Dynamic Knowledge Injection for AIXI Agents

Samuel Yang-Zhao[1], **Kee Siong Ng**[1], **Marcus Hutter**[1, 2]

[1]Australian National University
[2]Google DeepMind
samuel.yang-zhao@anu.edu.au, keesiong.ng@anu.edu.au, www.hutter1.net

Abstract

Prior approximations of AIXI, a Bayesian optimality notion for general reinforcement learning, can only approximate AIXI's Bayesian environment model using an a-priori defined set of models. This is a fundamental source of epistemic uncertainty for the agent in settings where the existence of systematic bias in the predefined model class cannot be resolved by simply collecting more data from the environment. We address this issue in the context of Human-AI teaming by considering a setup where additional knowledge for the agent in the form of new candidate models arrives from a human operator in an online fashion. We introduce a new agent called DynamicHedgeAIXI that maintains an exact Bayesian mixture over dynamically changing sets of models via a time-adaptive prior constructed from a variant of the Hedge algorithm. The DynamicHedgeAIXI agent is the richest direct approximation of AIXI known to date and comes with good performance guarantees. Experimental results on epidemic control on contact networks validates the agent's practical utility.

Introduction

The AIXI agent (Hutter 2005) is a Bayesian solution to the general reinforcement learning problem that combines Solomonoff induction (Solomonoff 1997) with sequential decision theory. At time t, after observing the action-observation-reward sequence $h_{1:t-1} := aor_{1:t-1}$, the AIXI agent computes the action a_t to choose via an expectimax search up to a horizon H with a mixture model:

$$a_t = \arg\max_{a_t} \sum_{or_t} \max_{a_{t+1}} \sum_{or_{t+1}} \dots \max_{a_{t+H}} \sum_{or_{t+H}} \left[\sum_{j=t}^{t+H} r_j \right] \sum_{\rho \in M_U} 2^{-K(\rho)} \rho(or_{1:t+H} | a_{1:t+H}). \quad (1)$$

The Bayesian mixture $\sum_{\rho \in M_U} 2^{-K(\rho)} \rho(or_{1:t+H} | a_{1:t+H})$ is AIXI's environment model and is computed as a mixture over the set M_U of all enumerable chronological semi-measures with each element $\rho \in M_U$ assigned a prior according to its Kolmogorov complexity $K(\rho)$. Note that M_U is formally equivalent to the set of all computable distributions; from this perspective, AIXI can be considered the ultimate Bayesian reinforcement learning agent. Furthermore, (Hutter 2005) shows that AIXI's environment model converges rapidly to the true environment and its policy is pareto optimal and self-optimising.

The AIXI agent can be viewed as containing all possible knowledge as its Bayesian mixture is performed over all computable distributions. From this perspective, AIXI's performance does not suffer due to limitations in its modelling capacity. In contrast, all previous approximations of AIXI are limited to having a finite pre-defined model class containing a subset of computable probability distributions, presenting an irreducible source of error. To address this issue, we introduce *dynamic knowledge injection*, a setting where an external source is used to provide additional knowledge that is then integrated into new candidate environment models. In particular, dynamic knowledge injection models human-AI teaming constructs where the human can provide additional domain knowledge that the agent can use to model aspects of the environment. Once a new environment model is proposed, the central issue is then to determine how it can be incorporated to improve the agent's performance. Utilising a variation of the GrowingHedge algorithm (Mourtada and Maillard 2017), itself an extension of Hedge (Cesa-Bianchi and Lugosi 2006), we construct an adaptive *anytime* Bayesian mixture algorithm that incorporates newly arriving models and also allows the removal of existing models. DynamicHedgeAIXI is the richest direct approximation of AIXI to date and comes with strong value-convergence guarantees against the best available environment sequence. We validate the agent's performance empirically on multiple experimental domains, including the control of epidemics on large contact networks, and our results demonstrate that DynamicHedgeAIXI is able to quickly adapt to new knowledge that improves its performance.

Related Work

While AIXI is only asymptotically computable, it serves as a strong guiding principle in the design of general purpose AI agents. (Veness et al. 2011) gives the first tractable approximation of AIXI by using the Context Tree Weighting algorithm (CTW) (Willems, Shtarkov, and Tjalkens 1995) to restrict the Bayesian mixture to a set of variable-order

Markov environments and Monte-Carlo Tree Search to approximate the expectimax operation. This was followed by a body of work on extending the Bayesian mixture learning to larger classes of history-based models (Veness et al. 2012),(Veness et al. 2013), (Bellemare, Veness, and Bowling 2013), (Bellemare, Veness, and Talvitie 2014). The current best approximation of AIXI is given in (Yang-Zhao, Wang, and Ng 2022), which introduces the Φ-AIXI-CTW agent to extend AIXI's approximation to non-Markovian and structured environments. This is achieved using the Φ-BCTW data structure, a model that extends the CTW algorithm's representation capacity by combining state abstraction, motivated by the Feature Reinforcement Learning framework (Hutter 2009), with a rich logical formalism. One limitation of Φ-AIXI-CTW is that it models the environment using state abstractions that are fixed after performing feature selection at the start. The current work alleviates this issue by extending Φ-AIXI-CTW to the dynamic knowledge injection setting.

Background and Notation

General Reinforcement Learning

We consider finite action, observation and reward spaces denoted by $\mathcal{A}, \mathcal{O}, \mathcal{R}$ respectively. The agent interacts with the environment in cycles: at any time, the agent chooses an action from $\mathcal{A}$ and the environment returns an observation and reward from $\mathcal{O}$ and $\mathcal{R}$. We will denote a string $x_1 x_2 \dots x_n$ of length n by $x_{1:n}$ and its length $n-1$ prefix as $x_{<n}$. An action, observation and reward from the same time step will be denoted aor_t. A history h is an element of the history space $\mathcal{H} := (\mathcal{A} \times \mathcal{O} \times \mathcal{R})^* \cup (\mathcal{A} \times \mathcal{O})^*$. An environment ρ is a sequence of probability distributions $\{\rho_0, \rho_1, \rho_2, \dots\}$, where $\rho_n : \mathcal{A}^n \to \mathcal{D}((\mathcal{O} \times \mathcal{R})^n)$, that satisfies $\forall a_{1:n} \ \forall or_{<n} \ \ \rho_{n-1}(or_{<n}|a_{<n}) = \sum_{or \in \mathcal{O} \times \mathcal{R}} \rho_n(or_{1:n}|a_{1:n})$. We will drop the subscript on ρ_n when the context is clear. The predictive probability of the next percept given history and a current action is given by $\rho(or_n|aor_{<n}, a_n) = \rho(or_n|h_{n-1}, a_n) := \frac{\rho(or_{1:n}|a_{1:n})}{\rho(or_{<n}|a_{<n})}$ for all $aor_{1:n}$ such that $\rho(or_{<n}|a_{<n}) > 0$.

The general reinforcement learning problem (GRL) is for the agent to learn a *policy* $\pi : \mathcal{H} \to \mathcal{D}(A)$ mapping histories to a distribution on possible actions that will allow it to maximise its future expected reward. In this paper, we consider the future expected reward up to a finite horizon $H \in \mathbb{N}$. Given the history $h_{<t} = aor_{<t}$ up to time t and a policy π the value function with respect to the environment ρ is given by $V_\rho^\pi(h_{<t}) := \mathbb{E}_\rho^\pi \left[\sum_{i=t}^{t+H} r_i \;\middle|\; h_{<t} \right]$, where r_i denotes the random variable distributed according to ρ for the reward at time i. The action value function is defined similarly as $Q_\rho^\pi(h_{<t}, a_t) := \mathbb{E}_\rho^\pi \left[\sum_{i=t}^{t+H} r_i \;\middle|\; h_{<t}, a_t \right]$. The agent's goal is to learn the optimal policy π^*, which is the policy that results in the value function with the maximum reward for any given history.

Abstract Environment Models

In the dynamic knowledge injection setting, the candidate models received by the agent may only approximate the underlying environment. State abstractions provides a framework for defining such models. A state abstraction is a mapping $\phi : \mathcal{H} \to \mathcal{S}_\phi$ that maps the space of history sequences into an abstract state space. For the history $h_{1:t}$ at time t, the state at time t is given by $s_t = \phi(h_{1:t})$. In this manner, the interaction sequence of the original process is mapped to a state-action-reward sequence. For a given ϕ, an abstract Markov Decision Process (MDP) predicts the next state and reward according to a distribution $\rho_\phi : \mathcal{S}_\phi \times \mathcal{A} \to \mathcal{D}(\mathcal{S}_\phi \times \mathcal{R})$ that factorises into a state transition and reward distribution as $\rho_\phi(s', r|s, a) = \rho_\phi(s'|s, a)\rho_\phi(r|s, a, s')$. Let $\rho_\phi := (\rho_{t,\phi})_{t \geq 1}$, where $\rho_{t,\phi} : \mathcal{S}_\phi \times \mathcal{A} \to \mathcal{D}(\mathcal{S}_\phi \times \mathcal{R})$ for all t. We refer to (ϕ, ρ_ϕ) together as an abstract environment model. Note that the state of an abstract MDP model will in general not be a sufficient statistic for the underlying environment at hand and thus presents a source of bias; a simple example is when ϕ maps all histories into a single state.

An abstract MDP can help simplify the environment's dynamics but pushes a lot of the complexity into the design of the state abstraction function and a sufficiently powerful representation is required to ensure as little generality is lost. In particular, the quality of an abstract environment model will determine how closely its reward distribution approximates the underlying environment's reward distribution. Following (Yang-Zhao, Wang, and Ng 2022), we consider the class of predicate environment models, which are abstract environment models that can be constructed from a set of predicate functions on histories. More precisely, we consider abstract environment models where the state abstraction is of the form $\phi(h) = (p_1(h), \dots, p_n(h))$ and $p_i : \mathcal{H} \to \{0, 1\}$ are predicates. Under a sufficiently powerful knowledge representation and reasoning language (such as the one described in (Lloyd and Ng 2011; Lloyd 2003; Farmer 2008)), such models are capable of representing a large class of non-Markovian and structured environments as abstract MDPs. The Φ-BCTW data structure, described in the next section, will be used to model the distributions under the predicate state abstractions.

Bayesian Mixtures and Φ-BCTW

The importance of Bayesian mixture models in general reinforcement learning is that they converge rapidly to the 'good' models in the model class.

Theorem 1. *(Hutter 2005) Let μ be the true environment and ξ be the mixture environment model over a model class $\mathcal{M}$. Let $x_k = or_k$ and $h_{<k} = aor_{<k}$. For all $n \in \mathbb{N}$ and for all $a_{1:n}$,*

$$\sum_{k=1}^{n} \sum_{x_{1:k}} \mu(x_{<k}|a_{<k}) \left(\mu(x_k|h_{<k}a_k) - \xi(x_k|h_{<k}a_k)\right)^2 \leq \min_{\rho \in \mathcal{M}} \left\{ \ln \frac{1}{w_0^\rho} + KL(\mu||\rho) \right\} \quad (2)$$

Context Tree Weighting (CTW) (Willems, Shtarkov, and Tjalkens 1995) is a rare case where a Bayesian mixture over prediction suffix trees can be computed efficiently. The Φ-BCTW data structure introduced in (Yang-Zhao, Wang, and Ng 2022) generalises CTW by allowing predicates $p_i : \mathcal{H} \to$

$\{0, 1\}$ from a set Φ to act as the internal nodes of the tree. In more details, in the Φ-BCTW tree, each sub-tree of depth d is a Φ-prediction suffix tree (Φ-PST) model. For a history h, a Φ-PST with predicates p_i at depth i computes a path from root to leaf node as $p_1(h)p_2(h)\dots p_d(h)$, which forms a state abstraction. At each leaf node resides a KT estimator (Krichevsky and Trofimov 2006) maintaining a distribution over the next bit. By chaining together multiple Φ-BCTW trees, the data structure can be used to predict the binary representation of arbitrary symbols.

A Φ-BCTW data structure constructed using a set Φ of D predicates is able to perform an exact Bayesian mixture over $2^{(2^D)}$ Φ-PST models in $\mathcal{O}(D)$ time. Since each Φ-PST model represents a different predicate environment model, applying Φ-BCTW to predicting state and reward symbols then amounts to computing a mixture environment model over all predicate environment models that can be constructed from Φ. The following result states the environment mixture result for Φ-BCTW.

Proposition 1 ((Yang-Zhao, Wang, and Ng 2022))**.** Suppose a state and reward symbol can be binarized in k bits. Let $\mathcal{T}_d$ be the set of Φ-PST models that can be constructed from a single Φ-BCTW tree of depth d. Let $P(\cdot|a_{1:n}, T)$ denote the action-conditional distribution under model T. Then the Φ-BCTW computes a mixture environment model of the form, where $\Gamma(T)$ is the coding length of T:

$$\xi(sr_{1:n}|a_{1:n}) = \sum_{T \in \mathcal{T}_d \times \dots \times \mathcal{T}_{d+k-1}} 2^{-\Gamma(T)} P(sr_{1:n}|a_{1:n}, T)\ . \tag{3}$$

Prediction with Expert Advice

The prediction with expert advice setting is a well-established framework providing theoretically sound strategies on how to aggregate the forecasts provided by many experts in a sequential setting (Cesa-Bianchi and Lugosi 2006). This setting is characterised by a game played between a learner and an adversary. Initially, a loss function $\ell : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is provided where $\mathcal{X}$ is the vector space of predictions and $\mathcal{Y}$ is the outcome space. The learner has access to a set of fixed experts M. At time t, a learner receives prediction $x_{t,i} \in \mathcal{X}$ from expert i. The learner then must combine the predictions from all experts and outputs $x_t \in \mathcal{X}$. An adversary then chooses an outcome $y_t \in \mathcal{Y}$ causing the learner to incur loss $\ell_t = \ell(x_t, y_t)$ and observe the loss $\ell_{t,i} = \ell(x_{t,i}, y_t)$ for each expert i. Learners are typically designed to minimise the *regret* $L_T - L_{T,i} = \sum_{t=1}^{T} \ell_t - \sum_{t=1}^{T} \ell_{t,i}$, a measure of the relative performance of the agent with respect to any fixed expert $i \in M$. The *Hedge* (exponential weights) algorithm is a simple yet fundamental algorithm in this setting (Cesa-Bianchi and Lugosi 2006; Vovk 1998). Given a prior distribution $\boldsymbol{\nu}$ over M and $\eta > 0$, Hedge predicts

$$x_t = \frac{\sum_{i \in M} w_{t,i} x_{t,i}}{\sum_{i \in M} w_{t,i}}$$

where $w_{t,i} = \nu_i e^{-\eta L_{t-1,i}}$. The weights of the Hedge algorithm can be viewed as the posterior probabilities of each expert (Jordan 1995). The following is a standard regret bound for the Hedge algorithm.

Proposition 2 ((Cesa-Bianchi and Lugosi 2006))**.** If the loss function ℓ is η-exp-concave, then for any $i \in M$, Hedge with prior $\boldsymbol{\nu}$ has regret bound $L_T - L_{T,i} \leq \frac{1}{\eta} \log \frac{1}{\nu_i}$.

Dynamic Knowledge Injection

In this section we formalise the Dynamic Knowledge Injection setting. We first present an extension of the prediction with expert advice setting known as the *specialists* setting. The Dynamic Knowledge Injection setting can then be naturally described using the specialists framework.

The Specialists Setting

Incorporating expert advice from novel experts arriving in an online fashion can be cast into the specialists setting (Freund et al. 1997). The specialist setting extends the prediction with expert advice setting by introducing specialists: experts that can abstain from prediction at any given time step. In this setting, the learner has access to a set M of specialists where at time t, only specialists in a subset $M_t \subseteq M$ output predictions $x_{t,i} \in \mathcal{X}$. The crucial idea to adapt the Hedge algorithm to this setting was presented in (Chernov and Vovk 2009) where inactive specialists $j \notin M_t$ are attributed a forecast equal to that of the learner. More precisely, choosing $x_{t,j} = \frac{\sum_{i \in M_t} w_{t,i} x_{t,i}}{\sum_{i \in M_t} w_{t,i}}$ for $j \notin M_t$ results in

$$x_t = \frac{\sum_{i \in M} w_{t,i} x_{t,i}}{\sum_{i \in M} w_{t,i}} = \frac{\sum_{i \in M_t} w_{t,i} x_{t,i}}{\sum_{i \in M_t} w_{t,i}}.$$

The *specialist aggregation* algorithm uses the above 'abstention' trick where the weight for expert i at time t is given by $w_{t,i} = \nu_i e^{-\eta L_{t-1,i}}$ and $L_{t,i} := \sum_{s \leq t : i \in M_s} \ell_{s,i} + \sum_{s \leq t : i \notin M_s} \ell_s$. This abstention trick helps maintain enough weight on abstaining experts to ensure the regret is well controlled. We use this fundamental technique to incorporate newly arriving models for Bayesian agents.

The Dynamic Knowledge Injection Setting

The Dynamic Knowledge Injection setting can now be naturally defined using the Specialists framework. In this setting, a specialist i is an abstract MDP of the form (ϕ_i, ρ_i) where ϕ_i is a state abstraction function constructed from a set of predicate functions and $\rho_i = (\rho_{t,i})_{t \geq 1}$. As an abstract MDP, $(\rho_{t,i})_{t \geq 1}$ is a sequence of distributions where $\rho_{t,i} : \mathcal{S}_i \times \mathcal{A} \to \mathcal{D}(\mathcal{S}_i \times \mathcal{R})$. Modelling the environment as a sequence allows us to naturally represent environment models that update and learn their distributions online, such as Φ-BCTW. Over the course of the agent's lifetime, it is given new abstract environment models by a human operator at intermittent time steps. A key example of this setting is when the initial models our agent starts with are inadequate and a separate training process is able to submit new and improved models over time.

Algorithm 1: DynamicHedge (modifies GrowingHedge (Mourtada and Maillard 2017))

1: **Require:** Learning rate $\eta > 0$, weights $\boldsymbol{\nu} = (\nu_i)_{i \geq 1}$, sequence of sets of contiguous specialists $(M_t)_{t \geq 1}$.
2: **Initialize:** $L_0 = 0$. For $i \in M_1$, set $w_{1,i} = \nu_i$.
3: **for** $t = 1, 2, \ldots, T$ **do**
4: For all $i \in M_t$, receive prediction $x_{t,i} \in \mathcal{X}$.
5: Predict $x_t = \frac{\sum_{i \in M_t} w_{t,i} x_{t,i}}{\sum_{i \in M_t} w_{t,i}}$.
6: Observe $y_t \in \mathcal{Y}$.
7: Set $\ell_t = \ell(x_t, y_t)$ and $\ell_{t,i} = \ell(x_{t,i}, y_t)$.
8: Set $L_t = L_{t-1} + \ell_t$.
9: For $i \in M_t \cap M_{t+1}$, set $w_{t+1,i} = w_{t,i} e^{-\eta \ell_{t,i}}$.
10: For $i \in M_{t+1} \setminus M_t$ set $w_{t+1,i} = \nu_i e^{-\eta L_t}$.
11: **end for**

Incorporating New Models

The key issue for the agent is to determine how the newly arriving models can be integrated in an online fashion. The GrowingHedge algorithm (Mourtada and Maillard 2017) applies to the setting where the set of active experts grows over time and achieves the same regret bound as specialist aggregation. In practice, the growing experts setting is infeasible as computational constraints dictate that the set of available experts cannot grow unboundedly over time. Instead, we consider the setting where arriving experts are *contiguous specialists* that can only be active for contiguous periods of time before becoming inactive forever. This captures the situation in practice whereby the set of models can grow until resource limits are reached and a newly entering model must then replace an older model. Formally, over T steps the set $T_i = \{t \in [T] : i \in M_t\}$ for any contiguous specialist i is a contiguous set of integers. Let $\tau_i = \min(T_i)$ and $\kappa_i = \max(T_i)$, representing the arrival and 'death' times for specialist i.

Under this restriction we present the DynamicHedge algorithm (Algorithm 1). DynamicHedge modifies GrowingHedge's weight update to account for contiguous specialists; Algorithm 1 line 9 removes contiguous specialists that are no longer active. Like GrowingHedge, DynamicHedge can use an unnormalized prior and does not require knowledge of the entire set of experts a-priori.

DynamicHedgeAIXI Agent

Our technique for incorporating Dynamic Knowledge Injection, DynamicHedge, can now be naturally integrated into a reinforcement learning agent. The DynamicHedgeAIXI agent is presented in Algorithm 2. We consider the case where specialists are abstract MDPs. Each specialist $i \in M_t$ produces a function $Q_i^{\pi_i} : \mathcal{H} \times \mathcal{A} \to \mathbb{R}$ denoting the expected utility of action a_t under a policy $\pi_i : \mathcal{S}_i \to \mathcal{A}$ up to a horizon H:

Algorithm 2: DynamicHedgeAIXI

1: **Require:** Environment μ, initial history h_0, learning rate $\eta > 0$, weights $\boldsymbol{\nu} = (\nu_i)_{i \geq 1}$,
2: **Require:** Sequence of sets of contiguous (abstract MDP) specialists $(M_t)_{t \geq 1}$,
3: **Require:** Sequence of composite policies $\boldsymbol{\pi} = (\boldsymbol{\pi_t})_{t \geq 1}$, where $\boldsymbol{\pi_t} = (\pi_i)_{i \in M_t}$ and $\pi_i : \mathcal{S}_i \to \mathcal{A}$.
4: **Initialize:** $L_0 = 0$. For $i \in M_1$, set $w_{1,i} = \nu_i$.
5: **for** $t = 1, 2, \ldots, T$ **do**
6: Set $\hat{w}_{t,i} = \frac{w_{t,i}}{\sum_{j \in M_t} w_{t,j}}$.
7: Select $a_t = \arg\max_a \sum_{i \in M_t} \hat{w}_{t,i} Q_i^{\pi_i}(h_{<t}, a)$.
8: Observe $o_t, r_t \sim \mu(\cdot | h_{<t}, a_t)$.
9: For all $i \in M_t$, $s_t^i = \phi_i(h_{<t} a o r_t)$.
10: For all $i \in M_t$, $x_{t,i} = \rho_{t,i}(\cdot | s_{t-1}^i, a_t, s_t^i)$ where $\rho_{t,i} : \mathcal{S}_i \times \mathcal{A} \times \mathcal{S}_i \to \mathcal{D}(\mathcal{R})$.
11: Set $x_t = \frac{\sum_{i \in M_t} w_{t,i} x_{t,i}}{\sum_{i \in M_t} w_{t,i}}$
12: Set $\ell_t = -\log x_t(r_t)$, $\ell_{t,i} = -\log x_{t,i}(r_t)$.
13: Set $L_t = L_{t-1} + \ell_t$.
14: For $i \in M_t \cap M_{t+1}$, set $w_{t+1,i} = w_{t,i} e^{-\eta \ell_{t,i}}$.
15: For $i \in M_{t+1} \setminus M_t$, set $w_{t+1,i} = \nu_i e^{-\eta L_t}$.
16: **end for**

$$Q_i^{\pi_i}(h_{<t}, a_t) = \sum_{sr_{t:t+H}} \left[\sum_{j=t}^{t+H} r_j \right] \rho_i(sr_{t:t+H} | h_{<t}, a_{t:t+H}^i). \tag{4}$$

Also $a_t^i = a_t$ and for $k = 1, \ldots, H$ the actions are selected via $a_{t+k}^i = \pi_i(s_{t+k}^i)$. The agent then selects the action that maximises the weighted sum of the given Q values.

DynamicHedgeAIXI tracks existing and newly entering specialists by computing the weights $w_{t,i}$ using DynamicHedge. At each time step, specialist i predicts a state-action conditional distribution over the next reward and is evaluated based on the log loss $\ell_{t,i} = -\log \rho_{t,i}(r_t | s_{t-1}^i, a_t, s_t^i)$. Thus, over time DynamicHedgeAIXI will weight specialists based on how well they predict the reward sequence over time. In this manner, DynamicHedgeAIXI can also avoid the objective mismatch issue common to other system identification approaches to model-based reinforcement learning (Lambert et al. 2021; Eysenbach et al. 2023).

Properties

Expanding line 7, Algorithm 2 with Equation 4 shows that actions are selected according to:

$$a_t = \arg\max_{a_t} \left(\sum_{sr_{t:t+H}} \left[\sum_{j=t}^{t+H} r_j \right] \sum_{i \in M_t} \hat{w}_{t,i} \, \rho_i(sr_{t:t+H} | h_{<t}, a_{t:t+H}^i) \right) \tag{5}$$

where $\hat{w}_{t,i} = \frac{w_{t,i}}{\sum_{j \in M_t} w_{t,j}}$. Let $\boldsymbol{\pi_t} = (\pi_i)_{i \in M_t}$ denote the composite policy consisting of the policy over each specialist $i \in M_t$. DynamicHedgeAIXI's mixture environment model can then be defined as

$$\xi^{\boldsymbol{\pi_t}}(r_{t:t+H}|h_{<t}) = \sum_{i \in M_t} \hat{w}_{t,i}\, \rho_i(r_{t:t+H}|h_{<t}, a^i_{t:t+H}),$$

where

$$\rho_i(r_{t:t+H}|h_{<t}, a^i_{t:t+H}) = \sum_{s_{t:t+H}} \rho_i(sr_{t:t+H}|h_t, a^i_{t:t+H}).$$

Expanding $\hat{w}_{t,i} = \frac{w_{t,i}}{\sum_{j \in M_t} w_{t,j}}$ gives

$$\xi^{\boldsymbol{\pi_t}}(r_{t:t+H}|h_{<t}) = \sum_{i \in M_t} w^i_t\, \rho_i(r_{\tau_i:t-1}|sa^i_{\tau_i:t-1})\rho_i(r_{t:t+H}|h_{<t}, a^i_{t:t+H}), \quad (6)$$

where $w^i_t = \frac{\nu_i e^{-L_{\tau_i - 1}}}{\sum_{j \in M_t} \nu_j e^{-L_{\tau_j - 1}} \rho_j(r_{\tau_j:t-1}|sa^j_{\tau_j:t-1})}$ and $\phi_i(h_j) = s^i_j$ for $j < t$.

Equation 6 reveals that w^i_t can be viewed as the prior weight given to each model in the mixture and that DynamicHedgeAIXI computes an exact Bayesian mixture over the available set of models at each time step. When a model i first becomes active, its initial weight $w^i_{\tau_i}$ in the mixture environment model depends upon the relative performance of DynamicHedge to each of the available models before time τ_i. In this sense, the prior is *adaptive* as it is path dependent.

Value Convergence

Our main theoretical result shows that DynamicHedgeAIXI will achieve good value convergence rates against the best sequence of environment models available to the agent. Since each specialist can potentially operate over a different state space, we first convert the state-reward distribution into a representative observation-reward distribution for our analysis. For a specialist (ϕ, ρ), let $\bar{\rho}$ be an environment distribution such that the following hold:

$$\sum_{o_t \in \mathcal{O}: \phi(h_{<t} a o_t) = s_t} \bar{\rho}(o_t|h_{<t}, a_t) = \rho(s_t|s_{t-1}, a_t),$$
$$\text{and} \quad \bar{\rho}(r_t|h_{<t}, a_t, o_t) = \rho(r_t|s_{t-1}, a_t, s_t),$$

where $\phi(h_{<t}) = s_{t-1}$ and $\phi(hao) = s'$. The specialist can then be identified as $(\phi, \bar{\rho})$ and we will drop the bar on $\bar{\rho}$ when the context is clear.

Most standard Bayesian consistency results compare performance in the realizable setting, where it is assumed that the model class contains the true environment. To generalise to the dynamic knowledge injection setting where the model class dynamically changes, we instead compare performance against an *admissible* environment sequence.

Definition 1 (Admissible environment sequence)**.** Let $(M_t)_{t \geq 1}$ be the sequence of sets of specialists. Let $\mu = (\mu_t)_{t \geq 1}$ be an environment sequence such that for all $t \geq 1$ $\mu_t \in M_t$, i.e. there exists $i \in M_t$ where $\rho_i = \mu_t$. We say μ is admissible if for $j \geq 1$, if $\mu_j \neq \mu_{j+1}$, then $\mu_{j+1} \in M_{j+1} \setminus M_j$.

Denote by $V^H_i(h_{<t}, \pi_i) = \mathbb{E}^{\pi_i}_{\rho_i}\left[\sum_{j=t}^{t+H} r_j \;\middle|\; h_{<t}\right]$ the expected future value whilst actions are selected according to π_i with specialist i. Also, let $V^H_\xi(h_{<t}, \boldsymbol{\pi_t}) = \sum_{i \in M_t} \hat{w}_{t,i} V^H_i(h_{<t}, \pi_i)$ denote the expected future value for DynamicHedgeAIXI. To simplify analysis, we consider when $\eta = 1$ and $\boldsymbol{\nu} = \mathbf{1}$.

Theorem 2. *Let $\mu = (\mu_i)_{1 \leq i \leq T}$ be an admissible sequence of environments. Let $\sigma = (\sigma_i)_{1 \leq i \leq k}$ be the switching times such that for all $1 \leq i < k$, $\sigma_i < \sigma_{i+1}$ and for all $1 \leq j \leq T$, $\mu_{j-1} \neq \mu_j$ iff $j \in \sigma$. Let $\bar{M}_t = \bigcup_{0 \leq s \leq t} M_s$. Then for any sequence of composite policies $\boldsymbol{\pi} = (\boldsymbol{\pi_t})_{1 \leq t \leq T}$ with $\pi_{\mu_t} \in \boldsymbol{\pi_t}$,*

$$\sum_{t=1}^{T} \mathbb{E}_{h_{<t} \sim \mu}\left[\left(V^H_\xi(h_{<t}, \boldsymbol{\pi_t}) - V^H_{\mu_t}(h_{<t}, \pi_{\mu_t})\right)^2\right] \leq 2H^3 r^2_{\max} \log \frac{1}{w(\mu)}, \quad (7)$$

where $w(\mu) = \prod_{j=0}^{k} \hat{w}^j_{\tau_j}$ and $\hat{w}^j_{\tau_j} = \frac{w_{\tau_j, j}}{\sum_{i \in \bar{M}_T} w_{\tau_j, i}}$.

For a fixed horizon H, Theorem 2 shows that the cumulative squared difference of the value under DynamicHedgeAIXI is bound as a function of $\log \frac{1}{w(\mu)}$. The term $w(\mu)$ can be viewed as the prior weight assigned to the sequence μ by DynamicHedgeAIXI. In the context of AIXI agents, $w(\mu)$ is a measure of μ's complexity, with smaller values for $w(\mu)$ indicating a more complex sequence. The complexity of $w(\mu)$ is a function of how often μ switches and the performance of the models available to the agent. The dependence on the number of switches and available models is given next.

Theorem 3. *Let $\mu = (\mu_i)_{1 \leq i \leq T}$ be an admissible sequence of environments. Let $\sigma = (\sigma_i)_{1 \leq i \leq k}$ be the switching times such that for all $1 \leq i < k$, $\sigma_i < \sigma_{i+1}$ and for all $1 \leq j \leq T$, $\mu_{j-1} \neq \mu_j$ iff $j \in \sigma$. Let $\bar{M}_t = \bigcup_{0 \leq s \leq t} M_s$. Then for any sequence of composite policies $\boldsymbol{\pi} = (\boldsymbol{\pi_t})_{1 \leq t \leq T}$ with $\pi_{\mu_t} \in \boldsymbol{\pi_t}$,*

$$\sum_{t=1}^{T} \mathbb{E}_{h_{<t} \sim \mu}\left[\left(V^H_\xi(h_{<t}, \boldsymbol{\pi_t}) - V^H_{\mu_t}(h_{<t}, \pi_{\mu_t})\right)^2\right] \leq 4kH^3 r^2_{\max} \log|\bar{M}_T|. \quad (8)$$

Theorem 3 makes clear that the cumulative error grows at the rate $\mathcal{O}\left(k \log|\bar{M}_T|\right)$. If the number of switches k grows sub-linearly and the total number of models seen by the agent over T steps $|\bar{M}_T|$ grows sub-exponentially in T, then DynamicHedgeAIXI's value will converge.

Proof Sketch

For brevity, we only provide a proof sketch for Theorem 2 as Theorem 3 follows directly from Theorem 2. The full proofs of Theorems 2 and 3 are provided in the extended version of the paper (Yang-Zhao, Ng, and Hutter 2023).

Let $\bar{M}_T = \bigcup_{0 \leq s \leq T} M_s$ denote the set of all specialists seen by DynamicHedgeAIXI up to time T and let $\boldsymbol{\pi} =$

$(\pi_i)_{i\in\bar{M}_T}$. For each specialist $i \in \bar{M}_T$ we define

$$\hat{\rho}_i^{\pi_i}(or_{t:t+H}|h_{<t}) := \begin{cases} \rho_i^{\pi_i}(or_{t:t+H}|h_{<t}) & \text{if } i \in M_t \\ \sum_{j\in M_t} \hat{w}_{t,j}\rho_j^{\pi_j}(or_{t:t+H}|h_{<t}) & \text{if } i \notin M_t \end{cases}$$

where $\rho_i^{\pi_i}(or_{t:t+H}|h_{<t}) = \rho_i(or_{t:t+H}|h_{<t}, a^i_{t:t+H})$ with $a^i_{t+k} = \pi_i(s^i_{t+k})$ and $s^i_{t+k} = \phi(h_{1:t+k})$ for $k = 0, \ldots, H$. Using the abstention trick, DynamicHedgeAIXI's value function can then be expressed as follows:

$$V_\xi^H(h_{<t}, \boldsymbol{\pi}) = \sum_{or_{t:t+H}} \left[\sum_{j=t}^{t+H} r_j\right] \sum_{j\in\bar{M}_T} \hat{w}_{t,j}\hat{\rho}_j^{\pi_j}(or_{t:t+H}|h_{<t}).$$

We define the mixture model as:

$$\xi_t^{\boldsymbol{\pi}}(or_{t:t+H}|h_{<t}) = \sum_{j\in\bar{M}_T} \hat{w}_{t,j}\hat{\rho}_j^{\pi_j}(or_{t:t+H}|h_{<t}).$$

After expressing the value function in this way, we are ready to bound the error. The sum over T time steps can be first split into a double sum over segments where μ does not change distribution:

$$\sum_{t=1}^{T} \mathbb{E}\left[\left(V_\xi^H(h_{<t}, \boldsymbol{\pi}) - V_{\mu_t}^H(h_{<t}, \pi_{\mu_t})\right)^2\right] = \sum_{i=0}^{k}\sum_{t=\sigma_i}^{\sigma_i+1} \mathbb{E}\left[\left(V_\xi^H(h_{<t}, \boldsymbol{\pi}) - V_{\rho_i}^H(h_{<t}, \pi_i)\right)^2\right],$$

where ρ_i is such that $\rho_i = \mu_t$ for $t = \sigma_i, \ldots, \sigma_{i+1} - 1$. We then look to bound the error in each segment. We first convert the squared error between the value functions into a squared error between the mixture distribution and the underlying distribution before applying Pinsker's inequality:

$$\begin{aligned}
&\sum_{t=\sigma_i}^{\sigma_i+1} \mathbb{E}\left[\left(V_\xi^H(h_{<t}, \boldsymbol{\pi}) - V_{\mu_t}^H(h_{<t}, \pi_{\mu_t})\right)^2\right] \\
&\quad\leq H^2 r_{\max}^2 \sum_{t=\sigma_i}^{\sigma_{i+1}-1} \mathbb{E}\left[D_{t:t+H}(\mu_i||\xi_t^{\boldsymbol{\pi}})\right] \\
&\quad= H^2 r_{\max}^2 \sum_{t=\sigma_i}^{\sigma_{i+1}-1}\sum_{n=t}^{t+H} \mathbb{E}\left[D_{n:n}(\mu_i||\xi_t^{\boldsymbol{\pi}})\right]
\end{aligned}$$

where $D_{i:j}(\mu||\rho) := \sum_{or_{i:j}} \mu(or_{i:j}|h_{<i}) \log \frac{\mu(or_{i:j}|h_{<i})}{\rho(or_{i:j}|h_{<i})}$ denotes the KL divergence on the sequence from time i to j. The standard argument is to then apply the chain rule for KL divergence to collapse the double sum into a single KL divergence term. This cannot be done naively in our case however as the mixture model $\xi_t^{\boldsymbol{\pi}}$ uses a different set of weights at each time step. We are however able to apply the chain rule in the standard manner by noticing that the weights that maximize each KL divergence term occur at the start of each segment. This allows us to recover an upper bound of $-\ln \hat{w}_{\sigma_i,i}$ per segment. Summing over all segments gives the final result.

DynamicHedgeAIXI in Practice

In practice, we let $\eta = 1$, $\boldsymbol{\nu} = \mathbf{1}$ and instantiate DynamicHedgeAIXI in the case where each specialist is a Φ-BCTW model. In this case, each specialist is itself a mixture environment model over a set of Φ-PST environment models. Whilst DynamicHedge itself uses an agnostic prior, using Φ-BCTW models as specialists means that DynamicHedgeAIXI's mixture environment model incorporates a model complexity prior:

$$\xi^{\pi_t}(r_{t:t+H}|h_{<t}) = \sum_{i\in M_t}\sum_{T\in\mathcal{T}_i} w_t^i\, 2^{-\Gamma(T)} P(r_{\tau_i:t-1}|sa_{\tau_i:t-1}, T)P(r_{t:t+H}|s_{t-1}, a_{t:t+H}, T)\,, \quad (9)$$

where $\mathcal{T}_i$ is the set of Φ-PST models in the Φ-BCTW specialist i. The Q value functions for each specialist (Algorithm 2, line 4) also need to be estimated in practice. Each specialist uses the UCT policy (Kocsis and Szepesvári 2006) and Monte-Carlo Tree Search to approximate the finite-horizon expectimax operation in calculating Q. We show in our experiments that this set of design choices works well in practice.

Experiments

Experiment Setup

In the Dynamic Knowledge Injection setting, new knowledge arrives from a human operator in the form of new domain-specific predicates. A predicate environment model is then generated from these predicates for the agent to utilise. To display the adaptive behaviour of our agent, we consider the setting where better models are generated for the agent over time. We simulate the dynamic knowledge injection setting by maintaining two sets of predicates I and U representing the informative and uninformative predicates for each domain. Given $p \in [0, 1]$, a new Φ-BCTW model of depth d is constructed by sampling $\lfloor p \cdot d \rfloor$ predicates from I and $d - \lfloor p \cdot d \rfloor$ predicates from U. The proportion p initially starts out small and increases over time.

In all our experiments, we drop the Φ-BCTW model with the lowest weight and introduce a new Φ-BCTW model every 4K steps. The model is also pre-trained on the preceding 4K steps to ensure it does not perform too poorly to when it is first introduced. The parameter p starts out at $p = 0.05$ and increases by 0.05 every 4K steps. The full details of the experiment design are provided in the extended version of the paper (Yang-Zhao, Ng, and Hutter 2023).

Baseline Methods. We compare DynamicHedgeAIXI against three baseline methods. We compare against two decision-tree based, iterative state abstraction methods using splitting criteria as defined in U-Tree (McCallum 1996) and PARSS-DT (Hostetler, Fern, and Dietterich 2017). These two models were chosen as they are able to be modified to use predicate functions as state abstraction criteria appropriately. In U-Tree, an existing node (representing a state) is split on a given predicate if splitting results in a statistically significant difference in the resulting Q-values as computed by a Kolmogorov-Smirnov test. In PARSS-DT, nodes are split on a given predicate if the resulting value functions

are sufficiently far apart. To modify each method to the dynamic knowledge injection setting, each method can only consider splits from the currently available set of informative predicates as well as the uninformative predicates. This also ensures that the baseline methods do not have an informational advantage over DynamicHedgeAIXI. The third method we compare against is a non-adaptive version of DynamicHedgeAIXI we call HedgeAIXI. HedgeAIXI proceeds in the same way as DynamicHedgeAIXI but does not re-initialize a model's weight when a newly entering model replaces an older model.

Domains

Biased Rock-Paper-Scissors (RPS). This domain is taken from (Farias et al. 2010). In biased RPS, the agent plays RPS against an environment with a biased strategy. The environment plays randomly but plays rock if it won the previous round playing rock. There are two informative predicates in this setting: $IsRock_{t-1}(h)$ and $IsLose_{t-1}(h)$, indicating whether the environment played rock and whether the agent lost in the previous time step respectively.

Taxi. The Taxi environment was first introduced in (Dietterich 2000). The agent acts as a Taxi in a grid world and must move to pick up a passenger and drop the passenger off at their desired destination. Instead of the 5x5 grid traditionally considered, we consider a 2x5 grid with no intermediate walls and increase the reward for completing the task. These modifications were made to shrink the planning horizon as well as make the original problem less sensitive to parameters for the three algorithms tested. The predicates available to the agent are indicator functions on the binary representation of the history indicating whether a given bit equals 1. The agent can recover the original MDP if it captures the indicator functions comprising the last received observation.

Epidemic Control Over Contact Networks. The epidemic control problem we use was introduced in (Yang-Zhao, Wang, and Ng 2022). In this environment, an SEIRS epidemic process evolves over a contact network and the agent's goal is to perform actions to slow or stop the spread of the disease (Pastor-Satorras et al. 2015; Nowzari, Preciado, and Pappas 2016; Newman 2018). A contact network is an undirected graph where the nodes represent individuals and the edges represent interactions between individuals. We use the network dataset from (Rossi and Ahmed 2015; Guimerà et al. 2003), which contains 1133 nodes and 5451 edges. Each node is in one of four states corresponding to their infection status: Susceptible (S), Exposed (E), Infectious (I), and Recovered (R). Each node also maintains an immunity level. The environment is partially observable and the environment emits an observation on each node from $\{+, -, ?\}$ corresponding to whether a node tests positive, negative or is unknown/untested. The agent can select actions from a set of 11 possible actions $\{DoNothing, Vaccinate(i, j), Quarantine(i)\}$, where $i \in [0, 0.2, 0.4, 0.6, 0.8, 1.0]$ and $j = i + 0.2$. Quarantine actions quarantine the top ith percent of nodes ranked by betweenness centrality by removing all edges incident on those nodes for one time step. Vaccinate actions increase the immunity level (up to a maximum value) for the top ith percent of nodes ranked in the same way. At each time step, the instantaneous reward is given by

$$r_t(o_t, a_{t-1}) \coloneqq -Positives(o_t) - Action\_Cost(a_{t-1})$$

where $Positives(o_t)$ counts the number of positive tests in the observation o_t and $Action\_Cost(a_{t-1})$ is a function determining the cost of each action. If the agent successfully terminates the epidemic, i.e. there are no more Exposed or Infectious nodes, the agent receives a positive reward of 2 per node. A full description of the transition and observation models is provided in the extended version of the paper (Yang-Zhao, Ng, and Hutter 2023).

The exact predicates that are considered perfectly useful in this domain are unknown. However, a few are known to provide some information in helping the agent represent the problem. Some examples of the types of functions the agent gains access to over time are as follows:

- $ObservedInfectionRate_{t,\nu}$ takes a history sequence $h \in \mathcal{H}$ and computes the observed infection rate at time t over the set of nodes $\nu \subseteq V$. Predicates comparing whether the observed infection rate is greater than or smaller than certain values are received over time.
- $InfectionRateOfChange_{t,\nu}$ takes $h \in \mathcal{H}$ and computes the change in infection rate between timesteps $t-1$ and t over the set of nodes $\nu \subseteq V$. Predicates comparing whether the infection rate of change is greater than or smaller than certain values are received over time.
- $PercentAction_{a,N}$ takes a history and returns the percentage of time action a was selected in the last N timesteps. Predicates comparing whether the action selection percentage is greater than or smaller than certain values are received over time.

Epidemic control is a topical subject given the recent prevalence of COVID-19 and approaches to this question vary wildly in terms of the how the problem is modelled (Arango and Pelov 2020; Charpentier et al. 2020; Colas et al. 2020; Brauer and Castillo-Chávez 2012; Anderson 2013; Berestizshevsky et al. 2021). With no consensus on the appropriate model to use, our model is chosen as it is sufficiently complex to demonstrate the efficacy of our agent.

Results

Figures 1 and 2 display the mean learning curves for DynamicHedgeAIXI, U-Tree, PARSS and HedgeAIXI with standard deviations computed over five random seeds. DynamicHedgeAIXI and HedgeAIXI vastly outperform U-Tree and PARSS on the Taxi and Epidemic Control domains. PARSS and U-Tree were likely unable to perform because these two domains require a large number of predicates to be evaluated together rather than individually for node splits. Also, both methods can be susceptible to producing spurious splits, making the resulting state space too difficult to learn with. On both Taxi and Epidemic Control, the learning curves for both DynamicHedgeAIXI and HedgeAIXI start

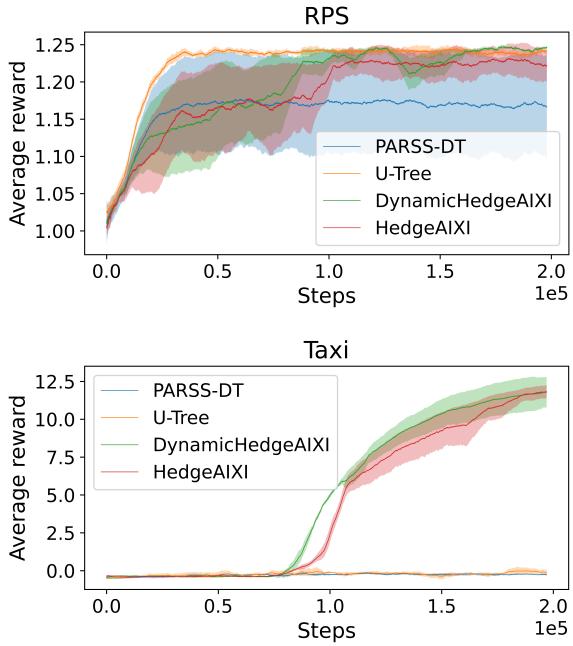

Figure 1: Learning curve on biased RPS and Taxi domains.

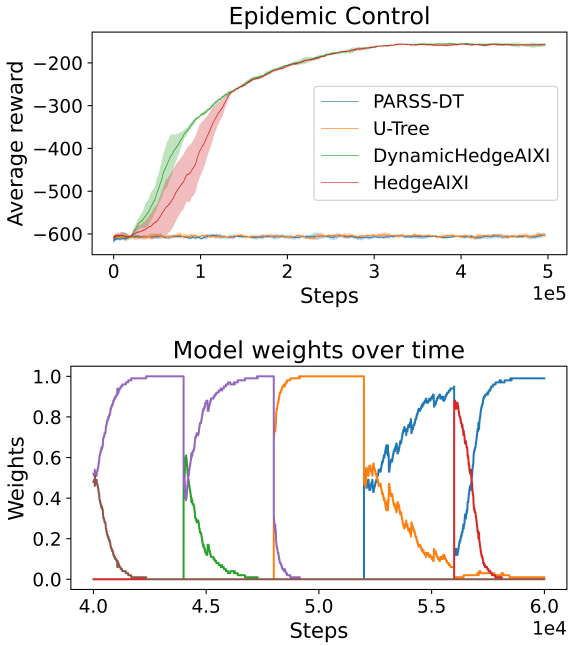

Figure 2: Epidemic control: plot of learning curve and agent's model weights over time.

out flat due to the lack of useful environment models initially. As useful models are introduced, both agents' performance begin to pick up. DynamicHedgeAIXI's performance picks up faster than HedgeAIXI. This effect is especially pronounced in the Taxi domain. HedgeAIXI's slower convergence on these more complex domains is likely due to it taking longer to overcome the bad weights left behind by previous models. This highlights the fast adaptation that DynamicHedge can provide. On the RPS environment, DynamicHedgeAIXI and HedgeAIXI were more unstable compared to U-Tree. DynamicHedgeAIXI's initial instability likely comes from the frequent re-distribution of posterior weight towards newly entering models. In contrast, since the environment is rather small and requires only two predicates to fully model, U-Tree was able to find the correct splits quickly and did not consider additional splits. Nevertheless, DynamicHedgeAIXI eventually converges to optimal performance.

To further demonstrate DynamicHedgeAIXI's adaptive behaviour, Figure 2 displays the model weights over time during the 40k to 60k steps period of a single run of the Epidemic Control problem. The sharp drops in weight for the highest weighted model correspond to when a new model enters. Similarly, the sharp spikes in a coloured line indicate when a previous model has been removed and replaced with a new model. Between models being introduced, the weights converge quickly to the best performing model. Over this period of time, the highest weighted model switches to a newly entering model twice. Considering the algorithm's performance in the Epidemic Control domain, DynamicHedgeAIXI can be seen as adapting effectively.

Discussion and Conclusion

The key contribution of this paper is the introduction of an exact and efficient Bayesian mixture modelling algorithm for general reinforcement learning agents in the dynamic knowledge injection setting, a setting that formalises a form of Human-AI teaming construct. The algorithm generalizes and integrates Hedge and CTW, two highly successful Bayesian mixture models, and we provide theoretical and empirical results that demonstrate the algorithm's practical utility. Our research opens the door to the principled construction of collaborative Human-AI teaming systems for complex environments, where (a) the human expert can use an expressive knowledge representation formalism to supply the agent with new background knowledge in an interactive manner to reduce model bias in the agent's initial knowledge base, and (b) the AI agent can learn to act optimally with respect to an exact Bayesian mixture model that comes with performance guarantees measured against the best aspects of human knowledge injection. Our work is another proof point that the AIXI theory can provide good guidance on the design of new general reinforcement learning agents. A key limitation of our approach is the transfer of a huge burden of learning to the human expert. This can be alleviated by augmenting the human expert with a semi-automatic predicate generation process. The predicate-selection algorithm described in (Yang-Zhao, Wang, and Ng 2022) can be run in micro-batches to instantiate that external process. More generally, the use of statistical predicate invention (Cropper, Morel, and Muggleton 2020; Kok and Domingos 2007) techniques is an important direction for future work.

Acknowledgments

The authors would like to thank Joel Veness, Elliot Catt, Jordi Grau-Moya, and Tim Genewein for their thoughtful feedback on an earlier draft. This work was in parts supported by ARC grant DP150104590.

References

Anderson, R. M. 2013. *The population dynamics of infectious diseases: theory and applications*. Springer.

Arango, M.; and Pelov, L. 2020. COVID-19 Pandemic Cyclic Lockdown Optimization Using Reinforcement Learning. *CoRR*, abs/2009.04647.

Bellemare, M. G.; Veness, J.; and Bowling, M. 2013. Bayesian Learning of Recursively Factored Environments. In *Proceedings of the 30th International Conference on Machine Learning*, 1211–1219.

Bellemare, M. G.; Veness, J.; and Talvitie, E. 2014. Skip Context Tree Switching. In *Proceedings of the 31th International Conference on Machine Learning*, 1458–1466.

Berestizshevsky, K.; Sadzi, K.-E.; Even, G.; and Shahar, M. 2021. Optimization of resource-constrained policies for COVID-19 testing and quarantining. *Journal of Communications and Networks*, 23(5): 326–339.

Brauer, F.; and Castillo-Chávez, C. 2012. *Mathematical Models in Population Biology and Epidemiology*. Springer.

Cesa-Bianchi, N.; and Lugosi, G. 2006. *Prediction, learning, and games*. Cambridge university press.

Charpentier, A.; Elie, R.; Laurière, M.; and Tran, V. C. 2020. COVID-19 pandemic control: balancing detection policy and lockdown intervention under ICU sustainability.

Chernov, A.; and Vovk, V. 2009. Prediction with Expert Evaluators' Advice. In Gavaldà, R.; Lugosi, G.; Zeugmann, T.; and Zilles, S., eds., *Algorithmic Learning Theory*, 8–22. Springer Berlin Heidelberg. ISBN 978-3-642-04414-4.

Colas, C.; Hejblum, B.; Rouillon, S.; Thiébaut, R.; Oudeyer, P.-Y.; Moulin-Frier, C.; and Prague, M. 2020. EpidemiOptim: A Toolbox for the Optimization of Control Policies in Epidemiological Models.

Cropper, A.; Morel, R.; and Muggleton, S. H. 2020. Learning Higher-Order Programs through Predicate Invention. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 13655–13658. AAAI Press.

Dietterich, T. G. 2000. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *J. Artif. Intell. Res.*, 13: 227–303.

Eysenbach, B.; Khazatsky, A.; Levine, S.; and Salakhutdinov, R. 2023. Mismatched No More: Joint Model-Policy Optimization for Model-Based RL. *arXiv:2110.02758*.

Farias, V. F.; Moallemi, C. C.; Roy, B. V.; and Weissman, T. 2010. Universal reinforcement learning. *IEEE Trans. Inf. Theory*, 56(5): 2441–2454.

Farmer, W. 2008. The Seven Virtues of Simple Type Theory. *Journal of Applied Logic*, 6(3): 267–286.

Freund, Y.; Schapire, R. E.; Singer, Y.; and Warmuth, M. K. 1997. Using and Combining Predictors That Specialize. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 334–343. ACM. ISBN 0897918886.

Guimerà, R.; Danon, L.; Díaz-Guilera, A.; Giralt, F.; and Arenas, A. 2003. Self-similar community structure in a network of human interactions. *Phys. Rev. E*, 68: 065103.

Hostetler, J.; Fern, A.; and Dietterich, T. 2017. Sample-Based Tree Search with Fixed and Adaptive State Abstractions. *J. Artif. Int. Res.*, 60(1): 717–777.

Hutter, M. 2005. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Berlin: Springer. ISBN 3-540-22139-5.

Hutter, M. 2009. Feature Reinforcement Learning: Part I. Unstructured MDPs. In *J. Artif. Gen. Intell.*

Jordan, M. I. 1995. Why the logistic function? A tutorial discussion on probabilities and neural networks.

Kocsis, L.; and Szepesvári, C. 2006. Bandit Based Monte-Carlo Planning. In Fürnkranz, J.; Scheffer, T.; and Spiliopoulou, M., eds., *ECML 2006*, 282–293. Springer Berlin Heidelberg. ISBN 978-3-540-46056-5.

Kok, S.; and Domingos, P. M. 2007. Statistical predicate invention. In Ghahramani, Z., ed., *Machine Learning, Proceedings of the Twenty-Fourth International Conference*, 433–440. ACM.

Krichevsky, R.; and Trofimov, V. 2006. The Performance of Universal Encoding. *IEEE Trans. Inf. Theor.*, 27(2): 199–207.

Lambert, N.; Amos, B.; Yadan, O.; and Calandra, R. 2021. Objective Mismatch in Model-based Reinforcement Learning. arXiv:2002.04523.

Lloyd, J. W. 2003. *Logic for Learning: Learning Comprehensible Theories from Structured Data*. Springer.

Lloyd, J. W.; and Ng, K. S. 2011. Declarative programming for agent applications. *Autonomous Agents Multi Agent Systems*, 23(2): 224–272.

McCallum, A. K. 1996. *Reinforcement learning with selective perception and hidden state*. University of Rochester.

Mourtada, J.; and Maillard, O.-A. 2017. Efficient tracking of a growing number of experts. In *International Conference on Algorithmic Learning Theory*, 517–539.

Newman, M. 2018. *Networks*. Oxford University Press.

Nowzari, C.; Preciado, V. M.; and Pappas, G. J. 2016. Analysis and control of epidemics: A survey of spreading processes on complex networks. *IEEE Control Systems Magazine*, 36(1): 26–46.

Pastor-Satorras, R.; Castellano, C.; Van Mieghem, P.; and Vespignani, A. 2015. Epidemic processes in complex networks. *Reviews of Modern Physics*, 87(3).

Rossi, R. A.; and Ahmed, N. K. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*.

Solomonoff, R. J. 1997. The Discovery of Algorithmic Probability. *J. Comput. Syst. Sci.*, 55(1): 73–88.

Veness, J.; Ng, K. S.; Hutter, M.; and Bowling, M. H. 2012. Context Tree Switching. In Storer, J. A.; and Marcellin, M. W., eds., *2012 Data Compression Conference*, 327–336. IEEE Computer Society.

Veness, J.; Ng, K. S.; Hutter, M.; Uther, W. T. B.; and Silver, D. 2011. A Monte-Carlo AIXI Approximation. *J. Artif. Intell. Res.*, 40: 95–142.

Veness, J.; White, M.; Bowling, M.; and György, A. 2013. Partition Tree Weighting. In Bilgin, A.; Marcellin, M. W.; Serra-Sagristà, J.; and Storer, J. A., eds., *2013 Data Compression Conference*, 321–330. IEEE.

Vovk, V. 1998. A Game of Prediction with Expert Advice. *Journal of Computer and System Sciences*, 56(2): 153–173.

Willems, F.; Shtarkov, Y.; and Tjalkens, T. 1995. The context-tree weighting method: basic properties. *IEEE Transactions on Information Theory*, 41(3): 653–664.

Yang-Zhao, S.; Ng, K. S.; and Hutter, M. 2023. Dynamic Knowledge Injection for AIXI Agents. *arXiv preprint arXiv:2312.16184*.

Yang-Zhao, S.; Wang, T.; and Ng, K. S. 2022. A Direct Approximation of AIXI Using Logical State Abstractions. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.