Option2Vec: Learning Temporal-State Abstraction Embeddings on the MDP

Chang Li, Dongjin Song, and Dacheng Tao, *Fellow, IEEE*

Abstract—The option framework develops options on a sample inefficient Semi-Markov Decision Process (SMDP) and represents each option computational expensively as two distributions and one initiation set. In this paper, we present the option-induced SMDP as a simple Hidden-Markov-Model-style Probabilistic Graphical Model (PGM), which enables representing each option as efficient as one embedding vector (hidden variable). We consider temporal and state abstraction jointly. In particular, we formulate the option framework as a Hidden-Markov-Model-style Probabilistic Graphical Model (PGM), thus enabling each option to be parameterized as a compact embedding vector. To optimize this PGM, we first propose a novel *Markovian Option-Value function*, and prove it is an unbiased estimation of the conventional value function. We then derive a two-stage policy gradient theorem based on the above value function. Finally, we implement this PGM upon the Transformer architecture to encode options into fixed-length embeddings, named Option2Vec. Extensive experiments on challenging *Mujoco* environments demonstrate Option2Vec's efficiency and effectiveness: under widely used configuration, with merely 15.8% parameters, Option2Vec achieves competitive, if not better, performance compared to the state-of-the-art baselines on all finite horizon and transfer learning environments. Moreover, Option2Vec significantly outperforms all baselines on infinite horizon environments while exhibiting smaller variance, faster convergence, and interpretability.

Index Terms—The Option Framework, Hierarchical Reinforcement Learning, Markov Decision Process

1 Introduction

The option framework [1] is one of the most promising framework to extend RL methods to lifelong learning agents [2] and has been proved beneficial in speeding learning [3], improving exploration [4], and facilitating transfer learning [5]. However, conventional options tend to over-complicate methods in ways that are less suited to leveraging computation [3].

The first deficiency is that the option framework is developed on Semi-Markov Decision Process, we refer to this as *SMDP-Option*. In *SMDP-Option*, an option is a temporally abstracted action whose execution cross a various amount of time steps. *SMDP-Option* has been identified [5] as sample inefficient and unstable to optimize. Since RL is notoriously sample expensive and hyper-parameters sensitive [6], the SMDP formulation severely impair options' applicability in broader context [7].

We address this issue by first proving that *SMDP-Option* has a sample efficient Markov Decision Process equivalence (*bisimulation relation* [8]), we refer to this as *MDP-Option*. In RL [5, 9] and Imitation Learning [10, 11, 12, 13] areas, similar formulations have been employed as "one-step option" [5, 10], such approximations either drop the dependency on $\mathbf{o}_{t-1}$ (option executed from last steps) thus lose the temporal abstraction functionality, or use an inaccurate value function to update policies. Instead, we are the first identify the issue that the conventional *Value Function* $V[\mathbf{s}_t]$ no longer yields the Bellman equation [1] under the one-step setting, and preserve the temporal abstraction by proposing a novel *Markovian Option-Value Function* $\bar{V}[\mathbf{s}_t, \mathbf{o}_{t-1}]$, which is an unbiased estimation of $V[\mathbf{s}_t]$ and its variance is up-bounded by $V[\mathbf{s}_t]$, and derive the novel Bellman equation for *MDP-Option*. Based on the Bellman equation, we not only prove the equivalence to *SMDP-Option*, but also derive policy gradient theorems for learning *MDP-Option*. As a result, *MDP-Option* is a general-purpose MDP which can be combined with any MDP-style [5] policy optimization algorithms (such as PPO [14]) off-the-shelf.

The second deficiency of *SMDP-Option* is that it is extremely expensive to learn and scale up. Each option is represented as a triple containing three components: one *intra-option policy*, one *termination function*, and one initiation set. Learning options is amenable to learning local representations [3] on a statistical manifold [15]. As pointed out by **(author?)** [3] (Chapter 3.6), local representations do a poor job at representing knowledge compactly and require more samples than distributed representations.

In this paper, we make the first attempt to learn options with embeddings (distributed representations [16]). Distributed representations have proved its efficacy in representing entities and played a central role in recent advances of large-scale frameworks in both CV [17, 18] and NLP [19, 20, 21] areas. As shown in Section 4, *MDP-Option* naturally gives rise to representing each option as a single embedding vector and the option space as an ambient space of the state space. Therefore, options defined in *MDP-Option* combine temporal abstraction together with state abstraction [22]. To learn option embeddings, we propose *Option2Vec* (O2V) architecture, a simple yet effective Attention [19] based Encoder-Decoder architecture. Complexities of learning option distributions and classification hyperplanes on statistical manifold are simplified as an efficient clustering mechanism

- *Chang Li and Dacheng Tao were with the UBTECH Sydney AI Centre, School of Computer Science, University of Sydney, Sydney, NSW 2000, Australia. E-mail:* chli4934@uni.sydney.edu.au dacheng.tao@sydney.edu.au
- *Dongjin Song University.*

Manuscript received December 19, 2021

over option embedding centroids on a homeomorphic parametric space [15].

It is worth to point out that, due to space limitation we have to solely focus on proposing *MDP-Option* and O2V, and designing experiments to address that O2V achieves at least the same performance as *SMDP-Option*. Since *MDP-Option* is equivalent to *SMDP-Option*, it still shares many identified limitations such as "the dominant skill problem" [5, 23] as identified in Section 5.4. As briefly discussed in Appendix **??**, *MDP-Option* actually gives rise to efficient solutions to many identified limitations of *SMDP-Option* yet have to be deferred to our future works. Our main contributions are: (1) proposing the *MDP-Option*, a MDP equivalence of *SMDP-Option*, and developing the bellman equation and gradient theorems; (2) proposing option embedding vectors, the first distributed representations of options; (3) proposing the first Attention [19] based Encoder-Decoder architecture, the Option2Vec (O2V) architecture, for learning options with much better scalability, smaller variance and faster convergence; (4) demonstrating option embeddings are interpretable, which is a key property for developing real-world RL applications (e.g. ensuring safety for human).

2 Background

Markov Decision Process: A Markov Decision Process [24] $M = \{\mathbb{S}, \mathbb{A}, R, P, \gamma\}$ consists of a state space $\mathbb{S}$, an action space $\mathbb{A}$, a state transition function $P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) : \mathbb{S}\times\mathbb{A} \to \mathbb{S}$, a discount factor $\gamma \in \mathbb{R}$, and a reward function $R(\mathbf{s}, \mathbf{a}) = \mathbb{E}[r|\mathbf{s}, \mathbf{a}] : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ which is the expectation of the reward $r_{t+1} \in \mathbb{R}$ received from the environment after executing action $\mathbf{a}_t$ at state $\mathbf{s}_t$. A policy $\pi = P(\mathbf{a}|\mathbf{s}) : \mathbb{A} \times \mathbb{S} \to [0, 1]$ is a probability distribution defined over actions conditioning on states. A discounted return is defined as $G_t = \sum_k^N \gamma^k r_{t+k+1}$, where $\gamma \in (0, 1)$ is a discounting factor. The value function $V[\mathbf{s}_t] = \mathbb{E}_{\tau\sim\pi}[G_t|\mathbf{s}_t]$ is the expected return starting at state $\mathbf{s}_t$ and the trajectory $\tau = \{\mathbf{s}_t, \mathbf{a}_t, r_{t+1}, \mathbf{s}_{t+1}, \dots\}$ follows policy π thereafter. The action-value function is defined as $Q[\mathbf{s}_t, \mathbf{a}_t] = \mathbb{E}_{\tau\sim\pi}[G_t|\mathbf{s}_t, \mathbf{a}_t]$.

Bisimulation Relation: Given two processes $M = \{\mathbb{S}, \mathbb{A}, R, P, \gamma\}$ with the trajectory τ and $\tilde{M} = \{\tilde{\mathbb{S}}, \mathbb{A}, \tilde{R}, \tilde{P}, \tilde{\gamma}\}$ with the trajectory $\tilde{\tau}$. Assume both M and $\tilde{M}$ share the same action space $\mathbb{A}$. The equivalence relation between M and $\tilde{M}$ is defined by **(author?)** [8]. An equivalence relation $\tilde{B} : \tilde{\mathbb{S}} \to \mathbb{S}$ is a *Bisimulation Relation* if 1) for any state $\mathbf{s}$, there exists an *one-to-one correspondence* equivalent state $\tilde{\mathbf{s}}$ that $\mathbf{s}/\tilde{B} = \tilde{\mathbf{s}}/\tilde{B}$, or denoted as $\tilde{B}(\tilde{\mathbf{s}}) = \mathbf{s}$, 2) and the following conditions hold:

1) $P(\tau/\tilde{B}) \equiv P(\tilde{\tau}/\tilde{B})$, and $\tilde{B}$ is a *bijection*,

2) $V[\tau/\tilde{B}] \equiv V[\tilde{\tau}/\tilde{B}]$

In this paper, we follow this definition to prove the equivalence relationship.

The SMDP-based Option Framework: In *SMDP-Option* [1, 3], an option is a triple $(\mathbb{I}_o, \pi_o, \beta_o) \in \mathcal{O}$, where $\mathcal{O}$ denotes the option set; the subscript $o \in \mathbb{O} = \{1, 2, \dots, K\}$ is a positive integer index which denotes the oth triple where K is the number of options; $\mathbb{I}_o$ is an initiation set indicating where the option can be initiated; $\pi_o = P_o(\mathbf{a}|\mathbf{s}) : \mathbb{A} \times \mathbb{S} \to [0, 1]$ is the action policy of the oth option; $\beta_o = P_o(\mathbf{b} = 1|\mathbf{s}) : \mathbb{S} \to [0, 1]$ where $\mathbf{b} \in 0, 1$ is a *termination function*. For clarity reasons, we use $P_o(\mathbf{b} = 1|\mathbf{s})$ instead of β_o which is widely used in previous option literatures (e.g. [1, 25]).

A *master policy* $\pi(\mathbf{o}|\mathbf{s}) = P(\mathbf{o}|\mathbf{s})$ where $\mathbf{o} \in \mathbb{O}$ is used to sample which option will be executed. Note that we use the bold-case $\mathbf{o}$ to denote unrealized random variables and the light-italic-case o to denote a realized instantiation. Conventionally, the execution of an option employs the call-and-return model [1]: at time step t, an agent either continues the previously executed option $\mathbf{o}_{t-1} = o$ with probability $P_o(\mathbf{b} = 0|\mathbf{s})$ and sets $\mathbf{o}_t = \mathbf{o}_{t-1} = o$, or terminates o with probability $P_o(\mathbf{b} = 1|\mathbf{s})$ and samples a new option $\mathbf{o}_t$ from the master policy $P(\mathbf{o}_t|\mathbf{s}_t)$. Therefore, the dynamics (stochastic process) of the option framework is written as:

$$P(\tau) = P(\mathbf{s}_0)P(\mathbf{o}_0)P_{o_0}(\mathbf{a}_0|\mathbf{s}_0)\prod_{t=1}^{\infty} P(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1})P_{o_t}(\mathbf{a}_t|\mathbf{s}_t) [P_{o_{t-1}}(\mathbf{b}_t = 0|\mathbf{s}_t)\mathbf{1}_{\mathbf{o}_t=o_{t-1}} + P_{o_{t-1}}(\mathbf{b}_t = 1|\mathbf{s}_t)P(\mathbf{o}_t|\mathbf{s}_t)]. \tag{1}$$

where $\tau = \{\mathbf{s}_0, \mathbf{o}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{o}_1, \mathbf{a}_1, \dots\}$ denotes the trajectory of the option framework. $\mathbf{1}$ is an indicator function and is only true when $\mathbf{o}_t = o_{t-1}$ (notice that o_{t-1} is the realization at $\mathbf{o}_{t-1}$). Therefore, under this formulation the option framework is defined as a Semi-Markov process since the dependency on an activated option o can cross a variable amount of time [1].

3 MDP Equivalences of the SMDP-based Option Framework

We prove that *MDP-Option* is equivalent to *SMDP-Option* under the definition of *bisimulation* [8]. To derive Bellman equation for *MDP-Option*, we develop a novel *Markovian skill-value function* $\bar{V}[\mathbf{s}_t, \mathbf{o}_t]$, which is an unbiased estimation of the conventional value function $V[\mathbf{s}_t]$ and the variance of $\bar{V}[\mathbf{s}_t, \mathbf{o}_t]$ is up-bounded by $V[\mathbf{s}_t]$. Based on Bellman equation, policy gradient theorems for *MDP-Option* are then derived. As a result, *MDP-Option* is a general-purpose MDP which can be combined with any policy optimization algorithm off-the-shelf.

In this section, we propose *MDP-Option*, a simple yet effective option-induced MDP and prove its equivalence (as shown in Figure 1) to *SMDP-Option*. For clarity, in Section 3.1 we first prove an intermediate equivalence *MDP-Mixture* to bridge the equivalence between *SMDP-Option* and *MDP-Option*. Based on *MDP-Mixture*, in Section **??** we propose the *MDP-Option*, a marginalized variation of the *SMDP-Option*. *MDP-Option* uses the *skill policy* (Eq. **??**), which is a marginal distribution, to replace the *master policy* and *termination function*. In order to derive *MDP-Option*'s Bellman equation, we propose the novel *Markovian skill-value function* (Eq. 9) and prove that it is an unbiased estimation of the conventional value function and its variance is up-bounded by the conventional value function. Policy gradient theorems for *MDP-Option* are then derived basing on the Bellman equation. In Section 4, we propose O2V, which is an implementation of the *MDP-Option* by employing the Embedding and Attention [19] techniques.

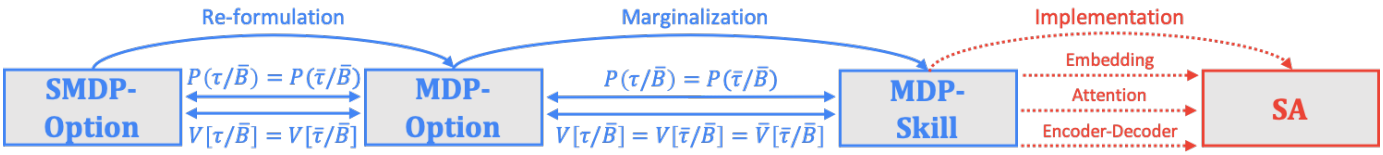


Fig. 1: Bisimulation Relations. Blue terms are Decision Processes. Solid arrows indicate that their equivalences are theoretically justified. O2V is an Architecture which implements the MDP-Option by employing Embedding, Attention, and Encoder-Decoder techniques. Dashed connections indicate they are architecture choices.

3.1 The MDP Equivalence (MDP-Mixture) of the Option Framework

With the definitions of *SMDP-Option* in hand, we now show how to reformulate it into an MDP-based equivalence *MDP-Mixture*. The first reformulation is that we follow **(author?)** [26]'s formulation of mixture distributions and redefine the option random variable $\mathbf{o} \in \mathbb{O} = \{1, 2, \ldots, K\}$, which was originally defined as an integer index, but now as a K-dimensional one-hot vector $\bar{\mathbf{o}} \in \bar{\mathbb{O}} = \{0, 1\}^K$ where K is the number of options. The second reformulation is that we exploit the one-hot vector to reformulate the *termination function* and action function of each option into two mixture distributions by introducing extra dependencies on $\bar{\mathbf{o}}$:

$$P(\mathbf{a}_t|\mathbf{s}_t, \bar{\mathbf{o}}_t) = \prod_{o \in \bar{\mathbf{o}}_t} P_o(\mathbf{a}_t|\mathbf{s}_t)^o,$$
$$P(\mathbf{b}_t|\mathbf{s}_t, \bar{\mathbf{o}}_{t-1}) = \prod_{o \in \bar{\mathbf{o}}_{t-1}} P_o(\mathbf{b}_t|\mathbf{s}_t)^o, \tag{2}$$

Since the option random variable $\bar{\mathbf{o}}$ is now a one-hot vector, for $\bar{\mathbf{o}}_t = o_t$, by definition only the entry $o_t = 1$ and all the other entry $o \in \mathbb{O} - \{o_t\} = 0$. Therefore, we have $P_{o_t}(\mathbf{a}_t|\mathbf{s}_t) = P(\mathbf{a}_t|\mathbf{s}_t, \bar{\mathbf{o}}_t = o_t)$ and $\beta_{o_{t-1}} = P_{o_{t-1}}(\mathbf{b}_t = 1|\mathbf{s}_t) = P(\mathbf{b}_t = 1|\mathbf{s}_t, \bar{\mathbf{o}}_{t-1} = o_{t-1})$.

The third reformulation is that we propose a novel *MDP mixture master policy* $P(\bar{\mathbf{o}}_t|\mathbf{s}_t, \mathbf{b}_t, \bar{\mathbf{o}}_{t-1})$, which is a mixture distribution containing the *SMDP master policy* and a degenerate probability as mixture components by adding two extra dependencies on $\mathbf{b}_t$ and $\bar{\mathbf{o}}_{t-1}$:

$$P(\bar{\mathbf{o}}_t|\mathbf{s}_t, \mathbf{b}_t, \bar{\mathbf{o}}_{t-1}) = P(\bar{\mathbf{o}}_t|\mathbf{s}_t)^{\mathbf{b}_t} P(\bar{\mathbf{o}}_t|\bar{\mathbf{o}}_{t-1})^{1-\mathbf{b}_t}, \tag{3}$$

where the indicator function $\mathbf{1}_{\mathbf{o}_t = o_{t-1}}$ used in Eq.1 is now redefined as a degenerate probability distribution [24]:

$$P(\bar{\mathbf{o}}_t|\bar{\mathbf{o}}_{t-1}) = \begin{cases} 1 & \text{if } \bar{\mathbf{o}}_t = \bar{\mathbf{o}}_{t-1}, \\ 0 & \text{if } \bar{\mathbf{o}}_t \neq \bar{\mathbf{o}}_{t-1}. \end{cases}$$

We define a function $\bar{B}(\bar{\mathbf{o}}) = \bar{\mathbf{o}} \cdot \mathbf{d}^T : \bar{\mathbb{O}} \to \mathbb{O}$ which maps $\bar{\mathbf{o}}$ to $\mathbf{o}$, where $\mathbf{d} = [1, 2, \ldots, K]^T$ is a K-dimensional constant integer vector and hence $\bar{B}(\bar{\mathbf{o}}) = \mathbf{o}$. Note that $\bar{B}$ is a *Bijection* since it is a linear function defined on a finite integer space. Therefore, by following the definition of *Bisimulation Relation*, the dynamics of the *SMDP-Option* in Eq.1 under the Bijection $\bar{B}$ can be reformulated as:

$$\begin{aligned} P(\tau/\bar{B}) =& P(\bar{\tau}/\bar{B}) \\ =& P(\mathbf{s}_0)P(\bar{\mathbf{o}}_0)P(\mathbf{a}_0|\mathbf{s}_0, \bar{\mathbf{o}}_0) \prod_{t=1}^{\infty} P(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1}) \\ & P(\mathbf{a}_t|\mathbf{s}_t, \bar{\mathbf{o}}_t) \sum_{\mathbf{b}_t} P(\mathbf{b}_t|\mathbf{s}_t, \bar{\mathbf{o}}_{t-1}) P(\bar{\mathbf{o}}_t|\mathbf{b}_t, \mathbf{s}_t, \bar{\mathbf{o}}_{t-1}) \end{aligned} \tag{4}$$

where $\bar{\tau} = \{\mathbf{s}_0, \bar{\mathbf{o}}_0, \mathbf{a}_0, \mathbf{s}_1, \bar{\mathbf{o}}_1, \mathbf{a}_1, \ldots\}$ is the trajectory of the *MDP-Mixture*.

With $P(\tau/\bar{B}) = P(\bar{\tau}/\bar{B})$ in hand, to prove the equivalence between the *SMDP-Option* and *MDP-Mixture*, we move on to prove both of them share the same expected reward. This is non-trivial since compared to the *SMDP-Option*, the MDP formulation introduces extra dependencies on $\bar{\mathbf{o}}$ and $\mathbf{b}$ in Eq.4 as described above. However, in Appendix **??**, by exploiting conditional independencies we prove that they do have the same expected return under the Bijection $\bar{B}$. Therefore, the SMDP-based option framework has an MDP-based equivalence:

Theorem 3.1. *By the definition of Bisimulation Relation, the SMDP-based option framework, which employs Markovian options, has an underlying MDP equivalence because:*

1) $P(\tau/\bar{B}) = P(\bar{\tau}/\bar{B})$ (Eq. 4) and $\bar{B}$ is a Bijection.

2) $V[\tau/\bar{B}] = V[\bar{\tau}/\bar{B}]$ (Proofs in Appendix **??**).

3.2 Representing Options as Temporal-State Abstraction Embeddings

In this subsection we answer two questions: (1) how to encode knowledge of $Option^{\text{Triple}}$ with option embeddings, *i.e.*, $Option^{\text{Embed}}$; and (2) how to achieve temporal abstractions by replacing the *call-and-return* mode with a simple *clustering* mechanism. We argue that for learning temporal abstractions, high-level actions, *i.e.*, $Option^{\text{Triple}}$, and the *call-and-return* mode are not mandatory: temporal abstractions can be achieved by replacing $Option^{\text{Triple}}$ with the a newly defined temporal-state transition-invariant embeddings of hidden variables on which the action policy is conditioned.

We start by reviewing the weaknesses of defining options as high-level actions. Specifically, each $Option^{\text{Triple}}$, *i.e.*, a set of distributions, is analogously a point on a statistical manifold [15], which is computationally expensive to learn. Most literatures ignore *the initiation* set because of difficulties in learning it from data [**?**]. Learning *master policy* is analogously learning classification hyperplanes [2] on the statistical manifold. For M hyperplanes, theoretically there

are 2^M options to be learned as shown in Figure 2 (left). As pointed out by **(author?)** [3] (Chapter 3.6), $Option^{\text{Triple}}$ cannot at represent knowledge compactly and requires more samples to achieve good performance.

On the other hand, as suggested by Eq. (**??**), temporal abstractions can be achieved more efficiently only by a simple *option policy* $P(\bar{\mathbf{o}}_t|\mathbf{s}_t, \bar{\mathbf{o}}_{t-1})$ as long as there is sufficient context information encoded in $\bar{\mathbf{o}}$. Inspired by word embeddings in Word2Vec [**?**], we address this issue by representing options as more compact and computationally efficient embeddings (*i.e.*, semantic space of options). Specifically, we define a parameter matrix $\boldsymbol{W}_{\mathbf{o}} = [\hat{\mathbf{o}}_1, ..., \hat{\mathbf{o}}_K] \in \mathbb{R}^{D\times K}$, where D is the dimension of the vector and K is number of options. Each real-valued vector $\hat{\mathbf{o}}$ is referred to as an $Option^{\text{Embed}}$. As in transformer [19], the option embedding is retrieved from the embedding matrix by $\hat{\mathbf{o}} = \boldsymbol{W}_{\mathbf{o}}\bar{\mathbf{o}}$. Under this formulation, the hidden variable $\bar{\mathbf{o}}$ remains the random variable of the *option policy*, while the option embedding matrix $\boldsymbol{W}_{\mathbf{o}}$ is simply the distribution's parameters:

$$\begin{aligned} P(\bar{\tau}) =& P(\mathbf{s}_0)P(\bar{\mathbf{o}}_0)P(\mathbf{a}_0|\mathbf{s}_0, \bar{\mathbf{o}}_0)\prod_{t=1}^{\infty} P(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1}) \\ & P(\mathbf{a}_t|\mathbf{s}_t, \hat{\mathbf{o}}_t)P(\bar{\mathbf{o}}_t|\mathbf{s}_t, \bar{\mathbf{o}}_{t-1}; \boldsymbol{W}_{\mathbf{o}}) \end{aligned} \tag{5}$$

where the action policy employs $\boldsymbol{W}_{\mathbf{o}}$ as a realized hidden variable rather than parameters, such that $P(\mathbf{a}_t|\mathbf{s}_t, \hat{\mathbf{o}}_t = \boldsymbol{W}_{\mathbf{o}}\bar{\mathbf{o}}) = P(\mathbf{a}_t|\mathbf{s}_t, \bar{\mathbf{o}}_t, \boldsymbol{W}_{\mathbf{o}})$. Note that rather than a mixture of K distributions defined by Eq. (**??**), the action policy is now a single decoder which is shared by all option embeddings. All local information of an $Option^{\text{Triple}}$ (where to initiate, what actions to emit and when to terminate) are parameterized as an embedding vector, *i.e.*, $Option^{\text{Embed}}$.

As shown in Figure 2 (right), $Option^{\text{Embed}}$s are actually clustering centroids on a parametric space, which is homeomorphic to the statistical manifold. The *option policy* maps the observed pair $[\mathbf{s}_t, \hat{\mathbf{o}}_{t-1}]$ to a point (detailed implementation is available in Section 4) on the parametric space. The decision of selecting an option $\hat{\mathbf{o}}_t$ can be made by simply assigning the point to the closest centroid $\hat{\mathbf{o}}^*$, whose distance is estimated efficiently by employing the Attention mechanism [19]. Because a vector is closest to itself, this mechanism has a natural tendency to continue $\hat{\mathbf{o}}_t = \hat{o}_{t-1}$, yet a significantly different state $\mathbf{s}_t$ will pull the point far enough from $\hat{\mathbf{o}}_{t-1}$ and result in another option centroid being assigned. Because the parametric space is now an ambient space of both state space and option space, $Option^{\text{Embed}}$ combines advantages from both temporal abstraction and state abstraction [22].

3.3 Policy Gradient-based Optimization Algorithms

Computing gradients of Eq. (5) directly is intractable because it depends on the state transition probability that is unknown under the model-free RL setting. We address this issue by first following the RL literature [32] and define value functions as expected returns conditioned on the PGM's dynamics. We then propose a novel *Markovian option-value function* to enable the derivation of the Bellman equation. We finally derive a sample efficient policy gradient-based theorems [32], which do not involve the state distribution's gradients, along the expansion of the Bellman equation. Due to space limitations, we provide the details of all proofs in Appendix **??**.

Following the RL literature [32], we derive the Bellman equation by recursively expanding value functions. Similar to the value function in Section **??**, we define the *option-value function* as the expected return $G_t = \sum_k^N \gamma^k r_{t+k+1}$ conditioned on the current state and option:

$$Q_O[\mathbf{s}_t, \bar{\mathbf{o}}_t] = \mathbb{E}[G_t|\mathbf{s}_t, \bar{\mathbf{o}}_t] \tag{6}$$

We can further expand the *option-value function* as the expectation of an *option-action-value function* $Q_A[\mathbf{s}_t, \bar{\mathbf{o}}_t, \mathbf{a}_t]$ by exploiting conditional independencies encoded in the PGM:

$$\begin{aligned} Q_O[\mathbf{s}_t, \bar{\mathbf{o}}_t] &= \mathbb{E}[G_t|\mathbf{s}_t, \bar{\mathbf{o}}_t] = \sum_{\mathbf{a}_t} P(\mathbf{a}_t|\mathbf{s}_t, \bar{\mathbf{o}}_t)\mathbb{E}[G_t|\mathbf{s}_t, \bar{\mathbf{o}}_t, \mathbf{a}_t] \\ &= \sum_{\mathbf{a}_t} P(\mathbf{a}_t|\mathbf{s}_t, \bar{\mathbf{o}}_t)Q_A[\mathbf{s}_t, \bar{\mathbf{o}}_t, \mathbf{a}_t] \end{aligned} \tag{7}$$

In order to establish the connection between the current value and its successors, *i.e.*, deriving the Bellman equation, we need to further expand the $Q_A[\mathbf{s}_t, \bar{\mathbf{o}}_t, \mathbf{a}_t]$ to the next time step:

$$\begin{aligned} Q_A[\mathbf{s}_t, \bar{\mathbf{o}}_t, \mathbf{a}_t] &= \mathbb{E}[G_t|\mathbf{s}_t, \bar{\mathbf{o}}_t, \mathbf{a}_t] \\ &= r(s, a) + \gamma \sum_{\mathbf{s}_{t+1}} P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\mathbb{E}[G_t|\mathbf{s}_{t+1}, \bar{\mathbf{o}}_t] \end{aligned} \tag{8}$$

However, as shown in Eq. (8), the last term has an extra dependency on the hidden variable $\bar{\mathbf{o}}_t$. As a result, the conventional value function $V[\mathbf{s}_t]$ does not yield the recursive formulation required for deriving the Bellman equation. This issue has not been identified in related literatures [5, 10, 11, 12, 13] which employ the *option policy* as "one-step option". In order to address this issue, we propose a novel *Markovian option-value function* $\bar{V}[\mathbf{s}_{t+1}, \bar{\mathbf{o}}_t] = \mathbb{E}[G_t|\mathbf{s}_{t+1}, \bar{\mathbf{o}}_t]$ and prove that:

Proposition 3.2. *$\bar{V}[\mathbf{s}_t, \bar{\mathbf{o}}_{t-1}]$ is an unbiased estimation of $V[\mathbf{s}_t]$.*

Proposition 3.3. *The variance of $\bar{V}[\mathbf{s}_t, \bar{\mathbf{o}}_{t-1}]$ is up-bounded by $V[\mathbf{s}_t]$.*

Proofs are available in Appendix **??**. The variance-reduction effect is empirically verified in Section 5. We can then employ the *Markovian option-value function* $\bar{V}[\mathbf{s}_t, \bar{\mathbf{o}}_{t-1}]$ to derive the Bellman equation:

$$\bar{V}[\mathbf{s}_{t+1}, \bar{\mathbf{o}}_t] = \mathbb{E}[G_{t+1}|\mathbf{s}_{t+1}, \bar{\mathbf{o}}_t] = \sum_{\bar{\mathbf{o}}_{t+1}} P(\bar{\mathbf{o}}_{t+1}|\mathbf{s}_{t+1}, \bar{\mathbf{o}}_t)Q_O[\mathbf{s}_{t+1}, \bar{\mathbf{o}}_{t+1}]. \tag{9}$$

Expanding Q_O in Eq. (9) through Eq. (6) to Eq. (8) gives the recursive formulation of the Bellman equation. With the Bellman equation in hand, we now are able to derive policy gradient theorems (Note that to keep notations uncluttered, we use $\theta_{\bar{o}}$ to denote *option policy*'s parameters $P(\bar{\mathbf{o}}_t|\mathbf{s}, \bar{\mathbf{o}}_{t-1}; \theta_{\bar{o}})$ and θ_a to denote action policy's parameters $P(\mathbf{a}_t|\mathbf{s}_t, \bar{\mathbf{o}}_t; \theta_a)$):

Theorem 3.4. ***Option Policy Gradient Theorem:*** *Given a stochastic* option policy *differentiable in its parameter vector $\theta_{\bar{o}}$, the gradient of the expected discounted return with respect to $\theta_{\bar{o}}$ is:*

$$\frac{\partial \bar{V}[\mathbf{s}_t, \bar{\mathbf{o}}_{t-1}]}{\partial \theta_{\bar{o}}} = \mathbb{E}[\ \frac{\partial P(\bar{\mathbf{o}}'|\mathbf{s}', \bar{\mathbf{o}})}{\partial \theta_{\bar{o}}} Q_O[\mathbf{s}', \bar{\mathbf{o}}'] \mid \mathbf{s}_t, \bar{\mathbf{o}}_{t-1}], \tag{10}$$

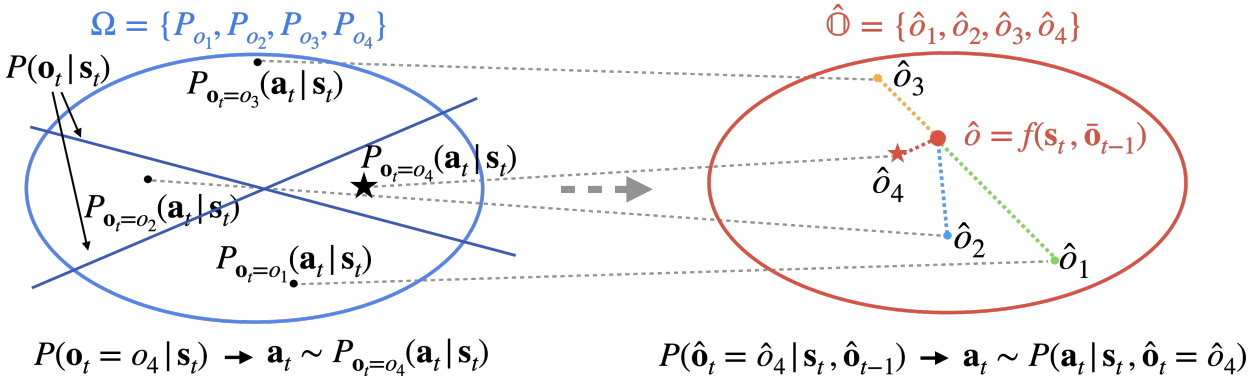


Fig. 2: $Option^{\text{Triple}}$ Classification on Statistical Manifold (left) v.s. $Option^{\text{Embed}}$ Clustering on Parametric Space (right). On Statistical Manifold, for M options there are M intra-option policies to learn (for clarity we omit $P_o(\mathbf{b}|\mathbf{s})$ and $\mathbb{I}_o$). Selecting options is analogously learning classification hyperplanes (blue lines). On Parametric Space, for M options there are M embedding centroids to learn yet all embeddings share one action policy (decoder). Selecting options is analogously assigning the closest centroid to the point $[\mathbf{s}_t, \mathbf{o}_{t-1}]$.

where $\bar{\mathbf{o}}'$ is one time step later than $\bar{\mathbf{o}}$.

Theorem 3.5. ***Action Policy Gradient Theorem:*** *Given a stochastic action policy differentiable in its parameter vector θ_a, the gradient of the expected discounted return with respect to θ_a is:*

$$\frac{\partial Q_O[\mathbf{s}_t, \bar{\mathbf{o}}_t]}{\partial \theta_a} = \mathbb{E}[\ \frac{\partial P(\mathbf{a}|\mathbf{s}, \bar{\mathbf{o}})}{\partial \theta_a} Q_A[\mathbf{s}, \bar{\mathbf{o}}, \mathbf{a}] \mid \mathbf{s}_t, \bar{\mathbf{o}}_t]. \tag{11}$$

Detailed proofs of these two theorems are available in Appendix **??**. Similar to DAC [5], our gradient theorems enable a two-stage optimization scheme (Appendix **??**) for learning options, in which sample efficient MDP-based algorithms (such as PPO [14]) can be employed directly. It is worth to clarify that the derivation of our theorems is significantly different from DAC. In DAC, the random variable $\mathbf{o}$ is part of the augmented space $\mathbb{S}^H$ of the high-level MDP. On the contrary, we employ an HMM-style PGM and $\bar{\mathbf{o}}$ is a hidden variable of the *option policy* and the *action policy*.

4 The Option2Vec Architecture (O2V)

As illustrated in Section 3.2, our main contributions are representing options as embeddings ($Option^{\text{Embed}}$) and replacing the *call-and-return* mode with the *clustering mechanism*. Inspired by the *Transformer* [19], in this section we implement above mechanisms as Option2Vec, a simple yet effective Multi-Head Attention (MHA) [19] based Encoder-Decoder architecture as shown in Figure 3. Due to space limitations, we explain MHA in Appendix **??**. In Option2Vec, we implement the *option policy* $P(\bar{\mathbf{o}}_t|\mathbf{s}_t, \bar{\mathbf{o}}_{t-1}; \boldsymbol{W}_{\mathbf{o}})$ as the encoder and treat the option embedding matrix $\boldsymbol{W}_{\mathbf{o}}$ as encoder's parameters. Specifically, we define the *option encoder* as:

$$\bar{\mathbf{o}}_t \sim Categorical(Clustering(\mathbf{s}_t, \bar{\mathbf{o}}_{t-1}, \boldsymbol{W}_{\mathbf{o}})) \tag{12}$$

where $Categorical(\cdot)$ is a K-dimensional categorical distribution, K is the number of options, and distances between the pair $[\mathbf{s}_t, \hat{\mathbf{o}}_{t-1}]$ (where $\hat{\mathbf{o}}_{t-1} = \boldsymbol{W}_{\mathbf{o}}\bar{\mathbf{o}}_{t-1}$) and all clustering

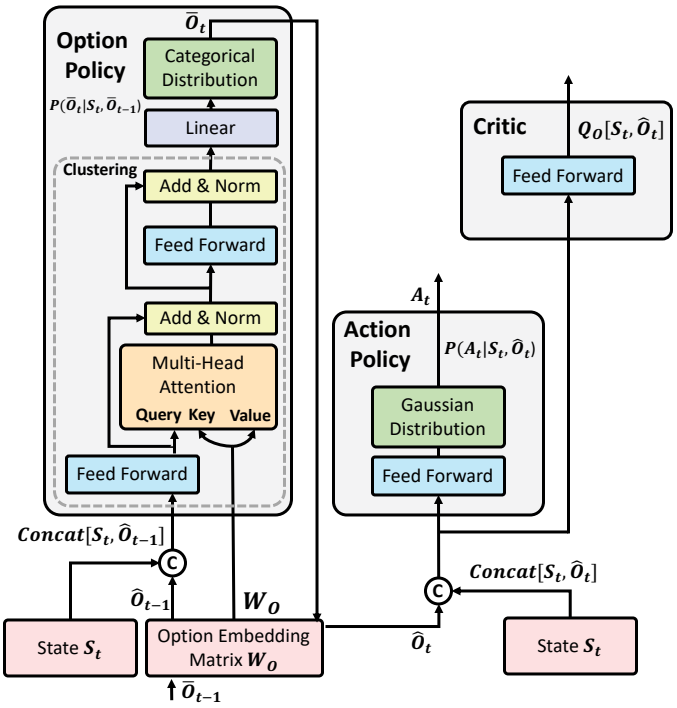


Fig. 3: The Option2Vec Architecture

centroids $\hat{\mathbf{o}}$ in embedding matrix $\boldsymbol{W}_{\mathbf{o}} = [\hat{\mathbf{o}}_1, ..., \hat{\mathbf{o}}_K]$ are measured by an efficient MHA-based *clustering module*:

$$\begin{aligned} &Clustering(\mathbf{s}_t, \bar{\mathbf{o}}_{t-1}, \boldsymbol{W}_{\mathbf{o}}) \\ &= \text{FFN}(\text{MHA}(\text{Query} = \text{FFN}([\mathbf{s}_t, \hat{\mathbf{o}}_{t-1}]), \text{Key=Value} = \boldsymbol{W}_{\mathbf{o}})) \end{aligned} \tag{13}$$

where the concatenated vector $[\mathbf{s}_t, \hat{\mathbf{o}}_{t-1} = \boldsymbol{W}_{\mathbf{o}}\bar{\mathbf{o}}_{t-1}]$ is mapped back to a point in the parametric space by employing a simple Feed-Forward Network. Option2Vec largely improves the option framework's scalability: under the $Option^{\text{Triple}}$ formulation, adding one option means adding two distributions (normally implemented as neural networks); on the contrary, in Option2Vec adding one option is as simple as adding one embedding vector $\hat{\mathbf{o}}$. Because the *clustering module* is MHA-based, the number of parameters

for the network stays unchanged with respect to the number of embeddings in $\boldsymbol{W}_{\mathbf{o}}$. With all knowledge of an option (*e.g.*, where to initiate, what actions to emit, and when to terminate) encoded as an embedding vector $Option^{\text{Embed}}$, the *action policy* can be simply implemented as one decoder, which learns to decode $\hat{\mathbf{o}}_t$ and $\mathbf{s}_t$ into primary actions $\mathbf{a}_t$.

$$\mathbf{a}_t \sim Gaussian(\text{FFN}([\mathbf{s}_t, \hat{\mathbf{o}}_t])) \tag{14}$$

which is shared by all $Option^{\text{Embed}}$. This design choice largely improves Option2Vec's sample efficiency and speed of convergence: unlike in $Option^{\text{Triple}}$ that only the activated option ω_o's *intra-option policy* gets updated, the *action policy* learns to decode $Option^{\text{Embed}}$ at every time step.

Because of the *Markovian option-value function* $\bar{V}[\mathbf{s}_{t+1}, \bar{\mathbf{o}}_t]$ is an expectation of the *option-value function* $Q_O[\mathbf{s}_{t+1}, \bar{\mathbf{o}}_{t+1}]$ in Eq. (9), we only need to model only one critic function: $Q_O = \text{FFN}(\mathbf{s}_t, \bar{\mathbf{o}}_t)$, where Q_O is also a decoder of $\mathbf{s}_t$ and $\bar{\mathbf{o}}_t$. We summarize the detailed algorithm in Appendix **??** and upload our code in supplemental materials.

5 Experiments

In this section, we design experiments to answer five questions: (Q1) Under the "one-step option" setting (Section **??**), whether $Option^{\text{Embed}}$ is a more compact and effective representation than $Option^{\text{Triple}}$? (Q2) Beyond "one-step option" setting, whether Option2Vec can achieve better performance than other option variants and non-option baselines? (Q3) Does $Option^{\text{Embed}}$ have a performance boost over other option variants in transfer learning settings? (Q4) Is $Option^{\text{Embed}}$ interpretable? (Q5) Whether Option2Vec can temporally extend options under the *clustering* mechanism?

For baselines, we follow DAC [5]'s open source implementations and compare our algorithm with six baselines, five of which are option variants, *i.e.*, DAC+PPO, AHP+PPO [9], IOPG [27], PPOC [28] and OC [25] The non-option baseline is PPO [29]. All baselines' parameters used by DAC remain unchanged other than the maximum number of training steps: Option2Vec only needs 1 million steps to converge rather than the 2 million used in DAC. For single task learning, experiments are conducted on all OpenAI Gym MuJoCo environments (10 environments) [30]. For transfer learning, we follow DAC and run 6 pairs of transfer learning tasks based on DeepMind Control Suite [31]. Figures are plotted using DAC's original script: curves are averaged over 10 independent runs and smoothed by a sliding window of size 20. Shaded regions indicate standard deviations. All experiments are run on an Intel® Core™ i9-9900X CPU @ 3.50GHz with a single thread and process. Our implementation details are summarized in Appendix **??**.

For a fair comparison, we follow DAC and use four options in all implementations. We fix the embedding dimension of $Option^{\text{Embed}}$ to 40 in all environments, *i.e.*, $\boldsymbol{W}_{\mathbf{o}} \in \mathbb{R}^{40\times 4}$. Under this configuration, Option2Vec only use 15.8% parameters in all experiments (single task and transfer learning) compared to the other option variants. Our code is available in supplemental materials.

5.1 Single-Task Learning (Q1 & Q2)

To answer Q1, we need to compare the effectiveness of $Option^{\text{Embed}}$ and $Option^{\text{Triple}}$ under the same "one-step option" setting. As explained in Section **??**, although DAC is derived under the "one-step option" setting, it still employs *termination function* and *master policy* for better performance. Therefore, we implement a "DAC-OneStep" that only employs the *option policy* (code is available in supplemental materials) and compare its performance with ours in Figure 4a. Experiments show that $Option^{\text{Embed}}$ outperforms $Option^{\text{Triple}}$ in all environments by a large margin. This is because, as explained in Section **??**, under the "one-step option" setting, the $Option^{\text{Triple}}$'s integer *option index* $\mathbf{o}$ in Eq. (1) contains little context information for the *option policy* to exploit compared to the $Option^{\text{Embed}}$. Moreover, an $Option^{\text{Embed}}$ vector only has 40 parameters to train compared to $Option^{\text{Triple}}$'s 19K parameters (in DAC each *intra-option policy* is a 3-layer FFN). Therefore, $Option^{\text{Embed}}$ is a more compact and effective representation of options compared to $Option^{\text{Triple}}$.

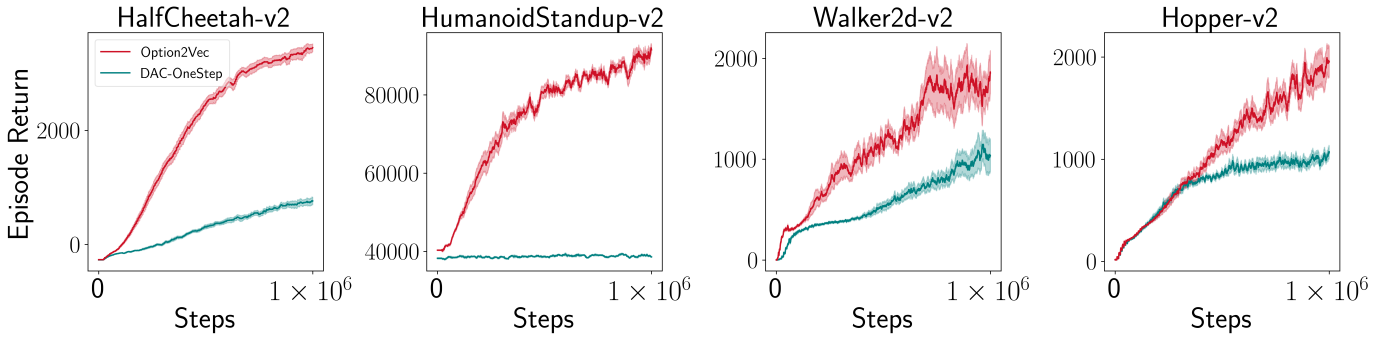
To answer Q2, we compare Option2Vec against five different option variants (*i.e.*, DAC+PPO [5], AHP+PPO [9], IOPG [27], PPOC [28] and OC [25]) and PPO [29]. It is surprising that Option2Vec shows two different performance on infinite (Figure 4b) and and finite (Figure 4c) horizon environments. Previous literatures [4, 5, 27, 28] find that option-based algorithms do not have advantages over hierarchy-free algorithms on single-task environments. Option2Vec is also able to achieve comparable performance with state of the arts (Figure 4c) but with only 15.8% parameters. More importantly, on infinite horizon environments (Figure 4b), Option2Vec's performance significantly outperforms all baselines with respect to episodic return, convergence speed, variance between steps, and variance between 10 runs (Proposition 3.3). Because theoretically infinite and finite horizon environments are identical [32], we do not have a theoretical explanation for this performance difference. In Appendix **??** we conceptually explain that this might because conventional value functions are insufficient to approximate environments in which hidden variables $\mathbf{o}$ only affect rewards but not states. In conclusion, experiment results show that by employing embeddings and the *clustering* mechanism (Section 3.2), Option2Vec is at least as effective as other option variants yet with significantly less computational cost, while has a significant advantage on infinite horizon.

5.2 Transfer Learning (Q3)

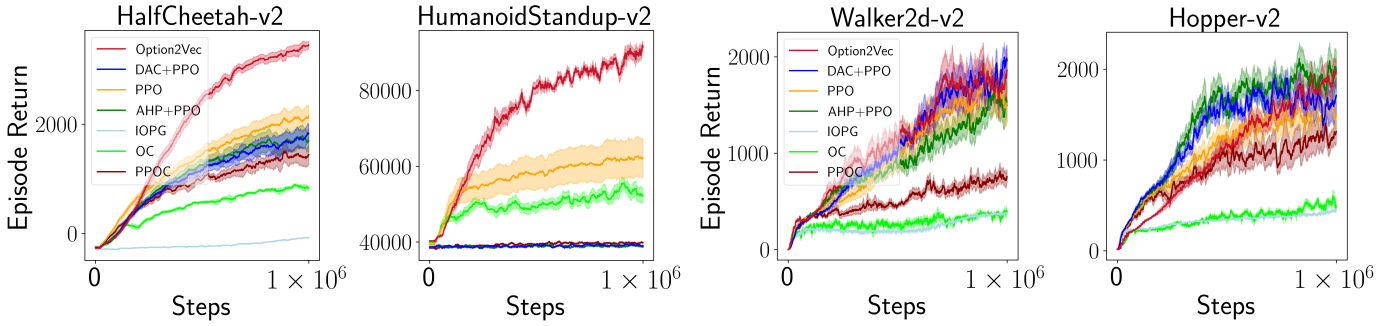
We run 6 pairs of transfer learning tasks constructed in DAC based on DeepMind Control Suite [31]. Each pair contains two different tasks. To keep consistent with DAC, we train all models one million steps on the first task and switch to the second (with $Option^{\text{Embed}}$ frozen) to run another one million steps. Results are reported in Figure 5. On the transfer learning (the second) task, Option2Vec's performance ranks the first in 5 out of 6 environments. This shows Option2Vec's advantages in knowledge reuse tasks and its performance is at least comparable with other option variants with significantly less computational cost.

5.3 Interpretation of Option Embeddings (Q4)

Interpretability is a key property (*e.g.*, ensuring safety to human, *etc.*) to apply RL agents in real-world applications.



(a) Q1: Option2Vec v.s. DAC-OneStep



(b) Q2: Infinite horizon environments

(c) Q2: Finite horizon environments

Fig. 4: Single-task episodic returns in 4 different environments (*i.e.*, HalfCheetah-v2, HumanoidStandup-v2, Walker2d-v2, and Hopper-v2). Results in all 10 environments are available in Appendix **??**.

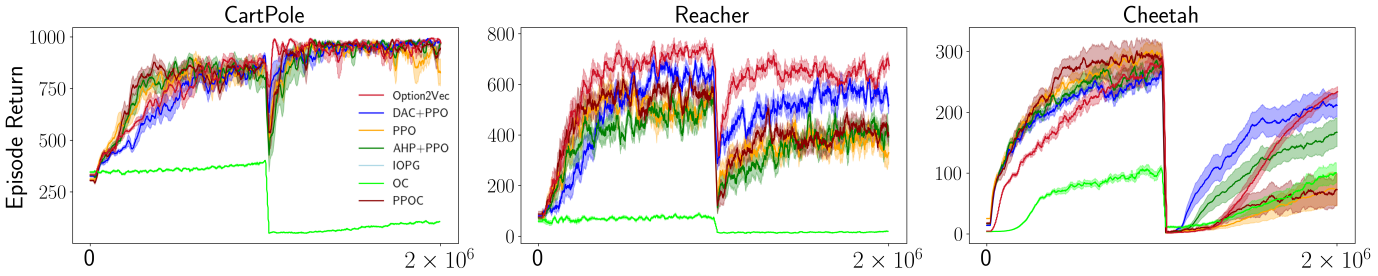


Fig. 5: Transfer learning results on three pairs of tasks. All 6 pairs of results are available in Appendix. **??**

$Option^{\text{Embed}}$ has a nature advantage over $Option^{\text{Triple}}$ on interpretability: as word embeddings [19], $Option^{\text{Embed}}$ learns a semantic space of options with each dimension encodes a particular property and can be interpreted explicitly. As in the capsule network [33], we first reason each embedding dimension's semantic by adding perturbations on to it, and inspecting perturbations' effects on primary actions $\mathbf{a}$ (Figure 6(b)). Once each dimension is understood, embeddings become straight forward to interpret by simply inspecting on which dimensions each embedding $\hat{\mathbf{o}}$ (Figure 6(a)) has significant weights, and interpreting properties of those dimensions (Figure 6(c)). Due to space limitations, more details and GIFs are provided in Appendix **??**.

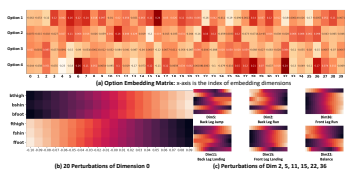


Figure 6: Interpretation of option embeddings

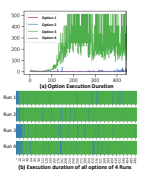


Figure 7: Option Duration Patterns

5.4 Temporal Extension (Q5)

We explained in Section 3.2 that *clustering mechanism* is able to temporally extend an option without the *termination function*. In this subsection we empirically demonstrate this in Figure 7 (more details in Appendix **??**). At the start of training, all options' durations are short, while Option 3's duration quickly grows. Although this proves that Option2Vec does temporally extend an option, it also shows that Option2 quickly dominates the whole episode. The "dominant skill problem" is not unique to Option2Vec, it is a long identified problem of the option framework [34]. However, as shown in the video[1], Option2Vec actually learns distinguishable options. Option 3 is a running forward skill thus it dominates the whole episode. Option 2 is mainly used to recover from falling down thus its duration decreases with training. In Appendix **??**, we argue that the dominant skill problem is actually a problem of learning options at multi-level granularities. Since distances between embeddings are trivial

1. https://www.youtube.com/watch?v=F26tcSsIoMA

to measure, $Option^{\text{Embed}}$ potentially provides an elegant solution to this problem. We focus this paper on proposing Option2Vec and will further explore this problem in future works.

6 Related Works

To discover options automatically, **(author?)** [1] proposed Intra-option Q-learning to update the master Q value function at every time step. However, all policies under this formulation are approximated implicitly using the Q-learning method. AHP [9] is proposed to unify the Semi-Markov process into an augmented Markov process and explicitly learn an "overall policy" by applying MDP-based policy gradient algorithms. However, their method for updating the master policy is still SMDP-style thus sample inefficient. OC [25] proposes a policy gradient based framework for explicitly learning intra-option policies and *termination function*s in an intra-option manner. However, for the master policy's policy gradients learning, OC still remains SMDP-style. DAC [5] reformulated the option framework into two augmented MDPs. Under this formulation, all policies can be modeled explicitly and learned in MDP-style. Policy gradient theorems we derive also follow DAC's efficient two-stage optimization scheme yet is significantly different from DAC as explained in Section 3.3. All above option variants employ computationally expensive $Option^{\text{Triple}}$ and *call-and-return* mode, while Option2Vec employs more simple yet efficient $Option^{\text{Embed}}$ and *clustering mechanism*. We must appreciate that **(author?)** [3] in his Ph.D. thesis (Chapter 3.5, 3.6) first conceptually discussed the possibility of introducing distributed representations into the option framework. However, to the best of our knowledge, Option2Vec is the first concrete work that enables learning options as distributed representations and deriving learning algorithms.

As for state abstractions, cognitive science experiments [35] conducted with human participants prove that human structure, generalize, and adapt past knowledge to new environments by employing both state abstraction and temporal abstraction. However, existing RL literature [36, 37, 38] only use latent variables to learn state abstractions. Typically, PEARL [39] learns a latent context vector for each task under the meta-reinforcement learning framework to improve the agent's sample and transfer learning efficiency. However, embeddings learned by RL frameworks only encode state abstraction while Option2Vec is the first option variant learns abstractions at both state and temporal dimensions.

7 Conclusions

In this paper, we proposed a compact and effective embedding representation for the option framework, $Option^{\text{Embed}}$. For learning $Option^{\text{Embed}}$, we developed a novel *Markovian Option-Value function* and derived sample efficient policy gradient theorems based on this value function. We implemented the whole mechanism as Option2Vec, a simple yet effective, transformer-like Attention-based Encoder-Decoder architecture. Empirical studies showed that $Option^{\text{Embed}}$ can significantly outperform $Option^{\text{Triple}}$ under the same "One-step option" setting in all environments. We also showed that Option2Vec achieves at least comparable performance to other option variants and non-option baselines on finite and transfer learning environments but with only 15.8% parameters, while has a significant performance boost on infinite environments. Moreover, option embedding has good interpretability, which is a key property for applying RL agents in real-world applications.

It is also worth mentioning that, the $Option^{\text{Embed}}$ establishes a direct connection to causal reinforcement learning [40, 41]. If trained under a model-based setting, $Option^{\text{Embed}}$ is a minimal causal feature set (Theorem 3 [42]) on both temporal and state dimensions. We will investigate the causal properties of Option2Vec in our future works.

8 Conclusion

The conclusion goes here.

8.1 Subsection Heading Here

Subsection text here.

8.1.1 *Subsubsection Heading Here*

Subsubsection text here.

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgments

The authors would like to thank...

References

[1] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[2] D. J. Mankowitz, T. A. Mann, and S. Mannor, "Adaptive skills, adaptive partitions (asap)," *arXiv preprint arXiv:1602.03351*, 2016.

[3] P.-L. Bacon, "Temporal representation learning," Ph.D. dissertation, McGill University Libraries, 2018.

[4] J. Harb, P.-L. Bacon, M. Klissarov, and D. Precup, "When waiting is not an option: Learning options with a deliberation cost," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[5] S. Zhang and S. Whiteson, "DAC: The double actor-critic architecture for learning options," in *Advances in Neural Information Processing Systems*, 2019, pp. 2,012–2,022.

[6] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[7] N. K. Jong, T. Hester, and P. Stone, "The utility of temporal abstraction in reinforcement learning." in *AAMAS (1)*. Citeseer, 2008, pp. 299–306.

[8] R. Givan, T. Dean, and M. Greig, "Equivalence notions and model minimization in markov decision processes," *Artificial Intelligence*, vol. 147, no. 1-2, pp. 163–223, 2003.
[9] K. Y. Levy and N. Shimkin, "Unified inter and intra options learning using policy gradient methods," in *European Workshop on Reinforcement Learning*. Springer, 2011, pp. 153–164.
[10] P. Henderson, W.-D. Chang, P.-L. Bacon, D. Meger, J. Pineau, and D. Precup, "Optiongan: Learning joint reward-policy options using generative adversarial inverse reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.
[11] A. Sharma, M. Sharma, N. Rhinehart, and K. M. Kitani, "Directed-info gail: Learning hierarchical policies from unsegmented demonstrations using directed information," *arXiv preprint arXiv:1810.01266*, 2018.
[12] T. Shankar and A. Gupta, "Learning robot skills with temporal variational inference," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8624–8633.
[13] S.-H. Lee and S.-W. Seo, "Learning compound tasks without task-specific knowledge via imitation and self-supervised learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5,747–5,756.
[14] P. Witoonchart and P. Chongstitvatana, "Application of structured support vector machine backpropagation to a convolutional neural network for human pose estimation," *Neural Networks*, vol. 92, pp. 39–46, 2017.
[15] S.-i. Amari, "Differential geometrical theory of statistics," *Amari et al. ABNK+87*, pp. 19–94, 1987.
[16] G. E. Hinton *et al.*, "Learning distributed representations of concepts," in *Proceedings of the eighth annual conference of the cognitive science society*, vol. 1. Amherst, MA, 1986, p. 12.
[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.
[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5,998–6,008.
[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[21] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.
[22] C. A. Knoblock, "Learning abstraction hierarchies for problem solving." in *AAAI Conference on Artificial Intelligence*, 1990, pp. 923–928.
[23] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3540–3549.
[24] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.
[25] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
[26] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
[27] M. Smith, H. Hoof, and J. Pineau, "An inference-based policy gradient method for learning options," in *International Conference on Machine Learning*, 2018, pp. 4,703–4,712.
[28] M. Klissarov, P.-L. Bacon, J. Harb, and D. Precup, "Learnings options end-to-end for continuous action tasks," *arXiv preprint arXiv:1712.00004*, 2017.
[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[30] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.
[31] Y. Tassa, S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, and N. Heess, "dmcontrol: Software and tasks for continuous control," 2020.
[32] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[33] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, 2017, pp. 3,856–3,866.
[34] T. Haarnoja, K. Hartikainen, P. Abbeel, and S. Levine, "Latent space policies for hierarchical reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1851–1860.
[35] L. Xia and A. G. E. Collins, "Temporal and state abstractions for efficient learning, transfer and composition in humans," *bioRxiv*, 2020.
[36] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller, "Learning an embedding space for transferable robot skills," in *International Conference on Learning Representations*, 2018.
[37] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," in *Advances in Neural Information Processing Systems*, 2017, pp. 3,812–3,822.
[38] D. Tirumala, H. Noh, A. Galashov, L. Hasenclever, A. Ahuja, G. Wayne, R. Pascanu, Y. W. Teh, and N. Heess, "Exploiting hierarchy for learning and transfer in kl-regularized rl," *arXiv preprint arXiv:1903.07438*, 2019.
[39] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *International Conference on Machine Learning*, 2019, pp. 5,331–5,340.
[40] A. Kolobov, D. S. Weld *et al.*, "Discovering hidden structure in factored mdps," *Artificial Intelligence*, vol. 189, pp. 19–47, 2012.
[41] C. F. Perez, F. P. Such, and T. Karaletsos, "Generalized hidden parameter mdps transferable model-based rl in a handful of trials," *arXiv preprint arXiv:2002.03072*, 2020.

[42] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," *arXiv preprint arXiv:2006.10742*, 2020.

PLACE PHOTO HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.