Automating Feature Subspace Exploration via Multi-Agent Reinforcement Learning

Kunpeng Liu
University of Central Florida
Florida, US
kunpengliu@knights.ucf.edu

Yanjie Fu*
University of Central Florida
Florida, US
Yanjie.Fu@ucf.edu

Pengfei Wang
CNIC, Chinese Academy of Sciences
Beijing, China
wpf@cnic.cn

Le Wu
Hefei University of Technology
Hefei, China
lewu@hfut.edu.cn

Rui Bo
Missouri Univ. of Sci. and Tech.
Missouri, US
rbo@mst.edu

Xiaolin Li
Nanjing University
Nanjing, China
lixl@nju.edu.cn

ABSTRACT

Feature selection is the preprocessing step in machine learning which tries to select the most relevant features for the subsequent prediction task. Effective feature selection could help reduce dimensionality, improve prediction accuracy and increase result comprehensibility. It is very challenging to find the optimal feature subset from the subset space as the space could be very large. While much effort has been made by existing studies, reinforcement learning can provide a new perspective for the searching strategy in a more global way. In this paper, 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. Then, we obtain the state of environment in three ways, i.e., statistic description, autoencoder and graph convolutional network (GCN), in order to make the algorithm better understand the learning progress. We show how to learn the state representation in a graph-based way, which could tackle the case when not only the edges, but also the nodes are changing step by step. In addition, we study how the coordination between different features would be improved by more reasonable reward scheme. The proposed method could search the feature subset space globally and could be easily adapted to the real-time case (real-time feature selection) due to the nature of reinforcement learning. Also, we provide an efficient strategy to accelerate the convergence of multi-agent reinforcement learning. Finally, extensive experimental results show the significant improvement of the proposed method over conventional approaches.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent reinforcement learning**; **Feature selection**.

*Contact Author.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330868

KEYWORDS

feature selection; automated exploration; multi-agent reinforcement learning

ACM Reference Format:
Kunpeng Liu, Yanjie Fu, Pengfei Wang, Le Wu, Rui Bo, and Xiaolin Li. 2019. Automating Feature Subspace Exploration via Multi-Agent Reinforcement Learning. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330868

1 INTRODUCTION

Feature selection aims to select an optimal subset of relevant features for a downstream predictive task [2, 34]. Effective feature selection can help to reduce dimensionality, shorten training times, enhance generalization, avoid overfitting, improve predictive accuracy, and provide better interpretation and explanation. In this paper, we study the problem of automated feature subspace exploration for improving downstream predictive tasks.

Prior studies in feature selection can be grouped into three categories: (i) filter methods (e.g., univariate feature selection [5, 33], correlation based feature selection [10, 34]), in which features are ranked by a specific score; (ii) wrapper methods (e.g., evolutionary algorithms [11, 31], branch and bound algorithms [13, 20]), in which optimal feature subset is identified by a search strategy that collaborates with predictive tasks; (iii) embedded methods (e.g., LASSO [29], decision tree [26]), 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 feature number. Embedded methods are subject to the strong structured assumptions of predictive models. As can be seen, feature selection is a complicated process that requires (i) strategic design of feature significance measurement, (ii) accelerated search of near-optimized 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 [17, 35]. These traits provide great potential to automate feature subspace exploration. Existing studies [4, 14] 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, 11, 31], which are NP-hard and can merely obtain local optima. In this paper, we intend to propose a better solution using reinforcement learning for feature selection. However, several challenges arise toward this goal.

First, how can we reformulate the problem so that the action space could be limited? We reformulate the problem with multi-agent reinforcement learning. We assign an 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. One key challenge in multi-agent learning is to coordinate the interactions between agents. The interactions can be from two aspects: (i) cooperation and (ii) competition between agents, which can be quantified by feature-feature mutual information in our case. We propose to integrate feature-feature mutual information with predictive accuracy as the reward scheme. In this way, we guide the cooperation and competition between agents for effective feature exploration.

Second, how can we accurately describe the state representation in multi-agent reinforcement learning? We regard the selected feature subspace as the state of environment. To construct state representations, traditional methods are to extract descriptive statistics (e.g., mean, variance) of the state distribution. However, in feature subspace exploration, the number of selected features changes over time during the exploratory process. If we also extract the mean and variance of each selected feature to describe the state, length of the state representation vector will change over time. But, the policy networks in multi-agent reinforcement learning require a fixed-length state representation. To tackle this challenge, we develop three different methods: (i) meta descriptive statistics, (ii) auto-encoder based deep representation, (iii) dynamic graph based graph convolutional network (GCN). In Method i, after extracting the first round descriptive statistics of each selected feature, we extract the second-round descriptive statistics of these first-round descriptive statistics as the state representation vector, so that the state length will not change along with the varying number of selected features. In Method ii, since the number of rows are static in the selected feature subspace, we construct a row-row similarity graph, namely static subspace graph, to describe the state. An autoencoder method is applied to learn the state representation. In Method iii, 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.

Third, how can we improve the robustness of our framework against a vastly different state distribution, while accelerating the exploration of optimal features? Traditionally, we can use experience replay [18, 19] 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, noise, outliers, or low-rewarded 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 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 independences of samples between different training steps, as it is equivalent to reduce the memory size. 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 a sufficient number of independent samples from different mixture distribution components for reinforcement learning.

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 subspace exploration with a multi-agent reinforcement learning framework and integrate the interactions between features into a new reward scheme. (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 a GMM-based generative rectified sampling method to improve the training and exploration. (4) We conduct extensive experiments to demonstrate the enhanced performances of our method.

2 PROBLEM FORMULATION

We study the problem of feature subspace exploration, which is formulated as a multi-agent reinforcement learning task. Figure 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.

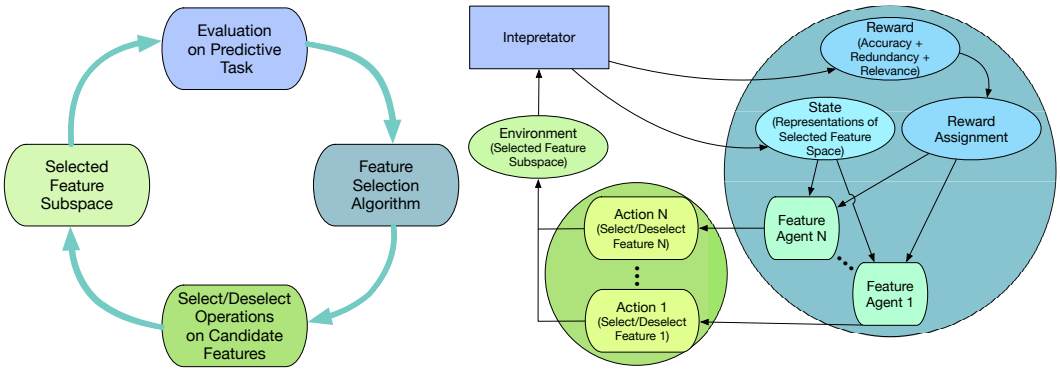
Multi-Agent. Assuming there are N features, we define N agents for the N features. For one agent, it is designed to make the selection decision for the corresponding feature.

Actions. For the i-th feature agent, the feature action $a_i = 1$ indicates the i-th feature is selected, and $a_i = 0$ indicates the i-th 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 3.3.

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



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

Figure 1: From traditional feature selection to multi-agent reinforcement learning based feature subspace exploration.

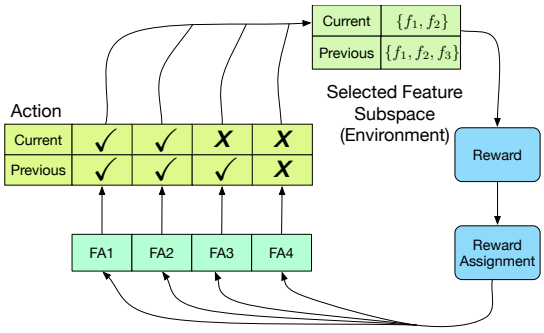


Figure 2: The demonstration of reward assignment process. The feature agent 1 and 2 issue an action to select the feature 1 and feature 2. The feature agent 3 issues an action to disselect the feature 3.

subset *Acc*, (ii) redundancy of the selected feature subset *Rv*, and (iii) relevance of the selected feature subset *Rd*.

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. Figure 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 unselected. 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.

3 PROPOSED METHOD

We first present the multi-agent reinforcement learning framework for automated feature subspace exploration. Later, we discuss how to measure reward, how to improve state representation, and how to accelerate feature subspace exploration.

3.1 Framework Overview

Figure 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 3.3). 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. We then propose a GMM-based generative rectified sampling (refer to Section 3.4) to derive mini-batches from memories. 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 [27]:

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

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

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

3.2 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* 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

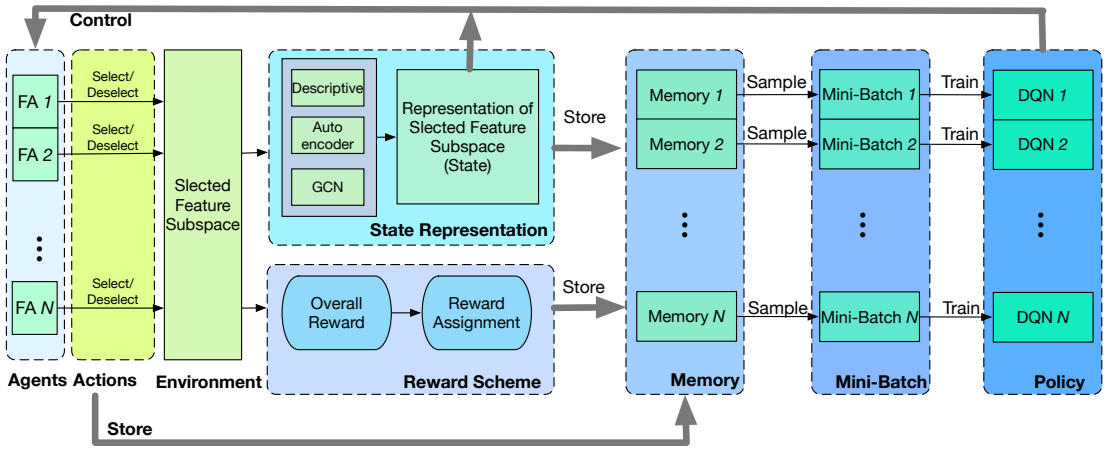


Figure 3: Framework.The framework consists of two stages. In the control stage, feature agents select or drop their corresponding features based on policies. In the training stage, the policies are trained via samples from memories.

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:

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

where x_i, y_i is the i-th and j-th 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:

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

  where $\boldsymbol{S}$ is the feature subset, $\boldsymbol{x_i}$ is the i-th 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.

3.3 Improving State Representation

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

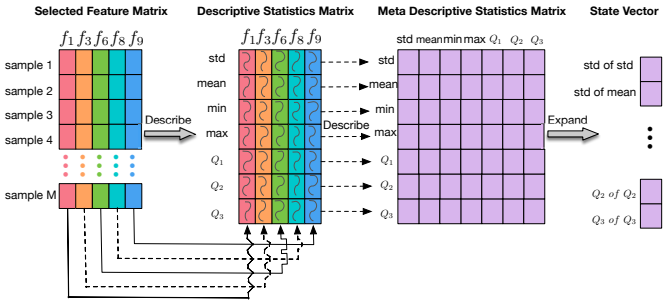


Figure 4: Meta descriptive statistics. We extract descriptive statistics twice from the feature subspace to obtain a fixed-length state vector.

Method 1: Meta Descriptive Statistics of Feature Subspace. Figure 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 $\boldsymbol{S}$, 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 $\boldsymbol{S}$, 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 $\boldsymbol{s}$ 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 [1]. 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.

Figure 5 shows a two-step algorithm to learn the state vector.

Step 1: Assuming at the j-th exploration step, $\boldsymbol{S}$ is the selected data matrix, and n_j is the number of selected features. For each feature (**column**) in $\boldsymbol{S}$, 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 j-th 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 $\boldsymbol{s}$ with a fixed length of $k * o$.

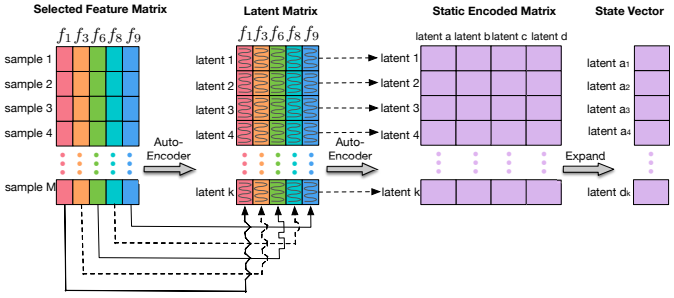


Figure 5: Autoencoder based deep representations. We use two auto-encoders to map the feature subspace into a fixed-length state vector.

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. Figure 6 shows how the GCN works. To better capture the relationship among features, we first convert the selected data matrix $\boldsymbol{S}$ into a dynamic complete graph G, where a node is a feature column in $\boldsymbol{S}$. With the *feature correlation graph* 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 $\boldsymbol{S}$ be the selected data matrix with a dimension of $M * N$, $\boldsymbol{Z}$ 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:

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

where $\boldsymbol{H}^{(0)} = \boldsymbol{S}$, $\boldsymbol{H}^{(L)} = \boldsymbol{Z}$, L is the layer number, $\boldsymbol{A}$ 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 [12]:

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

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

By solving GCN, we obtain the latent representations $\boldsymbol{Z}$ of each feature. We average the representation of each feature into the k-legnth state representation vector.

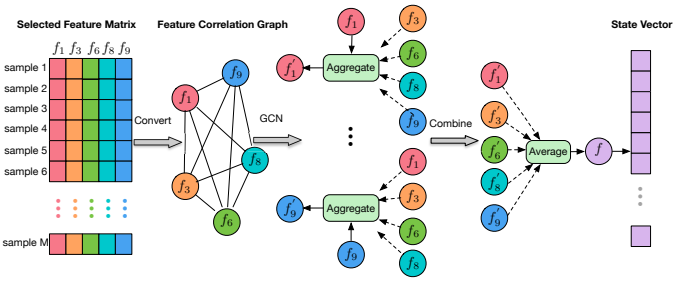


Figure 6: Dynamic-graph based GCN. We denote the feature subspace by a dynamic graph and use GCN to update representations of each node.

3.4 Accelerating Feature Subspace Exploration via Generative Rectified Sampling

Experience replay is widely used to improve training efficiency of neural networks in reinforcement learning [18, 19]. 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 ($s^{'}$), is stored into the memory to replace the oldest sample. In training step, a mini-batch of samples are picked to update the policy network.

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 [24, 30]. However, such strategy create a new problem: the sampler repeatedly select a very limited number of high-quality samples. Consequently, prior studies can not guarantee the independences 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 shown in algorithm 1, we take a set of memory samples $\boldsymbol{T} = \{< a, r, s, s^{'} >\}$ as inputs. We firstly cluster the memory samples into two groups: $\boldsymbol{T_0}$ and $\boldsymbol{T_1}$. Samples with the selected action ($a = 0$) are assigned to group $\boldsymbol{T_0}$, while samples with the deselected action ($a = 1$) are assigned to group $\boldsymbol{T_1}$. Later, we rank the memory samples in $\boldsymbol{T}$ 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 [3]. 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 $\boldsymbol{T}$.
Output : A mini-batch of samples $\boldsymbol{B}$.

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

4 EXPERIMENTAL RESULTS

We evaluate the proposed method in feature selection with real-world data.

4.1 Data Description

The experiments are conducted on a publicly available 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.

4.2 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 for all cover types. The latter three metrics measure classification performance of each cover type 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 k-th ($k \in [1, 7]$) type of cover.

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 k-th ($k \in [1, 7]$) type of cover.

[1] https://www.kaggle.com/c/forest-cover-type-prediction/data

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.

4.3 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 [33] firstly 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 [21] firstly 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 [29] conducts feature selection and 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 [7] selects features by recursively selecting smaller and smaller feature subsets. Firstly, 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 [15] selects features by firstly 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 [14], 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 $\epsilon = 0.15$, $\delta = 0.10$.

4.4 Overall Performances

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 2000. 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. 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

Table 1: Overall accuracy of feature selection algorithms on different predictors.

		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**

random forest with 100 decision trees. Figure 7 shows that our method exceeds all of the baseline methods in the task of exploring a qualified feature subset.

4.5 Robustness Check

The predictive accuracy relies on not just feature selection, but also predictors. We apply our method to different 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. Table 1 shows 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.

4.6 Study of Reward Function

We study the impacts of the reward function design in our method. 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.

Figure 8 shows 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, Figure 8(a) shows the comparisons of overall accuracy over exploration steps. Figure 8(b), 8(c) and 8(d) show the comparisons of precision, recall and F-measure over different classes with 3000 exploration steps.

4.7 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 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.

Figure 9 shows GCN outperform 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, Figure 9(a) shows the comparisons of overall accuracy over exploration steps. Figure 9(b), 9(c) and 9(d) show the comparisons of precision, recall and F-measure over different classes with 3000 exploration steps.

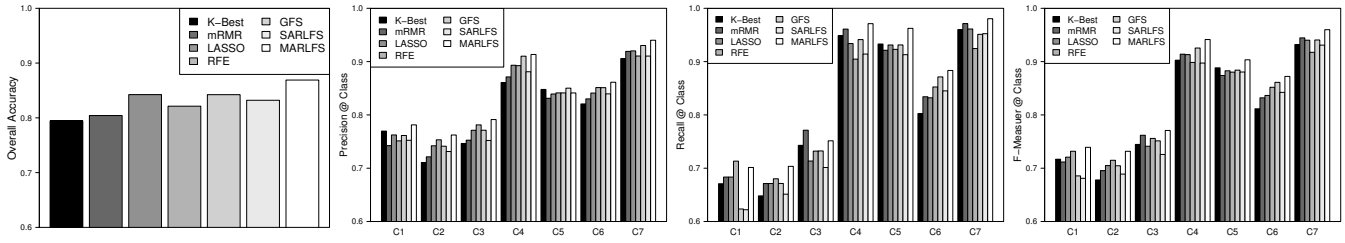
4.8 Study of GMM based Generative Rectified Sampling

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 will be reduced into the traditional sampling strategy, where samples are considered as high-quality. We call the method with $p = 1$ as the non GMM method.

Figure 10 all GMM-based sampling methods ($p < 1$) outperform the non-GMM method. Among the all these methods, $p = 0.2$ shows the best performances and can quickly explore a quality feature space. Specifically, 10(a) shows the comparisons of overall accuracy over exploration steps. Figure 10(b), 10(c) and 10(d) show the comparisons of precision, recall and F-measure over different classes with 3000 exploration steps.

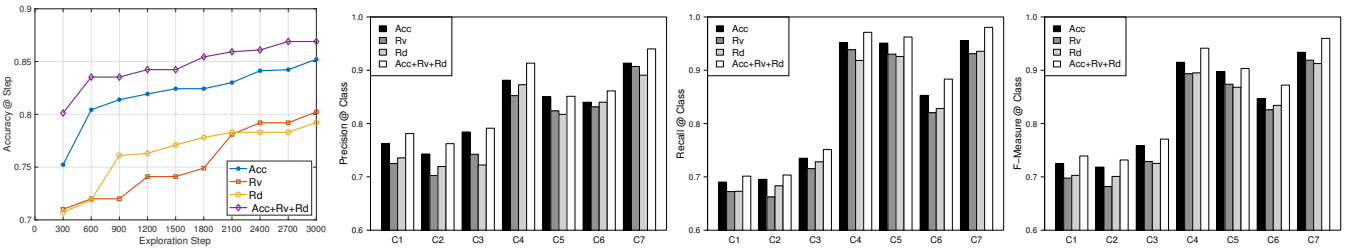
5 RELATED WORK

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 [2, 23]. Filter methods rank the features merely by relevance scores and only top-ranking features are selected. The representative filter methods are univariate feature selection [5, 33] and correlation based feature selection [10, 34]. 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 [8]. The representative wrapper methods are branch and bound algorithms [13, 20]. 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, 11, 31] low down the computational cost but could only promise local optimum results. Embedded methods combine feature



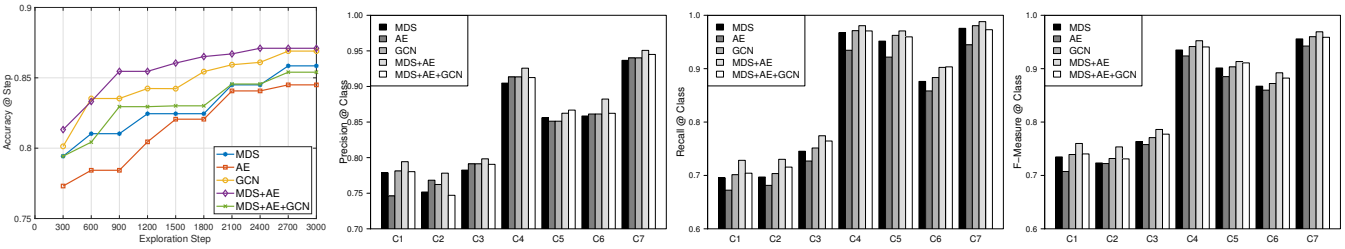
(a) Overall Accuracy (b) Precision (c) Recall (d) F-Measure

Figure 7: Performance comparison of different feature selection algorithms.



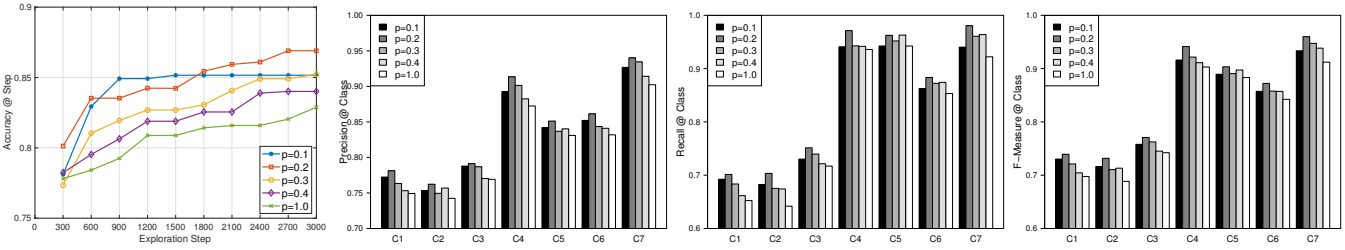
(a) Overall Accuracy (b) Precision (c) Recall (d) F-Measure

Figure 8: Performance comparison of different reward functions.



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

Figure 9: Performance comparison of different representation learning methods.



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

Figure 10: Performance comparison of different GMM sampling strategies.

selection with predictors more closely than wrapper methods, and actually they incorporate feature selection as part of predictors. The most widely used embedded methods are LASSO [29], decision tree [26] and SVM-RFE [9]. Embedded methods could have supreme performance on the incorporated predictors, but normally not very compatible with other predictors.

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 [28]. *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 [25]. *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 [16]. *Yang et al.* developed deep reinforcement learning algorithms which could handle large scale agents with effective communication protocol [22, 32]. *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 [17]. 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.

Reinforcement Learning in Feature Selection. Existing studies [4, 14] create a single agent to make decisions. However, this agent has to determine the selection or disselection 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, 11, 31], which are NP-hard problems and can merely obtain local optima.

6 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 method. Finally, we present extensive experiments on a real world dataset to demonstrate the effectiveness of the proposed method.

ACKNOWLEDGEMENTS

This research was supported by the National Science Foundation (NSF) via the grant number: 1755946.

REFERENCES

[1] Yoshua Bengio et al. 2009. Learning deep architectures for AI. *Foundations and trends® in Machine Learning* 2, 1 (2009), 1–127.
[2] Girish Chandrashekar and Ferat Sahin. 2014. A survey on feature selection methods. *Computers & Electrical Engineering* 40, 1 (2014), 16–28.
[3] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)* (1977), 1–38.
[4] Seyed Mehdi Hazrati Fard, Ali Hamzeh, and Sattar Hashemi. 2013. Using reinforcement learning to find an optimal set of features. *Computers & Mathematics with Applications* 66, 10 (2013), 1892–1904.
[5] George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research* 3, Mar (2003), 1289–1305.
[6] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research* 13, Jul (2012), 2171–2175.
[7] Pablo M Granitto, Cesare Furlanello, Franco Biasioli, and Flavia Gasperi. 2006. Recursive feature elimination with random forest for PTR-MS analysis of agroindustrial products. *Chemometrics and Intelligent Laboratory Systems* 83, 2 (2006), 83–90.
[8] Dihua Guo, Hui Xiong, Vijay Atluri, and Nabil Adam. 2007. Semantic feature selection for object discovery in high-resolution remote sensing imagery. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 71–83.
[9] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. 2002. Gene selection for cancer classification using support vector machines. *Machine learning* 46, 1-3 (2002), 389–422.
[10] Mark A Hall. 1999. Feature selection for discrete and numeric class machine learning. (1999).
[11] YeongSeog Kim, W Nick Street, and Filippo Menczer. 2000. Feature selection in unsupervised learning via evolutionary search. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 365–369.
[12] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[13] Ron Kohavi and George H John. 1997. Wrappers for feature subset selection. *Artificial intelligence* 97, 1-2 (1997), 273–324.
[14] Mark Kroon and Shimon Whiteson. 2009. Automatic feature selection for model-based reinforcement learning in factored MDPs. In *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*. IEEE, 324–330.
[15] Riccardo Leardi. 1996. Genetic algorithms in feature selection. In *Genetic algorithms in molecular modeling*. Elsevier, 67–86.
[16] HL Liao, QH Wu, and L Jiang. 2010. Multi-objective optimization by reinforcement learning for power system dispatch and voltage stability. In *Innovative Smart Grid Technologies Conference Europe (ISGT Europe), 2010 IEEE PES*. IEEE, 1–8.
[17] Kaixiang Lin, Renyu Zhao, Zhe Xu, and Jiayu Zhou. 2018. Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning. *arXiv preprint arXiv:1802.06444* (2018).
[18] Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning* 8, 3-4 (1992), 293–321.
[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.
[20] Patrenahalli M. Narendra and Keinosuke Fukunaga. 1977. A branch and bound algorithm for feature subset selection. *IEEE Transactions on computers* 9 (1977), 917–922.
[21] Hanchuan Peng, Fuhui Long, and Chris Ding. 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence* 27, 8 (2005), 1226–1238.
[22] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. 2017. Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games. *arXiv preprint arXiv:1703.10069* (2017).
[23] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. 2007. A review of feature selection techniques in bioinformatics. *bioinformatics* 23, 19 (2007), 2507–2517.
[24] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952* (2015).
[25] Milos Stankovic. 2016. Multi-agent reinforcement learning. In *Neural Networks and Applications (NEUREL), 2016 13th Symposium on*. IEEE, 1–1.
[26] V Sugumaran, V Muralidharan, and KI Ramachandran. 2007. Feature selection using decision tree and classification through proximal support vector machine for fault diagnostics of roller bearing. *Mechanical systems and signal processing* 21, 2 (2007), 930–942.
[27] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.
[28] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one* 12, 4 (2017), e0172395.
[29] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), 267–288.
[30] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. 2018. IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2496–2505.
[31] Jihoon Yang and Vasant Honavar. 1998. Feature subset selection using a genetic algorithm. In *Feature extraction, construction and selection*. Springer, 117–136.
[32] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. 2018. Mean Field Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:1802.05438* (2018).
[33] Yiming Yang and Jan O Pedersen. 1997. A comparative study on feature selection in text categorization. In *Icml*, Vol. 97. 412–420.
[34] Lei Yu and Huan Liu. 2003. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*. 856–863.
[35] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep Reinforcement Learning for Page-wise Recommendations. *arXiv preprint arXiv:1805.02343* (2018).