

Pattern-Based Transformation Approach to Relational Domain Learning Using Dynamic Aggregation for Relational Attributes

Rayner Alfred and Dimitar Kazakov

Abstract—Due to the widespread use of relational databases (mySQL, Oracle, DB2, MsSQL), most data are stored as multiple tables in what can be a very large database. As a result, more efficient algorithms for mining data from multi-relational domain need to be implemented. Inductive Logic programming (ILP) techniques are useful for analyzing data in multi-relational databases. Unfortunately, even though not complex in structure, such business data are often large and contain highly non-determinate components, making them difficult for ILP learners geared towards structurally complex tasks. In this paper, we build a novel transformation-based approach to relational domain learning and describe the transformation process implemented through relational aggregation based on pattern distance. In this paper, we present the prototype of “Dynamic Aggregation of Relational Attributes” (hence called DARA) that is capable of mapping *one-to-many* relationship into *one-to-one* relationship, while preventing loss of information, in handling classification task in relational domains. We experimentally show these results in a multi-relational domain that show higher percentage of correctly classified instances and illustrate set of rules extracted using our approach.

Keywords— Data Pre-processing, Relational Data Mining, Attribute Encoding, Feature Transformation, Distance-based Clustering

I. INTRODUCTION

The processing power to acquire and store large amounts of data has increased dramatically over the last few years. Despite the growing computational power of modern computers, our abilities to analyze these data sets are limited for data stored in a relational model (multiple tables). Data representation stored in a relational model differs from the traditional feature-vector (single table) representation used in traditional data mining tasks. A relational model is more expressive than attribute value model in capturing and describing complex structure and relationships in the medical domain because it allows for multiple tables to be linked through primary and foreign keys. Mining knowledge from relational databases has a few advantages over mining knowledge from a single table, namely standardized relational format for most database, highly expressive power in capturing complex data, and the ability to integrate background knowledge.

Fig. 1 illustrates the schema of a relational database describing part of the financial dataset from PKDD CUP

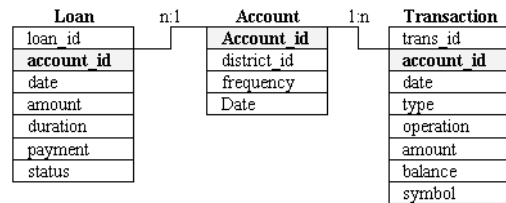


Fig. 1. Financial Dataset Schema Example (PKDD CUP 99)

1999. For simplicity, we only take three tables to illustrate our technique. The diagram shown above consists of 3 tables:

- relation **ACCOUNT** - describes char of an account
- relation **LOAN** - describes a loan granted for a given account,
- relation **TRANSACTION** - describes one transaction.

The line in Fig. 1 indicates the relationship between relations via the primary key (*account_id* in table Account) and foreign key (*account_id* in both table Loan and Transaction). For instance, suppose the bank wants to improve their services. However, bank managers may have a vague idea of the characteristics of a good client (whom to offer some additional services) and a bad client (whom to watch carefully to minimize the bank losses) based on the loan status. The bank managers hope to improve their understanding of customers and seek specific actions to improve services. A mere application of a discovery OLAP tool will not be convincing for them. To get more information, we need to join these relations to get more information about each account. However, this leads to another problem in which relation LOAN has *one-to-many* relationship with relations TRANSACTION, and most traditional data mining tools cannot handle relational datasets unless data reduction or data transformation is applied to convert this relational data into a single table.

Given the availability of relational data, its expressiveness and rich background knowledge, there is a strong need to develop automated methods that enable relational data to be processed more effectively. This work presents a new modelling approach that is capable of learning classification-clustering hybrid model from a relational database directly. In this modelling approach, we present a novel approach of generalizing or mapping data with *one-to-many* relationship in learning from relational domain and show how we preserve the information stored in the relational models by using hierarchical multi-attribute models [2] (Shown in Fig. 4). This paper is organized as follows. In section II we cover related work in multi-relational data mining. Section III

Rayner Alfred is with the Computer Science Department, University of York, YORK, YO105DD United Kingdom (ralfred@cs.york.ac.uk), on study leave from the Universiti Malaysia Sabah, Kota Kinabalu, 88999 Sabah, Malaysia (ralfred@ums.edu.my).

Dimitar Kazakov is with the Computer Science Department, University of York, YORK, YO105DD United Kingdom (kazakov@cs.york.ac.uk)

introduces the concept of feature construction and types of aggregation. We describe the concept of pattern-based aggregation technique in section IV. In section V, we describe the implementation of our approach. Experimental results are presented in section VI and this paper is summed up in section VII.

II. RELATED WORK

Relational learning research is not a new research area and it has a long history. Muggleton [13] introduced the concept of Inductive Logic Programming (ILP) and its theory, methods and implementations in learning multi-relational domains. ILP methods learn a set of existentially quantified first-order Horn clauses that can be applied as a classifier. ILP systems, using refinement graph search, usually apply two refinement operators: i) adding a predicate to the body of a clause, involving one or more variables already present, and possibly introducing one or more new variables, or ii) a single variable substitution [5]. In ILP, aggregation of one-to-many relationships is achieved through existential quantification and is part of the search process through the model space. ILP's challenges are in two other directions. Firstly, it is quite normal that current databases are highly non-determinate and ILP's limitation is the deterministic nature of the rules discovered. Secondly, even though the number of involved relations may not be huge, the total size of these relations often will be, and scalability to perhaps millions of tuples is important.

In a relational learner based on logic-based propositionalization [11], instead of searching the first-order hypothesis space directly, one uses a transformation module to compute a large number of propositional features and then apply a propositional learner. Both ILP and binary propositionalization use existential unification of first-order logic clauses for aggregation. However, both ILP and binary propositionalization lack support for numerical aggregation. In general, propositionalisation approaches may outperform ILP or MRDM systems in terms of speed, as has been suggested before in the literature [4], [20]. The choices of aggregation methods and parameters also have significant effects on the results in noisy real-world domains [9]. Krogel [12] have conducted a comparative evaluation of approaches to Boolean and numeric aggregation in propositionalization; however their results are inconclusive. In contrast, other researchers have found that logic-based relational learning and logic-based (binary) propositionalization perform poorly on a noisy domain compared to numerical propositionalization [16], [9].

Distance-based methods [8] are another variant of relational learning. Their central idea is that it is possible to compute the mutual distance (or similarity) for each pair of object in a particular domain under consideration. Once the mutual distance between object pairs is computed, one can apply a clustering technique to group a set of instances I from an instance space X into different subsets (clusters) such that objects within a cluster are maximally similar to each other while simultaneously being maximally dissimilar

to objects from other clusters. The two prominent clustering approaches are bottom-up *agglomerative* clustering [3] and iterative *k-means* clustering [15]. Kirsten [8] mentioned about the scalability issue in which distance-based methods require all instances to be stored until classification time and also the task of computing distances between first order objects is an expensive operation compared to computing a Euclidean distance [7]. This method can be used to aggregate multiple bags of objects related to a number of cases defined by calculating the minimum distance of all possible pairs of objects [16].

In learning relational domains, Inductive Logic Programming has difficulties in modelling uncertainty over the attributes of objects in the domain and uncertainty over the relations between the objects. Probabilistic Relational Models (PRMs) [6], [10] provide another approach to relational data mining that is grounded in a sound statistical framework. In PRMs, they introduced a model that specifies for each attribute of an object its (probabilistic) dependence on other attributes of that object and on attributes of related objects. The dependence model is defined at the level of classes of objects. PRMs are particularly well suited to exploratory data analysis, since they generally do not focus on a single classification task. Propescul et al. proposed a combined approach called Structural Logistic Regression (SLR) that combines relational and statistical learning [18]. They used logistic regression with FOIL-supplied features in a relatively loosely coupled manner, in which SLR improves the classification performance when high precision is required.

Database numeric aggregation [9] technique proposes a method in which aggregation is done by using some of the built-in functions of common relational database systems such as *count*, *min*, *max*, *sum*, *avg* and *exist*. Knobbe [9] used a selection graph to illustrate the algorithm. In this technique, numerical aggregates in combination with logic-based feature construction were proposed. The experimental results in [9] demonstrate that the proposed approach is at least competitive with existing multi-relational tools, such as Progol and Tilde. The proposed approach has two major differences from multi-relational techniques, which may be the source of the better performance: the use of aggregates and the use of propositionalisation. However this technique can only be applied to bags of single attributes and cannot express patterns over multiple attributes in relational domain. Another approach proposed by [17] uses vector distances for dimensionality reduction and is capable of aggregating high-dimensional categorical attributes that traditionally have posed a significant challenge in relational modeling.

III. THE ROLE OF FEATURE CONSTRUCTION

Traditionally, hypotheses are based on the rules constructed during the learning process and typically, rules are constructed by defining the heads and bodies of clauses in which the bodies consist of relations of features, which need to be constructed in a separate step. Distinguishing

feature construction as a separate step also leads to a new transformation-based learning approach such as propositionalization [11]. Propositionalization can be considered as static feature generation approach in which features are first constructed from the relational representation and then presented to a propositional algorithm. However, some Inductive Logic Programming systems perform dynamic feature generation. For example, Progol [14] uses A*-like algorithm to direct the search for relevant features.

A relevant feature describes subsets of the training set that are unusual or interesting. For instance, the class distribution among the instances described by the new constructed feature may be different from the class distribution over the original training set in a statistically significant way or there is a sufficiently large proportion of the training set that shared the constructed feature. Aggregation is one of the feature construction techniques in a relational domain and in fact, aggregation is an essential component of relational model induction and has significant impact on generalization performance for domains with important *one-to-many* relationships. In the next section, we will describe the degree of aggregation in a relational data model. Fig. 2 illustrates the types of aggregation in a relational data model. Aggregation can be defined as a summarization of the underlying pattern or distribution from which the related objects were sampled. Therefore, the design and selection of appropriate aggregations depend on the characteristics of the value distributions. Generally, there are three types of aggregation; *Vertical*, *Horizontal* and *Cross Aggregations*.

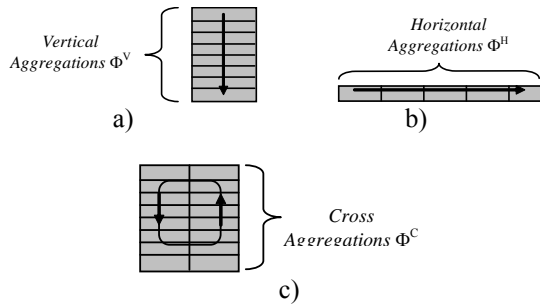


Fig. 2. The type of aggregation a) Vertical Aggregation b) Horizontal Aggregation and c) Cross Aggregation

A. Vertical Aggregation

Let Φ denotes the aggregation of multiset values to a categorical or numerical value. In Fig. 2(a), a *vertical aggregation* Φ^V is a mapping from multiset values to a categorical or numerical value, λ_V . It is also known as basic aggregation. Most relational database systems have these aggregation functions for multiset values. For example, aggregation techniques used for categorical value are *mode* and *count*. However, these operators (*mode*, *count*) can capture only limited discriminative information. On the other hand, relational database's functions like *sum*, *average*, *count* functions can be used to aggregate numerical values. A vertical aggregation can be computed by

projecting the column and aggregating the column's multiset values based on a predefined mapping function (sum, count, average, mode), grouped by the target column. For example, suppose we have two relations R1 and R2, with attributes $(R1.A1, \dots, R1.A_n)$ and $(R2.B1, \dots, R2.B_n)$ and R1 has *one-to-many* (1: n) relationship with R2, then the aggregation Φ^V of attribute B_m , where $(1 \leq m \leq n)$, to a categorical or numerical value, λ_V , would be as follows;

$$\Phi^V(\Pi_{R2.B_m}(R1 \underset{R1.A_i = R2.B_j}{\otimes}_{1:n} R2)) \rightarrow \lambda_V \quad (1)$$

where $1 \leq m, i, j \leq n$, $\Pi_{R2.B_m}$ is the projection of column B_m from the joined relation of R1 and R2 based on the condition $R1.A_i = R2.B_j$ and the λ_V denotes the categorical or numerical value resulted from the aggregation function and $\otimes_{1:n}$ denotes the left inner join of relations R1 and R2 with *one-to-many* (1: n) relationship. Notice that when we have *one-to-one* (1:1) or *many-to-one* (n:1) relationship, multiset value aggregation will not be required.

B. Horizontal Aggregation

A *horizontal aggregation* Φ^H is a mapping from n sets of attributes' values to a categorical or numerical value, λ_H , where the number of attributes given for aggregation should be more than one, $n > 1$, as shown in Figure 2(b). Horizontal aggregation normally describes the association of two or more attributes in a table. Given n attributes in the form of a feature vector (a_1, \dots, a_n) , one can map this feature vector to a categorical value (e.g. increasing, decreasing, stable) or to a numerical value (e.g. differences or summation). A horizontal aggregation Φ^H can be described as follows;

$$\Phi^H(\Pi_{A1 \dots A_n}(R1 \underset{R1.A_i = R2.A_j}{\otimes}_{1:1 \vee n:1} R2)) \rightarrow \lambda_H \quad (2)$$

In this concept, $\Pi_{A1 \dots A_n}$ is the projection of columns $A1$ through A_n from the joined relation of R1 and R2 based on the condition $R1.A_i = R2.A_j$ and the λ_H denotes the categorical or numerical value resulted from the aggregation function Φ^H and $\otimes_{1:1 \vee n:1}$ denotes the left inner join of relations R1 and R2 with *one-to-one* (1:1) or *many-to-one* (n:1) relationship. With *one-to-one* (1:1) or *many-to-one* (n:1) relationship, a tuple in R1 will correspond to exactly one tuple in R2. So, only the values across the attributes that can be aggregated based on predefined concept. The mapping function for horizontal aggregation is more complicated compared to the vertical aggregation since the aggregation's values are reflected by the attributes' independencies or dependencies (e.g. time-series data). Logic-based propositionalization [11] proposes a horizontal aggregation that expresses a feature vector implements a Boolean conditioning,

C. Cross Aggregation

A *cross aggregation* Φ^C combines the vertical Φ^V and horizontal Φ^H aggregation techniques. A *cross aggregation* (Figure 2(c)) is a mapping from n sets of attributes with

multiset values to a categorical or numerical value, λ_{VH} . In this aggregation, an interpretation of the data is required to represent a block of data with a single categorical or numerical value.

$$\Phi^C(\prod_{A1...An}(R1_{R1.Ai=R2.Aj} \otimes_{m:n} R1)) = \Phi^V(\Phi^H(\prod_{A1...An}(R1_{R1.Ai=R2.Aj} \otimes_{m:n} R2))) \rightarrow \lambda_{VH} \quad (3)$$

For example, consider finding a frequent pattern in employee's account balances. This aggregation requires the employees' salary at the beginning of every month and the balance of the account at the end of the month. In particular, cross aggregation has received little treatment due to the fact that this technique requires domain knowledge of the problem at hand to properly aggregate the dataset.

IV. PATTERN-BASED FEATURE AGGREGATION

A common method to aggregate a single categorical attribute with numerous patterns is the selection of a subset of pattern that appears most often or based on the distribution. In this approach, each record is viewed as a vector whose dimensions correspond to patterns stored in the target table in relational domain. The component magnitudes of the vector are the *pf-irf* weights of the patterns which is adapted from [12].

$$pf-irf = pf(p, r) \cdot irf(p) \quad (4)$$

Pf-irf, as described in (4), is the product of pattern frequency $pf(p, r)$, which refers to the number of times pattern p occurs in the corresponding record r , and the inverse record frequency, as described in (5),

$$irf(p) = \log \frac{|R|}{rf(p)} \quad (5)$$

where $|R|$ is the number of records in the table and $rf(p)$ is the number of records in which pattern p occurs at least once. For instance, client X may have three out of ten transactions use cash worth below 150. So, we can say that client X has three occurrence of pattern p ($p = 3, |R| = 10$), where p is the pattern of making cash transaction with the amount less than 150. The similarity between two records is then

$$sim(r_i, r_j) = \frac{r_i \cdot r_j}{||r_i|| \cdot ||r_j||} \quad (6)$$

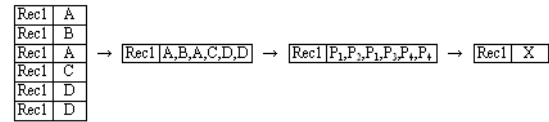
where r_i and r_j are vectors with *pf-irf* coordinates as described above. As mentioned before, aggregation can be defined as a summarization of the underlying pattern or distribution from which the related objects were sampled. Once we compute the *pf-irf* weights, then we can compute the distance between each record and cluster them based on their weights.

By grouping them into clusters or templates, we are aggregating them based on the underlying pattern or distribution from which the related objects were sampled.

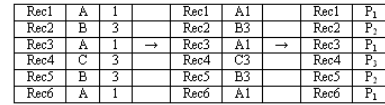
The key question is: how many clusters do we need? A widely adopted definition of optimal clustering is a partitioning that minimizes distances within and maximizes distances between clusters. Bezdek discussed variation of several ways of within-clusters and between-clusters distances [1].

V. TRANSFORMATION PROCESS OF RELATIONAL DOMAINS USING PATTERN FREQUENCY

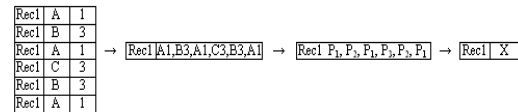
Fig. 3(a) shows how a record with bag of multiset values is generalized into a single value. The generalization task is done by computing the similarities (6) between records, by first computing the *pattern-frequency* and *inverse-record frequency* (4) and then grouping them based on the distance between records. This individual-centered concept, in which a row that belongs to a specific record, illustrates a series of patterns characterize the individualism of each record. As a result, a high-dimensional *one-to-many* relationship between two tables can be treated as a document (record in database)



a)



b)



c)

Fig. 3. Transformation of Frequent Pattern a) Vertical aggregation b) Horizontal aggregation c) Cross aggregation

with keywords (multiple rows that correspond to the particular record) describing the content of the document.

Fig. 3(b) illustrates the generalization of more than one attribute into a single value and grouping them based on the pattern observed. This type of generalization is useful to form a time series pattern (temporal values). Finally, Fig. 3(c) shows how a single record could be generalized based on a block of data (rows of patterns).

VI. IMPLEMENTATION

A. Data Representation

The database contains a set of relations. One of them is the target relation (*Loan*), with class labels on its tuples as shown in Fig. 1. The other relations (*Account*, *Transaction*) have no class labels. We may propagate the class labels to their tuples. Each relation may have one primary key and several foreign keys, each pointing to the primary key of some other relation. Based on the schema shown in Fig. 1, we would like to classify account holders according to the status based on the characterization of their transaction

patterns. Fig. 4 shows hierarchical multi-attribute models [2] of relations *Loan* and *Transaction* that possibly joined through the primary key/foreign-key stored in relation *account*. Relation *Loan* has a *one-to-many* relationship with relation *Transaction*. We use the *DARA algorithm* to generalize each individual in relation *Loan* that corresponds to the data stored in relation *Transaction* and insert the newly generated generalization feature (Transaction in Fig. 4) into the relation *Loan* to perform the classification task of the status of the clients.

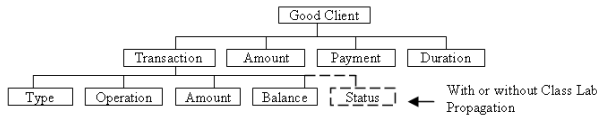


Fig. 4. Hierarchical multi-attribute models of Dataset PKDD CUP 1999

Algorithm DARA

Input: A relational database

Output: a set of rules that distinguish the class label.

Procedure:

```

Rule set R = empty
Create-Pattern();
Compute-Similarity-And-Transform()
Update-Target-Table()
Rule r1 = Find-Rule-Target-Table()
Add r1 to the R.
Rule r2 = Find-Rule-Support-Table()
Add r2 to the R
Return R
End Procedure

```

Fig. 5. Algorithm of Dynamic Aggregations of Relational Attributes

B. Dynamic Aggregations of Relational Attributes (DARA) Algorithm

Given a dataset contains one target relation *T*, and another relation *A* (called support table) that is linked through a foreign key, and the relationship of *T* and *A* is *one-to-many* relationship, *DARA algorithm* builds a pattern-based transformation dataset for learning in a multi-relational domains. The main idea is to generate a cluster that generalizes the relationship between *T* and *A*, and insert the newly generated features into the target table and build the main classifier. *DARA* supports clustering using *k-means* clustering and *hierarchical agglomerative* clustering. In the following experiment, we will show the results of these two techniques of clustering. However, for simplicity, we will only show the set of rules generated by the *k-means* clustering as shown in Fig. 8. In addition to the rules generated by the main classifier, we may add rules *r1* with rules *r2* (Fig. 5) induced from the *frequent-pattern* induced at the earlier stage during the aggregation of patterns. The *DARA algorithm* is shown in Fig. 5.

Create-Pattern() method generates a pattern that illustrates the characteristics of individual record in the database and once all patterns are generated, *Compute-Similarity-And-Transform()* method is used to segment the records based on their similarities or differences. Based on

the transformed dataset, the new generated attribute will be inserted into the target table (*Loan*) that represents the individual characteristics of individual record obtained from the support table. In *Find-Rule-Target-Table()*, a set of rules thus can be induced using any existing attribute-value classifiers such as C4.5, Conjunctive Rules and Naïve Bayes. We use the *Weka* software [21] to extract set of rules and test the performance of the C4.5 (J48), Naïve Bayes and Conjunctive Rules classifiers. Additionally, in *Find-Rule-Support-Table()*, a different set of rules (Fig. 8) could be extracted from the transformation table that describes each cluster/segment/group by using the *frequent-item* algorithm (association rule).

VII. EXPERIMENTAL RESULTS

Here we run our algorithm on the Financial Dataset in PKDD CUP 1999 to convert it to DARA data representation and we use the *k-means* clustering technique to generalize the data. The schema is described in Fig. 1. Relation *LOAN* is the target relation and the class label (status) indicates whether each loan is paid on time ('A' stands for contract finished, no problems, 'B' stands for contract finished, loan not paid, 'C' stands for running contract, OK so far, 'D' stands for running contract, client in debt). There are 682 instances with 203 instances with 'A' status, 31 instances with 'B' status, 403 instances with 'C' status and 45 instances of 'D' status.

Table 1: Weka Classifiers' Performance for Financial Dataset

Cluster	C4.5	Naive Bayes	Conjunctive Rules
0	69.79	68.48	66.72
2	69.35	68.04	66.72
4	68.77	68.18	66.72
6	68.48	68.62	66.72
8	70.97	73.02	63.49
10	71.70	71.99	66.72
12	70.23	73.17	66.72
15	71.41	72.73	66.72
20	67.74	72.14	66.72
25	68.48	71.99	66.72
30	69.79	71.99	66.72
35	68.33	71.55	66.72
40	68.33	70.23	66.72
45	69.79	70.09	66.72
50	69.79	69.65	66.72

The accuracy estimates from *10-fold* cross validation results shown in Table 1 (*k* = 10, method = *k-means* clustering), the percentage of correctly classified instances increases significantly (*w.r.t.* t-test, *p* = 0.05) by 1.91% for C4.5 and increases very highly significantly (*w.r.t.* t-test, *p* = 0.001) 3.52% for Naïve Bayes using *DARA algorithm*, compared to the results when *k* = 0. Part of the data summarization for the financial data when *k* = 10 can be referred from Figure 8. By finding the most frequent term in the cluster, we are actually computing the association rules for most frequent pattern in each cluster. Hence, Figure 8 illustrates the association rules extracted from the financial dataset based on pattern's similarities.

Fig. 6 and 7 indicate the C4.5 results for *DARA* with C10 (10 clusters) using *k-means* clustering without class

label propagation and we obtained set of rules both from the classification task. In addition to that, we also extract a set of rules from the clustered as shown in Fig. 8. For instance, suppose we have a rule

```
duration <= 24 and duration > 12 and
C10 = 1 and amount <= 93960 → A
```

and the generalization of cluster 1 as shown below.

CLUSTER 1

- a) withdrawal,cash,less than 8730.0,less than 49374.094
- b) withdrawal,remittance to another bank,less than 17460.0,less than 49374.094

We may induce that a client is highly predicted to finish the contract with no difficulties if the duration of payment between 12 and 24 months and amount of loan is less than 93960. In addition to that, the client is also generalized as a person who does a withdrawal either by cash with the amount of transaction is less than 8730 and has a balance less than 49374.1 or a person who does a remittance to another bank, with the amount of transaction is less than 17460.0 and has a balance of less than 49374.1. In contrast, modelling the problem only using the LOAN table alone will generate a simple rule such as “duration <= 24 and amount <= 167100: A” as shown in Fig 6.

```
Instances: 682
-----
duration <= 24
| amount <= 167100: A (251.0/99.0)
| amount > 167100
| | amount <= 192744: B (13.0/7.0)
| | amount > 192744: C (5.0/2.0)
duration > 24: C (413.0/94.0)

Number of Leaves : 4
Size of the tree : 7
```

Fig. 6. Decision tree C4.5 obtained for k-means clustering with k = 10 without class label propagation without DARA Transformation

VIII. CONCLUSION AND FUTURE WORK

Multi-relational classification is a very important research area because of the popularity of relational database and more very-large database implemented. Unfortunately most existing approaches are not scalable with respect to the number of relations and the complexity of the database schema. In this paper, we propose Dynamic Aggregation of Relational Attributes (DARA), an efficient approach to transform a multi-relational datasets into an attribute-value data (or normally known as single table) that will promisingly solve the scalability problem in multi-relational classification. It uses pattern-based and distance methods to generalize the *one-to-many* high-dimensional relationship between target table with support table through the primary and foreign keys. In our experiments, it is shown that *DARA algorithm* generates rules that improve the performance of the classifier. There are some other techniques that can be used to perform the transformation such as Self Organizing Map (SOM) technique. SOM is very effective to be used when we have a lot of missing data and this could improve the transformation-based approach in multi-relational domain. Another interesting experiment is integrating and running the cluster index that will also improve the

transformation process by ensuring the best number of clusters for the generalization of the *one-to-many* relationship in relational database.

```
Instances: 682
-----
duration <= 24
| duration <= 12: A (131.0/38.0)
| duration > 12
| | C10 <= 8
| | | C10 <= 7
| | | | C10 <= 4
| | | | | C10 <= 3
| | | | | | C10 <= 2
| | | | | | | C10 <= 1
| | | | | | | | amount <= 93960: A (5.0)
| | | | | | | | amount > 93960
| | | | | | | | | amount <= 116496: C (3.0)
| | | | | | | | | amount > 116496
| | | | | | | | | | amount <= 131292: A
| | | | | | | | | | amount > 131292: C
| | | | | | | | | | C10 > 1: C (5.0)
| | | | | | | | | | C10 > 2: A (32.0/12.0)
| | | | | | | | | C10 > 3
| | | | | | | | | | amount <= 119136: D (2.0)
| | | | | | | | | | amount > 119136: B (3.0)
| | | | | | | | | C10 > 4
| | | | | | | | | | amount <= 74916: C (6.0)
| | | | | | | | | | amount > 74916: A (28.0/12.0)
| | | | | | | | | C10 > 7
| | | | | | | | | | amount <= 57360: A (2.0)
| | | | | | | | | | amount > 57360
| | | | | | | | | | | amount <= 153936
| | | | | | | | | | | | amount <= 100260: B (7.0/3.0)
| | | | | | | | | | | | amount > 100260: C (4.0/1.0)
| | | | | | | | | | | | amount > 153936: B (3.0)
| | | | | | | | | C10 > 8: A (31.0/13.0)
duration > 24
| duration <= 36
| | amount <= 214596: C (104.0/29.0)
| | amount > 214596
| | | C10 <= 3: C (6.0/1.0)
| | | C10 > 3
| | | | C10 <= 8
| | | | | C10 <= 6
| | | | | | C10 <= 5: B (3.0/1.0)
| | | | | | C10 > 5
| | | | | | | amount <= 232560: A (3.0)
| | | | | | | amount > 232560: C (3.0/1.0)
| | | | | | C10 > 6
| | | | | | | amount <= 244560: B (2.0)
| | | | | | | amount > 244560: D (4.0/1.0)
| | | | | | C10 > 8: A (5.0/1.0)
duration > 36
| C10 <= 8
| | C10 <= 7
| | | C10 <= 3: C (101.0/10.0)
| | | C10 > 3
| | | | C10 <= 4: D (11.0/1.0)
| | | | C10 > 4: C (69.0/8.0)
| | | C10 > 7: D (21.0/8.0)
| | C10 > 8: C (81.0/4.0)
```

Fig. 7. Decision tree C4.5 obtained for k-means clustering with k = 10 without class label propagation with DARA Transformation

Generalization of the Transformed Cluster

CLUSTER 1 (type of transaction, operation, amount of transaction, balance)
 withdrawal,cash,less than 8730.0,less than 49374.094
 withdrawal,remittance to another bank,less than 17460.0,less than 49374.094

CLUSTER 2
 credit,collection from another bank,less than 17460.0,less than 26479.398
 withdrawal,cash,less than 8730.0,less than 49374.094

CLUSTER 3
 withdrawal,cash,less than 8730.0,less than 49374.094
 withdrawal,cash,less than 8730.0,less than 72268.8

CLUSTER 4
 withdrawal,remittance to another bank,less than 8730.0,less than 26479.398
 withdrawal,cash,less than 8730.0,less than 26479.398

CLUSTER 5
 credit,collection from another bank,less than 17460.0,less than 49374.094
 withdrawal,cash,less than 8730.0,less than 49374.094

CLUSTER 6
 credit,collection from another bank,less than 43650.0,less than 95163.5
 credit,collection from another bank,less than 43650.0,less than 72268.8

CLUSTER 7
 withdrawal,cash,less than 8730.0,less than 49374.094
 withdrawal,remittance to another bank,less than 8730.0,less than 49374.094

CLUSTER 8
 withdrawal,cash,less than 8730.0,less than 3584.6992
 withdrawal,remittance to another bank,less than 8730.0,less than 3584.6992

CLUSTER 9
 withdrawal,cash,less than 8730.0,less than 26479.398
 withdrawal,cash,less than 8730.0,less than 49374.094

CLUSTER 10
 withdrawal,remittance to another bank,less than 8730.0,less than 72268.8
 withdrawal,cash,less than 8730.0,less than 49374.094

Fig. 8. Set of rules obtained from the C4.5 classifier with the generalization of each cluster

REFERENCES

- [1] J.C. Bezdek, 1998. Some new indexes of cluster validity, *IEEE Trans. Syst., Man, Cybern. B*, vol. 28, pp. 301-315.
- [2] B. Marko, 2001. *Decision Support*. In D. Mladenic, N. Lavrač, M. Bohanec, and S. Moyle, 2003. *Data Mining and Decision Support: Integration and Collaboration*, Kluwer Aca. Publishers.
- [3] W. Dillon, and M. Goldstein, 1984. *Multivariate analysis*, pages 157-208. John Wiley and Sons, Chichester.
- [4] S. Džeroski, H. Blockeel, B. Kompare, S. Kramer, B. Pfahringer, W. Van Laer, 1999. *Experiments in Predicting Biodegradability*, In *Proceedings of Inductive Logic Programming 1999*.
- [5] S. Džeroski and N. Lavrač, editors. 2001. *Relational Data mining*. Springer-Verlag.
- [6] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning Probabilistic relational models. In S. Džeroski and N. Lavrač, editors. 2001. *Relational Data mining*. Springer-Verlag.
- [7] T. Horvath, S. Wrobel, and U. Bohnebeck. 2001. Relational instance-based learning with lists and terms. *Machine Learning*, 43(1/2): 53-80.
- [8] M. Kirsten, S. Wrobel and T. Horvath. 2001. Distance based approaches to relational learning and clustering. In S. Džeroski and N. Lavrač, editors. *Relational Data mining*. Springer-Verlag.
- [9] A. Knobbe, M. De Haas, and A. Siebes. Propositionalization and aggregates. In *LNAI*, volume 2168, pages 277-288, 2001.
- [10] D. Koller and A. Pfeffer, 1998. Probabilistic frame-based systems. In *AAAI/IAAI*, pages 580-587.
- [11] S. Kramer, N. Lavrač and P. Flach. Propositionalization approaches to relational data mining. In S. Džeroski and N. Lavrač, editors. 2001. *Relational Data mining*. Springer-Verlag.
- [12] M.A. Krogel, S. Rawles, F. Železny, P.A. Flach, N. Lavrač, and S. Wrobel, 2003. Comparative evaluation of approaches to propositionalization. In *13th International Conference on Inductive Logic Programming (ILP)*, pages 197-214.
- [13] S.H. Muggleton and L. DeRaedt, 1994. Inductive Logic programming: Theory and Methods. *The Journal of Logic Programming*, 19 & 20:629-680.
- [14] S.H. Muggleton, 1995. Inverse Entailment and Progol. *New Generation Computing*, 13:245-286.
- [15] J. McQueen, 1967. Some Methods of classification and analysis of multivariate observations. In *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281-293.
- [16] C. Perlich and F. Provost, 2003. Aggregation-based feature invention and relational concept classes. In *Proceedings of the Ninth ACM International Conference on Knowledge Discovery and Data Mining (KDD)*.
- [17] C. Perlich and F. Provost, 2005. ACORA: Distribution-based aggregation for relational learning from identifier attributes. *Journal of Machine Learning*.
- [18] A. Propescul, L. H. Ungar, S. Lawrence, and D. M. Pennock, 2002. Structural Logistic Regression: Combining relational and statistical learning. In *Proceedings of the workshop on Multi-Relational Data Mining (MRDM-2002)*, pages 130-141. University of Alberta, Edmonton, Canada.
- [19] A. Srinivasan and R.D. King, 1999. Feature Construction with Inductive Logic Programming: A Study of Quantitative Predictions of Biological Activity Aided by Structural Attributes. *Data Mining and Knowledge Discovery*, 3(1):37-57.
- [20] A. Srinivasan, R.D. King, D.W. Bristol, 1999. An Assessment of ILP-Assisted Models for Toxicology and the PTE-3 Experiment, In *Proceedings of Inductive Logic Programming 1999*.
- [21] I. Witten, and E. Frank. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufman.
- [22] G. Salton, J. Michael, and McGill, 1986. *Introduction to Modern Information Retrieval*, McGraw-Hill, Inc., New York, NY.