Automating DBSCAN via Deep Reinforcement Learning

Ruitong Zhang\(^1\), Hao Peng\(^1\), Yingtong Dou\(^2\), Jia Wu\(^3\), Qingyun Sun\(^1\), Jingyi Zhang\(^1\), Philip S. Yu\(^2\)

\(^1\)Beihang University; \(^2\)University of Illinois Chicago; \(^3\)Macquarie University

\(^{∗}\)Corresponding author (penghao@buaa.edu.cn)

ABSTRACT

DBSCAN is widely used in many scientific and engineering fields because of its simplicity and practicality. However, due to its high sensitivity parameters, the accuracy of the clustering result depends heavily on practical experience. In this paper, we first propose a novel Deep Reinforcement Learning guided automatic DBSCAN parameters search framework, namely DRL-DBSCAN. The framework models the process of adjusting the parameter search direction by perceiving the clustering environment as a Markov decision process, which aims to find the best clustering parameters without manual assistance. DRL-DBSCAN learns the optimal clustering parameter search policy for different feature distributions via interacting with the clusters, using a weakly-supervised reward training policy network. In addition, we also present a recursive search mechanism driven by the scale of the data to efficiently and controllably process large parameter spaces. Extensive experiments are conducted on five artificial and real-world datasets based on the proposed four working modes. The results of offline and online tasks show that the DRL-DBSCAN not only consistently improves DBSCAN clustering accuracy by up to 26% and 25% respectively, but also can stably find the dominant parameters with high computational efficiency. The code is available at GitHub.

CCS CONCEPTS

- Information systems → Clustering; - Computing methodologies → Artificial intelligence.

KEYWORDS

Density-based clustering, hyperparameter search, deep reinforcement learning, recursive mechanism

ACM Reference Format:
Ruitong Zhang\(^1\), Hao Peng\(^1\), Yingtong Dou\(^2\), Jia Wu\(^3\), Qingyun Sun\(^1\), Jingyi Zhang\(^1\), Philip S. Yu\(^2\). 2022. Automating DBSCAN via Deep Reinforcement Learning. In Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM’22). ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXX

1 INTRODUCTION

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [15] is a typical density based clustering method that determines the cluster structure according to the tightness of the sample distribution. It automatically determines the number of final clusters according to the nature of the data, has low sensitivity to abnormal points, and is compatible with any cluster shape. In terms of application areas, benefiting from its strong adaptability to datasets of unknown distribution, DBSCAN is the preferred solution for many clustering problems, and has achieved robust performance in fields such as financial analysis [27, 54], commercial research [18, 52], urban planning [37, 43], seismic research [19, 31, 50], recommender system [25, 34], genetic engineering [20, 41], etc.

However, the two global parameters of DBSCAN, the distance of the cluster formation $Eps$ and the minimum data objects required inside the cluster $MinPts$, that need to be manually specified, bring Three challenges to its clustering process. First, parameters free challenge. $Eps$ and $MinPts$ have considerable influence on the clustering effect, but it needs to be determined a priori. The method based on the $k$-distance [39, 40] estimates the possible values of the $Eps$ through significant changes in the curve, but it still needs to manually formulate $Minpts$ parameters in advance. Although some improved DBSCAN methods avoid the simultaneous adjustment of $Eps$ and $MinPts$ to varying degrees, but they also necessary to pre-determine the cutoff distance parameter [13], grid parameter [12], Gaussian distribution parameter [47] or fixed $Minpts$ parameter [2, 26]. Therefore, the first challenge is how to perform DBSCAN clustering without tuning parameters based on expert knowledge. Second, adaptive policy challenge. Due to the different data distributions and cluster characteristics in clustering tasks, traditional DBSCAN parameter searching methods based on fixed patterns [39, 40] encounter bottlenecks in the face of unconventional data problems. Moreover, hyperparameter optimization methods [5, 30, 36] using external clustering evaluation index based on label information as objective functions are not effective in the
absence of data label information. The methods [35, 57] that only use the internal clustering evaluation index as the objective function, are limited by the accuracy problem, despite that they do not require label information. In addition, for streaming data that needs to be clustered continuously but the data distribution is constantly changing, the existing DBSCAN parametric search methods do not focus on how to use past experience to adaptively formulate search policies for newly arrived data. Thus, to effectively and adaptively adjust the parameter search policy of the data and be compatible with the lack of label information is regarded as the second challenge. **Third, computational complexity challenge.** Furthermore, the parameter search is limited by the parameter space that cannot be estimated. Searching too many invalid parameters will increase the search cost [29], and too large search space will bring noise interfering with clustering accuracy [14]. Hence, how to quickly search for the optimal parameters while securing clustering accuracy is the third challenge that needs to be addressed.

In recent years, Deep Reinforcement Learning (DRL) [22, 38] has been widely used for tasks lacking training data due to its ability to learn by receiving feedback from the environment [44]. In this paper, to handle the problem of missing optimal DBSCAN parameter labeling and the challenges above, we propose DRL-DBSCAN, a novel, adaptive, recursive Deep Reinforcement Learning DBSCAN parameter search framework, to obtain optimal parameters in multiple scenarios and tasks stably. We first take the cluster changes after each step of clustering as the observable state, the parameter adjustment direction as the action, and transform the parameter search process into a maze game problem [8, 56] in the parameter space, set, action space, reward function and policy optimization algorithm. In this section, we give the definitions of DBSCAN clustering, parameter search of DBSCAN clustering, and parameter search in data stream clustering.

**Definition 1. (DBSCAN clustering).** The DBSCAN clustering is the process of finding the optimal parameter combination $P = \{\text{Eps}, \text{MinPts}\}$ for clustering in all possible parameter spaces. Here, the feature set $X$ of data objects in block $V$ is $\{x_1,...,x_j,x_{j+1},...\}$.

**Definition 2. (Parameter search in offline DBSCAN clustering).** Given the data block $V = \{v_1,...,v_j,v_{j+1},...\}$ for all data objects $\{v_1,...,v_j,v_{j+1},...\}$ in a data block $V$ based on the parameter combination $P = \{\text{Eps}, \text{MinPts}\}$. Here, Eps is the maximum distance that two adjacent objects can form a cluster, and MinPts refers to the minimum number of adjacent objects within Eps that an object can be a core point. The formation of the clusters can be understood as the process of connecting the core points to its adjacent objects within Eps [15] (as shown in Fig. 1).

**Definition 3. (Parameter search in online DBSCAN clustering).** Given continuous and temporal $T$ data blocks $\{V_t,...,V_1\}$ as the online data stream, we define the parameter search in online clustering as the process of obtaining the parameter combination $P_t = \{\text{Eps}_t, \text{MinPts}_t\}$ of the data block $V_t = \{v_{t,1},...,v_{t,j},v_{t,j+1},...\}$ at each time $t \in T$.

### 3 DRL-DBSCAN FRAMEWORK

The proposed DRL-DBSCAN has a core model (Fig. 2) and four working modes (Fig. 3) that can be extended to downstream tasks. We firstly describe the basic Markov decision process for parameter search (Sec. 3.1), and the definition of the clustering parameter space and recursion mechanism (Sec. 3.2). Then, we explain the four DRL-DBSCAN working modes (Sec. 3.3).

#### 3.1 Parameter Search with DRL

Faced with various clustering tasks, the fixed DBSCAN parameter search policy no longer has flexibility. We propose an automatic parameter search framework DRL-DBSCAN based on Deep Reinforcement Learning (DRL), in which the core model can be expressed as a Markov Decision Process $MDP(S,A,R,P)$ including state set, action space, reward function and policy optimization algorithm [42]. This process transforms the DBSCAN parameter search process into a maze game problem [8, 56] in the parameter space, aiming to train an agent to search for the end point parameters step by step from the start point parameters by interacting with the environment, and take the end point (parameters in the last step) as the final search result of an episode of the game (as shown in Fig. 2). Specifically, the agent regards the parameter space and DBSCAN clustering algorithm as the environment, the search position
and clustering result as the state, and the parameter adjustment direction as the action. In addition, a small number of samples are used to reward the agent with exceptional behavior in a weakly supervised manner. We optimize the policy of agent based on the Actor-Critic [33] architecture. Specifically, the search process for episode \( e = 1, 2, \ldots \) is formulated as follows:

**State:** Since the state needs to represent the search environment at each step as accurately and completely as possible, we consider building the representation of the state from two aspects.

Firstly, for the overall searching and clustering situation, we use a 7-tuple to describe the global state of the \( i \)-th step (\( i = 1, 2, \ldots \)):

\[
s^{(e)(i)}_{\text{global}} = P^{(e)(i)} \cup S^{(e)(i)} \cup \left\{ R^n_{c_{\text{cen}}} \right\}.
\]  

(1)

Here, \( P^{(e)(i)} \) is the set of quaternary distances, including the distances of \( Ep_s^{(e)(i)} \) from its space boundaries \( B_{Eps,1} \) and \( B_{Eps,2} \), the distances of \( MinPts_{s}^{(e)(i)} \) from its boundaries \( B_{MinPts,1} \) and \( B_{MinPts,2} \), \( C^{(e)(i)}_{\text{cen}} \) is the ratio of the number of clusters \( |C^{(e)(i)}_{\text{cen}}| \) to the total object number of data block \( |V| \). Here, the specific boundaries of parameters will be defined in Sec. 3.2.

Secondly, for the description of the situation of each cluster, we define the \( \{d + 2\} \)-tuple of the \( i \)-th local state of cluster \( c_n \in C \) as:

\[
s^{(e)(i)}_{\text{local},n} = X^{(e)(i)}_{\text{cen},n} \cup \left\{ D^{(e)(i)}_{\text{cen},n} \mid |c_n^{(e)(i)}| \right\}.
\]  

(2)

Here, \( X^{(e)(i)}_{\text{cen},n} \) is the central object feature of the \( c_n \), and \( d \) is its feature dimension. \( D^{(e)(i)}_{\text{cen},n} \) is the Euclidean distance from the cluster center object to the center object of the entire data block. \( |c_n^{(e)(i)}| \) means the number of objects contained in cluster \( c_n \).

Considering the change of the number of clusters at different steps in the parameter search process, we use the Attention Mechanism [48] to encode the global state and multiple local states into a fixed-length state representation:

\[
s^{(e)(i)} = \sigma \left( F_G(s^{(e)(i)}_{\text{global}}) \parallel \sum_{c_n \in C} a_{\text{att},n} \cdot F_L(s^{(e)(i)}_{\text{local},n}) \right),
\]  

(3)

where \( F_G \) and \( F_L \) are the Fully-Connected Network (FCN) for the global state and the local state, respectively. \( \sigma \) represents the ReLU activation function. And \( \parallel \) means the operation of splicing. \( a_{\text{att},n} \) is the attention weight of cluster \( c_n \), which is formalized as follows:

\[
a_{\text{att},n} = \sigma \left( F_S(F_G(s^{(e)(i)}_{\text{global}})) \parallel F_L(s^{(e)(i)}_{\text{global}}) \right) \cdot \frac{1}{\sum_{c_n \in C} \sigma(F_S(F_G(s^{(e)(i)}_{\text{global}})) \parallel F_L(s^{(e)(i)}_{\text{local},n})\cdot|c_n|)}. \]  

(4)

We concatenate the global state with the local state of each cluster separately, input it into a fully connected network \( F_S \) for scoring, and use the normalized score of each cluster as its attention coefficient. This approach establishes the attention to the global search situation when local clusters are expressed. At the same time, it also makes different types of cluster information have different weights in the final state expression, which increases the influence of important clusters on the state.

**Action:** The action \( a^{(e)(i)} \) for the \( i \)-th step is the search direction of parameter. We define the action space \( \mathcal{A} \) as \{left, right, down, up, stop\}, where left and right means to reduce or increase the \( Eps \) parameter, down and up means to reduce or increase the MinPts parameter, and stop means to stop searching. Specifically, we build an Actor [33] as the policy network to decide action \( a^{(e)(i)} \) based on the current state \( s^{(e)(i)} \):

\[
a^{(e)(i)} = \text{Actor}(s^{(e)(i)}).
\]  

(5)
where
\[ R(e) \]

Therefore, we consider using both the maximum immediate reward
\[ M_{\text{Max}} \] and episode end point parameters. And
\[ \text{reward} \]

Here, the labels are only used in the training process, and the testing

\[ Y \]

\begin{equation}
    R(s^{(e)}(i), a^{(e)}(i)) = NMI(DBSCAN(X, P^{(e)}(i+1)), Y').
\end{equation}

Here, \( NMI(\cdot) \) stands for the external metric function, Normalized
Mutual Information (NMI) [16] of DBSCAN clustering. \( X \) is the
feature set. \( Y' \) is a set of partial labels of the data block. Note that
the labels are only used in the training process, and the testing
process performs search on unseen data blocks without labels.

In addition, the excellent parameter search action sequence for
one episode is to adjust the parameters in the direction of the
optimal parameters, and stop the search at the optimal parameters.
Therefore, we consider using both the maximum immediate reward
for subsequent steps and the endpoint immediate reward as the
reward in the \( i \)-th step:

\begin{equation}
    r^{(e)}(i) = \beta \cdot \max \{ R(s^{(e)}(m), a^{(e)}(m)) \} + \delta \cdot R(s^{(e)}(i), a^{(e)}(i)),
\end{equation}

where \( R(s^{(e)}(i), a^{(e)}(i)) \) is the immediate rewards for the \( i \)-th step
end point parameters. And \( \max \) is used to calculate the future
maximum immediate reward before stopping the search in the current
episode \( e \). \( \beta \) and \( \delta \) are the impact factors of reward, where \( \beta = 1 - \delta \).

- **Termination**: We determine the termination conditions for a
complete episode search process as follows:

\[
\min (\bar{D}_b^{(e)}(i)) < 0, \quad \text{Out of bounds stop,} \\
i >= I_{\text{Max}}, \quad \text{Timeout stop,} \\
\text{a}^{(e)}(i) = \text{stop, where } i \geq 2, \quad \text{Active stop.}
\]

Here, \( I_{\text{Max}} \) is the maximum search size in an episode.

- **Optimization**: The parameter search process in the episode \( e \)
is expressed as: 1) observe the current state \( s^{(e)}(i) \) of DBSCAN
clustering; 2) predict the action \( a^{(e)}(i) \) of the parameter
adjustment direction based on \( s^{(e)}(i) \) through the \( \text{Actor} \); 3) then,
the new state \( s^{(e)}(i+1) \); 4) repeat the above process until the end of
episode, and get reward \( r^{(e)}(i) \) for each step. The core element of
the \( i \)-th step is extracted as:

\begin{equation}
    T^{(e)}(i) = (s^{(e)}(i), a^{(e)}(i), s^{(e)}(i+1), r^{(e)}(i)).
\end{equation}

We put \( T \) of each step into the memory buffer and sample \( M \) core
elements to optimize the policy network \( \text{Actor} \), and define the key
loss functions as follows:

\begin{equation}
    L_c = \sum_{i=0}^{M} \mathcal{L}_{\text{Buffer}}(r^{(e)}(i) + y \cdot \text{Crit}ic(s^{(e)}(i+1), a^{(e)}(i+1)) - \text{Val}(s^{(e)}(i), a^{(e)}(i)))^2,
\end{equation}

\begin{equation}
    L_a = - \frac{\sum_{i=0}^{M} \mathcal{L}_{\text{Buffer}} \text{Crit}ic(s^{(e)}(i), \text{Actor}(s^{(e)}(i)))}{M}.
\end{equation}

Here, we define a three-layer MLP as the \( \text{Crit}ic \) to learn the action
value of state [33], which is used to optimize the \( \text{Actor} \). And \( y \) means
reward decay factor. Note that we use the policy optimization algorithm
named Twin Delayed Deep Deterministic strategy gradient algorithm (TD3) [22] in our framework, and it can be replaced with
other DRL policy optimization algorithms [33, 38].

### 3.2 Parameter Space and Recursion Mechanism

In this section, we will define the parameter space of the agent
proposed in the previous section and the recursive search mechanism
based on different parameter spaces. Firstly, in order to deal
with the fluctuation of the parameter range caused by different data
distributions, we normalize the data features, thereby transforming
the maximum \( Eps \) parameter search range into the \( [0, \sqrt{D}] \) range.
Unlike \( Eps \), the \( \text{MinPts} \) parameter must be an integer greater than
0. Therefore, we propose to delineate a coarse-grained \( \text{MinPts} \) maximum
parameter search range according to the size or dimension of
the dataset. Subsequently, considering the large parameter space
affects the efficiency when performing high-precision parameter
search, we propose to use a recursive mechanism to perform a
progressive search. The recursive process is shown in Fig. 2. We
narrow the search range and increase the search precision layer
delay, and assign a parameter search agent \( \text{agent}^{(l)}(0) \) defined in
Sec. 3.1 to each layer \( l \) to search for the optimal parameter
combination \( P^{(l)}(0) = \{Eps^{(l)}, \text{MinPts}^{(l)}(0)\} \) under the requirements of
the search precision and range of the corresponding layer. The
minimum search boundary \( B_p^{(l)} \) and maximum search boundary \( E_p^{(l)} \)
of parameter \( p \in \{ \text{Eps}, \text{MinPts} \} \) in the \( l \)-th layer \((l = 1, 2, \ldots)\) are defined as:
\[
\begin{align*}
B_{p, 1}^{(l)} &= \max \left\{ b_{p, 1}^{(0)}, p_{o}^{(l-1)} - \frac{\pi_p}{2} \theta_{p}^{(l)} \right\}, \\
B_{p, 2}^{(l)} &= \min \left\{ p_{o}^{(l-1)} + \frac{\pi_p}{2} \theta_{p}^{(l)}, b_{p, 2}^{(0)} \right\}.
\end{align*}
\]
(13)
Here, \( \pi_p \) is the number of searchable parameters in the parameter space of parameter \( p \) in each layer. \( B_{p, 1}^{(l)} \) and \( B_{p, 2}^{(l)} \) are the space boundaries of parameter \( p \) in the 0-th layer, which define the maximum parameter search range. \( p_{o}^{(l-1)} \) is the optimal parameter searched by the previous layer, and \( p_{o}^{(0)} \) is the midpoint of \( B_{p, 1}^{(0)} \) and \( B_{p, 2}^{(0)} \). In addition, \( \theta_{p}^{(l)} \) is the step size in the search process, which is defined as follows:
\[
\theta_{p}^{(l)} = \begin{cases} 
\frac{\theta_{p}^{(l-1)}}{\pi_p}, & \text{if } p = \text{Eps}; \\
\max \left\{ \frac{\theta_{p}^{(l-1)}}{\pi_p} + \frac{1}{2}, 1 \right\}, & \text{otherwise}. 
\end{cases}
\]
(14)
Here, \( \theta_{p}^{(l-1)} \) is the step size of the previous layer and \( \theta_{p}^{(0)} \) is the size of the parameter maximum search range. \( |\cdot| \) means round down.

**Complexity Discussion:** It is known that the minimum search step size of the recursive structure with layers \( L \) is \( \theta_{p}^{(L)} \). Then the computational complexity when there is no recursive structure is \( O(N) \), where \( N \) is the size of the parameter space \( \theta_{p}^{(0)}/\theta_{p}^{(L)} = (\pi_p)^L \).

And DRL-DBSCAN with \( L \)-layer recursive structure only takes \( O(L \cdot (\pi_p)) \), reducing the complexity from \( O(N) \) to \( O(\log N) \).

### 3.3 Proposed DRL-DBSCAN

Algorithm 1 shows the process of the proposed DRL-DBSCAN core model. Given a data block \( V \) with partial label \( Y' \), the training process repeats the parameter search process (Lines 6-10) for multiple episodes at each layer to optimize the agent (Line 14). In this process, we update the optimal parameter combination (Line 15 and Line 17 based on the immediate reward (Eq. (7))). In order to improve efficiency, we build a hash table to record DBSCAN clustering results of searched parameter combinations. In addition, we established early stopping mechanisms to speed up the training process when the optimal parameter combination does not change (Line 16 and Line 18). It is worth noting that the testing process uses the trained agents to search directly with one episode, and does not set the early stop. Furthermore, the testing process does not need labels, and the end point parameters of the unique episode of the last layer are used as the final optimal parameter combination.

In order to better adapt to various task scenarios, we define four working modes of DRL-DBSCAN as shown in Fig. 3. Their corresponding definitions are as follows: (1) **Retraining Mode (DRLre)**. The optimal parameters are searched based on the training process. When the dataset changes, the agent at each layer is reinitialized. (2) **Continuous Training Mode (DRLcon)**. The agents are pre-trained in advance. When the dataset changes, continue searching based on the training process using the already trained agents. (3) **Pretraining Testing Mode (DRLpre)**. The agents are pre-trained in advance. When the dataset changes, searching directly based on the testing process without labels. (4) **Maintenance Testing Mode (DRLm)**. The agents are pre-trained in advance. When the dataset changes, searching directly based on the testing process without labels. After pre-training, regular maintenance training is performed with labeled historical data.

### 4 EXPERIMENTS

In this section, we conduct experiments mainly including the following: (1) performance comparison over DRL-DBSCAN and baseline in offline tasks, and explanation of the search process for Reinforcement Learning (RL) (Sec. 4.2); (2) performance analysis of DRL-DBSCAN and its variants, and advantage comparison between four working modes in online tasks (Sec. 4.3); (3) sensitivity analysis of hyperparameters and their impact on the model (Sec. 4.4).

#### 4.1 Experiment Setup

**Datasets.** To fully analyze our framework, the experimental dataset consists of 4 artificial clustering benchmark datasets and 1 public real-world streaming dataset (Table 1). The benchmark datasets [21] are the 2D shape sets, including: Aggregation [23], Compound [55], Pathbased [11], and D31 [49]. They involve multiple density types such as clusters within clusters, multi-density, multi-shape, closed loops, etc., and have various data scales. Furthermore, the real-world streaming dataset Sensor [58] comes from consecutive information (temperature, humidity, light, and sensor voltage) collected from 54 sensors deployed by the Intel Berkeley Research Lab. We use a subset of 80,864 for experiments and divide these objects into 16 data blocks \( \{Y'_1, \ldots, Y'_8 \} \) as an online dataset.

**Baseline and Variants.** We compare proposed DRL-DBSCAN with three types of baselines: (1) traditional hyperparameter search
schemes: random search algorithm Rand [6], Bayesian optimization based on Tree-structured Parzen estimator algorithm BO-TPE [5]; (2) meta-heuristic optimization algorithms: the simulated annealing optimization Anneal [32], particle swarm optimization PSO [46], genetic algorithm GA [36], and differential evolution algorithm DE [45]; (3) existing DBSCAN parameter search methods: KDist (V-DBSCAN) [39] and BDE-DBSCAN [30]. The detailed introduction to the above methods are given in Sec. 5. We also implement four variants of DRL-DBSCAN to analysis of state, reward and recursive mechanism settings in Sec. 3.1. Compared with DRL-DBSCAN, DRL\textsubscript{no-att} does not join local state based on the attention mechanism. DRL\textsubscript{only-max} only uses the maximum future immediate reward as final reward, DRL\textsubscript{recu} has no early stop mechanism, and DRL\textsubscript{recu} has no recursion mechanism.

**Implementation Details.** For all baselines, we use open-source implementations from the benchmark library Hyperopt [7] and Scikit-opt [1], or provided by the author. All experiments are conducted on Python 3.7, 36 core 3.00GHz Intel Core i9 CPU, and NVIDIA RTX A6000 GPUs.

**Experimental Setting.** The evaluation of DRL-DBSCAN is based on the four working modes proposed in Sec. 3.3. Considering the randomness of most algorithms, all experimental results we report are the means or variances of 10 runs with different seeds (except KDist because it’s heuristic and doesn’t involve random problems). Specifically, for the pre-training and maintenance training processes of DRL-DBSCAN, we set the maximum number of episodes $E_{max}$ to 50, and do not set the early stop mechanism. For the training process for searching, we set the maximum number of episodes $E_{max}$ to 15. In offline tasks and online tasks, the maximum number of recursive layers $L_{max}$ is 3 and 6, respectively, and the maximum search boundary in the 0-th layer of $\text{MinPts}$ $p_{\text{MinPts}}^{(0)}$ is 0.25 and 0.0025 times the size of block, respectively. In addition, we use the unified label training proportion 0.2, the $E_{ps}$ parameter space size $\pi_{E_{ps}}$, 5, the $\text{MinPts}$ parameter space size $\pi_{\text{MinPts}}$, 4, the maximum number of search steps $l_{max}$ 30 and the reward factor $\delta$ 0.2. The FCN and MLP dimensions are uniformly set to 32 and 256, the reward decay factor $\gamma$ of Critic is 0.1, and the number of samples $M$ from the buffer is 16. Furthermore, all baselines use the same objective function (Eq. (7)), parameter search space, and parameter minimum step size as DRL-DBSCAN if they support the settings.

**Evaluation Metrics.** We evaluate the experiments in terms of accuracy and efficiency. Specifically, we measure the clustering accuracy based on normalized mutual information (NMI) [16] and adjusted rand index (ARI) [51]. For the efficiency, we use the consumed DBSCAN clustering rounds as the measurement.

### 4.2 Offline Evaluation

Offline evaluation is based on four artificial benchmark datasets. Since there is no data for pre-training in offline scenarios, we only compare the parameter search performance of DRL-DBSCAN using the retraining mode $DRL_{recu}$ with baselines.

**Accuracy and Stability Analysis.** In Table 2, we summarize the means and variances of the NMI and ARI corresponding to the optimal DBSCAN parameter combinations that can be searched by $DRL_{recu}$ and baselines within 30 clustering rounds. It can be seen from the mean accuracy results of ten runs that in the Pathbased, Compound, Aggregation, and D31 datasets, $DRL_{recu}$ can effectively improve the performance of 4% & 6%, 3% & 3%, 20% & 26% and 5% & 26% on NMI and ARI relative to the best performing baselines. At the same time, as the dataset size increases, the advantage of $DRL_{recu}$ compared to other baselines in accuracy gradually increases. Furthermore, the experimental variances shows that $DRL_{recu}$ improves stability by 4% & 6%, 1% & 1%, 9% & 13% and 2% & 17% on NMI and ARI, relative to the best performing baselines. The obvious advantages in terms of accuracy and stability indicate that $DRL_{recu}$ can stably find excellent parameter combinations in multiple rounds of parameter search, compared with other hyperparameter optimization baselines under the same objective function. Besides, $DRL_{recu}$ is not affected by the size of the dataset. Among all the baselines, PSO and DE are relatively worse in terms of accuracy, because their search in the parameter space is biased towards continuous parameters, requiring more rounds to achieve optimal results. BO-TPE learns previously searched parameter combinations through a probabilistic surrogate model and strikes a balance between exploration and exploitation, with significant advantages over other baselines. The proposed DRL-DBSCAN not only narrows the search space of parameters of each layer progressively through a recursive structure, but also learns historical experience, which is more suitable for searching DBSCAN clustering parameter combinations.

**Efficiency Analysis.** We present the average historical maximum NMI results for $DRL_{recu}$, and all baselines when consuming a different number of clustering rounds in Fig. 4. The shade in the figure represents the fluctuation range (variance) of NMI in multiple runs (only BO-TPE is also shown with shade as the representative in the baselines). The results suggest that in the four datasets, $DRL_{recu}$ can maintain a higher speed of finding better parameters than baselines, and fully surpass all baselines after the 5-th, 12-th, 17-th, and 16-th rounds, respectively. In the Pathbased dataset, the clustering rounds of $DRL_{recu}$ is 2.49 times faster than that of BO-TPE when the NMI of the parameter combination searched by $DRL_{recu}$ reaches 0.72. Besides, the results show that, with the increase of clustering rounds, the shadow area of the $DRL_{recu}$ curve gradually decreases, while the shadow range of BO-TPE tends to be constant. The above observation also demonstrates the good stability of the search when the number of rounds of DRL-DBSCAN reaches a specific number.

**DRL-DBSCAN Variants.** We compare the $DRL_{recu}$ with $DRL_{no-recu}$ which without the recursion mechanism in Fig. 5. Note that, for $DRL_{no-recu}$, we turn off the early stop mechanism so that it can search longer to better compare with $DRL_{no-recu}$. The results show that the first 100 episodes of the recursive mechanism bring the maximum search speedup ratio of 6.5, which effectively proves the contribution of the recursive mechanism in terms of efficiency.

**Label Proportion Comparison.** Considering the influence of the proportion of labels participating in training on the DRL-DBSCAN
### Automating DBSCAN via Deep Reinforcement Learning

CIKM’22, October 17-22, 2022, Hybrid Conference, Hosted in Atlanta, Georgia, USA

In different episodes, the termination types of the episodes, the

Meta-heuristic

| Parameter | Rand | BO-TPE | Anneal | PSO | GA | DE | KDist | BDE | DRLRec (Mean) | Var |
|-----------|------|--------|--------|-----|-----|----|-------|-----|---------------|-----|
| NMI       | 0.66±23 | 0.78±07 | 0.65±24 | 0.60±28 | 0.68±19 | 0.22±28 | 0.40±.77 | 0.51±33 | 0.82±03 | ↑0.04 |
| ARI       | 0.63±21 | 0.79±10 | 0.66±25 | 0.55±38 | 0.67±26 | 0.18±28 | 0.38±.79 | 0.48±40 | 0.85±04 | ↑0.06 |
| NMI       | 0.75±05 | 0.70±24 | 0.52±36 | 0.46±34 | 0.70±25 | 0.33±35 | 0.39±.72 | 0.72±25 | 0.78±04 | ↑0.03 |
| ARI       | 0.73±04 | 0.68±24 | 0.51±35 | 0.42±36 | 0.68±24 | 0.31±34 | 0.39±.70 | 0.70±25 | 0.76±03 | ↑0.03 |
| NMI       | 0.76±11 | 0.72±14 | 0.75±27 | 0.59±35 | 0.75±15 | 0.28±37 | 0.60±.63 | 0.63±28 | 0.96±02 | ↑0.20 |
| ARI       | 0.68±16 | 0.63±19 | 0.70±27 | 0.51±37 | 0.68±19 | 0.25±35 | 0.52±.54 | 0.54±28 | 0.96±03 | ↑0.26 |
| NMI       | 0.31±33 | 0.23±24 | 0.17±19 | 0.36±33 | 0.23±20 | 0.24±26 | 0.07±.41 | 0.41±36 | 0.67±02 | ↑0.26 |
| ARI       | 0.14±26 | 0.04±05 | 0.03±04 | 0.09±22 | 0.04±04 | 0.06±09 | 0.00±.21 | 0.28±28 | 0.26±02 | ↑0.05 |

**Table 2: Offline evaluation performance.** The best results are bolded and second-best are underlined.

### 4.3 Online Evaluation

The learnability of RL enables DRL-DBSCAN to better utilize past experience in online tasks. To this end, we comprehensively evaluate four working modes of DRL-DBSCAN on a streaming dataset, Sensor. Specifically, the first eight blocks of the Sensor are used for the pre-training of DRLRec, DRLall, and DRLone, and the last eight blocks are used to compare the results with baselines. Since baselines cannot perform incremental learning for online tasks, we initialize the algorithm before each block starts like DRLRec. In addition, both experiments of DRLall and DRLone use unseen data for unlabeled testing, whereas DRLone uses the labeled historical data for model maintenance training after each block ends testing.

### Accuracy and Stability Analysis

We give the performance comparison of training-based modes (DRLRec and DRLall) and testing-based modes (DRLall and DRLone) of DRL-DBSCAN with the baselines in Table 4 and Table 5, respectively. Due to the action space of DRL-DBSCAN has the stop action, it can automatically terminate the search. To control the synchronization of baselines’ experimental conditions, we use the average clustering rounds consumed when DRL-DBSCAN is automatically terminated as the maximum round of baselines for the corresponding task (30 for Table 4 and
Table 4: Online evaluation NMI for training-based modes. The best results are bolded and second-best are underlined.

| Blocks | Rand BO-TPE | BO-TPE | Anneal PSO GA DE | KDist BDE | DRL\textsubscript{all} DRL\textsubscript{one} | (Mean) | (Var.) |
|--------|-------------|--------|------------------|-----------|---------------------------------|---------|--------|
| $V_0$  | .67±.24     | .83±.03| .53±.37          | .74±.10   | .65±.29                         | .91±.31 | .86±.01 |
| $V_{10}$ | .36±.15    | .50±.07| .45±.17          | .50±.20   | .43±.15                         | .15±.17 | .64±.06 |
| $V_{20}$ | .40±.06    | .43±.10| .32±.26          | .55±.16   | .43±.08                         | .09±.12 | .68±.02 |
| $V_{30}$ | .44±.23    | .62±.16| .27±.35          | .66±.07   | .50±.24                         | .19±.28 | .72±.10 |
| $V_{40}$ | .84±.06    | .87±.04| .72±.38          | .68±.26   | .76±.17                         | .38±.38 | .85±.00 |
| $V_{50}$ | .74±.12    | .82±.04| .54±.37          | .63±.24   | .54±.24                         | .25±.25 | .85±.07 |
| $V_{60}$ | .68±.24    | .76±.04| .66±.34          | .55±.25   | .62±.27                         | .28±.32 | .85±.07 |
| $V_{70}$ | .73±.13    | .77±.09| .77±.10          | .40±.35   | .67±.22                         | .49±.31 | .86±.01 |

Table 5: Online evaluation NMI for testing-based modes. The best results are bolded and second-best are underlined.

| Blocks | Rand BO-TPE | BO-TPE | Anneal PSO GA DE | KDist BDE | DRL\textsubscript{all} DRL\textsubscript{one} | (Mean) | (Var.) |
|--------|-------------|--------|------------------|-----------|---------------------------------|---------|--------|
| $V_0$  | .34±.31     | .49±.33| .22±.34          | .14±.29   | .27±.37                         | .10±.26 | .68±.30 |
| $V_{10}$ | .11±.14    | .28±.17| .17±.21          | .24±.01   | .20±.21                         | .12±.18 | .33±.16 |
| $V_{20}$ | .16±.15    | .29±.24| .23±.18          | .33±.29   | .23±.23                         | .02±.05 | .33±.15 |
| $V_{30}$ | .23±.25    | .19±.24| .10±.22          | .38±.26   | .34±.27                         | .03±.06 | .46±.09 |
| $V_{40}$ | .58±.35    | .70±.24| .47±.40          | .44±.31   | .36±.28                         | .08±.14 | .70±.27 |
| $V_{50}$ | .36±.19    | .34±.28| .47±.35          | .37±.33   | .27±.25                         | .11±.24 | .60±.27 |
| $V_{60}$ | .45±.35    | .38±.36| .37±.33          | .30±.34   | .36±.32                         | .09±.18 | .70±.03 |
| $V_{70}$ | .22±.32    | .45±.24| .32±.29          | .19±.27   | .36±.27                         | .12±.20 | .60±.27 |

Efficiency Analysis. DRL\textsubscript{all} and DRL\textsubscript{one} automatically search for the optimal DBSCAN parameters (end point parameters) without labels. In order to better analyze these two testing-based parameter search modes, we compare the number of clustering rounds required of other methods to reach the NMI scores of the DRL\textsubscript{all} end-point parameters in the online tasks (Fig. 6(a)). In the figure, the short vertical lines are the result variances. We can see that DRL\textsubscript{all} reaches the optimal results within the consumption range of 11-14 rounds, while other baselines require more rounds when reaching the corresponding NMI. Moreover, many baselines’ round consumption over different blocks fluctuates significantly, and the variance in the same block is also large. The above observation suggests that the parameter search efficiency of DRL-DBSCAN’s testing-based modes without labels exceeds that of the baselines which require labels. Additionally, DRL\textsubscript{one} consumes fewer rounds than DRL\textsubscript{all} when reaching the same NMI, which also proves the advantage of DRL-DBSCAN’s learning ability in terms of efficiency.
DRL-DBSCAN Variants. To better evaluate the design of states and rewards in Sec. 3.1, we compare two variants with DRL_all in the online tasks, namely DRL_no_att (state has no attention mechanism) and DRL_only_max (reward only based on future maximum immediate reward). The results in Fig. 6(c) show that the full structure of DRL_all has better NMI scores than the variants, and brings the maximum performance increase of 0.16, which represents the necessity of setting the local state and end point immediate reward.

4.4 Hyperparameter Sensitivity

Fig. 7 shows the results of the offline evaluation of DRL_all on the Pathbased for four hyperparameters. Fig. 7(a) and Fig. 7(b) compare a set of parameter space sizes of Eps and MinPts involved in the Eq. (13), respectively. It can be found that the parameter space that is too large or too small for Eps will cause performance loss and a decrease in search efficiency, while MinPts is less sensitive to the parameter space’s size change. Fig. 7(c) analyzes the effect of different numbers of recursive layers on the search results. The results show that a suitable number of recurrent layers helps to obtain stable performance results. It is worth noting that the number of layers does not require much tuning, as we use the early-stop mechanism described in Sec. 3.3 to avoid overgrowing layers. Fig. 7(d) compares the different influence weights of end point immediate reward and future maximum immediate reward on the final reward (Eq. 8). The results show that equalizing the contribution of the two immediate rewards to the final reward can help improve the performance of the DRL-DBSCAN.

5 RELATED WORK

Automatic DBSCAN parameter determination. DBSCAN is heavily dependent on two sensitive parameters (Eps and MinPts) requiring prior knowledge for tuning. Numerous works propose different solutions for tuning the above. OPTICS [5] is an extension of DBSCAN, which establishes cluster sorting based on reachability to obtain the Eps. However, it needs to pre-determine the appropriate value of MinPts, and the acquisition of Eps needs to interact with the user. V-DBSCAN [39] and KDDClus [40] plot the curve by using the sorted distance of any object to its k-th nearest object, and use the significant change on the curve as a series of candidate values for the Eps parameter. Similar methods include DSets-DBSCAN [26], Outlier [2] and RNN-DBSCAN [10], all of which require a fixed MinPts value or a predetermined the number of nearest neighbors k, and the obtained candidate Eps parameters may not be unique. Beside the above work, there are some works [12, 13] that consider combining DBSCAN with grid clustering to judge the density trend of raw samples according to the size and shape of each data region through pre-determined grid partition parameters. Although these methods reduce the difficulty of parameter selection to a certain extent, they still require the user to decide at least one parameter heuristically, making them inflexible in changing data.

Hyperparameter Optimization. For the parameters of DBSCAN, another feasible parameter decision method is based on the Hyperparameter Optimization (HO) algorithm. The classic HO methods are model-free methods, including grid search [12] that searches for all possible parameters, and random search [6] etc. Another approach is Bayesian optimization methods such as BO-TPE [5], SMAC [28], which optimize search efficiency using prior experience. In addition, meta-heuristic optimization methods, such as simulated annealing [32], genetic [36], particle swarm [46] and differential evolution [45], can solve non-convex, non-continuous and non-smooth optimization problems by simulating physical, biological and other processes to search [53]. Based on meta-heuristic optimization algorithms, some works propose HO methods for DBSCAN. BDE-DBSCAN [30] targets an external purity index, selects MinPts parameters based on a binary differential evolution algorithm, and selects Eps parameters using a tournament selection algorithm. MOGA-DBSCAN [17] proposes the outlier-index as a new internal index method for the objective function and selects parameters based on a multi-objective genetic algorithm. Although HO methods avoid handcrafted heuristic decision parameters, they require an accurate objective function (clustering external/internal metrics) and cannot cope with the problem of unlabeled data and the error of internal metrics. While DRL-DBSCAN can not only perform DBSCAN clustering state-aware parametric search based on the objective function, but also retain the learned search experience and conduct searches without the objective function.

Reinforcement Learning Clustering. Recently, some works that intersect Reinforcement Learning (RL) and clustering algorithms have been proposed. For example, MCTS Clustering [9] in particle physics task builds high-quality hierarchical clusters through Monte Carlo tree search to reconstruct primitive elementary particles from observed final-state particles. [24] which targets the health and medical domain leverages two clustering algorithms, and RL to cluster users who exhibit similar behaviors. Both of these works are field-specific RL clustering methods. Compared with DRL-DBSCAN, the Markov process they constructed is only applicable to fixed tasks, and is not a general clustering method. Besides the above work, [4] proposes an improved K-Means clustering algorithm that selects the weights of distance metrics in different dimensions through RL. Although this method effectively improves the performance of traditional K-Means, it needs to pre-determine the number of clusters k, which has limitations.

6 CONCLUSION

In this paper, we propose an adaptive DBSCAN parameter search framework based on Deep Reinforcement Learning. In the proposed DRL-DBSCAN framework, the agents that modulate the parameter search direction by sensing the clustering environment are used to interact with the DBSCAN algorithm. A recursive search mechanism is devised to avoid the search performance decline caused by a large parameter space. The experimental results of the four working modes demonstrate that the proposed framework not only has high accuracy, stability and efficiency in searching parameters based on the objective function, but also maintains an effective performance when searching parameters without external incentives.

ACKNOWLEDGMENTS

The authors of this paper were supported by the National Key R&D Program of China through grant 2021YFB1700000, NSFC through grants U2032103 and 62028467. S&T Program of Henan through grant 202101310025. Beijing Natural Science Foundation through grant 4222001, and the Fundamental Research Funds for the Central Universities. Philip S. Yu was supported by NSF under grants IIS-1763223, IIS-1909323, IIS-2104708, and SaTC-1930941. Thanks for computing infrastructure provided by Huawei MindSpore platform.
REFERENCES

[1] 2022. scikit-opt. https://github.com/guofei9987/scikit-opt

[2] Zohreh Akbari and Rainer Unland. 2016. Automated determination of the input parameter of DBSCAN based on outlier detection. In fijl international conference on artificial intelligence applications and innovations. Springer, 286–291.

[3] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jorg Sander. 1999. OPTICS: Ordering points to identify the clustering structure. ACM Sigmod record 28, 2 (1999), 49–60.

[4] Abraham Bagherjeiran, Christoph F Eick, and Ricardo Villalta. 2005. Adaptive clustering: Better representatives with reinforcement learning. Department of Computer Science, University of Houston, Houston (2005).

[5] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balazs Kégl. 2011. Algorithms for hyper-parameter optimization. Advances in neural information processing systems 24 (2011).

[6] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. Journal of machine learning research 13, 2 (2012).

[7] James Bergstra, Daniel Yamins, and David Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In International conference on machine learning. PMLR, 115–123.

[8] Luuk Bom, Rusul Henken, and Marco Wiering. 2013. Reinforcement learning to train Ms. Pac-Man using higher-order action-relative inputs. In 2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL). IEEE, 156–163.

[9] Johann Breu, Sebastian Macaluso, Duccio Pappadopulo, and Kyle Crammer. 2020. Hierarchical clustering in particle physics through reinforcement learning. arXiv preprint arXiv:2011.08190 (2020).

[10] Avery Bryant and Krzysztof Cios. 2017. RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates. IEEE Transactions on Knowledge and Data Engineering 30, 6 (2017), 1109–1121.

[11] Hong Chang and Dit-Yan Yeung. 2008. Robust path-based spectral clustering. Pattern Recognition 41, 1 (2008), 191–203.

[12] Huang Darong, and Wang Peng. 2012. Grid-based DBSCAN algorithm with referential parameters. Physics Procedia 24 (2012), 1166–1170.

[13] Kejng Diao, Yongqiang Liang, and Jianong Fan. 2018. An improved DBSCAN algorithm using local parameters. In International CCF Conference on Artificial Intelligence. Springer, 1–12.

[14] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. arXiv preprint arXiv:1706.03762 (2017).

[15] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation in social tagging systems using Intuitionistic Fuzzy DBSCAN. In 2018 10th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC). IEEE, 361–364.

[16] Chun Guan, Yuen Kevin Kam Fung, and Yung Yue. 2012. Towards a Personalized Item Recommendation Approach in Social Tagging Systems Using Intuitionistic Fuzzy DBSCAN. In 2018 10th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC). Vol. 1. IEEE, 361–364.

[17] Anshihasi Varwans, Noam Shazeer, Niko Paramar, Jakob Uszkoreit, Liion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in neural information processing systems 30 (2017).

[18] Mengxing Huang, Qili Bao, Yu Zhang, and Wenlong Feng. 2019. A hybrid algorithm for forecasting financial time series data based on DBSCAN and SVR. Information Technology 10, 3 (2019), 103.

[19] Frank Hutter, Holger H Hoo, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In International conference on learning and intelligent optimization. Springer, 507–523.

[20] Amin Karimzadeh, Christian Schaller, and Ville Hautamäki. 2020. Action space shaping in deep reinforcement learning. In 2020 IEEE Conference on Games (ICG). IEEE, 479–486.

[21] Amin Karami and Ronnie Johansson. 2014. Choosing DBSCAN parameters automatically using differential evolution. International Journal of Computer Applications 91, 7 (2014), 1–11.

[22] Mohammad Kazemi-Beydokhti, Rahim Ali Abbaspour, and Masoud Mojarab. 2017. Spatio-Temporal Modeling of Seismic Provinces of Iran Using DBSCAN Clustering Algorithm. Pure & Applied Geophysics 174, 5 (2017).

[23] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. science 220, 4588 (1983), 671–680.

[24] Vijay R Konda and John N Tsitsiklis. 2000. Actor-critic algorithms. In Proceedings of the NIPS. 1008–1014.

[25] Urszula Kuzielewska and Krzysztof Wochowski. 2015. A modified clustering algorithm DBSCAN used in a collaborative filtering recommender system for music recommendation. In International conference on dependability and complex systems. Springer, 245–254.

[26] Wenhao Lai, Mengran Zhou, Feng Hu, Kai Bian, and Qi Song. 2019. A new DBSCAN parameters determination method based on improved MVO. IEEE Access 7 (2019), 104085–104095.

[27] Luuk Bom, Rusul Henken, and Marco Wiering. 2013. Reinforcement learning to train Ms. Pac-Man using higher-order action-relative inputs. In 2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL). IEEE, 156–163.

[28] Mengxing Huang, Qili Bao, Yu Zhang, and Wenlong Feng. 2019. A hybrid algorithm for forecasting financial time series data based on DBSCAN and SVR. Information Technology 10, 3 (2019), 103.

[29] Frank Hutter, Holger H Hoo, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In International conference on learning and intelligent optimization. Springer, 507–523.

[30] Amin Karimzadeh, Christian Schaller, and Ville Hautamäki. 2020. Action space shaping in deep reinforcement learning. In 2020 IEEE Conference on Games (ICG). IEEE, 479–486.

[31] Amin Karami and Ronnie Johansson. 2014. Choosing DBSCAN parameters automatically using differential evolution. International Journal of Computer Applications 91, 7 (2014), 1–11.

[32] Mohammad Kazemi-Beydokhti, Rahim Ali Abbaspour, and Masoud Mojarab. 2017. Spatio-Temporal Modeling of Seismic Provinces of Iran Using DBSCAN Clustering Algorithm. Pure & Applied Geophysics 174, 5 (2017).

[33] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. science 220, 4588 (1983), 671–680.

[34] Vijay R Konda and John N Tsitsiklis. 2000. Actor-critic algorithms. In Proceedings of the NIPS. 1008–1014.

[35] U rszula Kuzielewska and Krzysztof Wochowski. 2015. A modified clustering algorithm DBSCAN used in a collaborative filtering recommender system for music recommendation. In International conference on dependability and complex systems. Springer, 245–254.

[36] Wenhao Lai, Mengran Zhou, Feng Hu, Kai Bian, and Qi Song. 2019. A new DBSCAN parameters determination method based on improved MVO. IEEE Access 7 (2019), 104085–104095.

[37] Luuk Bom, Rusul Henken, and Marco Wiering. 2013. Reinforcement learning to train Ms. Pac-Man using higher-order action-relative inputs. In 2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL). IEEE, 156–163.

[38] Mengxing Huang, Qili Bao, Yu Zhang, and Wenlong Feng. 2019. A hybrid algorithm for forecasting financial time series data based on DBSCAN and SVR. Information Technology 10, 3 (2019), 103.
[53] Li Yang and Abdallah Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (2020), 295–316.

[54] Yan Yang, Bin Lian, Lian Li, Chen Chen, and Pu Li. 2014. DBSCAN clustering algorithm applied to identify suspicious financial transactions. In *2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. IEEE, 60–65.

[55] Charles T Zahn. 1971. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on computers* 100, 1 (1971), 68–86.

[56] Kun Zheng, Husheng Li, Robert C Qiu, and Shuping Gong. 2012. Multi-objective reinforcement learning based routing in cognitive radio networks: Walking in a random maze. In *2012 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 359–363.

[57] Hong Bo Zhou and Jun Tao Gao. 2014. Automatic method for determining cluster number based on silhouette coefficient. In *Advanced Materials Research*, Vol. 951. Trans Tech Publ, 227–230.

[58] X. Zhu. 2010. Stream Data Mining Repository. [http://www.cse.fau.edu/~xqzhu/stream.html](http://www.cse.fau.edu/~xqzhu/stream.html)