Jointly learning relevant subgraph patterns and nonlinear models of their indicators[*]

Ryo Shirakawa
Graduate School of Information Science and Technology,
Hokkaido University
sira@art.ist.hokudai.ac.jp

Yusei Yokoyama
Research & Development Group, Hitachi, Ltd
Graduate School of Information Science and Technology,
Hokkaido University
yusei.yokoyama.qk@hitachi.com

Fumiya Okazaki
Graduate School of Information Science and Technology,
Hokkaido University
fokazaki.w.h.i@gmail.com

Ichigaku Takigawa
Graduate School of Information Science and Technology,
Hokkaido University
PRESTO, Japan Science and Technology Agency (JST)
takigawa@ist.hokudai.ac.jp

ABSTRACT

Classification and regression in which the inputs are graphs of arbitrary size and shape have been paid attention in various fields such as computational chemistry and bioinformatics. Subgraph indicators are often used as the most fundamental features, but the number of possible subgraph patterns are intractably large due to the combinatorial explosion. For such a combinatorially large feature space, we propose a novel efficient algorithm to jointly learn relevant subgraph patterns and nonlinear models of their indicators. Previous methods for such joint learning of subgraph features and models are based on search for single best subgraph features with specific pruning and boosting procedures of adding their indicators one by one, which result in linear models of subgraph indicators. In contrast, the proposed approach is based on directly learning regression trees for graph inputs using a newly derived bound of the total sum of squares for data partitions by a given subgraph feature, and thus can learn nonlinear models through standard gradient boosting. An illustrative example we call the Graph-XOR problem to consider nonlinearity, numerical experiments with real datasets, and scalability comparisons to naïve approaches using explicit pattern enumeration are also presented.

KEYWORDS

Graph classification and regression, subgraph pattern mining, nonlinear supervised learning

1 INTRODUCTION

Graphs are fundamental data structures for representing combinatorial objects. However, precisely because of their combinatorial nature, it is usually difficult to understand the underlying trends in large datasets of graphs. The rapid increase in data in recent years also includes data represented as graphs, and thus supervised learning in which the inputs are graphs of arbitrary size and shape has gained considerable attention in diverse fields such as cheminformatics [11, 15, 22–24, 26, 27, 30, 31], bioinformatics [3, 10, 28] , computer vision [1, 2, 9, 19], and natural language processing [14].

The present paper investigates this problem of supervised learning with graphs of arbitrary size and shape as inputs. A typical example is QSAR (quantitative structure-activity relationship) prediction in virtual screening for drug discovery. The input graphs are molecular graphs representing the structural formulas of chemical compounds, and the output would be their activities such as binding affinities to a specific target, or properties such as ADMET (absorption, distribution, metabolism, excretion, toxicity).

In this setting, the most fundamental and widely used features are indicators of substructural features, i.e., subgraph patterns. In QSAR cases, the target activities or properties of chemical compounds are characterized by their substructural features such as the existence of specific functional groups or several fragmental patterns. However, we need to solve a difficult problem on how to appropriately pre-select or spot the relevant subgraph patterns to be incorporated as features for supervised learning. Since the number of possible subgraph patterns are intractably large due to the combinatorial explosion, this problem is mostly solved by heuristics and domain knowledge. But, supervised learning is now applied to very diverse targets and data, it has been demanded to develop a general and widely applicable approach to this problem of feature learning from such a combinatorially large feature space.

In the present paper, we first characterize supervised learning with all possible substructural indicators, and then develop a method for jointly learning the relevant subgraph patterns and *nonlinear* models of their indicators. The following are the contributions:

- We present two lesser-recognized facts to make sure the difference between linear and nonlinear models of substructural indicators as pseudo Boolean functions: (1) For a closely related problem of supervised learning from itemsets, the hypothesis space of the nonlinear model of all possible subitemset indicators is equivalent to that of the linear model; (2) Nevertheless, for the indicators of connected subgraphs, the hypothesis space of the nonlinear model is strictly larger than that of the linear model. **(Section 4)**
- We develop a novel efficient supervised learning algorithm for joint learning of all relevant subgraph features and a

[*] The present study was supported in part by JSPS/MEXT KAKENHI #17H01783, #17K19953, #15H05711 and JST PRESTO #JPMJPR15N9.

MLG'18, August 2018, London, United Kingdom
.

nonlinear models of their indicators. Unlike existing approaches based on ℓ_1-regularized linear models, the proposed algorithm is based on gradient tree boosting with base regression trees selecting each splitter out of all subgraph indicators with an efficient pruning based on the new bound in Theorem 5.1. **(Section 5)**

- We empirically demonstrate that (i) for the Graph-XOR dataset, the proposed nonlinear method actually outperforms several linear methods, which implies the existence of problems requiring nonlinear hypotheses, (ii) For several real datasets, we also observe similar superiority of the nonlinear models for some datasets, while it also turns out that the performance of linear models is fairly comparable for some datasets. **(Section 6)**

1.1 Related Research and Our Motivation

This line of research starts with the fact that practical *graph kernels* require a heuristically limited class of subgraphs [1–3, 9, 11, 15, 24, 31], and constricting *all-subgraphs* kernel is theoretically hard [6]. In order not to overlook any important features, joint learning of relevant subgraph patterns and classification/regression models by their indicators has been developed [14, 19, 22, 23, 27]. Although not discussed explicitly, these previous studies are based on ℓ_1-regularized boosting that yielded *linear* models of subgraph indicators because they used *decision stumps* with respect to a single subgraph feature as base learners.

The research of the present paper starts with our observation that replacing the decision stumps with decision trees in these existing methods is far from straightforward. This is because the previous methods are based on efficient pruning with specifically derived bounds to find a single best subgraph pattern, and use the indicator as a base learner at each iteration.

One naïve method to obtain nonlinear models of subgraph indicators is to enumerate some candidate subgraphs from training graphs, explicitly construct 0-1 indicator-feature vectors of test graphs by solving subgraph-isomorphism directly, and apply a general nonlinear supervised learning to those feature vectors. The performance with all small-size subgraphs occurred in the given graphs is known to be comparable for cheminformatics datasets[32]. However these approaches would not scale well as we see later in Section 6.3. Another good known heuristic idea is to use r-neighborhood subgraphs with radius r at each node as seen in ECFP [21] and graph convolutions [7, 13]. Unfortunately, the complete enumeration would not scale well either in this case, and usually requires some tricks such as feature hashing, feature folding, or feature embedding through neural nets, all of which are very interesting approaches but beyond the scope of this paper.

2 PRELIMINARIES

2.1 Notations

Let $[n]$ be $\{1, 2, \ldots, n\}$, and let $\mathbb{I}(P)$ denote the indicator of P, i.e., $\mathbb{I}(P) = 1$ if P is true, else 0. We denote as $G \sqsupseteq g$ the subgraph isomorphism that G contains a subgraph that is isomorphic to g and its negation as $G \not\sqsupseteq g$. Thus, a subgraph indicator $\mathbb{I}(G \sqsupseteq g) = 1$ if $G \sqsupseteq g$, otherwise 0. We also denote the training set of input graphs $G_i \in \mathcal{G}$ and output responses $y_i \in \mathcal{Y}$ as

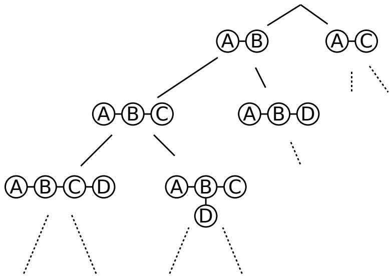

Figure 1: An enumeration tree

$$\mathcal{D} = \{(G_1, y_1), (G_2, y_2), \ldots, (G_N, y_N)\}, \quad (1)$$

where $\mathcal{G}$ is a set of all finite-size, connected, discretely-labeled, undirected graphs. We denote $\mathcal{G}_N = \{G_i \mid i \in [N]\}$, and the set of all possible connected subgraphs as $\mathcal{S}_N = \bigcup_{G \in \mathcal{G}_N} \{g \mid G \sqsupseteq g\}$.

2.2 Search Space for Subgraphs

In supervised learning from graphs, we represent each input graph $G_i \in \mathcal{G}_N$ by the characteristic vector $(\mathbb{I}(G_i \sqsupseteq g) \mid g \in \mathcal{S})$ with a set $\mathcal{S}$ of relevant subgraph features. However, since $\mathcal{S}$ is not explicitly available when the learning phase starts, we need to jointly search and construct $\mathcal{S}$ during the learning process. In order to define an efficient search space for $\mathcal{S}_N$, i.e., any subgraphs occurring in $\mathcal{G}_n$, the techniques for *frequent subgraph mining*, which enumerates all subgraphs that appear in more than m input graphs for a given m, are useful. Note that any subgraph feature $g \in \mathcal{S}_N$ can occur multiple times at multiple locations in a single graph, but $\mathbb{I}(G_i \sqsupseteq g) = 1$. We consider the indicators only for simplicity, but the number of occurrences can be taken into account as weighted indicators.

In the present paper, we use the search space of the gSpan algorithm [33], which performs a depth-first search on the tree-shaped search spaces on $\mathcal{S}_N$, referred to collectively as an *enumeration tree*, as shown in Figure 1. Each node of the enumeration tree holds a subgraph feature g' that extends the subgraph feature g at the parent node by one edge, namely, $g' \sqsupseteq g$. The following *anti-monotone* property of subgraph isomorphism over the enumeration tree on $\mathcal{S}_N$ can be used to derive the efficient search-space pruning of the gSpan algorithm:

$$G_i \not\sqsupseteq g \Rightarrow G_i \not\sqsupseteq g' \quad \text{for} \quad g' \sqsupseteq g. \quad (2)$$

2.3 Gradient Tree Boosting

Gradient tree boosting (GTB) [5, 16] is a general algorithm for supervised learning to predict a response y from a predictor $\boldsymbol{x}$. For a given hypothesis space $\mathcal{H}$, the goal is to minimize the expected risk $\mathbb{E}_{y,\boldsymbol{x}}\{\ell(y, f(\boldsymbol{x}))\}$ of $f \in \mathcal{H}$ by minimizing $N^{-1} \sum_{i \in [N]} \ell(y_i, f(\boldsymbol{x}_i))$ over the training data $\{(\boldsymbol{x}_i, y_i)\}_{i \in [N]}$ under some regularization.

GTB maintains an additive ensemble of k regression trees $T_t(\boldsymbol{x})$ of the form for fixed-stepsize cases:

$$f_k(\boldsymbol{x}) = T_0 + \eta \sum_{t \in [k]} T_t(\boldsymbol{x})$$

where T_0 is the mean of response variables in the training data, η is the stepsize, and $T_t(\boldsymbol{x})$ is the t-th regression tree as a base learner.

To fit the model to the training data, GTB performs the following gradient-descent-like iterations as a boosting procedure:

$$f_0(\boldsymbol{x}) \leftarrow \arg\min_c \sum_{i\in[N]} \ell(y_i, c),$$
$$\text{and} \quad f_t(\boldsymbol{x}) \leftarrow f_{t-1}(\boldsymbol{x}) + \eta\, T_t(\boldsymbol{x}),$$

where $T_t(\boldsymbol{x})$ is a regression tree to best approximate the values of negative functional gradient $-\nabla_f \ell(y, f(\boldsymbol{x}))$ at f_{t-1} obtained by fitting a regression tree to the data $\{(\boldsymbol{x}_i, r_i)\}_{i\in[N]}$, where

$$r_i = -\ell'(y_i, f_{t-1}(\boldsymbol{x}_i)) \text{ where } \ell'(y, \hat{y}) = -\frac{\partial\, \ell(y,\hat{y})}{\partial \hat{y}}.$$

Our experiments focus on binary classification tasks with $y \in \{-1, +1\}$, and thus we use the logistic loss $\ell(y, \hat{y}) = \log(1 + \exp(-2y\hat{y}))$. Note that even for classification, GTB must fit regression trees instead of classification trees in order to approximate real-valued functions.

The primary hyperparameters of GTB that we consider are the following three parameters: a max tree-depth d, a stepsize η, and the number of trees k.

2.4 Regression Trees

The internal regression-tree fitting for $r \approx T_t(\boldsymbol{x})$ is performed by the recursive partitioning below:

(1) Each node in the regression tree receives a subset $\mathcal{D}' \subseteq \mathcal{D}$ from the parent node.

(2) If a terminal condition is satisfied, the node becomes a leaf decision node with a prediction value by the average of the response values in $\mathcal{D}'$.

(3) Otherwise, the node becomes an internal node that tries to find the best partition of $\mathcal{D}'$ to $\mathcal{D}_1$ and $\mathcal{D}_0 = \mathcal{D}' \setminus \mathcal{D}_1$ that minimizes the total sum of squares for r:

$$\min_{\mathcal{D}_1, \mathcal{D}_0} \left[\mathrm{TSS}(\mathcal{D}_1) + \mathrm{TSS}(\mathcal{D}_0)\right].$$

The subsets $\mathcal{D}_1$ and $\mathcal{D}_0$ are further sent to the child nodes, and Step (1) is then recursively applied to each subset at the child nodes.

TSS here is the total sum of squares for r:

$$\mathrm{TSS}(\mathcal{D}) = \frac{1}{2} \sum_{i\in[N]} (r_i - \bar{r})^2, \;\; \bar{r} = \frac{1}{N} \sum_{i\in[N]} r_i. \qquad (3)$$

3 PROBLEM SETTING AND CHALLENGES

Our goal is learning a nonlinear model f over all possible subgraph indicators $\mathbb{I}(G \sqsupseteq g)$ for $g \in \mathcal{S}_N$. As we will see in Section 4, arbitrary functions of subgraph indicators have a unique multi-linear polynomial form

$$f(G) = \sum_{S\subseteq \mathcal{S}_N} c_S \prod_{g\in S} \mathbb{I}(G \sqsupseteq g).$$

Input graphs are implicitly represented as a bag of subgraph features, and hence as feature vectors in which the elements are an intractably large number of binary variables of each subgraph indicator. The main challenge is how to learn the relevant features g from such a combinatorially large space $\mathcal{S}_N$ with also jointly learning the classifier f over those features $\mathbb{I}(G \sqsupseteq g)$ for $g \in \mathcal{S}_N$. Other technical challenges are (1) feature vectors are binary valued and takes finite discrete values only at the vertices of a very high-dimensional Boolean hypercube; (2) feature vectors are strongly correlated due to subgraph isomorphism.

4 PSEUDO-BOOLEAN FUNCTIONS OF SUBSTRUCTURAL INDICATORS

We first investigate the difference between linear and nonlinear models in such a combinatorially large feature space. Any subgraph indicator $\mathbb{I}(G \sqsupseteq g)$ is a 0-1 Boolean variable, and thus the hypothesis space that we can consider with respect to these variables is a family of *pseudo-Boolean functions*, regardless of whether they are linear or nonlinear. A real-valued function $f : \{0,1\}^d \to \mathbb{R}$ on the Boolean hypercube $\{0,1\}^d$ is called pseudo-Boolean.

In this section, we explain the inequivalence of linear and nonlinear models of all possible subgraph indicators. Theorem 4.1, contrasting the difference from closely related problems for itemsets, suggests an advantage of the proposed nonlinear approach, and an illustrative example we call Graph-XOR that linear models cannot learn is presented in Section 6.1.

Theorem 4.1. *(1) The hypothesis space of the nonlinear model of all possible sub-itemset indicators is equivalent to that of the linear model. (2) The hypothesis space of the nonlinear model of all possible connected subgraph indicators is strictly larger than that of the linear model.*

This result is based on the following fundamental property of pseudo-Boolean functions.

Lemma 4.2. *[8, 20] Every pseudo-Boolean function $f : \{0,1\}^d \to \mathbb{R}$ has a unique multi-linear polynomial representation:*

$$f(x_1, \ldots, x_d) = \sum_{S\subseteq[d]} c_S \prod_{j\in S} x_j, \quad x_j \in \{0,1\},\; c_S \in \mathbb{R}.$$

4.1 Sub-Itemset Indicators

Let $x_j \in \{0,1\}$ be a Boolean variable defined by $x_j = \mathbb{I}(j \in I)$ for an item $j \in [d]$ in a itemset $I \subseteq [d]$. Then we can see that linear and nonlinear models of *sub-itemset indicators* $\mathbb{I}(S \subseteq I), S \subseteq [d]$ are equivalent as a hypothesis space on itemsets. For any function $f : 2^{[d]} \to \mathbb{R}$, we have

$$f(I) = \sum_{P\subseteq 2^{[d]}} c_P \prod_{S\in P} \mathbb{I}(S \subseteq I) = \sum_{U\in 2^{[d]}} c_U \mathbb{I}(U \subseteq I)$$
$$= \sum_{U\subseteq[d]} c_U \prod_{j\in U} \mathbb{I}(j \in I)$$

where $U = \bigcup_{S\in P} S$. Theorem 4.1 (1) follows from this simple fact. Note that including the negation terms, as in decision tree learning, does not change the hypothesis space because it can be represented as $\mathbb{I}(S \not\subseteq I) = 1 - \mathbb{I}(S \subseteq I)$.

4.2 Connected-Subgraph Indicators

As for subgraph indicators $\mathbb{I}(G \sqsupseteq g)$, the standard setting implicitly assumes that subgraph feature g is a *connected* graph. This would be primarily because the complete search for subgraph patterns, including disconnected graphs, is practically impossible, given that even a set of all connected graphs $\mathcal{S}_N$ in $\mathcal{G}_N$, is already intractably huge in practice.

The difference between linear model $f_L(G)$ and nonlinear model $f_{NL}(G)$ is not constantly zero:

$$\begin{aligned} f_L(G) &= c_0 + \sum_{g \in \mathcal{S}_N} c_g \mathbb{I}(G \sqsupseteq g) \\ f_{NL}(G) &= c_0 + \sum_{g \in \mathcal{S}_N} c_g \mathbb{I}(G \sqsupseteq g) \\ &\quad + \sum_{S \subseteq \mathcal{S}_N, |S| \geqslant 2} c_S \prod_{g \in S} \mathbb{I}(G \sqsupseteq g) \end{aligned}$$

from which Theorem 4.1 (2) follows. This also implies that if we consider all *connected-subgraph-'set' indicators* for the co-occurrence of several connected subgraphs, then the linear model is equivalent to the nonlinear model as a hypothesis space, and more importantly it is identical to $f_{NL}(G)$, which is the hypothesis space covered by the proposed algorithm in Section 5. Note that connected-subgraph-set indicators differ from the indicators of general subgraphs, including disconnected-subgraph-set indicators, because any subgraph feature $g \in \mathcal{S}$ can occur multiple times at different partially overlapped locations in a single graph. The hypothesis space by general subgraph indicators is beyond the scope of the present paper, and, in practice, the complete search for such indicators is computationally too challenging.

5 PROPOSED METHOD

In this section, we present a novel efficient method to produce a nonlinear prediction model based on gradient tree boosting with all possible subgraph indicators. Existing boosting-based methods [14, 19, 23] are based on simple but efficiently searchable base-learners of decision stumps (equivalent to subgraph indicators, as demonstrated previously) and construct an efficient pruning algorithm for this single best subgraph search at each iteration. In contrast, the proposed approach involves this subgraph search at finding an optimal split at each internal node of regression trees, while keeping the other outer loops the same as in GTB, as explained in Section 2.3. More specifically, we need to efficiently perform the following optimization over all possible subgraphs in $\mathcal{S}_N$:

$$\min_{g \in \mathcal{S}_N} \Big[\mathrm{TSS}(\mathcal{D}_1(g)) + \mathrm{TSS}(\mathcal{D}_0(g)) \Big] \tag{4}$$

where TSS is the total sum of squares defined as (3), $\mathcal{D}_1(g) = \{(G_i, r_i) \in \mathcal{D} \mid G \sqsupseteq g\}$ and $\mathcal{D}_0(g) = \{(G_i, r_i) \in \mathcal{D} \mid G \not\sqsupseteq g\}$.

Here, $|\mathcal{S}_N|$ is too intractably huge to solve (4) by exhaustively testing subgraph $g \in \mathcal{S}_N$ in order, and thus we perform a branch and bound search over the enumerate tree on $\mathcal{S}_N$ with the following lower bound for the total sum of squares of expanded subgraphs:

Theorem 5.1. *Given $\mathcal{D}_1(g)$ and $\mathcal{D}_0(g)$, for any subgraph $g' \sqsupseteq g$,*

$$\mathrm{TSS}(\mathcal{D}_1(g')) + \mathrm{TSS}(\mathcal{D}_0(g')) \geq$$
$$\min_{(\diamond, k)} \Big[\mathrm{TSS}(\mathcal{D}_1(g) \setminus S_{\diamond,k}) + \mathrm{TSS}(\mathcal{D}_0(g) \cup S_{\diamond,k}) \Big] \tag{5}$$

where $(\diamond, k) \in \{\leq, >\} \times \{2, \ldots, |\mathcal{D}_1(g) - 1|\}$, and $S_{\diamond,k} \subset \mathcal{D}_1(g)$, such that $S_{\leq,k}$ is a set of k pair (G_i, r_i) selected from $\mathcal{D}_1(g)$ in descending order of r_i, and $S_{>,k}$ is that in increasing order. Note that $\setminus$, $\cup$ are set difference and set union respectively.

Alg. 1: Gradient Tree Boosting for Graphs

```
Input: Training data
    $\mathcal{D} = \{(G_1, y_1), (G_2, y_2), \ldots, (G_N, y_N)\}$, and stepsize
    $\eta$
Output: Prediction model $f : G \to \mathcal{Y}$
Function GradientTreeBoosting ($\mathcal{D}$)
    $f \leftarrow 1/N \sum_{i=1}^{N} y_i$ ;
    for $k$ = 1, 2, ... do
        for $i \leftarrow 1$ to $N$ do
            $r_i^{(k)} \leftarrow -\left[\frac{\partial \ell(y_i, T(G_i))}{\partial T(G_i)}\right]_{T(G)=T_{k-1}(G_i)}$
        end
        $T_k \leftarrow$ BuildRegressionTree
          ($\{(G_i, r_i^{(k)}) \mid i \in [N]\}$) ;                ▷ Alg. 2
        $f \leftarrow f + \eta T_k$ ;
    end
    return $f$
```

Alg. 2: Regression Tree Learning for Graphs

```
Input: Training data $\mathcal{D} = \{(G_1, r_1), (G_2, r_2), \ldots, (G_N, r_N)\}$
Output: Regression tree $T$
Function BuildRegressionTree ($\mathcal{D}$)
    if the terminal condition is satisfied then
        make a leaf node in $T$ with the mean of $r_i$ ;
    else
        $g \leftarrow$ FindBestSplit ($\mathcal{D}$) ;                ▷ Alg. 3
        make an internal node $v$ in $T$ with $g$ ;
        the left child of $v \leftarrow$ BuildRegressionTree
          ($\mathcal{D}_1(g)$) ;
        the right child of $v \leftarrow$ BuildRegressionTree
          ($\mathcal{D}_0(g)$) ;
    end
    return $T$
```

Proof. The result follows from the property (2). See Appendix A for details. □

The entire procedure of the proposed algorithm is illustrated in Alg. 1 and 2. The novel algorithm for the optimal subgraph search of (4) using the bound (5) is described in detail in Alg. 3. In order to solve (4), the proposed algorithm uses a depth-first search on the enumerate tree over $\mathcal{S}_N$. The procedure at each subgraph g is as follows:

1. Calculate the total sum of squares tss $\leftarrow$ $\mathrm{TSS}(\mathcal{D}_1(g) + \mathcal{D}_0(g))$ of subgraph g.
2. Update min_tss by tss if min_tss > tss.
3. Calculate bound (5) and if min_tss < bound, then prune all child nodes of g.

In the entire procedure, the most time-consuming part is the subgraph search (Alg. 3), which is repeatedly called during the learning process. Hence, introducing memorization, whereby we store expensive calls and return the cached results when the same pattern occurs again, can considerably speed up the entire process. First, we can store the result of minimization $\min_{g \in \mathcal{S}_N}(\mathrm{TSS}(\mathcal{D}_1(g))+$

```
Alg. 3: Optimal Subgraph Search for Best Split
  Input: Training data $\mathcal{D} = \{(G_1, r_1), (G_2, r_2), \ldots, (G_N, r_N)\}$
  Output: Minimizer subgraph $g^*$ of (4)
  Function FindBestSplit ($\mathcal{D}$)
      repeat
          $g \leftarrow$ the next node of the enumeration tree by DFS ;
          tss $\leftarrow$ TSS($\mathcal{D}_1(g)$) + TSS($\mathcal{D}_0(g)$) ;
          if tss < min_tss then
              min_tss $\leftarrow$ tss;
              $g^* \leftarrow g$;
          end
          bound $\leftarrow$
            $\min_{(\diamond, k)}[\mathrm{TSS}(\mathcal{D}_1(g) \setminus S_{\diamond,k}) + \mathrm{TSS}(\mathcal{D}_0(g) \cup S_{\diamond,k})]$ ;
            ▷ Theorem 5.1
          if min_tss < bound then
              prune all children of $g$ in the enumeration tree ;
          end
      until the enumeration tree search ends;
      return $g^*$ ;
```

TSS($\mathcal{D}_0(g)$)) for each already checked g and $\mathcal{D}$. Second, the subgraph search can be entirely skipped until we need to check any subgraph that has not been checked in any previous iterations.

6 NUMERICAL EXPERIMENTS

6.1 The Graph-XOR Problem

The linear separability has long been discussed using the XOR (or parity in general) example, and we present the same key example for graphs, referred to as *Graph-XOR*, where linear models cannot learn the target rule even when noiseless examples are provided.

The Graph-XOR dataset includes 1,035 graphs of seven nodes and six edges, where 506 are positives with $y = +1$ and 529 negatives with $y = -1$. As illustrated in Figure 2, each graph is generated by connecting two subgraphs by one node Ⓓ. The component subgraphs are selected from the 18 types shown in Figure 3, where all three-node path graphs with candidate nodes {Ⓐ, Ⓑ, Ⓒ}, and are randomly classified into two groups. Note that Ⓐ–Ⓑ–Ⓒ is isomorphic to Ⓒ–Ⓑ–Ⓐ and this duplicate redundancy due to the graph isomorphism is removed. The response value y of a graph is -1 if two subgraphs are selected from the same group, otherwise $+1$.

Table 2 shows the performance results for the Graph-XOR data by two-fold cross validations. We use the proposed nonlinear method and two linear methods, namely, the proposed algorithm with maximum tree-depth (d) = 1, i.e., with decision stumps, and a state-of-the-art (but linear) method for graphs, gBoost [23]. The hyperparameter tuning is performed for the ranges described in Table 1, and the best parameters are also listed in Table 2. Figures 4 and 5 show the accuracy and loss changes for the test data with regard to the max tree depth (d) and the max subgraph size (x). Here "subgraph size" means the number of edges.

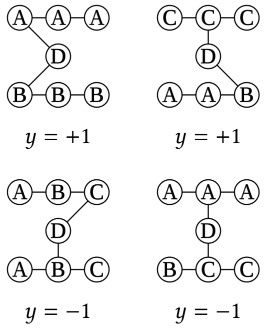

Figure 2: Examples of the Graph-XOR data

Group 1	Group 2
Ⓐ–Ⓐ–Ⓐ	Ⓑ–Ⓑ–Ⓑ
Ⓒ–Ⓒ–Ⓒ	Ⓐ–Ⓐ–Ⓑ
Ⓐ–Ⓑ–Ⓑ	Ⓐ–Ⓑ–Ⓐ
Ⓑ–Ⓐ–Ⓑ	Ⓑ–Ⓑ–Ⓒ
Ⓑ–Ⓒ–Ⓒ	Ⓑ–Ⓒ–Ⓑ
Ⓒ–Ⓑ–Ⓒ	Ⓐ–Ⓐ–Ⓒ
Ⓐ–Ⓒ–Ⓒ	Ⓐ–Ⓒ–Ⓐ
Ⓒ–Ⓐ–Ⓒ	Ⓐ–Ⓑ–Ⓒ
Ⓐ–Ⓒ–Ⓑ	Ⓑ–Ⓐ–Ⓒ

Figure 3: Subgraph groups

Table 1: Hyperparameter settings for Graph-XOR

Common			
max subgraph size (edges)		x	2, 3, 4, 5, ∞
Model specific			
Proposed	max tree depth	d	1, 2, 3, 4, 5
	stepsize	η	1, 0.7, 0.4, 0.1, 0.01
	# trees	k	1-500
gBoost	regularization	ν	0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.01

Table 2: Prediction accuracy (%) for the Graph-XOR

Nonlinear models	*Linear models*	
Proposed	Proposed ($d1$)	gBoost
100.0	64.3	70.0
$x2$ $d2$ $\eta0.7$ $k221$	$x6$ $d1$ $\eta0.7$ $k26$	$x6$ $\nu0.01$

The results shown in Table 2 clearly demonstrate that the linear models, including our model with $d = 1$, fail, but the nonlinear methods work well. This is also theoretically supported by Theorem 4.1. Figure 4 also shows that only the behavior of $d = 1$ differ from those of the other depths. Moreover, note that this problem at least requires subgraph features of size 2 (i.e., two edges), but searching excessively large subgraphs results in overfitting, as we see for $x \geqslant 4$ in Figure 5. The dataset is generated by subgraphs of size 3 in Figure 3, and search for larger subgraphs results in overspecification to memorize the training dataset. Good regularization to tolerate misspecification for x would be a future problem.

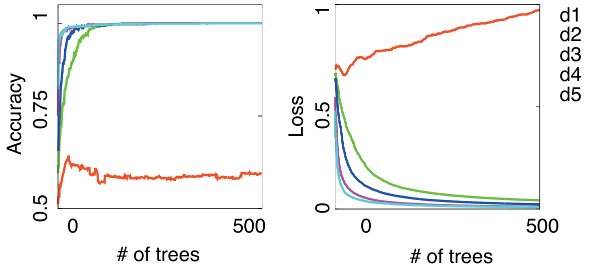

Figure 4: Test accuracy and loss with tree depth d

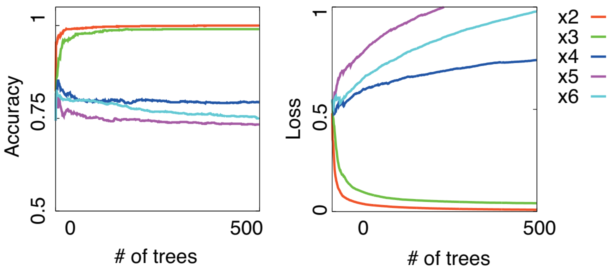

Figure 5: Test accuracy and loss with subgraph size x

6.2 QSAR with Molecular Graphs

We also evaluate the performance based on the most typical benchmark for graph classification on real datasets: the quantitative structure-activity relationship (QSAR) prediction with molecular graphs. We select four binary-classification datasets (CPDB, Mutag, NCI1, NCI47) in Table 3: two data (CPDB, Mutag) for mutagenicity tests and two data (NCI1, NCI47) for tumor growth inhibition tests from PubChem BioAssay[1]. NCI1 and NCI47 are balanced by randomly sampling negatives of the same size as the positives in order to avoid imbalance difficulty in evaluation. All chemical structures are encoded as molecular graphs using RDKit[2], and some structures in the raw data are removed by chemical sanitization[3]. We simply apply a node labeling by the RDKit default atom invariants (edges not labeled), i.e., atom type, # of non-H neighbors, # of Hs, charge, isotope, and inRing properties. These default atom invariants use connectivity information similar to that used for the well-known ECFP family of fingerprints [21]. See [13] for more elaborate encodings.

Tables 5 and 6 show the performance results obtained by 10-fold cross validations using the same three methods used for the Graph-XOR cases with different hyperparameter settings in Table 4. We can observe that nonlinear methods often outperform the linear methods. At the same time, we can also observe, in some cases, that the linear methods work fairly well for the real datasets. The real datasets would not have explicit classification rules compared to noiseless problems such as the Graph-XOR cases. Thus, it is necessary to tolerate some noises and ambiguity. Although they may seem limited, linear hypothesis classes are known to be very powerful in such cases, because they are quite stable estimators and the input features can themselves include nonlinear features of data as implied in Theorem 4.1. It would be an interesting open question how to determine which to use, linear or nonlinear, from data.

We also provide the normalized feature importance scores from GTB and the search space size in Figure 6 for the CPDB dataset. In Figure 6, *searched* corresponds to the searched subgraphs, and *selected* to the subgraph selected as internal nodes. This would also implies that (i) the proposed approach can provide information on selected relevant subgraph features and (ii) searches and uses only a portion of the entire search space.

Table 3: Dataset summary

Dataset	Graph- XOR	CPDB	Mutag	NCI1	NCI47
# data	1035	600	187	4252	4202
# nodes	7	13.7	17.9	26.3	26.3
# edges	6	14.2	19.7	28.4	28.4

# of nodes and edges are average.

Table 4: Hyperparameter settings for the QSAR

Common			
max subgraph size (# edges)		x	4, 6, 8
Model specific			
Proposed	max tree depth	d	1, 3, 5
	stepsize	η	1.0, 0.7, 0.4, 0.1
	# trees	k	1-500
gBoost	regularization	ν	0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.01

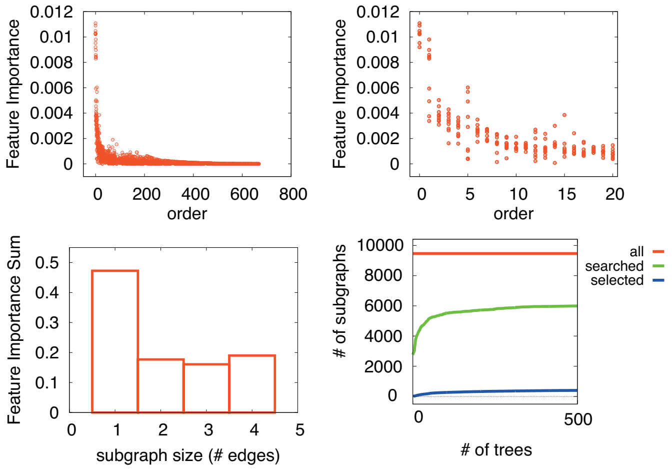

Figure 6: Feature importance and search space for CPDB

6.3 Scalability comparison to Naïve approach

As previously mentioned in Section 1.1, there exists a simple naïve two-step approach to obtain nonlinear models of subgraph indicators. Figure 7 shows the scalability of this “enumerate & learn” approach by first enumerating all small-size subgraphs and applying general supervised learning to their indicators. The values in the figure are the average values to process each fold in 10-fold cross validation on a single PC with Pentium G4560 3.50GHz and 8GB memory. We enumerate all subgraphs with limited subgraph size[4], and feed their indicator features to GradientBoostingClassifier with 100 trees (depth $\leqslant$ 5) of scikit-learn[5]. The proposed method is also tested with the same setting (100 trees, d5). Since

[1] https://pubchem.ncbi.nlm.nih.gov/bioassay/⟨AID⟩ (⟨AID⟩ = 1, 47)

[2] http://www.rdkit.org/

[3] Due to this pre-processing, the number of datasets differs from that in the simple molecular graphs in the literature, where the nodes are labeled by atom type, and the edges are labeled by bond type.

[4] Small-size subgraphs are known to be more appropriate for this supervised-learning purpose than frequent subgraphs [32].

[5] http://scikit-learn.org

Table 5: Prediction accuracy (%) for the QSAR

	CPDB		Mutag		NCI1		NCI47	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Nonlinear models								
Proposed	**79.3** (±4.8)	84.5 (±3.6)	87.8 (±6.6)	91.6 (±6.3)	**84.7** (±1.7)	**90.8** (±1.3)	**84.5** (±1.7)	**90.3** (±1.1)
Linear models								
Proposed (d1)	**79.3** (±4.4)	83.9 (±3.3)	87.8 (±6.6)	91.6 (±6.3)	83.1 (±1.6)	89.8 (±1.3)	82.8 (±1.4)	88.9 (±1.1)
gBoost	77.1 (±2.7)	73.6 (±4.9)	**91.4** (±5.8)	**93.9** (±5.0)	82.7 (±2.2)	83.9 (±2.2)	81.3 (±1.4)	81.8 (±2.6)
Reported values in literature								
L1-LogReg [27]	78.3	-	-	-	-	-	-	-
MGK [23]	76.5	75.6	80.8	90.1	-	-	-	-
freqSVM [23]	77.8	84.5	80.8	90.6	-	-	-	-
gBoost [23]	78.8	**85.4**	85.2	92.6	-	-	-	-
WL shortest path [24]	-	-	83.7	-	84.5	-	-	-
Random walk [24]	-	-	80.7	-	64.3	-	-	-
Shortest path [24]	-	-	87.2	-	73.4	-	-	-

Table 6: Best hyperparameters

	CPDB	Mutag	NCI1	NCI47
Proposed	$x4\ d5\ \eta 0.1\ k120$	$x4\ d1\ \eta 1\ k22$	$x4\ d5\ \eta 0.1\ k452$	$x4\ d3\ \eta 0.4\ k308$
Proposed(d1)	$x8\ d1\ \eta 0.4\ k128$	$x4\ d1\ \eta 1\ k22$	$x4\ d1\ \eta 0.4\ k499$	$x4\ d1\ \eta 0.4\ k499$
gBoost	$x8\ \nu 0.5$	$x7\ \nu 0.1$	$x8\ \nu 0.3$	$x8\ \nu 0.4$

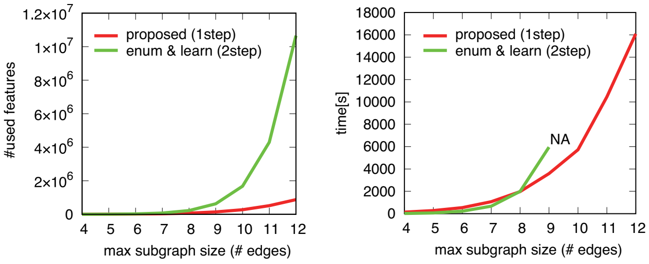

Figure 7: Scalability comparison to naïve approach

this case both use GTB and thus the performance is the same in principle up to implementation details (empirically both 0.75-0.77 for this setting), we focus on scalability comparisons using the fixed hyperparameters. Because the number of subgraph patterns to be enumerated increases exponentially, off-the-shelf packages such as scikit-learn cannot handle them at some point even when pattern enumeration can be done. In Figure 7, we can observe pattern enumeration can be done for max subgraph size = 1 to 12 (green line, left), but the 2nd scikit-learn step fails for max subgraph size ⩾ 10 (green line, right). In this CPDB examples, the numbers of subgraphs, i.e., the dimensions of feature vectors, were 66336.1, 145903.7, 275422.3, 512904.1, 874540.0 for max subgraph size = 8, 9, 10, 11, 12, respectively, and scikit-learn was only feasible for max subgraph size up to 9. Note that we also need to solve a large number of subgraph isomorphism known to be NP-complete.

7 CONCLUSIONS

In summary, we investigated nonlinear models with all possible subgraph indicators and provide a novel efficient algorithm to learn from the nonlinear hypothesis space. We demonstrated that this hypothesis space is identical to the (pseudo-Boolean) functions of these subgraph indicators, which are, in general, strictly larger than those of the linear models. This is also empirically confirmed through our Graph-XOR example. Although most existing studies focus only on real datasets, this would also promote interest in whether graph-theoretic classification problems can be approximated in a supervised learning manner. At the same time, the experimental results of the present study also strongly suggest that we need a nonlinear hypothesis space for the QSAR problems based on some real datasets, which would also support a standard cheminformatics approach of applying nonlinear models, such as random forests and neural networks, to 0-1 feature vectors by the existence of substructural features, referred to as *molecular fingerprints* such as ECFP [21].

Since research on classification and regression trees originates from the problem of *automatic interaction detection* [4, 12, 17], our approach can provide insights on the question whether such higher-order interactions between input features exist. In this sense, our methods and findings would also be informative to consider a recent hot topic of detecting such interactions in combinatorial data [18, 25, 29].

REFERENCES

[1] Lu Bai, Luca Rossi, Horst Bunke, and Edwin R. Hancock. 2014. Attributed Graph Kernels Using the Jensen-Tsallis q-Differences. In *Proceedings of the 2014 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2014)*. Nancy, France, 99–114.

[2] Vincent Barra and Silvia Biasotti. 2013. 3D shape retrieval using Kernels on Extended Reeb Graphs. *Pattern Recognition* 46, 11 (2013), 2985–2999.

[3] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. 2005. Protein Function Prediction via Graph Kernels. *Bioinformatics* 21, 1 (2005), i47–i56.

[4] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles. J. Stone. 1984. *Classification and Regression Trees.* Chapman and Hall/CRC, Belmont, California, U.S.A.

[5] Jerome H Friedman. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of statistics* (2001), 1189–1232.

[6] Thomas Gärtner, Peter A. Flach, and Stefan Wrobel. 2003. On Graph Kernels: Hardness Results and Efficient Alternatives. In *Proceedings of the 16th Annual Conference on Computational Learning Theory (COLT) and 7th Kernel Workshop.* 129–143.

[7] Justin Gilmer, Samuel Schoenholz, Patrick F. Riley, Oriol Vinyals, and George Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*. Washington, DC, USA.

[8] Peter L Hammer and Sergiu Rudeanu. 1968. *Boolean Methods in Operations Research and Related Areas.* Springer.

[9] Zaïd Harchaoui and Francis Bach. 2007. Image Classification with Segmentation Graph Kernels. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Minneapolis, Minnesota, USA, 1–8.

[10] Yan Karklin, Richard F. Meraz, and Stephen R. Holbrook. 2005. Classification of non-coding RNA using graph representations of secondary structure. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*. Hawaii, USA, 4–15.

[11] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. 2003. Marginalized Kernels Between Labeled Graphs. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*. Washington, DC, USA, 321–328.

[12] Gordon V. Kass. 1975. Significance Testing in Automatic Interaction Detection (A.I.D.). *Journal of the Royal Statistical Society. Series C* 24, 2 (1975), 178–189.

[13] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. 2016. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design* 30, 8 (01 Aug 2016), 595–608.

[14] Taku Kudo, Eisaku Maeda, and Yuji Matsumoto. 2005. An Application of Boosting to Graph Classification. In *Advances in Neural Information Processing Systems 17*, Lawrence K. Saul, Yair Weiss, and Léon Bottou (Eds.). MIT Press, Cambridge, MA, 729–736.

[15] Pierre Mahé and Jean-Philippe Vert. 2009. Graph kernels based on tree patterns for molecules. *Machine Learning* 75, 1 (2009), 3–35.

[16] Llew Mason, Jonathan Baxter, Peter L Bartlett, and Marcus R Frean. 2000. Boosting Algorithms as Gradient Descent. In *Advances in neural information processing systems.* 512–518.

[17] James N. Morgan and John A. Sonquist. 1963. Problems in the analysis of survey data, and a proposal. *J. Amer. Statist. Assoc.* 58, 302 (1963), 415–434.

[18] Kazuya Nakagawa, Shinya Suzumura, Masayuki Karasuyama, Koji Tsuda, and Ichiro Takeuchi. 2015. Safe Feature Pruning for Sparse High-Order Interaction Models. *arXiv preprint arXiv:1506.08002* (2015).

[19] Sebastian Nowozin, Koji Tsuda, Takeaki Uno, Taku Kudo, and Gökhan Bakır. 2007. Weighted substructure mining for image analysis. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Minneapolis, Minnesota, USA, 1–8.

[20] S. Rudeanu P.L. Hammer, I. Rosenberg. 1963. On the determination of the minima of pseudo-Boolean functions. *Studii ¸si cercetari matematice* 14 (1963), 359–364. in Romanian.

[21] David Rogers and Mathew Hahn. 2010. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling* 50, 5 (2010), 742–754.

[22] Hiroto Saigo, Nicole Krämer, and Koji Tsuda. 2008. Partial least squares regression for graph mining. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining.* Las Vegas, Nevada, USA, 578–586.

[23] Hiroto Saigo, Sebastian Nowozin, Tadashi Kadowaki, Taku Kudo, and Koji Tsuda. 2009. gBoost: a mathematical programming approach to graph classification and regression. *Machine Learning* 75 (2009), 69–89. Issue 1.

[24] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research* 12, Sep (2011), 2539–2561.

[25] Mahito Sugiyama, Felipe Llinares López, Niklas Kasenburg, and Karsten M Borgwardt. 2015. Significant Subgraph Mining with Multiple Testing Correction. In *Proceedings of the 2015 SIAM International Conference on Data Mining.* SIAM, 37–45.

[26] Ichigaku Takigawa and Hiroshi Mamitsuka. 2013. Graph mining: procedure, application to drug discovery and recent advances. *Drug Discovery Today* 18, 1-2 (2013), 50–57.

[27] Ichigaku Takigawa and Hiroshi Mamitsuka. 2017. Generalized Sparse Learning of Linear Models Over the Complete Subgraph Feature Set. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 3 (2017), 617–624.

[28] Ichigaku Takigawa, Koji Tsuda, and Hiroshi Mamitsuka. 2011. Mining Significant Substructure Pairs for Interpreting Polypharmacology in Drug-Target Network. *PLoS ONE* 6, 2 (2011), e16999. https://doi.org/10.1371/journal.pone.0016999

[29] Aika Terada, Mariko Okada-Hatakeyama, Koji Tsuda, and Jun Sese. 2013. Statistical significance of combinatorial regulations. *Proceedings of the National Academy of Sciences* 110, 32 (2013), 12996–13001.

[30] Koji Tsuda. 2007. Entire regularization paths for graph data. In *Proceedings of the 24th International Conference on Machine learning (ICML)*. Banff, Alberta, Canada, 919–926.

[31] S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. 2010. Graph Kernels. *Journal of Machine Learning Research* 11 (2010), 1201–1242.

[32] Nikil Wale, Ian A. Watson, and George Karypis. 2008. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems* 14, 3 (2008), 347–375.

[33] Xifeng Yan and Jiawei Han. 2002. gSpan: Graph-Based Substructure Pattern Mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM)*. Washington, DC, USA, 721–724.

A PROOF OF THEOREM 5.1

Proof. Given $\mathcal{D}_1(g)$ and $\mathcal{D}_0(g)$,

$$\begin{aligned}
\text{bound} &= \min_{g'}\Big[\mathrm{TSS}(\mathcal{D}_1(g')) + \mathrm{TSS}(\mathcal{D}_0(g'))\Big] \\
&= \min_{S\,\subset\,\mathcal{D}_1(g)}\Big[\mathrm{TSS}(\mathcal{D}_1(g)\setminus S) + \mathrm{TSS}(\mathcal{D}_0(g)\cup S)\Big] && (6) \\
&= \min_{(\diamond,\,k)}\Big[\mathrm{TSS}(\mathcal{D}_1(g)\setminus S_{\diamond,k}) + \mathrm{TSS}(\mathcal{D}_0(g)\cup S_{\diamond,k})\Big] && (7)
\end{aligned}$$

where $(\diamond, k) \in \{\leq, >\}\times\{2, \ldots, |\mathcal{D}_1(g) - 1|\}$. From the anti-monotone property (2), we have $\mathcal{D}_1(g') \subseteq \mathcal{D}_1(g)$ for $g' \sqsupseteq g$ for the training set $\mathcal{D}$ from which the equation (6) directly follows. Thus, we show (7) in detail. For simplicity, let $A =, \{a_1, \ldots, a_n \mid a_i \in \mathbb{R}\}$ denote $\mathcal{D}_1(g)$, and $B = \{b_1, \ldots, b_m \mid b_i \in \mathbb{R}\}$ denote $\mathcal{D}_0(g)$. Then, the goal of (6) is to minimize the total sum of squares $\mathrm{TSS}(A\setminus S) + \mathrm{TSS}(B\cup S)$ by tweaking $S = \{s_1, \ldots, s_k\} \subset A$. Let $\bar{a}$, $\bar{a}_{-S}$, $\bar{b}$, and $\bar{b}_{+S}$ be the means of A, $A \setminus S$, B, and $B \cup S$, respectively. The key fact is that $\mathrm{TSS}(A \setminus S) + \mathrm{TSS}(B \cup S)$ can be regarded as a quadratic equation of $\sum_{i=1}^{k} s_i$ when the size of S is fixed to k. More precisely,

$$\begin{aligned}
&\mathrm{TSS}(A\setminus S) + \mathrm{TSS}(B\cup S) \\
&= \sum_{i\in[n]}(a_i - \bar{a}_{-S})^2 - \sum_{i\in[k]}(s_i - \bar{a}_{-S})^2 + \sum_{i\in[m]}(b_i - \bar{a}_{+S})^2 + \sum_{i\in[k]}(s_i - \bar{a}_{+S})^2 \\
&= -\sum_{i\in[k]}(s_i - \bar{a})^2 - \frac{\big(\sum_{i\in[k]}(s_i - \bar{a})\big)^2}{n-k} + \sum_{i\in[n]}(a_i - \bar{a})^2 \\
&\qquad + \sum_{i\in[k]}(s_i - \bar{b})^2 - \frac{\big(\sum_{i\in[k]}(s_i - \bar{b})\big)^2}{m+k} + \sum_{i\in[m]}(b_i - \bar{b})^2 \\
&= -\left(\frac{1}{n-k} + \frac{1}{m+k}\right)\Big(\sum_{i\in[k]} s_i\Big)^2 + \left(2\bar{a}\frac{n}{n-k} - 2\bar{b}\frac{m}{m+k}\right)\sum_{i\in[k]} s_i \\
&\qquad - \frac{nk}{n-k}\bar{a}^2 + \frac{mk}{m+k}\bar{b}^2 + \sum_{i\in[n]}(a_i - \bar{a})^2 + \sum_{i\in[m]}(b_i - \bar{b})^2
\end{aligned}$$

Therefore, $\mathrm{TSS}(A \setminus S) + \mathrm{TSS}(B \cup S)$ is minimized when $\sum_{i=1}^{k} s_i$ is maximized or minimized. In other words, (6) becomes minimum when the mean of $S \subset \mathcal{D}_1(g)$ is maximized or minimized. □