A Method for Multi-relational Classification Using Single and Multi-feature Aggregation Functions

Richard Frank, Flavia Moser, and Martin Ester

Simon Fraser University
Burnaby BC, Canada V5A 1S6
{rfrank,fmoser,ester}@cs.sfu.ca

Abstract. This paper presents a novel method for multi-relational classification via an aggregation-based Inductive Logic Programming (ILP) approach. We extend the classical ILP representation by aggregation of multiple-features which aid the classification process by allowing for the analysis of relationships and dependencies between different features. In order to efficiently learn rules of this rich format, we present a novel algorithm capable of performing aggregation with the use of virtual joins of the data. By using more expressive aggregation predicates than the existential quantifier used in standard ILP methods, we improve the accuracy of multi-relational classification. This claim is supported by experimental evaluation on three different real world datasets.

Keywords: multi-relational datamining, multi-relational classification, multi-feature aggregation, existential quantifier.

1 Introduction

Multi-relational (MR) data mining [4] deals with gathering knowledge from multiple related tables by exploring their own features as well as the relationships between them. Classical mining algorithms are not applicable to MR data since tuples linked to the table studied, referred to as the target table, and stored in directly or indirectly related tables have potentially valuable information about target tuples which is not expressible in single-relational (SR) data mining without loss of knowledge [1, 4, 13].

MR classification is inherently different from SR classification because all tables have to be searched for valuable information, and relationships between the features present in the database have to be explored. There are a few techniques extending SR methods into the MR domain. One method of classifying MR data is to adopt the framework of Inductive Logic Programming (ILP) [1, 3] and use it to find rules such that they entail one of the classes. CrossMine [13] is such an ILP based MR classifier using TupleID propagation, propagating data into related tables through foreign key relationships instead of performing a physical-join in the database. As the propagation was expressed in [13], there is not enough data moved to perform aggregations over related tables, other than the target table, since the IDs from those related tables are missing. MDRTL-2 [2] uses selection graphs to represent rules which visually depict the SQL statements used to describe the rules. [6] extends this technique to include

J.N. Kok et al. (Eds.): PKDD 2007, LNAI 4702, pp. 430–437, 2007.

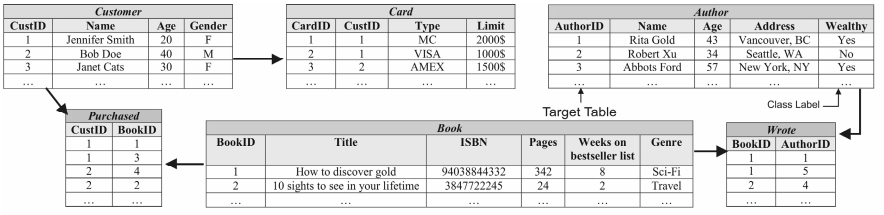

Fig. 1. Customers, books and their authors. (Arrows denote 1-many relationships).

single-feature aggregation functions. The approach of [9] uses virtual features to summarize tuples not contained in the target table. It relies on discretization resulting in information loss [5]. [11] proposes a probabilistic model to learn cluster labels. This model results in information loss as well since the information of non-target table tuples is aggregated into a single value. Decision trees were extended to the MR domain while incorporating single-feature aggregation and probability estimates for the classification labels [8]. [7] introduces a model for MR classification using attribute values and link distributions. It requires a self-join of the target table and hence cannot be applied to datasets used in this paper.

In this paper, we propose CLAMF (***CL***assification with ***A***ggregation of ***M***ultiple ***F***eatures), a method that classifies MR data using aggregation involving single and multiple features without physical joins of the data. As our running example we use the database shown in Fig. 1 consisting of the target table **`Author`** with attribute 'wealthy' as a class label. There are three other entity tables, **`Book`**, **`Customer`** and **`Card`,** and two relationship tables **`Wrote`** and **`Purchased`**. Our contributions are:

- selecting and using appropriate aggregation functions for different numbers of features and different data-types,
- incorporating multi-feature aggregation predicates into MR classification rules,
- the extension of TupleID propagation [13] in order to efficiently perform single- and multi-feature aggregation over related tables, and
- the extensive experimental evaluation of our method on three real life databases, demonstrating that the incorporation of multi-feature aggregation predicates substantially improves classification performance and produces meaningful rules.

2 Classification Rules with Aggregation Predicates

To build classification rules on a dataset, a target table T_t and a class label from T_t are selected by the user. Each tuple in T_t, called a target tuple, has exactly one class label assigned. The goal of the proposed MR classification method is to find rules, using the format of Horn clauses, that predict which class a target tuple belongs to given its own feature values, relationships to other tuples and their features. The tuples in T_t can either be interrelated to other tuples also in T_t or related to tuples in other tables. If they are interrelated, the class label may depend on other tuples in the same table and

their class labels [11, 8, 7]. Our algorithm does not deal with this special case. While we assume data to be managed in a relational database management system, we use an ILP format, ILP clauses, to represent multi-relational classification rules.

Definition 1. ILP clauses, referred to as rules, are of the form *R:head* ← *body* and are made up of a one-literal *head* specifying a class assignment, and a *body* $L_1 \wedge L_2 \wedge ... \wedge L_n$, represented simply as $L_1, L_2, ..., L_n$, which is a conjunction of literals L_i.

In this paper, a literal is restricted to be either a predicate or a comparison between two terms. As an example, the rule *wealthy(A,'Yes')* ← *author(A), wrote(B,A), book(B)* states that "*if an author has written a book then the author is wealthy*". Here book *B* is implicitly existentially quantified. Although the rule expresses that there does exist a book for the author, it is very likely that not all authors are rich. Hence this rule is relatively weak. What could help is determining how many books were written by an author but such rules cannot be expressed in classical ILP format.

2.1 Integrating Single-feature Aggregation Functions into ILP Rules

Most state-of-the-art multi-relational classification methods use only the existential quantifier but not aggregation functions to build rules. The authors of [12] extended the ILP formalism that allows for single-feature aggregation (SFA) functions to be used in ILP via *single-feature aggregation predicates* as follows:

Definition 2. A *single-feature aggregation function* maps a bag of elements from the domain of a feature to a single value from another (possibly different) domain.

Definition 3. A *single-feature aggregation predicate Agg* has the form *Agg(input, {conditions}, result)* where *input* specifies the bag of feature values to be aggregated, constrained by the *conditions*, and the *result* is a variable referencing the result of the single-feature aggregation function corresponding to *Agg*.

To use an aggregation predicate, we need an additional literal, called a comparison literal, comparing the result of the SFA predicate against a term *t*, i.e. *result* θ *t* where $\theta \in \{=, <, \leq, >, \geq\}$. For example, *Avg(A,{purchased(B,C),age(A,C)},N),N<30* formulates that *"the average age of customers C who purchased a book B is less than 30"*. *Avg* is the aggregation function, *Avg(…)* the corresponding aggregation predicate, and *N<30* the comparison literal. The different aggregation functions based on the different input data-types are below:

Numerical: *sum*, *min, average*, *median* and *standard deviation*.
Date: *difference* between the earliest and latest date, *earliest* date and *latest* date.
Categorical: Contains a fixed set of unordered values. A category is not 'better' or 'greater' than another category, hence only the equivalency comparison can be used. *Count*, *most frequent* and *least frequent* can also be performed.
Ordinal: This is ordered categorical data, hence in addition to the aggregation functions for the categorical values, the greater/less-than operator can be applied.

2.2 Integrating Multi-feature Aggregation Functions into ILP Rules

Analyzing and aggregating multiple features of the same table simultaneously can yield valuable information. Using multi-feature aggregation (MFA) functions, dependencies between features can be discovered which then aid in classification. For example, an increasing income of a person over time could indicate wealth. We adapt SFA predicates to MFA predicates by allowing multiple features as arguments.

Definition 4. A *multi-feature aggregation function* maps multiple lists of elements from the domains of the corresponding features to a single value.

Definition 5. A *multi-feature aggregation predicate Agg* has the form $Agg(\{input_1, \ldots input_i\}, \{conditions\}, result)$ where $\{input_1, \ldots input_i\}$ specifies the lists of feature values to be aggregated and constrained by *conditions.* The *result* is a variable referencing the result of the multi-feature aggregation function corresponding to *Agg*.

As an example, if book *B* has features *pages* and 'weeks on best seller list' (*WoBSL*) then the input to the aggregation function consists of vectors *{pages(B), WoBSL(B)}* with the selection condition *{wrote(B,A),pages(B)≤200, WoBSL(B)≥3}*. The correlation can be calculated by applying the function *corr* to *page* and *WoBSL* to get:

corr(<pages(B),WoBSL(B)>, {wrote(B,A),pages(B) ≤ 200,WoBSL(B) ≥ 3},R), R>0.5

Restrictions are then placed on the result *R* of the multi-feature aggregation, for example, requiring that the correlation be larger than 0.5. In this paper we restrict discussion to the case of aggregating two features only. However, the framework can be generalized to express aggregation of any number of features, for example to see how the *age-gender* distribution changes over time for each book sold.

For 2-dimensional feature analysis, the features are analyzed pair-wise by taking into account both feature-types. Dates and numbers can be binned and analyzed as ordinal data. The MFA functions we use are discussed below:

Numerical vs. Numerical: The slope of line of best fit, correlation, covariance or the T-test can be applied in order to show a relationship between numerical features.

Date vs. Numerical: Slope of line-of-best-fit can illustrate a temporal trend or cycle. Correlation, covariance and T-test can also be calculated by treating dates as numbers (by calculating the number of days from a certain date).

Categorical/Ordinal vs. Categorical/Ordinal: Pair-wise frequency tables can be built and analyzed to find the least and most frequent combination of values. The Chi-Square test indicates whether there is a dependency between two variables.

3 Learning Rules with Aggregation Predicates

In this section, we show how to learn rules with SFA and MFA predicates. Our algorithm, CLAMF (***CL***assification with ***A***ggregation of ***M***ultiple ***F***eatures), is an adaptation of the sequential covering algorithm and is based on the idea of the well-known CrossMine algorithm [13]. The task is to address the two class classification problem by finding rules which predict the class label of a target tuple.

3.1 Learning Rules

The building of rules is done by generating one rule at a time and refining them incrementally until some termination condition applies. When refining a rule, a method (*GetBestLit*) extends the rule by at least one literal at a time. The 'goodness' of a literal is determined via FOILGain [10].

GetBestLit employs a look-ahead strategy extending a given rule by possibly multiple literals at a time. In addition to the standard cases, we allow the extension by SFA or MFA predicates and a corresponding comparison literal. The search-space of all allowed literals is explored by recursively searching all referenced tables T_r in the rule being built. If T_r is referenced in the rule then the tuple IDs are propagated to the linked table which is then explored by recursively applying *GetBestLit* to that table.

To illustrate further our algorithm, we give the following example. Once the IDs have been propagated (Fig. 2), existential quantification (EQ), SFA, and MFA over any previously referenced table can be performed since all necessary IDs are in T_r. Using only the information in T_r, for each previously referenced table over which aggregation is to occur, a new table is created (Fig. 3) and scanned for the best combination of one EQ, SFA, or MFA predicate and a threshold. The overall highest FOILGain is selected and the search for the next best literal is restarted until there are insufficiently many tuples left which are not covered by a rule.

3.2 Extending TupleID Propagation for Aggregation

CrossMine [13] introduced the concept of TupleID propagation to efficiently mine MR classification rules using the existential operator. The propagation appends to each tuple of a non-target table the IDs and class labels of tuples in target table T_t that are related to it. The following example illustrates how TupleID, in the context of aggregation, cannot do what we need. Using our running example, starting with the target table **`Author`**, the data is propagated to **`Book`**. During the next iteration of propagations, **`Book`** becomes the source table for the propagation and the related table **`Customer`** becomes the destination. For each tuple in **`Customer`**, the related tuples in **`Book`** are determined and the IDs, along with the class labels of **`Author`** are appended to **`Customer`**. The result is similar to Fig. 2 but without BookID. Aggregation of authors can now be performed to find the '*total number of unique customers each author has sold to*'. Aggregating **`Customer`** over books to determine the '*number of customers each book sold to*' however is not possible since there is no information from **`Book`** in **`Customer`**. Due to this, TupleID propagation does not allow for aggregation over previously referenced tables but only the target table.

Customer					
CustID	Name	…	AuthorID	Class	BookID
1	Jennifer Smith	…	1, 5, 3	Y, Y, Y	1, 1, 3
2	Bob Doe	…	2, 4	N, Y	4, 2
3	Janet Cats	…	1, 5	Y, Y	1, 1
4	Terry Wolfe	…	4	Y	2

Fig. 2. Result of our propagation. Aggregation over *Books* can be done.

Aggregated Customer Table							
BookID	Class	Age			…	Gender	
		Min	Max	Avg	…	Most Common	…
1	Y, Y	20	30	25	…	F	…
2	Y	18	40	29	…	M	…
3	Y	20	20	20	…	M	…

Fig. 3. Aggregated table can now be analyzed for literal selection

To allow for aggregation over all tables we extend TupleID for aggregation over any previously referenced table in the rule being built. We do this by iteratively propagating not just the IDs of T_t, but the IDs of all the previously referenced tables as well (represented as a comma-delimited ordered string in Fig. 2). This allows the calculation of an aggregation involving **`Customer`** and **`Book`**, such as: '*the number of unique customers each book sold to*'. Further propagation, e.g. from **`Customer`** to **`Card`**, takes the IDs of **`Customer`** and **`Book`**, and moves it along with the AuthorID and class label. The resulting table would contain all information required to aggregate over authors, books and customers. We are now able to detect, for example, '*what is the most frequent credit-card type used to purchase a science fiction book*'.

In order to perform aggregation over any previously referenced table, we simply summarize by the ID of that table and can immediately determine which tuples of the current table are associated to each ID. For example, from Fig. 2, book 2 is associated to customers 2 and 4 and book 3 is associated to customer 1. Aggregation can be performed by aggregating over each BookID. The result is shown in Fig. 3. FOILGain is then applied to this table to determine the best combination of aggregation function, feature(s) and threshold value.

4 Experimental Results

We performed extensive experiments on the Financial and Medical datasets from the PKDD'99[1] and the Hepatitis dataset from the PKDD'02[2] Discovery Challenges. The main objective of our experiments was to demonstrate the gain in classifier performance achievable by SFA and MFA. Three classifiers were built per dataset: the first classifier (EQ) used only the existential quantifier, the second (SFA) used single-feature aggregation and EQ, and the third (MFA) used multi-feature aggregation, SFA and EQ. The experiments evaluated the classification accuracy of MFA against SFA and EQ. 5-fold cross validation was performed to evaluate the classifier performance. The results are presented in Fig. 4.

PKDD'99 Financial Dataset. The dataset contains 606 successful and 76 not successful loans along with their information and transactions. Bad loans were chosen as the target class and the *transaction* table was pruned by removing all transactions which occurred after a loan was approved. EQ achieved very poor precision, between 20% and 45%, similar to [5]. Adding SFA resulted in an increase to 90% in the best case, 60% in the worst. With MFA the precision reached 100% and was still above 90% in the worst case. This gain was not at the expense of recall, as can be seen in the F-Measure results. A sample rule that was found is:

R_1: loan(L,'bad') ← loan(L), max(A,{transaction(T),trans_of_loan(T,L),amount(A,T)},M),M<99.6,
correlation({B,D},{transaction(T),trans_of_loan(T,L),balance(B,T),date(D,T)},corr), corr<0.143.

According to R_1, a loan is bad if the maximum transaction amount corresponding to this loan is smaller than 99.6 (the average transaction amount in the entire dataset is 9,101), and the correlation of the balance and date is less than 0.143. Intuitively, this

[1] http://lisp.vse.cz/pkdd99/
[2] http://lisp.vse.cz/challenge/ecmlpkdd2002/

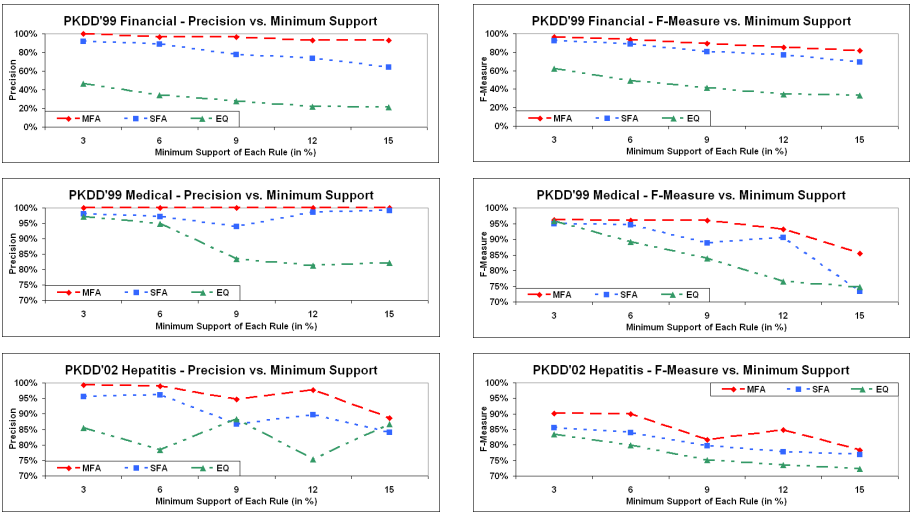

Fig. 4. Precision and F-Measure for the three datasets from PKDD'99 and PKDD'02

indicates that the payments made on the loan are very small and there is no correlation between the balance of the transaction and the date.

PKDD'99 Medical Dataset. Classification was done on the 41 instances of patients with Thrombosis. 56,197 exams, dates, and results, with each exam having 33 numerical results, allowed MFA to perform numerous multi-feature analyses to detect relationships between different features. MFA classification precision was 100% with SFA being above 94% for all minimum support values. EQ was quite competitive for small minimum support values, but for higher values EQ dropped to 82% precision. The gain in classification precision of MFA was not at the expense of recall since MFA consistently also had the highest F-Measure. As anecdotal evidence of the meaningfulness of the rules, for example, the following rule was discovered:

$$R_2: \text{thrombosis(S,'bad')} \leftarrow \text{patient(P),slope(\{T,H\},\{exams(E,P),TBIL(T,E),HCT(H,E)\},S),}$$
$$\text{S<-4.5,correlation(\{D,B\},\{exams(E,P),RBC(B,E),date(D,E)\},C),C<0.91}$$

R_2 states that a person will have thrombosis if the relationship between the results of the TBIL and HCT tests is negatively proportional, and the RBC test values and date are not very highly correlated.

PKDD'02 Hepatitis Dataset. The classifier was built on 206 instances of Hepatitis B (contrasting them against 484 cases of Hepatitis C). The *inhospital* table had to be preprocessed such that each unique test was in a column and all tests for a patient on a given date were in a single tuple. The resulting *inhospital* table had 12,614 tuples and 120 features. Due to the transformation, a lot of columns contained insignificant numbers of non-NULL values, and were removed, leaving only 25 features for classification. MFA consistently outperformed SFA and EQ both in precision and F-Measure. Precision was up to 8% higher than SFA and up to 20% higher than EQ.

5 Conclusions

In this paper we presented a novel method for classification of MR data using an aggregation-based ILP approach. We proposed an ILP framework for representing rules with multi-feature aggregation predicates. The different types of aggregations available for different feature-types, and their combinations, were discussed. For efficient classifier construction, we extended TupleID propagation so that it allows for aggregation over related tables and not only the target table. Experiments on three real-world datasets showed substantial gains in precision and F-Measure compared to existing approaches. Anecdotal evidence was provided to illustrate the rule meaning.

In temporal databases, classification with multi-feature aggregation could provide very interesting rules that are much more meaningful to the end-user by allowing for temporal trends. Another direction is to investigate spatial classification where dependencies between features are prevalent since the spatial relationship of objects impacts their mutual influences. We plan to explore these important applications.

References

1. Appice, A., Ceci, M., Lanza, A.: Discovery of spatial association rules in geo-referenced census data: a relational mining approach. Intelligent Data Analysis 7, 541–566 (2003)
2. Atramentov, A., Leiva, H., Honavar, V.: A multi-relational decision tree learning algorithm -implementation and experiments. In: Horváth, T., Yamamoto, A. (eds.) ILP 2003. LNCS (LNAI), vol. 2835, Springer, Heidelberg (2003)
3. De Raedt, L., Lavrac, N.: Multiple Literal Learning in two Inductive Logic Programming Settings. Journal on Pure and Applied Logic (1996)
4. Dzeroski, S.: Multi-relational data mining: an introduction. SIGKDD Explorations 5(1) (2003)
5. Keim, D.A., Wawryniuk, M.: Identifying Most Predictive Items. In: Workshop on Pattern Representation and Management, Heraklion, Hellas (2004)
6. Knobbe, A.J., Siebes, A., Marseille, B.: Involving Aggregate Functions in Multi-Relational Search. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) PKDD 2002. LNCS (LNAI), vol. 2431, Springer, Heidelberg (2002)
7. Lu, Q., Getoor, L.: Link-based Text Classification. In: Proceedings of ICML (2003)
8. Neville, J., Jensen, D., Friedland, L., Hay, M.: Learning relational probability trees. In: Proceedings of KDD (2003)
9. Perlich, C., Provost, F.: Aggregation-Based Feature Invention and Relational Concept Classes. In: Proceedings of KDD (2003)
10. Quinlan, J.R., Cameron-Jones, R.M.: FOIL: – A midterm report. In: Brazdil, P.B. (ed.) Machine Learning: ECML-93. LNCS, vol. 667, Springer, Heidelberg (1993)
11. Taskar, B., Segal, E., Koller, D.: Probabilistic classification and clustering in relational data. In: Proceedings IJCAI (2001)
12. Vens, C., Van Assche, A., Blockeel, H., Dzeroski, S.: First order random forests with complex aggregates. In: Camacho, R., King, R., Srinivasan, A. (eds.) ILP 2004. LNCS (LNAI), vol. 3194, Springer, Heidelberg (2004)
13. Yin, X., Han, J., Yang, J., Yu, P.S.: CrossMine: Efficient Classification Across Multiple Database Relations. In: Proceedings of ICDE (2004)