Approximate Latent State Transitions for Model-Based Reinforcement Learning

Ryan Theisen

***Abstract*— We explore a technique to model state transitions for use in model-based reinforcement learning. We use a latent encoding of the state space to model a transition distribution between states conditioned on actions. We show that, in addition to being used in a fully model based setting, this method can be used to augment a variety of different reinforcement learning techniques.**

I. INTRODUCTION

In model based reinforcement learning (RL), the representative agent is assumed to have full knowledge of the state-transition distribution $p(s'|s,a)$. For example, in Q-learning, the agent can learn the optimal policy π by iteratively applying the Bellman backup equation:

$$Q(s,a) = R(s) + \gamma \mathbb{E}_{s' \sim p(s'|s,a)}[\text{argmax}_{a'} Q(s',a')]$$

where $\pi(s) = \text{argmax}_a Q(s,a)$. However, in most real-world cases, the state transition dynamics are unknown, leading to alternate approaches to the RL problem, which are unable to make use of the information provided by $p(s'|s,a)$.

We propose a new method for estimating the state transition distribution, except defining the transition instead on a latent representation $z \in \mathcal{Z}$ of the actual state $s \in \mathcal{S}$. Approaches to encoding input in such a way have been widely studied in the machine learning literature, and have been shown to be useful estimators of the latent data manifold. For examples, in variational autoencoders (VAEs), the latent distribution $p(z|x)$ is derived by minimizing a variation lower bound of the data likelihood. We employ a similar approach to encode the state s into a latent distribution $p(z|s)$, which is in turn used to estimate a distribution $p(z'|z,a)$ that approximates the transition dynamics $p(s'|s,a)$.

II. BACKGROUND

to do

III. GENERATING APPROXIMATE LATENT STATE TRANSITIONS

A. General Method

Let (s,a,s'), be a sample trajectory where $s' \sim p(s'|s,a)$. We would like that $p(z'|z,a)$ models the transition dynamics $p(s'|s,a)$ as closely as possible. We model this with the following

$$\mathcal{L}_T(\theta,\phi) = \frac{1}{N}\sum_{i=1}^{N} \|\mathbb{E}[f_\theta(s')] - z_i'\|_2$$

where $z_1', ..., z_N' \sim p(z'|z,a)$. Here we consider s' to be a point estimate of $\mathbb{E}[p(s'|s,a)]$, with $f_\theta(s') \sim p(z|s')$ the latent encoding of this estimate.

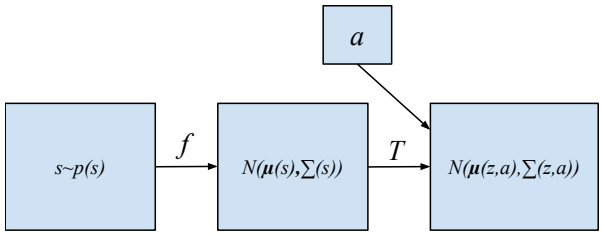

Fig. 1. A diagram of the latent transition model. In practice, f_θ and T_ϕ are parameterized as neural networks that output the mean and covariance of Gaussian distributions.

Furthermore, we would like that the latent encoding $p(z|s)$ encodes as little information about the state as possible while still being sufficient to recover the transition. One way to do this is to penalize mutual information between the encoding z and the state s; that is, we want to make

$$\begin{aligned}\mathcal{L}_f(\theta) = I(z;s) &= \mathbb{E}_{s\sim p(s)}[D_{KL}(p_\theta(z|s)\|p(z)] \\ &\approx \frac{1}{M}\sum_{j=1}^{M} D_{KL}(p_\theta(z|s_i)\|p(z))\end{aligned}$$

where $p(z)$ is a prior on the latent encoding z. Such a penalty is known to be an effective regularizer of the state-encoder f_θ.

We then consider the aggregate objective

$$\mathcal{L}(\theta,\phi) = \mathcal{L}_T(\theta,\phi) + \beta\mathcal{L}_f(\theta)$$

B. Special case: $\mathcal{A}$ finite and $p(z|s)$, $p(z'|z,a)$ Gaussian

As stated above, if the action space $\mathcal{A} = \{a_1, ..., a_M\}$ is finite, and actions are taken at randomly, we have

$$p(z'|z) = \frac{1}{M}\sum_{j=1}^{M} p(z'|z,a_j)$$

In the case when $p(z'|z,a)$ is Gaussian, we have that $p(z'|z)$ is a finite sum of Gaussians and hence Gaussian itself. Furthermore, if the prior $p(z')$ is also Gaussian, then the KL divergence term can be computed analytically. The same is true whenever $p(z|s)$ and $p(z)$ are both Gaussians (see appendix).

C. Trajectory Estimation and Planning

One useful property of our design if that, if $p(z)$ and $p(z')$ are chosen to correspond, and $p(z'|z)$ and $p(z|s)$ are minimal, then one can consider estimating multiple steps into the future by applying $p(z'|z,a)$ iteratively. That is, given

$z \sim p(z)$, one has that $z' \sim p(z'|z)$ can be used to sample $z'' \sim p(z''|z', a)$ given some action a.

It would be nice to have some theory justifying this, rather than just intuition

IV. EXAMPLE: MODEL-BASED Q-LEARNING

A. Estimating Reward Function

Given trajectories (s, a, s', r) and a state encoder $f : \mathcal{S} \to \mathcal{Z}$, we can estimate a reward function $R : \mathcal{S} \to \mathbb{R}$ with $R(s) = R_\theta \circ f(s)$ where $R_\theta : \mathcal{Z} \to \mathbb{R}$ is taken to be a neural network with parameters θ (see Deep Mind paper). $R_\theta(z)$ can then be used to evaluate rewards of latent state transitions. R_θ can be trained straightforwardly using supervised learning.

B. Monte Carlo Model-Based Q Learning

Consider the standard Bellman Equation:

$$Q(s, a) = R(s) + \gamma \mathbb{E}_{s' \sim p(s'|s,a)}[\text{argmax}_{a'} Q(s', a')]$$

Under the hypothesis that the representation z is sufficient for the given task, we can reframe this problem in terms of the latent encoding z:

$$\begin{aligned} Q(z, a) &= R(z) + \gamma \mathbb{E}_{z' \sim p(z'|z,a)}[\text{argmax}_{a'} Q(z', a')] \\ &= R(z) + \gamma \int_{\mathcal{Z}} \text{argmax}_{a'} Q(z', a') p(z'|z, a) dz' \end{aligned}$$

Since $p(z'|z, a)$ is known, we can estimate this integral using a Monte Carlo approach:

$$\begin{aligned} Q(z, a) &= R(z) + \gamma \int_{\mathcal{Z}} \text{argmax}_{a'} Q(z', a') p(z'|z, a) dz' \\ &\approx R(z) + \frac{\gamma}{N} \sum_{k=1}^{N} \text{argmax}_{a'} Q(z'_k, a') \end{aligned}$$

where $z'_1, ..., z'_N \sim p(z'|z, a)$ and $R(z) \equiv R_\theta(z)$ is approximated by the method outlined in B.

Using this set up, we can estimate Q using a modified version of the DQN algorithm, minimizing the objective

$$\mathcal{L} = \frac{1}{2}\Big(Q(z, a) - \big(R(z) + \frac{\gamma}{N} \sum_{k=1}^{N} \text{argmax}_{a'} Q(z'_k, a')\big)\Big)^2$$

Note that, unlike the standard DQN algorithm, given the distribution $p(z'|z, a)$ and the function $R(z)$, this formulation can be trained entirely off-line, with no active interaction with the environment.

V. EXPERIMENTS

VI. CONCLUSION AND FUTURE WORK