

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Alex Hämäläinen

Differentiable User Models

Master's Thesis
Espoo, July 30th, 2021

Supervisor: Professor Samuel Kaski
Advisor: Mustafa Mert Çelikok M.Sc. (Tech.)

Author:	Alex Hämäläinen		
Title:	Differentiable User Models		
Date:	July 30th, 2021	Pages:	54
Major:	Machine Learning, Data Science and Artificial Intelligence	Code:	SCI3044
Supervisor:	Professor Samuel Kaski		
Advisor:	Mustafa Mert Çelikok M.Sc. (Tech.)		
<p>The increasing interaction between humans and computers has pressured computational systems to customize their behavior individually for each user. User modeling is a branch of research that has been frequently proposed to address this challenge. Notably, user models are computational representations of individual users constructed based on observed user behaviors. Current user modeling methods, however, often have limited applicability in practice. For instance, existing methods may be explicitly designed for specific user modeling scenarios or might not be compatible with gradient-based methods.</p> <p>The main contribution of this work is a methodology for learning differentiable user models. In particular, the proposed method trains Neural Process surrogates to imitate existing generative processes for user behavior. Enforcing differentiability on the structure of the surrogates guarantees compatibility with existing gradient-based methods. This allows, for instance, gradient-based online inference for individual user model parameters and utilization of modern computational infrastructures. The proposed approach aims to be utilizable in a wide range of user modeling scenarios with few assumptions.</p> <p>The proposed methodology is experimented with in two user modeling scenarios. Both experiments evaluate the capability of the learned surrogates to represent the likelihood of user behavior as a function of user parameters. As a result of these experiments, it was found that the differentiable surrogates may learn to explain user behavior with similar predictive performance to the ground-truth generative processes of user behavior if given correct user parameters. In addition, it was found that the surrogates can learn to capture the uncertainty regarding the stochasticity of generative processes for user behavior. The obtained results suggest that the learned surrogates may be successfully utilized for user parameter inference and for learning individualized user models.</p>			
Keywords:	user modeling, differentiability, meta-learning, neural processes		
Language:	English		

Tekijä:	Alex Hämäläinen		
Työn nimi:	Differentioituvat käyttäjämallit		
Päiväys:	30. heinäkuuta 2021	Sivumäärä:	54
Pääaine:	Machine Learning, Data Science and Artificial Intelligence	Koodi:	SCI3044
Valvoja:	Professori Samuel Kaski		
Ohjaaja:	Diplomi-insinööri Mustafa Mert Çelikok		
<p>Ihmisten ja tietokonejärjestelmien välisen vuorovaikutuksen lisääntyminen on aiheuttanut kysynnän järjestelmille, jotka kykenevät yksilöimään toimintaansa erikseen yksittäisille käyttäjille. Käyttäjämallinnusta on usein ehdotettu ratkaisuksi tähän haasteeseen. Käyttäjämallit ovat laskennallisia malleja yksittäisistä käyttäjistä, joita voidaan rakentaa hyödyntämällä havaittua käyttäjän toimintaa. Nykyiset menetelmät saattavat kuitenkin olla suunniteltu toimimaan vain tietyissä mallinnusskenaarioissa tai ne saattavat tehdä muita rajoittavia oletuksia.</p> <p>Tämän työn pääkontribuutio on metodologia differentioituvien käyttäjämallien oppimiseksi. Ehdotetun menetelmän keskiössä on neuroprosessien kouluttaminen olemassa olevien käyttäjien toimintaa selittävien prosessien surrogaateiksi. Surrogaattien differentioituva rakenne mahdollistaa gradienttipohjaisten menetelmien hyödyntämisen. Differentioituvuus mahdollistaa esimerkiksi yksilöllisten käyttäjäparametrien gradienttipohjaiseen päättelyyn tai nykyaikaisten laskentainfrastruktuurien hyödyntämiseen. Kyseinen metodologia pyrkii tekemään mahdollisimman vähän rajoittavia oletuksia ja on sovellettavissa laajalti erilaisiin käyttäjämallinnusongelmiin.</p> <p>Työn kokeellisessa osuudessa ehdotettua menetelmää testataan kahdessa käyttäjämallinnusongelmassa. Molemmat kokeet arvioivat opittujen surrogaattien kykyä kuvata käyttäjien käyttäytymisen todennäköisyyksiä käyttäjäparametrien funktiona. Kokeissa selvisi, että differentioituvat surrogaatit oppivat selittämään käyttäjien toimintaa käytännössä yhtä hyvin kuin käyttäjän toiminnan alun perin generoinut prosessi, olettaen että tarkat käyttäjäparametrit ovat saatavilla. Kokeissa lisäksi selvisi, että surrogaatit kykenevät ottamaan käyttäjien toimintaa generoivien prosessien stokastisuuden huomioon. Tuloksista voidaan tulkita, että opittuja surrogaatteja voidaan hyödyntää yksilöllisten käyttäjäparametrien päättelyyn, ja siten myös yksilöllisten käyttäjämallien oppimiseen.</p>			
Asiasanat:	käyttäjämallinnus, differentioituvuus, meta-oppiminen, neuroprosessit		
Kieli:	Englanti		

Acknowledgements

This work was supported by the Academy of Finland (Flagship programme: Finnish Center for Artificial Intelligence FCAI, project 319264) together with the computational resources provided by the Aalto Science-IT Project from Computer Science IT. The thesis work was performed in the Probabilistic Machine Learning Group.

I wish to thank my advisors M.Sc. Mustafa Mert Çelikok and M.Sc. Sebastiaan De Peuter for sharing their expertise and providing guidance during the thesis.

I am also grateful to my supervisor Prof. Samuel Kaski for his invaluable advice and extensive feedback.

Finally, I wish to thank my family, girlfriend and cats for their continuous support.

Espoo, July 30th, 2021

Alex Hämmäläinen

Abbreviations and Acronyms

ABC	Approximate Bayesian Computation
AI	Artificial Intelligence
ANN	Artificial Neural Network
BO	Bayesian Optimization
ELBO	Evidence lower-bound
GP	Gaussian Process
HCI	Human-Computer Interaction
HMC	Hamiltonian Monte Carlo
IL	Imitation Learning
IRL	Inverse Reinforcement Learning
KL-divergence	Kullback-Leibler divergence
MCTS	Monte Carlo Tree Search
MDP	Markov Decision Process
MLP	Multilayer Perceptron
NN	Nearest Neighbour
NP	Neural Process
RL	Reinforcement Learning
ToM	Theory of Mind
VAE	Variational Autoencoder
VI	Value Iteration

Contents

Abbreviations and Acronyms	5
1 Introduction	8
1.1 Motivation	8
1.2 Research Objectives	9
1.3 Thesis Structure	10
2 Background	11
2.1 Artificial Neural Networks	11
2.2 Markov Decision Processes	12
2.3 Reinforcement Learning	14
2.3.1 Q-Learning	15
2.3.2 Model-Based Reinforcement Learning	16
2.3.3 Monte Carlo Tree Search	18
2.4 Meta-Learning	19
2.5 Neural Processes	20
2.6 User Modelling	22
3 Proposed Methodology	25
3.1 Setting	25
3.2 Population Modelling	27
3.3 Model Architecture and Pre-Training	29
3.4 Online Interaction	31
4 Experiment 1: Gridworld Environment	33
4.1 Setting	33
4.2 Implementation	35
4.3 Results	35
4.4 Discussion	37

5	Experiment 2: Menu Search Environment	39
5.1	Setting	39
5.2	Implementation	41
5.3	Results	41
5.4	Discussion	43
6	Discussion	45
6.1	Related Work	45
6.2	Limitations & Directions for Future Research	47
7	Conclusions	49

Chapter 1

Introduction

1.1 Motivation

Recent decades have witnessed an ever-increasing amount of daily interaction between humans and computer systems. Simultaneously, computational systems have become increasingly used by ordinary people with different backgrounds and experts of varying domains. The increased interaction, in turn, has naturally asserted pressure concerning the usability and customizability of these systems - interactive systems should be able to behave as if they were built individually for each user.

As a motivating example, consider the following scenario: An Artificial Intelligence (AI) -assistant helps users navigate a specific website by customizing the user interface to match the intentions and objectives of individual users. It is clear that, if successful, such technology could shorten the interaction time spent by the users on the website. However, on the other hand, the assistant's behavior must be consistent with the user objectives. Otherwise, the assistant might, for instance, accidentally hide website elements that are relevant to the specific user.

Building intelligent systems that are successful in scenarios such as the previous AI-assistant setting has proven challenging. In many scenarios, the users' intentions (and other properties) may be unknown for the computer system. For instance, the user might not be able to specify the relevant information for the assistant, or she might want to focus on her task instead of informing the assistant. Consequently, it is beneficial that computer systems can infer these properties through the behavior of the user. It often also holds that both the inference of the user properties and adaptation according to the properties should be achieved during online interaction to be helpful for the user.

1.2 Research Objectives

User modeling is a branch of research at the intersection of AI and Human-Computer Interaction (HCI) which has been frequently considered for addressing problems similar to the scenario mentioned above (McTear 1993, Papatheodorou 1999). The issue, however, with numerous traditional user modeling methods is that they are often explicitly designed for particular applications or interaction scenarios. On the other hand, more recent approaches that generalize into a broader range of application domains may feature other drawbacks. For instance, approaches of Chandramohan et al. (2011) and Reddy et al. (2020) are limited by that their methods may distinguish individual users only in terms of their objectives. (Chapter 6 provides a more detailed discussion concerning similar literature).

Commonly, user models constructed with existing methods are not differentiable. As such, they are not compatible with existing gradient-based methods leading to, for instance, limited usability with current computational infrastructures. The main contribution of this thesis is a methodology for training differentiable user models. In particular, the approach proposed in this work trains Neural Processes (NP) (Garnelo et al. 2018) as surrogates of existing generative processes for user behavior. Differentiability of the NP-surrogates effectively enables gradient-based online inference of user model parameters and data-efficient training. Furthermore, the proposed method aims to be utilizable in a wide range of user modeling scenarios with few assumptions.

This thesis, including the research questions, revolves around the notion of *behaviour generative process*. In this thesis, behavior generative processes are defined to comprise (stochastic) processes, such as planning, responsible for generating user behaviors. Importantly, these processes are parametrized with user/task¹-specific parameters, which explain the differences in behaviors within a user/task population. However, the mapping between the user/-task parameters and user behaviors is not necessarily deterministic - user behaviors are always associated with a specific instantiation of its corresponding generative process. For instance, repeating a planning algorithm with fixed parameters may result in different behaviors.

As the notion of behaviour generative process is now briefly introduced, the research questions may be specified. The main research questions of this thesis are:

¹In addition to user parameters, interactive scenarios might consider modeling users over multiple tasks, i.e., environments.

- Are the differentiable user models able to learn to imitate generative processes for user behavior?
- Can the differentiable user models generalize to scenarios with previously unseen user/task-specific parameters with little context information?
- Can the differentiable user models model the uncertainty regarding generative process instantiations when conditioned with specific user behaviors?

These research questions act as a direct proxy for the quality of inference and, hence, full user modeling performance achievable with the proposed method. As such, this thesis may be considered as a feasibility study regarding the applicability of the proposed method in full-scale user modeling tasks where user/task-specific parameters must be inferred through user behavior.

1.3 Thesis Structure

The thesis is structured as follows. Chapter 2 familiarizes the reader with relevant background information required in the following parts of the work. In particular, the background section covers the methodological background and briefly reviews the basic definitions and objectives of user modeling. Chapter 3 describes the problem formulation and the details regarding the proposed methodology. The following Chapters, i.e. 4 and 5, are dedicated to experiment scenarios the methodology is applied to. Both chapters describe the underlying experimental settings, implementation details, and results. Chapter 6 provides general discussion on the limitations and directions for future research. This section also compares the proposed methodology to similar approaches in previous literature. Finally, Chapter 7 concludes the thesis.

Chapter 2

Background

This chapter summarizes the most relevant background information utilized in the following parts of the work. In addition, this section gives a brief review of the basic concepts of user modeling.

2.1 Artificial Neural Networks

Artificial neural networks (ANNs), usually referred to as neural networks, are universal function approximators. The basic structure of a neural network consists of a structured network of individual computational units called neurons. Each neuron is responsible for transforming the input signal with a simple linear operation, parametrized by its corresponding weight and bias terms, before passing the transformed signal onto subsequent neurons in the network structure. Additionally, the output signal undergoes a transformation (typically non-linear) before being transmitted forward. Figure 2.1 visualizes an example of a generic feed-forward ANN.

It has been proven that single-layer ANNs with unlimited neurons can represent almost any function (Hornik 1991). However, adding depth (that is, hidden layers) to the network structure allows for representing more complex functions in practice. In particular, hidden layers effectively allow the network to transform the raw input signals into features that may be considered more informative from the perspective of the actual prediction objective. ANNs with multiple hidden layers are commonly referred to as deep neural networks.

Training an ANN usually consists of modifying the parameters of individual neurons such that the ANN optimizes a specific objective. ANNs are typically trained with variants of gradient descent methods (Ruder 2016) which exploit the differentiability of ANNs. The training objective is expressed as a

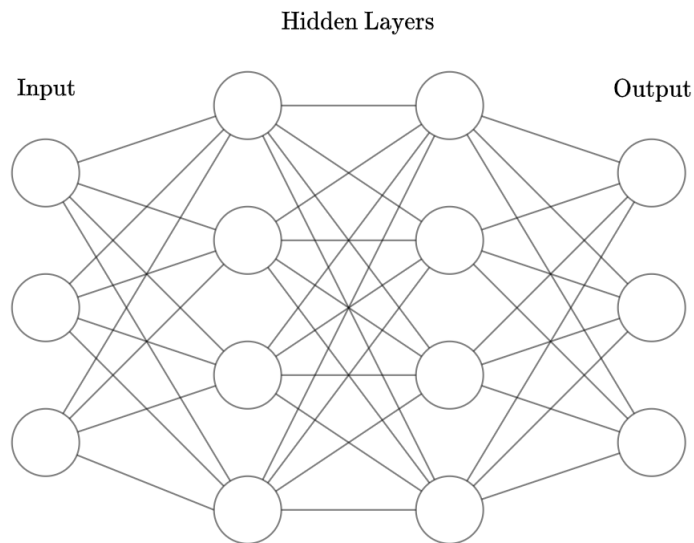


Figure 2.1: A generic feedforward ANN with three input and output neurons and two hidden layers with four neurons.

loss function, which essentially describes the goodness of a model prediction concerning the ground truth label. The choice of the loss function is fundamentally dependent on the approximation task and the design of the network. For instance, cross-entropy is typically utilized for classification tasks while Evidence lower-bound (ELBO) (Kingma & Welling 2013) is a loss typically utilized with Variational Autoencoder (VAE) -based methods.

Differentiable user models utilize ANNs and their variants as basic building blocks. This design choice is deliberately taken to guarantee differentiability of the model structure, enabling utilizing gradient-based methods in conjunction with the model. Chapter 3 describes the structure of the proposed model in more detail.

2.2 Markov Decision Processes

In the context of user modeling, Markov Decision Processes (MDPs) (Bellman 1957) and different MDP variants are commonly utilized for constructing computational models of the underlying environment for user behavior. Furthermore, Markov Decision Processes act as a foundation for different Reinforcement Learning methods, which may act as generative processes for the behavior of users and other agents.

MDPs offer a mathematical framework for modeling dynamic decision-

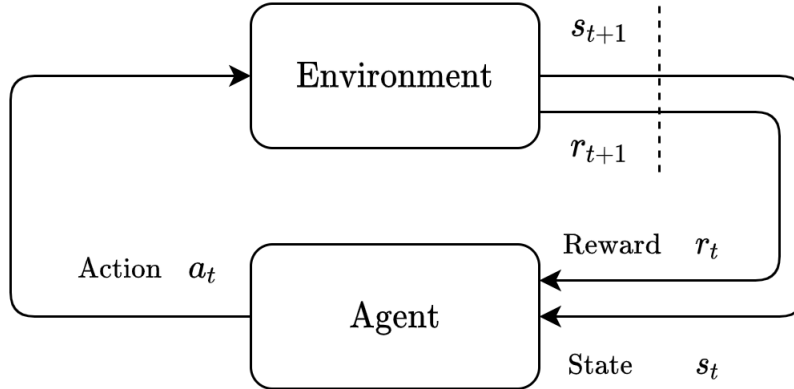


Figure 2.2: Illustration of an MDP interaction cycle. Here, the agent observes the current MDP state, s_t , and acts according to its policy $\pi(a_t | s_t)$. The action triggers a state transition in the environment based on the transition kernel $\mathcal{T}(s_{t+1} | s_t, a_t)$. Simultaneously, a reward $\mathcal{R}(r_{t+1} | s_{t+1}, s_t, a_t)$ is emitted.

making processes in diverse environments. The basic intuition of MDP concerns modeling the behavior of a decision-maker as a result of an optimization process. Traditionally, MDPs are utilized to design behaviors for artificial agents that optimize specific objectives, but they are also utilized for various alternative purposes. The main abstractions of an MDP framework include an environment, which spans all possible world states included in the modeling, and an agent, which corresponds to the decision-maker in the system. In addition, a complete MDP definition describes the optimization objective and the dynamics between the environment and the agents.

Formally, an MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$. Here, \mathcal{S} denotes a (possibly infinite) set of world states included in the decision process modeling. The MDP evolves in time by traversing the state space following the transition dynamics defined by the transition kernel $\mathcal{T} : (s, a, s') \mapsto [0, 1]$. Essentially, the transition kernel assigns to each state transition, $s \rightarrow s'$, conditioned on the actions $a \in \mathcal{A}_s$ ¹ of the agent, a probability density. Finally, a reward function $\mathcal{R} : (s, a, s') \mapsto \mathbb{R}$ describes rewards (or penalties) obtained by the agent for transitions (s, a, s') . Visualization of an MDP interaction step is provided in Figure 2.2. Depending on the modeling setting of interest, the MDP definition may also be extended to feature concepts such as partial observability or multiple simultaneous decision-makers.

¹ $\mathcal{A}_s \subseteq \mathcal{A}$ is here used to denote the set of available actions for the agent at state s .

The MDP formalism acts as a frame for planning (and modeling) the behavior of an agent in decision-making scenarios. \mathcal{S} , \mathcal{A} and \mathcal{T} describe the design and limitations of a decision process while \mathcal{R} works as a metric to evaluate the goodness of a specific transition. Intuitively, a successful agent should aim to find a 'policy' $\pi(a | s)$, i.e., a distribution over actions conditioned on the states, maximizing the accumulated rewards from its sequential actions.

However, rewards alone do not necessarily contain sufficient information about the decision-maker preferences to construct a policy that would capture its objectives. For instance, the agent might assign less value to delayed rewards while holding immediate rewards in higher value. Thus, the agent is often coupled with an optimal criterion which fundamentally assigns each state with a scalar value.

The particular design for optimal criterion is usually carefully selected depending on the problem setting. One commonly used criterion is the infinite discount model, which introduces a discount factor γ to weigh the rewards obtained from future transitions. Crucially, γ is exponentially scaled to implement a larger discount for rewards that are further in the future. By assuming that policy π is followed for future decisions, the discount model allows for assigning each state with its corresponding value as

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_t \gamma^t r_t \right], \quad (2.1)$$

which again induces an optimization objective over the space of policies:

$$\pi^*(s) = \arg \max_{\pi} \mathbb{E}_\pi \left[\sum_t \gamma^t r_t \right]. \quad (2.2)$$

There exists numerous different approaches for optimizing the objective expressed in Equation 2.2 and the particular method is often selected depending on the problem setting.

2.3 Reinforcement Learning

Reinforcement Learning (RL) comprises methods for solving different MDP-based scenarios. Consequently, RL methods may be utilized in modeling generative processes for user behavior in different MDP-based scenarios. This approach is mainly motivated by utility maximization theory, which states that decision-makers aim to conduct behavioral patterns that maximize their

expected utility gained. Furthermore, RL methods allow for utilizing the notion of *computational rationality* (Lewis et al. 2014) which essentially states that the policies are, in particular, constructed as a result of optimization given the limitations of cognitive architectures. Therefore, this approach assumes that the effect of the underlying cognitive architecture may be appropriately encoded in the design of the underlying MDP and the RL algorithm. As noted by Kangasrääsiö et al. (2019), the framework of computational rationality has motivated numerous previous approaches for modeling agents in HCI.

The main focus regarding the generative processes for user behavior is here similarly around RL methods. This section provides a concise overview of RL and two particular methods used in the experimental settings of this work: Q-learning and Monte Carlo Tree Search (MCTS). It should be emphasized, however, that the proposed methodology is, in the end, relatively agnostic in terms of the choice of generative processes and is compatible with also other processes than RL algorithms.

Reinforcement learning covers a variety of methods aiming to approximate the optimal policy for incompletely-known MDP-based environments. In their essence, RL methods assume that dynamics between agents and environment are fully described by an MDP (or by some of its variants) while explicit knowledge about the design of the underlying MDP may be entirely or partially hidden from the agents. Consequently, in addition to the planning problem posed by the MDP task, the agent is responsible for gathering the information required for solving the planning problem by interacting with the environment.

The property mentioned above describes the essence of reinforcement learning, commonly known as the exploration-exploitation dilemma. At the extreme, a purely explorative strategy would only care about gathering information about the system's dynamics, while a thoroughly exploitative strategy would only focus on exploiting the best policy based on current knowledge. However, both of these extreme strategies are almost always strictly sub-optimal, while the optimal strategy comprises a careful balance of these two elements. Although widely studied, a universal solution for the exploration-exploitation dilemma has not been found (Sutton & Barto 2018).

2.3.1 Q-Learning

Q-learning (Watkins 1989) is a reinforcement learning method for constructing the optimal policy in MDP-based scenarios. Q-learning may be categorized as an off-policy temporal difference control algorithm (Sutton & Barto 2018) and is characterized by learning an action-value function Q to approxi-

mate the optimal policy π^* directly. The action-value function, Q , is updated as

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right], \quad (2.3)$$

where γ and α denote the discount rate and step-size parameters respectively. The algorithm essentially trains the action-value function by acting on the MDP based on the current approximate of Q and repeating update steps at each action.

Watkins & Dayan (1992) showed that the algorithm eventually converges to the true optimal policy. Notably, the convergence proof requires that all action-value pairs are continuously updated during training. In order to guarantee that this condition remains satisfied during training, the method is typically coupled with an exploration policy. For instance, ϵ -greedy is a simple exploration policy that states that, at each step, the agent will take a random action with probability ϵ instead of exploiting the optimal one.

The condition for the convergence of the Q-learning algorithm may assert significant limitations in terms of the method's applicability. Specifically, the method becomes computationally expensive for domains with large state or action spaces while being entirely intractable for continuous spaces. In order to address this problem, various function approximation methods have been proposed to replace the tabular format of the action-value function. For instance, Mnih et al. (2013) constructed a Q-network based on convolutional neural networks to approximate the value function based on high-dimensional state representations.

Currently, Q-learning methods utilizing ANNs for approximating value function are commonly referred to as deep Q-learning methods. In contrast to the original tabular Q-learning, this methodology has shown success in domains with high-dimensional state or action spaces by generalizing into unseen states and actions based on on-policy observations. Deep Q-learning has been further extended with methodologies such as double Q-learning (Van Hasselt et al. 2016), dueling architecture (Wang et al. 2016) and prioritized experience replay (Schaul et al. 2015).

2.3.2 Model-Based Reinforcement Learning

The domain of reinforcement learning may be divided into two sub-areas: *model-free* reinforcement learning and *model-based* reinforcement learning (Sutton & Barto 2018). The core emphasis of model-free RL methods lies in learning approximations for the value function based on previous experience. For instance, the Q-learning methodology represents specifically model-free

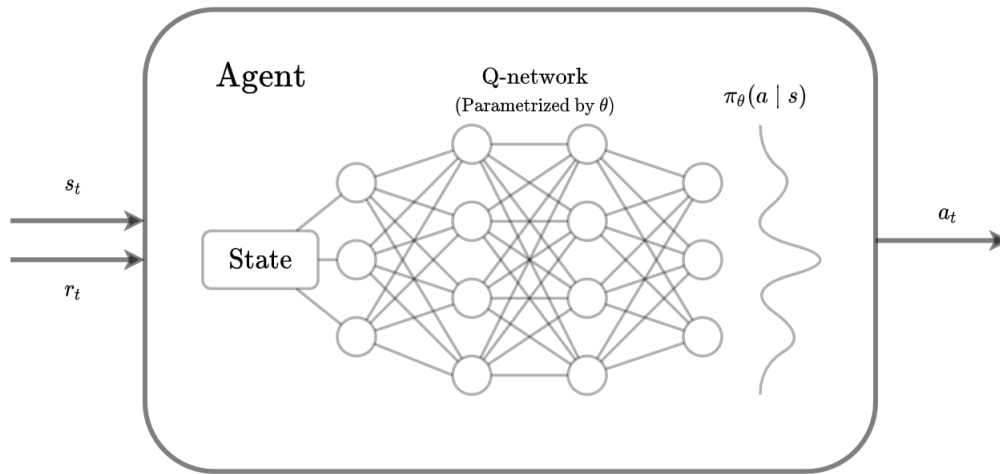


Figure 2.3: Illustration of the agent architecture for standard deep Q-learning.

RL. In contrast, while approximating the value function is similarly the objective of model-based RL methods, they additionally utilize planning on internal world models to learn these approximations. In other words, model-based methods may learn from both actual and simulated experiences. Model-based RL may be considered particularly interesting from the user modeling perspective as it provides tools for approximating the planning process of users. Furthermore, model-based RL enables interactive systems to utilize user models for planning their behavior for individual users.

In the context of model-based RL, Sutton & Barto (2018) define planning as a process that maps the environment model into a policy. Following this definition, a model is simply any basis on which planning can be performed. For instance, a scenario with fully known MDP can be thought of as a model-based RL setting where the agent is equipped with a model that perfectly describes the dynamics of the environment. In such a setting, the agent may, in principle, converge into the optimal policy by planning with the Value Iteration (Bellman 1957) algorithm even without interacting with the environment, given that the scenario is simple enough to support VI.

In most RL problem settings, complete environment models are usually not simply 'given' to the agents but are instead learned based on interaction with the environment simultaneously with the policy. The gathered rewards are used for direct policy updates for each interaction episode, similar to typical model-free approaches. Simultaneously, the environment model is updated utilizing all observed transitions. Especially in domains with sparse

reward functions, this may allow for learning policies with significantly fewer actual interaction episodes as simulated experience may be utilized for the updates.

2.3.3 Monte Carlo Tree Search

Monte Carlo Tree Search is a search algorithm for simulating planning in decision-making domains. MCTS has attracted particular interest in decision domains involving immense state spaces, which would be impossible to solve with, for instance, the standard VI approach. Intuitively, MCTS attempts to solve decision-making problems by simulating random trajectories in the decision-making domain. The results from these trajectories are then collected to guide the decisions for the following trajectory simulations.

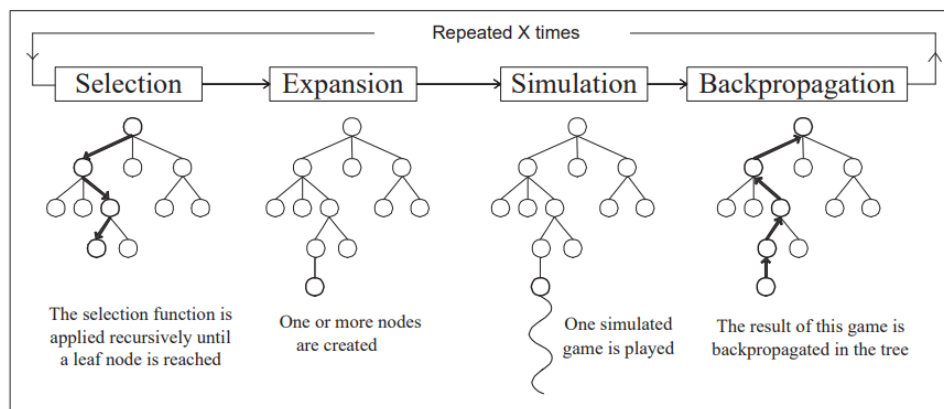


Figure 2.4: Individual search iteration of standard MCTS algorithm (Chaslot et al. 2007). The constructed decision tree naturally poses a strong analogy towards the (MDP) task definition of interest - the tree nodes and edges correspond to the state and action spaces of the task. This correspondence, however, should be distinguished from typical graphic MDP representations as individual states may have multiple occurrences in the tree search structure.

In practice, the MCTS algorithm runs individual search trajectories until a predefined (typically computational) threshold is exceeded, after which the optimal action at the current world state is returned. Each search iteration comprises of four individual steps:

1. **Selection:** The decision tree is iteratively progressed in descending order from the root node according to a child selection policy. The tree

is traversed until the first non-terminal node with unvisited child nodes is reached.

2. **Expansion:** The node selected in the previous step is expanded, i.e., one of its child nodes is added to the tree.
3. **Simulation:** A trajectory is simulated starting from the child node to produce an outcome. For the simulated trajectory, a predefined *rollout policy*² is followed (e.g., random policy).
4. **Backpropagation:** The result obtained from the simulated trajectory is 'backpropagated' through the tree along the selected nodes. The approximated values of these nodes are updated accordingly.

These steps are visualized in Figure 2.4. Upon termination (e.g., due to exceeding a computational threshold), the algorithm returns an optimal action based on the current state (i.e., search tree root).

The behavior of the MCTS algorithm is highly dependent on the choices for child selection and rollout policies. For instance, the child-selection policy often balances between the approximated values of the child nodes and their robustness (e.g., based on their number of visits) (Chaslot et al. 2007). In addition, the child selection policy might balance exploration and exploitation. The rollout policy, on the other hand, aims to approximate the true values of the environment states (Browne et al. 2012). Typically, the rollout policy represents a simple, fast-to-evaluate random policy determining the policy utilized for simulated trajectories after a leaf node is reached. However, recent approaches have replaced the rollout simulations with ANNs that directly approximate the leaf node's value.

2.4 Meta-Learning

Meta-Learning, often informally referred to as *learning to learn*, is a study concentrating on the performance of machine learning algorithms over multiple different learning tasks (Vanschoren 2018). To be more specific, meta-learning studies how knowledge and solutions from previous tasks may be utilized for solving new, previously unknown tasks faster than otherwise possible (Vilalta & Drissi 2002).

The core intuition of meta-learning may be easier to grasp via contrasting with traditional machine learning. Formally, traditional machine learning aims to find parameters \mathbf{w} for a model f such that $f_{\mathbf{w}}$ optimizes a specific

²A (computationally cheap) policy to determine agent behavior during the simulation.

objective (i.e., a loss-function) L . The objective L is commonly evaluated on a single dataset $\mathcal{D} = (\mathbf{x}, \mathbf{y})$ by contrasting the model predictions $\bar{\mathbf{y}} = f_{\mathbf{w}}(\mathbf{x})$ with the ground-truth values \mathbf{y} . The corresponding optimization objective may be written as

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} L(\mathbf{y}, f_{\mathbf{w}}(\mathbf{x})). \quad (2.4)$$

In contrast to the traditional machine learning approach, meta-learning considers learning over multiple learning tasks. Following the definitions of Hospedales et al. (2020), each learning task $\mathcal{T} \sim p(\mathcal{T})$ may be characterized with its corresponding dataset and objective $\mathcal{T} = \{\mathcal{D}, L\}$. The meta-learning process commonly consists of collecting meta-data to summarize prior learning tasks and learned models (Vanschoren 2018) while also defining a metric for measuring task similarity such that the collected meta-data can be utilized appropriately in new tasks.

Similarly, the meta-learning objective augments the standard machine learning optimization objective with the meta-data ω . Utilizing the definitions of Hospedales et al. (2020), the meta-learning optimization objective may be formalized as

$$\min_{\omega} \mathbb{E}_{(\mathcal{D}, L) \sim p(\mathcal{T})} L(\mathbf{y}, f_{\hat{\mathbf{w}}}(\mathbf{x}; \omega)). \quad (2.5)$$

Intuitively, this objective is equivalent to learning to recover meta-data from tasks such that the model performs well in previously unseen tasks with little data.

In the context of this work, it is recognized that user modeling shares several objectives in parallel with those of meta-learning. In particular, this work aims to enforce generalizability between various tasks associated with different users - a successful model should utilize prior experience gathered with previous users and tasks in new environments with new users. Furthermore, it is here emphasized that this generalizability should be achieved with a limited amount of user/task-specific observations. A specific family of models that fits these objectives particularly well is known as neural processes (Garnelo et al. 2018) which is described in detail in the following section.

2.5 Neural Processes

Neural processes (NP) (Garnelo et al. 2018) is a family of neural latent variable models which combines properties of Gaussian processes (GP) and neural networks. In particular, neural processes can be utilized to capture distributions over functions similarly to GPs. Simultaneously, they are efficient

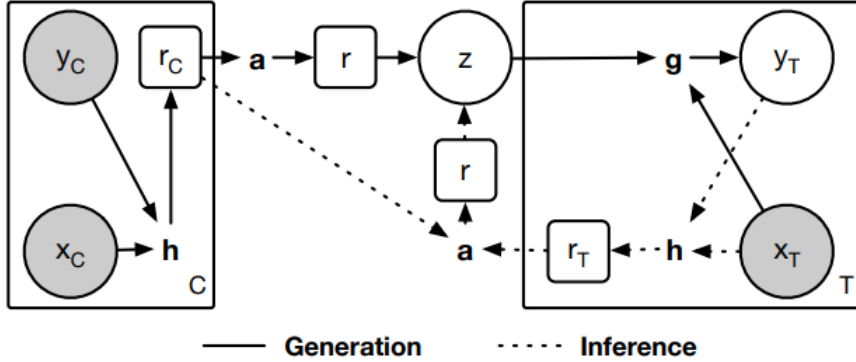


Figure 2.5: The computational diagram of neural processes used in the work of Garnelo et al. (2018). Here, the variables in circles denote the context and target pairs accordingly, while z denotes the global latent variable. The letters h , a and g correspond to the encoder, aggregator and conditional decoder modules. Squared variables denote the intermediate representations utilized by the computational modules. Gray background indicates observability of the variables.

to train and evaluate due to their compatibility with gradient-based methods. On the other hand, neural processes may be considered as a method for few-shot function approximation. From the perspective of this work, this property (coupled with the differentiability of NPs) is especially desirable as it effectively allows for approximating behavioral policies based on few context observations.

Garnelo et al. (2018) summarize the basic idea behind neural processes as follows. Consider a set of functions $f : X \rightarrow Y$, which represent instantiations of an underlying stochastic process F . Neural processes essentially aim to model any function f_d by approximating the underlying stochastic process F with a neural network g . In particular, their approach introduces a latent variable z which aims to capture the instance dependent variation in F such that $F(x) = g(x, z)$.

The practical structure of neural processes consists of an encoder, an aggregator and a conditional decoder as illustrated in Figure 2.5. Here, the encoder is implemented by a neural network h to construct representations $r_i = h((x, y)_i)$ for given context inputs. The aggregator, a , constructs summarizations on the encoded representations e.g. as $r = \frac{1}{n} \sum_{i=1}^n r_i$ to parametrize the (multivariate gaussian) latent distribution $z \sim \mathcal{N}(\mu(r), I\sigma(r))$. Intuitively, the latent distribution aims to summarize the underlying function f_d through the given context. Finally, the conditional decoder $g(x_T, z)$ is inputted with

samples from the (conditioned) latent distribution together with the target inputs x_T to estimate the mapping implemented by the underlying instantiation $f_d(x_T) = y_T$.

NPs are trained by sampling individual instantiations $f_d \sim F$ of the underlying stochastic process F . Essentially, each function $f_d(x)$ is evaluated at a limited number of inputs to produce datasets of tuples $(x, y)_i$ with $y_i = f_d(x_i)$. Each dataset is divided into separate context $(x_{1:m}, y_{1:m})$ and target $(x_{m+1:n}, y_{m+1:n})$ sets. These sets may then be fed to the encoder and conditional decoder as described above. NPs may then be trained on each dataset according to the training objective (NP variant of ELBO) derived by Garnelo et al. (2018):

$$\log p(y_{m+1:n} \mid x_{1:n}, y_{1:m}) \geq \mathbb{E}_{q(z \mid x_{1:n}, y_{1:n})} \left[\sum_{i=m+1}^n \log p(y_i \mid z, x_i) + \log \frac{q(z \mid x_{1:m}, y_{1:m})}{q(z \mid x_{1:n}, y_{1:n})} \right]. \quad (2.6)$$

Here, the $q(z \mid x_{1:m}, y_{1:m})$ and $q(z \mid x_{1:n}, y_{1:n})$ terms correspond to variational posteriors over the latent variable when conditioned on the context and full data respectively. For further information regarding the interpretation of the ELBO (also known as variational lower bound) above, the reader is encouraged to refer to Kingma & Welling (2013) and Garnelo et al. (2018).

In the context of this thesis, NPs act as a central basis for the proposed differentiable user models. In particular, NPs may be viewed as a differentiable method that enables meta-learning models for user behavior. Moreover, as it will be discussed in Chapter 3, NPs allow suitable tools for adaptive and individualized user modeling as they can adapt to specific instantiations of RL-algorithms as generative processes of user behavior.

2.6 User Modelling

This section summarizes well-established notions and definitions of user modeling and reflects them with the proposed method. This section may be considered as an introduction to Chapter 3, which describes the proposed method together with a more refined definition for the user modeling task utilized in this work. The proposed method and approach, together with the main contributions of this work, are contrasted with contemporary approaches in Chapter 6.

User modeling can be viewed as a field of research that explicitly aims to construct computational models of users during use time (Fischer 2001). Such models may be utilized, for instance, for inferring user-specific information or

model-based RL, which ultimately allows the underlying system to customize for individual users.

Papatheodorou (1999) specify two objectives user modeling might aim to capture: specifying a user model representation and acquiring individualized models. In addition to its primary objectives, the proposed methodology may be considered consistent also with the objectives above. In particular, the proposed differentiable user models implicitly capture the first objective by inheriting user model representations utilized by the existing generative processes they are trained to imitate. Additionally, the proposed models may learn latent representations of individual users to capture the uncertainty regarding their behavior generative process instantiations. Finally, the individualized models are acquired here by inferring individual user parameters with gradient-based methods.

Webb et al. (2001) mention four different aspects user models may aim capture:

- Cognitive process behind user behaviour
- Differences between user's skills and expert skills
- Behavioural patterns and preferences of users
- User characteristics.

Traditional user modeling approaches often concentrate on modeling users in terms of specific user aspects. In contrast, the user modeling methodology proposed in this work avoids explicitly committing to any specific properties. The surrogates may, in principle, be trained to imitate behavior generative processes that feature virtually any combination of the aspects mentioned. However, the proposed method (and the experimental section) is originally intended to be utilized for modeling users that generate their behavior by utilizing RL or planning algorithms. As such, the learned models would mainly capture the cognitive processes (i.e., specific RL-algorithms), preferences (i.e., reward functions), and characteristics (i.e., RL-algorithm parametrizations).

Existing user modeling approaches commonly lack compatibility with gradient-based methods. While this alone limits, for instance, their applicability with current computational infrastructures, traditional approaches are often also limited by that they are explicitly designed for specific application domains. In contrast, the proposed methodology for training differentiable user models aims to be utilizable in a broad range of modeling domains. Traditional approaches have been furthermore limited by factors such as data requirements and computational complexity (Webb et al. 2001). The proposed method can utilize experience from previous tasks/users to allow for

data-efficient generalization to previously unseen scenarios. Furthermore, the differentiability of the model guarantees computational efficiency during online interaction.

Chapter 3

Proposed Methodology

The main contribution of this work is a methodology for learning differentiable user models. Specifically, the proposed method trains differentiable surrogate models to imitate existing behavior generative processes in terms of user/task-specific properties θ . This chapter describes the details regarding the assumptions and structure behind differentiable user models and the proposed methodology for training them.

Two aspects are given a particular emphasis in this work: differentiability and generalizability. Enforcing differentiability on the model structure allows utilization of gradient-based methods in conjunction with the surrogate model. For instance, methods for gradient-based inference on θ are available. In addition, differentiable models are fast to evaluate, which allows inference and hence also individualized modeling during online interaction. Finally, differentiability allows for learning user models efficiently with current computational infrastructures allowing scaling into complex, data-intensive modeling domains.

On the other hand, it is also essential to consider the meta-learning - angle on the structure of the surrogate models. The surrogates should utilize experience and knowledge from previous tasks and users to generalize into new, previously unseen tasks with limited amounts of data.

3.1 Setting

In the scope of this work, user models are defined as models capable of describing properties and simulating the behavior of individual users. Formally, the true generative process of user behavior is here denoted as a process \mathcal{P} which may be parametrized with θ to represent the generative process of a specific user in a specific task. A parametrized process, \mathcal{P}_θ , may

construct instances of behavioral policies, π_ϵ , which are consistent with the user/task-specific parameters θ . Here, ϵ is used to denote possible uncertainty/stochasticity related to instantiations of the process (e.g., stochasticity of RL-algorithms). Intuitively, this formalization emphasizes that all variation in behavioral policies between individual users can be explained in terms of θ and ϵ .

The user modelling task is here formalized as follows:

Definition 3.1 *Given a learned approximation, $\bar{\mathcal{P}}$, of the true generative process of user behaviour \mathcal{P} and a behavioural context $(\mathbf{s}_c, \mathbf{a}_c)$, find user/task-specific parameters θ such that instantiations of $\bar{\mathcal{P}}_\theta$ consistent with $(\mathbf{s}_c, \mathbf{a}_c)$ optimize a separately specified objective.*

The definition is deliberately vague to allow different, more specific variants of this task. For instance, in some modeling scenarios, it might be interesting to construct a full posterior over θ conditioned on the behavioral context. In some other scenarios, it might be sufficient to recover θ for which instantiations of $\bar{\mathcal{P}}_\theta$ consistent with the behavioral context maximize the likelihood of behavioral context on expectation. Finally, $\bar{\mathcal{P}}_\theta$ consistent with $(\mathbf{s}_c, \mathbf{a}_c)$ is here generally used to refer to the parametrized process $\bar{\mathcal{P}}_\theta$ that accounts for the uncertainty regarding the instantiations of the ground-truth process \mathcal{P}_θ conditioned on the observed $(\mathbf{s}_c, \mathbf{a}_c)$.

The task of Definition 3.1 induces three sub-problems:

1. Learning the approximation $\bar{\mathcal{P}}$,
2. Inferring θ based on behavioural contexts $(\mathbf{s}_c, \mathbf{a}_c)$, and
3. Representing parametrized processes $\bar{\mathcal{P}}_\theta$ consistent with behavioural contexts.

This work focuses on a methodology for solving the sub-problems 1 and 3. The second problem is solved with existing gradient-based methods that exploit the differentiability of the model. As mentioned, the proposed method solves the first problem by training differentiable surrogate models to imitate existing computational representations of \mathcal{P} . This approach naturally assumes that existing models for describing the generative process of user behavior are available for pre-training the surrogate.

Solving the third sub-problem is equivalent to the ability of the learned user models to take possible stochasticity of the underlying generative processes into account. Here, this problem is solved by constructing the surrogate as an NP-based latent variable model. Conditioning the latent of the surrogate model on behavioral contexts allows the model to represent

distributions over instantiations of the process conditioned on the context. Intuitively, this approach exploits the fact that observed user/task-specific behavior represents a particular instantiation, π_ϵ , of the underlying generative process, thus indirectly leaking information about ϵ . The importance of addressing the third sub-problem is accurately demonstrated in the experiments.

The proposed method assumes that the interaction settings can be feasibly specified as MDPs (or MDP variants) with sufficient accuracy and that a prior over behavior generative processes over the user population of interest is available with sufficient precision. The main emphasis of this work does not directly address methods for specifying suitable MDPs or other priors over the interaction setting. Instead, the work concentrates on methodology for learning differentiable models for individual users, given that the assumptions mentioned above are satisfied. The most important steps for learning and utilizing differentiable user models in online scenarios are listed below:

1. **Prior Specification:** (Not addressed)
 - (a) Specifying the underlying interaction environment as MDP (or similar)
 - (b) Modelling the user population in terms of behavior generative process(es) and specifying a prior over user/task-specific properties (i.e., specifying a computational model for \mathcal{P} and $p(\theta)$)
2. **Pre-Training:**
 - (a) Pre-training the differentiable surrogate with the behavior of simulated users (users are sampled based on the population prior)
3. **Online Interaction:**
 - (a) Initializing the user model based on the population prior
 - (b) Updating user model based on observed interaction

3.2 Population Modelling

An essential part of pre-training differentiable user models is specifying a prior over behavior generative process of the user population. In practice, the behavior generative processes, \mathcal{P} , are often described as sets of one or multiple RL-algorithms. Hence, the prior determines the distribution over user/task-specific parameters, θ , such as the MDPs, reward functions, and an

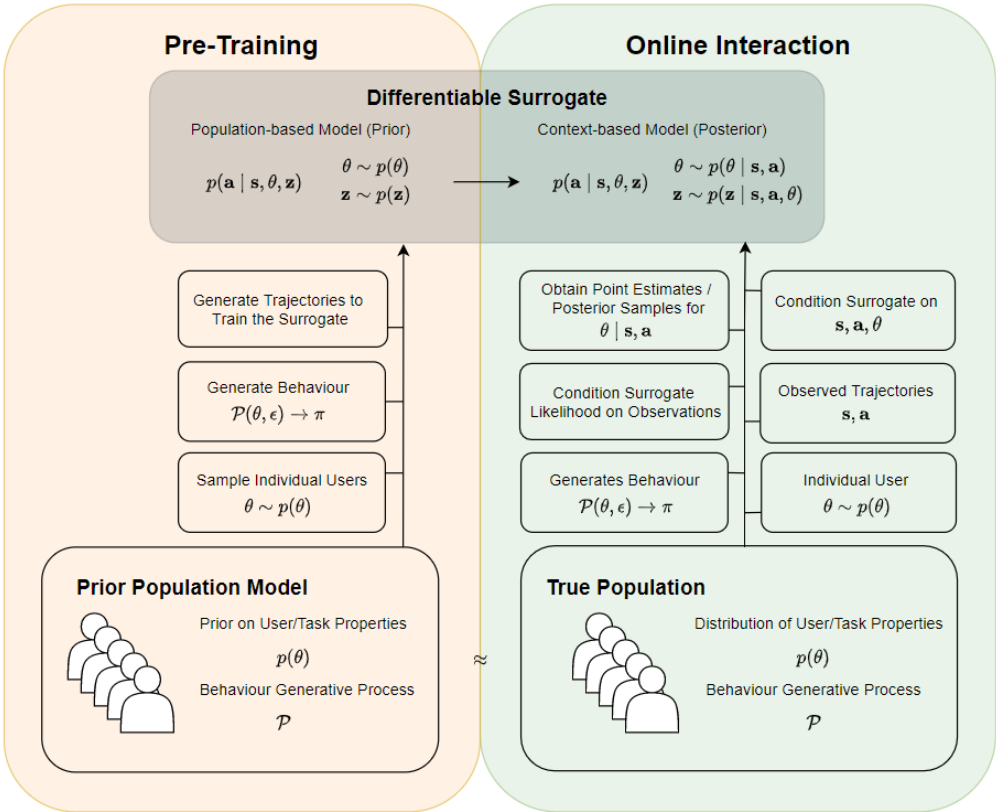


Figure 3.1: Illustration of user modelling workflow with differentiable user models.

RL-algorithm with its corresponding parameters (also, other properties may be specified here). For instance, the prior could be defined over the space of MCTS agents in a static MDP to describe user parameters θ corresponding to reward functions and MCTS parameters. In the end, selecting a suitable prior over the behavior generative processes for practical applications is highly dependent on the underlying setting and is not explicitly addressed in this work.

After the behavior generative process for the user population and a prior over the corresponding parameters have been determined, the surrogate may be pre-trained. During pre-training, individual user parametrizations θ are sampled according to the specified prior to parametrize the behavior generative process \mathcal{P} . The process \mathcal{P}_θ is then utilized to produce behavioral policies, π_ϵ , which are then utilized to generate datasets of simulated behavior trajectories on the environment. The surrogate model may then be trained on the constructed trajectories to approximate the generative process of simulated behaviors, $\mathcal{P} : (\theta, \epsilon) \rightarrow \pi$.

3.3 Model Architecture and Pre-Training

As mentioned, the model structure should be consistent with two important factors: differentiability and generalizability. In particular, the model should utilize experience from previous tasks and users to allow for data-efficient generalization into new, previously unseen contexts. In addition, the model should be able to approximate both

- Individual parametrizations \mathcal{P}_θ of the true generative process for user behavior and
- Distributions over instantiations of \mathcal{P}_θ (i.e., π_ϵ) conditioned on observed behavioral contexts.

Here, the model is built purely as a surrogate of existing generative processes for user behaviour. Motivated by the aforementioned requirements, Neural Process (Garnelo et al. 2018), is chosen as a basis for the surrogate model. The model architecture (Figure 3.2) is characterized by the following components:

Encoder:

$$q(\mathbf{z} \mid \mathbf{s}, \mathbf{a}, \theta) \quad (3.1)$$

Decoder:

$$p(\mathbf{a} \mid \mathbf{s}, \theta, \mathbf{z}). \quad (3.2)$$

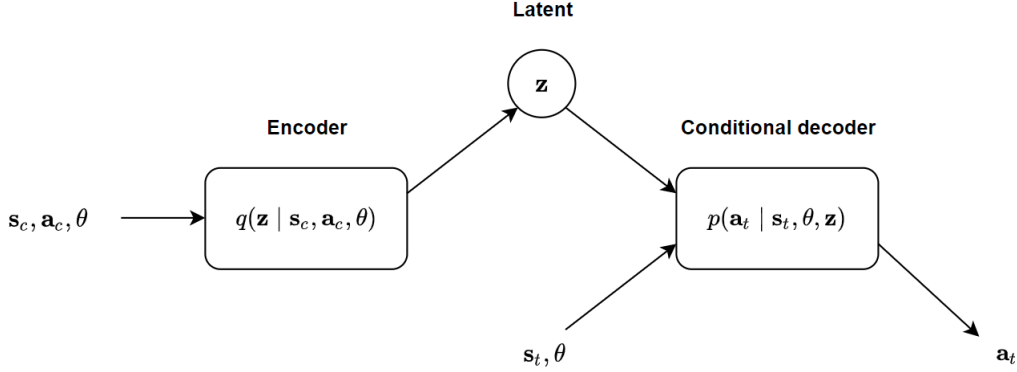


Figure 3.2: Model architecture. The encoder takes sets of context transitions as an input to condition the latent distribution. The conditional latent distribution can be furthermore utilized to obtain predictions for target states.

In practice, the model is trained on trajectories $((\mathbf{s}, \pi_\epsilon(\mathbf{s})), \theta)$ obtained by generating policies $\mathcal{P} : \theta \rightarrow \pi_\epsilon$ on sampled users $\theta \sim p(\theta)$ (Figure 3.1). More precisely, the trajectories of each sampled user are split into context $(\mathbf{s}_c, \mathbf{a}_c)$ and target $(\mathbf{s}_t, \mathbf{a}_t)$ sets. Following the standard NP training procedure, the context data is first used to condition the latent distribution $q(\mathbf{z} | \mathbf{s}_c, \mathbf{a}_c, \theta)$. The conditioned latent distribution is then utilized to construct the prediction for target states as $p(\mathbf{a}_t | \mathbf{s}_t, \theta, \mathbf{z})$. In this work, the model is trained using the NP-variation of ELBO-loss as suggested by Garnelo et al. (2018):

$$\mathbb{E}_{q(\mathbf{z} | \mathbf{s}_{all}, \mathbf{a}_{all}, \theta)} \left[\sum_{s_t \in \mathbf{s}_t, a_t \in \mathbf{a}_t} \log p(a_t | s_t, \theta, \mathbf{z}) + \log \frac{q(\mathbf{z} | \mathbf{s}_c, \mathbf{a}_c, \theta)}{q(\mathbf{z} | \mathbf{s}_{all}, \mathbf{a}_{all}, \theta)} \right] \quad (3.3)$$

Finally, as suggested by Kim et al. (2019), the NP structure is further augmented with attention to avoid the problem of underfitting NP models are commonly encountering.

Intuitively, the NP methodology explicitly trains the surrogate model to construct predictions on target states given a behavioral context. As described above, the model constructs its predictions not only based on the user/task properties θ but also considers the latent variable \mathbf{z} . Here, the latent is explicitly enforced to summarize the underlying behavioral policy instantiation π_ϵ in terms of ϵ . As such, the model may be conditioned on a behavioral context and user/task-specific parameters, θ , to represent a user model which has been explicitly adapted to a particular user/task and the corresponding instantiation of \mathcal{P}_θ . Furthermore, this adaptation does not require any model updates and may be achieved during online interaction.

Finally, it should be noted that conditioning the surrogate model on behavioral contexts may allow data-efficient generalization for previously unseen θ , as will be demonstrated in the experimental section.

3.4 Online Interaction

After a differentiable surrogate has been trained based on priors over a specific user/task population, it may be applied in online settings with previously unseen users/tasks. At the beginning of an interaction setting, the user model generally corresponds to a population-specific prior implemented by the differentiable surrogate. As the interaction setting evolves, and actual user-specific behaviour becomes available, the surrogate model may condition its latent distribution, $p(\mathbf{z} \mid \mathbf{x}, \mathbf{y}, \theta)$, on the observations. Respectively, sampling from the latent allows for constructing the likelihood $p(\mathbf{y} \mid \mathbf{x}, \theta, \mathbf{z})$ with respect to user/task-specific properties θ . The obtained likelihood may now be combined with a prior over θ to construct a target posterior. As the likelihood is here implemented by a differentiable model (and hence gradients of the target distribution are available), point estimates for θ may be obtained. Methods, such as Hamiltonian Monte Carlo (HMC), for obtaining posterior samples over θ are also applicable.

The obtained θ may be utilized to condition the user model together with the observed behavior. As new behavior is observed, the inference step may be repeated for continuous updates over θ . It should be noted that the steps mentioned for adapting the user model w.r.t. both θ and ϵ do not require any actual model updates and can thus be conducted during online interaction.

The population prior used for training the surrogate might not necessarily capture the variation regarding individual instantiations of \mathcal{P}_θ perfectly, or it might feature other biases. As the system interacts with numerous users, new behavior data generated by the true generative process becomes available. Repeating the offline training process by utilizing the obtained user data either directly or by refining the instance-specific variation on the prior model may allow the surrogate to move beyond the biases of the original population model.

To conclude this chapter, the proposed methodology constructs differentiable user models as surrogates of existing generative processes for user behavior. The proposed approach makes few assumptions regarding modeling scenarios it may be applied to and suits for modeling a wide range of different generative processes of user behavior. Furthermore, the user models are compatible with gradient-based methods allowing utilization of existing efficient computational infrastructures. Finally, the approach can adapt to

individual instantiations of generative processes of user behavior, offering robustness in practical scenarios. The following Chapters evaluate the proposed method in two modeling scenarios.

Chapter 4

Experiment 1: Gridworld Environment

The experimental part of this work consists of two separate experiment scenarios and is divided into two separate chapters accordingly. Both scenarios primarily aim to experiment on the ability of differentiable user models to construct the likelihood for user-specific behavior as a function of user/task parameters while conditioned on observed behavior. This metric evaluates the ability of the proposed differentiable user model to imitate behavior generative processes. More importantly, the metric acts as a proxy for the ability of the differentiable user to learn models of individual users. Thus, the experiments act as a feasibility study regarding the method’s applicability to realistic user modeling scenarios incorporating parameter inference. Evaluating the method regarding its practical ability to infer user/task-specific parametrizations is left for future work.

4.1 Setting

The first experiment scenario builds on a simple 10x10 gridworld. The objective of this experiment is to apply the proposed method for modeling MCTS-agents acting on the gridworld. While the scenario does not directly act as a meaningful user modeling scenario, it still features similarities with typical user modeling scenarios. In particular, analogously to realistic interactive settings, each interaction scenario is described by a behavior generative process, here MCTS, parametrized by the user/task-specific properties: reward function, action noise, and MCTS parameters. Hence, this experiment aims to act as a proof of concept regarding the viability of the proposed method.

Figure 4.1 illustrates examples of the gridworlds used in the experiment.

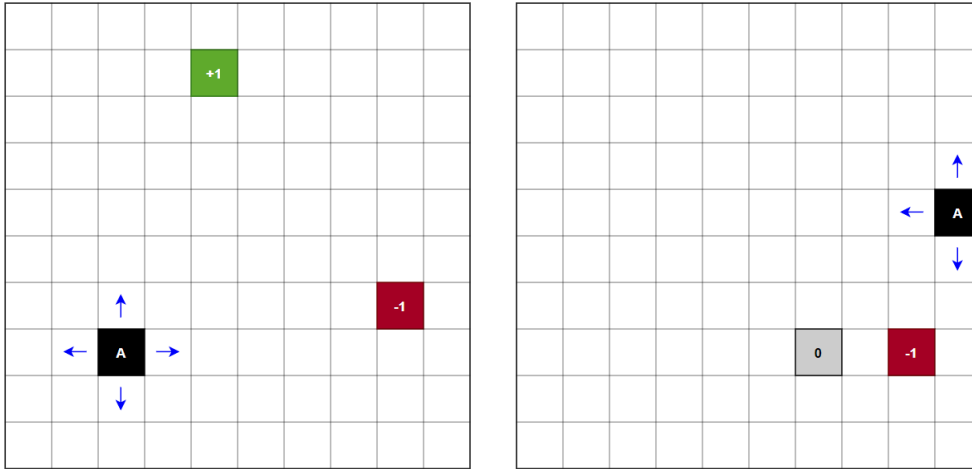


Figure 4.1: Two examples of possible 10x10 grid-worlds generated for this experiment. The black squares with "A" denote the current location of the agent. The coloured squares represent the reward states with their respective rewards (the gray square represents a reward state with zero reward). The blue arrows denote the available actions for the agent.

Each agent is characterized by a reward function describing locations and reward values for a fixed number of reward states. The agents can relocate from their current coordinates into adjacent squares, i.e., the action space consists of four possible actions. Based on the MDP spanned by the given description, the MCTS agents are assigned to plan and execute a trajectory that maximizes the expected obtained reward. Each episode terminates upon reaching a reward state or when a maximum number of actions is exceeded.

In this experiment, the surrogate model is trained to imitate MCTS as a generative process for user behavior. In particular, the environment induces the problem of modeling individual MCTS agents in terms of their reward functions, MCTS parametrizations, and the underlying action noise, which are all collected in θ .

The surrogate model is trained on simulated agents uniformly sampled from the parameter intervals presented in Table 4.1. Here, it is assumed that the prior utilized for simulating agents captures the ground-truth generative process accurately. After the agent's initial state is sampled, the agent alternates between planning and acting until the episode terminates. Each agent constructs multiple trajectories with different initial states. The trajectories are then permuted n times such that a random state-action -pair is selected as the target from the 'latest' (i.e., current) trajectory ordered by the per-

Table 4.1: Uniform prior on user model parameters for the user population in gridworld experiment.

User Parameter	Values
Reward State (x, y)	$\{1, 2, \dots, 10\}$
Reward Value	$\{-1, 0, 1\}$
Action Noise	$[0.00, 0.25]$
Exploration Constant	$[1.0, 10.0]$
Tree Depth	$[10, 20]$

mutation. Here, the context behavior corresponds to all previous trajectories and all transitions of the current trajectory preceding the target transition.

4.2 Implementation

The NP-surrogate is implemented by modifying the `NeuralProcesses.jl` library, a Julia variant of the neural process library of Dubois et al. (2020), to match the problem formulation discussed in previous sections. The data for training the surrogate model is generated by utilizing `POMDPs.jl` (Egorov et al. 2017).

The model is trained on a single GPU for 150 epochs. The data utilized at each epoch is generated independently of other epochs. Each epoch consists of 256 batches of data, each containing 8 individual users with 50 trajectories of length 10. Each epoch took approximately 2 minutes resulting in an overall training time of roughly 5 hours. However, the majority of the training time was spent on generating the training data. The overall training procedure, including model architecture and its hyperparameters, were not fully optimized - the training and selection of model architecture details were conducted by simply making choices that seemed reasonable.

4.3 Results

The results obtained on the proposed model, NP-surrogate, are compared against its simplified counterpart, MLP-surrogate, and two baseline models. In particular, comparison against the baseline models should reflect the absolute predictive performance of the model. On the other hand, contrasting the results with the MLP-surrogate aims to emphasize the importance of building the surrogate specifically as a latent-variable model capable of adapting to behavioral contexts. The two baseline models used in this experiment are

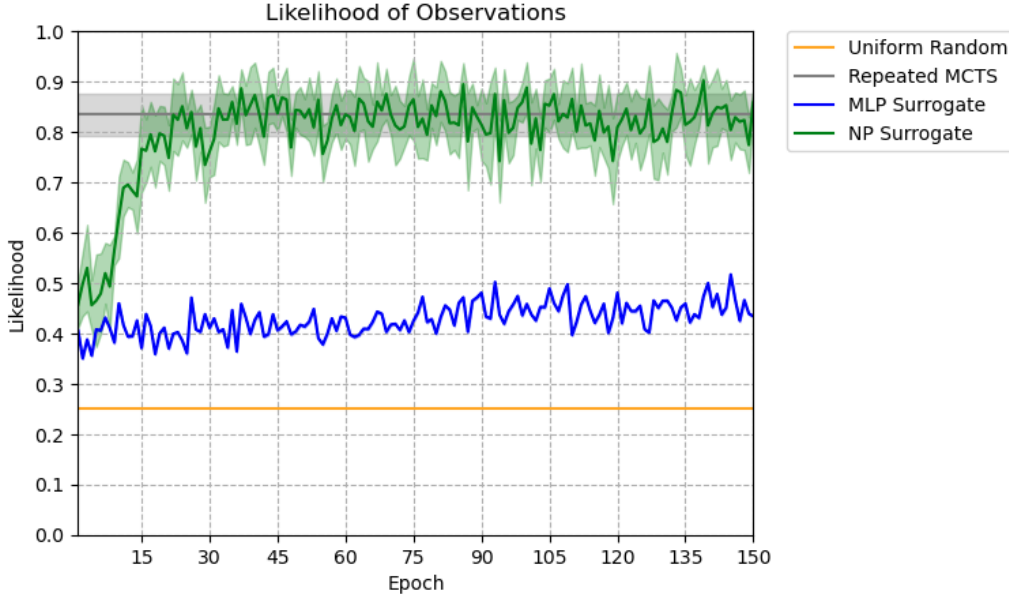


Figure 4.2: Visualization of performance of the two surrogate models and two baseline models on predicting the behaviour generated by MCTS planners.

Table 4.2: Summary on evaluation results after convergence (i.e. epochs 40-150).

Model	Likelihood
NP Surrogate	0.829 ± 0.057
MLP Surrogate	0.429 ± 0.002
Repeated MCTS	0.835 ± 0.042
Uniform Random	0.250 ± 0.000

a random classifier (predicting equal chance for all actions) and an MCTS predictor, which repeats the planning steps conducted by the ground truth MCTS using the original user parameters θ .

Figure 4.2 summarizes the obtained results. The figure illustrates the development of the evaluation performance of both surrogate models during training over 150 epochs. For the repeated MCTS baseline, the figure summarizes the obtained results over predictions on 5000 trajectories. Here, the metric used for describing the results is the likelihood of ground truth actions¹ at ground-truth θ under different models. The models are evaluated on simulated users, which were not utilized for training the models.

¹Although less descriptive than, e.g., KL-divergence, this metric is chosen due to its cheap computation and effortless interpretability.

It can be observed that after around 40 training epochs, the NP surrogate achieves performance almost similar to the repeated MCTS baseline. This baseline may be interpreted to approximate the maximum performance that may be obtained in the experiment. This indicates that the NP surrogate has converged into nearly the best possible performance in terms of the likelihood metric. This observation may be considered a surprisingly positive result given that the model architecture, training procedure, or any of the corresponding hyperparameters are not fully optimized for this experiment.

This experiment is also repeated for a similar surrogate model that utilizes a simple MLP with one hidden layer as a basis instead of an NP. This variant acts as a small-scale ablation study to evaluate the importance of conditioning the model on observed context. In particular, it can be observed that the MLP surrogate overall seems to fail to generalize for previously unseen agents resulting in relatively poor performance.

4.4 Discussion

The obtained results indicate that the proposed method may be successfully utilized for constructing individualized models in the given scenario. However, to fully understand the implications of the results, a specific property of MCTS as a generative process must be first reviewed.

As discussed, the proposed models are explicitly encouraged to adapt to individual instantiations of the underlying generative process \mathcal{P} specified by ϵ . However, the approach for achieving this adaptation assumes that behavioral contexts and future behaviors are consistent with particular instantiations of the generative process (that is, the actions are generated with singular π_ϵ).

As it turns out, this assumption is not satisfied in this experiment: MCTS is a planning algorithm that generates new policies at each time step. Consequently, behavioral contexts do not contain information regarding the policies that are responsible for future actions. The symmetries² induced by the grid-world MDP guarantee that there exists variation between multiple repeated MCTS planning steps with fixed θ .

In concrete terms, the property of MCTS mentioned above implies that achieving perfect predictive performance and thus learning perfect user models should be virtually impossible in the given experiment setting. This is similarly reflected in the results on the repeated MCTS baseline, which achieves an averaged likelihood of "only" 0.835. Given that here the modeling scenario offers no information to reduce the uncertainty regarding instantiations of \mathcal{P}_θ ,

²For instance, it is usual that the agent may have multiple equally optimal trajectories towards a reward state.

the repeated MCTS baseline may be considered as the model with the highest possible predictive performance for user behavior if parametrized with correct θ .

In conclusion, the obtained results for the NP-surrogate imply that the proposed differentiable user models are capable of representing the ground-truth generative process for user behavior almost perfectly if contrasted with the baseline. Furthermore, the proposed models may be utilized for inferring θ , which is not possible with the baseline. Although the proposed method is not explicitly evaluated in terms of inference, the ability of the surrogate to represent behavior generative processes acts as a proxy for the quality of inference. Finally, comparison with the MLP-surrogate demonstrates how conditioning on behavioral contexts improves the model’s ability to generalize for previously unseen agents with limited contextual data.

Chapter 5

Experiment 2: Menu Search Environment

5.1 Setting

The second experiment setting is based on the Menu Search Model of Kangasrääsiö et al. (2017), an extended version of the model originally proposed by Chen et al. (2015). Their work essentially describes a model of user search behavior when searching for a target item from a dropdown menu. The model is heavily influenced by *computational rationality* (Lewis et al. 2014), a framework for cognitive modeling, which is characterized by an assumption that behavioral policies of rational agents are optimal concerning their cognitive architecture. Respectively, the menu search model considers users as Q-learning agents, which optimize their search behaviour, i.e. eye movements, with respect to their cognitive properties: *eye fixation duration* (f_{dur}), *target item selection delay* (d_{sel}) and *menu recall probability* (p_{rec}).

The underlying MDP used by the menu search model is defined in terms of the menu-related information available to the user, the current focus (i.e., the element the user is looking at), and information regarding whether or not the user has quit. The menu considered in this model consists of eight menu elements. Each element is described in terms of its semantic relevance and length in comparison to the target word. The user takes sequential actions to fixate on different menu elements. Fixating on an element has a chance to reveal its semantic relevance and length while also having a chance to reveal the same information on the adjacent items via peripheral vision. In addition, when entering the menu (i.e., when taking the first action), there is a chance, p_{rec} , that the user recalls the menu layout from memory, revealing all semantic relevances and lengths of the menu elements. The user is given a

Table 5.1: Distributions for user cognitive properties used in the second experiment (proposed by Kangasrääsio et al. (2019)).

User Parameter	Values
p_{rec}	Beta(3.0, 1.35)
f_{dur}	$\mathcal{N}(3.0, 1.0)$
d_{sel}	$\mathcal{N}(0.3, 0.3)$

negative reward based on the duration taken by its actions (the durations are specified by cognitive properties). If an action results in fixating on the target word, the element is automatically selected, and a significant positive reward is given. The target word of the user is present in 90% of the generated menus. If the user can recognize that the target word is not present and quits the scenario, a large reward is given.

In this experiment, the surrogate model is trained to imitate Q-learning as a generative process for user search behavior. The setting induces the problem of modeling individual Q-learners in terms of their cognitive parameters and target words. The cognitive properties are modeled, together with the target words of users, as user/task-specific parameters θ . In terms of the population model used in this experiment, the parameter distributions follow the suggestions of Kangasrääsio et al. (2019) and are summarized in Table 5.1.

The NP-surrogate is trained similarly as in the first experiment: individual users are sampled according to $\theta \sim p(\theta)$ to generate behavioral trajectories, which are then utilized for training the surrogate. For each model simulation, a new menu, together with its element-wise information w.r.t. the target word, is sampled according to the menu search environment proposed by the authors.

In contrast to the original setting, the users (i.e., Q-learning agents) are trained only until approximately optimal instead of entirely optimal. This design choice is taken to keep the time required for data generation feasible but, more importantly, to deliberately induce variance in the generative process for user behavior. As it turns out, this variance is particularly meaningful as it further emphasizes the ability of the proposed differentiable user models to adapt to individual instantiations of the generative process. The behavior generative process (Q-learning) is utilized once to generate one behavioral policy π_ϵ , which is used for the complete trajectory. Hence, the behavioral context and future actions are consistent with a specific instantiation of the generative process enabling adaptation in this experiment.

5.2 Implementation

Similar to the first experiment, the NP-surrogate is implemented with the modified `NeuralProcesses.jl` library. The MDP environment is implemented by translating the original Python-based implementation into Julia utilizing the aforementioned `POMDPs.jl` library (Egorov et al. 2017). The actual training of simulated users was implemented with the deep Q-learning library `DeepQLearning.jl` and utilized double Q-learning (Van Hasselt et al. 2016), dueling architecture (Wang et al. 2016) and prioritized experience replay (Schaul et al. 2015). To speed up the training process of individual agents further, each agent began their training with a pre-trained Q-network trained with average cognitive parameters w.r.t. the prior. The agents are encouraged to learn a policy corresponding to their actual parameters instead of only adopting the policy of the pre-trained Q-network.

For this experiment, the NP surrogate is pre-trained for 60 epochs. Each epoch consists of 128 batches of data, each comprising of 8 individual users with 5 pairs of *context-target* trajectories of length 5. The pairs are permuted such that each trajectory is utilized once as the context and once as the target. Training the model for one epoch took approximately 12 minutes, of which the majority of the time was spent generating the data. As in the first experiment, the training procedure was not fully optimized.

5.3 Results

Analogously to the first experiment, the proposed NP surrogate is evaluated regarding the likelihood of ground truth observations at ground-truth parameters and compared against its MLP counterpart alongside two baseline models. The baseline models correspond to a repeated deep Q-learning predictor and a nearest neighbor deep Q-learning predictor. Here, the former baseline corresponds to repeating the ground truth generative process with the correct user parameters. The latter is otherwise similar except that the predictions are conducted by the nearest model (in terms of Euclidean distance w.r.t. θ) among the 27 pre-trained models distributed evenly along with the parameter space.

The obtained results are visualized in Figure 5.1, which summarizes the evaluation performance of both surrogate models over the training. The figure illustrates the most important result of this experiment: the NP surrogate exceeds the predictive performance of the ground truth generative process (repeated deep-Q). In contrast to the first experiment, the modeling problem induced by the menu search model is more straightforward, allowing

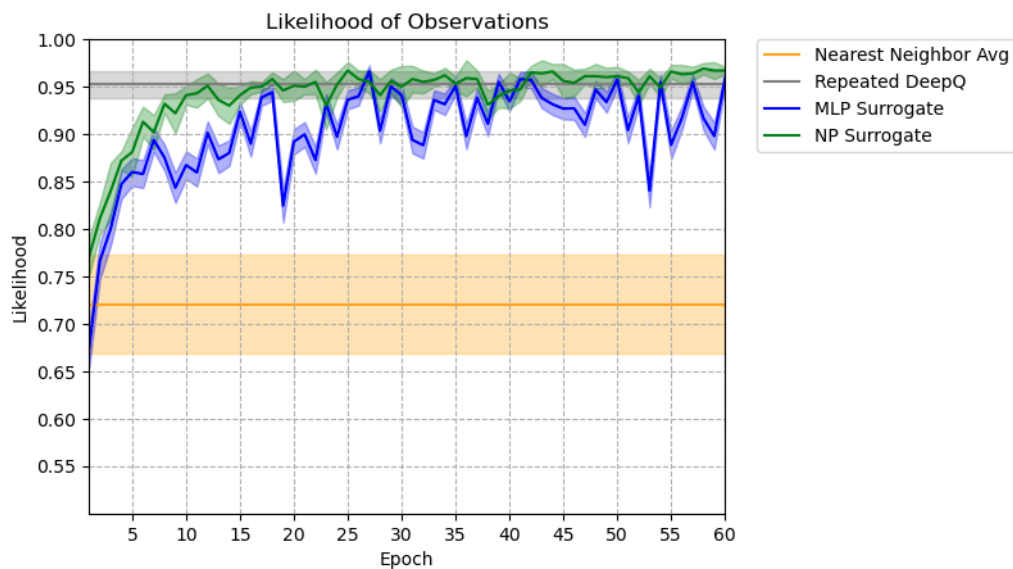


Figure 5.1: Visualization of performance of the two surrogate models and two baseline models on predicting the behaviour generated by deep Q-learning agents.

Table 5.2: Summary on evaluation results after convergence.

Model	Likelihood
NP Surrogate	0.961 ± 0.008
MLP Surrogate	0.921 ± 0.012
Repeated DeepQ	0.953 ± 0.014
NN Avg	0.725 ± 0.049

the MLP surrogate to generalize w.r.t. previously unseen θ to some extent. It can be observed that the nearest neighbor deep Q-learning achieves the worst results.

5.4 Discussion

The most prominent result of this experiment is that the NP-surrogate manages to surpass the performance of the repeated deep-Q baseline in terms of the likelihood metric. As discussed, this is essentially a result of the NP-surrogate’s ability to represent distributions over the instantiations of generative processes conditioned behavioral contexts. In contrast, repetitions of the ground-truth generative process may produce behavioral policies inconsistent with the appropriate contexts.

Similar conditioning is, in principle, available also for the deep-Q baseline in terms of accepting only predictions with policies consistent with the observed context. This solution, however, comes with an expensive and inconsistent computational cost - for lengthy contexts generated with ‘rare’ policies, the conditioning step may require an immense number of training attempts. After all, the repeated deep-Q baseline is already infeasible for inference tasks even without the conditioning step.

Comparison between the results of the NP and MLP -surrogates motivates constructing the surrogates specifically as NP-based latent variable models. As discussed, the NP-surrogate may adapt to instantiations of \mathcal{P} , which is not possible for the MLP-surrogate. Consequently, while the NP-surrogate may exceed the predictive performance of the repeated deep-Q baseline, the baseline still acts as a limit for the MLP-surrogate.

It should be noted that despite the observed results, adapting to individual instantiations of \mathcal{P} is still relatively unimportant in this experiment - the variance between individual policies generated by \mathcal{P}_θ with fixed θ remains, after all, relatively small. In contrast, consider the same menu search environment used in this experiment but with a population model that does not label the generated behavior in terms of the user-specific cognitive properties. Suddenly, the variation between instantiations of the generative process features the stochasticity of the utilized deep Q-learning and the uncertainty regarding the underlying cognitive properties. Figure 5.1 illustrates how the nearest neighbor deep-Q baseline already has a significant performance decrease when compared to the repeated deep-Q baseline due to not utilizing the exact values of the cognitive properties - this is an optimistic scenario if compared to complete uncertainty w.r.t. these parameters.

In conclusion, the obtained results demonstrate the importance of cap-

turing the uncertainty regarding instantiations of behavior generative processes. Furthermore, this experiment demonstrated that this property effectively allows the surrogate model to surpass the predictive performance of the ground-truth generative process, assuming that the correct user parameters θ were available. While the experiment did not evaluate the proposed method in terms of inferring θ , the obtained results should again translate to the inference quality.

Chapter 6

Discussion

This chapter reviews and compares the proposed methodology to similar contemporary methods. In addition, this section discusses its limitations and directions for future research.

6.1 Related Work

To the author’s knowledge, this work distinguishes from previous approaches by representing the first method for learning differentiable user models that is relatively agnostic in terms of user modeling scenarios it may be applied to. In particular, the contribution of this thesis may be distinguished from previous research at least by one of the following aspects:

- Utilizability for modeling a wide range of different generative process and user/task-specific properties
- Differentiability, i.e., compatibility with gradient-based methods
- Ability to utilize experience from previous users/tasks for adapting to previously unseen scenarios with little data
- Ability to learn user models that may be conditioned on behavioral contexts to adapt to instantiations of behavior generative processes.

In addition, this work distinguishes from many similar works due to its explicit focus on user modeling.

Approaches such as Value Iteration Networks (Tamar et al. 2016) and MCTSnets (Guez et al. 2018) implement traditional algorithmic methods as differentiable structures. Although these approaches are not explicitly

designed for user modeling, they should be reasonably straightforwardly utilizable for user modeling tasks. Similar to this work, the differentiability of these models would allow gradient-based inference and thus individualized user modeling. However, these approaches are limited in applicability as they explicitly assume specific underlying generative processes and user/task-specific properties θ . In addition, these models may not necessarily be utilizable for adapting to individual instantiations of generative processes.

The proposed approach shares numerous similarities with methodologies such as Inverse Reinforcement Learning (IRL) (Ng et al. 2000) and Imitation Learning (IL) (Hussein et al. 2017). The former (IRL) intuitively aims to find reward functions explaining the observed agent’s behavior. It has been previously proposed for specific user modeling tasks, e.g., by (Chandramohan et al. 2011), but is, in the end, limited by that it may be utilized for modeling individual users only in terms of their reward functions. Similar limitations seem to be present also in other previous approaches not explicitly utilizing IRL. For instance, the approach of Reddy et al. (2020) learns human objectives by evaluating hypothetical behavior.

Imitation learning, commonly applied in robotics, considers building models to imitate human (expert) behavior on a given task. IL explicitly emphasizes learning policies that may generalize into previously unseen tasks (Hussein et al. 2017). Duan et al. (2017) demonstrate a one-shot imitation learning method in the context of robotic control tasks, which may adapt into new tasks based on only a few context observations. Although their approach operates on a different domain (robotics) than this thesis, their method utilizes numerous similar intuitions and ideas, including constructing context embeddings.

Computational methods motivated by cognitive science represent a common approach for modeling human behavior and inferring human features. The work of Acerbi et al. (2014) considers computational models for explaining human suboptimality in the context of probabilistic inference. Approaches of Kangasrääsio et al. (2017) and Kangasrääsio et al. (2019) utilize the notion of *computational rationality* (Lewis et al. 2014) for justifying modelling user population as agents optimizing their behaviour w.r.t. their cognitive properties. Furthermore, they utilize methods such as Bayesian Optimization (BO) and Approximate Bayesian Computation (ABC) for conducting inference on user-specific cognitive properties on the menu search environment discussed before based on the behavioral data collected by Bailly et al. (2014). However, in contrast to the method proposed in this thesis, these methods are not fully compatible with gradient-based methods.

Theory of Mind (ToM) (Premack & Woodruff 1978) is a branch of cognitive science that aims to model how humans can recognize mental states,

such as goals and beliefs, of oneself and others based on their actions. During recent years, this branch has motivated numerous computational approaches (such as Baker et al. (2011), Rabinowitz et al. (2018) and Peltola et al. (2018)) for transferring this ability also to computational systems. In terms of this thesis, probably the most relevant of these approaches is, however, the Machine Theory of Mind (Rabinowitz et al. 2018). Although they are not directly aiming for user modeling, they similarly construct a differentiable model utilizing latent embeddings for obtaining summaries for agent characterizations and mental states. Their approach emphasizes the meta-learning angle, and their method may be simultaneously utilized for predicting trajectories and mental states of agents without explicit inference steps. Their solution is also capable of modeling stateful agents.

Galashov et al. (2019) build surrogate models for sequential decision making, such as model-based RL, with neural processes. Their approach shares numerous similarities with the proposed differentiable user models but, similarly to the case of Machine Theory of Mind, is not explicitly targeted for user modeling.

6.2 Limitations & Directions for Future Research

One of the limitations of the proposed method is that it assumes that prior models over the generative processes over the user/task population are available for training the surrogate. However, settings and applications with sufficient prior user data feature an exciting direction for future research. In principle, given sufficient data, the proposed NP surrogate could be trained thoroughly without explicit prior modeling of the user population in terms of user/task-specific properties θ . All variation present in the data could be possibly captured by the latent through the observed behavioral context. As such, the model would not require any explicit inference steps with respect to θ . If the underlying application scenario, however, features interest towards obtaining information regarding individual users, exploring, e.g., representation learning (Bengio et al. 2013) in the context of this work would provide an interesting direction for future research. For instance, enforcing disentanglement on the latent, as in the work of Rabinowitz et al. (2018), may provide tools for extracting user-specific information based on only observed behavior.

The proposed method does not fully support the modeling of stateful agents. This property may be considered relevant as in various interactive scenarios, the individual properties of users, such as objectives and plans, might evolve during the interaction. User models should be able to cap-

ture, for instance, possible changes in user objectives. The current approach should, however, be utilizable for modeling stateful behavior within certain limits. In particular, the latent may be enforced to summarize the user’s current state based on the previous behavior on the same trajectory before the prediction. However, this approach might fail to adequately capture the sequential nature and possible recurrent structures of the behavioral data. The works of Willi et al. (2019) and Qin et al. (2019) already extend the structures of neural processes and attentive neural processes with an ability to capture temporal structures. Utilizing these approaches might allow for extending the proposed method also for modeling stateful users.

In terms of experimentation, the proposed methodology is explored from a relatively narrow perspective. In particular, the obtained results indicate that the proposed methodology may indeed be utilized for learning individualized models in terms of their ability to produce similar trajectories as the ground truth processes. However, the experiments do not provide information on sensitivity to prior population models. Ideally, the model should be able to move beyond the possible inaccuracies of the prior and adapt to actual user populations if re-trained offline on the data obtained through user interaction. The experiments, however, do not explicitly explore this direction and, hence, do not provide any concrete indications on the actual amounts of data required for this adaptation. Consequently, an exciting future direction would be to apply the proposed method on a full-scale interactive scenario, possibly featuring distinct models for ground-truth user population and population prior.

Chapter 7

Conclusions

This thesis introduced a methodology for learning differentiable user models. The method asserts few assumptions regarding the possible user modeling scenarios it may be applied to. The method’s main limitation is that it assumes that priors over the user and task populations are available with sufficient accuracy. The approach should ultimately allow the learned models to move beyond possible biases of prior population models by utilizing behavioral data obtained from the actual user population.

The results obtained on the experimental part of the thesis suggest that the differentiable user models trained with the proposed methodology may achieve excellent performance in various user modeling tasks. Specifically, it was shown that the differentiable user models learn to imitate generative processes of user behavior accurately. Furthermore, the obtained results indicate that the models can utilize experience from previous users and tasks to generalize to new, previously unseen user/task-specific parameters with little data. Finally, it was found that the models benefit from conditioning on observed behavior contexts. In particular, the learned models can model the uncertainty regarding instantiations of behavior generative processes when conditioned with behavioral contexts. Hence, the method may be utilized for learning individualized user models that take, for instance, the stochasticity of reinforcement learning algorithms into account.

The thesis conducted a small-scale ablation study to explore the further benefits of contextual conditioning. This study discovered that, while contextual conditioning is not always strictly necessary in simple scenarios, it allows the differentiable user models to utilize knowledge from previous users and tasks to generalize for unseen task/user-specific properties with limited data.

The experimental section concentrated on exploring the performance of the differentiable user models in terms of their ability to imitate ground truth

generative processes for user behavior. The obtained results essentially act as a proxy for the performance in terms of inference and modeling performance during online interaction, effectively indicating success also for these tasks. An interesting direction for future work would be to explore the method in complete interactive scenarios explicitly. Finally, it should be noted that, in practice, the method is evaluated only in terms of two alternative experiment settings. Hence, this work leaves a more comprehensive study regarding its applicability in different user modeling scenarios for future research.

Bibliography

- Acerbi, L., Vijayakumar, S. & Wolpert, D. M. (2014), ‘On the origins of suboptimality in human probabilistic inference’, *PLoS Comput Biol* **10**(6), e1003661.
- Bailly, G., Oulasvirta, A., Brumby, D. P. & Howes, A. (2014), Model of visual search and selection time in linear menus, *in* ‘Proceedings of the sigchi conference on human factors in computing systems’, pp. 3865–3874.
- Baker, C., Saxe, R. & Tenenbaum, J. (2011), Bayesian theory of mind: Modeling joint belief-desire attribution, *in* ‘Proceedings of the annual meeting of the cognitive science society’, Vol. 33.
- Bellman, R. (1957), ‘A markovian decision process’, *Journal of mathematics and mechanics* **6**(5), 679–684.
- Bengio, Y., Courville, A. & Vincent, P. (2013), ‘Representation learning: A review and new perspectives’, *IEEE transactions on pattern analysis and machine intelligence* **35**(8), 1798–1828.
- Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S. & Colton, S. (2012), ‘A survey of monte carlo tree search methods’, *IEEE Transactions on Computational Intelligence and AI in games* **4**(1), 1–43.
- Chandramohan, S., Geist, M., Lefevre, F. & Pietquin, O. (2011), User simulation in dialogue systems using inverse reinforcement learning, *in* ‘Twelfth Annual Conference of the International Speech Communication Association’.
- Chaslot, G., Winands, M., Uiterwijk, J., Van Den Herik, H., Bouzy, B. & Wang, P. (2007), Progressive strategies for monte-carlo tree search, *in* ‘Proceedings of the 10th Joint Conference on Information Sciences (JCIS 2007)’, pp. 655–661.

- Chen, X., Bailly, G., Brumby, D. P., Oulasvirta, A. & Howes, A. (2015), The emergence of interactive behavior: A model of rational menu search, *in* ‘Proceedings of the 33rd annual ACM conference on human factors in computing systems’, pp. 4217–4226.
- Duan, Y., Andrychowicz, M., Stadie, B. C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P. & Zaremba, W. (2017), ‘One-shot imitation learning’, *arXiv preprint arXiv:1703.07326* .
- Dubois, Y., Gordon, J. & Foong, A. Y. (2020), ‘Neural process family’, <http://yanndubs.github.io/Neural-Process-Family/>.
- Egorov, M., Sunberg, Z. N., Balaban, E., Wheeler, T. A., Gupta, J. K. & Kochenderfer, M. J. (2017), ‘POMDPs.jl: A framework for sequential decision making under uncertainty’, *Journal of Machine Learning Research* **18**(26), 1–5.
URL: <http://jmlr.org/papers/v18/16-300.html>
- Fischer, G. (2001), ‘User modeling in human–computer interaction’, *User modeling and user-adapted interaction* **11**(1), 65–86.
- Galashov, A., Schwarz, J., Kim, H., Garnelo, M., Saxton, D., Kohli, P., Eslami, S. & Teh, Y. W. (2019), ‘Meta-learning surrogate models for sequential decision making’, *arXiv preprint arXiv:1903.11907* .
- Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S. & Teh, Y. W. (2018), ‘Neural processes’, *arXiv preprint arXiv:1807.01622* .
- Guez, A., Weber, T., Antonoglou, I., Simonyan, K., Vinyals, O., Wierstra, D., Munos, R. & Silver, D. (2018), Learning to search with mctsnet, *in* ‘International Conference on Machine Learning’, PMLR, pp. 1822–1831.
- Hornik, K. (1991), ‘Approximation capabilities of multilayer feedforward networks’, *Neural networks* **4**(2), 251–257.
- Hospedales, T., Antoniou, A., Micaelli, P. & Storkey, A. (2020), ‘Meta-learning in neural networks: A survey’, *arXiv preprint arXiv:2004.05439* .
- Hussein, A., Gaber, M. M., Elyan, E. & Jayne, C. (2017), ‘Imitation learning: A survey of learning methods’, *ACM Computing Surveys (CSUR)* **50**(2), 1–35.

- Kangasrääsio, A., Athukorala, K., Howes, A., Corander, J., Kaski, S. & Oulasvirta, A. (2017), Inferring cognitive models from data using approximate bayesian computation, *in* ‘Proceedings of the 2017 CHI conference on human factors in computing systems’, pp. 1295–1306.
- Kangasrääsio, A., Jokinen, J. P., Oulasvirta, A., Howes, A. & Kaski, S. (2019), ‘Parameter inference for computational cognitive models with approximate bayesian computation’, *Cognitive science* **43**(6), e12738.
- Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O. & Teh, Y. W. (2019), ‘Attentive neural processes’, *arXiv preprint arXiv:1901.05761* .
- Kingma, D. P. & Welling, M. (2013), ‘Auto-encoding variational bayes’, *arXiv preprint arXiv:1312.6114* .
- Lewis, R. L., Howes, A. & Singh, S. (2014), ‘Computational rationality: Linking mechanism and behavior through bounded utility maximization’, *Topics in cognitive science* **6**(2), 279–311.
- McTear, M. F. (1993), ‘User modelling for adaptive computer systems: a survey of recent developments’, *Artificial intelligence review* **7**(3), 157–184.
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. & Riedmiller, M. (2013), ‘Playing atari with deep reinforcement learning’, *arXiv preprint arXiv:1312.5602* .
- Ng, A. Y., Russell, S. J. et al. (2000), Algorithms for inverse reinforcement learning., *in* ‘Icml’, Vol. 1, p. 2.
- Papatheodorou, C. (1999), Machine learning in user modeling, *in* ‘Advanced Course on Artificial Intelligence’, Springer, pp. 286–294.
- Peltola, T., Celikok, M. M., Daege, P. & Kaski, S. (2018), ‘Modelling user’s theory of ai’s mind in interactive intelligent systems’, *arXiv preprint arXiv:1809.02869* .
- Premack, D. & Woodruff, G. (1978), ‘Does the chimpanzee have a theory of mind?’, *Behavioral and brain sciences* **1**(4), 515–526.
- Qin, S., Zhu, J., Qin, J., Wang, W. & Zhao, D. (2019), ‘Recurrent attentive neural process for sequential data’, *arXiv preprint arXiv:1910.09323* .

- Rabinowitz, N., Perbet, F., Song, F., Zhang, C., Eslami, S. A. & Botvinick, M. (2018), Machine theory of mind, *in* ‘International conference on machine learning’, PMLR, pp. 4218–4227.
- Reddy, S., Dragan, A., Levine, S., Legg, S. & Leike, J. (2020), Learning human objectives by evaluating hypothetical behavior, *in* ‘International Conference on Machine Learning’, PMLR, pp. 8020–8029.
- Ruder, S. (2016), ‘An overview of gradient descent optimization algorithms’, *arXiv preprint arXiv:1609.04747* .
- Schaul, T., Quan, J., Antonoglou, I. & Silver, D. (2015), ‘Prioritized experience replay’, *arXiv preprint arXiv:1511.05952* .
- Sutton, R. S. & Barto, A. G. (2018), *Reinforcement learning: An introduction*, MIT press.
- Tamar, A., Wu, Y., Thomas, G., Levine, S. & Abbeel, P. (2016), ‘Value iteration networks’, *arXiv preprint arXiv:1602.02867* .
- Van Hasselt, H., Guez, A. & Silver, D. (2016), Deep reinforcement learning with double q-learning, *in* ‘Proceedings of the AAAI Conference on Artificial Intelligence’, Vol. 30.
- Vanschoren, J. (2018), ‘Meta-learning: A survey’, *arXiv preprint arXiv:1810.03548* .
- Vilalta, R. & Drissi, Y. (2002), ‘A perspective view and survey of meta-learning’, *Artificial intelligence review* **18**(2), 77–95.
- Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M. & Freitas, N. (2016), Dueling network architectures for deep reinforcement learning, *in* ‘International conference on machine learning’, PMLR, pp. 1995–2003.
- Watkins, C. J. C. H. (1989), ‘Learning from delayed rewards’.
- Watkins, C. J. & Dayan, P. (1992), ‘Q-learning’, *Machine learning* **8**(3-4), 279–292.
- Webb, G. I., Pazzani, M. J. & Billsus, D. (2001), ‘Machine learning for user modeling’, *User modeling and user-adapted interaction* **11**(1), 19–29.
- Willi, T., Masci, J., Schmidhuber, J. & Osendorfer, C. (2019), ‘Recurrent neural processes’, *arXiv preprint arXiv:1906.05915* .