Back to the Future: Solving Hidden Parameter MDPs with Hindsight

Canmanie T. Ponnambalam
Delft University of Technology
The Netherlands
c.t.ponnambalam@tudelft.nl

Danial Kamran
Karlsruhe Institute of Technology
Germany
danial.kamran@kit.edu

Thiago D. Simão
Radboud University, Nijmegen
The Netherlands
thiago.simao@ru.nl

Frans A. Oliehoek
Delft University of Technology
The Netherlands
f.a.oliehoek@tudelft.nl

Matthijs T. J. Spaan
Delft University of Technology
The Netherlands
m.t.j.spaan@tudelft.nl

ABSTRACT

Reinforcement learning is limited by how the task is defined at the start of learning and is generally inflexible to accommodating new information during training. In contrast, humans are capable of learning from hindsight and can easily incorporate new information to gain insight into past experience. Humans also learn in a more modular fashion that facilitates transfer of knowledge across many different types of problems, resulting in flexible and sample efficient learning. This ability is often missing in reinforcement learning, as agents should generally be trained from scratch even when there are minor disruptions or changes in the environment. We aim to empower reinforcement learning agents with a modular approach that allows learning from hindsight, giving them the ability to learn from their past experience after new information is revealed. We address partially-observable problems that can be modeled as hidden parameter MDPs, where crucial state information is not observable during action selection but is later revealed. Our work focuses on the benefits of separating the tasks of policy optimization and hidden parameter estimation. By decoupling the two, we enable more data-efficient learning that is flexible to changes in the environment and can readily make use of existing predictors or offline data-sets. We demonstrate in discrete and continuous experiments that learning from hindsight offers scalable and sample efficient performance in HiP-MDPs and enables transfer of knowledge between tasks.

KEYWORDS

Reinforcement Learning, Partial Observability, Transfer Learning

1 INTRODUCTION

Reinforcement learning is often touted as the closest mathematical framework we have to human learning, though it provides a relatively rigid mimicry of the human decision-making process. One property humans exhibit in decision making is modularity, or breaking tasks into hierarchies of separable problems [27]. This leads to efficient learning and generalizing capabilities. Humans can also expand and compress their local models seamlessly, adding or removing information input as needed [8]. In most reinforcement learning algorithms, the state space is specified and remains fixed either for the entire life of the agent, or for a specified task with known boundaries. This means excessive information can be included in the state that may not be relevant at all times. A truly intelligent agent should have separable components and capabilities that can be distributed and shared and should not rely on only a single representation of their world [3].

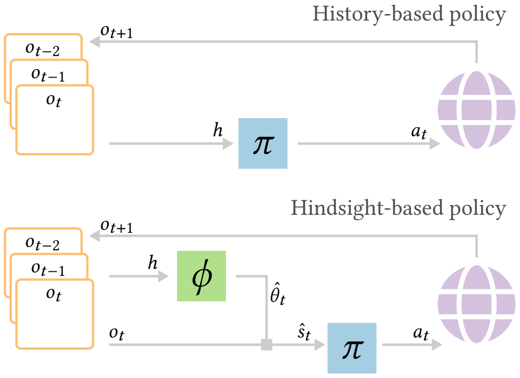

Figure 1: An overview of hindsight in comparison to learning from a history. The history-based policy maps stacks of observations to actions. The hindsight-based policy takes a predicted hidden parameter value in conjunction with a single observation as input and maps this to actions.

An example of adaptive information use that comes very naturally to humans is the idea of hindsight: *"If only I had known this sooner, I could have done something differently"*. After new knowledge is revealed, we reason that we would have done something else at a particular time, given this knowledge. This implies that we can expand the input to our decision-making model to include additional information, a flexibility not inherent to the current design of reinforcement learning agents.

In this paper, we introduce a method to embed hindsight reasoning into a reinforcement learning algorithm suitable for partially-observable problems. We consider settings which can be modeled as hidden parameter Markov decision processes (HiP-MDPs). Many types of partially-observable problems can be adapted to the HiP-MDP model, for example a robot navigating terrains of varying friction levels [4], manipulating objects of differing mass [28], or interacting with opponents that follow one of several possible strategies [31]. The HiP-MDP model has also proven to be applicable

Proc. of the Adaptive and Learning Agents Workshop (ALA 2022), Cruz, Hayes, da Silva, Santos (eds.), May 9-10, 2022, Online, https://ala2022.github.io/. 2022.

to real-world problems such as wildlife conservation [19] and is highly applicable in transfer learning as a way to model related tasks [11, 32]. We consider a specific case of HiP-MDP where the hidden information is revealed at some point in the future (after action selection). The intuition behind our method is that learning from the certain future is easier than learning from the uncertain past. As an example, consider an autonomous vehicle (AV) that is merging onto a highway that can only find an optimal policy if it knows how cooperative the other drivers in the desired lane are. Whether or not a driver is cooperative is revealed only *after* the AV attempts to merge. Recognizing this after it has been revealed is not difficult; the harder task is to predict whether a vehicle is cooperative from its observable behaviour.

In this work, we focus on problems where the hidden parameter is revealed *in hindsight*. Existing methods absorb the task of predicting the hidden parameter into the policy optimization task, stacking observations into histories or attempting to model latent parameters. These methods do not have a mechanism to learn directly from instances where the hidden parameter is revealed. Our approach (pictured in 1) uses hindsight to label past experience and trains a supervised learning model on the labeled data to predict the hidden parameter from observations. The output of the supervised learning model is used online at the time of action selection. As the agent gains more data and more confidence in its prediction, it also learns to take information-gathering actions when doing so will improve its value-maximizing policy. We utilize two separable learning components, enabling the use of predictors learned on offline data or for integrating models provided by an expert. This paper introduces our approach and describes its several benefits: it (1) conducts policy optimization over a compact state space rather than over histories, (2) enables abstracting away information that is not relevant to the prediction task, (3) allows for incorporation of existing prediction models or models trained on offline data, (4) makes explicit the agent's prediction of the hidden state at every time step, and (5) facilitates transfer of either the policy or predictor between related tasks.

The empirical evaluation demonstrates in both discrete and continuous problems that by approaching the problem this way, we can outperform an agent learning a policy on a history of observations by making efficient use of data. It also shows how learning from hindsight enables transfer of both the policy and the predictor between related tasks, exhibiting faster convergence to the same reward as an agent trained from scratch.

2 RELATED WORK

A common approach to solving partially-observable problems with reinforcement learning is to learn a policy on a history of observations and actions, rather than on a single observation (which breaks the Markov assumption required otherwise). We apply hindsight to a particular case of partially-observable problem where the full state is never observed but is revealed at some point. In contrast to an agent that learns from history, our method provides explicit feedback on the agent's prediction and confidence of the true state.

Other methods that model learning with hidden parameters commonly involve reasoning over the latent parameter space particularly to facilitate transfer learning between related tasks [10, 21, 28] or reason about an opponent strategy modeled by those latent variables [30]. We consider a similar model but focus on how to incorporate hindsight knowledge.

Our setting is conceptually related to the idea of influence-based abstraction [18]. The hidden parameter can be considered as the 'influence source' and our supervised learning model ϕ is related to the idea of an influence predictor [7]. However, we do not assume that we have access to the model that describes how the parameter influences our local state. Moreover, we derive a measure of uncertainty in our parameter predictor, and use this to encourage information-gathering actions in the policy.

There has been recent emergence of methods that are more closely related to the work we present here. Hindsight has been applied to the learning problem as an additional signal that corresponds to how much a state-action pair contributes to a particular outcome in the future [6, 15]. Other recent applications of hindsight fall into two categories: those where hindsight is reflected in the reward function (or goal), and those were hindsight reveals state information. Hindsight Experience Replay lies in the first category, and saves trajectories in memory in order to reuse the experience after re-labeling these trajectories according to different goals they may be better suited for [1]. This approach has been extended to policy gradient methods [22] and was later generalized to tasks which are not specifically goal-oriented by applying techniques from inverse reinforcement learning [13]. These approaches have in common that they label saved trajectories in order to reuse data and learn multiple tasks more efficiently. In their case, the labels are reflected in the rewards of a trajectory, which differ between tasks while the transition dynamics are unaffected. In contrast, we consider hindsight not in terms of the goal but in the form of state information.

In the partially-observable setting, learning from hindsight has been applied in the form of posterior value functions to achieve better sample efficiency from revealed state information [17]. Our approach also considers problems with partial observability, however, we define an auxiliary supervised learning problem and use hindsight to label past experience resulting in a modular learning approach that provides flexibility in the application of hindsight and facilitates transfer.

3 BACKGROUND

This section introduces preliminary material that serves as a foundation for our method.

3.1 Markov Decision Processes

We use reinforcement learning to learn to optimize a task modeled as a *Markov decision process* (MDP) where the environment dynamics are unknown. An MDP is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ that describes a sequential decision making problem [20]. The variables $\mathcal{S}$ and $\mathcal{A}$ denote finite state and action spaces, $\mathcal{T}$ and $\mathcal{R}$ are transition and reward functions, and γ is a discounting factor ($0 \leq \gamma \leq 1$) that determines how far into the future to take into account. At each time step t, a reinforcement learning agent observes the state of the environment $s_t \in \mathcal{S}$ and chooses an action $a_t \in \mathcal{A}$. Upon executing action a_t, the environment transitions to a new state $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)$, according to the environment transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ which maps state-action pairs to a distribution over next states. The agent receives a reward r_t according to the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$. The goal of the reinforcement learning agent is to learn a policy π that maps states to actions. The value $V^{\pi}(s)$ of following a policy π from a state s is the sum of the expected discounted reward of the state, given by $V^{\pi}(s) = \mathbb{E}_{\pi} \sum_{t=0}^{\infty} \gamma^t r_t$. An optimal policy π^* is one that maximizes this value for every state.

A *partially-observable MDP* (POMDP) generalizes the case described above to problems where the full state cannot be observed [9].

3.2 Hidden Parameter Markov Decision Proceses

A *hidden parameter MDP* (HiP-MDP) is a type of the more general POMDP with two strong assumptions: that the hidden parameter has no dynamics and remains fixed for the entire duration of each task [5]. A HiP-MDP is a class of tasks where each instance forms a complete MDP. Formally, a HiP-MDP is a tuple: $\langle \mathcal{S}, \mathcal{A}, \Theta, \mathcal{T}, \mathcal{R}, \gamma, \mathcal{P}_{\Theta} \rangle$, where as in an MDP, $\mathcal{S}$ and $\mathcal{A}$ refer to the state and action spaces, $\mathcal{R}$ to the reward function and γ corresponds to the discount factor. For simplicity, we can assume the reward function is shared across instances. At the beginning of an episode, a hidden parameter $\theta \in \Theta$ is sampled from $\mathcal{P}_{\Theta}(\cdot)$, defining the underlying MDP, however the value of θ is not revealed to the RL agent. The transition function is conditional on the hidden parameter, and gives the probability of transitioning to state s' after taking action a from state s when the hidden parameter has value θ: $\mathcal{T}(s'|s, \theta, a)$. The joint observed state and hidden parameter together form a Markovian state. In the HiP-MDP formulation, the state space is akin to the observation space of a general POMDP.

3.2.1 The tiger problem.

The *tiger* problem is a classical POMDP example [9] that can be modeled as a HiP-MDP. We will use this problem as our running example throughout the paper. In this problem, an agent decides which of two doors to open; behind one of the doors is a tiger and behind the other a treasure. The agent can open the left door, open the right door or listen; if it listens, it can hear the tiger either behind the left or right door, with some probability of noise. There is a -1 reward for every action taken, a +10 reward for opening the door to the treasure and a -100 reward for opening the door to the tiger. This problem has two states describing the true location of the tiger (behind the left door or behind the right door) and two noisy observations (sound from the left and sound from the right). The agent must execute listening actions to gain information about where the tiger is located. Only when it is confident should it open the door it believes holds the treasure. The amount of noise in the observation dictates how much history is needed in order to infer the location of the tiger with high probability. The tiger problem can be modeled as a HiP-MDP by defining the true location of the tiger as the hidden parameter and the noisy observations as the states.

4 ENCODING THE PAST WITH THE FUTURE

We consider problems where crucial information is hidden from the agent while it interacts with the environment, but revealed after the episode has terminated. Our approach is specified for tasks that can be modeled as a hidden parameter Markov decision process (HiP-MDP). In this section, we describe our method to incorporate hindsight and decouple hidden state identification from policy optimization. We start by formalizing our framework then show how an RL agent can use this framework to speed-up the learning process.

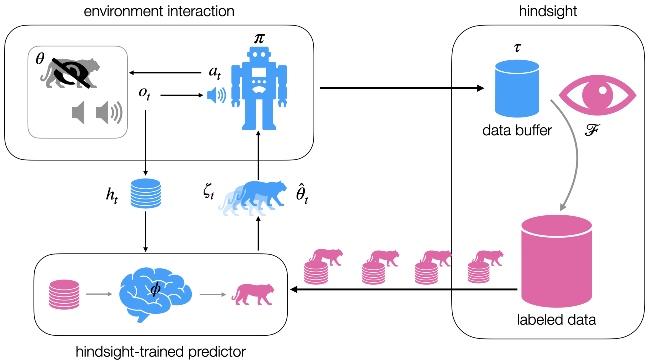

Figure 2: An overview of our method for hindsight-enabled reinforcement learning. The agent learns to select actions based on its joint observation, prediction of the hidden parameter value and uncertainty in the prediction. Past experience is labeled *in hindsight* and fed to a supervised learning model that learns to predict the hidden parameter from histories.

4.1 Learning from hindsight

Let us consider again the example of the tiger problem [9] modeled as an HiP-MDP, where the hidden parameter is the location of the tiger. *After* the agent opens a door and ends the episode, it is immediately obvious where the tiger was located based on the reward received. This is the hindsight knowledge that was not available at the time the agent selected actions.

In our framework, the agent learns (from hindsight) to predict the hidden parameter during action selection, and acts according to its observation, hidden parameter prediction and uncertainty in the prediction, which together form a Markovian state. A depiction of the two parallel learning cycles, one where the policy is trained online based on interaction with the environment and the output from the supervised learner, and the other were the supervised learner is trained on hindsight-labeled past experience, is shown in Figure 2.

Formally, a hindsight-enabled HiP-MDP is an extension of a HiP-MDP, defined as $\mathcal{H}_{\mathcal{F}} = \langle \mathcal{S}, \mathcal{A}, \Theta, \mathcal{T}, \mathcal{R}, \gamma, \mathcal{P}_{\Theta}, \mathcal{F} \rangle$, where $\mathcal{F}$ is an additional parameter that encompasses hindsight in the form of expert knowledge. The hindsight function ($\mathcal{F} : (\mathcal{S} \times \mathcal{A} \times \mathcal{R})^m \rightarrow \Theta$) maps m (state, action, reward) transitions from a single episode to the true value of the hidden parameter:

$$\mathcal{F}(\tau) = \theta,$$

where $\tau \in (\mathcal{S} \times \mathcal{A} \times \mathcal{R})^m$ refers to an m-length sequence of transitions in an episode.

While $\mathcal{F}$ is potentially defined over a large space, it is, in practice, dependent on very few indicator variables. The hindsight function for the tiger problem needs only the last (action, reward) transition (i.e. $m = 1$) to return the true location of the tiger for the elapsed episode. After an episode is completed and the parameter is revealed, the correct label can be applied to all transitions in the episode. The intuition is that hindsight is easy to define (it is easy to realized something after it has occurred). The more difficult part is predicting the hidden parameter online from observations at the point of action selection; we delegate this task to a supervised learning model.

The hindsight-labeled data is used to train a model to predict hidden parameters from histories. The trained supervised learner ϕ takes a history and outputs the predicted value of the hidden parameter $\hat{\theta}$ and a measure ζ of uncertainty in this prediction.

$$\phi(h_k) = (\hat{\theta}, \zeta),$$

where h_k is a k-length sequence of states and actions from the same episode. We propose a method for the hindsight-enabled HiP-MDP that learns as though it is in a simple MDP with an augmented state space composed by the observed state, a prediction of the hidden parameter, and uncertainty space. Before introducing this formally, we elaborate on why conditioning the policy on the uncertainty in the prediction is crucial. Without doing this we run into an issue, as even with a perfect model the history may not be informative enough to predict the hidden parameter. The agent must consider the uncertainty in the prediction in order to learn to take information-gathering actions.

4.2 Mitigating uncertainty

In the prediction task, our method requires a measure of uncertainty ζ to be output with each prediction. In a Bayes-Adaptive POMDP, the uncertainty is represented by visitation counts [23]. In our experiments, this measure is given according to the class probabilities output by a supervised learner performing classification. In more complex models or regression tasks, we suggest the use of an ensemble of models to provide a practical heuristic for this uncertainty [12, 26].

The prediction uncertainty can be reduced in two ways: improve the prediction model or provide a more informative history. As the agent interacts with the environment, it collects more data which is used to train the model. In theory, with this continuous training the model will improve until it can no longer be improved with additional (or more diverse) data.

There is some amount of uncertainty that is irreducible given a particular history. For example, without any observations, a predictor will be (at best) as uncertain in its prediction as dictated by the initial hidden parameter distribution of the environment. After exhaustive data collection and model training, the remaining uncertainty forms a measure of how certain a prediction can be made given the input. This uncertainty can only be reduced by taking informative actions. By conditioning our policy on the uncertainty in the model, we empower the agent to learn to take uncertainty-reducing (or information-gathering) actions when doing so will improve its policy. This is achieved by augmenting the state-space with this uncertainty.

The hindsight agent learns on the predicted state:

$$\hat{s} = \langle s, \hat{\theta}, \zeta \rangle,$$

where s is the observed state, $\hat{\theta}$ is the predicted hidden parameter and ζ is the uncertainty in the prediction. This corresponds closely to a belief state in the solution to a POMDP, however calculating such a belief generally requires knowledge of the observation probabilities and dynamics in the environment. Our approach assumes neither are available and instead provides a mechanism to use hindsight in a reinforcement learning setting.

Algorithm 1: Learning in hindsight-enabled HiP-MDPs

```
   Input: Hindsight function $\mathcal{F}$
   Input: Length of history $k$
   Input: A value to represent empty history elements $\iota$
   Input: Number of learning steps $N$
   Input: Learning rates for Q and $\phi$ as $\alpha$ and $\beta$
1  Initialize policy $\pi$
2  Initialize predictor $\phi$
3  Set $t = 0$
4  while $t < N$ do
5      $h = \{\iota_0, ..., \iota_k\}$
6      $\tau = \emptyset$
7      Observe $s_t$
8      $(\hat{\theta}_t, \zeta_t) = \phi(h)$
9      $\hat{s}_t = \langle s_t, \hat{\theta}_t, \zeta_t \rangle$
10     repeat
11         $a_t = \pi(\hat{s}_t)$
12         Execute $a_t$, receive $s_{t+1}, r_t$
13         $\tau \leftarrow Append(\tau, \langle o_t, a_t \rangle)$
14         $h \leftarrow PushToQueue(h, \langle s_t, a_t \rangle)$
15         $(\hat{\theta}_{t+1}, \zeta_{t+1}) \leftarrow \phi(h)$
16         $\hat{s}_{t+1} \leftarrow \langle s_{t+1}, \hat{\theta}_{t+1}, \zeta_{t+1} \rangle$
17         $Q(\hat{s}, a_t) \leftarrow (1-\alpha)Q(\hat{s}, a_t) + \alpha(r_t + \gamma \max_a Q(\hat{s}_{t+1}, a))$
18         $t \leftarrow t + 1$
19         $s_t \leftarrow s_{t+1}$
20         $\hat{s}_t \leftarrow \hat{s}_{t+1}$
21     until IsTerminal();
22     $\theta = \mathcal{F}(\tau)$
23     $X \leftarrow [[x_i, \cdots, x_{i+k}] : \forall i \in [0, |\tau| - k - 1]]$
24     $y \leftarrow [\theta : \forall i \in [0, |\tau| - k - 1]]$
25     $\phi \leftarrow Train(\phi, X, y, \beta)$
26 end
   Result: State predictor $\phi$ and policy $\pi$
```

4.3 Algorithm

An overview of our method for learning in a hindsight-enabled HiP-MDP is described in Algorithm 1.

At the beginning of training, the episode memory τ is empty and the history queue is initialized to values that indicate empty elements. An initial prediction of the hidden parameter is made and forms the augmented state jointly with the prediction uncertainty and current observed state (Line 9). During an episode, the agent selects an action according to its policy given the augmented state (Line 11). It executes the action and the environment returns an observation and reward (Line 12). The transition is stored both in the episode memory τ and pushed to the history queue (lines 13 and 14). The agent predicts the next state hidden parameter according its updated history (Line 16) and updates its Q-values (Line 17). This learning loop continues until the episode terminates (Line 21). At this point, the hindsight function provides the true value of the hidden parameter for the elapsed episode (Line 22). This label is applied to k-length sequences of the episode memory τ (lines 23 and 24) and the resulting labeled data trains the supervised learner ϕ (Line 25). In practice, ϕ is not trained after every episode but after a number of episodes defined by the designer. When doing so, to improve stability it can be trained on an appropriately sampled batch of data rather than the entire memory.

5 BENEFITS OF DECOUPLING HIDDEN PARAMETER PREDICTION AND POLICY OPTIMIZATION

By separating the task of predicting the hidden parameter from learning the policy our method offers several benefits, two of which are detailed in this section. Because the observation space provided to the predictor is independent of the space over which the policy is learned, it allows for different abstractions of the problem. In addition, decoupling the tasks facilitates transfer of components under certain conditions.

5.1 Abstraction

Our method provides an opportunity to give to the hindsight agent only the information that is relevant for predicting the hidden parameter. The space of the history provided to the supervised learner is independent of the problem space the policy is learned on. An expert can isolate the parts of the observation space that are relevant to the hidden parameter predictor from the observation space that is relevant to the policy. For example, when predicting the cooperativeness of another vehicle, its dynamics and distance to other vehicles around it may be relevant. However, when learning a policy to merge into traffic, only the predicted cooperativeness and the distance from the other vehicle to the agent are relevant.

5.2 Transfer Learning

Another crucial benefit to taking modular approach is that we maintain separable pieces that can be reused for new tasks, avoiding the need to learn everything from scratch. Our method facilitates transfer of either the predictor or the policy. Consider two hindsight-enabled HiP-MDPs $\mathcal{H}_\mathcal{F}$ and $\overline{\mathcal{H}_\mathcal{F}}$ that share the same state, action and hidden parameter space but where the reward and transition dynamics differ. Under the following sufficient conditions, the predictor ϕ or the policy learned when optimizing $\mathcal{H}_\mathcal{F}$ can be re-used when optimizing $\overline{\mathcal{H}_\mathcal{F}}$.

5.2.1 *Predictor transfer.*

The problem of predicting the hidden parameter from state history is preserved between two tasks $\mathcal{H}_\mathcal{F}$ and $\overline{\mathcal{H}_\mathcal{F}}$ if, given the same k-length history of state observations, a perfect hidden parameter predictor ϕ^* would yield the same prediction and uncertainty:

$$(\theta, \zeta_t) = \phi^*(h_t) = \phi^*(\overline{h}_t) = (\overline{\theta}, \overline{\zeta}_t)$$

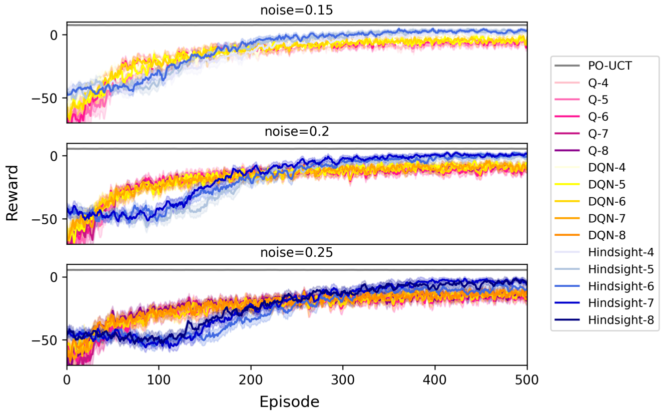

Figure 3: Results from comparing hindsight agents to history-based agents in the tiger domain for varying lengths of history provided. Total episode rewards are averaged over 100 trials with the 90% confidence interval depicted by the shaded region.

In this case, assuming the predictor ϕ learned from task $\mathcal{H}_\mathcal{F}$ has converged to the optimal predictor ϕ^*, we can immediately inject it as $\overline{\phi}$ when training on task $\overline{\mathcal{H}_\mathcal{F}}$, requiring only the policy $\overline{\pi}$ to be learned.

5.2.2 *Policy transfer.*

The policy can be transferred between tasks if the same policy optimizes both tasks. Two tasks $\mathcal{H}_\mathcal{F}$ and $\overline{\mathcal{H}_\mathcal{F}}$ with value functions $\mathcal{V}$ and $\overline{\mathcal{V}}$ share an optimal policy $\pi^*(\cdot \mid \hat{s} = \langle o, \hat{\theta}, \zeta \rangle)$, if:

$$\mathcal{V}(\pi^*) \geq \mathcal{V}(\pi') \ \forall \pi'$$
$$\overline{\mathcal{V}}(\pi^*) \geq \overline{\mathcal{V}}(\overline{\pi}') \ \forall \overline{\pi}'$$

Under this condition, the policy π^* learned from task $\mathcal{H}_\mathcal{F}$ can be directly applied to task $\overline{\mathcal{H}_\mathcal{F}}$ requiring only the predictor to be retrained during policy optimization of task $\overline{\mathcal{H}_\mathcal{F}}$.

6 EVALUATION

We evaluate our hindsight-enabled method in two problems, a discrete task and a continuous control task. We show how using hindsight allows for efficient use of data while scaling with increased history length. We also demonstrate transfer of both the policy and the predictor enabled by our method.

6.1 Discrete State Space

In the tiger problem, we apply our hindsight model and compare an agent learning with hindsight to both a Q-learning agent and a deep Q-learning agent that observe a history of observations which form the new state space. The history-based space grows exponentially with the history length and a tabular agent can quickly outgrow the memory capacity of a standard machine. The deep Q-learning agent serves as a second baseline as it uses the same sized neural network as each hindsight agent to handle the large history-based observation space, avoiding the memory issues of the tabular agent.

The hindsight agents apply Q-learning [29] to the joint (observation, hidden parameter prediction, uncertainty) space according to predictions made by a supervised learner. We use a multi-layer perceptron as a classifier ϕ that predicts the true location of the tiger, trained on past experiences. The class probability output by the model is used as the measure of uncertainty in the prediction; it is discretized into three bins ([0 - 0.50), [0.50 - 0.99), [0.99 - 1.0]). While this can introduce some approximation errors, it also makes the solution more compact, ensuring the method remains scalable [24]. The augmented state space of the hindsight agent grows only linearly with the number of bins used in this discretization, so its size is independent of the length of history. The input space of the predictor is the same history length as the input space of the tabular agent and the deep Q-learning agents.

We compare hindsight agents and history-based agents with different history input lengths k for varying noise probabilities in the environment. All Q-learning parameters were tuned individually to ensure good performance for each agent. The hyper-parameters relevant to the hindsight agents are displayed in Table 1.

The Partially-Observable Upper Confidence Bound (PO-UCT) [25] solution is included for reference, implemented using the pomdp_py library [33]. The performance of PO-UCT is generally not achievable by a reinforcement learning agent, as applying it requires complete knowledge of the observation, transition and reward functions. We plot the expected return of the policy returned by PO-UCT after 15000 simulations.

In another experiment, we demonstrate transfer of the policy considering two versions of the tiger domain where the noise probabilities differ. The policy is first trained by applying our method to the tiger task with noise probability 0.15. We then transfer the Q-values to a hindsight agent faced with a new task where the probability has been changed to 0.20. Only the predictor is retrained in the second task while the Q-values remain fixed; this also means that exploration is not necessary. We compare the transfer agent to another hindsight agent that learns a new policy from scratch to show the effectiveness of the transferred policy. We ensure that all non-transfer agents start with as low an exploration factor as possible and found that setting an initial ϵ of 0.1 resulted in the best performance; both higher and lower values affected the speed of convergence negatively.

6.1.1 Hindsight out-performs the history-based agent.

The results of our first experiments are shown in Figure 3. The hindsight agents converge to a better performing policy in each of the tasks. The performance of the tabular and deep Q-learning agents are very similar, and both initially achieve a faster increase in performance, but ultimately they do not converge to the same policy found by the hindsight agent within the 500 episodes shown. Empirically, we believe the initial speed-up to be due to the higher learning rates these agents can use, whereas the hindsight agent may demonstrate oscillatory behaviour with similarly high learning rates. The hindsight agents all avoid the lowest rewards at the very start of learning, as our approach enables them to quickly plan against the worst case scenario (where uncertainty is high).

6.1.2 The predictor uncertainty converges to the environment stochasticity.

We examined empirically whether the uncertainty output of our supervised learning is converging to the irreducible uncertainty in the environment as expected. In Figure 4, the prediction probability for hidden state "tiger left" is plotted against training iterations of the predictor ϕ. In the top plot, we see that the predictor cannot improve beyond 50% certainty when the agent has not received any observations; this aligns with the initial distribution of the two hidden parameter values (uniformly random). When the agent has heard the tiger behind the left door 10 times in a row, the predictor is almost certain (we expect this to converge to $1 - 0.10^{10}$). When the agent has received one observation that the tiger was heard behind the left door, the certainty converges to around 90%. This

Table 1: Experimental hyper-parameters for hindsight agents in tiger domain

Parameter	Noise		
	0.15	**0.20**	**0.25**
Training frequency (steps)	30	50	50
Batch size	50	50	100
k (Hidden layer size)	4 (20,)	5 (50,)	6 (60,)
	5 (25,)	6 (60,)	7 (84,)
	6 (36,)	7 (70,)	8 (112,)
Learning rate	0.01	0.001	0.0005

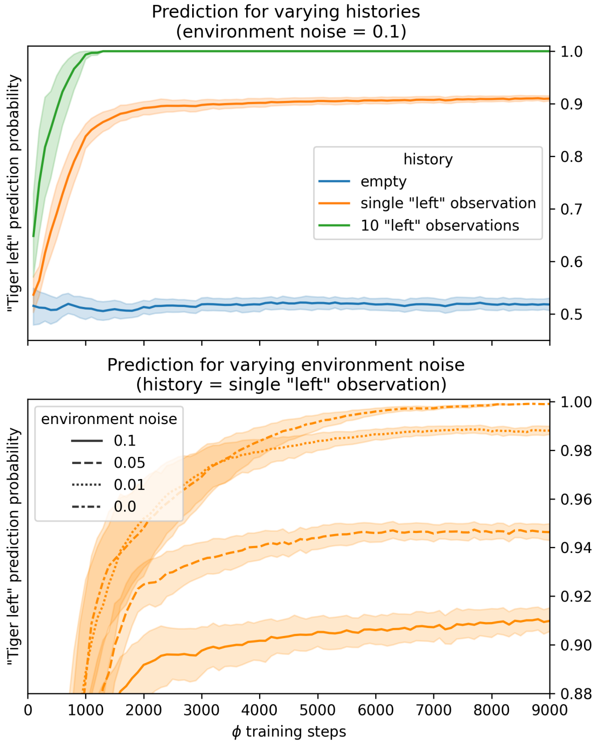

Figure 4: Visualizing the uncertainty output of the supervised learner ϕ averaged over 25 trials with shaded areas depicting the 98% confidence interval.

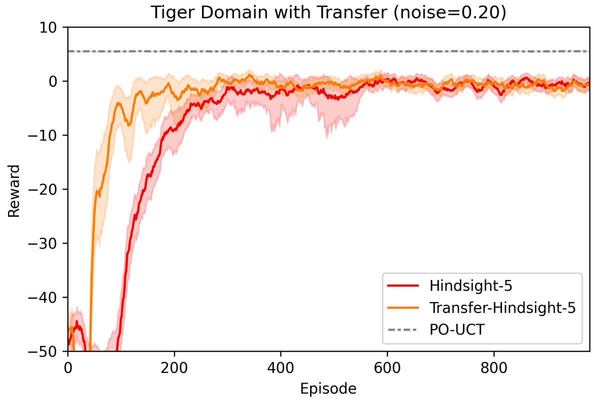

Figure 5: Results for transfer of the policy learned from the tiger domain with 0.15 probability of noise and applied to a new task where noise probability is changed to 0.20. The average of 50 trials is shown with shaded areas depicting the 98% confidence interval.

aligns with the noise in the environment, i.e. the probability of incorrectly observing where the tiger is.

In the bottom plot of Figure 4, we investigate this last case further. The prediction probability given a single observation of the tiger behind the left door is shown against training iterations for four different noise probabilities exhibited in the environment. The uncertainty reflects the environment noise, albeit with a small deviation from the theoretical expectations.

6.1.3 The hindsight policy can speed-up learning in new tasks. The results of the transfer learning experiment are presented in Figure 5. There is a considerable benefit to using the pre-trained policy in the new environment (even in this small problem), and it converges to the same performance as the hindsight agent trained from scratch.

6.2 Continuous State Space

We also apply our method to a modified version of the cartpole environment from OpenAI Gym [2]. In this modification, the agent has to keep the pole balanced on the cart held at a specific target location (left, middle or right of the track). The target position is randomly selected at the beginning of each episode and is not observable by the agent. The reward function depends on the noisy distance to the target $\hat{d}(s_t, \theta_t) \sim \mathcal{N}(d(s_t, \theta_t), (\sigma d(s_t, \theta_t))^2)$ where $d(s_t, \theta_t)$ is the true value:

$$r(s_t, \theta_t) = 1.0 - \frac{\hat{d}(s_t, \theta_t)}{d_{\max}} \tag{1}$$

This partially-observable problem requires some form of memory and is not solvable by a naïve RL agent. In our experiments, we keep a history of previous rewards and states, which provides crucial information about the true target position and forms the state-space for the history-based RL agents as well as the input to the hindsight predictor.

The deep Q-network (DQN) [16] for the history-based agent has a $(2k+4)\times 256\times 512\times 3$ topology where k is the length of the history and 3 is the output layer for estimating the Q values of three possible actions. For the hindsight agent, the DQN has a $6 \times 256 \times 512 \times 3$ topology. In addition to the DQN, the hindsight agent deploys a multi-layer perceptron as the hidden parameter classifier (ϕ) with a $(2k) \times 16 \times 8 \times 8 \times 1$ topology for target position estimation. Due to the different input size (and therefore different network topology), we trained the agents with different learning rates to prevent early over-fitting in order find the best trade-off between fast convergence and stability. This was particularly necessary for the more sensitive history-based agent. We used learning rates of $5e^{-5}$ and $1e^{-4}$ for training the history-based and hindsight agents respectively.

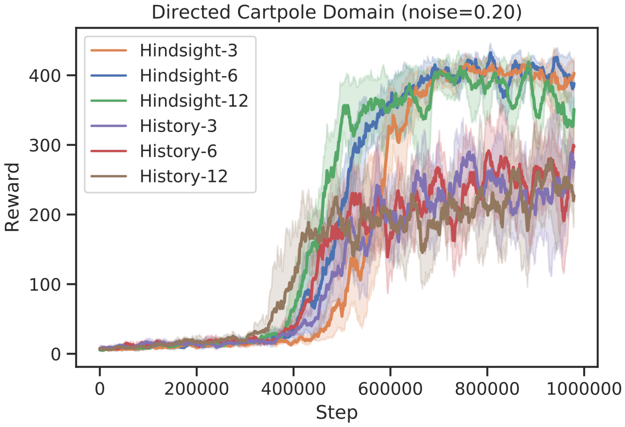

Figure 6: Results comparing the history-based and hindsight agents with different history lengths k in the directed cart-pole environment with target distance noise σ=0.2. The average of 8 trials is shown with shaded areas depicting the 96% confidence interval.

We evaluated transfer of the predictor in two experiments. In the first experiment, we pre-train the predictor on the same environment and provide it to the hindsight agent to learn a policy without updating the predictor. In the second transfer experiment, we consider two different environments that share the same prediction task but different optimal policies. The predictor learned on the previous task is provided to the hindsight agent and used while training a new policy in a more complex environment where the pole length and mass are no longer fixed and instead randomly selected from $[0.1, 1.0]$ interval at the beginning of each episode.

6.2.1 The hindsight agent achieves higher reward faster. Figure 6 shows the total reward for the hindsight and history-based agents considering different history lengths (k=3, 6 or 12). The hindsight agent achieves a much higher total reward with fewer interactions. Increasing the history length results in slightly faster initial learning for both the hindsight and history-based agents, but slower convergence afterward. It appears that the effect of increasing k on the final converged reward is more pronounced in the history-based agent performance than the hindsight agent, though the difference is minor.

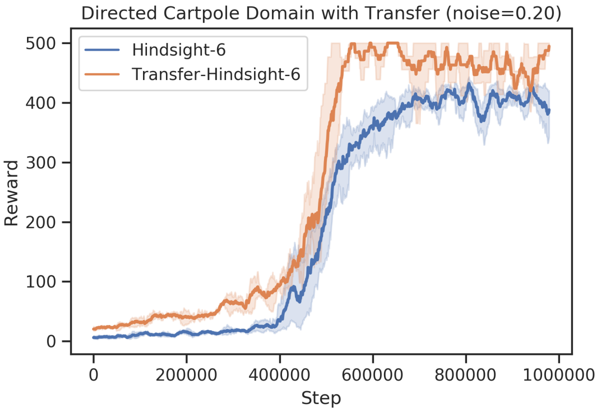

Figure 7: Results for transfer of a pre-trained predictor in the directed cartpole problem. The average of 5 trials is shown with shaded areas depicting the 96% confidence interval. Note that 500 is the maximum achievable reward in one episode.

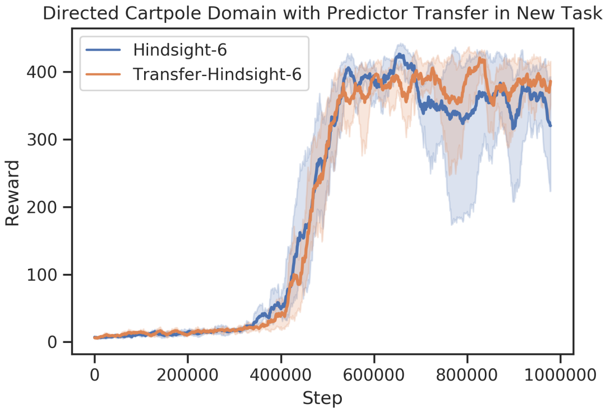

Figure 8: Results for transfer of a predictor trained in a different task in the directed cartpole problem for a new task where pole mass and length are randomly sampled each episode. The average of 5 trials is shown with shaded areas depicting the 96% confidence interval.

6.2.2 The hindsight-trained predictor can speed-up learning in new tasks. Figure 7 depicts the total reward of the hindsight agent during training with and without the pre-trained predictor. The agent with the transferred predictor converges to a better policy faster than the agent training both its predictor and policy from scratch. The advantage of utilizing a frozen predictor instead of training both the policy and predictor together is clear here, as the initial non-stationarity of the predictor no longer provides noisy transitions that need unlearning. Instead, the policy receives more stable estimations from a model that has already converged.

The results of the second transfer experiment are pictured in Figure 8. The policy with the transferred predictor still converges to the same solution even though its predictor was trained in a different environment, demonstrating potential to reuse a predictor trained in other environments without retraining. We observed that for some trials the performance of the hindsight agent worsened after it had seemingly converged. This may be due to catastrophic forgetting in the DQN or the predictor network [14] or due to instability caused by the mismatch in data distribution between the data that trained the predictor and the data generated by the policy learned on the new task. We leave to future work the investigation of how performance of the predictor can be affected by the policy by which data is collected and how stability can be improved in more complex domains.

7 CONCLUSION

This paper introduces a modular method for incorporating hindsight state information into a reinforcement learning algorithm. We believe that many partially-observable problems reveal crucial state information with certainty in the future, even if this information cannot be observed at action selection (and execution) time. We offer a framework for such problems that can be modeled as HiP-MDPs and a method that we show in experiments has the potential to converge quickly and scale efficiently with the required memory of the agent. We further demonstrate that taking such a decoupled approach can facilitate transfer of both the policy and the predictor between tasks where the environment dynamics differ. We believe that our method can enable designers to apply existing predictors or those learned on offline data-sets to further improve the efficiency of reinforcement learning agents. In future work, we would like to see learning from hindsight generalized to any POMDP. In addition, while we focused here on partially-observable problems, it has been shown that hindsight-enabled learning can offer improvements even in fully-observable problems [17], which we hope to explore further.

REFERENCES

[1] Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 5048–5058.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540*.

[3] Rodney A. Brooks. 1991. Intelligence without representation. *Artificial Intelligence* 47, 1 (1991), 139–159. https://www.sciencedirect.com/science/article/pii/000437029190053M

[4] Carlos Diuk, Lihong Li, and Bethany R. Leffler. 2009. The adaptive k-meteorologists problem and its application to structure learning and feature selection in reinforcement learning. In *Proceedings of the 26th International Conference on Machine Learning*. ACM, 249–256.

[5] F. Doshi-Velez and G Konidaris. 2016. Hidden Parameter Markov Decision Processes: A Semiparametric Regression Approach for Discovering Latent Task Parametrizations. In *Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*. IJCAI/AAAI Press, 1432–1440.

[6] Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, and Rémi Munos. 2019. Hindsight Credit Assignment. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 12467–12476.

[7] Jinke He, Miguel Suau, and Frans A. Oliehoek. 2020. Influence-Augmented Online Planning for Complex Environments. In *Advances in Neural Information Processing Systems 33*. Curran Associates, Inc., 4392–4402.

[8] W A Johnston and V J Dark. 1986. Selective Attention. *Annual Review of Psychology* 37, 1 (1986), 43–75.

[9] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101, 1 (1998), 99–134.

[10] Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. 2017. Robust and Efficient Transfer Learning with Hidden Parameter Markov Decision Processes. In *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 6250–6261.

[11] George Konidaris and Finale Doshi-Velez. 2014. Hidden Parameter Markov Decision Processes: An Emerging Paradigm for Modeling Families of Related Tasks. In *AAAI Fall Symposia*. AAAI Press, 46–48.

[12] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 6405–6416.

[13] Alexander Li, Lerrel Pinto, and Pieter Abbeel. 2020. Generalized Hindsight for Reinforcement Learning. In *Advances in Neural Information Processing Systems 33*. Curran Associates, Inc., 7754–7767.

[14] Michael McCloskey and Neal J. Cohen. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *Psychology of Learning and Motivation* 24 (1989), 109–165. https://doi.org/10.1016/S0079-7421(08)60536-8

[15] Thomas Mesnard, Theophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Thomas S Stepleton, Nicolas Heess, Arthur Guez, Eric Moulines, Marcus Hutter, Lars Buesing, and Remi Munos. 2021. Counterfactual Credit Assignment in Model-Free Reinforcement Learning. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 7654–7664.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:arXiv:1312.5602*.

[17] Chris Nota, Philip Thomas, and Bruno C. Da Silva. 2021. Posterior Value Functions: Hindsight Baselines for Policy Gradient Methods. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 8238–8247.

[18] Frans A. Oliehoek, Stefan Witwicki, and Leslie P. Kaelbling. 2021. A Sufficient Statistic for Influence in Structured Multiagent Environments. *J. Artif. Intell. Res.* 70 (2021), 789–870.

[19] Luz Valerie Pascal, Marianne Akian, Sam Nicol, and Iadine Chades. 2021. A Universal 2-state n-action Adaptive Management Solver. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 14884–14892.

[20] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., Hoboken, NJ, USA. 672 pages.

[21] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. 2019. Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 5331–5340.

[22] Paulo Rauber, Avinash Ummadisingu, Filipe Mutz, and Jürgen Schmidhuber. 2019. Hindsight policy gradients. In *International Conference on Learning Representations*. OpenReview.net, 1–9. https://openreview.net/forum?id=Bkg2viA5FQ

[23] Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. 2008. Bayes-Adaptive POMDPs. In *Advances in Neural Information Processing Systems 20*. Curran Associates, Inc., 1225–1232.

[24] Nicholas Roy, Geoffrey J. Gordon, and Sebastian Thrun. 2005. Finding Approximate POMDP solutions Through Belief Compression. *J. Artif. Intell. Res.* 23 (2005), 1–40.

[25] David Silver and Joel Veness. 2010. Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems 23*. Curran Associates, Inc., 2164–2172.

[26] Jordi Smit, Canmanie T. Ponnambalam, Matthijs T.J. Spaan, and Frans A. Oliehoek. 2021. PEBL: Pessimistic Ensembles for Offline Deep Reinforcement Learning. Presented at the IJCAI Workshop on Robust and Reliable Autonomy in the Wild (R2AW).

[27] Alec Solway, Carlos Diuk, Natalia Córdova, Debbie Yee, Andrew G. Barto, Yael Niv, and Matthew M. Botvinick. 2014. Optimal Behavioral Hierarchy. *PLoS Computational Biology* 10, 8 (14 August 2014), 1–9. https://doi.org/10.1371/journal.pcbi.1003779

[28] Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. 2018. Meta Reinforcement Learning with Latent Variable Gaussian Processes. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI Press, 642–652.

[29] Christopher John Cornish Hellaby Watkins. 1989. *Learning From Delayed Rewards*. Ph.D. Dissertation. King's College, Cambridge, United Kingdom.

[30] Annie Xie, Dylan Losey, Ryan Tolsma, Chelsea Finn, and Dorsa Sadigh. 2021. Learning Latent Representations to Influence Multi-Agent Interaction. In *Proceedings of the 2020 Conference on Robot Learning*. PMLR, 575–588.

[31] Annie Xie, Dylan P. Losey, Ryan Tolsma, Chelsea Finn, and Dorsa Sadigh. 2020. Learning Latent Representations to Influence Multi-Agent Interaction. In *4th Conference on Robot Learning*.

[32] Jiayu Yao, Taylor Killian, George Konidaris, and Finale Doshi-Velez. 2018. Direct Policy Transfer via Hidden Parameter Markov Decision Processes. Presented at the ICML 2018 Workshop on Lifelong Learning: A Reinforcement Learning Approach.

[33] Kaiyu Zheng and Stefanie Tellex. 2020. pomdp_py: A Framework to Build and Solve POMDP Problems. *arXiv preprint arXiv:2004.10099*.