Positive and Unlabeled Multi-Graph Learning

Jia Wu, Shirui Pan, Xingquan Zhu, *Senior Member, IEEE*, Chengqi Zhang, *Senior Member, IEEE*, and Xindong Wu, *Fellow, IEEE*

***Abstract*—In this paper, we advance graph classification to handle multi-graph learning for complicated objects, where each object is represented as a bag of graphs and the label is only available to each bag but not individual graphs. In addition, when training classifiers, users are only given a handful of positive bags and many unlabeled bags, and the learning objective is to train models to classify previously unseen graph bags with maximum accuracy. To achieve the goal, we propose a positive and unlabeled multi-graph learning (puMGL) framework to first select informative subgraphs to convert graphs into a feature space. To utilize unlabeled bags for learning, puMGL assigns a confidence weight to each bag and dynamically adjusts its weight value to select "reliable negative bags." A number of representative graphs, selected from positive bags and identified reliable negative graph bags, form a "margin graph pool" which serves as the base for deriving subgraph patterns, training graph classifiers, and further updating the bag weight values. A closed-loop iterative process helps discover optimal subgraphs from positive and unlabeled graph bags for learning. Experimental comparisons demonstrate the performance of puMGL for classifying real-world complicated objects.**

***Index Terms*—Graph, multi-instance (MI), subgraph, features, positive and unlabeled (PU) learning, classification.**

I. Introduction

LEARNING and classifying objects have been commonly used for many applications, such as text mining [1], [2] and image recognition [3]. For learning purposes, objects are required to be represented as instances by using feature vector and class label to denote characteristics and categorizes of the objects, respectively. This representation has been commonly used to represent objects with simple features and class labels, such as using bag-of-word features and a class label (e.g., positive versus negative) to represent a news article for automatic news categorization [4].

Manuscript received August 21, 2015; revised December 14, 2015; accepted January 24, 2016. Date of publication March 23, 2016; date of current version March 15, 2017. This work was supported by the Australian Research Council Discovery Projects under Grant DP140100545, Grant DP140102206, and Grant LP120100566. This paper was recommended by Associate Editor X. Li. *(Corresponding author: Shirui Pan.)*

J. Wu, S. Pan, 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: jia.wu@uts.edu.au; shirui.pan@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).

X. Wu is with the Department of Computer Science, University of Vermont, Burlington, VT 05405 USA (e-mail: xwu@uvm.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.2016.2527239

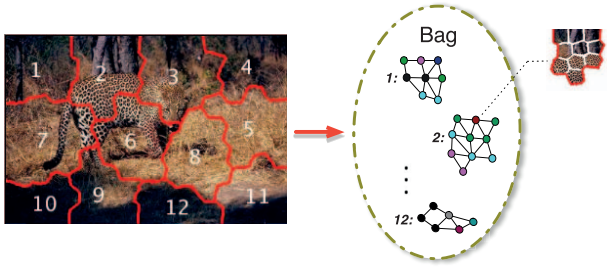

Fig. 1. Multi-graph learning for content-based image annotation. Each image (i.e., bag) consists of a number of regions (with each region corresponding to one graph for preserving local structure of the region). The compartments in each region (i.e., nodes in graphs) are superpixels [8].

In reality, simple feature and class representation may be inadequate for certain applications, which involve objects with variant characteristics or complex behaviors. For instance, research in drug discovery has observed that activities of molecules may vary significantly and show different behaviors in response to changing environments. Therefore, for a specific molecule, its feature values may vary under different experimental conditions. An efficient way to accommodate such changing behaviors and characteristics is to represent the molecule as a bag of instances with each instance representing molecule's behavior in one single experiment. If, for a number of experiments, the molecule demonstrates positive/interested behavior, the bag will be labeled as positive. If, for all experiments, the molecule does not exhibit positive behaviors, the bag will be labeled as negative. Similarly, in content-based image annotation, an image can be represented as a bag with each region inside the image denoting an instance inside the bag. If an image region contains an object of interest (e.g., a leopard), the image will be labeled as positive.

In order to tackle the above mentioned complications, multi-instance (MI) is emerged as a new classification tool [5] with each object provided for learning (or classifying) being a bag of instances. The label is only available for the bag (i.e., an instance set) but not for each individual instance. For MI learning, existing methods [6], [7] require that training samples are provided and represented in vector space, which inherently prohibits them from being applied to complicated objects containing structure information.

Structure dependency exists in many applications and plays crucial role for describing content and structure of the objects. Take content-based image annotation in Fig. 1 as an example. For traditional MI learning, the whole image is represented as

a bag, with region #2 being treated as a single instance represented by using visual features, such as color histogram or texture. In other words, although region #2 contains multiple subregions (i.e., "tree," "grass," and "leopard") with special structures and layout, existing MI leaning approaches discard the structure information and only consider visual features of the whole region for learning. Instead, a more effective data representation is to explicitly explore complex relationships among the data by using effective data structures, such as graphs, to represent data for learning. In this case, an image region can be naturally represented as a graph in order to preserve and represent local structure information inside the region. This representation is more accurate than simply treating the whole region as one feature instance.

Another prerequisite of existing MI learning methods is that they require both positive and negative bags being provided. In reality, it is common to have only positive bags to describe users' interests and there is no negative bags, so learning is supported by a handful of positive bags and some unlabeled bags, which may be positive, negative, or even irrelevant to the underlying learning task. For example, in image retrieval, query images provided by users can be considered as positive bags to express his/her retrieval interest. During the search process, users might click one or multiple images interesting to them. The clicked images can be regarded as positive bags whereas majority images will remain unclicked and therefore being unlabeled bags. In this case, we do not know whether unclicked images are of interest to users or not. In other words, only positive bags (i.e., clicked images) and unlabeled bags (i.e., unclicked images) are available for learning.

The above observations raise a special learning setting where only positive and unlabeled (PU) graph bags exist for learning. In this paper, we refer to this problem as positive and unlabeled multi-graph learning (puMGL). Indeed, to date, existing graph classification approaches [9]–[12] mainly consist of global distance-based approaches and local subgraph feature-based methods. For the former, including graph kernels, graph embedding, and graph transformation-based methods, they directly work on the graph data in order to calculate graph similarity. In contrast, the latter translates a graph into a feature-vector instance by using subgraph patterns mined from the training graph data set. After that, graph classification becomes a traditional learning task where existing supervised learning approaches [e.g., support vector machine (SVM)] can be adopted to train the models.

For existing graph classification methods, a label is assigned to each graph, so they cannot be directly applied to multi-graph setting where the label is only available for a bag (i.e., graph set). On the other hand, traditional MI learning requires data to be represented in vector space, but cannot tackle graph data. Accordingly, puMGL is facing the following main challenges.

1) *Multi-Graph Representation:* Because graphs do not have feature values required for derive learning models, an important process to deal with graph data is to explore informative subgraphs to represent them into vector space. In addition, traditional MI learning methods are incapable of handling structure data (e.g., graph). Therefore, we need to design a novel framework to represent complicated objects for classification.
2) *Unlabeled Bags:* Because our problem setting only contains PU bags and there is no negative bag, in order to make existing MI or supervised learning methods useful for multi-graph classification, we need solutions to identify "reliable negative" bags from unlabeled bags for learning.
3) *Uncertainty Inside Bags:* For multi-graph with PU bags, the uncertainty inside bags is twofold.
    a) The genuine labels of graphs inside a positive bag are unknown, so existing subgraph feature mining approaches are unable [13], [14] to extract subgraph features from multi-graph bags because they require each graph to be explicitly labeled.
    b) The identified reliable negative bags may be incorrect (i.e., a negative bag may contain positive graph), and contain incorrect bag labels. Accordingly, an effective multi-graph learning algorithm needs to take such uncertainty into consideration.

When PU graph bags are provided for learning, because genuine labels of graphs in positive bag are unknown, a straightforward graph-level solution for multi-graph learning is to propagate the bag label to graphs inside each bag. In this case, the problem becomes "PU graph classification," which has been addressed by existing research [15]. Unfortunately, simple label transmission for positive bags may result in incorrect class labels for negative graphs. Alternatively, one can first explore some frequent subgraph features to represent the graphs into vector space, and represent each graph bag as one instance (i.e., a bag-level solution). By doing so, existing PU learning algorithms [16], [17] can be applied to solve the problem. More specifically, some initially identified "reliable negative graph bags," which will be iteratively updated, are used to help the positive bags train the MI classifier. However, this type of bag-level strategy is still inefficient mainly because their frequent subgraph features are selected without taking multi-graph bag constraints and uncertainty inside the bag into consideration [13], [14]. As a result, their subgraph features may not be discriminative for classifying multi-graph bags and result in suboptimal classification performance.

To solve the above challenges, we put forward a novel puMGL framework, which relies on an iterative discriminative subgraph-based learning model for maximum classification accuracy. PuMGL has three notable technical contributions: 1) a graph feature scoring algorithm with capability of pruning the search space; 2) a margin graph pool (MGP)-based approach to solve the dual uncertainty for graphs in positive bags and reliable negative graph bags selected from unlabeled bags; and 3) a multi-graph learning algorithm with only PU graph bags. Experimental comparisons on real-world learning tasks confirm the effectiveness of the proposed designs.

The remaining part of this paper is structured as follows. Section II reviews related works. Preliminary and problem statement are addressed in Section III. Section IV outlines the proposed puMGL framework, followed by experiments in Section V. We conclude the paper in Section VI.

II. Related Works

MI learning was first proposed by Dietterich *et al.* [5] for drug activity prediction, and later applied to applications [18], such as image classification, text categorization, web mining, and 3-D object recognition [19]. In summary, existing MI learning algorithms can be broadly categorized into two major groups.

1) Upgrading existing single-instance-based learning algorithms to support MI learning. Examples include MI logistic regression (MILR) learning [20].
2) Specifically designed MI learning algorithms which directly utilize bag constraints to reorganize instances inside each bag into specific formats for learning [6].

For all existing MI learning methods, one prerequisite is that their training data must contain both positive and negative bags. In reality, many applications only have positive bags to indicate users' learning interests and remaining bags are unlabeled (which may be positive, negative, or irrelevant to the learning task). This problem is referred to as PU learning in the literature. One popular solution to tackle unlabeled data is the heuristic labeling approach [21], which follows a two-step strategy: 1) select a number of reliable negative instances from the unlabeled set by using various techniques, such as the expectation maximization [22] and 2) the identified reliable negative instances, together with the positive samples, are combined to train traditional classification models. Density-estimation-based PU learning approaches estimate the conditional probability of the positive class for predication [23], [24]. For these methods, some probability density functions, such as marginal probability of the positive class or the probability of the positive instance, are estimated as an intermediate learning step. In reality, the estimation of conditional probability densities is a rather challenging task, especially with a very limited number of labeled data [23].

For existing MI learning algorithms, they require instances in each bag to be represented in a tabular feature-vector format, which makes them incapable of handling graphs. To date, existing graph classification methods mainly rely on two types of approaches, including: 1) global distance-based approaches, such as graph kernel or graph embedding [25] and 2) local subgraph pattern features-based methods, which find a feature representation for graphs by using subgraph patterns discovered from the graph set. Compared to distance-based methods, subgraph feature-based approaches can explicitly indicate which substructures (a portion of the graph) contribute to the classification. The most popular subgraph feature evaluation criterion is frequency. For example, gSpan [26] uses a lexicographic order-based coding to discover frequently connected subgraphs that can be used as features. Recently, Zhao *et al.* [15] proposed to use PU graphs for learning.

For all existing graph classification methods, the object for learning is a single graph, so they cannot be directly applied to multi-graph setting where an object to be classified is a graph bag (i.e., a graph set), which is also different from existing graph-based learning using graph-regularization on vectors [27], [28]. For existing MI learning techniques, the classification object is represented in feature-vector space, so they cannot be applied to graphs. This naturally raises the necessity of designing new methods to handle bags containing PU graph bags.

III. Preliminaries and Problem Statement

In this section, we first introduce important notations and definitions, and then state our research problem.

Definition 1 (Connected Graph): A graph is represented as $G = (\mathcal{V}, E, \mathcal{L}, l)$, where $\mathcal{V}$ is a set of vertices $\mathcal{V} = \{v_1, \ldots, v_{n_v}\}$, $E \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, and $\mathcal{L}$ is the set of labels for the vertices and edges. $l : \mathcal{V} \cup E \to \mathcal{L}$ is the function assigning labels to the vertices and edges. A connected graph is a graph such that there is a path between any pair of vertices.

Definition 2: (Graph Bag): Denote $\boldsymbol{\mathcal{B}} = \{B_1, \ldots, B_n\}$ a set with n bags, and B_i is the ith bag in the set, which can be positive B_i^+ and unlabeled B_j^u. Let $Y = [y_1, \ldots, y_n]$, where y_i is the label of B_i. Generally, a positive and a negative bag's label can be denoted by $y_i = +1$ and $y_i = -1$, respectively. In a puMGL setting, an unlabeled bag B_j^u's label is denoted by $y_j = 0$. The collections of PU bag sets can be denoted by $\boldsymbol{\mathcal{B}}^+$ and $\boldsymbol{\mathcal{B}}^u$, respectively. During the learning process, the algorithm may identify a set of unlabeled bags to form a negative bag set, which is denoted by $\boldsymbol{\mathcal{B}}^-$.

We use $G_{i,j}$ (G_j for abbreviation) whose label is y_j to denote the jth graph in the bag $B_i = \{G_{i,1}, \ldots, G_{i,n_i}\}$. To tackle unreliable bag labels in puMGL setting (challenge #3 in Section I), we use a weight value w_i to indicate the label confidence of each bag B_i. So for a positive bag B_i^+, its weight value w_i is 1 (because it is genuinely positive), whereas for an identified negative bag B_j^-, its weight value $w_j \in (0, 1]$, with a higher w_j value indicating that B_j is more likely being negative.

Definition 3: (Subgraph): Let $G = (\mathcal{V}, E, \mathcal{L}, l)$ and $g = (\mathcal{V}', E', \mathcal{L}', l')$ be two graphs. g is a subgraph of G ($g \subseteq G$), iff there exists an injective function $\varphi : \mathcal{V}' \to \mathcal{V}$ subject to: 1) $\forall v \in \mathcal{V}', l'(v) = l(\varphi(v))$ and 2) $\forall (u, v) \in E', (\varphi(u), \varphi(v)) \in E$ and $l'(u, v) = l(\varphi(u), \varphi(v))$. If g is a subgraph of G, G is a supergraph of g.

Definition 4: (Subgraph Feature Representation): Denote $\mathcal{S}_g = \{g_1, \ldots, g_s\}$ a set of subgraphs discovered from a given graph set $\mathcal{G}$. For each graph G_i, we use a subgraph feature vector $\mathbf{x}_i = [x_i^{g_1}, \ldots, x_i^{g_s}]^\top$ to represent G_i in graph domain, where $x_i^{g_k} = 1$ iff g_k is a subgraph of G_i (i.e., $g_k \subseteq G_i$) and $x_i^{g_k} = 0$, otherwise.

Given a set of bags $\boldsymbol{\mathcal{B}}$ contains a number of positive $\boldsymbol{\mathcal{B}}^+$ and unlabeled $\boldsymbol{\mathcal{B}}^u$ graph bags, puMGL learning aims to build a prediction model from $\boldsymbol{\mathcal{B}}$ to accurately predict labels of previously unseen bags.

IV. Positive and Unlabeled MG Learning

A. Overall Framework

The proposed puMGL framework is shown in Fig. 2. In summary, puMGL uses the following three major procedures to tackle the main challenges discussed in Section I.

1) *Embedding Subgraph Feature Exploration:* In order to explore informative subgraph patterns from the graph set in MGP, we propose a confidence weight value embedding approach (detailed in Section IV-B2) to identify

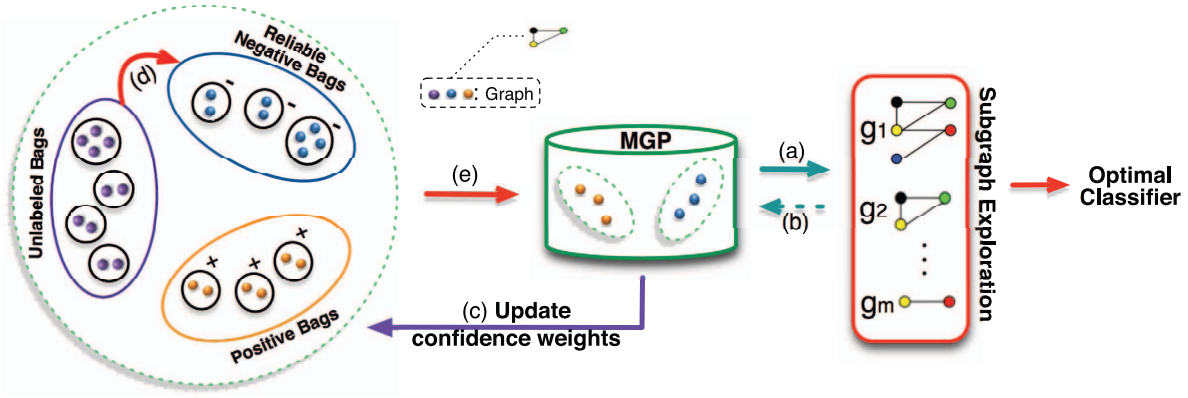

Fig. 2. Conceptual view of the puMGL framework: given training multi-graph set consisting of PU graph bags, each unlabeled bag is initially assigned a confidence value (i.e., weight) in order to establish the MGP for further learning. During the learning process, (b) some informative subgraphs are explored to represent graphs in MGP and training set by (a) utilizing the proposed confidence weight embedding approach. After that, a classifier is built based on the MGP (in vector feature space) to (c) update the weight values of unlabeled bags. Accordingly, (d) some "reliable negative bags" are identified from unlabeled set to (e) help update the graphs in MGP, which consists of the "least negative graph" from the unlabeled set and those "most positive graph" from the positive set. The quality of the reliable negative bags and the corresponding graphs in MGP are continuously improved through the iterative process until the algorithm converges.

discriminative subgraph patterns to represent graphs in multi-graph bags for learning.

2) *Margin Graph Pool:* Because there is no negative bag for training multi-graph classification models, after some graph bags are identified as reliable negative bags, a maximum margin strategy is proposed to select most positive subgraphs from the positive set, and select least negative subgraphs from identified reliable negative graph bags, respectively, to form MGP. Because MGP contains signature patterns with respect to positive and identified negative bags, it will help identify decision boundaries for separating positive bags.
3) *Unlabeled Bag Weight Updating:* In order to properly identify negative bags from unlabeled bags, every unlabeled bag is assigned an initial confidence weight that will be used to identify the reliable negative bags. Obviously, the initial weight is inaccurate, so the weight updating process will be carried out to improve the quality of identified negative bags.

B. puMGL Learning

In order to tackle the multi-graph learning problem with only PU graph bags, we need to consider the following two research subproblems: 1) how to design an effective approach to construct MGP, which contains graphs with relatively reliable labels for learning; and 2) how to design a subgraph feature evaluation criterion to assess subgraphs discovered from MGP to represent graphs for learning.

The above two problems are closely related to each other. In order to build the MGP, the reliable negative bags should be first identified. In other words, MGP needs reliable negative bags to be selected from the unlabeled bag set. Nevertheless, the reliable negative graph bags are selected based on subgraph features. On the other hand, the discriminative power of subgraph features, which is used to represent graphs into vector space, is directly dependent on the quality of graphs in the MGP. Accordingly, an optimization framework is proposed to alternately optimize the subgraph feature selection and the selection of reliable negative graph bags.

1) Optimization Framework: Assume that each bag $\mathcal{B}_i$ from the training bag set $\boldsymbol{\mathcal{B}}$ uses w_i to represent its confidence weight. The collected graphs in MGP, which can be denoted by $\mathcal{G} = \{\hat{G}_1, \ldots, \hat{G}_j, \ldots, \hat{G}_p\}$ and p indicates the number of graphs in the MGP, can be obtained by using the weight values of the bags. Each $\hat{G}_j$ in $\mathcal{G}$ has a weight $\hat{w}_j$, which is determined by the related bag. Let $\mathcal{S}_g$ denote the total subgraph set explored from graph set $\mathcal{G}$. The proposed learning task aims to extract a number of discriminative subgraphs (i.e., features) $\mathbf{g}$, ($\mathbf{g} \subseteq \mathcal{S}_g$) to represent graphs, and also uses the confidence weight vector $\hat{\mathbf{w}}$ to build the MGP for training classifiers. To achieve this goal, we use an objective function $\mathcal{J}(\mathbf{g}, \hat{\mathbf{w}})$ in (1) to estimate the subgraph feature dependency of $\mathbf{g}$ given the weight $\hat{\mathbf{w}}$. In (1), $|\cdot|$ denotes the cardinality of a set, and m denotes the number of subgraphs selected from $\mathcal{S}_g$

$$\left(\mathbf{g}^{\star}, \hat{\mathbf{w}}^{\star}\right) = \underset{\mathbf{g}\subseteq\mathcal{S}_g, \hat{w}_i\in(0,1]}{\arg\max} \; \mathcal{J}\left(\mathbf{g}, \hat{\mathbf{w}}\right) \;\; \text{s.t.} \;\; |\mathbf{g}| \leq m. \tag{1}$$

In order to maximize the objective function $\mathcal{J}(\mathbf{g}, \hat{\mathbf{w}})$, we can utilize the graph confidence weight to find subgraph features which maximally separate graphs in MGP. To this end, the subgraph feature set should comply to the following constraints.

1) *Weighted Must-Link:* For two graphs selected from the MGP, if they have the same label, they should have a high similarity in the feature space. In addition, considering each graph in puMGL is assigned with a confidence weight, denoted by $\hat{w}_i$, the subgraph features should make sure that graphs with similar weight values are close to each other.
2) *Weighted Cannot-Link:* Graphs with different labels should be separated from each other, as far as possible. Meanwhile, two graphs with different classes but

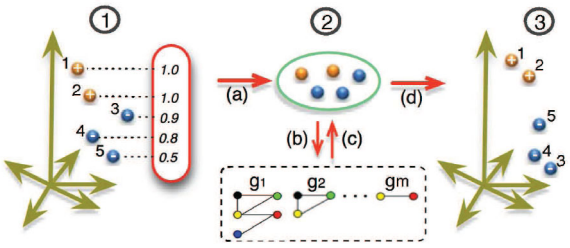

Fig. 3. Confidence weight embedding aims to ② discover optimal subgraphs to ① represent graphs to a ③ new space with the following constraints: graphs with the same label are close to each other, especially for graphs with similar confidence weights (e.g., two blue balls with their flags 3 and 4), and graphs with different labels are separated from each other, especially for graphs with similar confidence weights (e.g., two balls with their flags 1 and 3). The process starts from (a) use instance distributions (b) to discover optimal subgraphs, and (c) further represents (d) graphs in vector space.

similar weight value should also be separated far way from each other [29], [30].

Combining the above constraints, the subgraph feature estimation $\mathcal{J}(\mathbf{g}, \hat{\mathbf{w}})$ is formulated as

$$\mathcal{J}(\mathbf{g}, \hat{\mathbf{w}}) = \frac{1}{2}\sum_{i,j} K_{\mathbf{g}}\left(\hat{G}_i, \hat{G}_j\right)\hat{W}_{i,j}. \tag{2}$$

In (2), $\hat{W}_{i,j}$ embeds weight value information between $\hat{G}_i$ and $\hat{G}_j$ selected from MGP. $K_{\mathbf{g}}(\hat{G}_i, \hat{G}_j)$ calculates the graph distance between $\hat{G}_i$ and $\hat{G}_j$, with subgraph feature set $\mathbf{g}$ being used to represent each graph in the vector space [detailed in (3)].

The above optimization problem can be treated as a nonlinear nonconvex issue, which is difficult to solve. An alternative approach is to optimize the two variables $\mathbf{g}$ and $\hat{\mathbf{w}}$ in an alternative way. In the following, we first explain subgraph feature $\mathbf{g}$ selection and weight updating $\hat{\mathbf{w}}$, and then combine two parts to form a closed-loop framework.

2) Subgraph Features: In order to describe subgraph feature exploration, we introduce the following notations in the description.

1) $\hat{\mathcal{X}}$: $\hat{\mathcal{X}}$ can be denoted by $\hat{\mathcal{X}} = [\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_p] = [\boldsymbol{f}_{g_1}, \ldots, \boldsymbol{f}_{g_s}]^\top \in \{0,1\}^{p\times s}$, which indicates the feature representation matric of the graphs in MGP, with a row vector representing a single graph representation. Besides, $\boldsymbol{f}_{g_k} = [f_{g_k}^{\hat{G}_1}, \ldots, f_{g_k}^{\hat{G}_p}]^\top$ $(g_k \in S_g)$ is an indicator vector based on MGP, i.e., $\{\hat{G}_1, \ldots, \hat{G}_p\}$, where $f_{g_k}^{\hat{G}_i} = 1$ iff $g_k \subseteq \hat{G}_i$ and $f_{g_k}^{\hat{G}_i} = 0$, otherwise.
2) *A and B:* $A = \{(i,j)|y_i y_j = 1\}$ represents the pairwise set under weighted must-link constraint from $\mathcal{G}$ with $B = \{(i,j)|y_i y_j = -1\}$ indicating pairwise set for weighted cannot-link constraint.

a) Confidence weight embedding: In order to take full advantage of the confidence weight information of graphs, we propose an embedding strategy, as shown in Fig. 3. In summary, the embedding process is to utilize weighted label distributions to help find informative subgraph features to represent graphs.

In order to calculate $\hat{W}_{i,j}$ in (2), we employ a radial basis kernel function to estimate the $\hat{W}_{i,j} = \langle \hat{w}_i y_i, \hat{w}_j y_j \rangle$. Moreover, $K_{\mathbf{g}}(\hat{G}_i, \hat{G}_j)$ can be formulated as

$$K_{\mathbf{g}} = \begin{cases} -\left\|\mathcal{D}_{\mathbf{g}}\hat{\mathbf{x}}_i - \mathcal{D}_{\mathbf{g}}\hat{\mathbf{x}}_j\right\|^2/|A|, y_i y_j = 1 \\ \left\|\mathcal{D}_{\mathbf{g}}\hat{\mathbf{x}}_i - \mathcal{D}_{\mathbf{g}}\hat{\mathbf{x}}_j\right\|^2/|B|, y_i y_j = -1. \end{cases} \tag{3}$$

In (3), $\mathcal{D}_{\mathbf{g}}$ (i.e., a diagonal matrix) can be represented as $\mathrm{diag}(d(\mathbf{g}))$, where diagonal elements indicate subgraphs $\mathbf{g}$ which are selected from the set S_g for further graph representation. $d(\mathbf{g})_i = I(g_i \in \mathbf{g})$ denotes the vector representation of each graph by using an indicator function $I(\cdot)$, which returns value 1 iff the condition (i.e., $g_i \in \mathbf{g}$) is satisfied. In addition, we use $M = [M_{ij}]^{p\times p}$ as a confidence weight embedded matrix to the conversion of formulas, where $M_{ij} = \{-\hat{W}_{i,j}/|A|, y_i y_j = 1; \hat{W}_{i,j}/|B|, y_i y_j = -1\}$. Accordingly, (2) can be rewritten as follows:

$$\begin{aligned} \mathcal{J}(\mathbf{g}, \hat{\mathbf{w}}) &= \frac{1}{2}\sum_{i,j}\left\|\mathcal{D}_{\mathbf{g}}\hat{\mathbf{x}}_i - \mathcal{D}_{\mathbf{g}}\hat{\mathbf{x}}_j\right\|^2 M_{i,j} \\ &= \mathrm{tr}\left(\mathcal{D}_{\mathbf{g}}^\top \hat{\mathcal{X}}(D-M)\hat{\mathcal{X}}^\top \mathcal{D}_{\mathbf{g}}\right) \\ &= \mathrm{tr}\left(\mathcal{D}_{\mathbf{g}}^\top \hat{\mathcal{X}} L \hat{\mathcal{X}}^\top \mathcal{D}_{\mathbf{g}}\right) \\ &= \sum_{g_k\in\mathbf{g}} \boldsymbol{f}_{g_k}^\top L \boldsymbol{f}_{g_k} \end{aligned} \tag{4}$$

where D, as a diagonal matrix, is generated from M with $D_{i,i} = \sum_j M_{ij}$. The operator $\mathrm{tr}(\cdot)$ is the trace of a given matrix. Moreover, L is a Laplacian matrix, denoted by $[L_{i,j}]^{p\times p} = D - M$. By using function $z(g_k, L)$ to denote the $\boldsymbol{f}_{g_k}^\top L \boldsymbol{f}_{g_k}$, the original optimization problem in (1) can be translated to maximize the sum of $z(g_k, L)$ with respect to optimal subgraph set $\mathbf{g}$ as

$$\max_{\mathbf{g}} \sum_{g_k\in\mathbf{g}} z(g_k, L) \qquad \text{s.t.} \quad \mathbf{g} \subseteq S_g, |\mathbf{g}| \le m. \tag{5}$$

3) Subgraph Exploration:

Definition 5 (puScore): Given a PU graph bag set $\mathcal{B}$, with M denoting the related confidence weight embedding matrix. $L = D - M$ is a Laplacian matrix. For a given subgraph $g_k \in S_g$, its informativeness score can be estimated as follows:

$$r(g_k) = z(g_k, L) = \boldsymbol{f}_{g_k}^\top L \boldsymbol{f}_{g_k}. \tag{6}$$

In (6), L is a positive semi-definite Laplacian matrix [31], and $\boldsymbol{f}_{g_k}$ is a nonnegative vector with binary value 0 or 1. For a given subgraph $g_k \in S_g$, the formula $r(g_k) = \boldsymbol{f}_{g_k}^\top L \boldsymbol{f}_{g_k}$ is greater than or equal to zero. In this case, the solution to the optimization problem in (5) is to select top-m subgraph features $\mathbf{g} = \{g_1, \ldots, g_m\}$ from $\mathcal{S}_g$ according to their descending order [e.g., $r(g_1) \ge r(g_2) \ge \cdots \ge r(g_s)$] of the subgraph puScore.

One straightforward solution to identify the final optimal subgraph set is to employ an exhaustive enumeration strategy, which enumerates all subgraphs of a given graph and calculates its puScore for ranking. Nevertheless, the number of subgraph candidates increases exponentially with respect to the size of the search space (i.e., the graph set collected from the graph bag set). The huge runtime consumption makes any greedy subgraph searching methods impractical for real-world learning tasks. Accordingly, we apply gSpan [26], which is

an efficient subgraph mining approach based on depth-first search (DFS) strategy, to find subgraph feature candidates. gSpan establishes a lexicographic order to encode each graph, through which all frequent subgraphs are discovered. Some recent graph classification approaches [13], [31] incorporate constraints to prune the search space of gSpan. In this paper, we derive a puScore upper bound so as to prune the DFS code tree (i.e., subgraph searching space) as:

Theorem 1 (Upper Bound of puScore): Given two subgraphs g_k, $g_{k'} \in S_g$, g'_k is a supergraph of g_k (i.e., $g'_k \supseteq g_k$). The puScore value g'_k ($r(g'_k)$) is bounded by $\hat{r}(g_k)$, i.e., $r(g'_k) \leq \hat{r}(g_k)$, where $\hat{r}(g_k)$ is defined as follows:

$$\hat{r}\big(g'_k\big) \triangleq \boldsymbol{f}_{g_k}^{\top} \hat{L} \boldsymbol{f}_{g_k} \tag{7}$$

where the matrix $\hat{L}$ is defined as $\hat{L}_{ij} \triangleq \max(0, L_{ij})$.

Proof:

$$r\big(g'_k\big) = \boldsymbol{f}_{g'_k}^{\top} L \boldsymbol{f}_{g'_k} = \sum_{i,j:\hat{G}_i,\hat{G}_j \in \mathcal{G}(g'_k)} L_{ij} \tag{8}$$

where $\mathcal{G}(g'_k) \triangleq \{\hat{G}_j | g'_k \subseteq \hat{G}_j, 1 \leq j \leq p\}$. Since g_k is the subgraph of g'_k (i.e., $g'_k \supseteq g_k$), according to the anti-monotonic property, we have $\mathcal{G}(g'_k) \subseteq \mathcal{G}(g_k)$. Besides, $\hat{L}_{ij} \triangleq \max(0, L_{ij})$, so $\hat{L}_{ij} \geq L_{ij}$ and $\hat{L}_{ij}$ is greater than or equal to zero. Thus, (8) can be rewritten as

$$\begin{aligned} r\big(g'_k\big) &= \sum_{i,j:\hat{G}_i,\hat{G}_j \in \mathcal{G}(g'_k)} L_{ij} \leq \sum_{i,j:\hat{G}_i,\hat{G}_j \in \mathcal{G}(g'_k)} \hat{L}_{ij} \\ &\leq \sum_{i,j:\hat{G}_i,\hat{G}_j \in \mathcal{G}(g_k)} \hat{L}_{ij} = \boldsymbol{f}_{g_k}^{\top} \hat{L} \boldsymbol{f}_{g_k} = \hat{r}(g_k). \end{aligned} \tag{9}$$

Thus, for any $g'_k \supseteq g_k$, $r(g'_k) \leq \hat{r}(g_k)$. ■

From (6), we can obtain that if $\hat{r}(g_k) \leq \tau$, for any $g'_k \supseteq g_k$, we have $r(g'_k) \leq \hat{r}(g_k) \leq \tau$. This upper bound can be utilized to prune DFS code tree in gSpan by using branch-and-bound pruning. Together with the proposed confidence embedding discussed in Section IV-B2a, the complete subgraph feature exploration approach is listed in Algorithm 1. In summary, the algorithm enumerates subgraph features by searching the whole DFS code tree. If a current subgraph g_k is infrequent, both g_k and its related subtree will be discarded (lines 4 and 5). If not, the puScore of the g_k [i.e., $r(g_k)$] will be calculated based on the matrix L, which embeds the confidence weight distribution information. If $r(g_k)$ is greater than the minimum puScore in $\mathbf{g}$ as τ, or the size of $\mathbf{g}$ is less than m (i.e., $\mathbf{g}$ is not full), g_k will be selected as one item in $\mathbf{g}$ (lines 8 and 9). If $\mathbf{g}$ overflows, one subgraph with the smallest puScore value is removed to maintain its size (lines 10 and 11). Subsequently, the upper bound pruning module will check if $\hat{r}(g_k)$ is less than the threshold τ. If so, it means that the puScore value of any supergraph g'_k of g_k (i.e., $g'_k \supseteq g_k$) will not be greater than τ. Therefore, the subtree rooted from g_k is safely pruned. If $\hat{r}(g_k)$ is indeed greater than the threshold τ, the search process will sequentially visit nodes from the subtree of g_k (lines 13 and 14).

Algorithm 1 ESE: Embedding Subgraph Exploration

Input:

$\mathcal{G}$: A graph set in MGP with confidence weight $\hat{\mathbf{w}}$
min_sup: The threshold of the frequent subgraph;
m: the number of subgraph features to be selected;

Output:

$\mathbf{g} = \{g_1, \cdots, g_m\}$: A set of subgraph features;

1: $\mathbf{g} = \emptyset$, $\tau = 0$;
2: **while** Recursively visit the DFS Code Tree in gSpan **do**
3: $g_k \leftarrow$ current visited subgraph in DFS tree of $\mathcal{G}$;
4: **if** $freq(g_k) < min\_sup$, **then**
5: **return**;
6: $L \leftarrow$ Apply $\hat{\mathbf{w}}$ to $\mathcal{G}$ and obtain the embedding matrix;
7: $r(g_k) \leftarrow$ Apply L to compute puScore of subgraph g_k;
8: **if** $|\mathbf{g}| < m$ or $r(g_k) > \tau$, **then**
9: $\mathbf{g} \leftarrow \mathbf{g} \bigcup g_k$;
10: **if** $|\mathbf{g}| \geq m$, **then**
11: $\mathbf{g} \leftarrow \mathbf{g}/\arg\min_{g_i \in \mathbf{g}} r(g_i)$;
12: $\tau = \min_{g_i \in \mathbf{g}} r(g_i)$;
13: **if** $\hat{r}(g_k) \geq \tau$, **then**
14: Depth-first search the subtree rooted from node g_k;
15: **end while**
16: **return** $\mathbf{g}$;

4) Confidence Weight Optimization: By following the above process, the optimal subgraphs $\mathbf{g}$ can be obtained. In the next step, we resolve the second research subproblem to optimize the confidence weight $\hat{\mathbf{w}}$, which is identified by the output probability from the classifier built on the MGP (we use SVMs in our experiments). The technical details about the construction of MGP will be explained in the subsequent section, followed by the procedure of bag confidence weight optimization.

a) Margin graph pool: In the proposed puMGL framework, a number of most positive graphs from positive bag set and least negative graphs from reliable negative graph bag set are selected to form an MGP, through which the classifier is built to distinguish positive and negative bags. The motivation of constructing MGP is the margin concept, where samples near to the decision boundary play an important role for differentiating samples in different classes. Specifically, a confidence weight will be assigned to each unlabeled bag, through which some reliable negative bags are identified to construct MGP. During this process, a core issue is to find proper models to assess graphs in the MGP. Fortunately, the distribution in negative bag set is known, because graphs in a negative bag are genuinely negative in the multi-graph setting. Accordingly, a weighted kernel density estimator [32] is employed to model the distribution of negative instances (representation by a given subgraph set) in reliable negative bags as follows:

$$p\big(\mathbf{x}|\mathcal{X}^-\big) = \frac{1}{\sum_i n_i^-} \sum_{i,j} K\Big(w_x \mathbf{x}, w_j \mathbf{x}_{i,j}^-\Big) \tag{10}$$

where w_x represents the weight value of the bag which contains graph x. $\mathbf{x}_{i,j}^-$ indicates the graph representation of the jth graph in the ith reliable negative bag with n_i^- graphs. K, as

Algorithm 2 CWO: Confidence Weight Optimization

Input:
$\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{B}}^{+} \cup \boldsymbol{\mathcal{B}}^{u}$: An graph bag data set;
$\mathbf{g}$: subgraph features; $\mathcal{G}$: A graph set inMGP;
Output:
$\hat{\mathbf{w}}$: A set of confidence weight for MGP;
// Unlabeled Bag Weight Optimization:
$\hat{\mathcal{X}} \leftarrow$ Apply $\mathbf{g}$ to $\mathcal{G}$ to obtain subgraph feature vectors.
$\mathcal{H} \leftarrow$ Apply $\hat{\mathcal{X}}$ to build the classifier.
for each bag B_i in $\boldsymbol{\mathcal{B}}^{u}$ **do**
  for each graph $G_{i,j}$ in B_i **do**
    $\mathbf{x}_{i,j} \leftarrow$ Apply $\mathbf{g}$ to $G_{i,j}$ to obtain its feature vector.
    $p_{i,j} \leftarrow$ Apply $\mathcal{H}$ to $\mathbf{x}_{i,j}$ and estimate probability;
  end for
  $w_i \leftarrow \sum_{j}^{n_i} p_{i,j}/n_i$;
end for
// MGP Weight Optimization:
$\boldsymbol{\mathcal{B}}^{-} \leftarrow$ Apply $\mathbf{w}$ to $\boldsymbol{\mathcal{B}}^{u}$ to form reliable negative bag set.
$\mathcal{D} \leftarrow$ Generate a distribution from $\boldsymbol{\mathcal{B}}^{-}$ via Eq. (10).
$\hat{\mathbf{w}} \leftarrow$ Apply $\mathcal{D}$ to $\boldsymbol{\mathcal{B}}^{+}$, $\boldsymbol{\mathcal{B}}^{-}$ and update $\mathcal{G}$ via Eq. (11).
return $\hat{\mathbf{w}}$;

a Gaussian kernel function, is adopted to estimate the similarity between two instances (i.e., the vector representation of two graphs). $\mathcal{X}^{-}$ denotes the set, consisting of the graph feature representation vectors in reliable negative bags $\mathcal{B}^{-}$. Accordingly, the most positive pattern or least negative pattern $\mathbf{x}_i^{\rho}$ in MGP can be formulated as

$$\mathbf{x}_i^{\rho} = \arg \min_{\mathbf{x}_{i,j} \in X_i, j=1,\ldots,n_i} p\left(\mathbf{x}_{i,j} | \mathcal{X}^{-}\right) \tag{11}$$

where X_i denotes the feature vector set collected from all the graph representation using subgraph set $\mathbf{g}$ in the ith bag.

b) Confidence weight updating: Arguably, not all initial subgraphs have good discriminative power, because graphs in MGP set are based on the initial random weight values. As the learning process continues, the underlying subgraph feature set will have a better quality, so the weights of all graphs in the unlabeled set should be updated according to the re-evaluation results derived from the classifier trained from MGP. This is because a few unlabeled graphs, which are further selected to be reliable negative set, may not be negative because subgraph features are not reliable in a previous iteration. In this case, the confidence weight updating strategy will make the reliable negative set more accurate, through which the quality of mined subgraph feature set can also be improved. In our problem setting, the positive bags are given (i.e., the related conference weight value is set to 1.0), so only the weights of unlabeled bags need to be evaluated. By doing so, we can ensure that reliable negative set will have continuously improved quality, and also avoid introducing noise to positive bags.

For a bag containing a set of graphs, the following strategy will be designed to calculate its probability of being negative. We first utilize the approach in [15] to calculate the probability

Algorithm 3 puMGL: PU Multi-Graph Learning

Input:
$\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{B}}^{+} \cup \boldsymbol{\mathcal{B}}^{u}$: An graph bag data set;
min_sup: The threshold of the frequent subgraph;
m: the number of subgraph features to be selected;
Output:
The target class label y_t of a test bag B_t.
// Training Phase:
Set the labels of the unlabeled bags $\mathcal{B}_j^{u}$ to be -1;
$\mathcal{G} \leftarrow$ Initialize MGP $\mathcal{G}$ by randomly selecting one graph from each positive bag $B_i \in \boldsymbol{\mathcal{B}}^{+}$ and each unlabeled bag $B_j \in \boldsymbol{\mathcal{B}}^{u}$, respectively;
while not convergence for $\hat{\mathbf{w}}$ **do**
  // Optimal Subgraph Features:
  $\mathbf{g} \leftarrow ESE(m, min\_sup, \mathcal{G}, \hat{\mathbf{w}})$; //Algorithm 1
  $\mathcal{G} \leftarrow$ Apply $\mathbf{g}$ to represent the graphs in MGP $\mathcal{G}$.
  // Confidence Weight Optimization:
  $\hat{\mathbf{w}} \leftarrow CWO(\boldsymbol{\mathcal{B}}, \mathbf{g}, \mathcal{G})$; //Algorithm 2
end while
$\mathbf{g}^{*} \leftarrow \mathbf{g}$; $\hat{\mathbf{w}}^{*} \leftarrow \hat{\mathbf{w}}$; // Optimal subgraphs and weights.
// Test Phase:
$\mathcal{H}^{*} \leftarrow$ Apply $\mathbf{g}^{*}$ and $\hat{\mathbf{w}}^{*}$ to $\mathcal{G}$ to build the classifier.
$\mathbf{x}_{t,i} \leftarrow$ Apply $\mathbf{g}^{*}$ to each $G_{t,i}$ in B_t to obtain its vector.
$Y_t \leftarrow$ Apply $\mathcal{H}^{*}$ to each $\mathbf{x}_{t,i}$ to predict its bag label;
return Y_t;

of each graph in the bag by tackling the optimization problem

$$\min_{\mathbf{p}} \quad \sum_{i}^{k} \sum_{j:j \neq i} \left(r_{\rho_i,\rho_j} p_{\rho_j} - r_{\rho_j,\rho_i} p_{\rho_i}\right)^2$$
$$\text{s.t.} \quad p_{\rho_i} \geq 0, \sum_{i} p_{\rho_i} = 1 \tag{12}$$

where k represents the class number of the training set. In puMGL setting, k is set to 2, i.e., positive ($\rho_1 = +1$) and negative ($\rho_2 = -1$). $r_{\rho_i,\rho_j} = P(y = \rho_i | y = \rho_i \text{ or } \rho_j)$ denotes the pairwise class probabilities of ρ_i and ρ_j. Besides, p_{ρ_i} is the probability estimation of ρ_i. The value of p_{ρ_2} is used as the confidence weight value for each graph in unlabeled set.

For a graph $G_{i,j} \in \mathcal{B}_i^{u}$, its probability of being classified as negative is denoted by $p_{i,j}$. Meanwhile, the corresponding confidence weight for the bag $B_i \ni G_{i,j}$ can be calculated as $w_i = \sum_{j}^{n_i} p_{i,j}/n_i$ (i.e., the confidence bag weight $\mathbf{w}$ update). Accordingly, unlabeled bags with high confidence weights are considered as reliable negative bags to further construct MGP. In addition, the weight vector consisting of all confidence weights $\hat{w}_i$ for each graph collected from MGP is denoted by $\hat{\mathbf{w}}$. Algorithm 2 outlines the confidence weight optimization with two major parts including unlabeled bag confidence weight optimization followed by MGP updating.

C. puMGL Algorithm

We can now utilize solutions in Sections IV-B2 (subgraph features) and IV-B4 (confidence weight optimization), respectively, to optimize the evaluation criterion in (2). The complete procedures of the proposed puMGL framework are listed in Algorithm 3. For initialization, the labels of unlabeled bags

are set to be -1, and the affiliated confidence weights w_j are randomly set within $(0, 1]$. After that, puMGL initializes the MGP by randomly selecting one graph from each positive bag B_i^+ and each unlabeled bag B_j^-, respectively. By doing so, we intend to make the initial instance distributions in MGP similar to the bag distributions, and avoid some bags dominating the MGP (lines 1 and 2).

During the while loop, because graphs in MGP have no feature values in the vector space, puMGL first explores initial subgraph features using Algorithm 1 (line 4). Because subgraphs in the first loop is not optimal due to the random initial weights used to construct MGP, we can use subgraph features to transfer graphs in MGP into vector space, and train SVM classifiers to classify graphs in unlabeled bags. As a result, the confidence weight $\mathbf{w}$ can be updated and help obtain better confidence weight $\hat{\mathbf{w}}$ for MGP in the next iteration (line 6). By iteratively running the producers "subgraph feature optimization" (Algorithm 1) and "confidence weight optimization" (Algorithm 2), the optimal subgraph set $\mathbf{g}^*$ and confidence weights $\hat{\mathbf{w}}^*$ can be obtained (line 8), until no further change occurs in the confidence weight $\hat{\mathbf{w}}$.

During the testing phase, the feature representation $\mathbf{x}_{t,i}$ for all graphs $G_{t,i}$ in test bag B_t based on $\mathbf{g}^*$ will be classified by the model $\mathcal{H}^*$ to obtain its bag label y_t under the MGL setting. A bag is classified as positive only if one or multiple graphs inside the bag are classified as positive, and negative otherwise (lines 9–11).

The runtime complexity of puMGL is mainly attributed to the alternating optimization between: 1) subgraph feature optimization and 2) confidence weight optimization. For the former, the calculation of puScore will cost $O(p^2)$, with p representing the size of MGP. The subgraph mining based on puScore will take $O(l(p)+s\cdot p^2)$, where s is the number of subgraphs, with l being the function based on the total number of vertices and edges in MGP. For the later, the construction of the MGP has the complexity of $O(\log p \cdot p \cdot n^-)$, with n^- denoting the number of graphs in reliable negative bags. The confidence weight updating will cost $O(p^2+n^u)$, with n^u denoting the number of graphs in unlabeled bags. Assume the number of iterations during the training process is M, the total runtime complexity of puMGL is $M \times (O(l(p)+s\cdot p^2+\log p \cdot p\cdot n^- + p^2 + n^u) \leq O(l(p)+s\cdot p^2\cdot n^- + n^u))$.

V. Experiments

A. Benchmark Data Sets

1) Bio-Pharmaceutical Data: For bio-pharmaceutical activity test, labeling individual molecules (which are commonly represented as graphs) is expensive and time-consuming [33]. To reduce labeling costs, molecular group activity prediction can be used to investigate activities of a group (i.e., bag) of molecules. Moveover, for molecular activity detection on a given disease, emphasis can be focused on molecules whose outcomes are active (i.e., positive) with respect to certain testing conditions. In this case, detailed investigations, on individual graph, are carried out on active molecule group only (i.e., a positive bag).

In this section, we report multi-graph learning task for chemical compound anti-cancer activity prediction on "nonsmall cell lung" cancer. The anti-cancer activity prediction cancer screening data sets [i.e., National Cancer Institute (NCI)] are commonly used as a graph classification benchmark (http://pubchem.ncbi.nlm.nih.gov). Each NCI data set belongs to a bio-assay task for anti-cancer activity prediction, where each chemical compound is a graph, with atoms representing nodes and bonds as edges. We build a multi-graph data set with PU graph bags by using NCI data set with ID 1. To build active molecule bag (i.e., a bag with multiple graphs), we randomly select 1–4 active graphs and several inactive graphs to form an active bag. An unlabeled bag is formed by randomly selecting a number of inactive graphs to form a bag. The number of graphs in each bag varies from 1 to 10. In total, we build 300 positive bags with 1460 graphs and 300 unlabeled bags with 1640 graphs.

2) Online Product Recommendation Data: Online product recommendation learning task includes beer review data set collected from http://snap.stanford.edu/data/ (Stanford Large Network Dataset Collection). The data set includes reviews of different brand of beer products in Amazon.com. Each review report consists of the following information, including beer ID, reviewer ID, product score (varying from 1 to 5), and reviewers' detailed comments [34]. Because each beer brand may receive multiple reports from different reviewers, we regard that a beer product is potentially interesting to certain customers, if one or more core characters of the beer, such as "durability" or "affordability" commented in the reports are very good (i.e., the received average review score is ≥ 4). On the other hand, if all review scores are < 4, it implies that this beer product is not favored by customers. Our goal is to utilize the information in the review reports for online product recommendation. More specifically, to represent detailed comment texts as graphs, a fuzzy cognitive map [35] approach is employed to build the graph, where the nodes denote keywords with edges representing the relationships between keywords [29]. During the graph construction, edges with its correlation coefficient less than 0.006 are removed. We select 600 beer products as the benchmark data set, where each beer consists of approximately 1–10 reviews. In summary, we obtain 300 favored products (i.e., positive bags) with 1756 comments (i.e., graphs) and 300 unfavored products (not favored by customers) with 1528 graphs as unlabeled graph bags.

3) Content-Based Image Retrieval Data: The benchmark images are collected from Corel data set [36]. All images are preprocessed by VLFeat system (http://www.vlfeat.org/) to obtain image regions, with each region being converted to a graph. By doing so, each image can be regraded as a bag of graphs. For an individual region, a state-of-the-art superpixel-based method, simple linear iterative clustering [8], is applied to convert the region to a graph, where each node corresponds to one superpixel and each edge represents the adjacency relationship between superpixels. For each region, it consists of a number of superpixels, denoted by the RGB-color histogram. We utilize 16-bins per channel to generate 4096-dimensional histograms. Although one can use histogram features to represent the image region, this feature representation for each superpixel will ignore structure information in images.

Therefore, a clustering process is applied to all superpixels to generate a multi-graph representation for image. In our experiments, the superclass "cats," including three subclasses "tiger," "lion," and leopard is regarded as positive images (300 bags with 2679 graphs). In addition, 300 images from all animals are randomly selected as unlabeled bags, with 2668 segments (i.e., graphs) in total.

4) Scientific Publication Data: The DBLP data set consists of bibliography data in computer science (http://dblp.uni-trier.de/xml/). Each record in DBLP is a paper containing attributes such as abstract, authors, year, venue, title, and references. To build a puMGL task, we select papers published in Machine Learning and Artificial Intelligence conferences (AI: IJCAI, UAI, NIPS, ECML, AAAI, ICML, COLT, ACL, KR, and IJCNN) as positive bags and randomly select papers from all fields, including computer vision, multimedia, and pattern recognition etc., as unlabeled bags. A multi-graph bag representation is used to represent a research paper. More specifically, each paper is converted to an undirected graph by using the correlation of keywords in the abstract with edges denoting keyword correlations. Because each reference cited in this paper also represents a graph, a bag is formed by using graphs built from this paper and references cited in this paper. The graph representation for abstract is similar to the above online product data set. Notice that positive bags (i.e., AI) and unlabeled bags (e.g., computer vision, multimedia, and pattern recognition) may have overlapped research topics, which helps build a challenging puMGL learning task. In our experiments, we choose 600 papers which correspond to 600 bags, with each paper containing 1–10 references. Among all 600 bags (papers), 300 papers in the AI field are selected as positive bags (with 1756 references cited in 300 AI papers). The remaining 300 papers are unlabeled (randomly selected from all fields). As a result, the total number of graphs in PU bags are 1756 and 1755, respectively.

B. Experimental Settings

To validate the effectiveness of the proposed puMGL framework, we use F-score$= 2 \times P \times R/(P + R)$, which combines recall R and precision P, as our performance measure. F-score is popularly used to evaluate the performance of PU learning in previous researches [15]–[17]. We employ LibSVM as the learning algorithm to train classifiers from MGP. For all data sets, 70% of graph bags are used as training set, and the remaining bags are used as testing set. To validate the performance of puMGL with different numbers of positive bags, we randomly choose $r \times 100\%$ (from 10% to 70%) bags that have positive labels as positive bags, and combine remaining positive bags and all other bags as unlabeled set. Unless specified otherwise, the size of subgraph features m is set to 60, with r being 0.4, and min_sup (i.e., minimum support threshold) as 8% for online product review, 3% for region-based image, 15% for NCI bio-pharmaceutical activity test, and 4% for the DBLP Scientific Publication data set. In addition, all reported results are based on the mean accuracy over ten times. All experiments are carried out on a Linux cluster node with an Intel Xeon @3.33 GHz CPU and 3 GB fixed memory size.

C. Baseline Methods

Because no existing methods are available to solve the proposed research problem, we use the following two types of baselines for comparative studies. More specifically, bag-level approaches first use some informative subgraphs to represent graphs in the bag set, so a multi-graph bag is transferred to an MI bag. After that, an existing PU learning strategy [17] is applied to MI bags for learning. Meanwhile, for graph-level methods, we propagate multi-graph bag label to all graphs inside the bag, so all positive bags are converted to positive graphs. After that, the proposed puMGL can be solved by using an existing positive and unlabeled graph learning (puGL) method [15].

1) Bag-Level Methods: For this baseline approach, we first select a number of top-k frequent subgraphs to represent graphs as feature instances, and then convert the multi-graph learning problem to a positive and unlabeled MI learning (puMIL) task. This problem, however, still does not have effective solution. Accordingly, we employ puMIL to directly train MI classifiers by treating unlabeled bags as negative bags, with an additional penetration strategy [17]. More specifically, to handle the lack of the negative bag set, puMIL randomly includes a number of positive graph bags (i.e., "spies") into the unlabeled set, with the whole temporary bag set being regarded as pseudo-negative bags. After that, an MILR [20] is trained for classification, with the output probability estimation results being used to help construct reliable negative bag set. At the final stage, once the algorithm converges, an MI classier MISVM [37] is trained to construct the classifier for classification.

2) Graph-Level Methods: Because genuine labels of graphs in a positive bag are unknown, graph-level methods directly propagate bag labels to all graphs inside each bag. By doing so, the puMGL problem is converted to a graph learning with only puGL task. After that, graph-level approaches build a PU graph classifier [15] to classify all graphs in a test bag. During the iterative process, all graphs inside unlabeled bags are initially treated as reliable negative graphs, through which a dependency evaluation for subgraph features is proposed for optimization. A test bag is classified as positive if one or more graphs inside the bag is classified as positive, and negative otherwise.

We also implement a positive naive Bayes (PNB) [38] based puMGL approach. PNB uses the probability-based Bayesian estimation technique to obtain a set of reliable negatives for further learning. On the other hand, it is also possible to treat the proposed multi-graph PU learning as one-class problem, by simply discarding all unlabeled data. In this case, one-class SVM [39] can be applied for classification.

D. Experimental Results

1) Classification Performance Comparisons With Respect to Different r Values: Fig. 4 reports the classification results (i.e., F-score) with r values varying from 10% to 70% on four different data sets. Overall, the results show that one-class-based approach is unsatisfactory for identifying subgraphs for puMGL, largely because one-class learning does not consider useful information in unlabeled bags but only intends to

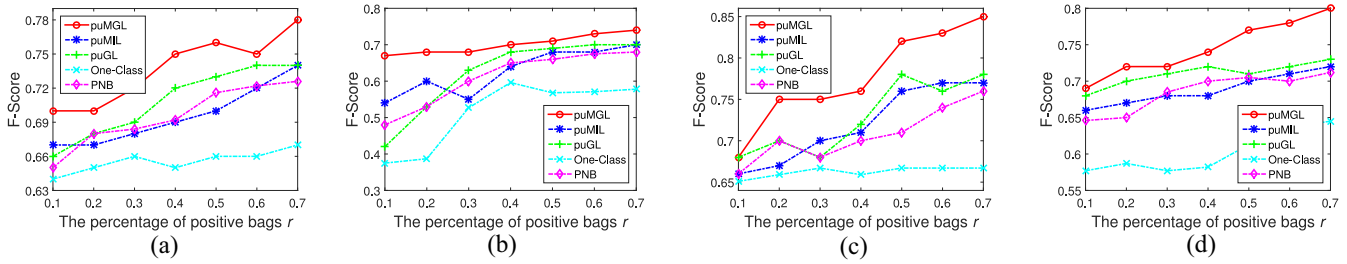

Fig. 4. *F*-score comparisons with respect to different r values. (a) NCI bio-pharmaceutical activity. (b) Online product review. (c) Corel region-based image. (d) DBLP scientific publication.

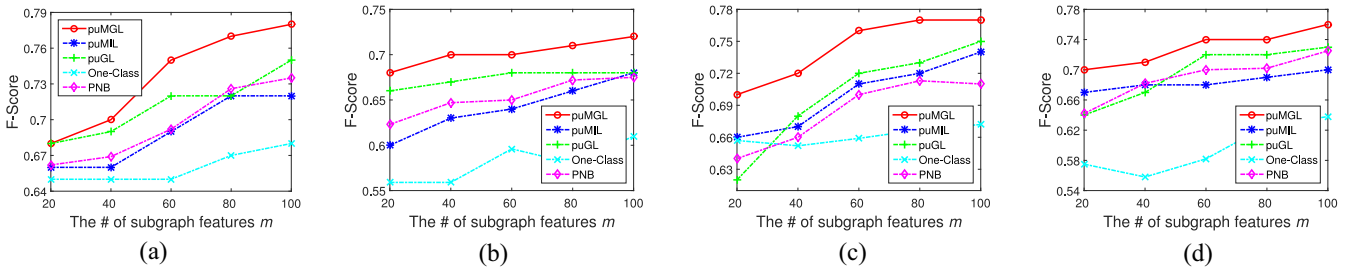

Fig. 5. *F*-score comparisons with respect to different m values. (a) NCI bio-pharmaceutical activity. (b) Online product review. (c) Corel region-based image. (d) DBLP scientific publication.

model the distributions of positive samples. Since genuine positive graphs are unknown in multi-graph setting, samples used to train the one-class classifier are inherently error prone and have many mislabels which significantly deteriorates its performance.

Furthermore, when r value increases, all *F*-score values improve, mainly because a higher percentage of positive bags provide more information for deriving discriminative subgraph features from graphs in the bags. When the number of positive bags is small, i.e., the corresponding percentage r is less than 30%, puMIL, puGL, and PNB achieve the competitive performance. Nevertheless, for large r values, puGL can achieve a superior performance compared to puMIL and PNB. This is possibly because that the increase number of positive bags provide enough knowledge to help dynamically extract infrequent subgraphs with discriminative capability for multi-graph classification, whereas puMIL and PNB just carry out frequent subgraph feature selection in a static way. Obviously, puMGL has the best performance compared to other baselines, especially with a small number of positive bags (e.g., $r \leq 20\%$). This indicates that puMGL can effectively unitize useful information in unlabeled set.

2) Classification Performance Comparisons With Respect to Different m Values: Furthermore, we also demonstrate the performance of puMGL by varying the size of the subgraph set from 20 to 100, as shown in Fig. 5. Similarly, one-class-based approach has the worst performance, which confirms that only utilizing one-class data in the training process cannot result in satisfactory discriminative models. With the increase of subgraph number m, the learning performance also continuously improves. This is mainly because that a larger number of subgraphs potentially provide more useful features for graph representation. Meanwhile, PNB and puMIL both achieve a high learning performance gain with an increase number of m, but these two baselines are inferior to the best performance achieved by puGL. For example, on DBLP scientific publication data set, puMIL outperforms PNB when the r value is less than 40, as shown in Fig. 5(d). As the r value continuously increases, PNB shows significant superiority over puMIL. However, they are all inferior to the proposed puMGL, especially when the number of selected subgraphs is greater than 40. The above analysis demonstrates that puMGL is capable of handling the classification of complicated objects for bio-pharmaceutical activity test.

3) Effectiveness of MGP: A main component in the puMGL framework is the utilization of MGP, consisting of most positive patterns and least negative patterns from positive and reliable negative bag set, respectively. As a result, MGP can utilize those samples, which are close to the decision boundary, to improve the underlying classifier's learning performance.

To validate the quality of MGP, and check whether informative subgraphs are indeed included in MGP, we examine all most positive patterns by using a region representation, and report the results in the first row of Fig. 6. The results show that the subgraph pattern g discovered from MGP is indeed shared by four images in the first row (i.e., the body part of the leopard). This observation indicates that MGP is capable of exploring important regions, with common structure patterns, to represent complicated objects for classification. On the other hand, our experiments also show that not all regions from MGP have good representation capability. In other words, some selected regions (e.g., #10 in the second row in Fig. 6) are not genuinely "positive" due to the interference of the object's surrounding environment or other factors [40].

4) Efficiency of the Pruning Strategy: To evaluate the efficiency of the pruning module of puMGL as described in Section IV-B3, we implement a UpuMGL approach with no pruning module and compare its runtime performance with puMGL, through which we can demonstrate the efficiency of the pruning module. UpuMGL first exploits gSpan to mine a frequent subgraph set, and then finds the optimal subgraph

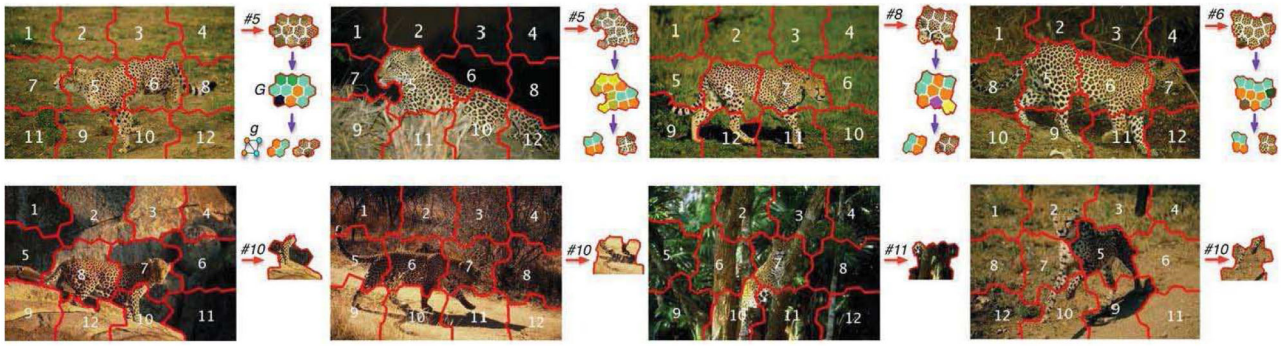

Fig. 6. Some "most positive patterns" examples explored by using the proposed MGP framework. Each item in MGP corresponds to one region (e.g., #5 in the image on the left corner), which is selected under the criterion proposed in Section IV-B4. The objective of building MGP is to mine subgraph feature (e.g., g as shown in the first row) with high puScore for further learning. Example of less positive examples are also reported in the second row.

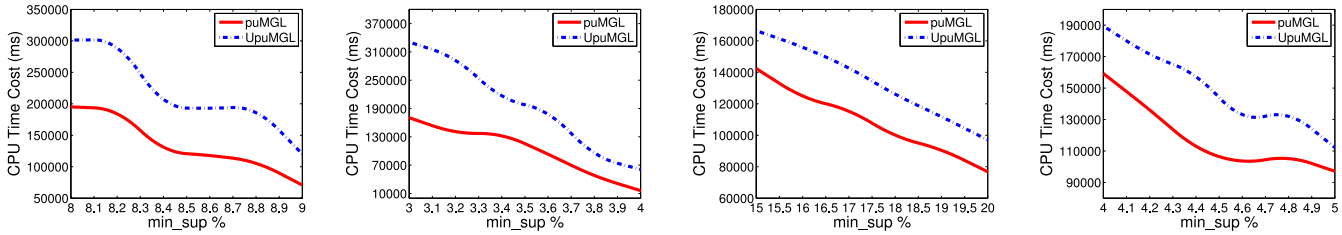

Fig. 7. Average CPU runtime for puMGL versus unpruned UpuMGL with different min_sup under a fixed number of subgraphs $m = 60$ on (a) online product review, (b) Corel region-based image, (c) NCI bio-pharmaceutical activity, and (d) DBLP scientific publication data set, respectively.

features by using the same criteria as puMGL. In Fig. 7(a)–(d), we report the average CPU runtime performance with respect to different minimum support min_sup values (the number of selected subgraphs is fixed to 60) on the above four data sets, respectively. The results show that, as the min_sup values increase, the runtime of both puMGL and UpuMGL decreases, mainly because a large min_sup value will reduce the number of candidates for validation. puMGL demonstrates much better runtime performance than its unpruned version. This is mainly attributed to the pruning module, as shown in Algorithm 1, to dynamically prune the candidate set for better runtime efficiency.

VI. Conclusion

This paper investigated a novel puMGL task for representing and classifying complicated objects containing rich content and structure information. In the proposed multi-graph representation, the object for classification is a graph bag, whose class label is only available at the bag level. We argued that many applications involve multi-graph learning with only PU bags, where the lack of feature-vector representation for graphs and the unavailability of negative bags make the learning problem very challenging. In order to tackle the challenges, we proposed an iterative optimization framework to carefully select some graphs to form MGP which contains positive graphs and reliable negative graphs. After that, a set of discriminative subgraph features are extracted from MGP to represent graphs for multi-graph classification. Experiments and comparisons on real-world tasks demonstrated that the proposed MGP-based puMGL framework significantly outperforms baseline methods.

References

[1] J. Cao, Z. Wu, J. Wu, and H. Xiong, "SAIL: Summation-based incremental learning for information-theoretic text clustering," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 570–584, Apr. 2013.

[2] D. Song, F. Sun, and L. Liao, "A hybrid approach for content extraction with text density and visual importance of DOM nodes," *Knowl. Inf. Syst.*, vol. 42, no. 1, pp. 75–96, 2015.

[3] X. Zhu, X. Li, and S. Zhang, "Block-row sparse multiview multilabel learning for image classification," *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 450–461, Feb. 2016.

[4] O. Frunza, D. Inkpen, and T. Tran, "A machine learning approach for identifying disease-treatment relations in short texts," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 6, pp. 801–814, Jun. 2011.

[5] T. G. Dietterich, R. H. Lathrop, and T. L. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artif. Intell.*, vol. 89, nos. 1–2, pp. 31–71, 1997.

[6] Z.-H. Zhou, M.-L. Zhang, S.-J. Huang, and Y.-F. Li, "Multi-instance multi-label learning," *Artif. Intell.*, vol. 176, no. 1, pp. 2291–2320, 2012.

[7] J. Amores, "MILDE: Multiple instance learning by discriminative embedding," *Knowl. Inf. Syst.*, vol. 42, no. 2, pp. 381–407, 2015.

[8] R. Achanta *et al.*, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[9] 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.

[10] J. Wu *et al.*, "Multi-graph-view subgraph mining for graph classification," *Knowl. Inf. Syst.*, pp. 1–26, Sep. 2015, doi: 10.1007/s10115-015-0872-1.

[11] M. Flores-Garrido, J.-A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "AGraP: An algorithm for mining frequent patterns in a single graph using inexact matching," *Knowl. Inf. Syst.*, vol. 44, no. 2, pp. 385–406, 2015.

[12] 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.

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

[14] H. Cheng, X. Yan, J. Han, and P. S. Yu, "Direct discriminative pattern mining for effective classification," in *Proc. ICDE*, Cancún, Mexico, 2008, pp. 169–178.

[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] X. Li and B. Liu, "Learning to classify texts using positive and unlabeled data," in *Proc. IJCAI*, Acapulco, Mexico, 2003, pp. 587–592.
[17] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, "Building text classifiers using positive and unlabeled examples," in *Proc. ICDM*, Melbourne, FL, USA, 2003, pp. 179–188.
[18] F. Briggs, X. Z. Fern, and R. Raich, "Context-aware MIML instance annotation: Exploiting label correlations with classifier chains," *Knowl. Inf. Syst.*, vol. 43, no. 1, pp. 53–79, 2015.
[19] M. Wang, Y. Gao, K. Lu, and Y. Rui, "View-based discriminative probabilistic modeling for 3D object retrieval and recognition," *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1395–1407, Apr. 2013.
[20] S. Ray and M. Craven, "Supervised versus multiple instance learning: An empirical comparison," in *Proc. ICML*, Bonn, Germany, 2005, pp. 697–704.
[21] K. Zhou, X. Gui-Rong, Q. Yang, and Y. Yu, "Learning with positive and unlabeled examples using topic-sensitive PLSA," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 1, pp. 46–58, Jan. 2010.
[22] B. Liu, W. S. Lee, P. S. Yu, and X. Li, "Partially supervised classification of text documents," in *Proc. ICML*, Sydney, NSW, Australia, 2002, pp. 387–394.
[23] J. T. Zhou, S. J. Pan, Q. Mao, and I. W. Tsang, "Multi-view positive and unlabeled learning," in *Proc. ACML*, Singapore, 2012, pp. 555–570.
[24] N. An *et al.*, "Toward detection of aliases without string similarity," *Inf. Sci.*, vol. 261, pp. 89–100, Mar. 2014.
[25] K. Riesen and H. Bunke, "Graph classification by means of Lipschitz embedding," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1472–1483, Dec. 2009.
[26] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," in *Proc. ICDM*, Maebashi City, Japan, 2002, pp. 721–724.
[27] M. Wang *et al.*, "Unified video annotation via multigraph learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 5, pp. 733–746, May 2009.
[28] X. Yang, M. Wang, and D. Tao, "Robust visual tracking via multi-graph ranking," *Neurocomputing*, vol. 159, pp. 35–43, Jul. 2015.
[29] J. Wu, X. Zhu, C. Zhang, and Z. Cai, "Multi-instance multi-graph dual embedding learning," in *Proc. ICDM*, Dallas, TX, USA, 2013, pp. 827–836.
[30] 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.
[31] X. Kong and P. S. Yu, "Semi-supervised feature selection for graph classification," in *Proc. KDD*, Washington, DC, USA, 2010, pp. 793–802.
[32] Z. Fu, A. Robles-Kelly, and J. Zhou, "MILIS: Multiple instance learning with instance selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 958–977, May 2011.
[33] 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.
[34] J. J. McAuley and J. Leskovec, "From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews," in *Proc. WWW*, Rio de Janeiro, Brazil, 2013, pp. 897–908.
[35] X. Luo, Z. Xu, J. Yu, and X. Chen, "Building association link network for semantic link on Web resources," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 3, pp. 482–494, Jul. 2011.
[36] J. Li and J. Z. Wang, "Real-time computerized annotation of pictures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 6, pp. 985–1002, Jun. 2008.
[37] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning," in *Proc. NIPS*, Whistler, BC, Canada, 2003, pp. 577–584.
[38] F. Denis, R. Gilleron, and M. Tommasi, "Text classification from positive and unlabeled examples," in *Proc. IPMU*, Montpellier, France, 2002, pp. 1927–1934.
[39] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, 2001.
[40] K. Wang, N. An, B. N. Li, Y. Zhang, and L. Li, "Speech emotion recognition using Fourier parameters," *IEEE Trans. Affect. Comput.*, vol. 6, no. 1, pp. 69–75, Jan./Mar. 2015.

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

He is currently a Research Associate with the Centre of Quantum Computation and Intelligent Systems, UTS. His current research interests include data mining and machine learning. Since 2009, he has published over 20 refereed journal and conference papers, such as in the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the IEEE TRANSACTIONS ON CYBERNETICS, *Pattern Recognition*, IJCAI, the IEEE International Conference on Data Mining, SDM, and CIKM, in the above areas.

Shirui Pan 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 of 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*, IJCAI, the IEEE International Conference on Data Mining, SDM, CIKM, PAKDD, and IJCNN. His current research interests include data mining and machine learning.

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 multimedia systems. Since 2000, he has published over 200 refereed journal and conference papers in the above areas.

Dr. Zhu was a recipient of two Best Paper Awards and one Best Student Paper Award. He was an Associate Editor of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING from 2008 to 2012. He has been an Associate Editor of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING since 2014.

Chengqi Zhang (SM'95) received the Ph.D. degree from the University of Queensland, Brisbane, QLD, Australia, in 1991, and the D.Sc. degree (higher doctorate) 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 in Sydney, the Local Arrangements Chair of IJCAI 2017 in Melbourne, VIC, Australia, and a fellow of the Australian Computer Society.

Xindong Wu (F'11) received the Ph.D. degree in artificial intelligence from the University of Edinburgh, Edinburgh, U.K.

He is a Professor of Computer Science with the University of Vermont, Burlington, VT, USA, and a Yangtze River Scholar with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, China. His current research interests include data mining and big data analytics.

Prof. Wu is the Steering Committee Chair of the IEEE International Conference on Data Mining and the Editor-in-Chief of *Knowledge and Information Systems*. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING from 2005 to 2008. He is a fellow of the American Association for the Advancement of Science.