Automated Feature Selection: A Reinforcement Learning Perspective

Kunpeng Liu, Yanjie Fu, Le Wu, Xiaolin Li, Charu Aggarwal, *Fellow, IEEE*, and Hui Xiong, *Fellow, IEEE*

Abstract—Feature selection is a critical step in machine learning that selects the most important features for a subsequent prediction task. Effective feature selection can help to reduce dimensionality, improve prediction accuracy, and increase result comprehensibility. It is traditionally challenging to find the optimal feature subset from the feature subset space as the space could be very large. While much effort has been made on feature selection, reinforcement learning can provide a new perspective towards a more globally-optimal searching strategy. In the preliminary work, we propose a multi-agent reinforcement learning framework for the feature selection problem. Specifically, we first reformulate feature selection with a reinforcement learning framework by regarding each feature as an agent. Besides, we obtain the state of the environment in three ways, i.e., statistic description, autoencoder, and graph convolutional network (GCN), in order to derive a fixed-length state representation as the input of reinforcement learning. In addition, we study how the coordination among feature agents can be improved by a more effective reward scheme. Also, we provide a GMM-based generative rectified sampling strategy to accelerate the convergence of multi-agent reinforcement learning. Our method searches the feature subset space more globally and can be easily adapted to real-time scenarios due to the nature of reinforcement learning. In the extended version, we further accelerate the framework from two aspects. From the sampling aspect, we show the indirect acceleration by proposing a rank-based softmax sampling strategy. From the exploration aspect, we show the direct acceleration by proposing an interactive reinforcement learning (IRL)-based exploration strategy. Extensive experimental results show the significant improvement of the proposed method over conventional approaches.

Index Terms—Feature selection, multi-agent reinforcement learning, interactive reinforcement learning

1 Introduction

Feature selection aims to select the optimal subset of relevant features for a downstream predictive task [1], [2]. Effective feature selection can help to reduce dimensionality, shorten training time, enhance generalization, avoid overfitting, improve predictive accuracy, and provide better interpretation and explanation. In this paper, we study the problem of automated feature selection to improve the performance of subsequent predictive tasks.

Prior studies in feature selection can be grouped into three categories: (i) filter methods (e.g., univariate feature selection [3], [4], correlation based feature selection [2], [5]), in which features are ranked by a specific score; (ii) wrapper methods (e.g., evolutionary algorithms [6], [7], branch and bound algorithms [8], [9]), in which optimal feature subset is identified by a search strategy that collaborates with predictive tasks; (iii) embedded methods (e.g., LASSO [10], decision tree [11]), in which feature selection is part of the optimization objective of predictive tasks. However, these studies have shown not just strengths but also some limitations. For example, filter methods ignore the feature dependencies and interactions between feature selection and predictors. Wrapper methods have to search a very large feature space of 2^N feature subspace candidates, where N is the number of features. Embedded methods are subject to the strong structured assumptions of predictive models, i.e., in LASSO, the non-zero weighted features are considered to be important. As can be seen, feature selection is a complicated process that requires (i) strategic design of feature significance measurement, (ii) accelerated search of optimal feature subset, and (iii) meaningful integration of predictive models.

Reinforcement learning can interact with environments, learn from action rewards, balance exploitation and exploration, and search for long-term optimal decisions [12], [13]. These traits provide great potential to automate feature subspace exploration. Existing studies of automated feature selection in [14], [15] create a single agent to make decisions. In these models, the single agent has to determine the selection or deselection of all N features. In other words, the action space of this agent is 2^N. Such formulation is similar to the evolutionary algorithms [6], [7], [16], which tend to obtain local optima.

In this paper, we intend to propose a solution for automated feature selection using reinforcement learning.

- *Kunpeng Liu and Yanjie Fu are with the University of Central Florida, Orlando, FL 32816 USA. E-mail: {kunpengliu0827, yanjiefoo}@gmail.com.*
- *Le Wu is with the Hefei University of Technology, Hefei 230002, China. E-mail: lewu.ustc@gmail.com.*
- *Xiaolin Li is with Nanjing University, Nanjing 210023, China. E-mail: lixl@nju.edu.cn.*
- *Charu Aggarwal is with IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA. E-mail: charu@us.ibm.com.*
- *Hui Xiong is with Rutgers University, New Brunswick, NJ 08901-8554 USA. E-mail: hxiong@rutgers.edu.*

Manuscript received 12 March 2020; revised 3 June 2021; accepted 24 August 2021. Date of publication 24 September 2021; date of current version 3 February 2023.
This work was supported in part by the National Natural Science Foundation of China under Grant 61773199.
(Corresponding author: Yanjie Fu.)
Recommended for acceptance by M. Wang.
Digital Object Identifier no. 10.1109/TKDE.2021.3115477

However, several challenges arise toward this goal. First, how can we reformulate the problem so that the action space in reinforcement learning could be limited? Second, how can we accurately describe the state representation in reinforcement learning? Third, how can we efficiently accelerating the exploration of optimal features?

To address the aforementioned challenges, we propose to reformulate the feature selection problem with multi-agent reinforcement learning framework. Specifically, we first assign one agent to each feature, the actions of these feature agents are to select or deselect their corresponding features, and the state of environment is characteristics of the selected feature subspace. We then propose to integrate feature-feature redundancy and feature-label relevance with predictive accuracy as the reward scheme. In this way, we guide the cooperation and competition between agents for effective feature exploration. Moreover, we propose improved methods to derive a fixed-length representation vector from the dynamically changing selected feature subset, e.g., in dynamic graph based graph convolutional network (GCN), we construct a feature-feature similarity graph to describe the state. Since nodes are features, the number of nodes changes over time. We exploit GCN to learn state representations from dynamic graphs. Finally, we propose to accelerate the framework by improving the sampling strategy in the experience replay.

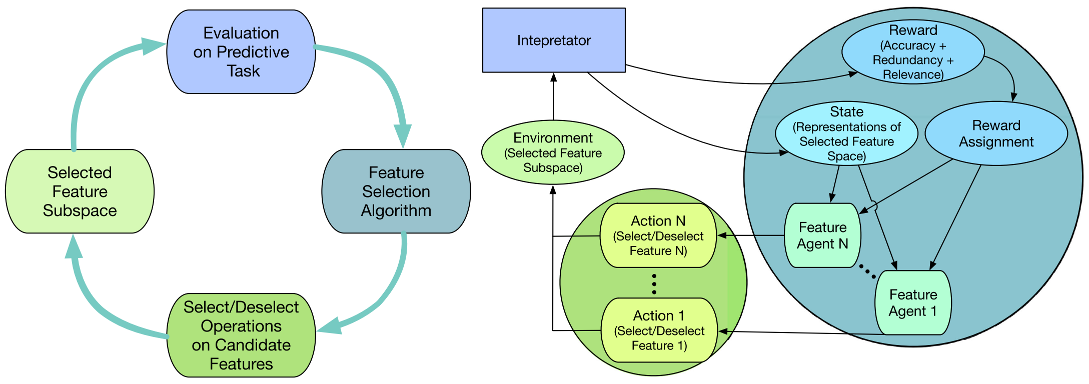
Traditionally, we use experience replay [17], [18] to train our multi-agent framework. In the experience replay, an agent takes samples from the agent's memory that stores different types of training samples to train the model. In automatic control area, reinforcement learning usually considers all of the samples in the memory, because all possible states need to be evaluated. However, in feature selection, noises, outliers, or low-reward data samples can lead to inaccurate understanding of a feature and feature-feature correlations, and, thus, jeopardize the accuracy of feature selection. Can we create a new sampling strategy to select sufficient high-quality samples and avoid low-quality samples? An intuitive method is to oversample high-quality samples by increasing their sampling probabilities. But, this method can not guarantee the independence of samples between different training steps, because the same high-quality samples repeatedly appear in different steps. To address this issue, we develop a Gaussian mixture model (GMM) based generative rectified sampling strategy. Specifically, we first train a GMM with high-quality samples. The trained GMM is then used to generate new independent samples from different mixture distribution components for reinforcement learning.

We find that there exist three limitations in the GMM-based sampling strategy: i) The Gaussian mixture model may not be the perfect model to fit sample's distribution; ii) The fitting of GMM is costly; iii) Noise may pollute samples. To address these issues, in this extended version, we develop a softmax based sampling strategy. Specifically, we first rank the samples by their reward, and then derive their priority by their inverse rank. The sampling probability is derived by the softmax of priority thereafter. In the exploration strategy aspect, reinforcement learning agent explores the environment and learns from the reward. With more and more experience accumulated, the agent can find a more and more promising exploration direction. This exploration strategy is simple and general, which can be easily adapted to almost every reinforcement learning problems. However, when the state space is extremely large, its exploration efficiency would be rather low. To reduce the exploration space, we introduce interactive reinforcement learning (IRL) [19], [20]. In IRL, a pre-trained naive feature selection plays the role of 'advisor' to guide the reinforcement learning feature selection algorithm to quickly pass its apprenticeship period. Specifically, we first derive a feature subset $\mathcal{S}^K$ via a naive feature Selection algorithm. In the apprenticeship steps, we randomly choose half of the features in $\mathcal{S}^K$ to add them in the selected feature subset. Through this addition, the state representation is changed and thus guides the reinforcement learning to a better exploration direction. After the apprenticeship period, the multi-agent reinforcement learning leaves $\mathcal{S}^K$ and does feature selection independently.

In summary, in this paper, we develop an enhanced multi-agent reinforcement learning framework for feature subspace exploration. Specifically, our contributions are as follows: (1) We reformulate feature selection problem with a multi-agent reinforcement learning framework and design a new reward scheme to guide the cooperation and competition between agents. (2) We develop three different methods: meta descriptive statistics, autoencoder based deep representation, and dynamic graph based graph convolutional network (GCN), to derive accurate state representation. (3) We develop three different strategies: GMM-based generative rectified sampling strategy, rank-based softmax sampling strategy and IRL-based exploration strategy to improve the training and exploration. (4) We conduct extensive experiments to demonstrate the enhanced performances of our methods.

2 Related Work

Feature Selection. Feature selection can be categorized into three types, based on how the feature selection algorithm combines with the machine learning tasks, i.e., filter methods, wrapper methods and embedded methods [1], [21]. Filter methods rank the features merely by relevance scores and only top-ranking features are selected. The representative filter methods are univariate feature selection [3], [4] and correlation based feature selection [2], [5]. With very simply computation complexity, filter methods are very fast and thus they're efficient on high-dimensional datasets. However, they ignore the feature dependencies, as well as interactions between feature selection and the subsequent predictors. Unlike filter methods, wrapper methods take advantage of the predictors and consider the prediction performance as the objective function [22]. The representative wrapper methods are branch and bound algorithms [8], [9]. Wrapper methods are supposed to achieve better performance than filter methods since they search on the whole feature subset space. However, the feature subset space exponentially increases with the number of features, making traversing the feature subset space a NP-hard problem. Evolutionary algorithms [6], [7], [16] low down the computational cost but could only promise locally optimal results. Embedded methods combine feature selection with


Evaluation on Predictive Task
Feature Selection Algorithm
Select/Deselect Operations on Candidate Features
Selected Feature Subspace
Intepretator
Reward (Accuracy + Redundancy + Relevance)
State (Representations of Selected Feature Space)
Reward Assignment
Environment (Selected Feature Subspace)
Action N (Select/Deselect Feature N)
Action 1 (Select/Deselect Feature 1)
Feature Agent N
Feature Agent 1

(a) General Process of Feature Selection (b) Multi-Agent Reinforcement Learning on Feature Selection

Fig. 1. From traditional feature selection to multi-agent reinforcement learning based feature subspace exploration. In the reinforcement learning based feature selection, the interpreter represents the selected feature subset into a state vector by representation learning methods and obtains the overall reward by evaluate the feature subset via downstream machine learning tasks.

predictors more closely than wrapper methods, and actually they incorporate feature selection as part of predictors. The most widely used embedded methods are LASSO [10], decision tree [11] and SVM-RFE [23]. Embedded methods could have supreme performance on the incorporated predictors, but normally not very compatible with other predictors. Recently, an emerging technology called reinforced feature selection has demonstrated significant improvement by tackling the feature selection problem with reinforcement learning technology.

Multi-Agent Reinforcement Learning. Our work is related to multi-agent reinforcement learning, where multiple agents share a complex environment and interact with each other [24]. In the single-agent formulation, the reinforcement learning agent takes an action to change the environment, and get a reward as feedback to evaluate its action, so as to improve its next decision on action [25]. In the multi-agent formulation, agents not only need to interact with the environment, but also need to interact with each other. *Stankovic et al.* proposed new algorithms for multi-agent distributed iterative value function approximation where the agents are allowed to have different behavior policies while evaluating the response to a single target policy [26]. *Liao et al.* proposed Multi-objective Optimization by Reinforcement Learning (MORL) to solve the optimal power system dispatch and voltage stability problem, which is undertaken on individual dimension in a high-dimensional space via a path selected by an estimated path value which represents the potential of finding a better solution [27]. *Yang et al.* developed deep reinforcement learning algorithms which could handle large scale agents with effective communication protocol [28], [29]. *Lin et al.* proposed to tackle the large-scale fleet management problem using reinforcement learning, and proposed a contextual multi-agent reinforcement learning framework which successfully tackled the taxi fleet management problem [13]. However, these methods define their states by handcraft rules instead of by representation learning, which may leave out important information provided by the environment. And also, as we know the training speed of multi-agent reinforcement learning is low due to the large action space, but these methods rarely study how to improve the training efficiency. Existing studies [14], [15] create a single agent to make decisions. However, this agent has to determine the selection or deselection of all N features. In other words, the action space of this agent is 2^N. Such formulation is similar to the evolutionary algorithms [6], [7], [16], which tend to obtain local optima.

3 Problem Formulation

We study the problem of feature subspace exploration, which is formulated as a multi-agent reinforcement learning task. Fig. 1 shows an overview of our proposed multi-agent reinforcement learning based feature exploration framework. Given a set of features to be explored, we first create a feature agent for each feature. This feature agent is to decide whether its associated feature is selected or not. The selected feature subset is regarded as the environment, in which feature agents interact with each other. The correlations between features are schemed by reward assignment. Specifically, the components in our multi-agent reinforcement learning framework includes agents, state, environment, reward, reward assignment strategy, and agent actions.

Agent. Assuming there are N features, we define N agents for the N features. For one agent, it is designed to make the selection decision for the corresponding feature.

Actions. For the ith feature agent, the feature action $a_i = 1$ indicates the ith feature is selected, and $a_i = 0$ indicates the ith feature is deselected.

Environment. In our design, the environment is the feature subspace, representing a selected feature subset. Whenever a feature agent issue an action to select or deselect a feature, the state of feature subspace (environment) changes.

State. The state s is to describe the selected feature subset. To extract the representation of s, we explore three different strategies, i.e., meta descriptive statistics, autoencoder based deep representation and dynamic graph based graph convolutional network (GCN). We will elaborate these three state representation techniques in Section 4.3.

Reward. We design a measurement to quantify the overall reward R generated by the selected feature subset, which is defined the weighted sum of (i) predictive accuracy of the selected feature subset Acc, (ii) redundancy of the selected feature subset Rv, and (iii) relevance of the selected feature subset Rd.

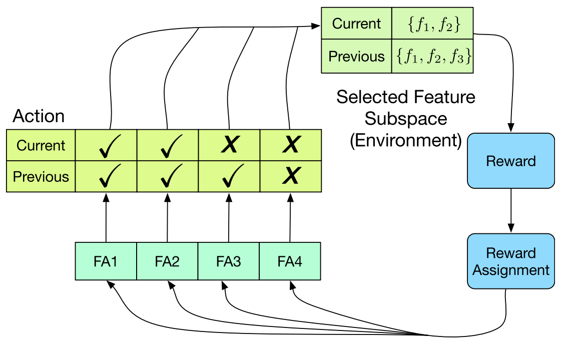


Fig. 2. The demonstration of reward assignment process. The feature agent 1 and 2 issue an action to select the feature 1 and feature 2. The feature agent 3 issues an action to deselect the feature 3.

Reward Assignment Strategy. We develop a strategy to allocate the overall reward to each feature agent. The assignment of the overall reward to each agent, indeed, shows the coordination and competition relationship among agents. In principle, we should recognize and reward all of the participated feature agents. Fig. 2 shows an example of reward assignment. There are four features with four corresponding feature agents. In the previous iteration, the feature 1, 2, 3 are selected, and the feature 4 is not selected. In the current iteration, feature agent 1 and feature agent 2 issue actions to select feature 1 and feature 2; feature agent 3 issues an action to deselect feature 3; feature agent 4 does not participate and issue any action to change the status of feature 4. In summary, there are only three feature agents (FA1, FA2, FA3) that participate and issue actions. Therefore, the current reward R is equally shared by these three agents.

4 Multi-Agent Reinforcement Learning Feature Selection

In this section, we propose a multi-agent reinforcement learning framework for automated feature subspace exploration. Later, we discuss how to measure the reward, how to improve the state representation and how to accelerate the proposed framework.

4.1 Framework Overview

Fig. 3 shows our proposed framework consists of many feature subspace exploration steps. Each exploration step includes two stages, i.e., control stage and training stage.

In the control stage, each feature agent takes actions based on their policy networks, which take current state as input and output recommended actions and next state. The select/deselect actions of each feature agent will change the size and contents of the selected feature subset, and thus, lead to a new selected feature subspace. We regard the selected feature subset as environment. The state represents the statistical characteristics of the selected feature subspace. We derive a comprehensive representations of the state through three different methods, i.e., descriptive statistics, autoencoder and GCN (refer to Section 4.3). Meanwhile, the actions taken by feature agents generate an overall reward. This reward will then be assigned to each of the participating agents.

In the training stage, agents train their policy via experience replay independently. For agent i, at time t, a newly-created tuple $\{s_i^t, a_i^t, r_i^t, s_i^{t+1}\}$, including the state ($s_i^t$), the action ($a_i^t$), the reward ($r_i^t$) and the next state ($s_i^{t+1}$), is stored into each agent's memory. The agent i uses its corresponding mini-batch samples to train its Deep Q-Network (DQN), in order to obtain the maximum long-term reward based on the Bellman Equation [25]

$$Q(s_i^t, a_i^t|\theta_t) = r_i^t + \gamma \max Q(s_i^{t+1}, a_i^{t+1}|\theta_{t+1}), \tag{1}$$

where θ is the parameter set of Q network, and γ is the discount factor.

The exploration of feature subspace continues until convergence or meeting several predefined criteria.

4.2 Measuring Reward

We propose to combine the predictive accuracy Acc, the feature subspace relevance Rv, and the feature subspace redundancy Rd as the reward R of actions.

Predictive Accuracy. Our goal is to explore and identify a satisfactory feature subset, which will be used to train a predictive model in a downstream task, such as classification and outlier detection. We propose to use the accuracy Acc

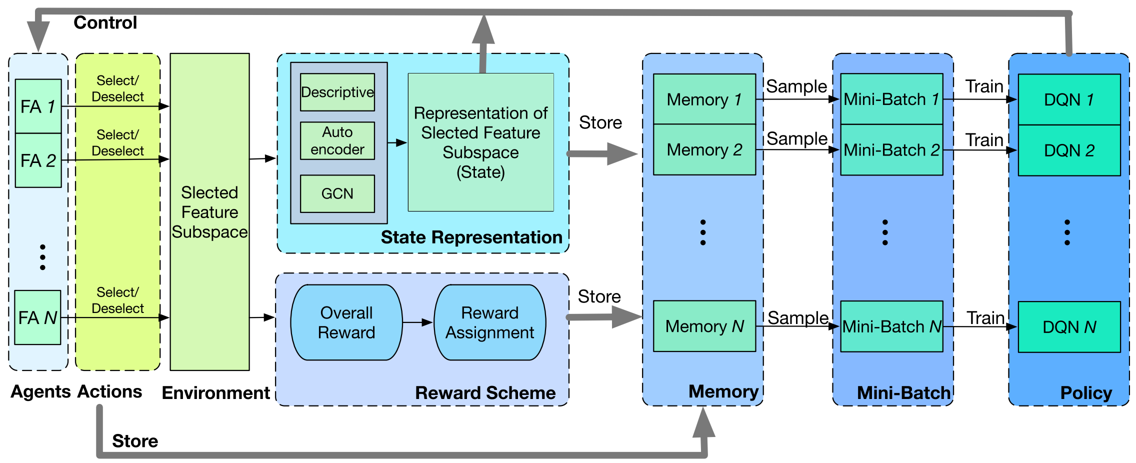


Fig. 3. Framework. The framework consists of two stages. In the control stage, feature agents select or drop their corresponding features based on policies. In the training stage, the policies are trained via samples from memories.

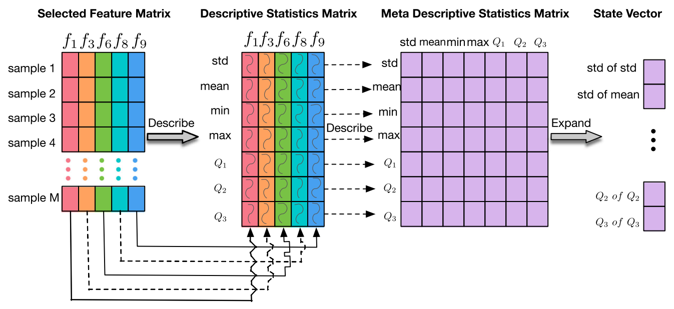


Fig. 4. Meta descriptive statistics. We extract descriptive statistics twice from the feature subspace to obtain a fixed-length state vector.

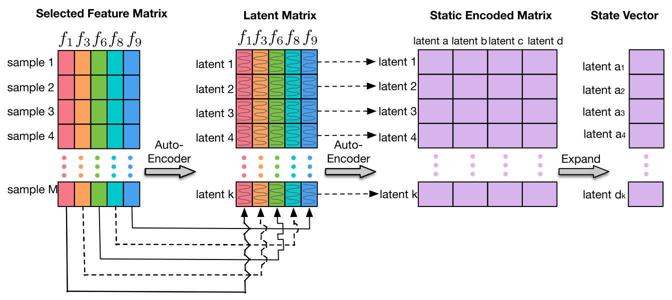


Fig. 5. Autoencoder based deep representations. We use two autoencoders to map the feature subspace into a fixed-length state vector.

of the predictive model to quantify the reward. Specifically, if the predictive accuracy is high, the actions that produce the selected feature subset should receive a high reward; if the predictive accuracy is low, the actions that produce the selected feature subset should receive low rewards.

Feature Subspace Characteristics. Aside from exploiting the predictive accuracy as reward, we propose to take into account the characteristics of the selected feature subset. Specifically, a qualified feature subset is usually of low information redundancy and of high information relevance to the predictive labels (responses). Both the information relevance and redundancy can be quantified by the mutual information, denoted by I. Formally, I by:

$$I(\boldsymbol{x};\boldsymbol{y}) = \sum_{i,j} p(x_i, y_j) log\left(\frac{p(x_i, y_j)}{p(x_i)p(y_j)}\right), \tag{2}$$

where x_i, y_i is the ith and jth feature, $p(\boldsymbol{x},\boldsymbol{y})$ is the joint distribution of $\boldsymbol{x}$ and $\boldsymbol{y}$, while $p(\boldsymbol{x})$ and $p(\boldsymbol{y})$ are marginal distribution of $\boldsymbol{x}$ and $\boldsymbol{y}$.

- The *information redundancy* of a feature subset, denoted by Rd, can be quantified by the sum of pairwise mutual information among features. Formally, Rd is given by:

$$Rd = \frac{1}{|S|^2} \sum_{\boldsymbol{x_i},\boldsymbol{x_j} \in S} I(\boldsymbol{x_i};\boldsymbol{x_j}), \tag{3}$$

  where S is the feature subset, $\boldsymbol{x_i}$ is the ith feature,
- The *information relevance* of a feature subset, denoted by Rv, can be quantified by the mutual information between features and labels. Formally, Rv is given by:

$$Rv = \frac{1}{|S|} \sum_{\boldsymbol{x_i} \in S} I(\boldsymbol{x_i};\boldsymbol{c}), \tag{4}$$

  where $\boldsymbol{c}$ is the label vector.

4.3 Improving State Representation

Assuming there is a $M * N$ dataset D, which includes M data samples and N features. Let n_j be the number of selected features at the jth exploration step. Then, $M * n_j$ is the dimension of the selected data matrix S, which varies over exploration steps. However, the policy network and target network in DQN require the state representation vector s to be a fixed-length vector all the time. We thus, need to derive a fixed-length state vector s from the selected data matrix S, whose dimensions change over time.

To derive accurate state representation with fixed length, we develop three different methods, including (i) meta descriptive statistics of feature subspace; (ii) static subspace graphs based autoencoder; (iii) dynamic feature-feature similarity graphs based graph convolutional network (GCN). The commonness between these three methods is that they all first learn representations for each feature, and then aggregate them to get a state representation. The differences between them lie on the representation learning algorithms and aggregation strategies.

Method 1: Meta Descriptive Statistics of Feature Subspace. Fig. 4 shows how we extract the meta data of descriptive statistics from the selected data matrix through a two-step procedure.

Step 1. We extract descriptive statistics of the selected data matrix S, including the standard deviation, minimum, maximum and Q1 (the first quartile), Q2 (the second quartile), and Q3 (the third quartile). Specifically, we extract the seven descriptive statistics of each feature (*column*) in S, and thus, obtain a *descriptive statistics matrix* $\boldsymbol{D}$ with size of $7 * n_j$.

Step 2: We extract the seven descriptive statistics of each *row* in the descriptive statistics matrix $\boldsymbol{D}$, and obtain a *meta descriptive statistics matrix* $\boldsymbol{D'}$ with a size of $7 * 7$.

Finally, we link each column $\boldsymbol{D'}$ together into the state vector s with a fixed length of 49.

Method 2: Autoencoder Based Deep Representation of Feature Subspace. Autoencoder has been widely used for representation learning by minimizing the reconstruction loss between an original input and a reconstructed output [30]. An autoencoder contains an encoder that maps the input into a latent representation, and an decoder that reconstructs the original input based on the latent representation.

Fig. 5 shows how we extract the state vector from the selected data matrix through a two-step algorithm.

Step 1: Assuming at the jth exploration step, S is the selected data matrix, and n_j is the number of selected features. For each feature (*column*) in S, we apply an autoencoder to convert each feature column into a k-length latent vector, and thus, obtain a *latent matrix* $\boldsymbol{L}$ with a dimension of $k * n_j$. However, $\boldsymbol{L}$ cannot represent the state, because the size of $\boldsymbol{L}$ is not static and still varies over number of selected features n_j at the jth exploration step.

Step 2: We apply another auto-encoder to map each *row* of $\boldsymbol{L}$ into a o-length latent vector, and obtain a *static encoded matrix* $\boldsymbol{L'}$ with a fixed dimension of $k * o$.

Finally, we link each column in $\boldsymbol{L'}$ together into the state vector s with a fixed length of $k * o$.

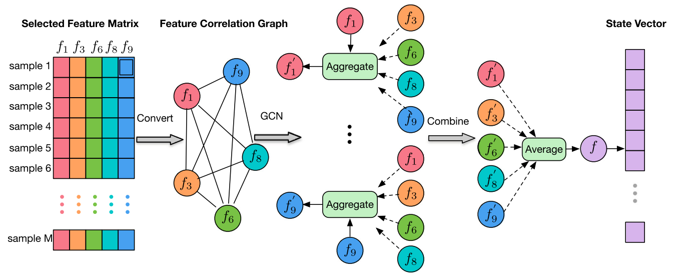


Fig. 6. Dynamic-graph based GCN. We denote the feature subspace by a dynamic graph and use GCN to update representations of each node.

Method 3: Dynamic-Graph Based Graph Convolutional Network (GCN). Method 1 and method 2 extract explicit and latent representations of each feature. In this method, we consider not just a feature's individual representations, but also the correlations among features. Fig. 6 shows how the GCN works. To better capture the relationship among features, we first convert the selected data matrix S into a dynamic complete graph $\mathbf{G}$, where a node is a feature column in S. With the *feature correlation graph* $\mathbf{G}$, any graph node embedding techniques could be used for node latent representation by exploiting the correlation among features. As the focus of this paper is not to design more sophisticated node embedding models, we choose GCN as it is a state-of-the-art graph embedding models and shows competing effectiveness in many graph based tasks.

Let S be the selected data matrix with a dimension of $M * N$, Z be the representation matrix of nodes (features) with a dimension of $k * N$, k is the length of updated representation. The neural network layer in GCN is given by:

$$H^{(l+1)} = f(H^{(l)}, A), \tag{5}$$

where $H^{(0)} = S$, $H^{(L)} = Z$, L is the layer number, A is the adjacency matrix of graph G. The regular GCN can be reduced into a simplified version by considering the node's own representation (rather than merely the neighbor structures) and performing symmetric normalization [31]

$$f(H^{(l)}, A) = \sigma(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}), \tag{6}$$

where $\hat{A} = A + I$ with I being an identity matrix, $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$.

By solving GCN, we obtain the latent representations Z of each feature. We average the representation of each feature into the k-length state representation vector.

4.4 Improving Experience Replay via Generative Rectified Sampling

Experience replay is widely used to improve training efficiency of neural networks in reinforcement learning [17], [18]. After taking each action, the latest sample, in the form of a tuple that consists of the action (a), the reward (r), the state (s) and the next state (s'), are stored into the memory to replace the oldest sample. In training step, a mini-batch of samples are picked to update the policy network. The vanilla experience replay treats the samples equal, thus it uniformly samples the data from its memory.

In the task of feature subspace exploration, we are particularly interested in exploiting high-quality samples to accelerate the exploration speed. Prior studies tackle this problem by increasing the sampling probabilities of high-quality samples [32], [33]. However, such strategy creates a new problem: the sampler repeatedly selects a limited number of high-quality samples. Consequently, prior studies can not guarantee the independence of selected samples between different training steps, and can not cover a comprehensive space in the unknown high-quality sample population.

To deal with this problem, we propose a Gaussian mixture model (GMM) based generative rectified sampling algorithm. For each agent, as Algorithm 1 shows, we take a set of memory samples $T = \{< a, r, s, s' >\}$ as inputs. We first cluster the memory samples into two groups: T_0 and T_1. Samples with the selected action ($a = 0$) are assigned to group T_0, while samples with the deselected action ($a = 1$) are assigned to group T_1. Later, we rank the memory samples in T in terms of reward (r) and select the top p proportion of high-reward samples in each group as high-quality samples. The selected high-quality samples are then used to train two GMM based generative models for their corresponding groups via an Expectation Maximization (EM) algorithm [34]. After that, for each group, we use its corresponding well-trained GMM model to generate simulated samples to replace the $1 - p$ proportion of low-reward samples in the corresponding group. In this way, we create two high-reward, large-size, yet independent memory sample sets for the select-action and deselect-action groups. We combine the two simulated memory sample sets into a new high-quality dataset. The agent will take a mini-batch of samples from the new high-quality dataset for accelerating training.

Algorithm 1. The GMM-Based Generative Rectified Sampling Algorithm

```
**Input:** Memory dataset $T$.
**Output:** A mini-batch of samples $B$.
1: $p \leftarrow$ high-quality sample proportion of $T$.
2: Stratify $T$ into two groups. Samples with $a = 0$ are assigned
   to group $T_0$ and samples with $a = 1$ are assigned to group
   $T_1$.
3: **for** $i = 0$ *to* $1$ **do**
4:     $N_i \leftarrow$ sample number of $T_i$.
5:     $K_i \leftarrow$ component number of GMM model $\mathcal{G}^i$.
6:     Rank samples in $T_i$ by their reward $r$, then select top $N_i *$
       $p$ samples from $T_i$ to form the high-quality dataset $H_i$.
7:     Use $H_i$ to train the GMM $\mathcal{G}^i = \sum_1^{K_i} \phi_i \mathcal{N}(\mu_i, \Sigma_i)$ via EM
       algorithm.
8:     Generate $N_i * (1 - p)$ samples from $G^i$ to form the gener-
       ated dataset $G_i$.
9:     Join $H_i$ and $G_i$ to create high-quality dataset of action $i$,
       $T_i'$.
10: **end**
11: Join $T_0'$ and $T_1'$ to get high-quality dataset $T'$.
12: Sample a mini-batch of samples $B$ from $T'$.
```

5 Extension: Accelerating Feature Subspace Exploration

In the previous version, we initially explores how to accelerate the feature subspace exploration process via improving sampling strategy, which accelerates the exploration indirectly by improving the training of exploration strategy. (Section 4.4). In the extended version, we further dive into improving the sampling strategy, and also, we study how to accelerate the exploration process via improving exploration strategy directly.

5.1 Accelerating Feature Subspace Exploration via Rank-Based Softmax Sampling

The GMM-based generative rectified sampling strategy can make full use of the samples in experience replay, thus accelerate the training process of reinforcement learning policy. However, it naturally has three potential drawbacks: 1) It is based on an assumption that the sample are generated by a Gaussian Mixture Model, while their actual distribution could be different; 2) The fitting of GMM model is computationally expensive. To make things worse, it needs to fit the GMM model every time it samples; 3) There could exist noise in the samples, which affects the fitting accuracy of GMM model. Here we have one question: can we propose a more simple sampling strategy yet effective than the sampling strategy in Section 4.4, which could have lower computational burden and higher robustness?

To tackle these problems, we introduce a rank-based softmax sampling strategy. In this sampling strategy, we measure the importance of each samples by their ranks in the experience replay memory. The sampling probability for each sample is then designed based on their ranks

$$P(i) = \frac{exp(p_i)}{\sum_{n=1}^{N_E} exp(p_n)}, \tag{7}$$

where N_E is the size of experience replay memory, p_i is the priority of ith sample, and we make $p_i = \frac{1}{rank(i)}$, where $rank(i)$ is the rank of sample i based on its reward. The softmax operation promises the sum of all probabilities equals 1. Since $rank(i)$ is a relative value, it has high tolerance of noise and could be very robust. There is no assumption of GMM distribution, thus there is no need of fitting, making the computational burden very low. Specifically, as Algorithm 2 shows, we first derive the rank for each sample, and then obtain their sampling probabilities by Equation (7). The agent will take a mini-batch of samples based on the rank-based sampling probabilities. Compared with the GMM-based generative rectified sampling strategy, it is efficient due to the low computational burden, and effective due to the robustness on noise.

5.2 Accelerating Feature Subspace Exploration via Interactive Reinforcement Learning

In the classic reinforcement learning framework, the agent repeatedly explores the state space and gets reward, after which it gets more and more experience and behaves better and better. This exploration strategy is general and universal, meaning that it can be applied to any formulated reinforcement learning problems. However, in our case, the state space is extremely large, and if we simply adapt the conventional exploration strategy, the exploration efficiency would be rather low. Here we have one question: Can we propose a more advanced exploration strategy, which could explore along a more promising direction, so that the feature space exploration process would be accelerated?

Algorithm 2. The Softmax Sampling Algorithm

```
Input: Memory dataset $T$.
Output: A mini-batch of samples $B$.
1: $N_E \leftarrow$ size of experience replay memory $T$.
2: Rank samples in $T$ by their reward, and let $p_i = \frac{1}{rank(i)}$ be
   their priority.
3: Derive the sampling probability for each data sample by
   Equation (7).
4: Sample a mini-batch of samples $B$ from $T$.
```

The proposed multi-agent reinforcement learning framework improves itself step by step, and it has a apprenticeship period in the beginning when its performance is very bad. To reduce its exploration space, we introduce interactive reinforcement learning (IRL) [19], [20]. In IRL, a naive feature selection method, i.e., K-Best Selection [3], works as the 'advisor' to guide reinforcement learning to explore along a relatively good direction. After pre-defined steps, the reinforcement learning abandons the advisor and explore the state space independently.

Specifically, as Algorithm 3 shows, we first derive a feature subset $\mathcal{S}^K$ via K-Best Selection. In the apprenticeship steps, we randomly choose half of the features in $\mathcal{S}^K$ to add them in the selected feature subset. Through this addition, the state representation is changed and thus guides the reinforcement learning to a better exploration direction. The reason we don't use all of the features in $\mathcal{S}^K$ every step is to avoid over-fitting, and to keep the feature selection process different from the K-Best Selection. After the apprenticeship period, the multi-agent reinforcement learning would do feature selection independently.

Algorithm 3. The Interactive Reinforcement Learning Enhanced Exploration Strategy

```
Input: Feature number $K$, apprenticeship step $N_A$, overall
       step $N_O$, feature set $\mathcal{S}$.
Output: Optimal feature subset $\mathcal{S}'$.
1: Derive a feature subset $\mathcal{S}^K$ via K-Best Selection.
2: Randomly initialize selected feature subset $\mathcal{S}'_0$.
3: for $i = 1$ to $N_A$ do
4:     Derive a selected feature subset $\mathcal{S}'_i$ by multi-agent rein-
       forcement learning feature selection proposed in Sec-
       tion 4. (Note: This step relies on the selected feature subset
       $\mathcal{S}'_{i-1}$ from the last step, as $\mathcal{S}'_{i-1}$ decides the state representation.)
5:     Randomly choose $K/2$ features from $\mathcal{S}^K$, denoted as
       $\mathcal{S}_i^{K/2}$.
6:     Let $\mathcal{S}'_i = \mathcal{S}'_i + \mathcal{S}_i^{K/2}$.
7: end
8: for $i = N_A + 1$ to $N_O$ do
9:     Derive a selected feature subset $\mathcal{S}'_i$ by multi-agent rein-
       forcement learning feature selection proposed in Section 4.
10: end
```

6 Experimental Results

We evaluate the proposed methods in feature selection with real-world datasets. We also design more experiments to study the newly proposed softmax sampling strategy and IRL-based exploration strategy.

6.1 Data Description

The experiments are conducted on two publicly available datasets as follows:

Dataset 1. This dataset is a cartographic dataset from Kaggle.[1] There are 15120 samples with 54 features that describe the characteristics of different wilderness areas. The class labels are categorical values that range from 1 to 7, and represent seven forest cover types (the predominant kind of tree cover) in the areas. Among the 54 features, 10 of them are continuous, and the remaining 44 are categorical. The downstream task is to predict the cover type of wilderness area.

Dataset 2. This dataset is a place localization dataset from UCI[2] [35]. There are 34465 samples with 119 features that describe the characteristics of different places. The class label are categorical values that range from 1 to 2, and represent two geographical spots. Among the 119 features, 89 of them are continuous, and the remaining 30 are categorical. The downstream task is to predict the geographical spot with spot descriptions.

6.2 Evaluation Metrics

To show the effectiveness of the proposed method, we use the following metrics for evaluation.

Overall Accuracy is the ratio of number of correct predictions to number of all predictions. Formally, the overall accuracy is given by $\frac{TP+TN}{TP+TN+FP+FN}$, where TP, TN, FP, FN are true positive, true negative, false positive and false negative for all classes. We use this metric to measure the accuracy of a classifier on test dataset. The latter three metrics measure classification performance of each label from different aspects.

Precision is given by $\frac{TP_k}{TP_k+FP_k}$ which represents the ratio of true positive to true positive plus false positive with respect to the kth ($k \in [1,7]$ for Dataset 1 and $k \in [1,2]$ for Dataset 2) label.

Recall is given by $\frac{TP_k}{TP_k+FN_k}$ which represents the ratio of true positive to true positive plus false negative with respect to the kth ($k \in [1,7]$ for Dataset 1 and $k \in [1,2]$ for Dataset 2) label.

F-measure considers both precision and recall in a single metric by taking their harmonic mean. Formally, F-measure is given by $2 * P * R/(P+R)$, where P and R are precision and recall respectively.

6.3 Baseline Algorithms

We compare performance of our proposed Multi-Agent Reinforcement Learning Feature Selection (MARLFS) against the following six baseline algorithms, where K-Best Selection and mRMR belong to filter methods; LASSO and Recursive Feature Elimination (RFE) belong to embedded methods; Genetic Feature Selection (GFS) and Single-Agent Reinforcement Learning Feature Selection (SARLFS) belong to wrapper methods.

(1) *K-Best Selection.* The K-Best Selection [3] first ranks features by their χ^2 scores with the target vector (label vector), and then selects the K highest scoring features. In the experiments, we make K equal to the number of selected features in MARLFS.

(2) *mRMR.* The mRMR [36] first ranks features by minimizing feature's redundancy, while maximizing their relevance with the target vector (label vector), and then selects the K highest ranking features. In the experiments, we make K equal to the number of selected features in MARLFS.

(3) *LASSO.* LASSO [10] conducts feature selection and feature space shrinkage via l_1 penalty, which drops the feature variables whose coefficients are 0. The hyper parameter in LASSO is its regularization weight λ, which is set to 1.0 in the experiments.

(4) *Recursive Feature Elimination (RFE).* RFE [37] selects features by recursively selecting smaller and smaller feature subsets. First, the predictor is trained by all features and the importance of each feature are scored by the predictor. After that, the least important features are deselected. This procedure process recursively until the desired number of features are selected. In the experiments, we set the selected feature number half of the feature space.

(5) *Genetic Feature Selection (GFS).* Genetic Feature Selection [38] selects features by first calculating the fitness level for each feature and then generates better feature subsets via crossover and mutation. In the experiments, we set crossover probability to 0.5, mutation probability to 0.2, crossover independent probability to 0.5 and mutation independent probability to 0.05.

(6) *Single-Agent Reinforcement Learning Feature Selection (SARLFS).* In SARLFS [15], the agent learns a KWIK (Knows What It Knows) model, which is represented by a dynamic Bayesian network, deduces a minimal feature set from this network, and computes a policy on this feature subset using dynamic programming methods. In the experiments, the two accuracy thresholds in the KWIK are set to $\epsilon = 0.15$, $\delta = 0.10$.

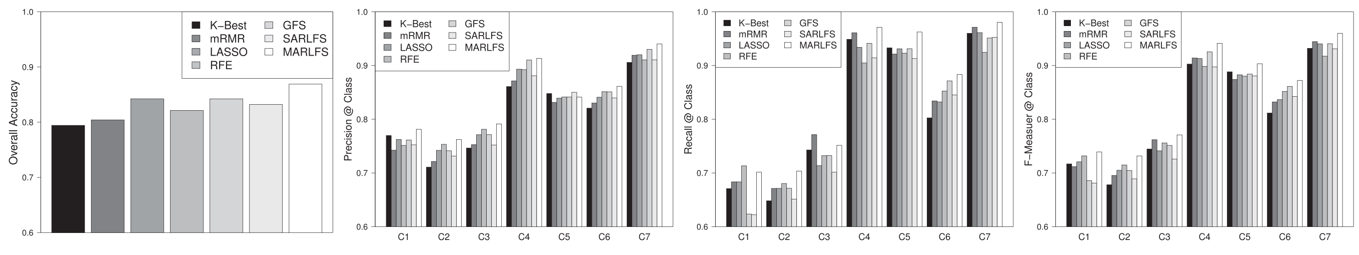
6.4 Overall Performance

We compare our method MARLFS with baseline methods in terms of overall accuracy as well as precision, recall and F-measure of the seven classes on the real-world data. In the experiments, for all deep networks, we set mini-batch size to 32 and use AdamOptimizer with a learning rate of 0.01. For all experience replays, we set memory size to 200. We set the Q network in our methods as a two-layer ReLU with 64 and 8 nodes in the first and second layer. The high-quality proportion in GMM sampling is 0.20 for Dataset 1 and 0.30 for Dataset 2. Unless specified, we use GCN method as the representation learning algorithm in the experiments, whose network is a two-layer ReLU with 128 and 32 nodes in the first and second layer. The predictor we use is a random forest with 100 decision trees. Figs. 7 and 8 show that our method exceeds all of the baseline methods in the task of exploring a qualified feature subset.

6.5 Robustness Check

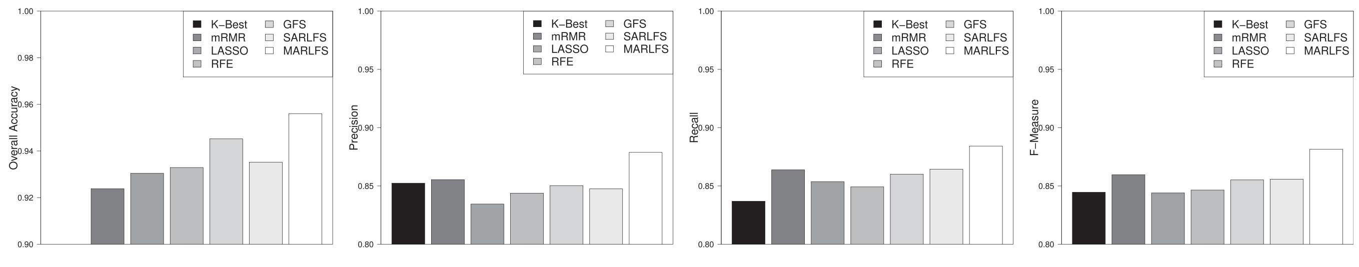
The predictive accuracy relies on not just feature selection, but also predictors. We apply our method to different

1. https://www.kaggle.com/c/forest-cover-type-prediction/data
2. https://archive.ics.uci.edu/ml/datasets/Nomao



(a) Overall Accuracy (b) Precision (c) Recall (d) F-Measure

Fig. 7. Performance comparison of different feature selection algorithms on Dataset 1.



(a) Overall Accuracy (b) Precision (c) Recall (d) F-Measure

Fig. 8. Performance comparison of different feature selection algorithms on Dataset 2.

predictors in order to investigate whether our explored feature subset are consistently stable and can consistently outperform other baseline methods on various predictors. In this way, we can examine the robustness of our methods. Aside from the random forest (RF) predictor, we use (i) LASSO; (ii) Decision Tree (DT); (iii) SVM with a rbf kernel, and (iv) XGBoost as predictors for this experiment. Tables 1 and 2 show that our MARLFS outperforms the baselines methods over almost all of the predictors. However, when we use LASSO to perform both feature selection and target prediction, the accuracy of our method is slightly lower than LASSO. This might be explained by the reason that both feature section and prediction optimization are integrated and unified in a single model framework. However, when we use LASSO to perform feature selection, and use other classification models for prediction, our method outperform such type of baselines.

6.6 Study of Reward Function

We study the design of the reward function in our framework. Specifically, we consider four cases: (i) *Acc* that only considers accuracy in the reward function; (ii) *Rv* that only considers relevance in the reward function; (iii) *Rd* that only considers redundancy in the reward function; (iv) *Acc+Rv +Rd* that considers accuracy, relevance and redundancy in the reward function.

Figs. 9 and 10 show that Acc is the second best reward function, since it leads the exploration to the direction of improving accuracy. Rv and Rd are less satisfactory. This is because both are unsupervised indicators of rewards and are not directly relevant to prediction accuracy. Acc+Rv+Rd achieve the best performances since it considers both supervised indicator and unsupervised indicator into account. Specifically, Fig. 9a shows the comparisons of overall accuracy over exploration steps. Figs. 9b, 9c and 9d show the comparisons of precision, recall and F-measure over different classes with 3000 exploration steps.

6.7 Study of State Representation Learning

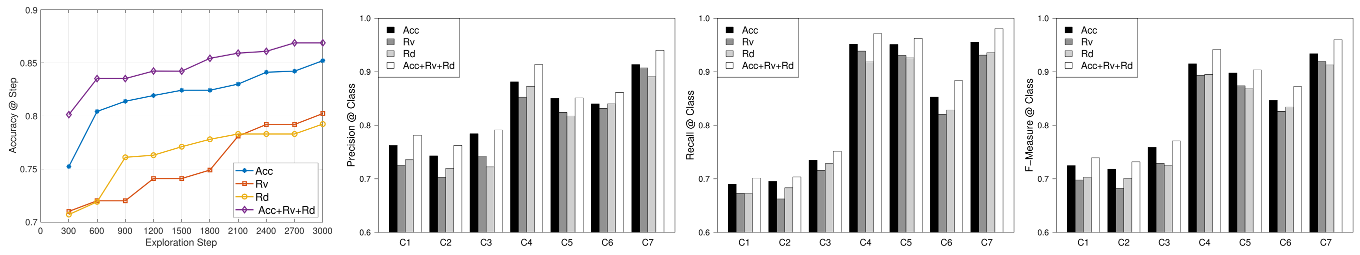
We compare the performances different representation learning methods. We consider five cases, i.e., (i) *MDS*: meta descriptive statistics, which uses the meta data of descriptive statistics of feature subspace to represent the state; (ii) *AE*: auto-encoder based deep representation, which uses deep auto-encoder to encode feature subspace twice to obtain state representation; (iii) *GCN*: uses

TABLE 1
Overall Accuracy of Feature Selection Algorithms on Dataset 1

Dataset 1		Predictors				
		RF	LASSO	DT	SVM	XGBoost
Algorithms	K-Best	0.7943	0.8246	0.8125	0.8324	0.8076
	mRMR	0.8042	0.8124	0.8096	0.8175	0.8239
	LASSO	0.8426	**0.8513**	0.8241	0.8131	0.8434
	RFE	0.8213	0.8236	0.8453	0.8257	0.8348
	GFS	0.8423	0.8318	0.8350	0.8346	0.8302
	SARLFS	0.8321	0.8295	0.8401	0.8427	0.8450
	MARLFS	**0.8690**	0.8424	**0.8583**	**0.8542**	**0.8731**

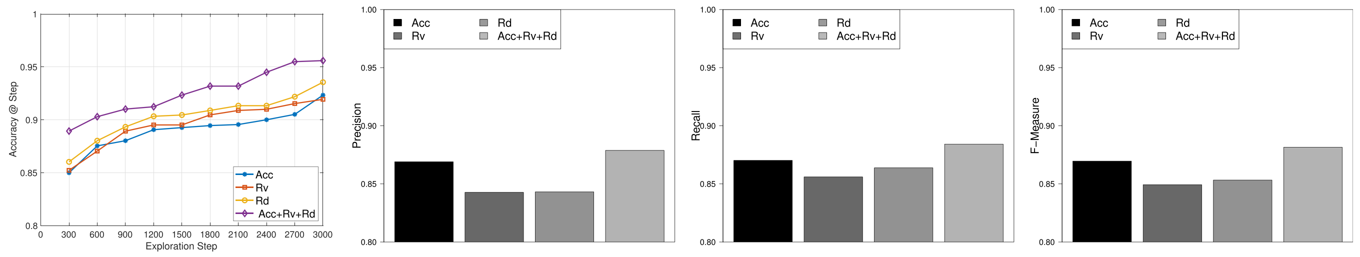
TABLE 2
Overall Accuracy of Feature Selection Algorithms on Dataset 2

Dataset 2		Predictors				
		RF	LASSO	DT	SVM	XGBoost
Algorithms	K-Best	0.9032	0.9213	0.9093	0.9180	0.9248
	mRMR	0.9238	0.9102	0.9199	0.9127	0.9041
	LASSO	0.9304	**0.9554**	0.9246	0.9301	0.9191
	RFE	0.9329	0.9024	0.9138	0.9241	0.9333
	GFS	0.9452	0.9239	0.9346	0.9404	0.9349
	SARLFS	0.9352	0.9292	0.9201	0.9302	0.9402
	MARLFS	**0.9560**	0.9532	**0.9592**	**0.9559**	**0.9603**



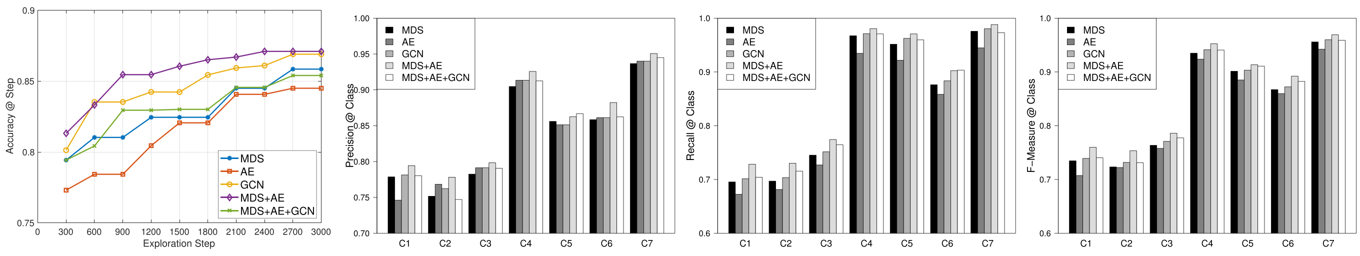
(a) Overall Accuracy (b) Precision (c) Recall (d) F-Measure

Fig. 9. Performance comparison of different reward functions on Dataset 1.



(a) Overall Accuracy (b) Precision (c) Recall (d) F-Measure

Fig. 10. Performance comparison of different reward functions on Dataset 2.



(a) Overall Accuracy (b) Precision (c) Recall (d) F-Measure

Fig. 11. Performance comparison of different representation learning methods on Dataset 1.

Dynamic-Graph Based GCN; (iv) *MDS+AE*: combines the variables of (i) and (ii) to represent the state; (v) *MDS+AE +GCN*: combines the variables of (i), (ii), and (iii) to represent the state.

Figs. 11 and 12 show GCN outperforms MDS and AE. This is because GCN could better capture the relationship between features in the feature subspace. After taking the two combined methods into account, MDS+AE achieves the best performance, since it considers both explicit and implicit information from the selected features. An interesting observation is that MDS+AE+GCN doesn't have better performance than MDS+AE+GCN. This might be explained by the fact that there is potential training loss in the training phrase of AE and GCN. In other words, integrating AE and GCN might possibly introduce more model biases. Specifically, Figs. 11a and 12a show the comparisons of overall accuracy over exploration steps. Figs. 11b and 12b, 11c and 12c, 11d and 12d show the comparisons of precision, recall and F-measure over different classes with 3000 exploration steps.

6.8 Study of GMM-Based Generative Rectified Sampling

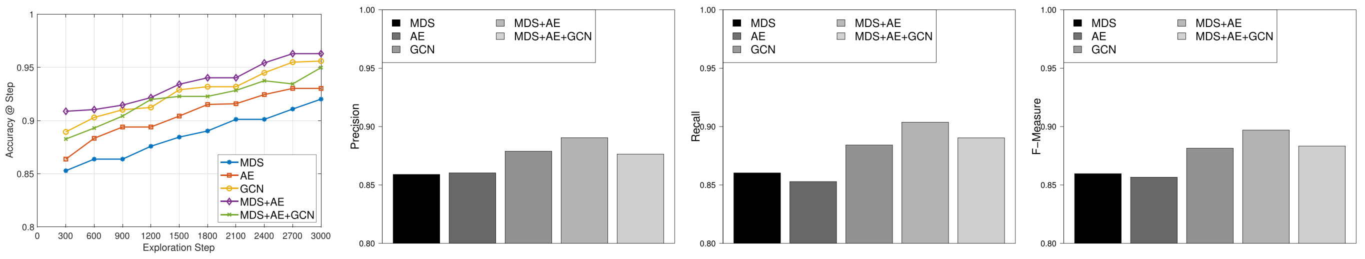
We study the impacts of GMM-based generative rectified sampling, where the high-quality proportion $p \in [0.1, 0.2, 0.3, 0.4, 1]$ respectively. Here, when $p = 1$, our GMM-based method reduces to the traditional sampling strategy, where samples are considered as high-quality. We call the method with $p = 1$ as the non-GMM method.

Figs. 13 and 14 show all GMM-based sampling methods ($p < 1.0$) outperform the non-GMM method ($p = 1.0$). For Dataset 1, $p = 0.2$ shows the best performances and can quickly explore a quality feature space, while For Dataset 2, $p = 0.3$ shows the best performances and can quickly explore a quality feature space. Specifically, Figs. 13a and 14a show the comparisons of overall accuracy over exploration steps. Figs. 13b and 14b, 13c and 14c, 13d and 14d show the comparisons of precision, recall and F-measure over different classes with 3000 exploration steps.

6.9 Study of Softmax Sampling

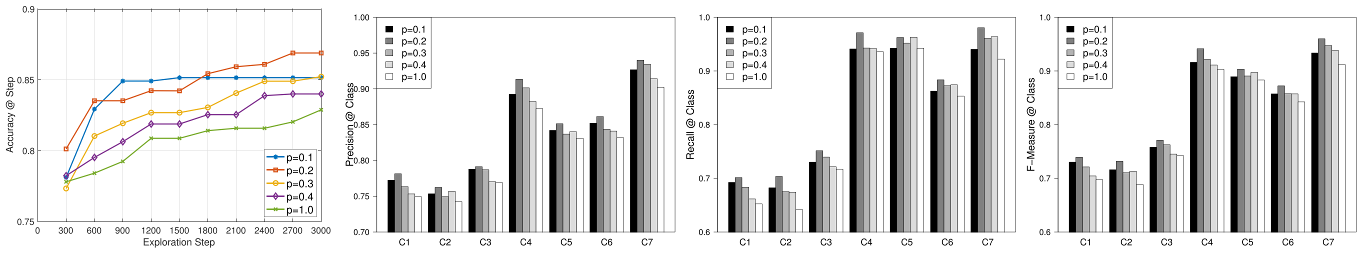
We study the impacts of softmax sampling and compare it with vanilla experience replay (uniformly sampling) and GMM-based sampling. We use meta descriptive statistics as the state representation method, and we use MARLFS as feature selection method. We run each setting 5 times to compare their averaged execution time per step. The exploration step is set to 3000. Our experiments were conducted on a machine with the following specification:

- Processor: 64-bit Intel I9-9920X @ 4.40 GHz with 12 core(s)



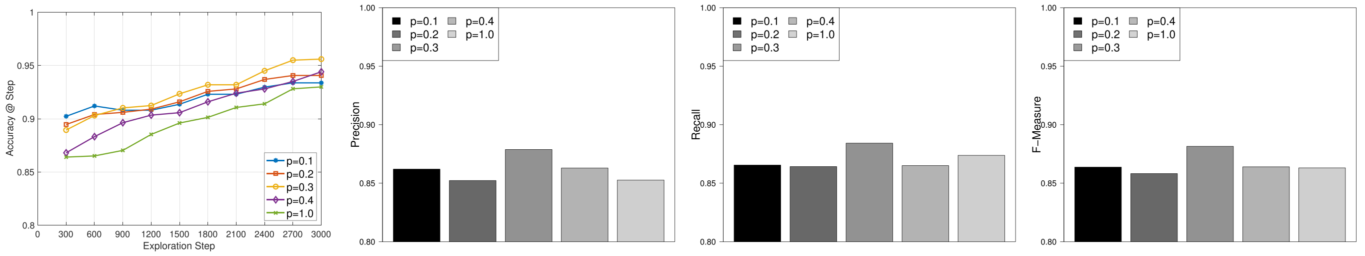
(a) Overall Accuracy (b) Precision (c) Recall (d) F-Measure

Fig. 12. Performance comparison of different representation learning methods on Dataset 2.



(a) Overall Accuracy (b) Precision (c) Recall (d) F-Measure

Fig. 13. Performance comparison of different GMM sampling strategies on Dataset 1.



(a) Overall Accuracy (b) Precision (c) Recall (d) F-Measure

Fig. 14. Performance comparison of different GMM sampling strategies on Dataset 2.

- Memory: 128GiB DIMM DDR4 2666MHz
- SSD: 2 TB PCIe NVMe - M.2

Tables 3 and 4 show that the GMM-based sampling method achieves the best accuracy, and the softmax sampling method has a parellel accuracy with GMM-based sampling, which are both higher than the uniform sampling. Tables 5 and 6 show that the softmax sampling method costs a bit more time than uniform sampling, and both of them require less execution time than the GMM-based sampling method.

6.10 Study of Interactive Reinforcement Learning

We study the impacts of IRL, where its apprenticeship steps $N_A \in [0, 0.1, 0.2, 0.3, 0.4, 1] * 3000$ respectively. We set $K = 38$ for K-Best Selection on Dataset1 and $K = 82$ for K-Best Selection on Dataset2. Here, when $N_A = 0$, the IRL strategy will be reduced into the traditional exploration strategy, and when $N_A = 3000$, the IRL spends all its exploration

TABLE 4
Overall Accuracy Comparison of Sampling Strategies on Dataset 2

Dataset 2		Predictors				
		RF	LASSO	DT	SVM	XGBoost
Sampling	Uniform	0.9202	0.9330	0.9347	0.9287	0.9391
	GMM	0.9560	0.9532	0.9592	0.9559	0.9603
	Softmax	0.9423	0.9492	0.9533	0.9440	0.9586

TABLE 3
Overall Accuracy Comparison of Sampling Strategies on Dataset 1

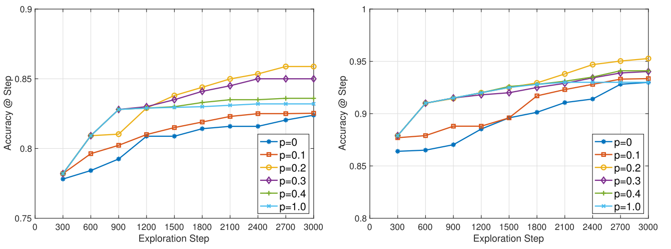
Dataset 1		Predictors				
		RF	LASSO	DT	SVM	XGBoost
Sampling	Uniform	0.8585	0.8381	0.8392	0.8406	0.8627
	GMM	0.8690	0.8424	0.8583	0.8542	0.8731
	Softmax	0.8633	0.8400	0.8583	0.8496	0.8659

TABLE 5
Execution Time Comparison of Sampling Strategies on Dataset 1

Dataset 1		Predictors				
		RF	LASSO	DT	SVM	XGBoost
Sampling	Uniform	1.04 s	0.46 s	0.50 s	17.34 s	4.97 s
	GMM	1.44 s	0.93 s	0.94 s	17.86 s	5.42 s
	Softmax	1.06 s	0.51 s	0.63 s	17.46 s	5.05 s

TABLE 6
Execution Time Comparison of Sampling Strategies on Dataset 2

Dataset 2		Predictors				
		RF	LASSO	DT	SVM	XGBoost
Sampling	Uniform	1.25 s	0.63 s	0.73 s	26.54 s	7.07 s
	GMM	1.69 s	1.15 s	1.13 s	27.20 s	7.52 s
	Softmax	1.32 s	0.74 s	0.85 s	26.63 s	7.18 s



(a) Overall Accuracy on Dataset 1 (b) Overall Accuracy on Dataset 2

Fig. 15. Performance comparison of different IRL sampling strategies.

steps being advised by K-Best Selection method. Fig. 15 shows that all IRL sampling methods ($p = 0.0$) outperform the non-GMM method ($p > 0.0$). For both Dataset 1 and Dataset 2, $p = 0.2$ shows the best performances and can quickly explore a quality feature space.

7 Conclusion Remarks

In this paper, we study the problem of automated feature subspace exploration. Through this method, we can reduce dimensionality, shorten training times, enhance generalization, avoid overfitting, and improve predictive accuracy in order to support downstream predictive tasks. We formulate the problem of automated feature subspace exploration as a multi-agent reinforcement learning framework, in which each feature is associated to a feature agent, a feature agent can decide to select or drop a feature, the reward function is a combination of accuracy, redundancy, and relevance, and the environment is the characteristics of the selected feature subspace. To better represent the environment, we propose three different representation learning methods. To accelerating feature exploration, we develop a GMM-based generative rectified sampling strategy a softmax sampling strategy and an IRL-based exploration strategy. Finally, we present extensive experiments on two real world datasets to demonstrate the effectiveness of the proposed method. The proposed feature selection method can be widely used in the data mining and machine learning areas, and it is more helpful when users are not experts in feature selection but need this technology in their work and research.

References

[1] G. Chandrashekar and F. Sahin, “A survey on feature selection methods,” *Comput. Elect. Eng.*, vol. 40, no. 1, pp. 16–28, 2014.
[2] L. Yu and H. Liu, “Feature selection for high-dimensional data: A fast correlation-based filter solution,” in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 856–863.
[3] Y. Yang and J. O. Pedersen, “A comparative study on feature selection in text categorization,” in *Proc. Int. Conf. Mach. Learn.*, 1997, pp. 412–420.
[4] G. Forman, “An extensive empirical study of feature selection metrics for text classification,” *J. Mach. Learn. Res.*, vol. 3, no. Mar., pp. 1289–1305, 2003.
[5] M. A. Hall, “Feature selection for discrete and numeric class machine learning,” Dept. Comput. Sci., Univ. Waikato, Hamilton, New Zealand, 1999.
[6] J. Yang and V. Honavar, “Feature subset selection using a genetic algorithm,” in *Feature Extraction, Construction and Selection*. Berlin, Germany: Springer, 1998, pp. 117–136.
[7] Y. Kim, W. N. Street, and F. Menczer, “Feature selection in unsupervised learning via evolutionary search,” in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2000, pp. 365–369.
[8] P. M. Narendra and K. Fukunaga, “A branch and bound algorithm for feature subset selection,” *IEEE Trans. Comput.*, vol. C-26, no. 9, pp. 917–922, Sep. 1977.
[9] R. Kohavi and G. H. John, “Wrappers for feature subset selection,” *Artif. Intell.*, vol. 97, no. 1/2, pp. 273–324, 1997.
[10] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *J. Roy. Stat. Soc. Ser. Methodol.*, vol. 58, pp. 267–288, 1996.
[11] V. Sugumaran, V. Muralidharan, and K. Ramachandran, “Feature selection using decision tree and classification through proximal support vector machine for fault diagnostics of roller bearing,” *Mech. Syst. Signal Process.*, vol. 21, no. 2, pp. 930–942, 2007.
[12] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang, “Deep reinforcement learning for page-wise recommendations,” 2018, *arXiv: 1805.02343*.
[13] K. Lin, R. Zhao, Z. Xu, and J. Zhou, “Efficient large-scale fleet management via multi-agent deep reinforcement learning,” 2018, *arXiv:1802.06444*.
[14] S. M. H. Fard, A. Hamzeh, and S. Hashemi, “Using reinforcement learning to find an optimal set of features,” *Comput. Math. Appl.*, vol. 66, no. 10, pp. 1892–1904, 2013.
[15] M. Kroon and S. Whiteson, “Automatic feature selection for model-based reinforcement learning in factored MDPs,” in *Proc. Int. Conf. Mach. Learn. Appl.*, 2009, pp. 324–330.
[16] F.-A. Fortin, F.-M. D. Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, “DEAP: Evolutionary algorithms made easy,” *J. Mach. Learn. Res.*, vol. 13, no. Jul., pp. 2171–2175, 2012.
[17] L.-J. Lin, “Self-improving reactive agents based on reinforcement learning, planning and teaching,” *Mach. Learn.*, vol. 8, no. 3/4, pp. 293–321, 1992.
[18] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, 2015, Art. no. 529.
[19] F. Cruz, S. Magg, Y. Nagai, and S. Wermter, “Improving interactive reinforcement learning: What makes a good teacher?” *Connection Sci.*, vol. 30, no. 3, pp. 306–325, 2018.
[20] F. Cruz, J. Twiefel, S. Magg, C. Weber, and S. Wermter, “Interactive reinforcement learning through speech guidance in a domestic scenario,” in *Proc. Int. Joint Conf. Neural Netw.*, 2015, pp. 1–8.
[21] Y. Saeys, I. Inza, and P. Larrañaga, “A review of feature selection techniques in bioinformatics,” *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.
[22] D. Guo, H. Xiong, V. Atluri, and N. Adam, “Semantic feature selection for object discovery in high-resolution remote sensing imagery,” in *Proc. Pacific-Asia Conf. Knowl. Discov. Data Mining*, 2007, pp. 71–83.
[23] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, “Gene selection for cancer classification using support vector machines,” *Mach. Learn.*, vol. 46, no. 1–3, pp. 389–422, 2002.
[24] A. Tampuu *et al.*, “Multiagent cooperation and competition with deep reinforcement learning,” *PLoS One*, vol. 12, no. 4, 2017, Art. no. e0172395.
[25] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[26] M. Stankovic, “Multi-agent reinforcement learning,” in *Proc. 13th Symp. Neural Netw. Appl.*, 2016, pp. 1–1.
[27] H. L. Liao, Q. H. Wu, and L. Jiang, “Multi-objective optimization by reinforcement learning for power system dispatch and voltage stability,” in *Proc. Innov. Smart Grid Technol. Conf. Europe*, 2010, pp. 1–8.
[28] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, “Mean field multi-agent reinforcement learning,” 2018, *arXiv:1802.05438*.
[29] P. Peng *et al.*, “Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play StarCraft combat games,” 2017, *arXiv:1703.10069*.

[30] Y. Bengio *et al.*, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[31] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[32] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*.

[33] H. Wei, G. Zheng, H. Yao, and Z. Li, "IntelliLight: A reinforcement learning approach for intelligent traffic light control," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2496–2505.

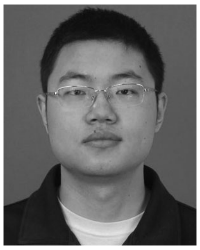
[34] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc. Ser. Methodol.*, vol. 39, pp. 1–22, 1977.

[35] L. Candillier and V. Lemaire, "Design and analysis of the Nomao challenge active learning in the real-world," in *Proc. Act. Learn. Real-World Appl. Workshop ECML-PKDD*, 2012, pp. 1–15.

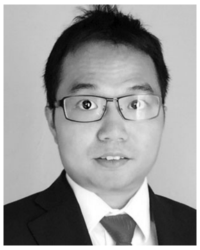
[36] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1226–1238, Aug. 2005.

[37] P. M. Granitto, C. Furlanello, F. Biasioli, and F. Gasperi, "Recursive feature elimination with random forest for PTR-MS analysis of agroindustrial products," *Chemometrics Intell. Lab. Syst.*, vol. 83, no. 2, pp. 83–90, 2006.

[38] R. Leardi, "Genetic algorithms in feature selection," in *Genetic Algorithms in Molecular Modeling*. Amsterdam, The Netherlands: Elsevier, 1996, pp. 67–86.



Kunpeng Liu received the BE and ME degrees from the University of Science and Technology of China (USTC), in 2011 and 2015, respectively. He is currently working toward the PhD degree with the University of Central Florida (UCF). His general research interests include data mining and reinforcement learning. He has published prolifically in refereed journals and conference proceedings, such as ACM SIGKDD, IEEE ICDM, and SIAM SDM.



Yanjie Fu received the BE degree from the University of Science and Technology of China, in 2008, the ME degree from the Chinese Academy of Sciences, in 2011, and the PhD degree from Rutgers, the State University of New Jersey, in 2016. He is an assistant professor with the Department of Computer Science, University of Central Florida. His research interests include data mining and Big Data analytics. He has published prolifically in refereed journals and conference proceedings, such as the *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Mobile Computing*, *ACM Transactions on Knowledge Discovery from Data* and ACM SIGKDD. He received US NSF CAREER Award in 2021, ACM SIGSpatial Best Paper Runner-Up Award in 2020, US NSF CRII Award in 2018, ACM SIGKDD Best Student Paper Runner-Up Award in 2018, Microsoft Research Azure Research Award in 2016, and IEEE ICDM Best Paper Runner-Up Award in 2014.

Le Wu received the PhD degree in computer science from the University of Science and Technology of China (USTC), in 2016. She is currently an associate professor with the Lab of Media Computing, School of Computer and Information, Hefei University of Technology, China. Her general research interests include data mining, recommender system, and social network analysis. She has published more than 40 papers in referred journals and conferences, such as the *IEEE Transactions on Knowledge and Data Engineering* (TKDE), *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *ACM Transactions on Intelligent System and Technology* (TIST). She is the recipient of the Best of SDM 2015 Award, and recipient of the Distinguished Dissertation Award from China Association for Artificial Intelligence (CAAI).

Xiaolin Li received the PhD degree from Jilin University, China. She is currently a professor with the School of Management, Nanjing University, China. Her main research interests include business intelligence, data mining, and decision-making. She has published a number of papers in refereed journals and conference proceedings, such as the *Information Sciences* (INS), ECML, and PAKDD. She also has served as a reviewer and PC member for numerous journals and conferences, such as the the *IEEE Transactions on Evolutionary Computation*, *Knowledge and Information Systems*, and KDD15.

Charu Aggarwal (Fellow, IEEE) received the BS degree from IIT Kanpur, in 1993, and the PhD degree from the Massachusetts Institute of Technology, in 1996. He is a research scientist with the IBM T.J. Watson Research Center in Yorktown Heights, New York. He has since worked in the field of performance analysis, databases, and data mining. He has served on the program committees of most major database/data mining conferences, and served as program vice-chairs of SDM 2007, ICDM 2007, WWW 2009, and ICDM 2009. He served as an associate editor of the *IEEE Transactions on Knowledge and Data Engineering* (TKDE) from 2004 to 2008. He is an associate editor of the *ACM Transactions on Knowledge Discovery from Data* (TKDD), an action editor of the *Data Mining and Knowledge Discovery*, an associate editor of SIGKDD Explorations, and an associate editor of *Knowledge and Information Systems*. He is a fellow of the ACM.

Hui Xiong (Fellow, IEEE) received the BE degree from the University of Science and Technology of China (USTC), China, the MS degree from the National University of Singapore (NUS), Singapore, and the PhD degree from the University of Minnesota (UMN), USA. He is currently a distinguished professor with Rutgers, the State University of New Jersey. His general research interests include data and knowledge engineering, with a focus on developing effective and efficient data analysis techniques for emerging data intensive applications. He currently serves as a co-editor-in-chief of *Encyclopedia of GIS* (Springer) and an associate editor of the *IEEE Transactions on Data and Knowledge Engineering* (TKDE), *IEEE Transactions on Big Data* (TBD), *ACM Transactions on Knowledge Discovery from Data* (TKDD) and *ACM Transactions on Management Information Systems* (TMIS). He is a fellow of the AAAS.