Aggregation-Based Feature Invention and Relational Concept Classes

Claudia Perlich and Foster Provost

Department of Information, Operations and Management Sciences

Leonard N. Stern School of Business, New York University

44 West 4th Street, New York, NY 10012

cperlich@stern.nyu.edu, fprovost@stern.nyu.edu

Center for Digital Economy Research
Stern School of Business
Working Paper IS-03-03

Aggregation-Based Feature Invention and Relational Concept Classes

Claudia Perlich
Stern School of Business
44 West 4th Street
New York, NY 10012
cperlich@stern.nyu.edu

Foster Provost
Stern School of Business
44 West 4th Street
New York, NY 10012
fprovost@stern.nyu.edu

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*Learning*

Keywords

Relational Learning, Aggregation, Feature Invention

ABSTRACT

Due to interest in social and economic networks, relational modeling is attracting increasing attention. The field of relational data mining/learning, which traditionally was dominated by logic-based approaches, has recently been extended by adapting learning methods such as naive Bayes, Baysian networks and decision trees to relational tasks. One aspect inherent to all methods of model induction from relational data is the construction of features through the aggregation of sets. The theoretical part of this work (1) presents an ontology of relational concepts of increasing complexity, (2) derives classes of aggregation operators that are needed to learn these concepts, and (3) classifies relational domains based on relational schema characteristics such as cardinality. We then present a new class of aggregation functions, ones that are particularly well suited for relational classification and class probability estimation. The empirical part of this paper demonstrates on real domain the effects on the system performance of different aggregation methods on different relational concepts. The results suggest that more complex aggregation methods can significantly increase generalization performance and that, in particular, task-specific aggregation can simplify relational prediction tasks into well-understood propositional learning problems.

1. MOTIVATION AND INTRODUCTION

Relational learning has attracted significant attention because of the expressive power of relational models and the techniques' ability to incorporate relational background knowledge. The field of relational learning was originally dominated by Inductive Logic Programming (ILP)[15]. New approaches include distance-based methods[7], binary propositionalization[10], SQL-based numeric aggregation[8], and upgraded propositional learners such as rule learners [11], Structural Logistic Regression [16], Relational Decision Trees [5] and Probabilistic Relational Models (PRM)[9]. One essential component that has significant impact on generalization performance for domains with important one-to-n relationships is the aggregation of sets into values. However, with the exception of [8], aggregation has received little direct attention. Aggregation methods can be characterized along a number of dimensions including the underlying calculus (numeric or logical), the cardinality of the relation in question, and the complexity of the object being aggregated (atomic values or feature vectors).

The objective of this paper is to shed new light on the role of aggregation methods in relational learning. We develop a hierarchy of types of relational learning problems, and evaluate relational learners on a business domain in order to draw conclusions about the applicability and performance of different aggregation operators. For this paper we have chosen the relational database formalism for expressing relational data and concepts. However the ideas and methods carry over directly to learning from a graph or first-order-logic representation.

The paper is organized as follows: Section 2 presents an ontology of increasingly more complex relational concepts, and discusses the complexity of domains and the relationship between domain properties and concept complexity. Section 3 presents an overview over existing aggregation methods, their limitations, and the systems that use them. We present in addition a novel target-dependent aggregation method, to begin to flesh out the higher levels of the ontology. The subsequent empirical study in section 4 compares a number of aggregation methods on a relational business domain and shows evidence of the superiority of more complex methods (viz., target-dependent set aggregation). We conclude with suggestions for future work with particular focus on more complex aggregation methods than are currently used.

2. CONCEPT ONTOLOGY

Consider a predictive (rather than clustering or unsupervised) relational learning task as finding a mapping

$M : (T, RDB) \rightarrow y$ where T is the target table,[1] including a target variable y (either numeric for a regression task or categorical for classification) and RDB is a relational database containing additional tables of background knowledge. The relational database RDB can vary from simple to complex, in terms of the number of tables, the number of relationships between tables through shared categorical variables (*keys*), and the cardinality of those relationships (one-to-one, one-to-n, or n-to-m).

Similar to the complexity of the RDB, relational concepts can have various complexities. In this paper we adopt the view that a relational concept $(X, RDB) \rightarrow y$ is a function F of aggregations A of attributes of objects that are related to the target case through keys. For example, given the target table T = Customer(CustomerId,Gender,Age) and the RDB table Transaction(CustomerId,Date,Price,ProductId), which are related through the key CustomerId that appears in both tables, one concept of an ActiveCustomer can be expressed using COUNT for aggregation, JOIN on CustomerId to establish the relationship between Customer and Transaction, and a comparison COUNT$\leq$20 as the function F. More generally, the complexity of a relational concept is determined by

- the complexity of the relationships (e.g. cardinalities),
- the complexity of the aggregation operator A,
- and the complexity of F.

The complexity of the relationships is determined by the domain and the prediction task. Standard machine learning methods (not to mention relational learning methods) learn relatively complex functions F. The complexity of the aggregation, however, has received relatively little treatment.

Definition: A *simple aggregation* A_1 is a mapping $\star \times 1 \rightarrow a$ that takes as input a single bag of variable size ($\star$) of atomic (1) values (either numerical or categorical).

Examples of simple aggregation operations for numeric values are the mean and the maximum. Typical aggregates of categorical values are the most common value or the count of the most common value.

Definition: A *multidimensional aggregation* A_n is a mapping $\star \times N \rightarrow a$ that takes as input a bag of objects in the form of feature vectors $(a_1, ..., a_n)$. The number of objects $(a_1, ..., a_n)$ in the bag is variable, denoted by $(\star)$, and can in particular be zero.

The important difference between using multiple simple aggregations and a multidimensional aggregation is that the attributes in the vectors are not independent and have to be aggregated together. "The total amount spent over the last two weeks" is a multidimensional aggregation, since both Date and Price have to be aggregated jointly. However, "the number of purchases in the last two weeks" can be expressed with a simple aggregation using count over the attribute Date.

Definition: A *multi-type aggregation* $A_{n,m}$ is a mapping $\star \times N, \star \times M \rightarrow a$ that takes as inputs two bags of objects from different tables. The objects in bag one have a feature vector of length n, the objects in bag two of length m.

Consider that the RDB also contains the table ReturnedItems(CustomerId,ProductId) and we want to find the most recent date on which a customer bought a product that was commonly returned before (by other customers). This aggregation has to incorporate two sets: the products bought by the customer and the products commonly (for example more than 20 times) returned. For this example $A_{n,1}$ would be sufficient assuming that the products-returned-before are one-dimensional.

Given these definitions, we now can present an ontology of relational concept classes, which are partially ordered by their expressive power (their complexity). A concept class C_A is considered more complex than class C_B if any concept c_b in C_B can be expressed in C_A. We will assume a target table T with target column y, and background tables B, C, and D that are related to T and potentially to each other via keys. A lowercase expression t denotes one row in a table T. Objects t in T and b in B are related by keys: k_{T_i,B_j} appears in T as column i and in B as column j, and is commonly a categorical variable with a large number of possible values. The operator $T \overline{\frac{}{t_i=b_j,1:n}} B_{1,...,f}$ denotes a database join under the condition $t_i = b_j$ and the subsequent selection of columns 1,...,f from B. The notation 1:n (join cardinality) declares that for every value t_i there can be one or more rows in B fulfilling the equality condition $t_i = b_j$. Given the complexity of notation we will keep the simple form of single joins; however note that it is straightforward to extend the hierarchy replacing $T \overline{\frac{}{t_i=b_j,1:n}} B_{1,...,f}$ by a chain of such operators joining across multiple tables $T \overline{\frac{}{t_i=b_j,1:n}} B \overline{\frac{}{b_k=c_g,1:n}} C_{1,...,f}$.

So, the following list shows relational concept classes in order of increasing complexity. [2]

1. **Propositional:**
   $y = F(t)$ or $y = F(t, F(T \overline{\frac{}{t_i=b_j,1:1}} B_{1,...,h}))$
   A join of cardinality 1:1 returns exactly one object (feature vector) for each object in T. There is no need for aggregation and the features 1,...,h can be concatenated directly to the feature vector in T. A typical case is a Customer table T and a Demographics table D that contains additional information for each customer.
2. **Abstraction hierarchy:**
   $y = F(t, F(T \overline{\frac{}{t_i=b_j,n:1}} B_{1,...,h}))$
   If the cardinality of a join is n:1 there will be exactly one observation in B for each observation in T

[1] T is a table of traditional feature vectors, including categorical variables possibly with large numbers of possible values.

[2] In this paper we assume F to be deterministic but the observations may be noisy: $\hat{y} = F(x) + \epsilon$.

and the features 1,...,h can be concatenated to the feature vector. For example, B might be a Product(ProductId,ProductType) table where ProductType is a classification of particular product as book or computer.

3. **Bag of attribute values:**
$y = F(t, A_1(T_{\overline{t_i=b_j,1:n}} B_h))$
The least complex case that requires aggregation is a one-to-n relationship selecting only one attribute. The example given before of the average price of the products that a customer bought falls into this category: T is the customer table, B is the Transactions table, the key is CustomerId and h is the Price.

4. **Independent bags of values within or across tables:**
$y = F(t, A_1(T_{\overline{t_i=b_j,1:n}} B_k), \ldots, A_1(T_{\overline{t_i=b_j,1:n}} B_h))$ or
$y = F(t, A_1(T_{\overline{t_i=b_j,1:n}} B_k)), \ldots, A_1(T_{\overline{t_i=C_j,1:n}} C_h))$
The next more complex class contains cases where two bags are necessary for F, but it is possible to them independently of each other. For example, what if the learning needs to consider the proportion of products returned. The first aggregation would be the count of the products in the transaction table, and the second aggregation would be the count of products in the ReturnedItems table. Calculating the proportion would be part of the function F, not the aggregation. Note that this is not really a more complex case than the bag of attribute values, since the only difference it the use of multiple aggregations. However it is important to note that in many cases the aggregation of objects can be simplified by aggregating the object attributes independently (and clarifies the next concept class).

5. **Dependent bags within one table:**
$y = F(t, A_n(T_{\overline{t_i=b_j,1:n}} B_{l,\ldots,n}))$
The selected n attributes from table B cannot be aggregated independently of each other. In this case the aggregation A_n has to be multidimensional. For example, time series often harbor such concepts where the observation clearly is dependent on a time field, and separate aggregation would not be meaningful. The prior example of the total amount spent over the last two weeks falls into this category.

6. **Single-type, dependent bags across tables:**
$y = F(t, A_n(T_{\overline{t_i=b_j,1:n}} B_{l,\ldots,n}, \ldots, T_{\overline{t_k=b_l,1:n}} B_{l,\ldots,n}))$
In some cases it is necessary to aggregate the results of multiple joins producing one result table. An example is the average age of a person's cousins if only Parent, Brother and Sister tables are available as background knowledge. In order to reach all cousins multiple joins are necessary. However, the results of those joins can be combined into one bag and the aggregation can be done on this bag. In contrast, the *total number* of cousins is not part of this class, but rather falls in to the (simpler) bag of attribute values; the size of each join can be counted individually and the sum can be left to the function F.

7. **Multiple-type, dependent bags across tables:**
$y = F(t, A_{n,m}(T_{\overline{t_i=b_j,1:n}} B_{1,\ldots,l}, T_{\overline{t_i=C_j,1:n}} C_{1,\ldots,h}))$

Level	Characterisitics	Concept Class
1	Single table	1
2	Single table with multiple occurrences of the same key in a 1:1 self-join	1,3,5
3	Single table with multiple occurrences of the same key and a m:n relationship	1,3,5,7
4	Multiple tables with 1:1 relationships between them	1,2
5	Multiple tables with n:1 from the target table to background tables	1,2,7
6	Multiple tables with 1:n relationships from the target table to the background table and each background table has only one attribute (besides the key)	1,3,4,6
7	Multiple tables with 1:n relationship to background tables	1,3,4,5,6
8	Multiple tables with n:m relationships of objects	1,2,3,4,5,6,7

Table 1: Domain complexity and potential concept class

At times, as in the example of products that are commonly returned, it is necessary to aggregate two bags resulting from two joins into different tables. Since the two bags have different types they cannot simply be combined into one.

8. **Global graph features:**
$y = F(A_1(T_{\overline{t_i=b_j,1:n}} B_{A_*(\Omega)}))$ where
$\Omega = TC(B_{\overline{b_k=C_g,1:n}} C, \ldots, D_{\overline{d_i=b_m,1:n}} b_{1,\ldots,f})$
TC stands for transitive closure. An example of such a concept is one's ability to find a job as a function of the reputation of her advisor. Reputation is a global concept from social network analysis that may require from an aggregation operator A_* the construction of an adjacency matrix and the calculation of its Eigenvalues and Eigenvectors.

The concept of relational autocorrelation has been identified as a common property of relational domains[6]. To use relational autocorrelation for learning, one might consider it to be a special case of single-set-value concept class where the join links back to the target relation and the target variable is aggregated.[3]

Table 1 relates the complexity of a RDB based on the cardinalities of its relationships and the potential complexity of concepts embedded within it. The mapping in table 1 suggests that even fairly simple relational domains can contain very complex relational concepts.

[3]Direct circular dependence to the target on some categorical value $T_{\overline{t_i=t_1,1:n}} T_y))$ is essentially propositional learning. This join addresses the question how many cases in the training with a particular categorical value for attribute t_i are positive or negative.

3. RELATIONAL AGGREGATION

We now discuss aggregation methods and how they relate to the concept classes that can be learned by different approaches. Relational learning has taken two approaches: (1) aggregation-based feature construction/invention and subsequent model estimation or (2) direct learning of the relational mapping M. Aggregation must take place in either approach; the main difference between the two is whether the aggregation is optimized jointly with the estimation of F or whether they are performed independently. As shown for some cases in the presentation of the ontology, there are interactions between the aggregation and the function. That notwithstanding, for the remainder of this paper we focus on the aggregation operators A, assuming the existence of some strategy for the identification of related objects[4] as well as an appropriate learner for F.

Existing aggregation methods differ with respect to the underlying calculus (numeric or logic-based) and to what extent they are part of the actual learning process (e.g. whether the aggregation is dependent on task or on the target). The following sections present three common approaches to aggregation and one novel approach that combines set-distances with target-dependent aggregation.

3.1 First-order Logic

The field of relational learning was originally dominated by Inductive Logic Programming (ILP)[15] and focused on classification tasks. These first-order-logic-based approaches search for sets of clauses that identify positive examples such as:

RichCustomer(x)← Customer(X,Y,Z),
Transaction(X,V,P,W), P≤100

The prediction of an ILP concept is positive if at least one of the clauses is true for the particular case. Binary propositionalization ([19],[10]) also learns sets of (first-order) clauses, but rather than using them directly for prediction it construct binary features and then uses a traditional learner (e.g. logistic regression or decision tree induction) to learn the function F.

Both approaches use existential unification of first-order-logic clauses as aggregation mechanism. Given the tables from section 2 the example clause
Customer(X,Y,Z),Transaction(X,V,P,W),P≤100 is 1 for a particular customer X if he bought a product that cost more than USD 100. The bag of products that are related to a customer is aggregated into a single binary value (0 or 1) based on the condition P≤100. The major advantage of logic-based aggregation is its ability to address all levels of complexity as outlined in section 2, including dependent bags across tables. The task of identifying customers that bought a product that was returned by another customer who bought it after 2001 can be expressed in FOL as:
Customer(X),Transaction(X,V,P,W),ReturnedItem(Y,W),
Transaction(A,B,C,W),B≤2001

The disadvantage of logic-based aggregation is the lack of support for numeric aggregation. In particular, it is impossible to express that the product was returned more than 20 times. A clause can test whether the maximum of a numeric set is larger than a particular value but it can not estimate the mean or the cardinality of the set. [5]

3.2 Set Distances

Kirsten, Wrobel and Horwath [7] proposed a distance-based method for relational learning. The approach uses a KNN methodology to classify objects and uses a predefined relational distance metric. This metric essentially works as an aggregation of the two bags of objects related to two cases. Given two sets of objects of the same type, the distance measure calculates the minimum vector distance of all possible pairs of objects, choosing one from one bag and one from the other. The vector distance is the difference for numeric values and the edit distance for categorical values, normalized by the number of attributes. If an attribute is a key, rather than taking the edit distance the algorithm proceeds recursively and estimates the distance of all objects related to the current vector using that key.
This form of aggregation can only address identical types since the vectors have to have identical entries. Additionally all attributes of the vector are aggregated independently. It also does not allow for numeric aggregation like count and average.

3.3 Numeric Aggregation using SQL

Numeric aggregates in combination with logic-based feature construction were originally proposed by Knobbe et al ([8]). A number of relational approaches including Probabilistic Relational Model (PRM)[9] and 'upgraded propositional learners such and Relational Decision Trees [5] rely on a small set of simple (mostly SQL-based) aggregation operators such as mean, min, max, count for numerical values, proportions and most common value for categorical variables. These operators apply only to bags of single attributes and cannot express concepts that require dependent aggregation.

3.4 Target-Dependent Set Aggregation

We now describe a methodology that integrates vector distances and task-specific aggregation. We are not aware of any existing system to follow this approach, but it illustrates the higher levels of the concept-class hierarchy. The method is motivated by the observation that relational databases commonly have attributes with large numbers of possible values—and these attributes are unsuitable for learning. The methodology is easily extended to numeric values using discretization and coding numerical values as categorical dummies.

A common method to aggregate a single attribute with numerous categorical values is the selection of a subset of values that appear most often and convert them into dummy variables or counts. A bag of colors {red,green,green,red,blue,red} could be aggregated to dummies 0,1 or counts 0,3 if across all bags 'yellow and 'red were the most common and the first position stands for yellow.

We extend this approach by aggregating with reference to

[4]For example, graph traversal using foreign keys as links and tables as nodes.

[5]There have been some efforts to extend the numeric capabilities of CProgol4.4 [14].

(conditioned on) the classes of the training cases. For this we define:

Definition: Given an order (pairs of value v: index i) of all possible values of the categorical attribute BG_j, a *case vector* $CV^t_{(T_{\overline{t_i=bg_k,1:n}}BG_j)}$ at position i is equal to the number of values v $(v : i)$ in the bag returned from the join $T_{\overline{t_i=bg_k,1:1}}BG_j$ for the case t in the target table T.

As an example, the bag {red,green,green,red,blue,red} for case t under the order (yellow:1,red:2,green:3,blue:4) would result in $CV^t = (0, 3, 2, 1)$.

Definition: Given an order $(v{:}i)$ of all possible values of the attribute BG_j, a *reference vector* $RV^c_{(T_{\overline{t_i=bg_k,1:n}}BG_j)}$ under the condition c at position i is equal to the sum of values $CV^t[i]$ for all cases t for which c was true.

Definition: The *variance vector* $VV^c_{(T_{\overline{t_i=bg_k,1:n}}BG_j)}$ at position i is equal to the variance $\frac{(CV^t[i])^2}{N_c-1}$ over all case t for which c was true. N_c is the number of cases for which the condition was true.

3.4.1 Single Categorical Value

Rather than selecting values that are most common across all related objects, a target-dependent approach will select categorical values that are most commonly related to positive training cases (y=1) and analogously those that are most commonly related to negative cases. To create positive dummies, we select given $RV^{y=1}$ those values v given the order $(v : i)$ for which the $RV^{y=1}[i]$ is maximal across all entries in $RV^{y=1}$. Similarly we select those values v given the order $(v : i)$ for which the $RV^{y=0}[i]$ is maximal across all entries in $RV^{y=1}$.

A more complex, comparative approach selects categorical values that are common for one class but not common for the other. In particular we select the values I for which the absolute value of $RV^{y=1}[i] - RV^{y=0}[i]$ is maximal.

The Mahalanobis distance[12] improves over this approach by normalizing the scores by the variances before selecting the maximum:

$$|\frac{RV^{y=1}[i] - RV^{y=0}[i]}{VV^{y=1}[i] + VV^{y=0}[i]}| \quad (1)$$

In total we have five aggregates based on single categorical values grouped into three groups of increasing complexity: target independent (first row), dependent on either positive or negative reference vector (second and third row) and difference between the positive and negative reference vectors (fourth row and fifth row).

Method	Definition
MOC	$CV[i]$ where i is the index with maximum value in unconditional refercence vector RV
MOP	$CV[i]$ where i is the index with maximum value in positive refercence vector $RV^{y=1}$
MON	$CV[i]$ where i is the index with maximum value in negative refercence vector $RV^{y=0}$
MOD	$CV[i]$ where i is the index with maximum absolute value in vector $RV^{y=1} - RV^{y=0}$
MON	$CV[i]$ where i is the index with maximum absolute value in vector $\frac{RV^{y=1}[i]-RV^{y=0}[i]}{VV^{y=1}[i]+VV^{y=0}[i]}$

3.4.2 Categorical Vectors

The vector distance methodology integrates all entries in the reference vectors rather than picking only the value with the largest counts in the reference vectors. From the case vector VC and a reference vector RV we estimate four vector distances: edit distance (ED), Euclidean distance (EU), Mahalanobis distance (MA), and Cosine distance (COS). In addition we calculate for each of the four measures the difference of the vector distances to the positive and negative reference vectors:
$\cos(RV^{y=1}, CV) - \cos(RV^{y=1}, CV)$
Combining the options for distance and target conditions, we have a three-by-four matrix of vector-based aggregations.

Reference	Euclidean	Edit	Cosine	Mahalanobis
unconditional	EU	ED	COS	MA
$y = 1$	EUP	EDP	COSP	MAP
$y = 0$	EUN	EDN	COSN	MAN
$y = 1, y = 0$	EUD	EDD	COSD	MAD

3.4.3 Distance Groupings

The vector distances can be grouped similarly into three increasingly more complex groups: target independent (first row), dependent on either positive or negative reference vector (second and third row) and difference between the class distances (fourth row).

It should be noted that since these aggregations use the target to estimate features, the subsequent model can be overly optimistic about the value of the feature, which can lead to overfitting when these features are used for learning. Therefore, for the results that follow, the reference vectors, vector distances and special categorical values are estimated on 50% of the training set and the model is estimated using the other 50% of the training set.

4. EXPERIMENTAL RESULTS

In this section we present results comparing different aggregation methods on a relational learning problem concerning initial public stock offerings. We include the comparative performance of four logic-based relational learners (FOIL[18], Tilde[1],Lime[13],Progol[14]) since they provide conditional aggregation methods. The next section gives a brief overview over the methodology from which the aggregation results were produced.

4.1 Domain: Initial Public Offerings

Initial public stock offerings have a unique ticker for the firm that is selling shares of their equity. An IPO is typically headed by one or occasionally two banks and supported by a number of additional banks as underwriters. The task of the bank is to put shares on the market, to set a price, and to guarantee with their experience and reputation that the stock of the issuing firm is indeed valued correctly.

The IPO domain consists of 5 tables:

- IPO(Date,Size,Price,Ticker,Exchange,SIC,Runup)
- HEAD(Ticker,Bank)
- UNDER(Ticker,Bank)
- IND(SIC,Ind2)
- IND2(Ind2,Ind)

The last two relations, IND and IND2 are different abstraction levels of SIC classifications. For example the industry code 7372 identifies the division of "Prepackaged software". This is particular category of industry group is a particular member of the major group "Business Services" with the 2 digit code 73.

In this domain, Date, Size, Price and Runup are numerical variables; Ticker, Bank, SIC, Ind3 are keys, and Ind2 and Exchange are categorical attributes. The classification task is to classify whether the offer was (would be) made on the NASDAQ exchange.

4.2 Methods

We compared the generalization performance of 4 general approaches: target-dependent set aggregation, simple numeric aggregation, ILP, and logic-based feature construction. We also constructed two other features from the relational background data: if there is an instance of an abstraction hierarchy (a sequence of n:1 joins) we include the values directly in the feature vector (AH). We also wanted to test for (and potentially take advantage of) relational autocorrelation. Therefore, we allowed joins to go back to the target table and created an "autocorrelation" aggregation (AC) representing the proportion of linked, positive training cases (excluding the particular case in question of course).

For the evaluation of the aggregation methods we had to implement (1) an exploration strategy that finds related objects, (2) a feature selection step to reduce the number of features, and (3) a learner that finds a model to predict the target given the aggregates. We used straightforward approaches for each of these steps.

Exploration: Given a set of tables and keys, the system constructs a graph with tables as nodes and keys (Ticker, Bank, SIC, Ind2) as links between tables and executes a breadth-first search starting from the target relation over all possible exploration chains of increasing length. The exploration stops once the number of chains exceeds the stopping criteria. The second number in the size column in table 2 shows the stopping criterion (maximal number of joins) for the exploration. For each exploration chain the system executes the corresponding join and selects all attributes from the last table joined to. It then applies the aggregation methods of varying complexity to every attribute independently. The resulting values (one for every row in the target table) are appended to the original feature vector in the target table.

Feature Selection: Once the stopping criterion is met the system selects (10 times) a subset of 10 features using performance-based weighted sampling. We tried alternative methods for feature selection without much impact on the performance.

Model: C4.5[17] was used to learn the model for each of the 10 feature sets and average the result as the final prediction. The results did not change significantly using logistic regression for the modeling.

Logic-Based Feature Construction: In order to evaluate logic-based feature construction we used the ILP system FOIL[18] to learn n FOL clauses and appended the corresponding binary features to the feature vector in the target table IPO. This methodology has been applied successfully by King[20] and Populescul et al [16] to text classification.

ILP: We selected four ILP system based on availability, platform independence and diversity. FOIL[18] uses a top-down, separate-and-conquer strategy adding literals to the originally empty clause until a minimum accuracy is achieved. Tilde[1] learns a relational decision tree using FOL clauses in the nodes to split the data. Lime[13] is a top-down ILP system that uses Bayesian criteria to select literals. Progol[14] learns a set of clauses following a bottom-up approach that generalizes the training examples.

Evaluation: Generalization performance is evaluated in terms of classification accuracy and area under the receiver operating curve (ROC)[2]. Note that ILP systems only produce class labels but no probability scores. We therefore included for ILP only the accuracy. All reported results are generalization performance on a test set of size 800 averaged over 5 runs. We refrained from including the error bars in the table but included them in the figures.

4.3 Results

Table 2 shows the generalization performance of the set of aggregation methods as a function of training-set size and the number of joins allowed. The methods are grouped into four classes of increasing complexity: no feature construction (NO), target-independent set aggregation (MOC, VD, MVD), target-dependent set aggregation on either positive or negative class (MPN, VDPN, MVDPN), and target-dependent set aggregation on the difference between the positive and negative reference vectors (MD,VDD, MVDD). To help with the abbreviations (needed to make the table legible)a condensed summary of the different methods under comparison can be found in table 6. Within each class of methods, the first column presents an aggregation method that uses only single categorical aggregation, the second only vector distances, and the third both.

The best performance for each training size is highlighted in

Size	NO	MOC	VD	MVD	MPN	VDPN	MVDPN	MD	VDD	MVDD
250: 6	0.619	0.641	0.679	0.634	0.627	0.683	0.671	0.635	0.675	0.69
250: 9	0.619	0.685	0.665	0.665	0.664	0.685	0.697	0.695	0.682	**0.703**
250:12	0.619	0.674	0.655	*0.706*	0.675	*0.714*	0.694	0.659	0.697	**0.703**
500: 6	0.635	0.663	0.674	0.679	0.674	0.679	0.685	0.675	0.711	**0.741**
500: 9	0.635	0.706	0.686	0.684	0.692	0.705	*0.721*	0.725	0.697	0.737
500:12	0.635	0.689	0.689	*0.71*	0.706	0.707	0.696	0.711	**0.741**	0.739
1000: 6	0.671	0.677	0.691	0.685	0.667	0.717	0.709	0.702	0.713	0.747
1000: 9	0.671	0.705	*0.71*	0.688	0.715	*0.745*	*0.745*	0.735	0.747	0.747
1000:12	0.671	0.702	0.705	0.708	0.711	0.723	0.727	0.715	**0.767**	0.759
2000: 6	0.699	0.675	0.689	0.681	0.667	0.709	0.729	0.691	0.73	0.758
2000: 9	0.699	0.729	0.69	0.719	0.731	0.728	*0.76*	0.731	0.753	0.764
2000:12	0.699	0.715	0.709	*0.73*	0.718	0.733	0.723	0.72	**0.779**	0.758

Table 2: Classification accuracy of set aggregation methods grouped by complexity

bold, and the best performance for each of the complexity classes in italics. The results show that as the complexity of the aggregation method increases, the performance increases as well. The best performance within a block is always one of the two aggregations including vector distances and using only single categorical values is almost always outperformed by vector-distance aggregation. Increasing the exploration depth (number of joins) improves performance in most cases, however the marginal effect decreases. Specifically, the increase in performance moving from 6 joins to 9 is larger than moving from 9 to 12 joins. In some cases moving from 9 to 12 joins hurts the performance for two reasons: (1) the longer the chain that relates objects to a target case, the further away the objects and the less relevant they are; (2) since features are constructed from every join, the number of features increases linearly in the number of joins and the feature selection becomes less effective due to multiple comparison problems[4].

Figure 1 shows learning curves for classification accuracy, including error bars of ± one standard deviation for the experiments exploring 12 joins. The learning curves show that increasing the training-set size always improves the generalization performance. The graph also highlights the different performance levels of the 4 classes. The higher the complexity of the aggregation class, the higher the performance. In addition, the most complex aggregation (VDD) has the smallest variance of the four contrasted methods.

To compare with the results presented in Table 2, Table 3 presents the results for methods that are independent of the number of joins: abstraction hierarchies (AH) in the table IND2 and IND, the four ILP systems FOIL, Tilde, Lime, and Progol, relational autocorrelation (AC), and logic-based feature construction (LF).

These results suggest that there is significant degree of autocorrelation in this domain. AC outperforms all methods in this table only falling short of the best set-aggregation method MVDD in Table 2. Including the values of the abstraction hierarchy improves slightly over no relational background knowledge, but cannot compete with target-based set aggregation.

The two ILP systems FOIL and Tilde are still competitive for small datasets but for larger training sets fall short of

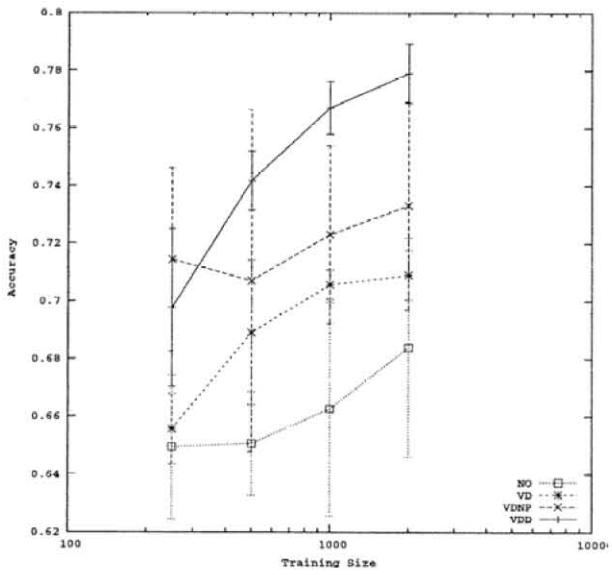

Figure 1: Learning curves: accuracy as a function of training-set size for NO, VD, VDNP, and VDD

simply using no relational background knowledge. There are three potential reasons for the low performance of the logic-based methods: (1) the task is noisy and the search mechanism within the system is overly sensitive to noise; (2) ILP systems are not optimized for numeric values, and/or (3) the relational domain properties (e.g. cardinality of the relationships) are not suitable for the particular systems. Logic-based systems can be used on simple feature-vector domains and have (on those domains) the same expressive power as a decision tree or a rule learner. However doing worse than C4.5 on the mostly numerical feature vectors suggests that the search strategy itself is not optimal for this task or that the regularization mechanism is insufficient and the systems overfit.

The low performance of LF is caused entirely by overfitting of the training data since it containes, in addition to the binary features, the original attribtues from the target table IPO used by NO. The binary features are learned from the

Size	NO	AH	FOIL	Tilde	Lime	Progol	AC	LF
250	0.649	0.641	0.645	0.646	0.568	0.594	0.73	0.59
500	0.65	0.665	0.664	0.628	0.563	0.558	0.719	0.643
1000	0.662	0.701	0.658	0.63	0.53	0.53	0.724	0.638
2000	0.681	0.711	0.671	0.65	0.51	0.541	0.753	0.641

Table 3: Classification accuracy of methods independent of join depth

training set by optimizing classification performance. They are therefore very predictive on the training set and the decision tree overestimates their predictive performance.

The results for probability estimation (reported in Table 4) are similar to the results on accuracy. The most complex aggregation methods (MVDD or VDD) outperform the other methods and the performances increase in training size. Figure 2 shows the learning curves of NO,VD,VDPN, and VDD including error bars of ± one standard deviation for 12 joins.

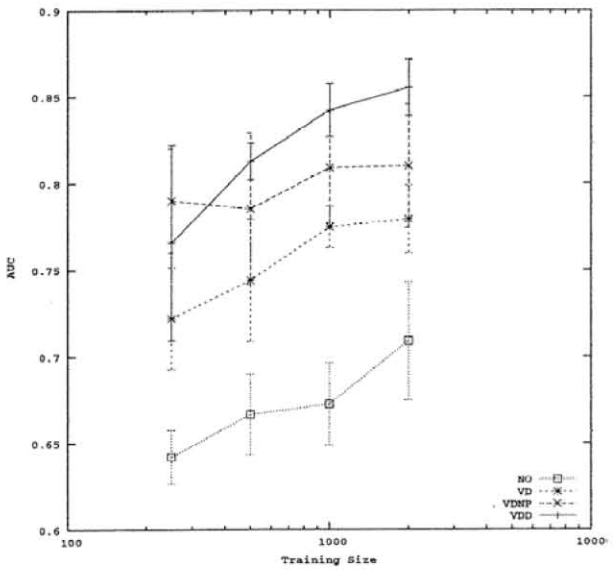

Figure 2: Learning curves: AUC as a function of training-set size for NO, VD, VDNP, and VDD

Figure 3 shows the ROC curves for NO, MVD, MVDNP, and MVDD exploring 12 joins. MVDD and MVDNP present an interesting case where the ROC curves are crossing. MVDNP is better for high thresholds whereas MVDD performs better for lower thresholds.

Analysing the probability estimation performances of methods that are independent of join depth in table 4.3 shows that the autocorrelation aggregator (AC) performs very well and almost reaches the performance of MVDD. Abstraction hierarchies (AH) are not as useful for probability estimation as they were for classification. Note, that ILP systems only predict a class label and therefore do not appear in the table.

5. CONCLUSION AND FUTURE WORK

The primary contribution of this work is the first detailed look at aggregation for relational learning. Along with search

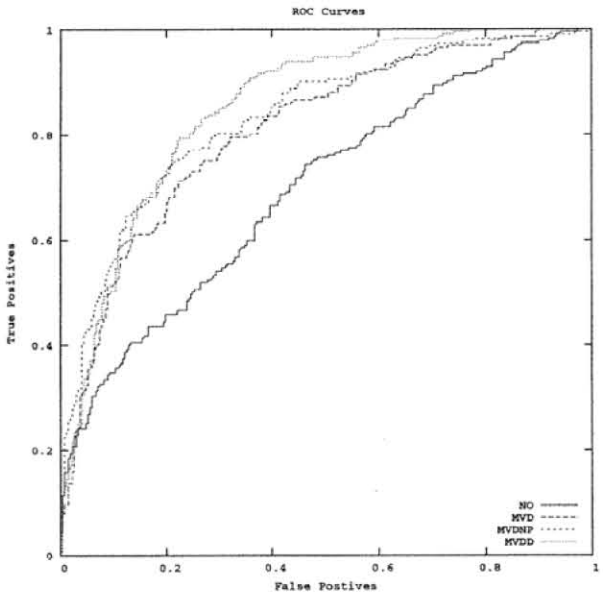

Figure 3: ROC curves for NO, MVD, MVDNP and MVDD

Size	No	AH	AC	LF
250	0.642	0.63	0.79	0.626
500	0.666	0.673	0.814	0.694
1000	0.672	0.699	0.821	0.703
2000	0.709	0.714	0.838	0.702

Table 5: Probability estimation using AUC of methods independent of join depth

through the relationship graph, aggregation is a major component of any relational learning method. We have shown, through the ontology, that with respect to aggregation there are various classes of relational learning problems, and that problems with high aggregation complexity can be deceptively simple in description.

We have shown that looking carefully at aggregation for relational learning creates a considerable design space for relational feature construction (either separately from learning or internally to a learning program). We are not aware of any learning program that considers even a small fraction of these aggregation operators, nor any that uses the more successful, target-dependent set aggregations.

Although quite suggestive, the generalizability of our positive findings (in favor of the more complex aggregators)

Size	NO	MOC	VD	MVD	MPN	VDPN	MVDPN	MD	VDD	MVDD
250: 6	0.642	0.697	0.717	0.691	0.672	0.748	0.716	0.68	0.729	0.734
250: 9	0.642	0.707	0.711	0.74	0.725	0.756	0.761	0.749	0.75	**0.764**
250:12	0.642	0.729	0.722	*0.755*	0.715	*0.79*	0.74	0.713	0.763	0.76
500: 6	0.666	0.702	0.738	0.741	0.72	0.746	0.739	0.75	0.774	0.79
500: 9	0.666	0.775	0.753	0.757	0.758	0.77	*0.802*	0.796	0.775	**0.821**
500:12	0.666	0.741	0.744	*0.787*	0.775	0.785	0.76	0.792	0.812	0.812
1000: 6	0.672	0.743	0.754	0.749	0.735	0.793	0.797	0.767	0.788	0.802
1000: 9	0.672	0.765	0.768	0.763	0.787	0.808	*0.825*	0.797	0.818	0.826
1000:12	0.672	0.778	0.774	*0.781*	0.78	0.809	0.797	0.793	**0.842**	0.829
2000: 6	0.709	0.727	0.744	0.752	0.732	0.795	0.796	0.787	0.794	0.824
2000: 9	0.709	0.785	0.772	0.781	0.807	0.805	*0.835*	0.799	0.832	0.838
2000:12	0.709	0.791	0.779	*0.801*	0.79	0.81	0.788	0.798	**0.855**	0.836

Table 4: Probability estimation performance (AUC) for set aggregation methods, grouped by complexity

Method Name	Description
NO	No feature construction, only the attribtues in the IPO table
MOC	Attributes in IPO table and counts of most common categoricals (MOC)
VD	Attributes in the IPO table and vector distances EU, ED, COS, MA to unconditional reference vector
MVD	Attributes in IPO table, most common categoricals and unconditional vector distances (EU, ED, COS, MA)
MPN	Attributes in IPO table and counts of most common positive (MOP) and negative (MON) categoricals
VDPN	Attributes in the IPO table and vector distances to positive and negative reference vectors (EUP, EUN, EDP, EDN, COSP, COSN, MAP, MAN)
MVDPN	Attributes in the IPO table, most common positive (MOP) and negative (MON) categoricals, and vector distances to positive and negative reference vectors (EUP, EUN, EDP, EDN, COSP, COSN, MAP, MAN)
MD	Attributes in IPO table and counts of most common discriminative categoricals (MOD, MOM)
VDD	Attributes in the IPO table and differences of the vector distances to positive and negative reference vectors (EUD, EDD, COSD, MAD)
MVDD	Attributes in the IPO table, and counts of most common discriminative categoricals (MOD, MOM), and differences of the vector distances to positive and negative reference vectors (EUD, EDD, COSD, MAD)
AH	Attribtues in IPO and attribute Ind in table Ind2 related throght abstraction hierarchy
AC	Attributes in IPO and proportion of positive training cases (excluding the particular case) that a case was related to
LF	Logic-based features extracted from the clauses learned by FOIL
FOIL	ILP system
Tilde	ILP system
Lime	ILP system
Progol	ILP system

Table 6: Method description

is limited due to the focus on one particular domain and the limited maximum training size of 2000. Clearly there is at least one learning task where existing aggregation approaches are not adequate. Future work includes extending these experiments to multiple domains with different relational characteristics. Within the scope of the IPO domain the empirical results demonstrate that aggregation operators of higher complexity can significantly improve the generalization performance of relational learners. The best methods (VDD, MVDD) use target-dependent vector-distance aggregators (that transform the relational task into a conventional feature-vector representation that allows the use of conventional learning methods). An advantage of this transformation-based approach is its general applicability to regression, classification, and probability estimation tasks.

Our results furthermore show that for the same level of performance, increased aggregation complexity can trade off exploration depth. This is an important point since the size of the space increases exponentially in the search depth if the relations have a one-to-n or m-to-n cardinality. Scalability of relational learning is still an important research topic in relational learning[3].

The presented aggregation methods are certainly not complete. Our findings motivate further exploration of potential aggregation methods. In particular there is still an open issue of numeric multidimensional and multi-type aggregation. Another open issue is the joint optimization of aggregation and model estimation. (Rather than treating them separately, as we have done.)

More generally, this work highlights that existing approaches to relational classification can show major performance differences. The field of relational learning still needs to develop a better understanding of why certain methods outperform on certain domains.

6. REFERENCES

[1] H. Blockeel and L. DeRaedt. Top-down induction of logical decision trees, 1997.

[2] A. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. In *Pattern Recognition*, volume 30(7), pages 1145–1159, 1997.

[3] J. Fuernkranz. Dimensionality reduction in ilp: A call to arms, 1998.

[4] D. Jensen and P. Cohen. Multiple comparisons in induction algorithms, 2000.

[5] D. Jensen and J. Neville. Data mining in social networks. In *Dynamic Social Networks Modeling and Analysis*, 2002.

[6] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *19th International Conference on Machine Learning*, 2002.

[7] M. Kirsten, S. Wrobel, and T. Horvath. Distance based approaches to relational learning and clustering. In D. Lavrac, editor, *RDM*, pages 213–232. Springer Verlag, 200.

[8] A. Knobbe, M. D. Haas, and A. Siebes. Propositionalisation and aggregates. In *LNAI*, volume 2168, pages 277–288, 2001.

[9] D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *AAAI/IAAI*, pages 580–587, 1998.

[10] S. Kramer and P. F. N. Lavrac. *Propositionalization Approaches to Relational Data Mining*, pages 262–291. Springer-Verlag, 2001.

[11] L. D. L. De Raedt, H. Blockeel and W. V. Laer. Three companions for data mining in first order logic. In S. Dzeroski and N. Lavrac, editors, *Relational Data Mining*, pages 105–139. Springer-Verlag, 2001.

[12] P. Mahalanobis. *On the generalized distance in Satistics*, volume 12. 1936.

[13] E. McCreath. Induction in first order logic from noisy training examples and fixed example set size. In *PhD Thesis*, 1999.

[14] S. Muggleton. Cprogol4.4: a tutorial introduction.

[15] S. Muggleton and L. DeRaedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19 & 20:629–680, May 1994.

[16] A. Popescul, L. H. Ungar, S. Lawrence, and D. M. Pennock. Structural logistic regression: Combining relational and statistical learning. In S. Džcroski, L. D. Raedt, and S. Wrobel, editors, *Proceedings of the Workshop on Multi-Relational Data Mining (MRDM-2002)*, pages 130–141. University of Alberta, Edmonton, Canada, July 2002.

[17] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, Los Altos, California, 1993.

[18] J. Quinlan and R. Cameron-Jones. Foil: A midterm report. In P. Brazdil, editor, *Proceedings of the 6th European Conference on Machine Learning*, volume 667, pages 3–20. Springer-Verlag, 1993.

[19] B. P. S. Kramer and C. Helma. Stochastic propositionalization of non-determinate background knowledge. In *International Workshop on Inductive Logic Programming*, pages 80–94, 1998.

[20] A. Srinivasan and R. King. Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes. In S. Muggleton, editor, *Proceedings of the 6th International Workshop on Inductive Logic Programming*, pages 352–367. Stockholm University, Royal Institute of Technology, 1996.