2022 American Control Conference (ACC)
Atlanta, USA, June 8-10, 2022

Representation Learning for Context-Dependent Decision-Making

Yuzhen Qin, Tommaso Menara, Samet Oymak, ShiNung Ching, and Fabio Pasqualetti

***Abstract*— Humans are capable of adjusting to changing environments flexibly and quickly. Empirical evidence has revealed that representation learning plays a crucial role in endowing humans with such a capability. Inspired by this observation, we study representation learning in the sequential decision-making scenario with contextual changes. We propose an online algorithm that is able to learn and transfer context-dependent representations and show that it significantly outperforms the existing ones that do not learn representations adaptively. As a case study, we apply our algorithm to the Wisconsin Card Sorting Task, a well-established test for the mental flexibility of humans in sequential decision-making. By comparing our algorithm with the standard Q-learning and Deep-Q learning algorithms, we demonstrate the benefits of adaptive representation learning.**

I. Introduction

Real-world decision-making is complicated since environments are often complex and rapidly changing. Yet, human beings have shown the remarkable ability to make good decisions in such environments. At the core of this ability is the flexibility to adapt their behaviors in different situations [1]. Such adaption is usually fast since humans learn to abstract experiences into compact representations that support the efficient construction of new strategies [2].

Lacking the ability to adapt to new environments and abstract compressed information from experiences, existing learning techniques often struggle in complex scenarios that undergo contextual changes. To elaborate on this point, let us consider a running example – the Wisconsin Card Sorting Task (WCST). The WCST is one of the most frequently used neuropsychological tests to assess people's ability to abstract information and shift between contexts [3]. Illustrated in Fig. 1, participants are initially given four cards and are required to associate a sequence of stimulus cards with these four cards according to some sorting rules – number, color, and shape. Participants have no prior knowledge of the current sorting rule, thus need to learn it by trial and error. They receive a feedback indicating whether their sort action is correct or incorrect. What makes the task more challenging is that the sorting rule changes every once in a while without informing the participants. Thus, the participants need to learn the changes and adjust their strategy.

Y. Qin and F. Pasqualetti are with the Department of Mechanical Engineering ({yuzhenqin,fabiopas}@engr.ucr.edu), and S. Oymak is with the Department of Electrical and Computer Engineering (oymak@ece.ucr.edu), University of California, Riverside, CA, USA. T. Menara is with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, La Jolla, CA 92093, USA. S. Ching is with the Department of Electrical and Systems Engineering and Biomedical Engineering, Washington University in St. Louis, MO, USA. This material was based upon work supported by awards ARO W911NF1910360 and NSF NCS-FO-1926829.

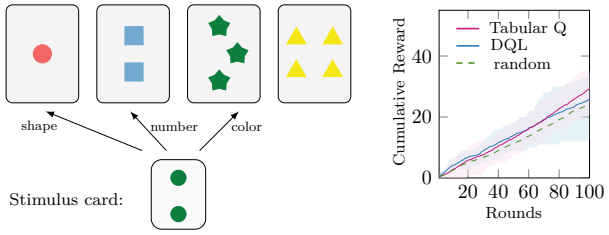

Fig. 1. The Wisconsin Card Sorting Task. Left: Illustration of the task. Participants need to sort a sequence of stimulus cards into four categories according to unknown changing rules: number, color, and shape. Right: Performance of classical reinforcement learning algorithms in this task where the sorting rule changes after every 20 rounds. Here, we consider that participants receive reward 1 for a correct sorting action and 0 otherwise Each shaded area contains 20 realizations of the corresponding algorithm.

Healthy humans usually perform very well in the WCST. Some neuroimaging studies have found that different brain regions, such as the dorsolateral prefrontal cortex and the anterior cingulate cortex, play crucial roles in context shifting, error detection, and abstraction, all of which are needed by the WCST [4]. By contrast, classical learning algorithms such as tabular-Q-learning and Deep-Q-learning struggle in the WCST, especially when the sorting rule changes rapidly. It can be seen from Fig. 1 that standard reinforcement learning (RL) algorithms[1] perform barely better than the strategy that takes random sorting actions at every round.

Motivated by these observations, we aim to develop decision-making strategies that have more human-like performance. In this paper, we focus on demonstrating the benefits of the ability to abstract compact information (i.e., learn the representation) and adapt to changing contexts in the framework of a sequential decision-making model – linear multi-armed bandits. As we will show later, the WCST can be readily modeled in this framework.

Related Work. As a classical model for decision-making, multi-armed bandits have attracted extensive interests. The Upper Confidence Bound (UCB) algorithm and its variants have proven their strength in tackling multi-armed bandit problems (e.g., see [5], [6]). Various generalizations of the classical bandit problem have been studied, in which non-stationary reward functions [7], [8], restless arms [9], satisficing reward objectives [10], risk-averse decision-makers [11], heavy-tailed reward distributions [12], and multiple players [13] are considered. Recently, increasing attention has been also paid to tackling bandit problems in a distributed fashion (e.g., see [14]–[17]).

[1]For the Deep-Q-learning, we considered a three-layer structure with 12 nodes in the hidden layers (more details can be found in Section IV). Deeper or wider networks were also tried, but similar performances were observed.

Representation learning has been applied to a wide range of practical problems including natural language processing, computer vision, and reinforcement learning [18]. Some recent studies have shown that representation learning improves data efficiency in the multi-task linear regression [19]. Representation learning has been proven to be beneficial for multi-task bandit problems, e.g., see [20]–[23]. Most of the aforementioned studies focus on batch learning where all the tasks are played simultaneously. Despite some attempts (e.g., see [24]), results on sequential bandits are sparse, although one often needs to execute tasks sequentially in real life.

Paper Contribution. In this paper, we consider a decision-making scenario with changing contexts. A multi-task decision-making model with tasks sequentially drawn from distinct sets is used to describe a dynamic environment. Our main contribution is an algorithm that is able to abstract low-dimensional representations and adapt to contextual changes. We further derive some analytical results, showing the benefits of adaptive representation learning in complex and dynamic environments. To demonstrate our theoretical findings, we apply our algorithm to the WCST and show that it significantly outperforms classical RL algorithms.

Notation. Let $\mathbb{R}$, $\mathbb{R}^+$, and $\mathbb{Z}^+$ be the sets of real numbers, positive reals, and positive integers, respectively. Given a matrix $A \in \mathbb{R}^{m\times n}$, $\text{span}(A)$ denotes its column space, $A_\perp$ denote the matrix with orthonormal columns that form the perpendicular complement of $\text{span}(A)$, $\|A\|_F$ denotes its Frobenius norm, and $[A]_i$ denotes its ith column. For any $x \in R^+$, $\lceil x \rceil$ denotes the smallest integer larger than x. Given two functions $f, g : \mathbb{R}^+ \to \mathbb{R}^+$, we write $f(x) = O(g(x))$ if there is $M_o > 0$ and $x_0 > 0$ such that $f(x) \le M_o g(x)$ for all $x \ge x_0$, and $f(x) = \tilde{O}(g(x))$ if $f(x) = O(g(x)\log^k(x))$. Also, we denote $f(x) = \Omega(g(x))$ if there is $M_\Omega > 0$ and $x_0 > 0$ such that $f(x) \ge M_\Omega g(x)$ for all $x \ge x_0$, and $f(x) = \Theta(g(x))$ if $f(x) = O(g(x))$ and $f(x) = \Omega(g(x))$.

II. Problem Setup

Motivated by real-world tasks like the WCST, we consider the following sequential decision-making model:

$$y_t = x_t^\top \theta_{\sigma(t)} + \eta_t, \tag{1}$$

where $x_t \in \mathcal{A} \subseteq \mathbb{R}^d$ is the action taken from the action set $\mathcal{A}$ at round t, and $y_t \in \mathbb{R}$ is the reward received by the agent (i.e., decision maker). The reward depends on the action in a linear way determined by the *unknown* coefficient $\theta_{\sigma(t)}$, and is also affected by the 1-sub-Gaussian noise η_t that models the uncertainty. To make good decisions, the agent needs to learn $\theta_{\sigma(t)}$ under the influence of uncertainty. This decision-making model is also known as linear bandits [25]. Note that the coefficient $\theta_{\sigma(t)}$ is time-varying, and $\sigma(t)$ is the switching signal. For simplicity, we assume that each task is played for N rounds, i.e., $\sigma(t)$ changes its value after every N rounds. Further, we assume that the agent plays S tasks in total, and denote $\mathcal{S} = \{\theta_1, \theta_2, \ldots, \theta_S\}$ as the task sequence.

To model the context changes that underlie real-world tasks like the WCST, we assume that $\theta_{\sigma(t)}$ takes values from different sets. Specifically, we assume there are m sets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$ from which $\theta_{\sigma(t)}$ takes values in sequence. In each $\mathcal{S}_k$, there are n_k (n_k can be infinite) tasks $\theta_1^k, \ldots, \ldots, \theta_{n_k}^k$, and we assume that they share a common linear feature extractor. Different sets have different feature extractors. Specifically, there is $B_k \in \mathbb{R}^{d\times r_i}$ with orthonormal vectors such that for any θ_i^k there exists $\alpha_i^k \in \mathbb{R}^{r_i}$ so that $\theta_i^k = B_k \alpha_i^k$ (see Fig. 2). For simplicity, we assume that all the extractors have the same dimension r, i.e., $r_i = r$ for all i. Here, each of these mutually different matrices $B_1, \ldots, B_m$ are also referred to as a linear *representation* [26] for the tasks in the respective set.

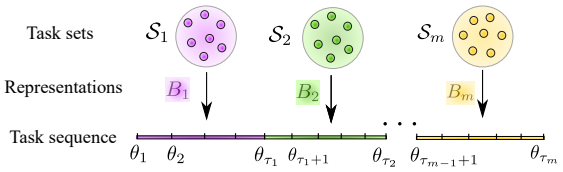

Fig. 2. Sequential decision-making scenario with contextual changes. Tasks are taken from distinct sets in sequence. The tasks in each set share a low-dimensional representation. The length of each subsequence is unknown.

As for real-world problems like the WCST, B_k describes the low-dimensional information that participants can abstract. For different contexts, participants usually need to abstract distinct low-dimensional features. Similar to the WCST in which participants do not know when the sorting rule changes, we further assume that the agent is not informed when $\theta_{\sigma(t)}$ starts to take values from a different task set. Denote $\tau_k, k = 1, \ldots, m$, as the *unknown* number of sequential tasks that $\theta_{\sigma(t)}$ takes from $\mathcal{S}_k$.

The agent's goal is to maximize the cumulative reward over the course of SN rounds. To measure the performance, we introduce the regret $R_{SN} = \sum_{t=1}^{SN}(x_t^* - x_t)^\top \theta_{s(t)}$, where x_t^* is the optimal action that maximizes the reward at round t. Given θ, denote $g(\theta) = \arg\max_{x\in\mathcal{A}} x^\top\theta$, and then $x_t^* = g(\theta_{s(t)})$. The agent's objective is then equivalent to minimizing the regret R_{SN}.

We next make some standard assumptions on the action set $\mathcal{A}$ and the task coefficients following existing studies (e.g., see [27], [28]), which are considered to be satisfied throughout the remainder of this paper.

Assumption 2.1: We assume that: (a) the action set $\mathcal{A}$ is a unit ball centered at the origin, i.e., $\mathcal{A} := \{x \in \mathbb{R}^d : \|x\| \le 1\}$, and (b) there are positive constants $\phi_{\min}$ and $\phi_{\max}$ so that $\phi_{\min} \le \|\theta_s\| \le \phi_{\max}$ for all $s \in \{1, 2, \ldots, S\}$.

Inspired by humans' strategy, we seek to equip the agent with the ability to learn and exploit representations and to quickly adjust to contextual changes so that it can perform well even in complex environments with context changes.

III. Adaptive Representation Learning

In this section, we present our main results. We first analytically demonstrate why representation learning is beneficial especially for complex tasks that have high dimensions. Second, we propose a strategy to explore and transfer the representation under the setting of sequential tasks. Finally, we present our main algorithm that has the ability to adjust to contextual changes.

A. Benefits of representation learning

To demonstrate the benefits of representation learning, we first restrict our attention to a single-task model

$$y_t = x_t^\top \theta + \eta_t, \tag{2}$$

where the task θ is played for N times. For this classical model, existing studies have established the lower bound for its regret [25], [27], presented in the next lemma.

Lemma 3.1 **(Classical Lower Bound)**: *Let $\mathcal{P}$ be the set of all policies, and $\mathcal{I}$ be the set of all the possible tasks. Then, for any $d \in \mathbb{Z}^+$ and $N > d^2$, the regret R_N for the task (2) satisfies $\inf_{\mathcal{P}} \sup_{\mathcal{I}} \mathbb{E}\, R_N = \Omega(d\sqrt{N})$.* △

This lemma indicates that there is a constant $c > 0$ such that the expected regret incurred by any policy is no less than $cd\sqrt{N}$ for any $d \in \mathbb{Z}^+$ and $N > d^2$. Next, we show how some additional information on θ affects this lower bound.

Lemma 3.2 **(Lower Bound with a Representation)**: *Suppose there is a known matrix $B \in \mathbb{R}^{d\times r}$ with $r < d$ such that $\theta = B\alpha$ for some $\alpha \in \mathbb{R}^r$. Let $\mathcal{P}$ be the set of all policies, and $\mathcal{I}$ be the set of all the possible tasks. Then, for any $d \in \mathbb{Z}^+$ and $N > d^2$, the regret R_N for the task (2) satisfies $\inf_{\mathcal{P}} \sup_{\mathcal{I}} \mathbb{E}\, R_N = \Omega(r\sqrt{N})$.* △

Proof: Let $z_t = B^\top x_t$, and then the model in (2) becomes $y_t = z_t^\top \alpha + \eta_t$. As a consequence, the problem becomes to deal with a task with dimension r instead of d. Following similar steps as in [27], it can be shown that the minimax lower bound for the regret is $\inf_{\mathcal{P}} \sup_{\mathcal{I}} \mathbb{E}\, R_N = \Omega(r\sqrt{N})$, which completes the proof. ■

Comparing Lemma 3.2 with Lemma 3.1, one finds that the regret lower bound decreases dramatically if $r \ll d$. This is because, with the knowledge of the representation $B \in \mathbb{R}^{d\times r}$, one does not need to explore the entire $\mathbb{R}^d$ space to learn the task coefficient θ for decision-making. Instead, one only needs to learn α by exploring a much lower-dimensional subspace $\mathrm{span}(B)$ and estimate θ by $\hat{\theta} = B\hat{\alpha}$. As a consequence, θ can be learned much more efficiently, which helps the agent make better decisions at earlier stages.

Yet, such a representation B is typically unknown beforehand. The agent usually needs to estimate B from its experiences before utilizing it. In the next subsection, we show how to explore and transfer the representation in the setting of sequential tasks.

B. Representation learning in sequential tasks

Representation learning in the setting of sequential tasks is challenging, particularly when the agent has no knowledge of the number of sequential tasks that share the same representation. There is a trade-off between the need to explore more tasks to construct a more accurate estimate of the underlying representation and the incentive to exploit the learned representation for more efficient learning and higher instant rewards.

To investigate how to balance the trade-off, we consider that the agent plays τ tasks in sequence, i.e., $\mathcal{T} = \{\theta_1, \theta_2, \ldots, \theta_\tau\}$, without knowing the number of tasks τ. There is an unknown matrix $B \in \mathbb{R}^{d\times r}$ such that for any i it holds that $\theta_i = B\alpha_i$ for some $\alpha_i \in \mathbb{R}^r$. In this setting we aim to find a representation learning policy for each task subsequence in Fig. 2, i.e., a within-context policy.

Algorithm 1 Representation Exploration (RE)

Input: Horizon N, exploration length $N_1 = \lceil d\sqrt{N} \rceil$
for $t = 1 : N_1$ **do** take $x_t = a_i$, $i = (t-1 \mod d) + 1$, where $[a_1, \ldots, a_d]$ is any orthonormal basis of $\mathbb{R}^d$;
compute $\hat{\theta} = (X_{\mathrm{re}} X_{\mathrm{re}}^\top)^{-1} X_{\mathrm{re}} Y_{\mathrm{re}}$, where $X_{\mathrm{re}} = [x_1, \ldots, x_{N_1}]$, $Y_{\mathrm{re}} = [y_1, \ldots, y_{N_1}]^\top$;
for $t = N_1 + 1 : N$ **do** take $x_t = \arg\max_{x\in\mathcal{A}} x^\top \hat{\theta}$

Algorithm 2 Representation Transfer (RT)

Input: Horizon N, $\hat{B} \in \mathbb{R}^{d\times r}$, exploration length $N_2 = \lceil r\sqrt{N} \rceil$
for $t = 1 : N_2$ **do** take $x_i = a'_i$, $i = (t-1 \mod r) + 1$, where $[a'_1, \ldots, a'_r]$ is any orthonormal basis of $\mathrm{span}(\hat{B})$;
compute $\hat{\alpha} = (\hat{B}^\top X_{\mathrm{rt}} X_{\mathrm{rt}}^\top \hat{B})^{-1} \hat{B}^\top X_{\mathrm{rt}} Y_{\mathrm{rt}}$ and $\hat{\theta} = \hat{B}\hat{\alpha}$, where $X_{\mathrm{rt}} = [x_1, x_2, \ldots, x_{N_2}]$ and $Y_{\mathrm{rt}} = [y_1, y_2, \ldots, y_{N_2}]^\top$;
for $t = N_2 + 1 : N$ **do** take $x_t = \arg\max_{x\in\mathcal{A}} x^\top \hat{\theta}$

We propose an algorithm, the sequential representation learning algorithm (SeqRepL, see Algorithm 3), that alternates between two sub-algorithms – representation exploration (RE) and representation transfer (RT) algorithms. Let us first elaborate on these two sub-algorithms, respectively.

1) RE algorithm. RE, shown in Algorithm 1, is an explore-then-commit (ETC) algorithm, which contains two phases: *exploration* and *commitment*, consisting of N_1 and $N - N_1$ rounds, respectively. The central goal of RE is to construct an accurate $\hat{\theta}_i$ for each task so that the collection of $\hat{\theta}_i$'s can recover an accurate representation $\hat{B}$ (which will be shown soon). Meanwhile, we want to ensure that the algorithm does not incur too much regret. To strike the balance, we set the exploration length $N_1 = \lceil d\sqrt{N} \rceil$. The exploration phase is accomplished on the entire $\mathbb{R}^d$ space, in which d linearly independent actions are repeatedly taken in sequence. Then, θ is estimated by the least-square regression $\hat{\theta} = (X_{\mathrm{re}} X_{\mathrm{re}}^\top)^{-1} X_{\mathrm{re}} Y_{\mathrm{re}}$, where $X_{\mathrm{re}} = [x_1, \ldots, x_{N_1}]$, $Y_{\mathrm{re}} = [y_1, \ldots, y_{N_1}]^\top$. In the commitment phase, the greedy action $x_t = \arg\max_{x\in\mathcal{A}} x^\top \hat{\theta}$ is taken. Note that the choice of N_1 ensures that the upper bound of RE is $O(d\sqrt{N})$, which matches the lower bound in Lemma 3.1. The proof follows similar steps as those for Theorem 3.1 in [27].

2) RT algorithm. RT, shown in Algorithm 2, is also an ETC algorithm. Its key feature is the utilization of $\hat{B}$. Thanks to $\hat{B}$, the exploration phase of RT is just carried out in the r-dimensional subspace $\mathrm{span}(\hat{B})$. Consequently, much fewer exploration rounds are required ($N_2 = \lceil r\sqrt{N} \rceil$ rather than $\lceil d\sqrt{N} \rceil$). In the exploration phase, r linear independently actions in $\mathrm{span}(\hat{B})$ are repeatedly taken before the N_2 rounds are exhausted. Unlike RE wherein $\hat{\theta}$ is directly constructed, RT first estimates α by the least-square regression $\hat{\alpha} = (\hat{B}^\top X_{\mathrm{rt}} X_{\mathrm{rt}}^\top \hat{B})^{-1} \hat{B}^\top X_{\mathrm{rt}} Y_{\mathrm{rt}}$ with $X_{\mathrm{rt}} = [x_1, x_2, \ldots, x_{N_2}]$ and $Y_{\mathrm{rt}} = [y_1, y_2, \ldots, y_{N_2}]^\top$, and then recovers θ by $\hat{B}\hat{\alpha}$.

With a perfect estimate $\hat{B} = B$, RT can achieve a regret upper bounded by $O(r\sqrt{N})$. This can be proven straightforwardly since the original model can be rewritten into a r-dimensional one $y_t = z_t^\top \alpha + \eta_t$ by letting $z_t = \hat{B}^\top x_t$.

Yet, constructing a perfect $\hat{B}$ is usually impossible given the noisy environment. The next theorem provides an upper bound for the regret of RT when there is some error between $\hat{B}$ and B.

Theorem 3.3 **(Upper Bound Given an Estimated Representation)**: *Assume that an estimate $\hat{B}$ of the true representation B satisfies $\|\hat{B}^\top B_\perp\|_F \le \varepsilon$. If the agent plays the task described by Eq. (2) for N rounds using Algorithm 2 with $\hat{B}$, then the regret satisfies $\mathbb{E}\, R_N = O(r\sqrt{N} + N\varepsilon^2)$.* △

The upper bound in Theorem 3.3 is less than the lower bound $\Omega(d\sqrt{N})$ in Lemma 3.1 if $\varepsilon < \sqrt{d}/N^{\frac{1}{4}}$. This implies that the knowledge of an imperfect estimate of the representation improves the performance as long as it is sufficiently accurate (i.e., small $\|\hat{B}^\top B_\perp\|_F$).

Proof of Theorem 3.3: Since $\theta = B\alpha$, then the model becomes $y_t = x_t^\top B\alpha + \eta_t$. From Algorithm 2, it holds that $\hat{\alpha} = (\hat{B}^\top X_{rt} X_{rt}^\top \hat{B})^{-1} \hat{B}^\top X_{rt} Y_{rt}$. Without loss of generality, we assume N_2 is a multiple of r. Then, it can be calculated that $X_{rt}X_{rt}^\top = \frac{N_2}{r} AA^\top$ with $A = [a_1', \dots, a_r']$, then we have $\hat{\alpha} = \big(N_2 \hat{B}^\top AA^\top \hat{B}/r\big)^{-1} \hat{B}^\top X_{rt} Y_{rt}$. As $Y_{rt} = X_{rt}^\top B\alpha + \eta$ with $\eta = [\eta_1, \dots, \eta_{N_2}]^\top$, we have $\hat{\alpha} = \big(N_2 \hat{B}^\top AA^\top \hat{B}/r\big)^{-1} N_2 \hat{B}^\top AA^\top B\alpha/r + \big(\frac{N_2}{r} \hat{B}^\top AA^\top \hat{B}\big)^{-1} \hat{B}^\top X_{rt}\eta$. As $\hat{\theta} = \hat{B}\hat{\alpha}$ and $\theta = B\alpha$, it follows that

$$\hat{B}\hat{\alpha} - B\alpha = \underbrace{\hat{B}\big(N_2 \hat{B}^\top AA^\top \hat{B}/r\big)^{-1} N_2 \hat{B}^\top AA^\top B\alpha/r - B\alpha}_{s_1} + \underbrace{\hat{B}\big(N_2 \hat{B}^\top AA^\top \hat{B}/r\big)^{-1} \hat{B}^\top X\eta}_{s_2}.$$

Then, it holds that $\mathbb{E}\left[\|\hat{\theta}(c) - \theta\|^2\right] \le \mathbb{E}\,\|s_1\|^2 + \mathbb{E}\,\|s_2\|^2$ since η_t is an independent random variable with zero mean. It can be derived (more details can be found in the extended version of this paper [29]) that $\mathbb{E}\,\|s_1\|^2 \le 2c\phi_{\max}^2 \varepsilon^2$ for some constant c and $\mathbb{E}\,\|s_2\|^2 \le r/\sqrt{N}$. Combining $\mathbb{E}\,\|s_1\|^2$ and $\mathbb{E}\,\|s_2\|^2$, we have $\mathbb{E}\left[\|\hat{\theta} - \theta\|^2\right] \le r/\sqrt{N} + 2c\phi_{\max}^2\varepsilon^2$.

From [27], it follows that

$$\mathbb{E}[\max_{x\in\mathcal{A}} x^\top\theta - \max_{x\in\mathcal{A}} x^\top\hat{\theta}] \le J\frac{\hat{r}}{\phi_{\min}\sqrt{N}} + 2\frac{1}{\phi_{\min}} J\phi_{\max}^2(1+\mu)\varepsilon^2. \quad (3)$$

For the commitment phase, there are $N - N_2$ steps. Thus, the overall regret satisfies $\mathbb{E}\, R_N \le N_2\phi_{\max} + (N - N_2)\left(\max_{x\in\mathcal{A}} x^\top\theta - \mathbb{E}\max_{x\in\mathcal{A}} x^\top\hat{\theta}\right)$. Substituting Eq. (3) into the right-hand side we obtain $\mathbb{E}\, R_N \le O(\hat{r}\sqrt{N} + N\varepsilon^2)$, which completes the proof. ■

3) SeqRepL algorithm. Let us now present the main algorithm in this subsection, which performs sequential representation learning (SeqRepL). It operates in a cyclic manner, alternating between RE and RT (see Algorithm 3). In each cycle, there are two phases. In the RE phase of the nth cycle, L tasks are played using RE. Then, the representation are estimated. Specifically, let $\hat{P} = \sum \hat{\theta}_i\hat{\theta}_i^\top$, where $\hat{\theta}_i$'s are the learned coefficients in all the previous n RE phases. Then,

Algorithm 3 Sequential Representation Learning (SeqRepL)

Input: $\mathcal{S}_\tau = \{\theta_1, \dots, \theta_\tau\}$, $L = c_1 r$, $\hat{P} = 0_{d\times d}$ **Initialize**: $n = 1$;
for each cycle n:
  RE phase: play L tasks in $\mathcal{S}_\tau$ using RE algorithm, $\hat{P} = \hat{P} + \hat{\theta}_i\hat{\theta}_i^\top$, $\hat{B} \leftarrow$ top r singular vector of $\hat{P}$;
  RT phase: play nL tasks in $\mathcal{S}_\tau$ using RT algorithm with latest $\hat{B}$; update $n = n + 1$.

$\hat{B}$ is constructed by performing *singular value decomposition* (SVD) to $\hat{P}$ in the following way:

$$\text{SVD}: \hat{P} = [U_1, U_2]\Sigma V \quad \longrightarrow \quad \hat{B} = U_1,$$

where the columns of $U_1 \in \mathbb{R}^{d\times r}$ are the singular vectors that are associated with the r-largest singular values of $\hat{P}$. In the RT phase, nL tasks are played using RT with the estimated $\hat{B}$. Notice that L more tasks are played using RT in each cycle than the previous one. This alternating scheme balances representation exploration and transfer well.

Next, we make an assumption and provide an upper bound for SeqRepL.

Assumption 3.4: For the task sequence $\mathcal{T} = \{\theta_1, \dots, \theta_\tau\}$, suppose that there exists $L = c_1 r$ for some constant $c_1 > 0$ such that any subsequence of length L in $\mathcal{T}$ satisfies $\sigma_r(W_s W_s^\top) \ge \nu > 0$ for any s, where $W_s = [\theta_{s+1}, \dots, \theta_{s+\ell}]$ and $\sigma_r(\cdot)$ denote the rth largest singular value of a matrix. △

This assumption states that the sequential tasks covers all the directions of the r-dimensional subspace $\text{span}(B)$, which ensures that B can be recovered in a sequential fashion.

Theorem 3.5 **(Upper Bound of SeqRepL)**: *Let the agent play a series of tasks $\{\theta_1, \theta_2, \dots, \theta_\tau\}$ using SeqRepL in Algorithm 3, where $\tau > r^2$. Suppose that Assumption 3.4 is satisfied, then the regret, denoted by $R_{\tau N}$, satisfies $\mathbb{E}\, R_{\tau N} = \tilde{O}\left(d\sqrt{\tau r N} + \tau r\sqrt{N}\right)$.* △

Note that if one uses a standard algorithm, e.g., a UCB algorithm [25] or a PEGE algorithm [27], to play the sequence of tasks without learning the representation, the optimal regret would be $\Theta(\tau d\sqrt{N})$. This bound is always larger than the two terms in our bound since $\tau > r^2$. This indicates that our algorithm outperforms the standard algorithms that do not learn the representations.

Proof of Theorem 3.5: After the RE phase of nth cycle in the SeqRepL algorithm, it can be derived (more details can be found in the extended version of this paper [29]) that the estimate $\hat{B}$ and the true representation B satisfy $\|\hat{B}^\top B_\perp\|_F = \tilde{O}\left(\frac{d}{\nu}\sqrt{\frac{1}{nLd\sqrt{N}}}\right)$. The regret incurred in this phase of the nth cycle, denoted by $R_{\text{RE}}(n)$, satisfies $R_{\text{RE}} = O(Ld\sqrt{N})$ Then, nL tasks are played in sequence utilizing the RT algorithm with input $\hat{B}$. It follows from Lemma 3.3 that the regret in the RT phase of the nth cycle, denoted as $R_{\text{RT}}(n)$, satisfies $\mathbb{E}\, R_{\text{RT}}(n) \lesssim nLr\sqrt{N} + nLN\frac{d^2}{\nu^2}\frac{1}{nLd\sqrt{N}} = \tilde{O}(nLr\sqrt{N} + d\sqrt{N})$. Observe that there are at most $\bar{L} = \lceil\sqrt{2\tau/L}\rceil$ cycles in the sequence of length τ since $L\bar{L} + L\bar{L}(\bar{L}+1)/2 \ge \tau$. Summing up the regret in Phases 1 and 2 in every cycle, we

Algorithm 4 Outlier Detection (OD)

Input: $\hat{B} \in \mathbb{R}^{d\times r}$, n_{od}, generate a random orthonormal matrix $Q \in \mathbb{R}^{(d-r)\times n_{\text{od}}}$, and let $M = \hat{B}_\perp Q$.
for $t = 1, \ldots, n_{\text{od}}$ **do** $x_t = \delta[M]_t \in \mathcal{A}$, collect y_t **end for**
if $Y_{n_{\text{od}}} \notin \mathcal{C}_{n_{\text{od}}}$ **then** outlier indicator $\mathbb{I}_{\text{od}} = 1$ **end if**

Algorithm 5 Adaptive Representation Learning (AdaRepL)

Input: k_c **Initialize:** $n_c = 0$ (outlier counter), $\hat{B} = I_d$
for $\theta_1, \theta_2, \ldots, \theta_S$ **do:**
invoke OD algorithm, return $\mathbb{I}_{\text{od}}$
if $\mathbb{I}_{\text{od}} = 1$ **do** invoke RE algorithm, $P = P + \hat{\theta}_i \theta_i^\top$, $n_c = n_c + 1$
else $n_c = 0$, invoke the cyclic SeqRepL
end if
if $n_c = k_c$ **do** restart SeqRepL **end if**

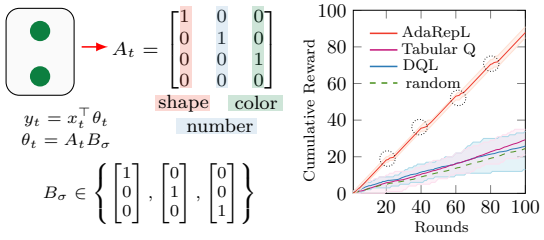

Fig. 3. Left: key steps to model WCST into a sequential decision-making model with linear reward functions. Right: performance comparison between our algorithm and standard RL algorithms in WCST. Sorting rules change every 20 rounds. Dotted circles indicate that our algorithm is able to adapt to new contexts and learn new representations quickly.

obtain $\mathbb{E}\, R_{N\tau} \lesssim \bar{L}Ld\sqrt{N} + \sum_{m=1}^{\bar{L}} \left(nLr\sqrt{N} + d\sqrt{N}\right) \leq \bar{L}Ld\sqrt{N} + \tau r\sqrt{N} + \bar{L}d\sqrt{N}$. Since $L = c_1 r$ for some constant c_1 and $\bar{L} = \lceil\sqrt{2\tau/L}\rceil$, then $\mathbb{E}\, R_{\tau N} = \sum_{n=1}^{\bar{L}} R_{\text{RE}}(n) + R_{\text{RT}}(n) = \tilde{O}\left(d\sqrt{\tau r N} + \tau r_i\sqrt{N} + d\sqrt{\tau N/r}\right)$, which completes the proof. ■

C. Representation learning with contextual changes

Finally, we are ready to address the problem that we set up in Section II, i.e., representation learning in sequential tasks with changing contexts.

In the WCST, humans are able to realize of sorting rule changes quickly. Inspired by that, we equip our algorithm with the ability to detect context switches, which enables it to adapt to new environments.

As shown in Algorithm 4, the key idea is to take n_{od} probing actions for every new task. These actions are randomly generated in the perpendicular complement of $\text{span}(\hat{B})$. Specifically, we generate a random orthonormal matrix $Q \in \mathbb{R}^{(d-r)\times n_{\text{od}}}$. The probing actions are taken from the columns of the matrix $M = \delta\hat{B}_\perp Q$, where $\delta > 0$ ensures that the actions are within the action set $\mathcal{A}$. If the current task θ satisfies $\theta = \hat{B}\alpha$ for some α, it holds that $y_t = x_t^\top\theta + \eta_t = \eta_t$ since $Q^\top \hat{B}_\perp^\top \hat{B}\alpha = 0$. Therefore, if the received rewards considerably deviate from the level of noise, the new task is an outlier to the current context (i.e., a task that does not lie in the subspace $\text{span}(B)$) with high probability.

Let $Y_{n_{\text{od}}} = [y_1, \ldots, y_{n_{\text{rsd}}}]^\top$ collect the rewards. Also, we build a confidence interval for $Y_{n_{\text{od}}}$, which is $\mathcal{C}_{n_{\text{od}}} = \left\{Y_{\text{od}} \in \mathbb{R}^{n_{\text{od}}} : \left|\|Y\|_2 - \sqrt{n_{\text{od}}}\right| \leq \xi_{\text{od}}\right\}$, where ξ_{od} is the detection threshold chosen by the agent. If the observed Y_{od} is beyond $\mathcal{C}_{n_{\text{od}}}$, we decide that the new task is an outlier.

The main algorithm in this paper, which we call *Adaptive Representation Learning algorithm* (AdaRepL), is provided in Algorithm 5, which invokes both SeqRepL and OD sub-algorithms. The former well balances representation exploration and transfer in the sequential setting, and the latter enables the algorithm to adapt to changing environments. To make our algorithm robust to occasional outliers, we set a threshold k_c so that the algorithm considers that a context switch has occurred only when k_c outliers have been detected consecutively.

It is worth mentioning that with the aid of the OD algorithm, the agent can detect context changes with high probability by properly selecting the detection threshold ξ_{od} and the length of probing actions n_{od}. Within each context, the regret of AdaRepL has an upper bound presented in Theorem 3.5. Although context change detection incurs some regret, the overall performance will still surpass the standard algorithms that are unable to learn representations adaptively. We will verify this point in the next section by revisiting the WCST.

IV. Experimental Study of WCST

First, we provide more details on the tabular-Q learning and Deep-Q learning algorithms in Fig. 1. We assume that the agent receives reward 1 if it takes the classification action x_t satisfies $x_t = \theta_t$, otherwise, it receives reward 0.

For the tabular-Q learning, the problem is to construct the $4^3 \times 4$ Q table. This is because there are 4^3 possible stimulus cards (4 colors, 4 numbers, 4 shapes) and each stimulus card can be taken as a state, and there are 4 sorting actions.

For the Deep-Q learning, we formalize each input state by a 3-dimension vector $(\text{shape}, \text{number}, \text{color})^\top \in \{1, 2, 3, 4\}^3$. The result shown in Fig. 1 is based on a three-layer network with 3, 12, and 4 nodes in the input, hidden, and output layers, respectively. We also considered deeper or wider structures but obtained similar performances.

It can be observed from Fig. 1 that these two algorithms struggle in the WCST. The reason is that a large number of samples (certainly more than 4^4 samples) are needed to construct the Q table or train the network weights. However, if the sorting rule changes much earlier than 4^4 rounds, it is impossible to find the optimal policy. Also, being unaware of the sorting rule changes worsens the performance.

Next, we demonstrate how our proposed algorithm, which explore and exploit the representation in the WCST and detect sorting rule changes, has a much better performance.

To do that, we model the WCST into a sequential decision-making model. Specifically, we use a matrix $A_t \in \mathbb{R}^{4\times 3}$ to describe the stimulus card at round t. The first, second, and third columns of A_t represent shape, number, and color, respectively, and they take values from the set $\{e_1, e_2, e_3, e_4\}$ with e_i being the ith standard basis of $\mathbb{R}^4$. In each column, e_i indicates that this card has the same shape/number/color

as the ith card on table (see Fig. 3). For example, the stimulus card (with two green circles) in Fig. 1 can be represented by the matrix $A = [e_1, e_2, e_3]$ (see Fig. 3). Moreover, we use a standard unit vector B_σ, which takes values from $\{b_1, b_2, b_3\}$ with b_i being the standard basis of $\mathbb{R}^3$, to respectively describe the 3 sorting rules – shape, number, and color. In addition, the action x_t also takes value from the set $\{e_1, e_2, e_3, e_4\}$. The action $x_t = e_i$ means to sort the stimulus card to the ith card on table.

Consequently, the WCST can be described by the sequential decision-making model $y_t = x_t^\top \theta_t$ with $\theta_t = A_t B_\sigma$. Here the unit vector B_σ can be taken as the current *representation* since the correct sorting action can always be computed by $x_t^* = A_t B_\sigma$ no matter what card the agent sees. For instance, suppose the rule is number (i.e., $B_\sigma = b_2$), if the agent sees the stimulus card with two green circles, i.e., $A = [e_1, e_2, e_3]$, then correct sort is the second card on table since it can be computed that $x_t^* = A_t B_\sigma = [0, 1, 0, 0]^\top$.

The problem then reduces to learn the underlying representation B_σ, a task that is much easier than constructing the Q table or training the weights in a Deep-Q network. Remarkably, one does not even need to learn individual θ_t to construct B_σ. Instead, B_σ can be recovered by $B_\sigma = (\sum_{t=1}^k A_t^\top x_t x_t^\top A_t)^{-1} \sum_{t=1}^k A_t^\top x_t y_t$ immediately after $\sum_{t=1}^k A_t^\top x_t x_t^\top A_t$ becomes invertible. This indicates that our idea in this paper can apply to more general situations.

It can be observed in Fig. 3 that our algorithm significantly outperforms the other two, which demonstrates the power of being able to abstract compact representations and adapt to new environments.

V. Concluding Remarks

In this paper, we have studied representation learning for decision-making in environments with contextual changes. To describe such context-changing environments, we employ a decision-making model in which tasks are drawn from distinct sets sequentially. Inspired by strategies taken by humans, we propose an online algorithm that is able to learn and transfer representations under the sequential setting and has the ability to adapt to changing contexts. Some analytical results have been obtained, showing that our algorithm outperforms existing ones that are not able to learn representations. We also apply our algorithm to a real-world task (WCST) and verify the benefits of the ability to learn representations flexibly and adaptively. We are interested in studying representation learning in more general RL frameworks such as Markovian or non-Markovian processes.

References

[1] A. Radulescu, Y. S. Shin, and Y. Niv, "Human representation learning," *Annual Review of Neuroscience*, vol. 44, no. 1, pp. 253–273, 2021.

[2] N. T. Franklin and M. J. Frank, "Generalizing to generalize: humans flexibly switch between compositional and conjunctive structures during reinforcement learning," *PLoS Computational Biology*, vol. 16, no. 4, p. e1007720, 2020.

[3] B. R. Buchsbaum, S. Greer, W.-L. Chang, and K. F. Berman, "Meta-analysis of neuroimaging studies of the wisconsin card-sorting task and component processes," *Human Brain Mapping*, vol. 25, no. 1, pp. 35–45, 2005.

[4] C.-H. Lie, K. Specht, J. C. Marshall, and G. R. Fink, "Using fMRI to decompose the neural processes underlying the Wisconsin Card Sorting Test," *Neuroimage*, vol. 30, no. 3, pp. 1038–1049, 2006.

[5] P. Auer, "Using confidence bounds for exploitation-exploration tradeoffs," *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 397–422, 2002.

[6] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári, "Improved algorithms for linear stochastic bandits," in *Advances in Neural Information Processing Systems*, vol. 11, 2011, pp. 2312–2320.

[7] Y. Russac, C. Vernade, and O. Cappé, "Weighted linear bandits for non-stationary environments," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[8] L. Wei and V. Srivastava, "Nonstationary stochastic multiarmed bandits: UCB policies and minimax regret," *arXiv preprint arXiv:2101.08980*, 2021.

[9] T. Gafni and K. Cohen, "Learning in restless multiarmed bandits via adaptive arm sequencing rules," *IEEE Transactions on Automatic Control*, vol. 66, no. 10, pp. 5029–5036, 2021.

[10] P. Reverdy, V. Srivastava, and N. E. Leonard, "Satisficing in multi-armed bandit problems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3788–3803, 2016.

[11] M. Malekipirbazari and O. Cavus, "Risk-averse allocation indices for multi-armed bandit problem," *IEEE Transactions on Automatic Control*, 2021, in Press.

[12] L. Wei and V. Srivastava, "Minimax policy for heavy-tailed bandits," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1423–1428, 2020.

[13] M. K. Hanawal and S. Darak, "Multi-player bandits: A trekking approach," *IEEE Transactions on Automatic Control*, 2021, in Press.

[14] D. Kalathil, N. Nayyar, and R. Jain, "Decentralized learning for multiplayer multiarmed bandits," *IEEE Transactions on Information Theory*, vol. 60, no. 4, pp. 2331–2345, 2014.

[15] P. Landgren, V. Srivastava, and N. E. Leonard, "Distributed cooperative decision making in multi-agent multi-armed bandits," *Automatica*, vol. 125, p. 109445, 2021.

[16] U. Madhushani and N. E. Leonard, "A dynamic observation strategy for multi-agent multi-armed bandit problem," in *2020 European Control Conf.*, 2020, pp. 1677–1682.

[17] J. Zhu and J. Liu, "A distributed algorithm for multi-armed bandit with homogeneous rewards over directed graphs," in *American Control Conference*, 2021, pp. 3038–3043.

[18] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[19] N. Tripuraneni, C. Jin, and M. I. Jordan, "Provable meta-learning of linear representations," *arXiv preprint arXiv:2002.11684*, 2020.

[20] S. Lale, K. Azizzadenesheli, A. Anandkumar, and B. Hassibi, "Stochastic linear bandits with hidden low rank structure," *arXiv preprint arXiv:1901.09490*, 2019.

[21] K.-S. Jun, R. Willett, S. Wright, and R. Nowak, "Bilinear bandits with low-rank structure," in *International Conference on Machine Learning*, 2019, pp. 3163–3172.

[22] Y. Lu, A. Meisami, and A. Tewari, "Low-rank generalized linear bandit problems," *arXiv preprint arXiv:2006.02948*, 2020.

[23] J. Yang, W. Hu, J. D. Lee, and S. S. Du, "Impact of representation learning in linear bandits," in *International Conference on Learning Representations*, 2021.

[24] M. G. Azar, A. Lazaric, and E. Brunskill, "Sequential transfer in multi-armed bandit with finite set of models," in *Advances in Neural Information Processing Systems*, 2013, p. 2220–2228.

[25] V. Dani, T. P. Hayes, and S. M. Kakade, "Stochastic linear optimization under bandit feedback," 2008.

[26] J. Hu, X. Chen, C. Jin, L. Li, and L. Wang, "Near-optimal representation learning for linear bandits and linear RL," *arXiv preprint arXiv:2102.04132*, 2021.

[27] P. Rusmevichientong and J. N. Tsitsiklis, "Linearly parameterized bandits," *Mathematics of Operations Research*, vol. 35, no. 2, pp. 395–411, 2010.

[28] Y. Li, Y. Wang, X. Chen, and Y. Zhou, "Tight regret bounds for infinite-armed linear contextual bandits," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 370–378.

[29] Y. Qin, T. Menara, S. Oymak, S. Ching, and F. Pasqualetti, "Non-stationary representation learning in sequential linear bandits," *arXiv preprint arXiv:2201.04805*, 2022.