frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 23 December 2020
doi: 10.3389/fnbot.2020.578675

Detecting Changes and Avoiding Catastrophic Forgetting in Dynamic Partially Observable Environments

Jeffery Dick[1]*, *Pawel Ladosz*[1], *Eseoghene Ben-Iwhiwhu*[1], *Hideyasu Shimadzu*[2,3], *Peter Kinnell*[4], *Praveen K. Pilly*[5], *Soheil Kolouri*[5] and *Andrea Soltoggio*[1]*

[1] Department of Computer Science, Loughborough University, Loughborough, United Kingdom, [2] Mathematical Sciences, Loughborough University, Loughborough, United Kingdom, [3] Teikyo University Graduate School of Public Health, Tokyo, Japan, [4] School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, Loughborough, United Kingdom, [5] HRL Laboratories, Malibu, CA, United States

The ability of an agent to detect changes in an environment is key to successful adaptation. This ability involves at least two phases: learning a model of an environment, and detecting that a change is likely to have occurred when this model is no longer accurate. This task is particularly challenging in partially observable environments, such as those modeled with partially observable Markov decision processes (POMDPs). Some predictive learners are able to infer the state from observations and thus perform better with partial observability. Predictive state representations (PSRs) and neural networks are two such tools that can be trained to predict the probabilities of future observations. However, most such existing methods focus primarily on static problems in which only one environment is learned. In this paper, we propose an algorithm that uses statistical tests to estimate the probability of different predictive models to fit the current environment. We exploit the underlying probability distributions of predictive models to provide a fast and explainable method to assess and justify the model's beliefs about the current environment. Crucially, by doing so, the method can label incoming data as fitting different models, and thus can continuously train separate models in different environments. This new method is shown to prevent catastrophic forgetting when new environments, or tasks, are encountered. The method can also be of use when AI-informed decisions require justifications because its beliefs are based on statistical evidence from observations. We empirically demonstrate the benefit of the novel method with simulations in a set of POMDP environments.

Keywords: POMDP, PSR, continual learning, catastrophic forgetting, lifelong learning, neural network

OPEN ACCESS

Edited by:
James Leland Olds, George Mason University, United States

Reviewed by:
Eiji Uchibe, Advanced Telecommunications Research Institute International (ATR), Japan
Juan V. Sanchez-Andres, University of Jaume I, Spain

***Correspondence:**
Jeffery Dick
j.dick@lboro.ac.uk
Andrea Soltoggio
a.soltoggio@lboro.ac.uk

Received: *30 June 2020*
Accepted: *20 November 2020*
Published: *23 December 2020*

Citation:
Dick J, Ladosz P, Ben-Iwhiwhu E, Shimadzu H, Kinnell P, Pilly PK, Kolouri S and Soltoggio A (2020) Detecting Changes and Avoiding Catastrophic Forgetting in Dynamic Partially Observable Environments. Front. Neurorobot. 14:578675. doi: 10.3389/fnbot.2020.578675

1. INTRODUCTION

A useful skill for an agent that explores the world and learns to act in it is the ability to predict what happens next (Geisser, 1993). One way is to try to learn a model of the world so that predictions are generated within the agent and compared with observations to improve the model. However, this idea assumes that it is possible to learn a large and static model of the entire world. In reality, it is more feasible to try to learn a model of a subset of the world, i.e., an environment. Therefore,

an agent may interact in different environments at different points in time. This is a condition that challenges learning algorithms that often need to be set manually by a user that labels tasks or substitutes models for each new task or environment.

A more effective agent would be able to learn different tasks or environments more autonomously, incorporating new knowledge without forgetting the skills learned in a previous task or environment. To accomplish this, an agent needs to learn an environment and detect when a change occurs, or when a completely new environment is met. By doing so, a lifelong learning agent will also remember previously learned environments, and quickly recover their models if old conditions return.

The majority of current machine learning approaches, however, assume that learning occurs in one static environment. These approaches make it possible to optimize policies for one problem, but do not scale well to learning multiple problems, or to learning in an incremental way more tasks over a lifetime (Thrun, 1998). Approaches known as meta reinforcement learning (Finn et al., 2017; Rothfuss et al., 2018; Rakelly et al., 2019; Zintgraf et al., 2019) are designed to learn multiple tasks, but they assume that a signal is given when the task changes. While this is a step toward learning more tasks sequentially, such algorithms still require a teaching signal that labels different tasks. Additionally, reinforcement learning algorithms are intended to optimize a policy that maximizes reward (Sutton and Barto, 2018). Therefore, the knowledge of the environment is implicit in the policy and shaped by the reward function.

One method to quickly adapt to changes in the environment is provided by reinforcement learning approaches that explicitly model the environment, and therefore can rapidly search the parameter space when those change. These approaches, known as model-based reinforcement learning (Doya et al., 2002; Nagabandi et al., 2018; Lecarpentier and Rachelson, 2019), require the model to be hand-designed, and also learn policies with the aim to maximize a reward. In short, because reinforcement learning aims to provide actions that maximize reward, their applicability is limited when there are no actions or rewards available.

If an environment does not provide rewards, it can still be explored and learned with the aim, e.g., to predict future events. The concepts of Markov chain and of Markov decision process (MDP) are abstractions to model an environment when actions are, respectively, absent or present (Bellman, 1957). An extension of MDPs are partially observable Markov decision processes (POMDPs) that account for the fact that observations from a system do not always reveal the state. POMDPs have many important applications in the real world, e.g., the airplane collision avoidance system ACAS X is based on POMDP models (Kochenderfer et al., 2015). POMDPs have also been used to model brain information processing for decision making under uncertainty (Rao, 2010), and to model working memory in the brain (Todd et al., 2009). While POMDPs are a flexible tool to model a variety of real world systems, the assumption of partial observability of the underlying states in the observed environment is precisely what makes it difficult to derive accurate POMDPs from data.

Because POMDPs can include a reward function, much of the research in learning POMDPs falls under reinforcement learning theory and is intended to find an optimal policy in a rewarding environment (see e.g., Shani et al., 2005). One exception is the Baum-Welch algorithm (Rabiner, 1989), designed to generate Hidden Markov Models, that can be adapted for POMDP environments by incorporating actions. One limitation of this method is that it requires knowing the number of states in advance, and works best when provided with an initial estimate of the underlying POMDP.

Predictive state representation (PSR) is a general representation of a POMDP that does not need to learn the underlying states. PSRs, instead, learn the probabilities of future observations. Due to the nature of PSR methods, which learn directly from observations rather than trying to find hidden underlying states, discovering and learning an accurate PSR of a POMDP environment is faster than trying to reconstruct the underlying MDP (Littman et al., 2002). A variety of algorithms have been proposed to improve the learning of PSRs such as transformed predictive state representations (TPSRs) (Rosencrantz et al., 2004) and compressed predictive state representations (CPSRs) (Hamilton et al., 2013). Some algorithms improve the learning method, often utilizing TPSRs and CPSRs (McCracken and Bowling, 2005; Yun-Long and Ren-Hou, 2009; Liu et al., 2016; Downey et al., 2017), while others allow the agent to learn in more complex domains, e.g., with continuous action or observation spaces (Wingate and Singh, 2007; Boots et al., 2013).

Other parametric models such as neural networks (Bishop, 1995) can also take a history of recent actions and observations as input, and be trained to predict the next observation. These approaches are less explainable, but have grown in popularity with the resurgence of neural networks, the use of deep and recurrent networks, and powerful hardware for training (Schmidhuber, 2015).

The approaches cited so far assume that the environment is stationary. Therefore, we can hypothesize that under dynamic conditions where the agent switches between environments over time, such approaches will either learn an average of the environments, or learn accurately the most recent environment while forgetting the previous ones. One example of dynamic conditions is an autonomous driving problem in which a vehicle encounters different environments, such as different driving rules or weather conditions. In theory, two or more POMDP models that alternate over time can be modeled as one single POMDP in which a non-observable state determines the sub-part of the model that generates the current data. However, this approach is likely to increase the complexity of the model significantly. Thus, a hierarchical approach in which different POMDP models are used to predict different environments may be more scalable.

Assume, e.g., a system in which a transition from a state A to B occurs consistently with probability 1, but after some time has passed, the environment dynamics change such that state A leads to C with probability 1. The challenge in learning this case with one single model is that rather than capturing the hidden state,

the model could learn that the environment transitions from A to B or to C with a probability of 0.5 for each state. This is true on average, but inaccurate at any particular point in time. As a consequence, the result will be either a model with low accuracy, if a slow learning rate is used, or catastrophic forgetting if a faster learning rate is used.

The idea presented in this paper is to explicitly learn such hierarchically nested hidden states by means of a statistical framework that selects different predictive models to fit a data stream at different times. The proposed approach tracks the probability of a current window of data of fitting different models, and thus the probability of an agent being in one of many possible environments when the only cues, e.g., observations and actions, are implicit in the data stream. This is done by comparing the expected frequencies of observations derived from predictive models with the observed frequencies in the current data stream. Discrete observations and actions follow multinomial distributions, thus, performing χ^2 tests is a viable method of estimating the probability of the new observed data to fit a learned predictive model.

An important consequence of assessing a model's probability of fitting the current data is that data points at different times can be assigned to different models to improve them separately and independently. By doing so, we can learn different models under the assumption of stationary conditions and implement continual learning of multiple environments, thus avoiding catastrophic forgetting in dynamic POMDPs. The proposed method provides an evidence-based and explainable algorithm to justify the belief of the system. Moreover, the novel approach does not need reward signals to learn models for different environments, making it a more general method than reward-based approaches such as reinforcement learning.

The next section provides the background on POMDPs that is necessary to introduce the novel algorithm that we name adaptive model detection (AMD). We also briefly introduce PSRs and a simple neural network as baseline model learners to be employed by the novel AMD algorithm explained in section 3. Simulation results are presented in section 4 followed by a discussion and conclusion.

2. BACKGROUND

Predictive models such as predictive state representations (Littman et al., 2002), neural networks and POMDPs have been extensively used in the past to model dynamical systems with discrete representations. This section provides the background for these approaches that lay the framework for the method in this paper.

2.1. Predictive State Representation (PSR) and POMDP

Predictive state representation (PSR) is a model to predict observations in stochastic environments, including POMDPs. A POMDP is defined as a hextuple $\{\mathcal{S}, \mathcal{A}, T, \mathcal{O}, \Omega, R\}$, where $\mathcal{S}$ is the set of underlying MDP states; $\mathcal{A}$ is the set of actions; T is the transition function, $T : \mathcal{A} \times \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$, which gives the probability of transitioning from one state to another given the action taken; R is the reward function; $\mathcal{O}$ is the set of observations; and Ω is the set of conditional observation probabilities. POMDPs differ from MDPs in that the current observation is not sufficient for an agent to be able to determine its underlying state.

Let t be a finite stream of action-observation pairs. Then t is a test, and we let $\mathcal{T}$ be the set of all tests. Let the history $h_i \in H$ of the agent at time i be the stream of action-observation pairs $a_j \in \mathcal{A}, o_j \in \mathcal{O}, \forall j \in [0, i) \cap \mathbb{N}$ observed up to time i.

A PSR (Littman et al., 2002) of an environment consists of a set of core tests, Q; a set of $|Q|$-dimensional $m_{a,o,t}$ vectors for all $a \in \mathcal{A}, o \in \mathcal{O}, t \in Q$; and an initial state.

The set of core tests, Q, is a finite subset of $\mathcal{T}$, with the property that $P(t \mid h)$ for all $t \in \mathcal{T}, h \in H$ can be found as some *linear combination* of the probabilities $P(q \mid h)$ for all $q \in Q$. The empty test, $\epsilon = \{\}$, is always included in Q, such that $P(\epsilon \mid h) = 1$ for all h which are possible under the PSR model. The PSR state vector after observing history h, $y(h)$, is a $(1 \times |Q|)$ vector which holds $P(q \mid h)$ for each $q \in Q$. By stacking the rows of $m_{a,o,t}$ for all $t \in Q$, we obtain a $(|Q| \times |Q|)$ *projection vector* $M_{a,o}$ for every length 1 test (a, o). For all $h \in H$ we have that $P(o \mid h, a) = y(h) \times M_{a,o}^T$. Projection vectors for longer tests can be created by multiplying projection vectors for shorter tests. For example, $M_{a_1,o_1,a_2,o_2} = M_{a_2,o_2} \times M_{a_1,o_1}$, where $\times$ is matrix multiplication.

To maintain an accurate state vector, $m_{a,o,t}$ must be available for all $a \in \mathcal{A}, o \in \mathcal{O}, t \in Q$. Let $Q = \{t_1, t_2, \ldots, t_n\}$. This can be used to obtain the state vector at time i, $y(h_i)$, given the a_i, o_i action observation pair observed at timestep i, and $y(h_{i-1})$. Recall that the PSR state vector contains the probabilities of each core test, and let $y_j(h)$ denote the element in $y(h)$ corresponding to core test t_j. Then, the probability of each core test can be found as follows. For all $t_j \in Q$:

$$y_j(h_i) = P(t_j \mid h_{i-1}, a_i, o_i) = \frac{y(h_{i-1}) \times m_{a_i,o_i,t_j}^T}{y(h_{i-1}) \times m_{a_i,o_i}^T} \quad . \tag{1}$$

Equivalently, we can use the previously defined projection vectors:

$$y(h_i) = \frac{y(h_{i-1}) \times M_{a_i,o_i}^T}{y(h_{i-1}) \times m_{a_i,o_i}^T} \quad . \tag{2}$$

From the definition above, it follows that PSRs can give an indication of the probability of certain transitions to occur. In particular, the following theorem specifies the probability of observing particular action observation pairs:

THEOREM 1. Given that the state vector is $y(h_{i-1})$ at time $i - 1$, the probability of seeing observation o_i after taking action a_i at time step i is given by the following equation

$$P(o_i | h_{i-1}, a_i) = y(h_{i-1}) \times m_{a_i,o_i}^T \quad . \tag{3}$$

Proof: By construction. Note that the state vector contains enough information to accurately predict future observations even in partially observable environments. The state vector acts not only as a prediction, but also as an internal state for the PSR model.

A natural question to ask is, 'what is the predictive state of an empty history $y(\epsilon)$?'. If the environment is known to always start in a given underlying state, the corresponding predictive state may be used. However, this is a strong assumption in general. If we assume the agent is likely to start in each underlying state proportionally to the amount of time previously spent in that state, $y(\epsilon)$ can be set to the *stationary distribution*. The stationary distribution, which is the weighted expected value of y over all time steps, is given by

$$y(\epsilon) = \frac{\sum_{h \in H} y(h)}{|H|} \tag{4}$$

where H is the set of all histories. As H is an infinite set, this is calculated instead from the histories that the agent has seen. When calculated this way, the stationary distribution may change depending on the policy followed by the agent. Additionally, as the stationary distribution represents a distribution over all states, it may not be a state that the agent can reach through normal exploration; however, it represents a positive probability for all states previously visited.

The state space of the PSR is the set of states that can be generated recursively by $y = \frac{y' \times M^T_{a,o}}{y' \times m^T_{a,o}}$, for all $a \in \mathcal{A}, o \in \mathcal{O}$, where y' is known to be in the state space of the PSR, and $y \times m^T_{a,o}$ is non-zero. In order to generate the set of states, an initial state y' must be assumed to be in the state space. Sometimes an initial state is provided when the starting state of the environment is known. However, when no initial state is provided, the stationary distribution may be used.

There are several algorithms for learning PSRs offline, but relatively few for learning and discovering tests online. One such algorithm is the constrained gradient algorithm (McCracken and Bowling, 2005), which we use in our experiments for learning PSRs.

2.2. Neural Network Predictors

A simple neural network (Bishop, 1995) can be trained to predict observations in a given environment. Given a time window of duration i, a neural network can take the observations $o \in \mathcal{O}$ and the actions $a \in \mathcal{A}$ and predict the new observation o at time $i + 1$. If the softmax transfer function is used to produce output probabilities, the network can also be trained to predict the probability of each observation at $i + 1$ in stochastic environments. The difference between the prediction and the observation can be used to train such a system with gradient descent. As for the PSR model explained in the previous section, a neural network predictor can be trained effectively only if stationary conditions are assumed during the training phase. Thus, changes in the environment such as those occurring in dynamic POMDPs (see next section) are challenging conditions that this study addresses.

Neural networks, PSRs, and other predictive models have one thing in common: they give as output a prediction of the probability of seeing each possible observation next. The sum of probabilities of seeing each observation is 1. Let K be a predictive model, $\mathcal{O}$ be the set of possible observations, and $\mathcal{A}$ be the set of possible actions. Then, we define $P(o \mid K_i)$ to be the predicted probability of observing o at time i given by K. Additionally, we define $P(\mathcal{O} \mid K_i)$ as the probability distribution over all observations at time i given by K.

2.3. Dynamic POMDPs

Assume that an environment remains stationary for a certain amount of time. Under this assumption, it is possible to learn a model of the environment (e.g., a PSR, as detailed in McCracken and Bowling, 2005). If, after a certain amount of time, the environment changes, the continual training of the same model will lead to inaccurate predictions at first, and catastrophic forgetting of the first environment in the long term. This is because the assumption of stationary conditions is not valid, and one model tries to learn an average distribution of two or more distributions that occur in different environments. These changing environments are similar to switching hidden Markov models (SHMMs) investigated in Chuk et al. (2020) and Höffken et al. (2009).

Let $\mathbb{V} = \{V_1, V_2, \ldots, V_m\}$ be a set of distinct POMDP environments. A dynamic POMDP environment, D, behaves as a single environment $V_i \in \mathbb{V}$ for a number of time steps, n_0, before changing its behavior to another environment, $V_j \in \mathbb{V}$. The dynamic environment D may switch to any environment in $\mathbb{V}$ every n_k time steps, with $k \in \mathbb{N}$ and $n_k \gg 1$. Effectively, these dynamics can be seen as hierarchical large stationary models where a state variable $z \in \mathbb{N}$ determines the specific environment V_z at a given time. However, z is not observable and can only be derived inferring which specific environment from the set $\mathbb{V}$ matches the current stream of data. We assume that transitions between environments in $\mathbb{V}$ occur with considerably lower frequency than state transitions within the environments V_z. This assumption reflects two points: (1) in real world scenarios, generally, two environments can be thought of as being distinct when transitions from one to another occur rarely, otherwise it makes more sense to consider them as one environment; (2) for a model to learn and predict one environment, it is necessary to experience the environment for a minimum number of steps that capture transitions within it.

Such dynamic conditions occur typically when an agent learns to predict an environment, e.g., to navigate in a house, and is then required to learn a new somewhat different environment, e.g., to navigate in an office. The new environment might bear a similarity with the previous one, but also significant differences. Desirable skills of an agent include the ability to detect that there is a new environment, the ability to learn the new environment quickly, possibly exploiting previous knowledge, and also retain the knowledge of the previous environment (avoiding catastrophic forgetting). The aim of this study is precisely to achieve such capabilities as explained in the next section.

3. ADAPTIVE MODEL DETECTION

The idea and the method for detecting different environments and training different models according to such detection is explained in this section. We name our new approach adaptive model detection (AMD) because it detects likely models to fit the

data, and works with adaptive models that evolve as new data is collected.

3.1. Statistical Model Selection

Given a set of statistical models that each predict an environment, a question that can be asked is: what model is best at predicting a given stream of data? Several approaches have been proposed in the literature to perform *model selection* (Cox, 2006). Approaches for model selections are based on the information theory such as the Akaike information criterion (Akaike, 1974) and the Bayesian (or Schwarz) information criterion (Schwarz, 1978). The idea is to measure the information that is lost due to the difference in statistical distributions between the model and the data using the Kullback-Leibler divergence (Kullback, 1997). By doing so, it is possible to select a best fitting model that minimizes this difference. Different statistical methods, however, may be more or less appropriate or accurate according to a number of factors including the number of data points and the assumptions on the statistical distributions that are being observed.

Given a recent window of data from one environment, *hypothesis testing* can be used to accept or reject the null hypothesis that the distribution of the environment data matches the distribution of the model-generated data (Lehmann and Romano, 2006). The ability to accept or reject such a null hypothesis is a valuable tool, particularly in the context of explainable AI applications in which a model is used to predict a data stream. If the current data stream has a very low p-value, it is reasonable to reject the null hypothesis that the model is correct. On the contrary, if the null hypothesis cannot be rejected, the model offers a good approximation and it is therefore reasonable to (1) consider it as a good-enough predictor and (2) use new data to further improve it via a training process.

We assume that (1) a data stream is generated by one environment V_i from the set $\mathbb{V}$; (2) that we want to learn and identify which model $K_i \in \mathbb{K}$ describes the current data stream, including if none of the current models describe the data; (3) the data stream produces a set of finite discrete or categorical outcomes. Therefore, we use hypothesis testing instead of model selection, so that we can determine when none of the existing models fit the data, allowing us to create a new model. To identify which specific V_i is generating the data, the problem can be formulated as selecting the corresponding model K_i that maximizes the likelihood

$$\underset{i}{\operatorname{argmax}}\, \hat{L} = P(h|K_i) \tag{5}$$

where h is a limited time window of recent input data.

It is important to note that there are no guarantees that a current set of models, $\mathbb{K}$, can accurately predict a corresponding set of environments $\mathbb{V}$. However, the key idea tested is: assuming we can select the most fitting model given by Equation (5), then we can associate a particular time window of the data stream and use it to improve the model K_i to maximize $\hat{L}$. Therefore, such an approach can be used both to learn and to find the best set of models that fit a set of environments.

The data generated by the set of actions and observations in a POMDP as described in the previous section form a multinomial distribution. Thus, the χ^2 test can be used to accept or reject the null hypothesis that a recent time window of data is generated by a given model. In the next sections, the procedure to compute the degrees of freedom and the χ^2 p-values is presented.

3.2. Calculating Degrees of Freedom

The degrees of freedom (DF) of a statistical model K corresponds to the number of independent parameters in the model, and is required to perform statistical tests such as χ^2. Predictive models have in common the ability to predict the probability distribution of the next observation given a history or internal state. Note that we are not trying to find the number of independent parameters in the predictive agent, but in the underlying model the agent predicts. Let $o' \in \mathcal{O}$. If, for all $o \in \{\mathcal{O} \setminus o'\}$ we know the value of $P(o \mid K_i)$, then

$$P(o' \mid K_i) = 1 - \sum_{o \in \{\mathcal{O} \setminus o'\}} P(o \mid K_i) \quad . \tag{6}$$

Therefore, the number of independent parameters for each prediction is $|\mathcal{O}| - 1$ as the final observation's probability can be inferred from the others. Each prediction may be independent in the predictive model, thus, we assume that the predictive model is an approximation to a statistical model (the underlying POMDP) with a number of independent states that is much smaller than the length of a history of data points. To estimate the number of independent states in the underlying model, we cluster together similar predictions in the predictive model. These can be clustered according to the predictive model's hidden states, the prediction of the next observation, or the prediction of several next observations.

3.3. Sampling and Clustering Probabilities

The adaptive model detection algorithm (AMD) keeps a history window W of up to L recent prediction observation pairs, where L is a parameter of the algorithm. The choice of L affects the behavior of the algorithm as shown in the results section with an analysis of different values for L.

Knowing how many times each prediction has been made is necessary to determine whether the environment behaves as the predictive model expects. To count the number of times different predictions are given by a learning model, it is necessary to cluster such predictions that are expressed as vectors with possibly slightly different probabilities values. Let $\mathcal{C}_K$ be the set of all clusters made by AMD for the predictive model. For a given cluster $c \in \mathcal{C}_K$, let $\bar{c}$ be the mean of the distributions $P(\mathcal{O} \mid K_i) \in c$ in the cluster, and $\bar{c}(o)$ therefore be the mean probability of observing observation o.

AMD uses the DBSCAN clustering algorithm (Ester et al., 1996) to form clusters of predictions given by a model. The scikit-learn (Pedregosa et al., 2011) vanilla implementation of the algorithm is used. As the data changes over time, the DBSCAN algorithm may form clusters differently on each timestep. This does not pose an issue, as $\bar{c}$ can be recalculated at each timestep, meaning that even when the clusters change, the expected and observed frequencies will be similar in an accurate agent. DBSCAN does not include into clusters those outlier points

which seem to not fit into larger clusters. Such a property is advantageous in our context because: (1) the mean distribution of a cluster $\bar{c}$ is therefore not affected by an outlier that was forced into it, and (2) clusters must be formed of a certain minimum size, the advantage of which is explained in section 3.5.

3.4. Calculating χ^2 *p*-Value

To compute the χ^2, AMD counts as $X_{c,o}$ the number of times in the history that each observation o follows predictions in each cluster c for all $c \in C$ and $o \in \mathcal{O}$. The number of times each observation o is expected to follow predictions in a given cluster is given by $E_{c,o} = |c| \times \bar{c}(o)$.

Thus, from Pearson (1900),

$$\chi^2 = \sum_{c \in \mathcal{C}} \sum_{o \in \mathcal{O}} \frac{X_{c,o} - E_{c,o}}{E_{c,o}} \quad . \tag{7}$$

Effectively, χ^2 measures how well the actual data matches the expected data, with higher values meaning that the observed data does not match the expected data well. For a general model, some values of $X_{c,o}$ and $E_{c,o}$ may be equal or close to 0, corresponding to impossible (or never observed) transitions. For these cases when $X_{c,o} = E_{c,o} = 0$, the value $\frac{X_{c,o} - E_{c,o}}{E_{c,o}}$ is set to 0.

Knowing the χ^2 value and DF allows us to find the p-value, which represents the probability of the observed data coming from the expected data distribution. This function is available in most statistical programs, code libraries, and toolkits as part of the χ^2 implementation[1].

3.5. χ^2 Testing for Adaptive Model Detection

Following the guidelines in (Yates et al., 1999), a χ^2 test is considered to be sufficiently accurate when "no more than 20% of the expected counts are less than five, and all individual expected counts are one or greater." Accordingly, a minimum history length is necessary to be able to perform the test, and the longer the history, the more accurate the test is expected to be. Unfortunately, with an arbitrary large POMDP, even a long history does not guarantee that all possible prediction observation pairs have been seen. Additionally, while a long history length makes p-values more accurate, a long history means that the assessment of the probability for a model can be done only over a long period of time, potentially losing granularity on the precise point of the transition.

Even with a long history length, if the agent encounters a sequence of observations it does not expect, it may produce an internal state or prediction unlike any it has generated before. Due to this, the cluster containing the unusual state or prediction could be small enough that the expected number of times a given observation follows states in the cluster is lower than 1. Conveniently, DBSCAN has a minimum cluster size, so such small clusters can be avoided entirely, unless of course the probability of observing a particular observation from states in a cluster are very low.

[1]In our implementations, the scipy function scipy.special.chdtrc is used.

An AMD that tests data against multiple predictive models $K_1, K_2, \ldots, K_n$ keeps track of the p-value of each one, and uses this tracked p-value to determine which environment it is most likely to be in. This allows AMD to select which model is trained at any point in time in a statistically motivated way, contributing to explainable decisions in AI. Additionally, detecting which environments are likely at given points in time opens up the possibility of applying a different policy based on the current environment.

Note that as long as the p-value is above the threshold at which the null hypothesis is rejected, it does not matter whether the value is low, high, or fluctuating. The AMD algorithm is summarized with pseudo code in Algorithm 1.

4. SIMULATION RESULTS

The effectiveness of the algorithm in a variety of settings is tested with computer simulations. The set of environments is introduced in section 4.1. The effect of the history length parameter L on the detection speed and stability is investigated in section 4.2. In section 4.3, the algorithm is extended to demonstrate how labeling incoming data can be used to continually train separate models, and thus avoid catastrophic forgetting.

4.1. Chosen Environments

The proposed algorithm was tested on a set of POMDPs of various size and complexity. The first set of problems (**Figures 1A–D**) is a series of uncontrolled POMDP environments, i.e., Markov chain environments, with only 2 observations (as there are more states than observations, the observation is given by the color of the state in **Figure 1**). These environments appear simple at first, however, they have different states and transition probabilities, and, due to their stochastic nature, some environments could be mistaken for others based on the data generated by exploration. For example, environment C creates a data stream that can be produced by environment D. However, when interacting with the environment over a longer period, the observations reveal the data stream is more likely to be generated by environment C than environment D.

The next set of environments, **Figures 1E1,E2**, represents a decision process originating in S0 where one sequence of actions takes the agent to S7, and all other sequences lead to S9. The challenge in this set is that the observations do not reveal the distance from S0, and thus make it hard for an agent to locate itself along the graph as it progresses from left to right. E1 and E2 have two further variations, E3 and E4 (not shown). E3 is the same as E1 but the transitions from S1 have inverted actions. Similarly, E4 is the same as E2 but transitions from S1 have inverted actions. These four environments are very similar in structure, but they require different policies to be traversed from $S0$ to $S7$.

Finally, environments **Figures 1F1,F2** have most of their transition probabilities being exactly the same. This means that transitions in the data stream distinguishing the two environments occur less frequently than in some other environments.

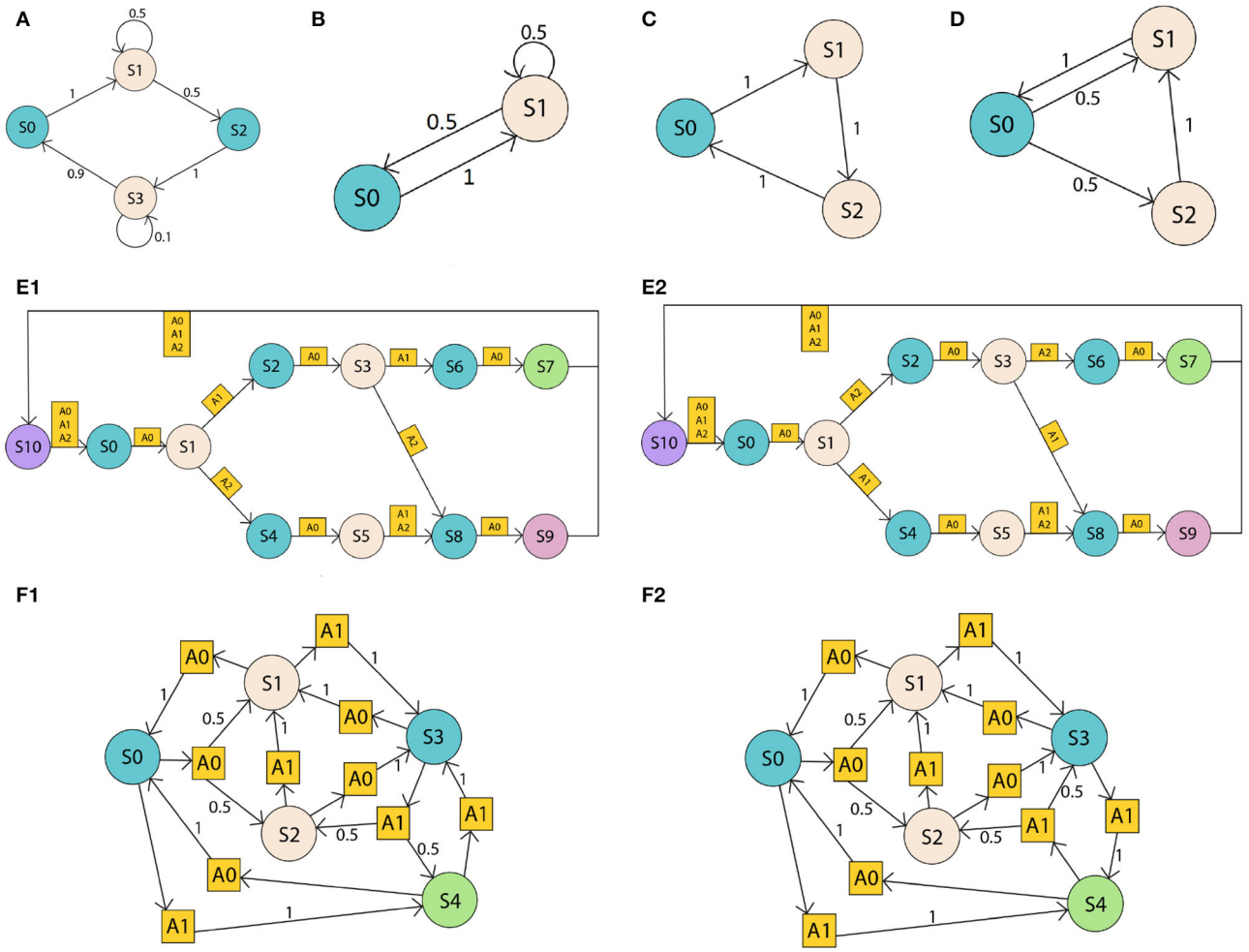

FIGURE 1 | Set of POMDP environments. States are represented with circles, actions with squares, and transition probabilities with numbers on arrows. The color of the state corresponds to the observation. Partial observability derives from the fact that different states provide the same observation. The first set **(A–D)** are simple Markov chains where transitions occurs without actions and probabilities indicated near the transition arrows. The second set **(E1,E1)** are more complex problems where actions determines different paths in a cycle but have deterministic transitions. Environments **(F1,F2)** are POMDPs with some deterministic and some stochastic transitions.

4.2. Speed and Reliability of Detections: Impact of History Length

The experiments in this section assess overall stability of the detection and the impact of the history length expressed by the parameter L. **Figure 2** shows the effect of different history lengths when tracking the probability of an accurate PSR model for environment C. In **Figure 2** (left), the values of L for 40, 60, 80, 100, and 120 are shown. In all cases, the AMD shows consistent p-values of 0 and 1, indicating that the model can confidently determine which model the data stream belongs to. The longer the history, the slower the change in p-value, confirming the intuitive notion that longer histories require more time steps to reveal a change in the environment. When assessed on the stochastic environments A and B (**Figure 2**, right), the dropping p-values indicate that stochasticity is a significant confounding factor in the detection of the environment.

It can be concluded that a small L is advantageous to detect changes more readily only if the environment is predominantly deterministic. When the environment has stochastic transitions, a longer history might be necessary to guarantee the stability of the detection. To further assess these dynamics, **Figure 3** shows the comparison of a short and a long history window ($L = 20$ and $L = 120$) on the deterministic environment C and the stochastic environment D. The AMD p-values show that while C can be tracked reliably with both $L = 20$ and $L = 120$, environment D (orange line) causes the p-value to oscillate, although with less amplitude, even with $L = 120$.

Other factors that can affect the stability of the p-values could include the complexity and the similarity with

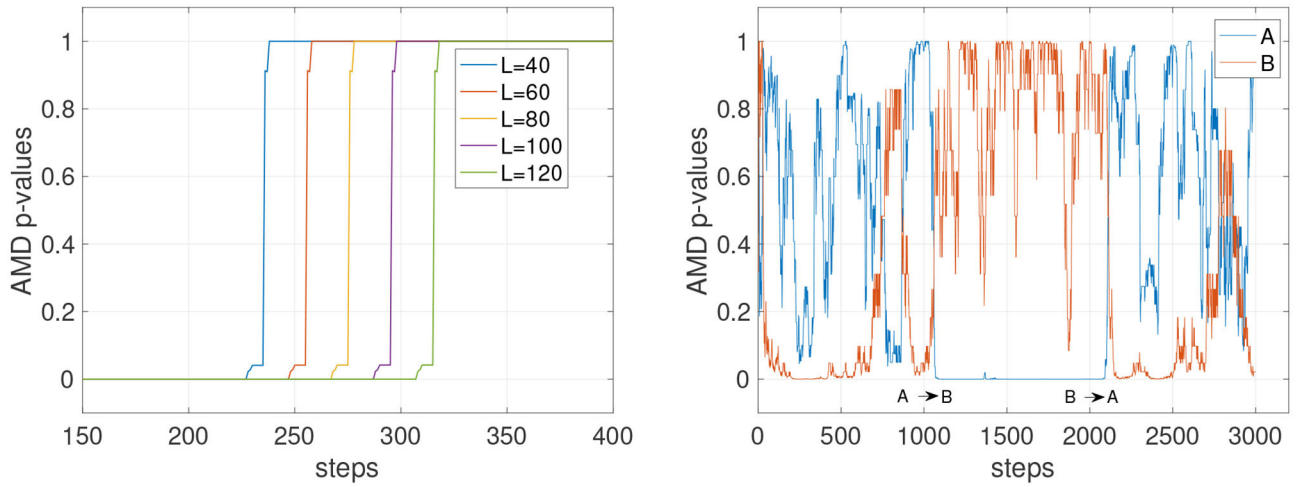

FIGURE 2 | AMD p-values with deterministic and stochastic environments. **(Left)** Tracking of environment C with different history lengths (parameter L). C is deterministic and the p-values are consistent at 0 and 1, but the time taken to detect the change in environment varies with the history length. **(Right)** AMD p-values for the model A and B with the stochastic environments A and B and $L = 120$. Tracking stochastic environments results in p-values predicting the wrong environment more frequently.

other environments. In **Figure 4** (left), the tracking of the environments E1 and E2 is shown with L varying from 60 to 220. The faster settings ($L = 60$ and $L = 100$) appear to detect E2 when still in E1. With $L = 140$, the detection becomes more reliable, and with $L = 180$ and $L = 220$ the detection is accurate, although slower after step 2,000 to detect the transition from E1 to E2. **Figure 4** (right) shows the p-values for all four E models tracked simultaneously. Despite the similarity of these four environments, the p-values show high confidence in determining which model currently matches the data stream. A similar result is also observed in **Figure 5** where the environments of the set F are tested. The stochasticity in this set does not affect significantly the stability of the p-values. This is because the environment is primarily deterministic, and although the points where the data streams differ do not occur often, the p-value drops significantly when they do occur.

4.3. Avoiding Catastrophic Forgetting With Continual Learning of Multiple Models

The ability to detect which environment the current stream of data belongs to allows the system to train different models independently, and thus implement continual learning and avoid catastrophic forgetting. The scenario devised in this section is when two unknown environments X and Y alternate while a learning system is trying to learn them from the data stream. This condition is particularly challenging because the data stream is generated by two different environments, both unknown. Therefore, there is an obvious bootstrap problem. How can the AMD know when the environment changes before any environment has been learned?

A reasonable assumption to overcome this problem is to assume that the data stream is initially generated by a single environment for a certain amount of time, so that one single model can be at least partially trained on the initial data stream.

4.3.1. AMD With Constrained Gradient PSRs

Two simple but highly stochastic environments, shown in **Figure 6**, are chosen for this test as depicted in **Figure 7**. The learning setup for this first test uses the constrained gradient PSR learner. It starts to learn a first model while environment X produces data for the first 10k steps. When the p-value suddenly drops and remains low at step 10,000, the AMD clearly indicates that the first model is not valid anymore. Thus, the new data is used to train a new model. A similar process occurs for the following environment changes: the PSR with the highest p-value is trained, and the other is left idle. The AMD continues to track the probabilities of each model. Effectively, each chunk of data that is identified by the AMD as belonging to one model is used to train that model only and thus enables continual learning and prevents catastrophic forgetting.

As a baseline, we ran the constrained gradient PSR learning algorithm with the same parameters, but without AMD detecting switches in the environment. The agent learns well until time step 10,000. At time step 10,000–20,000, the agent also learns the second environment well, although it is not able to reach the same performance as the PSR with AMD. From time step 20,000 onwards, it is clear that the agent has experienced catastrophic forgetting, as each time the environment is switched, the performance decreases dramatically.

In these experiments, the error is calculated as the average prediction error (the difference between the predicted probabilities of the next observation and the actual probabilities of the next observation) over 10,000 time steps in an independent data stream.

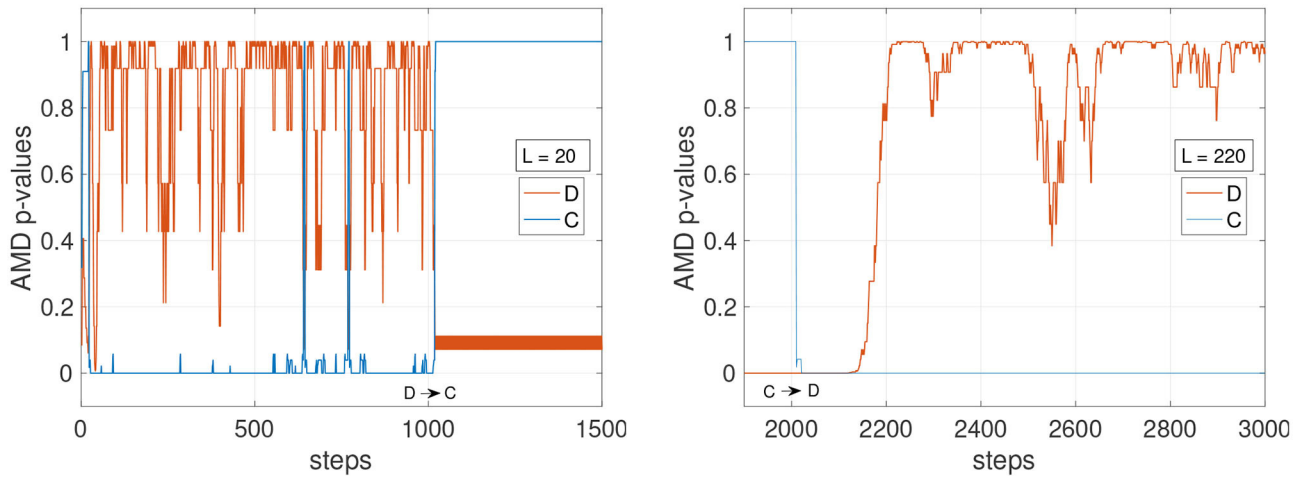

FIGURE 3 | AMD p-values for the models of environments C and D. **(Left)** With a short history length ($L = 20$), the p-values for the environment D occasionally indicate the data matches environment C, and not environment D. **(Right)** With a longer history length ($L = 220$), the p-value for model D becomes more stable, not dropping below 0.3.

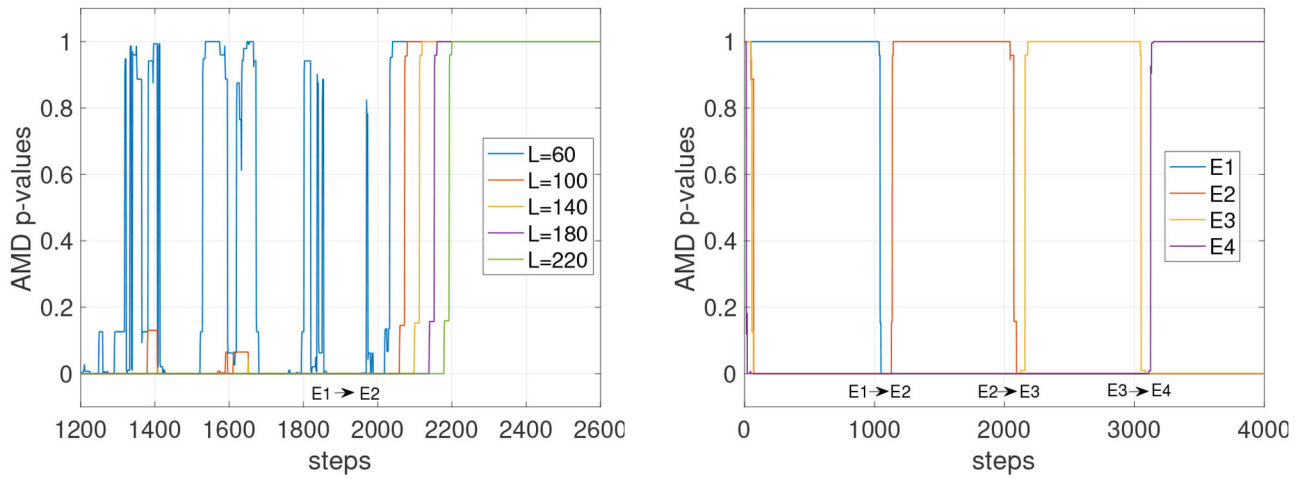

FIGURE 4 | Stability of detections for the deterministic set of models E. The agents traversing this environment use a random policy, where actions are chosen from the set of possible actions which can be taken at each state. **(Left)** The environment transitions from E1 to E2. The graph shows p-values for the environment E2. **(Right)** p-values for all four models are shown while all four environments alternate ($L = 220$).

4.3.2. AMD With Neural Network Learners

To validate the AMD with the neural network learning models, we employed a simple three-layer network whose details are specified in the **Appendix 6.1.2**. **Figure 8** shows the performance and p-values when the AMD is applied in combination with the neural network models.

We observe the same learning dynamics that were obtained with the PSR learner, although the neural network learner appears to achieve slightly better performance. The single model (baseline) shows the typical learning curves when different tasks or environments are learned sequentially, with catastrophic forgetting occurring each time the environment switches. The AMD instead can accurately determine when the environment switches and use the data to train two different models.

5. DISCUSSION

The results presented in the previous section show that the idea to use statistical tests to determine the best fitting model to label data is a promising venue of research. Various aspects of the algorithm and of the experimental results prompt interesting questions.

The first important aspect that was investigated in section 4.2, the impact of history length, shows that the readiness in detection and stability are opposed objectives that need to be balanced. However, stochasticity appears to be the main factor that requires

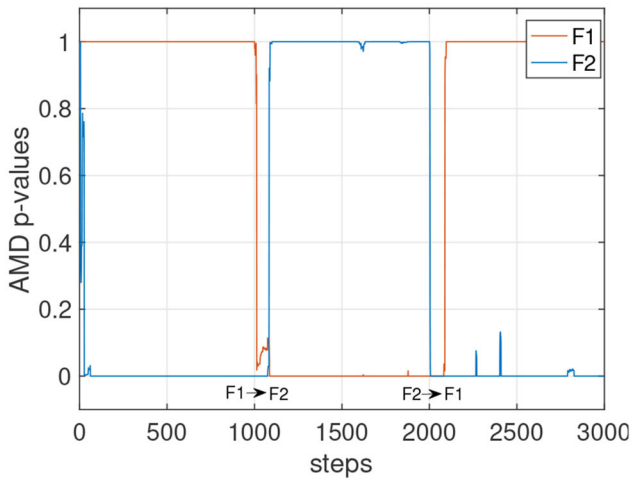

FIGURE 5 | p-values for the models of the set F with $L = 120$. Agents traversing this environment utilize a random policy.

longer histories to guarantee stability. We speculate that, while high levels of stochasticity are an obvious obstacle to learning, it would be possible to learn also such features of the environments and incorporate them in future developments of the AMD. One possibility is to introduce an adaptive history length that could reduce if an environment is predominantly deterministic, and increase in length when highly stochastic transitions require more data points to acquire meaningful statistics.

A second observation is that the dynamic properties of a series of POMDPs can include both sudden changes and slow progressive drifts. In the case of drifts that progressively increase the distance between the distributions of the model and the environment, there will be a race between the adaptation speed of a learning algorithm and the AMD p-values. If the learning algorithm is fast enough to track the drift, the AMD will maintain a high p-value, thus maintaining confidence in the current model. This condition, however, would lead to progressive forgetting of the original distribution. If, on the other hand, the drift in the environment is faster than the speed at which the learning algorithm can adapt, the AMD will see the corresponding p-value drop and either select a different model, or create a new model to learn the new data distribution. While we did not investigate these conditions, the problem of deciding whether an environment is drifting to a new distribution, or changing significantly to warrant the instantiation of a new model, is a relevant aspect of lifelong learning worth of future studies.

The AMD is intended as a framework that is independent of the specific learning algorithm used to learn a model. However, it is worth pointing out that the AMD is limited by the underlying learning model. In fact, while the p-values of sub-optimal models could be low, and thus lead to model rejection or further learning, there are cases in which this is not true, leading to simpler models having higher p-values than more accurate ones. In fact, a p-value could be high when the environment is more complex than the model. Consider, e.g., an environment that generates a data stream of observations $0, 0, 1, 1, 0, 0, 1, 1, \ldots$, where each observation 1 or 0 occurs twice in a row. A model that predicts that after each 1, the next observation will be 0 or 1 with equal probability will score a high p-value although a better model could be learned. In short, the AMD p-values might not always provide the best metrics to assess the quality of a model. While choosing the simplest model to fit the data might prove effective to prevent overfitting and agrees with the Occam's razor principle, further analysis might reveal how best to integrate the AMD algorithm with specific learning methods.

Given a set of models, one interesting question is what approach is more effective when a new model is necessary to learn a new environment. In the context of lifelong learning, a desirable property is that of exploiting previous knowledge to accelerate the learning of new tasks. It is possible that the AMD could facilitate such a forward transfer by instantiating a new model that matches some properties of the new data. While this problem was not touched in this study, the AMD may provide useful statistical insights to inform the creation of new models.

The set of problems proposed in this study appears simple at first. However, it is worth noting that partial observability and stochasticity make it difficult to derive the correct model from observations even in relatively simple environments. Additionally, the complexity of an environment might derive from a large input space, e.g., when using raw images in a navigation task. We speculate that the use of the AMD in combination with large neural models for feature extraction could allow the extension of this method to more complex problems.

Finally, it is important to note that this study introduces the idea of model selection from statistics in the domain of dynamic POMDPs without rewards. We could not identify existing methods that could be used for a direct performance comparison. However, with the addition of a reward function, this study could be extended to incorporate a policy component, and thus place the approach in the field of reinforcement learning. Given the large amount of research in reinforcement learning, this extension would open several exciting research directions and comparisons with recent RL and meta-RL approaches.

6. CONCLUSION

This study introduces an algorithm that aims to address a limitation of many current learning systems: the inability to monitor a non-stationary data stream while learning from it. The proposed system, named adaptive model detection (AMD), monitors the data stream generated by partially observable Markov decision processes with the aim to assess the probability of the data fitting a given model. Statistical tests determine (1) whether the null hypothesis that a current model produces the data can be accepted or rejected and (2) which specific model from a set is more accurate to predict a recent window of data. The novel algorithm was tested with

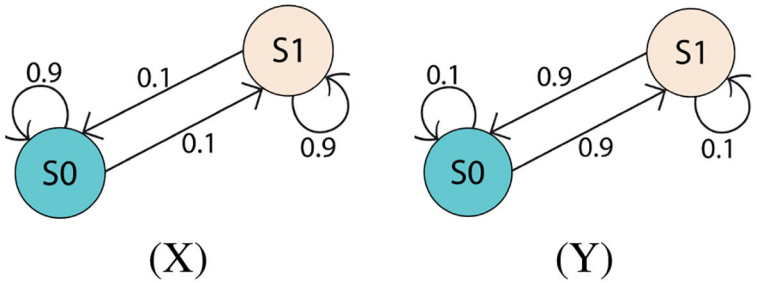

FIGURE 6 | Environments X and Y. These environments have only two states, but are stochastic. Environment X is more likely to remain in its current state at each time step, whereas environment Y is more likely to transition between states.

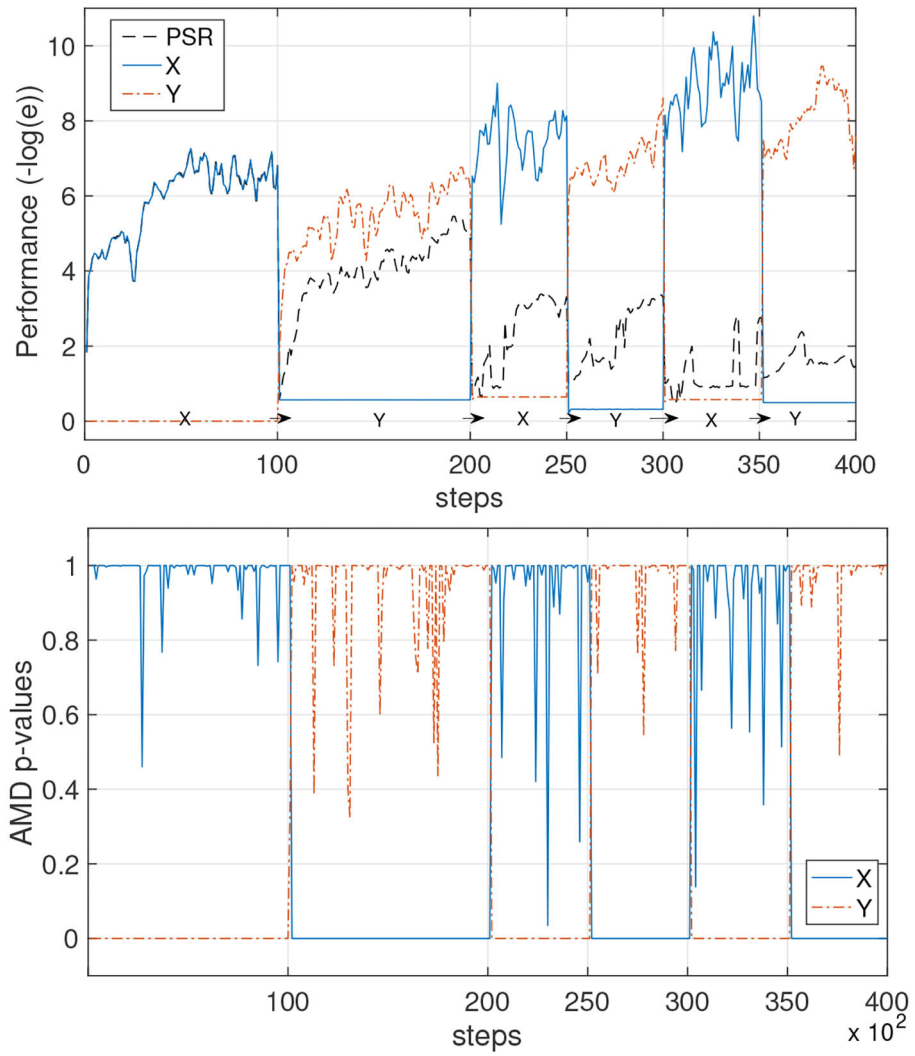

FIGURE 7 | Avoiding catastrophic forgetting with AMD on PSR learners. The unknown environments X and Y alternate overtime. **(Top)** The performance is measured as the negative log of the prediction error of each model. A single PSR (black line) is trained on the data stream and it learns an average of the two environments. The azure continuous line and the orange dash line show the AMD-guided learning: first X is learned, and when the data switches to environment Y, a new model Y is also learned. Subsequently, data points originating from the two environments are used to further improve each model separately. **(Bottom)** The AMD *p*-values for the AMD-guided learners are shown during the process.

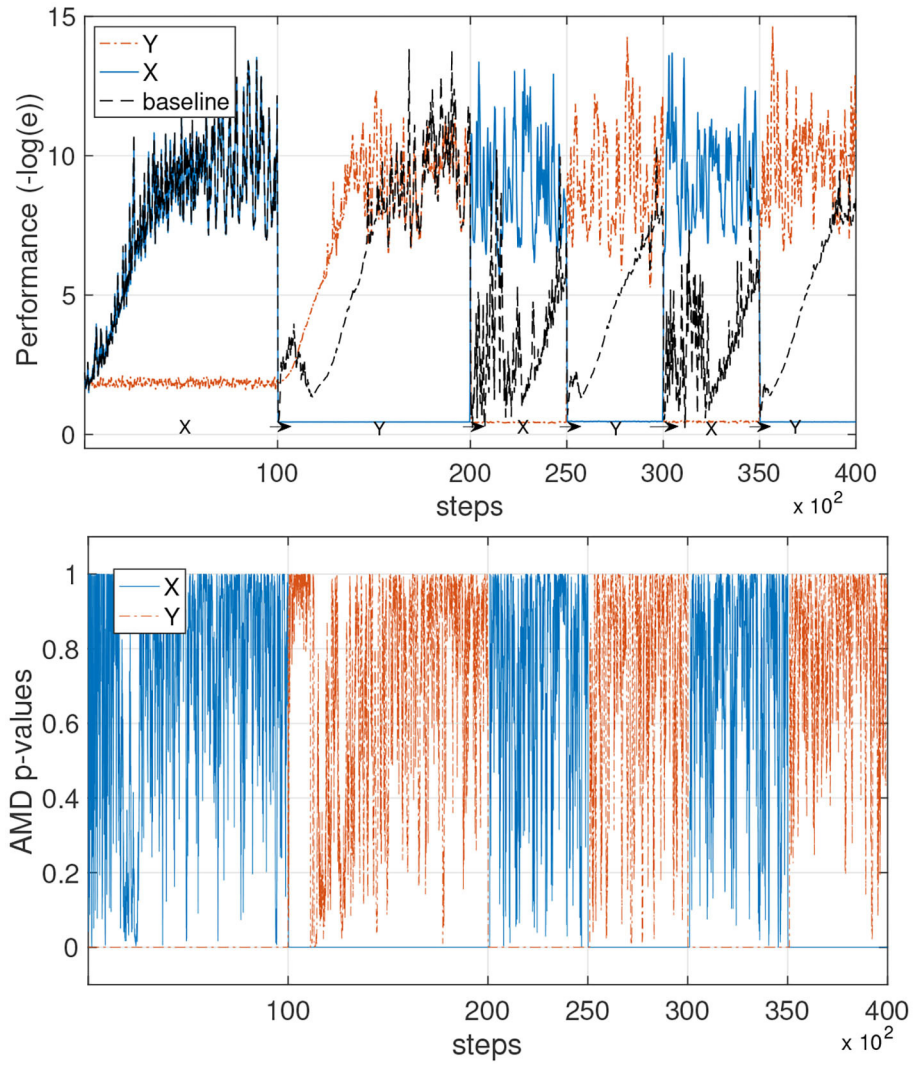

FIGURE 8 | Avoiding catastrophic forgetting with AMD on neural networks models. This experiment is the same as that in the previous figure, but uses neural network learners instead of the constrained gradient algorithm. **(Top)** The performance of the AMD-enabled models is shown in the environments X and Y. The baseline is capable of continuous learning, but forgets the previous environment each time there is a switch. **(Bottom)** The AMD *p*-values for the AMD-guided learner are shown during the process.

two types of predictive models, PSRs and neural networks. The simulations show that the approach is not only useful for quickly adapting to changes in an environment, but can also be useful to associate a stream of data to a particular environment. By doing so, it is possible to continuously train different models for different environments, and thus prevent catastrophic forgetting while learning multiple environments. The approach can be extended to address a wide set of problems beyond the limited scope of the environments tested here. The method could be valuable in AI applications where critical decisions require an evidence-based and justifiable process. When multiple environments are presented sequentially and require incremental learning without labels, rewards, or signals that a change has occurred, the approach presented can be used to implement continuous lifelong learning abilities.

DATA AVAILABILITY STATEMENT

Publicly available data were analyzed in this study. This data can be found here: https://github.com/JupiLogy/adaptive-model-detection.

AUTHOR CONTRIBUTIONS

JD developed the novel algorithm, wrote the computer code and performed the experiments. JD and AS devised the research plan and methods, analyzed the results, plotted the graphs and wrote the paper. PL and EB-I performed and analyzed experimental results. PP, SK, and PK contributed to the formulation of the research hypotheses. HS and SK provided support for the statistical method. All authors contributed to writing the final manuscript.

FUNDING

This material was based upon work supported by the United States Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-18-C-0103. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA).

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnbot.2020.578675/full#supplementary-material

REFERENCES

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Trans. Automat. Control* 19, 716–723. doi: 10.1109/TAC.1974.1100705

Bellman, R. (1957). A Markovian decision process. *Indiana Univ. Math. J.* 6, 679–684. doi: 10.1512/iumj.1957.6.56038

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press. doi: 10.1201/9781420050646.ptb6

Boots, B., Gretton, A., and Gordon, G. J. (2013). "Hilbert space embeddings of predictive state representations," in *Uncertainty in Artificial Intelligence - Proceedings of the 29th Conference, UAI 2013* (Bellevue, WA), 92–101.

Chuk, T., Chan, A. B., Shimojo, S., and Hsiao, J. H. (2020). Eye movement analysis with switching hidden Markov models. *Behav. Res. Methods* 52, 1026–1043. doi: 10.3758/s13428-019-01298-y

Cox, D. R. (2006). *Principles of Statistical Inference*. Cambridge: Cambridge University Press. doi: 10.1017/CBO9780511813559

Downey, C., Hefny, A., Li, B., Boots, B., and Gordon, G. (2017). "Predictive state recurrent neural networks," in *Advances in Neural Information Processing Systems*, eds I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Long Beach, CA: Curran Associates), 6054–6065.

Doya, K., Samejima, K., Katagiri, K.-I., and Kawato, M. (2002). Multiple model-based reinforcement learning. *Neural Comput.* 14, 1347–1369. doi: 10.1162/089976602753712972

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96* (Portland, OR: AAAI Press), 226–231.

Finn, C., Abbeel, P., and Levine, S. (2017). "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning* (Sydney, NSW).

Geisser, S. (1993). *Predictive Inference*, Vol. 55. New York, NY: CRC Press.

Hamilton, W. L., Fard, M. M., and Pineau, J. (2013). "Modelling sparse dynamical systems with compressed predictive state representations," in *30th International Conference on Machine Learning, ICML 2013* (Atlanta, GA), 178–186.

Höffken, M., Oberhoff, D., and Kolesnik, M. (2009). "Switching hidden Markov models for learning of motion patterns in videos," in *Lecture Notes in Computer Science*, eds C. Alippi, M. Polycarpou, C. Panayiotou, and G. Ellinas (Limassol: Springer, Berlin, Heidelberg), 757–766. doi: 10.1007/978-3-642-04274-4_78

Kochenderfer, M. J., Amato, C., Chowdhary, G., How, J. P., Davison Reynolds, H. J., Thornton, J. R., et al. (2015). "Optimized airborne collision avoidance," in *Decision Making Under Uncertainty: Theory and Application* (MIT Press), 249–276.

Kullback, S. (1997). *Information Theory and Statistics*. New York, NY: Courier Corporation.

Lecarpentier, E., and Rachelson, E. (2019). "Non-stationary Markov decision processes, a worst-case approach using model-based reinforcement learning," in *Advances in Neural Information Processing Systems 32*, eds H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Vancouver, BC: Curran Associates, Inc.), 7216–7225.

Lehmann, E. L., and Romano, J. P. (2006). *Testing Statistical Hypotheses*. New York, NY: Springer Science & Business Media.

Littman, M. L., Sutton, R. S., and Singh, S. (2001). "Predictive representations of state," in *Advances in Neural Information Processing Systems*, eds T. Dietterich and S. Becker and Z. Ghahramani (Vancouver, BC: MIT Press), 1555–1561.

Liu, Y., Zhu, H., Zeng, Y., and Dai, Z. (2016). "Learning predictive state representations via Monte-Carlo tree search," in *IJCAI International Joint Conference on Artificial Intelligence* (New York, NY), 3192–3198.

McCracken, P., and Bowling, M. (2005). "Online discovery and learning of predictive state representations," in *Advances in Neural Information Processing Systems*, eds Y. Weiss, B. Schölkopf, and J. Platt (Vancouver, BC: Curran Associates), 875–882.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. (2018). "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (Brisbane, QLD), 7559–7566. doi: 10.1109/ICRA.2018.8463189

Pearson, K. (1900). On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Lond. Edinb. Dubl. Phil. Mag.* 50, 157–175. doi: 10.1080/14786440009463897

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* 12, 2825–2830. doi: 10.5555/1953048.2078195

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE* 77, 257–286. doi: 10.1109/5.18626

Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. (2019). "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *International Conference on Machine Learning* (Long Beach, CA), 5331–5340.

Rao, R. P. (2010). Decision making under uncertainty: a neural model based on partially observable Markov decision processes. *Front. Comput. Neurosci.* 4:146. doi: 10.3389/fncom.2010.00146

Rosencrantz, M., Gordon, G., and Thrun, S. (2004). "Learning low dimensional predictive representations," in *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004* (Banff, AB), 695–702. doi: 10.1145/1015330.1015441

Rothfuss, J., Lee, D., Clavera, I., Asfour, T., and Abbeel, P. (2018). "Promp: Proximal meta-policy search," in *International Conference on Learning Representations* (Vancouver, BC).

Schmidhuber, J. (2015). Deep learning in neural networks: an overview. *Neural Netw.* 61, 85–117. doi: 10.1016/j.neunet.2014.09.003

Schwarz, G. (1978). Estimating the dimension of a model. *Ann. Stat.* 6, 461–464.

Shani, G., Brafman, R. I., and Shimony, S. E. (2005). "Model-based online learning of POMDPs," in *Proceedings of 16th European Conference on Machine Learning*, eds J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo (Porto; Berlin; Heidelberg: Springer), 353–364. doi: 10.1007/11564096_35

Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Thrun, S. (1998). "Lifelong learning algorithms," in *Learning to Learn*, eds S. Thrun, and L. Pratt (Boston, MA: Springer), 181–209. doi: 10.1007/978-1-4615-5529-2_8

Todd, M. T., Niv, Y., and Cohen, J. D. (2009). "Learning to use working memory in partially observable environments through dopaminergic reinforcement," in *Advances in Neural Information Processing Systems*, eds D. Koller and D. Schuurmans and Y. Bengio and L. Bottou (Vancouver, BC: Curran Associates), 1689–1696.

Wingate, D., and Singh, S. (2007). On discovery and learning of models with predictive representations of state for agents with continuous actions and observations. *Proc. Int. Conf. Auton. Agents* 5, 1136–1143. doi: 10.1145/1329125.1329352

Yates, D., Moore, D., and McCabe, G. (1999). *The Practice of Statistics*. New York, NY: H. Freeman & Company.

Yun-Long, L., and Ren-Hou, L. (2009). Discovery and learning of models with predictive state representations for dynamical systems without reset. *Knowledge Based Syst*. 22, 557–561. doi: 10.1016/j.knosys.2009.01.001

Zintgraf, L. M., Shiarlis, K., Kurin, V., Hofmann, K., and Whiteson, S. (2019). *Fast Context Adaptation via Meta-Learning. Long Beach*, CA: ICML.

Conflict of Interest: SK and PP were employed by HRL Laboratories.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.