Differentiable User Models

Abstract

Probabilistic user modeling is essential for building collaborative AI systems within probabilistic frameworks. However, modern advanced user models, often designed as cognitive behavior simulators, are computationally prohibitive for interactive use in cooperative AI assistants. We address this problem by introducing widely-applicable differentiable surrogates for bypassing this computational bottleneck; the surrogates enable using modern behavioral models with online computational cost which is independent of their original computational cost. We show experimentally that modeling capabilities comparable to the only available solution, existing likelihood-free inference methods, are achievable with a computational cost suitable for online applications. Finally, we demonstrate how AI-assistants can now use cognitive models for online interaction in a menu-search task, which has so far required hours of computation during interaction.

1 INTRODUCTION

User modeling constructs informative representations of individual users to enable computational systems to customize and adapt their behavior for them (Li and Zhao, 2020). It has been extensively studied over the years, also recently in recommender systems (Yu et al., 2019; Yuan et al., 2020), human-in-the-loop machine learning (Daee et al., 2018) and AI-assistants (Horvitz et al., 2013; Dafoe et al., 2021). Machine learning is needed in user modeling to infer user-specific information based on observed user behavior. Depending on the application, the inferred information can be the end result or, for instance, used to parameterize a user simulator to form predictions of user behaviors to guide the behavior of the system.

Traditionally, salient use cases of user modeling, such as recommendation engines, have utilized user-specific preference profiles based on users' history. However, these approaches are not sufficiently powerful in more complex interactive applications, where the user plans and interacts strategically, for instance in human-AI collaboration and human-in-the-loop decision-making. On the other hand, while recent ML research has shown significant success in learning accurate neural models directly from data, this is infeasible approach in user modeling in general, as typical user-driven applications are data starved. In contrast, recent approaches, e.g., Kangasrääsiö et al. (2019); Moon et al. (2022), have utilized advanced general-purpose behavioral models, based on cognitive science, in a Bayesian setting and received encouraging results with limited data. The probabilistic treatment of the problem enables taking uncertainty of the inferences into account — which fundamentally allows the system to balance between exploration and exploitation in interaction with the user.

A prominent body of these advanced cognitive models is based on computational rationality (Lewis et al., 2014; Gershman et al., 2015), which posits that seemingly irrational behaviors of humans are rational under their cognitive bounds. It follows that human behaviors can be accurately modeled as a result of RL-based optimization given that the underlying decision-theoretic framework and the optimization procedure are specified such that they accurately capture the appropriate bounds governing human cognition, such as the limits of computational capacity. A concrete example of such a model, which we consider in our experiments, is the model of rational menu search (Chen et al., 2015). This cognitive model describes human search behavior in terms of eye movements when searching for a target item in a computer dropdown menu, while encoding the limitations of human cognition and perception when processing visual information.

Despite their many benefits, these advanced cognitive models are currently not used beyond small-scale practical applications due to two important factors: (1) they are expressed

Submitted to the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023). **To be used for reviewing only**.

as non-differentiable simulators and hence incompatible with modern machine learning pipelines and (2) they are computationally infeasible to be used directly in realistic applications.

In this paper, we address these limitations: we enable computationally efficient probabilistic user modeling suitable for real-time applications — even with advanced cognitive models that lack a closed-form likelihood. We do this by introducing a way of learning differentiable surrogates for cognitive user models, and to generalize them over tasks and users through meta-learning. As a result, the surrogates become widely applicable with online computational cost independent of the complexity of the original models. The contributions of this work are:

- We propose reformulating and solving simulator-based modeling problems as meta-learning problems to enable online inference and simulation in user modeling.
- We demonstrate a flexible way of leveraging any existing user data during surrogate training to address possbile model misspecification in cognitive models.
- With neural processes as example surrogate models, we demonstrate comparable user modeling accuracy to current methods with a computational time suitable for online applications.

Our work removes a key computational bottleneck currently hindering incorporation of users into probabilistic programming models. Probabilistic user modeling based on cognitive models can now be applied widely without extensive computational budgets.

2 DIFFERENTIABLE USER MODELS

This work considers computationally efficient probabilistic user modeling for interactive settings, between a user from a user population and an interactive AI system. User modeling is brought in to guide user-specific adaptation of the behavior (policy) of such a system; user modeling is needed for (i) inference of user-specific information from observations of user behavior and (ii) user behavior prediction. The following subsections detail the specifics of current approaches and their limitations, together with our approach for addressing these limitations.

2.1 PROBABILISTIC USER MODELING WITH COGNITIVE MODELS

Following the intuition presented by Kangasrääsiö et al. (2019), we formulate the standard probabilistic user modeling setting as follows: a population of users is engaged with a distribution of decision-making tasks denoted by $p(\theta_T)$. Each modeling scenario involves a user $\theta_U \sim p(\theta_U)$ and is fully described by the respective user and task specific parameters $\theta = \{\theta_T, \theta_U\}$. The users are assumed to generate their policies π through an implicit process $\mathcal{P}_\theta$ which they execute to generate pairs of states and actions $(\mathbf{s}, \mathbf{a})$ observable by the modeling system. The system has access to a cognitive model $p(\pi \mid \theta)$ approximating the true process and a corresponding prior $p(\theta)$. The key task in probabilistic user modeling is the inference problem of approximating the posterior

$$p(\theta \mid (\mathbf{s}, \mathbf{a})) \propto p((\mathbf{s}, \mathbf{a}) \mid \pi) p(\pi \mid \theta) p(\theta). \tag{1}$$

For cognitive models, the likelihood $p(\pi \mid \theta)$, required for solving this Bayesian inference task, is typically not evaluable in closed-form due to the simulator-type nature of these models. So far, this issue has been circumvented by utilizing exclusively likelihood-free inference (LFI) methods, such as approximate Bayesian computation (ABC) (Sisson et al., 2018; Sunnåker et al., 2013) and Bayesian optimization for likelihood-free inference (BOLFI) (Gutmann et al., 2016), as proposed by Kangasrääsiö et al. (2019) and Moon et al. (2022). The basic idea of LFI is to replace the computationally expensive simulator $p(\pi \mid \theta)$ with an approximation that is separately learned on the observed data (Gutmann et al., 2016), in this case the $(\mathbf{s}, \mathbf{a})$ from each user. For user modeling, this approach has two problems: this process requires numerous computationally expensive simulations with the cognitive model and the data in typical user modeling applications often is too scarce for learning a new model independently for each user.

Moon et al. (2022) proposed circumventing the computational complexity by learning a generalizable policy-modulation network as a surrogate for the original model, i.e. $p(\pi \mid \theta)$, and obtained significant speed-ups for inference. However, as noted by the authors, their approach is still prohibited by the computational cost of LFI needed for approximating the posterior, and is not feasible for real-time inference. Similarly, the simulation costs of cognitive models $p(\pi \mid \theta)$ are often too expensive to enable prediction and online adaptation in real-time applications. Furthermore, even though LFI methods are developing fast, practical interactive settings may require hierarchical approaches for generalizing across the user models, which has been traditionally difficult with LFI (Turner and Van Zandt, 2014). An additional problem with LFI-based modeling is the sensitivity to model misspecification, which is very likely in user models.

2.2 SURROGATES FOR COGNITIVE MODELS

Computing the posterior $p(\theta \mid (\mathbf{s}, \mathbf{a}))$ and sampling from $p(\pi \mid \theta)$ are the Bayesian choices for user-specific inference and prediction. Our approach removes the limitations of current approaches by training generalizable amortized surrogates for both of these operations offline, such that

they are readily usable during interaction without extensive computation.

While this approach would solve the issue of online computational complexity, the offline complexity of training them will still be an issue due to the vast diversity of different behaviors the cognitive models are able to express. In particular, even if training a generalizable surrogate for the cognitive model would be achievable, as done in the work of Moon et al. (2022), training a surrogate directly for approximating the posterior would ultimately be computationally intractable as constructing the training data would require numerous repeated evaluations with LFI (for the reference, Kangasrääsiö et al. (2019) reported that even a single LFI result would require at least 700 CPU hours with the menu search model). Furthermore, as we will later discuss in Section 2.5, data-efficiency in training the surrogates is also otherwise a desirable factor as it helps combating model misspecification in cognitive models.

2.3 CASTING SIMULATOR-BASED MODELING AS META-LEARNING

Motivated by the conceptual similarity between Bayesian methods and meta-learning (previously discussed, e.g., by Grant et al. (2018) and Garnelo et al. (2018b)), our approach towards bypassing these problems is to view inference and simulation analogous to task-specific adaptation during meta-testing and prediction. The key insight is that both of these operations can be meta-learned jointly with an appropriate policy approximation task, without ever needing to approximate the true posterior $p(\theta \mid (\mathbf{s}, \mathbf{a}))$ for generating the training data, if one is satisfied with using a latent representation $z \in \mathcal{Z}$ to capture user-specific information. We generalize the operations of inference and simulation into mappings h_ψ and g_ϕ, with optimizable parameters $\{\psi, \phi\}$:

Definition 2.1 (Amortization for cognitive models)**.** Let $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces of the environment and $\mathcal{O} = \bigcup_n (\mathcal{S} \times \mathcal{A})^n$ be a collection of n observations over behavior generated by an individual user θ_U in a task θ_T. Amortization for cognitive models here means learning the following functions such that they are evaluable during online interaction:

1. Inference of user and task representations, done by the mapping $h_\psi : \mathcal{O} \to P(\mathcal{Z})$, where $P(\mathcal{Z})$ denotes a probability distribution over a joint user and task representation space $\mathcal{Z}$, which aims to capture the properties governing user behavior.
2. User behavior simulation, done by the mapping $g_\phi : \mathcal{S} \times \mathcal{Z} \to P(\mathcal{A})$, where $P(\mathcal{A})$ is a probability distribution over user action space.

To formalize the learning task, let $s \in \mathcal{S}$, $a \in \mathcal{A}$ and $(\mathbf{s}, \mathbf{a}) = \{(s_1, a_1), \ldots, (s_n, a_n)\} \in \mathcal{O}$. Learning the mappings h_ψ and g_ϕ, i.e., the posterior $p_\psi(z \mid (\mathbf{s}, \mathbf{a}))$ and the likelihood $p_\phi(a \mid s, z)$, is performed by minimizing the following meta-learning objective for policy approximation, while generalizing over the ground-truth user $\theta \sim p(\theta)$ and task population $\pi \sim p(\pi \mid \theta)$:

$$\min_{\phi,\psi} \mathbb{E}_{\theta \sim p(\theta), s \in \mathcal{S}, z \sim p_\psi(z \mid (\mathbf{s}, \mathbf{a}))} \Big[\delta\big[\pi(a \mid s), p_\phi(a \mid s, z)\big] \Big], \tag{2}$$

where δ is a dissimilarity function and the observations $(\mathbf{s}, \mathbf{a})$ are assumed to have been generated by executing π in the underlying environment. In Section 3, we discuss approximating a solution to this problem with neural processes.

Consistently with numerous current meta-learning approaches (e.g., Finn et al. (2017), Garnelo et al. (2018b)), we propose a modeling workflow with separate offline (meta-training) and online (meta-testing) phases described below. We additionally expand on the role of existing prior user models and adaptation w.r.t. user and task populations.

2.4 META-TRAINING AND META-TESTING

Algorithm 1 specifies the proposed meta-training procedure in a generic form, which still needs to be complemented with an appropriate meta-learning method for optimizing $\{\psi, \phi\}$, for the chosen implementations for h and g. However, in contrast to many popular meta-learning approaches, h must be here implemented as a probabilistic mapping to fully support interactive systems to balance between exploration and exploitation, as specified in Definition 2.1. This rules out many popular meta-learning approaches, such as model-agnostic meta-learning (MAML) (Finn et al., 2017) and Reptile (Nichol and Schulman, 2018). In section 3, we exemplify the procedure by implementing h, g and the optimization with neural process models.

Algorithm 1 Meta-training surrogates for cognitive models

Require: A distribution over users: $p(\theta_U)$
Require: A distribution over tasks: $p(\theta_T)$
Require: A cognitive model: $p(\pi \mid \theta)$
Initialize h and g with $\{\psi, \phi\}$
repeat
  Sample $\theta = \{\theta_U, \theta_T\}$, $\theta_U \sim p(\theta_U)$, $\theta_T \sim p(\theta_T)$
  Initialize environment $\mathcal{M}_{\theta_T}$
  Generate $\pi \sim p(\pi \mid \theta)$
  Generate n trajectories $(\mathbf{s}, \mathbf{a})$ by executing π on $\mathcal{M}_{\theta_T}$
  Optimize $\{\psi, \phi\}$ with respect to $(\mathbf{s}, \mathbf{a})$
until done

The meta-testing phase is straightforward: mappings h and g can be used to infer user representations $z \sim p_\psi(z \mid (\mathbf{s}, \mathbf{a}))$ w.r.t. observed $(\mathbf{s}, \mathbf{a})$, and for simulating user behaviors $a \sim p_\phi(a \mid s, z)$ on states of interest $s \in \mathcal{S}$. Again, the implementations for h and g should be chosen such that

they are evaluable during online interaction — for instance, otherwise seemingly valid approaches such as Probabilistic MAML (Finn et al., 2018) are not applicable due to their reliance on gradient computation during test time.

2.5 MODEL MISSPECIFICATION IN COGNITIVE MODELS.

Model misspecification is a relevant issue in behavioral user modeling. While typical LFI-approaches are highly sensitive to misspecification, this can be mitigated with our approach by combining observed user data with simulated data and meta-training the surrogates again, when new observations become available. We demonstrate in Section 4.2 that this approach enables elegant balancing between modeling accuracy and data requirements — especially in practical interactive user modeling applications which only have limited collections of user behavior datasets available.

3 USER MODELING WITH NEURAL PROCESSES

We use neural processes (NP) (Garnelo et al., 2018b) as an example solution for implementing and learning the mappings h and g of Definition 2.1. First, we briefly cover the relevant background on NPs and then explain in detail how they can be adapted for user modeling.

3.1 BACKGROUND ON NEURAL PROCESSES

Neural processes (Garnelo et al., 2018b) are a family of neural latent-variable models combining properties of neural networks and Gaussian processes (GP). Specifically, they are differentiable solutions for representing uncertainty over functions that may be utilized for few-shot approximation. For our purposes, NPs are particularly fitting as they match Definition 2.1 and that the meta-learning objective (Eqn. 2) can be readily computed for them.

NPs model a set of functions $\{f_d\}_d$ where each $f_d : X \to Y$ is assumed to be drawn from an underlying stochastic process $f_d \sim F$. NP approximates the underlying process F with a neural network g. As each function f_d drawn from the process F represents an individual instantiation of the process, a latent variable z is introduced for capturing the instance-dependent variation in F as $f_d(x) = g(x, z)$. NPs consist of an encoder, an aggregator and a conditional decoder. The encoder is a neural network for constructing representations $r_i = h_\phi((x, y)_i)$ at given observations $(x, y)_i$. The aggregator, α, constructs permutation-invariant summaries of the encoded representations as $r = \alpha(\{r_i\}) = \frac{1}{n}\sum_{i=1}^{n} r_i$. The summaries are further utilized to parametrize a (multivariate Gaussian) latent distribution $z \sim \mathcal{N}(\mu(r), I\sigma(r))$. The conditional decoder, $g_\psi(x_T, z)$, is a neural network that is conditioned on samples from the latent distribution to estimate $f_d(x_T) = y$ at locations x_T.

NP meta-training procedure samples individual instantiations $f_d \sim F$ of the stochastic process F. Here, each function f_d is evaluated at a varying number of inputs to produce a dataset of tuples $(x, y)_i^d$. Each dataset is then divided into separate *context* $(x_{1:m}, y_{1:m})$ and *target* $(x_{m+1:n}, y_{m+1:n})$ sets. Intuitively, here the context set represents the fully observed function evaluations while the target $x_{m+1:n}$ represents the locations at which the model aims to approximate $y_{m+1:n}$. The context and target sets are input to the encoder and the conditional decoder respectively, and the model parameters $\{\phi, \psi\}$ are optimized with respect to the NP-variant of Evidence lower-bound (ELBO). For further information about NPs and their training, see Garnelo et al. (2018b). Finally, note that the low complexity of NPs ($\mathcal{O}(n + m)$) makes them suitable for real-time scenarios.

3.2 ADAPTING NEURAL PROCESSES FOR USER MODELING

Neural processes can be adapted as concrete implementations for the required mappings h and g and for approximating a solution for Equation 2 within the proposed meta-training procedure (Algorithm 1). First, we recognize that Equation 2 is essentially a function approximation problem to which NPs can be applied — the true behavior-generative process $p(\pi \mid \theta)$ can essentially be treated as a stochastic process $\mathcal{P}$ where each instantiation $\pi \sim \mathcal{P}$ represents a policy. The NP latent variable z is utilized for capturing user/task representations and the mappings h and g can be implemented with the NP encoder $p_\phi(\pi \mid z)$ and conditional decoder $p_\psi(z \mid (\mathbf{s}, \mathbf{a}))$. The meta-training procedure is adapted as follows: the sampled behaviors $(\mathbf{s}, \mathbf{a})$ are split into context and target sets and the parameters $\{\psi, \phi\}$ can be optimized according to NP-ELBO.

In addition to the vanilla NPs, we consider also attentive neural processes (ANP) Kim et al. (2019), conditional neural processes (CNP) (Garnelo et al., 2018a) and attentive conditional neural processes (ACNP). ANPs are essentially NPs with the difference of including attention in the encoder architecture. The attention acts as a local latent variable, allowing ANPs to capture both global and local information affecting user behaviors. CNPs (and ACNPs) implement the latent encoding h as a deterministic mapping, thus lacking the required ability of sampling on $\mathcal{Z}$.

4 EXPERIMENTS

We conduct three experiments where we compare our approach against other ways one could conceivably try to solve the problem — although this has not been previously done.

The first is a demonstration in a benchmark gridworld environment. The second is a menu search task where a cognitive user model, justified and validated by earlier cognitive science studies, allows us to study real-user performance with simulations. The third experiment is a reasonably realistic menu search assistant scenario.

Comparison methods and baselines. We assess the modeling capabilities of the proposed solution in terms of its ability to predict the actions of individual agents, as a function of the number of observed context trajectories as measured by Eqn. 2. This metric directly evaluates the simulation quality but also indirectly the quality of the user representations $z \in \mathcal{Z}$, because the simulations are dependent on them. Unless otherwise specified, the experiments aim to simulate realistic user modeling applications by limiting the training data to observations from ~ 1000 simulated users.

Even though it would be an important baseline, we were not able to produce any representative results with LFI due to its immense computational requirements in these tasks. For reference, Kangasrääsiö et al. (2019) compared several LFI-methods, including BO, on exactly the same Menu Search model used in our experiments. They gave 700h of CPU-time for each method to run only one individual inference task and noted that it is likely that none of the methods converged. Obtaining conclusive accuracy results with LFI in our experiment setting is practically intractable as at least hundreds of individual inference results would be needed. This highlights the necessity of our solution — we reduce the inference time to less than 0.01 seconds in the same setting, corresponding to a speed-up of more than eight orders of magnitude.

Instead, we compare our approach against two baselines and two alternative surrogate architectures. The alternative surrogates are a transformer trained with MAML and a standard transformer. These solutions aim to represent the closest available methods which one might consider applying to our task, even though they do not fully satisfy the Definition 2.1, thus having limited practical applicability in comparison to NPs. We also include comparisons between several alternative NP architectures. Implementation details are included in the Supplement.

The two baselines are a Bayesian Optimization (BO) model and a context-free predictor. Furthermore, we provide results from an oracle, acting as an upper bound for the performance of any solution, including LFI. Conversely, we consider the BO-baseline as an approximate lower-bound for LFI performance. Both the BO baseline and the oracle utilize the ground-truth user model $p(\pi \mid \theta)$ for prediction — the oracle parametrizes the user model with the true user parameters while BO utilizes MAP estimates. Although converging faster than LFI, BO is still computationally very heavy, due to the expensive simulation costs, and is feasible only in our first experiment.

4.1 EXPERIMENT 1: GRIDWORLD ENVIRONMENT

Setting. The first experiment scenario is based on a simple 10×10 gridworld environment. In this setting, we consider modeling Monte Carlo Tree Search (MCTS) (Browne et al., 2012) agents with unknown reward functions and MCTS parameters. This benchmark scenario evaluates the modeling system's ability to approximate the uncertainty over user policies. In this experiment, we assume that a generative user model and a parameter prior are available, capturing the true generative process of the population.

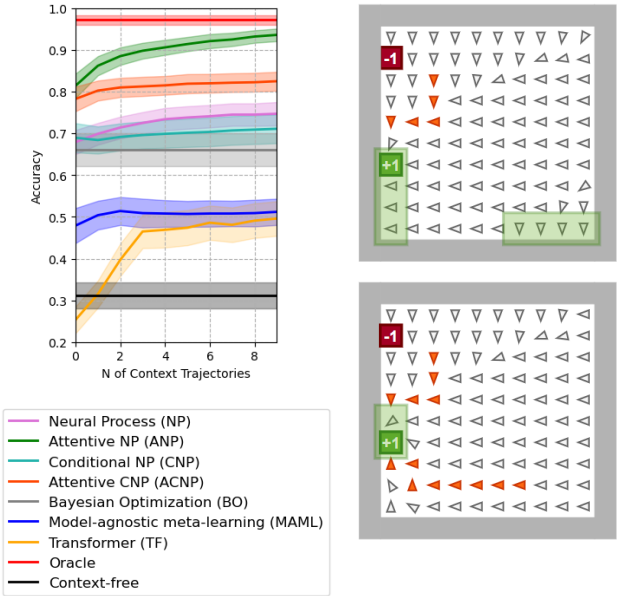

Figure 1: Gridworld results. **Left:** Modeling accuracy as a function of the number of observed context trajectories. The best NP-based model (here ANP) achieves comparable results to the upper bound given by an oracle; all NP-based models are clearly better than alternatives. The figure illustrates the gradual improvement of the predictions of NPs as more context trajectories are perceived. The BO results are averaged over the number of context trajectories due to the small sample size. **Right:** Illustration of ANP uncertainty update on policy predictions. The predictions (gray arrows) align towards the implicitly inferred possible goal states (green rectangles). In the upper figure, the predictions are conditioned on one observed trajectory (orange arrows). In the lower figure, we observe that the system implicitly infers the location of the positive reward, within the accuracy of two squares, after perceiving the second trajectory. The dark green and red squares correspond to the true positive and negative reward states.

The gridworld environment is defined as a partially observable Markov decision process (POMDP) with deterministic transition dynamics. The action space consists of four possible actions that correspond to the agent relocating from

Table 1: Modeling accuracy as a function of the number of observed context trajectories in the menu-search setting of Section 4.2.

N of context trajectories	ANP	MAML	Transformer	Oracle	Context-Free
0	$\mathbf{0.937 \pm 0.031}$	0.774 ± 0.033	0.920 ± 0.035	$\mathbf{0.970 \pm 0.002}$	0.921 ± 0.012
1	$\mathbf{0.953 \pm 0.008}$	0.916 ± 0.026	0.922 ± 0.021	...	...
2	$\mathbf{0.954 \pm 0.007}$	0.928 ± 0.025	0.931 ± 0.017	...	...
5	$\mathbf{0.955 \pm 0.006}$	0.943 ± 0.021	0.926 ± 0.012	...	...
9	$\mathbf{0.955 \pm 0.006}$	0.952 ± 0.014	0.928 ± 0.009	...	...

its current state to adjacent states. Each gridworld scenario always contains two reward states - one with a positive reward and one with a negative reward. The agents gain no rewards or penalties other than from the given states. The full setting details are given in the Supplement.

Modeling task. The modeling task consists of predicting the subsequent actions of agents sampled from the population. Each scenario assigns the modeling system with a varying number of context trajectories and a truncated target trajectory generated by the agent. The task is to predict the remaining actions of the target trajectory. All information regarding the agent, excluding the trajectories, is hidden during both training and evaluation (except for the oracle).

Results. The NP-family models are able to outperform all the baselines (Fig. 1) with the ANP converging close to the performance of the oracle. It is likely that the BO-baseline has not properly converged, although given clearly the largest amount of computation time, and it may not act as a reliable approximation for LFI performance. Finally, the transformer and MAML are unable to generalize to the task, likely due to the too limited amount of training data.

Comparisons between the NP-family models suggest that, in terms of NP architecture, the most impactful factor contributing to the modeling performance is attention, i.e., local latent variables, as ANP and ACNP outperform their non-attentive counterparts. Consistent with the probabilistic treatment of h (Def. 2.1), stochasticity of the global latent variables z (ANP and NP) also seems to improve the results. Because ANP was clearly the best of the NP methods, and hence remaining NP-models would not affect conclusions, we omit the NP, CNP and ACNP models for the following experiments, to save computation.

4.2 EXPERIMENT 2: MENU SEARCH ENVIRONMENT

Setting. Our second experiment is based on the Menu Search model of Kangasrääsiö et al. (2019), a modified version of Chen et al. (2015). The Menu Search model is a cognitive model describing human search behavior in terms of eye movements (saccades) when searching for a target item in a computer dropdown menu. Motivated by *computational rationality* (Gershman et al., 2015), the model simulates user behavior as a result of optimizing the search behavior with RL given their cognitive constraints of the user. The details are given in the Supplement.

Modeling task. In this experiment, we apply our method for modeling agents whose search behavior is specified by the Menu Search model. As in the first experiment, we train the model parameters on data simulated with the given cognitive model. For each simulation, we sample a new menu, together with its element-wise information about the target word, as specified by Kangasrääsiö et al. (2019).

Modeling accuracy. Table 1 summarizes the obtained modeling accuracies. We notice that after one observed trajectory, the ANP-based model achieves results comparable to the oracle upper bound. Unlike in the previous experiment, most of the users seemed to converge to a relatively narrow and finite set of search strategies, simplifying the difficulty of the modeling problem. As a result, MAML and the transformer achieve clearly higher relative accuracy than in the previous experiment, despite the limited training data.

Model misspecification. We study the effects of model misspecification in cognitive models by repeating the modeling task with a misspecified model. This model represents an otherwise accurate Menu Search model, but it models roughly 35% of the saccades into incorrect locations (full implementation details in the Supplement). We repeat the meta-training with different percentages of data obtained from the true user population. We explore both our solution's robustness against the misspecified model and its ability to adapt to the true generative process.

The results are illustrated in Figure 2. First, we observe that our approach can remain robust against user model misspecification: the most significant absolute and percentual difference to the oracle is observed at the highest rates of misspecification. Secondly, it can be seen how our solution adapts to the ground-truth generative process when the proportion of the ground-truth data increases. We repeated the scenario by meta-training the ANP only on data from the true user population. We found that the misspecified model improved the results when the number of observed real users was under 200 (i.e., here percentage $< 20\%$), after which

Table 2: User search times and modeling/simulation times per assistant action with different assistant systems in Section 4.3.

Assistant	Search time (s)	Time saved (%)	Modeling time per action (ms)
No assistance	4.774 ± 0.235	–	–
MAML	3.705 ± 0.409	22.4	1250.264 ± 62.387
Transformer	2.918 ± 0.191	38.8	$\mathbf{1.140 \pm 0.442}$
ANP	$\mathbf{2.590 \pm 0.226}$	**45.7**	8.460 ± 6.495
Full knowledge	$\mathbf{2.577 \pm 0.162}$	**46.0**	–

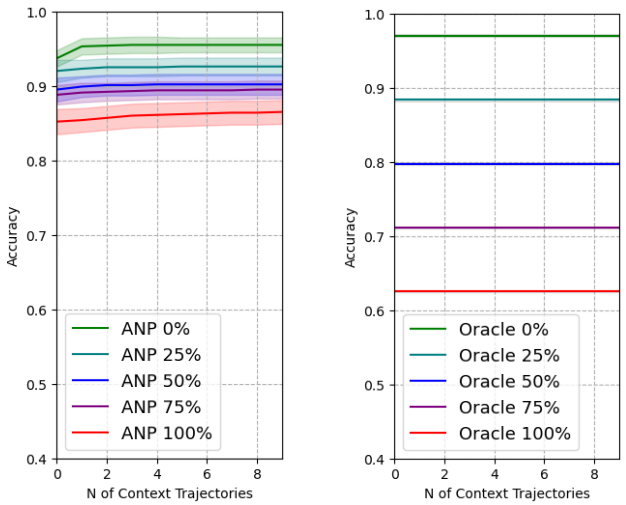

Figure 2: **Left:** Modeling accuracies of the ANP-based system when trained with data partially from a misspecified user model and partially from the true population. Here, the percentages denote the share of the training data generated with the misspecified user model. **Right:** Modeling accuracies of the oracle at different misspecification rates of the user model. Key: see Figure 1

it had a hindering effect on the predictions. However, we expect that the utility of misspecified models can be significantly higher in more complex modeling problems where more data is required to generalize to the problem.

4.3 EXPERIMENT 3: MENU SEARCH ASSISTANT

Setting. In our third experiment, we aim to demonstrate the practical utility of the proposed approach for interactive systems by extending the Menu Search environment into a reasonably realistic AI-assistant scenario. First, we scale the environment to consider two levels of hierarchy: the menu consists of a main menu whose elements act as links to sub-menus; we use the menus of the previous experiment.

Secondly, we introduce an AI assistant equipped with the proposed user modeling system. The task of the assistant is to utilize the modeling system to infer the target elements of the users based on observed search behaviors in the current menu, and to propose sub-menus for the users. Intuitively, a successful assistant should guide the users to menus that are likely to contain the true target for them, to shorten their search time. We implement the assistant as a simple rule-based agent that conditions its actions on the simulated user behaviors $a \sim p_\phi(a \mid s, z)$, $z \sim p_\psi(z \mid (\mathbf{s}, \mathbf{a}))$. Further details on the experiment setting are in the Supplement.

Results. Table 2 compares the performance of the resulting assistant against a non-assisted user and a MAML-based and a transformer-based solutions. The MAML-based solution requires gradient-computation during test-time leading to modeling times greatly higher than the response time between human actions ($\approx$ 300ms) in this experiment. This prevents online user model updates, hence hindering the effectiveness of the assistance. We also include results with an assistant that has full knowledge of the users' target elements to provide an upper-bound. We notice that the ANP-guided assistant can significantly reduce the user's search time and almost reaches the upper-bound performance of the assistant that has perfect knowledge. The observed results are encouraging regarding the ability of our solution to guide the behaviors of real-time interactive systems.

5 RELATED WORK

Inference for decision making agents has been previously studied with probabilistic programming. However, most of the approaches make restricting assumptions either regarding the behavior generative processes of the users or the inference objectives and could be applied only for very limited types of problems. For instance, Zhi-Xuan et al. (2020) consider online inference of boundedly-rational agents but their approach can be applied only in discrete and deterministic environments to capture only agent goals. Furthermore, their solution assumes that the agents start planning their policy from scratch during interaction — in practical interactive settings, humans might already have a partial or complete plans at the beginning of interaction. On the other hand, many other works, such as by Seaman et al. (2018), assume that the likelihood for the generative process $p(\pi \mid \theta)$ can be evaluated for MCMC, which is often an unrealistic assumption with advanced cognitive models.

The connection between probabilistic modeling and meta-learning tasks has been previously discussed and several approaches have been proposed based on this idea, includ-

ing by Grant et al. (2018) and Gordon et al. (2018). However, to our knowledge, our work is the first to propose a reformulation for connecting meta-learning with simulator-based modeling important in small-data domains. Unlike previous works, we incorporate information from existing cognitive models to offset the small amount of available data and address the problem of model misspecification.

Using transfer and meta-learning in RL problems has been previously widely studied. For instance, Yao et al. (2018) used HiP-MDPs (Doshi-Velez and Konidaris, 2016) for modeling differences in environment dynamics and to further parametrize a policy. Similarly, Galashov et al. (2019) propose a probabilistic framework for sequential decision-making that they instantiate with NPs for meta-learning. In contrast to this line of works, the novelty of our work is not about a generalizable solution to distributions of RL tasks, but rather about a generalizable method for making modeling with cognitive models practical. This is an important distinction because cognitive models are not necessarily compatible with the RL formalism — even when they are, they are based on computational rationality, and specifically tailored to account for cognitive limitations. Adapting these limitations to existing frameworks, such as HiP-MDPs, is not trivial and necessarily requires manual effort.

Inverse reinforcement learning (IRL) (Ng et al., 2000) aims to recover agents' reward functions based on observed behaviors. Although it has been previously utilized also in user-centric problems (Chandramohan et al., 2011), our perspective is more general as we consider inference over arbitrary user parameters (instead of only rewards) and over varying policy-generative algorithms/processes. This allows our approach to be utilized for inference also with cognitive models, where user behaviors are not necessarily optimal and are governed by human biases.

Many computational approaches motivated by cognitive science share parallels with our objectives. For instance, computational rationality (Lewis et al., 2014; Gershman et al., 2015) and Theory of Mind (ToM) (Premack and Woodruff, 1978) have motivated numerous computational approaches such as Bayesian ToM (Baker et al., 2011), Machine Theory of Mind (Rabinowitz et al., 2018) and the Menu Search model (Chen et al., 2015) for modeling human behaviors. Furthermore, Peltola et al. (2019) utilize ToM for modeling users with their own models of the interactive system in bandit settings. Among others, these models are prime candidates our method can be applied to.

6 DISCUSSION

In this work, we have addressed the so-far unaddressed problem of enabling probabilistic user modeling with complex cognitive models in real-time applications. We introduced a meta-learning approach for training widely applicable differentiable surrogates for cognitive models. We studied neural process models as example implementations for the surrogates and demonstrated comparable modeling performance to likelihood-free inference with computational cost suitable for online applications. We also showed that the proposed solution allows AI-assistants to utilize cognitive user models computationally feasibly, for instance in a previously studied menu-search task. In a larger scale, the solution not only removes a computational bottleneck towards incorporating users into probabilistic programming models, but also enables real-time user modeling in various applications where they currently are not possible within usually available computational budgets.

We also demonstrated how the effects of model misspecification in cognitive models can be mitigated in the surrogates by incorporating observed user data in the training. Importantly, we observed that our approach provided robustness against misspecified cognitive models while adapting to the true population as more behavior data became available. Based on these observations, we conclude that the proposed solution can be particularly useful in application domains where user data are limited or behavioral user models can be slightly misspecified.

It is crucial to note that the problem of probabilistic user modeling has very specific requirements (discussed in Section 2) and that apart from LFI, which is computationally intractable for our problems, we are not aware of any solutions which could act as either relevant baselines or alternatives to the proposed neural process surrogates. Specifically, all the experimented alternative surrogate implementations, such as MAML and transformers, were affected by a clear competitive disadvantage against neural processes mainly due to their non-probabilistic structure and higher data requirements. We further note that also neural processes feature several compromises in comparison to the original cognitive models: they cannot be directly adapted for Bayesian inference in an explicit, predefined parameter space and they do not necessarily follow all constraints coming from the known structure of the behavioral model due to amortization. Future research should adapt and develop alternative surrogate solutions to address these drawbacks.

Interesting avenues for future research include utilizing the surrogates in full probabilistic programming pipelines, although we hypothesize that this should already be possible with our approach. Other attractive extensions could consider alternative surrogate architectures to handle, for instance, non-stationarity in cognitive models and settings with multiple data modalities. Regarding ethical considerations, user modeling has always been a double-edged tool and can potentially be abused to serve other interests than those of users — this should be taken carefully into account in all of its applications. As a generic tool to mitigate some of these issues, we recommend combining user systems with privacy preservation with differential privacy.

References

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

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefevre, and Olivier Pietquin. User simulation in dialogue systems using inverse reinforcement learning. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

Xiuli Chen, Gilles Bailly, Duncan P Brumby, Antti Oulasvirta, and Andrew Howes. 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*, pages 4217–4226, 2015.

Pedram Daee, Tomi Peltola, Aki Vehtari, and Samuel Kaski. User modelling for avoiding overfitting in interactive knowledge elicitation for prediction. In *23rd International Conference on Intelligent User Interfaces*, pages 305–310, 2018.

Allan Dafoe, Yoram Bachrach, Gillian Hadfield, Eric Horvitz, Kate Larson, and Thore Graepel. Cooperative ai: machines must learn to find common ground, 2021.

Finale Doshi-Velez and George Konidaris. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *IJCAI: proceedings of the conference*, volume 2016, page 1432. NIH Public Access, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018.

Alexandre Galashov, Jonathan Schwarz, Hyunjik Kim, Marta Garnelo, David Saxton, Pushmeet Kohli, SM Eslami, and Yee Whye Teh. Meta-learning surrogate models for sequential decision making. *arXiv preprint arXiv:1903.11907*, 2019.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

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

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Meta-learning probabilistic inference for prediction. *arXiv preprint arXiv:1805.09921*, 2018.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

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

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

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

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. *arXiv preprint arXiv:1901.05761*, 2019.

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

Sheng Li and Handong Zhao. A survey on representation learning for user modeling. In *IJCAI*, pages 4997–5003, 2020.

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

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

Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2 (3):4, 2018.

Tomi Peltola, Mustafa Mert Çelikok, Pedram Daee, and Samuel Kaski. Interactive ai with a theory of mind. In *Computational Modeling in Human-Computer Interaction*, 2019.

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

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

Iris Rubi Seaman, Jan-Willem van de Meent, and David Wingate. Nested reasoning about autonomous agents using probabilistic programs. *arXiv preprint arXiv:1812.01569*, 2018.

Scott A Sisson, Yanan Fan, and Mark Beaumont. *Handbook of approximate Bayesian computation*. CRC Press, 2018.

Mikael Sunnåker, Alberto Giovanni Busetto, Elina Numminen, Jukka Corander, Matthieu Foll, and Christophe Dessimoz. Approximate bayesian computation. *PLoS computational biology*, 9(1):e1002803, 2013.

Brandon M Turner and Trisha Van Zandt. Hierarchical approximate bayesian computation. *Psychometrika*, 79 (2):185–209, 2014.

Jiayu Yao, Taylor Killian, George Konidaris, and Finale Doshi-Velez. Direct policy transfer via hidden parameter markov decision processes. In *LLARLA Workshop, FAIM*, volume 2018, 2018.

Zeping Yu, Jianxun Lian, Ahmad Mahmoody, Gongshen Liu, and Xing Xie. Adaptive user modeling with long and short-term preferences for personalized recommendation. In *IJCAI*, pages 4213–4219, 2019.

Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. Parameter-efficient transfer from sequential behaviors for user modeling and recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1469–1478, 2020.

Tan Zhi-Xuan, Jordyn Mann, Tom Silver, Josh Tenenbaum, and Vikash Mansinghka. Online bayesian goal inference for boundedly rational planning agents. *Advances in Neural Information Processing Systems*, 33:19238–19250, 2020.