Front. Comput. Sci., 2025, 19(4): 194313
https://doi.org/10.1007/s11704-024-3194-1

RESEARCH ARTICLE

Clustered Reinforcement Learning

Xiao MA[1], Shen-Yi ZHAO[1], Zhao-Heng YIN[2], Wu-Jun LI (✉)[1]

1 National Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China

2 Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720-1770, USA

Abstract Exploration strategy design is a challenging problem in reinforcement learning (RL), especially when the environment contains a large state space or sparse rewards. During exploration, the agent tries to discover unexplored (novel) areas or high reward (quality) areas. Most existing methods perform exploration by only utilizing the novelty of states. The novelty and quality in the neighboring area of the current state have not been well utilized to simultaneously guide the agent's exploration. To address this problem, this paper proposes a novel RL framework, called clustered reinforcement learning (CRL), for efficient exploration in RL. CRL adopts clustering to divide the collected states into several clusters, based on which a bonus reward reflecting both novelty and quality in the neighboring area (cluster) of the current state is given to the agent. CRL leverages these bonus rewards to guide the agent to perform efficient exploration. Moreover, CRL can be combined with existing exploration strategies to improve their performance, as the bonus rewards employed by these existing exploration strategies solely capture the novelty of states. Experiments on four continuous control tasks and six hard-exploration Atari-2600 games show that our method can outperform other state-of-the-art methods to achieve the best performance.

Keywords deep reinforcement learning, exploration, count-based method, clustering, *K*-means

1 Introduction

Reinforcement learning (RL) [1], especially deep RL (DRL), has recently attracted much attention and achieved significant performance in a variety of applications, such as game playing [2−6], robotics control [7−12], natural language processing [13,14], document styling restoration [15], scheduling problem [16,17], finance [18−22] and large language models [23]. However, there are still many problems impeding RL from being applied more broadly in practice. A critical problem in RL is how to design a reasonable exploration strategy that can balance the relationship between exploration and exploitation effectively [1,24,25]. If the agent explores novel states excessively, it might never find any positive rewards to guide the learning direction. On the other hand, if the agent exploits rewards too intensely, it might converge to sub-optimal behaviors and have a low opportunity to discover unexplored areas with higher rewards. This problem gets severe in many real-world scenarios where the state space of the environment is continuous and high-dimensional, or rewards from the environment are extremely sparse.

Existing exploration strategies can be classified into heuristic exploration strategies [1,2,26−31] and exploration strategies with environmental information [24,32−46], according to whether they use environmental information. Heuristic exploration strategies are limited to environments with small state spaces or well-shaped rewards. Heuristic exploration strategies cannot yield satisfactory performance for hard applications or games because they ignore the impact of environmental information on exploration. Compared with heuristic exploration strategies, exploration strategies with environmental information [24,32−42] can utilize environmental information to assist the agent in exploring. These exploration strategies with environmental information have shown advancement, particularly in hard games, by leveraging the novelty of states to assist the agent in exploring the environment. However, it is not enough to explore only based on the novelty of states. This is because focusing only on the novelty of states might divert the agent's attention to high novelty areas excessively while ignoring high reward (quality) areas. Conversely, a disproportionate emphasis on the quality of states could lead the agent to overlook the discovery of unexplored areas with potentially higher rewards. Hence, areas worth exploring for the agent should be considered from both novelty and quality. That is to say, in addition to utilizing the novelty of states, the quality of states is also crucial for effectively guiding the agent's exploration direction. To the best of our knowledge, the novelty and quality in the neighboring area of the current state have not been well utilized to simultaneously guide the agent's exploration. To address this problem, this paper proposes a novel RL framework, called clustered reinforcement learning (CRL), for efficient exploration in RL. The contributions of CRL are briefly outlined as follows:

Received March 6, 2023; accepted February 28, 2024

E-mail: liwujun@nju.edu.cn

- CRL adopts clustering to divide the collected states into several clusters. The states from the same cluster have similar features. Hence, the clustered results in CRL provide a possibility to share meaningful information among different states from the same cluster.
- CRL proposes a novel bonus reward that reflects both novelty and quality in the neighboring area of the current state. Here, the neighboring area is defined by the states that share the same cluster with the current state. This bonus reward can guide the agent to perform efficient exploration by seamlessly integrating novelty and quality of states.
- CRL can be combined with existing exploration strategies that use the novelty of states as bonus rewards to guide the agent's exploration. Hence, CRL enables utilizing both the quality and novelty of states to effectively guide the agent towards efficient exploration and then improve the performance of existing exploration strategies.
- Experimental results on four continuous control tasks [10] with sparse rewards and six hard-exploration Atari-2600 games [47] demonstrate that CRL can outperform other state-of-the-art methods to achieve the best performance in most cases. On several games that are hard for heuristic exploration strategies, CRL achieves significant improvement over baselines.

2 Related work

2.1 Exploration strategies

In RL, exploration strategies can be classified into two categories: heuristic exploration strategies and exploration strategies with environmental information. Heuristic exploration strategies help the agent in exploration without utilizing environmental information and are effective in environments with small state spaces or well-shaped reward schemes. Representative heuristic exploration strategies include ϵ-greedy [1], uniform sampling [2], Thomson sampling [26], entropy loss term [27], and noise-based exploration [28−31]. However, in many real-world scenarios, the state space of the environment is continuous and high-dimensional, or rewards from the environment are extremely sparse, which renders heuristic strategies ineffective in exploration. To address these issues, exploration strategies that utilize environmental information are necessary.

Exploration strategies with environmental information help the agent perform exploration by utilizing environmental information. These strategies include domain knowledge-based methods, count-based methods, intrinsic-motivation methods, and more. The main idea of these methods is to encourage agents to explore novel (curious) areas. Domain knowledge-based methods [43−46] require a world model with rich domain knowledge. However, having or learning a world model for the agent is often impractical in complex environments.

Count-based methods count the visited experiences as a direct way to measure the novelty of states. In the tabular setting and finite Markov decision processes (MDPs), the number of states or state-action pairs can be counted directly based on the table because the number of states or state-action pairs is finite. These works, called table-based methods, include E^3 [32], R-Max [33], MBIE-EB [48], UCRL [43], and UCAGG [49]. However, these table-based methods become impractical in continuous and high-dimensional state spaces. This is because the vast number of states make direct counting infeasible and only a few states are visited more than once. To address these issues, researchers propose pseudo-counts-based methods [34,35] and hash-based method [24]. Pseudo-counts-based methods [34,35] require learning a density model to estimate visit counts of states, which is subsequently used to formulate the bonus reward for exploration. The hash-based method [24] uses a hash function to encode the states and a table to record the visit counts of states for each encoded state (similar to table-based methods). It explores by using the reciprocal of visit counts of states as a form of bonus reward, which performs well on some hard exploration Atari-2600 games.

Intrinsic-motivation methods [36−38] aim to encourage agents to explore states which are novel for agents. These methods are inspired by the intrinsic motivation literature [50] in psychology. Intrinsic motivation is a psychological mechanism that explains the exploratory behaviors observed in humans [51]. VIME [36] uses information gain about the agent's belief of environment dynamics as intrinsic rewards for exploration. Exploration-AE [37] and ICM [38] both construct an environment model to evaluate the novelty of states. Additionally, empowerment [39] and prediction by exemplar model [40] can serve as measures of the novelty of states. Another line of research in intrinsic-motivation methods uses the state-difference as the bonus reward for the agent. Both RND [41] and NovelD [42] use the difference between a random fixed target network and a trainable predictor network to evaluate the novelty of states. These two methods demonstrate strong performance on Montezuma's Revenge, a hard-exploration Atari 2600 game. Furthermore, RND [41] discusses balancing intrinsic returns (total discounted bonus rewards) and returns (total discounted rewards, where rewards are from the environment). Although the bonus reward can reflect the novelty of the current state, the reward from the environment can not reflect the quality in the neighboring area of the current state. Hence, RND does not use both the novelty and quality of states for exploration. Among all exploration strategies with environmental information, count-based methods and intrinsic-motivation methods are more practical. However, these methods primarily focus on utilizing the novelty of states for exploration while overlooking the importance of the quality of states.

2.2 Clustering algorithms in RL

In [52], researchers adopt clustering algorithms to partition the state space to different regions and then use different policies in different regions. In [53], researchers use clustering algorithms for value function approximation. RODE [54] and SePS [55] adopt clustering algorithms to discover a set of roles in a multi-agent setting. Although these methods [52−55] utilize clustering algorithms, clustering algorithms in these methods are not employed for exploration in MDPs.

Furthermore, TCRL [56] utilizes clustering of the dynamics and posterior sampling to balance exploration and exploitation. However, TCRL is designed for discrete MDPs with small/medium state spaces, which encounters difficulties when applied to large state spaces [56]. This constraint limits its practicality in real-world environments with large state spaces, particularly those with continuous and high-dimensional states.

3 Clustered reinforcement learning

This section presents the details of our proposed RL framework, called clustered reinforcement learning (CRL). The key idea of CRL is to adopt clustering to divide the collected states into several clusters and then design a novel clustering-based bonus reward for exploration by integrating both the novelty and quality of the states.

3.1 Notation

In this paper, we adopt similar notations as those in [24]. More specifically, we model the RL problem as a finite-horizon discounted MDP, which can be defined by a tuple $(\mathcal{S},\mathcal{A},\mathcal{P},r,\rho_0,\gamma,T)$. Here, $\mathcal{S}\subseteq\mathbb{R}^m$ denotes the state space. $\mathcal{A}\subseteq\mathbb{R}^n$ denotes the action space. $\mathcal{P}:\mathcal{S}\times\mathcal{A}\times\mathcal{S}\to[0,1]$ denotes a transition probability distribution. $r:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ denotes a reward function. ρ_0 is an initial state distribution. $\gamma\in(0,1]$ is a discount factor. T denotes the horizon time. In this paper, we assume $r\geqslant 0$. For cases with negative rewards, we can transform them into cases without negative rewards. The goal of RL is to find a policy π with the parameter θ for maximizing $J(\theta)=\mathbb{E}_{\pi_\theta,\rho_0}\left[\sum_{t=0}^{T-1}\gamma^t r(s_t,a_t)\right]$, which is the expected total discounted reward over the policy π_θ.

3.2 Clustering

Intuitively, both novelty and quality of states are helpful for the design of an exploration strategy. If the agent only cares about novelty, it might explore some unexplored areas intensively without any reward. If the agent only cares about quality, it might converge to sub-optimal behaviors and have a low opportunity to discover unexplored areas with higher rewards. Hence, it is better to integrate both novelty and quality into the same exploration strategy.

We find that clustering can provide the possibility of integrating both novelty and quality. Intuitively, a cluster of states represents an area. The number of collected states in a cluster reflects the count (inverse novelty) information of that area. The average reward of the collected states in a cluster reflects the quality of that area. Hence, based on the clustered results, we can design an exploration strategy considering both novelty and quality. Furthermore, states from the same cluster share similar characteristics, and hence the clustered results provide a possibility to share meaningful information among different states from the same cluster. The details of the exploration strategy design based on clustering will be left to the following subsection. Here, we only describe the clustering algorithm.

In CRL, we perform clustering on states. Assume the number of clusters is K, and we have collected N state-action samples $\{(s_i,a_i,r_i)\}_{i=1}^N$ with the same policy. We need to cluster the collected states $\{s_i\}_{i=1}^N$ into K clusters by using some clustering algorithm $f:\mathcal{S}\to C$, where $C=\{C_k\}_{k=1}^K$ and C_k is the center of the kth cluster. We can use any clustering algorithm in the CRL framework. Although more sophisticated clustering algorithms might achieve better performance, we choose the K-means algorithm [57] in this paper for illustration. K-means is one of the simplest clustering algorithms and has been applied across broad applications. The goal of K-means is to minimize the within-cluster sum-of-squares criterion:

$$L(C)=\sum_{i=1}^{N}\min_{C_k\in C}\left(\|s_i-C_k\|^2\right), \tag{1}$$

where $\|s_i-C_k\|$ denotes the euclidean distance between s_i and the kth cluster center C_k, and $k\in\{1,...,K\}$. The first step of K-means is to choose the initial cluster centers from the samples $\{s_i\}_{i=1}^N$. After the first step, K-means consists of looping between the second step and the third step. The second step is to allocate each sample to its nearest cluster by $\phi(s_i)=\arg\min_k\|s_i-C_k\|,\forall i:1\leqslant i\leqslant N,k:1\leqslant k\leqslant K$. The third step is to update new cluster centers as follows:

$$C_k=\frac{1}{\sum_{i=1}^{N}\mathbb{I}(\phi(s_i)=k)}\sum_{i=1}^{N}s_i\mathbb{I}(\phi(s_i)=k),1\leqslant k\leqslant K, \tag{2}$$

where $\mathbb{I}(\cdot)$ is an indicator function. The loop repeats until the difference between the value of Eq. (1) with the old cluster centers and that with the new cluster centers is less than a threshold. Finally, we obtain the cluster centers $C=\{C_k\}_{k=1}^K$.

3.3 Clustering-based bonus reward

As stated above, clustering can provide the possibility of integrating both novelty and quality for exploration. Here, we propose a novel clustering-based bonus reward, based on which many *policy updating algorithms* can be adopted to get an exploration strategy considering both novelty and quality.

Given a state s_i, it will be allocated to the nearest cluster by the cluster assignment function $\phi(s_i)=\arg\min_k\|s_i-C_k\|$, where $\|s_i-C_k\|$ denotes the euclidean distance between s_i and the k-th cluster center C_k, and $k\in\{1,...,K\}$. The sum of rewards in the kth cluster is denoted as R_k, which can be computed as follows:

$$R_k=\sum_{i=1}^{N}r_i\mathbb{I}(\phi(s_i)=k). \tag{3}$$

R_k is also called *cluster reward* of cluster k in this paper. The number of states in the kth cluster is denoted as N_k, which can be computed as follows:

$$N_k=\sum_{i=1}^{N}\mathbb{I}(\phi(s_i)=k). \tag{4}$$

Intuitively, a larger N_k typically means that the area corresponding to cluster k has been explored more times which implies the novelty of this area is lower. Hence, the bonus reward should be negatively correlated with N_k from the novelty perspective. The average reward of cluster k,

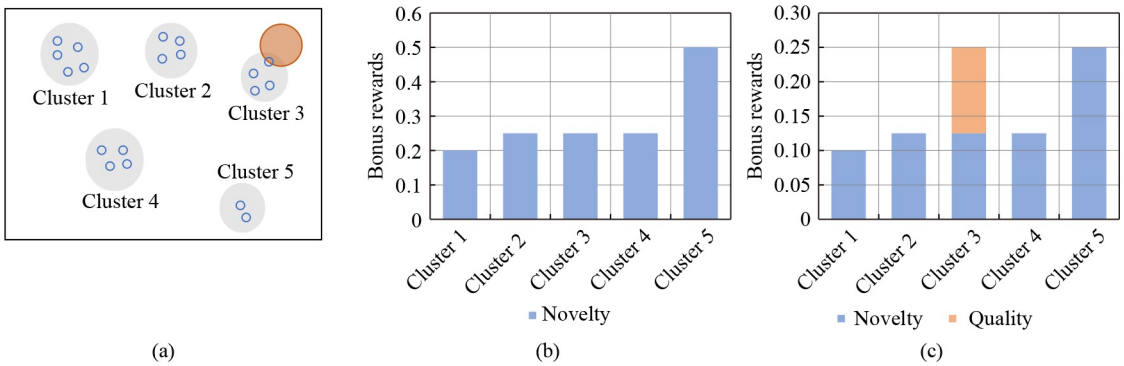

Fig. 1 (a) Using clustering to divide the collected states (blue dots) into 5 clusters. The agent is rewarded with 1 in the orange area and receives no reward in other areas; (b) the clustering-based bonus rewards with novelty alone ($\eta = 1.0$); (c) the clustering-based bonus rewards ($\eta = 0.5$). The blue bar represents the portion of bonus rewards reflecting the novelty of states, and the orange bar represents the portion reflecting the quality of states

denoted as R_k/N_k, can be used to represent the quality of the corresponding area of cluster k. Hence, the bonus reward should be positively correlated with R_k/N_k from the quality perspective.

With the above intuition, we propose a clustering-based bonus reward $b: \mathcal{S} \to \mathbb{R}$ to integrate both novelty and quality of the neighboring area of the current state s, which is defined as follows:

$$b(s) = \eta \times \frac{1}{N_{\phi(s)}} + (1-\eta) \times \frac{R_{\phi(s)}}{N_{\phi(s)}}. \tag{5}$$

Here, $\eta \in (0,1)$ is the coefficient to balance the trade-off between novelty and quality.

The first term in Eq. (5) reflects the novelty of the state, which is negatively correlated with the number of states in the allocated cluster. A larger $N_{\phi(s)}$ will result in a smaller bonus reward $b(s)$. This will guide the agent to explore novel areas corresponding to clusters with fewer visits (exploration), which is reasonable. Moreover, the first term in Eq. (5) can also be substituted with the bonus rewards from existing exploration strategies, as these bonus rewards similarly reflect the novelty of states. The second term in Eq. (5) reflects the quality of the state. The cluster with a higher average reward (higher quality) will be more likely to be explored, which is also reasonable. Hence, the clustering-based bonus reward function defined in Eq. (5) is intuitively reasonable, and it can seamlessly integrate both novelty and quality into the same bonus reward function.

Figure 1 compares clustering-based bonus rewards (where η is set to 0.5 as an intuitive example) and clustering-based bonus rewards with novelty alone (where η is set to 1.0). Here, the states have been divided into $K = 5$ clusters, shown in Fig. 1(a). We observe that the areas corresponding to Cluster 3 (high quality) and Cluster 5 (high novelty) are worthy of exploration for the agent. In Fig. 1(b), we find that clustering-based bonus rewards with novelty alone might cause the agent to overlook the area corresponding to Cluster 3 (high quality). Our method, which integrates both novelty and quality of states into clustering-based bonus rewards, could ensure that the agent simultaneously focuses on the area corresponding to Cluster 3 and Cluster 5, as shown in Fig. 1(c).

Finally, the agent will adopt $\{(s_i, a_i, r_i + \beta b(s_i))\}_{i=1}^N$ to update the policy (perform exploration), where $\beta \in \mathbb{R}^+$ is the bonus coefficient. Many *policy updating algorithms*, such as trust region policy optimization (TRPO) [58], can be adopted. Algorithm 1 briefly presents the learning framework of CRL, using the clustering-based bonus reward defined in Eq. (5). We can see that CRL is actually a general framework, and we can get different variants of CRL by taking different clustering algorithms and different policy updating algorithms. We can also obtain various variants by combining CRL with existing exploration strategies that use the novelty of states as bonus rewards to guide the agent's exploration. Please note that $r_i + \beta b(s_i)$ is only used for training Algorithm 1. The performance evaluation (test) is measured without $\beta b(s_i)$, and our method can be directly compared with existing RL methods without extra bonus rewards.

Algorithm 1 Framework of clustered rein-forcement learning (CRL) using clustering-based bonus rewards defined in Eq. (5)

Initialize the number of clusters K, bonus coefficient β, trade-off coefficient η;
Initialize policy π_0;
for iteration $j = 0, \ldots, J-1$ **do**
  Collect a set of state-action samples $\{(s_i, a_i, r_i)\}_{i=1}^N$ with policy π_j;
  Cluster the states $\{s_i\}_{i=1}^N$ with $f: \mathcal{S} \to C$, where $C = \{C_k\}_{k=1}^K$ and f is some *clustering algorithm*;
  Compute the cluster assignment for each state $\phi(s_i) = \arg\min_k \|s_i - C_k\|, \forall i: 1 \leqslant i \leqslant N, k: 1 \leqslant k \leqslant K$;
  Compute the sum of rewards R_k using Eq. (3) and the number of states N_k using Eq. (4), $\forall k: 1 \leqslant k \leqslant K$;
  Compute the bonus $b(s_i)$ using Eq. (5), $\forall i: 1 \leqslant i \leqslant N$;
  Update the policy π_j using rewards $\{r_i + \beta b(s_i)\}_{i=1}^N$ with some *policy updating algorithm*;
end for

4 Experiment

We carry out experiments to evaluate CRL and baselines on four continuous control tasks and six hard-exploration Atari-2600 games.

4.1 Environments

4.1.1 MuJoCo

The rllab benchmark [10] consists of various continuous control tasks to test RL algorithms. We select Mountain Car, Cart-Pole Swing up, Double Pendulum, and Half-Cheetah with sparse rewards to evaluate our method and other baselines. Heuristic exploration strategies are hard to solve these tasks.

The sparse reward setting of Mountain Car, Cart-Pole Swing up, Double Pendulum, and Half-Cheetah follows previous works [24,36]. In Mountain Car, $\mathcal{S} \subseteq \mathbb{R}^2$, $\mathcal{A} \subseteq \mathbb{R}^1$. The agent receives a reward of $+1$ when the car escapes the valley from the right side; otherwise, the agent receives a reward of 0. In Cart-Pole Swing up, $\mathcal{S} \subseteq \mathbb{R}^4$, $\mathcal{A} \subseteq \mathbb{R}^1$. The agent receives a reward of $+1$ when the cosine of the pole angle is larger than 0.8. Otherwise, the agent receives a reward of 0. In Double Pendulum, $\mathcal{S} \subseteq \mathbb{R}^6$, $\mathcal{A} \subseteq \mathbb{R}^1$. The goal of the Double Pendulum is to swing the pendulum up until it stays upright. The agent receives a reward of $+1$ when the distance between the position of the pendulum and the target position (stay upright) is smaller than 1. Otherwise, the agent receives a reward of 0. In locomotion task Half-Cheetah, $\mathcal{S} \subseteq \mathbb{R}^{20}$, $\mathcal{A} \subseteq \mathbb{R}^6$. The agent needs to learn how to move forward. The agent receives a reward of $+1$ when the x-coordinate is larger than 5. Otherwise, the agent receives a reward of 0.

4.1.2 Arcade learning environment

The Arcade Learning Environment (ALE) [47] is a commonly used benchmark for RL algorithms. It provides a wide variety of video games with high-dimensional state spaces. As in [24], we select six Atari-2600 games from ALE for evaluation. They are Freeway, Frostbite, Gravitar, Montezuma's Revenge[1], Solaris, and Venture. According to the taxonomy in [34], these games are classified into the hard exploration category because they are hard for heuristic exploration strategies.

In [59], the authors have investigated different features to represent the frames of games, including raw pixels, random features, and so on. The random feature is a simple but effective feature representation. We use a randomly initialized and fixed embedding network to get random features of the frames of games. Basic Abstraction of the ScreenShots (BASS) feature [24,47] is adopted as a generic feature representation for all Atari-2600 games. In our experiments with Atari-2600 games, we evaluate our method using raw pixels, random features, and BASS features for clustering and counting.

4.2 Baselines

For baselines, we choose TRPO [58], Hash[2] [24], RND[3] [41], and NovelD[4] [42]. We choose Hash as one of the baselines due to its similarity to our method, enabling a meaningful comparison. We include TRPO [58] as one of the baselines since it is adopted by Hash as its policy updating algorithm. We also include RND and NovelD as baselines because they represent the state-of-the-art exploration strategies [60] in recent years. Furthermore, we choose VIME[5] [36] as one of the baselines for continuous control tasks.

TRPO [58] is a classic policy updating algorithm, which uses trust region to guarantee the monotonic improvement of policy and can handle both discrete and continuous action space. Furthermore, this method is not too sensitive to hyper-parameters. TRPO adopts a Gaussian noise as a heuristic exploration strategy.

Hash [24] is a generalization of the classic count-based method for high-dimensional and continuous state spaces. For a fair comparison, we use SimHash [61] as the hash function following [24]. Hash uses raw pixels as the feature representation. Hash_{RF} denotes the variant of Hash using the random feature for counting. Hash_{BASS} denotes the variant of Hash using the BASS feature for counting.

RND [41] encourages the agent to explore novel areas of the environment. It proposes to use the difference between a random fixed target network and a trainable predictor network to evaluate the novelty of states.

NovelD [42] is a state-of-the-art exploration strategy, especially for hard exploration tasks. Based on RND [41], NovelD tries to weigh every novel area equally and sufficiently help the agent explore multiple novel areas.

VIME [36] is an intrinsic-motivation exploration strategy, which seeks out unexplored state-action regions by maximizing the information gain of the agent's belief of environments. Here, we select VIME to compare with our method in continuous control tasks because this method only supports continuous action space and performs well on various robotic problems.

CRL is a general framework that can adopt many different *policy updating (optimization) algorithms* to get different variants. To ensure a fair comparison, CRL employs the same policy updating algorithm as the corresponding baseline. We denote our method as CRL when using Eq. (5) to compute bonus rewards. Moreover, as discussed in Section 3.3, our work can be combined with existing exploration strategies to further boost their performance, such as Hash, RND, and NovelD, where the bonus rewards of existing exploration strategies also reflect the novelty of states. Specifically, we use the bonus rewards computed by existing exploration strategies to replace the first term in Eq. (5). When combined with Hash, RND, and NovelD, the resulting variants of CRL are called CRL-Hash, CRL-RND, and CRL-NovelD, respectively. We use raw features for clustering by default.

---

[1] Montezuma's Revenge is abbreviated as Montezuma.
[2] Hash is trained by using the code provided by its authors.
[3] We trained RND for Atari-2600 games using the code from its authors. However, as the original paper does not cover MuJoCo as an environment, we implement RND for MuJoCo by ourselves.
[4] As the original paper does not provide the code for MuJoCo and Atari-2600 games, we implement NovelD for MuJoCo and Atari-2600 games by ourselves.
[5] VIME is trained by using the code provided by its authors.

When we use random features or BASS features for clustering, we use subscripts to indicate.

4.3 Issues of exploration only based on novelty

In this section, we use Hash to empirically illustrate the issues of exploration only based on novelty. For the environment, we choose a simple environment, Mountain Car. A snapshot of Mountain Car is shown in Fig. 2. When the agent masters the solution of Mountain Car, the mean average return (MAR) converges to 1. Please note that, TRPO is bound to fail and obtains a zero return at the end of each episode.

For Hash, the longer the hash code is, the better the agent can distinguish the novelty of states. This is because states assigned to the same hash code would have a stronger similarity [24], intuitively. Figure 3(a) shows that the length of hash codes d and the total number of novel states (visited hash codes) N_{Novelty} are positively correlated, confirming the intuition mentioned above. Better performance is expected if the agent uses a longer hash code, as it can help the agent to distinguish the novelty of states better. Figure 3(b) shows the impact of the length of hash codes d on the performance of Hash when the bonus coefficient β is set to 0.1. We can find that a longer hash code might lead to worse performance, which contradicts the aforementioned expectation. Hash fails because the states with relatively high novelty provide the agent with relatively high bonus rewards, diverting the agent's attention. This means that only using novelty for exploration is not enough, which motivates us to design new exploration strategies like CRL.

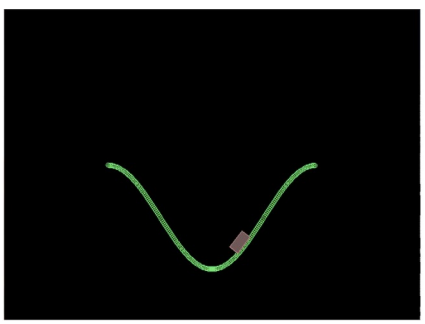

Fig. 2 A snapshot of Mountain Car

4.4 Demonstration of CRL on Mountain Car

Since the dimension of Mountain Car's state space is only 2 and each dimension of Mountain Car's state space has its physical meaning, we use Mountain Car as the test environment to illustrate how CRL uses clustering-based bonus rewards to guide the agent to perform efficient exploration. This version of Mountain Car is referred to as the coordinate-based Mountain Car. To better illustrate the CRL's ability to work in the large state space, we also use a pixel-based Mountain Car [62] as the test environment and the dimension of the state space is 600. On pixel-based Mountain Car, CRL uses the pixel representation of the states for clustering and then computing bonus rewards. Please note that other parts of the training phase are the same. Here, we set $K = 32$, $\beta = 0.01$ and $\eta = 0.1$. For clarity, we denote states and cluster centers in iteration j as $\{s_i^j\}_{i=1}^N$ and $\{C_{k^{(j)}}\}_{k=1}^{32}$, respectively. We allocate each state to its nearest cluster based on the cluster assignment function and the cluster assignment function in iteration j can be detailedly written as $\phi(s_i^j) = \arg\min_{k^{(j)}} \|s_i^j - C_{k^{(j)}}\|, \forall i : 1 \leqslant i \leqslant N, k^{(j)} : 1^{(j)} \leqslant k^{(j)} \leqslant K^{(j)}$. Based on Eqs. (3) and (4), we can get $N_{k^{(j)}}$ and $R_{k^{(j)}}$ to generate the clustering-based bonus rewards for each state. To better illustrate the effect of the clustering-based bonus reward on exploration, we propose $\Delta N_{k^{(j)}}$ and $\Delta R_{k^{(j)}}$ as monitoring indicators. $\Delta N_{k^{(j)}}$ denotes the change in the number of states in the same area corresponding to cluster $k^{(j)}$, and $\Delta R_{k^{(j)}}$ denotes the change in the sum of rewards in the same area corresponding to cluster $k^{(j)}$. The formulas of $\Delta N_{k^{(j)}}$ and $\Delta R_{k^{(j)}}$ are shown as follows:

$$\Delta N_{k^{(j)}} = \sum_{i=1}^{N} \mathbb{I}(\psi(s_i^{j+1}) = k^{(j)}) - \sum_{i=1}^{N} \mathbb{I}(\phi(s_i^j) = k^{(j)}), \tag{6}$$

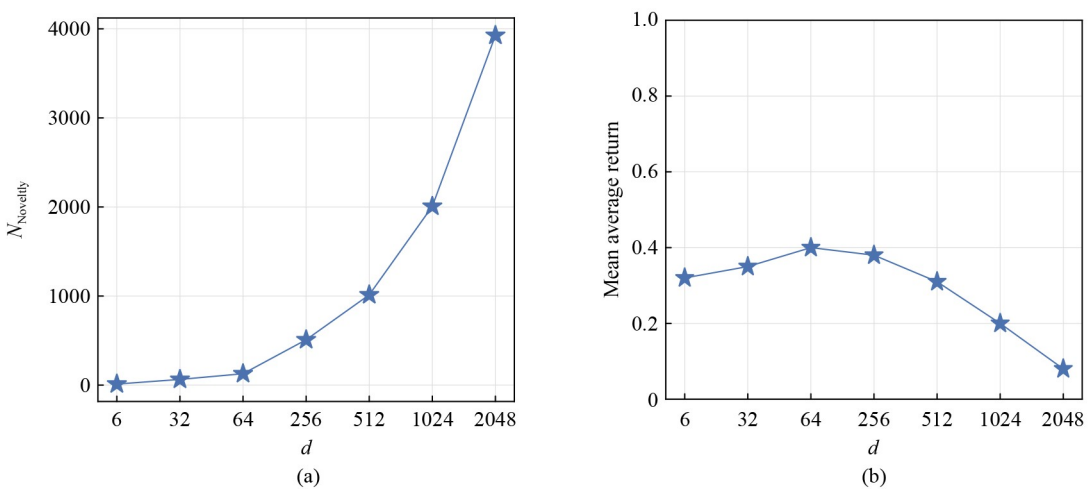

Fig. 3 (a) The total number of novel states N_{Novelty} (y-axis) of Hash on Mountain Car with different lengths of hash codes d (x-axis) when the bonus coefficient is set to 0.1; (b) the mean average return (y-axis) of Hash on Mountain Car with different lengths of hash codes (x-axis) when the bonus coefficient is set to 0.1. The results are averaged over 100 random seeds

$$\Delta R_{k^{(j)}} = \sum_{i=1}^{N} r_i^{j+1} \mathbb{I}(\psi(s_i^{j+1}) = k^{(j)}) - \sum_{i=1}^{N} r_i^{j} \mathbb{I}(\phi(s_i^{j}) = k^{(j)}), \quad (7)$$

where $\psi(s_i^{j+1}) = \arg\min_{k^{(j)}} \|s_i^{j+1} - C_{k^{(j)}}\|, \forall i : 1 \leqslant i \leqslant N, k^{(j)} : 1^{(j)} \leqslant k^{(j)} \leqslant K^{(j)}$. If $\Delta N_{k^{(j)}} \geqslant 0$ or $\Delta R_{k^{(j)}} \geqslant 0$, it indicates that the agent explores the corresponding area of cluster $k^{(j)}$ more extensively in the next iteration, which is intuitive.

4.4.1 Coordinate-based Mountain Car

Figure 4 shows the learning curve of CRL on coordinate-based Mountain Car. We can see that CRL achieves the optimal return at iteration 6. Here, we choose two iterations $j \in \{0, 5\}$ as examples. For clarity, we only focus on two clusters with the top 2 highest bonus rewards among all clusters in iteration j, and these two clusters are marked as clusters $1^{(j)}$ and $2^{(j)}$ respectively, as shown in Figs. 5(a) and 5(b). Please note that in K-means, each state is assigned to its nearest cluster, which is based on the distances between the state and the cluster centers. As a result, even if clusters are visually close to other states, K-means can effectively distinguish clusters and allocate states to their respective nearest clusters.

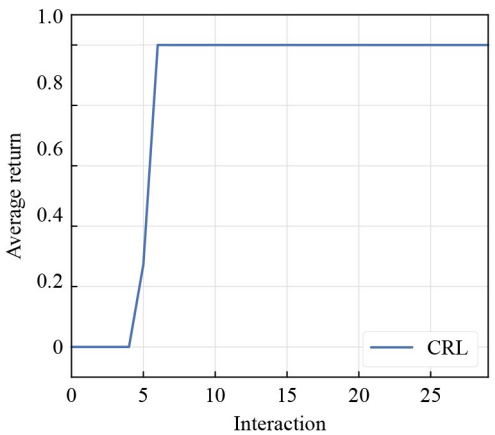

Fig. 4 The training curve of CRL on coordinate-based Mountain Car when $\{K, \beta, \eta\} = \{32, 0.01, 0.1\}$

In iteration 0, we find that $\sum_{i=1}^{N} r_i^0 = 0$, which means that all bonus rewards would only reflect the novelty in the neighboring area of the states. Since the areas corresponding to clusters $1^{(0)}$ and $2^{(0)}$ have been explored fewer times (higher novelty) than other areas, our method gives the states in these two areas high bonus rewards to guide the agent to explore these two novel areas. The states in these two novel areas have a common characteristic: a high absolute horizontal velocity. On Mountain Car, if the agent has a high absolute horizontal velocity, the agent would have a significant possibility of escaping from the valley, which is also the goal of this task. Therefore, allocating high bonus rewards to the states in these two areas and encouraging the agent to explore these two areas is reasonable. After the agent updates its policy, we can get $(\Delta N_{1^{(0)}}, \Delta R_{1^{(0)}}) = (8, 0)$ and $(\Delta N_{2^{(0)}}, \Delta R_{2^{(0)}}) = (16, 0)$ based on Eqs. (6) and (7). It verifies that our proposed bonus rewards can encourage the agent to explore these two novel areas corresponding to clusters $1^{(0)}$ and $2^{(0)}$.

In iteration 5, we observe that $\sum_{i=1}^{N} r_i^5 = 3$, so the clustering-based bonus rewards could simultaneously reflect both novelty and quality in the neighboring area of the states. Figure 5(b) shows the states allocated to clusters $1^{(5)}$ and $2^{(5)}$. We can find that the area corresponding to cluster $1^{(5)}$ has been explored more than some other areas, such as, the area corresponding to cluster $2^{(5)}$. This illustrates that the novelty of this area might not be high. Nonetheless, this cluster has a relatively large cluster reward $R_{1^{(5)}}$, which means this area has high quality. For cluster $2^{(5)}$, despite its cluster reward $R_{2^{(5)}}$ being 0, the area corresponding to cluster $2^{(5)}$ has been explored less than other clusters (high novelty). Since our method utilizes the novelty and quality in the neighboring area of states to guide the agent's exploration, we allocate comparably high bonus rewards to the states within these two areas. Furthermore, we can find that the goal state is in the area corresponding to cluster $1^{(5)}$ and states in the area

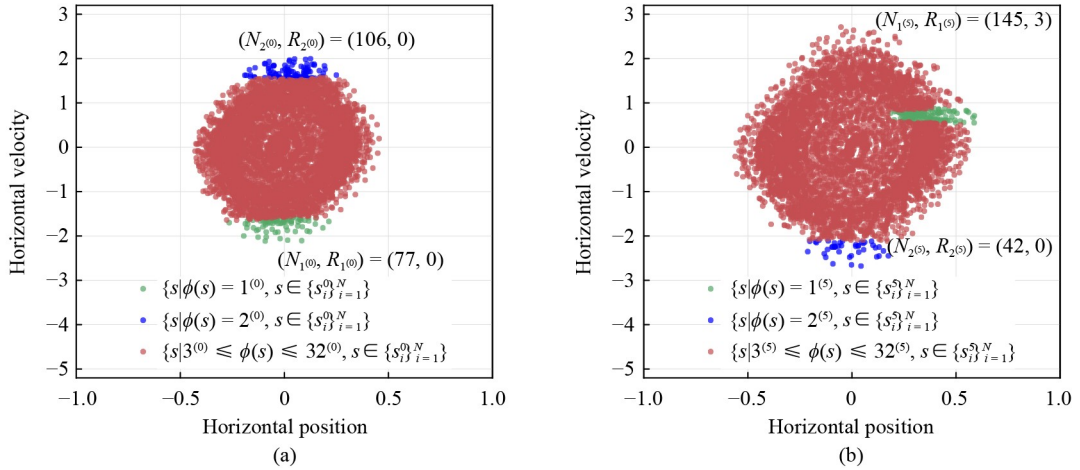

Fig. 5 Coordinate-based Mountain Car. (a) All dots represent states collected in iteration 0. Green dots represent states allocated to cluster $1^{(0)}$ and blue dots represent states allocated to cluster $2^{(0)}$; (b) all dots represent states collected in iteration 5. Green dots represent states allocated to cluster $1^{(5)}$ and blue dots represent states allocated to cluster $2^{(5)}$. The x-coordinate is the agent's horizontal position, and the y-coordinate is the agent's horizontal velocity

corresponding to cluster $2^{(5)}$ tend to exhibit high absolute horizontal velocity. This implies that exploring these areas can assist the agent in achieving the goal of Mountain Car. After the agent collects samples with the updated policy π_6, we find that the agent has mastered a good policy. We also get $(\Delta N_{1^{(5)}}, \Delta R_{1^{(5)}}) = (28, 10)$ and $(\Delta N_{2^{(5)}}, \Delta R_{2^{(5)}}) = (120, 0)$, all of which are greater than or equal to 0. It verifies again that our method can use the bonus rewards to reflect the novelty and quality to assist the agent in exploration.

4.4.2 Pixel-based Mountain Car

Figure 6 shows the learning curve of CRL on pixel-based Mountain Car. We can see that CRL achieves the optimal return at iteration 15. Here, we choose two iterations $j \in \{10, 14\}$ as examples. We also only focus on two clusters with the top 2 highest bonus rewards among all clusters, as shown in Figs. 7(a) and 7(b). Due to the high-dimensional nature of states, we choose to plot the corresponding coordinates of states for visibility. Note that pixel features are employed for clustering and computing clustering-based bonus rewards. This might result in clusters appearing visually coupled to states from other clusters.

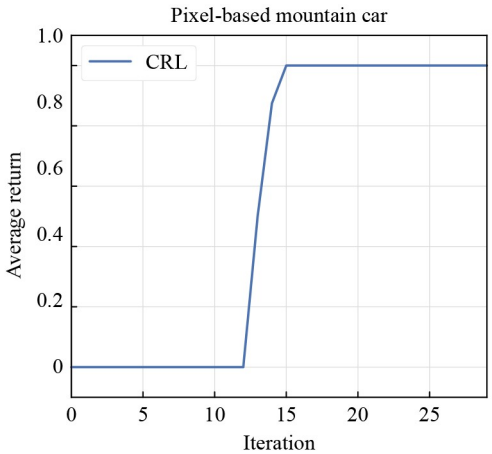

Fig. 6 The training curve of CRL on pixel-based Mountain Car when $\{K, \beta, \eta\} = \{32, 0.01, 0.1\}$

In iteration 10, we observe that $\sum_{i=1}^{N} r_i^{10} = 0$, so all clustering-based bonus rewards would only reflect the novelty in the neighboring area of the states. Our method gives the states in the areas corresponding to clusters $1^{(10)}$ and $2^{(10)}$ high bonus rewards to guide the agent to explore these two novel areas. These areas have been explored fewer times than other areas. The states in these two novel areas share a common characteristic: a high absolute horizontal position. Here, the high absolute horizontal position for the agent means that the agent has a significant possibility of reaching the goal position. Hence, encouraging the agent to explore these two areas by allocating high bonus rewards to the states in these two areas is reasonable. After the agent updates its policy, the changes in the number of visits and the sum of rewards for clusters $1^{(10)}$ and $2^{(10)}$ are $(\Delta N_{1^{(10)}}, \Delta R_{1^{(10)}}) = (95, 0)$ and $(\Delta N_{2^{(10)}}, \Delta R_{2^{(10)}}) = (83, 0)$, calculated using Eqs. (6) and (7). It verifies that our proposed bonus rewards successfully encourage the agent to explore the novel areas corresponding to clusters $1^{(10)}$ and $2^{(10)}$.

As for iteration 14, we observe that $\sum_{i=1}^{N} r_i^{14} = 14$, indicating that clustering-based bonus rewards could simultaneously reflect both novelty and quality in the neighboring area of the states. By observing $(N_{1^{(14)}}, R_{1^{(14)}})$, we can find that the area corresponding to cluster $1^{(14)}$ exhibits relatively high quality and relatively low novelty. Additionally, the goal state is located within the area corresponding to cluster $1^{(14)}$. In the case of cluster $2^{(14)}$, we can find that the area corresponding to this cluster has a relatively high novelty. Although this area is close to the goal state, focusing heavily on exploring this area may not substantially aid in achieving the goal state. That is to say, if exploration strategies focus on the novelty of states, the area corresponding to cluster $2^{(14)}$ could potentially divert the agent's attention, leading to the risk of overlooking the high quality area associated with cluster $1^{(14)}$. However, our method utilizes both the novelty and quality in the neighboring area of states to guide the agent's exploration. Consequently, the agent still pays attention to the area corresponding to cluster

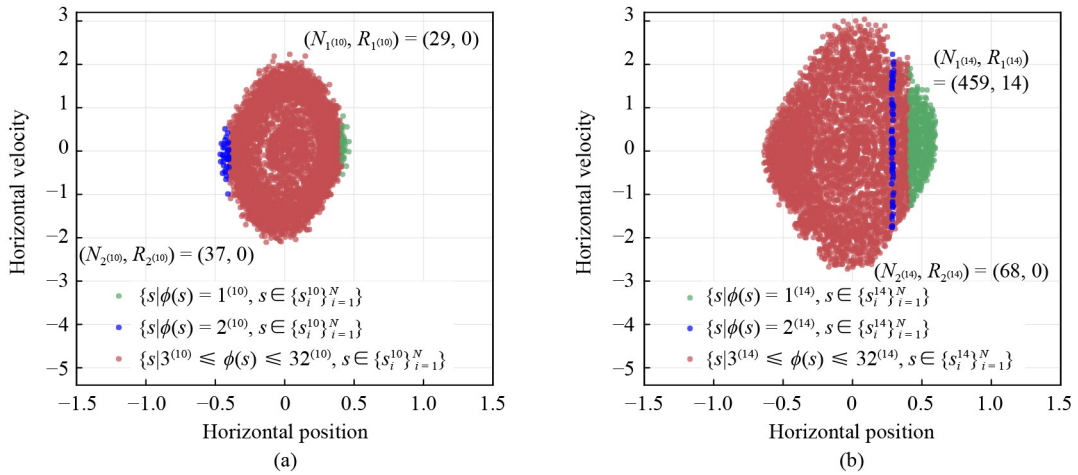

Fig. 7 Pixel-based Mountain Car. (a) All dots represent states collected in iteration 10. Green dots represent states allocated to cluster $1^{(10)}$ and blue dots represent states allocated to cluster $2^{(10)}$; (b) all dots represent states collected in iteration 14. Green dots represent states allocated to cluster $1^{(14)}$ and blue dots represent states allocated to cluster $2^{(14)}$. The x-coordinate is the agent's horizontal position, and the y-coordinate is the agent's horizontal velocity

$1^{(14)}$ and the area corresponding to cluster $2^{(14)}$ simultaneously. Upon collecting samples using the updated policy π_{15}, we observe that the agent has successfully mastered a good policy. We also observe $(\Delta N_{1^{(14)}}, \Delta R_{1^{(14)}}) = (40, 6)$ and $(\Delta N_{2^{(14)}}, \Delta R_{2^{(14)}}) = (11, 0)$, all of which are larger than or equal to 0. This once again verifies that our method utilizes the bonus rewards to reflect novelty and quality, thereby assisting the agent in exploration. The experimental results on the pixel-based Mountain Car also illustrate that states can be well-clustered and our method can perform well on a large state space.

4.5 Performance on MuJoCo

On MuJoCo, the hyper-parameter setting of TRPO and Hash is the same as that in [24], and the hyper-parameter setting for VIME is the same as that in [36]. As the original papers of RND and NovelD do not cover the MuJoCo environment, we also employ TRPO [58] as their policy updating algorithm on MuJoCo in this paper. We configure the predictor network as a multilayer perceptron (MLP) with 128 outputs. This MLP comprises a single hidden layer with 264 hidden units. The rectified linear unit (ReLU) [63] is used as its activation function. The architecture of the target network is the same as that of the predictor network. The time horizon T is set to 500 for all tasks. For the policy updating algorithm, we set the batch size to $5K$ and the step size to 0.01. For CRL and its variants, we set $\beta = 0.01$ for all tasks and the values of η are documented in Table 1. The number of clusters K is set to 32 for Mountain Car, Cart-Pole Swing up and Half-Cheetah, and K is set to 64 for Double Pendulum.

Table 1 summarizes the results of our method and all baselines. We observe that both CRL and CRL-RND achieve the best performance on Mountain Car. CRL-Hash achieves the best performance on Cart-Pole Swing up, while CRL-NovelD achieves the best performance on Double Pendulum. On Half-Cheetah, we can find that CRL outperforms TRPO, Hash, RND, and NovelD, which implies that the agent of our method has the ability to move forward, although the performance of our method is not as good as VIME. Furthermore, when combined with other exploration strategies, the variants of CRL can perform better than the corresponding exploration strategies. This verifies that using novelty and quality of states can effectively guide the agent to perform efficient exploration. Figure 8 shows the training curves of TRPO, Hash, VIME, and CRL on Mountain Car, Cart-Pole Swing up, Double Pendulum, and Half-Cheetah. We can find that on Mountain Car, our method is the first to reach the goal state and master a good policy finally.

Table 1 The results of our method and all baselines on four continuous control tasks over 5 random seeds. For our method, the numbers in parentheses indicate the values of η. Boldface numbers are the best results among all methods

Method	Mountain Car	Cart-Pole Swing up	Half-Cheetah	Double Pendulum
TRPO [58]	0	145.16	0	294.71
VIME [36]	**1**	256.04	**19.46**	298.77
CRL	**1** (0.1)	346.58 (0.1)	2.06 (0.1)	375.51 (0.1)
Hash [24]	0.40	268.01	0	279.14
CRL-Hash	0.40 (0.5)	**356.15** (0.9)	0 (0.9)	367.42 (0.5)
RND [41]	0.65	310.96	0	368.81
CRL-RND	**1** (0.5)	331.52 (0.5)	0 (0.9)	381.02 (0.5)
NovelD [42]	0.27	326.38	0	366.96
CRL-NovelD	0.38 (0.25)	336.39 (0.5)	0 (0.9)	**392.23** (0.5)

4.6 Performance on Atari-2600 games from ALE

On Atari-2600 games from ALE, the agent is trained for 200M frames in all experiments and the agent selects an action every four frames. The policy and baseline of the policy updating algorithm take every four frames as input. The last frame of every four frames is used for clustering and counting. For the random feature, the architecture is illustrated in Table 2. It contains three convolutional layers (conv 1−3) and one fully-connected layer (full 4). The activation function for all convolutional layers is the Leaky RElectification Linear Unit (Leaky RELU) [64]. After initialization, the parameters of the random feature are fixed during the whole training phase. For the BASS feature, the hyper-parameters are the same as those in [24].

For Hash and its variants, the length of hash codes is set to 64, which is sufficient for encoding all states in the training process and the bonus coefficient is set to 0.01. Other hyper-parameters are the same as those in [24]. For hyper-parameters used in CRL and its variants (i.e., K used in the clustering algorithm, η used in the bonus rewards, and the bonus coefficient β), we conduct a random search with a coarse parameter range for each game. Regarding CRL and CRL-Hash, we choose $K = 16$, $\beta = 0.01$ for Frostbite, Gravitar and Venture. For Freeway, we choose $K = 16$, $\beta = 0.1$. For Montezuma's Revenge, we choose $K = 264$, $\beta = 0.1$. For Solaris, we choose $K = 16$, $\beta = 0.01$. The choices of η can be found in Table 3. For the variants of CRL that combine RND or NovelD, we use the same choices of K in different games as those in CRL. The remaining hyper-parameter settings match those presented in the corresponding original paper. And the choices of η are documented in Table 3. As for the variants of CRL with different feature representations, their hyper-parameter settings are the same as those of CRL across games.

The mean average return of our methods and baselines after training for 200M frames over 5 random seeds are summarized in Table 3. We find that CRL achieves the best performance on Freeway, Frostbite, and Solaris, and CRL-RND achieves the best performance on Gravitar, Montezuma, and Venture. Combining our method with existing exploration strategies, we observe enhanced performance compared to the corresponding individual exploration strategies in most games. This verifies that utilizing novelty and quality of states simultaneously is helpful in guiding the agent's exploration and our method is flexible to be combined with other

Table 2 Configuration of the network for the random feature on Atari-2600 games

Layer	Configuration
conv 1	filter $32 \times 8 \times 8$, stride 4, Leaky RELU
conv 2	filter $64 \times 4 \times 4$, stride 2, Leaky RELU
conv 3	filter $64 \times 3 \times 3$, stride 1, Leaky RELU
full 4	256 units

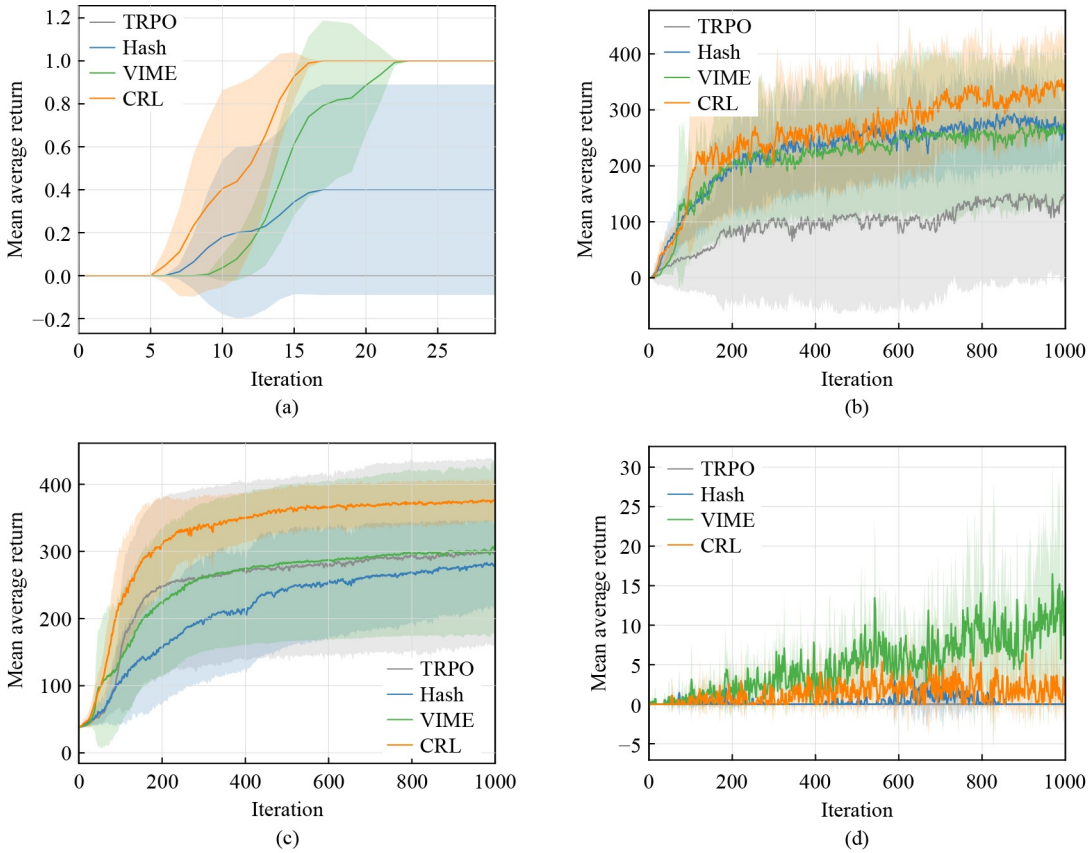

Fig. 8 Training curves of TRPO, Hash, VIME, and CRL on Mountain Car, Cart-Pole Swing up, Double Pendulum and Half-Cheetah. The results are averaged over 5 random seeds. The solid line represents the mean average return, and the shaded area represents one standard deviation. On Mountain Car and Half-Cheetah, the training curves of TRPO coincide with the x-axis. (a) Mountain Car; (b) Cart-Pole Swing up; (c) Double Pendulum; (d) Half-Cheetah

Table 3 The mean average return of our method and baselines after training for 200M frames on six hard-exploration Atari-2600 games over 5 random seeds. For our method, the numbers in parentheses indicate the values of η. The Boldface numbers are the best results among all methods

Method	Freeway	Frostbite	Gravitar	Montezuma	Solaris	Venture
TRPO [58]	17.55	1229.66	500.33	0	2110.22	283.48
CRL	**30.80** (0.75)	**4337.98** (0.1)	552.46 (0.1)	0 (0.75)	**3672.55** (0.5)	312.40 (0.1)
Hash [24]	22.29	2954.10	577.47	0	2619.32	299.61
CRL-Hash	28.38 (0.75)	4148.90 (0.1)	585.79 (0.1)	0 (0.75)	2741.48 (0.5)	328.50 (0.1)
RND [41]	21.52	2837.70	867.30	2188.80	765.47	966.00
CRL-RND	20.85 (0.9)	4076.60 (0.9)	**1002.40** (0.75)	**2453.30** (0.5)	1021.60 (0.5)	**981.20** (0.9)
NovelD [42]	21.39	3476.46	677.90	1744.80	975.52	283.60
CRL-NovelD	19.97 (0.9)	3520.06 (0.9)	971.50 (0.5)	2323.40 (0.5)	980.16 (0.5)	498.60 (0.9)

exploration strategies. Given the large state space of Atari 2600 games, the good performance of our method in these games further demonstrates its capability in large state space.

Furthermore, since Hash [24] is the most similar method to our method, we conduct a comparative analysis with Hash when utilizing distinct features as the feature representation, which follows Hash [24]. Table 4 summarizes the mean average return of CRL and Hash with different feature representations after training for 200M frames over 5 random seeds. We can find that our method outperforms Hash across the majority of games, regardless of the chosen feature representation. This reaffirms the effectiveness of our method as an exploration strategy. Additionally, we observe that the BASS feature consistently yields superior performance compared to other feature representations in most games, particularly on Montezuma. Figure 9 shows the training curves of TRPO, Hash$_{BASS}$ and CRL$_{BASS}$.

4.7 Sensitivity to hyper-parameters

In this section, we choose Freeway to study the performance sensitivity to hyper-parameters because CRL with different feature representations can obtain good performance on Freeway. Here, the hyper-parameters include K in K-means, bonus coefficient β, and η in the bonus reward. We choose CRL and CRL$_{BASS}$ to study the performance sensitivity to hyper-parameters because CRL is a basic method with the raw feature and the BASS feature is better than other feature representations in most Atari-2600 games.

4.7.1 Sensitivity analysis of K

We investigate the effect of the number of clusters K when

Table 4 The mean average return of CRL and Hash when adopting different features after training for 200M frames on six hard-exploration Atari-2600 games over 5 random seeds

Method	Freeway	Frostbite	Gravitar	Montezuma	Solaris	Venture
Hash [24]	22.29	2954.10	577.47	0	2619.32	299.61
CRL	30.80	4337.98	552.46	0	3672.55	312.40
Hash_{RF} [24]	27.28	5530.79	520.67	0	2470.54	72.30
CRL_{RF}	28.60	4444.63	572.74	0	2891.14	190.18
Hash_{BASS} [24]	32.18	2958.44	524.28	265.16	2372.05	401.08
CRL_{BASS}	31.60	6173.75	602.60	379.68	3397.51	582.69

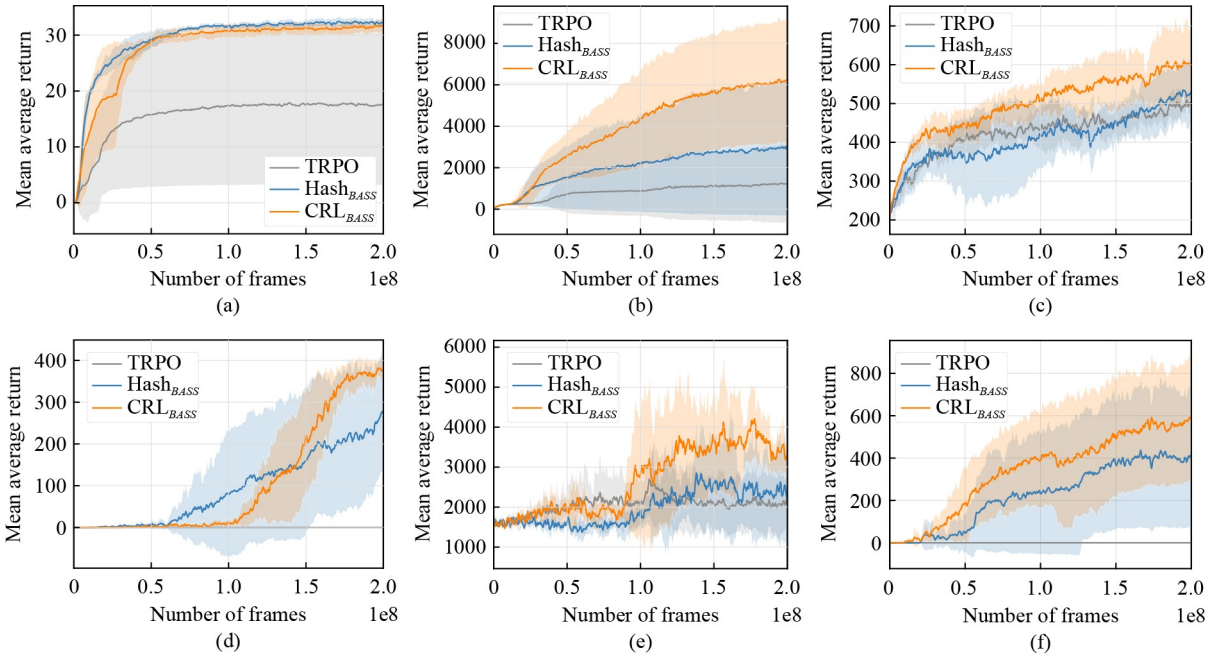

Fig. 9 Training curves of TRPO, Hash_{BASS} and CRL_{BASS} on six hard-exploration Atari-2600 games. The results are averaged over 5 random seeds. The solid line represents the mean average return and the shaded area represents one standard deviation. (a) Freeway; (b) Frostbite; (c) Gravitar; (d) Montezuma's Revenge; (e) Solaris; (f) Venture

$\beta = 0.1$, $\eta = 0.25$, and $\beta = 0.1$, $\eta = 0.75$. On Freeway, we first use the elbow method to find the optimal $K = 10$ for K-means. Please note that $K = 10$ might not be the best choice for CRL. And then we investigate K in a relatively small parameter range around $K = 10$. Here, we choose K from $\{8, 12, 16, 20, 32\}$. The mean average return of different choices of K is summarized in Fig. 10.

For clustering, a larger value of K will divide the state space more precisely, but the statistic of R_k/N_k might become less meaningful. A smaller value of K will mix information from different areas, which might be too coarse for exploration. We can find that the performance is not too sensitive to K in a relatively large range. We can observe that $K = 16$ is a reasonable choice for clustering in Freeway. Hence, in the following section, we investigate the influence of the hyper-parameters β and η when K is set to 16 for Freeway.

Furthermore, we employ the states collected during iteration 0 on Freeway to illustrate whether states from a large state space are well clustered. Please note that we employ the raw pixels as the feature representation, and the raw pixels of states are used for clustering. We subsequently visualize the collected states during iteration 0 and their corresponding

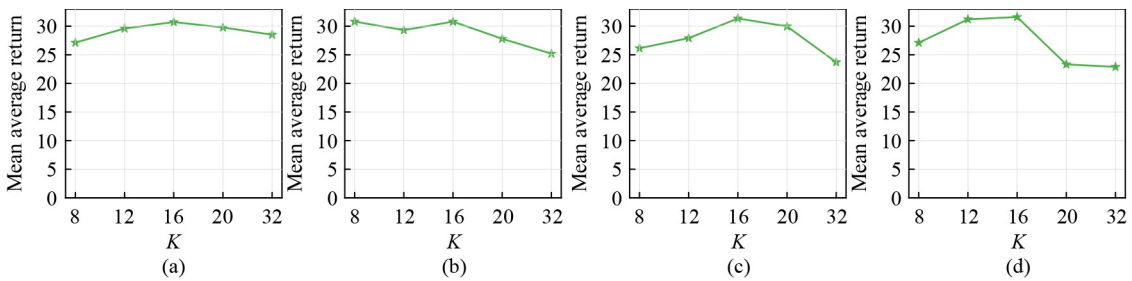

Fig. 10 Sensitivity to K (x-axis) of CRL and CRL_{BASS} on Freeway when $\beta = 0.1$, $\eta = 0.25$ and $\beta = 0.1$, $\eta = 0.75$. The results are averaged over 5 random seeds. The y-axis represents mean average return, which is averaged over 5 random seeds after training for 500 iterations. (a) CRL. β=0.1, η=0.25; (b) CRL. β=0.1, η=0.75; (c) CRL_{BASS}. β=0.1, η=0.25; (d) CRL_{BASS}. β=0.1, η=0.75

cluster centers by t-SNE [65], as shown in Fig. 11. Here, we choose $K = 16$. We observe that the states have been effectively grouped into clusters. It is worth noting that certain clusters are intertwined with others because the clustering is conducted within a high-dimensional space, but our visualization is constrained to a 2D space.

4.7.2 Sensitivity analysis of β and η

We study the influence of β and η on Freeway and compare the performance of CRL and CRL$_{BASS}$ with the coefficient $\beta \in \{0, 0.01, 0.1, 1\}$ and $\eta \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$ on Freeway when K is fixed to 16. The performance of different choices of $\{\beta, \eta\}$ is summarized in Table 5.

Table 5 summarizes the results of different β when η is chosen from $\{0.1, 0.25, 0.5, 0.75, 0.9\}$. Please note that when $\beta = 0$, CRL and CRL$_{BASS}$ will be degenerated to TRPO. We can find that as long as β is not too large, the performance with $\beta > 0$ is better than that with $\beta = 0$. This implies that the bonus reward does provide useful information for better exploration. Furthermore, the performance will deteriorate when β is too large, which is reasonable because a large β might make the bonus reward dominate the true reward from the environment.

The value of η reflects the degree of exploration for novel areas, while the value of $1-\eta$ reflects the degree of exploration for quality areas. In Table 5, for $\beta = 0.01$, the performances of CRL and CRL$_{BASS}$ are roughly concave, peaking at $\eta = 0.25$ and $\eta = 0.5$, respectively. For $\beta = 0.1$, the performances are robust when $\eta \in \{0.25, 0.5, 0.75\}$. With $\beta = 1$, the performance roughly peaks at $\eta = 0.25$. The above results about η on Freeway illustrate that using both novelty and quality in the neighboring area of the current state is better than using only one kind of information, either novelty or quality, of states to guide the exploration of the agent.

In all, our method, CRL, with different hyper-parameter settings, can outperform TRPO and Hash in most cases, which verifies that our method can provide the agent with a better exploration direction.

Furthermore, we have illustrated that only using novelty for exploration is insufficient by evaluating Hash with different hyper-parameter settings on Mountain Car in Section 4.3. For completeness, we evaluate our method CRL on Mountain Car in different settings to illustrate that the bonus reward reflecting the novelty and quality of states can provide the agent with a better exploration direction. In Table 6, we report the results of CRL with different K chosen from $\{16, 24, 32, 40\}$ when β is chosen from $\{0.01, 0.1\}$ and η is chosen from $\{0.25, 0.75\}$. We find that our method CRL can master the solution of Mountain Car (converge to 1) in all hyper-parameter settings. This verifies again that using both novelty and quality in the neighboring area of the current state is better than using novelty to provide the agent with the exploration direction.

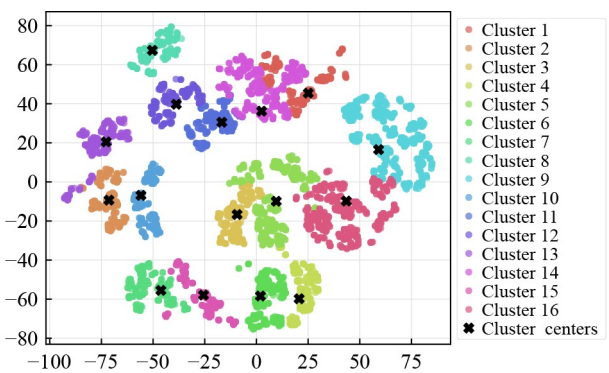

Fig. 11 Visualization of states collected during iteration 0 on Freeway, along with their corresponding cluster centers for $K = 16$

4.8 Time cost analysis of CRL

In this section, we experimentally analyze the time cost of CRL. Here, we still choose Freeway as the test environment for illustration. The hyper-parameter settings of CRL are the same as those in Section 4.6. As shown in Fig. 12, we report the mean average return with respect to wall-clock time for CRL and Hash on Freeway during the first 50 iterations using 12-core 12-thread Intel Xeon CPU E5-2620 v2. We can find that the performance of our method is better than that of Hash in the mean average return within the same amount of time.

Table 5 Effect of β and η using CRL and CRL$_{BASS}$ on Freeway when β is chosen from $\{0, 0.01, 0.1, 1\}$, η is chosen from $\{0.1, 0.25, 0.5, 0.75, 0.9\}$, and K is set to 16. The results are averaged over 5 random seeds after 500 iterations

MAR	CRL					MAR	CRL$_{BASS}$				
	η						η				
β	0.1	0.25	0.5	0.75	0.9	β	0.1	0.25	0.5	0.75	0.9
0	17.55	17.55	17.55	17.55	17.55	0	17.55	17.55	17.55	17.55	17.55
0.01	25.60	27.67	24.21	25.09	24.38	0.01	23.52	24.42	27.15	24.92	23.54
0.1	30.10	30.72	28.43	30.80	29.66	0.1	30.07	31.35	29.89	31.60	23.28
1	25.19	28.52	28.75	23.79	24.35	1	23.53	29.15	22.85	22.82	24.22

Table 6 The mean average return of CRL on Mountain Car with different settings. The results are averaged over 5 random seeds

MAR	$\beta = 0.01$		$\beta = 0.1$	
K	$\eta = 0.25$	$\eta = 0.75$	$\eta = 0.25$	$\eta = 0.75$
16	1.0	1.0	1.0	1.0
24	1.0	1.0	1.0	1.0
32	1.0	1.0	1.0	1.0
40	1.0	1.0	1.0	1.0

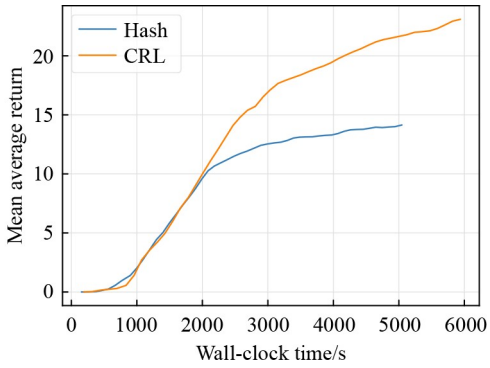

Fig. 12 Mean average return of CRL and Hash with respect to wall-clock time on Freeway during the first 50 iterations

5 Conclusion

In this paper, we propose a simple yet effective RL framework, CRL, for efficient exploration. By using clustering, CRL provides a general framework to adopt both novelty and quality in the neighboring area of the current state for exploration. CRL can also be combined with existing exploration strategies to further improve their performance. Experiments on four continuous control tasks and six Atari-2600 games show that our method can outperform other state-of-the-art methods to achieve the best performance in most cases.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Gtant No. 62192783) and Fundamental Research Funds for the Central Universities (No. 020214380108).

Competing interests The authors declare that they have no competing interests or financial conflicts to disclose.

References

1. Sutton R S, Barto A G. Reinforcement Learning: an Introduction. Cambridge: MIT Press, 1998
2. Mnih V, Kavukcuoglu K, Silver D, Rusu A A, Veness J, Bellemare M G, Graves A, Riedmiller M, Fidjeland A K, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D. Human-level control through deep reinforcement learning. Nature, 2015, 518(7540): 529–533
3. Silver D, Huang A, Maddison C J, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever I, Lillicrap T P, Leach M, Kavukcuoglu K, Graepel T, Hassabis D. Mastering the game of go with deep neural networks and tree search. Nature, 2016, 529(7587): 484–489
4. Lample G, Chaplot D S. Playing FPS games with deep reinforcement learning. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence. 2017, 2140−2146
5. Badia A P, Piot B, Kapturowski S, Sprechmann P, Vitvitskyi A, Guo D, Blundell C. Agent57: Outperforming the Atari human benchmark. In: Proceedings of the 37th International Conference on Machine Learning. 2020, 48
6. Ma X, Li W J. State-based episodic memory for multi-agent reinforcement learning. Machine Learning, 2023, 112(12): 5163–5190
7. Singh B, Kumar R, Singh V P. Reinforcement learning in robotic applications: a comprehensive survey. Artificial Intelligence Review, 2022, 55(2): 945–990
8. Wen Y, Si J, Brandt A, Gao X, Huang H H. Online reinforcement learning control for the personalization of a robotic knee prosthesis. IEEE Transactions on Cybernetics, 2020, 50(6): 2346–2356
9. Lillicrap T P, Hunt J J, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. Continuous control with deep reinforcement learning. In: Proceedings of the 4th International Conference on Learning Representations. 2016
10. Duan Y, Chen X, Houthooft R, Schulman J, Abbeel P. Benchmarking deep reinforcement learning for continuous control. In: Proceedings of the 33rd International Conference on Machine Learning. 2016, 1329−1338
11. Modares H, Ranatunga I, Lewis F L, Popa D O. Optimized assistive human-robot interaction using reinforcement learning. IEEE Transactions on Cybernetics, 2016, 46(3): 655–667
12. Amarjyoti S. Deep reinforcement learning for robotic manipulation-the state of the art. 2017, arXiv preprint arXiv: 1701.08878
13. Xu Y, Fang M, Chen L, Du Y, Zhou J, Zhang C. Perceiving the world: Question-guided reinforcement learning for text-based games. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. 2022, 538−560
14. Ghalandari D, Hokamp C, Ifrim G. Efficient unsupervised sentence compression by fine-tuning transformers with reinforcement learning. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. 2022, 1267−1280
15. Li H, Hu Y, Cao Y, Zhou G, Luo P. Rich-text document styling restoration via reinforcement learning. Frontiers of Computer Science, 2021, 15(4): 154328
16. Yau K L A, Kwong K H, Shen C. Reinforcement learning models for scheduling in wireless networks. Frontiers of Computer Science, 2013, 7(5): 754–766
17. Qin Y, Wang H, Yi S, Li X, Zhai L. A multi-objective reinforcement learning algorithm for deadline constrained scientific workflow scheduling in clouds. Frontiers of Computer Science, 2021, 15(5): 155105
18. Lin Y C, Chen C T, Sang C Y, Huang S H. Multiagent-based deep reinforcement learning for risk-shifting portfolio management. Applied Soft Computing, 2022, 123: 108894
19. Zhang Y, Zhao P, Wu Q, Li B, Huang J, Tan M. Cost-sensitive portfolio selection via deep reinforcement learning. IEEE Transactions on Knowledge and Data Engineering, 2022, 34(1): 236–248
20. Li X, Cui C, Cao D, Du J, Zhang C. Hypergraph-based reinforcement learning for stock portfolio selection. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. 2022, 4028−4032
21. Xu K, Zhang Y, Ye D, Zhao P, Tan M. Relation-aware transformer for portfolio policy learning. In: Proceedings of the 29th International Joint Conference on Artificial Intelligence. 2020, 641
22. Wang Z, Huang B, Tu S, Zhang K, Xu L. DeepTrader: A deep reinforcement learning approach for risk-return balanced portfolio management with market conditions embedding. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence. 2021, 643−650
23. Ouyang L, Wu J, Jiang X, Almeida D, Wainwright C L, Mishkin P, Zhang C, Agarwal S, Slama K, Ray A, Schulman J, Hilton J, Kelton F, Miller L, Simens M, Askell A, Welinder P, Christiano P F, Leike J, Lowe R. Training language models to follow instructions with human feedback. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. 2022
24. Tang H R, Houthooft R, Foote D, Stooke A, Chen X, Duan Y, Schulman J, De Turck F, Abbeel P. #exploration: A study of count-based exploration for deep reinforcement learning. In: Proceedings of the 31th International Conference on Neural Information Processing Systems. 2017, 2753−2762
25. Qian H, Yu Y. Derivative-free reinforcement learning: a review. Frontiers of Computer Science, 2021, 15(6): 156336
26. Chapelle O, Li L. An empirical evaluation of Thompson sampling. In:

Proceedings of the 24th International Conference on Neural Information Processing Systems. 2011, 2249−2257
27. Mnih V, Badia A P, Mirza M, Graves A, Harley T, Lillicrap T P, Silver D, Kavukcuoglu K. Asynchronous methods for deep reinforcement learning. In: Proceedings of the 33rd International Conference on Machine Learning. 2016, 1928−1937
28. Fortunato M, Azar M G, Piot B, Menick J, Hessel M, Osband I, Graves A, Mnih V, Munos R, Hassabis D, Pietquin O, Blundell C, Legg S. Noisy networks for exploration. In: Proceedings of the 6th International Conference on Learning Representations. 2018
29. Plappert M, Houthooft R, Dhariwal P, Sidor S, Chen R Y, Chen X, Asfour T, Abbeel P, Andrychowicz M. Parameter space noise for exploration. In: Proceedings of the 6th International Conference on Learning Representations. 2018
30. Osband I, Blundell C, Pritzel A, Van Roy B. Deep exploration via bootstrapped DQN. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. 2016, 4033−4041
31. Osband I, Van Roy B, Russo D J, Wen Z. Deep exploration via randomized value functions. Journal of Machine Learning Research, 2019, 20(124): 1–62
32. Kearns M, Singh S. Near-optimal reinforcement learning in polynomial time. Machine Learning, 2002, 49(2−3): 209−232
33. Brafman R I, Tennenholtz M. R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning. Journal of Machine Learning Research, 2003, 3: 213–231
34. Bellemare M G, Srinivasan S, Ostrovski G, Schaul T, Saxton D, Munos R. Unifying count-based exploration and intrinsic motivation. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. 2016, 1479−1487
35. Ostrovski G, Bellemare M G, Van Den Oord A, Munos R. Count-based exploration with neural density models. In: Proceedings of the 34th International Conference on Machine Learning. 2017, 2721−2730
36. Houthooft R, Chen X, Duan Y, Schulman J, De Turck F, Abbeel P. VIME: variational information maximizing exploration. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. 2016, 1117−1125
37. Stadie B C, Levine S, Abbeel P. Incentivizing exploration in reinforcement learning with deep predictive models. 2015, arXiv preprint arXiv: 1507.00814
38. Pathak D, Agrawal P, Efros A A, Darrell T. Curiosity-driven exploration by self-supervised prediction. In: Proceedings of the 34th International Conference on Machine Learning. 2017, 2778−2787
39. Klyubin A S, Polani D, Nehaniv C L. Empowerment: a universal agent-centric measure of control. In: Proceedings of the IEEE Congress on Evolutionary Computation. 2005, 128−135
40. Fu J, Co-Reyes J D, Levine S. EX2: exploration with exemplar models for deep reinforcement learning. In: Proceedings of the 31th International Conference on Neural Information Processing Systems. 2017, 2577−2587
41. Burda Y, Edwards H, Storkey A J, Klimov O. Exploration by random network distillation. In: Proceedings of the 7th International Conference on Learning Representations. 2019
42. Zhang T, Xu H, Wang X, Wu Y, Keutzer K, Gonzalez J E, Tian Y. NovelD: A simple yet effective exploration criterion. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. 2021, 25217−25230
43. Auer P, Ortner R. Logarithmic online regret bounds for undiscounted reinforcement learning. In: Proceedings of the 19th International Conference on Neural Information Processing Systems. 2006, 49−56
44. Osband I, Russo D, Van Roy B. (More) efficient reinforcement learning via posterior sampling. In: Proceedings of the 26th International Conference on Neural Information Processing Systems. 2013, 3003−3011
45. Ecoffet A, Huizinga J, Lehman J, Stanley K O, Clune J. Go-explore: a new approach for hard-exploration problems. 2019, arXiv preprint arXiv: 1901.10995
46. Ecoffet A, Huizinga J, Lehman J, Stanley K O, Clune J. First return, then explore. Nature, 2021, 590(7847): 580–586
47. Bellemare M G, Naddaf Y, Veness J, Bowling M. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 2013, 47: 253–279
48. Strehl A L, Littman M L. An analysis of model-based interval estimation for Markov decision processes. Journal of Computer and System Sciences, 2008, 74(8): 1309–1331
49. Ortner R. Adaptive aggregation for reinforcement learning in average reward Markov decision processes. Annals of Operations Research, 2013, 208(1): 321–336
50. Barto A G. Intrinsic motivation and reinforcement learning. In: Baldassarre G, Mirolli M, eds. Intrinsically Motivated Learning in Natural and Artificial Systems. Berlin: Springer, 2013, 17−47
51. Berlyne D E. Structure and Direction in Thinking. Hoboken: Wiley, 1965
52. Mannor S, Menache I, Hoze A, Klein U. Dynamic abstraction in reinforcement learning via clustering. In: Proceedings of the 21st International Conference on Machine Learning. 2004
53. Tziortziotis N, Blekas K. A model based reinforcement learning approach using on-line clustering. In: Proceedings of the IEEE International Conference on Tools with Artificial Intelligence. 2012, 712−718
54. Wang T, Gupta T, Mahajan A, Peng B, Whiteson S, Zhang C J. RODE: learning roles to decompose multi-agent tasks. In: Proceedings of the 9th International Conference on Learning Representations. 2021
55. Christianos F, Papoudakis G, Rahman A, Albrecht S V. Scaling multi-agent reinforcement learning with selective parameter sharing. In: Proceedings of the 38th International Conference on Machine Learning. 2021, 1989−1998
56. Mandel T, Liu Y E, Brunskill E, Popovic Z. Efficient Bayesian clustering for reinforcement learning. In: Proceedings of the 25th International Joint Conference on Artificial Intelligence. 2016, 1830−1838
57. Coates A, Ng A Y. Learning feature representations with *K*-means. In: Montavon G, Orr G B, Müller K R, eds. Neural Networks: Tricks of the Trade. 2nd ed. Berlin: Springer, 2012, 561−580
58. Schulman J, Levine S, Moritz P, Jordan M, Abbeel P. Trust region policy optimization. In: Proceedings of the 32nd International Conference on Machine Learning. 2015, 1889−1897
59. Burda Y, Edwards H, Pathak D, Storkey A J, Darrell T, Efros A A. Large-scale study of curiosity-driven learning. In: Proceedings of the 7th International Conference on Learning Representations. 2019
60. Wang K, Zhou K, Kang B, Feng J, Yan S. Revisiting intrinsic reward for exploration in procedurally generated environments. In: Proceedings of the 11th International Conference on Learning Representations. 2023
61. Charikar M S. Similarity estimation techniques from rounding algorithms. In: Proceedings of the 34th Annual ACM Symposium on Theory of Computing. 2002, 380−388
62. Voloshin C, Le H M, Jiang N, Yue Y. Empirical study of off-policy policy evaluation for reinforcement learning. In: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks. 2021
63. Nair V, Hinton G E. Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning. 2010, 807−814
64. Maas A L, Hannun A Y, Ng A Y. Rectifier nonlinearities improve

neural network acoustic models. In: Proceedings of the 30th International Conference on Machine Learning. 2013

65. Van Der Maaten L, Hinton G. Visualizing data using t-SNE. Journal of Machine Learning Research, 2008, 9(86): 2579–2605

Xiao Ma received the BSc degree in information and computing science from Xi'an Jiaotong University, China. She is currently working toward the PhD degree in the Department of Computer Science and Technology, Nanjing University of China. Her research interests are in reinforcement learning, machine learning, and artificial intelligence.

Shen-Yi Zhao received the BSc degree in mathematics from the Nanjing University of China and the PhD degree in computer science from the Nanjing University of China. His research interests are in machine learning, convex optimization, parallel and distributed optimization.

Zhao-Heng Yin received the BSc degree in computer science from the Nanjing University of China and received the MPhil degree in electronic and computer engineering from The Hong Kong University of Science and technology, China. He is currently working toward the PhD degree in the Department of Electrical Engineering and Computer Sciences at the University of California, Berkeley, USA. His research interests include machine learning and robotics.

Wu-Jun Li received the BSc and MEng degrees in computer science from the Nanjing University of China, and the PhD degree in computer science from The Hong Kong University of Science and Technology, China. He started his academic career as an assistant professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. He then joined Nanjing University, China, where he is currently a professor in the Department of Computer Science and Technology. His research interests are in machine learning, big data, and artificial intelligence.