}_2^{(3)}$ and $\mathbf{a}_{12}^{(3)}$).

The weighting factor in the Tucker decomposition, calculated as the norm of the 8 slices after multiplying the core tensor $\mathcal{G}$ with factor matrices $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$, can be seen in Table 5.4:

Table 5.4 The 8 weighting factors of the Tucker decomposition, calculated as the norm of each slice after multiplying the core tensor $\mathcal{G}$ with factor matrices $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$.

Vector Space	Weight
$\|\|\mathbf{S}_1\|\|_F$	57.1
$\|\|\mathbf{S}_2\|\|_F$	0.80
$\|\|\mathbf{S}_3\|\|_F$	0.76
$\|\|\mathbf{S}_4\|\|_F$	0.76
$\|\|\mathbf{S}_5\|\|_F$	0.73
$\|\|\mathbf{S}_6\|\|_F$	0.64
$\|\|\mathbf{S}_7\|\|_F$	0.64
$\|\|\mathbf{S}_8\|\|_F$	0.57

As for the CP decomposition, the slice representing the mean structure of the data had the largest weight (slice $\mathbf{S}_1$). The fact that this slice describes the mean structure can be seen as it includes the combination of latent variable columns $\mathbf{a}_1^{(2)}$ and $\mathbf{a}_1^{(3)}$, and will be combined with the latent variable $l_1^{(1)}$. All other slices had similar weights (between 0.57 to 0.8), implying the proposed similarity measure did not affect their relation as much.

An example of how the weighting factor affected the latent vector $\mathbf{v}_1^{(1)}$ (session 1) can be seen in Table 5.5:

Table 5.5 Latent vector $\mathbf{v}_1^{(1)}$ (session 1) before and after multiplied with weight and normalized to sum up to 1.

$l_r^{(n)}$	$l_1^{(1)}$	$l_2^{(1)}$	$l_3^{(1)}$	$l_4^{(1)}$	$l_5^{(1)}$	$l_6^{(1)}$	$l_7^{(1)}$	$l_8^{(1)}$	**Sum**
$\mathbf{v}_1^{(1)}$ w.o. weight	0.329	0.129	0.120	0.438	0.047	0.272	0.234	0.632	2.20
$\mathbf{v}_1^{(1)}$ w. weight	0.938	0.005	0.005	0.017	0.002	0.009	0.007	0.018	1.00

Studying Table 4.2, the weighting factor enhanced the first latent variable value $l_1^{(1)}$ and depressed the other ones. The normalization (making the latent vector sum up to 1.0) makes it possible to see how each latent variable value can be viewed as the session's percentage of variability.

The weighted latent vectors for all sessions from the Tucker decomposition can be seen in Table 5.6.

Table 5.6 Weighted and normalized (to sum up to 1.0) latent vectors $\mathbf{v}_s^{(1)}$ for all sessions s.

$\mathbf{v}_s^{(1)} \setminus l_r^{(n)}$	$l_1^{(1)}$	$l_2^{(1)}$	$l_3^{(1)}$	$l_4^{(1)}$	$l_5^{(1)}$	$l_6^{(1)}$	$l_7^{(1)}$	$l_8^{(1)}$
$\mathbf{v}_1^{(1)}$	0.938	0.005	0.005	0.017	0.002	0.009	0.007	0.018
$\mathbf{v}_2^{(1)}$	0.932	0.002	0.013	0.007	0.013	0.016	0.014	0.003
$\mathbf{v}_3^{(1)}$	0.954	0.035	0.003	0.001	0.003	0.001	0.003	0.001
$\mathbf{v}_4^{(1)}$	0.938	0.007	0.010	0.008	0.000	0.004	0.019	0.013
$\mathbf{v}_5^{(1)}$	0.937	0.002	0.018	0.024	0.003	0.008	0.002	0.007
$\mathbf{v}_6^{(1)}$	0.941	0.003	0.001	0.008	0.003	0.018	0.015	0.011
$\mathbf{v}_7^{(1)}$	0.935	0.006	0.022	0.013	0.017	0.001	0.002	0.003
$\mathbf{v}_8^{(1)}$	0.936	0.008	0.011	0.003	0.025	0.012	0.004	0.001

Again, latent variable $l_1^{(1)}$ was of most importance when describing the data of the original tensor. For the same reasons as the CP decomposition, it can be questioned if the combination of components that describes the mean value of the data should be included or not. Throughout this project, we have kept it when comparing similarity between sessions.

Selection Results The Euclidean distances between sessions were presented in normalized distance matrices, see Figure 5.10. For baseline comparison, the Riemannian distance between sessions were also presented in a normalized distance matrix in the same figure. A short distance (close to 0) implies similar sessions.

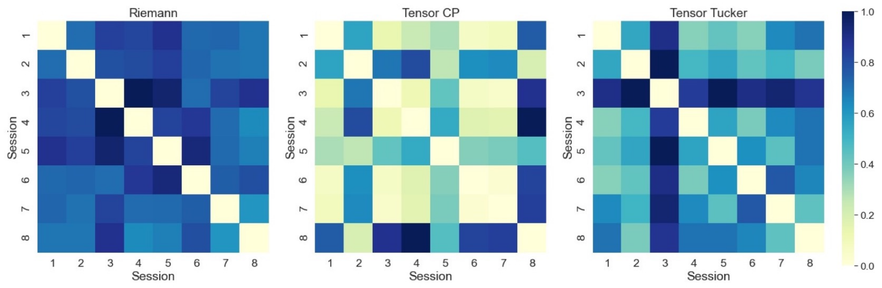


Figure 5.10 Normalized distance matrices for Riemann selection (left) and tensor selection (middle and right). An element d_{ij} answers to the similarity between sessions i and j.

From the distance matrices, it is clear that the three selection methods disagree on which sessions are (non-)similar. Additionally, in the Riemannian distance matrix the colors varied little (different shades of blue), implying all sessions were of relatively similar distance to each other. The tensor decomposition distance matrices had more color variation, meaning some sessions were a lot further away from each other than others.

Comparing the distance matrices in Figure 5.10 to the latent vectors in Table 5.3 and 5.6, it is clear that the mean value component has a great impact when it comes to distances between sessions. This is particularly evident for the sessions that are far away from all other sessions, for example session 8 in the CP decomposition and session 3 in the Tucker decomposition.

An example of which sessions were selected as most similar to session 1 can be seen in Table 5.7, corresponding to the shortest distance for Riemann and tensor selection. The table translates to the first row (or column) in the distance matrices in Figure 5.10.

Table 5.7 Similar session selection for session 1, from most to least similar.

Selection Method	**Selection (most to least similar)**
Riemann	[8, 2, 6, 7, 4, 3, 5]
Tensor CP	[6, 7, 3, 4, 5, 2, 8]
Tensor Tucker	[4, 6, 5, 2, 7, 8, 3]

Prediction Results The prediction accuracy for the three different classifiers (Linear SVC, SVC rbf and Random Forest) and the four different session selection methods can be seen in Figure 5.11. Each method selected 1-7 training sessions, and all 8 sessions were selected as the target once (LOSO). The plotted accuracy was the average of the prediction accuracy for all 8 sessions.

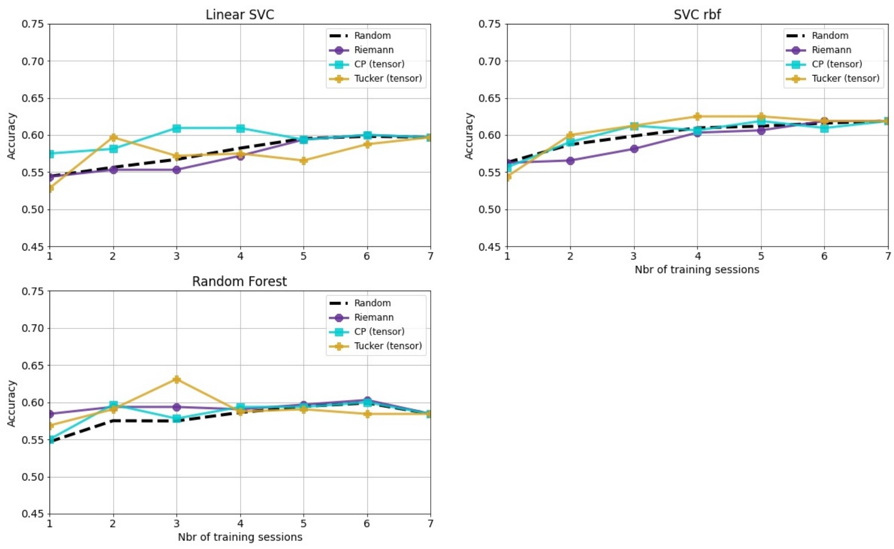


Figure 5.11 Average prediction accuracy for the three classifiers Linear SVC (upper left), SVC rbf (upper right) and Random Forest (lower left) for the four session selection methods.

As seen in Figure 5.11, the accuracy reaches at most barely 65%, which in comparison with today's state of the art results above 85% can be considered relatively poor [Zhang et al., 2021]. Furthermore, no clear effect in performance could be seen between the different selection methods. This is expected as the training database was small, meaning it was less likely there were any training session(s) similar to the target session. A more detailed evaluation of performance will be made for the larger dataset SA Drivers.

SA Driving

Data Exploration After applying the ANOVA filter during LOSO, it became evident that the frequency range 4-6 Hz was the most relevant for this classification problem. The most frequently occurring channels were the ones placed on the central/back rows of the head: T3, C3, Cz, C4, T4, TP7, CP3, CPz, CP4, TP8, T5, P3, Pz, P4 and T6. The proportion of this frequency band and channels of the total 10×33 features selected (10 each for the 33 subjects) were the following:

- Frequency range 4-6 Hz appeared in 100 % of the features, and was then 87 % of the cases in channels T3, C3, Cz, C4, T4, TP7, CP3, CPz, CP4, TP8, T5, P3, Pz, P4 and T6.

The corresponding frequency range for the two classes (averaged over all trials), was plotted in topomaps with all channels marked out. The result can be seen below in Figure 5.12. Additionally, the absolute difference PSD between the two classes is displayed to emphasize where the classes differ the most for this specific frequency range.

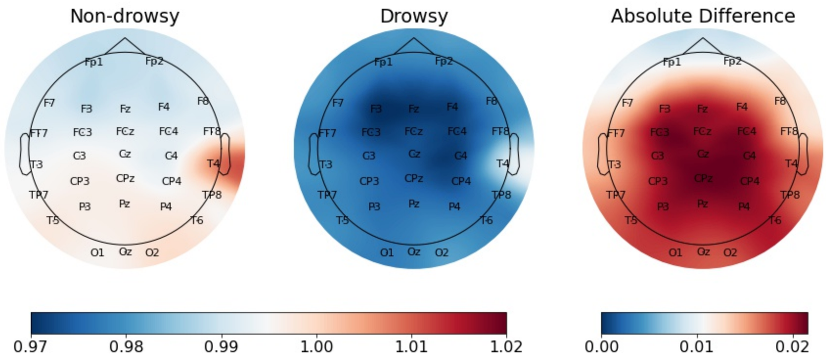


Figure 5.12 Average logarithmic normalized PSD for the frequency range 4-6 Hz for class non-drowsy (left), class drowsy (middle) and absolute difference between the two classes (right). Note that the normalization is such that low values correspond to higher PSD, and opposite.

As seen in Figure 5.12, the frequency range 4-6 Hz differed at almost all channels for the two classes non-drowsy and drowsy. The class drowsy had lower values, which correspond to higher power in the data. This was considered reasonable since lower frequencies are enhanced when the consciousness level decreases, as mentioned in Table 1.1 in Chapter 1 **Introduction**. Notable is that channel T4 in Figure 5.12 stood out, as it had higher values for both classes. No obvious reason why such a local activity on the side of the head occurred was found. It was therefore considered likely that channel T4 was an outlier during data collection. Furthermore, as the majority of the channels seemed to be of interest in this classification problem, a frequency-channel vector space adapted to this specific problem was harder to find than for the Alex MI dataset.

Selection Results For the chosen significance $\alpha = 0.95$, the rank of the CP decomposition was computed to $R = 19$. The rank of the Tucker decomposition was correspondingly $R_1 = 29$, $R_2 = 19$ and $R_3 = 22$. As for Alex MI, a detailed analysis of the structures captured by the CP/Tucker decomposition was made, but will not be as thoroughly discussed in this section. This is due to the fact that it was harder to filter out what channels that were of interest, and that the ranks of the decompositions were higher. Additionally, the high rank made the variety in structures greater, and in combination with the small prior knowledge of what frequency-channel pairs that could be of importance in this regression problem, it was hard to visually inspect which columns that were of interest. In summary, the behavior was similar to that of Alex MI in terms of the mean value of the data captured by the combination of latent variable $l_1^{(1)}$ in vector space $\{\mathbf{a}_1^{(2)}, \mathbf{a}_1^{(3)}\}$ for the CP decomposition, and in slice $\boldsymbol{S}_1$ for the Tucker decomposition. Additionally, there was no single frequency column $\mathbf{a}_r^{(3)}$ that captured the 4-6 Hz structure particularly well, but peaks around those frequencies could be found in several columns. All resulting plots of columns $\mathbf{a}_r^{(j)}$, $j = 2, 3$ and $r = 1...19$ in the CP decomposition, and columns $\mathbf{a}_{r_j}^{(j)}$, $j = 2, 3$, $r_2 = 1...19$ and $r_3 = 1...22$ in the Tucker decomposition can be found in the Appendix, Figures 8.1-8.4. The weights of the 19 different vector spaces $\{\mathbf{a}_r^{(2)}, \mathbf{a}_r^{(3)}\}$ from the CP decomposition, and 29 different slices $||\mathbf{S}_{r_1}||$ from the Tucker decomposition, followed the same behavior as for Alex MI. All weights contributed to the selection as they affected the impact of the latent variables. Again, vector space $\{\mathbf{a}_1^{(2)}, \mathbf{a}_1^{(3)}\}$ for CP and slice $\boldsymbol{S}_1$ for Tucker had the largest weights, and represented the mean behavior when combined with latent variable $l_1^{(1)}$.

The Euclidean distance between sessions for the two tensor decomposition methods can be seen in the normalized distance matrices in Figure 5.13. For baseline comparison, the Riemann distance between sessions was also presented in a normalized distance matrix in the same figure. A short distance (close to 0) implies similar sessions.

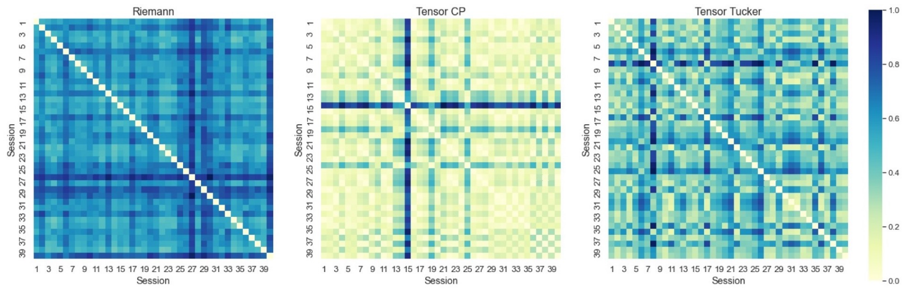


Figure 5.13 Normalized distance matrices for Riemann selection (left) and tensor selection (middle and right). An element d_{ij} answers to the similarity between sessions i and j.

Looking at the Riemannian distance matrix in Figure 5.13, the colors were all similar (different shades of blue), implying all sessions were of relatively similar distance to each other. The tensor decompositions show more color variations, meaning some sessions were regarded as a lot further away from each other than others. Additionally, the three different similarity measures did not agree on what sessions were similar/non similar. For instance, sessions 27, 29 and 40 were in general most non-similar to all other sessions in the Riemann selection method. In the CP decomposition method, session 15 instead stood out in this aspect. For the Tucker decomposition, single sessions standing out from the entire group were not as clearly seen (e.g. session 8 does not differ from *all* other sessions).

Prediction Results An example of the prediction results for the Random Forest regressor can be seen in Figure 5.14. The example shows true and predicted DI of 330 trials in session 19, when selecting a subset of 8 sessions during training for all four selection methods (random, Riemann, CP and Tucker). Note that this was one of the sessions with best performance overall, and where the tensor methods showed superiority compared to random.

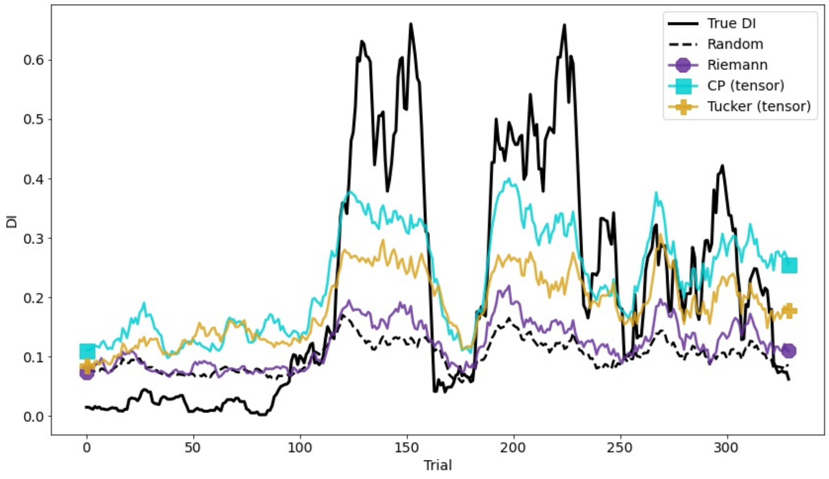


Figure 5.14 Example of the prediction results for session 19 with a Random Forest regressor, trained with a subset of 8 sessions. The black line corresponds to the true DI, which varies between 0 and 0.65 throughout the session. All other lines correspond to regressor results from prediction, were the subset of 8 sessions used for training has varied between the methods. As seen in the figure, the result from the CP and Tucker selection methods look more similar to the true DI, both in terms of amplitude and increasing/decreasing trends, compared to random and Riemann. See legend for colors/symbols corresponding to each selection method.

The corresponding performance measurements (Pearson correlation and MAE) for the results in Figure 5.14, can be seen in in Table 5.8:

Table 5.8 Performance measurements (Pearson correlation and MAE) for the results of session 19 in Figure 5.14. Note that this was one of the sessions with best performance.

	Random	**Riemann**	**CP (tensor)**	**Tucker (tensor)**
Correlation	0.781	0.716	0.837	0.816
MAE	0.158	0.139	0.100	0.113

As seen in Figure 5.14 and Table 5.8, the prediction and performance were affected by the selection methods. In this particular case, the tensor decomposition selection methods (CP and Tucker) had the best results, while random and Riemann selection had the worst, both in terms of Pearson correlation and MAE.

The two evaluation methods (correlation and MAE) capture different characters of the prediction, which becomes evident when comparing the results for random and Riemann prediction in Table 5.8. The random selection method resulted in higher correlation, while the Riemann method resulted in lower MAE. In a BCI application, it is reasonable that both measurements can be of interest, which will be argued for next.

The correlation evaluates how well trends of fluctuations in drowsiness are captured. This would be of high interest in drowsiness detection, as a high correlation makes it possible to recreate the true DI by rescaling. Although, to find the correct rescaling of a new session, this would require some initial calibration. Initial calibration is what to a large extent is aimed to be avoided in transfer learning applications, which hence is a disadvantage with this evaluation measurement.

The MAE on the other hand is a good way of evaluating how useful the regressors are without any extra calibration. The downside with this measurement is that it benefits a DI series that does not vary much over time, which clearly makes the regression task unequally hard for different sessions. The prediction results for all sessions, corresponding to the results in Figure 5.14, can be found in Figure 8.5 in the Appendix. From Figure 8.5, the variation in difficulty level of different sessions from an MAE perspective is clearly seen, as some sessions for example always have DIs below 0.3 while others often vary between 0 to above 0.6.

The average performance results for the full database is presented in Figures 5.15 and 5.16. The three different regressors (Linear SVR, SVR rbf and Random Forest), and four different session selection methods (random, Riemann, CP and Tucker) are plotted together for comparison. For each selection method, a subset of 1-37 sessions was selected for training, and all 40 sessions were selected as the target session once (LOSO).

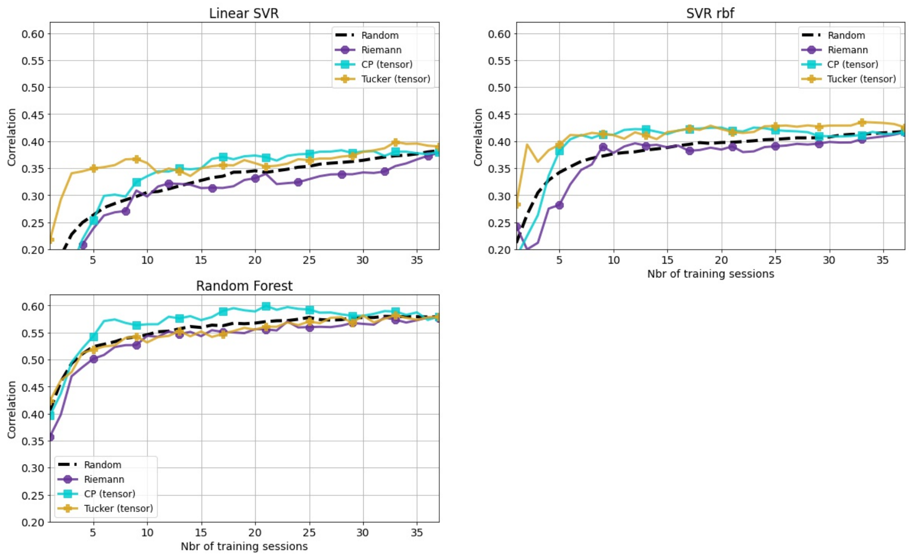


Figure 5.15 Average Pearson correlation plotted against the number of sessions selected as the training subset. All three regressors Linear SVR (upper left), SVR rbf (upper right) and Random Forest (lower left) for the four different session selection methods are shown. The two SVRs reached a maximum average performance of around 0.4, while the Random Forest regressor showed its superiority by reaching almost 0.6. The results show that the two tensor selection methods (CP and Tucker) on average performed better than random/Riemann selection for the SVRs, for most subsets of sessions. For the Random Forest regression, the CP decomposition selection method performed best. See legends for colors/symbols corresponding to each selection method.

Studying Figure 5.15 above, the CP and Tucker decomposition selection methods often resulted in higher correlation than random and Riemann selection (on average). This was particularly evident for a subset of 5-10 sessions. For the Linear SVR, the Tucker decomposition initially stood out, as it on average increased the correlation with 5-12 percentage units when the number of training sessions was less than 10 (compared to random). For the SVR with an rbf kernel, both the CP and Tucker decompositions increased the correlation on average, for most subsets of sessions. The increase was relatively similar and constant for a small range of subsets; for 5-12 training sessions around 5 percentage units better than random. For the random forest regressor, the CP decomposition selection method stood out, as it on average increased the correlation with 2-5 percentage units throughout the interval 5-24 training sessions.

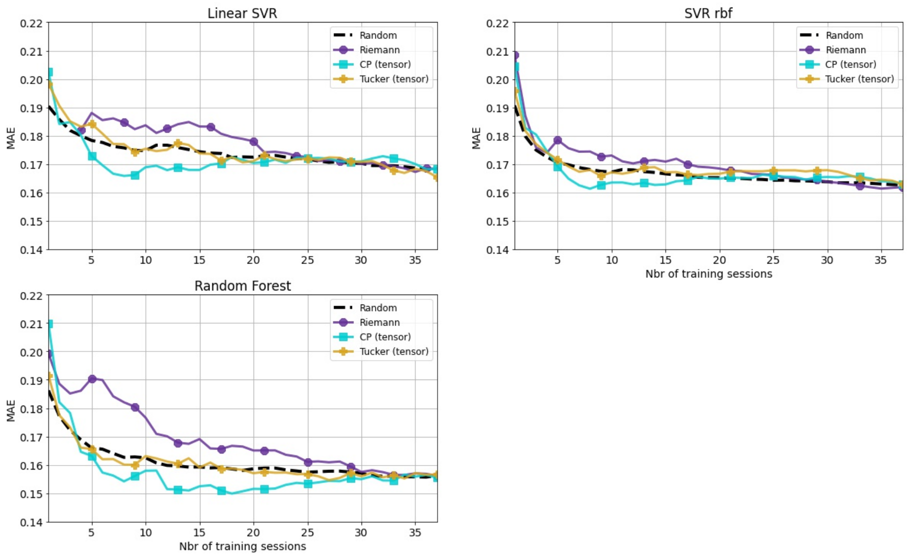


Figure 5.16 Average MAE performance against number of sessions selected as the training subset. All three regressors Linear SVR (upper left), SVR rbf (upper right) and Random Forest (lower left) for the four different session selection methods are shown. The three regressors reached minimum MAEs of 0.15-0.17. For all three regressors, the CP decomposition selection method performed best by reaching a minimal MAE for a small subset of training sessions. See legend for colors/symbols corresponding to each selection method.

Looking at the MAE results above, the CP decomposition selection method performed best for all regressors. In particular, for the two SVRs, the minimum was found for a subset of 8 training sessions, and for the Random Forest regressor, for a subset of 18 training sessions. For all three regressors, these minimas were lower than training with the full training database.

From Figures 5.15 and 5.16, it can be seen that there, in general, was a relatively large difference in performance between the regressors. The Random Forest regressor clearly stood out, as it reached both higher correlation and lower MAE than the two SVRs, almost already from a subset of 1 session during training. The goal of the regressors was to find distinctions in characteristics between trials with different DIs. The results indicate that the Random Forest regressor, which utilizes recursive binary splittings, captures such distinctions better than the two SVRs do. As EEG signals measure the macroscopic, complex behavior of the brain, it is reasonable that the differences between drowsy and non-drowsy trials are most efficiently separated in a nonlinear manner. Worth noting is that both tensor decomposition methods, that were used to select sessions, were linear. It may be of interest to instead consider a nonlinear tensor decomposition when comparing similarity of sessions, as nonlinear relationships in the data look to be of more use in a regressor. However, it is not evident from the figures above that these linear tensor decompositions were more beneficial for the linear SVR, as one then could have expected, than for the other two regressors. A non-linear tensor decomposition extension will be further discussed in Chapter 7 **Conclusions and Future Work**.

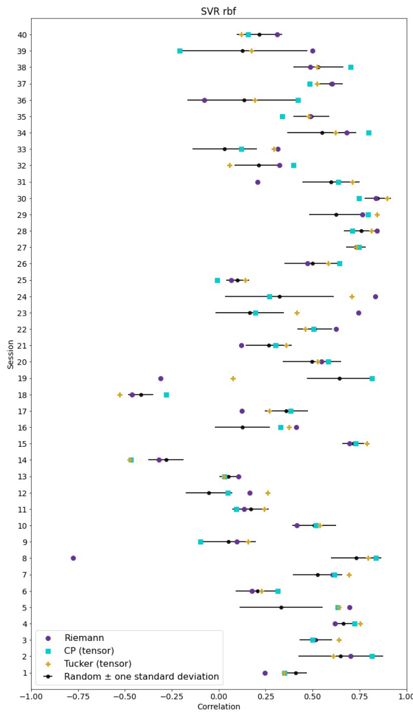
When comparing the regressors' performances, one needs to account for a particular factor. As mentioned in the method description, minimal amount of tuning was performed. This was due to the computational expense, and to avoid benefiting one or several selection methods. It is therefore possible that the SVRs could reach the same performance as the Random Forest regressor if more time would have been spent on tuning. However, it could also be the case that the performance of the Random Forest regressor could be increased as well, creating an even greater difference between the regressors.

Notable is that in both Figures 5.15 and 5.16, the Riemann selection method did not outperform random. On the contrary, it often performed worse than random. This was not expected and does not agree with the results in [Jeng et al., 2021] and the EEG classification field in general. The reason for these results is probably that little adaptation in the pipeline was made for the Riemann selection method. For instance, the data normalization was mainly done to suit the non-negative tensor decompositions, while the Riemann method may have benefited from a different choice. As a future extension, it could be interesting to improve the Riemann selection method to be able to better compare the tensor methods to a "state of the art" method.

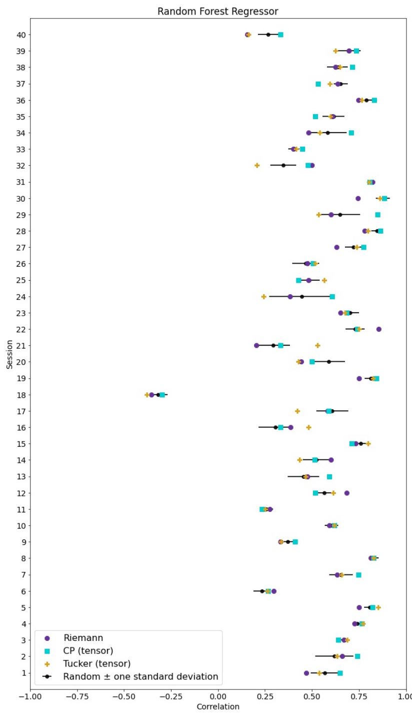
Even though the tensor selection methods often outperformed random on average, it was found that there was a big uncertainty in the results. This was mainly due to two reasons, which made it questionable both if the regressors could be considered reasonable to use in a future BCI application, and how certain one can be that the selection methods performed better than random. To visualize these two uncertainties, two of the most promising regressor-number of training session-combinations in Figures 5.15 and 5.16 were studied in detail:

1. The SVR with an rbf kernel when selecting 8 training sessions.
2. The Random Forest regressor when selecting 18 training sessions

Combination 1 yielded a clear increase in average correlation performance compared to random selection for the CP and Tucker methods, and in the MAE performance for the CP method. Additionally, it used a relatively small subset of sessions for training, speeding up the training process. Combination 2 yielded a clear increase in average Pearson correlation and MAE performance compared to random for the CP selection method. Additionally, combination 2 resulted in one of the overall best performances achieved (correlation 0.595 and MAE 0.150) compared to all other regressors and selection methods. Error plots of the two combinations, visualizing the performance results for all sessions compared to random, can be seen in Figures 5.17 (correlation) and 5.17b (MAE).



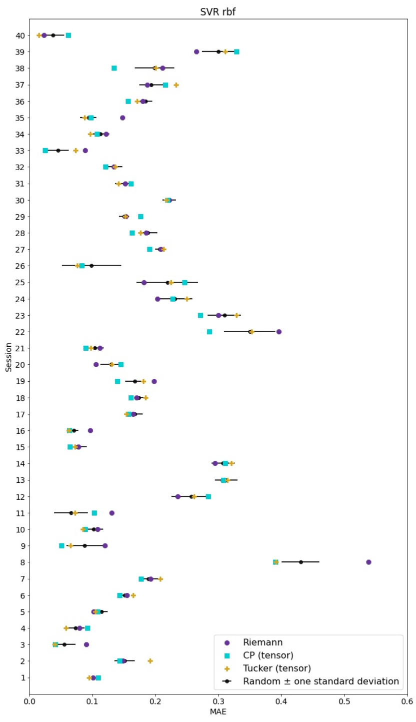
(a) Combination 1.



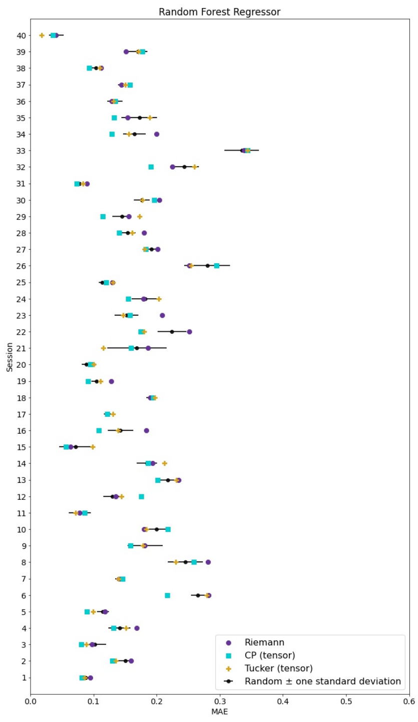
(b) Combination 2.

Figure 5.17 Error plots of two promising regressor and number of training sessions-combinations. Each error bar represents the mean correlation for a session and its standard deviation during random selection (which was performed 20 times). The markers show the resulting correlation when using the selection methods to find a subset of sessions for training.

As seen in Figure 5.17, the Pearson correlation performance varied extensively between sessions. Looking only at the results from random selection (black dot and error lines) in Figure 5.17a, the mean correlation value for each session was between -0.42 and 0.85, and in Figure 5.17b it was correspondingly between -0.32 and 0.88. This means that for a completely new, unseen test session, it will be hard to know if the performance will be sufficiently good for a BCI. This of course depends on the performance requirements, but a correlation close to or less than 0 would definitely not be useful. As seen in Figure 5.17, occasionally, a session had correlation below or close to 0, even for the high-performing combination 2.



(a) Combination 1.



(b) Combination 2.

Figure 5.18 Error plots of two promising regressor-number of training sessions-combinations. Each error bar represents the mean MAE for a session and its standard deviation during random selection (which was repeated 20 times). The markers show the resulting MAE when using the selection methods to find a subset of sessions for training.

Studying Figure 5.18, the MAE performance also varied significantly between sessions. Notable was that the sessions with high MAE (bad performance) were usually not the same as those with low correlations. For example, session 18 had the clearly lowest correlation for the Random Forest regressor, while it had close to average MAE. Instead, session 8 stood out as the one with highest MAE, but with also high correlation on average. This is most likely due to the fact that the regression problem was not equally simple for all sessions in terms of MAE evaluation, as discussed earlier. For a session that never reached high DIs, it was easier to achieve a low MAE.

Comparing the two regressors both in terms of correlation and MAE, it can be seen that the standard deviations for every session (length of black error lines) were higher in the SVR rbf than for Random Forest. The main reason for this was that a bigger subset of training sessions was used in the Random Forest regressor example (18 vs 8). The probability of selecting useful sessions increased as the subset size grew and thus, made the standard deviation decrease.

As seen in the Figures 5.17 and 5.18, the selection methods were not consistent between sessions in terms of performance increase/decrease. The Riemannian, CP, and Tucker selection methods could be found to result in performance far above, far below or close to the mean of a specific sessions, for both the SVR and Random Forest case. To summarize the performance change for these two combinations of regressors/number of training sessions, an increase count was created and presented in Tables 5.9 and 5.10.

Table 5.9 Count and % of sessions that performed better than random for combination 1.

	Combination 1					
	Pearson Correlation			MAE		
	Riemann	**CP**	**Tucker**	**Riemann**	**CP**	**Tucker**
# better than random	20	25	27	19	27	19
% better than random	50%	62.5%	67.5%	47.5%	67.5%	47.5%

Table 5.10 Count and % of sessions that performed better than random for combination 2.

	Combination 2					
	Pearson Correlation			MAE		
	Riemann	**CP**	**Tucker**	**Riemann**	**CP**	**Tucker**
# better than random	12	27	19	16	24	18
% better than random	30%	67.5%	47.5%	40%	60%	45%

As seen in Tables 5.9 and 5.10, there were no evident results for a specific selection method in terms of selecting better sessions than random. The CP selection method stood out as the best and most likely for a new, unseen session to perform better than random, but still only with 60-67.5% likelihood. The Riemann method was often outperformed by random, and the Tucker method was usually similar to random.

Simply counting the results in terms of "better or worse" than random can to some extent be regarded as incomplete. It is also of interest to what extent the selection methods made the performance increase/decrease. Consider for example, a case where 50 % of the sessions had 0.1 percentage units worse performance than random, and the other 50% had 20 percentage units better. Clearly, for a new session, it would be more reasonable to use this selection method, as there is a lot to win from a gain, and far less to loose. For a case like this, one could approximate all sessions with a distribution and weight the change in performance differently. This could for example be done by looking at each result's p-value, and combining them to get a more representative idea of the significance in increase of performance. As seen in Figures 5.17 and 5.18, the selection methods could for some sessions result in a lot better performance results than random. But as the performance result almost as often was a lot worse than random, and it was not evident what type of distribution every session should be approximated as when analyzing their histograms, such an evaluation was not made.

Looking closer at Figures 5.17 and 5.18, unique sessions with a lot better/worse performance for the CP or Tucker decomposition methods than random selection can be identified. The results can be compared to the distance matrices of the selection methods in Figure 5.13, to see if these sessions stood out in any way. No such connection was found, but further analysis could be an interesting future extension to this work.

Another thing to consider is if using a subset of training sessions can achieve the same, or even better performance, compared to training with the full training database. From Figures 5.15 and 5.16, there are consistent indications that the initial increase in performance appears faster as the number of training sessions increases for the tensor methods, compared to random selection. Additionally, the peak performance sometimes appears for a smaller subset of training sessions than when training with the full database (see for example, the Random Forest regressor). This result can be combined with the differences in time for training with a subset of sessions versus the full training database. The fastest regressor was the Linear SVR, which for each session took approximately 50 seconds when training with the full database. For a subset of 10 sessions, it took around 1/3 of that time. Second was the SVR rbf, which correspondingly took about 5 minutes when training with the full database. For a subset of 10 sessions, it took around 1/8 of that time. Lastly, the most time consuming regressor was the Random Forest, which took around 20 minutes for each session with the full database. For a subset of 10 sessions, it took around 1/5 of that time. This highlights the relevance of finding a method that can find appropriate subsets of data to use instead of the full database, and the selection method's future potential despite its current uncertainties.

6

Tensor Decompositions for Feature Extraction and Session Selection

Using a tensor decomposition as a lower dimensional approximation of the original tensor, the decomposition can be viewed as features of the data. During the past twenty years, various papers have investigated the power of tensor decomposition for precisely this; *feature extraction*. This chapter answers to our second pipeline; how tensors can be used for both session selection and feature extraction. Unlike the previous chapter, tensor representations are not only used to measure the similarity between sessions, but also to compute features of the data.

Section 6.1 **Background** presents an overview and explanation of the approach, inspired by the work in *Tensor Decompositions for Feature Extraction and Classification of High Dimensional Datasets* [Phan and Cichocki, 2010]. Next, 6.2 **Our Contribution** accounts for our extensions to their method, followed by a detailed description of our approach in 6.3 **Method**. Finally, the results are presented and evaluated in 6.4 **Result and Discussion**.

6.1 Background

The approach in this section was based on the article *Tensor Decompositions for Feature Extraction and Classification of High Dimensional Datasets* [Phan and Cichocki, 2010]. Their idea was to compute a mutual feature space from the Tucker decomposition, where every trial answered to a location in this space. In the report, they tested their approach for binary motor imagery classification (left or right hand) using 64-channel EEG recordings, and splitting the trials of each session into training and test data[1]. Since our purpose was to investigate how tensor decompositions can be used for transfer learning between sessions, our approach slightly differed from theirs. The next section 6.1 **The Tucker Feature Space**, includes how to compute the mutual feature space as made by Phan and Chichocki. The following section 6.2 **Our Contribution** accounts for how we have applied their method for transfer learning purposes and chosen to interpret the feature space.

The Tucker Feature Space

As presented in Section 2.2 **The Tucker Decomposition**, the Tucker decomposition returns a set of factor matrices and a core tensor. The factor matrices $(\boldsymbol{A}^{(1)}, \dots, \boldsymbol{A}^{(N)})$ can be viewed as the basis of a lower dimensional space in which the core lies. Assuming the core captures characteristics of the original data, it can be translated as features of $\boldsymbol{\mathcal{X}}$. Identifying mutual factor matrices for several input tensors can thus be seen as identifying a common feature space for a dataset.

Assume a dataset consists of M trials, where every trial can be represented as a tensor $\boldsymbol{\mathcal{X}}^{(m)}$. It can for example be a set of EEG data recordings, where a trial is represented by the frequency in every channel over time. The data thereby consists of M trials, answering to M tensors of shape (channel×frequency×time). Computing a Tucker decomposition where the factor matrices are restricted to be identical for all trials, while the cores are allowed to differ, yields:

$$\begin{aligned}
\boldsymbol{\mathcal{X}}^{(1)} &\approx \boldsymbol{\mathcal{G}}^{(1)} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)} \times_3 \boldsymbol{A}^{(3)} \\
\boldsymbol{\mathcal{X}}^{(2)} &\approx \boldsymbol{\mathcal{G}}^{(2)} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)} \times_3 \boldsymbol{A}^{(3)} \\
&\vdots \\
\boldsymbol{\mathcal{X}}^{(M)} &\approx \boldsymbol{\mathcal{G}}^{(M)} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)} \times_3 \boldsymbol{A}^{(3)}
\end{aligned} \tag{6.1}$$

[1] The results showed an accuracy above 85%. For details, we refer to their report [Phan and Cichocki, 2010].

or for general tensors of order N:

$$\begin{aligned}
\boldsymbol{\mathcal{X}}^{(1)} &\approx \boldsymbol{\mathcal{G}}^{(1)} \times_1 \boldsymbol{A}^{(1)} \times_2 \cdots \times_N \boldsymbol{A}^{(N)} \\
\boldsymbol{\mathcal{X}}^{(2)} &\approx \boldsymbol{\mathcal{G}}^{(2)} \times_1 \boldsymbol{A}^{(1)} \times_2 \cdots \times_N \boldsymbol{A}^{(N)} \\
&\vdots \\
\boldsymbol{\mathcal{X}}^{(M)} &\approx \boldsymbol{\mathcal{G}}^{(M)} \times_1 \boldsymbol{A}^{(1)} \times_2 \cdots \times_N \boldsymbol{A}^{(N)}
\end{aligned} \tag{6.2}$$

The factor matrices that solve (6.2) constitute the basis of the feature space in which the features (the core) of every trial lie. To calculate the cores $\boldsymbol{\mathcal{G}}^{(m)}$ and factor matrices $\boldsymbol{A}^{(1)},\ldots,\boldsymbol{A}^{(N)}$, one must compute the Tucker decompositions under the restriction of identical factor matrices. To apply this restriction, one can concatenate all trials along an additional $(N+1)$th mode, forming a new tensor $\boldsymbol{\mathcal{X}}_{N+1} \in \mathbf{R}^{I_1\times\cdots\times I_N\times M}$. Fixing this tensor along the $(N+1)$th mode and extracting the elements along index m, one obtains the elements of the mth trial. Unfolding $\boldsymbol{\mathcal{X}}_{N+1}$ along mode $(N+1)$, every trial translates to a row $\text{vec}(\boldsymbol{X}^{(m)})^T$ in the matrix $\boldsymbol{X}_{(N+1)}$:

$$\boldsymbol{X}_{(N+1)} = \left[\text{vec}(\boldsymbol{\mathcal{X}}^{(1)}) \quad \text{vec}(\boldsymbol{\mathcal{X}}^{(2)}) \quad \cdots \quad \text{vec}(\boldsymbol{\mathcal{X}}^{(M)})\right]^T \tag{6.3}$$

To compute a common feature space for all trials $\boldsymbol{\mathcal{X}}^{(1)},\ldots,\boldsymbol{\mathcal{X}}^{(M)}$, one calculates the Tucker decomposition of $\boldsymbol{\mathcal{X}}_{N+1}$:

$$\boldsymbol{\mathcal{X}}_{N+1} \approx \boldsymbol{\mathcal{G}}_{N+1} \times_1 \boldsymbol{A}^{(1)} \times_2 \cdots \times_N \boldsymbol{A}^{(N)} \tag{6.4}$$

where the resulting core tensor $\boldsymbol{\mathcal{G}}_{N+1}$ has the shape $(R_1 \times \cdots R_N \times M)$. Note that the factor matrix of the final mode $N+1$ is excluded, meaning the trials-dimension of the core tensor is the same as for the original tensor. By (6.4), every trial is approximated by a Tucker decomposition and (6.3) can be written as:

$$\begin{aligned}
\boldsymbol{X}_{(N+1)} &= \left[\text{vec}(\boldsymbol{\mathcal{X}}^{(1)}) \quad \text{vec}(\boldsymbol{\mathcal{X}}^{(2)}) \quad \cdots \quad \text{vec}(\boldsymbol{\mathcal{X}}^{(M)})\right]^T \\
&= \left[\text{vec}(\boldsymbol{\mathcal{G}}^{(1)}) \quad \text{vec}(\boldsymbol{\mathcal{G}}^{(2)}) \quad \cdots \quad \text{vec}(\boldsymbol{\mathcal{G}}^{(M)})\right]^T (\boldsymbol{A}^{(N)} \otimes \cdots \otimes \boldsymbol{A}^{(2)} \otimes \boldsymbol{A}^{(1)})^T \\
&= \boldsymbol{G}_{(N+1)} (\boldsymbol{A}^{(N)} \otimes \cdots \otimes \boldsymbol{A}^{(2)} \otimes \boldsymbol{A}^{(1)})^T
\end{aligned} \tag{6.5}$$

where it has been utilized that the n-mode product between a core $\boldsymbol{\mathcal{G}}^{(m)}$ and the factor matrices $\boldsymbol{A}^{(1)},\ldots,\boldsymbol{A}^{(N)}$ translates to the Kronecker product when unfolding $\boldsymbol{\mathcal{X}}^{(m)}$. From (6.5), it can be seen that the core $\boldsymbol{\mathcal{G}}_{N+1}$ and $\boldsymbol{\mathcal{X}}_{N+1}$ share the same structure; fixing $\boldsymbol{\mathcal{G}}_{N+1}$ along the $(N+1)$th mode and extracting the elements along index m, they answer to the features of the mth trial. In the same manner, the mth row of $\boldsymbol{G}_{(N+1)}$ represents the features of trial m.

For a dataset of M trials, one can thus simply compute the Tucker decomposition of the concatenated tensor $\boldsymbol{\mathcal{X}}_{N+1}$. Using the concatenated core $\boldsymbol{\mathcal{G}}_{N+1}$, one can extract the features $\boldsymbol{\mathcal{G}}^{(1)}, \dots, \boldsymbol{\mathcal{G}}^{(M)}$ of all trials, and from these train a classifier. When predicting the label of a new trial $\boldsymbol{\mathcal{X}}^{(m+1)}$, one simply uses the factor matrices to map the trial to the mutual feature space:

$$\boldsymbol{\mathcal{G}}^{(m+1)} = \boldsymbol{\mathcal{X}}^{(m+1)} \times_1 \boldsymbol{A}^{(1)^T} \times_2 \cdots \times_N \boldsymbol{A}^{(N)^T} \tag{6.6}$$

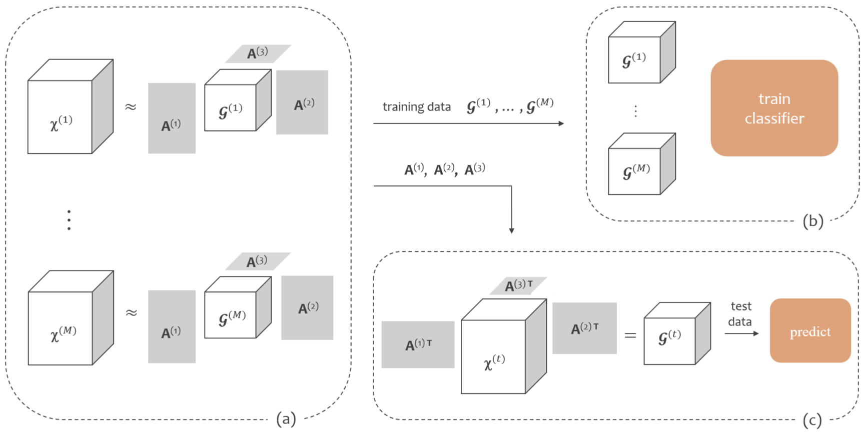
after which the features $\boldsymbol{\mathcal{G}}^{(m+1)}$ are the input to the trained classifier. Note that the expression for computing $\boldsymbol{\mathcal{G}}^{(m+1)}$ answers to the optimal solution presented in 2.2 **Computing the Tucker Decomposition**, see (2.16).

Computing the Feature Space To calculate the Tucker decomposition of the concatenated tensor $\boldsymbol{\mathcal{X}}_{N+1}$, Phan and Chichocki proposed using approaches such as HOOI, or ALS. It should be mentioned though, that these algorithms require a modification for the purpose. By default, HOOI and ALS form one factor matrix for every mode of the tensor. For the concatenated tensor $\boldsymbol{\mathcal{X}}_{N+1}$, we only seek the factors of the N first modes and thus want to ignore mode $N+1$. This, as we want the interaction within the data and not between the trials, which are represented by the $N+1$ mode. Recalling the updating rule for HOOI and ALS (see (2.19) and (2.8) respectively), this is easily accounted for by excluding the n-mode multiplication with the final factor matrix; what should have been $\boldsymbol{A}^{(N+1)}$. One thereby iterates from 1 to N, computing the latent factor matrices and excluding the final mode $N+1$:

$$\|\boldsymbol{A}^{(n)^T}\boldsymbol{W}\|^2 \text{ with } \boldsymbol{W} = \boldsymbol{X}_{(n)}(\boldsymbol{A}^{(N)} \otimes \cdots \otimes \boldsymbol{A}^{(n-1)} \otimes \boldsymbol{A}^{(n+1)} \otimes \cdots \otimes \boldsymbol{A}^{(1)}) \tag{6.7}$$

where $\boldsymbol{X}_{(n)}$ is the concatenated tensor $\boldsymbol{\mathcal{X}}_{N+1} \in \mathbf{R}^{I_1 \times \cdots \times I_N \times M}$, unfolded along the nth mode. The factor matrix $\boldsymbol{A}^{(n)}$ is then computed as the R_n most significant left singular vectors of $\boldsymbol{W}$.

To summarize this section, Figure 6.1 below displays a visualization of the approach:



$\mathcal{X}^{(1)}$ ≈ $A^{(1)}$ $\mathcal{G}^{(1)}$ $A^{(2)}$ $A^{(3)}$
$\mathcal{X}^{(M)}$ ≈ $A^{(1)}$ $\mathcal{G}^{(M)}$ $A^{(2)}$ $A^{(3)}$
(a)
training data $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(M)}$
$A^{(1)}, A^{(2)}, A^{(3)}$
$\mathcal{G}^{(1)}$
$\mathcal{G}^{(M)}$
train classifier
(b)
$A^{(1)T}$ $\mathcal{X}^{(t)}$ $A^{(2)T}$ $A^{(3)T}$ = $\mathcal{G}^{(t)}$
test data
predict
(c)

Figure 6.1 (a) Phan and Chichocki's approach for computing a feature space from the Tucker decomposition; concatenating the tensors of all M training trials and from this, calculating the Tucker decomposition. (b) Training a classifier from the computed features; the core tensors of each trial. (c) Predicting for a (new) test trial $\boldsymbol{\mathcal{X}}^{(t)}$ by projecting it to the computed feature space and extracting its core tensor, which is the input to the (trained) classifier.

6.2 Our Contribution

EEG Data Normalization

In the article *Tensor Decompositions for Feature Extraction and Classification of High Dimensional Datasets*, [Phan and Cichocki, 2010] calculate the *time frequency representation* (TFR) of each EEG trial, forming a fourth order tensor (#trials) × (#channels) × (#freqs) × (#time bins). The TFR was computed (with Morlet Wavelets) and resulted in a frequency range from 8 to 30Hz with increments of 1Hz, and a total of 50 time frames answering to 2 seconds of EEG data. Apart from this, no further normalization was performed.

To benefit the identification of possible linear relationships, and to subject a more well conditioned problem, we suggest normalizing the TFR of each session. Firstly, the TFR was computed using a multitaper method. Then, the normalization was done by applying the logarithm to every trial, dividing by the median of the corresponding session's pre-trials and lastly, subtracting the median of the pre-trials. This resulted in a tailored normalization for each session, where the data was centered around zero.

Computing the Feature Space

In *Tensor Decompositions for Feature Extraction and Classification of High Dimensional Datasets* by [Phan and Cichocki, 2010], a mutual feature space was computed from data belonging to one subject only. Consequently, the number of feature spaces equals the number of subjects, and they must create another feature space when predicting for a session that belongs to a new subject. For each subject, they computed the feature space from 60 training trials, leaving 120 trials as test data.

With the purpose of removing the subject-dependency, we propose applying transfer learning to this problem. When computing the Tucker decomposition to create a mutual feature space, we thereby use data from all subjects except that of the target session. Applying leave one session out (LOSO) cross validation, this results in one feature space per subject.

For the motor imagery problem, Phan and Cichocki trained their classifier using the core tensors of the training trials. To compute the features of a test trial, they projected its corresponding input tensor onto the mutual feature space, recall (2.16). The prediction was carried out using these features as input to the (trained) classifier.

For our approach, the features of a target session was calculated in the same manner; projecting the corresponding trials to the target session's feature space. When training the classifier though, we only use a subset of all training sessions, selecting the sessions most similar to the target session. To measure the similarity between sessions, we propose using the *distance between core tensors*. For each session, a mean core tensor was calculated from its pre-trials. The Euclidean distances between the mean cores was thereafter used as a similarity measure, meaning small values answered to high similarity. Note that the mutual feature spaces were essential in this matter, since the position of cores would not be comparative otherwise.

For the mean pre-trial core tensors, the Euclidean distance was calculated by (6.8) below:

$$\begin{aligned} &\|\bar{\mathcal{G}}_{k_1} - \bar{\mathcal{G}}_{k_2}\| \\ &\text{where } \bar{\mathcal{G}}_k = \sum_{p=1}^{P} \mathcal{G}_k^{(p)} \end{aligned} \tag{6.8}$$

where the norm is the tensor norm, i.e. the squared root of the summation over the squared elements (see Appendix). $\bar{\mathcal{G}}_{k_1}$ and $\bar{\mathcal{G}}_{k_2}$ are the mean pre-trial core tensors of two arbitrary sessions k_1 and k_2, and are calculated from the corresponding pre-trials; the P first trials.

The proposed similarity measure is interpreted as that similar sessions have cores that lie close to each other in the mutual feature space. Assuming the pre-trials of a session are representative for its characteristics, the measure should result in choosing sessions with similar features. Further on, we will refer to this approach as Tucker Feature Extraction (TFE).

6.3 Method

The following section accounts for how the tensor representations were chosen when applying TFE for two classification problems: motor imagery classification using Alex MI and drowsiness level regression using SA Drivers. The intention is to explain and motivate the chosen pipeline to the reader, as well as for reproducibility.

Firstly, the method for classification of the small, binary dataset Alex MI is accounted for. Next, the regression problem for the larger dataset SA Drivers is presented. As explained in the previous chapter, the two datasets illuminate different aspects of the tensor decompositions. For Alex MI, the focus will mainly be on what structures are captured by the TFE method. This, as there exists more pre-knowledge for the chosen motor imagery problem, and that it is easier to visualize a small dataset. For SA Drivers, the focus will instead lie in evaluating the TFE method's effect on the performance of the regression. This, as a larger dataset combined with regression allows for greater variation between sessions and complexity in the evaluation, and hence, a more thorough analysis of the method's performance.

The preprocessing in this chapter differs from that of the first pipeline. First, inspired by the work of [Phan and Cichocki, 2010], the time-frequency-domain was utilized instead of solely the frequency-domain. Second, due to not restricting the tensor decompositions to find only non-negative solutions, the normalization of the data differed too. A more thorough explanation of the procedure is accounted for next.

Alex MI

Preprocessing As accounted for in 4.1 **Alex Motor Imagery**, every trial answered to three seconds long EEG data recordings from 16 channels. To compute the tensor representation, the time-frequency representation (TFR) of every trial was extracted using a multitaper method. The parameters were chosen such that the resulting TFR answered to EEG data sampled at 128 Hz, and contained frequencies in the range 8 to 32 Hz with 1 Hz resolution. For all 8 subjects and 320 trials, this resulted in a fourth order tensor of shape $(320 \times 16 \times 25 \times 384)$, answering to the number of trials, channels, frequency- and time-bins. Next, a moving average filter with a window and slide of 10 samples was taken with respect to the final mode; the time domain. This lowered the dimension to $(320 \times 16 \times 25 \times 38)$ and thereby decreased the memory allocation and computational expense during the classification significantly.

For the normalization, the logarithm was firstly applied to every trial. Thereafter, the sessions were divided by the median of their pre-trials, followed by subtracting the median of their pre-trials. Here, the pre-trials answered to the first 8 trials of a session. The final step of subtracting the median was thus, the only difference from the normalization of the PSD representation, see section 5.3 **Method** in Chapter 5 **Tensor Decompositions for Session Selection**. The reason for the additional normalization around zero lied in how the Tucker decomposition was computed. Since the decomposition needed to exclude the final mode, the function *partial_tucker()* from *TensorLy* was used as it allows for ignoring modes. However, the algorithm does not support the constraint of non-negative decompositions. It was thereby no reason for preserving the positivity in the data, and it was normalized around zero to allow for a more well conditioned problem.

Computing the Feature Space To carry out the session selection, and later classification, a mutual feature space for all sessions but the target session was calculated with the Tucker decomposition. Given all eight sessions being viewed as the target session once, eight feature spaces were computed in total. The feature space of each target session then corresponded to the feature space which had excluded this session during creation. Since every session included 40 trials, there were 280 core tensors for every feature space, which were used as the features of the corresponding 280 trials. In accordance with the theory presented in section 6.1 **Background**, the features of the target session was calculated from its feature space and tensor of the original data.

The Tucker decomposition was calculated such that 95% of the variability in the original tensor $\boldsymbol{\mathcal{X}}$ should be explained by the core tensor $\boldsymbol{\mathcal{G}}$, see section 2.2 **Rank Selection of Decompositions**. The 95% significance was chosen by the same argument as the in previous chapter; with the aim of excluding noise and yet, do not risk removing structures that could be of relevance.

Data Exploration To allow for an analysis of what structures the tensor decomposition captured, the mean approximated tensor $\boldsymbol{\mathcal{X}}_c$ was calculated for the two labels. Given all trials, it was defined as:

$$\boldsymbol{\mathcal{X}}_c = \frac{1}{N_c} \sum_{k=1}^{N_c} \boldsymbol{\mathcal{G}}^{(k)} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)} \times_3 \boldsymbol{A}^{(3)} \quad \text{for } c = \textit{right hand} \text{ or } \textit{feet} \tag{6.9}$$

where N_c is the number of trials belonging to class c. The tensor $\boldsymbol{\mathcal{X}}_c$ was in turn, studied in each of the dimensions. Additionally, the feature space of each session was analyzed by studying the factor matrices and resulting distance matrix.

Subject Selection Using the computed feature spaces, a distance matrix was calculated for every session. For each target session, a subset of the k most similar sessions were selected using the distance matrix. Additionally, a random selection was applied for baseline comparison, meaning k sessions were randomly selected from the database. None of the 8 subjects answered to more than one session, allowing k to range from 1 to 7.

Training and Evaluation For training and evaluation, a LOSO cross-validation was applied, as for the previous methods. For each of the eight sessions, $k = 1,\ldots,7$ other sessions were chosen using the distance matrix and random selection. To maintain consistent classification results, the selected sessions were sorted by their numbering. The core tensors of the selected sessions were used as features to train three classifiers; a Linear SVC, an SVC rbf, and a Random Forest classifier. Lastly, the classifiers were used to predict each trial's label in target session. For the random selection, the training and prediction was repeated 20 times and the average performance was calculated. Once again, the prediction accuracy was used to evaluate the result.

SA Driving

Preprocessing The tensor representation for the regression problem used all sessions and trials from SA Drivers (in total 23 subjects, answering to 40 sessions and 18455 trials). The label of a trial was defined by its DI, see section 4.2 **Sustained-Attention Driving Task Dataset** for more details. Note that the number of trials varied with the session, Table 4.2 in 4.2 **Sustained-Attention Driving Task Dataset** presents an overview of the data for every subject.

The tensor representation was defined from the TFR of the data, computed using a multitaper method. Here, the resulting frequency range was 1 to 35Hz with a resolution of 1Hz, and the sampling rate 128Hz. The resulting tensor had a shape of $(18\,455 \times 30 \times 25 \times 384)$, corresponding to $(\textit{trials} \times \textit{channels} \times \textit{frequency} \times \textit{time})$.

Similarly to Alex MI, every trial was normalized by: applying the logarithm and a causal moving average over 10 samples, dividing by the median of the pre-trials, and subtracting the (new) median of the pre-trials. The only difference lied in the number of pre-trials, which was set to $k = 10$ as the data now contained more trials per session. The final tensor had a shape of $(18\,455 \times 30 \times 25 \times 38)$, answering to the number of trials, channels, frequency- and time bins respectively.

As motivated in the previous chapter, the normalization around zero was applied, since there was no reason in preserving the positivity in the data. This, as the algorithm for TFE required using *partial_tucker()*, which did not support non-negative solutions.

Computing the Feature Space To perform session selection and regression, a feature space was computed for every session. This yielded 40 feature spaces, answering to the 40 sessions. The number of features (i.e. cores) varied for different target sessions, as the number of trials varied by session. For a session of a subject, the training data corresponded to the trials of all sessions that did not belong to this subject.

The Tucker decomposition was defined for a rank such that 95% of the variability in the data $\boldsymbol{\mathcal{X}}_{N+1}$ was explained by the core tensor $\boldsymbol{\mathcal{G}}_{N+1}$. The reason for setting $\alpha = 0.95$ was still to exclude noise and not risk removing structures that could be of relevance for the regression.

Data Exploration To allow for an analysis of what structures the tensor decomposition captured, the mean approximated tensor was calculated for a binary class of drowsiness; non-drowsy or drowsy. The labels were determined as in the previous chapter:

$$\begin{aligned} \textit{non-drowsy} \text{ if } \mu &\leq 1.5\mu_0 \\ \textit{drowsy} \text{ if } \mu &\geq 2.5\mu_0 \end{aligned}$$

for the response time μ of a trial, and the median response time μ_0 of the corresponding session's ten first trials. For the resulting subset of trials (trials with a μ between $1.5\mu_0$ and $2.5\mu_0$ were excluded), the mean approximated tensor for each label (non-drowsy or drowsy) was computed by (6.10).

$$\begin{aligned} &\boldsymbol{\mathcal{X}}_c = \frac{1}{N_c} \sum_{k \in \mathcal{N}_c} \boldsymbol{\mathcal{G}}^{(k)} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)} \times_3 \boldsymbol{A}^{(3)} \\ &\text{for } c = \textit{non-drowsy} \text{ or } \textit{drowsy}, \\ &\text{the number of trials } N_c \text{ belonging to class } c\text{, and their indices } \mathcal{N}_c \end{aligned} \tag{6.10}$$

The mean tensor $\boldsymbol{\mathcal{X}}_c$ of each label was in turn, studied in each of the dimensions. Additionally, the feature space of each session was analyzed by studying the factor matrices and the resulting distance matrix.

Session Selection Using the computed feature space, a distance matrix was defined from the similarity measure presented in 6.2 **Our Contribution**; the Euclidean distance between the mean pre-trial cores. For a target session, the k most similar sessions were chosen to train a classifier. For baseline comparison, a random selection was applied, meaning k sessions that did not belong to the same subject were randomly chosen. At most, three sessions were recorded from the same subject, allowing k to range between 1 and 37.

Training and Evaluation For training and evaluation, LOSO cross validation was applied. Every session was thus viewed as the target session once. For $k = 1,\ldots,37$, three regressors were trained; a linear Support Vector Regressor (*linear SVR*), a Support Vector Regressor with a radial basis function kernel (*SVR rbf*), and a Random Forest Regressor. As for the other approaches, the chosen sessions were sorted by their numbering to maintain consistent classification results. Lastly, the DIs of the target session's trials were predicted. For the random selection, the training and prediction was performed 20 times, after which the average performance was used for evaluation. The mean average error (MAE) and Pearson correlation were used as evaluation metrics.

6.4 Result and Discussion

Alex MI

Data Exploration Studying the computed Tucker decomposition, it became evident that the decomposition captured activity in the assumed regions of interest; channel activity over the brain's motor cortex. Figures 6.2-6.4 below visualize the mean of every dimension, computed from the mean approximated tensor for the two labels; right hand and feet (see (6.9)). In Figure 6.2, the mean of is computed with respect to the channel-dimension, while Figures 6.3 and 6.4 answer to the mean of the frequency- and time-dimension. Note that the figures display the absolute values, as it is the amplitude and not the sign that answers to the impact of an element. A large absolute value thereby answers to a high amplitude in relation to the performed normalization. Because of this, another colormap has been used compared to the previous chapter.

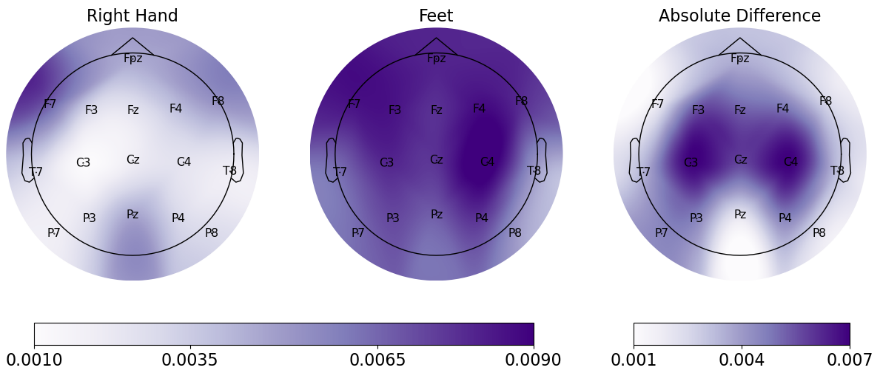


Figure 6.2 The absolute mean with respect to the channel-dimension, calculated from the mean approximated tensor for the two labels; right hand and feet. The right plot answers to the absolute difference between the two labels, i.e. the difference between the left and right plots.

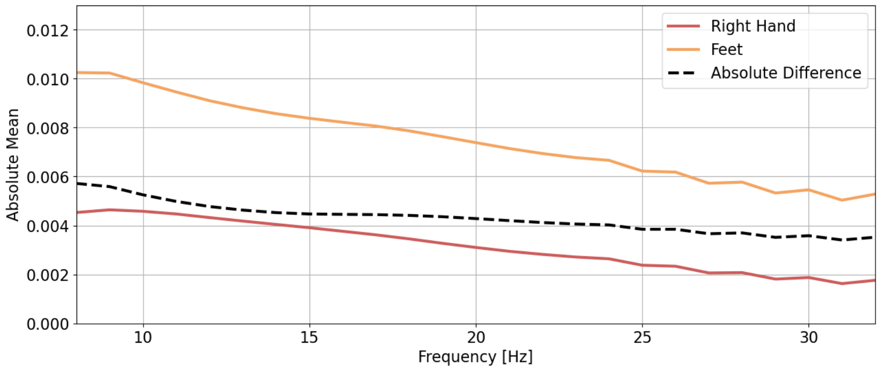


Figure 6.3 The absolute mean with respect to the frequency-dimension, calculated from the mean approximated tensor for the two labels; right hand and feet. The dashed black line represents the absolute difference between the two labels, while the two solid lines answer to each of the two labels.

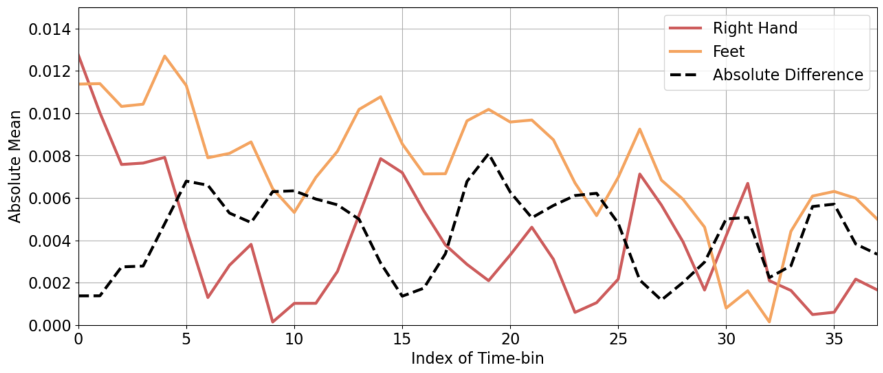


Figure 6.4 The absolute mean with respect to the time-dimension, calculated from the mean approximated tensor for the two labels; right hand and feet. The dashed black line represents the absolute difference between the two labels, while the two solid lines answer to each of the two labels.

Looking at the absolute difference in Figure 6.2 of the channel-dimension, the decomposition has captured a clear difference over the channels C3 and C4. The decomposition thereby indicates that brain activity mainly varies in the motor cortex when imagining the movement of right hand or feet. Continuing to study Figure 6.3 of the frequency-dimension, the results coincide; on average, the imagined movement of feet corresponds to the same frequencies as when imagining moving the right hand, only with a larger amplitude. A similar behavior is also found for the time-dimension in Figure 6.4, where the imagined feet movement overall answers to a larger amplitude over time. Looking at the absolute difference for the two modes though, only the time-dimension shows a clear variation between the two labels. Additionally, the time-dimension shows a difference in trend for the two labels around the time-bin 30. These results point to that the time course for the two labels differs, and that the time-dimension itself includes time-dependent information that can separate the two classes. For the frequency-dimension however, it appears that a mean component would be sufficient to differ between the two labels.

The Feature Space For the chosen significance $\alpha = 0.95$, the rank of the Tucker decomposition was $R_1 = 11$ for the channels dimension, $R_2 = 3$ for the frequency dimension, and $R_3 = 27$ or 28 for the time dimension. Note that as one feature space was computed for every session, the shape of the factor matrices could differ. Since all sessions did not answer to equally large feature spaces, a mean feature space could not be calculated.

The feature space do not model any of the characteristics itself, but only contain the structures necessary for the core tensors to model the data. Thus, the factor matrices simply constitute an orthogonal basis (as orthogonality was a constraint). Studying the columns of each factor matrix, this appears to be the case. Figures 6.5-6.7 show the columns of the corresponding factor matrix, for channels, frequency, and time respectively. The factors answer to those of the 6th session's feature space, and the values correspond to the absolute values of the factors. The used colormap was chosen for the same reason as for Alex MI.

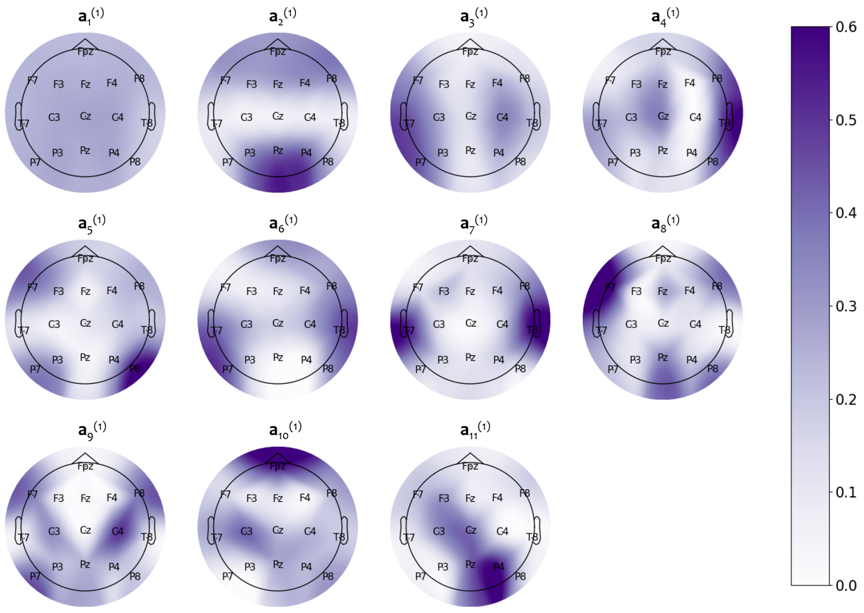


Figure 6.5 Spatial plot of each column of the 6th session's feature space channels-factor $\boldsymbol{A}^{(1)} \in \mathbf{R}^{16\times 11}$. The 6th session was thus excluded when computing these factor matrices. The plot should be read as the deeper the color, the larger the amplitude.

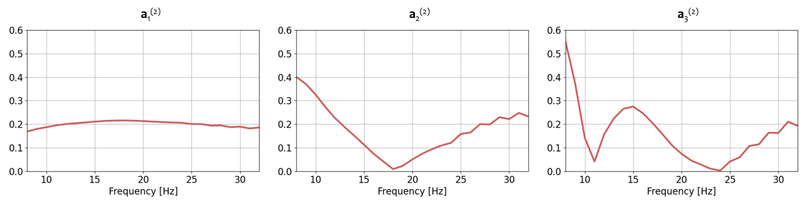


Figure 6.6 Plot of each column of the 6th session's feature space frequency-factor $\boldsymbol{A}^{(2)} \in \mathbf{R}^{25\times 3}$. The 6th session was thus excluded when computing these factor matrices. Note that the plots present the absolute values of a column. A large value thus corresponds to a large amplitude.

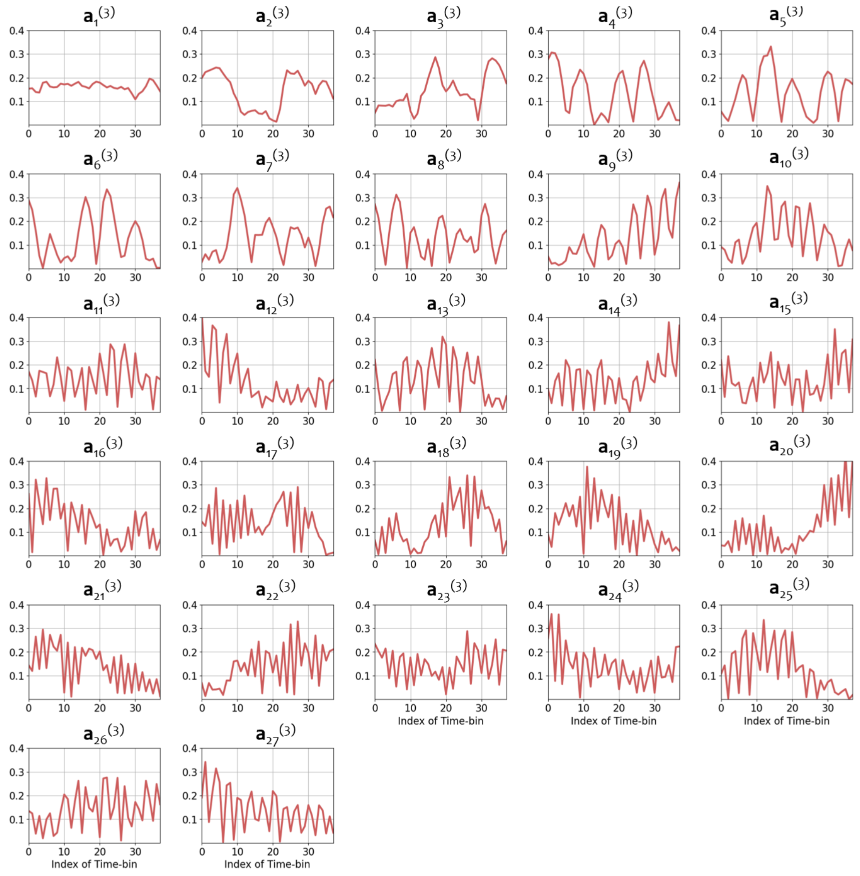


Figure 6.7 Plot of each column of the 6th session's feature space time-factor $\boldsymbol{A}^{(3)} \in \mathbf{R}^{38\times 27}$. The 6th session was thus excluded when computing these factor matrices. Note that the plots present the absolute values of a column. A large value thus corresponds to a large amplitude.

Continuing to study the times factor in Figure 6.7, the three first columns ($\boldsymbol{a}_1^{(3)}$ and $\boldsymbol{a}_3^{(3)}$) show a peek after the 30th time bin, coinciding with when the time characteristics of the two labels differed (see Figure 6.4). A tendency of the same peak is also seen in $\boldsymbol{a}_2^{(3)}$, and $\boldsymbol{a}_4^{(3)}$-$\boldsymbol{a}_7^{(3)}$. This indicates that the basis of the feature space was affected by this characteristics of the EEG signal, meaning the Tucker decomposition would have been tailored by the problem instead of answering to a general basis[2]. The same indication is found from looking at the channel factor in Figure 6.5, where several columns cover the motor cortex. Recalling section 5.4

[2] e.g. a sinusoidal basis with different periods and phases.

Result and Discussion - Alex MI - Data Exploration, C3, C4, and Cz were among the most significant channels when applying ANOVA filters for this motor imagery classification. Thus, it is reasonable that the classification would benefit if the feature space modeled these channels well, e.g. by computing a basis that capture them. Additionally, it should be noted that the discussed characteristics are found in *several* of the columns. Since the features of a trial are defined as the core of this Tucker decomposition, it is reasonable that the combination of columns is sufficient to model the characteristics of each trial.

Returning to the thought of a sinusoidal basis, it should enable a good modeling to general data, as a variation in amplitude, phase, and period allows for covering a variety of signals. Considering the complex and noisy structure of EEG data, a sinusoidal basis could thereby be beneficial. Thus, it is reasonable that the columns' oscillating behavior is similar to that of a sinusoidal basis. In this way, other macroscopic EEG characteristics, apart from the patterns unique to this classification task, is included when approximating the original tensor.

Selection Results The Euclidean distances between sessions were presented in a normalized distance matrix, see Figure 6.8. In accordance with the presented similarity measure in section 6.2 *Our Contribution*, a small distance answered to a large similarity.

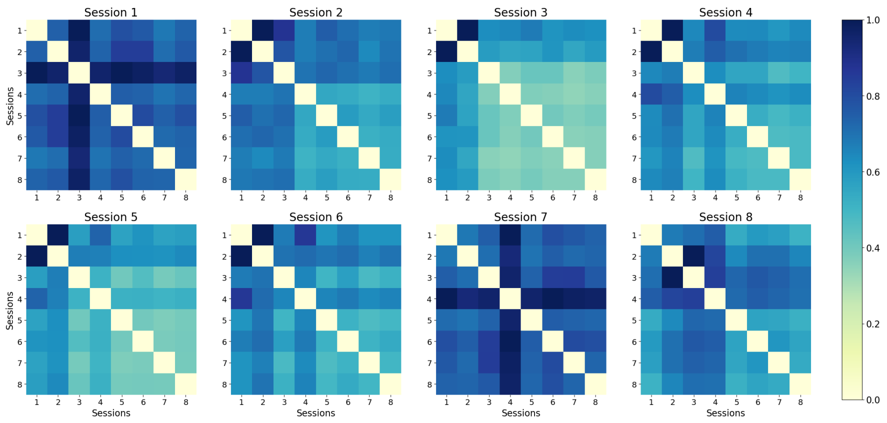


Figure 6.8 The distance matrix for each of the eight sessions. An element d_{ij} answers to the similarity between sessions i and j and represents the Euclidean distance between the mean pre-trial core tensors of the two sessions. Recall that one feature space was computed for each session. The distance matrix of each session was thus calculated from that corresponding feature space.

Note that the corresponding distance matrix of a target session was used for selecting training data for that session only. It was thus, only the corresponding row (or column) that was utilized when selecting training sessions.

Studying the distance matrices in Figure 6.8, the set of similar sessions differ between the matrices. For example, the distance matrix of the first session says that session 4 is one of its most similar sessions, while the distance matrix of the fourth session claims that session 1 is the least similar. Considering the feature spaces were computed from different sets of training data (excluding the target session), the matrices are expected to vary. If the similarities between sessions should be consistent is however not obvious.

Assuming two sessions are similar when looking at their original EEG data, i.e. the $16 \times 25 \times 38$ tensors of the trials, their corresponding core tensors do not need to be similar. This, as we want the Tucker decomposition to project the trials onto a lower dimensional space, which models prominent characteristics of each trial's EEG. For motor imagery, this could answer to capturing the activity over motor cortex, rather than the macroscopic activity in the brain. The macroscopic activity can however, still represent a high activity in the original EEG data and thereby increase the similarity between sessions with non-similar motor cortex activity. Measuring the distance between the original tensors is thus, not necessarily representative for measuring the desired similarity. For the feature spaces, this means the similarity between cores does not need to be consistent as we do not know which similarities to preserve. The inconsistency in similarity between sessions is therefore expected. To obtain a consistency between the distance matrices of difference sessions (i.e. different feature spaces), one would probably need a much larger dataset. The more the subjects, the more similar would the training data between sessions become. Assuming the decomposition is unique, the feature spaces should thus converge to the same solution, leading to identical distance matrices.

Prediction Results The mean prediction accuracy over sessions can be seen in Figure 6.9. Each graph answers to one of the three classifiers (Linear SVC, SVC rbf, and Random Forest).

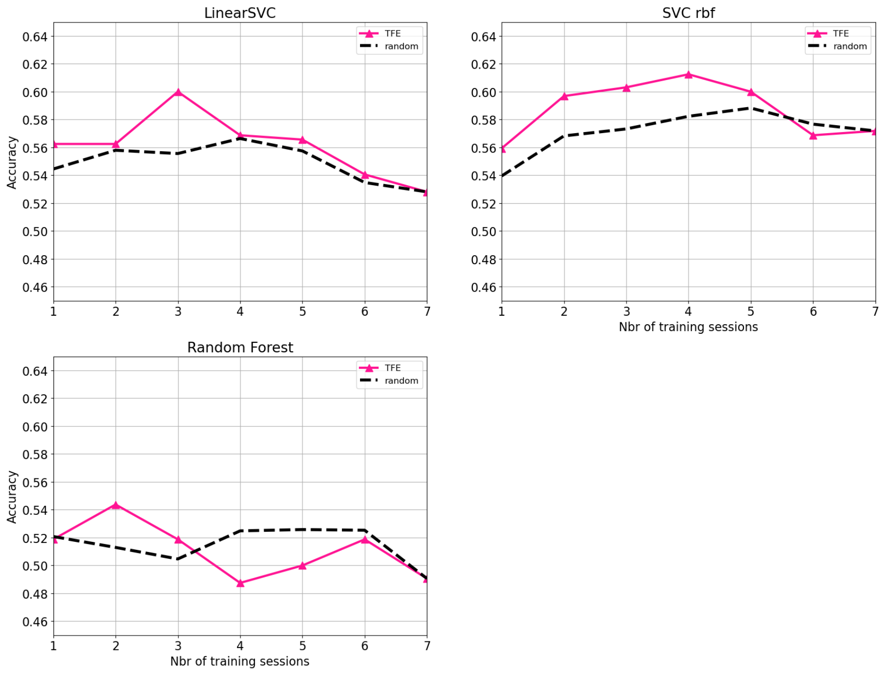


Figure 6.9 The mean accuracy of all sessions, plotted against the number of training sessions included in the training data. The black, dashed line answer to selecting sessions randomly, while the pink, solid line with triangles represent selecting sessions with the presented similarity measure; the Euclidean distance between cores.

Studying Figure 6.9, the prediction accuracy reaches at most about 61%. As of today, BCI state of the art results for binary motor imagery classification with transfer learning reach above 80% [Zhang et al., 2021]. The obtained accuracy can thus be considered poor, even if it exceeds 50% for the majority of training sessions and classifiers. Furthermore, the variance in accuracy when using random selection was large, meaning no conclusions could be drawn of the proposed method being better or not. Considering Alex MI is a small dataset with respect to the number of sessions, this is expected. The contribution from the results rather lie in the study of the decomposition itself, as mentioned in previous sections.

SA Driving

Data Exploration Looking into the computed Tucker decomposition, Figures 6.10-6.12 show the activity captured by the mean approximated tensor, calculated for the binary labels non-drowsy and drowsy, in accordance with (6.10).

Figure 6.10 below displays the absolute values when projecting the mean tensors onto the channels dimension. The projection was done by calculating the mean with respect to the other modes; frequency and time. Similarly, Figures 6.11 and 6.12 view the mean activity over the frequency and time dimension respectively.

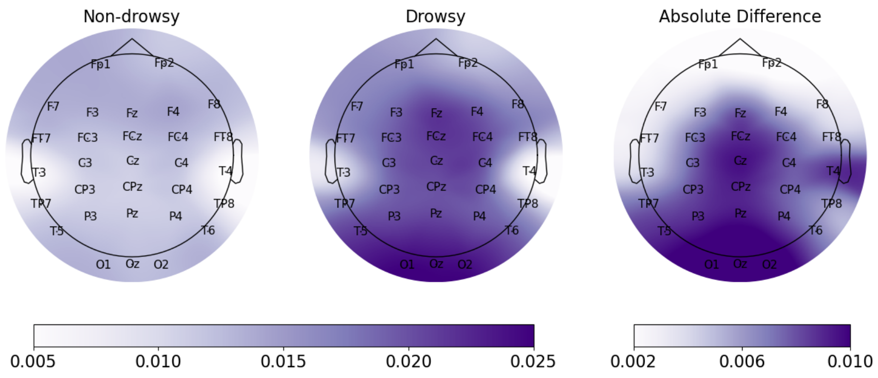


Figure 6.10 The absolute mean with respect to the channel-dimension, calculated from the mean approximated tensor for the two labels non-drowsy and drowsy. The right plot represents the absolute difference between the labels, i.e. the difference between the left and middle plots.

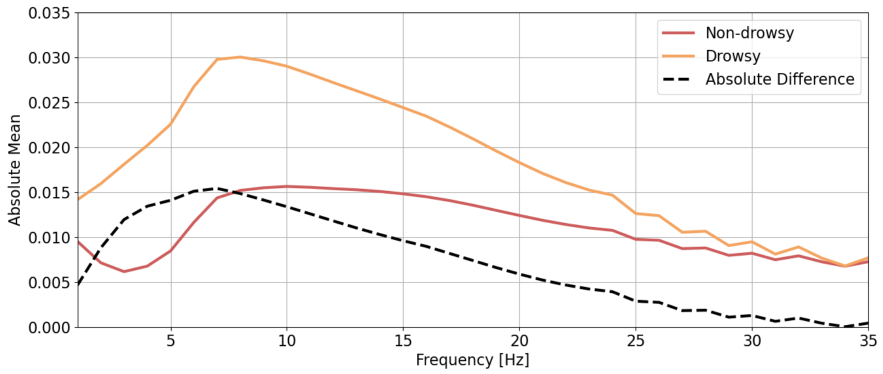


Figure 6.11 The absolute mean with respect to the frequency-dimension, calculated from the mean approximated tensor for the two labels non-drowsy and drowsy. The dashed black line represents the absolute difference between the labels, while the two solid lines represent each of the binary labels.

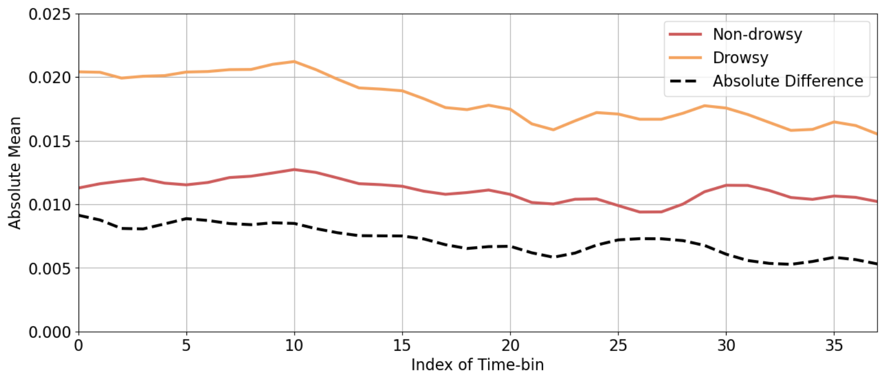


Figure 6.12 The absolute mean with respect to the time-dimension, calculated from the mean approximated tensor for the two labels non-drowsy and drowsy. The dashed black line represents the absolute difference between the labels, while the two solid lines represent each of the binary labels.

Studying the activity over the channels in Figure 6.10, the mean non-drowsy corresponds to a significantly lower amplitude than drowsy for the majority of channels. Further looking at the absolute difference between the labels, the largest variation is found in the middle and back of the brain, and at channel T4. Recalling the pattern found by the ANOVA filter for frequencies 4-6 Hz (see Figure 5.12 in the previous chapter), the results seem reasonable. Continuing to study the activity over frequencies, non-drowsy consistently answers to a lower amplitude than drowsy. In theory though, a higher activity should occur for lower frequencies only. Looking at the absolute difference of the two labels, the support decreases the higher the frequency, yet shows a peak for frequencies in the Theta- and Alpha-bands (4-7 Hz and 8-13 Hz). As those bands, by Table 1.1, correspond to drowsiness, and relaxation and eyes closing, the result can be considered reasonable. Even so, it is surprising that it does not agree with features recognized by the ANOVA filter.

Further looking at Figure 6.12 over the time dimension, no clear difference between the two labels can be seen, apart from that drowsy consistently corresponds to a higher amplitude. Since the same information can be seen from the frequency factor matrix, it is questionable if the time dimension contributes to modeling the drowsiness. This seems reasonable as no particular occasion during a three second trial is expected to stand out. The drowsiness level is measured throughout the entire interval and should not be particularly evident at certain times of a trial. The Alex MI dataset is an example of the opposite: structure could be found in the time dimension as a trial went from relaxing, thinking of moving and then back to relaxing.

The Feature Space For the chosen significance of 95%, the rank of the Tucker decomposition varied between $R_1 = 15, 16$, $R_2 = 5$, and $R_3 = 20, 21$ for the channels, frequency and time dimension respectively. Due to the varying shape, a mean feature space could not be calculated. It is, however, worth to mention that the sessions belonging to the same subject always corresponded to identical feature spaces, indicating the computed decomposition was unique.

As mentioned, the feature space itself does not affect the similarity measure, it only normalizes the data such that all cores lie in the same space. Furthermore, the space is orthogonal (as orthogonality was a constraint), meaning the columns of the factor matrices constitute an orthonormal basis. Studying the columns of each mode's factor thereby allows for a visualization of what patterns the core tensor can capture. Figures 6.13-6.15 show the factor matrices of the mutual feature space corresponding to session 15.

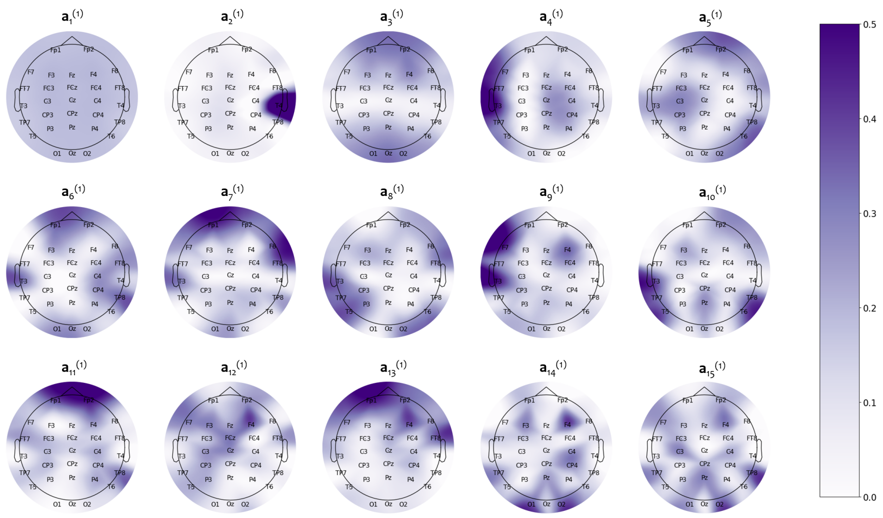


Figure 6.13 Spatial plot of each column of the 15th session's feature space channels-factor $\boldsymbol{A}^{(1)} \in \mathbf{R}^{30\times 15}$. The 5th session was thus excluded when computing the feature space. The plots should be read as the deeper the color, the larger the amplitude.

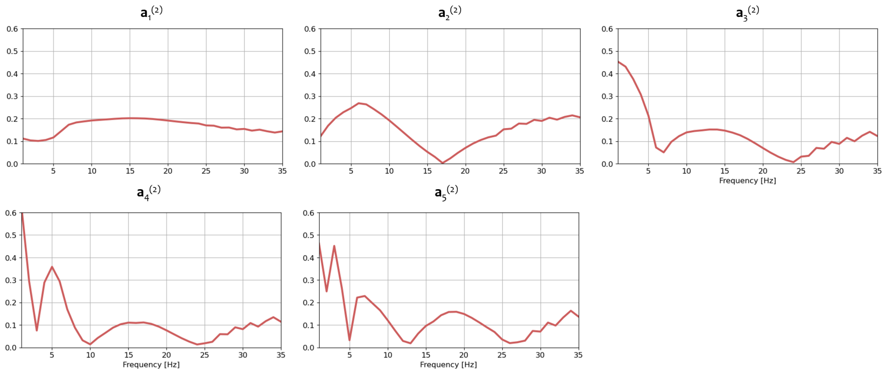


Figure 6.14 Plot of each column of the 15th session's feature space frequency-factor $\boldsymbol{A}^{(2)} \in \mathbf{R}^{25\times 5}$. The 5th session was thus, excluded when computing the feature space. Note that the plots present the absolute values of a column. A large value thus corresponds to a large amplitude.

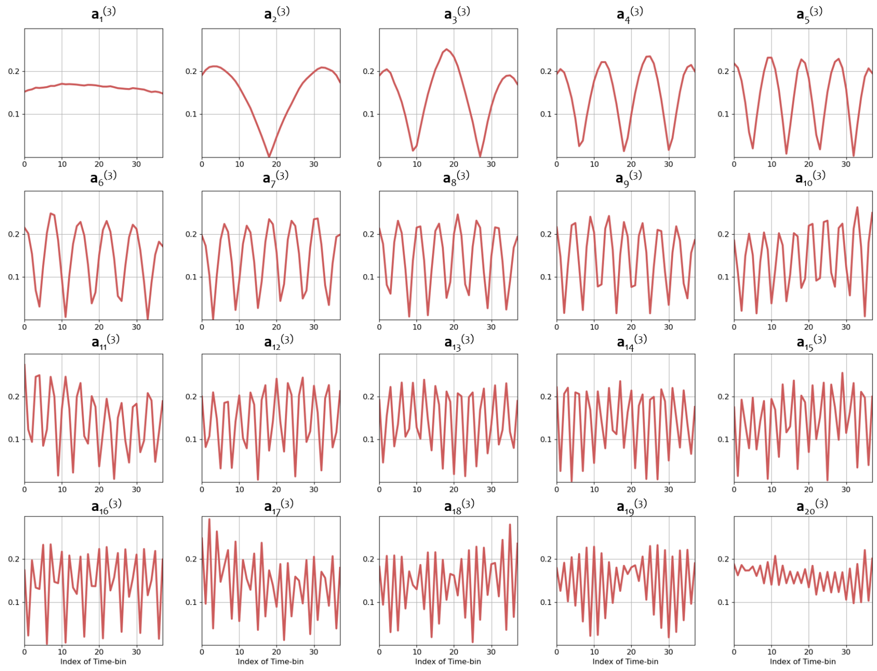


Figure 6.15 Plot of each column of the 15th session's feature space time-factor $A^{(2)} \in \mathbf{R}^{25\times5}$. The 5th session was thus, excluded when computing the feature space. Note that the plots present the absolute values of a column. A large value thus corresponds to a large amplitude.

Looking at the channels-factor in Figure 6.13, the first column $\boldsymbol{a}_1^{(1)}$ appears to answer to a mean. Next, the second column $\boldsymbol{a}_2^{(1)}$ solely cover the activity around channel T4, coinciding with one of the identified structures for the difference between the binary labels non-drowsy and drowsy. Studying the other columns though, no apparent pattern can be seen as they seem to model activity around arbitrary channels. For the frequency factor in Figure 6.14, the columns show an oscillating behavior. For the first column, one oscillation is identified, followed by two for $\boldsymbol{a}_2^{(2)}$, three for $\boldsymbol{a}_3^{(2)}$, and so on. The peaks are, however, crowding to the left rather than covering the whole frequency span equally. Considering the mean tensor showed a decreasing support the higher the frequency, the result can be considered reasonable. Comparing to the factor matrix of the time dimension, the columns in Figure 6.15 do not seem biased for a certain interval, but rather resemble a sinusoidal basis. As discussed for Alex MI, a sinusoidal basis could be beneficial for modeling EEG. Recalling the mean tensor for the time-dimension though, no apparent variation during a trial was recognized. A mean component may thus have been enough to account for the time dimension. Considering the time domain is presented by the same original signal as the frequency domain, one could argue that one of them is unnecessary for modeling the drowsiness. Based on the results, the time domain appears to be redundant in this case. Due to this, a brief study was conducted of whether using the PSD representation only would benefit the regression. Since the results showed a significant loss in performance, the TFR was kept, and we leave a more thorough analysis to future work.

Note that all dimensions showed a consistently larger activity for drowsy than non-drowsy. This indicates a mean component for all modes may have been sufficient to classify if someone is drowsy or not. If this is the case for the continuous drowsiness level can however not be fully concluded.

Once again studying the frequency factor in Figure 6.14, it should be noted that the first column is almost identical to the mean approximated tensor for non-drowsy, recall Figure 6.11. For the frequency domain, this implies the tensor decomposition has chosen a mean a little more similar to non-drowsy, modeling drowsy as a deviation further away from this mean. Looking at the distribution of non-drowsy and drowsy throughout the dataset though, the majority of trials answer to low drowsiness levels (i.e. non-drowsy). The appearance of the mean component is thus, reasonable as the majority of data tends to this mean. The fact that the decomposition identifies this structure though, still argues for tensor decompositions being a good tool in identifying characteristics of data.

Finally, it should be noted that sessions belonging to the same subject resulted in identical feature spaces. The factor matrices were in other words the same for sessions corresponding to the same subject. Since the feature spaces of these sessions were computed from the same data, the result is expected and indicates the Tucker decomposition was unique.

Selection Results As accounted for, one feature space was created for every session. Given all 40 sessions in SA Drivers, there were thus 40 distance matrices. Figure 6.16 below displays a subset of those, answering to session 15, 18, and 26. The three sessions belonged to different subjects and their feature spaces were thereby calculated from different sets of data. Note that the distance matrices are normalized, such that the largest distance answer to 1.

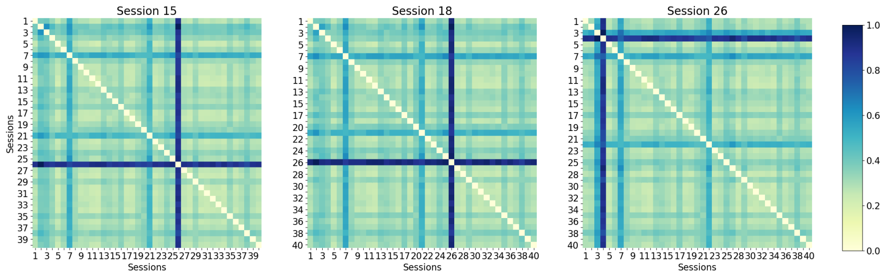


Figure 6.16 The normalized distance matrices of session 15 (left), 18 (middle), and 26 (right). The three sessions belong to different subjects and their feature spaces were thus computed from different sets of data.

Studying the distance matrices in Figure 6.16, session 15 and 18 appear rather similar. Most prominently, they show a clear dissimilarity to session 26 for all sessions. When creating the feature spaces of session 15 and 18, session 26 was included in the data. The Tucker decomposition thereby appears to have seen session 26 as an outlier in order to minimize the distances between the other sessions. What is interesting though, is that the distance matrix of session 26 does not show the same pattern. This, as the 26th row (or column) does not answer to unusually large distances. Assuming the data of session 26 actually answered to outliers, it should result in large dissimilarities when projecting it onto any feature space. The same pattern as for session 26 can be seen for other sessions (e.g. session 4 look like an outlier in the distance matrix of session 26, but not in those of session 15 and 18). The results thus imply the Tucker decomposition tends to view certain sessions as outliers to compute the optimal mutual feature space (recall (6.4) for the problem formulation).

Further looking at the distance matrices in Figure 6.16, sessions belonging to the same subject do not necessarily lie close to each other. For example, the 2×2 bright cluster answering to the 17th and 18th diagonal elements of the 15th session's distance matrix, corresponds to different subjects. Similarly, the 2×2 cluster around the 36th and 37th diagonal elements of the same matrix answers to the same subject, but so does session 38 which is not included. Based on the often large inter-variability in EEG data, it is reasonable to assume that sessions of the same subject are, in general, more similar than sessions of different subjects. Based on these results, it seems like the Tucker decomposition have computed feature spaces that do not only favor the inter-variability between subjects, but also the intra-variability between sessions of the same subject.

A more detailed analysis of how the above discussed patterns affect the performance for a session will be presented next.

Prediction Results Figures 6.17 and 6.18 show the mean Pearson correlation and MAE over sessions, plotted against the number of sessions used when training the regressor. The three graphs in each figure correspond to the three regressors used; Linear SVR, SVR rbf, and Random Forest regressor.

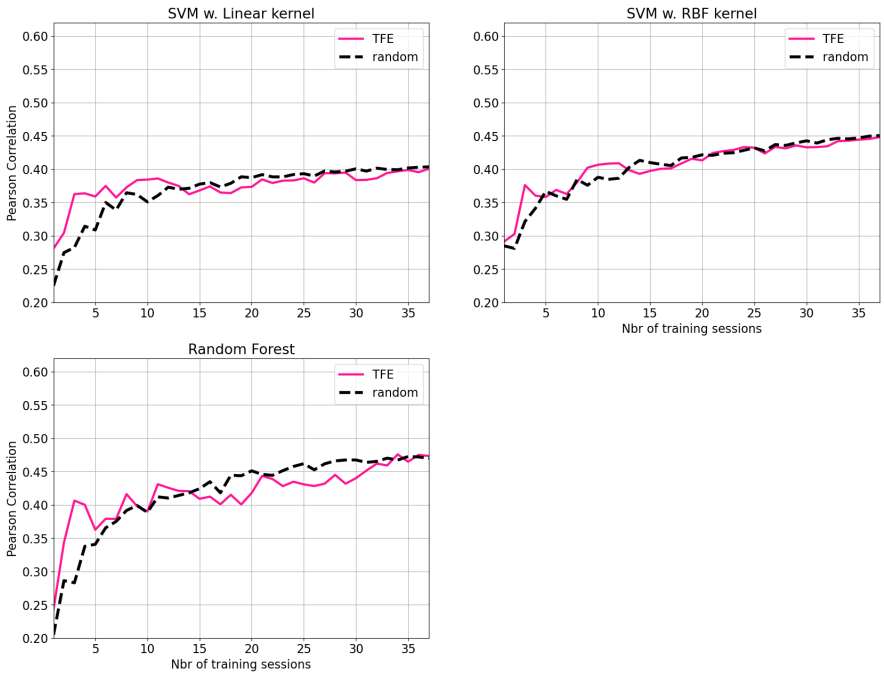


Figure 6.17 The mean Pearson correlation given all 40 sessions, plotted against the number of sessions used when training the regressor. The dashed, black line answers to the performance when using random selection, while the solid, pink line shows the performance when using the proposed similarity measure.

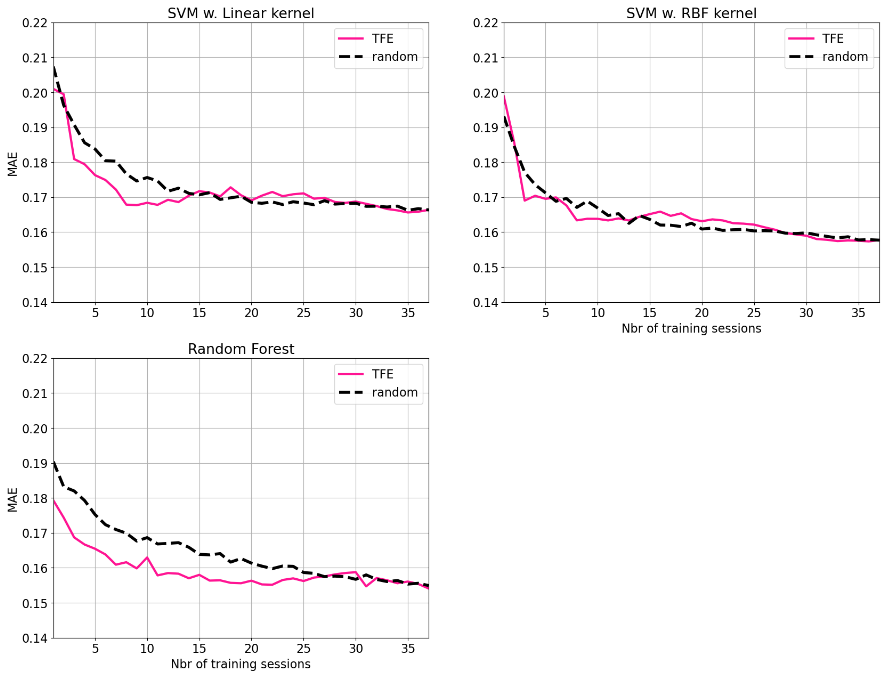


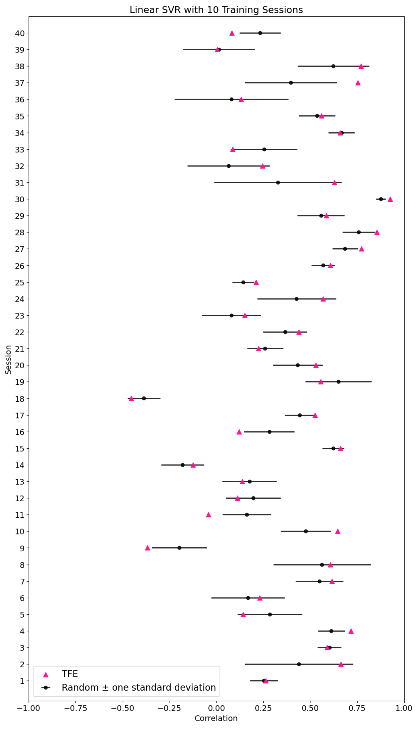
Figure 6.18 The mean MAE given all 40 sessions, plotted against the number of sessions used when training the regressor. The dashed, black line answers to the performance when using random selection, while the solid, pink line shows the performance when using the proposed similarity measure.

From the Pearson correlation results in Figure 6.17, a difference between the two methods is hard to distinguish for any of the regressors. Furthermore, the correlation of the proposed method shows an increasing trend in performance with increasing amount of training data. Continuing to look at the MAE in Figure 6.18, a similar trend can be seen. The proposed similarity measure is consistently better up to 25 training sessions only for the Random Forest regressor, compared to random selection. For the Linear SVR, the proposed method seems to outperform the random selection for training data consisting of about 3-12 sessions. The SVR with an rbf-kernel however, shows no clear difference between the proposed similarity measure and random selection. Studying the combination of a low MAE and a high Pearson correlation, the best results are found at 10 training sessions for the two SVRs, and at 11 training sessions for the Random Forest regressor, as well as training with the full database for all regressors.

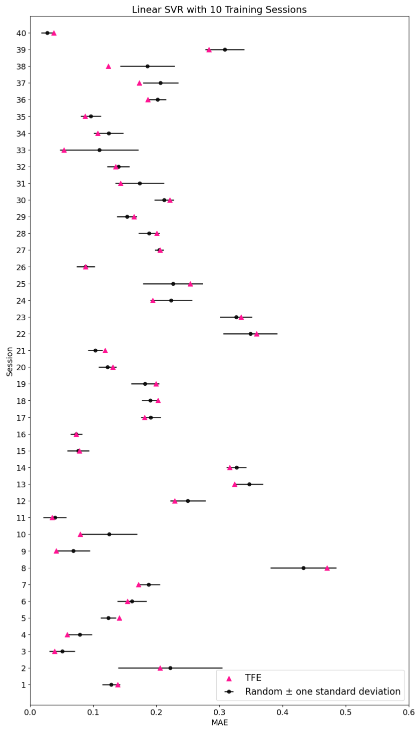
To get an idea of the prediction, Figure 8.6 in the Appendix shows the predicted drowsiness index for all sessions, using 11 training sessions and the Random

Forest Regressor.

To further study the promising results for 10 training sessions for the Linear SVR, the performance of each session was investigated. Figure 6.19 below shows the Pearson correlation and MAE of every session, plotted together with the error plot of the performance when using random selection.



(a) Pearson correlation



(b) MAE

Figure 6.19 Error plots for the Linear SVR with 10 training sessions. Each error bar presents the mean correlation/MAE for a session and its standard deviation during random selection (computed from 20 evaluations). The pink triangles answer to the correlation/MAE when using the proposed similarity measure.

Studying the plots for Pearson correlation, the majority of the sessions perform in the same range independently of using the proposed similarity measure or a random selection. The same applies for the MAE, see Figure 6.19b. Additionally for the Pearson correlation, there is consistently a large variation in performance between sessions. For example, session 15 shows a high correlation (between 0.6 and 0.75) and small variance. Session 18 though, corresponds to the worst correlation (< 0) and a small variance. The small variance of both sessions can thus, be seen as an indication of some sessions being more beneficial to predict the drowsiness level of. From the presented results, there are however no obvious way of telling which these sessions are on beforehand. Recalling the distance matrices in Figure 6.16, no apparent difference between session 15 and 18 could be seen. On a second note, session 18 belongs to the same subject as session 17, which does not show a similar (poor) performance. The supposed non-beneficial sessions thereby do not necessarily correspond to subjects with large inter-variability, but may equally well depend on a large intra-variability.

Another thing that is highlighted by Figure 6.19a, is the large variation in uncertainty between sessions. As the uncertainty is dependent on the random selection, a large standard deviation corresponds to a large sensitivity when selecting training sessions, meaning which sessions we choose have a big impact on the regressor's performance. Naturally, the uncertainty decreases with more training sessions, since the possible combinations of sessions become fewer. However, this also means it becomes harder to evaluate the significance of the proposed method being better than random selection. Once again looking at the mean correlation and MAE by session (Figures 6.17 and 6.18), it is desirable to know if the proposed similarity measure, with some significance, outperforms random selection for a subset of all sessions (here, probably 10 or 11 as discussed). To further study this, Table 6.1 displays a summary of the results in Figure 6.19.

Table 6.1 Count (out of all 40) and percentage of sessions that performed better than random for the presented number of training sessions in Figure 6.19 (10 for Linear SVR).

	Linear SVR	
	Pearson	MAE
# better than random	26	17
% better than random	65.0%	42.5%

As seen in Table 6.1, the best results are obtained for the Pearson correlation, where the proposed method results in a higher correlation than random selection for 65% of the sessions. For the MAE, the result is poorer, as only 42.5% of the sessions performed better than random. From the above analysis, the proposed method can thereby not be concluded to outperform random selection with a quantitative certainty for the chosen number of training sessions (10).

Viewing the problem from another perspective, selecting a subset of sessions to train from can be desirable in terms of computational expense, as accounted for in the previous chapter. By Figure 6.17, one can on average gain about 2, 4, and 5 percentage units in correlation when using all sessions instead of a subset of

10, 10, and 11 sessions for Linear SVR, SVR rbf, and Random Forest Regressor respectively. Comparing the computational expense for training a regressor from 10 or all sessions, using the subset cuts the run time to a 1/2, 1/13, and 1/6 for the Linear SVR, SVR rbf, and Random Forest Regressor respectively. Considering BCIs are often desired to use as real-time applications, the run time is substantial if the application is to work in practice. Improved performance is hence not all that matters, but reducing training times and still achieving comparable performance is of high relevance too (as concluded in the previous chapter).

7

Conclusions and Future Work

Throughout this project, several assumptions have been made, and their credibility and consequences are going to be considered next in section 7.1 **General Discussion**. Additionally, general results and conclusions that apply to Chapters 5 and 6 will be examined. Thereafter, possible future extensions will be discussed in section 7.2 **Future Work**, and the summarizing conclusions of the report will be enhanced in section 7.3 **Conclusions**.

7.1 General Discussion

Feature comparison In the two Chapters 5 and 6, two different ways of representing EEG data to train classifiers/regressors were presented. The first one simply used the EEG signals PSD. The second one included the TFR of the signal, formulated as a tensor and for which a Tucker decomposition was computed to find features. Comparing the performance of the two methods for the bigger dataset SA Driving, the first approach was superior with the Random Forest regressor. For the two SVRs, the two methods produced similar performance. As of now, the second method (which used tensor decompositions as inputs to the regressors instead of a simple frequency representation) can not be considered a better choice than the first. This is as it did not show an increase in performance, and that each trial was represented with more data points, which increased storage requirements and computational expense. However, with future extensions as supervised and/or nonlinear decompositions (described in Section 7.2 **Future Work**), the results may become more satisfying.

Data Collection Throughout this project, we have worked with data that we have not collected ourselves. This has made it possible to concentrate on the methods and analysis. The datasets we have investigated (AlexMI and SA Drivers) are well used in literature and thoroughly documented, which has been convenient. However, it is an evident drawback when analyzing results to not have collected data ourselves, especially when comparing similarity of sessions and subjects. In all our conclusions regarding similarity of sessions, we assume that there have not been any other factors than the EEG signals themselves that affect the data. If we would have collected data ourselves, it may have been evident to us if the subjects behaved differently in other senses, or if the equipment was more/less convenient to use between sessions. Additionally, one could have communicated with the subjects before/during/after the data collection to receive more information of mood, current mental state and other characteristics of interest that could affect the EEG signals. This is a clear drawback in our analysis, and can have contributed to the big difference in performance between sessions.

Similarity of Sessions Assuming data collection did not answer to the main impact on session similarity, let us consider other reasons for the big difference in performance between sessions. For example, does the quality of the selected subsets vary between sessions? As we are only looking at relatively small datasets of 8 or 40 sessions, this is likely the case. Most probable is that some sessions have more similar sessions to choose from than others. Indications of this can be seen in almost all distance matrices, where there are sessions that seem further away from all other sessions in an inconsistent manner. Additionally, from the classification/regression results, some sessions are consistently harder to receive high performance for than others. This is seen in for example Figures 5.17 and 5.18 in Chapter 5, and Figure 6.19 in Chapter 6.

Imagine placing an infinite amount of sessions from different subjects in a multi-dimensional space, where the true similarity can be measured. Would the sessions cluster into different groups, or would they make up a continuous distribution? This thought is of interest, as it would supply a constraint on how good of a performance we can reach for dataset with limited amount of training sessions. Assuming the data is clustered, it might be enough to have a few samples from each group in the training database and categorize a target session accordingly. To investigate the clustering-capability of the data, a more in-depth analysis of the sessions and their neighbors would have to be made. Additionally, such an analysis would profit from an even bigger dataset.

Independent of the "true" distribution of sessions, from small datasets (like the ones we have used), it is unlikely that all sessions benefit from the same number of training sessions. In our analysis, we have only compared all fixed number of subsets possible, and averaged the performance for all sessions. More realistically, there is no mutual maximum. Some sessions may reach maximal performance by selecting a training dataset of few sessions, while others from more. A more dynamic session selection approach, including a flexible decision on how many training sessions to include, would probably be beneficial.

Pre-trial Similarity vs Drowsiness Response For SA Drivers dataset, the goal has been to examine data in a way that would be interesting in future BCI applications. Therefore, only the pre-trial data has been considered when comparing similarity between sessions. The pre-trial data can be quickly collected for a new session (less than 3 minutes), and does not include drowsy states of the subject. This means that in a real-world application, the subject does not have to be exposed to dangerous traffic situations for calibration.

For the SA Drivers dataset, it is assumed that two sessions that are similar in their pre-trial data will also have a similar EEG drowsiness response. As the brain is complex and EEG data only capture macroscopic behavior, this assumption is hard to evaluate and not necessarily true. It could be that two sessions initially have similar EEG data, but have extensively different characteristics when turning drowsy. If that is the case, there would be no evident benefit from finding similar sessions during the pre-trial phase to train a regressor.

7.2 Future Work

Supervised Selection of Latent Variables When working with tensors, it has been done in an unsupervised manner. We have thus been able to examine all structures enhanced in the decompositions as a result of the input EEG data. However, the tensor decompositions have never been impacted by the specific classification problem, and have therefore never adapted their components to it. As seen in the analyses of the Alex MI dataset in Chapters 5 and 6, the columns of the factor matrices in the channel/frequency/time-dimensions can be connected to the specific classification problem if pre-knowledge of certain relevant channels-frequencies-time combinations exists. For future extension, we propose examining *supervised* tensor decompositions for tasks that clearly enhance certain EEG channels/frequencies/times. This could be made in a fashion that compares how well columns in the factor matrices separate different classes. The weighting factor of the latent variables could thereafter be adapted from this, when using tensor decompositions as session selection as proposed in Chapter 5. Or, only these columns could be included as features, when using tensor decomposition for feature extraction as proposed in Chapter 6. Another approach could be to formulate an optimization problem that penalizes the decomposition if it does not separate the classes.

As a starting point for this extension, a brief check of whether the prediction accuracy of AlexMI could benefit from using a subset of the latent variables was conducted. For feature extraction and session selection (as in Chapter 6), a gain of 3 to 7 percentage units in accuracy was on average obtained when manually selecting which columns of the factor matrices to use. The chosen columns covered the assumed interesting characteristics; activity over the motor cortex, the frequency ranges 9-12Hz and 17-19Hz, and a time interval around the 30th time bin. Because of the promising results, we confirm that this future extension has potential.

Lastly, the effect of the decomposition's rank should be mentioned. As presented, the rank was derived by choosing a significance α, answering to how much of the data variability the decomposition should explain. Requiring a small significance is thus a way of further reducing the dimension of the decomposition. In the presented results, α was set to 0.95, answering to a 95 % significance. Some method(s) were however, tested for a lower alpha too. Since the lower significance resulted in a worsened performance, where the factor matrices tended to a mean component only, the 95 % significance was concluded better for our unsupervised problem. In a supervised problem though, a lower rank should allow for deriving a lower dimensional decomposition that captures features of the wanted labels, rather than the overall characteristics of the whole dataset. For a supervised approach, we thus see potential in investigating the rank of the decomposition further.

Non-linear Decompositions As the non-linear regressors (SVR rbf and Random Forest) have proven their superiority to the linear one (Linear SVR), it is reasonable to assume that relationships in EEG data benefit from non-linear comparisons. We therefore propose non-linear tensor decomposition methods as a future extension to this project. We believe that non-linear structures can be beneficial both for similarity comparison and feature extraction. In literature, an *infinite Tucker decomposition* [Xu et al., 2012] has been proposed, which we consider a good starting point for this.

Classifiers/Regressors As the classifiers/regressors differed in performance, it would be reasonable to look further at how to maximize their performance. To do so, more time should be spent on tuning. This should be done in a non-overfitting manner, that do not benefit one/several particular selection methods or number of training sessions.

Additionally, other classifiers/regressors, which use different methods to separate data, could be of interest. In particular, we propose investigating the impact of neural networks. Neural networks are generally good at finding complex, non-linear relationships in data, and they have showed promising results for EEG data lately. A good starting point could be the EEG net [Lawhern et al., 2016], which has proven to perform well for EEG data in different tasks. EEG net can be extended from a classification to regression task by for example adjusting the last layer, as was done in [Cui and Wu, 2018].

7.3 Conclusions

This report contains an analysis of two methods for evaluation of tensor decompositions in the field of transfer learning. In it, we have developed a new similarity selection method. The method involves *weighting* the latent vectors of the mode of interest, for example representing a session, with the corresponding size of what it scales during recreation/approximation of the original tensor. We consider this a more representative measurement of how similar two sessions are, compared to simply studying the unweighted factor matrices which has been done previously in literature.

Additionally, two main questions were brought up in the introduction, which have been answered in this work.

Can tensor decompositions, performed unsupervised, capture structures relevant and useful to a classification problem? Based on the analysis of the Alex MI dataset in Chapters 5 and 6, it can be said that tensor decompositions can extract structures that are of more or less relevance to a classification problem. For example, a subset of columns of the channel and frequency factor matrices in Chapters 5 and 6 did enhance the same channels/frequencies as was selected during the ANOVA filter investigation. With this knowledge, we propose extending the tensor decomposition methods to supervised ones, both for session selection and feature extraction. In addition, tensor decompositions for feature extraction of EEG trials did not outperform simply transforming the trials to the frequency domain (PSD). However, with future extensions as supervised and/or nonlinear decompositions, the results may be more satisfying.

Are the captured structures efficient for similarity comparison between sessions in a transfer learning setting? From the SA Driving dataset analysis in Chapter 5 (session selection), there are indications of the tensor selection methods in many cases performing better than random. Furthermore, maximal performance (for the average of all sessions) was often reached when using a tensor selection method and training with a subset smaller than the full database. Similar results were obtained in Chapter 6 when using tensor decompositions for both feature extraction and session selection, even if the performance continuously increased with the number of training sessions, in contrast to Chapter 5. For both chapters, these are promising results which show potential in the tensor selection methods. However, looking closer at every single session, there is a wide spread in performance and an inconsistency in performance increase/decrease when involving the tensor selection methods. Therefore, for a new session, little can be said about performance and it is relatively uncertain if the tensor selection methods would lead to better results than random selection. This makes these regressor models far from applicable for a BCI in a real-world environment today.

Bibliography

Acar, E., T. G. Kolda, and D. M. Dunlavy (2011). "All-at-once Optimization for Coupled Matrix and Tensor Factorizations". URL: `http://arxiv.org/abs/1105.3422`.

Barachant, A., S. Bonnet, M. Congedo, and C. Jutten (n.d.). "Classification of covariance matrices using a Riemannian-based kernel for BCI applications" (). DOI: `10.1016/j.neucom.2012.12.039{\"{i}}`. URL: `https://hal.archives-ouvertes.fr/hal-00820475`.

Barachant, A. and S. Chevallier (2021). *moabb.datasets.AlexMI*. URL: `http://moabb.neurotechx.com/docs/generated/moabb.datasets.AlexMI.html`.

Berger, M. (2002). *A Panoramic View of Riemannian Geometry*. Springer, p. 124.

Bolagh, S. N. G. and G. D. Clifford (2017). "Subject Selection on a Riemannian Manifold for Unsupervised Cross-subject Seizure Detection". URL: `http://arxiv.org/abs/1712.00465`.

Cao, Z., C. H. Chuang, J. K. King, and C. T. Lin (2019). "Multi-channel EEG recordings during a sustained-attention driving task". *Scientific data* **6**:1, p. 19. ISSN: 20524463. DOI: `10.1038/s41597-019-0027-4`.

Cichocki, A., S. Profile, and R. Zdunek (2009). *Nonnegative Matrix and Tensor Factorizations-Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*, p. 50. URL: `https://www.researchgate.net/publication/237145400`.

Congedo, M., A. Barachant, and R. Bhatia (2017). "Riemannian geometry for EEG-based braincomputer interfaces; a primer and a review". *Taylor & Francis*, pp. 155–174.

Cui, Y. and D. Wu (2018). "EEG-Based Driver Drowsiness Estimation Using Convolutional Neural Networks". URL: `http://arxiv.org/abs/1809.00929`.

De Lathauwer, L., B. De Moor, and J. Vandewalle (2000). *ON THE BEST RANK-1 AND RANK-(R 1 , R 2 ,. .. , R N) APPROXIMATION OF HIGHER-ORDER TENSORS **. Tech. rep. 4, pp. 1324–1342. URL: `https://epubs.siam.org/terms-privacy`.

Drew, L. (2019). "The ethics of brain–computer interfaces". *Nature* **571**:7766. ISSN: 14764687. DOI: `10.1038/d41586-019-02214-2`.

Encyclopaedia Britannica (2007). *Riemannian geometry*. URL: `https://www.britannica.com/science/Riemannian-geometry`.

Energy Efficiency of Medical Devices and Healthcare Applications (2020). Elsevier, p. 28. DOI: `10.1016/c2018-0-04773-8`.

Gramfort, A., M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, and M. Hämäläinen (2013). "MEG and EEG data analysis with MNE-Python". *Frontiers in Neuroscience* 7 DEC. ISSN: 1662453X. DOI: `10.3389/fnins.2013.00267`.

James, G., D. Witten, T. Hastie, and R. Tibshirani (2013). *An introduction to statistical learning : with applications in R*. Springer, New York, pp. 320–352. ISBN: 9781461471370. URL: `https://www.ime.unicamp.br/~dias/Intoduction%20to%20Statistical%20Learning.pdf`.

Jeng, P. Y., C. S. Wei, T. P. Jung, and L. C. Wang (2021). "Low-Dimensional Subject Representation-Based Transfer Learning in EEG Decoding". *IEEE Journal of Biomedical and Health Informatics* **25**:6, pp. 1915–1925. ISSN: 21682208. DOI: `10.1109/JBHI.2020.3025865`.

Kim, Y.-D. and S. Choi (2007). *Nonnegative Tucker Decomposition*. Tech. rep.

Kolda, T. G. (2006). *Multilinear operators for higher-order decompositions*. Tech. rep. URL: `http://www.doe.gov/bridge`.

Kolda, T. G. and B. W. Bader (2009). "Tensor decompositions and applications". *SIAM Review* **51**:3, pp. 455–500. ISSN: 00361445. DOI: `10.1137/07070111X`.

Kossaifi, J., Y. Panagakis, A. Anandkumar, and M. Pantic (2019). *TensorLy: Tensor Learning in Python*. Tech. rep., pp. 1–6. URL: `http://jmlr.org/papers/v20/18-277.html.`.

Lars-Christer Böiers (2010). *Mathematical Methods of Optimization*. 2nd ed. Vol. 1. Studentlitteratur AB.

Lawhern, V. J., A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance (2016). "EEGNet: A Compact Convolutional Network for EEG-based Brain-Computer Interfaces". DOI: `10.1088/1741-2552/aace8c`. URL: `http://arxiv.org/abs/1611.08024%20http://dx.doi.org/10.1088/1741-2552/aace8c`.

Lee, D. D. and H. S. Seung (1999). *Algorithms for Non-negative Matrix Factorization*. Tech. rep.

Lin, C. Y., L. C. Wang, and K. H. Tsai (2018). “Hybrid Real-Time Matrix Factorization for Implicit Feedback Recommendation Systems”. *IEEE Access* **6**, pp. 21369–21380. ISSN: 21693536. DOI: `10.1109/ACCESS.2018.2819428`.

National Council of Educational & Research Training (2021). “Introduction to Euclid’s Geometry”. In: *Mathematics*, pp. 80–83.

Pedregosa, F., V. Michel, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, J. Vanderplas, D. Cournapeau, F. Pedregosa, G. Varoquaux, A. Gramfort, B. Thirion, O. Grisel, V. Dubourg, A. Passos, M. Brucher, M. Perrot andÉdouardand, a. Duchesnay, and F. Duchesnay (2011). *Scikit-learn: Machine Learning in Python.* Tech. rep., pp. 2825–2830. URL: `http://scikit-learn.sourceforge.net..`

Phan, A. H. and A. Cichocki (2010). “Tensor decompositions for feature extraction and classification of high dimensional datasets”. *Nonlinear Theory and Its Applications, IEICE* **1**:1, pp. 37–68. ISSN: 2185-4106. DOI: `10.1587/nolta.1.37`.

Rowland, T. (n.d.). *Manifold.* URL: `https://mathworld.wolfram.com/Manifold.html`.

Shashua, A. and T. Hazan (2005). “Non-negative tensor factorization with applications to statistics and computer vision”. In: *ICML 2005 - Proceedings of the 22nd International Conference on Machine Learning*, pp. 793–800. ISBN: 1595931805. DOI: `10.1145/1102351.1102451`.

Sörnmo, L. (2005). *Bioelectrical signal processing in cardiac and neurological applications.* Biomedical engineering: 8. Elsevier, p. 34. ISBN: 9780124375529.

Stoica, P. and R. L. Moses (2005). *Spectral analysis of signals.* Pearson/Prentice Hall, pp. 1–12. ISBN: 0131139568.

Thomson, D. J. (1982). “Spectrum Estimation and Harmonic Analysis”. *Proceedings of the IEEE* **70**:9, pp. 1055–1096. ISSN: 15582256. DOI: `10.1109/PROC.1982.12433`.

Wei, C. S., Y. P. Lin, Y. T. Wang, C. T. Lin, and T. P. Jung (2018). “A subject-transfer framework for obviating inter- and intra-subject variability in EEG-based drowsiness detection”. *NeuroImage* **174**, pp. 407–419. ISSN: 10959572. DOI: `10.1016/j.neuroimage.2018.03.032`.

Xu, Z., F. Yan, and Y. Qi (2012). *Infinite Tucker Decomposition: Nonparametric Bayesian Models for Multiway Data Analysis.* Tech. rep.

Zhang, K., N. Robinson, S. W. Lee, and C. Guan (2021). “Adaptive transfer learning for EEG motor imagery classification with deep Convolutional Neural Network”. *Neural Networks* **136**, pp. 1–10. ISSN: 18792782. DOI: `10.1016/j.neunet.2020.12.013`.

8

Appendix

8.1 Basic Operations

The Outer Product

Assuming two first order tensors $\boldsymbol{a}$ and $\boldsymbol{b}$ of shape $A \times 1$ and $B \times 1$ respectively, their outer product is a second order tensor $\boldsymbol{N}$ of shape $A \times B$:

$$\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{a}\boldsymbol{b}^T = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_A \end{bmatrix} \begin{bmatrix} b_1 & b_2 & \cdots & b_B \end{bmatrix} = \begin{bmatrix} a_1 b_1 & a_1 b_2 & \cdots & a_1 b_B \\ a_2 b_1 & a_2 b_2 & \cdots & \vdots \\ \vdots & & \ddots & \\ a_A b_1 & \cdots & & a_A b_B \end{bmatrix}$$

where an element n_{ij} answers to the product between the ith and jth element of $\boldsymbol{a}$ and $\boldsymbol{b}$ respectively. The outer product can thus be seen as scaling a tensor $\boldsymbol{a}$ with each of the elements of another tensor $\boldsymbol{b}$. Taking the outer product between three rank-one tensors $\boldsymbol{a}, \boldsymbol{b}$, and $\boldsymbol{c} = [c_1\ c_2\ \cdots\ c_C]^T$, the resulting tensor will have the shape $A \times B \times C$ and represent C stacked matrices, each answering to $\boldsymbol{a} \circ \boldsymbol{b}$ weighted by the corresponding element of c.

More generally, the outer product of M first order tensors $\boldsymbol{a}, \boldsymbol{b}, \ldots, \boldsymbol{m}$ is a Mth order tensor of shape $A \times B \times\ \ldots\ \times M$ and the $(i_A, \ldots, i_M)$:th element answers to:

$$n_{i_A, i_B, \ldots, i_M} = a_{i_A} b_{i_B} \cdots m_{i_M}$$

The Kronecker Product

The Kronecker product generalizes the outer product to matrices; second order tensors. As for the outer product, a tensor is distributed to another tensor, weighted by the corresponding element. For two second order tensors $\boldsymbol{A} \in \mathbf{R}^{I \times J}$ and $\boldsymbol{B} \in \mathbf{R}^{K \times L}$, the Kronecker product is defined as:

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots & a_{1J}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots & a_{2J}\boldsymbol{B} \\ \vdots & & \ddots & \vdots \\ a_{I1}\boldsymbol{B} & \cdots & & a_{IJ}\boldsymbol{B} \end{bmatrix} = \boldsymbol{N}$$

where the resulting tensor $\boldsymbol{N}$ is a second order tensor of shape $(IK) \times (JL)$.

Furthermore, it holds that:

$$(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \otimes \boldsymbol{D}) = (\boldsymbol{AC}) \otimes (\boldsymbol{BD}) \tag{8.1}$$

which results in that the Kronecker product $(\boldsymbol{A} \otimes \boldsymbol{B})$ between (semi) orthogonal[1] matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ is (semi) orthogonal:

$$(\boldsymbol{A} \otimes \boldsymbol{B})^T(\boldsymbol{A} \otimes \boldsymbol{B}) = (\boldsymbol{A}^T \otimes \boldsymbol{B}^T)(\boldsymbol{A} \otimes \boldsymbol{B}) = (\boldsymbol{A}^T\boldsymbol{A}) \otimes (\boldsymbol{B}^T\boldsymbol{B}) = \boldsymbol{I} \otimes \boldsymbol{I} = \boldsymbol{I} \tag{8.2}$$

The Khatri-Rao Product

The Khatri-Rao product can be seen as a columnwise Kronecker product. Assuming two second order tensors $\boldsymbol{A} \in \mathbf{R}^{I \times K}$ and $\boldsymbol{B} \in \mathbf{R}^{J \times K}$, it is defined as:

$$\begin{aligned} \boldsymbol{A} \odot \boldsymbol{B} &= \begin{bmatrix} \boldsymbol{a}_1 \otimes \boldsymbol{b}_1 & \boldsymbol{a}_2 \otimes \boldsymbol{b}_2 & \cdots & \boldsymbol{a}_K \otimes \boldsymbol{b}_K \end{bmatrix} \\ &= \begin{bmatrix} a_{11}\boldsymbol{b}_1 & a_{12}\boldsymbol{b}_2 & \cdots & a_{1K}\boldsymbol{b}_K \\ a_{21}\boldsymbol{b}_1 & a_{22}\boldsymbol{b}_2 & & a_{2J}\boldsymbol{b}_K \\ \vdots & & \ddots & \vdots \\ a_{I1}\boldsymbol{b}_1 & \cdots & & a_{IK}\boldsymbol{b}_K \end{bmatrix} \end{aligned}$$

where $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ are the ith column vectors of $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively. The resulting product is a second order tensor of shape $(IJ) \times K$. Note that $\boldsymbol{A}$ and $\boldsymbol{B}$ must share the number of columns for the Khatri-Rao product to be possible, which is not the case with the Kronecker product.

[1] A *semi* orthogonal matrix $\boldsymbol{A}$ does not need be a square matrix but must fulfill $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$ or $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}$

The Hadamard Product

The Hadamard Product answers to the elemental product between matrices, and the matrices must thereby have the same shape. For two matrices $\mathbf{A} \in \mathbf{R}^{I\times J}$ and $\boldsymbol{B} \in \mathbf{R}^{I\times J}$ the product is defined as:

$$\boldsymbol{A} * \boldsymbol{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2J}b_{2J} \\ \vdots & & \ddots & \vdots \\ a_{I1}b_{I1} & \cdots & & a_{IJ}b_{IJ} \end{bmatrix}$$

The Tensor Norm

For matrices, the Frobenius norm answers to the squared root of the summation over the squared elements. For the tensor norm of a tensor $\boldsymbol{\mathcal{X}} \in \mathbf{R}^{I_1\times I_2\times\cdots\times I_N}$ of order N, the principle is the same:

$$\|\boldsymbol{\mathcal{X}}\| = \sqrt{\sum_{i_1,\ldots,i_N}^{I_1,\ldots,I_n} x^2_{i_1,\ldots,i_N}}$$

where the sum is taken over the squared elements of the tensor.

The Moore Penrose Pseudo-inverse

The Moore Penrose pseudo-inverse is a generalization of the matrix inverse since it does not require a matrix to be quadratic. However, it still satisfies properties similar to those of the matrix inverse. For a matrix $\boldsymbol{A}$ with the pseudo-inverse $\boldsymbol{A}^\dagger$ it holds that:

1. $\boldsymbol{A}\boldsymbol{A}^\dagger\boldsymbol{A} = \boldsymbol{A}$
2. $\boldsymbol{A}^\dagger\boldsymbol{A}\boldsymbol{A}^\dagger = \boldsymbol{A}^\dagger$
3. $\boldsymbol{A}^\dagger\boldsymbol{A}$ and $\boldsymbol{A}\boldsymbol{A}^\dagger$ are Hermitian matrices

From the above conditions, the pseudo inverse is unique; i.e. it exists exactly one matrix $\boldsymbol{A}^\dagger$ that fulfills the criteria above. Furthermore, if $\boldsymbol{A}$ is an invertible matrix, then its inverse is the Moore-Penrose pseudo-inverse $\boldsymbol{A}^{-1} = \boldsymbol{A}^\dagger$.

For a system of linear equations, the pseudo inverse answers to the least squares solution. By the normal equations, it follows that:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \quad \Longleftrightarrow \quad \boldsymbol{x} = \boldsymbol{A}^\dagger\boldsymbol{b}$$

where the solution satisfies the normal equations ($\boldsymbol{A}^H\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^H\boldsymbol{b}$). For the case of $\boldsymbol{A}$ being invertible, this coincides with the well known solution $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$.

Non-negative Tucker Decomposition Derivation

Below follows an extension of the section 2.2 *The Tucker Decomposition - Non Negative Tucker Decomposition.* It accounts for a derivation of the update rule of the core tensor $\mathcal{G}$.

Deriving the update rule of the second factor, we are interested in $\mathbf{G}_{(n)}$ and not $\mathbf{G}_{\boldsymbol{A}}^{(-n)}$. Therefore, we rewrite the expression in (2.28) as $\boldsymbol{X}_{(n)} \approx \boldsymbol{A}^{(n)} \boldsymbol{G}_{(n)} \boldsymbol{A}_{\otimes}^{(-n)^T}$. To arrive at the final expression of $\mathcal{G}$, we utilize the following property:

$$\mathrm{vec}(\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T) = (\boldsymbol{V} \otimes \boldsymbol{U})\mathrm{vec}(\boldsymbol{S}) \tag{8.3}$$

which for a vectorization of $\boldsymbol{X}_{(n)}$ yields [Kim and Choi, 2007]:

$$\begin{aligned} \mathrm{vec}(\boldsymbol{X}_{(n)}) &\approx \mathrm{vec}(\boldsymbol{A}^{(n)} \boldsymbol{G}_{(n)} \boldsymbol{A}_{\otimes}^{(-n)^T}) \\ &= (\boldsymbol{A}_{\otimes}^{(-n)} \otimes \boldsymbol{A}^{(n)})\mathrm{vec}(\boldsymbol{G}_{(n)}) \end{aligned} \tag{8.4}$$

By [Lee and Seung, 1999], this answers to the updating rule:

$$\begin{aligned} \mathrm{vec}(\boldsymbol{G}_{(n)}) &\leftarrow \mathrm{vec}(\boldsymbol{G}_{(n)}) * \boldsymbol{K} \\ \boldsymbol{K} &= \frac{(\boldsymbol{A}_{\otimes}^{(-n)} \otimes \boldsymbol{A}^{(n)})^T \mathrm{vec}(\boldsymbol{X}_{(n)})}{(\boldsymbol{A}_{\otimes}^{(-n)} \otimes \boldsymbol{A}^{(n)})^T (\boldsymbol{A}_{\otimes}^{(-n)} \otimes \boldsymbol{A}^{(n)})\mathrm{vec}(\boldsymbol{G}_{(n)})} \end{aligned} \tag{8.5}$$

where the factor $\boldsymbol{K}$ can be simplified using (2.28) and (8.7):

$$\begin{aligned} \boldsymbol{K} &= \frac{\mathrm{vec}(\boldsymbol{A}^{(n)^T} \boldsymbol{X}_{(n)} \boldsymbol{A}_{\otimes}^{(-n)})}{\mathrm{vec}(\boldsymbol{A}^{(n)^T} \boldsymbol{A}^{(n)} \boldsymbol{G}_{(n)} \boldsymbol{A}_{\otimes}^{(-n)^T} \boldsymbol{A}_{\otimes}^{(-n)})} \\ &= \frac{\mathrm{vec}\left(\left[\boldsymbol{\mathcal{X}} \times_1 \boldsymbol{A}^{(1)^T} \cdots \times_N \boldsymbol{A}^{(N)^T}\right]_{(n)}\right)}{\mathrm{vec}\left(\left[\boldsymbol{\mathcal{G}} \times_1 (\boldsymbol{A}^{(1)^T} \boldsymbol{A}^{(n)}) \cdots \times_N (\boldsymbol{A}^{(N)^T} \boldsymbol{A}^{(N)})\right]_{(n)}\right)} \end{aligned} \tag{8.6}$$

To arrive at the final expression of $\mathcal{G}$, we utilize the following property:

$$\mathrm{vec}(\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T) = (\boldsymbol{V} \otimes \boldsymbol{U})\mathrm{vec}(\boldsymbol{S}) \tag{8.7}$$

which for a vectorization of $\boldsymbol{X}_{(n)}$ yields [Kim and Choi, 2007]:

$$\begin{aligned} \mathrm{vec}(\boldsymbol{X}_{(n)}) &\approx \mathrm{vec}(\boldsymbol{A}^{(n)} \boldsymbol{G}_{(n)} \boldsymbol{A}_{\otimes}^{(-n)^T}) \\ &= (\boldsymbol{A}_{\otimes}^{(-n)} \otimes \boldsymbol{A}^{(n)})\mathrm{vec}(\boldsymbol{G}_{(n)}) \end{aligned} \tag{8.8}$$

For the tensor $\mathcal{G}$, the update rule in (8.5) thereby translates to [Kim and Choi, 2007]:

$$\mathcal{G} \leftarrow \mathcal{G} * \frac{\mathcal{X} \times_1 \boldsymbol{A}^{(1)^T} \times_2 \cdots \times_N \boldsymbol{A}^{(N)^T}}{\mathcal{G}_k \times_1 \boldsymbol{A}^{(1)^T} \boldsymbol{A}^{(1)} \times_2 \cdots \times_N \boldsymbol{A}^{(N)^T} \boldsymbol{A}^{(N)}} \tag{8.9}$$

8.2 Euclidean Geometry

Euclidean geometry is the commonly taught geometry that is intuitive to visualize. In Euclidean geometry, there exists a 3-dimensional space which is built up by points, lines, surfaces and solids. Each step adds one dimension, so from a point with no dimension, a line has one, a surface has two and a solid has three. Euclid defined five postulates for how these shapes can interact [National Council of Educational & Research Training, 2021]:

1. Given two points, there is a straight line that joins them.
2. A straight line segment can be extended indefinitely.
3. A circle can be constructed from a center point and a distance for its radius.
4. All right angles are equal.
5. If a straight line falling on two straight lines makes the interior angles on the same side of it taken together less than two right angles, then the two straight lines, if produced indefinitely, meet on that side on which the sum of angles is less than two right angles.

8.3 Riemannian Geometry

Description

Riemannian geometry (sometimes referred to as *elliptic geometry*) is a non-Euclidean geometry. To describe Riemannian geometry, it is appropriate to start by defining a *manifold*. A manifold is a locally Euclidean topological space. This means that the manifold itself does not have to fulfill the Euclidean constraints, but the tangent space will. A good visualizing example of this is the earth, which has a spherical shape, but when looking at a small, local area, it appears to be flat [Rowland, n.d.]

The *Riemannian manifold* is a C^∞ (infinitely differentiable) curved manifold, where differential calculus and a change of coordinates can be performed [Berger, 2002]. An evident difference between Euclidean and Riemannian geometry is that Euclid's fifth postulate is no longer valid. The fifth postulate implies that through a point not on a given line, there is only one line parallel to the given line. In Riemannian geometry, there are no parallel lines. Additionally, the sum of angles in a triangle is not equal to the sum of two right angles as in the Euclidean case, but in fact larger [Encyclopaedia Britannica, 2007].

Application on EEG Data

Riemannian Geometry has showed to be a useful tool to describe EEG data. Let $\boldsymbol{X}_i \in \mathbb{R}^{C \times T_s}$ denote a trial of EEG data, measured from C channels with T_s time samples. The covariance matrix of the trial can be considered a feature, describing characteristics of this specific trial. It can be estimated using for example a Sample Covariance Matrix (SCM), see (8.10):

$$\Sigma_i = \frac{1}{T_s - 1} \boldsymbol{X}_i \boldsymbol{X}_i^T \tag{8.10}$$

Such a covariance matrix belongs to the space of symmetric positive definite (SPD) matrices, which is a Riemannian manifold [Bolagh and Clifford, 2017].

To compare the similarity of two covariance matrices Σ_i and Σ_j in the Riemannian manifold, the Riemannian distance between them can be computed as:

$$\delta_R(\Sigma_i, \Sigma_j) = ||\log(\Sigma_i^{-1/2} \Sigma_j \Sigma_i^{-1/2})||_F = \left[\sum_{c=1}^{C} \log^2 \lambda_c \right] \tag{8.11}$$

where $||\cdot||_F$ denotes the Frobenius norm, λ_c are the real eigenvalues of $\Sigma_i^{-1/2} \Sigma_j \Sigma_i^{-1/2}$ and C the number of channels [Bolagh and Clifford, 2017].

To create a covariance matrix representation of a specific subject or session from several trials, a mean covariance matrix can be calculated using the Riemannian geometric mean $\mathfrak{G}$. To describe this mean, an optimization problem can be formulated as:

$$\mathfrak{G}(\Sigma_1 \ldots \Sigma_I) = \underset{\Sigma}{\operatorname{argmin}} \sum_{i=1}^{I} \delta^2(\Sigma, \Sigma_i) \tag{8.12}$$

There is no closed form solution to this equation for $I > 2$, that is, for finding the mean covariance matrix from a session containing more than two trials [Bolagh and Clifford, 2017]. It can although be computed iteratively, following an algorithm described in [Barachant et al., n.d.] The method consists of three steps which are performed until convergence. Firstly, the covariance matrices in the Riemannian manifold are projected to the tangent space of the estimated Riemannian mean. Secondly, the arithmetic mean in the tangent space is calculated. Thirdly, the arithmetic mean is projected back to the Riemannian space and used as the updated Riemannian mean.

8.4 Results

Tensor Decompositions for Session Selection

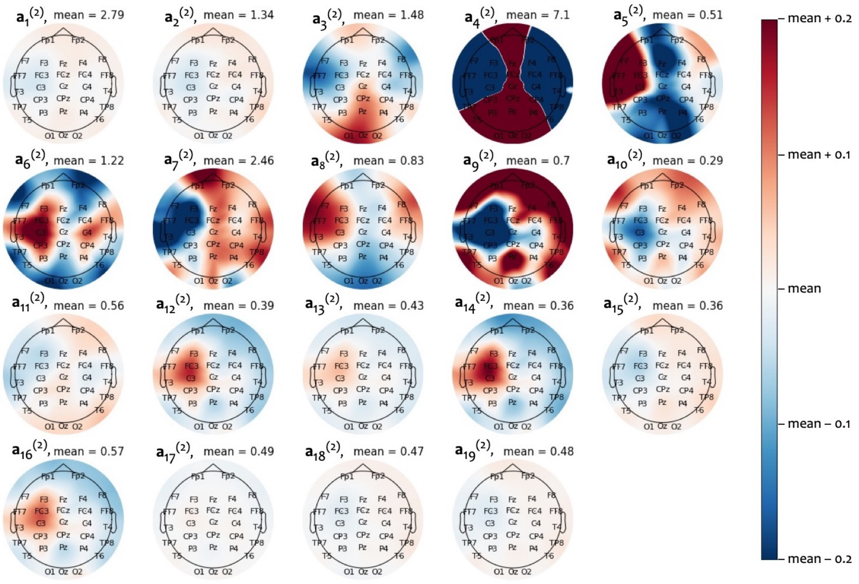


Figure 8.1 The columns $\mathbf{a}_r^{(2)}$ $(r = 1, \ldots, 19)$ corresponding to the factor matrix $\mathbf{A}^{(2)}$ in the CP decomposition of the SA Drivers dataset. The colors of the topomaps represent the deviation from the column's mean value (see colorbar).

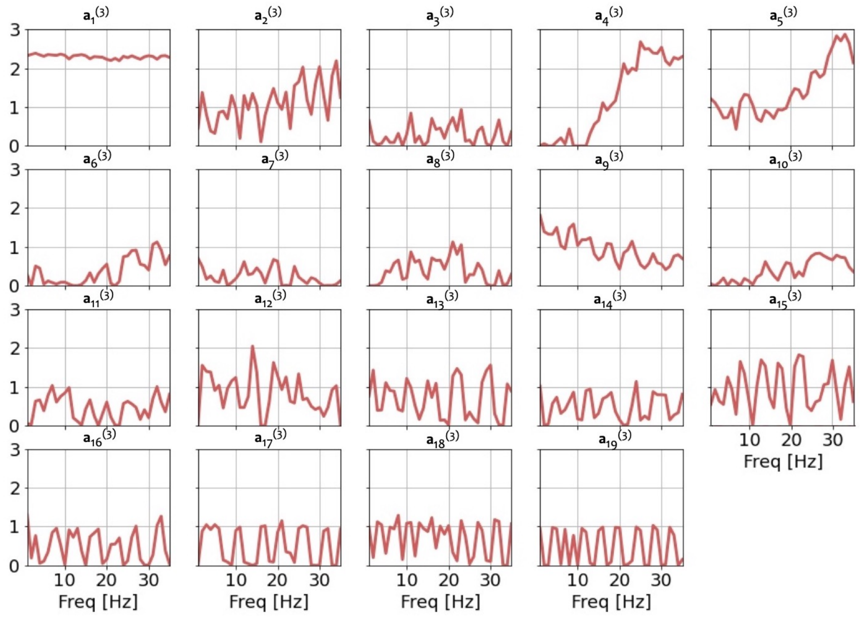


Figure 8.2 The columns $\mathbf{a}_r^{(3)}$ $(r = 1, \ldots, 19)$ corresponding to the factor matrix $\mathbf{A}^{(3)}$ in the CP decomposition of the SA Drivers dataset.

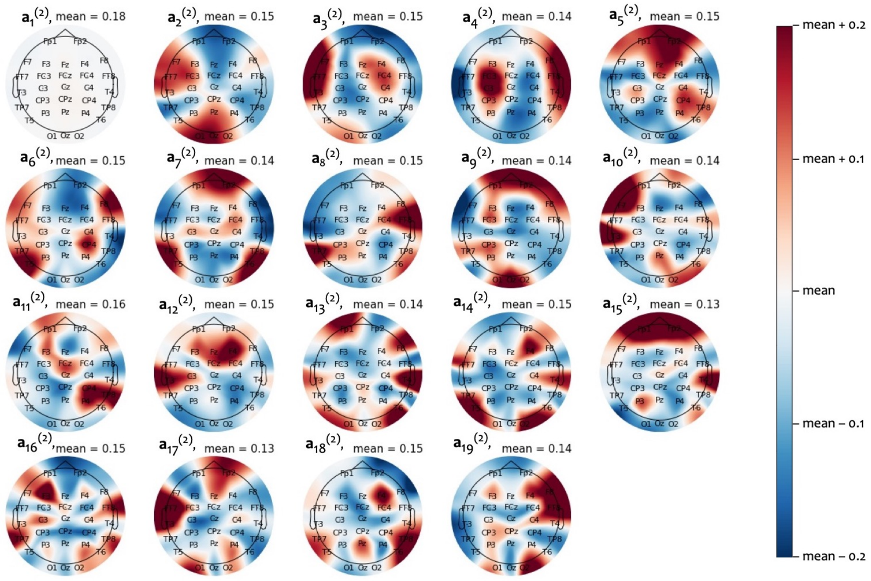


Figure 8.3 The columns $\mathbf{a}_{r_2}^{(2)}$ ($r_2 = 1, ..., 19$) corresponding to the factor matrix $\mathbf{A}^{(2)}$ in the Tucker decomposition of the SA Drivers dataset. The colors of the topomaps represent the deviation from the column's mean value (see colorbar).

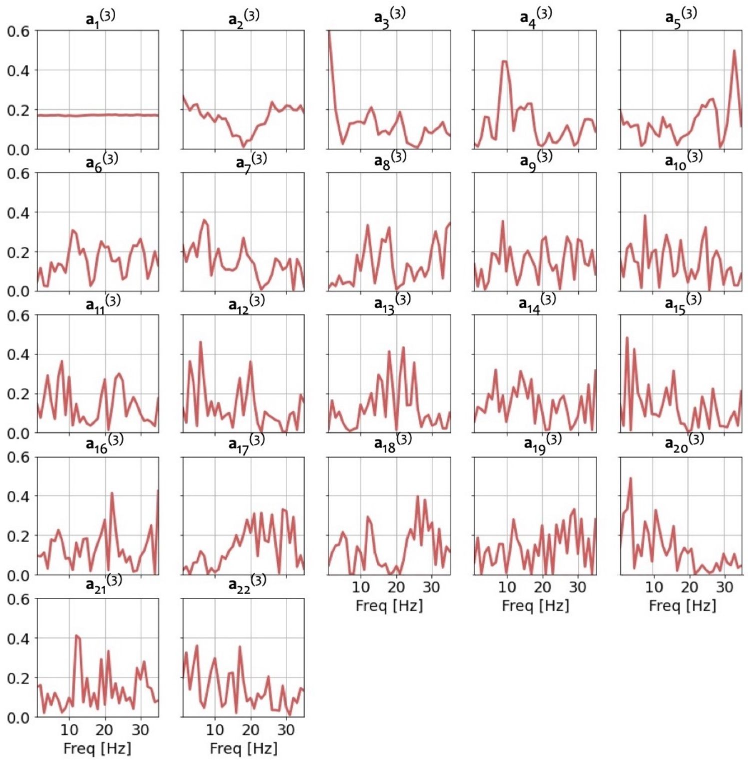


Figure 8.4 The columns $\mathbf{a}_{r_3}^{(3)}$ ($r_3 = 1, \ldots, 22$) corresponding to the factor matrix $\mathbf{A}^{(3)}$ in the Tucker decomposition of the SA Drivers dataset.

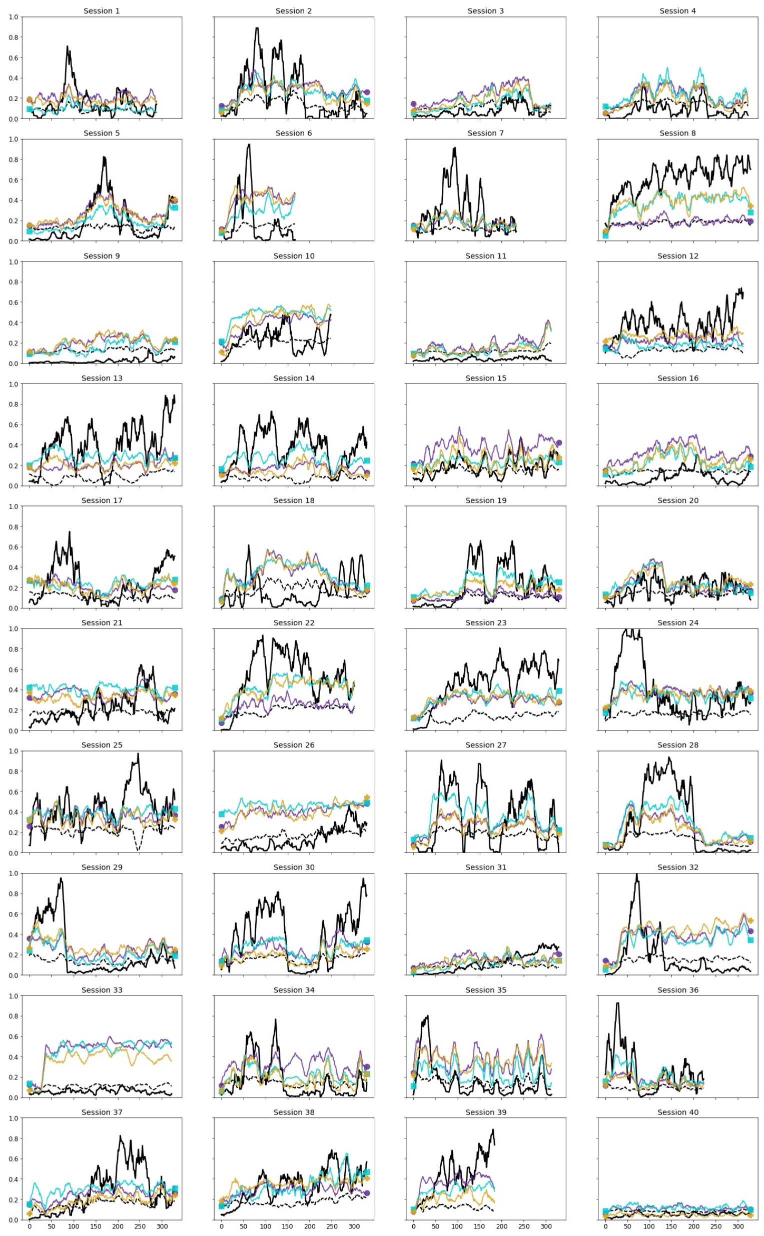


Figure 8.5 Prediction result for all sessions with a Random Forest regressor, trained with a subset of 8 sessions. See legend in Figure 5.14 for colors/symbols corresponding to each selection method.

Tensor Decompositions for Feature Extraction and Session Selection

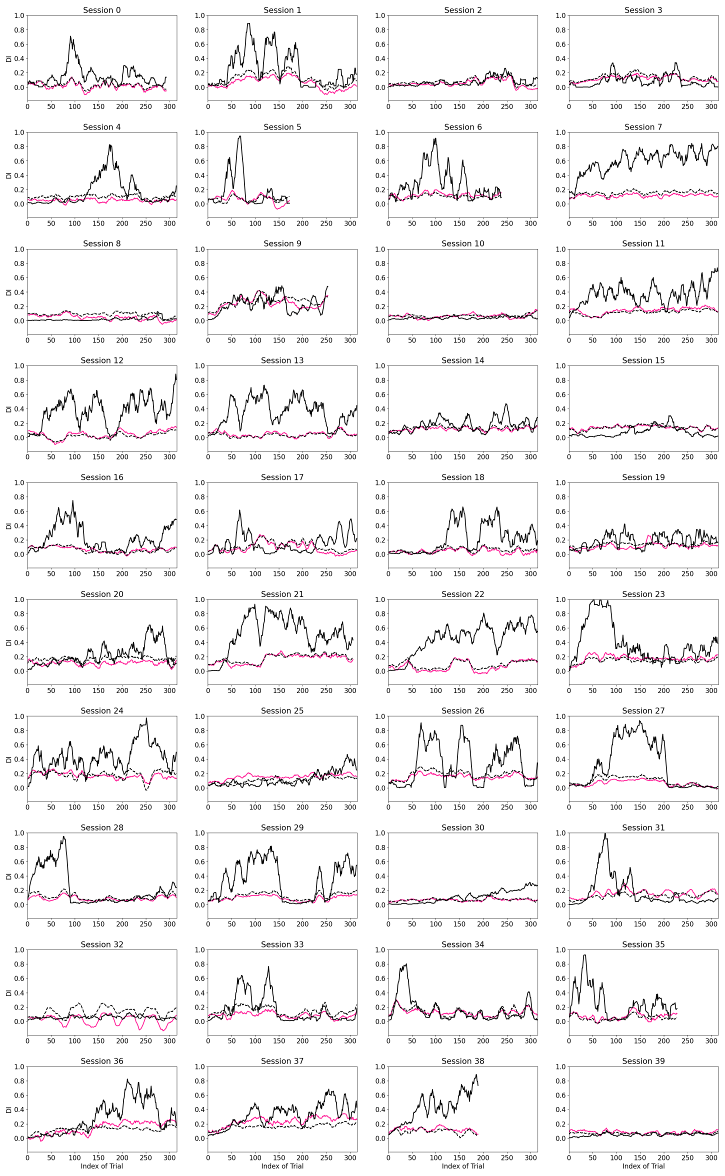


Figure 8.6 Prediction result for all sessions with a Linear SVR, trained with a subset of 10 sessions. The solid pink line answers to the prediction using the proposed (TFE) method, the dashed black line to when using random selection, and the solid black line to the true DI.

Lund University **Department of Automatic Control** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER'S THESIS *Date of issue* June 2022 *Document Number* TFRT-6172
Author(s) Emma Fallenius Linda Karlsson	*Supervisor* Carolina Bergeling, Dept. of Automatic Control, Lund University, Sweden Bo Bernhardsson, Dept. of Automatic Control, Lund University, Sweden Pontus Giselsson, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle

Tensor Decompositions of EEG Signals for Transfer Learning Applications

Abstract

In this report, tensor decomposition methods of EEG signals have been evaluated for the purpose of transfer learning. The aim has been to address the person-to-person Brain-Computer Interface (BCI) calibration problem by transferring training data between sessions, which can shorten calibration times, extend the amount of training data, and enable using data from simulated environments in real world applications. For this, the datasets AlexMI (binary motor imagery) and SA Driving (drowsiness detection during simulated driving) have been analyzed. Tensor decompositions were performed unsupervised in two pipelines, with aim of capturing universal structures relevant to BCI tasks.
For the first pipeline, two decompositions (Canonical Polyadic and Tucker) were computed to compare similarity between sessions. From that, a subset of sessions were selected that during classification, were aimed to outperformrandom selection and training with the full training database. In the first pipeline, a new similarity measure was designed, which included weighting of the factor matrix in the mode of interest. We consider this a more representative measure of how similar two sessions are, compared to simply studying the unweighted factor matrices, which was done in previous literature. For the second pipeline, one tensor decomposition (Tucker) was used for feature extraction and similarity comparison between sessions. The aim was the same as for pipeline one, with the addition of investigating the properties of tensor decompositions as features. The results show that unsupervised tensor decompositions can extract structures of varying relevance to a classification problem but did not result in superior performance when used as features.With this knowledge, we propose extending tensor decompositions to supervised and/or nonlinear ones. Additionally, the proposed session selection methods showed potential in classification, but no significant conclusions could be drawn of their superiority compared to random selection or trainingwith the full database. Additionally, the classifiers had a large variation in performance between sessions, making them far from applicable for a BCI in a real-world environment today.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 1-129	*Recipient's notes*
Security classification		

http://www.control.lth.se/publications/