

A Probabilistic Substructure-Based Approach for Graph Classification

H.D.K. Moonesinghe, Hamed Valizadegan, Samah Fodeh, Pang-Ning Tan

Department of Computer Science & Engineering

Michigan State University

East Lansing, MI 48824

(moonessin, valizade, fodehsam, ptan) @cse.msu.edu

Abstract

The classification of graph based objects is an important challenge from a knowledge discovery standpoint and has attracted considerable attention recently. In this paper, we present a probabilistic substructure-based approach for classifying a graph-based dataset. More specifically, we use a frequent subgraph mining algorithm to construct substructure based descriptors and apply the maximum entropy principle to convert the local patterns into a global classification model for graph data. Empirical studies conducted on real world data sets showed that the maximum entropy substructure-based approach often outperforms existing feature vector methods using AdaBoost and Support Vector Machine.

Keywords: *Graph classification, Maximum entropy, frequent subgraph mining.*

1. Introduction

Developing computational techniques based on classification to determine the identity of data represented as graphs has become a focused theme in data mining research. Graph classification has huge potential benefits in a variety of applications. For example, detecting spam web pages has become an important research problem because of the dramatic growth of spam web sites and the adverse effect it causes to degrade the quality of search results produced by the web search engines. Graph-based classification can also be applied to the drug discovery problems, where it is useful to learn the structural properties of a chemical compound and the effect that they have on the treatment of a particular disease.

Over the years, several innovative classification algorithms for graph-based data have been developed [18, 19, 20, 21, 22, 24]. Most of the algorithms are based on the underlying assumption that the intrinsic properties

of a graph are rendered by its underlying substructures (nodes, edges, paths, strongly connected components, trees, subgraphs, etc). These substructure-based algorithms identify the important components present in each graph and subsequently use them to discriminate graphs from different classes. Most of the early works in graph-based classification have focused on the use of heuristic based search techniques to discover such discriminative components present in the graph database [24]. More recently, however, frequent subgraph based approach was proposed in which frequent subgraph mining algorithms are employed to generate all the subgraphs that occur a sufficiently large number of times in the graph database and to construct feature vectors based on the extracted subgraphs [18]. By transforming each graph into its corresponding feature vector, we can subsequently apply any of the existing classification algorithms such as Support Vector Machine (SVM), boosting, decision trees, or rule-based classifiers to build a classification model for the graph database.

Since the main focus of these algorithms is on extracting a local set of relevant patterns that provide useful insight into the data, it is not obvious how this local information can be best synthesized into a global prediction model that can be used to determine the class of previously unknown graphs. In this paper, we present a principled approach for building a global classification model from the local patterns using the maximum entropy principle. The idea behind this approach is to learn a conditional probability distribution of the class for a graph given its underlying features (i.e., frequent subgraphs) by using the support of the features as constraints imposed on the probability model. Using an iterative technique known as the *improved iterative-scaling* algorithm [6], the parameters of the probability model can be inferred from the data. While the idea of using maximum entropy principle for constructing a global model based on its underlying local patterns is not new [4], to the best of our knowledge, this approach has not been applied to graph-based data.

Another aspect of this study is to compare the performance of the probabilistic substructure-based approach against existing substructure-based approaches that apply standard classification algorithms (such as support vector machine and boosting) to the feature vectors constructed from the frequent subgraphs. Experimental results using real world data sets show that the maximum entropy substructure-based approach often outperforms other approaches and yields better accuracy.

The remainder of the paper is organized as follows. In section 2, we describe the preliminaries and related research in graph classification. Section 3 introduces our proposed maximum entropy approach for classification. Section 4 presents our graph classification algorithm. In section 5, we perform an extensive performance evaluation on real world data sets. Finally, we conclude our work in section 6.

2. Background

In this section we describe the preliminaries, related research in graph classification, and the algorithm we used to generate frequent subgraph patterns.

2.1 Preliminaries

We first revisit some of the basic definitions from graph theory and graph mining.

Let $g = (V, E)$ be a graph, where V is a finite set of objects called vertices (or nodes) and E is a set of 2-element subsets of V called edges. A labeled graph is represented by a 4-tuple $g = (V, E, L, l)$, where L is a set of labels, and a label function $l: V \cup E \rightarrow L$ maps a vertex or an edge to a label. If $e = uv$ is an edge of a graph g , then we say vertices u and v are *adjacent* in g . A graph $g_1 = (V_1, E_1, L_1, l_1)$ is a *subgraph* of another graph $g_2 = (V_2, E_2, L_2, l_2)$ if the following conditions hold: $V_1 \subseteq V_2$, $E_1 \subseteq E_2$, $L_1 \subseteq L_2$, and $l_1 = l_2$; i.e. there exists a subgraph isomorphism from g_1 to g_2 , denoted as $g_1 \subseteq g_2$. Also, graph g_2 is called a *super-graph* of g_1 .

In frequent subgraph mining, a subgraph g is said to be *frequent* if the occurrence of g (i.e. $\sigma(g)$) in a given graph database $G = \{g_i | i = 0..n\}$ is greater than or equal to a user specified minimum support threshold ξ . The problem of *frequent subgraph mining* is to find all frequent subgraphs in the graph database G .

2.2 Related Research

Graph classification has received lots of attention because of its wide range of applicability to real-world data such as biological data, chemical compound data, and semi structured data to name a few. Several approaches have been developed [18, 19, 20, 21, 22] for

building efficient classifiers for this task. In most of these approaches a graph is represented using various descriptors and a classification model is built using statistical or machine learning techniques. Karypis et al. [18] use similar approach by generating frequent substructure based descriptors using a frequent subgraph mining algorithm and selecting the best substructures to define the feature vectors, which is subsequently used by Support Vector Machine (SVM) classifiers to build the classification model. Han et al. [22] showed that frequent closed graphs based descriptor space is a better approach than the frequent subgraphs based descriptors, and it generates typically lower dimensional feature vectors. A number of methods have been proposed in recent years using cyclic patterns [20], acyclic, path and tree fragments [19] to define the descriptor space and to generate features. With this feature based representation any classification technique can be used for the classification task.

The state of the art classification algorithms includes SVM [7], Adaboost [10-12] and Maximum entropy model. SVM, in a very simple case, chooses a maximum margin linear classifier among all the existing linear classifiers. The margin is defined as the distance of the linear classifier to its nearest training samples, known as support vectors. This idea can be developed to non-linear case by mapping the samples to a high dimension space where they can be classified linearly. This mapping is performed by kernel function which defines the distance between all pairs of samples in the new high dimensional space. Unlike traditional approach to classification, SVM minimizes the empirical classification error and maximizes the geometric margin at the same time and it has been shown that this increases the generalization power of the model (decreases the classification error of unseen examples). SVM are widely applied to classify graph objects. Cai et al. use SVMs to classify protein sequences [13]. Dobson et al. applied SVM to distinguish enzyme from non-enzyme proteins [14]. Much of the recent focus on applying SVM in graph application is on how to build efficient and valid kernel functions on graphs. Most of these approaches are usually based on constructing a feature space with decomposing a graph into subgraphs and counting the number of these subgraphs. Watkins in [15] shows that the scores produced by certain dynamic alignment algorithms can be considered as valid kernel functions. Kashima et al. in [16] uses the counts of labels paths produced by random walks on graph to generate the kernel. Borgwardt et al. in [17] construct the kernel by combining the similarity measures based on different data types for different source of information including sequential, structural and chemical.

Adaboost is a meta-algorithm which can be applied over any given classifier. It constructs a series of weak

classifiers and uses a linear combination of these classifiers as the final model. Adaboost uses an iterative process during which the error rate of the model decreases gradually. Given the model created from the linear combination of the current series of weak classifiers, it makes a new weak classifier with concentrating on the areas with many wrongly labeled samples. This is performed with a sampling procedure from a weighted training sample set with high weight for wrongly labeled samples and low weight for correctly labeled samples.

Maximum entropy has been used widely in different areas such as machine learning and Natural Language Processing (NLP). For example, in machine learning it is used to build translation models, such that for a given word (x), its translation could be any (y) in a set that contains candidate values for (y) [1]. Zhu and Rosenfeld in [2] used maximum entropy models and minimum discriminative information methods to build n -gram language models. Furthermore, Khudanpur Wu in [3] used Maximum entropy to integrate n -grams and topic dependencies in conversational speech recognition. On the other hand, Mannila et al. [4] have applied the maximum entropy principle to synthesize global models from frequent itemsets, and sequential patterns for query selectivity and protein sequence modeling applications. Maximum entropy is also applied in classification. In [5] Nigam et al. illustrated how maximum entropy could be used for text classification.

2.3 Frequent Subgraph Mining

In frequent subgraph mining the goal is to develop algorithms to discover frequently occurring subgraphs in the graph database. Although there are many efficient and scalable frequent pattern mining algorithms exist for itemset mining and sequence mining, developing efficient and scalable algorithms for subgraph mining is particularly challenging as subgraph isomorphism, which is a computationally expensive operation, plays a key role throughout the mining process. Despite that, several subgraph mining algorithms such as FSG [25], and gSpan [26] are available and can be used in many practical situations.

In this paper, we use FSG algorithm [25] to generate frequent subgraph patterns, which are subsequently used to build the prediction model. FSG takes a graph database and a minimum support ξ and generates all connected subgraphs that occur in at least $\xi\%$ of the graphs. FSG follows an *Apriori* style level-by-level approach (breadth first search) to generate subgraph patterns. It starts by enumerating frequent subgraphs consisting of one edge and proceeds to generate larger subgraphs by joining them. At each level sub graphs are

grown by adding one edge at a time. Once a subgraph is generated its occurrence in the graph database is computed to determine whether it is frequent. FSG uses number of optimization techniques to join subgraphs efficiently, and to compute the frequency of the subgraphs. For more information readers should refer to [25].

3. Maximum Entropy Model

Maximum entropy modeling synthesizes a set of local patterns into a global model by choosing a probability model that is consistent with the constraints imposed by the local patterns, but otherwise, is as uniform as possible. Mathematically, the approach corresponds to finding a probability model P that minimizes the Kullback-Leibler divergence $D(P||Q)$, subject to a set of constraints φ , with Q chosen to be a uniform probability distribution. The resulting model can be shown to be equivalent to a model P^* that maximizes the entropy:

$$P^* = \arg \max_P H(P) \quad (1)$$

where $H(P)$ is the entropy of P .

3.1 Formulation

Let $G = \{g_1, g_2, \dots, g_n\}$ be a collection of n graphs in the database and $F = \{X_1, X_2, \dots, X_k\}$ be the set of frequent subgraphs, where each subgraph $X_i \in F$ satisfies the minimum support threshold. Furthermore, we denote s_i as the support for feature X_i in the database G . We also assume that each graph g_i is assigned a class label, y , chosen from a set of discrete labels Y . In this paper, we assume that the classes are binary, even though the approach can be generalized to multi-class problems using one-versus-one, one-versus-all, or error correcting output coding (ECOC) methods.

Given a graph g , let $X_g \subseteq F$ be the set of frequent subgraphs contained in g . Our objective is to find a global probability model $P(y|X_g)$ based on the subgraphs in F and their corresponding support s_i using the maximum entropy principle. To apply the maximum entropy principle, we first define a set of features constructed from the frequent subgraphs. We then create a database of binary transactions, where each transaction corresponds to a graph $g \in G$. Each transaction also contains a set of binary features, also called items, which is defined as follows:

$$f_g(X_i, y) = \begin{cases} 1 & \text{if } X_i \subseteq g \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

As previously noted, the maximum entropy principle seeks to find a probability model P^* that maximizes the following objective function:

$$P^* = \max_P \left[- \sum_{g \in G} P(y | X_g) \log P(y | X_g) \right] \quad (3)$$

subject to the following constraints:

$$\sum_g P(y | X_g) f_g(X_i, y) = s_i \quad (4)$$

where s_i corresponds to the support of the subgraph X_i in the database G . Note that Equation (4) states that the expected value for every feature is constrained to be identical to the support of the corresponding subgraph. It can be shown that the probability model that maximizes Equation (3) subject to the linear constraints in Equation (4) has the following exponential form::

$$P^*(y | X_g) = \frac{1}{Z(X_g)} \exp \left[\sum_i \lambda_i f_g(X_i, y) \right] \quad (5)$$

where $Z(X_g)$ is a normalization factor, and λ 's are the parameters to be optimized.

3.2 Parameter Estimation

Let $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_d\}$ be the set of parameters to be estimated. The Λ vector can be interpreted as the significance or weight of the corresponding features and can be estimated using the maximum likelihood approach. Specifically, the likelihood function for the training data G is:

$$L(\Lambda) = \prod_{g \in G} P^*(y | X) \tilde{p}^{(X,y)} \quad (6)$$

where $\tilde{p}(X, y)$ is the estimated counts of (X, y) in the training data G .

The conditional maximum likelihood model above is solved using the Improved Iterative Scaling algorithm (IIS) [1]. The goal of using IIS is to prevent overfitting in the produced model $P^*(y/X)$, and this is done by using the Gaussian Prior. Here, instead of maximizing the likelihood function we maximize the posteriori function:

$$L(\Lambda) = \prod_{g_i \in G} P^*(y_i | X_i) \tilde{p}^{(y_i/X_i)} \times P(\Lambda) \quad (7)$$

where the parameters Λ are assumed to have a Gaussian prior with zero mean and the same variance σ^2 :

$$P(\Lambda) = \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[- \frac{\lambda_i^2}{2\sigma^2} \right] \quad (8)$$

Solving the log likelihood which has a concave shape for Λ that maximizes the likelihood starts by initializing $\Lambda = \{\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_m\}$ with some arbitrary values. Then it proceeds to find a new set of parameters which yields a model of presumably higher log likelihood.

$$L'(\Lambda) = L(\Lambda + \Delta) - L(\Lambda) \quad (9)$$

Solving the equation above gives us a model in terms of Δ , so taking the derivative with respect to a certain δ_i yields:

$$\sum_{g \in G} \tilde{p}(X, y) f_i(X, y) - \sum_{g \in G} \tilde{p}(X) p^*(y | X) f_i(X, y) \exp(\delta_i f_i^{\#}(X, y)) = 0 \quad (10)$$

where $f_i^{\#}(X, y) = \sum_{i=1}^n f_i(X, y)$ for a particular graph g .

We solve this for each δ_i and increment the corresponding λ_i iteratively. We continue with this search procedure until a stationary point of the vector Λ is found, which causes the model to converge.

4. Algorithm

This section describes our proposed algorithm (*MaxEnt*) based on the above framework for classifying graphs in a given dataset.

In our approach, we first generate frequent subgraphs for a given minimum support threshold using a frequent subgraph mining algorithm. Once the frequent subgraphs are generated binary feature vectors are constructed for each graph in the graph database. The i^{th} entry of the feature vector is set to one if that feature (frequent subgraph) occurs in the graph, and it is set to zero if the feature is not present.

Using these feature vectors, we repeatedly compute the model described in Equation (10) until it converges to a stationary point. The following algorithm describes the procedure we used to classify graphs.

Algorithm MaxEnt

Input: Graph database G , min support threshold ξ Empirical distribution $\tilde{p}(X, y)$ **Output:** Optimal parameters set $(\lambda_i) \forall i \in \{1, 2, \dots, n\}$
Optimal model p^* **Method:**

- 1: generate feature functions f_1, f_2, \dots, f_n for G
 - 2: construct binary feature vector for each $g \in G$
 - 3: initialize $\lambda_i = 0 \forall i \in \{1, 2, \dots, n\}$
 - 4: repeat
 - 5: for $i = 1$ to n do
 - a. compute δ_α by solving Equation (10)
 - b. update $\lambda_i \leftarrow \lambda_i + \delta_i$
 - c. $\forall g \in G$ compute $P^*(y|X)$ using Λ .
 - 6: until convergence
-

5. Experimental Evaluation

In this section we describe the experimental environment used to evaluate our algorithms and the results obtained.

5.1. Datasets

We used 4 different real world datasets to evaluate the algorithms.

The first dataset, *Predictive Toxicology Data -PTC*¹, contains 417 chemical compounds evaluated on 4 types of laboratory animals: male mouse (MM), female mouse (FM), male rat (MR), and female rat (FR). Each compound is assigned a class label indicating the toxicity (positive or negative) of the compound for that animal. Therefore, in this dataset we have 4 binary classification problems corresponding to each laboratory animal (MM/FM/MR/FR).

The second dataset, *Aids*², contains 42,682 chemical compounds evaluated for evidence of anti HIV activity. We have formulated a binary classification problem, where a compound is labeled as active if it can provide protection to the human CEM cells from HIV infection, and as inactive otherwise.

The third dataset, *Cancer*³, contains 42,247 chemical compounds evaluated for evidence of the ability to inhibit the growth of human tumor cell lines. We have formulated a binary classification problem using this dataset.

The fourth dataset is the *WebSpam*⁴ dataset containing 4,188 graphs generated for our purpose. This dataset

contains a web graph of nodes corresponds to hostnames, and edges corresponds to the directed links between hosts. We first discretized this data set by calculating the average number of links between the nodes and applying a threshold to form new edges. Then for each node its neighborhood is determined, and each such node and its corresponding neighborhood are used to form a single graph. This way we were able to generate multiple graphs and each graph is classified as normal or spam.

Table 1 shows various characteristics of these datasets.

Table 1. Characteristics of the datasets

Dataset	Num. Instances	Min. Support
MM	417	10%
MR	417	10%
FM	417	10%
FR	417	10%
Aids	42,682	20%
Cancer	42,247	10%
WebSpam	4,188	25%

5.2. Experimental Methodology

In order to generate feature vectors, we used a frequent sub graph mining algorithm – FSG [25] and generated frequent subgraphs for a given minimum support threshold. Table 1 shows the selected minimum support threshold for each of the datasets. Then binary feature vectors are constructed for each graph in the graph database as discussed in Section 4. Once the feature vectors are built any of the existing classification algorithms (SVM, Adaboost) can potentially be used for classification.

In this paper, classification is done by performing 5-fold cross validation on the dataset; that is, we divide the datasets into five equal sized subsets and in each round we choose one subset as the test set and remaining four subsets as the training set and report the average accuracy.

5.3. Results

In this section we describe the performance of our approach when compared with the existing methods. We analyze the performance using two different descriptor spaces: frequent subgraphs, and maximal frequent subgraphs.

5.3.1 Comparison with Other Approaches

We compared the performance of our algorithm (*MaxEnt*) against two existing methods: *Adaboost* and

¹ <http://www.predictive-toxicology.org/ptc/>² http://dtp.nci.nih.gov/docs/aids/aids_data.html³ http://dtp.nci.nih.gov/docs/cancer/cancer_data.html⁴ <http://www.yr-bcn.es/webspam/datasets/>

SVM. For the SVM classifier, we used SVM^{Light} package provided in [8] and described in [7]. For Adaboost we used the MATLAB package provided in [9]. We set the number of iteration for Adaboost as 20. Table 2 shows the accuracy (Acc) and F-Measure (F) of each of these methods.

Table 2. Experimental results (Accuracy and F-Measure)

Dataset	Adaboost		SVM		MaxEnt	
	Acc	F	Acc	F	Acc	F
MM	0.584	0.58	0.561	0.50	0.582	0.59
FM	0.567	0.56	0.547	0.46	0.558	0.57
MR	0.585	0.53	0.569	0.42	0.586	0.63
FR	0.558	0.61	0.565	0.64	0.578	0.54
Cancer	0.654	0.59	0.669	0.60	0.661	0.58
Aids	0.974	0.00	0.964	0.00	0.965	0.07
WebSpam	0.935	0.68	0.906	0.00	0.934	0.67

As can be seen from the results, *MaxEnt* approach achieved a reasonably better accuracy compared to other approaches. In MR, and FR datasets it shows the best performances. Overall it achieved better performance than SVM in all of the datasets except in *Cancer* data. In the *WebSpam* dataset, performance of *MaxEnt* is better than that of SVM and is comparable to *Adaboost*. *Adaboost* shows better performance in MM, FM, and *Aids* datasets when compared with *MaxEnt*.

5.3.2 Classification Using Maximal Subgraph-Based Descriptors

In the previous section we used frequent subgraph based descriptor space for defining feature vectors. Instead of frequent subgraphs here we analyze maximal frequent subgraphs as descriptors. For this analysis, we used 4 datasets (MM, FM, MR, FR) and generated maximal frequent subgraphs with the same support threshold (10%) used in the previous case. Table 3 shows the classification accuracy and F-Measure for each of the algorithms.

Table 3. Experimental results (Accuracy and F-Measure) with maximal subgraphs

Dataset	Adaboost		SVM		MaxEnt	
	Acc	F	Acc	F	Acc	F
MM	0.577	0.58	0.602	0.61	0.565	0.55
FM	0.553	0.54	0.547	0.49	0.542	0.54
MR	0.568	0.51	0.561	0.38	0.554	0.57
FR	0.603	0.63	0.592	0.68	0.563	0.50

Use of maximal subgraph based descriptors changes the performance of different algorithms dramatically. However while it improves the performance of SVM in terms of both accuracy and F-Measure, it reduces the performance of Adaboost and *MaxEnt*.

As the number of frequent subgraphs generated for these datasets is always higher than the number of maximal frequent subgraphs, these extra features provided by frequent subgraphs can be very effective for classification. To understand the values of extra features in frequent subgraphs, we measured the matching percentage of the first n features used by *MaxEnt* methods with maximal subgraphs, i.e., we find n features which obtained the highest weight in the *MaxEnt* algorithms and counted how many of them were maximal. We changed n from 10 to the number of maximal subgraphs and plotted the matching percentage. Figure 1 and 2 show this plot for MM and FR⁵ datasets. As can be seen in these plots, the matching percentage falls quickly when n increases which shows that *MaxEnt* uses non-maximal features as some of the most effective features for the purpose of classification. Therefore, *MaxEnt* approach can automatically choose the best features for classification and thus it shows better performance with frequent subgraphs based descriptors than the maximal ones.

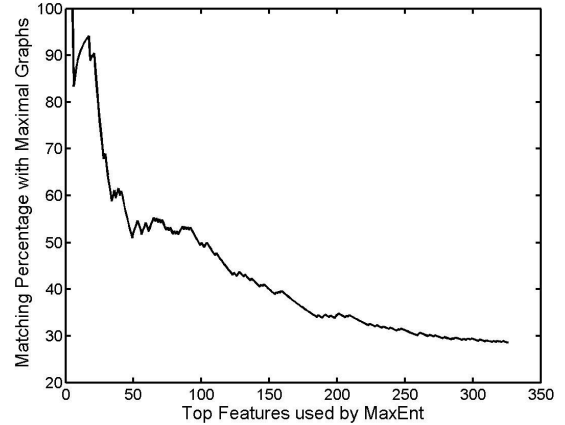


Figure 1. The matching percentage of top frequent subgraphs features for *MaxEnt* with Maximal subgraphs features in MM dataset.

6. Conclusions

This paper investigated maximum entropy based approach for the problem of graph classification. Similar to some of the existing methods, our algorithm is based

⁵ Other datasets exhibit similar characteristics.

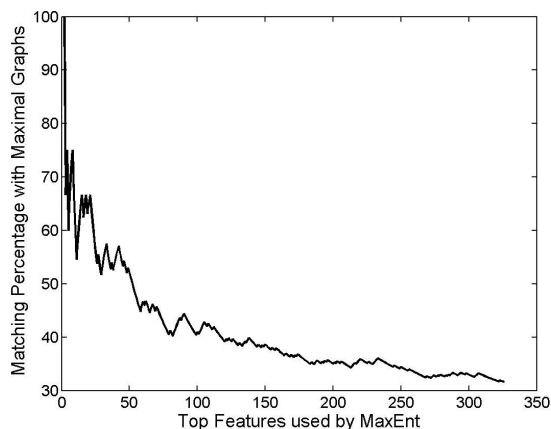


Figure 2. The matching percentage of top frequent subgraphs features for *MaxEnt* with Maximal subgraphs features in FR dataset.

on the frequent subgraph patterns extracted from the graph database. Instead of using these patterns directly to build feature vectors, we combined them into a coherent global model that can be used for prediction. This method, which is relatively unexplored in the context of graphs, has the advantage of providing better accuracy and efficiency. Also, it offers coherent and consistent way for predicting the class of the graph objects.

Experimental results using real-world data sets confirmed that this approach is generally more effective at classifying most understandable graph objects. Also, the results revealed that the precision of our approach does not depend heavily on the support threshold used to generate frequent subgraph patterns, which makes our approach stable, compared to existing SVM based graph classifiers.

In our past work [23], we have explored data classification in an unsupervised setting. In the future we plan to investigate unsupervised learning techniques for graph classification. This will allow us to look for patterns that have not previously been considered, which is a key challenge in anomaly detection.

Acknowledgements

We would like to thank Dr. George Karypis for providing the *PAFI* software containing *FSG* graph mining algorithm.

References

[1] A.L. Berger, V.J. Della Pietra, and S.A. Della Pietra. Maximum Entropy Approach for Natural Language

Processing. *Computational Linguistics*.22(1) P: 39 – 71, 1996.

- [2] X. Zhu and R. Rosenfeld. Improving Trigram Language Modeling with the World Wide Web. *In Proc of ICASSP*, P:533–536, 2001.
- [3] S. Khudanpur, and J. Wu. A Maximum Entropy Language Model Integrating N- Grams and Topic Dependencies for Conversational Speech Recognition. *In Proc of ICASSP*, 1999.
- [4] H. Mannila, and P. Smyth. Prediction with Local Patterns using Cross-Entropy. *In Proc. of the fifth ACM SIGKDD*, P: 357 – 361, ISBN:1-58113-143-7, 1999.
- [5] K. Nigam, J. Lafferty, and A. Maccallum. Using Maximum Entropy for Text Classification. *IJCAI-99 Workshop on Machine Learning for Information*, 1999.
- [6] A. Berger. The Improved Iterative Scaling Algorithm: A gentle Introduction. Technical report, Carnegie Mellon University, 1997.
- [7] T. Joachims. *Making large-Scale SVM Learning Practical*. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT Press, 1999.
- [8] <http://svmlight.joachims.org/>
- [9] <http://research.graphicon.ru/machine-learning/gml-adaboost-matlab-toolbox.html>
- [10] R.E. Schapire and Y. Singer Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297-336, December 1999.
- [11] J. Friedman, T. Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, April 2000.
- [12] A. Vezhnevets, and V. Vezhnevets. Modest AdaBoost - Teaching AdaBoost to Generalize Better. *In Proc of Graphicon*, Novosibirsk Akademgorodok, Russia, 2005.
- [13] C.Z Cai., W.L. Wang, L.Z. Sun and Y.Z. Chen. Protein function classification via support vector machine approach. *Math. Biosci.*, 185, 111–122, 2003.
- [14] P.D. Dobson and A.J. Doig, Distinguishing enzyme structures from non-enzymes without alignments. *J. Mol. Biol.*, 330, 771–783, 2003.
- [15] C. Watkins. Dynamic Alignment Kernels, Department of Computer Science, Royal Holloway, University of London, Technical Report, CSD-TR-98-11, January 1999.
- [16] H. Kashima, K. Tsuda, and A. Inokuchi, Marginalized kernels between labeled graphs. *In Proceedings of 20th International Conference on Machine Learning (ICML 2003)*, Washington, DC, 2003.
- [17] K.M. Borgwardt, C.S. Ong, S. Schö'ouner, S. Vishwanathan, A.J. Smola, and H.P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics* 21(1), 2005.
- [18] M. Deshpande, M. Kuramochi, Nikil Wale, and G. Karypis, Frequent Substructure-Based Approaches for Classifying Chemical Compounds, *IEEE Transactions on*

- Knowledge and Data Engineering*, vol. 17, no. 8, pp. 1036-1050, Aug., 2005.
- [19] N. Wale and G. Karypis. Acyclic Subgraph-based Descriptor Spaces for Chemical Compound Retrieval and Classification. *In Proc of IEEE International Conference on Data Mining (ICDM)*, 2006.
- [20] T. Horvath, T. Grtner, and S. Wrobel. Cyclic pattern kernels for predictive graph mining. *In Proc. of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 158-167, 2004.
- [21] T. Kudo, E. Maeda, and Y. Matsumoto. An Application of Boosting to Graph Classification, *NIPS* 2004.
- [22] C. Liu, X. Yan, H. Yu, J. Han, and P. S. Yu. Mining Behavior Graphs for Backtrace of Noncrashing Bugs. *In Proc. of SIAM Int. Conf. on Data Mining (SDM'05)*, 2005.
- [23] H.D.K. Moonesinghe and Pang-Ning Tan. Outlier Detection using Random Walk. *In Proc. of IEEE International Conference on Tools with Artificial Intelligence (ICTAI-06)*, Washington D.C., 2006.
- [24] M.R. Berthold and C. Borgelt. Mining Molecular Fragments: Finding Relevant Substructures of Molecules, *In Proc. Int'l Conf. on Data Mining*, 2002.
- [25] M. Kuramochi and G. Karypis. Frequent subgraph discovery. *In Proc of IEEE International Conference on Data Mining (ICDM)*, 2001.
- [26] X. Yan, and J. Han. gSpan: Graph-based substructure pattern mining. *In Proc. of Int'l Conf. on Data Mining (ICDM)*, 2002.