Multi-Graph Learning with Positive and Unlabeled Bags

Jia Wu * ‡ Zhibin Hong * Shirui Pan * Xingquan Zhu† Chengqi Zhang * Zhihua Cai‡

Abstract

In this paper, we formulate a new multi-graph learning task with only positive and unlabeled bags, where labels are only available for bags but not for individual graphs inside the bag. This problem setting raises significant challenges because bag-of-graph setting does not have features to directly represent graph data, and no negative bags exits for deriving discriminative classification models. To solve the challenge, we propose a puMGL learning framework which relies on two iteratively combined processes for multi-graph learning: (1) deriving features to represent graphs for learning; and (2) deriving discriminative models with only positive and unlabeled graph bags. For the former, we derive a subgraph scoring criterion to select a set of informative subgraphs to convert each graph into a feature space. To handle unlabeled bags, we assign a weight value to each bag and use the adjusted weight values to select most promising unlabeled bags as negative bags. A margin graph pool (MGP), which contains some representative graphs from positive bags and identified negative bags, is used for selecting subgraphs and training graph classifiers. The iterative subgraph scoring, bag weight updating, and MGP based graph classification forms a closed loop to find optimal subgraphs and most suitable unlabeled bags for multi-graph learning. Experiments and comparisons on real-world multi-graph data demonstrate the algorithm performance.

1 Introduction

Multi-instance learning (MIL), originated from drug activity prediction [1], is a special learning task where labels are only available for a bag of instances. The niche of MIL stems from its capability to accommodate label ambiguity, with no label required for individual instances. Such properties have made MIL very suitable for many real-world applications, such as image retrieval and text categorization [2]. For example, in content based image retrieval, an image can be repressed as a bag and regions inside the images can be represented as instances. A bag is labeled as positive if any region inside the image contains objects interesting to users, *e.g.* a leopard as shown in Figure 1.

Although MIL has been used for many applications, existing multi-instance learning algorithms cannot handle complex data objects, and they that instances inside each bag are represented as feature vectors (*e.g.* instance-feature format). In reality, many real-world objects are inherently complicated, where each sample may contain instances with dependency structures (*i.e.* graphs). Such dependency allows relationships between objects to play important roles [4], but are, unfortunately, discarded in traditional instance-feature representations. For example, in Figure 1, region #2, from an actual image segmentation algorithm, contains mixed content including "tree","glass", "leopard". For existing MIL methods, they will regard the whole region #2 as one instance and use some visual features, such as

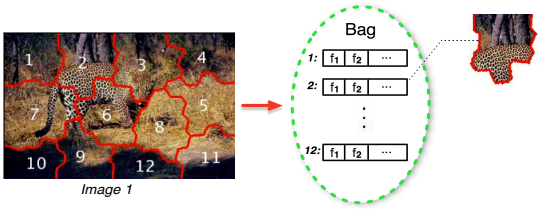

Figure 1: Multi-instance learning for images where each bag contains a number of regions (*i.e.* instances).

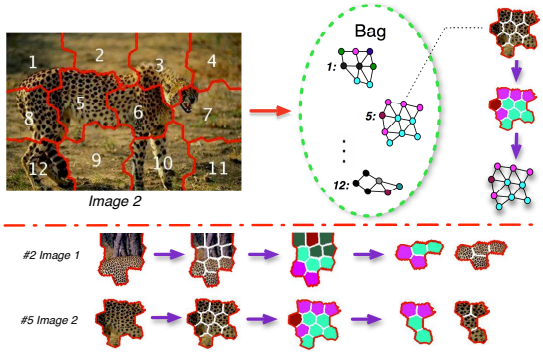

Figure 2: Multi-graph learning for images where each bag contains several regions (*i.e.* graphs). Each small area in a region is a superpixel [3]. Regions #2 (image 1) and #5 (image 2) share a common subgraph structure (two cyan nodes and two purple nodes) which corresponds to the animal body. By using a graph to denote each region, an image is represented as a bag of graphs. A graph is more powerful to represent the local structures inside the region than simply representing each region as an instance.

*Centre for Quantum Computation & Intelligent Systems, FEIT, University of Technology Sydney, Australia {jia.wu@student., zhibin.hong@student., shirui.pan@student., chengqi.zhang@ }uts.edu.au.

†Dept. of Computer & Electrical Engineering and Computer Science, Florida Atlantic University, USA, xzhu3@fau.edu.

‡Dept. of Computer Science, China University of Geosciences, Wuhan 430073, China, zhcai@cug.edu.cn.

color histogram, to represent the region. This will inevitably result in errors in the feature representation because features can hardly capture the complex content inside the region. Instead of treating each region as one single instance, we can use a graph to represent each region, as shown in Figure 2, where a number of small areas inside each region form a graph to represent the content and their structural relationships. This representation not only allows us use fine-grained small regions to represent image content, it also provides an effective way to use graph structures to represent rich semantic information inside the image regions. For example, Figure 2 shows that #2 in Image 1 and #5 in Image 2 share a common graph structure (a graph with two cyan nodes and two purple nodes) which represent the leopard body. By using a graph to represent each region, we can denote each image as a collection (or a bag) of graphs, where each graph captures the local structures inside a region which are inherently discarded in the existing multi-instance representation.

A second major limitation of existing MIL algorithms is they all require both positive and negative bags to be provided for learning. In reality, it would be more beneficial if only positive bags are needed to describe users' interests. For example, in image retrieval, during the search process users may click one or multiple images (*i.e.* bags) which are interesting to them (*i.e.* positive bags), but majority images are unchecked. For unchecked images, they may not contain users' retrieval concepts (*i.e.* negative bags) or users simply overlook the images. In this case, there is no negative bag but only positive and unlabeled bags are available.

The above observations raise a new problem setting where labels are only available for a bag of graphs and the learning is based on a limited number of positive and unlabeled bags. When learning from graph data, existing methods roughly fall into two categories [5, 6]: (1) global distance based methods, including graph kernels, graph embedding, and graph transformation etc., or (2) local subgraph feature based approaches, which represent graphs into a vector space for existing supervised learning methods to derive classification models. All these graph classification methods require each graph being explicitly labeled and cannot handle the proposed bag-of-graph setting where labels are only available for a bag of graphs. On the other hand, existing MIL algorithms cannot handle structured data, such as graphs, and are only applicable to bags with tabular instance-feature representations.

In summary, the main challenge of our new problem setting is threefold: (1) *Graph representation:* finding informative subgraph features to represent each bag is crucial for multi-graph learning, mainly because of the lack of features to represent graph data; (2) *Utilization of unlabeled bags:* without having negative bags, it is very hard, if not impossible, to derive multi-instance or multi-graph learning models. Some "reliable negative bags" need to be identified for learning; and (3) *Uncertainty inside positive bags:* because genuine label of each graph inside a positive bag is unknown, we need effective designs to handle such uncertainty and accurately identify positive graphs.

Indeed, a straightforward solution (graph-level) to solve the above problem is to propagate bag label to each graph inside the bag, so the problem becomes positive and unlabeled (PU) learning for graph classification [7]. This simple approach is ineffective because not all graphs in a positive bag are positive, so it will result in a significant amount of label errors. A slightly more intelligent (bag-level) design is to first mine a set of frequent subgraphs as features to represent all graphs as instances. Then the problem can be solved by using PU strategy [8] to first treat all unlabeled bags as negative bags and train an MI classifier, and iteratively refine identified negative bags by using trained classifiers. This solution is still ineffective because the selection of subgraph features fails to take unique bag constraint into consideration, some discriminative subgraph features will be missed even though they are not frequent subgraphs [9].

To solve the above challenges, we propose a puMGL framework with two alternately and iteratively combined steps, including: (1) deriving features to represent graph; and (2) deriving discriminative learning models with only positive and unlabeled graph bags. Experiments on two real-world learning tasks confirm the effectiveness of the proposed designs.

2 Preliminaries and Problem Statement

DEFINITION 1. **Connected Graph:** *A graph is represented as* $G = (\mathcal{V}, E, \mathcal{L}, l)$ *where* $\mathcal{V}$ *is a set of vertices* $\mathcal{V} = \{v_1, \cdots, v_{n_v}\}$, $E \subseteq \mathcal{V} \times \mathcal{V}$ *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. **Bag:** *Denote* $\boldsymbol{\mathcal{B}} = \{B_1, \cdots, B_n\}$ *a set with n bags, and* B_i *is the ith bag, which can be positive* B_i^+ *or unlabeled* B_i^u*. The collections of positive and unlabeled bag sets are denoted by* $\boldsymbol{\mathcal{B}}^+$ *and* $\boldsymbol{\mathcal{B}}^u$*, respectively. Meanwhile, we use* $\boldsymbol{\mathcal{B}}^-$ *to denote the set of unlabeled bags which are identified as negative bags.*

Let $\mathcal{Y} = [Y_1, \cdots, Y_n]$ where Y_i is the label of bag B_i. Generally, a positive and a negative bag's label can be denoted by $Y_i = +1$ and $Y_i = -1$,

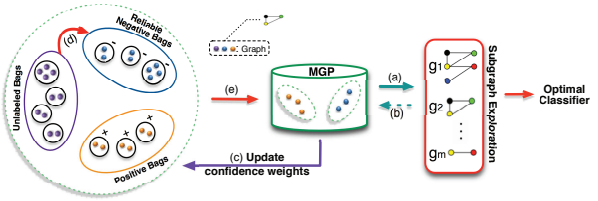

Figure 3: A conceptual view of puMGL framework: Given positive and unlabeled graphs bags, puMGL starts from assigning a confidence weight for each bag and builds the initial Margin Graph Pool (MGP). After that a subgraph feature set will be discovered by using confidence weight embedding (a) to represent graphs in MGP (b). By training a classifier from MGP, we can update the confidence weights for unlabeled bags (c) to help identify some unlabeled bags as reliable negative bags (d). These reliable negative bags help build a weighted distribution to measure graphs in positive and reliable negative bags, respectively. The "most positive pattern" for positive bags and the "least negative pattern" for reliable negative bags are selected to update MGP (e). The iterative process continues, with the objective of improving the quality of reliable negative bags and the quality of MGP, until the algorithm converges.

respectively. In a puMGL setting, an unlabeled bag B_i^u's label is denoted by $Y_i = 0$. In addition, we also use $G_{i,j}$ (G_j for abbreviation) with its label $y_{i,j}$ (y_j for abbreviation) to denote the jth graph in the bag $B_i = \{G_{i,1}, \cdots, G_{i,n_i}\}$. To tackle unreliable bag labels in puMGL setting (challenge # 2), we use a weight value w_i to indicate the confidence of each bag B_i's label. 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 $g = (\mathcal{V}', E', \mathcal{L}', l')$ being two graphs. g is a subgraph of G ($g \subseteq G$), iff there exists an injective function $\varphi : \mathcal{V}' \to \mathcal{V}$ s.t. (1) $\forall v \in \mathcal{V}', l'(v) = l(\varphi(v))$; (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 $\mathcal{S}_g = \{g_1, \cdots, g_s\}$ a subgraph set discovered from a given graph set. For each graph G_i, we use a subgraph feature vector $\boldsymbol{x}_i = [x_i^{g_1}, \cdots, x_i^{g_s}]^\top$ 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 $\boldsymbol{\mathcal{B}}$ containing 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.

3 Overall Framework of puMGL Learning

Figure 3 outlines the framework of the proposed puMGL algorithm with three major steps to solve the key challenges identified in Section 1: (1) **Margin Graph Pool (MGP):** To carry out multi-graph learning with positive and unlabeled bags only, we propose to use maximum margin idea [10] to build a positive margin graph pool (MGP), which consists of "most positive pattern" from the positive bags and "least negative pattern" from identified negative bags to help differentiate positive bags; (2) **Subgraph Exploration:** For exploring informative subgraphs from MGP, we employ a confidence weight embedding strategy (detailed in Section 4.1) as follows: the confidence weight information is embedded into the objective function to select an informative subgraph set to represent graphs in MGP; and (3) **Update Confidence Weight:** To properly identify most negative bags from unlabeled bags, we assign a confidence weight for each bag and will update bag weight to find reliable negative bags. The weight updating ensures high quality negative bags are identified to form MGP for further learning.

4 Positive and unlabeled MG Learning

Based on the framework in Figure 3, the main research problem is twofold: (a) How to identify reliable negative bags to further build the MGP? and (b) How to evaluate the usefulness of subgraphs based on graphs in MGP?

The above two problems are closely related to each other. To build the MGP, the reliable negative bags need to be identified first. However, prior to that, a set of subgraph features should have been identified to represent the graphs. On the other hand, for problem (b), the evaluation of the usefulness of subgraphs is based on their performance in differentiating positive *vs.* negative graphs. Without a high quality MGP, there is no way to properly assess the usefulness of subgraphs. Therefore, we propose the following optimization framework to concurrently identify reliable negative bags and explore subgraph features.

4.1 Alternating Optimization Framework

Assume that each bag $\mathcal{B}_i$ in training set $\boldsymbol{\mathcal{B}}$ is assigned with a confidence weight w_i. The collections of graphs in MGP are denoted by $\mathcal{G} = \{\hat{G}_1, \cdots, \hat{G}_p\}$ with its size being p. Each $\hat{G}_j$ in $\mathcal{G}$ has a weight $\hat{w}_j$. In our design, because only one graph is selected from each bag, the weight confidence of the graph is equal to the weight confidence value of the bag to which the bag belongs to. Let $\mathcal{S}_g$ denote the complete set of subgraphs discovered from $\mathcal{G}$. Our optimization **aims** to find a set of most informative features $\mathbf{g}$, ($\mathbf{g} \subseteq \mathcal{S}_g$), based on the confidence

weight vector $\hat{\mathbf{w}}$. To this end, we define $\mathcal{J}(\mathbf{g}, \hat{\mathbf{w}})$ as an evaluation function to estimate dependency between the selected feature set $\mathbf{g}$ and the confidence label weights of the bags as defined in Eq. (4.1), where $|\cdot|$ represents the cardinality of the set, and m is the number of subgraphs to be selected from $\mathcal{S}_g$.

$$(\mathbf{g}^\star, \hat{\mathbf{w}}^\star) = \underset{\mathbf{g}\subseteq\mathcal{S}_g, \hat{w}_i\in(0,1]}{\arg\max} \mathcal{J}(\mathbf{g}, \hat{\mathbf{w}}) \;\; s.t. \;\; |\mathbf{g}| \leq m \tag{4.1}$$

Confidence Weight Embedding: To maximize the evaluation $\mathcal{J}(\mathbf{g}, \hat{\mathbf{w}})$, we impose confidence weight embedded constraints to graphs $\mathcal{G}$ in MGP as follows: (a) *weight embedded must-link:* for any two graphs with the same labels, they should be close to each other. Meanwhile, because each graph is associated with a confidence weight $\hat{w}_i$, the selected subgraph should ensure that graphs with similar weights have a high similarity; (b) *weight embedded cannot-link:* graphs in different classes should be far away from each other. The smaller the confidence weight difference between two graphs, the more distinct the graphs are with respect to the selected subgraphs. In summary, the embedding process is to utilize confidence weight information to help find informative subgraphs as features to represent graphs. Similar assumptions have also been used in the previous work which handles graphs with labeled and unlabeled graphs [11].

Based on the above constraints, we derive an evaluation criterion for $\mathcal{J}(\mathbf{g}, \hat{\mathbf{w}})$ as follows:

$$\mathcal{J}(\mathbf{g}, \hat{\mathbf{w}}) = \frac{1}{2}\sum\nolimits_{i,j} K_{\mathbf{g}}\Big(\hat{G}_i, \hat{G}_j\Big) M_{i,j} \tag{4.2}$$

Where $M_{i,j}$ embeds confidence weight information (*i.e.* $\hat{\mathbf{w}}$) between two graphs $\hat{G}_i$ and $\hat{G}_j$ in MGP. $K_{\mathbf{g}}(\hat{G}_i, \hat{G}_j)$, which is defined in Eq. (4.3), denotes distance between two graphs $\hat{G}_i$ and $\hat{G}_j$ in subgraph feature space based on the selected subgraph set $\mathbf{g}$.

Accordingly, the problem defined in Eq. (4.2) can be regarded as a nonlinearly constrained nonconvex optimization problem. To the best of our knowledge, there is no direct solution to find its global optimum. In this paper, we propose an iterative algorithm by using the alternating optimization to obtain a local optimal solution. The optimization iteratively updates $\mathbf{g}$ and $\hat{\mathbf{w}}$ in an alternating fashion. In the following part, we will first show how to optimize the above two subproblems, and then integrate them to form our puMGL framework.

4.2 Optimal Subgraph Features In order to derive solutions to find subgraphs as features, we first formally introduce notations as follows:

- $\hat{\mathcal{X}}$: the matrix using subgraphs S_g to represent all graphs $\mathcal{G}$ in MGP, $\hat{\mathcal{X}} = [\hat{\mathbf{x}}_1, \cdots, \hat{\mathbf{x}}_p] = [\boldsymbol{f}_{g_1}, \cdots, \boldsymbol{f}_{g_s}]^\top \in \{0,1\}^{s\times p}$, where $\boldsymbol{f}_{g_k}$, $g_k \in S_g$, is an indicator vector of subgraph g_k with respect to all graphs in $\mathcal{G}$, *i.e.*, $\boldsymbol{f}_{g_k} = [f_{g_k}^{\hat{G}_1}, \cdots, f_{g_k}^{\hat{G}_p}]^\top$, where $f_{g_k}^{\hat{G}_i} = 1$ iff $g_k \subseteq \hat{G}_i$ and $f_{g_k}^{\hat{G}_i} = 0$ otherwise.
- A and B: $A = \{(i,j)|y_i y_j = 1\}$ denotes the *weight embedded must-link* pairwise constraint sets among $\mathcal{G}$, while $B = \{(i,j)|y_i y_j = -1\}$ denotes the *weight embedded cannot-link* pairwise constraint sets.

To calculate $M_{i,j}$ in Eq. (4.2), we adopt a radial basis kernel function to measure $M_{i,j} = K(\hat{w}_i, \hat{w}_j)$. While $K_{\mathbf{g}}(\hat{G}_i, \hat{G}_j)$ in Eq. (4.2) is defined as

$$K_{\mathbf{g}} = \begin{cases} -\left\|\mathcal{D}_{\mathbf{g}}\hat{\mathbf{x}}_i - \mathcal{D}_{\mathbf{g}}\hat{\mathbf{x}}_j\right\|^2/|A|, y_i y_j = 1 \\ \left\|\mathcal{D}_{\mathbf{g}}\hat{\mathbf{x}}_i - \mathcal{D}_{\mathbf{g}}\hat{\mathbf{x}}_j\right\|^2/|B|, y_i y_j = -1 \end{cases} \tag{4.3}$$

where $\mathcal{D}_{\mathbf{g}} = diag(d(\mathbf{g}))$ is a diagonal matrix indicating which subgraph features $\mathbf{g}$ are selected from S_g to represent the graphs, $d(\mathbf{g})_i = I(g_i \in \mathbf{g})$ with $I(\cdot)$ being an indicator function. By using a confidence weight embedded matrix $C = [C_{ij}]^{p\times p}$, $C_{ij} = \{-M_{i,j}/|A|, y_i y_j = 1;\ M_{i,j}/|B|, y_i y_j = -1; 0, otherwise\}$, through which the confidence weight information is embedded in the matrix C, Eq. (4.2) can be rewritten as follows,

$$\begin{aligned} \mathcal{J}(\mathbf{g}, \hat{\mathbf{w}}) &= \frac{1}{2}\sum_{i,j}\left\|\mathcal{D}_{\mathbf{g}}\hat{\mathbf{x}}_i - \mathcal{D}_{\mathbf{g}}\hat{\mathbf{x}}_j\right\|^2 C_{i,j} \\ &= tr(\mathcal{D}_{\mathbf{g}}^\top \hat{\mathcal{X}}(D-C)\hat{\mathcal{X}}^\top \mathcal{D}_{\mathbf{g}}) \\ &= tr(\mathcal{D}_{\mathbf{g}}^\top \hat{\mathcal{X}} L \hat{\mathcal{X}}^\top \mathcal{D}_{\mathbf{g}}) \\ &= \sum\nolimits_{g_k\in\mathbf{g}} \boldsymbol{f}_{g_k}^\top L \boldsymbol{f}_{g_k} \end{aligned} \tag{4.4}$$

where $tr(\cdot)$ is the trace operator for a matrix, D is a diagonal matrix generated from C, *i.e.,* $D_{i,i} = \sum_j C_{ij}$. And $L = [L_{i,j}]^{p\times p} = D - C$ is a Laplacian matrix. By denoting the function as $z(g_k, L) = \boldsymbol{f}_{g_k}^\top L \boldsymbol{f}_{g_k}$, the problem of maximizing $\mathcal{J}(\mathbf{g}, \hat{\mathbf{w}})$ in Eq. (4.1) is equal to finding a subset of subgraphs that can maximize the sum of $z(g_k, L)$, which can be represented as:

$$\max_{\mathbf{g}} \sum\nolimits_{g_k\in\mathbf{g}} z(g_k, L) \qquad s.t. \;\; \mathbf{g}\subseteq S_g, |\mathbf{g}| \leq m. \tag{4.5}$$

DEFINITION 5. **puScore Criterion:** *Given $\boldsymbol{\mathcal{B}}$ with positive and unlabeled graph bags, and a confidence weight embedded matrix C, with L denoting an Laplacian matrix as $L = D - C$. The informativeness score of a subgraph g_k can be measured by:*

$$\mathbb{r}(g_k) = z(g_k, L) = \boldsymbol{f}_{g_k}^\top L \boldsymbol{f}_{g_k} \tag{4.6}$$

Since the Laplacian matrix L is positive semi-definite [11], for any subgraph g_k, $\boldsymbol{f}_{g_k}^\top L \boldsymbol{f}_{g_k} \geq 0$, *i.e.,* $\mathbb{r}(g_k) \geq 0$. In order to find the optimal subgraph set $\mathbf{g}$ which maximizes the criterion defined in Eq.

Algorithm 1 ESE: Embedding Subgraph Exploration

Require:
$\mathcal{G}$: 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;
Ensure:
$\mathbf{g} = \{g_1, \cdots, g_m\}$: 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: $C \leftarrow$ Apply $\hat{\mathbf{w}}$ to $\mathcal{G}$ and obtain the embedding matrix;
7: $\mathbb{r}(g_k) \leftarrow$ Apply C to compute puScore of subgraph g_k;
8: **if** $|\mathbf{g}| < m$ or $\mathbb{r}(g_k) > \tau$, **then**
9: $\mathbf{g} \leftarrow \mathbf{g} \bigcup g_k$;
10: **if** $|\mathbf{g}| \geq m$, **then**
11: $\mathbf{g} \leftarrow \mathbf{g}/\arg\min_{g_i \in \mathbf{g}} \mathbb{r}(g_i)$;
12: $\tau = \min_{g_i \in \mathbf{g}} \mathbb{r}(g_i)$;
13: Depth-first search the subtree rooted from node g_k;
14: **end while**
15: **return** $\mathbf{g}$;

(4.5), we can calculate puScore of each subgraph in $\mathcal{S}_g$ and sort them in a descending order, *i.e.*, $\mathbb{r}(g_1) \geq \mathbb{r}(g_2) \cdots \geq \mathbb{r}(g_s)$ and then collect top-m subgraphs $\mathbf{g} = \{g_1, \cdots, g_m\}$.

Subgraph Exploration: One straightforward solution for finding an optimal subgraph set is exhaustive enumeration, *i.e.*, enumerate all subgraphs in a graph set to find the ones with maximum puScore values. Unfortunately, the number of subgraphs grows exponentially with respect to the size of graphs in bags, which makes the exhaustive enumeration approach impractical for real-world data. To solve the problem, we employ a Depth-First-Search (DFS) based algorithm gSpan [12] to iteratively enumerate subgraphs. The key idea of gSpan is to first assign a unique minimum DFS code to each graph, and then discover all frequent subgraphs by a pre-order traversal of the tree.

Algorithm 1 lists the proposed subgraph feature exploration approach, which starts with an empty feature set $\mathbf{g}$ and the minimum puScore $\tau = 0$. The algorithm continuously enumerates subgraphs by recursively visiting the DFS code tree in gSpan. If a subgraph g_k is not frequent, both g_k and its subtree will be pruned (lines 4-5). Otherwise, we calculate g_k's puScore value $\mathbb{r}(g_k)$ using the confidence weight embedded matrix C (line 6). If $\mathbb{r}(g_k)$ is larger than τ which is the minimum puScore of the current set $\mathbf{g}$, or $\mathbf{g}$ has less than m subgraphs (*i.e.* $\mathbf{g}$ is not full), g_k is added to the subgraph set $\mathbf{g}$ (lines 8-9). If the size of $\mathbf{g}$ exceeds the predefined value m, we need to remove one subgraph with the least discriminative power (lines 10-11). After that, the minimum puScore of $g_i \in \mathbf{g}$ (*i.e.* $\min_{g_i \in \mathbf{g}} \mathbb{r}(g_i)$) will be used to update the threshold τ (line 12). Finally, the depth-first search will continue by following the children of g_k (line 13), until the subgraph mining process is completed.

4.3 Confidence Weight Optimization Following the above process, we can obtain an optimal subgraph set $\mathbf{g}$. After that, we will solve the second subproblem: optimize the confidence weight $\hat{\mathbf{w}}$ for MGP. The optimization can be approximated by building a classifier from MGP using obtained subgraph features, and utilizing the classifier to predict class labels of unlabeled bags. In the following, we first explain the construction of MGP, and then introduce detailed process for updating optimal bag confidence weight values.

Margin Graph Pool (MGP): The main purpose of building MGP is to identify some "most positive patterns" from positive bags and "least negative patterns" from reliable negative bags, so MGP can help build classifiers to differentiate positive *vs.* negative bags for learning. This also provides solutions to tackle unreliable bag labels obtained from unlabeled bags.

The construction of MGP is motivated by the margin principle, which states that samples close to the decision boundary play critical roles in improving the performance of the underlying classifier. In proposed puMGL, we assign a confidence weight w_i to each bag, so it can help identify unlabeled bags which are most likely being negative as "reliable negative bags" and then extract most positive patterns from positive bags and least negative patterns from identified negative bags to form MGP. According to the multi-graph setting, a negative bag does not contain positive graphs, and graphs in negative bags can have very general distributions. So we use a weighted Kernel Density Estimator [13] (WKDE) to model distributions of negative graphs in reliable negative bags as follows.

$$\rho(\mathbf{x}|\mathcal{X}^-) = \frac{1}{\sum_i n_i^-} \sum\nolimits_{i,j} K(w_x\mathbf{x}, w_i\mathbf{x}_{i,j}^-) \tag{4.7}$$

where w_x denotes weight of the bag to which the graph x belongs to, and $\mathbf{x}_{i,j}^-$ denotes the subgraph feature representation of the jth graph in the ith reliable negative bag. n_i^- denotes the size of ith reliable negative bag, and an isotropic Gaussian kernel K is used to measure to the similarity between two graphs under the feature representation by using $\mathbf{g}$. The collection of the subgraph feature vectors for all graphs in reliable negative bags $\mathcal{B}^-$ is denoted by $\mathcal{X}^-$. Depending on whether the underlying bag is positive or is identified as negative, the "most positive pattern" or "least negative

Algorithm 2 CWO: Confidence Weight Optimization

Require:
$\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{B}}^+ \cup \boldsymbol{\mathcal{B}}^{\boldsymbol{u}}$: An graph bag data set;
$\mathbf{g}$: subgraph features; $\mathcal{G}$: A graph set inMGP;
Ensure:
$\hat{\mathbf{w}}$: A set of confidence weight for MGP;
// **Unlabeled Bag Weight Optimization:**
1: $\hat{\mathcal{X}} \leftarrow$ Apply $\mathbf{g}$ to $\mathcal{G}$ to obtain subgraph feature vectors.
2: $\mathcal{H} \leftarrow$ Apply $\hat{\mathcal{X}}$ to build the classifier.
3: **for** each bag B_i in $\boldsymbol{\mathcal{B}}^{\boldsymbol{u}}$ **do**
4: **for** each graph $G_{i,j}$ in B_i **do**
5: $\mathbf{x}_{i,j} \leftarrow$ Apply $\mathbf{g}$ to $G_{i,j}$ to obtain its feature vector.
6: $p_{i,j} \leftarrow$ Apply $\mathcal{H}$ to $\mathbf{x}_{i,j}$ and estimate probability;
7: **end for**
8: $w_i \leftarrow \sum_j^{n_i} p_{i,j}/n_i$;
9: **end for**
// **MGP Weight Optimization:**
10: $\boldsymbol{\mathcal{B}}^- \leftarrow$ Apply $\mathbf{w}$ to $\boldsymbol{\mathcal{B}}^{\boldsymbol{u}}$ to form reliable negative bag set.
11: $\mathcal{D} \leftarrow$ Generate a distribution from $\boldsymbol{\mathcal{B}}^-$ via Eq. (4.7).
12: $\hat{\mathbf{w}} \leftarrow$ Apply $\mathcal{D}$ to $\boldsymbol{\mathcal{B}}^+$, $\boldsymbol{\mathcal{B}}^-$ and update $\mathcal{G}$ via Eq. (4.8).
13: **return** $\hat{\mathbf{w}}$;

pattern" $\mathbf{x}_i^\rho$ in MGP can be obtained by Eq.(4.8).

$$\mathbf{x}_i^\rho = \arg \min_{\mathbf{x}_{i,j} \in X_i, j=1,\cdots,n_i} \rho(\mathbf{x}_{i,j}|\mathcal{X}^-) \tag{4.8}$$

where X_i is the subgraph feature vectors for the ith bag in $\mathcal{B}$ under the given subgraph set $\mathbf{g}$.

Confidence Weight Updating: Presumably the quality of the initial subgraph features is unsatisfactory, but can be improved iteratively. After we build the classifier based on MGP, we reevaluate each unlabeled bag including those previously identified as negative bags, because some of them may be false negatives due to the ineffectiveness of the subgraphs used previously. If an improved set of subgraphs with a better quality are discovered, we will use these features to help identify unlabeled bags which are likely being negative. Because positive graph bags are initially given in the data set with confidence 1.0, we only update the confidence weight of unlabeled bag to indicate their likelihood of being an negative bag. Any classifier which can output probability estimation is sufficient for this purpose.

Assume $p_{i,j}$ denotes the probability of a graph $G_{i,j}$ in bag $\mathcal{B}_i^u$ being classified as a negative graph, we can calculate the confidence weight for each bag using $w_i = \sum_j^{n_i} p_{i,j}/n_i$, which also helps update confidence weights $\mathbf{w}$ for all unlabeled bags. At last, the bags in unlabeled set with high confidence weights will be selected as reliable negative bag set (which has the same number of bags as $\boldsymbol{\mathcal{B}}^+$) and are further used to build the MGP. In this case, the collection of each weight $\hat{w}_i$ for each graph in MGP will be used as the optimal $\hat{\mathbf{w}}$. Algorithm 2 outlines the detailed confidence weight optimization process which includes two iterative parts: (1) unlabeled bag weight optimization, and (2) MGP weight optimization.

Algorithm 3 puMGL: PU Multi-Graph Learning

Require:
$\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{B}}^+ \cup \boldsymbol{\mathcal{B}}^u$: An graph bag data set;
min_sup: The threshold of the frequent subgraph;
m: the number of subgraph features to be selected;
Ensure:
The target class label y_t of a test bag B_t.
// **Training Phase:**
1: Set the labels of the unlabeled bags $\mathcal{B}_j^u$ to be -1;
2: $\mathcal{G} \leftarrow$ Initialize MGP $\mathcal{G}$ by randomly selecting one graph ($\hat{w}_i = 1$) from each $B_i \in \boldsymbol{\mathcal{B}}^+$ and similarly for each $B_j \in \boldsymbol{\mathcal{B}}^u$ under the same amount ($\hat{w}_j \in (0, 1]$, randomly);
3: **while not** convergence for $\hat{\mathbf{w}}$ **do**
// **Optimal Subgraph Features:**
4: $\mathbf{g} \leftarrow ESE(m, min\_sup, \mathcal{G}, \hat{\mathbf{w}})$; //Algorithm 1
5: $\mathcal{G} \leftarrow$ Apply $\mathbf{g}$ to represent the graphs in MGP $\mathcal{G}$.
// **Confidence Weight Optimization:**
6: $\hat{\mathbf{w}} \leftarrow CWO(\boldsymbol{\mathcal{B}}, \mathbf{g}, \mathcal{G})$; //Algorithm 2
7: **end while**
8: $\mathbf{g}^* \leftarrow \mathbf{g}$; $\hat{\mathbf{w}}^* \leftarrow \hat{\mathbf{w}}$; // Optimal subgraphs and weights.
// **Test Phase:**
9: $\mathcal{H}^* \leftarrow$ Apply $\mathbf{g}^*$ and $\hat{\mathbf{w}}^*$ to $\mathcal{G}$ to build the classifier.
10: $\mathbf{x}_{t,i} \leftarrow$ Apply $\mathbf{g}^*$ to each $G_{t,i}$ in B_t to obtain its vector.
11: $Y_t \leftarrow$ Apply $\mathcal{H}^*$ to each $\mathbf{x}_{t,i}$ to predict its bag label;
12: **return** Y_t;

4.4 puMGL Algorithm Algorithm 3 lists detailed procedures of the proposed puMGL framework. At the first step, puMGL sets all unlabeled bags to have label -1, with their weight w_j being a random value within $(0, 1]$. The algorithm initializes the graphs $\mathcal{G}$ in MGP by randomly selecting one graph from each positive bag B_i^+ and uses the same approach to select the same number of graphs from all $B_j^- \in \boldsymbol{\mathcal{B}}^u$ (lines 1-2).

During the first *while* loop, puMGL selects initial subgraph features using Algorithm 1 (line 4), because the initial graphs in MGP have no feature representation. At this stage, the quality of initial subgraphs is not optimal because the initial MGP consists of randomly selected graphs. With the help of subgraph features, we can re-represent the MGP to build classifiers in feature space, which will be used to evaluate all graphs in unlabeled bags. After that, puMGL updates the confidence weight $\mathbf{w}$ for MGP by using Algorithm 2 (line 8). By using "optimize subgraph feature" and "optimize confidence weight" in an iterative way, we can obtain optimal subgraphs $\mathbf{g}^*$ and confidence weights $\hat{\mathbf{w}}^*$ at the same time, until the confidence weight $\hat{\mathbf{w}}$ for MGP becomes stable. At the test phase, all graphs $G_{t,i}$ in test bag B_t are transferred into a feature vector by using $\mathbf{g}^*$,

and are predicted by the classifier $\mathcal{H}^*$ to obtain their labels y_t by using multi-graph constraints: a test bag is classified as positive if any graph inside is classified as positive, and negative otherwise (lines 9-11).

5 Experiments

5.1 Experimental Settings To evaluate the effectiveness of puMGL, we use F-score, which combines precision P and recall R: $2 \times P \times R/(P+R)$, to measure the performance of PU learning [7, 8]. In addition, LibSVM, which has been popularly used for probability estimation [14], is employed as the classifier in puMGL. For benchmark data sets used in the experiments, 30% of bags are randomly selected as testing set in each run, with the remaining bags being used as training set. Unless specified otherwise, the default value of r is 0.4, the number of selected subgraph features is 60, and the minimum support threshold $min\_sup = 8\%$ for Online Product Review data (Sect. 5.3) and $min\_sup = 3\%$ for content-based image retrieval (Sect. 5.4). Moreover, all results are based on the average performance over 10 repetitions and all experiments are conducted on a Linux cluster node with an Intel(R) Xeon(R) @3.33GHZ CPU and 3GB fixed memory.

5.2 Baseline Methods Because there is no existing method to solve the proposed problem, for comparison purposes we implement the following baseline approaches from bag- and graph-level perspectives. The former directly employ PU learning [8] at the bag levels, and the latter propagates bag label to graphs and transfer puMGL as a generic PU graph learning problem (puGL) which can be solved by an existing method [7].

Bag-level approaches: A set of (Top-m) frequent subgraphs is firstly mined to represent graphs. So the problem becomes PU multi-instance learning (puMIL), which still has no known solution but can be solved in a naive way. As a baseline, puMIL directly builds MI classifiers by treating unlabeled bags as negative bags. Moreover, a "spy" mechanism, which has shown good performance on text documents [8], is used in puMIL (Adding spies to the unlabeled set allows algorithm understand the characteristics of unknown positive bags in the unlabeled set). In summary, puMIL first randomly samples a set of positive bags as "spies", and marks them as unlabeled bags, then uses the MILR [15] to obtain a set of reliable negative bags from the unlabeled bag set. After that, it runs MISVM [10] iteratively on the positive set and reliable negative set until converges.

Graph-level approaches: This method directly propagates bag labels to graphs inside each positive bag with graphs in unlabeled bags remaining unlabeled. As a result, the problem is converted to a puGL task [7]. More specifically, puGL first derives an evaluation criterion based on Hilbert-Schmidt Independence Criterion [16] to estimate the dependency between subgraph features and class labels based on a set of estimated negative graphs. Then puGL devises an integrated framework to evaluate the usefulness of subgraph features based on positive and unlabeled graphs. A test bag is predicted as positive if one or more graphs in the bag is predicted as positive, and negative otherwise.

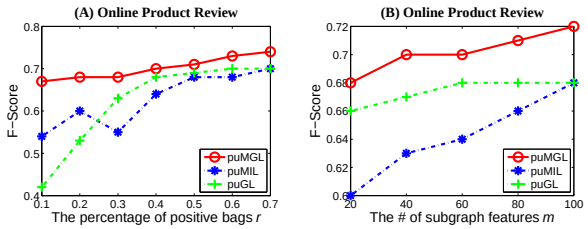

Figure 4: F-score comparisons on *Online Product Review* $w.r.t$ different r values (A) and # of subgraphs m (B).

5.3 Online Product Recommendation This data set is downloaded from Stanford Network Dataset Collection (http://snap.stanford.edu/data/). The beer review dataset *Beers* contains numerous beer related user reviews. Each review is associated with some attributes such as product ID, reviewer ID, review score (rating of the product varying from 1 to 5), and detailed text descriptions [17]. If the average score over all the reviews for the product is greater or equal to 4, we believe that one or more key properties (*e.g.* "affordability" and "durability") of the product may receive very positive reviews. On the other hand, the customer may be not interested in this product, if all of the review scores are less than 4. Our goal is to recommend good products to users based on the review text. For each review report, we use fuzzy cognitive map (E-FCM) [18] to form a graph representation with each node in the graph denoting one keyword and edges representing correlations between keywords. Similar graph representation approach can be found in our previous work [19].

In our experiments, all edges whose correlation values less than a certain threshold (0.006) are discarded. We choose 600 beer products, each of which containing 1 to 10 reviews, to form 300 interesting positive bags with 1756 graphs (average score ≥ 4) and 300 products (bags) with 1528 graphs (score< 4) as unlabeled bags. To validate the performance of puMGL with respect to different sizes of positive bags, we randomly select $r \times 100\%$ interesting products (varying from 0.1 to 0.7) as positive bags, and combine remaining products and

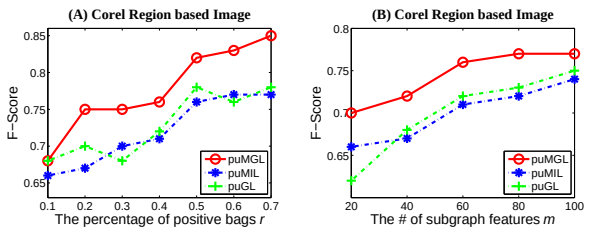

Figure 5: F-score comparisons on *Corel Image* *w.r.t.* different r values (A) and different # of subgraphs m (B).

all other products/bags as unlabeled bag set.

Figure 4(A) reports the results with respect to different r values, which show that puGL is inferior to puMIL when r is smaller than 0.3. However, for large r values, puGL achieves better performance than puMIL, this is possibly because a large number of positive bags can provide sufficient information to help discover subgraphs with good discriminative power, which may also include those infrequent subgraph features compared with puMIL merely using static subgraph frequency. Meanwhile, puMGL clearly outperforms all baselines, especially when only a small portion of positive bags are labeled (*i.e.* $r \leq 0.2$). This suggests that puMGL is effective over a wide range of percentage of labeled positive bags. Besides, we also test the effectiveness of puMGL by varying the number of selected subgraphs from 20 to 100. The results in Figure 4(B) show that puMGL consistently outperforms baselines.

5.4 Region-based Image Retrieval In this section, we report puMGL's performance for content based image retrieval. The original images from Corel data set [20] are preprocessed and segmented using VLFeat System (http://www.vlfeat.org/). In this case, each image is considered as a bag with each region (or a segment) inside the image denoting a graph. For each region, we use a state-of-the-art superpixel based approach, Simple Linear Iterative Clustering (SLIC) [3], to form a graph representation with each node denoting one superpixel and edges representing adjacency between superpixels. Specifically, each region outputs a set of superpixels, with each of which being labeled with the RGB-color histogram within each corresponding superpixel. We use 16 bins per channel, which yields 4096-dimensional histograms. In order to reduce the quantity of nodes in each region, a clustering method is employed to label superpixels. By doing so, each superpixel corresponds a cluster, with adjacent superpixels being linked through an edge. A multi-graph representation example for image from the database is shown in Figure 2.

To build positive bags, we use category "Cats", which consists of "Tiger", "Lion" and "Leopard", as positive bags (300 bags with 2679 graphs) and randomly draw 300 images of other animals to form unlabeled bags with 2668 graphs (*i.e.* regions). To validate the performance of puMGL *w.r.t.* different sizes of positive bags, we randomly select $r \times 100\%$ "Cats" images ($r \in [0.1, 0.7]$) as positive bags, and combine remaining "Cats" images and all other images as unlabeled bags.

The results in Figures 5(A) and 5(B) show the performance of puMGL compared to the baselines *w.r.t.* different r and m values. From Figure 5(A), puMGL archives a much better performance than puGL and puMIL. We further notice that the F-score will increase when the r increases. This is quite natural because a larger number of positive examples provide more useful information to help represent graph bags. Similarly, the classification quality usually improves when the number of selected subgraphs features m increases, as shown in Figure 5(B). This is due to the fact that a larger number of selected subgraph features can provide more structure information to support learning. Although puGL and puMIL achieve comparable performance with the increasing m values, they cannot reach the best performance achievable by the proposed puMGL.

5.5 Effectiveness of MGP in puMGL To further illustrate the effectiveness of puMGL for building MGP and validate whether MGP indeed contains highly quality samples, we report some "most positive patterns" in MGP, which include the common pattern (*i.e.* subgraph with high puScore) as shown in the first row of Figure 6. The samples (*i.e.* regions) selected from the four images all correspond to a part of the leopard, which share a common pattern (*i.e.* common local structures). This suggests that MGP is able to find the key region of the object for training multi-graph classifiers. Meanwhile, our experiments also notice that not all regions in MGP are highly representative (*i.e.* the selected regions might be less "positive" compared to other regions). This is due to the fact that the surrounding environment sometimes can affect the results, as demonstrated in the second row of Figure 6.

6 Conclusions

In this paper, we investigated a new multi-graph learning task with only positive and unlabeled bags, where a bag contains a number of graphs, and labels are only available for positive bags. This problem setting is significantly more challenging than traditional multi-instance learning because no feature representation is immediately available to represent graphs in each bags, and furthermore, no negative bags are available for deriving discriminative models. To address the challenge,

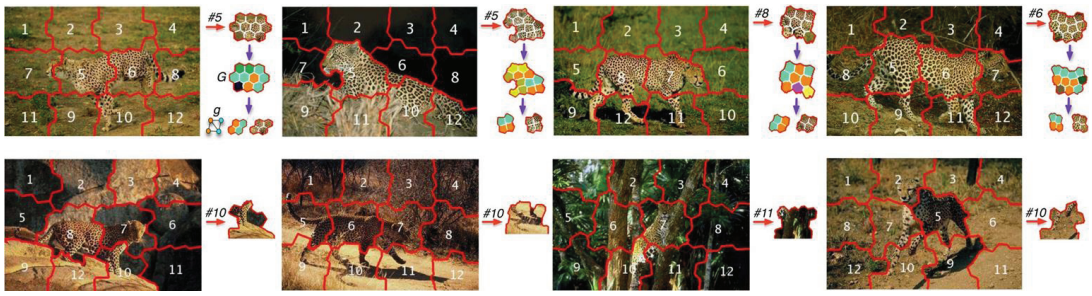

Figure 6: Examples of "most positive patterns" obtained by MGP. Taking the image on left corner for example, the region #5 is selected as "most positive pattern" according to MGP strategy in Section 4.3. The selected regions in MGP further help find the optimal common pattern g (*i.e.* subgraph with high puScore), under the graph representation G for the first row. The second row shows some examples, where selected regions are relatively less "positive" compared to other regions.

we proposed an optimization framework to iteratively explore subgraph features and then evaluate unlabeled graphs to build a margin graph pool to help train multi-graph classifiers. The iterative subgraph exploration and unlabeled graph bag updating will ensure that the whole process can find high quality subgraph features and most probable negative bags for graph bag classification. Experiments and comparisons on real-world tasks show that the proposed puMGL approach significantly outperforms baselines.

Acknowledgment

The work was supported by the Key Project of the Natural Science Foundation of Hubei Province, China (Grant No. 2013CFA004), and the National Scholarship for Building High Level Universities, China Scholarship Council (No. 201206410056).

References

[1] T. Dietterich., R. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artif. Intell.*, vol. 89, 1997.

[2] Z. Zhou, M. Zhang, S. Huang, and Y. Li, "Multi-instance multi-label learning," *Artif. Intell.*, vol. 176, pp. 2291–2320, 2012.

[3] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Ssstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 2274–2282, 2012.

[4] J. Tang, T. Wang, Q. Lu, J. Wang, and W. Li, "A wikipedia based semantic graph model for topic tracking in blogosphere," in *IJCAI*, 2011.

[5] S. Pan, X. Zhu, C. Zhang, and P. S. Yu, "Graph stream classification using labelled and unlabeled graphs," in *ICDE*, 2013, pp. 398–409.

[6] D. Surian, Y. Tian, D. Lo, H. Cheng, and E.-P. Lim, "Predicting project outcome leveraging socio-technical network patterns," in *CSMR*, 2013, pp. 47–56.

[7] Y. Zhao, X. Kong, and P. S. Yu, "Positive and unlabeled learning for graph classification," in *ICDM*, 2011, pp. 962–971.

[8] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, "Building text classifiers using positive and unlabeled examples," in *ICDM*, 2003, pp. 179–186.

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

[10] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning," in *NIPS*, 2003, pp. 561–568.

[11] X. Kong and P. Yu, "Semi-supervised feature selection for graph classification," in *KDD*, 2010, pp. 793–802.

[12] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *ICDM*, 2002, pp. 721–724.

[13] Z. Fu, A. Robles-Kelly, and J. Zhou, "Milis: Multiple instance learning with instance selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, pp. 958–977, 2011.

[14] T.-F. Wu, C.-J. Lin, and R. C. Weng, "Probability estimates for multi-class classification by pairwise coupling," *J. Mach. Learn. Res.*, vol. 5, 2004.

[15] S. Ray and M. Craven, "Supervised versus multiple instance learning: an empirical comparison," in *ICML*, 2005, pp. 697–704.

[16] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with hilbert-schmidt norms," in *ALT*, 2005, pp. 63–77.

[17] J. J. McAuley and J. Leskovec, "From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews," in *WWW*, 2013, pp. 897–908.

[18] X. Luo, Z.X., J. Yu, and X. Chen, "Building association link network for semantic link on web resources," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, 2011.

[19] J. Wu, X. Zhu, C. Zhang, and Z. Cai, "Multi-instnace multi-graph dual embedding learning," in *ICDM*, 2013, pp. 827–836.

[20] J. Li and J. Z. Wang, "Real-time computerized annotation of pictures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, pp. 985–1002, 2008.