Deep Recurrent Belief Propagation Network for POMDPs

Yuhui Wang, Xiaoyang Tan
College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics
MIIT Key Laboratory of Pattern Analysis and Machine Intelligence
{y.wang, x.tan}@nuaa.edu.cn

Abstract

In many real-world sequential decision-making tasks, especially in continuous control like robotic control, it is rare that the observations are perfect, that is, the sensory data could be incomplete, noisy or even dynamically polluted due to the unexpected malfunctions or intrinsic low quality of the sensors. Previous methods handle these issues in the framework of POMDPs and are either deterministic by feature memorization or stochastic by belief inference. In this paper, we present a new method that lies somewhere in the middle of the spectrum of research methodology identified above and combines the strength of both approaches. In particular, the proposed method, named Deep Recurrent Belief Propagation Network (DRBPN), takes a hybrid style belief updating procedure – an RNN-type feature extraction step followed by an analytical belief inference, significantly reducing the computational cost while faithfully capturing the complex dynamics and maintaining the necessary uncertainty for generalization. The effectiveness of the proposed method is verified on a collection of benchmark tasks, showing that our approach outperforms several state-of-the-art methods under various challenging scenarios.

Introduction

Significant progress has been made in reinforcement learning (RL) to solve a large number of tasks, such as Atari games, board games and robotic control (Mnih et al. 2015; Silver et al. 2017; Schulman et al. 2016). These methods usually assume the states of the environment are fully observable. However, in realistic control tasks, the observation data could be *incomplete* and *noisy*. Especially, we refer *incomplete* to that the the observation is *dynamically missing*, which means that the missing components are changing over timestep. This problem is very common in realistic continuous control task. Take robotic control as an example. The agent perceives information of environment through sensors, e.g., position sensor and velocity sensor. The incompleteness could result from any malfunction in the sensor, too much time of preprocessing, or intrinsically, the sensors' different sampling frequency from each other (Randlv and Alstrm 1997). In addition, due to the low quality of sensors, the observation could also be contaminated with noise.

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The incomplete and noisy data may lack information which is critical for an agent to make the right decision. However, in practice, the agents often have to take actions continuously even when their perception is not perfect. For example, a running automatic driving car or robot cannot suddenly stop itself and then move again simply due to that one or two frames from its radar perception sensors are missing or partially occluded. In such cases, traditional methods addressing this issue (e.g., simply ignore the imperfect state information or set a default action when that happens) are not appropriate and could even bring disaster to the system. Consequently, there is a great need to tackle such problems.

Such incomplete and noisy observation problem in continuous control can be formalized as a case of partially observable Markov decision processes (POMDPs). The recent most successful scheme for POMDPs is to aggregate the history data over time. There are mainly two types of approaches to achieve this goal. The first type is to remember the features of the past, e.g., by employing Recurrent Neural Networks (RNNs) (Hausknecht and Stone 2015; Zhu, Li, and Poupart 2017; Karkus, Hsu, and Lee 2017). These RNN-based methods are computational efficient and can be trained end-to-end. However, these methods remember past by deterministic feature computation and do not explicitly incorporate the knowledge of the learned environment models. The second type are probabilistic methods which infer the distribution of the latent state (called *belief*) from history of partially observable data. Compared to the first type, this type of methods is able to characterize the uncertainty of the knowledge about the current state. However, this is achieved at the cost of requiring both the transition and observation model, which are either assumed to be known (McAllester and Singh 1999; Ross et al. 2008; Roijers, Whiteson, and Oliehoek 2015) or learned. Previous methods usually make strong assumption on the generative model to allow tractable belief updating (Doshi-Velez et al. 2015; Katt, Oliehoek, and Amato 2017; Azizzadenesheli, Lazaric, and Anandkumar 2016).

In this paper, we propose a new method for POMDPs with incomplete and noisy observation, named Deep Recurrent Belief Propagation Network (DRBPN). The DRBPN combines the strength of both previous two types of approaches, i.e., it is computationally efficient but preserves the uncertainty for belief description as well. In particular, to balance the subtle tradeoff between efficiently updating the belief and accurately characterizing the dynamics itself, we address the former by clamping the belief as some parametrized form and focus on modelling the latter with a highly nonlinear Deep Neural Network (DNN). This leads to a hybrid type of belief updating – an RNN-type feature extraction step followed by an analytical belief inference. Compared to those methods that directly model the whole belief distribution (Igl et al. 2018), our method significantly reduces the computational cost (through analytical inference), while faithfully capturing the complex environment dynamics and maintaining the necessary uncertainty for generalization. Last but not least, our computational framework can be trained in an end-to-end learning scheme, thus can be thought of as a new strategy for directly applying the policy gradient in the POMDP setting. Extensive experiments on high dimensional benchmark tasks show that our approach outperforms several state-of-the-art methods under various challenging POMDP scenarios.

Related Work

Most methods working on POMDPs do planning with known transition model, generative model of observation (McAllester and Singh 1999; Ross et al. 2008; Roijers, Whiteson, and Oliehoek 2015). However, these models are not known in realistic scenarios. Previous work also tried to learn these models based on a Bayesian approach (Katt, Oliehoek, and Amato 2017; Ross et al. 2011) or nonparametric methods (Doshi-Velez et al. 2015). However, these methods could hardly scale to large or continuous state and observation spaces.

Our DRBPN has a similar recurrent structure as Recurrent Neural Network (RNN), which is employed by several methods focusing on POMDPs (Hausknecht and Stone 2015; Zhu, Li, and Poupart 2017; Zhang et al. 2015). However, our DRBPN method explicitly employs the knowledge of environment models, which could aid learning and inference. Besides, our algorithm can explicitly reason about the transition and observation model.

Our work is most related to DVRL (Igl et al. 2018), which also adopts the same fundamental idea of maintaining belief and learning the models simultaneously as ours. Apart from the approximated environment models, DVRL needs to train an additional inference model to approximate the belief inference procedure. Whereas in our method, the belief inference is solved analytically. By avoiding co-training an additional inference model with the learned environment models, our method could reduce the difficulty of training. In addition, to approximate the intractable computation of the belief, DVRL relies on a particle filter method, which could increase computation cost.

There are several works which also clamp the state distribution to be a parametrized Gaussian to enable efficient inference. PILCO employs the Gaussian Process (GP) to learn the transition model (Deisenroth and Rasmussen 2011). McAllister and Rasmussen (2017) extended PILCO with Bayesian filtering to enable learning under observation noise. However, these methods could only work with complete observation to our knowledge. Besides, the relying on the GPs prohibits their applicability to problems that require a large number of sample data. While our DRBPN benefits from the the Gaussian parametrized by DNNs which scales linearly with the number of data.

Guo et al. and Gregor et al. uses predictive learning to learn a belief representation (Guo et al. 2018; Gregor et al. 2019). Several works focus on deep filtering (Karl et al. 2016; Lim, Zohren, and Roberts 2019; Becker et al. 2019; Willi et al. 2019). Many of them rely on complicated approximate inference techniques which may hurt the efficiency and scalability (Karl et al. 2016; Lim, Zohren, and Roberts 2019). While we primarily aim to use efficient and accurate inference with imperfect perception for robust control.

Preliminaries

Reinforcement Learning

Partially Observable Markov Decision Processes (POMDP). A POMDP is described as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{P}, r, \gamma)$. $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{O}$ represent the state space, action space and observation space respectively; $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the transition probability; $\mathcal{P}(x_t|s_t, a_{t-1})$ is the observation probability distribution where $x_t \in \mathcal{O}$, $s_t \in \mathcal{S}$ and $a_{t-1} \in \mathcal{A}$; $r : \mathcal{S} \times A \to \mathbb{R}$ is the reward function and $\gamma \in (0, 1)$ is the discount factor.

Proximal Policy Optimization (PPO). PPO is a prominent policy optimization algorithm for Markov Decision Process (which is a special case of POMDP that $\mathcal{O} = \mathcal{S}$ and $x_t = s_t$) (Schulman et al. 2017). PPO learns the parameter ζ of the policy π_ζ by optimizing a clipped "surrogate" objective

$$\begin{aligned} L^p(\zeta) = \mathbb{E}_{s_t, a_t} \Bigg[\min \Bigg(& \frac{\pi_\zeta(a_t|s_t)}{\pi_{\zeta_{old}}(a_t|s_t)} \hat{A}_t, \\ & clip\left(\frac{\pi_\zeta(a_t|s_t)}{\pi_{\zeta_{old}}(a_t|s_t)}, 1-\delta, 1+\delta \right) \hat{A}_t \Bigg) \Bigg], \end{aligned} \tag{1}$$

where δ is the clipping parameter; $\hat{A}_t = \sum_{k=0}^{\infty} (\gamma\lambda)^k \left[r_t + \gamma \hat{V}_\phi(s_{t+k+1}) - \hat{V}_\phi(s_{t+k}) \right]$ is the Generalized Advantage Estimator with a trade-off coefficient λ (Schulman et al. 2016); and $\hat{V}_\phi$ is the approximated value function, which is trained by $L^v(\xi) = \mathbb{E}_{s_t} \left\| \hat{V}_\xi(s_t) - \hat{V}_t \right\|^2$, where $\hat{V}_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the discounted accumulated reward from timestep t onwards. PPO has achieved great success across a wide range of challenging MDP tasks. However, for POMDPs case, the agent needs to aggregate all history data to produce enough information which is critical for policy π_ζ to output correct action.

Generative Model of Observations

The traditional POMDPs work on the case where the observation $x_t \in \mathbb{R}^D$ is noisy but complete. Whereas we devote to a more challenging case that the observation $x_t \in (\mathbb{R} \cup \{*\})^D$ is also incomplete, where "$*$" means the data on that component is missing. Following Little and Rubin (2019), the generative process of the observation model

$\mathcal{P}(x_t|s_t, a_{t-1})$ can be described below,

$$
\begin{aligned}
x_t^{\text{complete}} &\sim P_\theta(x_t^{\text{complete}}|s_t, a_{t-1}),\\
m_t &\sim P_\eta(m_t|x_t^{\text{complete}}, a_{t-1}),\\
x_t &= x_t^{\text{complete}} \odot m_t
\end{aligned}
\tag{2}
$$

where $m_t \in \{0,1\}^D$ is the missing indicator vector (1 for observed, 0 for missing), the operator $\odot$ is defined as $(x \odot m)^{(i)} \triangleq \begin{cases} x^{(i)} & m^{(i)} = 1 \\ * & m^{(i)} = 0 \end{cases}$, θ and η denote the unknown parameters of the corresponding distributions. We now illustrate the specific models used in this paper.

We adopt the commonly used linear Gaussian for observation modelling (Karl et al. 2016; Becker et al. 2019). Formally, the generation of the complete observation is modelled as $x_t^{\text{complete}} = \theta_s s_t + \theta_a a_{t-1} + \theta_b + \epsilon_t$, where $\theta_s \in \mathbb{R}^{D \times dim(\mathcal{S})}$, $\theta_a \in \mathbb{R}^{D \times dim(\mathcal{A})}$, $\theta_b \in \mathbb{R}^D$ are parameters to be learned and $\epsilon_t \sim \mathcal{N}(0, \Sigma^\epsilon)$ is an additive noise. Let $\theta \triangleq (\theta_s, \theta_a, \theta_b, \Sigma^\epsilon)$ denote all the parameters to be learned, we have

$$
\begin{aligned}
&P_\theta(x_t^{\text{complete}}|s_t, a_{t-1})\\
=&\mathcal{N}(x_t^{\text{complete}}|\theta_s s_t + \theta_a a_{t-1} + \theta_b, \Sigma^\epsilon)
\end{aligned}
\tag{3}
$$

Note that although the observation is modelled to be linearly w.r.t. the latent state, a non-linear DNN is employed to learn the latent state and the models are trained end-to-end. With these highly non-linear models, it is sufficient to learn a compact feature of the latent state which satisfies the linear assumption. We illustrate the detail in Sec .

As with Little and Rubin (2019), the missing mechanism $P_\eta(m_t|x_t^{\text{complete}}, a_{t-1})$ is characterized in terms of independence relations between the missing indicator m_t and the data $x_t^{\text{complete}} = (x_t^o, x_t^m)$, where x_t^o and x_t^m are the observed and the missing elements of x_t^{complete} respectively. The relations can be roughly divided into two groups:

1) The missingness depends on the unobserved components of data, including

- Not missing at random (NMAR): m_t depends on both x_t^o and x_t^m (and also a_{t-1}).

2) The missingness does not depend on the values of the unobserved components of data, including

- Missing completely at random (MCAR): $P_\eta(m_t|x_t^{\text{complete}}, a_{t-1}) = P_\eta(m_t|a_{t-1})$[1],
- Missing at random (MAR): $P_\eta(m_t|x_t^{\text{complete}}, a_{t-1}) = P_\eta(m_t|x_t^o, a_{t-1})$,

Especially, cases like malfunction or different sample frequencies of sensors can be categorized into such setting, in the sense that the missingness depends on factors like the quality of sensor or time. Such setting is used in most of the missing data work (Little and Rubin 2019; Lizotte et al. 2008) and also adopted in this paper, since with these assumptions we have

$$
\begin{aligned}
\mathcal{P}(x_t|s_t, a_{t-1}) =&P(x_t^o, m_t|s_t, a_{t-1})\\
=&P_\theta(x_t^o|s_t, a_{t-1})P_\eta(m_t|x_t^o, a_{t-1})
\end{aligned}
\tag{4}
$$

[1]In our case, the action a_{t-1} is also taken into account.

With such decoupling, the missingness information can be ignored when learn the parameters of the generative model.

To derive the form of $P_\theta(x_t^o|s_t, a_{t-1})$ in eq. (4), let us introduce a *sub-permutation matrix* $M_t \in \mathbb{R}^{dim(x_t^o) \times D}$, which is designed to obtain $x_t^o = M_t x_t$. For example, given $x_t = (1, *, 2)^\top$, the corresponding sub-permutation matrix is $M_t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. M_t is constructed by $M_t^{j, I_t^{(j)}} = 1$ where $I_t = \{i | x_t^{(i)} \neq *\}$ denote the observed indexes and $j = 1, 2, \ldots, |I_t|$ (while other entries are 0). By eq. (3),

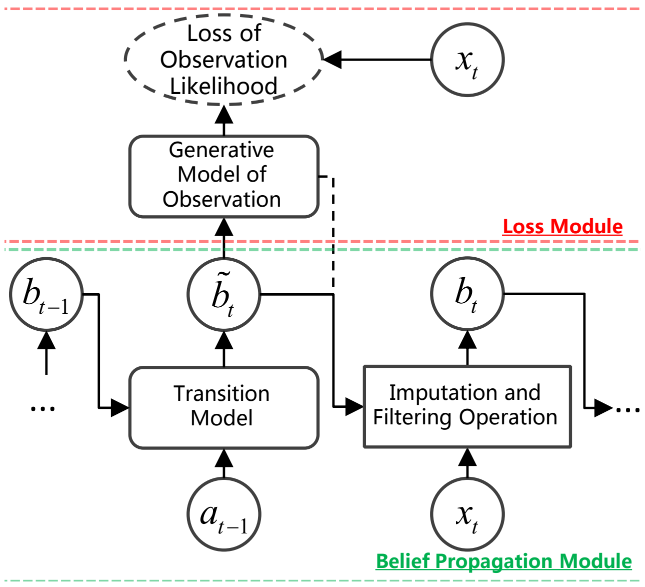
$$
\begin{aligned}
&P_\theta(x_t^o|s_t, a_{t-1})\\
=&\mathcal{N}(M_t x_t | M_t(\theta_s s_t + \theta_a a_{t-1} + \theta_b), M_t \Sigma^\epsilon M_t^\top)
\end{aligned}
\tag{5}
$$

Deep Recurrent Belief Propagation Network

In this section, we present our method, Deep Recurrent Belief Propagation Network (DRBPN). Similar to prior methods for POMDPs (Igl et al. 2018; McAllister and Rasmussen 2017), DRBPN adopts the scheme of maintaining a belief of the state based on past trajectory data and propagating the belief recurrently during the execution phase. The belief computation relies on approximated models of the environment, which are trained end-to-end by the log-likelihood loss. Fig. 1 summarises the architecture of DRBPN.

Belief Propagation

A *belief* is a posterior distribution over possible original state space based on the historical trajectory data, denoted



Loss of Observation Likelihood
x_t
Generative Model of Observation
Loss Module
b_{t-1}
$\tilde{b}_t$
b_t
...
Transition Model
Imputation and Filtering Operation
...
a_{t-1}
x_t
Belief Propagation Module

Figure 1: Overview of DRBPN. The intermediate belief $\tilde{b}_t$ is obtained through the transition model $\mathcal{T}(s_t|s_{t-1}, a_{t-1})$ with the last belief b_{t-1} and the action a_{t-1}; the belief b_t is obtained by an imputation and filtering operation with the observation x_t, the learned generative model of observation, and $\tilde{b}_t$.The models are trained end-to-end by backpropagation of the log likelihood of observations $x_{1:T}$.

as b_t. Given a belief b_{t-1} at timestep $t-1$, an action a_{t-1} is taken based on this belief. Following that, a new observation x_t, which is incomplete and noisy, is received. Then, an *"intermediate" belief* $\tilde{b}_t$ at timestep t can be directly obtained through a learned transition model $\hat{\mathcal{T}}(s_t|s_{t-1}, a_{t-1})$. Next, an imputation and filtering operation on the new observation x_t is applied based on $\tilde{b}_t$. As a result, the new belief b_t is obtained. The entire propagation process starts with an initial intermediate belief $\tilde{b}_1(s_1)$ and an initial observation x_1. Our DRBPN embeds this procedure into the architecture of the network, as Fig. 1 (the green box) depicts.

According to the definition, we have $b_t(s_t) \triangleq P(s_t|x_{1:t}, a_{1:t-1})$. By applying Bayes rule, the updating of belief $b_t(s_t)$ has the form

$$b_t(s_t) = \frac{\mathcal{P}(x_t|s_t, a_{t-1})\tilde{b}_t(s_t)}{\int \mathcal{P}(x_t|s'_t, a_{t-1})\tilde{b}_t(s'_t)ds'_t}, \text{for } t = 1, 2, \dots \tag{6}$$

where $\tilde{b}_t(s_t) \triangleq P(s_t|x_{1:t-1}, a_{1:t-1})$ is called an "intermediate" belief,

$$\tilde{b}_t(s_t) = \int b_{t-1}(s_{t-1})\mathcal{T}(s_t|s_{t-1}, a_{t-1})ds_{t-1}, \text{for } t = 2, 3, \dots \tag{7}$$

where $\mathcal{T}$ is the transition probability function in original state space. The initial intermediate belief $\tilde{b}_1(s_1)$ is an approximated prior belief of initial state. In what follows we will describe in detail the belief propagation procedure adopted in this work and the corresponding approximation which is made to perform an analytical propagation.

First, the intermediate belief $\tilde{b}_t(s_t)$ is obtained by applying Eq. (7) through an approximated transition model. Generally, Eq. (7) can be approximated by replacing the nonlinear $\mathcal{T}$ with first-order (linear) approximation w.r.t. s_{t-1} at $\mathbb{E}_{s_{t-1}\sim b_{t-1}}[s_{t-1}]$, which reduces to $\tilde{b}_t(s_t) \doteq \mathcal{T}(s_t|\mathbb{E}_{s_{t-1}\sim b_{t-1}}[s_{t-1}], a_{t-1})$. In addition, the transition distribution function $\mathcal{T}(s_t|s_{t-1}, a_{t-1})$ is approximated as a conditional multivariate Gaussian $\hat{\mathcal{T}}(s_t|s_{t-1}, a_{t-1}) = \mathcal{N}\left(s_t|f^{\mu}_{\phi}(s_{t-1}, a_{t-1}), f^{\Sigma}_{\phi}(s_{t-1}, a_{t-1})\right)$, in which f^{μ}_{ϕ} and f^{Σ}_{ϕ} are DNNs parameterized by ϕ. Nevertheless, the highly non-linear DNNs are sufficient to capture the complex dynamics of the environment. Therefore, the intermediate state $\tilde{b}_t(s_t)$ is computed by

$$\begin{aligned}\tilde{b}_t(s_t) \doteq \mathcal{N}\Big(s_t|\tilde{\mu}_t = f^{\mu}_{\phi}\left(\mathbb{E}_{s_{t-1}\sim b_{t-1}}[s_{t-1}], a_{t-1}\right),\\ \tilde{\Sigma}_t = f^{\Sigma}_{\phi}\left(\mathbb{E}_{s_{t-1}\sim b_{t-1}}[s_{t-1}], a_{t-1}\right)\Big)\end{aligned} \tag{8}$$

The initial intermediate belief $\tilde{b}_1(s_1)$ is also approximated by $\mathcal{N}(s_1|\tilde{\mu}_1, \tilde{\Sigma}_1)$. It is worth mentioning that in practice the Gaussian belief model may not be as restrictive as one may think in representing uncertainty provided that it is built on good feature representations, which for example can be learnt through layers of non-linear transformations. Indeed, to the best of our knowledge, most of the existing works de facto adopt this assumption, either explicitly or implicitly — even for those who may claim otherwise, the Gaussian appears in their implementation in different forms, through the reparameterization trick, variational approximation, and so on (Igl et al. 2018; Karl et al. 2016; Lim, Zohren, and Roberts 2019; Willi et al. 2019).

Then, a data imputation and filtering operation on the new observation x_t is applied using Eq. (6). According to Eq. (4),

$$b_t(s_t) = \frac{P_\theta(x^o_t|s_t, a_{t-1})\tilde{b}_t(s_t)}{\int P_\theta(x^o_t|s'_t, a_{t-1})\tilde{b}_t(s'_t)ds'_t} \tag{9}$$

Since the likelihood $P_\theta(x^o_t|s_t, a_{t-1})$ is a linear Gaussian (see Eq. (5)) and $\tilde{b}_t(s_t)$ is also a Gaussian, the belief $b_t(s_t)$ can be obtained analytically:

$$\begin{aligned}&b_t(s_t)\\ =&\mathcal{N}\Big(s_t|\mu_t = \tilde{\mu}_t - F_t\left(\theta_s\tilde{\mu}_t + \theta_a a_{t-1} + \theta_b - x_t\right),\\ &\quad\Sigma_t = \tilde{\Sigma}_t - F_t\theta_s\tilde{\Sigma}_t\Big),\end{aligned} \tag{10}$$

where $F_t = \tilde{\Sigma}_t\theta_s^\top M_t^\top\Big[M_t\left(\Sigma^\epsilon + \theta_s\tilde{\Sigma}_t\theta_s^\top\right)M_t^\top\Big]^{-1}M_t$. Let us see how the equation above take effect in data imputation and filtering. For data imputation, $\tilde{\mu}_t$ gives the expectation of s_t; and the covariance matrix $\theta_s\tilde{\Sigma}_t\theta_s^\top$ provides the correlation between the observed and missing components, which makes it possible to infer missing components from the observed ones. For data filtering, the difference term $\theta_s\tilde{\mu}_t + \theta_a a_{t-1} + \theta_b - x_t$ is the difference between the internal belief of the observation and the observed one; and F_t can be regarded as the extent that the difference term is used to update the belief. By the form of F_t, we can know that larger noise covariance Σ^ϵ leads to smaller F_t, which reduces the impact of current observation x_t on the expectation μ_t and enlarges the uncertainty Σ_t.

Finally, the belief b_t is employed by the policy π to decide an action, where we can use the exception μ_t of belief for decision directly, $\pi(b_t) \triangleq \pi(\mu_t)$, or as well incorporate the uncertainty term Σ_t,

$$\pi(b_t) \triangleq \pi(\mu_t, \Sigma_t), \tag{11}$$

End-to-end Learning of the Models

The analytical belief propagation makes it tractable to implement end-to-end learning of the models. For simplification, let $\psi \triangleq (\phi, \theta, \tilde{\mu}_1, \tilde{\Sigma}_1)$ denote all the parameters to be learned for the belief computation. The objective function is the log likelihood of the observations $x_{1:T+1}$:

$$\begin{aligned}&L^m(\psi)\\ =&\log\left[P(x_{1:T+1}|a_{1:T};\psi)\right]\\ =&\log P(x_1;\psi)\prod_{t=2}^{T+1}P(x_t|x_{1:t-1}, a_{1:t-1};\psi)\end{aligned}$$

$$
\begin{aligned}
&= \log \int \tilde{b}_1(s_1;\tilde{\mu}_1,\tilde{\Sigma}_1)\mathcal{P}_\theta(x_1|s_1)ds_1 + \\
&\sum_{t=2}^{T+1}\left[\log \int \tilde{b}_t(s_t;\psi)\mathcal{P}_\theta(x_t|s_t,a_{t-1})ds_t\right] \\
&= \log \mathcal{N}\left(M_1x_1|M_1(\theta_s\tilde{\mu}_1+\theta_b), M_1\left(\theta_s\tilde{\Sigma}_1\theta_s^\top+\Sigma^\epsilon\right)M_1^\top\right) \\
&\quad + \sum_{t=2}^{T+1}\Bigg[\log \mathcal{N}\Big(M_tx_t|M_t(\theta_s\tilde{\mu}_t+\theta_a a_{t-1}+\theta_b), \\
&\qquad M_t\left(\theta_s\tilde{\Sigma}_t\theta_s^\top+\Sigma^\epsilon\right)M_t^\top\Big)\Bigg] \quad (12)
\end{aligned}
$$

The architecture of the loss module is demonstrated in Fig. 1. The belief propagation network is integrated into the policy search scheme. Especially, we employ PPO method as the policy search algorithm. The input to the policy and value network of PPO is modified to be the belief b_t (see Eq. (11)). All the three networks, the policy network, the value network, and the transition network can share parameters with each other and be trained jointly. This architecture could aid policy learning by the auxiliary learning tasks of the environment models. The overall loss function is

$$L^{\text{DRBPN}}(\zeta,\xi,\psi) = -L^p(\zeta) + \lambda_v L^v(\xi) - \lambda_m L^m(\psi) \quad (13)$$

where λ_v and λ_m are the coefficients to trade-off the losses. We adopt the normalized advantage values and rewards to train policy and value network, therefore all the three terms have relatively similar magnitude across different tasks. Our DRBPN algorithm is presented in Algorithm 1.

Algorithm 1 DRBPN

```
Input: Hyperparameters of loss function $\lambda_v$ and $\lambda_m$.
    Hyperparameters of training TIMESTEPS_MAX, $T$, Epoches_Max.
  for $i = 1$ to TIMESTEPS_MAX$/T$ do
    // Execution Phase
    Receive an observation $x_1$ from the environment
    Start with $\tilde{b}_1 = \mathcal{N}(\tilde{\mu}_1, \tilde{\Sigma}_1)$
    for $t = 1$ to $T$ do
      Update the belief $b_t = \mathcal{N}(\mu_t, \Sigma_t)$ by (10)
      Execute $a_t = \pi(b_t)$ by (11)
      Receive a reward $r_t$ and a new observation $x_{t+1}$
      Update the next intermediate belief $\tilde{b}_{t+1}$ by (8)
    end for
    // Training Phase
    for $k = 1$ to Epoches_Max do
      Compute $\hat{A}_t$ and $\hat{V}_t$ for $t = 1, \dots, T$
      Train the networks by (13) with
      $(\{x_t, a_t, \hat{A}_t, \hat{V}_t\}_{t=1}^T, x^{T+1})$
    end for
  end for
Output: $(\zeta, \xi, \psi)$, which are the parameters of policy,
    value, and belief networks respectively.
```

Experiments

We designed experiments to answer the following questions:

1. Could DRBPN be robust in POMDPs with incomplete and noisy observations? And to what extent could DRBPN be robust?
2. Is DRBPN more computational efficient compared to other POMDPs methods?
3. Is the new architecture able to learn better policy or improve sample efficiency?

To answer 1 and 2, we evaluate DRBPN under a POMDP case with different settings of incompleteness and noise, discussed in the first section. Concerning 3, we compare DRBPN with several prior policy optimization algorithms under a fully observable case, discussed in the second section.

We implement our DRBPN algorithm by extending PPO. DRBPN adopts the same hyperparameters of the policy search components of PPO given in (Dhariwal et al. 2017), except that an additional transition network in DRBPN is set up. The covariance of the transition are state-independent and is a parameter of matrix, denoted as $\tilde{\Sigma}$ (thus we have $\tilde{\Sigma}_t = \tilde{\Sigma}$ for all t). We use ReLU as the activation function. We empirically set the penalty coefficient in Eq. (13) to be $\lambda_v = 1.0, \lambda_m = 1.0$. We evaluated the methods on 8 benchmarks simulated locomotion tasks, which is implemented in OpenAI Gym (Brockman et al. 2016) using the MuJoCo physics engine (Todorov, Erez, and Tassa 2012).

We compare our method against the following algorithms: 1) *Deep Variational RL (DVRL)* (Igl et al. 2018), which performs belief inference relying on a particle filter and learns the environment models simultaneously. 2) *Deep Recurrent Q-network (DRQN)* (Hausknecht and Stone 2015), which employ a recurrent architecture of the neural network. 3) *Action-specific DRQN (ADRQN)* (Zhu, Li, and Poupart 2017), which differs from DRQN by extra inputting the action to the RNNs. 4) *Fill Adjacent (FA)*, a baseline method which fill the missing components with adjacent earlier values of observation. We implement all the methods by extending PPO; thus the latter three methods are called *DR-PPO, ADR-PPO and FA-PPO* respectively. Especially, for both *DR-PPO* and *ADR-PPO*, we add an additional auxiliary learning task with log likelihood loss of the next observation. The incomplete observation x_t (with zero-padding for the missing elements) and the missing indicator vector m_t are input to the network. Each algorithm was run with 3 random seeds on each task. We report the episode rewards of the trained polices and the precision of the inferred state.

Performance under a POMDP case

We evaluate the algorithms under a POMDP case where the observations are incomplete and noisy. We test the performance for both the MAR mode and NMAR mode (see Appendix B.1 for the results of the NMAR mode). Recall that our methods can theoretically work on MCAR and MAR mode, while MAR is a relatively harder case as the missingness also depend on the observed data. For the MAR mode,

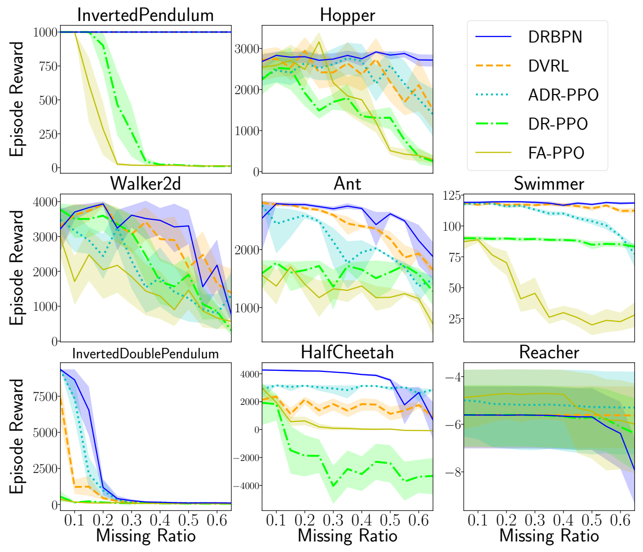


Figure 2: Episode rewards of the trained policies under dynamical missingness across different missing ratios. The elements of the observations are dynamically missing over timesteps. The horizontal axis shows different missing ratios. The shaded area depicts the mean and 80% confidence intervals averaged over 30 episodes.

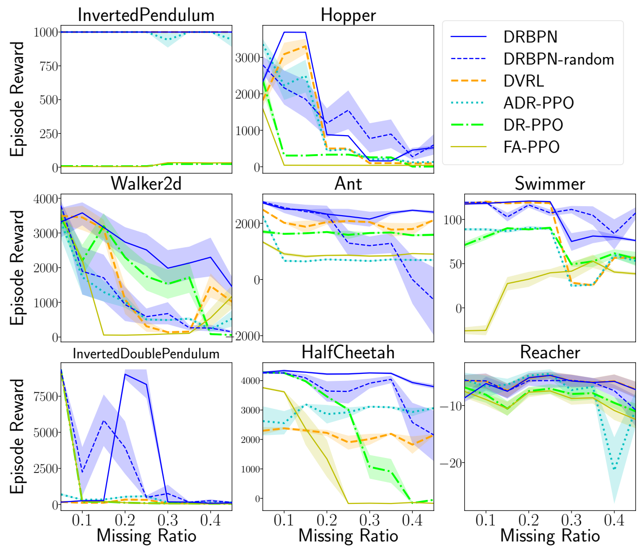


Figure 3: Episode rewards of the trained policies for sensor reduction across different reduction ratios. The missing elements of the observations are fixed over timesteps. The horizontal axis shows different reduction ratios. The shaded area depicts the mean and 80% confidence interval averaged over 30 episodes.

the probability distribution of i-th missing indicator is designed as

$$P_\eta(m_t^{(i)}|a_{t-1}, x_t^o) \triangleq \text{Bernoulli}(m_t^{(i)}|f(a_{t-1}, \hat{x}_t^o, \eta))$$

where $f(a_{t-1}, \hat{x}_t^o, \eta) = 1 - min\left(g(\beta_{1,i}^\top a_{t-1} + \beta_{2,i}^\top \hat{x}_t^o + \beta_{0,i}), \eta\right)$; g is the sigmoid function; $\beta_{0,i}$, $\beta_{1,i}$, $\beta_{2,i}$ are constants whose entries are randomly sampled from $\mathcal{N}(0,1)$; $\hat{x}_t^o$ are the part of the observation which are always to be observed; η is a parameter to control the missing ratio. Recall that $m_t^{(i)} = 0$ means the i-th component is missing. The noise ϵ is sampled from $\mathcal{N}(0, \Sigma^\epsilon)$, where $\Sigma^\epsilon = (\sigma \times 0.01 \times I)^2$. The algorithms are run for 1×10^6 timesteps. Then the trained models are tested under cases of dynamical missingness and sensor reduction with different ratios respectively.

Generalization Performance under Dynamical Missingness: We evaluate the trained models under dynamical missingness cases with different missing ratios, which means that the missing components is dynamically changing over time. Fig. 2 plots the episode rewards of the policies, averaged over 30 episodes (10 episodes for every 3 random seeds). In general, DRBPN (blue line) outperforms the compared methods on almost all tasks across different missing ratio expect Reacher. Compared to DVRL, DRBPN achieves about 20%, 75%, 100% higher performance than DVRL on Hopper, Walker2d, and HalfCheetah, under the missing ratio of 0.5. DRBPN also outperforms DR-PPO and ADR-PPO by a large margin especially on high-dimensional tasks like Hopper, Walker2d, and Ant. FA-PPO does not work well as missing ratio increases on almost all tasks. It seems that the adjacent earlier values of the observations are not sufficient for outputting correct actions.

The qualitative differences between domains in Fig. 2 are mainly due to the different difficulties across the tasks. For example, all the methods fail on InvertedDoublePendulum (left bottom of Fig. 2) once the missing ratio increases to 0.2 (while these methods do not perform such badly on other tasks even the missing ratio increases to 0.4). The InvertedDoublePendulum task need to keep two jointed pendulums to be upright, making it easily fail unless it is controlled in the right way without any break. Thus any careless actions could result in catastrophic failure on this task. Nevertheless, DRBPN still performs more robust than the other methods. In summary, DRBPN shows robustness against significant missingness.

Generalization Performance for Sensor Reduction: By "sensor reduction", it means that part of the sensors are removed and the corresponding components of the observations are always missing over timesteps. We choose the components which could optimally reduce the uncertainty of the belief according to the learned observation model of DRBPN (see Appendix A.1 for more detail). Fig. 3 plots the episode rewards for sensor reduction across different reduction ratio. In general, DRBPN (blue solid line) outperforms the compared methods on all the tasks. It performs well at a reduction ratio of 0.4 on InvertedPendulum, Ant, Swimmer, and HalfCheetah. We also test a version by randomly removing sensors, denoted as *DRBPN-random* (blue dashed line). As can be seen, DRBPN-random fails on most of the tasks like Walker2d, Ant, InvertedDoublePendulum, and HalfCheetah. These results show the effectiveness of the sensor reduction technique by the learned model of DRBPN.

	(a) Top 10 averaged Rewards				(b) Timesteps ($\times 10^3$) to hit threshold				
	DRBPN	PPO(ours)	PPO	ACKTR	Threshold	DRBPN	PPO(ours)	PPO	ACKTR
HalfCheetah	**4158**	1643	1549	3233	2100	**228**	410	/	850
Hopper	**3845**	3551	3622	3309	2000	**183**	221	209	502
Ant	2994	1782	2448	**3566**	2500	1110	/	/	**825**
InvertedDoublePendulum	**9344**	9339	9333	9342	9100	**114**	157	155	348
InvertedPendulum	**1000**	1000	1000	1000	950	**17**	25	18	110
Swimmer	116	106	**118**	47	90	**490**	877	550	/
Reacher	-5	-3	**-1**	-3	-7	172	232	**137**	250
Walker2d	**4356**	4117	4285	2398	3000	468	431	**252**	/

Table 1: Results of top 10 averaged episode rewards and timesteps to hit a threshold within 2 million timesteps, averaged over 3 random seeds.

Learned Observation Model: Fig. 4 depicts the learned covariance matrices of the observation components. The plotting matrices are obtained by $\theta_s \tilde{\Sigma} \theta_s^\top + \Sigma^\epsilon$, which could reflect the correlation between each components of the observation. For example, in InvertedPendulum, the 3rd and the 4th component means the horizontal and vertical velocity respectively, and the covariance between them are relatively larger than others (see the left top of Fig. 4).

As can be seen, DRBPN requires much less training time than DVRL; this is mainly due to that the inference of DRBPN is made analytically while not requiring additional training to approximate the inference procedure. Whereas compared to DRPPO and ADRPPO, DRBPN incorporates an additional imputation and filtering operation to perform robust inference, thus requires more computation.

Evaluation under a Fully Observable Case

In this section, we investigate whether the auxiliary learning tasks of DRBPN could aid policy learning under a fully observable case, where the observations are complete and clean. We compared DRBPN against the original PPO (Schulman et al. 2017) and ACKTR (Wu et al. 2017), which are popular model-free RL methods. In addition, we test a version of PPO (denoted as *PPO(ours)*) which has exactly the same implementations and hyperparameters as DRBPN except removing the auxiliary learning task. We implement these algorithms based on the OpenAI baselines (Dhariwal et al. 2017). Each algorithm was run for 2 million timesteps.

Episode Rewards: Table 1 (a) shows the top 10 averaged episode rewards within 2 million timesteps. As can be seen, DRBPN outperforms the original PPO and PPO(ours) on almost all tasks except Reacher. Especially on HalfCheetah and Ant, DRBPN is 268% and 120% higher than the original PPO. DRBPN also performs better than ACKTR by a large margin on most of the tasks.

Sample Efficiency: Table 1 (b) shows the timesteps required by algorithms to hit a prescribed threshold within 2 million timesteps. As can be seen, DRBPN is more sample efficient than PPO and ACKTR on most of the tasks.

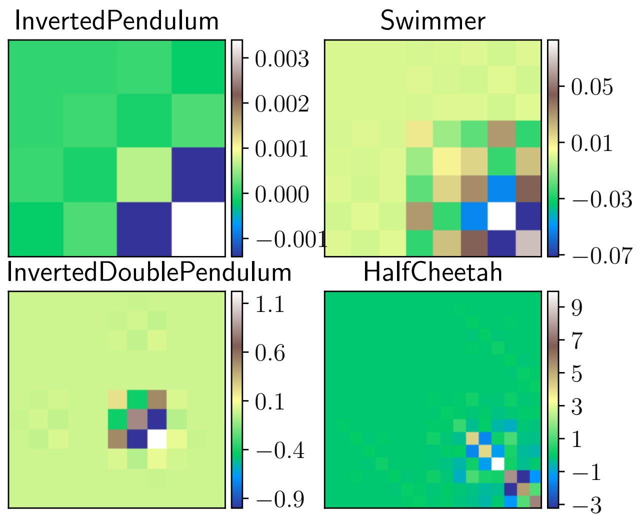


Figure 4: The learned covariances matrix between different dimensions of the observation. The x and y axes are the observation dimensions.

Conclusions

In this paper, we present a new algorithm for POMDPs with incomplete and noisy observations, named DRBPN. The DRBPN combines the strength of both feature memorization and belief inference approach by explicitly embedding the belief updating procedure into the architecture of RNN. We show that this hybrid type of recurrent structure incorporating an analytical belief inference can benefit on both computational efficiency and inference accuracy. Besides, the auxiliary learning task of the generative models can aid policy learning in performance and sample efficiency.

Recent works for POMDPs still have limited ability generalizing to complex realistic domains. In this paper, we investigate a special case where the partial observation data is also dynamically missing. However, more realistic circumstances need to be tackled down, e.g., some components of observations suddenly produce different values. An approach handling this case based on this work may be to treat such values as outliers and weigh down their impact on the belief updating.

Acknowledgements

This work is partially supported by National Science Foundation of China (61732006,61976115), AI+ Project of NUAA(XZA20005,56XZA18009), and research project(315025305), Postgraduate Research & Practice Innovation Program of Jiangsu Province (KYCX19_0195). We would also like to thank the anonymous reviewers, for offering thoughtful comments and helpful advice on earlier versions of this work.

References

Azizzadenesheli, K.; Lazaric, A.; and Anandkumar, A. 2016. Reinforcement Learning of POMDPs using Spectral Methods. *Conference on Learning Theory* 49: 193–256.

Becker, P.; Pandya, H.; Gebhardt, G.; Zhao, C.; Taylor, J.; and Neumann, G. 2019. Recurrent kalman networks: Factorized inference in high-dimensional deep feature spaces. *arXiv preprint arXiv:1905.07357* .

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym.

Deisenroth, M. P.; and Rasmussen, C. E. 2011. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *Proceedings of the 28th International Conference on Machine Learning*, 465–472.

Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; and Wu, Y. 2017. OpenAI Baselines. https://github.com/openai/baselines.

Doshi-Velez, F.; Pfau, D.; Wood, F. D.; and Roy, N. 2015. Bayesian Nonparametric Methods for Partially-Observable Reinforcement Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37(2): 394–407.

Gregor, K.; Jimenez Rezende, D.; Besse, F.; Wu, Y.; Merzic, H.; and van den Oord, A. 2019. Shaping belief states with generative environment models for rl. *Advances in Neural Information Processing Systems* 32: 13475–13487.

Guo, Z. D.; Azar, M. G.; Piot, B.; Pires, B. A.; and Munos, R. 2018. Neural predictive belief representations. *arXiv preprint arXiv:1811.06407* .

Hausknecht, M. J.; and Stone, P. 2015. Deep Recurrent Q-Learning for Partially Observable MDPs. *2015 AAAI Fall Symposium Series* 29–37.

Igl, M.; Zintgraf, L. M.; Le, T. A.; Wood, F.; and Whiteson, S. 2018. Deep Variational Reinforcement Learning for POMDPs. *International Conference on Machine Learning* 2122–2131.

Karkus, P.; Hsu, D.; and Lee, W. S. 2017. Qmdp-net: Deep learning for planning under partial observability. In *Advances in Neural Information Processing Systems*, 4694–4704.

Karl, M.; Soelch, M.; Bayer, J.; and Van der Smagt, P. 2016. Deep variational Bayes Filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432* .

Katt, S.; Oliehoek, F. A.; and Amato, C. 2017. Learning in POMDPs with Monte Carlo Tree Search. *International Conference on Machine Learning* 1819–1827.

Lim, B.; Zohren, S.; and Roberts, S. 2019. Recurrent Neural Filters: Learning Independent Bayesian Filtering Steps for Time Series Prediction. *arXiv preprint arXiv:1901.08096* .

Little, R. J.; and Rubin, D. B. 2019. *Statistical analysis with missing data*, volume 793. John Wiley & Sons.

Lizotte, D. J.; Gunter, L.; Laber, E. B.; and Murphy, S. A. 2008. Missing Data and Uncertainty in Batch Reinforcement Learning. *NIPS-08 Workshop on Model Uncertainty and Risk in Reinforcement Learning* 1–8.

McAllester, D. A.; and Singh, S. P. 1999. Approximate planning for factored POMDPs using belief state simplification. *Uncertainty in Artificial Intelligence* 409–416.

McAllister, R.; and Rasmussen, C. E. 2017. Data-Efficient Reinforcement Learning in Continuous State-Action Gaussian-POMDPs. In *NIPS 2017*, 2040–2049.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529.

Randlv, J.; and Alstrm, P. 1997. Reinforcement Learning based on Incomplete State Data. URL https://pdfs.semanticscholar.org/ea66/983af0df5e9b26ab68c2ed2b18ead79cb19d.pdf. Accessed on 2020-09-11.

Roijers, D. M.; Whiteson, S.; and Oliehoek, F. A. 2015. Point-based planning for multi-objective POMDPs. In *IJCAI'15 Proceedings of the 24th International Conference on Artificial Intelligence*, 1666–1672.

Ross, S.; Pineau, J.; Chaib-draa, B.; and Kreitmann, P. 2011. A Bayesian Approach for Learning and Planning in Partially Observable Markov Decision Processes. *Journal of Machine Learning Research* 12: 1729–1770.

Ross, S.; Pineau, J.; Paquet, S.; and Chaib-draa, B. 2008. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research* 32(1): 663–704.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M. I.; and Abbeel, P. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *International Conference on Learning Representations* .

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* .

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2017. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv preprint arXiv:1712.01815* .

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. In *Ieee/rsj International Conference on Intelligent Robots and Systems*, 5026–5033.

Willi, T.; Masci, J.; Schmidhuber, J.; and Osendorfer, C. 2019. Recurrent Neural Processes. *arXiv preprint arXiv:1906.05915* .

Wu, Y.; Mansimov, E.; Grosse, R. B.; Liao, S.; and Ba, J. 2017. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. *Neural Information Processing Systems* 5279–5288.

Zhang, M.; Levine, S.; McCarthy, Z.; Finn, C.; and Abbeel, P. 2015. Policy Learning with Continuous Memory States for Partially Observed Robotic Control. *arXiv: Learning* .

Zhu, P.; Li, X.; and Poupart, P. 2017. On Improving Deep Reinforcement Learning for POMDPs. *arXiv preprint arXiv:1804.06309* .