A Decision-Theoretic Approach for Adaptive User Interfaces in Interactive Learning Systems

Harold Soh
University of Toronto
harold.soh@utoronto.ca

Scott Sanner
Oregon State University
scott.sanner@oregonstate.edu

Greg Jamieson
University of Toronto
jamieson@mie.utoronto.ca

Abstract

Designing user-interfaces for interactive machine learning systems remains a complex, time-consuming task. In this work, we adopt a decision-theoretic approach to the problem and develop a plausible general POMDP model for an adaptive interface, which functions as a communication layer between the human user and machine learning model. We discuss the practical issues that arise due to the complex state, observation and action spaces, and highlight key research areas that are necessary to enable the practical application of these models.

1 Introduction

Handcrafting a user-interface is a complex and time-consuming task complicated by variety of factors including uncertainty, noisy inputs, and evolving user and environmental states. Interface choices often involve difficult trade-offs and may entail negative side-effects [1]. Furthermore, behavioral changes in the software due to machine learning come at a risk of increased unpredictability—systems that are deemed volatile and unintuitive may be abandoned in favor of less effective but more user-friendly alternatives.

In this workshop paper, we adopt a decision-theoretic mindset and view the user interface (UI) as an adaptive agent managing the flow of information between the user and the machine learning model to achieve an overarching goal[1]. Through contextual self-modification, adaptive user interfaces (AUIs) can influence both the user and ML model to yield improved system outcomes.

Given the opaque nature of environmental and internal user states, we develop a general partially-observable Markov decision process (POMDP) model of adaptive user interfaces that captures the salient aspects of interactive machine learning (IML) systems. It then becomes apparent that challenges arise when attempting to apply such a model in practice. In particular, the large, complex and continuous observation and action spaces render state-of-the-art solvers ineffective. We highlight these (and other) issues as promising areas for future work in this domain.

2 The Problem of Adapting User Interfaces

In general, a well-designed interface enables a human user to accomplish a task in an efficient manner, that is, with a minimal exchange of information between the human and machine. To orga-

[1]We distinguish two different classes of IML systems based on desired outcomes. One class is designed primarily to train an ML model. The second class aims to achieve some system goal and a ML model/method is used in furtherance of this objective. We believe adaptive user interfaces are useful for both classes of systems.

nize our discussion, we adopt the perceive-select-act model of adaptive systems proposed by Feigh, Dorneich and Hayes [1]. In this view (Fig. 1), contextual changes *trigger* adaptations in system's behavior or interface (e.g., a change in modality or level-of-detail). The problem of adapting a user interface is one of developing appropriate *contextual triggers*, which we cast as finding an optimal mapping from contexts to adaptation types.

As previously mentioned, learning can cause system behavior changes that are confusing to users, leading to a fall in overall satisfaction and productivity. In addition, the use of *active learning* methods [2] may steepen learning rates, but at the cost of burdening the user with queries. How should the system trade-off learning (which tends to confer long-term benefits) against shorter-term goals? Moreover, recognizing that users are not infallible oracles [3, 2], should all user interactions be weighted equally as useful training data?

Given our discussion above, adapting the UI can be seen as a sequential decision making problem (which adaptations to perform) under uncertainty, which makes the POMDP an appealing framework. If we are able to construct an appropriate POMDP model for a user-interface, then an optimized policy will address the discussed trade-offs in an suitable manner.

3 A POMDP Formulation of Adaptive User Interfaces for IML

Formally, a POMDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \gamma \rangle$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of possible actions and $\mathcal{O}$ is the set of observations available to the agent. The (Markovian) transition function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ defines the probability of the next state given the action and previous state. The observation function $\mathcal{Z} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \rightarrow [0, 1]$ models the probability of an observation given the outcome of a state after executing an action. The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ gives the real-valued reward of taking an action in a state. Finally, the discount factor $0 \leq \gamma \leq 1$, discounts rewards t time steps into the future by γ^t.

Since observations only give partial information, the agent has to maintain a belief distribution over states. A finite history is denoted as $h_t = \{\mathbf{a}_0, \mathbf{o}_1, \ldots, \mathbf{a}_{t-1}, \mathbf{o}_t\} \in \mathcal{H}$ where $\mathbf{a}_t$ and $\mathbf{o}_t$ are the action and observation at time t respectively. A policy π maps elements of $\mathcal{H}$ to actions $\mathbf{a} \in \mathcal{A}$, or a distribution over actions in the case of stochastic policies. Given an initial belief distribution $\mathbf{b}$ over the starting states, the expected value of a policy π is $V_\pi(\mathbf{b}) = \mathbf{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid \mathbf{b}_0 = \mathbf{b} \right]$ where r_t is the reward received at time t. The optimal policy π^* maximizes this expectation.

3.1 Adaptive User Interface POMDP

At a high-level, our POMDP models the human user (U) and machine-learning method (M) as agents within the system. The user-interface (I) acts a *communication layer* between these two entities and receives (partial) observations of their actions and internal states. At each time-step, the interface agent decides how to manage the communication between these agents (and possibly the environment) to achieve an overall system goal. Our factored POMDP model is similar to William

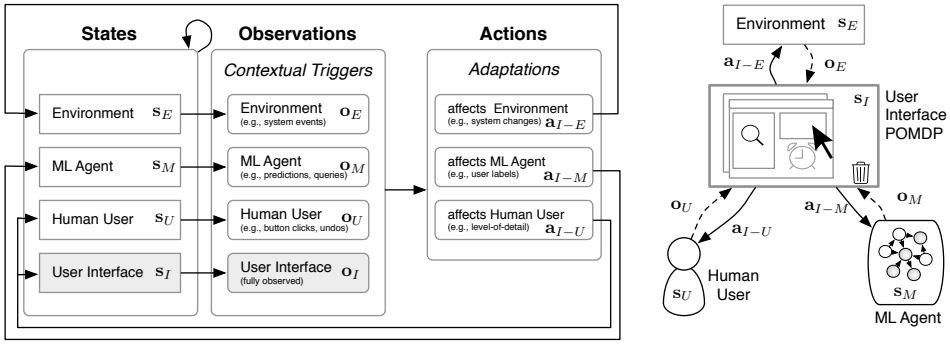

Figure 1: A high-level system overview of the adaptation framework [1] where we have mapped entities to elements of our factored adaptive user interface POMDP for interactive machine learning.

and Young's POMDP for spoken-dialog systems [4], but we address the additional issue of a machine learner embedded in the system.

To clarify our exposition, we use the running example of *alarm prioritization*—a scaled-down version of network alarm triage [5]—where incoming alarms are labelled as either being low, medium or high priority. Similar to CueT, the system attempts to learn interactively from human operators, but the interface elements may be altered and the ML agent may receive additional information regarding the user's state. For simplicity, we assume a single human and machine agent, but the model can be extended to multiple agents of either type.

States The system states $\mathcal{S}$ comprise the possible configurations of the environment, user, ML agent and UI states. As such, we factor the state into separate variables, $\mathbf{s} = (\mathbf{s}_E, \mathbf{s}_U, \mathbf{s}_M, \mathbf{s}_I) \in \mathcal{S}$, representing the environment, the human user, the ML agent, and the interface, respectively. Adopting this state decomposition in our alarm prioritization example,

- $\mathbf{s}_E$ represents the underlying network where each network entity, $n_i \in \mathcal{N}$ can be either be in a NORMAL or ERROR state with down-time, an assigned prioritization, and a boolean variable indicating if the error is currently being addressed;
- $\mathbf{s}_U$ encodes the human operator's current objective and relevant properties, for example, cognitive load and skill-level, and traits such as frustration, distractibility, independence and neediness [6, 7];
- $\mathbf{s}_M$ would reflect the ML model and include parameters, hypotheses and performance characteristics;
- $\mathbf{s}_I$ represents the state of different interface elements e.g., which alarms are currently displayed, the level of detail presented and active modalities.

Actions The possible actions denoted $\mathbf{a}_I \in \mathcal{A}_I$ map to the system adaptions described in [1]. As a communicative layer, the UI's actions can be split into three basic parts $\mathbf{a}_I = (\mathbf{a}_{I-E}, \mathbf{a}_{I-U}, \mathbf{a}_{I-M})$, that is, actions that affect the environment, user, and machine learner. In our running example,

- $\mathbf{a}_{I-E}$ would be the assigned prioritization for each network entity, which would impact whether it was being fixed and the length of down-time.
- The user-affecting actions $\mathbf{a}_{I-U}$ would include changes to "how" of the interaction—style, modality, frequency—and "what", i.e., the informational content including the quantity, quality and abstraction level presented to the user. Each alarm is associated with a visual element $v_i \in V$ and a subset of possible actions are to show or hide elements, $\{\forall v_i \in \mathcal{V} \quad \text{SHOW}_i, \text{HIDE}_i\} \subseteq \mathcal{A}_I$. An optimized policy would likely select actions partially based on U's current state, e.g., fewer alarm elements would be shown to a human operator with low-skill and high mental workload. Finally, to assist the user, the interface may display recommended prioritization levels from the ML agent. Label queries may also be shown to U, i.e., alarms that are not currently active, but in M's database[2].
- As the human operator labels alarms, actions $\mathbf{a}_{I-M}$ would be needed to "show" the ML agent the human labelings. Estimated noise levels or "weights" that depend on the inferred user state can also be provided.

Transitions Given our factored state representation, the state transitions are similarly decomposed for each state variable,

$$p(\mathbf{s}'|\mathbf{s}, \mathbf{a}_I) = p(\mathbf{s}'_E|\mathbf{a}_I, \mathbf{s}_E)p(\mathbf{s}'_U|\mathbf{a}_I, \mathbf{s}_U)p(\mathbf{s}'_M|\mathbf{a}_I, \mathbf{s}_M)p(\mathbf{s}'_I|\mathbf{a}_I, \mathbf{s}_I) \tag{1}$$

Notably, our model assumes that each state variable evolves conditioned on the actions made by the interface (and its previous state). In other words, the user and ML agents only interact with each other (and the environment) via the interface. This assumption holds for alarm prioritization and simplifies the transition model. Nevertheless, these independence assumptions may be altered depending on application requirements.

[2]In our example, these "fictional alarm" queries would be displayed separately to avoid confusing the user about the current state of the network.

Observations For many real-world systems, observations are rich and complex. We decompose each observation into emissions from each state variable $\mathbf{o} = (\mathbf{o}_E, \mathbf{o}_U, \mathbf{o}_M, \mathbf{o}_I)$, which entails a factored probability distribution,

$$p(\mathbf{o}|\mathbf{s}, \mathbf{a}_I) = p(\mathbf{o}_E|\mathbf{s}_E, \mathbf{a}_I) \cdot p(\mathbf{o}_U, |\mathbf{s}_U, \mathbf{a}_I) \cdot p(\mathbf{o}_M, |\mathbf{s}_M, \mathbf{a}_I) \cdot p(\mathbf{o}_I|\mathbf{s}_I, \mathbf{a}_I). \quad (2)$$

The state of the interface would normally be fully visible to itself, $p(\mathbf{o}_I = \mathbf{s}_I|\mathbf{s}_I, \mathbf{a}_I) = 1$, but other variables are likely to be only partially-observable. For example, erroneous network nodes may fail to emit alarms. A portion of the ML agent's properties $\mathbf{s}_M$ may be transparent, depending on how tightly integrated the interface and learner are within the system. Typical observations from the ML agent, $\mathbf{o}_M$ would include the queries it is making of the user, and its predictions, i.e., the alarm prioritization levels.

Of particular interest is the user's state and current objective[3]. Hence, inferences about the user have to be drawn from visible actions, $\mathbf{a}_U$ (e.g., mouse clicks, menu selections) and sensor readings, $\mathbf{x}_U$ (e.g., eye gaze, pupillometry, biometric). As such, we specify

$$p(\mathbf{o}_U|\mathbf{s}_U, \mathbf{a}_I) = p(\mathbf{a}_U, \mathbf{x}_U|\mathbf{s}_U, \mathbf{a}_I). \quad (3)$$

Rewards Finally, the reward function $\mathcal{R}(\mathbf{s}, \mathbf{a}_I)$ depends on the application. Rewards can be specified for goal achievement, while costs (negative rewards) may be related to increased user mental workload, and length of time till task completion. For alarm prioritization, we can assign to each state a cost (negative reward) proportional to the number of nodes in an error state. For IML systems focussed on ML model creation, the estimated model accuracy may be used as a reward.

4 Discussion: Key Benefits and Challenges Ahead

Once the POMDP has been specified, a variety of solvers [8, 9] are available for computing an (approximately) optimal policy. Unlike *ad hoc* methods, the presented POMDP formalizes system adaptations as a decision making process, and naturally captures the sequential nature of the problem and the inherent trade-offs.

Importantly, the modeler is freed from specifying the causes or contextual triggers for adaptations. Rather, the triggers emerge from the solution process embedded in the policy. In other words, the triggers are *automatically determined* from the model specifications. As such, this approach conveys another notable advantage: instead of simply being executed, policies can studied to gain insights into the optimal triggers and adaptations under varying conditions and modeling assumptions.

Despite the aforementioned benefits, several challenges still impede practical application of the formulation as presented:

Model Specification In practice, the modeler would have to specialize the model to their given application, and complex POMDPs can be difficult to describe. A high-level language such as RDDL [10] may be useful in this regard, but what if the model is only partially known? This problem is particularly apparent when attempting to define the user state transition function—a task that requires cognitive modeling, human factors and/or HCI expertise. One can turn to Bayes-adaptive POMDPs [11] and learn the POMDP during execution, but scaling up Bayes-adaptive methods for this task remains a significant challenge.

Solution Derivation The large, complex and possibly continuous state, observation and action spaces pose a problem to even state-of-the-art solvers. Large state spaces can be mitigated by online Monte-Carlo planning [8, 12] that break the curse of dimensionality. But despite recent innovations [13, 14, 15, 16], POMDP solvers have yet to adequately handle complex continuous observation and action spaces. In the interim, it is possible to discretize these spaces *a priori*, but this may result in suboptimal policies. Since the UI performs several sub-actions concurrently, it may be possible to exploit the factored nature of the action space to derive efficient specialized methods.

Moving forward, addressing these key challenges would enable POMDPs to simplify and systemize the creation of adaptive user interfaces, a key ingredient for improving the usability and effectiveness of IML systems.

[3] We distinguish between the user's current objective and the system goal. For example, the overall system goal is the correct prioritization of alarms, but the user's current objective may be to look-up information regarding a particular alarm type.

References

[1] K. M. Feigh, M. C. Dorneich, and C. C. Hayes, "Toward a Characterization of Adaptive Systems: A Framework for Researchers and System Designers," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 54, no. 6, pp. 1008–1024, 2012.

[2] B. Settles, "From theories to queries: Active learning in practice," in *Active Learning and Experimental Design Workshop (in conjunction with AISTATS 2010)*, vol. 16, pp. 1–18, 2010.

[3] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza, "Power to the People : The Role of Humans in Interactive Machine Learning," *AI Magazine*, vol. 35, no. 4, pp. 105–120, 2013.

[4] J. D. Williams and S. Young, "Partially observable Markov decision processes for spoken dialog systems," *Computer Speech and Language*, vol. 21, no. 2, pp. 393–422, 2007.

[5] S. Amershi, B. Lee, A. Kapoor, R. Mahajan, and B. Christian, "Human-guided machine learning for fast and accurate network alarm triage," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 2564–2569, 2011.

[6] J. M. Carroll, *HCI models, theories, and frameworks: Toward a multidisciplinary science*. Morgan Kaufmann, 2003.

[7] B. Hui and C. Boutilier, "Who's Asking For Help? A Bayesian Approach to Intelligent Assistance," *Proceedings of the 11th international conference on Intelligent user interfaces*, pp. 186–193, 2006.

[8] D. Silver and J. Veness, "Monte-Carlo Planning in Large POMDPs," in *Advances in Neural Information Processing Systems*, pp. 2164–2172, 2010.

[9] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 1025–1030, 2003.

[10] S. Sanner, "Relational dynamic influence diagram language (RDDL): Language description," tech. rep., 2010.

[11] S. Ross, B. Chaib-draa, and J. Pineau, "Bayes-adaptive POMDPs," in *Advances in Neural Information Processing Systems*, pp. 1225–1232, 2007.

[12] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT : Online POMDP Planning with Regularization," in *Advances in Neural Information Processing Systems*, pp. 1772–1780, 2013.

[13] J. Hoey and P. Poupart, "Solving POMDPs with continuous or large discrete observation spaces," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 1332–1338, 2005.

[14] A. Pas, *Simulation Based Planning for Partially Observable Markov Decision Processes with Continuous Observation Spaces*. Masters thesis, Maastricht University, 2012.

[15] Chris Mansley, A. Weinstein, and M. Littman, "Sample-based Planning for Continuous Action Markov Decision Processes," *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling*, pp. 335–338, 2010.

[16] V. D. Broeck and K. Guy Driessens, "Automatic Discretization of Actions and States in Monte-Carlo Tree Search," *Proceedings of the ECML/PKDD 2011 Workshop on Machine Learning and Data Mining in and around Games*, 2011.