Automated Construction of Relational Attributes

ACORA: A Progress Report

Claudia Perlich

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

August 2002

Center for Digital Economy Research
Stern School of Business
Working Paper IS-02-04

Automated Construction of Relational Attributes

ACORA: A Progress Report

Claudia Perlich
August 2002

Department of Information, Operations and Management Science
Information Systems Group

Abstract

Data mining research has not only development a large number of algorithms, but also enhanced our knowledge and understanding of their applicability and performance. However, the application of data mining technology in business environments is still no very common, despite the fact that organizations have access to large amounts of data and make decisions that could profit from data mining on a daily basis. One of the reasons is the mismatch between data representation for data storage and data analysis. Data are most commonly stored in multi-table relational databases whereas data mining methods require that the data be represented as a simple feature vector. This work presents a general framework for feature construction from multiple relational tables for data mining applications. The second part describes our prototype implementation ACORA (Automated Construction of Relational Features).

Introduction

Given the successful development of a multitude of data mining algorithms, approaches and methods, one would expect their application to be standard practice in organizations that have access to large amounts of data and make decisions on a daily basis. However, the number of organizations that apply data mining methods for decision support is still relatively small. One of the reasons is the mismatch between data representation for data storage and data analysis. Data are most commonly stored in multi-table relational databases. Every table has a number of attributes and the tables can be related to each other using keys. A customer database for example can contain two tables **CUSTOMER** and **PAST_PURCHASES** and both tables contain the attribute CUSTOMERID.

On the other hand data mining methods (e.g. decision trees and neural networks) and statistical analysis tools (e.g. linear regression) require that the data be represented as a feature vector $(x_1,\ldots, x_n, y)$ of n attributes for each observation. In the example above the important information about past purchases is stored in a separate table with potentially multiple transactions per customer and the number of entries varies for each customer. For some inactive customers, only one past purchase may be available whereas, for some highly active customers there could be more than hundred. If the objective of a marketing expert is to identify those customers who are likely to buy a new product after a direct marketing action, we would like to identify those customers who are most likely to respond to the direct marketing based on their past purchases. Faced with this situation, a marketing expert would manually have to generate the features x_i by joining both tables and aggregating the information about the sets of past purchases. A typical aggregate is, for instance, "amount spent within the last 2 months".

Not only does this manual feature construction require significant technical expertise (e.g., knowledge of SQL), but also strong prior ideas what kind of features might be important. The manual process of feature construction is very time consuming and becomes infeasible for a large number of tables.

This work provides a framework for the automated construction of relevant features for data mining from multiple relational tables. We use the term data mining in a very broad sense including machine learning and statistical analysis. Our approach will however be focused on predictive tasks (e.g., regression, classification, and probability estimation). Throughout the paper we will use the words *features* and *attributes* as synonyms.

A Transformation-Based Framework for Relational Data Mining

Our framework as presented in figure 1 takes a transformation-based approach to data mining, where learning is perceived as a sequence of successive steps of data transformation. In the first step T1, we construct features from relational tables. The result is a feature vector that is augmented in T2 and can be used in the last step T3 to learn a predictive model. The transformation T2 creates new non-relational features from mathematical combinations and transformations of existing features. A typical example is the introduction of interaction effects and higher order terms in linear models and re-scaling of features using logs (e.g., $x_{new} = \ln(x_{old})$).

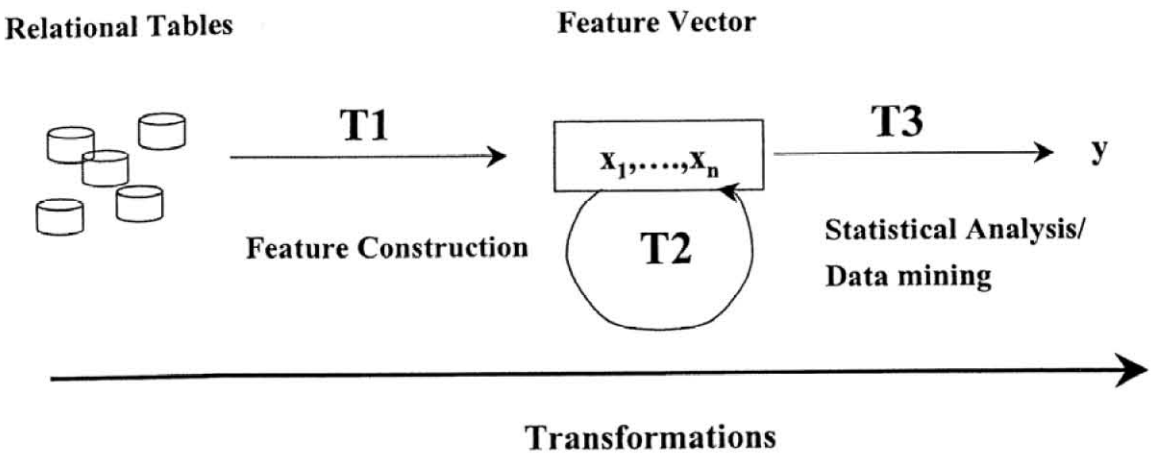


Figure 1: Transformation-based framework of data mining

The focus of our work is the feature construction step T1. The traditional manual approach to relational

data analysis has been expert-guided feature construction T1. It is possible to view the process of feature construction from relational data as a special case of preprocessing. However we argue that this step is extremely important and should receive its due attention as a main part of the modeling effort, as noted by Langley and Simon (1995) among many others.

De Raedt (1998) showed that for most relational databases (no recursive concepts) complete transformations (without loss of information) are possible, but result in an exponentially growing feature space with potentially sparse feature vectors. However the objective of the transformation T1 is not a complete transformation but rather the extraction and creation of valuable features for the modeling task.

Constructing Features from Relational Data

Figure 2 shows our modular framework of feature construction (T1) as a three-step procedure. We will use the expression "target relation" to refer to the table that contains the objects of interest for which a prediction will be made. In the previous example it is the customer table since we want to predict the probability that a customer will buy the product.

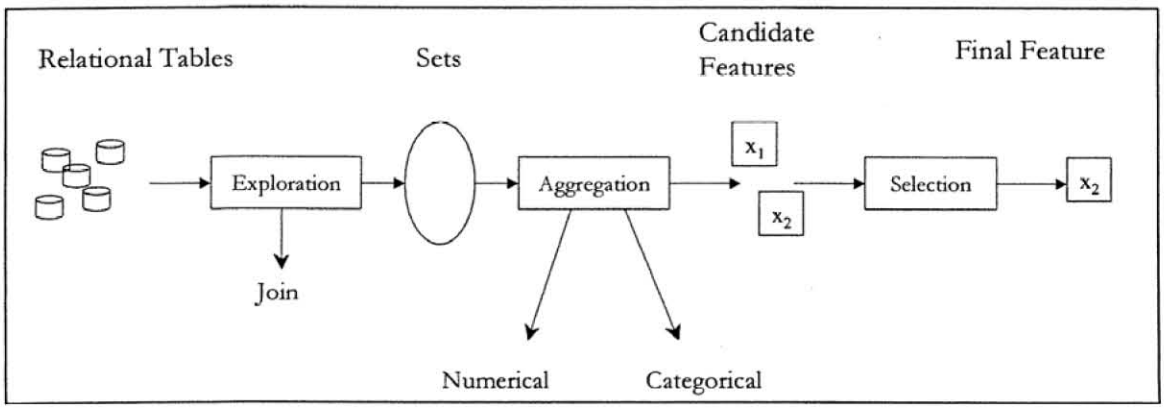


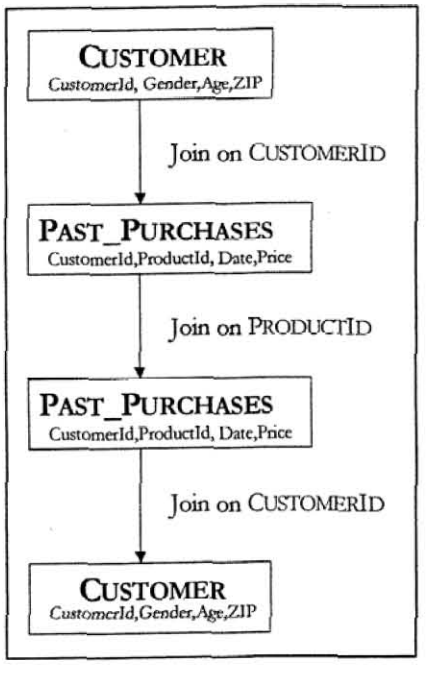
Figure 2: Framework for feature construction from relational tables

In the exploration step we join the target relation with other tables in the database that have at least one key attribute in common with the target relation. The second step is the application of aggregation methods to the results of the join. Each of those aggregation methods generates one value (either numeric

or categorical) per row in the target relation. After this generation of new features a selection procedure identifies valuable features for the modeling task T3. The three steps are described in more detail in the following paragraphs.

Structural Exploration

For the target relation $R_T(a_1,\ldots, a_i,\ldots,a_n)$ there is a set of background relations R_B $(b_1,\ldots, b_i,\ldots,b_n)$ which can be related to R_T, *given a key attribute. Formally, the operation is a join* $a_i=b_j$, where we only select the attributes $(b_1,\ldots, b_i,\ldots,b_n)$ from relation R_B. Depending on the cardinality of the relationship between the two relations, this may produce the same number of objects (as in the case of a 1-to-1), or a variable number (as in the case of 1-to-n). For example, two tables **CUSTOMER** and **PAST_PURCHASES** could be joined on the key CUSTOMERID (see Figure 3). The result of the join has multiple rows for each customer and the number of rows is equal to the number of transactions the customer has performed in the past. Conceptually there is a tree of all possible joins (potentially over multiple tables) with the target relation being the top of the tree. The nodes in layer of depth n correspond to all possible joins over n relations.



Each path from the top to a node involves multiple joins and generates one table. The only requirement is that every node has to share at least one key variable with one of the nodes in its path. It is possible to explore this tree using any standard tree-search algorithm (e.g., breadth-first search). Figure 3 shows the tree (after pruning identical joins) of depth 3 for the marketing example assuming that available keys are CUSTOMERID and PRODUCTID. The 3 joins corresponding to the full path would return the sets of all customers that bought at least one product identical to any one bought by the customer in the target relation.

Figure 3: Exploration tree for marketing example

Aggregation:

Structural exploration produces a table with a set of entries for each object in the target relation as shown on the example of customer transactions. The task of the aggregation operators is to construct single-valued attributes from each column of those sets. Depending on the data type, different aggregation operators can be applied. Figure 2 shows the 2 main groups of aggregation operators based on the data type (categorical or numerical) of the column. The column "Price" of the **PAST_PURCHASE** table, for example, is numeric and can be aggregated using simple sample statistics such as median, mean, min, and max. Categorical aggregation can involve the creation of dummies or counts for each value. One can however extend the current approach with much more complex operators like estimation of set- and density-distances (e.g., Kullback-Leibler) to a reference set.

Selection:

The aggregation step can create a significant number of potential features. Most predictive data mining methods show decreasing performance as the dimensionality of the input space increases ("the curse of dimensionality"). It is therefore useful to implement a feature selection step that assesses the value of a feature for the modeling task. It is a design question how much effort should be put into the evaluation of the usefulness of a feature since it involves a significant computational effort. In the simplest case this can be done using only the particular feature in question. The disadvantage is that one might miss features that are conditionally predictive. The decision as to how much computational effort is optimal depends also on the desired run time behavior, the choice of model class, and domain properties such as number of tables.

Implementation: ACORA

We have implemented a prototype of our feature construction framework called **ACORA** (**A**utomated **C**onstruction **O**f **R**elational **A**ttributes). The general philosophy is to allow the user to provide as rich a specification as he or she wants and to provide additionally default heuristics for all things the user does

not want to specify. This flexibility allows the comparison of methods and enables users with different technical competencies to interact with the system. The goal is to develop robust mechanisms that can guaranty good performance on a number of domains and prediction tasks. The following table gives a short overview of the system parameters and guidelines for the development of heuristics.

Heuristic for structural exploration	The simplest heuristic is breadth-first tree traversal. There are a number of more elaborate schemes that prune the search space aggressively. This is of particular importance if the number of tables is large.
Heuristic to select aggregation operators	Operators are constrained by the attribute type. However other factors that determine the appropriateness of an operator are for instance the average and variance of the set sizes.
Selection criterion	Data mining methods differ in their ability to learn from high-dimensional input spaces with many irrelevant features. Selection should be more aggressive for neural network models than for decision trees. In the simplest case the selection can be based on a minimum performance of a model that uses only one attribute to predict the target.
Stopping criterion	Given the large number of possible joins and features, there should be a limit to the creation of features. Stopping can be based on the tree depth N, the model performance (at a high computational cost), or on the number of features (e.g., the ratio of observations to features should be at least 5 for linear models). If the selection is not sufficiently aggressive, the stopping criteria might be reached without sufficient exploration of the feature space. A heuristic that takes both factors into account could create a high number of features, rank them based on the selection criteria and pick the top k features.

Table 1: System Parameters

One open research question is the degree of automation that can be achieved without losing much performance on a number of domains. Of additional interest are the interaction effects of those heuristics on run time and prediction performance of various data mining methods.

The system is currently in a development stage with most of the structural exploration and aggregation implemented. We plan to provide interfaces to a number of machine learning techniques to evaluate our approach on a number of prediction tasks. In particular we will include decision trees (C4.5), logistic and linear regression. Our prototype currently requires at least a specification of all relational tables names and the types of their attributes. After reading in the table specification **CUSTOMER**(CUSTOMERID, GENDER, INCOME, ZIP) for the customer table and **PAST_PURCHASES**(CUSTOMERID, DATE, PRICE, PRODUCTID), the system identifies based on attribute name automatically all possible keys (CUSTOMERID and PRODUCTID) on which tables can be joined unless the keys were specified explicitly. The system architecture is modular and open to extensions with new aggregation operators.

Related Work

This work is related to a number of research areas: feature construction, inductive logic programming, propositionalization, OLAP, and learning from relational databases.

OLAP enables an analyst to create high-dimensional representations from joins of multiple relations and to select subsets and aggregations. The current capabilities of OLAP systems provide an infrastructure for data analysis but lack mechanisms for automated analysis and predictive modeling. Machine learning and data mining have developed a suite of algorithms for automated learning of predictive models from feature vectors (x_1, x_2, x_3, x_4...x_n). The reader is referred to a standard textbook (Mitchell 1997) for an overview of existing methods. Kramer et al. (1998) coined the term "propositionalization" for the transformation of relational data into a feature-vector (propositional) representation. There exist a number of systems (e.g., LINUS, STILL, REPART discussed in Kramer et al. 2001) that perform propositionalization on relational data automatically and apply a data mining method for prediction. However those efforts focus exclusively on binary features in form of logical clauses and are not capable of performing numeric aggregation. We are aware of a number of non-automated attempts to use aggregation for propositionalization. Knobbe (2001) applied SQL operators successfully to a banking domain. Morik and Brockhausen (1996) implemented a prototype for propositionalization called TOLKIEN as part of the MiningMart system.

There exist a number of specialized data mining algorithms that can learn classification models directly from multi-relational tables without intermediate feature construction. We are aware of two SQL extensions that enable users to learn predictive rules directly from relational data. The DBMiner system by Han et al. (1996) integrates a set of discovery modules into an SQL-accessible database. Imielinski and Virmani (1999) proposed MSQL as a modeling extension to SQL. Both methods require a very detailed specification of the model form. In the case of DMQL, the form of the rule has to be stated explicitly.

The user has to know all relevant features in advance in order to formulate such a rule. MSQL similarly requires all possible rules to be stored beforehand in a special table. Both methods lack a sufficient degree of automation to enable efficient data analysis with minimal user interaction. Morik and Brockhausen (1996) showed that the previous approaches are closely related to inductive logic programming (ILP) methods. ILP algorithms (see Muggleton and De Readt (1994) for a good introduction) are capable of learning relational classification models from multi-table data. The resulting model is a set of existentially unified first-order Horn clauses. Known disadvantages of ILP approaches are high sensitivity to noise, limited support for numeric features, high computational complexity and the limitation to classification tasks. This is a significant drawback for applications in business domains, since we need probability estimates for decisions involving expected cost-benefit tradeoffs.

Discussion

There are a number of dimensions along which our approach has the potential to improve over existing technology.

Automation: ILP and logic-based propositionalization are currently the only automated approaches to relational learning, but both require significant technical knowledge to specify search heuristics for particular domains.

Robustness: Current implementations of relational learners often have inherent search heuristics that are tailored to particular domains. Additionally, they are often criticized for low performance on noisy domains due their origins in logic. There is no consensus upon guidelines as to when an approach is likely to perform well.

Usability: All existing relational learners require substantial technical understanding of the implementation and very low-level specifications of search constraints. This will deter most domain experts from using such a system.

Probability estimation: One shortcoming of logic-based approaches is their inability to produce probability estimates, which is essential for decision-making based on cost-benefit analyses.

Run time: Most implementations of ILP as well as propositionalization suffer from unacceptable run time behavior. The presented feature construction approach can take advantage of a number of strategies to improve run time behavior. The structural exploration and aggregation can easily be parallelized by assigning each path to a different processor. Due to the modularity of operators it is comparatively simple to run a scaled-down version with fewer operators and a more aggressive pruning of the feature-search space. The use of sub-samples for feature selection can additionally improve the run-time behavior.

Modularity: The modular design of our framework might result in a less efficient implementation but improves the ability of a domain expert to provide at least partial guidance for smaller subtasks. The expert might be able to tell that mean aggregation is less useful than maxima. Additionally the modularity enhances the comprehensibility of the system.

Extensions

As the very next step, we will to conduct a thorough empirical comparison of our feature construction approach on a number of relational domains such as Web-logfiles from an online vendor (KDD Cup 2000), news stories on business co-occurrence (from Yahoo), patent references, initial public offerings, medical records, LINUX developer communication, contract killing, citation networks, movies, and nuclear smuggling. (Those datasets have already been obtained). We will compare the performance of feature construction in combination with decision trees and logistic regression against publicly available ILP implementation as well as no feature construction. Many of the domains mentioned above contain information about relationships between humans or business entities. Social Network Analysis (SNA) (Scott 1991) was developed in the social sciences for the analysis of the interaction of individuals, with particular focus on reputation and centrality measures. SNA has successfully been employed for example

for profiling in the investigation of money laundering (Sparrow 1991). SNA can easily be introduced into our framework using graph extraction as a method for structural exploration and centrality measures for aggregation.

Acknowledgements

This work is sponsored 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.

References

De Raedt, L. "Attribute Value Learning versus inductive logic programming: The missing links (extended abstract)," Proceedings of the Eights International Conference on Inductive Logic Programming, Springer Verlag, Berlin, 1998, pp. 1-8.

Han, J., Fu, Y., Wang, W., Chiang, J., Gong, W., Koperski, K., Li, D., Lu, Y., Rajan, A., Stefanovic, N., Xia, B., and Zaiane, O. R. "DBMiner: A System for Mining Knowledge in Large Relational Databases," Proceedings of the International Conference on Data Mining and Knowledge Discovery, 1996, pp. 250-255.

Imielinski, T., and Virmani, A. "MSQL: A query language for database mining," Data Mining and Knowledge Discovery 3(4), 1999, pp. 373-408.

Knobbe, A. J., De Haas, M., and Siebes, A. "Propositionalisation and Aggregates," Lecture Notes in Artificial Intelligence (2168), Springer Verlag, Berlin, 2001, pp. 277-288.

Kramer, S., Lavrac, N., and Flach, P. "Propositionalization Approaches to Relational Data Mining," Relational Data Mining, Springer Verlag, 2001, pp. 262-291.

Kramer S., Pfahringer, B., and Helma, C. "Stochastic Propositionalization of Non-determinate Background Knowledge," International Workshop on Inductive Logic Programming, 1998, pp. 80-94.

Langley, P., & Simon, H.A. (1995). Applications of machine learning and rule induction. Communications of the Association for Computing Machinery, 38(11), pp. 54-64.

Mitchell T. M. Machine Learning, McGraw-Hill, New York, 1997.

Morik, K. and Brockhausen P. "A Multistrategy Approach to Relational Knowledge Discovery in Databases," Proceedings of the 3rd International Workshop on Multistrategy Learning, AAAI Press, 1996, pp. 17-28.

Muggleton, S. "Inverse entailment and Progol," New Generation Computing (13), 1995, pp. 245-286.

Muggleton, S., and De Raedt, L. "Inductive Logic Programming; Theory and methods," Journal of Logic Programming (19,20), 1994, pp. 629-679

Scott, J. Social Network Analysis: A Handbook, Newbury Park, CA: Sage Publications, 1991.

Sparrow, M. K. "The application of network analysis to criminal intelligence: An assessment of the prospects," Social Networks (13), 1991, pp. 251-274.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-02-04

Automated Construction of Relational Attributes

ACORA: A Progress Report

Claudia Perlich
August 2002

Department of Information, Operations and Management Science
Information Systems Group

Abstract

Data mining research has not only development a large number of algorithms, but also enhanced our knowledge and understanding of their applicability and performance. However, the application of data mining technology in business environments is still no very common, despite the fact that organizations have access to large amounts of data and make decisions that could profit from data mining on a daily basis. One of the reasons is the mismatch between data representation for data storage and data analysis. Data are most commonly stored in multi-table relational databases whereas data mining methods require that the data be represented as a simple feature vector. This work presents a general framework for feature construction from multiple relational tables for data mining applications. The second part describes our prototype implementation ACORA (Automated Construction of Relational Features).

Introduction

Given the successful development of a multitude of data mining algorithms, approaches and methods, one would expect their application to be standard practice in organizations that have access to large amounts of data and make decisions on a daily basis. However, the number of organizations that apply data mining methods for decision support is still relatively small. One of the reasons is the mismatch between data representation for data storage and data analysis. Data are most commonly stored in multi-table relational databases. Every table has a number of attributes and the tables can be related to each other using keys. A customer database for example can contain two tables **CUSTOMER** and **PAST_PURCHASES** and both tables contain the attribute CUSTOMERID.

On the other hand data mining methods (e.g. decision trees and neural networks) and statistical analysis tools (e.g. linear regression) require that the data be represented as a feature vector $(x_1,\ldots, x_n, y)$ of n attributes for each observation. In the example above the important information about past purchases is stored in a separate table with potentially multiple transactions per customer and the number of entries varies for each customer. For some inactive customers, only one past purchase may be available whereas, for some highly active customers there could be more than hundred. If the objective of a marketing expert is to identify those customers who are likely to buy a new product after a direct marketing action, we would like to identify those customers who are most likely to respond to the direct marketing based on their past purchases. Faced with this situation, a marketing expert would manually have to generate the features x_i by joining both tables and aggregating the information about the sets of past purchases. A typical aggregate is, for instance, "amount spent within the last 2 months".

Not only does this manual feature construction require significant technical expertise (e.g., knowledge of SQL), but also strong prior ideas what kind of features might be important. The manual process of feature construction is very time consuming and becomes infeasible for a large number of tables.

This work provides a framework for the automated construction of relevant features for data mining from multiple relational tables. We use the term data mining in a very broad sense including machine learning and statistical analysis. Our approach will however be focused on predictive tasks (e.g., regression, classification, and probability estimation). Throughout the paper we will use the words *features* and *attributes* as synonyms.

A Transformation-Based Framework for Relational Data Mining

Our framework as presented in figure 1 takes a transformation-based approach to data mining, where learning is perceived as a sequence of successive steps of data transformation. In the first step T1, we construct features from relational tables. The result is a feature vector that is augmented in T2 and can be used in the last step T3 to learn a predictive model. The transformation T2 creates new non-relational features from mathematical combinations and transformations of existing features. A typical example is the introduction of interaction effects and higher order terms in linear models and re-scaling of features using logs (e.g., $x_{new} = \ln(x_{old})$).

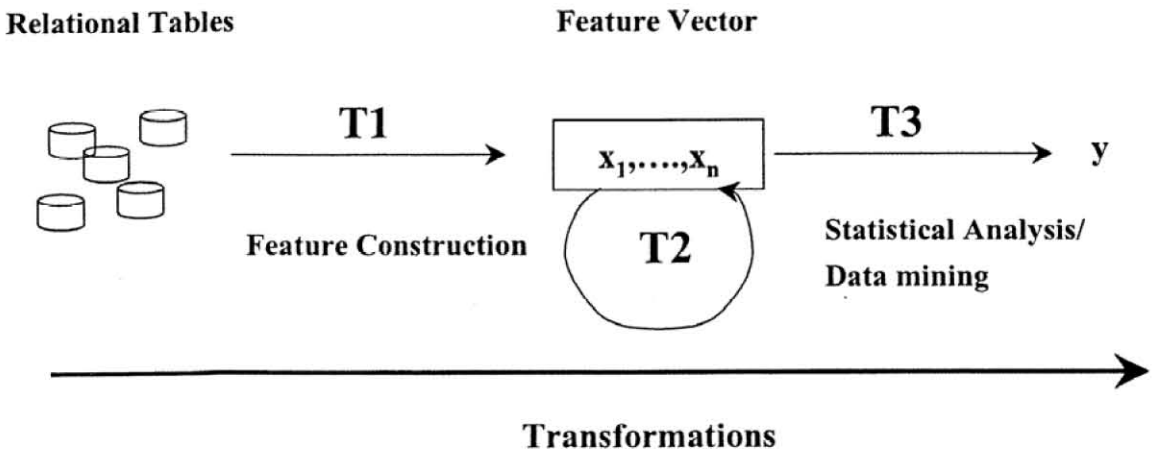


Figure 1: Transformation-based framework of data mining

The focus of our work is the feature construction step T1. The traditional manual approach to relational

data analysis has been expert-guided feature construction T1. It is possible to view the process of feature construction from relational data as a special case of preprocessing. However we argue that this step is extremely important and should receive its due attention as a main part of the modeling effort, as noted by Langley and Simon (1995) among many others.

De Raedt (1998) showed that for most relational databases (no recursive concepts) complete transformations (without loss of information) are possible, but result in an exponentially growing feature space with potentially sparse feature vectors. However the objective of the transformation T1 is not a complete transformation but rather the extraction and creation of valuable features for the modeling task.

Constructing Features from Relational Data

Figure 2 shows our modular framework of feature construction (T1) as a three-step procedure. We will use the expression "target relation" to refer to the table that contains the objects of interest for which a prediction will be made. In the previous example it is the customer table since we want to predict the probability that a customer will buy the product.

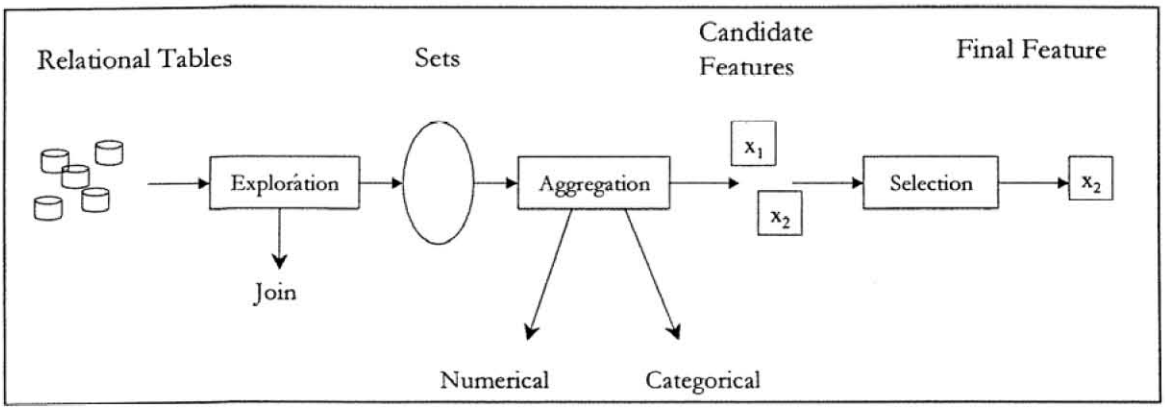


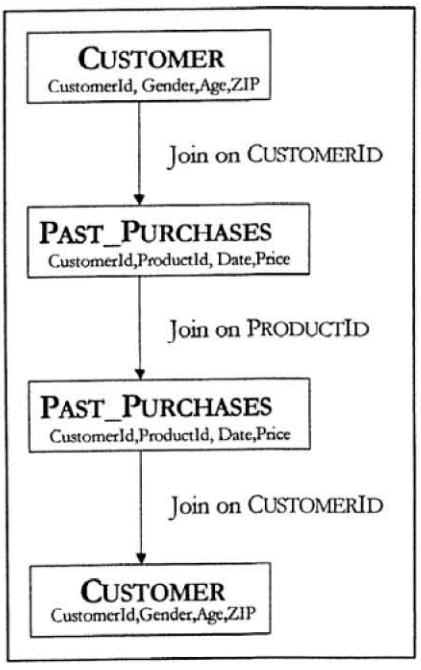
Figure 2: Framework for feature construction from relational tables

In the exploration step we join the target relation with other tables in the database that have at least one key attribute in common with the target relation. The second step is the application of aggregation methods to the results of the join. Each of those aggregation methods generates one value (either numeric

or categorical) per row in the target relation. After this generation of new features a selection procedure identifies valuable features for the modeling task T3. The three steps are described in more detail in the following paragraphs.

Structural Exploration

For the target relation $R_T(a_1,\ldots, a_i,\ldots,a_n)$ there is a set of background relations R_B $(b_1,\ldots, b_j,\ldots,b_n)$ which can be related to R_T, given a key attribute. Formally, the operation is a join $a_i=b_j$, where we only select the attributes $(b_1,\ldots, b_i,\ldots,b_n)$ from relation R_B. Depending on the cardinality of the relationship between the two relations, this may produce the same number of objects (as in the case of a 1-to-1), or a variable number (as in the case of 1-to-n). For example, two tables **CUSTOMER** and **PAST_PURCHASES** could be joined on the key CUSTOMERID (see Figure 3). The result of the join has multiple rows for each customer and the number of rows is equal to the number of transactions the customer has performed in the past. Conceptually there is a tree of all possible joins (potentially over multiple tables) with the target relation being the top of the tree. The nodes in layer of depth n correspond to all possible joins over n relations.



Each path from the top to a node involves multiple joins and generates one table. The only requirement is that every node has to share at least one key variable with one of the nodes in its path. It is possible to explore this tree using any standard tree-search algorithm (e.g., breadth-first search). Figure 3 shows the tree (after pruning identical joins) of depth 3 for the marketing example assuming that available keys are CUSTOMERID and PRODUCTID. The 3 joins corresponding to the full path would return the sets of all customers that bought at least one product identical to any one bought by the customer in the target relation.

Figure 3: Exploration tree for marketing example

Aggregation:

Structural exploration produces a table with a set of entries for each object in the target relation as shown on the example of customer transactions. The task of the aggregation operators is to construct single-valued attributes from each column of those sets. Depending on the data type, different aggregation operators can be applied. Figure 2 shows the 2 main groups of aggregation operators based on the data type (categorical or numerical) of the column. The column "Price" of the **PAST_PURCHASE** table, for example, is numeric and can be aggregated using simple sample statistics such as median, mean, min, and max. Categorical aggregation can involve the creation of dummies or counts for each value. One can however extend the current approach with much more complex operators like estimation of set- and density-distances (e.g., Kullback-Leibler) to a reference set.

Selection:

The aggregation step can create a significant number of potential features. Most predictive data mining methods show decreasing performance as the dimensionality of the input space increases ("the curse of dimensionality"). It is therefore useful to implement a feature selection step that assesses the value of a feature for the modeling task. It is a design question how much effort should be put into the evaluation of the usefulness of a feature since it involves a significant computational effort. In the simplest case this can be done using only the particular feature in question. The disadvantage is that one might miss features that are conditionally predictive. The decision as to how much computational effort is optimal depends also on the desired run time behavior, the choice of model class, and domain properties such as number of tables.

Implementation: ACORA

We have implemented a prototype of our feature construction framework called **ACORA** (**A**utomated **C**onstruction **O**f **R**elational **A**ttributes). The general philosophy is to allow the user to provide as rich a specification as he or she wants and to provide additionally default heuristics for all things the user does

not want to specify. This flexibility allows the comparison of methods and enables users with different technical competencies to interact with the system. The goal is to develop robust mechanisms that can guaranty good performance on a number of domains and prediction tasks. The following table gives a short overview of the system parameters and guidelines for the development of heuristics.

Heuristic for structural exploration	The simplest heuristic is breadth-first tree traversal. There are a number of more elaborate schemes that prune the search space aggressively. This is of particular importance if the number of tables is large.
Heuristic to select aggregation operators	Operators are constrained by the attribute type. However other factors that determine the appropriateness of an operator are for instance the average and variance of the set sizes.
Selection criterion	Data mining methods differ in their ability to learn from high-dimensional input spaces with many irrelevant features. Selection should be more aggressive for neural network models than for decision trees. In the simplest case the selection can be based on a minimum performance of a model that uses only one attribute to predict the target.
Stopping criterion	Given the large number of possible joins and features, there should be a limit to the creation of features. Stopping can be based on the tree depth N, the model performance (at a high computational cost), or on the number of features (e.g., the ratio of observations to features should be at least 5 for linear models). If the selection is not sufficiently aggressive, the stopping criteria might be reached without sufficient exploration of the feature space. A heuristic that takes both factors into account could create a high number of features, rank them based on the selection criteria and pick the top k features.

Table 1: System Parameters

One open research question is the degree of automation that can be achieved without losing much performance on a number of domains. Of additional interest are the interaction effects of those heuristics on run time and prediction performance of various data mining methods.

The system is currently in a development stage with most of the structural exploration and aggregation implemented. We plan to provide interfaces to a number of machine learning techniques to evaluate our approach on a number of prediction tasks. In particular we will include decision trees (C4.5), logistic and linear regression. Our prototype currently requires at least a specification of all relational tables names and the types of their attributes. After reading in the table specification **CUSTOMER**(CUSTOMERID, GENDER, INCOME, ZIP) for the customer table and **PAST_PURCHASES**(CUSTOMERID, DATE, PRICE, PRODUCTID), the system identifies based on attribute name automatically all possible keys (CUSTOMERID and PRODUCTID) on which tables can be joined unless the keys were specified explicitly. The system architecture is modular and open to extensions with new aggregation operators.

Related Work

This work is related to a number of research areas: feature construction, inductive logic programming, propositionalization, OLAP, and learning from relational databases.

OLAP enables an analyst to create high-dimensional representations from joins of multiple relations and to select subsets and aggregations. The current capabilities of OLAP systems provide an infrastructure for data analysis but lack mechanisms for automated analysis and predictive modeling. Machine learning and data mining have developed a suite of algorithms for automated learning of predictive models from feature vectors (x_1, x_2, x_3, x_4...x_n). The reader is referred to a standard textbook (Mitchell 1997) for an overview of existing methods. Kramer et al. (1998) coined the term "propositionalization" for the transformation of relational data into a feature-vector (propositional) representation. There exist a number of systems (e.g., LINUS, STILL, REPART discussed in Kramer et al. 2001) that perform propositionalization on relational data automatically and apply a data mining method for prediction. However those efforts focus exclusively on binary features in form of logical clauses and are not capable of performing numeric aggregation. We are aware of a number of non-automated attempts to use aggregation for propositionalization. Knobbe (2001) applied SQL operators successfully to a banking domain. Morik and Brockhausen (1996) implemented a prototype for propositionalization called TOLKIEN as part of the MiningMart system.

There exist a number of specialized data mining algorithms that can learn classification models directly from multi-relational tables without intermediate feature construction. We are aware of two SQL extensions that enable users to learn predictive rules directly from relational data. The DBMiner system by Han et al. (1996) integrates a set of discovery modules into an SQL-accessible database. Imielinski and Virmani (1999) proposed MSQL as a modeling extension to SQL. Both methods require a very detailed specification of the model form. In the case of DMQL, the form of the rule has to be stated explicitly.

The user has to know all relevant features in advance in order to formulate such a rule. MSQL similarly requires all possible rules to be stored beforehand in a special table. Both methods lack a sufficient degree of automation to enable efficient data analysis with minimal user interaction. Morik and Brockhausen (1996) showed that the previous approaches are closely related to inductive logic programming (ILP) methods. ILP algorithms (see Muggleton and De Readt (1994) for a good introduction) are capable of learning relational classification models from multi-table data. The resulting model is a set of existentially unified first-order Horn clauses. Known disadvantages of ILP approaches are high sensitivity to noise, limited support for numeric features, high computational complexity and the limitation to classification tasks. This is a significant drawback for applications in business domains, since we need probability estimates for decisions involving expected cost-benefit tradeoffs.

Discussion

There are a number of dimensions along which our approach has the potential to improve over existing technology.

Automation: ILP and logic-based propositionalization are currently the only automated approaches to relational learning, but both require significant technical knowledge to specify search heuristics for particular domains.

Robustness: Current implementations of relational learners often have inherent search heuristics that are tailored to particular domains. Additionally, they are often criticized for low performance on noisy domains due their origins in logic. There is no consensus upon guidelines as to when an approach is likely to perform well.

Usability: All existing relational learners require substantial technical understanding of the implementation and very low-level specifications of search constraints. This will deter most domain experts from using such a system.

Probability estimation: One shortcoming of logic-based approaches is their inability to produce probability estimates, which is essential for decision-making based on cost-benefit analyses.

Run time: Most implementations of ILP as well as propositionalization suffer from unacceptable run time behavior. The presented feature construction approach can take advantage of a number of strategies to improve run time behavior. The structural exploration and aggregation can easily be parallelized by assigning each path to a different processor. Due to the modularity of operators it is comparatively simple to run a scaled-down version with fewer operators and a more aggressive pruning of the feature-search space. The use of sub-samples for feature selection can additionally improve the run-time behavior.

Modularity: The modular design of our framework might result in a less efficient implementation but improves the ability of a domain expert to provide at least partial guidance for smaller subtasks. The expert might be able to tell that mean aggregation is less useful than maxima. Additionally the modularity enhances the comprehensibility of the system.

Extensions

As the very next step, we will to conduct a thorough empirical comparison of our feature construction approach on a number of relational domains such as Web-logfiles from an online vendor (KDD Cup 2000), news stories on business co-occurrence (from Yahoo), patent references, initial public offerings, medical records, LINUX developer communication, contract killing, citation networks, movies, and nuclear smuggling. (Those datasets have already been obtained). We will compare the performance of feature construction in combination with decision trees and logistic regression against publicly available ILP implementation as well as no feature construction. Many of the domains mentioned above contain information about relationships between humans or business entities. Social Network Analysis (SNA) (Scott 1991) was developed in the social sciences for the analysis of the interaction of individuals, with particular focus on reputation and centrality measures. SNA has successfully been employed for example

for profiling in the investigation of money laundering (Sparrow 1991). SNA can easily be introduced into our framework using graph extraction as a method for structural exploration and centrality measures for aggregation.

Acknowledgements

This work is sponsored 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.

References

De Raedt, L. "Attribute Value Learning versus inductive logic programming: The missing links (extended abstract)," Proceedings of the Eights International Conference on Inductive Logic Programming, Springer Verlag, Berlin, 1998, pp. 1-8.

Han, J., Fu, Y., Wang, W., Chiang, J., Gong, W., Koperski, K., Li, D., Lu, Y., Rajan, A., Stefanovic, N., Xia, B., and Zaiane, O. R. "DBMiner: A System for Mining Knowledge in Large Relational Databases," Proceedings of the International Conference on Data Mining and Knowledge Discovery, 1996, pp. 250-255.

Imielinski, T., and Virmani, A. "MSQL: A query language for database mining," Data Mining and Knowledge Discovery 3(4), 1999, pp. 373-408.

Knobbe, A. J., De Haas, M., and Siebes, A. "Propositionalisation and Aggregates," Lecture Notes in Artificial Intelligence (2168), Springer Verlag, Berlin, 2001, pp. 277-288.

Kramer, S., Lavrac, N., and Flach, P. "Propositionalization Approaches to Relational Data Mining," Relational Data Mining, Springer Verlag, 2001, pp. 262-291.

Kramer S., Pfahringer, B., and Helma, C. "Stochastic Propositionalization of Non-determinate Background Knowledge," International Workshop on Inductive Logic Programming, 1998, pp. 80-94.

Langley, P., & Simon, H.A. (1995). Applications of machine learning and rule induction. Communications of the Association for Computing Machinery, 38(11), pp. 54-64.

Mitchell T. M. Machine Learning, McGraw-Hill, New York, 1997.

Morik, K. and Brockhausen P. "A Multistrategy Approach to Relational Knowledge Discovery in Databases," Proceedings of the 3rd International Workshop on Multistrategy Learning, AAAI Press, 1996, pp. 17-28.

Muggleton, S. "Inverse entailment and Progol," New Generation Computing (13), 1995, pp. 245-286.

Muggleton, S., and De Raedt, L. "Inductive Logic Programming; Theory and methods," Journal of Logic Programming (19,20), 1994, pp. 629-679

Scott, J. Social Network Analysis: A Handbook, Newbury Park, CA: Sage Publications, 1991.

Sparrow, M. K. "The application of network analysis to criminal intelligence: An assessment of the prospects," Social Networks (13), 1991, pp. 251-274.