

Efficient Reinforced Feature Selection via Early Stopping Traverse Strategy

Kunpeng Liu*, Pengfei Wang†, Dongjie Wang*, Wan Du‡, Dapeng Oliver Wu§, Yanjie Fu*

* University of Central Florida, Orlando, United States

† DAMO Academy, Alibaba Group, China

‡ University of California, Merced, United States

§ University of Florida, Gainesville, United States

*{kunpengliu, wangdongjie}@knights.ucf.edu, yanjie.fu@ucf.edu

†{wpf2106}@gmail.com, ‡{wdu3}@ucmerced.edu, §{dpwu}@ufl.edu

arXiv:2109.14180v2 [cs.LG] 12 Oct 2021

Abstract—In this paper, we propose a single-agent Monte Carlo based reinforced feature selection (MCRFS) method, as well as two efficiency improvement strategies, i.e., early stopping (ES) strategy and reward-level interactive (RI) strategy. Feature selection is one of the most important technologies in data preprocessing, aiming to find the optimal feature subset for a given downstream machine learning task. Enormous research has been done to improve its effectiveness and efficiency. Recently, the multi-agent reinforced feature selection (MARFS) has achieved great success in improving the performance of feature selection. However, MARFS suffers from the heavy burden of computational cost, which greatly limits its application in real-world scenarios. In this paper, we propose an efficient reinforcement feature selection method, which uses one agent to traverse the whole feature set, and decides to select or not select each feature one by one. Specifically, we first develop one behavior policy and use it to traverse the feature set and generate training data. And then, we evaluate the target policy based on the training data and improve the target policy by Bellman equation. Besides, we conduct the importance sampling in an incremental way, and propose an early stopping strategy to improve the training efficiency by the removal of skew data. In the early stopping strategy, the behavior policy stops traversing with a probability inversely proportional to the importance sampling weight. In addition, we propose a reward-level interactive strategy to improve the training efficiency via reward-level external advice. Finally, we design extensive experiments on real-world data to demonstrate the superiority of the proposed method.

search strategy that collaborates with predictive tasks (e.g., evolutionary algorithms [5], [6], branch and bound algorithms [7], [8]); (iii) embedded methods, in which feature selection is part of the optimization objective of predictive tasks (e.g., LASSO [9], decision tree [10]). 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 directly search a very large feature space of 2^N feature subspace candidates, where N is the number of features. Embedded methods are subject to the strong structured assumptions of predictive models, i.e., in LASSO, the non-zero weighted features are considered to be important.

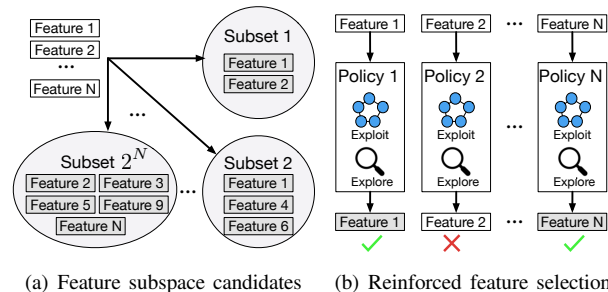


Fig. 1: Reinforced feature selection explores the feature subspace by assigning each feature one agent, and the agent’s policy decides the selection of its corresponding feature.

I. INTRODUCTION

In general data mining and machine learning pipelines, before proceeding with machine learning tasks, people need to preprocess the data first. Preprocessing technologies include data cleaning, data transformation and feature engineering. As one of the most important feature engineering technique, feature selection aims to select the optimal feature subset from the original feature set for the downstream task. Traditional feature selection methods can be categorized into three families: (i) filter methods, in which features are ranked by a specific score (e.g., univariate feature selection [1], [2], correlation based feature selection [3], [4]); (ii) wrapper methods, in which optimal feature subset is identified by a

Recently, reinforcement learning has been incorporated with feature selection and produces an emerging feature selection method, called reinforced feature selection [11], [12]. In the reinforced feature selection, there are multiple agents to control the selection of features, one agent for one feature. All agents cooperate to generate the optimal feature set. It has been proved to be superior to traditional feature selection methods due to its powerful global search ability. However, each agent adapts a neural network as its policy. Since the agent number equals the feature number (N agents for N features), when the feature set is extremely large, we need to train a large number of neural networks, which is computationally high and

• Pengfei Wang is a co-first author.
• Yanjie Fu is the corresponding author.

not applicable for large-scale datasets. Our research question is: Can we design a more practical and efficient method to address the feature selection problem while preserving the effectiveness of reinforced feature selection? To answer these questions, there are three challenges.

The first challenge is to reformulate the feature selection problem with smaller number of agents. Intuitively, we can define the action of the agent as the selected feature subset. For a given feature set, we input it to the agent’s policy and the agent can directly output the optimal subset. However, the feature subset space is as large as 2^N , where N is the feature number. When the dataset is large, the action space is too large for the agent to explore directly. To tackle this problem, we design a traverse strategy, where one single agent visit each feature one by one to decide its selection (to select or deselect). After traversing all the feature set, we can obtain the selected feature subset. We adapt the off-policy Monte Carlo method to our framework. In the implementation, we design two policies, i.e., one behavior policy and one target policy. The behavior policy is to generate the training data and the target policy is to generate the final feature subset. In each training iteration, we use the behavior policy to traverse the feature set, and generates one training episode. The training episode consists of a series of training samples, each of which contains the state, the action and the reward. Similar with [11], we regard the selected feature subset as the environment, and its representation as the state. The action 1/0 denotes selection/deselection, and the reward is composed of predictive accuracy, feature subset relevance and feature subset redundancy. Using the training episode, we evaluate the target policy by calculating its Q value with importance sampling, and improve it by the Bellman equation. After more and more iterations, the target policy becomes better and better. After the training is done, we use the target policy to traverse the feature set and can derive the optimal feature subset. Besides, the behavior policy is supposed to cover the target policy as much as possible so as to generate more high-quality training data, and should introduce randomness to enable exploration [13]. We design an ϵ -greedy behavior policy, to better balance the coverage and the diversity.

The second challenge is to improve the training efficiency of the proposed traverse strategy. In this paper, we improve the efficiency from two aspects. One improvement is to conduct the importance sampling in an incremental way, which saves repeated calculations between samples. In the off-policy Monte Carlo method, since the reward comes from the behavior policy, when we use it to evaluate the target policy, we need to multiply it by an importance sampling weight. We decompose the sampling weight into an incremental format, where the calculation of the sampling weight can directly use the result of previous calculations. The other improvement is to propose an early stopping criteria to assure the quality of training samples as well as stopping the meaningless traverse by behavior policy. In Monte Carlo method, if the behavior policy is too far away from the target policy, the samples from the behavior policy are considered harmful to the evaluation

of the target policy. As the traverse method is continuous and the importance sampling weight calculation depends on the previous result, once the sample at time t is skew, the following samples are skew. We propose a stopping criteria based on the importance sampling weight, and re-calculate a more appropriate weight to make the samples from the behavior policy more close to the target policy.

The third challenge is how to improve the training efficiency by external advice. In classic interactive reinforcement learning, the only source of reward is from the environment, and the advisor does have access to the reward function. However, in many cases, the advisor can not give direct advice on action, but can evaluate the state-action pair. In this paper, we define a utility function \mathcal{U} which can evaluate state-action pair and provide feedback to the agent just like the environment reward does. When integrating the advisor utility \mathcal{U} with the environment reward \mathcal{R} to a more guiding reward \mathcal{R}' , we should not change the optimal policy, namely the optimal policy guided by \mathcal{R}' should be identical to the optimal policy guided by \mathcal{R} . In this paper, we propose a state-based reward integration strategy, which leads to a more inspiring integrated reward as well as preserving the optimal policy.

To summarize, the contributions of this paper are: (1) We reformulate the reinforced feature selection into a single-agent framework by proposing a traverse strategy; (2) We design an off-policy Monte Carlo method to implement the proposed framework; (3) We propose an early stopping criteria to improve the training efficiency. (4) We propose a reward-level interactive strategy to improve the training efficiency. (5) We design extensive experiments to reveal the superiority of the proposed method.

TABLE I: Commonly Used Notations.

Notations	Definition
s_t, a_t	state at time t and action at time t
s^i, a^j	the i -th state and the j -th action
\mathcal{S}	state space defined as $\{s^i i < inf\}$
\mathcal{A}	action space defined as $\{a^j j \in [1, N]\}$
γ	discount factor in range $[0, 1]$
$\mathcal{P}(s_t, a_t, s_{t+1})$	transition probability
$\pi(s)/\pi^*(s)$	policy/optimal policy
\mathcal{M}	Markov decision process (MDP) defined as $\{\mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma, \mathcal{P}\}$
\mathcal{U}	utility function from advisor’s perspective
\mathcal{F}	feature space (set) defined as $\{f^k k \in [0, M]\}$

II. PRELIMINARIES

We first introduce some preliminary knowledge about the Markov decision process(MDP) and the Monte Carlo method to solve MDP, then we give a brief description of multi-agent reinforced feature selection.

A. Markov Decision Process

Markov decision process (MDP) is defined by a tuple $\mathbf{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma, \mathcal{P}\}$, where state space \mathcal{S} is finite, action space \mathcal{A} is pre-defined, reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a mapping function from state-action pair to a scalar, $\gamma \in [0, 1]$ is a discount factor and $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the transition

probability from state-action pair to the next state. In this paper, we study the most popular case when the environment is deterministic and thus $\mathcal{P} \equiv 1$. We use superscripts to discriminate different episode, and use subscripts to denote the time step inside the episode, e.g., s_t^i , a_t^i denote the state and action at time t in the i -th episode.

B. Monte Carlo for Solving MDP

Monte Carlo method can take samples from the MDP to evaluate and improve its policy. Specifically, at the i -th iteration, with a behavior (sampling) policy b^i , we can derive an episode $\mathbf{x}^i = \{x_1^i, x_2^i, \dots, x_t^i, \dots, x_N^i\}$, where $x_t^i = (s_t^i, a_t^i, r_t^i)$ is a sample consisting state, action and reward. With the episode, we can evaluate the Q value $Q_{\pi^i}(s_t^i, a_t^i)$ over our policy (detailed in Section 1), and improve it by Bellman optimality:

$$\pi^{i+1}(s) = \operatorname{argmax}_a Q_{\pi^i}(s, a) \quad (1)$$

With the evaluation-improvement process going on, the policy π becomes better and better, and can finally converge to the optimal policy. The general process is:

$$\pi^0 \xrightarrow{E} Q_{\pi^0} \xrightarrow{I} \pi^1 \xrightarrow{E} Q_{\pi^1} \xrightarrow{I} \dots \xrightarrow{E} Q_{\pi^M} \xrightarrow{I} \pi^M \quad (2)$$

where \xrightarrow{E} denotes the policy evaluation and \xrightarrow{I} denotes the policy improvement. After M iterations, we can achieve an optimal policy. As Equation 2 shows, the policy evaluation and improvement need many iterations, and each iteration needs one episode \mathbf{x} (\mathbf{x}^i for the i -th iteration) consisting N samples.

C. Multi-Agent Reinforced Feature Selection

Feature selection aims to find an optimal feature subset \mathcal{F}' from the original feature set \mathcal{F} for a downstream machine learning task \mathcal{M} . Recently, the emerging multi-agent reinforced feature selection (MARFS) method [11] formulates the feature selection problem into a multi-agent reinforcement learning task, in order to automate the selection process. As Figure 2 shows, in the MARFS method, each feature is assigned to a feature agent, and the action of feature agent decides to select/deselect its corresponding feature. It should be noted that the agents simultaneously select features, meaning that there is only one time step inside an iteration, and thus we omit the subscript here. At the i -th iteration, all agents cooperate to select a feature subset \mathcal{F}^i . The next state s^{i+1} is derived by the representation of selected feature subset \mathcal{F}^i :

$$s^{i+1} = \operatorname{represent}(\mathcal{F}^i) \quad (3)$$

where \mathcal{F}^i is the selected feature subset at time t . *represent* is a representation learning algorithm which converts the dynamically changing \mathcal{F}_t^i into a fixed-length state vector s^{i+1} . The *represent* method can be meta descriptive statistics, autoencoder based deep representation and dynamic-graph based GCN in [11]. The reward r^i is an evaluation of the selected feature subset \mathcal{F}^i :

$$r^i = \operatorname{eval}(\mathcal{F}^i) \quad (4)$$

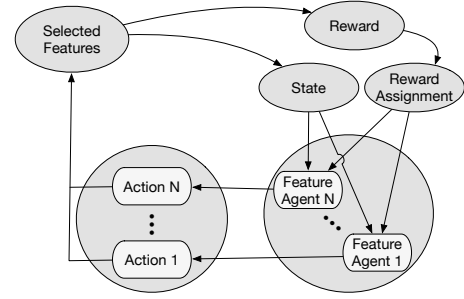


Fig. 2: Multi-agent Reinforced feature selection. Each feature is controlled by one feature agent.

where *eval* is evaluations of \mathcal{F}^i , which can be a supervised metric with the machine learning task \mathcal{F} taking \mathcal{F}^i as input, unsupervised metrics of \mathcal{F}^i , or the combination of supervised and unsupervised metrics in [11]. The reward is assigned to each of the feature agent to train their policies. With more and more steps' exploration and exploitation, the policies become more and more smart, and consequently they can find better and better feature subsets.

III. PROPOSED METHOD

In this section, we first propose a single-agent Monte Carlo based reinforced feature selection method. And then, we propose an episode filtering method to improve the sampling efficiency of the Monte Carlo method. In addition, we apply the episode filtering Monte Carlo method to the reinforced feature selection scenario. Finally, we design a reward shaping strategy to improve the training efficiency.

A. Monte Carlo Based Reinforced Feature Selection

The MARFS method has proved its effectiveness, however, the multi-agent strategy greatly increases the computational burden and hardware cost. Here, we propose a single-agent traverse strategy and use Monte Carlo method as the reinforcement learning algorithm.

1) *Traverse strategy*: As Figure 3 shows, rather than using N agents to select their corresponding features in the multi-agent strategy, we design one agent to traverse all features one at a time.

In the i -th episode, beginning from time $t = 1$, the behavior policy b^i firstly decides the selection decision (select or not select) for feature 1, and then, at time $t = 2$, b^i decides the selection decision for feature 2. With time going on, the features are traversed one by one, and the selected features forms a selected feature subset \mathcal{F}_t^i . Meanwhile, this process also generates an episode $\mathbf{x}_N^i = \{x_1^i, x_2^i, \dots, x_t^i, \dots, x_N^i\}$, where $x_t^i = (s_t^i, a_t^i, r_t^i)$ is a tuple of state, action and reward. The action $a_t^i = 1/0$ is the selection/deselection decision of the t -th feature, the next state s_{t+1}^i is derived by $\operatorname{represent}(\mathcal{F}_t^i)$ and the reward r_t^i is derived by $\operatorname{eval}(\mathcal{F}_t^i)$.

2) *Monte Carlo Method for Reinforced Feature Selection*: With the episode generated by the behavior policy b^i , we can evaluate our target policy π^i and improve π^i . Both the behavior

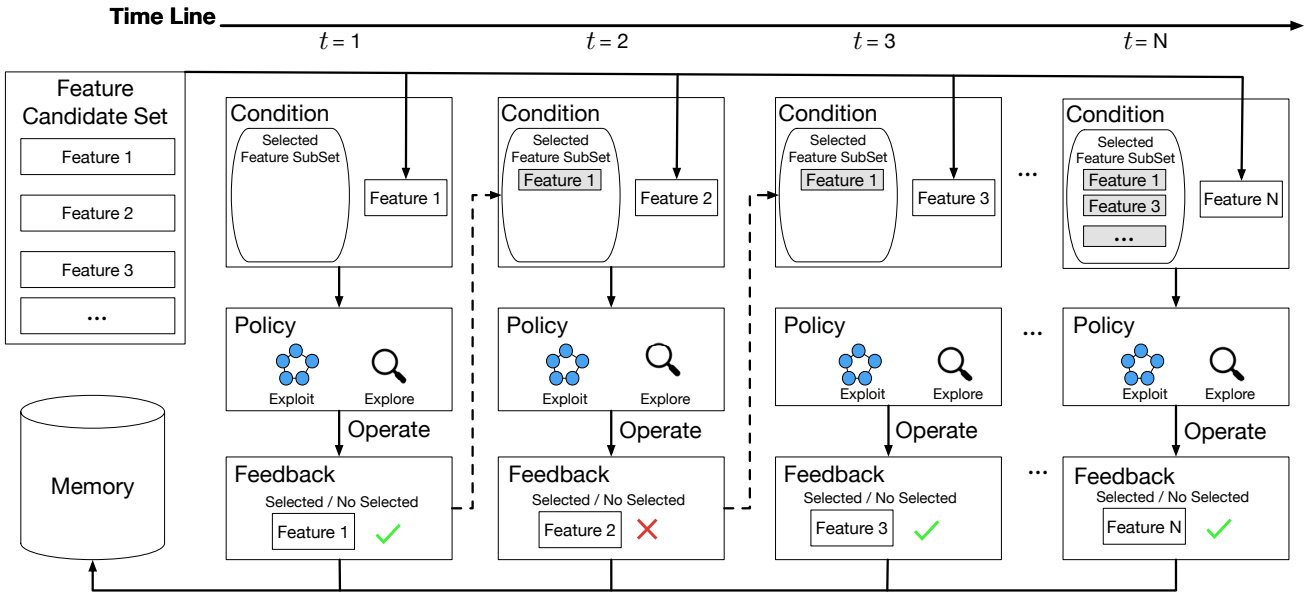


Fig. 3: Single-agent Reinforced feature selection with traverse strategy. At each step, the agent transverses features one by one to decide their selection. The traverse data are stored in the memory to form a training episode.

policy b^i and the target policy π^i provide the probability of taking action a given a specific state s .

Specifically, We generate an episode \mathbf{x}_N^i by b^i . Then, we calculate the accumulated reward by:

$$G^i(s_t^i, a_t^i) = \sum_{j=0}^t \gamma^{(t-j)} \cdot r_j^i \quad (5)$$

where $0 \leq \gamma \leq 1$ is a discount factor.

As the state space is extremely large, we use a neural network $Q(s, a)$ to approximate $G(s, a)$.

The target policy π is different from the behavior policy b , and the reward comes from samples derived from policy b , therefore the accumulated reward of π should be calculated by multiplying an importance sampling weight:

$$\rho_t^i = \frac{\prod_{j=0}^t \pi^i(a_j^i | s_j^i)}{\prod_{j=0}^t b^i(a_j^i | s_j^i)} \quad (6)$$

The $Q_{\pi^i}(s, a)$ can be optimized by minimizing the loss:

$$\mathcal{L}_{\pi^i} = \|Q_{\pi^i}(s_t^i, a_t^i) - \rho_t^i * G^i(s_t^i, a_t^i)\|^2 \quad (7)$$

The probability of taking action a for state s under policy π in the next iteration can be calculated by:

$$\pi^{i+1}\{a|s\} = \frac{\exp(Q_{\pi^i}\{a, s\})}{\exp(Q_{\pi^i}\{a=0, s\}) + \exp(Q_{\pi^i}\{a=1, s\})} \quad (8)$$

We develop an ϵ -greedy policy of b based on the Q value from π :

$$b^{i+1}\{a|s\} = \begin{cases} 1 - \epsilon & a = \operatorname{argmax}_a Q_{\pi^i}(s, a); \\ 0 & \text{otherwise;} \end{cases} \quad (9)$$

Algorithm 1: Monte Carlo Based Reinforced Feature Selection with Traverse Strategy

Input: Feature set $\mathcal{F} = \{f_1, f_2, \dots, f_N\}$, downstream machine learning task \mathcal{T} .

Output: Optimal feature subset \mathcal{F}' .

- 1 Initialize the behavior policy b^1 , target policy π^1 , exploration number M , $\mathcal{F}' = \Phi$.
 - 2 **for** $i = 1$ to M **do**
 - 3 Initialize state s_1^i .
 - 4 **for** $t = 1$ to N **do**
 - 5 Derive action a_t^i with behavior policy $b^i(s_t^i)$.
 - 6 Perform a_t^i , getting selected feature subset \mathcal{F}_t^i .
 - 7 Obtain the next state s_{t+1}^i by $\operatorname{represent}(\mathcal{F}_t^i)$ and reward r_t^i by $\operatorname{eval}(\mathcal{F}_t^i)$.
 - 8 **end**
 - 9 Update target policy π^{i+1} by Equation 8 and behavior policy b^{i+1} by Equation 9.
 - 10 **if** $\operatorname{eval}(\mathcal{F}_N^i) > \operatorname{eval}(\mathcal{F}')$ **then**
 - 11 $\mathcal{F}' = \mathcal{F}_N^i$.
 - 12 **end**
 - 13 **end**
 - 14 **Return** \mathcal{F}' .
-

Algorithm 1 shows the process of Monte Carlo based feature selection (MCRFS) with traverse strategy.

B. Early Stopping Monte Carlo Based Reinforced Feature Selection

In many cases, the feature set size N can be very large, meaning that there can be a large number of samples in one episode \mathbf{x}_N^i . The problem is, if the sample at time T is bad

(the Chi-squared distance between $b^i(s_t^i)$ and $\pi^i(s_t^i)$ is large), all the subsequent samples (from T to N) in the episode are skew [14]. The skew samples not only are a waste time to generate, but also do harm to the policy evaluation, therefore we need to find some way to stop the sampling when the episode becomes skew.

1) *Incremental Importance Sampling*: Rather than calculating the importance sampling weight for each sample directly by Equation 6, we here decompose it into an incremental format. Specifically, in the i -th iteration, we define the weight increment:

$$w_t^i = \frac{\pi^i(a_t^i | s_t^i)}{b^i(a_t^i | s_t^i)} \quad (10)$$

and the importance sampling weight can be calculated by:

$$\rho_t^i = \rho_{t-1}^i \cdot w_t^i \quad (11)$$

Thus, at each time, we just need to calculate a simple increment to update the weight.

2) *Early Stopping Monte Carlo Method for Reinforced Feature Selection*: We first propose the stopping criteria, and then propose a decision history based traversing strategy to enhance diversity.

Early stopping criteria. We stop the traverse by probability:

$$p_t^i = \max(0, 1 - \rho_t^i/v) \quad (12)$$

where $0 \leq v \leq 1$ is the stopping threshold.

And for the acquired episode, we recalculate the importance sampling weight for each sample by:

$$w_t^i = p_v^i \cdot \rho_t^i / p_t^i \quad (13)$$

where the p_v can be calculated by:

$$p_v^i = \int \max(0, 1 - \rho_t^i/v) b^i(s_t) ds_t \quad (14)$$

As p_v^i is identical for all samples in the i -th episode regardless of t , the calculation of Equation 13 does almost no increase to the computation.

Decision history based traversing strategy. In the i -th iteration, the stopping criteria stops the traverse at time t , and the features after t are not traversed. With more and more traverses, the front features (e.g., f_1 and f_2) are always selected/deselected by the agent, while the backside features (e.g., f_N and f_{N-1}) get very few opportunity to be decided. To tackle this problem, we record the decision times we made on each feature, and re-rank their orders to diversify the decision process in the next traverse episode. For example, in the past 5 episodes, if the decision times of feature set $\{f_1, f_2, f_3\}$ are $\{5, 2, 4\}$, then in the 6-th episode, the traverse order is $f_2 \rightarrow f_3 \rightarrow f_1$.

Algorithm 2 shows the process of Monte Carlo based feature selection (MCRFS) with early stopping traverse strategy. specifically, we implement the early stopping Monte Carlo based reinforced feature selection method as follows:

Algorithm 2: Monte Carlo Based Reinforced Feature Selection with Early Stopping Traverse Strategy

Input: Feature set $\mathcal{F} = \{f_1, f_2, \dots, f_N\}$, downstream machine learning task \mathcal{T} .

Output: Optimal feature subset \mathcal{F}' .

```

1 Initialize the behavior policy  $b^1$ , target policy  $\pi^1$ ,
  exploration number  $M$ ,  $\mathcal{F}' = \Phi$ .
2 for  $i = 1$  to  $M$  do
3   Initialize state  $s_1^i$ .
4   Rank features with their decision history.
5   for  $t = 1$  to  $N$  do
6     Derive action  $a_t^i$  with behavior policy  $b^i(s_t^i)$ .
7     Perform  $a_t^i$ , getting selected feature subset  $\mathcal{F}_t^i$ .
8     Obtain the next state  $s_{t+1}^i$  by represent( $\mathcal{F}_t^i$ )
       and reward  $r_t^i$  by eval( $\mathcal{F}_t^i$ ).
9     Break the loop with probability  $p_t^i$  derived from
       Equation 12;
10  end
11  Update target policy  $\pi^{i+1}$  by Equation 8 and
    behavior policy  $b^{i+1}$  by Equation 9.
12  if eval( $\mathcal{F}_N^i$ ) > eval( $\mathcal{F}'$ ) then
13    |  $\mathcal{F}' = \mathcal{F}_N^i$ .
14  end
15 end
16 Return  $\mathcal{F}'$ .
```

1. Use a random behavior policy b^0 to traverse the feature set. Stop the traverse with the probability in Equation 12 and get an episode $\mathbf{x}_{N_0}^0$.
2. Evaluate the policy π^0 to get the Q value Q^0 by minimizing Equation 7, and derive the updated policy π^1 and b^1 from Equation 8 and 9 respectively.
3. Update the record of traverse times for each feature. Re-rank feature order. The smaller times one feature was traversed, the more forward order it should get.
4. Use the updated policy π^1 and b^1 to traverse the re-ranked feature set for the next M steps. Derive the policy π^M and b^M . Use π^M to traverse the feature set without stopping criteria, and derive the final feature subset.

C. Interactive Reinforcement Learning

As all the steps in this section belong to the same iteration, we omit the superscript i in each denotation for simplicity.

Reinforcement learning is proposed to develop the optimal policy $\pi_{\mathcal{M}}^*(s) = \operatorname{argmax}_a Q_{\mathcal{M}}^*(s, a)$ for an MDP \mathcal{M} . The optimal Q-value can be updated by Bellman equation [15]:

$$Q_{\mathcal{M}}^*(s_t, a_t) = \mathcal{R}(s_t, a_t) + \gamma * \max_{a_{t+1}} Q_{\mathcal{M}}^*(s_{t+1}, a_{t+1}) \quad (15)$$

Interactive reinforcement learning (IRL) is proposed to accelerate the learning process of reinforcement learning (RL) by providing external action advice to the RL agent [16]. As Figure 4 shows, for selected advising states, the action of RL agent is decided by the advisor's action advice instead of its

own policy. The algorithm to select advising states varies with the problem setting. Typical algorithms for selecting advising states include early advising, importance advising, mistake advising and predictive advising [17]. To better evaluate the utility of the state-action pair (s_t, a_t) , we define a utility function $\mathcal{U}(s_t, a_t)$. The utility function can give a feedback of how the action benefits from the state from the advisor's point of view.

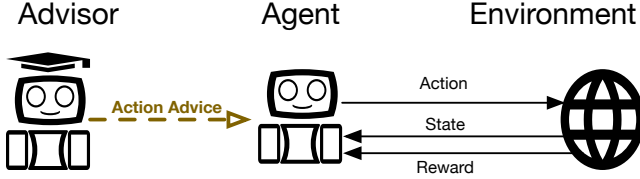


Fig. 4: Classic interactive reinforcement learning. The advisor gives the agent advice at the action level.

Reward-Level Interactive Reinforcement Learning. In reinforcement learning (RL), we aim to obtain the optimal policy for the MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma, \mathcal{P}\}$. However, in IRL, when we change the reward function \mathcal{R} to a more inspiring reward function \mathcal{R}' , the original MDP \mathcal{M} is changed to a new MDP $\mathcal{M}' = \{\mathcal{S}, \mathcal{A}, \mathcal{R}', \gamma, \mathcal{P}\}$. Without careful design, the optimal policy derived from \mathcal{M}' would be different from the optimal policy for \mathcal{M} . Here we give a universal form of reward advice without limitation on the form of utility function $\mathcal{U}(s, a)$:

$$\mathcal{R}'(a_t, s_t) = \mathcal{R}(a_t, s_t) + c * (\gamma * \mathcal{U}(s_{t+1}) - \mathcal{U}(s_t)) \quad (16)$$

where $\mathcal{U}(s_t) = E_{a_t}[\mathcal{U}(a_t, s_t)]$, c is the weight to balance the proportion of the utility function.

We prove that the optimal policies of \mathcal{M} and \mathcal{M}' are identical when the reward advice is Equation 16:

We firstly subtract $c * \mathcal{U}(s_t)$ from both sides of Equation 15, and we have:

$$\begin{aligned} Q_{\mathcal{M}}^*(s_t, a_t) - c * \mathcal{U}(s_t) &= \mathcal{R}(s_t, a_t) \\ &+ \gamma * \max_{a_{t+1}} Q_{\mathcal{M}}^*(s_{t+1}, a_{t+1}) - c * \mathcal{U}(s_t) \end{aligned} \quad (17)$$

We add and subtract $c * \gamma * \mathcal{U}(s_{t+1})$ on the right side:

$$\begin{aligned} Q_{\mathcal{M}}^*(s_t, a_t) - c * \mathcal{U}(s_t) &= \mathcal{R}(s_t, a_t) \\ &+ \gamma * \max_{a_{t+1}} Q_{\mathcal{M}}^*(s_{t+1}, a_{t+1}) - c * \mathcal{U}(s_t) \\ &+ c * \gamma * \mathcal{U}(s_{t+1}) - c * \gamma * \mathcal{U}(s_{t+1}) \\ &= \mathcal{R}(s_t, a_t) + c * \gamma * \mathcal{U}(s_{t+1}) - c * \mathcal{U}(s_t) \\ &+ \gamma * \max_{a_{t+1}} [Q_{\mathcal{M}}^*(s_{t+1}, a_{t+1}) - c * \mathcal{U}(s_{t+1})] \end{aligned} \quad (18)$$

We define:

$$Q_{\mathcal{M}'}^{\partial}(s_t, a_t) = Q_{\mathcal{M}}^*(s_t, a_t) - c * \mathcal{U}(s_t) \quad (19)$$

Then Equation 18 has the new form:

$$\begin{aligned} Q_{\mathcal{M}'}^{\partial}(s_t, a_t) &= \mathcal{R}(s_t, a_t) + c * [\gamma * \mathcal{U}(s_{t+1}) - \mathcal{U}(s_t)] \\ &+ \gamma * \max_{a_{t+1}} Q_{\mathcal{M}'}^{\partial}(s_{t+1}, a_{t+1}) \\ &= \mathcal{R}'(s_t, a_t) + \gamma * \max_{a_{t+1}} Q_{\mathcal{M}'}^{\partial}(s_{t+1}, a_{t+1}) \end{aligned} \quad (20)$$

Algorithm 3: Reward-Level Interactive Reinforcement Learning

- 1 Initialize replay memory \mathcal{D} ; Initialize the Q -value function with random weights; Initialize the advising state number N_a , stop time T ;
 - 2 **for** $t = 1$ **to** T **do**
 - 3 $a_t = \begin{cases} \text{random action} & \text{with probability } \epsilon; \\ \max_{a_t} Q(s_t, a_t) & \text{with probability } 1 - \epsilon; \end{cases}$
 - 4 Perform a_t , obtaining reward $\mathcal{R}(a_t, s_t)$ and next state s_{t+1} ;
 - 5 $\mathcal{R}'(s_t, a_t) = \begin{cases} \mathcal{R}(s_t, a_t) & t > N_a; \\ \mathcal{R}(s_t, a_t) + c * (\gamma * \mathcal{U}(s_{t+1}) - \mathcal{U}(s_t)) & t \leq N_a; \end{cases}$
 - 6 Store transition $(s_t, a_t, \mathcal{R}'(s_t, a_t), s_{t+1})$ in \mathcal{D} ;
 - 7 Randomly sample mini-batch of data from \mathcal{D} ;
 - 8 Update $Q(s, a)$ with the sampled data;
 - 9 **end**
-

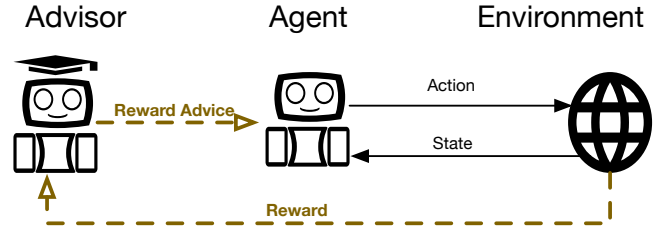


Fig. 5: Reward-level interactive reinforcement learning. The advisor gives advice at the reward level.

which is the Bellman equation of $Q_{\mathcal{M}'}^{\partial}(s_t, a_t)$ with reward \mathcal{R}' , meaning $Q_{\mathcal{M}'}^{\partial}(s_t, a_t)$ is the optimal policy Q -value for MDP \mathcal{M}' , i.e.,

$$Q_{\mathcal{M}'}^*(s_t, a_t) = Q_{\mathcal{M}'}^{\partial}(s_t, a_t) \quad (21)$$

We combine Equation 19 and Equation 21 and have:

$$Q_{\mathcal{M}'}^*(s_t, a_t) = Q_{\mathcal{M}}^*(s_t, a_t) - c * \mathcal{U}(s_t) \quad (22)$$

Obviously,

$$\begin{aligned} \operatorname{argmax}_{a_t} Q_{\mathcal{M}'}^*(s_t, a_t) &= \operatorname{argmax}_{a_t} [Q_{\mathcal{M}}^*(s_t, a_t) - c * \mathcal{U}(s_t)] \\ &= \operatorname{argmax}_{a_t} Q_{\mathcal{M}}^*(s_t, a_t) \end{aligned} \quad (23)$$

which reveals the optimal policy of MDP \mathcal{M}' with reward \mathcal{R}' is identical to the optimal policy of MDP \mathcal{M} with reward \mathcal{R} .

As the reward advice \mathcal{R}' consists of more information than the original reward \mathcal{R} , it can help the reinforcement learning agent explore the environment more efficiently. We give a detailed description of reward-level IRL in Algorithm 3. Specifically, we adapt the early advising strategy [17] to select the advising states, i.e., the advisor gives advice for the first n states the IRL agent meets.

D. Comparison with Prior Literature

Compared with filter methods, our methods capture feature interactions; Compared with wrapper methods, our methods reduce the search space; Compared with embedded methods, our methods don't rely on strong structured assumptions; Compared with multi-agent reinforcement learning feature selection, our methods achieve parallel performance with lower computational cost.

IV. EXPERIMENTAL RESULTS

We conduct extensive experiments on real-world datasets to study: (1) the overall performance of early stopping Monte Carlo based reinforced feature selection (**ES-MCRFS**); (2) the training efficiency of the early stopping criteria; (3) the sensitivity of the threshold in the early stopping criteria; (4) the computational burden of the traverse strategy; (5) the decision history based traverse strategy; (6) the behavior policy in the ES-MCRFS.

A. Experimental Setup

1) *Data Description*: We use four publicly available datasets on classification task to validate our methods, i.e., Forest Cover (FC) dataset [18], Spambase (Spam) dataset [19], Insurance Company Benchmark (ICB) dataset [20] and Arrhythmia (Arrhy) dataset [21]. The statistics of the datasets are in Table II.

TABLE II: Statistics of datasets.

	FC	Spam	ICB	Arrhy
Features	54	57	86	274
Samples	15120	4601	5000	452

2) *Evaluation Metrics*: In the experiments, we have classification as the downstream task for feature selection problem, therefore we use the two most popular evaluation metrics for classification task:

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 is given by $F1 = \frac{2*P*R}{P+R}$, where $P = \frac{TP}{TP+FP}$ is precision and $R = \frac{TP}{TP+FN}$ is recall.

3) *Baseline Algorithms*: We compare our proposed ES-MCRFS method with the following baselines: (1) **K-Best** ranks features by unsupervised scores with the label and selects the top k highest scoring features [1]. In the experiments, we set k equals to half of the number of input features. (2) **LASSO** conducts feature selection via $l1$ penalty [9]. The hyper parameter in LASSO is its regularization weight λ which is set to 0.15 in the experiments. (3) **GFS** selects features by calculating the fitness level for each feature to generate better feature subsets via crossover and mutation [22]. (4) **mRMR** ranks features by minimizing feature's redundancy and maximizing their relevance with the label [23]. (5) **RFE** selects features by recursively selecting smaller and smaller feature subsets [24]. (6) **MARFS** is a multi-agent reinforcement learning based feature selection method [11]. It uses M feature

agents to control the selection/deselection of the M features. Besides, we also compare our method with its variant without early stopping strategy, i.e., Monte Carlo based reinforced feature selection **MCRFS**.

4) *Implementation*: In the experiments, for all deep networks, we set mini-batch size to 16 and use AdamOptimizer with a learning rate of 0.01. For all experience replays, we set memory size to 200. We set the Q network in our methods as a two-layer ReLU with 64 and 8 nodes in the first and second layer. The classification algorithm we use for evaluation is a random forest with 100 decision trees. The stop time is set to 3000 steps. The state representation method in reinforced feature selection is an auto-encoder method whose encoder/decoder network is a two-layer ReLU with 128 and 32 nodes in the first and second layer.

5) *Environmental Setup*: The experiments were carried on a server with an I9-9920X 3.50GHz CPU, 128GB memory and a Ubuntu 18.04 LTS operation system.

B. Overall Performance

We compare the proposed ES-MCRFS method with baseline methods and its variant with regard to the predictive accuracy. As Table III shows, the MCRFS, which simplify the reinforced feature selection into a single-agent formulation, achieves similar performance with the multi-agent MARFS. With the help of traverse strategy and early stopping criteria, the ES-MCRFS outperforms all the other methods.

C. Sensitivity Study of Early Stopping Criteria

We study the threshold sensitivity in the early stopping criteria by differing the threshold v and evaluate the predictive accuracy. Figure 6 shows that the optimal threshold for the four datasets are 0.4, 0.5, 0.7, 0.7. It reveals that the early stopping criteria is sensitive to the pre-defined threshold, and the optimal threshold varies on different datasets.

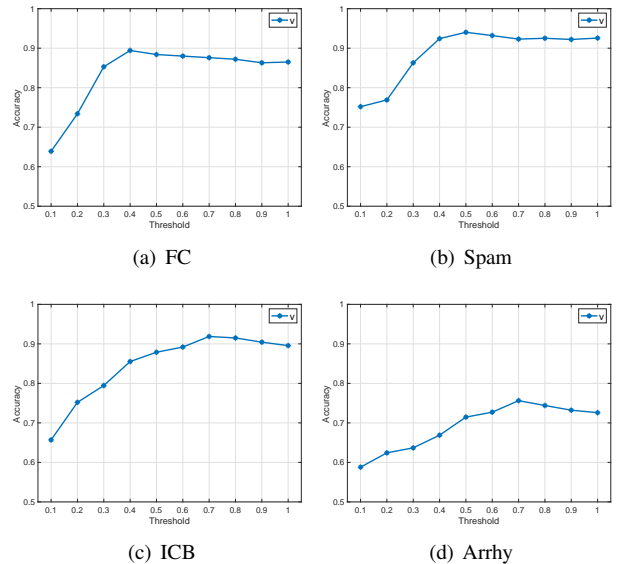


Fig. 6: Threshold sensitivity of early stopping criteria.

TABLE III: Overall performance.

		FC		Spam		ICB		Arrhy	
		Acc	F1	Acc	F1	Acc	F1	Acc	F1
Algorithms	K-Best	0.7904	0.8058	0.9207	0.8347	0.8783	0.8321	0.6382	0.6406
	LASSO	0.8438	0.8493	0.9143	0.8556	0.8801	0.8507	0.6293	0.6543
	GFS	0.8498	0.8350	0.9043	0.8431	0.9099	0.637	0.6406	0.6550
	mRMR	0.8157	0.8241	0.8980	0.8257	0.8998	0.8423	0.6307	0.6368
	RFE	0.8046	0.8175	0.9351	0.8480	0.9045	0.8502	0.6452	0.6592
	MARFS	0.8653	0.8404	0.9219	0.8742	0.8902	0.8604	0.7238	0.6804
	MCRFS	0.8688	0.8496	0.9256	0.8738	0.8956	0.8635	0.7259	0.7152
	ES-MCRFS	0.8942	0.8750	0.9402	0.9067	0.9187	0.8803	0.7563	0.7360

D. Training Efficiency of Early Stopping Criteria

We compare the predictive accuracy with different numbers of training episodes to study the training efficiency of the early stopping. Figure 7 shows that with early stopping criteria, the Monte Carlo reinforced feature selection can achieve convergence more quickly, and the predictive accuracy can be higher after convergence.

E. Study of the Behavior Policy

We study the difference between random behavior policy and the ϵ -greedy policy presented in Equation 9. We combine the two policies with MCRFS and ES-MCRFS respectively. Figure 8 shows that the ϵ -greedy policy outperforms the random behavior policy on all datasets.

F. Computational Burden of Traverse Strategy

We compare the computational burden of the MCRFS which uses single-agent and the traverse strategy to substitute the multi-agent strategy in the MARFS. Table IV shows that the CPU and memory cost when implementing the two methods. Our method MCRFS requires less computational resources than the multi-agent MARFS.

TABLE IV: CPU and memory (in MB) occupation.

	FC		Spam		ICB		Arrhy	
	CPU	Mem	CPU	Mem	CPU	Mem	CPU	Mem
MARFS	72%	1531	75%	1502	86%	1797	97%	4759
MCRFS	57%	1429	54%	1395	59%	1438	55%	1520

G. Decision History Based Traverse Strategy

We study the decision history based traverse strategy by comparing its performance with the vanilla traverse strategy on ES-MCRFS. Table V shows that the decision history can significantly improve performance of the traverse strategy.

TABLE V: Traverse strategy ablation. DH for decision history.

	FC		Spam		ICB		Arrhy	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1
No DH	0.75	0.82	0.83	0.79	0.75	0.81	0.59	0.56
With DH	0.89	0.88	0.94	0.91	0.92	0.88	0.76	0.74

H. Training Efficiency of reward-level interactive strategy

We compare the predictive accuracy with different numbers of training episodes to study the training efficiency of the reward-level interactive (RI) strategy. Figure 9 shows that with RI, the Monte Carlo reinforced feature selection can achieve

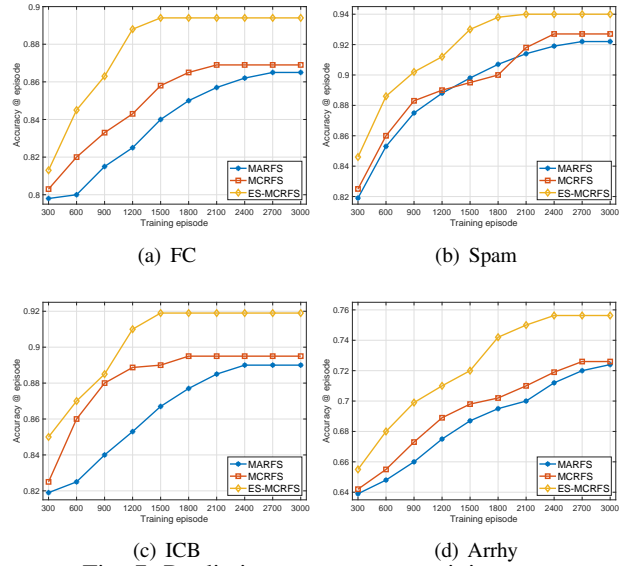


Fig. 7: Predictive accuracy on training step.

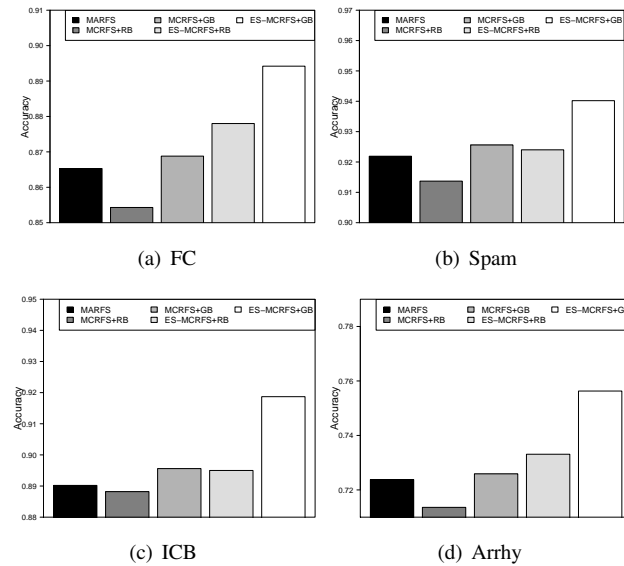


Fig. 8: Predictive accuracy on different training strategies.

RB for random behavior policy and GB for ϵ -greedy behavior policy.

TABLE VI: Performance with different utility function

		FC		Spam		ICB		Arrhy	
		Acc	F1	Acc	F1	Acc	F1	Acc	F1
Utility	Rd	0.8689	0.8433	0.9250	0.8749	0.8997	0.8788	0.7340	0.6955
	Rv	0.8703	0.8507	0.9317	0.8831	0.9001	0.8793	0.7393	0.7143
	$Rv - Rd$	0.8842	0.8650	0.9402	0.8949	0.9117	0.8903	0.7492	0.7258

convergence more quickly. However, as the ES-MCRFS already achieves good performance, the RI can not improve its final performance.

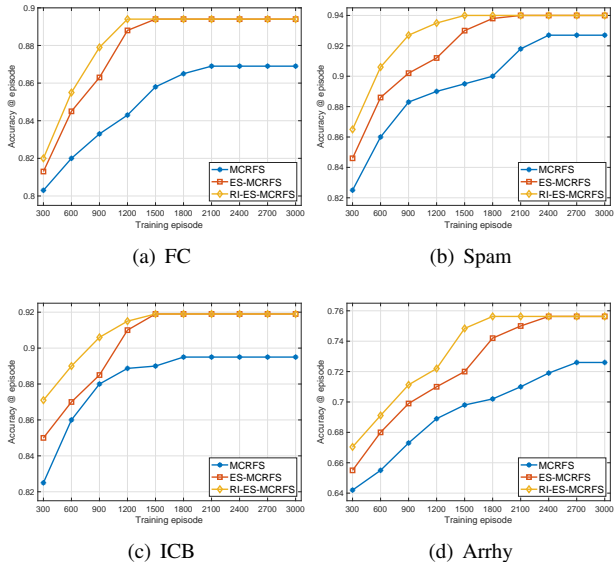


Fig. 9: Predictive accuracy on training step.

I. Study of the Utility Function

We define the utility function \mathcal{U} as the combination of relevance (Rv) function and redundancy (Rd) function. Here we study the impact of the two components for the utility function. Table VI shows that when we use Rv independently as the utility function, its performance is better than the Rd . This is because Rv evaluates the relationship between features and the label, which is directly related to the classification task, while Rd evaluates the relationship among features, which is an indirect evaluation to the classification task. The combination of the two functions ($Rv - Rd$) as the utility function significantly outperforms each of the independent functions, revealing the Rd and Rv coordinate and make up each other's shortage.

V. RELATED WORK

Efficient Sampling in Reinforcement Learning. Reinforcement learning is a trial-and-error based method, which requires high-quality samples to train its policy. It is always a hot topic to pursue efficient sampling for reinforcement learning. One research direction is to generate training samples with high quality based on the importance sampling technology, such as rejection control [25] and marginalized importance sampling [26]. These methods basically control the sampling process based on the importance sampling weight. Another research

direction is to sample diversified sample from different policy parameters. The diversity partially contributes to the exploration and thus have better performance on some specific tasks [27]. However, these methods suffer from slow convergence and no theoretical guarantee [28]. Besides, there are other attempts to develop sample efficient reinforcement learning, such as curiosity-driven exploration and hybrid optimization [29], [30].

Feature Selection. Feature selection can be categorized into three types, i.e., filter methods, wrapper methods and embedded methods [31], [32]. Filter methods rank features only by relevance scores and only top-ranking features are selected. The representative filter methods is the univariate feature selection [2]. The representative wrapper methods are branch and bound algorithms [7], [8]. Wrapper methods are supposed to achieve better performance than filter methods since they search on the whole feature subset space. Evolutionary algorithms [5], [6] low down the computational cost but could only promise local optimum results. Embedded methods combine feature selection with predictors more closely than wrapper methods. The most widely used embedded methods are LASSO [9] and decision tree [10].

Interactive Reinforcement Learning. Interactive reinforcement learning (IRL) is proposed to accelerate the learning process of reinforcement learning. Early work on the IRL topic can be found in [33], where the authors presents a general approach to making robots which can improve their performance from experiences as well as from being taught. Unlike the imitation learning which intends to learn from an expert other than the environment [34], [35], IRL sticks to learning from the environment and the advisor is only an advice-provider in its apprenticeship [36], [37]. As the task for the advisor is to help the agent pass its apprenticeship, the advisor has to identify which states belong to the apprenticeship. In [17], the authors study the advising state selection and propose four advising strategies, i.e., early advising, importance advising, mistake correcting and predictive advising.

VI. CONCLUSION REMARKS

Summary. In this paper, we study the problem of improving the training efficiency of reinforced feature selection (RFS). We propose a traverse strategy to simplify the multi-agent formulation of the RFS to a single-agent framework, an implementation of Monte Carlo method under the framework, and two strategies to improve the efficiency of the framework. **Theoretical Implications.** The single-agent formulation reduces the requirement of computational resources, the early stopping strategy improves the training efficiency, the decision history based traversing strategy diversify the training process,

and the interactive reinforcement learning accelerates the training process without changing the optimal policy.

Practical Implications. Experiments show that the Monte Carlo method with the traverse strategy can significantly reduce the hardware occupation in practice, the decision history based traverse strategy can improve performance of the traverse strategy, the interactive reinforcement learning can improve the training of the framework.

Limitations and Future Work. Our method can be further improved from the following aspects: 1) The framework can be adapted into a parallel framework, where more than one (but much smaller than the feature number) agents work together to finish the traverse; 2) Besides reward level, the interactive reinforcement learning can obtain advice from action level and sampling level. 3) The framework can be implemented on any other reinforcement learning frameworks, e.g., deep Q-network, actor critic and proximal policy optimization (PPO).

ACKNOWLEDGEMENTS

This research was partially supported by the National Science Foundation (NSF) via the grant numbers: 2008837, 2007210, 1755946, I2040950 and 2006889.

REFERENCES

- [1] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Icml*, vol. 97, 1997, pp. 412–420.
- [2] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1289–1305, 2003.
- [3] M. A. Hall, "Feature selection for discrete and numeric class machine learning," 1999.
- [4] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, 2003, pp. 856–863.
- [5] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," in *Feature extraction, construction and selection*. Springer, 1998, pp. 117–136.
- [6] Y. Kim, W. N. Street, and F. Menczer, "Feature selection in unsupervised learning via evolutionary search," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 365–369.
- [7] P. M. Narendra and K. Fukunaga, "A branch and bound algorithm for feature subset selection," *IEEE Transactions on computers*, no. 9, pp. 917–922, 1977.
- [8] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.
- [9] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.
- [10] 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*, vol. 21, no. 2, pp. 930–942, 2007.
- [11] 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.
- [12] 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.
- [13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [14] S. N. MacEachern, M. Clyde, and J. S. Liu, "Sequential importance sampling for nonparametric bayes models: The next generation," *Canadian Journal of Statistics*, vol. 27, no. 2, pp. 251–267, 1999.
- [15] R. Bellman and R. Kalaba, "Dynamic programming and statistical communication theory," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 43, no. 8, p. 749, 1957.
- [16] 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.
- [17] L. Torrey and M. Taylor, "Teaching on a budget: Agents advising agents in reinforcement learning," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013, pp. 1053–1060.
- [18] J. A. Blackard, "Kaggle forest cover type prediction," [EB/OL], 2015, <https://www.kaggle.com/c/forest-cover-type-prediction/data>.
- [19] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: <http://archive.ics.uci.edu/ml>
- [20] P. Van Der Putten and M. van Someren, "Coil challenge 2000: The insurance company case," Technical Report 2000-09, Leiden Institute of Advanced Computer Science . . . , Tech. Rep., 2000.
- [21] H. A. Guvenir, B. Acar, G. Demiroz, and A. Cekin, "A supervised machine learning algorithm for arrhythmia analysis," in *Computers in Cardiology 1997*, 1997, pp. 433–436.
- [22] R. Leardi, "Genetic algorithms in feature selection," in *Genetic algorithms in molecular modeling*. Elsevier, 1996, pp. 67–86.
- [23] 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*, vol. 27, no. 8, pp. 1226–1238, 2005.
- [24] 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*, vol. 83, no. 2, pp. 83–90, 2006.
- [25] J. S. Liu, R. Chen, and W. H. Wong, "Rejection control and sequential importance sampling," *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 1022–1031, 1998.
- [26] T. Xie, Y. Ma, and Y.-X. Wang, "Towards optimal off-policy evaluation for reinforcement learning with marginalized importance sampling," in *Advances in Neural Information Processing Systems*, 2019, pp. 9668–9678.
- [27] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin *et al.*, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.
- [28] Y. Yu, "Towards sample efficient reinforcement learning," in *IJCAI*, 2018, pp. 5739–5743.
- [29] M. Raginsky, A. Rakhlin, and M. Telgarsky, "Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis," *arXiv preprint arXiv:1702.03849*, 2017.
- [30] D. Wang, P. Wang, J. Zhou, L. Sun, B. Du, and Y. Fu, "Defending water treatment networks: Exploiting spatio-temporal effects for cyber attack detection," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 32–41.
- [31] K. Liu, Y. Fu, L. Wu, X. Li, C. Aggarwal, and H. Xiong, "Automated feature selection: A reinforcement learning perspective," *IEEE Transactions on Knowledge and Data Engineering*, 2021.
- [32] 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," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 871–880.
- [33] L. J. Lin, "Programming robots using reinforcement learning and teaching," in *AAAI*, 1991, pp. 781–786.
- [34] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.
- [35] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in neural information processing systems*, 2016, pp. 4565–4573.
- [36] 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.
- [37] D. Wang, P. Wang, K. Liu, Y. Zhou, C. E. Hughes, and Y. Fu, "Reinforced imitative graph representation learning for mobile user profiling: An adversarial training perspective," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4410–4417.