



universität  
wien

## MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Analysis of an EEG Database of Depression Patients by means  
of Graphical Granger Causality“

verfasst von / submitted by

Christina Pacher BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of  
Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt /  
degree programme code as it appears on  
the student record sheet:

UA 066 921

Studienrichtung lt. Studienblatt /  
degree programme as it appears on  
the student record sheet:

Masterstudium Informatik UG2002

Betreut von / Supervisor:

Univ.-Prof. Dipl-Inform. Univ. Dr. Claudia Plant

Mitbetreut von / Co-Supervisor:

RNDr. CSc. Katerina Schindlerova



# Acknowledgements

I would like to thank my supervisors Prof. Claudia Plant and Dr. Katerina Schindlerova ( = Kateřina Hlaváčková-Schindler) for their support in the creation of this thesis. Special thanks go to Dr. Schindlerova, whose input and feedback were an invaluable aid along all steps of the process.

I would also like to thank our partners from the international research project that this thesis is a part of: Dr. Milan Paluš, Dr. Jaroslav Hlinka, and Dr. Aditi Kathpalia from the Czech Academy of Sciences, and Dr. Martin Brunovsky from the National Institute of Mental Health of the Czech Republic.



# Abstract

This master thesis was created as part of the international research project "Learning Synchronization Patterns in Multivariate Neural Signals for Prediction of Response to Antidepressants", a joint international project by the University of Vienna, the Czech Academy of Sciences, and the National Institute of Mental Health of the Czech Republic. This project investigates possibilities to predict the response of depression patients to antidepressant treatment by applying data mining and machine learning methods to EEG data.

The approach that is employed in this thesis uses graphical Granger causality based on stochastic complexity, as introduced by Hlaváčková-Schindler and Plant in 2020, to extract features in the form of Granger-causal networks from EEG recordings. A Python implementation of this method was created as part of this thesis. As it was designed with focus on ease of use and extendability and has been made publicly available at GitHub, it will hopefully also prove to be a valuable resource for other projects looking to apply the same method.

The new Python implementation was used to extract features from a data set of EEG recordings of 176 depression patients, split into a training set of 134 subjects and a test set of 42 subjects. In an exploratory analysis, the extracted features were used as input for the classification problem of differentiating between treatment responders and nonresponders. A number of different variations of the feature extraction process were carried out, encompassing the application of various preprocessing methods to the EEG signal and the use of subsets of the available electrode channels. Five common classifiers (the support vector, K nearest neighbors, decision tree, random forest, and naive bayes classifiers) were trained on the resulting features using a cross-validation approach.

Initially, the classifiers were not able to predict treatment response with satisfying accuracy, as they did not manage to outperform baseline results created by dummy classifiers. Splitting the data set by biological sex and training the classifiers for male and female subjects separately improved the classification quality. These results did not generalize well to the test set of previously unseen data. Still, the fact that this split improved the classification result on the training set is a valuable insight that may also inform future work within the research project. This thesis furthermore discusses possible remedies that could increase the classification quality of the approach and sets objectives for future research based on the acquired insights.



# Kurzfassung

Diese Masterarbeit ist Teil des internationalen Forschungsprojekts "Learning Synchronization Patterns in Multivariate Neural Signals for Prediction of Response to Antidepressants". Hierbei handelt es sich um ein gemeinsames Projekt der Universität Wien, der Tschechischen Akademie der Wissenschaften und des Nationalen Instituts für psychische Gesundheit der Tschechischen Republik. Ziel dieses Projekts ist es, den Erfolg von Behandlungsmaßnahmen gegen Depressionen vorherzusagen, indem Methoden aus den Bereichen Data Mining und Machine Learning auf EEG-Gehirnstrommessungen angewendet werden.

In dieser Masterarbeit kommt eine auf dem Kriterium der stochastic complexity basierende Variation von graphical Granger causality zum Einsatz, die 2020 von Hlaváčková-Schindler und Plant vorgestellt wurde. Mithilfe dieser Methode werden Granger-kausale Graphen aus den EEG-Daten berechnet. Eine Python-Implementierung dieser Methode wurde im Rahmen dieser Masterarbeit entwickelt. Die Implementierung legt Fokus auf Benutzerfreundlichkeit und leichte Erweiterbarkeit und steht auf der Plattform GitHub öffentlich zur Verfügung. Sie kann somit auch für andere Anwendungen dieser Methode zukünftig eine wertvolle Unterstützung sein.

Mithilfe der neuen Implementierung wurden Feature-Vektoren aus einem Datenset extrahiert, das EEG-Messungen von 176 Patient\*innen mit Depressionen enthält (aufgeteilt in ein Trainingsset aus 134 und ein Testset aus 42 Patient\*innen). Diese Feature-Vektoren wurden in einer explorativen Analyse als Input für das Klassifizierungsproblem zwischen Behandlungserfolg und -misserfolg verwendet. Mehrere Varianten wurden erforscht, bei denen z.B. verschiedene Methoden zur Datenvorverarbeitung zum Einsatz kamen oder unterschiedliche Teilmengen der vorhandenen Elektroden verwendet wurden. Fünf gängige Klassifizierungsmethoden (support vector, K nearest neighbors, decision tree, random forest und naive bayes) wurden mithilfe einer Kreuzvalidierung für die Vorhersage trainiert.

Auf Basis des ursprünglichen Datensets war es nicht möglich, den Behandlungserfolg mit zufriedenstellender Genauigkeit vorherzusagen: Keine der Klassifizierungsmethoden konnte die Dummy-Ergebnisse übertreffen, die als Vergleichswert dienten. Die Ergebnisse konnten verbessert werden, indem das Datenset nach biologischem Geschlecht unterteilt wurde und die Klassifizierung für männliche und weibliche Patient\*innen getrennt durchgeführt wurde. An dem separaten Test-Datenset ließen sich die Ergebnisse nicht reproduzieren. Dennoch handelt es sich bei der Tatsache, dass diese Aufteilung das Ergebnis für das Trainings-Datenset verbessern konnte, um eine wertvolle Erkenntnis, auch in Hinblick auf die weitere Arbeit im Rahmen des Forschungsprojekts. Weiters bespricht diese Arbeit basierend auf den gewonnenen Erkenntnissen verschiedene Möglichkeiten, die die Qualität der Ergebnisse verbessern könnten, und setzt neue Ziele für weiterführende Experimente.



# Contents

|                                                                       |             |
|-----------------------------------------------------------------------|-------------|
| <b>Acknowledgements</b>                                               | <b>i</b>    |
| <b>Abstract</b>                                                       | <b>iii</b>  |
| <b>Kurzfassung</b>                                                    | <b>v</b>    |
| <b>List of Tables</b>                                                 | <b>ix</b>   |
| <b>List of Figures</b>                                                | <b>xi</b>   |
| <b>List of Algorithms</b>                                             | <b>xiii</b> |
| <b>Listings</b>                                                       | <b>xv</b>   |
| <b>1. Introduction</b>                                                | <b>1</b>    |
| <b>2. Related Work</b>                                                | <b>3</b>    |
| <b>3. Background</b>                                                  | <b>7</b>    |
| 3.1. Granger Causality . . . . .                                      | 7           |
| 3.2. Multivariate Granger Causality . . . . .                         | 8           |
| 3.3. Granger Causality with Information-Theoretic Criterion . . . . . | 8           |
| 3.4. Limitations and Challenges . . . . .                             | 10          |
| <b>4. Method</b>                                                      | <b>13</b>   |
| 4.1. EEG Recordings . . . . .                                         | 13          |
| 4.1.1. Ethics Statement . . . . .                                     | 14          |
| 4.2. Granger Causality Computations . . . . .                         | 15          |
| 4.3. Classification based on Granger Causality Results . . . . .      | 17          |
| <b>5. Implementation</b>                                              | <b>21</b>   |
| 5.1. API Convention . . . . .                                         | 21          |
| 5.2. Class Structure . . . . .                                        | 22          |
| 5.3. UI and the Wrapper Class . . . . .                               | 22          |
| 5.4. Information-theoretic Criteria . . . . .                         | 25          |
| 5.4.1. The Criterion Duck Typing Interface . . . . .                  | 26          |
| 5.4.2. Stochastic Complexity . . . . .                                | 26          |
| 5.5. Optimizers . . . . .                                             | 27          |
| 5.5.1. Exhaustive Search . . . . .                                    | 28          |
| 5.5.2. Genetic Algorithm . . . . .                                    | 29          |
| <b>6. Experiments</b>                                                 | <b>35</b>   |
| 6.1. Granger Causality Computations . . . . .                         | 35          |
| 6.1.1. Proof of Concept: Synthetic Data Sets . . . . .                | 35          |
| 6.1.2. Parameter Estimation . . . . .                                 | 37          |
| 6.2. Classification Experiments . . . . .                             | 42          |
| 6.2.1. Splitting the Data Set . . . . .                               | 42          |

## Contents

|                                                    |           |
|----------------------------------------------------|-----------|
| 6.2.2. Test Set Evaluation . . . . .               | 43        |
| 6.3. Discussion . . . . .                          | 45        |
| 6.3.1. Limitations of the Work . . . . .           | 46        |
| 6.3.2. Contributions of the Work . . . . .         | 47        |
| <b>7. Conclusion</b>                               | <b>49</b> |
| 7.1. Future Work . . . . .                         | 49        |
| <b>Bibliography</b>                                | <b>51</b> |
| <b>A. Appendix</b>                                 | <b>57</b> |
| A.1. Depression Treatment Methods . . . . .        | 57        |
| A.2. Test Subjects . . . . .                       | 57        |
| A.3. Hyperparameters for the Classifiers . . . . . | 57        |

# List of Tables

|                                                                                        |    |
|----------------------------------------------------------------------------------------|----|
| 5.1. Constructor keyword arguments for the GrangerCausality class. . . . .             | 24 |
| 5.2. Description of the constructor arguments of the StochasticComplexity class. . . . | 27 |
| 6.1. Results of the lag estimation process for each input data setting. . . . .        | 40 |
| 6.2. Settings used for the test set evaluation. . . . .                                | 45 |
| A.1. Subject distribution in the training set. . . . .                                 | 57 |
| A.2. Subject distribution in the test set. . . . .                                     | 57 |



# List of Figures

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |    |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.1. | Example for how the ITGGM method expresses a hypothesis about Granger-causal relationships. In this example, $i = 3$ , which means that the target time series is $x_3$ . The vector $Q_3$ describes the presumed Granger-causal influences on $x_3$ . In this case, $x_1$ , $x_3$ , and $x_4$ are Granger-causal for $x_3$ , while $x_2$ is not Granger-causal for $x_3$ . The graph to the right visualizes these relationships. . . . .                                                      | 9  |
| 3.2. | Example for how the Granger-causal relationships between time series are represented in the ITGGM method. The matrix on the left describes the relationships in the format that is created as output of the ITGGM method. On the right, these relationships are visualized in graph form. . . . .                                                                                                                                                                                               | 10 |
| 4.1. | The 19 standardized electrode locations on the scalp that were used for the EEG recordings. The used positions are Fp1, Fp2, F3, F4, C3, C4, P3, P4, O1, O2, F7, F8, T3, T4, T5, T6, Fz, Cz, and Pz. . . . .                                                                                                                                                                                                                                                                                    | 14 |
| 4.2. | Example EEG data showing the first two seconds of a subject's recording. To improve legibility, only data from three electrodes is shown. The top plot shows the unfiltered data. The other two plots show the same data after it has been filtered to include only frequencies in the alpha band (center) or in the theta band (bottom). . . . .                                                                                                                                               | 16 |
| 5.1. | Class diagram showing the relationships between the classes in the implementation. In this diagram, <b>Optimizer</b> and <b>Criterion</b> are generic placeholders for any concrete optimizer and criterion classes. Accordingly, only the methods defined by the duck typing interfaces are shown for these classes. . . . .                                                                                                                                                                   | 23 |
| 5.2. | Genetic algorithm operators . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 32 |
| 6.1. | True Granger-causal adjacency matrices for the synthetic data sets. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                     | 36 |
| 6.2. | Results of the Granger causality computations with exhaustive search on the synthetic data sets. False positives are marked in color (the results do not contain any false negatives). The five adjacency matrices per data set correspond to five versions of this data set that have different random noise components. . . . .                                                                                                                                                               | 37 |
| 6.3. | Comparison of the Granger causality computations with genetic algorithm and exhaustive search on the synthetic ten-point network. . . . .                                                                                                                                                                                                                                                                                                                                                       | 37 |
| 6.4. | Demonstration of the lag estimation process on synthetic data. Three versions of each data set were generated to account for possible effects of the random noise components in the data. In each plot, the true lag is marked by a dashed line, while the minimum score is indicated by a green dot. . . . .                                                                                                                                                                                   | 39 |
| 6.5. | Lag estimation process for data from the left brain hemisphere, using the unfiltered EEG data. Left: Average stochastic complexity scores achieved with different lags on each of the sample epochs. The minimum result for each sample epoch is marked with a green dot. Right: Bar plot that shows how many epochs had their minimum score at each lag value. The value 3 was the optimal lag for the majority of the sample epochs, and therefore this value was chosen as the final lag. 40 |    |

## List of Figures

|       |                                                                                                                                                                                                                                                                                                                                                                                                         |    |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 6.6.  | Estimation process for the mutation probability on the unfiltered full-brain data. Top: Average stochastic complexity scores achieved on the sample epochs for different mutation probabilities. The minimum score for each epoch is marked by a black square. Bottom: Bar plot that shows how many epochs had their minimum score at each value. Both plots show that there is no distinctive optimum. | 41 |
| 6.7.  | Dummy classification results, showing the average and standard deviation of the F1 score over a 10-fold cross validation.                                                                                                                                                                                                                                                                               | 42 |
| 6.8.  | Best results achieved by any of the classifiers on any of the input data. The plots show the mean and standard deviation of the F1 scores produced by a 10-fold cross validation.                                                                                                                                                                                                                       | 43 |
| 6.9.  | Classification results for female subjects. The plots show the mean and standard deviation of the F1 scores produced by a 5-fold cross validation.                                                                                                                                                                                                                                                      | 44 |
| 6.10. | Classification results for male subjects. The plots show the mean and standard deviation of the F1 scores produced by a 5-fold cross validation.                                                                                                                                                                                                                                                        | 44 |
| 6.11. | Confusion matrices comparing the prediction results for the test sets to the true class labels.                                                                                                                                                                                                                                                                                                         | 46 |

# List of Algorithms

|    |                                                                |    |
|----|----------------------------------------------------------------|----|
| 1. | Summary of the Granger causality computation process . . . . . | 18 |
| 2. | Summary of the classifier training process . . . . .           | 20 |



# Listings

|       |                                                                                                                                                                                                                                                                                                                         |    |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 5.1.  | Minimal code required to compute Granger-causal relationships in a data set. . .                                                                                                                                                                                                                                        | 23 |
| 5.2.  | Constructor signature of the GrangerCausality class. . . . .                                                                                                                                                                                                                                                            | 24 |
| 5.3.  | Example instantiation of the GrangerCausality class with user-defined arguments.<br>In this case, the user wants to use the stochastic complexity criterion with a lag of<br>5, and a genetic algorithm optimization that runs for 20 generations, while leaving<br>all other settings at their default values. . . . . | 24 |
| 5.4.  | The fit method of the GrangerCausality class. This method is responsible for<br>instantiating and coordinating the optimizer and criterion objects. . . . .                                                                                                                                                             | 25 |
| 5.5.  | Method signature of the score method, with type annotations, as defined by the<br>criterion duck typing interface. . . . .                                                                                                                                                                                              | 26 |
| 5.6.  | Constructor signature of the StochasticComplexity class. . . . .                                                                                                                                                                                                                                                        | 27 |
| 5.7.  | Constructor signature of the ExhaustiveSearch class. . . . .                                                                                                                                                                                                                                                            | 28 |
| 5.8.  | The fit method of the ExhaustiveSearch class. . . . .                                                                                                                                                                                                                                                                   | 28 |
| 5.9.  | Constructor signature of the GeneticAlgorithm class. . . . .                                                                                                                                                                                                                                                            | 33 |
| 5.10. | The fit method of the GeneticAlgorithm class. . . . .                                                                                                                                                                                                                                                                   | 33 |



# 1. Introduction

Depression is a common mental illness worldwide: According to the World Health Organization [1], an estimated 5% of adults around the globe suffer from depression. Over 700000 people commit suicide every year as a direct consequence of depression.

Next to therapeutic treatment approaches, such as interpersonal psychotherapy or cognitive behavioural therapy, antidepressants also play an important role in the treatment of depression. However, treatment with antidepressants often takes weeks to produce a response, and the likelihood of producing a response at all is not very high (only up to 65%) [2]. This is the motivation for the research project "Learning Synchronization Patterns in Multivariate Neural Signals for Prediction of Response to Antidepressants" [3, 4], for short the "Synchronization project". It is a joint international project by the University of Vienna, in particular the research group "Data Mining and Machine Learning" at the Institute of Computer Science, together with the Czech Academy of Sciences and the National Institute of Mental Health of the Czech Republic. The main goal of this project is to analyze a data base of brain scan data that was recorded from 176 patients who are being treated for depression, and to find out if it is possible to predict a patient's ultimate treatment response from this brain data. A number of different treatment approaches were used on the subjects in this data set, including a variety of antidepressant drugs, but also brain stimulation methods using electric or magnetic currents.

The data was recorded using non-invasive electroencephalography (EEG), a technique that measures electrical fields in the brain that are caused by the activation of large numbers of neurons. While there are invasive forms of EEG, where sensors are inserted surgically into the patient's head, this approach is rather risky for the subject and is therefore very rarely used. For the more common non-invasive version, the signal is recorded via electrodes that are placed on the subject's scalp; the signal is less clear than with its invasive counterpart, but the recording process is basically risk-free and has comparatively small requirements in terms of cost and equipment [5]. The output of an EEG consists of a number of time series or "channels" describing the electrical signal measured at the different electrode locations over time.

If the Synchronization project is successful in its aim to predict subjects' response to depression treatment, this could lead to a vast improvement for the lives of people afflicted with depression. The ability to determine early on whether a treatment approach will be successful could drastically reduce the time until an effective medication is found, thus reducing the amount of time that patients have to suffer from untreated or wrongly treated symptoms.

In the frame of this research project, multiple different methods to analyze the EEG data are investigated. One of them is the focus of this master thesis. In particular, a multivariate Granger-causal model [6, 7] is used to find networks of Granger-causal relationships between the multivariate time series in an EEG recording. Subsequently, five common classifiers (the support vector, K nearest neighbors, decision tree, random forest, and naive bayes classifiers) are trained to distinguish between treatment responders and nonresponders based on the discovered Granger-causal networks.

The first main scientific contribution of this thesis is the utilization of the method detailed in [7] for a classification problem. The second main scientific contribution is the exploratory analysis of the classification performance of the five classifiers to separate responders and nonresponders based on features in the form of Granger causality networks. Various signal transformations were applied to the EEG data and their influence on the performance of the classifiers was studied.

Additionally, a software library was created that implements this method in Python 3 [8]. While the authors of [7] provided a Matlab implementation of their method, to the best of my

## *1. Introduction*

knowledge no implementation in a free-to-use programming language existed until now. The implementation created in the frame of this thesis is publicly available, follows well-established API conventions for data mining libraries, and is easy to extend, thus hopefully also making this method more easily available for other scientific applications in the future.

The thesis is structured as follows: Chapter 2 summarizes related work. Chapter 3 introduces the necessary mathematical background, and Chapter 4 explains in detail the method that is used to discover the Granger-causal networks and perform the classification. The implementation that was created to compute the Granger-causal networks is described in Chapter 5. Chapter 6 presents the experiment results. Finally, Chapter 7 summarizes the thesis and provides an outlook for possible future avenues of research.

## 2. Related Work

Classification of diseases based on brain scan data has been the focus of a large number of scientific studies over the years. This is not only limited to using EEG data, but has also been done with e.g. functional magnetic resonance imaging (fMRI) or magnetoencephalography (MEG) data. These techniques measure different kinds of signals (electrical vs. magnetic fields) and have different temporal and spatial resolutions, but the underlying idea is always to measure a certain kind of signal at different locations of the brain. A variety of neurological diseases have been examined in such experiments, including for example Alzheimer’s disease, Mild Cognitive Impairment, or ADHD (e.g. [9, 10]).

A large amount of research focuses specifically on EEG data and depressive disorders. Most of these studies aim to differentiate between patients with depression and healthy controls. Extensive reviews of relevant work can be found in [11] and [12]; the former focuses on work in the years from 2014 to 2018, the latter on work from 2018 to 2021.

In general, the experiments conducted in these studies consist of the following steps: EEG data acquisition; preprocessing (different filtering and artefact removal methods); feature extraction; feature selection and/or dimensionality reduction; and classification [12]. According to [13], the feature extraction process is the most important point in this pipeline: The choice of classifier plays a smaller role for the quality of the results than an appropriate choice of feature extraction method.

So far, there is no consensus about which biomarkers in EEG data indicate depression – in fact, sometimes the results of different studies directly contradict each other. These apparent incongruities might at least partially be caused by the fact that studies can differ strongly from each other in their experiment setup. The differences are not limited to the experiment design, such as for example the employed classification method or the use of resting-state vs. task-based EEG, but can even concern technical foundations such as the positions of recording electrodes or the definition of frequency band limits. Additionally, the number of test subjects is often so small that results might not be statistically significant. All of these points make it difficult to arrive at an objective comparison of the results of different studies [11, 12].

It has also been pointed out that characteristics such as subjects’ sex and age or the type of depression treatment can act as confounding influences during the classification process [11]. For example, one study found biomarkers that could be used as predictors to treatment response, but only in women and only for certain drugs [14].

However, in spite of all of this ambiguity in the current research, it is possible to observe a number of trends across different studies.

EEG data is often analyzed in individual frequency bands. While just about every possible frequency band has been mentioned as important for recognizing depressed patients at some point (cf. [11]), a large portion of the results point to the alpha band (e.g. [15, 16]) and theta band (e.g. [17, 18]) as containing discriminatory biomarkers.

Biomarkers that distinguish depressive subjects are also often linked to specific regions of the brain, although which brain region is considered relevant differs between studies. For example, the asymmetry between signals in the left and right hemisphere is often found to be more pronounced in depressive subjects, with the activity in the left hemisphere being lower than in the right hemisphere [16, 19]. Some studies report that the biomarkers for depression are located only in the left brain hemisphere [20], or only in the right hemisphere [21]. Other studies have achieved good classification results by only using EEG signals recorded at the frontal region of the brain [16, 18].

## 2. Related Work

Among studies that investigated network structures in the brain, there is a consensus that the EEG dynamics of depressed patients tend to be more complex and exhibit more randomness than those of healthy subjects [11, 12], even though it remains controversial if and which network characteristics can be considered as depression biomarkers.

While the vast majority of papers focus on a binary classification problem (depressed vs. healthy subjects), there have been some approaches that try to further subdivide patients based on the severity of their depression. For example, [22] train a temporal convolutional network to predict the Beck Depression Inventory (BDI) score from EEG recordings (the BDI score indicates the severity of patients' depression based on questions about their subjective wellbeing and mental state [23]). [24] found patterns in EEG sleep recordings that correlate with depression severity.

Classification methods that distinguish between healthy and depressed subjects based on EEG data are often reported to be very successful, in many cases achieving classification accuracies well above 90%. However, as already mentioned, the sample sizes used in these studies are often quite small, and therefore it is unclear if the results are statistically significant. It should also be pointed out that the vast majority of papers only report the classification accuracy, even though this metric can be misleading (in the case of imbalanced data sets, the classification accuracy is likely to overestimate the classification quality). The F1 score, which is the harmonic mean of precision and recall, is a more robust metric, but is hardly ever reported.

There are also studies that focus on predicting the outcome of depression treatment. This problem comes in two different variations: Some studies analyze EEG recordings taken before the beginning of treatment in order to determine which treatment approach is most likely to lead to a response, while others analyze EEG recordings that were taken after treatment has begun and try to predict if there actually is a response. Most of these studies focus on either a single treatment or a small number of treatment approaches (maybe two or three different kinds of medication). For example, [25] predicts the response to repetitive transcranial magnetic stimulation, while [26] considers subjects treated with two different medications for treatment prediction. Even in studies that include multiple treatment approaches, the detected predictors for response are often specific to a certain treatment method (e.g. [26, 14]). [12] summarizes the results of such studies in recent years.

Generally speaking, there are fewer studies that focus on treatment prediction than on the classification of healthy vs depressed subjects, and the prediction accuracies reported for these experiments tend to be lower, indicating that treatment prediction is a more challenging problem. In this context, it is also interesting to consider the results from [27], where a classification experiment was performed including subjects who suffered from depression at the time of the EEG recording, subjects who had been diagnosed with depression at some point previously in their lives but were not currently considered depressive, and healthy controls who had never had depression in their lives. The authors found that the same features that allowed them to classify acute depression could also be used to classify the long-lasting effects of depression in subjects who had suffered from depression in the past. This in turn could be an indication that it is much more difficult to distinguish currently and previously depressed subjects based on their EEG recordings than it is to distinguish depressive subjects from healthy controls.

Granger causality has been repeatedly used for the analysis of brain scan data. A number of studies use features based on variations of Granger causality to classify depressive patients vs healthy controls: This has been done on EEG recordings [24], but also on fMRI data [28, 29] and MEG data [30]. [31] extend this use case and analyze Granger-causal networks extracted from fMRI data to not only distinguish between depressed subjects and healthy controls, but also to differentiate between early onset depression and late onset depression.

The application of Granger causality analysis to brain scan data has also repeatedly been used in emotion recognition experiments [32, 33, 34]. The results of these studies are interesting for the work presented in this thesis, since depression is known to change patients' emotional state,

including their emotional response to outside stimuli [1]. In one study, it was even found that the Granger-causal graphs created from EEG data during an emotion recognition task differ between depressed subjects and healthy controls [35].

Granger-causal analysis of brain scan data has also been used for classification experiments with a variety of diseases and conditions besides depression, such as HIV associated neurocognitive disorder [36] or dyslexia [37]. Other applications of Granger causality analysis to brain recordings include for example motor imagery tasks [38, 39], anesthesia recognition [40], or the classification of "congruent" vs "conflicting" situations that subjects were presented with during experiments [41].



## 3. Background

The main component of the experiments presented in this master thesis is the *Graphical Granger Causality by Information-Theoretic Criteria* (ITGGM) method [7]. This chapter will explain the necessary mathematical background to understand the concepts behind Granger causality in general and the ITGGM method in particular.

### 3.1. Granger Causality

The concept of Granger Causality was introduced in 1969 by C.W.J. Granger [6]. Given two time series  $x$  and  $y$ , Granger postulates that  $y$  can be said to cause  $x$  if the knowledge of past values of  $y$  improves the prediction of future values of  $x$ .

While in his original paper, Granger simply referred to this concept as "causality", there has since been some debate over the usage of this term (cf. [42]). Criticism argues for example that Granger's causality concept only reflects correlation or that the tests only find "predictive causality" [43]. But while it is debatable whether Granger's method uncovers true causality, it is still a valuable empirical method that can discover useful information about relationships between time series. A common solution in literature is to refer to this concept as "Granger causality", in order to distinguish it from other definitions of causality. This thesis follows this convention.

To define Granger causality in mathematical terms (cf. [6, 42]):

Define the following time series of length  $n$ :  $x = \{x_1, \dots, x_n\}$  and  $y = \{y_1, \dots, y_n\}$ .

Let  $U$  be all available information in the universe, and let  $U - y$  be this information with  $y$  excluded.  $P(x|U)$  is the best predictor for  $x$  that can be achieved by using the information in  $U$ . Similarly,  $P(x|U - y)$  is the best predictor for  $x$  than can be achieved using all information in  $U$  except for the information contained in  $y$ .

Granger causality is then defined by comparing the performance of the two predictors. If the prediction created by  $P(x|U)$  is better than the prediction created by  $P(x|U - y)$ , then this means that knowledge about  $y$  improves the prediction of  $x$ , and therefore  $y$  is considered Granger-causal for  $x$ .

It is assumed that the quality of a predictor for  $x$  can be judged from the variance in the prediction error: the smaller this variance, the better the predictor. In other words,  $y$  is said to Granger-cause  $x$  if  $\text{var}(x - P(x|U)) < \text{var}(x - P(x|U - y))$ .

So far, this definition is a purely theoretical construct, because it is obviously impossible to use all available information in the universe. In practice, one would instead use a vector autoregressive (VAR) model. In a VAR model, the present value of a time series is modeled as a linear combination of its own past values and the past values of other relevant time series. Accordingly, following the general idea described above, the present value of  $x$  can either be modeled as a linear combination of only its own past values, or as a linear combination of past values of itself and  $y$ . The *lag* parameter  $d$  ( $d \geq 1$ ) determines how many of the past values of  $x$  (and potentially  $y$ ) are considered; this is also known as the order of the VAR model.

This leads to the following two possibilities of expressing the value of  $x$  at time  $t$ :

$$x_t = a_1x_{t-1} + a_2x_{t-2} + \dots + a_dx_{t-d} + \epsilon_t \quad (3.1)$$

### 3. Background

$$x_t = a_1x_{t-1} + a_2x_{t-2} + \cdots + a_dx_{t-d} + b_1y_{t-1} + b_2y_{t-2} + \cdots + b_dy_{t-d} + \epsilon_t \quad (3.2)$$

The scalars  $a_i$  and  $b_i$  ( $1 \leq i \leq d$ ) are the regression coefficients.  $\epsilon_t$  is an error term which follows a normal distribution with zero mean and unit variance.

The model in Equation 3.1 uses only past values of  $x$ , while the model in Equation 3.2 uses past values of both  $x$  and  $y$  (note that this requires at least one of the regression coefficients  $b_i$  in Equation 3.2 to be nonzero). If the prediction of  $x$  achieved by the model in 3.2 is better than the prediction achieved by the model in Equation 3.1, then this means that including past values of  $y$  in the computation improves the prediction of future values of  $x$ . Consequently, if this is the case, then according to the definition given in the beginning of this Section,  $y$  is Granger-causal for  $x$ .

## 3.2. Multivariate Granger Causality

The example given in Chapter 3.1 describes the bivariate case, i.e. only two time series are involved. However, in reality, systems often consist of multiple processes that are represented as time series. Even if only two time series are of interest, it is still necessary to account for outside influences in order to be able to detect the true underlying structure. For example, if both time series  $x$  and  $y$  are actually caused by a third process  $z$ , but only  $x$  and  $y$  are included in the computations, then the result will likely show some spurious causal relationships either from  $x$  to  $y$  or from  $y$  to  $x$  because not all relevant information was available for the computations.

Approaches to discover Granger-causal relationships between systems of more than two time series are variously referred to in the literature as "multivariate Granger causality", "graphical Granger causality", or "network Granger causality". There are various methods to compute such multivariate Granger causality networks; a comprehensive overview can be found in [42].

One common approach is to use an extension of the VAR model described in Chapter 3.1. For this purpose, consider multiple time series  $x_1, \dots, x_p$ . Each of them consists of  $n$  features, similar to the definitions of  $x$  and  $y$  in Chapter 3.1:  $x_1 = \{x_{1,1}, \dots, x_{1,n}\}$ ,  $x_2 = \{x_{2,1}, \dots, x_{2,n}\}$ , and so forth. Then, consider  $X$  as the  $p \times n$ -dimensional matrix of all of these time series:  $X = (x_1, \dots, x_p)^T$ . Furthermore,  $X_t$  denotes the vector of all variables at time  $t$ :  $X_t = (x_{1,t}, \dots, x_{p,t})^T$ .

The VAR model for  $p$  time series with lag  $d \geq 1$  is then defined as follows:

$$X_t = A^1X_{t-1} + \cdots + A^dX_{t-d} + e_t \quad (3.3)$$

$A^1, \dots, A^d$  are  $p \times p$ -dimensional matrices of regression coefficients, and  $e_t$  is a  $p$ -dimensional vector of normally distributed error terms. The VAR model is fitted to obtain the regression coefficient matrices  $A^1, \dots, A^d$  that minimize the error terms in  $e_t$ . Nonzero entries in these matrices indicate Granger-causal relationships:  $x_j$  is Granger-causal for  $x_i$  if and only if  $A_{ij}^k$  is nonzero for some  $k$  ( $1 \leq k \leq d$ ).

To compute the coefficient matrices, a least squares approach can be used. However, the solutions generally tend to be very dense. In order to favour simpler, sparser solutions, a common remedy is to use penalties that punish overly complex models. The result in this case is a tradeoff between minimizing the prediction error and minimizing the model complexity.

## 3.3. Granger Causality with Information-Theoretic Criterion

The ITGGM method presented by Hlaváčková-Schindler and Plant [7] leverages information-theoretic criteria to find Granger-causal relationships between multiple time series. Instead

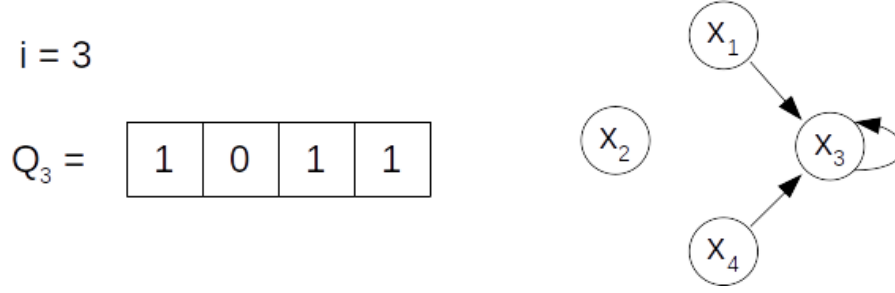


Figure 3.1.: Example for how the ITGGM method expresses a hypothesis about Granger-causal relationships. In this example,  $i = 3$ , which means that the target time series is  $x_3$ . The vector  $Q_3$  describes the presumed Granger-causal influences on  $x_3$ . In this case,  $x_1$ ,  $x_3$ , and  $x_4$  are Granger-causal for  $x_3$ , while  $x_2$  is not Granger-causal for  $x_3$ . The graph to the right visualizes these relationships.

of using a penalized VAR model, this approach re-interprets the problem as a combinatorial optimization problem.

Once again, consider  $p$  time series  $x_1, \dots, x_p$ . Pick one of these time series, denoted by  $x_i$ , as the "target" series. The goal of the optimization problem is to discover which of the  $p$  time series have a Granger-causal influence on  $x_i$ .

A hypothesis about the Granger-causal relationships influencing  $x_i$  can be expressed as a binary vector  $Q_i$  of length  $p$  in the following way: Each index  $j$  ( $1 \leq j \leq p$ ) corresponds to one time series. If the entry in  $Q_i$  at index  $j$  is 1, this means that  $x_j$  is assumed to Granger-cause  $x_i$ , and if the entry at index  $j$  is 0, this means that  $x_j$  is assumed to not Granger-cause  $x_i$ . An example for such a hypothesis is shown in Figure 3.1.

The combinatorial optimization problem is then to find the vector  $Q_i$  that best describes the true Granger-causal influences on  $x_i$ .

In order to assess which  $Q_i$  is best suited for a given  $x_i$ , information-theoretic criteria are used. Generally speaking, such a criterion is a measure for how well a model fits a given data set, measuring the tradeoff between goodness of fit and model complexity. The ITGGM method uses information-theoretic criteria to compute a score for a given pair  $(x_i, Q_i)$ : the lower the score, the better this hypothesis fits the data set. For a fixed  $x_i$ , the  $Q_i$  that produces the lowest score is assumed to describe the true Granger-causal influences on  $x_i$ .

Multiple different criteria are proposed for use with this method, such as e.g. the stochastic complexity (SC), the minimum message length (MML), the Akaike information criterion (AIC), and the Bayes information criterion (BIC). The exact steps needed to derive the formula for the score computation with each criterion are described in detail in [7]. The following explanation only gives a rough sketch of the computation process, which is sufficient to understand the experiments presented in this thesis.

As a precondition for the computations, it is assumed that all time series follow a Gaussian distribution.

In a first step, the time series that are considered as Granger-causal for the target series according to the given  $Q_i$  are assembled into a design matrix (details about how exactly this is done can be found in [7]), and the maximum-likelihood-estimates for the mean  $\hat{R}_i$  and variance  $\hat{\sigma}_i^2$  of the resulting model are computed. This step also includes the computation of a vector  $\hat{\beta}_i$ , which contains the regression coefficients for the Granger-causal relationships described by  $Q_i$ .

Additional input parameters for the criterion computation are the length of the time series  $n$ , the lag  $d$ , and the number of non-zero entries in the hypothesis vector  $Q_i$ , denoted by  $k_i$ . As in previous explanations, the lag  $d$  indicates the maximum number of past timesteps that are considered for finding the Granger-causal relationships.

All of these parameters are then used to compute the criterion score. For this thesis, the

### 3. Background

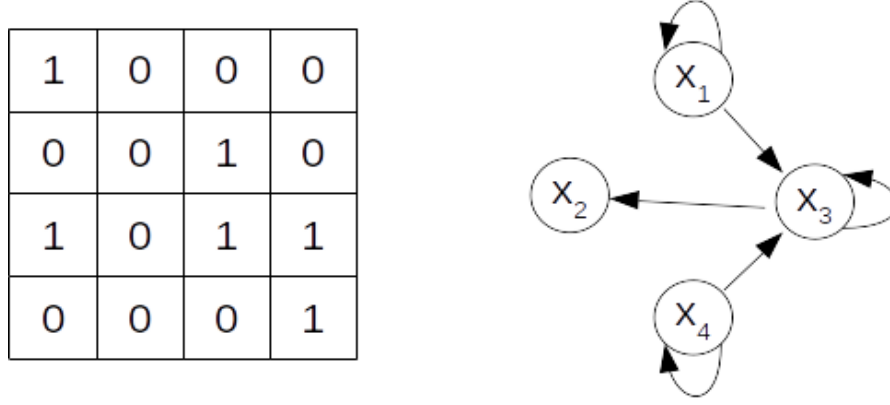


Figure 3.2.: Example for how the Granger-causal relationships between time series are represented in the ITGGM method. The matrix on the left describes the relationships in the format that is created as output of the ITGGM method. On the right, these relationships are visualized in graph form.

stochastic complexity was used as criterion in all experiments, due to its consistently high performance on a number of synthetic data sets in previous experiments (cf. [7]). In the case of the stochastic complexity, the criterion score is computed according to the following formula:

$$SC(x_i, Q_i) = \frac{n - d - k_i}{2} \log \hat{\sigma}_i^2 + \frac{k_i}{2} \log \hat{R}_i - \log \Gamma\left(\frac{n - d - k_i}{2}\right) - \log \Gamma\left(\frac{k_i}{2}\right) \quad (3.4)$$

where  $\Gamma(\cdot)$  denotes the gamma function.

The most straightforward optimization strategy to find the best  $Q_i$  for a given  $x_i$  is to evaluate each possible version of  $Q_i$ . However, this quickly becomes unfeasible for even moderately large  $p$ : For each  $x_i$ , there are  $2^p$  different vectors that have to be evaluated. Instead of such an exhaustive search, a genetic algorithm is proposed in [7] as optimization method. The implementation created for this thesis provides both of these possibilities (cf. Chapter 5, which also describes the functionality of the genetic algorithm in detail).

The optimization procedure is repeated for each of the  $p$  time series, leading to  $p$  result vectors. Stacking those  $p$  vectors creates a matrix that describes the network of Granger-causal relationships among all  $p$  time series. This matrix can also be considered as the adjacency matrix of a graph representation of the Granger-causal relationships; an example for this is shown in Figure 3.2. Note however that generally, the adjacency matrix  $A$  of a directed graph is defined such that an entry of 1 in the matrix element  $A_{ij}$  indicates a directed edge from node  $i$  to node  $j$ . In the case of the matrix produced by the ITGGM method, this convention is reversed: An entry of 1 in element  $A_{ij}$  indicates a Granger-causal influence from time series  $j$  to time series  $i$ . This is a direct consequence of the way the method is defined and does not make a difference for the information content of the result: The user simply has to be aware that if they want to use the generated matrix as input for any graph property computations or graph visualization libraries, they most likely have to transpose the matrix first.

In the rest of this thesis, the ITGGM output matrix will simply be referred to as "adjacency matrix". All experiments described in this thesis use the output matrix as it is, since for these experiments it is irrelevant whether the matrix has been transposed or not.

### 3.4. Limitations and Challenges

Like virtually any model that attempts to depict reality, Granger causality makes a number of simplifying assumptions. Possibly the most important limitation of Granger causality as

described in this Chapter is that it only models linear and additive relationships between time series, while in reality it is well possible that interactions follow nonlinear patterns. Another important assumption is that the time series are stationary, even though in many common application scenarios it is reasonable to assume that the true patterns in the data will change over time. Any application that uses Granger causality for the analysis of time series has to be aware of these assumptions. (Note that variations of Granger causality have been proposed that aim to address these shortcomings; see e.g. [42] for a review.)

Another challenge is the selection of the proper lag  $d$ , as an unsuitable choice may distort the results or even make them completely unusable. The lag selection strategy used in this thesis is described in Chapter 6.

Finally, the ITGGM method assumes that the time series follow a Gaussian distribution, which might not be the case with real-world data. However, there are preprocessing methods that can be used to improve the "Gaussianity" of data sets, such as e.g. Box-Cox transformations [44].



## 4. Method

This Chapter describes the method behind the experiments in this thesis. This encompasses the acquisition of EEG recordings, the computation of Granger-causal networks from the recordings, and the training of classifiers to distinguish between responders and nonresponders based on the Granger-causal networks.

### 4.1. EEG Recordings

The EEG data was recorded by the Synchronization project members at the Czech Academy of Sciences, who also performed the preprocessing and labelling steps described in this Section.

**Subjects.** Data was recorded from subjects who suffered from depression and were just starting a new treatment method with either an antidepressant drug or a brain stimulation method. A list of all used treatments can be found in Appendix A.1.

The recordings were taken from a total of 228 subjects. However, some of the recordings were unusable, for example due to technical issues during the recording or an excessive number of artefacts. The final data set therefore contains data from 176 subjects. Details about age range, biological sex, and number of responders in the data set can be found in Appendix A.2.

**Recording process.** Each subject was recorded multiple times during separate visits to the recording lab. The recordings taken during the first two visits were used for the data analysis presented in this thesis. The first visit took place on day 0 (i.e. just before starting the new treatment), and the second visit took place on day 7 (i.e. one week after the beginning of treatment). Data was recorded at 19 electrodes placed on the subject's scalp in standardized positions, according to the international 10-20 system [45]. The names and locations of these electrode positions are shown in Figure 4.1. Recordings were taken in resting state, with closed eyes, for a duration of 10 minutes. Subjects were roused if signs of drowsiness appeared during the recording.

**Preprocessing steps.** The following processing steps were applied to the EEG recordings:

- The original sampling rate varied between 250 Hz and 1000 Hz; to unify this across all subjects, the 1000 Hz recordings were downsampled by only keeping every 4th sample.
- The first and last 30 seconds were removed from each recording, since they are likely to contain more artifacts.
- The EEG recordings were transformed to use the Average Reference as referencing method.<sup>1</sup>
- A bandpass filter was applied to only keep frequencies in the range from 1 to 40 Hz.

---

<sup>1</sup>An EEG signal measures the difference in electrical potential between the recording electrode and a reference electrode. The choice of reference electrode can have an important influence on the quality of the EEG signal, e.g. in terms of the amount of noise in the signal. It is possible to change the reference electrode after the recording ("offline") by applying mathematical transformations to the signal. When using the Average Reference method, the signal is recomputed to use the average of the signals recorded at all electrodes as reference. [46]

#### 4. Method

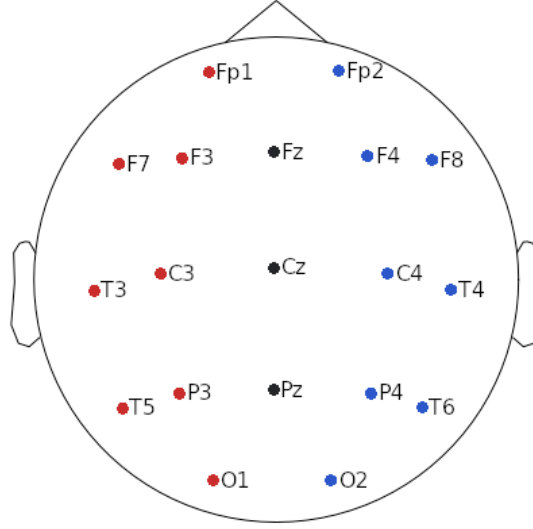


Figure 4.1.: The 19 standardized electrode locations on the scalp that were used for the EEG recordings. The used positions are Fp1, Fp2, F3, F4, C3, C4, P3, P4, O1, O2, F7, F8, T3, T4, T5, T6, Fz, Cz, and Pz.

- Segments containing high-power artifacts or missing elements were removed from the data. For this purpose, the recording was subdivided into "epochs" with a length of 2 seconds (i.e.  $n = 500$  samples) each, with no overlap between them, and epochs containing problematic data were removed entirely. This guarantees that every 2-second epoch in the remaining data is continuous, which is important for analysis methods that consider time-lagged data.

Preprocessing was done using Matlab's EEGLAB software [47].

**Class labels.** The effectiveness of each subject's treatment was evaluated via a psychological assessment, which was carried out both before the beginning of treatment and after 28 days. For this purpose, subjects were rated on the Montgomery-Åsberg Depression Rating Scale (MADRS) [48], which assigns a score between 0 and 60 to each subject to indicate the severity of their depression. The MADRS diagnostic questionnaire is intentionally designed to be highly sensitive to changes due to treatment success.

Based on this information, labels were assigned to the subjects, marking them as either responder (i.e. treatment was effective) or nonresponder (i.e. treatment did not lead to improvements). These labels can be used as class labels for supervised learning tasks. Among the 176 subjects in the data set there are 84 responders and 92 nonresponders.

Treatment success can be defined in two different ways: *response*, which means that the MADRS score after 28 days is less than 50% of the initial MADRS score before treatment; and *remission*, which means that the MADRS score after 28 days lies below a fixed threshold (usually in the range of 8 – 10). The labelling system used in this thesis indicates response, not remission.

##### 4.1.1. Ethics Statement

This study was reviewed and approved by the Ethical committee of the Prague Psychiatric Centre and the Ethical committee of the National Institute of Mental Health, Klecany, Czech Republic. All subjects gave their written informed consent to participate in this study.

## 4.2. Granger Causality Computations

The ITGGM method was used to compute networks of Granger-causal relationships between the time series in each EEG recording. The computations were repeated multiple times for each recording, exploring a number of different input settings, which are described in this Section.

**Input time series.** Each EEG recording consists of 19 time series. This data can be used as input for the Granger causality computations as it is, but it also makes sense to apply some transformations beforehand, since this might increase the discriminative power of the Granger causality features.

- **Brain hemisphere:** The standard setting is to compute the network of Granger-causal relationships between all 19 electrodes. But as pointed out in Chapter 2, depression biomarkers have often been linked to specific brain regions; one area that is frequently pointed out as important is the left brain hemisphere (e.g. [19, 20]). The Granger-causal relationships were therefore also computed only between the 8 electrodes on the left side of the scalp (electrodes marked in red in Figure 4.1). As a third setting, relationships were computed for the left-hemisphere electrodes plus the three center electrodes (marked in black in Figure 4.1), using a total of 11 input channels.
- **Frequency band:** Analysis of EEG data is often done on the level of individual frequency bands. In particular, research on depression often focuses on the alpha band (8-12 Hz) and theta band (4-8 Hz) ( cf. [11, 12]). Accordingly, Granger causality computations were carried out using the original unfiltered recordings, data filtered to the alpha band, and data filtered to the theta band. Figure 4.2 shows part of an EEG recording in both the filtered and unfiltered state, in order to demonstrate the differences between the frequency bands.

In total, the Granger causality computations were repeated nine times per EEG recording, since each brain hemisphere setting was combined with each frequency band setting. The EEG data transformations (channel extraction and frequency filtering) were done using the MNE Python library [49].

**Box-Cox transformation.** As pointed out in Section 3.3, the ITGGM method requires Gaussian input data. Preliminary analyses showed that a Box-Cox transformation [44] improves the "Gaussianity" of the EEG data. Therefore, this transformation was applied to all time series before computing the Granger-causal networks. The SciPy Python library [50] was used for this purpose.

**Epoch-wise computations.** Due to the removal of faulty data during the preprocessing step, the continuity of the EEG recordings is only guaranteed in epochs of a length of 2 seconds. This is important because the ITGGM method considers input from previous time steps (as specified by the lag parameter), which would be distorted by discontinuities in the data. As a consequence, the Granger-causal relationships can only be computed for one epoch at a time.

Additionally, even though the EEG data was recorded for a duration of 10 minutes, it does not make sense to use all of this information anyways: It is reasonable to assume that the data is nonstationary, i.e. the true causal connections very probably change over time.

A good solution to these two combined challenges is to compute the Granger-causal relationships for a small number of consecutive epochs, such that for this small part of the recording the signal can be considered as locally stationary. This step results in one adjacency matrix for each of the considered epochs. The element-wise average of the resulting adjacency matrices is then used as the final result for this recording. In this thesis, the first 8 epochs (i.e. a total recording duration of 16 seconds) in each recording were used for this purpose. Note that due to the averaging step,

#### 4. Method

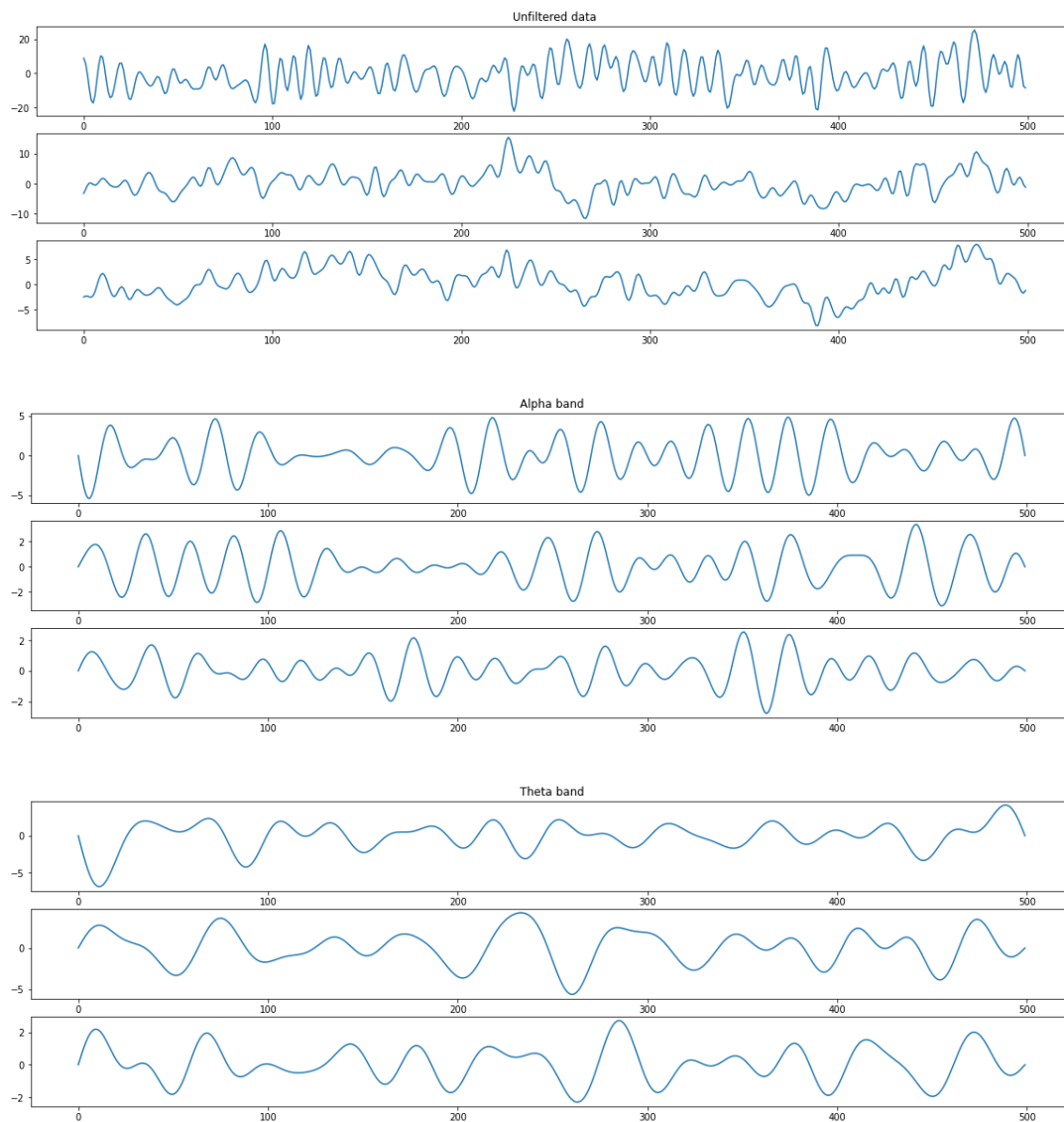


Figure 4.2.: Example EEG data showing the first two seconds of a subject's recording. To improve legibility, only data from three electrodes is shown. The top plot shows the unfiltered data. The other two plots show the same data after it has been filtered to include only frequencies in the alpha band (center) or in the theta band (bottom).

the final output matrix no longer has to be binary; instead each of its elements is a multiple of  $\frac{1}{8}$  in the range from 0 to 1.

The Granger-causal relationships were computed using the Python implementation of the ITGGM method which was created as part of this thesis (the implementation is described in Chapter 5).

**Connection strength matrices.** As pointed out in Chapter 3.3, one intermediate result in the ITGGM method is a vector  $\hat{\beta}_i$  of regression coefficients. Using this vector, it is possible to map a real value to each 1 in the corresponding vector  $Q_i$ , which can be interpreted as the strength of this Granger-causal connection.<sup>2</sup>

In order to obtain some more nuanced information about the Granger-causal relationships than is contained in the binary adjacency matrices, the  $\hat{\beta}_i$  vectors were used to create "connection strength matrices" or " $\beta$ -matrices". For each 1 in an adjacency matrix, its corresponding  $\beta$ -matrix contains a real value in the same position that indicates the strength of this Granger-causal connection.

Similar to the adjacency matrices, one  $\beta$ -matrix was computed for each of the first 8 epochs in a recording. The final  $\beta$ -matrix for the recording is the element-wise average of those 8 intermediate  $\beta$ -matrices.

**Output summary.** The Granger causality computations were repeated nine times, each time for a different combination of brain hemisphere and frequency band. This leads to nine sets of results.

Within each set, there are two output matrices per EEG recording: one adjacency matrix and one  $\beta$ -matrix. (Both of them are actually the average of the results of the first 8 epochs. From now on, if not explicitly specified otherwise, the terms "adjacency matrix" and " $\beta$ -matrix" refer to the averaged results.)

As explained in the previous Section, two separate EEG recordings were used for each subject: one that was taken directly before the beginning of treatment, and one taken on the 7th day of treatment. This means that within each set of results, there are four matrices per subject: the adjacency matrices for the first and second visit, and the  $\beta$ -matrices for the first and second visit.

The Granger causality computation process is summarized in Algorithm 1.

### 4.3. Classification based on Granger Causality Results

The final part of the experiments is to train classifiers to distinguish between responders and nonresponders based on the Granger causality features that were extracted in the previous step.

The 176 subjects were divided into a training set of 134 subjects and a test set of 42 subjects. For each of the nine result sets obtained from the previous step, multiple training scenarios were tried out, using only subjects from the training set as input. The test set was kept aside and was only used at the very end to evaluate the best classification scenario over all result sets and all training scenarios on previously completely unseen data.

**Training data creation.** To create the training data for the classifiers, every subject needs to be represented by a one-dimensional feature vector. It is therefore necessary to apply some transformations to obtain such feature vectors from a set of Granger causality results.

For the following explanations, consider a single subject, who is represented by four matrices in the given result set. Let the subject's adjacency matrices for the first and second visit be

---

<sup>2</sup>To be more precise: With a lag of  $d$ , the  $\hat{\beta}_i$  vector contains  $d$  regression coefficients for each 1 in  $Q_i$ . They are reduced to a single value by taking the maximum absolute value of the  $d$  possibilities, since this can be interpreted as the strongest Granger-causal influence over the  $d$  time steps.

#### 4. Method

**Input :** Data set containing two EEG recordings per subject

```

for hemisphere in [left, left-center, both] do
  for frequency band in [unfiltered, alpha, theta] do
    for each subject in the training set do
      for visit in [visit1, visit2] do
        Load the subject's EEG data for this visit;
        Extract the electrode channels that belong to the brain hemisphere;
        Extract the first eight 2-second epochs;
        for each epoch do
          Extract the frequency band;
          Apply a Box-Cox transformation;
          Apply the ITGGM method to compute the binary adjacency matrix and
            the matrix of  $\beta$  values;
        end
        Compute the average adjacency matrix over all eight epochs;
        Compute the average  $\beta$ -matrix over all eight epochs;
      end
    end
  end
end

```

**Algorithm 1:** Summary of the Granger causality computation process

called  $A_1$  and  $A_2$ , respectively. Similarly, let the  $\beta$ -matrices for the first and second visit be called  $B_1$  and  $B_2$ .

Two different hypotheses were explored:

- Hypothesis 1: For subjects who are responding to treatment, the difference between the EEGs of the first and second visit is greater than for those who are not responding. The relevant data for classification is therefore contained in the matrix  $A_{diff} = A_2 - A_1$  or in the matrix  $B_{diff} = B_2 - B_1$ .
- Hypothesis 2: After a week of treatment, there is a difference between the EEGs of responders and nonresponders. The relevant data for classification can therefore be found in  $A_2$  or  $B_2$ . Since the class labels describe a relative improvement (response), as opposed to an absolute final state (remission), this hypothesis is less likely than the first one, but for completeness it was still included in the experiments.

This leads to four different possibilities to obtain a one-dimensional feature vector for a subject: by using either  $A_{diff}$ ,  $B_{diff}$ ,  $A_2$ , or  $B_2$ . The feature vector is created by flattening the chosen matrix, i.e. taking all of its rows and concatenating them to create a vector. For a matrix with  $p$  rows and  $p$  columns, the resulting vector therefore contains  $p * p$  entries.

The training data is then assembled by stacking the feature vectors of all subjects in the training set. Accordingly, the final training data is represented as a matrix with  $n\_subjects$  rows and  $p * p$  columns, where  $n\_subjects$  is the number of subjects in the training set.

**Dimensionality reduction.** Depending on the number of electrodes used for the Granger causality computations, the feature vectors can become very high-dimensional: In the case of the full-brain experiments with 19 electrodes, each feature vector contains  $19 * 19 = 361$  entries. An optional preprocessing step is therefore to apply a Principal Component Analysis (PCA) to the assembled training data, aiming to reduce the dimensionality of the data while hopefully preserving all important information. PCA was applied to variously retain 80%, 90%, and 95%

of the variance in the data, in addition to running classification experiments without PCA.

**Classification.** The following classifiers were used:

- Support vector classifier with linear, radial basis function (RBF), and polynomial kernel
- K nearest neighbor classifier
- Decision tree classifier
- Random forest classifier
- Naive bayes classifier

The classifiers were taken from the scikit-learn library [51].

Each of these classifiers has a number of hyperparameters that can be fine-tuned in order to improve the classification result. For example, changing the number of neighbors considered by the K nearest neighbor classifier or the number of estimators in the random forest might have a considerable impact on the result. Therefore, for each classifier, a grid search over different hyperparameter value combinations was performed in order to find the combination that leads to the best results. The hyperparameter values that were explored for each classifier can be found in Appendix A.3.

For each combination of hyperparameter values, a 10-fold cross validation was performed. For this process, the training data is subdivided into 10 "folds" of approximately equal size. 9 of the folds are used to train the classifier, and the remaining fold is used as validation set. The classification performance is evaluated by computing the F1 score between the classifier's predicted labels for the validation fold and the corresponding true class labels. This process is repeated 10 times, each time using a different fold as validation set. The mean and standard deviation of the F1 score over these 10 results are reported.

The complete process of creating training data and training the classifiers is summarized in Algorithm 2.

#### 4. Method

**Input :** One set of Granger causality results (defined by one choice of brain hemisphere and one choice of frequency band), where each subject is represented by four matrices  $A_1$ ,  $A_2$ ,  $B_1$  and  $B_2$   
List of classifiers with their appropriate hyperparameter spaces

```

for each subject in the training set do
    Load the matrices  $A_1$ ,  $A_2$ ,  $B_1$  and  $B_2$  for this subject from the Granger causality
    results;
    Compute  $A_{diff} := A_2 - A_1$ ;
    Compute  $B_{diff} := B_2 - B_1$ ;
    /* Create four feature vectors for this subject */
    f1 := flatten( $A_{diff}$ );
    f2 := flatten( $B_{diff}$ );
    f3 := flatten( $A_2$ );
    f4 := flatten( $B_2$ );
end

/* Create four training data versions */
td1 ← stack the f1 feature vectors of all subjects in the training set;
td2 ← stack the f2 feature vectors of all subjects in the training set;
td3 ← stack the f3 feature vectors of all subjects in the training set;
td4 ← stack the f4 feature vectors of all subjects in the training set;

/* Repeat the training process for each of the four training data versions */
for  $td$  in [td1, td2, td3, td4] do
    for variance in None, 80%, 90%, 95% do
        data ← apply PCA to td, retaining the given amount of variance ; /* None means
        PCA is not applied */
        for each classifier do
            for each combination of hyperparameter values for this classifier do
                Perform 10-fold cross validation on the data, report the mean and standard
                deviation of the F1 score;
            end
            Report the best result over all hyperparameter combinations;
        end
    end
end

```

Find the best result achieved by any of the classifiers for any input settings

**Algorithm 2:** Summary of the classifier training process

## 5. Implementation

This Chapter describes the implementation of the *Graphical Granger Causality by Information-Theoretic Criteria* (ITGGM) method [7] that was created as part of this thesis. It is written in Python 3 and is available at GitHub [8].

The implementation follows the API conventions established by the well-known *scikit-learn* library [51] and comes ready to be installed as a local pip module [52]. All that is required for the user to do is to download the code from the GitHub repository, navigate to the implementation's root directory (the directory that contains the `setup.py` file) in the terminal, and run the following command:

```
pip install .
```

After this, the newly installed module can be accessed from any Python script on the local system with `import itgraphicalgranger`.

### 5.1. API Convention

Scikit-learn is a very popular machine learning library, the de-facto standard for machine learning in Python. Its API convention [53] is therefore largely familiar to data scientists who work in Python. Furthermore, it was designed with the aim to make it as easy as possible for new users to start working with the library. By following this convention, this implementation will therefore be easy to use for both new and experienced users.

The scikit-learn API convention presents a number of general design guidelines. The most relevant ones for this implementation are as follows:

- Reasonable default values should be provided for all parameters that can be set by the user.
- Parameters that are learned by an algorithm, as well as user-defined hyperparameters, should be stored as public attributes to facilitate model inspection.
- Data sets should be represented as NumPy arrays or SciPy matrices, and parameters should be stored in standard Python data types such as strings, floats etc.
- External dependencies should be kept to a minimum. Ideally, only Python, NumPy, and SciPy should be required to install the library.
- Interfaces should not be defined explicitly, nor should it be necessary to inherit from (abstract) base classes in order to create new classes that fulfil certain roles. Instead, the *duck typing* principle should be used. This means that interfaces are specified implicitly, by declaring a number of attributes or methods that an object is expected to have. Any object that provides these attributes and methods automatically fulfils the duck typing interface.

Sklearn defines three different duck typing interfaces: *estimators* for building and fitting models, *predictors* for making predictions, and *transformers* for converting data. For this implementation, the estimator interface is most relevant. It is intended for classical learning algorithms such as classifiers and clustering algorithms, but also for preprocessing routines and feature extraction techniques. Since the ITGGM method can be considered a feature extraction method, it is a logical choice to implement it as an estimator.

## 5. Implementation

The duck typing interface for a scikit-learn estimator strictly separates instantiation from the learning or feature extraction process. All relevant hyperparameters are set during instantiation in the form of constructor arguments. No training data is passed to the constructor, and no actual learning is performed yet. The learning process is started by calling the estimator's `fit` method, which receives the training data as an argument. Learned parameters are stored as public attributes; they are marked by trailing underscores to distinguish them from the user-defined hyperparameters, which are also saved as public attributes to facilitate model inspection. The return value of the `fit` method is the estimator object that it was called on (this allows the chaining of method calls).

### 5.2. Class Structure

This implementation of the ITGGM method has two main components: an optimizer and an information-theoretic criterion. The criterion is responsible for taking a single binary vector  $Q_i$  indicating Granger-causal influences on one target time series (cf. Chapter 3.3) and computing a score that indicates how well this relationship pattern fits the data. The optimizer on the other hand is responsible for trying out different versions of  $Q_i$  for a target series, passing each of them to the criterion in order to obtain its score, and finding the  $Q_i$  that yields the overall best score.

There are a number of different optimization strategies and information-theoretic criteria that can be used with this method.<sup>1</sup> It should be possible to combine every available optimization strategy with every available criterion version. Accordingly, it was a logical design decision to represent each of these options as a separate class.

Every optimizer class follows the estimator interface that was described in the previous Section. The information-theoretic criteria, on the other hand, do not really fall into any of the categories that this interface is intended for. Instead, a custom duck typing interface was defined, which will be called the *criterion interface*. This interface will be described in Chapter 5.4.

The interaction between any chosen optimizer-criterion-pair is handled by a wrapper class called **GrangerCausality**, which also follows the estimator interface. This class provides the user interface for the implementation and handles the instantiation and organization of all necessary objects. The user only has to choose their preferred optimization strategy and information-theoretic criterion, which can be done by passing two string parameters to the constructor of the **GrangerCausality** class. All further details of how exactly optimizer and criterion are represented and how they interact with each other are then taken care of inside the wrapper class and are completely transparent to the user.

The relationships between the classes are shown in Figure 5.1.

For maximum flexibility, the implementation also allows the user to define their own information-theoretic criteria. Instead of the string parameter to choose the criterion, the user can pass any object that follows the criterion interface to the constructor of the **GrangerCausality** class. This object will then be used in place of the pre-defined criterion options.<sup>2</sup>

### 5.3. UI and the Wrapper Class

As specified by the API convention, the implementation provides default values for all user-defined parameters. In its most basic version, computing the Granger-causal relationships between the

---

<sup>1</sup>While the implementation in its current state only provides one information-theoretic criterion, the stochastic complexity, it was intentionally designed in such a way that it can easily be extended with additional criteria (or additional optimization strategies, for that matter). The explanations in this Chapter therefore consider the generic case where multiple options are available.

<sup>2</sup>It was not considered necessary to provide a way to pass in a user-defined optimizer object, since the use case of trying out new optimizers is likely to be much less common than that of trying out new criteria. However, it is easily possible to tie a new optimizer class in with the existing implementation through some minor modifications to the source code (cf. Chapter 5.3).

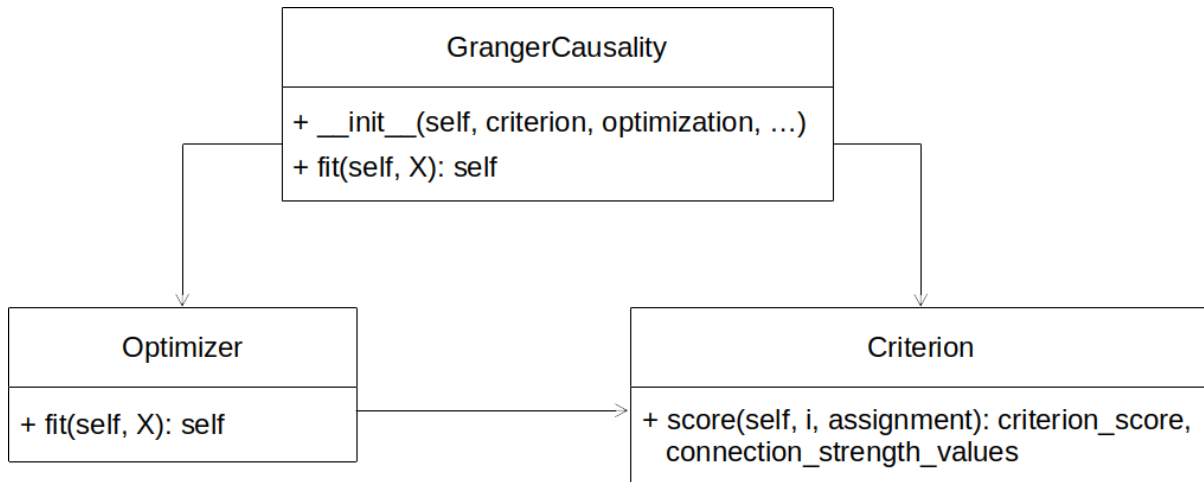


Figure 5.1.: Class diagram showing the relationships between the classes in the implementation. In this diagram, **Optimizer** and **Criterion** are generic placeholders for any concrete optimizer and criterion classes. Accordingly, only the methods defined by the duck typing interfaces are shown for these classes.

time series in a data set is therefore as simple as instantiating an object of the **GrangerCausality** class and passing the data set to its `fit` method. This basic process is demonstrated in Listing 5.1.

```

1 from itgraphicalgranger import GrangerCausality
2
3 # assumption: the data set is stored in the variable X
4
5 # instantiate the estimator object
6 gc = GrangerCausality()
7
8 # compute the Granger-causal relationships
9 gc.fit(X)
10
11 # access the resulting adjacency matrix
12 res = gc.adjacency_matrix_
13
14 # do something with the result
15 # ...

```

Listing 5.1: Minimal code required to compute Granger-causal relationships in a data set.

Arguably, in most cases relying entirely on the default parameters will not be enough – usually the user will want to tailor at least some parameters to their particular use case. This can be done by passing keyword arguments to the **GrangerCausality** constructor.

The **GrangerCausality** class has two main constructor arguments: two string arguments called **criterion** and **optimization**. They determine which information-theoretic criterion and optimizer classes will be instantiated by the wrapper class. (If a user wants to use their own custom criterion instead of a predefined one, they can also pass an object as the **criterion** constructor argument.)

All other keyword arguments in the constructor are hyperparameters that determine the behaviour of these two classes, and the available keywords depend on the choice of optimizer and criterion: For example, an exhaustive search optimizer has different hyperparameters than a genetic algorithm optimizer. The **GrangerCausality** class in fact does not do anything with these hyperparameter arguments other than pass them on to the criterion and optimizer classes upon instantiation. In order to keep this process simple, a handy Pythonic shortcut is used: The **GrangerCausality** constructor has an argument called **\*\*kwargs**, which assembles all keyword

## 5. Implementation

arguments that are not explicitly named in the constructor signature into a dictionary called `kwargs`. The arguments in this dictionary can then be passed on as a whole to the constructors of the other classes. As a consequence, there is no need for the `GrangerCausality` class to know anything about the hyperparameter keyword arguments that are accepted by the various optimizer and criterion classes. This not only keeps the code concise and readable, but also makes it very easy to later on extend the implementation with additional criteria and optimizers that require different hyperparameters.

The signature of the `GrangerCausality` class's constructor is shown in Listing 5.2. An example usage of the implementation with user-defined parameters is shown in Listing 5.3.

```
1 def __init__(self, criterion = "SC", optimization = "genetic", **kwargs)
```

Listing 5.2: Constructor signature of the `GrangerCausality` class.

```
1 # instantiate the estimator object and pass keyword arguments to the
  constructor
2 gc = GrangerCausality(criterion="SC", optimization="genetic", lag=5,
  n_generations=20)
3
4 # compute the Granger-causal relationships
5 gc.fit(X)
6
7 # access the resulting adjacency matrix
8 res = gc.adjacency_matrix_
```

Listing 5.3: Example instantiation of the `GrangerCausality` class with user-defined arguments.

In this case, the user wants to use the stochastic complexity criterion with a lag of 5, and a genetic algorithm optimization that runs for 20 generations, while leaving all other settings at their default values.

The keyword arguments that are accepted by the `GrangerCausality` constructor are summarized in Table 5.1. This table does not show the keyword arguments for the optimizer and criterion hyperparameters, which will be listed later on during the description of these classes.

| Argument name | Possible values                                                                                                                   |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------|
| criterion     | "SC": use the stochastic complexity criterion<br>any object that implements the criterion interface: use this object as criterion |
| optimization  | "exhaustive": use exhaustive search<br>"genetic": use a genetic algorithm                                                         |

Table 5.1.: Constructor keyword arguments for the `GrangerCausality` class.

In the constructor, the `GrangerCausality` class only saves the user input, it does not yet actually do anything with it. The instantiation of objects and the feature extraction process happen once the `fit` method is called.

In the `fit` method, the wrapper class first creates an information-theoretic criterion instance, passing it the data set as well as all user-defined keyword arguments as constructor arguments. The keyword arguments were previously collected in a standard Python dictionary, which was stored as an attribute of `self`; the unpacking operator `**` is used to automatically extract all key-value pairs from this dictionary. The criterion class will react to the keyword arguments that are relevant for its behaviour and ignore the rest (more details on how exactly this is achieved will follow in Chapter 5.4). If a custom criterion object was provided by the user, this instantiation is skipped and the provided object is used instead.

Next, an optimizer instance is created according to the user's specification. It receives the previously instantiated (or user-provided) criterion object as a constructor argument, as well as all provided keyword arguments; similar to the criterion, it will only regard the keyword arguments that are relevant for itself.

Finally, the wrapper class calls the optimizer's `fit` method to start the computation of the Granger-causal relationships. Once the feature extraction process is complete, it accesses the results provided by the optimizer and sets them as its own attributes, in order to expose them directly to the user.

Listing 5.4 shows the code for this method. This Listing also shows why it is extremely easy to extend the implementation. To tie new criterion or optimizer classes in with the existing implementation, it suffices to add additional `elif`-clauses in the appropriate places in this method (lines 9 and 17).

```

1 def fit(self, X):
2     # if the user did not provide a criterion instance, create one
3     if self.criterion is None:
4         if self.criterionname == "SC":
5             criterion = crit.StochasticComplexity(X, **self.user_kwargs)
6             # additional criteria can be added here
7         else:
8             raise ValueError("Invalid criterion name: " + self.criterionname)
9             # set the criterion as public attribute so it can be inspected by the
              user
10            self.criterion = criterion
11
12            # create the optimizer
13            if self.optimization == "genetic":
14                optimizer = optim.GeneticAlgorithm(self.criterion, **self.user_kwargs)
15            elif self.optimization == "exhaustive":
16                optimizer = optim.ExhaustiveSearch(self.criterion, **self.user_kwargs)
17            # additional optimizers can be added here
18            else:
19                raise ValueError("Unknown optimization method: " + self.optimization)
20            # set the optimizer as public attribute so it can be inspected by the user
21            self.optimizer = optimizer
22
23            # make sure the data is in the form of a numpy array
24            X = np.asarray(X)
25
26            # run the optimizer
27            optimizer.fit(X)
28
29            # get the results from the optimizer and set them as public attributes of
              self for easier user access
30            self.adjacency_matrix_ = optimizer.adjacency_matrix_
31            self.beta_matrix_ = optimizer.beta_matrix_
32            self.adjacency_matrix_scores_ = optimizer.adjacency_matrix_scores_
33
34            # scikit-like convention: fit() method should return self
35            return self

```

Listing 5.4: The `fit` method of the `GrangerCausality` class. This method is responsible for instantiating and coordinating the optimizer and criterion objects.

## 5.4. Information-theoretic Criteria

The purpose of an information-theoretic criterion class in this implementation is to determine how well a certain assignment of Granger-causal influences fits a given data set.

To recall: The data set consists of  $p$  time series, all of which can have Granger-causal influences on each other. One of these  $p$  time series is chosen as target. A binary vector of length  $p$  is created to represent one possible scenario of Granger-causal influences on this target: If there is a 1 at the  $j$ -th position in the vector, this means that the  $j$ -th time series is assumed to have a Granger-causal influence on the target series (as demonstrated earlier in the example in Figure

## 5. Implementation

3.1). The criterion computes a score that indicates how well this scenario fits the data set, with lower scores indicating a better fit.

### 5.4.1. The Criterion Duck Typing Interface

A custom duck typing interface was defined for the criterion classes. It demands the existence of one method, which is called `score`. This method accepts two parameters: `i` and `assignment`.

- `i` is the index of the target time series in the data set; it is an integer value that can range from 0 to  $p - 1$ .<sup>3</sup>
- `assignment` is the binary vector that indicates the Granger-causal influences on the target time series. It is equivalent to the vector  $Q_i$  that was described in Chapter 3.3.

The first return value of the `score` method is a float, which is the score that indicates how well `assignment` describes the true Granger-causal influences on the target time series, according to this information-theoretic criterion.

The duck typing interface also specifies a second return value, which is a vector of the same length as the `assignment` vector. It can be used to indicate the strength of each Granger-causal connection in `assignment`. In the case of the stochastic complexity, this vector would contain the  $\beta$ -values as described in Chapter 4. However, the duck typing interface also allows implementations to return a meaningless dummy vector with arbitrary content in this place, in case a criterion does not provide a possibility to compute indicators for the connection strength. It is the responsibility of the user to read the documentation for the criterion they are using in order to know whether the second return value contains meaningful information.

The signature of the `score` method is shown in Listing 5.5.

```
1 def score(self, i: int, assignment: ArrayLike) -> Tuple[float, ArrayLike]:
```

Listing 5.5: Method signature of the `score` method, with type annotations, as defined by the criterion duck typing interface.

Finally, the `score` method may raise a `ValueError` in order to refuse an assignment vector. This is a necessary safety mechanism because not all possible assignments might be valid for a given criterion. For example, with the stochastic complexity, if the assignment vector contains only zeros, then the score computations break down; therefore such a vector has to be considered as invalid. Optimizers need to be aware of this possibility when they call the `score` method and provide mechanisms to skip or replace invalid assignments.

### 5.4.2. Stochastic Complexity

The `StochasticComplexity` class implements the stochastic complexity criterion as described in [7]. It follows the criterion duck typing interface.

In its constructor, the class accepts the data set `X`, as well as a `lag` argument that indicates the lag for the Granger causality computations. There are also a number of additional keyword arguments that are not directly related to the mathematical formulation of the criterion, but that control the numerical and computational behaviour of the class. For example, the computation of the vector of regression coefficients  $\hat{\beta}_i$  requires solving a linear system with a least squares solver; some of the keyword arguments allow the user to control the behaviour of this solver, in case the default settings lead to numerical problems. However, for the vast majority of use cases these arguments can simply be left at their default values.

A summary of the constructor arguments is given in Table 5.2.

---

<sup>3</sup>In the explanations in Chapter 3.3 the index could range from 1 to  $p$ , but in the Python implementation, indices are zero-based.

| Name                     | Type       | Explanation                                                                                                                                                                                                                                                                                                                           |
|--------------------------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| X                        | array-like | Time series data                                                                                                                                                                                                                                                                                                                      |
| lag                      | int        | Lag for the Granger causality computations                                                                                                                                                                                                                                                                                            |
| rcond                    | float      | Cutoff ratio for small singular values during the least squares computations, as defined in <code>numpy.linalg.lstsq</code> : When solving a linear system with a least squares solvers, all singular values of the system matrix that are smaller than <code>rcond</code> times its largest singular value will be treated as zeros. |
| precompute_matrix_blocks | bool       | If True, intermediate results that are needed repeatedly are precomputed and stored. This increases the memory requirements, but can lead to considerable runtime improvements.                                                                                                                                                       |
| save_lstsq_norms         | bool       | If True, the solution norm and residual norm of the most recent least squares computation are stored (allows the user to inspect the behaviour of the least squares solver).                                                                                                                                                          |
| use_lapack_gelsy         | bool       | Controls the underlying computation method for the least squares solver. If True, a QR decomposition based method will be used instead of the default SVD-based method.                                                                                                                                                               |

Table 5.2.: Description of the constructor arguments of the `StochasticComplexity` class.

The final constructor argument, `**_`, exists to gather any remaining keyword arguments into a dictionary (which is then never used). This allows the `StochasticComplexity` class to tolerate the presence of keyword arguments that it does not expect. Such a mechanism is necessary because, as explained earlier, the wrapper class passes on all user-provided keyword arguments, without distinguishing whether they are intended for the criterion or the optimizer. Any keyword arguments that are actually intended for the optimizer will end up in this dictionary, where they can be safely ignored by the `StochasticComplexity` class.

The constructor signature is shown in Listing 5.6.

```
1 def __init__(self, X, lag = 3, rcond = None, precompute_matrix_blocks = True,
    save_lstsq_norms = False, use_lapack_gelsy = False, **_):
```

Listing 5.6: Constructor signature of the `StochasticComplexity` class.

## 5.5. Optimizers

An optimizer searches for the best **assignment** vector, i.e. the vector that best describes the true Granger-causal influences on a target time series. For this purpose, it receives a criterion object as a constructor argument. During the optimization, it calls this object's `score` method (as defined by the criterion duck typing interface) in order to evaluate different assignment vectors. The vector among them that produces the lowest score is chosen as the final result. In addition, the optimizer also tracks the vector of connection strength values (i.e. the second return value of the `score` method) that belongs to the best assignment vector.

The optimizer repeats this search for each of the  $p$  time series in the data set. In this way, it finds  $p$  optimal **assignment** vectors (one for each time series), which are stacked to create the  $p \times p$ -dimensional adjacency matrix.

## 5. Implementation

The implementation offers two optimization methods: an exhaustive search and a genetic algorithm.

### 5.5.1. Exhaustive Search

The idea of the exhaustive search is simple: Iterate over all possible versions of the **assignment** vector and find the one with the lowest score.

For a data set with  $p$  time series, this approach has to evaluate  $2^p$  different vectors to find one row in the final adjacency matrix. Overall, the criterion's **score** method is called  $p * 2^p$  times to find the complete matrix. This approach quickly becomes unfeasible for even moderately large data sets, especially if the score computations contain time-consuming operations such as matrix multiplications (which is the case with the stochastic complexity); but as long as the number of time series is small enough, using an exhaustive search approach is a viable option.

The **ExhaustiveSearch** class, which implements this strategy, follows the estimator interface. All user-defined input is set in the constructor call. There is in fact not much user input needed for this approach: In addition to the criterion object, there is an optional boolean flag to switch on verbose output, which can serve as a useful progress tracker. Other than that, the only remaining constructor argument serves to catch any unused keyword arguments, using the same mechanism that was already explained in Chapter 5.4.2. The constructor signature is shown in Listing 5.7.

```
1 def __init__(self, criterion, verbose=False, **_):
```

Listing 5.7: Constructor signature of the ExhaustiveSearch class.

The feature extraction process is started by calling the **fit** method, as specified by the estimator interface. For each of the  $p$  target time series, this method evaluates all possible assignment vectors and appends the best vector to the final adjacency matrix. The most important parts of the code are shown in Listing 5.8. Code that is only relevant for progress tracking or optional verbosity output was left out of this Listing in order to keep it concise.

```
1 def fit(self, X):
2     # access the criterion object that was passed to the constructor
3     criterion = self.criterion
4
5     # find the dimensionality of the data
6     n_timeseries, n_measurements = X.shape
7
8     # create a list of all possible assignment vectors
9     assignments = list(map(list, itertools.product([0,1], repeat=n_timeseries)))
10
11     # initialize empty lists that will in the end store the results: a pxp
12     # adjacency matrix, a pxp matrix of beta values, and a list of p scores (
13     # one per row in the adjacency matrix)
14     adjmat = []
15     betamat = []
16     adjmat_scores = []
17
18     # find the best fit for each time series
19     for i in range(n_timeseries):
20
21         # initialize the current best score as infinity
22         best_result = float('inf')
23
24         # go through all possible assignments and evaluate them
25         for assignment in assignments:
26
27             # try to compute the score and beta value
28             try:
```

```

27         current_result, current_beta = criterion.score(i, assignment) #
           this line might raise a ValueError
28
29         # check if there is a new best score
30         if current_result < best_result:
31             best_result = current_result
32             best_assignment = assignment
33             best_beta = current_beta
34         # if an assignment is invalid, just ignore it
35         except ValueError:
36             pass
37
38         # the best assignment will be the next row in the adjacency matrix
39         adjmat.append(best_assignment)
40
41         # also save the beta vector and score
42         betamat.append(best_beta)
43         adjmat_scores.append(best_result)
44
45         # turn the results into numpy arrays and expose them as public attributes
46         self.adjacency_matrix_ = np.asarray(adjmat)
47         self.beta_matrix_ = np.asarray(betamat)
48         self.adjacency_matrix_scores_ = np.asarray(adjmat_scores)
49
50         # return self to be consistent with the sklearn estimator interface
51         return self

```

Listing 5.8: The fit method of the ExhaustiveSearch class.

### 5.5.2. Genetic Algorithm

If the number of time series in the data set is too large, conducting an exhaustive search is no longer feasible, and search heuristics are needed to find the best assignment vectors. For this purpose, the implementation provides a genetic algorithm optimizer, as proposed in [7], which is implemented in the `GeneticAlgorithm` class. It follows the estimator interface.

#### General Concept

Genetic algorithms (GA) are inspired by natural genetic processes [54]. Consider an extremely simplistic sketch of survival and procreation of populations: Each population member has certain traits that are determined by its genes. Depending on those genes, some individuals are fitter for survival in their environment than others. Since fitter individuals are more likely to survive, they are also more likely to be able to procreate and pass their genes on to the next generation. The genes of their children will then be recombinations of the genes that enabled the parents to survive in the first place, ideally creating new individuals with even better fitness.

A GA takes this general idea and applies it to an optimization problem. The population members are possible solutions to this optimization problem; usually, they are represented in binary form. Each bit in a solution is considered as a gene. The fitness of individuals is determined by the objective function that should be optimized. For a maximization problem, higher objective function values symbolize better fitness, and for a minimization problem, lower objective function values indicate better fitness.

A more detailed introduction to genetic algorithms can for example be found in [54]. The following explanations use the same terms and definitions that are established in this source.

A GA starts with a population of randomly created individuals. It then applies three operators to create a new generation:

- **Reproduction:** A subset of the current population is chosen to survive into the next generation. The fitter an individual is, the more likely it is to be included in the reproduction.

## 5. Implementation

- **Crossover:** Genes from existing population members are recombined to create new population members. Usually, two parents are chosen, and some of their genes (i.e. bits) are exchanged. This step is based on the assumption that a recombination of two solutions with good fitness values is likely to create a new solution with even better fitness.
- **Mutation:** In some population members, random genes are mutated (i.e. random bits are flipped in the binary representation). The probability of such mutations is usually kept rather low, since otherwise the GA would degenerate to a random search. However, the presence of a small number of mutations is essential in order to maintain diversity in the population and keep the algorithm from converging prematurely to a suboptimal solution.

Once the new generation is complete, the fitness scores of its members are computed. The algorithm is checked for convergence, which is achieved either once a user-defined number of generations have been created, or when some additional convergence criterion is fulfilled (e.g. a solution with a target fitness value was found, or the best solution hasn't changed for a certain number of generations). If the algorithm hasn't converged, then it continues at the reproduction stage with the creation of the next generation. There exist a large number of variations for each of the three operators, but the basic structure of a GA always follows this general outline.

### Operators Used in this Implementation

In the case of the ITGGM method, a population consists of different assignment vectors, i.e. binary vectors of length  $p$ . The fitness of each population member is determined by its information-theoretic criterion score, where lower scores indicate better fitness.

The general setup of the GA is controlled by two user-defined parameters: `n_generations` and `population_size`. `population_size` defines how many individuals there are in the population in each generation, while `n_generations` defines the number of new generations that will be created after the initial random population (there is no additional convergence criterion).

While there are default values provided for both of these parameters, their applicability depends strongly on the use case. The more bits ("genes") there are in the binary representation of the population members, the larger the search space becomes, and therefore more generations and larger populations might be needed to obtain a good result. These parameters should therefore always be specified by the user.

The `GeneticAlgorithm` class uses the following reproduction, crossover, and mutation operators:

- **Reproduction operator.** The reproduction strategy used in this implementation is simply to copy the population members with the best fitness scores into the next generation; these population members are referred to as the "elite". The user-defined `elite_fraction` parameter controls the size of the elite, which is expressed as a fraction of the total population size. Per default, it is set to 0.5, which means that the half of the population with the best scores gets copied into the next generation.
- **Crossover operator.** The remaining members of the new generation are created by crossover of elite members. The size of the population is kept constant over all generations, therefore the `elite_fraction` parameter implicitly also regulates the number of population members that are created via crossover. If for example the elite fraction is 0.3, then 30% of the new generation are created by copying the best members from the previous generation, and the remaining 70% are created by crossover of these elite individuals.<sup>4</sup>

---

<sup>4</sup>In the literature, the number of population members that are created by crossover and reproduction is often regulated by a parameter called *crossover probability* instead of the elite fraction. The two concepts are very closely related; in this implementation the elite fraction is used in order to stay closer to the conventions established in [7].

For a crossover operation, two parents are chosen from the current elite. This happens via a tournament selection: For each parent, a number of candidates are drawn at random from the elite, and the one among them with the best score is chosen as parent. This ensures that fitter individuals have a higher likelihood to be chosen as parents, while still keeping a certain amount of diversity. The default tournament size is 3 (i.e. each parent is the best of three randomly chosen candidates), but this can be changed by the user via the `tournament_size` parameter.

Once the parents are chosen, a random crossover location is selected. The child is created by copying all genes (i.e. bits) to the left of the crossover location from the first parent, and copying all genes at and to the right of the crossover location from the second parent.

The implementation does not verify whether the two parents are actually distinct from each other. As the algorithm nears convergence, population members are increasingly likely to be duplicates of each other anyways, which is an expected property of a GA; therefore, the possibility of creating a crossover of two parents with identical bit values, or even by chance picking the same individual as first and second parent, is an accepted part of this crossover operator.

- **Mutation operator.** Once all spots in the new generation have been filled through reproduction and crossover, mutation is applied to some random "genes" (bits) in the population. The likelihood of mutations is controlled by the user-defined `mutation_probability` parameter, which expresses the probability that a single gene will change its value. For each gene in the population, a random number between 0 and 1 is drawn from a uniform distribution, and if this number is less than the mutation probability, then the value of the gene is changed.

Figure 5.2 shows visualizations of these operators.

### Additional Details

The GA in this implementation has a few additional characteristics that are worth pointing out.

The code remembers the best individual that was encountered over all generations. Even if this best individual is not part of the final generation (which might happen due to the mutations), it is still returned as the final result.

As already mentioned briefly, GAs are expected to create duplicate population members during their trajectory, especially in later generations. It is desirable to not have to recompute the fitness values for these duplicate individuals over and over again, since the computation of the criterion score can be quite time-consuming. By default, the implementation therefore keeps a history of the scores of all previously encountered population members. This behaviour can be turned off by setting the `remember_scores` parameter to `False`.

Contrary to standard GAs, this implementation also has to deal with the possibility that score (i.e. fitness) computations might raise exceptions if invalid population members are encountered. It reacts to such a case by replacing the faulty population member with a newly generated individual. The aim here is to choose a replacement strategy that changes the algorithm's behaviour as little as possible compared to a standard GA. The implementation therefore distinguishes between three different cases:

- If an exception is raised in the very first generation, where the population still consists entirely of randomly generated individuals, then the invalid member is replaced by a new random member.
- If the invalid member is one of the individuals that were copied over from the previous generation's elite, then the problem can only have been created by a mutation of this

## 5. Implementation

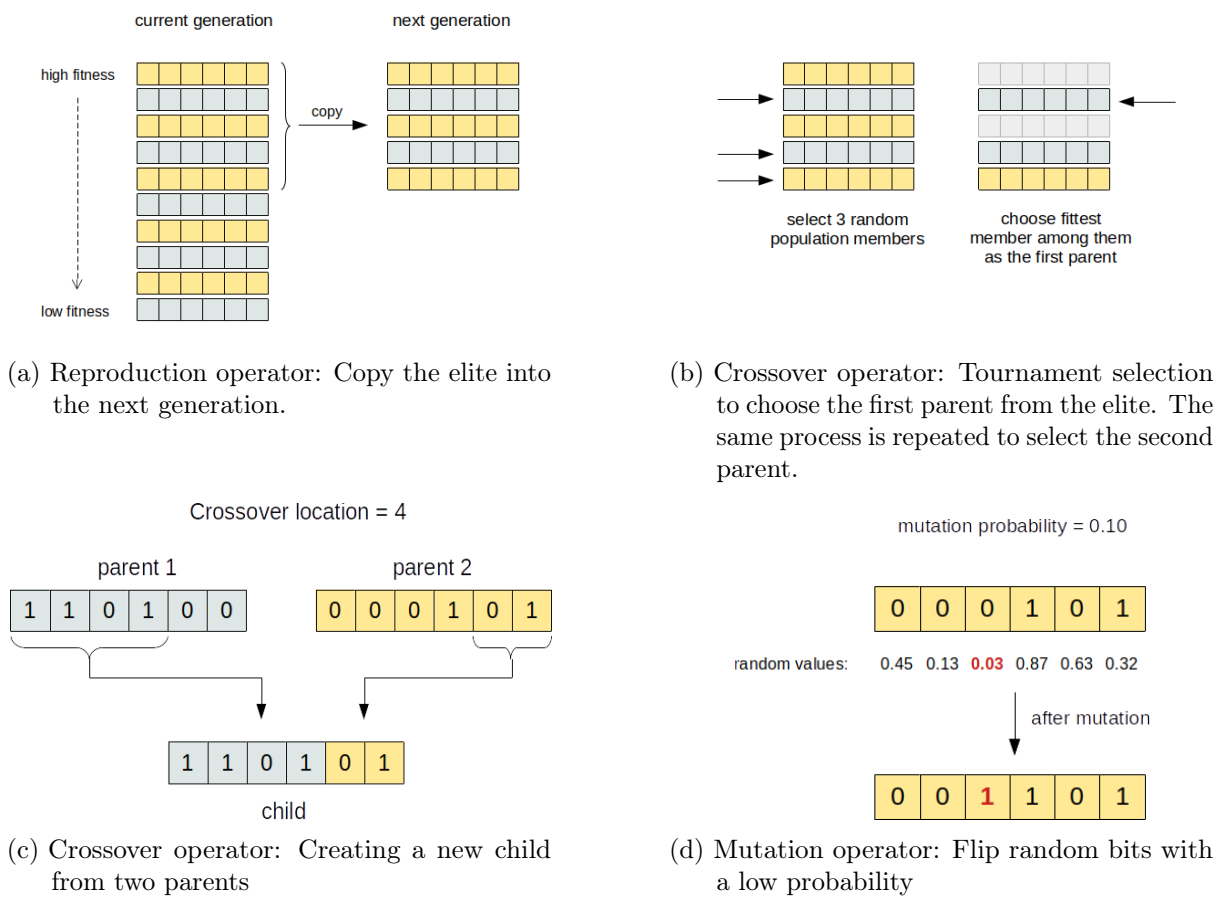


Figure 5.2.: Genetic algorithm operators

individual. Replacement is performed by simply introducing another mutation, i.e. flipping a random bit in this individual.

- Any of the remaining population members in the current generation were created with the crossover operator (and then might or might not have had mutations applied to them). If one of them is invalid, it is therefore replaced by a new crossover of two elite members.

This replacement strategy does not exactly reproduce the behaviour of the undisturbed GA, as it might undo existing mutations or add new ones, but these slight deviations are considered acceptable. If the replacement process should happen to create yet another invalid individual, the same process is repeated until a valid replacement has been found.

There are two more parameters that can be defined by the user in this GA version. The `n_runs` parameter is a commonly used parameter in scikit-learn algorithms that work with random initializations. It is intended to counteract possible effects of an unlucky random starting population: The whole optimization process is repeated `n_runs` times with different random initializations, and the overall best result is reported. However, as already pointed out, the GA's mutation probability parameter also has the purpose of helping it to escape from local optima, so in most cases using multiple runs should not be necessary if the other GA parameters are chosen reasonably. The `random_seed` parameter on the other hand offers the possibility to seed the random generator, thus making it possible to exactly reproduce previous results.

## Code Extracts

The constructor signature of the `GeneticAlgorithm` class is shown in Listing 5.9. It lists all keyword arguments that are accepted by this class and shows their default values. As with the `StochasticComplexity` and `ExhaustiveSearch` classes, the final constructor argument exists to catch any unused keyword arguments.

```
1 def __init__(self, criterion, n_generations = 10, population_size = 10,
    elite_fraction = 0.5, tournament_size = 3, mutation_probability = 0.1,
    remember_scores = True, n_runs = 1, random_seed = None, **_):
```

Listing 5.9: Constructor signature of the `GeneticAlgorithm` class.

The optimization procedure is started by calling the optimizer's `fit` method, as specified by the estimator interface. The code for this method is shown in Listing 5.10. This Listing only provides a high-level view of the implementation logic, as the `fit` method contains multiple calls to other methods which are not shown here. As in the case of the exhaustive search, the optimization with the genetic algorithm is repeated  $p$  times in order to create the  $p$  rows in the adjacency matrix.

```
1 def fit(self, X):
2     # find the dimensionality of the data
3     self.n_timeseries, self.n_measurements = X.shape
4
5     # initialize empty lists that will in the end store the results
6     adjmat = [] # the adjacency matrix of the causal graph
7     betamat = [] # beta values corresponding to the adjacency matrix entries
8     adjmat_scores = [] # the criterion scores for each row in the adjacency
        matrix
9
10    # find the best fit for each time series
11    for i in range(self.n_timeseries):
12        # initialize the dictionary to remember all previously computed scores
            (saves computation time)
13        # the score also depends on i, so we need to create a new history
            whenever the value of i changes
14        if self.remember_scores:
15            self._history = dict()
```

## 5. Implementation

```
16
17     # keep track of the overall best result
18     overall_best_score = float('inf')
19
20     # repeat the optimization n_runs times
21     for _ in range(self.n_runs):
22         # create the initial random population
23         population = self._create_random_population()
24
25         # compute the score for each population member
26         # any invalid members will be replaced by random individuals
27         scores, beta_vectors = self._compute_scores(population, i,
28             replacement_strategy=self._ReplacementStrategy.RANDOM)
29
30         # keep track of the overall best result
31         best_member, best_beta, best_score = self._get_best_member(
32             population, beta_vectors, scores)
33         if best_score < overall_best_score:
34             overall_best_score = best_score
35             overall_best_assignment = np.copy(best_member)
36             overall_best_beta = np.copy(best_beta)
37
38         # create the specified number of generations
39         for _ in range(self.n_generations):
40             # create the new generation by applying reproduction and
41             # crossover
42             population = self._create_next_generation(population, scores)
43
44             # randomly mutate some members of the population
45             population = self._apply_mutations(population)
46
47             # compute the score for each population member
48             # invalid members will be replaced by either mutation or
49             # crossover
50             scores, beta_vectors = self._compute_scores(population, i,
51                 replacement_strategy=self._ReplacementStrategy.OFFSPRING)
52
53             # check if there is a new best result
54             best_member, best_beta, best_score = self._get_best_member(
55                 population, beta_vectors, scores)
56             if best_score < overall_best_score:
57                 overall_best_score = best_score
58                 overall_best_assignment = np.copy(best_member)
59                 overall_best_beta = np.copy(best_beta)
60
61         # pick the overall best member and score as result
62         adjmat.append(overall_best_assignment)
63         betamat.append(overall_best_beta)
64         adjmat_scores.append(overall_best_score)
65
66     # turn the results into numpy arrays and expose them as public attributes
67     self.adjacency_matrix_ = np.asarray(adjmat)
68     self.beta_matrix_ = np.asarray(betamat)
69     self.adjacency_matrix_scores_ = np.asarray(adjmat_scores)
70
71     # return self to be consistent with the sklearn estimator interface
72     return self
```

Listing 5.10: The fit method of the GeneticAlgorithm class.

## 6. Experiments

This Chapter describes the results of the experiments that were conducted as part of this thesis.

There are two distinct parts to the experiments. In the first step, Granger-causal networks are computed from the EEG recordings. In the second step, classifiers are trained to distinguish between responders and nonresponders based on these networks.

### 6.1. Granger Causality Computations

The implementation described in Chapter 5 was used to find Granger-causal relationships in the EEG data, according to the computation process detailed in Chapter 4.2. However, before this, some preliminary experiments were carried out, firstly to verify the functionality of the implementation on synthetic data, and secondly to estimate appropriate parameters for the Granger causality computations on the EEG data sets.

#### 6.1.1. Proof of Concept: Synthetic Data Sets

In order to demonstrate that the Granger causality computations produce meaningful results, they were applied to synthetic data where the true causal relationships are known.

For this purpose, two synthetic data sets were used that have already been employed elsewhere in the literature for testing Granger causality applications. The first one, a five-point network, was published as part of a Matlab implementation of Kernel Granger Causality [55]. Its definition is shown in Equation 6.1. The second data set is a ten-point network, which is defined in the supplementary material of [7] and shown in Equation 6.2.

$$\begin{aligned}x_0(t) &= +0.95 * \sqrt{2} * x_0(t-1) - 0.9025 * x_0(t-2) + \epsilon_0(t) \\x_1(t) &= +0.5 * x_0(t-2) + \epsilon_1(t) \\x_2(t) &= -0.4 * x_0(t-3) + \epsilon_2(t) \\x_3(t) &= -0.5 * x_0(t-2) + 0.25 * \sqrt{2} * x_3(t-1) + 0.25 * \sqrt{2} * x_4(t-1) + \epsilon_3(t) \\x_4(t) &= -0.25 * \sqrt{2} * x_3(t-1) + 0.25 * \sqrt{2} * x_4(t-1) + \epsilon_4(t)\end{aligned}\tag{6.1}$$

$$\begin{aligned}x_0(t) &= 0.4 * x_0(t-1) - 0.5 * x_1(t-2) + 0.4 * x_4(t-1) + \epsilon_0(t) \\x_1(t) &= 0.4 * x_1(t-1) - 0.3 * x_0(t-4) + 0.4 * x_4(t-2) + \epsilon_1(t) \\x_2(t) &= -0.5 * x_2(t-1) - 0.7 * x_2(t-2) - 0.3 * x_4(t-3) + \epsilon_2(t) \\x_3(t) &= 0.8 * x_3(t-3) + 0.4 * x_0(t-2) + 0.3 * x_1(t-2) + \epsilon_3(t) \\x_4(t) &= 0.7 * x_4(t-1) - 0.5 * x_4(t-2) - 0.4 * x_3(t-1) + \epsilon_4(t) \\x_5(t) &= 0.4 * x_3(t-1) - 0.5 * x_2(t-2) - 0.4 * x_1(t-1) + \epsilon_5(t) \\x_6(t) &= -0.5 * x_4(t-1) - 0.5 * x_1(t-2) - 0.4 * x_0(t-1) + \epsilon_6(t) \\x_7(t) &= 0.5 * x_4(t-1) - 0.5 * x_0(t-2) - 0.4 * x_6(t-1) + \epsilon_7(t) \\x_8(t) &= 0.4 * x_2(t-1) - 0.5 * x_1(t-2) - 0.4 * x_0(t-1) + \epsilon_8(t) \\x_9(t) &= 0.4 * x_4(t-1) - 0.5 * x_1(t-2) - 0.4 * x_8(t-1) + \epsilon_9(t)\end{aligned}\tag{6.2}$$

## 6. Experiments

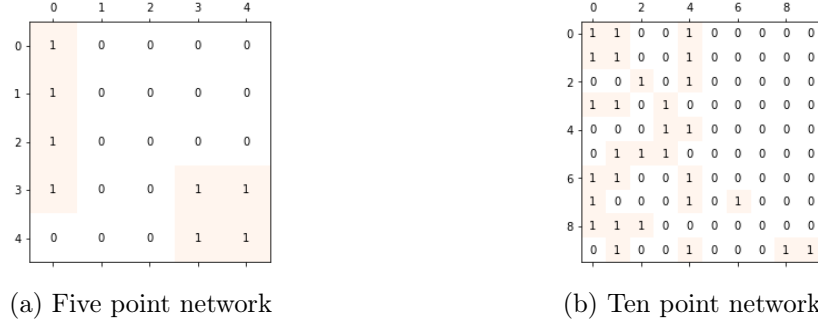


Figure 6.1.: True Granger-causal adjacency matrices for the synthetic data sets.

The  $\epsilon$  terms in the data set definitions are random noise that follows a normal distribution. For the experiments presented here, no seed was provided for the random generator that produced the noise components. As a consequence, the data sets are slightly different each time they are generated.

Knowing the defining equations, it is possible to infer the true adjacency matrices for both networks. They are shown in Figure 6.1. In the same way it is possible to determine the true lag for each network: This is the maximum lag in any of the equations in the network's definition. For the five-point network, the true lag is 3, while the ten-point network has a true lag of 4.

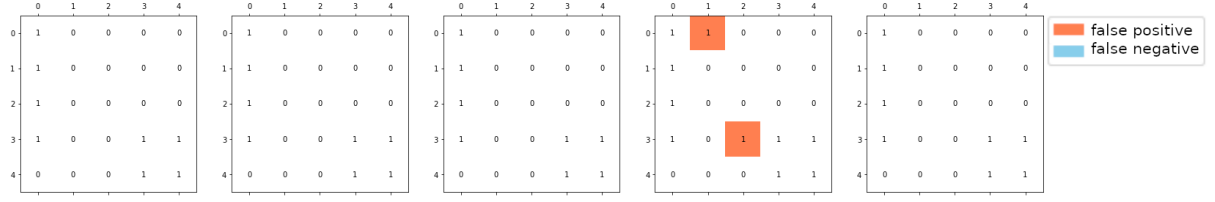
The implementation described in Chapter 5 was used to compute Granger-causal relationships within these two data sets.

In a first run of experiments, the exhaustive search optimizer was used, which is still feasible with this number of time series. To investigate the influence of the random noise terms, five versions of each data set were generated (which are identical except for their noise components), and the Granger causality computations were repeated for each of these versions. The resulting Granger-causal matrices were compared to the known true adjacency matrices. As a measure for the quality of the results, the F1 score between the output matrices and the true adjacency matrices was computed.

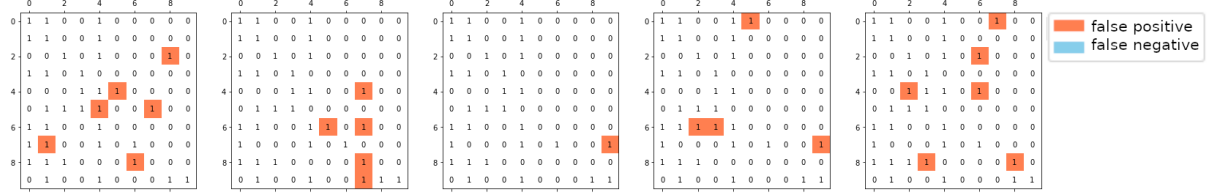
For the five-point network, the ITGGM method was almost always able to recover the true causal influences perfectly, with only a very few wrong matrix entries in one of the results. The F1 score for this result was 0.89 (for the other results that perfectly recovered the ground truth, the F1 score was of course 1.0, which is the best possible result). For the ten-point network, there are some wrong entries in all of the output matrices, but the majority of the Granger-causal influences were correctly recovered. The results had F1 scores ranging from 0.91 to 0.98. From these findings, it can be inferred that the Granger causality computations indeed capture the true causal relationships quite well, although the error in the result is influenced by the noise in the data. The results are visualized in Figure 6.2.

An interesting observation is that all the wrong results in any of these experiments are false positives - there is not a single false negative in the created adjacency matrices. In other words, the ITGGM method achieved a recall of 100% on all versions of the synthetic data sets: It sometimes assumed causal relationships where there were none, but it never missed any of the true relationships.

In the next step, experiments with the genetic algorithm optimizer were carried out on the ten-point network. The purpose of these experiments was to demonstrate how an inappropriate choice of GA parameters can impact the quality of the result. This time, only one version of the data was generated, which was used as input for multiple optimization runs with varying GA parameters. Different combinations of number of generations and population size were tried out, the lag was set to the true lag for the ten-point network, and all other parameters were left at their default values. Since the random noise in the data was the same for all of these runs, any changes in the result are due to changes in the parameters. The results are shown in Figure 6.3a.

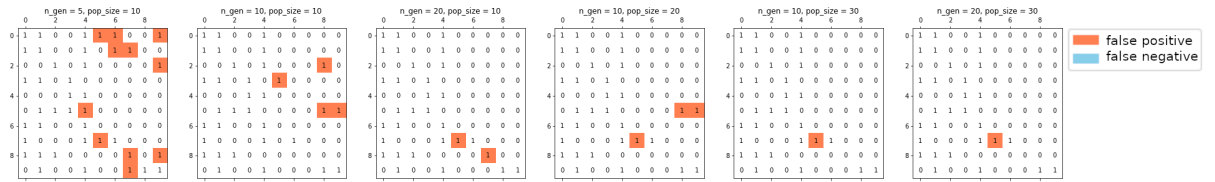


(a) Result of the exhaustive search on data sets created according to the five-point network specification.

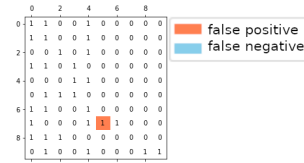


(b) Result of the exhaustive search on data sets created according to the ten-point network specification.

Figure 6.2.: Results of the Granger causality computations with exhaustive search on the synthetic data sets. False positives are marked in color (the results do not contain any false negatives). The five adjacency matrices per data set correspond to five versions of this data set that have different random noise components.



(a) Genetic algorithm results with different parameters on a data set created according to the ten-point network specification.



(b) Result of the exhaustive search on the same data set.

Figure 6.3.: Comparison of the Granger causality computations with genetic algorithm and exhaustive search on the synthetic ten-point network.

For comparison, Figure 6.3b shows the result of the exhaustive search optimizer on the same data set.

As expected, an appropriate choice of number of generations and population size is required to achieve high-quality results. The best GA results are the same as the result of the exhaustive search: In other words, if the parameter values were set appropriately, the GA was able to recover the optimal solution. (Note that "optimal" in this case refers to the solution with the minimum stochastic complexity score for each matrix row, i.e. the result produced by the exhaustive search, and not necessarily to the true solution.)

### 6.1.2. Parameter Estimation

The computation of the Granger-causal networks from the EEG data requires a number of input parameters, some of which can have a huge impact on the result quality. The most important one of them is probably the lag parameter. Since the optimal values for these parameters are unknown in real-world data, preliminary experiments had to be performed to estimate them.

## 6. Experiments

For this process, a useful property of the ITGGM method was exploited. As explained in Chapter 3.3, the method assigns an information-theoretic criterion score to each row in the adjacency matrix that indicates how well this assignment of Granger-causal relationships fits the data: Lower scores indicate a better fit. It is therefore reasonable to assume that if parameter values are chosen appropriately, then the scores in the result will be lower than if the chosen parameters are suboptimal for the data set. Consequently, the average value of the  $p$  scores belonging to an adjacency matrix can be used as an indication of the goodness of fit of this matrix.

To estimate the best parameters for the EEG data, 50 epochs were randomly sampled from the data set. Different parameters were tried out on them, and the average stochastic complexity scores of the resulting adjacency matrices were evaluated. The parameters that yielded the best average score for the majority of those epochs were chosen as the final settings.<sup>1</sup>

As explained in Chapter 4, the Granger causality computations are carried out for nine different settings, defined by the different combinations of brain hemisphere and frequency band. The parameter estimation process was performed for each of these scenarios separately. In the cases where data from only eight or eleven electrodes is considered, the exhaustive search optimizer was used. In the cases with nineteen electrodes however, this was not feasible even for only 50 epochs. Instead, the genetic algorithm optimizer was used, with the number of generations and the population size intentionally chosen very high (150 and 300, respectively) to make the result as reliable as possible.

### Synthetic Data Sets

To verify that the parameter estimation approach is reasonable, it was first used to estimate the lag of the synthetic data sets, since here the true lags are known and therefore the performance of the estimation method can be assessed. Once again, the experiments were repeated multiple times to account for the effect of the random noise components. Figure 6.4 shows the average stochastic complexity scores that resulted from experiments with different lag values on the synthetic data sets.

There are two distinct scenarios in the results:

With the five-point network, there is a clear minimum in the result, which coincides with the network's true lag. In this case, the estimation method was able to identify the correct lag.

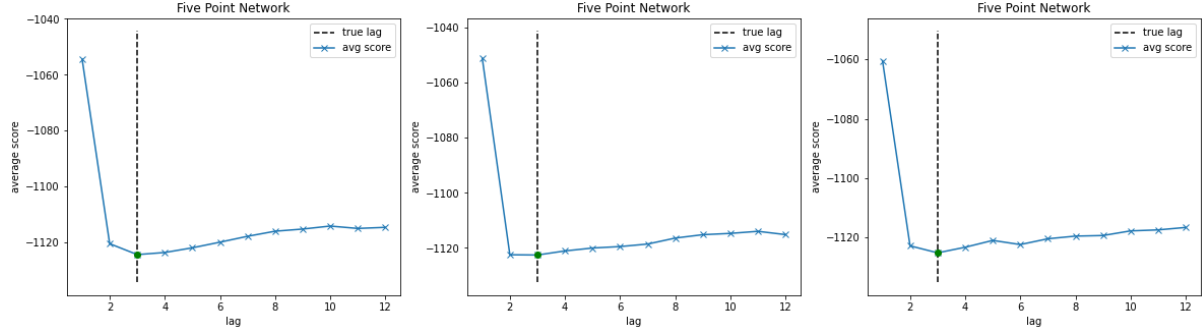
With the ten-point network, the average score continuously decreases as the lag is increased; there is no clear minimum. In this case, it makes sense to use the *elbow method* to choose the point as optimum where there is the most pronounced bend, or "elbow", in the plot. Based on the plots in Figure 6.4b, one would likely choose the value 3 as lag by following the elbow method, even though the true lag for this network is 4. However, when looking more closely at the defining equations for this network, it turns out that there is only one single term in the definitions of all ten time series that actually uses the lag 4, while all other terms have a lag of 3 or less. By choosing the lag according to the elbow method, one would therefore lose hardly any information (which is probably the reason why the decrease in average score from lag 3 to lag 4 is so small to begin with).

### Lag

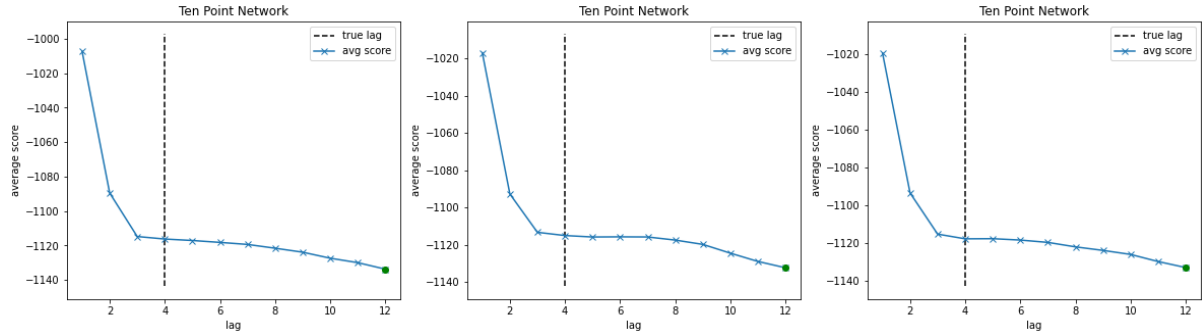
For each of the nine input settings, lags in the range from 1 to 20 were evaluated on the 50 sample epochs. Figure 6.5 shows a visualization of the resulting average stochastic complexity scores for one input setting.

---

<sup>1</sup>Evaluating an information-theoretic criterion for different lag lengths in order to find the best value is in fact the standard process for lag selection in autoregressive problems; the Akaike Information Criterion is a common choice for this purpose (cf. [56]). In the present case, it was a logical choice to use the stochastic complexity, since this criterion is also used for the computation of the Granger-causal networks.



(a) Lag estimation process for the synthetic five-point network. The true lag coincides with the minimum score.



(b) Lag estimation process for the synthetic ten-point network. There is no clear minimum score, instead the average score generally tends to decrease if the lag is increased. However, a value close to the true lag can be recovered using the elbow method.

Figure 6.4.: Demonstration of the lag estimation process on synthetic data. Three versions of each data set were generated to account for possible effects of the random noise components in the data. In each plot, the true lag is marked by a dashed line, while the minimum score is indicated by a green dot.

Table 6.1 summarizes the optimal lags that were found for all nine settings. The lag seems to be largely determined by the frequency band: In general, if the brain hemisphere is changed, but the frequency band stays the same, then the optimal lag also stays the same. The only exception to this is in the full-brain alpha band data, which has a different optimal lag than the other input settings that use the alpha band.

## Genetic Algorithm Parameters

For the scenarios that use data from all 19 electrodes, computations have to be done with the genetic algorithm optimizer, which depends on a number of parameters. The estimation process was conducted for the number of generations, population size, and mutation probability. All other GA parameters were left at their default values.

Interestingly, none of the examined GA parameters seem to have a large impact on the quality of the result. As an example, Figure 6.6 shows a visualization of the estimation process for the mutation probability on the unfiltered data. In contrast to the lag estimation results shown in Figure 6.5, the different mutation probabilities hardly seem to influence the average stochastic complexity score. Additionally, none of the values is chosen as optimum significantly more often than all others. The results for the alpha and theta frequency bands, which are not shown here, exhibit very similar behaviour.

Since the mutation probability apparently only has a limited influence on the quality of the result, it was not considered necessary to choose a different mutation probability for each

## 6. Experiments

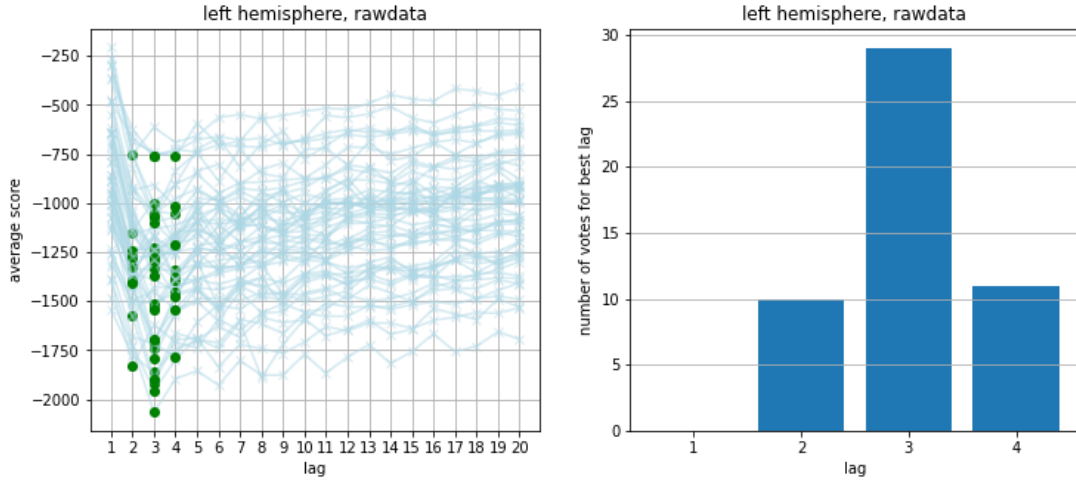


Figure 6.5.: Lag estimation process for data from the left brain hemisphere, using the unfiltered EEG data. Left: Average stochastic complexity scores achieved with different lags on each of the sample epochs. The minimum result for each sample epoch is marked with a green dot. Right: Bar plot that shows how many epochs had their minimum score at each lag value. The value 3 was the optimal lag for the majority of the sample epochs, and therefore this value was chosen as the final lag.

| Hemisphere                  | Frequency Band | Lag |
|-----------------------------|----------------|-----|
| Left                        | Alpha          | 5   |
|                             | Theta          | 2   |
|                             | Unfiltered     | 3   |
| Left with center electrodes | Alpha          | 5   |
|                             | Theta          | 2   |
|                             | Unfiltered     | 3   |
| Both                        | Alpha          | 2   |
|                             | Theta          | 2   |
|                             | Unfiltered     | 3   |

Table 6.1.: Results of the lag estimation process for each input data setting.

frequency band. Instead, one single value was chosen that did reasonably well for all three bands. Accordingly, the mutation probability was set to 0.11 for all experiments that use the GA.

For the number of generations and population size, the aim was not so much to find a distinctive optimum, since increasing these parameters can only improve the quality of the result. However, if they become too large, at some point this will have a negative impact on the runtime without leading to any further quality improvements. The estimation process was therefore conducted to find the smallest possible values for these parameters that do not negatively impact the result quality.

It turned out that changing the number of generations and population size did not seem to make much of a difference in the resulting scores, as long as neither of these parameters was set to values below 100. To be on the safe side, both the number of generations and the population size were set to 150 for the final experiments.

### Parameters for the Least Squares Solver

With data from all 19 electrodes, using the default SVD-based least squares solver for the score computations occasionally led to numerical issues. Therefore, the QR-decomposition-based

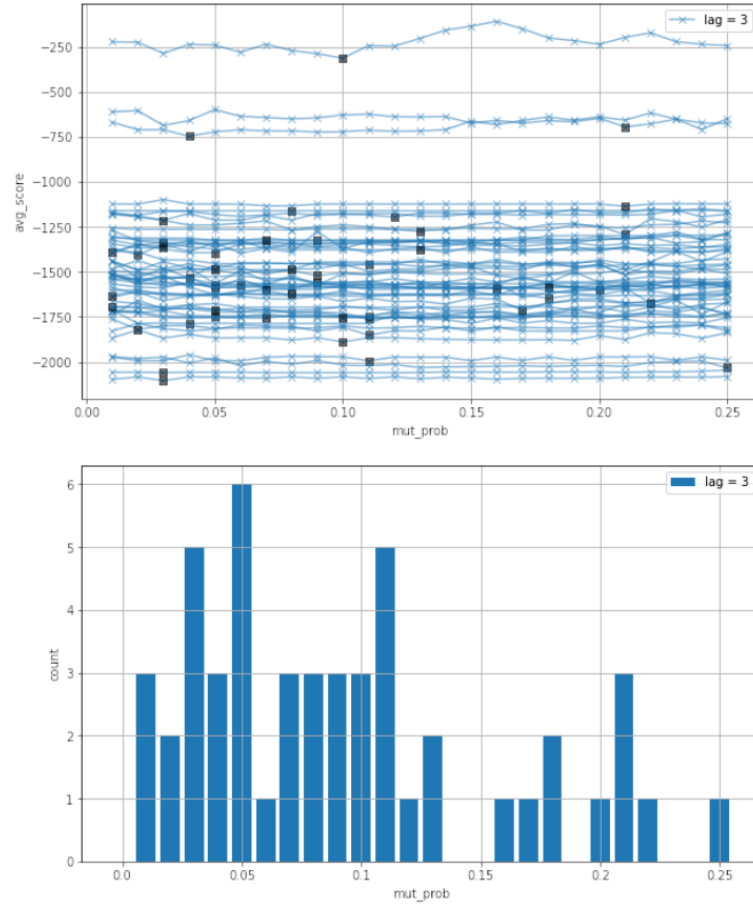


Figure 6.6.: Estimation process for the mutation probability on the unfiltered full-brain data. Top: Average stochastic complexity scores achieved on the sample epochs for different mutation probabilities. The minimum score for each epoch is marked by a black square. Bottom: Bar plot that shows how many epochs had their minimum score at each value. Both plots show that there is no distinctive optimum.

## 6. Experiments



Figure 6.7.: Dummy classification results, showing the average and standard deviation of the F1 score over a 10-fold cross validation.

version was used in the scenarios with 19 electrodes. The input data for the least squares problem also tends to be ill-conditioned (cf. [7]); to avoid numerical problems due to very small singular values, the `rcond` parameter (cf. Table 5.2) was set to  $5 * 10^{-5}$  for all experiments.

### 6.2. Classification Experiments

The data set was split into a training set and a test set, and a number of different classifiers were trained on the training set in various settings using a 10-fold cross validation, as described in Chapter 4.

In order to have a baseline to compare the results to, two dummy classifiers were used. These are classifiers that output class labels according to a pre-defined strategy, while completely ignoring the input data. The "uniform" dummy chooses the class label for each sample at random, with equal probability for each class; the "constant" dummy predicts every subject to be a responder. A real classifier can be considered useful if it achieves a higher F1 score than the dummies, i.e. it provides an information gain compared to the baseline.

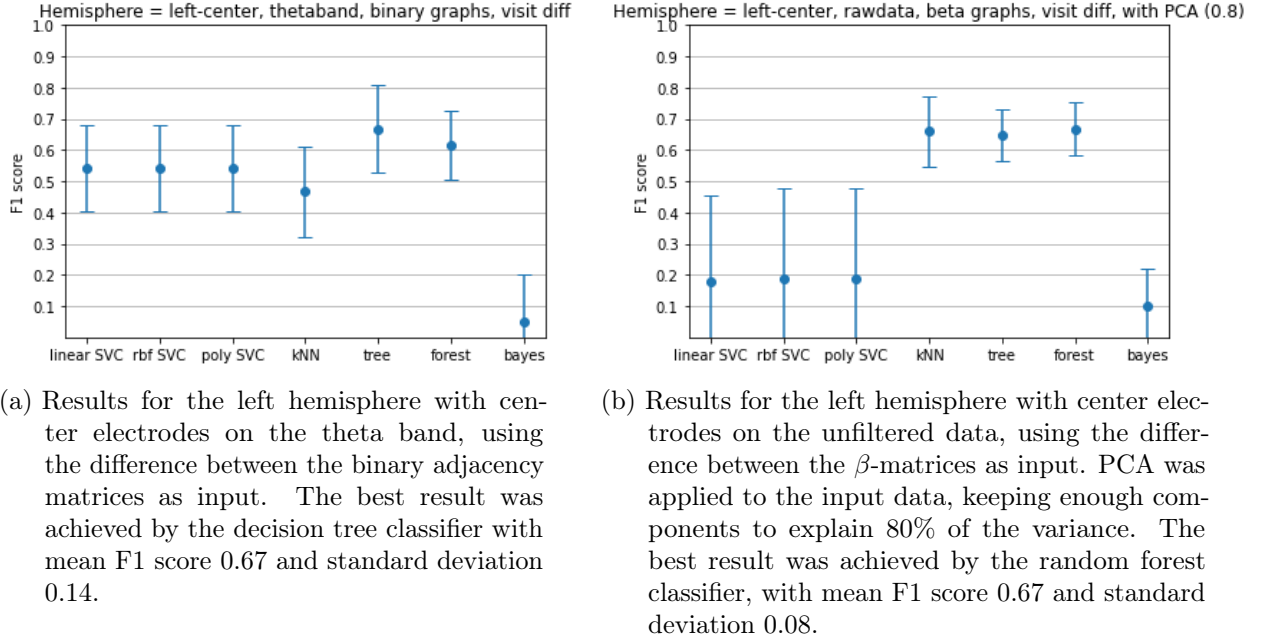
The results achieved by the dummy classifiers are shown in Figure 6.7. The best results were achieved by the constant classifier with a mean F1 score of 0.67 and a standard deviation of 0.03.

Unfortunately, the best results achieved by any of the real classifiers are no better than the results of the best dummy; in fact, the vast majority of them are significantly worse. Visualizations of the best results obtained during cross validation by the real classifiers are shown in Figure 6.8.

#### 6.2.1. Splitting the Data Set

Van Putten et al. [57] point out that biological sex is a confounding factor for EEG recordings, and that there are significant differences between the EEGs of biologically male and female subjects. Bučková et al. [58] replicate this finding on the same data set that is also used in this thesis. Therefore, as a next step, the data set was split into biologically male and female subjects, and the classifiers were trained for each of these groups separately, in order to try to improve the classification performance.

Among the 134 subjects in the training set, there are 103 women and 31 men (the detailed distribution of the training and test sets with respect to biological sex and responder status can be found in Appendix A.2). In the case of the men, this might not be enough samples to produce generalizable results. Furthermore, with so few samples, using 10-fold cross validation is suboptimal, because folds consisting of only 3 samples are unlikely to be representative of the underlying data set. In order to mitigate this problem, 5-fold cross validation was used during the experiments on the split data set. Other than that, the process of computing the baseline



(a) Results for the left hemisphere with center electrodes on the theta band, using the difference between the binary adjacency matrices as input. The best result was achieved by the decision tree classifier with mean F1 score 0.67 and standard deviation 0.14.

(b) Results for the left hemisphere with center electrodes on the unfiltered data, using the difference between the  $\beta$ -matrices as input. PCA was applied to the input data, keeping enough components to explain 80% of the variance. The best result was achieved by the random forest classifier, with mean F1 score 0.67 and standard deviation 0.08.

Figure 6.8.: Best results achieved by any of the classifiers on any of the input data. The plots show the mean and standard deviation of the F1 scores produced by a 10-fold cross validation.

results and training the classifiers was performed in exactly the same way as during the original experiments.

### Results for Female Subjects

On the training set consisting only of women, the best dummy classifier achieved a mean F1 score of 0.68. The dummy classifier results are shown in Figure 6.9a.

The best result achieved by a real classifier on this data set was a mean F1 score of 0.72. This result is shown in Figure 6.9b. While there is not much difference between this result and the dummy result, the fact that the classifier can outperform the dummies at all is already a big improvement compared to the results using the full data set.

### Results for Male Subjects

On the training set consisting only of men, the best dummy classifier achieved a mean F1 score of 0.62. These dummy classifier results are shown in Figure 6.10a.

The best real classification result on this data set produced a mean F1 score of 0.86, which is a significant improvement compared to the dummies. This result is shown in Figure 6.10b.

#### 6.2.2. Test Set Evaluation

Once the settings that lead to the best results on the training set were identified, the classification performance was verified on a test set of hitherto completely unseen data.

Since the best training results were achieved on the split training set, the test set was also split by biological sex. There are 25 women and 17 men in the test set.<sup>2</sup>

<sup>2</sup>It should be pointed out that for the men, the number of responders and nonresponders in the test set is very unbalanced. At the time when the train-test-split was chosen, the working hypothesis was that sex does not make a difference in the classification, and therefore no attention was paid to balancing the number of responders and nonresponders for men and women separately. Since the goal of the experiments with the

## 6. Experiments

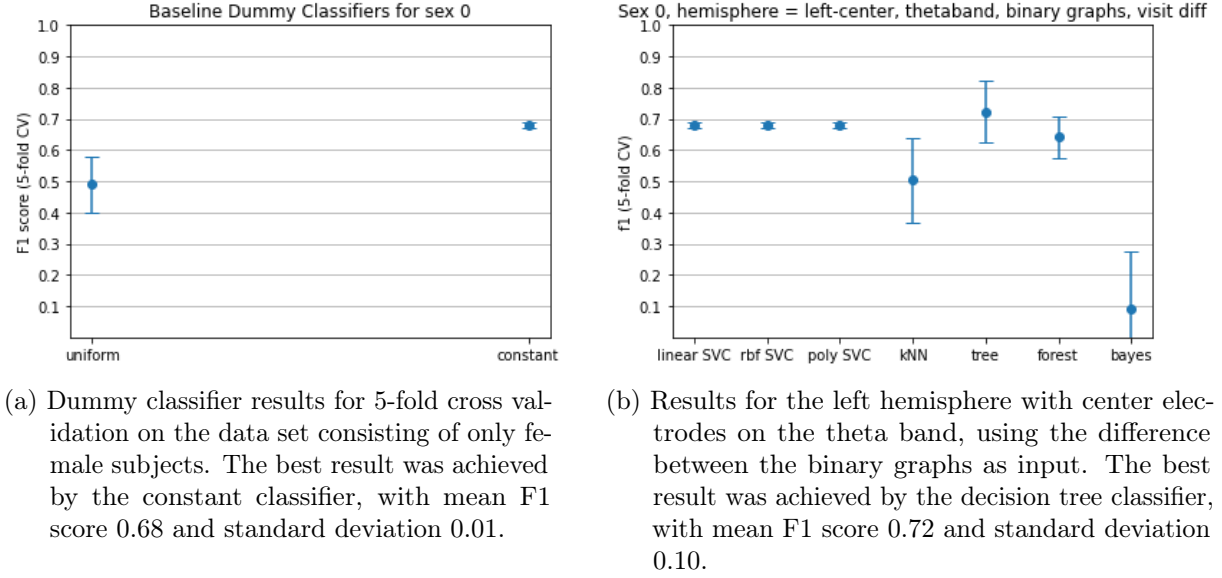


Figure 6.9.: Classification results for female subjects. The plots show the mean and standard deviation of the F1 scores produced by a 5-fold cross validation.

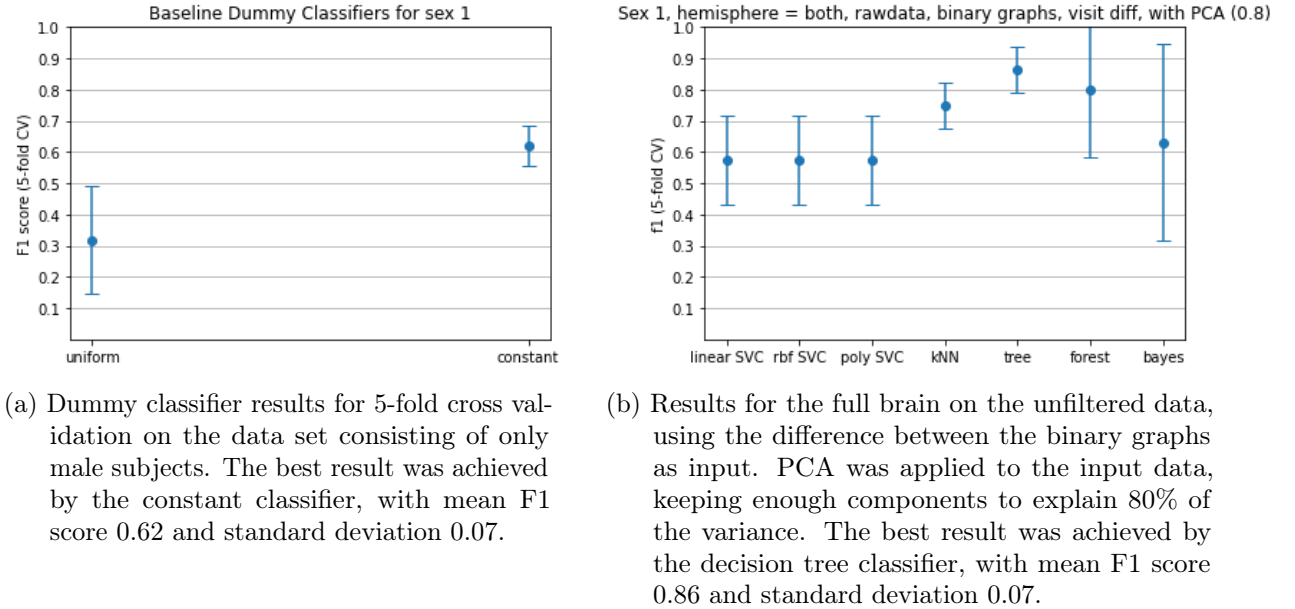


Figure 6.10.: Classification results for male subjects. The plots show the mean and standard deviation of the F1 scores produced by a 5-fold cross validation.

For each of the two groups, the classifier that had previously led to the best results was retrained using all subjects from the training set as input. The test sets were prepared according to the same specifications as their respective training sets; in the case where PCA was used, the test set data was projected onto the Principal Components derived from the training set. All settings that were used for the final evaluation are summarized in table 6.2.

|                                         | Female subjects                        | Male subjects           |
|-----------------------------------------|----------------------------------------|-------------------------|
| <b>Brain hemisphere</b>                 | Left hemisphere with center electrodes | Full brain              |
| <b>Frequency band</b>                   | Theta band                             | Unfiltered              |
| <b>Adjacency matrix (binary / beta)</b> | Binary                                 | Binary                  |
| <b>PCA</b>                              | Not used                               | 80% of variance         |
| <b>Classifier</b>                       | Decision tree                          | Decision tree           |
| <b>Classifier parameters</b>            | 'criterion': 'gini'                    | 'criterion': 'log_loss' |
|                                         | 'max_depth': 3                         | 'max_depth': 7          |
|                                         | 'splitter': 'best'                     | 'splitter': 'best'      |

Table 6.2.: Settings used for the test set evaluation.

For female subjects, the classifier achieved an F1 score of 0.61 on the test set. This indicates a certain level of generalization to unseen data, but is still significantly lower than the score achieved on the training set.

For male subjects, the F1 score on the test set was only 0.2, indicating extremely poor class label predictions. This is the more disappointing as the classification result on the training set was remarkably good in this case. Apparently, the training set size was really too small to find a classifier that can generalize even partially to unseen data.

To examine the performance of the classifiers on the test set a bit more closely, Figure 6.11 shows the confusion matrices for the test set results. These matrices show the number of true positives (bottom right), true negatives (top left), false positives (top right), and false negatives (bottom left) in the classification results.

From them it can be seen that the classifiers perform better at recognizing nonresponders than responders.

The *true negative rate* or *specificity* describes how many of the true nonresponders (top row of the confusion matrix) were correctly identified as nonresponders by the classifier. For female subjects, the true negative rate is 0.75, and for male subjects it is 0.62. In other words, in both cases the majority of nonresponders were identified correctly.

With the true responders (bottom row of the confusion matrix) on the other hand, the result is quite different. The *recall* indicates how many of the true responders were correctly classified as responders. For the female subjects, the recall is 0.54: only about half of the true responders were correctly classified. For the male subjects, the result is even worse: Here the recall is only 0.25, the classifier was only able to correctly identify a single responder.

### 6.3. Discussion

Unfortunately, none of the classification methods that were examined in the frame of this thesis were able to reliably predict treatment response. While for male subjects the results on the training set were quite promising, experiments with the test set showed that the results do not

---

split data set was to find out if this improves the classification result compared to the original training set, it was not possible to redo the train-test-split, as this would have made it hardly justifiable to compare the classification quality to the previous results.

## 6. Experiments

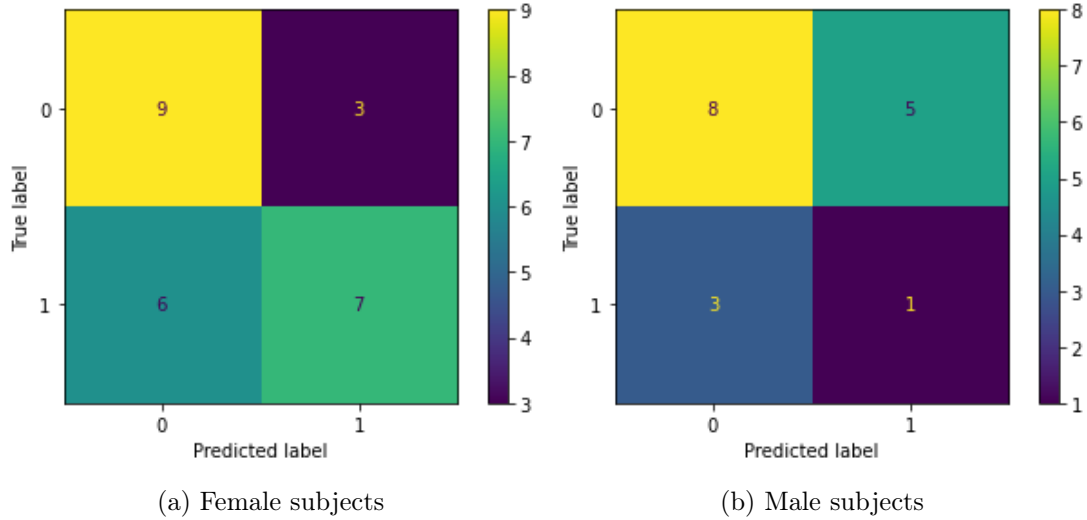


Figure 6.11.: Confusion matrices comparing the prediction results for the test sets to the true class labels.

generalize to unseen data at all. This could simply be due to the small number of subjects (31) in the training set, which might not be enough to extract statistically significant results. For female subjects, the results were more stable; but still an F1 score of only 0.61 on unseen data is not enough to rely on this classifier to inform treatment decisions.

One important finding from these experiments is that splitting the data set by biological sex gave better results than training classifiers on the whole data set. This is in line with previous findings, which indicate that not only are there significant differences between the EEG recordings of male and female subjects (e.g. [58]), but also that biomarkers for treatment prediction might be different between men and women (e.g. [14]).

It is possible that subdividing the data set further could improve the results. For example, the large number of different treatment approaches that were used on the subjects in this data set might make it hard for the classifiers to find conclusive patterns that are common for all treatment approaches. Similar reflections can be made about different age groups, which have also sometimes been pointed out as confounding factors [11]. However, introducing further subdivisions would not only significantly reduce the available amount of training data, but also defeat the research goal of finding a predictor for treatment success that is as generic as possible, so this expedient should be used sparingly.

### 6.3.1. Limitations of the Work

The work presented here is a first exploratory analysis of a complex problem, which depends on a large number of hyperparameters along all steps of the experimental process. As a consequence, only a subset of all possible scenarios could be tried out within the scope of this thesis. The choice of settings for this first exploration was informed by previous results in the literature or preliminary experiments conducted as part of this thesis, but still a wide array of interesting possibilities remain open for further analysis.

One important issue in the experimental setup is the definition of treatment response. In this thesis, "response" indicates a relative improvement compared to the subject's state before treatment; however, this is not the only valid possibility to divide subjects into responders and nonresponders. It is worth investigating whether the information in the Granger causality networks is better suited to pick up on an alternative labelling system.

During the preprocessing of the EEG signals, only two frequency bands were analyzed in addition to the unfiltered data. While previous results (cf. Chapter 2) indicated that the alpha

and theta bands are a good choice, it is certainly possible that necessary information for the classifiers might be contained in different frequency bands than the ones examined so far, or even in relationships across frequency bands.

The focus on electrode channels in the left brain hemisphere was similarly motivated by results in the literature, but this is only one of multiple interesting brain regions. For example, many studies achieved good results when focusing on the frontal region of the brain (cf. Chapter 2). Furthermore, the feature spaces created by the choice of electrodes in this thesis might still be too high-dimensional for the classifiers to reliably separate responders and nonresponders. Even when considering only 8 electrodes, the feature vectors still have  $8 * 8 = 64$  dimensions. PCA was used to reduce the dimensionality of the training data to a certain extent, but so far the focus was on retaining most of the variance in the data (at least 80%). It might be worth investigating the consequences of a more aggressive dimensionality reduction approach, where only a few PCA components are kept. Another possibility of dealing with the high dimensionality could be to introduce more sparsity into the data by only keeping the strongest Granger-causal connections (i.e. only the entries with the largest values in the  $\beta$ -matrices), similar to the 2-Levels-Thresholding approach proposed by Pereverzyev and Hlaváčková-Schindler [59].

In this thesis, only the first 16 seconds of each EEG recording were used to compute the Granger-causal networks, in order to limit the confounding influence of nonstationarity over time. However, there is arguably no guarantee that the necessary information to distinguish responders from nonresponders is always present in these first 16 seconds. Predictions could potentially be improved by summarizing data from a larger recording duration, e.g. considering one epoch every minute. At the same time, such an approach would have to find a possibility to depict the changing Granger-causal connections over time; simply taking the average of the results for multiple epochs would no longer be sufficient in such a scenario.

### 6.3.2. Contributions of the Work

To the best of my knowledge, the work presented in this thesis is the first to apply a combination of (Granger-)causal discovery tools and classification methods to the problem of predicting depression treatment outcome. The exploration presented in this thesis is an important milestone towards understanding how the ITGGM method can be applied to such a problem and which improvements might lead to a more reliable classifier.

Generally speaking, the size of the available data base can play an important role in the reliability and generalizability of classification results (cf. [60]). Unfortunately, many of the studies that focus on classifying depression had to make do with very small data sets. For example, the majority of the studies reviewed in [12] use data sets that consist of fewer than 30 depression patients and 30 healthy controls; in some studies, data sets contain as few as 10 depressed subjects and 10 controls. With its detailed analysis of a data set of 176 subjects, this thesis adds to the small number of studies that investigate the performance of classification methods on a more extensive data basis of depression patients.

Furthermore, the number of different treatment approaches that were used in this data set is much larger than that in many other studies focusing on treatment prediction.

Another important contribution of this work is the analysis of the final classifiers on a separate test set. In the literature, there are many studies which use cross-validation schemes to select the best classification method for similar problems, but do not verify their results on separate test data (e.g. [13, 18, 20, 27, 28]). Furthermore, many of these studies use data sets that are comparable in size to the training set of only male subjects in this thesis, where the best classifier achieved a very good result during cross-validation on the training data, but did not generalize at all to the test data. The results of this thesis therefore seem to strongly support observations made in e.g. [11], which point out that sample sizes in many studies that aim at classifying depression from EEG data might be too small for statistical significance.



## 7. Conclusion

In this thesis, the *Graphical Granger Causality by Information-Theoretic Criteria* (ITGGM) method [7] was applied to EEG data recorded from depression patients, with the aim of predicting subjects' treatment response. This is one of several prediction methods that are investigated as part of the research project "Learning Synchronization Patterns in Multivariate Neural Signals for Prediction of Response to Antidepressants" [3, 4].

A Python implementation of the ITGGM method was created and made publicly available at Github [8]. It follows the API convention defined by the *scikit-learn* library, making it easy to use for both novice and experienced users. Additionally, it was designed in a way that allows it to be easily extended with additional information-theoretic criteria or optimization methods. This implementation will hopefully be a valuable aid in further exploration of the potentials of this method.

In this thesis, Granger-causal networks were computed in the alpha and theta frequency bands in the EEG data, as well as the unfiltered data containing all frequencies. Measurements from all 19 available electrodes were used, in addition to measurements only on the left brain hemisphere (with or without center electrodes). In a first set of experiments, no satisfactory classification results could be achieved, as the classifiers were not able to outperform baseline results created by dummy classifiers. Splitting the data set by biological sex allowed the classifiers to beat the baseline models during experiments on the training set, but the results did not generalize well to a test set of completely unseen data. The fact that considering male and female subjects separately improves the classification performance is however an important finding, which might also inform future experiments in the frame of the Synchronization project.

### 7.1. Future Work

At this point, the newly created Python implementation of the ITGGM method only provides one information-theoretic criterion, the stochastic complexity. Extending it with additional criteria will make it suitable for a wider range of use cases.

With regards to the prediction of treatment response in the examined data set, there are still many options left to explore that did not fit in the scope of this thesis. So far, only two frequency bands and the unfiltered data were investigated, and only same-frequency connections were computed. Analyzing Granger-causal connections in the remaining frequency bands or searching for cross-frequency connections could lead to valuable insights. Similarly, focusing on different brain regions (e.g. using only data from the frontal areas of the brain) might reveal new information. In order to better capture relevant information in the data, approaches to reduce the dimensionality of the feature vectors should be explored, as well as ways to model the (possibly changing) Granger-causal structure in the data over larger periods of time. Finally, it is also possible that the EEG recordings are better suited for discovering remission than response. Training classifiers on the input data using a label set that indicates each subject's remission status will therefore also be an important aspect to focus on in future work.



# Bibliography

- [1] World Health Organization, “Depression.” <https://www.who.int/news-room/fact-sheets/detail/depression>, 2021.
- [2] A. M. Hunter, I. A. Cook, and A. F. Leuchter, “The promise of the quantitative electroencephalogram as a predictor of antidepressant treatment outcomes in major depressive disorder,” *The Psychiatric Clinics of North America*, vol. 30, no. 1, pp. 105–124, 2007.
- [3] University of Vienna, “Learning synchronization patterns in multivariate neural signals for prediction of response to antidepressants.” <https://dm.cs.univie.ac.at/research/projects/project/347/>, 2021.
- [4] National Institute of Mental Health of the Czech Republic (NIMH), “Learning synchronization patterns in multivariate neural signals for prediction of response to antidepressants (21-14727k).” <https://www.nudz.cz/en/research/clinical-research-program/grants-and-projects/struktury-synchronizace-v-mnohorozmernych-neuralnich-signalech-strojove-uceni-a-predikce>, 2022.
- [5] S. Sanei, *EEG signal processing*. Chichester, England: John Wiley & Sons, 2007.
- [6] C. W. J. Granger, “Investigating causal relations by econometric models and cross-spectral methods,” *Econometrica*, vol. 37, no. 3, pp. 424–438, 1969.
- [7] K. Hlaváčková-Schindler and C. Plant, “Graphical Granger causality by information-theoretic criteria,” in *24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain* (G. Giacomo, ed.), pp. 1459–1466, Amsterdam: IOS Press, 2020.
- [8] C. Pacher, “ITGGM-Python.” <https://github.com/ChristinaPacher/ITGGM-Python>, 2023.
- [9] G. Fiscon, E. Weitschek, A. Cialini, G. Felici, P. Bertolazzi, S. De Salvo, A. Bramanti, P. Bramanti, and M. C. De Cola, “Combining EEG signal processing with supervised methods for Alzheimer’s patients classification,” *BMC Medical Informatics and Decision Making*, vol. 18, no. 1, pp. 35–35, 2018.
- [10] R. Catherine Joy, S. Thomas George, A. Albert Rajan, and M. Subathra, “Detection of ADHD from EEG signals using different entropy measures and ANN,” *Clinical EEG and Neuroscience*, vol. 53, no. 1, pp. 12–23, 2022.
- [11] F. S. de Aguiar Neto and J. L. G. Rosa, “Depression biomarkers using non-invasive EEG: A review,” *Neuroscience and Biobehavioral Reviews*, vol. 105, pp. 83–93, 2019.
- [12] A. Khosla, P. Khandnor, and T. Chand, “Automated diagnosis of depression from EEG signals using traditional and deep learning approaches: A comparative analysis,” *Biocybernetics and Biomedical Engineering*, vol. 42, no. 1, pp. 108–142, 2022.
- [13] M. Čukić, M. Stokić, S. Simić, and D. Pokrajac, “The successful discrimination of depression from EEG could be attributed to proper feature extraction and not to a particular classification method,” *Cognitive Neurodynamics*, vol. 14, no. 4, pp. 443–455, 2020.

- [14] M. Arns, “First EEG results of the iSPOT study in depression: EEG alpha asymmetry as a gender specific predictor of SSRI treatment outcome,” *Brain Stimulation*, vol. 8, no. 2, pp. 337–337, 2015.
- [15] P. C. Koo, J. Thome, C. Berger, P. Foley, and J. Hoeppner, “Current source density analysis of resting state EEG in depression: A review,” *Journal of Neural Transmission*, vol. 124, no. Suppl 1, pp. 109–118, 2017.
- [16] L. Zhao, L. Yang, B. Li, Z. Su, and C. Liu, “Frontal alpha EEG asymmetry variation of depression patients assessed by entropy measures and Lemple–Ziv complexity,” *Journal of Medical and Biological Engineering*, vol. 41, no. 2, pp. 146–154, 2021.
- [17] M. Arns, A. Etkin, U. Hegerl, L. M. Williams, C. DeBattista, D. M. Palmer, P. B. Fitzgerald, A. Harris, R. deBeuss, and E. Gordon, “Frontal and rostral anterior cingulate (rACC) theta EEG in depression: Implications for treatment outcome?,” *European Neuropsychopharmacology*, vol. 25, no. 8, pp. 1190–1200, 2015.
- [18] H. Cai, J. Han, Y. Chen, X. Sha, Z. Wang, B. Hu, J. Yang, L. Feng, Z. Ding, Y. Chen, and J. Gutknecht, “A pervasive approach to EEG-based depression detection,” *Complexity (New York, N.Y.)*, vol. 2018, pp. 1–13, 2018.
- [19] J. L. Stewart, J. A. Coan, D. N. Towers, and J. J. B. Allen, “Resting and task-elicited prefrontal EEG alpha asymmetry in depression: Support for the capability model,” *Psychophysiology*, vol. 51, no. 5, pp. 446–455, 2014.
- [20] J. Shen, X. Zhang, X. Huang, M. Wu, J. Gao, D. Lu, Z. Ding, and B. Hu, “An optimal channel selection for EEG-based depression detection via kernel-target alignment,” *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 7, pp. 2545–2556, 2021.
- [21] G. G. Knyazev, A. N. Savostyanov, A. V. Bocharov, S. S. Tamozhnikov, and A. E. Saprigyn, “Task-positive and task-negative networks and their relation to depression: EEG beamformer analysis,” *Behavioural Brain Research*, vol. 306, pp. 160–169, 2016.
- [22] S. Hashempour, R. Boostani, M. Mohammadi, and S. Sanei, “Continuous scoring of depression from EEG signals via a hybrid of convolutional neural networks,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 30, pp. 176–183, 2022.
- [23] A. T. Beck, R. A. Steer, G. K. Brown, *et al.*, “Manual for the Beck depression inventory-II,” 1996.
- [24] M. Hein, J.-P. Lanquart, G. Loas, P. Hubain, and P. Linkowski, “Alterations of neural network organisation during rapid eye movement sleep and slow-wave sleep in major depression: Implications for diagnosis, classification, and treatment,” *Psychiatry Research. Neuroimaging*, vol. 291, pp. 71–78, 2019.
- [25] N. Bailey, K. Hoy, N. Rogasch, R. Thomson, S. McQueen, D. Elliot, C. Sullivan, B. Fulcher, Z. Daskalakis, and P. Fitzgerald, “Differentiating responders and non-responders to rTMS treatment for depression after one week using resting EEG connectivity measures,” *Journal of Affective Disorders*, vol. 242, pp. 68–79, 2019.
- [26] C. Mulert, G. Juckel, M. Brunnmeier, S. Karch, G. Leicht, R. Mergl, H.-J. Möller, U. Hegerl, and O. Pogarell, “Prediction of treatment response in major depression: Integration of concepts,” *Journal of Affective Disorders*, vol. 98, no. 3, pp. 215–225, 2007.
- [27] E. Avots, K. Jermakovs, M. Bachmann, L. Päeske, C. Ozcinar, and G. Anbarjafari, “Ensemble approach for detection of depression using EEG features,” *Entropy (Basel, Switzerland)*, vol. 24, no. 2, p. 211, 2022.

- [28] R. Cîrstian, J. Pilmeyer, A. Bernas, J. F. A. Jansen, M. Breeuwer, A. P. Aldenkamp, and S. Zinger, “Objective biomarkers of depression: A study of Granger causality and wavelet coherence in resting-state fMRI,” *Journal of Neuroimaging*, 2023.
- [29] J. Hamilton, G. Chen, M. Thomason, M. Schwartz, and I. Gotlib, “Investigating neural primacy in major depressive disorder: Multivariate Granger causality analysis of resting-state fMRI time-series data,” *Molecular Psychiatry*, vol. 16, no. 7, pp. 763–772, 2011.
- [30] Q. Lu, K. Bi, C. Liu, G. Luo, H. Tang, and Z. Yao, “Predicting depression based on dynamic regional connectivity: A windowed Granger causality analysis of MEG recordings,” *Brain Research*, vol. 1535, pp. 52–60, 2013.
- [31] N. Mai, Y. Wu, X. Zhong, B. Chen, M. Zhang, Q. Peng, and Y. Ning, “Different modular organization between early onset and late onset depression: A study base on Granger causality analysis,” *Frontiers in Aging Neuroscience*, vol. 13, pp. 625175–625175, 2021.
- [32] J. Zhang, X. Zhang, G. Chen, L. Huang, and Y. Sun, “EEG emotion recognition based on cross-frequency Granger causality feature extraction and fusion in the left and right hemispheres,” *Frontiers in Neuroscience*, vol. 16, pp. 974673–974673, 2022.
- [33] Y. Gao, X. Wang, T. Potter, J. Zhang, and Y. Zhang, “Single-trial EEG emotion recognition using Granger causality/transfer entropy analysis,” *Journal of Neuroscience Methods*, vol. 346, pp. 108904–108904, 2020.
- [34] J. Zhang, X. Zhang, G. Chen, and Q. Zhao, “Granger-causality-based multi-frequency band EEG graph feature extraction and fusion for emotion recognition,” *Brain Sciences*, vol. 12, no. 12, p. 1649, 2022.
- [35] W. Mao, L. Wei, Y. Tang, and Y. Li, “P27-3 the Granger causality changes in the EEG with depression in response to different facial expressions,” *Clinical Neurophysiology*, vol. 121, pp. S264–S264, 2010.
- [36] A. Wismüller, A. M. Dsouza, M. A. Vosoughi, and A. Abidin, “Large-scale nonlinear Granger causality for inferring directed dependence from short multivariate time-series data,” *Scientific Reports*, vol. 11, no. 1, pp. 7817–7817, 2021.
- [37] I. Rodríguez-Rodríguez, A. Ortiz, M. A. Formoso, N. J. Gallego-Molina, and J. L. Luque, “Inter-channel Granger causality for estimating EEG phase connectivity patterns in dyslexia,” in *Artificial Intelligence in Neuroscience: Affective Analysis and Health Applications*, Lecture Notes in Computer Science, pp. 53–62, Springer International Publishing, 2022.
- [38] H. Varsehi and S. M. P. Firoozabadi, “An EEG channel selection method for motor imagery based brain–computer interface and neurofeedback using Granger causality,” *Neural Networks*, vol. 133, pp. 193–206, 2021.
- [39] C. Yi, Y. Qiu, W. Chen, C. Chen, Y. Wang, P. Li, L. Yang, X. Zhang, L. Jiang, D. Yao, F. Li, and P. Xu, “Constructing time-varying directed EEG network by multivariate nonparametric dynamical Granger causality,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 30, pp. 1412–1421, 2022.
- [40] N. Nicolaou, S. Hourris, P. Alexandrou, and J. Georgiou, “EEG-based automatic classification of ‘awake’ versus ‘anesthetized’ state in general anesthesia using Granger causality,” *PloS one*, vol. 7, no. 3, pp. e33869–e33869, 2012.
- [41] W. Hesse, E. Möller, M. Arnold, and B. Schack, “The use of time-variant EEG Granger causality for inspecting directed interdependencies of neural assemblies,” *Journal of Neuroscience Methods*, vol. 124, no. 1, pp. 27–44, 2003.

- [42] A. Shojaie and E. B. Fox, “Granger causality: A review and recent advances,” *Annual Review of Statistics and its Application*, vol. 9, no. 1, 2022.
- [43] F. Diebold, *Elements of Forecasting*. Thomson South-Western, 2007.
- [44] G. E. P. Box and D. R. Cox, “An analysis of transformations,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 26, no. 2, pp. 211–252, 1964.
- [45] H. Jasper, “The 10-20 electrode system of the International Federation,” *Electroencephalogr. Clin. Neuropsychiol.*, vol. 10, pp. 370–375, 1958.
- [46] D. Yao, Y. Qin, S. Hu, L. Dong, M. L. Bringas Vega, and P. A. Valdés Sosa, “Which reference should we use for EEG and ERP practice?,” *Brain Topography*, vol. 32, no. 4, pp. 530–549, 2019.
- [47] A. Delorme and S. Makeig, “EEGLAB: An open source toolbox for analysis of single-trial EEG dynamics including independent component analysis,” *Journal of Neuroscience Methods*, vol. 134, no. 1, pp. 9–21, 2004.
- [48] S. A. Montgomery and M. Åsberg, “A new depression scale designed to be sensitive to change,” *The British Journal of Psychiatry*, vol. 134, no. 4, p. 382–389, 1979.
- [49] A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, and M. S. Hämäläinen, “MEG and EEG data analysis with MNE-Python,” *Frontiers in Neuroscience*, vol. 7, no. 267, pp. 1–13, 2013.
- [50] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, Í. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, “SciPy 1.0: Fundamental algorithms for scientific computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.
- [51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [52] The pip developers, “Pip documentation.” <https://pip.pypa.io/en/stable/>. Accessed: 2023-04-03.
- [53] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, “API design for machine learning software: Experiences from the scikit-learn project,” in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.
- [54] K. Deb, “An introduction to genetic algorithms,” *Sadhana (Bangalore)*, vol. 24, no. 4-5, pp. 293–315, 1999.
- [55] D. Marinazzo, “Kernel Granger causality.” <https://github.com/danielemarinazzo/KernelGrangerCausality>, 2022.
- [56] V. Liew, “Which lag selection criteria should we employ?,” *Economics Bulletin*, vol. 3, pp. 1–9, 01 2004.

- [57] M. J. A. M. van Putten, S. Olbrich, and M. Arns, “Predicting sex from brain rhythms with deep learning,” *Scientific Reports*, vol. 8, no. 1, pp. 3069–7, 2018.
- [58] B. Bučková, M. Brunovský, M. Bareš, and J. Hlinka, “Predicting sex from EEG: Validity and generalizability of deep-learning-based interpretable classifier,” *Frontiers in Neuroscience*, vol. 14, 2020.
- [59] S. Pereverzyev Jr and K. Hlaváčková-Schindler, “Graphical Lasso Granger method with 2-levels-thresholding for recovering causality networks,” in *IFIP Advances in Information and Communication Technology*, vol. AICT-443 of *System Modeling and Optimization*, pp. 220–229, Springer Berlin Heidelberg, 2014.
- [60] A. Althnian, D. AlSaeed, H. Al-Baity, A. Samha, A. B. Dris, N. Alzakari, A. Abou Elwafa, and H. Kurdi, “Impact of dataset size on classification performance: An empirical evaluation in the medical domain,” *Applied Sciences*, vol. 11, no. 2, p. 796, 2021.



# A. Appendix

## A.1. Depression Treatment Methods

The following treatment methods were used on the subjects:

- selective serotonin reuptake inhibitors
- serotonin and norepinephrine reuptake inhibitors
- norepinephrine and dopamine reuptake inhibitors
- repetitive transcranial magnetic stimulation
- transcranial direct current stimulation
- noradrenergic and specific serotonergic antidepressants
- tricyclic antidepressants

## A.2. Test Subjects

The data set used in this thesis contains EEG recordings from 176 subjects, ranging from age 18 to 65. Among them are 84 responders and 92 nonresponders. 48 of the subjects are male and 128 female.

The data set was split into a training set of 134 subjects and a test set of 42 subjects. The distributions of biological sex and responder status in each of these sets are shown in Tables A.1 and A.2.

|                      | male | female | total |
|----------------------|------|--------|-------|
| <b>responders</b>    | 14   | 53     | 67    |
| <b>nonresponders</b> | 17   | 50     | 67    |
| <b>total</b>         | 31   | 103    | 134   |

Table A.1.: Subject distribution in the training set.

|                      | male | female | total |
|----------------------|------|--------|-------|
| <b>responders</b>    | 4    | 13     | 17    |
| <b>nonresponders</b> | 13   | 12     | 25    |
| <b>total</b>         | 17   | 25     | 42    |

Table A.2.: Subject distribution in the test set.

## A.3. Hyperparameters for the Classifiers

Support vector classifier (SVC) with linear kernel

- C (regularization parameter): [0.1, 1, 10, 50, 100]

## A. Appendix

SVC with radial basis function (RBF) kernel

- C (regularization parameter): [0.1, 1, 10, 50, 100]
- gamma: ['scale', 'auto']

SVC with polynomial kernel

- C (regularization parameter): [0.1, 1, 10, 50, 100]
- degree: [2, 3, 4, 5]
- gamma: ['scale', 'auto']

K nearest neighbors

- n\_neighbors: [2, 3, 4, 5, 6, 7]
- weights: ['uniform', 'distance']
- p (power parameter for the Minkowski metric): [1, 2]

Decision tree

- criterion: ['gini', 'log\_loss']
- splitter: ['best', 'random']
- max\_depth: [3, 5, 7, 10, None]

Random forest

- n\_estimators: [5, 10, 20, 50]
- criterion: ['gini', 'log\_loss']
- max\_depth: [3, 5, 7, 10, None]

Naive bayes

- var\_smoothing: [1e-5, 1e-7, 1e-9]