AutoGFS: Automated Group-based Feature Selection via Interactive Reinforcement Learning

Wei Fan* Kunpeng Liu* Hao Liu† Ahmad Hariri* Dejing Dou† Yanjie Fu*‡

Abstract

Feature selection is a fundamental component of data mining, aiming to select optimal feature subsets for downstream task. Recently, an emerging feature selection method called reinforced feature selection applies reinforcement learning into feature selection. Reinforced Feature Selection (RFS) automates feature selection process and can effectively find the optimal subset. Generally, RFS can be categorized into single-agent RFS and multi-agent RFS. Single-agent RFS uses one reinforcement learning agent to select features, but its action space is exponentially-increasing with feature number and can merely obtain local optima. Multi-agent RFS uses multiple agents to select features; this method can achieve global optima, but it needs to optimize as many policy networks as feature number which costs huge computational resources and thus becomes computationally inefficient. This dilemma naturally leads to a research question: How can we synthesize the advantages of single-agent RFS and multi-agent RFS while avoiding their disadvantages? To answer this question, we propose a Group-based Interactive Reinforced Feature Selection (GIRFS) framework. This framework balances single-agent RFS and multi-agent RFS for better feature selection. Specifically, we formulate the feature selection problem into a group-based RFS problem. In this formulation, we first assign the given features into several groups based on feature similarity measurement. Then, we create agents for each group, where each agent decides to select/deselect features in its corresponding group. This design balances the size of action space and number of policy networks and thus makes RFS more effective and efficient. Moreover, to further improve learning efficiency, we propose a hierarchical teacher-like trainer to provide external action advice for agents. This trainer provides advice by intra-group selection and inter-group selection and fuses knowledge from mRMR and decision tree to help agents explore and learn. Finally, we present extensive experiments on real-world datasets to demonstrate the improved performances of our method.

1 Introduction

Feature selection is an important component in data mining and machine learning, which aims to select the optimal feature subset for downstream predictive tasks. Traditionally, feature selection can be categorized into: (i) filter methods (e.g., univariate feature selection [9]), (ii) wrapper methods (e.g., branch and bound algorithms [19]), and (iii) embedded methods (e.g., LASSO [26]). Recently, an emerging kind of methods called reinforced feature selection applies reinforcement learning into feature selection. Reinforced Feature Selection (RFS) automates the selection process, interacts with the downstream task, learns from the agents' actions, and shows improved effectiveness in feature selection.

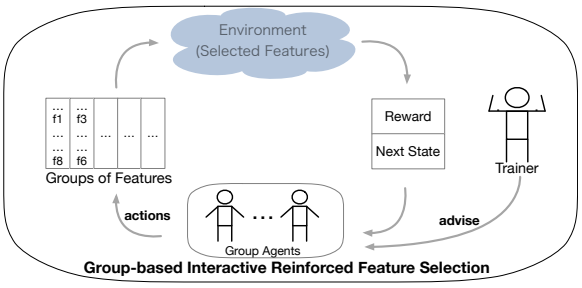


Figure 1: Group-based Interactive Reinforced Feature Selection (GIRFS) framework.

After exploring different kinds of reinforced feature selection methods, we observe RFS can be categorized into two kinds of methods: Single-agent RFS and Multi-agent RFS. Single-agent RFS [13, 8] creates only one reinforcement learning agent to decide the selection of all the features. This design has only one policy network to optimize and needs little computational resources; however, the agent's action space (2^N) is exponentially-increasing with feature number (N) and can merely obtain local optima. Multi-agent RFS [16, 7] creates multiple agents to select features and assigns each feature an agent. This design makes each agent select its corresponding feature and thus controls action space; however, it needs to optimize as many policy networks as feature number, which costs huge computational resources and makes training become computationally inefficient. This dilemma naturally leads us to a research question: How can we synthesize the advantages of single-agent RFS and multi-agent RFS while avoiding their disadvantages?

Motivated by this problem, we aim to design a new reinforcement learning based framework to make reinforced feature selection more effective and efficient. However, several challenges arise towards this goal.

* Universirty of Central Florida, Orlando, US. Email: {weifan, kunpengliu, ahmadgjh}@knights.ucf.edu, {yanjie.fu}@ucf.edu.

† Business Intelligence Lab, Baidu Research, Beijing, China, Email: {liuhao30, doudejing}@baidu.com

‡Contact Author

First, how can we reformulate the feature selection problem with reinforcement learning? In this paper, we formulate the feature selection problem into a new framework, namely Group-based Interactive Reinforced Feature Selection (GIRFS). Figure 1 shows its architecture. In this framework, we first categorize the input features into several groups. Then, different from single-agent RFS or multi-agent RFS, we propose to create one agent for each group; we call them group agents. Each group agent is responsible for its corresponding group by deciding the selection of every feature in this group. This group-based design can be better than the existing work with smaller action space than single-agent RFS [8] and less policy networks to train than multi-agent RFS [16]. Thus, our framework balances single-agent RFS and multi-agent RFS, and can make selection process more efficient while still being able to identify the best feature subset.

Second, how can we group features for better reinforced feature selection? Intuitively, we can randomly select features into groups; however, this naive solution neglects feature-group hierarchy and cannot make full use of the feature-feature correlations. In this paper, we propose to select most similar features into one group. Specifically, we first measure the similarity between feature to feature and then propose an algorithm trying to greedily categorize similar features into one group. In this way, each group including several similar features can reflect specific characteristics of input data matrix. The input feature space can be better represented by these groups, which accordingly makes each group agent better understand the input data and thus more easily to learn the optimal selection decisions.

Third, how can we further improve learning efficiency of the group agents in our framework? Recently, Interactive Reinforcement Learning (IRL) [21], as an emerging technology, has shown its superiority on speeding up the agents' exploration by advice from external teacher-like trainers. Previous work [7] has also proved external trainers can actually improve multi-agent RFS learning. In this paper, considering feature-group hierarchy of our framework, we propose a hierarchical teacher-like trainer for group agents, in order to make them more efficient. Specifically, this hierarchical trainer provides advice by a two-stage process: intra-group selection and inter-group selection. The intra-group selection utilizes mRMR [20] to remove redundant features in groups; the inter-group selection utilizes decision tree to select important features towards labels. By fusing the experience from two selection process with two kinds of external knowledge, the advice becomes diversified and is helpful to guide agent learning.

In summary, in this paper, our contributions are as follows: (i) We propose a new framework called Group-based Interactive Reinforced feature selection (GIRFS), which formulates feature selection into a group-based reinforcement learning selection problem. (ii) We propose to group features by feature-feature similarity, trying to represent the input feature space in different aspects for better reinforced feature selection. (iii) We propose a hierarchical teacher-like trainer to bring up external knowledge to improve agent exploration efficiency. (iv) We conduct extensive experiments which illustrate the improved performances of our methods.

Table 1: Notations.

Notations	Definition
$\lvert \cdot \rvert$	The length of a set
$[x]$	The greatest integer less than x
f_i	The i-th feature
H_i	The i-th group
agt_i	The i-th agent

2 Preliminaries

We first introduce some background definitions. Then, we present the problem formulation of reinforced feature selection. Table 1 shows some notations.

2.1 Backgrounds

Definition 2.1.1 **Reinforcement Learning (RL).** RL is an efficient searching technology, which makes RL agents interact with environments, learn from action rewards, balance exploitation and exploration, and search for long-term optimal decisions [23].

Definition 2.1.2 **Interactive Reinforcement Learning (IRL).** IRL [4, 24] as an emerging technology, can accelerate the exploration of RL. Its intuitive idea is to include external advice from teacher-like trainers in the apprenticeship of exploration. After taking action advice, agents take more advantageous actions and thus learn more efficiently.

2.2 Problem Statement

In this paper, we study the problem of efficiency and effectiveness of reinforced feature selection. Generally, the aim of feature selection is to select a well-performed feature subset for the downstream task. Formally, given a data matrix X including N features denoted by $F = \{f_1, f_2, ..., f_N\}$, feature selection aims to select an optimal feature subset $F^* \subseteq F$, making the downstream task model have a good prediction performance.

Reinforced feature selection applies reinforcement learning into feature selection. It creates reinforcement learning agent to use policy network π to select features. RL automates feature selection by automatically interacting with environment and can select a better subset.

3 Group-based Interactive Reinforced Feature Selection Framework

In this section, we first present the process of how to categorize features into groups; then, we introduce the Group-based Interactive Reinforced feature selection (GIRFS) framework in detail.

3.1 Group Features Based on Feature Similarity Measurement

In this part, we propose to categorize the given features into several different groups for more effective and efficient reinforced feature selection.

Intuitively, we can randomly select features into groups; however, this solution doesn't consider feature-group hierarchy and neglects the relationship of feature with regard to group. To this end, we propose a greedy algorithm to select most similar features into one group. Our assumption is: the input feature space can be represented in different aspects by groups, which makes group agent better understand feature environment and learn optimal decisions. Specifically, each group including several similar features can be regarded as part of representations of the input data matrix X, which can reflect its characteristics in different aspects. Each agent pays attention to the features in its corresponding group and can be more aware of input feature space based on group architecture. Thus, group agents can wisely make decisions in order to maximize the long-term reward. We present the details of how to group features step by step as follows:

Step 1: We first measure the relationship between feature to feature. The measurement is based on the Pearson correlation coefficient [2]. Formally, for any feature f_i and feature f_j in F, the correlation ρ is by:

$$(3.1)\ \rho_{f_i,f_j} = \frac{\mathrm{cov}(f_i, f_j)}{\sigma_{f_i}\sigma_{f_j}} = \frac{E[(f_i - \mu_{f_i})(f_j - \mu_{f_j})]}{\sigma_{f_i}\sigma_{f_j}}$$

where σ is standard deviation, cov is covariance, μ is the average value.

Step 2: We devise a greedy policy to select features into groups. Specifically, considering a set of tuples $R = \{(\rho_{f_i,f_j}, f_i, f_j) \,|\, f_i, f_j \in F\}$, we sort the tuples in R by correlation measurement between features; then, we traverse the sorted tuples and greedily put the feature pair (f_i, f_j) with higher correlation into a group as much as possible. More details are in Algorithm 1.

Step 3: We collect N features into h groups, each of which has K features. After the traversal, for the rest features that are not categorized into groups, we randomly insert them into the groups that still have vacant positions (have less than K features). Finally, we get the h groups of features $\{H_1, H_2, ..., H_h\}$, where i-th group $H_i = \{f_{i_1}, f_{i_2}, ..., f_{i_K}\}$.

Algorithm 1: Greedily Grouping Features with Similarity Measurement.

Input: Set of features $F\{f_1, f_2, ..., f_N\}$
Output: Group set of features $\{H_1, H_2, ..., H_h\}$

initialize group set $\{H_1, H_2, ..., H_h\}$ with $\emptyset$;
initialize relation set R with $\emptyset$;
Add all tuples $(\rho_{f_i,f_j}, f_i, f_j)$ to R where for $i, j \in [1, N]\,\&\, i < j$;
Sort each tuple in R by $abs(\rho_{f_i,f_j})$ in descending;
initialize a vector $flag = (0, 0, ..., 0)_N$ to flag whether features are grouped or not ;
for *item in R* **do**
    $f_i, f_j = item[1], item[2]$;
    if $flag[i] + flag[j] == 0$ **then**
        if $H_k \in \{H_1, H_2, ..., H_h\}$ *is* $\emptyset$ **then**
            Add f_i, f_j to H_k;
            $flag[i] = 1, flag[j] = 1$
        end
    end
    if $flag[i] == 1\,\&\, flag[j] == 0$ **then**
        find H_k where $f_i \in H_k$;
        if $|H_k| < K$ **then**
            Add f_j to H_k, $flag[j] = 1$
        end
    end
    if $flag[i] == 0\,\&\, flag[j] == 1$ **then**
        find H_k where $f_j \in H_k$;
        if $|H_k| < K$ **then**
            Add f_i to H_k, $flag[i] = 1$
        end
    end
end
Randomly **insert** feature f_i into H_k for $i \in [1, N], flag[i] = 0\,\&\, k \in [1, h], H_k < K$;
return $\{H_1, H_2, ..., H_h\}$

3.2 Framework Overview

We introduce some basic components of GIRFS framework and show an overview of this framework.

3.2.1 Group Agents. Different from using single agent to select all the features [8, 11] or assigning each feature a agent [16, 7], we propose to create multiple agents for the selection of each group of features, called group agents. Specifically, for h group of features $\{H_1, H_2, ..., H_h\}$, we create h group agents $\{agt_1, agt_2, ..., agt_h\}$ where each agent decides the selection of features in its corresponding group.

3.2.2 Actions. Each group agent decides to select or deselect every feature in its corresponding group.

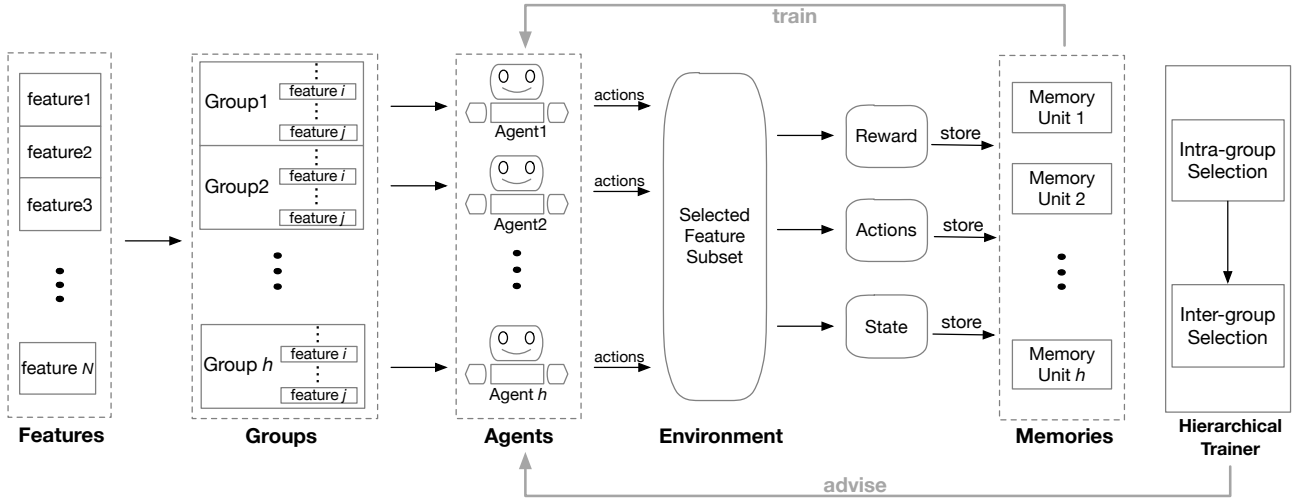


Figure 2: Framework Overview. The features are grouped for reinforced feature selection. Memories are used to train agents; the hierarchical trainer helps to advise group agents for better exploration.

Specifically, for features in i-th group H_i, agent agt_i's actions are to select or deselect all K feature $\{f_{i_1}, f_{i_2}, ..., f_{i_K}\}$; accordingly, the action space is 2^K.

3.2.3 State. The state s in reinforced feature selection is defined as the representation of the environment, which is the selected feature subset in each step of the exploration. Previous work [16] has tried to represent the selected feature subset in many different ways. In this framework, we apply spatial-based Graph Convolutional Network (GCN) [32] into our framework, in order to model the correlation between feature to feature for better representation. Specifically, we first transform selected feature subset $F_s\{f_{p_1}, f_{p_2}, ..., f_{p_q}\}$ into a fully-connected graph where each node represents a feature. To update the feature representation, we apply spatial-based GCN in this complete graph. For i-th feature f_{p_i} in F_s, the updated representation h_{p_i} is by:

$$h_{p_i} = \sigma(\sum_{f_{p_k} \in F_s} W_{p_i,p_k} f_{p_k}) \tag{3.2}$$

where σ is activation function, W_{p_i,p_k} is the weight parameters measured by Pearson correlation coefficient $\rho_{f_{p_i},f_{p_k}}$. Then, the state representation s can be calculated by:

$$s = \frac{1}{|F_s|} \sum_{f_{p_k} \in F_s} h_{p_k} \tag{3.3}$$

3.2.4 Reward. The reward r is to inspire the feature subspace exploration process. We firstly measure the reward based on the predictive performances (e.g., accuracy) of the selected feature subset in the downstream task. Then, we equally assign the reward to the group agents as the feedback of their exploration.

3.2.5 Trainers. We bring up teacher-like trainer ('trainer' for short) concept from interactive reinforcement learning to our framework. In this paper, we propose a hierarchical teacher-like trainer for this framework in order to provide advice to the group agents. More details are in next section.

3.2.6 Framework. Figure 2 shows the overview of our framework. For given features in input data matrix, we first categorize them into different groups; we use a greedy algorithm to make each group include several most similar features. We create multiple reinforcement learning agents for these groups which we call group agents. Different from previous work, each agent decides the selection of all the features in its corresponding group. The proposed hierarchical trainer guides group agents by providing them action advice, in order to make them learn more efficiently. After the agents take actions, they generate a selection subset, which is regarded as the environment of our framework. Then, the reward, actions and state are stored into memory units. The mini-batches sampled from memory units are to train agents' policy networks, in order to maximize the long-term reward based on Bellman Equation [23]:

$$Q(s_i^t, a_i^t|\theta^t) = r_i^t + \gamma \max Q(s_i^{t+1}, a_i^{t+1}|\theta^{t+1}) \tag{3.4}$$

where s is the state, a is the action, r is the reward, θ is the parameter set of Q network and γ is the discount.

4 Interactive Reinforcement Learning with Hierarchical Teacher-like Trainer

To make our group agents learn more efficiently, we aim to leverage the external knowledge to help for better exploration. Teacher-like trainers with prior knowledge can provide action advice to group agents; the advised actions are more advantageous for agents to perform and thus can guide agents to be more aware of the selected feature subspace. An intuitive idea to help reinforced feature selection agent is to utilize classic feature selection as external knowledge for guidance. However, single selection method always identifies similar feature subset and provides similar advice; this jeopardize agent learning [7].

To this end, we propose a hierarchical teacher-like trainer, which fuses knowledge from mRMR [20] and decision tree and is fit for our group-based reinforced feature selection framework. As Figure 3 shows, this hierarchical trainer provides advice by a two-stage process: intra-group feature selection and inter-group feature selection. This two-stage selection can provide more diversified advice to guide agent learning. We introduce this advising process along this line.

4.1 Intra-group Selection for Advice

The hierarchical trainer firstly selects intra-group features for advice. Features in one group are most similar; thus, the selected feature subset can be more redundant with too many similar features. The proposed hierarchical trainer first considers relationship among features in a group and select or deselect them. Specifically, we propose to apply Max-Relevance and Min-Redundancy (mRMR) selection to finish intra-group selection. Specifically, mRMR tries to maximize relevance $D(F, y)$, formally by:

$$D(F, y) = \frac{1}{|F|} \sum_{f_i \in F} I(f_i, y) \tag{4.5}$$

where I is mutual information, y is label for classification; simultaneously, it tries to minimize redundancy $R(F)$, formally by:

$$R(F) = \frac{1}{|F|^2} \sum_{f_i, f_j \in F} I(f_i, f_j) \tag{4.6}$$

where I is the mutual information. The joint consideration tries to maximize $\Phi(D, R)$ where $\Phi(D, R) = D - R$. Through mRMR selection, we aim to remove the redundant features in groups. Each group is represented by few features for the inter-group selection, which is more beneficial to provide better advice. The intra-group selection for advice is detailed by:

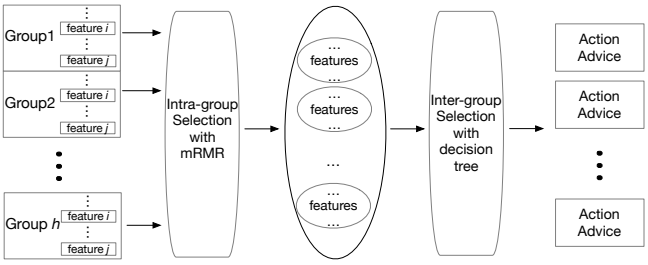


Figure 3: General process of providing advice by the hierarchical teacher-like trainer.

Step 1: For i-the group, we select k_i features with mRMR selection, where k_i is a random integer between 1 to K. We denote the selected results by $F_{mrmr} = \{mrmr(H_1, k_1), mrmr(H_2, k_2), ..., mrmr(H_h, k_h)\}$.

Step 2: For features in F_{mrmr}, they are the selected results of the intra-group selection, which are taken as input to second stage: inter-group selection.

4.2 Inter-group Selection for Advice

After the intra-group selection, in order to generate more diversified advice, this hierarchical teacher-like trainer further selects features from the intra-group selection results; we call this selection stage as inter-group selection. Decision Tree is widely used in data mining and machine learning predictive tasks. The hierarchy of decision tree can reflect the feature importance of the decision features towards prediction labels; considering this, we propose to apply the decision tree selection with feature importance into the inter-group selection. The inter-group selection for advice process is detailed by:

Step 1: For the intra-group selection results, we first train a decision tree T on them F_{mrmr}. Then, we get the feature importance of each feature in F_{mrmr}, denoted by $\{imp_1, imp_2, ..., imp_d\}$ where $d = k_1 + k_2 + \cdots + k_h$ is the number of elements in F_{mrmr}.

Step 2: We utilize the feature importance to generate the selection probability of each feature $\{p_1, p_2, ..., p_d\}$ to select features. Note that the selection probability changes with the features (inter-group selection results). Formally, the i-th feature selection probability is measured by:

$$p_i = \begin{cases} 1 & imp_i > \frac{1}{N} \\ N\, imp_i & imp_i \leq \frac{1}{N} \end{cases} \tag{4.7}$$

Step 3: We select features based on the selection probability $\{p_1, p_2, ..., p_d\}$. The selected results regarded as the action advice for group agents: if a feature is selected by trainer, agents are advised to select it; otherwise, agents are advised to deselect it. After the two-stage advising process, this hierarchical trainer gives more diversified advice to guide agents' exploration.

5 Experiment

In this section, we conduct extensive experiments to evaluate the feature selection performance in terms of effectiveness and efficiency.

5.1 Data Description

Musk Dataset describes a set of 102 molecules of which 39 are judged by human experts to be musks and the remaining 63 molecules are judged to be non-musks. It includes 6598 samples and the aim is to classify the label 'musk' and 'non-musk' [5].

Spambase Dataset is a collection of spam emails which came from the postmaster and individuals who had filed spam. It includes 4601 samples and 57 different features, indicating whether a particular word or character was frequently occurring in the email [5].

Insurance Company Benchmark (ICB) Dataset contains information about customers, which consists of 86 variables and 5000 samples and includes product usage data and socio-demographic data derived from zip area codes [27].

Forest Cover (FC) Dataset is a publicly available dataset from Kaggle [1]. This dataset includes 15120 samples and 54 different characteristics of wilderness areas, which are used to predict forest cover type. The class labels (forest cover type) are from 1 to 7.

Table 2: Description of the dataset.

	Musk	Spam	ICB	FC
Features	168	57	86	54
Samples	6598	4601	5000	15120

5.2 Evaluation Metrics

In feature selection, we aim to find the optimal feature subset to perform well in the downstream task. We use the following metrics to evaluate the performance.

Accuracy is the ratio of the number of correct predictions to the number of all predictions. Formally, the accuracy is given by $Acc = \frac{TP+TN}{TP+TN+FP+FN}$, where TP, TN, FP, FN are true positive, true negative, false positive and false negative for all classes.

F1-score considers both precision and recall collaboratively by taking their harmonic mean. Formally, F-measure is given by $\frac{2*P*R}{P+R}$, where P and R are precision and recall respectively.

5.3 Baseline Algorithms

We compare the feature selection performance of our proposed method with the following several baselines:

(1) K-Best Feature Selection. This algorithm [30] ranks features by the ranking scores with the labels; it selects the top k highest scoring features. In the experiments, we set k equals to half of the number of input features.

(2) LASSO. LASSO [26] conducts feature selection and shrinkage via $l1$ penalty. In this method, features whose coefficients are 0 will be dropped. The hyper parameter in LASSO is its regularization weight λ which is set to 1.0 in the experiments.

(3) DT-RFE (Decision Tree Recursive Feature Elimination). RFE [10] selects features by selecting smaller and smaller feature subsets until finding the subset with certain number of features. DT-RFE trains decision tree on the features to get feature importance; then it recursively deselects the least important features. In the experiments, we set the selected feature number to half of the feature space.

(4) RF-RFE (Random Forest Recursive Feature Elimination). It [10] trains random forest classifier to get feature importance; then it recursively deselects the least important features. In the experiments, we set the selected feature number half of the feature space.

(5) Single-agent Reinforced Feature Selection. Single-agent RFS [13] is a model-based reinforcement learning method, where the agent learns a model of the environment, represented as a dynamic Bayesian network that describes rewards and state transitions.

(6) Multi-agent Reinforced Feature Selection. Multi-agent RFS [16] is a reinforcement learning based feature selection method, which creates multiple agents to select features. To compare fairly, its downstream task is set the same as our framework.

5.4 Implementations

In the experiments, the downstream task is set to decision tree with the default parameters in scikit-learn library [2]. We randomly split the data into train data (70%) and test data (30%). In the experience replay [15, 18], for agent agt_i at step t, we store a tuple $\{s_i^t, r_i^t, s_i^{t+1}, a_i^t\}$ to the memory unit. The memory size is set to 500. In the reinforcement learning, the discount factor γ is set to 0.9, and we use ϵ-greedy exploration with ϵ equal to 0.9. The policy networks are set as two linear layers of 512 middle states with ReLU as activation function. For batch training, we set batch size to 16 and use Adam Optimizer with learning rate of 0.01. All the evaluations are performed on Intel E5-1680 3.40GHz CPU in a x64 machine whose RAM is 128GB and the operation system is CentOS 7.4.

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

[2] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Table 3: Performance of different feature selection methods on four real-world datasets.

Model	Musk		Spam		ICB		FC	
	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
KBest	0.9575	0.8609	0.9094	0.8874	0.8866	0.8838	0.7667	0.7670
LASSO	0.9646	0.8863	0.8667.	0.8336	0.9072	0.9132	0.7590	0.7596
DT-RFE	0.9566	0.8581	0.9029	0.8805	0.8843	0.8819	0.7669	0.7678
RF-RFE	0.9591	0.8666	0.9058	0.8851	0.8923	0.8911	0.7682	0.7701
Single-agent RFS	0.9632	0.8788	0.9109	0.8818	0.9112	0.8907	0.7722	0.7731
Multi-agent RFS	0.9712	0.9070	0.9174	0.8963	0.9192	0.9149	0.7767	0.7772
GIRFS (Ours)	**0.9772**	**0.9251**	**0.9254**	**0.9074**	**0.9232**	**0.9291**	**0.7870**	**0.7884**

5.5 Overall Performances

We evaluate the feature selection performance in the downstream task on four different real-world datasets. Table 3 shows the performance comparisons in terms of accuracy and F1-score. Compared with other feature selection methods, we can oberve that our proposed Group-based Interactive Reinforced Feature Selection (GIRFS) achieves the best predictive performance in the downstream task, showing advantages of our method. From the table, while our method has highest scores, generally the reinforced feature selection methods including Single-agent RFS, Multi-agent RFS and GIRFS always outperform traditional selection methods. This signifies reinforcement learning makes feature selection more effective through automatic interaction with the downstream task.

5.6 Study of Groups in Reinforced Feature Selection

In this section, we aim to study the impacts of groups in GIRFS framework. Specifically, the number of groups h equals to $[N/K]$ where N is the number of all the features and K is number of features in one group. We consider the methods with different group size K; we denote the GIRFS framework with K features in one group as GIRFS-K. Figure 4 shows the performance comparison on four datasets across different Ks. First, we can easily observe that feature selection performance in overall decreases with K increasing. A potential interpretation is that the action space of agents is 2^K which increases exponentially with K. For example, when K is greater than 7, the action choice is larger than 128. This makes it difficult for policy networks to find the optimal decisions, because reinforcement learning has to explore much more steps to learn the optimal policies. When K decreases, the performances are better but the models need more time for training; thus it is a trade-off between effectiveness and efficiency.

5.7 Study of Hierarchical Teacher-like Trainer

In this section, we study the impacts of the proposed hierarchical teacher-like trainer on exploration efficiency.

Table 4: Training time comparison on different datasets

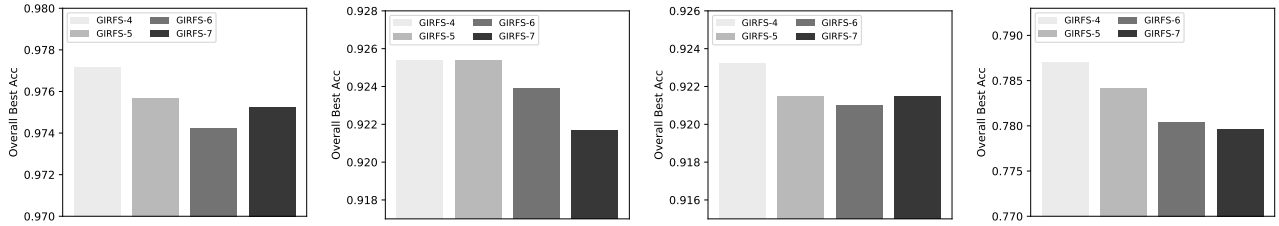
Models	Musk	Spam	ICB	FC
Multi-agent RFS	7h54m	2h49m	4h07m	3h49m
GIRFS-4	5h50m	1h52m	2h54m	2h8m
GIRFS-5	4h56m	1h34m	2h27m	1h46m
GIRFS-6	3h45m	1h19m	2h08m	1h31m
GIRFS-7	3h11m	1h16m	1h53m	1h22m

We consider three different variant methods towards the two-stage process in the hierarchical trainer: (i) **Non-trainer**: this variant method removes the hierarchical trainer from GIRFS framework and let agents explore without external knowledge. (ii) **Intra-group**: this variant method only considers intra-group selection for advice and removes inter-group selection stage in the hierarchical trainer. (iii) **Intra-group+Inter-group**: this method fully considers the two-stage process of hierarchical trainer and uses inter-group selection and intra-group selection to provide advice.

Figure 5 shows the exploration efficiency comparison of these methods. We can observe that both intra-group and inter-group methods can achieve higher accuracy than Non-trainer method in the beginning of the exploration. This signifies the action advice from external trainer can really help reinforcement learning agents to learn better and identify optimal subsets sooner. Also, when two stages of selection for advice are adopted, the performance is further improved, which demonstrates that fusing the knowledge coming from two stages can better improve agent's learning efficiency.

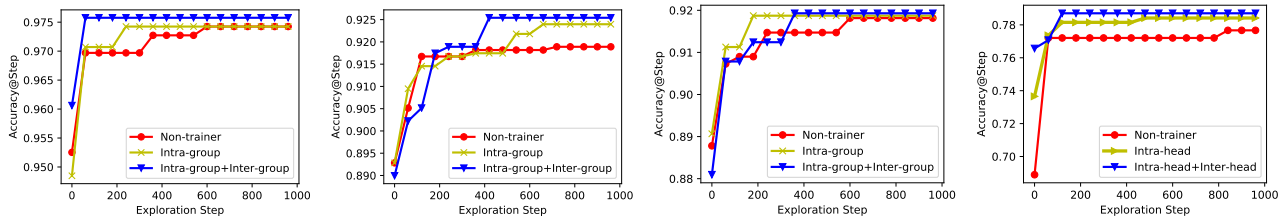
5.8 Study of Computational Efficiency

In this section, we study the computational efficiency of our proposed GIRFS framework. Specifically, we record the training time of different methods while controlling other factors. All the agents explore 3000 steps and have the same parameters in policy networks. Table 4 shows the training time comparisons, where 'h' and 'm' stands for 'hour' and 'minute'. From the table, we find out that when K increases, the number of policy networks decreases so that it needs less time to train the reinforcement learning agents. Moreover, Multi-agent RFS costs about double time of GIRFS-6 or GIRFS-



(a) Musk Dataset (b) Spam Dataset (c) ICB Dataset (d) FC Dataset

Figure 4: Accuracy comparison of GIRFS framework with different group size.



(a) Musk Dataset (b) Spam Dataset (c) ICB Dataset (d) FC Dataset

Figure 5: Exploration efficiency comparison of variant methods.

7 for training. This signifies our methods can make reinforced feature selection more efficiently.

6 Related Work

Machine learning in large scale needs feature selection [28]. Traditional feature selection can be grouped into: (1) Filter methods calculate relevance scores, rank features and then select top-ranking ones (e.g., univariate feature selection [9, 30]). (2) Wrapper methods make use of predictors, considering the prediction performance as objective function. (e.g., branch and bound algorithms [12]). (3) Embedded methods take advantages of predictors, incorporation feature selection as part of predictors.(e.g., LASSO [26], decision tree [22]).

Reinforced feature selection applies reinforcement learning into feature selection. Some existing studies creates a single agent to make decisions [8, 13, 31] whose action space is 2^N which is too large. Another kind of existing studies create multi-agents to make decisions and every agent determines the selection and deselection of its corresponding feature [16, 7, 6].

Reinforcement Learning [23] is a good method to address optimal decision-making problem by developing action strategies, which has different applications in feature selection [8, 16], urban computing [29], robotics [17, 4]. In reinforcement learning, the next action is from the highest state-action pair. An intuitive idea to speed up the learning process is to include external advice in the apprenticeship [4]. In Interactive Reinforcement Learning (IRL), an action is interactively encouraged by a trainer with prior knowledge [11, 25]. Using a trainer to directly advise on future actions is known as policy shaping [3, 1]. For advice, they could come from humans and robots [14] and also from previously trained artificial agent [24].

7 Conclusion

In this paper, we study the effectiveness and efficiency of the reinforced feature selection. We propose a new reinforcement learning based framework for feature selection, namely Group-based Interactive Reinforced Feature Selection (GIRFS) framework. This framework collects features into different groups and creates group agents for the features, which can better select the optimal subsets. Moreover, inspired by interactive reinforcement learning, we propose a hierarchical teacher-like trainer, which includes a two-stage advice process with external knowledge from mRMR and decision tree. Finally, we conduct extensive experiments to demonstrate the improved performances.

8 Acknowledgment

This research was partially supported by the National Science Foundation (NSF) via the grant numbers: 1755946, I2040950, 2006889.

References

[1] O. Amir, E. Kamar, A. Kolobov, and B. Grosz, *Interactive teaching strategies for agent training*, (2016).

[2] J. Benesty, J. Chen, Y. Huang, and I. Cohen, *Pearson correlation coefficient*, in Noise reduction in speech processing, Springer, 2009, pp. 1–4.

[3] T. Cederborg, I. Grover, C. L. Isbell, and A. L. Thomaz, *Policy shaping with human teachers*, in Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[4] F. Cruz, S. Magg, Y. Nagai, and S. Wermter, *Improving interactive reinforcement learning: What makes a good teacher?*, Connection Science, 30 (2018), pp. 306–325.

[5] D. Dua and C. Graff, *UCI machine learning repository*, 2017.

[6] W. Fan, K. Liu, H. Liu, Y. Ge, H. Xiong, and Y. Fu, *Interactive reinforcement learning for feature selection with decision tree in the loop*, arXiv preprint arXiv:2010.02506, (2020).

[7] W. Fan, K. Liu, H. Liu, P. Wang, Y. Ge, and Y. Fu, *Autofs: Automated feature selection via diversity-aware interactive reinforcement learning*, arXiv preprint arXiv:2008.12001, (2020).

[8] S. M. H. Fard, A. Hamzeh, and S. Hashemi, *Using reinforcement learning to find an optimal set of features*, Computers & Mathematics with Applications, 66 (2013), pp. 1892–1904.

[9] G. Forman, *An extensive empirical study of feature selection metrics for text classification*, Journal of machine learning research, 3 (2003), pp. 1289–1305.

[10] P. M. Granitto, C. Furlanello, F. Biasioli, and F. Gasperi, *Recursive feature elimination with random forest for ptr-ms analysis of agroindustrial products*, Chemometrics and Intelligent Laboratory Systems, 83 (2006), pp. 83–90.

[11] W. B. Knox, P. Stone, and C. Breazeal, *Teaching agents with human feedback: a demonstration of the tamer framework*, in Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion, 2013, pp. 65–66.

[12] R. Kohavi, G. H. John, et al., *Wrappers for feature subset selection*, Artificial intelligence, 97 (1997), pp. 273–324.

[13] M. Kroon and S. Whiteson, *Automatic feature selection for model-based reinforcement learning in factored mdps*, in 2009 International Conference on Machine Learning and Applications, IEEE, 2009, pp. 324–330.

[14] L. J. Lin, *Programming robots using reinforcement learning and teaching.*, in AAAI, 1991, pp. 781–786.

[15] L.-J. Lin, *Self-improving reactive agents based on reinforcement learning, planning and teaching*, Machine learning, 8 (1992), pp. 293–321.

[16] K. Liu, Y. Fu, P. Wang, L. Wu, R. Bo, and X. Li, *Automating feature subspace exploration via multi-agent reinforcement learning*, in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 207–215.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, *Playing atari with deep reinforcement learning*, arXiv preprint arXiv:1312.5602, (2013).

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., *Human-level control through deep reinforcement learning*, Nature, 518 (2015), pp. 529–533.

[19] P. M. Narendra and K. Fukunaga, *A branch and bound algorithm for feature subset selection*, IEEE Transactions on computers, (1977), pp. 917–922.

[20] H. Peng, F. Long, and C. Ding, *Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy*, IEEE Transactions on pattern analysis and machine intelligence, 27 (2005), pp. 1226–1238.

[21] H. B. Suay and S. Chernova, *Effect of human guidance and state space size on interactive reinforcement learning*, in 2011 Ro-Man, IEEE, 2011, pp. 1–6.

[22] 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*, Mechanical systems and signal processing, 21 (2007), pp. 930–942.

[23] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[24] M. E. Taylor, N. Carboni, A. Fachantidis, I. Vlahavas, and L. Torrey, *Reinforcement learning agents providing advice in complex video games*, Connection Science, 26 (2014), pp. 45–63.

[25] A. L. Thomaz, C. Breazeal, et al., *Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance*, in Aaai, vol. 6, Boston, MA, 2006, pp. 1000–1005.

[26] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society: Series B (Methodological), 58 (1996), pp. 267–288.

[27] P. Van Der Putten and M. van Someren, *Coil challenge 2000: The insurance company case*, tech. rep., Technical Report 2000–09, Leiden Institute of Advanced Computer Science . . . , 2000.

[28] M. Wang, W. Fu, X. He, S. Hao, and X. Wu, *A survey on large-scale machine learning*, IEEE Transactions on Knowledge and Data Engineering, (2020).

[29] S. Wang, J. Cao, and P. Yu, *Deep learning for spatio-temporal data mining: A survey*, IEEE Transactions on Knowledge and Data Engineering, (2020).

[30] Y. Yang and J. O. Pedersen, *A comparative study on feature selection in text categorization*, in Icml, vol. 97, Nashville, TN, USA, 1997, p. 35.

[31] X. Zhao, K. Liu, W. Fan, L. Jiang, X. Zhao, M. Yin, and Y. Fu, *Simplifying reinforced feature selection via restructured choice strategy of single agent*, arXiv preprint arXiv:2009.09230, (2020).

[32] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, *Graph neural networks: A review of methods and applications*, arXiv preprint arXiv:1812.08434, (2018).