Entropy **2017**, *1*, 1-1; doi:10.3390/——

OPEN ACCESS
entropy
ISSN 1099-4300
www.mdpi.com/journal/entropy

Article

Multi-Graph Multi-Label Learning Based on Entropy

Zixuan Zhu [1,*] and Yuhai Zhao [1,]

[1] College of Computer Science and Engineering, Northeastern University, Wenhua Rd., Heping Dist. Shenyang, Liaoning, China

* Author to whom correspondence should be addressed; Zixuan.Zhu.CN@gmail.com

Abstract: We propose an innovate algorithm of *Multi-Graph Multi-Label Learning* to classify images. Firstly, it uses the structure *Graph* instead of *Instance* to represent an image so that the accuracy of *Machine Learning* will be improved. Furthermore, it uses multiple labels as the output to eliminate the semantic ambiguity of the image. Lastly, it calculates the entropy to mine the informative subgraphs instead of just mining the frequent subgraphs, which enables to select the more precise features. We degenerate the *Multi-Graph Multi-Label Learning* into the *Multi-Instance Multi-Label Learning* and solve it by *MIML-ELM*. Our performance study shows that our algorithm outperforms than the previous ones.

Keywords: graph; label; ELM

1. Introduction

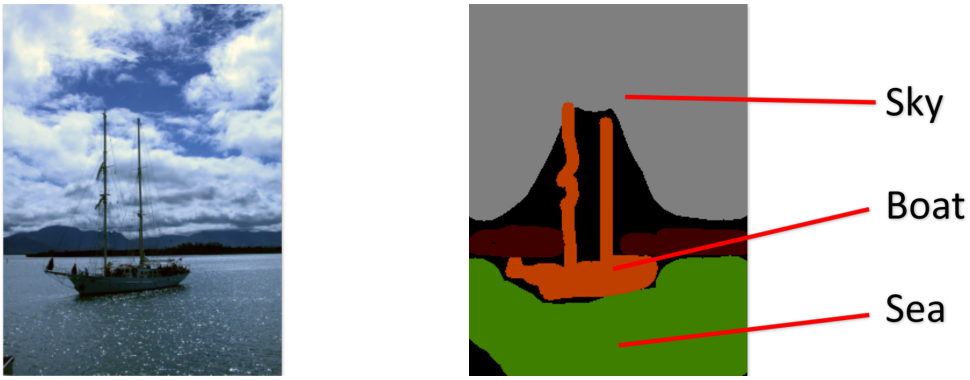
Due to the advanced smart phones, nowadays people uploaded a great number of photos to the Internet. Updating photos becomes easier, but searching them becomes more difficult. Though the technology of searching images by images has appeared, most people rely on the traditional way to searching an image, which is searching images by typing key words. For that, we need to add labels for each of image, but that cannot be accomplished by human beings due to the great number of unlabeled images. Thus, it is important to use *Machine Learning* to automatically classify images[4][13] and add correct labels for them, and that is the reason why we proposed the algorithm *Multi-Graph Multi-Label* (*MGML*) *Learning*. This algorithm may also be used in calculating the similarity of biological sequences, predicting the function of chemical compounds, analyzing structured texts such as HTML and XML, etc.

The advantages of the *MGML* is twofold: 1) **Multi-Graph**. *Multi-Instance Learning* in image classification is widely used[3]. It uses a kind of data structure *Feature Vector* to represent a real image[10]. However, it will be better to use the data structure *Graph* to represent it[11], because a

Graph has nodes and edges. The nodes can indicate the texture or color of pixels in an image and the edges can indicate the adjacency relation of nodes. Like the following image in Figure 1a, a *Feature Vector* can only tell you there are white, blue and green pixels in this image, but a *Graph* can tell the adjacency relation between each nodes like the Figure 1c. The latter one restores more details in a real image, which will be more benefited in the learning part. 2) **Multi-Label**. In the real life, it is impossible that one image just contains one subject. It often includes multiple semantic information[12]. Like Figure 2a, you cannot just add it with a single label like "sky", "boat" or "sea". For example, the following image is just labeled by "sea". Then, once the user wants to search the "boat", this image will not be shown in the result. Cases of other labels are in the similar way. Consequently, three labels must be added simultaneously.

(a) Original image **(b)** Segmented **(c)** Represented in Graph

Figure 1. An example figure with structure Graph



(a) Original image **(b)** Segmented and labeled

Figure 2. An example figure with Multi-Label

Although *MGML* is useful, there still exists many difficulties to overcome. The first and most difficult one is how to choose the appropriate features as the category basic. One training dataset has hundreds or even thousands of images and one image can break up into multiple graphs[9], so there will be too many graphs to be analyzed. Therefore, we need to select some most informative subgraphs among them and use them as the category features. A widespread way is to mine the most frequent subgraphs[2]. It is common to consider a subgraph is more informative if it is more frequent in the dataset, but it is not always feasible. In the Figure 3, there are eight graph structures. The above four structures are classified

as positive class and the below ones are classified as negative class (This can also be seen as two kinds of graphs with different labels). If we use the frequent-subgraphs mining, the subgraph $A-B$ has the frequency of eight because it appears eight times in all structures, but if we use it as an informative subgraph (a.k.a. distinguishing feature), it cannot tell the positive class from the negative one. Another subgraph $B-C$ only has the frequency of three, but it appears three times in the positive structures and does not appear in the negative structures, which is very suitable to be as an informative subgraph. The above idea of selecting features uses the concept of *Entropy* (a.k.a. *Information Gain*), and we can see it is not enough to use only frequent subgraphs for classifying a complex dataset[8].

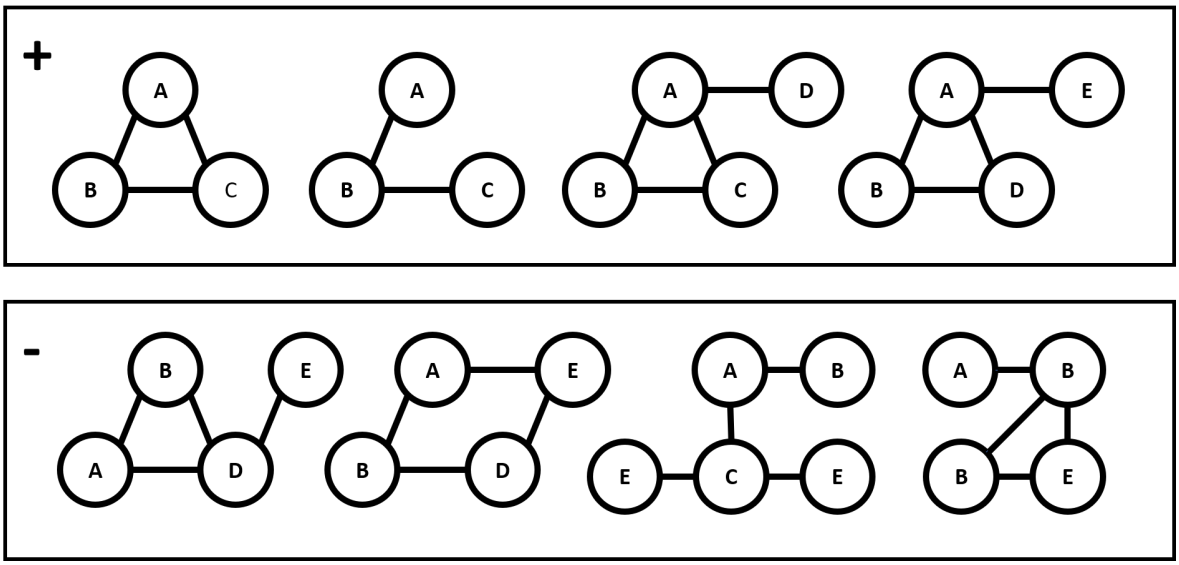


Figure 3. A graph dataset with class label

The second problem is how to classify graph-structure data. As far as we known, the currently existed subgraph-feature methods cannot solve our *MGML* problem. Most methods can only solve the single-graph classification because they need each graph to be explicitly labeled, but in the multi-graph classification, a label is only available (i.e. accessible) for a bag of graphs. Therefore, those methods cannot be used directly in our problem. Recently, A multi-graph learning algorithm *gMGFL*[5][17][21][23] was proposed by Jia Wu et al, but it cannot deal with multi-label problem. What's more, this algorithm just degenerates the multi-graph problem into single-graph one by using *Principal Components Analysis* (*PAC*), which still uses frequent subgraphs to distinguish and may cause the above issue mentioned in the last paragraph. In short, **there is still no algorithms which can solve the *MGML* problem properly and correctly**.

In this paper, we proposed an algorithm *Multi-Graph Multi-Label Learning* which using the *Entropy* to solve our problem. It has the following major contributions:

(1) As far as we known, we are the first one to propose the *MGML* algorithm.

(2) We propose a novel algorithm called *Informative Subgraph Mining* (*ISM*), which calculate the entropy for each graph and uses this to mine the most informative subgraphs.

(3) After mining informative subgraphs, we use the *Extreme Learning Machine* (*ELM*) to build an image classifier.

The rest part of this paper is organized as the following. Related works are introduced in Section 2. The algorithm description is provided in Section 3. Experimental results are presented in Section 4. We conclude this paper in Section 5.

2. Related Works

Our research is related to some previous works of graph-structure classification, *Multi-Instance Multi-Label* (*MIML*) learning and *Extreme Learning Machine* (*ELM*). We will briefly review them respectively in Section 2.1, 2.2 and 2.3.

2.1. Graph-Structure Classification and gSpan

There are two kinds of algorithms about graph-structure classification: (1) basing on global distance and (2) basing on subgraph feature, and it has been proved that the subgraph-feature approach is better[31]. It converts a set of subgraphs into feature vector so most of the current algorithms can be used in the graph classification problem. Almost all this kind of algorithms (such as *AGM*[41], *Gaston*[38], *gSpan*[32][39], *gHSIC*[28][29]) extract sub-graph feature by using frequent substructure pattern mining, and the most widespread mining algorithm among them is *gSpan*.

Recently, *gMGFL*[5][17][21][23] based on *gSpan* was proposed by Jia Wu et al, but it can only handle a set of graphs with one label and it regards the frequent subgraphs as the informative subgraphs which may cause the problem we discussed in Section 1.

2.2. Multi-Instance Multi-Label (MIML) Learning

MIML Learning is a supervised learning algorithm, which represents real-world objects with bags of instances and labels. The most widespread algorithm is *MIML-SVM*[22], which solves the problem by degenerating it into a *Single-Instance Multi-Label* (*SIML*) problem through a specific clustering process and transform the multiple labels into some binary classification tasks through *Support Vector Machine* (*SVM*). *MIML-SVM* was proposed by Zhou et al in [34], and recently improved by Li et al in [26]. Then, a more efficient and effective algorithm was proposed: *MIML-ELM*[14]. Firstly, the degeneration from *MIML* to *SIML* will be more effective because *MIML-ELM* develops a method of theoretical guarantee to determine the number of clusters automatically. Secondly, it uses *Extreme Learning Machine* (*ELM*) instead of *SVM* to improve the precdiction performance.

2.3. Extreme Learning Machine (ELM)

ELM is one of the models in *Neural Network* and is widely used in *Single Hidden-layer Feed-forward Network*. Recently, Li et al used the *ELM* in solving the *Multi-Instance Multi-Label* (*MIML*) problem in [14]. Firstly, it provides a guarantee to theoretically determine the number of clusters so that the *MIML* can transform into *SIML* (*Single-Instance Multi-Label*). Furthermore, it uses *ELM* instead of the traditional way *SVM* to improve the efficiency of the two-phase framework. Lastly, it proposed a genetic algorithm based on *ELM* to improve the prediction performance.

3. MGML Algorithm

To begin with, we introduce some relative concepts in Section 3.1. Furthermore, we discuss our proposed approach in Section 3.2, Section 3.3 and Section 3.4.

3.1. Problem Definition

In this section, we define some related concepts[21] before discussing the whole algorithm.

Definition 3.1. (Connected Graph): A graph G is represented as $G = (V, E, L, l)$ where V is a set of vertices $V = \{v_1, \ldots, v_n\}$, E is a set of edges and $E \subseteq V \times V$, and L is the set of labels for the vertices and edges. l is the function assigning labels to the vertices and edges which has $l : V \cup E \to L$. A connected graph is a graph that has a path between any pair of vertices. Besides, all graphs discussed in our paper are connected graphs.

Definition 3.2. (Subgraph): Let $G = (V, E, L, l)$ and $g_k = (V', E', L', l')$ each denote a connected graph. g_k is a sub-graph of G, i.e. $g_k subseteq G$, iff there exists an injective function $\varphi : V' \to V$ st. (1) $\forall v \in V', l'(v) = l(\varphi(v))$; (2) $\forall (u, v) \in E', (\varphi(u), \varphi(v)) \in E$, and $l'(u, v) = l(\varphi(u), \varphi(v))$. If g_k is a sub-graph of G, then G is a super-graph of g_k.

Definition 3.3. (Graph Bag): A graph bag $B_i = \{G_1^i, \ldots, G_j^i, \ldots, G_{n_i}^i\}$ contains a number of graphs, where G_j^i and n_i denotes the jth graph and the total number of graphs in the bag, respectively. For ease of representation, we also use G_j to denote the jth graph in a given bag. A graph bag B_i's labels is denoted by $y_i \in \{0, 1\}^M$.

Definition 3.4. (Subgraph Feature Representation for Graph): Let $S_g = \{g_1, \ldots, g_k, \ldots, g_s\}$ denote a set of sub-graphs discovered from a given graph set. For each graph G_i, we use a subgraph feature vector $X_i^G = [(X_i^{g_1})^G, \ldots, X_i^{g_k})^G, \ldots, X_i^{g_s})^G]^T$ to represent G_i in the feature space, where $(X_i^{g_k})^G = 1$ iff g_k is a subgraph of G_i and $(X_i^{g_k})^G = 0$ otherwise.

Definition 3.5. (Subgraph Feature Representation for Bag): Given a set of sub-graphs $S_g = \{g_1, \ldots, g_k, \ldots, g_s\}$, a graph bag B_i can be represented by a feature vector $X_i^B = [(X_i^{g_1})^B, \ldots, X_i^{g_k})^B, \ldots, X_i^{g_s})^B]^T$ where $(X_i^{g_k})^B = 1$ iff g_k is a sub-graph of any graph G_i in bag B_i and $(X_i^{g_k})^B = 0$ otherwise.

3.2. Overall Framework of MGML

Here is an overall framework of *MGML* learning, which includes the following steps: 1) **Mining Informative Subgraphs Based on Entropy**. Mining the most informative subgraphs is the key step to find the suitable features to represent the multi-graph bags. The greater entropy a subgraph has, the more distinguishing it should be. In order to find all the graph subgraphs and their derive structures, we use the improved gSpan[39] algorithm in the graphs set. 2) **Degenerating into MIML**. One kind of informative subgraph could be seen as one a kind of distinguishing feature. One graph could represent into an instance based on what kinds of distinguishing features it contains. One graph bag could transform into

an instance bag with the labels which originally attached to the graph bag. After that, we can use the MIML-ELM, an efficient and effective MIML algorithm to build a classifier.

3.3. Mining Informative Subgraphs Based on Entropy

In this section, we discuss how to evaluate the most informative subgraphs and then hwo to find the informative subgraph set.

3.3.1. Evaluation Function of Informative Subgraphs

The evaluation function of informative subgraphs is used to evaluate all candidate subgraphs and find a group of subgraphs which is the most informative. Then we can use this group of subgraphs to represent the multi-graph bags.

Assume there is a graph bag B and a set of graphs S_B which is the complete set of subgraphs discovered from B. g is a set of subgraphs mining from S_B and $g = \{g_1, \ldots, g_m\}$. We want to find a set of subgraphs g' which is the most informative. For that, we define $E(g)$ as the evaluation function to measure the informative ability of g, so it can be defined in the Equation 1.

$$g^* = \arg\max_{g \subseteq S_g}(E(g)) \quad s.t. \quad |g| = m \tag{1}$$

m is the number of subgraphs selected from S_B. The Equation 1 indicates that the set of subgraphs g^* should be the most informative one. Then we need to define what is the most informative subgraph. At first, we assume the graph bag only has one label, so each graph among it is either positive or negative.

Consequently, the definition for the entropy of subgraphs will be like this: A set of graphs S_B has a set of subgraphs $g = \{g_1, \ldots, g_m\}$. E is the entropy of the subgraph in g and $E = -p_{pos}log_2(p_{pos}) - p_{neg}log_2(p_{neg})$. The p_{pos} stands for the possibility of the subgraph appears in the positive graphs and the p_{pos} stands for the one in the negative graphs, which has $p_{pos} = \frac{times \quad appeared \quad in \quad positive \quad graphs}{num \quad of \quad positive \quad graphs}$ and $p_{neg} = \frac{times \quad appeared \quad in \quad negative \quad graphs}{num \quad of \quad negative \quad graphs}$. Thus, the entropy for g is $E(g) = \{E_1, \ldots, E_m\}$.

As for the informative subgraph, the definition will be: The entropy $E(g)$ of subgraphs $g = \{g_1, \ldots, g_m\}$ is $E(g) = \{E_1, \ldots, E_m\}$. E_i is the minimum entropy among $E(g)$, which is $E_i = \min\{E_1, \ldots, E_m\}$, and the i is the number of subgraph. Thus, g_i has the greatest entropy among g and we define it as the most informative subgraph.

The reason why we take the subgraph with the minimum entropy as the informative one is that: if the possibility of the subgraph appears in the positive graphs and the negative ones is equal, $p_{pos} = p_{neg} = \frac{1}{2}$, then the entropy $E = 1$. The entropy reaches to its maximum, and that subgraph is not what we want because if we use it as a feature, it cannot tell the positive graphs from the negative one. If the subgraph only appears in the positive graphs or the negative ones, $p_{pos} = 1$ or $p_{neg} = 1$, then the entropy $E = 0$. The entropy reaches to its minimum, and that subgraph is what we want.

Above definitions only work in the single-label problem. We still need to define the informative subgraphs in the multi-label problem: A set of graphs S_B has a set of subgraphs $g = \{g_1, \ldots, g_m\}$ and

it has a set of labels $l = \{l_1, \ldots, l_n\}$. For each label, the subgraphs g has an entropy $E(g)_i \quad 1 \leq i \leq n$ and $E(g)_i = \{E_{i,1}, \ldots, E_{i,m}\}$. The matrix of entropy is

$$e(g) = \begin{bmatrix} E_{1,1} & \ldots & E_{1,m} \\ \ldots & \ldots & \ldots \\ E_{n,1} & \ldots & E_{n,m} \end{bmatrix} \tag{2}$$

For each subgraph, it has a set of entropy $E(g)_j \quad 1 \leq j \leq m$ and $E(g)_j = [E_{1,j}, \ldots, E_{n,j}]^T$. The average entropy for one subgraph is $E'(g)_j = \arg\{E_{1,j}, \ldots, E_{n,j}\}$, so the average entropy for the set of subgraphs is $E'(g) = \{E'(g)_1, \ldots, E'(g)_m\}$. Similarly, E'_p is the minimum entropy among $E'(g)$, which is $E'_p = \min\{E'(g)_1, \ldots, E'(g)_m\}$, and the p is the number of subgraph. Thus, g_p has the greatest entropy among g and we define it as the most informative subgraph in the multi-label problem.

With that, our goal is to find a set of informative subgraphs which is the top-n informative subgraphs among the graph bag(n is the number of subgraphs we want).

3.3.2. Informative Subgraphs Mining

Currently existed algorithms about classifying graph-structure data can be categorized into two groups: (1) basing on global distance, including *graph kernel*[20][25], *graph embedding*[30] and *graph transformation*[16], which calculates the rate of similarity between two graphs; (2) basing on subgraph feature[37], including *AGM*[41], *Gaston*[38], *gSpan*[32][39] and *gHSIC*[28][29], which convert a set of subgraphs into feature vector so that most widespread algorithms can be used. It has been proved that the latter one is better[31]. It converts a set of subgraphs into feature vector so most of the current algorithms can be used in the graph classification problem.

In order to select the most informative subgraphs as subgraph features, one of the straightforward approach is to find all the derive subgraphs of the graph set and rank these subgraphs with the evaluation function in Section 3.3.1, but such approach will cause a problem: the number of derive subgraphs grows exponentially when the size of graph set grows, so the enumeration will be a tough work. Alternatively, we use a *Depth-First-Search* (*DFS*) based on algorithm *gSpan* to find all the subgraphs, within comparing them with our evaluation.

The key idea of *gSpan* is to builds a lexicographic order among graphs which need to be mined, and then give each graph a unique label named *minimum DFS(Depth-First-Search) code*. *gSpan* uses the depth-first search strategy to mine the frequent subgraph with the DFS code. Each time which it needs to generate a new subgraph, it just recurs the character string (i.e. DFS code), so a subgraph-mining problem can be transformed into a substring-matching problem. Thus, the *gSpan* performs better than previous similar algorithms.

Generally, *gSpan* is used for mining the frequent subgraphs, but we do not care the frequency of subgraph. We only use the *gSpan* to traverse all subgraphs and evaluate them with the entropy during the traversal. The algorithm of *ISM* is described in Algorithm 1, 2.

Note that Algorithm 1 uses Algorithm 2 and Algorithm 2 is a recursion function. Firstly, in Algorithm 1 line 4-8, it will discover all subgraphs by growing the graph from one edge. The database will be shrunk at the end of each turn (Algorithm 1 line 7). Algorithm 2 is to grow all the graphs and find all their subgraphs. It will stop when the code of the subgraph is not a minimum code (Algorithm 2

Algorithm 1: GraphSet_Projection

Input : D: graph dataset with labels;
m: the number of sub-graph features to be selected;
δ: the maximum entropy of the selected graph set;
Output: S: mining result;

```
1 S^1 ←all 1-edge graphs in D;
2 sort S^1 in DFS lexicographic order;
3 S ← S^1;
4 foreach edge e ∈ S^1 do
5 |   initialize subgraph s with e, set s.D by graphs which contains e;
6 |   Informative_Subgraphs_Mining(D, m, δ);
7 |   D ← D − e;
8 end
```

line 1). In Algorithm 2 line 4-11, it will compute the entropy of the subgraph and build the result set. In Algorithm 2 line 12-16, it will grow the subgraph and do this function recursively.

3.4. Degenerating into MIML

After getting the informative subgraph, we can transform them into instances based on Definition 3.4. The labeled *MIML* set is $D = \{(B_i, Y_i)|i = 1, \ldots, N\}$, where N is the number of bags in dataset D and $B_i = \{X_1^i, \ldots, X_j^i, \ldots, X_{n_i}^i\}$ is an instance bag with n_i instances, $Y_i \in \{0, 1\}^M$ is the labels vector of bag B_i. Therefore, the *MGML* learning problem is degenerated into an *MIML* learning problem.

Traditionally, *Support Vector Machine* (*SVM*) is used to solve the *MIML*, but it has some drawbacks. First, *SVM* requires the user to input much number of parameters. Second, using *SVM* to build a classifier may cause a high computing cost and the performance depends on the specific parameters user defined. Thus, we choose to use the *ELM* to solve the *MIML* problem.

Firstly, *ELM* develops a theoretical guarantee to determine the number of clusters by *AIC*[7]. Secondly, a k-medoids cluster process is performed to transform from *MIML* into *SIML*. Then, the Hausdorff distance[6] is measure between two different multi-instance bags. Based on the Hausdorff distance, k-medoids cluster divides the dataset into k parts, the medoids of which are $M_1, \ldots, M_k$. At last, we train the classifier for each label with k-dimensional numerical vectors.

4. Experimental Section

The following experiments are performed on a PC with 2.60GHz Intel Core 2 CPU, 16GB main memory running Linux.

4.1. Datasets

Algorithm 2: Informative_Subgraphs_Mining

Input : D: graph dataset with labels;
m: the number of sub-graph features to be selected;
δ: the maximum entropy of the selected graph set;
Output: S: mining result;
s: subgraph;

```
if s ≠ min(s) then
    return;
end
compute the entropy E(s) of the subgraph s;
if |S| < m or E(s) > δ then
    S ← S ∪ {s};
end
if |S| > m then
    S ← S/ arg max_{s_i∈S} E(s_i);
end
δ = max_{s_i∈S} E(s_i);
enumerate s in each graph in D and count its children;
foreach c, c is s' child do
    s ← c;
    Informative_Subgraphs_Mining(D_s, m, δ);
end
```

Experiments are conducted on three image datasets with different size, including a small size of dataset named *MSRC v2* (Winn, Criminisi, and Minka 2005)[36], a middle size of dataset named *Scenes* (Zhou and Zhang 2007)[24][33] and a large size of dataset named *Corel5K*(Duygulu et al. 2002)[40].

(1) **MSRC v2**: *MSRC v2* is a subset of the Microsoft Research Cambridge (MSRC) image dataset[36]. It contains 591 images and 23 classes of labels. About 80% of images belong to more than one label and there are about three labels per image on average.

(2) **Scenes**: *Scene* contains 2,000 natural scene images and 5 classes of labels[24][33]. About 22% of images belong to more than one label.

(3) **Corel5K**: *Corel5K* has become the benchmark for image annotation recently[40]. It contains 5000 images, which are collected from a larger dataset named *Corel CD*, and 260 classes of labels. There are about 3.5 labels per image on average.

The summary of three datasets is given in the Table 1.

4.2. Evaluation Criteria

In multi-label learning, each object may be of several labels simultaneously. The commonly used evaluation criteria, such as accuracy, precision and recall, are not suitable in this case. In this paper,

Table 1. The information of datasets

Dataset	# of bags	# of labels	labels per bag
MSRC v2	591	23	2.5
Scenes	2,000	5	1.2
Corel5K	5,000	260	3.5

four widespread multi-label learning evaluation criteria[14], i.e., $one-error(OE)$, $coverage(CO)$, $rankingloss(RL)$ and $averageprecision(AP)$, are used to measure the performance of the proposed algorithm. Given a test data set $S = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_p, Y_p)\}$, the four criteria are defined as below, where $h(X_i)$ returns a set of proper labels of X_i, $h(X_i, y)$ returns a real-value indicating the confidence for to be a proper label of X_i, $rank^h(X_i, y)$ returns the rank of y derived from $h(X_i, y)$.

(1) $one-error_S(h) = \frac{1}{p}\sum_{i=1}^{p}[\arg\max_{y\in Y} h(X_i, y) \notin Y_i]$. The one-error evaluates how many times the top-ranked label is not a proper label of the object. The performance is perfect when $one-error_S(h) = 0$; the smaller the value of $one-error_S(h)$, the better the performance of h.

(2) $coverage_S(h) = \frac{1}{p}\max_{y\in Y} rank^h(X_i, y) - 1$. The coverage evaluates how far it is needed, on the average, to go down the list of labels in order to cover all the proper labels of the object. It is loosely related to precision at the level of perfect recall. The smaller the value of $coverage_S(h)$, the better the performance of h.

(3) $rloss_S(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{||Y_i||||\overline{Y_i}||}|\{(y_1, y_2)|h(X_i, y_1) \leq h(X_i, y_2), (y_1, y_2) \in Y_i \times \overline{Y_i}|$, where Y_i denotes the complementary set of Y_i in Y. The ranking loss evaluates the average fraction of label pairs that are misordered for the object. The performance is perfect when $rloss_S = 0$; the smaller the value of $rloss_S(h)$, the better the performance of h.

(4) $avgprec_S(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{|Y_i|}\sum_{y\in Y_i}\frac{|\{y'|rank^h(X_i,y')\leq rank^h(X_i,y')\leq rank^h(X_i,y), y'\in Y_i\}|}{rank^h(X_i,y)}$. The average precision evaluates the average fraction of proper labels ranked above a particular label $y \in Y_i$. The performance is perfect when $avgprec_S(h) = 1$; the larger the value of $avgprec_S(h)$, the better the performance of h.

4.3. Effectiveness

The four criteria in Section 4.2 are used for performance evaluation, but currently there is no other methods of *MGML* learning which can be compared to our algorithm. Thus, we use the *MIMLSVM*[30], one of the state-of-the-art algorithms for *MIML* learning, as the competitor. Besides, we use the *MGMLSVM* as the baseline algorithm for competitions. The *MGMLSVM* algorithm is generally as same as the *MGMLELM*. It also needs to degenerate the *MGML* problem into the MIML problem, but then it uses the *SVM* instead of *ELM* in the next step.

The *MIMLSVM* and *MGMLSVM* algorithms are implemented with a *Gaussian* kernel and the penalty factor *Cost* is set to 1, 2, 3, 4, 5. The *MGMLELM* is implemented with the number of hidden layer nodes *hn*, which is set to 50, 100, 150, 200. On each dataset, the data are randomly divided into a training set and a testing set according to the ratio $2:1$. The training set is used to build a predictive model, and the

testing set is used to evaluate its performance. All experiments repeatedly run thirty times, and each time the training set and the testing set will be divided randomly. The final average results are in Tables 2-4. The best performance on each criterion is bolded. The ↓ indicates *the smaller the better*, while the ↑ indicates *the bigger the better*.

Table 2. MSRC v2 Dataset

MSRC v2		Evaluation Criterion			
		RankingLoss↓	OneError↓	Coverage↓	Average Precision↑
MGMIELM	hn = 50	0.070079	0.183039	3.92824	0.809013
	hn = 100	**0.071367**	**0.172589**	**3.928934**	**0.820388**
	hn = 150	0.075182	0.19797	3.989848	0.804771
	hn = 200	0.07181	0.187817	3.86802	0.808192
MGMLSVM	Cost = 1	0.154664	0.19797	7.35533	0.754622
	Cost = 2	0.171189	0.229066	7.122844	0.761665
	Cost = 3	0.183357	0.233503	7.634518	0.735131
	Cost = 4	0.140284	0.219557	7.628352	0.735253
	Cost = 5	**0.137361**	**0.187817**	**6.187817**	**0.762115**
MIMLSVM	Cost = 1	0.105581	0.295073	5.267476	0.710714
	Cost = 2	0.104209	0.292204	5.218223	0.715079
	Cost = 3	0.100998	0.253995	5.044584	0.721987
	Cost = 4	0.097587	0.247775	4.890471	0.73787
	Cost = 5	**0.0955**	**0.240682**	**4.880897**	**0.745322**

As seen from the results in Tables 2-4, in the dataset *MSRC v2*, our algorithm *MGMLELM* performs best when the $hn = 100$ and the precision reaches to 82%, while the precision of *MGMLSVM* reaches to 76% at best when $Cost = 5$ and *MIMLSVM* reaches to 75% at best when $Cost = 5$; in the dataset *Scenes*, our *MGMLELM* reaches to 81% at best when the $hn = 150$, while *MGMLSVM* reaches to 74% at best when $Cost = 3$ and *MIMLSVM* reaches to 25% at best when $Cost = 5$; in the dataset *Corel5K*, our *MGMLELM* reaches to 23% at best when the $hn = 200$, while *MGMLSVM* reaches to 13% at best when $Cost = 5$ and *MIMLSVM* reaches to 22% at best when $Cost = 5$. Thus, we can say that our *MGMLELM* achieves the best performance in all cases.

4.4. Efficiency

In this experiment, we test the efficiency of *MGMLELM*. For either *MGML* or *MIML*, as long as the input numbers of features to *ELM* or *SVM* are the same, the runtimes for both of these two algorithm equals. Therefore, the only way to test the efficiency is focusing on the part of generating the features. Our *MGMLELM* uses the original algorithm *ISM*, which is based on the entropy, to mining the informative subgraphs as the features, while the *MGML* of Jia, Wu mines the tranditional frequently subgraphs as the features, which is based on the boosting *gSpan* named *gboost*[35]. We compared the

Table 3. Scenes Dataset

Scene		Evaluation Criterion			
		RankingLoss↓	OneError↓	Coverage↓	Average Precision↑
MGMIELM	hn = 50	0.16927	0.318	0.771	0.798919
	hn = 100	0.172833	0.33	0.81	0.798367
	hn = 150	**0.165**	**0.304**	**0.78**	**0.811867**
	hn = 200	0.160667	0.312	0.762	0.8102
MGMLSVM	Cost = 1	0.299667	0.806	1.324	0.555683
	Cost = 2	0.298835	0.434	1.324	0.694689
	Cost = 3	**0.2935**	**0.34**	**1.284**	**0.7401**
	Cost = 4	0.252933	0.36	1.312	0.630968
	Cost = 5	0.237167	0.458	1.062	0.71515
MIMLSVM	Cost = 1	0.910205	0.950815	3.810687	0.242251
	Cost = 2	0.91073	0.95178	3.844675	0.242479
	Cost = 3	0.91348	0.956172	3.864988	0.245989
	Cost = 4	0.914378	0.957471	3.86676	0.246726
	Cost = 5	**0.917939**	**0.958322**	**3.867907**	**0.249013**

Table 4. Corel5K Dataset

Corel5K		Evaluation Criterion			
		RankingLoss↓	OneError↓	Coverage↓	Average Precision↑
MGMIELM	hn = 50	0.202493	0.750168	113.8968	0.224968
	hn = 100	0.197103	0.743487	113.354709	0.224146
	hn = 150	0.21584	0.755511	120.549098	0.219783
	hn = 200	**0.205424**	**0.751503**	**119.306613**	**0.225752**
MGMLSVM	Cost = 1	0.30124	0.857229	139.0005	0.120073
	Cost = 2	0.301264	0.86724	140.9756	0.121792
	Cost = 3	0.301906	0.868129	141.8067	0.122804
	Cost = 4	0.304838	0.870358	143.3142	0.123633
	Cost = 5	**0.307872**	**0.880693**	**144.8687**	**0.128766**
MIMLSVM	Cost = 1	0.191867	0.768118	110.4207	0.217195
	Cost = 2	0.191899	0.768204	110.4322	0.217209
	Cost = 3	0.191922	0.768299	110.4657	0.217219
	Cost = 4	0.191978	0.768416	110.4719	0.217285
	Cost = 5	**0.191997**	**0.768899**	**110.5187**	**0.217299**

time of our *ISM* for mining subgraphs with *gboost* in three datasets: *MSRC v2*, *Scenes* and *Corel5K*, respectively.

We use two ways to segement the original datasets: the first one is to segement each image into 6 graph structures and each graph has 8 nodes and 12 edges roughly($6 * 8 * 12$); the second one is to

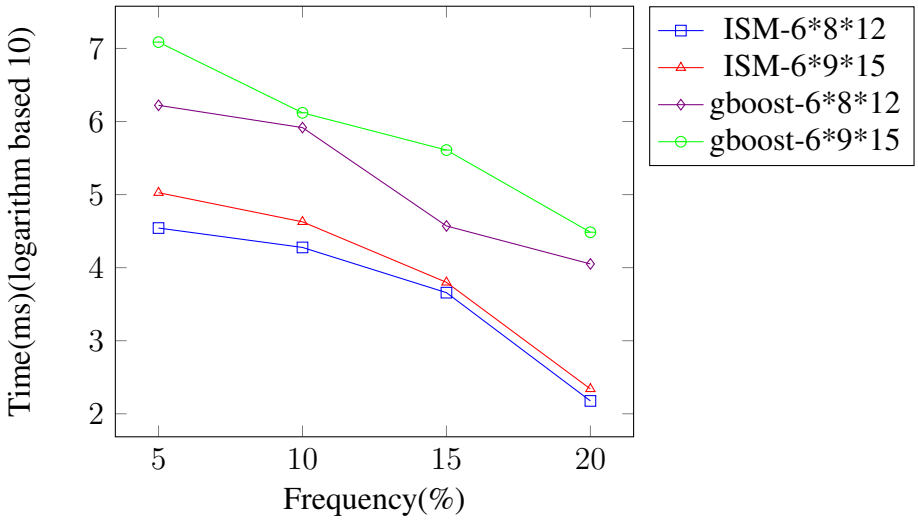


Figure 4. MSRC v2 Dataset

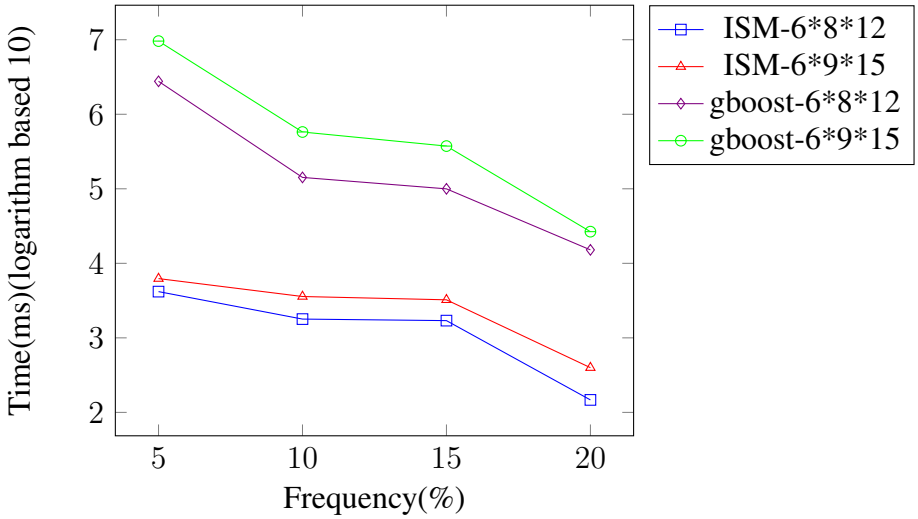


Figure 5. Scenes Dataset

segement each image into 6 graph structures and each graph has 9 nodes and 15 edges roughly($6*9*15$). We implement *ISM* and *gboost* in these two segmented sets and the minimum frequency is set to 5%, 10%, 15%, 20%. In *ISM*, if the frequency of a graph is lower than the minimum frequency we set, we won't calculate the entropy for this graph; while in *gboost*, if the frequency of a graph is lower than the minimum frequency, we won't continue to mine the subgraphs for this graph. In short, graphs in $6 * 9 * 15$ set are more complex to mining than the $6 * 8 * 12$ one; the lower the minimum frequency is, the more graphs the algorithm needs to mine. All experiments repeatedly run thirty times. The final average results are in Figure 4-6. The time results are logarithmic.

As seen from the results in Figure 4-6, *gboost* takes hours to mining the huge datasets, but *ISM* takes only several minutes to generate results. These figures show that *ISM* achieves better performance by 100-1000 times in comparison with *gboost*.

5. Conclusions

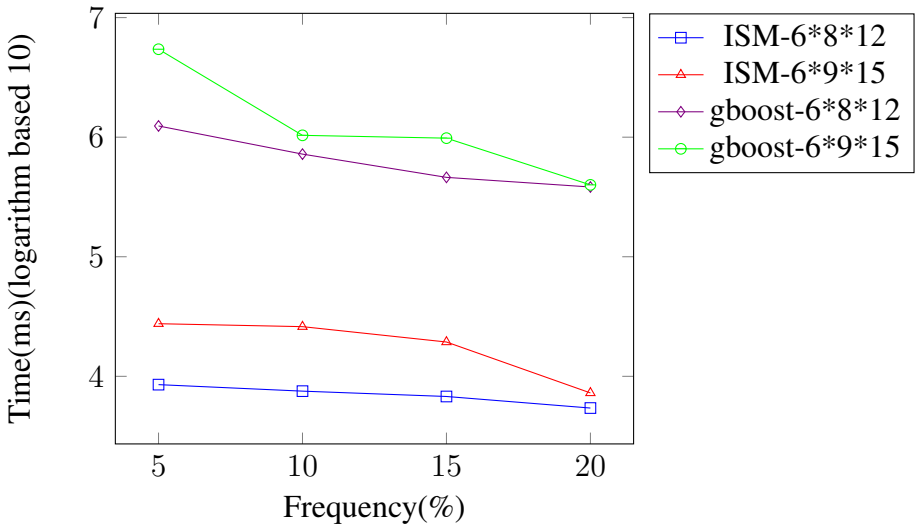


Figure 6. Corel5K Dataset

In this paper, we have shown how the *MGML* learning organized. The *MGML* uses multiple graphs instead of instances to represent an image, which can improve the accuracy of classification dramatically. Also, it uses multiple labels as the output to eliminate the ambiguity of description. Then we show how to degenerate the *MGML* problem into the *MIML* by mining the informative subgraphs. At last, we use the *ELM* as the base classifier. As the experiences showing, *MGMLELM* achieved a good performance in three image datasets. What we are interested in the future steps is to improve the performance in the dataset owning sparse labels by using other algorithms as the classifier.

References

1. Ibrahim, W.; Abadeh, M. Extracting features from protein sequences to improve deep extreme learning machine for protein fold recognition. *Journal of Theoretical Biology* **21 May 2017**, *Vol.421*, 1-15.
2. Nair, J.; Thomas, S.; Thampi, S.; El-Alfy, E. Improvised Apriori with frequent subgraph tree for extracting frequent subgraphs. *Journal of Intelligent & Fuzzy Systems* **03/29/2017**, *Vol.32(4)*, 3209-3219.
3. Tang, P.; Wang, X.; Feng, B.; Liu, W. Learning Multi-Instance Deep Discriminative Patterns for Image Classification. *IEEE Transactions on Image Processing* **July 2017**, *Vol.26(7)*, 3385-3396.
4. Wang, W.; Vong, C.; Yang, Y.; Wong, P. Encrypted image classification based on multilayer extreme learning machine. *Multidimensional Systems and Signal Processing* **2017**, *Vol.28(3)*, 851(15).
5. Wu, J.; Pan, S.; Zhu, X.; Zhang, C.; Wu, X. Positive and Unlabeled Multi-Graph Learning. *IEEE Transactions on Cybernetics* **Apr. 2017**, *vol. 47, no. 4*, 818-829.
6. Balankin, A.; Mena, B.; Cruz, M. Topological Hausdorff dimension and geodesic metric of critical percolation cluster in two dimensions. *Physics Letters A* **2017**, *381(33)*, 2665-2672.
7. Maruping, L.; Venkatesh, V.; Brown, S. Going beyond intention: Integrating behavioral expectation into the unified theory of acceptance and use of technology. *Journal of the Association for Information Science and Technology* **2017**, *68(3)*, 623-637.

8. Bai, L.; Hancock, E. Fast depth-based subgraph kernels for unattributed graphs. *Pattern Recognition* **February 2016**, *Vol.50*, 233-245.
9. Wang, W.; Li, Z.; Yue, J.; Li, D. Image segmentation incorporating double-mask via graph cuts. *Computers and Electrical Engineering* **August 2016**, *Vol.54*, 246-254.
10. Chaiyakhan, K.; Kerdprasop, N.; Kerdprasop, K. Feature Selection Techniques for Breast Cancer Image Classification with Support Vector Machine. *Lecture Notes in Engineering and Computer Science* **01 March 2016**, *Vol.2221(1)*, 327-332.
11. Bashier, H.; Hoe, L.; Hui, L.; Azli, M.; Han, P.; Kwee, W.; Sayeed, M. Texture classification via extended local graph structure. *Optik - International Journal for Light and Electron Optics* **January 2016**, *Vol.127(2)*, 638-643.
12. Wei, Y.; Xia, W.; Lin, M.; Huang, J.; Ni, B.; Dong, J.; Zhao, Y.; Yan, S. HCP: A Flexible CNN Framework for Multi-Label Image Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **Sept. 2016**, *Vol.38(9)*, 1901-1907.
13. Yan, K.; Li, Z.; Zhang, C. A New multi-instance multi-label learning approach for image and text classification. *Multimedia Tools and Applications* **2016**, *Vol.75(13)*, 7875-7890.
14. Yin, Y.; Zhao, Y.; Li, C.; Zhang, B. Improving Multi-Instance Multi-Label Learning by Extreme Learning Machine *Applied Sciences* **01 May 2016**, *Vol.6(6)*, 160.
15. Li, F.; Zhang, Z.; Jin, C. Feature selection with partition differentiation entropy for large-scale data sets. *Information Sciences* **Feb. 2016**, *Vol. 329*, 690-700.
16. Hidaka, S.; Tisi, M.; Cabot, J.; Hu, Z. Feature-based classification of bidirectional transformation approaches. *Software & Systems Modeling* **2016**, *Vol.15(3)*, 907-928.
17. Wu, J.; Hong, J.; Pan, S. et al. Multi-graph-view subgraph mining for graph classification. *Knowledge and Information Systems* **2016**, *Vol.48(1)*, 29-54.
18. Jurado, S.; Nebot, ÃĂ.; Mugica, F.; Avellana, N. Hybrid methodologies for electricity load forecasting: Entropy-based feature selection with machine learning and soft computing techniques. *Energy* **Jun. 2015**, *Vol. 86*, 276-291.
19. Jiang, F.; Sui, Y.; Zhou, L. A relative decision entropy-based feature selection approach. *Pattern Recognition* **Jul. 2015**, *Vol. 48(7)*, 2151-2163.
20. Martino, G.; Navarin, N.; Sperduti, A. Graph Kernels exploiting Weisfeiler-Lehman Graph Isomorphism Test Extensions. *Neural Information Processing* **2014**, *Volume 8835*, 93-100.
21. Wu, J.; Zhu, X.; Zhang, C.; Yu, P.S. Bag Constrained Structure Pattern Mining for Multi-Graph Classification. *IEEE Transactions on Knowledge and Data Engineering* **Oct. 2014**, *vol. 26, no. 10*, 2382-2396.
22. Wu, J., Huang, S., Zhou, Z. Genome-wide protein function prediction through multi-instance multi-label learning. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 11(5)* **2014**, 891-902.
23. Wu, J.; Zhu, X.; Zhang, C.; Cai, Z. Multi-instance Multi-graph Dual Embedding Learning. *2013 IEEE 13th International Conference on Data Mining* **Dec. 2013**, 827-836.
24. Zhou, Z.; Zhang, M.; Huang, S.; Li, Y. Multi-instance multilabel learning. *Artif. Intell.* **2012**, *vol. 176, no. 1*, 2291-2320.

25. Seeland, M.; Kramer, A.K.S. A Structural Cluster Kernel for Learning on Graphs. *Proc. 18th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD)* **2012**, 516-524.
26. Li, Y., Ji, S., Kumar, S., Ye, J., Zhou, Z. Drosophila gene expression pattern annotation through multi-instance multi-label learning. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 9, no. 1* **2012**, 98-112.
27. Largeron, C.; Moulin, C.; GÃl'ry, M. Entropy based feature selection for text categorization. *2011 ACM Symposium on Applied Computing* **Mar. 2011**, 924-928.
28. Zhao, Y.; Kong, X.; Yu, P.S. Positive and Unlabeled Learning for Graph Classification. *Proc. IEEE 11th Int'l Conf. Data Mining (ICDM)* **2011**, 962-971.
29. Kong, X.; Yu, P. Multi-Label Feature Selection for Graph Classification. *Proc. 10th Int'l Conf. Data Mining (ICDM)* **2010**, 274-283.
30. Riesen, K.; Bunke, H. Graph Classification by Means of Lipschitz Embedding. *IEEE Trans. Systems, Man, and Cybernetics Part B: Cybernetics* **2009**, *vol. 39*, 1472-1483.
31. Ketkar, N.; Holder, L.; Cook, D. Empirical Comparison of Graph Classification Algorithms. *Proc. IEEE Symp. Computational Intelligence and Data Mining (CIDM)* **March 2009**, 259-266.
32. Saigo, H.; Nowozin, S.; Kadowaki, T.; Kudo, T.; Tsuda, K. gBOOST: A Math. Programming Approach to Graph Classification And Regression. *Machine Learning* **2009**, *vol. 75*, 69-89.
33. Zhou, Z.; Zhang, M. Multi-instance multi-label learning with application to scene classification. *Advances in Neural Information Processing Systems 19. Cambridge, MA, USA: MIT Press* **2007**, 1609-1616.
34. Zhou, Z.; Zhang, M. Multi-instance multi-label learning with application to scene classification. *In Advances in neural information processing systems* **2007**, 1609-1616.
35. Nowozin, S.; Tsuda, K.; Uno, T.; Kudo, T; Bakir, G. Weighted Substructure Mining for Image Analysis. *Computer Vision and Pattern Recognition* **2007**, 1-8.
36. Winn, J.; Criminisi, A.; Minka, T. Object categorization by learned universal visual dictionary. *IEEE 10th Int. Conf. Comput. Vis.* **2005**, *vol. 2*, 1800-1807.
37. Deshpande, M.; Kuramochi, M.; Wale, N.; Karypis, G. Frequent Substructure-Based Approaches for Classifying Chemical Compounds. *IEEE Trans. Knowledge and Data Eng.* **Aug. 2005**, *vol. 17, no. 8*, 1036-1050.
38. Nijssen, S.; Kok, J. A Quickstart in Frequent Structure Mining can Make a Difference. *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD)* **2004**, 647-652.
39. Yan, X.; Han, J. gSpan: Graph-Based Substructure Pattern Mining. *Proc. IEEE Int'l Conf. Data Mining (ICDM)* **2002**, 721-724.
40. Duygulu, P.; Barnard, K.; Freitas, J.; Forsyth, D. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. *Proc. 7th Eur. Conf. Comput. Vis.* **2002**, 97-112.
41. Inokuchi, A.; Washio, T.; Motoda, H. An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. *Proc. Fourth European Conf. Principles of Data Mining and Knowledge Discovery (PKDD)* **2000**, 13-23.

© 2017 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/).