Aggregation-Based Feature Invention and Relational Concept Classes

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

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

ABSTRACT

Model induction from relational data requires aggregation of the values of attributes of related entities. This paper makes three contributions to the study of relational learning. (1) It presents a hierarchy of relational concepts of increasing complexity, using relational schema characteristics such as cardinality, and derives classes of aggregation operators that are needed to learn these concepts. (2) Expanding one level of the hierarchy, it introduces new aggregation operators that model the distributions of the values to be aggregated and (for classification problems) the differences in these distributions by class. (3) It demonstrates empirically on a noisy business domain that more-complex aggregation methods can increase generalization performance. Constructing features using target-dependent aggregations can transform relational prediction tasks so that well-understood feature-vector-based modeling algorithms can be applied successfully.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: [Learning]

Keywords

Relational Learning, Aggregation, Feature Construction, Constructive Induction, Propositionalization

1. MOTIVATION AND INTRODUCTION

The Internet, the Web, and social networks such as terrorist groups have raised awareness of the need to analyse relational data. In addition, business data routinely are stored in relational databases. Manually transforming these data into the feature-vector format required by most modeling approaches is time-consuming, and there is little guidance for producing effective transformations. Relational learning is attractive because of the expressive power of relational models and the ability of the learning methods to incorporate relational background knowledge.

Until recently, relational learning research has been dominated by Inductive Logic Programming (ILP) [18]. Other approaches include distance-based methods [9], binary propositionalization [12], simple numeric aggregation [10], and upgraded propositional learners such as rule learners [4], Structural Logistic Regression [19], Relational Decision Trees [7] and Probabilistic Relational Models [11]. The aggregation of bags (multisets) of related objects into single attributes is an essential component of relational model induction, and has significant impact on generalization performance for domains with important 1-to-n relationships. However, aggregation has received little direct attention [10]. Aggregation methods can be characterized along a number of dimensions including the underlying calculus (numeric or logical), the cardinality of the bags, and the complexity of the objects being aggregated (atomic values or feature vectors, single-type or multi-type objects).

The objective of this paper is to shed new light on the role of aggregation methods in relational learning. We present a hierarchy of classes of relational concepts requiring increasingly more-complex aggregations; different aggregation operators are appropriate for different classes. We introduce novel target-dependent aggregation methods. We also evaluate relational learners on a noisy business domain and draw conclusions about the applicability and performance of different aggregation operators—and show evidence of the superiority of more-complex methods (viz., the target-dependent aggregations). For this paper we have chosen the relational database formalism for expressing relational data and concepts. The ideas and methods carry over directly to learning from graph or first-order-logic representations.

The paper is organized as follows. Section 2 presents the hierarchy of relational concepts and discusses the relationship between domain properties and concept complexity. Section 3 presents an overview of existing aggregation methods, their limitations, and systems that apply them. Here we also present the novel target-dependent aggregation methods. The subsequent empirical study in section 4 compares the different aggregation methods on a relational business domain. We conclude with a discussion of implications, limitations, and future work.

SIGKDD '03, August 24-27, 2003, Washington, DC, USA

2. RELATIONAL CONCEPT HIERARCHY

We consider predictive (rather than clustering or unsupervised) relational learning tasks, i.e., finding a mapping $M : (t, RDB) \rightarrow y$ where t is a row of the target table T,[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 related background knowledge. Figure 1 shows a simple example of a relational database schema with four tables, the target table **Customer** with target attribute y and the background tables **Transaction**, **ReturnedItems** and **Products**, related through the keys CustomerId and ProductId. We will use this example to illustrate the examples in the following sections. The database RDB can vary from simple

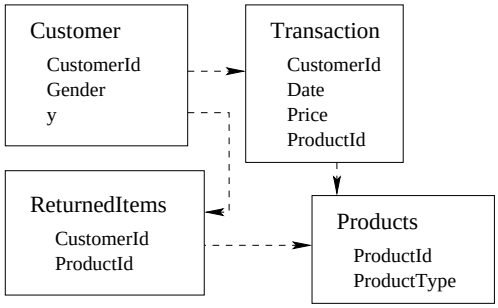

Figure 1: Transaction database

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 (1-to-1, 1-to-n, or n-to-m).

Relational concepts also have various complexities. In this paper we adopt the view that a relational concept is a function Φ including as input t and a fixed number of aggregates of objects that are related to the target case through keys. In this paper we assume Φ to be deterministic given a vector of k aggregates, but the target observations may be noisy. Formally, $y = \Phi(t, \Psi(RDB)) + \epsilon$, where aggregation operation Ψ is a mapping $RDB \rightarrow (\psi_0, \psi_1, \ldots, \psi_k)$ and ϵ is a noise term (which for clarity we will ignore in presenting the hierarchy).

The complexity of a relational concept is determined by:

- the complexity of the relationships (e.g., their cardinalities),
- the complexity of the aggregation function Ψ,
- and the complexity of the function Φ.[2]

The complexity of the relationships is determined by the domain and the prediction task. The relative complexity of different functions Φ is comparatively well understood (and we have methods capable of learning very complex functions Φ from feature vectors). The complexity of aggregations, however, has received comparatively little treatment. Consider three levels of aggregation complexity:

A *simple aggregation* Ψ^s is a mapping from a bag of zero or more atomic values to a categorical or numerical value ψ. Examples of simple aggregation operations for numeric values are count, mean, and maximum. Examples for categorical values are the most common value (mode) and the count of the most common value.

A *multi-dimensional aggregation* Ψ^m is a mapping that takes as input a bag of zero or more objects, each with p attributes in form of a feature vector $(x_1, ..., x_p)$.

It is possible to aggregate a bag of objects with p attributes using p simple aggregation operators Ψ^s for each attribute, but in contrast to a multi-dimensional aggregation operator Ψ^m this approach implicitly assumes attribute independence. The increased complexity of Ψ^m accounts for dependencies between object attributes. A common subset of multi-dimensional aggregations can be expressed as a boolean conditioning on one attribute (selection) followed by a simple aggregation of a different attribute of all selected objects. More generally, a multi-dimensional aggregation captures any relationship between two or more attributes, for instance the slope of prices over time (explaining whether a customer is buying increasingly more-expensive products). Time-series data often harbor concepts where an important predictor value is dependent on a temporal field and independent aggregation would be meaningless.

A *multi-type aggregation* Ψ^t is a mapping that takes as input two (or more) bags of objects of different types, possibly with feature vectors of different lengths.

Some useful aggregations cannot be achieved with a multidimenisonal aggregation or any function thereof. For example, consider finding the total value of the products that a customer has returned. This aggregation incorporates two bags: the products bought by the customer, along with their prices, and the products returned.

Given our definitions, we now can present a hierarchy of relational concept classes. A concept class M_2 is more complex than class M_1 if any concept δ in M_1 can be expressed in M_2 and there are concepts in M_2 that cannot be expressed in M_1. We will assume a target table T with target column y, and background tables A and B 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 through 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 $\pi_{A.(f,\ldots,l)}(T \frac{\bowtie}{t_i=a_j,(1:n)} A)$ denotes a left outer join $\bowtie$ of tables T and A under the condition $t_i = a_j$ and the subsequent projection of columns $f, \ldots, l$[3] from A. The notation 1:n (join cardinality) declares that for every value t_i there can be zero or more rows in A fulfilling the equality condition $t_i = a_j$. 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 $\pi_{A.(f,\ldots,l)}(T \frac{\bowtie}{t_i=a_j,(1:n)} A)$ by a chain of such operators joining across multiple tables

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

[2] There is a tradeoff between aggregation complexity and function complexity since parts of Φ can be integrated into Ψ. However, it is generally not possible to make up for lack of complexity in Ψ through a more complex function Φ, since aggregation involves loss of information that cannot be recovered.

[3] We denote the columns for the projection uniformly as $f, \ldots, l$; clearly the values may be different for different projections.

$\pi_{B.(f,\ldots,l)}(T \underset{t_i=a_j,(1:n)}{\bowtie} A \underset{a_k=b_g,(1:n)}{\bowtie} B)$.

The following list presents relational concept classes in order of increasing complexity. For the sake of simplicity, where possible we only include one aggregation in a function Φ. However, the distinction between classes is not in the number of aggregations needed to express a particular concept, but rather in the complexity of the most complex aggregation used.

1. Propositional:
$y = \Phi(t)$ or $y = \Phi_1(t, (\pi_{B.(f,\ldots,l)}(T \underset{t_i=b_j,(1:1)}{\bowtie} B)))$
A relationship of cardinality 1:1 returns exactly one object (feature vector) for each object in T. There is no need for aggregation and the features of objects in B can be concatenated directly to the feature vector in T. A typical case is a customer table T and a demographics table B that contains additional information for each customer.
$y = \Phi(t, (\pi_{B.(f,\ldots,l)}(T \underset{t_i=b_j,(n:1)}{\bowtie} B)))$
If the relationship has a cardinality of n:1 there will also be exactly one observation in B for each observation in T. An example is the abstraction hierarchy in the Product table (cf., Figure 1) where ProductType is an abstraction of a particular product into a class, for instance 'book'.

2. Independent attributes:
$y = \Phi_2(t, \Psi^s(\pi_{B.h}(T \underset{t_i=b_j,(1:n)}{\bowtie} B)))$
The least complex *relational* concept class requires only simple aggregations. The object of a 1-to-n relationship may have a number of attributes, each of which can be aggregated independently. For example, simple concepts like 'the average price of products bought' fall into this category. An example that requires multiple simple aggregations from different tables is 'the proportion of productions returned by the customer'; this requires the count of the products in the Transaction table and the count of products in the ReturnedItems table for this customer. Calculating the proportion would be part of the function Φ, not the part of the aggregation.

3. Dependent attributes within one table:
$y = \Phi_3(t, \Psi^m(\pi_{B.(f,\ldots,l)}(T \underset{t_i=b_j,(1:n)}{\bowtie} B)))$
The attributes from the objects in table B cannot be aggregated independently as before but have to be considered jointly using a multidimensional aggregation. The number of products bought on December 22^{nd} is an example that could be expressed using conditioning (on Date), selection, and then a simple aggregation. A more complex example is the slope of the price over time, explaining whether a customer is buying increasingly more-expensive products.

4. Dependent attributes across tables:
$y = \Phi_4(t, \Psi^t(\pi_{A.(f_A,\ldots,l_A)}(T \underset{t_i=a_j,(1:n)}{\bowtie} A),$
$\pi_{B.(f_B,\ldots,l_B)}(T \underset{t_i=b_j,(1:n)}{\bowtie} B)))$
The total amount spent on items later returned requires information from two tables (Transaction, ReturnedItems) during the aggregation. Since the two tables may have different types they cannot simply be merged into one. Note that even if they have the same type (the joins end at the same table) it may be important to know from which join a particular object has come. This information is lost if the results are simply merged.[4]

5. Global graph features:
$y = \Phi_5(t, \Psi^t(\Theta))$ where Θ stands for transitive closure over a set of possible joins. Such a global concept could for instance be a function of customer reputation. The aggregation Ψ^t for reputation may require the construction of an adjacency matrix and the calculation of its eigenvalues and eigenvectors.

Sublevels in the hierarchy
These five concept classes constitute a coarse hierarchy where each class comprises a number of sublevels. Within a given class, various dimensions span the space of subclasses. For example, if there is relational autocorrelation [8] in the target values and if the target values are known for some rows of T, Φ may be able to take advantage [20]. In such domains, joins should link back to the target relation and the (known) values of the target variable can be aggregated through a simple aggregation:
$\Psi^s(\pi_{T.y}(T \underset{t_i=b_k,(1:n)}{\bowtie} B, \ldots, A \underset{a_f=t_j,(1:n)}{\bowtie} T))$.
Another example of a subclass dimension is the assumed form of attribute dependence within concept class 3. The simplest case is a pairwise dependence of attributes with boolean conditioning. A more complex case is a (linear) correlation between pairs of numeric attributes. And so on. For the empirical results presented below, we primarily examine sublevels of class 2 (independent attributes), except where noted otherwise. We introduce novel aggregation functions that summarize the distributions of the values to be aggregated. More on this below.

3. RELATIONAL AGGREGATION

The following sections present three common approaches to aggregation in relational learning, a novel approach that combines vector distances with target-dependent aggregation, and how they relate to the presented concept classes. Relational learning has taken two approaches: (1) aggregation-based feature construction/invention with subsequent model estimation and (2) direct learning of a relational mapping. Aggregation must take place in either approach; the main difference between the two is whether the aggregation is optimized jointly with the estimation of Φ (often rendering the aggregation more complex) or whether they are performed independently. As shown for some cases in the presentation of the hierarchy, there are interactions between the aggregation and the function. That notwithstanding, for the remainder of this paper we focus on the aggregation operators Ψ, assuming the existence of some strategy for the identification of related objects[5] as well as an appropriate learner for Φ.

3.1 First-Order Logic

The field of relational learning for years has been dominated by Inductive Logic Programming (ILP) [18], focus-

[4]The question of whether two equal-type bags can be merged prior to aggregation in combination with a simpler multidimensional aggregation depends on whether the aggregation is transitive under the bag-merging operation. The *minimum* for instance is transitive, since the minimum of the minimums of two sets is the same as the minimum of the merged set.

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

ing on classification tasks. These first-order-logic-based approaches search for sets of clauses, typically that identify positive examples. For instance a clause learned to characterize a rich customer (cf., Figure 1) might be:

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

The prediction of an ILP model is positive if at least one of the clauses is true for the particular case. Binary propositionalization [13] [12] [1] also learns sets of (first-order) clauses, but rather than using them directly for prediction it constructs binary features that are given as input to a traditional learning method (e.g., decision tree induction) to learn the function Φ. Both ILP and binary propositionalization use existential unification of first-order-logic clauses for aggregation. Given the tables from section 2, the example clause Customer(X,Y,Z), Transaction(X,V,P,W), P≥100 is true 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 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 level 4 task of identifying customers who bought a product that was returned by another customer who bought it after 2001 can be expressed in first-order logic as:

Customer(X),Transaction(X,V,P,W),ReturnedItem(Y,W),
Transaction(Y,B,C,W),B≥2001

The disadvantage of logic-based aggregation is the common lack of support for numeric aggregation. In particular, it is difficult in pure logic to express that a product was returned more than 20 times. A function-free clause can test whether the maximum of a numeric set is larger than a particular value but it cannot estimate the mean or the cardinality of the set. In order for an ILP system to apply numeric aggregates they have to be declared by the user of the system as intensional background knowledge (as for instance proposed by Muggleton [17]) and only a few systems support such intensional declarations.

ILP currently is the only approach that explores concepts up to level 4 in the hierarchy. However, without explicit numeric support through intensional background knowledge, ILP methods are severely limited in expressive power in comparison to the methods discussed below.

3.2 Simple Numeric Aggregation

A number of relational approaches including Probabilistic Relational Models [11] and 'upgraded' propositional learners such as Relational Decision Trees [7] rely on a set of simple aggregation operators such as mean, min, max, count for numerical values, and proportions and most common value for categorical variables. Numeric aggregates in combination with logic-based feature construction were proposed by Knobbe et al. [10]. These operators apply only to bags of single attributes and cannot express concepts above level 2, which require dependent aggregation.

3.3 Set Distances

Kirsten, Wrobel and Horwath [9] proposed a distance-based method for relational learning. The approach classifies objects using a k-nearest-neighbor method with a predefined relational distance metric. This metric aggregates two bags of objects related to two cases by calculating the minimum distance of all possible pairs of objects, choosing one from one bag and one from the other. The distance between two objects is the sum of squared distances for numeric values and edit distances 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 implicitly assumes attribute independence and does not take advantage of numeric aggregates like count or average.

3.4 Value Distributions and Target-Dependent Aggregation

The objective of aggregation can be understood as a summarization of the underlying distribution from which the related objects were sampled. Therefore, the design and selection of appropriate aggregations depends on the characteristics of the value distributions. For example, limiting the aggregation to mean and variance for a numeric attribute is sufficient description for a Normal distribution since all higher moments are zero. Typically, simple aggregations such as mean, mode, and max, provide only a very coarse summarization.

We now present a more complex approach to independent-attribute aggregation, based on nonparametric density estimation combined with the observation that there may be important differences in the distributions for different classes. The methods we present here are for categorical attributes, but the general notion also applies to numeric attributes. Furthermore, the methods themselves are easily extended to numeric attributes after discretizing and coding of numerical values as categorical dummies.[6]

A common method to aggregate a single categorical attribute with numerous values is the selection of a subset of values that appear most often and convert them into dummy variables or counts. However, the most common values may not be the most predictive for a given relational learning task. Our approach examines the distributions of values conditioned on the classes of the training cases in order to select predictive values and to construct "reference vectors" that capture the typical distributions of values related to positive or negative cases. Particular cases will be compared to the reference vectors to produce aggregated values.

Consider a *value order* to be an ordering of the values of a categorical attribute. We consider a value order to be a list of (value v_i:index i) pairs, for clarity when referring to the indices instead of the values themselves. For example, a value order for ProductType might be (watch:1,book:2,CD:3,DVD:4).

Definition 1: Given a particular case t (a row of target table T) and an arbitrary value order for categorical attribute $B.j$, *case vector* $CV^t_{B.j}(\pi_{B.j}(T \underset{t_i=b_k,(1:n)}{\bowtie} B))$ has at position i the number of instances of v_i in the bag returned by the join and projection. For example, the bag of ProductTypes {book,CD,CD,book,DVD,book} for a specific case t under the order shown above would result in $CV^t_{Products.ProductType} = (0, 3, 2, 1)$.

[6] In order to preserve the native distances of numeric values one should use a thermometer coding scheme ([23],[15]) to code the dummies. This approach corresponds to estimating the cumulative distribution rather than the probability distribution function.

Definition 2: Given a selection condition c and a value order for attribute $B.j$, *reference vector* $RV^c_{B.j}(\pi_{B.j}(T \frac{\bowtie}{t_i=b_k,(1:n)} B))$ has at position i the sum of values $CV^t[i]$ for all cases t for which c was true.

Definition 3: Given a selection condition c and a value order for attribute $B.j$, *variance vector* $VV^c_{B.j}(\pi_{B.j}(T \frac{\bowtie}{t_i=b_k,(1:n)} B))$ has at position i the variance $\frac{(CV^t_{B.j}[i])^2}{N_c-1}$ of the corresponding case-vector values over all cases t for which c was true. N_c is the number of cases for which the condition c was true.

We now will use these vectors to construct two types of feature: (i) useful individual values, and (ii) distribution-based features (here, using vector distances). We focus on constructing features that take into account the different value distributions for positive and negative cases.

3.4.1 Target-Dependent Individual Values

As mentioned above, a straightforward method for creating features is to create dummy variables based on frequently occurring values; specifically, one could choose the most common (MOC) value(s) in the unconditional reference vector, RV.

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 (MOP) and analogously those that are most commonly related to negative cases (MON). Specifically, for MOP, given an arbitrary value ordering and $RV^{y=1}$ we select those values v_i for which $RV^{y=1}[i]$ is largest. Similarly for MON we select those values v_i for which $RV^{y=0}[i]$ is largest.

A more complex, difference-based approach selects categorical values that are common for one class but not common for the other. In particular we select the values for which the absolute value of $RV^{y=1}[i] - RV^{y=0}[i]$ is largest (MOD). The Mahalanobis distance [14] normalizes the scores by the variances before selecting the largest (MOM):

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

Table 1 summarizes the five strategies to select dummies from single categorical values grouped into three groups of increasing complexity: target independent most common (first row), target-dependent most common positive or negative value (second row and third row), and the value(s) with the maximum difference between the positive and negative reference vectors (fourth row and fifth row).

3.4.2 Vector Distances

Using vector distances, features can be constructed that compare distributions of values more comprehensively, rather than summarizing using only the most frequent values. Specifically, for each case vector CV and reference vector RV we compute four vector distances: edit distance (ED), Euclidean distance (EU), Mahalanobis distance (MA), and cosine distance (COS). Since it is not clear a priori which of the distances will best capture the underlying concept, we compute all of them and leave it to the function estimator for Φ to select among them.

Similarly to the difference-based methods in the previous section, in addition to these distances we also calculate for each of the four measures the difference between the distances to the positive and negative reference vectors:

$$EDD = ED(RV^{y=1}, CV) - ED(RV^{y=0}, CV)$$
$$EUD = EU(RV^{y=1}, CV) - EU(RV^{y=0}, CV)$$
$$COSD = COS(RV^{y=1}, CV) - COS(RV^{y=0}, CV)$$
$$MAD = MA(RV^{y=1}, CV) - MA(RV^{y=0}, CV)$$

Combining the options for distance and target conditions, we have a three-by-four matrix of vector-based aggregations shown in Table 2. As with the methods of the previous section, the vector-distance aggregations can be grouped into three increasingly more complex groups: target independent (first row), dependent on either positive or negative reference vector (second and third row), and dependent on the difference between the target-dependent distances (fourth row).

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

Method	Definition
MOC	$CV[i]$ where i is the index (one of the indices) for which $RV[i]$ is largest (RV is the unconditional reference vector)
MOP	$CV[i]$ where i is the index (one of the indices) for which $RV^{y=1}[i]$ is largest (the positive reference vector)
MON	$CV[i]$ where i is the index (one of the indices) for which $RV^{y=0}[i]$ is largest (the negative reference vector)
MOD	$CV[i]$ where i is the index (one of the indices) where the absolute value is largest in vector $RV^{y=1} - RV^{y=0}$
MOM	$CV[i]$ where i is the index (one of the indices) with maximum absolute value of $\frac{RV^{y=1}[i]-RV^{y=0}[i]}{VV^{y=1}[i]+VV^{y=0}[i]}$

Table 1: Summary of aggregation methods producing features representing single categorical values

Reference Vector	Euclidean	Edit	Cosine	Mahalanobis
All	EU	ED	COS	MA
Positive	EUP	EDP	COSP	MAP
Negative	EUN	EDN	COSN	MAN
Positive vs. Negative	EUD	EDD	COSD	MAD

Table 2: Summary of vector-based aggregation methods

4. EXPERIMENTAL RESULTS

In this section we present results comparing the various aggregation methods on a relational learning problem in the domain of initial public stock offerings. We include the comparative performance of four logic-based relational learners

(FOIL [22], Tilde [2], Lime [16], and Progol [17]), which can express concepts of up to level 4. All other methods reside within level 2. The next sections present the domain description, a brief overview of the methodology, and our results.

4.1 Domain: Initial Public Offerings

Initial public stock offerings have a unique ticker for the firm that is selling shares of its equity. An IPO typically is headed by one or occasionally two banks and is supported by a number of additional banks as underwriters. The job 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 represent an instance of an abstraction hierarchy on SIC (Standard Industrial Classification) codes. For example SIC code 7372 identifies the division of 'Prepackaged software'. This particular industry group is a member of the major group 'Business Services' with the two-digit code 73.

In this domain, Date, Size, Price and Runup are numerical variables; Ticker, Bank, SIC, Ind2 are categorical and keys, and Ind and Exchange are simple categorical attributes. Note that the background tables contain only categorical attributes whereas the target table has mostly numeric attributes. The classification task is to predict whether the offer was (would be) made on the NASDAQ exchange.

4.2 Methods

We compare the generalization performance of 4 general approaches: ILP, logic-based feature construction, selection of specific individual values, and target-dependent vector aggregation. We also constructed two other features from the relational background data: when there is an instance of an abstraction hierarchy (via a sequence of n:1 joins) we include the values directly in the feature vector (AH). For comparison, we also include a constructed feature that takes advantage of relational autocorrelation (as Provost et al. advise [20]). The "autocorrelation" aggregation (AC) takes advantage of joins back to the target table, and computes the proportion of linked training cases that are positive (excluding the particular case in question of course). Table 3 summarizes the approaches.

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 builds a model to predict the target given the aggregates and the features in the target table. 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 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 a stopping criterion. The second number in the size column in Table 4 shows the stopping criterion (maximal number of chains) for the exploration. For each exploration chain, the system executes the corresponding join(s) 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 from the target table.

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

Model Estimation: We used C4.5 [21] to learn a tree for each of the 10 feature sets. For each class-probability prediction, we used the average over the 10 trees of the class frequencies at the leaves. For classification, we thresholded the estimates at 0.5. 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 [22] to learn n first-order-logic (FOL) clauses and appended the corresponding binary features to the feature vector in the target table IPO. This methodology was applied successfully to text classification by King [24] and by Populescul et al. [19].

ILP: We selected four ILP systems based on availability, platform independence and diversity. FOIL [22] uses a top-down, separate-and-conquer strategy adding literals to the originally empty clause until a minimum accuracy is achieved. Tilde [2] learns a relational decision tree using FOL clauses in the nodes to split the data. Lime [16] is a top-down ILP system that uses Bayesian criteria to select literals. Progol [17] learns a set of clauses following a bottom-up approach that generalizes the training examples. We did not provide any additional (intensional) background knowledge beyond the facts in the database. We supplied a declarative language bias for Tilde, Lime, and Progol (as required).

Evaluation: Generalization performance is evaluated in terms of classification accuracy and area under the receiver operating curve (AUC) [3]. Note that ILP systems only produce class labels but no probability scores. We therefore included ILP only in the accuracy comparisons. All results represent generalization performance averaged over 5 runs on test sets of size 800. We show error bars of $\pm$ one standard deviation in the figures but refrained from including them in the tables.

4.3 Results

Table 4 shows the generalization performance of the aggregation methods as a function of training-set size and the number of joins allowed. The methods are grouped into four (vertical) blocks of increasing complexity: no feature construction (NO), target-independent aggregation (MOC, VD, MVD), target-dependent aggregation dependent on either positive or negative class (MPN, VDPN, MVDPN), and target-dependent aggregation based on the difference between the positive and negative reference vectors (MD,VDD, MVDD). To help with the abbreviations (needed to make

Method Name	Description
NO	No feature construction, no relational information—only the attributes 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 the 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	Attributes in IPO and attribute Ind in table Ind2 related through the abstraction hierarchy
AC	Attributes in IPO and the proportion of training cases (excluding the particular case) that a case was related to that are positive
LF	Logic-based features extracted from the clauses learned by FOIL

Table 3: Summary of aggregation approaches compared in experiments

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	0.690
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 4: Classification accuracy of aggregation methods grouped by complexity

the table legible) a condensed summary of the different methods under comparison can be found in Table 3. Within each block of methods in Table 4, the first column presents an aggregation method that uses only specific categorical values (the five with the largest entries in the reference vector), the second column only vector distances, and the third column both.

The best performance for each training size (three-row horizontal division) is highlighted in bold, and the best performance for each 3x3 sub-block is in italics (if it is not already bold). The results show that as the complexity of aggregation increases, performance increases as well. The best performance within a sub-block is always one of the two aggregations including vector distances. Using only single categorical values is almost always outperformed by vector-distance aggregation. Increasing the exploration limit (maximum number of joins) improves performance in most cases; however the marginal improvement decreases. Specifically, the increase in performance moving from 6 joins to 9 is larger than moving from 9 to 12 joins (the latter sometimes hurts performance). Moving to a larger number of joins can hurt performance for two reasons: (1) the longer the chain that relates objects to a target case, the further away and less relevant the linked entities 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 problems of multiple comparison [6].

Figure 2 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, and that more-complex aggregation leads to better the performance. The graph also shows that the most complex aggregation (VDD) has the smallest variance of the four methods.

Analyzing the tree learned by C4.5 for the most complex model MVDD identifies the following variables as predictive: whether one underwriter was 'Hambrecht', the difference between the edit distances to the positive and negative reference vectors of the underwriting banks, the number of IPO's previously underwritten by the head bank, the date of the IPO, the difference between the positive and negative edit distances of the head bank, the two-digit industry code, and the difference in Mahalanobis distance to the IPOs previously performed by the underwriting banks. This provides

Size	NO	AH	FOIL	Tilde	Lime	Progol	AC	LF
250	0.649	0.641	0.645	0.646	0.568	0.594	0.73	0.592
500	0.650	0.665	0.664	0.628	0.563	0.558	0.719	0.643
1000	0.662	0.701	0.658	0.630	0.530	0.530	0.724	0.638
2000	0.681	0.711	0.671	0.650	0.512	0.541	0.753	0.641

Table 5: Classification accuracy of aggregation methods that are independent of join depth

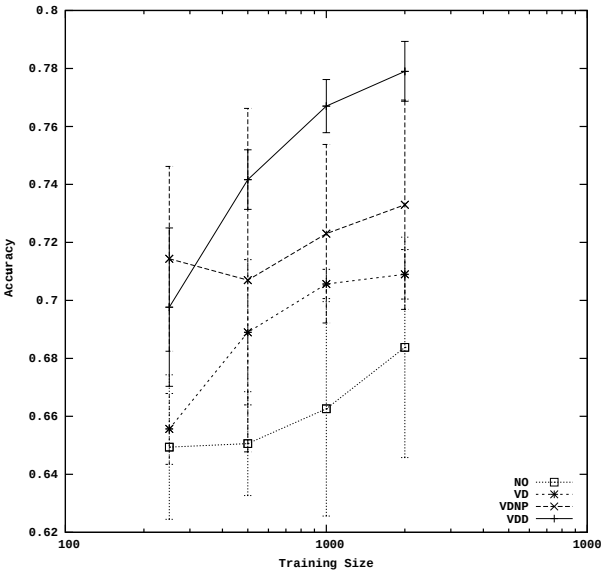

Figure 2: Learning curves: accuracy as a function of training-set size for NO, VD, VDNP, and VDD

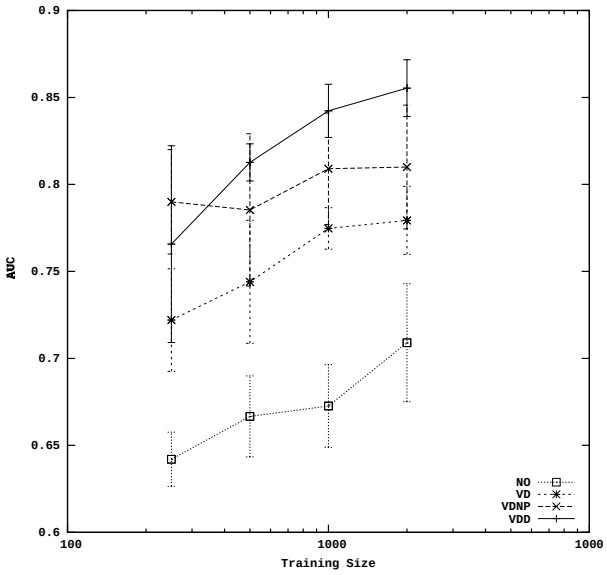

Figure 3: Learning curves: AUC as a function of training-set size for NO, VD, VDNP, and VDD

confirming evidence that the differences between the vector distances play an important role for this task.

Table 5 contrasts the results for the other methods, which are independent of the number of joins: abstraction hierarchies (AH) in the domain tables IND2 and IND, the four ILP systems FOIL, Tilde, Lime, and Progol, relational autocorrelation (AC), and logic-based feature construction (LF).

Including the values of the abstraction hierarchy (AH) improves slightly over no relational background knowledge (suggesting that the industry classes are linked to exchange), but cannot compete with target-based aggregation. The AC results show that there is a significant degree of autocorrelation in this domain. Banks seem to operate primarily on one exchange or the other. AC outperforms all methods in this table, and only falls short of the best aggregation method MVDD in Table 4.

The two ILP systems FOIL and Tilde are still competitive for small data sets but for larger training sets do not do as well as 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 the training data since it contains, in addition to the binary features, all the original attributes from the target table IPO used by NO. The binary features are learned from the 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 6) are similar to the results for accuracy. The most complex aggregation methods (MVDD or VDD) outperform the other methods and the performances increase with training size. Figure 3 shows the learning curves of NO,VD,VDPN, and VDD including error bars of $\pm$ one standard deviation for 12 joins.

Figure 4 shows the ROC curves for NO, MVD, MVDNP, and MVDD exploring 12 joins. MVDD and MVDNP present an interesting case where the ROC curves cross. MVDNP is better for high thresholds whereas MVDD is better for lower thresholds. Analyzing the probability estimation performances of methods that are independent of join depth in Table 7 shows again that the autocorrelation aggregation (AC) performs very well, only surpassed by the best performing aggregations. Abstraction hierarchies (AH) are not as useful for probability estimation as they were for classi-

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	0.760
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*	0.790	0.81	0.788	0.798	**0.855**	0.836

Table 6: Probability estimation performance (AUC) for aggregation methods grouped by complexity

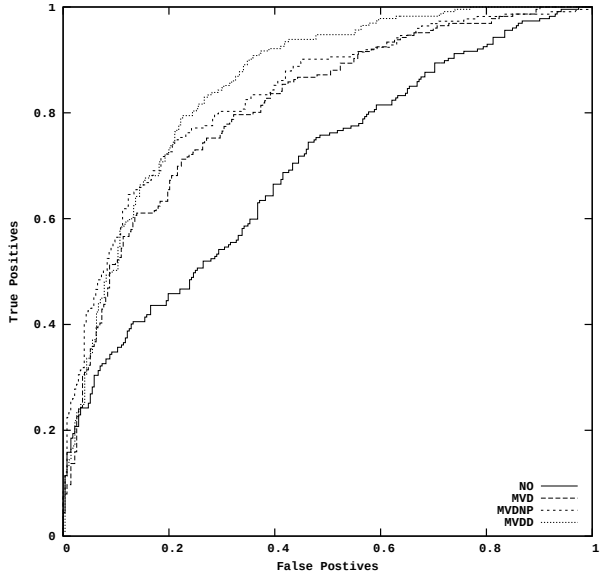

Figure 4: ROC curves for NO, MVD, MVDNP and MVDD

fication. Note, that ILP systems only predict a class label and therefore do not appear in the table.

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 7: Probability estimation performance (AUC) of aggregation methods that are independent of join depth

5. CONCLUSIONS AND FUTURE WORK

The main contribution of this work is the first detailed examination of aggregation for relational learning. Along with search through the relationship graph, aggregation is a major component of any relational learning method. We have shown, via the concept hierarchy, 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.

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). Within level 2 of the hierarchy, we have presented aggregation methods of various complexities. 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 aggregations.

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 aggregation. These aggregations transform the relational learning task into a conventional feature-vector learning task, enabling the application of conventional learning methods. An advantage of this transformation-based approach is its general applicability to regression, classification, and probability estimation tasks.

Our results further show that for the same level of performance, increased aggregation complexity can reduce the amount of exploration necessary. This is important 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 remains an important research topic in relational learning [5].

Although quite suggestive, the generalizability of our positive findings (in favor of the more complex aggregations) is limited due to the focus on one particular domain and the limited maximum training size of 2000. Future work includes extending these experiments to multiple domains with different relational characteristics.

The collection of aggregation methods we have presented are certainly is not complete (even within level 2 of the hierarchy). Our findings motivate further exploration of potential aggregation methods. In particular numeric multi-dimensional and multi-type aggregation has received little treatment. 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 de-

velop a better understanding of why methods perform better or worse on certain domains.

Acknowledgments

We are grateful to Jeff Siminoff, Sofus Macskassy, and William Greene for valuable comments and discussions. This work is sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-01-2-585. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.

6. REFERENCES

[1] J.M. Aronis and F.J. Provost. Efficiently constructing relational features from background knowledge for inductive machine learning. In U. Fayyad and R. Uthurusamy, editors, *In Working Notes of the AAAI-94 Workshop on Knowledge Discovery in Databases (KDD-94)*, pages 347–358, 1994.

[2] Hendrik Blockeel and Luc De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.

[3] A.P. 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.

[4] L. De Raedt, H. Blockeel, L. Dehaspe, and W. Van Laer. Three companions for data mining in first order logic. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 105–139. Springer-Verlag, 2001.

[5] J. Fürnkranz. Dimensionality reduction in ILP: A call to arms. In Raedt L. and Muggleton S., editors, *Proceedings of the IJCAI-97 Workshop on Frontiers of Inductive Logic Programming*, 1997.

[6] D. Jensen and P.R. Cohen. Multiple comparisons in induction algorithms. In *Machine Learning*, volume 38, pages 309–338, 2000.

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

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

[9] M. Kirsten, S. Wrobel, and T. Horvath. Distance based approaches to relational learning and clustering. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 213–232. Springer Verlag, 2001.

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

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

[12] S. Kramer, N. Lavrac, and P. Flach. Propositionalization approaches to relational data mining. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 262–291. Springer-Verlag, 2001.

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

[14] P.C. Mahalanobis. On the generalized distance in satistics. In *Proc. Natl. Institute of Science of India*, volume 12, pages 49–55, 1936.

[15] T. Masters. *Practical Neural Network Recipes in C++*. San Diego: Academic Press, 1993.

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

[17] S.H. Muggleton. Cprogol4.4: a tutorial introduction. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 105–139. Springer-Verlag, 2001.

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

[19] A. Popescul, L. H. Ungar, S. Lawrence, and D. M. Pennock. 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, July 2002.

[20] F. Provost, C. Perlich, and S. Macskassy. Relational learning problems and simple models. In Lise Getoor and David Jensen, editors, *Working Notes of the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data*, pages 116–120, 2003.

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

[22] J.R. Quinlan and R.M. 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.

[23] M. Smith. *Neural Networks for Statistical Modeling*. Boston: International Thomson Computer Press, 1996.

[24] A. Srinivasan and R.D. 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.