Task Sensitive Feature Exploration and Learning for Multi-Task Graph Classification

Shirui Pan, Jia Wu, Xingquan Zhu, Guodong Long, Chengqi Zhang

Abstract

Multi-task learning (MTL) is commonly used for jointly optimizing multiple tasks for learning. To date, all existing MTL methods are designed for tasks with feature-vector represented instances, but cannot be applied to structured data, such as graphs. More importantly, when carrying out MTL, existing methods mainly focus on exploring overall commonality or disparity between tasks for learning, but cannot explicitly capture task relationships in the feature space, so they are unable to answer some fundamental questions, such as what exactly is shared between tasks and what is the uniqueness of one task differentiating from others?

In this paper, we formulate a new multi-task graph learning problem, and propose a task sensitive feature exploration and learning algorithm for multi-task graph classification. As graphs do not have features available, we advocate a task sensitive feature exploration and learning paradigm to jointly discover discriminative subgraph features across different tasks. In addition, a learning process in the feature space is carried out to categorize each subgraph feature into three categories: common feature, task auxiliary feature, and task specific feature, indicating whether the feature is shared by all tasks, by a subset of tasks, or by only one specific task, respectively. The learned features and multiple tasks are iteratively co-regularized to form a multi-task graph classification model with a global optimization goal. Experiments on real-world functional brain analysis and chemical compound categorization demonstrate the algorithm's performance. Results confirm that our method can be used to explicitly capture task correlations and uniqueness in the feature space, and explicitly answer what are shared between tasks and what is the uniqueness of a specific task.

Index Terms

Graph Classification, Subgraph Selection, Feature Selection, Multi-task Learning, Supervised Learning

1 Introduction

Graph classification is becoming increasingly important in recent years due to rapid growth of complex data which exhibit structural and interdependent relationships. Examples of graph applications range from chemical compound categorization [1], functional brain analysis [2], [3], to malware detection [4], and biomedical document classification [5].

The key challenge of graph classification lies in the fact that no feature is readily available for learning algorithms to derive classification model. This challenge has motivated numerous methods to represent graphs in a suitable format for learning, including (1) Kernel-based algorithms [6], [7], which learn effective kernels to measure similarity between graph objects, so that the pair-wise similarity matrix can be fed to learning algorithms such as a Support Vector Machine (SVM) for learning; (2) Subgraph-based algorithms [8], [9], [10], [11], [12], which aim to discover discriminative subgraph features to represent graph objects into vector space, so that generic machine learning algorithms can be applied.

To date, graph classification has been advanced to many complicated settings, such as multi-label classification [12], multi-graph learning [13], [14], [15], semi-supervised learning [11], [16], imbalanced learning [17], [18], and cost-sensitive learning [19], but all these methods are designed to handle single learning tasks. In reality, several relevant graph learning tasks may co-exist and each has a rather limited number of training graphs. Two motivation examples are given as follows:

Functional Brain Analysis aims to map human brain as a network (or a graph) to model relationships between diseases and functions of brain regions [20]. In order to carry out a specific learning task, such as diagnosing Attention Deficit Hyperactivity Disorder (ADHD) [21], each object has to go through functional magnetic resonance imaging (fMRI) and intensive data preprocessing to collect training data. This severely limits each task to maximum of only a couple of hundred objects. One the other hand, institutions may have data collected for different but relevant learning tasks, such as Gender [22] or Alzheimer's disease study. The limited samples for each individual tasks, and the commonality between

S. Pan, J. Wu, G. Long, and C. Zhang are with the Centre for Quantum Computation & Intelligent Systems, FEIT, University of Technology Sydney, NSW 2007, Australia; E-mail: shirui.pan@student.uts.edu.au; jia.wu@student.uts.edu.au; guodong.long@uts.edu.au; chengqi.zhang@uts.edu.au
X. Zhu is with the Dept. of Computer and Electrical Engineering & Computer Science, Florida Atlantic University. E-mail: xzhu3@fau.edu;
*Manuscript received ** **, 20xx; revised ** **, 20xx.*

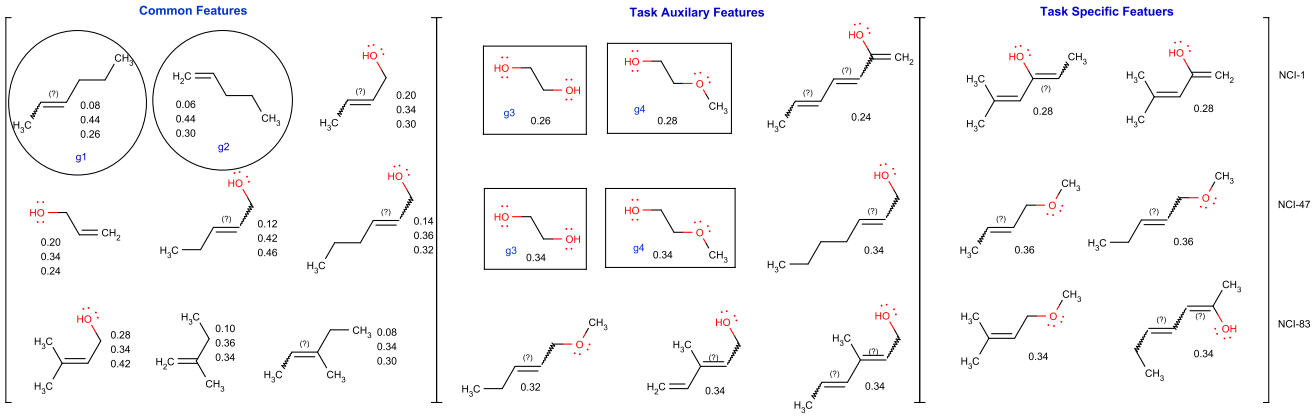

Fig. 1. Feature learning and categorization for three graph classification tasks (detailed in Experiments). The first column shows the top nine subgraphs shared by all tasks (learned by our algorithm). Numeric values next to each subgraph features indicate the feature utility for NCI-1, 47, and 83 tasks (measured by $|\frac{1}{n_t}\sum_{i=1}^{n_t} y_i x_i^j|$, 0 means the feature is not discriminative while 1 means the feature perfectly classifies all graphs). The second and third columns show the task auxiliary features and task specific features learned by our algorithms (each row corresponding to one task). Existing common feature based MTL algorithm might not find all discriminative features as they ignore the uniqueness of each task. For instance, g_1 and g_2 are selected as common features but they have limited capability in classifying graphs from task NCI-1. By considering task auxiliary features and task specific features, the unique property of each task can be well preserved. It is evident that the task specific subgraphs are even more discriminative than g_1 and g_2 for NCI-1. Note that task auxiliary features are used by a subset of tasks. For instance g_3 and g_4 are only used by NCI-1 and NCI-47, but not by NCI-83.

tasks raise an interesting question as to whether multiple brain function classification tasks can be combined to learn a multi-task model for maximum performance gain.

Chemical Compound Categorization is important in biomedical research for testing whether a chemical compound is active to a specific cancer, such as melanoma. For melanoma cancer, determining activities of a molecule is expensive as it requires time, efforts, and expensive resources [23] to conduct biological assay. In reality, some similar bioassay tasks [1], such as an anti-cancer test for prostate, may be available. As the graph data for different types of cancer may share common substructures, learning multiple related tasks together may potentially help improve the generalization performance of each single task.

Indeed, when several relevant tasks are provided for learning, existing research on multi-task learning has demonstrated that exploring commonality between tasks can improve the generalization performance. To support MTL, existing algorithms commonly rely on two types of approaches: (1) multi-task feature learning, which explores common feature space shared by all tasks. The models, including mixed $\ell_{2,1}$ norm sparsity inducing methods [24], [25], composite regularized algorithms [26], [27], and the most recent calibration based multi-task approach [28], can be formulated as a regularized loss minimization problem aiming to explore shared feature space among tasks for learning; and (2) task relationship learning, which simultaneously exploits task relationships and parameters [29], such as task clustering [30], [31] or isolating [32], so that knowledge can be shared by a group of tasks instead of all tasks.

Although MTL has been applied to many applications, all existing methods work on domains with feature-vector representation, but cannot be applied to structured data, such as graphs. Intuitively, one can first convert graphs into feature-vector representation by first discovering a set of frequent subgraphs as features and transferring graphs into vector format, and then employ state-of-the-art MTL algorithms [24], [28]. Unfortunately, this simple approach separates subgraph feature selection from the multi-task learning process, and may fail to find discriminative subgraph features for learning, as evidenced in previous single task graph classification algorithms [8].

In addition, even if one can find good subgraphs to represent graphs as feature-vector instances, existing MTL methods still cannot effectively handle multi-task graph classification. This is mainly because shared commonality, either in the feature space or in task correlations, is a commonly believed theme of existing MTL methods. Accordingly, these methods emphasize on comparing tasks and finding common feature space or correlations for learning, but not on the uniqueness of individual tasks. This is potentially harmful for graph classification domains, because graphs from similar domains usually share high global similarity but only differ in a small set of substructures, which may be beneficial for model learning and should be carefully preserved. More importantly, these unique features are helpful for users to uncover actual patterns shared by different tasks. For instance, in the drug discovery process, experts are expected to find common substructures shared by a set of cancer types, as well as discovering features unique to a specific cancer. Existing MTL methods, unfortunately, mainly focus on exploring the overall commonality or disparity between tasks,

1. https://pubchem.ncbi.nlm.nih.gov/

but cannot explicitly capture detailed relationships between tasks and individual features. An example is showing Fig. 1 where three types of cancer diagnose tasks share some common subgraph features for all tasks (1st column). Some features are shared by a subset of tasks (2nd column), and some subgraph features are unique for each individual task (3rd column).

Motivated by the above observations, in this paper, we propose a task sensitive feature exploration and learning algorithm for multi-task graph classification (namely FelMuG). A unique feature of FelMuG is that it explores and learns to classify subgraph features into three groups: (1) common features, (2) task auxiliary features, and (3) task specific features. Common features are the ones shared by all tasks, task auxiliary features can be shared by any subset of tasks, and a task specific feature is unique to a single task. Task sensitive feature learning not only allows FelMuG to improve the performance of multi-task learning but also enhances the understanding of task relationships at the feature level. At the final stage, the learned subgraph features and graph classification tasks are iteratively co-regularized to form an optimization function, with subgraph exploration and multi-task learning being iteratively optimized for maximum performance gain.

Our work makes noticeable contributions in the following three aspects:

- **Feature learning and categorization for multi-task learning:** we propose a novel task sensitive feature learning algorithm to select and categorize features into different groups. It not only helps improve the classification accuracy but also provides solutions for understanding relationships and the uniqueness of different tasks at the feature level.
- **Cross task subgraph exploration:** we derive an effective pruning bound to reduce the exponential subgraph search space, which facilitates our algorithm to explore discriminative subgraph features without specifying any support threshold.
- **Multi-task graph classification:** we advance the single task graph classification setting to multi-task scenarios, which jointly explore and learn multiple classification models to improve the classification performance over single task graph classification.

2 DEFINITIONS & PRELIMINARIES

2.1 Problem Definition

Definition 1: **Connected Graph:** A graph is denoted by $G = (\mathcal{V}, E, L)$, where $\mathcal{V} = \{v_1, \cdots, v_{n_v}\}$ is a set of vertices, $E \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, and L is a labelling function assigning labels to a node or an edge. A connected graph is a graph with a path between any pair of vertices.

In this paper, we focus on connected graphs and assume that each graph G has a class label y, $y \in \mathcal{Y} = \{-1, +1\}$, indicating label information of the graph, such as an active/negative response of a chemical compound [33].

Definition 2: **Subgraph:** Given two graphs $G = (\mathcal{V}, E, L)$ and $g_k = (\mathcal{V}', E', L')$, g_k is a subgraph of G (*i.e.* $g_k \subseteq G$) if there is an injective function f: $\mathcal{V}' \rightarrow \mathcal{V}$, such that $\forall (a, b) \in E'$, we have $(f(a), f(b)) \in E$, $L'(a) = L(f(a))$, $L'(b) = L(f(b))$, $L'(a, b) = L(f(a), f(b))$. If g_k is a subgraph of G ($g_k \subseteq G$), G is a supergraph of g_k ($G \supseteq g_k$).

Multi-task Graph Classification: Given a set of graph classification tasks, where each task $t \in \{1, 2, \cdots, T\}$ has a set of labeled graphs $\{(G_{t,1}, y_{t,1}), \cdots, (G_{t,n_t}, y_{t,n_t})\}$, we use $G_{t,i} \in \mathcal{G}$ ($\mathcal{G}$ is the graph space) to denote the i^{th} graph in task t, and $G_{t,i}$'s class label is $y_{t,i} \in \mathcal{Y} = \{+1, -1\}$. Multi-task graph classification **aims** to learn T functions (classification models) $f_t : \mathcal{G} \rightarrow \mathcal{Y}, t \in [1, T]$, which have the best classification accuracy on test graphs over *all* tasks.

2.2 PRELIMINARIES

Single Task Graph Classification: To support graph classification, state-of-the-art algorithms [8], [10] use a set of subgraphs from training graphs as features. Each subgraph g_k can map a given graph $G_{t,i}$ into the class label space $\mathcal{Y} = \{+1, -1\}$ by using a simple decision stump as follows:

$$\hbar_{g_k}(G_{t,i}) = 2I(g_k \subseteq G_{t,i}) - 1; \tag{1}$$

Here $I(a) = 1$ if a holds, and 0 otherwise. The rule simply maps a graph G_{ti} into +1 if $g_k \in G_{ti}$ otherwise -1.

Let $\mathcal{F} = \{g_1, \cdots, g_m\}$ be the full set of subgraphs in $\mathcal{G}$. We can use $\mathcal{F}$ as features to represent each graph $G_{t,i}$ into a vector space as $\boldsymbol{x}_{t,i} = [\hbar_{g_1}(G_{t,i}), \cdots, \hbar_{g_m}(G_{t,i})]^T$, with $\boldsymbol{x}^k_{t,i} = \hbar_{g_k}(G_{t,i})$. In this paper, $G_{t,i}$ and $\boldsymbol{x}_{t,i}$ are used interchangeably as they are both referred to the same graph (*i.e.*, the i-th graph in task t). Given full subgraphs $\mathcal{F}$, the prediction function of task t is a linear classifier:

$$f_t(\boldsymbol{x}_{t,i}) = \boldsymbol{w}_t^T \cdot \boldsymbol{x}_{t,i} + b_t = \sum_{g_k \in \mathcal{F}} w_{t,k} \hbar_{g_k}(G_{t,i}) + b_t \tag{2}$$

where $\boldsymbol{w}_t = [\boldsymbol{w}_{t,1}, \cdots, \boldsymbol{w}_{t,m}]^T$ is the weight vector of all features for task t, and b_t is the bias of the model. The predicted class of $\boldsymbol{x}_{t,i}$ is +1 if $f_t(\boldsymbol{x}_{t,i}) > 0$, or -1 otherwise.

Note that for graph data, the feature set $\mathcal{F}$ is unavailable and is exponentially large (even infinite). In next section, we first propose a novel task sensitive feature learning algorithm for graph tasks with feature-vector representation (*i.e.*,

assuming $\mathcal{F}$ is known). In Section 4, we will show how this algorithm can be integrated into subgraph mining process to explore subgraph features $\mathcal{F}$ for general graph tasks.

3 MULTI-TASK GRAPH CLASSIFICATION

In this section, we first formulate a task sensitive feature learning algorithm for multi-task classification (Section 3.1). Because the formulated problem is a mixed integer problem, we relax it to a convex semi-infinite problem (SIP) in Section 3.2. Because SIP relaxation results in infinite constraints, we further propose an advanced cutting plane optimization algorithm in Section 3.3 to solve the problem.

3.1 Task Sensitive Feature Learning for Multi-Task Learning

In traditional support vector machines (SVM), one learns a linear function $f_t(\boldsymbol{x}_{t,i}) = \boldsymbol{w}_t^T \cdot \boldsymbol{x}_{t,i} + b_t$ by solving the following ℓ_2-norm regularized problem:

$$\min_{\boldsymbol{w_t}, b_t} \frac{1}{2}\|\boldsymbol{w}_t\|^2 + C\sum_{i=1}^{n_t} \mathcal{L}(y_{t,i}, f_t(\boldsymbol{x}_{t,i})) \tag{3}$$

where $L(y_{t,i}, f_t(\boldsymbol{x}_{t,i})) = \max(1 - y_{t,i}, f_t(\boldsymbol{x}_{t,i}), 0)$ is a hinge loss function, and C is a parameter controlling the regularization part.

When multiple tasks are given, existing MTL mainly focuses on exploring commonality between tasks, such as common feature space or task similarity, for learning. Such approaches are unsuitable for graph classification tasks because we need to gradually explore subgraph feature space, as well as model task relationships, by using explored features, for maximum performance gain.

Accordingly, our research advocates a new *task sensitive feature learning theme* to explore and categorize features into different non-overlapping groups: **common features**, **task auxiliary features**, and **task specific features**. Common features are the ones shared by all tasks, task auxiliary features are shared by a subset of tasks, and a task specific feature is only used by a particular task. By doing so, we can model common feature space among tasks, like most existing MTL algorithms do, and also capture discriminative features with respect to any subset of tasks or a single task. The explicit capturing of interrelation between features and tasks allows our method to uncover fine-grained task relationships in the feature space.

As feature learning aims to select non-overlapping groups, the three groups impose hard constraints on the features. Specifically, for each task we introduce three feature scaling vectors, with $\boldsymbol{\delta}_0 = [\delta_0^1, \cdots, \delta_0^m] \in \{0,1\}^m$ corresponding to common features, $\boldsymbol{\delta}_{ts} = [\delta_{ts}^1, \cdots, \delta_{ts}^m] \in \{0,1\}^m$ for task specific features, and $\boldsymbol{\delta}_{ta} = [\delta_{ta}^1, \cdots, \delta_{ta}^m] \in \{0,1\}^m$ for task auxiliary features, respectively.

For $\boldsymbol{\delta}_0$, $\boldsymbol{\delta}_{ts}$ or $\boldsymbol{\delta}_{ta}$, the j-th feature is selected as a common feature if $\delta_0^j = 1$, as a task specific feature if $\delta_{ts}^j = 1$, or as a task auxiliary feature if $\delta_{ta}^j = 1$, exclusively. As a result, we can obtain a re-scaled instance for $\boldsymbol{x}_{t,i}$ as follows:

$$\hat{\boldsymbol{x}}_{t,i} = \boldsymbol{x}_{t,i} \odot \boldsymbol{\delta}_t, \qquad \boldsymbol{\delta}_t = \boldsymbol{\delta}_0 + \boldsymbol{\delta}_{ta} + \boldsymbol{\delta}_{ts} \tag{4}$$

where $\odot$ is an element-wise product of two vectors. To control selected features for final classification model, we propose following constraints on the feature indicated vectors:

$$\begin{aligned}
&\|\boldsymbol{\delta}_0\|_1 = \textstyle\sum_{j=1}^m \delta_0^j \le K_0, \delta_0^j \in \{0,1\} \\
&\|\boldsymbol{\delta}_{ts}\|_1 = \textstyle\sum_{j=1}^m \delta_{ts}^j \le K_s, \delta_{ts}^j \in \{0,1\} \\
&\|\boldsymbol{\delta}_{ta}\|_1 = \textstyle\sum_{j=1}^m \delta_{ta}^j \le K_a, \delta_{ta}^j \in \{0,1\} \\
&\boldsymbol{\delta}_0^T \cdot \boldsymbol{\delta}_{ts} = 0; \quad \boldsymbol{\delta}_0^T \cdot \boldsymbol{\delta}_{ta} = 0; \quad (\boldsymbol{\delta}_{ts})^T \cdot \boldsymbol{\delta}_{ta} = 0 \\
&\textstyle\sum_{t=1}^T \delta_{ts}^j \le 1;
\end{aligned} \tag{5}$$

where K_0, K_s, and K_a are integers indicating the least number of features used in the final models. The fourth constraint enforces non-overlapping between groups, and the last constraint enforces that a task specific feature is preserved and unique for one task only.

In order to learn multiple tasks via feature learning, we formulate the following objective function:

$$\begin{aligned}
\min_{\boldsymbol{\delta}_t \in \mathcal{D}} \min_{\boldsymbol{w_t}, b_t, \xi_{t,i}} \quad & \tfrac{1}{2}\textstyle\sum_{t=1}^T \|\boldsymbol{w}_t\|^2 + C\sum_{t=1}^T \sum_{i=1}^{n_t} \xi_{t,i} \\
s.t. \quad & y_{t,i}\Big(\boldsymbol{w}_t^T\big(\boldsymbol{x}_{t,i} \odot \boldsymbol{\delta}_t\big) + b_t\Big) + \xi_{t,i} \ge 1, \\
& \xi_{t,i} \ge 0, t = 1, \cdots T, i = 1 \cdots n_t
\end{aligned} \tag{6}$$

where $\mathcal{D} = \{\|\boldsymbol{\delta}_0\|_1 \leq K_0; \|\boldsymbol{\delta}_{ts}\|_1 \leq K_s; \|\boldsymbol{\delta}_{ta}\|_1 \leq K_a; \boldsymbol{\delta}_0^T\boldsymbol{\delta}_{ts} = 0;\ \boldsymbol{\delta}_0^T\boldsymbol{\delta}_{ta} = 0;\ (\boldsymbol{\delta}_{ts})^T\boldsymbol{\delta}_{ta} = 0; \sum_{t=1}^{T}\delta_{ts}^j \leq 1; \boldsymbol{\delta}_0, \boldsymbol{\delta}_{ts}, \boldsymbol{\delta}_{ta} \in \{0,1\}^m\}$.

Merit of our design It is worth noting that we use $\boldsymbol{\delta}_0$, $\boldsymbol{\delta}_t$, and $\boldsymbol{\delta}_a$ to directly learn and categorize features, with learnt features being used to learn the optimization model. In [34], the authors decompose the weight as $\boldsymbol{w}_t = \boldsymbol{w}_0 + \boldsymbol{v}_t$; and existing composite regularization methods such as dirty model [26] and rMTFL method [27] factorizes the weight matrix of all tasks $W = P + Q$ with different sparsity inducing regularizers. However their formulations cannot explicitly capture unique discriminative features for a specific task or for a subset of tasks. Furthermore, the ℓ_1 or mixed norm ℓ_{21} regularizers used in these methods attempt to control the number of select features and the model performance simultaneously. When the number of selected feature is small, the learnt model will be biased and under-fit the training data, resulting poor performance. This is attributed to the biased of ℓ_1 norm regularization effects [35]. Comparing with state-of-the-art multi-task feature learning methods, the merit of our design is fourfold:

1) It selects features naturally with desired cardinality. This is more effective than sparsity induced cardinality methods such as $\ell_{2,1}$ regularization.
2) The selected features are automatically categorized into different groups, *i.e.*, common features, task specific features, and task auxiliary features. This is particularly important for graph classification tasks. Because with categorized features, experts can easily identify common substructures active against several types of cancers, or find unique features for a specific type of cancer.
3) The proposed model can be transferred to a convex programming problem, based on which an effective solver can be developed. Similar scheme has been used in [35], which is, in fact, a special case of our multi-task learning formula, with only one task being used.
4) The proposed method can be naturally integrated to the subgraph mining process to facilitate graph classification.

The optimization problem in Eq.(6) is a mixed integer problem (MIP), which is non-convex when considering $\mathbf{W} = [\boldsymbol{w}_1, \cdots \boldsymbol{w}_t]$ and $\boldsymbol{\delta}_t$ together. Furthermore, the number of valid constants $\boldsymbol{\delta}_t \in \mathcal{D}$ is complicated and infinitely large. Next, we will relax this formula to a convex problem.

3.2 Convex Relaxation of MTL

Considering the inner minimization problem of Eq.(6) as a whole, its dual problem becomes:

$$\max_{\alpha_{t,i}\in\mathcal{A}} -\frac{1}{2}\sum_{t=1}^{T}\|\sum_{i=1}^{n_t}\alpha_{t,i}y_{t,i}\big(\boldsymbol{x}_{t,i}\odot\boldsymbol{\delta}_t\big)\|^2 + \sum_{t=1}^{T}\sum_{i=1}^{n_t}\alpha_{t,i} \tag{7}$$

where $\mathcal{A} = \{\alpha_{t,i}|\sum_{i=1}^{n_t}\alpha_{t,i}y_{t,i} = 0, \forall t \in [1,T]; 0 \leq \alpha_{t,i} \leq C\}$. For convenience, let $s_{t,j}$ be the feature score of the first term on the j-th dimension of task t, *i.e.*,

$$s_{t,j} = [\sum_{i=1}^{n_t}\alpha_{t,i}y_{t,i}\boldsymbol{x}_{t,i}^j]^2 \tag{8}$$

Let $s_j = \sum_{t=1}^{T} s_{t,j}$ be the feature score on the j-th dimension over all tasks, then the first term of Eq.(7) can be re-written as the sum of feature score over all features, *i.e.*,

$$\sum_{t=1}^{T}\|\sum_{i=1}^{n_t}\alpha_{t,i}y_{t,i}\big(\boldsymbol{x}_{t,i}\odot\boldsymbol{\delta}_t\big)\|^2 = \sum_{j=1}^{m}\delta^j\sum_{t=1}^{T}s_{t,j} = \sum_{j=1}^{m}\delta^j s_j$$

where $\delta^j = \delta_0^j + \delta_{ts}^j + \delta_{ta}^j$. For convenience, let $\boldsymbol{\alpha}$ be all lagrangian multipliers $\alpha_{t,i}$, and $\boldsymbol{\delta}$ be all indicated vectors $\boldsymbol{\delta}_t$ for all tasks, respectively, then

$$F(\boldsymbol{\alpha},\boldsymbol{\delta}) = -\frac{1}{2}\sum_{j=1}^{m}\delta^j s_j + \sum_{t=1}^{T}\sum_{i=1}^{n_t}\alpha_{t,i} \tag{9}$$

Then the original objective function Eq.(6) becomes:

$$\min_{\boldsymbol{\delta}\in\mathcal{D}}\max_{\boldsymbol{\alpha}\in\mathcal{A}} F(\boldsymbol{\alpha},\boldsymbol{\delta}) \tag{10}$$

According to the minmax inequality [36], we have the following lower bound to Eq.(6):

$$\min_{\boldsymbol{\delta}\in\mathcal{D}}\max_{\boldsymbol{\alpha}\in\mathcal{A}} F(\boldsymbol{\alpha},\boldsymbol{\delta}) \geq \max_{\boldsymbol{\alpha}\in\mathcal{A}}\min_{\boldsymbol{\delta}\in\mathcal{D}} F(\boldsymbol{\alpha},\boldsymbol{\delta}) \tag{11}$$

Moreover, this relaxed problem can be further transferred to a semi-infinite problem (SIP). Motivated by [35], [37], we introduce another variable $\omega \in R$, the last term of Eq.(11) can be formulated as the following SIP.

$$\max_{\boldsymbol{\alpha}\in\mathcal{A}}\Big\{\max_{\omega}\omega:\quad \omega \leq F(\boldsymbol{\alpha},\boldsymbol{\delta})\Big\}, \forall\boldsymbol{\delta}\in\mathcal{D} \tag{12}$$

Algorithm 1 Task Sensitive Feature Learning for Multi-task Graph Classification (FelMuG)

1: $\alpha_{t,i} = 1/n_t$; $\mathcal{C} \leftarrow \emptyset$; $o \leftarrow 0$;
2: Select the most violated constraint $\boldsymbol{\delta}^{(k)}$based on $\alpha_{t,i}^{(o)}$; // **Algorithm 2**
3: $\mathcal{C} \leftarrow \mathcal{C} \bigcup \boldsymbol{\delta}^{(k)}$;
4: Solve MKMT problem Eq.(14) to get the optimal $\boldsymbol{\alpha}^{(o)}$ and $\boldsymbol{\mu}^{(o)}$;
5: $o \leftarrow o + 1$;
6: repeat 2-5 until convergence.

3.2.1 From MTL to Multi-Kernel Learning

The problem in Eq.(12) can be solved in its dual domain. Introducing another set of Lagrangian variables $\boldsymbol{\mu} = \{\mu_k\}$, we will have the new Lagrangian function, *i.e.*,

$$L(\omega, \boldsymbol{\mu}) = \omega + \sum_{\boldsymbol{\delta}^{(k)} \in \mathcal{D}} \mu_k (F(\boldsymbol{\alpha}, \boldsymbol{\delta}) - \omega)$$

Setting its first derivative to 0 *w.r.t* ω, we have $\sum_{\boldsymbol{\delta}^{(k)} \in \mathcal{D}} \mu_k = 1$. Let $\mathcal{M} = \{\boldsymbol{\mu} | \sum_{\boldsymbol{\delta}^{(k)} \in \mathcal{D}} \mu_k = 1, \mu_k \geq 0\}$ be the domain of $\boldsymbol{\mu}$. The dual problem of Eq.(12) can be written as follows:

$$\begin{aligned} & \max_{\boldsymbol{\alpha} \in \mathcal{A}} \min_{\boldsymbol{\mu} \in \mathcal{M}} \mu_k F(\boldsymbol{\alpha}, \boldsymbol{\delta}) \\ = & \min_{\boldsymbol{\mu} \in \mathcal{M}} \max_{\boldsymbol{\alpha} \in \mathcal{A}} -\tfrac{1}{2} \sum_{t=1}^{T} (\boldsymbol{\alpha}_{t,\cdot} \odot \boldsymbol{y}_{t,\cdot})^T \Big(\sum_{\boldsymbol{\delta}^{(k)} \in \mathcal{D}} \mu_k \boldsymbol{X}_{t,k} \boldsymbol{X}_{t,k}^T \Big) \\ & (\boldsymbol{\alpha}_{t,\cdot} \odot \boldsymbol{y}_{t,\cdot}) + \alpha_{t,\cdot} \mathbf{1} \end{aligned} \tag{13}$$

where we have $\boldsymbol{X}_{t,k} = [\boldsymbol{x}_{t,i} \odot \boldsymbol{\delta}^{(k)}, \cdots, \boldsymbol{x}_{t,n_t} \odot \boldsymbol{\delta}^{(k)}]$, and $\boldsymbol{\alpha}_{t,\cdot}$ and $\boldsymbol{y}_{t,\cdot}$ are the lagrangian multiplier and training class labels for task t. Accordingly, the problem becomes a convex optimization problem, whose global optimal can be obtained. More specifically, it is a **multi-kernel multi-task problem** (MKMT) [38], where $\boldsymbol{X}_{t,k}$ can be seem as a kernel defined on a subset of features $\boldsymbol{\delta}^{(k)}$ on the t-th task (with T tasks in total). For each task, we aims to learn a convex combination of a set of kernels by $\sum_{\boldsymbol{\delta}^{(k)} \in \mathcal{D}} \mu_k \boldsymbol{X}_{t,k} \boldsymbol{X}_{t,k}^T$. Across tasks, they share a set of common kernel functions whose index is defined by μ_k.

3.3 Cutting Plane for Infinite Constraint Optimization

The main challenge to solve MKMT problem (Eq.(13)) is that there are an infinite number of constraints ($\boldsymbol{\delta}^{(k)} \in \mathcal{D}$). Fortunately, not all constraints are active at optimality. In this paper, we propose to solve this problem by using Cutting Plane algorithm [39]. The idea of cutting plane algorithm is to start with an empty working set $\mathcal{C}$ and iteratively selects the most violated constraint $\boldsymbol{\delta}^{(k)}$ to be included into the working set $\mathcal{C}$ and re-solve the reduced problem of Eq.(13). The whole process continues until there is no more active constraint. Because in each iteration, the size of $\mathcal{C}$ is relatively small, the cutting plane algorithm is very efficient for large-scale data/features optimization. Algorithm 1 list the procedures of using cutting plane for feature learning. After convergence, the final prediction rule for $\boldsymbol{x}_{t,q}$ from task t is $f(\boldsymbol{x}_{t,q}) = \sum_{k=1}^{|\mathcal{C}|} \mu_k \sum_{i=1}^{n_t} \alpha_{t,i} y_{t,i} (\boldsymbol{x}_{t,i} \odot \boldsymbol{\delta}^{(k)}) \boldsymbol{x}_{t,q}^T$.

The main steps of cutting plane algorithm consist of two components: (1) MKMT subproblem solving, and (2) The most violated constraint selection. In the following, we introduce solutions to the two sub-problems.

3.3.1 Solving the MKMT Subproblem

Multiple Kernel Multi-task learning (MKMT) was studied recently in [38], which assumes that over a set of T tasks there are a set of kernel matrices in each task. By selecting a common kernel representation, multiple tasks can be mutually beneficial to each other. Based on the current selected kernel set $\mathcal{C}$ in step 3 of Algorithm 1, our subproblem is equal to solve the following reduced MKMT subproblem of Eq.(13):

$$\begin{array}{cc} \min\limits_{\boldsymbol{\mu} \in \mathcal{M}} \max\limits_{\boldsymbol{\alpha} \in \mathcal{A}} & -\tfrac{1}{2} \sum\limits_{t=1}^{T} (\boldsymbol{\alpha}_{t,\cdot} \odot \boldsymbol{y}_{t,\cdot})^T \mathbf{K} (\boldsymbol{\alpha}_{t,\cdot} \odot \boldsymbol{y}_{t,\cdot}) + \alpha_{t,\cdot} \mathbf{1}; \\ \text{with} & \mathbf{K} = \sum_{\boldsymbol{\delta}^{(k)} \in \mathcal{C}} \mu_k \boldsymbol{X}_{t,k} \boldsymbol{X}_{t,k}^T \end{array} \tag{14}$$

The problem in Eq.(14) can be effectively solved by using existing MKMT solvers [38]. Specifically, we can solve Eq.(14) via an iterative procedure: (1) fixing $\boldsymbol{\mu}$ and solving Eq.(14) to update $\boldsymbol{\alpha}$ based on a set of $\boldsymbol{\delta}^{(k)} \in \mathcal{C}$, which boils down to solving T independent standard SVM problem; then (2) fixing $\boldsymbol{\alpha}$ and using reduced gradient method [40] to update $\boldsymbol{\mu}$. The two steps continue until they converge.

3.3.2 Most Violated Constraint Selection

Finding the most violated constraint given $\boldsymbol{\alpha}$ in Eq.(12) is the same as solving the problem of $\min_{\boldsymbol{\delta}\in\mathcal{D}} F(\boldsymbol{\alpha},\boldsymbol{\delta})$. Because the second term in Eq.(9) is constant given $\boldsymbol{\alpha}$, the problem is reduced to $\max_{\boldsymbol{\delta}} \sum_{j=1}^{m} \delta^j s_j$. The complicated constraint in $\mathcal{D}$ can be rewritten as a set of summarization constraints, leading to the following problem:

$$\begin{array}{ll} \max\limits_{\boldsymbol{\delta}_0,\boldsymbol{\delta}_{ts},\boldsymbol{\delta}_{ta}} & \sum_{j=1}^{m} \delta^j s_j \\ s.t. & \|\boldsymbol{\delta}_0\|_1 = \sum_{j=1}^{m} \delta_0^j \le K_0, \delta_0^j \in \{0,1\}; \\ & \|\boldsymbol{\delta}_t\|_1 = \sum_{j=1}^{m} \delta_{ts}^j \le K_s, \delta_{ts}^j \in \{0,1\}; \\ & \|\boldsymbol{\delta}_a\|_1 = \sum_{j=1}^{m} \delta_{ta}^j \le K_a, \delta_{ta}^j \in \{0,1\}; \\ & \delta^j = \delta_0^j + \delta_a^j + \delta_t^j \le 1; \\ & \sum_{t=1}^{T} \delta_t^j \le 1; \end{array} \tag{15}$$

Accordingly, the problem becomes a binary and linear programming problem, *i.e.*, the knapsack problem [41]. If the features are known, many methods such as dynamic programming or the greedy algorithm can be applied to solve the problem. Various optimization solvers such as the optimization toolbox provided by MATLAB can address this problem effectively.

Challenge for Graph Data: For graph data, Eq.(15) has a major difficulty, because (1) subgraph feature set $\boldsymbol{x}_{t,i} = [\hbar_{g_1}(G_{t,i}), \cdots, \hbar_{g_m}(G_{t,i})]^T$ is **unknown**, and (2) the number of subgraph features is **exponentially large** (or infinite). For instance, if the maximum number of nodes in the graph database is n_v, the number of subgraphs is $m = 2^{n_v}$. In next section, we seamlessly integrate this problem into the subgraph exploration process to explore subgraph features for graph classification tasks.

4 Multi-Task Subgraph Exploration

For graph classification, finding the most violated constraint, *i.e.*, solving Eq.(15), in each iteration for our algorithm is NP hard as it requires enumeration of the whole subgraph space. One possible way is to first mine a set of frequent subgraphs, and then apply the multi-task algorithm. But this is subject to the risk of missing discriminative subgraphs, because not every subgraph is checked and evaluated across all tasks. In this section, we propose an effective algorithm to handle the infinite subgraph problem. Our idea is to employ subgraph mining algorithm gSpan [42] to mine a small set of potential subgraph features $\mathcal{F}_p$, and then solve Eq.(15) based on $\mathcal{F}_p$. For the potential subgraph feature set $\mathcal{F}_p$, we ensure that if a subgraph does not appear in $\mathcal{F}_p$, it will not be selected in the optimal set defined in Eq.(15). As $\mathcal{F}_p$ is very small, Eq.(15) can be solved efficiently.

4.1 Multi-Task Subgraph Selection

Although we cannot directly obtain the subgraph features which maximize $\sum_{j=1}^{m} \delta^j s_j$ subject to $\boldsymbol{\delta} \in \mathcal{D}$, we can reduce the potential subgraph features by employing top K subgraph mining procedures. For convenience, define $K = K_0 + \sum_t K_s + \sum_t K_a$, and let $\mathcal{F}_t$ be the set of subgraphs with top K utility scores defined by $s_{t,j} = [\sum_{i=1}^{n_t} \alpha_{t,i} y_{t,i} \boldsymbol{x}_{t,i}^j]^2$ (Eq. (8)) on task t. Then we can construct a small set of potential subgraphs $F_s = \bigcup F_t$.

***Proposition** 1:* $\forall g_j \in \mathcal{F}$, if $g_j \notin \mathcal{F}_p$, then δ_0^j, δ_{ts}^j, and δ_{ta}^j will be 0 defined in Eq.(15).

This proposition can be assured as we select at most $K = K_0 + \sum_t K_s + \sum_t K_a$ subgraph in Eq.(15). If one graph is not the top K highest subgraph in any task, *i.e.*, $g \notin \mathcal{F}_p$, it will not be selected by solving Eq.(15).

Now the problem in Eq.(15) is decomposed to T independent top K subgraph mining problems. Specifically, for each task, we aim to select K subgraph features with highest discriminative scores, defined by $s_{t,j}$ in Eq.(8). Then we can solve Eq.(15) effectively via binary and linear problem solvers. The algorithm for most violated constraint solving for graph data is illustrated in Algorithm 2.

Top K subgraph Mining In order to discover top K subgraphs, it requires the search of exponentially large subgraph space. In this subsection, we derive an upper-bound for each subgraph score, and use the branch-and-bound scheme to reduce the subgraph space.

***Theorem** 1:* **(Single Task Feature Score Upper-bound:)** Let g_j and g_q be two subgraph patterns, and $g_j \subseteq g_q$, for the subgraph g_j, we define on the t-th task,

$$\begin{array}{rcl} A_1(g_j) & = & 2\sum_{\{i|y_{t,i}=+1, g_j \in G_{t,i}\}} \alpha_{t,i} \\ A_2(g_j) & = & 2\sum_{\{i|y_{t,i}=-1, g_j \in G_{t,i}\}} \alpha_{t,i} \\ A_3 & = & \sum_{i=1}^{n_t} \alpha_{t,i} y_{t,i} \end{array}$$

$$\hat{\Theta}(g_j,t) = \begin{cases} \max\{[A_1(g_j) - A_3]^2, [A_2(g_j)]^2\} & : \quad A_3 < 0 \\ \max\{[A_2(g_j) + A_3]^2, [A_1(g_j)]^2\} & : \quad A_3 \ge 0 \end{cases}$$

Algorithm 2 Most Violated Constraint Selection for Exponentially Large Subgraph Space

$K = K_0 + \sum_t K_s + \sum_t K_a$;
$\mathcal{F}_t \leftarrow$ mine top K subgraph by **Algorithm 3**, $\forall t \in 1 \cdots T$;
$\mathcal{F}_p \leftarrow \bigcup \mathcal{F}_t$;
Calculate $s_j = \sum_{t=1}^{T} s_{t,j}$ for each subgraph $g_j \in \mathcal{F}_p$ based on Eq. (8);
Solve Eq. (15) based on $\mathcal{F}_p$ to get $\boldsymbol{\delta}_0, \boldsymbol{\delta}_{tp}$, and $\boldsymbol{\delta}_{ta}$.

Algorithm 3 Top K Subgraph Mining

Require:
$\{(G_{t,i}, y_{t,i})\}_{i=1}^{n_t}$: Graph Datasets for the task t;
$\alpha_{t,i}$: Weight for each graph example;
K: Number of optimal subgraph patterns;
Ensure:
$\mathcal{F}_t = \{g_j\}_{j=1,\cdots,K}$: The top K subgraphs;
1: $\eta = 0$, $\mathcal{F}_t \leftarrow \emptyset$;
2: **while** Recursively visit the DFS Code Tree in gSpan **do**
3: $g_j \leftarrow$ current visited subgraph in DFS Code Tree;
4: **if** g_j has been examined **then**
5: **continue**;
6: **end if**
7: Compute scores $s_{t,j}$ for subgraph g_j according Eq.(8);
8: **if** $s_{t,j} > \eta$ **then**
9: $\mathcal{F}_t \leftarrow \mathcal{F}_t \bigcup g_j$;
10: **end if**
11: **if** $|\mathcal{F}_t| > K$ **then**
12: $g^\star \leftarrow \arg\min_{g_k \in \mathcal{F}_t} \Theta(g_k)$;
13: $\mathcal{F}_t \leftarrow \mathcal{F}_t / g^\star$;
14: $\eta \leftarrow \min_{g_k \in \mathcal{P}} \Theta(g_k)$;
15: **end if**
16: **if** $\hat{\Theta}(g_j, t) > \eta$ **then**
17: Depth-first search the subtree rooted from node g_p;
18: **end if**
19: **end while**
20: **return** $\mathcal{F}_t = \{g_j\}_{j=1,\cdots,K}$;

then $s_{t,q} \leq \hat{\Theta}(g_j)$, where $s_{t,q}$ is defined as the single task feature score in Eq.(8).

Proof: We start with the definition of $s_{t,j}$:

$$
\begin{aligned}
s_{t,q} &= [\textstyle\sum_{i=1}^{n_t} \alpha_{t,i} y_{t,i} \boldsymbol{x}_{t,i}^q]^2 \\
&= [\sum_{i=1}^{n_t} y_{t,i}\alpha_{t,i} \cdot \{2I(g_q \subseteq G_{t,i}) - 1\}]^2 \\
&= [2 \sum_{g_q \subseteq G_t} y_{t,i}\alpha_{t,i} - \textstyle\sum_{i=1}^{n_t} \alpha_{t,i} y_{t,i}]^2 \\
&= |A_1(g_q) - A_2(g_q) - A_3|^2 \\
&\leq \begin{cases} \max\{[A_1(g_q) - A_3]^2, [A_2(g_q)]^2\} & : \quad A_3 < 0 \\ \max\{[A_2(g_q) + A_3]^2, [A_1(g_q)]^2\} & : \quad A_3 \geq 0 \end{cases} \\
&\leq \begin{cases} \max\{[A_1(g_j) - A_3]^2, [A_2(g_j)]^2\} & : \quad A_3 < 0 \\ \max\{[A_2(g_j) + A_3]^2, [A_1(g_j)]^2\} & : \quad A_3 \geq 0 \end{cases} \\
&= \hat{\Theta}(g_j, t)
\end{aligned}
$$

The first inequality holds as for $\alpha_{t,i} \geq 0$, $A_1(g_q) \geq 0$ and $A_2(g_q) \geq 0$, so the upper-bound depends on A_3. If $A_3 < 0$, $A_1(g_q)$ and A_3 will have different signs, then the upper-bound is a maximum of $\{[A_1(g_q) - A_3]^2, [A_2(g_q)]^2\}$. The case is similar for $A_3 \geq 0$. The second inequity holds because $A_1(g_q) \leq A_1(g_j)$ and $A_2(g_q) \leq A_2(g_j)$ for $g_j \subseteq g_q$. □

According to Theorem 1, once a subgraph g_j is generated, the feature scores for all its super-graphs are upper-bounded by $\hat{\Theta}(g_j, t)$. Therefore, we use this rule to prune exponentially unpromising candidates effectively.

Multi-Task Subgraph Mining Algorithm: Our multi-task subgraph mining algorithm is listed in Algorithm 3. The minimum value η in optimal set $\mathcal{F}_t$ is initialized on step 1. Duplicated subgraph features are pruned on steps 4-5, and the discriminative score $s_{t,j}$ for g_j are calculated on step 6. If $s_{t,j}$ is larger than η, we add g_j to the feature set $\mathcal{F}_t$ (steps 7-8). If the size of $\mathcal{F}_t$ exceeds the predefined size K, the subgraph with the minimum discriminative score is removed (steps 10-12), and the minimum optimal value η is updated. We use our branch-and-bound pruning rules, Theorems 1, to prune the search space on steps 13-14. Finally, the optimal set $\mathcal{F}_t$ is returned on step 15.

The above pruning process is a key feature of our algorithm, because it does not require any support threshold for subgraph mining. As a result, no discriminative subgraph will be missed by our algorithm.

5 TIME COMPLEXITY ANALYSIS

Our FelMuG algorithm can be applied to general multi-task learning with vector representation as well as graph data, as will be shown in our

Generic Multi-task Learning: Algorithm 1 runs iteratively in two steps: (1) solve MTML subproblem, and (2) select most violated constraints. The first step consists of T SVM training and updating the $\boldsymbol{\mu}$ vector. The SVM training requires approximately $O(n^{2.5})$ *w.r.t* the number of training instances n, or linear $O(\hat{m}n)$ *w.r.t* the number of instances n and selected features $\hat{m}$ by using advanced solvers, such as LIBLinear [43]. Updating $\boldsymbol{\mu}$ requires $O(T \cdot |\mathcal{C}| \cdot n_{t,sv}^2)$ with $n_{t,sv}$ denoting the number of support vectors related to task t [38]. For the second subproblem, we use Matlab function *bintprog* with polynomial time complexity in our experiment. Other methods such as dynamic programming may achieve a linear time complexity $O(\hat{m}\tau)$, where τ is the maximum capacity of the Knapsack problem. Assume the number of iterations for Algorithm 1 is S, and the number of iteration to solve MKMT problem is S_1, the time complexity of FelMuG for general multi-task learning is

$$O\Big(ST\big(S_1(\hat{m}n + |\mathcal{C}| \cdot n_{t,sv}^2) + \hat{m}\tau\big)\Big)$$

Note that both $|C|$ and $n_{t,sv}$ are very small because only a small number of constraints and a small amount of support vectors are involved. For S and S_1, both of them are very small as well. In our experiments, only tens of iterations are required to reach convergence. As a result, FelMuG is very efficient for multi-task classification.

Multi-task Graph Classification: For graph data, the time complexity becomes

$$O\Big(ST\big(S_1(\hat{m}n + |\mathcal{C}| \cdot n_{t,sv}^2) + \hat{m}\tau \cdot O(gSpan)\big)\Big)$$

where $O(gSpan)$ is the time complexity of gSpan style algorithm for top K subgraph mining. Intuitively, this is an NP-complete problem because the subgraph space is exponentially large. However, we have derived an upper-bound and use the branch-and-bound scheme to prune unpromising subgraphs. Our experiments will show the effectiveness of the pruning scheme. Furthermore, other technique such as re-using the subgraph space [8], [19] can be employed to construct a DFS tree in the first iteration. During the remaining iterations, one can search and expand this DFS tree effectively. In general, Note that $O(gSpan)$ time complexity is inevitable for subgraph based graph classification [8], [11].

6 EXPERIMENTS

In this section, we first validate the feature learning module of FelMuG on synthetic vector datasets, and then evaluate FelMuG's performance on three real-world domains for multi-task graph classification. The source code of FelMuG algorithm and the benchmark graph datasets are available online [2].

6.1 Experimental Settings

6.1.1 Benchmark data

We employ two types of benchmark data, synthetic vector data and real-world graph data, in our experiments. For synthetic data, we predefine a set of tasks with known feature relationships, so we can validate how effective FelMuG can capture/retrieve the pre-defined relationships. For real-world graphs, we demonstrate that FelMuG's feature learning and multi-task co-regularization achieves much better performance than its peers for function brain analysis and chemical compound classification.

Synthetic vector data are modified from the one used in [38] and include four tasks. Each task is a binary classification problem with m features. Of these m features, there are d_0 features shared by all tasks (*i.e.* common features). In addition, there are d_{ts} task specific features and d_{ta} task auxiliary features which are relevant to task t. Therefore, for each task t, there are $d_t = d_0 + d_{ts} + d_{ta}$ effective features corresponding to it. The d_t effective features follow a Gaussian probability density function with mean u and $-u$, respectively, to define a two class classification problem (the covariance matrices of Gaussian distributions are randomly drawn from a Wishart distribution). The mean values u are randomly drawn from $\{-1, +1\}^{d_t}$. The other $m - d_t$ non-effective features follow an independent and identically distributed Gaussian probability distribution with zero mean and unit variance for both classes. Similar to [38], we generate four tasks, each with $n = 100, n_v = 100$, and $n_{the} = 5000$ samples for training, validation, and testing, respectively. Before learning, all data are normalized to zeros mean and unit variance. Because we know ground-truth features relevant to a task t, we

2. http://www.cse.fau.edu/∼xqzhu/FelMuG/index.html

TABLE 1
Description of Graph Datasets

Collections	ID	#Pos	#Total	Dataset Description
NCI	1	1793	37349	Non-Small Cell Lung
	33	1467	37022	Melanoma
	41	1350	25336	Prostate
	47	1735	37298	Central Nerv Sys
	81	2081	37549	Colon
	83	1959	25550	Breast
	109	1773	37518	Ovarian
	123	2715	36903	Leukemia
	145	1641	37043	Renal
BrainNet	KKI	46	83	ADHD
	OHSU	44	79	Hyper/Impulsive (HI)
	Peking_1	36	85	Gender (GD)
PTC	Sub_{MR}	32	87	Male Rat (MR)
	Sub_{FR}	35	85	Female Rat (FR)
	Sub_{MM}	29	85	Male Mouse (MM)
	Sub_{FM}	35	88	Female Mouse (FM)

can easily evaluate the feature learning performance by comparing the learned final feature vectors $\boldsymbol{\delta}_t = \boldsymbol{\delta}_0 + \boldsymbol{\delta}_{ts} + \boldsymbol{\delta}_{ta}$ with the ground-truth $d = d_0 + d_{ts} + d_{ta}$.

NIC Anti-cancer activity prediction data[3] are benchmark for predicting biological activities of small molecules for different types of cancers. Each molecule is represented as a graph, with atoms representing nodes and bonds denoting edges. A molecule is positive if it is active against a certain type of cancer, or negative otherwise. Table 1 summarizes nine NCI graph classification tasks used in our experiments. We randomly select #Pos number of negative graphs from each original graph set to create a multi-task graph classification problem with balanced training graphs for each task. Note that although each of the nine tasks focuses on a specific type of cancer, all these tasks are relevant in cancer prediction and some common substructures may exist for all types of cancers (as shown in Fig. 1). This makes NCI an ideal benchmark for multi-task graph classification.

PTC Predictive Toxicology Challenge Data include a number of carcinogenicity tasks for toxicology prediction of chemical compounds[4]. The dataset we selected contains 417 compounds from four types of test animals: MM (male mouse), FM (female mouse), MR (male rat), and FR (female rat). Each compound is with one label selected from {CE, SE, P, E, EE, IS, NE, N}, which stands for Clear Evidence of Carcinogenic Activity (CE), Some Evidence of Carcinogenic Activity (SE), Positive (P), Equivocal (E), Equivocal Evidence of Carcinogenic Activity (EE), Inadequate Study of Carcinogenic Activity (IS), No Evidence of Carcinogenic Activity (NE), and Negative (N). Similar to [44], we set {CE, SE, P} as positive label, and {NE, N} as negative label. In order to formulate multiple tasks, we randomly split 417 compounds into four equal non-overlapping subsets. For each subset, we only consider one type of carcinogenicity test as its learning task. The subset information is also listed in Table 1.

BrainNet Functional Brain Network Analysis Data are constructed from the whole brain functional magnetic resonance image (fMRI) atlas [20]. The purpose of the study is to map brain as a network (or a graph) where each node corresponds to a region of Interest (ROI) and the edge indicates correlations between two ROIs. In our experiments, we use functional parcellation results, CC200, from [20], which parcellate each brain into 200 regions of interest. In order to discover relationships between ROIs, the mean values of each ROI are recorded with respect to certain voxel time courses. By using Pearson correlations between two time courses, we can calculate correlation between two ROIs, and a graph is constructed by connecting ROIs whose correlations is higher than a threshold value (r¿0.7 in our experiment). An example of fMRI functional parcellation results and the corresponding brain graph are shown in Fig. 2. In our experiment, we construct three brain classification tasks, corresponding to Attention Deficit Hyperactivity Disorder (ADHD) classification, Hyperactive-Impulsive classification (HI), and gender classification (GD). The detailed graph information is summarized in Table 1. For ADHD and HI tasks, the functional response is real values, so we discretize the functional response to binary values by using a simple threshold (f=50 in our experiment).

6.1.2 Baseline Methods

Multi-Task Learning Baselines: We compare the feature learning with state-of-the-art multi-task learning methods, *i.e.*, Sep-L_1, Logistic-L_1, Logistic-L_{21} [24], rMTFL [27], Dirty [26], and Calibration [28] approaches. Sep-L_1 performs

3. http://pubchem.ncbi.nlm.nih.gov
4. http://www.predictive-toxicology.org/ptc/

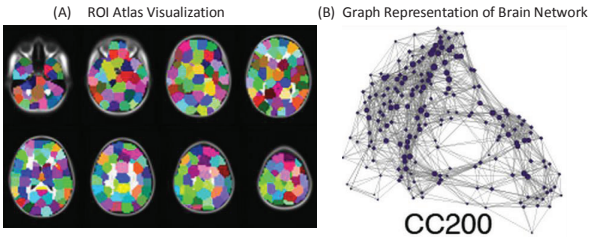

Fig. 2. An example of brain functional parcellation results from fMRI image (A), and corresponding brain graph (B). Each color region in (A) is a region of interest (ROI), which corresponds to a node in (B). (**rf.[20]**).

feature learning separately on each task and Logistic-L_1 jointly learns multi-task features with Lasso regularization and the logistic loss function. The calibration approach is the most recent multi-task feature learning algorithm [28]. All these algorithms are available in the MALSAR toolbox [45].

Graph Classification Baselines: We compare our method with three state-of-the-art graph classification methods, *i.e.*, gBoost [8], gSemi [11], and gMLC [12]. These methods learn graph classification task separately, without considering graph samples from other tasks. To build multi-task graph classification baselines, we use the above methods to firstly mine a set of frequent subgraphs from all training graphs (we set minimum support as 0.1, which results in over 2500 subgraph features on NCI datasets), and then use discovered subgraph features to transfer each graph dataset into a vector format, and then apply above multi-task learning baselines to the transferred vector datasets.

Unless otherwise specified, the parameters for FelMuG are empirically set as follows: C=0.1, K_a=2, and K_s=1. K_0 is selected from {5, 10, 15, 20}. Let the maximum iteration number in Algorithm 1 be S=15, then FelMuG will select at most $K \times S$ features with $K = K_0 + \sum K_a + \sum K_s$. For the comparing algorithms, we select the parameters from a set of candidates according to the property of each algorithm. For instance, for gBoost, the parameter v is selected from {0.2, 0.3, 0.4, 0.5}; and for Logistic-L21 algorithm, the parameter λ is chosen from 0.02 to 0.1 step by 0.02. The best parameters are selected based on the validation set. For graph classification, we repeat 10 times of each experiment for NCI graphs and conduct 10-fold cross-validation for PTC and brain networks. Following [8], we report the best average accuracy and average AUC values over all tasks for graph classification tasks.

6.2 Feature Learning Result Comparisons

Fig. 3 reports the feature learning results of different algorithms, which show that all MTL algorithms can obtain sparse solutions and find features shared by all tasks. Among baseline algorithms, Logistic-L21, rMTFL, and Calibration select similar features, because they aim to select common features (for rMTFL, no outlier task exists thus it achieves similar results to ℓ_{21} regularization). Meanwhile, because ℓ_1 regularization is used in Sep-L1, Logistic-L1, and Dirty methods, the learned features are more sparse for these algorithms. Nonetheless, the results show that FelMuG recovers the relevant features best among all algorithm because it explicitly captures common features, task specific features, and task auxiliary features for each task.

In Fig. 4, we vary the number of relevant features d_0 and report the feature selection and multi-task classification results, which show that FelMuG can recover much more relevant features than other algorithms. In addition, Fig. 4.(B) shows that FelMuG achieves the best feature recovery quality (F-measure) among all methods. This is because existing MTL algorithms mainly focus on common features but inherently overlook task specific and task auxiliary features. In contrast, our design not only learns common features, but also captures the underlying difference between each task. Fig. 4.(C) shows that FelMuG is comparable or outperforms state-of-the-art MTL algorithms in terms of accuracy.

6.3 Brain and Chemical Compound Graph Classification

Performance on NCI: For NCI tasks, we randomly label a small set of graphs as training graphs for each task, and the remaining graphs are used for testing. The number of training graphs in each task varies from 50 to 400. We report the average accuracies and AUC values over all tasks under 10 times of train/test split in Fig. 5.

The results in Fig. 5 show that when increasing training data for each task, all algorithms achieve continuous improvement in accuracy and AUC values. FelMuG outperforms graph classification baselines (including gBoost, gSemi, and gMLC). This is main because these baselines are single task algorithms which ignore relevant graphs from similar tasks. Because subgraph features are represented by the connections of some common atoms, there might be some common discriminative structures which are beneficial to relevant learning tasks.

For existing MTL algorithms, regardless of which regularizers are used in the algorithm, they will first mine a set of frequent subgraph as features and then employ multi-task learning techniques for classification. Although these methods enjoy the benefits of MTL by jointly optimizing related learning tasks, they still suffer from severe disadvantages: (1) their

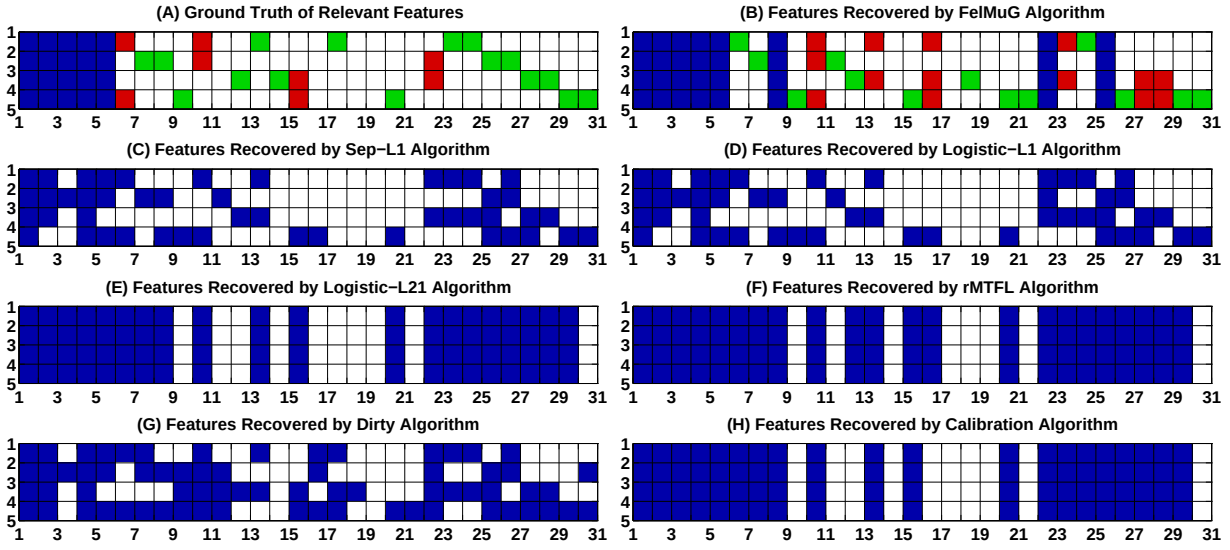

Fig. 3. Relevant features recovered by different algorithms for four tasks with $m = 30$, $d_0 = 5, d_{ta} = 2, d_{ts} = 2$. Blue, red, and green colored cells refer to common features, task auxiliary features, and task specific features, respectively. (A): Ground truth of the effective features; (B): Features learned by FelMuG from $\boldsymbol{\delta}_t$ with $K_0 = 2$, $K_s = K_a = 1$; (C-H): Features discovered by baselines from $\mathbf{W}$. Only FelMuG (B) can explicitly answers which features are shared by all tasks, by set of tasks, or by a single task.

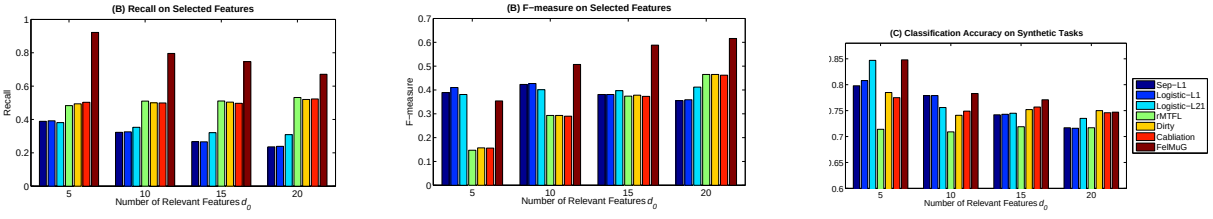

Fig. 4. Feature learning performance on synthetic data with varying d_0. We generate data with $d_{ta} = d_{ts} = d_0/2$, and m=100. FelMuG is learned with $K_0 = K_a = K_t$=2. A Recall value "1" means an algorithm can recover all relevant features, and F-measure with "1" means perfect recovery of all relevant features. The results show FelMuG achieves competitive or better classification accuracy on all tasks (C), and the recovered features are much better than the other algorithms (A) and (B).

subgraph mining process is not driven by multi-task learning objective, and will therefore miss discriminative subgraphs at the first step; (2) they ignore task specific and task auxiliary features for each task, so cannot capture the underlying unique discriminative subgraphs of each task. As the subgraph features are crucial for graph classification, a simple adaptation of existing MTL algorithm for graph classification is suboptimal.

In contrast, the proposed FelMuG method not only has the advantages of utilizing graph samples from relevant tasks, but also unifies multi-task subgraph feature learning and model learning into one objective function. This design helps FelMuG outperform single task graph classification and MTL algorithms with significant performance gains.

Performance on Functional Brain Analysis and PTC Tasks: For functional brain analysis and PTC graph classification tasks, the number of training graphs for each task is very limited. So instead of varying the training samples for each task (such as for NCI tasks), we conduct 10-fold cross-validation on these tasks. In this way, we can reduce the bias of each method caused by limited training samples.

The results in Figures and 6 and 7 show that FelMuG achieves considerable performance gains over single task graph classification and two-step multi-task methods for graph classification. Note that for PTC tasks, AUC values are more important because they are all imbalanced tasks.

6.4 Multi-Task Subgraph Exploration Efficiency

In this subsection, we investigate the efficiency of FelMuG in reducing the search space [Theorem 1 in Sec.4.1] for subgraph feature exploration. Because the search space is infinitely large, it is challenging to assess the pruning effectiveness of FelMuG. Accordingly, we introduce a threshold value min_sup, which denotes the minimum frequency of each qualified subgraph feature in the training graph dataset, to bound the number of subgraphs in the search space. In

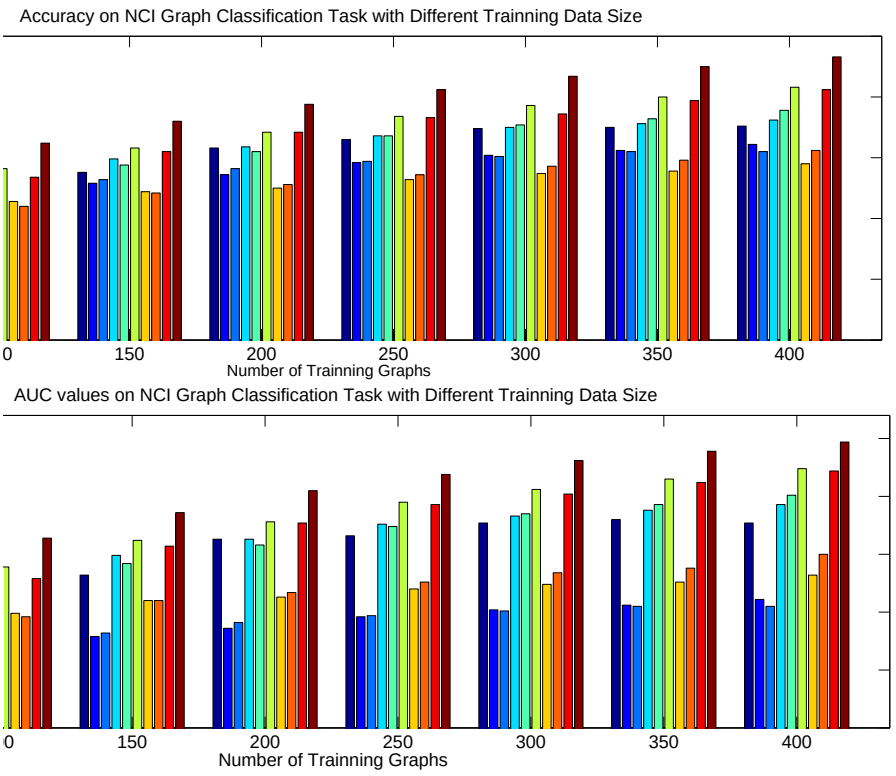

Fig. 5. Average accuracy and AUC values for NCI graph classification tasks with different number of training graphs in each task.

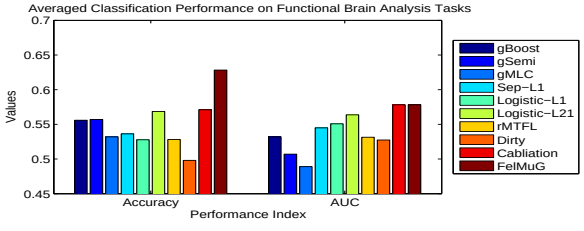

Fig. 6. Average accuracy and AUC values for functional brain analysis tasks.

doing so, we know the total number of subgraph candidates, and can then assess the pruning efficiency by checking the percentage of candidates pruned by the pruning process. The frequent subgraph based algorithm is termed Fre-FelMuG.

The results in Figs. 8.(A) and (C) show that with the increase of the support threshold value min_sup, all methods experience reduced running time. This is because a large support value will result in a small number of subgraph features (Figs. 8.(B) and (D)). It is obvious that our algorithm can reduce unpromising subgraph features significantly while Fre-FelMuG needs to enumerate an exponentially large number of candidates. Our algorithm is an order of magnitude faster than the non-prune baseline.

It is worth noting that using a threshold value min_sup in the subgraph pattern mining may result in missing of discriminative subgraph features, because some subgraph features may be very informative for classification but are not frequent to meet the support threshold value. However, discarding the support threshold value (*i.e.*, $min\_sup = 0$) will make most algorithms unable to find subgraph patterns. For example, in our experiments, we have tried to further reduce the support threshold min_sup for Fre-FelMuG, but it caused an out-of-memory error on a 16 GB memory machine for NCI tasks. In comparison, our FelMuG algorithm is able to mine discriminative subgraphs very quickly, even if the support threshold is removed (*i.e.*, $min\_sup = 0$).

7 Related Work

Our work is closely related to graph classification and multi-task learning.

Graph Classification. Existing methods for graph classification [46], [8], [10], [47], [11], [17], [48], [49] can be roughly distinguished into two groups: kernel based methods and subgraph feature based methods.

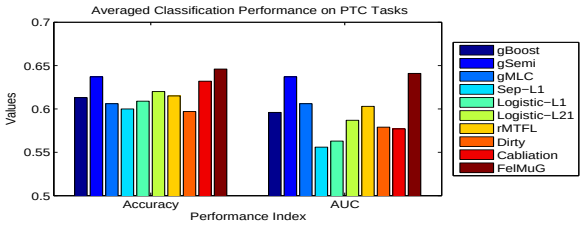

Fig. 7. Average accuracy and AUC values for PTC graph classification tasks.

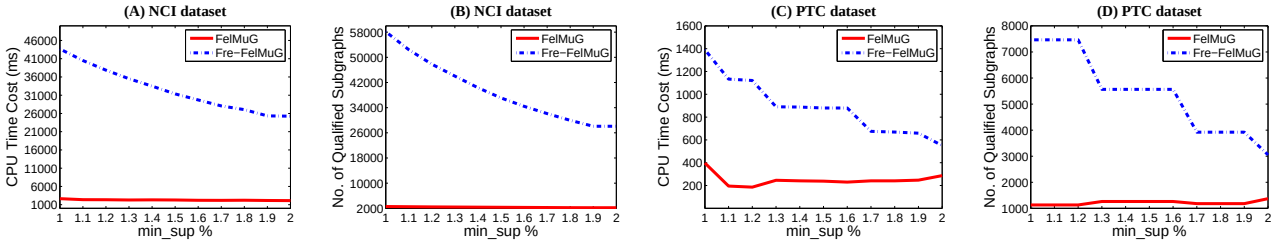

Fig. 8. Pruning efficiency on NCI and PTC graph classification Tasks.

The kernel based approaches aim to directly learn global similarities between graphs by using some graph kernels [47], [6], [7]. Then global similarities are fed into similarity based classifiers, such as KNN or SVM, for learning. One obvious drawback of global similarity based approaches is that the similarity is calculated based on the global graph structures directly, such as random walks or embedding space. Therefore, it is not clear which substructures are mostly important for classifying graphs between different classes.

In many graph classification domains, such as molecule classification, graphs from one specific class may have low global similarities but they actually share some unique substructures. Therefore, using discriminative substructure as features and transferring graphs into vector space becomes a popular solution. For subgraph based methods, a key issue is to define a proxy measurement to assess the utility of each subgraph. Yan *et al.* proposed [50] a LEAP algorithm to exploit the correlation between structural similarity and significance similarity, so that a branch-and-bound rule can be derived to prune out unpromising searching space efficiently. Ranu and Singh [51] proposed a scalable GraphSig algorithm, which is able to mine the significant subgraph with low frequencies. Thoma et al. [9] propose a CORK algorithm, in combination with frequent subgraph mining algorithm such as gSpan [42], to find subgraph features. Instead of carrying out explicit subgraph feature mining, a recent work [46] proposes to find class-conditional "signal-subgraph" which is defined as the connection of edges that are probabilistically different between classes. Recently, researchers have also proposed to address complicated graph classification tasks, such as semi-supervised classification [11], [16], multi-label classification [12]. We have extended the graph classification task to multi-view-graph learning [52] and multi-graph classification scenarios [15], [13], [14]. For multi-view-graph learning, an object consists of multiple views each is represented as a graph structure. For multi-graph classification, the objective is to classify a bag which consists of multiple graphs.

Boosting methods [44], [8], [17], [10] are also popular for graph classification. In [10], the authors proposed to boost the subgraph decision stumps from frequent subgraphs. In [8], the authors proposed a gBoost algorithm which iteratively selects subgraphs from the whole subgraph space (instead of frequent subgraphs only), which has shown better performance for graph classification than using frequent subgraph filter-based methods. Recently, we have extended gBoost algorithm to imbalanced graph classification settings in [17], [18] and cost-sensitive learning for large-scale graph classification [19].

The aforementioned graph classification methods, regardless similarity based methods or subgraph based approaches, only consider one single task, thus they are ineffective for multi-task settings where several graph classification tasks are related to each other.

Multi-Task Learning. State-of-the-art algorithms on multi-task learning [34], [24], [53], [45], [54], [29], [55] can also be roughly divided into two categories: (1) Regularized multi-task feature learning methods [34], [24], [45], which assume all tasks are homogeneous and the learning is to discover common feature representation across all tasks. (2) Task relationship exploration methods [53], [54], [29], [56], which either exploit task relationships via trace norm regularization to achieve some similar parameters among similar tasks [56], or try to learn a task covariance matrix from data if the task relationship is not known in advance [53], [29].

Note that multi-task learning is closely related to transfer learning [57], but the difference is fundamental. Transfer learning aims to improve the learning on a single target task by using data from other tasks as auxiliary information. For

multi-task learning, all tasks are equally important and should be learned simultaneously. A recent work [58] address transfer learning for graph databases, but its scope and objective are different from the proposed MTG.

8 Conclusion

In this paper, we formulated a new multi-task graph learning problem. We argued that existing multi-task learning algorithms are inapplicable to graphs, mainly because they cannot handle structured data, and they cannot explicitly capture relationships between tasks and individual features, which are crucial for classifying and understanding graph tasks. Accordingly, we proposed, FelMuG, a task sensitive feature exploration and learning algorithm for multi-task graph classification. The unique feature of FelMuG lies on its task sensitive feature exploration and learning module, which explicitly categorizes each subgraph feature into three categories: common feature, task auxiliary feature, and task specific feature. The learned features and multiple tasks are iteratively co-regularized to form a multi-task graph classification model with a global optimization goal. Experiments on synthetic and real-world data confirm that (1) FelMuG can accurately capture feature relationships to tasks; (2) cross task subgraph feature exploration and learning can effectively discover discriminative subgraph features for learning; and (3) FelMuG outperforms all baselines for real-world multi-task functional brain analysis and chemical compound classification.

References

[1] O. Ivanciuc, "Chemical graphs, molecular matrices and topological indices in chemoinformatics and quantitative structure-activity relationships," *Current Computer Aided Drug Designs*, vol. 9, no. 2, pp. 153–163, 2013.

[2] J. Richiardi, S. Achard, H. Bunke, and D. Ville, "Machine learning with brain graphs," *IEEE Signal Processing Magazine*, vol. 30, no. 4, 2013.

[3] X. Kong and P. Yu, "Brain network analysis: a data mining perspective," *KDD Exploration*, vol. 15, 2013.

[4] Y. Park, D. S. Reeves, and M. Stamp, "Deriving common malware behavior through graph clustering," *Computers and Security*, vol. 39, no. B, 2013.

[5] S. Bleik, M. Mishra, J. Huan, and M. Song, "Text categorization of biomedical data sets using graph kernels and a controlled vocabulary," *IEEE/ACM Trans. on Computational Biology adn Bioinformatics*, vol. 10, no. 5, 2013.

[6] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *JMLR*, vol. 12, pp. 2539–2561, 2011.

[7] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *JMLR*, vol. 11, 2010.

[8] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda, "gboost: a mathematical programming approach to graph classification and regression," *Machine Learning*, vol. 75, pp. 69–89, 2009.

[9] M. Thoma, H. Cheng, A. Gretton, J. Han, H. Kriegel, A. Smola, L. Song, P. Yu, X. Yan, and K. Borgwardt, "Near-optimal supervised feature selection among frequent subgraphs," in *Proc. of SIAM Data Mining*, USA, 2009.

[10] H. Fei and J. Huan, "Boosting with structure information in the functional space: an application to graph classification," in *ACM SIGKDD*, Washington DC, USA, 2010.

[11] X. Kong and P. Yu, "Semi-supervised feature selection for graph classification," in *ACM SIGKDD*, 2010.

[12] X. Kong and P. S. Yu, "Multi-label feature selection for graph classification," in *ICDM*, 2010, pp. 274–283.

[13] J. Wu, X. Zhu, C. Zhang, and P. Yu, "Bag constrained structure pattern mining for multi-graph classification," *IEEE TKDE*, vol. 26, no. 10, 2014.

[14] J. Wu, Z. Hong, S. Pan, X. Zhu, C. Zhang, and Z. Cai, "Multi-graph learning with positive and unlabeled bags," in *Proc. of SIAM Data Mining*. SIAM, 2014, pp. 1586–1592.

[15] J. Wu, S. Pan, X. Zhu, and Z. Cai, "Boosting for multi-graph classification," *IEEE Transactions on Cybernetics*, 2014.

[16] S. Pan, X. Zhu, C. Zhang, and P. S. Yu, "Graph stream classification using labeled and unlabeled graphs," in *Proc. of ICDE*. IEEE, 2013.

[17] S. Pan and X. Zhu, "Graph classification with imbalanced class distributions and noise," in *IJCAI*, 2013.

[18] S. Pan, J. Wu, X. Zhu, and C. Zhang, "Graph ensemble boosting for imbalanced noisy graph stream classification," *Cybernetics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[19] S. Pan, J. Wu, and X. Zhu, "Cogboost: Boosting for fast cost-sensitive graph classification," *IEEE TKDE*, 2015.

[20] R. Craddock, C. James, P. Holtzheimer, X. Hu, and H. Mayberg, "A whole brain fmri atlas generated via spatially constrained spectral clustering," *Human Brain Mapping*, vol. 33, 2012.

[21] A. Teneva, S. Markovska-Simoskab, L. Kocarevb, J. Pop-Jordanovb, A. Mullerc, and G. Candrianc, "Machine learning approach for classification of adhd adults," *International Journal of Psychophysiology*, vol. 93, 2014.

[22] J. Vogelstein, W. Roncal, R. Vogelstein, and C. Priebe, "Graph classification using signal-subgraphs: Applications in statistical connectomics," *IEEE Trans. on PAMI*, vol. 35, 2013.

[23] P. Anchuri, M. Zaki, O. Barkol, S. Golan, and M. Shamy, "Approximate graph mining with label costs," in *ACM SIGKDD*, 2013, pp. 518–526.

[24] A. Evgeniou and M. Pontil, "Multi-task feature learning," in *NIPS*, vol. 19. The MIT Press, 2007, p. 41.

[25] J. Liu, S. Ji, and J. Ye, "Multi-task feature learning via efficient l 2, 1-norm minimization," in *Proc. of UAI*. AUAI Press, 2009, pp. 339–348.

[26] A. Jalali, S. Sanghavi, C. Ruan, and P. K. Ravikumar, "A dirty model for multi-task learning," in *NIPS*, 2010.

[27] P. Gong, J. Ye, and C. Zhang, "Robust multi-task feature learning," in *ACM SIGKDD*. ACM, 2012, pp. 895–903.

[28] P. Gong, J. Zhou, W. Fan, and J. Ye, "Efficient multi-task feature learning with calibration," in *ACM SIGKDD*. ACM, 2014, pp. 761–770.

[29] Y. Zhang and D.-Y. Yeung, "A convex formulation for learning task relationships in multi-task learning," in *Proc. of UAI*, 2010, pp. 733–442.

[30] W. Zhong and J. T. Kwok, "Convex multitask learning with flexible task clusters," in *Proc. of ICML*, 2012, pp. 49–56.

[31] A. Kumar and H. Daume, "Learning task grouping and overlap in multi-task learning," in *Proc. of ICML*, 2012, pp. 1383–1390.

[32] B. Romera-Paredes, A. Argyriou, N. Berthouze, and M. Pontil, "Exploiting unrelated tasks in multi-task learning," in *AI Statistics*, 2012, pp. 951–959.

[33] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis, "Frequent substructure-based approaches for classifying chemical compounds," *IEEE TKDE*, pp. 1036–1050, 2005.

[34] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *ACM SIGKDD*. ACM, 2004, pp. 109–117.

[35] M. Tan, I. W. Tsang, and L. Wang, "Towards ultrahigh dimensional feature selection for big data," *JMLR*, vol. 15, no. 1, pp. 1371–1429, 2014.

[36] S.-J. Kim and S. Boyd, "A minimax theorem with applications to machine learning, signal processing, and finance," *SIAM J. on Optimization*, vol. 19, no. 3, pp. 1344–1367, 2008.

[37] P. Liu, J. T. Zhou, I. W.-H. Tsang, Z. Meng, S. Han, and Y. Tong, "Feature disentangling machine-a novel approach of feature selection and disentangling in facial expression analysis," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 151–166.

[38] A. Rakotomamonjy, R. Flamary, G. Gasso, and S. Canu, "lp-lq penalty for sparse linear and sparse multiple kernel multitask learning," *Neural Networks, IEEE Transactions on*, vol. 22, no. 8, pp. 1307–1320, 2011.
[39] J. E. Kelley, Jr, "The cutting-plane method for solving convex programs," *J. of the Soc. for Indus. & Appl. Math.*, vol. 8, no. 4, pp. 703–712, 1960.
[40] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "Simplemkl," *JMLR*, vol. 9, pp. 2491–2521, 2008.
[41] L. A. Wolsey, *Integer programming*. Wiley New York, 1998, vol. 42.
[42] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *Proc. of ICDM*, 2002.
[43] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *JMLR*, vol. 9, pp. 1871–1874, 2008.
[44] T. Kudo, E. Maeda, and Y. Matsumoto, "An application of boosting to graph classification." in *Advances in Neural Information Processing Systems (NIPS)*, 2004.
[45] J. Zhou, J. Chen, and J. Ye, "Malsar: Multi-task learning via structural regularization," *Arizona State Univ*, 2012.
[46] J. Vogelstein, W. Roncal, R. Vogelstein, and C. Priebe, "Graph classification using signal-subgraphs: Applications in statistical connectomics," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 7, pp. 1539–1551, 2013.
[47] H. Kashima, K. Tsuda, and A. Inokuchi, *Kernels for Graphs*. Cambridge (Massachusetts): MIT Press, 2004, ch. In: Schlkopf B, Tsuda K, Vert JP, editors. Kernel methods in computational biology.
[48] N. Jin, C. Young, and W. Wang, "GAIA: graph classification using evolutionary computation," in *ACM SIGMOD*. ACM, 2010, pp. 879–890.
[49] ——, "Graph classification based on pattern co-occurrence," in *Proc. of ACM CIKM*, 2009.
[50] X. Yan, H. Cheng, J. Han, and P. S. Yu, "Mining significant graph patterns by leap search," in *ACM SIGMOD*. ACM, 2008, pp. 433–444.
[51] S. Ranu and A. Singh, "Graphsig: A scalable approach to mining significant subgraphs in large graph databases," in *Proc. of ICDE*. IEEE, 2009, pp. 844–855.
[52] J. Wu, Z. Hong, S. Pan, X. Zhu, and C. Zhang, "Multi-graph-view learning for graph classification," in *International Conference on Data Mining*, 2014, pp. 590–599.
[53] Y. Zhang and D.-Y. Yeung, "Multi-task boosting by exploiting task relationships," in *ECML/PKDD*. Springer-Verlag, 2012, pp. 697–710.
[54] H. Fei and J. Huan, "Structured feature selection and task relationship inference for multi-task learning," *Knowledge and information systems*, vol. 35, no. 2, pp. 345–364, 2013.
[55] Z. Hong, X. Mei, D. Prokhorov, and D. Tao, "Tracking via robust multi-task multi-view joint sparse representation," in *Proc. of ICCV*. IEEE, 2013, pp. 649–656.
[56] B. Bakker and T. Heskes, "Task clustering and gating for bayesian multitask learning," *JMLR*, pp. 83–99, 2003.
[57] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, pp. 1345–1359, 2010.
[58] X. Shi, X. Kong, and P. S. Yu, "Transfer significant subgraphs across graph databases," in *Proceedings of the SIAM Data Mining*, 2012, pp. 552–563.