Multiple Instance Learning with Genetic Programming for Web Mining

A. Zafra[1], S. Ventura[2], E. Herrera-Viedma[1], and C. Romero[2]

[1] Department of Computer Science and Artificial Intelligence. University of Granada
[2] Department of Computer Science and Numerical Analysis. University of Córdoba

Abstract. The aim of this paper is to present a new tool of multiple instance learning which is designed using a grammar based genetic programming (GGP) algorithm. We study its application in Web Mining framework to identify web pages interesting for the users. This new tool called GGP-MI algorithm is evaluated and compared with other available algorithms which extend a well-known neighborhood based algorithm (k-nearest neighbour algorithm) to multiple instance learning. Computational experiments show that, the GGP-MI algorithm obtains competitive results, solves problems of other algorithms, such as sparsity and scalability and adds comprehensibility and clarity in the knowledge discovery process.

1 Introduction

Multiple instance learning, or multi-instance learning (MIL) introduced by Dietterich et al. [1] is a recent learning framework which has attracted interest in the machine learning community. In this new learning the training set is composed of many bags where each one contains many instances. A bag is positively labeled if it contains at least one positive instance. Otherwise it is labeled as a negative bag. The task is to learn some concepts from the training set for correctly labeling unseen bags.

Since MIL appearance, a wide variety of tasks have been formulated as multiple-instance learning problems. Among these tasks, we can find Web Index Recommendation. This problem has been resolved both from a traditional perspective with several techniques such as k-nearest neighbour [2] and inverse document frequencies [3], and from multiple-instace perspective adapting k-nearest neighbour algorithm [4]. However, we should point out that nearest neighbor algorithms present two major limitations. The first one is related to sparsity and to scalability. That is, they become hard to scale them to a large number of items, maintaining reasonable prediction performance and accuracy. The problem is that they require computations that grow linearly with the number of items. The second one is related to interpretability of discovered knwoledge. That is, they are as black box, where no information about the user preferences are shown.

In this paper, to overcome the drawbacks aforementioned we propose a new tool of MIL defined using grammar based genetic programming (GGP) which allows to discover user preferences in web page recommendation tasks. This new tool, called GGP-MI algorithm generates a simple rule based classifier that increases generalization ability, includes interpretability and clarity in the discovery knowledge providing information about user's interest and classifies new examples (web pages) quickly. Experimental results for solving this problem show that this approach obtains competitive results in terms of accuracy, recall and precision.

The rest of this paper is organized as follows. Section 2 reviews previous works about MIL. Section 3 presents Web Index Recommendation problem. Section 4 describes the proposed GGP-MI algorithm. Section 5 reports experimental results. Finally, section 6 presents the conclusions and future works.

2 Multi-instance Learning

In the middle of 1990's, Dietterich et al. [1] conceived MIL term when they investigated the problem of drug activity prediction. For solving this problem they described three Axis-Parallel Rectangle (abbreviated as APR) algorithms. Following this work, it has appeared a great number of new methods of MIL. Auer [5] focused on theoretical research and presented the MULTINST algorithm. Maron et al. [6] propused one of the most famous multi-instance learning algorithms, Diverse Density (DD). This algorithm was combined with Expectation Maximization (EM) algorithm, appearing EM-DD [7]. Long and Tan [8] showed from theoretical aspect that it is possible to PAC-learn an axis-parallel concept from multi-instance examples.

The first approaches using lazy learning, decision trees and rule learning were investigated during 2000 year. In lazy learning context, Whang and Zucker [9] proposed two variants of the k nearest neighbour algorithm (KNN) that they referred as Citation-KNN and Bayesian-KNN. With respect to decision trees and learning rules, Zucker and Chevaleyre [10] implemented ID3-MI and RIPPER-MI. At same time, Ruffo [11] presented a multi-instance version of C4.5 decision tree, which was named as RELIC. Later, Zhou et al. [4] presented Fretcit-KNN algorithm, which is a variant of Citation-KNN.

There are also many other practical multiple instance (MI) algorithms, such as extension of standard neural networks to MIL [12,13]. Also there are proposals about adapting Support Vector Machines to multi-instance framework [14,7,15] and the use of ensembles to learn multiple instance concept [12].

3 Web Index Recommendation as Multiple Instance Problem

There are a great number of web index pages on the Internet, these pages contain references or brief summaries of other pages. This problem tries to analyze automatically them and to show to the user only the pages which contain issues interesting for him or her. To do that, it is neccesary to identify the users' interests and decide on if a new web index page will interest the user or not. This problem is called Web Index Recommendation [4], which is a specific web usage mining task. Here the difficulty lies in that the user only specifies whether he or she is interested in an index page, instead of specifying the concrete links that he or she is really interested in.

This problem could be viewed as a multi-instance problem where the goal is to label unseen web index pages as positive or negative. A positive web index page is such a page that the user is interested in at least one of its linked pages. A negative web index page is such a page that none of its linked pages interested the user. In this problem, each web index page will be considered as a bag while their linked pages will be considered as instances in the bag. Each instance can be represented by a term vector $T = [t_1, t_2, , t_n]$, where t_i $(i = 1, 2, .., n)$ is one of the n most frequent terms appearing in the corresponding linked page. All the pages are described by the same number of frequent terms, although their components may be quite different. Also, for different bags, their corresponding web index pages may contain different number of links, that is, the number of instances in the bags may be different.

4 Multiple-Instance Genetic Programming

Genetic programming (GP) introduced by John Koza [16] is becoming in a paradigm with a growing interest on classification task. We can find different proposals which show that GP is a mature field that can efficiently achieve low error rates on supervised learning [17,18], therefore it would be interesting to adapt it to MIL and check its performance.

The next sections describe our grammar based GP system which has been implemented in JCLEC framework [19].

4.1 Individual Representation

An individual consists of two components, *a genotype* which is encoded as tree structures with limitations on the tree depth to avoid size too large. A *phenotype* which represents the full rule with antecedent and consequent. The antecedent consists of tree structure and represents a rule which can contain multiple antecedents attached by conjunction or disjunction and the consequent specifies the class for the instance that satisfies all conditions of antecedent. We consider that all individuals classify the positive class and all examples that do not satisfy the individuals rule set are implicitly classified as belonging to negative class.

We use a grammar to enforce syntactic constrains and satisfy the closure property (see Fig(1)). This grammar mantains syntactical and semantic constraints both the generation of individuals in the initial population and the production of new individuals via crossover.

4.2 Genetic Operators

The elements of the next population are generated by means of three operators: crossover, copy and elitism.

```
antecedent ->      comparison
                 | OR comparison antecedent
                 | AND comparison antecedent

comparison ->    comparator valuesToCompare

comparator ->    <
               | ≥

valuesToCompare ->  attribute  value
```

(a) Using rank of terms

```
antecedent ->      comparison
                 | OR comparison antecedent
                 | AND comparison antecedent

comparison ->    comparator valuesToCompare

comparator ->     CONTAIN
                | NOT_CONTAIN

valuesToCompare ->  attribute
```

(b) No using rank of terms

Fig. 1. Grammars used for individual representation

Crossover. The crossover is performed by swapping the subtrees of two parents between two compatible points randomly selected in each parent. Two tree nodes are compatible if their operators can be swapped without producing an invalid individual according to defined grammar. If any of the two offsprings is an invalid individual, due to a violation of the condition of uniqueness for each attribute in each rule antecedent or a too large tree, they will be replace by one of their parents.

Copy. The copy operator simply chooses an individual in the current population and copies it without changes into the new population.

Elitism. Elitism operator copies the best individual of a generation and it is passed unchanged to the next generation, to prevent the stochastic process of evolution from losing that individual.

4.3 Fitness Function

The fitness function evaluates the quality of each individual according to three basic criteria: accuracy, recall and precision. We comment some necessary concepts to undestand before measures, P_a positive bags which are recommended, P_r positive bags which are rejected, N_a negative bags which are recommended and N_r negative bags which are rejected. Being P the number of positive bags and N negative bags, $P = P_a + P_r$ and $N = N_a + N_r$. The predictive accuracy, recall and precision are defined in (1).

$$accuracy = \frac{P_a + N_r}{P + N}, \quad recall = \frac{P_a}{P}, \quad precision = \frac{P_a}{P_a + N_a} \tag{1}$$

Our fitness function is defined as the product of these three indicators, see (2). The goal is to maximize them at same time. These measures are relationed, thus if we maximize the value of any of them, the value of others can be significantly reduce.

$$fitness = Accuracy * Recall * Precision \tag{2}$$

5 Experiments and Results

We carry out two types of experiments. A first experiment compares performance of our GGP-MI algorithm using different configurations on the Web Index Recommendation problem. All experiments are repeated five times with different seeds, and the average values of accuracy, recall and precision are reported in the next sections. The second experiment evaluates our best performed algorithm against other classification techniques.

5.1 Data Sets and Running Parameters

Experiments have been done about nine data sets, in each one of them one different volunteer labelled 113 web index pages according to his/her interests. For each data set, 75 web index pages are randomly selected as training bags while the remaining 38 index pages are used as test bags. Although we follow exactly the same setup as [4], data sets are adapted for working with our algorithm, thus it is established as a vector where each element of vector is the frecuency of each word considered on training set and is added to each instance two attributes, the first one named bag_{id} identifying the bag it belongs to and the second one named bag_{class} encoding the class of its bag.

All comparisons of results is carried out by association of data sets. Data sets are organized in three groups, the first group consists of V1, V2 and V3 data sets that contain more negative bags than positives bags (this group is referenced as majority negative sets). The second group consists of V4, V5 and V6 data sets that contain more positive bags than negatives bags (this group is referenced as majority positive sets) and the third group consists of V7, V8 and V9 data sets that contain same number of positive bags than negative bags (this group is referenced as balanced sets). This association it is interesting to notice and study the behaviour of algorithms between balanced and unbalanced data sets.

The parameters used in all GP runs were: population size: 1000, generations: 100, crossover probability: 95%, reproduction probability: 5%, selection method for both parents: fitness proportional (roulette wheel), maximum tree depth: 15. The initial population was generated using ramped-half-and-half method [16].

5.2 Comparison of GGP-MI

Several experiments are performed to evaluate the performance of our GGP-MI algorithm on the Web Index Recommendation problem. Thus, we do two comparisons, the first one is between data sets that consider all words (referenced as *all words*) and data sets that consider words that appear in more of 3 bags and less of 40 bags (referenced as *less words*), this case tries to eliminate both very exclusive words and very usual words. These words are not useful in classification tasks and increase the search space. The second one is between classifiers that have into account frecuencies of words and classifiers that only have into account if words appear or do not appear. In Table 1 is reported a comparative summary between the average values obtained.

Table 1. Summary results with GGP-MI

	Majority negative sets			Majority positive sets			Balanced sets		
Algorithm	Acc	Rec	Prec	Acc	Rec	Prec	Acc	Rec	Prec
GGP-MI All Words	0.842	0.468	0.369	0.807	1.000	0.801	0.693	0.927	0.621
GGP-MI Less Words	0.807	0.690	0.379	0.825	0.877	0.895	0.711	0.750	0.727
GGP-MI All Words[1]	0.842	0.559	0.660	0.816	1.000	0.809	0.474	1.000	0.474
GGP-MI Less Words[1]	0.886	0.821	0.518	0.825	1.000	0.816	0.526	1.000	0.500

We can see that in the case of using a data set with a reduced number of words, moreover of reducing the search space and to require less computation time, the results are improved. Precision and accuracy values appear a increased with respect to use *all words*, but recall values are a bit decreased. This behaviour may be explained because the generated classifier is used to classify positive bags and the bags which do not cover this classifier, are considered negative bags. For this, if we eliminate common and specific words, we favour that certain positive examples are not included because we do not use common words (decrease recall), but we make less mistakes, because without common words, it does that negatives bags are not considered positive bags (increase precision).

If we compare between methods which consider rank information of the frequent terms and do not, we can see that the first ones obtain better values of recall, but worse values of precision. This can be explained if we study the number of accuracy and error produced with respect to each class. An increment of 'recall' means classifying correctly more positive class bags, this involves that rule is more specific for this class. Thus, an increase of positive true (positive cases correctly identified) is associated with an increase of false positive (cases identified as positive but they are negative), and a decrement of true negative (negative cases correctly identified). Therefore, any increase in recall used to go accompanied by a decrease in precision.

5.3 Comparison to Other Algorithms

Comparisons are made with Frecit-KNN, Citation-KNN and Txt-KNN. They are describe in [4]. Txt-KNN is obtained through adapting the standard KNN algorithm to textual objects. Citation-KNN is similar to before algorithm but considers both the references and the citers of an unseen object in prediction. Frecit-KNN is similar to Citation-KNN, but is a multi-instance learning algorithm while the former is a single-instance learning algorithm. After analyzing our algorithm in before section, we choose the configuration which uses *less words* to compare with these techniques. Table 2 shows a summary with average values obtained by different algorithms for each association of data sets.

[1] Using frecuency of words.

Table 2. Summary results with all words

	Majority negative sets			Majority positive sets			Balanced sets		
Algorithm	Acc	Rec	Prec	Acc	Rec	Prec	Acc	Rec	Prec
Fretcit-kNN	0.879	0.622	0.514	0.854	0.925	0.896	0.698	0.603	0.718
Txt-KNN	0.795	0.652	0.335	0.805	0.985	0.809	0.570	0.722	0.538
Citation-KNN	0.803	0.433	0.336	0.796	0.864	0.875	0.674	0.561	0.701
GGP-MI Less Words	0.807	0.690	0.379	0.825	0.877	0.895	0.711	0.750	0.727
Fretcit-kNN [2]	0.870	0.627	0.486	0.810	0.915	0.856	0.732	0.605	0.802
Txt-KNN [2]	0.795	0.533	0.326	0.812	0.967	0.821	0.600	0.741	0.562
Citation-KNN [2]	0.833	0.456	0.429	0.782	0.852	0.858	0.674	0.594	0.695
GGP-MI Less Words [2]	0.886	0.821	0.518	0.825	1.000	0.816	0.526	1.000	0.500

If we compare all methods, we can see that our GGP-MI obtains better results than Txt-KNN and Citation-KNN. With respect to Frecit-KNN, in case of not considering frequencies of words, our algorithm obtains slightly better results for balanced data sets and although obtains slightly worse results for unbalanced data sets, in our opinion, it is counterbalanced by the availability of generating intelligible rules that once interpreted provide an interesting information for knowing user's interest. Following, one classifiers extracted from V1 data set is shown. We can get representative information about the user's interest to know his/her preferences.

```
IF ( (contains planet AND contains forecast) OR
     (contains atmospheric) OR (not_contains football) )
THEN Recommend page to V1 user.
ELSE No recommend page to V1 user.
```

If we see the classifier generated for V1 user, we can learn what topics can be recommeded to user. Thus, we can discovery that user is interested on topics such as ecology and environment (with words as planet, forecast and atmospheric) and is not interested in football.

Finally, we conclude that there are two main motivations for using GGP-MI:

1. It is not necessary to establish number of terms of each instance considered in run of algorithm. As we known, the k-nearest algorithm is an expensive technique which requires computations that grow linearly with the number of terms. Moreover, an increase in number of terms considered used to go accompanied by an increment in results.

[2] Using frecuency of words.

2. The available algorithms for solving this problem do not generate interpretable hypotheses such as rule set. Rule based system, as our algorithm, generates simple, intuitive and modular knowledge. In addition, the rule set induced by GGP-MI is very short (it only contains about five or six literals), being easily interpretable.

6 Conclusions

This paper describes the first attempt of applying GGP techniques for MIL. It is compared with three main techniques used for solving the Web Index Recommendation problem, Txt-KNN, Citation-KNN and Frecit-KNN. GGP-MI obtains competitive results in terms of accuracy, recall and precision, its results are significantly better than two techniques and the other technique is lightly worse than ours for balanced data sets and lightly better than ours for unbalanced data sets. However, the most relevant characteristic of GGP-MI is that adds the benefits of interpretability and clarity in the knowledge discovery process providing information about users' preferences, whereas the other algorithms are not able to generate easily interpretable knowlegde.

The Web Index Recommendation is a hard problem for algorithms such as GP because it has a large number attributes, which causes some non predictive hypotheses to be consistent with the training data because the search space is too wide. Currently, we are investigating feature selection algorithms based on prototype selection techniques to overcome this problem and thus to decrease search space and to improve the results.

Acknowledgment

This work has been financed in part by the TIN2005-08386-C05-02 project of the Spanish Inter-Ministerial Commission of Science and Technology (CICYT), FEDER funds and Research Project SAINFOWEB P05-TIC-602.

References

1. Dietterich, T.G., Lathrop, R.H., Lozano-Perez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artifical Intelligence 89(1-2), 31–71 (1997)
2. Shakhnarovich, G., Darrell, T., Indyk, P.: Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing). The MIT Press, Cambridge (2006)
3. Joachims, T.: A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In: ICML'97: Proceedings of 14th International Conference on Machine Learning, Nashville, US, pp. 143–151. Morgan Kaufmann Publishers, San Francisco (1997)
4. Zhou, Z.H., Jiang, K., Li, M.: Multi-instance learning based web mining. Applied Intelligence 22(2), 135–147 (2005)

5. Auer, P.: On learning from multi-instance examples: Empirical evaluation of a theoretical approach. In: ICML'97: Proceedings of the Fourteenth International Conference on Machine Learning, San Francisco, CA, USA, pp. 21–29. Morgan Kaufmann, San Francisco (1997)
6. Maron, O., Lozano-Pérez, T.: A framework for multiple-instance learning. In: NIPS'97: Proceedings of Neural Information Processing System 10, pp. 570–576. MIT Press, Cambridge, MA, USA (1997)
7. Zhang, Q., Goldman, S.: EM-DD: An improved multiple-instance learning technique. In: NIPS'01: Proceedings of Neural Information Processing System 14 (2001)
8. Long, P.M., Tan, L.: PAC learning axis-aligned rectangles with respect to product distributions from multiple-instance examples. Machine Learning 30(1), 7–21 (1998)
9. Wang, J., Zucker, J.D.: Solving the multiple-instance problem: A lazy learning approach. In: ICML'00: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 1119–1126. Kaufmann Publishers Inc, San Francisco, CA, USA (2000)
10. Zucker, J.D., Chevaleyre, Y.: Solving multiple-instance and multiple-part learning problems with decision trees and decision rules. Application to the mutagenesis problem. In: Proceedings of the 14th Canadian Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence, Ottawa, Canada, pp. 204–214 (2000)
11. Ruffo, G.: Learning single and multiple instance decision tree for computer security applications. PhD thesis, Department of Computer Science. University of Turin, Torino, Italy (2000)
12. Zhang, M.L., Zhou, Z.H.: Ensembles of multi-instance neural networks. In: Intelligent information processing II. vol. 163, pp. 471–474. London, UK, Springer Boston (2005)
13. Zhang, M.L., Zhou, Z.H.: Adapting rbf neural networks to multi-instance learning. Neural Processing Letters 23(1), 1–26 (2006)
14. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: NIPS'02: Proceedings of Neural Information Processing System, pp. 561–568 (2002)
15. Tao, Q., Scott, S., Vinodchandran, N.V., Osugi, T.T.: SVM-based generalized multiple-instance learning via approximate box counting. In: ICML'04: Proceedings of the twenty-first international conference on Machine learning, New York, NY, USA, pp. 799–806. ACM Press, New York (2004)
16. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
17. Muni, D.P., Pal, N.R., Das, J.: A novel approach to design classifiers using genetic programming. IEEE Trans. Evolutionary Computation 8(2), 183–196 (2004)
18. Zhang, M., Smart, W.: Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification. Pattern Recognition Letters 27(11), 1266–1274 (2006)
19. Ventura, S., Romero, C., Zafra, A., Delgado, J.A., Hervás, C.: JCLEC: A java framework for evolutionary computation soft computing. Soft Computing (In Press) (2007)