Computer Science

Real-time and sample-efficient learning of computationally rational user models

Antti Keurulainen

Aalto University

DOCTORAL THESES

Aalto University publication series
DOCTORAL THESES 61/2024

Real-time and sample-efficient learning of computationally rational user models

Antti Keurulainen

A doctoral thesis completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall AS2 of the school on 26 March 2024 at 12:00.

Aalto University
School of Science
Computer Science

Supervising professor
Professor Samuel Kaski, Aalto University, Finland

Thesis advisor
Professor Samuel Kaski, Aalto University, Finland

Preliminary examiners
Professor Anna Rafferty, Carleton College, USA
Dr. Xiaoyang Wang, University of Exeter, UK

Opponent
Dr. John Williamson, University of Glasgow, UK

Aalto University publication series
DOCTORAL THESES 61/2024

ISBN 978-952-64-1731-8 (printed)
ISBN 978-952-64-1732-5 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-64-1732-5

Unigrafia Oy
Helsinki 2024

Finland

NORDIC SWAN ECOLABEL
Printed matter
4041-0619

Author
Antti Keurulainen

Name of the doctoral thesis
Real-time and sample-efficient learning of computationally rational user models

Publisher School of Science

Unit Department of Computer Science

Series Aalto University publication series DOCTORAL THESES 61/2024

Field of research Artificial Intelligence, Machine Learning

Manuscript submitted 27 February 2024 **Date of the defence** 26 March 2024

Permission for public defence granted (date) 21 February 2024 **Language** English

☐ **Monograph** ☒ **Article thesis** ☐ **Essay thesis**

Abstract

To effectively collaborate with humans, Artificial Intelligence (AI) systems must understand human behavior and the factors influencing it, including their goals, preferences, and abilities. Interactions with humans are typically costly, and in many real-life situations, AI must adapt to human behavior after only a few interactions. Additionally, when AI interacts with humans to learn about their behavior, the interactions need to be conducted without any noticeable delay for the human, which in turn necessitates adaptation in real-time.

This thesis investigates how an AI system can learn about other agents in a sample-efficient and real-time manner, using methods based on reinforcement learning. The first contribution of this thesis is a method for learning representations of goal-driven agents' behaviors with neural networks from incomplete observations, showing that they can be used for improving performance in cooperative decision-making tasks. The second contribution concerns the creation of an automated method for producing task distributions and related ground truth data for training a meta-learner to assess the skill level and adapt quickly to the behavior of a cooperating partner. The third contribution presents a novel method for designing informative experiments for estimating the parameters of simulation-based user models without closed-form likelihood functions, and which models are grounded in cognitive science. This method simultaneously amortizes the estimation of these parameters and the designing of experiments.

These contributions cover a wide range of settings where useful representations of behavior for improving cooperation are learned, along with the efficient learning of complex user models. The implications of the methods developed, as well as their strengths and limitations, are discussed.

Keywords Deep Learning, reinforcement Learning

ISBN (printed) 978-952-64-1731-8 **ISBN (pdf)** 978-952-64-1732-5

ISSN (printed) 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2024

Pages 142 **urn** http://urn.fi/URN:ISBN:978-952-64-1732-5

Tekijä
Antti Keurulainen

Väitöskirjan nimi
Reaaliaikaisia ja näytetehokkaita menetelmiä rationaalisten käyttäjämallien oppimiseen.

Julkaisija Perustieteiden korkeakoulu

Yksikkö Tietotekniikan laitos

Sarja Aalto University publication series DOCTORAL THESES 61/2024

Tutkimusala Tekoäly, Koneoppiminen

Käsikirjoituksen pvm 27.02.2024 **Väitöspäivä** 26.03.2024

Väittelyluvan myöntämispäivä 21.02.2024 **Kieli** Englanti

☐ **Monografia** ☒ **Artikkeliväitöskirja** ☐ **Esseeväitöskirja**

Tiivistelmä

Tehdäkseen tehokasta yhteistyötä ihmisten kanssa, tekoälyjärjestelmien on ymmärrettävä ihmisen käyttäytymistä ja sen taustalla vaikuttavia tekijöitä, kuten tavoitteita, mieltymyksiä ja kykyjä. Koneiden ja ihmisten väliset vuorovaikutukset tuottavat tyypillisesti vain rajoitetun määrän näytteitä, joten monissa käytännön tilanteissa tekoälyn on kyettävä ymmärtämään ihmisen tavoitteita ja käyttäytymistä vain muutaman interaktion perusteella. Lisäksi tekoälyn on kyettävä tekemään päättelyä ihmisen käyttäytymisestä reaaliaikaisesti ilman viiveitä, jotka saattaisivat häiritä koneen ja ihmisen välistä vuorovaikutusta.

Tässä väitöskirjassa tutkitaan, kuinka tekoälyjärjestelmä voi oppia havainnoimaan ja päättelemään toisen osapuolen käyttäytymisestä reaaliaikaisesti ja näytetehokkaasti hyödyntäen vahvistusoppimiseen perustuvia menetelmiä. Väitöskirjan ensimmäinen tulos on menetelmä, jossa neuroverkkojen avulla opitaan luomaan representaatioita toisen osapuolen käyttäytymisestä epätäydellisistä havainnoista. Näitä representaatioita voidaan edelleen käyttää parantamaan suorituskykyä yhteistyötä vaativissa päätöksentekotehtävissä. Toinen tulos on menetelmä, jossa luodaan automatisoidusti tehtäväjakauma sekä annotoituja näytteitä metaoppijan harjoittamista varten. Metaoppijan avulla tekoäly kykenee nopeasti arvioimaan toisen osapuolen taitotasoa ja mukauttamaan omaa käyttäytymistään yhteistyön parantamiseksi. Kolmas tulos on uusi menetelmä, jossa tekoäly kykenee tuottamaan spesifikaation informatiiviselle kokeelle simulointipohjaisen käyttäjämallin parametrien estimoimiseen asetelmassa, jossa kognitiotieteeseen perustuvan käyttäjämallin parametrit voidaan estimoida ilman uskottavuusfunktion suljetun muodon ratkaisua. Esitetyssä menetelmässä tuotetaan samanaikaisesti käyttäjämallin parametrien estimaatti sekä informatiivisen kokeen spesifikaatio neuroverkkojen avulla.

Väitöskirjassa kehitetyt menetelmät kattavat laajasti asetelmia, joissa tekoäly oppii tuottamaan representaatioita toisen osapuolen käyttäytymisestä parantaakseen yhteistyötä, sekä estimoimaan monimutkaisten käyttäjämallien parametreja. Väitöskirjassa arvioidaan myös kehitettyjen menetelmien hyödynnettävyyttä sekä luodaan katsaus kehitettyjen menetelmien vahvuuksista ja rajoituksista.

Avainsanat Syvät neuroverkot, vahvistusoppiminen.

ISBN (painettu) 978-952-64-1731-8 **ISBN (pdf)** 978-952-64-1732-5

ISSN (painettu) 1799-4934 **ISSN (pdf)** 1799-4942

Julkaisupaikka Helsinki **Painopaikka** Helsinki **Vuosi** 2024

Sivumäärä 142 **urn** http://urn.fi/URN:ISBN:978-952-64-1732-5

Preface

Throughout my career, I have been deeply fascinated by the interplay between humans and machines, particularly in how technologies can be harnessed to assist human learning. In 2015, nearly two decades after founding Bitville Oy—a company specializing in digital learning solutions for businesses—I became confident that artificial intelligence (AI) would present unparalleled new opportunities in this field. From that moment forward, my focus turned to rigorously investigating AI technologies, dedicated to uncovering how they could advance human-machine interactions and aid in human learning.

Having completed my licentiate thesis at Aalto University, I felt an increased commitment to my research field, which inspired me to advance to doctoral studies. In 2018, I approached Professor Samuel Kaski, who agreed to supervise and advise me throughout my doctoral journey. I am profoundly grateful for this opportunity. I would also like to extend my gratitude to Professor Aleksander Ilin, whose expertise in deep neural networks and deep reinforcement learning significantly enriched my understanding of these complex subjects. His guidance and supervision, in collaboration with Professor Samuel Kaski, were instrumental in the successful publication of my first two papers.

As my research evolved, it became crucial to apply the developed AI technologies and concepts towards understanding and improving the dynamics of human behavior and the collaboration between humans and machines. The Finnish Center for Artificial Intelligence (FCAI) features an extensive network of premier researchers in artificial intelligence and cognitive science. A pivotal moment in my studies came when Professor Andrew Howes, a specialist in cognitive science from the University of Birmingham, began mentoring our team through the FCAI collaboration. Professor Howes is a world-renowned expert in computational rationality, decision-making, and human behavior modeling. His participation and mentoring were instrumental in helping our research team understand how to integrate AI methods with human cognition. Our team was privileged to benefit from his expertise and dedication over several years.

I was fortunate to engage in numerous meetings, brainstorming sessions, and discussions with a variety of colleagues at Aalto University. I wish to express my gratitude to Jyri Kivinen, Vikas Verma, Mert Celikok, Tomi Peltola, and all the members of the Probabilistic Machine Learning (PML) group for their insightful and valuable contributions to these discussions. Furthermore, I had the pleasure of participating in many enriching conversations with Professor Antti Oulasvirta and the members of the Computational Behavior Lab at Aalto, led by Professor Oulasvirta.

I am also grateful to Isak Westerlund, who has worked alongside me during my PhD project, taking on a significant amount of responsibility in coding the applications and training the various AI models. My thanks also go to Ariel Kwiatkowski, who contributed to our second paper.

I am deeply grateful to the pre-examiners, Professor Anna Rafferty and Dr. Xiaoyang Wang, for their essential insights and constructive feedback on my thesis. In addition, I am thankful to Dr. John Williamson for willingly taking on the role of the opponent

I want to thank Bitville Oy CEO Jouko Ranta, as well as all my colleagues at Bitville Oy, for making it possible for me to concentrate practically full-time on the research projects during these years.

I had the unique opportunity to share my research journey with my son, Oskar, who has now embarked on his own PhD journey at Aalto University as I write this preface. Thank you for the hundreds of hours of brainstorming, discussion, and reflection over nearly the last 10 years, and for your invaluable contribution as a member of the research team. Our discussions have been of the utmost importance to me, whether they were about mathematical details or the extensive influence of AI on society.

Last but not least, I want to extend my heartfelt thanks to my daughter, Erika, and my wife, Sofia, for their encouragement and patience. I am aware that discussions about AI have prominently featured in our family conversations over the last few years. It is evident that this project could not have reached its conclusion without your unwavering support.

Espoo, February 27, 2024,

Antti Keurulainen

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Antti Keurulainen, Isak Westerlund, Samuel Kaski, Alexander Ilin. Learning to Assist Agents by Observing Them. In *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks*, Bratislava, Slovakia, pp. 519-530 September 2021.

II Antti Keurulainen, Isak Westerlund, Ariel Kwiatkowski, Samuel Kaski, Alexander Ilin. Behaviour-conditioned policies for cooperative reinforcement learning tasks. In *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks*, Bratislava, Slovakia, pp. 493-504, September 2021.

III Antti Keurulainen, Isak Westerlund, Oskar Keurulainen, Andrew Howes. Amortised Experimental Design and Parameter Estimation for User Models of Pointing. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, Hamburg, pp. 1-17, April 2023.

IV Antti Keurulainen, Isak Westerlund, Oskar Keurulainen, Andrew Howes. Amortised Design Optimization for Item Response Theory. In *International Conference on Artificial Intelligence in Education*, Tokyo, pp. 359-364, July 2023.

Author's Contribution

Publication I: "Learning to Assist Agents by Observing Them"

The doctoral candidate (DC) developed the main idea of the research and designed the experiments. DC wrote 70 percent of the code while Westerlund was responsible for the remaining 30 percent. The simulations were conducted by the DC and he also wrote the majority of the paper. Westerlund contributed by commenting on and suggesting improvements to the draft. Prof. Ilin and Prof. Kaski supervised the research.

Publication II: "Behaviour-conditioned policies for cooperative reinforcement learning tasks"

DC developed the main idea of the research and designed the experiments. DC wrote 50 percent of the code, while Westerlund and Kwiatkowski were each individually responsible for 25 percent. DC carried out the majority of the simulations and also wrote most of the paper. Westerlund participated in the paper writing process by providing comments and suggesting improvements. Supervision for the research was provided by Prof. Ilin and Prof. Kaski.

Publication III: "Amortised Experimental Design and Parameter Estimation for User Models of Pointing"

All authors, including DC, developed the main idea of the research and the experiments as equal contributors. The bulk of the code was written by Westerlund and O. Keurulainen, each providing 40 percent, while DC contributed the remaining 20 percent. Most of the simulations were carried out by DC. In terms of paper writing, DC took the most responsibility with

70 percent, while the rest was split equally among the other authors. Prof. Howes supervised the research.

Publication IV: "Amortised Design Optimization for Item Response Theory"

DC was mainly responsible for developing the main idea of the research and designing the experiments. The code was divided between Westerlund, who wrote 60 percent, and DC, who wrote the remaining 40 percent. Westerlund conducted the majority of the simulations (70 percent), with DC carrying out the remaining 30 percent. When it came to writing the paper, DC wrote the majority (80 percent) and the rest was split among all other authors. The research was supervised by Prof. Howes.

List of Figures

List of Tables

Abbreviations

ABC Approximative Bayesian Computation

ACT-R Adaptive Control of Thought—Rational

ANN Artificial Neural Network

AI Artificial Intelligence

BOED Bayesian Optimal Experimental Design

CNN Convolutional Neural Network

CTDE Centralised Training Decentralised Execution

DL Deep Learning

DNN Deep Neural Network

DRL Deep Reinforcement Learning

ECM Ensemble Cognitive Model

EIG Expected Information Gain

EPIC executive-process interactive control

GOMS Goals, Operators, Methods, Selection Rules

HCI Human-Computer Interaction

IRT Item Response Theory

KL Kullback-Leibler

LSTM Long Short Term Memory

MDP Markov Decision Process

Meta-RL Meta Reinforcement Learning

MHP Model Human Processor

MI Mutual Information

ML Machine Learning

MLE Maximum Likelihood Estimation

MLP Multilayer Perceptron

NLP Natural Language Processing

OED Optimal Experimental Design

POMDP Partially Observable Markov Decision Process

PPO Proximal Policy Optimiser

RL Reinforcement Learning

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

ToM Theory of Mind

VI Variational Inference

Symbols

α Learning rate

A Action space in Markov Decision Process

$a(t)$ Input to the activation function, action at time step t

b Bias in neural network, difficulty of an item in IRT

d Design

γ Discount rate in Markov Decision Process

$h(t)$ Hidden layer output at time t

H Horizon in Markov Decision Process

e Embedding

E Expected value

f Function, neural network

f_ϕ neural network parametrised by ϕ

$g(.)$ Activation function

I Mutual Information, Input image to the convolutional neural network

$J(\pi)$ Objective function in Reinforcement Learning

K Kernel in convolutional neural network

L Loss function, objective function

N Normal distribution or number of samples

∇ Gradient operator

$p(x, y)$ Joint probability of x and y

$o(t)$ Output at the time t, input object in relational networks

O Set of objects in relational networks

P Transition function in Markov Decision Process

ψ parameters of neural network

R Reward function

ρ_0 Initial state distribution

σ^2 Variance

S State space in Markov Decision Process

$S(i,j)$ Feature map in position i,j

τ State action trajectory $(s_0, a_0, s_1, a_1, ...)$

θ User parameter

W, w The learnable parameters in the neural network

y Ground truth, label, or target value

$\hat{y}$ Predicted value, or output from a neural network

1. Introduction

In recent years, Artificial Intelligence (AI) systems have made significant strides in acquiring advanced cognitive and motor skills, enabling them to perform increasingly complex tasks [6, 86, 94, 128, 22]. As these systems continue to evolve, their ability to understand human behavior, adapt to individual preferences, and cooperate effectively with humans and other intelligent systems becomes crucial [77, 31].

Creating AI systems that cooperate with humans by adapting to human behavior and preferences is an ambitious and challenging goal. For this purpose, AI systems require several non-trivial capabilities such as assessing the abilities of other agents, inferring their mental states, understanding their goals and ambitions, communicating intentions and beliefs, and adapting to norms [31, 11, 111]. An additional challenge emerges because AI methods, particularly those based on machine learning, typically necessitate a substantial amount of data for their learning processes. Specifically, deep neural networks and reinforcement learning, the technologies driving many of the recent advances in machine learning, are known to be data-hungry. However, when AI interacts with humans, these interactions do not usually generate extensive amounts of data. This creates a need for utilising efficiently existing data sources and automating the generation of useful training data. One strategy to facilitate AI in learning about human behavior is to construct computational user models that incorporate domain knowledge [7, 23, 67, 26, 95]. These models can be parameterized so that a range of parameters would cause these user models to produce a range of behaviors close enough to human behavior. Once these user models are available, AI can use them to learn about human-like behaviour and interact with them so that every interaction generates a maximal amount of information about the other party's behavior. "Learning a user model" corresponds to inferring the variables that best explain the observed behavior. This in turn can be used to enhance the understanding of human behavior and improve cooperative decision making.

Another challenge is the requirement of interacting with humans with-

out noticeable delays, necessitating real-time calculations. Recent work has suggested employing amortized methods, where the computational burden is shifted to training a neural network using prior information about the data-generating process [69, 47, 39, 61]. Consequently, real-time calculations can be achieved by executing forward passes through the neural network without the need for further neural network parameter updates. In this thesis, the emphasis is on leveraging amortized methods by training deep neural networks to address the challenge of real-time requirements. Specifically, when a user interacts with the system, each interaction generates new data points to be used for further processing. By employing neural networks that have been pre-trained on relevant simulated data, the system is equipped to rapidly process these new data points. Hence, the use of amortized methods via deep neural networks makes it feasible to achieve seamless interactions, characterized by the absence of noticeable delays.

While there is active research advancing the scenarios above, there are still numerous open problems and challenges to resolve. This research aims to develop novel methods based on deep reinforcement learning for real-time and sample-efficient learning of computationally rational user models. This motivates the three research questions for this dissertation.

Research Question 1: *How can existing data sources of user behavior be utilized for learning representations which can improve the performance of a cooperative decision making policy implemented by a deep reinforcement learning agent?*

The contribution of the first publication, "Learning to Assist Agents by Observing Them", is a method for learning low-dimensional representations of the behavior of a goal-driven agent from past partial trajectories, where the representations are shown to be useful for a cooperative decision-making task. In contrast to previous work, this work shows that the ability to quickly infer useful representations can be learned by only utilizing existing data sources without any knowledge about the eventual cooperative task or transition dynamics, meaning that by using meta-learning, similar capabilities to inverse reinforcement learning (IRL) [1, 8] arise spontaneously. A helper AI agent, with the same state space but a different action space than the goal-driven agent, infers a latent variable that explains the behavior of a goal-driven agent and learns useful assistive actions in a gridworld environment. This work is formulated as meta-reinforcement learning, as the goal was to build the capability to adapt quickly to an unseen situation by observing several partial trajectories of the other party. By conditioning the policy with several examples of the partial episodes, the meta-learning formulation allows to optimise the performance over several episodes to achieve a fast adaptation.

While research question 1 addresses the challenge of a limited amount of

data by utilising previously collected data, research question 2 considers the possibility of automating the generation of training data:

Research Question 2: *How can RL-based training methods for ad-hoc cooperation be generalised to settings where the partnering agents have varying skill levels?*

In the second publication, "Behaviour-conditioned policies for cooperative reinforcement learning tasks", a novel method is introduced for zero-shot co-operation with unknown partners. The key idea is a self-play mechanism for synthetically generating a task distribution and associated ground truth data such that it can be used for training a meta-learner for fast adaptation to the behavior of a partner in cooperative tasks. A population of agents with various skill levels is generated, where two agents cooperate in a shared environment. One agent is also trained to assess the skill level quickly during the first steps of the episode and learns to adjust its own behavior to improve cooperation. It is shown that cooperation with an unknown partner agent type can be improved by using this method.

While resolving the research questions above offers building blocks towards reasoning about human behavior, they consider cooperation between two AI agents in abstract gridworld tasks and using fully rational user models without any bounds. A natural next step is to use insights from cognitive science to apply similar techniques to human-like behavior, which can be modelled through rationally bounded user models. This motivates the third research question:

Research Question 3: *How can the ability to design informative experiments for learning of complex user models in a real-time manner be trained in-silico with deep reinforcement learning?*

The third publication, "Amortised Experimental Design and Parameter Estimation for User Models of Pointing," uses a learned computationally rational user model capable of mimicking human behavior related to mouse pointing and human gaze. The publication introduces a novel method where user model parameters can be inferred in real-time in a sample-efficient manner by amortizing simultaneously the design selections and parameter estimations.

The fourth publication, "Amortised Design Optimization for Item Response Theory," extends the work to closed-form user models and for assessing student capabilities in the Item Response Theory (IRT) framework. This publication introduces a novel method that can use existing item response data and learns to select the most informative items for the synthetic students from the existing bank of items.

Resolving these research questions above offers new valuable techniques towards cooperative AI and developing intelligent interactive systems, as

it allows AI to learn about human-like behavior with limited data in an interactive setting.

1.1 Outline of the thesis

This dissertation is comprised of a summary part and four publications. In the summary part, an introduction is first presented, where the general setting, research questions, and the relationship of the publications to the research questions are described. In Chapter 2, Deep Learning is introduced, and the Deep Learning architectures and methods present in the publications are discussed. In Chapter 3, Deep Reinforcement Learning (DRL) is introduced and the Reinforcement Learning techniques relevant to this thesis are discussed. Chapter 4 delves into user models, introducing the concepts of rule-based, closed-form and computationally rational user models. Chapter 5 introduces Optimal Experimental Design (OED), which is a key technology in this thesis. Chapter 6 provides a summary of all four publications. Finally, Chapter 7 offers conclusions.

2. Deep Learning

All four publications rely on Deep Learning (DL) methods, which are important technologies in the context of this thesis. In this chapter, Deep Learning technology is introduced, and the DL architectures that play a significant role in the publications are presented.

2.1 Introduction to Deep Learning

Deep Learning is a class of Machine Learning (ML) algorithms that utilize Deep Neural Networks (DNN), which are nested sequences of functions consisting of multiple hidden layers of neurons between the input and output layers. In contrast to traditional machine learning methods, many Deep Learning approaches can be considered as representation learning algorithms, where abstractions at various levels are learned automatically during the training process without the need for human-driven feature engineering [45, 78]. Deep Learning has a long history, with significant milestones towards modern Deep Neural Networks including the idea of learning representations by backpropagating errors in 1986 by Rumelhart et al. [121], and various subsequent proposals for creating layers of feature detectors in an unsupervised manner by using reconstruction as an objective for learning representations [51, 16].

The year 2012 marked a significant turning point in Deep Learning when a solution utilizing Deep Learning won the computer vision competition 'ImageNet' by a considerably large margin [74]. Since then, remarkable breakthrough applications using DL have emerged in various fields such as medicine [64, 27], robotics [82, 49, 94, 3], speech synthesis [100, 138], speech recognition [113, 9], computer vision [74, 129, 112], image generation [117, 120, 46], text generation with Large Language Models [22, 28, 124], 3D modeling [91], gaming [93, 128, 125, 17, 21], and Natural Language Processing (NLP) [33, 114].

2.2 Supervised Learning and self-supervised learning

In this section, the concepts of supervised and self-supervised learning are presented, as they are employed in Publications I, II, and III. The fundamental concepts of deep learning are introduced alongside the introduction of supervised learning.

2.2.1 Supervised Learning

The supervised learning method has traditionally been the most common training approach in machine learning. Supervised learning approaches based on deep learning typically require a large number of data points with appropriate *labels* (classification) or *targets* (regression). During training, the machine learning system processes input data samples and generates output. Using a predefined objective function, the output is compared with the desired outcome. In deep learning, the error signal from this comparison is backpropagated into the neural network to adjust the parameters defining the mapping from the input to the output of the network.

In supervised learning, a loss (or objective) function is employed to compute a numerical scalar value representing the error of the output. The choice of the loss function depends on the specific application. One example of a loss function is the L2 loss, which is defined as

$$L(W) = E_{p(y,x)}||y - f_W(x)||_2^2. \tag{2.1}$$

In this equation, $E_{p(y,x)}$ denotes the expectation under the joint probability distribution of y (true output) and x (input), representing the average error across all samples in the data. $f_W(x)$ is the predicted output generated by the neural network with parameters W for a given input x. The L2 norm (Euclidean distance) between the true output y and the predicted output $f_W(x)$ is calculated by finding the squared difference between the corresponding elements of y and $f_W(x)$ and summing up these squared differences. In practice, the loss is approximated by sampling items from a dataset and computing

$$L(W) \approx \frac{1}{N}\sum_{i=1}^{N}||y_i - f_W(x_i)||_2^2, \tag{2.2}$$

where N is the number of samples in the dataset and x_i, y_i are sampled from the data generating process $p(x, y)$.

The L2 loss is also related to maximum likelihood estimation. By assuming that the true output y is distributed normally around the predicted output $f_W(x)$ with variance σ^2, the likelihood of observing the true output y given the input x can be expressed as

$$p(y|x, W, \sigma^2) = N(y|f_W(x), \sigma^2) \tag{2.3}$$

The log-likelihood is proportional to the squared L2 norm (Euclidean distance) between the true output y and the predicted output $f_W(x)$:

$$\log p(y|x, W, \sigma^2) \propto -||y - f_W(x)||_2^2. \tag{2.4}$$

The proportionality constant is determined by the variance σ^2.

In the context of maximum likelihood estimation, the objective is to find the parameters W that maximize the likelihood of the observed data. Equivalently, since the logarithm is a monotonically increasing function, we can maximize the log-likelihood. As the log-likelihood is proportional to the squared L2 norm, minimizing the L2 loss (as shown in the previous formulas) is equivalent to maximizing the log-likelihood.

Once the error signal is obtained, it can be utilized in an optimization algorithm that updates the parameter matrix W_l in each layer l of the deep neural network. The optimization process involves calculating the gradient of each parameter with respect to the error. The basic optimization algorithm, Stochastic Gradient Descent (SGD), can be expressed as

$$W \leftarrow W - \alpha \nabla_W L(W), \tag{2.5}$$

where α represents the learning rate, which controls the step size for each parameter update during the training process.

2.2.2 Self-supervised learning

The bottleneck in the supervised learning method is typically the cost of acquiring a large enough amount of human-labelled data. This issue can be addressed if labels or targets can be extracted automatically from unlabelled data. A well-known example is the training of language models, where the system can be trained by predicting the next word or a hidden word in the existing text. This method capitalizes on the inherent structure of language, where subsequent or surrounding words provide contextual cues, thereby eliminating the need for external labels. Extending beyond natural language processing, self-supervised learning has found utility across a broad spectrum of applications [63, 87]. Publication I exemplifies its application in a reinforcement learning setting. In this context, where an agent is pretrained by predicting the next action of the other agent and use this capability to improve co-operation.

2.3 Deep Learning Architectures

The publications employ multilayer perceptrons (Publications I, II, III, IV), convolutional networks (Publication I), and recurrent networks (Publication II). Each of these architectures is briefly introduced in this chapter. Additionally, a specific network structure called *relation network* is discussed, as it plays an important role in Publications II and III.

2.3.1 Multilayer Perceptron

A Multilayer Perceptron (MLP), also known as a fully-connected network, is a simple Deep Learning architecture. In every layer, every neuron output is connected to every neuron input in the next layer. Formally, an MLP network can be considered as a chain of functions:

$$\begin{aligned} h_1 &= g_1(W_1 x + b_1) \\ h_2 &= g_2(W_2 h_1 + b_2) \\ h_3 &= g_3(W_3 h_2 + b_3) \\ &\cdots \\ h_L &= g_L(W_L h_{L-1} + b_L) \\ \hat{y} &= h_L, \end{aligned} \tag{2.6}$$

where h_l denotes the output of the l-th layer, g_l denotes the activation function of the l-th layer, W_l denotes the weight matrix of the l-th layer, b_l denotes the biases of the l-th layer, x denotes the input to the network, and $\hat{y}$ denotes the output of the network. Alternatively, a compact formulation of an MLP network can be written as

$$\hat{y} = g_L(W_L(g_{L-1}(W_{L-1}(\ldots g_2(W_2(g_1(W_1 x + b_1)) + b_2)\ldots) + b_{L-1})) + b_L). \tag{2.7}$$

In all publications in this thesis, the most commonly used activation function is the ReLU (Rectified Linear Unit) function [44]

$$g(x) = \max(0, x). \tag{2.8}$$

2.3.2 Convolutional Neural Networks (CNN)

Convolutional Neural Networks implement a stronger assumption about the structure of the data in their architecture compared to MLP networks. One common application for CNNs is image processing, as the inductive bias implemented in their architecture benefits from the correlation of nearby pixels. The basic idea involves sliding a kernel over the image and

producing an output at each position. These outputs form *feature maps*, which are further processed to generate the overall network output.

A feature map from the convolutional layer for a 2D convolutional network can be formulated as [45]

$$S(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n), \tag{2.9}$$

where $S(i,j)$ is the output of the convolutional operation, I represents the input image, and K denotes the convolutional kernel. A deep convolutional network comprises several layers, including further processing such as with non-linear activations, pooling operations, and fully connected layers. More details can be found in [45].

2.3.3 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) are specifically designed to process sequential data. Much like how Convolutional Neural Networks utilize an inductive bias to capitalize on the correlation of nearby pixels, RNNs employ an inductive bias to leverage the correlation between samples that occur close in time. While MLPs and CNNs are feedforward networks without connections that feed back to previous stages in the network, RNNs have connections that feed information from later stages back to earlier stages. This functionality allows RNNs to implement a form of memory by preserving information about past inputs to the network. A simple example of an RNN can be given by the following formulas:

$$a(t) = b + Wh(t-1) + Ux(t) \tag{2.10}$$

$$h(t) = tanh(a(t)) \tag{2.11}$$

$$o(t) = c + Vh(t), \tag{2.12}$$

where $a(t)$ represents the output from the neural network layer before the activation function, $h(t)$ is the hidden state (output after the nonlinear activation function), W, U, and V are the parameter matrices that contain the trainable neural network parameters, and $o(t)$ is the output of the RNN.

Publication II utilizes a specialized version of RNN called LSTM (Long Short-Term Memory) [52], which has the capability to internally learn and control the preservation or forgetting of information. More details about the LSTM architecture can be found in [45].

2.3.4 Relation Networks

Relation Networks focus on generating representations of the relationships between data points, such as the distances between two points in a grid [14, 123]. The general form of a Relation Network can be expressed as

$$RN(O) = f_\phi \left(\sum_{i,j} g_W(o_i, o_j) \right), \tag{2.13}$$

where $O = (o_1, o_2, o_3 ... o_n)$ is a set of objects, function g_W processes the input objects pairwise, and function f_ϕ processes the sum of pairwise calculations to produce the final output. The term $g_W(o_i, o_j)$ is calculated for every pair of objects (o_i, o_j) in the set O. The functions f_ϕ and g_W are typically implemented as neural networks, enabling the Relational Network to learn complex relationships and mappings between input objects and desired outputs.

The Relational Network achieves permutation invariance through the summation operation and pairwise processing with g_W, ensuring consistent output despite changes in the order or arrangement of objects and allowing it to focus on the relationships between objects for effective relational reasoning.

3. Reinforcement Learning (RL)

Deep Reinforcement Learning plays a central role in all four publications of this thesis. In Publications I and II, various behaviors are learned using the MDP formulation and by training agents with RL. In Publication I, a reinforcement learning-based helper agent learns to perform assistive actions. In Publication II, the policy function of an agent is conditioned on an inferred variable that explains the behavior of another agent. In Publication III, the computational and rational user model is formulated as a POMDP. Additionally, the synthetic "Analyst," which learns to design experiments that result in informative interactions and estimate user model parameters, is also formulated as a POMDP. In Publication IV, the "ADOIRT" concept, which learns to infer student ability, is formulated as a POMDP model.

3.1 Markov Decision Process (MDP) and Partially Observable Markov Decision Process (POMDP)

The fundamental mathematical formulation of an MDP is defined as a tuple M [131]:

$$M = \langle S, A, P, R, \gamma, \rho_0, H \rangle \tag{3.1}$$

- S represents the state space, which is the set of all possible states in the system.
- A denotes the action space, encompassing all possible actions that the agent can take.
- The transition function P is defined as a function of the action space and the state space as $P : S \times A \times S \to [0,1]$. It specifies the probability $P(s'|s,a)$ of transitioning from the current state s to the next state s' when the agent takes action a.

- $R : S \times A \rightarrow \mathbb{R}$ is the reward function, which specifies the immediate reward that the agent receives after taking an action a in the state s.

- γ represents the discount factor, a value between 0 and 1, that determines the value of rewards received further in the future.

- ρ_0 defines the distribution of initial states at the beginning of episodes.

- H denotes the horizon, which is the maximum number of steps that the agent can take in the environment.

An essential concept in the MDP formulation is the Markov property, which asserts that the future of the stochastic process depends only on the current state, not on any past states.

The POMDP formulation accounts for systems that are not fully observable by introducing uncertainty of the current state through noisy or incomplete observations. Formally, the POMDP extends the MDP formulation by adding an observation function $o = f(s,a)$ to it.

Reinforcement Learning (RL) is a pivotal concept in both machine learning and artificial intelligence, and its influence extends into other fields as well, although the terminology can vary across disciplines. For example, what is commonly referred to as a 'reward function' in RL may be termed a 'utility function' in the context of computational rationality [60].

3.2 RL Agent

The RL agent makes sequential decisions on how to interact with the environment by performing actions within it. The environment changes its state based on the received action, current state, and its internal transition function. At every timestep in the sequence, the environment returns a reward signal along with information about the new state. The RL agent uses this information to learn a policy function $\pi(a_t|s_t)$. Notably, the goal of the RL agent is to maximize the expected sum of discounted rewards over the episodes:

$$J(\pi_w) = E_{\tau \sim p_\pi(\tau)}[\sum_{t=0}^{H} \gamma^t R(s_t, a_t)], \tag{3.2}$$

where $\tau = (s_0, a_0, ...)$ represents the trajectory of states s_t and actions a_t of the agent at time step t for an episode of length H, and w denotes the parameters of the policy function.

3.3 Policy Gradient

One way to classify various RL algorithms is by dividing them into three classes: policy gradient, value-based algorithms, and model-based algorithms. In all publications of this thesis, policy gradient algorithms are used, which are introduced below.

The main idea of the policy gradient is to directly optimize the parameters in the policy function with respect to the objective function $J(\pi_w)$ using gradient ascent optimization

$$W \leftarrow W + \alpha \nabla_W J(\pi_W), \tag{3.3}$$

where W denotes the policy function parameters, α is the learning rate hyperparameter, and π is the policy function.

The basic form of the policy gradient method is the REINFORCE algorithm [140], where the objective function is

$$J(\pi_W) = E_{\tau \sim \pi_W} R(\tau) = E_{\tau \sim \pi_W} \sum_{t=0}^{H} \left[\gamma^t r(s_t, a_t) \right], \tag{3.4}$$

where τ is a trajectory generated by the policy, $R(\tau)$ is the return for the trajectory τ, and a_t and s_t are the action and state at time step t.

The gradient for the objective function can be derived as

$$\nabla_W J(\pi_W) = E_{\tau \sim \pi_W} \left[\sum_{t=0}^{H} \nabla_W \log \pi_W(a_t|s_t) R(\tau) \right]. \tag{3.5}$$

In all publications of this thesis, a more advanced policy gradient algorithm, Proximal Policy Optimization (PPO) [126], is used. The objective function for the PPO algorithm can be written as

$$L(W)^{CLIP} = E \left[\min \left(r_t(W) \hat{A}_t, \text{clip}(r_t(W), 1-\epsilon, 1+\epsilon) \hat{A}_t \right) \right], \tag{3.6}$$

where ϵ is a hyperparameter. In order to avoid too large deviations from the previous policy, the clipping function $\text{clip}(x, x_{low}, x_{high})$ clips the value of x to lie within the range $[x_{low}, x_{high}]$. The PPO uses the advantage function $\hat{A}(s_t, a_t) = R_t - \hat{V}(s_t, a_t)$ instead of the return R_t in the REINFORCE algorithm to reduce variance in the gradient estimate. Intuitively, the advantage function measures how much better it is to take a specific action in a given state over the average of all possible actions in this state. PPO also uses the likelihood ratio between the current policy and the policy that was used for the latest data collection:

$$r_t(W) = \frac{\pi_W(a_t|s_t)}{\pi_{W_{old}}(a_t|s_t)}. \tag{3.7}$$

The overall objective function in PPO includes two additional terms: the value loss and the entropy bonus.

The value loss function measures the discrepancy between the agent's predictions of the expected cumulative future rewards (value estimates) from the current state until the end of the episode, based on its current policy, and the actual returns that are observed from the environment. These actual returns can be calculated using the rewards the agent receives by interacting with the environment.

The role of the entropy bonus is to encourage the agent to explore new states. By adding an entropy bonus to the objective function, the algorithm is incentivised to maintain a certain level of randomness in its action selection process. This helps prevent the agent from converging too quickly to a suboptimal policy that might result from insufficient exploration.

The hyperparameters c_1 and c_2 control the impact of each term in the overall objective function

$$L^{PPO}(W) = L^{CLIP}(W) + c_1 L^{VF}(W) - c_2 L^{S}(W). \tag{3.8}$$

More details of the PPO algorithm can be found in [126].

3.4 Meta-RL

Meta reinforcement learning (Meta-RL) is a subcategory of reinforcement learning that aims for agents to quickly learn to adapt to new situations [34, 137]. In traditional reinforcement learning, the agent learns to optimize the policy through trial and error while interacting with the environment. In Meta-RL, the agent learns to optimize its learning process across multiple tasks, allowing the agent to "learn to learn". The Meta-RL objective can be written as

$$J(\pi_W)^{meta} = E_{p(M)}\left[E_{\tau\sim\pi_W}\left[\sum_{t=0}^{H}\gamma^t r_M(s_t,a_t)\right]\right], \tag{3.9}$$

where $p(M)$ denotes the task distribution, and γ is the discount factor. The policy is conditioned on past transitions which makes it possible for the policy to adapt to a particular task within the task distribution.

3.5 Self-Play

Self-play has been shown to be a successful method for producing an automated curriculum for learning in reinforcement learning settings [128, 10, 79]. Examples of applying self-play in reinforcement learning settings include Silver et al. [128], in which old versions of the opponent were stored during the self-play training to prevent overfitting to the latest policy and for stabilizing the self-play training procedure. Additionally,

Bansal et al. [13] show that sampling opponents from a pool of past versions of the policy improves the performance and robustness of training agents in adversarial settings. In Publication II, we benefit from the sampling of old opponents, but instead of adversarial training, we use self-play to produce a suitable task distribution for meta-learning.

4. Computational User Models

Computational user models are computer programs that are utilized to theorize about and clarify cognitive processes, as well as to explain their resulting behavioral outcomes [20, 65, 55, 132]. They encompass a wide variety of forms, ranging from hand-crafted rule-based systems [23] to learned models trained through machine learning [26]. User models represent one class of cognitive models, which can be regarded as computable representations of psychological constraints on interaction.

In Publication I, an AI agent creates an implicit model of another AI agent using behavior embedding. In Publication II, the AI agent forms an explicit representation of the partnering agent's behavior through skill assessment. In Publication III, the user model is structured as a POMDP and learned via reinforcement learning. This user model is employed to train another AI agent to concurrently select specifications for informative interactions and infer the parameters of the user model. In Publication IV, a closed-form user model is utilized to train an AI agent to deduce student ability within the Item Response Theory (IRT) framework.

In all publications, user models, whether implicit or explicit, are applied to train other AI agents to reason about the behavior of the other party in a sample-efficient and real-time manner. Successfully learning to infer the user model parameters enables fitting the model to a specific human user, which paves the way for adaptive user interfaces, personalized applications, and the foundation for cooperative applications.

4.1 Closed-form and Rule-based User Models

Closed-form and rule-based user models have a long history in Human-Computer Interaction (HCI) research. GOMS (Goals, Operators, Methods, Selection rules) [23] is an example of an early rule-based model that provides a theory of how people perform tasks using computers, and describes the user's cognitive structure with the four components mentioned above. The main idea is that the user model is created by decomposing the user's

task into sub-tasks, and further breaking down these sub-tasks into goals, operators, methods, and selection rules.

The Model Human Processor (MHP) is another example of an early closed-form cognitive model [23]. The MHP is based on the idea that human cognition can be conceptualized as a series of processing stages occurring in the brain, such as perception, attention, memory, and motor control. The MHP models these processing stages as a series of mathematical equations that represent the time and effort required for each stage.

ACT-R (Adaptive Control of Thought-Rational) also consists of multiple modules to model cognitive processes and explains how these modules are integrated to produce coherent cognition [7]. In ACT-R, complex behavior arises from the interaction between two components: procedural knowledge (production rules that specify how tasks are performed based on declarative knowledge) and declarative knowledge (chunks that describe environmental facts and can be modified by production rules). The model initializes declarative knowledge based on the initial state of the environment and applies production rules until a terminal state is reached to simulate user behavior. In Cognitive Science, there are numerous other approaches to user modeling, such as Bayesian [53], Connectionist [36], computational neuroscience [73], among others.

In some cases, cognitive functions can be formulated with simple mathematical equations. For example, in memory research, the retention rate can be modeled with Bernoulli models, where the probability of correct recall can be modeled by $p = a(t+1)^{-b}$ (Power law model) or by $p = ae^{-bt}$ (exponential model), where t is the time since the stimulus, and a and b are the model parameters [25]. A drawback with these types of closed-form models is that they are not able to capture complex human behavior in general, limiting their use to simple behaviors.

Publication IV uses a closed-form user model, which is defined as part of the Item Response Theory (IRT) framework. IRT is a method for automatically inferring student abilities and test item characteristics by observing the outcomes from tests conducted with students [50]. In IRT, the relationship between an individual's ability and their response to a test item is typically modeled with logistic regression, which describes the probability that an individual with a certain ability will give a correct response to a test item with a given difficulty. There are many variants of IRT, the simplest of which is the 1PL model, also known as the Rasch model [118]. The probability that a student i gives a correct answer for the item j is defined as

$$P(y_{i,j} = 1|\theta_i, b_j) = \frac{1}{1 + e^{-(\theta_i - b_j)}}, \tag{4.1}$$

where $y_{i,j}$ is a Bernoulli-distributed outcome, θ_i is the ability of student i, and b_j is the difficulty of the item j. The task is to infer the parameters θ and b from the observations, i.e., from the outcomes of the tests.

4.2 Computationally Rational User Models

For an organism to be seen as rational, it needs to be capable of responding to situations in a way that leads to beneficial results. A common viewpoint suggests that to be considered rational, a being must have the ability to reason or think logically. This capacity involves maintaining a unique set of inner mental states and thought processes which typically include beliefs and desires [139, 60].

Rationality and reinforcement learning can be seen as connected [60]. By interacting with the environment, an agent collects experiences of the external world. The agent does internal processing, updates the internal states and produces actions that can change the environment to the favour of the agent. The meaning and purpose of mental states and desires can be seen as the way to align the mind with the external world, and the way to align the world to the mind, respectively [60]. This view can be formalised by defining an utility function $u : S \times A \rightarrow \mathbb{R}$, which will measure the goodness of the situation, and thus provide a well defined goal for the agent. The agent will attempt to find a strategy that will eventually lead to higher utility. In many cases this requires advanced reasoning and sequential decision making. In reinforcement learning terminology, this refers to the agent finding a policy function that generates a sequence of actions leading to high utility. There is typically uncertainty about the state s, which needs to be taken into account in the decision-making process. The common way is to formulate expected utility $E_{s\sim p(s)}u(s,a)$, where s is drawn with probability $p(s)$. The practical meaning is to sum the utilities over all possible states and weighting the terms with the probabilities of each state [60].

The concept of *computational rationality* [54, 83, 43, 84] is central when considering the context of this thesis. Computational rationality is about assuming that human behaviour can be modelled by finding optimal solutions to sequential decision problems. This assumption is a departure from the view, held by many in the behavioural sciences, that people are irrational or suboptimal [83]. By assuming computational rationality, we can use optimisation algorithms to find human-like policies given models of human perceptual/motor and memory systems. In many cases, the environment might provide only limited amount of information about the state of the world, and thus the decision-making process needs to operate under uncertainty caused by incomplete information. In addition to the external limitations, it is fundamental to consider the limitations, restrictions and preferences which are internal to the agent. This might include inaccuracies in the motor system [135], noise in the perception system [42] as well as limitations in the processing capacity [18] and memory [56]. The utility is subjective to the individual and includes personal preferences. For example, a person might prefer conducting a task quickly and tolerate

some amount of errors in the execution [145]. Various preferences are reflected in the subjective utility function. In summary, computational rationality is used to explain the contribution of subjective utility, capacities and experiences of an agent to observed behaviour by calculating an optimal policy [103, 54].

Solving the modelling problem within the computational rationality framework offers valuable benefits. Using such models allow predicting accurately what behaviour is to be expected in various situations, allowing counterfactual reasoning. Various future scenarios could be simulated and the outcome of various possible trajectories can be observed and analysed. Another benefit is gaining additional information about the reasons for particular behaviour. If the model includes latent variables that explain some particular behaviour, the resulting impact to the behaviour can be observed, and these parameters can also be inferred from observation.

A prevalent mathematical formulation for computationally rational user models is a POMDP, which offers a general framework for sequential decision-making [54, 60]. Within the context of computationally rational theory, POMDPs are expanded to include both internal and external environments [103]. Through this approach, POMDPs enable a comprehensive representation of user behavior and cognitive processes, taking into consideration mental processes, cognitive bounds and interactions with the surrounding environment.

Various technologies have similar goals as computational rationality. As an example, the goal in inverse reinforcement learning (IRL) [98, 66] is to find out the utility function from e.g. demontrations, and then find a policy to optimise that utility. The limitation is that IRL does not take into account cognitive limitations, such as noise in motor movement, noise in perception, and bounds on computation and memory, which can be important determinants of human behavior [54].

4.3 Implicit Models and Explicit Likelihood Models

A model is referred to as an *explicit likelihood model* when there is an option to evaluate the likelihood function $p(y|\theta,d)$. This often occurs when closed-form user models are employed and the likelihood function is mathematically formulated. Conversely, if it is only possible to sample from the likelihood without evaluating its density, the model is considered an *implicit likelihood model*. Specifically, computationally rational user models, such as the one employed in Publication III, might often have intractable likelihoods. Due to the prevalence of implicit likelihood models, numerous likelihood-free methods have been developed within the Bayesian framework [48, 85, 65, 106].

5. Methods for Optimizing Information Gain

Optimizing information gain plays a crucial role in all publications within this dissertation. Publications I and II employ the meta-RL formulation, which can be viewed as an approach to optimize information gain through exploration and fast adaptation to novel situations. Publications III and IV introduce novel methods for integrating amortized user parameter estimation with amortized experimental design.

5.1 Introduction to Experimental Design

In numerous disciplines, conducting experiments is essential for gaining insight into the underlying process [35, 136, 96, 25, 88]. However, performing experiments can often be challenging, time-consuming, expensive, or hazardous. Therefore, it is crucial to minimize the number of experiments needed while maximizing the information obtained from them. Identifying the most informative experiments is widely studied in the research fields of Optimal Experimental Design (OED) [122] and Active Inference. In the context of this dissertation, the goal of an experiment is to maximize the information gained about parameters that describe meaningful aspects of human behaviour.

Advances in Deep Learning have introduced new possibilities for Optimal Experimental Design. Firstly, it enables the amortization of the OED process. Amortization refers to a method where the burden of heavy calculations can be shifted to a pre-computing phase, allowing near real-time execution during deployment. This is a desired feature in experiments involving human participants, as any lag due to computations may be a distraction, leading to impractical experiment settings. Amortization can be achieved by pretraining a Deep Neural Network before deployment using existing or synthetic data. During execution, the pretrained neural network is able to generalize to unseen models within the prior distribution used in the amortization stage.

Moreover, the deep neural network can be conditioned on the outcomes of

previous interactions, allowing for the learning of an adaptive policy where all previous interactions affect the selection of the next design. Additionally, if it is known that multiple experiments need to be conducted, the objective should be to maximize the information gain over the entire sequence of tests, rather than maximizing (myopically) only the next upcoming test. This ability to maximize the entire sequence is referred to as non-myopic capability.

5.2 Bayesian Optimal Experimental Design (BOED)

Bayesian approaches to experimental design start with the assumption that the posterior probability of a parameter value given an experiment is proportional to the likelihood of the data multiplied by the prior. Mathematically, this can be expressed as: $p(\theta|y,d) \propto p(y|\theta,d) \times p(\theta)$. Here, $p(y|\theta,d)$ represents the likelihood of the data given the parameter value and experimental conditions, while $p(\theta)$ represents the prior probability distribution of the parameter value. A common goal in BOED is to run experiments with design parameter d, which maximizes the Expected Information Gain (EIG) $I(d)$:

$$d^* = \arg\max_d [I(d)], \tag{5.1}$$

where $I(d)$ is

$$I(d) = E_{p(\theta)p(y|\theta,d)}[\log p(y|\theta,d) - \log p(y|d)]. \tag{5.2}$$

The idea is to estimate the utilities of hypothetical experiments carried out with each design so that each design's "expected utility" can be computed. This is done by considering every possible observation that could be obtained from an experiment with each design and then evaluating these observations' likelihoods. The design with the highest expected utility value is then chosen as optimal.

The EIG is equivalent to the Mutual Information (MI) between the outcomes y and the user parameter θ, which offers another perspective on EIG. The MI between two random variables is the difference between entropy and conditional entropy [24, 15]:

$$I(P;Q) = H(P) - H(P|Q), \tag{5.3}$$

where $H(P) = -\sum_x p(x)\log p(x)$ and $H(P|Q) = \sum_x p(x)H(P|Q=x)$. Thus, MI measures how much uncertainty of one random variable decreases when information is available about another random variable. Using Bayes' Theorem, formula 5.2 can be written as

$$I(d) = E_{p(\theta)p(y|\theta,d)}[\log p(\theta|y,d) - \log p(\theta)], \tag{5.4}$$

which is the reduction in entropy from prior to posterior.

Another way to view MI is to define it through Kullback-Leibler (KL) divergence, interpreting MI as KL-divergence [75, 15]:

$$I(P;Q) = D_{KL}((P,Q)||PQ), \tag{5.5}$$

where (P,Q) is the joint distribution of P and Q, PQ is the product of their marginal distributions, and the KL-divergence $D_{KL}(P,Q)$ is given by:

$$D_{KL}(P,Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)} \tag{5.6}$$

Thus, maximizing the KL-divergence is an alternative useful objective for design selections.

Ideally, the designs for subsequent trials in an experiment are defined adaptively based on the outcomes of previous trials, and by taking into account the benefit of optimizing the experiment trials over the entire sequence of trials in a non-myopic way. However, this would require calculating posterior distributions at every step of the experiment, resulting in costly calculations. As the posteriors would be used for calculating the nested expected values for maximizing the EIG objective, the calculations become even more costly and are referred to as doubly intractable quantities [116, 40].

5.3 Active Learning

Active Learning is closely related to Optimal Experimental Design. The primary goal in Active Learning is for an algorithm to improve its accuracy using fewer labeled training instances by selectively choosing the data from which it learns [127, 41]. In practice, the algorithm may request the labeling of additional data points from an oracle, such as a human annotator, to enhance its performance. These additional data points are typically unlabeled and are selected by the algorithm based on their potential to improve its accuracy.

An example of an Active Learning application is a system where the designs d refer to unlabelled images or sentences to be selected for annotation, and the outcomes y refer to the labels provided by a human annotator. In this case, the model could be a classifier with parameters θ, and the goal is to predict y from input x [41]. While a typical application of Active Learning involves querying a label from an existing pool of unlabeled data, a common setting in Optimal Experimental Design is to design an experiment that produces a new informative data point.

5.4 Methods based on approximating EIG

To overcome the challenges related to the intractability of BOED-based methods, various suggestions have been made based on amortization or using some form of EIG approximation. Instead of the exact EIG, one alternative is to optimize the contrastive lower bound of the EIG [108, 39]:

$$I(d) \geq E\left[\log \frac{p(y|\theta_0, d)}{\frac{1}{L+1}\sum_{l=0}^{L} p(y|\theta_l, d)}\right], \tag{5.7}$$

where θ_0 is sampled from the prior $p(\theta)$, and which samples are used to generate the outcomes $y \sim p(y|\theta, d)$. The contrastive samples $\theta_{1...L}$ are drawn independently from the prior. Foster et al. suggest Deep Adaptive Design (DAD) [39], which provides non-myopic, amortized experimental design, where instead of an exact mutual information (MI) objective, the lower bound of the mutual information above is maximized. The goal is to learn a policy function by optimizing the parameters of the neural network rather than directly optimizing the designs. The neural network can be conditioned on the entire history of designs and outcomes. This approach solves the intractability problem by avoiding costly computations of exact MI. However, it requires that the likelihood $p(y|\theta, d)$ be known.

MI can also be approximated by training a deep neural network to estimate the MI lower bound, given as [70, 15]:

$$\hat{I}(d, \psi) = E_{p(\theta,y|d)}\left[T_\psi(\theta, y)\right] - e^{-1} E_{p(\theta)p(y|d)}\left[e^{T_\psi(\theta,y)}\right], \tag{5.8}$$

where $T_\psi(\theta, y)$ represents a neural network with parameters ψ, and θ and y represent the inputs to the neural network. This bound approaches MI when it is maximised with respect to ψ. Kleingesse et al. suggest MINEBED, a non-myopic and adaptive method [70], which uses the above approximation as a basis for experimental design optimization by solving

$$d^* = \arg\max_d \max_\psi \left[\hat{I}(d, \psi)\right]. \tag{5.9}$$

MINEBED is applicable to implicit models, i.e., situations where the likelihood function is unknown. It is also applicable when gradients with respect to the designs are unavailable, and there is no possibility to differentiate through the simulator. In this case, the authors suggest using a non-gradient-based method, such as Bayesian Optimization, for updating the designs. However, this approach relies on a surrogate model instead of the true objective.

Other methods for implicit models include various approaches to approximate the likelihood function [59, 104], using Approximate Bayesian Computation (ABC) [32, 110], and using likelihood ratios [133]. The follow-up paper from DAD [39] suggests iDAD [61], extending the DAD method to implicit models by learning an auxiliary critic network. The method

requires the ability to sample from $p(y|\theta,d)$ and to compute the derivative $\partial y/\partial d$.

One possibility to overcome the requirement of a differentiable simulator is to formulate the problem as an MDP and use Reinforcement Learning to learn a policy function. In principle, the Policy Gradient Theorem defines a score-based objective, where the reward function does not need to be differentiable. Other benefits of RL-based methods include applicability to discrete design spaces and the ability to trade off between exploration and exploitation. Blau et al. [19] suggest an RL-based method for optimizing sequential experimental designs. The objective function is the same as used in DAD, i.e., the contrastive lower bound of the EIG, meaning that there is still a need to evaluate the likelihood function $p(y|\theta,d)$.

5.5 User model parameter inference objective

As discussed in Chapter 4, Publication III utilizes a rational user model trained by Reinforcement Learning, whereas Publication IV employs a closed-form user model. Specifically, rational user models formulated as POMDPs have the potential to simulate complex, human-like behaviors which cannot be expressed with closed-form models. In addition, there exists a class of user models referred to as ensemble user models. These models can be conditioned on a range of user parameters to produce various behaviors without the need to retrain the models for each new behavior [95, 76]. The goal is to infer the user parameters with a small number of interactions. However, using implicit user models raises a challenge, as in this setting the likelihood function can not be evaluated. In this dissertation the suggested solution is to learn a near-optimal policy to specify an experiment (i.e. select the experimental design parameters) and conduct parameter inference simultaneously. The policy network is a Deep Neural Network which allows amortisation by pretraining the policy with the data generated by the user model. This user model can be configured to sample from a prior distribution with a wide enough range for its parameters, ensuring sufficient coverage of various behaviors. Consequently, the policy network does not need to generalize beyond the training distribution. The design selection is learned indirectly by using a discrepancy function between the ground truth parameter values from the simulator and estimated parameter values. While many discrepancy functions would work, we have selected a simple L1 objective function which generates a reward by measuring the similarity of predicted user parameter θ_p and true parameters of the user model θ_e as the negative L1 error:

$$r(s_t,a_t) = -||\theta_p - \theta_e||_1. \tag{5.10}$$

The reward is directly influenced by the ability to estimate parameters, but it is crucially also indirectly influenced by the ability to design informative experiments that allow efficient inference of the user model parameters. By using this setting it is possible to construct a flexible system which produces amortised, adaptive, non-myopic sequentially optimised designs for implicit models without the need to differentiate through the simulator.

5.6 Bayes optimal meta-RL

Reinforcement Learning agents need to conduct exploration when interacting with the environment in order to improve their policy. The purpose is gaining information in order to exploit this information to maximise cumulative sum of rewards during the episode. Interestingly, meta-RL can be seen as one way to amortise experimental design. Meta-learning is a simple alternative that conditions a system with training samples from past episodes to learn useful inductive biases required during deployment. Ortega et al [102] describe Bayes-optimal meta-RL in the following way: *"the key insight is that the meta-training process generates training samples that are implicitly filtered according to Bayes rule, i.e. the samples are drawn directly from the posterior predictive distribution"*. By utilizing a suitable objective function, meta-learning can leverage these Bayes-filtered samples to develop an adaptive strategy that can solve a task quickly by implicitly performing Bayesian updates "under the hood" without explicitly computing the typically intractable Bayesian updates. As a result, memory-based meta-learning agents can perform as if they have a probabilistic model [102, 101].

6. Summary of contributions

6.1 Publication I: Learning to Assist Agents by Observing Them

Publication I focuses on research question 1. The goal is to enable an external artificial agent (a helper agent) to observe another agent (goal-driven agent) and learn a representation of its behavior. The problem is formulated as meta-learning, and it is demonstrated that capabilities similar to those in inverse reinforcement learning emerge spontaneously.

The publication introduces three different scenarios:

1. Learning representations of the goal-driven agent's behavior from past partial trajectories in an unsupervised way.

2. Learning representations of the goal-driven agent's behavior from past partial trajectories in a self-supervised way, by predicting the actions of the goal-driven agent.

3. Learning representations of the goal-driven agent's behavior from past partial trajectories in a supervised way, by predicting the goal of the goal-driven agent.

6.1.1 Setting

A small population of goal-driven agents is trained to navigate with a minimum number of steps to their randomly positioned goals in a rectangular gridworld with a randomly positioned wall. The helper agent observes several partial episodes, learns about the goal-driven agent's behavior, and creates a behavior embedding, which includes information about the agent's goal. The helper agent has the ability to affect the transition dynamics of the MDP of the goal-driven agent by opening a shortcut for the

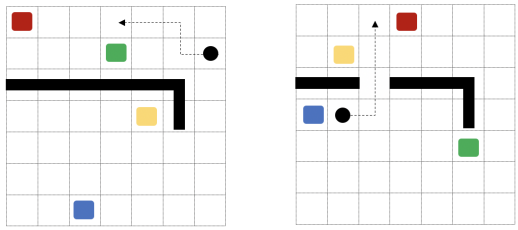

Figure 6.1. Illustration of the 7x7 gridworld. An agent trajectory is illustrated with a dotted line, when the goal-driven agent is walking towards the red goal. Left: the helper agent has not opened a shortcut. Right: The helper agent has opened a shortcut

goal-driven agent. The action space of the goal-driven agent consists of taking a step in any of the four directions, while the action space of the helper agent includes either opening or not opening the shortcut. The goal-driven agent's reward function penalizes every step, thus incentivizing the agent to reach the goal as quickly as possible. The reward function of the helper agent is the same but augmented with a small penalty for opening the shortcut.

Figure 6.1 illustrates a situation in which the agent is navigating towards the red goal. In the left panel, the helper agent has not opened the shortcut because it would not offer any benefit to the agent. The right panel illustrates the situation after the helper agent has opened the shortcut.

6.1.2 Summary of results

The performance was evaluated in three different scenarios, with varying levels of information available to the helper agent. In scenario 1, the helper agent observes several partial episodes of the goal-driven agent in different sampled layouts of the gridworld before acting. By observing the trajectories, which include states and actions of the goal-driven agent, the helper agent forms a behavior embedding that includes information about the agent type. First, embeddings of partial episodes are formed using a convolutional network, followed by mean pooling and a fully connected layer. Then, the information from the observed partial episode embeddings is aggregated by feeding these embeddings into a fully connected layer, followed by mean pooling. The helper agent architecture is illustrated in Figure 6.2. During the training, the goal predictor network learns to construct 2-dimensional behavior embeddings by clustering the agents based on their goals, with each cluster containing agents sharing the same goal (Figure 6.3). In this scenario, the helper agent learns in an end-to-end manner without any pre-training.

In the second scenario, the helper agent undergoes an initial pre-training phase in which it observes multiple partial episodes where the goal-driven agents behave without assistance. This setting models a situation where

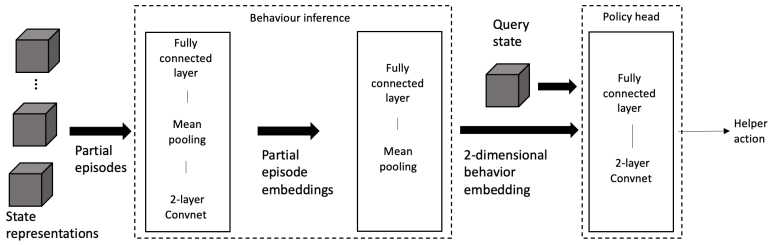

Figure 6.2. The helper agent architecture in scenario 1. The representations of partial episodes are generated by a 2-layer Convolutional Neural Network, mean pooling, and a fully connected layer. The 2-dimensional embedding is formed by combining these representations through mean pooling and a fully connected layer.

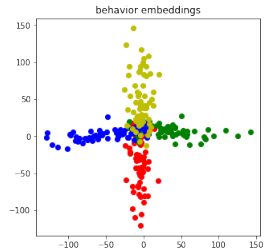

Figure 6.3. Visualization of the 2-dimensional behavior embedding in scenario 1. The behavior inference part of the policy has spontaneously clustered the goal-driven agents based on their goals.

previously collected offline data of the assisted agent behavior is available. The pre-training is conducted in a self-supervised fashion by predicting the action distributions of the goal-driven agents.

In the third scenario, the helper agent once again undergoes an initial pre-training phase in which it observes multiple partial episodes where goal-driven agents behave without assistance. During the pre-training phase, the helper agent is provided with information about the agent's goal in each episode. However, this information is not available during the execution after the pre-training.

In addition to the performance measurement of the scenarios presented above, the mean return over episodes was also assessed when the wall was permanently closed, permanently open, and under the optimal policy. All key results are summarized in Table 6.1. The findings indicate that even when the helper agent cannot observe the behavior of the goal-driven agent, performance is better than without assistance. In such cases, the helper agent attempts to identify the optimal performance by observing the general behavior of all agent types and discerning a more universal policy that is not reliant on a single agent's goal. Performance is further augmented when the helper agent can observe partial episodes of the goal-driven agent (scenario 1). The self-supervised pre-training (scenario 2) and supervised pre-training (scenario 3) achieve the highest overall performance, as they are closest to the optimal performance.

Model	Mean return
Baseline: Wall always open (with penalty for opening the wall)	-5.19
Baseline: Wall always closed	-5.85
Baseline: Helper agent with ground truth goal	-4.48 ±0.03
(oracle) optimal policy	-4.47
Baseline: End-to-end without agent specific inference	-4.82 ±0.03
Scenario 1: End-to-end with agent-specific inference	-4.66 ±0.08
Scenario 2: Self-supervised pre-training with action predictions	-4.59 ±0.04
Scenario 3: Supervised pre-training with goal predictions	-4.59 ±0.03

Table 6.1. As helper agents possess progressively more capabilities, their ability to assist goal-driven agents improves correspondingly.

Figure 6.4 demonstrates the improvement in mean reward throughout the training process for all scenarios and baselines.

6.1.3 Discussion

This publication introduced several scenarios in which a helper agent can assist goal-driven agents by manipulating the transition dynamics of the goal-driven agent's environment. It was demonstrated that capabilities similar to inverse reinforcement learning emerge spontaneously as a result

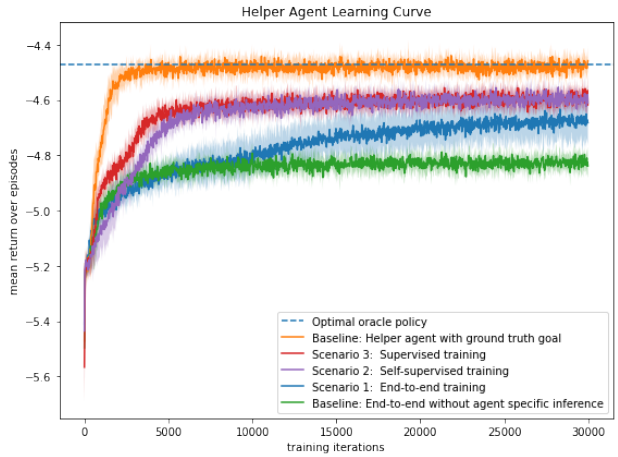

Figure 6.4. Average learning curves of simulations using 8 random seeds and various helper agent architectures. The green line in the figure represents the learning curve without behavior embeddings, serving as a lower bound for the helper agent performance. The orange line displays the performance when the policy network utilizes ground truth information about the agent behavior, acting as the upper bound for helper agent performance. The blue line represents end-to-end training with embeddings (scenario 1), the purple line corresponds to action prediction pre-training (scenario 2), and the red line indicates goal prediction pre-training (scenario 3). The shaded region illustrates the standard deviation over 8 seeds.

of meta-learning, as the helper agent implicitly learns the reward function of the goal-driven agent. This is indicated by observing that the agent trained in scenario 2 reaches the same performance as in scenario 3, which is directly supervised with the information about the goals. The results indicated that by learning to infer the agent's behavior, the helper agent's policy can be significantly improved. This publication directly addresses research question 1 and demonstrates how can existing data sources of user behavior be utilised for learning representations which can improve the performance of a cooperative decision making policy. However, this work is limited to an abstract gridworld environment with a simple distribution of possible behaviors. Although the helper agent can learn behavior embeddings in an end-to-end manner, it has the limitation of needing to observe several partial episodes before acting.

6.2 Publication II: Behaviour-conditioned policies for cooperative reinforcement learning tasks

Publication II extends the work from Publication I and primarily focuses on research question 2. In this study, the objective is to enable an agent to infer the latent variable that explains the behavior of the partner agent and then utilize the inferred information during the initial steps of the on-

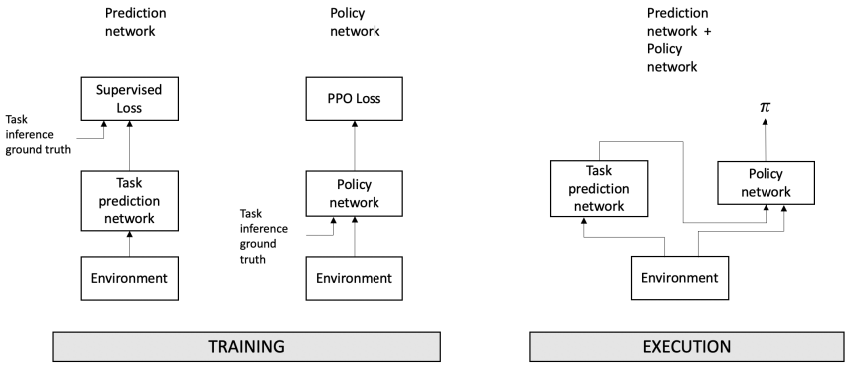

Figure 6.5. Illustration of the training and execution architectures. Two separate networks are trained with ground truth data to predict the task and to run the policy. The execution is carried out by using the task prediction, allowing for decentralized execution.

going task. A method called "Behaviour-conditioned policy" is introduced, which involves training a meta-learner using synthetically produced partner agent populations. This meta-learner can be integrated as part of the agent policy, allowing the agent to "learn to learn" and quickly adapt to unknown agent types in new situations. Synthetic populations with different behaviors are produced using self-play [10, 79]. The main contributions of Publication II can be summarized as:

- A method for automatically generating a task distribution and associated ground truth data from scratch through self-play for training a meta-learner.

- An architecture and training method that can efficiently utilize the generated task distribution and ground truth data.

6.2.1 Setting

The overall task can be considered to comprise two distinct subtasks, specifically inferring latent parameters that explain the behavior of the other party, and adjusting one's own policy based on the inferred latent parameter. In this publication, the latent variable is the skill level of the other party, and a population of various skill levels is trained using self-play. The training and execution architectures are illustrated in Figure 6.5.

6.2.2 Summary of results

In two different experiments, one agent (the main agent) is equipped with the behavior-conditioned policy and adapts to the behavior of the other agent (the partner agent).

The first experiment is a simple matrix game, where the task is to adapt to the partner policy. The purpose of this experiment is to investigate how well the suggested architecture can benefit from a synthetically generated task distribution compared to a strong meta-learning baseline. For the partner agent, the action distributions are generated by drawing them from a symmetric Dirichlet distribution $Dir(\alpha)$ with concentration parameter α. The main agent is equipped with the behavior-conditioned policy, allowing it to learn the behavior of the unknown partner agent during the episode and adjust its own policy accordingly. The partner agent uses the same sampled action distribution throughout the episode, and a new action distribution is sampled for each episode. In one training iteration, 50 partner action distributions are drawn, resulting in 50 different tasks in the meta-training scheme, which is considered to produce sufficient task distribution for the meta-learner. During the training, the task prediction network of the main agent learns to predict the task (the action distribution of the partner agent) using ground truth information, and the policy network learns to conduct the task using ground truth information of the action distribution. During execution, the ground truth information is replaced by the predictions from the task prediction network. For each time step, the shared reward of their joint action is defined by the payoff matrix.

The difficulty of the task can be modified by adjusting the concentration parameter, α. When α has a low value, there is a high likelihood of a single action occurring, making it easier to predict the behavior of the partner agent. Conversely, as α increases, the action distributions become more uniform, making it more challenging to anticipate the action distribution of the partner agent. The payoff matrix is designed so that a more accurate understanding of the other agent's behavior leads to a higher payoff. For example, predicting a specific action taken by the partnering agent can result in a high reward if the prediction is correct, but a substantial negative reward if incorrect. When the prediction of the partner agent's behavior is uncertain, the main agent may prefer to select a low-risk, low-reward action to avoid significant negative rewards. Figure 6.6 illustrates how the behavior-conditioned policy outperforms the RL^2 algorithm [34] as it can achieve a higher mean reward.

In the second experiment, the behavior-conditioned policy architecture is combined with the creation of a meta-learning task distribution using self-play. Essentially, the task is a travelling salesman-type problem for which self-play produces a distribution of agents with different skill levels. This

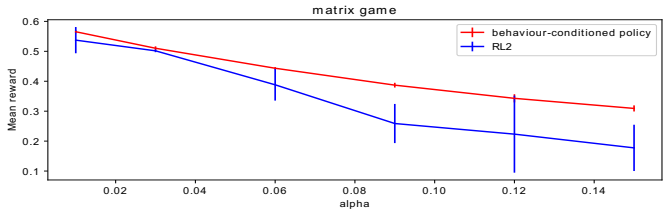

Figure 6.6. Results of the matrix game experiment. The behavior-conditioned policy is able to learn to predict the partner agent behavior during an episode and efficiently use that information in its policy function. The results show that the baseline RL^2 method does not reach as high a mean reward, and the variance across the seeds is considerably higher. Lines represent mean values over five random seeds, and error bars indicate the standard deviation. The rewards are measured at the last time step of the episodes.

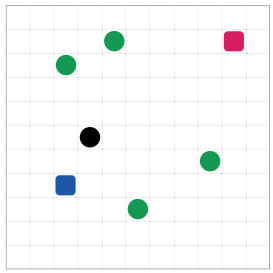

Figure 6.7. Illustration of the 11x11 gridworld. Red and blue squares represent the two agents, the black circle is the final goal, and green circles are the subgoals.

experiment is designed to encourage cooperation and quick task inference during execution. In this task, two agents collect subgoals together in a gridworld (Fig. 6.7), and once all subgoals have been collected, either one of the agents needs to collect the final goal.

The results in Table 6.2 show that both the behavior-conditioned policy and baseline reach strong absolute performance with all partner types when compared to the optimal policy. Furthermore, the performance of both methods consistently improves as the partner agent becomes more skilled. This demonstrates that the meta-learning distribution generated through self-play not only produces strong cooperative agents but also allows for gaining the ability to quickly adapt based on the behavior of the partner agent in novel situations. The results also show that the behavior-conditioned policy outperforms the baseline for all partner types. This indicates that the inductive bias present for exploiting the two-fold structure of the task is useful for fast adaptation, and that the labels produced during self-play can be used as an additional training signal.

6.2.3 Discussion

Publication II introduces methods to enhance cooperation with an unknown partner agent type by synthetically generating a population of agents

Method	partner type	Mean episode length	Mean return
Behaviour-conditioned policy	skilled	15.9 ±0.1	0.080 ±0.001
Behaviour-conditioned policy	intermediate	22.9 ±0.1	0.011 ±0.001
Behaviour-conditioned policy	novice	28.0 ±0.4	-0.040 ±0.004
LSTM policy	skilled	17.8 ±0.2	0.062 ±0.002
LSTM policy	intermediate	25.4 ±0.8	-0.015 ±0.009
LSTM policy	novice	31.8 ±0.6	-0.084 ±0.007
Optimal	skilled	12.9	0.121
Optimal	novice	26.2*	−0.012*

Table 6.2. Main results for simulations in 11 x 11 gridworld, two agents, four subgoals and one final goal. The results show mean values and standard deviations over five random seeds. *This is optimal when one agent collects everything. In our simulations, the other novice agent can accidentally pick up subgoals or the final goal, meaning that the simulation can exceed this value.

and training a meta-learner using these populations. Additionally, an architecture that efficiently exploits the two-fold structure common in many real-world scenarios is suggested. This structure comprises one part that infers the parameter explaining the behavior, and another part that represents the policy conditioned with the inferred parameter.

The results demonstrate that self-play can be used to construct useful task distributions for meta-learning the ability to quickly adapt to different partner types in novel cooperative situations. The results also indicate that utilizing data produced by self-play, along with suitable inductive biases, further improves the performance.

The experiments conducted focused on cases where the latent variable explaining the behavior of the partner agent was the skill level in an abstract gridworld environment. Future work could explore the developed methods for other latent variables that explain behavior and in situations where more human-like behavior is modeled.

6.3 Publication III: Amortised Experimental Design and Parameter Estimation for User Models of Pointing

Publication III shifts focus from abstract gridworld tasks to tasks grounded in cognitive science. This publication builds on the foundations of the previous two publications and concentrates on research question 3. A computationally rational user model is trained by constructing a POMDP formulation for the synthetic user behavior. The architecture of the user model is illustrated in Figure 6.8.

The suggested approach automates parameter estimation for pointing tasks, including mouse movement and gaze-based interactions. The ap-

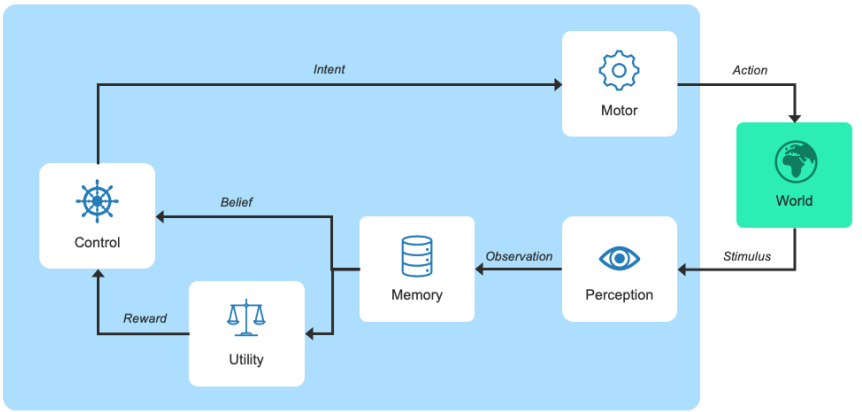

Figure 6.8. User model architecture. Human cognition is modelled as five processes in interaction with a World. The Motor process models movement noise. Perception models increasing visual noise with eccentricity from the fovea. Memory integrates multiple observations using Bayesian inference and Utility generates a reward signal that captures the trade-off between factors such as speed and accuracy.

proach involves designing an optimal sequence of experimental trials (or designs) that maximize the relevance of the available information. To achieve this, a synthetic *Analyst* is trained to conduct the best sequence of experiments for determining the model parameters. Similar to the user model, the Analyst's environment is also formulated as a POMDP. The Analyst implements a novel approach for estimating user model parameters by amortizing the cost of choosing experiments and parameter inference. Although the proposed method is general, its viability is demonstrated here for pointing tasks.

6.3.1 Setting

An overview of the approach is illustrated in Figure 6.9. The approach estimates the parameters of a user model for an individual human user (Phase 3), having previously computed an 'ensemble' user model for all possible parameter combinations (Phase 1) and then an optimal sequential experimental design policy (Phase 2). In Phase 1, the ensemble model of the space of possible users is trained to perform the task for the distribution of possible parameter values and the distribution of possible task environments. The ensemble approach is an important recent advance in user modeling [76, 95]. In Phase 2, the Analyst is trained to conduct the best sequence of experiments for determining the model parameters. Simulated users are randomly sampled given the parameter distribution, and the Analyst learns to fit the model parameters to the simulated user. In Phase 3, the trained Analyst conducts experiments on users and generates parameter fits.

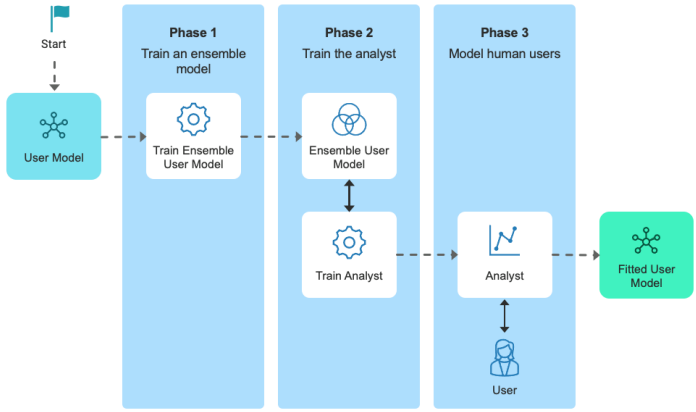

Figure 6.9. Our approach takes a user model as input. This user model has prior distributions over parameters θ but has not been fitted to individual human behavior. In Phase 1, an Ensemble Cognitive Model (ECM) is trained to perform the task for the distribution of possible parameter values. In Phase 2, the cognitive model Analyst is trained to conduct the best sequence of experiments for determining the model parameters. In Phase 3, the trained Analyst is deployed with users, and a fitted model is generated as output.

Phase 1 amortizes the cost of solving the class of POMDPs that represents the hypothesis space of user behaviours. Phase 2 amortizes the cost of computing a non-myopic sequential policy for choosing experiments, and Phase 3 takes advantage of the computation conducted in Phases 1 and 2 in order to optimally gather data and estimate user model parameters without any user-noticeable interaction latency.

6.3.2 Summary of results

Following Meyer's law, it is assumed that people make a series of submovements to achieve a goal [90]. One of the challenges in fitting learning-based user models to human data is that the control policy must be trained to make predictions for each set of possible parameters. For instance, in the gaze-based interaction model reported in [26], oculomotor noise and perceptual noise parameters are properties of an individual user. Both parameters introduce uncertainty in aimed movements to a target. The optimal control policy chooses an aim point for a target in accordance with these uncertainties. For example, when oculomotor noise is high, it makes sense to deliberately undershoot the target and then make a corrective submovement. This is because, on average, this results in a lower overall movement time than overshooting (which takes longer) and correcting.

In the first study in this publication, the approach is applied to a scenario in which only summary statistics are provided to the analyst at the end of each episode. The summary statistics include movement time from the beginning of the episode until the end of the episode, target location, target

width, and the location of the final submovement. Although individual submovements are not observed, the goal is to infer the noise parameter affecting these movements. The movement time is calculated with the formula $mt = \sum_i^I (\theta_a x_i + \theta_b)$, where I represents the number of user steps within the episode, x_i denotes the distance of the submovement during the current user step i, θ_a is the slope parameter, and θ_b is the intercept parameter of the movement time model. In study 1, both of these movement time parameters are fixed. The objective of study 1 is to estimate the movement noise parameter.

The larger the movement noise, the further the actual movements can be from the intended aim points. Consequently, the user model requires more steps to reach the target with increased noise values. In order to verify that the user model has adapted to various movement noise values, the average number of steps needed for the user model to reach the target is plotted. In panel (a) of Figure 6.10, it is demonstrated that the fully trained user model requires more steps to reach the target when noise levels are higher. As the user model's behavior is influenced by the parameter value, it should be feasible to train the Analyst to infer the movement noise from behavioral observations.

The ability of the Analyst to estimate parameters was tested, and the results are reported in Figure 6.11. In each of the panels (a), (b), and (c), the true value of the movement noise is plotted against the estimated value. A linear regression fit was performed on all three scatter plots ($a = 0.99, b = 0.91, R^2 = 0.80$ for panel (a), $a = 0.83, b = 0.01, R^2 = 0.77$ for panel (b), $a = 1.00, b = 0.03, R^2 = 0.54$ for panel (c)). The panels differ in the level of perceptual noise, with low perceptual noise in panel (a), higher perceptual noise in panel (b), and the highest perceptual noise in panel (c). Across all three panels, it is evident that the Analyst is capable of providing some level of estimate for the oculomotor noise. However, it is also apparent that the estimates are much better when the perceptual noise is lower (panel (a)).

Figure 6.11 additionally depicts the distribution of experimental design choices made for each level of perceptual noise. Panels (d) and (e), corresponding to panel (a), indicate that for low perceptual noise, experimental designs tend to favor higher eccentricities and larger targets (although some variance exists). However, as perceptual noise increases, the analyst is incentivized to select targets closer to the origin in order to avoid corrupting the data with highly noisy observations. This assertion is supported by panels (f) and (h), which display more experiments with smaller eccentricities at elevated perceptual noise values.

Finally, the quality of the selected designs can be investigated by training the analyst with random design values and comparing the accuracy of the parameter estimations with an analyst trained to optimize design selections. The right panel of Figure 6.10 is a plot of the error in the

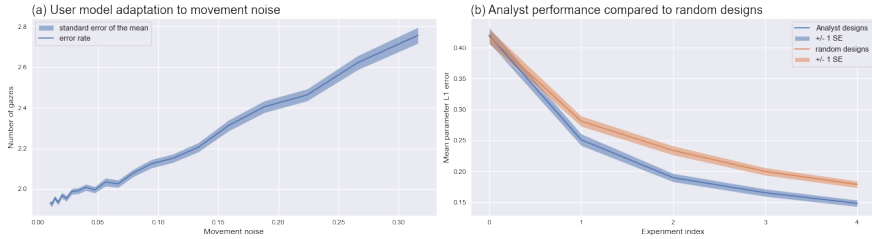

Figure 6.10. Study 1 results. Panel (a) illustrates the number of steps required by the user model to reach the target against increasing movement noise values. Panel (b) shows that the Analyst makes better parameter estimations as more experiments are conducted. The Analyst also performs better across stages compared to a baseline that samples designs uniformly. In panel (a), the shaded area represents ± 1 standard error over an evaluation batch with 1000 user episodes. In panel (b), the shaded area represents ± 1 standard error over an evaluation batch with 1000 models sampled from the prior.

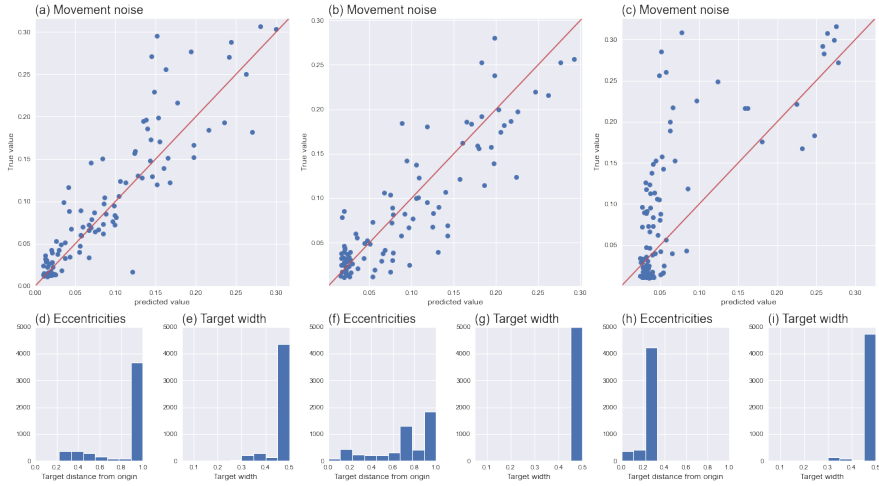

Figure 6.11. Study 1 results. Panels a-c illustrate the accuracy of the movement noise estimations when the user model has low (panel a), medium (panel b) or high (panel c) perceptual noise. Below the scatter plots, the corresponding design selections (target distance from origin and width) by the analyst are illustrated as histograms (panels d-i).

estimate against the experiment number. With zero experiments, there is no data and the parameter estimate is simply the learned prior of the parameter distribution. After incorporating the data from each experiment into a new parameter estimate, the error decreases, and it does so more rapidly with Analyst-designed experiments. This demonstrates a clear improvement in the accuracy of the parameter estimations with optimized designs.

In the second study in this publication, the Analyst was applied to a user model of eye-movements. Instead of using summary statistics as in study 1, the Analyst was allowed to observe the gaze fixations at each step during the episode. Additionally, the model was extended to include a target detection requirement. As a consequence, whether or not the user model observes the target is probabilistic. The probability of not observing the target increases when the target is far away and the target width is small.

As in study 1, the effect of changes in the user model parameters on behavior was first measured. When considering oculomotor and perceptual noise values, increasing one of these noise values while keeping the other fixed should cause the user model to require more gaze fixations to reach the target. This is clearly visible in panel (a) of Figure 6.12, where the perceptual noise value versus the required gaze steps to reach the target is plotted. The near-linear increase in the number of steps with increasing noise suggests that the parameter should be readily recoverable. The oculomotor noise is equivalent to movement noise in study 1, the effect of which is shown in Figure 6.10 of study 1.

In study 2, the goal of the Analyst is to select the most informative experiments for inferring the oculomotor noise, perceptual noise, and the intercept parameters for the movement time model. The Analyst's performance is illustrated in panels (e) to (g) of Figure 6.12. A linear regression was performed to assess the quality of the fits for each parameter ($a = 0.85, b = 0.01, R^2 = 0.76$ for panel (e), $a = 0.87, b = -0.01, R^2 = 0.84$ for panel (f), $a = 1.00, b = 0.04, R^2 = 0.99$ for panel (g)). The selected design values are illustrated in panels (b) and (c) of Figure 6.12. In this case, the Analyst has learned to design experiments with small targets and large eccentricities.

Finally, the performance of the Analyst is compared against an analyst trained using random experimental designs. Panel (d) in Figure 6.12 demonstrates a clear benefit of the optimized designs across experiments.

In summary, the results of study 2 extend those of study 1 by demonstrating that the Analyst can choose experimental designs and accurately infer multiple parameters simultaneously. Additionally, it can do so when each experiment returns a sequence of data (fixation locations and movement times) rather than just summary statistics. Finally, estimating parameters with selected designs outperforms doing so with random designs.

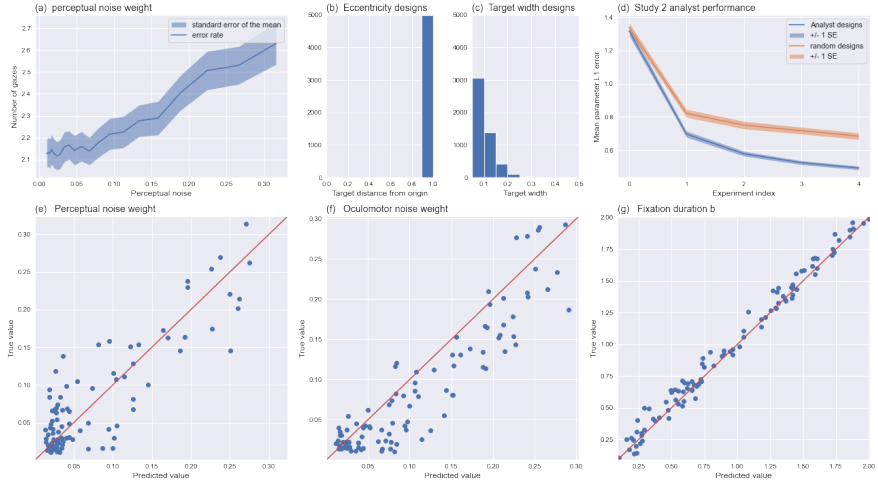

Figure 6.12. Results of Study 2. Panel (a) shows the effect of the user model's perceptual noise parameter on number of steps taken. Panels (b) and (c) show the distribution of the experimental designs chosen by the Analyst. Panel (d) shows the the improvement in parameter estimates with Analyst designed experiments versus with random experiments. The panels on the lower row show the accuracy of the parameter estimations for perceptual noise (e), oculomotor noise (f), movement time interception parameter (g). In panel (a), the shaded area represents ± 1 standard error over an evaluation batch with 1000 user episodes. In panel (d), the shaded area represents ± 1 standard error over an evaluation batch with 1000 models sampled from the prior.

For the most comprehensive study in this publication (study 3), the Analyst estimates these two parameters simultaneously together with a speed-accuracy tradeoff parameter and a movement time parameter. The figure 6.13 illustrates the adaptation of the user model (panels a-d) and the scatter plots of the parameter estimations (panels e-h). The figure 6.14 illustrates the histograms of the selected design values (panels a-b) and the comparison to the random designs baseline (panel c). The reader is adviced to see Publication III for more details and for summary of all studies in this publication.

This publication includes also additional sections for demonstrating how the Analyst is capable of conducting non-myopic design selection and adapting to the previous trials of the experiment.

6.3.3 Discussion

In this publication, the properties of a novel method for user model parameter estimation have been explored, demonstrating that, for three progressively complex pointing tasks, it can make rapid estimates of parameter values for individual simulated users. This is achieved by learning a near-optimal policy for selecting the experiments that are most likely to generate informative data. Properties of the approach include its adaptive, non-myopic designs and its ability to be trained with arbitrary non-

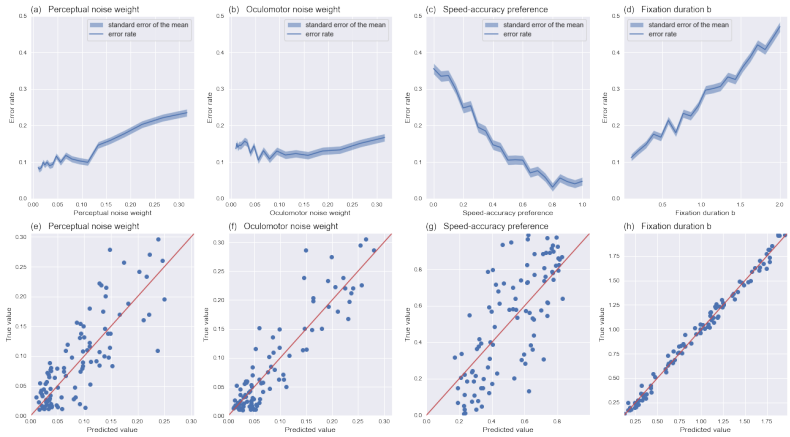

Figure 6.13. Results of Study 3. The upper row illustrates the adaptation of the user model to parameters for the perceptual noise (a), oculomotor noise (b), speed-accuracy preference (c) and movement time intercept (d). The lower row illustrates the corresponding accuracy of the amortized parameter estimates. In panels (a) - (d), the shaded area represents ± 1 standard error over an evaluation batch with 1000 user episodes.

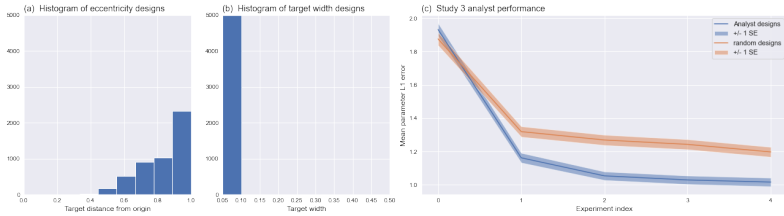

Figure 6.14. Results for Study 3. Panels (a) and (b) show the histograms of experimental designs selected by Analyst. Panel (c) shows the performance gain that follows from inferring parameters on the basis of the designs selected by the analyst compared to random designs. In panel (c), the shaded area represents ± 1 standard error over an evaluation batch with 1000 models sampled from the prior.

differentiable simulators with intractable likelihoods. It is argued that the presented approach has the potential to enhance the contribution that user models make to cooperative AI and personalization by providing interactive systems with the means to automatically and rapidly fit user models to individual users, thereby personalizing interaction to best fit the requirements of the individual.

6.4 Publication IV: Amortised Design Optimization for Item Response Theory

Publication IV extends the work from Publication III to closed-form user models and applies experimental design to the inference of student ability under the Item Response Theory (IRT) framework. Item Response Theory (IRT) is a well-known method for assessing responses from humans in education and psychology. In education, IRT is used to infer student abilities and characteristics of test items from student responses. Interactions with students are expensive, necessitating methods that efficiently gather information for inferring student abilities. Methods based on Optimal Experimental Design (OED) are computationally costly, making them inapplicable for interactive applications. In response, the incorporation of amortized experimental design into IRT is proposed.

As a contribution of this article, amortized experimental design is incorporated into IRT, and a system is constructed in which the next test for the student, conditioned on the previous test outcomes, can be provided in near-real time. The approach involves training a Deep Reinforcement Learning agent, which allows for amortizing both the OED and parameter estimation tasks simultaneously.

6.4.1 Setting

A design optimization approach, named Amortised Design Optimisation for IRT (ADOIRT), is proposed. Design selection and parameter estimation are formulated as a Partially Observable Markov Decision Process (POMDP), and an approximately optimal policy is found using Reinforcement Learning. The goal of the RL agent is to select actions that maximize the expected cumulative reward over an episode of trials, given the history of experiments and outcomes.

The training architecture is illustrated in the left panel of Figure 6.15. At the start of each episode, the simulator is initialized by sampling new model parameters from the prior. ADOIRT requests the item difficulties, and the simulator produces an outcome using the obtained design and sampled student ability. The outcome and obtained design are concatenated to the history of observations.

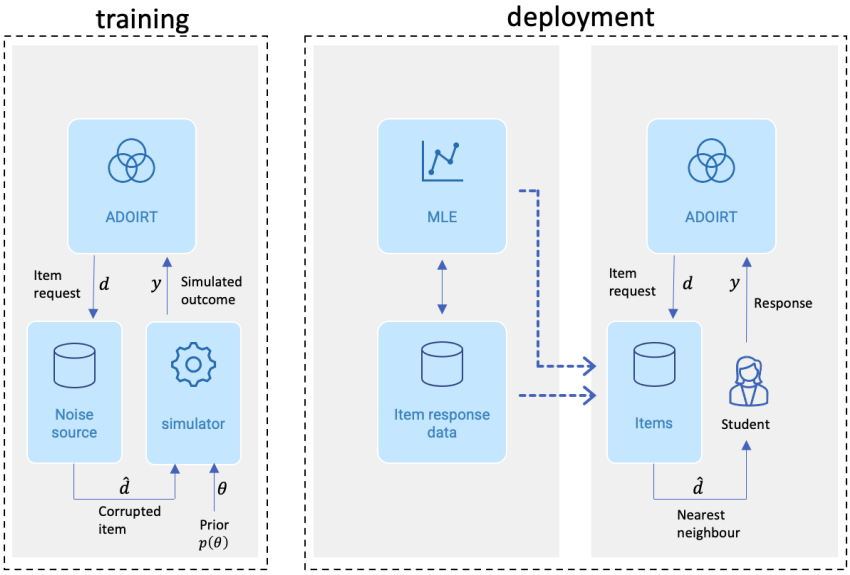

Figure 6.15. ADOIRT architecture. Left panel (training): During training, synthetic data is generated by sampling the student abilities and item difficulties from a prior distribution. The simulator uses the obtained item parameters and the IRT model to produce outcomes. ADOIRT is trained by reinforcement learning to select the most informative items. Right panel (deployment): In the deployment phase, the item parameters can be estimated beforehand with a standard method, such as MLE, by using any existing data. The item requests by ADOIRT are mapped to the closest estimated item difficulty in the collection of available items. Based on the outcomes, ADOIRT selects the most informative next item based on the previous interactions.

During the deployment phase, the trained ADOIRT can be used in conjunction with existing item response data. First, a Maximum Likelihood Estimation (MLE) for the item difficulties is produced using Stochastic Gradient Descent (SGD). Once the estimates of the item difficulties are available, the item requests by ADOIRT are mapped to the closest items from the existing dataset. Based on the outcomes of the student, the agent can select informative items from the existing collection of items.

6.4.2 Summary of results

We assess the performance of the ADOIRT method in estimating student abilities within a realistic, real-world scenario. In this scenario, we assume that there is an existing set of items with unknown difficulties, which are inferred using maximum likelihood estimation. Synthetic datasets with 200 students and 50 items are simulated for training and evaluation. The item request by the agent is then mapped to the closest estimated difficulty in this collection.

We compare the performance of ADOIRT against two baselines: randomly selected designs and a version of ADOIRT trained by masking the

Method	MSE mean	MSE s.e.
Inference with ADOIRT	**1.91**	**0.03**
Inference with non-adaptive designs	3.05	0.07
Inference with random designs	3.39	0.06

Table 6.3. Inference of student abilities with ADOIRT outperforms inference with non-adaptive designs and random designs. The evaluation is performed over 50 datasets, each with their own MLE estimate, and by aggregating the performance over 1000 episodes for each dataset. The mean and standard error are computed by repeating the training with 5 random seeds.

experiment outcomes from observations until the last time step of the episode. In this case, the agent learns an optimized non-adaptive design strategy. Figure 6.16 illustrates the key results. It shows that experiments chosen by ADOIRT result in lower error in student ability estimation when compared to the baselines (panels a-c). In the case of non-adaptive designs (panel c), the inferred student abilities are clustered into 10 clusters. This is a reasonable non-adaptive design strategy, as it effectively covers the design space with a limited number of points. Furthermore, the clusters are denser closer to zero, as incorrect predictions for student abilities far from the mean lead to higher penalties.

The quantitative results are presented in Table 6.3.

6.4.3 Discussion

In this article, a novel method for amortized experimental design and parameter estimation for the IRT setting, named ADOIRT, has been introduced. The joint experimental design and parameter estimation task are formulated as a POMDP, and an adaptive policy is trained using reinforcement learning. The results demonstrate that ADOIRT outperforms non-adaptive and random design strategies, indicating its capability to infer student abilities from a small number of interactions. The results shows that, after the item difficulties have been estimated with a standard method (such as MLE) from a collection of existing items, ADOIRT can then efficiently use these items for experimental design with novel students. The results indicate that ADOIRT is robust to errors that can arise from the quantization when there is limited item availability, and also to the estimation error in the presence of unknown item difficulties.

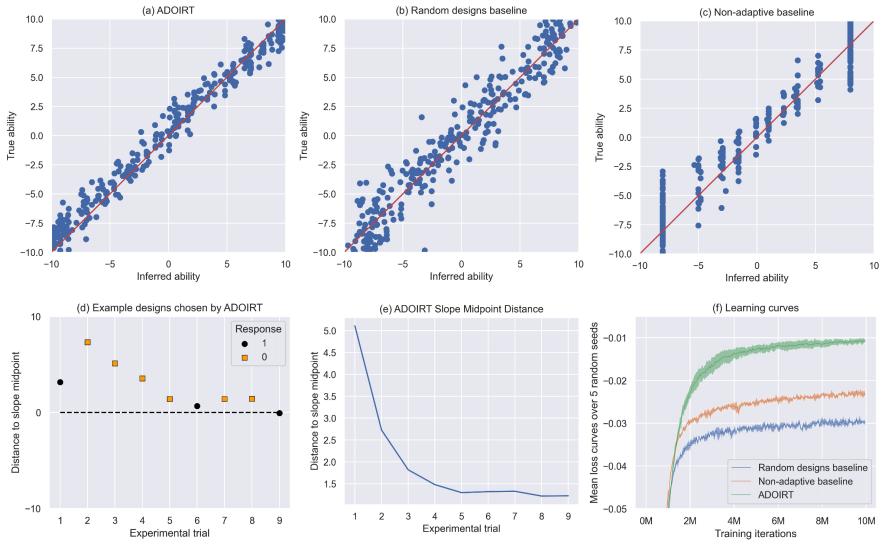

Figure 6.16. Panel (a): True ability against inferred ability for ADOIRT. Panel (b): True ability versus inferred ability for random designs. Panel (c): True ability versus inferred ability for non-adaptive optimised designs. (In (c), an agent is trained with masked outcomes. In this case the agent can only learn an optimised non-adaptive strategy). Panel (d): An example case where 10 items are presented to the student and the agent successfully converges on designs close to the midpoint of the sigmoid function. Panel (e): Illustration of how the design values selected by ADOIRT converge towards the midpoint of the sigmoid function, averaged over 1000 different models. Panel (f): Learning curves of the agent training, training iteration versus mean return of the episodes. The shaded area represents 1 std over 5 training runs. Panels (a)-(e) have been produced with the seed that reached the lowest training error.

7. Discussion and Conclusions

7.1 Comparison to related approaches

The first publication addresses the problem of inferring the goals of another agent and creating representations of their behavior from observations, followed by learning an assistive policy that aims to maximise the objective of the other agent. A rich literature on reasoning about behavior and inferring the goals of other agents [4] is related to this work. In contrast to most works related to modeling other agents, the specific goal of this research is to build agents capable of performing assisting actions based on inferred representations of the other agent's behavior, and by making use of existing data sources.

This work is inspired by Theory of Mind, as behavioral embeddings of goal-driven agents are trained solely through observation. An individual is considered to possess a Theory of Mind (ToM) if mental states are imputed to themselves or others [80, 109]. An inferred mental state can be employed to infer goals and make predictions about the behavior of other agents. Examples of previous work implementing Theory of Mind in a machine learning context include a Bayesian Theory of Mind model [12], and Machine Theory of Mind [111], which is based on deep learning. In the approach presented in Publication I, the behavior of the goal-driven agent is inferred by a separate helper agent, which produces behavior embeddings as in [111]. These embeddings are subsequently utilized for manipulating the environment in a beneficial manner.

The second publication addresses situations in which two agents share the same action and state space. One of the agents is equipped with the capability to assess the skill level of the other agent and adjust its own policy to facilitate cooperation. A long history of research exists in opponent modeling [5]. For instance, Raileanu et al. [115] present an idea where one agent predicts the behavior of another agent by placing itself in a similar situation, and Hu et al. [58] introduce a method for breaking symmetries in

the underlying task, thereby improving cooperation. The former example assumes that the partner agent is similar to oneself, while the latter assumes that the partner agent is optimal but might have converged to a different convention than oneself. In this study, weaker assumptions are made, as the partner agent is not presumed to be optimal. Methods for shaping the learning of other agents are presented by Foerster et al. [38] and Letcher et al. [81]. However, this research does not aim to influence the learning of the partner agent; instead, it focuses on adapting to the behavior of the agent, which is embedded in the transition dynamics.

Applying centralized training with decentralized execution (CTDE) in multiagent systems is an active research area, and determining how to best benefit from centralized training when private information is unavailable during execution remains an open problem [99, 72]. For example, MADPPG [89] employs a centralized critic with decentralized actors, and Foerster et al. [37] propose a method based on using a centralized critic and a specific version of the baseline in an actor-critic algorithm to address the multi-agent credit assignment challenge. Since centralized critics appear advantageous, this concept is incorporated into the baselines, thereby reducing variance in the policy gradient estimates.

Self-play has been demonstrated as a successful method for generating an automated curriculum for learning in multi-agent reinforcement learning settings [128, 10, 79]. However, when training is conducted with a copy of the agent itself, cooperation with other agent types exhibiting different behavior may be limited. One potential solution for improving generalization and robustness in self-play is to interleave older versions of the policy during the training procedure. Silver et al. [128] introduced a method for sampling old opponents, which were stored during self-play training in an adversarial setting. This method was employed to prevent overfitting to the latest policy and to stabilize the self-play training procedure. Additionally, Bansal et al. [13] demonstrated that sampling opponents from a pool of past versions of the policy enhances the performance and robustness of training agents in adversarial settings. The proposed method in Publication II also benefits from sampling old opponents; however, instead of adversarial training, self-play is utilized to produce a suitable task distribution for meta-learning.

A typical approach for encouraging neural networks to learn more useful representations for a specific task involves incorporating auxiliary tasks into the learning process [130]. For instance, auxiliary tasks can be employed to improve representations in classification tasks [119, 146] or in the context of deep reinforcement learning [92, 62, 57]. The mechanism presented in this work differs from the conventional use of auxiliary heads; instead of adding an extra auxiliary head to the common representation, a distinct prediction network is trained with a supervised loss, and the policy network is conditioned on this prediction.

In the third publication, computationally rational user models are trained over a distribution of parameters that generate a range of human-like behaviors. A synthetic "Analyst" is trained to design experiments for inferring these parameters with a minimal number of interactions. This work drew inspiration and information from recent research in both HCI and machine learning. Particularly significant are studies on ensemble user models [95, 76] and research on reinforcement learning-based experimental design and inference [19]. Also important is the work on Bayesian approaches to optimal experimental design [97, 24, 39, 70, 71] and the relation network [123], which provides a useful inductive bias for relational reasoning in the context of gaze-based selection.

Bayesian experimental design offers the potential advantages of mathematical rigor and estimates of the posterior distribution of parameter values, as opposed to point estimates. The Bayesian framework addresses experimental design by optimizing the expected information gain (EIG), which measures the decrease in uncertainty regarding the values of the parameters being fitted. EIG is equivalent to maximizing the Mutual Information (MI) between the model parameters and data obtained from designed experiments. However, it is unclear how to utilise the standard Bayesian approach when the inference is amortised, as it does not account for errors in the amortised approximation. Instead of optimizing EIG, the reinforcement learning (RL) approach enables direct optimization of designs for amortized parameter estimation through a joint objective. Furthermore, estimating the MI in the conventional BOED framework is doubly intractable [116, 61]. Due to this computational complexity, searching for optimal designs is often not feasible when conducting experiments, especially when considering the short time-scales of experiments involving human interaction. This has led to approaches that amortize the cost to a pre-trained deep network by using a tractable approximation to the MI [61, 39, 19]. Tractable computation of the MI objective for implicit models is further complicated by the fact that the likelihood function is unknown [30, 85]. Ivanova et al. [61] address this problem by introducing a separate critic network. However, their approach requires the use of a differentiable simulator. Utilizing RL alleviates this need, as the score function gradient estimator directly computes gradients from the specified reward.

Table 7.1 summarizes the capabilities of various recent approaches for optimal experimental design. The proposed method demonstrates the highest level of flexibility, as it is capable of generating amortized, non-myopic, and adaptive solutions in likelihood-free environments without the necessity to differentiate through the simulator.

Publication IV applies amortized experimental design and parameter estimation for inferring student ability with a limited number of interactions under the Item Response Theory (IRT) framework. A considerable amount of recent work has focused on inferring IRT parameters using

Algorithm	Amortized	differentiable simulator not required	non-myopic	Adaptive	likelihood-free
DAD [39]	✔	✔	✔	✔	✗
Blau et al [19]	✔	✔	✔	✔	✗
MINEBED [70]	✗	✔*	✔	✔	✔
Valentin et al [134]	✗	✔	✔	✔	✔
Our method	✔	✔	✔	✔	✔

Table 7.1. Comparison of the capabilities of recent methods for optimal experimental design. * In MINEBED, a backup method is also described based on Bayesian optimisation, which does not require differentiable simulator.

variants of deep neural networks. Wu et al. [141] propose inferring IRT by employing Variational Inference (VI). Paassen et al. [105] combine sparse factor analysis with Variational Autoencoders (VA). Deep-IRT [143] integrates key-value networks with IRT to enhance the explainability of deep learning-based knowledge tracing. Closely related to IRT is Knowledge Tracing (KT), in which students' knowledge is tracked and modeled over time based on their responses during a series of interactions [29]. Knowledge Tracing has been combined with Deep Learning in Deep Knowledge Tracing (DKT) [107], which employs the LSTM architecture to predict student performance based on previous test outcomes. Since then, several methods utilizing deep neural networks have been proposed, such as Deep Learning Based Knowledge Tracing (DLKT), which incorporates attention layers [68]. Other examples of combining Deep Learning and Knowledge Tracing include Ai et al. [2], who propose a personalized exercise recommendation system for an online self-directed learning service, and Dynamic Knowledge Embedding and Tracing by Xu et al. [142]. Within the context of experimental design in education, Zavaleta-Bernuy et al. [144] suggest an adaptive method for designing experiments by using Thompson sampling. In Publication IV, the proposed solution formulates the joint experimental design and parameter estimation task as a Partially Observable Markov Decision Process (POMDP). An adaptive policy is trained using reinforcement learning, and the parameter estimations and experimental designs are simultaneously amortized with a joint reward, leading to a real-time, sample-efficient solution.

7.2 Conclusions and Future Work

All four publications aim at understanding the behavior of a partner agent in cooperative settings, striving to do so in a sample-efficient manner and in real-time, using methods based on reinforcement learning. The first two publications develop methods to understand hidden factors (latent variables) that influence the behavior of the other party in simple gridworld

scenarios. Additionally, these identified hidden factors are then used in learning how to create cooperative strategies. The methods developed are used as a foundation to learn and understand more complex behaviors of user models, which are based on findings in cognitive science. These methods are studied and developed in publications III and IV. The three first publications make use of learned user models with increasing level of complexity, while the fourth publication demonstrates how the developed methods can be applied to a closed-form user model, which is used in real world applications. The ideas presented in the two first publications are applicable for few-shot learning scenarios, where the policy function can be conditioned on several examples of episodes, and instead of optimising the return (utility) over one episode, the return over several episodes can be optimised, allowing the capability for fast adaptation. The methods developed in publications III and IV aim for high sample efficiency through optimal experimental design which is achieved by simultaneously amortising the experimental design and parameter estimation of the user model. These four publications cover a wide range of settings where representations are learned by meta-RL that are useful for optimising a specific co-operative task objective, and where such a capability is learned in the absence of such an objective by incorporating amortized experimental design.

The findings in this thesis suggest several directions for future research. Perhaps most importantly, the effectiveness of the method should be tested with human participants. Although the simulated participants used to test the approach are sampled from the distribution of real users and previous studies have demonstrated the human-like behavior of these simulations [26], further investigation is required. While further user studies would be required to fully establish the empirical validity of the full system, the developed system already facililitates the deployment of learned policies in interactive software. This software can seamlessly choose the best experimental design, update model parameters, update the observation history, and repeat without lag.

Additionally, the approach should be tested on a broader range of tasks to empirically establish its generality. While the method has been formalized in terms believed to be fully general, it has only been evaluated on abstract problems, pointing tasks, and item response theory within this thesis.

In conclusion, the potential utility of learning about behavior, creating useful behavior embeddings, and applying amortized experimental design and parameter estimation based on an RL algorithm, has been demonstrated in-silico. This algorithm learns how to choose experiments for estimating user model parameters, thereby reducing the time cost of interaction with humans and the amount of human data required. In the future, user models with rapid parameter estimation could help improve interactions for each individual user. This paves the way for personalized interactions, including efficient personal assistants and AI-assisted

teaching, among other applications.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[2] Fangzhe Ai, Yishuai Chen, Yuchun Guo, Yongxiang Zhao, Zhenzhu Wang, Guowei Fu, and Guangyan Wang. Concept-aware deep knowledge tracing and exercise recommendation in an online learning system. *International Educational Data Mining Society*, 2019.

[3] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[4] Stefano V Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018.

[5] Stefano V Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 2018.

[6] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

[7] John R Anderson, Daniel Bothell, Michael D Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. An integrated theory of the mind. *Psychological review*, 111(4):1036, 2004.

[8] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

[9] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

[10] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

[11] Chris Baker, Rebecca Saxe, and Joshua Tenenbaum. Bayesian theory of mind: Modeling joint belief-desire attribution. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

[12] Chris L. Baker, Julian Jara-Ettinger, Rebecca Saxe, and Joshua B. Tenenbaum. Rational quantitative attribution of beliefs, desires and percepts in human mentalizing. *Nature Human Behaviour*, 2017.

[13] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

[14] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[15] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International conference on machine learning*, pages 531–540. PMLR, 2018.

[16] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19, 2006.

[17] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[18] Rahul Bhui, Lucy Lai, and Samuel J Gershman. Resource-rational decision making. *Current Opinion in Behavioral Sciences*, 41:15–21, 2021.

[19] Tom Blau, Edwin V Bonilla, Iadine Chades, and Amir Dezfouli. Optimizing sequential experimental design with deep reinforcement learning. In *International Conference on Machine Learning*, pages 2107–2128. PMLR, 2022.

[20] Matthew M Botvinick, Todd S Braver, Deanna M Barch, Cameron S Carter, and Jonathan D Cohen. Conflict monitoring and cognitive control. *Psychological review*, 108(3):624, 2001.

[21] Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019.

[22] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[23] Stuart K Card. *The psychology of human-computer interaction*. Crc Press, 2018.

[24] Daniel R Cavagnaro, Richard Gonzalez, Jay I Myung, and Mark A Pitt. Optimal decision stimuli for risky choice experiments: An adaptive approach. *Management science*, 59(2):358–375, 2013.

[25] Daniel R Cavagnaro, Jay I Myung, Mark A Pitt, and Janne V Kujala. Adaptive design optimization: A mutual information-based approach to model discrimination in cognitive science. *Neural computation*, 22(4):887–905, 2010.

[26] Xiuli Chen, Aditya Acharya, Antti Oulasvirta, and Andrew Howes. An adaptive model of gaze-based selection. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–11, 2021.

[27] Jun Cheng, Guido Novati, Joshua Pan, Clare Bycroft, Akvilė Žemgulytė, Taylor Applebaum, Alexander Pritzel, Lai Hong Wong, Michal Zielinski, Tobias Sargeant, Rosalia G. Schneider, Andrew W. Senior, John Jumper, Demis Hassabis, Pushmeet Kohli, and Žiga Avsec. Accurate proteome-wide missense variant effect prediction with alphamissense. *Science*.

[28] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[29] Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[30] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

[31] Allan Dafoe, Yoram Bachrach, Gillian Hadfield, Eric Horvitz, Kate Larson, and Thore Graepel. Cooperative ai: machines must learn to find common ground. *Nature*, 593(7857):33–36, 2021.

[32] Mahasen B Dehideniya, Christopher C Drovandi, and James M McGree. Optimal bayesian design for discriminating between models with intractable likelihoods in epidemiology. *Computational Statistics & Data Analysis*, 124:277–297, 2018.

[33] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[34] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[35] Sergey Dushenko, Kapildeb Ambal, and Robert D McMichael. Sequential bayesian experiment design for optically detected magnetic resonance of nitrogen-vacancy centers. *Physical review applied*, 14(5):054036, 2020.

[36] Jerome A Feldman and Dana H Ballard. Connectionist models and their properties. *Cognitive science*, 6(3):205–254, 1982.

[37] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[38] Jakob N Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*, 2017.

[39] Adam Foster, Desi R Ivanova, Ilyas Malik, and Tom Rainforth. Deep adaptive design: Amortizing sequential bayesian experimental design. In *International Conference on Machine Learning*, pages 3384–3395. PMLR, 2021.

[40] Adam Foster, Martin Jankowiak, Elias Bingham, Paul Horsfall, Yee Whye Teh, Thomas Rainforth, and Noah Goodman. Variational bayesian optimal experimental design. *Advances in Neural Information Processing Systems*, 32, 2019.

[41] Adam Evan Foster. *Variational, Monte Carlo and policy-based approaches to Bayesian experimental design*. PhD thesis, University of Oxford, 2021.

[42] Wilson S Geisler. Contributions of ideal observer theory to vision research. *Vision research*, 51(7):771–781, 2011.

[43] Samuel J Gershman, Eric J Horvitz, and Joshua B Tenenbaum. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science*, 349(6245):273–278, 2015.

[44] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[45] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[46] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[47] Alex Graves, Rupesh Kumar Srivastava, Timothy Atkinson, and Faustino Gomez. Bayesian flow networks. *arXiv preprint arXiv:2308.07037*, 2023.

[48] Michael U Gutmann and Jukka Corander. Bayesian optimization for likelihood-free inference of simulator-based statistical models. *Journal of Machine Learning Research*, 2016.

[49] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[50] Ronald K Hambleton and Hariharan Swaminathan. *Item response theory: Principles and applications*. Springer Science & Business Media, 2013.

[51] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[52] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[53] Eric J Horvitz, John S Breese, David Heckerman, David Hovel, and Koos Rommelse. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. *arXiv preprint arXiv:1301.7385*, 2013.

[54] Andrew Howes, Jussi PP Jokinen, and Antti Oulasvirta. Towards machines that understand people. *AI Magazine*, 2023.

[55] Andrew Howes, Richard L Lewis, and Alonso Vera. Rational adaptation under task and processing constraints: implications for testing theories of cognition and action. *Psychological review*, 116(4):717, 2009.

[56] Andrew Howes, Paul A Warren, George Farmer, Wael El-Deredy, and Richard L Lewis. Why contextual preference reversals maximize expected value. *Psychological review*, 123(4):368, 2016.

[57] Hengyuan Hu and Jakob N Foerster. Simplified action decoder for deep multi-agent reinforcement learning. In *ICLR*, 2020.

[58] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, 2020.

[59] Xun Huan and Youssef M Marzouk. Simulation-based optimal bayesian experimental design for nonlinear systems. *Journal of Computational Physics*, 232(1):288–317, 2013.

[60] Thomas F Icard. Resource rationality. 2023.

[61] Desislava Ivanova, Adam Foster, Steven Kleinegesse, Michael U Gutmann, and Thomas Rainforth. Implicit deep adaptive design: Policy-based experimental design without likelihoods. *Advances in Neural Information Processing Systems*, 34, 2021.

[62] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *5th International Conference on Learning Representations*, 2017.

[63] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

[64] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[65] Antti Kangasrääsiö, Jussi PP Jokinen, Antti Oulasvirta, Andrew Howes, and Samuel Kaski. Parameter inference for computational cognitive models with approximate bayesian computation. *Cognitive science*, 43(6):e12738, 2019.

[66] Antti Kangasrääsiö and Samuel Kaski. Inverse reinforcement learning from summary data. *Machine Learning*, 107(8-10):1517–1535, 2018.

[67] David E Kieras and Anthony J Hornof. Towards accurate and practical predictive models of active-vision-based visual search. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 3875–3884, 2014.

[68] Minsam Kim, Yugeun Shim, Seewoo Lee, Hyunbin Loh, and Juneyoung Park. Behavioral testing of deep neural network knowledge tracing models. *International Educational Data Mining Society*, 2021.

[69] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[70] Steven Kleinegesse and Michael U Gutmann. Bayesian experimental design for implicit models by mutual information neural estimation. In *International Conference on Machine Learning*, pages 5316–5326. PMLR, 2020.

[71] Steven Kleinegesse and Michael U Gutmann. Gradient-based bayesian experimental design for implicit models using mutual information lower bounds. *arXiv preprint arXiv:2105.04379*, 2021.

[72] Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 2016.

[73] Nikolaus Kriegeskorte and Pamela K Douglas. Cognitive computational neuroscience. *Nature neuroscience*, 21(9):1148–1160, 2018.

[74] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[75] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[76] Minhae Kwon, Saurabh Daptardar, Paul R Schrater, and Xaq Pitkow. Inverse rational control with partially observable continuous nonlinear dynamics. *Advances in neural information processing systems*, 33:7898–7909, 2020.

[77] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.

[78] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[79] Joel Z Leibo, Edward Hughes, Marc Lanctot, and Thore Graepel. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *arXiv preprint arXiv:1903.00742*, 2019.

[80] Alan M Leslie, Ori Friedman, and Tim P German. Core mechanisms in 'theory of mind'. *Trends in cognitive sciences*, 8(12):528–533, 2004.

[81] Alistair Letcher, Jakob Foerster, David Balduzzi, Tim Rocktäschel, and Shimon Whiteson. Stable opponent shaping in differentiable games. *arXiv preprint arXiv:1811.08469*, 2018.

[82] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[83] Richard L Lewis, Andrew Howes, and Satinder Singh. Computational rationality: Linking mechanism and behavior through bounded utility maximization. *Topics in cognitive science*, 6(2):279–311, 2014.

[84] Falk Lieder and Thomas L Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and brain sciences*, 43:e1, 2020.

[85] Jarno Lintusaari, Michael U Gutmann, Ritabrata Dutta, Samuel Kaski, and Jukka Corander. Fundamentals and recent developments in approximate bayesian computation. *Systematic biology*, 66(1):e66–e82, 2017.

[86] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[87] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE transactions on knowledge and data engineering*, 35(1):857–876, 2021.

[88] Thomas J Loredo. Bayesian adaptive exploration. In *AIP Conference Proceedings*, volume 707, pages 330–346. American Institute of Physics, 2004.

[89] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, 2017.

[90] David E Meyer, David E Irwin, Allen M Osman, and John Kounois. The dynamics of cognition and action: mental processes inferred from speed-accuracy decomposition. *Psychological review*, 95(2):183, 1988.

[91] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[92] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[93] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[94] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[95] Hee-Seung Moon, Seungwon Do, Wonjae Kim, Jiwon Seo, Minsuk Chang, and Byungjoo Lee. Speeding up inference with user simulators through policy modulation. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–21, 2022.

[96] Jay I Myung, Daniel R Cavagnaro, and Mark A Pitt. A tutorial on adaptive design optimization. *Journal of mathematical psychology*, 57(3-4):53–67, 2013.

[97] Jay I Myung and Mark A Pitt. Model comparison in psychology. *The Stevens' Handbook of Experimental Psychology and Cognitive Neuroscience*, 5, 2016.

[98] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[99] Frans A Oliehoek. Decentralized pomdps. In *Reinforcement Learning*. Springer, 2012.

[100] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[101] A Emin Orhan and Wei Ji Ma. Efficient probabilistic inference in generic neural networks trained with non-probabilistic feedback. *Nature communications*, 8(1):138, 2017.

[102] Pedro A Ortega, Jane X Wang, Mark Rowland, Tim Genewein, Zeb Kurth-Nelson, Razvan Pascanu, Nicolas Heess, Joel Veness, Alex Pritzel, Pablo Sprechmann, et al. Meta-learning of sequential strategies. *arXiv preprint arXiv:1905.03030*, 2019.

[103] Antti Oulasvirta, Jussi PP Jokinen, and Andrew Howes. Computational rationality as a theory of interaction. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2022.

[104] Antony Overstall and James McGree. Bayesian design of experiments for intractable likelihood models using coupled auxiliary models and multivariate emulation. 2020.

[105] Benjamin Paaßen, Malwina Dywel, Melanie Fleckenstein, and Niels Pinkwart. Sparse factor autoencoders for item response theory. In *Proceedings of the 15th International Conference on Educational Data Mining*, page 17, 2022.

[106] Henri Pesonen, Umberto Simola, Alvaro Köhn-Luque, Henri Vuollekoski, Xiaoran Lai, Arnoldo Frigessi, Samuel Kaski, David T Frazier, Worapree Maneesoonthorn, Gael M Martin, et al. Abc of the future. *International Statistical Review*, 2022.

[107] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. *Advances in neural information processing systems*, 28, 2015.

[108] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

[109] David Premack and Guy Woodruff. Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(4):515–526, 1978.

[110] David J Price, Nigel G Bean, Joshua V Ross, and Jonathan Tuke. On the efficient determination of optimal bayesian experimental designs using abc: A case study in optimal observation of epidemics. *Journal of Statistical Planning and Inference*, 172:1–15, 2016.

[111] Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *International conference on machine learning*, pages 4218–4227. PMLR, 2018.

[112] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[113] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*, 2022.

[114] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[115] Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In *International conference on machine learning*, 2018.

[116] Tom Rainforth, Rob Cornish, Hongseok Yang, Andrew Warrington, and Frank Wood. On nesting monte carlo estimators. In *International Conference on Machine Learning*, pages 4267–4276. PMLR, 2018.

[117] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[118] Georg Rasch. *Probabilistic models for some intelligence and attainment tests.* ERIC, 1993.

[119] Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, 2015.

[120] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[121] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[122] Elizabeth G Ryan, Christopher C Drovandi, James M McGree, and Anthony N Pettitt. A review of modern computational algorithms for bayesian optimal design. *International Statistical Review*, 84(1):128–154, 2016.

[123] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017.

[124] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[125] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[126] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[127] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[128] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[129] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[130] Steven C Suddarth and YL Kergosien. Rule-injection hints as a means of improving network performance and learning time. In *European Association for Signal Processing Workshop*, pages 120–129. Springer, 1990.

[131] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[132] Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285, 2011.

[133] Owen Thomas, Ritabrata Dutta, Jukka Corander, Samuel Kaski, and Michael U Gutmann. Likelihood-free inference by ratio estimation. *Bayesian Analysis*, 17(1):1–31, 2022.

[134] Simon Valentin, Steven Kleinegesse, Neil R Bramley, Michael U Gutmann, and Christopher G Lucas. Bayesian optimal experimental design for simulator models of cognition. *arXiv preprint arXiv:2110.15632*, 2021.

[135] Robert J Van Beers, Patrick Haggard, and Daniel M Wolpert. The role of execution noise in movement variability. *Journal of neurophysiology*, 91(2):1050–1063, 2004.

[136] Joep Vanlier, Christian A Tiemann, Peter AJ Hilbers, and Natal AW van Riel. A bayesian approach to targeted experiment design. *Bioinformatics*, 28(8):1136–1142, 2012.

[137] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[138] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.

[139] PALPH WEDGWOOD. Internalism explained. *Philosophy and Phenomenological Research*, 65(2):349–369, 2002.

[140] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32, 1992.

[141] Mike Wu, Richard L Davis, Benjamin W Domingue, Chris Piech, and Noah Goodman. Variational item response theory: Fast, accurate, and expressive. *arXiv preprint arXiv:2002.00276*, 2020.

[142] Liangbei Xu and Mark A Davenport. Dynamic knowledge embedding and tracing. *arXiv preprint arXiv:2005.09109*, 2020.

[143] Chun-Kit Yeung. Deep-irt: Make deep learning based knowledge tracing explainable using item response theory. *arXiv preprint arXiv:1904.11738*, 2019.

[144] Angela Zavaleta-Bernuy, Qi Yin Zheng, Hammad Shaikh, Jacob Nogas, Anna Rafferty, Andrew Petersen, and Joseph Jay Williams. Using adaptive experiments to rapidly help students. In *International Conference on Artificial Intelligence in Education*, pages 422–426. Springer, 2021.

[145] Shumin Zhai, Jing Kong, and Xiangshi Ren. Speed–accuracy tradeoff in fitts' law tasks—on the equivalency of actual and nominal pointing precision. *International journal of human-computer studies*, 61(6):823–856, 2004.

[146] Yuting Zhang, Kibok Lee, and Honglak Lee. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In *International conference on machine learning*. ICML, 2016.

To effectively collaborate with humans, Artificial Intelligence (AI) systems must understand human behavior and the factors influencing it, including their goals, preferences, and abilities. Interactions with humans are typically costly, and in many real-life situations, AI must adapt to human behavior after only a few interactions. Additionally, when AI interacts with humans to learn about their behavior, the interactions need to be conducted without any noticeable delay for the human, which in turn necessitates adaptation in real-time. This thesis investigates how an AI system can learn about other agents in a sample-efficient and real-time manner, using methods based on reinforcement learning. The contributions of this thesis cover a wide range of settings where useful representations of behavior for improving cooperation are learned, along with the efficient learning of complex user models.

Aalto-DT 61/2024

ISBN 978-952-64-1731-8 (printed)
ISBN 978-952-64-1732-5 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Computer Science
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL THESES