2023 IEEE 39th International Conference on Data Engineering (ICDE) | 979-8-3503-2227-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDE55515.2023.00037

PA-FEAT: Fast Feature Selection for Structured Data via Progress-Aware Multi-Task Deep Reinforcement Learning

Jianing Zhang†, Zhaojing Luo‡, Quanqing Xu§, Meihui Zhang†*
†*Tangshan Research Institute, BIT*, ‡*National University of Singapore*, §*OceanBase*
zhangjianingbit@gmail.com, zhaojing@comp.nus.edu.sg, xuquanqing.xqq@oceanbase.com, meihui_zhang@bit.edu.cn

***Abstract*—Feature selection is an effective technique for structured data analytics, aiming to eliminate redundant features and irrelevant features for downstream tasks (e.g., classification). With the deepening of data-driven decision-making applications in various industries, the demand for real-time structured data analysis is constantly increasing. At this time, high requirements are placed on the time cost of feature selection. However, existing feature selection methods may easily fall into the dilemma of efficiency and effectiveness when faced with this situation due to the huge feature space. In this paper, we study a novel fast feature selection scenario, which is to generalize the knowledge of feature selection from historical structured data analytics tasks (seen tasks) and then quickly apply it to the process of feature selection for future structured data analytics tasks (unseen tasks). We propose a novel Progress-Aware multi-task deep reinforcement learning method for Fast fEAture selecTion (PA-FEAT), which makes full use of various progress-related information generated during the knowledge generalization process to achieve efficiency and effectiveness simultaneously. Extensive results on eight real-world datasets show that PA-FEAT consistently outperforms eight baselines in terms of efficiency and effectiveness.**

***Index Terms*—Data Analytics, Feature Selection, Multi-Task Learning, Deep Reinforcement Learning**

I. Introduction

Structured data contains a huge wealth of information, which is crucial for data-driven decision making [1]–[4]. For better discovering valuable insights from structured data, Feature Selection (FS [5], [6]) is indispensable, aiming to eliminate redundant and irrelevant features in advance and select the optimal feature subset for downstream tasks. Effective FS can help to improve predictive accuracy, reduce dimensionality, shorten training time and increase comprehensibility.

Interactive Structured Data Analysis (ISDA [7], [8]) is an important form of structured data analysis, which empowers users to interactively explore data and make data-driven decisions in a timely manner. As such, efficient feature selection for downstream tasks during ISDA is required to ensure low-latency processing. However, existing feature selection methods can not be directly adopted into ISDA applications in terms of effectiveness and efficiency. This is because traditional methods (e.g., K-Best [9] and RFE [10]) are usually unable to identify feature subsets with satisfactory prediction accuracy. The emerging reinforced selection methods [11]–[13] are more promising to discover best feature subset, but usually require time-consuming calculation process.

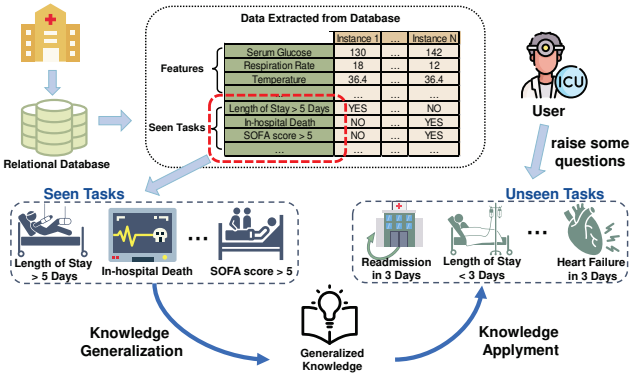


Fig. 1. Illustration of Fast Feature Selection.

We notice that in real applications, it is more often than not to perform various analytics on the same set of data. Consider the healthcare application as an example as illustrated in Figure 1, where patients' hospital data are extracted to support various predictive medical tasks. There could be tasks such as predicting patients' in-hospital death and length of stay in the past, and additional analysis such as readmission prediction may be needed in the future. At the same time, it is not difficult for data analytics systems to collect historical tasks that share the same feature space. *Our key insight is to leverage the implicit associations between multiple analytics tasks over the same feature space to expedite the feature selection for future tasks.* Specifically, we consider utilizing these associations by capturing knowledge in feature selection across multiple historical tasks (seen tasks) and quickly transferring the knowledge to enhance future tasks (unseen tasks).

To the best of our knowledge, there is no previous works that study the problem of leveraging historical tasks to realize fast feature selection for future tasks. Multi-label feature selection methods [14]–[16] can be twisted to address the problem to some extent by considering historical and future tasks at the same time. These methods utilize the correlation between multiple labels and focus on providing a unified feature subset for all labels (each label can be considered as a task) over the same feature space. Thus, such methods can only generate the same feature set for various future tasks, which is apparently

* contact author

2375-026X/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDE55515.2023.00037

ignoring task-specific characteristics and therefore undesired.

An effective way of generalizing knowledge from multiple tasks is Multi-Task Learning (MTL [17]). MTL is a subfield of machine learning in which multiple tasks are learned simultaneously, while using a shared representation for each task. Then the knowledge obtained from seen tasks can be quickly applied to unseen tasks through shared representation for tasks [18]. Observing the recent success of introducing DRL into single task feature selection [13], [19], to take advantage of the powerful global search ability of DRL in feature selection, we extend DRL with MTL, called Multi-Task DRL (MT-DRL), to address the target problem. To this end, we propose a novel MT-DRL framework for Fast fEAture selecTion, called "FEAT", which first generalizes knowledge from multiple historical seen tasks simultaneously, and then quickly applies the knowledge to the feature selection for unseen tasks through shared representation for tasks. However, there are still challenges to be solved in FEAT to ensure effectiveness and efficiency, which largely stem from the difficulties in capturing the knowledge about feature selection across historical tasks.

First, in terms of MTL, simply treating all tasks equally will lead to unbalanced knowledge generalization over multiple seen tasks. Since different tasks have different learning difficulties, they require different amount of learning resources to ensure the quality of learning. For achieving balanced learning, existing MTL methods consider to reduce the contribution of simple tasks to the generalized knowledge by normalizing the gradients from different tasks [20] or weighting different tasks' losses according to their reward magnitudes [21]. However, the balance relying on the gradient similarity or reward magnitudes is usually unstable, especially when gradient or reward vary largely within each task. Besides, due to the lack of capability for allocating learning resources on demand, the above methods are not able to make full use of learning resources to extensively explore hard tasks. To address the problem, we propose a novel adaptive Inter-Task Scheduler (ITS), which first utilizes progress-related information (i.e., uncertainty and distance ratio) to measure the learning needs of different tasks, and then dynamically allocate learning resources for each seen task according to their learning needs to make full use of limited learning resources.

Second, in terms of DRL, how to efficiently search the large feature space is another challenge. DRL-based methods usually suffer from low efficiency. Thus, many research efforts are conducted to make DRL-based methods more efficient. Some researches focus on reward enhancement by using intrinsic reward [22] or introducing a reward randomization mechanism [23] to make exploration more efficient. But at the beginning of each exploration episode, they always start from default initial states, ignoring the possibility of defining more appropriate initial states based on the progress of exploration and exploitation. Go-Explore [24] and its extension [25] propose to use a simple policy (e.g., random) to find valuable experiences by exploring from the appropriate initial states, and then use these experiences to train a learning policy. Since they completely decouple the exploration from the exploitation of learning policy, the exploitation progress of the learning policy is not considered in the selection of appropriate initial states, which weakens the role of appropriate initial states. In this paper, we propose an Intra-Task Explorer (ITE), which contains a novel Experience-Tree (E-Tree) to efficiently organize the information of feature space searching progress of each task from two aspects, i.e., exploration and exploitation, and dynamically customize the initial states based on E-Tree.

In summary, we propose a novel Progress-Aware multi-task deep reinforcement learning method for Fast fEAture selecTion over structured data (PA-FEAT). Our contributions are as follows:

- We propose a DRL based framework for structured data feature selection, which can generalize knowledge from multiple historical tasks efficiently and effectively, and perform fast feature selection for future tasks.
- We propose a novel adaptive Inter-Task Scheduler (ITS), which monitors the learning process and dynamically allocates resources to ensure balanced learning over historical tasks.
- We propose an Intra-Task Explorer (ITE), which contains a novel Experience-Tree structure and an initial state customization strategy to enable efficient search over large feature space for each task.
- We conduct extensive experiments on eight real-world datasets. The results show that PA-FEAT outperforms all other baselines in terms of efficiency and effectiveness.

II. Preliminaries and Problem Formulation

A. Preliminaries

Structured Data. Structured data is widely used in almost every industry [26], [27], and there have been growing interests in designing models for better predictive analysis over structured data [28]–[31]. Specifically, structured data is generally stored in multiple tables (relations) $\{T_1, T_2, ...\}$, and conforms to a tabular format with relationship between different rows and columns. Each column of the logical table corresponds to a specific feature or a predictive attribute and each row represents a data sample in learning models.

For ease of discussion, we formulate structured data as one relational table T of n rows and $m+k$ columns, including m determinant attributes (feature vector) and k dependent attributes (prediction target). Specifically, the i-th row can be denoted as a tuple $(X_i, Y_i) = (x_{i1}, x_{i2}, ..., x_{im}, y_{i1}, y_{i2}, ..., y_{ik})$, where $x_{ij} \in X_i$ is the j-th determinant attribute value and $y_{ij} \in Y_i$ is the j-th dependent attribute value in the i-th row. In our scenario, the prediction analysis will be carried out for more than one dependent attribute.

Reinforcement Learning (RL). RL is known to learn through trial-and-error interactions with a dynamic environment and make long-term optimal decisions. In an ideal reinforcement learning solution, a Markov Decision Process (MDP) is defined as a tuple $(\mathcal{S}, \mathcal{A}, \Pr, R, \mathcal{O}, \rho, \rho_0, \gamma)$, where $\mathcal{S}$, $\mathcal{O}$ and $\mathcal{A}$ are the set of states, observations and actions of the environment, respectively. The initial state s_0 is drawn from the

distribution ρ_0. During time slot $[t, t+1)$, the agent draws an observation $\mathbf{o}_t \in \mathcal{O}$, from the conditional observation probability $\Pr(\mathbf{o}_t|\mathbf{s}_t)$, and then chooses an action $\mathbf{a}_t$ to take. The next state $\mathbf{s}_{t+1}$ is drawn from transition probability $\rho(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ after taking actions $\mathbf{a}_t$ based on state $\mathbf{s}_t$. Subsequently, the agent receives a reward $r_t = R(\mathbf{s}_t, \mathbf{a}_t)$. $\gamma \in [0, 1]$ is a discount factor. In this paper, we use Dueling Deep Q-network (Dueling DQN [32]) to train the policy. Dueling DQN separates the advantage of taking some action in a state from the value of state in value function and approximates value function and advantage function by deep neural network. The Loss function of Dueling DQN is defined as:

$$L_t(\boldsymbol{\theta}_t) = \mathbb{E}_{\mathbf{s}_t, a, r, \mathbf{s}_{t+1}}\big[\big(y_t^{DuelingDQN} - Q(\mathbf{s}_t, a; \boldsymbol{\theta}_t)\big)^2\big], \quad (1a)$$

$$y_t^{DuelingDQN} = r_t + \gamma \max_{a'} Q(\mathbf{s}_{t+1}, a'; \boldsymbol{\theta}^-), \quad (1b)$$

$$Q(\mathbf{s}, a; \boldsymbol{\theta}, \alpha, \beta) = V(\mathbf{s}; \boldsymbol{\theta}, \beta) + \big(A(\mathbf{s}, a; \boldsymbol{\theta}, \alpha) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(\mathbf{s}, a'; \boldsymbol{\theta}, \alpha)\big), \quad (1c)$$

where $Q(\mathbf{s}, a; \boldsymbol{\theta})$ is the value function approximated using deep networks (i.e., Q-network), consisting of the state value estimation $V(\mathbf{s}; \boldsymbol{\theta}, \beta)$ and the zero-centered advantage function $A(\mathbf{s}, a'; \boldsymbol{\theta}, \alpha)$, providing separate estimates of the value of the state and the advantage of actions; $\boldsymbol{\theta}_t$ represents the parameters of the target network $Q(\mathbf{s}, a; \boldsymbol{\theta})$, and $\boldsymbol{\theta}^-$ is a frozen parameter of the target network for a fixed number of iterations while updating the online network $Q(\mathbf{s}, a; \boldsymbol{\theta})$ by gradient descent; α, β represents the parameters of value estimation $V(\mathbf{s}; \boldsymbol{\theta}, \beta)$ and advantage function $A(\mathbf{s}, a'; \boldsymbol{\theta}, \alpha)$ respectively. The Q-network is trained by sampling mini-batches of experiences from the replay buffer uniformly at random, calculating loss according to Eqn.1a and updating parameters.

B. Problem Formulation

Definition 1 (**Task**)**.** Given a structured data table T of n tuples with m determinant attributes and one dependent attribute y, a task denoted as $\mathcal{T} = \{\mathcal{X}, \mathcal{Y}, f(\cdot)\}$ is defined by three components: feature space $\mathcal{X} = \{X_1, ..., X_n\}$ where each $X_j = [x_{j1}, ..., x_{jm}]$ includes the values of m determinant attributes, label space $\mathcal{Y} = \{y_1, ..., y_n\}$ and an objective predictive function $f(\cdot)$ used to predict the target $f(X_j)$ of an instance X_j with the goal of $f(X_j) = y_j$.

When a task $\mathcal{T}$ is defined, the key point is to find an optimal function to approximate $f(\cdot)$, where Feature Selection is required to eliminate redundant and irrelevant features for dimension reduction and better prediction performance.

Definition 2 (**Feature Selection**)**.** Given a task $\mathcal{T}$, let $F = \{att_1, att_2, ..., att_m\}$ denote the set of determinant attributes included in $\mathcal{X}$, Feature Selection is the process of selecting a subset $F' \subset F$, such that $f(\cdot)$ obtained with $\mathcal{X}_{F'}$ achieves better prediction performance for task $\mathcal{T}$, where $\mathcal{X}_{F'}$ denotes the feature space of $\mathcal{X}$ projected on F'.

We now give the definitions of seen task and unseen task.

Definition 3 (**Seen Task**)**.** A predictive task denoted as $\mathcal{T}^s$ is defined as a seen task if and only if the label space $\mathcal{Y}$ of $\mathcal{T}$ has been observed before.

Definition 4 (**Unseen Task**)**.** A predictive task denoted as $\mathcal{T}^u$ is defined as an unseen task if and only if the label space $\mathcal{Y}$ of $\mathcal{T}$ has never been observed before.

For an unseen task $\mathcal{T}^u$, the label space $\mathcal{Y}$ is invisible, however, the feature space $\mathcal{X}$ is available.

We now give the problem statement.

Fast Feature Selection. Given a set of seen tasks $\mathcal{H} = \{\mathcal{T}_1^s, \mathcal{T}_2^s, ..., \mathcal{T}_t^s\}$ and an unseen task $\mathcal{T}^u$ which share the same feature space $\mathcal{X}$, Fast Feature Selection aims to utilize the information about multiple seen tasks in $\mathcal{H}$ to enhance and accelerate feature selection for the unseen task $\mathcal{T}^u$.

The label space of the unseen task $\mathcal{T}^u$ is invisible before $\mathcal{T}^u$ is received. Fast Feature Selection requires a quick response to the arrival of the unseen task and giving the results of feature selection with low time cost.

For making full use of information about multiple seen tasks to guide feature selection on the unseen task, we formulate fast feature selection as a multi-task learning model for knowledge generalization across multiple seen tasks, where exploration on each seen task contributes to knowledge generalization. When the unseen task is received, the generalized knowledge will be transferred quickly. Thus, Multi-Task DRL is utilized to capture the general knowledge through exploration of feature selection on multiple seen tasks and accumulate the knowledge by the updating decision mechanism in deep reinforcement learning. As a result, when the unseen task is received, the knowledge can be quickly transferred through exploiting the decision mechanism on the unseen task.

For obtaining the above mentioned decision mechanism, sufficient exploration on each seen task is required. To this end, feature selection on each seen task is formulated in a sequential decision manner [33], where features are sequentially scanned one by one and each time the scanning feature is decided to be selected or not.

In our fast feature selection scenario, the common elements in reinforcement learning are reformulated as follows:

Agent. The agent scans features sequentially and decides whether to select the feature being scanned (take an action).

Action. For feature selection problem, the action a decides to select ($a = 1$) or deselect ($a = 0$) the feature being scanned.

Environment. The environment interacts with the agent and provides reward as feedback. As soon as the agent take an action a (select the scanning feature or not), the original environment state $\mathbf{s}$ will change to a new state $\mathbf{s}'$ and provide a reward as feedback simultaneously.

State. Observing the current state of the environment is important for agent to determine subsequent action. In our design, for better observing the environment, the state is to mark the corresponding seen task, record the selected features and the current scanning position.

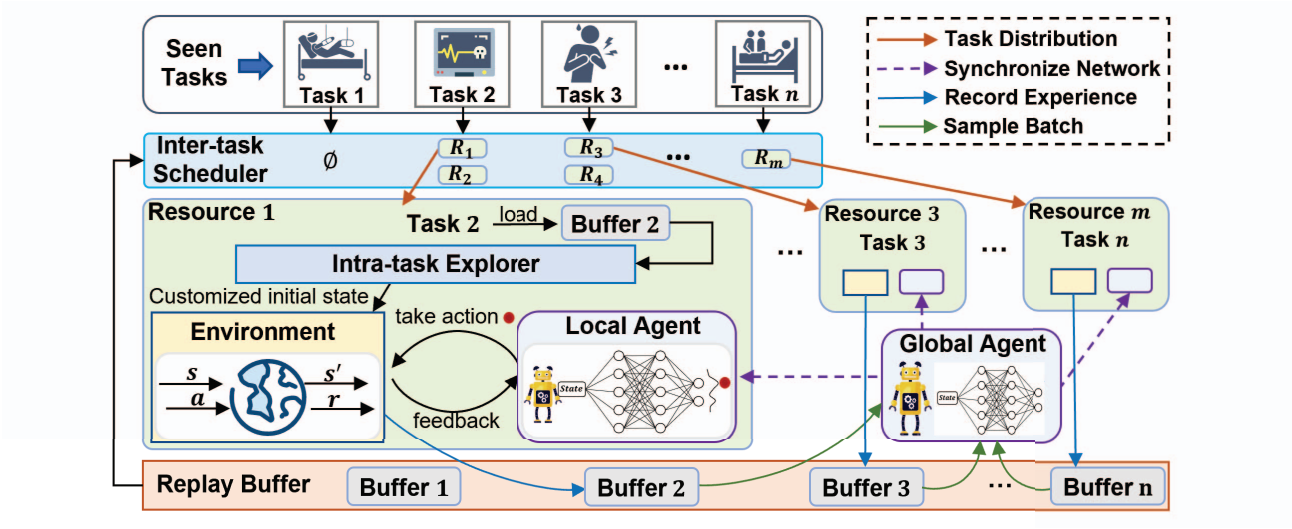


Fig. 2. Overall proposed model PA-FEAT.

Reward. The reward is a feedback given by the environment and the agent evaluates the action through the reward. In feature selection, a well-performing reward function is supposed to make an accurate evaluation of feature subsets. Inspired by the existing reward functions [33]–[35], in our framework, the reward function is defined as:

$$r = P(CLS(X^{F^{'}}), Y), \tag{2}$$

where $F^{'}$ denotes the selected feature subset; $X^{F^{'}}$ denotes the masked feature vectors of samples (the values of unselected features are masked by zero or mean value); Y denotes the ground truth of the samples' labels; $CLS(\cdot)$ denotes the classifier that predicts the class of each sample; $P(\cdot)$ denotes the performance of classification given by the selected feature subset $F^{'}$. In addition, since training an RL-based method needs to call the reward function frequently, instead of training a classifier from scratch each time we have a new feature subset, we pretrain a classifier using all features in advance, which uses masked feature vectors to do classification. More details will be introduced in Section IV-A4.

III. Proposed Method

In this section, we first present the overview of PA-FEAT and then we elaborate on each module of PA-FEAT.

A. Overview

Our proposed method PA-FEAT consists of three components: a novel DRL framework called "FEAT", a inter-task resource scheduler called "Inter-task Scheduler" and a tree-based intra-task optimizer to discover valuable states called "Intra-task Explorer", as shown in Fig. 2.

In PA-FEAT, for obtaining the general knowledge across multiple seen tasks by the updating decision mechanism based on reinforcement learning, an agent for feature selection is required to be trained for accumulating experience in interactions with the environments corresponding to all seen tasks. Specifically, a global agent is initialized at first. Subsequently, Inter-task Scheduler is invoked to allocate the probability that the task will be chosen by resources based on progress-related information of each seen task for the learning balance between multiple seen tasks. Then, each resource is distributed to a seen task by Inter-task Scheduler. For each resource, interactions between the local agent (synchronized from the global agent) and the corresponding environment to the distributed task will go on. At the beginning of the interaction, Inter-task Explorer is invoked to get a customized initial state for deepening the exploration on the seen task corresponding to the resource. Finally, the experience generated in the interactions is recorded into the corresponding replay buffer to the seen task. PA-FEAT periodically sample a batch from each replay buffer to update the Q-Network (DNN) of the global agent.

The above process will repeat until the global agent is well-trained. After a well-performing global agent is obtained, as soon as an unseen task is received, the global agent can quickly response through exploiting the Q-Network and give the feature selection result for the unseen task.

B. FEAT

FEAT is the basic framework to achieve fast feature selection based on Multi-task Deep Reinforcement Learning. The essential concern of FEAT is to utilize the experience generated from the interactions between the local agent (by synchronization from the global agent) and each environment of seen tasks to update the Q-Network of the global agent. Here, the global agent and local agents are essentially neural networks with same architecture, which can map environment states to actions. As shown in Fig. 2, the local agents are responsible to interact with multiple environments corresponding to each seen task and collect experience data. The global agent uses these experience data to update network parameters and copies the latest parameters to each local agent after parameter update (i.e., Synchronize Network in Fig. 2).

Algorithm 1: FEAT

1 Initialize $\boldsymbol{\theta}$.
2 Initialize $\mathcal{B}^k$ for each seen task k.
3 **for** $iteration = 1, 2, ...$ **do**
4 **for** $i = 1, 2, ..., N$ **do**
5 Choose a task k evenly from all seen tasks;
6 Get default initial state $\mathbf{s}_0^k$;
7 **for** $t = 0, 1, ..., T-1$ **do**
8 Get state $\mathbf{s}_t^k$;
9 Get the number of selected feature F_s from $\mathbf{s}_t^k$;
10 **if** $F_s/F_{all} > mfr$ **then**
11 break
12 Use Eqn. (3) to calculate $Q_{\boldsymbol{\theta}}(\mathbf{a}_t^k|\mathbf{s}_t^k)$;
13 Sample action $\mathbf{a}_t^k \sim Q_{\boldsymbol{\theta}}(\mathbf{a}_t^k|\mathbf{s}_t^k)$;
14 Execute actions $\mathbf{a}_t^k$, receive r_t^k and reach the new state $\mathbf{s}_{t+1}^k$;
15 Collect trajectory $\tau_{0:T}^k$;
16 Compute $\hat{R}_{0:T-1}^k$ and $A_{0:T-1}^k$ on $\tau_{0:T}^k$;
17 **for** $t = 0, 1, ..., T-1$ **do**
18 $\mathcal{B}^k = \mathcal{B}^k \cup \{(\tau_{t:t+1,i}^k, A_{t:t+1}^k, \hat{R}_{t:t+1}^k)\}$;
19 **for** $i = 1, 2, ..., K$ **do**
20 Optimize L in Eqn. (1) w.r.t $\boldsymbol{\theta}$, with batch size M using samples from $\mathcal{B}^k$;
21 $\boldsymbol{\theta}_{old} \leftarrow \boldsymbol{\theta}$;
22 **for** $j = 1, ..., M$ **do**
23 Execute an episode to form trajectory $\tau_{0:T}^j$ using trained policy;
24 Map trajectory $\tau_{0:T}^j$ into a feature subset for unseen task j;

In order to enable the local agent to distinguish which task corresponding environment it is currently interacting with, we embed the presentation for each task into the state presentation the environment. Specifically, we use a vector where each item is the absolute value of the Pearson correlation coefficient [36] between each feature and the label set of the corresponding task as the presentation of the task. Thus, for local agents in the environment corresponding to different seen tasks, the reward mechanism is related to the corresponding seen task respectively and different from each other. Through the task representation, when an unseen task is received, FEAT can quickly create an environment for the unseen task and utilize the global agent with a well-trained Q-Network to select features for the unseen task.

Experiences are generated and collected in each environment as follows. Firstly, the local agent of FEAT gets the environment state and chooses an action to take. Subsequently, the local agent takes the action and receives reward. The environment reaches next state and information about action, state transition and reward are collected. Then, the local agent gets the new environment state and repeats above operations from the new state. Experiences will be taken as training data to update the Q-Network of the global agent.

The whole structure of FEAT can be depicted by the following equations:

$$V_{\boldsymbol{\theta}}(\mathbf{s}_t^k) = \mathrm{f}^V(\mathbf{s}_t^k), \tag{3a}$$

$$A_{\boldsymbol{\theta}}(\mathbf{a}_t^k|\mathbf{s}_t^k) = \mathrm{f}^A(\mathbf{s}_t^k), \tag{3b}$$

$$Q_{\boldsymbol{\theta}}(\mathbf{a}_t^k|\mathbf{s}_t^k) = \mathrm{f}^E(V_{\boldsymbol{\theta}}(\mathbf{s}_t^k)) + \mathrm{f}^N(A_{\boldsymbol{\theta}}(\mathbf{a}_t^k|\mathbf{s}_t^k)), \tag{3c}$$

where $\boldsymbol{\theta}$ denotes all the learnable parameters; $\mathrm{f}^V(\cdot)$ and $\mathrm{f}^A(\cdot)$ are DNNs; $A_{\boldsymbol{\theta}}$ represents advantage function, and $V_{\boldsymbol{\theta}}$ represents value function; f^E represents the function to broadcast a scalar as a vector and f^N represents zero centering which subtract the average value from each item. The key idea of FEAT is that when the current state $\mathbf{s}_t^k$ of the environment corresponding to Task k is acquired, through $A_{\boldsymbol{\theta}}$ and $V_{\boldsymbol{\theta}}$, the state presentation is mapped to an action distribution $Q_{\boldsymbol{\theta}}(\mathbf{a}_t^k|\mathbf{s}_t^k)$, as shown in Eqn. (3c).

The main process of FEAT is shown in Algorithm 1. At first, we initialize learnable parameters $\boldsymbol{\theta}$ and a replay buffer $\mathcal{B}^k$ for each seen task k (Line 1-2). Next, we start the loop for sampling and training (Line 3). Specifically, the loop can be divided into two phases: a buffer filling phase (Line 4-18) and a parameter updating phase (Line 19-21).

Buffer Filling Phase. We start the loop for sampling in N parallel DRL environments (Line 4), which is denoted as Resource in Fig. 2. At the beginning of the loop, we choose a seen task k for current DRL environment E_i and obtain default initial state $\mathbf{s}_0^k$ for the specific task k (Line 5-6). Then the simulation process starts from state $\mathbf{s}_{init}^k$ to terminal state $\mathbf{s}_T^k$ for interacting with the environment and generating experiences (Line 7-14). After an episode completes, we collect the experiences generated above, and put all the information about the experiences into buffer $\mathcal{B}^k$ (Line 15-18).

Parameter Updating Phase. After collecting the training samples, we start the training process for each seen task k (Line 19). We optimize θ by Eqn. (1) with batch size M and learning rate l_r (Line 20-21).

Finally, for each unseen task j, FEAT executes an episode via trained policy (line23); then maps the formed trajectory into a feature subset for fast feature selection (line24).

For an unseen task, FEAT mitigates the damage of irrelevant information in the seen tasks through the combined action of the different representation of tasks and the learned internal policy in trained Q Network. Through the task representation embedded into environment state, the agent can distinguish which seen task is being learned by perceiving the state of the current environment. Thus, when an unseen task is received, FEAT can combine the representation of the new task and the learned internal policy to make a customized decision and tends to give a more similar feature selection solution for tasks with similar representation, so as to alleviate the interference of seen tasks that are not related to the unseen task.

Note that we introduce *max feature ratio* (denoted as mfr in line 10 in Algorithm 1) to limit the maximal number of

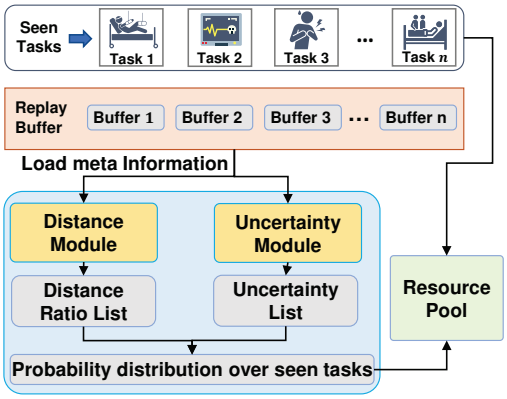


Fig. 3. Overview of Inter-Task Scheduler.

selected features. This is because the computing resources of analysis systems are limited and the number of optional features is also limited by the systems due to computing cost considerations. FEAT utilizes the limitation on the feature number during training and enables the global agent to find solutions in a trimmed smaller search space, which not only makes full use of the computing resources of the system, but also reduces the search space for the feature selection agent.

C. Inter-task Scheduler

For establishing a balance between the requirements of multiple tasks that are competing for the limited resources of the learning system, Inter-task Scheduler improves FEAT by assigning each seen task a probability of being selected based on the real-time learning situation dynamically, which corresponds to line 5 in Algorithm 1. The structure of the Inter-task Scheduler is shown in Fig. 3. The process of Inter-task Scheduler can be divided into two phases: *Information Collecting Phase* and *Probability Determination Phase*. Firstly, Inter-task Scheduler collects progress-related information (i.e., uncertainty and distance ratio) for each seen task to measure the learning needs of different tasks (Information Collecting Phase). Subsequently, Inter-task Scheduler dynamically allocates learning resources for each seen task by adjusting the probability for each seen task chosen by computing resources (Probability Determination Phase).

The process of the Information Collecting Phase *for each seen task* can be depicted by the following equations:

$$\psi_k^t = \text{load}(\mathcal{B}^k), \tag{4a}$$
$$\zeta_k^t = \text{dist}(\psi_k^t, k), \tag{4b}$$
$$\xi_k^t = \text{uncertainty}(\psi_k^t, k). \tag{4c}$$

Firstly, when Inter-task Scheduler is invoked at moment t, the load module is used to load recent n trajectories for seen task k from replay buffer $\mathcal{B}^k$. Each trajectory is mapped to a selected feature subset, and $\psi_k^t = \{F_1^k, ..., F_n^k\}$ is the set of all selected feature subsets. Performance of the feature subsets on seen task k generated from the n recent trajectories reflect the learning level of the current FEAT in feature selection for task k. Thus, we analyze these feature subsets mapped from the recent trajectories in the subsequent process.

Secondly, for estimating the potential promotion space of FEAT in feature selection, we propose the dist module to calculate distance ratio using the average performance of classifiers trained with each feature subset in ψ_k^t and the performance on same metric of a pre-trained classifier with all features in the dataset. We use the performance of the classifier trained with all features as a baseline for each seen task, and simultaneously consider current performance of FEAT in feature selection on each task and the baseline performance to evaluate the potential promotion on each seen task.

Next, the uncertainty module is introduced to evaluate the stability of FEAT for feature selection of each seen task. Better stability requires that the feature subsets in ψ_k^t tend to be more stable with the process of training and a feature selection solution with less randomness can be given in the end. The uncertainty module calculates the uncertainty through collecting and analysing the selection or deselection of each feature in each subset $F^k \in \psi_k^t$, which can easily evaluate whether the result of PA-FEAT is stable and provide a quantified result.

After collecting the progress-related information (i.e., distance ratio and uncertainty), Information Collecting Phase is finished and the collected progress-related information is processed in the Probability Determination Phase.

$$p_{1:k}^t = \text{output}(\zeta_{1:k}^t, \xi_{1:k}^t). \tag{5}$$

Specifically, the output module blends the distance ratio and uncertainty of each seen task and generates a probability of being selected by resources for each seen task.

Now, we give the definitions of *Distance Ratio* and *Uncertainty*, the core of the dist module and uncertainty module.

Definition 5 (Distance Ratio). Given a set of feature subsets $\psi_k^t = \{F_1^k, F_2^k, ..., F_n^k\}$ for training task k at moment t, Distance Ratio denoted as ζ_k^t is defined as Eqn. 6.

$$\zeta_k^t = \frac{P_{all}^k - P_{avg}^k}{P_{all}^k}, \tag{6a}$$
$$P_{avg}^k = \frac{1}{n}\sum\nolimits_{F_i^k \in \psi_k^t} P(F_i^k), \tag{6b}$$

where P_{all}^k represents the performance (i.e., F1-measure) of the classifier trained with all features on seen task k, P_{avg}^k represents the average performance of the classifiers trained with each feature subset $F_i^k \in \psi_k^t$ and the set of feature subset ψ_k^t is obtained in the load module.

Definition 6 (Performance Uncertainty). Given a set of feature subsets $\psi_k^t = \{F_1^k, F_2^k, ..., F_n^k\}$ for training task k at moment t, Performance Uncertainty ξ_k^t is defined as Eqn. 7.

$$\xi_k^t = 1 - \frac{1}{m}\sum\nolimits_{i=1}^{m} \left| \frac{1}{2} - p(i) \right|, \tag{7}$$

where $p(i)$ represents the probability that the i-th feature is selected by each subset $F \in \psi_k^t$ and m represents the number of all the features for task k.

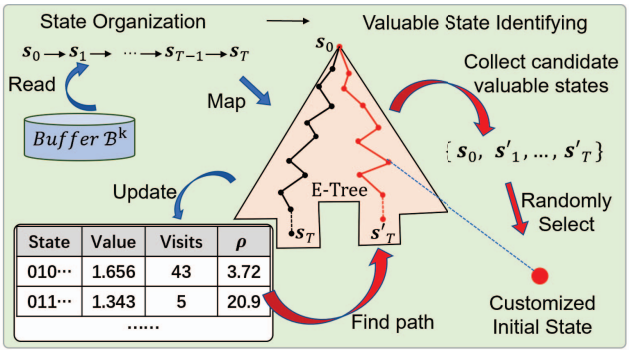


Fig. 4. Overview of Intra-task Explorer.

Now, we clarify the calculation in the output module. We first collect the scores of Distance Ratio $\{\zeta_1^t, \zeta_1^t, ..., \zeta_n^t\}$ and Performance Uncertainty $\{\xi_1^t, \xi_2^t, ..., \xi_n^t\}$ for all the seen tasks. Then, the output module integrates Distance Ratio with Performance Uncertainty by summing the two normalized scores at first and feeding the result into a softmax function (as shown in Eqn. 8).

$$d_k = \frac{\zeta_k^t}{\sum_{i=1}^n \zeta_i^t} + \frac{\xi_k^t}{\sum_{i=1}^n \xi_i^t}, \tag{8a}$$

$$D = [d_1, d_2, ..., d_n], \tag{8b}$$

$$p_{1:k}^t = softmax(D), \tag{8c}$$

where n represents the number of seen tasks, ζ_i^t represents the distance ratio of seen task i, and ξ_i^t represents the performance uncertainty of seen task i.

D. Intra-task Explorer

For sufficient exploration of environments corresponding to each seen task, we design Intra-task Explorer, an optimization for seeking trajectories with high quality and making full use of these experiences to enhance the training process. Fig.4 shows the structure of Intra-task Explorer. Intra-task Explorer integrates the past trajectories and organizes the visited states into a tree-based structure, called E-Tree. Based on E-Tree, Intra-task Explorer provides an evaluation mechanism for each state and can identity the states requiring further explore. Once Intra-task Explorer is called, this mechanism will be enabled to return the most exploratory state that has been visited. Then, that state will be restored and the agent will start to explore from that state guiding by the current Q-Network of the agent until the terminal state. The transition trajectory generated in this process will be collected for training PA-FEAT. Finally, all novel states encountered will be added to state organization and information attached to these states will be updated.

1) State Organization: In our fast feature selection, one major characteristic is that though the state space will be high-dimensional with the increasing of feature dimension, the action space is always discrete. This means that no matter which state the agent is visiting, the number of actions it can take is limited, namely select or deselect the corresponding feature. Thus, we yield an Experience Tree (E-Tree) (as shown in Fig.4) to organize the visited state.

TABLE I
CHARACTERISTICS OF DATASETS.

Dataset	#Instances	#Features	#Seen tasks	#Unseen tasks
Emotions	593	72	4	2
Water-quality	1060	16	7	7
Yeast	2417	103	7	7
Physionet2012	12000	41	12	17
Computers	12440	159	7	11
Mediamill	43910	120	7	9
Business	5192	520	7	5
Entertainment	4208	1020	7	5

Specifically, firstly, past trajectories are read from the corresponding buffer to the seen task to build E-Tree. Then, according to the state transition in the trajectories, E-Tree extends a new leaf node when a new state is visited in the trajectories. E-Tree will extend a leaf node every time a transition to a new state occurs until terminal state is coming.

2) Valuable State Identifying: When E-Tree begins to take shape, the main issue is how to identify the state that is worth further exploration from numerous visited states. Inspired by UCT, we design an initial state customization strategy for valuable-state searching, as illustrated in Fig.4. Specifically, when searching in E-Tree from root node, the child node $F^{'}$ which maximizes ρ in Eqn.9 will be visited in every selection.

$$\rho(F^{'}) = \hat{\mu}_{F^{'}} + \sqrt{\frac{c_e \ln(T_F)}{T_{F,F^{'}}}}, \tag{9}$$

where T_F denotes the number of times node F has been visited, $T_{F,F^{'}}$ is the number of times child node $F^{'}$ has been selected in node F , and $\hat{\mu}_{F^{'}}$ represents the value estimation for node $F^{'}$, which is formulated as the accumulation performance of the feature subset that mapped from the trajectories with the corresponding state.

After finding the valuable node based on Eqn.9, we argue that this state is associated with a better trajectory (higher-performing with as few features as possible). Thus, the agent is pushed to explore straightly from this valuable state to find better trajectories and the involved state transitions are used to train the Q-Network in PA-FEAT, meanwhile, these trajectories will be also used to update E-Tree. On the one hand, experience acquired in the training process fuels a trend where more and more information is available for building the E-Tree, which enhances the performance of the E-Tree. On the other hand, the discovery of trajectories with high quality through the E-Tree will enhance the training process and get a policy network with more effectiveness and robustness.

For each seen task, PA-FEAT maintains an E-Tree during training. Each time when ITE is invoked in certain environment, E-Tree for the corresponding seen task is quickly located and the valuable state is returned according to Eqn. 9. Subsequently, the environment is recovered to the valuable state and begins interactions with the local agent.

IV. PERFORMANCE EVALUATION

A. Experimental Setup

1) Datasets.: We use eight real datasets for evaluation, as shown in Table I. The datasets are available at Mulan[1] and PhysioNet Challenge 2012[2]. Detailed description of these datasets are as follows:

- **Emotions** contains 593 samples and 72 different characteristics. Each task in this dataset is to predict whether the song arouses a certain emotion.
- **Water-quality** consists of 16 physico-chemical characteristics, such as temperature and alkalinity, which are used to predict the quality of water of Slovenian rivers.
- **Yeast** contains 103 characteristics which describe microarray expressions and phylogenetic profiles for 2417 yeast genes. Each task is to predict whether the gene belongs to a functional category (e.g. Metabolism, energy, etc).
- **Physionet2012** contains medical index measurement records of 12,000 ICU stays (e.g. Glucose, etc). Each task is to predict patient outcomes (e.g. 3-day readmission, in-hospital death, Interval of SOFA score, etc).
- **Computers**, **Business** and **Entertainment** is from web pages linked from the "yahoo.com" domain. In these 3 datasets, each task is to predict whether the web page belongs to a certain subcategory.
- **Mediamill** contains international broadcast news data where each video instance is represented as a 120-dimensional feature vector of numeric features. Each task is to predict whether the video instance contains certain concept (e.g. People, Sky, Cloud, etc).

2) Baseline Methods: We compare PA-FEAT with three categories of baselines: (1) multi-task enhanced methods, which generalize knowledge from seen tasks to facilitate feature selection on unseen tasks or can be twisted to address multi-task enhanced feature selection setting, namely methods for comparing with ITS (i.e., PopArt [21]) and ITE (i.e., GO-Explore [25] and RR [23]) and multi-label methods GRRO-LS [14], Ant-TD [15] and MDFS [16]. (2) single-task feature selection methods, which learn from scratch for unseen tasks instead of utilizing information from seen tasks, namely Kbest [9], RFE [10], SADRLFS [33] and MARLFS [35]. (3) baselines with no feature selection, namely DNN and SVM.

- **GRRO-LS** [14] takes feature relevance, feature redundancy and label relevance into account based on information theory for multi-label feature selection.
- **Ant-TD** [15] is an Ant Colony Optimization (ACO) based multi-label feature selection method, use RL to get heuristic information for ants in ACO.
- **MDFS** [16] a multi-label feature selection method via manifold regularization, exploiting local label correlations shared by instances and global label correlations.
- **PopArt** [21] is based on MT-DRL and aims to balance learning over multiple seen tasks via adapting the contribution of each task to knowledge generalization according to reward magnitudes, which is implemented in our proposed framework FEAT and compared with our ITS.
- **Go-Explore** [25] aims to realize efficient exploration through using a simple policy to navigate in the searching space from customized initial states, which can provide valuable experiences for downstream learning methods (e.g. DRL or Imitation Learning). We implement it under FEAT for comparing with our ITE.
- **Reward Randomization (RR)** [23] is a state-of-the-art approach to realize efficient exploration of DRL-based methods through a novel reward randomization mechanism. We implement it with FEAT for comparing it with our ITE.
- **K-Best** [9] first ranks features by their mutual information with the label vector, and then selects the K features with the highest scores.
- **Recursive Feature Elimination (RFE)** [10] selects features by recursively selecting smaller and smaller feature subsets.
- **SADRLFS** [33] is a Single-Agent DRL based method for single-task feature selection, which trains an RL agent for each unseen task from scratch.
- **MARLFS** [35] is a single-task Multi-Agent RL based method and explores feature subsets by assigning an agent for each feature to decide selection for corresponding feature.
- **Deep Neural Network (DNN)** is a fully connected neural network trained with data without feature selection for each unseen task. For all networks, we use Adam Optimizer [37] with a learning rate searched in $\{1e^{-1}$, $1e^{-2}$, ..., $1e^{-5}\}$ and batch size in $\{32, 64, 128, 256\}$.
- **Support Vector Machine (SVM) [38]** is a SVM model trained for each unseen task using data without feature selection.

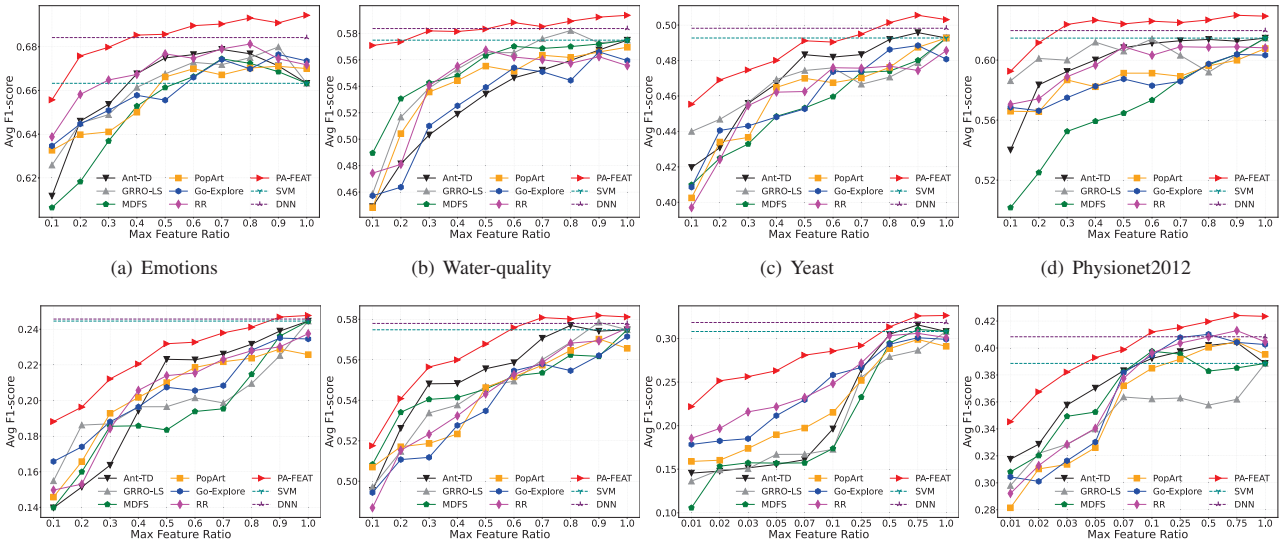
3) Evaluation Metrics: We use the performance of unseen tasks under the selected feature subsets to evaluate the ability of the feature selection methods. Specifically, we regard the widely used classification as the downstream data analysis task, and train an SVM [38] model for each unseen task using the corresponding selected feature subset to measure the quality of the feature subset. In this case, we use average F1-score (Avg F1-score) and average AUC (Avg AUC) among unseen tasks as two evaluation metrics.

- **Avg F1-score** evaluates the average performance of feature selection methods in terms of the F1-score, namely the harmonic mean of the precision and recall.
- **Avg AUC** evaluates the average performance of feature selection methods in terms of AUC, the area under the Receiver Operating Characteristics (ROC) curve.

4) Implementation Details.: For the sake of robustness, all reported results are averaged over 5 independent runs. In each run, we randomly select 70% instances from each dataset to form the training set for our proposal and other baselines, and use the remaining part for testing. As for the reward function mentioned in Eqn. (2), we use DNN as the $CLS(\cdot)$

[1]http://www.uco.es/kdis/mllresources/

[2]https://physionet.org/content/challenge-2012/1.0.0/



(a) Emotions (b) Water-quality (c) Yeast (d) Physionet2012

(e) Computers (f) Mediamill (g) Business (h) Entertainment

Fig. 5. Impact of max feature ratio over Avg F1-score.

and consider AUC as the $P(\cdot)$. We use Dueling DQN [32] to train the policy in FEAT and the loss function of Dueling DQN is shown as Eqn. 1a. In all the experiments, we use Pytorch 1.8.1 to implement our proposed model, and all the codes are run on Ubuntu 18.04.2 LTS with 8 GeForce RTX 3090 graphic cards.

TABLE II
COMPARISON ON AVERAGE ITERATION TIME DURING DURING AND AVERAGE EXECUTION TIME (IN SECONDS)

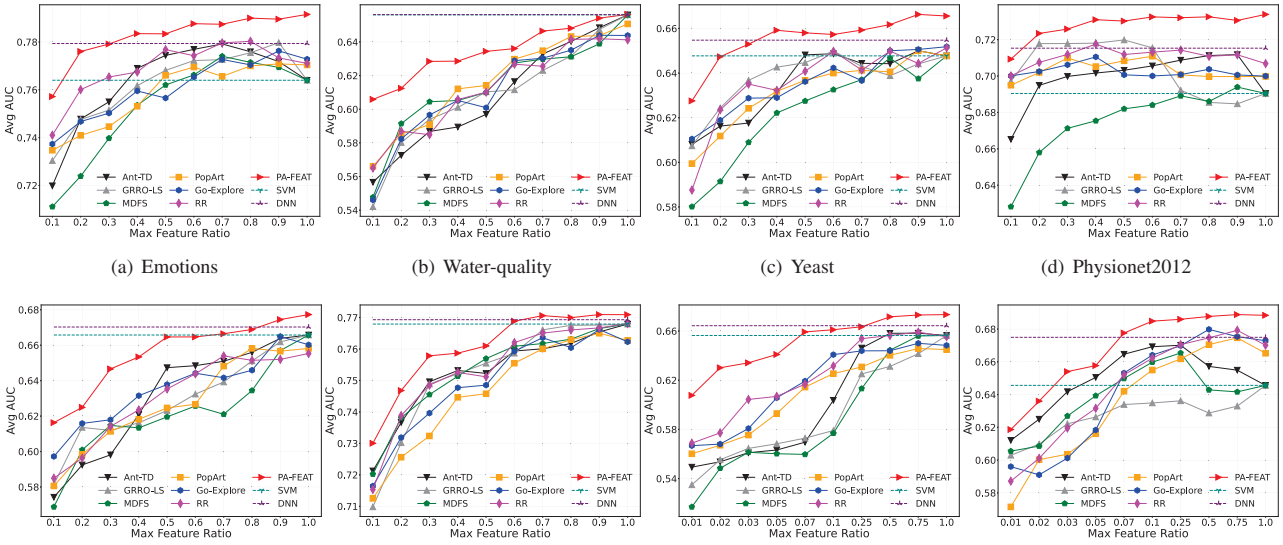
	PopArt		Go-Explore		RR		PA-FEAT	
	Iter	Exec	Iter	Exec	Iter	Exec	Iter	Exec
Emotions	0.596	0.0764	0.438	0.0712	0.853	0.0714	0.771	0.0713
Water-quality	0.338	0.0153	0.331	0.0132	0.685	0.0134	0.506	0.0132
Yeast	1.205	0.1174	0.651	0.1143	1.573	0.1143	1.441	0.1143
Physionet2012	0.686	0.0562	0.579	0.0547	1.117	0.0544	0.911	0.0541
Computers	1.313	0.2845	1.176	0.2800	1.974	0.2804	1.890	0.2804
Mediamill	3.026	0.4479	2.665	0.4423	4.027	0.4422	3.384	0.4423
Business	6.765	0.5889	4.902	0.5813	7.539	0.5820	7.214	0.5824
Entertainment	13.628	1.1660	10.690	1.1452	16.068	1.1481	14.378	1.1455

B. Comparison with Baselines

1) Comparison With multi-task enhanced Baselines: We compare PA-FEAT with six fast feature selection baselines and two baselines with no feature selection on eight real-world datasets on Avg F1-score and Avg AUC. Results are show in Fig. 5 and Fig. 6. We also examine the efficiency and compare the time cost of each iteration during training before unseen tasks arrive and the execution time of each method to provide feature selection for a single unseen task. The result is shown in Table II. We have the following findings.

From the figures, we observe that our proposed model consistently outperforms all the other feature selection baselines in terms of Avg F1-score and Avg AUC on all datasets. The reason is as follows. The goal of multi-label feature selection methods, including GRRO-LS, Ant-TD and MDFS, is to find an optimal feature subset for all currently seen tasks, thus cannot yield customized feature selection results when facing different unseen tasks. Although we extend these methods for unseen tasks by considering historical seen tasks and target unseen task at the same time, they still fail to fit the target unseen task since seen tasks with overwhelming numbers dominate the feature selection results. As for PopArt, reward magnitude is not accurate to measure the learning difficulty of each task when there is a large reward variance between tasks. Go-Explore decouples exploration from exploitation, which weakens the role of appropriate initial states. RR ignores that information more valuable than reward randomization can be obtained from exploration experiences, thus, improvement brought by adding randomness to reward is far less reliable than in-depth analysis and utilization of existing experience. On large datasets Business and Entertainment, PA-FEAT also consistently outperforms all baselines. When increasing the max feature ratio, performance of all methods increase significantly in the early stage and levels off later. This is because datasets with large number of features may have more redundancy, which leads to fewer performance gains when the number of features used has reached a certain number. Compared with baselines without feature selection, PA-FEAT can find feature subsets with better performance on Avg F1-score and Avg AUC on all datasets. On Physionet2012 dataset, PA-FEAT outperforms baseline SVM and DNN in terms of Avg F1-score and Avg AUC using less than 30% of the raw features. This confirms the effectiveness of feature selection.

When increasing the value of max feature ratio, the Avg F1-score and Avg AUC of PA-FEAT will go up first and then get saturated gradually on all eight datasets. However, we observe that the performance of all baselines will rise first and then decline on some datasets. Use Emotions dataset as an example, as shown in Fig. 5(a) and Fig. 6(a), the performance of our model keeps rising steadily when the limitation on the



(a) Emotions (b) Water-quality (c) Yeast (d) Physionet2012

(e) Computers (f) Mediamill (g) Business (h) Entertainment

Fig. 6. Impact of max feature ratio over Avg AUC.

TABLE III
ABLATION STUDY (MAX FEATURE RATIO = 1).

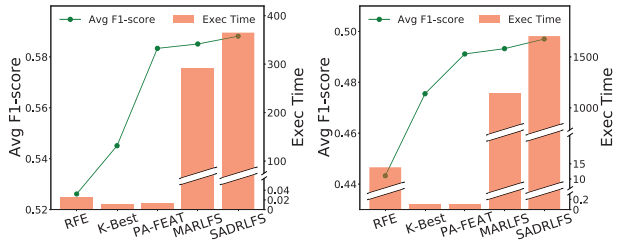
	Emotions		Water-quality		Yeast		Physionet2012		Computers		Mediamill		Business		Entertainment	
	F1-score	AUC	F1-score	AUC	F1-score	AUC	F1-score	AUC	F1-score	AUC	F1-score	AUC	F1-score	AUC	F1-score	AUC
Ours: PA-FEAT	**0.6944**	**0.7915**	**0.5937**	**0.6564**	**0.5031**	**0.6656**	**0.6293**	**0.7339**	**0.2478**	**0.6773**	**0.5812**	**0.7709**	**0.3264**	**0.6733**	**0.4234**	**0.6883**
Ours w/o ITS	0.6798	0.7788	0.5691	0.6489	0.4903	0.6541	0.6152	0.7233	0.2443	0.6654	0.5737	0.7671	0.3006	0.6556	0.4023	0.6609
Ours w/o PE	0.6767	0.7756	0.5683	0.6434	0.4853	0.6520	0.6142	0.7219	0.2412	0.6656	0.5750	0.7674	0.3109	0.6611	0.4118	0.6664
Ours w/o ITE	0.6729	0.7738	0.5678	0.6416	0.4786	0.6432	0.6115	0.7214	0.2294	0.6537	0.5684	0.7657	0.2964	0.6509	0.3893	0.6545
Ours w/o ITS, ITE	0.6633	0.7640	0.5522	0.6265	0.4602	0.6229	0.6053	0.7194	0.2172	0.6435	0.5424	0.7512	0.2895	0.6288	0.3771	0.6453

maximum number of selectable features is gradually relaxing. While the performance of baselines starts to deteriorate when the value of max feature ratio exceeds 0.8. This is because baselines without FEAT for assistance lack a mechanism to adjust the learning process based on max feature ratio, thus can only give the largest feature subset whose number of features does not exceed the limit of max feature ratio.. Three other baselines implemented with FEAT, including PopArt, Go-Explore and RR, as mentioned above, have some defects in multi-task scheduling or effective exploration. Therefore, an excessively large search space due to the big max feature ratio will make the performance of them get worse.

All methods under FEAT framework (i.e., PA-FEAT, PopArt, Go-Explore and RR) need to carry out model training before the arrival of unseen tasks for quick response to unseen tasks with well-trained Q-network. In contrast to methods with FEAT, Ant-TD, GRRO-LS and MDFS consider both the newly coming unseen task and seen tasks and can't make any preparations before unseen tasks come, thus, the three multi-label methods do not require training process but require more calculations after receiving unseen tasks. We can observe that the time cost of each training iteration for methods under FEAT framework is similar and related to the size of the dataset, especially the number of raw features. This is because the more features, the more steps to scan for agents in each episode of RL and the more time consumed. We can see that with similar time consumption, our method achieves better performance. For all methods, We train for $2,000$ iterations that reach complete convergence.

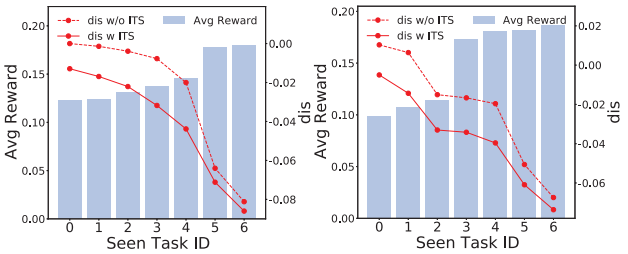
The average execution time of our method is less than half a second across all datasets, which ensures the applicability of our approach in practice. For all methods implemented with our proposed framework FEAT, including PA-FEAT, PopArt, Go-Explore and RR, the average execution time is nearly the same. This is because methods implemented with FEAT all follow the pattern that Q-network training is carried out before the unseen task arrives and the learned policy in well-trained Q-network is directly used after the unseen task arrives. The difference between these methods is different multi-task scheduling or search efficiency enhancement strategy during training process which is compared with our ITS and ITE respectively, while all the execution processes of these methods for unseen tasks are environment initialization and Q-network inference. PopArt takes a little more time than others with FEAT, because of an additional DNN layer to realize target rescaling. We do not show the average execution time of multi-label selection methods here, because when extending these methods' ability for unseen tasks, they take a very long time (up to thousands of seconds) to generate feature subsets, which is several orders of magnitude larger than other methods.

2) Comparison with single-task feature selection baselines: To further discuss the ability of PA-FEAT to feature selection,



(a) Water-quality (b) Yeast

Fig. 7. Comparison on average execution time (in seconds) and Avg F1-score.



(a) Water-quality (b) Yeast

Fig. 8. Average reward and distance ratio on each seen task.

we compare PA-FEAT with two traditional methods and two DRL based methods for single-task feature selection. Different from the multi-task setting in PA-FEAT, these four methods conduct feature selection for single task and can only learn from scratch for unseen task after it arrives instead of utilizing any information from seen tasks. So they are not suitable for fast feature selection scenario, especially when the training time is too long to tolerate. We compare PA-FEAT the prediction performance and execution time for unseen tasks. The results are shown in Fig. 7. Due to the limited space, here we only show the results on two datasets, i.e., Water-quality and Yeast and only report the results on Avg F1-score. The results on other datasets and Avg AUC share the same trend.

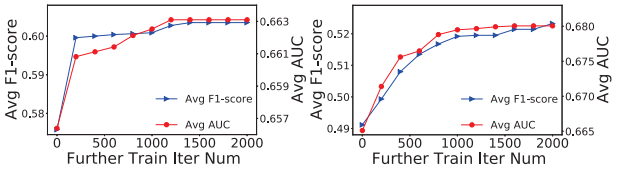
We can observe that compared with SADRLFS and MARLFS, the response speed for unseen tasks of PA-FEAT is highly remarkable and a little less competitive for performance on Avg F1-score. Although compared with PA-FEAT, SADRLFS and MARLFS achieve no more than 1.1% gains on Avg F1-score, their required time at least $9,991$ times that of PA-FEAT on Yeast dataset. This is because SADRLFS and MARLFS train a feature selection policy for each unseen task from scratch separately. Before giving a feature selection result for a newly coming unseen task, SADRLFS needs to complete time-consuming network training. The training process of MARLFS is even more complex than SADRLFS, because the complexity of the MARLFS increases with the number of agents and each agent has to maintain its own policy network, training strategy and memory storage. Thus, though SADRLFS and MARLFS achieve better results in optimal feature subset exploration, they have to take much longer time to yield a feature selection result for a new coming task, which is intolerable in time-sensitive applications. K-Best only concentrates on the relevance between each feature and the specific task, ignoring the dependencies and redundancies between features, thus although it is computationally efficient and consumes slightly less time than PA-FEAT, PA-FEAT achieves much more reliable feature selection results with comparable time cost. When response to an unseen task, K-Best ranks features by their mutual information with the label, while PA-FEAT calculates the Pearson Correlation Coefficient between each feature and label to form task representation. Time complexity of mutual information and Pearson correlation coefficient calculation are all $O(n)$ (n denotes the number of features), and the time cost of DNN inference in PA-FEAT is negligible compared with the former two operations, thus the execution times of PA-FEAT and K-Best are similar. RFE, as a wrapper-based method, tries to fit a specific predictive model from scratch for each unseen task by gradually removing the weakest feature, thus requires significantly more time than PA-FEAT. But due to subject to the strong structured assumptions of the used predictive model, the selected feature subset normally cannot compatible with other predictive models.

C. Ablation Study

The ablation study is performed by gradually removing two key components of PA-FEAT, i.e., Inter-Task Scheduler (ITS) and Intra-Task Explorer (ITE). As shown in Table III, complete PA-FEAT achieves an average of 3.49% and 1.89% higher performance than that of PA-FEAT w/o ITS on the eight datasets in terms of Avg F1-score and Avg AUC. The improvement confirms that dynamically allocating resources according to the learning progress of each task indeed improves the effectiveness of multi-task learning. Also, the complete model PA-FEAT performs better than PA-FEAT w/o ITE, as Avg F1-score and Avg AUC increase by 5.62% and 2.84% on average for eight datasets. This shows that it is effective to search from the customized initial state based on our proposed E-Tree. Finally, Avg F1-score and Avg AUC of the complete model PA-FEAT are 8.97% and 4.86% higher than that of PA-FEAT w/o both ITS and ITE on average, which confirms the benefits of putting ITS and ITE together.

To clarify the performance gains from the policy exploitation (PE) in ITE, we add a variant by removing the exploitation of the learned policy in ITE from the complete model PA-FEAT (denoted as ours w/o PE in Table III), which means that the experience data used to construct E-Tree is derived under a random policy. As shown in Table III, the complete model PA-FEAT yields better performance than PA-FEAT w/o PE, where Avg F1-score and Avg AUC increase by 3.10% and 1.89% on average for eight datasets. This indicates that utilizing learned RL policy to enhance E-Tree is effective.

In order to further study the benefits brought by our proposed ITS module, we examine the average reward of each seen task at the later stage of training and calculate the distance ratio of each seen task with and without ITS. The rewards generated during the later stage of training are stable enough, thus we can take the average of the rewards and use them to measure each task's learning difficulty. The lower the average reward, the higher the difficulty of the task. We



(a) Water-quality (b) Yeast

Fig. 9. Further Train on Unseen Tasks.

study the benefits of ITS for tasks with different difficulties, as shown in Fig. 8. We can observe that the improvement brought by ITS is more obvious on difficult tasks, and this improvement gradually becomes stronger as the difficulty of the task increases. This shows that dynamically allocating learning resources according to task learning progress makes multi-task learning more balanced, since it improves the learning effectiveness on tasks that were previously difficult to master.

D. Further Train on Unseen Tasks

PA-FEAT can also achieve a better result through further training when the time budget is relatively ample in practice. When more time is acceptable for calculation, an DRL environment is initialized for the unseen task at first and then the agent of PA-FEAT interacts with the corresponding environment. The experience generated during the interaction is collected and used for training Q-network. In further training process, PA-FEAT explores the feature space of each unseen task under the given time budget, attempting to obtain a more suitable agent for the unseen task based on the result of the knowledge generalization process.

To illustrate the effectiveness, we further train each unseen task for $2,000$ iterations, which takes an average of 1.68 seconds per 100 iterations. The performance growth curves in terms of Avg F1-score and Avg AUC are shown in Fig. 9. With training for more iterations, Avg F1-score and Avg AUC will go up first and get saturated gradually after. This is because the algorithm has reached convergence and more iterations have little effect on the results after convergence.

V. Related Work

Feature Selection. Feature selection methods can be divided into single-label and multi-label methods.

Single-label feature selection methods can be categorized into filter, wrapper and embedded methods. Filter methods [9], [39]–[41] are fast, simple and efficient, but ignore the interactions between feature selection and the subsequent predictors. Wrapper methods [10], [42]–[44] consider the learning algorithms as evaluation function and require massive amounts of computation. Embedded methods [6], [45] embed feature selection into the training phase of learning algorithms and could achieve supreme performance with the combined predictors, but not very compatible with other predictors. Traditional single-label feature selection methods can be subject to various strong assumptions. For example, K-Best [9] ignores the dependencies and redundancies between features. RFE [10] relies on the predictive model used in downstream tasks, which leads to limited generalizability to other predictive models.

Multi-label feature selection methods, including MDFS [16], GRRO-LS [14], Ant-TD [15], LLSF [46] and LRFS [47], can select informative features from multi-label data directly and attempt to avoid the loss of label information. Although these methods always take label relevance into consideration, they handle the balance between multiple labels by evaluating feature importance through simply adding up the coefficient corresponding to each label, which cannot estimate accurately the feature importance for unseen labels.

Balance in Multi-Task Learning. Gradnorm [20] normalizes the gradients from different tasks according to gradient similarity for balancing multi-task losses. But optimization relying on the gradient similarity is usually unstable, especially when there is a large gradient variance within each task itself. A more recent work [21] studies the problem of parallel learning of multiple sequential decision tasks and weights different tasks by their reward magnitudes. However, the above methods pay more attention to adjusting the contribution of each task to the update of the model, but rarely study how to divide the limited learning resources among the tasks.

Search Efficiency for DRL. DRL-based feature selection methods [11]–[13] are superior in optimal feature subset exploration due to their powerful global search ability. However, they usually require time-consuming calculation process to obtain feature selection results. Many existing works [22]–[25] study how to make DRL-based methods more efficient. ICM [22] models curiosity, which serves as an intrinsic reward signal, to enable the DRL agent to obtain more knowledge when exploring the search space. A more recent work [23] introduces a reward randomization mechanism to drive exploration diversity. Go-Explore [24] and its extension [25] attempt to find the experiences with high-quality by exploring from the appropriate initial states, but decouple the exploration from the exploitation of the learning policy.

VI. Conclusion

In this paper, we propose a novel progress-aware Multi-Task Deep Reinforcement Learning (MT-DRL) based method, called PA-FEAT, for fast feature selection over structured data. It consists of a basic framework for knowledge generalization and transfer based on MT-DRL, called "FEAT", a dynamic inter-task resource scheduler called "Inter-Task Scheduler" which can dynamically allocate resources to ensure balanced learning over multiple historical tasks, and a tree-based intra-task optimizer called "Intra-Task Explorer" to enable efficient search over large feature space for each task. We conduct extensive experiments on eight real-world datasets, and the results confirm the effectiveness of our proposed model.

Acknowledgment

This work is supported by National Key Research and Development Program of China (2020YFB1708100), National Natural Science Foundation of China (62072033) and the Ant Group.

REFERENCES

[1] S. Li, L. Chen, and A. Kumar, "Enabling and optimizing non-linear feature interactions in factorized linear algebra," in *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*. ACM, 2019, pp. 1571–1588.

[2] M. A. Khamis, H. Q. Ngo, X. Nguyen, D. Olteanu, and M. Schleich, "Learning models over relational data using sparse tensors and functional dependencies," *ACM Trans. Database Syst.*, vol. 45, no. 2, pp. 7:1–7:66, 2020.

[3] M. Nikolic, H. Zhang, A. Kara, and D. Olteanu, "F-IVM: learning over fast-evolving relational data," in *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. ACM, 2020, pp. 2773–2776.

[4] Z. Luo, S. H. Yeung, M. Zhang, K. Zheng, L. Zhu, G. Chen, F. Fan, Q. Lin, K. Y. Ngiam, and B. C. Ooi, "Mlcask: Efficient management of component evolution in collaborative data analytics pipelines," in *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 2021, pp. 1655–1666.

[5] P. M. Narendra and K. Fukunaga, "A branch and bound algorithm for feature subset selection," *IEEE Transactions on computers*, vol. 26, no. 09, pp. 917–922, 1977.

[6] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[7] T. Xu, D. Wang, and G. Liu, "Banian: A cross-platform interactive query system for structured big data," *Tsinghua Science and Technology*, vol. 20, no. 1, pp. 62–71, 2015.

[8] R. Khan and M. Gubanov, "Weblens: towards interactive large-scale structured data profiling," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 3425–3428.

[9] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), Nashville, Tennessee, USA, July 8-12, 1997*. Morgan Kaufmann, 1997, pp. 412–420.

[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*, vol. 83, no. 2, pp. 83–90, 2006.

[11] 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.

[12] S. M. H. Fard, A. Hamzeh, and S. Hashemi, "Using reinforcement learning to find an optimal set of features," *Computers & Mathematics with Applications*, vol. 66, no. 10, pp. 1892–1904, 2013.

[13] 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.

[14] J. Zhang, Y. Lin, M. Jiang, S. Li, Y. Tang, and K. C. Tan, "Multi-label feature selection via global relevance and redundancy optimization," in *IJCAI*, 2020.

[15] M. Paniri, M. B. Dowlatshahi, and H. Nezamabadi-pour, "Ant-td: Ant colony optimization plus temporal difference reinforcement learning for multi-label feature selection," *Swarm Evol. Comput.*, vol. 64, p. 100892, 2021.

[16] J. Zhang, Z. Luo, C. Li, C. Zhou, and S. Li, "Manifold regularized discriminative feature selection for multi-label learning," *Pattern Recognit.*, vol. 95, pp. 136–150, 2019.

[17] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[18] A. Kumagai, T. Iwata, and Y. Fujiwara, "Transfer metric learning for unseen domains," in *Data Science and Engineering*, vol. 5, 2020, pp. 140–151.

[19] 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," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[20] Z. Chen, V. Badrinarayanan, C. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, vol. 80. PMLR, 2018, pp. 793–802.

[21] M. Hessel, H. Soyer, L. Espeholt, W. Czarnecki, S. Schmitt, and H. van Hasselt, "Multi-task deep reinforcement learning with popart," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 3796–3803.

[22] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, vol. 70. PMLR, 2017, pp. 2778–2787.

[23] Z. Tang, C. Yu, B. Chen, H. Xu, X. Wang, F. Fang, S. S. Du, Y. Wang, and Y. Wu, "Discovering diverse multi-agent strategic behavior via reward randomization," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[24] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune, "Go-explore: a new approach for hard-exploration problems," *arXiv preprint arXiv:1901.10995*, 2019.

[25] Ecoffet, Adrien and Huizinga, Joost and Lehman, Joel and Stanley, Kenneth O and Clune, Jeff, "First return, then explore," *Nature*, vol. 590, no. 7847, pp. 580–586, 2021.

[26] Z. Luo, S. Cai, G. Chen, J. Gao, W. Lee, K. Y. Ngiam, and M. Zhang, "Improving data analytics with fast and adaptive regularization," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 2, pp. 551–568, 2021.

[27] Z. Yang, C. Yang, F. Han, M. Zhuang, B. Yang, Z. Yang, X. Cheng, Y. Zhao, W. Shi, H. Xi, H. Yu, B. Liu, Y. Pan, B. Yin, J. Chen, and Q. Xu, "Oceanbase: A 707 million tpmc distributed relational database system," *Proc. VLDB Endow.*, vol. 15, no. 12, pp. 3385–3397, 2022.

[28] S. Ö. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 6679–6687.

[29] M. Cvitkovic, "Supervised learning on relational databases with graph neural networks," *CoRR*, vol. abs/2002.02046, 2020.

[30] S. Rendle, "Scaling factorization machines to relational data," *Proc. VLDB Endow.*, vol. 6, no. 5, pp. 337–348, 2013.

[31] S. Cai, K. Zheng, G. Chen, H. V. Jagadish, B. C. Ooi, and M. Zhang, "Arm-net: Adaptive relation modeling network for structured data," in *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*. ACM, 2021, pp. 207–220.

[32] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, ser. JMLR Workshop and Conference Proceedings, vol. 48. JMLR.org, 2016, pp. 1995–2003.

[33] 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 *20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020*. IEEE, 2020, pp. 871–880.

[34] K. Malialis, J. Wang, G. Brooks, and G. Frangou, "Feature selection as a multiagent coordination problem," *CoRR*, vol. abs/1603.05152, 2016.

[35] 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, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 2019, pp. 207–215.

[36] J. L. Rodgers and W. A. Nicewander, "Thirteen ways to look at the correlation coefficient," *The American Statistician*, vol. 42, pp. 59–66, 1988.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[38] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, 2011.

[39] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *J. Mach. Learn. Res.*, vol. 3, pp. 1289–1305, 2003.
[40] 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.
[41] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proceedings of the Twentieth International Conference on Machine Learning, (ICML 2003), August 21-24, 2003, Washington, DC, USA*. AAAI Press, 2003, pp. 856–863.
[42] P. M. Narendra and K. Fukunaga, "A branch and bound algorithm for feature subset selection," *IEEE Trans. Computers*, vol. 26, no. 9, pp. 917–922, 1977.
[43] 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, Boston, MA, USA, August 20-23, 2000*. ACM, 2000, pp. 365–369.
[44] J. Yang and V. G. Honavar, "Feature subset selection using a genetic algorithm," *IEEE Intell. Syst.*, vol. 13, no. 2, pp. 44–49, 1998.
[45] V. Sugumaran, V. Muralidharan, and K. I. 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, pp. 930–942, 2007.
[46] J. Huang, G. Li, Q. Huang, and X. Wu, "Learning label-specific features and class-dependent labels for multi-label classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, pp. 3309–3323, 2016.
[47] P. Zhang, G. Liu, and W. Gao, "Distinguishing two types of labels for multi-label feature selection," *Pattern Recognit.*, vol. 95, pp. 72–82, 2019.