Task Sensitive Feature Exploration and Learning for Multitask Graph Classification

Shirui Pan, Jia Wu, Xingquan Zhu, *Senior Member, IEEE*, Guodong Long, and Chengqi Zhang, *Senior Member, IEEE*

***Abstract*—Multitask learning (MTL) is commonly used for jointly optimizing multiple learning tasks. To date, all existing MTL methods have been designed for tasks with feature-vector represented instances, but cannot be applied to structure 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 important questions, such as what exactly is shared between tasks and what is the uniqueness of one task differing from others? In this paper, we formulate a new multitask graph learning problem, and propose a task sensitive feature exploration and learning algorithm for multitask graph classification. Because 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 feature learning process is carried out to categorize each subgraph feature into one of three categories: 1) common feature; 2) task auxiliary feature; and 3) 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 feature learning and the multiple task learning are iteratively optimized to form a multitask 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*—Feature selection, graph classification, multitask learning (MTL), subgraph mining, supervised learning.**

I. Introduction

GRAPH classification has become increasingly important in recent years due to the rapid growth of complex data with structural and interdependent relationships. For many applications, such as chemical compound categorization [1], functional brain analysis [2], [3], malware detection [4], and biomedical document classification [5], there is an immediate need to automatically classify data with structural information into meaningful categories.

To classify graphs, the key challenge lies in the fact that no feature is readily available for learning algorithms to derive classification model. This challenge has motivated numerous methods of representing graphs in a suitable format for learning, including: 1) kernel-based algorithms [6], [7], which learn kernels to measure the similarity between graph objects, so that a pair-wise similarity matrix can be fed into learning algorithms such as a support vector machine (SVM) for learning and 2) subgraph-based algorithms [8]–[15], which aim to discover discriminative subgraph features to represent graphs into vector space, so that generic machine learning algorithms can be applied.

A. Multitask Graph Classification: Motivation

Although graph classification has drawn significant attentions, existing methods typically share two major deficiencies in their designs: 1) in order to explore subgraph structures for training good classification models, a large number of training graphs are required and 2) they can only work on a single learning task. In reality, due to the inherent complexity of the graph data and the costs involved in the labeling process, collecting a large number of labeled graphs for a specific task is difficult. However, it is quite common that multiple similar graph classification tasks, each having a small number of training samples, may co-exist and need to be handled. Two motivating examples are given as follows.

Functional brain analysis aims to map human brain as a network (or a graph) to model correlations between diseases and functions of brain regions [16]. In order to carry out a specific learning task, such as diagnosing attention deficit hyperactivity disorder (ADHD) [17], each object needs to go through functional magnetic resonance imaging (fMRI) and intensive data preprocessing to collect training data. This severely limits each task to have a maximum of only a couple of hundred objects. On the other hand, institutions may have data collected for different but relevant learning tasks, such as gender [18] or Alzheimer's disease study. The limited samples for each individual tasks, and the commonality between tasks raise an interesting question as to whether multiple brain function classification tasks can be combined to

Manuscript received May 15, 2015; revised September 29, 2015; accepted January 26, 2016. Date of publication March 10, 2016; date of current version February 14, 2017. This work was supported by the Centre of Excellence for Environmental Decisions, Australian Research Council Discovery Projects under Grant DP140100545, Grant DP140102206, and Grant LP120100566. This paper was recommended by Associate Editor J. Basak. *(Corresponding author: Jia Wu.)*

S. Pan, J. Wu, G. Long, and C. Zhang are with the Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: shirui.pan@uts.edu.au; jia.wu@uts.edu.au; guodong.long@uts.edu.au; chengqi.zhang@uts.edu.au).

X. Zhu is with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: xzhu3@fau.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2016.2526058

learn a multitask model for maximum performance gain for all tasks.

Chemical compound categorization is important in biomedical research for testing whether a compound is active to a specific cancer, such as melanoma. In melanoma cancer, determining activities of a molecule is expensive, as it requires time, efforts, and expensive resources [19] to conduct a biological assay. In reality, some similar bioassay tasks,[1] such as anti-cancer test for prostate, may be available. As graph data for different types of cancer may share common substructures, learning multiple related tasks concurrently may potentially help improve the generalization performance of each single task.

B. Existing MTL for Graph Data: Weakness

Indeed, existing research on multitask learning (MTL) 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) Multitask feature learning, which explores common feature space shared by all tasks. These models, including mixed $\ell_{2,1}$ norm sparsity inducing methods [20], [21], composite regularized algorithms [22], [23], and the most recent calibration-based multitask approach [24], can be formulated as a regularized loss minimization problem aiming to explore shared feature space among tasks for learning.
2) Task relationship learning, which simultaneously exploits task relationships and parameters [25], such as task clustering [26], [27] or isolating [28], 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 only work on data with feature-vector representation, but cannot be directly applied to structure data, such as graphs. In graph classification, one needs to first find subgraph features to represent graphs into vector space. To apply MTL to graph classification, one simple solution is to first mine a set of frequent subgraphs as features, and then employ state-of-the-art MTL algorithms [20], [24]. Unfortunately, this simple adaption is far from optimal because there is an exponentially large number of subgraph features so one has to use a threshold to limit the number of frequent subgraphs. As a result, MTL is only carried out on a reduced subgraph space (frequent subgraphs), and some genuine discriminative subgraphs that are infrequent may be missed. This leads to a deterioration of classification performance. The ineffectiveness of this two-step approach for graph classification calls for effective MTL algorithms that can explore the exponentially large subgraph space.

Another drawback of exiting MTL methods is that they mainly aim to exploit the commonality in feature space or task correlations, but ignore the uniqueness of individual tasks. This is potentially harmful for graph classification domains, because graphs from similar domains usually share high global similarity and only differ in a small set of substructures. These, however, are crucial for model learning and should be carefully preserved. More importantly, these unique features are helpful for users to uncover 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 discover features unique to a specific cancer. Existing MTL methods, unfortunately, mainly focus on exploring overall commonality or disparity between tasks, but cannot explicitly capture detailed relationships between tasks with respect to their individual features. An example is shown in Fig. 1 where three types of cancer diagnosis tasks share some common subgraph features for all tasks (first column). Some features are shared by a subset of tasks (second column), and some subgraph features are unique for each individual task (third column).

C. Proposed Algorithm: Novelty and Contributions

Motivated by the above observations, in this paper, we propose a task sensitive feature exploration and learning algorithm for multitask graph (FelMuG) classification. Two key features that differentiate FelMuG from existing MTL are: 1) exponential feature space exploration and 2) task sensitive feature learning. In order to explore exponential subgraph feature space, we derive an effective pruning bound to reduce unpromising candidates, so that all discriminative subgraphs can be discovered to help improve the classification performance.

A unique feature of FelMuG is its task sensitive feature learning module which automatically learns and categorizes 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 MTL but also enhances the understanding of task relationships at the feature level. In the final stage, the learned subgraph features and graph classification tasks are iteratively optimized to form an optimization function, with subgraph exploration and MTL being iteratively optimized for maximum performance gain.

This paper makes noticeable contributions in the following three aspects.

1) *Feature Learning and Categorization for MTL:* 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 uniqueness of different tasks at the feature level.
2) *Cross Task Subgraph Exploration:* We derive an effective pruning bound to explore discriminative subgraph features, from the exponential subgraph search space, without requiring any support threshold.
3) *Multitask Graph Classification:* We advance the single task graph classification setting to multitask scenarios, which jointly explore and learn multiple classification

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

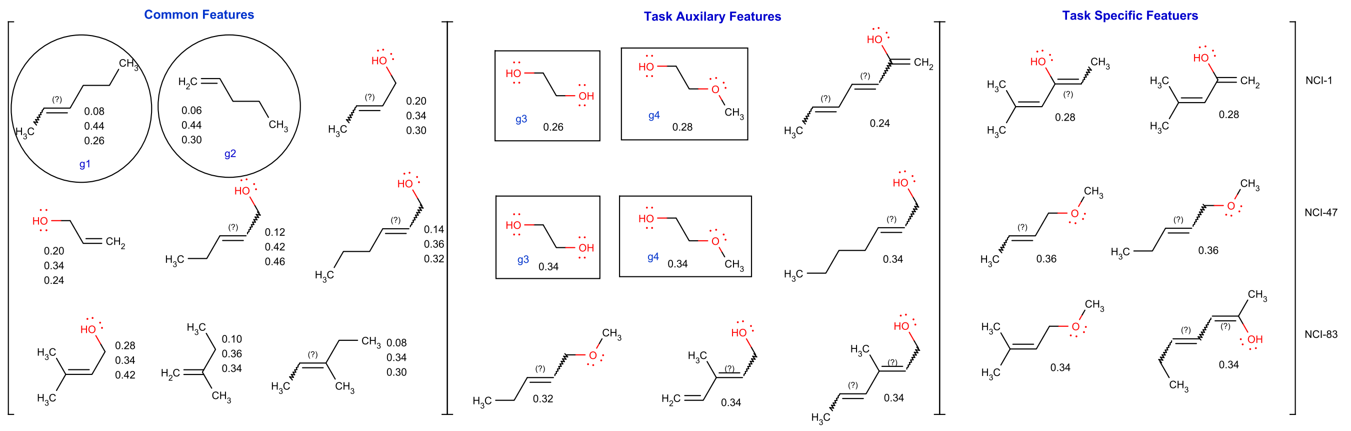

Fig. 1. Feature learning and categorization for three graph classification tasks (detailed in Section VII). The first column shows the top nine subgraphs shared by all tasks (learned by our algorithm). Numeric values next to each subgraph features indicate the feature utility for NCI-1, 47, and 83 tasks (measured by $[(1/n_t)\sum_{i=1}^{n_t} y_i x_i^j]$, which will be derived in Eq. (8), 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.

models to improve the classification performance over single task graph classification.

The remainder of this paper is structured as follows. We review related work in Section II. The problem definition and preliminaries are given in Section III. The task sensitive multitask graph classification formulation is presented in Section IV, followed by the multitask subgraph exploration method in Section V. The time complexity is analyzed in Section VI. Experimental results are described in Section VII, and we conclude the paper in Section VIII.

II. Related Work

This paper is closely related to graph classification and MTL.

A. Graph Classification

Existing methods for graph classification [8], [11], [12], [29]–[34] can be roughly distinguished into two groups: 1) kernel-based methods and 2) subgraph feature-based methods.

Kernel-based approaches aim to directly learn global similarities between graphs by using some graph kernels [6], [7], [30]. The global similarities are then fed into similarity-based classifiers, such as K-nearest neighbors or SVM, for learning. An obvious drawback of global similarity-based approaches is that similarity is calculated based on global graph structures, such as random walks or embedding space. Therefore, it is not clear which substructures are mostly important for classifying graphs between different classes.

In subgraph-based methods, the key issue is defining a measurement to assess the utility of each subgraph. Yan *et al.* [35] proposed an LEAP algorithm to exploit the correlation between structure similarity and significance similarity, with a branch-and-bound rule being derived to prune subgraph space. Ranu and Singh [36] proposed a scalable GraphSig algorithm to mine significant subgraphs with low frequencies. Thoma *et al.* [10] proposed a CORK algorithm to find subgraph features. Kong *et al.* [14] proposed to select subgraph features and instances simultaneously for active learning. In [15], discriminative subgraph feature selection for PU learning is studied. Our method is similar to these methods [10], [14], [15], [35], [36] in the sense that we use the gSpan algorithm [37] to explore exponential subgraph space and derive effective pruning bounds to prune unpromising subgraph candidates. However, the difference between FelMuG and these methods is fundamental. From a feature selection perspective, existing methods [10], [14], [15], [35], [36] are filter-based whereas FelMuG is an embedding-based algorithm. The filter-based graph classification methods first mine a set of subgraphs as features and transfer graphs into vector representation. It then learns a traditional classifier (e.g., SVM) for graph classification. One possible drawback is that the selected subgraph features may not fit the classification model very well. Our embedding-based graph classification aims to simultaneously select features and learn the classification model. As a result, the selected subgraph features are directly customized to fit the classification model for better classification results.

In line with embedding subgraph-based graph classification approaches, boosting-style algorithms [8], [31], [38]–[40] are very popular. In [38], an Adaboost style algorithm was proposed and later extended to a linear programing boosting algorithm in [8]. Some boosting algorithms are designed to handle imbalanced graph classification [31], [39] and cost-sensitive graph classification [40]. Most recently, an regularized loss minimization-driven algorithm [9] was proposed to minimize the regularized loss function for graph classification. These algorithms [8], [9], [39], [40] iteratively select subgraph features to reoptimize an objective function, demonstrating superior performance to filter-based algorithms.

Recently, researchers have also addressed complicated graph classification tasks, such as semi-supervised classification [12], [41] and multilabel classification [13]. We have extended graph classification task to multiview-graph learning [42], [43] and multigraph classification scenarios [44]–[46]. In multiview-graph learning, an object consists of multiple graph structure views. For multigraph classification, the objective is to classify a bag that consists of multiple graphs.

For above graph classification methods, regardless of similarity-based methods or subgraph-based approaches, they can only handle one learning task, and are, therefore, ineffective or inapplicable for multitask settings where multiple related graph classification tasks co-exist for learning.

B. Multitask Learning

State-of-the-art algorithms on MTL [20], [25], [47]–[51] can also be roughly divided into two categories.

1) Regularized multitask feature learning methods [20], [47], [49], which assume all tasks are homogeneous and the learning is to discover common feature representation across all tasks.
2) Task relationship exploration methods [25], [48], [50], [52], which either exploit task relationships via trace norm regularization to achieve some similar parameters among similar tasks [52], or try to learn a task covariance matrix from data if the task relationship is unknown in advance [25], [48].

Note that MTL is closely related to transfer learning [53], 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. In MTL, all tasks are equally important and should be optimized simultaneously. A recent work [54] addresses transfer learning for graph databases, but its scope and objective are different from the proposed FelMuG.

III. Definitions and Preliminaries

A. Problem Definition

Definition 1 (Connected Graph): A graph is denoted by $G = (\mathcal{V}, E, L)$, where $\mathcal{V} = \{v_1, \dots, v_n\}$ is a set of vertices, $E \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, and L is a labeling 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 [55].

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 $f : \mathcal{V}' \to \mathcal{V}$, 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 ($G \supseteq g_k$).

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

B. Preliminaries

1) Single Task Graph Classification: To support graph classification, state-of-the-art algorithms [8], [11] use a set of subgraphs explored from training graphs as features. Let $\mathcal{F} = \{g_1, \dots, g_m\}$ be the full set of subgraphs in $\mathcal{G}$. Each subgraph $g_k \in \mathcal{F}$ can map a given graph $G_{t,i}$ to the class label space $\mathcal{Y} = \{+1, -1\}$ by using a simple decision stump as follows:

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

Here $I(a) = 1$ if a holds, or 0 otherwise. The rule simply maps a graph $G_{t,i}$ as a numeric feature value $+1$ if subgraph g_k appears in $G_{t,i}$, i.e., $g_k \in G_{t,i}$, or -1 otherwise.

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

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

where $\boldsymbol{w}_t = [\boldsymbol{w}_{t,1}, \dots, \boldsymbol{w}_{t,m}]^T$ 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 the 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). Section V proposes solutions which integrate this algorithm into subgraph mining process to explore subgraph features $\mathcal{F}$ for general graph tasks.

IV. Multitask Graph Classification

In this section, we first formulate a novel task sensitive feature learning algorithm for multitask classification (Section IV-A). Because the formulated problem is a mixed integer problem (MIP), we relax it to a convex semi-infinite problem (SIP) in Section IV-B. Since the resulting SIP relaxation has infinite constraints, we further propose an advanced cutting plane optimization algorithm in Section IV-C to solve the problem.

A. Task Sensitive Feature Learning for Multitask Learning

When applying traditional SVMs to a learning task t, 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:

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

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, n_t is the number of training samples of task t, and C is a parameter controlling the regularization part.

Given multiple tasks, 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 there is no feature immediately available to represent graphs. Instead, one needs 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 nonoverlapping 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 exclusively used by a single 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 nonoverlapping groups, the three groups impose hard constraints to the features. Specifically, for each task we introduce three feature scaling vectors, with $\boldsymbol{\delta}_0 = [\delta_0^1, \ldots, \delta_0^m] \in \{0, 1\}^m$ corresponding to common features, $\boldsymbol{\delta}_{\text{ts}} = [\delta_{\text{ts}}^1, \ldots, \delta_{\text{ts}}^m] \in \{0, 1\}^m$ for task specific features, and $\boldsymbol{\delta}_{\text{ta}} = [\delta_{\text{ta}}^1, \ldots, \delta_{\text{ta}}^m] \in \{0, 1\}^m$ for task auxiliary features, respectively.

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

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

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

$$\begin{aligned} \|\boldsymbol{\delta}_0\|_1 &= \sum_{j=1}^{m} \delta_0^j \le K_0, \delta_0^j \in \{0, 1\} \\ \|\boldsymbol{\delta}_{\text{ts}}\|_1 &= \sum_{j=1}^{m} \delta_{\text{ts}}^j \le K_s, \delta_{\text{ts}}^j \in \{0, 1\} \\ \|\boldsymbol{\delta}_{\text{ta}}\|_1 &= \sum_{j=1}^{m} \delta_{\text{ta}}^j \le K_a, \delta_{\text{ta}}^j \in \{0, 1\} \\ \delta^j &= \delta_0^j + \delta_{\text{ta}}^j + \delta_{\text{ts}}^j \le 1, \forall j \\ \sum_{t=1}^{T} &\delta_{\text{ts}}^j \le 1, \forall j \end{aligned} \tag{5}$$

where K_0, K_s, and K_a are integers indicating the least number of features used in the final models. In biological applications, due to expensive bio-diagnosis and limited resources, biologists prefer to select a relatively small number of genes, such as less than 100, from thousands of genes [56], [57]. For graph classification, we also prefer to select a small number of discriminative subgraphs, for further analysis. The forth constraint, $\delta^j = \delta_0^j + \delta_{\text{ts}}^j + \delta_{\text{ta}}^j$ is the sum of $\boldsymbol{\delta}_0$, $\boldsymbol{\delta}_{\text{ts}}$, and $\boldsymbol{\delta}_{\text{ta}}$ on the jth dimension. $\delta^j \le 1$ enforces that the jth feature belongs to only one group. 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:

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

where $\mathcal{D} = \{\boldsymbol{\delta}_0 + \boldsymbol{\delta}_{\text{ta}} + \boldsymbol{\delta}_{\text{ts}}\}$, and $\boldsymbol{\delta}_0$, $\boldsymbol{\delta}_{\text{ta}}$, and $\boldsymbol{\delta}_{\text{ts}}$ are subject to (5).

1) Merit of Our Design: We use $\boldsymbol{\delta}_0$, $\boldsymbol{\delta}_{\text{ta}}$, and $\boldsymbol{\delta}_{\text{ts}}$ to directly learn and categorize features, with learned features being used to characterize the optimization model. Evgeniou and Pontil [47] decomposed the weight as $\boldsymbol{w}_t = \boldsymbol{w}_0 + \boldsymbol{v}_t$; and existing composite regularization methods such as dirty model [22] and rMTFL method [23] 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 learned model will be biased and under-fit the training data, resulting poor performance. This is attributed to the bias of ℓ_1 norm regularization effects [58].

Compared to the state-of-the-art multitask feature learning methods, the merit of our design is fourfold.

1) It naturally selects features 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. A similar scheme has been used in [58], which is, in fact, a special case of our MTL 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 (6) is an MIP, which is nonconvex when considering $\mathbf{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t]$ and $\boldsymbol{\delta}_t$ together. As a result, this problem is computationally intractable if solved directly. Next, we will relax this formula to a convex problem.

B. Convex Relaxation of MTL

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

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

where $\mathcal{A}=\{\alpha_{t,i}|\sum_{i=1}^{n_t}\alpha_{t,i}y_{t,i}=0,\forall t\in[1,T];0\le\alpha_{t,i}\le C\}$. For convenience, let $s_{t,j}$ be the feature score of the first term on the jth dimension of task t,[2] that is

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

Let $s_j=\sum_{t=1}^{T}s_{t,j}$ be the feature score on the jth dimension over all tasks, then the first term of (7) can be rewritten as the sum of feature score over all features, that is

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

where $\delta^j=\delta_0^j+\delta_{\text{ts}}^j+\delta_{\text{ta}}^j$. 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 let

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

The original objective function (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 [59], we have the following lower bound to (10):

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

So instead of solving (6) or (10), which is computationally intractable, we can solve the following relaxed problem:

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

Moreover, this relaxed problem can be further transferred to an SIP. Motivated by [58] and [60], we introduce another variable $\omega\in R$, so the relaxed problem can be formulated as the following SIP:

$$\max_{\boldsymbol{\alpha}\in\mathcal{A},\omega\in R}\omega:\ \ \omega\le F(\boldsymbol{\alpha},\boldsymbol{\delta}),\forall\boldsymbol{\delta}\in\mathcal{D}. \tag{12}$$

Equation (12) is a convex quadratically constrained quadratic programming, where each $\boldsymbol{\delta}\in\mathcal{D}$ defines a quadratic constraint with respect to $\boldsymbol{\alpha}$. The main challenge to solve this problem lies in the fact that there are an infinite number of constraints, i.e., the number of elements in $\mathcal{D}$ is infinitely large. To solve this challenge, we turn to its dual form, based on which the problem is formulated as a multikernel learning problem [61], [62], so that a cutting plane algorithm can be derived, and exiting optimization toolbox can be used to solve the multikernel problem.

[2]The feature utility in Fig. 1 is a simplified version of (8), i.e., $\sqrt{s_{t,j}}=[(1/n_t)\sum_{i=1}^{n_t}y_i x_i^j]$, with $\alpha_{t,i}=1/n_t$.

Algorithm 1 Task Sensitive FelMuG Classification

```
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.
```

1) From MTL to Multikernel Learning: Introducing another set of Lagrangian variables $\boldsymbol{\mu}=\{\mu_k\}$, we will have the new Lagrangian function, that is

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

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

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

where we have $\boldsymbol{X}_{t,k}=[\boldsymbol{x}_{t,i}\odot\boldsymbol{\delta}^{(k)},\ldots,\boldsymbol{x}_{t,n_t}\odot\boldsymbol{\delta}^{(k)}]$, and $\boldsymbol{\alpha}_{t,\cdot}$ and $\boldsymbol{y}_{t,\cdot}$ are the Lagrangian multiplier and training class labels for task t. Accordingly, the problem is also a convex optimization problem, whose global optimal can be found. More specifically, it is a multikernel multitask (MKMT) problem [62], where $\boldsymbol{X}_{t,k}\boldsymbol{X}_{t,k}^T$ can be seen as a kernel defined on a subset of features $\boldsymbol{\delta}^{(k)}$ on the tth 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.

C. Cutting Plane for Infinite Constraint Optimization

The main challenge to solve MKMT problem (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 the cutting plane algorithm [63]. The idea of cutting plane algorithm is to start with an empty working set $\mathcal{C}$ and iteratively select the most violated constraint $\boldsymbol{\delta}^{(k)}$ to be included into the working set $\mathcal{C}$, and resolve the reduced problem of (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 lists detailed 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 key steps of the 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 subproblems.

1) Solving the MKMT Subproblem: MKMT learning was studied recently in [62]. It 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 $\mathcal{C}$ on step 3 of Algorithm 1, our subproblem is equal to solving the following reduced MKMT subproblem of (13):

$$\min_{\boldsymbol{\mu}\in\mathcal{M}} \max_{\boldsymbol{\alpha}\in\mathcal{A}} -\frac{1}{2}\sum_{t=1}^{T}(\boldsymbol{\alpha}_{t,\cdot}\odot \boldsymbol{y}_{t,\cdot})^T \mathbf{K}(\boldsymbol{\alpha}_{t,\cdot}\odot \boldsymbol{y}_{t,\cdot}) + \alpha_{t,\cdot}\mathbf{1}$$
$$\text{with} \quad \mathbf{K} = \sum_{\boldsymbol{\delta}^{(k)}\in\mathcal{C}} \mu_k \boldsymbol{X}_{t,k}\boldsymbol{X}_{t,k}^T. \tag{14}$$

The problem in (14) can be effectively solved by using existing MKMT solvers [62]. Specifically, we can solve (14) via an iterative procedure: 1) fixing $\boldsymbol{\mu}$ and solving (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 problems and 2) fixing $\boldsymbol{\alpha}$ and using reduced gradient method [61] to update $\boldsymbol{\mu}$. The two steps continue until they converge.

2) Most Violated Constraint Selection: To find the most violated constraint, we need to refer to (12). Given fixed $\boldsymbol{\alpha}$ in (12), the problem is reduced to

$$\max_{\omega} \omega : \omega \le F(\boldsymbol{\alpha}, \boldsymbol{\delta}), \forall \boldsymbol{\delta} \in \mathcal{D}.$$

Note that there is an infinite number of constraints $\omega \le F(\boldsymbol{\alpha}, \boldsymbol{\delta})$ with different $\boldsymbol{\delta}$. Here $\min_{\boldsymbol{\delta}\in\mathcal{D}} F(\boldsymbol{\alpha}, \boldsymbol{\delta})$ is the key to this problem as if $\omega \le \min_{\boldsymbol{\delta}\in\mathcal{D}} F(\boldsymbol{\alpha}, \boldsymbol{\delta})$ then there is no other $\boldsymbol{\delta}$ which will result in a valid constraint. Therefore, finding the most violated constraint is the same as solving the problem of $\min_{\boldsymbol{\delta}\in\mathcal{D}} F(\boldsymbol{\alpha}, \boldsymbol{\delta})$. Because the second term in (9) is constant given $\boldsymbol{\alpha}$, the problem is reduced to $\max_{\boldsymbol{\delta}} \sum_{j=1}^{m} \delta^j s_j$. Thus the problem can be formulated as

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

Accordingly, the problem becomes a binary and linear programming problem, i.e., the knapsack problem [64]. If the features are known, many methods such as dynamic programming or 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.

Algorithm 2 Most Violated Constraint Selection for Exponentially Large Subgraph Space

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

a) Challenge for graph data: Equation (15) still faces a major technical barrier with graph data, because: 1) subgraph feature set $\boldsymbol{x}_{t,i} = [\hbar_{g_1}(G_{t,i}), \ldots, \hbar_{g_m}(G_{t,i})]^T$ is unknown and 2) the number of subgraph features is exponentially large (or infinite). In the next section, we seamlessly integrate this problem into the subgraph exploration process to explore subgraph features for graph classification.

V. Multitask Subgraph Exploration

For graph classification, finding the most violated constraint, i.e., solving (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 a multitask 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 exponentially large subgraph problem. Our idea is to employ the subgraph mining algorithm gSpan [37] to mine a small set of potential subgraph features $\mathcal{F}_p$, and then solve (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 (15). As $\mathcal{F}_p$ is very small, (15) can be solved efficiently.

A. Multitask Subgraph Selection

Although we cannot directly obtain 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 $s_{t,j} = [\sum_{i=1}^{n_t} \alpha_{t,i} y_{t,i} \boldsymbol{x}_{t,i}^j]^2$ (8) on task t. Then we can construct a small set of potential subgraphs $\mathcal{F}_p = \cup \mathcal{F}_t$.

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 (15).

This proposition can be assured as we select at most $K = K_0 + \sum_t K_s + \sum_t K_a$ subgraph in (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 (15).

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

1) Top K Subgraph Mining: Discovering the top K subgraphs requires the a search of an exponentially large subgraph space. In this section, 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 tth task

$$A_1(g_j) = 2 \sum_{\{i|y_{t,i}=+1, g_j \in G_{t,i}\}} \alpha_{t,i}$$

$$A_2(g_j) = 2 \sum_{\{i|y_{t,i}=-1, g_j \in G_{t,i}\}} \alpha_{t,i}$$

$$A_3 = \sum_{i=1}^{n_t} \alpha_{t,i} y_{t,i}$$

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

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

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

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

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 exponentially prune/reduce unpromising candidates effectively.

2) Top K Subgraph Mining Algorithm: Our top K 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 4–6. This step involves subgraph isomorphism test. For gSpan algorithm, if two subgraphs have the same minimum depth-first search (DFS) codes, they are identical subgraphs. The discriminative score $s_{t,j}$ for g_j are calculated on step 7. If $s_{t,j}$ is larger than η, we add g_j to the feature set $\mathcal{F}_t$ (steps 8–10). If the size of $\mathcal{F}_t$ exceeds the predefined size K, the subgraph with the minimum discriminative score is removed (steps 11–15), 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 16–18. Finally, the optimal set $\mathcal{F}_t$ is returned on step 20.

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.

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:
    $\mathcal{F}_t = \{g_j\}_{j=1,\ldots,K}$:   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:         $\eta \leftarrow \min_{g_k \in \mathcal{F}_t} \Theta(g_k)$;
15:     end if
16:     if $\hat{\Theta}(g_j, t) > \eta$ then
17:         Depth-first search the subtree rooted from node $g_j$;
18:     end if
19: end while
20: return $\mathcal{F}_t = \{g_j\}_{j=1,\ldots,K}$;
```

VI. Time Complexity Analysis

Our FelMuG algorithm can be applied to general MTL with feature-vector representation as well as graph data, as we will soon demonstrate in our experiments in Section VII. Accordingly, we analyze the time complexity for these two cases.

A. Generic Multitask 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[3] requires approximately $O(n^{2.5})$ with respect to the number of training instances n, or linear $O(\hat{m}n)$ with respect to the number of instances n and selected features $\hat{m}$ by using advanced solvers, such as LIBLinear [65]. Updating $\boldsymbol{\mu}$ requires $O(T \cdot |\mathcal{C}| \cdot n_{t,\text{sv}}^2)$, where $n_{t,\text{sv}}$ denotes the number of support vectors related to task t [62] and $|\mathcal{C}|$ is the total number of selected constraints. For the second subproblem, we use MATLAB function bintprog with polynomial time complexity

[3]For simplicity, we assume all tasks have the same number of training data n.

in our experiments. 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 iterations to solve MKMT problem is S_1, the time complexity of FelMuG for general MTL is

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

Note that both $|C|$ and $n_{t,\text{sv}}$ are very small because only a small number of constraints and a small amount of support vectors are involved. Because cutting plane algorithm is used, S is very small. S_1 is the number of iterations for multikernel learning, numerous studies [61], [62] have shown that it can be finished efficiently. So S_1 is very small as well. In our experiments, only tens of iterations are required to reach convergence. As a result, FelMuG is very efficient for multitask classification.

B. Multitask Graph Classification

For graph data, the time complexity becomes

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

where $O(\text{gSpan})$ 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 in Section VII-D will show the effectiveness of the pruning scheme. Furthermore, other techniques, such as reusing the subgraph space [8], [40], 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, $O(\text{gSpan})$ time complexity is inevitable for subgraph-based graph classification [8], [12].

VII. Experiments

In this section, we first validate the feature learning module of FelMuG on synthetic vector datasets, and then evaluate FelMuGs performance on three real-world domains for multitask graph classification. The source code of FelMuG algorithm and the benchmark graph datasets are available online.[4]

A. Experimental Settings

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 ground truth feature relationships, so we can validate how effective FelMuG can capture/retrieve the predefined relationships. For real-world graphs, we demonstrate that FelMuGs feature learning and multitask classification achieve much better performance than its peers for functional brain analysis and chemical compound classification.

Synthetic vector data are modified from the one used in [62] 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 $d_t = d_0 + d_{\text{ts}} + d_{\text{ta}}$ 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 $m - d_t$ noneffective features follow an independent and identically distributed Gaussian probability distribution with zero mean and unit variance for both classes. Similar to [62], we generate four tasks, each with $n = 100, n_v = 100$, and $n_{\text{the}} = 5000$ samples for training, validation, and testing, respectively. Before learning, all data are normalized to zeros mean and unit variance. Because we know the ground-truth features relevant to a task t, we can easily evaluate the feature learning performance by comparing the learned final feature vectors $\boldsymbol{\delta}_t = \boldsymbol{\delta}_0 + \boldsymbol{\delta}_{\text{ts}} + \boldsymbol{\delta}_{\text{ta}}$ with the ground-truth $d = d_0 + d_{\text{ts}} + d_{\text{ta}}$.

TABLE I
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)

National Cancer Institute (NCI) anti-cancer activity prediction data[5] is a set of benchmarks for predicting biological activity 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, otherwise it is negative. Table I summarizes the 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 multitask 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 multitask graph classification.

Predictive toxicology challenge (PTC) data includes a number of carcinogenicity tasks to predict the toxicology of chemical compounds.[6] The dataset we selected contains

[4]http://www.cse.fau.edu/~xqzhu/FelMuG/index.html

[5]http://pubchem.ncbi.nlm.nih.gov

[6]http://www.predictive-toxicology.org/ptc/

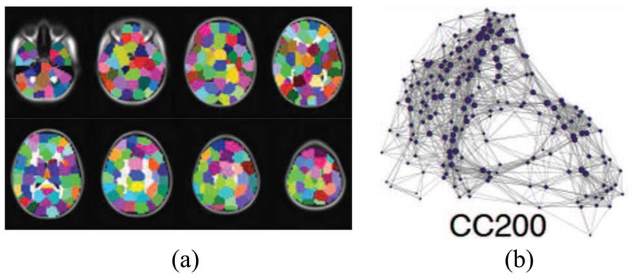

Fig. 2. Example of brain functional parcellation results from (a) fMRI image, and (b) corresponding brain graph. Each color region in (a) is an ROI, which corresponds to a node in (b) [16].

417 compounds from four types of test animals: 1) male mouse (MM); 2) female mouse (FM); 3) male rat (MR); and 4) female rat (FR). Each compound has 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 [38], 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 nonoverlapping subsets. We only consider one type of carcinogenicity test for each subset as its learning task. The subset information is also listed in Table I.

BrainNet functional brain network analysis data is constructed from the whole brain fMRI atlas [16]. 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 [16], 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 ADHD classification, hyperactive-impulsive (HI) classification, and gender classification (GD). The detailed graph information is summarized in Table I. 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).

2) Baseline Methods:

a) Multitask learning baselines: We compare the feature learning with state-of-the-art MTL methods, i.e., sep-L_1, logistic-L_1, logistic-L_{21} [20], rMTFL [23], dirty [22], and calibration [24] approaches. sep-L_1 performs feature learning separately on each task and logistic-L_1 jointly learns multitask features with LASSO regularization and the logistic loss function. The calibration approach is the most recent multitask feature learning algorithm [24]. All these algorithms are available in the MALSAR toolbox [49].

To build multitask graph classification baselines, we first 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 MTL baselines to the transferred vector datasets.

b) Graph classification baselines: We compare our method with three state-of-the-art graph classification methods, i.e., gBoost [8], gSemi [12], and gMLC [13]. These methods learn graph classification task separately, without considering graph samples from other tasks.

3) Measurements: The graph classification performance is measured in terms of accuracy and the area under the curve (AUC).

The accuracy of a classification task is the percentage of samples that are correctly classified. This is a widely used measurement for relatively balanced classification tasks. However, a more nature criterion for imbalanced data is AUC [66], which is defined by the area under an receiver operating characteristic (ROC) curve. The ROC curve for a binary classification problem plots the true positive rate as a function of the false positive rate.

For feature learning results, we measure the quality of discovered features, compared to the ground truth relevant features. In particular, we consider recall, precision, and F-measure. Suppose the ground truth relevant features is T_g, and the feature discovered by an algorithm is A_g. The recall is denoted by $\mathrm{Re} = |T_g \bigcap A_g|/|T_g|$, and the precision is defined as $\mathrm{Pr} = |T_g \bigcap A_g|/|A_g|$. Because a higher recall may imply a low precision, or vice versa, we use F-measure, defined as $2\mathrm{Pr} \times \mathrm{Re}/\mathrm{Pr} + \mathrm{Re}$, to measure preferred algorithm performance (i.e., a high precision and a high recall value).

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. We repeat ten times of graph classification experiments on NCI graphs and conduct tenfold cross-validation on PTC and brain networks. In keeping with [8], we report the best average accuracy and average AUC values over all tasks for graph classification tasks.

B. Feature Learning Result Comparisons

Fig. 3 reports the feature learning results of different algorithms on synthetic dataset and shows that all MTL algorithms obtain sparse solutions and find features shared by all tasks. Among the baseline algorithms, logistic-L21, rMTFL, and calibration select similar features, because they aim to select common features (no outlier task exists for rMTFL thus it

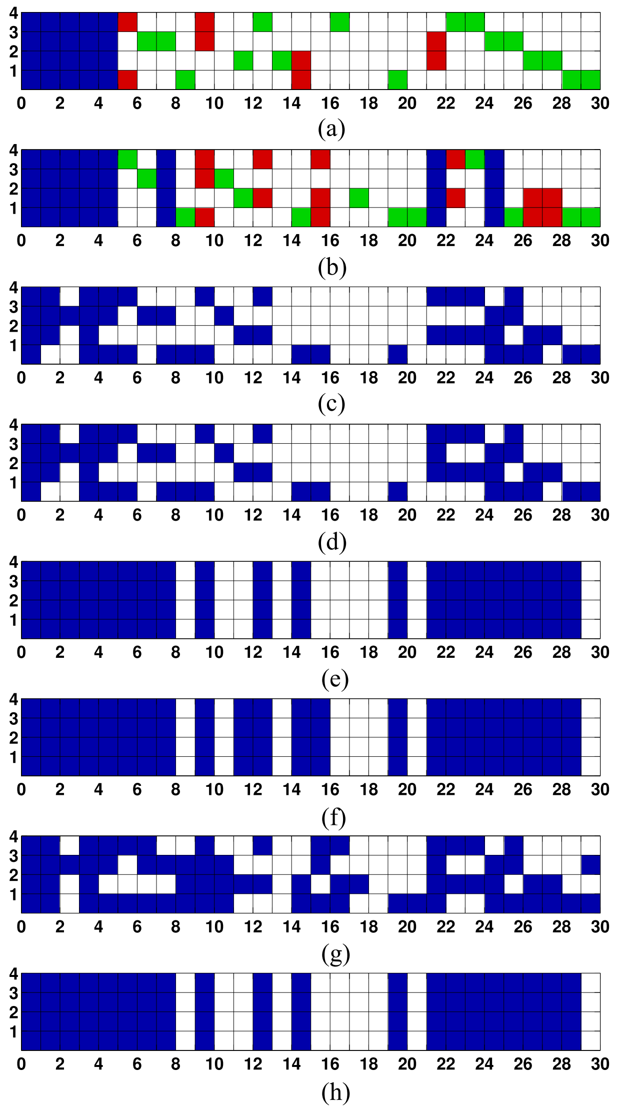

Fig. 3. Relevant features recovered by different algorithms on synthetic dataset for four tasks with $m = 30$, $d_0 = 5$, $d_{\text{ta}} = 2$, and $d_{\text{ts}} = 2$. Blue, red, and green colored cells refer to common features, task auxiliary features, and task specific features, respectively. (a) Ground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

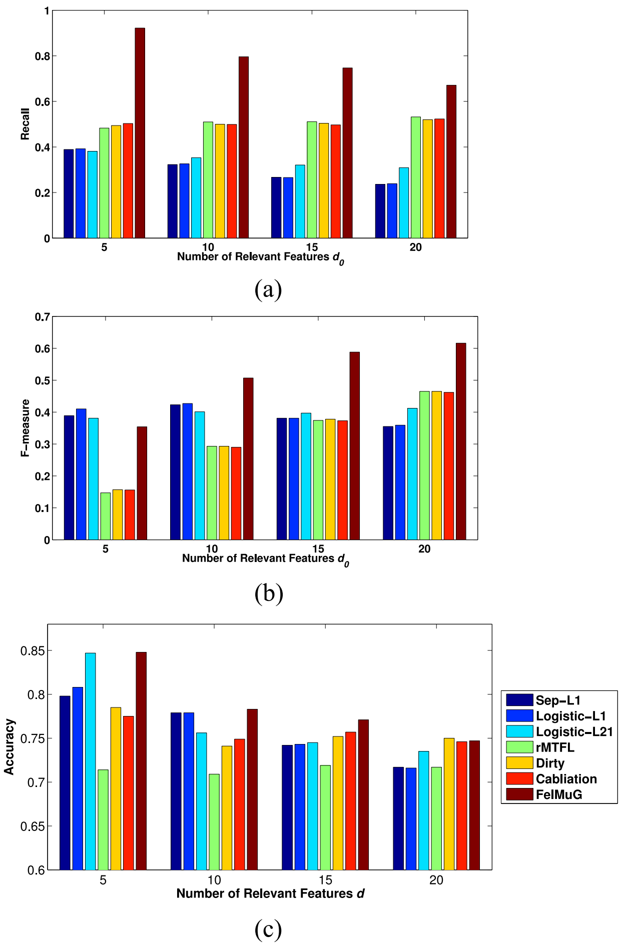

Fig. 4. Feature learning performance on synthetic data with varying d_0. We generate data with $d_{\text{ta}} = d_{\text{ts}} = d_0/2$, and $m = 100$. FelMuG is learned with $K_0 = K_a = K_t = 2$. A Recall value "1" means an algorithm can recover all relevant features, and F-measure with 1 means perfect 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). (a) Recall on selected features. (b) F-measure on selected features. (c) Classification accuracy on synthetic tasks.

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 achieves the best performance in recovering relevant features 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 multitask 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. By contrast, FelMuG 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.

C. Brain and Chemical Compound Graph Classification

1) Performance on NCI: We randomly label a small set of graphs as training graphs for each task of NCI graphs, 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 ten 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 mainly because these baselines are single task algorithms which ignore relevant graphs from similar tasks. Because subgraph features are represented by the edge

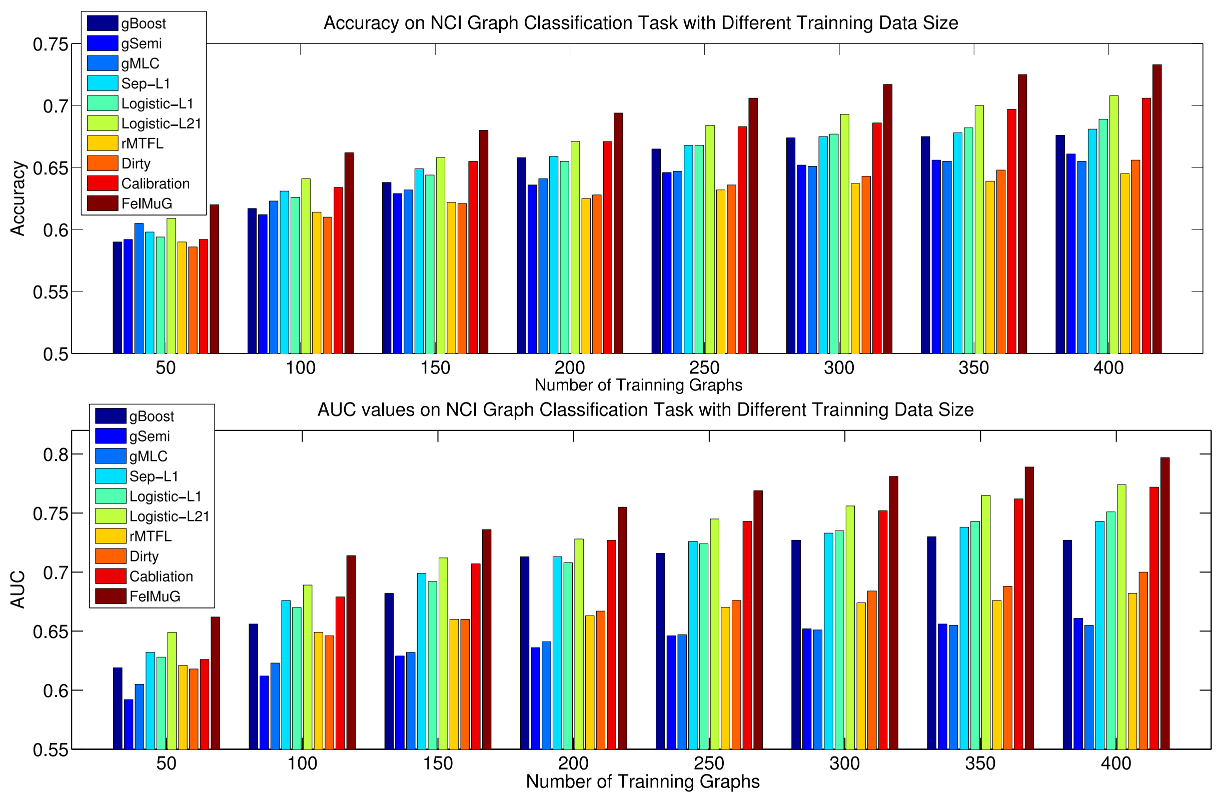

Fig. 5. Average accuracy and AUC values for NCI graph classification tasks with different number of training graphs in each task.

connection of some common atoms, some common discriminative structures may exist in multiple graph learning tasks and are therefore beneficial for multitask graph classification.

Regardless of which regularizers are used in the algorithm, existing MTL algorithms will first mine a set of frequent subgraph as features and then employ MTL 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 subgraph mining process is not driven by MTL objective, and may therefore miss discriminative subgraphs at the first place and 2) they ignore task specific and task auxiliary features for each task, so cannot capture the underlying unique discriminative subgraphs of each task. Since subgraph features are crucial for graph classification, a simple adaptation of existing MTL algorithm for graph classification is suboptimal.

By contrast, the proposed FelMuG method not only has the advantage of using graph samples from relevant tasks, but also unifies multitask 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.

2) 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 tenfold cross-validation on these tasks. In this way, we can reduce the bias of each method caused by limited training samples.

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

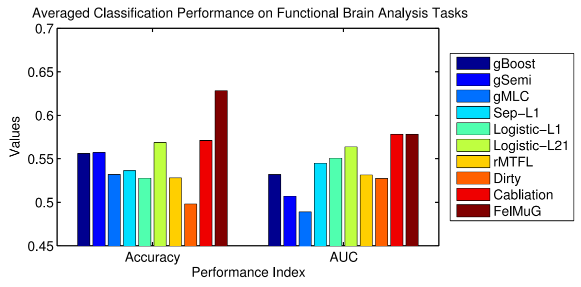

Fig. 6. Average accuracy and AUC values for functional brain analysis tasks.

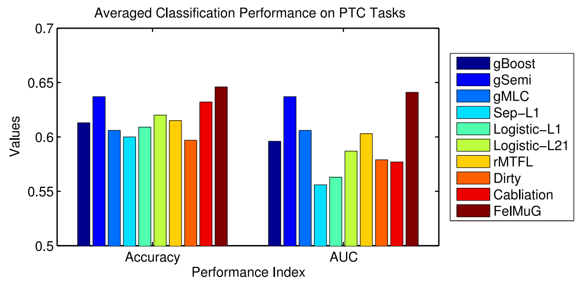

Fig. 7. Average accuracy and AUC values for PTC graph classification tasks.

D. Multitask Subgraph Exploration Efficiency

In this section, we investigate the efficiency of FelMuG in reducing the search space (Theorem 1 in Section V-A)

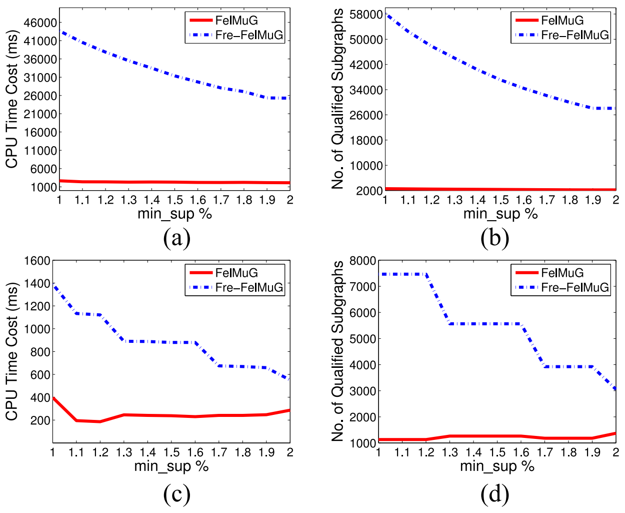

Fig. 8. Pruning efficiency on NCI and PTC graph classification tasks. (a) and (b) NCI datasets. (c) and (d) PTC datasets.

for subgraph feature exploration. Because the search space is exponentially large, assessing the pruning effectiveness of FelMuG with respect to the whole search space is challenging. Accordingly, we introduce a threshold value min_sup, to denote the minimum frequency of each qualified subgraph feature in the training graph dataset and bound the number of subgraphs in the search space. In doing so, we know the total number of subgraph candidates, and can then assess its pruning efficiency by checking the percentage of candidates pruned in the process. The frequent subgraph-based algorithm is termed Fre-FelMuG.

The results in Fig. 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 [Fig. 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 nonprune 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) 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).

VIII. Conclusion

In this paper, we formulated a new multitask graph learning problem. We argued that existing MTL algorithms are inapplicable to graphs, mainly because they cannot handle structure data, and cannot explicitly capture relationships between tasks and individual features, which are crucial for classifying and understanding graph classification tasks. Accordingly, we proposed a task sensitive FelMuG multitask graph classification algorithm. The uniqueness of FelMuG lies in 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 the underlying multiple learning tasks are iteratively optimized to form a multitask graph classification model with a global optimization goal. Experiments on synthetic and real-world data confirm that: 1) FelMuG can accurately capture feature-task relationships; 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 multitask 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 Comput. Aided Drug Designs*, vol. 9, no. 2, pp. 153–163, 2013.
[2] J. Richiardi, S. Achard, H. Bunke, and D. Van De Ville, "Machine learning with brain graphs: Predictive modeling approaches for functional imaging in systems neuroscience," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 58–70, May 2013.
[3] X. Kong and P. Yu, "Brain network analysis: A data mining perspective," *KDD Explor.*, vol. 15, no. 2, pp. 30–38, 2013.
[4] Y. Park, D. S. Reeves, and M. Stamp, "Deriving common malware behavior through graph clustering," *Comput. Security*, vol. 39, pp. 419–430, Nov. 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. Comput. Biol. Bioinformat.*, vol. 10, no. 5, pp. 1211–1217, Sep. 2013.
[6] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-Lehman graph kernels," *J. Mach. Learn. Res.*, vol. 12, pp. 2539–2561, Sep. 2011.
[7] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *J. Mach. Learn. Res.*, vol. 11, pp. 1201–1242, Apr. 2010.
[8] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda, "gBoost: A mathematical programming approach to graph classification and regression," *Mach. Learn.*, vol. 75, no. 1, pp. 69–89, 2009.
[9] S. Pan, J. Wu, X. Zhu, G. Long, and C. Zhang, "Finding the best not the most: Regularized loss minimization subgraph selection for graph classification," *Pattern Recognit.*, vol. 48, no. 11, pp. 3783–3796, 2015.
[10] M. Thoma *et al.*, "Near-optimal supervised feature selection among frequent subgraphs," in *Proc. SIAM Data Mining*, Sparks, NV, USA, 2009, pp. 1075–1086.
[11] H. Fei and J. Huan, "Boosting with structure information in the functional space: An application to graph classification," in *Proc. ACM SIGKDD*, Washington, DC, USA, 2010, pp. 643–652.
[12] X. Kong and P. S. Yu, "Semi-supervised feature selection for graph classification," in *Proc. ACM SIGKDD*, 2010, pp. 793–802.
[13] X. Kong and P. S. Yu, "Multi-label feature selection for graph classification," in *Proc. ICDM*, Sydney, NSW, Australia, 2010, pp. 274–283.
[14] X. Kong, W. Fan, and P. S. Yu, "Dual active feature and sample selection for graph classification," in *Proc. ACM SIGKDD*, San Diego, CA, USA, 2011, pp. 654–662.
[15] Y. Zhao, X. Kong, and P. S. Yu, "Positive and unlabeled learning for graph classification," in *Proc. ICDM*, Vancouver, BC, Canada, 2011, pp. 962–971.
[16] R. C. Craddock, G. A. James, P. E. Holtzheimer, X. P. Hu, and H. S. Mayberg, "A whole brain fMRI atlas generated via spatially constrained spectral clustering," *Human Brain Mapping*, vol. 33, no. 8, pp. 1914–1928, 2012.
[17] A. Tenev *et al.*, "Machine learning approach for classification of ADHD adults," *Int. J. Psychophysiol.*, vol. 93, no. 1, pp. 162–166, 2014.

[18] J. T. Vogelstein, W. G. Roncal, R. J. Vogelstein, and C. E. Priebe, "Graph classification using signal-subgraphs: Applications in statistical connectomics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 7, pp. 1539–1551, Jul. 2013.

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

[20] T. Evgeniou, A. Argyriou, and M. Pontil, "Multi-task feature learning," in *Proc. NIPS*, vol. 19. Vancouver, BC, Canada, 2007, p. 41.

[21] J. Liu, S. Ji, and J. Ye, "Multi-task feature learning via efficient l2, 1-norm minimization," in *Proc. UAI*, Montreal, QC, Canada, 2009, pp. 339–348.

[22] A. Jalali, S. Sanghavi, C. Ruan, and P. K. Ravikumar, "A dirty model for multi-task learning," in *Proc. NIPS*, Vancouver, BC, Canada, 2010, pp. 964–972.

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

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

[25] Y. Zhang and D.-Y. Yeung, "A convex formulation for learning task relationships in multi-task learning," in *Proc. UAI*, 2010, pp. 733–742.

[26] W. L. Zhong and J. T. Y. Kwok, "Convex multitask learning with flexible task clusters," in *Proc. ICML*, Edinburgh, U.K., 2012, pp. 49–56.

[27] A. Kumar and H. Daumé, III, "Learning task grouping and overlap in multi-task learning," in *Proc. ICML*, Edinburgh, U.K., 2012, pp. 1383–1390.

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

[29] J. T. Vogelstein, W. G. Roncal, R. J. Vogelstein, and C. E. Priebe, "Graph classification using signal-subgraphs: Applications in statistical connectomics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 7, pp. 1539–1551, Jul. 2013.

[30] H. Kashima, K. Tsuda, and A. Inokuchi, "Kernels for graphs," in *Kernel Methods in Computational Biology*, B. Schoölkopf, K. Tsuda, and J. P. Vert, Eds. Cambridge, MA, USA: MIT Press, 2004.

[31] S. Pan and X. Zhu, "Graph classification with imbalanced class distributions and noise," in *Proc. IJCAI*, Beijing, China, 2013, pp. 1586–1592.

[32] N. Jin, C. Young, and W. Wang, "GAIA: Graph classification using evolutionary computation," in *Proc. ACM SIGMOD*, Indianapolis, IN, USA, 2010, pp. 879–890.

[33] N. Jin, C. Young, and W. Wang, "Graph classification based on pattern co-occurrence," in *Proc. ACM CIKM*, Hong Kong, 2009, pp. 573–582.

[34] H. Wang, P. Zhang, I. Tsang, L. Chen, and C. Zhang, "Defragging subgraph features for graph classification," in *Proc. ACM Int. Conf. Inf. Knowl. Manag.*, Melbourne, VIC, Australia, 2015, pp. 1687–1690.

[35] X. Yan, H. Cheng, J. Han, and P. S. Yu, "Mining significant graph patterns by leap search," in *Proc. ACM SIGMOD*, Vancouver, BC, Canada, 2008, pp. 433–444.

[36] S. Ranu and A. K. Singh, "GraphSig: A scalable approach to mining significant subgraphs in large graph databases," in *Proc. ICDE*, Shanghai, China, 2009, pp. 844–855.

[37] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," in *Proc. ICDM*, Maebashi, Japan, 2002, pp. 721–724.

[38] T. Kudo, E. Maeda, and Y. Matsumoto, "An application of boosting to graph classification," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Vancouver, BC, Canada, 2004, pp. 729–736.

[39] S. Pan, J. Wu, X. Zhu, and C. Zhang, "Graph ensemble boosting for imbalanced noisy graph stream classification," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 954–968, May 2015.

[40] S. Pan, J. Wu, and X. Zhu, "CogBoost: Boosting for fast cost-sensitive graph classification," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 11, pp. 2933–2946, Nov. 2015.

[41] S. Pan, X. Zhu, C. Zhang, and P. S. Yu, "Graph stream classification using labeled and unlabeled graphs," in *Proc. ICDE*, Brisbane, QLD, Australia, 2013, pp. 398–409.

[42] J. Wu *et al.*, "Multi-graph-view learning for graph classification," in *Proc. Int. Conf. Data Mining*, Shenzhen, China, 2014, pp. 590–599.

[43] J. Wu, S. Pan, X. Zhu, Z. Cai, and C. Zhang, "Multi-graph-view learning for complicated object classification," in *Proc. IJCAI*, Buenos Aires, Argentina, 2015, pp. 3953–3959.

[44] J. Wu, S. Pan, X. Zhu, and Z. Cai, "Boosting for multi-graph classification," *IEEE Trans. Cybern.*, vol. 45, no. 3, pp. 416–429, Mar. 2015.

[45] J. Wu, X. Zhu, C. Zhang, and P. S. Yu, "Bag constrained structure pattern mining for multi-graph classification," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2382–2396, Oct. 2014.

[46] J. Wu *et al.*, "Multi-graph learning with positive and unlabeled bags," in *Proc. SIAM Data Mining*, Philadelphia, PA, USA, 2014, pp. 1586–1592.

[47] T. Evgeniou and M. Pontil, "Regularized multi-task learning," in *Proc. ACM SIGKDD*, Seattle, WA, USA, 2004, pp. 109–117.

[48] Y. Zhang and D.-Y. Yeung, "Multi-task boosting by exploiting task relationships," in *Proc. ECML/PKDD*, Bristol, U.K., 2012, pp. 697–710.

[49] J. Zhou, J. Chen, and J. Ye, "MALSAR: Multi-task learning via structural regularization," Arizona State Univ., Tempe, AZ, USA, 2011. [Online]. Available: http://www.public.asu.edu/~jye02/Software/MALSAR.

[50] H. Fei and J. Huan, "Structured feature selection and task relationship inference for multi-task learning," *Knowl. Inf. Syst.*, vol. 35, no. 2, pp. 345–364, 2013.

[51] Z. Hong, X. Mei, D. Prokhorov, and D. Tao, "Tracking via robust multi-task multi-view joint sparse representation," in *Proc. ICCV*, Sydney, NSW, Australia, 2013, pp. 649–656.

[52] B. Bakker and T. Heskes, "Task clustering and gating for Bayesian multitask learning," *J. Mach. Learn. Res.*, vol. 4, pp. 83–99, May 2003.

[53] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[54] X. Shi, X. Kong, and P. S. Yu, "Transfer significant subgraphs across graph databases," in *Proc. SIAM Data Mining*, Anaheim, CA, USA, 2012, pp. 552–563.

[55] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis, "Frequent substructure-based approaches for classifying chemical compounds," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 8, pp. 1036–1050, Aug. 2005.

[56] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Mach. Learn.*, vol. 46, nos. 1–3, pp. 389–422, 2002.

[57] T. R. Golub *et al.*, "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, no. 5439, pp. 531–537, 1999.

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

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

[60] P. Liu *et al.*, "Feature disentangling machine—A novel approach of feature selection and disentangling in facial expression analysis," in *Computer Vision—ECCV 2014*. Zurich, Switzerland: Springer, 2014, pp. 151–166.

[61] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *J. Mach. Learn. Res.*, vol. 9, pp. 2491–2521, Nov. 2008.

[62] A. Rakotomamonjy, R. Flamary, G. Gasso, and S. Canu, "$l_p - l_q$ penalty for sparse linear and sparse multiple kernel multitask learning," *IEEE Trans. Neural Netw.*, vol. 22, no. 8, pp. 1307–1320, Aug. 2011.

[63] J. E. Kelley, Jr., "The cutting-plane method for solving convex programs," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 4, pp. 703–712, 1960.

[64] L. A. Wolsey, *Integer Programming*, vol. 42. New York, NY, USA: Wiley, 1998.

[65] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Aug. 2008.

[66] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

Shirui Pan received the Ph.D. degree in computer science from the University of Technology Sydney (UTS), Sydney, NSW, Australia, in 2015.

He is a Research Associate with the Centre for Quantum Computation and Intelligent Systems, UTS. He has published over 20 research papers in top-tier journals and conferences, including the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the IEEE TRANSACTIONS ON CYBERNETICS, *Pattern Recognition*, International Joint Conference on Artificial Intelligence (IJCAI), International Conference on Data Engineering, International Conference on Data Mining, SIAM International Conference on Data Mining, International Conference on Information and Knowledge Management, and Pacific-Asia Conference on Knowledge Discovery and Data Mining. His current research interests include data mining and machine learning.

Jia Wu received the Ph.D. degree in computer science from the University of Technology Sydney (UTS), Sydney, NSW, Australia.

He is a Research Associate with the Centre for Quantum Computation and Intelligent Systems, UTS. His current research interests include data mining. Since 2009, he has published over 20 refereed journal and conference papers, such as the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING and the IEEE TRANSACTIONS ON CYBERNETICS, in the above area.

Xingquan Zhu (SM'12) received the Ph.D. degree in computer science from Fudan University, Shanghai, China.

He is an Associate Professor with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL, USA, and a Distinguished Visiting Professor (Eastern Scholar) with the Shanghai Institutions of Higher Learning. His current research interests include data mining, machine learning, and bioinformatics. Since 2000, he has published over 200 refereed journal and conference papers in the above area.

Prof. Zhu was a recipient of two Best Paper Awards and one Best Student Paper Award.

Guodong Long received the Ph.D. degree in computer science from the University of Technology Sydney (UTS), Sydney, NSW, Australia, in 2014.

He is a Researcher Associate and a Core Member with the Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology, UTS. His current research interests include machine learning, data mining, and cloud computing.

Chengqi Zhang (SM'95) received the Ph.D. degree from the University of Queensland, Brisbane, QLD, Australia, in 1991, and the D.Sc. degree from Deakin University, Geelong, VIC, Australia, in 2002.

Since 2001, he has been a Professor of Information Technology with the University of Technology Sydney (UTS), Sydney, NSW, Australia, and the Director of the UTS Priority Investment Research Centre for Quantum Computation and Intelligent Systems since 2008. His current research interests include data mining and its applications.

Prof. Zhang is the General Co-Chair of KDD 2015, Sydney, the Local Arrangements Chair of IJCAI 2017, Melbourne, VIC, Australia, and a fellow of the Australian Computer Society.