# Deep Bayesian Nonparametric Learning of Rules and Plans from Demonstrations with a Learned Automaton Prior – Supplement

**Brandon Araki,**[1] **Kiran Vodrahalli,**[2] **Thomas Leech,**[1, 3]
**Cristian-Ioan Vasile,**[1] **Mark Donahue,**[3] **and Daniela Rus**[1]
[1]CSAIL, Massachusetts Institute of Technology
[2] Columbia University
[3] MIT Lincoln Laboratory
araki@mit.edu, knv2609@columbia.edu, tfleech@mit.edu, cvasile@mit.edu, mark.donahue@ll.mit.edu, rus@csail.mit.edu

## 1 Nonparametric Bayesian Model

In the paper, we specify an approximation to the nonparametric Bayesian model by assuming that the number of FSA states is upper-bounded by $2P$. We state here the actual nonparametric model. Compared to the approximation in the paper, we replace the finite Dirichlet-Categorical distribution from which we draw $F$ with a nonparametric equivalent. In this model, $\alpha$ is the scaling parameter of the GEM distribution. The GEM distribution returns a theoretically infinite list of real numbers which can be thought of as weights on positive integers. Such a list can be generated using the stick-breaking process. We can use this list as the weights for an infinite Categorical distribution, from which we can sample $F$.

$$
\begin{aligned}
\alpha \in \mathbb{R}_{>0},\ \overline{\beta} \in (\beta^1, \dots),\ \overline{\gamma} \in (\gamma^1, \dots) \\
\theta \sim \text{GEM}(\alpha) \\
F \sim \text{Categorical}(\theta) \\
\beta^F \in \mathbb{R}_{>0}^{F \times P \times F},\ \gamma^F \in \mathbb{R}^F \times \mathbb{R}_{>0}^F \\
TM^F | \beta^F \sim \text{Dirichlet}(\beta^F) \\
\mathcal{R}^F | \gamma^F \sim \text{Normal}(\gamma^F) \\
\pi := \text{LVIN}(TM^F, \mathcal{R}^F) \\
a_t | s_{t-1}, f_{t-1} \sim \pi(s_{t-1}, f_{t-1}) \\
s_t := T(a_t, s_{t-1}),\ p_t := M(s_t) \\
f_t | p_t, f_{t-1}, TM^F \sim \text{Categorical}(TM^F(f_{t-1}, p_t))
\end{aligned}
$$

## 2 Experiments & Results

We test our model on 7 different domains, only 3 of which appear in the main paper. We present results from the other four domains here. We also describe how we modified the lunchbox TM.

### 2.1 Environments

**Gridworld Domains** The gridworld domains are simple $8 \times 8$ gridworlds with sequential goals. Gridworld1 has goals $a$ and $b$ (shown in Fig. 5a); Gridworld2 (Fig. 5b) adds goal $c$, and Gridworld3 (Fig. 5c) adds goal $d$. The specification of Gridworld1 is $\diamondsuit(a \wedge \diamondsuit b) \wedge \square \neg o$. Gridworld2's specification is $\diamondsuit(a \wedge \diamondsuit(b \wedge \diamondsuit c)) \wedge \square \neg o$ and Gridworld3's is $\diamondsuit(a \wedge \diamondsuit(b \wedge \diamondsuit(c \wedge \diamondsuit d))) \wedge \square \neg o$. The results for Gridworld1 in Table 1 show that LVIN achieves almost perfect performance on the domain; the LSTM achieves a success rate of 88.8% on the training data, which decreases to 64% on the test data. Similar results hold for Gridworld2 and 3; however, the LSTM performs much better on these domains. This is likely because these two domains have fewer obstacles than the Gridworld1 domain (the LSTM seems to struggle to avoid randomly placed obstacles).

**Dungeon Domain** The dungeon domain is a $12 \times 9$ gridworld and shows our model's ability to learn complex sequential specifications. In this environment (Fig. 11) there are 10 propositions: keys $ka$, $kb$, $kc$, $kd$ that unlock doors $da, db, dc$, and $dd$, respectively; and $g$ for the goal and $o$ for obstacles. To progress to the goal, the agent must follow the specification $\diamondsuit g \wedge \square \neg o \wedge (\neg da\ \mathcal{U}\ ka) \wedge (\neg db\ \mathcal{U}\ kb) \wedge (\neg dc\ \mathcal{U}\ kc) \wedge (\neg dd\ \mathcal{U}\ kd)$ – it must first pick up Key A, then go get Key D, then Key B, then Key C, before it can access the room in which the goal is located. The results in Table 1 show that whereas our model achieves a 100% success rate on test data, the LSTM is completely incapable of learning how to solve the environment, probably due to the large number of steps required to complete the task (up to 60) and the relative complexity of the specification.

### 2.2 Interpretability

One of the main benefits of our model is that it learns an interpretable model of the rules of an environment in the form of a transition matrix (TM). The learned vs. true TMs of all the domains are shown in Figs. 6, 9, 12, and 10.

To reiterate what was said in the main paper, the goal and trap states are not shown to save space, and also because their structures are trivial. The goal state consists of self-transitions back to the goal state, and the trap state consists of self-transitions back to the trap state. This is because they are both end states.

Recall that the TM contains the probability of a transition from a current state $f$ to a next state $f'$ given that a certain

| | Training | | Test | |
|---|---|---|---|---|
| | LVIN | LSTM | LVIN | LSTM |
| **Gridworld1** | | | | |
| Set size | 500 | 15000 | 3000 | 3000 |
| Success Rate | 99.80% | 88.80% | **99.97%** | 64.00% |
| **Gridworld2** | | | | |
| Set size | 600 | 10000 | 1200 | 2000 |
| Success Rate | 99.50% | 99.20% | **99.33%** | 98.90% |
| **Gridworld3** | | | | |
| Set size | 500 | 10000 | 1200 | 2000 |
| Success Rate | 100.0% | 98.00% | **99.99%** | 98.55% |
| **Lunchbox** | | | | |
| Set size | 500 | 9000 | 1800 | 1800 |
| Success Rate | 99.60% | 91.44% | **99.94%** | 82.44% |
| **Cabinet** | | | | |
| Set size | 550 | 6000 | 1800 | 1800 |
| Success Rate | 100.00% | 93.60% | **100.0%** | 89.58% |
| **Driving** | | | | |
| Set size | 500 | 6000 | 1800 | 1800 |
| Success Rate | 100.0% | 58.54% | **100.0%** | 58.60% |
| **Dungeon** | | | | |
| Set size | 800 | 6000 | 1800 | 1800 |
| Success Rate | 100.0% | 0.00% | **100.0%** | 0.00% |

Table 1: Training and test performance of LVIN vs LSTM

proposition $p$ is true. The TM of a given domain consists of a set of matrices. Each matrix corresponds to a current state $f$. Given a current state $f$ and a proposition $p$, the column associated with $p$ gives the probability of transitioning to each next state $f'$. White represents 0 probability, and black represents a probability of 1.

The learned TMs are represented somewhat differently. Instead of showing probabilities, they show the weights of the learned TM prior $\widehat{\beta}^{F^*}$. White corresponds to low weight and black corresponds to high weight. We show the weights because the weights of the TM prior give an indication of how "certain" the prior is of each set of transitions. For many $(f, p)$ pairs, the model is very uncertain about what the next state $f'$ will be (perhaps because that transition never occurs in the dataset), so the column will be mostly white because the values in the prior are low.

The learned vs true TMs of the gridworld domains are shown in Fig. 6. Inspecting the learned TM of Gridworld1 (Fig. 6a), we see that in the initial state $S0$, $a$ leads to the next state $S1$, whereas $b$ leads back to $S0$. In $S1$, going to $b$ leads to the goal state $G$. In both states, going on an obstacle $o$ leads to the trap state $T$. Therefore simply by inspection we can understand the specification that the agent is following. Gridworld2 and Gridworld3 follow analogous patterns.

Most of the learned TMs match closely with their expected TMs. There are two exceptions – one is the driving domain, discussed in the main paper. The other is the dungeon domain (Fig. 12), which instead of learning the intended general rules ("Door A is off-limits until Key A is picked up") learns domain-specific rules ("pick up Key A; then go to Door A; then pick up Key B; etc"). Crucially, however, this learned TM is still easy to interpret. In the initial state $S0$, most of columns are blank because the model is uncertain as to what the transitions are. The only transition it has learned (besides the obstacle and empty state transitions) is for Key A ($ka$), showing a transition to the next state. In $S1$, the only transition occurs at Door A ($da$). Then Key D ($kd$), Door D ($dd$), Key B ($kb$), Door B ($db$), Key C ($kc$), Door C ($dc$), and finally the goal state $g$. So we can see by inspecting the learned TM that the model has learned to go to those propositions in sequence. Although this differs from what we were expecting, it is still a valid set of rules that is also easy to interpret.

### 2.3 Manipulability Experiments on Jaco Arm

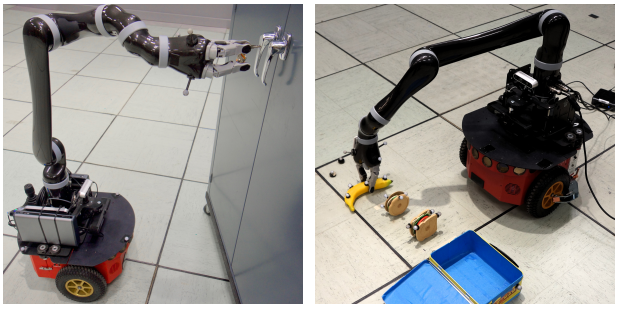


Figure 1: The Jaco mobile arm platform opening a cabinet and packing a lunchbox following rules learned from demonstrations.

To show how LVIN can be applied to the real world, we implemented the algorithm on a Jaco arm. The experimental setup is described in the main paper.

We modified two learned TMs – one from the lunchbox domain and one from the cabinet domain. We call the original lunchbox specification $\phi_{l1}$. There are three modifications – pick up only the sandwich first, then the banana ($\phi_{l2}$), pick up the burger first, then the banana ($\phi_{l3}$), and pick up the banana, then either the sandwich or the burger ($\phi_{l4}$). (We also modify the learned cabinet TM – these results are discussed in the main paper.)

In the lunchbox domain, $a$ indicates that the sandwich has been picked up, $b$ that the burger has been picked up, and $c$ that the banana has been picked up. $d$ indicates that the robot has dropped whatever it was holding into the lunchbox. Fig. 4a shows the modifications made to the lunchbox TM to cause it to pick up only the sandwich, and not the burger, first. In order to achieve this, in $S0$ we set $b$ (the burger proposition) to transition back to $S0$ rather than to $S1$, indicating to the agent that picking up the burger does not accomplish anything. With this change, the agent will only pick up the sandwich ($a$). Fig. 4b shows the analogous changes to make the agent pick up only the burger first. Fig. 4c shows how to make the agent pick up the banana first, and then either the sandwich or burger. In order to do this,

we first modify $S0$ so that picking up the banana ($c$) leads to the next state, whereas picking up the sandwich or burger ($a$ or $b$) leads back to $S0$. This change makes the agent pick up the banana first. $S1$ does not need to be modified; the agent should still put whatever it is holding into the lunchbox. We then modify $S2$ so that this time, picking up the banana leads back to $S2$, whereas picking up the sandwich or burger leads to the next state, $S3$. With these changes, the agent will not pick up the banana but will instead pick up either the sandwich or burger.

Each specification was tested 20 times on our experimental platform; as shown in Table 2 there were only a few failures, and these were all due to mechanical failures of the Jaco arm, such as the manipulator dropping an object or losing its grasp on the cabinet key.

| | **Lunchbox** | | | | **Cabinet** | |
|---|---|---|---|---|---|---|
| | $\phi_{l1}$ | $\phi_{l2}$ | $\phi_{l3}$ | $\phi_{l4}$ | $\phi_{c1}$ | $\phi_{c2}$ |
| Successes out of 20 | 20 | 20 | 19 | 19 | 20 | 17 |

Table 2: Performance of Jaco robot in executing learned lunchbox and cabinet tasks

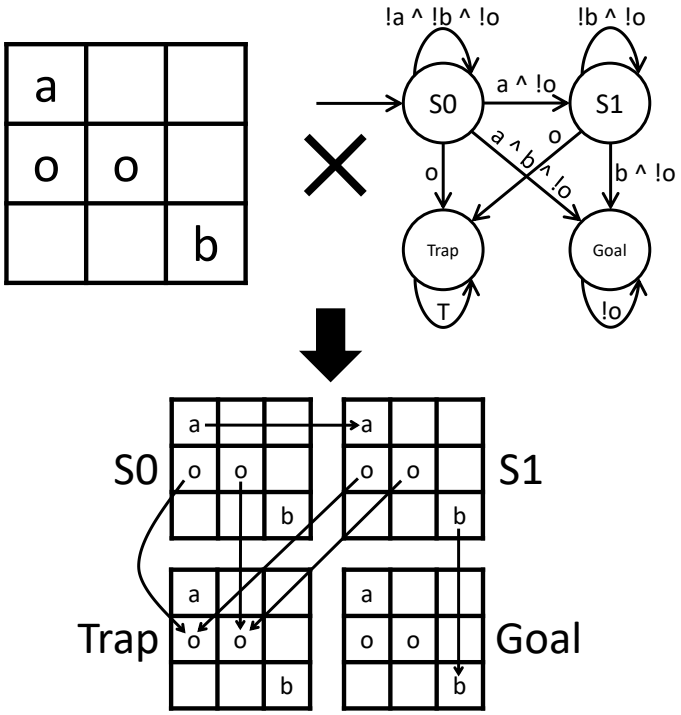



Figure 2: An illustration of how an MDP and an FSA create a product MDP. The MDP is a 2D gridworld with propositions $a$, $b$, and $o$. The FSA describes the rules "go to $a$, then $b$, and avoid $o$. The resulting product MDP represents how these rules interface with the 2D gridworld.

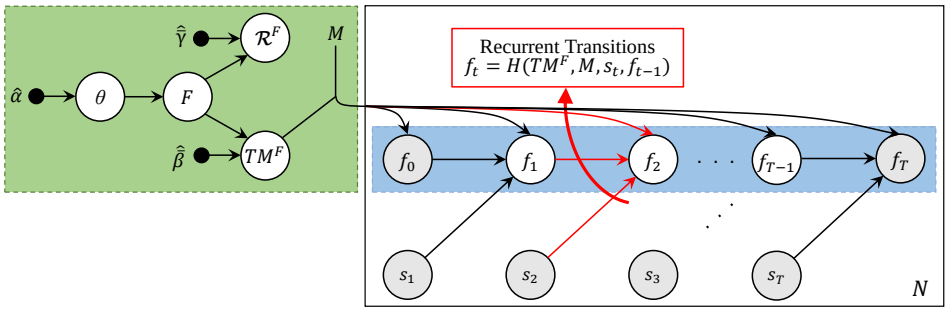
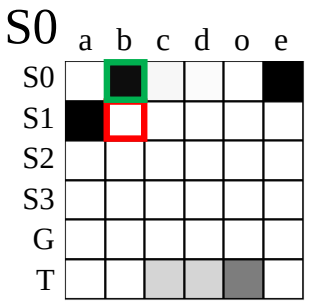
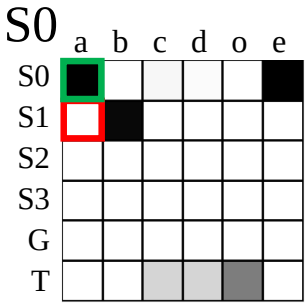


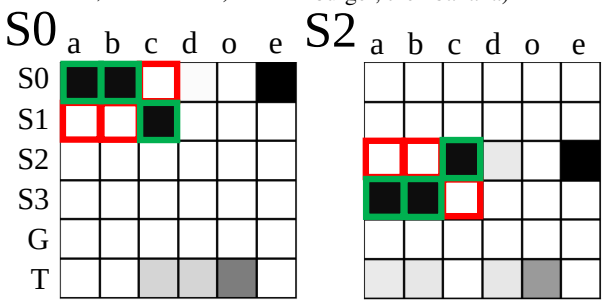

Figure 3: The graphical model of the variational approximation.




(a) $\phi_{p1} \rightarrow \phi_{p2}$ (pick up only sandwich, then banana)




(b) $\phi_{p1} \rightarrow \phi_{p3}$ (pick up only burger, then banana)

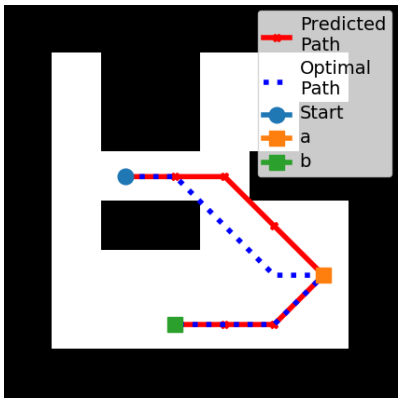




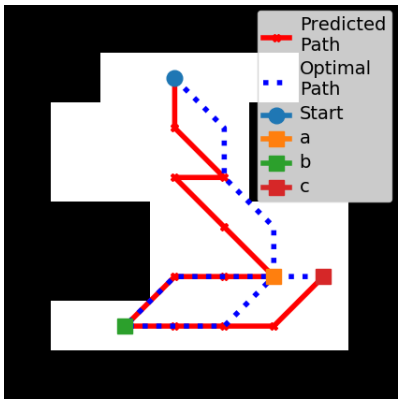


(c) $\phi_{p1} \rightarrow \phi_{p4}$ (pick up banana, then sandwich/burger)

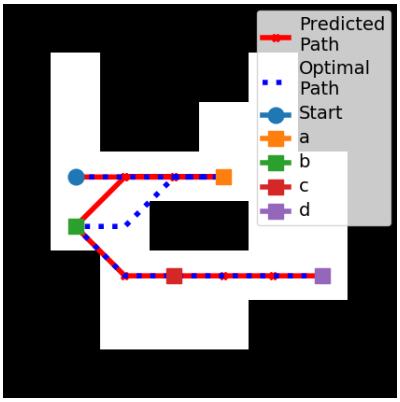
Figure 4: Modifications to the learned TM of the lunch-box domain so that the agent follows the new specifications. Deleted values are marked in red, and added values in green.




(a) Gridworld1 domain

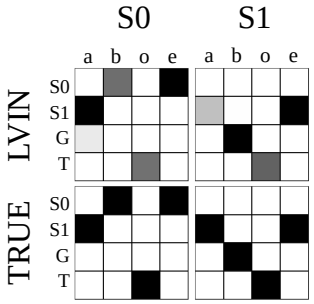



(b) Gridworld2 domain

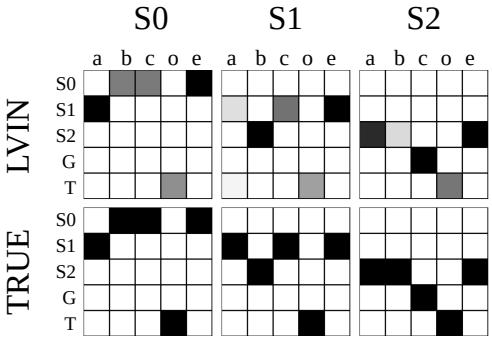



(c) Gridworld3 domain

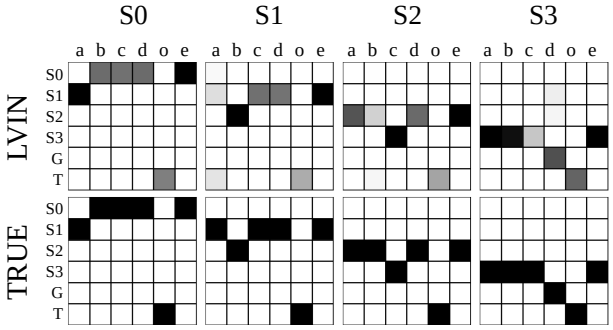
Figure 5: The gridworld domains




(a) Gridworld1 TM

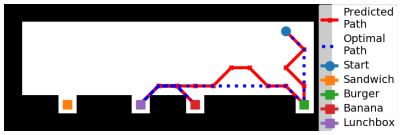



(b) Gridworld2 TM

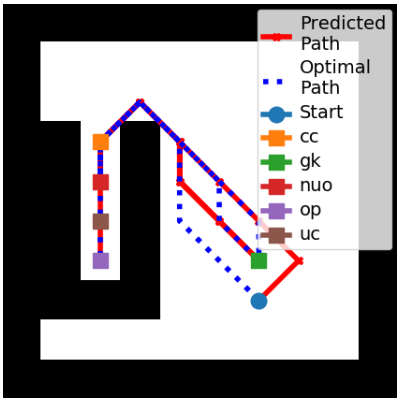



(c) Gridworld3 TM

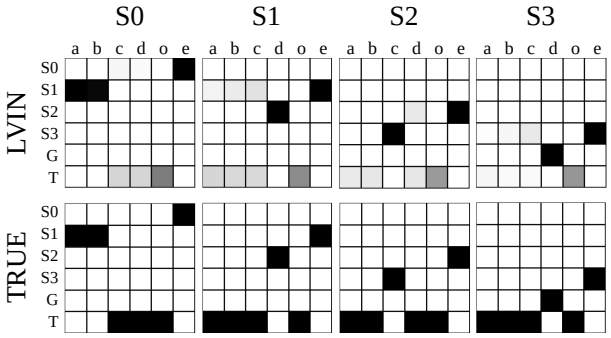
Figure 6: Learned vs true TMs for the gridworld domains



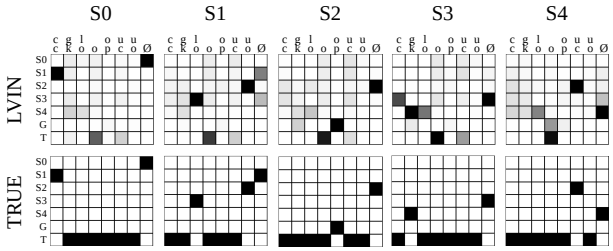

(a) Lunchbox domain




(b) Cabinet domain




(a) Lunchbox TM




(b) Cabinet TM

Figure 9: Learned vs. true TMs for the lunchbox and cabinet domains

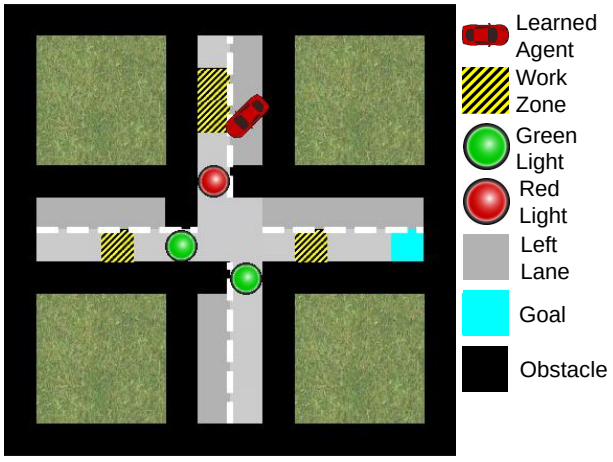



Figure 8: Driving domain

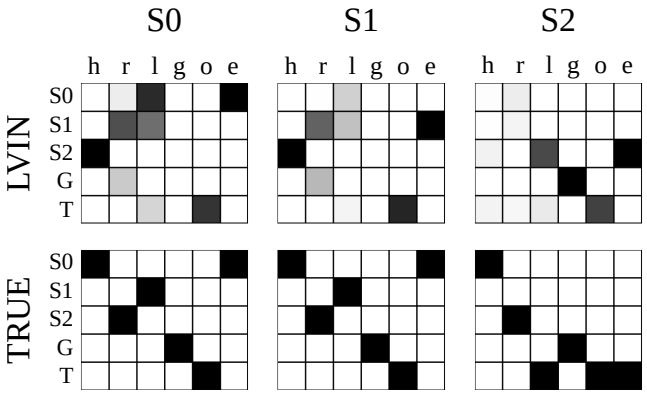



Figure 10: Learned vs. true TMs for the driving domain

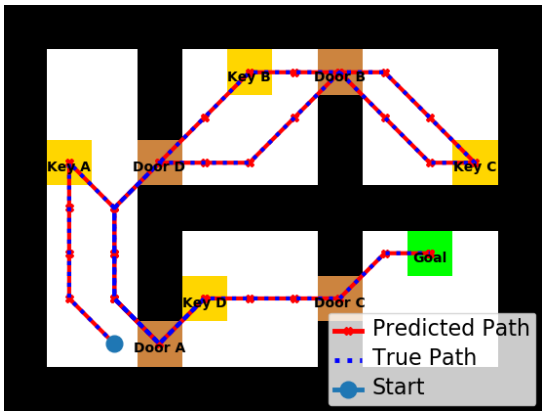



Figure 11: Dungeon domain

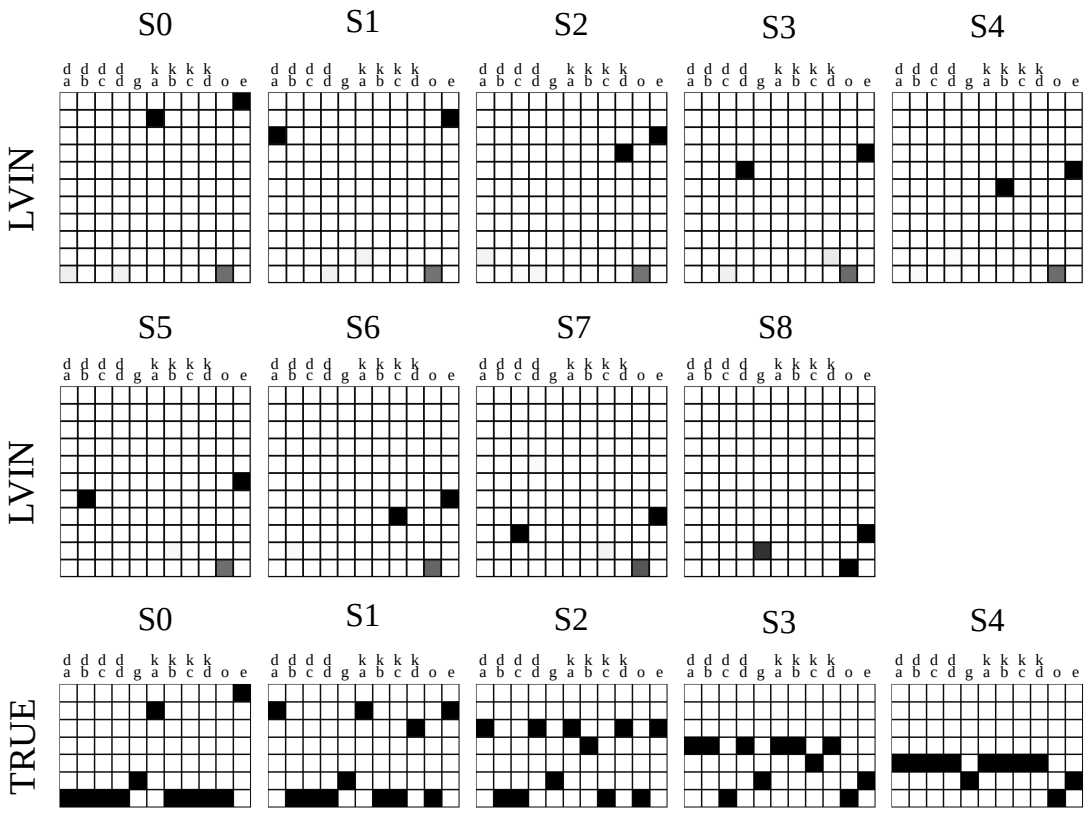
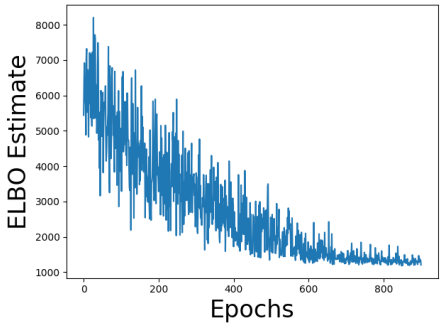
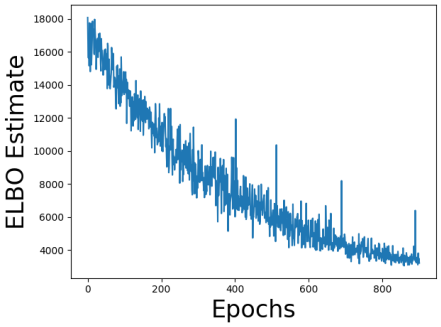


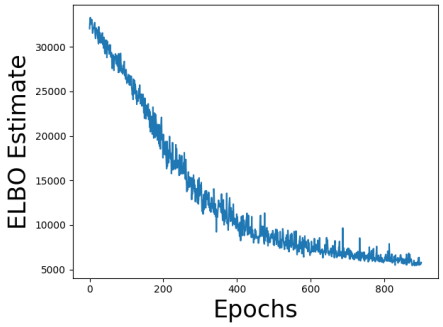

Figure 12: Learned vs. true TM for the dungeon domain



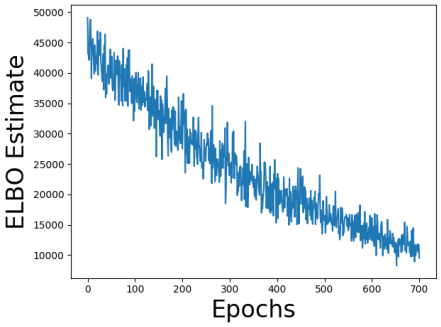

(a) Gridworld1 domain



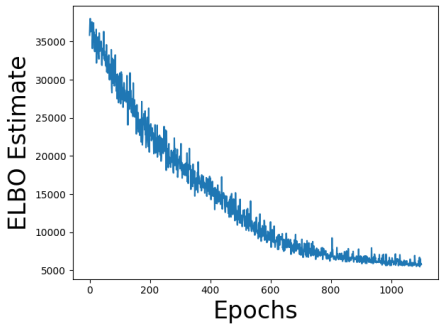

(b) Gridworld2 domain



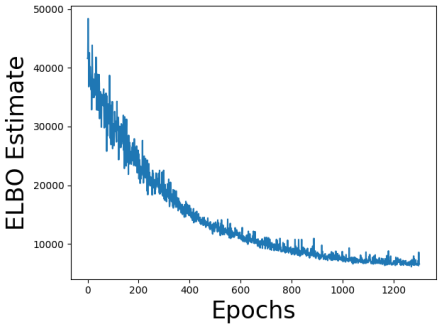

(c) Gridworld3 domain



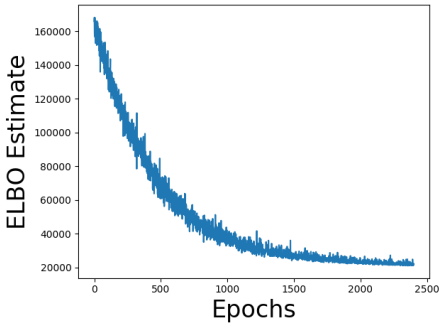

(d) Lunchbox domain




(e) Cabinet domain




(f) Driving domain




(g) Dungeon domain

Figure 13: Loss in terms of an estimate of the ELBO over epochs for each domain.

| Domain | Runtime (hours) |
|---|---|
| Gridworld1 | 1.35 |
| Gridworld2 | 4.04 |
| Gridworld3 | 5.89 |
| Lunchbox | 3.94 |
| Cabinet | 33.24 |
| Driving | 13.77 |
| Dungeon | 179.97 |

Table 3: Runtimes of the seven domains.

| **Variable** | **Interpretation** |
|---|---|
| $\alpha \in \mathbb{R}^{2P}_{>0}$ | Dirichlet prior for number of FSA states $F$ |
| $\bar{\beta} \in (\beta^1, \ldots, \beta^{2P})$ | The bar over $\bar{\beta}$ indicates that it is a list. The list is over TM priors for a given number of FSA states. |
| $\beta^i \in \mathbb{R}^{i \times P \times i}$ | The superscript $i$ represents the number of FSA states. $\beta^i$ is the prior for the transition matrix TM. It is a collection of $i \times P$ Dirichlet priors of $i$ parameters each. In other words, given current FSA state $f$ and proposition $p$, $TM(f, p)$ returns a distribution over possible next FSA states. |
| $\bar{\gamma} \in (\gamma^1, \ldots, \gamma^{2P})$ | The bar over $\bar{\gamma}$ indicates that it is a list. The list is over reward function priors for a given number of FSA states. |
| $\gamma^i \in \mathbb{R}^i \times \mathbb{R}^F_{>0}$ | The superscript $i$ represents the number of FSA states. $\gamma^i$ is the prior for the reward function $\mathcal{R}$ drawn from a Normal distribution. The first set of $i$ real numbers defines the means for each FSA state; the next set of positive real numbers defines the variance for each state. |
| $\theta \sim \text{Dirichlet}(\alpha)$ | Distribution over number of FSA states $F$. |
| $F \sim \text{Categorical}(\theta)$ | Number of FSA states. |
| $TM^F \vert \beta^F \sim \text{Dirichlet}(\beta^F)$ | The transition matrix TM given $F$ FSA states. |
| $\mathcal{R}^F \vert \gamma^F \sim \text{Normal}(\gamma^F)$ | The reward function given $F$ FSA states. Note that in this paper, we assume that the reward function is only a function of current FSA state $f$ and not low-level states $s$ or actions $a$. |
| $\pi := \text{LVIN}(TM^F, \mathcal{R}^F)$ | $\pi$ represents the policy found using the LVIN algorithm. |
| $a_t \vert s_{t-1}, f_{t-1} \sim \pi(s_{t-1}, f_{t-1})$ | The action taken at time $t$. |
| $s_t := T(a_t, s_{t-1})$ | $s_t$ is the low-level state $(x, y)$ at time $t$ after taking action $a_t$. $T$ is the deterministic low-level transition function. |
| $p_t := M(s_t)$ | $p_t$ is the proposition that is true at time $t$. In this work, a proposition is true if the agent is on top of a grid point $(x, y)$ that is associated with the proposition. Proposition map $M$ is the function that associates each low-level state $s$ with a single proposition $p$. |
| $f_t \vert p_t, f_{t-1}, TM^F \sim \text{Categorical}(TM^F(f_{t-1}, p_t))$ | $f_t$ is the FSA state at time $t$. |

Table 4: Notation used in the definition of the Bayesian model.

| Variable | Interpretation |
|---|---|
| $\hat{\alpha}$ | Variational parameter/Dirichlet prior for number of FSA states $F$. Equivalent to $\alpha$. |
| $\hat{\bar{\beta}}$ | List of variational parameters/Dirichlet priors for TM. Equivalent to $\bar{\beta}$. |
| $\hat{\beta}^i$ | Variational parameter/Dirichlet prior for TM given $i$ FSA states. Equivalent to $\beta^i$. |
| $\hat{\bar{\gamma}}$ | List of variational parameters/Normal priors for the reward function. Equivalent to $\bar{\gamma}$. |
| $\hat{\gamma}^i$ | Variational parameter/Normal prior for reward function given $i$ FSA states. Equivalent to $\gamma^i$. |
| $s_t^i$ | Low-level state $(x, y)$ at time $t$ for trajectory $i$ |
| $a_t^i$ | Action at time $t$ for trajectory $i$ |
| $d_i = \langle (s_0^i, a_0^i), \ldots, (s_{T_i}^i, a_{T_i}^i) \rangle$ | Trajectory $i$, composed of state-action pairs up to time step $T_i$. |
| $\mathcal{D} = \langle d_0, \ldots, d_N \rangle$ | Dataset of $N$ trajectories. |
| $f_t^{i,F}$ | The current FSA state at time $t$ for trajectory $i$ given number of FSA states $F$ |

Table 5: Additional notation used in the definitions of the joint likelihood function and the variational approximation.