Aggregating Multiple Instances in Relational Database Using Semi-Supervised Genetic Algorithm-based Clustering Technique

Rayner Alfred[1,2] and Dimitar Kazakov[1]
[1] York University, Computer Science Department, Heslington
YO105DD York, England
[2] On Study Leave from Universiti Malaysia Sabah,
School of Engineering and Information Technology,
88999, Kota Kinabalu, Sabah, Malaysia
{ralfred, kazakov}@cs.york.ac.uk

Abstract. In solving the classification problem in relational data mining, traditional methods, for example, the C4.5 and its variants, usually require data transformations from datasets stored in multiple tables into a single table. Unfortunately, we may loss some information when we join tables with a high degree of *one-to-many* association. Therefore, data transformation becomes a tedious trial-and-error work and the classification result is often not very promising especially when the number of tables and the degree of *one-to-many* association are large. In this paper, we propose a genetic semi-supervised clustering technique as a means of aggregating data in multiple tables for the classification problem in relational database. This algorithm is suitable for classification of datasets with a high degree of *one-to-many* associations. It can be used in two ways. One is user-controlled clustering, where the user may control the result of clustering by varying the compactness of the spherical cluster. The other is automatic clustering, where a non-overlap clustering strategy is applied. In this paper, we use the latter method to dynamically cluster multiple instances, as a means of aggregating them, and illustrate the effectiveness of this method using the semi-supervised genetic algorithm-based clustering technique.

Keywords: Aggregation, Clustering, Semi-supervised clustering, Genetic Algorithm, Relational data Mining.

1 Introduction

Relational database requires effective and efficient ways of extracting pattern based on content stored in multiple tables. In this process, significant features must be extracted from datasets stored in multiple tables with *one-to-many* relationships format. In relational database, a record stored in a target table can be associated to one or more records stored in another table due to the *one-to-many* association constraint. Traditional data mining tools require data in relational databases to be transformed into an attribute-value format by joining multiple tables. However, with the large

Y. Ioannidis, B. Novikov and B. Rachev (Eds.): Local Proceedings of ADBIS 2007, pp. 136-147

volume of relational data with a high degree of *one-to-many* associations, this process is not efficient as the joined table can be too large to be processed and we may lose some information during the join operation. In fact, most data mining tools ignore the *multiple-instance* problem and treated all of the positive instances as positive examples. For example, suppose we have a *multi-instance* problem as shown in Fig. 1.

ID	F1	F2	F3	Class
1	A1	B2	C4	A
1	A1	B2	C3	A
1	A1	B2	C3	A
1	A1	B2	C3	A
1	A1	B1	C3	A
2	A2	B1	C4	B
2	A2	B1	C5	B
2	A2	B2	C5	B
2	A2	B1	C5	B
3	A1	B2	C3	A
3	A2	B2	C3	A
3	A1	B2	C2	A
3	A1	B1	C2	A

Fig. 1. Multi-Instances Problem

Using a traditional data mining tool, all rows in Fig. 1 are treated as positive and negative examples. The *multiple-instance* problem is ignored. The *multiple-instance* problem arises due to the fact that, relational databases are designed to handle records with *one-to-many* association. The 10-fold cross-validation accuracy of the J48 classifier, WEKA [20], for the above sample data is 92% with extracted rules as follows; A1 → A and A2 → B. On the other hand, one may also cluster these multiple instances that exist for an individual object and get higher percentage of accuracy in predicting the class membership of unseen bag of instances of an individual object.

The most common pattern extracted from relational database is *association* rules [3,4,5,6]. However, to extract *classification* rules from relational database with more effectively and efficiently, taking into consideration of *multiple-instance* problem, we need to aggregate these multiple instances. In this paper, we use genetic algorithm-based clustering technique to aggregate multiple instances of a single record in relational database as a means of data reduction. Before a clustering technique can be applied, we transform the data to a suitable form.

In relational database, a record stored in the target table is often associated with one or more records stored in another table. We can treat these multiple instances of a record, stored in a non-target table, as a bag of terms. There are a few ways of transforming these multiple instances into bag of terms, which will be described and explained in section 2. Once we have transformed the data representation applicable to clustering operations [10,11,12,21], we can use any clustering techniques to aggregate these multiple instances.

In this paper, we use semi-supervised genetic algorithm-based clustering technique to improve the aggregation of multiple instances in relational database. In section 2, we describe the DARA [7,8] transformation process converting these multiple instances into a bag of terms, and, effectively, a vector space model, before any clustering operations can be used to aggregate them. Section 3 describes genetic algorithm-based clustering technique. Experimental results comparing varieties of cluster algorithms, as a means of aggregating multiple instances, that include the

semi-supervised GA-clustering DARA-based algorithm (SSGA-DARA), GA-clustering DARA-based algorithm (GA-DARA) and the k-means DARA-based clustering algorithm (K-DARA) are provided in section 4. In addition to that, we also compare the result of performance accuracy of SSGA-DARA with other previously published results from propositional algorithms, in section 4. Finally, the paper is concluded in Section 5.

2 Data Transformation

In relational database, records are stored separately in different tables and they are associated through the match of primary and foreign keys. When a high degree of *one-to-many* associations is present, a single record, *R*, stored in a main table can be associated with a large volume of records stored in another table, as shown in Fig. 2.

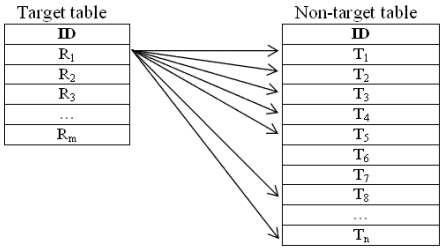

Fig. 2. A o*ne-to-many* association between target and non-target relations

Let *R* denote a set of *m* records stored in the target table and let *S* denote a set of *n* records (T_1, T_2, T_3, … , T_n), stored in the non-target table. Let S_i is in the subset of S, $S_i \in S$, and associated with a single record R_a stored in the target table, $R_a \in R$. Thus, the association of these records can be described as $R_a \rightarrow S_i$. Since a record can be characterized by the bag of term/records that are associated with it, we use the vector space model to cluster these records, as described in the work of Salton *et al.* [9]. In a vector space model, a record is represented as a vector or 'bag of terms', i.e., by the terms it contains and their frequency, regardless of their order. These terms are encoded and represent instances stored in the non-target table referred by a record stored in the target table. The non-target table may have a single attribute or multiple attributes and the process of encoding the terms is as follows:

Case I: *Non-target table with a single attribute*

- Step 1) Compute the cardinality of the attribute's domain in the non-target table. For continuous and discrete values, discretizes these values first and take the number of bins as the cardinality of the attribute's domain
- Step 2) To encode values, find the appropriate number of bits, n, that can represent these values, where $2^{n-1} < |\text{Attribute's Domain}| \leq 2^n$. For example, if the attribute has 5 different values (London, New York, Chicago, Paris, Kuala Lumpur), then

we just need 3 ($5 < 2^3$) bits to represent each of these values (001, 010, 011, 100, 101).

- Step 3) For each encoded term, add this term to the bag of terms that describes the characteristics of a record associated with them.

Case II: *Non-target table with multiple attributes*

- Step 1) Repeat step 1) and step 2) in Case I, for all attributes
- Step 2) For each instance of a record stored in the non-target table, concatenate *p*-number of columns' values, where *p* is less than or equal to the total number of attributes. For example, let $F = (F_1, F_2, F_3,\ldots, F_k)$ denote *k* field columns or attributes in the non-target table. Let $F_1 = (F_{1,1}, F_{1,2}, F_{1,3}, \ldots, F_{1,n})$ denote *n* values that are allowed to be used by field/column F_1. So, we can have instances of record in the non-target table with these values ($F_{1,a}, F_{2,b}, F_{3,c}, F_{4,d},\ldots, F_{k-1,b}, F_{k,n}$), where $a \leq |F_1|$, $b \leq |F_2|$, $c \leq |F_3|$, $d \leq |F_4|,\ldots, b \leq |F_{k-1}|$, $n \leq |F_k|$. If $p = 1$, we have a bag of a single attribute's values, $F_{1a}, F_{2,b}, F_{3,c}, F_{4,d},\ldots, F_{k-1,b}, F_{k,n}$, as the produced terms. If $p = 2$, then we have a bag of paired attribute's values, $F_{1,a}F_{2,b}, F_{3,c}F_{4,d},\ldots, F_{k-1,b}F_{k,n}$ (provided we have an even number of fields). Finally, if we have $p = k$, then we have $F_{1,a}F_{2,b}F_{3,c}F_{4,d}\ldots F_{k-1,b}F_{k,n}$ as a single term produced.
- Step 3) For each encoded term, add this term to the bag of terms that describes the characteristics of a record associated with them.

The encoding processes transform relational datasets into data represented in a vector-space model [9], had been implemented in DARA [7,8]. Given this data representation in a vector space model, we can use clustering techniques [10,11,12] to cluster these datasets as a means of aggregating them.

3 Genetic Algorithm-Based Clustering Technique

3.1. Genetic Algorithm (GA)

A Genetic Algorithm (GA) is a computational abstraction of biological evolution that can be used to some optimisation problems [13,14]. In its simplest form, a GA is an iterative process applying a series of genetic operators such as *selection*, *crossover* and *mutation* to a population of elements. These elements, called chromosomes, represent possible solutions to the problem. Initially, a random population is created, which represents different points in the search space. An *objective* and *fitness* function is associated with each chromosome that represents the degree of *goodness* of the chromosome. Based on the principle of the survival of the fittest, a few of the chromosomes are selected and each is assigned a number of copies that go into the mating pool. Biologically inspired operators like *crossover* and *mutation* are applied to these strings to yield a new generation of strings. The process of selection, crossover and mutation continues for a fixed number of generations or till a termination condition is satisfied. More detailed surveys of GAs can be found in [15].

3.2. Clustering Algorithm

Clustering [10,11,12,21] is an important unsupervised classification technique where a set of patterns, usually vectors in a multi-dimensional space, are grouped into clusters in such a way that patterns in the same cluster are similar in some sense and patterns in different clusters are dissimilar in the same sense. The clustering algorithm that we apply in this experiment uses the vector-space model [9] to represent each record. In this model, each record R_a is considered to be a vector in the term-space or pattern-space. In particular, we employed the *tf-idf* term weighting model, in which each record can be represented as

$$(tf_1 \log(n/df_1),\ tf_2 \log(n/df_2),\ \ldots,\ tf_m \log(n/df_m))$$

where tf_i is the frequency of the i-th term in the record, df_i is the number of records that contain the i-th term and n is the number of records. To account for records of different length, the length of each record vector is normalized so that it is of unit length ($||d_{tfidf}||= 1$), that is each record is a vector on the unit hypersphere. In this experiment, we will assume that the vector representation for each record has been weighted using *tf-idf* and it has been normalized so that it is of unit length. In the vector-space model, the cosine similarity is the most commonly used method to compute the similarity between two records R_i and R_j , which is defined to be $cos(R_i,R_j) = R_i^t R_j/(||R_i||\cdot|||R_j||)$. The cosine formula can be simplified to $cos(R_i,R_j) = R_i^t R_j$, when the record vectors are of unit length. This measure becomes one if the records are identical, and zero if there is nothing in common between them.

Since clustering works in an unsupervised fashion, The user has no control on the result. In this experiment, we introduce supervision to the learning scheme through some measure of cluster impurity. The basic idea is to find a set of clusters that minimize a linear combination of the cluster dispersion and cluster impurity measures.

3.3. Semi-Supervised Clustering Algorithm

As a base to our semi-supervised algorithm, we use an unsupervised clustering method optimized with a genetic algorithm incorporating a measure of classification accuracy used in decision tree algorithm, the GINI index [1]. Here, we examine the clustering algorithm that minimizes some objective function applied to k-cluster centers. Each point is assigned to the nearest cluster center by Euclidean distance. The main objective is to choose the number of clusters that minimize some measure of cluster quality. Typically cluster dispersion metric is used, such as the Davies-Bouldin Index (DBI) [2]. DBI uses both the within-cluster and between clusters distances to measure the cluster quality. Let $d_{centroid}(Q_k)$, defined in (1), denote the centroid distances within-cluster Q_k, where $x_i \in Q_k$, N_k is the number of samples in cluster Q_k, c_k is the center of the cluster and $k \leq K$ clusters. Let $d_{between}(Q_k, Q_l)$, defined in (3), denote the distances between-clusters Q_k and Q_l, where c_k is the centroid of cluster Q_k and c_l is the centroid of cluster Q_l.

$$d_{centroid}(Q_k) = \frac{\sum_i \left\| x_i - c_k \right\|}{N_k} \quad \textbf{(1)}$$

$$c_k = 1/N_k \left(\sum_{x_i \in Q_k} x_i \right) \quad \textbf{(2)}$$

$$d_{between}(Q_k, Q_l) = \left\| c_k - c_l \right\| \quad \textbf{(3)}$$

$$\text{DBI} = \frac{1}{K} \sum_{k=1}^{K} \max_{l \neq k} \left\{ \frac{d_{centroid}(Q_k) + d_{centroid}(Q_l)}{d_{between}(Q_k, Q_l)} \right\} \quad \textbf{(4)}$$

Therefore, given a partition of the *N* points into *K*-clusters, DBI is defined in (4). This cluster dispersion measure can be incorporated into any clustering algorithm to evaluate a particular segmentation of data. The *Gini index* (GI) has been used extensively in the literature to determine the impurity of a certain split in decision trees [1]. Clustering using *K* cluster centers partitions the input space into *K* regions. Therefore clustering can be considered as a *K*-nary partition at a particular node in a decision tree, and GI can be applied to determine the impurity of such partition. In this case, GI of a certain cluster, *k*, is computed, as defined in (5), where *n* is the number of class, P_{kc} is the number of points belong to *c*th class in cluster *k* and N_k is the total number of points in cluster *k*.

$$GiniC_k = 1.0 - \sum_{c=1}^{n} \left(\frac{P_{kc}}{N_k} \right)^2 \quad \textbf{(5)}$$

The purity of a particular partitioning into K clusters is (6) where *N* is the number of points in the dataset and T_{Ck} is the number of points in cluster k. In order to get better quality of cluster, we have to minimize the measure of purity.

$$purity = \frac{\sum_{k=1}^{K} T_{Ck} * GiniC_k}{N} \quad \textbf{(6)}$$

By minimizing the objective function that minimizes a linear combination of the cluster dipersion measure (DBI) and the cluster impurity measure (GI), the algorithm becomes semi-supervised. More specifically, given *N* points and *K*-clusters, select *K* cluster centers that minimize the following objective function:

f(N,K) = Cluster Dispersion + Cluster Impurity

$$\text{f(N,K)} = \frac{1}{K} \sum_{k=1}^{K} \max_{l \neq k} \left\{ \frac{d_{centroid}(Q_k) + d_{centroid}(Q_l)}{d_{between}(Q_k, Q_l)} \right\} + \frac{\sum_{k=1}^{K} T_{Ck} * GiniC_k}{N} \quad \textbf{(7)}$$

3.4. Semi-Supervised GA Clustering Algorithm

There are two phases in our method for the semi-supervised Genetic Algorithm-based Clustering algorithm. In the first phase, which is modified steps from [19], we reduce the *N* points data by grouping all points to their nearest neighbour. The purpose of this data reduction is to speed up the process of genetic clustering. In phase II, we use genetic algorithm to cluster the *m* data points based on the objective function, defined in (7), $m < N$.

Phase I: Data Reduction

1) For every object O_i, find the distance to its nearest neighbour, $d_{NNj}(O_i) = \|O_i - O_j\|$, where O_j is the nearest neighbour to O_i and $i \neq j$.
2) Compute the average distance of all objects to their nearest neighbour, $d_{AVE} = \frac{1}{N}\sum_{i=1}^{N} d_{NNj}(O_i)$
3) Let $d = scale \cdot d_{AVE}$, where d is computed based on the scale's value (Initial value 0.5 in this experiment). Now, view the n objects as nodes of a graph, and connect all nodes that has distance less than or equal to d. Increment *scale* by 0.1.
4) Repeat step 3) as far as there is no overlapping of connected nodes. This is to ensure that all the connected objects are close enough to one another without overlapping cluster.
5) Find all the connected nodes and let the data sets represented by these connected nodes be denoted by $B_1, B_2, B_3, \ldots, B_{m-1}, B_m$ where m is the number of connected nodes and $m < N$, since B_i consists of 1 or more connected nodes, $i \leq m$.
6) Compute m cluster centers $z_1, z_2, z_3, \ldots, z_m$ from all connected components $B_1, B_2, B_3, \ldots, B_{m-1}, B_m$ from 5), where $z_i = \frac{1}{N_i}\sum_{xj \in Bi} x_j$, $i = 1,2,3,\ldots,m$, where N_i is the number of nodes connected in B_i

Phase II: GA-Clustering Algorithm

Population Initialization Step. A population of X strings of length m is randomly generated, where m is the number of the sets (connected components) obtained from the first part (Phase I: data reduction). X strings are generated with the number of 1's in the strings uniformly distributes within $[1,m]$. Each string represents a subset of $\{B_1, B_2, B_3, \ldots, B_{m-1}, B_m\}$. If B_i is in this subset S, the ith position of the string will be 1; otherwise, it will be 0, where $i = 1,2,3,\ldots,m$. Each B_i in the subset S is used as a seed to generate a cluster. If B_j is not in the subset, they will be merged to the nearest B_k in the subset S, where $j,k = 1,2,3,\ldots,m$ and $j \neq k$. The merging of these two components, B_j and B_k, is based on the distance between their centers, and this forms a new cluster. After merging, the size and the center of the new cluster will be recomputed. We will repeat the merging process for all components that are not listed in the subset S.

Fitness Computation. The computation of the objective or fitness function has two parts; Cluster Dispersion and cluster impurity. In order to get the best number of clusters, we need to minimize the DBI (4). On the other hand, in order to group the same type of objects together in a cluster, we need to minimize the impurity function. Since in GA, we want to maximize the objective and fitness function, the objective fitness function (OFF) that we want to maximize will be as follows (8).

OFF = 1/Cluster Dispersion + 1/Cluster Impurity

$$\text{OFF} = \frac{1}{\frac{1}{K}\sum_{k=1}^{K} \max_{l \neq k} \left\{ \frac{d_{centroid}(Q_k) + d_{centroid}(Q_l)}{d_{between}(Q_k, Q_l)} \right\}} + \frac{1}{\frac{\sum_{k=1}^{K} T_{Ck} * GiniC_k}{N}} \quad \textbf{(8)}$$

Selection Process. For the selection process, a rouleete wheel with slots sized according to the fitness is used. The construction of such a roulette wheel is as follows;

- Calculate the fitness value for each chromosome, f_i and $i \leq X$, and get the total overall fitness for X strings of chromosome, $T_{Fitness}$.
- Calculate the probability of a selection p_i for each chromosome, $i \leq X$, $p_i = f_i/T_{Fitness}$.
- Calculate the cumulative probability q_i for each chromosome, $q_i = \sum_{j=1}^{i} p_j$.

The selection process is based on spinning the roulette wheel, X times; each time we select a single chromosome for a new population in the following way:

- Generate a random (float) number r from the range of [0..1].
- Select the i-th chromosome such that $q_{i-1} < r \leq q_i$

Crossover. A pair of chromosome, c_i and c_j, are chosen for applying the crossover operator. One of the parameters of a genetic system is probability of crossover p_c. In this experiment, we set $p_c = 0.25$. This probability gives us the expected number $p_c \cdot X$ of chromosomes, which undergo the crossover operation. We proceed by

- Generate a random (float) number of r from the range [0..1].
- Do crossover of if $r < p_c$. For each pair of coupled chromosomes we generate a random integer number *pos* from the range [1..*m-1*] (where m is the length of the chromosome), which indicates the position of the crossing point.

Mutation. The mutation operator performs a bit-by-bit basis. Another parameter of the genetic system, probability of permutation p_m gives the expected number of mutated bits $p_m \cdot m \cdot X$. In this experiment, we set $p_m = 0.01$. In the mutation process, for each chromosome and for each bit within the chromosome

- Generate a random (float) number of r from the range [0..1].
- Do mutation of each bit if $r < p_m$.

Following selection, crossover and mutation, the new population is ready for its next generation. This evaluation is used to build the probability distribution for a construction of a roulette wheel with slots sized according to current fitness values.

The rest of the evolution is just a cyclic repetition of selection, crossover and mutation until a number of specified generations or specific threshold has been achieved.

Once the generation of new chromosomes stops, clusters with few numbers of objects will be removed and its members are moved to the nearest cluster (based on the distance between centres of the clusters).

4 Experiment and Experimental Results

Our experiments are designed to demonstrate: 1) the performance gain of semi-supervised genetic algorithm-based clustering technique (SSGA-DARA) over the K-clusters clustering technique (K-DARA), genetic algorithm-based clustering (GA-DARA) and 2) that our proposed Genetic Algorithm-based DARA method of data transformation outperforms other relational data mining approach to relational data mining. In this experiment, we chose two well-known datasets, the Mutagenesis [16], Musk [17].

DARA transformation process is performed with three different settings, called K-DARA, GA-DARA, SSGA-DARA. In K-DARA transformation, records are clustered based on K_K number of clusters, where K_K is manually defined by user. We ran this algorithm ten times with 10 different values of K and we took the average accuracy of the J48 (C4.5) classifier implemented in WEKA [20]. Next, in GA-DARA transformation, records are automatically clustered to K_{GA} number of clusters, using the genetic algorithm (GA) by taking the measure of cluster's dispersion as the objective fitness function. Other parameters were set to p_c = 0.25 (crossover probability), and p_m = 0.01 (permutation probability). In contrast, in SSGA-DARA transformation, records are automatically clustered to K_{SSGA} number of clusters, using the genetic algorithm (GA) by taking the measure of cluster's dispersion and impurity as the objective fitness function. Other parameters were set to p_c = 0.25 (crossover probability), and p_m = 0.01 (permutation probability).

Table 3 shows the results of DARA-based performance accuracy, in which three data transformation algorithms are compared; K-DARA, GA-DARA and SSGA-DARA. The aggregation method that uses a semi-supervised genetic algorithm-based clustering technique, done by transforming data first into vector space model using DARA [7,8], proved particularly successful on large datasets.

The accuracy estimations, from *leave-one-out* performance result for the classification of the transformed Musk and Mutagenesis datasets, using genetic algorithm-based clustering technique (GA-DARA) is much lower compared to the one with semi-supervised genetic algorithm-based clustering technique (SSGA-DARA). It is not surprising that the K-DARA and GA-DARA algorithms did poorly, since neither of these algorithms addresses the cluster impurity problem. However, it is somewhat surprising that the GA-DARA algorithm performs virtually the same as the K-DARA algorithm, since GA-DARA algorithm implicitly considers the best number of clusters to minimize the DBI measure, without limiting the number of clusters, as in K-DARA.

Table 1. Leave-One-Out CV performance on Musk and Mutagenesis

Data Transformation and Algorithm	Musk	Mutagenesis B1	Mutagenesis B2	Mutagenesis B3
K-DARA+C4.5	76%	77%	79%	82%
GA-DARA+C4.5	62%	77%	75%	78%
SSGA-DARA+C4.5	**76%**	**90%**	**88%**	**87%**

Table 2. Results of paired one-sided *t*-tests (p=0.0025): symbol ● indicates significant improvement performance by method in row over method in column and symbol ○ indicates no significant improvement performance by method in row over method in column, on datasets Musk and Mutagenesis (A = K-DARA, B = GA-DARA and C = SSGA-DARA)

	Musk			Mutagenesis **B1**			Mutagenesis B2			Mutagenesis **B3**		
Method	A	B	C	**A**	**B**	**C**	A	B	C	**A**	**B**	**C**
A	-	●	○	-	○	○	-	●	○	-	○	○
B	○	-	○	○	-	○	○	-	○	○	-	○
C	○	●	-	●	●	-	●	●	-	●	●	-

Table 2 shows the results of paired one-sided *t*-tests (p=0.0025), where the symbol '●' indicates significant improvement performance by method in row over method in column and symbol '○' indicates no significant improvement performance by method in row over method in column, on dataset Musk and Mutagenesis datasets, and A represents the K-DARA algorithm, B represents the GA-DARA algorithm and C represents the SSGA-DARA algorithm.

Table 3 shows the results of DARA-based (SSGA-DARA + C4.5) performance accuracy compared to other previously published results, such as the C4.5 decision tree algorithm, the back-propagation neural network algorithm, PROGOL, FOIL and TILDE, all of which ignored the multiple-instance problem and treated all of the positive instances as positive examples.

Based on Table 3, the accuracy estimations from *leave-one-out cross-validation* performance result for the classifier J48 [20] on the transformed Musk and Mutagenesis datasets using semi-supervised genetic algorithm-based clustering technique (SSGA-DARA) is at least better than the other published results.

Table 3. Comparison of performance accuracy on Musk and Mutagenesis

Data Transformation and Algorithm	Musk	Mutagenesis B1	Mutagenesis B2	Mutagenesis B3
PROGOL	-	76%	81%	83%
FOIL	-	61%	61%	83%
TILDE	87%	75%	79%	85%
SSGA-DARA+C4.5	**76%**	**90%**	**88%**	**87%**
C4.5	67%	-	-	-
Back Propagation	75%	-	-	-

5 Conclusions

The multiple instances problem is an important problem that arises in real-world tasks where the training examples are ambiguous: a single record may have many associations with feature vectors that describe it, and yet only a few of those vectors may be relevant for the observed classification of the record. In this paper, we have presented an algorithm to transform multiple instances problem into a vector space model that is suitable to clustering operations. A novel method for semi-supervised learning that combines supervised and unsupervised learning techniques has been also introduced to extract patterns from multiple tables with a high degree of *one-to-many* association. The basic idea is to treat a series of records, associated with a single record in the target table, as a bag of terms, and take an unsupervised clustering method and simultaneously optimize the misclassification error of the resulting clusters. Experimental results show that using DBI for cluster dispersion and GI for cluster impurity finds solutions with much greater accuracy. The basic ideas in this paper: incorporating classification information into an unsupervised algorithm to aggregate records with multiple instances in relational datasets are promising areas for future research.

References

1. Breiman, L., Friedman, J., Olshen, T., Stone, C., Classification and Regression Trees. Wadsworth International, California, 1984.
2. Davies, D.L., Bouldin, D.W., A cluster separation measure. IEEE Transactions and Pattern Analysis and Machine Intelligence, 1979, 1(2):224-227.
3. Agrawal, R., Imieliński, T., Swami, A., Mining association rules between sets of items in large databases, Proceedings of the 1993 ACM SIGMOD international conference on Management of data, Washington, D.C., United States, May 25-28, 1993, 207-216.
4. Agrawal, R., Srikant, R., Fast Algorithms for Mining Association Rules in Large Databases, Proceedings of the 20th International Conference on Very Large Data Bases, September 12-15, 1994, 487-499.
5. Han, J., Fu, Y., Discovery of Multiple-Level Association Rules from Large Databases, Proceedings of the 21th International Conference on Very Large Data Bases, September 11-15, 1995, 420-431.
6. Maurice, A., Houtsma, W., Arun, N., Swami, A., Set-Oriented Mining for Association Rules in Relational Databases, Proceedings of the Eleventh International Conference on Data Engineering, March 06-10, 1995, 25-33.
7. Alfred, R., Kazakov, D., Data Summarization Approach to Relational Domain Learning Based on Frequent Pattern to Support the Development of Decision Making. In the Proceedings of The Second International Conference of Advanced Data Mining and Applications, (ADMA 2006), 14-16 August, Xi'An, China, 2006. LNAI 4093, eds. X. Li, O.R. Zaiane, and Z. Li, 2006, 889-898.
8. Alfred, R., Kazakov, D., Pattern-Based Transformation Approach to Relational Domain Learning Using DARA. In the Proceedings of The 2006 International Conference on Data Mining (DMIN 2006), LAS VEGAS, Nevada, USA, CSREA Press, eds. S.F.Crone, S. Lessmann, R. Stahlbock, ISBN. 1-60132-004-3, June 25-29, 2006, 296-302.

9. Salton, G., Wong, A., Yang, C.S., A vector space model for automatic indexing. Communications of the ACM, 1975, 18:613–620.
10. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P., Automatic subspace clustering of high dimensional data for data mining applications. Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, ACM Press, 1998, 94-105.
11. Hofmann, T., Buhnmann, J.M., Active data clustering. In Advance in Neural Information Processing System 10, 1998.
12. Hartigan, J.A., Clustering Algorithms, Wiley, New York, 1975.
13. Holland, J., Adaption in Natural and Artificial Systems. Univeristy of Michigan Press, 1975.
14. Goldberg, D.E., Genetic Algorithms –in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Inc, 1989.
15. Filho, J.L.R., Treleaven, P.C., Alippi, C., Genetic algorithm programming environments, IEEE Compu. 1994, 27: 28-43.
16. Srinivasan, A., Muggleton, S.H., Sternberg, M.J.E., King, R.D., Theories for mutagenicity: A study in first-order and feature-based induction. Artificial Intelligence, 85, 1996.
17. Dietterich, T.G., Lathrop, R.H., Lozano-Perez, T., Solving the multiple-instance problem with axis-parallel rectangles, Artificial Intelligence, 1997, 89(1-2) p. 31-71.
18. Workshop notes on Discovery Challenge PKDD '99, 1999.
19. Tseng, L.Y., Yang, S.B., A genetic approach to the automatic clustering problem. Pattern Recognition 34, 2001, 415-424.
20. Witten, I., Frank, E., Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufman, 1999.
21. Zhao, Y., Karypis, G., Hierarchical Clustering Algorithms for Document Datasets. Technical Report 03-027, Dept. of Computer Science, University of Minnesota, 2003.