AUTOMATED FEATURE ENGINEERING WITH REINFORCEMENT LEARNING

A Thesis

by

ZIYU XIANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2020

Major Subject: Electrical & Computer Engineering

ABSTRACT

Automatic Feature Engineering (AFE) aims to extract useful knowledge for interpretable predictions given data for the machine learning tasks of interest. Here, we develop AFE to extract dependency relationships that can be interpreted with functional formulas in order to discover physics meaning or new hypotheses for the problems of interest. We focus on materials science applications, where interpretable predictive modeling may provide a principled understanding of materials systems and guide new materials discovery. It is often computationally prohibitive to exhaust all the potential relationships to construct and search the whole feature space to identify interpretable and predictive features. We develop and evaluate new AFE strategies by exploring a feature generation tree with deep Q-network for scalable and efficient exploration policies. The developed reinforcement learning based AFE strategies are benchmarked with the existing AFE methods on several materials science datasets.

ACKNOWLEDGMENTS

I would like to thank Dr. Xiaoning Qian's and Dr. Dileep Kalathil's valuable guidance for this project. I would also like to thank Dr. Tie Liu and Dr. Xia Hu for carefully reviewing the presented work. Special thanks to Drs. Raymundo Arróyave and Xiaofeng Qian for providing datasets for the experiments. We acknowledge the support of Texas A&M High Performance Research Computing (HPRC) for implementing the experiments and obtaining the results.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Xiaoning Qian, Professor Dileep Kalathil and Professor Tie Liu of the Department of Electrical & Computer Engineering and Professor Xia Hu of the Department of Computer Science and Engineering.

The data analyzed for Chapter 3 were provided by Dr. Raymundo Arróyave and Xiaofeng Qian in the Department of Materials Science and Engineering as well as data from a published article [1]. The dataset description for regression section in Chapter 3 is provided by Guillermo Vazquez and William Trehern.

The implementing of the experiments reported in Chapter 3 is supported by Texas A&M High Performance Research Computing (HPRC) platform.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

No outside funding was received for the research and writing of this document

NOMENCLATURE

AFE	Automated Feature Engineering
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
FE	Feature engineering
FGT	Feature Generation Tree
HPRC	Texas A&M High Performance Research Computing
LASSO	Least Absolute Shrinkage and Selection Operator
MDP	Markov Decision Process
MFE	Manual Feature Engineering
ML	Machine Learning
RL	Reinforcement Learning
SISSO	Sure Independence Screening and Sparse Operation

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

Feature engineering (FE) is the process of creating features that capture hidden dependency relationships in data and improve the prediction performances of machine learning (ML) algorithms [2]. It usually involves two aspects: generating new feature representations by transforming the original raw or primary features in the given dataset, and selecting those engineered features that are important (interpretable and predictive) for the ML tasks. Such prepossessing pipelines and data transformations are crucial and often take most of the actual efforts in deploying ML algorithms since the prediction performance of ML algorithms is heavily dependent on features [3, 4].

However, traditional FE is a labor-intensive and time-consuming task, which requires complex exercises, being performed in an iterative manner with trial and error and being driven by domain knowledge developed over time [3, 5]. Thus such methods are usually problem-specific and not generally applicable to different datasets, limiting their direct adoption in corresponding applications, especially when both domain knowledge and available training data are scarce. Compared to traditional FE methods, Automated Feature Engineering (AFE) [5, 6] has been recently introduced to automatically extract complex feature representations based on raw features (using deep learning for example), which has attracted much attention in recent literature. Many black-box deep neural network based AFE models [7, 8] have shown their great potential to improve the corresponding ML algorithms' performance and be general to be implemented on different datasets without too much additional manual labor. However, on the one hand, the brute force way to generate and select features by exhausting the possible feature transformations costs too much time and is difficult to complete with reasonable scalability to the number of original raw features. On the other hand, a good interpretability to the generated features is usually hard to attain through such black-box methods.

For the materials science problems we study in this paper, finding the actuating mechanisms of a certain property or function and describing it in terms of a set of physically meaningful variables is the desired scientific solution [9]. Such a set of physical variables with corresponding parameters that uniquely describe the material and its function of interest, can be denoted as "descriptors". One of the purposes of discovering descriptors in materials-science data, is to predict a target functional property of interest for a given complete class of materials [10]. Hence, AFE for materials science faces two main challenges to build better ML models: a good interpretability of the engineered descriptors from the raw features, and the scalability and efficiency to select important descriptors from the often enormous generated feature space for the given target of interest.

In this paper, we propose a Feature Generation Tree (FGT) and focus on novel AFE strategies by combining FGT exploration with Deep Reinforcement Learning (DRL) [11] to address both the interpretability and scalability challenges. Instead of employing a brute-force way to perform algebraic operations on the raw features in a given dataset and then selecting important descriptors, we combine the generating and selecting processes together by constructing FGTs and developing the corresponding tree exploration policies guided by deep reinforcement learning. An efficient exploration of the prominent descriptors can be attained in the growing feature space based on the allowed algebraic operations, and our new AFE strategies, constructing interpretable descriptors based on a list of operations according to the DRL learned policies, are more scalable and flexible with the performance-complexity trade-off with the help of adjustable batch size for generating intermediate features. When additional prior knowledge, such as physics constraints on applying appropriate algebraic operations to specific feature groups, is available, it can be readily incorporated in our new AFE procedure to produce physically-meaningful descriptors. Our experiments on several materials science datasets show that better predictions of material properties can be achieved using our new AFE methods with less memory and running time compared to existing brute-force AFE methods [1].

1.2 Related Work

An desirable FE method should attain considerable improvement of model prediction performance, generalizability, as well as good interpretability with little manual labor. Thus, the first AFE method proposed in [12], Deep Feature Synthesis, extracts features based on explicit functional relationships without experts' domain knowledge based on an algorithm to automatically generate features through stacking multiple primary features and implementing operations or transformations on them to get engineered features. But it suffers from efficiency and scalability problems due to its brute force way to generate and select features. Kaul et al. [6] proposed an Autolearn framework using a regression-based feature learning algorithm to generate and select features by mining pairwise feature associations to select those relationships that are stable and improve the prediction performance. While such an AFE method avoids overfitting, to which deep Learning based FE methods are amenable, and improves the efficiency by selecting subsets of engineered features according to stability and information gain, it does not directly produce a set of intepretable features. Khurana et al. [5] introduced the Cognito framework, which formulates the feature engineering problem as a search on the transformation tree, and developed an incremental search strategy to explore the prominent features and later extended the framework by combining reinforcement learning with a linear functional approximation [13] to improve the efficiency. This framework enables learning the policy to generate features from different datasets. A similar framework has recently been developed by Zhang et al. [14], who also used a tree-like transformation graph with the search policy derived by deep reinforcement learning. It improves the policy learning capability by utilizing deep neural networks compared to the linear functional approximation in Cognito. However, both frameworks generate features without explicitly incorporating available prior knowledge into the AFE procedures.

For AFE in materials science applications, several methods have been developed, such as the method based on compressed sensing [10] and more recent Sure Independent Screening and Sparse Operation (SISSO) method [1] that uses the brute force way to generate features and then select subsets of generated features by the sure independent screening [15] together with sparse operators such as Least Absolute Shrinkage and Selection Operator (LASSO) [16]. These methods pose an scalability challenge with the exponentially growing memory requirement to store intermediate features and significantly high computational complexity to search for desired features.

1.3 Statement of the Problem

Given a dataset $D_0 =< F_0, y >$, where F_0 denotes the finite set of p variables as raw or primary features $\{f_0^0, f_0^1, ..., f_0^p\}$ and y denotes the target vector, we need to construct sets of engineered features $F_i = \{g_1(F_0, c_1), g_2(F_0, c_2), \dots\}$ based on functional forms with allowed algebraic operations to generate interpretable and predictive descriptors for y. The function $g_m(\cdot)$ consists of a set of algebraic operations ϕ, from an operation set O, implemented on features in F_0. The operation set O can be pre-defined, for example, with the following unary and binary operations:

$$O = \{exp(\cdot), log(\cdot), (\cdot)^2, (\cdot)^3, (\cdot)^{-1}, \sqrt{\cdot}, \sqrt[3]{\cdot}, +, -, \times, \div\}. \tag{1.1}$$

For each function, c_i represents the complexity of the corresponding generated feature—the number of algebraic operations in one function. For example, the function $\exp(f_0^0) \times (f_0^1)^2 + \sqrt{(f_0^2)}$ has the complexity of 5. Here, F_i denotes the iteratively generated feature set with the maximum allowed complexity c_i.

From the whole feature space $\mathbb{F} = F_0 \cup F_i's$, our goal is to find such an optimal feature set $F^* = \{(f^1)^*, (f^2)^*, \dots\}(\forall (f^d)^* \in F^*, (f^d)^* \in \mathbb{F})$ that maximizes the prediction performance score, for example by classification or regression accuracy, $A_L\{F', y\}$:

$$F^* = \underset{\forall f^k \in F', f^k \in \mathbb{F}, c_i < c_{max}}{\arg\max} A_L\{F', y\}, \tag{1.2}$$

where L denotes the prediction model, which can be linear regression or Support Vector Machine (SVM) for interpretability with generated features, and F' denotes the set of all generated features.

2. METHODOLOGY

In this section, we introduce our new AFE strategies, which are based on the formulated feature generation tree (FGT) exploration guided by DEEP Q-network. More critically, we facilitate a flexible intermediate feature generation procedure that helps achieve good performance-complexity trade-off as well as flexible integration of available physics constraints as prior knowledge.

2.1 Feature Generation Tree (FGT)

To approximate the optimal feature set F^*, we introduce the Feature Generation Tree (FGT) illustrated in Figure 2.1 to iteratively construct the feature space and transform the problem into a tree search problem for efficient AFE. Each node in FGT represents a feature set F_i and each edge represents an operation ϕ. We denote $(F^d)^* = \{(f^1)^*, (f^2)^*, \ldots, (f^d)^*\}$ as the top d optimal features when we choose the cardinality of F^* as d, and $(f^d)^*$ as the selected optimal feature for the dth dimension of $(F^d)^*$. The FGT exploration aims to search for the optimal features $(f^1)^*, (f^2)^*, \ldots$ one by one. The corresponding complete AFE procedure constructs the feature subspace F^d sequentially as the search space of each $(f^d)^*$ exploration. Iterations start from the root node F_0, which represents the primary feature set. At some node F_i, we choose an operation ϕ_i according to the policy π as detailed in the following subsection, then move forward to the next node, generate the new feature set $F_j (j > i)$ and add the generated features to F^d. Meanwhile, the generated new feature set $F' = (F^{d-1})^* \cup \{f^d\}$ will be fed to the predictive model L to obtain the score $A_L\{F', y\}$, where $(F^{d-1})^* = \{(f^1)^*, (f^2)^*, \ldots, (f^{d-1})^*\}$ for each $(f^k)^* \in F^k (1 \leqslant k \leqslant d-1)$, representing the top $d-1$ optimal feature set chosen from the previous feature subspace F^k; and $f^d \in F^d$. The FGT will grow by repeating the operations above until it attains the maximum complexity c_{max}. Then a new iteration will start again to grow another FGT to find $(f^d)^*$. When F' achieves our desired prediction performance score, we will stop or look for the next optimal descriptor $(f^{d+1})^*$. Note that the feature subspace F^d is the union of all F_i's in all explored FGT's when looking for $(f^d)^*$, and the whole feature space $\mathbb{F}$ is the union of all F^d's.

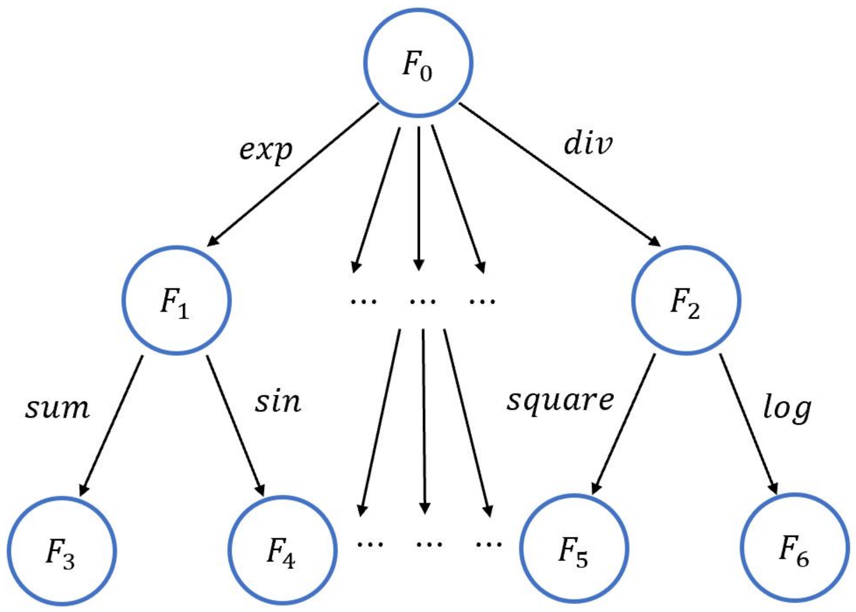


Figure 2.1: Example of a feature generation tree

2.2 Reinforcement learning for FGT exploration

AFE by our FGT exploration can be considered as a finite Markov Decision Process (MDP) problem. Considering the huge feature space and the large available operation set we may have, we adopt the Deep Q-Network (DQN) [11] with experience replay to learn the policy π for choosing ϕ's during FGT exploration. Formally, we define the states, actions and rewards for our AFE formulation as follows:

- **state**: F_i^d, which denotes a primary or generated feature set when looking for the dth optimal descriptor;
- **action**: $\pi(F_i^d) = \phi_i$, which denotes an operation in the operation set **O**;
- **reward**: $R(F_i^d, \phi_i) = \max\limits_{F'}(1.001 - A_L\{F', y\})^{-1}$, where $0 \leqslant A_L\{F', y\} \leqslant 1$ and $F' = (F^{d-1})^* \cup \{f_d\}$, for $(F^{d-1})^* = \{(f^1)^*, (f^2)^*, \ldots, (f^{d-1})^*\}$ for each $(f^k)^* \in F^k (1 \leqslant k \leqslant d-1)$ and $f^d \in \phi_i(F_i^d)$.

Each node in FGT represents a state and each edge in FGT represents an action. We always want to find the optimal F^* with the lowest cardinality to meet the desired prediction performance score threshold and therefore our AFE will first look for $(F^1)^*$, and then look for $(F^2)^*$, or more if the set of resulting descriptors does not meet the score threshold within the allowed budget (maximum number of exploration iterations).

To have a flexible exploration procedure for both performance-complexity trade-off and incorporation of prior knowledge, for each $(f^d)^*$ in F^*, it can be chosen from the top n features with highest rewards in the corresponding feature subspace F^d, which compose a candidate set S^d. So $(F^{d-1})^*$ can have multiple combinations according to the whole candidate sets $\mathbb{S} = \{S^1, \ldots, S^{d-1}\}$, and F' also has multiple combinations according to different $(F^{d-1})^*$ and f^d. Consequently the reward is computed as the maximum reward over F'.

It is worth noticing that we have different types of algebraic operations in O. When we apply unary operations ϕ_u on a feature set F_i, it will apply ϕ_u on all the features in F_i and results in the new generated feature set $F_j = \{\phi_u(f^1), \phi_u(f^2), \ldots\}$. However, when we apply binary operations ϕ_b on F_i, beside the one feature in the F_i, we have to choose another one feature in the whole feature space to complete the operation, resulting the exploding of the corresponding feature subspace with the new generated feature set as $F_j = \{\phi_b(f^1, f^s)\} \cup \{\phi_b(f^2, f^s)\} \cup \ldots$. Clearly, if we enumerate f^s from all the features in $\mathbb{F}$, it is computationally prohibitive as F_j grows exponentially. Thus, we introduce the flexible batch sampling to randomly sample a feature subspace B from $\mathbb{F}$ as a "Batch Set" each time and enumerate f^s only from B to achieve the performance-complexity trade-off, and take the maximum reward from all the combinations as the reward. When prior knowledge is available as physics constraints on applying corresponding operations to specific feature groups, this batch sampling procedure can naturally take care of them. The pseudo-code for the basic AFE strategy of FGT exploration with DQN is provided in Algorithms 1 and 2.

Algorithm 1 DQN for Automatic Feature Engineering

```
1: input: Primary features $F_0$, Action set $H$
2: for $d = 1, 2, \ldots$ do
3:     Construct new DQN
4:     Clear Buffer
5:     for $episode = 1, 2, \ldots, N$ do
6:         for $i = 0, 1, \ldots,$ do
7:             $\phi_i = \epsilon$-Greedy Method($F_i, \epsilon$)
8:             $F_{i+1}, R_i, c_i =$ FGT_Grow $\{F_i, \phi_i, c_i\}$
9:             Buffer $\leftarrow \{F_i, F_{i+1}, \phi_i, R_i, c_i\}$
10:            Train DQN with experience replay
11:            if $R_i > threshold$ then
12:                goto Output
13:            end if
14:            if $c_i \geq c_{max}$ then
15:                break
16:            end if
17:        end for
18:    end for
19:    $\mathbb{S} \leftarrow$ Candidate set $S^d$ with $n$ features of highest $R_i$
20: end for
21: Output: Optimal feature set $F^*$ chosen from $\mathbb{S}$
```

Algorithm 2 FGT_Grow

```
1: input: Feature set $F_i$, action $\phi_i$, complexity$c_i$
2: if $\phi_i$ is unary then
3:     $F_{i+1} = \phi_i(F_i)$, $c_{i+1} = c_i + 1$, $R_i = R(F_i, \phi_i)$
4: else
5:     $L = \emptyset$
6:     Randomize $B$ from feature space $\mathbb{F}$
7:     for each $f^s$ in $B$ do
8:         $L \leftarrow \phi_i(F_i, f^s)$
9:     end for
10:    Select $F_{i+1}$ from $L$ with maximum $R(F_i, \phi_i)$
11:    $R_i = R(F_i, \phi_i)$, $c_{i+1} = c_i + (c_s$ of $f^s)$
12: end if
13: $\mathbb{F} \leftarrow F_{i+1}$
14: Output: Feature set $F_{i+1}$, Reward $R_i$, Complexity $c_{i+1}$
```

3. EXPERIMENTS AND DISCUSSION

To evaluate our proposed AFE strategies, we perform experiments with three real-world materials science datasets: one for classification of metal/non-metal materials, one for regression to get alloy elastic behavior based on alloy compositions, and the third dataset for predicting material's phase transition temperature with the physics constraints of feature groups. In these experiments, we assume that we have a limited computation budget and set the same upper bound of the allowed runtime for each experiment. We run all the experiments on Texas A&M High Performance Research Computing(HPRC) platform with the hardware configuration of Intel Xeon E5-2670, 64GB 1866MHz RAM and 2 NVIDIA k20 GPUs. In the following experiments, "descriptors" denote the engineered features in the final optimal feature set. For each dataset, we perform each algorithm with the same setup 5 times to report descriptors based on the best performing sets of descriptors obtained in the given runtime budget in one run, as well as use the average of the size of feature space and running time for the scalability and efficiency comparison.

For DQN, we have adopted a two-layer Q-network with the corresponding hidden dimensions {150,120} and the $relu$ activation function is used for both layers. The following hyperparameters are set for DQN training: Learning rate: 0.001; Experience replay batch size: 64; Gamma: 0.99; Epsilon: 1.0 (decay 0.99 and min 0.05). For SISSO, we used the code provided by [1], and set the same max complexity, operation set and the number of descriptors as DQN's, as well as SIS-selected subspace to be 5,000.

3.1 Classification

The classification problem is based on a dataset of 9 prototype structures $(NaCl, CsCl, ZnS, CaF_2, Cr_3Si, SiC, TiO_2, ZnO, FeAs, NiAs)$ with a total number of 260 materials from one of the experiments reported in [1], which includes seven primary features $(IE_A, IE_B, \chi_A, \chi_B, x_A, x_B, V_{Cell}/\sum V_{atom})$ for each material. The problem is to predict whether a given material is metal or not as a classification problem.

In this experiment, we adopt a Support Vector Machine (SVM) with the linear kernel as the classification model. When generating complex features/descriptors, we limit to an operation set including $\{exp(\cdot), log(\cdot), (\cdot)^2, \sqrt{\cdot}, +, -, \times, \div\}$ to search a set of two descriptors. We have limited the upper bound of the running time for each descriptor to be 4 hours and also set the same maximum feature complexity c_{max} to 5 to have the fair comparison with the adopted AFE strategy SISSO in [1].

When applying binary operations to generate new features, we explore different sizes of Batch Sets as **B** in Table 3.2 to evaluate the performance-complexity trade-off by limiting the total number of combinations of feature pairs for corresponding binary operations to speed up the feature generating process. In order to study the influence of the size of Batch Sets, we have tested five batch sizes: $\{5, 00, 1, 000, 2, 000, 4, 000, 8, 000\}$. We also compare our AFE strategies with one-step greedy AFE, which replace DQN with selecting engineered features giving the best prediction performance at each step in FGT exploration without considering future predictive power, and SISSO [1], which constructs and selects features by brute-force exhausting the potential feature space with increasing complexity.

To further investigate the generalizability of generated descriptors, we perform hold-out testing with two ways of splitting the samples into the training and test sets respectively. One way is to randomly split the dataset to be 7:3 with 182 materials in the training set and the remaining 78 materials in the test set, which are not used for feature generation. However, as materials belonging to the same prototype may share information, to further guarantee that the AFE and model training processes do not get additional information about testing data, the other hold-out testing setup is to use one prototype of materials, for example CaF_2, to be the test set, and all the remaining prototypes of materials as the training set. In this setup, 225 materials are in the training set and 35 materials are in the test set. Table 3.1 provides the best performing descriptors in 5 runs for both the "random" and "type" splits for training and test sets.

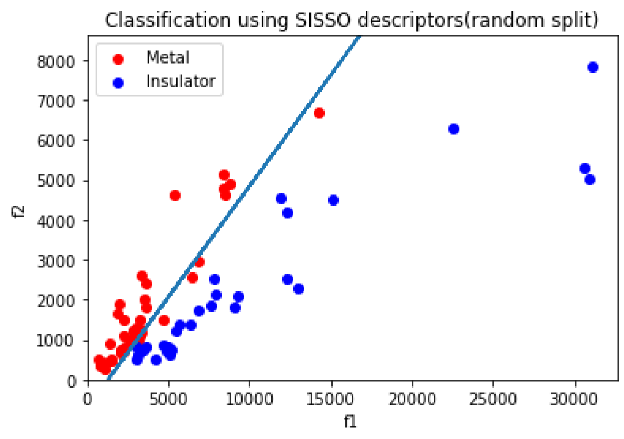
In Table 3.1, we can see that for the random split test set, compared with the primary features in the dataset, both one-step greedy and our AFE strategies with DQN can engineer predictive

Test Set	Methods	Descriptors	Accuracy	
			Training	Prediction
Random Split	Primary Features	$IE_A, IE_B, \chi_A, \chi_B, x_A, x_B, V_{Cell}/\sum V_{atom}$	0.9780	0.9615
	SISSO	$(f^1)^* = (IE_B/x_A)^2(\chi_B + \frac{V_{Cell}}{\sum V_{atom}})/x_B$ $(f^2)^* = (IE_B/x_A)^2 \exp{(\chi_A/\frac{V_{Cell}}{\sum V_{atom}})}$	1.000	0.9359
	One-step Greedy	$(f^1)^* = x_B + \frac{V_{Cell}}{\sum V_{atom}} + \chi_B/\chi_A$ $(f^2)^* = \frac{V_{Cell}}{\sum V_{atom}} - IE_B + \chi_A + IE_A IE_B/\chi_A$	0.9725	0.9615
	DQN	$(f^1)^* = x_A - \chi_A + \chi_B + \frac{V_{Cell}}{\sum V_{atom}} - \sqrt{\chi_A}$ $(f^2)^* = (\chi_B + \frac{V_{Cell}}{\sum V_{atom}}(IE_A + \chi_B)/\chi_A)/IE_A$	0.9890	0.9872
Type Split	Primary Features	$IE_A, IE_B, \chi_A, \chi_B, x_A, x_B, V_{Cell}/\sum V_{atom}$	0.9822	0.9429
	SISSO	$(f^1)^* = (IE_B\frac{V_{Cell}}{\sum V_{atom}})^2/(\chi_A(\chi_A - x_A))$ $(f^2)^* = \chi_B\chi_B^2 \exp{(IE_A/\chi_A)}$	1.0000	0.8857
	One-step Greedy	$(f^1)^* = \log\chi_B - \chi_A/\frac{V_{Cell}}{\sum V_{atom}}$ $(f^2)^* = \log IE_B + \chi_B/\chi_A - x_A$	0.9777	0.9429
	DQN	$(f^1)^* = \frac{V_{Cell}}{\sum V_{atom}} + x_B + \chi_B - \chi_A - \chi_A x_B$ $(f^2)^* = \chi_B - x_A - \log IE_B$	0.9822	0.9429

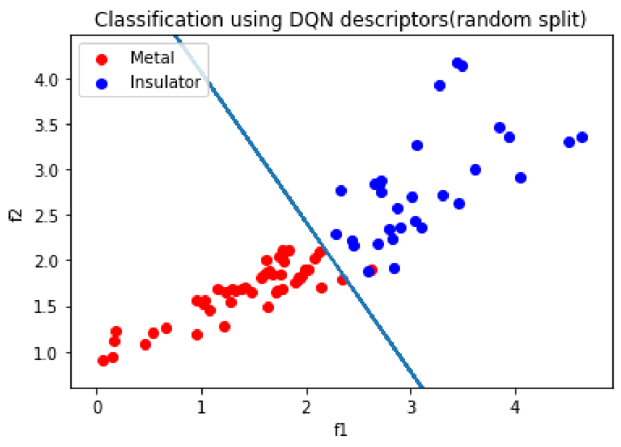
Table 3.1: Metal vs. non-metal classification descriptors

Test Set	Average Performance	Methods					
		SISSO	DQN				
			B=5,00	B=1,000	B=2,000	B=4,000	B=8,000
Random Split	Feature Space	59,597,864	182,022	163,563	232,880	298,149	162,974
	Runtime (hours)	6.2	3.4	2.6	3.8	4.7	2.8
Type Split	Feature Space	58,614,708	168,668	228,652	276,840	210,435	263,928
	Runtime (hours)	3.9	2.7	3.2	3.9	2.8	4.1

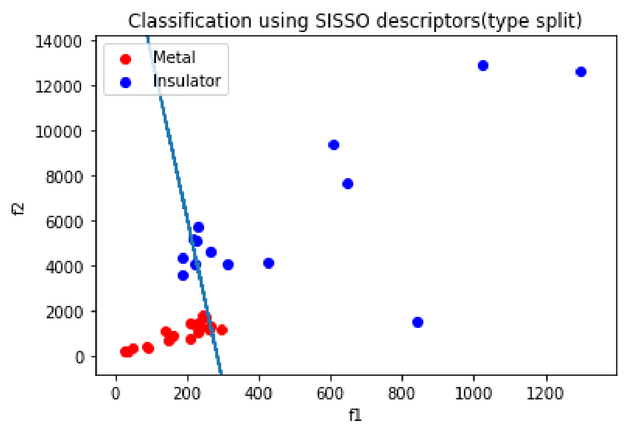
Table 3.2: Scalability and efficiency comparison for metal vs. non-metal classification (Note: 'Feature Space' here refers to the total number of generated intermediate features until the final descriptors are found.)



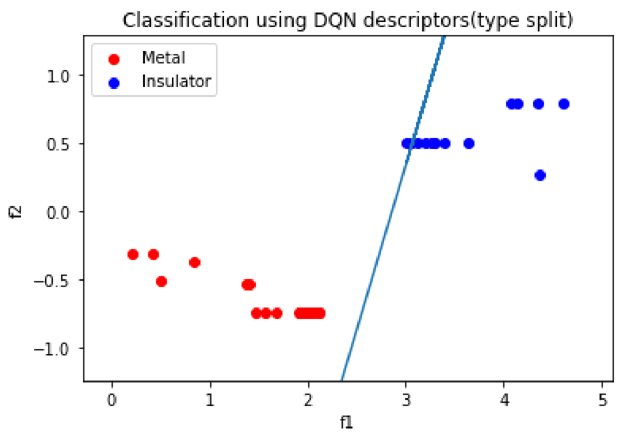
(a) SISSO descriptors with random split



(b) DQN descriptors with random split



(c) SISSO descriptors with type split



(d) DQN descriptors with type split

Figure 3.1: Metal vs. non-metal classification results in the test set

descriptors that improve the prediction accuracy compared to the model with primary features. When constructing features using training samples, SISSO and DQN strategies can get the best training accuracy. However, as shown in the table, SISSO is prone to overfitting and has a rather lower prediction accuracy than the model with primary features. By contrast, DQN has more stable and significant improvement on the prediction accuracy. For the type split test set, SISSO has the best training accuracy, but again it overfits. Our AFE strategies with DQN has the best prediction accuracy among all feature engineering strategies but we note that none of them improves over the predictive model with primary features, which demonstrates the difficulty in applying machine learning methods in materials science applications as the materials systems are often heterogeneous with potential nonlinear phase transitions. Without physics knowledge or sufficient data, simpler predictive models may be more robust. Particularly when we are interested in using these simpler models to suggest new materials to synthesize in order to meet a specific performance requirement. Figure 3.1 shows the classification results of the SISSO strategy and DQN strategy in both split methods and proves that DQN strategy is more robost compared with SISSO strategy in both split methods to avoid overfitting.

In Table 3.2, we compare the scalability and efficiency of SISSO and DQN by calculating the average number of generated intermediate features and the total runtime to identify interpretable and predictive descriptors. For our FGT exploration with DQN, different sizes of Batch Sets B have been tested. In this experiments, all these different setups attain the similar best training accuracy with the same runtime budget. Together with the results in Table 3.1, we can see that our AFE strategies with DQN can attain similar training accuracy and better prediction accuracy compared to the results by SISSO while our strategies construct a much smaller feature space than SISSO's to identify the predictive descriptors. In addition to better scalability, with a suitable Batch Set size, for example, $B =$500, 1,000, 2,000 or 4,000, our AFE strategies with DQN can also be computationally more efficient than SISSO, taking less runtime to find the descriptors. This is expected as with a small Batch Set size, FGT exploration can iterate faster and DQN can also converge to stable policies faster; while with a large Batch Set size, our AFE strategies generate more intermediate features in the binary operation steps and thus can lower the chance of overlooking some promising features with long-term benefits in increasing predictive power but higher memory and runtime requirements.

3.2 Regression

We implement our AFE strategies for another materials science dataset studying alloy elastic constants, in which 14 primary features are made from elemental properties using the rule-of-mixtures: crystal radius (CR), atomic radius (AR), electron affinity (ea), ionization potential (phi), melting point (m), density (rho), number of valence electrons (val), electronegativity (chi). And the elastic constants: C_{11}, C_{12}, C_{44}, and bulk modulus (K), shear modulus (G), and Young's modulus (E). For brevity, we refer average properties, e.g., $C_{11average}$ as C_{11} throughout the paper (unless specified). The target elastic constant C_{11m} is the first-principle calculated value for an alloy in the quinary alloy system Mo-W-Ta-V-Nb.

Our AFE with DQN strategies are implemented to search for a set of three descriptors with the maximum complexity c_{max} set to 5 for this regression problem. Specifically, we use the operation set $\{exp(\cdot), log(\cdot), (\cdot)^2, \sqrt{\cdot}, +, -, \times, \div\}$ and limit the upper bound of the runtime for each descriptor to be 6 hours for DQN and one-step greedy methods. Linear regression is adopted to predict C_{11_m} by engineered descriptors. The prediction performance is evaluated based on the R2 score and root-mean-square error (RMSE). For the training vs. test set split, we randomly separate the dataset with the ratio 7:3, resulting in 58 materials in the training set and 25 materials in the test set. Table 3.3 shows the best performing descriptors in 5 runs for each methods and Table 3.4 shows the average size of feature space and running time when DQN with different batch size attain the same maximum training R2 score in the time budget for the first time for scalability and efficiency comparison.

From Tables 3.3 and 3.4, all the feature engineering strategies can improve the prediction performance compared with the model with primary features. SISSO has the highest training R2 score with the much higher cost of significantly larger feature space and longer runtime to search for three final descriptors than our AFE strategies with DQN. On the other hand, without

Test Set	Methods	Descriptors	R2 Score RMSE Training	 Prediction
Random Split	Primary Features	$C_{11}, C_{12}, C_{44}, K, G, E, AR, CR,$ $ea, phi, m, rho, val, chi$	0.9736 13.923	0.9545 17.045
	SISSO	$(f^1)^* = \sqrt{(K \times ea)} - chi \log E$ $(f^2)^* = AR \times fi + G^2/\exp rho$ $(f^3)^* = (rho + \log ea)/m^2$	0.9870 9.774	0.9729 13.931
	One-step Greedy	$(f^1)^* = val - C_{44} \times AR \times CR/C_{12}$ $(f^2)^* = \sqrt{AR} - CR \times rho \times chi/C_{44})$ $(f^3)^* = fi + chi - (ea + E - C_{11})/G$	0.9821 11.458	0.9659 15.633
	DQN	$(f^1)^* = ea\sqrt{CR}/chi - ea/K$ $(f^2)^* = AR^2 + C_{12}/(fi \times AR \times rho)$ $(f^3)^* = E\sqrt{CR}/(C_{44}\sqrt{ea})$	0.9830 11.162	0.9755 13.258

Table 3.3: Alloy elastic behavior regression descriptors with prediction performances

Test Set	Average Performance	Methods SISSO	 DQN B=500	 B=1,000	 B=2,000	 B=4,000	 B=8,000
Random Split	Feature Space	4,806,301 ,260	675,318	931,356	1,010,109	919,548	799,882
	Runtime (hours)	28.4	6.3	8.2	9.1	7.9	5.6

Table 3.4: Scalability and efficiency comparison for alloy elastic behavior regression (Note: 'Feature Space' here refers to the total number of generated intermediate features until the final descriptors are found.)

Test Set	Methods	Descriptors	R2 Score RMSE Training	 Prediction
Random Split	Primary Features	$G_1, \ldots, G_{14}$	0.7070 78.303	0.5762 97.446
	DQN (Constrained method 1)	$(f^1)^* = (G_3(G_1 + G_5)/G_1)/G_{13}$ $(f^2)^* = (G_7^{\frac{1}{2}} + (G_2(G_1 - G_2)))$ $\times(G_5 + G_3^{-2})$	0.7301 75.159	0.5816 96.817
	DQN (Constrained method 2)	$(f^1)^* = G_3^2 + (G_1 + G_{10})(\sqrt{G_5} - G_{12})$ $(f^2)^* = G_8(G_2 - G_6 G_{13})$ $-G_6^2 G_{13} - G_4^2 - G_1^2$	0.7268 75.605	0.5472 100.724

Table 3.5: Descriptors derived by physics-constrained AFE and their performance (Note: Descriptors for constrained method 1 and 2 can be interpreted as $g_m(G_1, \ldots, G_{14}) = g_m(\sum_j (f_1^j), \ldots, \sum_j (f_{14}^j))$ and $g_m(G_1, \ldots, G_{14}) = \sum_j g_m(f_1^j, \ldots, f_{14}^j)$.)

considering potential long-term benefits when constructing intermediate features, one-step greedy cannot perform better than SISSO or DQN-based strategies. With small datasets, SISSO again shows the tendency of overfitting. Our AFE strategies with DQN attain similar training R2 scores as SISSO does but have the highest prediction R2 score with the derived descriptors, showing its potential to derive physics meaningful descriptors with better scalability and computational efficiency.

3.2.1 Regression with physics constraints

We also implement our AFE algorithm on a physics-constrained dataset, which consists of 14 elemental properties as primary features $\{G_1, G_2, \ldots, G_{14}\}$ for each of 40 possible constituting elements. Here $G_i = \{f_i^1, f_i^2, ..., f_i^{40}\}$ with f_i^j representing the ith property of the jth element. The elemental properties are weighted based on each element contribution for a given material. We apply our AFE strategies to derive sets of descriptors based on these 14×40 primary features and fit a regression model to predict the target of the transformation temperature A_f.

The reason for the constraints is that, in materials systems, rules must be followed to ensure fundamental science laws are not violated. A material can not have elemental contributions that sum to less than or greater than one. To reveal underlying physical meaning, the elemental properties can not be combined in a way that results in an element preference, such as leaving specific elements and the corresponding properties out of the model. These rules constrain the possible AFE to two methods. The first is to sum up all the features with the same property across the different elements, and then apply the AFE on them. The second is to implement AFE on features with the same element across different properties and then sum them up to perform the regression.

In both methods for the experiments, we choose the action space to be $\{exp(\cdot), log(\cdot), (\cdot)^2, (\cdot)^{-1}, \sqrt{\cdot}, +, -, \times, \div\}$. Due to the increasing number of primary features in this dataset compared to previous datasets, SISSO is not computationally feasible to identify predictive descriptors within the runtime budget. We report the results based on our AFE strategies with DQN. Specifically, we search for a set of two descriptors of maximum allowed complexity c_{max} at 15 with the runtime budget of 6 hours for each descriptor. We compute the R2 score and RMSE with linear regression to evaluate the prediction performances of engineered descriptors. For the training vs. test set split, we randomly separate the dataset with the ratio 8:2, resulting in 472 materials in the train set and 118 materials in the test set.

Table 3.5 shows the results from the two physics-constrained methods. We can see that the derived predictors by the first constrained AFE method achieves the best training and prediction R2 scores and RMSEs, demonstrating again that our AFE can help identify interpretable and predictive descriptors. We note that, although the second constrained AFE method can find the descriptors with better training R2 score than the model with only primary features, the testing R2 score is low. One reason is that the second strategy searches for descriptors considering algebraic operations on primary features from 40 possible elements. With the limited number of training samples, this is more prone to overfitting than the first constrained AFE method operating on the 14 summarized primary features (i.e. $\sum_j f_i^j$). We expect that the performance can be improved when more prior physics knowledge and constraints are available to further restrict the feasible feature space.

4. CONCLUSIONS

In this thesis, we present a physics-constrained AFE framework based on the feature generation tree exploration with Deep Q-Network for interpretable predictive modeling in materials science applications.

Compared to the recently developed AFE method—SISSO [1], which learns prediction models by first generating all the generated complex features and then using the sure independent screening method to select these "explanable" features for classification and regression, we explore reinforcement learning (RL) for more scalable and efficient AFE. The key difference of our AFE strategies lies in the way of engineering features. SISSO generates all the available features with the given complexity in the brute-force way and then finds the promising descriptors in the final prediction models, which requires generating an exploding number of complex features (double exponential with respect to the number of allowed operators for feature generation). Such a strategy is not only time consuming and computationally expensive, but also becomes computational prohibitive with the increasing maximum complexity of engineered features, leading to the exponential memory complexity. On the other hand, one-step greedy AFE methods can lose some promising features due to the non-monotonic relationship between features and prediction accuracy. The proposed AFE strategies with DQN-guided FGT exploration tackle these problems by approximating the expectation of the future reward of generating feature policy through DQN, and replacing the brute-force feature generation by exploring feature generation trees. Consequently, our reinforcement learning based AFE has better scalability and computational efficiency without sacrificing prediction performance due to better DQN approximation of future feature predictive power compared to one-step greedy methods.

The results of our real-world materials science experiments have demonstrated the potential of our AFE with RL-based tree exploration in reducing the runtime and enhancing the scalability for automatic feature engineering. More importantly, the engineered descriptors are interpretable with the corresponding lists of algebraic operations on the original primary features. Such a framework can be especially useful for scientific research, including materials science and biomedicine, where interpretable instead of “blackbox” machine learning can lead to new knowledge discovery and better decision making.

REFERENCES

[1] R. Ouyang, S. Curtarolo, E. Ahmetcik, M. Scheffler, and L. M. Ghiringhelli, "Sisso: A compressed-sensing method for identifying the best low-dimensional descriptor in an immensity of offered candidates," *Physical Review Materials*, vol. 2, no. 8, p. 083802, 2018.

[2] J. Brownlee, "Discover feature engineering, how to engineer features and how to get good at it," *Machine Learning Process*, 2014.

[3] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[4] J. Heaton, "An empirical analysis of feature engineering for predictive modeling," in *SoutheastCon 2016*, pp. 1–6, IEEE, 2016.

[5] U. Khurana, D. Turaga, H. Samulowitz, and S. Parthasrathy, "Cognito: Automated feature engineering for supervised learning," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp. 1304–1307, IEEE, 2016.

[6] A. Kaul, S. Maheshwary, and V. Pudi, "Autolearn—automated feature generation and selection," in *2017 IEEE International Conference on data mining (ICDM)*, pp. 217–226, IEEE, 2017.

[7] W. Long, Z. Lu, and L. Cui, "Deep learning-based feature engineering for stock price movement prediction," *Knowledge-Based Systems*, vol. 164, pp. 163–173, 2019.

[8] C. Fan, Y. Sun, Y. Zhao, M. Song, and J. Wang, "Deep learning-based feature engineering methods for improved building energy prediction," *Applied Energy*, vol. 240, pp. 35–45, 2019.

[9] L. M. Ghiringhelli, J. Vybiral, S. V. Levchenko, C. Draxl, and M. Scheffler, "Big data of materials science: critical role of the descriptor," *Physical review letters*, vol. 114, no. 10, p. 105503, 2015.

[10] L. M. Ghiringhelli, J. Vybiral, E. Ahmetcik, R. Ouyang, S. V. Levchenko, C. Draxl, and M. Scheffler, "Learning physical descriptors for materials science by compressed sensing," *New Journal of Physics*, vol. 19, no. 2, p. 023017, 2017.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[12] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pp. 1–10, IEEE, 2015.

[13] U. Khurana, H. Samulowitz, and D. Turaga, "Feature engineering for predictive modeling using reinforcement learning," in *AAAI Conference on Artificial Intelligence*, pp. 3407–3414, 2018.

[14] J. Zhang, J. Hao, F. Fogelman-Soulié, and Z. Wang, "Automatic feature engineering by deep reinforcement learning," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2312–2314, 2019.

[15] J. Fan and J. Lv, "Sure independence screening for ultrahigh dimensional feature space," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 5, pp. 849–911, 2008.

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