Scaling Distributed Multi-task Reinforcement Learning with Experience Sharing

Sanae Amani
samani@ucla.edu
University of California, Los Angeles

Khushbu Pahwa
khushbu16pahwa@g.ucla.edu
University of California, Los Angeles

Vladimir Braverman
vb21@rice.edu
Rice University

Lin F. Yang
linyang@ee.ucla.edu
University of California, Los Angeles

ABSTRACT

Recently, DARPA launched the ShELL program, which aims to explore how experience sharing can benefit distributed lifelong learning agents in adapting to new challenges. In this paper, we address this issue by conducting both theoretical and empirical research on distributed multi-task reinforcement learning (RL), where a group of N agents collaboratively solves M tasks without prior knowledge of their identities. We approach the problem by formulating it as linearly parameterized contextual Markov decision processes (MDPs), where each task is represented by a context that specifies the transition dynamics and rewards. To tackle this problem, we propose an algorithm called DistMT-LSVI. First, the agents identify the tasks, and then they exchange information through a central server to derive ϵ-optimal policies for the tasks. Our research demonstrates that to achieve ϵ-optimal policies for all M tasks, a single agent using DistMT-LSVI needs to run a total number of episodes that is at most $\tilde{O}(d^3H^6(\epsilon^{-2} + c_{\text{Sep}}^{-2}) \cdot M/N)$, where $c_{\text{Sep}} > 0$ is a constant representing task separability, H is the horizon of each episode, and d is the feature dimension of the dynamics and rewards. Notably, DistMT-LSVI improves the sample complexity of non-distributed settings by a factor of $1/N$, as each agent independently learns ϵ-optimal policies for all M tasks using $\tilde{O}(d^3H^6M\epsilon^{-2})$ episodes. Additionally, we provide numerical experiments conducted on OpenAI Gym Atari environments that validate our theoretical findings.

ACM Reference Format:

Sanae Amani, Khushbu Pahwa, Vladimir Braverman, and Lin F. Yang. 2023. Scaling Distributed Multi-task Reinforcement Learning with Experience Sharing. In *KDD FL4Data-Mining '23, August 7, 2023, Long Beach, CA, USA.*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD FL4Data-Mining '23, August 7, 2023, Long Beach, CA, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In recent years, there has been a growing interest in the development of distributed learning agents, which refer to multiple agents collaborating and communicating to collectively solve learning or decision-making problems with improved efficiency [5, 26]. Concurrently, the field of multi-task learning has emerged, focusing on agents that face multiple tasks and aim to learn policies that optimize performance across all tasks [10, 11, 41]. The intersection of these two research areas presents scenarios where multiple learning agents cooperate to build multi-purpose embodied intelligence, such as robots operating in weakly structured environments [29]. Motivated by these developments, the Defense Advanced Research Projects Agency (DARPA) has launched the Shared-Experience Lifelong Learning (ShELL) program. The program seeks to address how experience sharing can assist distributed lifelong learning agents in effectively adapting to new challenges [12]. This research aims to explore the potential benefits of collaborative learning approaches in enhancing the capabilities of distributed agents in dynamic environments.

In this work, we delve into the theoretical and empirical aspects of distributed multi-task reinforcement learning (RL) [22, 25]. In this setting, a group of N agents collaboratively tackles M tasks in a pure exploration manner, with the task identifications initially unknown. It is assumed that the tasks exhibit variations in rewards and transition dynamics, but share the same state and action spaces [30]. In consecutive learning rounds, the agents are assigned tasks drawn from a uniform distribution (Unif($[M]$)). The ultimate objective is for the agents to cooperate effectively, ensuring that by the end of the exploration phase, they all possess ϵ-optimal policies for all tasks, while minimizing the total number of episodes required during the exploration phase to execute a policy.

Formally, we consider an episodic setup based on the framework of contextual MDP [1, 16]. It repeats the following steps: *1)* At the beginning of each learning round, each agent receives an unknown context specifying the assigned task. *2)* Each agent initially spends certain number of episodes interacting with the corresponding task's environment, which together with communications with others through the server, help it identify the task and whether it has been solved by any other agents before. *3)* If the agent determines

that the task has already been solved, it obtains the necessary statistics from the agent who previously solved the task to determine the task's ϵ-optimal policy. Otherwise, it initiates the learning process for the ϵ-optimal policy from scratch.

The performance of each agent is evaluated based on the total number of episodes required for it to interact with the unknown environments of assigned tasks and communicate with other agents through the server, ultimately gaining access to ϵ-optimal policies for all M tasks. To maximize the benefits of the collaborative learning process, we aim for this number to scale as $\frac{M}{N\epsilon^2}$, ensuring efficient utilization of the agents' cooperative nature. Remarkably, this scaling results in a multiplicative reduction of exploration episodes by a factor of $1/N$ compared to non-distributed settings.

On the multi-task side, the closest lines of work are Abbasi-Yadkori and Neu [1], Hallak et al. [16], Kakade et al. [20], Modi et al. [27], Modi and Tewari [28] for contextual MDP and Abels et al. [4], Wu et al. [34] for the dynamic setting of multi-objective RL, which study the sample complexity for arbitrary task sequences; however, they either assume the problem is tabular with finite state and action spaces or require a model-based planning oracle with unknown complexity. Importantly, none of the existing works properly addresses the need for a distributed learning framework, which creates a large gap between the abstract setup and practice need of speeding up the learning process.

In this paper, we aim to establish a foundation for designing agents meeting these practically important requirements. As the first step, here, we study distributed multi-task RL with linear representation. We suppose that the contextual MDP is linearly parameterized [19, 36] and agents need to learn a multi-task policy based on this linear representation. However, the fact that tasks' identities are unknown to the agents make the efficient communications challenging. To overcome this hurdle, tasks must possess distinguishable features that enable agents to identify them and effectively share knowledge by communicating specific measurements related to the environments/tasks they interact with. In particular, we introduce a task-separability assumption, which is sufficient to ensure that the agents are able to distinguish between their assigned tasks throughout the learning process so that they can share information only when needed and when a task has already been solved. Under these assumptions, we propose a provably efficient distributed multi-task RL algorithm, Distributed Multi-Task Least Value Iteration (DistMT-LSVI). We show that in order to obtain ϵ-optimal policies for all M tasks, the total number of episodes a single agent needs to run DistMT-LSVI is at most $\tilde{O}(d^3H^6(\epsilon^{-2} + c_{\text{Sep}}^{-2})\cdot M/N)$, where $c_{\text{Sep}} > 0$ is a constant characterizing task separability, H is horizon of each episode and d is the feature dimension of the dynamics and rewards. Remarkably, DistMT-LSVI improves the sample complexity of non-distributed settings, where each agent separately learns all M tasks' ϵ-optimal policies using $\tilde{O}(d^3H^6M\epsilon^{-2})$ episodes, by a factor of $1/N$.

Finally, we present numerical experiments on OpenAI Gym Atari environments that corroborate our theoretical findings.

Notation. Throughout, we use lower-case letters for scalars, lower-case bold letters for vectors, and upper-case bold letters for matrices. The Euclidean norm of $\mathbf{x}$ is denoted by $\|\mathbf{x}\|_2$. We denote the transpose of any column vector $\mathbf{x}$ by $\mathbf{x}^\top$. For any vectors $\mathbf{x}$ and $\mathbf{y}$, we use $\langle \mathbf{x}, \mathbf{y}\rangle$ to denote their inner product. Let $\mathbf{A}$ be a positive semi-definite $d \times d$ matrix and $\boldsymbol{\nu} \in \mathbb{R}^d$. The weighted 2-norm of $\boldsymbol{\nu}$ with respect to $\mathbf{A}$ is defined by $\|\boldsymbol{\nu}\|_{\mathbf{A}} = \sqrt{\boldsymbol{\nu}^\top \mathbf{A}\boldsymbol{\nu}}$. For a positive integer n, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. Finally, we use standard $\tilde{O}$ notation for big-O notation that ignores logarithmic factors.

2 PROBLEM FORMULATION

Finite-horizon contextual MDP. We consider the problem of learning M tasks, each of which is modeled by an MDP. Each task $m \in [M]$ is associated with an MDP $Mc_m = (\mathcal{S}, \mathcal{A}, H, \mathbb{P}_m, r_m)$, where the state space $\mathcal{S}$, the action space $\mathcal{A}$, and the horizon H (length of each episode) are shared amongst all tasks, and $\mathbb{P}_m = \{\mathbb{P}_{m,h}\}_{h=1}^H$ are the unknown transition probabilities, and $r = \{r_{m,h}\}_{h=1}^H$ are the unknown reward functions specific to task m. For $(m, h) \in [M] \times [H]$, $r_{m,h}(s, a)$ denotes the reward function of task m at step h, whose range is assumed to be in $[0, 1]$, and $\mathbb{P}_{m,h}(s'|s, a)$ denotes the probability of transitioning to state s' upon playing action a at state s while solving task m at step h. To simplify the notation, for any function f, we write $\mathbb{P}_{m,h}[f](s, a) \coloneqq \mathbb{E}_{s'\sim\mathbb{P}_{m,h}(.|s,a)}[f(s')]$.

Policy and value functions. A policy $\pi = \{\pi_h\}_{h=1}^H$ is a sequence where $\pi_h : \mathcal{S} \to \mathcal{A}$ determines the agent's action at step h. Given π, we define state value function of task m as $V_{m,h}^\pi(s) \coloneqq \mathbb{E}[\sum_{h'=h}^H r_{m,h'}\left(s_{h'}, \pi_{h'}(s_{h'})\right)|s_h = s]$, where the expectation is with respect to policy π and transition probability $\mathbb{P}_m$. We also define its action-value function as $Q_{m,h}^\pi(s, a) \coloneqq r_{m,h}(s, a) + \mathbb{P}_{m,h}[V_{m,h+1}^\pi(.)](s, a)$, where $Q_{m,H+1}^\pi = 0$. We denote the optimal policy of task m as $\pi_{m,h}^*(s) \coloneqq \sup_\pi V_{m,h}^\pi(s)$, and let $V_{m,h}^* \coloneqq V_{m,h}^{\pi^*}$ and $Q_{m,h}^* \coloneqq Q_{m,h}^{\pi^*}$ denote the optimal value functions associated with task m. Lastly, we recall the Bellman equation of the optimal policy:

$$Q_{m,h}^*(s, a) = r_{m,h}(s, a) + \mathbb{P}_{m,h}[V_{m,h+1}^*(.)](s, a), \quad V_{m,h}^*(s) = \max_{a\in\mathcal{A}} Q_{m,h}^*(s, a). \tag{1}$$

Interaction protocol. We consider a network of N agents acting cooperatively to efficiently solve the above-stated M tasks. The learning framework consists of multiple learning *rounds*, each of which consists of multiple *episodes*. At each round t, each agent $i \in [N]$ is given a task $m_{i,t} \sim \text{Unif}([M])$, whose ID is *not* known to the agent. Agent i interacts with task $m_{i,t}$ in a number of episodes and collects trajectory $s_{1,i}^{k,t}, a_{1,i}^{k,t}, r_{1,i}^{k,t}, s_{2,i}^{k,t}, a_{2,i}^{k,t}, r_{2,i}^{k,t}, \ldots, s_{H,i}^{k,t}, a_{H,i}^{k,t}, r_{H,i}^{k,t}$ at episode k, where the initial state $s_{1,i}^{k,t}$ is a fixed initial state s_0. The agents are also allowed to communicate with each other via a central server. Both policies and the communicated information of each agent may only depend on previously observed rewards and communication received from other agents.

Linear Function Approximation. We focus on MDPs with linear transition kernels and reward functions [19, 32, 36] that are encapsulated in the following assumption.

Definition 1. $Mc = (\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ is a linear MDP with feature map $\boldsymbol{\phi} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ if for any $h \in [H]$, there exist a vector $\boldsymbol{\eta}_h$ and d measures $\boldsymbol{\mu}_h := [\mu_h{}^{(1)}, \ldots, \mu_h{}^{(d)}]^\top$ over $\mathcal{S}$ such that $\mathbb{P}_h(.|s,a) = \langle \boldsymbol{\mu}_h(.), \boldsymbol{\phi}(s,a) \rangle$ and $r_h(s,a) = \langle \boldsymbol{\eta}_h, \boldsymbol{\phi}(s,a) \rangle$, for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. Without loss of generality, $\|\boldsymbol{\phi}(s,a)\|_2 \leq 1$, $\|\boldsymbol{\mu}_h(s)\|_2 \leq \sqrt{d}$, and $\|\boldsymbol{\eta}_h\|_2 \leq \sqrt{d}$ for all $(s,a,h) \in \mathcal{S} \times \mathcal{A} \times [H]$.

Assumption 1 (Linear MDPs). *For each task $m \in [M]$, $Mc_m = (\mathcal{S}, \mathcal{A}, H, \mathbb{P}_m, r_m)$ is a linear MDP with feature map $\boldsymbol{\phi} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$.*

Here, we introduce a task-separability assumption that allows agents to distinguish and identify different tasks they are assigned throughout the learning process.

Assumption 2 (Task separability). *There exists a known positive constant $c_{\text{Sep}} > 0$, such that for any task pair $(m, m') \in [M] \times [M]$, $m \neq m'$ it holds that $\left|V^*_{m,1}(s_0) - V^*_{m',1}(s_0)\right| > c_{\text{Sep}}$.*

Goal. We say a policy π is ϵ-optimal with respect to task m if $V^{\pi}_{m,1}(s_0) \geq V^*_{m,1}(s_0) - \epsilon$. The goal is to design a cooperative and exploratory algorithm that will be run by all the agents in parallel and when it stops, all the agents have access to ϵ-optimal policies for all tasks while it minimizes the total number of episodes during the exploration phase at which a policy must be executed.

3 DISTRIBUTED MULTI-TASK LSVI

In this section, we give a description of our algorithm, Distributed Multi-Task Least Value Iteration (DistMT-LSVI), presented in Algorithm 1 for a single agent i, and is run in parallel by all N agents. This algorithm is inspired by the exploration phase for the reward-free setting in Wang et al. [32]. At the beginning of each round, after task allocation, every agent first needs to determine if its assigned task has been already solved or not. However, since the task labels are unknown, communicating task labels is not an option to figure out whether a task has been solved or not. Therefore, agents need to communicate another measure that under Assumption 2 will potentially help agents distinguish and identify tasks. In view of this, when tasks are allocated at the beginning of learning round t, each agent i first needs to spend K_1 episodes to calculate certain statistics of its corresponding task's unknown parameters (Lines 3 and 4). These statistics then will be shared with the server, who relying on Assumption 2 and checking a certain separability measure (Line 6), informs agent i whether its assigned task has been solved before, and if so, lets it know of its task's label. In such cases, agent i will screen the look-up table F, which is continuously being updated and shared with all the agents by the server to find its task's label and its corresponding ϵ-optimal policy. If the server determines that a task assigned to agent i has not been solved before, it lets agent i know that it needs to spend another K_2 episodes of exploration to learn the ϵ-optimal policy of this new task (Lines 11 and 12). Agent i then share the necessary statistics to calculate this policy with the server. At the end of round t, when the server has gathered the necessary information from all the agents, it updates two look-up tables G, which includes statistics determining task-separability and F, which includes task labels and statistics determining their corresponding ϵ-optimal policies (Line 15). Finally, agent i will receive all the changes made to F by the server.

Algorithm 1: DistMT-LSVI (δ, ϵ) for agent i

1 **Set:** $\beta_1 = c_{\beta_1} Hd\sqrt{\log(dHM\delta^{-1}c_{\text{Sep}}^{-1})}$ for some $c_{\beta_1} > 0$,
$K_1 = c_{K_1} d^3H^6 \log(dHM\delta^{-1}c_{\text{Sep}}^{-1})/c_{\text{Sep}}^2$ for some $c_{K_1} > 0$,
$\beta_2 = c_{\beta_2} Hd\sqrt{\log(dHM\delta^{-1}\epsilon^{-1})}$ for some $c_{\beta_1} > 0$,
$K_2 = c_{K_2} d^3H^6 \log(dHM\delta^{-1}c_{\text{Sep}}^{-1})/\epsilon^2$ for some $c_{K_2} > 0$,
$T = \frac{6M\log(M/\delta)}{N}$, $\ell = 0$, $G_m = F_m = \emptyset$, $\forall m \in [M]$
2 **for** rounds $t = 1, \ldots, T$ **do**
3 $\quad \mathcal{D}_i^{1,t} = \text{ExpPh}(\beta_1, K_1)$ with unknown parameters $\mathbb{P}_{m_{i,t}}$ and $r_{m_{i,t}}$
4 $\quad \left(\{(\boldsymbol{\theta}^{1,t}_{h,i}, \Lambda^{1,t}_{h,i})\}_{h\in[H]}, V^{1,t}_{1,i}(s_0)\right) = \text{Planning}(\mathcal{D}_i^{1,t})$
5 $\quad$ Send $\left(\{(\boldsymbol{\theta}^{1,t}_{h,i}, \Lambda^{1,t}_{h,i})\}_{h\in[H]}, V^{1,t}_{1,i}(s_0)\right)$ to the server
6 $\quad$ **if** $\exists m \in [\ell]$ *such that for all* $V \in G_m$, $\left|V - V^{1,t}_{1,i}(s_0)\right| \leq c_{\text{Sep}}/2$ **then**
7 $\quad\quad$ Server informs agent i of the task label $m_{i,t} = m$ Agent has already access to the ϵ-optimal policy for task $m_{i,t}$
8 $\quad\quad$ Let $\{(\boldsymbol{\theta}_h, \Lambda_h)\}_{h\in[H]} = F_m$
9 $\quad\quad$ Return policy $\pi_i^t = \{\pi^t_{h,i}\}_{h=1}^H$, where
$\pi^t_{h,i}(.) = \arg\max_{a\in\mathcal{A}} Q_h(.,a)$,
$Q_h(.,.) = \min\left\{\langle \boldsymbol{\theta}_h, \boldsymbol{\phi}(.,.)\rangle + u_h(.,.), H\right\}$,
$u_h(.,.) = \min\left\{\beta_2 \|\boldsymbol{\phi}(.,.)\|_{\Lambda_h^{-1}}, H\right\}$.
10 $\quad$ **else**
11 $\quad\quad \mathcal{D}_i^{2,t} = \text{ExpPh}(\beta_2, K_2)$ with unknown parameters $\mathbb{P}_{m_{i,t}}$ and $r_{m_{i,t}}$
12 $\quad\quad \left(\{(\boldsymbol{\theta}^{2,t}_{h,i}, \Lambda^{2,t}_{h,i})\}_{h\in[H]}, V^{2,t}_{1,i}(s_0)\right) = \text{Planning}(\mathcal{D}_i^{2,t})$
13 $\quad\quad$ Send $\{(\boldsymbol{\theta}^{2,t}_{h,i}, \Lambda^{2,t}_{h,i})\}_{h\in[H]}$ to the server
14 $\quad\quad$ Return policy $\pi_i^t = \{\pi^t_{h,i}\}_{h=1}^H$, where
$\pi^t_{h,i}(.) = \arg\max_{a\in\mathcal{A}} Q_h(.,a)$,
$Q_h(.,.) = \min\left\{\langle \boldsymbol{\theta}^{2,t}_{h,i}, \boldsymbol{\phi}(.,.)\rangle + u_h(.,.), H\right\}$,
$u_h(.,.) = \min\left\{\beta_2 \|\boldsymbol{\phi}(.,.)\|_{\left(\Lambda^{2,t}_{h,i}\right)^{-1}}, H\right\}$
15 $\quad$ Server updates $\{G_m\}_{m=1}^M, \{F_m\}_{m=1}^M, f = \text{Group}(\{G_m\}_{m=1}^M, \{F_m\}_{m=1}^M, \ell, t)$
16 $\quad$ Receive $\{F_m\}_{m=\ell+1}^{\ell+f}$ from the server.
17 $\quad \ell = \ell + f$

Theorem 1. *Let Assumptions 1 and 2 hold. Let $\delta \in (0,1)$, $\epsilon \in (0,1)$, $T = 6M\log(M/\delta)/N$, $K_1 = c_{K_1} d^3H^6 \log(dHM\delta^{-1}c_{\text{Sep}}^{-1})/c_{\text{Sep}}^2$ for sufficiently large $c_{K_1} > 0$, and $K_2 = c_{K_2} d^3H^6 \log(dHM\delta^{-1}c_{\text{Sep}}^{-1})/\epsilon^2$ for sufficiently large $c_{K_2} > 0$. Then, if all the agents run Algorithm 1 in parallel, with probability at least $1-\delta$, every agent $i \in [N]$ at every round $t \in [T]$ returns an ϵ-optimal policy π_i^t of a given task with unknown label $m_{i,t}$ and at the end of T rounds, every agent $i \in [N]$*

has access to all the tasks' ϵ-optimal policies using a total number of at most $T(K_1 + K_2)$ episodes.

3.1 Proof Sketch of Theorem 1

In this section, we give a proof sketch for Theorem 1. We start by introducing the following lemmas, which are the foundation of our analysis, and whose complete proofs are given in Appendix B. The following lemma shows that the estimated value functions are optimistic with high probability and close to the optimal value functions. It confirms that the condition in Line 6 of Algorithm 1 enable agents to identify the tasks and prevents them from solving a task that has already been solved, and an agent can directly inquire about an ϵ-optimal policy of such a task from the server.

LEMMA 1. *Under the setting of Theorem 1, for all $(i,t) \in [N] \times [T]$, with probability at least $1-\delta$, it holds that*

$$0 \leq V_{1,i}^{1,t}(s_0) - V_{m_{i,t},1}^{*}(s_0) \leq c_{\text{Sep}}/8. \tag{2}$$

Using Lemma 1, in the following lemma, we design the condition in Line 6.

LEMMA 2. *Let K_1 be chosen as in Lemma 1. Then, conditioned on the event introduced in Lemma 1, if $\left|V_{1,i}^{1,t}(s_0) - V_{1,j}^{1,t'}(s_0)\right| > c_{\text{Sep}}/2$, then $m_{i,t}$ and $m_{j,t'}$ are two different tasks; otherwise, they are the same tasks.*

LEMMA 3. *Under the setting of Theorem 1, and conditioned on the event defined in Lemma 2, for all $(i,t) \in [N] \times [T]$, with probability at least $1-\delta$, it holds that*

$$0 \leq V_{m_{i,t},1}^{*}(s_0) - V_{m_{i,t},1}^{\pi_i^t}(s_0) \leq \epsilon. \tag{3}$$

LEMMA 4. *Under the setting of Theorem 1, with probability at least $1-\delta$, after T rounds all the tasks have been solved by at least one agent.*

Conditioned on the events introduced in Lemmas 2, 3, and 4, only $T = 6M\log(M/\delta)/N$ rounds of learning are sufficient such that all the agents have access to the ϵ-optimal policies of all tasks $m \in [M]$, which proves the second part of the Theorem 1's statement.

4 EXPERIMENTS

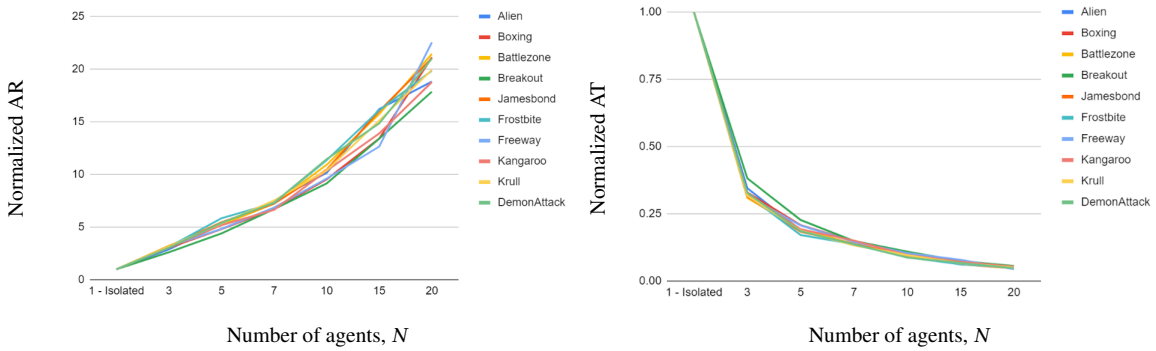
The experiments use the OpenAI Gym Atari environments for training and evaluation. Our task setting consists of $N = 20$ agents and $M = 10$ tasks. Following our proposed algorithm, DistMT-LSVI, the agents share information among each other through a central server which we also refer to as the hub. As discussed earlier, the objective is to design a cooperative algorithm so that when the algorithm stops, each agent has access to the ϵ-optimal policies for all the tasks.

In the learning stage, each agent learns a sequence of 10 randomly assigned tasks. The number of learning rounds (T) equals 10. Each agent runs Algorithm 1 in parallel. At the beginning of each round, each agent is assigned a task. The agent first needs to check if the assigned task has already been completed by other agents or not. Under Assumption 2, we achieve this by letting each agent measure task similarity by using a small neural network, which we refer to as SimNet. SimNet is a small Deep Q-Network (DQN) that consists of only two linear layers with one ReLU in between. Before an agent learns a new task, it always initializes SimNet from the same set of parameters. This allows the agent to be trained to rapidly capture patterns of the new task for 10,000 frames within one minute. Then, we use the updated parameters to compare similarity between parameters of other SimNets shared by neighboring agents on their own past tasks with distance metrics,such as Euclidean distance of parameter matrices. In this fashion, we find out the most similar tasks to our current tasks. The agent then queries features such as replay buffers of those learned tasks to train a Lifelong Learning model.

The Lifelong Learning property of the algorithm is implemented by combining an Elastic Weight Consolidation (EWC) module, which is a continual learning algorithm that effectively slows down catastrophic forgetting of past tasks during the convergence of an ongoing task by selectively retaining elasticity of important parameters, and a DQN algorithm. In particular, in the beginning of the training of a new task, the agent first obtains the experience dataset and initialization parameters for a similar task from its own experience or from other agents, then the agent continues to train its model by interacting with the environment. The existing experience will greatly reduce the training time of the new task. For each new task the agent receives, the agents are able to obtain a concise feature map of the task, and use it to query the central server, i.e., obtain the experience from the neighboring agents who have already seen the task. Each agent queries the central server that hosts the neighbors' SimNet parameters to compare the task similarity with its own SimNet, deciding which relevant task information to acquire from the server. Having obtained a list of "similar" agents to communicate with, the agent decompresses learning representations of those tasks for continual learning its model. If there is no similar task available, the LL agent trains the new task from scratch. Otherwise, it leverages learnable information from neighbors to perform LL. The LL agent frees up memory consumed by shared information. In this way, our proposed approach prevents the agents from solving a task that has already been solved, and an agent can directly inquire about an ϵ-optimal policy of a task that has already been solved by other agents from the hub.

Before learning each task, every agent encounters one of the following scenarios: 1) If neither the agent nor its neighbors has encountered a new task before, the agent needs to interact with the Atari server for at least 10000 frames for learning samples to train the Lifelong Learning model. This is equivalent to single-agent training where the agent needs to train each task from scratch. 2) If the task has been learned by its peers, our agent requests the experience replay buffers (ERB). It leverages learnable information, e.g. by performing memory replay on borrowed ERB locally instead of interacting with the Atari-server, saving large amounts of time in learning from more data in a short frame of time.

We implement our RL algorithm with a deep Q-learning (DQN) framework. In this framework, a deep neural network stores the value of a Q-function, which represents the value of a possible move given an observation. The Q-function takes input a game state (e.g.,



(a) Normalized Agent Reward **(b) Normalized Agent Time**

Figure 1: Normalized AR and AT.

an image, or an internal memory of the Atari game) and an action (e.g., a button on the game sticker), and outputs a value of the action.

To train an RL agent, we need an initialization parameter for the Q-function and an experience dataset of a given task. The dataset consists of tuples of 1) observation embeddings, 2) action, 3) reward, 4) termination status of the current state, and 5) observation embeddings of the next state e.g. RoadRunner imported from OpenAI Gym library.

The experiments include comparison of the following two baselines: (i) Isolated agent: In this baseline, a single agent performs the conventional lifelong RL to sequentially learn the Atari environments, i.e. without sharing of experience replay buffers. (ii) DistMT-LSVI Agent: In addition to performing lifelong RL on each agent, these agents also share experiences across each other, and hence learning the optimal policies faster.

In order to evaluate our experiment, we use two metrics, defined as Agent Time (AT) and Agent Reward (AR). Assume there are N agents that collaboratively learn M distinct tasks. Each agent learns a unique sequence of these M tasks. More formally, AT and AR are defined as follows: *1)* Agent Time (AT) refers to the average time of each of the N agents to achieve 90% of average normalized scores of all tasks over a permutation of sequences of M distinct tasks. This metric is proportional to the notion of sample complexity for which we provide an upper bound in Theorem 1. *2)* Agent Reward (AR) refers to the average reward of N agents after being trained for a fixed amount of time on each of the M distinct sequential tasks.

We have conducted the following experiments to evaluate the performance improvement of the DistMT-LSVI agent over the Isolated Lifelong Learning agent. The goal of this experiment is to demonstrate the increase of Agent Reward when the number of agents is increasing. We use different numbers of agents that interpolate between 1 and 20. The case of one agent corresponds to the Isolated agent case.

Figure 1a shows the difference in normalized average reward result with different numbers of agents. In normalizing the reward, we take the reward in the random setting as 0 and the isolate agent as 1. We can see that across all games, the DistMT-LSVI system achieves a linear increase across different numbers of agents in normalized Agent Reward. In general, the DistMT-LSVI agent is able to achieve a $0.88N$ scaling on the reward. In this procedure: *1)* We train the single agent (isolated Continual/Lifelong Learning) agent on a random permutation of 10 tasks. *2)* We repeat this experiment 5 times with 5 different random seeds. At the end of 10 tasks, we save the final/last model. *3)* We then use the 5 last saved models and evaluate the performance for each task for instance Alien. *4)* We finally report the results after evaluation on the best model (out of the 5 runs). *5)* We emulate the above experiment setting for DistMT-LSVI experiments (3 agents, 5 agents, 7 agents, 10 agents, and 20 agents), and report the normalized AR.

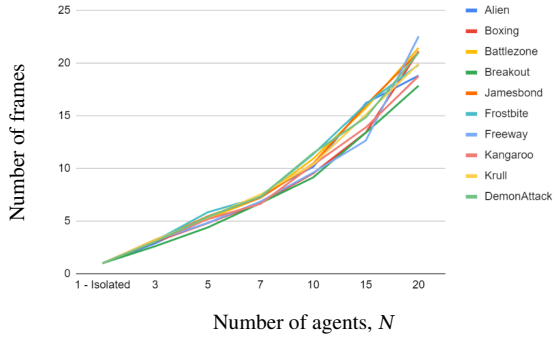


Figure 2: Number of frames required by DistMT-LSVI agents to reach x% {20, 40, 60, 80, 99} of the Single Agent reward

We also demonstrate the decrease of Agent Time when the number of agents is rising. The agent time is measured by the percentage of number of frames for the agent on average to reach the reward in the Isolated agent that performs single agent lifelong learning. We use different numbers of agents that interpolate between 1 and 20 with the following specifications: *1)* Hypothesis: increasing the number of agents will decrease the agent time to reach the desired reward on each game. *2)* Independent variable: Number of agents (1, 3, 5, 7, 10, 15, 20), Atari environments *3)* Dependent variable: Agent Time (AR) - Percentage of number of frames to reach the reward of the

isolated agent *4)* Procedure: Randomly sample 10 unique tasks from the 20 total tasks.Run training on the DistMT-LSVI system with different iterations per epoch (1, 3, 5, 7, 10, 15, 20) and train all 10 tasks.

Figure 1b shows the difference in AT with different numbers of agents. We can see that across all games, our proposed DistMT-LSVI system achieves a continuous decrease across different numbers of agents in Agent Time.

Figure 2 indicates the Agent Time (measured in terms of number of frames at the time of training). The frame rate is 60 fps (frames/second). Here we show that scaling the number of agents leads to speedup of performance. On the y-axis, we have the number of frames. On the x-axis, we have the isolated agent (Lifelong Learning agent without experience sharing), 3 agents DistMT-LSVI, 5 agents DistMT-LSVI, 7 agents DistMT-LSVI, 10 agents DistMT-LSVI, and 20 agents DistMT-LSVI. The following procedure has been adopted to achieve the above plot: *1)* We log the agent reward from the single agent (Lifelong Learning agent without experience sharing) experiment along with the number of frames. We then perform the DistMT-LSVI experiment for 3 agents, 5 agents, 7 agents, 10 agents, 15 agents, and 20 agents. *2)* In each DistMT-LSVI experiment, we measure the number of frames taken to achieve 20% , 40%, 60%, 80% and 99% of the isolated agent reward, and we plot . *3)* This plot will help us guide new design experiments to balance the trade-off between utility and computational time.

5 RELATED WORK

We consider the sample-complexity setup of distributed multi-task RL under the contextual MDP framework, where N agents receive tasks specified by unknown contexts from a pool of M tasks and they share information with each other through a server. Below, we contrast our work with related work in the literature.

Multi-task RL and Lifelong RL.. Multi-task RL studied in Brunskill and Li [8], Fifty et al. [15], Hessel et al. [17], Sodhani et al. [30], Yang et al. [37], Zhang and Wang [38] assumes that tasks are chosen from a known finite set, and in Brunskill and Li [8], Sun et al. [31], Wilson et al. [33], Yang et al. [37], tasks are sampled from a fixed distribution, and in both, task identities are assumed to be known. By contrast, our setting provides theoretical guarantees for task sequences whose identities are unknown. Another closely related line of work is lifelong RL, which studies how to learn to solve a streaming sequence of tasks. Historically many works on lifelong RL [2, 3, 6, 9, 23] assume that the tasks are i.i.d. (similar to multi-task RL). There are works for adversarial sequences, but they all assume the tasks are known when assigned to the agent and most of them are purely empirical [35]. The work by Isele et al. [18] uses contexts to enable zero-shot learning like here, but it (as well as most works above) does not provide formal regret or sample-complexity guarantees.

Distributed/Multi-agent RL.. Distributed/Multi-agent RL is a domain with a relatively long history, beginning from classical algorithms in the fully-cooperative setting [7], where all agents share identical reward functions to setting with multi-agent MDPs [21] and it has recently re-emerged due to advances in single-agent RL techniques. In the most closely related line of work, Dubey and Pentland [14], Zhang et al. [39, 40] study a more general heterogeneous reward setting, where each agent has unique rewards. However, from a multi-task learning viewpoint, in all these works, each agent is assigned to only one fixed task, i.e., $M = N$, and its goal is to learn optimal policy for only that task. In contrast, in our work, M and N are not necessarily the same, and the goal is for all N agents to achieve ϵ-optimal policies for all M tasks.

Contextual MDP and multi-objective RL.. Our setup is closely related to the exploration problem studied in the contextual MDP literature. Most contextual MDP works allow adversarial contexts, but a majority of them focuses on the tabular setup [1, 16, 24, 27, 28, 34], whereas our setup allows continuous state and action spaces. Kakade et al. [20] and Du et al. [13] allow continuous state and action spaces, but the former assumes a planning oracle with unclear computational complexity and the latter focuses on only LQG problems.

6 CONCLUSION

Motivated by DARPA's ShELL program, we conducted a comprehensive theoretical and empirical study on distributed multi-task RL. In this framework, N agents collaborate to solve a set of M tasks, without prior knowledge of the task identities. To address this challenge, we formulated the problem using linearly parameterized contextual MDPs, where each task is represented by a context that specifies its transition dynamics and rewards. To tackle this problem, we introduced a novel algorithm called DistMT-LSVI that enables agents to first identify the tasks and then leverage a central server to facilitate information sharing, ultimately obtaining ϵ-optimal policies for all M tasks. Our theoretical analysis establishes that the total number of episodes required for a single agent to execute DistMT-LSVI is bounded by $\tilde{O}(d^3H^6(\epsilon^{-2} + c_{\text{Sep}}^{-2}) \cdot M/N)$, where $c_{\text{Sep}} > 0$ is a constant characterizing task separability, H is horizon of each episode and d is the feature dimension of the dynamics and rewards. Notably, DistMT-LSVI significantly improves the sample complexity compared to non-distributed settings by a factor of $1/N$. In the non-distributed approach, each agent independently learns ϵ-optimal policies for all M tasks, necessitating $\tilde{O}(d^3H^6M\epsilon^{-2})$ episodes per agent. Our extensive numerical experiments using OpenAI Gym Atari environments provided empirical evidence supporting the efficacy and performance of our proposed methodology. That said, our work's limitations motivate further investigations in the following directions: *1)* extension to more general class of MDPs, potentially using general function approximation/representation tools, *2)* studying settings with adversarial streaming sequence of tasks.

REFERENCES

[1] Yasin Abbasi-Yadkori and Gergely Neu. 2014. Online learning in MDPs with side information. *arXiv preprint arXiv:1406.6812* (2014).

[2] David Abel, Dilip Arumugam, Lucas Lehnert, and Michael Littman. 2018. State abstractions for lifelong reinforcement learning. In *International Conference on Machine Learning*. PMLR, 10–19.

[3] David Abel, Yuu Jinnai, Sophie Yue Guo, George Konidaris, and Michael Littman. 2018. Policy and value transfer in lifelong reinforcement learning. In *International Conference on Machine Learning*. PMLR, 20–29.

[4] Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. 2019. Dynamic weights in multi-objective deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 11–20.

[5] Maryam Alavi, George M Marakas, and Youngjin Yoo. 2002. A comparative study of distributed learning environments on learning outcomes. *Information Systems Research* 13, 4 (2002), 404–415.

[6] Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, and Matthew Taylor. 2014. Online multi-task learning for policy gradient methods. In *International conference on machine learning*. PMLR, 1206–1214.

[7] Craig Boutilier. 1996. Planning, learning and coordination in multiagent decision processes. In *TARK*, Vol. 96. Citeseer, 195–210.

[8] Emma Brunskill and Lihong Li. 2013. Sample complexity of multi-task reinforcement learning. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*. 122–131.

[9] Emma Brunskill and Lihong Li. 2014. Pac-inspired option discovery in lifelong reinforcement learning. In *International conference on machine learning*. PMLR, 316–324.

[10] Rich Caruana. 1997. Multitask learning. *Machine learning* 28 (1997), 41–75.

[11] Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796* (2020).

[12] Darpa. 2021. Shared-Experience Lifelong Learning (ShELL). https://sam.gov/opp/1afbf600f2e04b26941fad352c08d1f1/view#general.

[13] Simon S Du, Ruosong Wang, Mengdi Wang, and Lin F Yang. 2019. Continuous control with contexts, provably. *arXiv preprint arXiv:1910.13614* (2019).

[14] Abhimanyu Dubey and Alex Pentland. 2021. Provably efficient cooperative multi-agent reinforcement learning with function approximation. *arXiv preprint arXiv:2103.04972* (2021).

[15] Chris Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. 2021. Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems* 34 (2021).

[16] Assaf Hallak, Dotan Di Castro, and Shie Mannor. 2015. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259* (2015).

[17] Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. 2019. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3796–3803.

[18] David Isele, Mohammad Rostami, and Eric Eaton. 2016. Using Task Features for Zero-Shot Knowledge Transfer in Lifelong Learning.. In *IJCAI*, Vol. 16. 1620–1626.

[19] Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. 2020. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*. 2137–2143.

[20] Sham Kakade, Akshay Krishnamurthy, Kendall Lowrey, Motoya Ohnishi, and Wen Sun. 2020. Information theoretic regret bounds for online nonlinear control. *Advances in Neural Information Processing Systems* 33 (2020), 15312–15325.

[21] Martin Lauer and Martin Riedmiller. 2004. Reinforcement learning for stochastic cooperative multi-agent systems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*. Citeseer, 1516–1517.

[22] Alessandro Lazaric and Mohammad Ghavamzadeh. 2010. Bayesian multi-task reinforcement learning. In *ICML-27th international conference on machine learning*. Omnipress, 599–606.

[23] Erwan Lecarpentier, David Abel, Kavosh Asadi, Yuu Jinnai, Emmanuel Rachelson, and Michael L Littman. 2021. Lipschitz Lifelong Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 8270–8278.

[24] Orin Levy and Yishay Mansour. 2022. Learning Efficiently Function Approximation for Contextual MDP. *arXiv preprint arXiv:2203.00995* (2022).

[25] Youjie Li, Iou-Jen Liu, Yifan Yuan, Deming Chen, Alexander Schwing, and Jian Huang. 2019. Accelerating distributed reinforcement learning with in-switch computing. In *Proceedings of the 46th International Symposium on Computer Architecture*. 279–291.

[26] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[27] Aditya Modi, Nan Jiang, Satinder Singh, and Ambuj Tewari. 2018. Markov decision processes with continuous side information. In *Algorithmic Learning Theory*. PMLR, 597–618.

[28] Aditya Modi and Ambuj Tewari. 2020. No-regret exploration in contextual reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*. PMLR, 829–838.

[29] Nicholas Roy, Ingmar Posner, Tim Barfoot, Philippe Beaudoin, Yoshua Bengio, Jeannette Bohg, Oliver Brock, Isabelle Depatie, Dieter Fox, Dan Koditschek, et al. 2021. From Machine Learning to Robotics: Challenges and Opportunities for Embodied Intelligence. *arXiv preprint arXiv:2110.15245* (2021).

[30] Shagun Sodhani, Amy Zhang, and Joelle Pineau. 2021. Multi-task reinforcement learning with context-based representations. In *International Conference on Machine Learning*. PMLR, 9767–9779.

[31] Yanchao Sun, Xiangyu Yin, and Furong Huang. 2021. TempLe: Learning Template of Transitions for Sample Efficient Multi-task RL. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 9765–9773.

[32] Ruosong Wang, Simon S Du, Lin Yang, and Russ R Salakhutdinov. 2020. On reward-free reinforcement learning with linear function approximation. *Advances in Neural Information Processing Systems* 33 (2020).

[33] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. 2007. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*. 1015–1022.

[34] Jingfeng Wu, Vladimir Braverman, and Lin Yang. 2021. Accommodating picky customers: Regret bound and exploration complexity for multi-objective reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021), 13112–13124.

[35] Annie Xie and Chelsea Finn. 2021. Lifelong Robotic Reinforcement Learning by Retaining Experiences. *arXiv preprint arXiv:2109.09180* (2021).

[36] Lin Yang and Mengdi Wang. 2019. Sample-optimal parametric Q-learning using linearly additive features. In *International Conference on Machine Learning*. PMLR, 6995–7004.

[37] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. 2020. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems* 33 (2020), 4767–4777.

[38] Chicheng Zhang and Zhi Wang. 2021. Provably efficient multi-task reinforcement learning with model transfer. *Advances in Neural Information Processing Systems* 34 (2021).

[39] Kaiqing Zhang, Zhuoran Yang, and Tamer Basar. 2018. Networked multi-agent reinforcement learning in continuous spaces. In *2018 IEEE conference on decision and control (CDC)*. IEEE, 2771–2776.

[40] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. 2018. Fully decentralized multi-agent reinforcement learning with networked agents. In *International Conference on Machine Learning*. PMLR, 5872–5881.

[41] Yu Zhang and Qiang Yang. 2018. An overview of multi-task learning. *National Science Review* 5, 1 (2018), 30–43.

A OMITTED ALGORITHMS

Algorithm 2: ExpPh (β, K)

Unknown parameters: $\mathbb{P}, r$
Set: $Q^k_{H+1}(.,.,.) = 0,\ \forall k \in [K]$
for $k = 1, \ldots, K$ **do**
  Observe the initial state $s^k_1 = s_0$
  for $h = H, H-1, \ldots, 1$ **do**
    $\Lambda^k_h = \mathbf{I}_d + \sum_{\tau=1}^{k-1} \boldsymbol{\phi}^\tau_h \boldsymbol{\phi}^{\tau\top}_h$
    $u^k_h(.,.) = \min\left\{\beta\left\|\boldsymbol{\phi}(.,.)\right\|_{\left(\Lambda^k_h\right)^{-1}}, H\right\}$
    Define the exploration-driven reward function $\tilde{u}^k_h(.,.) = u^k_h(.,.)/H$
    $\boldsymbol{\theta}^k_h = (\Lambda^k_h)^{-1} \sum_{\tau=1}^{k-1} \boldsymbol{\phi}^\tau_h [V^k_{h+1}(s^\tau_{h+1})]$
    $Q^k_h(.,.) = \min\left\{\langle \boldsymbol{\theta}^k_h, \boldsymbol{\phi}(.,.)\rangle + u^k_h(.,.) + \tilde{u}^k_h(.,.), H\right\}$ and $V^k_h(.) = \max_{a\in\mathcal{A}} Q^k_h(.,a)$
    $\pi^k_h(.) = \arg\max_{a\in\mathcal{A}} Q^k_h(.,a)$
  for $h = 1, 2, \ldots, H$ **do**
    Take action $a^k_h = \pi^k_h(s^k_h)$, and observe $s^k_{h+1} \sim \mathbb{P}^k_h(.|s^k_h, a^k_h)$ and $r^k_h = r_h(s^k_h, a^k_h)$
Return: $\mathcal{D} = \{(s^k_h, a^k_h, r^k_h)\}_{(h,k)\in[H]\times[K]}$

Algorithm 3: Planning $(\{(s^k_h, a^k_h, r^k_h)\}_{(h,k)\in[H]\times[K]})$

for $h = H, H-1, \ldots, 1$ **do**
  $\Lambda_h \coloneqq \mathbf{I}_d + \sum_{\tau=1}^{K} \boldsymbol{\phi}^\tau_h \boldsymbol{\phi}^{\tau\top}_h$
  $u_h(.,.) = \min\left\{\beta\left\|\boldsymbol{\phi}(.,.)\right\|_{\Lambda_h^{-1}}, H\right\}$
  $\boldsymbol{\theta}_h = (\Lambda_h)^{-1} \sum_{\tau=1}^{K} \boldsymbol{\phi}^\tau_h [r^\tau_h + V_{h+1}(s^\tau_{h+1})]$
  $Q_h(.,.) = \min\left\{\langle \boldsymbol{\theta}_h, \boldsymbol{\phi}(.,.)\rangle + u_h(.,.), H\right\}$ and $V_h(.) = \max_{a\in\mathcal{A}} Q_h(.,a)$
Return: $\left(\{(\boldsymbol{\theta}_h, \Lambda_h)\}_{h\in[H]}, V_1(s_0)\right)$

Algorithm 4: Group for the server$(\{G_m\}_{m=1}^M, \{F_m\}_{m=1}^M, \ell, t)$

Initialization: $f = 0$
for $i = 1, \ldots N$ **do**
  if $\exists m \in [\ell]$ *such that for all* $V \in G_m, \left|V - V^{1,t}_{1,i}(s_0)\right| \le c_{\text{Sep}}/2$ **then**
    Add $V^{1,t}_{1,i}(s_0)$ to G_m.
  else
    $\ell = \ell + 1, f = f + 1$
    Add $V^{1,t}_{1,i}(s_0)$ to G_ℓ and add $\left(\{(\boldsymbol{\theta}^{2,t}_{h,i}, \Lambda^{2,t}_{h,i})\}_{h\in[H]}, \ell\right)$ to F_ℓ.
Return $\{G_m\}_{m=1}^M, \{F_m\}_{m=1}^M, f$

B PROOFS OF SECTION 3

In order to prove Theorem 1, and Lemmas 1 and 3, we first state the following lemma.

LEMMA 5. *Fix $\delta \in (0,1)$ and $\epsilon \in (0,1)$ and $\beta = c_\beta Hd\sqrt{\log(dHM\delta^{-1}\epsilon^{-1})}$ for some $c_\beta > 0$, $K = c_K d^3H^6 \log(dHM\delta^{-1}\epsilon^{-1})/\epsilon^2$ for some $c_K > 0$. Let $\mathcal{D} = \{(s_h^k, a_h^k, r_h^k)\}_{(h,k)\in[H]\times[K]}$ be the dataset generated in Algorithm 2 in an environment defined with MDP $Mc = (\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ with unknown parameters $\mathbb{P}$ and r and be the input to the Algorithm 3. Let Λ_h^k, u_h^k, $\boldsymbol{\theta}_h^k$, Q_h^k, V_h^k be defined as in Algorithm 2 and Λ_h, u_h, $\boldsymbol{\theta}_h$, Q_h, V_h be defined as in Algorithm 3 and $\pi_h(.) = \arg\max_{a\in\mathcal{A}} Q_h(.,a)$. Then with probability at least $1-\delta$, it holds that*

$$V_1^*(s_0) \le V_1(s_0) \tag{4}$$

and

$$V_1(s_0) - V_1^{\pi}(s_0) \le \epsilon. \tag{5}$$

PROOF. .

First part:

We define the backups as

$$\tilde{\boldsymbol{\theta}}_h \coloneqq \boldsymbol{\eta}_h + \int_{\mathcal{S}} V_{h+1}(s')d\boldsymbol{\mu}_h(s'), \tag{6}$$

Thanks to the linear MDP structure in Assumption 1, it holds that

$$r_h(s,a) + \mathbb{P}_h\left[V_{h+1}(.)\right](s,a) = \left\langle \tilde{\boldsymbol{\theta}}_h, \boldsymbol{\phi}(s,a)\right\rangle. \tag{7}$$

$$\begin{aligned}
\tilde{\boldsymbol{\theta}}_h - \boldsymbol{\theta}_h &= \tilde{\boldsymbol{\theta}}_h - \Lambda_h^{-1}\sum_{\tau=1}^{K}\boldsymbol{\phi}_h^\tau[r_h^\tau + V_{h+1}(s_{h+1}^\tau)] \\
&= \Lambda_h^{-1}\left(\Lambda_h\tilde{\boldsymbol{\theta}}_h - \sum_{\tau=1}^{K}\boldsymbol{\phi}_h^\tau[r_h^\tau + V_{h+1}(s_{h+1}^\tau)]\right) \\
&= \underbrace{\Lambda_h^{-1}\tilde{\boldsymbol{\theta}}_h}_{\mathbf{q}_1} \underbrace{-\Lambda_h^{-1}\left(\sum_{\tau=1}^{K}\boldsymbol{\phi}_h^\tau\left(V_{h+1}(s_{h+1}^\tau) - \mathbb{P}_h[V_{h+1}(.)](s_h^\tau, a_h^\tau)\right)\right)}_{\mathbf{q}_2}.
\end{aligned}$$

Thus, in order to upper bound $\left\|\tilde{\boldsymbol{\theta}}_h - \boldsymbol{\theta}_h\right\|_{\Lambda_h}$, we bound $\|\mathbf{q}_1\|_{\Lambda_h}$ and $\|\mathbf{q}_2\|_{\Lambda_h}$ separately.

From Lemma Assumption 1, we have

$$\|\mathbf{q}_1\|_{\Lambda_h} = \left\|\tilde{\boldsymbol{\theta}}_h\right\|_{\Lambda_h^{-1}} \le \left\|\tilde{\boldsymbol{\theta}}_h\right\|_2 \le H\sqrt{d}. \tag{8}$$

Thanks to Lemma 6, for all $h \in [H]$, with probability at least $1-\delta$, it holds that

$$\|\mathbf{q}_2\|_{\Lambda_h} \le \left\|\sum_{\tau=1}^{K}\boldsymbol{\phi}_h^\tau\left(V_{h+1}(s_{h+1}^\tau) - \mathbb{P}_h[V_{h+1}(.)](s_h^\tau, a_h^\tau)\right)\right\|_{\Lambda_h^{-1}} \le c_0 Hd\sqrt{\log((c_\beta+1)dHK/\delta)}, \tag{9}$$

where c_0 and c_β are two independent absolute constants. Combining (8) and (9), for all $h \in [H]$, with probability at least $1-\delta$, it holds that

$$\left\|\boldsymbol{\theta}_h - \tilde{\boldsymbol{\theta}}_h\right\|_{\Lambda_h} \le cHd\sqrt{\log(dHK/\delta)} = \beta$$

for some absolute constant $c > 0$. Therefore, for all $h \in [H]$, with probability at least $1-\delta$, it holds that

$$\begin{aligned}
\left|r_h(s,a) + \mathbb{P}_h\left[V_{h+1}(.)\right](s,a) - \langle\boldsymbol{\theta}_h, \boldsymbol{\phi}(s,a)\rangle\right| &= \left|\langle\tilde{\boldsymbol{\theta}}_h - \boldsymbol{\theta}_h, \boldsymbol{\phi}(s,a)\rangle\right| \\
&\le \beta\left\|\boldsymbol{\phi}(s,a)\right\|_{\Lambda_h^{-1}}.
\end{aligned} \tag{10}$$

Note that conditioned on the event defined in (10) for all $(s,a) \in \mathcal{S} \times \mathcal{A}$, it holds that

$$\begin{aligned}
&\left|\left\langle \boldsymbol{\theta}_h, \boldsymbol{\phi}(s,a)\right\rangle - Q_h^{\pi}(s,a) - \mathbb{P}_h\left[V_{h+1}(.) - V_{h+1}^{\pi}(.)\right](s,a)\right| \\
&= \left|\left\langle \boldsymbol{\theta}_h, \boldsymbol{\phi}(s,a)\right\rangle - r_h(s,a) - \mathbb{P}_h\left[V_{h+1}(.)\right](s,a)\right| \\
&\leq \beta\left\|\boldsymbol{\phi}(s,a)\right\|_{\left(\Lambda_h^k\right)^{-1}}, \qquad (11)
\end{aligned}$$

for any policy π. Now, we are ready to prove the first part of the lemma by induction. The statement holds for H because $Q_{H+1}(.,.) = Q_{H+1}^*(.,.) = 0$ and thus conditioned on the event defined in (11), for all $(s,a) \in \mathcal{S} \times \mathcal{A}$, we have

$$\left|\left\langle \boldsymbol{\theta}_H, \boldsymbol{\phi}(s,a)\right\rangle - Q_H^*(s,a)\right| \leq \beta\left\|\boldsymbol{\phi}(s,a)\right\|_{\boldsymbol{\Lambda}_H^{-1}}.$$

Therefore, for all $(s,a) \in \mathcal{S} \times \mathcal{A}$, we have

$$Q_H^*(s,a) \leq \left\langle \boldsymbol{\theta}_H, \boldsymbol{\phi}(s,a)\right\rangle + \beta\left\|\boldsymbol{\phi}(s,a)\right\|_{\boldsymbol{\Lambda}_H^{-1}}.$$

Since $Q_H^*(s,a) \leq H$, we have

$$Q_H^*(s,a) \leq \min\left\{\left\langle \boldsymbol{\theta}_H, \boldsymbol{\phi}(s,a)\right\rangle + \min\left\{\beta\left\|\boldsymbol{\phi}(s,a)\right\|_{\boldsymbol{\Lambda}_H^{-1}}, H\right\}, H\right\} = Q_H(s,a),$$

which implies that

$$V_H^*(s) = \max_{a\in\mathcal{A}} Q_H^*(s,a) \leq \max_{a\in\mathcal{A}} Q_H^(s,a) = V_H(s).$$

Now, suppose the statement holds at time-step $h+1$ and consider time-step h. Conditioned on the event defined in (11), for all $(s,a) \in \mathcal{S} \times \mathcal{A}$, we have

$$\begin{aligned}
0 &\leq \left\langle \boldsymbol{\theta}_h, \boldsymbol{\phi}(s,a)\right\rangle - Q_h^*(s,a) - \mathbb{P}_h\left[V_{h+1}(.) - V_{h+1}^*(.)\right](s,a) + \beta\left\|\boldsymbol{\phi}(s,a)\right\|_{\boldsymbol{\Lambda}_h^{-1}} \\
&\left\langle \boldsymbol{\theta}_h, \boldsymbol{\phi}(s,a)\right\rangle - Q_h^*(s,a) + \beta\left\|\boldsymbol{\phi}(s,a)\right\|_{\boldsymbol{\Lambda}_h^{-1}}. \qquad \text{(Induction assumption)}
\end{aligned}$$

Since $Q_h^*(s,a) \leq H$, we have

$$Q_h^*(s,a) \leq \min\left\{\left\langle \boldsymbol{\theta}_h, \boldsymbol{\phi}(s,a)\right\rangle + \min\left\{\beta\left\|\boldsymbol{\phi}(s,a)\right\|_{\boldsymbol{\Lambda}_h^{-1}}, H\right\}, H\right\} = Q_h(s,a),$$

which means that

$$V_h^*(s) = \max_{a\in\mathcal{A}} Q_h^*(s,a) \leq \max_{a\in\mathcal{A}} Q_h^(s,a) = V_h(s).$$

This completes the proof.

Second part:

From Lemma 3.2 in [32], with probability at least $1-\delta$, it holds that

$$V_1^*(s_0, u/H) \leq c'\sqrt{d^3H^4\log(dKH/\delta)/K}, \qquad (12)$$

for some absolute constant $c' > 0$.

Note that

$$\begin{aligned}
V_1(s_0) - V_1^{\pi}(s_0) &= Q_1(s_0, \pi_1(s_0)) - Q_1^{\pi}(s_0, \pi_1(s_0)) \\
&\leq \mathbb{E}_{s_2 \sim \mathbb{P}_1(.|s_0, \pi_1(s_0))} \left[V_2(s_2) + 2u_1(s_0, \pi_1(s_0)) - V_2^{\pi}(s_2) \right] \\
&\leq E_{s_3 \sim \mathbb{P}_2(.|s_1, \pi_2(s_1))} E_{s_2 \sim \mathbb{P}_1(.|s_0, \pi_1(s_0))} \left[V_3(s_3) - V_3^{\pi}(s_3) + 2u_1(s_0, \pi_1(s_0)) + +2u_2(s_2, \pi_2(s_2)) \right] \\
&\vdots \\
&\leq V_1^{\pi}(s_0, u) \\
&\leq V_1^{*}(s_0, u) \\
&= H V_1^{*}(s_0, u/H) \\
&\leq c' \sqrt{d^3 H^6 \log(dKH/\delta)/K}. \qquad \text{(Eqn (12))}
\end{aligned}$$

Therefore, by taking $K = c_K d^3 H^6 \log(dHM\delta^{-1}\epsilon^{-1})/\epsilon^2$ for some $c_K > 0$, we have

$$V_1(s_0) - V_1^{\pi}(s_0) \leq c' \sqrt{d^3 H^6 \log(dKH/\delta)/K} \leq \epsilon, \tag{13}$$

which completes the proof. □

B.1 Proof of Lemma 1

Both inequalities in Lemma 1 can be proven using Lemma 5. Note that for every $(i, t) \in [N] \times [T]$, $V_{1,i}^{1,t}$ is computed based on output quantities of Algorithm 3 whose input is a dataset generated by interacting with task $m_{i,t}$ environment (see Lines 3 and 4 in Algorithm 1). Thus, from (4), with probability at least $1 - \delta$, it holds that

$$0 \leq V_{1,i}^{1,t}(s_0) - V_{m_{i,t},1}^{*}(s_0). \tag{14}$$

Now, let $\tilde{\pi}_i^t = \{\tilde{\pi}_{h,i}^t\}_{h=1}^H$, where $\tilde{\pi}_{h,i}^t(.) = \arg\max_{a \in \mathcal{A}} Q_h(., a)$, $Q_h(.,.) = \min\left\{\langle \boldsymbol{\theta}_{h,i}^{1,t}, \boldsymbol{\phi}(.,.)\rangle + u_h(.,.), H\right\}$, $u_h(.,.) = \min\left\{\beta_1 \left\|\boldsymbol{\phi}(.,.)\right\|_{\left(\Lambda_{h,i}^{1,t}\right)^{-1}}, H\right\}$. Therefore, from (5), with probability at least $1 - \delta$, it holds that

$$V_{1,i}^{1,t}(s_0) - V_{m_{i,t},1}^{*}(s_0) \leq V_{1,i}^{1,t}(s_0) - V_{m_{i,t},1}^{\tilde{\pi}_i^t}(s_0) \leq c_{\text{Sep}}/8, \tag{15}$$

which completes the proof.

B.2 Proof of Lemma 2

First, we prove that if $\left|V_{1,i}^{1,t}(s_0) - V_{1,j}^{1,t'}(s_0)\right| > c_{\text{Sep}}/2$, then $m_{i,t}$ and $m_{j,t'}$ are two different tasks. We have

$$\begin{aligned}
\left|V_{1,m_{i,t}}^{*}(s_0) - V_{1,m_{j,t'}}^{*}(s_0)\right| &= \left|V_{1,m_{i,t}}^{*}(s_0) - V_{1,i}^{1,t}(s_0) + V_{1,i}^{1,t}(s_0) - V_{1,j}^{1,t'}(s_0) + V_{1,j}^{1,t'}(s_0) - V_{1,m_{j,t'}}^{*}(s_0)\right| \\
&\geq \left|V_{1,i}^{1,t}(s_0) - V_{1,j}^{1,t'}(s_0)\right| - \left|V_{1,m_{i,t}}^{*}(s_0) - V_{1,i}^{1,t}(s_0) + \right| - \left|V_{1,j}^{1,t'}(s_0) - V_{1,m_{j,t'}}^{*}(s_0)\right| \qquad \text{(Triangle Inequality)} \\
&> c_{\text{Sep}}/2 - c_{\text{Sep}}/8 - c_{\text{Sep}}/8 = c_{\text{Sep}}/4 \qquad \text{(Lemma 1)}
\end{aligned}$$

which means that $V_{1,m_{i,t}}^{*}(s_0) \neq V_{1,m_{j,t'}}^{*}(s_0)$, and therefore, $m_{i,t}$ and $m_{j,t'}$ cannot be the same tasks.

Now, we prove that $\left|V_{1,i}^{1,t}(s_0) - V_{1,j}^{1,t'}(s_0)\right| \leq c_{\text{Sep}}/2$, then $m_{i,t}$ and $m_{j,t'}$ are the same tasks. We have

$$\begin{aligned}
\left|V_{1,m_{i,t}}^{*}(s_0) - V_{1,m_{j,t'}}^{*}(s_0)\right| &= \left|V_{1,m_{i,t}}^{*}(s_0) - V_{1,i}^{1,t}(s_0) + V_{1,i}^{1,t}(s_0) - V_{1,j}^{1,t'}(s_0) + V_{1,j}^{1,t'}(s_0) - V_{1,m_{j,t'}}^{*}(s_0)\right| \\
&\leq \left|V_{1,i}^{1,t}(s_0) - V_{1,j}^{1,t'}(s_0)\right| + \left|V_{1,m_{i,t}}^{*}(s_0) - V_{1,i}^{1,t}(s_0) + \right| + \left|V_{1,j}^{1,t'}(s_0) - V_{1,m_{j,t'}}^{*}(s_0)\right| \qquad \text{(Triangle Inequality)} \\
&\leq c_{\text{Sep}}/2 + c_{\text{Sep}}/8 + c_{\text{Sep}}/8 = 3c_{\text{Sep}}/4, \qquad \text{(Lemma 1)}
\end{aligned}$$

which means that $m_{i,t}$ and $m_{j,t'}$ cannot be two different tasks as $\left|V_{1,m_{i,t}}^{*}(s_0) - V_{1,m_{j,t'}}^{*}(s_0)\right|$ is not greater than c_{Sep} (See Assumption 2).

B.3 Proof of Lemma 3

Thanks to the definition of optimal policy and $V^*_{m_{i,t},1}(s_0)$, it is trivial to show the first inequality holds and $V^{\pi_i^t}_{m_{i,t},1}(s_0) \leq V^*_{m_{i,t},1}(s_0)$. To prove the second inequality, we use Lemma 5. Note that for every $(i,t) \in [N] \times [T]$, $V^{1,t}_{1,i}$ is computed based on output quantities of Algorithm 3 whose input is a dataset generated by interacting with task $m_{i,t}$ environment (see Lines 11 and 12 in Algorithm 1). Therefore, for all $(i,t) \in [N] \times [T]$, with probability at least $1-\delta$, it holds that

$$V^*_{m_{i,t},1}(s_0) - V^{\pi_i^t}_{m_{i,t},1}(s_0) \leq V^{2,t}_{1,i}(s_0) - V^{\pi_i^t}_{m_{i,t},1}(s_0) \quad \text{(Eqn. (4))}$$
$$\leq \epsilon, \quad \text{(Eqn. (5))}$$

which completes the proof.

B.4 Proof of Lemma 4

Let $\mathbb{I}^m_{i,t}$ be an indicator random variable, which is if task m is assigned to agent i at round t and 0 otherwise. Let $k_m = \sum_{i\in[N]} \sum_{t\in[T]} \mathbb{I}^m_{i,t}$ be the random variable specifying the number of times task m were assigned to an agent over the course of T rounds. Therefore, we have

$$\mu_m = \mathbb{E}[k_m] = \frac{NT}{M}. \tag{16}$$

Using multiplicative Chernoff bound, we have

$$\mathbb{P}(k_m < 1) \leq e^{\frac{-\left(1-\frac{1}{\mu_m}\right)^2 \mu_m}{2}}. \tag{17}$$

Our choice of T guarantees that

$$\mathbb{P}(\exists m \in [M], k_m < 1) \leq \delta, \tag{18}$$

which completes the proof.

C AUXILIARY LEMMAS

Notations. $\mathcal{N}_\epsilon(\mathcal{V})$ denotes the ϵ-covering number of the class $\mathcal{V}$ of functions mapping $\mathcal{S}$ to $\mathbb{R}$ with respect to the distance $\text{dist}(V, V') = \sup_s \left|V(s) - V'(s)\right|$.

LEMMA 6 (LEMMA D.4 IN JIN ET AL. [19]). *Let $\{s_\tau\}_{\tau=1}^\infty$ be a stochastic process on state space $\mathcal{S}$ with corresponding filtration $\{\mathcal{F}_\tau\}_{\tau=0}^\infty$. Let $\{\boldsymbol{\phi}_\tau\}_{\tau=0}^\infty$ be an $\mathbb{R}^d$-valued stochastic process where $\boldsymbol{\phi}_\tau \in \mathcal{F}_{\tau-1}$, and $\left\|\boldsymbol{\phi}_\tau\right\| \leq 1$. Let $\boldsymbol{\Lambda}_k = \mathbf{I}_d + \sum_{\tau=1}^{k-1} \boldsymbol{\phi}_\tau \boldsymbol{\phi}_\tau^\top$. Then with probability at least $1-\delta$, for all $k \geq 0$ and $V \in \mathcal{V}$ such that $\sup_{s\in\mathcal{S}} \left|V(s)\right| \leq H$, we have*

$$\left\| \sum_{\tau=1}^{k} \boldsymbol{\phi}_\tau \cdot \left(V(s_\tau) - \mathbb{E}\left[V(s_\tau) | \mathcal{F}_{\tau-1}\right] \right) \right\|^2_{\boldsymbol{\Lambda}_k^{-1}} \leq 4H^2 \left(\frac{d}{2} \log\left(\frac{k+\lambda}{\lambda}\right) + \log\left(\frac{\mathcal{N}_\epsilon(\mathcal{V})}{\delta}\right) \right) + \frac{8k^2\epsilon^2}{\lambda}.$$

LEMMA 7. *For any $\epsilon > 0$, the ϵ-covering number of the Euclidean ball in $\mathbb{R}^d$ with radius $R > 0$ is upper bounded by $(1 + 2R/\epsilon)^d$.*

LEMMA 8. *For a fixed w, let $\mathcal{V}$ denote a class of functions mapping from $\mathcal{S}$ to $\mathbb{R}$ with following parametric form*

$$V(.) = \min\left\{ \max_{a\in\mathcal{A}} \left\langle \mathbf{y}, \boldsymbol{\phi}(., a) \right\rangle + \beta \sqrt{\boldsymbol{\phi}(., a)^\top \mathbf{Y} \boldsymbol{\phi}(., a)}, H \right\},$$

where the parameters $\beta \in \mathbb{R}$, $\mathbf{y} \in \mathbb{R}^d$, and $\mathbf{Y} \in \mathbb{R}^{d\times d}$ satisfy $0 \leq \beta \leq B$, $\left\|\mathbf{y}\right\| \leq y$, and $\|\mathbf{Y}\| \leq \lambda^{-1}$. Assume $\left\|\boldsymbol{\phi}(s,a)\right\| \leq 1$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. Then

$$\log\left(\mathcal{N}_\epsilon(\mathcal{V})\right) \leq d \log(1 + 4y/\epsilon) + d^2 \log\left(\frac{1 + 8B^2\sqrt{d}}{\lambda\epsilon^2}\right).$$

PROOF. First, we reparametrize $\mathcal{V}$ by letting $\tilde{\mathbf{Y}} = \beta^2 \mathbf{Y}$. We have

$$V(.) = \min\left\{ \max_{a\in\mathcal{A}} \left\langle \mathbf{y}, \boldsymbol{\phi}(., a) \right\rangle + \sqrt{\boldsymbol{\phi}(., a)^\top \tilde{\mathbf{Y}} \boldsymbol{\phi}(., a)}, H \right\},$$

for $\left\|\mathrm{y}\right\| \leq y$ and $\left\|\tilde{\mathrm{Y}}\right\| \leq \frac{B^2}{\lambda}$. For any two functions $V_1, V_2 \in \mathcal{V}$ with parameters $\left(\mathrm{y}^1, \tilde{\mathrm{Y}}^1\right)$ and $\left(\mathrm{y}^2, \tilde{\mathrm{Y}}^2\right)$, respectively, we have

$$\begin{aligned}
\mathrm{dist}(V_1, V_2) \leq & \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} \left| \left[\left\langle \mathrm{y}^1, \boldsymbol{\phi}(s,a) \right\rangle + \sqrt{\boldsymbol{\phi}(s,a)^\top \tilde{\mathrm{Y}}^1 \boldsymbol{\phi}(s,a)} \right] \right. \\
& \left. - \left[\left\langle \mathrm{y}^2, \boldsymbol{\phi}(s,a) \right\rangle + \sqrt{\boldsymbol{\phi}(s,a)^\top \tilde{\mathrm{Y}}^2 \boldsymbol{\phi}(s,a)} \right] \right| \\
\leq & \sup_{\boldsymbol{\phi}:\|\boldsymbol{\phi}\|\leq 1} \left| \left[\left\langle \mathrm{y}^1, \boldsymbol{\phi} \right\rangle + \sqrt{\boldsymbol{\phi}^\top \tilde{\mathrm{Y}}^1 \boldsymbol{\phi}} \right] - \left[\left\langle \mathrm{y}^2, \boldsymbol{\phi} \right\rangle + \sqrt{\boldsymbol{\phi}^\top \tilde{\mathrm{Y}}^2 \boldsymbol{\phi}} \right] \right| \\
\leq & \sup_{\boldsymbol{\phi}:\|\boldsymbol{\phi}\|\leq 1} \left| \left\langle \mathrm{y}^1 - \mathrm{y}^2, \boldsymbol{\phi} \right\rangle \right| + \sup_{\boldsymbol{\phi}:\|\boldsymbol{\phi}\|\leq 1} \sqrt{\left| \boldsymbol{\phi}^\top \left(\tilde{\mathrm{Y}}^1 - \tilde{\mathrm{Y}}^2 \right) \boldsymbol{\phi} \right|} \qquad \left(\text{because } \left|\sqrt{a} - \sqrt{b}\right| \leq \sqrt{|a-b|} \text{ for } a, b \geq 0\right) \\
= & \sqrt{\left\| \tilde{\mathrm{Y}}^1 - \tilde{\mathrm{Y}}^2 \right\|} \\
\leq & \left\| \mathrm{y}^1 - \mathrm{y}^2 \right\| + \sqrt{\left\| \tilde{\mathrm{Y}}^1 - \tilde{\mathrm{Y}}^2 \right\|_F}. \qquad (19)
\end{aligned}$$

Let C_{y} be $\epsilon/2$-covers of $\{\mathrm{y} \in \mathbb{R}^d : \left\|\mathrm{y}\right\| \leq y\}$ with respect to the 2-norm and C_{Y} be an $\epsilon^2/4$-cover of $\{\mathrm{Y} \in \mathbb{R}^{d\times d} : \|\mathrm{Y}\|_F \leq \frac{B^2\sqrt{d}}{\lambda}\}$, with respect to the Frobenius norm. By Lemma 7, we know

$$\left|C_{\mathrm{y}}\right| \leq (1 + 4y/\epsilon)^d, \quad |C_{\mathrm{Y}}| \leq \left(\frac{1 + 8B^2\sqrt{d}}{\lambda\epsilon^2} \right)^{d^2}.$$

According to (19), it holds that $\mathcal{N}_\epsilon(\mathcal{V}) \leq \left|C_{\mathrm{y}}\right| |C_{\mathrm{Y}}|$, and therefore

$$\log\left(\mathcal{N}_\epsilon(\mathcal{V})\right) \leq d \log(1 + 4y/\epsilon) + d^2 \log\left(\frac{1 + 8B^2\sqrt{d}}{\lambda\epsilon^2} \right).$$

□