2024

Data-Centric AI: Taming AI-ready Feature Space from Decision-Making to Generative-AI Perspectives

Dongjie Wang
University of Central Florida

DATA-CENTRIC AI: TAMING AI-READY FEATURE SPACE FROM DECISION-MAKING TO GENERATIVE-AI PERSPECTIVES

by

DONGJIE WANG
B.S. Sichuan University, 2016
M.S. Southwest Jiaotong University, 2019

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2024

Major Professor: Yanjie Fu

ABSTRACT

Unlike humans, AI systems are brittle and not robust. They often struggle when faced with novel situations, and are highly sensitive to small perturbations, which can lead to catastrophically poor performance. These systems comprise two main components: the model and the data. In recent decades, research has primarily focused on models, emphasizing advanced structures or algorithms to enhance AI performance. However, the data-centric aspect consumes most of the time and resources of human experts and greatly influences AI systems. Furthermore, the gains from the model-centric part are reaching a plateau. Thus, I shifted my research focus toward data-centric AI in order to identify the ideal feature space for preparing the AI-readiness of data. To realize this, this dissertation introduces two main research perspectives and the corresponding frameworks: 1) the decision-making perspective and 2) the generative AI perspective. The decision-making perspective formulates feature selection and feature generation as Markov decision-making processes. Within this perspective, reinforcement learning is used to develop practical frameworks due to its proficiency in optimizing such processes. Specifically, for feature selection, a single agent is employed to determine the selection of individual features in an iterative manner. For feature generation, a cascading reinforced agent structure is proposed to select candidate features and operations for generating new features. The generative AI perspective assumes that the knowledge derived from discrete feature learning records can be effectively integrated into a continuous space. This integration facilitates the exploration of an optimal feature space inspired by the successes of generative AI techniques. Thus, a unified framework is proposed to optimize both tasks, which has four key steps: data collection, continuous space construction, enhanced embedding search, and feature space reconstruction. The effectiveness of both perspectives underscores the potential for building up foundation models in the data-centric AI domain.

I dedicate this dissertation to my family.

ACKNOWLEDGMENTS

I am profoundly grateful for the invaluable assistance and support I received throughout my Ph.D. journey.

First and foremost, I extend my sincere thanks to my Ph.D. advisor, Prof. Yanjie Fu. His unwavering support and guidance have been instrumental in my academic growth. His enthusiasm for research, commitment to professionalism, and dedication to science have been a significant source of inspiration, motivating me to aim higher in my research endeavors. His approach to critical thinking and his innovative solutions to complex problems have enriched not only my academic pursuits but also my personal development.

I am also grateful to my dissertation committee members: Prof. Liqiang Wang, Prof. Rui Xie, and Prof. Chen Chen. Their insightful feedback and professional advice have significantly enhanced the quality of my research.

My appreciation extends to all my co-authors for their collaborative efforts. I am especially thankful to my internship mentors, Dr. Zhengzhang Chen, Dr. Jin Cao, and Dr. Lingfei Wu, for their extensive support in various facets of my professional growth. The lessons learned about time management and industry coordination have been invaluable, offering a perspective distinct from the academic environment. I am equally grateful to my fellow labmates: Kunpeng Liu, Pengyang Wang, Pengfei Wang, Meng Xiao, and Wei Fan, for their invaluable friendship and the shared journey of learning and growth.

Lastly, my deepest gratitude goes to my parents. Their unwavering love, understanding, constant encouragement, and selfless support have been and will always remain the cornerstone of my strength and perseverance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Motivation

Artificial Intelligence (AI) has profoundly revolutionized various areas, such as transportation, urban planning, economy, etc. The typical AI development process includes three key phases: i) data collection; ii) data representation (a.k.a. feature space); and iii) machine learning (ML) model construction. Creating a high-quality feature space is crucial, as it enhances the measurement of distances and discriminative patterns, thereby improving the structural and predictive aspects of data for more effective AI applications. There has been considerable research focusing on data-centric AI to achieve these improvements. This dissertation addresses two primary tasks within this domain: feature selection, which enhances the feature space by eliminating redundant features, and feature generation, which augments the feature space through mathematically transforming original features. However, existing methods are limited by: 1) Requiring extensive domain knowledge and human intervention, reducing their flexibility across different domains; 2) Tending to produce latent and untraceable representations, which hinders further analysis of the feature space; 3) Exponentially growing number of combinations as the dimensionality of the feature space increases, making these methods unsustainable for high-dimensional feature spaces. These challenges bring a key question: How can we autonomously develop an effective, traceable, and interpretable feature space? To address this, I introduce the novel concept of ”Transformation Learning”, comprising ’feature generation learning’ and ’feature selection learning’. This research presents two innovative research perspectives to reformulate the two tasks: 1) The Decision-Making Perspective, which views them as discrete Markov decision-making processes, employing reinforcement learning to develop practical frameworks; 2) The Generative-AI Perspective, which integrates feature learning knowledge into an embedding space, facilitating the search for an enhanced feature space.

Research Gap in Existing Literature

This dissertation includes two main topics: automated feature generation and automated feature selection. We will first review the existing works in these areas. Then, we will identify the research gaps and limitations of them.

Automated Feature Generation can enhance the feature space by automatically mathematically transforming the original feature space [13, 51, 13]. Existing works can be divided into three categories: 1) expansion-reduction based approaches [40, 45, 52, 38, 43]. Those approaches first expand the original feature space by explicitly [42] or greedily [16] decided mathematical transformation, then reduce the space by selecting useful features. However, it is hard for them to produce or evaluate both complicated and effective mathematical compositions, leading to inferior transformation performance. 2) evolution-evaluation approaches [44, 88, 112, 101]. These methods iteratively generate effective features and keep the most useful ones until achieving the maximum iteration number. Evolutionary algorithm models optimize the entire process. However, they still focus on how to simulate the discrete decision-making process in feature engineering. Thus, they are still time-consuming and unstable. 3) Auto ML-based approaches [12, 113]. Auto ML aims to find the most suitable model architecture automatically [18, 57, 35, 41]. The success of auto ML in many area [110, 100, 3, 92] and the similarity between auto ML and AFT inspire researchers to formulate AFT as an auto ML task to resolve.

Automated Feature Selection can be widely categorized into three types, i.e., filter methods, wrapper methods, and embedded methods [62, 111, 11, 75]. Filter methods rank features only by relevance scores and only top-ranking features are selected. The representative filter method is the univariate feature selection [21] and correlation based feature selection [108]. Filter methods are very fast and thus they are efficient on high-dimensional datasets. The representative wrapper methods are branch and bound algorithms [67, 48]. Wrapper methods are supposed to achieve

better performance than filter methods since they search on the whole feature subset space. Evolutionary algorithms [105, 46] low down the computational cost but can 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 [85] and decision tree [82]. Embedded methods could have supreme performance on the incorporated predictors, but are normally not very compatible with other predictors.

Research Gaps: Regarding automated feature generation, current approaches face limitations: i) Overlooking the heterogeneity between various feature pairs; ii) Experiencing exponential growth in time complexity with an increase in exploration steps; iii) Struggling to maintain consistent feature generation performance across diverse scenarios. Regarding automated feature selection, existing methods are limited by: i) Exponential growth in time complexity as the dimensionality of the feature space increases; ii) Limited generalization ability for cross-domain feature sets; iii) Inconsistent performance in feature selection across different scenarios.

Research Thinking and Project Aims

To address these gaps, this dissertation aims to propose automated, robust, interpretable, and traceable feature transformation approaches to enhance the input of AI models. Thus, we revisit the feature generation and feature selection tasks from three sides:

Optimization Thinking: The process of feature generation is iteratively selecting candidate features and applying mathematical operations to create new, informative features that enhance the performance of downstream ML tasks. Similarly, the procedure of feature selection refers to the iterative identification and selection of optimal feature subsets with the same objective. Both processes can be formulated as discrete decision-making processes. There are two research questions: i) How can we automatically optimize the two tasks to obtain better feature space? ii) Given the

similarities in the principle of the two tasks, is it possible to develop a unified optimization framework that effectively addresses both?

Framework Thinking: To establish practical frameworks for feature generation and feature selection, it is significant to construct concrete components, addressing the following aspects: i) The criteria that ought to be employed for selecting candidate features and mathematical operations. ii) The approach to modeling the knowledge inherent in feature learning to enhance the quality of the feature space. iii) The specific learning objective of the framework, whether it is centered on feature selection or feature generation. iv) The generalization way of the feature selection and generation frameworks when confronted with different underlying downstream ML tasks.

Computational Thinking: The third research thinking is from the computational perspective, which is essential in assessing the practical applicability of the proposed feature selection and generation frameworks in real-world scenarios. The significant research question is this: i) How can we enhance the computational efficiency of these frameworks, particularly when dealing with high-dimensional feature spaces?

This dissertation focuses on developing automated feature transformation systems. They are designed to enhance the feature space, aiming to improve the downstream ML tasks' performance and make the feature space easier to understand and interpret.

Organization

To tackle these research thinking questions, we propose two main research perspectives for reformulating feature generation and feature selection: i) A Decision-making perspective, where we treat these tasks as discrete Markov decision-making processes and use reinforcement learning to find the best solutions. ii) A Generative-AI Perspective, which combines both tasks into one

optimization system and uses generative AI approaches to model feature learning knowledge.

Specifically, this dissertation can be categorized into two thrusts:

- Thrust 1: Traceable Feature Space Refinement: A Decision-Making Perspective.

  Regarding feature generation learning, we examine the feature engineering workflow employed by human experts, identifying it as an iterative process. A cascading reinforced agent structure is proposed for selecting candidate features and operations. This structure comprises three agents: two dedicated to selecting candidate features and one focused on choosing candidate mathematical operations. Utilizing these selections, novel and informative features are generated and incorporated into the feature space for the next iterative updates. The primary objective of this process is to optimize both the performance of downstream machine learning tasks and the utility score of the features.

  Regarding feature selection learning, we formulate the feature selection task as a Markov discrete decision-making process. We introduce a single-agent Monte Carlo-based Reinforced Feature Selection (MCRFS) method, accompanied by two strategies for efficiency enhancement: an early stopping strategy and a reward-level interactive strategy. More specifically, we first develop a behavior policy to navigate through the feature set for generating training data. This training data is then used to evaluate and enhance the target policy through the Bellman equation. After that, we employ incremental importance sampling with the early stopping strategy to heighten training efficiency by discarding skewed data. This strategy entails getting the behavior policies traversal, with the cessation probability inversely related to the importance of the sampling weight. Moreover, we introduce a reward-level interactive strategy to augment training efficiency by the integration of reward-level external advice.

- Thrust 2: Traceable Feature Space Refinement: A Generative-AI Perspective. Feature selection and feature generation can both be formulated as discrete decision-making processes.

This formulation naturally leads us to propose a unified optimization framework to integrate the two tasks. We expect that such integration can alleviate the inherent complexities in modeling, potentially enhancing the performance of both tasks. Inspired by the success of ChatGPT, this dissertation assumes that extensive knowledge in feature engineering can be effectively embedded into a continuous embedding space. This strategic compression can facilitate a more efficient search for the optimal feature space. Thus, we propose a unified framework with four key steps: 1) Sequential training data collection; 2) Deep sequential feature knowledge embedding; 3) Gradient-steered better embedding search; 4) Autoregressive-based feature space reconstruction. More specifically, in Step 1, we can employ the self-optimizing feature selection or generation learning framework to collect sufficient feature-accuracy pairs as training data. In Step 2, we propose an encoder-evaluator-decoder model structure to preserve the knowledge in the training data into a distinguishable continuous embedding space. In Step 3, we select local optimal embeddings based on the model performance. Then, we move these embeddings along the direction of maximizing the downstream task performance to get enhanced ones. In Step 4, these enhanced embeddings are input into the well-trained decoder to generate the feature space in an autoregressive manner. We output the feature space with the highest downstream task performance as the optimal feature generation or selection result.

In the following chapters, this dissertation delves into two distinct perspectives. Chapter 2 and Chapter 3 focus on the decision-making perspective. I will explore self-optimizing feature generation learning and self-optimizing feature selection learning in detail. Chapter 3 and Chapter 4 shift to the generative-AI perspective, where I will comprehensively examine autoregressive feature generation learning and autoregressive feature selection learning. Chapter 5 will summarize the key findings and contributions of this research. The future research directions in these areas will also be discussed in this chapter.

CHAPTER 2: DECISION-MAKING PERSPECTIVE: SELF-OPTIMIZING FEATURE GENERATION LEARNING

In this chapter, I will introduce the framework for self-optimizing feature generation learning. This task is formulated as the collaboration of three discrete decision-making processes and reinforcement learning is employed to develop a practical framework.

Introduction

Classic Machine Learning (ML) mainly includes data prepossessing, feature extraction, feature engineering, predictive modeling, and evaluation. The evolution of deep AI, however, has resulted in a new principled and widely used paradigm: i) collecting data, ii) computing data representations, and iii) applying ML models. Indeed, the success of ML algorithms highly depends on data representation [6]. Building a good representation space is critical and fundamental because it can help to 1) identify and disentangle the underlying explanatory factors hidden in observed data, 2) easy the extraction of useful information in predictive modeling, 3) reconstruct distance measures to form discriminative and machine-learnable patterns, 4) embed structure knowledge and priors into representation space and thus make classic ML algorithms available to complex graph, spatiotemporal, sequence, multi-media, or even hybrid data.

In this paper, we study the problem of learning to reconstruct an optimal and explainable feature representation space to advance a downstream ML task (**Figure 2.1**). Formally, given a set of original features, a prediction target, and a downstream ML task (e.g., classification, regression), the objective is to automatically reconstruct an optimal and explainable set of features for the ML task.

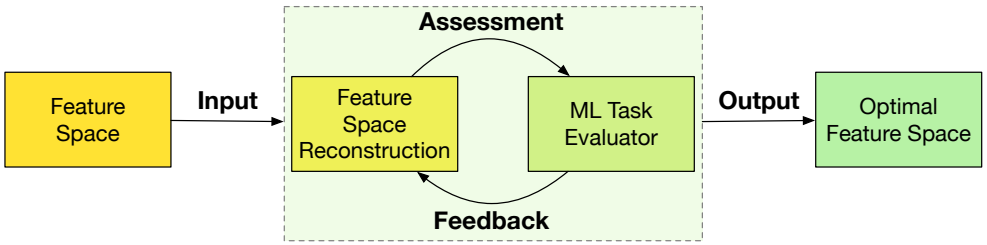


Figure 2.1: We want to uncover the optimal feature space that is explainable and performs optimally in a downstream ML task by iteratively reconstructing the feature space.

Prior literature has partially addressed the problem. The first relevant work is feature engineering, which designs preprocessing, feature extraction, selection [56, 30], and generation [44] to extract a transformed representation of the data. These techniques are essential but labor-intensive, showing the low applicability of current ML practice in the automation of extracting a discriminative feature representation space. **Issue 1 (full automation):** *how can we make ML less dependent on feature engineering, construct ML systems faster, and expand the scope and applicability of ML?* The second relevant work is representation learning, such as factorization [24], embedding [25], and deep representation learning [97, 95]. These studies are devoted to learning effective latent features. However, the learned features are implicit and non-explainable. Such traits limit the deployment of these approaches in many application scenarios (e.g., patient and biomedical domains) that require not just high predictive accuracy but also trusted understanding and interpretation of underlying drivers. **Issue 2 (explainable explicitness):** *how can we assure that the reconstructing representation space is traceable and explainable?* The third relevant work is learning based feature transformation, such as principle component analysis [10], traversal transformation graph based feature generation [44], sparsity regularization based feature selection [34]. These methods are either deeply embedded into or totally irrelevant to a specific ML model. For example, LASSO regression extracts an optimal feature subset for regression, but not for any given ML model. **Issue 3 (flexible optimal):** *how can we create a framework to reconstruct a new representation space for any given predictor?* The three issues are well-known challenges. Our goal is to develop a new

perspective to address these issues.

Our Contributions: A Traceable Group-wise Reinforcement Generation Perspective. We propose a novel principled framework to address the automation, explicitness, optimal issues in representation space reconstruction. We view feature generation and selection from the lens of Reinforcement Learning (RL). We show that learning to reconstruct representation space can be accomplished by an interactive process of nested feature generation and selection, where feature generation is to generate new meaningful and explicit features, and feature selection is to remove redundant features to control feature sizes. We highlight that the human intuition and domain expertise in feature generation and selection can be formulated as machine-learnable policies. RL is an emerging technique to automatically generate experiences data and learn globally optimized policies. Such traits have sparked considerable interest in recent years. We demonstrate that the iterative sequential feature generation and selection can be generalized as an RL task. We find that crossing features of high information distinctness is more likely to generate meaningful variables in a new representation space, and, thus, leveraging group-group crossing can accelerate learning efficiency.

Summary of Proposed Approach. Based on our findings, we develop a generic and principled framework: group-wise reinforcement feature generation, for optimal and explainable representation space reconstruction. This framework learns a representation space reconstructor that can 1) **Goal 1: explainable explicitness:** provide traceable generation process and understand the meanings of each generated feature. 2) **Goal 2: self optimization:** automatically generate an optimal feature set for a downstream ML task without much professional experience and human intervention; 3) **Goal 3: enhanced efficiency and reward augmentation:** enhance the generation and exploration speed in a large feature space and augment reward incentive signal to learn clear policies.

To achieve Goal 1, we propose an iterative feature generation and selection strategy, where the generation step is to apply a mathematical operation to two features to create a new feature and the selection step is to control the feature set size. This strategy allows us to explicitly trace the generation process and extract the semantic labels of generated features. To achieve Goal 2, we decompose feature generation into three Markov Decision Processes (MDPs): one is to select the first meta feature, one is to select an operation, and one is to select the second meta feature. We develop a new cascading agent structure to coordinate agents to share states and learn better selection policies for feature generation. To achieve Goal 3, we design a group-operation-group based generation approach, instead of intuitive feature-operation-feature based generation, in order to accelerate representation space reconstruction. In particular, we first cluster the original features into different feature groups by maximizing intra-group feature cohesion and inter-group feature distinctness, where we propose a novel feature-feature information distance. We then let agents select and cross two feature groups to generate multiple features each time. The benefits of this strategy are two folds: i) it explores feature space faster; ii) if we use feature-operation-feature based generation to add a single feature each time, the state of a feature set cannot be sufficiently changed, thus, restricting the agents from gaining enough reward to learn effective policies. Instead, the group-operation-group based generation can alleviate this issue by augmenting the reward signal.

Definitions and Problem Statement

In this section, we present several important definitions and then outline the problem statement.

Important Definitions

Definition 1 ***Feature Group.*** *We aim to reconstruct the feature space of such datasets $\mathcal{D} < \mathcal{F}, y >$. Here, $\mathcal{F}$ is a feature set, in which each column denotes a feature and each row denotes a data sample; y is the target label set corresponding to samples. To effectively and efficiently produce new features, we divide the feature set $\mathcal{F}$ into different feature groups via clustering, denoted by $\mathcal{C}$. Each feature group is a feature subset of $\mathcal{F}$.*

Definition 2 ***Operation Set.*** *We perform a mathematical operation on existing features in order to generate new ones. The collection of all operations is an operation set, denoted by $\mathcal{O}$. There are two types of operations: unary and binary. The unary operations include "square", "exp", "log", and etc. The binary operations are "plus", "multiply", "divide", and etc.*

Definition 3 ***Cascading Agent.*** *To address the feature generation challenge, we develop a new cascading agent structure. This structure is made up of three agents: two feature group agents and one operation agent. Such cascading agents share state information and sequentially select feature groups and operations.*

Problem Statement

The research problem is learning to reconstruct an optimal and explainable feature representation space to advance a downstream ML task. Formally, given a dataset $D < \mathcal{F}, y >$ that includes an original feature set $\mathcal{F}$ and a target label set y, an operator set $\mathcal{O}$, and a downstream ML task A (e.g., classification, regression, ranking, detection), our objective is to automatically reconstruct an optimal and explainable feature set $\mathcal{F}^*$ that optimizes the performance indicator V of the task A.

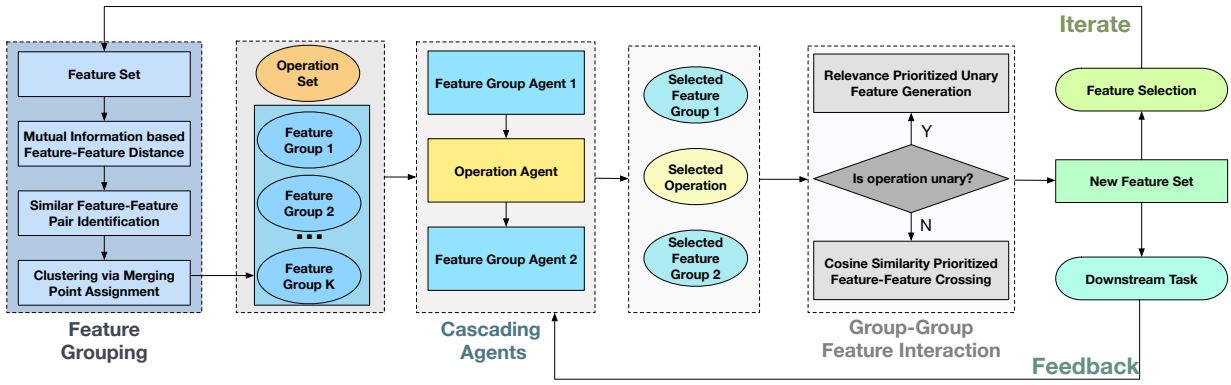


Figure 2.2: An overview of GRFG. First, we cluster the feature set into feature groups. Second, we employ cascading agents to select two feature groups and one operation. Next, we conduct group-group feature interaction to generate new features and combine them with original features to create a new feature set. Then, the updated feature set is fed into a downstream task to assess the selection process of cascading agents for parameter update. Meanwhile, we adopt feature selection to control the size of the feature set and iterate the process until the best feature set is discovered or the maximum number of iterations is reached.

The optimization objective is to find a reconstructed feature set $\hat{\mathcal{F}}$ that maximizes:

$$\mathcal{F}^* = argmax_{\hat{\mathcal{F}}}(V_A(\hat{\mathcal{F}}, y)), \tag{2.1}$$

where $\hat{\mathcal{F}}$ can be viewed as a subset of a combination of the original feature set $\mathcal{F}$ and the generated new features $\mathcal{F}^g$, and $\mathcal{F}^g$ is produced by applying the operations $\mathcal{O}$ to the original feature set $\mathcal{F}$ via a certain algorithmic structure.

Methodology

In this section, we present an overview, and then detail each technical component of our framework.

Framework Overview

Figure 2.2 shows our proposed framework, **G**roup-wise **R**einforcement **F**eature **G**eneration (**GRFG**). In the first step, we cluster the original features into different feature groups by maximizing intra-group feature cohesion and inter-group feature distinctness. In the second step, we leverage a group-operation-group strategy to cross two feature groups to generate multiple features each time. For this purpose, we develop a novel cascading reinforcement learning method to learn three agents to select the two most informative feature groups and the most appropriate operation from the operation set. As a key enabling technique, the cascading reinforcement method will coordinate the three agents to share their perceived states in a cascading fashion, i.e., (agent1, state of the first feature group), (agent2, fused state of the operation and agent1's state), and (agent3, fused state of the second feature group and agent2's state), in order to learn better choice policies. After the two feature groups and operation are selected, we generate new features via a group-group crossing strategy. In particular, if the operation is unary, e.g., sqrt, we choose the feature group of higher relevance to target labels from the two feature groups, and apply the operation to the more relevant feature group to generate new features. if the operation is binary, we will choose the K most distinct feature-pairs from the two feature groups, and apply the binary operation to the chosen feature pairs to generate new features. In the third step, we add the newly generated features to the original features to create a generated feature set. We feed the generated feature set into a downstream task to collect predictive accuracy as reward feedback to update policy parameters. Finally, we employ feature selection to eliminate redundant features and control the dimensonality of the newly generated feature set, which will be used as the original feature set to restart the iterations to regenerate new features until the maximum number of iterations is reached.

Generation-oriented Feature Clustering

One of our key findings is that group-wise feature generation can accelerate the generation and exploration, and, moreover, augment reward feedback of agents to learn clear policies. Inspired by this finding, our system starts with generation oriented feature clustering, which aims to create feature groups that are meaningful for group-group crossing. Our another insight is that crossing features of high (low) information distinctness is more (less) likely to generate meaningful variables in a new representation space. As a result, unlike classic clustering, we aim to cluster features into different feature groups, with the optimization objective of maximizing inter-group feature information distinctness while minimizing intra-group feature information distinctness. To achieve this goal, we propose the **M-Clustering** for feature clustering, which starts with each feature as a feature group and then merges the closest feature group pair at each iteration.

Distance Between Feature Groups: A Group-level Relevance-Redundancy Ratio Perspective. To achieve the goal of minimizing intra-group feature distinctness and maximizing inter-group feature distinctness, we develop a new distance measure to quantify the distance between two feature groups. We highlight two interesting findings: 1) relevance to predictive target: if the relevance between one feature group and predictive target is similar to the relevance of another feature group and predictive target, the two feature groups are similar; 2) mutual information: if the mutual information between the features of the two groups are large, the two feature groups are similar. Based on the two insights, we devise a feature group-group distance. The distance can be used to evaluate the distinctness of two feature groups, and, further, understand how likely crossing the two feature groups will generate more meaningful features. Formally, the distance is given by:

$$dis(\mathcal{C}_i, \mathcal{C}_j) = \frac{1}{|\mathcal{C}_i| \cdot |\mathcal{C}_j|} \sum_{f_i \in \mathcal{C}_i} \sum_{f_j \in \mathcal{C}_j} \frac{|MI(f_i, y) - MI(f_j, y)|}{MI(f_i, f_j) + \epsilon}, \tag{2.2}$$

where $\mathcal{C}_i$ and $\mathcal{C}_j$ denote two feature groups, $|\mathcal{C}_i|$ and $|\mathcal{C}_j|$ respectively are the numbers of features

in $\mathcal{C}_i$ and $\mathcal{C}_j$, f_i and f_j are two features in $\mathcal{C}_i$ and $\mathcal{C}_j$ respectively, y is the target label vector. In particular, $|MI(f_i, y) - MI(f_j, y)|$ quantifies the difference in relevance between y and f_i, f_j. If $|MI(f_i, y) - MI(f_j, y)|$ is small, f_i and f_j have a more similar influence on classifying y; $MI(f_i, f_j)+\epsilon$ quantifies the redundancy between f_i and f_j. ϵ is a small value that is used to prevent the denominator from being zero. If $MI(f_i, f_j) + \epsilon$ is big, f_i and f_j share more information.

Feature Group Distance based M-Clustering Algorithm: We develop a group-group distance instead of point-point distance, and under such a group-level distance, the shape of the feature cluster could be non-spherical. Therefore, it is not appropriate to use K-means or density based methods. Inspired by agglomerative clustering, given a feature set $\mathcal{F}$, we propose a three step method: 1) INITIALIZATION: we regard each feature in $\mathcal{F}$ as a small feature cluster. 2) REPEAT: we calculate the information overlap between any two feature clusters and determine which cluster pair is the most closely overlapped. We then merge two clusters into one and remove the two original clusters. 3) STOP CRITERIA: we iterate the REPEAT step until the distance between the closest feature group pair reaches a certain threshold. Although classic stop criteria is to stop when there is only one cluster, using the distance between the closest feature groups as stop criteria can better help us to semantically understand, gauge, and identify the information distinctness among feature groups. It eases the implementation in practical deployment.

Cascading Reinforcement Feature Groups and Operation Selection

To achieve group-wise feature generation, we need to select a feature group, an operation, and another feature group to perform group-operation-group based crossing. Two key findings inspire us to leverage cascading reinforcement. **Firstly**, we highlight that although it is hard to define and program the optimal selection criteria of feature groups and operation, we can view selection criteria as a form of machine-learnable policies. We propose three agents to learn the policies by

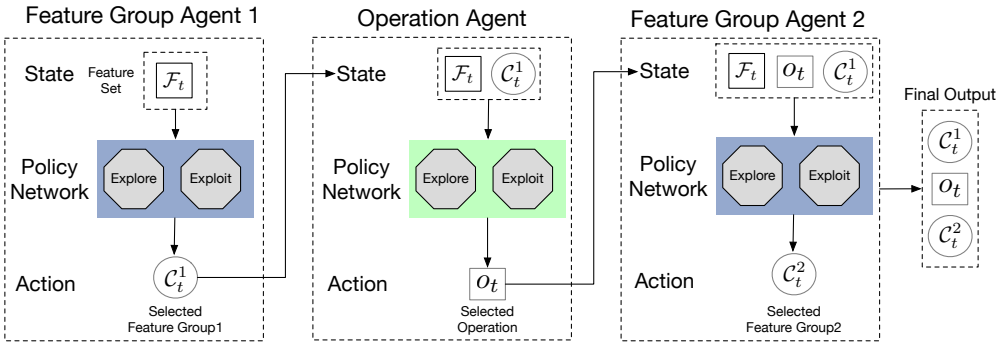


Figure 2.3: The cascading agents are comprised of the feature group agent1, the operation agent, and the feature group agent2. They collaborate to choose two candidate feature groups and a single operation.

trials and errors. **Secondly**, we find that the three selection agents are cascading in a sequential auto-correlated decision structure, not independent and separated. Here, "cascading" refers to the fact that within each iteration agents make decision sequentially, and downstream agents await for the completion of an upstream agent. The decision of an upstream agent will change the environment states of downstream agents. As shown in Figure 2.3, the first agent picks the first feature group based on the state of the entire feature space, the second agent picks the operation based on the entire feature space and the selection of the first agent, and the third agent chooses the second feature group based on the entire feature space and the selections of the first and second agents.

We next propose two generic metrics to quantify the usefulness (reward) of a feature set, and then form three MDPs to learn three selection policies.

Two Utility Metrics for Reward Quantification. The two utility metrics are from the supervised and unsupervised perspectives.

Metric 1: Integrated Feature Set Redundancy and Relevance. We propose a metric to quantify feature set utility from an information theory perspective: a higher-utility feature set has less re-

dundant information and is more relevant to prediction targets. Formally, given a feature set $\mathcal{F}$ and a predictive target label y, such utility metric can be calculated by

$$U(\mathcal{F}|y) = -\frac{1}{|\mathcal{F}|^2}\sum_{f_i,f_j\in\mathcal{F}} MI(f_i,f_j) + \frac{1}{|\mathcal{F}|}\sum_{f\in\mathcal{F}} MI(f,y), \tag{2.3}$$

where MI is the mutual information, f_i, f_j, f are features in $\mathcal{F}$ and $|\mathcal{F}|$ is the size of the feature set $\mathcal{F}$.

Metric 2: Downstream Task Performance. Another utility metric is whether this feature set can improve a downstream task (e.g., regression, classification). We use a downstream predictive task performance indicator (e.g., 1-RAE, Precision, Recall, F1) as a utility metric of a feature set.

Learning Selection Agents of Feature Group 1, Operation, and Feature Group 2. Leveraging the two metrics, we next develop three MDPs to learn three agent policies to select the best feature group 1, operation, feature group 2.

Learning the Selection Agent of Feature Group 1. The feature group agent 1 iteratively select the best meta feature group 1. Its learning system includes: **i) Action:** its action at the t-th iteration is the meta feature group 1 selected from the feature groups of the previous iteration, denoted group $a_t^1 = \mathcal{C}_{t-1}^1$. **ii) State:** its state at the t-th iteration is a vectorized representation of the generated feature set of the previous iteration. Let Rep be a state representation method, the state can be denoted by $s_t^1 = Rep(\mathcal{F}_{t-1})$. We will discuss the state representation method in the next section. **iii) Reward:** its reward at the t-th iteration is the utility score the selected feature group 1, denoted by $\mathcal{R}(s_t^1, a_t^1) = U(\mathcal{C}_{t-1}|y)$.

Learning the Selection Agent of Operation. The operation agent will iteratively select the best operation (*e.g.* +, -) from an operation set as a feature crossing tool for feature generation. Its learning system includes: **i) Action:** its action at the t-th iteration is the selected operation, denoted

by $a_t^o = o_t$. **ii) State:** its state at the t-th iteration is the combination of $Rep(\mathcal{F}_{t-1})$ and the representation of the feature group selected by the previous agent, denoted by $s_t^o = Rep(\mathcal{F}_{t-1}) \oplus Rep(\mathcal{C}_{t-1}^1)$, where $\oplus$ indicates the concatenation operation. **iii) Reward:** The selected operation will be used to generate new features by feature crossing. We combine such new features with the original feature set to form a new feature set. Thus, the feature set at the t-th iteration is $\mathcal{F}_t = \mathcal{F}_{t-1} \cup g_t$, where g_t is the generated new features. The reward of this iteration is the improvement in the utility score of the feature set compared with the previous iteration, denoted by $\mathcal{R}(s_t^o, a_t^o) = U(\mathcal{F}_t|y) - U(\mathcal{F}_{t-1}|y)$.

Learning the Selection Agent of Feature Group 2. The feature group agent 2 will iteratively select the best meta feature group 2 for feature generation. Its learning system includes: **i) Action:** its action at the t-th iteration is the meta feature group 2 selected from the clustered feature groups of the previous iteration, denoted by $a_t^2 = \mathcal{C}_{t-1}^2$. **ii) State:** its state at the t-th iteration is combination of $Rep(\mathcal{F}_{t-1})$, $Rep(\mathcal{C}_{t-1}^1)$ and the vectorized representation of the operation selected by the operation agent, denoted by $s_t^2 = Rep(\mathcal{F}_{t-1}) \oplus Rep(\mathcal{C}_{t-1}^1) \oplus Rep(o_t)$. **iii) Reward:** its reward at the t-th iteration is improvement of the feature set utility and the feedback of the downstream task, denoted by $\mathcal{R}(s_t^2, a_t^2) = U(\mathcal{F}_t|y) - U(\mathcal{F}_{t-1}|y) + V_{A_t}$, where V_A is the performance (e.g., F1) of a downstream predictive task.

State Representation of a Feature Group and an Operation. We propose to map a feature group to a vector that characterizes the **State** of the given feature group. In detail, given a feature group $\mathcal{F}$, we first calculate the descriptive statistics (*i.e.* count, standard deviation, minimum, maximum, first, second, and third quartile) column by column. Then, row by row, we calculate the descriptive statistics of the outcome of the previous step to obtain the descriptive matrix that shape is $\mathbb{R}^{7\times7}$. After that, we obtain the feature feature's representation $Rep(\mathcal{F}) \in \mathbb{R}^{1\times49}$ by flatting the descriptive matrix. A fixed-size state vector is produced by the representation method, which accommodates the varying size of the feature set at each generation iteration. Second, for the

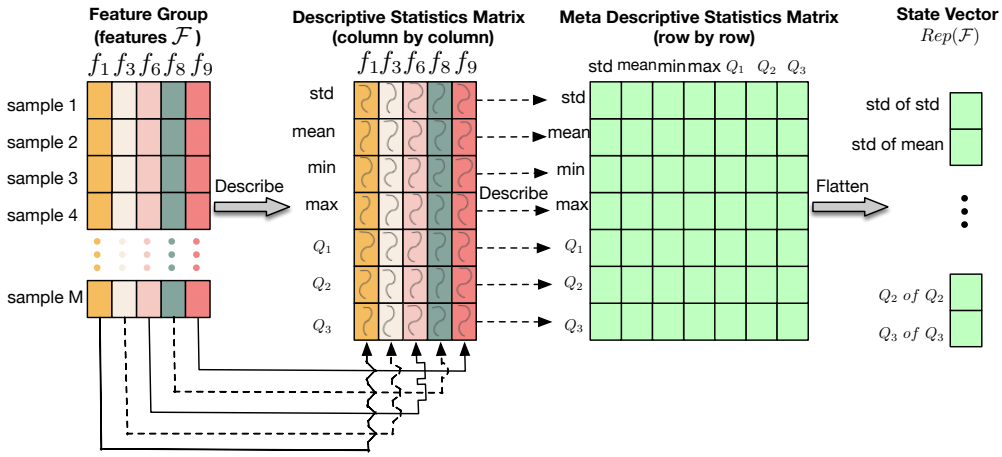


Figure 2.4: State Representation. Given a feature group $\mathcal{F}$ consisting of several features, we calculate the descriptive statistics of $\mathcal{F}$ column-by-column, then row-by-row to get a meta descriptive statistics matrix. Then, we flat the matrix to obtain the state representation vector $Rep(\mathcal{F})$.

operation, we use the one-hot encoding as its representation $Rep(o)$.

Solving the Optimization Problem. We train the three agents by maximizing the discounted and cumulative reward during the iterative feature generation process. In other words, we encourage the cascading agents to collaborate to generate a feature set that is independent, informative, and performs well in the downstream task. To accomplish this goal, we minimize the temporal difference error $\mathcal{L}$ converted from the Bellman equation, given by:

$$\mathcal{L} = Q(s_t, a_t) - (\mathcal{R}(s_t, a_t) + \gamma * \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})), \tag{2.4}$$

where $\gamma \in [0 \sim 1]$ is the discounted factor; Q denotes the Q function estimated by deep neural networks. After agents converge, we expect to discover the optimal policy π^* that can choose the most appropriate action (*i.e.* feature group or operation) based on the state via the Q-value, which can be formulated as follows:

$$\pi^*(a_t|s_t) = \text{argmax}_a Q(s_t, a). \tag{2.5}$$

Group-wise Feature Generation

We found that using group-level crossing can generate more features each time, and, thus, accelerate exploration speed, augment reward feedback by adding significant amount of features, and effectively learn policies. The selection results of our reinforcement learning system include **two generation scenarios**: (1) selected are a binary operation and two feature groups; (2) selected are a unary operation and two feature groups. However, a challenge arises: what are the most effective generation strategy for the two scenarios? We next propose two strategies for the two scenarios.

Scenario 1: Cosine Similarity Prioritized Feature-Feature Crossing. We highlight that it is more likely to generate informative features by crossing two features that are less overlapped in terms of information. We propose a simple yet efficient strategy, that is, to select the top K dissimilar feature pairs between two feature groups. Specifically, we first cross two selected feature groups to prepare feature pairs. We then compute the cosine similarities of all feature pairs. Later, we rank and select the top K dissimilar feature pairs. Finally, we apply the operation to the top K selected feature pairs to generate K new features.

Scenario 2: Relevance Prioritized Unary Feature Generation. When selected are an unary operation and two feature groups, we directly apply the operation to the feature group that is more relevant to target labels. Given a feature group C, we use the average mutual information between all the features in C and the prediction target y to quantify the relevance between the feature group and the prediction targets, which is given by: $rel = \frac{1}{|C|} \sum_{f_i \in C} MI(f_i, y)$, where MI is a function of mutual information. After the more relevant feature group is identified, we apply the unary operation to each feature of the feature group to generate new features.

Post-generation Processing. After feature generation, we combine the newly generated features with the original feature set to form an updated feature set, which will be fed into a downstream

task to evaluate predictive performance. Such performance is exploited as reward feedback to update the policies of the three cascading agents in order to optimize the next round of feature generation. To prevent feature number explosion during the iterative generation process, we use a feature selection step to control feature size. When the size of the new feature set exceeds a feature size tolerance threshold, we leverage the K-best feature selection method to reduce the feature size. Otherwise, we don't perform feature selection. We use the tailored new feature set as the original feature set of the next iteration.

Finally, when the maximum number of iterations is reached, the algorithm returns the optimal feature set $\mathcal{F}^*$ that has the best downstream performance over the entire exploration.

Experiments

Experimental Setup

Data Description

We used 24 publicly available datasets from UCI [73], LibSVM [14], Kaggle [39], and OpenML [72] to conduct experiments. The 24 datasets involve 14 classification tasks and 10 regression tasks. Table 5.1 shows the statistics of the data.

Evaluation Metrics

We used the F1-score to evaluate the recall and precision of classification tasks. We used 1-relative absolute error (RAE) to evaluate the accuracy of regression tasks. Specifically, 1-RAE $= 1 - \frac{\sum_{i=1}^{n} |y_i - \check{y}_i|}{\sum_{i=1}^{n} |y_i - \bar{y}_i|}$, where n is the number of data points, $y_i, \check{y}_i, \bar{y}_i$ respectively denote golden standard

target values, predicted target values, and the mean of golden standard targets.

Baseline Algorithms

We compared our method with five widely-used feature generation methods: (1) **RDG** randomly selects feature-operation-feature pairs for feature generation; (2) **ERG** is an expansion-reduction method, that applies operations to each feature to expand the feature space and selects significant features from the larger space as new features. (3) **LDA** [9] extracts latent features via matrix factorization. (4) **AFT** [38] is an enhanced ERT implementation that iteratively explores feature space and adopts a multi-step feature selection relying on L1-regularized linear models. (5) **NFS** [12] mimics feature transformation trajectory for each feature and optimizes the entire feature generation process through reinforcement learning. (6) **TTG** [44] records the feature generation process using a transformation graph, then uses reinforcement learning to explore the graph to determine the best feature set.

Besides, we developed four variants of GRFG to validate the impact of each technical component: (i) $\mathbf{GRFG}^{-c}$ removes the clustering step of GRFG and generate features by feature-operation-feature based crossing, instead of group-operation-group based crossing. (ii) $\mathbf{GRFG}^{-d}$ utilizes the euclidean distance as the measurement of M-Clustering. (iii) $\mathbf{GRFG}^{-u}$ randomly selects a feature group from the feature group set, when the operation is unary. (iv) $\mathbf{GRFG}^{-b}$ randomly selects features from the larger feature group to align two feature groups when the operation is binary. We adopted Random Forest, as the downstream ML model, in order to ensure the changes of results are mainly caused by the feature space reconstruction, not randomness or variance of the predictor. We performed 5-fold stratified cross-validation in all experiments.

Hyperparameters, Source Code and Reproducibility

The operation set consists of *square root, square, cosine, sine, tangent, exp, cube, log, reciprocal, sigmoid, plus, subtract, multiply, divide*. We limited iterations (epochs) to 30, with each iteration consisting of 15 exploration steps. When the number of generated features is twice of the original feature set size, we performed feature selection to control feature size. In GRFG, all agents were constructed using a DQN network with two linear layers activated by the RELU function. We optimized DQN using the Adam optimizer with a 0.01 learning rate, and set the limit of the experience replay memory as 32 and the batch size as 8. The parameters of all baseline models are set based on the default settings of corresponding papers.

Environmental Settings

All experiments were conducted on the Ubuntu 18.04.5 LTS operating system, Intel(R) Core(TM) i9-10900X CPU@ 3.70GHz, and 1 way SLI RTX 3090 and 128GB of RAM, with the framework of Python 3.8.5 and PyTorch 1.8.1.

Performance Evaluation

Overall Comparison

This experiment aims to answer: *Can our method effectively construct quality feature space and improve a downstream task?* Table 5.1 shows the comparison results in terms of F1 score or 1-RAE. We observed that GRFG ranks first on most datasets and ranks second on "Credit Default" and "SpamBase". The underlying driver is that the personalized feature crossing strategy in GRFG considers feature-feature distinctions when generating new features. Besides, the observation that

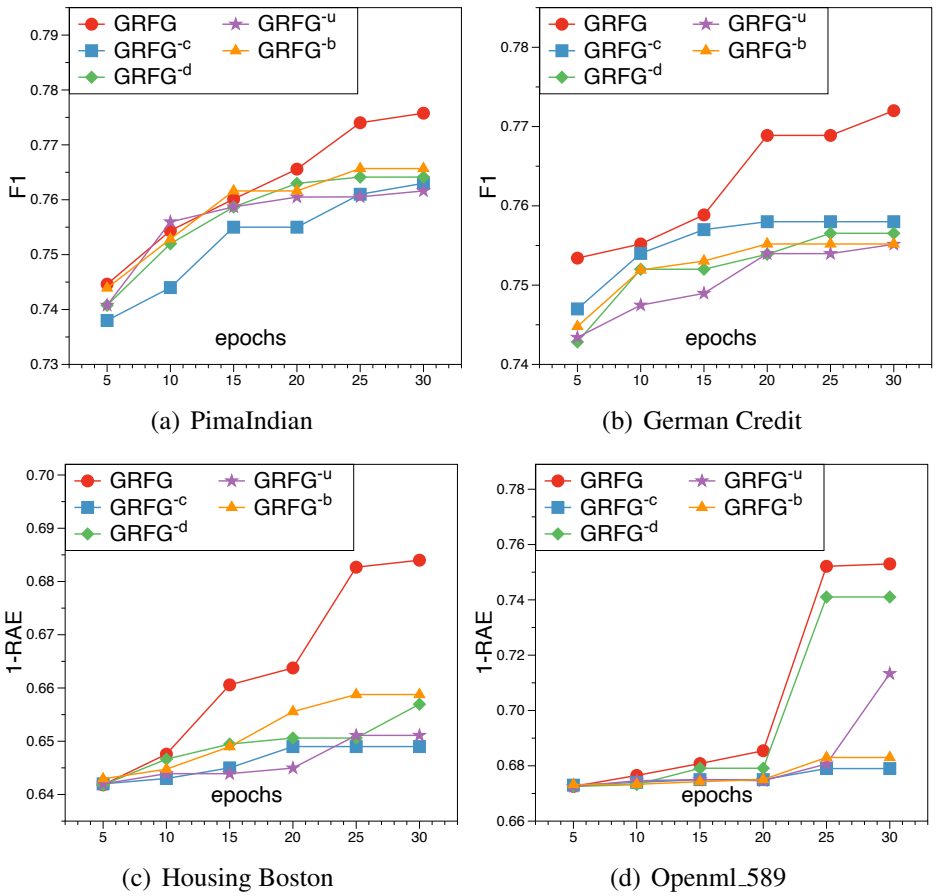
Table 2.1: Overall performance comparison. 'C' for classification and 'R' for regression.

Dataset	Source	C/R	Samples	Features	RDG	ERG	LDA	AFT	NFS	TTG	GRFG
Higgs Boson	UCIrvine	C	50000	28	0.683	0.674	0.509	0.711	0.715	0.705	**0.719**
Amazon Employee	Kaggle	C	32769	9	0.744	0.740	0.920	0.943	0.935	0.806	**0.946**
PimaIndian	UCIrvine	C	768	8	0.693	0.703	0.676	0.736	0.762	0.747	**0.776**
SpectF	UCIrvine	C	267	44	0.790	0.748	0.774	0.775	0.876	0.788	**0.878**
SVMGuide3	LibSVM	C	1243	21	0.703	0.747	0.683	0.829	0.831	0.766	**0.850**
German Credit	UCIrvine	C	1001	24	0.695	0.661	0.627	0.751	0.765	0.731	**0.772**
Credit Default	UCIrvine	C	30000	25	0.798	0.752	0.744	0.799	0.799	**0.809**	0.800
Messidor_features	UCIrvine	C	1150	19	0.673	0.635	0.580	0.678	0.746	0.726	**0.757**
Wine Quality Red	UCIrvine	C	999	12	0.599	0.611	0.600	0.658	0.666	0.647	**0.686**
Wine Quality White	UCIrvine	C	4900	12	0.552	0.587	0.571	0.673	0.679	0.638	**0.685**
SpamBase	UCIrvine	C	4601	57	0.951	0.931	0.908	0.951	0.955	**0.961**	0.958
AP-omentum-ovary	OpenML	C	275	10936	0.711	0.705	0.117	0.783	0.804	0.795	**0.818**
Lymphography	UCIrvine	C	148	18	0.654	0.638	0.737	0.833	0.859	0.846	**0.866**
Ionosphere	UCIrvine	C	351	34	0.919	0.926	0.730	0.827	0.949	0.938	**0.960**
Bikeshare DC	Kaggle	R	10886	11	0.483	0.571	0.494	0.670	0.675	0.659	**0.681**
Housing Boston	UCIrvine	R	506	13	0.605	0.617	0.174	0.641	0.665	0.658	**0.684**
Airfoil	UCIrvine	R	1503	5	0.737	0.732	0.463	0.774	0.771	0.783	**0.797**
Openml_618	OpenML	R	1000	50	0.415	0.427	0.372	0.665	0.640	0.587	**0.672**
Openml_589	OpenML	R	1000	25	0.638	0.560	0.331	0.672	0.711	0.682	**0.753**
Openml_616	OpenML	R	500	50	0.448	0.372	0.385	0.585	0.593	0.559	**0.603**
Openml_607	OpenML	R	1000	50	0.579	0.406	0.376	0.658	0.675	0.639	**0.680**
Openml_620	OpenML	R	1000	25	0.575	0.584	0.425	0.663	0.698	0.656	**0.714**
Openml_637	OpenML	R	500	50	0.561	0.497	0.494	0.564	0.581	0.575	**0.589**
Openml_586	OpenML	R	1000	25	0.595	0.546	0.472	0.687	0.748	0.704	**0.783**

GRFG outperforms random-based (RDG) and expansion-reduction-based (ERG, AFT) methods shows that the agents can share states and rewards in a cascading fashion, and, thus learn an effective policy to select optimal crossing features and operations. Moreover, because our method is a self-learning end-to-end framework, users can treat it as a tool and easily apply it to different datasets regardless of implementation details. Thus, this experiment validates that our method is more practical and automated in real application scenarios.

Study of the impact of each technical component

This experiment aims to answer: *How does each component in GRFG impact its performance?* We developed four variants of GRFG (Section 4). Figure 2.5 shows the comparison results in terms of F1 score or 1-RAE on two classification datasests (*i.e.* PimaIndian, German Credit) and two regression datasets (*i.e.* Housing Boston, Openml_589). First, we developed GRFG^{-c}



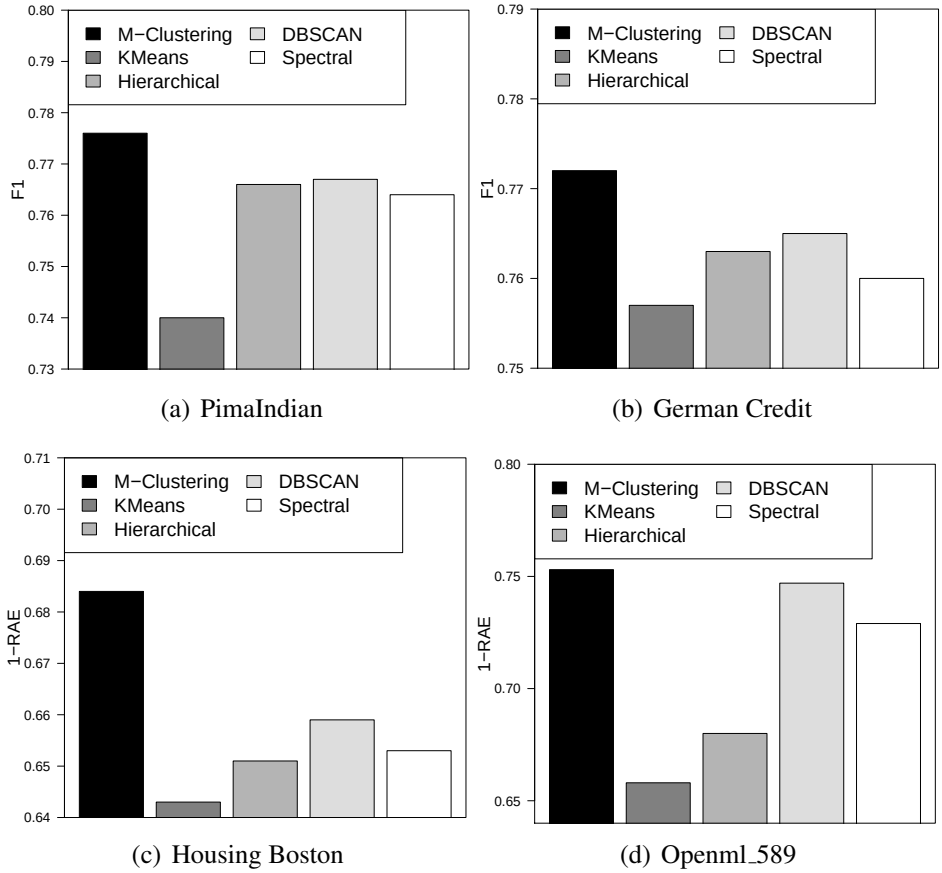
(a) PimaIndian (b) German Credit

(c) Housing Boston (d) Openml_589

Figure 2.5: Comparison of different GRFG variants in terms of F1 or 1-RAE.

by removing the feature clustering step of GRFG. But, GRFG^{-c} performs poorer than GRFG on all datasets. This observation shows that the idea of group-level generation can augment reward feedback to help cascading agents learn better policies. Second, we developed GRFG^{-d} by using euclidean distance as feature distance metric in the M-clustering of GRFG. The superiority of GRFG over GRFG^{-d} suggests that our distance describes group-level information relevance and redundancy ratio in order to maximize information distinctness across feature groups and minimize it within a feature group. Such a distance can help GRFG generate more useful dimensions. Third, we developed GRFG^{-u} and GRFG^{-b} by using random in the two feature generation scenarios (Section 2) of GRFG. We observed that GRFG^{-u} and GRFG^{-b} perform poorer than GRFG. This

validates that crossing two distinct features and relevance prioritized generation can generate more meaningful and informative features.



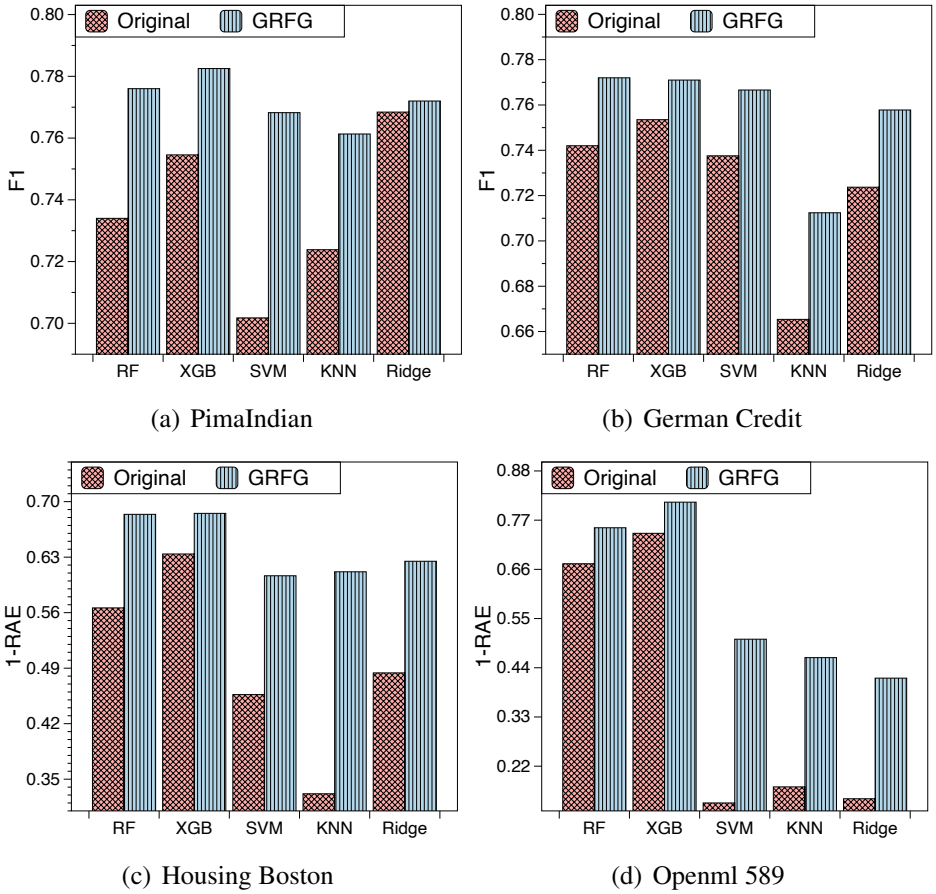
(a) PimaIndian (b) German Credit

(c) Housing Boston (d) Openml_589

Figure 2.6: Comparison of different clustering algorithms in terms of F1 or 1-RAE.

Study of the impact of M-Clustering

This experiment aims to answer: *Is M-Clustering more effective in improving feature generation than classical clustering algorithms?* We replaced the feature clustering algorithm in GRFG with KMeans, Hierarchical Clustering, DBSCAN, and Spectral Clustering respectively. We reported the comparison results in terms of F1 score or 1-RAE on the datasets used in Section 2. Figure 2.6 shows M-Clustering beats classical clustering algorithms on all datasets. The underlying driver



(a) PimaIndian (b) German Credit

(c) Housing Boston (d) Openml 589

Figure 2.7: Comparison of different machine learning models in terms of F1 or 1-RAE.

is that when feature sets change during generation, M-Clustering is more effective in minimizing information overlap of intra-group features and maximizing information distinctness of inter-group features. So, crossing the feature groups with distinct information makes it easier to generate meaningful dimensions.

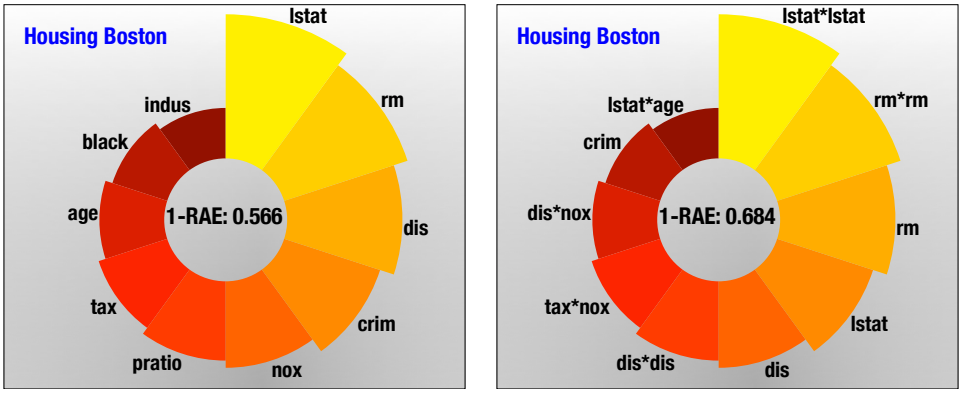
Robustness check of GRFG under different machine learning (ML) models.

This experiment is to answer: *Is GRFG robust when different ML models are used as a downstream task?* We examined the robustness of GRFG by changing the ML model of a downstream task to

Random Forest (RF), Xgboost (XGB), SVM, KNN, and Ridge Regression, respectively. Figure 2.7 shows the comparison results in terms of F1 score or 1-RAE on the datasets used in the Section 2. We observed that GRFG robustly improves model performances regardless of the ML model used. This observation indicates that GRFG can generalize well to various benchmark applications and ML models. We found that RF and XGB are the two most powerful and robust predictors, which is consistent with the finding in Kaggle.COM competition community. Intuitively, the accuracy of RF and XGB usually represent the performance ceiling on modeling a dataset. But, after using our method to reconstruct the data, we continue to significantly improve the accuracy of RF and XGB and break through the performance ceiling. This finding clearly validates the strong robustness of our method.

Study of the traceability and explainability of GRFG

This experiment aims to answer: *Can GRFG generate an explainable feature space? Is this generation process traceable?* We identified the top 10 essential features for prediction in both the original and reconstructed feature space using the Housing Boston dataset to predict housing prices with random forest regression. Figure 2.8 shows the model performances in the central parts of each sub-figure. The texts associated with each pie chart describe the feature name. If the feature name does not include an operation, the corresponding feature is original; otherwise, it is a generated feature. The larger the pie area is, the more essential the corresponding feature is. We observed that the GRFG-reconstructed feature space greatly enhances the model performance by 20.9% and the generated features cover 60% of the top 10 features. This indicates that GRFG generates informative features to refine the feature space. Moreover, we can explicitly trace and explain the source and effect of a feature by checking its name. For instance, the feature "lstat" measures the percentage of the lower status populations in a house, which is negatively related to housing prices. The most essential feature in the reconstructed feature space is "lstat*lstat" that



(a) Original Feature Space (b) GRFG-reconstructed Feature Space

Figure 2.8: Top 10 features for prediction in the original and GRFG-reconstructed feature space.

is generated by applying a "multiply" operation to "lstat". This shows the generation process is traceable and the relationship between "lstat" and housing prices is non-linear.

Related Work

Reinforcement Learning (RL) is the study of how intelligent agents should act in a given environment in order to maximize the expectation of cumulative rewards [83]. According to the learned policy, we may classify reinforcement learning algorithms into two categories: value-based and policy-based. Value-based algorithms (*e.g.* DQN [66], Double DQN [90]) estimate the value of the state or state-action pair for action selection. Policy-based algorithms (*e.g.* PG [84]) learn a probability distribution to map state to action for action selection. Additionally, an actor-critic reinforcement learning framework is proposed to incorporate the advantages of value-based and policy-based algorithms [77]. In recent years, RL has been applied to many domains (e.g. spatial-temporal data mining, recommended systems) and achieves great achievements [96, 99]. In this paper, we formulate the selection of feature groups and operation as MDPs and propose a new cascading agent structure to resolve these MDPs.

Automated Feature Engineering aims to enhance the feature space through feature generation and feature selection in order to improve the performance of machine learning models [13]. Feature selection is to remove redundant features and retain important ones, whereas feature generation is to create and add meaningful variables. *Feature Selection* approaches include: (i) filter methods (*e.g.*, univariate selection [21], correlation based selection [108]), in which features are ranked by a specific score like redundancy, relevance; (ii) wrapper methods (*e.g.*, Reinforcement Learning [63], Branch and Bound [48]), in which the optimized feature subset is identified by a search strategy under a predictive task; (iii) embedded methods (*e.g.*, LASSO [85], decision tree [82]), in which selection is part of the optimization objective of a predictive task. *Feature Generation* methods include: (i) latent representation learning based methods, e.g. deep factorization machine [28], deep representation learning [6]. Due to the latent feature space generated by these methods, it is hard to trace and explain the feature extraction process. (ii) feature transformation based methods, which use arithmetic or aggregate operations to generate new features [44, 12]. These approaches have two weaknesses: (a) ignore feature-feature heterogeneity among different feature pairs; (b) grow exponentially when the number of exploration steps increases. Compared with prior literature, our personalized feature crossing strategy captures the feature distinctness, cascading agents learn effective feature interaction policies, and group-wise generation manner accelerates feature generation.

Conclusion Remarks

In this chapter, we present a group-wise reinforcement feature generation (GRFG) framework for optimal and explainable representation space reconstruction to improve the performances of predictive models. This framework nests feature generation and selection in order to iteratively reconstruct a recognizable and size-controllable feature space via feature-crossing. Specifically, first,

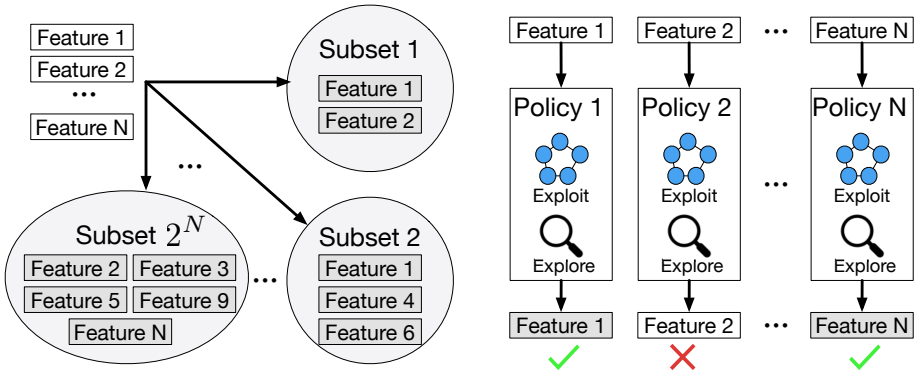
we decompose the process of selecting crossing features and operations into three MDPs and develop a new cascading agent structure for it. Second, we provide two feature generation strategies based on cosine similarity and mutual information to deal with two generation scenarios following cascading selection. Third, we suggest a group-wise feature generation manner to efficiently generate features and augment the rewards of cascading agents. To accomplish this, we propose a new feature clustering algorithm (M-Clustering) to produce robust feature groups from an information theory perspective. Through extensive experiments, we can find that GRFG is effective at refining the feature space and shows competitive results compared to other baselines. Moreover, GRFG can provide traceable routes for feature generation, which improves the explainability of the refined feature space. In the future, we aim to include the pre-training technique into GRFG to further enhance feature generation.

CHAPTER 3: DECISION-MAKING PERSPECTIVE: SELF-OPTIMIZING FEATURE SELECTION LEARNING

In this chapter, the framework for self-optimizing feature selection learning is introduced. Similar to feature generation learning, I formulate feature selection as a Markov decision-making process and employ a singular reinforced agent to construct a practical framework.

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 [106, 21], correlation based feature selection [31, 108]); (ii) wrapper methods, in which optimal feature subset is identified by a search strategy that collaborates with predictive tasks (e.g., evolutionary algorithms [105, 46], branch and bound algorithms [67, 48]); (iii) embedded methods, in which feature selection is part of the optimization objective of predictive tasks (e.g., LASSO [85], decision tree [82]). 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.



(a) Feature subspace candidates (b) Reinforced feature selection

Figure 3.1: 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.

Recently, reinforcement learning has been incorporated with feature selection and produces an emerging feature selection method, called reinforced feature selection [60, 20]. 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 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 fearure selection? To answer this 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 [60], 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 [83]. 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 3.1: 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 $\{s^i \vert i < inf\}$
$\mathcal{A}$	action space defined as $\{a^j \vert j \in [1, N]\}$
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 $\{\mathbf{f}^k \vert k \in [0, M]\}$

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.

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 $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a mapping function from state-action pair to a scalar, $\gamma \in [0, 1]$ is a discount factor and $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ 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.

Monte Carlo for Solveing 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 $\mathbf{x}^i = \{x_1^i, x_2^i, \ldots, x_t^i, \ldots x_N^i\}$, 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:

$$\pi^{i+1}(s) = argmax_a Q_{\pi^i}(s, a) \tag{3.1}$$

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:

$$\pi^0 \xrightarrow{E} Q_{\pi^0} \xrightarrow{I} \pi^1 \xrightarrow{E} Q_{\pi^1} \xrightarrow{I} \ldots \pi^M \xrightarrow{I} Q_{\pi^M} \tag{3.2}$$

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 3.2 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.

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 [60] formulates the feature selection problem into a multi-agent reinforcement learning task, in order to automate the selection process. As Figure 3.2 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 $\mathcal{F}^i$:

$$s^{i+1} = represent(\mathcal{F}^i) \tag{3.3}$$

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 [60]. The reward r^i is an evaluation of the selected feature subset $\mathcal{F}^i$:

$$r^i = eval(\mathcal{F}^i) \tag{3.4}$$

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 [60]. 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.

Methodology

In this section, we first propose a single-agent Monte Carlo-based reinforced feature selection method. 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

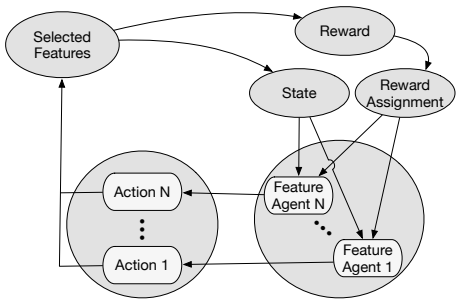


Figure 3.2: Multi-agent Reinforced feature selection. Each feature is controlled by one feature agent.

reinforced feature selection scenario. Finally, we design a reward-shaping strategy to improve the training efficiency.

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 Mote Carlo method as the reinforcement learning algorithm.

Traverse strategy

As Figure 3.3 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 form a selected feature subset $\mathcal{F}_t^i$. Meanwhile, this process also generates an episode

$\mathbf{x}_N^i = \{x_1^i, x_2^i, \ldots, x_t^i, \ldots x_N^i\}$, 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 $represent(\mathcal{F}_t^i)$ and the reward r_t^i is derived by $eval(\mathcal{F}_t^i)$.

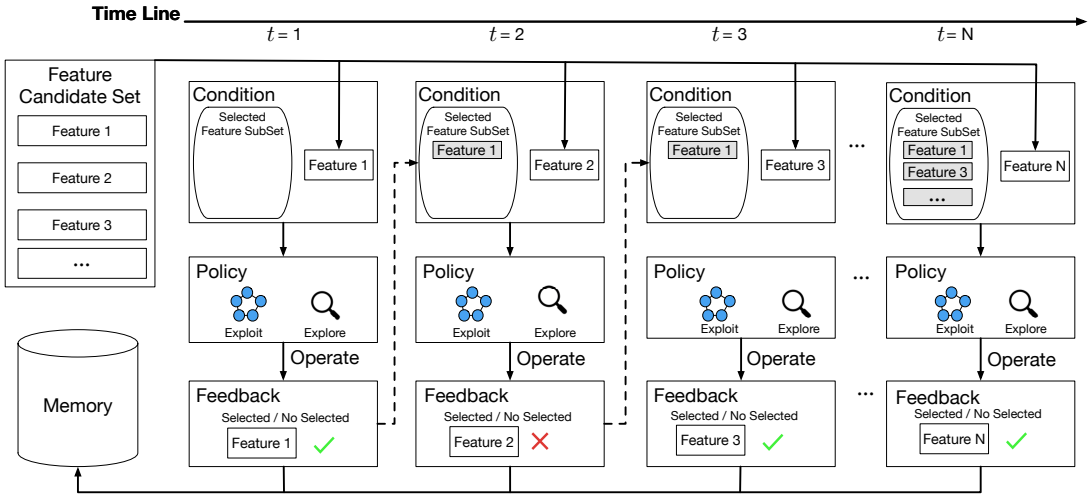


Figure 3.3: Single-agent Reinforced feature selection with traverse strategy. At each step, the agent transverses features one by one to decide their selection. The traverse data are stored in the memory to form a training episode.

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 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:

$$G^i(s_t^i, a_t^i) = \sum_{j=0}^{t} \gamma^{(t-j)} \cdot r_j^i \tag{3.5}$$

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:

$$\rho_t^i = \frac{\Pi_{j=0}^t \pi^i(a_j^i|s_j^i)}{\Pi_{j=0}^t b^i(a_j^i|s_j^i)} \tag{3.6}$$

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

$$\mathcal{L}_{\pi^i} = ||Q_{\pi^i}(s_t^i, a_t^i) - \rho_t^i * G^i(s_t^i, a_t^i)||^2 \tag{3.7}$$

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

$$\pi^{i+1}\{a|s\} = \frac{exp(Q_{\pi^i}\{a, s\})}{exp(Q_{\pi^i}\{a = 0, s\}) + exp(Q_{\pi^i}\{a = 1, s\})} \tag{3.8}$$

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

$$b^{i+1}\{a|s\} = \begin{cases} 1 - \epsilon & a = argmax_a Q_{\pi^i}(s, a); \\ \epsilon & otherwise; \end{cases} \tag{3.9}$$

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

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

Input: Feature set $\mathcal{F} = \{f_1, f_2, ..., f_N\}$, downstream machine learning task $\mathcal{T}$.
Output: Optimal feature subset $\mathcal{F}'$.

Initialize the behavior policy b^1, target policy π^1, exploration number M, $\mathcal{F}' = \Phi$.
for $i = 1$ *to* M **do**
    Initialize state s_1^i.
    for $t = 1$ *to* N **do**
        Derive action a_t^i with behavior policy $b^i(s_t^i)$.
        Perform a_t^i, getting selected feature subset $\mathcal{F}_t^i$.
        Obtain the next state s_{t+1}^i by $represent(\mathcal{F}_t^i)$ and reward r_t^i by $eval(\mathcal{F}_t^i)$.
    end
    Update target policy π^{i+1} by Equation 3.8 and behavior policy b^{i+1} by Equation 3.9.
    if $eval(\mathcal{F}_N^i) > eval(\mathcal{F}')$ **then**
        $\mathcal{F}' = \mathcal{F}_N^i$.
    end
end
Return $\mathcal{F}'$.

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 [65]. The skew samples not only are a waste time 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.

Incremental Importance Sampling

Rather than calculating the importance sampling weight for each sample directly by Equation 3.6, we here decompose it into an incremental format. Specifically, in the i-th iteration, we define the weight increment:

$$w_t^i = \frac{\pi^i(a_t^i|s_t^i)}{b^i(a_t^i|s_t^i)} \tag{3.10}$$

and the importance sampling weight can be calculated by:

$$\rho_t^i = \rho_{t-1}^i \cdot w_t^i \tag{3.11}$$

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

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:

$$p_t^i = max(0, 1 - \rho_t^i/v) \tag{3.12}$$

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:

$$w_t^i = p_v^i \cdot \rho_t^i/p_t^i \tag{3.13}$$

where the p_v can be calculated by:

$$p_v^i = \int max(0, 1 - \rho_t^i/v) b^i(\mathbf{s}_t)\ d\mathbf{s}_t \tag{3.14}$$

As p_v^i is identical for all samples in the i-th episode regardless of t, the calculation of Equation 3.13 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:

1. Use a random behavior policy b^0 to traverse the feature set. Stop the traverse with the probability in Equation 3.12 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 3.7, and derive the updated policy π^1 and b^1 from Equation 3.8 and 3.9 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.

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

Input: Feature set $\mathcal{F} = \{f_1, f_2, ..., f_N\}$, downstream machine learning task $\mathcal{T}$.
Output: Optimal feature subset $\mathcal{F}'$.

Initialize the behavior policy b^1, target policy π^1, exploration number M, $\mathcal{F}' = \Phi$.
for $i = 1$ *to* M **do**
    Initialize state s_1^i.
    Rank features with their decision history.
    for $t = 1$ *to* N **do**
        Derive action a_t^i with behavior policy $b^i(s_t^i)$.
        Perform a_t^i, getting selected feature subset $\mathcal{F}_t^i$.
        Obtain the next state s_{t+1}^i by $represent(\mathcal{F}_t^i)$ and reward r_t^i by $eval(\mathcal{F}_t^i)$.
        Break the loop with probability p_t^i derived from Equation 3.12;
    end
    Update target policy π^{i+1} by Equation 3.8 and behavior policy b^{i+1} by Equation 3.9.
    if $eval(\mathcal{F}_N^i) > eval(\mathcal{F}')$ **then**
        $\mathcal{F}' = \mathcal{F}_N^i$.
    end
end
Return $\mathcal{F}'$.

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 $\pi^*_{\mathcal{M}}(s) = argmax_a Q^*_{\mathcal{M}}(s, a)$ for an MDP $\mathcal{M}$. The optimal Q-value can be updated by Bellman equation [5]:

$$Q^*_{\mathcal{M}}(s_t, a_t) = \mathcal{R}(s_t, a_t) + \gamma * max_{a_{t+1}} Q^*_{\mathcal{M}}(s_{t+1}, a_{t+1}) \tag{3.15}$$

Interactive reinforcement learning (IRL) is proposed to accelerate the learning process of reinforcement learning (RL) by providing external action advice to the RL agent [81]. As Figure 3.4

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 [87]. 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.

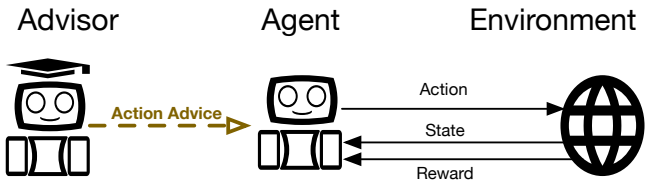


Figure 3.4: Classic interactive reinforcement learning. The advisor gives the agent advice at the action level.

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 $\mathcal{U}(s, a)$:

$$\mathcal{R}'(a_t, s_t) = \mathcal{R}(a_t, s_t) + c * (\gamma * \mathcal{U}(s_{t+1}) - \mathcal{U}(s_t)) \tag{3.16}$$

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 3.16:

Algorithm 3: Reward-Level Interactive Reinforcement Learning

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

We firstly subtract $c * \mathcal{U}(s_t)$ from both sides of Equation 3.15, and we have:

$$\begin{aligned} Q^*_{\mathcal{M}}(s_t, a_t) - c * \mathcal{U}(s_t) = \mathcal{R}(s_t, a_t) \\ + \gamma * max_{a_{t+1}} Q^*_{\mathcal{M}}(s_{t+1}, a_{t+1}) - c * \mathcal{U}(s_t) \end{aligned} \tag{3.17}$$

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

$$\begin{aligned} & Q^*_{\mathcal{M}}(s_t, a_t) - c * \mathcal{U}(s_t) = \mathcal{R}(s_t, a_t) \\ & + \gamma * max_{a_{t+1}} Q^*_{\mathcal{M}}(s_{t+1}, a_{t+1}) - c * \mathcal{U}(s_t) \\ & + c * \gamma * \mathcal{U}(s_{t+1}) - c * \gamma * \mathcal{U}(s_{t+1}) \\ & = \mathcal{R}(s_t, a_t) + c * \gamma * \mathcal{U}(s_{t+1}) - c * \mathcal{U}(s_t) \\ & + \gamma * max_{a_{t+1}} [Q^*_{\mathcal{M}}(s_{t+1}, a_{t+1}) - c * \mathcal{U}(s_{t+1})] \end{aligned} \tag{3.18}$$

We define:

$$Q^{\partial}_{\mathcal{M}'}(s_t, a_t) = Q^*_{\mathcal{M}}(s_t, a_t) - c * \mathcal{U}(s_t) \tag{3.19}$$

Then Equation 3.18 has the new form:

$$\begin{aligned} Q^{\partial}_{\mathcal{M}'}(s_t, a_t) &= \mathcal{R}(s_t, a_t) + c * [\gamma * \mathcal{U}(s_{t+1}) - \mathcal{U}(s_t)] \\ &+ \gamma * max_{a_{t+1}} Q^{\partial}_{\mathcal{M}'}(s_{t+1}, a_{t+1}) \\ &= \mathcal{R}'(s_t, a_t) + \gamma * max_{a_{t+1}} Q^{\partial}_{\mathcal{M}'}(s_{t+1}, a_{t+1}) \end{aligned} \tag{3.20}$$

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

$$Q^{*}_{\mathcal{M}'}(s_t, a_t) = Q^{\partial}_{\mathcal{M}'}(s_t, a_t) \tag{3.21}$$

We combine Equation 3.19 and Equation 3.21 and have:

$$Q^{*}_{\mathcal{M}'}(s_t, a_t) = Q^{*}_{\mathcal{M}}(s_t, a_t) - c * \mathcal{U}(s_t) \tag{3.22}$$

Obviously,

$$\begin{aligned} argmax_{a_t} Q^{*}_{\mathcal{M}'}(s_t, a_t) &= argmax_{a_t}[Q^{*}_{\mathcal{M}}(s_t, a_t) - c * \mathcal{U}(s_t)] \\ &= argmax_{a_t} Q^{*}_{\mathcal{M}}(s_t, a_t) \end{aligned} \tag{3.23}$$

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 [87] to select the advising states, i.e., the advisor gives advice for the first n states the IRL agent

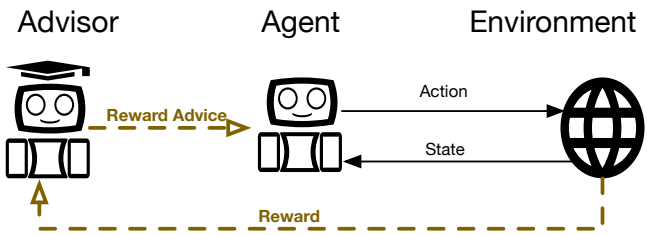


Figure 3.5: Reward-level interactive reinforcement learning. The advisor gives advice at the reward level.

meets.

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.

Experiments

Experimental Setup

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.

Data Description

We use four publicly available datasets on classification task to validate our methods, i.e., Forest Cover (FC) dataset [8], Spambase (Spam) dataset [17], Insurance Company Benchmark (ICB) dataset [89] and Arrhythmia (Arrhy) dataset [29]. The statistics of the datasets are in Table 3.2.

Table 3.2: Statistics of datasets.

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

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 the 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.

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 [106]. In the experiments, we set k equals to half of the number of input features.(2) **LASSO** conducts feature selection via $l1$ penalty [85]. 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 [53]. (4) **mRMR** ranks features by minimizing feature's redundancy and maximizing their relevance with the label[69]. (5) **RFE** selects features by recursively selecting smaller and smaller feature subsets [26].(6)**MARFS** is a multi-agent reinforcement learning based feature selection method [60]. It uses M feature agents to control the selection/deselection of the M features. Besides, we also compare our method with its variant without an early stopping strategy, i.e., Monte Carlo-based reinforced feature selection **MCRFS**.

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.

Environmental Settings

The experiments were carried out on a server with an I9-9920X 3.50GHz CPU, 128GB memory, and a Ubuntu 18.04 LTS operation system.

Performance Evaluation

Overall Performance

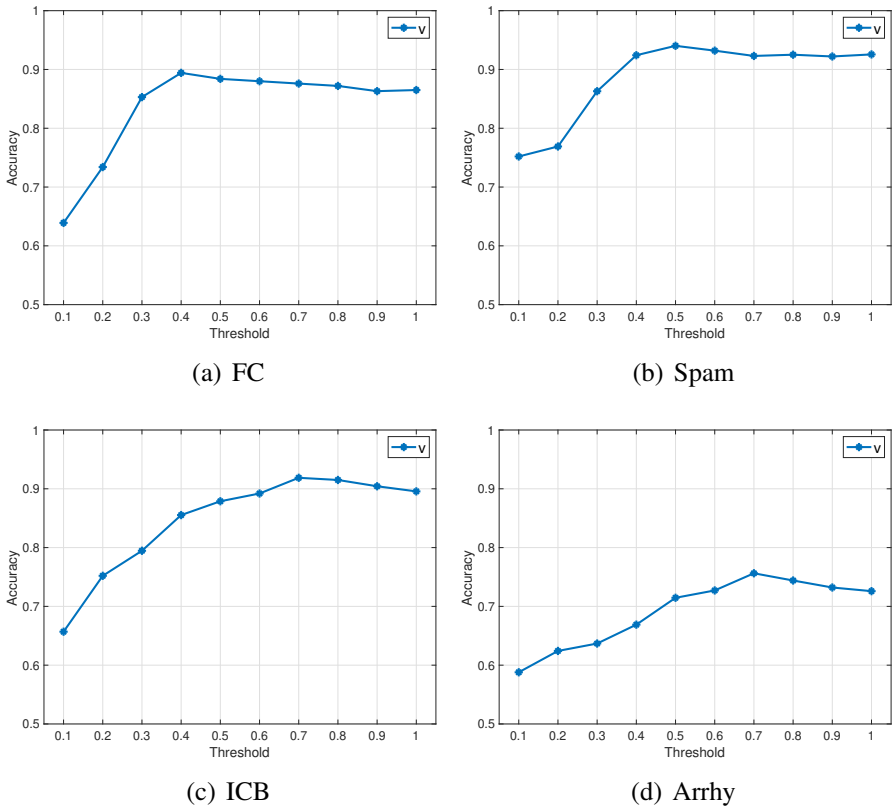
We compare the proposed ES-MCRFS method with baseline methods and its variants with regard to predictive accuracy. As Table 3.3 shows, the MCRFS, which simplifies the reinforced feature selection into a single-agent formulation, achieves similar performance with the multi-agent MARFS. With the help of a traverse strategy and early stopping criteria, the ES-MCRFS outperforms all the other methods.

Table 3.3: 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**

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 3.6 shows that the optimal threshold for the four datasets is 0.4, 0.5, 0.7, 0.7. It reveals that the early stopping criteria are sensitive to the pre-defined threshold, and the optimal threshold varies on different datasets.



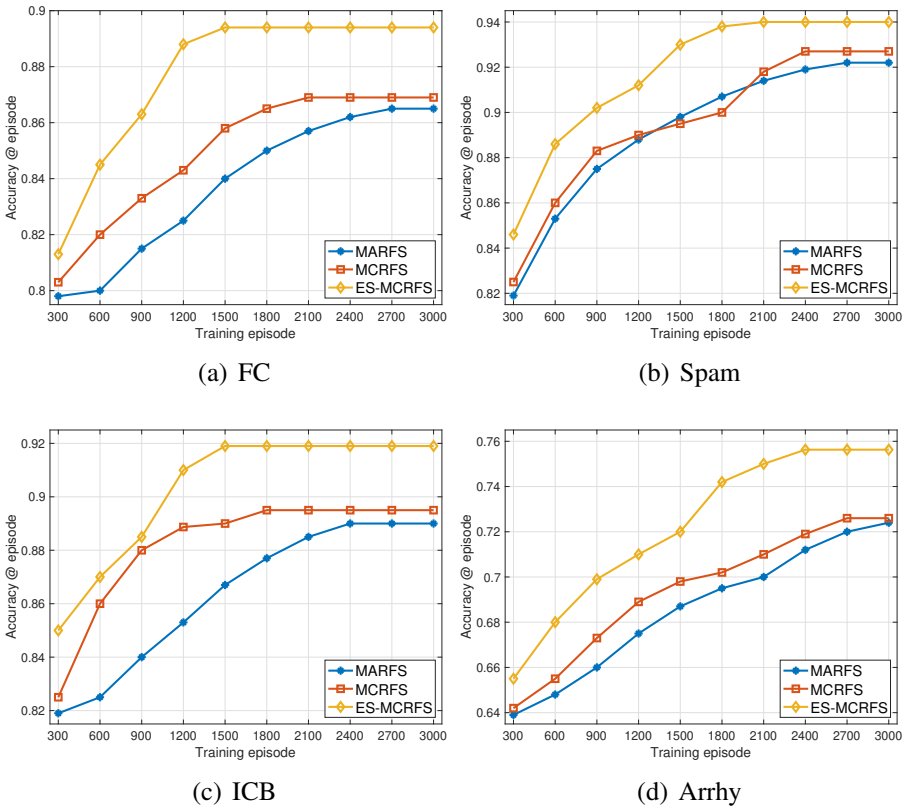
(a) FC (b) Spam

(c) ICB (d) Arrhy

Figure 3.6: Threshold sensitivity of early stopping criteria.

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 3.7 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.



(a) FC (b) Spam

(c) ICB (d) Arrhy

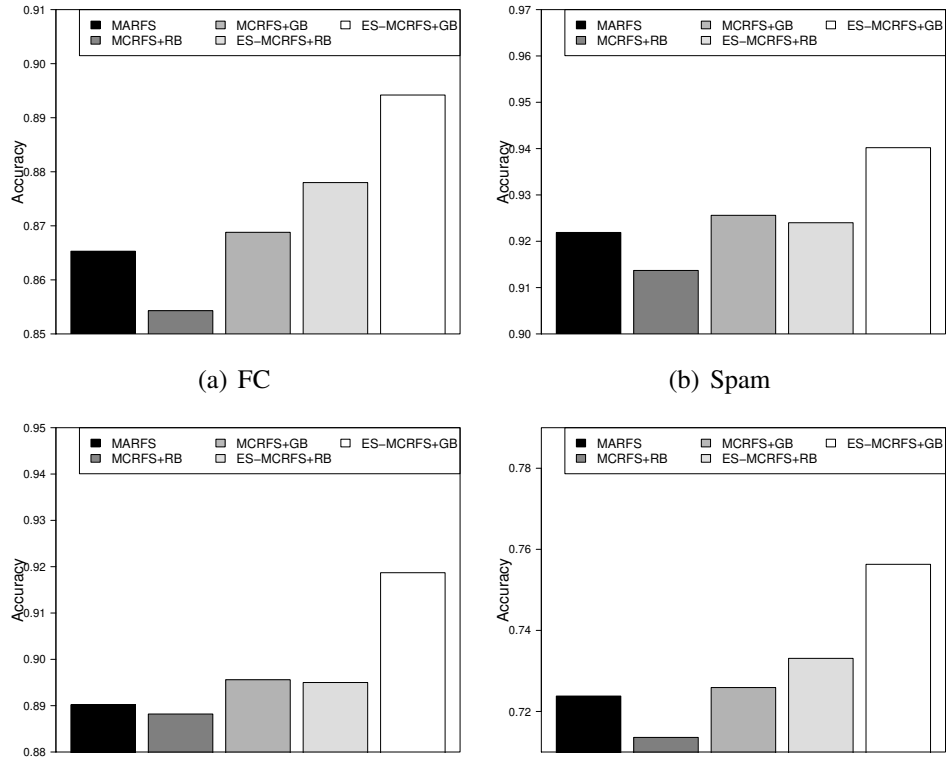
Figure 3.7: Predictive accuracy on training step.

Study of the Behavior Policy

We study the difference between random behavior policy and the ϵ-greedy policy presented in Equation 3.9. We combine the two policies with MCRFS and ES-MCRFS respectively. Figure 3.8 shows that the ϵ-greedy policy outperforms the random behavior policy on all datasets.

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 3.4 shows that the CPU and



(a) FC (b) Spam

(c) ICB (d) Arrhy

Figure 3.8: Predictive accuracy on different training strategies. RB for random behavior policy and GB for ϵ-greedy behavior policy.

memory cost when implementing the two methods. Our method MCRFS requires less computational resources than the multi-agent MARFS.

Table 3.4: 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

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 3.5 shows that the decision history can significantly improve performance of the traverse strategy.

Table 3.5: 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

Table 3.6: Performance with different utility function

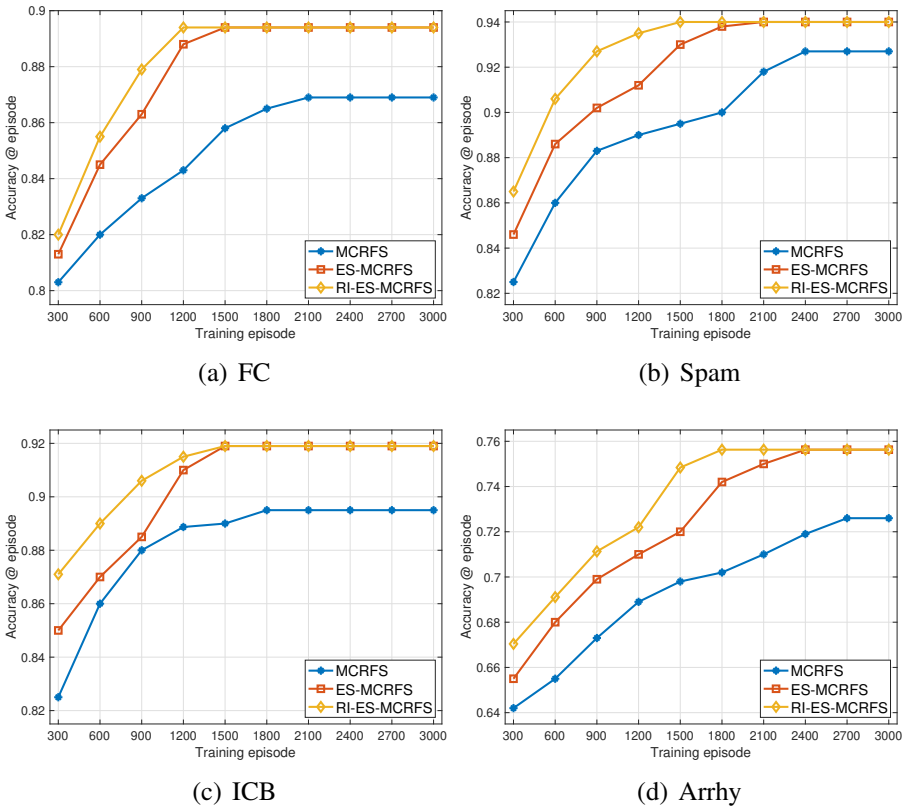
		FC		Spam		ICB		Arrhy	
		Acc	F1	Acc	F1	Acc	F1	Acc	F1
Utility	Rd	0.8689	0.8433	0.9250	0.8749	0.8997	0.8788	0.7340	0.6955
	Rv	0.8703	0.8507	0.9317	0.8831	0.9001	0.8793	0.7393	0.7143
	$Rv - Rd$	**0.8842**	**0.8650**	**0.9402**	**0.8949**	**0.9117**	**0.8903**	**0.7492**	**0.7258**

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 3.9 shows that with RI, the Monte Carlo reinforced feature selection can achieve convergence more quickly. However, as the ES-MCRFS already achieves good performance, the RI can not improve its final performance.

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 3.6



(a) FC (b) Spam

(c) ICB (d) Arrhy

Figure 3.9: Predictive accuracy on training step.

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.

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 [59] and marginalized importance sampling [104]. 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 [22]. However, these methods suffer from slow convergence and no theoretical guarantee [109]. Besides, there are other attempts to develop sample efficient reinforcement learning, such as curiosity-driven exploration and hybrid optimization [74, 98].

Feature Selection. Feature selection can be categorized into three types, i.e., filter methods, wrapper methods and embedded methods [62, 111]. Filter methods rank features only by relevance scores and only top-ranking features are selected. The representative filter methods is the univariate feature selection [21] The representative wrapper methods are branch and bound algorithms [67, 48]. Wrapper methods are supposed to achieve better performance than filter methods since they search on the whole feature subset space. Evolutionary algorithms [105, 46] 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 [85] and decision tree [82].

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 [58], 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 [76, 36], IRL sticks to learning from the environment and the advisor is only an advice-provider in its apprenticeship [47, 97]. 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 [87], the authors study the advising state selection and propose four advising strategies, i.e., early advising, importance advising, mistake correcting and predictive advising.

Conclusion Remarks

Summary. In this chapter, 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).

CHAPTER 4: GENERATIVE-AI PERSPECTIVE: AUTOREGRESSIVE FEATURE GENERATION LEARNING

In this chapter, I will introduce the framework of autoregressive feature generation learning. This framework embeds the knowledge of feature generation into a distinguishable embedding space, subsequently facilitating the identification of the optimal feature space.

Introduction

The objective of feature transformation is to derive a new feature space from the original features through the application of a series of operations, with the goal of enhancing the performance of subsequent machine learning tasks. However, feature transformation is usually manual, time-consuming, labor-intensive, and requires domain knowledge. These limitations motivate us to accomplish ***A****utomated* ***F****eature* ***T****ransformation* (**AFT**). AFT is a fundamental task because AFT can 1) reconstruct distance measures, 2) form a feature space with discriminative patterns, 3) ease machine learning, and 4) overcome complex and imperfect data representation.

There are two main challenges in solving AFT: 1) efficient feature transformation in a massive discrete search space; 2) robust feature transformation in an open learning environment. Firstly, it is computationally costly to reconstruct the optimal feature space from a given feature set. Such reconstruction requires a transformation operation sequence that includes features-operations combinations, each of which represents a newly generated feature. Since there are many features and operations, the number of feature-operations combinations exponentially grows, not to mention the number of candidate transformation operation sequences. The efficiency challenge seeks to answer: *how can we efficiently identify the best feature transformation operation sequence?* Sec-

ondly, identifying the best transformation operation sequence is unstable and sensitive to many factors in an open environment. For example, if we see identifying a transformation operation sequence as a searching problem, it is sensitive to starting points or the greedy strategy during iterative searching. If we see identifying transformation paths as a generation problem, it is sensitive to training data quality and the complexity of generation forms. The robustness challenge aims to answer: *how can we robustify the generation of feature transformation operation paths?*

Prior literature only partially addresses the two challenges. Existing AFT algorithms can be grouped into three categories: 1) expansion-reduction approaches [40, 38, 45], in which all mathematical operations are randomly applied to all features at once to generate candidate transformed features, followed by feature selection to choose valuable features. However, such methods are based on random generation, unstable, and not optimization-directed. 2) iterative-feedback approaches [93, 44, 88], in which integrates feature generation and selection are integrated, and generation and selection strategies are updated based on feedback in each iteration. Two example methods are Evolutionary Algorithms (EA) or Reinforcement Learning (RL) with downstream ML task accuracy as feedback. However, such methods are developed based on searching in a massive discrete space and are difficult to converge on compared with solving a continuous optimization problem. 3) Neural Architecture Search (NAS)-based approaches [12, 113]. NAS was originally to identify neural network architectures using a discrete search space containing neural architecture parameters. Inspired by NAS, some studies formulated AFT as a NAS problem. However, NAS-based formulations are slow and limited in modeling all transformation forms. Existing studies demonstrate the inability to address efficiency and robustness in feature transformation jointly. Therefore, we need a novel perspective to derive a novel formulation and solver of AFT.

Our Contribution: A Postfix Expression Embedding and Generation Perspective. We formulate the discrete AFT problem as a continuous optimization task. We highlight a postfix expression embedding and generation perspective for feature transformation: feature transformation is

described as a transformation operation sequence; the sequence is represented by a postfix expression, thereafter, embedded into a vector in a continuous latent space, with the continuous space, the best transformation identification is solved by an efficient gradient-based optimizer; finally a new feature set is reconstructed from the optimal embedding vector via generative modeling. We show that training data quality directly defines an effective embedding space, without which gradient-based optimization and generative reconstruction will not work well. Collecting quality feature transformation training data is never easy. We demonstrate that reinforcement intelligence can be used as a self-optimizing training data collector to explore high-quality feature transformations, evaluate downstream ML task accuracy, and automate training data collection. We find that integrating transformation sequence reconstruction loss and downstream task accuracy estimation loss can better measure the effectiveness of an embedding space and strengthen the denoising capability of gradient-based search of the optimal transformation embedding. We observe that beam search exhibits the ability to track multiple top candidate sequences during sequence reconstruction and increase the validity of generated feature transformation operation. We encode a feature transformation operation sequence as a single postfix expression so that the generative model can capture feature-feature interactions and automatically identify the dimensionality of the new feature space and the complexity of the transformation operation sequence.

Summary of Proposed Approach. Inspired by these findings, this paper presents a generic and principled framework for deep differentiable feature transformation, namely **GBFG**. To advance efficiency and robustness, this framework implements four steps: 1) reinforcement training data collection; 2) postfix expression embedding of transformation operation path; 3) gradient-based continuous optimization; 4) beam search-based transformation operation path reconstruction. Step 1 is to collect high-quality transformation operation sequence-accuracy pairs as training data. Specifically, we develop a cascading reinforcement learning structure to automatically explore transformation operation sequences and test generated feature spaces on a downstream predictor

(e.g., decision tree). The self-optimizing policies enable agents to collect high-quality transformation operation sequences. The key insight is that when training data is hard to collect, reinforcement intelligence can be used as an automated training data collector. Step 2 is to learn a continuous embedding space from transformation-accuracy training data. Specifically, we describe transformation operation sequences as postfix expressions, each of which is mapped into an embedding vector by jointly optimizing the transformation operation path reconstruction loss and accuracy estimation loss. Viewing feature transformation from the lens of postfix expression can reduce an exponentially growing discrete search problem into an alphabetical letter generation problem. The postfix reformulation has a much smaller space by only selecting an alphabetical feature-related or operator-related letter in each generation instead of considering high-order expansions. The postfix reformulation equips the generation model with the ability to self-decide the best number of generated features and the transformation operation segmentation of each developed feature. Step 3 is to leverage the gradient calculated from the improvement of the accuracy evaluator to guide the search for the optimal transformation operation sequence embedding. Step 4 is to develop a beam search-based generative module to reconstruct feature transformation operation sequences from embedding vectors. Finally, we present extensive experiments and case studies to show the enhanced performances of our framework.

Definitions and Problem Statement

Important Definitions

Definition 4 ***Operation Set.*** *To refine the feature space, we need to apply mathematical operations to existing features to generate new ones. All operations are collected in an operation set, denoted by $\mathcal{O}$. Based on the computation property, these operations can be classified as unary operations and binary operations. The unary operations such as "square", "exp", "log", etc. The binary*

operations such as "plus", "multiply", "minus", etc.

Definition 5 ***Cascading Agent Structure.*** *We create a cascading agent structure comprised of feature agent1, operation agent, and feature agent2 to efficiently collect quantities of high-quality feature transformation records. The selection process of the three agents will share the state information and sequentially select candidate features and operations for refining the feature space.*

Definition 6 ***Feature Transformation Operation Sequence.*** *Assuming that $D = \{X, y\}$ is a dataset, which includes the original feature set $X = [f_1, \cdots, f_N]$ and predictive targets y. As shown in Figure 4.1, we transform the existing ones using mathematical compositions τ consisting of feature ID tokens and operations to generate new and informative features. K transformation compositions are adopted to refine X to a better feature space $\tilde{X} = [\tilde{f}_1, \cdots, \tilde{f}_K]$. The collection of the K compositions refers to the feature transformation sequence, which is denoted by $\Gamma = [\tau_1, \cdots, \tau_K]$.*

$$\tau_1 : (f_1 + (\frac{(f_1 + f_2)}{f_3}) - f_2), \tau_2 : (f_1 - (\sqrt{f_3})), \cdots, \tau_K : ((f_2)^2)$$

Figure 4.1: An example of feature transformation sequence.

Problem Statement

We aim to develop an effective and robust deep differentiable automated feature transformation framework. Formally, given a dataset $D = \{X, y\}$ and an operation set $\mathcal{O}$, we first build a cascading RL-agent structure to collect n feature transformation accuracy pairs as training data, denoted by $R = \{(\Gamma_i, v_i)\}_{i=1}^{n}$, where Γ_i is the transformation sequence and v_i is the associated downstream predictive performance. We pursue two objectives thereafter: 1) building an optimal continuous

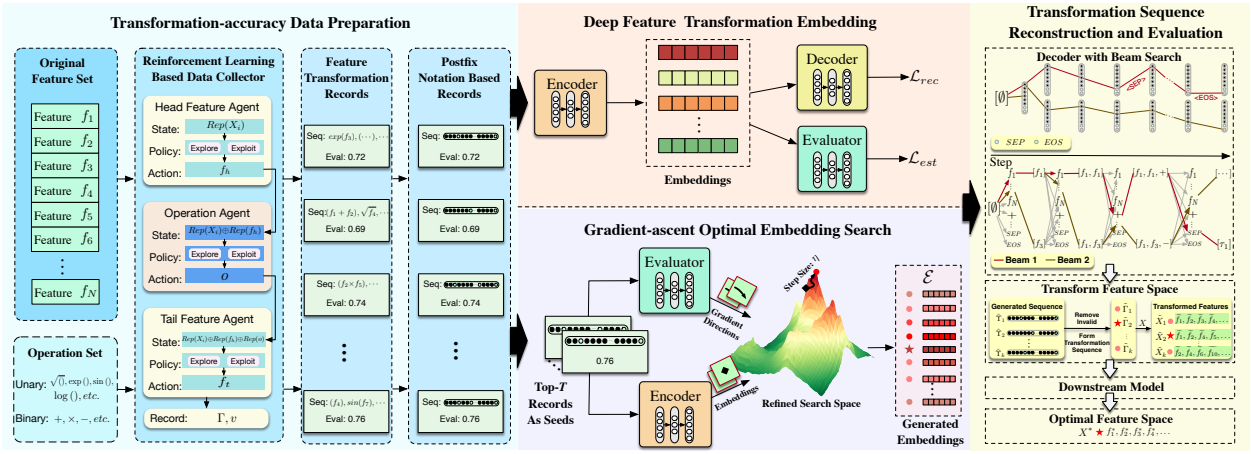


Figure 4.2: An overview of our framework. GBFG consists of four main components: 1) transformation-accuracy data preparation; 2) deep feature transformation embedding; 3) gradient-ascent optimal embedding search; 4) transformation sequence reconstruction and evaluation.

embedding space for feature transformation sequences. We learn a mapping function ϕ, a reconstructing function ψ, and an evaluation function ω to convert R into a continuous embedding space $\mathcal{E}$ via joint optimization. In $\mathcal{E}$, each embedding point is associated with a feature transformation sequence and corresponding predictive performance. 2) identifying the optimal feature space. We adopt a gradient-based search to find the optimal feature transformation sequence Γ^*, given by:

$$\Gamma^* = \psi(\mathbf{E}^*) = \underset{\mathbf{E} \in \mathcal{E}}{\operatorname{argmax}} \mathcal{A}(\mathcal{M}(\psi(\mathbf{E})(X)), y), \tag{4.1}$$

where ψ can reconstruct a feature transformation sequence from any embedding point of $\mathcal{E}$; $\mathbf{E}$ is an embedding vector in $\mathcal{E}$ and $\mathbf{E}^*$ is the optimal one; $\mathcal{M}$ is the downstream ML model and $\mathcal{A}$ is the performance indicator. Finally, we apply Γ^* to transform X to the optimal feature space X^* maximizing the value of $\mathcal{A}$.

Methodology

Framework Overview

Figure 5.1 shows the framework of GBFG including four steps: 1) reinforcement transformation-accuracy data preparation; 2) deep postfix feature transformation operation sequence embedding; 3) gradient-ascent optimal embedding search; 4) transformation operation sequence reconstruction.

In Step 1, a cascading agent structure consisting of two feature agents and one operation agent is developed to select candidate features and operators for feature crossing. The transformed feature sets are applied to a downstream ML task to collect the corresponding accuracy. The data collection process is automated and self-optimized by policies and feedback in reinforcement learning. We then convert these feature transformation operation sequences into postfix expressions. In Step 2, we develop an encoder-evaluator-decoder model to embed transformation operation sequence-accuracy pairs into a continuous embedding space by jointly optimizing the sequence reconstruction loss and performance evaluation loss. In detail, the encoder maps these transformation operation sequences into continuous embedding vectors; the evaluator assesses these embeddings by predicting their corresponding model performance; the decoder reconstructs the transformation sequence using these embeddings. In Step 3, we first learn the embeddings of top-ranking transformation operation sequences by the well-trained encoder. With these embeddings as starting points, we search along the gradient induced by the evaluator to find the acceptable optimal embeddings with better model performances. In Step 4, the well-trained decoder then decodes these optimal embeddings to generate candidate feature transformation operation sequences through the beam search. We apply the feature transformation operation sequences to original features to reconstruct refined feature spaces and evaluate corresponding downstream predictive performances. Finally, the best feature space is chosen as the optimal one.

Reinforcement Training Data Preparation

Why Using Reinforcement as Training Data Collector. Our extensive experimental analysis shows that the quality of embedding space directly determines the success of feature transformation operation sequence construction. The quality of the embedding space is sensitive to the quality and scale of transformation operation sequence-accuracy training data: training data is large enough to represent the entire distribution; training data include high-performance feature transformation cases, along with certain random exploratory samples. Intuitively, we can use random sample features and operations to generate feature transformation sequences. This strategy is inefficient because it produces many invalid and low-quality samples. Or, we can use existing feature transformation methods (e.g., AutoFeat [38]) to generate corresponding records. However, these methods are not fully automated and produce a limited number of high-quality feature transformation records without exploration ability. We propose to view reinforcement learning as a training data collector to overcome these limitations.

Reinforcement Transformation-Accuracy Training Data Collection. Inspired by [93, 103], we formulate feature transformation as three interdependent Markov decision processes (MDPs). We develop a cascading agent structure to implement the three MDPs. The cascading agent structure consists of a head feature agent, an operation agent, and a tail feature agent. In each iteration, the three agents collaborate to select two candidate features and one operation to generate a new feature. Feedback-based policy learning is used to optimize the exploratory data collection to find diversified yet quality feature transformation samples.

To simplify the description, we adopt the i-th iteration as an example to illustrate the reinforcement data collector. Given the former feature space as X_i, we generate new features X_{i+1} using the head feature f_h, operation o, and tail feature f_t selected by the cascading agent structure.

1) Head feature agent. This learning system includes: *State:* is the vectorized representation of X_i. Let $Rep(\cdot)$ be a state representation method, and the state can be denoted by $Rep(X_i)$. *Action:* is the head feature f_h selected from X_i by the reinforced agent.

2) Operation agent. This learning system includes: *State:* includes the representation of X_i and the head feature, denoted by $Rep(X_i) \oplus Rep(f_h)$, where $\oplus$ indicates concatenation. *Action:* is the operation o selected from the operation set $\mathcal{O}$.

3) Tail feature agent. This learning system includes: *State:* includes the representation of X_i, selected head feature, and operation, denoted by $Rep(X_i) \oplus Rep(f_h) \oplus Rep(o)$. *Action:* is the tail feature f_t selected from X_i by this agent.

4) State representation method, $Rep(\cdot)$. For the representation of the feature set, we employ a descriptive statistical technique to obtain the state with a fixed length. In detail, we first compute the descriptive statistics (*i.e.* count, standard deviation, minimum, maximum, first, second, and third quantile) of the feature set column-wise. Then, we calculate the same descriptive statistics on the output of the previous step. After that, we can obtain the descriptive matrix with shape $\mathbb{R}^{7\times7}$ and flatten it as the state representation with shape $\mathbb{R}^{1\times49}$. For the representation of the operation, we adopt its one-hot encoding as $Rep(o)$.

4) Reward function. To improve the quality of the feature space, we use the improvement of a downstream ML task performance as the reward. Thus, it can be defined as:

$$\mathcal{R}(X_i, X_{i+1}) = \mathcal{A}(\mathcal{M}(X_{i+1}), y) - \mathcal{A}(\mathcal{M}(X_i), y). \tag{4.2}$$

5) Learning to Collect Training Data. To optimize the entire procedure, we minimize the mean squared error of the Bellman Equation to get a better feature space. During the exploration pro-

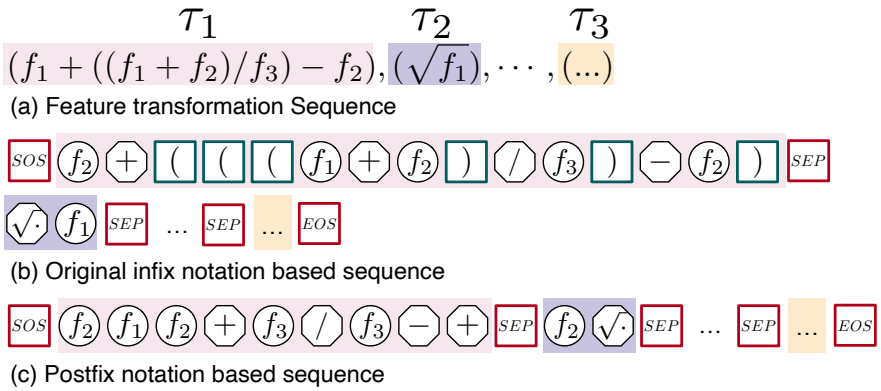


Figure 4.3: An example of feature transformation sequence and postfix notation-based transformation expression.

cess, we can collect amounts of high-quality records (Γ, v) for constructing an effective continuous embedding space, where Γ is the transformation sequence, and v is the downstream model performance.

Postfix Expressions of Feature Transformation Operation Sequences

Why Transformation Operation Sequences as Postfix Expressions. After training data collection, a question arises: how can we organize and represent these transformation operation sequences in a computationally-tangible and machine-learnable format?

Figure 4.3 (a) shows an example of a feature transformation operation sequence. To convert this sequence into a machine-readable expression, a naive idea is to enclose each calculation in a pair of brackets to indicate its priority (Figure 4.3(b)). However, the representation of Figure 4.3(b) has four limitations: (1) *Redundancy.* Many priority-related brackets are included to ensure the unambiguous property and correctness of mathematical calculations. (2) *Semantic Sparsity.* The quantity of bracket tokens can dilute the semantic information of the sequence, making model convergence difficult. (3) *Illegal Transformation.* If the decoder makes one mistake on bracket generation, the entire generated sequence will be wrong. (4) *Large Search Space.* The number of

combinations of features and operators from low-order to high-order interactions is large, making the search space too vast.

Using Postfix Expression to Construct Robust, Concise, and Unambiguous Sequences. To address the aforementioned limitations, we convert the transformation operation sequence Γ into a postfix-based sequence expression. Specifically, we scan each mathematical composition τ in Γ from left to right and convert it from the infix-based format to the postfix-based one. We then concatenate each postfix expression by the <SEP> token and add the <SOS> and <EOS> tokens to the beginning and end of the entire sequence. Figure 4.3(c) shows an example of such a postfix sequence. We denote it as $\Upsilon = [\gamma_1, \cdots, \gamma_M]$, where each element is a feature ID token, operation token, or three other unique tokens. The detailed pseudo code of this conversion process is provided in the Appendix.

The postfix sequences don't require numerous brackets to ensure the calculation priority. We only need to scan each element in the sequence from left to right to reconstruct corresponding transformed features. Such a concise and short expression can reduce sequential modeling difficulties and computational costs. Besides, a postfix sequence indicates a unique transformation process, thus, reducing the ambiguity of feature transformation and yielding a more robust feature space. Moreover, the most crucial aspect is the reduction of the search space from exponentially growing discrete combinations to a limited token set $\mathcal{C}$ that consists of the original feature ID tokens, operation tokens, and other three unique tokens. The length of the token set is $|\mathcal{O}| + |X| + 3$, where $|\mathcal{O}|$ is the number of the operation set, $|X|$ is the dimension of the original feature set, and 3 refers to the unique tokens <SOS>, <SEP>, <EOS>.

Data Augmentation for Postfix Transformation Operation Sequences. Big and diversified transformation operation sequence-accuracy training data can benefit the learning of a pattern discriminative embedding space. Be sure to notice that, a feature transformation operation sequence

consists of many independent segmentations, each of which is the composition of feature ID and operation tokens that are used to generate a single feature in the new feature space. These independent segmentations are order-agnostic. Our idea is to leverage the order agnostic characteristic to conduct data augmentation to increase the data volume and diversity for constructing a more robust and effective embedding space. For example, given a transformation operation sequence and corresponding accuracy $\{\Upsilon, v\}$, we first divide the postfix expression into different segmentations by <SEP>. We then randomly shuffle these segmentations and use <SEP> to concatenate them together to generate new postfix transformation sequences. After that, we pair the new sequences with the corresponding model accuracy performance to improve data diversity and data volume for better model training and to create the continuous embedding space.

Deep Feature Transformation Embedding

After collecting and converting large-scale feature transformation training data to a set of postfix expression-accuracy pairs $\{(\Upsilon_i, v_i)\}_{i=1}^n$, we develop an encoder-evaluator-decoder structure to map the sequential information of these records into an embedding space. Each embedding vector is associated with a transformation operation sequence and its corresponding model accuracy.

The Encoder ϕ: The Encoder aims to map any given postfix expression to an embedding (a.k.a., hidden state) $\mathbf{E}$. We adopt a single layer long short-term memory [37] (LSTM) as Encoder and acquire the continuous representation of Υ, denoted by $\mathbf{E} = \phi(\Upsilon) \in \mathbf{R}^{M \times d}$, where M is the total length of input sequence Υ and d is the hidden size of the embedding.

The Decoder ψ: The Decoder aims to reconstruct the postfix expression of the feature transformation operation sequence Υ from the hidden state $\mathbf{E}$. In GBFG, we set the backbone of the Decoder as a single-layer LSTM. For the first step, ψ will take an initial state (denoted as h_0) as input. Specifically, in step-i, we can obtain the decoder hidden state h_i^d from the LSTM. We use the dot

product attention to aggregate the encoder hidden state and obtain the combined encoder hidden state h_i^e. Then, the distribution in step-j can be defined as:

$$P_\psi(\gamma_i|\mathbf{E}, \Upsilon_{<i}) = \frac{\exp(W_{\gamma_i}(h_i^d \oplus h_i^e))}{\sum_{c\in\mathcal{C}} \exp(W_c(h_i^d \oplus h_i^e))}, \tag{4.3}$$

where $\gamma_i \in \Upsilon$ is the i-th token in sequence Υ, and $\mathcal{C}$ is the token set. W stand for the parameter of the feedforward network. $\Upsilon_{<i}$ represents the prediction of the previous or initial step. By multiplying the probability in each step, we can form the distribution of each token in Υ, given as:$P_\psi(\Upsilon|\mathbf{E}) = \prod_{i=1}^{M} P_\psi(\gamma_i|\mathbf{E}, \Upsilon_{<i})$. To make the generated sequence similar to the real one, we minimize the negative log-likelihood of the distribution, defined as: $\mathcal{L}_{rec} = -\log P_\psi(\Upsilon|\mathbf{E})$.

The Evaluator ω: The Evaluator is designed to estimate the quality of continuous embeddings. Specifically, we will first conduct mean pooling on $\mathbf{E}$ by column to aggregate the information and obtain the embedding $\bar{\mathbf{E}} \in \mathbf{R}^d$. Then $\bar{\mathbf{E}}$ is input into a feedforward network to estimate the corresponding model performance, given as: $\hat{v} = \omega(\mathbf{E})$. To minimize the gap between estimated accuracy and real-world gold accuracy, we leverage the Mean Squared Error (MSE) given by: $\mathcal{L}_{est} = \text{MSE}(v, \omega(\mathbf{E}))$.

Joint Training Loss $\mathcal{L}$: We jointly optimize the encoder, decoder, and evaluator. The joint training loss can be formulated as: $\mathcal{L} = \alpha\mathcal{L}_{rec} + (1-\alpha)\mathcal{L}_{est}$ where α is the trade-off hyperparameter that controls the contribution of sequence reconstruction and accuracy estimation loss.

Gradient-Ascent Optimal Embedding Search

To conduct the optimal embedding search, we first select top-T feature transformation operation sequences ranked by the downstream predictive accuracy. The well-trained encoder is then used to embed these postfix expressions into continuous embeddings, which later will be used as seeds

(starting points) of gradient ascent. Assuming that one search seed embedding is $\mathbf{E}$, we search, starting from $\mathbf{E}$, toward the gradient direction induced by the evaluator ω:

$$\tilde{\mathbf{E}} = \mathbf{E} + \eta \frac{\partial \omega}{\partial \mathbf{E}}, \tag{4.4}$$

where $\tilde{\mathbf{E}}$ denotes the refined embedding, η is the size of each searching step. The model performance of $\tilde{\mathbf{E}}$ is supposed to be better than $\mathbf{E}$ due to $\omega(\tilde{\mathbf{E}}) \geq \omega(\mathbf{E})$. For T seeds, we can obtain the enhanced embeddings $[\tilde{\mathbf{E}}_1, \tilde{\mathbf{E}}_2, \cdots, \tilde{\mathbf{E}}_T]$.

Transformation Operation Sequence Reconstruction and Evaluation

We reconstruct the transformation sequences by the well-trained decoder ψ using the collected candidate (i.e., acceptable optimal) embeddings $[\tilde{\mathbf{E}}_1, \tilde{\mathbf{E}}_2, \cdots, \tilde{\mathbf{E}}_T]$. This process can be denoted by:

$$[\tilde{\mathbf{E}}_1, \tilde{\mathbf{E}}_2, \cdots, \tilde{\mathbf{E}}_T] \xrightarrow{\psi} \{\tilde{\Upsilon}_i\}_{i=1}^{T}. \tag{4.5}$$

To identify the best transformation sequence, we adopt the beam search strategy [23, 91, 4] to generate feature transformation operation sequence candidates. Specifically, given a refined embedding $\tilde{\mathbf{E}}$, at step-t, we maintain the historical predictions with beam size b, denoted as $\{\Upsilon^i_{<t}\}_{i=1}^{b}$. For the i-th beam, the probability distribution of the token identified by the well-trained decoder ψ at the t-th step is is γ, which can be calculated as follows:

$$P^i_t(\gamma) = P_\psi(\gamma|\tilde{\mathbf{E}}, \tilde{\Upsilon}^i_{<t}) * P_\psi(\tilde{\Upsilon}^i_{<t}|\tilde{\mathbf{E}}), \tag{4.6}$$

where the probability distribution $P^i_t(\gamma)$ is the continued multiplication between the probability distribution of the previous decoding sequence and that of the current decoding step. We can collect the conditional probability distribution of all tokens for each beam. After that, we append

tokens with top-b probability values to the historical prediction of each beam to get a new historical set $\{\tilde{\Upsilon}^i_{<t+1}\}_{i=1}^b$. We can iteratively conduct this decoding process until confronted with the ¡EOS¿ token. We select the transformation sequence with the highest probability value as output.

Hence, T enhanced embeddings may produce T transformation sequences $\{\tilde{\Upsilon}_i\}_{i=1}^T$. We divide each of them into different parts according to the ¡SEP¿ token and check the validity of each part and remove invalid ones. Here, the validity measures whether the mathematical compositions represented by the postfix part can be successfully calculated to produce a new feature. These valid postfix parts reconstruct a feature transformation operation sequence $\{\tilde{\Gamma}_i\}_{i=1}^T$, which are used to generate refined feature space $\{\tilde{X}_i\}_{i=1}^T$. Finally, we select the feature set with the highest ML performance as the optimal feature space $\mathbf{X}^*$.

Compared with Prior Literature

Recent studies tried to convert the feature transformation into a continuous optimization task to search the optimal feature space efficiently. *DIFER* [113] is a cutting-edge method that is comparable to our work in the problem formulation. However, the following constraints limit its practicality: 1) DIFER collects transformation-accuracy data at random, resulting in many invalid training data with inconsistent transformation performances; 2) DIFER embeds and reconstructs each transformed feature separately and, thus, ignores feature-feature interactions; 3) DIFER needs to manually decide the number of generated features, making the reconstruction process ad-hoc. 4) the greedy search for transformation reconstruction in DIFER leads to suboptimal transformation results. To fill these gaps, we first implement an RL-based data collector to automate high-quality transformation record collection. We then leverage the postfix expression idea to represent the entire transformation operation sequence to model feature interactions and automatically identify the number of reconstructed features. Moreover, we employ beam search to advance the robustness,

quality, and validity of transformation operation sequence reconstruction.

Experiments

Experiment Setup

Datasets

We used 23 publicly available datasets from UCI [73], LibSVM [14], Kaggle [39], and OpenML [72] to conduct experiments. The 23 datasets involve 14 classification tasks and 9 regression tasks. Table 4.1 shows the statistics of these datasets.

Evaluation Metrics

We used F1-score, Precision, Recall, and ROC/AUC to evaluate classification tasks. We used 1-Relative Absolute Error (1-RAE) [93], 1-Mean Average Error (1-MAE), 1-Mean Square Error (1-MSE), and 1-Root Mean Square Error (1-RMSE) to evaluate regression tasks. We used the Valid Rate to evaluate the sequence generation. A valid feature transformation means it can successfully conduct mathematical compositions without any ambiguity and errors. The valid rate is the average of all correct transformation numbers divided by the total number of generated sequences. The greater the valid rate is, the superior the model performance is. Because it indicates that the built continuous space can capture the patterns of mathematical compositions, resulting in more effective exploration and fewer errors during searching for better feature transformations.

Baselines

We compared our method with eight widely-used feature generation methods: (1) **RDG** generates feature-operation-feature transformation records at random for generating new feature space; (2) **ERG** first applies operation on each feature to expand the feature space, then selects the crucial features as new features. (3) **LDA** [9] is a matrix factorization-based method to obtain the factorized hidden state as the generated feature space. (4) **AFAT** [38] is an enhanced version of ERG that repeatedly generate new features and use multi-step feature selection to select informative ones. (5) **NFS** [12] models the transformation sequence of each feature and uses RL to optimize the entire feature generation process. (6) **TTG** [44] formulates the transformation process as a graph, then implements an RL-based search method to find the best feature set. (7) **GRFG** [93] uses three collaborated reinforced agents to conduct feature generation and proposes a feature grouping strategy to accelerate agent learning. (8) **DIFER** [113] embeds randomly generated feature transformation records with a seq2seq model, then employs gradient search to find the best feature set.

Besides, we developed two variants of GBFG in order to validate the impact of each technical component: (i) **GBFG**$^{-d}$ replaces the RL-based data collection component with collecting feature transformation-accuracy pairs at random. (ii) **GBFG**$^{-a}$ removes the data augmentation component. We randomly split each dataset into two independent sets. The prior 80% is the training set, and the remaining 20% is the testing set. We conducted all experiments with the hold-out setting to ensure a fair comparison. This experimental setting means that the optimal feature transformation operation sequence learned on the training set was directly applied to the testing set without any data leakage. We adopted Random Forest as the downstream machine learning model and reported the performance of each method by running five-fold cross-validation on the testing set. Random Forest is a robust, stable, well-tested method, thus, we can reduce performance variation caused by the model, and make it easy to study the impact of feature transformation.

Hyperparameter Settings and Reproducibility

The operation set consists of *square root, square, cosine, sine, tangent, exp, cube, log, reciprocal, quantile transformer, min-max scale, sigmoid, plus, subtract, multiply, divide*. For the data collection part, we ran the RL-based data collector for 512 epochs to collect a large amount of feature transformation-accuracy pairs. For the data augmentation part, we randomly shuffled each transformation sequence 12 times to increase data diversity and volume. We adopted a single-layer LSTM as the encoder and decoder backbones and utilized 3-layer feed-forward networks to implement the predictor. The hidden state sizes of the encoder, decoder and predictor are 64, 64, and 200, respectively. The embedding size of each feature ID token and operation token was set to 32. To train GBFG, we set the batch size as 1024, the learning rate as 0.001, and λ as 0.95 respectively. For inferring new feature transformation sequences, we used top-20 records as the seeds with beam size 5.

Experimental Settings

All experiments were conducted on the Ubuntu 18.04.6 LTS operating system, AMD EPYC 7742 CPU, and 8 NVIDIA A100 GPUs, with the framework of Python 3.9.10 and PyTorch 1.8.1.

Performance Evaluation

Overall Performance

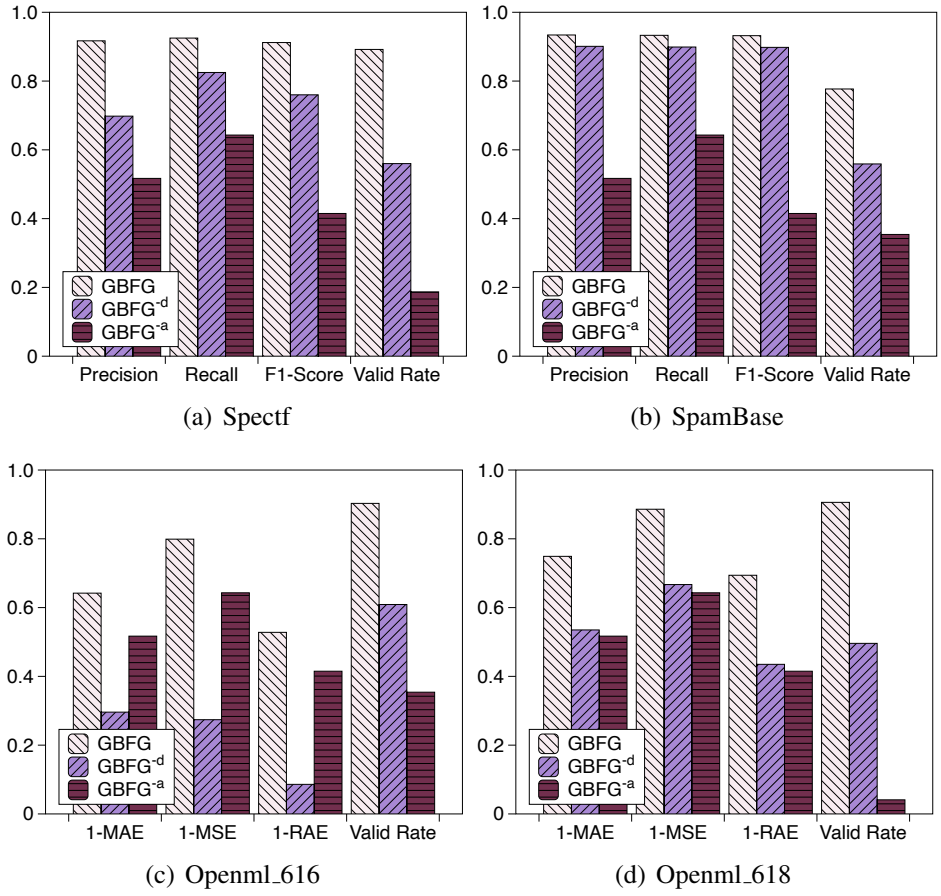
This experiment aims to answer: *Can GBFG effectively generate transformation sequence with excellent performance?* Table 4.1 shows the comparison results in terms of F1-score and 1-RAE. We noticed that GBFG outperforms other cutting-edge baselines on all datasets. The underly-

ing driver for this observation is that GBFG builds an effective embedding space to preserve the knowledge of feature transformation, making sure the gradient-ascent search module can identify the best-transformed feature space following the gradient direction. Another interesting observation is that GBFG significantly outperforms DIFER and has more stable performance. There are two underlying drivers: 1) RL-based data collector produces large quantities of high-quality transformation records, which provides a robust and powerful foundation for constructing an effective embedding space; 2) Postfix notation-based transformation sequence greatly decreases the search space and learning difficulties, resulting in a better transformation performance. Thus, this experiment validates that GBFG is effective and superior when confronted with different datasets and scenarios.

Table 4.1: Overall performance comparison. 'C' for binary classification, and 'R' for regression. The best results are highlighted in **bold**. The second-best results are highlighted in underline. (**Higher values indicate better performance.**)

Dataset	Source	C/R	Samples	Features	RDG	ERG	LDA	AFAT	NFS	TTG	GRFG	DIFER	GBFG
Higgs Boson	UCIrvine	C	50000	28	0.695	0.702	0.513	0.697	0.691	0.699	0.707	0.669	**0.712**
Amazon Employee	Kaggle	C	32769	9	0.932	0.934	0.916	0.930	0.932	0.933	0.932	0.929	**0.936**
PimaIndian	UCIrvine	C	768	8	0.760	0.761	0.638	0.765	0.749	0.745	0.754	0.760	**0.807**
SpectF	UCIrvine	C	267	44	0.760	0.757	0.665	0.760	0.792	0.760	0.818	0.766	**0.912**
SVMGuide3	LibSVM	C	1243	21	0.787	0.826	0.652	0.795	0.792	0.798	0.812	0.773	**0.849**
German Credit	UCIrvine	C	1001	24	0.680	0.740	0.639	0.683	0.687	0.645	0.683	0.656	**0.730**
Credit Default	UCIrvine	C	30000	25	0.805	0.803	0.743	0.804	0.801	0.798	0.806	0.796	**0.810**
Messidor_features	UCIrvine	C	1150	19	0.624	0.669	0.475	0.665	0.638	0.655	0.692	0.660	**0.749**
Wine Quality Red	UCIrvine	C	999	12	0.466	0.461	0.433	0.480	0.462	0.467	0.470	0.476	**0.559**
Wine Quality White	UCIrvine	C	4900	12	0.524	0.510	0.449	0.516	0.525	0.531	0.534	0.507	**0.536**
SpamBase	UCIrvine	C	4601	57	0.906	0.917	0.889	0.912	0.925	0.919	0.922	0.912	**0.932**
AP-omentum-ovary	OpenML	C	275	10936	0.832	0.814	0.658	0.830	0.832	0.758	0.849	0.833	**0.885**
Lymphography	UCIrvine	C	148	18	0.108	0.144	0.167	0.150	0.152	0.148	0.182	0.150	**0.267**
Ionosphere	UCIrvine	C	351	34	0.912	0.921	0.654	0.928	0.913	0.902	0.933	0.905	**0.985**
Housing Boston	UCIrvine	R	506	13	0.404	0.409	0.020	0.416	0.425	0.396	0.404	0.381	**0.467**
Airfoil	UCIrvine	R	1503	5	0.519	0.519	0.220	0.521	0.519	0.500	0.521	0.558	**0.629**
Openml_618	OpenML	R	1000	50	0.472	0.561	0.052	0.472	0.473	0.467	0.562	0.408	**0.692**
Openml_589	OpenML	R	1000	25	0.509	0.610	0.011	0.508	0.505	0.503	0.627	0.463	**0.656**
Openml_616	OpenML	R	500	50	0.070	0.193	0.024	0.149	0.167	0.156	0.372	0.076	**0.526**
Openml_607	OpenML	R	1000	50	0.521	0.555	0.107	0.516	0.519	0.522	0.621	0.476	**0.673**
Openml_620	OpenML	R	1000	25	0.511	0.546	0.029	0.527	0.513	0.512	0.619	0.442	**0.642**
Openml_637	OpenML	R	500	50	0.136	0.152	0.043	0.176	0.152	0.144	0.307	0.072	**0.465**
Openml_586	OpenML	R	1000	25	0.568	0.624	0.110	0.543	0.544	0.544	0.646	0.482	**0.700**

* We reported F1-Score for classification tasks, and 1-RAE for regression tasks.



(a) Spectf (b) SpamBase

(c) Openml_616 (d) Openml_618

Figure 4.4: The influence of data collection (GBFG^{-d}) and data augmentation (GBFG^{-a}) in GBFG.

Study of the impact of data collection and augmentation.

This experiment aims to answer: *Is it essential to collect feature transformation records and augment them to maintain GBFG performance?* To answer the question, we developed two model variants of GBFG: 1) GBFG^{-d}, which randomly collects feature transformation records for continuous space construction; 2) GBFG^{-a}, which removes the data augmentation step in GBFG. Figure 4.4 shows the comparison results in terms of Precision, Recall, F1-score, and Valid Rate for classification tasks (i.e., Spectf and SpamBase) and in terms of 1-MAE, 1-MSE, and 1-RAE, and Valid Rate for regression tasks (i.e., Openml_616 and Openml_618). Firstly, we found that

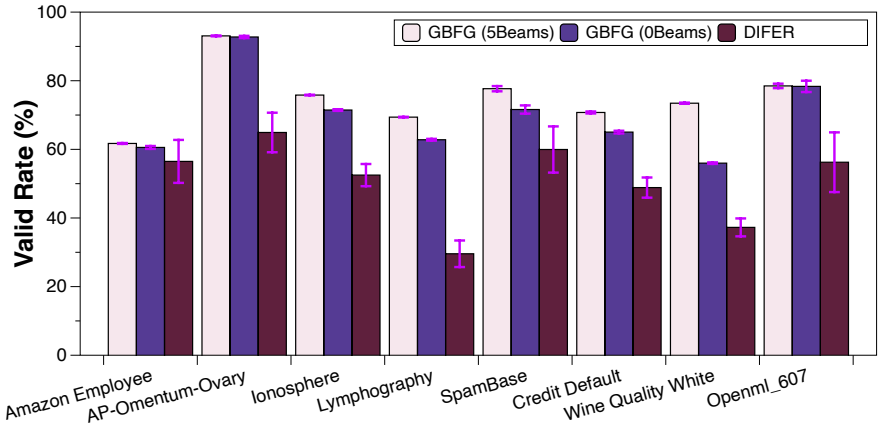


Figure 4.5: Comparison of the valid rate of GBFG in different beam search settings and DIFER.

the performance of GBFG is much better than GBFG^{-d}. The underlying driver is that the RL-based data collector can generate amounts of high-quality transformation records, which makes the continuous embedding space construction robust. Thus, it enables gradient-ascent search to identify the optimal feature space more effectively. Moreover, we observed that the performance of GBFG^{-a} is inferior to GBFG. This observation reflects that removing the data augmentation module significantly decreases data diversity and volume, leading to the learning process of the continuous embedding space construction being unstable and noisy. Thus, this experiment shows the importance of the data collection and augmentation components in the GBFG.

Study of the influence of beam search.

This experiment aims to answer: *Is it critical to introduce beam search into GBFG?* To establish the control group, we set the beam size as 5 and 0, respectively. In the meantime, we add DIFER as another comparison object. We compare the generation performance in terms of valid rate. Figure 4.5 shows the comparison result. We noticed that the result with the 5-beams search beats the 0-beams greedy search. The underlying driver is that introducing beam search may identify

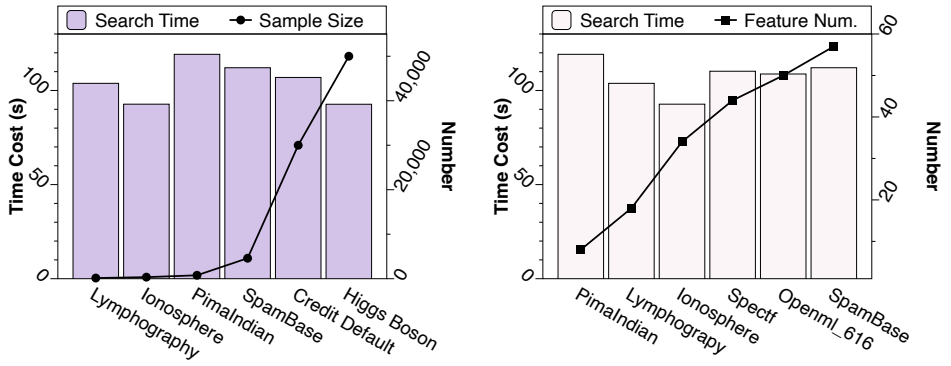
Table 4.2: Robustness check of GBFG with distinct ML models on Spectf dataset in terms of F1-score.

	RF	XGB	SVM	KNN	Ridge	LASSO	DT
RDG	0.760	0.818	0.750	0.792	0.718	0.749	0.864
ERG	0.757	0.813	0.753	0.766	0.778	0.750	0.790
LDA	0.665	0.715	0.760	0.749	0.759	0.760	0.665
AFAT	0.760	0.808	0.722	0.759	0.723	0.770	0.844
NFS	0.792	0.799	0.732	0.792	0.744	0.749	0.864
TTG	0.760	0.819	0.765	0.750	0.716	0.749	0.842
GRFG	0.818	0.842	0.580	0.760	0.729	0.744	0.786
DIFER	0.766	0.794	0.727	0.777	0.647	0.744	0.809
GBFG	**0.912**	**0.897**	**0.876**	**0.916**	**0.780**	**0.844**	**0.929**

a more effective and reasonable transformation sequence by enlarging the search space compared with greedy search (beam size=0). Another interesting observation is that DIFER is substantially worse than GBFG, and its error bar is extended. The underlying driver is that DIFER collects feature transformation records at random. Such a data collection method generates too many invalid sequences and error patterns, which distorts the constructed embedding space.

Robustness check of GBFG.

This experiment aims to answer: *Is GBFG robust with different downstream ML models?* We replaced the downstream ML models with Random Forest (RF), XGBoost (XGB), Support Vector Machine (SVM), K-Nearest Neighborhood (KNN), Ridge, LASSO, and Decision Tree (DT) to observe the robustness performance, respectively. Table 4.2 shows the comparison results on Spectf in terms of the F1-score. We observed that GBFG keeps the best performance regardless of downstream ML models compared with other baselines. A possible reason for this observation is that customized transformation records can be collected for different ML models. Then, the constructed embedding space may comprehend the preference and properties of distinct ML



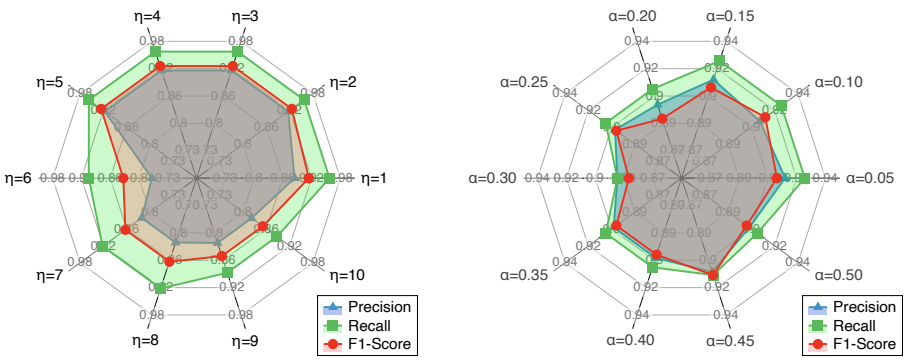
(a) Sample Size (b) Feature Number

Figure 4.6: Scalability check of GBFG in terms of search time for optimal feature space, sample size, and feature size.

models, thereby resulting in a globally optimal feature space. Therefore, this experiment shows the robustness and effectiveness of GBFG.

Study of the scalability of GBFG

This experiment aims to answer: *Can GBFG have good scalability to fit different datasets?* We visualized the changing trend of the time cost of searching for better feature spaces over sample size and feature dimensions of different datasets. Figure 5.3 shows the comparison results. We found that the time cost of GBFG keeps stable with the increase of sample size of the feature set. A possible reason is that GBFG only focuses on the decision-making benefits of feature ID and operation tokens instead of the information of the entire feature set, making the searching process sample size irrelevant. Another interesting observation is that the search time is still stable although the feature dimension of the feature set varies significantly. A possible explanation is that we map transformation records of varying lengths into a continuous space with a constant length. The searching time in this space is input dimensionality-agnostic. Thus, this experiment shows the GBFG has excellent scalability.

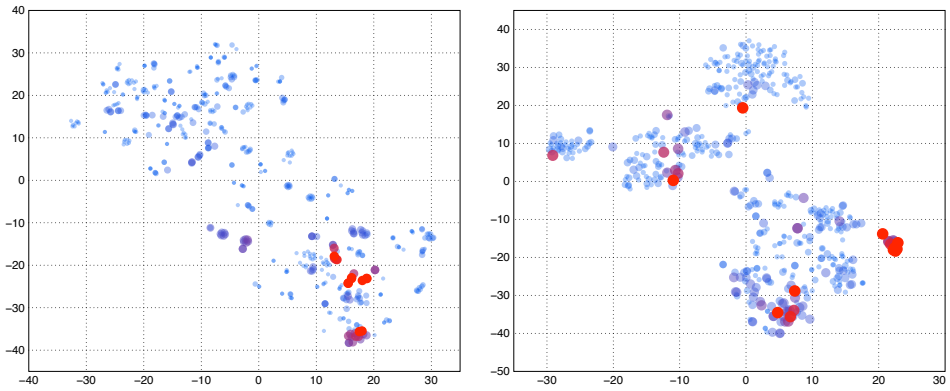


(a) Search Step Size η (under $\alpha = 0.05$) (b) Training Trade-off α (under $\eta = 1$)

Figure 4.7: Comparison of parameter sensitivity on search step size η and training loss trade-off α on Spectf dataset.

Study of the parameter sensitivity

To validate the parameter sensitivity of the search step size η (See section 4) and the trade-off parameter α in the training loss (See section 4), we set the value of η from 1 to 10, and set the value of α from 0.05 to 0.50 to observe the difference. Figure 4.7 shows the comparison results in terms of precision, recall, and F1-score. When the search step size grows, the downstream ML performance initially improves, then declines marginally. A possible reason for this observation is that a too-large step size may make the gradient-ascent search algorithm greatly vibrate in the continuous space, leading to missing the optimal embedding point and transformed feature space. Another interesting observation is that the standard deviation of the model performance is lower than 0.01 under different parameter settings. This observation indicates that GBFG is not sensitive to distinct parameter settings. Thus, the learning and searching process of GBFG is robust and stable.

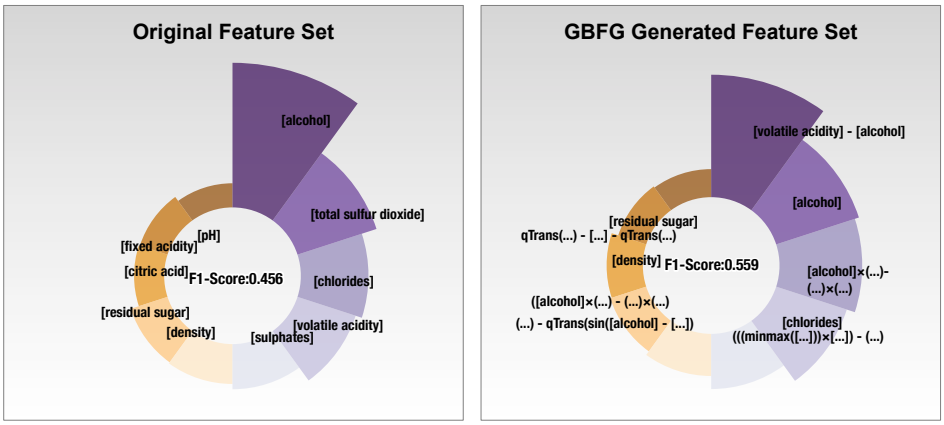


(a) Openml_616 (b) Airfoil

Figure 4.8: The visualization of learned transformation sequence embedding (from GBFG).

Study of the continuous embedding space.

This experiment aims to answer: *What is the continuous search space look like?* We selected Airfoil and Openml_616 as examples to visualize their continuous embedding space. In detail, we first collected the latent embeddings generated by the transformation records. Then, we use T-SNE to map them into a 2-dimensional space for visualization. Figure 4.8 shows the visualization results, in which each point represents a unique feature transformation sequence. The size of each point means its downstream performance. The bigger point size indicates that the downstream performance is superior. We colored the top 20 embedding points according to the performance in red. We found that the distribution locations of the top 20 embedding points are different. A potential reason is that the corresponding transformation sequences of the top 20 embedding points are different lengths. The sequence reconstruction loss distributes them to different areas of the embedding space. Moreover, we observed that the top 20 embedding points are close in the space even though the positions are different. The underlying driver is that the estimation loss makes these points with good performance clustered. Thus, this case study reflects that the reconstruction loss and estimation loss make the continuous space associate the transformation sequence and the corresponding model performance.



(a) Original Feature Space (b) GBFG Generated Feature Space

Figure 4.9: Comparison of traceability on the original feature space and the generated one produced by GBFG.

Study of the traceability and explainability of GBFG

This experiment aims to answer: *Is GBFG traceable, and what's new in the generated features?* We selected the top 10 essential features for prediction in the original, and GBFG transformed feature space of the Wine Quality Red dataset for comparison. Figure 4.9 shows the comparison results. The texts associated with each pie chart are the corresponding feature name. The larger the pie area is, the more critical the feature is. We found that almost 70% critical features in the new feature space are generated by GBFG and they improve the downstream ML performance by 22.6%. This observation indicates that GBFG really comprehends the properties of the feature set and ML models in order to produce a more effective feature space. Another interesting finding is that '[alcohol]' is the essential feature in the original feature set. But GBFG generates more mathematically composited features using '[alcohol]'. This observation reflects that GBFG not only can capture the significant features but also produce more effective knowledge for enhancing the model performance. Such composited features can make domain experts trace their ancestor resources and summarize new analysis rules for evaluating the quality of red wine.

Related Work

Automated Feature Transformation (AFT) can enhance the feature space by automatically mathematically transforming the original feature space [13, 51, 13]. Existing works can be divided into three categories: 1) expansion-reduction based approaches [40, 45, 52, 38, 43]. Those approaches first expand the original feature space by explicitly [42] or greedily [16] decided mathematical transformation, then reduce the space by selecting useful features. However, it is hard for them to produce or evaluate both complicated and effective mathematical compositions, leading to inferior transformation performance. 2) evolution-evaluation approaches [93, 44, 88, 112, 101]. These methods integrate feature generation and selection into a closed-loop learning system. They iteratively generate effective features and keep the most useful ones until achieving the maximum iteration number. The entire process is optimized by evolutionary algorithms or RL models. However, they still focus on how to simulate the discrete decision-making process in feature engineering. Thus, they are still time-consuming and unstable. 3) Auto ML-based approaches [12, 113]. Auto ML aims to find the most suitable model architecture automatically [18, 57, 35, 41]. The success of auto ML in many area [110, 100, 3, 92] and the similarity between auto ML and AFT inspire researchers to formulate AFT as an auto ML task to resolve. However, they are limited by: 1) incapable of producing high-order feature transformation; 2) unstable transformation performance. Thus, to fulfill these gaps, we propose a new framework GBFG. In this framework, we formulate AFT as a continuous optimization task to resolve. We develop an RL-based data collector to generate large quantities of high-quality transformation-accuracy records. We propose a postfix-based sequence expression way to model the entire transformation sequence, which saves computational resources, decreases the learning difficulties, and determines the transformation length automatically. We employ a beam search-based transformation sequence reconstruction module to generate more effective feature spaces for identifying the optimal one.

Conclusion Remarks

In this chapter, we propose a gradient-based automated feature transformation framework, namely GBFG. In detail, we first develop an RL-based data collector to gather large quantities of high-quality transformation-accuracy pairs. Then, we offer an efficient postfix notation-based sequence expression way to represent the transformation sequence in each pair. Moreover, we map them into a continuous embedding space using an encoder-decoder-evaluator model structure. In this space, each embedding point is associated with a transformation sequence and downstream ML performance. Finally, we employ a gradient-ascent search to identify better embeddings and then use beam search to reconstruct the transformation sequence and identify the optimal one. Extensive experiments show that the continuous optimization setting can efficiently search for the optimal transformed feature space. The RL-based data collector is essential to keep an excellent and stable transformation performance. The postfix expression sequence enables GBFG to automatically determine the transformation depth and length, resulting in more flexible and effective feature transformations. The beam search technique can increase the validity of feature transformation. In the future, we will focus on applying GBFG to more application domains to inspire and facilitate more areas.

CHAPTER 5: GENERATIVE-AI PERSPECTIVE: AUTOREGRESSIVE FEATURE SELECTION LEARNING

In this chapter, I will introduce the autoregressive feature selection learning framework. Within this unified optimization framework, the task of feature selection is well addressed.

Introduction

Feature selection aims to identify the most appropriate subset of features, which can be employed to optimize downstream predictive tasks. A proficiently executed feature selection can reduce dimensionality, shorten training time, bolster the generalization capability, mitigate the risk of overfitting, enhance predictive accuracy, and improve interpretation and explanation. Thus, feature selection can enrich our understanding of the underlying data patterns for conducting more comprehensive analyses.

Within the scope of practical application, feature selection encounters two prominent challenges: 1) The aspect of generalization, and 2) The issue of robustness. Firstly, different selection algorithms use different criteria to select representative features, making it challenging to find the best algorithm for cross-domain datasets. The notion of generalization within feature selection aims to address the following question: how can we facilitate consistently high accuracy across multiple domains? Secondly, certain domains (e.g., biomedical) involve a huge number of features, but sample sizes are limited due to costs, privacy, and ethnicity. When data are high-dimensional, point-point distances tend to be the same, and data patterns are non-discriminative. Concurrently, high dimensionality can amplify feature selection complexity and time costs in a discrete space. On the other hand, when data are low sample-sized, distributions are sparse, and data patterns

are unclear. Addressing robustness within feature selection intends to answer the following question: how can we automatically identify an effective yet small-sized feature subset with input dimensionality-agnostic time costs?

Relevant studies can only partially solve the two challenges. Classic feature selection algorithms can be grouped into three categories: (i) filter methods (e.g., univariate feature selection [107, 21], correlation-based feature selection [31, 108]), in which features are ranked by a specific score. However, feature relevance or redundancy scores are usually domain-specific, non-learnable, and cannot generalize well to all applications. (ii) wrapper methods (e.g., evolutionary algorithms [105, 46], branch and bound algorithms [67, 48]), in which optimal feature subset is identified by a search strategy that collaborates with predictive tasks. However, such methods have to search a large feature space of 2^N feature subspace candidates, where N is the number of input features. (iii) embedded methods (e.g., LASSO [86], decision tree [82]), in which feature selection is part of the optimization objective of predictive tasks. However, such methods are subject to the strong structured assumptions of predictive models (e.g., the L1 norm assumption of LASSO). That is, feature selection and predictive models are coupled together, lack flexibility, and are hard to generalize to other predictors. Therefore, we need a novel perspective to derive the novel formulation and solver of generalized and disruption-robust feature selection.

Our contributions: a discrete subsetting as continuous optimization perspective. We formulate the problem of discrete feature selection as a gradient-based continuous optimization task. We propose a new perspective: the discrete decisions (e.g., select or deselect) of feature selection can be embedded into a continuous embedding space, thereafter, solved by a more effective gradient-based solution. We show that this perspective can be implemented by feature subset encoding, gradient-optimized search, and reconstruction. Training deep models requires big training data, and we find that reinforcement feature selection can be used as a tool to automatically generate features-accuracy pairs in order to increase the automation, diversity, and scale of training data

preparation. We demonstrate that integrating both reinforcement-generated and classic selection algorithms-generated experiences learn a better feature subset embedding space because: 1) using reinforcement is to crowd-source unknown exploratory knowledge and 2) using classic selection algorithms is to exploit existing peer knowledge. We highlight that a globally-described discriminative embedding space, along with the joint objective of minimizing feature subset reconstruction loss and accuracy estimation loss, can strengthen the denoising ability of gradient search, eliminate noisy and redundant features, and yield an effective feature subset. We observe that, by representing feature subsets into fixed-sized embedding vectors, the time costs of gradient-based optimization are input dimensionality-agnostic (only relates to embedding dimensionality). The feature subset decoder can automatically reconstruct optimal selected features and doesn't need to manually identify the number of best features.

Summary of Proposed Approach. Inspired by these findings, this paper presents a generic and principled framework for deep differentiable feature selection. It has two goals: 1) generalized across domains; 2) disruption-robust: overcome training data bottlenecks, reduce feature subset size while maintaining accuracy, and control time costs against input dimensionality. To achieve Goal 1, we achieve feature selection via a pipeline of representation-search-reconstruction. Specifically, given a downstream predictor (e.g., decision tree), we collect the features-accuracy pairs from diverse feature selection algorithms so that training data represent the diversity of samples. An embedding space is learned to map feature subsets into vectors. We advance the representability and generalization of the embedding space by exploiting the diverse and representative train samples to optimize the joint loss of feature subset reconstruction and accuracy evaluation. We leverage the accuracy evaluator as feedback to infer gradient direction and degree to improve the search for optimal feature subset embeddings. To achieve Goal 2, we devise different computing strategies: 1) We develop a multi-agent reinforcement feature selection system that can automatically generate large-scale high-quality features-accuracy pairs in a self-optimizing fashion and

overcome training data bottlenecks. 2) We reformulate a feature subset as an alphabetical non-ordinal sequence. We devise the strategy of jointly minimizing not just accuracy evaluation loss but also sequence reconstruction loss in order to position the gradient toward a more denoising direction over the continuous embedding space to reconstruct a shorter feature subset. We find that the reduction of feature space size improves generalization. 3) Since we embed feature subsets into a fixed-size embedding space, gradient search in a fixed-size space is input dimensionality agnostic. Finally, we present extensive experimental results to show the small-sized, accurate, input-dimensionality agnostic, and generalized properties of our selected features.

Problem Statement

Our goal is to develop a generalized and robust deep differentiable FS framework. Formally, given a dataset $D = \{X, y\}$, where X is a feature set, and y is predictive target labels. We utilize classic FS algorithms to D to collect n feature subset-accuracy pairs as training data, denoted by $R = \{(\mathbf{f}_i, v_i)\}_{i=1}^{n}$, where $\mathbf{f}_i = [f_1, f_2, \cdots, f_T]$ is the feature ID token set of the i-th feature subset, and v_i is corresponding downstream predictive accuracy. Thereafter, we pursue two aims: 1) constructing an optimal continuous space of feature subsets. We learn a mapping function ϕ, a reconstructing function ψ, and an evaluation function ω via joint optimization to convert R into a continuous embedding space $\mathcal{E}$, in which each embedding vector represents the feature ID token set of a feature subset and corresponding model performance. 2) searching the optimal feature subset. We adopt gradient-based search to find the optimal feature token set $\mathbf{f}^*$, given by:

$$\mathbf{f}^* = \psi(\mathbf{E}^*) = \text{argmax}_{\mathbf{E} \in \mathcal{E}} \mathcal{A}(\mathcal{M}(X[\psi(\mathbf{E})]), y), \tag{5.1}$$

where ψ is a reconstruction function to reconstruct a feature ID token set from any embedding of $\mathcal{E}$; $\mathbf{E}$ is an embedding vector in $\mathcal{E}$ and $\mathbf{E}^*$ is the optimal one; $\mathcal{M}$ is the downstream ML model

and $\mathcal{A}$ is the performance indicator. We apply $\mathbf{f}^*$ to X to select the optimal feature subset $X[\mathbf{f}^*]$ to maximize downstream ML model performances.

Methodology

In this section, we present an overview of our method, and then introduce the technical details of each component.

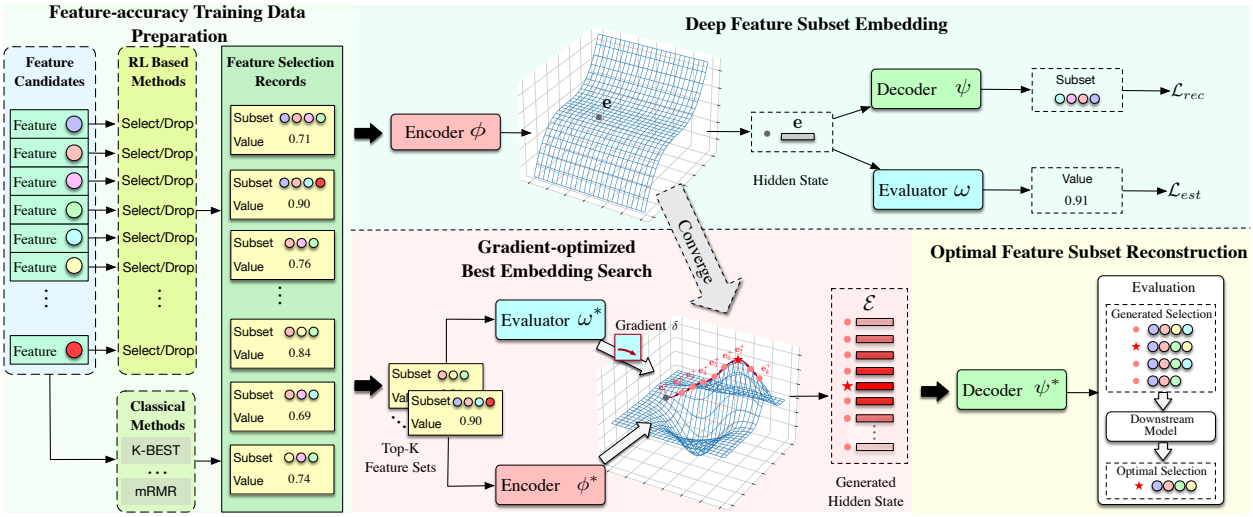


Figure 5.1: An overview of our framework. GAINS is made up of four main components: 1) feature-accuracy training data preparation, designed to quickly collect large quantities of qualified and valid training data; 2) deep feature subset embedding, purposed to preserve the knowledge of feature selection into a global continuous embedding space; 3) gradient-optimized best embedding search, intended to search for better embeddings in the learned space; 4) optimal feature subset reconstruction, devised to reconstruct and output the optimal feature subset.

Framework Overview

Figure 5.1 shows the overview of our framework, including four steps: 1) feature-accuracy training data preparation; 2) deep feature subset embedding; 3) gradient-optimized best embedding search;

4) optimal feature subset reconstruction. Step 1 is to collect selected feature subsets and corresponding predictive performances as training data for deep feature subset embedding. In particular, to exploit existing peer knowledge, we utilize classical feature selection methods (e.g., K-Best, mRMR) to collect feature subset-accuracy records; to explore crowdsource unknown knowledge, we utilize reinforcement learning (RL) based feature selector to collect diverse feature subset-accuracy records. Step 2 is to develop a deep encoder-evaluator-decoder model to learn the optimal embedding space of feature subsets. In particular, given a feature subset, the encoder maps the feature subset ID tokens into a continuous embedding vector; the evaluator optimizes the embedding vector along the gradient direction by predicting the corresponding model performance; the decoder reconstructs the feature ID tokens using feature subset embedding vectors. We learn the optimal embedding space by jointly optimizing the reconstruction and evaluation losses of the encoder, evaluator, and decoder. Step 3 aims to expedite the gradient-optimized search of optimal feature subset embedding vector in the embedding space. In particular, we first select top-K historical feature subset-accuracy pairs as search seeds (starting points) and obtain corresponding embeddings using the well-trained encoder. We then search by starting from these embeddings at a minute rate in the gradient direction of performance improvement to identify optimal embeddings. Step 4 is to exploit the well-trained decoder to reconstruct the optimal feature ID tokens as candidate feature subsets from these identified embeddings. We evaluate these candidate feature subsets with a downstream ML model to present the best feature subset with the highest performance.

Feature-Accuracy Training Data Preparation

We collect a set of feature-accuracy pairs as training data to construct an effective continuous space that depicts the properties of the original feature set.

Leveraging exploitation and exploration for automated training data preparation. The di-

versity, scale, and comprehensiveness of the training data population are essential for learning an effective embedding space. The training data is collected via two strategies. 1) *exploitation of existing feature selection algorithms*: we apply classical feature selectors (e.g., KBest, Lasso.) to the given feature set. Each algorithm represents one algorithmic perspective and produces a small number of feature subset-accuracy records. It is challenging to collect large-scale, diverse training data. 2) *exploration of other unknown candidate subsets*: We find that reinforcement learning can be used to automatically explore and select various feature subsets to evaluate corresponding feature subset performance [61]. In other words, reinforcement feature selection can be viewed as a tool for automated training data preparation. In particular, we develop a multi-agent reinforcement feature selection system, where each agent is to select or deselect a feature in order to progressively explore feature subsets and find high-accuracy low-redundancy feature subsets. The reinforcement feature selection system leverage randomness and self-optimization to automatically generate high-quality, diverse, comprehensive feature subset-accuracy records to overcome data sparsity.

Leveraging order agnostic property to augment training data. To learn a feature subset embedding space, we view a feature subset as a token sequence and exploit seq2vec models to embed a feature subset as a vector. Since the feature subset is order-agnostic, the token sequence is order-agnostic. We leverage the order-agnostic property to augment the feature subset-accuracy training data by shuffling and rearranging the orders of selected features in a feature token sequence. Formally, given a selected feature subset and corresponding predictive accuracy, denoted by $\{\mathbf{f}, v\}$, where $\mathbf{f} = [f_1, f_2, \cdots, f_T]$ is the ID tokens of selected features and v_i is corresponding performance. We shuffle the old token order and obtain a new order of the T-length feature subset $\tilde{\mathbf{f}} = [f_3, f_T, \cdots, f_1]$. We add the new shuffled token sequence and corresponding accuracy into the training data to enhance data diversity and comprehensiveness.

Why effective embedding space construction matters. Most conventional feature selection algorithms are of a discrete choice formulation. Such formulation results in suboptimal performance since it is difficult to enumerate all possible feature combinations in a high-dimensional dataset. To efficiently search the optimal feature subset, it is crucial to change feature selection from searching in discrete space to searching in continuous space, where gradient-based optimization can be used for faster and more effective selection.

But, How can the knowledge derived from feature selection be incorporated into an embedding space? There are two potential methods: 1) Sequential modeling, which learns distinct embeddings for the same feature subset irrespective of order; 2) Set modeling, which generates a consistent embedding for the same feature subset, even with altered order. In sequential modeling, the identical subset with varying orders yields different embeddings. These form a global continuous embedding space, where lighter regions signify poor model performance, while darker regions correspond to superior performance. It is easy to search for the optimal solution by shifting candidate embeddings to the optimal regions via gradients. Conversely, set modeling, which learns a consistent embedding for variations of the same feature subset, results in a locally optimal embedding space. This model cannot perceive the order of feature subsets that do not influence performance. Consequently, even after embedding updates, locating the optimal point remains challenging due to the lack of clear direction. Based on these observations, we opt for sequential modeling (i.e., LSTM) to integrate feature selection knowledge into a continuous embedding space.

Leveraging the encoder-evaluator-decoder framework to embed feature subset tokens. We aim to construct an effective continuous embedding space in order to search for better feature subsets using gradient-based search in the gradient direction of higher performance. We develop a novel learning framework that includes the encoder, evaluator, and decoder. Next, we use $\mathbf{f}$

to denote the feature ID tokens of selected features and v to denote the corresponding model performance.

The feature subset encoder ϕ: The encoder is to learn a mapping function ϕ that takes $\mathbf{f}$ as input, and outputs its continuous embedding $\mathbf{E}$, denoted by $\phi(\mathbf{f}) = \mathbf{E}$. The encoder is designed based on a single layer of Long Sort-Term Memory [37] (LSTM), where $\mathbf{f} \in \mathbb{R}^{1\times T}$ and T is the length of the feature subset. After inputting $\mathbf{f}$ into ϕ, we collect each feature ID token embedding to form $\mathbf{E}$. Here, $\mathbf{E} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_T] \in \mathbb{R}^{T\times d}$, where $\mathbf{h}_t \in \mathbb{R}^{1\times d}$ is the t-th token embedding with dimension d.

The feature subset decoder ψ: The decoder ψ aims to reconstruct the feature ID tokens $\mathbf{f}$ of of an embedding vector $\mathbf{E}$, denoted by $\psi(\mathbf{E}) = \mathbf{f}$. Similar to the encoder, we employ a single-layer LSTM to implement the decoder. We train the decoder in an autoregressive manner. The initial input of the decoder is the last embedding of $\mathbf{E}$, denoted by $\hat{\mathbf{h}}_0 = \mathbf{h}_T$. We take the i-th step in LSTM as an example to demonstrate the calculation process. Specifically, we input $\hat{\mathbf{h}}_{i-1}$ and the i-th feature ID token in $\mathbf{s}$ into the LSTM layer to get the current embedding $\hat{\mathbf{h}}_i$. The output of the decoder and the input of the encoder share the same feature ID token dictionary. Thus, to forecast the next most possible token easily, we utilize the attention mechanism [2] to learn the attention weights between $\hat{\mathbf{h}}_i$ and each token embedding in $\mathbf{E}$. Then, we generate the enhanced embedding $\tilde{\mathbf{h}}_i$ by aggregating the knowledge of $\mathbf{E}$ using the attention weights. This process can be defined by:

$$\tilde{\mathbf{h}}_i = \sum_{\mathbf{h}_j \in \mathbf{E}} a_{ij}\mathbf{h}_j, \text{ where } a_{ij} = \frac{\exp(\hat{\mathbf{h}}_i \cdot \mathbf{h}_j)}{\sum_{\mathbf{h}_k \in \mathbf{E}} \exp(\hat{\mathbf{h}}_i \cdot \mathbf{h}_k)}. \tag{5.2}$$

where $\mathbf{h}_j$ is the j-th embedding from $\mathbf{E}$ and a_{ij} is the attention weight between $\hat{\mathbf{h}}_i$ to $\mathbf{h}_j$. Later, we concatenate $\hat{\mathbf{h}}_i$ and $\tilde{\mathbf{h}}_i$ together and input them into a fully-connected layer activated by Softmax

to estimate the distribution of each feature ID token for the i-th step, which can be formulated by:

$$P_\psi(f_i|\mathbf{E}, \mathbf{f}_{<i}) = \frac{\exp(\mathbf{W}_{\mathbf{h}_i} \text{ concat}(\tilde{\mathbf{h}}_i, \hat{\mathbf{h}}_i))}{\sum_{\mathbf{h}_k \in \mathbf{E}} \exp(\mathbf{W}_{\mathbf{h}_k} \text{ concat}(\tilde{\mathbf{h}}_i, \hat{\mathbf{h}}_i))}, \tag{5.3}$$

where concat$(\cdot)$ is the concatenate operation, $\mathbf{W}_{(\cdot)}$ represents the weight matrix. $\mathbf{f}_{<i}$ represent the previous tokens before the i-th step. We may take the token with the highest probability as the predicted feature ID token, denoted by f_i. If multiple the probability at each step, the probability distribution of the whole token sequence $\mathbf{f}$ can be represented by:

$$P_\psi(\mathbf{f}|\mathbf{E}) = \prod_{t=1}^{T} P_\psi(f_t|\mathbf{E}, \mathbf{f}_{<t}). \tag{5.4}$$

The Feature Subset Evaluator ω: The evaluator ω aims to predict the model performance given the continuous embedding of $\mathbf{E}$. Specifically, we apply the mean pooling to the embedding $\mathbf{E}$ in a column-wise manner to obtain the embedding $\bar{\mathbf{e}} \in \mathbb{R}^{1\times d}$. We then input it into a fully-connected layer to predict the model performance $\ddot{v}$, which can be defined by: $\ddot{v} = \omega(\bar{\mathbf{e}})$.

The joint optimization. We jointly optimize the encoder, decoder, and evaluator. There are two goals: 1) reconstructing the feature ID tokens $\mathbf{f}$. To achieve this goal, we minimize the negative log-likelihood of $\mathbf{f}$ given $\mathbf{E}$, defined by:

$$\mathcal{L}_{rec} = -\log P_\psi(\mathbf{f}|\mathbf{E}) = -\sum_{t=1}^{T} \log P_\psi(f_t|\mathbf{E}, \mathbf{f}_{<t}). \tag{5.5}$$

2) minimizing the difference between the predicted performance $\ddot{v}$ and the real one v. To achieve this goal, we minimize the Mean Squared Error (MSE) between $\ddot{v}$ and v, defined by:

$$\mathcal{L}_{est} = \text{MSE}(v, \ddot{v}). \tag{5.6}$$

We integrate the two losses to form a joint training loss $\mathcal{L}$, defined by:

$$\mathcal{L} = \lambda \mathcal{L}_{rec} + (1 - \lambda)\mathcal{L}_{est}, \tag{5.7}$$

where λ is the trade-off hyperparameter to balance the contribution of the two objectives during the learning process.

Gradient-Optimized Best Embedding Search

Why selecting search seeds matters. When the encoder-evaluator-decoder model converges, we perform a gradient-based search in the well-built continuous space for better feature selection results. Similar to the significance of initialization for deep neural networks, good starting points can accelerate the search process and enhance its performance. These points are called "search seeds". Thus, we first rank the selection records based on the corresponding model performance. Then, the top-K selection records are chosen as search seeds for searching for better embeddings.

Gradient-ascent Optimizer Search. Assuming that one of the top-K selection records is $(\mathbf{f}, v)$. We input $\mathbf{f}$ into the encoder to obtain the embedding $\mathbf{E} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_T]$. Then, we move each embedding in $\mathbf{E}$ along the gradient direction induced by the evaluator ω:

$$\mathbf{h}_t^+ = \mathbf{h}_t + \eta \frac{\partial \omega}{\partial \mathbf{h}_t}, \mathbf{E}^+ = \{\mathbf{h}_1^+, \mathbf{h}_2^+, \cdots, \mathbf{h}_T^+, \}, \tag{5.8}$$

where η is the step size and $\mathbf{E}^+$ is the enhanced embedding. We should set η within a reasonable range (e.g., small enough) to make the model performance of the enhanced embedding $\mathbf{E}^+$ is better than $\mathbf{E}$, denoted by $\omega(\mathbf{E}^+) \geq \omega(\mathbf{E})$. We conduct the search process for each record and collect these enhanced embeddings, denoted by $[\mathbf{E}_1^+, \mathbf{E}_2^+, \cdots, \mathbf{E}_K^+]$. Since we have embedded discrete selection records into d dimensional embedding space, the time complexity of the searching process

is independent of the size of the original feature set.

Optimal Feature Subset Reconstruction

The enhanced embeddings indicate possible better feature selection results. To find the optimal one, we should reconstruct the feature ID tokens using them. Specifically, we input $[\mathbf{E}_1^+, \mathbf{E}_2^+, \cdots, \mathbf{E}_K^+]$ into the well-trained decoder to get the reconstructed feature ID tokens $[\mathbf{f}_1^+, \mathbf{f}_2^+, \cdots, \mathbf{f}_K^+]$. We do not need to set the number of feature ID tokens throughout the decoding process; instead, we identify each token until the stop token, similar to how natural language generation works. Then, we use them to select different feature subsets from the original feature set. Finally, we adopt the downstream ML model to evaluate these subsets and output the optimal feature ID tokens $\mathbf{f}^*$, which can be used to select the best feature subset with the highest model performance.

Experiments

In this section, we present detailed experimental setups and conduct comprehensive experimental analyses and case studies to validate the efficacy of the proposed model.

Experimental Setup

Dataset Description.

We conducted extensive experiments using 19 publicly available datasets from UCI, OpenML, CAVE, Kaggle, and LibSVM. These datasets are categorized into 3 folds based on the types of ML tasks: 1) binary classification (C); 2) multi-class classification (MC); 3) regression (R). Table 5.1

shows the statistics of these datasets, which is accessible via the website address provided in the Abstract.

Evaluation Metrics.

For the binary classification task, we adopted F1-score, Precision, Recall, and ROC/AUC. For the multi-classification task, we used Micro-F1, Precision, Recall, and Macro-F1. For the regression task, we utilized 1-Mean Average Error (1-MAE), 1-Mean Square Error (1-MSE), and 1-Root Mean Square Error (1-RMSE). For all metrics, the higher the value is, the better the model performance is.

Baseline Algorithms.

We compared GAINS with 10 widely used feature selection methods: (1) **K-Best** selects K features with the highest feature scores [107]; (2) **mRMR** intends to select a feature subset with the greatest relevance to the target and the least redundancy among themselves [70]; (3) **LASSO** selects features by regularizing model parameters, which shrinks the coefficients of useless features into zero [86]; (4) **RFE** recursively removes the weakest features until the specified number of features is reached [27]; (5) **LASSONet** designs a novel objective function to conduct feature selection in neural networks [55]; (6) **GFS** selects features using genetic algorithms, which recursively generates a population based on a possible feature subset, then uses a predictive model to evaluate it [54]; (7) **MARLFS** builds a multi-agent system for selecting features, wherein each agent is associated with a single feature, and feature redundancy and downstream task performance are viewed as incentives [61]; (8) **SARLFS** is a simplified version of MARLFS, which uses one agent to determine the selection of all features in order to alleviate computational costs [64]; (9) **RRA** first collects distinct selected feature subsets, and then integrates them based on statistical sorting

distributions [80]; (10) **MCDM** first obtains a decision matrix using the ranks of features, then assigns feature scores based on the matrix for ensemble feature selection [33]. Among the discussed baseline models, K-Best and mRMR fall under the category of filter methods. Lasso, RFE, and LassoNet are considered as embedded methods. GFS, SARLFS, and MARLFS are classified as wrapper methods. Lastly, RRA and MCDM are representative of hybrid feature selection methods.

We randomly split each dataset into two independent sets. The prior 80% is used to build the embedding space, and the remaining 20% is used to search for the optimal feature space. We conducted all experiments using the hold-out setting to ensure a fair comparison. We adopted Random Forest as the downstream machine learning model and reported the performance of each method by running five-fold cross-validation on the testing set. Random Forest is a robust, stable, well-tested method, thus, we can reduce performance variation caused by the model, and make it easy to study the impact of the result of feature selection.

Hyperparameter Settings and Reproducibility.

We ran MARLFS for 300 epochs to collect historical feature subsets and the corresponding downstream task performance. To augment data, we permutated the index of each feature subset 25 times. These augmented data can be used for GAINS training. The Encoder and Generator have the same model structure, which is a single-layer LSTM. The Predictor is made up of 2-layer feed-forward networks. The hidden state sizes of the Encoder, Decoder, and Predictor are 64, 64, and 200, respectively. The embedding size of each feature index is 32. To train GAINS , we set the batch size as 1024, the learning rate as 0.001, and λ as 0.8 respectively. During the model inference stage, we used the top 25 feature selection records as the initial points to search for the optimal feature space.

Environmental Settings

All experiments were conducted on the Ubuntu 18.04.6 LTS operating system, AMD EPYC 7742 CPU, and 8 NVIDIA A100 GPUs, with the framework of Python 3.9.10 and PyTorch 1.8.1 [68].

Performance Evaluation

Table 5.1: Overall performance comparison. 'C' for binary classification, 'MC' for multi-class classification, and 'R' for regression. The best results are highlighted in **bold**. The second-best results are highlighted in underline. (**Higher values indicate better performance.**)

Dataset	Task	#Samples	#Features	Original	K-Best	mRMR	LASSO	RFE	LASSONet	GFS	SARLFS	MARLFS	RRA	MCDM	GAINS
SpectF	C	267	44	75.96	78.21	79.16	75.96	80.80	75.96	75.01	75.96	75.96	79.16	80.36	**87.38**
SVMGuide3	C	1243	21	77.81	75.13	75.34	75.95	78.07	76.44	83.12	79.48	81.32	77.63	76.66	**83.68**
German Credit	C	1001	24	64.88	66.67	65.81	69.65	64.86	58.20	67.54	63.32	69.00	69.69	70.85	**75.34**
Credit Default	C	30000	25	80.19	80.59	80.59	77.94	80.28	80.05	79.96	80.05	80.24	75.39	74.46	**80.61**
SpamBase	C	4601	57	92.68	92.02	91.91	91.74	91.68	88.39	92.25	90.94	92.35	89.43	88.95	**92.93**
Megawatt1	C	253	38	81.60	78.55	80.08	83.78	80.08	81.03	77.42	82.75	79.33	87.78	85.11	**90.42**
Ionosphere	C	351	34	92.85	91.38	95.69	86.98	95.69	85.58	91.34	94.27	88.48	92.74	88.64	**97.10**
Activity	MC	10299	561	96.17	96.07	95.92	95.92	95.87	96.17	96.12	95.87	95.87	95.58	96.12	**96.46**
Mice-Protein	MC	1080	77	74.99	78.69	76.84	78.71	77.29	78.23	77.35	74.53	77.30	70.86	78.69	**79.16**
Coil-20	MC	1440	400	96.53	95.84	95.49	95.84	96.52	82.97	95.49	94.79	95.47	95.15	95.50	**97.22**
MNIST	MC	10000	784	92.70	92.75	92.35	92.20	92.25	87.05	91.60	91.70	91.75	90.40	92.65	**93.20**
MNIST fashion	MC	10000	784	80.15	80.80	80.00	79.90	80.50	78.85	80.60	80.10	80.15	78.85	80.20	**81.00**
Openml_586	R	1000	25	54.95	57.68	57.29	60.67	58.10	58.60	63.27	55.76	57.21	57.80	57.95	**64.00**
Openml_589	R	1000	25	50.95	54.09	54.03	59.74	54.25	54.80	44.72	38.64	52.77	54.54	55.43	**59.76**
Openml_607	R	1000	50	51.74	53.03	53.03	58.10	54.39	53.63	45.70	55.32	53.94	55.41	55.56	**66.01**
Openml_616	R	500	50	15.63	24.75	24.44	28.98	24.08	16.32	52.93	25.37	25.83	23.32	22.92	**47.39**
Openml_618	R	1000	50	46.89	51.33	51.33	47.41	50.64	50.69	52.40	50.18	51.71	50.93	50.90	**59.13**
Openml_620	R	1000	25	51.01	53.57	53.57	57.99	53.96	54.29	61.99	34.45	53.68	56.24	55.66	**62.58**
Openml_637	R	500	50	14.95	20.72	20.72	26.02	17.82	18.97	40.12	19.54	23.72	22.38	22.16	**42.12**

* We reported F1-Score for classification tasks, Micro-F1 for multi-class classification tasks, and 1-RAE for regression tasks.

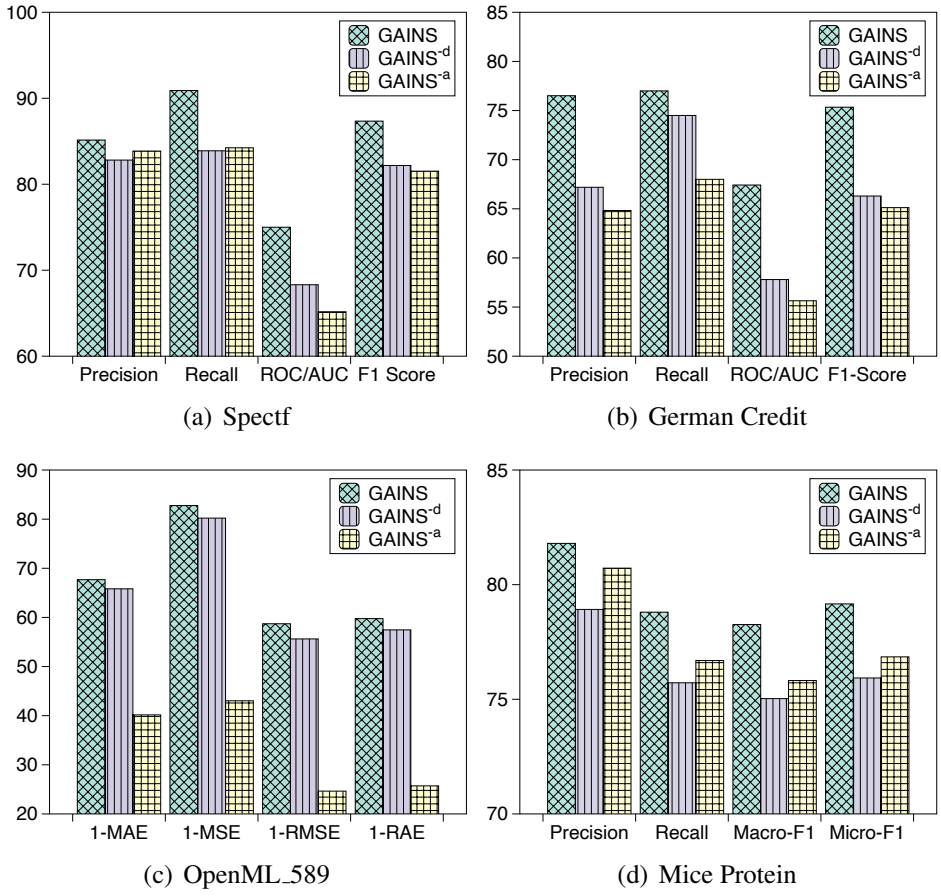
Overall Comparison.

This experiment aims to answer: *Can GAINS effectively select the feature subset with excellent performance?* Table 5.1 shows the overall comparison results in terms of F1-score, Micro-F1, and

1-RAE. We observed that GAINS outperforms other baseline models across all domains and various tasks. The underlying driver for this observation is that GAINS converts the discrete feature selection records into a discriminative and effective embedding space by integrating the knowledge of historical selection records. It enables the gradient-based search to effectively perceive the properties of the original feature set to obtain superior feature subsets. Another interesting observation is that the performances of various baseline models vary over datasets: high in certain datasets, yet low in other datasets. Such an observation indicates that classic feature selection methods can address the feature selection task in a limited number of data environments, but perform unstably in diverse data environments. Overall, this experiment demonstrates that GAINS has effective and robust performance in various data environments and application scenarios.

Examining the impact of data collection and augmentation.

This experiment aims to answer: *Is it essential to collect feature selection records and augment them to maintain GAINS performance?* To establish the control group, we developed two model variants: 1) GAINS^{-d}, we collected feature selection records at random rather than building basic feature selectors. 2) GAINS^{-a}, we removed the data augmentation process of GAINS. Figure 5.2 shows the comparison results on Spectf, German Credit, OpenML_589, and Mice Protein. We found that the performance of GAINS is much better than GAINS^{-d}. The underlying driver is that, compared to random generation, selection records generated by feature selectors are more robust and denoising, which is important for creating a more effective embedding space for searching better feature subsets. Moreover, we observed that GAINS is superior to GAINS^{-a} in all cases. The underlying driver is that data augmentation can increase the data diversity, resulting in a more robust and effective learning process for GAINS. Thus, this experiment validates that data collection and augmentation is significant for maintaining GAINS performance.



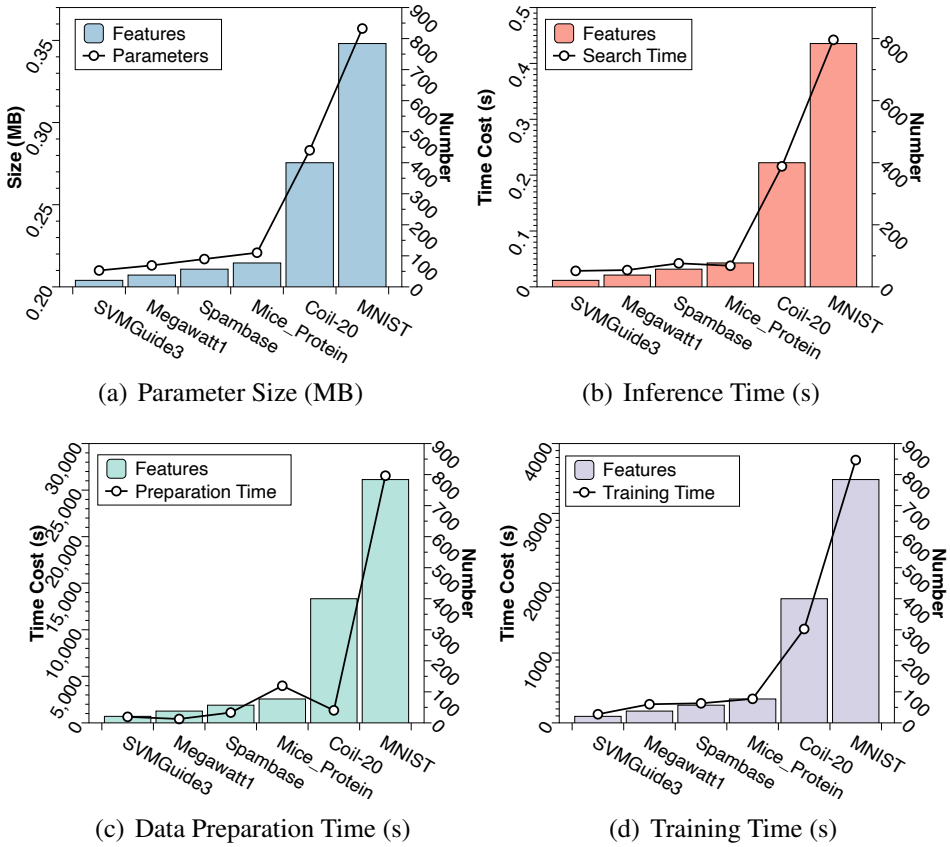
(a) Spectf (b) German Credit

(c) OpenML_589 (d) Mice Protein

Figure 5.2: The influence of data collection (GAINS^{-d}) and data augmentation (GAINS^{-a}) in GAINS.

Study of the scalability of GAINS.

This experiment aims to answer: *Does GAINS have excellent scalability in different datasets?* According to the dimensionality of the feature set, we chose 6 datasets: SVMGuide3, Megawatt1, Spambase, Mice Protein, Coil-20, and MNIST, from small to large. We analyzed the variation in the average time required to make an inference during a single search step and in the parameter size employed by the encoder-evaluator-decoder model. Figure 5.3 (a-b) shows the comparison results in terms of Parameter Size and Inference Time. We found that in spite of the almost 40-



(a) Parameter Size (MB)

(b) Inference Time (s)

(c) Data Preparation Time (s)

(d) Training Time (s)

Figure 5.3: Scalability check of GAINS in terms of feature size, parameter size, inference time, data preparation time, and training time.

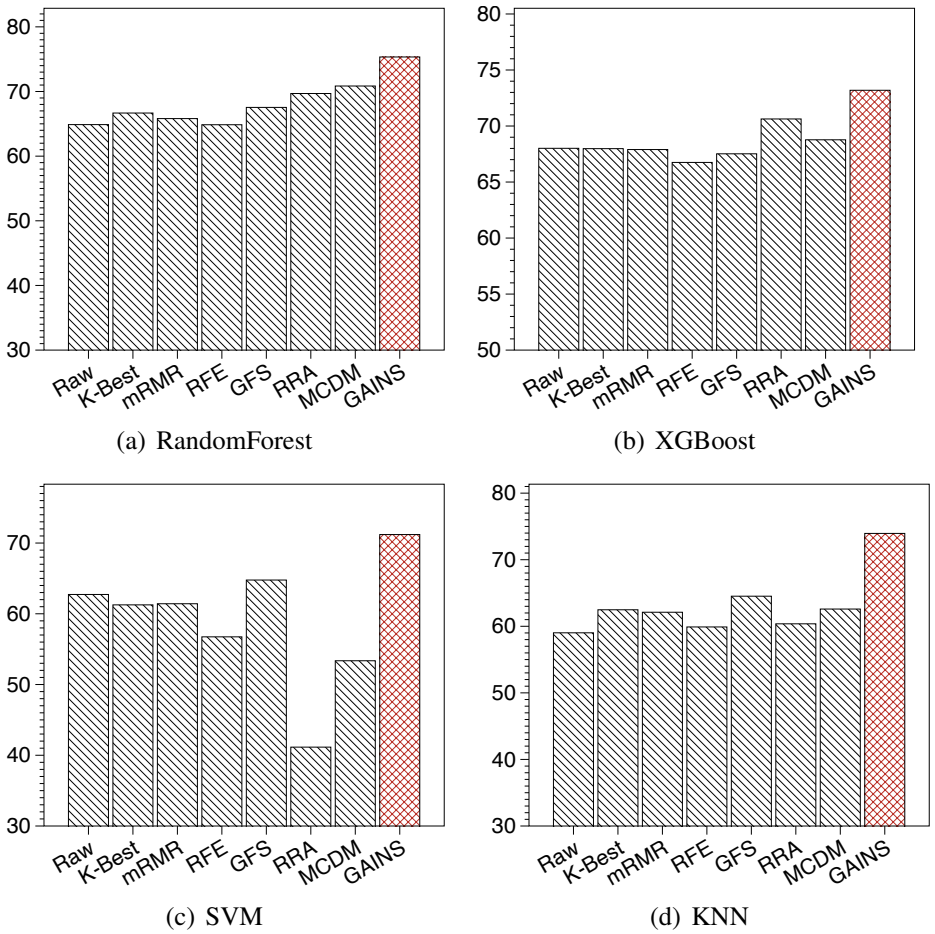
fold increase in feature dimension from SVMGuide3 to MNIST, the inference time increases by just about 4-fold, and the parameter size only enlarges by almost 2-fold. The underlying driver for this observation is that the deep feature subset embedding module has integrated the knowledge of discrete selection records into a fixed continuous embedding space, significantly reducing the searching time and embedding model size. The time-cost bottleneck is only caused by the downstream ML model validation. Therefore, this experiment validates that GAINS has strong scalability when dealing with datasets of varying dimensions.

Study of the time cost of data preparation and model training.

This experiment aims to address the following question: *What are the computational costs in terms of data preparation and model training?* The same as the scalability examination, we selected six datasets with varying dimensions, i.e., SVMGuide3, Megawatt1, Spambase, Mice Protein, Coil-20, and MNIST. Figure 5.3 (c-d) illustrates the comparison of the corresponding time costs. From Figure 5.3 (c) We found that the time cost for data preparation correlates with the feature dimension. For instance, the MNIST dataset, encompassing 784 feature columns, necessitates approximately 26,547.5 seconds for collecting sufficient samples. In contrast, the Coil-20 dataset, comprising 400 feature columns, demands a lesser duration of 1344.56 seconds. Interestingly, the Mice_Protein dataset, despite only encompassing 77 feature columns, necessitates 3985 seconds. This observation indicates that the time cost of the reinforcement learning-based data preparation module is associated with the number of feature columns, the sample size, and the complexity of the downstream task. Figure 5.3 (d) shows the model training duration increases proportionately with the number of feature columns, thereby validating the findings of the scalability examination. This illustrates that GAINS requires only a finite number of epochs for reinforced data collection, thereby significantly reducing the time cost for exploration relative to RL-based methods such as SARLFS and MARLFS. Secondly, the training time remains unaffected by the sample size or the complexity of the downstream task, indicating that our continuous optimization method can make the feature selection process more efficient.

Study of the impact of hold-out percentage.

This experiment aims to answer: *How does the variation in hold-out settings impact our proposed model?* In the experiment setup section, we partitioned the dataset randomly into two independent subsets. The larger subset, comprising 80% of the total, was allocated for data preparation, model



(a) RandomForest (b) XGBoost

(c) SVM (d) KNN

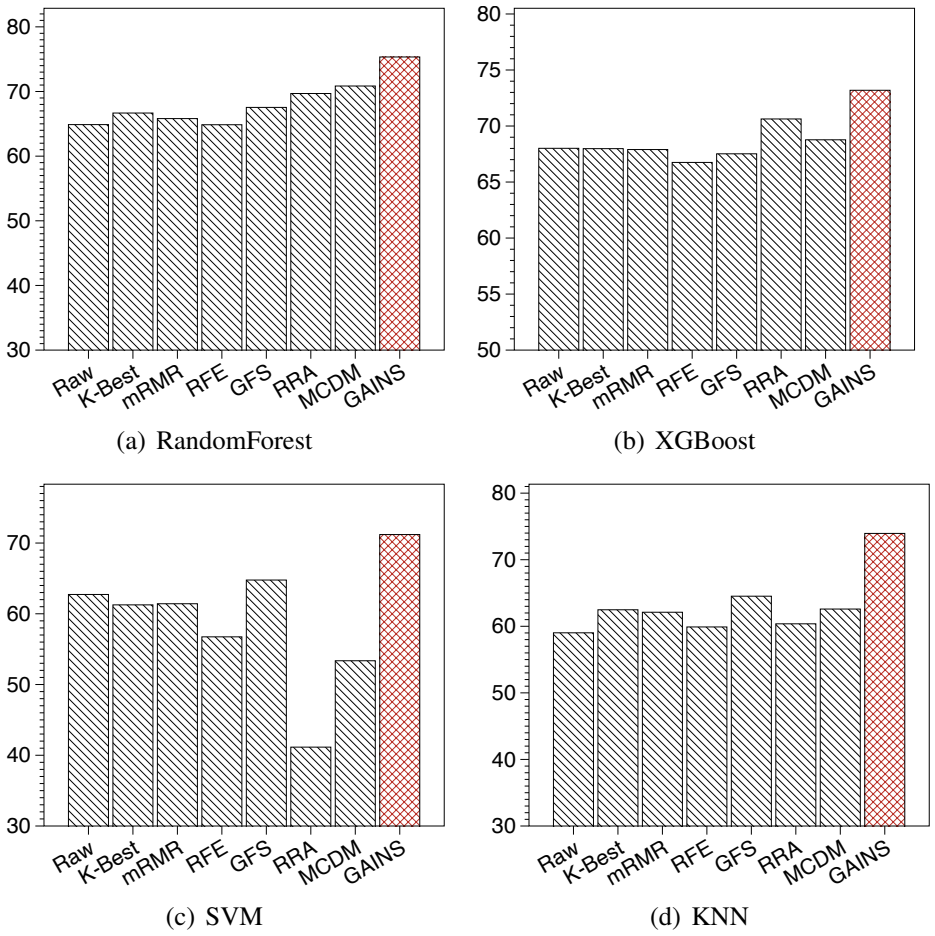
Figure 5.4: Robustness check of GAINS when confronted with distinct downstream ML models in terms of F1-score on German Credit.

training, and validation, whereas the remaining 20% functioned as a holdout set to evaluate the performance of the optimally constructed feature subset. We further explored this holdout setting and conducted an experiment to exemplify its effectiveness, with the results being depicted in Figure **??**. Figure **??** reveals that different datasets display distinct trends. For instance, the Ionosphere dataset's performance is enhanced with a smaller test split. Conversely, the performance on the Spectf dataset remains relatively constant, irrespective of fluctuations in the split setting. Moreover, a smaller test split shows a negative influence on the performance of the OpenML_607 and

Mice_Protein datasets. Various elements, such as the randomness of the dataset partitioning and the volume of the training data, can affect these datasets' performance. Notwithstanding, we observed consistently significant improvements between the performance lines of the optimized and original feature sets. This observation indicates that the holdout percentage setting does not influence our proposed method's optimization for the original datasets, thus substantiating the model's robustness. Therefore, his experiment demonstrates that GAINS can comprehend the feature selection knowledge and produce better feature selection results regardless of the hold-out percentages.

Robustness check of GAINS over downstream ML tasks.

This experiment aims to answer: *Does our proposed model exhibit robustness when confronted with various machine learning models serving as downstream tasks?* We proceeded to substitute the downstream machine learning model with Random Forest (RF), XGBoost (XGB), Support Vector Machine (SVM), and K-Nearest Neighborhood (KNN) respectively. Figure 5.5 shows the comparison results on the German Credit dataset in terms of the F1-score. It was observed that our proposed model consistently outperforms other baseline models regardless of the downstream model in use. The underlying driver for this observation is that GAINS has the capacity to customize the deep embedding space based on the performance evaluation conducted by a specific downstream machine learning model. This leads our model with a noteworthy degree of customization capability. Therefore, this experiment validates the robustness of our proposed model when confronted with different downstream tasks.



(a) RandomForest (b) XGBoost

(c) SVM (d) KNN

Figure 5.5: Robustness check of GAINS when confronted with distinct downstream ML models in terms of F1-score on German Credit.

Study of the size of selected feature subsets.

This experiment aims to answer this question: *Is our proposed model capable of selecting a small, yet effective, feature subset?* We randomly selected seven datasets and compared the size of the feature set of our proposed model with the best-performing baseline model and the original feature set. Figure 5.6 shows the comparative results, represented in terms of feature ratio. The feature ratio provides an indication of the proportion of selected features relative to the entire feature

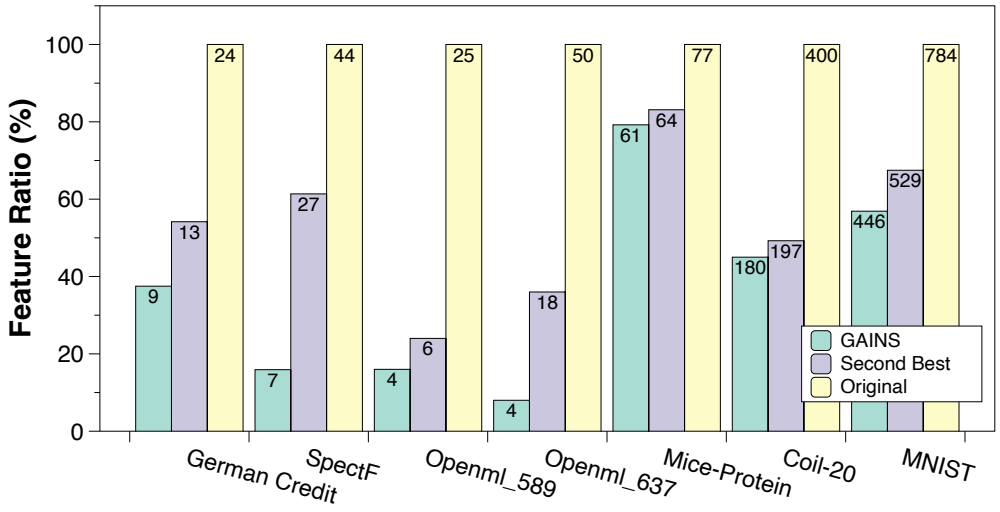


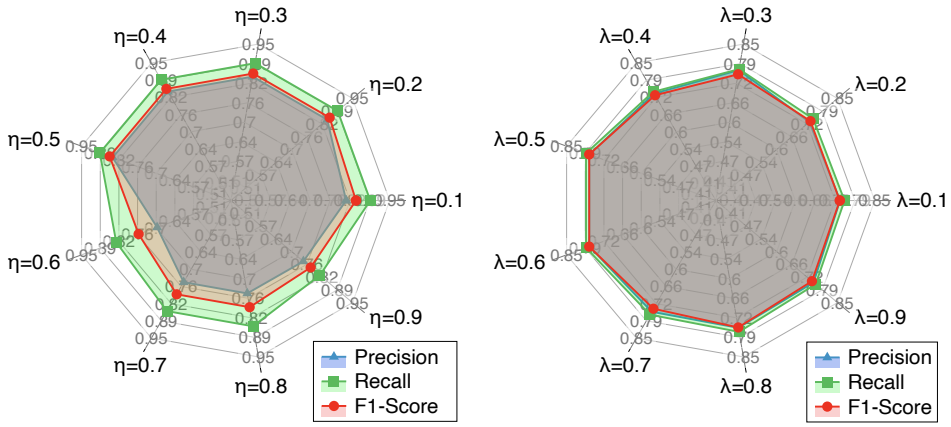
Figure 5.6: Comparison of the feature set size of GAINS, the best baseline model (Second Best), and the original feature set (Original).

set. We found that the feature subset selected by our model is substantially smaller than that of the second-best baseline, yet it maintains superior performance. A plausible explanation for this observation is that the joint optimization of feature subset reconstruction loss and accuracy estimation loss enhances the denoising capability of gradient search, reduces noise and redundancy in features, and ultimately selects a compact but effective feature subset. Therefore, this experiment demonstrates that the features selected by GAINS can not only improve model performance but also decrease computational expenses.

Study of the hyperparameter sensitivity of GAINS.

This experiment aims to answer: *To what extent is the performance of GAINS sensitive to the values of the trade-off parameter λ and step size η?* There are two major hyperparameters, training trade-off λ and the step size η of each search step. A higher λ will make the model more concentrated on the loss from the reconstruction of sequence, and a higher η make the model search more

aggressive. We set η varies from 0.1 to 1.0, and λ from 0.1 to 0.9, then train the GAINS on German Credit. The model performance is reported in Figure 5.7. Overall, we observed that the model performance will change slightly on different step size settings. Further, a balanced (i.e., 0.5 or 0.6) model training trade-off value will slightly bring a higher downstream performance. These findings provide insight into how η and λ impact the performance of our model and how we might choose optimal values for these hyperparameters.



(a) Step Size (b) Trade Off

Figure 5.7: The hyperparameter sensitivity test on German Credit.

Related Work

Feature Selection can be broadly categorized as wrapper, filter, and embedded methods according to the selection strategies [56]. Filter methods choose the highest-scoring features based on the feature relevance score derived from the statistical properties of the data [70, 7, 15]. For instance, K-Best [107] algorithm operates by ranking all features based on a specific criterion or metric. This criterion could be a correlation with the target variable, mutual information, or any other statistical measure that reflects the significance or importance of the feature. Once the features are

ranked, the top ”k” features are selected. These approaches have low computational complexity and can efficiently select features from high-dimensional datasets. But, they ignore feature-feature dependencies and interactions, resulting in suboptimal performance. Wrapper methods assess the quality of the selected feature subset based on a predefined machine learning (ML) model in an iterative manner [54, 64, 20, 19, 1]. The performance of these methods is typically superior to that of filter methods because they evaluate the entire feature set. MARLFS [61] adopted a multi-agent reinforcement approach to manipulate and construct the combination of the features, which is effective to search the solution. Besides, many reinforcement learning approaches [94, 103, 102] show great potential for feature engineering knowledge exploring. However, enumerating all possible feature subsets is an NP-hard problem, leading to cannot identify the optimal feature subset. Embedded methods convert the feature selection task into a regularization item in a prediction loss of the ML model to accelerate the selection process [86, 55, 27, 50, 49]. LassoNet [55] is a representative embedded method that combines the regularization properties of the Lasso and the expressive power of neural networks. The method is designed to handle high-dimensional data with intricate and non-linear relationships among the features. The key principle behind LassoNet is that it introduces sparsity, just like Lasso, but in the weights of a neural network. LassoNet, thus, combines Lasso (Least Absolute Shrinkage and Selection Operator) with a feedforward neural network. These methods may have outstanding performance on the incorporated ML model but are typically difficult to generalize to others. Moreover, other works have proposed hybrid feature selection methods, which have two technical categories: 1) homogeneous approach [78, 71]; 2) heterogeneous approach [32, 79]. Their performance is all limited by their basic aggregating methods. Unlike these works, GAINS proposes a new feature selection perspective, which maps historical discrete selection records into a continuous embedding space and then employs the gradient-based search to efficiently identify the optimal feature subset.

Conclusion Remarks

In chapter, our research has challenged traditional views of feature selection as a discrete choice problem and instead considered a novel research question: is it feasible to approach feature selection within a continuous space, thus enhancing its automation, effectiveness, and generalizability? In pursuit of an answer, we proposed a fresh perspective that transforms the discrete feature selection process into a gradient-optimized search, thereby reformulating feature selection as a continuous optimization task. Our study proposed a robust four-stage framework that integrates: 1) Automated reinforcement training data preparation, 2) Deep feature subset embedding, 3) Gradient-optimized feature subset search, and 4) Feature subset reconstruction. Our research findings have offered significant insights: 1) We demonstrated that reinforcement feature selection can act as a powerful tool for automated training data collection, significantly enriching the diversity, scale, and comprehensiveness of the training data. 2) We established that the joint optimization of the encoder, evaluator, and decoder can efficiently construct a continuous embedding representation space. 3) We found that treating discrete feature selection as a gradient search strategy can effectively reduce feature subset sizes and boost generalization. This novel method proved to be automated, effective, and input dimensionality agnostic. Ultimately, these findings underscore the potential of our proposed approach in improving the effectiveness and efficiency of automated feature selection, thereby opening up new avenues for future research in this domain.

CHAPTER 6: CONCLUSION AND FUTURE DIRECTIONS

Conclusion

In this dissertation, I focus mainly on two key areas in data-centric AI: feature generation learning and feature selection learning. More specifically, I propose two innovative research perspectives for the two tasks and present corresponding practical approaches:

1. *Decision-Making Perspective: Self-Optimizing Feature Generation Learning.* I introduce a group-wise reinforced feature generation framework designed to optimize feature space reconstruction. This framework formulates feature generation as an iterative process, structured into three Markov Decision Processes (MDPs). An innovative cascading agent structure is developed for selecting candidate features and operations, facilitating the creation and integration of new features into the existing space in each iteration. Furthermore, an M-clustering algorithm enhances the process's efficiency by forming robust feature clusters from the information theory perspective. This framework generates feature groups at each iteration instead of individual features. It aims to improve downstream task performance while reducing feature set redundancy.

2. *Decision-Making Perspective: Self-Optimizing Feature Selection Learning.* I formulate feature selection as a Markov Decision Process and introduce a reinforced, single-agent Monte Carlo-based approach for feature selection. This approach involves a reinforced agent specifically developed to navigate through the feature set. The agent is to make decisions on feature inclusion to optimize downstream task performance and minimize feature set redundancy. Moreover, an early stopping strategy is proposed to enhance training efficiency. The utilization of decision history, derived from the feature traversal strategy, diversifies training data,

thereby enhancing the robustness and effectiveness of the training process. Additionally, an interactive reinforcement learning strategy is employed to integrate domain knowledge to accelerate the learning procedure.

3. *Generative-AI Perspective: Autoregressive Feature Generation and Selection Learning.* I propose a unified optimization framework to reformulate feature generation and selection, reducing model complexity while improving performance. Drawing inspiration from generative AI, it hypothesizes that feature learning knowledge can be embedded into an embedding space for enhanced feature space exploration. The framework comprises four significant steps: 1) Data Collection: This stage gathers sufficient pairs of feature ID sequences and corresponding model accuracies, utilizing a self-optimizing feature learning framework. 2) Feature Learning Knowledge Embedding: An encoder-evaluator-decoder framework is introduced to embed feature learning knowledge within an embedding space. Each point in this space correlates to a specific feature generation or selection sequence and its associated model performance. 3) Gradient-steered Search: The process involves identifying locally optimal feature knowledge embeddings based on model performance. These embeddings are then navigated using gradients from a well-trained evaluator, aiming to maximize downstream task performance. 4) Feature Space Reconstruction: Enhanced embeddings are input into the well-trained decoder to regenerate feature generation or selection sequences. I then construct a new feature space following these sequences, ultimately selecting the highest-performing feature space as the final output.

Future Directions

In the future, the ultimate goal of this research is to develop a foundation model in the feature learning domain. To achieve this goal, I have outlined the potential research directions in the

following paragraphs:

D1: Large Sequential Model for Intelligent Feature Learning. The success of the sequential-generation-based feature space refinement framework I developed signifies the practicality of embedding feature learning knowledge into a continuous space. This motivates my intention to devise a robust sequential model optimized for the automated feature learning domain. Such a model can be efficiently and swiftly fine-tuned for varying sub-tasks. This process promises significant resource savings and improves the generalizability and applicability of existing feature learning techniques. To this end, I will focus on developing a practical large-scale model within the feature learning domain. This will be accomplished through three stages: 1) I will obtain a substantial collection of large, unlabeled datasets; 2) I aim to design a sequential model, potentially a Transformer, with the aim of embedding the acquired dataset knowledge into an effective embedding space; 3) under supervised conditions, I will fine-tune this well-trained model across different sub-domains to expedite the acquisition of optimal feature space.

D2: Federated Feature Learning for Distributed Data. In the practical context, data is often stored in disparate repositories across various regions, rendering the centralization of this data for feature engineering a challenging task due to constraints related to data confidentiality, transportation expenditures, and network bandwidth. In response to these challenges, I aim to develop a federated feature engineering pipeline. The execution of this objective will ensure through three distinct tasks: 1) The construction of a distributed learning framework capable of efficiently organizing and assimilating information from data stored in diverse repositories; 2 The formulation of privacy-preserving computation strategies tailored specifically for the aforementioned distributed learning framework; and 3) The proposal of parallel learning paradigms to accelerate the feature engineering process.

D3: Real-Time Feature Learning for Streaming Data. Despite the proven effectiveness and

efficiency of my prior work on automated feature engineering with static data, streaming data introduces a distinct set of challenges vital for real-world implementation. My proposed plan to adapt these techniques for real-time scenarios addresses the intricacies of feature engineering within streaming data, including distribution shifts, the discrepancy between slower feature engineering training speeds and data streaming speeds, and the constant fluctuation of feature addition and elimination. To effectively mitigate these challenges, I propose a threefold approach: 1) I will develop incremental updating strategies to reconcile the impact of both old and new data and capture distribution shifts; 2) I will design adaptive sampling windows to align the pace of feature engineering training with data streaming; 3) I propose the creation of innovative feature engineering pipelines adept at managing the complexities of cold-start feature space changes.

D4: Fairness-awareness in Feature Space Learning. This research trajectory, focused on instituting fairness in the feature learning domain, will specifically address two tasks: 1) ensuring fairness over demographic features and 2) maintaining fairness among noisy and distorted features. My research will explore the concept of fairness in feature learning through several key avenues: 1) Fairness Evaluation: The goal is to devise innovative feedback paradigms or metrics capable of justifying fairness within the framework of feature learning. 2) Unfairness Detection: This aims to identify any features that may introduce unfairness into downstream tasks. 3) Unfairness Elimination: This component is targeted at creating new feature engineering pipelines dedicated to eradicating discrimination. 4) Balancing Fairness and Discrimination: This entails the sensitive task of finding a compromise between fairness and other ethical considerations within the feature learning process.

LIST OF REFERENCES

[1] Mohammed Ghaith Altarabichi, Sławomir Nowaczyk, Sepideh Pashami, and Peyman Sheikholharam Mashhadi. Fast genetic algorithm for feature selection—a qualitative approximation approach. *Expert systems with applications*, 211:118528, 2023.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Maroua Bahri, Flavia Salutari, Andrian Putina, and Mauro Sozio. Automl: state of the art with a focus on anomaly detection, challenges, and research directions. *International Journal of Data Science and Analytics*, 14(2):113–126, 2022.

[4] Chunhui Bao and Qianru Sun. Generating music with emotions. *IEEE Transactions on Multimedia*, 2022.

[5] Richard Bellman and Robert Kalaba. Dynamic programming and statistical communication theory. *Proceedings of the National Academy of Sciences of the United States of America*, 43(8):749, 1957.

[6] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[7] Jacek Biesiada and Wlodzisław Duch. Feature selection for high-dimensional data—a pearson redundancy based filter. In *Computer recognition systems 2*, pages 242–249. Springer, 2008.

[8] Jock A. Blackard. Kaggle forest cover type prediction. [EB/OL], 2015. `https://www.kaggle.com/c/forest-cover-type-prediction/data`.

[9] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[10] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):1–37, 2011.

[11] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[12] Xiangning Chen, Qingwei Lin, Chuan Luo, Xudong Li, Hongyu Zhang, Yong Xu, Yingnong Dang, Kaixin Sui, Xu Zhang, Bo Qiao, et al. Neural feature search: A neural architecture for automated feature engineering. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 71–80. IEEE, 2019.

[13] Yi-Wei Chen, Qingquan Song, and Xia Hu. Techniques for automated machine learning. *ACM SIGKDD Explorations Newsletter*, 22(2):35–50, 2021.

[14] Lin Chih-Jen. Libsvm dataset download. [EB/OL], 2022. `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

[15] Hui Ding, Peng-Mian Feng, Wei Chen, and Hao Lin. Identification of bacteriophage virion proteins by the anova feature selection and analysis. *Molecular BioSystems*, 10(8):2229–2235, 2014.

[16] Ofer Dor and Yoram Reich. Strengthening learning algorithms by feature discovery. *Information Sciences*, 189:176–190, 2012.

[17] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[18] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[19] Wei Fan, Kunpeng Liu, Hao Liu, Ahmad Hariri, Dejing Dou, and Yanjie Fu. Autogfs: Automated group-based feature selection via interactive reinforcement learning. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 342–350. SIAM, 2021.

[20] Wei Fan, Kunpeng Liu, Hao Liu, Pengyang Wang, Yong Ge, and Yanjie Fu. Autofs: Automated feature selection via diversity-aware interactive reinforcement learning. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1008–1013. IEEE, 2020.

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

[22] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

[23] Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. *arXiv preprint arXiv:1702.01806*, 2017.

[24] Nicolo Fusi, Rishit Sheth, and Melih Elibol. Probabilistic matrix factorization for automated machine learning. *Advances in neural information processing systems*, 31:3348–3357, 2018.

[25] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[26] Pablo M Granitto, Cesare Furlanello, Franco Biasioli, and Flavia Gasperi. Recursive feature elimination with random forest for ptr-ms analysis of agroindustrial products. *Chemometrics and Intelligent Laboratory Systems*, 83(2):83–90, 2006.

[27] Pablo M Granitto, Cesare Furlanello, Franco Biasioli, and Flavia Gasperi. Recursive feature elimination with random forest for ptr-ms analysis of agroindustrial products. *Chemometrics and intelligent laboratory systems*, 83(2):83–90, 2006.

[28] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

[29] H. A. Guvenir, B. Acar, G. Demiroz, and A. Cekin. A supervised machine learning algorithm for arrhythmia analysis. In *Computers in Cardiology 1997*, pages 433–436, 1997.

[30] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[31] Mark A Hall. Feature selection for discrete and numeric class machine learning. 1999.

[32] Mohammad Nazmul Haque, Nasimul Noman, Regina Berretta, and Pablo Moscato. Heterogeneous ensemble combination search using genetic algorithm for class imbalanced data classification. *PloS one*, 11(1):e0146116, 2016.

[33] Amin Hashemi, Mohammad Bagher Dowlatshahi, and Hossein Nezamabadi-pour. Ensemble of feature selection algorithms: a multi-criteria decision-making approach. *International Journal of Machine Learning and Cybernetics*, 13(1):49–69, 2022.

[34] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC, 2019.

[35] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

[36] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[37] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[38] Franziska Horn, Robert Pack, and Michael Rieger. The autofeat python library for automated feature engineering and selection. *arXiv preprint arXiv:1901.07329*, 2019.

[39] Jeremy Howard. Kaggle dataset download. [EB/OL], 2022. `https://www.kaggle.com/datasets`.

[40] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pages 1–10. IEEE, 2015.

[41] Shubhra Kanti Karmaker, Md Mahadi Hassan, Micah J Smith, Lei Xu, Chengxiang Zhai, and Kalyan Veeramachaneni. Automl to date and beyond: Challenges and opportunities. *ACM Computing Surveys (CSUR)*, 54(8):1–36, 2021.

[42] Gilad Katz, Eui Chul Richard Shin, and Dawn Song. Explorekit: Automatic feature generation and selection. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 979–984. IEEE, 2016.

[43] Udayan Khurana, Fatemeh Nargesian, Horst Samulowitz, Elias Khalil, and Deepak Turaga. Automating feature engineering. *Transformation*, 10(10):10, 2016.

[44] Udayan Khurana, Horst Samulowitz, and Deepak Turaga. Feature engineering for predictive modeling using reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[45] Udayan Khurana, Deepak Turaga, Horst Samulowitz, and Srinivasan Parthasrathy. Cognito: Automated feature engineering for supervised learning. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 1304–1307. IEEE, 2016.

[46] YeongSeog Kim, W Nick Street, and Filippo Menczer. Feature selection in unsupervised learning via evolutionary search. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 365–369. ACM, 2000.

[47] W Bradley Knox, Peter Stone, and Cynthia 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*, pages 65–66, 2013.

[48] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[49] Kazuki Koyama, Keisuke Kiritoshi, Tomomi Okawachi, and Tomonori Izumitani. Effective nonlinear feature selection method based on hsic lasso and with variational inference. In *International Conference on Artificial Intelligence and Statistics*, pages 10407–10421. PMLR, 2022.

[50] Atsutoshi Kumagai, Tomoharu Iwata, Yasutoshi Ida, and Yasuhiro Fujiwara. Few-shot learning for feature selection with hilbert-schmidt independence criterion. *Advances in Neural Information Processing Systems*, 35:9577–9590, 2022.

[51] Andrew Kusiak. Feature transformation methods in data mining. *IEEE Transactions on Electronics packaging manufacturing*, 24(3):214–221, 2001.

[52] Hoang Thanh Lam, Johann-Michael Thiebaut, Mathieu Sinn, Bei Chen, Tiep Mai, and Oznur Alkan. One button machine for automating feature engineering in relational databases. *arXiv preprint arXiv:1706.00327*, 2017.

[53] Riccardo Leardi. Genetic algorithms in feature selection. pages 67–86, 1996.

[54] Riccardo Leardi. Genetic algorithms in feature selection. In *Genetic algorithms in molecular modeling*, pages 67–86. Elsevier, 1996.

[55] Ismael Lemhadri, Feng Ruan, and Rob Tibshirani. Lassonet: Neural networks with feature sparsity. In *International Conference on Artificial Intelligence and Statistics*, pages 10–18. PMLR, 2021.

[56] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–45, 2017.

[57] Yaliang Li, Zhen Wang, Yuexiang Xie, Bolin Ding, Kai Zeng, and Ce Zhang. Automl: From methodology to application. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 4853–4856, 2021.

[58] Long Ji Lin. Programming robots using reinforcement learning and teaching. In *AAAI*, pages 781–786, 1991.

[59] Jun S Liu, Rong Chen, and Wing Hung Wong. Rejection control and sequential importance sampling. *Journal of the American Statistical Association*, 93(443):1022–1031, 1998.

[60] Kunpeng Liu, Yanjie Fu, Pengfei Wang, Le Wu, Rui Bo, and Xiaolin Li. Automating feature subspace exploration via multi-agent reinforcement learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 207–215, 2019.

[61] Kunpeng Liu, Yanjie Fu, Pengfei Wang, Le Wu, Rui Bo, and Xiaolin Li. Automating feature subspace exploration via multi-agent reinforcement learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 207–215, 2019.

[62] Kunpeng Liu, Yanjie Fu, Le Wu, Xiaolin Li, Charu Aggarwal, and Hui Xiong. Automated feature selection: A reinforcement learning perspective. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[63] Kunpeng Liu, Pengfei Wang, Dongjie Wang, Wan Du, Dapeng Oliver Wu, and Yanjie Fu. Efficient reinforced feature selection via early stopping traverse strategy. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 399–408. IEEE, 2021.

[64] Kunpeng Liu, Pengfei Wang, Dongjie Wang, Wan Du, Dapeng Oliver Wu, and Yanjie Fu. Efficient reinforced feature selection via early stopping traverse strategy. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 399–408. IEEE, 2021.

[65] Steven N MacEachern, Merlise Clyde, and Jun S Liu. Sequential importance sampling for nonparametric bayes models: The next generation. *Canadian Journal of Statistics*, 27(2):251–267, 1999.

[66] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[67] Patrenahalli M. Narendra and Keinosuke Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on computers*, (9):917–922, 1977.

[68] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[69] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[70] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[71] Barbara Pes, Nicoletta Dessì, and Marta Angioni. Exploiting the ensemble paradigm for stable feature selection: a case study on high-dimensional genomic data. *Information Fusion*, 35:132–147, 2017.

[72] Public. Openml dataset download. [EB/OL], 2022. `https://www.openml.org`.

[73] Public. Uci dataset download. [EB/OL], 2022. `https://archive.ics.uci.edu/`.

[74] Maxim Raginsky, Alexander Rakhlin, and Matus Telgarsky. Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. *arXiv preprint arXiv:1702.03849*, 2017.

[75] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

[76] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

[77] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[78] Borja Seijo-Pardo, Verónica Bolón-Canedo, and Amparo Alonso-Betanzos. Testing different ensemble configurations for feature selection. *Neural Processing Letters*, 46(3):857–880, 2017.

[79] Borja Seijo-Pardo, Verónica Bolón-Canedo, and Amparo Alonso-Betanzos. On developing an automatic threshold applied to feature selection ensembles. *Information Fusion*, 45:227–245, 2019.

[80] Borja Seijo-Pardo, Iago Porto-Díaz, Verónica Bolón-Canedo, and Amparo Alonso-Betanzos. Ensemble feature selection: homogeneous and heterogeneous approaches. *Knowledge-Based Systems*, 118:124–139, 2017.

[81] Halit Bener Suay and Sonia Chernova. Effect of human guidance and state space size on interactive reinforcement learning. In *2011 Ro-Man*, pages 1–6. IEEE, 2011.

[82] V Sugumaran, V Muralidharan, and KI Ramachandran. Feature selection using decision tree and classification through proximal support vector machine for fault diagnostics of roller bearing. *Mechanical systems and signal processing*, 21(2):930–942, 2007.

[83] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[84] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[85] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[86] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[87] Lisa Torrey and Matthew 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*, pages 1053–1060, 2013.

[88] Binh Tran, Bing Xue, and Mengjie Zhang. Genetic programming for feature construction and selection in classification on high-dimensional data. *Memetic Computing*, 8(1):3–15, 2016.

[89] Peter Van Der Putten and Maarten van Someren. Coil challenge 2000: The insurance company case. 2000.

[90] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[91] Thibault Viglino, Petr Motlicek, and Milos Cernak. End-to-end accented speech recognition. In *Interspeech*, pages 2140–2144, 2019.

[92] Dakuo Wang, Josh Andres, Justin D Weisz, Erick Oduor, and Casey Dugan. Autods: Towards human-centered automation of data science. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2021.

[93] Dongjie Wang, Yanjie Fu, Kunpeng Liu, Xiaolin Li, and Yan Solihin. Group-wise reinforcement feature generation for optimal and explainable representation space reconstruction. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 1826–1834, New York, NY, USA, 2022. Association for Computing Machinery.

[94] Dongjie Wang, Yanjie Fu, Kunpeng Liu, Xiaolin Li, and Yan Solihin. Group-wise reinforcement feature generation for optimal and explainable representation space reconstruction. *Proceedings of the 28th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2022.

[95] Dongjie Wang, Kunpeng Liu, David Mohaisen, Pengyang Wang, Chang-Tien Lu, and Yanjie Fu. Automated feature-topic pairing: Aligning semantic and embedding spaces in spatial representation learning. In *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*, pages 450–453, 2021.

[96] Dongjie Wang, Pengyang Wang, Yanjie Fu, Kunpeng Liu, Hui Xiong, and Charles E Hughes. Reinforced imitative graph learning for mobile user profiling. *arXiv preprint arXiv:2203.06550*, 2022.

[97] Dongjie Wang, Pengyang Wang, Kunpeng Liu, Yuanchun Zhou, Charles E Hughes, and Yanjie Fu. Reinforced imitative graph representation learning for mobile user profiling: An adversarial training perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4410–4417, 2021.

[98] Dongjie Wang, Pengyang Wang, Jingbo Zhou, Leilei Sun, Bowen Du, and Yanjie Fu. Defending water treatment networks: Exploiting spatio-temporal effects for cyber attack detection. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 32–41. IEEE, 2020.

[99] Xiting Wang, Kunpeng Liu, Dongjie Wang, Le Wu, Yanjie Fu, and Xing Xie. Multi-level recommendation reasoning over knowledge graphs with reinforcement learning. In *Proceedings of the ACM Web Conference 2022*, pages 2098–2108, 2022.

[100] Marcel Wever, Alexander Tornede, Felix Mohr, and Eyke Hüllermeier. Automl for multi-label classification: Overview and empirical evaluation. *IEEE transactions on pattern analysis and machine intelligence*, 43(9):3037–3054, 2021.

[101] Meng Xiao, Dongjie Wang, Min Wu, Kunpeng Liu, Hui Xiong, Yuanchun Zhou, and Yanjie Fu. Self-optimizing feature transformation. *arXiv preprint arXiv:2209.08044*, 2022.

[102] Meng Xiao, Dongjie Wang, Min Wu, Kunpeng Liu, Hui Xiong, Yuanchun Zhou, and Yanjie Fu. Traceable group-wise self-optimizing feature transformation learning: A dual optimization perspective. 2023.

[103] Meng Xiao, Dongjie Wang, Min Wu, Ziyue Qiao, Pengfei Wang, Kunpeng Liu, Yuanchun Zhou, and Yanjie Fu. Traceable automatic feature transformation via cascading actor-critic agents. *arXiv preprint arXiv:2212.13402*, 2022.

[104] Tengyang Xie, Yifei Ma, and Yu-Xiang Wang. Towards optimal off-policy evaluation for reinforcement learning with marginalized importance sampling. In *Advances in Neural Information Processing Systems*, pages 9668–9678, 2019.

[105] Jihoon Yang and Vasant Honavar. Feature subset selection using a genetic algorithm. In *Feature extraction, construction and selection*, pages 117–136. Springer, 1998.

[106] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *Icml*, volume 97, page 35. Nashville, TN, USA, 1997.

[107] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *Icml*, volume 97, page 35. Nashville, TN, USA, 1997.

[108] Lei Yu and Huan 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)*, pages 856–863, 2003.

[109] Yang Yu. Towards sample efficient reinforcement learning. In *IJCAI*, pages 5739–5743, 2018.

[110] Ziwei Zhang, Xin Wang, and Wenwu Zhu. Automated machine learning on graphs: A survey. *arXiv preprint arXiv:2103.00742*, 2021.

[111] Xiaosa Zhao, Kunpeng Liu, Wei Fan, Lu Jiang, Xiaowei Zhao, Minghao Yin, and Yanjie Fu. Simplifying reinforced feature selection via restructured choice strategy of single agent. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 871–880. IEEE, 2020.

[112] Guanghui Zhu, Shen Jiang, Xu Guo, Chunfeng Yuan, and Yihua Huang. Evolutionary automated feature engineering. In *PRICAI 2022: Trends in Artificial Intelligence: 19th Pacific Rim International Conference on Artificial Intelligence, PRICAI 2022, Shanghai, China, November 10–13, 2022, Proceedings, Part I*, pages 574–586. Springer, 2022.

[113] Guanghui Zhu, Zhuoer Xu, Chunfeng Yuan, and Yihua Huang. Difer: differentiable automated feature engineering. In *International Conference on Automated Machine Learning*, pages 17–1. PMLR, 2022.