DISSERTATIONES
INFORMATICAE
UNIVERSITATIS
TARTUENSIS
43

ANTI INGEL

Algorithms using information theory: classification in brain-computer interfaces and characterising reinforcement-learning agents

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

43

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS
43

ANTI INGEL

Algorithms using information theory: classification in brain-computer interfaces and characterising reinforcement-learning agents

UNIVERSITY OF TARTU
Press
1632

Institute of Computer Science, Faculty of Science and Technology, University of Tartu, Estonia.

Dissertation has been accepted for the commencement of the degree of Doctor of Philosophy (PhD) in Computer Science on August 24, 2023 by the Council of the Institute of Computer Science, University of Tartu.

Supervisors

Prof.	Raul Vicente Institute of Computer Science University of Tartu Tartu, Estonia
Assoc. Prof.	Dirk Oliver Theis Institute of Computer Science University of Tartu Tartu, Estonia

Opponents

Prof.	Mikhail Prokopenko Centre for Complex Systems University of Sydney Australia
Assoc. Prof.	Luigi Bianchi Department of Civil Engineering and Computer Science Engineering Tor Vergata University of Rome Italy

The public defense will take place on the 22nd of September 2023 at 14:15 in Narva mnt 18-1021.

The publication of this dissertation was financed by the Institute of Computer Science, University of Tartu.

ISSN 2613-5906 (print) ISSN 2806-2345 (PDF)
ISBN 978-9916-27-318-0 (print) ISBN 978-9916-27-319-7 (PDF)

University of Tartu Press
`http://www.tyk.ee/`

ABSTRACT

This thesis introduces three algorithms that tackle problems in different machine learning frameworks. The problem of optimal classification in brain-computer interfaces (BCIs) and the problem of characterising the behaviour of reinforcement learning (RL) agents are considered. A BCI is a direct communication channel between the user's brain and an external device, for example a computer or an electronic wheelchair. Thus, a well-working BCI would allow a user to control devices using only brain signals measured with electrodes placed on the head. RL is a type of machine learning in which an agent learns by getting feedback from an environment.

The introduced algorithms make extensive use of information theory. In the case of BCIs, information theory is used to measure the amount of information the BCI can send in a unit of time, acting as an abstract performance measure. In the case of RL agents, information theory is used to quantify the agent's degree of autonomy, the agent's internalisation of the environment, and the agent's reliance on environment and memory.

This thesis's BCI-related contributions tackled the question whether it is possible to find an optimal classification rule for BCIs and under which conditions the rule is optimal. Two algorithms are introduced, one for finding the optimal classification rule from a fixed set of threshold-based classification rules and another for finding an optimal classification rule from a much more diverse class of rules. The first algorithm is based on multivariable function optimisation, while the second relies on tools of information theory. The introduced algorithms work well in practice, outperforming standard machine learning algorithms and other classification rules often used in the literature on BCIs.

The RL problem considered in this thesis tackle question related to measuring the autonomy or other characteristics of RL agents. This contribution relies on already existing information-theoretic definitions of autonomy and agent's internalisation of the environment, and on an already existing method called partial information decomposition (PID). Here, an algorithm is introduced to calculate the information-theoretic measures in the limiting process of time step approaching infinity. As an additional technique, merging the states in the underlying Markov chain is introduced. This method helps to define more high-level states to apply the algorithm in more situations. The introduced algorithm is applied to a simple partially observable Markov decision process (POMDP). The obtained results coincide with our intuitive understanding of autonomy and internalisation.

CONTENTS

List of original publications **9**

1. Introduction **10**
1.1. Brain-computer interfaces 11
1.2. Reinforcement learning 13
1.3. Notations, basic definitions and properties 15
1.3.1. Probability theory 15
1.3.2. Information theory 16
1.3.3. Markov chains 18
1.3.4. Time average of Markov chain and ergodicity 20
1.4. Information bottleneck 21
1.4.1. Problem statement 21
1.4.2. Optimisation method 22
1.4.3. Alternative methods 23
1.5. Partial information decomposition 23
1.5.1. Problem statement 24
1.5.2. Solution method 24
1.5.3. Alternative methods 25

2. Classification in brain-computer interfaces **26**
2.1. Background 26
2.1.1. Feature extraction methods 26
2.1.2. Classification methods 27
2.1.3. Information transfer rate 27
2.1.4. Constructing training dataset from EEG data 28
2.2. The first contribution: optimal classification with a specific rule . . 29
2.2.1. Classification rule 29
2.2.2. Modelling features as random variables 30
2.2.3. Calculating mean detection time 31
2.2.4. Calculating information transfer rate 32
2.2.5. Calculating gradient and Hessian of information transfer rate 32
2.2.6. Optimising information transfer rate 33
2.2.7. Summary 33
2.3. Empirical verification and comparison of the first contribution . . 35
2.3.1. Dataset 1 35
2.3.2. Signal processing and feature extraction 35
2.3.3. Results and comparison 36
2.3.4. Statistical analysis 37
2.4. The second contribution: optimal classification rule 38
2.4.1. Information bottleneck 38
2.4.2. Discretising features 40

2.4.3. Classification 40
2.4.4. Summary 42
2.5. Empirical verification and comparison of the second contribution . 42
2.5.1. Dataset 2 43
2.5.2. Signal processing and feature extraction 43
2.5.3. Calculating the number of bins 44
2.5.4. Results and comparison 44
2.5.5. Statistical analysis 47

3. Characterising reinforcement-learning agents 48
3.1. Background 48
3.1.1. Information-theoretic framework 48
3.1.2. Definition of autonomy and NTIC 49
3.1.3. Decomposing autonomy measures and NTIC 50
3.2. The third contribution: calculating autonomy, NTIC, and PID terms 51
3.2.1. Stationary distribution of the Markov Chain 51
3.2.2. Calculating autonomy, NTIC, and PID terms 52
3.2.3. Merging the agent's states 53
3.2.4. Summary 54
3.3. Empirical verification 54
3.3.1. Environment 54
3.3.2. Agent 55
3.3.3. Perturbations 55
3.3.4. Results 56

4. Conclusion 60
4.1. Characterising reinforcement-learning agents 60
4.2. Brain-computer interfaces 61

Bibliography 62

Acknowledgements 70

Sisukokkuvõte (Summary in Estonian) 71

Publications 75
Direct information transfer rate optimisation for SSVEP-based BCI . . . 78
Information Bottleneck as Optimisation Method for SSVEP-Based BCI . 94
Quantifying Reinforcement-Learning Agent's Autonomy, Reliance on Memory and Internalisation of the Environment 110

Curriculum Vitae 136

Elulookirjeldus (Curriculum Vitae in Estonian) 137

LIST OF ABBREVIATIONS

BCI brain-computer interface. 5, 10–13, 15–18, 22, 23, 26–28, 35, 36, 42, 43, 60, 61

CCA canonical correlation analysis. 26, 30, 36, 43, 44

CDF cumulative distribution function. 16, 30–32, 40

CI synergistic information. 24, 50, 51, 56

DIB deterministic information bottleneck. 22, 39, 41, 44

EEG electroencephalography. 12, 26, 28, 29, 33, 35, 36, 42, 43

GIB generalised information bottleneck. 10, 11, 22, 60

ITR information transfer rate. 27–30, 32–34, 36–39, 42, 45–47, 61

LDA linear discriminant analysis. 27, 30, 36

LSTM long short-term memory. 55, 56

MDP Markov decision process. 13, 49

MDT mean detection time. 28, 29, 31, 32, 36–38, 42, 45–47, 61

MLE maximum likelihood estimation. 31

MM method of moments. 21, 31

NN neural network. 13, 23, 27, 55

NTIC non-trivial informational closure. 48, 50, 51, 54, 56, 60

PDF probability density function. 16, 30–33

PID partial information decomposition. 5, 10, 11, 14, 23–25, 48, 50–52, 54–58, 60

POMDP partially observable Markov decision process. 5, 13

PSDA power spectral density analysis. 26, 30, 36

RL reinforcement learning. 5, 10, 11, 13–15, 18, 25, 48, 49, 52, 54, 60

SI shared information. 24, 50, 51, 56

SSVEP steady-state visual evoked potential. 11–13, 27, 35, 36, 43

TRCA task-related component analysis. 13, 26, 43, 44

UI unique information. 24, 50–52, 56, 58

LIST OF ORIGINAL PUBLICATIONS

Publications included in the thesis

[I] A. Ingel, I. Kuzovkin, and R. Vicente. Direct information transfer rate optimisation for SSVEP-based BCI. *Journal of Neural Engineering*, 16(1), 2018. DOI: 10.1088/1741-2552/aae8c7.

[II] A. Ingel and R. Vicente. Information Bottleneck as Optimisation Method for SSVEP-Based BCI. *Frontiers in Human Neuroscience*, 15, 2021. DOI: 10.3389/fnhum.2021.675091.

[III] A. Ingel, A. Makkeh, O. Corcoll, and R. Vicente. Quantifying Reinforcement-Learning Agent's Autonomy, Reliance on Memory and Internalisation of the Environment. *Entropy*, 24(3), 2022. DOI: 10.3390/e24030401.

Other published work of the author

[IV] A. Ingel, N. Shahroudi, M. Kängsepp, A. Tättar, V. Komisarenko, and M. Kull. Correlated daily time series and forecasting in the M4 competition. *International Journal of Forecasting*, 36(1):121–128, 2020. DOI: 10.1016/j.ijforecast.2019.02.018. M4 Competition.

Author's contribution

I Anti Ingel conceptualised the idea, performed the formal analysis, implemented the algorithm, performed the experiments, wrote the original draft, made the visualisations, improved the draft during peer review, and responded to the reviewers' questions.

II The article contains Author contributions section. Anti Ingel conceptualised the idea, performed the formal analysis, implemented the algorithm, performed the experiments, wrote the original draft, made the visualisations, improved the draft during peer review, and responded to the reviewers' questions.

III The article contains Author contributions section. Anti Ingel participated in the formal analysis (conceptualised the algorithm for calculating the measures and merging the states), implemented the algorithm (for calculating the measures and merging the states), performed some of the experiments (grid environment experiments), made the visualisations, wrote largest part of the original draft, improved the draft during peer review, and responded to the reviewers' questions.

1. INTRODUCTION

This thesis introduces three algorithms that tackle problems in different machine learning frameworks. These algorithms have already been published in research articles, referred to as Publications I–III [1, 2, 3]. The descriptions given in this thesis follow the original publications closely but give some additional theoretical insights. Differences from the original publications in terms of empirical evaluations are the following. The performance of algorithms from Publications I and II is re-evaluated in this thesis using a slightly different method compared to the original publications. Some experiments of Publication III were re-ran with slightly different perturbations in hopes of finding more interesting behaviour of the characterising measures.

This thesis considers the problem of optimal classification in brain-computer interfaces (BCIs) and the problem of characterising the behaviour of reinforcement learning (RL) agents. The introduced algorithms make extensive use of information theory. In the case of BCIs, information theory is used to measure the amount of information the BCI can send in a unit of time, acting as an abstract performance measure. In the case of RL agents, information theory is used to quantify the agent's degree of autonomy, the agent's internalisation of the environment, and the agent's reliance on environment and memory.

The introduced algorithms are described in the following chapters. The descriptions are given following the algorithm step-by-step. Along with the description of each step, there is an explanation why of these steps achieve the goal. Sometimes simplifying assumptions are made due to practical limitations, for example, the amount of available data. Thus, the algorithms are introduced together with a theoretical justification that the performed calculations achieve the set goal under certain assumptions. This approach of introducing algorithms can be rigorously formalised in type theory [4]. However, in this thesis, the algorithms are introduced in a more informal style similar to how a mathematician would write a proof. This presentation results in a readable format that focuses on important parts. The implementations of the introduced algorithms are available in code repositories[1][2][3].

More generally, this thesis tackles the question of how to apply some recently introduced information-theoretic algorithms in some practical applications. New generalised information bottleneck (GIB) [5] (discussed in section 1.4) and partial information decomposition (PID) [6] (discussed in section 1.5) algorithms are applied in the setting of BCIs and RL, respectively. In particular:

- Publication I [1] developed a machine-learning setting for BCIs and introduced an algorithm for finding an optimal classifier in a specific case,

[1] `https://github.com/antiingel/ITR-optimisation`

[2] `https://github.com/antiingel/information-bottleneck-BCI`

[3] `https://github.com/antiingel/RL-agent-autonomy`

considering only certain threshold-based classifiers.

- Publication II [2] makes use of the recently introduced information bottleneck algorithm [5] and solves the problem introduced in Publication I in a more general setting.
- Publication III [3] considers the application of characterising RL agents. Here, recent PID algorithm is used [6]. This method is unique among PID measures as the result is differentiable with respect to the underlying probability distribution [7], possibly allowing further machine learning applications.

Actually, also publications I and II consider differentiable information-theoretic measures. In Publication I, the performance measure is differentiable with respect to underlying probability distributions, which are determined by the thresholds of the classifier. In Publication II, the objective function of GIB optimisation task is differentiable with respect to underlying probability distributions. In both cases, the differentiability is used for solving the optimisation task.

The following sections introduce BCIs and RL, followed by the basics of probability theory, information theory and Markov chains. Briefly, the ergodicity of random processes is mentioned. The remainder of the thesis is structured as follows. The two contributions related to BCI [1, 2] are discussed in Chapter 2, starting with the notions relevant to both of these contributions. The discussion of the main contributions are split to sections 2.2 and 2.4, respectively. The evaluation of the introduced algorithms is given in sections 2.3 and 2.5, respectively. The contribution related to RL [3] is presented in Chapter 3 with a similar structure.

1.1. Brain-computer interfaces

Two of the three contributions in this thesis [1, 2] deal with the classification problem in BCIs. A BCI is a direct communication channel between the user's brain and an external device, for example a computer or electronic wheelchair. A possible use case for a BCI is to help people with severe motor disabilities control devices. However, performance in terms of efficiency is still considered a key obstacle to real-life applications [8, 9].

This thesis focuses on steady-state visual evoked potential (SSVEP)-based BCIs, however, the ideas should also be applicable to other types of BCIs. SSVEP is a brain response that is elicited by visual stimulation using constant and sufficiently high-frequency flickering. Constant stimulation frequency elicits a signal in the brain with the same fundamental frequency but also causes responses with the frequencies of the harmonics. SSVEP-based BCIs have recently received much attention due to their ease of use and good performance [9].

An SSVEP-based BCI uses visual stimuli, called targets, to elicit a response in the user's brain. SSVEP-based BCIs are often implemented with visual stimuli presented on a computer screen to the user while their brain activity is recorded

with an electroencephalography (EEG) device. EEG devices allow non-invasive recording of brain activity with a high temporal resolution. However, these devices lack good spatial resolution, which means that the origins of the signals are unknown [10]. Usually, multiple visual stimuli are presented to the user on the computer screen, each stimulus corresponding to one command that the user can send. When the user focuses their gaze on a stimulus, the command corresponding to it is sent to the device (if the classification is correct).

Thus, an SSVEP-based BCI consists of an interface for presenting the stimuli, an EEG device for recording the brain activity and an algorithm for processing the recorded EEG signal to determine the user's intent. The algorithm for processing the signal can be divided into different parts (see Figure 1): signal processing, feature extraction and classification. This thesis focuses on the classification algorithm. The other parts of the BCI are given less attention in this thesis. Publicly available datasets were used to evaluate the performance of the BCI. Thus no new EEG data was published. Signal processing and feature extraction were dealt with as much as needed to evaluate the performance of the introduced algorithms and compare these to previous results and different classification algorithms.

It is quite widespread in the SSVEP BCI literature to combine feature extraction and classification into a single step [11, 12, 13, 14]. This combination is often achieved by designing features so that the feature values have a simple interpretation. Often, features are designed so that a larger value corresponds to higher confidence for a class. In this setting, a sample can be classified using a trivial classification rule which chooses the class corresponding to the highest feature value. This approach would be sufficient if we could produce the best possible features this way. However, machine learning could reveal better classification rules than the trivial ones that choose the class corresponding to the maximum feature value. Standard machine learning methods have already been used in the literature [15, 16, 17, 18, 19, 20, 21].

A classification algorithm designed specifically for BCIs should be used to

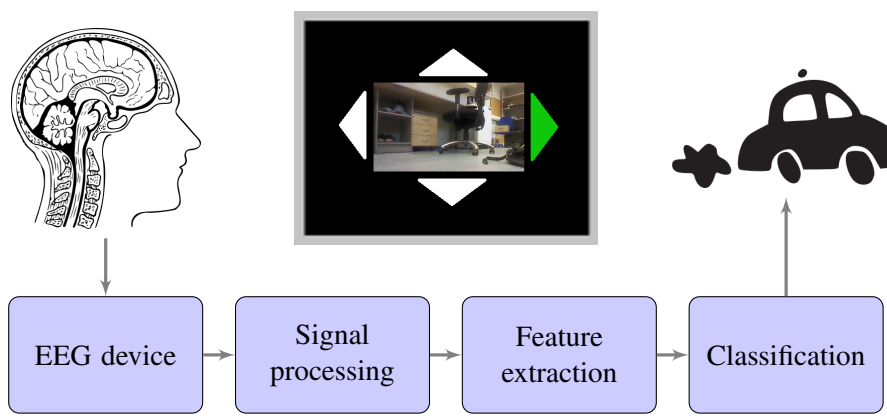


Figure 1: A visualisation of the signal pipeline of a BCI.

achieve the optimal behaviour of a BCI. In particular, in this thesis, a classification rule is considered optimal if it maximises the performance measure of the BCI. The performance measure is taken as the amount of information that the BCI can transfer. Introducing algorithms for finding optimal classifiers is the problem that this thesis tackles. Standard machine learning algorithms are not directly applicable to solve this problem because they aim to optimise different measures. In addition, BCI is a real-time system, and thus the efficiency depends also on the amount of time it takes to make a prediction. Therefore, it might be beneficial to allow the classifier to leave some samples unclassified and wait for new data instead. In this thesis, these specifics of BCIs are taken into account, and new classification algorithms that directly maximise the performance measure of the BCIs are introduced [1, 2].

Before starting with the introduction of RL, let us discuss the popular machine learning method applicable in both fields, neural networks (NNs). Up until a few years ago, the advantages of NN-based classifiers were not so clear in SSVEP-based BCIs [22, 23, 24, 21]. In a study by Podmore et al., a non-NN method is superior in 3 out of 4 settings [22]. In a study by Ravi et al., the competing method task-related component analysis (TRCA) was not utilised to its full potential as it was applied to shifted signals [23]. As shown in Publication II [2], TRCA can be used with shifted signals if it is trained separately for different shifts. Thus it could not be concluded that NN-based models are superior. Several studies make comparisons only in terms of accuracy [24, 25, 26, 27], which is not the main performance measure in BCI literature. Lack of enough training data was often reported as a problem for NNs [24, 21].

Recently, however, more SSVEP BCI datasets have become publicly available [28, 29], and in recent years more impressive results based on NN methods have been reported [30, 27]. Using an NN-based approach was also suggested as future work in Publication II [2]. A possible improvement to be made to the use of NNs in SSVEP-based BCIs is that in addition to handling well the domain-specific input data, they should also focus on the domain-specific performance measure, which is the main theme of this thesis. Although neural network models were not used here, the general idea of maximising the performance measure developed in this thesis could still be beneficial.

1.2. Reinforcement learning

The third contribution of this thesis [3] tackles the problem of characterising the behaviour of RL agents. RL is a type of machine learning in which an agent learns by getting feedback from an environment. A visualisation is given in Figure 2. This setting can, for example, be formalised as a Markov decision process (MDP) or partially observable Markov decision process (POMDP), depending on whether the state of the environment is directly observable to the agent or not. These frameworks assume a Markovian structure for the interactions. RL agents have

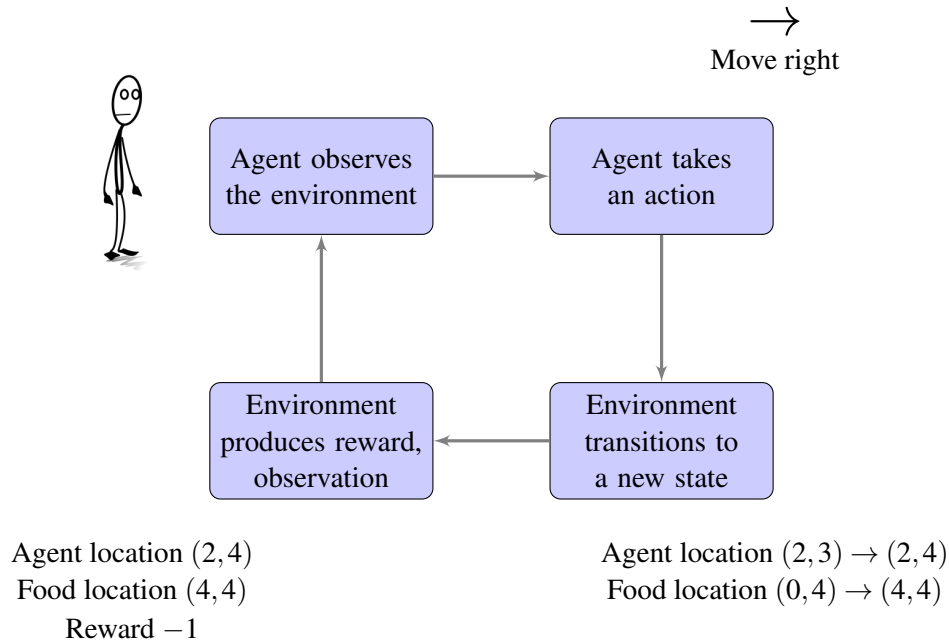


Figure 2: A visualisation of a typical RL pipeline.

achieved good performance in different settings, for example, in board games, video games, and robotics [31, 32, 33, 34].

This contribution uses the measures developed by Bertchinger et al. [35]. They introduced information-theoretic measures for quantifying an agent's autonomy and its internalisation of the environment. However, they did not give an explicit algorithm for calculating these measures. The third contribution introduces an algorithm for calculating these measures for RL agents in practice.

In addition to using the measures by Bertchinger et al., we used PID [36] to decompose these measures. PID is a method for partitioning the mutual information (introduced in section 1.3.2) between random variables into terms which give more insight into how these variables relate to each other. Unfortunately, there is no unique way of finding this partitioning, and different methods exist [37, 6]. PID is further discussed in section 1.5. Using PID also allowed us to quantify how much the agent relies on its memory and how much it relies on the environment or the observations.

Thus, using the measures of Bertchinger et al. and PID, several aspects of the RL agent's behaviour can be quantified under a common information-theoretic setting. Understanding RL agents' behaviour is important to developing flexible and robust agents. Adaptability and autonomy are necessary for dealing with changing real-world conditions [38]. Agents have to be tolerant to perturbations to be adaptable [38], and self-monitoring is a fundamental requirement to achieve perturbation tolerance [39]. Being able to quantify autonomy and perturbation

tolerance is thus essential for more advanced applications.

1.3. Notations, basic definitions and properties

This section introduces the notations used throughout the thesis, gives basic definitions of probability theory, information theory, and Markov chains, and briefly mentions the concept of ergodicity. Probability theory forms the basis of information theory. Thus, both of these are used in each of the contributions of this thesis. Markov chains are used to formulate the framework in which information-theoretic measures are defined in the third contribution. Ergodicity is used to justify the performance-measure estimation method used in the first two contributions.

1.3.1. Probability theory

This section gives an overview of the required concepts of probability theory and introduces notations used throughout the remainder of the thesis. Probability theory is extensively used in this work. It is the foundation on which information theory has been built. Information theory, in turn, has been used to define the most widely used performance measure for BCIs [40] and to define measures characterising the behaviour of RL agents [35]. In addition to probability theory being a basis for information theory, it is needed to introduce theoretically justified algorithms in this work.

A probability space, which allows logically consistent treatment of probabilities, is defined as a triplet consisting of a set, a collection of its measurable subsets, and a probability measure. The precise definitions of these concepts reach back to measure theory and are omitted from this work (see, for example, [41] for these definitions). Throughout this thesis, the probability measure of the underlying probability space is denoted as $\mathbf{P}$, and the corresponding sample space is denoted as Ω. For the third element of probability space, the collection of measurable subsets, no special notation is reserved.

Random variables are denoted by capital letters, possibly with a lower index, for example, X, Y, Z or X_1, X_2, X_3. Measure-theoretically, a random variable is a measurable function. For example, random variable X with real numbers as its values is a function from Ω to $\mathbb{R}$, denoted by $X : \Omega \to \mathbb{R}$. The standard simplified notation is used for the events that are defined using random variables. For example the set of elements of Ω which satisfy $X(\omega) = i$ is denoted by

$$\{X = i\} = \{\omega \in \Omega \mid X(\omega) = i\}.$$

The curly braces in the notation $\{X = i\}$ can be omitted if the set is an argument to $\mathbf{P}$. To further simplify the notation, a lower or upper index is used for this set

$$X^i = X_i = \{X = i\}$$

in the case of equality. When dealing with random processes, then random variables are denoted with lower indices, for example, $X_1, X_2, X_3, \ldots$. Whether X_i

refers to a random variable or an event is clear from the context. If X denotes a random variable, then X_i is an event. If X denotes a random process, then X_i is a random variable.

For a random variable $X : \Omega \to \mathbb{R}$, its cumulative distribution function (CDF) $F_X : \mathbb{R} \to [0,1]$ is defined by

$$F_X(x) = \mathbf{P}(X \leq x),$$

and its complementary CDF is defined by

$$\overline{F}_X(x) = \mathbf{P}(X > x).$$

Note that it is possible to denote a random variable as F_i, thus its CDF would be denoted as F_{F_i}.

The probability density function (PDF) of a random variable $X : \Omega \to \mathbb{R}$ is denoted as $f_X : \mathbb{R} \to [0,\infty]$ (if it exists). CDF and PDF satisfy the property that if f_X is continuous at x, then the derivative of CDF at x, denoted by $\frac{dF_X}{dx}(x)$, is equal to the value of f_X at x, that is

$$f_X(x) = \frac{dF_X}{dx}(x).$$

This property is used in optimising the performance measure for BCI in the first contribution of this thesis [1].

Lastly, the conditional probability of an event A, given an event B, is denoted by $\mathbf{P}(A \mid B)$, and the expectation of a random variable X is denoted by $\mathbf{E}(X)$.

1.3.2. Information theory

With the basic concepts of probability at hand, now basic concepts of information theory [42] can be introduced. In this thesis, notions of information theory are used only with random variables that have a finite number of possible values. Note, however, that many of the notions can be generalised even to continuous random variables [43].

Let X denote a random variable with a finite number of possible values, and let n denote the number of possible values. The values themselves are not important in the context of information theory. What is important is the probabilities for obtaining the values. Therefore, without loss of generality, let the possible values of X be denoted by labels $1,2,3,\ldots,n$. Since X is a random variable, it is defined on some probability space; let $\mathbf{P}$ denote the probability measure of that probability space as discussed in the previous section. Then the (Shannon) entropy of X is defined as

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathbf{P}(X_i) \log(\mathbf{P}(X_i))$$

where $0 \log 0$ is defined to be equal to 0. In this thesis, log denotes logarithm with base 2. Entropy obtains its largest value $\log n$ if X is uniformly distributed, and the smallest value 0 if X is a constant random variable. Intuitively, entropy shows the amount of uncertainty in the random variable X. Information is then interpreted as a reduction in this uncertainty, thus, the maximum amount of information that can be obtained about X is $\mathrm{H}(X)$.

Suppose now that Y is another random variable on the same probability space, taking values in $\{1,2,3,\ldots,m\}$. Note that the random vector (X,Y), mapping $\omega \in \Omega$ to the pair $(X(\omega),Y(\omega))$, is also a random variable and its entropy can thus be calculated. For simplicity, the double parenthesis are omitted, thus

$$\mathrm{H}(X,Y) = \mathrm{H}((X,Y)).$$

Conditional entropy of X given Y can be defined as

$$\mathrm{H}(X \mid Y) = \mathrm{H}(X,Y) - \mathrm{H}(Y)$$

and it satisfies

$$\mathrm{H}(X \mid Y) = -\sum_{j=1}^{m} \mathbf{P}(Y_j) \sum_{i=1}^{n} \mathbf{P}(X_i \mid Y_j) \log \mathbf{P}(X_i \mid Y_j),$$

which means that it is the expected amount of uncertainty of X given that the value of Y is already known. Conditional entropy obtains its largest value $\mathrm{H}(X)$ if X and Y are independent and smallest value 0 if each value of Y determines a value of X. These inequalities $0 \leq \mathrm{H}(X \mid Y) \leq \mathrm{H}(X)$ allow to interpret $\mathrm{H}(X \mid Y)$ as a part of $\mathrm{H}(X)$.

Next, mutual information is defined. Mutual information is closely related to the performance measure of BCIs. Mutual information between random variables X and Y can be defined as

$$\mathrm{I}(X,Y) = \mathrm{H}(X) - \mathrm{H}(X \mid Y). \tag{1.1}$$

This formula is the difference between the total amount of uncertainty in X and the expected amount of uncertainty left in X if Y is known. The result is the reduction in uncertainty or, equivalently, the amount of information obtained about X if Y is known, or vice-versa since

$$\mathrm{I}(X,Y) = \mathrm{I}(Y,X).$$

The largest and smallest value of mutual information correspond to the cases when the conditional entropy $\mathrm{H}(X \mid Y)$ obtains its lowest and largest value, respectively. These cases have already been discussed.

A third random variable is needed to define the conditional version of mutual information. Suppose there is a third random variable Z on the same probability

space with values in $\{1,2,3,\ldots,k\}$. Conditional mutual information between X and Y, given Z, is

$$\mathrm{I}(X,Y \mid Z) = \mathrm{H}(X \mid Z) - \mathrm{H}(X \mid Y,Z).$$

Interpretation is similar to mutual information. Here, conditioning on Z has been added to both terms. The argument's separator for mutual information and conditional mutual information can be denoted with a semi-colon or a colon instead of a comma, for example, $\mathrm{I}(X;Y,Z) = \mathrm{I}(X,(Y,Z))$, to avoid having to write parentheses for random vectors.

Finally, mutual information can be used to define channel capacity, which provides an interpretation for the performance measure of BCIs [44] and has also been used in RL [45]. While mutual information is symmetric, channel capacity differentiates between the arguments, interpreting one as an input and the other as an output. Suppose X denotes the input random variable and Y output random variable. Then the channel is fully characterised by probabilities $\mathbf{P}(Y_j \mid X_i)$. Channel capacity is the supremum of mutual information over the possible distributions of input X

$$\sup_{\mathbf{P}(X_i)} \mathrm{I}(X,Y).$$

A fruitful treatment of channel capacity requires using complex encoding schemes which might not be possible in BCIs since human is on one end of the channel [46].

1.3.3. Markov chains

The framework in which the information-theoretic measures are defined in the third contribution is a Markov chain. Thus, this section introduces the notation, basic definitions, and some properties of Markov chains that are needed later.

A Markov chain is a random process (a sequence of random variables) satisfying the Markov property, meaning that the probability of transferring to the next state depends only on the present state and not on the previous states. Looking at it as an information flow, it can be said that the next state obtains information about previous states only through the present state.

Let us denote the Markov chain as $X_0, X_1, X_2, \ldots$ and its state space as S. This means that $X_n : \Omega \to S$. In this thesis, only finite S is considered. Thus the elements of S can be denoted by integers $1,2,3,\ldots,N$, where N is the number of elements. With this notation, Markov property can be written as

$$\mathbf{P}(X_n = x_n \mid X_{n-1} = x_{n-1}) = \mathbf{P}(X_n = x_n \mid X_{n-1} = x_{n-1}, \ldots, X_0 = x_0).$$

Let us assume further that the Markov chain is homogeneous, that is

$$\mathbf{P}(X_{n+1} = x_{n+1} \mid X_n = x_n) = \mathbf{P}(X_n = x_n \mid X_{n-1} = x_{n-1})$$

for each n. Then the transition probabilities $\mathbf{P}(X_1 = x_1 \mid X_0 = x_0)$ fully determine the dynamics of the Markov chain. Let us organize the transition probabilities into a matrix

$$P = \begin{pmatrix} \mathbf{P}(X_1 = 1 \mid X_0 = 1) & \dots & \mathbf{P}(X_1 = 1 \mid X_0 = N) \\ \vdots & \ddots & \vdots \\ \mathbf{P}(X_1 = n \mid X_0 = 1) & \dots & \mathbf{P}(X_1 = n \mid X_0 = N) \end{pmatrix}.$$

Let us denote the probability distribution at time step n as a vector

$$\pi^n = (\mathbf{P}(X_n = 1), \dots, \mathbf{P}(X_n = N))^T$$

Here subscript T denotes the transpose of the vector. Then π^0 denotes the initial distribution of the Markov chain and the relation

$$\pi^{n+1} = P\pi^n$$

holds. In case the limit $\lim_n \pi^n$ exists, it characterises the asymptotic behaviour of the Markov chain. A vector π satisfying $\pi = P\pi$ and

$$\sum_{i=1}^{N} \pi_i = 1$$

is called a stationary distribution of the Markov chain. If for some initial distribution π^0 the limit $\lim_n \pi^n$ exists, then it is called a limiting distribution for initial distribution π^0 in this thesis. This limiting distribution is also a stationary distribution. However, theoretically more interesting is the case when a unique limit $\lim_n \pi^n$ exists for all initial distributions π^0. In that case, it is called a limiting distribution.

Two important properties of Markov chains that affect the limiting behaviour are periodicity and reducibility. State $i \in S$ is said to be accessible from state $j \in S$ if there exists an integer $n \geq 0$ such that

$$\mathbf{P}(X_n = i \mid X_0 = j) > 0.$$

If state i is accessible from j and state j is accessible from i, then these states are said to communicate. A set of all states that communicate with each other is called a communicating class. Markov chain is called irreducible if all its states communicate. Otherwise it is called reducible. Irreducible Markov chain with a finite state space has a unique stationary distribution [41].

A state $s \in S$ of the Markov chain is said to be periodic if the chain can return to the state only at time steps that are multiple of some integer that is larger than one. Formally, this condition can be written as

$$\gcd\{n \in \{1, 2, 3, \dots\} \mid \mathbf{P}(X_n = s \mid X_0 = s) > 0\} > 1.$$

Markov chain is periodic, if there is a state that is periodic. Otherwise it is called aperiodic. Irreducible and aperiodic Markov chain with a finite state space has a unique limiting distribution [41].

1.3.4. Time average of Markov chain and ergodicity

As mentioned in the previous section, Markov chains with only finite state space are considered in this thesis. For an irreducible Markov chain, its unique stationary distribution is [41]

$$\mu(x) = \lim_n \frac{1}{n} \sum_{i=0}^{n-1} \mathbf{P}(X_i = x).$$

A natural estimate for that is the proportion of time spent in each state in the long run

$$\hat{\mu}_n(x) = \frac{1}{n} \sum_{i=0}^{n-1} I_{\{X_i=x\}}$$

where $I_{\{X_i=x\}}$ denotes the indicator function

$$I_{\{X_i=x\}}(\omega) = \begin{cases} 1, & \omega \in \{X_i = x\} \\ 0, & \omega \notin \{X_i = x\} \end{cases}$$

of the corresponding set. As sample size n increases, $\hat{\mu}_n(x)$ converges to $\mu(x)$ with probability one [41].

If the Markov chain is aperiodic in addition to irreducible, then μ is the limiting distribution for the Markov chain. For a periodic Markov chain, μ is a stationary distribution corresponding to the current initial distribution.

Reducible Markov chains can be handled as follows. The Markov chain can be divided into a finite number of communicating classes. Each communicating class forms an irreducible Markov chain with a unique stationary distribution. These can be calculated separately for each class and combined together by weighting them with the probabilities of ending up in the corresponding communicating classes. This procedure gives a stationary distribution of the whole Markov chain. In an aperiodic case, this is the limiting distribution. In a periodic case, this is a limiting distribution for some fixed initial distribution(s).

The convergence of the time average can be generalised to other random processes. The irreducibility of Markov chains can be generalised to the ergodicity of random processes. Consider a stationary random process, meaning that for all m and k, the vectors $(X_0, \ldots, X_m)$ and $(X_{0+k}, \ldots, X_{m+k})$ have the same distribution. If this process is ergodic and f is a measurable function with $\mathbf{E}|f(X_0)| < \infty$, then with probability one [41]

$$\lim_n \frac{1}{n} \sum_{i=0}^{n-1} f(X_i) = \mathbf{E}(f(X_0)). \tag{1.2}$$

Thus, to estimate the expected value of $f(X_n)$, one can calculate the time average instead. This property gives a natural estimate for the expected value as a finite sum.

An ergodicity-like assumption is used in the first two contributions of this thesis. It is not assumed that ergodicity holds over the whole signal, but the signal is considered ergodic over small enough time windows, making it suitable to estimate statistics from multiple short enough trials [47]. Ergodicity is a useful assumption when considering theoretical questions and has been used in the literature [46, 47].

In fact, in the current work, it is enough to have the ergodicity of the extracted features. Ergodicity over the signal implies that the consecutive features extracted from the signal also form an ergodic sequence [41], which means that this is a weaker assumption. Furthermore, the convergence (1.2) for features is not needed for all functions f but only for three functions: the identity $f(x) = x$, the square $f(x) = x^2$ and the cube $f(x) = x^3$. The reason that these functions are needed is that the method of moments (MM) is used to estimate parameters for a three-parameter distribution. Finally, it is noted that in Publication II [2], an alternative histogram-based estimation method was also used, which would require convergence (1.2) for a set of functions of the form $f(x) = I_{(-\infty,b]}(x)$, which are determined during runtime.

1.4. Information bottleneck

This section gives an overview of the information bottleneck method [48, 5]. The information bottleneck provides a rich framework for discussing problems in signal processing and learning [48], with applications in representation learning or feature extraction in machine learning [49, 50, 51, 52]. The information bottleneck method is used in Publication II [2] for solving an optimisation task. Following subsections give an overview of the problem statement, a specific solution method, and discuss alternatives. Readers not interested in the details can skip this section, as a short discussion on this topic is also given later in section 2.4.1 discussing the second contribution.

1.4.1. Problem statement

The information bottleneck solves an information-theoretic optimisation task in which two random variables X, Y are given, and the goal is to find a third random variable T which is a compressed version of the first variable but still has much information about the second variable. The method aims to find the best trade-off between compressing the representation and preserving meaningful information.

In more detail, the information bottleneck method [48] solves the optimisation task

$$\arg\min[\mathrm{I}(T,X) - \beta\,\mathrm{I}(T,Y)],$$

where minimisation is over conditional distributions of T given X, β is a non-negative parameter, and minimisation is subject to the constraint that the sequence

Y, X, T satisfies Markov property. The parameter β can be used to tune the relative importance of the two terms, which are the compression $\mathrm{I}(T,X)$ and the relevance $\mathrm{I}(T,Y)$.

Thus, the original information bottleneck method outputs the conditional distribution of T given X that minimises the objective function. Relation to classification task in BCIs is more easily seen from the more recent method called deterministic information bottleneck (DIB) [5]. DIB solves similar optimisation problem as the original information bottleneck, the only difference is the first term in the objective function

$$\arg\min[\mathrm{H}(T) - \beta\,\mathrm{I}(T,Y)].$$

This method is deterministic in the sense that it outputs a function from values of X to values of T. In this thesis, the method is applied in such a way that the obtained function is the classification rule for the BCIs that takes features as input and outputs the predicted class.

1.4.2. Optimisation method

The previously mentioned optimisation tasks can be solved as special cases of a more general optimisation task, called GIB [5], by introducing additional parameter α and considering the task

$$\arg\min[\mathrm{H}(T) - \alpha\,\mathrm{H}(T \mid X) - \beta\,\mathrm{I}(T,Y)].$$

From here, DIB optimisation task is solved by taking the limit of the solutions in the process $\alpha \to 0$. The original information bottleneck is obtained by setting $\alpha = 1$. Values between 0 and 1 interpolate between the original information bottleneck and DIB.

A condition that the solution must satisfy can be derived as follows. Assume that the random variables T, X, Y have finite number of possible values. In the following, i, j, k denote some values of these random variables, and summation over these indices denote summations over all the possible values of the corresponding random variable.

Consider the problem a standard constrained optimisation task of the given objective function. The solution is constrained in a sense that the probabilities must sum to one

$$\sum_i \mathbf{P}(T_i \mid X_j) = 1.$$

This constraint can be enforced using Lagrange multipliers by turning the objective function into

$$\mathrm{H}(P) - \alpha\,\mathrm{H}(P \mid X) - \beta\,\mathrm{I}(P,C) - \sum_j \lambda_j \left(\sum_i \mathbf{P}(T_i \mid X_j) - 1 \right).$$

Now one proceeds with taking the derivative with respect to the probabilities $\mathbf{P}(T_i \mid X_j)$ and setting these derivatives equal to zero to obtain [5]

$$\mathbf{P}(T_i \mid X_j) = \frac{\mathbf{P}(T_i)}{Z(x,\alpha,\beta,\lambda_j)} \exp\left(\frac{\log \mathbf{P}(T_i)}{\alpha} - \frac{\beta}{\alpha}\sum_k \mathbf{P}(Y_k \mid X_j) \log \frac{\mathbf{P}(Y_k \mid X_j)}{\mathbf{P}(Y_k \mid T_i)}\right),$$

where $Z(x,\alpha,\beta,\lambda_j)$ denotes a normalisation term. The probabilities $\mathbf{P}(Y_k, X_j)$ are assumed to be known and hence the marginal and conditional probabilities involving only Y and X are also known. The remaining probabilities can be calculated as [5]

$$\mathbf{P}(Y_k \mid T_i) = \sum_j \mathbf{P}(Y_k \mid X_j)\,\mathbf{P}(X_j \mid T_i) = \frac{1}{\mathbf{P}(T_i)} \sum_j \mathbf{P}(Y_k \mid X_j)\,\mathbf{P}(T_i \mid X_j)\,\mathbf{P}(X_j)$$
$$\mathbf{P}(T_i) = \sum_j \mathbf{P}(T_i \mid X_j)\,\mathbf{P}(X_j),$$

where the first equality used the Markov property of Y, X, T. An iterative algorithm can be now used that provably converges to a local minimum of the objective function [5, 48].

1.4.3. Alternative methods

In the previous subsection the assumption was used that the involved random variables have finite number of possible values. This assumption is needed since there are no algorithms available that solve the task in a general setting [53]. Exact solutions are available for Gaussian and binary symmetric case [53]. In a discrete case, the method described in previous subsection can be used. For general continuous case, one can approximate a lower bound for the objective function using NNs [54].

In the context of BCIs, Publication II [2] proposes to use information bottleneck by taking X to be the feature vector, T to be the predicted class, and Y to be the correct class. More details are given in section 2.4. The finite case is enough for this setting as one can discretise the feature space and there is a finite number of classes in a typical BCI setting.

1.5. Partial information decomposition

This section gives an overview of PID. Its use to characterise and drive the learning of biological and artificial systems is a current direction of interest [55, 56, 57]. PID is used in Publication III [3] for decomposing mutual information and other information-theoretic quantities. Following subsections give an overview of the problem statement, a specific solution method, and discuss alternatives. Readers not interested in the details can skip this section, as a short discussion on this topic is also given later in section 3.1.3 discussing the third contribution.

1.5.1. Problem statement

PID is a method of decomposing mutual information between a target and source random variables into more specific components. In particular, given a random variable T called target and two random variables R_1, R_2 called sources, PID decomposes the mutual information $\mathrm{I}(T : R_1, R_2)$ into four components, called shared information (SI), synergistic information (CI), and unique information (UI) for each source variable [36]. The obtained four components can be characterised as follows:

- SI: information present in each individual source,
- $\mathrm{UI}(R_1)$: information provided by only one individual source R_1,
- $\mathrm{UI}(R_2)$: information provided by only one individual source R_2,
- CI: information only provided by the sources jointly, not individually.

Mathematically, the obtained four components are required to satisfy the equations [36]

$$\begin{aligned} \mathrm{I}(T : R_1, R_2) &= \mathrm{CI} + \mathrm{SI} + \mathrm{UI}(R_1) + \mathrm{UI}(R_2), \\ \mathrm{I}(T : R_1) &= \mathrm{SI} + \mathrm{UI}(R_1), \\ \mathrm{I}(T : R_2) &= \mathrm{SI} + \mathrm{UI}(R_2), \end{aligned}$$

but these do not uniquely determine the decomposition. In fact, there are many different decomposing methods available [6, 37, 58, 59]. Generalisations to more than two sources are also available but not needed in the current work. Even the method described in next section is applicable to more than two sources [6].

1.5.2. Solution method

Let us assume that the involved random variables T, R_1, R_2 have finite number of possible values. In the following, i, j, k denote some values of these random variables, and summation over these indices denote summations over all the possible values of the corresponding random variable.

In analogy to a possible derivation of mutual information as expected value of

$$i(i, j, k) = \log \frac{\mathbf{P}(T^i \mid R_1^j \cap R_2^j)}{\mathbf{P}(T^i)}$$

resulting in a formula

$$\mathrm{I}(T : R_1, R_2) = \sum_{i,j,k} \mathbf{P}(T^i \cap R_1^j \cap R_2^k) \log \frac{\mathbf{P}(T^i \mid R_1^j \cap R_2^j)}{\mathbf{P}(T^i)},$$

Makkeh et al. [6] derived a measure for shared information as expected value of

$$i_\cap^{sx}(i, j, k) = \log \frac{\mathbf{P}(T^i \mid R_1^j \cup R_2^k)}{\mathbf{P}(T^i)}.$$

resulting in a formula

$$\mathrm{I}_{\cap}^{sx} = \sum_{i,j,k} \mathbf{P}(T^i \cap R_1^j \cap R_2^k) \log \frac{\mathbf{P}(T^i \mid R_1^j \cup R_2^k)}{\mathbf{P}(T^i)}.$$

The consistency of $i_{\cap}^{sx}$ with requirements set for shared information [36] is justified through a decomposition to informative part $i_{\cap}^{sx+}$ and misinformative part $i_{\cap}^{sx-}$ [6, 60]

$$\begin{aligned} i_{\cap}^{sx}(i,j,k) &= i_{\cap}^{sx+}(i,j,k) - i_{\cap}^{sx-}(i,j,k) \\ i_{\cap}^{sx+}(i,j,k) &= \log \frac{1}{\mathbf{P}(R_1^j \cup R_2^k)} \\ i_{\cap}^{sx-}(i,j,k) &= \log \frac{\mathbf{P}(T^i)}{\mathbf{P}(T^i \cap (R_1^j \cup R_2^k))}. \end{aligned}$$

The PID terms are then calculated using a concept from combinatorics called Möbius inverse [61]. Calculations are done separately for the informative and misinformative parts [6].

1.5.3. Alternative methods

There have been many methods proposed to perform PID [6, 37, 58, 59]. The advantages of the $\mathrm{I}_{\cap}^{sx}$ measure is that, unlike the other referenced method, it has been shown to be differentiable with respect to the underlying probability distribution [62, 7]. Further, it is derived purely from information-theoretic principles and can be generalised to more than two source variables. Theoretically, it has been generalised to continuous and mixed systems of discrete and continuous variables, although an estimator for that case is still missing [62]. More recent work has appeared that has challenged some foundational ideas (use of inclusion-exclusion principle) underlying many methods of defining the PID terms [63].

In the context of RL, Publication III [3] proposes to use PID to characterise RL agents by taking agent's state as the target random variable T, and environment's state and agent's previous state as the source random variables R_1, R_2. The experiments were limited to the case of random variables with finite number of possible values since suitable estimator was not available for more general setting [3].

2. CLASSIFICATION IN BRAIN-COMPUTER INTERFACES

As mentioned in the introduction, the contributions of this thesis related to BCIs mainly concern the classification task in BCIs. Classification algorithms in BCIs take the features extracted from the processed EEG signal as input and output a prediction about the chosen command. Section 2.1 gives the required background information to discuss the contributions themselves. The main contributions are discussed in sections 2.2 and 2.4, respectively. The empirical verifications are given in sections 2.3 and 2.5, respectively.

2.1. Background

This section starts with a general discussion on feature extraction and classification methods in sections 2.1.1 and 2.1.2. Next, the main performance measure of BCIs, which allows defining the optimality of classification rules, is discussed in section 2.1.3. This section also introduces the notation used in the later sections. Section 2.1.4 addresses how the training dataset for the algorithms was compiled.

2.1.1. Feature extraction methods

Various feature extraction methods are available, and methods specific to BCIs have been introduced [64]. The earliest feature extraction methods include the power spectral density analysis (PSDA) [11], based on Fourier analysis of the signal, and canonical correlation analysis (CCA) [12], based on calculating correlations between different sets of signals. These methods have been widely used in the literature. A more recent feature extraction method that has achieved impressive results is TRCA [13, 65], which is based on analysing covariances between different parts of the signal. These three methods are used in this thesis to empirically verify the introduced algorithms and compare them to the literature. PSDA and CCA methods were chosen as they have been widely used in the literature and provide many options for comparison. TRCA method was used since it is a more recent method with a better performance.

All these different feature extraction methods are similar in the sense that for each possible command that the user can send through BCI, the feature extraction method extracts one feature, and a larger feature value is interpreted as higher confidence towards the corresponding command. Many other feature extraction methods satisfy the same properties [64, 20, 66, 67], and as a consequence, many BCIs simply predict the command corresponding to the highest feature value in the classification step. In this thesis, more advanced classification rules are proposed.

2.1.2. Classification methods

As mentioned in section 2.1.1, a classification rule often occurring in SSVEP-based BCI literature is to predict the command corresponding to the highest feature value. This rule can be written simply as

$$\arg\max_i f_i \tag{2.1}$$

where f_i denotes the feature value for command i. The classification rule (2.1) is called arg max classifier in this thesis and possible outcomes of the classification will be referred to as classes instead of commands in the following.

The simplicity of the classification rule raises the question of whether it would be beneficial to use more complex classification methods to achieve better performance. Standard machine learning methods like linear discriminant analysis (LDA) [15, 16], support vector machines [20, 17, 18, 19], and NNs [21] have been applied to the classification problem. However, note that BCI is a system to be used in real-time. In addition to accuracy, which is often maximised by standard machine learning methods, the efficiency of a BCI also depends on the amount of time it takes to make a prediction. Collecting more data could result in a more reliable prediction, but at the cost of the system's responsiveness. Also, BCIs can benefit from leaving some samples unclassified if there is not enough confidence in their predictions.

The standard performance measure for BCIs, called information transfer rate (ITR), takes into account the accuracy, detection time, and the number of possible commands, combining these into a single value characterising the amount of information the system can transfer in a unit of time. This thesis introduces classification rules that aim to directly maximise the ITR, giving it obvious benefits over standard machine learning algorithms.

2.1.3. Information transfer rate

Work towards the first two contributions of this thesis started from the observation that the standard performance measure for BCIs, the ITR, is defined using information theory. Let us denote the random variables corresponding to the correct class and the predicted class as C and P, respectively. Let us denote the number of classes as n and the classes themselves by $1, 2, 3, \ldots, n$. The formula for calculating ITR in units of bits per prediction is

$$\mathrm{ITR}_s = \log n + a \log_2 a + (1-a) \log_2 \left(\frac{1-a}{n-1} \right), \tag{2.2}$$

where a denotes the accuracy of the classifier. This formula was derived under the following assumptions

1. all classes are equally likely,

2. in case of misclassification, all incorrect classes are equally likely to be chosen,
3. the probability that correct classification is made has to be the same for each class [40, 44, 68].

If these assumptions are met, ITR is the amount of information that the predicted class gives about the correct class, that is, the mutual information $\mathrm{I}(P,C)$ [68]. In simpler terms, it is the average amount of information sent through the channel by a single prediction. Alternatively, the definition (2.2) could be interpreted under the listed assumptions 2–3 as a channel capacity of the BCI [44]. Appendix E of Publication I [1] gives the derivation in the case of mutual information.

A fruitful treatment of channel capacity would require using complex encoding schemes, which might not be possible in BCIs since human is on one end of the channel [46]. In the case of BCIs, the output is the predicted class, and the amount of complexity that can be introduced is limited. Thus, ITR is interpreted as mutual information instead in this thesis. Since channel capacity itself is defined through mutual information, this approach simply means that this thesis relies on a more basic notion. Mutual information provides an intuitive understanding of the performance measure—it is the amount of information obtained about the correct class through predicted class.

A widely occurring problem in BCI literature is that the assumptions 1–3 are often ignored, possibly resulting in an incorrect estimate of the amount of information transferred by the BCI [68, 69]. This thesis relies on the definition of mutual information (1.1) instead of formula (2.2) to avoid incorrect estimates due to assumptions being not met. Using mutual information allows the use of general optimisation procedures without the need to take into account the assumptions 1–3 while obtaining the correct estimate for the amount of transferred information.

Usually, ITR is reported in units of bits per minute, which is calculated as

$$\mathrm{ITR} = \mathrm{ITR}_s \frac{60}{\mathrm{MDT}},$$

where mean detection time (MDT) is the average amount of time in seconds it takes to make a classification. However, in this thesis, as discussed, a metric based on mutual information is used instead

$$\mathrm{ITR}_{mi} = \mathrm{I}(P,C) \frac{60}{\mathrm{MDT}}. \tag{2.3}$$

The details of calculating MDT are discussed in sections 2.2.3 and 2.3.3.

2.1.4. Constructing training dataset from EEG data

All the experiments reported in this thesis were conducted on publicly available EEG datasets [70, 28]. An overlapping sliding window was used on the EEG signal to obtain enough data for machine learning purposes. Denoting the window

length as w and step length as s, using a sliding window means that feature extraction is done on a signal segment of length w and the time step between consecutive feature extractions is s. Thus there is an overlap of $w-s$ between consecutive time windows.

The used datasets contain separate recordings, called trials, for each command. A trial consists of a short EEG recording from the subject while they are trying to send the specified command. The introduced algorithms are trained separately for each subject. The performance measure for a subject is estimated using cross-validation so that each fold corresponds to one trial of each command. Thus in each fold, the training dataset for a subject consists of all except one trial, and the test dataset consists of the remaining trial of each command. This cross-validation is used separately for each subject to estimate performance measures.

2.2. The first contribution: optimal classification with a specific rule

This section describes the main contribution of Publication I [1]. In this contribution, the task of finding an optimal classifier was solved for a specific set of classification rules. Condition for optimality was taken to be the highest value of the performance measure ITR_{mi}. In section 2.2.1, the set of considered classification rules is described. Section 2.2.2 describes how the distribution of features is estimated. The calculation of MDT, which is a part of the performance measure, is discussed in section 2.2.3. The last sections describe how the performance measure is calculated and optimised.

2.2.1. Classification rule

The classification rule used in this contribution makes use of the properties that most feature extraction methods have. In particular, one feature is extracted for every class, and a larger value for a feature is interpreted as higher confidence toward the corresponding class. Recall that classes are denoted $1,\ldots,n$ and corresponding features are denoted $f_1,\ldots,f_n$. The classification rule uses a threshold t_i for each feature f_i, and a sample is classified to a class as follows: a sample is classified into class k if for this sample $f_k \geq t_k$ and $f_j < t_j$ for all the other classes.

From the classification rule definition, it is clear that there might be samples left unclassified. Unclassified samples are interpreted as the case when the classifier is not confident enough in the prediction. Improving accuracy at the expense of detection time could be important for real-time systems. The goal of the following sections is to represent the performance measure ITR as a differentiable function of the thresholds and optimise it.

2.2.2. Modelling features as random variables

It is enough to know the probabilities $\mathbf{P}(P_i \cap C_k)$ to calculate the performance measure ITR_{mi}. For optimisation purposes, however, it is also useful to be able to calculate partial derivatives of ITR_{mi}. If a feature is modelled as a continuous random variable, the derivative of its CDF is the corresponding PDF. As will be seen later, knowing the derivatives of these CDFs is enough to find the partial derivatives of ITR_{mi}.

Let us introduce notation for the involved random variables. Let us denote the random variable modelling the feature corresponding to class i as F_i. Then the notation of feature value f_i, used in previous sections, corresponds to the realisation of F_i. Now the CDF for the feature F_i given the correct class C_k is denoted as $F_{F_i|C_k}$. Knowing these CDFs allows one to calculate the required probabilities.

The functions $F_{F_i|C_k}$ were estimated from the training data. By empirical observations (see Figure 3 for an example), these random variables were assumed to follow the skew-normal distribution. The PDF of a skew-normal distribution [71] has the form

$$f_{F_i|C_k}(x) = \frac{2}{\omega} f_N\left(\frac{x-\xi}{\omega}\right) F_N\left(\alpha \cdot \frac{x-\xi}{\omega}\right), \tag{2.4}$$

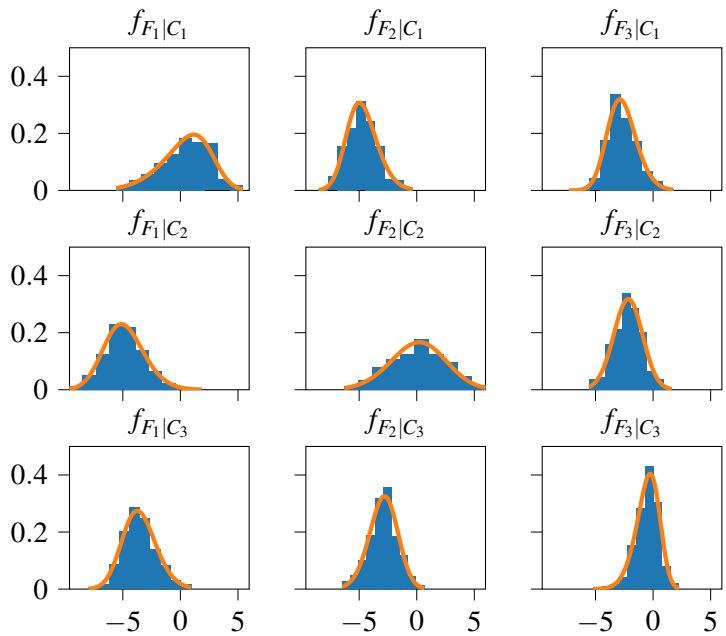


Figure 3: A comparison between histograms and fit skew-normal distributions of features. The features were calculated using trials 2–5 of Subject 1 of the dataset discussed in section 2.3.1. Here, the features are obtained by LDA applied to CCA and PSDA features, as discussed in section 2.3.2.

where f_N is the PDF of the standard normal distribution

$$f_N(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}, \tag{2.5}$$

F_N is the CDF of the standard normal distribution, and $\alpha, \xi, \omega \in \mathbb{R}$ are parameters with $\omega > 0$. Thus finding the distribution of $F_i \mid C_k$ is reduced to finding the best values for the parameters α, ξ, ω.

These values could be found, for example, by the MM. The MM estimates are consistent, meaning that they converge to true values, assuming that the underlying distribution is indeed a skew-normal distribution and assuming the ergodicity over trials mentioned in section 1.3.4.

Another method for finding the values of the parameters is maximum likelihood estimation (MLE). MLE can be easily applied in case the samples are independent since the likelihood function takes a known form in this case. The samples in the current setting are not independent due to the overlapping sliding window. However, since the MLE method was much faster than the MM and the results were very close, then MLE was used to estimate the parameters, with the likelihood calculated as if the samples were independent. If the samples were indeed independent, this estimation method would be consistent.

2.2.3. Calculating mean detection time

Since the introduced algorithm is allowed to leave samples unclassified, MDT is not a constant and must be estimated from the data. This section discusses how MDT is estimated in the training process. Recall that a sliding window with length w is used so that after a sample is left unclassified, the next classification attempt is performed on window shifted by step s as discussed in section 2.1.4.

The MDT is estimated with the help of probability theory. Recall that the notation P_i is used for the event $\{P = i\}$. The number of failed classification attempts before a successful one is modelled by a geometric random variable. In this setting, the probability of making a prediction can be written as

$$\mathbf{P}\left(\bigcup_{i=1}^{n} P_i\right),$$

and the expected number of failures before a successful prediction is

$$\frac{1}{\mathbf{P}\left(\bigcup_{i=1}^{n} P_i\right)} - 1,$$

assuming that the test samples are independent. Let us denote the window length by w and the time step between consecutive prediction attempts by s. Then MDT can be estimated by

$$\mathrm{MDT} = w + \left(\frac{1}{\mathbf{P}\left(\bigcup_{i=1}^{n} P_i\right)} - 1\right) \cdot s. \tag{2.6}$$

Here s is multiplied by the number of expected failures before a successful prediction. In the experiments of the first contribution, window length w was set to 1 second, and the time step s between the beginnings of consecutive windows was 0.125 seconds.

Alternatively, one could replace the term $\frac{60}{\mathrm{MDT}}$ in the performance measure definition (2.3) with some other function of prediction probabilities and interpret this term simply as providing a trade-off between the mutual information $\mathrm{I}(P,C)$ and the prediction probabilities. In simpler terms, it allows choosing whether to prefer accuracy over speed or the other way round.

2.2.4. Calculating information transfer rate

Modelling the features as continuous random variables makes it relatively easy to calculate the probabilities of predicting different classes. For our purposes, conditional distributions of the feature given the correct class are needed to calculate the performance measure ITR_{mi}. These distributions are estimated as described in section 2.2.2.

In order to calculate the required probabilities, a simplifying assumption is made. As in naïve Bayes, an assumption is made that the features are conditionally independent, conditioned on the correct class C. Then the probability of predicting class i given class k is

$$\mathbf{P}(P_i \mid C_k) = \overline{F}_{F_i|C_k}(t_i) \prod_{j \in \{1,\ldots,n\} \setminus \{i\}} F_{F_j|C_k}(t_j). \tag{2.7}$$

Note that this probability depends on the thresholds. The probabilities $\mathbf{P}(C_k)$ can be estimated from the training dataset. Altogether, the joint probabilities

$$\mathbf{P}(P_i \cap C_k) = \mathbf{P}(P_i \mid C_k)\,\mathbf{P}(C_k)$$

can be calculated from the thresholds. These probabilities determine the performance measure ITR_{mi}. Therefore ITR_{mi} is indeed a function of thresholds. Some technical details were omitted from this discussion. See Appendix G of Publication I [1].

2.2.5. Calculating gradient and Hessian of information transfer rate

The performance measure ITR_{mi} is a function of thresholds. Therefore, its gradient consists of partial derivatives with respect to each threshold. In order to calculate the derivatives, derivatives of $\mathbf{P}(P_i \mid C_k)$ are required. These probabilities are calculated from the CDFs of skew-normal distributions, see Equation (2.7).

Since skew-normal distribution is continuous, the derivative of its CDF is given by the PDF. Another important property is that the CDF of skew-normal distribution never obtains the value 0. Therefore the special case of $0 \log 0$ in the definition of mutual information never occurs. Thus it is clear that the derivatives can be easily calculated. The details are given in Appendix H of Publication I [1].

Finally, as a side note, note that the Hessian of the performance measure ITR_{mi} could also be relatively easily calculated. Using Hessian would allow using more sophisticated optimisation algorithms that use this second-order information. Hessian was neither used in the experiments in here nor in the corresponding publication [1].

In order to calculate the Hessian, it is enough to note that the derivative of the PDF of the skew-normal distribution (2.4) is

$$\frac{df_{F_i|C_k}}{dx}(x) = \frac{2}{\omega^2}\frac{df_N}{dx}\left(\frac{x-\xi}{\omega}\right)F_n\left(\alpha\frac{x-\xi}{\omega}\right) + \frac{2\alpha}{\omega^2}f_N\left(\frac{x-\xi}{\omega}\right)f_N\left(\alpha\frac{x-\xi}{\omega}\right)$$

where the derivative of the PDF of the standard normal distribution (2.5) is

$$\frac{df_N}{dx}(x) = \frac{-x}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}.$$

Using this knowledge together with standard derivative calculation rules the full Hessian can be calculated.

2.2.6. Optimising information transfer rate

The performance measure ITR_{mi} was optimised using an optimisation method that can use the gradient of the objective function. In particular, Publication I [1] used the gradient ascent algorithm. However, in Publication II [2], the gradient ascent algorithm was replaced with the basin-hopping algorithm [72], which seemed more stable than the original implementation of the gradient ascent.

The basin-hopping algorithm works well with objective functions that have a funnel-like landscape [73]. A visualisation of the objective function ITR_{mi} for a simple case is shown in Figure 4. It can be seen that near the optimum, the landscape is indeed funnel-like, suggesting that basin-hopping could be a good choice. Note that these optimisation methods are not guaranteed to find the global optimum of the performance measure. However, as shown in section 2.3, the obtained local optima were enough to outperform previous results.

2.2.7. Summary

The previous subsections described the optimisation algorithm introduced in Publication I [1]. The optimisation procedure takes as input training EEG data and outputs the classification thresholds for the classification rule introduced in section 2.2.1. These thresholds are obtained so that they optimise the performance measure ITR_{mi}. Performance measure ITR_{mi} itself is a function of the thresholds. Thus, optimising the performance measure is reduced to optimising a multivariable function. The optimisation procedure makes the following assumptions which have all already been discussed:

1. ergodicity over trials (section 2.2.2),

2. feature given correct class follows a known continuous distribution (section 2.2.2),
3. features are conditionally independent given the correct class (section 2.2.4),
4. larger feature value can be interpreted as higher confidence towards the corresponding class (section 2.2.1).

Assumption 1 could be replaced by a stronger assumption that classification attempts in the training set are independent if such data is available. Independence and assumption 2 are common assumptions when considering theoretical questions in machine learning. In the current setting, independence was replaced with a weaker assumption of ergodicity. Assumption 3 is essentially the assumption used in naïve Bayes. Therefore, these assumptions can be considered quite standard for machine learning purposes. Assumption 4 can be avoided by transforming the features into another set of features for which this condition holds. Also, multiple feature extraction methods could be used by using dimensionality reduction (see section 2.3.2).

In case the assumptions 1–4 are met, for a large enough sample size, the optimisation procedure finds a local maximum of the performance measure ITR_{mi} (to some precision). As could be expected from this theoretical result and as will be seen in section 2.3, this algorithm gives good results in practice.

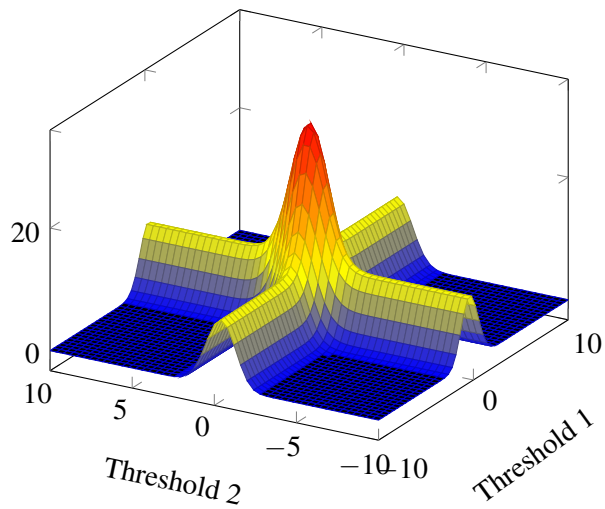


Figure 4: A simple visualisation of the objective function. It is assumed that there are two classes, thus two thresholds have to be found. Random variables $F_i \mid C_k$ are assumed to follow normal distribution. Means of $F_1 \mid C_1$ and $F_2 \mid C_2$ are 1 and means of $F_1 \mid C_2$ and $F_2 \mid C_1$ are -1. Standard deviations are all equal to 1.

2.3. Empirical verification and comparison of the first contribution

This section describes how the introduced algorithm was compared to existing algorithms. For a fair comparison, a publicly available EEG dataset was used. This dataset is introduced in section 2.3.1. Feature extraction methods had to be chosen since the introduced algorithm only deals with classification. The chosen methods are described in section 2.3.2. The results are presented in section 2.3.3, and statistical analysis is discussed in section 2.3.4. The exact implementation is available in the corresponding code repository[1].

2.3.1. Dataset 1

A publicly available dataset [70] was used for testing the introduced algorithm. This dataset is relatively small, containing EEG recordings of four subjects. An advantage of this dataset is that the recording trials are quite long compared to some newer datasets [28]. Thus there is enough data for machine learning purposes. Recall that the general setting of an SSVEP-based BCI was described in Introduction 1.1. Here, specifics of the current dataset are given.

The dataset contains EEG recordings of four healthy subjects with normal or corrected-to-normal vision. Subjects were not experienced with SSVEP-based BCIs. Three stimulation frequencies were used: 8, 14, and 28 Hz. For each subject, there are EEG recordings of five trials for each class. Each trial consists of 25 seconds of EEG data. The first 5 seconds and the last 5 seconds of each trial are without visual stimulation. The visual stimuli were displayed on a 21-inch CRT computer monitor with a 170 Hz refresh rate, placed approximately 90 cm away from the eyes of the subject. The visual stimuli were black and white checkerboards with 6×6 checks. The dataset was recorded using BIOSEMI EEG (Biosemi inc. Amsterdam) system with 128 channels and 2048 Hz sampling frequency, which was downsampled to 256 Hz. Only the data from channels O1 and O2 were used in the following experiments. For more details, refer to the dataset article [70].

Using a sliding window on each trial as described in section 2.1.4 with a window length of 1 second and step of 0.125 seconds, one trial gives $(15-1)/0.125+1 = 113$ samples. Thus there are altogether $113 \cdot 5 = 565$ samples for each class.

2.3.2. Signal processing and feature extraction

First, the signal of the Cz channel was subtracted from the signals of the O1 and O2 channels. This step was not used in Publication I [1]. Then, the signal was detrended to remove the piecewise linear trend. Four equally spaced breakpoints were used to divide the signal into five pieces for detrending.

[1] `https://github.com/antiingel/ITR-optimisation`

The standard PSDA [11] and CCA [12] feature extraction methods were used in these experiments. An LDA dimensionality reduction was used on the features to obtain a single feature for each class. These features are the distances to the decision borders found by LDA. This dimensionality reduction allows using multiple feature extraction methods.

For the PSDA method, another signal processing step was used. A Keiser window with parameter $\beta = \pi\alpha = 14$ was applied to the detrended signal (refer to [74] for the definition and meaning of parameters). The PSDA method uses the fast Fourier transform to estimate the signal's power spectrum. The power of the flickering frequency then serves as a good feature. In addition to the fundamental frequency of the flickering, the second and third harmonics were also used. To reduce the number of features, the corresponding powers from these channels were summed to form a single feature for each harmonic. Also, the sum of these resulting three features was used as a feature. The PSDA method is applied separately to each channel of the multichannel EEG signal. In this thesis, only the channels O1 and O2 were used.

The CCA method combines information from the multichannel EEG signal into a single feature. Essentially, the CCA method calculates a correlation between two sets of signals: multichannel EEG and a set of reference signals. The reference signals are sine and cosine signals with frequencies corresponding to the fundamental frequency of the flickering and its harmonics. Again, two harmonics, in addition to the fundamental frequency, were used.

2.3.3. Results and comparison

The obtained results were compared to related works [17, 75, 76, 18, 77, 19, 78, 79, 16, 80], where the same dataset [70] was used. The related works were found by searching through Google Scholar for publications citing the dataset article [70]. From these publications, the ones in which an SSVEP-based BCI was implemented, the dataset [70] was used to evaluate the performance, and the results were reported, were chosen for comparison. Together with the results of the two contributions of this thesis [1, 2], these results are shown in Table 1. One related article [80] is omitted from the comparison since only a subset of the available data was used, and it is not documented which trials of which subjects were used. In addition to the comparison given here, note that in Publication I [1], the introduced algorithm was shown to outperform a random forest classifier.

Table 1 contains the standard ITR, which is currently considered the main performance measure for comparison. The performance measures for the related works were obtained from the corresponding articles. The performance measures for the introduced algorithms were calculated slightly differently from Publications I and II [1, 2]. Publications I and II used a sliding window on the test data exactly the same way as on training data and calculated MDT (2.6) and ITR (2.3) by estimating the required probabilities from the confusion matrix obtained from

Table 1: Comparison of the results to related work. Notation S1–S4 refers to the four subjects. Table entries are the standard ITRs.

	[17]	[75]	[76]	[77]	[18]	[19]	[78]	[79]	[16]	**[1]**	**[2]**
S1	-	12.7	20.5*	-	33.1	-	17.6	-	29.9*	45.3	54.1
S2	-	12.2	6.5*	-	23.0*	-	17.1	-	24.3*	30.0	34.6
S3	-	5.3	0.5*	-	4.0*	-	22.3	-	32.7*	45.7	54.9
S4	-	2.4	8.7*	-	0.5*	-	11.6	-	15.1*	6.9	11.9
Avg	<8*	8.2	9.0*	14.2	15.5	17.0	17.2	18.3	25.5*	31.9	38.9

* These values were not explicitly reported in the corresponding articles but were computed based on other available data in the article. This included finding approximate values from graphs. For [16] MDT was taken as the average MDT of 2.25 seconds. The actual values might differ.

classifications on this test data.

In the results presented here, the MDT is calculated differently. The test trial is treated as an online signal instead of applying a sliding window to the whole signal in advance and estimating MDT for a hypothetical independent dataset. In particular, the first classification attempt occurs for the initial 1 second of data. If this classification fails, the next attempt is after 0.125 seconds. However, if the classification succeeds, the next classification attempt is after 1 second. The whole signal is processed this way by continuing similarly. This method avoids reusing the data. The performance measures estimated in this way are presented in Table 1[2].

2.3.4. Statistical analysis

The performance of algorithms from this work was compared to the related works by t-test [81] at the significance level of 0.05. Note that due to the amount of data, weaker tests like the sign test [82] or the Wilcoxon signed-rank test cannot be used to show a statistical significance of differences in ITR[3]. The t-test, however, makes the assumption that the differences in ITR are normally distributed. Whether this assumption holds was not verified in this work. Thus the following hypothesis testing results hold only under the assumption that the differences in ITR are normally distributed.

[2]In Publications I and II [1, 2], the test trial was not treated as an online signal as here, and higher ITR values were obtained there mainly due to the estimated MDTs being lower. Different results were obtained likely due to the independence assumption that the MDT formula (2.6) depends on being violated. Thus, the results presented here likely give more accurate estimates of the actual performance.

[3]Both the sign test and the Wilcoxon signed-rank test require a sample size of at least 5 to be able to reject the null hypothesis at the significance level of 0.05. The minimal difference in ITR that can be detected with a t-test at that significance level with power 0.8 is approximately $1.67\hat{\sigma}$ where $\hat{\sigma}$ denotes the sample standard deviation of the difference in ITR.

The t-test was used with the alternative hypothesis that the ITR of the algorithm from Publication I [1] is larger than the ITR from related work and with the null hypothesis that these ITRs are equal. The p-values for this t-test were 0.029, 0.049, 0.069, 0.066, and 0.156 for [75, 76, 18, 78, 16], respectively. The same test for the algorithm from Publication II [2] gave the p values 0.021, 0.033, 0.042, 0.038, and 0.062 for [75, 76, 18, 78, 16], respectively. Thus the second contribution significantly outperforms most of the related work. The significance of the difference in ITR could not be shown only for one reference [16]. However, note that the used ITR values might differ from the actual values for this related work, as noted in Table 1. Significance for larger ITR of the first algorithm could only be shown for [75, 76] with this amount of data. However, one could expect that with more data, the significance of the difference in performance could have been shown for more related works.

2.4. The second contribution: optimal classification rule

Building on the ideas of Publication I [1], in this contribution [2], a much more diverse set of classification rules is considered, and fewer assumptions are needed. The discussion of the second contribution starts with introducing a general information-theoretic optimisation method called information bottleneck in section 2.4.1. Using the information bottleneck allows optimising the mutual information $\mathrm{I}(P,C)$ between the predicted class P and correct class C. Recall that this mutual information occurs in the performance measure ITR_{mi}. This time, the classification rule is not fixed in advance as in the first contribution. However, a drawback is that the optimisation procedure does not take MDT directly into account. Thus, the amount of information transferred in a single prediction is maximised.

Section 2.4.2 deals with discretising the features since discrete features are required by the used information bottleneck algorithm. Section 2.4.3 describes how the output of the information bottleneck is used in classification.

2.4.1. Information bottleneck

This section gives an overview of the information bottleneck method [48], which is a required background knowledge for the second contribution. The information bottleneck solves an information-theoretic optimisation task in which two random variables are given, and the goal is to find a third random variable which is a compressed version of the first variable but still has much information about the second variable. The method aims to find the best trade-off between compressing the representation and preserving meaningful information.

In the context of this thesis, the given random variables are feature vector X, introduced in section 2.4.2, and the correct class C. The third random variable is the predicted class P. In more detail, the information bottleneck method takes the

probabilities $\mathbf{P}(X_k \cap C_j)$ as input and solves the optimisation task

$$\arg\min_{\mathbf{P}(P_i|X_k)} [\mathrm{I}(P,X) - \beta\, \mathrm{I}(P,C)],$$

where β is a non-negative parameter and minimisation is subject to the constraint

$$\mathbf{P}(P_i \mid C_j, X_k) = \mathbf{P}(P_i \mid X_k).$$

Here, the feature vector X is the random variable being compressed into the random variable P, modelling the predicted class, and P has to preserve meaningful information about the correct class C. As can be seen, the optimisation aims to maximise the mutual information $\mathrm{I}(P,C)$, which also occurs in the performance measure ITR_{mi}, in the presence of the discussed trade-off. The parameter β can be used to tune the relative importance of the two terms, which are the compression $\mathrm{I}(P,X)$ and the relevance to correct class $\mathrm{I}(P,C)$. In the current application, having a high mutual information $\mathrm{I}(P,C)$ is more important, so larger values of β are preferred.

How the result of the information bottleneck is related to the classification task is better seen from a related method called DIB [5]. Deterministic in the name of DIB comes from the fact that the resulting probabilities can be interpreted as a function since the resulting probabilities satisfy the condition that for each k, there is i such that $\mathbf{P}(P_i \mid X_k) = 1$. Since X is the feature vector and P is the predicted class, the function can be used as a classification rule.

DIB differs from the original information bottleneck only by the objective function. In particular, DIB solves the optimisation task

$$\arg\min_{\mathbf{P}(P_i|X_k)} [\mathrm{H}(P) - \beta\, \mathrm{I}(P,C)].$$

This task is solved in a more general framework by introducing additional parameter α and considering the task

$$\arg\min_{\mathbf{P}(P_i|X_k)} [\mathrm{H}(P) - \alpha\, \mathrm{H}(P \mid X) - \beta\, \mathrm{I}(P,C)]. \tag{2.8}$$

Now DIB optimisation task is solved by taking the limit of the solutions of (2.8) in the process $\alpha \to 0$. The task (2.8) is equivalent to the original information bottleneck if $\alpha = 1$. Values between 0 and 1 interpolate between the original information bottleneck and DIB.

The result of DIB can always be interpreted as a function. However, in the experiments of Publication II [2], it was observed that the original information bottleneck often also gives a result that is very close to deterministic (that is, $\mathrm{H}(P \mid X)$ is close to zero). Therefore, both the original information bottleneck and DIB were used in the experiments of Publication II [2]. In this thesis, only the results for the original information bottleneck are presented.

2.4.2. Discretising features

Contrary to the first contribution [1], in which the features had to be modelled by continuous random variables, so that the derivatives could be found, the features needed to be discrete in this contribution. Discretising is required because the information-bottleneck algorithm used in this thesis requires discrete random variables. There were no algorithms available which could solve the general continuous case exactly [53]. Therefore, the continuous features are divided into a fixed number of equal-size bins. The exact choice of bins used in the experiments of this thesis is discussed in section 2.5.3.

After the bins have been fixed, one way to estimate the required probabilities would be simply using a histogram. However, following the approach of the first contribution, the estimated CDFs of features were used instead. Publication II [2] compares these two methods and shows that using the estimated CDFs gives slightly better results.

In more detail, recall that the random variable corresponding to class i is denoted by F_i. Let us denote the random variable corresponding to the bin to which the feature belongs as B_i. With this notation, a bin $[\ell, h)$ satisfies $B_i = [\ell, h)$ if and only if $F_i \in [\ell, h)$. The required probabilities can then be denoted by $\mathbf{P}(B_i = [\ell, h) \mid C_k)$. As already mentioned, these could simply be estimated by a histogram, but by using the skew-normal distribution of F_i given C_k, we obtain

$$\mathbf{P}(B_i = [\ell, h) \mid C_k) = F_{F_i|C_k}(h) - F_{F_i|C_k}(\ell).$$

Combining random variables corresponding to bins into a vector

$$X = (B_1, B_2, B_3, \ldots, B_n)$$

and assuming that these random variables are conditionally independent, conditioned on C_k, we can calculate

$$\mathbf{P}(X = (b_1, \ldots, b_n) \mid C_k) = \prod_{i=1}^{n} \mathbf{P}(B_i = b_i \mid C_k).$$

Multiplying by the probability $\mathbf{P}(C_k)$, which can be estimated as the fraction of samples with correct class k, gives the required probabilities $\mathbf{P}(X_j \cap C_k)$ that are the input to the information bottleneck algorithm.

2.4.3. Classification

The information bottleneck takes the probabilities $\mathbf{P}(X_j \cap C_k)$ as input and outputs probabilities $\mathbf{P}(P_i \mid X_j)$. The easiest way to classify samples is then to predict class i for which

$$\mathbf{P}(P_i \mid X_j) = 1,$$

if there exists such a class and leave the sample unclassified otherwise. With DIB, there is always a class satisfying this condition. This classification method is called Classifier 1 in Publication II [2]. A visualisation of the feature space when classifying with this rule is shown in Figure 5. Better results were obtained with a different classifier that considers a larger set of feature space, considering, in addition, the neighbouring points in the feature space. This rule is described next.

The classification is essentially performed by counting the number of neighbouring regions (Cartesian products of bins) that are classified into the same class. If this number exceeds a certain threshold t for some class, the sample is classified to that class. The amount of neighbouring regions to consider is characterised by a parameter N. In more detail, let us denote the bin edges in increasing order for feature i as $b^i_1, \ldots, b^i_{m_i}$. Here, m_i denotes the number of bins for feature i. If the current sample corresponds to bins $B_i = [b^i_{j_i}, b^i_{j_i+1})$ for some $j_1, \ldots, j_n$, then the number of values $s_1, \ldots, s_n \in \{-N, -N+1, \ldots, N-1, N\}$ is counted for which

$$\mathbf{P}\left(P_i \mid X = \left([b^1_{j_1+s_1}, b^1_{j_1+1+s_1}), \ldots, [b^n_{j_n+s_n}, b^n_{j_n+1+s_n})\right)\right) = 1.$$

If this number exceeds a fixed threshold t, the sample is classified as class i. For example, if $N = 0$, this rule corresponds to Classifier 1. Considering another example, in the case of two classes, a $(2N+1) \times (2N+1)$ grid of regions centred at the region corresponding to the current sample is considered. The number of regions classified to same the class in this grid is then counted.

The threshold t is chosen large enough so that only one class can satisfy the condition. In the results presented in this thesis, the parameters N and t of the

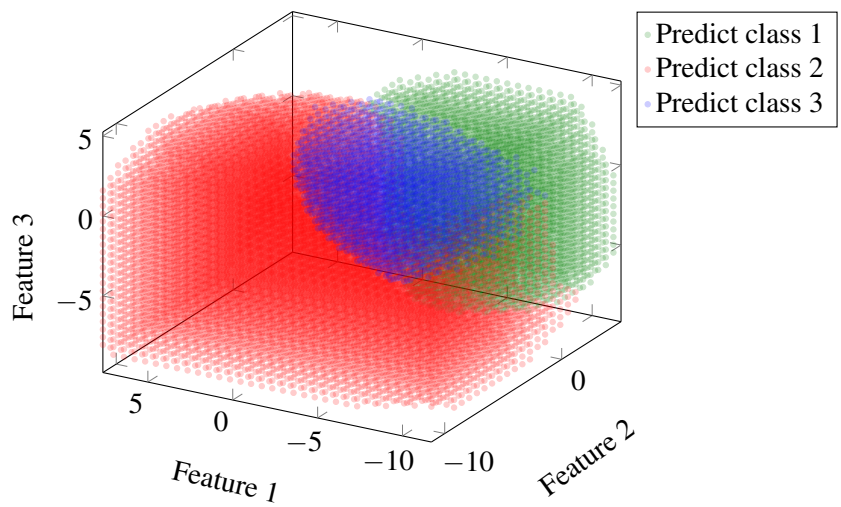


Figure 5: A visualisation of the feature space with classifications made by the Classifier 1. Here, the dataset and feature extraction method described in section 2.3 were used. The trials 1–4 of Subject 1 were used in training. Classes 1, 2, 3 correspond to frequencies of 8, 14, 28 Hz, respectively.

classifier were chosen using one fold of the cross-validation. Specifically, the values that maximised ITR were chosen.

2.4.4. Summary

The previous subsections described the optimisation algorithm introduced in Publication II [2]. The optimisation procedure takes the features calculated from the training EEG data as input and outputs a classification rule that optimises mutual information between correct class C and predicted class P while compressing the representation. Therefore, the amount of information that the BCI can transfer in a prediction is maximised. Note that the bins used to discretise the features are fixed beforehand, limiting what kind of classification rules are considered in the optimisation procedure. One possible way of choosing bins is given in section 2.5.3.

The optimisation procedure makes the following assumptions:

1. ergodicity over trials (section 2.2.2),
2. feature given correct class follows a known continuous distribution (section 2.2.2),
3. bins corresponding to features are conditionally independent given the correct class (section 2.4.2).

The same assumptions were made in the first contribution, with the only difference that assumption 3 was about features instead of bins. Note that this algorithm does not need to assume that a larger feature value can be interpreted as higher confidence towards the corresponding class. These assumptions were already discussed in section 2.2.7.

Compared to the first contribution [1], this algorithm does not take MDT directly into account, and scaling to a larger number of classes is worse than the first algorithm. Despite these drawbacks, it seems to slightly outperform the first algorithm, as seen in section 2.3.3. This better performance is not surprising as a much more diverse set of classification rules is considered.

Publication II [2] also mentions that information-theoretic tools can be used to choose other parts of the BCI's signal pipeline. Due to data processing inequality, the mutual information between features and the correct class can only decrease at each processing step. One should aim to keep this mutual information as high as possible since it is an upper bound for mutual information $\mathrm{I}(P,C)$.

2.5. Empirical verification and comparison of the second contribution

This section describes how the introduced algorithm was compared to existing algorithms. First, the algorithm was compared to the one introduced in Publication I [1] and to all the related articles [17, 75, 76, 18, 77, 19, 78, 79, 16]. The same dataset and feature extraction methods were used as in the first contribution

for that comparison. The algorithm was also tested with a more recent dataset (see section 2.5.1) and feature extraction method (see section 2.5.2), discussed in the following sections. How the number of bins was calculated to discretise the features is discussed in section 2.5.3. The results are presented in sections 2.5.4 and 2.3.3, and statistical analysis is discussed in sections 2.5.5 and 2.3.4. The exact implementation is available in the corresponding code repository[4].

2.5.1. Dataset 2

The introduced algorithm was tested on two publicly available datasets. These two datasets will be referred to as Datasets 1 and 2 hereafter. Datasets 1 was already discussed in section 2.3.1. The general setting of an SSVEP-based BCI was described in the Introduction. Here, specifics of Dataset 2 are given.

The second dataset [28] contains EEG recordings for 35 healthy subjects with normal or corrected-to-normal vision. Eight of the subjects had previous experience with their SSVEP-based BCI system. There are 40 stimulation frequencies ranging from 8 to 15.8 Hz with 0.2 Hz intervals. For each subject, there are EEG recordings of six trials for each class. Each trial consists of 6 seconds of data. The first 0.5 seconds and the last 0.5 seconds of each trial are without visual stimulation.

The visual stimuli were displayed on a 23.6 inch LCD monitor with 1920×1080 resolution and 60 Hz refresh rate, placed approximately 70 cm away from the subject. The visual stimuli were white squares on a black background. Each square had a letter, digit, or some other symbol on it. The dataset was recorded using the Synamps2 EEG system (Neuroscan Inc) with 64 channels and a 1000 Hz sampling frequency, which was downsampled to 250 Hz. Only the data from channels Pz, PO5, PO3, POz, PO4, PO6, O1, Oz, and O2 were used in the following experiments. For more details, refer to the dataset article [28].

Using a sliding window on each trial as described in section 2.1.4 with a window length of 0.5 seconds and step of 0.124 seconds, one trial gives $\lfloor (5 - 0.5)/0.124 \rfloor + 1 = 36$ samples. Thus there are altogether $36 \cdot 6 = 216$ samples for each class.

2.5.2. Signal processing and feature extraction

On the two datasets, different feature extraction methods were used. On Dataset 1, the signal processing and feature extraction method discussed in section 2.3.2 was used. On Dataset 2, the TRCA [13, 65] feature extraction method was used.

TRCA method assumes that the signal is represented by a linear combination of task-related components and task-unrelated components. The goal is to find the linear combination coefficients so that the task-related components have maximal similarity over trials. Similarly to CCA, TRCA combines information from mul-

[4] `https://github.com/antiingel/information-bottleneck-BCI`

tiple channels. Instead of having fixed reference signals, as standard CCA, TRCA can be viewed as finding subject-specific reference signals.

2.5.3. Calculating the number of bins

The first step in discretising the features is deciding the number of bins. This thesis used two methods for calculating the number of bins since different methods work well with different sample sizes. Different sample sizes occur because the introduced algorithm was tested on two different datasets. The Freedman Diaconis estimator [83] was used for larger sample sizes. This method estimates the number of bins by

$$2 \cdot \frac{\mathrm{IQR}}{\sqrt[3]{m}},$$

where m is the number of samples, and IQR is the interquartile range. For smaller sample sizes, Sturges' estimator [84] was used. This method estimates the number of bins by

$$\log_2 m + 1,$$

where m is again the number of samples. Note, however, that Sturges' rule has received criticism due to errors in the original derivation of the rule [85]. Using the rule gives reasonable results for a moderate sample size (which is the case here) due to a coincidence.

2.5.4. Results and comparison

The results on Dataset 1 were already presented in section 2.3.3. On Dataset 2, the introduced classifier was compared to the widely used arg max classifier (2.1). Tables 2 and 3 contain the results. These results are the average values over the sets of either 3 or 4 classes as described in Publication II [2]. The information-bottleneck parameters were set to $\alpha = 1$ and $\beta = 100$, as in Publication II [2][5]. The classifier parameters $N \in \{1, \ldots, 5\}$ were considered for Dataset 1, and $N \in \{1, 2\}$ were considered for Dataset 2. For both cases, values of t satisfying

$$(2N+1)^3 < t \leq (2N+1)^3$$

were considered.

Similarly, as for Dataset 1, the performance measures here were calculated slightly differently from Publication II [2]. In particular, the test data was treated

[5]For β, larger values were preferred since then it gives larger weight to the mutual information $\mathrm{I}(C,P)$ in the optimisation task. As noted in Publication II, the exact value of β did not seem to have much effect on the result. For α, only the values 1 and 0 were considered as these correspond to the most known cases of standard information bottleneck and DIB. Publication II achieved slightly better results for $\alpha = 1$.

Table 2: Introduced classifier compared to arg max classifier using three classes.

Subject	Introduced classifier					arg max classifier		
	ITR_{mi}	**ITR**	**Acc**	**MDT**	**No. pred.**	**ITR_{mi}**	**ITR**	**Acc**
1	76.2	**72.3**	0.888	1.04	26	73.0	69.1	0.863
2	90.4	**89.5**	0.99	1.03	26	88.9	86.3	0.974
3	89.1	**88.3**	0.990	1.05	26	88.2	84.9	0.973
4	90.9	89.3	0.983	1.01	27	91.5	**89.6**	0.983
5	93.5	92.7	0.992	1.00	27	94.0	**93.6**	0.996
6	88.1	**87.3**	0.991	1.06	26	85.0	80.7	0.962
7	81.2	**80.1**	0.988	1.14	24	83.6	79.4	0.955
8	80.4	**76.5**	0.956	1.05	26	74.7	68.2	0.919
9	71.2	**66.4**	0.940	1.14	24	69.2	61.2	0.893
10	82.4	**78.2**	0.954	1.03	26	76.7	70.8	0.926
11	61.0	**53.4**	0.878	1.12	24	56.1	47.5	0.83
12	89.0	**87.9**	0.987	1.04	26	88.5	85.6	0.974
13	86.3	**83.8**	0.976	1.04	26	85.0	80.6	0.959
14	91.3	**90.5**	0.991	1.02	27	89.3	86.9	0.977
15	71.7	**67.3**	0.946	1.15	23	64.7	57.3	0.872
16	67.9	**63.2**	0.925	1.14	24	62.8	55.7	0.868
17	78.1	**74.3**	0.955	1.07	25	71.4	64.4	0.902
18	67.2	**62.1**	0.925	1.17	23	64.2	56.8	0.869
19	29.2	**21.8**	0.725	1.42	19	27.2	19.0	0.64
20	83.7	**82.1**	0.984	1.1	25	82.8	78.1	0.953
21	69.3	**63.8**	0.920	1.12	24	65.4	57.5	0.871
22	91.6	89.6	0.982	1.0	27	91.4	**89.8**	0.985
23	75.1	**71.9**	0.951	1.15	24	74.5	68.3	0.913
24	88.2	**85.4**	0.974	1.01	27	87.2	83.9	0.969
25	91.0	**90.5**	0.992	1.03	26	89.8	87.4	0.981
26	90.4	**89.3**	0.987	1.03	26	89.8	87.3	0.978
27	92.1	**91.1**	0.989	1.00	27	90.4	88.5	0.981
28	86.6	**85.7**	0.989	1.07	25	86.3	83.0	0.966
29	67.5	**63.4**	0.937	1.20	22	62.8	55.3	0.866
30	79.4	**76.3**	0.962	1.09	25	76.6	70.3	0.925
31	93.4	92.5	0.992	1.00	27	94.3	**93.9**	0.996
32	93.0	**92.4**	0.995	1.01	27	91.7	90.4	0.987
33	27.3	15.0	0.623	1.25	22	27.3	**18.2**	0.628
34	90.8	**89.7**	0.99	1.03	26	89.1	86.6	0.977
35	78.1	**75.5**	0.968	1.12	24	73.5	66.8	0.915
Avg	79.5	**76.5**	0.949	1.08	25	77.3	72.7	0.921

Table 3: Introduced classifier compared to arg max classifier using four classes.

Subject	Introduced classifier					arg max classifier		
	ITR_{mi}	ITR	Acc	MDT	No. pred.	ITR_{mi}	ITR	Acc
1	93.8	**89.2**	0.883	1.13	32	91.9	85.0	0.844
2	110.8	**109.0**	0.987	1.06	34	112.0	107.5	0.969
3	111.9	**108.0**	0.973	1.03	35	110.9	105.5	0.965
4	112.8	107.6	0.963	1.0	36	115.0	**112.0**	0.977
5	116.4	113.3	0.981	1.01	36	118.3	**117.2**	0.994
6	110.4	**108.1**	0.985	1.06	34	106.2	98.9	0.948
7	107.7	**100.4**	0.953	1.02	35	104.8	97.5	0.942
8	98.9	**89.3**	0.923	1.02	35	94.4	83.5	0.9
9	85.5	73.5	0.878	1.08	33	85.7	**73.7**	0.859
10	99.8	**91.9**	0.932	1.06	34	95.4	85.5	0.9
11	76.2	**65.6**	0.864	1.19	30	70.8	57.2	0.789
12	112.1	**107.2**	0.967	1.01	36	111.5	106.6	0.967
13	105.9	**99.8**	0.958	1.03	35	106.0	99.3	0.945
14	113.9	**110.6**	0.98	1.02	35	113.4	109.8	0.975
15	89.3	**78.0**	0.893	1.07	34	81.2	68.7	0.839
16	80.5	**73.0**	0.91	1.23	29	79.5	67.4	0.834
17	92.4	**86.8**	0.950	1.18	31	88.7	77.3	0.872
18	83.1	**76.3**	0.921	1.25	29	81.1	69.1	0.838
19	35.6	**23.8**	0.670	1.52	24	33.7	20.8	0.566
20	107.0	**103.3**	0.976	1.07	34	102.7	94.7	0.934
21	81.8	**76.1**	0.938	1.33	27	82.7	70.8	0.841
22	115.7	**114.0**	0.99	1.02	35	114.8	111.6	0.98
23	92.2	**84.9**	0.929	1.15	31	94.0	84.6	0.893
24	108.3	**101.5**	0.955	1.02	35	107.3	100.7	0.95
25	114.2	**111.4**	0.981	1.02	35	110.0	104.9	0.96
26	112.8	**111.4**	0.988	1.05	34	113.0	108.8	0.972
27	116.0	**114.0**	0.988	1.01	36	114.2	111.0	0.977
28	110.0	102.8	0.955	1.0	36	109.8	**104.3**	0.962
29	76.8	**70.7**	0.927	1.37	26	74.9	61.4	0.806
30	97.4	**90.8**	0.941	1.11	33	94.9	84.9	0.898
31	113.8	110.7	0.971	1.02	35	118.8	**118.1**	0.995
32	115.6	112.1	0.978	1.01	36	115.7	**113.0**	0.984
33	32.3	16.5	0.577	1.55	23	37.1	**21.5**	0.565
34	113.2	**109.7**	0.978	1.02	35	112.2	107.6	0.969
35	96.3	**89.6**	0.943	1.11	32	92.4	81.5	0.891
Avg	98.3	**92.3**	0.931	1.11	33	97.0	89.2	0.9

as an online signal as described in section 2.3.3 instead of calculating the measures for a hypothetical independent dataset. For Dataset 2, a window length of 0.5 seconds was used. Abbreviation No. pred. in the table stands for the number of predictions made.

In order to get a more accurate estimate of MDT, a gaze shifting time has been added to MDT in the experiments on Dataset 2. The gaze shifting time is extra time added to take into account the time it takes to switch between targets. The gaze shifting time is considered to be about 0.5 seconds [8, 13, 86], which is added to the MDT. The gaze shifting time was not added for the experiments on Dataset 1 because the related work did not use it.

2.5.5. Statistical analysis

The statistical analysis of results obtained using Dataset 1 was already presented in section 2.3.4. Since Dataset 2 has more subjects than Dataset 1, using the t-test [81] does not require an extra assumption about the normality of the differences since this assumption is approximately fulfilled thanks to the central limit theorem. Furthermore, weaker tests like the sign test [82] that do not need the normality assumption can be used with this amount of data. Again, the tests are performed using 0.05 as the significance level[6].

Both tests, the t-test and the sign test were used with the alternative hypothesis that the ITR of the algorithm from Publication II [2] is larger than the ITR obtained with the arg max classifier (2.1) and with the null hypothesis that these ITRs are equal. Both tests show that the difference in ITR is highly significant. The resulting p-values are $1.4 \cdot 10^{-8}$ with t-test and $1.1 \cdot 10^{-5}$ with sign test for the experiments with three classes, and $1.0 \cdot 10^{-4}$ with t-test and $2.5 \cdot 10^{-4}$ with sign test for the experiments with four classes. Thus, although the average difference itself is not that large (3–4 bits/min), it is statistically highly significant.

[6]The minimal difference in ITR that can be detected at significance level 0.05 with a t-test with power 0.8 is $0.43\hat{\sigma}$, where $\hat{\sigma}$ denotes the sample standard deviation of the difference in ITR.

3. CHARACTERISING REINFORCEMENT-LEARNING AGENTS

This chapter gives an overview of the third contribution [3] of this thesis related to RL. In particular, Publication III [3] introduces an algorithm for calculating autonomy and other information-theoretic measures for RL agents. Section 3.1 first discusses the framework of interactions between the agent and the environment. Next, the information-measures themselves are introduced. PID is used to obtain more insight into the measures. Section 3.2 describes the introduced algorithm for calculating these measures in the limiting process of time step approaching infinity. Finally, some empirical results are given in section 3.3.

3.1. Background

This section gives the required background knowledge to discuss the introduced algorithm. First, the information-theoretic framework, in which Bertschinger et al. [35] introduced the information-theoretic measures, is discussed. The autonomy measures and non-trivial informational closure (NTIC) are introduced in section 3.1.2. Section 3.1.3 describes PID together with its application to the current use case. In particular, PID can be used to decompose the autonomy measures and NTIC.

3.1.1. Information-theoretic framework

This section describes the framework in which Bertschinger et al. [35] introduced the autonomy measures. In this framework, the agent and the environment are clearly separated. The agent's (internal) state and environment's state are modelled by sequences of random variables. Thus, let us denote the agent's state at any time step $n \in \{0,1,2,\dots\}$ as S_n and the environment's state as E_n.

In addition to the agent's state and environment's state, there are random variables for observations and motor actions, denoted by O_n and M_n, respectively. Observations model the information flow from the environment to the agent, and motor actions model the information flow from the agent to the environment. The values of S_n and O_n determine the distribution of S_{n+1}. Thus S_{n+1} cannot depend, for example, on E_{n+1}. Similarly, the distribution of E_{n+1} is determined by the values of E_n and M_n. See Figure 6a for a visualisation of the interactions.

Recall the notation S_n^i for the event $\{S_n = i\}$ for possible time step n and value i, and similarly, the notation E_n^j for possible value j. The agent's state and the environment's state are assumed to satisfy Markov property

$$\mathbf{P}(S_n^{i_n}, E_n^{j_n} \mid S_{n-1}^{i_{n-1}}, E_{n-1}^{j_{n-1}}) = \mathbf{P}(S_n^{i_n}, E_n^{j_n} \mid S_{n-1}^{i_{n-1}}, E_{n-1}^{j_{n-1}}, \dots, S_0^{i_0}, E_0^{j_0})$$

and are assumed to be conditionally independent

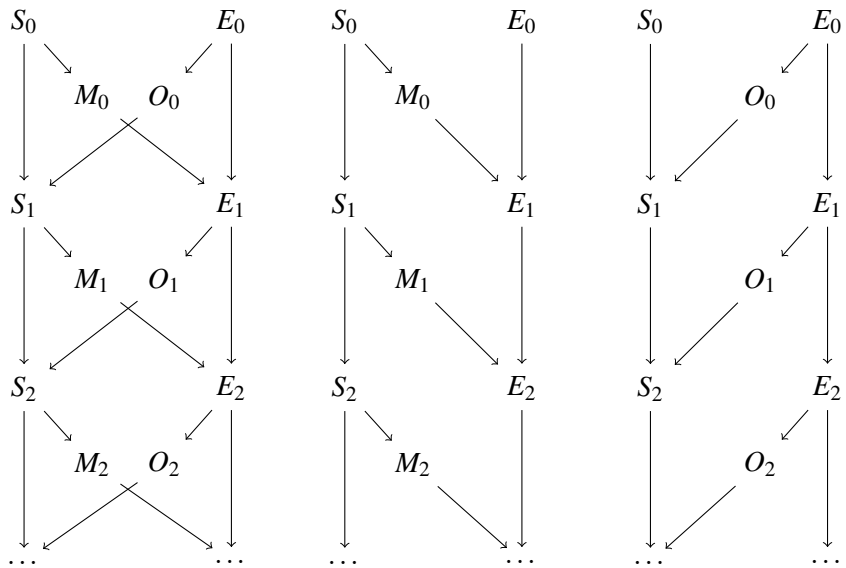
$$\mathbf{P}(S_n^{i_n}, E_n^{j_n} \mid S_{n-1}^{i_{n-1}}, E_{n-1}^{j_{n-1}}) = \mathbf{P}(S_n^{i_n} \mid S_{n-1}^{i_{n-1}}, E_{n-1}^{j_{n-1}})\,\mathbf{P}(E_n^{j_n} \mid S_{n-1}^{i_{n-1}}, E_{n-1}^{j_{n-1}}).$$

Thanks to these assumptions, the transition probabilities

$$\mathbf{P}(S_n^{i_n} \mid S_{n-1}^{i_{n-1}}, E_{n-1}^{j_{n-1}})$$
$$\mathbf{P}(E_n^{j_n} \mid S_{n-1}^{i_{n-1}}, E_{n-1}^{j_{n-1}})$$

fully determine the dynamics of the system. Let us assume that there are N agent's states and M environment's states. Then, in the homogeneous case, the transition probabilities can be given by two matrices of size $NM \times N$ and $NM \times M$, respectively. The whole system has a transition matrix of dimension $NM \times NM$.

In this thesis, it is assumed that transition probabilities are the same in each time step, thus the random process is homogeneous. In practice, the underlying models may not be homogeneous in an RL setting as the agent is learning and changing its strategy. However, it is assumed the Markov chain can be made homogeneous[1] by freezing the training and using fixed policy for the agent. Thus, in order to achieve homogeneity, the training process is frozen at different training steps, and the agent's behaviour is analysed for these training steps.



(a) General case. (b) The agent drives the environment. (c) The environment drives the agent.

Figure 6: Interactions between the agent and the environment.

3.1.2. Definition of autonomy and NTIC

Bertschinger et al. [35] introduced several interesting measures in this information-theoretic framework. They introduced two different autonomy measures. Let

[1] This is true, for example, if the setting is a MDP.

m denote a non-negative integer. The autonomy measure

$$A_m = \mathrm{I}(S_{n+1} : S_n \mid E_n, E_{n-1}, \ldots, E_{n-m})$$

corresponds to the case when the environment drives the agent, meaning that E_{n+1} depends only on E_n and not on M_n. These interactions are visualised in Figure 6c. The autonomy measure

$$A^* = \mathrm{I}(S_{n+1} : S_n)$$

corresponds to the case when the agent drives the environment, meaning that S_{n+1} depends only on S_n and not on O_n. These interactions are visualised in Figure 6b.

In addition to the autonomy measures, Bertschinger et al. [35] define

$$\mathrm{NTIC} = \mathrm{I}(S_{n+1} : E_n, \ldots, E_{n-m}) - \mathrm{I}(S_{n+1} : E_n \mid S_n).$$

In case $\mathrm{NTIC} > 0$, it shows how much the agent models the correlations in the environment. The other case, $\mathrm{NTIC} < 0$, refers to a synergistic situation where the agent's and environment's previous states together determine the agent's next state. For further discussion about these measures, refer to Bertschinger et al. [35] or Publication III [3].

3.1.3. Decomposing autonomy measures and NTIC

The autonomy measures and NTIC can be decomposed using PID. This decomposition gives a more refined look at these measures, possibly giving a better characterisation of the agent's behaviour. Decomposing the measures was introduced in Publication III [3] by coauthors of the paper. Note, however, that it had already appeared in the literature [87].

PID is a method for decomposing mutual information $\mathrm{I}(T : R_1, R_2)$ between random variables T and (R_1, R_2) into four components [36]. The random variable T is called the target and the random variables R_1 and R_2 are called the sources. The information of R_1 and R_2 about T is decomposed into four terms: SI, CI, and UI for each source variable. These terms can be described as follows:

- SI: redundant information in a sense that it can be obtained from both sources R_1 and R_2,
- $\mathrm{UI}(R_1)$: information that can be obtained only from R_1,
- $\mathrm{UI}(R_2)$: information that can be obtained only from R_2,
- CI: information that can be obtained synergistically from R_1 and R_2.

These terms satisfy the equations

$$\begin{aligned} \mathrm{I}(T : R_1, R_2) &= \mathrm{CI} + \mathrm{SI} + \mathrm{UI}(R_1) + \mathrm{UI}(R_2), \\ \mathrm{I}(T : R_1) &= \mathrm{SI} + \mathrm{UI}(R_1), \\ \mathrm{I}(T : R_2) &= \mathrm{SI} + \mathrm{UI}(R_2). \end{aligned}$$

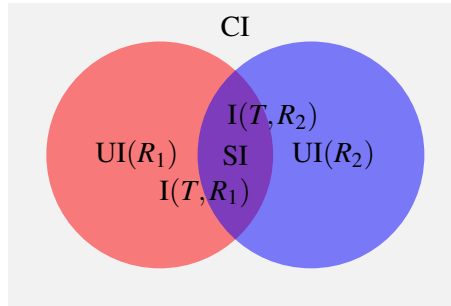


Figure 7: Visualisation of PID of mutual information $\mathrm{I}(T : R_1, R_2)$. Whole box represents the total mutual information. Discs represent the information provided by each of the sources.

The decomposition is visualised in Figure 7.

Applying PID with the target variable S_{n+1} and source variables S_n and E_n, the following decompositions of the autonomy measures and NTIC can be obtained

$$\begin{aligned} A_0 &= \mathrm{CI} + \mathrm{UI}(S_n), \\ A^* &= \mathrm{SI} + \mathrm{UI}(S_n), \\ \mathrm{NTIC} &= \mathrm{SI} - \mathrm{CI}. \end{aligned}$$

Furthermore, the PID terms themselves can help characterise the agent's behaviour. For example, $\mathrm{UI}(S_n)$ can be interpreted as showing how much the agent relies uniquely on its internal state, and $\mathrm{UI}(E_n)$ can be interpreted as showing how much the agent relies uniquely on the environment.

3.2. The third contribution: calculating autonomy, NTIC, and PID terms

Bertchinger et al. [35] introduced the autonomy measures and NTIC, but they did not give an explicit algorithm for calculating these measures. Here, an algorithm and a technique for combining the states are given. The homogeneity of the Markov chain is assumed, and the measures are calculated in the limiting process of time step approaching infinity. These assumptions are required to remove the time dependence of the measures. As already mentioned, homogeneity is achieved by stopping the training process.

3.2.1. Stationary distribution of the Markov Chain

Recall that we are working with a Markov chain $\{(S_n, E_n)\}_{n=0}^{\infty}$ with a finite state space. Applying the introduced algorithm is most natural in case the Markov chain is aperiodic since, in this case, a limiting distribution exists. The method can also be applied for a periodic chain, but in this case, the stationary distribution is a limiting distribution only for some initial distributions.

In the aperiodic case, it is possible to find the limiting distribution as follows. The Markov chain can be divided into a finite number of communicating classes. Each communicating class forms an irreducible Markov chain with a unique limiting distribution. These can be calculated separately for each class and combined together by weighting them with the probabilities of ending up in the corresponding communicating classes. This procedure gives a limiting distribution of the whole Markov chain for a fixed initial distribution.

The described method of finding the limiting distribution requires that the transition matrix P is known. In a practical RL setting, the transition matrix might not be known. Thus, the following stationary distribution

$$\mu(s,e) = \lim_n \frac{1}{n} \sum_{i=0}^{n-1} \mathbf{P}(S_i = s, E_i = e).$$

is used instead to handle the cases when P is not known, or the Markov chain is periodic. It has a simple interpretation: it is the expected proportion of time spent in each state in the long run since it is the expectation of

$$\lim_n \frac{1}{n} \sum_{i=0}^{n-1} I_{\{S_i=s,E_i=e\}},$$

where $I_{\{S_n=s,E_n=e\}}$ denotes the indicator function of the corresponding set. A natural estimate for $\mu(s,e)$ is given by sample average [88]

$$\hat{\mu}(s,e) = \frac{1}{n} \sum_{i=0}^{n-1} I_{\{S_i=s,E_i=e\}},$$

which can be calculated simply by finding the proportion of time spent in each state (s,e) for a sample of size n.

3.2.2. Calculating autonomy, NTIC, and PID terms

If the transition matrix P is known, μ is used as the distribution of (S_n, E_n) to calculate the information-theoretic measures. The distribution of (S_{n+1}, S_n, E_n) can then be obtained using the transition matrix P. Knowing the distribution of (S_{n+1}, S_n, E_n) is sufficient to calculate all the discussed measures.

Suppose the transition matrix P is not known. In that case, the distribution of (S_{n+1}, S_n, E_n) is estimated by calculating $\hat{\mu}(s', s, e)$ for each triple (s', s, e), where s and e denote the agent's and environment's current states and s' denotes the next agent's state. The sample average $\hat{\mu}(s', s, e)$ is calculated by naturally extending the definition of $\hat{\mu}(s,e)$ to triples (s', s, e). This method of estimating the stationary distribution, however, is expected to work reliably only for small state spaces.

Finally, it should be noted that there are multiple different PID measures. In Publication III [3], two different methods were used: $\widetilde{\mathrm{UI}}$ proposed by Bertschinger et al. [37] and $I_\cap^{sx}$ proposed by Makkeh et al. [6]. In the experiments presented here, only $I_\cap^{sx}$ is used.

3.2.3. Merging the agent's states

This section introduces a method for combining the agent's states that can be sometimes used. This merging is useful in case there are readily available states that could be used as the agent's states (called preliminary states), but are for some reason unsuitable, and combining these states results in more suitable states. Here, it is assumed that the transition matrix P is known.

Similar technique has been used in complex systems literature [89]. In Markov chain literature, the technique of combining states of a Markov chain and obtaining a new Markov chain is called lumping [90]. More precisely, here we are interested in weak lumpability in which case the new random process is required to be a Markov chain for only some initial distributions (similarly as we considered limiting distributions for only some initial distributions). There is a polynomial time algorithm available for determining whether weak lumping is possible [90]. Thus, if it is needed that the random process with combined states is a Markov chain, this should be checked with some available algorithm.

Let us now proceed with the details of the algorithm. Denote the random variables corresponding to the preliminary agent's state as L_n. Suppose a (non-injective) function f denotes the function that maps (ℓ, e) to $s = f(\ell, e)$, where ℓ is a preliminary state, e is an environment's state, and s is a more suitable state. Denote the inverse image of s when e is fixed by

$$H(s,e) = \{\ell \mid f(\ell,e) = s\}.$$

Then, the probability of being in the state (s,e) in the long run is

$$\begin{aligned}
&\mathbf{P}(f(L_{n+1},e') = s', E_{n+1} = e', f(L_n,e) = s, E_n = e) \\
&= \sum_{\ell' \in H(s',e')} \sum_{\ell \in H(s,e)} \mathbf{P}(L_{n+1} = \ell', E_{n+1} = e', L_n = \ell, E_n = e) \\
&= \sum_{\ell' \in H(s',e')} \sum_{\ell \in H(s,e)} \mathbf{P}(L_{n+1} = \ell', E_{n+1} = e' \mid L_n = \ell, E_n = e)\,\mathbf{P}(L_n = \ell, E_n = e).
\end{aligned}$$

Here the transition probabilities $\mathbf{P}(L_{n+1} = \ell', E_{n+1} = e' \mid L_n = \ell, E_n = e)$ are assumed to be known, and probabilities $\mathbf{P}(L_n = \ell, E_n = e)$ are taken to be the corresponding probabilities of the stationary distribution $\mu(\ell, e)$. Thus, the merging of states happens after finding the stationary distribution of the preliminary Markov chain.

Altogether, now the probabilities required for calculating the information-theoretic measures are known for the new states, and can be used in the subsequent analysis. Thus, the preliminary states L_n have successfully been replaced with more suitable states S_n. This merging is used only for the limiting/stationary situation.

3.2.4. Summary

The previous subsections described the setting considered and the algorithms introduced in Publication III [3]. Here, the goal is to characterise the behaviour of RL agents using available information-theoretic measures. Assuming that the environment and the agent interact as described in section 3.1.1, Bertschinger et al. [35] introduced information-theoretic measures of autonomy and NTIC. These measures can be further decomposed using PID [3]. The third contribution consists of an algorithm for calculating these measures in the limiting process of time step approaching infinity and an additional technique for merging the states. This merging of states also happens at the limit. Monitoring these information-theoretic measures during the training of an agent could give insights about whether the training is proceeding in the desired direction.

3.3. Empirical verification

In this section, the introduced method is applied to a specific RL setting. A simple setting is considered, in which the agent has to imitate the environment when the environment's state is visible and has to use its memory to reproduce the pattern when the environment's state is invisible. The environment is a repeating pattern of symbols which are visible in some iterations and invisible in others. This setting enables to analyse how the agent uses its memory. In Publication III [3], this environment was called repeating-pattern environment. Here, different perturbations are used than in Publication III [3].

3.3.1. Environment

In the current experiments, the environment is simply a repeating pattern of symbols, which can be visible or invisible to the agent. The repeating pattern is an ordered triple (p_1, p_2, p_3) of symbols where each symbol belongs to a fixed alphabet $\{1,2\}$. The pattern used in the experiments is $(p_1, p_2, p_3) = (2,1,1)$. In order to have invisible states, the possible observations for the agent are $\{0,1,2\}$ where the added 0 corresponds to invisibility. In particular, in case the pattern is visible, the agent observes the actual symbol as input, and when the pattern is invisible, the agent observes 0. Whether an iteration of the pattern is visible or invisible is decided randomly using a parameter h. With probability h, the pattern is invisible or hidden, and with probability $1-h$ the pattern is visible.

From the perspective of the information-theoretic framework, six environment's states are needed for this environment. Let us denote the states by v_1, v_2, v_3, i_1, i_2, i_3 where first three correspond to visible states and the last three to invisible states. These states can then be mapped to observations by a function O satisfying

$$O(v_j) = p_j$$
$$O(i_j) = 0.$$

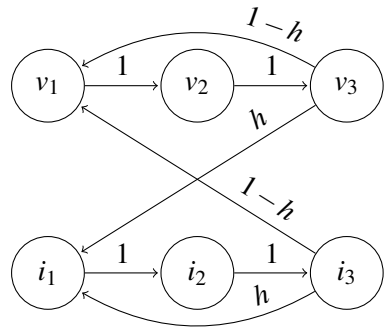


Figure 8: State transition diagram for the environment.

The transition diagram of the states is visualised in Figure 8.

3.3.2. Agent

The agent is controlled by NN in these experiments. Agent receives observations from the environment as an input and the set of possible actions is $\{1,2\}$. Agent gets rewarded by 1 if its action matches the current symbol of the pattern and punished by 0.9 otherwise. Note that since the pattern can be invisible, the agent cannot perfectly solve the task by simply deciding its action based on the observation. Thus, in order for the agent to be able to solve the task perfectly, it is provided with a memory in the form of an long short-term memory (LSTM) [91].

The NN controlling the agent consists of a linear layer of size 64 with ReLU activation, LSTM with 32 units as memory, and a linear layer to map the memory to the set of actions. The parameters of the network were trained using vanilla Deep-Q-Networks [92]. The NN was implemented and trained by a coauthor of Publication III [3]. See the corresponding publication for additional details.

From the perspective of information-theoretic framework, the LSTM's state was taken as the agent's state. Since there was no suitable estimator for PID in the case of continuous variables [3], the memory values were discretised by binning. Details of the binning are given in section 3.3.4.

3.3.3. Perturbations

In order to test whether the information-theoretic measures provide insights into the agent's behaviour, the agent trained in the original environment was put into slightly different environments. Two other environments were used where the observations were different than in the original environment. In these environments, with probability 0.2 the agent received an observation not corresponding to the current symbol. This means that if the current symbol is 2 the observation received by the agent is 1 and if the current symbol is 1 the observation is 2.

The two environments differ in the sense that in one of them the agent is rewarded for following the new observations (matching its actions with the observation) and in the other the agent is rewarded for ignoring the new observations

(following the original pattern). Agents with different learned strategies should be more successful in either one or the other new environment. An agent should be more successful in ignoring the perturbed observations if it relies more on its memory than the observations.

Reliance on memory and observations can be characterised by different PID terms when using PID for the triple (S_{n+1}, S_n, O_n) (note that this is different than in the case of decomposing information-theoretic measures discussed previously where the triple (S_{n+1}, S_n, E_n) was used). For example, $\mathrm{UI}(S_n)$ characterises how much the agent relies uniquely on previous memory state, while $\mathrm{UI}(O_n)$ characterises how much the agent relies uniquely on the observation. The remaining terms SI and CI show how much agent relies on S_n and O_n in shared or synergistic way.

3.3.4. Results

Recall that the environment has a parameter h corresponding to the probability that an iteration of the pattern is invisible to the agent. The agent was trained for different values of h as discussed in section 3.3.2. The values of h ranged from 0 to 1 with a 0.1 interval. The agent was trained for 30 different random initialisations, in each case for 63300 training steps. The training was frozen every 100 training steps to collect data for the analysis. With the training frozen, the agent was ran through 80 episodes, each lasting 30 time steps. At each of the 2400 time steps, the state of the LSTM was saved. Since there was no suitable estimator for PID in the case of continuous variables [3], the memory values were discretised into 15 equal length bins, with the smallest and the largest value in the collected data corresponding to the start of the first bin and end of the last bin, respectively.

The measure of success for the agent is taken as the fraction of correct actions by the agent over the last 80 episodes. Then the relationship between the success and the information-theoretic measures was analysed. It was of interest whether there is a monotonic relationship between these variables, thus Spearman correlations were calculated for different values of h and different random initialisations.

Figure 9 shows how the success is correlated to the suitable autonomy measure A_0 and NTIC for different values of h. The parameter h characterises how much the agent needs to use its memory to be successful. The measure NTIC shows how much the agent models the environment. As can be seen, as h increases, agent's with higher NTIC are more successful in the original environment. This is expected as modelling the environment is beneficial when observations do not provide enough information to the agent. For the autonomy measure A_0, for smaller values of h the agents with smaller autonomy are more successful. This is expected as the agent can be mostly following the observations and not exhibit autonomous behaviour. The correlation between h and success only becomes positive for the highest values of h. Thus, the measures indeed seem to correspond to our intuitive understanding of the measures.

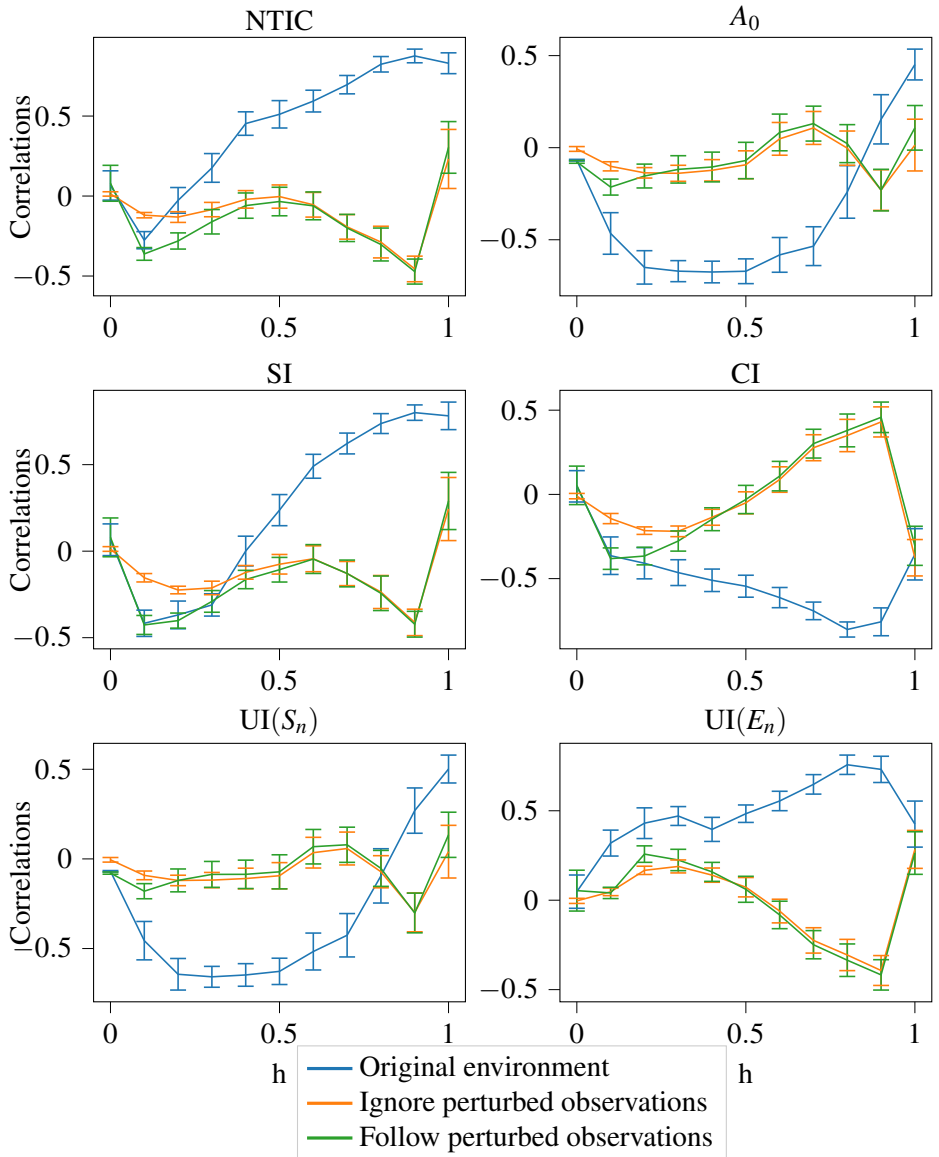


Figure 9: Correlations between PID terms and the success of the agent in different environments. PID is used for the triple (S_{n+1}, S_n, E_n). Agent was trained in the original environment and PID terms were calculated from the data obtained in the original environment. Success in other environments was calculated by putting the agent with training frozen into corresponding environments. Values correspond to the mean values obtained over the 30 seeds and error bars denote 95% confidence intervals (assuming that mean has normal distribution).

To better see differences between the perturbed environments, another analysis was performed. Figure 10 shows the correlations between the change in success

and the change in the corresponding PID term. This means that the correlations are calculated using the differenced series, that is the values are replaced with the differences of next and current value at each time step. Thus the correlations are between change in one value and change in the other value. Here, for PID the triple (S_{n+1}, S_n, O_n) is used instead of (S_{n+1}, S_n, E_n) since we are more interested in the observations than the environment's state. We see that agents that are uniquely depending more on their memory, indicated by a large $UI(S_n)$, are less successful if h is small. Correspondingly, the agents that are uniquely depending more on the observations, indicated by a large $UI(O_n)$, are more successful if h is small. Again, this is completely aligned by the expectations as our preliminary understanding was that h controls how much agent needs to use its memory. The

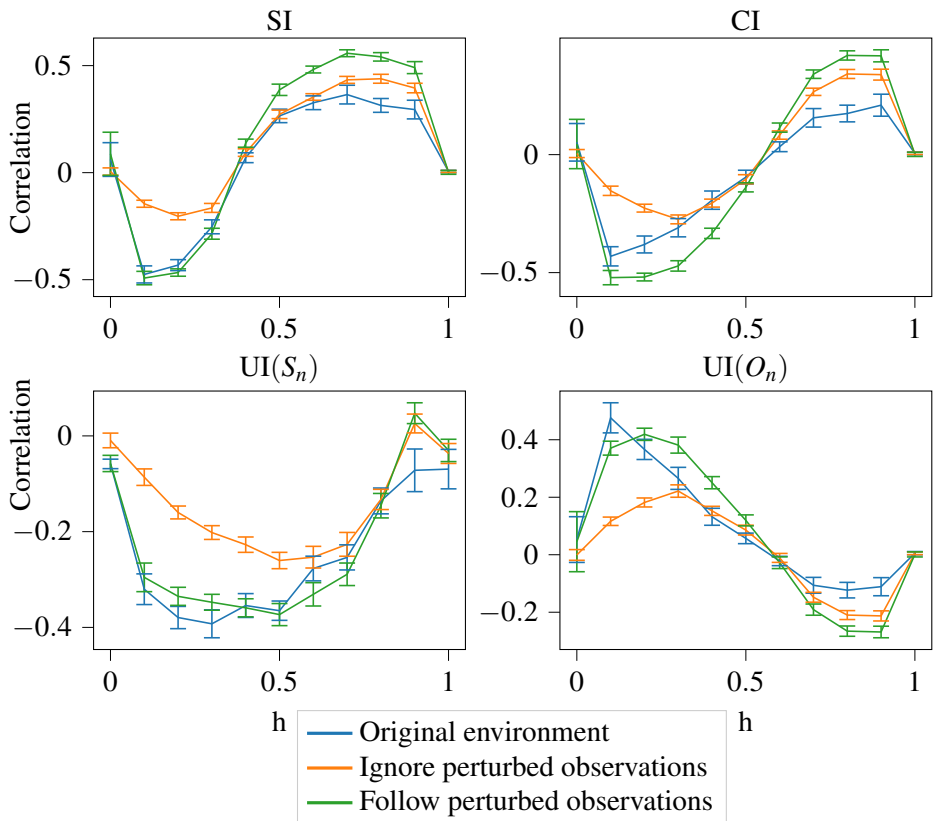


Figure 10: Correlations between changes in PID terms and changes in the success of the agent in different environments. PID is used for the triple (S_{n+1}, S_n, O_n). Agent was trained in the original environment and PID terms were calculated from the data obtained in the original environment. Success in other environments was calculated by putting the agent with training frozen into corresponding environments. Values correspond to the mean values obtained over the 30 seeds and error bars denote 95% confidence intervals (assuming that mean has normal distribution).

correlations are stronger in the perturbed environment where the agent has to follow the perturbations probably because in most cases the agent learned to follow any observation (perturbed or not) corresponding to visible state.

Overall, we can see clear patterns in the changes of the correlations, with most of them intuitively explainable and aligning with our expectations. This suggests that the information-theoretic measures could possibly be used to monitor whether the training is proceeding in the desired direction in terms of the behaviour of the agent. More results are presented in Publication III [3]. A future direction could be to try to optimise these measures to obtain agents behaving in desired ways.

4. CONCLUSION

In this thesis, algorithms for solving different tasks were introduced. The problems of classifying targets in BCIs and characterising the behaviour of RL agents were considered. The introduced algorithms make use of information theory. Most notably, the GIB [5] and PID [6] algorithms were used. The classification task for BCIs uses information theory to define the main performance measure for BCIs. In the case of RL agents, information theory is used to define the characterising measures.

4.1. Characterising reinforcement-learning agents

The characterisation of RL agents is based on the work of Bertschinger et al. [35] and PID measures [36, 37, 6]. Here, an algorithm is introduced for calculating the information-theoretic measures in an RL setting by considering the behaviour as the time step approaches infinity. A useful technique is introduced to merge states in the Markov chain to obtain more high-level states for the agent, possibly unlocking more practical applications. This work focused on the mathematical details and actually calculating the measures based on the definitions. Applications to a wider variety of practical situations are left as future work.

The results of the empirical experiments show that the measures indeed coincide with the intuitive understanding. For example, when successful agents are expected to model the environment, the corresponding information-theoretic measure is indeed correlated with the success of the agent in the performed experiments. Similarly, the measures provide a way of determining whether the agent relies more on their memory or the observations.

Typical use case of these measures could be monitoring them throughout the training of an RL agent. These measures could provide additional information about the agent. For example, it could happen that during the training the reward of the agent is not increasing during some time period, but at the same time there are changes in the information-theoretic measures. This would indicate that the agent is still learning something or at the very least changing its behaviour. Whether these changes are beneficial depends on the exact setting and the changes. Although the experiments here already provide evidence of the usability of the measures, verifying the usability of the measures in this exact setting is left as a future work.

Alternative future direction could be the following. Since the PID measure used here is differentiable with respect to the underlying probability measure, one could instead aim to optimise some of these measures instead of simply monitoring them. Exactly which measures should be optimised is again dependent on the considered setting. This could help in training agents with some required properties in their behaviour. For example, one could aim to obtain an agent that is the best (in the sense of having the highest NTIC) at modelling the environment.

4.2. Brain-computer interfaces

The classification algorithms for BCIs introduced in this thesis are, in some sense, optimal. Optimality of the classification rules was defined based on the main performance measure of BCIs, the amount of information they transfer. The condition for optimality was that the classification rule should maximise the ITR. This task was solved under some quite standard simplifying assumptions. Introduced algorithms give good results in practice, outperforming many other classification algorithms.

The first introduced algorithm solves the task by representing the performance measure as a function of classification thresholds. Then the optimisation task becomes a standard multivariable function optimisation. A similar approach could be useful in other settings as well for optimising parameters other than the thresholds. With this algorithm, the related work was outperformed by about 10 bits/min of ITR on average.

The second classification algorithm relies on tools of information theory, solving the task in a more general setting. The optimisation is essentially done over the space of classification functions. However, the second algorithm does not take explicitly into account the MDT and scales worse than the first algorithm. Compared to the widely used $\arg\max$ classification rule, the introduced algorithm gave a 3–4 bits/min increase in ITR on average (statistically highly significant difference).

To address the scaling problems of the second algorithm, as a future direction, either more efficient exact information bottleneck algorithm should be developed or an appropriate approximate algorithm should be used or developed. As mentioned, currently the approximate algorithms are only able to approximate a lower bound in the general setting [53].

Being able to choose at least a part of the BCI pipeline optimally is important to compare other parts of the pipeline too. A mathematically formulated optimality condition can be used for classification rules, as shown in this thesis, making it clear what the best rule should achieve. Moving backwards along the signal pipeline of a BCI, one can then extend this notion of optimality to other parts of the BCI, ultimately achieving an optimal behaviour for the whole pipeline. Extending the optimality from the classification rule to other parts is left as future work. In case the optimality cannot be achieved, then at least information theory provides tools to measure the amount of information kept at each step, guiding the choice of parts of the pipeline.

BIBLIOGRAPHY

[1] A. Ingel, I. Kuzovkin, and R. Vicente. Direct information transfer rate optimisation for SSVEP-based BCI. *Journal of Neural Engineering*, 16(1), 2018. DOI: 10.1088/1741-2552/aae8c7.

[2] A. Ingel and R. Vicente. Information Bottleneck as Optimisation Method for SSVEP-Based BCI. *Frontiers in Human Neuroscience*, 15, 2021. DOI: 10.3389/fnhum.2021.675091.

[3] A. Ingel, A. Makkeh, O. Corcoll, and R. Vicente. Quantifying Reinforcement-Learning Agent's Autonomy, Reliance on Memory and Internalisation of the Environment. *Entropy*, 24(3), 2022. DOI: 10.3390/e24030401.

[4] R. Nederpelt and H. Geuvers. *Type theory and formal proof: an introduction*. Cambridge University Press, 2014. ISBN: 9781107036505.

[5] D. Strouse and D. Schwab. The Deterministic Information Bottleneck. *Neural Computation*, 29(6):1611–1630, 2017. DOI: 10.1162/NECO_a_00961.

[6] A. Makkeh, A. J. Gutknecht, and M. Wibral. Introducing a differentiable measure of pointwise shared information. *Physical Review E*, 103(3):032149, 2021. DOI: 10.1103/PhysRevE.103.032149.

[7] J. W. Kay, J. M. Schulz, and W. A. Phillips. A comparison of partial information decompositions using data from real and simulated layer 5b pyramidal cells. *Entropy*, 24(8), 2022. DOI: 10.3390/e24081021.

[8] X. Chen, Y. Wang, M. Nakanishi, X. Gao, T.-P. Jung, and S. Gao. High-speed spelling with a noninvasive brain-computer interface. *Proceedings of the National Academy of Sciences*, 112(44):E6058–E6067, 2015. DOI: 10.1073/pnas.1508080112.

[9] M. Nakanishi, Y. Wang, T.-P. Jung, T. Tanaka, and M. Arvaneh. Spatial filtering techniques for improving individual template-based SSVEP detection. In *Signal Processing and Machine Learning for Brain-Machine Interfaces*, pages 219–242. IET, 2018. DOI: 10.1049/PBCE114E_ch.

[10] M. Doschoris and F. Kariotou. Mathematical foundation of electroencephalography. In P. Sittiprapaporn, editor, *Electroencephalography*, chapter 4. IntechOpen, Rijeka, 2017. DOI: 10.5772/68021.

[11] M. Cheng, X. Gao, S. Gao, and D. Xu. Design and implementation of a brain-computer interface with high transfer rates. *IEEE Transactions on Biomedical Engineering*, 49(10):1181–1186, 2002. DOI: 10.1109/TBME.2002.803536.

[12] Z. Lin, C. Zhang, W. Wu, and X. Gao. Frequency recognition based on canonical correlation analysis for SSVEP-based BCIs. *IEEE Transactions on Biomedical Engineering*, 54(6):1172–1176, 2007. DOI: 10.1109/TBME.2006.889197.

[13] M. Nakanishi, Y. Wang, X. Chen, Y. Wang, X. Gao, and T. Jung. Enhancing Detection of SSVEPs for a High-Speed Brain Speller Using Task-Re-

lated Component Analysis. *IEEE Transactions on Biomedical Engineering*, 65(1):104–112, 2018. DOI: 10.1109/TBME.2017.2694818.

[14] H. K. Lee and Y.-S. Choi. Enhancing SSVEP-based brain-computer interface with two-step task-related component analysis. *Sensors*, 21(4), 2021. DOI: 10.3390/s21041315.

[15] S. N. Carvalho, T. B. Costa, L. F. Uribe, D. C. Soriano, G. F. Yared, L. C. Coradine, and R. Attux. Comparative analysis of strategies for feature extraction and classification in SSVEP BCIs. *Biomedical Signal Processing and Control*, 21:34–42, 2015. DOI: 10.1016/j.bspc.2015.05.008.

[16] A. G. Yehia, S. Eldawlatly, and M. Taher. Principal component analysis-based spectral recognition for SSVEP-based Brain-Computer Interfaces. In *2015 Tenth International Conference on Computer Engineering Systems (ICCES)*, pages 410–415, 2015. DOI: 10.1109/ICCES.2015.7393085.

[17] S. F. Anindya, H. H. Rachmat, and E. Sutjiredjeki. A prototype of SSVEP-based BCI for home appliances control. In *2016 1st International Conference on Biomedical Engineering (IBIOMED)*, pages 1–6, 2016. DOI: 10.1109/IBIOMED.2016.7869810.

[18] M. Jukiewicz and A. Cysewska-Sobusiak. Implementation of Bilinear Separation algorithm as a classification method for SSVEP-based brain-computer interface. *Measurement Automation Monitoring*, 61(2):51–53, 2015.

[19] Y. Velchev, D. Radev, and S. Radeva. Features Extraction Based on Subspace Methods with Application to SSVEP BCI. *International Journal of Emerging Engineering Research and Technology*, 4(1):52–58, 2016.

[20] Z. Zhang, X. Li, and Z. Deng. A CWT-based SSVEP classification method for brain-computer interface system. In *2010 International Conference on Intelligent Control and Information Processing*, pages 43–48, 2010. DOI: 10.1109/ICICIP.2010.5564336.

[21] Y. Du, M. Yin, and B. Jiao. InceptionSSVEP: A Multi-Scale Convolutional Neural Network for Steady-State Visual Evoked Potential Classification. In *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, pages 2080–2085, 2020. DOI: 10.1109/ICCC51575.2020.9345194.

[22] J. J. Podmore, T. P. Breckon, N. K. N. Aznan, and J. D. Connolly. On the relative contribution of deep convolutional neural networks for SSVEP-based bio-signal decoding in BCI speller applications. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(4):611–618, 2019. DOI: 10.1109/TNSRE.2019.2904791.

[23] A. Ravi, N. H. Beni, J. Manuel, and N. Jiang. Comparing user-dependent and user-independent training of CNN for SSVEP BCI. *Journal of Neural Engineering*, 17(2):026028, 2020. DOI: 10.1088/1741-2552/ab6a67.

[24] H. Vu, B. Koo, and S. Choi. Frequency detection for SSVEP-based BCI using deep canonical correlation analysis. In *2016 IEEE International Con-*

ference on Systems, Man, and Cybernetics (SMC), pages 001983–001987, 2016. DOI: 10.1109/SMC.2016.7844531.

[25] N. K. Nik Aznan, S. Bonner, J. Connolly, N. Al Moubayed, and T. Breckon. On the classification of SSVEP-based dry-EEG signals via convolutional neural networks. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3726–3731, 2018. DOI: 10.1109/SMC.2018.00631.

[26] N. Waytowich, V. J. Lawhern, J. O. Garcia, J. Cummings, J. Faller, P. Sajda, and J. M. Vettel. Compact convolutional neural networks for classification of asynchronous steady-state visual evoked potentials. *Journal of Neural Engineering*, 15(6):066031, 2018. DOI: 10.1088/1741-2552/aae5d8.

[27] P. R. A. S. Bassi and R. Attux. FBDNN: filter banks and deep neural networks for portable and fast brain-computer interfaces. *Biomedical Physics & Engineering Express*, 8(3):035018, 2022. DOI: 10.1088/2057-1976/ac6300.

[28] Y. Wang, X. Chen, X. Gao, and S. Gao. A Benchmark Dataset for SSVEP-Based Brain-Computer Interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(10):1746–1752, 2017. DOI: 10.1109/TNSRE.2016.2627556. URL: http://bci.med.tsinghua.edu.cn/download.html.

[29] B. Liu, X. Huang, Y. Wang, X. Chen, and X. Gao. Beta: a large benchmark database toward ssvep-bci application. *Frontiers in Neuroscience*, 14, 2020. DOI: 10.3389/fnins.2020.00627.

[30] O. B. Guney, M. Oblokulov, and H. Ozkan. A deep neural network for SSVEP-based brain-computer interfaces. *IEEE Transactions on Biomedical Engineering*, 69(2):932–944, 2022. DOI: 10.1109/TBME.2021.3110440.

[31] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. Emergent tool use from multi-agent autocurricula, 2019. arXiv: 1909.07528 [cs.LG].

[32] O. Vinyals, I. Babuschkin, W. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. Agapiou, M. Jaderberg, and D. Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575:350–354, 2019. DOI: 10.1038/s41586-019-1724-z.

[33] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang. Dota 2 with large scale deep reinforcement learning, 2019. arXiv: 1912.06680 [cs.LG].

[34] A. Stooke, A. Mahajan, C. Barros, C. Deck, J. Bauer, J. Sygnowski, M. Trebacz, M. Jaderberg, M. Mathieu, N. McAleese, N. Bradley-Schmieg, N. Wong, N. Porcel, R. Raileanu, S. Hughes-Fitt, V. Dalibard, and W. M. Czarnecki. Open-ended learning leads to generally capable agents, 2021. arXiv: 2107.12808 [cs.LG].

[35] N. Bertschinger, E. Olbrich, N. Ay, and J. Jost. Autonomy: An information theoretic perspective. *Biosystems*, 91(2):331–345, 2008. DOI: 10.1016/j.biosystems.2007.05.018.

[36] P. L. Williams and R. D. Beer. Nonnegative decomposition of multivariate information, 2010. arXiv: 1004.2515 [cs.IT].

[37] N. Bertschinger, J. Rauh, E. Olbrich, J. Jost, and N. Ay. Quantifying unique information. *Entropy*, 16(4):2161–2183, 2014. DOI: 10.3390/e16042161.

[38] T. Froese and T. Ziemke. Enactive artificial intelligence: investigating the systemic organization of life and mind. *Artificial Intelligence*, 173(3):466–500, 2009. DOI: 10.1016/j.artint.2008.12.001.

[39] M. L. Anderson, T. Oates, W. Chong, and D. Perlis. The metacognitive loop i: enhancing reinforcement learning with metacognitive monitoring and control for improved perturbation tolerance. *Journal of Experimental & Theoretical Artificial Intelligence*, 18(3):387–411, 2006. DOI: 10.1080/09528130600926066.

[40] J. R. Wolpaw, H. Ramoser, D. J. McFarland, and G. Pfurtscheller. EEG-based communication: improved accuracy by response verification. *IEEE Transactions on Rehabilitation Engineering*, 6(3):326–333, 1998. DOI: 10.1109/86.712231.

[41] R. Durrett. *Probability: Theory and Examples*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2010. ISBN: 9780521765398.

[42] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[43] J. A. T. Thomas M. Cover. *Elements of information theory*. John Wiley & Sons, 2nd edition, 2006. ISBN: 9780471241959.

[44] T. B. da Silva Costa, L. F. S. Uribe, S. N. de Carvalho, D. C. Soriano, G. Castellano, R. Suyama, R. Attux, and C. Panazio. Channel capacity in brain-computer interfaces. *Journal of Neural Engineering*, 17(1):016060, 2020. DOI: 10.1088/1741-2552/ab6cb7.

[45] A. Klyubin, D. Polani, and C. Nehaniv. Empowerment: a universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 128–135, 2005. DOI: 10.1109/CEC.2005.1554676.

[46] B. Dal Seno, M. Matteucci, and L. T. Mainardi. The utility metric: a novel method to assess the overall performance of discrete brain–computer inter-

faces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(1):20–28, 2010. DOI: 10.1109/TNSRE.2009.2032642.

[47] V. Chandran. Time-varying bispectral analysis of visually evoked multichannel EEG. *EURASIP Journal on Advances in Signal Processing*, 2012(1), 2012. DOI: 10.1186/1687-6180-2012-140.

[48] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

[49] Y. Bengio, A. Courville, and P. Vincent. Representation learning: a review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. DOI: 10.1109/TPAMI.2013.50.

[50] R. Shwartz-Ziv and N. Tishby. Opening the black box of deep neural networks via information, 2017. arXiv: 1703.00810 [cs.LG].

[51] Z. Goldfeld and Y. Polyanskiy. The information bottleneck problem and its applications in machine learning. *IEEE Journal on Selected Areas in Information Theory*, 1(1):19–38, 2020. DOI: 10.1109/JSAIT.2020.2991561.

[52] S. Mukherjee. General information bottleneck objectives and their applications to machine learning, 2019. arXiv: 1912.06248 [cs.LG].

[53] A. Zaidi, I. Estella-Aguerri, et al. On the information bottleneck problems: models, connections, applications and information theoretic views. *Entropy*, 22(2):151, 2020. DOI: 10.3390/e22020151.

[54] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy. Deep variational information bottleneck. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[55] M. Wibral, V. Priesemann, J. W. Kay, J. T. Lizier, and W. A. Phillips. Partial information decomposition as a unified approach to the specification of neural goal functions. *Brain and Cognition*, 112:25–38, 2017. DOI: 10.1016/j.bandc.2015.09.004.

[56] M. Wibral, C. Finn, P. Wollstadt, J. T. Lizier, and V. Priesemann. Quantifying information modification in developing neural networks via partial information decomposition. *Entropy*, 19(9):494, 2017. DOI: 10.3390/e19090494.

[57] T. M. S. Tax, P. A. M. Mediano, and M. Shanahan. The partial information decomposition of generative neural network models. *Entropy*, 19(9):474, 2017. DOI: 10.3390/e19090474.

[58] M. Harder, C. Salge, and D. Polani. Bivariate measure of redundant information. *Physical Review E*, 87(1):012130, 2013. DOI: 10.1103/PhysRevE.87.012130.

[59] R. A. A. Ince. Measuring multivariate redundant information with pointwise common change in surprisal. *Entropy*, 19(7):318, 2017. DOI: 10.3390/e19070318.

[60] C. Finn and J. T. Lizier. Probability mass exclusions and the directed components of mutual information. *Entropy*, 20(11), 2018. DOI: 10.3390/e20110826.

[61] R. P. Stanley. *Enumerative Combinatorics*, volume 1. 2nd edition, 2011. ISBN: 9781107602625.

[62] K. Schick-Poland, A. Makkeh, A. J. Gutknecht, P. Wollstadt, A. Sturm, and M. Wibral. A partial information decomposition for discrete and continuous variables, 2021. arXiv: 2106.12393 [cs.IT].

[63] A. Kolchinsky. A novel approach to the partial information decomposition. *Entropy*, 24(3), 2022. DOI: 10.3390/e24030403.

[64] R. Zerafa, T. Camilleri, O. Falzon, and K. P. Camilleri. To train or not to train? A survey on training of feature extraction methods for SSVEP-based BCIs. *Journal of Neural Engineering*, 15(5):051001, 2018. DOI: 10.1088/1741-2552/aaca6e.

[65] H. Tanaka, T. Katura, and H. Sato. Task-related component analysis for functional neuroimaging and application to near-infrared spectroscopy data. *NeuroImage*, 64:308–327, 2013. DOI: 10.1016/j.neuroimage.2012.08.044.

[66] Y. Zhang, L. Dong, R. Zhang, D. Yao, Y. Zhang, and P. Xu. An Efficient Frequency Recognition Method Based on Likelihood Ratio Test for SSVEP-Based BCI. *Computational and Mathematical Methods in Medicine*, 2014, 2014. DOI: 10.1155/2014/908719.

[67] O. Friman, I. Volosyak, and A. Graser. Multiple Channel Detection of Steady-State Visual Evoked Potentials for Brain-Computer Interfaces. *IEEE Transactions on Biomedical Engineering*, 54(4):742–750, 2007. DOI: 10.1109/TBME.2006.889160.

[68] D. E. Thompson, L. R. Quitadamo, L. Mainardi, K. ur Rehman Laghari, S. Gao, P.-J. Kindermans, J. D. Simeral, R. Fazel-Rezai, M. Matteucci, T. H. Falk, L. Bianchi, C. A. Chestek, and J. E. Huggins. Performance measurement for brain-computer or brain-machine interfaces: a tutorial. *Journal of Neural Engineering*, 11(3):035001, 2014. DOI: 10.1088/1741-2560/11/3/035001.

[69] P. Yuan, X. Gao, B. Allison, Y. Wang, G. Bin, and S. Gao. A study of the existing problems of estimating the information transfer rate in online brain-computer interfaces. *Journal of Neural Engineering*, 10(2):026014, 2013. DOI: 10.1088/1741-2560/10/2/026014.

[70] H. Bakardjian, T. Tanaka, and A. Cichocki. Optimization of SSVEP brain responses with application to eight-command Brain-Computer Interface. *Neuroscience Letters*, 469(1):34–38, 2010. DOI: 10.1016/j.neulet.2009.11.039. URL: http://www.bakardjian.com/work/ssvep_data_Bakardjian.html.

[71] A. Azzalini and A. Capitanio. *The Skew-Normal and Related Families*. IMS monographs. Cambridge University Press, 2014. ISBN: 9781107029279.

[72] D. J. Wales and J. P. Doye. Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, 1997. DOI: 10.1021/jp970984n.

[73] B. Olson, I. Hashmi, K. Molloy, and A. Shehu. Basin hopping as a general and versatile optimization framework for the characterization of biological macromolecules. *Advances in Artificial Intelligence*, 2012. DOI: 10.1155/2012/674832.

[74] F. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978. DOI: 10.1109/PROC.1978.10837.

[75] A. F. Demir, H. Arslan, and I. Uysal. Bio-inspired filter banks for SSVEP-based brain-computer interfaces. In *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, pages 144–147, 2016. DOI: 10.1109/BHI.2016.7455855.

[76] M. Jukiewicz, M. Buchwald, and A. Czyż. Optimizing SSVEP-based brain-computer interface with CCA and Genetic Algorithms. In *2019 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pages 164–168, 2019. DOI: 10.23919/SPA.2019.8936758.

[77] P. Saidi, G. Atia, and A. Vosoughi. Detection of Visual Evoked Potentials using Ramanujan Periodicity Transform for real time brain computer interfaces. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 959–963, 2017. DOI: 10.1109/ICASSP.2017.7952298.

[78] A. F. Demir, H. Arslan, and I. Uysal. Bio-Inspired Filter Banks for Frequency Recognition of SSVEP-Based Brain–Computer Interfaces. *IEEE Access*, 7:160295–160303, 2019. DOI: 10.1109/ACCESS.2019.2951327.

[79] M. Jukiewicz, M. Buchwald, and A. Cysewska-Sobusiak. Finding Optimal Frequency and Spatial Filters Accompanying Blind Signal Separation of EEG Data for SSVEP-based BCI. *International Journal of Electronics and Telecommunications*, 64, 2018. DOI: 10.24425/123543.

[80] V. B. R. Karnati, U. S. G. Verma, J. S. Amerineni, and V. H. Shah. Frequency detection in Medium and High frequency SSVEP based Brain Computer Interface systems by scaling of sine-curve fit amplitudes. In *2014 First International Conference on Networks Soft Computing (ICNSC2014)*, pages 300–303, 2014. DOI: 10.1109/CNSC.2014.6906701.

[81] Student. The probable error of a mean. *Biometrika*:1–25, 1908.

[82] W. J. Dixon and A. M. Mood. The statistical sign test. *Journal of the American Statistical Association*, 41(236):557–566, 1946. DOI: 10.1080/01621459.1946.10501898.

[83] D. Freedman and P. Diaconis. On the histogram as a density estimator: L2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(4):453–476, 1981. DOI: 10.1007/BF01025868.

[84] H. A. Sturges. The choice of a class interval. *Journal of the american statistical association*, 21(153):65–66, 1926.

[85] R. J. Hyndman. The problem with sturges' rule for constructing histograms. *Monash University*:1–2, 1995.

[86] L. Liang, J. Lin, C. Yang, Y. Wang, X. Chen, S. Gao, and X. Gao. Optimizing a dual-frequency and phase modulation method for SSVEP-based BCIs. *Journal of Neural Engineering*, 17(4):046026, 2020. DOI: 10.1088/1741-2552/abaa9b.

[87] D. Krakauer, N. Bertschinger, E. Olbrich, J. C. Flack, and N. Ay. The information theory of individuality. *Theory in Biosciences*, 139(2):209–223, 2020. DOI: 10.1007/s12064-020-00313-7.

[88] K. B. Athreya and M. Majumdar. Estimating the stationary distribution of a markov chain. *Economic Theory*, 21(2/3):729–742, 2003.

[89] J. Zhang and K. Liu. Neural information squeezer for causal emergence. *Entropy*, 25(1), 2023. DOI: 10.3390/e25010026.

[90] L. Gurvits and J. Ledoux. Markov property for a function of a markov chain: a linear algebra approach. *Linear Algebra and its Applications*, 404:85–117, 2005. DOI: https://doi.org/10.1016/j.laa.2005.02.007.

[91] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[92] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*, 2013. arXiv: 1312.5602 [cs.LG].

ACKNOWLEDGEMENTS

I would like to thank the reviewers and opponents for their time and effort. The internal reviewer Sven Laur has not been mentioned elsewhere in this thesis and I would like to thank him for a thorough review.

I would also like to thank the Institute of Mathematics and Statistics of University of Tartu. The courses I took from the Institute of Mathematics and Statistics gave me the crucial theoretical understanding necessary to develop the algorithms introduced in this thesis.

SISUKOKKUVÕTE

Informatsiooniteooriat kasutavad algoritmid: klassifitseerimine aju-arvuti liidestes ja stiimulõppe agentide iseloomustamine

Informatsiooniteooria on matemaatika haru, mis on teoreetiliseks aluseks tänapäeva kommunikatsioonitehnoloogiatele. Samuti on informatsiooniteooria abil proovitud defineerida abstraktseid mõisteid nagu autonoomsus. Antud töös kasutataksegi äsja nimetatud informatsiooniteooria rakendusi masinõppes esile tulnud probleemide lahendamiseks. Kommunikatsioonivaldkonnast käsitleti aju-arvuti liideseid (AAL). AAL on otsene suhtluskanal kasutaja aju ja välise seadme vahel ning seega võimaldab "mõttejõul" seadmeid juhtida. Väliseks seadmeks võib olla näiteks arvuti või elektriline ratastool. AAL kasutab käskude tuvastamiseks kasutaja ajust pärit signaale, mida saab mõõta pea peale paigutatud elektroodidega. Teist rakendust, nimelt informatsiooniteoorial põhinevat autonoomsuse mõõtmist kasutati antud töös stiimulõppe agentide käitumise iseloomustamiseks. Stiimulõpe on teatavat tüüpi masinõpe, milles agent õpib keskkonnast saadud tagasiside põhjal.

Antud doktoritöös töötati välja kolm algoritmi erinevate masinõppes esile tulnud ülesannete lahendamiseks. Välja töötatud algoritmid põhinevad informatsiooniteooria tulemustel. AAL-ide puhul kasutatakse informatsiooniteooria vahendeid AAL-i informatsiooni edastamise kiiruse mõõtmiseks. Informatsiooni edastamise kiirus on peamine AAL-i efektiivsuse mõõdik. Informatsiooni edastamise kiirus võtab arvesse nii klassifitseerija täpsust, käskude arvu, kui ka klassifitseerimise kiirust. Tulemuseks on suurus, mis näitab mitu bitti informatsiooni minutis AAL edastab. Kui AAL on loodud mingi ühe kindla ülesande lahendamiseks, siis võib informatsiooni edastamise kiirus jääda liiga abstraktseks mõõdikuks, aga üldiseks AAL-ide võrdlemiseks on informatsiooni edastamise kiirus populaarseim mõõdik.

Informatsiooni edastamise kiiruse arvutusvalem pärineb informatsiooniteooriast, tegemist on erijuhuga vastastikusest informatsioonist ennustatud käsu ja tegeliku käsu vahel. Vastastikune informatsioon lihtsustub antud kujule ainult kui teatavad eeldused on täidetud. Kahjuks tuleb sageli kirjanduses ette situatsioone, kus neid eelduseid ignoreeritakse ning kasutatakse valemit olukordades, kus see pole põhjendatud. Antud töös juhime unustatud eeldustele tähelepanu ja kasutame efektiivsuse mõõdikuna üldisemat vastastikust informatsiooni, sest selle saab arvutada ka situatsioonides, kus informatsiooni edastamise kiirust kasutada ei tohiks. Üldisem meetod on vajalik, sest automaatselt optimaalset klassifitseerimisreeglit otsides võiksime muidu sattuda situatsiooni, milles informatsiooni edastamise kiiruse eeldused täidetud ei ole. Selles situatsioonis hinnataks efektiivsust valesti ning seetõttu võiksime saada mitteoptimaalse tulemuse.

Antud töö raames läbi viidud katsetes kasutati visuaalselt esilekutsutud potent-

siaalidel (VEP) põhinevat AAL-i, milles aju signaale mõõdetakse elektroentsefalograafia (EEG) seadmega, kuid välja töötatud teoreetilisi tulemusi saab rakendada ka teist tüüpi AAL-ides. VEP-il põhineva AAL-i tööpõhimõte seisneb selles, et kasutajale näidatakse erinevaid visuaalseid stiimuleid ning samal ajal loetakse EEG seadmega kasutaja ajutegevust. EEG signaalist on võimalik reaalajas kindlaks teha, millist visuaalset stiimulit kasutaja parasjagu vaatab. Arvuti ekraanil näidatakse kasutajale korraga mitut visuaalset stiimulit. Iga stiimul vastab ühele käsule, mille kasutaja saab arvutile edastada. Seega, kui kasutaja vaatab ühte stiimulit, siis sellele stiimulile vastav käsk edastatakse arvutile. Iga visuaalne stiimul arvuti ekraanil vilgub kindla sagedusega.

Töös kasutatakse kolme tuntud meetodit EEG singaalist vajaliku info eraldamiseks: võimsusspektri analüüs, kanooniline korrelatsioonanalüüs (KKA) ja ülesandega seotud komponentide analüüs. Võimsuspektri analüüs on traditsiooniline meetod signaalist sageduste kohta info eraldamiseks Fourier' pöörde abil. Selle meetodi miinuseks on, et mitmekanalilise EEG signaali korral rakendatakse seda iga kanali jaoks eraldi. KKA võimaldab kõiki kanaleid korraga analüüsida. Tegemist on meetodiga, mis võimaldab uurida korrelatsiooni mitmekanaliliste signaalide vahel. Valisime võimsusspektri analüüsi ja KKA meetodid, sest neid kasutati sarnastes eelnevates töödes ning seega nende meetodite kasutamine võimaldas otsest võrdlust varasemate tulemustega. Kolmandaks kasutatud meetodiks on hiljuti kasutusele võetud ning paremaid tulemusi saavutanud ülesandega seotud komponentide analüüsi meetod.

Tüüpiliselt kasutatakse nimetatud tunnuste eraldamise meetodeid nii, et iga käsu jaoks eraldatakse üks tunnus. Võimsusspektri analüüsi meetodi puhul on tegemist stiimuli sageduse võimsusega. Kanoonilise korrelatsioonanalüüsi meetodi puhul on tegemist EEG signaali ja stiimulile vastavate signaalide, mis on teatavate sagedustega sinusoidid, vahelise kanoonilise korrelatsiooniga. Enamik tunnuste eraldamise meetodeid VEP AAL-ide puhul eraldavadki iga käsu jaoks ühe tunnuse, kusjuures tunnuse väärtust tõlgendatakse usaldusväärsusena vastava käsu suhtes. Mida suurem tunnuse väärtus, seda tõenäolisemaks antud käsku peetakse.

Tüüpiline viis tunnuste põhjal käsu klassifitseerimiseks on valida käsk, millele vastav tunnus on suurim. Võib öelda, et selline lähenemine on muutunud VEP AAL kirjanduses traditsiooniliseks, kuid arvestades tänapäevaseid masinõppe algoritme, võiks antud töö autori arvates seda pidada ülelihtsustatuks. Parema klassifitseerimismeetodi leidmine oligi antud töö üks eesmärkidest.

AAL-ide jaoks töötati välja kaks algoritmi, millest üks leiab optimaalse klassifitseerimisreegli lävendi põhjal klassifitseerijate hulgast ning teine lahendab ülesande üldisemas situatsioonis, leides optimaalse klassifitseerimisreegli palju mitmekesisemast klassifitseerijate hulgast. Esimene algoritm põhineb mitme muutuja funktsioonide optimeerimisel ning teine kasutab informatsiooniteooriast pärit meetodit nimega informatsiooni pudelikael, mis võimaldab optimeerida üle funktsioonide hulga.

Lisaks teoreetilistele tulemustele sisaldab antud töö ka loodud algoritmide empiirilist testimist. Tulemused viitavad sellele, et uued algoritmid töötavad hästi ka praktikas, andes paremaid tulemusi kui standardsete masinõppe algoritmide otsene rakendamine ja kirjanduses tihti kasutuses olevad klassifitseerimisreeglid. Varasemate tulemustega võrdlemiseks kasutati antud töös avalikult kättesaadavaid andmestikke. Välja töötatud algoritmid saavutasid varasematest tulemustest kuni 10 bit/min kiirema informatsiooni edastamise kiiruse.

Stiimulõppe puhul keskenduti antud töös agentide autonoomsuse ja teiste käitumist iseloomustavate suuruste mõõtmisele. Antud töös tugineti olemasolevatele informatsiooniteooriast pärit definitsioonidele, mille abil saab mõõta agentide autonoomsust ja keskkonna internaliseeritust. Lisaks kasutati osalist informatsiooni lahutamise meetodit. Osaline informatsiooni lahutamine võimaldas anda kasutatud definitsioonidele üksikasjalikumad esitused ning seeläbi intuitiivsemad selgitused.

Stiimulõppe saab lihtsamatel juhtudel formaliseerida Markovi otsustusprotsessina. Antud töös seda formaliseeringut kasutatigi, kuna sel juhul leiduvad lihtsad treeningalgoritmid, mis koonduvad parima lahendini ning seega muudavad teoreetiliste küsimuste uurimise lihtsamaks. Vaatlesime ka juhtu, kus Markovi otsustusprotsess on vaid osaliselt vaadeldav, ning sel juhul kasutatisime treenimiseks tehisnärvivõrke.

Antud töö panus stiimulõppe valdkonnas on algoritm nende informatsiooniteooria suuruste mõõtmiseks protsessis, kus ajasamm läheneb lõpmatusele, see tähendab, et vaadeldi agendi käitumist piirjuhul. Algoritmi rakendati lihtsale Markovi otsustusprotsessile ning saadud tulemused olid kooskõlas intuitiivse arusaamaga autonoomsusest ja internaliseeritusest. Näiteks empiirilised tulemused näitasid, et üldiselt õppimise käigus agendi autonoomsus ja keskonna internaliseeritud kasvavad. Erandiks on juht, kus agent saavutab optimaalse strateegia, mis võib olla liialt piirav ning seega treeningu lõpus autonoomsus väheneb. Osaline informatsiooni lahutus võimaldas saadud tulemuste üksikasjalikumat analüüsi.

PUBLICATIONS

CURRICULUM VITAE

Personal data

Name:	Anti Ingel
Date of birth:	16.02.1993
e-mail:	antiingel@gmail.com

Education

2017–	University of Tartu, PhD studies, Computer Science
2015–2017	University of Tartu, Master's studies, Computer Science, *cum laude* diploma
2012–2015	University of Tartu, Bachelor's studies, Computer Science, *cum laude* diploma
2009–2012	Gustav Adolf Grammar School, Mathematics and English specialisation, graduated with silver medal
2000–2009	Aruküla Free Waldorf School

Employment

2019–2021	University of Tartu, Institute of Computer Science, Junior Research Fellow of Computational Neuroscience
2013–2017	University of Tartu, Teaching Assistant in different courses

Scientific work

Main fields of interest: machine learning, brain-computer interfaces

Obtained scholarships and awards

2015	Estonian Academy of Sciences, Contest for University Students, award
2015	Estonian Research Council, National Contest for University Students, diploma
2016	Rotalia Foundation scholarship
2017	Estonian-Revelia Academic Fund, Harald Raudsepp's scholarship
2012–2017	IT Academy scholarship

Other activities

2021–	Reviewer for IOP journals (obtained IOP trusted reviewer status, IOP Outstanding Reviewer award)
2017–2022	University of Tartu, Institute of Computer Science, Member of Programme council for Computer Science curriculum (Bachelor's and Master's level)

ELULOOKIRJELDUS

Isikuandmed

Ees- ja perekonnanimi:	Anti Ingel
Sünniaeg:	16.02.1993
e-mail:	antiingel@gmail.com

Haridus

2017–	Tartu Ülikool, doktoriõpe, informaatika eriala
2015–2017	Tartu Ülikool, magistriõpe, informaatika eriala, *cum laude* diplom
2012–2015	Tartu Ülikool, bakalaureuseõpe, informaatika eriala, *cum laude* diplom
2009–2012	Gustav Adolfi Gümnaasium, matemaatika-inglise keele klass, lõpetatud hõbemedaliga
2000–2009	Aruküla Vaba Waldorfkool

Teenistuskäik

2019–2021	Tartu ülikool, arvutiteaduse instituut, arvutusliku neuroteaduse nooremteadur
2013–2017	Tartu ülikool, praktikumide juhendamine erinevates ainetes

Teadustegevus

Peamised uurimisvaldkonnad: masinõpe, aju-arvuti liidesed.

Saadud stipendiumid ja auhinnad

2015	Eesti Teaduste Akadeemia üliõpilastööde konkursil auhind
2015	Eesti Teadusagentuuri üliõpilaste teadustööde riiklikul konkursil tänukiri
2016	Rotalia Foundation stipendium
2017	Estonian-Revelia Academic Fund'i Harald Raudsepa stipendium
2012–2017	IT Akadeemia stipendium

Muu tegevus

2021–	IOP ajakirjade retsensent (IOP trusted reviewer, IOP Outstanding Reviewer)
2017–2022	Tartu ülikool, arvutiteaduse instituut, informaatika bakalaureuse- ja magistriõppe programminõukogu liige

DISSERTATIONES INFORMATICAE PREVIOUSLY PUBLISHED IN DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS

19. **Helger Lipmaa.** Secure and efficient time-stamping systems. Tartu, 1999, 56 p.
22. **Kaili Müürisep.** Eesti keele arvutigrammatika: süntaks. Tartu, 2000, 107 lk.
23. **Varmo Vene.** Categorical programming with inductive and coinductive types. Tartu, 2000, 116 p.
24. **Olga Sokratova.** Ω-rings, their flat and projective acts with some applications. Tartu, 2000, 120 p.
27. **Tiina Puolakainen.** Eesti keele arvutigrammatika: morfoloogiline ühestamine. Tartu, 2001, 138 lk.
29. **Jan Villemson.** Size-efficient interval time stamps. Tartu, 2002, 82 p.
45. **Kristo Heero.** Path planning and learning strategies for mobile robots in dynamic partially unknown environments. Tartu 2006, 123 p.
49. **Härmel Nestra.** Iteratively defined transfinite trace semantics and program slicing with respect to them. Tartu 2006, 116 p.
53. **Marina Issakova.** Solving of linear equations, linear inequalities and systems of linear equations in interactive learning environment. Tartu 2007, 170 p.
55. **Kaarel Kaljurand.** Attempto controlled English as a Semantic Web language. Tartu 2007, 162 p.
56. **Mart Anton.** Mechanical modeling of IPMC actuators at large deformations. Tartu 2008, 123 p.
59. **Reimo Palm.** Numerical Comparison of Regularization Algorithms for Solving Ill-Posed Problems. Tartu 2010, 105 p.
61. **Jüri Reimand.** Functional analysis of gene lists, networks and regulatory systems. Tartu 2010, 153 p.
62. **Ahti Peder.** Superpositional Graphs and Finding the Description of Structure by Counting Method. Tartu 2010, 87 p.
64. **Vesal Vojdani.** Static Data Race Analysis of Heap-Manipulating C Programs. Tartu 2010, 137 p.
66. **Mark Fišel.** Optimizing Statistical Machine Translation via Input Modification. Tartu 2011, 104 p.
67. **Margus Niitsoo**. Black-box Oracle Separation Techniques with Applications in Time-stamping. Tartu 2011, 174 p.
71. **Siim Karus.** Maintainability of XML Transformations. Tartu 2011, 142 p.
72. **Margus Treumuth.** A Framework for Asynchronous Dialogue Systems: Concepts, Issues and Design Aspects. Tartu 2011, 95 p.
73. **Dmitri Lepp.** Solving simplification problems in the domain of exponents, monomials and polynomials in interactive learning environment T-algebra. Tartu 2011, 202 p.

74. **Meelis Kull.** Statistical enrichment analysis in algorithms for studying gene regulation. Tartu 2011, 151 p.
77. **Bingsheng Zhang.** Efficient cryptographic protocols for secure and private remote databases. Tartu 2011, 206 p.
78. **Reina Uba.** Merging business process models. Tartu 2011, 166 p.
79. **Uuno Puus.** Structural performance as a success factor in software development projects – Estonian experience. Tartu 2012, 106 p.
81. **Georg Singer.** Web search engines and complex information needs. Tartu 2012, 218 p.
83. **Dan Bogdanov.** Sharemind: programmable secure computations with practical applications. Tartu 2013, 191 p.
84. **Jevgeni Kabanov.** Towards a more productive Java EE ecosystem. Tartu 2013, 151 p.
87. **Margus Freudenthal.** Simpl: A toolkit for Domain-Specific Language development in enterprise information systems. Tartu, 2013, 151 p.
90. **Raivo Kolde.** Methods for re-using public gene expression data. Tartu, 2014, 121 p.
91. **Vladimir Šor.** Statistical Approach for Memory Leak Detection in Java Applications. Tartu, 2014, 155 p.
92. **Naved Ahmed.** Deriving Security Requirements from Business Process Models. Tartu, 2014, 171 p.
94. **Liina Kamm.** Privacy-preserving statistical analysis using secure multi-party computation. Tartu, 2015, 201 p.
100. **Abel Armas Cervantes.** Diagnosing Behavioral Differences between Business Process Models. Tartu, 2015, 193 p.
101. **Fredrik Milani.** On Sub-Processes, Process Variation and their Interplay: An Integrated Divide-and-Conquer Method for Modeling Business Processes with Variation. Tartu, 2015, 164 p.
102. **Huber Raul Flores Macario.** Service-Oriented and Evidence-aware Mobile Cloud Computing. Tartu, 2015, 163 p.
103. **Tauno Metsalu.** Statistical analysis of multivariate data in bioinformatics. Tartu, 2016, 197 p.
104. **Riivo Talviste.** Applying Secure Multi-party Computation in Practice. Tartu, 2016, 144 p.
108. **Siim Orasmaa.** Explorations of the Problem of Broad-coverage and General Domain Event Analysis: The Estonian Experience. Tartu, 2016, 186 p.
109. **Prastudy Mungkas Fauzi.** Efficient Non-interactive Zero-knowledge Protocols in the CRS Model. Tartu, 2017, 193 p.
110. **Pelle Jakovits.** Adapting Scientific Computing Algorithms to Distributed Computing Frameworks. Tartu, 2017, 168 p.
111. **Anna Leontjeva.** Using Generative Models to Combine Static and Sequential Features for Classification. Tartu, 2017, 167 p.
112. **Mozhgan Pourmoradnasseri.** Some Problems Related to Extensions of Polytopes. Tartu, 2017, 168 p.

113. **Jaak Randmets.** Programming Languages for Secure Multi-party Computation Application Development. Tartu, 2017, 172 p.
114. **Alisa Pankova.** Efficient Multiparty Computation Secure against Covert and Active Adversaries. Tartu, 2017, 316 p.
116. **Toomas Saarsen.** On the Structure and Use of Process Models and Their Interplay. Tartu, 2017, 123 p.
121. **Kristjan Korjus.** Analyzing EEG Data and Improving Data Partitioning for Machine Learning Algorithms. Tartu, 2017, 106 p.
122. **Eno Tõnisson.** Differences between Expected Answers and the Answers Offered by Computer Algebra Systems to School Mathematics Equations. Tartu, 2017, 195 p.

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

1. **Abdullah Makkeh**. Applications of Optimization in Some Complex Systems. Tartu 2018, 179 p.
2. **Riivo Kikas**. Analysis of Issue and Dependency Management in Open-Source Software Projects. Tartu 2018, 115 p.
3. **Ehsan Ebrahimi**. Post-Quantum Security in the Presence of Superposition Queries. Tartu 2018, 200 p.
4. **Ilya Verenich**. Explainable Predictive Monitoring of Temporal Measures of Business Processes. Tartu 2019, 151 p.
5. **Yauhen Yakimenka**. Failure Structures of Message-Passing Algorithms in Erasure Decoding and Compressed Sensing. Tartu 2019, 134 p.
6. **Irene Teinemaa**. Predictive and Prescriptive Monitoring of Business Process Outcomes. Tartu 2019, 196 p.
7. **Mohan Liyanage.** A Framework for Mobile Web of Things. Tartu 2019, 131 p.
8. **Toomas Krips.** Improving performance of secure real-number operations. Tartu 2019, 146 p.
9. **Vijayachitra Modhukur.** Profiling of DNA methylation patterns as biomarkers of human disease. Tartu 2019, 134 p.
10. **Elena Sügis.** Integration Methods for Heterogeneous Biological Data. Tartu 2019, 250 p.
11. **Tõnis Tasa.** Bioinformatics Approaches in Personalised Pharmacotherapy. Tartu 2019, 150 p.
12. **Sulev Reisberg.** Developing Computational Solutions for Personalized Medicine. Tartu 2019, 126 p.
13. **Huishi Yin.** Using a Kano-like Model to Facilitate Open Innovation in Requirements Engineering. Tartu 2019, 129 p.
14. **Faiz Ali Shah.** Extracting Information from App Reviews to Facilitate Software Development Activities. Tartu 2020, 149 p.
15. **Adriano Augusto**. Accurate and Efficient Discovery of Process Models from Event Logs. Tartu 2020, 194 p.
16. **Karim Baghery.** Reducing Trust and Improving Security in zk-SNARKs and Commitments. Tartu 2020, 245 p.
17. **Behzad Abdolmaleki.** On Succinct Non-Interactive Zero-Knowledge Protocols Under Weaker Trust Assumptions. Tartu 2020, 209 p.
18. **Janno Siim.** Non-Interactive Shuffle Arguments. Tartu 2020, 154 p.
19. **Ilya Kuzovkin.** Understanding Information Processing in Human Brain by Interpreting Machine Learning Models. Tartu 2020, 149 p.
20. **Orlenys López Pintado.** Collaborative Business Process Execution on the Blockchain: The Caterpillar System. Tartu 2020, 170 p.
21. **Ardi Tampuu.** Neural Networks for Analyzing Biological Data. Tartu 2020, 152 p.

22. **Madis Vasser.** Testing a Computational Theory of Brain Functioning with Virtual Reality. Tartu 2020, 106 p.
23. **Ljubov Jaanuska.** Haar Wavelet Method for Vibration Analysis of Beams and Parameter Quantification. Tartu 2021, 192 p.
24. **Arnis Parsovs.** Estonian Electronic Identity Card and its Security Challenges. Tartu 2021, 214 p.
25. **Kaido Lepik.** Inferring causality between transcriptome and complex traits. Tartu 2021, 224 p.
26. **Tauno Palts.** A Model for Assessing Computational Thinking Skills. Tartu 2021, 134 p.
27. **Liis Kolberg.** Developing and applying bioinformatics tools for gene expression data interpretation. Tartu 2021, 195 p.
28. **Dmytro Fishman.** Developing a data analysis pipeline for automated protein profiling in immunology. Tartu 2021, 155 p.
29. **Ivo Kubjas.** Algebraic Approaches to Problems Arising in Decentralized Systems. Tartu 2021, 120 p.
30. **Hina Anwar.** Towards Greener Software Engineering Using Software Analytics. Tartu 2021, 186 p.
31. **Veronika Plotnikova.** FIN-DM: A Data Mining Process for the Financial Services. Tartu 2021, 197 p.
32. **Manuel Camargo.** Automated Discovery of Business Process Simulation Models From Event Logs: A Hybrid Process Mining and Deep Learning Approach. Tartu 2021, 130 p.
33. **Volodymyr Leno.** Robotic Process Mining: Accelerating the Adoption of Robotic Process Automation. Tartu 2021, 119 p.
34. **Kristjan Krips.** Privacy and Coercion-Resistance in Voting. Tartu 2022, 173 p.
35. **Elizaveta Yankovskaya.** Quality Estimation through Attention. Tartu 2022, 115 p.
36. **Mubashar Iqbal.** Reference Framework for Managing Security Risks Using Blockchain. Tartu 2022, 203 p.
37. **Jakob Mass.** Process Management for Internet of Mobile Things. Tartu 2022, 151 p.
38. **Gamal Elkoumy.** Privacy-Enhancing Technologies for Business Process Mining. Tartu 2022, 135 p.
39. **Lidia Feklistova.** Learners of an Introductory Programming MOOC: Background Variables, Engagement Patterns and Performance. Tartu 2022, 151 p.
40. **Mohamed Ragab.** Bench-Ranking: A Prescriptive Analysis Approach for Large Knowledge Graphs Query Workloads. Tartu 2022, 158 p.
41. **Mohammad Anagreh.** Privacy-Preserving Parallel Computations for Graph Problems. Tartu 2023, 181 p.
42. **Rahul Goel.** Mining Social Well-being Using Mobile Data. Tartu 2023, 104 p.