Contextual Deep Reinforcement Learning with Adaptive Value-based Clustering

1st Yuhe Gao	2nd Runzhe Wan	3rd Ioannis Giannakakis	4th Jinxiang Gu	5th Rui Song†
Amazon	*Amazon*	*Amazon*	*Amazon*	*Amazon*
Seattle, USA	Jersey City, USA	Seattle, USA	Seattle, USA	Seattle, USA
gaoyuhe@amazon.com	wrunzhe@amazon.com	ioannisg@amazon.com	gjinxian@amazon.com	ruisong@amazon.com

***Abstract*—Applications of reinforcement learning (RL) in real-world scenarios are often limited by its generalizability across multiple different environments. Contextual RL offers a principled solution to this issue by capturing environmental heterogeneity through observable contextual variables. However, directly applying Contextual RL may not achieve optimal results when contexts exhibit high randomness and variance, and model complexity is constrained by computational resources. In this paper, we introduce a novel approach that automatically clusters contextual environments and learns customized policies for each cluster. Our algorithm leverages embedded contexts derived from the hidden layers of the value function of a pretrained RL agent, ensuring that environments within each cluster share similar transition kernels and reward functions. This general meta-framework can be applied with any RL algorithm with value functions. Empirical results from our simulations demonstrate that the composite policy, formed by aggregating contextual RL policies from each cluster, significantly outperforms a single baseline policy trained on all contexts.**

***Index Terms*—Deep Reinforcement Learning, Contextual Reinforcement Learning, Clustering, Embedding**

I. Introduction

Reinforcement learning (RL) has found various applications in robotics, health care and gaming. However, RL method lacks generalization when applied to real life cases, as one RL policy is trained for a specific task and cannot be well transferred to other tasks. Such challenge motivates contextual reinforcement learning (contextual RL [1]), where similar tasks are assumed to differ in the observable contextual variables. In a robot locomotion task, the context variables can include friction of the floor, mass of the object, location of the goal, etc. Different context determines the specific hyper-parameter values of transition kernel and reward functions of different tasks, while the function class of transition kernel and reward functions across the tasks are the same. A contextual RL policy is then trained to handle these task. Therefore, rather than training multiple RL policy for each specific task, contextual RL policy is a universal one trained on all the tasks. It can also generalize to unobserved tasks with a different context values than the tasks in training data.

However, there are two challenges in training contextual RL policy. First, the contexts can be highly diverse in practice. For instance, in a set of locomotion tasks, the targeting location of the objects and floor conditions can vary greatly in different application cases. High variance of contexts causes large noises in the rewards, which hampers the policy training. Moreover, the true model of context in value function and polices of RL agent can be complex, which can be difficult to learn with limited computing time. To handle the challenges above, we propose to divide contexts into clusters, as different tasks may display some inherent cluster structure, such that the tasks with context belonging to the same cluster share very similar state-transition and action-reward mechanisms to each other. Rather than training a uniform policy for all different tasks, we train different local policies on each cluster of contexts. In this sense, the noises of rewards are reduced as contexts are less variant in local clusters. The complicated model of contexts can be approximated locally when training on each cluster. Furthermore, directly clustering on contexts may not yield a reasonable cluster structure, as Euclidean distance of contexts may not accurately reflect the similarity among RL environments. We would like to use the hidden layers of value function networks in RL algorithms as guidance to embed contexts and perform clustering on the embedded contexts. We propose an algorithm that can cluster contexts in a hierarchical way, which resembles a top-down (divisive) hierarchical clustering procedure.

Our contributions are three-fold: We first demonstrate that local contextual RL policies trained on clusters of contexts outperform the contextual RL policy trained on the entire set of contexts, as training local policies avoids the large reward noises caused by diverse contexts and approximates the potentially complex models of contexts locally. Secondly, we propose a clustering method of the embedded contexts based on the value-functions during training of contextual RL policy. The simulation studies demonstrate the advantages of our method on multiple experiments in Gym environments. Finally, our theoretical results show the clustering results of embedded contexts from the value functions converge to the optimal clusters.

II. Related Works

The concept of contextual Markov Decision Process (cMDP) and contextual RL are first introduced in [1] to reflect real world scenarios such as web advertising with

† Corresponding author. Email:ruisong@amazon.com; Address: Department of Statistics Campus Box 8203 North Carolina State University Raleigh, NC 27695-8203; Telephone: 919-515-1955.

multiple users, where the environmental heterogeneity can be summarized as observable contextual variables that enable efficient training and generalization. Following this work, many contextual RL methods have been proposed: [2] and [3] study the optimal training curriculum in contextual RL, i.e., training on environments from the easier ones to more difficult ones; [4] studies representation learning for contextual RL; [5] directly models the relationship between the context and the transition/reward function, but due to the exponential increase of the dimension, the method is restrictive to settings where the MDP is tabular with each entry being a linear/GLM function of the context. Our research on clustering method, aimed at the optimal middle ground for both low variability and high generality, is orthogonal and complementary to these papers.

There is a huge volume of literature in the clustering algorithms, which can be generally divided into hierarchical algorithms (which includes bottom-up methods and top-down methods) and partitioning methods (which includes K-means, K-medoids and density based methods such as DBSCAN [6]). Reviews of clustering methods can be found in [7]. In particular, [8] combines top-down clustering and K-means clustering together, which is similar to our method of hierarchically clustering contexts. However, none of these methods is applied in a contextual RL setting.

There are several works that apply clustering methods in the settings of reinforcement learning or bandits. [9] divides the space of *initial state* into clusters to handle the high variance of estimated gradient caused by the diverse initial states. In their setting, different Markov Decision Processes (MDPs) differ by their initial state distributions rather than reward function or transition kernels. [10] and [11] incorporates clustering methods in an online recommendation setting. [10] adapts a multi-arm bandit algorithm and uses DBSCAN clustering on the features of items to recommend. The purpose of clustering in [10] is to reduce the number of items to be considered for recommendation, such that the recommendation candidates for each user will be a cluster of items rather than the entire item space. In [11], a clustering method for user information is proposed, and the distance of embedded user information to the cluster centroids are used to determine exploration and exploitation in an online RL setting. [12] applies clustering in a parcel delivery systems. In [12], the clustering procedure and RL policy training are two separate steps, where clustering assigns parcels to couriers and the followed policy trained by Q-learning assignes couriers to different regions in the city. Note that all the works reviewed above is in an RL or bandit setting, instead of contextual RL.

III. Backgrounds

A. Contextual Markov Decision Process

Traditional reinforcement learning formulates an ongoing decision process as a Markov Decision Process (MDP) such that one policy will be trained on this MDP to achieve the best utility. One MDP is defined as a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ containing state space, action space, transition function and reward function, respectively. In the setting of contextual RL, multiple MDPs are handled by a single policy to significantly reduce the training time. The information of state transition and reward function that specifies one MDP is called *context* (denoted as c), and the collection of MDP is defined as contextual Markov decision process (cMDP) $\mathcal{M}_{\mathcal{C}} = \{\mathcal{M}_c\}_{c\in\mathcal{C}}, \mathcal{M}_c = \{\mathcal{S}, \mathcal{A}, \mathcal{P}_c, \mathcal{R}_c\}$ for some context space $\mathcal{C}$ [1]. In the MDP collection $\{\mathcal{M}_c\}_{c\in\mathcal{C}}$, the state space $\mathcal{S}$ and action space $\mathcal{A}$ are the same across all the MDPs, but the state transition function and reward function are determined by the context c of each MDP. Therefore, cMDP represents a collection of different tasks with various environment dynamics sharing the same state and action space. In this study, we denote the set of context in training as $\{c_1, c_2, ..., c_N\}$, where N is the number of MDPs.

B. Online Policy Training and Evaluating

For the policy π of the contextual RL model, it maps the input state and context to an action at a decision time point t such that $A_t \sim \pi(.|S_t, C_t)$. We build a simulator to simulate the cMDP in Section III-A and train the policy π by interacting with the simulator along the planning horizon T. In particular, in each epoch of training, an MDP from the collection cMDP will be randomly selected to interact with the policy being trained. For evaluation, in each repetition, an MDP will be randomly sampled from the collections of cMDP and the policy will make decisions along this MDP for T episodes. The mean and standard errors of the cumulative reward from N_{mc} repetitions will be used to evaluate the corresponding policy.

This framework complies with many popular RL training algorithms, such as value-based algorithms (e.g. [13], [14]) and policy-based approaches (e.g. [15], [16]). We currently use the Proximal Policy Optimization (PPO) algorithm [17] as the default RL training method. This algorithm requires two neural networks to model the policy and corresponding value function respectively. For general contextual RL methods, the state and context are concatenated together and input to the policy/value network. In this sense, context is treated as part of the state in the policy or value model. We denote this network architecture as $\mathcal{F}_0$, and an illustration of it is shown in Figure 1. The policy obtained is denoted as π_{base}.

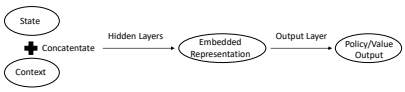

Fig. 1. Illustration of architecture $\mathcal{F}_0$ of policy/ value network for training the policy π_{base}.

IV. Proposed Method

When the tasks specified by different contexts are substantially diverse, the performance of the policy π_{base} may suffer, as learning the contextual semantics remains a significant challenge in contextual RL. Assuming there exists a general model $f(c)$ to handle the entire set of context $\mathcal{C}$, the complexity

of this model can pose a significant challenge for learning, especially with limited computational resources. Alternatively, we propose to divide $\mathcal{C}$ into K clusters, $\{G_1, G_2, \cdots, G_K\}$, such that each cluster contains MDPs that are similar to each other. We then train a local policy π_k on each cluster G_k. This approach allows us to learn a local approximation of the general model $f(c)$ of context within each cluster, which is much easier to learn than the global model. Moreover, training local policies on clusters has the advantage of reducing the noise of rewards caused by different contexts. Let Y be the stochastic cumulative reward of MDP. Then we have

$$Var(Y) = E_c[Var(Y|c)] + Var_c[E(Y|c)]. \quad (1)$$

When the context in $\mathcal{C}$ is highly variable, the second term in (1) can be large, resulting in significant noise in the cumulative reward during training on the entire $\mathcal{C}$. However, if we train a policy on each cluster such that $\mathbb{E}(Y|c)$ is similar for all contexts in the cluster, the variance of the cumulative reward will be smaller, leading to more efficient training. Thus, we expect the performance of the composite policy $\{\pi_k\}_{k=1}^K$ will significantly outperform π_{base}.

A. Clustering on Embedded Contexts

For the clustering method, if we adapt the same method in [9], then we can apply the K-means method to directly cluster on $\mathcal{C}$. However, this method may not effectively differentiate between various tasks because the geometric distance metrics on $\mathcal{C}$ may not reflect the differences across the MDPs. For instance, in a simplified setting where $\mathcal{C} \subset \mathbb{R}^2$ and $c = [c^{(1)}, c^{(2)}]^\top$, suppose a small amount of change in the second component $c^{(2)}$ significantly affects the reward function and state transition in the environment, while $c^{(1)}$ has almost no effect on them. In particular, suppose context c only affects the immediate reward such that $r(s, a; c) = \psi_a(s)(0.01c^{(1)} + c^{(2)}) + \epsilon$ for some random noise ϵ and s, a, c denotes state, action and context respectively. Then for $c_1 = [-5, 4]^\top, c_2 = [-5, 5]^\top, c_3 = [5, 4]^\top, c_4 = [5, 5]^\top$, K-means tends to assign c_1, c_2 into a cluster and c_3, c_4 into another cluster based on their Euclidean distance. However, c_1 and c_3 correspond to similar tasks as their second components are the same, so is c_2 and c_4, as is shown in Figure 2. This suggests that the reasonable way to cluster the contexts is not on the original space $\mathcal{C}$, but an embedded space of $\mathcal{C}$, $\{\phi(c)\}$ for some embedding function ϕ that can reflect the significance of different components in transition dynamics and reward. Unlike the simplified setting in Figure 2, ϕ can be a complex non-linear function in practice. To obtain such embedding, we propose to use the embedded context in the value function learned during training of contextual RL. In particular, we assume the value function of policy trained in contextual RL follow a generalized linear model for state s, such that

$$V^\pi(s, c) = \phi_s^\top(s)\phi_c(c). \quad (2)$$

This generalized linear model for s is motivated by similar settings in Section 3.1 of [18]. $\phi_s(\cdot), \phi_c(\cdot)$ are embedding

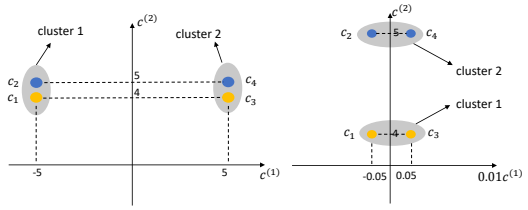

Fig. 2. Illustration of clustering. **Left:** cluster on original space of $(c^{(1)}, c^{(2)})$; **Right:** cluster on a scaled space of $(0.01 \times c^{(1)}, c^{(2)})$, such that two components of context are weighted according to their coefficients in the reward function, $r(s, a; c) = \psi_a(s)(0.01c^{(1)} + c^{(2)}) + \epsilon$.

maps of state and context, respectively. $\phi(c)$ would then be the embedded context used for clustering.

Such clustering method applies to any RL algorithms that includes a neural network for the value function. To be specific, in the value function model, we will separately embed state s and context c through hidden layers of neural networks and obtain the embedded $\phi(c)$. This value function architecture is motivated by [19], where the value function that separately embed state and context can be well generalized to the environments with unseen contexts during training. This architecture for value function neural network is shown in Figure 3. The function g in Figure 3 can be a cross product for the value function model, which represents the generalized linear relationship between contexts and states in the value function (as is shown in (2)) . This architecture for the value function will be denoted as $\mathcal{F}_1$.

To obtain $\phi(c)$, we can first train an contextual RL policy using some RL training algorithm Ψ (say, the PPO algorithm [17]) with value functions following architecture $\mathcal{F}_1$ on the entire set of cMDP $\mathcal{M}_\mathcal{C}$ with contexts $\{c_1, c_2, ..., c_N\}$. Denote the entire set of N MDPs as a single cluster $G_1^{(0)}$, such that the superscript 0 is the iteration index of our algorithm (set to 0 for initial input) and the subscript is the index of this cluster among all clusters (set to 1 for the single cluster case). We further denote the policy trained on this entire set of MDPs as $\pi_1^{(0)}$. Then we can obtain $\phi(c)$ using the neural network model of the value function from $\pi_1^{(0)}$. We will then perform clustering on $\phi(c)$ with K-means method to obtain K clusters $\{G_1^{(1)}, G_2^{(1)}, \cdots, G_K^{(1)}\}$. Then we can obtain local policies $\{\pi_1^{(1)}, \pi_2^{(1)}, \cdots, \pi_K^{(1)}\}$ by training RL policies using the RL algorithm Ψ on the K clusters correspondingly. Moreover, clustering for one iteration may not yield a desired grouping of different MDPs. Therefore, we can iteratively divide each group $G_k^{(1)}$ further into smaller clusters should the contexts within $G_k^{(1)}$ are sparse. The lower level clustering on $G_k^{(1)}$ is performed on $\phi(c)$ such that $\phi(\cdot)$ is updated to the embedding of context from policy $\pi_k^{(1)}$, which resembles a top-down hierarchical clustering process. This iterative procedure is illustrated in Figure 4 and formally presented in Algorithm 1.

In Algorithm 1, a stopping criterion is specified for each cluster to determine whether it should be further divided.

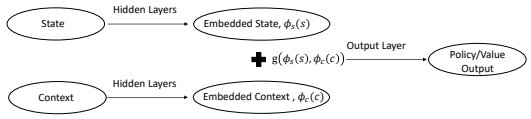

Fig. 3. Illustration of architecture $\mathcal{F}_1$ of the value network such that state s and context c are separately embedded.

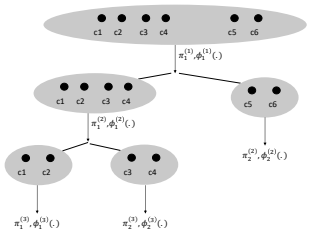

Fig. 4. Illustration of the Main Algorithm.

By default, the stopping criterion is based on the Silhouette coefficient, which measures the difference between the average intra-cluster distance and the average distance to contexts in the nearest neighboring cluster. A cluster will be further divided by K-means clustering if its Silhouette coefficient is below a threshold ϵ. If not manually specified, the number of clusters is determined by selecting the cluster count that results in the highest Silhouette coefficient. Other empirical methods, such as the elbow method, can also be used for selecting the optimal number of clusters. At the end of Algorithm 1, a list of local policies $L_\pi = \{\pi_1^{(j)}, \pi_2^{(j)}, \cdots, \pi_K^{(j)}\}$ are produced, corresponding to the local clusters $L_G = \{G_1^{(j)}, G_2^{(j)}, \cdots, G_K^{(j)}\}$. When applied on RL tasks, the aggregate policy based on L_π is used: if the context vector of the RL task's MDP belongs to cluster $G_k^{(j)}$, the corresponding policy $\pi_k^{(j)}$ is selected for this task.

B. Convergence Analysis

While the performance of Algorithm 1 is mainly supported by the numerical experiments in Section V, in this section, we provide some theoretical insights into its behavior, specifically on the convergence property of its clustering step. In the clustering literature, it has been well established that K-means clustering produces asymptotically optimal clustering results [20]. In our proposed Algorithm 1, K-means clustering is not directly applied on the original context space, but on the embedded context space derived from the estimated value function of RL. Therefore, existing theoretical guarantees for K-means clustering are not directly applicable in this setting. In this section, we prove the theoretical optimality of clustering in the embedded context space under specific conditions, providing a theoretical foundation for the accuracy of the clustering results in Algorithm 1. The main result is summarized in the following theorem.

Theorem IV.1. *Given the following two assumptions:*

Algorithm 1: Hierarchical K-means Clustering and RL Policy Training on Embedded Contexts

Data: The entire set of contexts $\{c_1, c_2, ..., c_N\}$ of cMDP (denoted as a single cluster $G_1^{(0)}$), an RL training algorithm Ψ with value network modeled by a neural network with architecture $\mathcal{F}_1$, maximum iteration number J, clustering stopping criteria ϵ, a list of clusters $L_G = \{G_1^{(0)}\}$, an empty list of local policies $L_\pi = \{\}$.

Train policy $\pi_1^{(0)}$ by the RL training algorithm Ψ with value network architecture $\mathcal{F}_1$ on $G_1^{(0)}$, and update $L_\pi = \{\pi_1^{(0)}\}$ as well as total cluster number $K = |L_G|$.

Update $\phi(c_i)$ for $i = 1, 2, ..., N$ where $\phi(\cdot)$ is the hidden layer of the value network corresponding to $\pi_1^{(0)}$.

for *Iteration index* $j = 1, \ldots, J$ **do**

  for *Each cluster* $G_k^{(j-1)} \in L_G$, $k = 1, \ldots, K$ **do**

    if *Silhouette coefficient of* $\{\phi(c_i) : c_i \in G_k^{(j-1)}\}$ *is less than* ϵ **then**

      Perform K-means clustering on $\{\phi(c_i) : c_i \in G_k^{(j-1)}\}$ to further divide it into smaller clusters $\{G_1^{(j)}, \cdots, G_{K^*}^{(j)}\}$ and determine the optimal cluster number K^* by the highest Silhouette coefficient.

      For each small cluster $G_{k^*}^{(j)}, k^* = 1, \cdots, K^*$, train policy $\pi_{k^*}^{(j)}$ on the corresponding MDPs on it with RL training algorithm Ψ and update $\phi(c_i)$ using the value network corresponding to $\pi_{k^*}^{(j)}$.

      Update L_G by replacing $G_k^{(j-1)}$ with the small clusters $\{G_1^{(j)}, \cdots, G_{K^*}^{(j)}\}$ and update L_π by replacing $\pi_k^{(j-1)}$ with local policies $\{\pi_1^{(j)}, \cdots, \pi_{K^*}^{(j)}\}$.

    end

    else

      Denote $G_k^{(j)} = G_k^{(j-1)}$ in L_G and denote $\pi_k^{(j)} = \pi_k^{(j-1)}$ in L_π

    end

  end

  Update the total cluster number $K = |L_G|$. If K is the same as previous iteration (no new cluster created), terminate the iterations and **Return** L_G and L_π.

end

Return L_G and L_π.

1) *The base RL algorithm Ψ has the property $\lim_{t\to\infty} \tilde{V}_t(s,c) = V^*(s,c)$, where $V^*(s,c)$ is the true value function and $\tilde{V}_t(s,c)$ is the estimated one at epoch t.*

2) *$V^*(s,c)$ is a generalized linear form of state s, $V^*(s,c) = \phi_s(s)^T\phi_c(c)$, where both $\phi_s(\cdot)$ and $\phi_c(\cdot)$ are some embedding functions that map from $\mathbb{R}^p$ to $\mathbb{R}^q$, satisfying one of the following two scenarios:*
   - ***Scenario (a)**: $\phi_s(\cdot)$ is the identity map (i.e., $V^*(s,c) = s^T\phi_c(c)$) and $\tilde{V}_t(s,c) = s^T\hat{\phi}_c(c)$;*
   - ***Scenario (b)**: $\phi_s(\cdot)$ is the same as $\phi_c(\cdot)$ (i.e., $V^*(s,c) = \phi_c(s)^T\phi_c(c)$) for some affine map function $\phi_c(\cdot)$ and $\tilde{V}_t(s,c) = \hat{\phi}_c(s)^T\hat{\phi}_c(c)$.*

Let $\bar{\boldsymbol{G}} = \bar{G}_1, \cdots, \bar{G}_K$ represent the optimal cluster centers in the space of $\phi_c(c)$ that minimize $\|\bar{G}_i - \phi_c(c)\|$. We have the following result: when K-means clustering is applied to the embeddings $\hat{\phi}_c(c)$ obtained from $\tilde{V}_t(s,c)$, the resulting cluster centers will almost surely converge to $\bar{\boldsymbol{G}}$.

Proof. To prove Theorem IV.1, we can begin with proving the consistency of clustering results in **Scenario (a)**, where $V^*(s,c) = s^T\phi_c(c)$ and $\tilde{V}_t(s,c) = s^T\hat{\phi}_c(c)$. Given $\lim_{t\to\infty}\tilde{V}_t(s,c) = V^*(s,c)$, it can be shown that the partial derivative on $\mathbb{R}^q$ for $\tilde{V}_t(s,c)$ and $V^*(s,c)$ are also the same. That is, $\frac{\partial\tilde{V}_t(s,c)}{\partial s_i} = \frac{V^*(s,c)}{\partial s_i}, \forall i \in \{1,2,\cdots,q\}$ when t goes to infinity, where q is the dimension of s. Note that $\frac{\partial\tilde{V}_t(s,c)}{\partial s_i} = \hat{\phi}_c(c)_i$ and $\frac{V^*(s,c)}{\partial s_i} = \phi_c(c)_i$, where $\hat{\phi}_c(c)_i$ and $\phi_c(c)_i$ are the i-th term of $\hat{\phi}_c(c)$ and $\phi_c(c)$ respectively. Therefore, it follows that $\hat{\phi}_c(c)_i \to \phi_c(c)_i, \forall i \in \{1,2,\cdots,q\}$ when $t \to \infty$. Since the estimated embedding $\hat{\phi}_c(c)$ converges to the true embedding $\phi_c(c)$, the results of K-means clustering on $\hat{\phi}_c(c)$ are equivalent to the results of K-means clustering on the space of $\phi_c(c)$, which will converge almost surely to the optimal clustering center $\bar{\boldsymbol{G}}$, based on the main theorem in [20].

To prove Theorem IV.1 in **Scenario (b)**, the general proof strategy is to prove that $\hat{\phi}_c(c)$ asymptotically preserves the Euclidean distance relations of $\phi_c(c)$, such that $\frac{\|\hat{\phi}_c(c_i)-\hat{\phi}_c(c_j)\|}{\|\hat{\phi}_c(c_k)-\hat{\phi}_c(c_l)\|} = \frac{\|\phi_c(c_i)-\phi_c(c_j)\|}{\|\phi_c(c_k)-\phi_c(c_l)\|}$, $\forall i,j,k,l$ when $\tilde{V}(s,c) \to V^*(s,c)$. Then in the final step, we can apply the main theorem in [20], which shows that K-means clustering results converge almost surely to the optimal cluster centers.

First, we try to propose an inverse map $\phi_c^{-1}(\cdot)$ that maps from $\mathbb{R}^q$ back to $\mathbb{R}^p$, recalling that $\phi_c(\cdot)$ is a map from $\mathbb{R}^p$ to $\mathbb{R}^q$. We need to define the quotient space of $\mathbb{R}^p$ by $\phi_c(\cdot)$, denoted as $K^*_{\phi_c}$. Essentially we have $\phi_c^{-1}(c') \in K^*_{\phi_c}$, such that $\phi_c^{-1}(c')$ is a subspace in $\mathbb{R}^p$ and this subspace is spanned on $\{c_i : \phi_c(c_i) = c'\}$. Then we can define the inner product $\langle,\rangle$ on this quotient space $K^*_{\phi_c}$, such that for two subspace $\phi_c^{-1}(c'_i)$ and $\phi_c^{-1}(c'_j)$ that belong to $K^*_{\phi_c}$, the inner product of them is $\langle\phi_c^{-1}(c'_i), \phi_c^{-1}(c'_j)\rangle = \min_{c_i\in\phi_c^{-1}(c'_i), c_j\in\phi_c^{-1}(c'_j)}(c_i^Tc_j)$. Then we know $\phi_c^{-1}(\cdot)$ is also an affine map.

Then we can construct a composite map $\hat{\phi}_c(\phi_c^{-1}(\cdot))$, which is a map from $\mathbb{R}^q$ to $\mathbb{R}^q$. It follows that this composite map $\hat{\phi}_c(\phi_c^{-1}(\cdot))$ must preserves the inner product on the space of $\mathbb{R}^q$ when $t \to \infty$, such that $\hat{\phi}_c(\phi_c^{-1}(c'_i))^T\hat{\phi}_c(\phi_c^{-1}(c'_j)) = c_i'^Tc'_j, \forall c'_i, c'_j \in \mathbb{R}^q$. To show this, we know that when $t \to \infty$, $\hat{\phi}(c_i)^T\hat{\phi}_c(c_j) = \phi_c(c_i)^T\phi_c(c_j)$ is true, as $\lim_{t\to\infty}\tilde{V}_t(s,c) = V^*(s,c)$. Then we know

$$\begin{aligned}&\hat{\phi}_c(\phi_c^{-1}(c'_i))^T\hat{\phi}_c(\phi_c^{-1}(c'_j))\\ &= \phi_c(\phi_c^{-1}(c'_i))^T\phi_c(\phi_c^{-1}(c'_j))\\ &= \phi_c(c_i)^T\phi_c(c_j),\end{aligned}$$

when t goes to infinity. Then based on Mazur–Ulam theorem [21], since the map $\hat{\phi}_c(\phi_c^{-1}(\cdot))$ is an isometry that preserves inner product on $\mathbb{R}^q$, $\hat{\phi}_c(\phi_c^{-1}(\cdot))$ must be an affine map. Since $\phi_c^{-1}(\cdot)$ is an affine map, $\hat{\phi}_c()$ should also be an affine map from $\mathbb{R}^p$ to $\mathbb{R}^q$.

Since affine map preserves the Euclidean distance, we know $\phi_c(c)$, such that $\frac{\|\hat{\phi}_c(c_i)-\hat{\phi}_c(c_j)\|}{\|\hat{\phi}_c(c_k)-\hat{\phi}_c(c_l)\|} = \frac{\|\phi_c(c_i)-\phi_c(c_j)\|}{\|\phi_c(c_k)-\phi_c(c_l)\|}, \forall i,j,k,l$ holds when $\tilde{V}(s,c) \to V^*(s,c)$. Therefore, K-means clustering on $\hat{\phi}_c(c)$ is equivalent to clustering on the space of $\phi_c(c)$, which will converge almost surely to the optimal clustering center $\bar{\boldsymbol{G}}$ based on the main theorem in [20].

□

Assumption 1 is common in contextual RL literature (see Theorem 1 of [3]), which can be satisfied by common RL algorithms such as Q-learning [22]. The generalized linear form of value function in Assumption 2 can also be commonly found in RL literature (see Section 3.1 of [18]).

In summary, Theorem IV.1 shows that K-means clustering on embedded contexts of Algorithm 1 is asymptotically consistent with the optimal centers on the embedded context space of the true value function, provided the value function estimation converges. In the next section, we present numerical studies demonstrating that inaccurate clustering results do not enhance RL policy performance, while our proposed RL training algorithm can enhance baseline policy. These findings underscore the critical role of accurate clustering in improving the effectiveness of RL policies.

V. Numerical Studies

In our numerical experiments, we compare policies returned by Algorithm 1, $\pi_{\text{Algo-1}}$, with two benchmarks: (1) π_{base}, trained on all contexts without clustering; and (2) $\pi_{\text{direct-cluster}}$, trained on clusters obtained by directly clustering the original input contexts with K-means. In Subsection V-A, context of RL includes dummy variables that won't affet MDP but will distract clustering algorithms, demonstrating that $\pi_{\text{Algo-1}}$ is robust to such distractions. In Subsection V-B, context variables of RL environment affect the MDP via a transition function, and $\pi_{\text{Algo-1}}$ still outperforms both baselines as context embedding from this algorithm is robust to this context transition. In Subsection V-A and V-B, RL policies are evaluated on contextual RL versions of three classic Gym environments: Pendulum, CartPole, and Acrobot. These environments are simulated using the CARL library [23]. In Subsection V-C, Algorithm 1 is applied in a realistic asset allocation environment from OR-Gym [24], where it consistently outperforms the baselines in this complicated RL-based stock trading scenario. In all the experiments, RL policies are trained on an NVIDIA T4 GPU.

TABLE I
EXPERIMENTS WHERE THE CONTEXT INPUT CONTAINS EFFECTIVE CONTEXT VARIABLES AND DUMMY CONTEXT VARIABLES.

Experiment	Dummy-Pendulum	
Method	Value	Standard Error
π_{base}	-840.71	14.45
$\pi_{direct-cluster}$	-818.44	11.35
π_{Algo-1}	**-781.41**	12.82
Experiment	Dummy-CartPole	
Method	Value	Standard Error
π_{base}	92.1	1.63
$\pi_{direct-cluster}$	94.045	0.89
π_{Algo-1}	**96.987**	0.68
Experiment	Dummy-Acrobot	
Method	Value	Standard Error
π_{base}	-99.34	0.26
$\pi_{direct-cluster}$	-77.03	1.76
π_{Algo-1}	**-71.102**	2.06

Both $\pi_{\text{Algo-1}}$ and $\pi_{\text{direct-cluster}}$ consist of multiple RL policies corresponding to K clusters. To represent the performance of the composite policy $\pi_1, \pi_2, \cdots, \pi_K$ from $\pi_{\text{Algo-1}}$ or $\pi_{\text{direct-cluster}}$ as a single value, we first evaluate each policy π_k individually, following the same procedure used for π_{base}. Specifically, each policy runs in the contextual RL simulator with 100 random seeds, where contexts are randomly sampled in these 100 evaluation iterations. The overall value of the composite policy is then calculated as a weighted average of the estimated values for the K policies, with weights proportional to the size of each cluster ($\frac{|G_k|}{\sum_{k=1}^{K}|G_k|}$). The standard error of the composite policy is computed based on the weighted average, assuming the policies trained on different clusters are independent.

A. Simulation on Contexts with Dummy Variables

In this section, we study contextual RL environments where the context input contains two parts: *effective* context variables that affect reward and state transition in the MDPs and *dummy* context variables that do not affect them. The contextual RL environments include contextual RL versions of the classic Pendulum, CartPole, and Acrobot gym environment, and the simulator of these environments are implemented by the CARL library in [23].

Contextual Pendulum environment. Focusing on the Contextual Pendulum environment, the contextual inputs include a gravity value variable (c_g) and a single dummy variable (c_d), denoted as $c = \{c_g, c_d\}$. The dummy variable c_d does not affect transitions or rewards in the MDP. We define $N = 200$ contextual MDPs (cMDPs), where c_g is drawn from a normal distribution $N(10.0, 1.0)$ for cMDPs indexed from 0 to 99, and from $N(50.0, 5.0)$ for those indexed from 100 to 199. We anticipate identifying a true cluster structure comprising two groups: one with a low gravity setting, including cMDPs indexed from 0 to 99, and the other with a high gravity setting, including those indexed from 100 to 199.

The dummy variable c_d is sampled from $N(400.0, 120.0)$ for cMDPs 50 to 149, and from $N(10.0, 3.0)$ for the remaining cMDPs. The dummy variable c_d only functions as a noise term

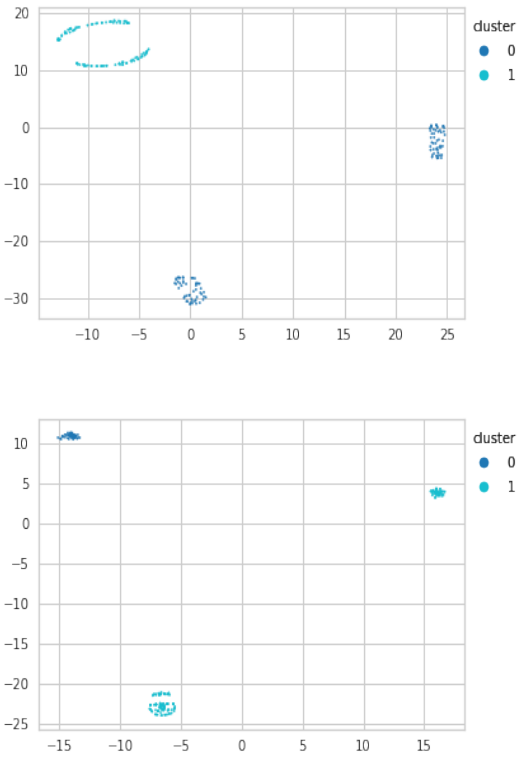

Fig. 5. **Upper:** Clustering results of contexts from direct clustering on contexts $c = \{c_g, c_d\}$ for Dummy-Pendulum environment; **Lower:** Clustering results of contexts from clustering on embedded contexts $\hat{\phi}_c(c)$ from the value network, as is proposed in Algorithm 1.

in the input of the observed context that increases the difficulty of clustering, such that directly clustering on $c = \{c_g, c_d\}$ can lead to inaccurate clustering labels. When specifying the cluster number $K = 2$, the clustering outcomes of the embedded contexts using Algorithm 1 and results from direct clustering on the original contexts are shown in Figure 5. In this setting, direct clustering on $c = \{c_g, c_d\}$ results in incorrect clusters. These clusters group contexts with high dummy variable values c_d into one cluster, and those with low dummy variable values into the other, where the two clusters fail to differentiate the contexts with low and high gravity. This occurs because the variation in the dummy variable c_d dominates the true underlying cluster structure dictated by the effective gravity context variable c_g.

In contrast, clustering in the embedded context space $\hat{\phi}_c(c)$, minimizes the influence of dummy variables and accurately identifies a cluster consisting only of cMDPs with low gravity settings. We assess the clustering effectiveness by comparing it to the actual cluster structure using the Pearson correlation coefficient. For direct clustering using K-means, the correlation is 0, showing no relation to the true clusters. In contrast, the embedded clustering from our algorithm shows a positive correlation of 0.58, indicating that clustering on the embedded context space is significantly more robust to noises

from dummy variables.

Other experiments with higher context dimensions. Similar settings that contain both an effective context variable and a dummy context variable are also adapted for the contextual versions of CartPole and Acrobot. In these two experiments, the dimensions of contexts are higher than the simple two-context setting of Pendulum. In particular, for Contextual CartPole environment, the context includes gravity (c_g), mass of the pole (c_m), length of the pole (c_l), and three dummy variables (c_{d1}, c_{d2}, c_{d3}). In the Contextual Acrobot environment, the context consists of the lengths of the first and second links (c_{l1}, c_{l2}), masses of the first and second links (c_{m1}, c_{m2}), and three dummy variables (c_{d1}, c_{d2}, c_{d3}). Similar to the cluster setting in Contextual Pendulum, in these two environments, the meaningful context variables that affect the MDP reward and state transitions, such as gravity, link lengths, and masses, naturally form two clusters: one with high magnitudes and the other with low magnitudes. The dummy variables, which are grouped into low- and high-value clusters, introduce noise that distracts clustering algorithms from identifying the true underlying structure.

The evaluated values of all policies on these environments are presented in Table I. The results show that the composite policy, $\pi_{\text{Algo-1}}$, generated by Algorithm 1, significantly outperforms the two baseline policies, π_{base} and $\pi_{direct-cluster}$, as confirmed by Welch's t-test at a 95% confidence level. In the Contextual Pendulum environment, the performance of $\pi_{direct-cluster}$ does not significantly differ from that of π_{base}. This outcome is expected, as direct clustering fails to identify meaningful clusters that differentiate low-gravity and high-gravity MDP environments. Consequently, the average values of the policies trained on the clusters formed by direct K-means do not show significant differences, as the gravity coefficients of environments in the clusters are evenly distributed. By contrast, the clusters identified by Algorithm 1 effectively separate low-gravity and high-gravity environments. Within these clusters, the policy trained on the low-gravity cluster achieves a much higher average value compared to the policy trained on the high-gravity cluster. This difference underscores the ability of Algorithm 1 to leverage meaningful context structures to improve performance.

Furthermore, the clustering method in Algorithm 1 can be substituted with other approaches. For example, when DBSCAN [6] is used, the evaluated values of π_{base}, $\pi_{direct-cluster}$, and π_{Algo-1} on Contextual Pendulum are -773.73, -780.64, and -737.58, respectively. In this setting, the proposed $\pi_{\text{Algo-1}}$ still significantly outperforms the two baselines. When applying K-means algorithm, either the number of clusters is pre-specified as $K = 2$ or automatically selected by Silhouette coefficient, we can observe similar trend of performance of π_{Algo-1}.

B. Simulation on Contexts Requiring Transformation

In this section, we consider another environment setting, where the optimal context space to cluster is not by removing dummy variables but by transformation the raw context variables. Specifically, we configure the context input as a vector $c = \{c_{g_1}, c_{g_2}\}$, where the key context variable—such as gravity in Pendulum/CartPole and the lower link length in Acrobot—is determined by the difference between the two context variables (say, $c_{g_2} - c_{g_1}$). This means the actual values for contexts such as gravity or link length require a linear transformation from the original context input.

In this setting, the underlying true cluster structure is set as one cluster with low values of $c_{g_2} - c_{g_1}$ and another with high values. Directly clustering on original context variables $c = \{c_{g_1}, c_{g_2}\}$ may still lead to sub-optimal cluster results, as the clustering will be distracted by dominating trends in invidual context variables rather than capturing the critical diverging trend of the difference between c_{g_1} and c_{g_2}.

The clustering results on original contexts c and embedded contexts $\hat{\phi}_c(c)$ in Contextual Pendulum environment are shown in Figure 6, where cluster number K is manually set as 2. Note that in this setting, the transformation of the original input vector $c = \{c_{g_1}, c_{g_2}\}$, $c_{g2} - c_{g1}$, is the true gravity value in Pendulum. It is shown that clustering results on $\hat{\phi}_c(c)$ (clusters retunred by Algorithm 1) are better as it successfully captures a cluster of environments with low gravity value. On the other hand, directly clustering is misguided by the diverging trend in c_{g_2}. Two clusters returned by direct K-means clustering contain similar true gravity distributions as the entire population and fail to differentiate the low gravity group and the high gravity group. The correlation coefficient of clusters by Algorithm 1 is 0.58, while the correlation coefficient of direct clustering results is 0.

The performance of RL policies are shown in Table II. From Table II, it is known that the composite policy returned by Algorithm 1 significantly outperforms the π_{base} and $\pi_{direct-cluster}$. Since the direct clustering results didn't differentiate the groups with high $c_{g2} - c_{g1}$ values and low $c_{g2} - c_{g1}$ values, the performance of $\pi_{direct-cluster}$ is not significantly different from π_{base} trained on the entire set of contexts. In contextual Acrobot environment, the baseline policy π_{base} trained on all cMDPs doesn't converge and the reward remains to be the lowest value in the environment, which is -200. For the policies trained on two clusters of Algorithm 1, the policy trained on the cluster with larger link length converges, leading to a higher performance.

C. Real-World Application: Stock Trading

Setup. In this section, we utilize the asset allocation Gym environment from OR-Gym [24], which studies RL-based stock trading. The RL agent starts with an initial cash amount and can buy or sell one of three stocks. The expected prices of the stocks are set with specific trends: stock 1's expected price consistently increases, stock 3's decreases, and stock 2's remains constant. The actual prices of the stocks are these expected values plus white noises. To transform the original RL environment into multiple contextual MDPs (cMDPs), we define, $c = \{c_{cash}, c_{buy,1}, c_{sell,1}, c_{buy,2}, c_{sell,2}, c_{buy,3}, c_{sell,3}\}$, a seven-dimensional context vector representing transaction fees for buying and selling each stock, and the initial cash

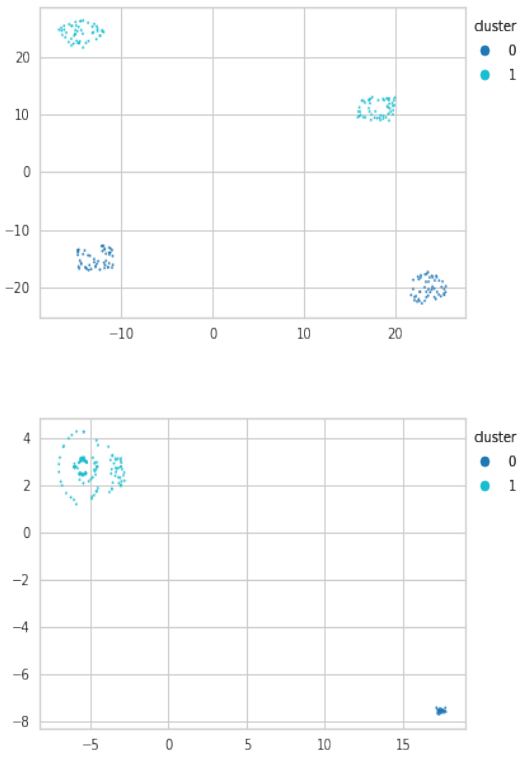

Fig. 6. **Upper:** Clustering results of contexts from direct clustering on contexts $c = \{c_{g_1}, c_{g_2}\}$ for Transformation-Pendulum environment; **Lower:** Clustering results of contexts from clustering on embedded space of contexts $\hat{\phi_c}(c)$ from the value network

TABLE II
EXPERIMENTS WHERE THE TRUE CONTEXTUAL VALUE THAT AFFECTS MDP REWARD AND STATE TRANSITION IS THE DIFFERENCE OF THE TWO CONTEXTUAL VARIABLES.

Experiment	Transformation-Pendulum	
Method	Value	Standard Error
π_{base}	-1637.94	28.39
$\pi_{direct-cluster}$	-1640.62	17.86
π_{Algo-1}	**-1539.44**	32.52
Experiment	Transformation-CartPole	
Method	Value	Standard Error
π_{base}	93.87	9.60
$\pi_{direct-cluster}$	95.21	6.72
π_{Algo-1}	**109.97**	6.72
Experiment	Transformation-Acrobot	
Method	Value	Standard Error
π_{base}	-200.0	0.0
$\pi_{direct-cluster}$	-200.0	0.0
π_{Algo-1}	**-171.21**	0.51

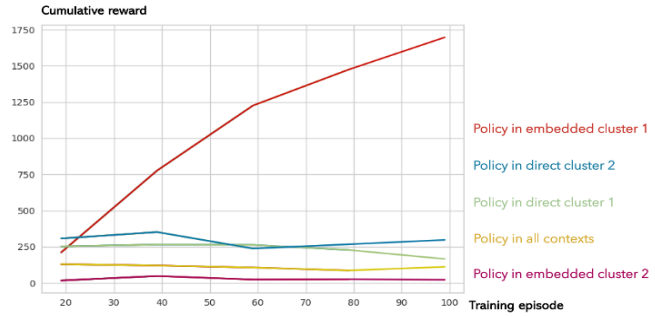

Fig. 7. Cumulative reward of local π_{Algo-1} and $\pi_{direct-cluster}$ policies trained on clusters, as well as the baseline policy π_{base} trained on the entire set of cMDPs

(c_{cash}) available to the trading agent. This experiment features 200 contextual MDPs (cMDPs) with four different transaction cost scenarios: 50 cMDPs have low costs for all stocks (around 1% of the initial stock price); 50 have low costs for stock 1 but high costs for stocks 2 and 3 (10% of the initial price for stocks 2 and 3); another 50 have low costs for stocks 2 and 3 but high costs for stock 1 (50% of stock 1's initial price); and the remaining 50 have high transaction costs for all stocks (50% for stock 1 and 10% for stocks 2 and 3). When clustering, the number of clusters is set to be 2.

Results. After applying Algorithm 1, clustering on embedded contexts set the 50 cMDPs with low transaction cost for all stocks as cluster 1 and all the other 150 cMDPS as cluster 2. On the other hand, if we directly apply K-means on the original context input, the 100 cMDPs with low transaction cost for stock 1 is grouped into cluster 1 while the other 100 cMDPs with high transaction cost for stock 1 is sent to cluster 2, regardless of the transaction cost of stock 2 and 3.

Table III shows that the performance of π_{Algo-1} is significantly better than the other two policies. In particular, in Figure 7, we plot the cumulative rewards during training for local π_{Algo-1} and $\pi_{direct-cluster}$ policies policies on different clusters, as well as the baseline policy π_{base}. It is observed that the local policy trained on the cMDPs with low transaction cost for all stocks from π_{Algo-1} (the red plot line) converges well, while all the other policies don't learn a efficient trading strategy. From this result, it is shown that π_{Algo-1} can successfully identify the group of cMDPs with low transaction costs for all stocks and train a successful policy on this cluster of cMDPs. However, $\pi_{direct-cluster}$ and π_{base} can not learn an efficient policy due to the challenges from high transaction costs for stock 2 and 3. This explains the superior performance of π_{Algo-1} among the three policies in Table III. When using other RL algorithms such as A2C and APPO, the relative performance of policies display similar trends.

VI. CONCLUSION

In this paper, we have shown that training contextual RL agents on clusters can improve the performance of the learned policies. In particular, we propose to perform clustering on

TABLE III
EXPERIMENTS ON A CONTEXTUAL RL ENVIRONMENT FOR ASSET ALLOCATION WITH THREE STOCKS, WHERE THE CONTEXT INPUT IS A DIMENSION 7 VECTOR CONTAINING BUYING/SELLING FEES AND INITIAL CASH AMOUNT.

Experiment	Contextual-Asset-Allocation		
RL Algorithm	Method	Value	Standard Error
PPO	π_{base}	120.94	3.19
PPO	$\pi_{direct-cluster}$	302.70	7.91
PPO	π_{Algo-1}	**461.31**	3.64
A2C	π_{base}	975.64	21.34
A2C	$\pi_{direct-cluster}$	851.53	7.78
A2C	π_{Algo-1}	**1556.04**	19.44
APPO	π_{base}	1465.56	21.54
APPO	$\pi_{direct-cluster}$	1230.93	10.14
APPO	π_{Algo-1}	**1543.51**	12.18

embedded contexts from value functions to ensure the MDP environments in each cluster are close to each other. Furthermore, we have shown the asymptotic convergence of clusters on embedded contexts to the optimal clusters.

An inherent trade-off of our method is the computational cost, as it involves first training the baseline RL policy to obtain context embeddings, performing clustering on these embeddings, and then training new policies within the identified clusters. The additional training step is essential to leverage the clustering information and achieve improved policy performance. Therefore, future work could explore more time-efficient RL training algorithms and improved clustering techniques, such as online clustering methods, to reduce computational overhead and improve scalability of this method. Additionally, given the widespread use of multi-agent RL in real applications [25], integrating our approach with multi-agent RL frameworks presents another promising research direction.

REFERENCES

[1] A. Hallak, D. Di Castro, and S. Mannor, "Contextual markov decision processes," *arXiv preprint arXiv:1502.02259*, 2015.

[2] P. Klink, C. D'Eramo, J. R. Peters, and J. Pajarinen, "Self-paced deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9216–9227, 2020.

[3] T. Eimer, A. Biedenkapp, F. Hutter, and M. Lindauer, "Self-paced context evaluation for contextual reinforcement learning," in *International Conference on Machine Learning*, pp. 2948–2958, PMLR, 2021.

[4] S. Sodhani, A. Zhang, and J. Pineau, "Multi-task reinforcement learning with context-based representations," in *International Conference on Machine Learning*, pp. 9767–9779, PMLR, 2021.

[5] A. Modi and A. Tewari, "Contextual markov decision processes using generalized linear models," *arXiv preprint arXiv:1903.06187*, 2019.

[6] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, pp. 226–231, 1996.

[7] P. Berkhin, "A survey of clustering data mining techniques," *Grouping multidimensional data: Recent advances in clustering*, pp. 25–71, 2006.

[8] T.-S. Chen, T.-H. Tsai, Y.-T. Chen, C.-C. Lin, R.-C. Chen, S.-Y. Li, and H.-Y. Chen, "A combined k-means and hierarchical clustering method for improving the clustering efficiency of microarray," in *2005 International symposium on intelligent signal processing and communication systems*, pp. 405–408, IEEE, 2005.

[9] D. Ghosh, A. Singh, A. Rajeswaran, V. Kumar, and S. Levine, "Divide-and-conquer reinforcement learning," *arXiv preprint arXiv:1711.09874*, 2017.

[10] A. Kabra, A. Agarwal, and A. S. Parihar, "Cluster-based deep contextual reinforcement learning for top-k recommendations," in *Proceedings of the International Conference on Computing and Communication Systems: I3CS 2020, NEHU, Shillong, India*, pp. 125–135, Springer, 2021.

[11] A. Kabra, A. Agarwal, and A. S. Parihar, "Potent real-time recommendations using multimodel contextual reinforcement learning," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 2, pp. 581–593, 2021.

[12] Y. Li, Y. Zheng, and Q. Yang, "Efficient and effective express via contextual cooperative reinforcement learning," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 510–519, 2019.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, R. Graves, A., F. M., G. A.K., Ostrovski, and S. Petersen, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[14] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[15] V. Mnih, A. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *International conference on machine learning*, pp. 1928–1937, 2016.

[16] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," *International conference on machine learning*, pp. 1889–1897, 2015.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[18] E. Y. Chen, R. Song, and M. I. Jordan, "Reinforcement learning with heterogeneous data: estimation and inference," *arXiv preprint arXiv:2202.00088*, 2022.

[19] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *International conference on machine learning*, pp. 1312–1320, PMLR, 2015.

[20] D. Pollard, "Strong consistency of k-means clustering," *The annals of statistics*, pp. 135–140, 1981.

[21] S. Mazur and S. Ulam, "Sur les transformations isométriques d'espaces vectoriels normés," *CR Acad. Sci. Paris*, vol. 194, no. 946-948, p. 116, 1932.

[22] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep q-learning," in *Learning for dynamics and control*, pp. 486–489, PMLR, 2020.

[23] C. Benjamins, T. Eimer, F. Schubert, A. Biedenkapp, B. Rosenhahn, F. Hutter, and M. Lindauer, "Carl: A benchmark for contextual and adaptive reinforcement learning," *arXiv preprint arXiv:2110.02102*, 2021.

[24] C. D. Hubbs, H. D. Perez, O. Sarwar, N. V. Sahinidis, I. E. Grossmann, and J. M. Wassick, "Or-gym: A reinforcement learning library for operations research problems," *arXiv preprint arXiv:2008.06319*, 2020.

[25] Y. Bai, Y. Gao, R. Wan, S. Zhang, and R. Song, "A review of reinforcement learning in financial applications," *Annual Review of Statistics and Its Application*, vol. 12, 2024.