Differentiable User Models (Supplementary Material)

Alex Hämäläinen[1] **Mustafa Mert Çelikok**[1] **Samuel Kaski**[1,2]

[1]Department of Computer Science, Aalto University
[2]Department of Computer Science, University of Manchester

A EXPERIMENT 1 DETAILS

Setting. The first experiment scenario considers modeling Monte Carlo Tree Search (MCTS) [Browne et al., 2012] agents in a simple 10×10 gridworld environment. The gridworld environment is defined as a partially observable Markov decision process (POMDP) with deterministic transition dynamics. The state and action spaces are shared across the all agents, including the transition function; the reward and observation functions are individual for each agent.

The state space $\mathcal{S} = \{1, \ldots, 10\}^2$ captures all possible agent locations, i.e., grid states. The action space is defined as $\mathcal{A} = \{\text{up}, \text{down}, \text{left}, \text{right}\}$, the transition function corresponds to transitioning to adjacent grid states in relation to the agent's current location according to the selected action. Actions attempting to relocate agents out of the grid do not cause state transitions.

The agents are fully described by their generative process $\pi \sim p(\pi \mid \theta)$ (MCTS) and parameters $\theta \sim p(\theta)$. Their respective distributions are described in Table 1. In addition to the reward function, determining one positive and one negative reward state, the prior determines the agent memory, observation function, and MCTS planning-tree depth. The agent memory is defined as a binary parameter determining if the agents utilize the planning tree from previous time steps for subsequent planning instead of starting from scratch. The agents are divided into two subspecies depending on their observation function. Specifically, the complete gridworld is fully observable for half of the agent population, while the other half cannot observe reward states that are located beyond planning-tree depth. In practice, such agents avoid exploring the environment if the positive reward is not directly observed.

Table 1: Uniform prior on user model parameters for the user population in gridworld experiment.

User Parameter	**Distribution**
Reward States (x, y)	$\mathcal{U}\{1, \ldots, 10\}$
Memory	$\mathcal{U}\{0, 1\}$
Observation function	$\mathcal{U}\{0, 1\}$
Tree Depth	$\mathcal{U}\{5, \ldots, 10\}$

Here the task is to model individual users sampled from the population $\theta \sim p(\theta)$. The users generate $n \sim \mathcal{U}\{1, \ldots, 8\}$ trajectories of length 10, each collected from one episode in the environment. A new initial user state, $x, y \sim \mathcal{U}\{1, \ldots, 10\}$, is sampled at each episode. The trajectories are then partitioned into context and target data for NP training by randomly selecting and then truncating one target trajectory at length $l \sim \mathcal{U}\{1, \ldots 9\}$. The beginning of the truncated trajectory, together with the other trajectories, forms the context dataset while the remaining half is held-out as modeling target. Based on the given context data, we evaluate prediction accuracy on the held-out target datasests over tasks. Unlike in the rest of our experiments, each method is here trained with data from 10000 users.

Implementation. All NP models are implemented on the basis of the `NeuralProcesses.jl` library, a Julia variant of the neural process library of Dubois et al., (2020). The MCTS agents are implemented by utilizing the `POMDPs.jl` (Egorov et al., 2017) library. All NP models are meta-trained on a single GPU (NVIDIA Quadro P2200). The code used to produce

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

all the results in this paper can be found at `https://github.com/hamalajaa/DifferentiableUserModels`.

The base-architecture is shared among all NP models (summarized in Table 2). We implement the MAML as first-order MAML [Finn et al., 2017] to reduce test-time computation. The Reptile, MAML and transformer details are included in Tables 3 and 4 (Remark: here, the 'fifth' action corresponds to 'staying still' when the positive reward is found.)

Table 2: Base-architecture shared by all NP models in experiment 1.

Encoder		**Decoder**	
Number of layers	6	Number of layers	6
Activations	Leaky ReLU	Activations	Leaky ReLU
Hidden dimensions	128	Hidden dimensions	128
Latent distribution	Gaussian	Output distribution	Categorical
Input dimensions	2	Output dimensions	5

B EXPERIMENT 2 DETAILS

Setting. The second experiment is based on the Menu Search model of Kangasrääsiö et al. [2019]. 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. The model simulates user behaviors as a result of optimization with RL given their cognitive constraints. In this experiment, we implement the users as deep Q-learning agents to reduce data generation costs.

Formally, the environment is specified as a POMDP where states contain information about (1) the user's current knowledge about the menu elements, (2) the current gaze focus, and (3) whether or not the user has closed the menu (i.e., quit). We consider a menu of eight elements, where each element is described with its semantic relevance and length given the user's target element. At each step, the user can either fixate on a menu element or quit the scenario. Fixating on a menu element has a chance to reveal the given element while also having a chance to reveal the adjacent elements via peripheral vision. If user action results in revealing the target element, the element is automatically selected, and a significant positive reward is given.

The target word is not present in 10% of the generated menus. If the user recognizes that the target element is not present and quits the menu, a large reward is emitted. For each action, the user is otherwise given a small negative reward based on the duration of the action specified by cognitive parameters presented in Table 5 as priors. When entering a menu for the first time, there is a chance, p_{rec}, that the user recalls the menu, completely revealing the entire menu layout.

Similarly as in the first experiment, each user completes $n \sim \mathcal{U}\{1, \ldots, 8\}$ search tasks with independently sampled menus and target elements constructing n trajectories. Similarly, as in the first experiment, we truncate one of the trajectories to form global and local contexts and the prediction target for ANP training. The ANP, Reptile, MAML, and transformer architectures follow the design used in the first experiment (except for the input and output dimensions).

Implementation. The implementation details for ANP, Reptile, MAML, and transformer architectures follow the design used in the first experiment (except for the input and output dimensions).

Table 3: The architecture shared by Reptile, MAML, and transformer in experiment 1.

Architecture	
Layers	Transformer + 6 MLP
Activations	ReLU (+ softmax)
Hidden dimensions	128
Input dimensions	2
Output dimensions	5

Table 4: Reptile, MAML, and transformer training and evaluation details in experiment 1.

Training	**Reptile**	**MAML**	**Transformer**
Optimizer and learning rate	-	-	Adam, $5 \cdot 10^{-4}$
Meta optimizer	Adam, $5 \cdot 10^{-3}$	Gradient descent, $5 \cdot 10^{-3}$	-
Batch optimizer	Gradient descent, $5 \cdot 10^{-3}$	Gradient descent, $5 \cdot 10^{-3}$	-
Loss	Cross entropy	Cross entropy	Cross entropy
Evaluation			
Optimizer and learning rate	Gradient descent, 0.01	Gradient descent, 0.01	-
N of gradient steps	32	32	-

Table 5: Distributions for user cognitive properties used in the second experiment.

User Parameter		**Distribution**
Menu recall probability	p_{rec}	$\mathit{Beta}(3.0, 1.35)$
Eye fixation duration	f_{dur}	$\mathcal{N}(3.0, 1.0)$
Target item selection delay	d_{sel}	$\mathcal{N}(0.3, 0.3)$

C EXPERIMENT 3 DETAILS

The third experiment extends the menu search environment into a relatively realistic AI-assistant scenario. First, the menu environments are scaled to consider two levels of hierarchy: each full menu consists of a main menu whose elements correspond to labels that (1) act as links to sub-menus and (2) summarize the contents of these menus. 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 propose sub-menus for the users. Intuitively, a successful assistant should guide the users to menus that are likely to contain the true target to shorten their search time.

Environment. The hierarchical menu search environment introduces an 8×8 two-level menu setting. Importantly, the environment behaves otherwise similarly to the original non-hierarchical version, with the exception of introducing a main menu that allows a user to navigate between multiple menus. In addition, we introduce a simple mapping between user observations (semantic relevancies and lengths w.r.t. the target element) and assistant observations (logical groups). Specifically, each scenario introduces a set of 8 logical groups $\mathcal{S}_{AI} = \{1, \ldots, 8\}$ and 4 semantic relevance groups $\mathcal{S}_{user} = \{target(1), high(2), medium(3), low(4)\}$ and an independently generated bidirectional mapping between $\mathcal{S}_{AI}$ and $\mathcal{S}_{user}$. The mapping initializes an ordered set of relevancies as $r = \{4, 4, 4, 3, 3, 2, 3, 3\}$ and assigns a relevance for each logical group with a randomized circular shift on r. The intuition of the mapping is simply to mask the semantic information regarding the target element (via randomization) while allowing a soft prior heuristic for the assistant by conserving semantic similarity between similar logical groups. We similarly mask the item lengths via randomization.

After the mapping between the observation spaces $\mathcal{S}_{AI}$ and $\mathcal{S}_{user}$ is constructed, we sample two logical groups for each sub-menu (such that each group occurs exactly twice in the full menu) and determine a semantic label for the menus summarizing the relevancies of their respective logical groups. The target element is then assigned randomly into one of the sub-menus that includes a logical group with $high$ relevance. The contents for each sub-menu are otherwise determined by mapping the semantic labels of their logical groups into individual items according to the original menu search model specifications. The main menu similarly follows the original specifications — however, we utilize the semantic labels of the corresponding sub-menus as the relevancies for the main menu elements. At the main menu level, we also replace the item length information with a binary variable denoting if the user has already opened the corresponding sub-menu. Finally, the transition dynamics between the main menu and sub-menus are defined as follows: selecting an element at the main menu -level transitions the user to the corresponding sub-menu, while quitting a sub-menu transitions the environment state back to the main menu. Otherwise, all the transition and reward dynamics follow the original environment specifications.

Assistant. The hierarchical menu search setting involves a simple search assistant guided by a pre-trained ANP-based user model (user model implementation and training details are described below). In each scenario, the assistant is initially inactive and only activates if the user fails to find the target element from the first sub-menu it explores. When activated, the assistant may suggest and highlight an individual main menu element when the user is at the main menu level. A highlighted main menu element is assumed to attract the attention of the user at its next action and the user's gaze is guided towards

the highlighted element. Simultaneously, we assume that the user features some degree of trust towards the assistant's suggestion and the semantic relevance score of the highlighted element is increased by one level. In practice, this allows the user also to reject poor suggestions.

We implement the assistant as a simple rule-based agent that continuously updates the user model as new user actions are observed. We assume that the assistant can track users' gaze locations but that it does not have access to the semantic relevancies of the items. Instead, the assistant updates its estimate on the currently observed (and unobserved) menu elements in terms of the observation space $\mathcal{S}_{AI}$ specified above. When activated, the assistant simulates one user action at fully observed main menu -level conditioned on the observed user search behavior: $a \sim p_\phi(a \mid s, z)$, $z \sim p_\psi(z \mid (\mathbf{s}, \mathbf{a}))$. The main menu element corresponding to the estimated most likely user action is then selected as the assistant's suggestion.

Implementation and training details. The ANP-based user model is meta-trained on a single GPU. Each user generates 1 trajectory which is split at length $l \sim \mathcal{U}\{2, \ldots, 10\}$ into context and target trajectories for ANP training. The base architectures of the ANP, Reptile, MAML and transformer models are identical to the previous experiment. The online prediction times are run on a laptop CPU (Intel Core i7-7700HQ).

References

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.

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.

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.