

Structure Learning-Based Task Decomposition For Reinforcement Learning In Non-Stationary Environments

Honguk Woo^{*†}, Gwangpyo Yoo*, Minjong Yoo*

Department of Computer Science and Engineering, Sungkyunkwan University
hwoo, necrocathy, mgyoo2@skku.edu

Abstract

Reinforcement learning (RL) agents empowered by deep neural networks have been considered a feasible solution to automate control functions in a cyber-physical system. In this work, we consider an RL-based agent and address the issue of learning via continual interaction with a time-varying dynamic system modeled as a non-stationary Markov decision process (MDP). We view such a non-stationary MDP as a time series of conventional MDPs that can be parameterized by hidden variables. To infer the hidden parameters, we present a task decomposition method that exploits CycleGAN-based structure learning. This method enables the separation of time-variant tasks from a non-stationary MDP, establishing the task decomposition embedding specific to time-varying information. To mitigate the adverse effect due to inherent noises of task embedding, we also leverage continual learning on sequential tasks by adapting the orthogonal gradient descent scheme with a sliding window. Through various experiments, we demonstrate that our approach renders the RL agent adaptable to time-varying dynamic environment conditions, outperforming other methods including state-of-the-art non-stationary MDP algorithms.

Introduction

Reinforcement learning (RL) shows its applicability in autonomous control systems where learning via continual interaction can be formulated as solving a Markov decision process (MDP). Although an MDP provides a strong mathematical model for RL, a learned agent on an MDP often has limitations in ensuring optimal performance when the agent is deployed in a real-world environment with time-varying dynamic conditions. For instance, time-varying road conditions are not fully observable for an autonomous vehicle; the tire-road friction might vary continuously due to its environment-dependent nature. This situation causes challenging problems modeled as a non-stationary MDP. Several studies on non-stationary MDPs have been introduced, e.g., model-based environments (Jaksch, Ortner, and Auer 2010, Gajane, Ortner, and Auer 2019) and context detection methods (Da Silva et al. 2006, Padakandla, Prabuchandran, and

Bhatnagar 2020). Recently, the task embedding technique using recurrent neural networks has been explored to infer changes in a non-stationary environment (Lee et al. 2019). That technique is notably effective, but often leads to suboptimal convergence when its target environment is complex (Igl et al. 2018).

In this paper, to address the learning difficulty issue in non-stationary, complex environments, we take an integrated approach of structural learning and continual learning. Specifically, we employ the task decomposition embedding (TDE) that enables the high-quality inference on hidden parameters by decomposing time-variant and time-invariant tasks from a non-stationary environment. In TDE, CycleGAN (Zhu et al. 2017) is used to represent common time-invariant tasks of multiple MDPs, so it makes inferences on time-variant tasks through the complements of the common time-invariant tasks. Furthermore, to mitigate the adverse effect caused by the inherent estimation errors on hidden parameters, we adapt the orthogonal gradient descent (OGD) (Farajtabar et al. 2020) with a sliding window. In continual learning, this preserves prior learned knowledge effectively by transforming the current gradient in a way of minimizing the interference.

In simulation experiments, our model shows a higher performance of up to 46.4% compared to a state-of-the-art non-stationary MDP method. Furthermore, the model achieves robust performance in complex non-stationary environments similar to real-world situations, e.g., showing 67.4% improvement over compared methods in the Airsim-based drone flying simulation where highly dynamic weather conditions are configured.

The main contributions of this paper are as follows.

- We present a novel RL solution to non-stationary MDPs, in which CycleGAN-based structural learning and OGD-based continual learning are used together.
- We devise the task embedding scheme specific to the decomposition of time-variant and time-invariant task information.
- We demonstrate the performance benefits of our approach in various non-stationary environment conditions, and show a case study with Airsim-based drone flying scenarios.

* These authors contributed equally.

† Honguk Woo is the corresponding author.

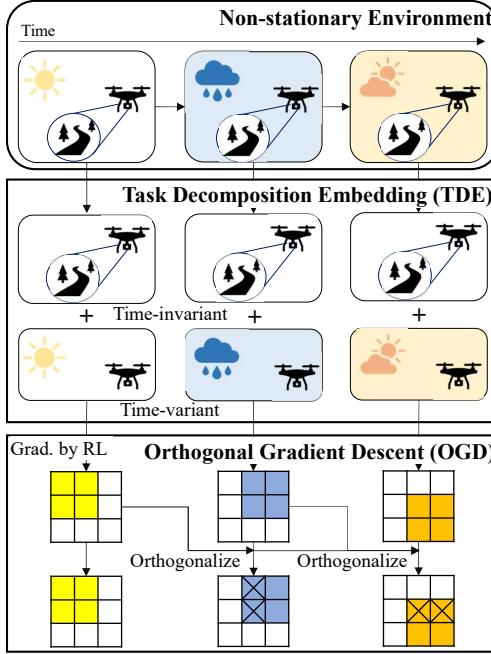


Figure 1: Our proposed model is learned through (1) TDE to decompose tasks into *time-variant* and *time-invariant* and (2) OGD-based RL to learn a policy robust to non-stationarity.

Overall Approach

In this section, we describe the problem of making optimal decisions upon a non-stationary MDP that is used to model real-world dynamic environments, and briefly present our approach to the problem.

Non Stationary Environment Problem

In conventional RL formulation, a learning environment is defined as an MDP of

$$(\mathcal{S}, \mathcal{A}, \mathcal{P}, R) \quad (1)$$

where \mathcal{S} is a state space, \mathcal{A} is an action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a transition probability, and $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a reward function. In the case of dealing with a real-world system, it is often required to formulate its non-stationary environment in a sequence of MDPs $\{\mathcal{T}^t\}_{t \in \mathbb{N}}$ sampled from some distribution $\mathcal{P}(\mathcal{T})$ (Bellman 1956, Chandak et al. 2020). This formulation is considered a special type of POMDPs (Choi, Yeung, and Zhang 2001).

In general, a recurrent policy, e.g., LSTM (Hochreiter and Schmidhuber 1997), is effective for POMDP problems where an action can be determined on the history of states and actions, but its performance often becomes restrictive in large-scale complex environments. It is because the recurrent policy remembers the history through deterministic feature computation, and does not explicitly combine the knowledge of the learned environment models and the history (Wang and Tan 2021).

In the following, we explain the issue of learning upon such a POMDP derived from a non-stationary MDP. For a non-stationary MDP that is formalized as a sequence of MDPs,

$$\mathcal{T}^t = (\mathcal{S}, \mathcal{A}, \mathcal{P}^t, R^t) \text{ where } \mathcal{T}^t \sim p(\mathcal{T}^{t-1}), \quad (2)$$

we introduce a hidden latent variable \mathbf{v}_t which parameterizes \mathcal{T}^t . That is, we have $R(s_t, \mathbf{v}_t, a_t) \mapsto R^t(s_t, a_t)$ and $\mathcal{P}(s_{t+1}, \mathbf{v}_{t+1}, a_t, s_t, \mathbf{v}_t) \mapsto \mathcal{P}^t(s_{t+1}, a_t, s_t)$ as shown in (Doshi-Velez and Konidaris 2016). The actual state space is extended to $\mathcal{S} \times \mathcal{V}$ where \mathcal{V} is the collection of latent variables \mathbf{v}_t . Then, the respective POMDP of $\{\mathcal{T}^t\}_{t \in \mathbb{N}}$ is modeled as

$$(\mathcal{S} \times \mathcal{V}, \mathcal{A}, \Omega = \mathcal{S}, \mathcal{P}, R, \mathcal{O}) \quad (3)$$

where the observation space $\Omega = \mathcal{S}$ and observation function $\mathcal{O}((s_t, \mathbf{v}_t)) \mapsto s_t$ are given.

To simplify the MDP parameterized by \mathbf{v}_t , we represent

$$\mathcal{T}^t = (\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \mathbf{v}_t), \quad \mathbf{v}_t \sim p(\mathbf{v}_{t-1}) \quad (4)$$

where $p(\mathbf{v}_{t-1})$ is some conditional distribution given \mathbf{v}_{t-1} (Doshi-Velez and Konidaris 2016).

Overall Approach for Non-Stationary MDPs

To address the difficulty of learning in a non-stationary MDP, we take an integrated approach using structural learning and continual learning techniques. Figure 1 illustrates the concept of our approach; The agent integrates knowledge by continuously learning from multiple tasks and adjusting the gradient (knowledge update) in a way that interference is minimized. In the figure, the basis vectors of gradients are represented by colored cells and they are adjusted not to conflict each other; X-marked cells correspond to the cancelled basis vectors via orthogonalization. Meanwhile, the agent acquires time-varying information through task decomposition to conduct task-specific actions upon uncertain environment conditions.

The structural learning is used to decompose a non-stationary MDP into two distinct groups of tasks such as time-invariant tasks and time-variant tasks, thereby alleviating task uncertainty induced by a time-variant distribution that the aforementioned hidden parameter \mathbf{v} follows. Furthermore, to mitigate the learning performance degradation of an RL agent due to the estimation error on \mathbf{v} , we employ the OGD variant with a sliding window.

Specifically, we first figure out both stationary features and time-variant features of a non-stationary environment. Then, to extract a time-variant MDP using stationary features from the environment, we employ **task decomposition embedding** (TDE), a surrogate mechanism that calculates decomposition between time-variant and time-invariant tasks. TDE is intended to learn the soft-homomorphism (Sorg and Singh 2009) of given sampled transitions of fixed tasks by random policies. It enables the mapping of a non-stationary MDP \mathcal{T}^t to a time-invariant stationary MDP \mathcal{T}_u , which identifies all time-variant parts. As a result, with TDE, it is possible to obtain latent variables \mathbf{v} . In terms of computational complexity, TDE is more efficient than direct decomposition of the whole model transition probability \mathcal{P}^t .

and reward function R^t . Even if \mathcal{P}^t and \mathcal{R}^t are known, it can be computationally intractable to directly decompose them for most cases unless \mathcal{S} or \mathcal{A} is small enough (Ravindran and Barto 2004).

Furthermore, to mitigate the performance degradation caused by inherent estimation errors on latent variables \mathbf{v} , we explore the sliding windowed OGD scheme in sequential multi-task learning. Such estimation errors lead to incorrect gradients in sequential multi-task learning, which can adversely affect learning, i.e., previous gradients are unintentionally cancelled by current gradients. In our approach, OGD is adapted to preserve the previous gradient directionality. Specifically, we exploit the sliding window mechanism for OGD, as exact time points at which tasks change are rarely detected.

Structure Learning for Task Decomposition

In this chapter, to extract time-variant features from a non-stationary MDP, we examine its simplified expression which identifies all time-variant part. Note that the complements of the reconstructed samples with only time-variant part in the authentic samples of a non-stationary environment represent the desired time-variant parameter of the non-stationary MDP.

To train a model for extracting time-variant latent variables \mathbf{v} , we need to map the transitions to the time-invariant part \mathbf{u} . In doing so, we sample a set of tasks in which \mathbf{v} is fixed. Then, using a random policy, the transitions are collected from the fixed tasks. Using CycleGAN, we create paired transitions: $((s, \mathbf{v}), a, R(s, \mathbf{v}, a), \mathcal{P}(s, \mathbf{v}, a))$ and $((s, \mathbf{v}'), a, R(s, \mathbf{v}', a), \mathcal{P}(s, \mathbf{v}', a))$, i.e., same transitions but time-variant latent variables \mathbf{v} and \mathbf{v}' are different. By training an encoder on paired transitions, we obtain such a mapping of transitions to time-invariant latent variables \mathbf{u} . By complementing the reconstruction from \mathbf{u} , we obtain \mathbf{v} . This TDE procedure is described in Algorithm 1 and the structure of auto-encoder and CycleGAN in TDE is illustrated in Figure 2.

To examine stationary features and time-variant features of a non-stationary environment, we introduce soft-homomorphism between MDPs. Consider a time-invariant MDP \mathcal{T}_u such as

$$\mathcal{T}_u = (\mathcal{U}, \mathcal{A}, \hat{R}, \mathcal{P}_u), \quad (5)$$

and a map such as

$$d_* : \mathcal{S} \times \mathcal{V} \times \mathcal{U} \rightarrow [0, 1] \quad (6)$$

which together satisfy the followings.

- I. $d_*(s, \mathbf{v}, \mathbf{u}) = \Pr[s, \mathbf{v} | \mathbf{u}]$
- II. $\sum_{s, \mathbf{v} \in \mathcal{S} \times \mathcal{V}} R(s, \mathbf{v}, a) \Pr[s, \mathbf{v} | \mathbf{u}] = \hat{R}(\mathbf{u}, a)$
- III. $\sum_{s, \mathbf{v} \in \mathcal{S} \times \mathcal{V}} \mathcal{P}(s, \mathbf{v}, a, s') \Pr(\mathbf{v}' | \mathbf{v}) \Pr[s, \mathbf{v} | \mathbf{u}] = \sum_{\mathbf{u}' \in \mathcal{U}} \mathcal{P}_u(\mathbf{u}, a, \mathbf{u}') \Pr[s', \mathbf{v}' | \mathbf{u}']$

The map d_* is an MDP soft-homomorphism between \mathcal{T}^t and \mathcal{T}_u (Sorg and Singh 2009). \mathcal{T}_u is simplified representation about \mathcal{T}^t , meaning that it is also an MDP that identifies all time-variant parts.

Here, we explain the procedure for adapting CycleGAN (Zhu et al. 2017) and auto-encoder to obtain soft-homomorphism d_* . The auto-encoder $dec \circ enc$ is trained and then its decoder dec is used to learn soft-homomorphism d_* . Since d_* is a probability of s and \mathbf{v} given \mathbf{u} , dec should be able to reconstruct transitions without \mathbf{v} . Accordingly, the reconstruction loss below is used to train $dec \circ enc$.

$$\mathcal{L}_{recon}(enc, dec) = \|\tau - dec(\mathbf{u}, \mathbf{0})\|_2 \quad (7)$$

The encoder enc should map paired transitions to the same point on time-variant latent space \mathcal{U} . That is, $\mathbb{E}[R(dec(\mathbf{u}, \mathbf{v}), a) | \mathbf{u}] = \mathbb{E}[R(s, \mathbf{v}, a) | \mathbf{u}]$ and $\mathbb{E}[\mathcal{P}(dec(\mathbf{u}, \mathbf{v}), a) | \mathbf{u}] = \mathbb{E}[\mathcal{P}(s, \mathbf{v}, a) | \mathbf{u}]$. Then, the loss in (7) enforces the decoder dec to keep soft-homomorphism properties I and II. For soft-homomorphism properties I, II, and III, the similarity loss below is additionally used.

$$\mathcal{L}_{sim}(enc) = \|\mathbf{u} - \mathbf{u}'\|_2 = \|enc(\tau) | \mathcal{U} - enc \circ G(\tau) | \mathcal{U}\|_2 \quad (8)$$

Note that $\cdot | \mathcal{U}$ is a projection to the \mathcal{U} , i.e., $(\mathbf{u}, \mathbf{v}) | \mathcal{U} \mapsto \mathbf{u}$. These transitions can be generated by the generator G of CycleGAN which will be explained below. According to the basic reconstruction loss of the auto-encoder such as

$$\mathcal{L}_{ae}(enc, dec) = \|\tau - dec \circ enc(\tau)\|_2, \quad (9)$$

the decoder can learn $\Pr[s, \mathbf{v} | \mathbf{u}, \mathbf{v}] = \Pr[s | \mathbf{u}]$ for homomorphism properties.

To keep the probabilistic property of \mathbf{u} for the encoder and given sampled fixed tasks, we adopt CycleGAN, where the cycle consistency loss is given by

$$\begin{aligned} \mathcal{L}_{con}(G^{(1)}, G^{(2)}) &= \mathbb{E}_{x \sim \mathcal{D}_{\mathcal{T}_{(2)}}} [\|G^{(2)} \circ G^{(1)}(x) - x\|_1] \\ &\quad + \mathbb{E}_{y \sim \mathcal{D}_{\mathcal{T}_{(1)}}} [\|G^{(1)} \circ G^{(2)}(y) - y\|_1]. \end{aligned} \quad (10)$$

Note that $\mathcal{D}_{\mathcal{T}_{(1)}}$ and $\mathcal{D}_{\mathcal{T}_{(2)}}$ denote sampled transitions in fixed tasks $\mathcal{T}_{(1)}$ and $\mathcal{T}_{(2)}$, respectively. This loss drives a model to find out common time-invariant latent variables \mathbf{u} and discriminate distinct time-variant latent variables \mathbf{v} . That is, it identifies all transitions with a same common feature $\{(\mathbf{u}, \cdot)\}$ (Gavranović 2020). The loss also drives G to learn rewards for the same transitions with different time-variant parts, and hence the encoder can learn proper representation of \mathbf{u} , thereby allowing the decoder to learn soft-homomorphic properties at the same time.

The adversarial loss is given by

$$\begin{aligned} \mathcal{L}_{gan}(G, D) &= \mathbb{E}_{y \sim \mathcal{D}_{\mathcal{T}_{(1)}}} [\log(D(y))] \\ &\quad + \mathbb{E}_{x \sim \mathcal{D}_{\mathcal{T}_{(2)}}} [\log(1 - D(G(x)))] \end{aligned} \quad (11)$$

With the adversarial loss, discriminator D finds out the distribution of transitions by maximizing the loss. Furthermore, the adversarial loss regularizes generator G to map its domain \mathcal{T}_1 to plausible transitions in task \mathcal{T}_2 . (e.g., it gives penalties to out of distribution transitions), and enforces a model to keep soft-homomorphism properties II and III (Zhu et al. 2017).

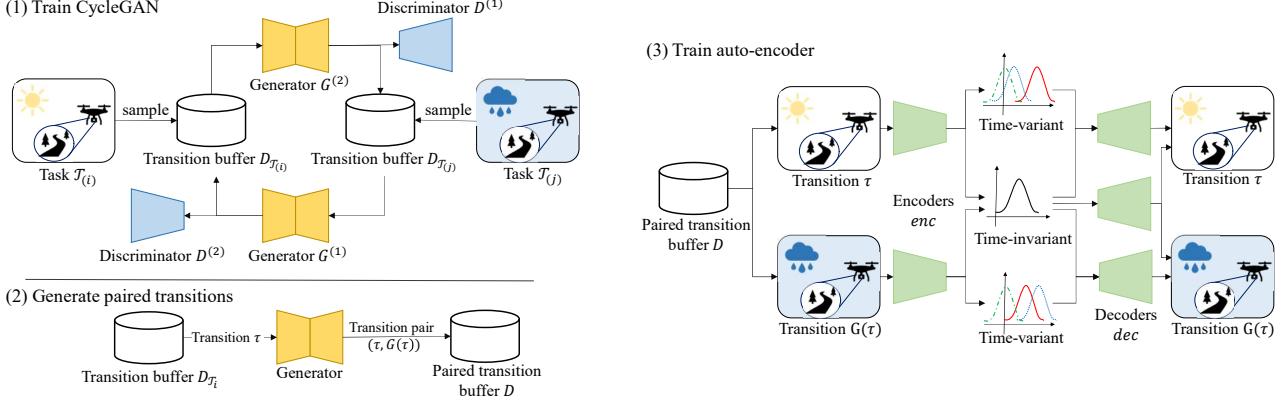


Figure 2: The overall structure of TDE

Algorithm 1: Task decomposition embedding

```

Task Set  $Set(\mathcal{T})$ , paired transition buffer  $\mathcal{D}$ 
transitions buffer  $\mathcal{D}_{\mathcal{T}(i)}$ , the number of sampled environment N
for i from 0 to N do
    Get task  $\mathcal{T}(i)$  by random sampling in  $Set(\mathcal{T})$ 
    Sample transitions  $\tau = (s, a, r, s')$  from  $\mathcal{T}(i)$  by random policy
     $\mathcal{D}_{\mathcal{T}(i)} = \mathcal{D}_{\mathcal{T}(i)} \cup \{(s, a, r, s')\}$ 
end for
loop
    // (1) Train CycleGAN
    Choose i, j in 0, 1, ..., N
    Initialize Generators  $G^{(1)}, G^{(2)}$ , Discriminators  $D^{(1)}, D^{(2)}$ 
    loop
        loss =  $\mathcal{L}_{gan}(G^{(1)}, D^{(2)}, \mathcal{D}_{\mathcal{T}(i)}, \mathcal{D}_{\mathcal{T}(j)})$ 
        +  $\mathcal{L}_{gan}(G^{(2)}, D^{(1)}, \mathcal{D}_{\mathcal{T}(j)}, \mathcal{D}_{\mathcal{T}(i)}) + \mathcal{L}_{con}(G^{(1)}, G^{(2)})$ 
        optimize( $(G^{(1)}, G^{(2)}, D^{(1)}, D^{(2)})$ , loss)
    end loop
    // (2) Generate paired transitions
    for  $\tau = (s, a, r, s')$  in  $\mathcal{D}_{\mathcal{T}(i)}$ ,  $\tau'$  in  $\mathcal{D}_{\mathcal{T}(j)}$  do
         $\mathcal{D} = \mathcal{D} \cup \{(\tau, G^{(1)}(\tau)) \cup \{(\tau', G^{(2)}(\tau')\}$ 
    end for
end loop
    Initialize Encoder  $enc$ , Decoder  $dec$ 
// (3) Train auto-encoder
for  $\tau, G(\tau)$  in  $\mathcal{D}$  do
    loss =  $\mathcal{L}_{ae}(enc, dec) + \mathcal{L}_{sim}(enc) + \mathcal{L}_{recon}(enc, dec)$ 
    optimize( $(enc, dec)$ , loss)
end for
return  $enc$ 

```

Continual Learning with Orthogonal Gradient Descent

Estimation errors in task embedding can cause incorrect gradients in RL. Specifically, if the gradients generated by such errors are in the same direction as the previous gradients, and the previous gradients are cancelled, the previously learned knowledge can be adversely affected. (Farajtabar et al. 2020,

Yu et al. 2020) To mitigate those erroneous situations, we adapt OGD (Farajtabar et al. 2020) so that gradient noise caused by task embedding does not affect the gradient previously generated. In general, OGD is known to converge on each task when exact time points of task changes are known. Unlike this assumption, non-stationary MDPs have limitations such as unknown time points of task changes. Thus, we adapt OGD with sliding window so that continual gradient update is enabled. To preserve the directionality of gradient $\nabla_{\theta}\mathcal{L}$ generated upon specific task learning, the gradient $\nabla_{\theta}\mathcal{L}'$ by learning a newly encountered task is orthogonalized based on the former gradient $\nabla_{\theta}\mathcal{L}$. Given the loss $\mathcal{L}_t(\theta)$ generated at time-mstep t , we have

$$\nabla_{\theta}\mathcal{L}_t(\theta) \perp \lambda_{t-i}(\theta) \quad \text{for } i = 1, 2, 3, \dots, N \quad (12)$$

where λ_{t-i} is a gradient applied at timestep $t - 1$.

Therefore, to be orthogonal to previous losses, $\nabla_{\theta}\mathcal{L}_t(\theta)$ is modified using the Gram-Schmidt method (Leon, de Pillis, and De Pillis 2015), i.e.,

$$\lambda(\theta) \leftarrow \nabla_{\theta}\mathcal{L}(\theta) - \sum_{\lambda' \in \Lambda} \text{proj}(\nabla_{\theta}\mathcal{L}(\theta), \lambda') \quad (13)$$

where $\text{proj}(\nabla_{\theta}\mathcal{L}(\theta), \lambda)$ is a projection $\nabla_{\theta}\mathcal{L}_t(\theta)$ onto λ and Λ is gradients buffer for previous gradients. That is, $\text{proj}(\mathbf{x}, \lambda) = \frac{\mathbf{x}^T \lambda}{\lambda^T \lambda} \lambda$. As a result, the previous gradients are set to be orthogonal each other. Then, model parameters θ are updated with learning rate α ,

$$\theta \leftarrow \theta - \alpha \lambda(\theta). \quad (14)$$

This sliding windowed OGD procedure is described in Algorithm 2.

To verify the convergence by sliding windowed OGD, we show that the loss function $\mathcal{L}(\theta)$ is strictly decreasing when model parameters update. Suppose that $\nabla_{\theta}\mathcal{L}$ is an L-Lipschitz continuous function, and \mathcal{L} is convex and differentiable. Let θ be a model parameter and θ' be the model parameter after one update. By mean value theorem, given

$\mathcal{L}(\theta')$ around $\mathcal{L}(\theta)$, we have the following inequality,

$$\begin{aligned}\mathcal{L}(\theta') &\leq \mathcal{L}(\theta) + \nabla_{\theta} \mathcal{L}^T(\theta)(\theta' - \theta) + \frac{1}{2} \Delta \mathcal{L}(\theta) \|\theta' - \theta\|^2 \\ \text{where } \Delta \mathcal{L}(\theta) &= \sum_i \frac{\partial^2 \mathcal{L}(\theta)}{\partial \theta_i^2}.\end{aligned}\quad (15)$$

By L -Lipschitz condition, $\frac{1}{2} \Delta \mathcal{L}(\theta) \leq \frac{1}{2} L$ holds. As a result, we obtain that

$$\mathcal{L}(\theta') \leq \mathcal{L}(\theta) + \nabla_{\theta} \mathcal{L}^T(\theta)(\theta' - \theta) + \frac{1}{2} L \|\theta' - \theta\|^2. \quad (16)$$

Let $\Sigma(\theta) = \sum_{\lambda' \in \Lambda} \text{proj}(\nabla_{\theta} \mathcal{L}(\theta), \lambda')$ in (13). Note that $\theta' = \theta - \alpha \lambda(\theta)$ holds by definition of θ' . Hence, the following inequality holds.

$$\mathcal{L}(\theta') \leq \mathcal{L}(\theta) - \alpha \nabla_{\theta} \mathcal{L}(\theta)^T \lambda(\theta) + \frac{1}{2} L \|\alpha \lambda(\theta)\|^2. \quad (17)$$

Since $\nabla \mathcal{L}(\theta) = \lambda(\theta) + \Sigma(\theta)$, it holds that

$$\begin{aligned}\nabla_{\theta} \mathcal{L}(\theta)^T \lambda(\theta) &= (\lambda(\theta) + \Sigma(\theta))^T \lambda(\theta) \\ &= \lambda(\theta)^T \lambda(\theta).\end{aligned}\quad (18)$$

It is because $\Sigma(\theta)^T \lambda(\theta) = 0$ by orthogonalization. Therefore, $\nabla_{\theta} \mathcal{L}(\theta)^T \lambda(\theta) = \|\lambda(\theta)\|^2$. From (17), we have

$$\begin{aligned}\mathcal{L}(\theta') &\leq \mathcal{L}(\theta) - \alpha \|\lambda(\theta)\|^2 + \frac{1}{2} L \alpha^2 \|\lambda(\theta)\|^2 \\ &\leq \mathcal{L}(\theta) - \alpha \left(1 - \frac{1}{2} L \alpha\right) \|\lambda(\theta)\|^2.\end{aligned}\quad (19)$$

For $\alpha < 2/L$, the loss $\mathcal{L}(\theta)$ strictly decreases when gradient updates. When $\|\lambda(\theta)\| \neq 0$ and learning rate α is sufficiently small, the loss strictly decreases. Note $\|\lambda(\theta)\| = 0$ occurs only when a model converges to its optimal parameter or $\nabla_{\theta} \mathcal{L}(\theta) = \Sigma(\theta)$ holds. Let $\theta \in \mathbb{R}^{n_{\theta}}$ and n_{θ} be the number of parameters of θ . Then, there are n_{θ} orthogonal basis of $\nabla_{\theta} \mathcal{L}(\theta)$. Thus, $\nabla_{\theta} \mathcal{L}(\theta) = \Sigma(\theta)$ rarely occurs when the number of parameters n_{θ} is much larger than the gradient buffer size (Farajtabar et al. 2020).

Experiments

In this section, we describe the implementation of our proposed method and evaluate its performance under various simulation conditions.

Experimental settings Our model is implemented using Python v3.7, Pytorch v1.8, and Tensorflow v1.14, and is trained on a system of an Intel(R) Core(TM) i9-10940X processor and an NVIDIA RTX 3090 GPU. Detailed experimental settings including hyperparameter settings and environment conditions can be found in the Appendix.

Comparative methods For comparison, we implement and test several algorithms such as SLAC (Lee et al. 2019), and LILAC (Xie, Harrison, and Finn 2020) in addition to our proposed model.

- SLAC conducts variational inference to learn latent representation in POMDPs. By using the history of observations and actions, it infers latent representation of an

Algorithm 2: Sliding windowed OGD

```

replay buffer  $D$ , policy  $\pi_{\theta}$ , environment  $env$ , learning rate  $\alpha$ 
done  $\leftarrow False$ 
enc  $\leftarrow \text{TDE}(Set(\mathcal{T}))$  // Algorithm 1.
gradient buffer  $\Lambda = \text{Queue}(\text{max\_len} = N)$ ,  $D = []$ 
loop
     $\mathbf{v}_t \leftarrow \mathbf{0}$ ,  $s_t \leftarrow \text{reset}(env)$ 
    while not done do
         $a_t \leftarrow \pi_{\theta}(s_t, \mathbf{v}_t)$ 
         $s_{t+1}, r_t, done \leftarrow \text{step}(env, a_t)$ 
         $\mathbf{v}_{t+1}, \mathbf{u}_{t+1} \leftarrow enc(s_t, a_t, r_t, s_{t+1})$  // Ignore  $\mathbf{u}_{t+1}$ 
         $D \leftarrow D \cup \{\tau_t = ((s_t, \mathbf{v}_t), , a_t, r_t, (s_{t+1}, \mathbf{v}_{t+1}))\}$ 
        // RL-Update
        Fetch minibatch from  $D$  and calculate  $\nabla_{\theta} \mathcal{L}(\pi_{\theta}; \theta)$ 
        // Gram Schmidt Procedure
         $\lambda_t(\theta) \leftarrow \nabla_{\theta} \mathcal{L}(\pi_{\theta}; \theta) - \sum_{\lambda' \in \Lambda} \text{proj}(\nabla_{\theta} \mathcal{L}(\pi_{\theta}; \theta), \lambda')$ 
         $\theta \leftarrow \theta - \alpha \lambda_t(\theta)$ 
        enqueue( $\Lambda, \lambda_t(\theta)$ )
    end while
end loop
return  $\theta, enc$ 

```

MDP explicitly. SLAC is used as an indicator showing the environment complexity, considering that general POMDP solvers (e.g., SLAC) are difficult to learn a non-stationary MDP without specific properties.

- LILAC is a state-of-the-art method for solving non-stationary MDPs, which performs variational inference based transitions for task embedding. To predict the current task from previous task information, it is learned on task information predicted from the history of transitions. Though LILAC aims at handling episodic changes, it can achieve compatible results in non-episodic experiments.

Learning Environments We build a 2-dimensional navigation environment using pyBox2D (Catto 2012), where an agent avoids moving obstacles to reach its goal position. The agent uses lidar-like observations including obstacle position, obstacle speed, and remain timesteps, and gets rewards according to the distance change between consecutive timesteps. We also evaluate our approach with the minitaur environment (Tan et al. 2018). In the minitaur, an agent observes 8 joints information and conducts actions to control the joint angle. The agent receives rewards for moving forward at a specific speed. We set the weight of minitaur base to continuously vary, simulating non-stationary environment conditions.

Performance Figure 3 shows the performance in rewards achieved by the proposed model (TDE+OGD) and other methods (SLAC, LILAC) in the 2-dimensional navigation and minitaur environments. As observed, TDE+OGD achieves better performance in terms of both learning speed and achieved rewards than the others, e.g., showing 46.4% improvement on average over LILAC. We notice that SLAC shows unstable performance; SLAC is known to be effective for conventional POMDPs, but it rarely considers the properties of non-stationary MDPs. Although LILAC achieves

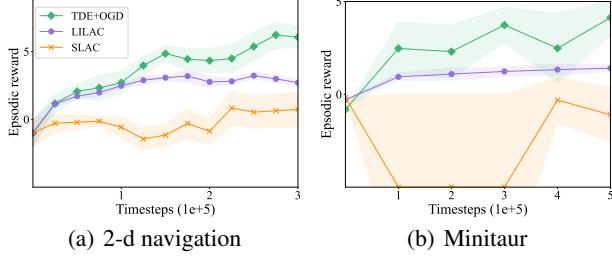


Figure 3: Performance in non-stationary environments (2-d navigation, minitaur)

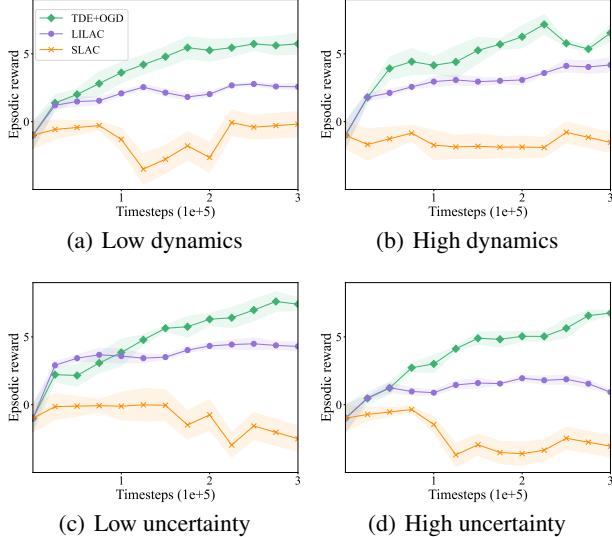


Figure 4: Various condition tests in 2-d navigation

relatively stable learning in both environments, it does not show competitive performance. It is because, in contrast to the given environment that changes rapidly even within a single episode, LILAC learns a policy under the assumption that tasks change episodically. Unlike LILAC, TDE+OGD exploits task embedding per timestep, so it can adapt itself to rapid environment changes.

Figure 4 shows the performance in achieved rewards by several methods with respect to task dynamics and uncertainty. Here, the task uncertainty is determined by auto-correlation of tasks, and the task dynamics is determined by the probability that a task changes at each timestep. For task dynamics, both TDE+OGD and LILAC achieve stable performance, showing no significant performance degradation between low dynamics environments and high dynamics environments. For task uncertainty, TDE+OGD shows a performance drop of 17.3% on average for high uncertainty compared to low uncertainty, whereas LILAC shows a performance drop of 62.1%.

Figure 5 shows the effects of TDE and OGD techniques in our model; TE+OGD denotes a model variant learned without CycleGAN through the auto-encoder, sample transitions,

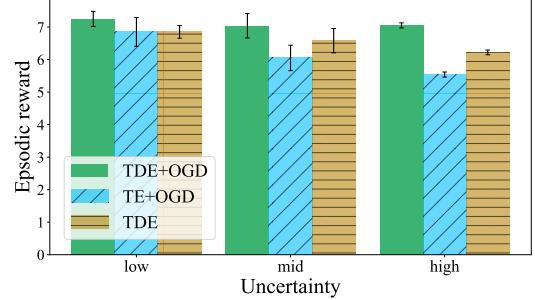


Figure 5: Effects of TDE and OGD

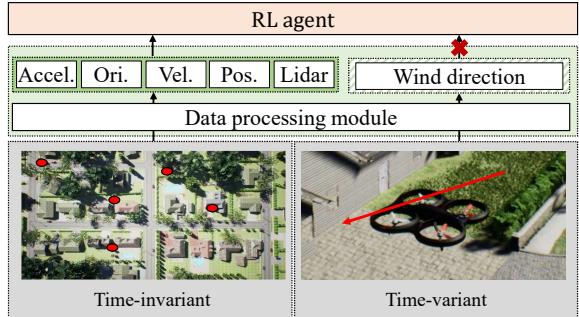


Figure 6: Airsim-based drone flying environments

and OGD. TDE denotes a model variant learned only with TDE. TDE+OGD is our proposed model. For low uncertainty, the performance gain of TDE+OGD over the others is about 6.2% (low), but for high uncertainty, it increases to about 14.4% (mid) and 18.6% (high) between TDE+OGD and TE+OGD, and about 8.3% (mid) and 10.1% (high) between TDE+OGD and TDE. This clarifies the benefits of TDE and OGD particularly for environments with high task uncertainty. In TDE, CycleGAN provides high-quality estimation on hidden parameters \mathbf{v} while OGD prevents incorrect gradients by task uncertainty.

Case Study To verify the applicability of our model in complex problem settings, we conduct a case study with autonomous quad-copter drones in the Airsim simulator (Shital Shah and Kapoor 2017). We configure wind velocity to

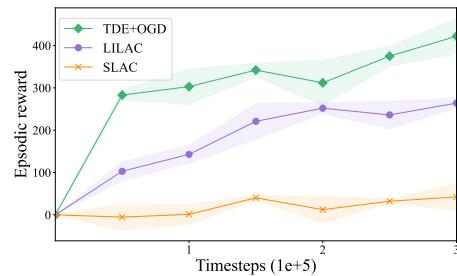


Figure 7: Performance in Airsim-based drone flying environments

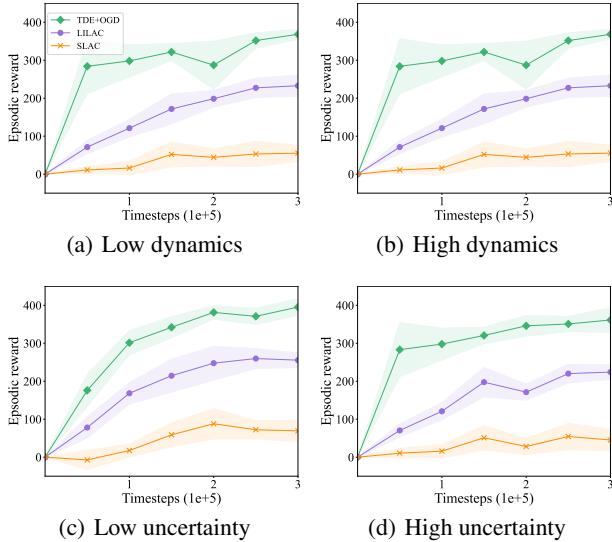


Figure 8: Various condition tests in Airsim-based drone flying environments

simulate non-stationary drone flying environments. In RL formulation, the drone agent observes its lidar data, position, speed, and angle of rotation, while it is set not to observe any data about wind that is used to intentionally influence environment dynamics. Figure 6 shows our implementation of Airsim-based environment. The drone agent continuously conducts control actions by manipulating the 3-dimensional acceleration for each timesteps, and receives rewards based on the goal distance.

In Figure 7, we observe that TDE+OGD performs better than the other methods, e.g., showing 67.4% improvement on average over LILAC, in terms of achieved average rewards in an episode. In Figure 8, we compare the performance with respect to task dynamics and uncertainty. For task dynamics, both TDE+OGD and LILAC show stability in performance, showing no significant degradation in low and high dynamics settings. For task uncertainty, TDE+OGD shows a performance drop of 8.6% on average for high uncertainty compared to low uncertainty, whereas LILAC shows a slightly more drop of 12.28%.

Related Work

The problem of non-stationary environments has been studied in several research works. Jaksch, Ortner, and Auer (2010) and Gajane, Ortner, and Auer (2019) presented the regret minimization algorithm that extends value iteration methods to deal with task uncertainty, proving its efficiency for simple non-stationary MDPs. Hallak, Di Castro, and Mannor (2015) attempted learning in a non-stationary environment by clustering transitions to identify tasks and classifying the tasks. Da Silva et al. (2006) proposed RLCD that measures the confidence value for a partial model, called a quality signal, which can be used to select a suitable model from a pool of models at each time. A new model is added into the pool after learning, when the qual-

ity signal value is less than some threshold. Similarly, Padakandla, Prabuchandran, and Bhatnagar (2020) introduced the context Q-learning method in which the online parameter Dirichlet checkpoint is used to detect points of task changes over time. Recently, Xie, Harrison, and Finn (2020) and Zintgraf et al. (2020) proposed task embedding methods by predicting the next state and reward value and finding the representation of a task.

Transfer learning and meta learning are relevant to non-stationary environment problems in that those are focused on the benefits of exploiting a set of heterogeneous tasks to build such a model that can be robust in adaptation to a specific target task (Abhishek Gupta and Levine 2017; Tan et al. 2018, Parisotto, Ba, and Salakhutdinov 2015, Laroche and Barlier 2017) (i.e., transfer learning) or new tasks (Finn, Abbeel, and Levine 2017, Humplík et al. 2019, Rakelly et al. 2019) (i.e., meta learning).

Abhishek Gupta and Levine (2017) proposed a knowledge transfer method between different environments in the presence of morphological differences. If two environments share similar representations in the latent space of states and actions, their method extracts invariant features from the environments by using autoencoders.

In the same vein of Abhishek Gupta and Levine, we find out invariant features from environments. While Abhishek Gupta and Levine focused on extracting common knowledge from different environments which have same structure, our method is intended to estimate the time-variant hidden parameters by inspecting the difference of time-variant tasks between two environments.

The structure learning (or functorial property of gradient descent) has been studied in the area of the applied category theory. Gavranović (2020) explained that gradient descent conducts the structure learning especially in CycleGAN. He claims that CycleGAN learns to decompose the tasks in images, maps them into each task space, and transforms the images to the specific task part. While his work provides theoretical foundation about structural learning of CycleGAN, our work focuses on implementing and evaluating the structural learning for non-stationary environments.

Conclusion

In this work, we presented a novel RL model to address the problem of learning a non-stationary MDP, which combines CycleGAN-based structural learning for task decomposition and continual learning for rapid adaptation. Our model demonstrates robust performance in non-stationary environments with various dynamics and task uncertainty settings, compared to other algorithms, and verifies its applicability to RL-based control scenarios.

The direction of our future works is to adapt our method with safe constraints for a mission-critical cyber-physical application in which the environment dynamics level increases over time, as multiple agents make continual control decisions independently without coordinating each other.

Acknowledgement

We would like to thank anonymous reviewers for their valuable comments and suggestions.

This work was supported by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant 2021-0-00900, by the ICT Creative Consilience Program supervised by the IITP under Grant IITP-2020-0-01821, by the National Research Foundation of Korea (NRF) funded by the Korean Government (MSIT) under Grant 2020M3C1C2A01080819, and by Samsung Electronics.

References

- Abhishek Gupta, Y. L. P. A., Coline Devin; and Levine, S. 2017. Learning invariant feature spaces to transfer skills with reinforcement learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.
- Bellman, R. 1956. A problem in the sequential design of experiments. *Sankhyā: The Indian Journal of Statistics (1933-1960)*, 16(3/4): 221–229.
- Catto, E. 2012. pybox2d. URL <https://github.com/pybox2d/pybox2d>.
- Chandak, Y.; Theocharous, G.; Shankar, S.; White, M.; Mahadevan, S.; and Thomas, P. 2020. Optimizing for the future in non-stationary mdps. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 1414–1425. PMLR.
- Choi, S. P. M.; Yeung, D.-Y.; and Zhang, N. L. 2001. *Hidden-Mode Markov Decision Processes for Nonstationary Sequential Decision Making*. Berlin, Heidelberg: Springer Berlin Heidelberg.
- Da Silva, B. C.; Basso, E. W.; Bazzan, A. L.; and Engel, P. M. 2006. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine learning (ICML)*, 217–224. PMLR.
- Doshi-Velez, F.; and Konidaris, G. 2016. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, 1432. NIH Public Access.
- Farajtabar, M.; Azizan, N.; Mott, A.; and Li, A. 2020. Orthogonal gradient descent for continual learning. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 3762–3773. PMLR.
- Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 1126–1135. PMLR.
- Gajane, P.; Ortner, R.; and Auer, P. 2019. Variational regret bounds for reinforcement learning. *arXiv:1905.05857*.
- Gavranović, B. 2020. Learning Functors using Gradient Descent. *arXiv:2009.06837*.
- Hallak, A.; Di Castro, D.; and Mannor, S. 2015. Contextual markov decision processes. *arXiv 1502.02259*.
- Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.
- Humplik, J.; Galashov, A.; Hasenclever, L.; Ortega, P. A.; Teh, Y. W.; and Heess, N. 2019. Meta reinforcement learning as task inference. *arXiv:1905.06424*.
- Igl, M.; Zintgraf, L.; Le, T. A.; Wood, F.; and Whiteson, S. 2018. Deep variational reinforcement learning for POMDPs. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2117–2126. PMLR.
- Jaksch, T.; Ortner, R.; and Auer, P. 2010. Near-optimal Regret Bounds for Reinforcement Learning. *Journal of Machine Learning Research*, 11(51): 1563–1600.
- Laroche, R.; and Barlier, M. 2017. Transfer reinforcement learning with shared dynamics. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*.
- Lee, A. X.; Nagabandi, A.; Abbeel, P.; and Levine, S. 2019. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv:1907.00953*.
- Leon, S. J.; de Pillis, L.; and De Pillis, L. G. 2015. *Linear algebra with applications*. Pearson Boston.
- Padakandla, S.; Prabuchandran, K.; and Bhatnagar, S. 2020. Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence*, 50(11): 3590–3606.
- Parisotto, E.; Ba, J. L.; and Salakhutdinov, R. 2015. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv:1511.06342*.
- Rakelly, K.; Zhou, A.; Finn, C.; Levine, S.; and Quillen, D. 2019. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 5331–5340. PMLR.
- Ravindran, B.; and Barto, A. G. 2004. *An algebraic approach to abstraction in reinforcement learning*. Ph.D. thesis, University of Massachusetts at Amherst.
- Shital Shah, C. L., Debadeepa Dey; and Kapoor, A. 2017. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In *Proceedings of the Field and Service Robotics*.
- Sorg, J.; and Singh, S. 2009. Transfer via soft homomorphisms. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 741–748.
- Tan, J.; Zhang, T.; Coumans, E.; Iscen, A.; Bai, Y.; Hafner, D.; Bohez, S.; and Vanhoucke, V. 2018. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv:1804.10332*.
- Wang, Y.; and Tan, X. 2021. Deep Recurrent Belief Propagation Network for POMDPs. volume 35, 10236–10244.
- Xie, A.; Harrison, J.; and Finn, C. 2020. Deep reinforcement learning amidst lifelong non-stationarity. *arXiv:2006.10701*.
- Yu, T.; Kumar, S.; Gupta, A.; Levine, S.; Hausman, K.; and Finn, C. 2020. Gradient surgery for multi-task learning. *arXiv:2001.06782*.
- Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the 16th IEEE International Conference on Computer Vision (ICCV)*, 2223–2232.

Zintgraf, L.; Shiarlis, K.; Igl, M.; Schulze, S.; Gal, Y.; Hofmann, K.; and Whiteson, S. 2020. Varibad: a very good method for bayes-adaptive deep rl via meta-learning. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*.