Information Sciences 614 (2022) 206–222

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Consolidation of structure of high noise data by a new noise index and reinforcement learning

Tianyi Huang [a], Zhiling Cai [a], Ruijia Li [a], Shiping Wang [b], William Zhu [a,*]

[a] Institute of Fundamental and Frontier Sciences, University of Electronic Science and Technology of China, Chengdu 610054, China
[b] College of Mathematics and Computer Sciences, Fuzhou University, Fuzhou 350000, China

ARTICLE INFO

Article history:
Received 3 July 2020
Received in revised form 23 March 2022
Accepted 3 October 2022
Available online 7 October 2022

Keywords:
Reinforcement learning
Denoising
Structure of data
Density gradient
Noisy data

ABSTRACT

Data denoising is an essential issue in machine learning and computer vision. However, most existing denoising methods can handle only low-noise data, because it is difficult for these methods to obtain the true structures of high-noise data. An existing partial solution is to consolidate the structures of data by moving each sample to its near high density region. It actually regards all non-noisy samples as noisy ones, but non-noisy samples are important for constructing the structures of high-noise data so they should not be moved. To address this problem, we propose a new denoising method called Denoising by a new Noise index and Reinforcement learning (DNR). Firstly, it detects each noisy sample by its density and the distance between this sample and the center of its neighbors. A noisy sample usually has a low density and most neighbors of this sample will be on the same side of it, leading to the center of its neighbors far from it, especially for high-noise data. Secondly, for each detected noisy sample, DNR models its movement as a Markov decision process to store the experience in this movement. Finally, NDR learns a policy to iteratively move each detected noisy sample to its near high density region by learns the experience of this movement in reinforcement learning. The learned experience can effectively help the movement adapt to the high-noise in real-world cases. In DNR, the structures of high-noise data can be well consolidated by our detection and movement of noisy samples. To prove the rationality of DNR, we theoretically analyze its convergence. Then, we perform the experiment to illustrate that DNR can better denoise high-noise data than existing denoising methods. The source code can be downloaded from https://github.com/TianyiHuang2022.

1. Introduction

In many research fields, such as data mining and pattern recognition, the success of machine learning depends on the learner's ability to deal with noise [11,18,42,50]. Most existing denoising methods can handle low-noise data. However, they cannot handle high-noise data resulting from varying cluster shapes and sample distributions [14,30,44], because it is hard for these methods to obtain the true structures of high-noise data. Some denoising methods consolidate the structures of high-noise data by moving each sample to its near high density region. For example, MD and MFD [6,22,44] move each sample to its near high density region by the manifold structures of data [1,13]. SAF moves each sample to its near high density

* Corresponding author.
E-mail address: wfzhu@uestc.edu.cn (W. Zhu).

https://doi.org/10.1016/j.ins.2022.10.008
0020-0255/

region by a Gaussian noise model [46]. Although these methods well consolidate the structures of noisy synthetic datasets, they cannot adapt to the high-noise in real-world cases, such as varying sample distributions. This is because they actually regard all samples in data as noisy samples and thus move all of them. In this way, the non-noisy samples which are important to construct structures of data will also be moved. Some other methods for noise, such as DBSCAN and DPC [8,15,37], regard the samples with low densities as noisy samples and then remove these samples. Although, they can handle low-noise data, they cannot obtain the true structures of high-noise data. One reason is that the density itself cannot well detect the noisy samples in high-noise cases. The other one is that they do not adjust the detected noisy samples. Therefore, to obtain the true structures of high-noise data in real-world, it is reasonable to combine an effective detection of noisy samples with an appropriate adjustment of these samples.

For high-noise data in real-world cases, an appropriate adjustment of the noisy samples can be realized based on the experience for this adjustment in reinforcement learning (RL) (i.e., the different policies of adjusting noisy samples and the corresponding adjustment performances) [40,25]. In general, much of the experience for expected purposes can be produced in RL and then used to learn a policy to reach the purposes. Recently, some researchers have already combined RL wirh their methods [34,38]. For example, Yu et al. use RL to restore images with the experience of rasing the PSNR value between a noised image and its reference image [48]. In weak-supervised learning, Wang et al. use RL to improve the performance of graph denoising with the experience in graph representation [43]. For secure visible light communication, Xiao et al. propose a RL-based VLC beamforming control scheme to achieve the optimal beamforming policy against the eavesdropper [47]. However, to the best of our knowledge, in unsupervised learning, no research uses RL to denoise data with the experience of adjusting the noisy samples.

To consolidate the structures of high-noise data, we propose a new unsupervised denoising method called Denoising by the Noise-index and Reinforcement-learning (DNR). It firstly detects the noisy samples in data by our new noise index, called Density Distance Index (DDI), then trains the agent of each noisy sample to move this sample to its near high density region in a continuous control RL framework. By the detection of DDI, a noisy sample has not only low density as in previous methods but also a long distance between this sample and the center of its neighbors. The reason for utilizing this distance is that most of the neighbors of a noisy sample will be on the same side of this sample, leading to the center of its neighbors far from it, especially in high-noise cases. In our RL framework, the agent of each noisy sample is trained to move this sample to its near high density region by maximizing the cumulative rewards. The reward includes the following two parts. One is the raised value of the density of each noisy sample after its movement. The other one is the similarity between the movement direction and the density gradient of this sample. In this denoising process, the structures of high-noise data can be well consolidated by our detection and movement of noisy samples. We analyze the convergence of our DNR to prove its rationality. Our experiment illustrates that compared with existing denoising methods, DNR can better denoise the high-noise data.

The rest of this paper is as follows. Section 2 reviews density gradient and existing RL methods for continuous control. In Section 3, we propose DDI to detect noisy samples and DNR method to consolidate the structures of high-noise data. In Section 4, we analyze the convergence of DNR. In Section 5, we compare DNR with existing denoising methods to illustrate the advantages of DNR. In the end, we conclude this paper in Section 6.

2. Preliminaries

Our work is highly related to density gradient and the RL for continuous control. Therefore, we review them in this section.

2.1. Density gradient

Most of the density based clustering methods cluster the samples with low densities to their neighbors with higher densities. However, these methods cannot well adapt to high-noise. To solve this problem, DENCLUE uses density gradient to mine the true structures of high-noise data [23]. Enlightened from DENCLUE, our DNR uses density gradient to consolidate the structures of these data.

In DENCLUE, the density gradient of a sample $\boldsymbol{x}$ can be computed by the following steps. To compute the density of $\boldsymbol{x}$, we firstly need to mathematically describe the influence of its neighbors in the computing of the density. This description can be given by the basic influence function of sample $\boldsymbol{y}$ to $\boldsymbol{x}$ as

$$f_B^{\boldsymbol{y}}(\boldsymbol{x}) = f_B(\boldsymbol{x},\boldsymbol{y}). \tag{1}$$

In principle, the basic influence function can be an arbitrary function of $\boldsymbol{x}$ and $\boldsymbol{y}$.

Example 1. The Gaussian influence function of $\boldsymbol{y}$ for $\boldsymbol{x}$ is

$$f_{Gaussian}^{\boldsymbol{y}}(\boldsymbol{x}) = e^{-\frac{d(\boldsymbol{x},\boldsymbol{y})^2}{2\sigma^2}}, \tag{2}$$

where $d(\boldsymbol{x},\boldsymbol{y})$ is Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{y}$. The Gaussian influence function is a frequently used basic influence function.

Then we can get the density of $\boldsymbol{x}$ with respect to its neighbors $\boldsymbol{D_x} = \{\boldsymbol{x_1}, \cdots, \boldsymbol{x_k}\}$ as

$$f_B^{\boldsymbol{D}_{\boldsymbol{x}}}(\boldsymbol{x}) = \sum_{i=1}^{k} f_B^{\boldsymbol{x}_i}(\boldsymbol{x}). \tag{3}$$

In general, $\boldsymbol{D}_{\boldsymbol{x}}$ can be defined as the k-nearest neighbors of $\boldsymbol{x}$ [9].

Based on the density of $\boldsymbol{x}$, the density gradient of $\boldsymbol{x}$ with respect to $\boldsymbol{D}_{\boldsymbol{x}}$ is defined by

$$\nabla f_B^{\boldsymbol{D}_{\boldsymbol{x}}}(\boldsymbol{x}) = \sum_{i=1}^{k} (\boldsymbol{x}_i - \boldsymbol{x}) f_B^{\boldsymbol{x}_i}(\boldsymbol{x}). \tag{4}$$

Example 2. The density gradient of $\boldsymbol{x}$ based on the Gaussian influence function is

$$\nabla f_{Gaussian}^{\boldsymbol{D}_{\boldsymbol{x}}}(\boldsymbol{x}) = \sum_{i=1}^{k} (\boldsymbol{x}_i - \boldsymbol{x}) e^{-\frac{d(\boldsymbol{x},\boldsymbol{x}_i)^2}{2\sigma^2}}. \tag{5}$$

2.2. RL Methods for continuous control

Recently, the success of RL has attracted much attention from researchers [33,35,40]. It learns a policy for the agent to reach the expected purpose by exploiting the experience from the actions taken by this agent for achieving this purpose [45]. Based on RL, DNR can well adapt to the high-noise data in real-world cases by the experience of consolidating the structures of these data.

When an agent interacts with its environment, RL aims to learn a policy to maximize the cumulative reward of this agent in a Markov decision process. At each discrete time step t, for a given state $\boldsymbol{s}^t \in \mathcal{S}$, the agent selects an action $\boldsymbol{a}^t \in \mathcal{A}$ with respect to a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$. Then the agent obtains a reward $r(\boldsymbol{s}^t, \boldsymbol{a}^t)$ and a new state $\boldsymbol{s}^{t+1}$ from the environment. The return of the agent is defined as the discounted cumulative reward $R^t = \sum_{i=t}^{T} \gamma^{i-t} r(\boldsymbol{s}^i, \boldsymbol{a}^i)$, where γ is a discount factor and T is the max time step in each episode [10,40]. The action-value function (Q-function) is used in many RL methods to estimate the expected return after taking an action $\boldsymbol{a}^t$ in state $\boldsymbol{s}^t$ following policy π_θ with a parameter set θ. Q-function is defined as

$$Q^{\pi_\theta}(\boldsymbol{s}^t, \boldsymbol{a}^t) = \mathbb{E}_{\boldsymbol{s}^i \sim p_{\pi_\theta}, \boldsymbol{a}^i \sim \pi_\theta}\left[R^t | \boldsymbol{s}^t, \boldsymbol{a}^t\right], \tag{6}$$

where $i > t, \mathbb{E}[*]$ means the mathematical expectation and $\boldsymbol{s}^i \sim p_{\pi_\theta}$ means that the next state, $\boldsymbol{s}^{i+1}$, is from a distribution $P(\pi_\theta | \boldsymbol{s}^i, \boldsymbol{a}^i)$. An important challenge in RL is how to keep stability and also achieve convergence in continuous action spaces [2].

Deep Deterministic Policy Gradient (DDPG) method aims to solve this challenge by learning a Q-function and a deterministic policy, $\boldsymbol{a}^t = \mu_\theta(\boldsymbol{s}^t)$ [21,28]. The critic network $Q_\phi(\boldsymbol{s}^t, \boldsymbol{a}^t)$ is used to estimate $Q^{\mu_\theta}(\boldsymbol{s}^t, \boldsymbol{a}^t)$ by a collected set $\boldsymbol{G}$ of transitions $(\boldsymbol{s}^i, \boldsymbol{a}^i, r^i, \boldsymbol{s}^{i+1})$. Thus the Q-learning in DDPG is by minimizing the following mean-squared Bellman error loss with stochastic gradient descent.

$$L(\phi, G) = \mathbb{E}_{(\boldsymbol{s}^t, \boldsymbol{a}^t, r^t, \boldsymbol{s}^{t+1}) \sim \boldsymbol{G}}\left[\left(Q_\phi(\boldsymbol{s}^t, \boldsymbol{a}^t) - z^t\right)^2\right], \tag{7}$$

where $z^t = r^t + \gamma Q_\phi(\boldsymbol{s}^{t+1}, \mu_\theta(\boldsymbol{s}^{t+1}))$. In the process of learning a policy, we want to learn a deterministic policy μ_θ which decides the action that maximizes $Q_\phi(\boldsymbol{s}^t, \boldsymbol{a}^t)$. Therefore, the actor network in DDPG performs gradient ascent to solve

$$\max_\theta \mathbb{E}_{\boldsymbol{s}^t \sim \boldsymbol{G}}\left[Q_\phi(\boldsymbol{s}^t, \mu_\theta(\boldsymbol{s}^t))\right]. \tag{8}$$

But there are still some problems in DDPG, such as the over estimation of Q-function and brittleness.

There are two important improvements for DDPG, TD3 and SAC [10,17]. TD3 aims to solve the overestimation of the Q-function. It trains two critic networks, one is $Q_{\phi_1}(\boldsymbol{s}^t, \boldsymbol{a}^t)$ and the other is $Q_{\phi_2}(\boldsymbol{s}^t, \boldsymbol{a}^t)$. Then it computes z^t by whichever of the two Q-functions has a smaller Q-value. SAC aims to solve the brittleness of DDPG by the approximate inference in prior off-policy maximum entropy algorithms based on soft Q-learning [16].

Both SAC and TD3 are effective RL methods for continuous control. However, in SAC, there is a parameter, reward scale, that needs to be tuned for different environments [17]. If we use SAC to perform our denoising task, we need to tune this parameter for different datasets, limiting the generalization of our method. Therefore, we use TD3 to perform our denoising task.

3. Denoise by DDI and RL

When the structure of the original data is clear, we are easy to accurately cluster the samples in the original data as shown in Fig. 1 (a). However, when the high-noise is added into the features of the samples in the original data, the sample distribution of these data will be changed and thus the structures of these data will be hidden in the added noise as shown in Fig. 1

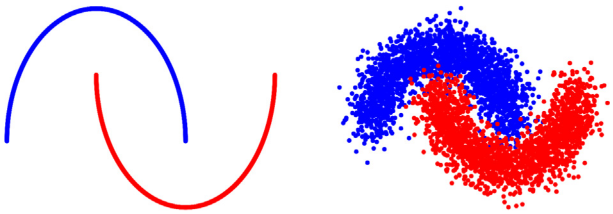


Fig. 1. The original data and noisy data. (a) Original data; (b) Noisy data.

(b). In this case, we are hard to accurately cluster the samples in the noisy data. To accurately cluster the samples in the noisy data, the consolidation of the structure of the data makes the sample distributions of the noisy data and the original data as similar as possible by moving the samples in the noisy data [46].

To consolidate the structures of high-noise data, in this section, we propose an effective denoising method, DNR. It firstly well detects the noisy samples in data by our DDI and then uses TD3 to train the agent of each detected noisy sample to move it to its near high density region in the direction of its density gradient. In this way, the non-noisy samples can well present the true structures of high-noise data and the movement of the noisy samples can well consolidate these structures.

3.1. DDI: a measure to detect noisy samples

To consolidate the structures of high-noise data, we move noisy samples to their high density regions and keep non-noisy ones. So it is very important to well detect noisy samples. Traditional methods detect noisy samples just by their densities. In this way, they cannot well detect noisy samples with high-noise. To deal with this issue, we propose DDI, an effective index for detecting noisy samples. It detects a noisy sample not only by its density as previous methods do, but also by the distance between this sample and the center of its neighbors. DDI works based on the following two principles. 1) A noisy sample usually has a low density and is at the border of a cluster. 2) Most of the neighbors of a noisy sample will be on the same side of this sample, so the center of its neighbors will be far away from it. The DDI of $\boldsymbol{x}$ is defined as

$$DDI_{\boldsymbol{x}} = \frac{d(\boldsymbol{x}, \overline{\boldsymbol{D}}_{\boldsymbol{x}})}{f_B^{\boldsymbol{D}_{\boldsymbol{x}}}(\boldsymbol{x})}, \tag{9}$$

where $d(\boldsymbol{x}, \overline{\boldsymbol{D}}_{\boldsymbol{x}})$ is the Euclidean distance between $\boldsymbol{x}$ and $\overline{\boldsymbol{D}}_{\boldsymbol{x}}, \boldsymbol{D}_{\boldsymbol{x}}$ is the k-nearest neighbors of $\boldsymbol{x}$ as $\boldsymbol{D}_{\boldsymbol{x}} = \{\boldsymbol{x_1}, \cdots, \boldsymbol{x_k}\}$, and $\overline{\boldsymbol{D}}_{\boldsymbol{x}}$ is the center of $\boldsymbol{D}_{\boldsymbol{x}}$ as

$$\overline{\boldsymbol{D}}_{\boldsymbol{x}} = \frac{\sum_{i=1}^{k} \boldsymbol{x_i}}{k}. \tag{10}$$

A large $DDI_{\boldsymbol{x}}$ means that $\boldsymbol{x}$ is a potential noisy sample. In practice, we set a cutoff value, DDI^c. If $DDI_{\boldsymbol{x}} > DDI^c$, we define $\boldsymbol{x}$ as a noisy sample; Otherwise, we define $\boldsymbol{x}$ as a non-noisy sample. DDI^c is selected from the $DDIs$ of the samples in data. After sorting the $DDIs$ of all the samples in ascending order as $DDI^1 < DDI^2 <, \dots, < DDI^N$, we can select DDI^c by

$$DDI^c = DDI^{\lceil N \times q \rceil}, \tag{11}$$

where N is the number of the samples in the dataset and $0 < q < 1$ is a free parameter to select the DDI^c.

Example 3. Fig. 2 and Fig. 3 are an example to show the effectiveness of DDI in high-noise data on circles dataset and iris dataset, respectively. In Fig. 2 (a) and Fig. 3 (a), the different labels are presented by different colors. In (b) and (c) of these two figures, the red points are the detected noisy samples and the blue points are non-noisy samples. The k for get the $\boldsymbol{D}_{\boldsymbol{x}}$ for circles is set as 150 and for iris is set as 30. As we can see, compared with the density used in previous methods, DDI can better detect the noisy samples in different classes of circles and the noisy samples in different classes of iris. After the noise detection of DDI, the remained non-noisy samples can well present the structures of these data.

3.2. DNR method

In DNR, we firstly detect the noisy samples by DDI, then we move each of them to its near high density region by its density gradient in an RL framework. The main components (i.e., agent, state, action, and reward) in our RL are described as follows. In iteration t, each noisy sample $\boldsymbol{x}_i^t$ can be moved by its agent. The state $\boldsymbol{s}_i^t$ is defined by the density and the density gradient of the $\boldsymbol{x}_i^t$ as $\boldsymbol{s}_i^t = \left[f_B^{\boldsymbol{D}_{\boldsymbol{x}_i^t}}(\boldsymbol{x}_i^t); \nabla f_B^{\boldsymbol{D}_{\boldsymbol{x}_i^t}}(\boldsymbol{x}_i^t)\right]$. If $DDI_{\boldsymbol{x}_i^t} > DDI^c$, $\boldsymbol{x}_i^t$ will be moved by action $\boldsymbol{a}_i^t$ as

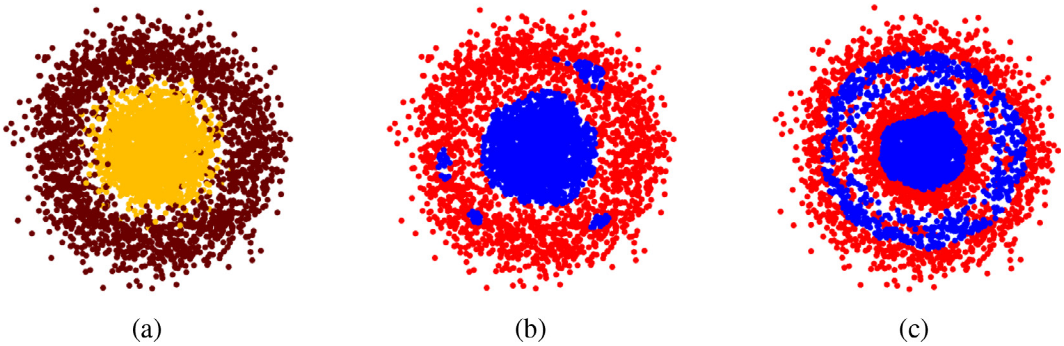


Fig. 2. Example on circles. (a) Original data with labels; (b) Noise detection by density; (c) Noise detection by DDI.

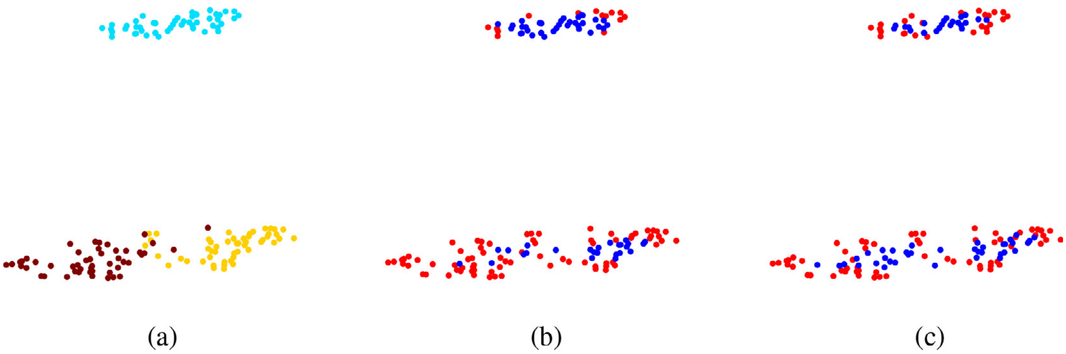


Fig. 3. Example on iris. (a) Original data with labels; (b) Noise detection by density; (c) Noise detection by DDI.

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{a}_i^t. \tag{12}$$

To avoid the excessive movement of $\boldsymbol{x}_i^t$, $b > 0$ is used to clip $\boldsymbol{a}_i^t$ as

$$a_{i,j}^t = \begin{cases} -b & a_{i,j}^t < -b \\ a_{i,j}^t & -b \leqslant a_{i,j}^t \leqslant b \\ b & a_{i,j}^t > b \end{cases} \tag{13}$$

The reward $r_i(\boldsymbol{s}_i^t, \boldsymbol{a}_i^t)$ is defined as

$$r_i(\boldsymbol{s}_i^t, \boldsymbol{a}_i^t) = \left(f_B^{\boldsymbol{D}_{\boldsymbol{x}_i^t}}(\boldsymbol{x}_i^{t+1}) - f_B^{\boldsymbol{D}_{\boldsymbol{x}_i^t}}(\boldsymbol{x}_i^t)\right) + \lambda\rho\left(\boldsymbol{a}_i^t, \nabla f_B^{\boldsymbol{D}_{\boldsymbol{x}_i^t}}(\boldsymbol{x}_i^t)\right), \tag{14}$$

where λ is a free parameter and $\rho(\boldsymbol{u}, \boldsymbol{v})$ is the similarity between the directions of $\boldsymbol{u}$ and $\boldsymbol{v}$. In practice, $\rho(\boldsymbol{u}, \boldsymbol{v})$ can be defined based on Pearson correlation coefficient or cosine function. By our carefully designed reward, the density of a noisy sample $\boldsymbol{x}_i$ will be higher and higher by moving it to its near high density region with the help of its density gradient. Thus DNR can well denoise the high-noise data while maintaining the structures of them. Note that it is very important to make the moving direction of x_i similar to its density gradient by the reward function. From our experiment, we find that without this operation, RL may move noisy samples to the same region and then form a new high density region.

In DNR, each noisy sample is moved by its corresponding agent. Thus, there are a lot of agents in our RL framework. It is hard to apply existing techniques of the multi-agent RL to train them, because of the extensive computational cost in training many networks for these agents [29]. To solve this issue, we put the experienced transitions of different agents in one memory pool. Then we train each agent by TD3, an effective RL method for continuous control. Thus, all agents can share parameters and learn effectively. In DNR the different agents perform by the same policy which makes each noisy sample move to its near high density region by its density gradient. Therefore, it is reasonable to train these agents by one memory pool.

The Q-function in our denosing task by TD3 is

$$Q^{\mu_\theta}(\boldsymbol{s}_i^t, \boldsymbol{a}_i^t) = \mathbb{E}_{\boldsymbol{s}_i^j \sim p_{\mu_\theta}, \boldsymbol{a}_i^j \sim \mu_\theta}\left[R_i^t | \boldsymbol{s}_i^t, \boldsymbol{a}_i^t\right]. \tag{15}$$

There are four critic networks, $Q_{\phi_1}, Q_{\phi_2}, Q_{\phi'_1}$ and $Q_{\phi'_2}$ for estimating this Q-function. There are also two actor networks, μ_θ and $\mu_{\theta'}$ for updating the policy. Q_{ϕ_1} and Q_{ϕ_2} are updated by minimizing the following loss functions as

$$\begin{aligned} L(\phi_1, \boldsymbol{G}) = & \; \mathbb{E}_{(\boldsymbol{s_i^t}, \boldsymbol{a_i^t}, r_i^t, \boldsymbol{s_i^{t+1}}) \sim \boldsymbol{G}} \sum_{i=1}^{n} \left[\left(Q_{\phi_1}(\boldsymbol{s_i^t}, \boldsymbol{a_i^t}) - z_i^t \right)^2 \right] \\ L(\phi_2, \boldsymbol{G}) = & \; \mathbb{E}_{(\boldsymbol{s_i^t}, \boldsymbol{a_i^t}, r_i^t, \boldsymbol{s_i^{t+1}}) \sim \boldsymbol{G}} \sum_{i=1}^{n} \left[\left(Q_{\phi_2}(\boldsymbol{s_i^t}, \boldsymbol{a_i^t}) - z_i^t \right)^2 \right], \end{aligned} \tag{16}$$

where n is the number of the noisy samples and

$$z_i^t = r_i^t + \gamma \min_{j=1,2} Q_{\phi_j}(\boldsymbol{s_i^{t+1}}, \mu_{\theta'}(\boldsymbol{s_i^{t+1}})). \tag{17}$$

Accordingly, $Q_{\phi'_i}$ is updated by

$$\begin{aligned} \phi'_1 \leftarrow \tau\phi_1 + (1-\tau)\phi'_1 \\ \phi'_2 \leftarrow \tau\phi_2 + (1-\tau)\phi'_2. \end{aligned} \tag{18}$$

The policy is learned by

$$\max_{\theta} \mathbb{E}_{\boldsymbol{s_i^t} \sim \boldsymbol{G}} \sum_{i=1}^{n} [Q_{\phi}(\boldsymbol{s_i^t}, \mu_{\theta}(\boldsymbol{s_i^t}))]. \tag{19}$$

Accordingly, $\mu_{\theta'}$ is updated by

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta'. \tag{20}$$

We use the delaying policy update in TD3 to reduce per-update error for improving the performance of optimizing continuous control. In this update, we use parameter e to control the delaying steps [10]. The algorithm of DNR is detailed in Algorithm 1.

Algorithm 1: The algorithm of DNR

Input: Noisy data $\boldsymbol{X} = \{\boldsymbol{x_1}, \cdots, \boldsymbol{x_N}\}$
parameters: The threshold $\boldsymbol{b}$, the number of epoches num_e, and the iteration number T
Output: $\boldsymbol{X^t} = \{\boldsymbol{x_1^t}, \boldsymbol{x_2^t}, \cdots, \boldsymbol{x_N^t}\}$
Initialize $\phi_1, \phi'_1, \phi_2, \phi'_2, \theta'$ and θ with $mean = 0$ and $std = 1$
Initialize $DDI^c = None$
for num = 1 : num_e **do**
  $\boldsymbol{X^t} \leftarrow \boldsymbol{X}$
  for t = 1 : T **do**
    Compute the $DDI_{\boldsymbol{x_i}}^t$ of each sample x_i^t in X^t by formula (10)
  end for
  if $DDI^c == None$
    Compute DDI^c by formula (11)
  end if
  Get the noisy samples $\left\{\boldsymbol{x_1^t}, \cdots, \boldsymbol{x_n^t} | DDI_{\boldsymbol{x_i}}^t > DDI^c\right\}$ from $\boldsymbol{X^t}$
  for i = 1 : n **do**
    Move noisy sample $\boldsymbol{x_i^t}$ by $\boldsymbol{a_i^t} = \mu_\theta(\boldsymbol{s_i^t})$ with $\boldsymbol{b}$ and store transition tuple $(\boldsymbol{s_i^t}, \boldsymbol{a_i^t}, r_i^t, \boldsymbol{s_i^{t+1}})$ to $\boldsymbol{G}$
    Sample mini-batch of the transitions from $\boldsymbol{G}$
    Update ϕ_1 and ϕ_2 by minimizing (16)
    if *imode*
      Update θ by (19)
      Update ϕ'_1 and ϕ'_2 by (18)
      Update θ' by (20)
    end if
  end for
end for
return $\boldsymbol{X^t}$

Example 4. Fig. 4 is an example to show the training process of DNR. We use TD3 to denoise 1000 samples of 5 classes in the MNIST dataset with $k = 20$ [5]. There are 100 iterations in each epoch of the training in TD3. As we can see, in the initial dataset, the yellow cluster and the blue cluster are close to each other while the samples in the same cluster are not close enough to each other. However, with the increasing of the epochs, TD3 can consolidate the structures of these samples better and we can get the satisfactory visual clustering result at 8-th epoch. In this epoch, the samples in the same cluster are close to each other and the different clusters are far from each other.

3.3. Complexity analysis

For space complexity, the main space complexity is from the distance matrix of the samples in the dataset. This space complexity is $O(n^2d)$. For the time complexity of the algorithm of DNR, one of the most computational parts is to compute the densities and the DDIs of the samples in each iteration. The time complexity of this part is $O\left(TN^2d\right)$, where T is the number of iterations, N is the number of the samples, and d is the number of the dimensions. The other part is training the neural network in each iteration. The time complexity of this part is $O\left(T\left(n \times d \times (l_1 + l_m) + \sum_{i=1}^{m-1} l_i \times l_{i+1}\right)\right)$, where n is the number of the noisy samples, m is the number of the hide layers in neural network, and l_i is the number of nodes in i-th hide layer. Therefore the time complexity of our method is $O\left(T\left(n^2d + n \times \left(d \times (l_1 + l_m) + \sum_{i=1}^{m-1} l_i \times l_{i+1}\right)\right)\right)$.

4. Convergence analysis

In this section, we analyze the convergence of DNR algorithm. In this algorithm, we want to move the noisy samples to their near high density regions with the directions similar to their density gradients. But the density gradient is given by a definition in formula (4) instead of taking the derivative of the density function. Therefore, to discuss the convergence of DNR, the monotonicity of density function on the direction of the density gradient needs to be discussed.

As we can see, $(\boldsymbol{x_i} - \boldsymbol{x})$ should exist in formula (4), because it is the direction of $\boldsymbol{x_i}$ for $\boldsymbol{x}$. However, $f_B(\boldsymbol{x}, \boldsymbol{x_i})$ can be changed to another function because this function is used to tune the weight of this direction. Based on this finding we have the following theorem.

Theorem 1. *$\forall f_B^{\boldsymbol{D_x}}(\boldsymbol{x})$, if $\exists h_B(\boldsymbol{x}, \boldsymbol{y}) \neq 0$ makes*

$$\frac{\partial}{\partial \boldsymbol{x}} f_B^{\boldsymbol{D_x}}(\boldsymbol{x}) = \sum_{i=1}^{k} (\boldsymbol{x_i} - \boldsymbol{x}) h_B(\boldsymbol{x}, \boldsymbol{x_i}). \tag{21}$$

$f_B^{\boldsymbol{D_x}}(\boldsymbol{x})$ will increase monotonously by moving $\boldsymbol{x}$ on the direction of $\frac{\partial}{\partial \boldsymbol{x}} f_B^{\boldsymbol{D_x}}(\boldsymbol{x})$.

It is easy to see that Theorem 1 is hold. Based on Theorem 1 we have the following propositions.

Proposition 1. If basic influence function $f_B(\boldsymbol{x}, \boldsymbol{y})$ is the Gaussian influence function, $f_B^{\boldsymbol{D_x}}(\boldsymbol{x})$ will increase monotonously by moving $\boldsymbol{x}$ on the direction of $\sum_{i=1}^{k} (\boldsymbol{x_i} - \boldsymbol{x}) f_B(\boldsymbol{x}, \boldsymbol{x_i})$, the density gradient defined in formula (4).

Proof. Take Gaussian influence function in basic influence function $f_B(\boldsymbol{x}, \boldsymbol{y})$. Then we have

$$\frac{\partial}{\partial \boldsymbol{x}} f_{Gaussian}^{\boldsymbol{D_x}}(\boldsymbol{x}) = \frac{1}{\sigma^2} \sum_{i=1}^{k} (\boldsymbol{x_i} - \boldsymbol{x}) e^{-\frac{d(\boldsymbol{x}, \boldsymbol{x_i})^2}{2\sigma^2}}. \tag{22}$$

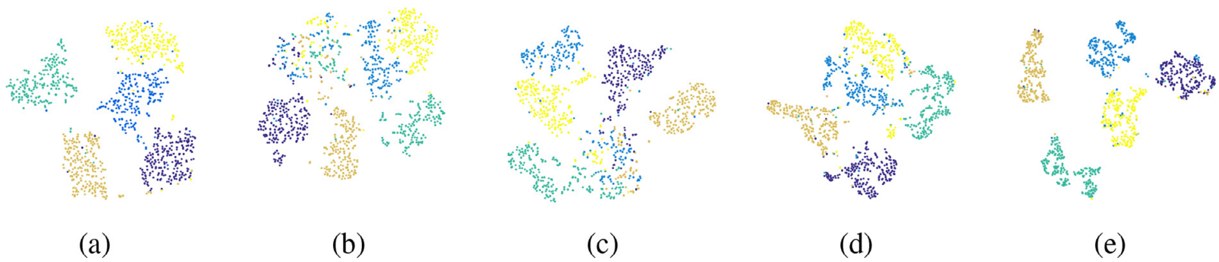


Fig. 4. Example of DNR. (a) Original data; (b) Epoch 1; (c) Epoch 2; (d) Epoch 4; (e) Epoch 8.

As we can see, $\frac{\partial}{\partial \boldsymbol{x}} f_{Gaussian}^{\boldsymbol{D_x}}(\boldsymbol{x})$ has the same direction as the density gradient of the Gaussian influence function in formula (5). Thus, Proposition 1 is hold.

Based on Proposition 1, the density gradient defined in formula (4) can be regarded as a special case in Theorem 1 with Gaussian influence function. Now we discuss the convergence of DNR.

Proposition 2. With a symmetrical basic influence function $f_B(\boldsymbol{x},\boldsymbol{y}) = f_B(\boldsymbol{y},\boldsymbol{x}), \sum_{i=1}^{N} f_B^{\boldsymbol{D_{x_i}}}(\boldsymbol{x_i})$ will increase monotonously by moving each $\boldsymbol{x_j}$ on the direction of $\frac{\partial}{\partial \boldsymbol{x_j}} f_{\boldsymbol{B}}^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j})$, where $\boldsymbol{D_{x_j}} = \boldsymbol{X} - \{\boldsymbol{x_j}\}$ and $\boldsymbol{X}$ is the whole dataset.

Proof. Because of the symmetry of the basic influence function, we have

$$\begin{aligned} \triangle_{\boldsymbol{a}(\boldsymbol{x_j})} f_B^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j}) &= \sum_{i=1, i\neq j}^{N} \triangle_{\boldsymbol{a}(\boldsymbol{x_j})} f_B^{\{\boldsymbol{x_j}\}}(\boldsymbol{x_i}) \\ &= \sum_{i=1, i\neq j}^{N} \triangle_{\boldsymbol{a}(\boldsymbol{x_j})} f_B^{\boldsymbol{D_{x_i}}}(\boldsymbol{x_i}). \end{aligned} \tag{23}$$

where $\triangle_{\boldsymbol{a}(\boldsymbol{x})} f_B^{\boldsymbol{D_y}}(\boldsymbol{y})$ is the increased value of $f_B^{\boldsymbol{D_y}}(\boldsymbol{y})$ after moving $\boldsymbol{x}$ by action $\boldsymbol{a}$. So, if $\boldsymbol{a}$ has the same direction of $\frac{\partial}{\partial \boldsymbol{x_j}} f_{\boldsymbol{B}}^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j}), f_B^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j})$ and $\sum_{i=1, i\neq j}^{N} f_B^{\boldsymbol{D_{x_i}}}(\boldsymbol{x_i})$ will increase monotonously. Thus $\sum_{i=1}^{N} f_B^{\boldsymbol{D_{x_i}}}(\boldsymbol{x_i})$ will also increase monotonously.

Under the conditions of Theorem 1 and Proposition 2, we can see that if $\boldsymbol{a_j}$ has the same direction with $\frac{\partial}{\partial \boldsymbol{x_j}} f_B^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j}), \sum_{i=1}^{N} f_B^{\boldsymbol{D_{x_i}}}(\boldsymbol{x_i})$ will be convergence with a bounded $f_B(\boldsymbol{x},\boldsymbol{y})$. However, it is hard for $\boldsymbol{a_j}$ to have the same direction with $\frac{\partial}{\partial \boldsymbol{x_j}} f_B^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j})$ by $\boldsymbol{a_j} = \mu_\theta(\boldsymbol{s_j})$ in TD3. We prove that with a relaxed condition $\boldsymbol{a_j}^\intercal \cdot \frac{\partial}{\partial \boldsymbol{x_j}} f_B^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j}) > 0, \sum_{i=1}^{N} f_B^{\boldsymbol{D_{x_i}}}(\boldsymbol{x_i})$ will also be convergence with a bounded $f_B(\boldsymbol{x},\boldsymbol{y})$, under the conditions of Theorem 1 and Proposition 2. $\boldsymbol{a_j}^\intercal$ is the transposed $\boldsymbol{a}$. From experiment, we find that it is easy to meet this relaxed condition by $\boldsymbol{a_j} = \mu_\theta(\boldsymbol{s_j})$ in TD3 with Gaussian influence function.

Theorem 2. *Let $\boldsymbol{a_j} = \mu_\theta(\boldsymbol{s_j})$ and $\boldsymbol{e_j} = \frac{\partial}{\partial \boldsymbol{x_j}} f_B^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j})$. $\forall j \in \{1, 2, \ldots, n\}$, if $\boldsymbol{a_j}^\intercal \boldsymbol{e_j} > 0, \sum_{i=1}^{N} f_B^{\boldsymbol{D_{x_i}}}(\boldsymbol{x_i})$ will be convergence with a bounded $f_B(\boldsymbol{x},\boldsymbol{y})$, under the conditions of Theorem 1 and Proposition 2.*

Proof. By Taylor's expansion, we have

$$\begin{aligned} \triangle_{\boldsymbol{a_j}(\boldsymbol{x_j})} f_B^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j}) &= f_B^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j} + \boldsymbol{a_j}) - f_B^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j}) \\ &= f_B^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j}) + \boldsymbol{a_j}^\intercal \frac{\partial}{\partial \boldsymbol{x_j}} f_B^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j}) + o(|\boldsymbol{a_j}|) - f_B^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j}) \\ &\approx \boldsymbol{a_j}^\intercal \boldsymbol{e_j}. \end{aligned} \tag{24}$$

Thus

$$\triangle_{\boldsymbol{a_j}(\boldsymbol{x_j})} f_B^{\boldsymbol{D_{x_j}}}(\boldsymbol{x_j}) > 0 \Longleftrightarrow \boldsymbol{a_j}^\intercal \boldsymbol{e_j} > 0. \tag{25}$$

Combining this result with formula (23), we have that if $\boldsymbol{a_j}^\intercal \boldsymbol{e_j} > 0, \sum_{i=1}^{N} f_B^{\boldsymbol{D_{x_i}}}(\boldsymbol{x_i})$ will increase monotonously by moving $\boldsymbol{x_j}$ on $\boldsymbol{a_j}$. As we can see, $f_B(\boldsymbol{x},\boldsymbol{y})$ is bounded. Thus, Theorem 2 is hold.

Theoretically, we need to compute the density of $\boldsymbol{x}$ by $\boldsymbol{D_{x_j}} = \boldsymbol{X} - \{\boldsymbol{x_j}\}$, thus it is hard to keep the local structures of high-noise data. We also need to set the value of b small enough, leading to the poor efficiency for iterations. Therefore in practice, we compute the density of $\boldsymbol{x}$ by its k-nearest neighbors and may not set the value of b small enough. In this case, RL can effectively tune the actions of the noisy samples, such that if the density of $\boldsymbol{x}$ is low, RL will make $\boldsymbol{x}$ move with a big step and if the density of $\boldsymbol{x}$ is high, RL will make $\boldsymbol{x}$ move with a little step.

5. Performance evaluation and analysis

In this section, we compare DNR with three existing denoising methods on nine datasets. Firstly, we compared DNR with these methods in a visualized experiment. Then, to illustrate its benefits for subsequent data processing, we cluster these datasets that denoised by the above different denoising methods [19,41,26]. As shown in our experiment, compared with the existing denoising methods, DNR can better consolidate the structures of high-noise data, and thus gets better clustering results.

5.1. Visualized experiment

In this subsection, we compare DNR with three existing denoising methods including MD, RMD, and SAF on four high-noise synthetic datasets and five real-world datasets in a visualized way [7,22,46]. The details of the above three methods are as follows.

1. **MD**: Manifold denosing uses Laplacian neighbor graph to denoise data in their mainfold structure.
2. **RMD**: Robust manifold denosing performs piecewise smooth manifold denosing by spectral graph wavelets and thus makes the denosing process more robust than MD.
3. **SAF**: Structure-aware filtering performs denosing by moving samples to their near high density regions with a Gaussian noise model.

The four high-noise synthetic datasets, circles-G, moons-G, circles-U and moons-U, are made by skit-learn package and the five real-world datasets include iris, wine, Reuters, Jaffe, and MNIST [36]. The added noise in circles-G and moons-G is from Gaussian distribution. The added noise in circles-U and moons-U is from the uniform distribution. For MNIST and Reuters, we respectively select their 5000 samples and 1000 samples to perform our experiment. All of the above datasets are described in Table 1. For the datasets which are more than two dimensions, we use t-SNE to show their 2D visualized results [31].

We use Gaussian influence function to compute the densities of the samples and find an effective $\rho(\boldsymbol{u},\boldsymbol{v})$ for training the networks in TD3 as

$$\rho(\boldsymbol{u},\boldsymbol{v}) = sgn(cos(\boldsymbol{u},\boldsymbol{v})) \times e^{cos(\boldsymbol{u},\boldsymbol{v})^2}, \tag{26}$$

where $sgn(\cdot)$ is a sign function as

$$sgn(cos(\boldsymbol{u},\boldsymbol{v})) = \begin{cases} 1 & cos(\boldsymbol{u},\boldsymbol{v}) > 0 \\ -1 & otherwise. \end{cases} \tag{27}$$

Some parameters need to be tuned. The k for k-nearest neighbors of each sample is selected from $\{10, 20, 30, 150, 300\}$, the values in movement bound b are selected from $\{0.005, 0.01\}$ in the max–min normalized data, λ is set as 10 and q is selected from $\{0.3 : 0.05 : 0.7\}$. To solve the problem of overfitting, we add the dropout layers in the networks of TD3 [39]. The experiment results are shown in Fig. 5 and Fig. 6.

From Fig. 5, we can see that there is high-noise in our four synthetic datasets. MD and RMD fail to maintain the structures in these four datasets, because they cannot well adapt to the high noise. SAF can not denoise moons-U very well. It may be because the added noise in this dataset is not from Gaussian distribution. However, DNR can well consolidate the structures of the four high noise synthetic datasets.

From Fig. 6, we can get the following conclusions. For iris, RMD and SAF mistakenly connect the cyan class and the red class. For the result of MD on iris, we cannot find the important information that the cyan class is much closer to the red class than the blue class in the original iris. For wine, MD fails to maintain the structures in the same class, RMD fails to part the different classes and SAF cannot well denoise. For Reuters, MD and RMD fail to keep the structure in this dataset and SAF cannot well perform the denoising task. For Jaffe and MNIST, all of the MD, RMD, and SAF fail to maintain the structure in the same class. But as we can see, DNR can well denoise all above real-world datasets while maintaining their structures.

Overall, MD and RMD cannot well denoise our synthetic datasets because they cannot adapt to the high-noise in these datasets. MD, RMD, and SAF cannot well denoise real-world datasets, because they cannot adapt to the complex conditions in these datasets. However, DNR can well denoise not only our synthetic datasets but also real-world datasets.

We also use an example to show the advantages of using RL in denoising with $k = 30$. Fig. 7 (a) is the denoising result without RL on MINST in Fig. 6 (e). In this case, the samples are directly moved by density gradient with the same bound in DNR. Fig. 7 (b) is the denoising result with RL (DNR) on MINST in Fig. 6 (e). To show the advantages of using RL, we also present the denoising results with labels. Comparing the result in Fig. 7 (a) and the result of DNR in (b), we can see that with-

Table 1
Data description.

DID	Dataset	Instances	Features	Classes
1	circles-G	4000	2	2
2	moons-G	4000	2	2
3	circles-U	4000	2	2
4	moons-U	4000	2	2
5	iris	150	4	3
6	wine	178	13	3
7	Reuters	2000	2000	4
8	Jaffe	213	4096	10
9	MNIST	5000	784	10

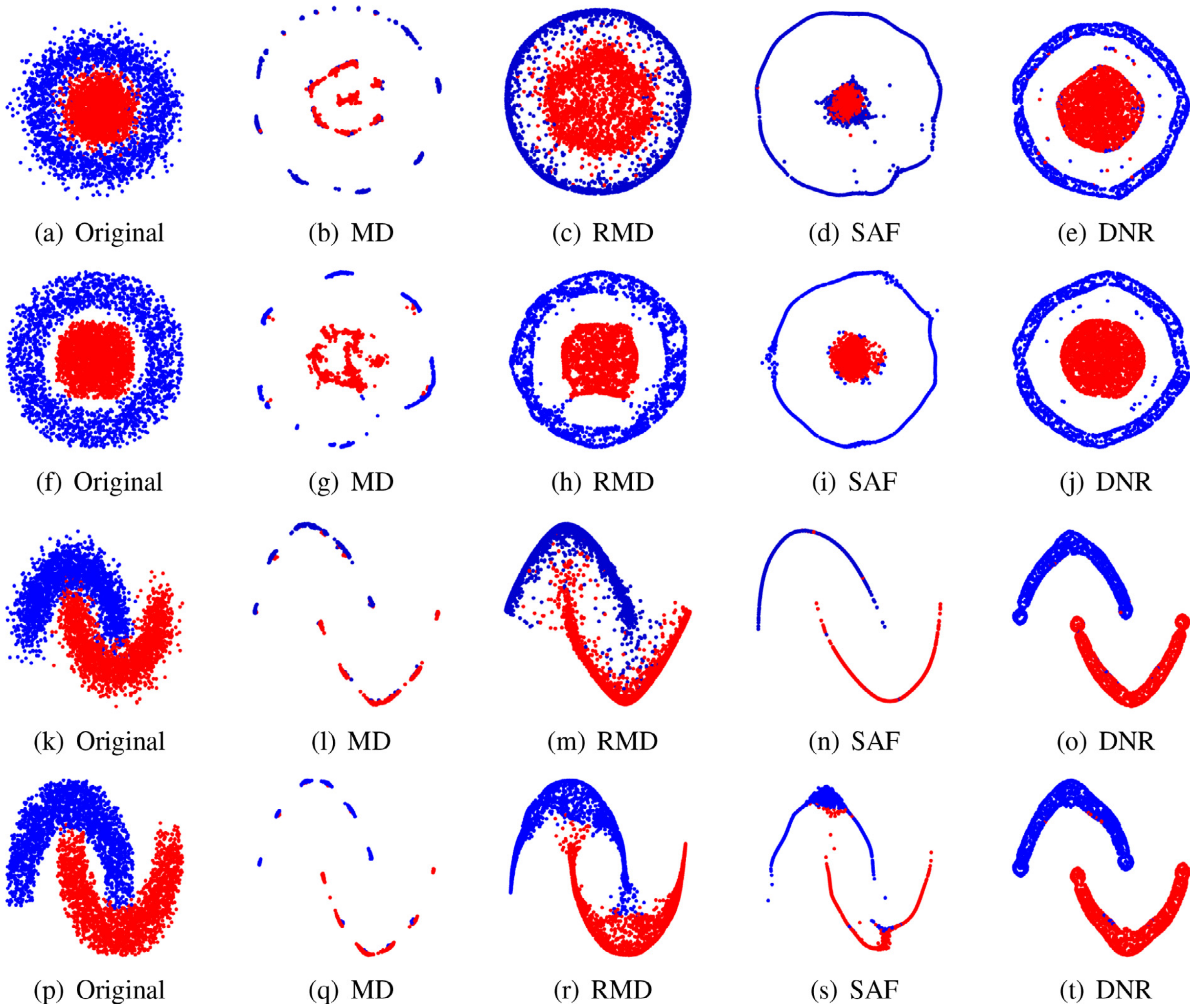


Fig. 5. Visualized results. (a)-(e), (f)-(j), (k)-(o), and (p)-(t) are the visualized results of circles-G, circles-G, moons-G and moons-U, respectively.

out RL, the structures of data cannot be well maintained in some classes. Detailly without RL, the black-blue class at the top of the figure is separated into two parts, some samples of the lemon class at the center of the figure are connected with the samples of the red class, and some samples of the green class at the right of the figure are connected with the samples of the brown class. Overall, with RL we can get better visual clustering.

Fig. 8 shows the normalized average rewards of TD3 in each iteration on with $k = 30$ and $k = 10$, respectively. As we can see, with the increasing of the iterations, the rewards of TD3 in our denoising task have become higher and higher untill convergence. Combining this phenomenon and the experiment results in Fig. 6 (e) and (j), we can find that in our RL framework, TD3 can effectively learn the policy to denoise.

In Fig. 9, to illustrate that DNR can well adapt the noises with various levels, we artificially introduce various levels of noise in the attributes of moons-G, and then use DNR to perform the denoise task with these noise levels. The noise levels are set as 0.09, 0.12, 0.15, and 0.18. As we can see, DNR can well perform the denoising task with the different levels of the noise in circles-G.

5.2. Experiment on clustering

In this subsection, we cluster the datasets that denoised by the different denoising methods. We use k-means, DPC, GDL and U-SENC to cluster the datasets in our experiment. The details of these clustering methods are as follows.

1. ***k*-means** [32] works by computing centres of the different clusters iteratively.
2. **DPC** [37] works by combining the sparse samples into the dense samples.

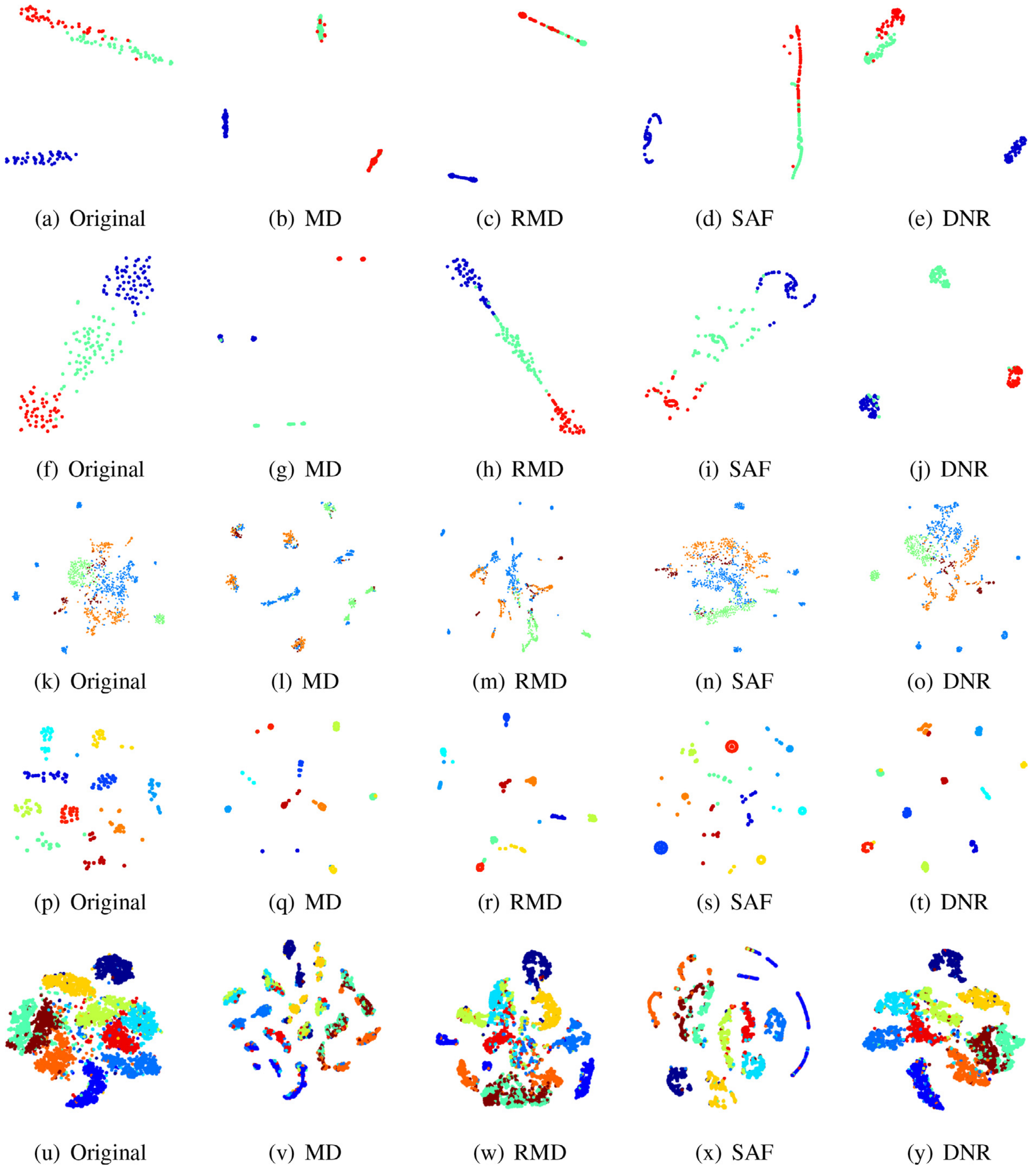
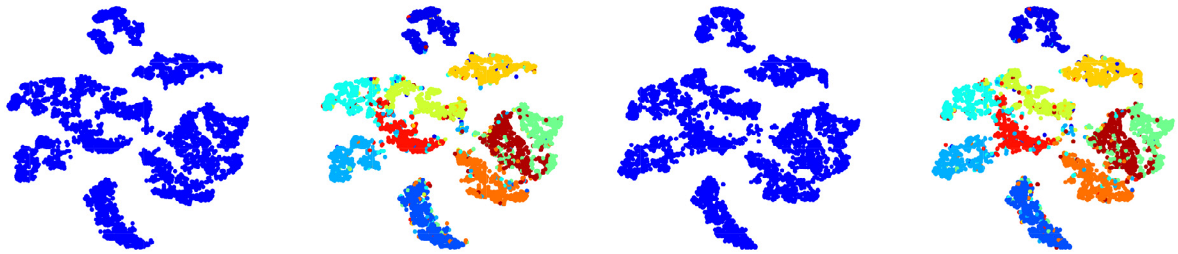


Fig. 6. Visualized results. (a)-(e), (f)-(j), (k)-(o), (k)-(o), (p)-(t), and (u)-(y) are the visualized results of iris, wine, Reuters, Jaffe, and MNIST, respectively.

3. **GDL** [49] works by the average indegree and the average outdegree in the adjacency graph of the samples.
4. **U-SENC** [24] works by interpreting the sparse sub-matrix as a bipartite graph and using transfer cut to efficiently partition the graph for the clustering result.

We use two clustering evaluation metrics: clustering accuracy (ACC) and normalized mutual information (NMI) to measure the clustering performance [27]. The definitions of ACC and NMI are as follows. Denote $\boldsymbol{Q} = \{\boldsymbol{q_1}, \boldsymbol{q_2}, \cdots, \boldsymbol{q_n}\}$ as the clustering results and $\boldsymbol{P} = \{\boldsymbol{p_1}, \boldsymbol{p_2}, \cdots, \boldsymbol{p_n}\}$ as the ground truth label of X. ACC is defined as:



(a) Denoising result by density gradient directly and this result with labels

(b) Denoising result by with RL (DNR) and this result with labels

Fig. 7. Comparing denoising with or without RL on MNIST. (a) Without RL; (b) With RL (DNR).

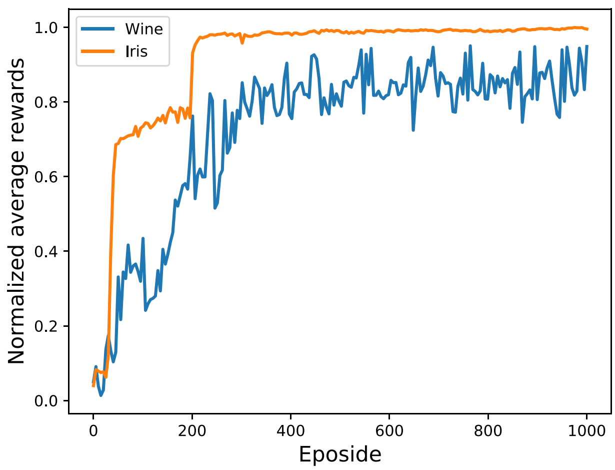
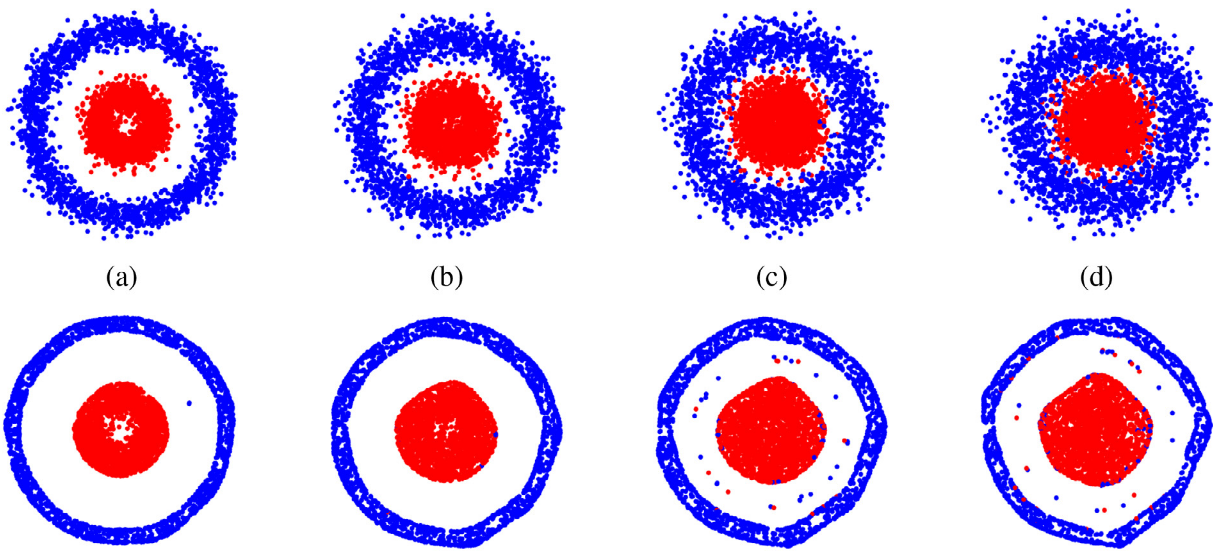


Fig. 8. Normalized average rewards of iris and wine in training.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 9. Visualized results. (a)–(e) are the circles-G with different noise levels and (f)–(j) are the denoising results of the different noise levels, respectively. The noise levels are set as 0.09, 0.12, 0.15, and 0.18.

$$ACC = \frac{\sum_{i=1}^{n} \delta(\boldsymbol{p}_i, map(\boldsymbol{q}_i))}{n}, \tag{28}$$

where $\delta(x, y) = 1$, if $x = y$ and $\delta(x, y) = 0$, otherwise. $map(\boldsymbol{q}_i)$ is the best mapping function that permutes clustering labels to match the given truth labels using the Kuhn-Munkres algorithm. The lager ACC is, the better clustering result is. The NMI is defined as follows [20,27].

$$NMI(\boldsymbol{Q}, \boldsymbol{P}) = \frac{MI(\boldsymbol{Q}, \boldsymbol{P})}{\sqrt{(H(\boldsymbol{Q})H(\boldsymbol{P}))}}, \tag{29}$$

where $H(\boldsymbol{Q})$ and $H(\boldsymbol{P})$ are the entropies of $\boldsymbol{Q}$ and $\boldsymbol{P}$. $MI(\boldsymbol{Q}, \boldsymbol{P})$ is the mutual information metric of $\boldsymbol{Q}$ and $\boldsymbol{P}$. The lager NMI is, the better clustering result is.

We show the NMIs and ACCs of the different denoising methods on different clustering methods in Table 2–5. As we can see that in most cases, DNR can effectively improve the clustering performances. However, MD and RMD are hard to improve the clustering performances because they cannot well deal with high noise and the complex structure in real-world datasets. SAF can well improve the clustering performance on synthetic datasets, but it still cannot well deal with the complex structure in real-world datasets. For the clustering results of k-means, it cannot well cluster the samples with the non-spherical distribution. Therefore, k-means cannot get the high clustering performances in some cases. However, k-means can well mine the global structure in data and thus well cluster the Reuters dataset with the help of DNR. For the clustering results of DPC, it can better cluster the samples with non-spherical distribution than k-means and DNR can well improve the clustering performance of DPC. U-SENC and GDL are two effective clustering methods and sometimes get high performances. As we can see, our DNR can improve their clustering performances even if their clustering performance on the original data is high. The clustering results are highly related to the corresponding visualized results. As we can see, in most cases, if a denoising method has a satisfying visualized performance in Fig. 5 or Fig. 6, it will have a corresponding good clustering performance with at least one clustering method. This phenomenon illustrates that a good denoising method has its benefits for subsequent data processing, such as clustering.

We also show the visualized clustering results of GDL on circles and MNIST datasets in Fig. 10 and Fig. 11 with $k = 150$ and $k = 30$, respectively. because, GDL is suit for clustering these two datasets. As we can see, with the help of DNR, we can get not only better visualized results from t-SNE but also better clustering results of GDL.

We compare the clustering performances with or without RL on MINIST in Table 6. As we can see, the clustering performance is improved by RL. This phenomenon illustrates that RL is very important to improve not only the visualization performance but also the clustering performance.

Table 2
Clustering results of different denoising methods on k-means.

Dataset	Original data		MD		RMD		SAF		DNR	
	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
circles-G	0.001	0.590	0.008	0.552	0.001	0.500	0.250	0.683	0.010	0.562
circles-U	0.001	0.502	0.032	0.588	0.001	0.508	0.247	0.690	0.002	0.510
moons-G	0.356	0.509	0.284	0.803	0.457	0.876	0.792	0.967	0.400	0.854
moons-U	0.386	0.848	0.561	0.896	0.375	0.844	0.692	0.942	0.395	0.852
wine	0.800	0.932	0.866	0.966	0.796	0.932	0.825	0.944	0.884	0.966
iris	0.741	0.886	0.754	0.907	0.816	0.940	0.816	0.940	0.815	0.934
Reuters	0.288	0.535	0.240	0.578	0.233	0.530	0.330	0.547	0.444	0.726
Jaffe	0.821	0.756	0.872	0.793	0.859	0.882	0.822	0.727	0.908	0.930
MNIST	0.496	0.561	0.554	0.582	0.500	0.563	0.452	0.542	0.593	0.592

Table 3
Clustering results of different denoising methods on DPC.

Dataset	Original data		MD		RMD		SAF		DNR	
	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
circles-G	0.012	0.566	0.150	0.603	0.278	0.721	0.250	0.683	0.375	0.770
circles-U	0.017	0.575	0.325	0.746	0.004	0.530	0.228	0.660	0.273	0.696
moons-G	0.842	0.976	0.535	0.886	0.835	0.975	0.323	0.758	0.351	0.810
moons-U	0.416	0.817	0.428	0.821	0.442	0.817	0.417	0.821	0.439	0.853
wine	0.546	0.780	0.697	0.859	0.764	0.910	0.796	0.938	0.884	0.966
iris	0.658	0.553	0.754	0.907	0.730	0.866	0.763	0.886	0.783	0.887
Reuters	0.077	0.425	0.335	0.572	0.140	0.428	0.133	0.430	0.205	0.531
Jaffe	0.853	0.835	0.811	0.652	0.893	0.892	0.881	0.803	0.908	0.930
MNIST	0.307	0.350	0.580	0.519	0.302	0.337	0.480	0.431	0.587	0.574

Table 4
Clustering results of different denoising methods on GDL.

Dataset	Original data		MD		RMD		SAF		DNR	
	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
circles-G	0.265	0.708	0.073	0.554	0.174	0.625	0.700	0.935	0.807	0.969
circles-U	0.313	0.727	0.058	0.523	0.990	0.999	0.948	0.993	0.990	0.999
moons-G	0.006	0.564	0.091	0.568	0.028	0.507	0.669	0.936	0.725	0.950
moons-U	0.821	0.972	0.115	0.573	0.618	0.923	0.872	0.978	0.867	0.982
wine	0.753	0.910	0.316	0.461	0.722	0.893	0.578	0.618	0.784	0.927
iris	0.885	0.967	0.754	0.907	0.743	0.833	0.763	0.887	0.880	0.967
Reuters	0.077	0.370	0.254	0.523	0.100	0.344	0.077	0.370	0.100	0.344
Jaffe	0.912	0.803	0.842	0.770	0.859	0.747	0.698	0.500	0.908	0.923
MNIST	0.785	0.808	0.476	0.468	0.772	0.800	0.616	0.516	0.761	0.853

Table 5
Clustering results of different denoising methods on U-SENC.

Dataset	Original data		MD		RMD		SAF		DNR	
	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
circles-G	0.036	0.610	0.011	0.559	0.403	0.809	0.700	0.935	0.807	0.969
circles-U	0.989	0.999	0.183	0.699	0.987	0.998	0.948	0.993	0.990	0.999
moons-G	0.450	0.872	0.221	0.678	0.284	0.713	0.762	0.961	0.838	0.976
moons-U	0.701	0.938	0.001	0.510	0.316	0.817	0.870	0.982	0.867	0.982
wine	0.825	0.944	0.412	0.680	0.455	0.747	0.158	0.601	0.886	0.972
iris	0.732	0.853	0.736	0.907	0.497	0.733	0.702	0.806	0.750	0.893
Reuters	0.183	0.382	0.189	0.436	0.138	0.356	0.161	0.355	0.166	0.437
Jaffe	0.915	0.863	0.871	0.788	0.878	0.774	0.906	0.920	0.908	0.923
MNIST	0.642	0.706	0.492	0.518	0.647	0.621	0.609	0.593	0.728	0.726

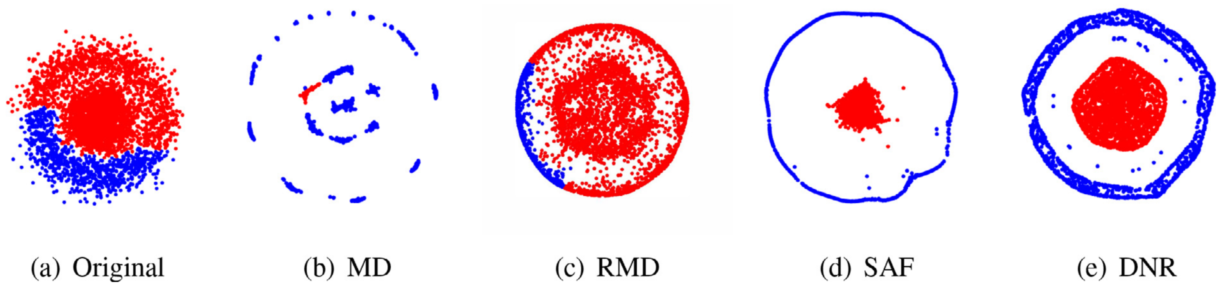


Fig. 10. Visualized results of clustering circles.

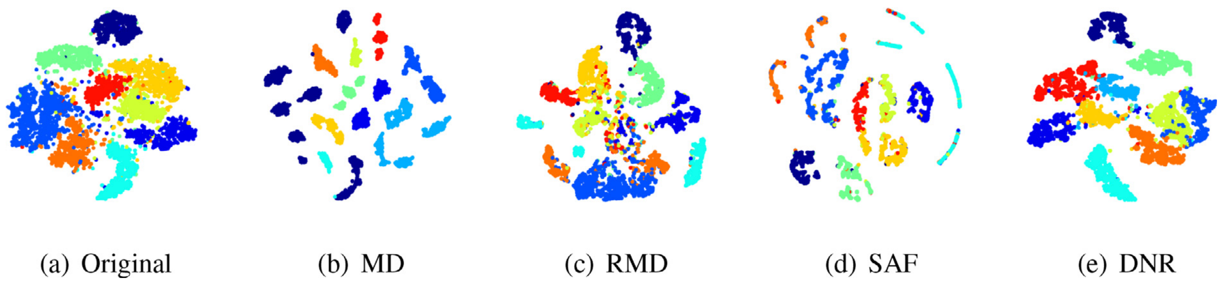


Fig. 11. Visualized results of clustering MNIST.

5.3. Parameter analysis

In Fig. 12, the sensitivities of k, b, and q are analyzed on circles-G dataset. When we show the results of k, it is selected from $\{125, 150, 175, 200\}, b$ is set as 0.005, and q is set as 0.55. When we show the results of b, it is selected from

Table 6
Comparing clustering performances with or without RL on MINIST.

Method	ACC	NMI
Without RL	0.812	0.741
DNR	0.853	0.761

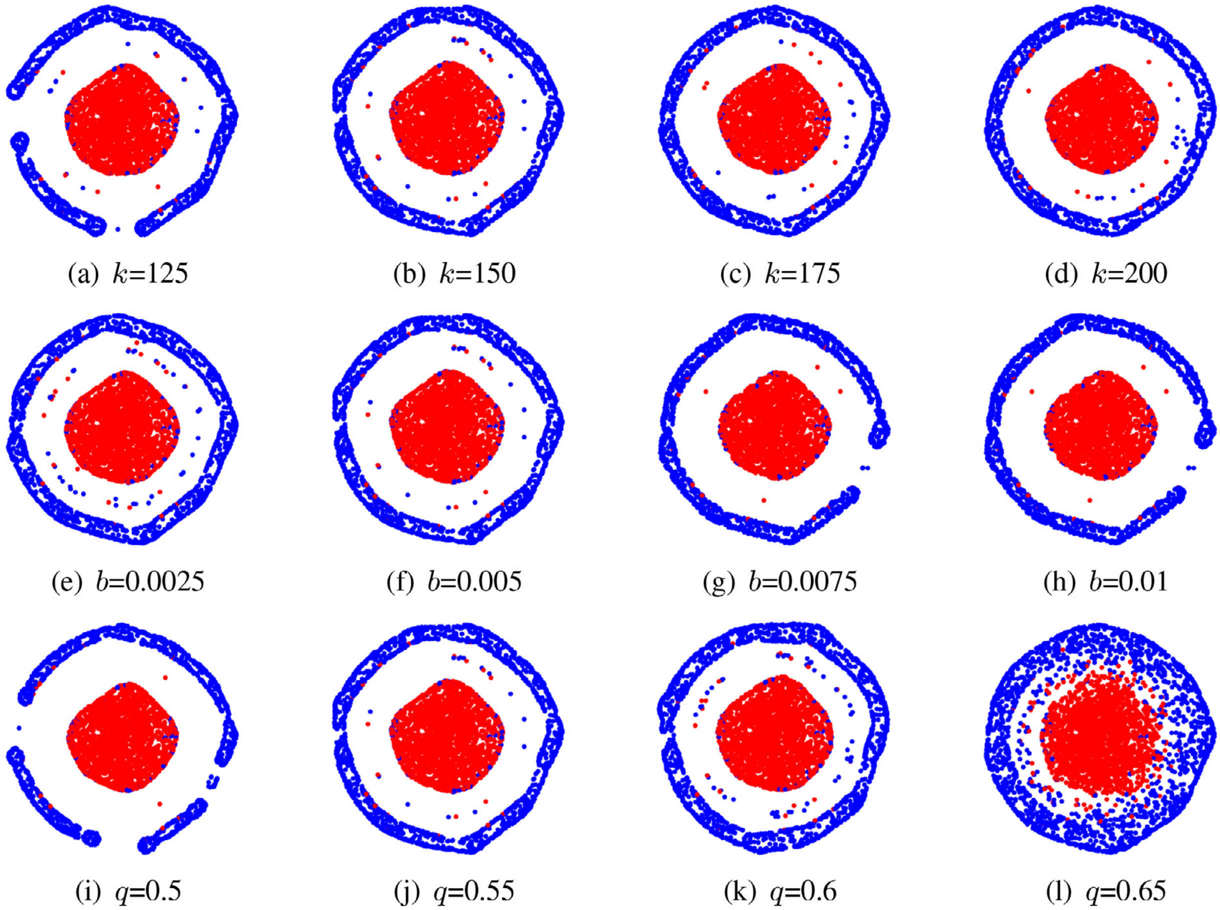


Fig. 12. The analysis of paramenter sensitivity.

$\{0.0025, 0.005, 0.075, 0.01\}$, k is set as 150, and q is set as 0.55. When we show the results of q, it is selected from $\{0.5, 0.55, 0.6, 0.65\}$, k is set as 150, and b is set as 0.005. As we can see, NDR is sensitive to q and a little sensitive to k and b.

6. Conclusion and future work

In this paper, we propose DNR method to consolidate the structures of high-noise data. Firstly, we propose DDI to detect each noisy sample in data by its density and the distance between this sample and the center of its neighbors. Then we propose DNR, which detects each noisy sample by DDI and then trains the agent of it to move this sample to its near high density region on the direction of its density gradient by TD3. In this way, these noisy samples can be adjusted to non-noisy samples and thus the structures of high-noise data can be consolidated. In the experiment, we get better visualized results and clustering results from the processed data than the corresponding original data.

There are some works that can be investigated in the future. On the one hand, DNR has it potential to deal with big data, such that we can use a small dataset to train our DNR model and then use this model to denoise big data. On the other hand, we can apply our framework to the denoising tasks in image and video by combining this framework with filter-based denoising [3,4,12].

CRediT authorship contribution statement

Tianyi Huang: Conceptualization, Methodology, Software, Writing - original draft, Visualization, Investigation. **Zhiling Cai:** Writing - review & editing. **Ruijia Li:** Writing - review & editing. **Shiping Wang:** Validation, Supervision. **William Zhu:** Funding acquisition, Writing - review & editing, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported in part by The National Nature Science Foundation of China under Grant Nos. 61772120.

References

[1] Mikhail Belkin, Partha Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. 15 (6) (2003) 1373–1396.
[2] Shalabh Bhatnagar, Doina Precup, David Silver, Richard S Sutton, Hamid R. Maei, Csaba Szepesvári, Residual algorithms: Reinforcement learning with function approximation, Advances in Neural Information Processing Systems, 22, Elsevier, 2009, pp. 1204–1212.
[3] Bo-Hao Chen, Hsiang-Yin Cheng, Yi-Syuan Tseng, Jia-Li Yin, Two-pass bilateral smooth filtering for remote sensing imagery, IEEE Geosci. Remote Sens. Lett. 19 (2021) 1–5.
[4] Bo-Hao Chen, Yi-Syuan Tseng, Jia-Li Yin, Gaussian-adaptive bilateral filter, IEEE Signal Process. Lett. 27 (2020) 1670–1674.
[5] Li Deng, The mnist database of handwritten digit images for machine learning research, IEEE Signal Process. Mag. 29 (6) (2012) 141–142.
[6] Shay Deutsch, Antonio Ortega, Gerard Medioni, Manifold denoising based on spectral graph wavelets, in: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) IEEE, 2016, pp. 4673–4677.
[7] Shay Deutsch, Antonio Ortega, Gérard Medioni, Robust denoising of piece-wise smooth manifolds, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) IEEE, 2018, pp. 2786–2790.
[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xu. Xiaowei, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA, 1996, pp. 226–231.
[9] Jerome H Friedman, Forest Baskett, Leonard J Shustek, An algorithm for finding nearest neighbors, IEEE Trans. Comput. 100 (10) (1975) 1000–1006.
[10] Scott Fujimoto, Herke Hoof, David Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning, 2018, pp. 1582–1591.
[11] Ryosuke Furuta, Naoto Inoue, Toshihiko Yamasaki, Fully convolutional network with multi-step reinforcement learning for image processing, Proceedings of the AAAI Conference on Artificial Intelligence 33 (2019) 3598–3605.
[12] Ruturaj G Gavaskar, Kunal N Chaudhury, Fast adaptive bilateral filtering, IEEE Trans. Image Process. 28 (2) (2018) 779–790.
[13] Alexander Grigoryan, Heat kernel and analysis on manifolds, volume 47, Am. Math. Soc. (2009).
[14] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In IJCAI, pages 1753–1759, 2017.
[15] Zhishuai Guo, Tianyi Huang, Zhiling Cai, William Zhu, A new local density for density peak clustering, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2018, pp. 426–438.
[16] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, Sergey Levine, Reinforcement learning with deep energy-based policies, in: International Conference on Machine Learning PMLR, 2017, pp. 1352–1361.
[17] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, Sergey Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, vol. 80, Stockholm, Sweden, 2018, pp. 1861–1870.
[18] Sariel Har-Peled, Dan Roth, and Dav Zimak. Maximum margin coresets for active and noise tolerant learning. In IJCAI, pages 836–841, 2007.
[19] John A Hartigan, Clustering algorithms, Wiley (1975).
[20] Xiaofei He, Deng Cai, Partha Niyogi, Laplacian score for feature selection, Advances in Neural Information Processing Systems 18 Annual Conference on Neural Information Processing Systems 2005, 5–8, Vancouver, British Columbia, Canada, December 2005, pp. 507–514.
[21] Robert Hecht-Nielsen et al. Theory of the backpropagation neural network. Neural Networks, 1(Supplement-1):445–448, 1988.
[22] Matthias Hein, Markus Maier, Manifold denoising, Advances in Neural Information Processing Systems 19 (2007) 561–568.
[23] Alexander Hinneburg, Daniel A Keim, et al, An efficient approach to clustering in large multimedia databases with noise, KDD 98 (1998) 58–65.
[24] Dong Huang, Chang-Dong Wang, Jiansheng Wu, Jian-Huang Lai, and Chee Keong Kwoh. Ultra-scalable spectral clustering and ensemble clustering. IEEE Trans. Knowl. Data Eng., 2019.
[25] Tianyi Huang, Min Li, Xiaolong Qin, William Zhu, A cnn-based policy for optimizing continuous action control by learning state sequences, Neurocomputing 468 (2022) 286–295.
[26] Tianyi Huang, Shiping Wang, William Zhu, An adaptive kernelized rank-order distance for clustering non-spherical data with high noise, Int. J. Mach. Learn. Cybern. (2020) 1–13.
[27] Harold W. Kuhn, The hungarian method for the assignment problem. In 50 Years of Integer Programming 1958–2008 - From the Early Years to the State-of-the-Art, Oxford University Press, 2010, pp. 29–47.
[28] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.
[29] Michael L. Littman, Markov games as a framework for multi-agent reinforcement learning, in: Machine learning proceedings 1994, Elsevier, 1994, pp. 157–163.
[30] He Lyu, Ningyu Sha, Shuyang Qin, Ming Yan, Yuying Xie, and Rongrong Wang. Manifold denoising by nonlinear robust principal component analysis. In Advances in Neural Information Processing Systems, pages 13390–13400, 2019.
[31] Laurens van der Maaten, Geoffrey Hinton, Visualizing data using t-sne, J. Mach. Learn. Res. 9 (11) (2008) 2579–2605.
[32] James MacQueen, Some methods for classification and analysis of multivariate observations, Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, Oakland, CA, USA, 1967, pp. 281–297.
[33] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.
[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, 2015.

[35] Zhen Ni, Haibo He, Jinyu Wen, Adaptive learning in tracking control based on the dual critic network design, IEEE Trans. Neural Networks Learn. Syst. 24 (6) (2013) 913–928.
[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[37] Alex Rodriguez, Alessandro Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[38] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587):484–489, 2016.
[39] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[40] Richard S Sutton, Andrew G Barto, Reinforcement learning: An introduction, MIT press, 2018.
[41] Robert Choate Tryon, Cluster analysis: Correlation profile and orthometric (factor) analysis for the isolation of unities in mind and personality, Incorporated, lithoprinters and publishers, Edwards brother, 1939.
[42] Sofie Verbaeten and Anneleen Van Assche. Ensemble methods for noise elimination in classification problems. In International workshop on multiple classifier systems, pages 317–325. Springer, 2003.
[43] Lu Wang, Wenchao Yu, Wei Wang, Wei Cheng, Wei Zhang, Hongyuan Zha, Xiaofeng He, and Haifeng Chen. Learning robust representations with graph denoising policy network. In 2019 IEEE International Conference on Data Mining (ICDM), pages 1378–1383. IEEE, 2019.
[44] Weiran Wang, Miguel A Carreira-Perpinán, Manifold blurring mean shift algorithms for manifold denoising, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition IEEE, 2010, pp. 1759–1766.
[45] Ronald J Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (3–4) (1992) 229–256.
[46] Wu. Shihao, Peter Bertholet, Hui Huang, Daniel Cohen-Or, Minglun Gong, Matthias Zwicker, Structure-aware data consolidation, IEEE Trans. Pattern Anal. Mach. Intell. 40 (10) (2017) 2529–2537.
[47] Liang Xiao, Geyi Sheng, Sicong Liu, Huaiyu Dai, Mugen Peng, Jian Song, Deep reinforcement learning-enabled secure visible light communication against eavesdropping, IEEE Trans. Commun. 67 (10) (2019) 6994–7005.
[48] Yu. Ke, Chao Dong, Liang Lin, Chen Change Loy, Crafting a toolchain for image restoration by deep reinforcement learning, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 2443–2452.
[49] Wei Zhang, Xiaogang Wang, Deli Zhao, Xiaoou Tang, Graph degree linkage: Agglomerative clustering on a directed graph, in: 12th European Conference on Computer Vision, Springer, 2012, pp. 428–441.
[50] William Zhu, Fei-Yue Wang, Reduction and axiomization of covering generalized rough sets, Inform. Sci. 152 (2003) 217–230.