A Robust Cognitive Architecture for Learning from Surprises

Thomas Joseph Collins and Wei-Min Shen

Information Sciences Institute, University of Southern California, Los Angeles, CA, USA
`collinst@usc.edu` and `shen@isi.edu`

Abstract
Learning from surprises is a cornerstone for building bio-inspired cognitive architectures that can autonomously learn from interactions with their environments. However, distinguishing true surprises – from which useful information can be extracted to improve an agent's world model – from environmental noise is a fundamental challenge. This paper proposes a new and robust approach for actively learning a predictive model of discrete, stochastic, partially-observable environments based on a concept called the Stochastic Distinguishing Experiment (SDE). SDEs are conditional probability distributions over the next observation given a variable-length sequence of ordered actions and expected observations up to the present that partition the space of possible agent histories, thus forming an approximate predictive representation of state. We derive this SDE-based learning algorithm and present theoretical proofs of its convergence and computational complexity. Theoretical and experimental results in small environments with important theoretical properties demonstrate the algorithm's ability to build an accurate predictive model from one continuous interaction with its environment without requiring any prior knowledge of the underlying state space, the number of SDEs to use, or even a bound on SDE length.

Keywords: active learning, prediction, surprise-based learning

1 Introduction

The ability to learn from *surprises*, which are mismatches between expected and actual sensory feedback in response to actions, is a cornerstone for building bio-inspired cognitive architectures that can learn autonomously from interactions with unknown environments. Theories and experiments in child psychology [19, 9] and even the philosophy of science [18] have emphasized the importance of surprises in the learning process. Experiments with AI agents that learn from surprises in biology (gene discovery) [26, 31], game playing [29], learning from large knowledge bases [28], learning on real robots in response to unexpected changes [21], and learning to detect and recover from interference and gaps in sensory information [22] have demonstrated the power of such a biologically-inspired approach. However, these existing techniques use logic-based rules to predict the results of future observations and modify their models in response to each *prediction failure*, which is a single mismatch between expected and actual observation. This leads to severe overfitting in noisy environments. Additionally, hidden-state detection and modeling [25] relies on properties that are only satisfied in deterministic environments.

In this paper, we adopt a probabilistic approach that broadens the definition of a *surprise* to changes in the frequencies of individual prediction failures, leading to a novel and robust approach for actively learning a predictive model of discrete, stochastic, partially-observable environments (represented as Partially-observable Markov Decision Processes (POMDPs)) based on a novel concept called Stochastic Distinguishing Experiments (SDEs), which are probability distributions over the next observation given a variable-length ordered sequence of actions and expected observations up to the present. SDEs are simultaneously used to partition the space of possible histories the learning agent could encounter (thus forming an approximate predictive representation of state) and discover structure in sequences of actions and observations that allow the agent to predict future observations accurately. We derive this learning algorithm and present theoretical proofs of its convergence and computational complexity. Theoretical and experimental results demonstrate the algorithm's ability to build an accurate predictive model from one continuous interaction with its environment without requiring prior knowledge of the underlying state space, the number of SDEs needed, or even a bound on the number of actions and observations in any SDE.

2 Related Work

Much learning work regarding discrete PODMPs is in the area of reinforcement learning (RL), where the goal is not to construct a state representation but rather to learn an optimal policy given a known state space. Traditional exact and approximate techniques include [17, 20, 24]. SDEs share similarities with instance-based (IB) RL methods [13, 11, 14, 15], which memorize interactions with their environment and organize them into a suffix tree of actions and observations that approximates an unknown state space nonparametrically. SDE learning requires no memorization of instances (reducing required memory and avoiding the difficult issue of determining sufficient memory size), and, in contrast to notable IB methods (e.g., [35]), does not require separate training episodes or an externally-generated reward function.

There exist techniques for growing a latent (unobserved) state space for unknown POMDPs based on an agent's interaction with its environment that do not require specifying the number of states as prior information, most notably the infinite Partially-observable Markov Decision Process (iPOMDP) [6]. Like other Bayesian approaches, specifying priors in iPOMDPs may be difficult in unknown environments, and there are strong modeling assumptions about these priors. One crucial difference between iPOMDPs and SDEs is that SDEs do not require an external reward function for action selection and learning. Additionally, as Littman notes in [10], predictive models (such as SDEs) built in terms of an agent's actions and quantities it can directly observe have the potential to be easier to learn and generalize better.

Predictive state representations (PSRs) [10] model the state of POMDPs in terms of the probabilities of *tests* (experiments), which consist of ordered actions and expected observations. For every controlled dynamical system, there exists a linearly independent set of tests to represent its state (i.e., is a sufficient statistic). Algorithms exist for finding a set of tests that forms a PSR [34, 16] and for learning the parameters of a set of tests forming a PSR [33, 34, 16], which generally must be performed separately. Traditional algorithms for finding a set of tests forming a PSR generally proceed in a passive guess-and-check fashion, where subsets of potential tests are examined for linear independence. More recent techniques [2, 8] essentially circumvent the issue of learning the set of tests by maintaining a large (likely redundant) set of tests or resort to an offline, stochastic search [12]. In contrast, SDEs, which also build a model from the probabilities of experiments an agent can perform, can be selected and their parameters learned in an integrated fashion. Additionally, SDEs need not be linearly independent because they partition possible histories, saving an expensive linear independence check at each step. This also makes it easier to incrementally grow larger SDEs from smaller ones or merge SDEs together as the agent's experience indicates a more (or less) complex representation is needed.

Finally, we note that SDEs are related to some techniques in the compression literature, such as Variable Order Markov Models (VMMs) for prediction [1, 4] and recent extensions of these methods to controlled processes [5] for online Bayesian inference and decision making. Traditional VMM approaches do not consider actions and the extensions to controlled processes select actions based on an externally-generated reward function, rather than actively generating actions to try to improve the agent's model of the world (as in SDE learning). Additionally, the priors used in Bayesian approaches such as [5] may be difficult to set or even detrimental when the environment is completely unknown, as is the case in this work.

In summary, SDE learning has several key advantages in learning discrete POMDPs: 1) SDE learning is active, with the agent selecting and performing experiments in its environment to look for surprises that can be analyzed to provide crucial information about improving its model of the world; 2) SDE learning does not require specifying prior distributions or knowledge about the underlying state space, the number of SDEs, or a bound on SDE length; 3) SDE learning requires no external reward function because increases in estimated predictive model accuracy are used as a self-generated reward/goal; 4) SDEs can be learned via a single sequence of interactions with the environment (with no separate training episodes required).

3 Stochastic Distinguishing Experiments (SDEs)

Stochastic Distinguishing Experiments (SDEs) are an extension of Shen's Local Distinguishing Experiments (LDEs) [25] for deterministic, partially-observable environments to stochastic environments. LDEs are ordered sequences of actions and expected observations that disambiguate states sharing the same observation.

As an example, consider the shape environment first presented in [23] and shown in Figure 1, in which an autonomous agent can perform actions x and y and receive observations *square* (in states I and II) and *diamond* (in states III and IV). The agent has no knowledge of the underlying state space or the consequences of its actions. In the deterministic version of this environment, states I and II can be disambiguated by observing *square*, taking action y, observing *diamond*, and, finally, taking action x, which, from state I, leads to a final observation of *square* and, from state II, leads to a final observation of *diamond*. Thus, the LDE $\{$*square*, y, *diamond*, $x\}$, informs the agent which of the two states with observation *square* it was at before it performed the LDE. The CDL algorithm [27] incrementally builds a predictive model consisting of the LDEs and transitions between them necessary to distinguish all hidden states. These LDEs form a predictive representation of the state space.

When learning a set of LDEs, CDL assumes that executing the same trajectory from the same state any number of times will result in the same final observation and uses the historical differences in trajectories originating in different states to create and modify LDEs. Clearly, stochastic transitions break this fundamental assumption because executing the same trajectory multiple times may result in different observations. Stochastic Distinguishing Experiments (SDEs) are designed to overcome this difficulty by modeling a probability distribution over

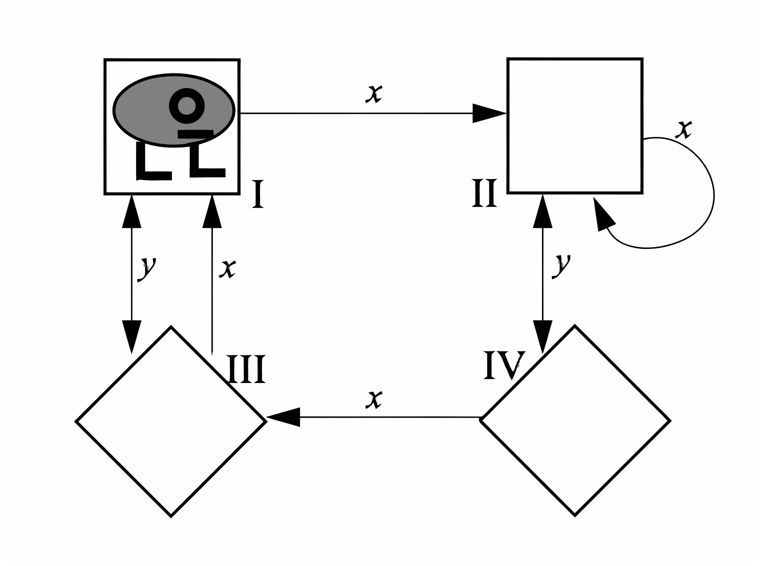


Figure 1: The shape environment, reproduced with kind permission from Shen [30].

final observations. Intuitively, the more peaked this distribution is on one observation, the more like an LDE it is, and, thus, the more useful it is for localizing the agent in its environment so that it can predict future observations accurately. Formally, an SDE is a conditional probability distribution over the next observation given a finite, variable-length ordered sequence of actions and expected observations up to the present. A k-action *simple SDE*, denoted v_k, is defined as follows:

$$v_k \triangleq P(O_{t+1} | a_{t-k+1}, o_{t-k+2}, ..., a_{t-1}, o_t, a_t) \tag{1}$$

The sequence of actions and observations behind the conditioning bar is called the *experiment* of the SDE, denoted e_{v_k}, and we say that e_{v_k} covers trajectories through the POMDP ending in the (ordered) k actions and $k-1$ observations matching those in e_{v_k}. k is called the *length* of the SDE and its experiment. We say that an experiment *succeeds* if executing its actions in order results in the expected ordered sequence of observations up to time t. SDEs can also have *compound experiments*, in which case the agent has a choice of actions to perform at one or more time-steps, with each choice corresponding to a set of one or more possible expected observations at the next time-step. Such an SDE is called a *compound SDE*.

As a specific example, consider a stochastic version of the shape environment in Figure 1 in which the transitions marked by the arrows have probability 0.925, and the remaining probability is divided equally among transitions to the other 3 states. An example 2-action simple SDE generated using the SDE learning algorithm presented in the next section is:

$$P(O = \{square, diamond\} | y,\ diamond,\ x) = \{0.386, 0.614\} \tag{2}$$

In words, in this environment, the probability that the observation at time $t+1$ is *square* given that the agent's history ends by executing the action y at time $t-1$, observing *diamond* at time t, and executing action x at time t is 0.386. The probability of observing *diamond* is likewise 0.614. A 4-action compound SDE from a different execution of the algorithm is:

$$P(O = \{square, diamond\}|y,\ diamond,\ \{x, y, y\},\\ \{diamond, square, diamond\},\ y,\ diamond,\ x) = \{0.308, 0.692\} \quad (3)$$

The semantics of this SDE are similar except that, at time $t-2$, the agent can either perform action x and observe *diamond* at time $t-1$, perform action y and observe *square* at time $t-1$, or perform action y and observe *diamond* at time $t-1$.

4 Learning Stochastic Distinguishing Experiments

Algorithm 1: Pseudocode for the SDE learning algorithm proposed in this work

```
Input:
A: a discrete set of actions the agent can perform
O: a discrete set of observations the agent can observe
sgain, rgain, mgain: gain parameters
numExp: number of experiments to perform
convergeTol: number of times each SDE must try splitting before convergence
Output:
V: a set of stochastic distinguishing experiments (SDEs)
1  Function LearnSDEs()
2      $V := \{P(O|null)\}$;
3      $successCounts = \{P(O|null) : 0\}$;
4      while $successCounts[v] < convergeTol$ for one or more $v \in V$ do
5          select random SDE, $v$, with $successCounts[v] < convergeTol$;
6          if $v$ is a compound SDE with multiple possible first actions then
7              $v$ := RefineBySurprise($v$, $rgain$) /* Section 4.3 */
8          for $o \in \{V - v\}$ do
9              $\{didMerge, v\}$ := MergeBySurprise($o$, $v$, $mgain$) /* Section 4.4 */
10             if $didMerge$ then
11                 break;
12         $E$ := getExperiments($v$) /* Section 4.1 */
13         $oneSteps$ := performExperiments($E$, $numExp$) /* Section 4.1 */
14         $didSplit$ := SplitBySurprise($v$, $oneSteps$, $sgain$) /* Section 4.2 */
15         if $didSplit == false$ then
16             $successCounts[v] = successCounts[v] + 1$
```

Let agent $\mathcal{M}$ be a learning agent placed in a discrete, stochastic, partially-observable environment, represented as POMDP $\mathcal{E} = \{S, A, O, T, \gamma\}$, where S is a set of states, A is a set of actions, O is a set of observations, $T^a_{s,s'}$ are the transition probabilities, and γ^o_s are the observation probabilities. Rewards are not considered in this work. $\mathcal{M}$ has no initial knowledge of the consequences of actions, nor does it know anything about T, γ, or S. Algorithm 1 gives the high-level pseudocode of the algorithm, which builds a predictive model of SDEs that partitions the space of possible agent histories, forming an approximate predictive representation of $\mathcal{E}$'s state. Intuitively, statistics on SDEs and the frequencies of *observations* and *prediction failures* are gathered via *Active Experimentation* (Section 4.1), leading to *surprises* that modify the predictive model via three key operations: *Splitting* (Section 4.2), which increases the length and number of SDEs in the model to increase estimated predictive accuracy; *Refining* (Section 4.3), which separates out individual trajectories with significantly different probability distributions from compound SDEs into their own simple SDEs, increasing the number and specialization of SDEs of the same size; and *Merging* (Section 4.4), which reduces the number of SDEs in the model by combining similar SDEs to attempt to create as compact of a model as possible.

The algorithm consists of a while loop (line 4) that runs until no SDE has a *successCount* less than $convergeTol$. The *successCount* of each SDE is incremented when the SDE Splitting procedure (line 14) fails to find sufficient evidence that increasing the length of v will increase the model's predictive accuracy. In line 5, a random SDE v with $successCounts[v] < convergeTol$ is selected for experimentation. If v is compound with multiple choices of action at the first step (line 6), line 7 performs SDE Refinement to separate out individual trajectories covered by v into their own simple SDEs, allowing individual trajectories that behave statistically differently to be treated separately in later operations. Next, the SDE Merging procedure (lines 8-11) searches through the other SDEs in the model and attempts to merge v with a statistically similar SDE to make the model more compact. In line 12, we expand k-action SDE v into a set of $|A||O|$ SDEs with length $k+1$ (*oneSteps*) by appending each possible pair of action and observation to the beginning of v's experiment and perform the experiments of these *oneSteps* uniformly at random *numExp* times (line 13), actively predicting and using the results of these experiments to estimate the conditional observation probabilities and predictive accuracies of the SDEs in *oneSteps* and, when possible, using sub-trajectories of those generated by experimentation to update estimates of these quantities for the SDEs currently in V. Finally, in line 14, after experimentation has been completed, SDE Splitting is used to partition k-action SDE v into a set of two or more $k+1$-action SDEs (created by grouping its *oneSteps* together according to predicted observation) when such a split causes an increase in estimated predictive accuracy.

4.1 Active Experimentation and Prediction

The key idea of SDE learning is that the learning agent continuously designs new experiments, predicts the results of these experiments, and performs these experiments in its environment, comparing the actual result of the experiment to the predicted result and gathering statistics about the frequencies of observations and *prediction failures* (mismatches between actual and predicted experimental results). These statistics generate surprises that are used to update the agent's world model via Splitting (Section 4.2), Refining (Section 4.3) or Merging (Section 4.4) in a way that increases the estimated predictive accuracy of the model or allows for further specialization of the model along specific agent histories. In line 12 of Algorithm 1, the function *getExperiments* returns the $|A||O|$ *one-step extension* experiments (each with length $k+1$ and stored in the list E) formed by appending each possible pair of a single action and observation to the beginning of v's k-length experiment. The exception to this is that, at the beginning of the algorithm, only one SDE with a *null* experiment exists (which trivially covers all trajectories and tracks the prior distribution over observations), and the one step extension experiments are formed by appending each possible action to this *null* experiment. SDEs are created for each one-step extension experiment and stored in the list *oneSteps* returned by $performExperiments$ (line 13). For example, the one step extension experiments for the SDE in equation 2 would be:

1. $\{x$, *square*, y, *diamond*, $x\}$

2. $\{x, \textit{diamond}, y, \textit{diamond}, x\}$
3. $\{y, \textit{square}, y, \textit{diamond}, x\}$
4. $\{y, \textit{diamond}, y, \textit{diamond}, x\}$

Note that these four one-step extension experiments do not cover all possible trajectories that can occur by the agent executing action x or y at time $t-2$ followed by action y at time $t-1$ and x at time t. In line 13, the agent performs *numExp* of these experiments in its environment uniformly at random, predicts the resulting observation at time $t+1$, and uses the actual resulting observation to update the observation and prediction counters associated with the matching SDE in *oneSteps* (if such a matching SDE exists for the generated trajectory). The estimated conditional probability of each observation given the success of the SDE's experiment is the value of its observation counter divided by the total number of the observation counters of all observations (a categorical distribution), while its *estimated predictive accuracy* is the number of times it successfully predicted the next observation divided by the total number of times it was used for prediction (each time the SDE's experiment succeeded). Regardless of whether or not the trajectory generated by each such experiment matches one of the *oneSteps*, sub-trajectories of this trajectory from length 0 (just observations) to length k are used to make predictions and update the observation counters and prediction counters of the SDEs currently in the model V, allowing parameter learning of existing SDEs in V to continue opportunistically, even when they are not being explicitly evaluated. As discussed next, the agent always predicts the observation with the highest conditional probability in the SDE whose experiment succeeded.

4.2 SDE Splitting

The key idea behind SDE splitting is that the learning agent should be surprised when the one-step extension SDEs of the selected SDE, taken together, have greater estimated predictive accuracy than the selected SDE, because it means that increasing the length of the experiment by one provides significantly more information about the next observation. Intuitively, this indicates to the learning agent that its model is too coarse along the possible agent trajectories covered by the selected SDE. Assume, for the moment, that agent $\mathcal{M}$ is given the probabilities of every possible trajectory of actions and observations through POMPD $\mathcal{E}$. In this case, $\mathcal{M}$ can easily calculate the conditional probability distribution associated with any k-length simple or compound SDE for any length k. Let X_{v_k} be a binomial random variable whose value represents the number of successful predictions of SDE v_k in n independent trials in which v_k's experiment succeeds. We wish to maximize $E[X_{v_k}] = np_{v_k}$ (the expected number of correct predictions) for any n, which occurs when the agent always predicts the most likely observation:

$$p_{v_k} \triangleq \max_{o_{t+1} \in O} P(o_{t+1} | a_{t-k+1}, o_{t-k+2}, ..., a_{t-1}, o_t, a_t) \tag{4}$$

p_{v_k} is just the probability of the most likely observation given that v_k's experiment succeeded and is called the *predictive accuracy* of v_k because, as $n \to \infty$, the empirical fraction of successful to total predictions made by the agent will converge to p_{v_k}. Note that this is the same as minimizing the expected number of *prediction failures*, in which the actual result of the experiment does not match the predicted result. Note also that we use the definition of a simple SDE for notational simplicity, but compound SDEs can also be split as described here. We wish to decide whether or not replacing k-action SDE v_k with its $|A||O|$ one-step extension SDEs $v_{i,k+1}$ (each with length $k+1$) will result in an increase in predictive accuracy.

The set of one-step extension SDEs covers the same trajectories as the original SDE in a mutually exclusive and exhaustive fashion but allows for specialization (differing predictions) amongst sub-trajectories that had the same prediction in v_k. Defining $p_{v_{i,k+1}}$ analogously to p_{v_k} for all one step extension SDEs, and defining $w_{i,k+1}$ as the conditional probability of one-step extension $v_{i,k+1}$'s experiment succeeding given that v_k's experiment succeeded, an increase in predictive accuracy occurs when:

$$(\sum_{i=1}^{|A||O|} p_{v_{i,k+1}} w_{i,k+1}) > p_{v_k} \tag{5}$$

We do not know p_{v_k}, $p_{v_{i,k+1}}$ or $w_{i,k+1}$. However, p_{v_k} can be estimated using the prediction counters of v_k in model V, $p_{v_{i,k+1}}$ can be estimated using the prediction counters of the one-step extension SDEs (*oneSteps*), and we can estimate $w_{i,k+1}$ as the fraction of times one-step SDE $v_{i,k+1}$'s experiment succeeded divided by the total number of times any one-step SDE's experiment succeeded during the active experimentation procedure described in the previous section. A *splitting surprise* occurs when:

$$\frac{\sum_{i=1}^{|A||O|} \hat{p}_{v_{i,k+1}} \hat{w}_{i,k+1}}{\hat{p}_{v_k}} > sgain \tag{6}$$

Where *sgain* is an algorithm parameter whose value is greater than or equal to 1 and the hats indicate that we have substituted in our empirical estimates for the actual probabilities. When a splitting surprise occurs, v_k is replaced in V with at most $|O|$ compound SDEs formed by grouping its one-step extension SDEs according to predicted observation. v_k is only eligible for splitting if *all* of its one-step extension experiments succeed *at least once* during experimentation. As a specific example, consider the following SDE generated using the proposed learning algorithm:

$$P(O = \{square, diamond\}|x) = \{0.777, 0.223\} \tag{7}$$

and its one-step extensions:

$$P(O = \{square, diamond\}|x,\ square,\ x) = \{0.942, 0.058\} \tag{8}$$

$$P(O = \{square, diamond\}|x,\ diamond,\ x) = \{0.913, 0.087\} \tag{9}$$

$$P(O = \{square, diamond\}|y,\ square,\ x) = \{0.943, 0.057\} \tag{10}$$

$$P(O = \{square, diamond\}|y,\ diamond,\ x) = \{0.368, 0.632\} \tag{11}$$

The estimated probability distribution over each of these one-step extensions occurring given that the original SDE's experiment succeeded ($\hat{w}_{i,k+1}$) is $\{0.409, 0.136, 0.209, 0.246\}$, which causes a *splitting surprise* for $sgain = 1.1$ because

$$\frac{\sum_{i=1}^{|A||O|} \hat{p}_{v_{i,k+1}} \hat{w}_{i,k+1}}{\hat{p}_{v_k}} = \frac{0.862}{0.777} = 1.11$$

Splitting v_k according to this splitting surprise would lead to it being replaced by two new SDEs in model V:

$$P(O=\{square, diamond\}|\{x,x,y\},\ \{square, diamond, square\},\ x) = \{\{0.942, 0.058\}, \{0.913, 0.087\}, \{0.943, 0.057\}\} \tag{12}$$

$$P(O=\{square, diamond\}|y,\ diamond,\ x) = \{0.368, 0.632\} \tag{13}$$

Notice that the individual probability distributions are kept associated in the compound SDEs with the appropriate trajectory for now, which is crucial for the refining operation discussed next, which attempts to separate out these trajectories from compound SDEs when they begin to behave statistically differently. However, only one set of prediction counters is kept. The *successCounts* of these new SDEs are set to 0. When a compound SDE with multiple choices of action at the first step, such as in equation 12, is split, a normalized average over each observation over all the combined trajectories is used in the above procedure. The *sgain* parameter controls how much of a predictive accuracy increase justifies making the current SDE one step longer, as longer SDEs are more difficult to estimate from data. Finally, we note that splitting is the only procedure by which the length of SDEs increases.

4.3 SDE Refining

The key idea behind SDE refining is that the learning agent should be surprised when different trajectories covered by the same compound SDE have significantly different conditional probability distributions over the next observation. Refining a k-length compound SDE with multiple choices of action at the first time step refers to the process of separating out one of its trajectories into its own k-length simple SDE, reducing the number of trajectories covered by the original compound SDE. Note that the grouping done by the splitting procedure discussed in the previous section ensures that the observation with the maximum probability will be the same for all probability distributions in the compound SDE.

Let p_{avg} denote the average of these maximum probability values over all trajectories in the compound SDE. In equation 12, for example, $p_{avg} = 0.933$. Let p_{max} be the individual probability value with the largest absolute value difference from p_{avg}. In the case of equation 12, $p_{max} = 0.913$. A *refinement surprise* is defined as the event that

$$\max(\frac{p_{max}}{p_{avg}}, \frac{p_{avg}}{p_{max}}) > rgain \tag{14}$$

Where $rgain \geq 1$ is a refinement gain parameter. If $rgain = 1.02$, the above example results in a *refinement surprise*, and the SDE in equation 12 would be replaced in V with the following two SDEs (with their *successCounts* set to 0) formed by refining out the middle trajectory:

$$P(O=\{square, diamond\}|\{x,y\},\ \{square, square\},\ x) = \{\{0.942, 0.058\}, \{0.943, 0.057\}\} \tag{15}$$

$$P(O=\{square, diamond\}|x,\ diamond,\ x) = \{0.913, 0.087\} \tag{16}$$

Note that refinement increases the number of SDEs in the model by 1 and that the new SDEs it creates have the same length as the original SDE. Furthermore, we note that refinement has no immediate impact on the model's estimated predictive accuracy but rather allows for increased specialization of the refined out trajectory later in the algorithm.

4.4 SDE Merging

The key idea behind SDE merging is that the learning agent should be surprised when the conditional probability distributions represented by two different SDEs start to behave similarly. In contrast to splitting, this indicates to the agent that its model may be unnecessarily complex along certain trajectories and represents an opportunity to make the model more compact. Merging reduces the size of predictive model V by 1 SDE by combining two simple or compound SDEs into a single SDE that covers the trajectories covered by both of the original SDEs. Two SDEs are only eligible to be merged if their experiments differ only in the first action (at time $t-k+1$) and observation (at time $t-k+2$) pair and if they predict the same observation.

Let v_1 and v_2 be two SDEs that meet the above eligibility criteria. Let $p_{max,1}$ be the probability of the maximum observation in v_1 (possibly averaged over multiple distributions if compound) and let $p_{max,2}$ be the probability of the maximum observation in v_2 (possibly averaged over multiple distributions). A *merging surprise* occurs when:

$$\max(\frac{p_{max,1}}{p_{max,2}}, \frac{p_{max,2}}{p_{max,1}}) < mgain \qquad (17)$$

Where $mgain \geq 1$ is a merging gain parameter. Merging may be considered an optional procedure in this SDE learning algorithm because, while it makes the model smaller (which is important in many applications), it may do so at the expense of the model's predictive accuracy. Finally, we note that it is possible that all $|A||O|$ one-step extension SDEs are merged together after being split and/or refined out. In this case, the probabilities of each observation are averaged and normalized into a single probability distribution, and the first action and observation of the experiment is dropped, returning the merged SDE to a k-length SDE rather than a $k+1$-length one. Though it is not explicitly mentioned in Algorithm 1, the pairs of experiments involved in all splitting, refining, and merging operations performed are saved (via a hash map) and, before performing any new operations, the algorithm ensures that the same operation has not been performed before. This prevents possible infinite loops of merging and splitting or merging and refining the same SDEs. As an example, merging the SDEs in equations 12 and 13 would result in the SDE in equation 7 but with a different probability distribution: $\{0.792, 0.208\}$, which is obtained by averaging the 4 probability values associated with each observation and ensuring the resulting distribution is normalized. The $successCount$ of the merged SDE is set to 0.

5 Convergence and Computational Complexity

In this section, we prove that the proposed SDE learning algorithm converges in finite time without any user-defined explicit limit on the length of SDEs and provide an analysis of its computational complexity. These proofs rely on a result from probability theory called the Borel-Cantelli lemma [7, 3] that says that if $G_1, G_2, ...$ is a sequence of events in a probability space and the sum of their probabilities is finite (i.e., $\sum_{n=1}^{\infty} P(G_n) < \infty$), then the probability that infinitely many of these events occur is 0. These results require that no observation probabilities in POMDP $\mathcal{E}$ be fully deterministic. We also limit our analysis to environments with more than one possible action and more than one possible observation.

As an example of the Borel-Cantelli Lemma (based on [32]), consider the set of Bernoulli random variables $\{X_n : n \geq 1\}$ such that $P(X_n = 1) = 1/n^2$ (e.g., rolling a 1 on a fair n^2-sided die). Define the sequence of events $G_n = \{X_n = 1\}$. Since, in this case, $\sum_{n=1}^{\infty} P(G_n) = \pi^2/6 < \infty$, $\{X_n = 1\}$ only occurs a finite number of times before X_n takes on the value 0 with probability 1. Thus, there must exist a random time N such that $X_n = 0$ for all $n > N$. We use this lemma to prove that any simple or compound experiment fails with probability 1 if it is longer than some finite but random length.

5.1 Definitions

Recall that we defined $\mathcal{E}$ as a discrete POMDP with states S, actions A, observations O, transition probabilities T, and observation probabilities γ, where γ_s^o is the probability of observing observation o in state s. Let $\mathcal{M}$ be an agent situated in environment $\mathcal{E}$.

We define a *k-action simple experiment* e_k performed by $\mathcal{M}$ on $\mathcal{E}$ as a sequence of k ordered actions beginning at time t with an initial action:

$$(a_t, ..., a_{t+k-1}) \tag{18}$$

with k ordered expected observations beginning at time $t+1$:

$$(o_{t+1}, ..., o_{t+k}) \tag{19}$$

Let E_k be a random variable that takes on the value 1 when the result of $\mathcal{M}$ executing the k actions in experiment e_k in order beginning at time t is the k expected observations in order, starting at time $t+1$, and takes on the value 0 otherwise. Then:

$$P(E_k = 1) = P(o_{t+1}, ..., o_{t+k} | a_t, ..., a_{t+k-1}) \tag{20}$$

$P(E_k = 1)$ is the probability that simple experiment e_k *succeeds.* $P(E_k = 0) = 1 - P(E_k = 1)$ is, then, the probability that simple experiment e_k *fails.* Note that we condition on the actions in a simple experiment since these values are fixed by the agent rather than drawn from a distribution. Actions are not random variables in a simple experiment because there is only one choice of action at each time step in a simple experiment.

We define a *k-action compound experiment* $e_{c,k}$ performed by $\mathcal{M}$ on $\mathcal{E}$ as a sequence of k ordered sets of possible actions that can be selected from at each time step beginning at time t:

$$(\{a_t^m\}, ..., \{a_{t+k-1}^m\}) \tag{21}$$

and k ordered sets of allowed observations corresponding to the action selected in the previous time step, beginning at time $t+1$:

$$(\{o_{t+1}^r\}, ..., \{o_{t+k}^r\}) \tag{22}$$

At each time step $i \in \{t, ..., t+k-1\}$, the agent can choose between at least one and no more than $|A|$ actions. The choice of action m, a_i^m, at time step i corresponds to a particular set of allowed observations at time step $i+1$, $\{o_{i+1}^r\}$, which will contain between 1 and $|O|$ allowed observations and will, in general, be different for each action. Seeing any observation r, o_{i+1}^r, in this set is valid. Crucially, however, at each time step i, there must exist at least one action such that $|\{o_{i+1}^r\}| < |O|$. Were this not the case, any observation would be allowed at time step $i+1$ for all possible selections of allowed action at time step i, and k-length compound experiment $e_{c,k}$ would be equivalent to a $k-1$-length experiment that ignored the action at time step i and corresponding observation at time step $i+1$.

Let $E_{c,k}$ be defined analogously to E_k for compound experiment $e_{c,k}$. In a compound experiment, the actions are modeled as random variables because, at each time step i, the learning

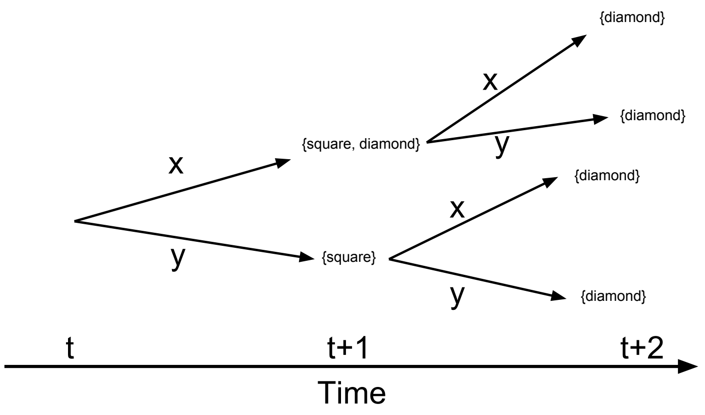


Figure 2: Visualization of the compound experiment in equation 23

agent chooses uniformly at random between the available actions. Consider the following example of a compound experiment in the Shape environment (Figure 1), beginning at time t, written in alternating action and observation form:

$$(\{x,\ y\}, \{\{square,\ diamond\}, \{square\}\}, \{x, y\}, \{\{diamond\}, \{diamond\}\}) \tag{23}$$

If the action x is executed at time t, either observation in the set $\{square,\ diamond\}$ is allowed at time $t+1$. In contrast, if y is executed at time t, then only *square* is allowed at time $t+1$. Likewise, either x or y may be executed at time $t+1$, and, in either case, only *diamond* is allowed to be observed at time $t+2$. Figure 2 provides a visualization of this experiment. We now state the main results and refer the reader to Appendix A for the proofs.

5.2 Statement of Main Results

Lemma 1 (Any simple experiment longer than some finite random length fails with probability 1)**.** *If, for all $o \in O$ and $s \in S$, $0 < \gamma_s^o < 1$, then, for some finite random integer D, k-action simple experiment e_k will fail for all $k > D$ with probability 1 (almost surely).*

Proof. Please see Appendix A □

Lemma 2 (Any compound experiment longer than some finite random length fails with probability 1)**.** *If, for all $o \in O$ and $s \in S$, $0 < \gamma_s^o < 1$, then, for some finite random integer D, k-action compound experiment $e_{c,k}$ will fail for all $k > D$ with probability 1 (almost surely).*

Proof. Please see Appendix A □

Theorem 1 (All SDEs have a finite length)**.** *Assume that, for all $o \in O$ and $s \in S$, $0 < \gamma_s^o < 1$. V is the set of SDEs learned in POMDP $\mathcal{E}$ by the SDE learning algorithm. $|v_i|$ denotes the length of $v_i \in V$'s experiment e_{v_i}. With probability 1 (almost surely), there exists a finite random integer D such that $\forall_{v_i \in V}\ |v_i| \leq D$.*

Proof. Please see Appendix A □

Theorem 2 (Computational Complexity)**.** *The worst-case computational complexity of the SDE learning algorithm is $O(D|A|^{2D+1}|O|^{2D+2})$, where D is the maximum length of any SDE in the learned SDE model V.*

Proof. Please see Appendix A □

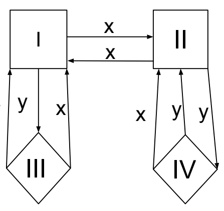
6 Experimental Evaluation

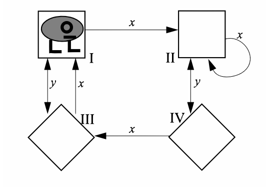
6.1 Environmental Hiddenness

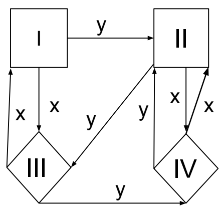
Consider a stochastic version of the shape environment in Figure 1 in which the arrows represent the *most probable* transitions between states under the given action (with probability p_{trans}), and the remaining $1 - p_{trans}$ probability mass is divided equally amongst transitions to the other 3 possible states. states I and II emit a *square* observation with high probability o_{emit}, while states III and IV emit a *diamond* observation with high probability o_{emit}. During our analysis of the deterministic and stochastic versions of this problem, we found that different environments displayed different levels of *hiddenness*, a fundamental property of the structure of the environment that makes it difficult for any learning algorithm. We can formally define *hiddenness* as the number of actions from states that appear the same with high probability to the agent that lead to different observations with high probability. These represent precisely the situations in which the agent must use its history to disambiguate results, even in a deterministic environment. This is not necessarily equivalent to the number of hidden states. We evaluated our algorithm on four structures (S0 through S3) with increasing levels of hiddenness (see Figures 3(a)-(d)). The *hiddenness value* of an environment is the total number of such actions in underlying states that behave differently with high probability, even though the situation (observation) looks the same to the agent with high probability. In structure S0 (Figure 3(a)), for instance, even though there are two hidden states, executing action x from states I and II (which both emit a *square* observation with high probability) leads to a *square* observation with high probability, while executing action y from states I and II leads to a *diamond* observation with high probability. An analogous argument can be made for executing either x or y in states III and IV, leading to a hiddenness value of 0 for S0. In structure S1 (Figure 3(b)), the only such action that demonstrates this property is executing x in states III and IV, which results in different observations with high probability, leading to a hiddenness value of 1. The hiddenness values for the other environments are calculated similarly. We use this metric to give a bound on the proposed algorithm's performance in this small shape environment, and, based on our extensive experimentation, these structures do, in fact, represent the range of performances exhibited by the algorithm in the shape environment.

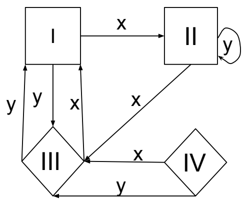
6.2 Experimental Results

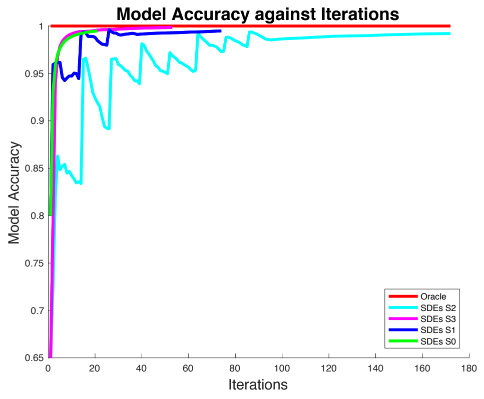
As an initial evaluation, we ran the proposed algorithm with $sgain = rgain = 1$, $convergeTol = 5$, $numExp = 20$ and with merging turned off on each representative structure in Figures 3(a)-(d) when the environment was completely deterministic ($p_{trans} = 1$, $o_{emit} = 1$) to ensure that it performed extremely closely to an *oracle* that had full knowledge of the environment and the current state (and could thus predict future observations with 100% accuracy). Figure 3(e) shows that, in each representative environment, the accuracy of the model in predicting next observations built by the proposed algorithm becomes extremely close to the predictive accuracy of the oracle (100%) as the number of iterations of the algorithm increases.


(a) Hiddenness: 0 (S0)

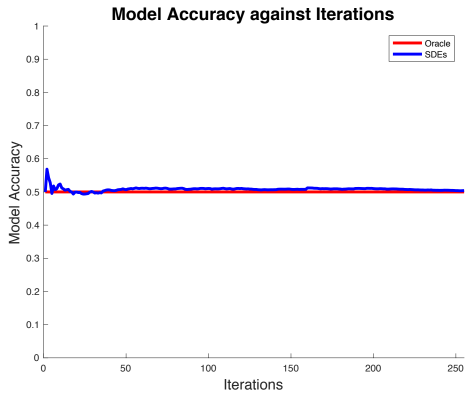

(b) Hiddenness: 1 (S1)


(c) Hiddenness: 2 (S2)

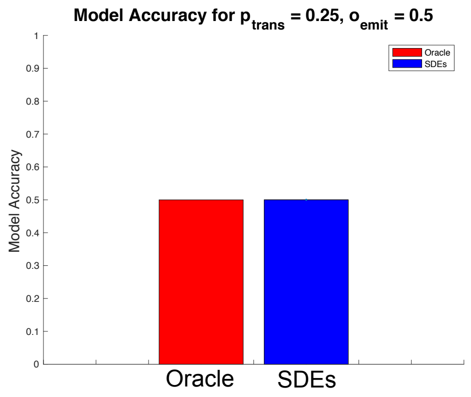

(d) Hiddenness: 4 (S3)



(e) Example runs in fully deterministic environments



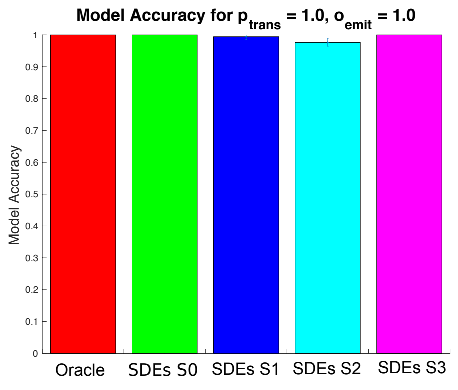
(f) Example runs in fully non-deterministic environments



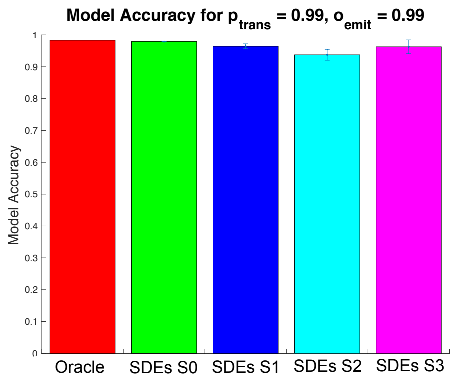
(g) Model accuracy in fully non-deterministic environments

Figure 3: Representative structures used for testing (a)-(d) and graphs of the algorithm's performance in fully deterministic (e) and fully non-deterministic environments (f),(g).

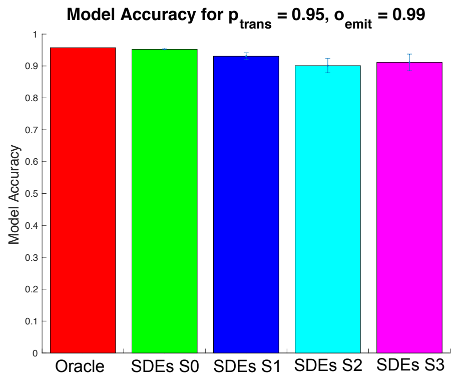
This behavior was consistent when averaged across many runs, and representative runs of the algorithm are used in Figure 3(e). Figure 3(f) shows that, in completely non-deterministic environments in which $p_{trans} = 0.25$ and $o_{emit} = 0.5$, the SDE model begins with an accuracy of 0.5 (equivalent to the oracle's accuracy) and stays very close to 0.5 throughout the execution of the algorithm, where the spikes and noisiness are due to the noisiness in probability estimates



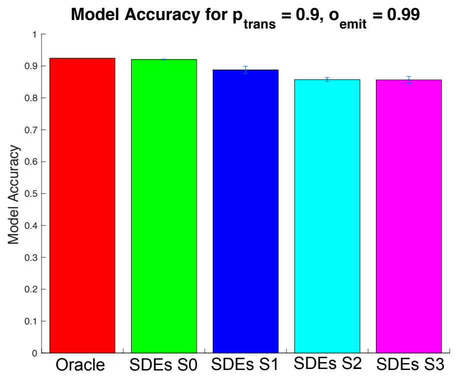
(a)



(b)



(c)

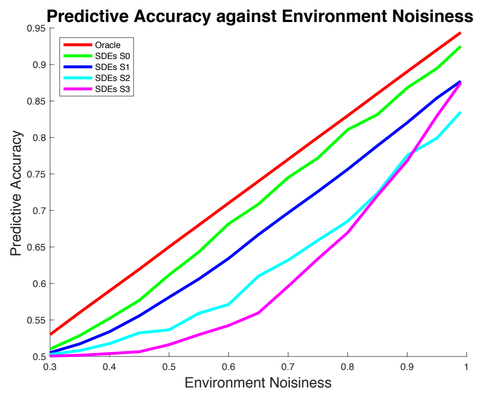


(d)

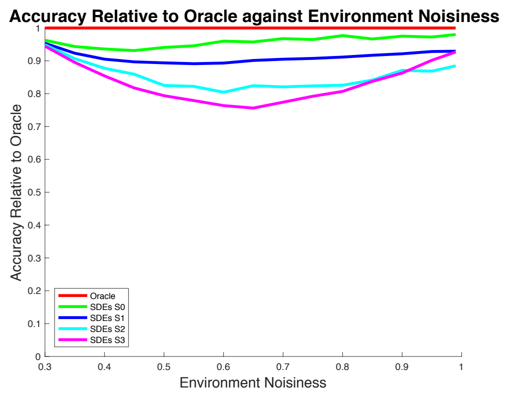
Figure 4: SDE learning results in environments with high determinism.

in the SDE learner's model as it makes changes throughout the algorithm. As the algorithm continues to run, we see that it correctly realizes that its model is only accurate 50% of the time. Note that when the environment is truly nondeterministic, all structures in Figures 3(a)-(d) behave equivalently and no algorithm can achieve better than 50% accuracy with two observations. Figure 3(g) further validates this by showing the performance of the SDE model and oracle in predicting 50000 observations averaged over 50 runs of the algorithm.

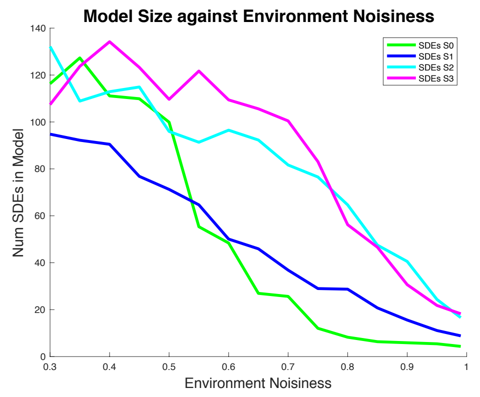
A second set of experiments evaluated the predictive performance of the SDE learning algorithm in highly deterministic environments, in which p_{trans} varied from 0.9 to 1.0 and the observation probability o_{emit} was kept at 0.99 (meaning that the agent's "sensor" for observations was almost perfect), except in the fully deterministic environment Figure 4(a), where it was set to 1.0. In each environment and for each set of parameters tested, the proposed SDE learning algorithm was used to build a model of the environment (with the agent initially placed in the environment at a state chosen uniformly at random). Once the SDE learning procedure converged, the agent was transported to a new state (uniformly at random) and predicted the next sequence of 50000 observations. $sgain = rgain = 1.07$, $numExp = 100$, $mgain = 1.03$, and $convergeTol = 5$ were used as parameters. The predictive accuracy in the bar graphs in Figures 4(a)-(d) is the number of times the agent succeeding in predicting the next observa-



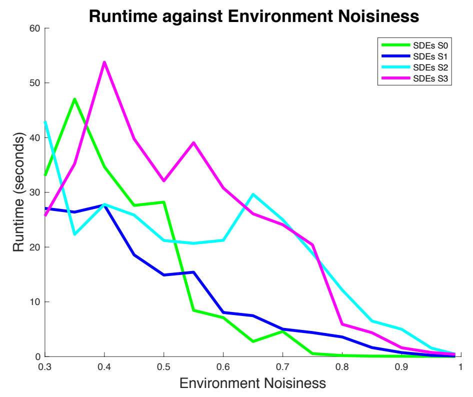
(a) Predictive Accuracy



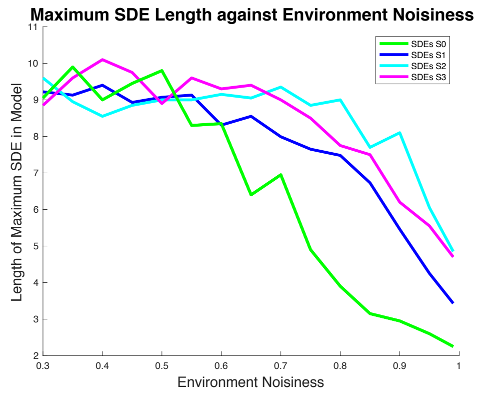
(b) Relative Predictive Accuracy



(c) Number of SDEs in Model



(d) Runtime



(e) Length of Largest SDE

Figure 5: SDE learning in the Shape environment as environment determinism increases.

tion using its model divided by 50000. Figure 4 illustrates that the SDE learning algorithm performs almost as well as the oracle, with an accuracy relative to the oracle of 91% in the most difficult (S2, S3) and noisy environments (see Figure 4(d)) and almost 100% in structure S0 (no hiddenness) regardless of noise. In deterministic environments (Figure 4(a)), the SDE model was between 98% and 100% accurate. It was perfectly accurate on two out of the four environments (S0 and S3). The oracle used its underlying knowledge of the state space and current state to always predict, for a given action, the observation most likely to be emitted in the state most likely to be transitioned into. Error bars indicate the standard deviation of each estimate, which is averaged over 50 runs.

A final set of experiments tracked the predictive accuracy of the SDE learning procedure and the oracle as the environment became increasingly deterministic. The predictive accuracy measures were calculated by the same process as in Figure 4. $numExp = 25$, $sgain = rgain = 1.1$, $mgain = 1.05$, and $convergeTol = 5$ were used as parameters. p_{trans} varied from 0.3 to 0.99 in increments of 0.05 (with 0.99 following 0.95), and o_{emit} was set to 0.95 for seeing *diamond* in states III and IV and for seeing *square* in states I and II. Each data point was averaged over 20 runs of the algorithm. Figure 5 illustrates the results.

As expected, the predictive accuracy of the algorithm was highly correlated with the hiddenness of the environment. More hidden environments were more difficult to learn, particularly when p_{trans} was low (Figures 5(a),(b)). The runtime also decreased as p_{trans} increased (Figure 5(d)), which is in line with our analysis and proof of convergence in the previous section. As p_{trans} increases, fewer trajectories through the POMDP have sufficient probability to drive the splitting procedure for larger SDE lengths. As expected, higher values of p_{trans} led to smaller model sizes (Figure 5(c)) and models with shorter SDEs (Figure 5(e)), with environments with higher hiddenness requiring more SDEs and longer SDEs than those with low hiddenness even for the highest values of p_{trans}. The increases in runtime observed with increasing environmental noisiness (nondeterminism) are examples of the fact that the D (maximum SDE length) from our analysis in Section 5 exists outside of the parameters of the algorithm itself. Changing p_{trans} indirectly affects D, leading to the changes in runtime observed. The results are encouraging and indicate that, even in environments with high hiddenness, high noise, and imperfect sensors, SDEs are capable of forming a good approximation of the state space of an unknown POMDP that allows for accurate prediction.

7 Conclusions and Future Work

In this paper, we presented a novel and robust biologically-inspired cognitive architecture for actively learning a predictive model of discrete, stochastic, partially-observable environments based on a novel *surprise*-based learning concept called Stochastic Distinguishing Experiments (SDEs). SDEs are probability distributions over the next observation given a variable-length ordered sequence of actions and expected observations up to the present that partition the space of possible agent histories. SDE learning does not require an external reward function, any prior knowledge of the state space (or the number of states), nor does it require a user-defined bound on maximum SDE length. Mathematical proofs were provided for the convergence and computational complexity of the proposed algorithm. Experimental results in small but representative environments on a prediction task served as a validation of our theoretical analyses and provided empirical evidence that SDEs, do, in fact, serve as an approximate representation of state. Future work will strive to prove theoretical bounds on how good the SDE approximation is and validate SDE learning in larger, more complex, and more diverse environments. Additional future work will investigate how an external reward function might be used by an SDE representation to learn approximately optimal policies in unknown environments, potentially leading to an SDE-based reinforcement learning algorithm for optimal decision making.

References

[1] Ron Begleiter, Ran El-Yaniv, and Golan Yona. On prediction using variable order markov models. *Journal of Artificial Intelligence Research*, 22:385–421, 2004.

[2] Byron Boots, Sajid M Siddiqi, and Geoffrey J Gordon. Closing the learning-planning loop with predictive state representations. *The International Journal of Robotics Research*, 30(7):954–966, 2011.

[3] Francesco Paolo Cantelli. Sulla probabilita come limite della frequenza. *Atti Accad. Naz. Lincei*, 26(1):39–45, 1917.

[4] Christos Dimitrakakis. Bayesian variable order markov models. In *AISTATS*, pages 161–168, 2010.

[5] Christos Dimitrakakis. Variable order markov decision processes: Exact bayesian inference with an application to pomdps. 2010.

[6] Finale Doshi-Velez. The infinite partially observable markov decision process. In *Advances in neural information processing systems*, pages 477–485, 2009.

[7] M Émile Borel. Les probabilités dénombrables et leurs applications arithmétiques. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 27(1):247–271, 1909.

[8] William L Hamilton, Mahdi Milani Fard, and Joelle Pineau. Efficient learning and planning with compressed predictive states. *Journal of Machine Learning Research*, 15(1):3395–3439, 2014.

[9] Barbara Koslowski and Jerome S Bruner. Learning to use a lever. *Child Development*, pages 790–799, 1972.

[10] Michael L Littman, Richard S Sutton, Satinder Singh, et al. Predictive representations of state. *Advances in neural information processing systems*, 2:1555–1562, 2002.

[11] F. Liu, X. Jin, and Y. She. No-fringe u-tree: An optimized algorithm for reinforcement learning. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 278–282, Nov 2016.

[12] Yunlong Liu, Hexing Zhu, Yifeng Zeng, and Zongxiong Dai. Learning predictive state representations via monte-carlo tree search. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 3192–3198. AAAI Press, 2016.

[13] Andrew Kachites McCallum. *Reinforcement learning with selective perception and hidden state.* PhD thesis, University of Rochester, 1996.

[14] R Andrew McCallum. Instance-based utile distinctions for reinforcement learning with hidden state. In *ICML*, pages 387–395, 1995.

[15] R Andrew McCallum, G Tesauro, D Touretzky, and T Leen. Instance-based state identification for reinforcement learning. *Advances in Neural Information Processing Systems*, pages 377–384, 1995.

[16] Peter McCracken and Michael Bowling. Online discovery and learning of predictive state representations. *Advances in neural information processing systems*, 18:875, 2006.

[17] Kevin P Murphy. A survey of pomdp solution techniques. *environment*, 2:X3, 2000.

[18] Charles Sanders Peirce and Andreas Hetzel. How to make our ideas clear. 1878.

[19] Jean Piaget. The origin of intelligence in the child, 1953.

[20] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, pages 1025–1032, 2003.

[21] Nadeesha Ranasinghe and Wei-Min Shen. Surprise-based learning for developmental robotics. In *Learning and Adaptive Behaviors for Robotic Systems, 2008. LAB-RS'08. ECSIS Symposium on*, pages 65–70. IEEE, 2008.

[22] Nadeesha Oliver Ranasinghe. *Learning to detect and adapt to unpredicted changes.* PhD thesis, University of Southern California, 2012.

[23] Ronald L Rivest and Robert E Schapire. Inference of finite automata using homing sequences. In *Machine Learning: From Theory to Applications*, pages 51–73. Springer, 1993.

[24] Nicholas Roy, Geoffrey J Gordon, and Sebastian Thrun. Finding approximate pomdp solutions through belief compression. *Journal of Artificial Intelligence Research (JAIR)*, 23:1–40, 2005.

[25] Wei-Mein Shen. Learning finite automata using local distinguishing experiments. In *IJCAI*, pages 1088–1093, 1993.

[26] Wei-Min Shen. *Learning from the environment based on percepts and actions.* PhD thesis, Carnegie Mellon University, 1989.

[27] Wei-Min Shen. Complementary discrimination learning with decision lists. In *AAAI*, pages 153–158, 1992.

[28] Wei-Min Shen. Discovering regularities from knowledge bases. *International Journal of Intelligent Systems*, 7(7):623–635, 1992.

[29] Wei-Min Shen. Discovery as autonomous learning from the environment. *Machine Learning*, 12(1):143–165, 1993.

[30] Wei-Min Shen. *Autonomous Learning from the Environment, (Forwarded by Herbert A. Simon).* Computer Science Press, WH Freeman, 1994.

[31] Wei-Min Shen and Herbert A Simon. Fitness requirements for scientific theories containing recursive theoretical terms. *The British journal for the philosophy of science*, 44(4):641–652, 1993.

[32] Karl Sigman. Lecture notes on borel-cantelli lemmas. `http://www.columbia.edu/~ks20/stochastic-I/stochastic-I-BC.pdf`, 2009.

[33] Satinder Singh, Michael L Littman, Nicholas K Jong, David Pardoe, and Peter Stone. Learning predictive state representations. In *ICML 2003*, pages 712–719.

[34] Britton Wolfe, Michael R James, and Satinder Singh. Learning predictive state representations in dynamical systems without reset. In *Proceedings of the 22nd international conference on Machine learning*, pages 980–987. ACM, 2005.

[35] Lei Zheng, Siu-Yeung Cho, and Chai Quek. Reinforcement based u-tree: A novel approach for solving pomdp. In *Handbook on Decision Making*, pages 205–232. Springer, 2010.

A Proofs of Convergence and Computational Complexity

For all time-steps i, let A_i be a random variable representing the chosen action, let O_i be a random variable representing the observation, and let S_i be a random variable representing the state. Note that all A_i are independent of one another, because each action is chosen uniformly at random from the available actions at that time step, regardless of the actions taken at previous or future time steps. Subscripted lowercase letters, e.g., o_{t+3}, a_{t+1}, refer to fixed but not necessarily known values assigned to these random variables at specific time steps. Summations over all possible values or all possible combinations of values of a random variable or sequence of random variables at certain time steps are denoted by putting the subscripted lowercase letter(s) underneath the summation sign. For example: $\sum\limits_{a_t}$ means to sum over all possible assignments to A_t and $\sum\limits_{s_{t+1},\dots,s_{t+4}}$ means to sum over all possible combinations of values that could be assigned to random variables S_{t+1}, S_{t+2}, S_{t+3}, S_{t+4}. In the proof of Lemma 2, A_i can take on the value of any action at any time step. Specific actions at time-step i are referred to with superscripts, $a_i^1, ...a_i^{|A|}$. Thus, as in the proof of Lemma 2, we can decompose summations over all possible values of A_i into separate summations over specific action values at time i. For example, $\sum\limits_{a_i} P(a_i)$ is equivalent to $\sum\limits_{j=1}^{|A|-1} P(a_i^j) + P(a_i^{|A|})$. Recall that we assume POMDP $\mathcal{E}$ is defined such that $|A|> 1$, $|O|> 1$.

Lemma 1 (Any simple experiment longer than some finite random length fails with probability

1). *If, for all $o \in O$ and $s \in S$, $0 < \gamma_s^o < 1$, then, for some finite random integer D, k-action simple experiment e_k will fail for all $k > D$ with probability 1 (almost surely).*

Proof. Recall that $E_k = 1$ is the event that experiment e_k succeeds. We calculate $P(E_k = 1)$ in terms of underlying POMDP $\mathcal{E}$ as follows:

$$P(E_k = 1) = P(o_{t+1}, ..., o_{t+k} | a_t, ..., a_{t+k-1}) \tag{24}$$

$$= \sum_{s_t,...,s_{t+k}} P(o_{t+1}, ..., o_{t+k}, s_t, ..., s_{t+k} | a_t, ..., a_{t+k-1}) \tag{25}$$

$$= \sum_{s_t,...,s_{t+k}} P(o_{t+1}, ..., o_{t+k} | s_t, ..., s_{t+k}, a_t, ..., a_{t+k-1}) P(s_t, ..., s_{t+k} | a_t, ..., a_{t+k-1}) \tag{26}$$

$$= \sum_{s_t,...,s_{t+k}} P(o_{t+1}, ..., o_{t+k} | s_{t+1}, ..., s_{t+k}) P(s_t, ..., s_{t+k} | a_t, ..., a_{t+k-1}) \tag{27}$$

$$= \sum_{s_t,...,s_{t+k}} (\prod_{i=t+1}^{t+k} P(o_i|s_i)) P(s_t, ..., s_{t+k} | a_t, ..., a_{t+k-1}) \tag{28}$$

Where the final two steps (equations 27 and 28) utilize the sensor Markov assumption of the underlying POMDP state space. Since we do not know the particular values $o_{t+1}, ..., o_{t+k}$, we will proceed by providing an upper bound on the probability of *any* k-length simple experiment and showing that summing up this upper bound from $k = 0$ to ∞ converges to a finite value. Define $\beta \triangleq \max_{o,s} \gamma_s^o$, i.e., the maximum probability of observing any observation in any state, and substitute this maximum value in for each $P(o_i|s_i)$:

$$P(E_k = 1) = \sum_{s_t,...,s_{t+k}} (\prod_{i=t+1}^{t+k} P(o_i|s_i)) P(s_t, ..., s_{t+k} | a_t, ..., a_{t+k-1}) \tag{29}$$

$$\leq \beta^k \sum_{s_t,...,s_{t+k}} P(s_t, ..., s_{t+k} | a_t, ..., a_{t+k-1}) \tag{30}$$

$$= \beta^k \tag{31}$$

Where the final step (equation 31) utilizes the fact that $\sum_{s_t,...,s_{t+k}} P(s_t, ..., s_{t+k} | a_t, ..., a_{t+k-1}) = 1$ since it is a valid probability distribution. Thus:

$$P(E_k = 1) \leq \beta^k \tag{32}$$

Recall that the Borel-Cantelli lemma states that if $G_1, G_2, ...$ is a sequence of events in a probability space and the sum of their probabilities is finite (i.e., $\sum_{k=1}^{\infty} P(G_k) < \infty$), then the probability that infinitely many of these events occur is 0. Consider the sequence of events $\{E_1 = 1\}, \{E_2 = 1\}, ...$, where, in general, G_k is the event $\{E_k = 1\}$. Since, by assumption, for all $o \in O$ and $s \in S$, $0 < \gamma_s^o < 1$, $0 < \beta < 1$, and:

$$\sum_{k=1}^{\infty} P(G_k) \leq \sum_{k=0}^{\infty} \beta^k = \frac{1}{1-\beta} < \infty \tag{33}$$

Thus, $E_k = 1$ occurs only finitely many times and, consequently, for $k > D$ for some finite random integer D, E_k always takes on the value 0 (i.e., fails) with probability 1 (almost surely), completing the proof. □

Lemma 2 (Any compound experiment longer than some finite random length fails with probability 1)**.** *If, for all $o \in O$ and $s \in S$, $0 < \gamma_s^o < 1$, then, for some finite random integer D, k-action compound experiment $e_{c,k}$ will fail for all $k > D$ with probability* 1 *(almost surely).*

Proof. We begin by noting the following: the more trajectories covered by compound experiment $e_{c,k}$, the higher its probability of succeeding must necessarily be. We will again proceed by finding an upper bound on the probability of *any* compound experiment of length k by assuming that, at each time step $i \in \{t, ..., t+k-1\}$, all $|A|$ possible actions are allowed to be selected, and that, for exactly one action allowed at each time step i, only $|O|-1$ observations are allowed at time step $i+1$. For all other actions at time step i, any observation is allowed at time step $i+1$. Without loss of generality, assume that, at each time step i, actions $a_i^1, ..., a_i^{|A|-1}$ are the $|A|-1$ actions for which any observation is allowed at time step $i+1$, while $a_i^{|A|}$ is the action for which only $|O|-1$ observations are allowed at time step $i+1$. Let α be the maximum, over all states, of the probability of observing any one of the most likely set of $|O|-1$ observations. Since all observations are mutually exclusive, this is simply the largest sum of $|O|-1$ probabilities possible in any state. Note that, since for all $o \in O$ and $s \in S$, $0 < \gamma_s^o < 1$ by assumption, $0 < \alpha < 1$. We now write $P(E_{c,k} = 1)$ in terms of the underlying POMDP:

$$P(E_{c,k} = 1) = \sum_{a_t,...,a_{t+k-1}} P(\{o_{t+1}^r\}, ..., \{o_{t+k}^r\} | a_t, ..., a_{t+k-1}) P(a_t, ..., a_{t+k-1}) \quad (34)$$

$$= \sum_{s_t,...,s_{t+k}} \sum_{a_t,...,a_{t+k-1}} P(\{o_{t+1}^r\}, ..., \{o_{t+k}^r\}, s_t, ..., s_{t+k} | a_t, ..., a_{t+k-1}) P(a_t, ..., a_{t+k-1}) \quad (35)$$

$$= \sum_{s_t,...,s_{t+k}} \sum_{a_t,...,a_{t+k-1}} P(\{o_{t+1}^r\}, ..., \{o_{t+k}^r\} | s_t, ..., s_{t+k}, a_t, ..., a_{t+k-1}) \quad (36)$$

$$P(s_t, ..., s_{t+k} | a_t, ..., a_{t+k-1}) P(a_t, ..., a_{t+k-1})$$

$$= \sum_{s_t,...,s_{t+k}} \sum_{a_t,...,a_{t+k-1}} (\prod_{i=t+1}^{t+k} P(\{o_i^r\} | s_i, a_{i-1}) P(s_i | s_{i-1}, a_{i-1}) P(a_{i-1})) P(s_t) \quad (37)$$

Where the final step (equation 37) is due to the sensor and transition Markov properties of the underlying POMDP state space and we have also used the fact that, since all action random variables at all time steps $i-1$, A_{i-1}, are independent, $P(a_t, ..., a_{t+k-1}) = P(a_t)P(a_{t+1}) \cdots P(a_{t+k-1})$. $P(a_{i-1}) = \frac{1}{|A|}$ for all time steps i because any action can be chosen at any time step and actions are chosen uniformly at random. We know that $|\{o_i^r\}| = |O|$ when $a_{i-1} \in \{a_{i-1}^1, ..., a_{i-1}^{|A|-1}\}$, and, thus, $P(\{o_i^r\} | s_i, a_{i-1}^j) = 1$ when $j \in \{1, ..., |A|-1\}$. $P(\{o_i^r\} | s_i, a_{i-1}^{|A|}) \leq \alpha < 1$ when $a_{i-1} = a_{i-1}^{|A|}$.

We see that, if we marginalize out over a_{i-1} and s_{i-1} at each time step i, equation 37 can be defined as a recursive procedure in which the (in general, unnormalized) joint distribution over the current state and observations $P(S_i, \{o_{t+1}^r\}, ..., \{o_i^r\})$ is computed from the (in general, unnormalized) joint distribution over the state and observations up to the previous time step $P(S_{i-1}, \{o_{t+1}^r\}, ..., \{o_{i-1}^r\})$. At the final time step, S_{t+k} is marginalized out to find $P(\{o_{t+1}^r\}, ..., \{o_{t+k}^r\})$. At the first time step, the normalized distribution $P(S_t)$ is used to compute $P(S_{t+1}, \{o_{t+1}^r\})$, where $P(\{o_{t+1}^r\}) = \sum_{s_{t+1}} P(s_{t+1}, \{o_{t+1}^r\})$. We proceed by finding the following upper bound: $P(\{o_{t+1}^r\}) \leq \psi$, which is analogous to β in the proof of Lemma 1. Clearly, due to the recursive nature of this computation, if $P(\{o_{t+1}^r\}) \leq \psi$ and $0 < \psi < 1$, then $P(\{o_{t+1}^r\}, \{o_{t+2}^r\}) \leq \psi^2$ and, in general, $P(\{o_{t+1}^r\}, ..., \{o_{t+k}^r\}) \leq \psi^k$. Thus:

$$P(\{o^r_{t+1}\}) = P(E_{c,1} = 1) \tag{38}$$

$$= \sum_{a_t} \sum_{s_{t+1}} P(\{o^r_{t+1}\}|s_{t+1}, a_t) \sum_{s_t} P(s_{t+1}|s_t, a_t)P(a_t)P(s_t) \tag{39}$$

$$= \sum_{a_t} \sum_{s_{t+1}} P(\{o^r_{t+1}\}|s_{t+1}, a_t) \sum_{s_t} P(s_{t+1}, s_t, a_t) \tag{40}$$

$$= \sum_{a_t} \sum_{s_{t+1}} P(\{o^r_{t+1}\}|s_{t+1}, a_t)P(s_{t+1}, a_t) \tag{41}$$

$$= \sum_{j=1}^{|A|-1} \sum_{s_{t+1}} P(\{o^r_{t+1}\}|s_{t+1}, a^j_t)P(s_{t+1}, a^j_t) + \sum_{s_{t+1}} P(\{o^r_{t+1}\}|s_{t+1}, a^{|A|}_t)P(s_{t+1}, a^{|A|}_t) \tag{42}$$

$$\leq \sum_{j=1}^{|A|-1} \sum_{s_{t+1}} P(s_{t+1}, a^j_t) + \sum_{s_{t+1}} \alpha P(s_{t+1}, a^{|A|}_t) \tag{43}$$

$$= \sum_{j=1}^{|A|-1} P(a^j_t) + \alpha P(a^{|A|}_t) \tag{44}$$

$$= \sum_{j=1}^{|A|-1} \frac{1}{|A|} + \alpha \frac{1}{|A|} = \frac{|A|-1+\alpha}{|A|} < 1 \tag{45}$$

Thus, $\psi = (|A|-1 + \alpha)/|A|$. Equation 42 separates out the value $a^{|A|}_t$ from the summation over the possible values of A_t such that all remaining terms in the summation from $j = 1$ to $|A|-1$ have the same upper bound in observation probability at time step $t + 1$. Equation 43 substitutes in these upper bounds for the observation probabilities, recalling that $P(\{o^r_{t+1}\}|s_{t+1}, a^{|A|}_t) \leq \alpha$ and $P(\{o^r_{t+1}\}|s_{t+1}, a^j_t) = 1$ for $j \in \{1, ..., |A|-1\}$. Equation 45 utilizes the fact that $P(a_t) = 1/|A|$ for all possible values of A_t and the fact that $0 < \alpha < 1$. Thus, we have that $P(E_{c,k} = 1) \leq \psi^k$ for a value ψ where $0 < \psi < 1$. Analogously to Lemma 1, we define the sequence of events $G_{c,k} = \{E_{c,k} = 1\}$, and:

$$\sum_{k=1}^{\infty} P(G_{c,k}) \leq \sum_{k=0}^{\infty} \psi^k = \frac{1}{1-\psi} < \infty \tag{46}$$

Thus, $E_{c,k} = 1$ occurs only finitely many times and, consequently, for $k > D$ for some finite random integer D, $E_{c,k}$ always takes on the value 0 (i.e., fails) with probability 1 (almost surely), completing the proof.

□

Theorem 1 (All SDEs have a finite length)**.** *Assume that, for all $o \in O$ and $s \in S$, $0 < \gamma^o_s < 1$. V is the set of SDEs learned in POMDP $\mathcal{E}$ by the SDE learning algorithm. $|v_i|$ denotes the length of $v_i \in V$'s experiment e_{v_i}. With probability* 1 *(almost surely), there exists a finite random integer D such that $\forall_{v_i \in V}$ $|v_i| \leq D$.*

Proof. Each $v_i \in V$ is associated with a $|v_i|$-action simple or compound experiment, e_{v_i}. $|v_i|$ increases by 1 every time SDE v_i is split (only splitting operations increase $|v_i|$), but splitting only occurs if *all* of v_i's one-step extension experiments succeed *at least one time* during some finite number of experiments. Assuming that for all $o \in O$ and $s \in S$, $0 < \gamma^o_s < 1$, we can use

Lemmas 1 and 2 to assert that, for each v_i, there exists a finite random integer D_i past which at least one of e_{v_i}'s one-step extension experiments will fail with probability 1 (almost surely). Define $D \triangleq \max_i D_i$. No SDE of length D will be split, and thus, no SDE has length longer than D (i.e., $\forall_{v_i \in V}\ |v_i| \leq D$), and the result is proved. $\square$

Theorem 2 (Computational Complexity)**.** *The worst-case computational complexity of the SDE learning algorithm is $O(D|A|^{2D+1}|O|^{2D+2})$, where D is the maximum length of any SDE in the learned SDE model V.*

Proof. We first note that, as discussed in Section 4.4, the SDE learning algorithm saves the pairs of SDE experiments involved in all split, refine, and merge operations in a hash map which is checked before each operation to ensure that the same operation is never performed more than once. This prevents any possible merge-split or merge-refine infinite loops, ensuring that the algorithm terminates in finite time. The complexity of each split, refine, and merge operation upper bounds the amount of time required for this check (and for the hashing of experiments), so this does not increase the overall complexity of the algorithm proved below.

We begin by showing three important bounds. First, Theorem 1 ensures us of the existence of a finite integer D bounding the number of actions and observations (and time steps represented) in any SDE's experiment. This immediately implies that the learning agent's model can contain no more than $O((|A||O|)^D)$ SDEs, one per simple D-length experiment. Second, in a compound experiment, the learning agent can choose from, at most, $|A||O|-1$ choices of action and associated observation at each time-step. This implies that processing all the elements of an SDE's experiment takes $O(D|A||O|)$ time. Third, a compound SDE can be associated with, at most $|A||O|-1$ probability distributions over the set of possible observations. Thus, processing all the elements of an SDE's probability distributions takes $O(|A||O|^2)$ time.

Each attempted or successful splitting operation takes $O(D|A|^2|O|^3)$ time, requiring a pass over each of the $O(D|A||O|)$ elements in the experiment and each of the $O(|A||O|^2)$ elements of the probability distributions of each of the $O(|A||O|)$ one-step extension SDEs of the SDE being split. Each successful or attempted refinement and merging operation requires $O(D|A||O|^2)$ time to process the elements in the experiments and probability distributions of at most two SDEs. We will now bound the worst-case runtime of the proposed SDE learning algorithm by bounding the number of *attempted* splits, refines, and merges, which obviously upper bounds the number of successful splits, refines, and merges.

Since the learning agent's model can contain no more than $O((|A||O|)^D)$ SDEs and no identical operations are ever repeated, the number of *successful* and *attempted* refines and the number of *successful* merges are both $O((|A||O|)^D)$. The number of *successful* splits is $O((|A||O|)^{D-1})$. We can bound the number of while loop iterations by noting that, after all successful splits, refines, and merges have completed in $O((|A||O|)^D)$ time, at each iteration, the algorithm performs the experimentation procedure (Section 4.1, which requires $O(D(|A||O|)^{D+1})$ time to process sub-trajectories) and attempts to perform a split. The learning agent's model has no more than $O((|A||O|)^D)$ SDEs, so the algorithm will perform experimentation and attempt to split $O((|A||O|)^D)$ additional times before every SDE has a $successCount >= convergeTol$, and the while loop terminates after $O((|A||O|)^D)$ iterations.

In the worst case, $O((|A||O|)^D)$ merges are attempted at each iteration, since we check every other SDE for merging eligibility with the selected SDE. The $O((|A||O|)^D(|A||O|)^D D|A||O|^2) = O(D|A|^{2D+1}|O|^{2D+2})$ time required by merging over the course of all iterations provides an upper bound on the time required for all experimentation, splits, refines, and merges throughout the course of the algorithm, and the result is proved.

$\square$