Learning Relational Model Trees

Annalisa Appice
Dipartimento di Informatica
Università degli Studi di Bari
Via Orabona, 4 - 70126 Bari, Italy
appice@di.uniba.it

Promotor: Prof. Donato Malerba

A dissertation submitted in partial satisfaction of the requirements
for the degree of
Doctor of Philosophy in Computer Science
in the Graduate Division of the
University of Bari, Italy

Credits

This dissertation was typeset using these shareware programs:

- TeXnicCenter 1 Beta 6.21
  available at: `http://www.texniccenter.org/`
- MikTEX 2.1
  available at: `http://www.miktex.de`

Thesis Supervisor

Prof. Donato Malerba

Chairperson of the Supervisory Committee

Member of the Supervisory Committee

Member of the Supervisory Committee

Submitted *January 2005*

Contents

List of Figures

List of Tables

List of Symbols

The following list indicates the meaning of symbols and acronyms which are commonly used throughout the text.

AOC	Area over Regression Error Characteristic (REC) curve
$Avg.RMSE$	Average square Root Mean Square Error
D	A set of data
$DBMS$	DataBase Management Systems
E	Expected value
G	A regression selection graph
γ_τ	Grafting operator
ILP	Inductive Logic Programing
KDD	Knowledge Discovery in Databases
MSE	Mean Square Error
MAD	Mean Absolute Deviation
$Mr-SMOTI$	Multi-Relational Stepwise Model Tree Induction
N_τ^I	Set of internal nodes in the tree τ
$N_\tau^{I_R}$	Set of regression nodes in the tree τ
$N_\tau^{I_S}$	Set of splitting nodes in the tree τ
N_τ^L	Set of leaves in the tree τ
π_τ	Pruning operator
$\Re$	The set of Real numbers
R	Resubstitution Error
R^2	Coefficient of determination
REP	Reduced Error Pruning
REG	Reduced Error Grafting
$\rho(t)$	Evaluation measure for regression node t in a SMOTI tree
$\sigma(t)$	Evaluation measure for splitting node t in a SMOTI tree
$SMOTI$	Stepwise Model Tree Induction
τ	A model tree
$TDIMT$	Top-Down Induction of Model Trees

Acknowledgments

A dissertation does not just appear out of nowhere, and although it is supposed to be a contribution by one person, I have been fortunate enough to have had the support of so many people and without it this would not have been possible. There are lots of people I would like to thank for a huge variety of reasons. Over these years I have learned a lot but I have also found good friends who have influenced my life.

First of all I would like to thank my supervisor, Donato Malerba, whose continual support and encouragement have kept me going over the last years first as a final year undergraduate project student, and then as a PhD student. Donato is undoubtedly the person who has had the strongest influence on this work. His tireless help and enthusiasm have inspired and challenged my work over the years and I have grown through his mentoring.

I would like to thank Floriana Esposito for constantly encouraging my research and giving me the opportunity to work in LACAM (Knowledge Acquisition and Machine Learning Lab) laboratory. I have immediately discovered that LACAM is not only a machine learning laboratory but a group of people whose invaluable professional and personal qualities make them an highly enjoyable company at work and free time. Most of them contributed in their own way to this work. I first want to mention Michelangelo Ceci. He joined the group around the same time I did and he is the phD student with whom I have co-operated most closely. Our discussions and our collaborations on several topics have strongly affected this work. Moreover, Antonietta Lanza and Francesca Alessandra Lisi contributed to my research on topics related to spatial data analysis. I specially thank Antonietta for all her useful comments and suggestions. A special word to Oronzo Altamura and Antonio Varlaro. They encouraged me in persevering my research work also when I was completely demoralized. I am specially indebted to Margherita Berardi for tolerating my complaints, encouraging my work and offering biscuits and nuts when I was hungry but too busy to go out and buy something to eat and Marco Degemmis for generously loaning me his desk (books, papers, wire, computer, pens, food, etc.). Their friendships made things much easier. Thanks to Paolo Buono and Pasquale Lops for all nice moments that allowed me to escape from my research preoccupation and Domenico Sacchi for his technical assistance. Thanks to Maria Teresa Basile, Nicola Di Mauro, Nicola Fanizzi, Stefano Ferilli and Oriana Licchelli. They share offices close to the office where I work and patiently answer my countless questions about ILP and LaTex. They have been an enjoyable company in several travels we

waged for participating to some conferences.

Part of this work was influenced when I was visiting the Machine Learning Group of University of Bristol. I thank my host Peter Flach and all his collaborators and students for making my experience in Bristol productive and enjoyable. A special thanks to all participants of Machine Learning Reading Group of Bristol, where I found many useful suggestions for my personal research.

I would like to thank Hendrik Blockeel (Department of Computer Science, Katholieke Universiteit, Leuven, Belgium) for providing mutagenesis and biodegradability dataset and giving me useful suggestions in using TILDE-RT; Jim Petch, Keith Cole and Mohammed Islam (MIMAS, University of Manchester, England) and Chrissie Gibson (Department of Environmental and Geographical Sciences, Manchester Metropolitan University, England) for providing access to census data and digital OS maps of Stockport Manchester.

Most importantly of all, I would like to thank my parents, brother and friends for supporting me in last years. It is through their encouragement and care that I have made it through all the steps to reach this point in life, and I could not have done it without them.

Abstract

The desire of knowledge, like the thirst of riches,
increases ever with the acquisition of it.
Laurence Sterne, Tristram Shandy (1760)

Developments in sensing, communications and storage technologies made it possible to collect and store large amount of business and scientific data. The process of discovering nuggets of knowledge in this data is the focus of the rapidly growing field known as *Knowledge Discovery in Databases* (KDD). *Data mining* is the central step of this process consisting of applying computational techniques that, under acceptable computational efficiency limitations, discover patterns (or even models) that are interesting or valuable over large datasets. Other steps in the KDD process include data selection and preparation as well as pattern evaluation.

Data mining research draws upon several research areas that is statistics, machine learning and databases. This justifies the existence of multifaceted contributions in data mining research. These contributions may be classified according to the kind of database to be mined (i.e. transactional, relational, object-relational, object-oriented, etc.), the kind of knowledge to be discovered, the kind of techniques to be applied and the kind of applications. Focusing on techniques they strongly depend on the representation of data that eventually mismatches the data model adopted by the database to be mined.

Classical data mining techniques look for patterns in a single table of data, i.e., data is represented as fixed-length vectors of variable values where each variable assumes only a single, primitive value. This representation is the traditional one adopted in statistics that makes it possible to devise efficient algorithms. However, real-world data is seldom in this form. Rather data is stored in multiple tables of a relational database.

To overcome limitations of classical data mining and deal with information about several object types scattered in separate tables of a relational database, a large body of research has recently devoted to *multi-relational data mining*. Multi-relational data mining cannot be simply defined as data mining in relational databases, but it denotes the study of methods and techniques to discovering multi-relational patterns over multi-relational data. This emphasizes that relational model is an invariant of the discovery process.

Multi-relational data mining techniques have been developed within the area of *Inductive Logic Programming* (ILP) that provides functionalities to navigate relational structure of data and generate potentially new form of evidence, not readily

available in flattened single table representation. Data in ILP is expected to be represented in Horn clausal logic, a subset of first-order logic which is implemented in the logic programming language Prolog. Unfortunately the use of first-order logic as a representational language entails a little attention to data stored in relational databases and how the knowledge of the data model can help to guide the search process. A solution is to mine multi-relational patterns taking advantage from a *tight-integration* with large relational databases and implementing ILP methods as special instances of a multi-relational data mining framework where both attribute-value and structural information (e.g. foreign key associations) embedded in the database schema are explicitly exploited.

In this dissertation, we focus on *regression* that is a fundamental task in data mining where the goal is to examine samples of past experience with known continuous answers (labels) and generalize in future cases through an inductive process. We explore different aspects of the induction of tree structured regression models from data and focus the attention on model trees that is trees associating multiple function to each leaf. A new method for the data-driven construction of model trees is presented, namely *Stepwise Model Tree Induction* (SMOTI) method. Its main characteristic is the induction of trees with two types of nodes: regression nodes, which perform only straight-line regression, and splitting nodes, which partition the feature space. The multiple linear model associated with each leaf is then built stepwise by combining straight-line regressions reported along the path from the root to the leaf. In this way, internal regression nodes contribute to the definition of multiple models and have a "global" effect, while straight-line regressions at leaves have only "local" effects. We also tackle the problem of simplifying model trees mined with SMOTI and propose two methods which are based on two distinct simplification operators, namely pruning and grafting. Experimental results show data model trees induced by SMOTI are generally simple and easily interpretable, and their analysis often reveals interesting patterns.

Nevertheless, SMOTI underlies data stored in a single table. This implies that aspects of internal structure of data cannot be processed and mined trees cannot refer to such a structural property. As a consequence, this might compromise the application of SMOTI in domains where the internal structure of the unit of analysis is naturally modeled by multiple tables of a relational database (e.g. chemistry, biology or geo-referenced data analysis). To overcome this limitation, we present a multi-relational data mining method, named *Multi-Relational Stepwise Model Tree Induction* (Mr-SMOTI), that upgrades SMOTI algorithm to multi-relational representations and takes advantage from a tight integration with database systems.

Finally, Mr-SMOTI is tested in some applications to both geo-referenced census data analysis and bioinformatics.

Chapter 1

Introduction

In many problems encountered in practice the prediction of an attribute associated with a case has a great potential payoff [WI98]. The two principal prediction problems are classification and regression. Samples of past experience with known answers (labels) are typically examined and generalized in future cases through an inductive process. In the usual setting, data is generated independently and identically distributed from an unknown distribution P on some domain $\boldsymbol{X}$ and labeled according to an unknown function g with range Y. The domain of g is spanned by m independent (or predictor) random attributes or variables X_i (both continuous and discrete), while the range of g is either a finite set of unordered category labels (i.e. classification) or a subset of real number $\Re$ (i.e. regression). Consequently, in regression problems, the dependent (target) variable Y is continuous. In this perspective, *regression analysis* can be loosely defined as a set of learning methods that receive a training sample $S = \{(\boldsymbol{x}, y) \in \boldsymbol{X} \times Y | y = g(\boldsymbol{x})\}$ and attempt to return a function f close to g on the domain $\boldsymbol{X}$. Closeness of f to g can be measured in many ways, for instance, by means of the expected square error. The discovered f can then be used either to predict behavior of the (unknown) target variable for given new values of the predictors or to reveal some yet unknown relationship in the domain S under analysis.

Regression problems have been deeply investigated in several scientific domains such as statistics, machine learning and data mining with interesting applications in social sciences, physical and biological sciences, business and technologies as well as humanities. Still in the majority of these studies, the regression model is assumed to be a linear combination of predictor variables and the coefficients of the combination are determined by the method of the least square regression [DS82]. Refinements and extensions to non-linear models are also well-known in statistics and applied in many real world applications.

However, classical statistical methods have several limitations. First, (non-) linear regression models are often hard to understand. Second, these regression models are based on the assumption that all predictor variables are equally relevant in the whole sample space. Third, the least square method does not allow prior domain knowledge to be used in the construction of the regression model. In addition,

standard linear regression has obvious drawbacks due to the linear structure that is imposed on data.

To face some of these issues and deal with data having more complex data structure, sophisticated techniques have been developed. Non-parametric approaches to regression are able to tackle a wide range of regression problems by not imposing any a-priori defined global form to the regression surface [Har90]. Consequently, they do not try to fit all training data with one single model, but assume a functional form only at local level. For instance, in *tree structured* approaches a regression model is typically *top-down* induced from training sample in form of a tree consisting of a hierarchy of nodes starting in a top node that is the root node. Each node is generally associated with a logical test on predictor variables with exception of leaves (i.e. bottom nodes in the hierarchy), which contain the prediction functions of the tree-based regression model. *Regression trees* associate a constant with each leaf so that the prediction is the same for all sample data falling in the same leaf. This means that they approximate the function g by means of a piecewise constant one. A generalization of regression trees is represented by *model trees*, which associate leaves with multiple (linear) models that is, they approximate the function g by a piecewise (linear) function. In model trees different values can be predicted for sample data falling in the same leaf.

Tree structured regression models have some nice properties such as the low computational complexity, the wide range of applicability in several research fields as well as the comprehensibility of mined regression models due to the use of a symbolic representation of the regression surface. However, as these models do not assume any fixed form of the regression surface, they can easily generate too many local areas to approximate the unknown surface and overfit training data. Furthermore, overfitting may lead to lower predictive accuracy on unreliable or noisy data. To keep the overfitting problem under control and deal with noise in training data, sophisticated simplification (pruning) methods have been developed. Simplification is usually regarded as a search through the space of all possible simplified tree of an overly large tree. Current approaches to this search problem are derived from those developed for decision trees [EMS97] or apply other criteria such as minimizing the binary description length of the tree model [RK98].

Although being one of the most successful regression analysis technique, the tree structured family of regression models also has its limitations. The central problem is properly the restrictive propositional data representation language due to the single table assumption [Wro01]. Training data must be represented as fixed-length vectors of variable values where each variable can have only a single, primitive value. They are generally stored in a table (or "relation" in database terminology), where each row corresponds to a *unit of analysis*[1] while each column corresponds to either a predictor variable or the target variable.

The situation is more complex in real world applications where units of analysis may involve separate units of observation, i.e. the entities that are observed and about which information is systematically collected in primary research. For in-

[1] In statistics, the unit of analysis is the basic entity or object about which generalizations can be made based on an analysis and for which data is collected in the form of variables.

stance, a major national study may use a form that collects information about each person in a dwelling and information about the housing structure, hence it collects data for two units of observation: persons and housing structures. From this data, different units of analysis may be constructed: household could be examined as a unit of analysis by combining data from people living in the same dwelling or family could be treated as the unit of analysis by combining data from all members in a dwelling sharing a familial relationship.

Units of observation describe objects sometime of different nature, therefore units of analysis cannot always be constructed by simply aggregating (i.e. min, max, count or average) properties of the corresponding units of observation. Conversely, it may be important to distinguish units of observation which represent *target objects* of analysis from other *target-relevant objects* and represent the relationships among them.

Modeling properties of these different objects as well as relationships among them is a key challenge in prediction problems that arise in complex domains, such as spatial domains [SSV+02] or biological domains [DBK+99], where the prediction of a property of a target object can be strongly affected by properties of target-relevant objects according to the relationships among them. For instance, if geographical data are jointly mined with census data to predict the number of inhabitants in a census enumeration district (ED), the target objects of analysis are the EDs (i.e., the smallest areal unit for which census data are published) while the target-relevant objects are the EDs forming the neighborhood as well as spatial objects in different geographic layers (e.g. urban areas, shopping areas or transport network) overlapping or intersecting the ED boundary.

The single table representation, supported by traditional regression analysis methods, appears totally inadequate in this case since different units of observation (EDs, urban areas, shopping areas, transport network and so on) may have different properties, which are properly modeled by as many data tables (*relational data model*) as the number of object types [KBSV99]. Moreover, relationships (e.g. topological, distance and direction relationships, which are implicitly defined by the location and the extension of spatial objects [Kop99]) among units of observation forming the same unit of analysis can be also explicitly modeled in a relational database by means of tables describing the relationship as well as foreign key associations between the table describing the relationship in question and the tables representing objects involved in the relationship.

In principle, it is also possible to consider a single relation reconstructed by performing a relational join operation on the tables. However, this approach is fraught with many difficulties in practice [De 02][Get01].

A solution can be found in resorting to the field of relational data mining [DL01a] which provides functionalities to navigate the relational structure and generate potentially new forms of evidence not readily available in a flattened single table representation. Discovered model (patterns) are multi-relational, that is, they involve multiple relations from a relational database. They are typically stated in a more expressive language (e.g. subsets of first-order logic) than patterns described on a single data table.

1.1 Objectives

The main objective of this dissertation is to revise the current state of art on top-down induction of model trees (TDIMT) in order to improve interpretability and accuracy of these regression models as well as to extend their applicability to practical problems where data is naturally stored in multiple tables of a relational database.

We start from the strengths and weaknesses of well known data-driven approaches to mine model trees and propose a new method named Stepwise Model Tree Induction (SMOTI). Its main characteristic is the induction of trees with two types of nodes: regression nodes, which perform only straight-line regression, and splitting nodes, which partition the training space. The multiple linear model associated with each leaf is then built stepwise by combining straight-line regressions reported along the path from the root to the leaf. In this way, SMOTI, solves the problem of modeling phenomena where some variables have a *global* effect while others have only a *local* effect. Indeed, internal regression nodes contribute to the definition of multiple models and have a global effect, while straight-line regressions at leaves have only local effects.

We then explore the idea of combining the stepwise construction supported by SMOTI with achievements of multi-relational data mining in and overcome limitations due to single table assumption. Hence, we illustrate how to upgrade the propositional SMOTI to relational setting and mine multi-relational model trees directly from data which resides in multiple tables of a tightly integrated Oracle® 9*i* relational database. Patterns associated with each node of the tree structure are multi-relational patterns since they may involve multiple tables from the training relational database.

Finally, we discuss an implementation of both SMOTI and its upgrade to multi-relational setting, named Mr-SMOTI, and evaluate them empirically.

1.2 Motivation and Contributions

The majority of research on mining model trees has reserved much attention to the propositional setting where training data is represented by a fixed set of single valued attributes. Some of the model tree induction methods developed are: M5 [Qui92], RETIS [Kar92], M5' [WW97], TSIR [Lub94], HTL [Tor97], which has been subsequently included in RT [Tor99], SUPPORT [CHLY94], which has been extended in GUIDE [Loh02], and SECRET [DG02].

All these methods perform a top-down induction of model trees (TDIMT) by recursively partitioning the training space. However, some of them (e.g., M5, M5' and HTL) first build the tree structure through recursive partitioning of the training data and *then* associate leaves with models. This means that the partitioning of training data (splitting stage) does not take into account the regression models that can be associated with the leaves (predictive stage). Consequently, the heuristic evaluation function used to select the best partition is computationally efficient, but it may compromise the discovery of the *correct* trees because of its incoherence with the linear model associated with leaves.

This problem is solved in RETIS, whose heuristic evaluation function used for a binary split minimizes a function of the mean square error (MSE) computed with respect to the regression planes found for both the left and the right child. In practice, for each possible partitioning the best regression planes at leaves are chosen, so that the selection of the optimal partitioning can be based on the result of the prediction stage. A different approach is followed in SUPPORT and SECRET, which reduce the computational complexity by transforming a regression problem into a classification problem, and then by choosing the best partition on the basis of computationally efficient evaluation functions developed for classification tasks.

A weakness of solution implemented in RETIS is that the regression planes involve all continuous variables. When some of the independent variables are linearly related to each other, that is, they are (approximately) collinear, several problems may occur [DS82]. First, if at least one of the independent variables is a perfect linear function of one or more other independent variables in the equation, the coefficients may not be uniquely determined. Second, estimates of the regression coefficients fluctuate markedly from sample to sample. Regression coefficients cannot be used as interpretive tools to evaluate the relative importance of the independent variables. Interestingly, problems due to collinearity do not show in the model's fit. The resulting model may have very small residuals, but the regression coefficients are actually poorly estimated. A treatment suggested in this case is deleting some of the variables in the full fitted model. Therefore, *variable subset selection* is a desirable part of regression analysis that is not supported by RETIS.

An additional problem of almost all TDIMT methods is that the regression model associated with a leaf is built on the basis of those training cases falling in the corresponding partition of the training data. Therefore, models in the leaves have only a local validity and do not consider the global effects that some variables might have in the underlying model. In model trees, global effects can be represented by variables that are introduced in the linear models at higher levels of the tree. However, this requires a different tree-structure, like that adopted in TSIR, where internal nodes can either define a partitioning of the training data or introduce some regression variables in the models to be associated with the leaves.

In this context, our main contribution with this work is to propose SMOTI as a means to overcome problems encountered in some existing TDIMT systems by exhibiting the following characteristics:

1. Induced model trees have two types of internal nodes: regression nodes, which perform only straight-line regression, and splitting nodes, which partition the training data. Leaves are always regression nodes.

2. A multiple linear model can be associated with each leaf. It involves all the continuous variables in the regression nodes along the path from the root the leaf.

3. Variables involved in regression nodes at top levels of the tree capture global effects, while those involved in regression nodes close to the leaves capture local effects.

4. The heuristic evaluation function is coherent with respect to the linear model associated with the leaves.

5. Only a subset of continuous variables may be involved in multiple linear models associated with the leaves, thus solving problems due to collinearity.

6. Induced model trees can be easily interpreted.

In addition, our method has been implemented as a module of the knowledge discovery system KDB2000[ACM02] (http://www.di.uniba.it/∼malerba/software/kdb2000/) that does not input training data from a file but interfaces a relational database. Nevertheless, SMOTI underlies the single table assumption and requires the training data to be stored in a single table. This implies that aspects of internal structure of data cannot be processed and mined trees cannot refer to such a structural property. As a consequence, this might compromise the application of SMOTI in domains where the internal structure of units of analysis is modeled by multiple tables of a relational database.

To overcome this restriction, one solution is to *mould* a relational database into a single table format that traditional attribute-value algorithms can handle [KHS01]. This approach corresponds with the concept of *propositionalization* in machine learning and has been employed into regression tasks as well. In [DTU95], the DINUS algorithm [LD94] is applied to transform a Datalog representation of a dynamic method into a propositional form (i.e., attribute-value pairs), so that the classical model tree induction method RETIS based on the single-table assumption can be applied (DINUS/RETIS). One way of obtaining an attribute-value representation is to create a single table by deriving attributes from other joined tables. However, this produces an extremely large table with much data being repeated, which is difficult to handle. A different approach is the construction of a single central table that summarizes and/or aggregates information that can be found in other tables. Also this approach has some drawbacks, since information about how the data were originally structured is lost. Therefore, a proper way of explicitly and efficiently dealing with multiple tables is necessary.

The idea of mining multi-relational regression models over data which resides in multiple relations has already been reported in [Dž95], where the multi-Relational Regression problem has been formulated in the normal ILP (Inductive Logic Programming) setting. Thus far, ILP has been proposed two approaches to solve multi-Relational Regression problems. An approach is based on *separate-and-conquer* (or sequential covering) strategy to build a set of Prolog clauses. The alternative approaches is based on the *divide-and-conquer* strategy to induce tree structured models and then translate these models into Prolog programs. Some examples of methods that follow the first approach are FORS [Kar95] [KB97] and FFOIL [Qui96], while three methods that follow the second approach are SRT [Kra96], S-CART [Kra99] [KW01] and TILDE-RT [Blo98] [BD98].

In contrast with DINUS/RETIS, these methods solve the multi-Relational Regression problem in its original form, and do not require transformation of the problem. Moreover, they can process relational background knowledge.

All these methods are based on processing *main-memory* data stored as *Prolog facts*. This requires some pre-processing to transform tuples in facts when data are originally stored in databases as in many real-world applications. Much of the pre-processing, which is often expensive in terms of computation and storage, may be unnecessary since that part of the hypothesis space may never be explored. Furthermore, in applications where data frequently changes, this pre-processing is frequently repeated.

Main-memory data processing is a common aspect of most data mining methods working not only in the multi-relational setting but also in the propositional one. This results in high performance for computationally intensive processes when enough memory is available to store all necessary data. However, most data mining algorithms are characterized by frequent access to data that satisfies some selection conditions. This suggests that for data intensive processes it may be useful to exploit powerful mechanisms for accessing, filtering and indexing data, such as those available in database management systems (DBMS).

In general, we may observe that little attention has been given to data stored in relational databases and to how knowledge of a data model can help to guide the search process [KBSV99]. A solution is to combine the achievements of the KDD (Knowledge Discovery in Database) field on the integration of data mining with database systems, with some results reported in the ILP field on how to correctly upgrade propositional data mining algorithms to multi-relational representations. For this purpose, we present a multi-relational data mining method, named Mr-SMOTI, that upgrades SMOTI algorithm to multi-relational representations and takes advantage from a tight integration with database systems.

From an inductive database perspective [IM96], this tight coupling also aims at supporting a direct and uniform access to both data and patterns stored in databases. Other equally important reasons are: i) the applicability of data mining algorithms to large datasets; ii) the exploitation of useful knowledge embedded in the data model available in the database schema free of charge, iii) the possibility to directly specify which data stored in a database have to be mined without any pre-processing.

1.3 Structure of the thesis

This dissertation is organized as follows. Chapter 2 presents an overview on knowledge discovery from database focusing on multi-relational approaches to data mining (MRDM) to deal with data scattered over multiple tables. We also provide a database perspective to KDD exploring the possibility of tightly integrating a data mining component with database systems. Finally, we exploit the achievements of ILP field on how to correctly upgrade propositional data mining algorithms to multi-relational representations. Indeed, considering the strong link with logics, it is not surprising that many algorithms for MRDM originate from ILP.

In Chapter 3 the problem of mining regression models is formally defined in both propositional and relational setting and related works in both statistics and

machine learning are reported.

In Chapter 4 the method SMOTI is introduced and its computational complexity is analyzed. Background and motivations of stepwise construction of models trees with both splitting and regression nodes are discussed. Some experimental results on both artificially generated datasets and benchmark datasets used for studies on regression and model trees are commented and a comparison with state of art model tree induction methods such as M5' and RETIS is performed. Regarding experiments on benchmark datasets, we also look for interesting patterns in model trees mined by SMOTI in order to reveal presence of variables that have a global effect in training data rather than a local one. Finally, we tackle the problem of simplifying SMOTI model trees in order to avoid overfitting of training data. We propose two methods which are based on two distinct simplification operators, namely pruning and grafting. Theoretical properties of the methods are reported and the effect of the simplification on several datasets is empirically investigated. Results are in favor of simplified trees in most cases.

In Chapter 5 we demonstrate how SMOTI can be upgraded toward multi-relational setting to mine model trees from data stored in multiple tables of a relational database. The resulting Mr-SMOTI exploits the stepwise strategy to build model trees with both regression and splitting nodes as well as detect the global or local effect of variables that arise in the domain to be modeled. However, differently from SMOTI, variables involved in both types of nodes can belong to different tables of a relational database. The join of these tables is dynamically determined on the basis of the database schema and aims at involving variables from several relations to build a predictive model for the target attribute. As a consequence, mined model trees are expressed by multi-relational patterns represented in a graphical language based on *selection graphs*, which can be translated into SQL, or equivalently into first-order formulae. Details of the tight integration of Mr-SMOTI with Oracle® 9*i* are also discussed.

In Chapter 6 some applications of Mr-SMOTI to both geo-referenced census data and bioinformatics are presented. The former involves data provided by the United Kingdom (UK) 1991 census where the goal is to investigate the migration phenomena (i.e. number of migrants) in Stockport census enumeration districts. The latter involves two datasets, namely Mutagenesis and Biodegradability, which are extensively used in ILP. Mutagenesis concerns the problem of predicting the mutagenic activity of molecules while Biodegradability concerns the investigation of the biodegradability in an aqueous environment under aerobic conditions.

Finally, Chapter 7 concludes by discussing the main contributions of this work and describing direction for future work.

Chapter 2

Relational Knowledge Discovery in Databases

The rapidly expanding amount of data gathered by collection tools, such as satellite systems or remote sensing systems has paved the way for advances in knowledge discovery and data mining. It can be argued that nuggets of useful information are hidden in masses of data and therefore the problem of how to turn collected data into useful information becomes a significant one. On the other hand, having reached sizes that defy even partial examination by humans, modern databases and collections of datasets are literally swamping users. This data firehouse phenomenon appears in many fields including science data analysis, medical and health care, corporate and marketing, and financial markets. Hence, (semi-) automatic methods for locating interesting information from data are useful.

On-line Analytical Processing (OLAP) [Cod93] offers tools of interactive data analysis by simply aggregating data and counting frequencies. However, OLAP does not perform any explorative modeling of data. In addition, traditional ad hoc mixtures of statistical techniques and data management tools are no longer adequate for analyzing the large collection of data that is typically stored in real-world databases.

In this context, knowledge discovery in databases (KDD) has emerged as a growing field of multidisciplinary research for discovering interesting/useful knowledge (models or patterns) from large databases. Data mining is the central step in this process, concerned with applying computational methods to discover patterns in data (i.e. generalization extracted from the data), while other steps in the KDD process include data selection and preparation as well as pattern evaluation.

Most of data mining techniques have been developed for data stored in the traditional matrix form, where rows represent units of analysis and columns represent variables. This representation, derived from statistics, makes it possible to employ matrix operations for representing several data analytic procedures quite succinctly and devise efficient algorithms. However, real-world data is seldom of this form. Rather relational databases are widely used. As a consequence, a large body of recent research has been devoted to the field of *relational data mining* in order

to discover *relational patters* in *relational data* scattered in multiple tables of a relational database.

Relational data mining techniques have been mainly developed within the field of Inductive Logic Programming (ILP) where data is expected to be represented in subsets of first-order logic such as Horn clausal logic. It is noteworthy that first-order logic makes ILP techniques able to fit relational data model. Nevertheless, they reserve little attention to data actually stored in relational databases and to how knowledge embedded in this data model can help to guide the search process. Indeed, most of them work on main-memory data stored as Prolog facts. Hence, some pre-processing is required to transform tuples stored in relational databases into facts.

A solution may be to exploit the bridge between ILP and relational database to correctly upgrade classical data mining methods to multi-relational representations as well as the achievements of KDD in tightly integrating data mining methods with databases.

This chapter introduces and illustrates the KDD process and focuses on the concept of inductive databases. Indeed, inductive databases are databases that in addition to data also contain patterns. This means that within the inductive database framework, KDD is modeled as in interactive process in which users can query patters as well as data by means of an inductive query language. This leads to combine data mining methods with database language such as SQL. We also explore how ILP can be adapted to KDD in order to search for patterns eventually expressed in relational algebra instead of logic. Such an algorithm strictly does not fall in the class of ILP algorithms but it does use techniques based on ILP. Finally, we investigate the achievements and drawbacks of multi-relational data mining approaches and draw some conclusions.

2.1 KDD and data mining

The term KDD was coined at the first KDD workshop in 1989 to emphasize that knowledge is the end product of a data-driven discovery. Indeed, KDD was initially defined as:

"*the non trivial extraction of implicit, previously unknown and potentially useful information from data*" [FPM91].

However, a revisited version of this definition states that:

"*knowledge discovery is the non-trivial process of identifying valid, novel, potentially useful, and understandable patterns in data*" [FPSSU96].

Here *data* is a set of facts (e.g. cases in a database), while *pattern* is an expression in some language representing a parsimonious description of a subset of data or a model (i.e. characterization of the global dataset) applicable to that subset. *Not trivial* means that a pattern is not straightforward computation of predefined quantities such as the average value of a set of numbers but some search or inference should be involved in its discovery.

In this context, the term KDD refers the overall process of discovering useful

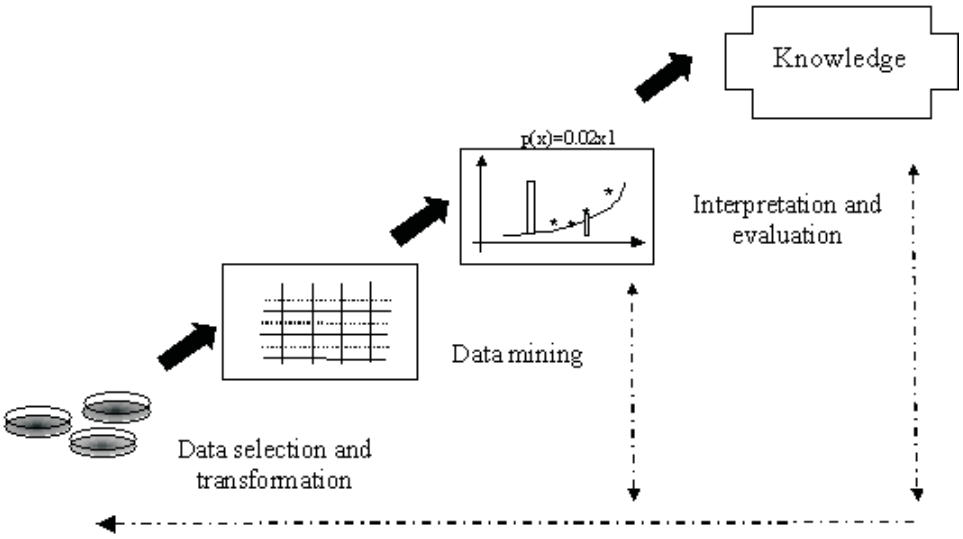


FIGURE 2.1: Knowledge Discovery in Database process.

knowledge from data while data mining refers to a particular step in this process. More precisely, data mining consists of applying computational techniques that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns (or even models) over the data.

The two primary goals of data mining in practice tend to be *prediction* and *description*. Prediction involves using some variables or fields in databases to predict unknown or future values of the other variable of interest. Description focuses on finding human-interpretable patterns describing data.

In a sense, data mining is a central step in KDD process. However, blind applications of data mining methods can be a dangerous activity easily leading to the discovery of meaningless patterns. Hence, the additional steps concerning with data selection and preparation, data mining and pattern evaluation are essential to ensure that useful knowledge is derived from data (see Figure 2.1).

KDD has evolved as an interdisciplinary field which merges together machine learning, statistics, databases, knowledge acquisition, data visualization, high performance computing and expert systems. Indeed, KDD overlaps with machine learning and pattern recognition in the study of data mining algorithms for modeling data and extracting patterns, but it puts a strong emphasis on working with large datasets stored in real-world databases. Thus scaling properties of algorithms are of fundamental interest. It also focuses on discovering patterns that can be interpreted as *interesting knowledge*. Since many (often infinitely) patterns or models may be discovered from the same data, criteria for deciding what structures constitute *knowledge* are needed. Classical measures such as *validity* (e.g. estimated predictive accuracy on new data) or *utility* (e.g. gain in running time or accuracy due to the better prediction) can be adopted from decision analysis or statistics. Conversely, other measures such as *novelty* and *understandability* are more difficult to define. For instance, understandability can be defined by simplicity (e.g. the

number of bits to describe a pattern). The *interestingness* is usually taken as the overall measure of pattern value combining validity, novelty, usefulness and simplicity [PM94] [ST95] and a pattern can be considered as knowledge if its evaluation exceeds some interestingness threshold.

KDD also has much in common with statistics, particularly exploratory data analysis. Indeed, a statistical approach offers method for quantifying the inherent uncertainty resulting when some patterns are inferred from a sample of an overall population. KDD systems (e.g. KDB2000 [ACM02] and WEKA [WF00]) embed statistical procedures for sampling and modeling data, evaluating hypotheses as well as handling noise. In contrast to traditional approaches in statistics, KDD approach typically employs more search in model extraction and operates in the context of large datasets with richer data structures.

Finally, another area related to KDD is data warehousing that refers to the popular business trend for collecting and cleaning large collections of data. The goal is to provide views of collected data that are non practical for individual transactional sources and make them available for on-line analysis and decision support. The most popular approach for analyzing data warehouse is OLAP. However, OLAP tools are targeted toward simplifying and supporting data analysis, while KDD tools aim at automating as much of the process as possible [FPS96].

2.2 A database perspective on KDD

The current generation of database systems have been designed mainly to support business applications. The success of Structured Query Language (SQL) [Cod72] for relational databases has capitalized on a small number of primitives sufficient to support a vast majority of such applications. However, these primitives are not completely able to capture the emerging family of novel applications dealing with knowledge discovery. Indeed, conventional database systems provide answers if the answers are in the databases. Deductive databases try to overcome this limitation by adding a database the capabilities for drawing deductive consequences from data using a rule base, which is called the intensional component of the database. When data mining methods remain not integrated with DBMSs only very limited domain knowledge is employed in the KDD process. This leads to an emerging new frontier for database research that states the growing need for Knowledge and Data Discovery Management Systems (KDDMS) to manage KDD applications just as DBMSs successfully manage business applications [IM96].

Queries in KDDMS are more general than SQL queries since queried objects are far more complex than tuples stored in a relational database. To achieve this, we need to distinguish between a KDD object and a KDD query. A KDD object is a pattern that may be a rule, clustering, classification or regression model. Rules are essentially probabilistic formulas or multidimensional correlation. Clustering refers to collections of sets of objects such that each set consists of objects grouped together by similarity according to a similarity measure. Conversely, classification and regression models are typically obtained using neural networks, decision or

regression trees, etc. A KDD query is a query that returns a set of objects, which can be either KDD objects or database objects such as tuples. The KDD objects may not exist a-priori, thus querying them requires their run time generation, but they may be simply retrieved in the case they have been previously generated and stored in a database.

This view gives rise to *inductive database* [Man97] that does not only store data but also patterns. As a consequence, it allows to answer queries that require drawing inductive inferences and deriving plausible conclusions. In addition, it leads to the tightly integration of conventional databases, databases of patterns and methods of inductive mining.

Hence, an inductive database $I(D, P)$ consists of a data component D and a patten component P. The assumption is that both data and pattern components D and P are sets of sets. This is motivated by an analogy with traditional relational databases. A relational database can be considered as a set of relations where each relation is a set of tuples. So relational databases are exactly sets of sets. This assumption is further justified because data mining step is often coping with different datasets (e.g. training set and testing set) [De 02].

Everything that is commented for data component D also applies to pattern component P. This means that during the knowledge discovery process one will often work with different sets of patterns, each of which may reside in the inductive database. These new sets may correspond to hypotheses constructed on different datasets during cross-validation or under various parameter setting as well as post-processed patterns. The consequence is that, within inductive databases framework, KDDMS should be able to persistently store and manage the KDD objects as well as provide the ability to query them. Thus, querying has two major roles: generation of new KDD objects and retrieval of the ones previously generated.

Obviously, this requires a new generation of query languages (in a wider sense, of DDL, DML too), called *KDD query languages* or *data mining query languages* that allows us to reinterpret the KDD process as a query processing. KDD queries allow the data miner to perform data selection and transformation (this is supported by conventional databases), specify patterns of interest, retrieve patterns of interest already stored in the database or induce new patterns of interest from pre-processed data, evaluate induced patterns on fresh data, specify domain knowledge and constraints for the inductive process and keep persistency of induced patterns.

It is worthwhile that a KDD query language also satisfies the closure principle [BKM99] which states that a query in relational database does not care if the argument is the original database or the result of another query. Thus, a KDD query may use multiple layers of nesting that involve several times both database objects layer and KDD objects layer. A KDD query can be also nested in a regular relational query and relational queries should form a proper subset of KDD queries. Some examples of KDD queries:

1. Discover the strongest rule (according to some predefined criteria) with some user specified attributes occurring in the antecedent and/or consequent of rule. Then find all tuples that violate this rule and discover a rule satisfied

by such set of tuples.

2. Find tuples belonging to the largest cluster in a clustering built according to a user specified similarity measure.

3. Generate a regression model (e.g. a regression tree) to predict the value of a continuous attribute trained on a user defined training set (specified through a SQL query) with user specified predictor attributes and target attribute. Then predict the unknown value of the target attribute in a testing set and use this set to mine a new regression model.

This inductive database view confirms that data mining cannot be simply another synonym for statistical data analysis or inductive learning on large datasets. The key new component is the ad hoc nature of KDD queries and the need for efficient query compilation into a multitude of existing and new data analysis methods.

Hence, data mining builds upon the existing body of work in statistics and machine learning, but it also provides completely new functionalities. The main task of the inductive databases is exactly to efficiently answer the queries expressed in some data mining query language and, as a consequence, the need of speeding up knowledge discovery in databases.

2.2.1 Knowledge discovery query languages

Several data mining (or knowledge discovery) query languages have been proposed in the literature. MSQL is a rule query language proposed by Imielinski and Virmani [IV99] for relational databases. It satisfies the closure property, that is, the result of a query is a relation that can be queried further. Moreover, a cross-over between data and rules is supported, i.e. there are primitives in the language that can map generated rules back to the source data, and vice-versa. The combined result of these two properties is that a data mining query can be nested within a regular relational query.

Another data mining query language for relational databases is DMQL [HFW+96]. Its design is based on five primitives, namely the set of data relevant to a data mining task, the kind of knowledge to be mined, the background knowledge to be used in the discovery process, the concept hierarchies, the interestingness measures and thresholds for pattern evaluation.

GMQL is based on DMQL and allows the user to specify the set of relevant data for a spatial mining process, the type of knowledge to be discovered, the thresholds to filter out interesting rules, and the concept hierarchies as the background knowledge [Kop99]. In the process of selecting data relevant to the mining task, the user has to specify (i) the relevant tables, (ii) the conditions that are satisfied by the relevant objects and (iii) the properties of the objects which the mining process is based on. Conditions may involve spatial predicates on topological relations, distance relations and direction relations. Although data can be selected from several tables, mining is performed only on a single table which results from an SQL query (single table assumption). GMQL queries can generate different types of knowledge, namely characteristic rules, comparison rules, clustering rules, classification

rules and association rules. All observations reported above for DMQL applies to GMQL as well.

DMQL is an object-oriented extension of DMQL [ESF01]. The design of ODMQL is based on the same primitives used for DMQL, so the main innovation is that each primitive is in an OQL[1]-like syntax. Path expressions are supported in ODMQL, while more advanced features of object-oriented query languages, such as the use of collections and methods, are not mentioned.

Finally, a spatial data mining object query language, named SDMOQL [MAC03], has been proposed to solve problems, due to the integration of different technologies, such as data mining, object-oriented (OO) DBMS, and Geographical Information System (GIS) in INGENS [MEL+03]. INGENS is a prototypical GIS that embeds data mining facilities support sophisticated end users in their topographic map interpretation tasks. SDMOQL is based on OQL and interfaces an ILP system, named ATRE [Mal03] that works with first-order representations of both input data and output patterns. It separates the logical representation of spatial objects from their physical or geometrical representation. SDMOQL does not allow users to formulate nested queries, however it supports some form of cross-over between data and mined rules. This characteristic is naturally supported as the result of the integration of deductive inferences for extracted rules and data selection queries expressed in OQL. Moreover, to face the challenging problems deriving from natural complexity of spatial domains seven primitives have been considered as guidelines for the design of SDMOQL. They are:

1. the set of objects relevant to a data mining task,
2. the kind of knowledge to be mined,
3. the set of descriptors to be extracted from a digital map,
4. the set of descriptors to be used for pattern description,
5. the background knowledge to be used in the discovery process,
6. the concept hierarchies,
7. the interestingness measures and thresholds for pattern evaluation,
8. the expected representation for visualizing the discovered patterns.

In this way, SDMOQL, differently from GMQL is able to separate the set of automatically generated (primitive) descriptors from the set of descriptors used to specify the patterns in geographical objects of interest for the application. An additional design principle is that of visualization, since in spatial data mining it is important to specify whether results have to be visualized or presented in a textual form.

[1]OQL is the standard defined by ODMG (Object Database Management Group) for designing object oriented models (www.odmg.org).

2.2.2 Speeding up knowledge discovery in databases

Scalability and efficiency are crucial in KDD due to the huge amount of data stored in real-world databases. Most data mining (or KDD) systems still process data in main memory. This results in high performance for computationally intensive processes when enough memory is available to store all necessary data. However, a common aspect of many data mining algorithms is their frequent access to data that satisfies some selection conditions. For data intensive processes, it is important to exploit powerful mechanisms for accessing, filtering and indexing data, such as those available in database management systems (DBMS). This motivates a tight coupling between data mining methods and database systems.

In the inductive database perspective, this tight coupling also aims at supporting a direct and uniform access to both data and patterns stored in databases. Other equally important reasons are: i) the applicability of data mining algorithms to large data sets; ii) the exploitation of useful knowledge of data model available, free of charge, in the database schema, iii) the possibility to specify directly what data stored in a database have to be mined, without any pre-processing.

The last reason is even more justified by the emergent trend in KDD research, namely multi-relational data mining [DL01a], which looks for patterns that involve multiple relations of a relational database. Thus data taken as input by multi-relational data mining systems typically consists of several tables and not just a single one. Conversely, the single-table assumption [Wro01] forces the user of traditional data mining systems to perform complex SQL queries in order to compute a single table whose rows (or tuples) describe independent units of analysis.

Some examples of integration of data mining and database systems are presented in [STA98] for association rules, in [OC00] for clustering and in [SD01] for decision trees. In [MS03], a system named MiningMart has been proposed for approaching the knowledge discovery in database by building upon database facilities and integrating data mining algorithms into the database environment. In all these works it is advocated the importance of implementing some data mining primitives [FL96] to implement them by exploiting DBMS extension facilities, e.g. packages, cartridges or extenders.

In [ACL+03] a package implemented in PL-SQL has been presented to support the extraction of spatial relations between geographical objects stored in an Oracle Spatial database. This is also a rare example of multi-relational data mining system, named SPADA [LM04], (loosely) integrated with an object-relational spatial database. Other two examples of tight integration of multi-relational data mining systems with a database are MRDTL [Lei02] and SubgroupMiner [KM02]. These three examples refer to the tasks of association rule mining, classification (with decision trees), and subgroup discovery, respectively.

2.3 Inductive logic programming for KDD

Inductive Logic Programming (ILP) is a research area situated at the intersection of several scientific domains. It is built on *logic programming* whence it borrows

a knowledge representation formalism but it also exploits achievements in *machine learning* [ND97] that are often relevant to the field of knowledge discovery and particularly data mining.

Logic programming is a programming paradigm in which a program consists of first-order logic formulae. The most common logic programming language is Prolog [Kow88], whose engine can *deduce* those facts which are certainly true from first-order formulae assumed to be true.

An example of this deductive reasoning is the Aristotelian syllogism:
"*if all humans are mortals and Socrates is human then Socrates is mortal*".

The premises of the reasoning may be expressed in Prolog as follows:
$mortal(X)$:- $human(X)$. (i.e. for each X such that X is human then X is mortal)
$human(socrates)$. (i. e. Socrates is human).

Thereby, it is possible to query the Prolog engine whether Socrates is mortal (i.e. :- $mortal(socrates)$.) or for which X it can be proved that X is mortal (i.e :- $mortal(X)$.). In the former case, the engine answers "yes", while in the latter case it returns "$X = socrates$".

Machine learning can be trivially defined as the study of how to make machines learn. Learning is generally considered one of the most important ability that characterizes any agent (be it a human, animal or machine) to be called intelligent. Indeed, Langley (1996) defines the learning as:

"*the improvement of performance in some environment through the acquisition of knowledge resulting from experience in that environment*" [Lan96].

According to this definition, the learning process consists of two subtasks: acquiring knowledge and putting it to use. Focusing on the first task, we may consider the acquisition of knowledge as the inference of a general theory (i.e. knowledge) from a set of examples (i.e. experience). This reasoning from specific to general is named *inductive* reasoning. In contrast to deductive reasoning which proceeds from general to particular, inductive reasoning does not guarantee that the answer is correct. Therefore, the result of inductive reasoning is usually referred as hypothesis. Such a hypothesis needs some external motivation, such as statistical evidence. Both machine learning and data mining depend heavily on inductive reasoning that is obviously harder than deduction.

Many different techniques and approaches exist, but at present only ILP offers a framework for inductive reasoning. Hence, ILP is an obvious candidate as a paradigm for both machine learning and data mining [Blo98]. Moreover, ILP, which is more powerful than most other techniques such as attribute-value (or propositional) learning, seems to flawlessly deal with current database systems where relational technologies with multiple tables has long been the standard. From a KDD perspective, ILP can be seen as the development of techniques and systems for multi-relational data mining [DL01b] to emphasize the ability of dealing with multi-relational data and discover multi-relational patterns. This allows problems to be treated that cannot be handled easily with classical data mining methods which only deal with data that resides in a single relation (or table). ILP also reduces the need for manual pre-processing to integrate data from multiple relations into a single relation before data mining method can be applied (see next Section).

In this case, the problem is that integrating data from multiple relations through joins or aggregations typically causes loss of information.

Finally, besides the ability to deal with data stored in multiple relations directly, ILP methods are usually able to take into account background knowledge and support qualitative reasoning.

Despite these clear advantages of ILP approaches to data mining, there are still few examples of applications in which ILP is considered. This depends on the typically high computational complexity of ILP methods, which is a problem in the context of data mining, where efficiency is crucial. A solution to efficiency problem is found in *learning from interpretations* [De 96]. In this setting, each example e is a Prolog program encoding the specific properties of the example. It is also possible to specify the background knowledge B in the form of a Prolog program[2]. In this way, learning from interpretation that is able to combine efficiency of attribute-value learning with expressive power of classical ILP also opens up new possibilities for employing ILP in knowledge discovery.

2.3.1 The single table assumption: problems and solutions

Classical data mining methods share a restrictive data representation formalism, known as *single table assumption* [Wro01]: it is assumed that data to be mined are represented in a single table (or relation) of a relational database, such that each row (or tuple) represents an independent unit of the sample population and columns correspond to properties of units. In complex real-world applications this assumption turns out to be a great limitation.

As an illustrative example of some research issues due to the single table assumption let us consider the census data table reported in Table 2.1, where each row represents an enumeration district (ED), the smallest areal unit for which census data are published in United Kingdom (UK)[3]. EDs are spatial objects, since they have a geographical location. Having recognized this peculiarity, the data analyst may be interested in investigating the socio-economic phenomenon of deprivation (e.g. percentage of cars) in association with the geographical distribution of EDs. To achieve this goal, the analyst may decide to augment the data table in Table 2.1 with information on neighboring units. In particular, for each ED, the analyst proposes the following data specifications:

1. the number of schools in the neighboring EDs,
2. the number of banks in the neighboring EDs, and
3. the number of commercial activities in the neighboring EDs,

[2]The interpretation corresponding to each example e is then the minimal Herbrand model of $e \wedge B$.

[3]NSIs make a great effort to collect census data, but they are not the only organizations that analyze them: data analysis is often done by different institutes. By law, NSIs are prohibited from releasing individual responses to any other government agency or to any individual or business enterprise, so data are summarized for reasons of privacy before being distributed to external agencies and institutes. Therefore, data analysts are confronted with the problem of processing data which summarize characteristics of groups of individuals.

since he/she suspects that the low percentage of cars can also be related to the number of services available in the neighborhood.

TABLE 2.1: An example of census data table. Data are summarized per enumeration district (ED).

c1	**c24**	**c25**	**c26**	**c27**	**c28**	**c30**	**c32**	**c33**	**c34**	**c35**	**c36**
03BSFA01	44	69	23	6	5	7	0	0	7	15	109
03BSFA02	56	108	36	8	11	22	0	2	12	27	233
03BSFA03	74	98	27	5	9	18	1	0	13	33	127
...	...	...	...	...	...	...	...	...	...	...	...

c1: ED level code, e.g. '03BSFA01', where '03' denotes a country/region (Greater Manchester), 'BS' denotes a district (Stockport), 'FA' denotes a ward (Bredbury) and '01' is the enumeration district.
c24: Total females of employees (full time) aged 16 and over
c25: Total males of employees (full time) aged 16 and over
c26: Total females of employees (part time) aged 16 and over
c27: Total males of employees (part time) aged 16 and over
c28: Total females of self-employed — with employees aged 16 and over
c30: Total males of self-employed — with employees aged 16 and over
c32: Total females of on a government scheme aged 16 and over
c33: Total males of on a government scheme aged 16 and over
c34: Total females of unemployed aged 16 and over
c35: Total males of unemployed aged 16 and over
c36: Total car availability in all households (households with three or more cars counted as having three cars)

If the analyst decides to represent the above data only for one neighboring ED, the data table in Table 2.1 can be extended by simply adding three attributes (see Table 2.2). What if he/she wants to represent the three attributes for all spatially adjacent EDs, which are variable in number? Under the single-table assumption he/she can create one entry for each adjacent ED in the original data table. However, this solution presents two main disadvantages:

1. we have the usual problems connected with non-normalized tables, such as redundancy and anomalies in the insertion and removal of data,

2. we have one line per neighboring ED, which means that the analysis results will really concern neighboring EDs. In other words, the units of analysis have deceptively changed.

The former is a typical database issue, while the latter is more related to the data analysis procedure. To solve these problems and keep the single-table assumption, the data analyst may try to summarize the information on the neighboring EDs, say, by averaging the number of schools, banks and commercial activities (see Table 2.3). It is noteworthy that in this case there is no redundancy and standard data mining methods work well. However, there is an information loss that might lead to the understanding of the underlying phenomenon. For instance, an ED can be

TABLE 2.2: Three additional attributes of the nearest neighbors added to the single table.

c1	**c24**	**c25**	**...**	**c36**	**No. schools**	**No. banks**	**No. commercial activities**
03BSFA01	44	69	...	109	1	1	13
03BSFA02	56	108	...	233	0	0	23
03BSFA03	74	98	...	127	0	1	6
...	...	...	...	...	...	...	...

adjacent to another ED with many services, as well as to other EDs with no services at all, since they fall into the green belt of the city. By averaging the number of services per neighboring ED, the analyst may give a totally wrong indication on the deprivation distribution.

TABLE 2.3: Three additional attributes computed by averaging corresponding values of the nearest neighbors added to the single table

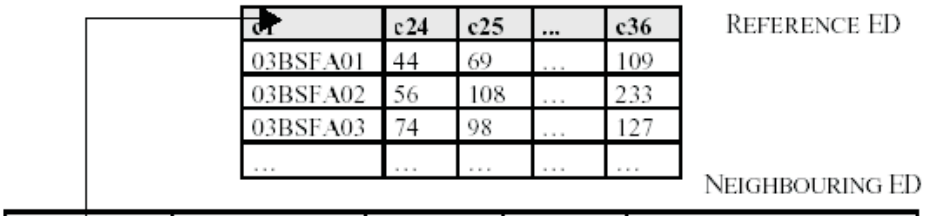
c1	**c24**	**c25**	**...**	**c36**	**Av. no. schools**	**Av. no. banks**	**Av. no. comm. activities**
03BSFA01	44	69	...	109	0.25	0.25	3.3
03BSFA02	56	108	...	233	0.33	0	0.36
03BSFA03	74	98	...	127	0	0.2	0.12

From a database perspective, the best representation of data would be that in Figure 2.2. In this database two relations are defined, one for the target EDs, that is, the EDs whose socio-economical factors are the subject of investigation, and one for the neighboring EDs, which are considered target relevant, because they are spatially adjacent to some target EDs. Obviously, mining this simple database requires for more powerful methods which go beyond the single table assumption. At this aim, the ILP research field has been exploiting relational analysis technologies for a number of years to now with significant results in dealing with relational data and analysis results by both representing and exploiting the relational structure of data in the learning process.

2.3.2 Relational learning in ILP

As stated before, ILP has been concerned with finding patterns expressed as logic programs. It initially focused on the automated synthesis of logic programs from examples and background knowledge. This task that is typically referred as a concept learning task (inducing binary classifiers), can be also formulated as the learning of logical (intensional) definitions of relations. More recent developments, however, have broadened the scope of ILP to consider all the main data mining tasks such as classification, regression, clustering or association analysis. Hence, ILP methods have been developed to learn multi-relational patterns in multi-relational data.

Traditional representation languages based on propositional logic (i.e. single



...	c24	c25	...	c36
03BSFA01	44	69	...	109
03BSFA02	56	108	...	233
03BSFA03	74	98	...	127
...	...	...	...	...

REFERENCE ED

NEIGHBOURING ED

Reference ED	Neighbouring ED	Number of schools	Number of banks	Number of commercial activities
03BSFA01	03BSFA16	0	1	2
03BSFA01	03BSFA11	1	0	3
03BSFA01	03BSFT22	0	0	1
03BSFA01	03BSFA07	0	0	4
03BSFA01	03BSFA12	0	0	2
03BSFA01	03BSFA10	0	0	1
03BSFA02	*03BSFW11*	0	1	1

FIGURE 2.2: A multi-relation representation of socioeconomic attributes of some reference EDs and of their neighboring ED. The attribute 'Reference ED' in the lower table is a foreign key of the upper table.

table assumption) appear completely inadequate to express these multi-relational patterns including relational association rules, relational decision tree and regression tree, among others. Therefore, ILP methods mainly employ languages based on logic programming that is a subset of first order logic also called relational logic.

A logic program consists of clauses that is first order rules where the conclusion part is termed the head and the condition part the body of the clause. The head and the body of a clause consist of atoms, where an atom is a predicate applied to some arguments. In particular, a program clause is in the form $H \leftarrow B_1 \wedge \ldots \wedge B_m$ such that H is an atom and $B_1 \wedge \ldots \wedge B_m$ are literals. A literal is either positive (atom) or negative (the negation of an atom). It is noteworthy that a program clause contains exactly one atom in the head. Logic programs are sets of program clauses, while the program clauses with the same predicate in the head forms a predicate definition.

Relational algebra, the formalism of relational databases, is also considered a subset of first-order logic [DL01a]. A relational database consists of a set of tables and a set of associations (i.e. constrains) between pairs of tables describing how tuples (or records) in one table relate to tuples in another table [Ull88]. Both tables and associations are known as relations. This suggests a relationship between database and logic programming terms that can be adequately modeled by resorting to the concept of deductive database.

A deductive database is a set of database clauses. A database clause is a typed program clause, i.e., a domain is associated with each argument. In this spirit, a database may be boiled down to a deductive relational database once the relations are expressed by means of predicates such that the arguments of a predicate correspond to the attribute of a relation. In deductive databases, relations can be expressed extensionally by a set of tuples (as in relational databases) or intentionally

as a set of database clauses (views). This means that ILP methods for relational learning may exploit the expressive power of first-order logic in databases to specify hierarchies or integrity constraints as well as domain specific knowledge expressed as sets of rules and support a qualitative reasoning.

Typically, ILP methods for relational learning have been upgraded from propositional logic (i.e. single table case) toward first order logic (i.e multi table case). For example distance-based algorithms for prediction and clustering have been upgraded from propositional to first-order logic by defining a distance measure between examples represented in first-order logic [KWH01]. The methodology for upgrading an existing propositional learner toward first order logic is explained into [VD01]. This methodology represents the most important lesson learned during the development of several ILP systems including TILDE [Blo98] [BD98], CLAUDIEN [DD97], ICL [DW95] and WARMR [De 97].

2.4 Multi-relational data mining

An important aspect of data mining methods and systems is that they have to scale well to large databases. Paying a lot of attention to efficiency is especially necessary in the case of databases that may contain very complex patterns. In such databases, experience has proved that classical single table mining methods scale well. The idea is to transform a relational representation of a learning problem into a propositional form [KLF01].

From a database perspective, this corresponds with *molding* a relational database into a single table format that traditional attribute-value algorithms can handle [KHS01]. However, the downside of the single table representation is that more complex patterns are simply not expressible in this format and, thus, cannot be discovered. One way to face this issue is to enlarge the expressiveness by generalizing from single table mining to multiple table mining. In such situations, ILP methods provide functionalities to navigate relational structure of data and generate potentially new form of evidence, not readily available in flattened single table representation.

Unfortunately the use of first-order logic as a representational language limits the application of these methods in many real world applications. Indeed, ILP approaches are mostly based on data stored as Prolog facts and little attention has been given to data stored in relational databases and how the knowledge of the data model can help to guide the search process.

Some pre-processing is required to transform data stored in relational databases into Prolog facts. This pre-processing may be expensive in terms of computation and storage and partially unnecessary since part of the hypothesis space may never be explored. Furthermore, in applications where data can frequently change, this pre-processing has to be frequently repeated.

The solution to efficiently mine multi-relational patterns directly from a large database can be found in implementing ILP methods as special instances of a multi-relational data mining framework where both attribute-value and structural infor-

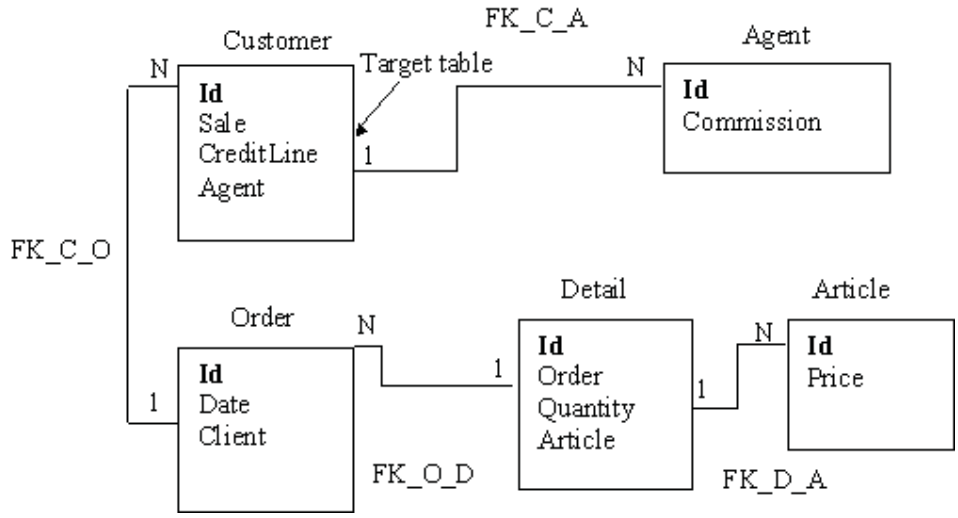


FIGURE 2.3: The relational data model of CustomerDB database.

mation embedded in the database are explicitly exploited. In this way, discovered patterns are relational since they concern structure of units of analysis composed by a single *target object* (primary data), zero, one or more *target relevant objects* (secondary data) as well as the relationships among them.

Both target objects and target relevant objects are naturally stored in a set S of multiple tables $T_1, \ldots, T_v$ of a relational database D. Hence, a relationship between objects can be modeled by means of a constraint describing an association between pairs of tables which states how tuples in one table relate to tuples in another table. The nature of a relationship between the tables T_i and T_j depends on the multiplicity of the association that determines whether several tuples in T_i relate to single or multiple tuples in T_j. A special case of association is a foreign key constraint $FK(T_i, T_j)$ that describes the relationship from a foreign key in T_i to the primary key in T_j.

In this way, a unit of analysis may consist of a *single* tuple t of some target table T in D joined with all tuples in S, which are related to t following a foreign key path in D.

Formally, a foreign key path is defined as follows:

Definition 2.1 *A foreign key path between a table T_h and a table T_k ($T_h \neq T_k$) in S is an ordered sequence of tables $\vartheta(T_h, T_k) = (T_{i_1}, T_{i_2}, \ldots, T_{i_s})$ such that:*

1. *$T_{i_1} = T_h$ and $T_{i_s} = T_k$,*
2. *$\forall j = 1, \ldots, s$, $T_{i_j} \in S$,*
3. *$j = 1, \ldots, s - 1$, there is a foreign key constraint from T_{i_j} to $T_{i_{j+1}}$, or vice-versa.*

♦

In Figure 2.3 some examples of *foreign key paths* are reported. In this case, S= {Customer, Agent, Order, Detail, Article} and the foreign keys are: FK_C_A, FK_C_O, FK_O_D, FK_D_A. If "Customer" is the target table then possible foreign key paths on Customer table are:

(*Customer*, *Agent*)

(*Customer*, *Order*)

(*Customer*, *Order*, *Detail*)

(*Customer*, *Order*, *Detail*, *Article*)

Foreign key paths are exploited in expressing the declarative bias to guide and constraint the search in the pattern space. The set of relational patterns derived from a relational database is bigger than the set of propositional patterns which can be derived from a single table. Therefore, some attention is given to reduce the search space and efficiently evaluate potentially interesting patterns. At this aim, constraints available in the conceptual data model of a relational database may be used to drastically prune the search space by considering only the structural information that is intended by the design of the database without wasting time on potentially large numbers of conceptually invalid patterns.

Considering the special attention reserved to database technologies, multi relational data mining differs from ILP in three other aspects. Firstly, it is typically restricted to the discovery of non recursive patterns. Secondly database primitives are typically employed to ensure efficiency. These primitives (e.g. summarization of both attribute values and structural information) are direct generalization of those adopted in many data mining architectures with extra facilities to cope with data in multiple tables [FL96]. They can be expressed in SQL and processed by either a conventional relational DBMS or a dedicated server that is optimized to efficiently compute primitives. Thirdly, the restriction to non-recursive patterns combined with data rapresented as tuples scattered in multiple tables of a relational database rather than first-order logic facts suggests that SQL besides first-order logic may be used to express discovered relational patterns. In [KSV99], the authors present a graph based language to graphically describe relational patterns containing decisions associated with each node of a multi-relational decision tree and also show their translation into SQL statement. In this way, the classification model can be completely expressed by the set of SQL queries associated with the leaves. Each SQL query corresponding with a node of the tree has also a class label associated with it, that is the classification based on the majority of training objects matched by the query itself.

2.5 Conclusions

In this chapter, we have introduced the basics of the KDD process and discussed the motivations of an inductive database framework where KDD can be modeled as an interactive process in which users query patters as well as data by means of an inductive query language. This leads to a tight integration between KDD process and databases.

Data mining is the central activity in this process concerning with mining patterns in data. Most data mining methods look for patterns in a single table (attribute-value representation) of a relational database. Since data typically resides in multiple tables of real world databases, much effort is devoted in transforming such data into a single table (e.g. through joins or aggregations) with a consequently loss of information or change of data structure.

The problem of directly dealing with data stored in multiple tables without any preliminary transformation has been tackled in ILP field. Indeed, ILP systems can be directly applied to multi-relational data to mine patterns that involve multiple relations. In addition, ILP methods are generally able to take into account valid background knowledge in form of a logic program. However, the use of first-order logic as a representational language may restricts the actual employment of these methods in real world applications since only little attention is given to data stored in relational databases and how the knowledge of the data model can help to guide the search process.

The idea is to combine the achievements in KDD field on the integration of data mining with database systems, with some results reported in ILP field on how to correctly upgrade attribute-value data mining algorithms to multi-relational representations. This leads to implement ILP methods as special instances of a multi-relational data mining framework where both attribute-value and structural information embedded in the coupled database are explicitly exploited. In this respect, differences between ILP and multi-relational data mining have been discussed.

Chapter 3

Induction of Regression Models

During the past fifty years regression analysis has been one of the most widely used methodologies for analyzing relationships among variables. Indeed, several regression methods have been developed in statistics, machine learning as well as data mining. At present, they have become a basic support for solving real world problems due to their flexibility, usefulness, applicability, theoretical and technical sauciness. Regression methods are typically based on the *inductive inference* that consists of hypothesizing H, given C and B in the following entailment:

$$H \cup B \models C,$$

where H is named the set of premises, while B is the background knowledge and C is a set of consequences. A special application of inductive inference is properly in the context of *supervised learning*, where observations (training cases) of some phenomenon are a-priori labeled by a domain expert. In regression tasks, this label is a number. This means that the regression training cases may be seen as instances of an unknown continuous function. Hence, the mining goal in regression consists of using the inductive inference to obtain a general description (or model) of this function from a set of labeled observations together with some background knowledge.

The representational language used to describe the observations, the background knowledge as well as the discovered model is a central choice that influences the pattern (regression model) space to be generated and evaluated in induction phase. The classical approach adopts a language based on propositional logic where each case is described by a fixed set of variables. An alternative is the use of a powerful representation language (e.g. subset of first-order logic) that encompasses most of the real world regression problems since it allows to represent both relational data and relational patterns.

The task of inducing regression models based on labeled observations is the main topic addressed by this dissertation. In the remaining of this chapter, we present an overview of existing approach to regression problems within machine learning as well as other research fields.

3.1 The regression problem

In the classical regression settings, training data D are described by a fixed set of m predictor (independent) variables X_i, either continuous or discrete, and a target (dependent) variable Y, which normally takes values in the set of real numbers $\Re$. Each variable has only a single, primitive value.

In a matrix notation, D may be represented as the matrix $\boldsymbol{D}$ whose i-th row is the input vector $X_{1_i}, \ldots, X_{m_i}, Y_i$, which describes the i-th training case. If there are n training cases, $\boldsymbol{D}$ is a matrix with dimension $n \times (m+1)$.

Regression analysis is mainly concerned with estimating the statistical expectation $E(Y|X_1, \ldots, X_m)$ of Y according to the values of $X_1, \ldots, X_m$. This estimation is actually based on the training set D of labeled data (i.e. observations of the phenomenon to be modeled), since the true underlying regression function is typically unknown.

The assumption is that the relationship between predictors and target variable assumes some fixed form (e.g. linear or kernel functions) and it can be described by:

$$Y = f(\boldsymbol{X}) + \varepsilon, \tag{3.1}$$

where $f(\boldsymbol{X})$ is a regression model on the vector of variables $\boldsymbol{X} = X_1, \ldots, X_m$, while ε are observation errors. Therefore, a regression method aims at discovering the best f according to a selected preference criterion. This search bias typically includes the estimation of the predictive accuracy of f.

As the target variable of regression problems is continuous, the predictive accuracy of f can be quantified by revolving around the difference between the values predicted by f and the true values for a set of regression cases *labeled a-priori*.

The *mean absolute deviation* (*MAD*) averages the absolute deviations of predictions performed by the regression model f on n labeled regression cases $(\boldsymbol{x}_i, y)$, such that:

$$MAD(f) = \frac{1}{n} \sum_{i=1\ldots n} |y_i - f(\boldsymbol{x}_i)|. \tag{3.2}$$

This measure leads to the least absolute deviation regression that determines the regression model f which minimizes the average absolute deviation on D.

However, for reasons due to ease of computations, the measure of accuracy really used in regression is the *mean square error* (*MSE*), that is:

$$MSE(f) = \frac{1}{n} \sum_{i=1\ldots n} (y_i - f(\boldsymbol{x}_i))^2. \tag{3.3}$$

The methodology revolving about this measure is the least square regression. As discussed in [BFOS84], the mean square error $MSE(f)$ can be also defined as:

$$MSE(f) = E(Y - f(\boldsymbol{X}))^2. \tag{3.4}$$

This suggests that the regression model f^* which minimizes $MSE(f)$ is:

$$f^*(\boldsymbol{x}) = E(Y|\boldsymbol{X} = \boldsymbol{x}), \tag{3.5}$$

which means that $f^*(x)$ is the conditional expectation of the response value, given that the measurement vector is $\boldsymbol{x}$. This depends on the elementary Lemma 3.1.

Lemma 3.1 *The constant a which minimizes:*

$$E(Y-a)^2$$

is $E(Y)$. *(A complete proof is presented in [BFOS84], pp. 266-269).*

♦

In [BB03], the authors discuss the use of regression error curve (REC) area as a valid measure of the expected performance of regression model f. The REC curve plots the error tolerance on x-axis versus the percentage of points predicted within the tolerance on y-axis. The resulting curve estimates the cumulative distribution function of the error. REC curve analysis represents an extension to regression tasks of Receiving Operating Characteristic (ROC) analysis [FF03] that defines an evaluation measure to take into account the confidence in classification tasks. In regression case, the area over REC curve (AOC) is proved to be a biased estimate of expected error.

Mean absolute deviation as well as mean square error and area over REC curve are statistical estimators of the error ϵ of a regression model f. However, when these estimates are obtained by the same data used to induce the regression model, they are known to be unreliable (or biased) [Tor99].

Obtaining a reliable estimation of ϵ is important, since it enables to estimate the future accuracy of f on new (un-labeled) data. It guides the choice of the best regression model among several alternative models generated by a regression method. Finally, reliable error estimate is important when combining models [BT90] or their predictions [Wol92] [Bre96].

There are several approaches to achieve reliable error estimates. For instance, *re-sampling* methods such as *holdout* [Hig62] [Rip96], *cross validation* [MW63] [Sto74] or *bootstrap* [Efr79] [ET93], which proceed by obtaining the estimates with data not already processed to mine the regression model f.

Conversely, *bayesian estimation* methods obtain the estimates by combining the prior expectation of the parameter being estimated (e.g. ϵ) and the observed value. An example is the m-probabilities estimates [Ces90], that is, a generalization of the Laplace's law of succession [Goo65].

Finally, a different approach consists of studying the sampling distribution properties of the estimates obtained on several sample data of size n. It is possible to look at these computed estimates as values of another variable named *sampling distribution*. Analyzing the sampling distribution of an estimate of ϵ (e.g. *MAD*, *MSE* or *AOC*) allows to draw important conclusions regarding the confidence on any particular estimate.

3.2 Statistical regression methods

Regression has been extensively investigated in statistical data analysis. This section presents the major regression paradigms developed in statistic field, distinguish-

ing among global parametric approaches, non parametric approaches and additive approaches. The main advantages and drawbacks of these different approaches are adequately pointed out.

3.2.1 Global parametric approaches

Global parametric methods construct a single functional model that is easily interpreted and fits the entire set D of n training cases. They have fast computational solutions, but impose a strong assumption, that is, an a-priori predefined functional form for the regression model f to be discovered. Obviously, this leads to lower accuracy when this hypothesized functional form does not correspond to the actual structure of the g unknown function really underlying the phenomenon to be modeled.

In *linear regression*, this assumption is that:

$$f(\boldsymbol{x}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_m x_m, \tag{3.6}$$

where $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_m)$ is estimated according to the *least square error criterion* as the parameter vector value $\widehat{\boldsymbol{\beta}} = (\widehat{\beta_0}, \ldots, \widehat{\beta_m})$, which minimizes the sum of square errors (SSE):

$$SSE(f, \widehat{\boldsymbol{\beta}}) = \min_{\boldsymbol{\beta}} \sum_{(\boldsymbol{x}_j, y_j) \in D} (y_j - (\beta_0 + \beta_1 x_{1_j} + \ldots + \beta_m x_{m_j}))^2. \tag{3.7}$$

This minimization problem has an elegant and efficient solution in matrix algebra [Wei85]. The linear regression model can be expressed as a matrix equation of the form:

$$\boldsymbol{D_Y} = \boldsymbol{D_X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{3.8}$$

where:

- $\boldsymbol{D_Y}$ is the $(n \times 1)$ vector of observed Y,
- $\boldsymbol{D_X} = (\mathbf{1}, \boldsymbol{X_1}, \ldots, \boldsymbol{X_m})$ is the $(n \times (m+1))$ matrix such that $\mathbf{1}$ is a column with only 1s, while each $\boldsymbol{X_i}$ is a column with the observed X_i,
- $\boldsymbol{\beta}$ is the $((m+1) \times 1)$ vector of the β_i parameters to be estimated,
- $\boldsymbol{\epsilon}$ is the $(n \times 1)$ vector of errors.

The sum of square error is then:

$$\begin{aligned}
\boldsymbol{\epsilon}^T\boldsymbol{\epsilon} &= (\boldsymbol{D_Y} - \boldsymbol{D_X}\boldsymbol{\beta})^T(\boldsymbol{D_Y} - \boldsymbol{D_X}\boldsymbol{\beta}) && (3.9)\\
&= \boldsymbol{D_Y}^T\boldsymbol{D_Y} - \boldsymbol{\beta}^T\boldsymbol{D_X}^T\boldsymbol{D_Y} - \boldsymbol{D_Y}^T\boldsymbol{D_X}\boldsymbol{\beta} + \boldsymbol{\beta}^T\boldsymbol{D_X}^T\boldsymbol{D_X}\boldsymbol{\beta} && (3.10)\\
&= \boldsymbol{D_Y}^T\boldsymbol{D_Y} - 2\boldsymbol{\beta}^T\boldsymbol{D_X}^T\boldsymbol{D_X} + \boldsymbol{\beta}^T\boldsymbol{D_X}^T\boldsymbol{D_X}\boldsymbol{\beta}, && (3.11)
\end{aligned}$$

where, $\boldsymbol{\beta}^T\boldsymbol{D_X}^T\boldsymbol{D_Y}$ is a 1×1 matrix (i.e a scalar value) and the corresponding transpose matrix $(\boldsymbol{\beta}^T\boldsymbol{D_X}^T\boldsymbol{D_Y})^T = \boldsymbol{D_Y}^T\boldsymbol{D_X}\boldsymbol{\beta}$ must have the same value.

The least square estimate of $\boldsymbol{\beta}$ is exactly the vector value $\widehat{\boldsymbol{\beta}}$ that minimizes $\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}$. It is obtained by differentiating Equation 3.11 with respect to $\boldsymbol{\beta}$[1] and setting the resultant matrix equation to zero, at the same time replacing $\boldsymbol{\beta}$ with $\widehat{\boldsymbol{\beta}}$. This leads to the normal equation:

$$-2\boldsymbol{D_X}^T\boldsymbol{D_Y} + 2\boldsymbol{D_X}^T\boldsymbol{D_X}\widehat{\boldsymbol{\beta}} = \mathbf{0}, \tag{3.12}$$

which consists of m equations in m unknowns. If the m equations are independent, $\boldsymbol{D_X}^T\boldsymbol{D_X}$ is not singular, and its inverse matrix exists. In this case the solution of the normal equation can be written as:

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{D_X}^T\boldsymbol{D_X})^{-1}\boldsymbol{D_X}^T\boldsymbol{D_Y}. \tag{3.13}$$

Example 3.1 *Let us consider a training set D on $X \times Y$ such that $m = 1$, the straight line $Y = \widehat{\beta_0} + \widehat{\beta_1}X$ fitted by least square is the one that makes the sum of all discrepancies between each point $(\boldsymbol{x}_j, y_j) \in D$ and the line as small as possible (see Figure 3.1). The estimate $\widehat{\beta_0}$ (intercept) and $\widehat{\beta_1}$ (slope) are computed according to Equation 3.13 as follows*[2]*:*

$$\widehat{\beta}_1 = \frac{\sum_{i=1\ldots n}(x_i - \overline{X})(y_i - \overline{Y})}{\sum_{i=1\ldots n}(x_i - \overline{X})^2} \tag{3.14}$$

$$\widehat{\beta}_0 = \overline{Y} - \widehat{\beta}_1\overline{X} \tag{3.15}$$

where $\overline{X} = \frac{1}{n}\sum_{1\ldots n} x_i$ and $\overline{Y} = \frac{1}{n}\sum_{1\ldots n} y_i$.

♦

There are many variants of this general approach that differ in the way they actually induce regression model from data. For instance, a multiple linear regression model may be *incrementally* built via a *forward stepwise* procedure, by sequencing straight line regression and removing the linear effect of the introduced variables each time a new independent variable is added to the model [DS82]. The forward stepwise procedure starts with an empty model and then iteratively alters the current model by adding a predictor variable until a stopping criterion (e.g. maximum number of steps) is satisfied.

Example 3.2 *Let us consider the problem of building a multiple regression model $Y = \widehat{\beta_0} + \widehat{\beta_1}X_1 + \widehat{\beta_2}X_2$ with two independent variables (X_1, X_2) through a sequence of straight-line regressions. The forward stepwise procedure starts regressing Y on X_1 according to least square criterion, so that the model: $Y = \widehat{\beta_{0_1}} + \widehat{\beta_{1_1}}X_1$ is built. The fitted equation does not predict Y exactly. By adding the new variable X_2, the prediction might improve. Instead of starting from scratch and building a model with both X_1 and X_2, the forward procedure builds a linear model for X_2 if X_1 is given: $X_2 = \widehat{\beta_{0_2}} + \widehat{\beta_{1_2}}X_1$, then computes the residuals on X_2 and Y: $X_2' = X_2 - (\widehat{\beta_{0_2}} + \widehat{\beta_{1_2}}X_1)$ and $Y' = Y - (\widehat{\beta_{0_1}} + \widehat{\beta_{1_1}}X_1)$,*

[1] Differentiating $\epsilon^T\epsilon$ with respect to the vector quantity β is equivalent to differentiating $\epsilon^T\epsilon$ separately with respect to each each element of β in order, writing down the resulting derivatives one below the other, and rearranging the whole in the matrix form.

[2] Details are reported in [DS82].

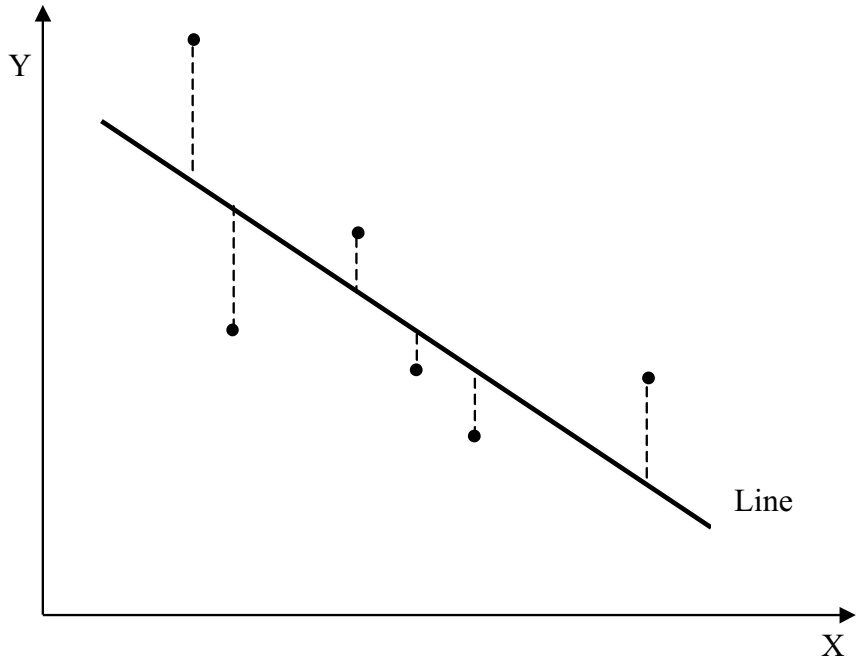


FIGURE 3.1: The vertical deviation whose sum of square is minimized according to the least square criterion.

and finally regresses Y' on X_2' alone:

$Y' = \widehat{\beta_{0_3}} + \widehat{\beta_{1_3}} X_2'$.

By substituting the equations of X_2' and Y' in the last equation we have:

$Y - (\widehat{\beta_{0_1}} + \widehat{\beta_{1_1}} X_1) = \widehat{\beta_{0_3}} + \widehat{\beta_{1_3}}(X_2 - (\widehat{\beta_{0_2}} + \widehat{\beta_{1_2}} X_1))$.

that is:

$Y = (\widehat{\beta_{0_3}} + \widehat{\beta_{1_1}} - \widehat{\beta_{0_2}\beta_{1_3}}) + (\widehat{\beta_{1_1}} - \widehat{\beta_{1_2}}\widehat{\beta_{1_3}})X_1 + \widehat{\beta_{1_3}} X_2$.

It can be proven ([DS82], pp. 196-201) that this last model coincides with the first model directly built solving Equation 3.13, that is, $\widehat{\beta_0} = \widehat{\beta_{0_3}} + \widehat{\beta_{0_1}} - \widehat{\beta_{0_2}}\widehat{\beta_{1_3}}$, $\widehat{\beta_1} = \widehat{\beta_{1_1}} - \widehat{\beta_{1_2}}\widehat{\beta_{1_3}}$ and $\widehat{\beta_2} = \widehat{\beta_{1_3}}$.

♦

The counterpart of stepwise forward procedure is the *backward stepwise* procedure to identify and filter out all variables do not significantly improve accuracy of induced regression model. This means that after the whole model containing all predictor variables $X_1, \ldots, X_m$ is built, a *removal statistic* (e.g. partial F-test value) is computed for each X_i treated as though it was the last variable to enter the regression equation. The removal statistic is associated with a test of:

$$H_0 : \beta_i = 0 \text{ versus } H_1 : \beta_i \neq 0$$

for any regression coefficient β_i. In the case the test is not rejected, the variable X_i is removed and the regression model f is recomputed on the remaining variables [DS82].

3.2.2 Non parametric approaches

Non-parametric regression belongs to a data analytic methodology known as *local modeling* [Fan95]. It often adopts different types of simple and local models in different portions of training data to build up an overall model of the data. The basic idea behind local regression consists of relaxing assumptions on the form of an unknown function of interest and letting data to choose a suitable function that fits well the given dataset. This makes non-parametric regression a good competitor to non-linear regression for modeling situations in which a theoretical model is not known or is difficult to fit.

Local regression is strongly related to the work on instance-based learning (e.g. [AKA91]) and includes locally parametric methods which obtain the prediction for a regression case $\boldsymbol{x}$ by fitting a regression function in the *neighborhood* of $\boldsymbol{x}$. Given a test case $\boldsymbol{x}$, the neighborhood $N(\boldsymbol{x}, D)$ is the set of training cases in D which are *most similar* to $\boldsymbol{x}$. This neighborhood is then used to build the local model $f_{N(\boldsymbol{x},D)}$ and perform the prediction $f_{N(\boldsymbol{x},D)}(\boldsymbol{x})$.

The instance-based inductive methodology is named *lazy learning* [Aha97] since it does not perform any kind of generalization of the given training data and delays learning till prediction time. This highlights the main drawback of local modeling that is no comprehensible model of training data is made available to data analysts and domain experts.

Cleveland and Loader have provided in [CL96] an historical survey of the work done on local regression traces since the 19^{th} century. The modern work on local modeling starts in 1950 with *kernel methods* which combine achievement of probability density estimation [Ros56][Par62] with requirements of regression setting [Nad64][Wat64]. Kernel methods have been proposed to fit a polynomial of degree zero (a constant) into a neighborhood. *Local polynomial regression* is a generalization of this early work on kernel regression that aims at fitting a polynomial of degree p around a query point (or test regression case) $\boldsymbol{x}_q$ using the training data in its neighborhood. This includes various settings such as kernel regression ($p = 0$), local linear regression ($p = 1$), etc. A further generalization consists of using a polynomial mixing where p takes non integer values [CL96].

Spline smoothing provides another powerful tool for estimating non parametric functions [Eub88][GS94] [Wah90][HT90]. It is essentially based on the assumption that the random errors are independent. Observations are often correlated in applications; for example, time series data, spatial data and clustered data. It is well known that correlation greatly affects the selection of smoothing parameters, which are critical to the performance of smoothing spline estimates. Popular methods for selecting smoothing parameters are generalized maximum likelihood (GML), generalized cross-validation (GCV) or unbiased risk (UBR) [Wah90].

Most of studies on local modeling in regression are carried out for a unique independent variable, but applications to the multivariate case have been proposed for several domains [AMS97] [MSD97]. Nevertheless, the application of local modeling to high input space dimensions has some limitations due to the *curse of dimensionality* that is training cases described by an high number of variables are so

sparse that the notion of local neighborhood cannot be seen as local [Har90][HT90]. Indeed, the sparseness of data in this setting inflates the variance of the estimates.

3.2.3 Additive approaches

Parametric methods such as linear regression are known to be attractively simple, however they often fail in real world applications since real life effects are generally not linear. Similarly, non-parametric methods have limitations due to both the curse of dimensionality and the low interpretability of kernel and smoothing spline estimates.

To overcome these difficulties, Stone [Sto85] has proposed *additive models* as a flexible statistical means to estimate an additive approximation of multiple regression function. The benefits of an additive approximation are at least twofold. Firstly, since each of the single additive terms is estimated using an univariate smoother, the curse of dimensionality is avoided at the cost of not being able to approximate universally. Secondly, estimates of terms explain how the dependent variable changes with the corresponding independent variables.

Hastie and Tibshirani [HT90] have proposed *generalized additive models*. These models assume that the mean of the target variable depends on an additive predictor through a nonlinear link function. Therefore, generalized additive models permit the response probability distribution to be any member of the exponential family of distributions.

The main drawback of this methodology is its computational complexity [Tor99]. One has a potentially large set of candidate basis functions to choose from, and afterwards one has to tune the parameters of each of these basis functions. Still, there are some simplifications that allow fast algorithms to be used. An example of an additive model is a regression tree [BFOS84], which can be efficiently obtained with a recursive partitioning algorithm.

Another example of an additive model is projection pursuit regression [FS81]. Projection pursuit provides a model of the form:

$$f(\boldsymbol{x}) = \sum_{i=1\ldots m} f_i(\sum_{j=1\ldots a} \beta_{j,i} x_i) \tag{3.16}$$

Adaptive regression splines are other additive models, which are generalizations of regression trees. They are implemented in the system MARS [Fri91] that builds a regression model of the form:

$$f(\mathbf{x}) = \beta_0 + \sum f_o(x_o) + \sum f_{o,p}(x_o, x_p) + \sum f_{o,p,q}(x_o, x_p, x_q) + \ldots, \tag{3.17}$$

where the first sum is over all basis functions involving only a single variable, the second sum is over all basis functions involving two variables, and so on.

Finally, many widely used statistical models belong to this general class, including additive models for Gaussian data, nonparametric logistic models for binary data, and nonparametric log-linear models for Poisson data.

3.3 *k*-Nearest Neighbor regression methods

k-Nearest Neighbor (k-NN) [Das91] is an instance-based method, which assumes all cases (or observations) correspond to points in m-dimensional space $\Re^m$. In regression tasks, k-NN estimates the continuous outcome y for a query point $\boldsymbol{x}$ on the basis of the *k-sized neighborhood* ($N_k(\boldsymbol{x}, D)$) that is the k examples over the training set D, which are closest in *distance*[3] to $\boldsymbol{x}$.

Assuming that all neighbors have equal influence on predicting the outcome y irrespective of their relative distance from the query point $\boldsymbol{x}$, this prediction is typically obtained by averaging the outcomes y_i of the k nearest neighbors of $\boldsymbol{x}$ belonging to $N_k(\boldsymbol{x}, D)$, that is:

$$y = \sum_{i=1\ldots k} \frac{1}{k} y_i. \tag{3.18}$$

An alternative approach is to give more importance to the cases which are closer to the query point [She68]. This is achieved using the so-called distance weighting as follows:

$$y = \frac{\sum\limits_{i=1\ldots k} w_i y_i}{\sum\limits_{i=1\ldots k} w_i}, \tag{3.19}$$

where $w_i \equiv \frac{1}{d(\boldsymbol{x},\boldsymbol{x}_i)^2}$, such that $d(\boldsymbol{x}, \boldsymbol{x}_i)$ is the distance between the query point $\boldsymbol{x}$ and a neighbor training example $\boldsymbol{x}_i \in N_k(\boldsymbol{x}, D)$.

Locally (weighted) regression [Mit97] is a generalization of k-NN approach, which builds an explicit approximation of f for the neighborhood surrounding $\boldsymbol{x}$ by fitting a linear (or quadratic, kernel ...) function to the k nearest (weighted) neighbors. Kibler and his colleagues [KAA89] have proposed the use of a kernel model within the k-sized neighborhood determined for $\boldsymbol{x}$ query point. Each neighbor enters in the prediction calculation with a weight proportional to its distance to $\boldsymbol{x}$. The work of Connel and Utgoff [CU87] have suggested a similar strategy with the difference that all training instances contribute to the prediction.

Example 3.3 *Let us consider the case of locally (weighted) regression in which the regression function f is approximated close to $\boldsymbol{x}$ as $f_{\boldsymbol{x}}$ using a linear regression function. Least square procedure is modified in order to derive a local approximation rather than a global one. This leads to redefine error estimation to be minimized. Three possible criteria are discussed in [Mit97], that is:*

1. *minimize the sum of square errors over just the k nearest neighbors in $N_k(\boldsymbol{x}, D)$:*

$$SSE_1(f_{\boldsymbol{x}}) = \sum_{x_i \in N_k(\boldsymbol{x},D)} (y_i - f_{\boldsymbol{x}}(\boldsymbol{x}_i))^2, \tag{3.20}$$

2. *minimize the sum of square errors over the entire set D of training cases weighting the error of each training example by some decreasing function K of its distance from $\boldsymbol{x}$:*

$$SSE_2(f_{\boldsymbol{x}}) = \sum_{x_i \in D} (y_i - f_{\boldsymbol{x}}(\boldsymbol{x}_i))^2 K(d(\boldsymbol{x}, \boldsymbol{x}_i)), \tag{3.21}$$

[3] One of the most popular choices to measure this distance is known as Euclidean [Mit97].

3. combine Equations 3.20 and 3.21:

$$SSE_2(f_{\boldsymbol{x}}) = \sum_{x_i \in N_k(\boldsymbol{x}, D)} (y_i - f_{\boldsymbol{x}}(\boldsymbol{x}_i))^2 K(d(\boldsymbol{x}, \boldsymbol{x}_i)), \tag{3.22}$$

♦

It is noteworthy that regression by k-NN is conceptually simple and easy to implement, but it appears slow at prediction time and easily fooled by irrelevant variables. Moreover, it generally suffers from opacity: the model does not reveal anything about the function that it represents. Finally, the accuracy of k-NN model is strongly affected by the choice of k. Hence, k can be regarded as one of the most important factors of the model since it strongly influences the quality of predictions. In the case all training examples contribute to predict the outcome value for the query point, k-NN works as a *global* method, while in the case only the k ($k < n$) nearest neighbor examples are considered, it works as a *local* method. In local k-NN only data local to the area around the test case contribute to obtain the prediction.

An appropriate way to look at the number of nearest neighbors k is to deal with it as with a smoothing parameter, where, for any given problem, a small value of k will lead to a large variance in predictions. Alternatively, setting k to a large value may lead to a large model bias. Thus, k should be set to a value large enough to minimize the error estimation and small enough (with respect to the number of cases in the training set) so that the k nearest points are close enough to the query point. Thus, like any smoothing parameter, there is an optimal value for k that achieves the right trade off between the bias and the variance of the model.

Some k-NN can provide an estimate of k using cross-validation [GW02][ADED04]. Alternative solutions for an adaptive determination of local k are discussed in [WD94].

3.4 Artificial neural network regression methods

Artificial neural networks (ANNs) provide a robust and flexible approach to induce approximation of real-valued as well as discrete-valued and vector-valued functions from examples.

The study of ANNs has been inspired in part by the observation that biological learning systems are built of very complex webs of interconnected neurons. In rough analogy, ANNs are built out of a densely interconnected set of simple units, where each unit (i.e. perceptron) takes a number of real-valued inputs (possibly the outputs of other units) and produces a single real-valued output (possibly input to many other units).

McCulloch and Pitts [MP43] have proposed the first model of perceptron. Since then many new more complex models have been proposed. Ronsenblatt [Ros58] has generalized the McCulloch-Pitts neural networks. This work has been then extended by Minsky and Papert [MP69].

A perceptron o (see Figure 3.2) is a computing unit that takes a vector of real-valued inputs with associated weights, and computes a linear combination of these

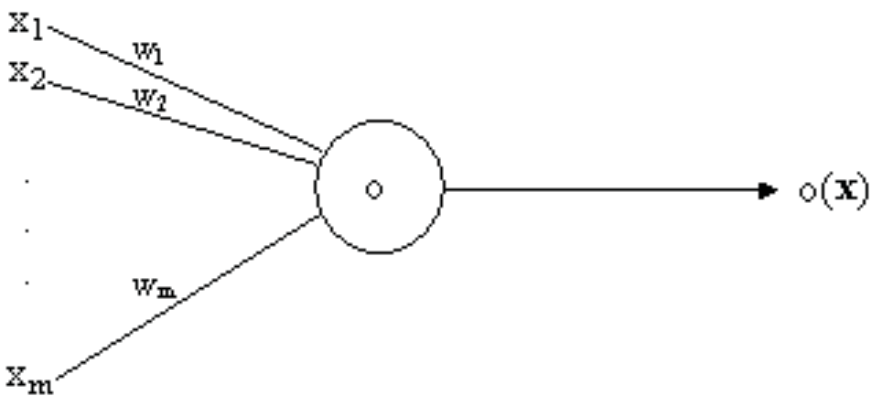


FIGURE 3.2: A perceptron unit.

inputs, then outputs 1 if the result is greater than a threshold (or bias) θ and -1 otherwise. More precisely, given the input vector $\boldsymbol{x} = x_1, \ldots, x_m$, the output $o(\boldsymbol{x})$ computed by the perceptron is:

$$o(\boldsymbol{x}) = \begin{cases} +1 \text{ if } w_o + w_1 x_1 + \ldots + w_m x_m > \theta \\ -1 \text{ otherwise.} \end{cases} \tag{3.23}$$

where each w_i is a real-valued constant, or weight, that determines the contribution of input x_i to the perceptron output $o(\boldsymbol{x})$. To simplify notation, an additional constant input $x_0 = 1$ is considered in order to write the above inequality as:

$$\sum_{i=0,\ldots,m} w_i x_i > \theta.$$

Learning with these units consists of finding the weights associated with each input variable. One way to learn an acceptable weight vector is to begin with random weights, then iteratively applying the perceptron to each training case, modifying the weights when predictions are wrong. Weights are modified at each step according to the *perceptron training rule*, that is:

$$w_i = w_i + \Delta w_i, \tag{3.24}$$

where $\Delta w_i = \eta(y - o(\boldsymbol{x}))x_i$. Here y is the actual target output for the current training example $\boldsymbol{x}$, while $o(\boldsymbol{x})$ is the output generated by the perceptron o, and η is a positive constant named *learning rate*. The role of learning rate is to moderate the degree to which weights are changed at each step. It is usually set to small value (e.g. $\eta = 0.1$) and it sometimes decays as the number of weight-tuning iterations increases. This learning procedure can be proved to coverage within a finite number of steps whenever the training cases are linearly separable and η value is sufficiently small [MP69]. If data are not linearly separable, convergence is not assured. Other learning algorithms have been proposed in literature such as instance linear programming techniques [Man91] or the Karmarkar's algorithm [Kar84].

In order to relax the requirement of linear separability, the *delta rule* may be used since it allows the learning to converge toward a best-fit approximation to the target function [Mit97].

The key idea behind the delta rule is the use of *gradient descent* to search the hypotheses space of possible weight vectors to find weights that best fit the training examples. This rule provides the basis of *backpropagation algorithm* [Wer75] [Wer96], which can learn network with many interconnected units. It is also interesting since gradient descent serves as basis for learning algorithms that must search through hypotheses space containing many different types of continuously parameterized hypotheses.

The delta rule is generally applied to an unthresholded unit, that is, a *linear unit* corresponding with the first stage of a perceptron without the threshold, for which the output is given by:

$$o(\boldsymbol{x}) = \sum_{i=0,\ldots,m} w_i x_i \tag{3.25}$$

The measure generally adopted to estimate the error of the weight vector $\boldsymbol{w} = (w_0, w_1 \ldots, w_m)$ is:

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{(\boldsymbol{x}_i, y_i) \in D} (y_i - o(\boldsymbol{x}))^2, \tag{3.26}$$

that is simply half the sum of square errors over training cases in D. The vector obtained by computing the derivative of E with respect each weight w_i is named gradient of E, written $(\boldsymbol{\nabla E(w)})$, that is the vector:

$$\boldsymbol{\nabla E(w)} = [\frac{\delta E}{\delta w_0}, \frac{\delta E}{\delta w_1}, \ldots, \frac{\delta E}{\delta w_m}]. \tag{3.27}$$

Consequently, the gradient can be interpreted as a vector in weight space, which specifies the direction that produces the steepest increase in E, while the negative of this vector gives the direction of steepest decrease. Therefore, by setting:

$$\Delta w_i = -\eta \frac{\delta E}{\delta w_i}, \tag{3.28}$$

the steepest descent is achieved by altering each weight w_i in proportion to $\frac{\delta E}{\delta w_i}$.

A practical algorithm for iteratively updating weights according to Equation 3.28 needs an efficient way of calculating the gradient at each step, that is:

$$\Delta w_i = \eta \sum_{(\boldsymbol{x}_j, y_j) \in D} (y_j - o(\boldsymbol{x}_j)) x_{i_j}. \tag{3.29}$$

Details about computation of 3.29 are reported in [Mit97] (chapter 4, pp. 91-92).

Generally, a single perceptron can only express linear decision surfaces. In contrast, *multi-layer networks* are capable of expressing a rich variety of non linear decision surfaces. They consist of an input layer related to the input variables, one or more hidden layers and an output layer.

Multiple layers of cascaded linear units still produce only linear functions. The alternative is to cascade perceptron units, but their discontinuous thresholds make them undifferentiable and hence unsuitable for gradient descent.

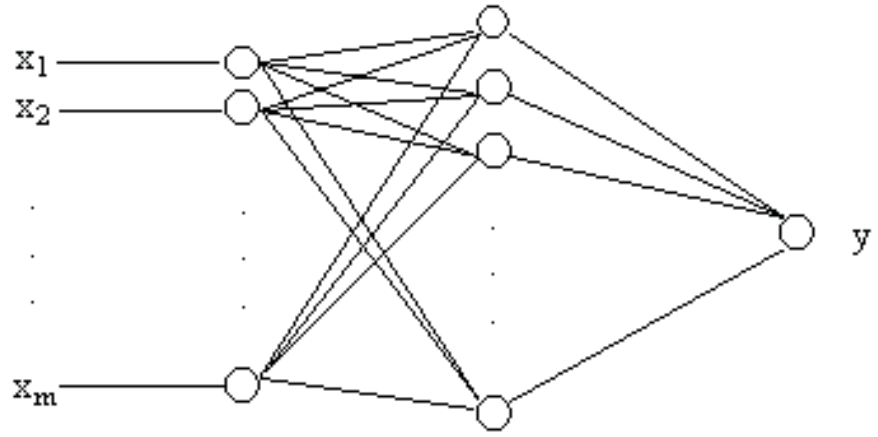


FIGURE 3.3: A multi-layer ANN architecture for regression.

A solution is the *sigmoid unit* that is a unit very much like a perceptron, but based on a smoothed, differentiable threshold function. The threshold output of a sigmoid unit is a continuous function of its input vector $\boldsymbol{x}$:

$$o(\boldsymbol{x}) = \frac{1}{1 + e^{-\sum\limits_{i=0\ldots m} w_i x_i}}. \tag{3.30}$$

The common set-up for applying ANNs in regression consists of using a 3-layered network with one hidden layer (see Figure 3.3). Both the input units and the hidden layer units are sigmoids, while the output unit is an unthresholded unit.

Neural networks are employed in several successfully applications [RWL94]. For instance, Wu and Zhu discuss a neural network regression model for relative dose computation [WZ00] that exhibits a good generalization and interpolation ability. Similarly, Subramanian and his colleagues use neural networks to optimize the formulation parameters of cytarabine liposomes [SYM04] obtaining a more accurate prediction when compared with the multiple regression analysis method. However, neural networks appear clearly inadequate in problems where interpretability is a key factor, due to the difficulty of users in interpreting their predictions.

3.5 Tree structured regression methods

Tree structured regression can be seen as a kind of additive model [HT90] also named *piecewise regression* that is in the form:

$$f(\boldsymbol{x}) = \sum_{i=1\ldots l} I(\boldsymbol{x} \in D_i) \times f_i(\boldsymbol{x}), \tag{3.31}$$

where:

- $D_1, \ldots, D_l$ is a partitioning of the training set D in a set of disjoint regions such that $\bigcup\limits_{i=1\ldots l} D_i = D$ and $\bigcap\limits_{i=1\ldots l} D_i = \phi$,

- $I(.)$ is an indicator function returning 1 if its argument is true and 0 otherwise,
- $f_1, \ldots, f_l$ are regression functions such that each f_i is fitted within the region D_i.

Models of this type provide a propositional logic representation of the regions corresponding to $D_1, \ldots, D_l$ in the graphical form of a tree and, at the same time, they fit a regression function f_i within each region D_i.

Definition 3.1 *A (rooted) tree τ is a finite set of nodes N_τ and an associated relation $B_\tau \subseteq N_\tau \times N_\tau$ for which the following properties hold:*

1. *there exists exactly one node $t_0 \in N_\tau$, named root, such that $\forall \langle t_i, t_j \rangle \in B_\tau : t_j \neq t_0$,*
2. *$\forall t_j \in N_\tau$ with $t_j \neq t_0$, there exists only one node $t_i \in N_\tau$ such that $\langle t_i, t_j \rangle \in B_\tau$.*

♦

The set of nodes N_τ can be partitioned into internal nodes N_τ^I and leaves N_τ^L such that:

- $N_\tau^I = \{t_i \in N_\tau |$ there exists at least a node $t_i \in N_\tau$ such that $\langle t_i, t_j \rangle \in B_\tau\}$,
- $N_\tau^L = \{t_i \in N_\tau |$ there exists no node $t_j \in N_\tau$ such that $\langle t_i, t_j \rangle \in B_\tau\}$.

Note that a binary tree τ is a special case of rooted tree such that for each internal node $n_i \in N_\tau^I$ there are no more than two distinct nodes $n_h, n_k \in N_\tau$ $(n_h \neq n_k)$ such that $(n_i, n_h) \in B_\tau$ and $(n_i, n_k) \in B_\tau$.

Definition 3.2 *Given a set D of n training regression cases, each of which is described by both m (continuous and discrete) predictors $X_1, \ldots, X_m$ and a target continuous variable Y, it is possible to build a binary tree structured regression model that is a binary rooted tree $\tau = (N_\tau, B_\tau)$, where:*

1. *each node $t_i \in N_\tau$ is associated with a subset of D, denoted by $D(t_i)$,*
2. *the root t_0 is associated with D itself,*
3. *each leaf node $t_i \in N_\tau^L$ is labeled with a regression function $f_{t_i} : X_1 \times \ldots \times X_m \rightarrow Y$,*
4. *each edge $\langle t_i, t_j \rangle \in B_\tau$ is labeled with $L_\tau(\langle t_i, t_j \rangle)$.*

$L_\tau(t_i, t_j)$ can be:

- *a test in the form $X_i \leq \alpha$ (or $X_i > \alpha$), which involves a continuous predictor variable X_i (*continuous splitting label*),*
- *a test in the form $X_i \in \{x_{i_1}, \ldots, x_{i_s}\}$ (or $X_i \notin \{x_{i_1}, \ldots, x_{i_s}\}$), which involves a discrete predictor variable X_i (*discrete splitting label*).*

♦

The general approach to build a binary tree structured regression model τ from a training set D is to start at the root and perform a *top-down* induction. At each internal node $t_i \in N_\tau^I$ the region $D(t_i)$ is recursively split according to the mutually exclusive tests (queries) labeling the two edges outgoing from t_i until the tree is sufficiently accurate. The best split is chosen according to a splitting rule.

An internal node t_i is a continuous (discrete) splitting node iff there exists an edge $\langle t_i, t_j \rangle \in B_\tau$ such that $L_\tau(\langle t_i, t_j \rangle)$ is a continuous (discrete) splitting label.

If p_i is the path from the root t_0 to a leaf node $t_i \in N_\tau^L$:

$$p_i = \langle t_0, t_1 \rangle, \langle t_1, t_2 \rangle, \ldots, \langle t_{i-1}, t_i \rangle,$$

then the label associated with p_i, $L_\tau(p_i)$, is the conjunction of labels:

$$L_\tau(p_i) = L_\tau(\langle t_0, t_1 \rangle) \wedge \ldots \wedge L_\tau(\langle t_{i-1}, t_i \rangle),$$

which provides an intentional description of the region corresponding to the partition $D(t_i)$.

The tree τ maps a generic regression case $\boldsymbol{x}$ into a leaf $t_i \in N_\tau^L$ through the path p_i as follows: starting with the root node t_0, it selects its child node t_1 such that $\boldsymbol{x}$ satisfies the test associated with $L_\tau(\langle t_0, t_1 \rangle)$. This procedure is recursively applied to t_1 until the leaf node t_i is reached and $y = f_{t_i}(\boldsymbol{x})$ is returned.

Using the more concise representation of Equation 3.31, the tree τ can be equivalently written as:

$$f(\boldsymbol{x}) = \sum_{t_i \in N_\tau^L} I(\boldsymbol{x} \in L_\tau(p_i(\boldsymbol{x}))) \times f_{t_i}(\boldsymbol{x}),$$

where $\boldsymbol{x} \in L_\tau(p_i(\boldsymbol{x}))$ when $\boldsymbol{x}$ satisfies $L_\tau(p_i(\boldsymbol{x}))$.

When each regression function f_{t_i} is a constant value, τ is named regression tree. Model trees are generalizations of regression trees, which associate multiple linear functions (or quadratic functions, kernel functions, etc.) with each leaf. In this way, regression trees perform the same prediction for all data falling in the same leaf while model trees may predict different values also for data falling in the same leaf.

Both regression and model trees are known for their comprehensibility as well as simplicity and efficiency when dealing with large domains. However, they can suffer of overfitting training data and lower predictive accuracy on unreliable data (i.e. noise). To avoid overfitting problems, a solution is the simplification of an overly large tree. In spite of their advantages, regression and model trees are also known for their instability [Bre96]. A small change in training set may lead to a different choice when building a node, which in turn may represent a drastic change in the tree, particularly if this change occurs in top level nodes. The function approximation provided by standard regression and model trees is highly non smoothed leading to very marked function discontinuities. Although there are applications where this may be advantageous, regression functions are typically supposed to have a certain degree of smoothness that is hardly captured by standard tree models. Some hybrid tree models that improve smoothness of tree structured approximation are presented in [Tor99].

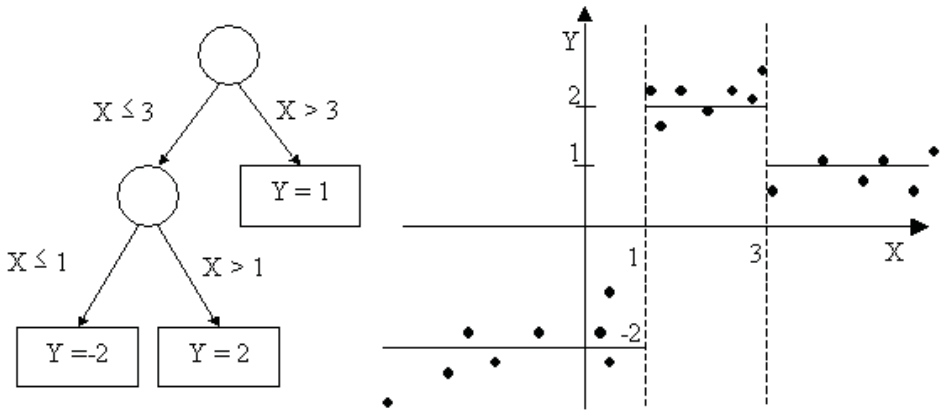


FIGURE 3.4: An example of a regression tree with one continuous predictor X.

3.5.1 Regression trees

A regression tree is a piecewise constant constructed by recursively partitioning the training data D (see Figure 3.4). Its name derives from the practice of displaying the partitions as a decision tree with continuous constant values associated with leaves. AID algorithm [MS63] is the first implementation of this idea. However, the major reference on regression trees is the seminal work of [BFOS84] that provides a thorough description of the system CART.

CART is suitable for the prediction of both a continuous target variable (i.e., regression trees) and a discrete target variable (i.e., classification trees). It performs the top-down induction of a regression tree τ by recursively partitioning D and growing τ by introducing, at each step, a splitting node t_i until stopping criteria are not fulfilled and a leaf is created. Branches of a splitting node t_i are then built by applying the same algorithm for partitions of $D(t_i)$ obtained according to splitting tests outgoing from t_i. At each step, the best node is chosen with respect to a local criterion, that is, least square (LS) regression or least absolute deviation (LAD) error.

LS regression trees induction is formally based on the Lemma 3.1 that has paved the way for building a regression tree τ by minimizing the expected value of the sum of square error with the expected value (i.e mean value) of the target variable Y. Each leaf $t_i \in N_\tau^L$ is then assigned with a constant $c(t_i)$ that is the average of Y values for the $n(t_i)$ training cases falling in $D(t_i)$:

$$f_{t_i}(\boldsymbol{x}) = c(t_i) = \frac{1}{n(t_i)} \sum_{(\boldsymbol{x}_j, y_j) \in D(t_i)} y_j. \tag{3.32}$$

Assuming the constant $c(t_i)$ obtained with Equation 3.32, LS regression suggests the natural goodness of choosing the splitting test outgoing from any node t_i, which most reduce the mean square error on $D(t_i)$.

We denote by:

$$MSE(t_i) = \frac{1}{n(t_i)} \sum_{(\boldsymbol{x}_j, y_j) \in D(t_i)} (y_j - c(t_i))^2 \tag{3.33}$$

the fitting error of a node $t_i \in N_\tau$. The mean square error of a regression tree τ is then the weighted average of the square errors in its leaves:

$$MSE(\tau) = \sum_{t_i \in N_\tau^L} p_i \times MSE(t_i) \tag{3.34}$$

$$= \sum_{t_i \in N_\tau^L} (\frac{n(t_i)}{n} \times \frac{1}{n(t_i)} \sum_{(\boldsymbol{x}_j, y_j) \in D(t_i)} (y_j - c(t_i))^2) \tag{3.35}$$

$$= \sum_{t_i \in N_\tau^L} p_i \times s^2(t_i). \tag{3.36}$$

where $p_i = \frac{n(t_i)}{n}$ is the estimated probability that a training case reaches the leaf t_i and $s^2(t_i) = \sum_{(\boldsymbol{x}_j, y_j) \in D(t_i)} (y_j - c(t_i))^2$ is the sample variance at t_i.

At each step, LS splitting rule chooses the split which maximizes the decrease in variance of the tree resulting from this splitting. In binary trees, where an internal node t_i has two children, this corresponds with minimizing the weighted average of variance on the resulting sub-nodes , that is:

$$MSE(t_i, t_{i_L}, t_{i_R}) = \frac{n(t_{i_L})}{n(t_i)} \times s^2(t_{i_L}) + \frac{n(t_{i_R})}{n(t_i)} \times s^2(t_{i_R}), \tag{3.37}$$

where t_{i_L} (t_{i_R}) is the left (right) child of t_i.

In the case t_i is a binary continuous splitting node then $L(t_i, t_{i_L}) = X_i \leq \alpha$ and $L(t_i, t_{i_R}) = X_i > \alpha$. Possible values of α are found by sorting the distinct values of X_i in $D(t_i)$, then identifying a threshold (e.g. midpoint) between each pair of adjacent values. If the cases falling in t_i have k distinct values for X_i, $k-1$ thresholds are considered. Obviously, the lower $MSE(t_i, t_{i_L}, t_{i_R})$ with $L(t_i, t_{i_L}) = X_i \leq \alpha$ and $L(t_i, t_{i_R}) = X_i > \alpha$, the better the split $X_i \leq \alpha$ according to LS criterion.

Conversely, if t_i is a binary discrete splitting node then $L(t_i, t_{i_L}) = X_i \in \{x_{i_1}, \ldots, x_{i_s}\}$ and $L(t_i, t_{i_R}) = X_i \notin \{x_{i_1}, \ldots, x_{i_s}\}$. We denote by k the number of distinct values for X_i and $S_{X_i} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$ the set of distinct values of X_i, S_{X_i} is sorted according to the sample mean of Y over all cases in $D(t_i)$. A theorem by Breiman et al. [BFOS84](Theorem 4.5, Proposition 8.16) proves that the best binary split is one of $k-1$ partitions $\{x_{i_1}, \ldots, x_{i_s}\}$ and $S_{X_i} - \{x_{i_1}, \ldots, x_{i_s}\}$, thus greatly reducing the search for the best subset of categories from 2^{k-1} to k-1 partitions.

The benefits of LS criterion have been extensively discussed in [BFOS84] and some simplifications of LS splitting criterion which lead to gains in computational efficiency are then proposed in [Tor99]. Buja and Lee [BL01] have argued that LS criterion, in spite of its computational advantages, can appear not adequate for several applications such as outlier processing. Outliers are generally a problem for LS regression trees since they may distort the selection of the best splits and have a large impact on the average value chosen for the leaves of the trees [FP99]. A solution can be found in adopting a LAD criterion, where the difference between LS and LAD lies in the use of medians instead of averages in the leaves and mean absolute deviation as error criterion.

The main advantage of LAD regression trees is the robustness of the obtained models since medians and absolute deviations are known to be more robust with

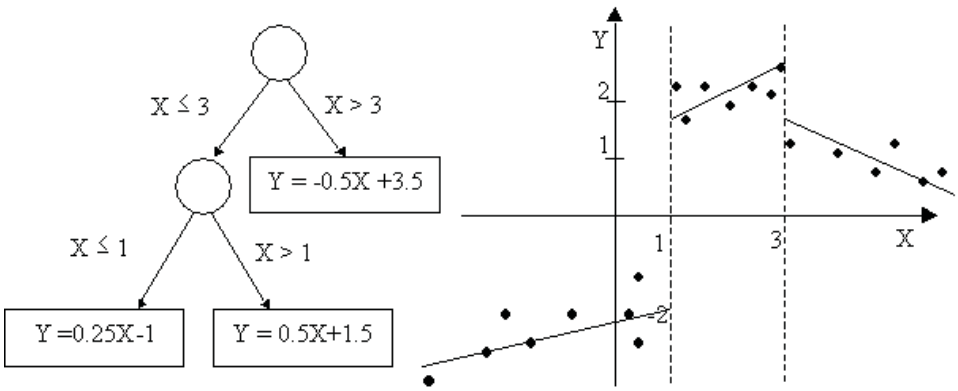


FIGURE 3.5: An example of a linear model tree with one continuous predictor X.

respect to the presence of outliers and skewed distributions. However, LAD criterion brings additional computational difficulties to the task of growing a tree. Moreover, with respect to discrete variables, Torgo [Tor99] shows that the theorem proved by Breiman et. al. [BFOS84] for subset splits in LS regression trees does not hold for the LAD error criterion. Still, he has experimentally observed that the use of the results of this theorem as a heuristic method of obtaining the best split does not degrade predictive accuracy.

Finally, a splitting criterion based on the F-measure [Van79] has been proposed in [TR03] with the objective of inducing regression trees that are accurate at predicting outliers but are also interpretable from the user perspective.

3.5.2 Model trees

Model trees are analogues to piecewise multiple functions: they are tree structured models, but whereas regression trees have constant values at their leaves, model trees may have multiple functions (see Figure 3.5). In this way, model tree induction combines the achievement of CART-like top-down construction of tree models with the main results in fitting training samples with regression models whose basis functions are splines [Fri91] or polynomial equations [DT95] [TLD04].

Some of the model tree induction systems developed are: M5 [Qui92] [Qui93b], RETIS [Kar92], M5' [WW97] [FWI+98], TSIR [Lub94], HTL [Tor97], which has been subsequently included in RT [Tor99], SUPPORT [CHLY94], which has been extended in GUIDE [Loh02], and SECRET [DG02]. All these systems perform a top-down induction of model trees (TDIMT) by recursively partitioning the training set D and associating multiple functions with leaves.

M5 [Qui92][Qui93b] and its further extension M5'[4] [FWI+98][WW97] partition feature space by choosing at each step the split that minimizes a measure of the standard deviation computed with respect to the average of Y values falling in the regions under analysis. A multiple linear model is then associated with each leaf. Regression coefficients of this model are locally determined by least square

[4] M5' is an implementation of M5 that also deals with enumerated variables and treats missing values.

regression on the set of training cases falling in the leaf. Only continuous variables involved in tests along the path from the root to the current leaf are actually included in regression model which is then simplified by dropping useless variables if this results in a lower expected error on future data (more specifically, if the decrease in the number of parameters outweighs the increase in the observed training error). Finally, a smoothing mechanism is applied to adjust the predicted value by combining the model at a leaf with the models on the path starting in the root to form the final model that is placed at the leaf.

RETIS [Kar92] partitions feature space and minimizes $\frac{n_{t_{i_L}}}{n(t_i)} s^2(t_{i_L}) + \frac{n_{i_R}}{n(t_{i_R})} s^2(t_{i_R})$ at each step. In this case, $s^2(t_{i_L})$ ($s^2(t_{i_R})$) is computed as the mean square error with respect to the regression plane $f_{t_{i_L}}$ ($f_{t_{i_R}}$) fitting training data falling in the left (right) child. Multiple linear functions are associated with leaves. Each regression function is locally determined by least square regression on training cases falling in the leaf and involves all continuous predictor variables.

SUPPORT, GUIDE and SECRET transform a regression problem into a classification problem, and then choose the best partition on the basis of computationally efficient evaluation functions developed for classification tasks. They associate a polynomial equation with each leaf. SUPPORT implements some smoothing mechanism that uses a weighted average to combine piecewise polynomial regression estimates into a single smooth one. Similarly GUIDE, which has been specifically designed to extend SUPPORT in order to eliminate variable selection bias, is able to associate either a complex model (e.g. polynomial) or a simple linear model with each leaf.

HTL [Tor97] chooses the split that minimizes a measure of the variance on child nodes. Similarly to evaluation measure adopted in both M5 and M5', this evaluation measure is computationally efficient since it involves the average of Y values falling in the regions under analysis. Moreover, HTL is a hybrid system since it integrates several approaches to regression from linear regression to kernel regression and k-nearest neighbor models.

Integrating partition-based methods with instance-based models is not a novel idea. The work of Weiss and Indurkhya [WI95] already integrates a rule-based partitioning method with k-nearest neighbors. These authors deal with regression by mapping it into a classification problem and transforming the original Y values into a set of I intervals.

Deng and Moore [DM95] have combined Kd-trees [Ben75][5] with kernel regression producing what they call kernel regression trees and looking at kernel regression as a form of lazy learner.

Torgo [Tor00] has proposed to merge partial linear regression and model tree induction. Partial linear regression [Har90] is a semi-parametric technique that integrates a linear polynomial with a kernel smoother applied to the residuals (error) of polynomials on the neighborhood. By integrating these two approaches partial linear functions at leaves loose some of the intelligibility of linear polynomials,

[5] Kd-trees are binary trees built in a similar fashion as regression trees. However, while regression trees are built with the goal of grouping training data with similar Y values, Kd-trees try to optimize the storage of these data in order to achieve faster access time.

however they may show significant accuracy improvement in non-linear domains.

In alternative to partial linear trees, Torgo and Costa [TP00] have investigated the combination of partial linear regression with a clustering step to obtain sub-set of training data before learning phase. After this resampling process a different regression model is fit for each cluster.

TSIR associates a multiple linear model with each leaf of induced model tree. Nevertheless, TSIR differs from other TDIMT systems (e.g. M5, RETIS) since it performs construction of multiple linear models according to the forward stepwise procedure [DS82]. This means that partitioning steps are intermixed with regression steps. The former correspond with splitting a region of D according to a boolean test on a predictor variable X_i, while the latter computes a single variable linear regression, $\hat{Y} = \beta_0 + \beta_1 X_i$, and passes down to its *unique* child the residuals $y_j - (\beta_0 + \beta_1 x_{i_j})$ as new values of the target variable.

It is noteworthy that TDIMT systems associate functional model with leaves but in general they do not use functional models in splitting tests. This is argued in [Gam04], where functional trees, which build a function linear model at each node and use it later in the pruning phase, are presented. Functional trees allow decision associated with internal nodes in functional form so that they divide training data into oblique decision surfaces[6]. Oblique splits are linear inequalities involving two or more predictor variables.

Finally, all TDIMT systems, here presented, operate on the entire training set, but incremental learning may be advantageous in many real-world regression applications such as when input data is a continuous stream data. The incremental induction of trees has received significant attention in literature for classification tasks [UBC97] [GRM03], while model tree incremental induction has only recently been addressed. Potts [Pot04b] has proposed to incrementally induce model trees according to an incremental evaluation function that is based on the Fisher statistic already used by Siciliano and Mola [SM94] to grow linear model trees in batch setting. A faster incremental splitting rule that is based on GUIDE evaluation function is discussed in [Pot04a].

3.5.3 Simplification methods

When building regression or model trees, it is common practice to discard parts of the tree that describe spurious effects in the training data rather than true features of underlying phenomenon. The application of model tree simplification (pruning) methods follows the generation (growing) of the tree itself and tries to keep the overfitting problem under control. Several simplification methods have been reported in the literature, most of which are derived from those developed for decision trees [EMS97].

CART prunes a large regression tree by means of error complexity pruning [BFOS84]. It consists of two steps:

1. selection of $\{\tau_0, \tau_1, \ldots, \tau_L\}$ that is a parametric family containing subtrees of the tree τ_{max},

[6] Oblique partitioning of training data is also supported by GUIDE as well as SECRET.

2. choice of the best tree τ_i in the parametric family.

Pruning is based on a measure of the error rate of a tree τ named *error-complexity*, which is defined as follows:

$$EC_\alpha(\tau) = MSE(\tau) + \alpha \times \#N_\tau^L, \tag{3.38}$$

where $MSE(\tau)$ is the mean square error of τ, $\#N_\tau^L$ is the cardinality of the set N_τ^L containing the leaves of τ, while α is named *complexity parameter* and defines the cost of each leaf.

As regards the first step, the basic idea is that τ_{i+1} is obtained from τ_i by pruning those branches that show the lowest increase in apparent error rate per pruned leaf. Therefore, when a tree τ is pruned in a node t_i, its apparent error rate increases by the amount $EC_\alpha(t_i) - EC_\alpha(\tau_{t_i})$ with τ_{t_i} sub-tree of τ rooted in t_i, while its number of leaves decreases by $\#N_{\tau_{t_i}}^L - 1$ units. Thus, the following ratio:

$$\alpha = \frac{EC_\alpha(t_i) - EC_\alpha(\tau_{t_i})}{\#N_{\tau_{t_i}}^L - 1}, \tag{3.39}$$

measures the increase in apparent error rate per pruned leaf. Then, τ_{i+1} in the parametric family is obtained by pruning all nodes in τ_i with the lowest value of α. The first tree τ_0 is obtained by pruning τ_{max} of those branches whose α value is 0, while the last tree τ_L is the root tree. Each tree τ_i is characterized by a distinct value α_i, such that $\alpha_i < \alpha_{i+1}$. Therefore, the set $\tau_0, \tau_1, \ldots, \tau_L$ is a parametric family of trees that may be denoted as $\tau_{max}(\alpha)$.

In the second step, the best tree in $\tau_{max}(\alpha)$ with respect to predictive accuracy is chosen. Breiman and his colleagues propose two distinct ways of estimating the accuracy of each tree in the family, one based on cross-validation sets, and the other on an independent pruning set [BFOS84].

The cost complexity pruning has been also adopted to prune model trees mined with GUIDE [Loh02].

RETIS bases its pruning algorithm on Niblett and Bratko's method [NB86], extended later by Cestnik and Bratko [CB91]. Contrarily to CART pruning, this method proceeds in a single step by running in a bottom-up fashion through all nodes of τ_{max} [Kar92]. Pruning is based on the idea that for every node an estimation of its prediction error on test cases is made. In each node, the algorithm makes an estimate of the static error e_s (i.e. the expected error on unknown cases if the tree is pruned at this node) and the backed-up error e_b (i.e. an estimation of expected error when the tree is not pruned at that node). If the static error is less than or equal to the backed-up error the subtree is pruned and the node is converted to a leaf. Karalic and Cestnik [KC91] have proposed to use the Bayesian approach (m-estimate) to estimate the errors of the tree nodes on unseen examples. The m parameter has intuitive meaning, but can also be set by a cross-validation.

M5 uses a pessimistic error pruning-like strategy [Qui92] [Qui93b] since it compares the error estimates obtained by pruning a node or not. The error estimate is based on training cases and corrected in order to take into account the complexity of the model in the node.

Similarly, in M5' the pruning procedure makes use of an estimate, at each node, of the expected error for the test data. The estimate is the *MSE* compensated by a factor that takes into account the number of training examples and the number of parameters in the linear model associated with the node [WW97].

A method à la error-based-pruning is adopted in HTL [Tor99], where the upper level of a confidence interval of the mean square error estimate is taken as the most pessimistic estimate of the error node [Qui92].

A different solution, based on MDL principle [Ris82] has been proposed by Robnik-Sikonja and Kononenko [RK98] for the TDIMT system CORE [Rob97]. MDL principle is based on the Occam's statement that it is vain to do with more what can be done with less. Consequently, its use can be summarized as: an code the possible solutions (i.e. tree models) to the problem and select the instance with the shortest code as the result. Encoding determines the binary code length of a tree-based model, where the binary code of a model tree consists of the code of both the model and its errors.

3.6 Regression rules methods

Rule induction model finds solutions to regression problems in disjunctive normal form such that the model:

$$\textit{if } \boldsymbol{x} \in D_i \textit{ then } y = f_i(\boldsymbol{x}) \tag{3.40}$$

can be applied to it. Each rule in the rule-set represents a single partition of the training set D (or region D_i). However, unlike tree regions, the regions for rules need not be disjoint.

For partially overlapping regions, several rules may be satisfied for a single case. Therefore, some mechanism is needed to resolve the conflicts in predicted target value when multiple rules, D_i regions, are involved. One standard model [WI93a] is to order the rules. Such ordered rule-sets have been referred to as *decision lists*. The first rule that is satisfied is selected as follows:

$$\textit{if } i < j \wedge \boldsymbol{x} \in D_i \wedge \boldsymbol{x} \in D_j \textit{ then } f(\boldsymbol{x}) = f_i(\boldsymbol{x}) \tag{3.41}$$

Weiss and Indurkhya [WI93b] have developed a regression rule induction system, named SWAP1R, that induces regression rules expressed in a propositional language by exploiting a covering strategy. This strategy is the analogous of rule induction procedure for classification tasks [MMHL86] [CN89]. The conditional part (i.e. antecedent) of the induced rules consists of a conjunction of tests on the input variables while the conclusion part (consequent) contains the target prediction. Originally these predictions consisted of the average Y value for the cases satisfying the conditional part, but later Weiss and Indurkhya [WI95] have extended the system with the possibility of using k-nearest neighbors in the prediction step. Indeed, the authors propose to map a regression problem into a classification problem. This means that the target values of the training cases are pre-processed by grouping them into a set of user-defined bins. These bins act as class labels in

the subsequent learning phase. Consequently, the algorithm induces a classification model of the data, where the classes are numbers representing the obtained bins. These numbers can then be used to make continuous predictions for unseen cases.

This idea of transforming a regression problem into a classification task has been taken further in the work of Torgo and Gama [TG96], where a system named RECLA acts as a generic pre-processing tool that allows virtually any classification system to be used in a regression task.

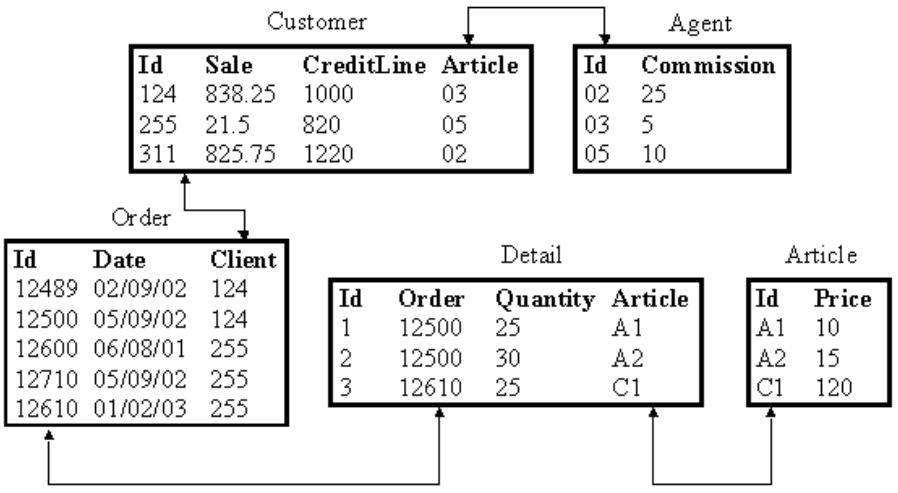
Torgo [Tor95] has developed a propositional regression rule learner (R2) that uses an if-then rule format. This algorithm can use several different functional models in the conclusion of the rules. R2 employs a covering algorithm that builds new models while there are uncovered cases to fit. The model is chosen from the model lattice that includes all possible regression models for the conclusion part of the rules. The result of this first step is a rule with empty conditional part and the built model as conclusion. This rule is then specialized by adding conditions to restrict the model domain of applicability with the goal of improving its fit. This restriction is guided by an evaluation function that weighs both the fitting as well as the degree of coverage of the rule. This means that the system is looking for rules with good fit but covering as many cases as possible.

CUBIST[7] is a commercial data mining system developed by Ross Quinlan, which learns a set of unordered if-then rules with linear models in the conclusion part. This system looks like a kind of C4.5rules [Qui93a] for M5. The system is able to deal with both continuous and discrete variables, and obtains a piecewise linear model of the data. CUBIST can also combine the predictions of this model with k-nearest neighbor predictions.

SAFE [Que00] is another machine learning system that induces regression rules from training cases. It extends the usual way to divide continuous values into intervals to all continuous variables, not only to the target one. The relevant intervals so found, in addition to discrete data, are then processed to discover the relationships between these two data types. The result, SAFE, is able to return a small list of accurate and interpretable production rules whose conclusions are linear functions.

Finally, Saito and Nakano [SN02] have proposed a method for extracting regression rules from neural networks trained with multivariate data containing both discrete and continuous variables. Each regression rule is expressed as a pair formed by a logical formula on the conditional part over discrete variables and a polynomial equation on the action part over continuous variables. The proposed extraction method first generates one such regression rule for each training sample, then utilizes the k-means algorithm [Mac67] to generate a much smaller set of rules having more general conditions, where the number of distinct polynomial equations is determined through cross-validation. Finally, this method invokes decision-tree induction to form logical formulae of discrete conditions as conditional parts of final regression rules.

[7] http://www.rulequest.com/cubist-info.html.



Id	Sale	CreditLine	Article
124	838.25	1000	03
255	21.5	820	05
311	825.75	1220	02

Id	Commission
02	25
03	5
05	10

Id	Date	Client
12489	02/09/02	124
12500	05/09/02	124
12600	06/08/01	255
12710	05/09/02	255
12610	01/02/03	255

Id	Order	Quantity	Article
1	12500	25	A1
2	12500	30	A2
3	12610	25	C1

Id	Price
A1	10
A2	15
C1	120

Target examples	Background knowledge
customer(124, 838.25, 1000, 03). customer(255, 21.5, 820, 05). customer(311, 825.75, 1220, 02)	order(12489, '02/09/02', 124). order(12500, '05/09/02', 124). order(12600, '06/08/01', 255). order(12710, '05/09/02', 255). order(12610, '01/02/03', 255). agent(02, 25). agent(03, 5). agent(05, 10). detail(1, 12500, 25, 'A1'). detail(2, 12500, 30, 'A2'). detail(3, 12610, 25, 'C1'). article('A1', 10). article('A2', 15). article('C1', 120).

FIGURE 3.6: Relational data: target examples and relational background knowledge.

3.7 The relational view

All regression methods reported above suffer from an important limitation, that is, the restriction to propositional representation (i.e. *single table assumption*) [Wro01]. Consequently, none of these methods is able to directly predict continuous values from not only target examples but also relational background knowledge (see Figure 3.6), though relational data occurs in many real-world application where data are naturally stored in *multiple* tables of a relational database.

Thus far, only few methods have been proposed to face with regression problems in the context of relational data. One approach is to propositionalize relational data into a single table format that traditional attribute-value algorithms can handle. This approach has been employed in regression tasks as well. Džeroski and his colleagues [DTU95] have proposed to apply DINUS [LD94] algorithm to transform a Datalog representation of a dynamic method into a propositional form (i.e., attribute-value pairs), so that the classical model tree induction method RETIS

based on the single-table assumption can be subsequently applied to the transformed problem. However, this transformation does not work for background knowledge, which is a limitation of this approach.

The alternative is to directly induce *relational regression models* over relational data without any transformation of the original problem. This idea has received some attention within the field of ILP. The problem of *multi-Relational Regression* [Dž95] has been formulated in the normal ILP framework[8]. Nevertheless relational regression differs from other ILP learning tasks in that there are no negative examples.

As a consequence, some methods, which face with relational regression problems in their original form, have been investigated within the field of ILP.

FORS [Kar95] [KB97] has been proposed to induce first-order logic concepts that incorporate real-valued variables. The system has been designed with the goal of mining background knowledge expressed in intensional form, modeling dynamic phenomena (e.g. learning from time series) as well as handling noisy data. More precisely, FORS receives, as input, samples of the unknown regression function and background knowledge and uses a *separate-and-conquer* strategy to partition training space into sub-regions and build a regression local model for each region separately.

The model built by FORS is a Prolog program consisting of the background knowledge and clauses in the form:

$$f(Y, X_1, \ldots, X_m) :- literal_1, \ldots, literal_k, !.$$

During the induction, FORS uses a covering approach named *sequential covering* similar to the one employed by FOIL [Qui90]. The algorithm repeatedly constructs a clause. When a clause is found, all examples covered by the clause are removed from the training set, and the procedure is repeated, while there are enough examples left.

Two kinds of clauses can be generated during clauses construction, that is, candidate clauses which do not instantiate the target variable (e.g. $f(Y, X, Z) : -X <= Z$) and candidate clauses, named complete clauses, which instantiate the target variable Y (e.g. $f(Y, X, Z) : -X <= Z, sqr(Y, X)$, presuming that background knowledge contains clause $sqr(Y, X) : -Y\ is\ X * X$).

The simplest approach to obtain a *complete clause* is to add a literal *Y is Mean*, where *Mean* is the mean value of the target variable on the examples covered by the clause. However, this approach is unable to detect linear dependencies among the target variable and the continuous predictor variables. To overcome this limitation, FORS builds a regression plane through the target values of the covered examples.

The clause construction is performed top-down. The algorithm starts with the most general candidate clause, covering the entire example set (i.e. the clause with no condition part) and then specializes it by adding literals. Clause construction uses beam search to guide the algorithm through the space of possible clauses. During a single specialization step for one clause, an attempt is first made to specialize

[8]Normal ILP framework is not part of non-monotonic ILP framework that includes the closed world assumption.

a clause with background knowledge literals then with variable-value literals[9] and finally with recursive literals.

For each candidate clause, the mean square error on training data covered by the clause, is computed to estimate the clause's error when predicting unlabeled cases.

Since mean square error on training data is used as error estimate, a method that simply minimizes the estimated error tends to overfit training data and predict unrealistic performance on un-labeled new data. To avoid overfitting training data, several pre-pruning mechanisms for controlling clause generation have been adopted to limit the amount of clause's adaptation to training data. They are based on the minimal number of examples that a clause must cover, maximal variable depth, call depth, minimal improvement of candidate clause with respect to its predecessor, allowed error of a clause and minimum description length (MDL) principle.

Finally, when FORS is faced with a time series each example represents the state of variables at one point in time. The time intervals between successive points need not be equal. FORS can compute time derivatives of any variable with respect to the time variable. This enables FORS to construct models of dynamical systems which are typically described with a set of differential equations.

FFOIL [Qui96] is another example of ILP system able to deal with relational regression. It is a derivation of the FOIL system [Qui90] implements a covering algorithm similar to FORS. It starts with an empty program and keeps adding clauses until all cases are covered. Each added clause starts with an empty body and literals are appended as a form of specializing the clause. A function with k arguments (the input variables) is represented by a k+1-ary relation where one argument holds the function outcome. The result obtained by FFOIL consists of a Prolog program with clauses of this relation.

An alternative to the separate-and-conquer strategy adopted by both FORS and FFOIL is the *divide-and-conquer* strategy. This leads to a tree-based algorithm rather than a covering algorithm with all advantages and disadvantages known from other algorithms of these type (tree-based vs covering). A discussion of both strategies in the context of top-down induction of logic programs can be found in [Bos95]. On one hand, the hypothesis space for separate-and-conquer is larger than for divide-et-conquer, so that more compact hypotheses can be found using separate-and-conquer. On the other hand, building a tree is computationally cheaper than searching for rules. In [WI95], a comparison between tree induction and rule induction in propositional regression basically draws the same conclusions, although the approach to rule-based regression in [WI95] is different from FORS since it involves the discretization of continuous variables.

SRT [Kra96], S-CART [Kra99][KW01] and TILDE-RT [Blo98] are some examples of ILP systems that implement the divide-and-conquer strategy. They perform top-down induction of multi-relational tree structured regression models and then translate these models into Prolog programs. Similarly to FORS and FFOIL, all these systems solve the multi-Relational Regression problem in its original form,

[9] A variable-value literal is a literal of the form $X_i <= \alpha$ or $X_i > \alpha$ for continuous variable, $X_i = \alpha$ for discrete variable.

and do not require any transformation of the problem. Moreover, they can utilize relational background knowledge.

SRT generates a series of increasingly complex trees containing a literal (an atomic formulation or its negation) or a conjunction of literals in each node, and subsequently returns the best tree according to a criterion based on minimum description length. A numerical constant value is assigned to each leaf.

SCART, that integrates and extends SRT, is a system that learns a theory for the prediction of either discrete classes (i.e. classification tasks) and continuous values (i.e. regression tasks) from examples and relational background knowledge.

At the top level, the main stages of S-CART are *growing the tree*, *pruning the tree* and *adding linear regression models*. An initial binary tree is recursively grown based on the training set, selecting a possible conjunction of one or more literals in each node as provided by user-defined schemata [SP91] until a stopping criterion is fulfilled. The selection of the best split in regression case corresponds with minimizing the resulting average *MSE* within the deriving subsets. The constant predicted in a node is simply the mean of the target variable values for all training individuals covered by the node. This tree is usually very complex, therefore it is cut back to appropriate size in order to avoid overfitting.

In S-CART, pruning consists of estimating the optimal value of the complexity parameter α on a separate pruning set or cross validation.

Optionally, S-CART may add a linear regression model to leaves of pruned tree. For each leaf in the pruned tree, multiple linear stepwise regression is applied to the examples in the respective leaf, with varying values of the significance parameter that determines whether a variable is included in the regression model or not. Only when a variable contributes significantly to explain the variation of target variable, it is effectively added to the linear regression model.

S-CART also keeps track of the individuals in each node and the conjunctions of literals in each path leading to the respective node. This information can be turned into a clausal theory that is a set of first order regression rules.

TILDE-RT is the regression sub-system of TILDE[10] [Blo98] [BD98]. Its splitting criterion is based on minimizing the resulting average variance, while F-test is used to verify that the split causes a significant reduction of variance for the target variable Y. Only a constant value (i.e. mean of Y values for training data falling in the leaf) is associated with each leaf.

3.8 Conclusions

Many real-world applications involve the prediction of continuous values. In particular, some domains are concerned with the prediction of continuous values from both examples and relational (mostly non-determinate) background knowledge.

In this chapter, we have provided an overview of existing approach to regression both in propositional and relational setting. We have analyzed the major regression paradigms developed in statistic, distinguishing among global paramet-

[10] TILDE is the upgrade of C4.5 [Qui93a] to multi-relational setting.

ric approaches, non parametric approaches and additive approaches and discussed the main advantages and drawbacks of k-nearest neighbor regression, artificial neural networks, regression and model trees as well as regression rules learning. We have also outlined prospects and challenges for the role of ILP in multi-relational regression.

With respect to computational efficiency, it is clear that multi-relational formalism explores a much larger search space than their propositional counterparts. This fact has two contradictory effects. While being able to find solutions not available to propositional systems, this increased expressiveness has a strong impact in the computational complexity of these systems. This makes them hardly applicable to extremely large domains that are found in some applications (like in a typical data mining situation). However, the expressiveness of first-order logic is too attractive a feature for data mining research to ignore. Moreover, computation power grows at a very fast rate and ILP may well become the major trend in machine learning and data mining approaches to regression.

Chapter 4

Stepwise Model Tree Induction

Model trees are tree structured regression models which associate leaves with multiple functions. They are used to solve prediction problems (i.e. regression problems) in which the target variable is continuous. In this chapter, we present SMOTI (Stepwise Model Trees Induction) that is a method for the data-driven construction of model trees. Its main characteristic is the construction of trees with two types of nodes: regression nodes, which perform only straight-line regressions, and splitting nodes, which partition training data. A multiple linear model is then built stepwise at each node by combining straight-line regressions reported along the path from the root to the current node. In this way, internal regression nodes contribute to the definition of multiple models and capture *global* effects, while straight-line regressions at leaves can only capture *local* effects. This allows SMOTI to potentially solve the problem of modeling phenomena, where some variables have a global effect while others have only a local effect and to prevent problems related to two-staged induction algorithms, which first build the tree structure and then associate leaves with multiple models. On the other hand, this stepwise construction provides a solution to problems of efficiency and collinearity by selecting only a subset of variables.

Experimental results on artificially generated datasets show that SMOTI outperforms two state of art model tree induction systems, M5' and RETIS, in accuracy. Results on benchmark datasets used for studies on both regression and model trees show that SMOTI performs better than RETIS in accuracy, while it is not possible to draw statistically significant conclusions on the comparison with M5'. Moreover, model trees induced by SMOTI are generally simple and easily interpretable, and their analysis often reveals interesting patterns.

However, similarly to other TDIMT approaches, SMOTI may generate model trees that overfit training data. We have investigated a posteriori simplification (or pruning) of model trees induced by SMOTI. In particular, we describe a general framework for simplifying model trees with regression and splitting nodes. This framework is helpful to define two simplification methods, named Reduced Error Pruning (REP) and Reduced Error Grafting (REG), and to investigate their theoretical properties. They are characterized by the use of an independent pruning

set.

The effect of simplification on model trees induced with SMOTI is empirically investigated. Results are in favor of simplified trees in most cases.

4.1 Background and motivations

Many problems encountered in common practice involve the prediction of a continuous attribute associated with a case. Let us consider here the classical (propositional) setting where observed data are generated independently and identically distributed from an unknown distribution P on some domain $\boldsymbol{X}$ and are labeled according to an unknown function g. The domain of g is spanned by m independent (or predictor) random variables X_i (both continuous and categorical), that is $\boldsymbol{X} = X_1 \times X_2 \times \ldots \times X_m$, while Y, the range of g, is a set of real numbers $\Re$. A TDIMT method inputs a training set $D = \{(\boldsymbol{x}, y) \in \boldsymbol{X} \times Y | y = g(\boldsymbol{x})\}$ and mines a piecewise function f in form of regression or model tree such that f is hopefully close to g on the domain $\boldsymbol{X}$. The general approach to inducing regression or model trees is to start at the root and perform top-down induction [1]. At each node, training cases are recursively partitioned according to a splitting test until some stopping criterion is fulfilled. If we denote by τ, the tree currently built, a potential split at a leaf t of τ corresponds with partitioning the training cases in t into two groups and obtaining a new tree τ', where t is an internal node with two children, say, t_L and t_R. Obviously, different splits generate distinct trees τ'. The original work by Breiman and his colleagues [BFOS84] proposes to choose the split that minimizes the corresponding mean square error $MSE(\tau')$ on training cases. Since, CART associates a constant with each leaf, the $MSE(\tau')$ is clearly minimized when this constant is the sample mean of the target variable on cases in the leaf. Therefore, the minimization of $MSE(\tau')$ is equivalent to minimizing the contribution to $MSE(\tau')$ provided by the split. This contribution is $p(t_L)s^2(t_L) + p(t_R)s^2(t_R)$, where $p(t_L)$ ($p(t_R)$) denotes the estimated probability that a training case reaches the leaf t_L (t_R), while $s^2(t_L)$ ($s^2(t_R)$) the sample variance at leaf t_L (t_R) of Y values.

This heuristic criterion, that has initially conceived for a regression tree, has also been used for model trees. In HTL the evaluation function is the same as that reported above, while in M5 and M5' the sample variance $s^2(t)$ is substituted by the sample standard deviation $s(t)$.

The problem with these evaluation functions, when used in model tree induction, is that they do not take into account the (multiple) models associated with the leaves of the tree, and the induced tree may fail to capture the underlying model. In principle, the optimal split should be chosen depending on how well each model fits the data. In practice, many model tree induction systems choose the optimal split on the basis of the spread of observed cases with respect to the *sample mean*. However, a model associated with a leaf of a model tree is generally more sophisticated than the sample mean. Therefore, *the evaluation function is incoherent with respect to the model tree being built.*

[1] An overview of methods for top-down induction of regression and model trees is presented in Chapter 3.

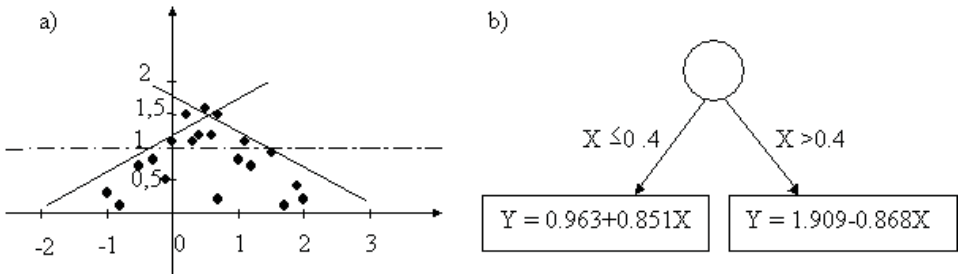


FIGURE 4.1: a) Scatter plot of twenty cases; the values of the only independent variable range between -1.0 and 2.0. A simple linear regression on the whole data set would give the dashed line. b) The underlying model tree partitions the training cases into two subgroups: $X \leq 0.4$ and $X > 0.4$.

To illustrate the problem, let us consider the dataset plotted in Figure 4.1.a and generated according to model tree in Figure 4.1.b. Neither M5 nor its commercial version CUBIST nor HTL are able to find the underlying model tree, because of net separation of the *splitting* stage from the *predictive* one, and in particular due to the fact that the partitioning of the feature space does not take into account the regression models that can be associated with the leaves. This seems to be inherited by regression tree learners, such as CART. In this case, however, the evaluation functions do take into account the models built in the leaves (the sample means). On the contrary, when we try to use the same heuristic criteria for model tree induction we are rating the effectiveness of a partition with respect to different models from the ones chosen in the subsequent predictive stage.

This problem is solved in RETIS, whose heuristic evaluation function used for a binary split minimizes a function of the mean square error (*MSE*) computed with respect to the regression planes found for both the left and the right child. In practice, for each possible partitioning, the best regression planes at leaves are chosen, so that the selection of the optimal partitioning can be based on the result of the prediction stage. The computational complexity of this evaluation function is cubic in the number of independent continuous[2] variables, m, and quadratic in the number of training observations, n. Indeed, the selection of the first split takes time $\boldsymbol{O}(m \times n \times \lg n)$ to sort all values of the m variables, plus time required to test $(n-1) \times m$ distinct cut points, at worst. Each test, in turn, requires the computation of two regression planes on the m independent variables, that is, twice the time of computing $(\boldsymbol{D_X}^T \boldsymbol{D_X})^{-1} \boldsymbol{D_X}^T \boldsymbol{D_Y}$, where $\boldsymbol{D_Y}$ is the $n(t)$-dimensional vector of values taken by the response variable in node t, while $\boldsymbol{D_X}$ is an $n(t) \times (m+1)$ matrix of observations plus a column with only 1s [DS82]. Solving this least-square problem by normal equations takes time $\boldsymbol{O}(n(t) \times (m+1)^2 + (m+1)^3)$. Since in general $n(t) > m$, the time complexity can be approximated to $\boldsymbol{O}(n(t) \times (m+1)^2)$. For at least one of the tests $n(t)$ is proportional to n, thus the choice of the first split takes time $\boldsymbol{O}(n \times (n-1) \times m \times (m+1)^2)$.

In SUPPORT and SECRET this inefficiency is solved by transforming a re-

[2]The complexity for discrete variables cannot be evaluated, since no specification is reported in the literature on the procedure that RETIS follows to select best subsets of attribute values.

gression problem into a classification problem. More precisely, Chaudhury and his colleagues [CHLY94] propose to label the residuals of a linear model at a node as either positive or negative, and to choose the best partition on the basis of computationally efficient statistical tests developed for classification tasks. The justification of this approach is that if a fitted model is unsatisfactory, the lack of fit would be reflected in the distributional pattern of the residuals. Dobra and Gehrke [DG02] observe that this justification is not theoretically founded, and propose to identify two normal multivariate distributions in the space $\boldsymbol{X} \times Y$, and then to classify observed data according to the probability of belonging to these two distributions. The best partition is selected by means of efficient techniques developed for decision trees. In the case of binary splits of continuous attributes, Torgo [Tor02] proposes a solution based on a recursive least squares algorithm, whose average complexity is quadratic in the number of different data points between two subsequent cut-points of the continuous variable.

In addition to high computational complexity, another problem may occur in RETIS, since the regression planes g_L and g_R involve all continuous variables. When some of the independent variables are related to each other, that is, they are (approximately) collinear, several problems may occur [DS82], such as indeterminacy of regression coefficients, unreliability of the estimates of the regression coefficients, and impossibility of evaluating the relative importance of the independent variables. Interestingly, problems due to collinearity do not show in the model's fit. The resulting model may have very small residuals, but the regression coefficients are actually poorly estimated. A treatment suggested for data that exhibit collinearity is that of deleting some of the variables from a fitted model. Therefore, variable subset selection is a desirable part of regression analysis.

Finally, RETIS, as well as many other TDIMT systems, is characterized by models at leaves that can take into account only local decisions. This depends on building multiple regression model associated with a leaf on the basis of those training cases falling in the corresponding partition of the feature space. Therefore, models in the leaves have only a *local* validity and do not consider the *global* effects that some variables might have in the underlying model functions. To explain this concept, let us consider the case of a region R of a feature space described by four continuous independent variables X_1, X_2, X_3, and X_4. The region R can be partitioned into two regions R_1 and R_2, such that cases with $X_4 \leq \alpha$ ($X_4 > \alpha$), for a constant threshold α, fall in R_1 (R_2). Two regression models can be built independently for each region R_i, say:

$$R_1 : \hat{Y} = \widehat{\beta'}_0 + \widehat{\beta'}_1 X_1 + \widehat{\beta'}_2 X_2 \tag{4.1}$$

$$R_2 : \hat{Y} = \widehat{\beta''}_0 + \widehat{\beta''}_1 X_1 + \widehat{\beta''}_3 X_3. \tag{4.2}$$

The presence of X_1 in both models simply indicates that this variable is relevant both when $X_4 \leq \alpha$ and when $X_4 > \alpha$, although its influence on the dependent variable Y can be very different for the two regions. In this case, we say that the effect of X_1 on Y is *local*, since it can be properly predicted by considering the test $X_4 \leq \alpha$. A *global* effect occurs when the contribution of X_1 to Y can be reliably

predicted on the whole region R. In this case, an initial regression model can be approximated by regressing on X_1 for the whole region R:

$$R : \hat{Y} = \widehat{\beta}_0 + \widehat{\beta}_1 X_1 \tag{4.3}$$

and then by adding the effect of the variables X_2 and X_3 locally to the subregions R_1 and R_2, respectively. This is a case of stepwise construction of regression models. As explained in [DS82], the correct procedure to follow in order to introduce the effect of another variable in the partially constructed regression model is to eliminate the effect of X_1. In practice, this means that we have to compute the following regression models for the whole region R:

$$R : \hat{X}_2 = \widehat{\beta}_{20} + \widehat{\beta}_{21} X_1 \tag{4.4}$$

$$R : \hat{X}_3 = \widehat{\beta}_{30} + \widehat{\beta}_{31} X_1 \tag{4.5}$$

and then to compute the residuals $X'_2 = X_2 - \hat{X}_2$, $X'_3 = X_3 - \hat{X}_3$ and $Y' = Y - \hat{Y} = Y - (\beta_0 + \beta_1 X_1)$. By regressing the residuals Y' on X'_2 and X'_3 for the regions R_1 and R_2, respectively, the following two models are built:

$$R_1 : \hat{Y} = \widehat{\beta}_0 + \widehat{\beta}_1 X_1 + \widehat{\beta}_2 X'_2 = \widehat{\beta}_0 + \widehat{\beta}_1 X_1 + \widehat{\beta}_2 X_2 - \widehat{\beta}_2 \widehat{\beta}_{20} - \widehat{\beta}_2 \widehat{\beta}_{21} X_1 \tag{4.6}$$

$$R_2 : \hat{Y} = \widehat{\beta}_0 + \widehat{\beta}_1 X_1 + \widehat{\beta}_3 X'_3 = \widehat{\beta}_0 + \widehat{\beta}_1 X_1 + \widehat{\beta}_3 X_3 - \widehat{\beta}_3 \widehat{\beta}_{30} - \widehat{\beta}_3 \widehat{\beta}_{31} X_1. \tag{4.7}$$

They show the *global* effect of X_1, since their first two common terms do not depend on the test $X_4 \leq \alpha$, that is, they are computed on the whole region R. Moreover, the last term of each model corrects the contribution of X_1 due to the *local* introduction of either X_2 or X_3.

In model trees, global effects can be represented by variables that are introduced in the multiple models at higher levels of the tree [MACM02]. This requires a different tree-structure that supports the stepwise construction of multiple linear models by intermixing regression steps with partitioning steps, as in TSIR. TSIR has two types of node: splitting nodes and regression nodes. A splitting node performs a Boolean test on a variable and has two children. A regression node computes a single variable regression, $\hat{Y} = a + bX$, and passes down to its *unique* child the residuals $y_i - (a + bx_i)$ as new values of the target (response) variable. Thus, descendants of a regression node will operate on a modified training set. Lubinsky claims that "each leaf of the TSIR tree corresponds with a different multiple linear regression", and that "each regression step adds one variable and its coefficients to an incrementally growing model". However, this interpretation is not correct from a statistical point of view, since the incremental construction of a multiple linear regression model must be made *by removing the linear effect of the introduced variables each time a new independent variable is added to the model*[3] [DS82]. Conversely, TSIR passes down the residuals of Y alone. Therefore, it is not possible to assert that the composition of straight-line models found along a path from the root to a leaf

[3] See Example 3.2 for the correct stepwise construction of multiple linear regression model through a sequence of straight-line regressions.

is equivalent to a multiple linear model associated with the leaf itself. Moreover, collinearity problems are not properly solved, although only a subset of variables may be involved in the models at leaves.

A summary of some characteristics discussed above is reported in Table 4.1. It is noteworthy that all systems can build multiple linear regression models at leaves. In addition, HTL can build kernel regressors, which are simply to implement but do not capture the structure of the domain as linear models do. Some systems involve both continuous and discrete variables in the linear models at leaves. The latter are treated as dichotomous variables in standard linear regression [DS82], however, their real contribution is unclear in the case of model trees. In some systems, linear models at leaves can be retrospectively simplified by deleting some variables.

TABLE 4.1: Systems comparison

System	**Coherent evaluation function**	**Local global effects**	**Variables in the nodes at leaves**	**Type of model**	**Simplification of models at leaves**	**Tree pruning**
M5	No	No	Discrete and continuous variables used in split nodes	Linear models	Yes	Yes
M5'	No	No	Discrete and continuous variables used in split nodes	Linear model	Yes	Yes
HTL	No	No	Continuous variables	Linear model or kernel regressor or hybrid	Yes	Yes
RETIS	Yes	No	All continuous variables	Linear model	No	Yes
GUIDE	No	No	Continuous variables	Linear model	Yes	Yes
SECRET	Yes	No	Continuous variables	Linear model	No	Yes
TSIR	Only for continuous variables	Yes	Continuous variables	Linear model (not statistically interpretable)	No	No
SMOTI	Yes	Yes	Continuous variables	Linear model	No	Yes

In the remaining of this chapter, the TDIMT method SMOTI is presented. It has four distinguishing features:

1. A selection measure is chosen which is coherent with respect to the (partial) linear model associated with the leaves.

2. Multiple regression models are constructed stepwise, by intermixing both regression and splitting nodes. Problems observed in TSIR are solved by removing the effect of the variable selected in a regression node before passing down training cases to deeper levels, and by adopting a look-ahead strategy when regression nodes and splitting nodes are compared for selection.

3. The multiple linear model associated with each leaf involves all the continuous variables in the regression nodes and the continuous variable in the straight-line regression performed at the leaf. In this way, both global and local effects of variables are considered.

4. The simplification strategies apply only to the tree structure, thus the deletion of some variables in the models at leaves is the result of pruning a regression node.

4.2 Stepwise construction of model trees

SMOTI induces model trees with both regression and splitting nodes. In this Section, we start with the formal definition of a SMOTI tree, that is a model tree with regression and splitting nodes, then we describe details about the stepwise construction of model trees performed by SMOTI and analyze its computational complexity. Finally, some practical considerations to improve SMOTI efficiency are discussed.

4.2.1 SMOTI tree

A SMOTI tree can be formally defined as follows:

Definition 4.1 *Given a set D of n training regression cases, each of which is described by both m (continuous and discrete) predictors $X_1, \ldots, X_m$ and a target continuous variable Y, it is possible to build a tree-structured model named SMOTI tree. This is a particular rooted tree $\tau = (N_\tau, B_\tau)$ in which:*

1. *each node $t_i \in N_\tau$ is associated with a subset of D, denoted by $D(t_i)$, possibly modified by removing the linear effect of the variables added to the model,*
2. *the root t_0 is associated with D itself,*
3. *each leaf node t_i is labeled with a straight-line regression $Y = \beta_0 + \beta_1 X_k$,*
4. *each edge $\langle t_i, t_j \rangle \in B_\tau$ is labeled with $L_\tau(\langle t_i, t_j \rangle)$.*

$LT(\langle t_i, t_j \rangle)$ can be:

- *a straight-line regression function $Y = \beta_0 + \beta_1 X_k$ (regression label),*
- *a test in the form $X_i \leq \alpha$ ($X_i > \alpha$), which involves a continuous predictor variable X_i (continuous splitting label),*
- *a test in the form $X_i \in \{x_{i_1}, \ldots, x_{i_s}\}$($X_i \notin \{x_{i_1}, \ldots, x_{i_s}\}$), which involves a discrete predictor variable X_i(discrete splitting label),*

♦

An internal node $t_i \in N_\tau^I$ is called regression (splitting) node iff there exists an edge $\langle t_i, t_j \rangle \in B_\tau$ such that $LT(\langle t_i, t_j \rangle)$ is a regression (continuous/discrete splitting) label. The set of regression (splitting) nodes is denoted by $N_\tau^{I_R}$ ($N_\tau^{I_S}$), while the set of leaves is denoted by $\in N_\tau^L$.

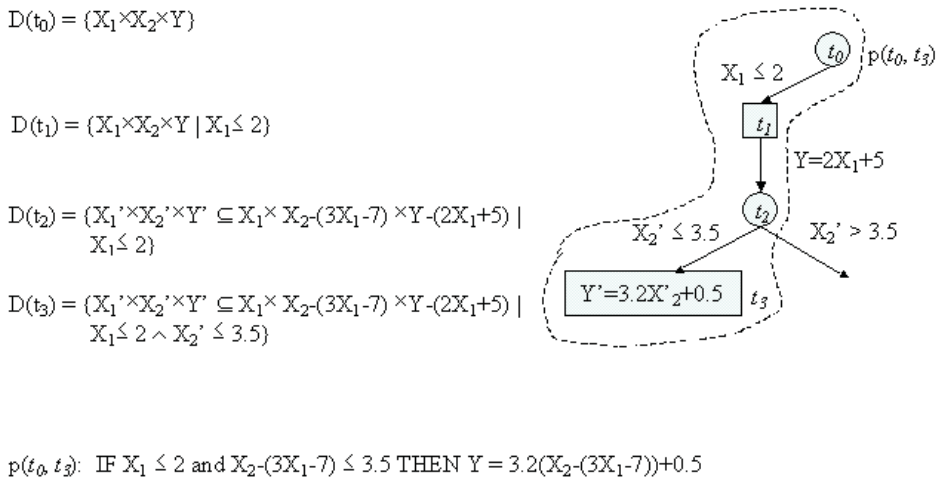


FIGURE 4.2: An example of model tree by intermixing regression steps and splitting test, removing residuals of variables introduced in the model and building a multiple linear function at each leaf by combining straight-line regressions.

4.2.2 The algorithm

In SMOTI [MECA04], the development of a tree structure is not only determined by a recursive partitioning procedure, but also by some intermediate prediction functions. This means that there are two types of nodes in the tree: splitting nodes and regression nodes. They pass down observations to their children in two different ways. For a splitting node t, only a subgroup of the $n(t)$ observations in t is passed down to each child, and no change is made on the variables. For a regression node t, all the observations are passed down to its only child, but both the values of the dependent variable and the values of the (continuous) independent variables not included in the model are transformed to remove the linear effect of those variables already included. This is done coherently with the statistical theory for the incremental construction of a multiple linear regression model, as explained in Example 3.2. Thus, descendants of a regression node will operate on a modified dataset. Consequently, the regression model effectively built at each leaf is a multiple function obtained by combining all straight-line regressions along the path from the root to the current leaf with straight-line regression labeling the leaf (see Figure 4.2). An extra consequence is that all candidate continuous splits in the subtree rooted into a regression node are effectively *oblique splits*, since they use oblique hyperplane to partition the data (see Figure 4.3). More precisely, a splitting test involving the residual of a continuous variable is a test on a linear combination of multiple continuous variables [Iye99].

The top-level description of SMOTI is presented in Algorithm 4.1. It points out that the validity of either a a splitting test or a regression step on a variable X_i is based on two distinct evaluation measures, $\sigma(t)$ and $\rho(t)$ respectively. The variable X_i is of any type in the former case, and of a continuous type in the latter

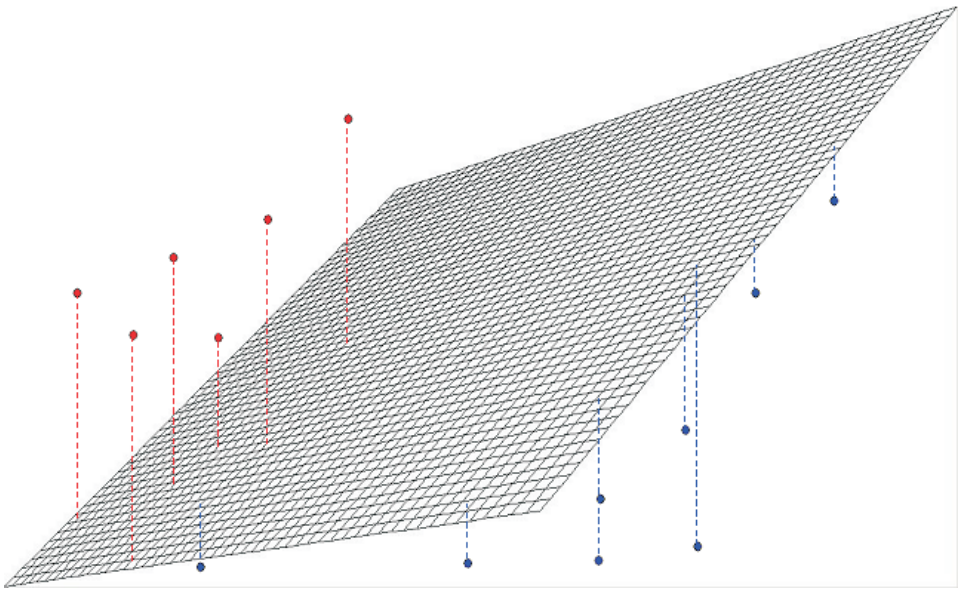


FIGURE 4.3: An oblique hyperplane to partition data.

case. Both $\sigma(t)$ and $\rho(t)$ are mean square errors,[4] therefore they can be actually compared to choose between three different possibilities:

1. growing the model tree by adding a regression node t;
2. growing the model tree by adding a splitting node t;
3. stopping the tree's growth at node t.

Algorithm 4.1 SMOTI Algorithm

PROCEDURE build-SMOTI-tree(X, Y, R, L, τ)
INPUT:
X: a set of m independent variables X_i;
Y: the dependent variable;
R: a subset of continuous variables in X;
L: a training set $\{(\mathbf{x_j}, y_j)|j = 1, \ldots, n\}$, where $\mathbf{x_j} = (x_{1_j}, \ldots, x_{m_j})$;
OUTPUT:
τ: a model tree with regression and splitting nodes built on $(\mathbf{X}, Y)$;
BEGIN
$Best - \rho = \infty$;
$Best - t_r =$ a node whose model is the estimated mean $\overline{Y}$;
FOR each continuous variable $X_i \in R$ DO
Compute the best regression node t_r with variable X_i of R ;

[4] This is different from TSIR, which, in the case of node selection, minimizes the absolute deviation between a *constant* value (the median) and the observed values Y. On the contrary, SMOTI coherently minimizes the square error with respect to the partially constructed regression model at each node.

```
      Compute the evaluation measure ρ(t_r) for t_r;
      IF ρ(t_r) ≤ Best − ρ THEN Best − ρ = ρ(t_r); Best − t_r = t_r;
      END IF;
   END FOR;
   IF stopping criteria THEN τ = leaf(Best − t_r);
   ELSE
      Best − σ = ∞; Best − t_s = nil;
      FOR each variable X_i ∈ X DO
         Compute the best splitting node t_s with variable X_i;
         Compute the evaluation measure σ(t_s) for t_s;
         IF σ(t_s) ≤ Best − σ THEN
            Best − σ = σ(t_s); Best − t_s = t_s;
         END IF;
      END FOR
      IF Best − σ > Best − ρ THEN
         build-SMOTI-tree(X, Y, R, {(x_j, y_j) ∈ L|test in Best − t_s
          is true}, τ_L);
         build-SMOTI-tree(X, Y, R, {(x_j, y_j) ∈ L|test in Best − t_s
          is false}, τ_R);
         τ = tree with root Best−t_s and left (right) branch τ_L (τ_R);
      ELSE
         Let X_r be the variable in Best − t_r;
         FOR EACH case (x_j, y_j) ∈ L DO
            FOR EACH continuous variable X_i ∈ X − X_r DO
               x'_{i_j} = residuals of x_{i_j} after removing the effect of
                X_r;
            END FOR;
            y'_j = residuals of y_j when the regression in Best-t_r is
             performed;
         END FOR
         build-SMOTI-tree(X'_j ∪ X_r, Y', X'_j, {(x'_j, y'_j)|(x_j, y_j) ∈ L}, τ');
         τ = tree with root in Best − t_r and child τ';
      END IF;
   END IF;
END PROCEDURE
```

The evaluation measure $\sigma(t)$ should be coherently defined on the basis of the multiple linear models at the leaves. In SMOTI, it is sufficient to consider the best straight-line regression associated with each left (right) leaf t_L (t_R), since regression nodes along the path from the root to t_L (t_R) already partially define a multiple regression model (see Figures 4.4.a and 4.4.b).

If X_i is continuous and α is a threshold value for X_i, then $\sigma(t)$, with t the split $X_i \leq \alpha$ vs. $X_i > \alpha$, is defined as:

$$\sigma(t) = \frac{n(t_L)}{n(t)} R(t_L) + \frac{n(t_R)}{n(t)} R(t_R), \tag{4.8}$$

where $n(t)$ is the number of cases reaching t, $n(t_L)$ ($n(t_R)$) is the number of cases passed down to the left (right) child $t_L(t_R)$, and $R(t_L)$ ($R(t_R)$) is the resubstitution error[5] of the left (right) child, computed as follows:

$$R(t_L) = \sqrt{\frac{1}{n(t_L)} \sum_{(\boldsymbol{x}_j, y_j) \in D(t_L)} (y_j - \hat{y}_j)^2} \tag{4.9}$$

$$R(t_R) = \sqrt{\frac{1}{n(t_R)} \sum_{(\boldsymbol{x}_j, y_j) \in D(t_R)} (y_j - \hat{y}_j)^2}. \tag{4.10}$$

The estimate:

$$\hat{y}_j = \widehat{\beta}_0 + \sum_s \widehat{\beta}_s x_s, \tag{4.11}$$

is computed by combining all univariate regression lines associated with regression nodes along the path from the root to t_L (t_R). Possible values of α are found by sorting the distinct values of X_i in the training set associated with t, then identifying a threshold (e.g. midpoint) between each pair of adjacent values. Therefore, if the cases in t have k distinct values for X_i, $k-1$ thresholds are considered. Obviously, the lower $\sigma(t)$, the better the split t: $X_i \leq \alpha$ vs. $X_i > \alpha$.

If X_i is discrete, SMOTI partitions attribute values into two sets, so that binary trees are always built. Some TDIMT systems, such as HTL and M5', use the same criterion applied in CART [BFOS84](pg. 247). More precisely, if k is the number of distinct values for X_i and $S_{X_i} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$ is the set of distinct values of X_i, S_{X_i} is sorted according to the sample mean of Y over all cases in t. A theorem by Breiman et al. [BFOS84](Theorem 4.5, Proposition 8.16) proves that the best binary split is one of $k-1$ partitions $\{x_{i_1}, \ldots, x_{i_h}\}$ and $S_{X_i} - \{x_{i_1}, \ldots, x_{i_h}\}$, thus greatly reducing the search for the best subset of categories from 2^{k-1} to k-1 partitions. However the theorem is based on the assumption that the models at leaves are the sample means, which is not the case of SMOTI[6]. Therefore, SMOTI relies on a non-optimal greedy strategy as suggested by [MAR96]. It starts with an empty set $\mathit{Left}_{X_i} = \phi$ and a full set $\mathit{Right}_{X_i} = S_{X_i}$. It moves one element from Right_{X_i} to Left_{X_i}, such that the move results in a better split. Let t be the split of $X_i \in \mathit{Left}_{X_i}$ vs. $X_i \notin \mathit{Left}_{X_i}$, then the evaluation measure $\sigma(t)$ is computed as in the case of continuous variables, and therefore, a better split decreases $\sigma(t)$. The process is iterated until there is no improvement in the splits. The computational complexity of this heuristic is $O(k^2)$. For all possible splits t, the measure $\sigma(t)$ is computed as in the case of continuous variables.

The split selection criterion explained above can be improved to consider the special case of identical regression models associated with both children. When this occurs, the best straight-line regression associated with t is the same as that associated with both t_L and t_R, up to some statistically insignificant difference. In other terms, the split is useless and can be filtered out from the set of alternatives.

[5] Resubstitution error is here computed as the root mean square error.

[6] Sample means are used only when all independent variables are discrete.

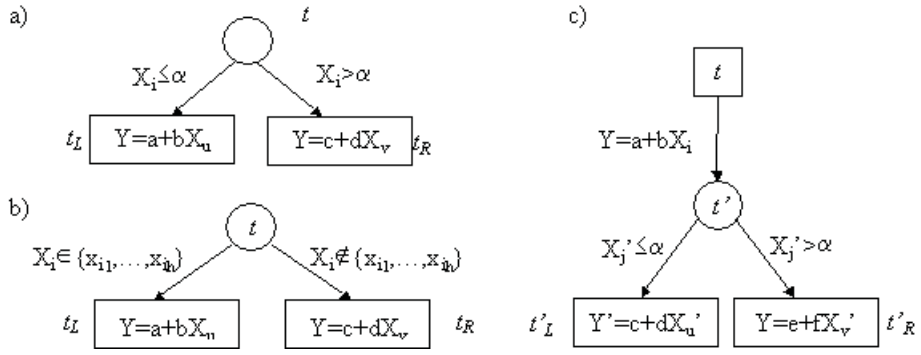


FIGURE 4.4: a) A continuous splitting node t with two straight-line regression models in the leaves. b) A discrete splitting node t with two straight-line regression models in the leaves. c) Evaluation of a regression step at node t, based on the best splitting test t' below.

To check this special case, SMOTI compares the two regression lines associated with the children according to a statistical test for coincident regression lines [Wei85](pp. 162-167).

The evaluation of a regression step $Y = \widehat{\beta}_0 + \widehat{\beta}_1 X_i$ at node t cannot be naïvely based on the resubstitution error $R(t)$:

$$R(t) = \sqrt{\frac{1}{n(t)} \sum_{(\boldsymbol{x}_j, y_j) \in D(t)} (y_j - \hat{y_j})^2}, \tag{4.12}$$

where the estimator $\hat{y}_j$ is computed by combining all univariate regression lines associated with regression nodes along the path from the root to t. This would result in values of $\rho(t)$ less than or equal to values of $\sigma(s)$ for some splitting test s involving X_i. Indeed, the splitting test *looks-ahead* to the best multiple linear regressions after the split on X_i is performed, while the regression step does not. A fairer comparison would be growing the tree at a further level in order to base the computation of $\rho(t)$ on the best multiple linear regressions after the regression step on X_i is performed (see Figure 4.4.c).

Let t' be the child of the regression node t, and let us suppose that it performs a splitting test. The best splitting test in t' can be chosen on the basis of $\sigma(t')$ for all possible variables X_j, as indicated above. Then $\rho(t)$ can be defined as follows:

$$\rho(t) = \min\{(R(t), \sigma(t')\}. \tag{4.13}$$

The possibility of statistically identical regression models associated with the children of t' may also occur in this case. When this happens, the splitting node is replaced by another regression node t' where the straight-line regression model is the same as that in the children of the splitting node. Therefore, in this special case, $\rho(t)$ can be defined as follows:

$$\rho(t) = \min\{R(t), R(t')\}. \tag{4.14}$$

Having defined both $\rho(t)$ and $\sigma(t)$, the criterion for selecting the best node is fully characterized as well. At each step of the model tree induction process,

SMOTI chooses the apparently most promising node, according to a greedy strategy. A continuous variable selected for a regression step is no longer considered for regression purposes, so that it can appear only once in a regression node along a path from the root to a leaf.

In SMOTI, five different stopping criteria are implemented. The first uses the partial F-test to evaluate the contribution of a new independent variable to the model [DS82]. The second requires the number of cases in each node to be greater than a minimum value. The third stops the induction process when all continuous variables along the path from the root to the current node are used in regression steps and there are no discrete variables in the training set. The fourth creates a leaf if the error in the current node is below a fraction of the error in the root node, as in [Tor99](pg. 60). Finally, the fifth stops the induction process when the coefficient of determination (R^2) is greater than a minimum value [Wei85](pp. 18-19). This coefficient is a scale-free one-number summary of the strength of the relationship between independent variables in the actual multiple model and the response variable. It is computed as follows:

$$R^2 = \frac{\sum\limits_{(\boldsymbol{x}_j, y_j) \in D(t)} (\widehat{y}_j - \bar{Y}(t))^2}{\sum\limits_{(\boldsymbol{x}_j, y_j) \in D(t)} (y_j - \bar{Y}(t))^2} \tag{4.15}$$

where $\widehat{y}_j = \tau(\boldsymbol{x}_j)$ when t is converted into a leaf node of τ, while $\bar{Y}(t) = \frac{\sum\limits_{(\boldsymbol{x}_j, y_j) \in D(t)} y_j}{n(t)}$.

In conclusion, SMOTI presents several advantages. First, it defines the best partitioning of the feature space coherently with respect to the model tree being built. Second, it provides a solution to the problems of collinearity at the same computational cost of RETIS. Third, the use of both regression and splitting nodes permits the system to discover both global and local effects of variables in the various regression models. This is evident in the experimental results.

4.2.3 Complexity analysis

The computational complexity of adding a splitting node t into SMOTI tree depends on the complexity of a splitting test selection in t and the complexity of the best regression step selection in the children nodes t_L and t_R. On the contrary, the computational complexity of adding a regression node t depends on the complexity of a regression step selection in t and the complexity of the best splitting test in its child t'.

A splitting test can be either continuous or discrete. In the former case, a threshold α has to be selected for a continuous variable. Let n be the number of examples in the training set, then the number of distinct thresholds can be $n-1$ at worst. They can be determined after sorting the set of distinct values. If m is the number of independent variables, the determination of all possible thresholds has a complexity $\boldsymbol{O}(m \times n \times \lg n)$ when an optimal algorithm is used to sort the values. For each of the $m \times (n-1)$ thresholds, SMOTI finds the best straight-line regression at both children, which has a complexity of $m \times (n-1)$ in the worst case. Therefore, the splitting test has a complexity $\boldsymbol{O}(m \times n \times \lg n + m^2 \times (n-1)^2)$, that is $\boldsymbol{O}(m^2 \times n^2)$.

Similarly, for a discrete splitting test, the worst case complexity is $\boldsymbol{O}(m \times k^2)$ where k is the maximum number of distinct values of a discrete variable. The selection of the best discrete splitting test has a complexity $\boldsymbol{O}(m^2 \times k^2 \times n)$. Therefore, finding the best splitting node (either continuous or discrete) has a complexity $\boldsymbol{O}(m^2 \times n^2 + m^2 \times k^2 \times n)$, and under the reasonable assumption that $k^2 \leq n$, that is, the number of distinct values of the a discrete variable is less then the square root of the number of cases, the worst case complexity is $\boldsymbol{O}(m^2 \times n^2)$.

The selection of the best regression step requires the computation, for each of the m variables, of m straight-line regressions (one for the regression node plus $m-1$ to remove the effect of the regressed variable) and the updating of the dataset. This takes time $\boldsymbol{O}(m^2 \times n))$, since the complexity of the computation of a straight-line regression is linear in n. Moreover, for each straight-line regression, a splitting test is required, which has a worst case complexity of $\boldsymbol{O}(m^2 \times n^2)$. Therefore, the selection of the best regression step has a complexity $\boldsymbol{O}(m^2 \times n + m^3 \times n^2)$, that is, $\boldsymbol{O}(m^3 \times n^2)$.

The above results lead to an $\boldsymbol{O}(m^3 \times n^2)$ worst case complexity for the selection of any node (splitting or regression). This means that, relative to node selection, SMOTI has the same complexity as RETIS but is less efficient than TSIR which adopts a v-fold cross-validation strategy without look-ahead for regression and splitting nodes. In TSIR the complexity is $\boldsymbol{O}(m \times v \times n)$ for regression nodes and $\boldsymbol{O}(m \times v \times n^2)$ for splitting nodes. However, the model that TSIR considers at the children of a discrete splitting node during its evaluation is the sample mean and not a linear regression, which means that it suffers from the problems of adopting a heuristic evaluation function which is not coherent with the models associated with the leaves.

4.2.4 Some practical considerations

The analysis of SMOTI computational complexity indicates that the key computational issue is evaluating a candidate split or regression.

The selection of the best split on a continuous variable X_a for a node t starts with ordering the training threshold values for X_a. After obtaining this ordering (say $\{\alpha_1, \ldots, \alpha_k\}$), SMOTI has to evaluate each possible test that leads to a different splitting on the cases through the left and right branches. In particular, when the candidate split $X_a \leq \alpha_h$ is evaluated: for each continuous variable X_i (or residual X_i') not yet included in the regression model stepwise built, two straight-line regression are obtained: one fitting with the cases satisfying this test (let slope and intercept be $\widehat{\beta}_{0_L}(\alpha_h, X_i)$ and $\widehat{\beta}_{1_L}(\alpha_h, X_i)$, respectively) and the other with ones not satisfying this test ($\widehat{\beta}_{0_R}(\alpha_h, X_i)$ and $\widehat{\beta}_{1_R}(\alpha_h, X_i)$). For each side, the best straight-line regression (i.e. straight-line regression which minimizes the corresponding resubstitution error) is returned and then used in evaluating the test according to $\sigma(t)$. When the candidate split $X_a \leq \alpha_{h+1}$ where $\alpha_{h+1} > \alpha_h$ is then evaluated, this corresponds with moving some cases from the right to the left branch and computing new values for each $\widehat{\beta}_{0_L}(\alpha_{h+1}, X_i)$ and $\widehat{\beta}_{1_L}(\alpha_{h+1}, X_i)$ as well as $\widehat{\beta}_{0_R}(\alpha_{h+1}, X_i)$ and $\widehat{\beta}_{1_R}(\alpha_{h+1}, X_i)$. Regression coefficients

$\widehat{\beta}_{0_L}(\alpha_{h+1}, X_i)$ and $\widehat{\beta}_{1_L}(\alpha_{h+1}, X_i)$ are estimated with the cases used to estimate $\widehat{\beta}_{0_L}(\alpha_h, X_i)$ and $\widehat{\beta}_{1_L}(\alpha_h, X_i)$ plus some other cases, namely the cases defined by the set $J(\alpha_h, \alpha_{h+1}) = \{(\boldsymbol{x}_j, y_j) \in D(t) | x_{a_j} > \alpha_h \wedge x_{a_j} \leq \alpha_{h+1}\}$. Similarly, regression coefficients $\widehat{\beta}_{0_R}(\alpha_{h+1}, X_i)$ and $\widehat{\beta}_{1_R}(\alpha_{h+1}, X_i)$ are estimated according to cases used to estimate $\widehat{\beta}_{0_R}(\alpha_h, X_i)$ and $\widehat{\beta}_{1_R}(\alpha_h, X_i)$ less the set of J observed cases.

Following a suggestion given by Torgo [Tor02], SMOTI bases the estimation of each regression coefficient $\widehat{\beta}_{i_L}(\alpha_{h+1}, X_i)$ ($\widehat{\beta}_{i_R}(\alpha_{h+1}, X_i)$) on the corresponding $\widehat{\beta}_{i_L}(\alpha_h, X_i)$ ($\widehat{\beta}_{i_R}(\alpha_h, X_i)$).

Let us consider the straight-line regression between Y and the continuous (residual) variable X_i, namely $Y = \widehat{\beta}_{0_L}(\alpha_h, X_i) + \widehat{\beta}_{1_L}(\alpha_h, X_i)X_i$, fitting with $n_L(\alpha_h)$ cases observed in t, which satisfy the test $X_a \leq \alpha_h$. The intercept $\widehat{\beta}_{1_L}(\alpha_h, X_i)$ is:

$$\widehat{\beta}_{1_L}(\alpha_h, X_i) = \frac{\sum_{j=1\ldots n_L(\alpha_h)} (x_{i_j} - \overline{X_i})(y_j - \overline{Y})}{\sum_{j=1\ldots n_L(\alpha_h)} (x_{i_j} - \overline{X_i})^2} \tag{4.16}$$

$$= \frac{\sum_{j=1\ldots n_L(\alpha_h)} (x_{i_j} y_j - \overline{X_i} y_j - x_{i_j}\overline{Y} + \overline{X_i Y})}{\sum_{j=1\ldots n_L(\alpha_h)} (x_{i_j}^2 + \overline{X_i}^2 - 2x_{i_j}\overline{X_i})} \tag{4.17}$$

$$= \frac{\sum_{j=1\ldots n_L(\alpha_h)} x_{i_j} y_j - \overline{X_i} \sum_{j=1\ldots n_L(\alpha_h)} y_j - \overline{Y} \sum_{j=1\ldots n_L(\alpha_h)} x_{i_j} + n_L \overline{X_i Y}}{\sum_{j=1\ldots n_L(\alpha_h)} x_{i_j}^2 + n_L(\alpha_h)\overline{X_i}^2 - 2\overline{X_i} \sum_{j=1\ldots n_L(\alpha_h)} x_{i_j}} \tag{4.18}$$

$$= \frac{\sum_{j=1\ldots n_L(\alpha_h)} x_{i_j} y_j - \frac{1}{n_L(\alpha_h)} \sum_{j=1\ldots n_L(\alpha_h)} x_{i_j} \sum_{j=1\ldots n_L(\alpha_h)} y_j}{\sum_{j=1\ldots n_L(\alpha_h)} x_{i_j}^2 - \frac{1}{n_L(\alpha_h}(\sum_{j=1\ldots n_L(\alpha_h)} x_{i_j})^2}, \tag{4.19}$$

while the slope $\widehat{\beta}_{0_L}(\alpha_h, X_i)$ is:

$$\widehat{\beta}_{0_L}(\alpha_h, X_i) = \overline{Y} - \widehat{\beta}_{0_L}(\alpha_h, X_i)\overline{X_i}, \tag{4.20}$$

where $\overline{X_i} = \frac{1}{n_L(\alpha_h)} \sum_{1\ldots n_L(\alpha_h)} x_{i_j}$ and $\overline{Y} = \frac{1}{n_L(v_h)} \sum_{1\ldots n_L(\alpha_h)} y_i$.

By denoting with $S_L(X_i, Y, \alpha_h) = \sum_{j=1\ldots n_L(\alpha_h)} x_{i_j} y_j$; $S_L(X_i, \alpha_h) = \sum_{j=1\ldots n_L(\alpha_h)} x_{i_j}$; $S_L(Y, \alpha_h) = \sum_{j=1\ldots n_L(\alpha_h)} y_j$ and $S_L(X_i, X_i, \alpha_h) = \sum_{j=1\ldots n_L(\alpha_h)} x_{i_j}^2$, then $\widehat{\beta}_{1_L}(\alpha_h, X_i)$ and $\widehat{\beta}_{0_L}(\alpha_h, X_i)$ can be written as:

$$\widehat{\beta}_{1_L}(\alpha_h, X_i) = \frac{S_L(X_i, Y, \alpha_h) - \frac{1}{n_L(\alpha_h)} S_L(X_i, \alpha_h) S_L(Y, \alpha_h)}{S_L(X_i, X_i, \alpha_h) - \frac{1}{n_L(\alpha_h)} S_L(X_i, \alpha_h)^2} \tag{4.21}$$

$$\widehat{\beta}_{0_L}(\alpha_h, X_i) = \frac{S_L(Y, \alpha_h)}{n_L(\alpha_h)} - \widehat{\beta}_{0_L} \frac{S_L(X_i, \alpha_h)}{n_L(\alpha_h)}. \tag{4.22}$$

Both $\widehat{\beta}_{0_L}(\alpha_{h+1}, X_i)$ and $\widehat{\beta}_{1_L}(\alpha_{h+1}, X_i)$ can be computed starting from $S_{X_i Y}(\alpha_h)$, $S_{X_i}(\alpha_h)$, $S_Y(\alpha_h)$ and $S_{X_i Y}(\alpha_h)$ and updating them on the basis of only cases ob-

served in $J(\alpha_h, \alpha_{h+1})$:

$$S_L(X_i, Y, \alpha_{h+1}) = S_L(X_i, Y, \alpha_h) + \sum_{(\boldsymbol{x}_j, y_j) \in J(\alpha_h, \alpha_{h+1})} x_{i_j} y_j, \tag{4.23}$$

$$S_L(X_i, \alpha_{h+1}) = S_L(X_i, \alpha_k) + \sum_{(\boldsymbol{x}_{i_j}, y_j) \in J(\alpha_h, \alpha_{h+1})} x_{i_j}, \tag{4.24}$$

$$S_L(Y, \alpha_{h+1}) = S_L(Y, \alpha_h) + \sum_{(\boldsymbol{x}_{i_j}, y_j) \in J(\alpha_h, \alpha_{h+1})} y_j, \tag{4.25}$$

$$S_L(X_i, X_i, \alpha_{h+1}) = S_L(X_i, X_i, \alpha_h) + \sum_{(\boldsymbol{x}_{i_j}, y_j) \in J(\alpha_h, \alpha_{h+1})} x_{i_j}^2, \tag{4.26}$$

such that:

$$\widehat{\beta}_{1_L}(\alpha_{h+1}, X_i) = \frac{S_L(X_i, Y, \alpha_{h+1}) - \frac{1}{n_L(\alpha_{h+1})} S_L(X_i, Y, \alpha_{h+1})}{S_L(X_i, X_i, \alpha_{h+1}) - \frac{1}{n_L(\alpha_{h+1})} S_L(X_i, \alpha_{h+1})^2}, \tag{4.27}$$

$$\widehat{\beta}_{0_L}(\alpha_{h+1}, X_i) = \frac{S_L(Y, \alpha_{h+1})}{n_L(\alpha_{h+1})} - \widehat{\beta}_{0_L} \frac{S_L(X_i, \alpha_{h+1})}{n_L(\alpha_{h+1})}. \tag{4.28}$$

Similarly, on right side, we denote by $Y = \widehat{\beta}_{0_R}(\alpha_h, X_i) + \widehat{\beta}_{1_R}(\alpha_h, X_i)$ the straight-line regression between Y and X_i fitting with $n_R(\alpha_h)$ cases observed in t and satisfying the test $X_a > \alpha_h$. Starting from $S_R(X_i, Y, \alpha_h)$, $S_R(X_i, \alpha_h)$, $S_R(Y, \alpha_h)$ and $S_R(X_i, X_i, \alpha_h)$, SMOTI computes:

$$S_R(X_i, Y, \alpha_{h+1}) = S_R(X_i, Y, \alpha_h) - \sum_{(\boldsymbol{x}_j, y_j) \in J(\alpha_h, \alpha_{h+1})} x_{i_j} y_j, \tag{4.29}$$

$$S_R(X_i, \alpha_{h+1}) = S_R(X_i, \alpha_h) - \sum_{(\boldsymbol{x}_{i_j}, y_j) \in J(\alpha_h, \alpha_{h+1})} x_{i_j}, \tag{4.30}$$

$$S_R(Y, \alpha_{h+1}) = S_R(Y, \alpha_k) - \sum_{(\boldsymbol{x}_{i_j}, y_j) \in J(\alpha_h, \alpha_{h+1})} y_j, \tag{4.31}$$

$$S_R(X_i, X_i, \alpha_{h+1}) = S_R(X_i, X_i, \alpha_h) - \sum_{(\boldsymbol{x}_{i_j}, y_j) \in J(\alpha_h, \alpha_{h+1})} x_{i_j}^2, \tag{4.32}$$

and obtains:

$$\widehat{\beta}_{1_R}(\alpha_{h+1}, X_i) = \frac{S_R(X_i, Y, \alpha_{h+1}) - \frac{1}{n_R(\alpha_{h+1})} S_R(X_i, Y, \alpha_{h+1})}{S_R(X_i, X_i, \alpha_{h+1}) - \frac{1}{n_R(\alpha_{h+1})} S_R(X_i, \alpha_{h+1})^2}, \tag{4.33}$$

$$\widehat{\beta}_{0_R}(\alpha_{h+1}, X_i) = \frac{S_R(Y, \alpha_{h+1})}{n_R(\alpha_{h+1})} - \widehat{\beta}_{0_R} \frac{S_R(X_i, \alpha_{h+1})}{n_R(v\alpha_{h+1})}. \tag{4.34}$$

Similar considerations can be adopted in selecting of best splitting test on a discrete variable X_a having domain S_{X_a}. SMOTI, starts (step 0) with an empty set $\mathit{Left}_{X_a}(0) = \phi$ and a full set $\mathit{Right}_{X_a}(0) = S_{X_a}$. It moves one element from Right_{X_a} to Left_{X_a}, such that the move results in a better split. This means that at step h, SMOTI is determining the best split having h values in $\mathit{Left}_{X_a}(h)$ such that $\mathit{Left}_{X_a}(h) = \{\alpha_1, \ldots, \alpha_h\}$ and $\mathit{Right}_{X_i}(h) = S_{X_a} - \{\alpha_1, \ldots, \alpha_h\}$. Then, at step $h+1$, SMOTI is evaluating all possible splits by moving one element from $\mathit{Right}_{X_i}(h)$ to $\mathit{Left}_{X_i}(h)$. This means that, for each candidate split, we define $J(\mathit{Left}_{X_a}(h), \mathit{Left}_{X_a}(h+1)) = \{(\boldsymbol{x}_j, y_j) \in D(t) | x_{a_j} \in \mathit{Left}_{X_i}(h) \cup \{v_{h+1}\}$ with $v_{h+1} \in \mathit{Right}_{X_i}(h)\}$. Both intercept and slope of candidate straight-line regressions on left

and right side of a candidate split at step $h+1$ are then computed as in the case of continuous split by updating the partial sums obtained for the best split at step h.

Finally, when SMOTI is evaluating a regression step on a continuous (residual) variable X_a, this corresponds with *looking-ahead* to the best split after the regression is performed. This suggests that when, SMOTI is introducing a regression node t in τ, it may also store the best split t' obtained in the look-ahead evaluation of t. Therefore, when SMOTI procedure is recursively applied to $D(t)$, the best candidate split is properly t', that must not be recomputed it again by improving efficiency.

4.3 Overfitting avoidance in SMOTI tree

Similarly to other TDIMT methods, SMOTI may generate model trees, which overfit training data. This means that the tree is capturing regularities of the training sample and not of the domain from which the sample is obtained.

Overfitting avoidance within tree models is usually achieved by *growing* an overly large tree and then *simplifying* the tree by discarding parts of the tree that describe spurious effects in the training sample rather than true features of the underlying phenomenon. However, as Schaffer [Sch93] pointed out, simplifying a tree cannot be only regarded as a statistical means to improve the true prediction accuracy. At this aim, Torgo [Tor99] enlightens that it is easy to find real world domains where simplification is actually harmful with respect to predictive accuracy on independent and large test samples. On the contrary, as suggested by Schaffer [Sch93], simplification should be regarded as a preference bias over *simpler* models. In this sense, simplification is coherent with Occam's razor principle[7] which is generally interpreted as counseling the use of *simpler* models rather than complex ones. Domingo [Dom99] underlines that this is the only useful interpretation of Occam's razor as a universal principle (as opposed to a domain-specific heuristic) and argues that simpler models should be preferred since they are more comprehensible to people. Nevertheless, simplicity is only an imperfect measure of comprehensibility, and precisely quantifying the latter does not seem possible.

In model trees, simplicity can be eventually quantified in terms of complexity of either the tree structure (e.g. number of leaves, depth of the tree, etc.) or the model at leaves (e.g. number of variables included in the model). In this perspective, simplification is regarded as a search problem aiming at looking for the *best* simplified tree. The final effect is that the intelligibility of the regression model is improved, without really affecting its predictive accuracy.

Simplification methods are computationally inefficient in the sense that it is not unusual to find domains where an extremely large tree with thousands of nodes is simplified into few hundred nodes. This clearly seems a waste of computation. An alternative consists of stopping the tree growth procedure as soon as further splitting is considered unreliable. However, this method incurs the danger of selecting a suboptimal tree [BFOS84] by stopping too soon.

[7] The Occam's razor principle "*entities should not be multiplied more than necessary*" (entia non sunt multiplicanda praeter necessitatem) has been attributed the medieval English philosopher and Franciscan monk William of Ockham (1285-1349).

Several simplification methods have been reported in literature to simplify model trees whose internal nodes are only splitting tests. On the other hand, no simplification method has been proposed in TSIR, the only other system that induces trees with two types of nodes. In this Section, we describe two methods, named Reduced Error Pruning (REP) and Reduced Error Grafting (REG), to adequately simplify model trees with regression nodes and splitting nodes. They are both based on the use of an independent pruning set, but they adopt two distinct simplification operators, namely: pruning operator and grafting operator. Some theoretical properties of these simplification methods are also reported.

4.3.1 A framework for SMOTI simplification methods

Simplification (pruning) methods have been initially proposed to solve the overfitting problem of induced decision trees. A unifying framework for their descriptions is reported in [EMS97]. We have followed the same idea and developed two methods, namely Reduced Error Pruning and Reduced Error grafting, to simplify model trees with both regression nodes and splitting nodes [CAM03a]. The first method uses the classical pruning operator extended to regression nodes as well. The second method is based on a new grafting operator that replaces a splitting node with a subtree.

To formally define both these simplification methods, some notations are introduced. We start with the definition of pruning and grafting relations on SMOTI trees, then we define the search spaces, and finally the operators.

Let T denote the set of all possible model trees that SMOTI can build from a training set D. It is possible to define two distinct partial order relations on T, denoted $\leq_P$ and $\leq_G$, which satisfy the properties of reflexivity, antisymmetry and transitivity. Let τ and τ' be two model trees in T. Then $\tau' \leq_P \tau$ iff for each path p' from the root of τ' to a leaf in τ', there exists a path p from the root of τ to a leaf of τ such that the label associated with p', $LT'(p')$, is a prefix of $LT(p)$. Moreover, $\tau' \leq_G \tau$, iff for each path p' from the root of τ' to a leaf in τ', there exists a path p from the root of τ to a leaf of τ, such that $LT'(p')$ is obtained from $LT(p)$ by dropping some continuous/discrete split labels in the sequence.

With reference to Figure 4.5, $\tau' \leq_P \tau$ and $\tau'' \leq_G \tau$, while the relations $\tau'' \leq_P \tau$ and $\tau' \leq_G \tau$ do not hold. Hence, given a SMOTI tree τ, it is possible to define two sets of trees, namely:

$$S_P(\tau) = \{\tau' \in \mathsf{T} | \tau' \leq_P \tau\}, \tag{4.35}$$

$$S_G(\tau) = \{\tau' \in \mathsf{T} | \tau' \leq_G \tau\}. \tag{4.36}$$

We observe that $S_P(\tau) \nsubseteq S_G(\tau)$ and $S_G(\tau) \nsubseteq S_P(\tau)$, since $\tau' \leq_P \tau$ does not imply $\tau' \leq_G \tau$ and viceversa.

The *pruning operator* is defined as the function:

$$\pi_\tau : N_\tau^I \rightarrow \mathsf{T} \tag{4.37}$$

The operator π_τ associates each internal node $t \in N_\tau^I$ with the tree $\pi_\tau(t)$, which has all the nodes of τ except the descendants of t.

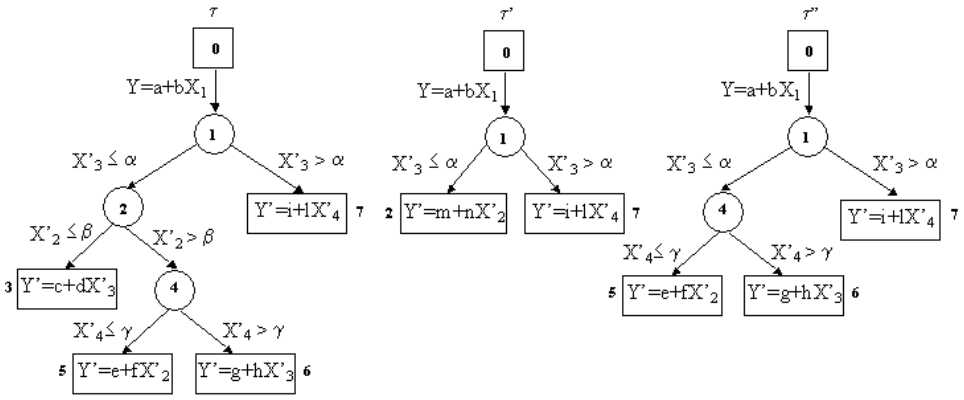


FIGURE 4.5: The model tree τ' is obtained by pruning τ in node 2, while τ'' is obtained by grafting the subtree rooted in node 4 onto the place of node 2.

Analogously, the *grafting operator* is defined as the function:

$$\gamma_\tau : N_\tau^{I_S} \times N_\tau^I \rightarrow \mathsf{T}, \tag{4.38}$$

that associates each couple of internal nodes $\langle t, t' \rangle \in N_\tau^{I_S} \times N_\tau^I$, with the tree $\gamma_\tau(\langle t, t' \rangle)$, which has all nodes of τ except those in the branch between t and t'.

Intuitively, the pruning operator applied to a node of a tree τ returns a tree $\tau' \leq_P \tau$ while the grafting operator returns a tree $\tau' \leq_G \tau$ (see Figure 4.5).

The problem of simplifying a model tree can be cast as a search in a state space, where states are trees in either $S_P(\tau)$ or $S_G(\tau)$, and pruning and grafting are the only operators that can be applied to move from one state to another.

In order to give a precise definition of a simplification method the goal of the search in the state space has to be defined. For this reason, a function f that estimates the goodness of a tree is introduced. It associates each tree in the space $S(\tau)$ with a continuous value, namely:

$$f : S(\tau) \rightarrow \Re, \tag{4.39}$$

where $\Re$ is the set of real values. The goal of the search is to find the state in $S(\tau)$ with the highest f value, so that pruning can be cast as a problem of function optimization.

Finally, the way in which the state space is explored also characterizes different simplification methods, which can be formally described by a 4-tuple:

$$(\textit{Space}, \textit{Operators}, \textit{Evaluation function}, \textit{Search strategy}), \tag{4.40}$$

where *Space* represents the search space of pruning methods, *Operators* is a set of simplification (pruning or grafting) operators, the *Evaluation function* associates each tree in the search space with a continuous value and the *Search strategy* is the way in which the state space is explored in order to find the optimal state.

This framework [CAM03c] is used to explain the two simplification methods we have proposed to simplify SMOTI trees.

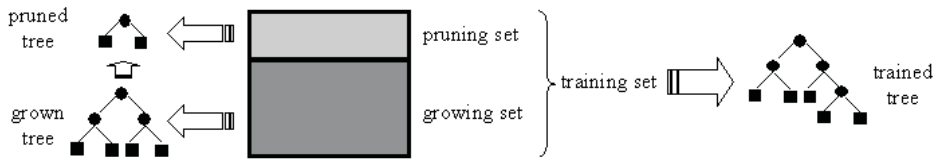


FIGURE 4.6: The original dataset can be split into two subsets: the growing set and the pruning set. The union of the growing and pruning set is called the training set. Trees learned from the growing/training set are called grown/trained trees, respectively. Pruning trees can be obtained by pruning either grown trees or trained trees. In the former case, a pruning set is used.

4.3.2 Reduced Error Pruning

This method is based on the Reduced Error Pruning (REP) proposed by Quinlan for decision trees [Qui87]. It uses a pruning set to evaluate the goodness of the subtrees of a model tree τ. This pruning set is independent of the set of observations used to build the tree τ, therefore, the training set must be partitioned into a *growing* set used to build the tree and a *pruning* set used to simplify τ (Figure 4.6).

The top-level description of REP (see Algorithm 4.2) shows that REP search is accomplished in the pruning state space, $(S_P(\tau), \{\pi_\tau\})$ by means of the first-better strategy, according to which we move from one state τ to a state τ' just generated if τ' is better than τ with respect to the evaluation function f. Differently from the hill-climbing search, there is no generation of all states directly reachable from τ in order to select the best one. Moreover, the first better strategy differs from the well-known best first strategy in the storing of only one generated state. Obviously, in this search strategy, the order in which states are generated is of crucial importance. It depends on:

1. the traversal order: pre-order or post-order,

2. the direction of pruning: bottom-up or top-down.

In REP, the traversal is post-order and the direction is bottom-up. The evaluation function f is defined as follows:

$$f(\tau) = \sum_{t \in N_\tau^L} MSE(t), \tag{4.41}$$

where $MSE(t)$ is the mean square error at leaf t computed on the pruning set. Consequently, the search in the space moves from a state τ_1 to a state $\tau_2 \in \pi_{\tau_1}(N_{\tau_1}^I)$ iff $f(\tau_1) \geq (\tau_2)$. More precisely, REP algorithm analyzes the complete tree τ and, for each internal node t, it compares the MSE made on the pruning set when the subtree τ_t rooted in t is kept, with the MSE made when τ_t is pruned and the best regression function is associated with the leaf t. If the simplified tree has a better performance than the original one, it is advisable to prune τ_t. This pruning operation is repeated on the simplified tree until further pruning increases the value of f.

Algorithm 4.2 Reduced Error Pruning Algorithm

```
PROCEDURE REP(τ, D_P, τ_P, f_P):
INPUT:
   τ:    model tree built by SMOTI on the growing set D_G;
   D_P:  pruning set independent from the growing set D_G;
OUTPUT:
   τ_P:  pruned tree;
   f_P:  value returned by f(τ_P);
BEGIN
   IF D_P is empty THEN τ_P = τ; f_P = 0;
   ELSE
      IF the tree τ is a leaf THEN τ_P = τ; f_P = f(τ, D_P);
      ELSE
         IF the root t_0 of τ is a splitting node with τ_L (τ_R) left
             (right) branch of t_0 in τ THEN
            partition D_P into {D_{P_L}, D_{P_R}} according to the split t_0;
            REP(τ_L, D_{P_L}, τ_{P_L}, f_{P_L}); REP(τ_R, D_{P_R}, τ_{P_R}, f_{P_R});
            f_P = f({t'_0}, D_P) where π_τ(t_0) = {t'_0};
            IF f_P ≤ f_{P_L} + f_{P_R} THEN τ_P = {t'_0};
            ELSE
               τ_P = tree(t_0,τ_{P_L},τ_{P_R}); f_P = f_{P_L} + f_{P_R};
            END IF;
         ELSE;
            IF the root t_0 is a regression node with τ_R the unique
                branch of t_0 in τ THEN
               remove the effect of the regression from D_P into D_P^1;
               REP(τ_R, D_P^1, τ_{P_R}, f_{P_R});
               f_P = f({t'_0}, D_P) where π_τ(t_0) = {t'_0};
               IF f_P ≤ f_{P_R} THEN τ_P = {t'_0};
               ELSE
                  τ_P = tree(t_0,τ_{P_R}); f_P = f_{P_R};
               END IF;
            END IF;
         END IF;
      END IF;
   END IF;
END PROCEDURE;
```

As already observed in [EK01], Quinlan's description of REP for decision trees does not specify how to choose the class associated with the pruned nodes. Following the majority class criterion, three alternatives are possible: the class is determined on the basis of the growing set, the pruning set or the training set. Analogously, in the case of model trees, the straight-line regression to be associated with a pruned node can be determined on the basis of one of the three sets: growing, pruning or training.

The following optimality theorem can be proven:

Theorem 4.1 *Given a model tree τ constructed on a set of growing observations D_G and a pruning set D_P, the REP version that determines the regression model on D_G returns the smallest tree in $S_P(\tau)$ with the lowest error with respect to D_P.*

The specification "*the REP version that determines the regression model on D_G*" refers to the fact that once a node t has been pruned, the model associated with t is determined on the basis of the same growing set D_G. Alternatively, it could be determined on the basis of either the pruning set or the whole training set.

Proof 4.1 We prove the theorem by induction on the depth of τ.

Base case. Let τ be a root tree $\{t_0\}$. Then τ is the only tree in $S_P(\tau)$ and REP returns τ, since no pruning operation is possible.

Inductive Step. The proof is based on the additive property of the *MSE* for model trees, according to which a local optimization on each branch τ_{t_i} of t_0 leads to a global optimization on τ.

We assume the inductive hypothesis to be true for all model trees τ' of depth d and we prove the theorem for the tree τ of depth $d+1$, $d \geq 0$. Since τ has a depth greater than 0, there are two possibilities:

1. τ is the tree rooted in t_0 that is a regression node with a child t_1 such that the subtree τ_{t_1} has depth d,

2. τ is the tree rooted in t_0 that is a splitting node with two children t_1 and t_2 such that both subtrees τ_{t_1} and τ_{t_2} have maximum depth d.

Case 1. REP, which follows the bottom-up direction when pruning, first prunes τ_{t_1} and then checks whether τ should be pruned in t_0. For the inductive hypothesis, REP finds the optimally pruned tree $\tau^*_{t_1}$ for the tree rooted in t_1. Let τ' be the tree rooted in t_0, whose subtree is $\tau^*_{t_1}$. Then according to the definition of f, $f(\tau') = f(\tau^*_{t_1})$, since τ' and $\tau^*_{t_1}$ have the same leaves. Moreover, for any tree $\tau'' \in S_P(\tau)$ of depth greater than 0 we have: $f(\tau') \leq f(\tau'')$, since

$$f(\tau') = \sum_{t \in N^L_{\tau^*_{t_1}}} MSE(t) \leq \sum_{t \in N^L_{\tau''_{t_1}}} MSE(t) = f(\tau'').$$

Therefore, if $f(\{t_0\}) \leq f(\tau')$ then REP prunes τ in t_0, and the returned tree is the best subtree of τ, since

$$MSE(\{t_0\}) = f(\{t_0\}) \leq f(\tau') \leq f(\tau''),$$

for any tree $\tau'' \in S_P(\tau)$ of depth greater than 0.

On the contrary, if $f(\{t_0\}) > f(\tau')$ then REP does not prune τ in t_0 and the returned tree is τ', which is the smallest tree in $S_P(\tau)$ with the lowest error with respect to D_P.

Case 2. Analogously, in the case of the splitting node, REP follows the bottom-up direction so that it first prunes τ_{t_1} and τ_{t_2} and then checks whether τ should be pruned in t_0. For the inductive hypothesis REP finds the optimally pruned tree $\tau^*_{t_1}$ for the tree rooted in t_1 and $\tau^*_{t_2}$ for the tree rooted in t_2.

Let τ' be the tree rooted in t_0, whose subtrees are $\tau^*_{t_1}$ and $\tau^*_{t_2}$. Then:

$$f(\tau') = f(\tau^*_{t_1}) + f(\tau^*_{t_2}),$$

since the leaves of τ' are leaves of either $\tau^*_{t_1}$ or $\tau^*_{t_2}$. Moreover, for any tree $\tau'' \in S_P(\tau)$ of depth greater than 0, we have $f(\tau') \leq f(\tau'')$ since

$$f(\tau') = \sum_{t \in N^L_{\tau^*_{t_1}}} MSE(t) + \sum_{t \in N^L_{\tau^*_{t_2}}} MSE(t) \leq$$

$$\sum_{t \in N^L_{\tau''_{t_1}}} MSE(t) + \sum_{t \in N^L_{\tau''_{t_2}}} MSE(t) = f(\tau'')$$

Therefore, if $f(\{t_0\}) \leq f(\tau')$ then REP prunes τ in t_0, and the returned tree is the best subtree of τ, since

$$MSE(\{t_0\}) = f(\{t_0\}) \leq f(\tau') \leq f(\tau''),$$

for any tree $\tau'' \in S_P(\tau)$ of depth greater than 0.

On the contrary, if $f(t_0) > f(\tau')$ then REP does not prune τ in t_0 and the returned tree is τ', which is the smallest tree in $S_P(\tau)$ with the lowest error with respect to D_P.

■

Finally, the computational complexity of REP is linear in the number of internal nodes, since each node is visited only once to evaluate the opportunity of pruning it.

4.3.3 Reduced Error Grafting

The Reduced Error Grafting (REG) is conceptually similar to REP and uses a pruning set to evaluate the goodness of τ', a subtree of τ. However, the search is performed in the grafting state space, $(S_G(\tau), \{\gamma_\tau\})$, according to a first-better strategy with bottom-up post-order traversal. The evaluation function is the same defined for REP.

The search in $S_G(\tau)$ moves from a state τ_1 to a state $\tau_2 \in \gamma_{\tau_1}(N^{I_S}_\tau, N^I_\tau)$ if the inequality $f(\tau_1) \geq f(\tau_2)$ holds. More precisely, REG algorithm operates recursively. It analyzes the complete tree τ and, for each splitting node t, it compares the mean square error made on the pruning set when the subtree τ_t is kept, with the mean square error error made on the pruning set when τ_t is turned into $\text{REG}(\tau_{t_1})$ or $\text{REG}(\tau_{t_2})$, where t_1 and t_2 are children of t. Sometimes, the simplified tree has a better performance than the original one. In this case, it appears convenient to replace t with its best simplified subtree (left or right). This grafting operation is repeated on the simplified tree until the *MSE* increases.

This method (see Algorithm 4.3) is theoretically favored with respect to REP, since it allows the replacement of a subtree by one of its branches. In this way, it is possible to overcome a limit of those simplification strategies that make use of the pruning operator alone. If t is a node that should be pruned according to some criterion, while t' is a child of t that should not be pruned according to the same

criterion, such simplification strategy either prunes and loses the accurate branch or does not prune at all and keeps the inaccurate branch τ_t. On the contrary, REG acts by grafting $\tau_{t'}$ onto the place of t, so saving the good sub-branch and deleting the useless node t.

Algorithm 4.3 Reduced Error Grafting Algorithm

PROCEDURE REG(τ, D_P, τ_G, f_G):
INPUT:
τ: model tree built by SMOTI on the growing set D_G;
D_P: pruning set independent from the growing set D_G;
OUTPUT:
τ_G: pruned tree;
f_G: value returned by $f(\tau_G)$;
BEGIN
IF D_P is empty THEN $\tau_G = \tau$; $f_G = 0$;
ELSE
IF the tree τ is a leaf THEN $\tau_G = \tau$; $f_G = f(\tau, D_P)$;
ELSE
IF the root t_0 is a splitting node with τ_L (τ_R) left (right) branch of t_0 in τ THEN
partition D_P into $\{D_{P_L}, D_{P_R}\}$ according to the split t_0;
$\tau_L^1 = \tau_L$; $\tau_R^1 = \tau_R$;
REG(τ_L, D_{P_L}, τ_{G_L}, f_{G_L}); REG(τ_R, D_{P_R}, τ_{G_R}, f_{G_R});
REG(τ_L^1, D_P, $\tau_{G_L}^1$, $f_{G_L}^1$); REG(τ_R^1, D_P, $\tau_{G_R}^1$, $f_{G_R}^1$);
IF $f_{G_L} + f_{G_R} < f_{G_L}^1$ and $f_{G_L} + f_{G_R} < f_{G_R}^1$ THEN
$\tau_G =$ tree(t_0,τ_{G_L},τ_{G_R}); $f_G = f_{G_L} + f_{G_R}$;
ELSE
IF $f_{G_L}^1 > f_{G_R}^1$ THEN $\tau_G = \tau_{G_R}^1$; $f_G = f_{G_R}^1$;
ELSE $\tau_G = \tau_{G_L}^1$; $f_G = f_{G_L}^1$;
END IF;
END IF;
ELSE
IF the root t_0 is a regression node THEN
remove the effect of the regression from D_P into D_P^1;
REG(τ_{t_0}, D_P^1, τ_{G_R}, f_G); $\tau_G =$ tree(t_0,τ_{G_R});
END IF
END IF;
END IF;
END IF;
END PROCEDURE;

Similarly to REP, a theorem on the optimality of the tree returned by REG can be proven.

Theorem 4.2 *Given a model tree τ constructed on a set of growing observations D_G and a pruning set D_P, the REG version that determines the regression model on D_G returns the smallest tree in $S_G(\tau)$ with the lowest error with respect to D_P.*

Proof 4.2 We prove the theorem by induction on the depth of τ.

Base case. Let τ be a root tree $\{t_0\}$. Then τ is the only tree in $S_G(\tau)$ and REG returns τ since no grafting operation is possible.

Inductive Step. We assume the inductive hypothesis to be true for all model trees τ' of depth d and we prove the theorem for the tree τ of depth $d+1$, $d \geq 0$. Since τ has a depth greater than 0, there are two possibilities:

1. τ is the tree rooted in t_0 that is a regression node with a child t_1, such that the subtree τ_{t_1} has depth d,
2. τ is the tree rooted in t_0 that is a splitting node with two children, t_1 and t_2 such that both subtrees τ_{t_1} and τ_{t_2} have maximum depth d.

Case 1. REG, which follows the bottom-up direction first simplifies τ_{t_1}. For the inductive hypothesis REG finds the optimally simplified tree $\tau^*_{\bar{t}_1}$ for the tree τ_{t_1}[8].

Let τ' be the tree rooted in t_0, whose subtree is $\tau^*_{\bar{t}_1}$. Then, according to the definition of f, $f(\tau') = f(\tau^*_{\bar{t}_1})$, since τ' and $\tau^*_{\bar{t}_1}$ have the same leaves. Moreover, for any tree $\tau'' \in S_G(\tau)$ of depth greater than 0 and rooted in t_0 with child t'', we have: $f(\tau') \leq f(\tau'')$, since

$$f(\tau') = \sum_{t \in N^L_{\tau^*_{\bar{t}_1}}} MSE(t) \leq \sum_{t \in N^L_{\tau''_{\bar{t}'_1}}} MSE(t) \leq f(T").$$

Therefore, REG returns τ', which is the smallest tree in $S_G(\tau)$ with the lowest error with respect to D_P.

Case 2. In the case of a splitting node, REG follows the bottom-up direction, simplifies τ_{t_1} and τ_{t_2} with respect to D_P and then checks whether one of the simplified subtrees should be grafted in t_0. We denote the set of examples which fall in τ_{t_1} (τ_{t_2}) as D_{P_1} (D_{P_2}). Let $\tau^*_{\bar{t}_1}$ ($\tau^*_{\bar{t}_2}$) be the subtree rooted in t_1 (t_2), returned by REG when pruned with respect to D_{P_1} (D_{P_2}). For the inductive hypothesis, $\tau^*_{\bar{t}_1}$ ($\tau^*_{\bar{t}_2}$) is the optimally grafted subtree rooted in t_1 (t_2) with respect to D_{P_1} (D_{P_2}), respectively, since the depth of $\tau^*_{\bar{t}_1}$ ($\tau^*_{\bar{t}_2}$) is not greater than d. Analogously, let $\tau^*_{\hat{t}_1}$ ($\tau^*_{\hat{t}_2}$) be the subtree rooted in t_1 (t_2), returned by REG when pruned with respect to D_P. For the inductive hypothesis, $\tau^*_{\hat{t}_1}$ ($\tau^*_{\hat{t}_2}$) is the optimally grafted subtree rooted in t_1 (t_2) with respect to D_P.

Let τ' be the tree rooted in t_0 whose subtrees are $\tau^*_{\bar{t}_1}$ and $\tau^*_{\bar{t}_2}$. Then, for any tree $\tau'' \in S_G(\tau)$ of depth greater than 0 and rooted in t_0 with children t'_1 and t'_2, we have $f(\tau') \leq f(\tau'')$, since

$$f(\tau') = \sum_{t \in N^L_{tau^*_{\bar{t}_1}}} MSE(t) + \sum_{t \in N^L_{tau^*_{\bar{t}_2}}} MSE(t) \leq.$$

[8] Note that the grafting operation applied to the tree τ_{t_1} may not produce a tree rooted in t_1, so we denote it as $\tau^*_{\bar{t}_1}$.

$$\sum_{t \in N^L_{tau''_{\widehat{t}_1}}} MSE(t) + \sum_{t \in N^L_{tau''_{\widehat{t}_2}}} MSE(t) = f(\tau'').$$

Therefore, if $f(\tau^*_{\widehat{t}_1}) \leq f(\tau')$ and $f(\tau^*_{\widehat{t}_1}) \leq f(\tau^*_{\widehat{t}_2})$, where $f(\tau^*_{\widehat{t}_1})$ and $f(\tau^*_{\widehat{t}_2})$ are computed with respect to D_P, then REG replaces τ with $\tau^*_{\widehat{t}_1}$, which is the best subtree of T, since

- $MSE(\tau^*_{\widehat{t}_1}) = f(\tau^*_{\widehat{t}_1}) \leq f(\tau') \leq f(\tau'')$ for any tree $\tau'' \in S_G(\tau)$ rooted in t_0 with depth greater than 0,
- $MSE(\tau^*_{\widehat{t}_1}) = f(\tau^*_{\widehat{t}_1}) \leq f(\tau'')$ for any tree $\tau'' \in S_G(\tau)$ not rooted in t_0 (both $\tau^*_{\widehat{t}_1}$ and τ'' have maximum depth d and the inductive hypothesis holds on D').

Otherwise, if $f(\tau^*_{\widehat{t}_2}) \leq f(\tau')$ and $f(\tau^*_{\widehat{t}_2}) \leq f(\tau^*_{\widehat{t}_1})$, REG replaces τ' with $\tau^*_{\widehat{t}_2}$, which is the best subtree of τ, since

- $MSE(\tau^*_{\widehat{t}_2}) = f(\tau^*_{\widehat{t}_2}) \leq f(\tau') \leq f(\tau'')$ for any tree $\tau'' \in S_G(\tau)$ rooted in t_0 with depth greater than 0,
- $MSE(\tau^*_{\widehat{t}_2}) = f(\tau^*_{\widehat{t}_2}) \leq f(\tau'')$ for any tree $\tau'' \in S_G\tau$ not rooted in t_0 (both $\tau^*_{\widehat{t}_1}$ and τ'' have maximum depth d and the inductive hypothesis holds on D_P).

Finally, if both $f(\tau^*_{\widehat{t}_1}) > f(\tau')$ and $\tau^*_{\widehat{t}_2} > f(\tau')$ then REG does not simplify τ in t_0 and the returned tree is τ'. Obviously, τ' is better than any tree $\tau'' \in S_G(\tau)$ rooted in t_0. Moreover, τ' is better than any tree $\tau'' \in S_G(\tau)$ not rooted in t_0, since either $\tau'' \in S_G(\tau_{t_1})$ and $f(\tau'') \geq f(\tau^*_{\widehat{t}_1}) > f(\tau')$ or $\tau'' \in S_G(\tau_{t_2})$ and $f(\tau'') \geq f(\tau^*_{\widehat{t}_2}) > f(\tau')$. Therefore, the returned tree τ' is the smallest tree in $S_G(\tau)$ with the lowest error with respect to D_P.

■

The complexity of REG is $\boldsymbol{O}(\#N_\tau \times \lg_2 \#N_\tau)$, where $\#N_\tau$ is the number of nodes in τ.

4.4 Experimental results

SMOTI has been implemented as a module of the knowledge discovery system KDB2000[ACM02] (http://www.di.uniba.it/∼malerba/software/kdb2000/) and has been empirically evaluated both on artificially generated data and on datasets typically used in the evaluation of regression and model trees. Each dataset is analyzed by means of a 10-fold cross-validation. The system performance is evaluated on the basis of the average root mean square error (*Avg.RMSE*) computed as follows:

$$Avg.RMSE = \frac{1}{k} \sum_{v \in V} \sqrt{\frac{1}{n(\bar{v})} \sum_{j \in v} (y_j - \hat{y}_j(\bar{v}))^2}, \tag{4.42}$$

where $V = \{v_1, \ldots, v_k\}$ is a cross-validation partition, each v_i is a set of indices of training cases, k is the number of folds (i.e., 10), $n(\bar{v})$ is the number of cases in

$V - v$, and $\hat{y}_j(\bar{v})$ is the value predicted for the j-th training case by the model tree built from $V - v$.

For pairwise comparison of methods, the non-parametric Wilcoxon two-sample paired signed rank test is used [OD90], since the number of folds (or "independent" trials) is relatively low and does not justify the application of parametric tests, such as the t-test. To perform the test, we assume that the experimental results of the two methods compared are independent pairs of sample data $\{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$. We then rank the absolute value of the differences $u_i - v_i$. The Wilcoxon test statistics W^+ and W^- are the sum of the ranks from the positive and negative differences, respectively. We test the null hypothesis H_0: "no difference in distributions" against the two-sided alternative H_1: "there is a difference in distributions". More formally, the hypotheses are: H_0: "$\mu_u = \mu_v$" against H_1: "$\mu_u \neq \mu_v$". Intuitively, when $W^+ \gg W^-$ and vice-versa, H_0 is rejected. Whether W^+ should be considered "much greater than" W^- depends on the significance level α. The basic assumption of the statistical test is that the two populations have the same continuous distribution (and no ties occur). Since, in our experiments, u_i and v_i are root mean square error *RMSE*, $W^+ \gg W^-$ implies that the second method (V) is better than the first (U). In all experiments reported in this empirical study, the significance level α used in the test is set at 0.05.

SMOTI has been compared to both M5', which is considered the state-of-the-art model tree induction system, and RETIS, which has an evaluation function coherent with the models at the leaves[9]. The empirical comparison with TSIR, which is the only other system with regression and splitting nodes, was not possible since the system is not publicly available.

Moreover, both REP and REG simplification performances (i.e. accuracy and tree size) have been evaluated on some benchmark datasets. SMOTI trees built on training/growing set are compared with the corresponding trees simplified with REP and REG as well as the un-pruned/pruned model trees induced with M5'.

4.4.1 Evaluating SMOTI on artificial datasets

SMOTI was initially tested on artificial data sets randomly generated for model trees with both regression and splitting nodes. These model trees were automatically built for learning problems with nine independent variables (five continuous and four discrete) where discrete variables take values in the set $\{A, B, C, D, E, F, G\}$. The model tree building procedure is recursively defined on the maximum depth of the tree to be generated. The choice of adding a regression or a splitting node is random and depends on a parameter $\theta \in [0, 1]$: the probability of selecting a splitting node is θ; conversely, the probability of selecting a regression node is $(1 - \theta)$. In the experiments reported in this paper, θ is fixed at 0.5, while the depth varies from four to nine. Fifteen model trees are generated for each depth value, for a total of ninety trees.

[9] When running M5'in empirically evaluating un-pruned trees, the pruning factor (parameter $-F$) is set to 0, in order to evaluate un-pruned trees. For the same reason, the pruning function is not invoked in RETIS. All remaining parameters are set to default values.

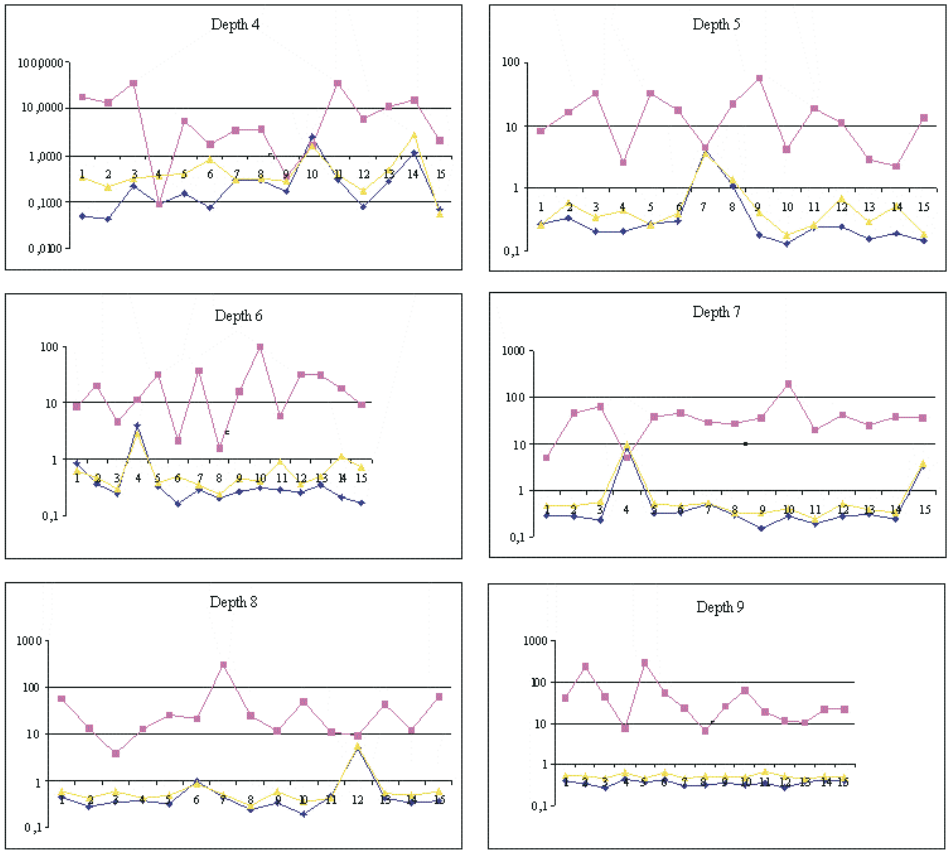


FIGURE 4.7: Average root mean square error (Y axis) in log scale for fifteen induced model trees (X axis) of different depth. The comparison concerns three systems: M5' (yellow triangles), RETIS (purple squares) and SMOTI (blue diamonds).

Sixty data points are randomly generated for each leaf, so that the size of the dataset associated with a model tree depends on the number of leaves in the tree itself. Data points are generated by considering the various constraints associated with both splitting nodes and regression nodes. In the case of a splitting node, the only constraint is that the distribution of cases between left and right children should take into account the number of leaves in each subtree. In the case of a regression node, the constraints are the (partial) multiple linear model associated with the node, as well as the linear models defined for the residuals of the variables passed down. The noise effect is introduced by adding a normally distributed error $\sim N(0,1)$ to the linear models relating independent variables and $\sim N(0,.001)$ to the linear models at the leaves involving the dependent variable. In all experiments, the thresholds for stopping criteria are fixed as follows: the significance level α used in the *F-test* is set to 0.075, the minimum number of cases falling in each internal node must be greater than the square root of the number of cases in the entire training set, the error in each internal node must be greater than the 0.01% of the error in the root node, the coefficient of determination in each internal node must be below 0.99.

In Figure 4.7, the *Avg.RMSE* is reported for each dataset generated by a theo-

TABLE 4.2: SMOTI vs RETIS: results of the Wilcoxon signed rank test on the accuracy of the induced model trees. The statistically significant values ($p - value \leq \alpha/2$) are in boldface. The symbol '-' means that SMOTI performs worse than RETIS. All statistically significant values are favorable to SMOTI.

	Depth 4	**Depth 5**	**Depth 6**	**Depth 7**	**Depth 8**	**Depth 9**
D_1	**0.001953**	**0.001953**	**0.001953**	**0.001953**	**0.001953**	**0.001953**
D_2	**0.001953**	**0.005859**	**0.001953**	**0.001953**	**0.001953**	**0.001953**
D_3	**0.001953**	**0.001953**	**0.001953**	**0.001953**	**0.001953**	**0.001953**
D_4	(-) 0.1602	**0.001953**	0.2754	(-) 0.0839	**0.001953**	**0.001953**
D_5	**0.001953**	**0.001953**	**0.001953**	**0.001953**	**0.001953**	**0.001953**
D_6	**0.001953**	**0.001953**	**0.001953**	**0.001953**	**0.001953**	**0.001953**
D_7	**0.003906**	(-) 0.8457	**0.001953**	**0.001953**	**0.001953**	**0.001953**
D_8	**0.009766**	(-) 0.0839	**0.001953**	**0.001953**	**0.001953**	**0.001953**
D_9	0.1309	**0.001953**	**0.001953**	0.08398	**0.001953**	**0.001953**
D_{10}	(-) 0.0488	**0.001953**	**0.001953**	**0.001953**	**0.001953**	**0.001953**
D_{11}	**0.001953**	**0.001953**	**0.001953**	**0.001953**	**0.001953**	**0.001953**
D_{11}	**0.003906**	**0.003906**	**0.001953**	**0.001953**	0.375	**0.001953**
D_{12}	**0.001953**	**0.001953**	**0.001953**	**0.001953**	**0.001953**	**0.001953**
D_{14}	**0.007812**	**0.005859**	**0.001953**	**0.001953**	**0.001953**	**0.001953**
D_{15}	(-) 1	**0.007812**	**0.009766**	**0.001953**	**0.001953**	**0.001953**

TABLE 4.3: SMOTI vs M5': results of the Wilcoxon signed rank test on the accuracy of the induced model trees. The statistically significant values ($p - value \leq \alpha/2$) are in boldface. The symbol '-' means that SMOTI performs worse than M5'. All statistically significant values are favorable to SMOTI.

	Depth 4	**Depth 5**	**Depth 6**	**Depth 7**	**Depth 8**	**Depth 9**
D_1	**0.0019**	(-) 0.375	0.275	**0.0019**	**0.0039**	**0.0097**
D_2	**0.0019**	**0.019**	0.0839	**0.0019**	**0.0019**	**0.0019**
D_3	0.0645	**0.0019**	0.0644	**0.0019**	**0.0019**	**0.0019**
D_4	**0.0019**	**0.0058**	(-)0.769	(-)0.375	**0.0136**	**0.0019**
D_5	**0.0019**	0.7695	0.4316	**0.0019**	**0.0019**	0.0644
D_6	**0.0019**	0.12	**0.0019**	**0.0058**	0.0839	**0.0019**
D_7	0.8457	(-)0.2754	**0.0136**	0.4922	0.0839	**0.0019**
D_8	**0.0234**	0.375	**0.0039**	0.2324	**0.0019**	**0.0019**
D_9	0.0644	**0.0019**	**0.0019**	**0.0019**	**0.0019**	**0.0019**
D_{10}	(-) 0.2754	0.1934	**0.0019**	**0.0097**	**0.0019**	**0.0039**
D_{11}	**0.0136**	0.2969	**0.0097**	0.0839	0.6953	**0.0195**
D_{12}	0.0273	**0.0019**	**0.0019**	**0.0019**	0.4316	**0.0019**
D_{13}	**0.0019**	**0.0019**	**0.0195**	**0.0019**	0.1309	**0.0039**
D_{14}	**0.0019**	**0.0019**	**0.0039**	**0.0019**	**0.0019**	**0.0019**
D_{15}	(-)0.375	0.1934	**0.0039**	0.1934	**0.0058**	**0.0039**

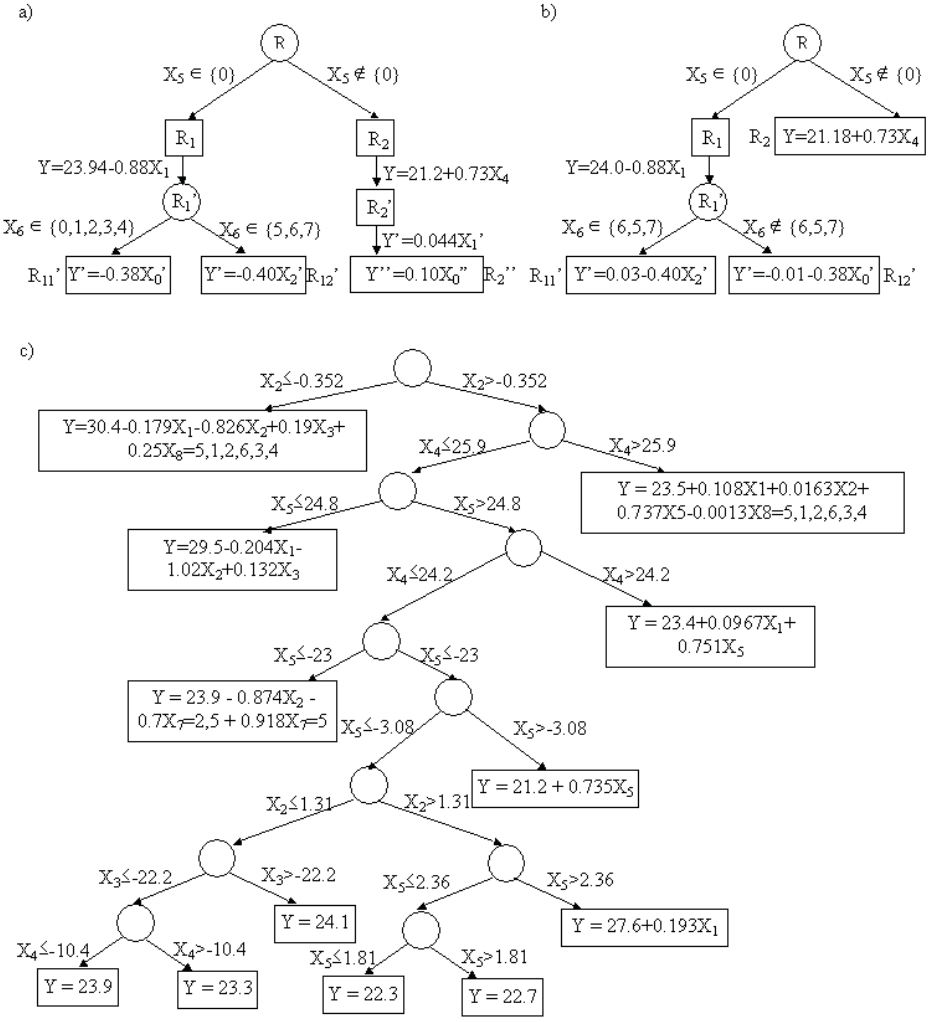


FIGURE 4.8: a) A theoretical model tree of depth 4 used in the experiments; b) the model tree induced by SMOTI from one of the cross-validated training sets; c) the corresponding model tree built by M5' for the same data.

retical model tree, while in Tables 4.2 and 4.3 the results of Wilcoxon test on the accuracy of trees induced by SMOTI, M5' and RETIS are reported. Three main conclusions can be drawn from these experimental results: first, SMOTI performs generally better than M5' and RETIS on data generated from model trees where both local and global effects can be represented; second, by increasing the depth of the tree, SMOTI tends to be more accurate than M5' and RETIS; third, when SMOTI performs worse than M5' and RETIS, this is due to relatively few hold-out blocks in the cross validation, so that the difference is never statistically significant in favor of M5' or RETIS.

An example of different results provided by SMOTI and M5' is reported in Figure 4.8. The underlying model tree, according to which a dataset of 180 cases is generated, is reported in 4.8a. It firstly partitions the feature space into two subregions:

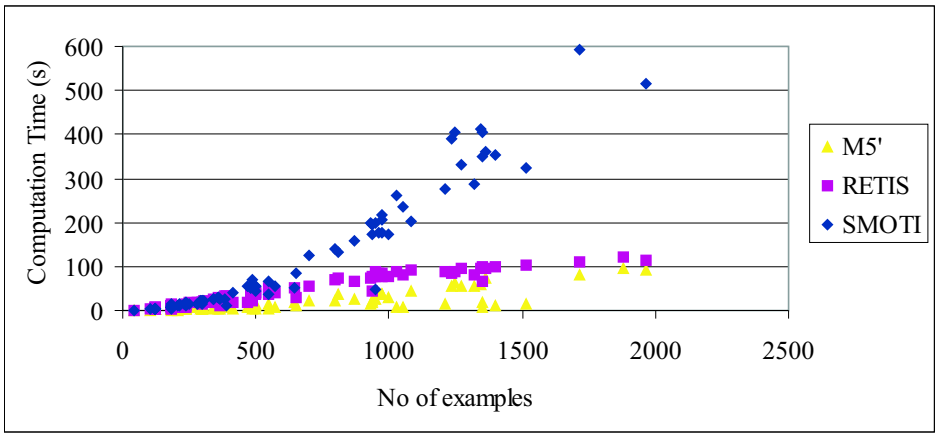


FIGURE 4.9: Running time on artificial data sets. Experiments are performed on a Pentium III PC - 366MHz running Windows 98.

$R_1 : \{(\boldsymbol{X}, Y)|X_5 \in \{0\}\}$; $R_2 : \{(\boldsymbol{X}, Y)|X_5 \in \{1,2,3,4,5,6,7\}\}$.

The subregion R_1 is in turn partitioned into two subregions:

$R_{11} : \{(\boldsymbol{X'}, Y')|X_6 \in \{0,1,2,3,4\}\}$; $R_{12} : \{(\boldsymbol{X'}, Y')|X_6 \in \{5,6,7\}\}$.

The variable X_1, which contributes to the regression models associated with both R_{11} and R_{12} has a global effect on the response variable Y since its coefficient can be reliably estimated on the region R_1. On the contrary, the variables X_0 and X_2 have a local effect, since their contributions to the regression models at the leaves can be estimated on the basis of the cases falling in the subregions R_{11}, R_{12} and R_2 associated with the leaves. Actually, straight-line regressions at the leaves involve variables X_0', X_2' and X_0'', which are obtained by removing the effect of other variables already introduced in the model. It is noteworthy that the intercepts of straight-line regressions associated with nodes below a regression node are all equal to zero, since we are using sets of residuals whose sums are zero, and thus the lines must pass through the origin.

The tree built by SMOTI on a cross-validated training set of 162 cases is shown in Figure 4.8.b. It well approximates the underlying model by discovering both global and local effects. The tree found by M5' (see Figure 4.8.c) is less accurate on the validation set of the remaining 18 cases and is not easily interpretable, especially because of the smoothing process adopted by the system to compensate for the sharp discontinuities that occur between linear models at adjacent regions [WW97].

The clear superiority of SMOTI on these datasets should not be surprising, since neither M5' nor RETIS have been designed to discover both global and local effects of variables in the underlying data model. However, this has a computational cost. Figure 4.9 plots the computation time of the three systems for the ninety artificial data sets. Naturally, M5' is the most efficient because of its evaluation function, which is incoherent with respect to the model tree being built. RETIS has time performance comparable to SMOTI for small datasets (about 650 cases), while it becomes surprisingly faster than SMOTI for larger datasets. Coherently with our theoretical analysis, SMOTI shows a quadratic behavior while RETIS

does not. There are two possible explanations of RETIS efficiency. First, our theoretical analysis of RETIS computational complexity refers only to continuous variables, since the case of discrete variables is undocumented. If RETIS uses the same criterion applied in CART and M5', then it would be more efficient than SMOTI, but its evaluation function could no longer be considered coherent with the models at the leaves. Second, an undocumented stopping criterion prevented the system from generating large model trees in the experiments[10].

4.4.2 Evaluating SMOTI on benchmark datasets

SMOTI was also tested on fourteen datasets (see Table 4.4) taken from either the UCI Machine Learning Repository (URL: http://www.ics.uci.edu/~mlearn/ ML-Repository.html) or the site of the system HTL (URL: http://www.niaad.liacc.up.pt /~ltorgo/Regression/DataSets.html) or the site of WEKA (URL: http://www.cs. waikato.ac.nz/ml/weka/). They have a continuous variable to be predicted and have been used as benchmarks in related studies on regression trees and model trees.

In all experimental results reported in this section, the thresholds for the stopping criteria are set at the same values used in the experiments on artificial datasets, except for the coefficient of determination which is set at 0.9. Experimental results are reported in Table 4.5, where SMOTI is compared to M5' and RETIS on the basis of the *Avg.RMSE*. For RETIS, not all values are available, since the system limits the number of attributes to 30 and the maximum number of distinct values for discrete attribute to 26. As in the previous experimentation, differences are considered statistically significant when the p-value is less than or equal to $\alpha/2$.

The comparison with RETIS is clearly in favor of SMOTI. Unfortunately, not all experimental results could be collected for RETIS, because of two limitations of the system on the maximum number of attributes and on the maximum number of distinct values for discrete attributes.

Differently from artificially generated data, SMOTI does not exhibit an irrefutable superiority with respect to M5', although results are still good. A deeper analysis of the experimental results evidenced that for some training sets, the thresholds defined for the stopping criteria prevented SMOTI from growing model trees more accurate than those built by M5'. This problem cannot be straightforwardly solved by defining higher thresholds, since that would lead to data overfitting problems. SMOTI can actually apply some post-pruning strategy to reduce data overfitting, however this aspect is beyond the scope of this paper.

The interesting aspect of this experimentation is that for some datasets, SMOTI detected the presence of global effects that no previous study on model trees has revealed. In the following, we account for some of them, thus proving another desirable characteristic of the system, that of easy interpretability of the induced trees. The comparison is made with M5' which outperforms RETIS.

Abalone. Abalones are marine crustaceans, whose age can be determined by counting under the microscope the rings in the cross section of the shell. Other measurements, which are easier to obtain, can be used to predict the age. For all

[10] The system often outputs the message "Too many nodes. Making a leaf".

Table 4.4: Datasets used in the empirical evaluation of SMOTI.

Dataset	Cases	Attr.	Continuous	Discrete	Goal
Abalone	4177	10	9	1	Predicting the age of abalone from physical measurements
AutoMpg	392	8	5	3	Predicting the city-cycle fuel consumption
AutoPrice	159	27	17	10	Predicting auto price
Bank8FM	4499	9	9	0	Predicting the fraction of bank customers who leave the bank because of full queues
Cleveland	297	14	7	7	Predicting the heart disease in a patient.
Delta Ailerons	7129	6	6	0	Predicting the variation in the control action on aircraft ailerons.
Delta Elevators	9517	7	7	0	Predicting the variation in the action taken on aircraft elevators.
Housing	506	14	14	0	Predicting housing values in areas of Boston
Kinematics	8192	9	9	0	Predicting the distance of the end-effector from a target in an 8 link all-revolute robot arm.
Machine CPU	209	7	7	0	Predicting CPU relative performance.
Pyrimidines	74	28	28	0	Predicting QSARs activity from the descriptive structural attributes
Stock	950	10	10	0	Predicting the daily stock price of an aerospace company from daily stock prices of other nine aerospace companies
Triazines	74	61	61	0	Predicting QSARs structure from the descriptive structural attributes
Wisconsin Cancer	186	33	33	0	Predicting the time to recur for a breast cancer case

TABLE 4.5: SMOTI vs M5' and RETIS: results of the Wilcoxon signed rank test on the accuracy of the induced models. The best Avg.RMSE is in italics. The statistically significant values (p-value$\leq \alpha/2$) are in boldface. The symbol '+' ('-') means that SMOTI performs better (worse) than M5' or RETIS. NA denotes not available. Most of statistically significant values are favorable to SMOTI.

	Avg.MSE			**SMOTI vs M5'**	**SMOTI vs RETIS**
Dataset	**SMOTI**	**M5'**	**RETIS**		
Abalone	*2.53637*	2.77242	6.03224	(+)0.1934	**(+)0.001953**
AutoMpg	*3.14938*	3.20106	NA	(+)0.5566	
AutoPrice	*2246.038*	2358.818	NA	(+)0.6953	
Bank8FM	*0.03833*	0.04099	0.46629	(+)0.064	**(+)0.001953**
Cleveland	1.31603	*1.24963*	2.97914	(-)0.2324	**(+)0.009766**
Delta Ailerons	0.000232	*0.0002*	0.00129	(-)0.6404	(+)0.02734
Delta Elevators	0.00476	*0.00163*	0.00579	(-)0.1934	(+)0.1309
Housing	*3.58*	4.27927	36.3662	(+)0.048	**(+)0.001953**
Kinematics	*0.1581*	0.194737	1.98614	**(+)0.0039**	**(+)0.001953**
MachineCPU	*55.31482*	57.35276	305.609	(+)0.5566	**(+)0.003906**
Pyrimidines	0.10566	0.09279	*0.07813*	(-)0.8457	(-)0.4316
Stock	1.8225	*1.10932*	1.59318	(-)0.03711	(-)0.4375
Triazines	0.2017	*0.15503*	NA	**(-)0.02**	
Wisconsin Cancer	51.41376	*45.40644*	NA	(-)0.625	

ten cross-validated training sets, SMOTI builds a model tree with a regression node in the root. The straight-line regression selected at the root is almost invariant for all model trees and expresses a linear dependence between the number of rings (dependent variable) and the shucked weight (independent variable). This is a clear example of a global effect, which cannot be grasped by examining the nearly 350 leaves of the un-pruned model tree induced by M5' on the same data. Interestingly, the child of the root is always a splitting test on the whole weight, or, more precisely, on the residuals of the whole weight once the effect of the shucked weight has been removed. As for the root, the threshold selected for this continuous split is almost the same for all ten induced model trees. Unfortunately, this stability of the tree structure occurs only at the root and its child.

Auto-Mpg. The data concerns city-fuel consumption in miles per gallon. For all ten cross-validated training sets, SMOTI builds a model tree with a discrete split test in the root. The split partitions the training cases in two subgroups, one whose *model year* is between 70 and 77 and the other whose *model year* is between 78 and 82. That can be easily explained with the measures for energy conservation prompted by the 1973 OPEC oil embargo. Indeed, in 1975 the U.S. Government set new standards on fuel consumption for all vehicles. These values, known as C.A.F.E. (Company Average Fuel Economy) standards, required that by 1985 automakers doubled average new car fleet fuel efficiency. These standards came into force only in 1978 and model trees induced by SMOTI capture this temporal

watershed. Moreover, in the case *model year* between 70 and 77, SMOTI performs another discrete splitting test on the number of cylinders, while in the case *model year* between 78 and 82 SMOTI introduces a regression step generally involving the variable *weight*. Also this difference seems reasonable, since it captures the different technologies (e.g., lightweight materials) adopted by automakers before and after the introduction of C.A.F.E. standards. Differently from SMOTI, model trees induced by M5' perform a first continuous splitting on the variable *displacement* (≤ 191 vs. > 191) and a second splitting on the variable *horsepower* for both left and right child. A test on the variable *model year* appears only at lower levels.

Auto-Price. This dataset consists of three types of entities: a) the specification of an auto in terms of various characteristics; b) its assigned insurance risk rating; c) its normalized depreciation as compared to other cars. Almost all induced trees have a regression node in the root, which expresses a linear dependence between the price (dependent variable) and the normalized losses (independent variable). Interestingly, one of the findings of a recent study (February 2000) from the Highway Loss Data Institute (HLDI) is that "sports cars and luxury cars continue to have the worst claims losses among passenger cars for crash damage repairs under insurance collision coverages. Passenger vans have the best loss result". Therefore, the global effect of *normalized losses* is confirmed by independent studies. On the contrary, the continuous splitting test on the variable *curb weight* generally performed by M5' at the root of the induced model trees seems less intuitive.

Bank8FM. This dataset is synthetically generated from a simulation of how bank-customers choose their banks. The goal is predicting the fraction of bank customers who leave the bank because of long queues. The models induced by SMOTI from all ten cross-validation sets are quite simple and are characterized by a chaining on six regression nodes starting from the root. In most of the trials the model tree is actually a chaining of only regression nodes, thus revealing the multiple linear regression nature of the problem. As shown in Table 2.3, M5' also finds good predictive model trees, although they have about 400 leaves with as many regression models associated with them.

Cleveland. The domain is heart disease diagnosis, and the data was collected from the Cleveland Clinic Foundation. The dependent variable refers to the presence of heart disease in a patient. It is an integer valued from 0 (no presence) to 4. The high *Avg.RMSE* measured for both SMOTI and M5' (> 1.2) shows the complexity of this prediction task. The tree models induced by SMOTI in almost all trials have a chaining of regression nodes involving the variables *ca* (number of major vessels (0-3) colored by flourosopy), *thalach* (maximum heart rate achieved), *age* (age in years) and *chol* (serum cholestoral in mg/dl). We actually do not know the criteria adopted by specialists to define the presence of heart disease in a patient, but it is likely that the final score was synthesized as a weighted linear combination of several factors with a global effect. Differently from SMOTI, M5' partitions by performing a test on the variable *thal*$\in$\{fixed defect; reversable defect\} vs. *thal*$\in$\{normal\} or on the variable *cp* (chest pain type): asymptomatic vs. \{typical angina, atypical angina, non-anginal pain\}. The error found by M5' on some leaves is null, since M5' approximates the dependent variable with one of the admissible values (e.g.,

constants 0 or 1).

Delta Ailerons. The problem is that of grafting the skills of flying a F16 aircraft in a flight simulator from behavioral traces of a human expert. In this control problem, the independent variables describe the status of the airplane, while the goal is to predict the control action on the ailerons of the aircraft. It is not obvious which independent variables the human pilot uses; he may build more complex variables out of simple ones or may extract them from the landscape image. What we observe is that in eight model trees induced through cross-validation, SMOTI selects regression nodes at the top four levels. Variables used in these nodes are the *roll-rate*, *diff-roll-rate*, *curr-roll* and *pitch-rate*. This means that the only variable that seems to have a local effect is *curr-pitch*. Model trees induced by M5' are more complex and therefore more difficult to interpret.

Delta Elevators. This dataset is also obtained from the task of controlling an F16 aircraft. The goal variable is related to an action taken on the elevators of the aircraft. As in the previous domain, for eight model trees induced through cross-validation, SMOTI selects regression nodes at the top five levels. Variables used in these nodes are the *diffclb*, *altitude*, *climb-rate*, *roll-rate* and *diff-diffclb*. This means that the only variable that seems to have a local effect is *curr-roll*. Once again, the model trees induced by M5' are more complex (generally more than 350 leaves).

Housing. This dataset concerns housing values in the suburbs of Boston. The goal is that of predicting the median value of owner-occupied homes in \$1000's. By treating the independent variable *chas* (an indicator variable equal to 1 if tract bounds the Charles River, 0 otherwise) as continuous, SMOTI generally creates a model tree with the regression step

$medv = 22.09 + 5.6\ chas$

in the root. Surprisingly, model trees induced by M5' almost totally neglect this indicator variable.

Kinematics. This dataset is synthetically generated from a realistic simulation of the forward kinematics of an 8 link all-revolute robot arm. The goal is predicting the distance of the end-effector from a target, given the angular position of the joints [GWJ96]. Despite the claimed high nonlinearity of the data, SMOTI finds a model tree whose top seven nodes are all regression nodes involving the independent variables *theta3*, *theta5*, *theta6*, *theta1*, *theta8*, *theta2*, and *theta4*. After the introduction of the seven nodes, the algorithm starts partitioning the dataset in many sub-regions, where linear dependencies on the remaining independent variable are considered. The simplicity of the model trees induced by SMOTI does not penalize their predictive accuracy, since M5' generates less accurate model trees with a thousand leaves.

Machine CPU. The problem concerns of relative CPU performance data. The estimated relative performance values have already be estimated [EF84] using a linear regression method and obtain 34% average deviation from actual values. The tree induced by SMOTI from nine trials has a chaining of four regression nodes as top of the tree. Attributes involved in these nodes are *MMIN* (minimum main memory in kilobytes), *CACH* (cache memory in kilobytes), *CHMIN* (minimum channels in units) and *MYCT* (machine cycle time in nanoseconds). This means

that the only attributes which seem to have a local effect are *CHMAX* (maximum channels in units) and MMAX (maximum main memory in kilobytes). Conversely, M5' with an accuracy almost equivalent to SMOTI is not able to detect the presence or absence of global/local effects.

Pyrimidines. The task consists of learning the Quantitative Structure Activity Relationships, in particular, the inhibition of dihydrofolate reductase by pyrimidines [KMLS92]. For this dataset, both M5' and SMOTI learns very simple model trees with few leaves. The model trees have almost the same predictive accuracy. Their main difference is that SMOTI detects the global effect of some variables. However, the limited training set size does not allow us to draw meaningful conclusions because of the instability of the tree structure built from the ten cross-validated training sets.

Stock. The data are daily stock prices from January 1988 through October 1991, for ten aerospace companies. The goal is to predict the daily stock price of a company with respect to daily stock price of other nine companies. Since no information is available about the effective relationship (e.g. temporal, spatial, social or economical relationship) among these companies, it is quite difficult to significantly interpret the quality of discovered regression models. Moreover, we observe that typically a different tree is built for each trial: one or more regression nodes occur in the top of the tree, but they involve different continuous variable. This does not lead to any significant conclusion about the global effect detected with a regression node in the top level of SMOTI trees. On the other hand, M5' builds model trees which are significantly more accurate than corresponding trees built by SMOTI, but they are also significantly more complex trees (average number of leaves in M5' trees equal to 120.8 against average number of leaves in SMOTI trees equal to 5).

Triazines. As for the Pyrimidines dataset, the problem is to learn a model tree which predicts the activity from the descriptive structural attributes. The data and methodology is described in detail in [HKS94] [KHS94]. M5' finds smaller and more accurate trees than those induced by SMOTI. Once again, the main difference is that SMOTI detects the global effect of some variables, but the limited training set size does not allow us to draw some conclusions on the tree structure.

Wisconsin Cancer. Data concerns consecutive patients seen by Dr. Wolberg since 1984, and include only those cases exhibiting invasive breast cancer and no evidence of distant metastases at the time of diagnosis. Model trees built by SMOTI are quite different for each trial. A regression node involving the mean fractal dimension occurs as root in the tree built from four different trials. However, different results obtained from different trials do not lead to any general conclusion on the tree structure. Conversely, M5' seems to well capture regression model underlying this data. On the other hand, model trees built by M5' are significantly more complex (i.e. average number of leaves equal to 32) than corresponding trees built by SMOTI (i.e. average number of leaves equal to 20.7).

4.4.3 Evaluating simplifying methods

Reduced Error Pruning and Reduced Error Grafting have been both implemented in KDB2000 and they have been empirically evaluated on 10 datasets, namely Abalone, Auto-mpg, Auto-price, Bank8FM, Cleveland, Housing, Machine CPU, Pyrimidines, Triazines and Wisconsin Cancer. All these datasets have been already used as benchmarks in the empirical evaluation of un-pruned model trees built with SMOTI.

Each dataset is analyzed by means of the same 10-fold cross-validation adopted in evaluating un-pruned trees. For each trial, the training set is, in turn, partitioned into growing (70%) and pruning set (30%). SMOTI is trained on the growing set, pruned on the pruning set and tested on the test set. Comparison is based on both average root mean square error (*Avg.RMSE*) made on the test sets and average number of leaves (*Avg.Leaves*). The stopping criteria used in the experimentation are fixed as follows: the significance level a used in the F-test is set to 0.075, the minimum number of cases falling in each internal node must be greater than the square root of the number of cases in the entire training set, the error in each internal node must be greater than the 0.01% of the error in the root node, the coefficient of determination in each internal node must be below 0.90 for model trees induced on the entire training set and 0.99 for model trees induced on the growing set and after simplified by means of REP or REG method.

TABLE 4.6: SMOTI (REP vs. REG) and M5' (PEP): results on accuracy (Avg.RMSE) of the induced/simplified model trees.

Dataset	**SMOTI un-pruned trees**		**SMOTI pruned trees**		**M5'**		
	Tree on training set	**Tree on growing set**	**REP**	**REG**	**Un-pruned trees (F=0)**	**Pruned trees (F=1)**	**Pruned trees (F=2)**
Abalone	2.5364	6.724	2.185	2.179	2.7724	2.180	2.126
AutoMpg	3.1493	4.4866	3.5633	3.7436	3.2010	2.969	2.835
AutoPrice	2246.0	2481.7	2746.3	2890.4	2358.8	2279.2	2390.1
Bank8FM	0.0383	0.0427	0.035	0.034	0.0409	0.0332	0.0319
Cleveland	1.3160	1.521	0.914	0.934	1.2496	0.9498	0.9028
Housing	3.58	5.717	4.080	3.912	4.2792	3.953	3.815
Machine-CPU	55.314	71.699	70.953	69.145	57.352	56.039	58.341
Pyrimidines	0.1056	0.1872	0.1034	0.1352	0.0927	0.0883	0.0864
Triazines	0.2017	0.1820	0.155	0.229	0.155	0.130	0.131
Wisconsin Cancer	51.413	72.376	33.464	37.455	45.406	37.062	34.397

TABLE 4.7: SMOTI (REP vs. REG) and M5' (PEP): results on size (Avg.Leaves) of the induced/simplified model trees.

Dataset	**SMOTI un-pruned trees**		**SMOTI pruned trees**		**M5'**		
	Tree on training set	**Tree on growing set**	**REP**	**REG**	**Un-pruned trees (F=0)**	**Pruned trees (F=1)**	**Pruned trees (F=2)**
Abalone	143	95.6	5.4	25.4	281.4	24.9	11
AutoMpg	13.7	19.2	3.1	8.5	22.6	7.6	4.6
AutoPrice	4.3	8	1.6	4.1	12.4	3.1	1.6
Bank8FM	2.2	68.8	5.6	30.2	417.7	63.4	27
Cleveland	21.7	17.3	2.3	5.2	28.1	3.4	1.6
Housing	8.8	19.6	3.1	7.6	50.7	23.5	14.5
MachineCPU	4.0	6	2.7	2.4	12	6.4	3.8
Pyrimidines	3.8	6.4	1.8	1.8	3.4	3.2	3
Triazines	16.6	13.3	1.2	3.8	9.1	5.3	3.5
Wisconsin Cancer	18.4	11.5	1.2	1.9	32.1	8	2.7

Experimental results are listed in Tables 4.6 and 4.7 which report *Avg.RMSE* and *Avg.Leaves* respectively of (un-pruned/pruned) SMOTI trees built on training/growing set. For comparison purposes, results obtained by M5'[11] are reported as well. They show that simplifying SMOTI trees is generally beneficial since REP and REG decrease the *Avg.RMSE* of SMOTI trees built on the growing set. The two methods drastically reduce the size of the induced trees, often of an order of magnitude, although REG tends to be more conservative than REP. The pruning method implemented in M5' outperforms both REP and REG in most data sets. However, the worst performance of REP and REG can be justified if we consider that M5' pruned a model tree which was originally more accurate than that pruned by REP and REG because of the full use of the cases in the training set.

This result is similar to that reported in [EMS97] for decision trees. Even in that case, it was observed that methods requiring an independent pruning set are at a disadvantage. This is due to the fact that the set of pre-classified cases is limited and, if part of the set is put aside for pruning, it cannot be used to grow a more accurate tree. A clear example is represented by the Auto-Price dataset, where the average number of leaves of REP and M5'(F=2) is the same (1.6) while the accuracy is different.

[11] M5' pruning [Qui92] [Qui93b] is based on a pessimistic error pruning-like strategy that compares the error estimates obtained by pruning a node or not. The error estimate is based on same cases used in training step and corrected in order to take into account the complexity of the model in the node. The compensation factor $F = 0$ when pruning is disabled. Conversely, either $F = 1$ or $F = 2$ in case of pruning.

A different view of results is offered in Table 4.8, which reports a percentage of the *Avg.RMSE* made by pruned trees on the test sets with respect to the *Avg.RMSE* made by un-pruned trees on the same testing sets. The table emphasizes the gain of the use of pruning. In particular, pruning is beneficial when the value is less than 100%, while it is not when the value is greater than 100%. Results reported confirm that pruning is beneficial for nine out of ten datasets. Moreover, the absolute difference of *Avg.RMSE* for REP and REG is below 5% in seven datasets. Finally, it is worthwhile to notice that the gain of REP and REG is better than the corresponding gain of M5' pruning method (F=2) in six datasets. This induces to hypothesize that the better absolute performances of M5' are mainly due to the fact that the tree to be pruned is more accurate because of the full use of training cases.

TABLE 4.8: Percentage of the *Avg.RMSE* for pruned trees w.r.t. the *Avg.RMSE* of un-pruned trees. *Avg.RMSE* is computed on the testing sets. Best values are in bold.

Dataset	**REP/un-pruned SMOTI**	**REG/un-pruned SMOTI**	**Pruned M5'(F=1)/ un-pruned M5'**	**Pruned M5'(F=2)/ un-pruned M5'**
Abalone	32.49554%	32.40631%	78.66124%	76.684461%
AutoMpg	79.42094%	83.43958%	92.75802%	88.566073%
AutoPrice	110.662%	116.4685%	96.62473%	101.326946%
Bank8FM	81.96721%	79.62529%	81.21493%	77.995110%
Cleveland	60.09204%	61.40697%	76.0129%	72.247119%
Housing	71.3661%	68.4275%	92.38422%	89.152178%
MachineCPU	98.95954%	96.43789%	97.71071%	101.724439%
Pyrimidines	55.23504%	72.22222%	95.19345%	93.203883%
Triazines	85.16484%	125.8242%	84.33206%	84.516129%
Wisconsin Cancer	46.23632%	51.75058%	81.62292%	75.754306%

4.5 Conclusions

TDIMT methods generally grow a tree structure in two phases. In the first splitting phase, leaf nodes are expanded and associated with split tests. In the second predictive phase, leaf nodes are labeled with a multiple (linear) model. One drawback with this tree-building strategy is that the choice of the split tests is often made independently of the type of model associated with the leaves. This could result in a model tree that does not capture the underlying data model, even in very simple cases that can be perfectly represented by a model tree. To overcome this problem, one of the TDIMT methods reported in the literature merges the two phases and chooses the best split test on the basis of the best multiple linear regression model associable to the leaves. Although correct, this approach considers only full regression models, while in statistics it is well known that models based on subsets may give more precise results than will models based on more variables. This is due to

the problem of collinearity. On the other hand, finding the best subset of variables while choosing the best split becomes too costly when applied to large data sets, since it may require the computation of a high number of multiple linear regression models.

In this chapter, we have presented a new TDIMT method, SMOTI, which integrates the splitting phase and the predictive phase. Specifically, model trees generated by SMOTI include two types of nodes: regression nodes and splitting nodes. The former are associated with straight-line regression, while the latter are associated with split tests. Both types of nodes are considered at the same level during the tree construction process. This allows SMOTI to build the model tree stepwise and to overcome the computational problem of testing a large number of multiple linear regression models, while choosing the best split test with respect to the best multiple linear regression model at the leaves. In addition, this approach potentially solves the problem of modeling phenomena, where some variables have a global effect while others have only a local effect. Indeed, variables of the regression nodes selected at higher levels in the tree have a "global" effect, since they affect several multiple models associated with the leaves.

A comparison with two TDMTI systems, namely M5' and RETIS, has been reported for laboratory-sized data sets. It proves that SMOTI can induce more accurate model trees, when both global and local behaviors are mixed in the underlying model. However, computation time of SMOTI is quadratic in the training set size, while is it linear for both M5' and RETIS. The low computation time of M5' can be explained by the more efficient TDIMT strategy (i.e., the split tests is chosen independently of the linear model associated with the leaves). Unfortunately, no clear justification can be given for RETIS efficiency, which is at variance with our computational complexity analysis.

The comparison has been extended to fourteen benchmark datasets typically used to test regression tree induction algorithms. In this second experimentation, SMOTI clearly outperforms RETIS in accuracy, while it is not possible to draw statistically significant conclusions on the comparison with M5'. Model trees induced by SMOTI are generally simpler and can more easily be interpreted than those generated by M5'. The interesting aspect of this second experimentation is that for some datasets, SMOTI detected the presence of global effects that no previous study on model trees has ever revealed.

The experimental results reported in this work are necessarily limited and do not include some important research tendencies. First, how model trees induced by SMOTI compare to other approaches, such as neural networks. Obviously, model trees offer some advantages over neural networks, both computationally (no repetitive data feeding to converge toward a solution) and with respect to usability (the user is not forced to make guesses about the structure of the network to obtain accurate results). However, the neural networks can partition the feature space into irregular regions (e.g., ellipsoids), while model trees perform axis-parallel partitioning (and oblique partitioning when continuous splitting nodes are descendants of regression nodes). The hierarchical mixture-of-experts architecture presents some interesting similarities with SMOTI [JJ94]. The comparison can also be extended

to support vector machines, which can be used for regression problems as well [MM00].

The second important research direction is the application of model trees induced by SMOTI to classification problems, as suggested by Frank et al. [FWI+98]. In this case, SMOTI can be used to predict class probabilities, and by learning multiple regression models instead of multiple linear models for each node, it would be possible to overcome the problem of building a separate tree for each class.

Similar to many decision tree induction algorithms, SMOTI may generate model trees that overfit training data. Therefore, a third research direction is the *a posteriori* simplification of model trees with both regression nodes and splitting nodes. Two simplification methods, named REP and REG, have been defined to simplify SMOTI trees. They are based on both pruning and grafting operators which require an independent pruning set. Some experimental results have been reported on the pruning methods and show that pruning is generally beneficial. Moreover, the comparison with another well-known TDIMT method, namely M5', which uses the training data both for growing and for pruning the tree, has shown that putting aside some data for pruning can lead to worse results. As future work, an extension of the MDL-based pruning strategies developed for regression trees [RK98] to the case of model trees with splitting and regression nodes could be also an interesting research direction, since MDL-based pruning algorithms do not use an independent pruning set, which can be a problem when the dataset is small.

Chapter 5

Upgrading SMOTI to multi-relational setting

Model trees are among the most popular typology of regression models and many algorithms to mine them have been developed in the past twenty years in statistics, machine learning as well as data mining. The popularity and success of these methods is not surprising, as model trees are known to be well accepted by data miners. Indeed, they are easily understood and can be easily interpreted by domain experts due to the symbolic representation of the regression surface. Moreover, sophisticated simplification techniques have been developed for dealing with noise in data.

Nevertheless, model trees family generally suffers from some limitations due to the restrictive attribute-value data representation language. Indeed, according to single table assumption, training data must be represented as fixed-length vectors of variable values where each variable has only a single, primitive value. As a consequence, aspects of internal data structure cannot be described and the induced trees cannot refer to such structural properties. This seriously compromises the application of model trees in real-world domains such as biology, chemistry or geo-referenced data analysis, where the structure of the subjects of study (i.e. units of analysis) is of central importance.

This problem also occurs with SMOTI. Therefore, in the remainder of this chapter, we present a multi-relational TDIMT method, named Mr-SMOTI (Multi-Relational Stepwise Model Tree Induction), which removes single-table limitation by enhancing the representational capabilities of SMOTI model trees mining and extending it to multi-relational representation. Mr-SMOTI is able to mine *multi-relational model trees* from a *tightly-coupled relational database* by exploring not only intra-tuple relationships among attributes but also inter-table relationships among tuples. In this case, tight-coupling guarantees a direct and uniform access to both data and patterns stored in databases without any pre-processing step as well as a proper exploitation of information embedded in the database schema to drive the mining process.

The induced model tree is a multi-relational pattern graphically represented by

means of a set of regression selection graphs, which can be translated into SQL statements (or equivalently first order expressions) and stored in XML format.

Mr-SMOTI algorithm is designed according to an *upgrading* strategy that starts from the propositional method SMOTI and turns it into a relational miner by devising suitable extensions of both the representation language and the associated algorithm.

Upgrading is conservative, therefore Mr-SMOTI has peculiarities similar to those already observed in the propositional version SMOTI. Mr-SMOTI model trees are characterized by two types of internal nodes that is regression nodes and splitting nodes. Hence, both *local* and *global* effect of a variable can be adequately captured in the regression model that is being stepwise built. The evaluation function is coherently defined with respect to the model tree being built and collinearity problems are eventually solved without any additional effort due to the stepwise construction.

Variables involved in both types of nodes may correspond with attributes of several tables in the relational database. The joining of these tables is dynamically determined on the basis of a foreign key path in database schema and aims at mining a predictive model for a target continuous field, which eventually involves multiple attributes from several tables. Search proceeds by alternating structural pattern generation and its attribute-value transformation to generate features which are evaluated for possible inclusion in the multi-relational regression model.

5.1 Background and motivations

In propositional setting regression data is stored in a single table (relation), where each row (tuple) corresponds to a unit of analysis (i.e. individual to be mined), while each column corresponds to either a predictor variable or the target numerical variable. Generally, only intra-tuple relationships between variables are found, while inter-tuple relationships are not considered and inter-table relationships between several tuples of distinct tables are not even explorable. Disregarding inter-table relationships can be a severe limitation in real-world application involving the prediction of continuous values from data that are naturally organized in a relational model involving several tables (multi-relational data model).

Multi-relational regression fulfills the need of adequately representing and treating the structure of units of analysis when they are composed by several units of observation that is a single target object, zero, one or more target relevant objects as well as the relationships among them.

From a database perspective, both target objects and target relevant objects are naturally stored in multiple tables $T_1, \ldots, T_v$ of a training relational database D. Relationships among tables are implicitly modeled by foreign key paths. The target table contains the target attribute corresponding to the continuous property to be predicted taking the relational structure of training units of analysis into account. This means that mined prediction models (e.g. model trees or regression rules) are typically multi-relational patterns since they may involve multiple tables from D.

An example of multi-relational regression rule is given in Table 5.1. The rule

TABLE 5.1: A relational regression rule expressed as SQL query (top) and Prolog program (down)

```
SELECT t1.Id, 12.7 as CreditLine
FROM Customer t1, Order t2
WHERE t1.Id = t2.Client AND t1. Sale ≤ 3
```

```
creditLine(Id, Y) :-
customer(IdCustomer,Sale,_,_), Sale≤3, order(_,_,IdCustomer), Y is 12.7, !.
```

---

involves two tables from the CustomerDB database[1], namely "Customer" (target table) and "Order" and predicts the credit line (target attribute) of a customer with number of sales greater than three, which have performed at least one order. Obviously, this rule supposes the existence of a foreign key path between Customer and Order in CustomerDB relational data model.

The multi-relational regression problem can be formally defined as follows:
Given:

- a training set $\boldsymbol{O}$ stored into v relations $S = \{T_0, T_1, \ldots, T_v\}$ of a relational dataset D,
- a set of v primary key constraints PK on relations in S,
- a set of w foreign key constraints FK on relations in S,
- a target relation $T(X_1, \ldots, X_n, Y) \in S$,
- a target continuous attribute Y in T, different from the primary key or foreign key in T.

Finds: a multi-relational regression model which predicts the value of Y for a structured individual that is described by means of a single tuple in T (with possibly UNKNOWN value for Y) and zero, one or more related tuples in S according to foreign key paths.

The problem of mining multi-relational regression models over data residing in multiple relations is not novel. It has been tackled in [Dž95], where the multi-relational Regression problem has been formalized in the normal ILP setting.

Thus far, two approaches have been proposed in ILP to directly solve multi-relational Regression problems in their original forms . The former uses a *separate-and-conquer* (or sequential covering) strategy to build a set of Prolog clauses (e.g. FORS and FFOIL). The latter uses a *divide-and-conquer* strategy to induce tree-based models and then translate these models into Prolog programs (e.g. SRT, S-CART and TILDE-RT)[2].

[1] The schema of the CustomerDB relational database is reported in Figure 2.3.
[2] An overview on multi-relational regression methods is presented in Section 3.7.

All these methods reserve little attention to data stored in relational database and to how knowledge of a data model can help to guide the search process [KBSV99]. They are based on main-memory data stored as Prolog facts, but in real-world applications these facts really correspond to tuples stored in tables of a relational database. Therefore, some pre-processing is required to transform tuples in facts. On the other hand, main-memory data processing guarantees in high performance for computationally intensive processes when enough memory is available to store all necessary data, but most data mining algorithms are characterized by frequent access to data that satisfies some selection conditions. This suggests that for data intensive processes it may be useful to exploit powerful mechanisms for accessing, filtering and indexing data, such as those available in database management systems (DBMS).

To provide functionalities to directly navigate a relational data structure and generate potentially new forms of evidence not readily available in a flattened single table representation, we follow the suggestion given in [KBSV99] of combining achievements of KDD field on the integration of data mining with database systems, with some results reported in the ILP field on how to correctly upgrade propositional data mining algorithms toward multi-relational representations. We propose Mr-SMOTI that is a prototypical multi-relational data mining method that upgrades SMOTI algorithm toward multi-relational representations and takes advantage from a tight integration with database systems i.e., Oracle® 9*i*.

From an inductive database perspective, this tight-coupling also aims at supporting a direct and uniform access to both data and patterns stored in databases. It guarantees the applicability of data mining algorithms to large datasets and the possibility to directly specify which data stored in a database have to be mined without any pre-processing. In this way, data model knowledge (e.g. foreign key path) is available from database schema free of charge.

Furthermore, Mr-SMOTI, similarly to the propositional SMOTI, induces model trees that contain both regression nodes which perform only straight-line regression and splitting nodes which partition the feature space. The model associated with each leaf is then the composition of the straight-line regressions reported along the path from the root to the leaf. Internal regression nodes contribute to the definition of multiple models and capture global effects, while straight-line regressions at leaves can only capture local effects.

5.2 Multi-relational regression framework

A multi-relational regression framework is based on the search for interesting predicting patterns in a relational database, where multi-relational patterns are pieces of substructures encountered in the structure of the units of analysis [KBSV99] such that not only attribute-value descriptions are considered, but also the structural information which is available through the associations between tables.

Definition 5.1 *A unit of analysis is covered by a multi-relational pattern iff the substructure described by a multi-relational pattern, in terms of both attribute-value*

conditions and structural conditions, occurs at least once in the unit of analysis [KBSV99].

♦

Multi-relational patterns can be viewed as subsets of units of analysis (multi-relational individuals to be mined) from database having some property. The most interesting subsets are chosen according to some measure (e.g. *MSE* for regression tasks or information gain for classification tasks), that guides the search in the space of all patterns.

In this section, we sketch the main idea of multi-relational model trees where each node is properly associated with a multi-relational pattern and discuss the special case of multi-relational patterns representing both splitting and regression nodes. We introduce the multi-relational pattern language of regression selection graphs that is an extension to regression tasks of the graphical language of selection graphs already proposed for classification purposes [KSV99] [Lei02] [ALH03]. Finally, we define a set of splitting and regression refinement operators for the regression selection graphs such that each pattern associated with either splitting or regression node of a multi-relational model tree can be expressed by means of regression selection graph and top-down induction of interesting regression patterns proceeds recursively applying such refinement operators to the best patterns.

5.2.1 Multi-relational model trees

Multi-relational model trees can be formally defined as follows:

Definition 5.2 *Given the multi-relational training data $\boldsymbol{O}$ stored into a set $S = \{T_1, \ldots, T_v\}$ of v relations of a relational dataset D, each unit of analysis $\boldsymbol{o} \in \boldsymbol{O}$ is described by both a single tuple stored into a target table $T \in D$ and zero, one or more tuples in the target relevant tables $T_i \in S$ ($T_i \neq T$) which are foreign key path associated with T in S. A binary multi-relational model tree τ can be built from D to predict the continuous target attribute Y in T such that τ is a binary model tree in which:*

1. *each node corresponds with a subset of individuals in $\boldsymbol{O}$ and it is associated with a portion of D intensionally described by a multi-relational pattern,*
2. *each variable that is eventually introduced in left branch of a node must not occur in the right branch of that node,*
3. *each leaf is associated with a (multiple) regression function which may involve variables representing attributes from several tables in D.*

♦

The root node is associated with the target table T. Hence, it contains the entire $\boldsymbol{O}$, but for each individual $\boldsymbol{o} \in \boldsymbol{O}$, only the attribute-value information in the target table is considered. Consequently, a splitting node t may either introduce a new test on a variable already included in the current description of individuals falling in

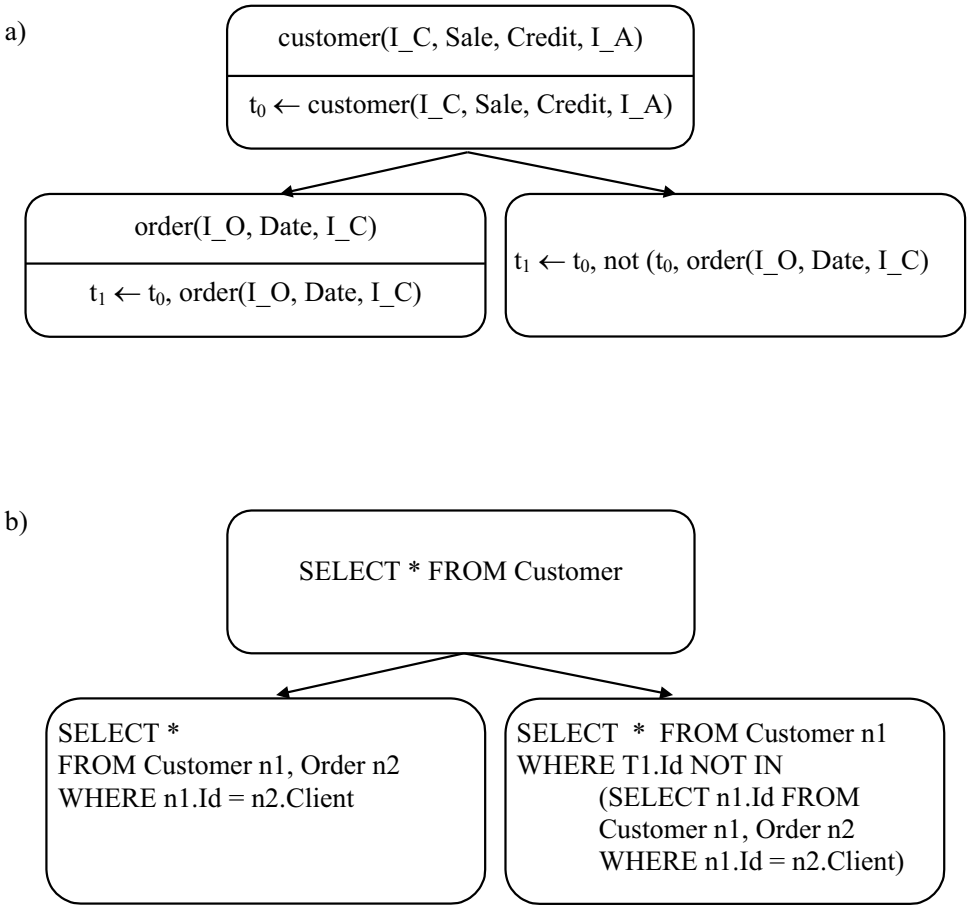


FIGURE 5.1: An example of multi-relational splitting node and queries added in a) a first order language and b) SQL.

$\boldsymbol{O}(t)$ or decide to test the existence or absence of tuples stored in T_i ($T_i \in S$) which are *foreign key path associated* with the target tuple in T according to a foreign key path of D between T and T_i. In both cases, a splitting node t passes down to each child only a sub-group of individuals falling in $\boldsymbol{O}(t)$. However, in the latter case the description of each individual $\boldsymbol{o} \in \boldsymbol{O}(t)$ that is passed down to the left child t_L is enriched with associated tuples in the new table involved in the test. In contrast, no structural modification occurs for each individual $\boldsymbol{o} \in \boldsymbol{O}(t)$ passed down to the right child t_R.

The subset $\boldsymbol{O}(t)$ can be intensionally described with a multi-relational pattern expressed as a conjunction of literals or an SQL query. More precisely, in the case D is represented as a Prolog program and each individual $\boldsymbol{o} \in \boldsymbol{O}$ is a set of facts (or interpretation), then a test at a splitting node t corresponds with checking whether the query "$\leftarrow Q_{t_L}$" to be associated with the left child t_L of t succeeds for any $\boldsymbol{o} \in \boldsymbol{O}(t)$, or not. The queries are defined in a way such that if the query associated with a leaf succeeds for $\boldsymbol{o}$, then the leaf contains the regression model to predict $\boldsymbol{o}(Y)$.

An important point is that the queries on the left and right child should be complementary. Let t_L and t_R be the left and right child respectively of t: for each individual $\boldsymbol{o} \in \boldsymbol{O}(t)$, exactly one of both queries (i.e. "$\leftarrow Q_{t_L}$" and "$\leftarrow Q_{t_R}$") should succeed. Now, if $test_t$ is the conjunction expressing the current test then "$\leftarrow Q_t, test_t$" and "$\leftarrow Q_t$, not $(Q_t, test_t)$" are complementary, but "$\leftarrow Q_t, test_t$" and "$\leftarrow Q_t$, not$(test_t)$", are not, when $test_t$ shares variables with Q_t (see Figure 5.1.a). This justifies the restriction in the second point of Definition 5.2 that directly follows from [Blo98], where the semantics of a multi-relational tree is extensively explained for classification tasks. A variable X that is introduced with a splitting test is existentially quantified within the conjunction at the node. The right subtree is only relevant when the conjunction fails (i.e. *there is no such* X), in which case further reference to X is meaningless.

If no recursive association among relations is expressed in the data model, then the query associated with each splitting node can be equivalently formulated in standard SQL [BD96] (see Figure 5.1.b).

When t is a leaf node ($t \in N_\tau^L$), the query Q_t also describes the regression function f_t to predict the (unknown) value of target attribute Y for an individual $\boldsymbol{o}$ such that Q_t succeeds for $\boldsymbol{o}$. In the simplest case, this regression function is a constant value simply computed as the mean (or median) of the Y values for all training individuals falling in $\boldsymbol{O}(t)$ [Blo98] [Kra96] [KW01].

The situation is more complex when a *multiple* regression function is associated with a leaf node $t \in N_\tau^L$. Indeed, f_t can be determined by considering not only the continuous variables corresponding with an attribute of the target table T but also continuous variables from the target relevant tables $T_i \in S$ ($T_i \neq T$) involved in left branches of a split along the path from the root to the current leaf. Since zero, one or *more* tuples belonging to a fixed target relevant table may be foreign key path associated with the same target tuple in T, this implies that a continuous variable, not included in the target table T, may assume multiple values for the same individual $\boldsymbol{o} \in \boldsymbol{O}(t)$.

A solution discussed in [ACM03] is to transform $\boldsymbol{O}(t)$ into an attribute-value set $\boldsymbol{O}_p(t)$ described by the $m+1$ single-value variables $X_1, \ldots, X_m$, Y, and then mine $\boldsymbol{O}_p(t)$ to build the prediction function $F_t : X_1 \times \ldots \times X_m \rightarrow Y$ (e.g F_t may be a multiple linear regression function whose regression coefficients are estimated according to least-square regression on $\boldsymbol{O}_p(t)$). The transformation from $\boldsymbol{O}(t)$ to $\boldsymbol{O}_p(t)$ can be actually performed by joining all tables involved in describing individuals $\boldsymbol{O}(t)$ with respect to tests along the path from the root to the current node. Since for each $\boldsymbol{o} \in \boldsymbol{O}(t)$, one or more attribute-values individuals $\boldsymbol{o}_{p_1}, \ldots, \boldsymbol{o}_{p_h}$ can be built according to this joining operation (p_t), the multi-relational regression function f_t, finally associated with the leaf node t, averages the prediction returned by F_t on the set of attribute-value individuals built by p_t.

Finally, in the case of multi-relational SMOTI trees, the definition of a multi-relational model tree must be further extended in order to support the *stepwise* construction of the multiple multi-relational regression function f_t associated with each leaf $t \in N_\tau^L$. Similarly to the propositional setting, a regression step corresponds with a regression node that performs straight-line regression on a numerical

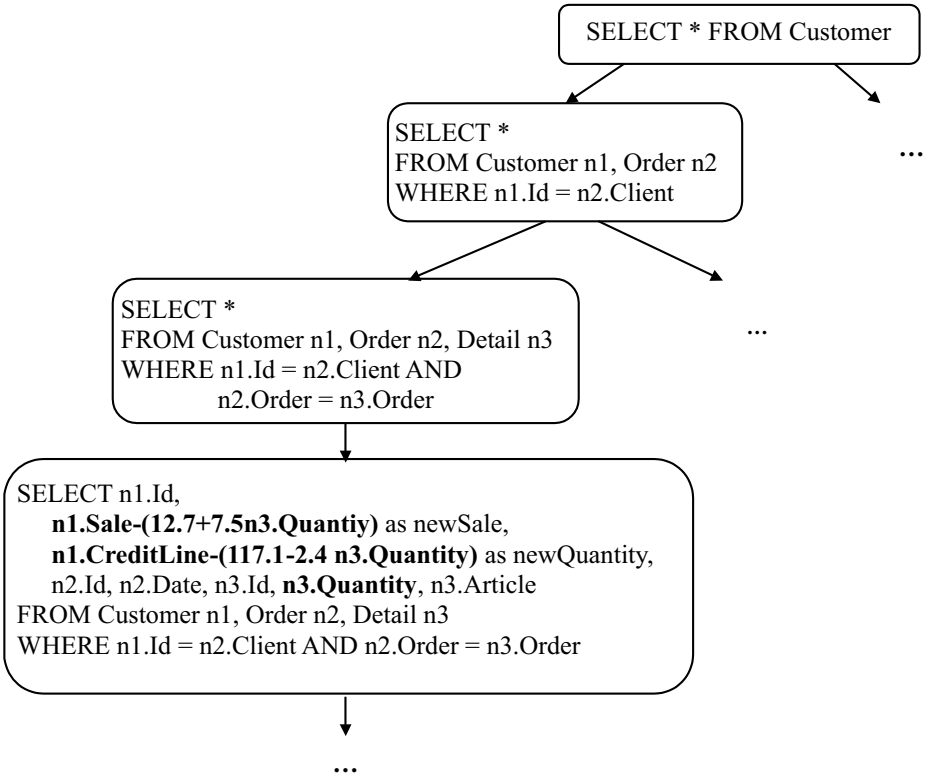


FIGURE 5.2: An example of a multi-relational regression node (i.e. n1.CreditLine = 12.7 + 7.5 n3.Quantity) and corresponding SQL query. All continuous variables not yet included in regression step are replaced with their residuals.

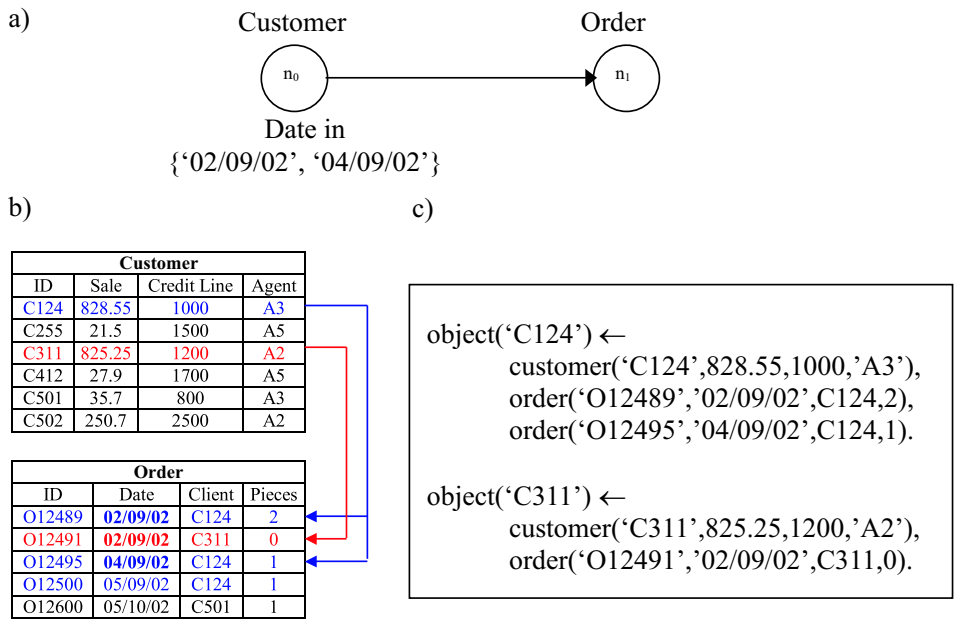
variable not yet included in the regression model. However, according to the multi-relational structure of $\boldsymbol{O}$, each variable involved into a regression step at t level, may belong either to the target table T or to a target relevant table T_i foreign key path associated with T, which appears in the left branch of a splitting node along the path from the root to t. Similarly to propositional case, a multi-relational regression node passes down to its unique child the entire set $\boldsymbol{O}(t)$, where each numerical variable involved in the description of $\boldsymbol{O}(t)$ not yet introduced in the regression model, is replaced with the corresponding residual computed according to attribute-value representation of individuals falling in $\boldsymbol{O}(t)$ (see Figure 5.2).

5.2.2 Regression selection graphs

The multi-relational pattern associated with each node t of a multi-relational model tree τ can be expressed in the graphical language of *regression selection graphs*. The classical definition of a selection graph is reported in [KBSV99] [KSV99] for the case of multi-relational decision trees. Nevertheless, we present here an extension of this definition in order to make the selection graphs more suitable for regression tasks.

Definition 5.3 *A regression selection graph G is a directed graph (N, A), such that:*

1. *N is a set of 4-tuples (T, C, R, s), namely regression selection nodes, where:*



Customer

ID	Sale	Credit Line	Agent
C124	828.55	1000	A3
C255	21.5	1500	A5
C311	825.25	1200	A2
C412	27.9	1700	A5
C501	35.7	800	A3
C502	250.7	2500	A2

Order

ID	Date	Client	Pieces
O12489	**02/09/02**	C124	2
O12491	**02/09/02**	C311	0
O12495	**04/09/02**	C124	1
O12500	05/09/02	C124	1
O12600	05/10/02	C501	1

FIGURE 5.3: a) An example of regression selection graph describing the set of customers, which have performed almost one order on either the 2nd September 2002 or the 4th September 2004, and a set of customers satisfying this regression selection graph described in terms of either (b) tuples stored in foreign key path associated tables of a relational database or (c) conjunction of first order ground atoms.

- *T is a table of a relational database D with both continuous and discrete attributes $A_1, \ldots, A_m$,*
- *R is the set $\{X_1, X_2, \ldots, X_m\}$ such that each X_i is either a residual of A_i[3] computed according to regression steps already performed or A_i itself,*
- *C is a set of conditions in the form "$T.X_i$ OP c", where X_i corresponds with the residual of A_i computed when the condition has been added. OP is one of the usual comparison operators ($<$, $\geq$, in, not in), while c is either a constant continuous value or a set of discrete values,*
- *s is a flag with possible values "open" or "closed";*

2. *A is a set of 4-tuples (p, q, fk, e), where:*

   - *p and q are regression selection nodes,*
   - *fk is a foreign key association between $p.T$ and $q.T$ (or vice-versa) in D,*
   - *e is a flag with possible values "present" or "absent".*

♦

Regression selection graphs can be graphically represented as directed labeled graphs (see Figure 5.3.a), which contain at least one regression selection node,

[3] X_i is properly the residual of A_i when A_i corresponds with either the dependent attribute or a continuous independent attribute of the table T, not yet included in already performed regression steps. Note that A_i is neither a primary key nor a foreign key of T in D.

namely the target node, that represents the target table T in D. The value of s is expressed by the absence or presence of a cross in the node, representing the *open* and *close* value, respectively. The value of e, in turn, is indicated by the presence (*absent* value) or absence (*present* value) of a cross on the corresponding arrow representing the labeled arc. The direction of the arrow (left-to-right and right-to-left) expresses the multiplicity of the association *fk* (one-to-many and many-to-one, respectively). Every arc between the nodes p and q imposes some constraints on how one or more tuples in the table $q.T$ are related to each tuple in the table $p.T$, according to the list of conditions in $q.C$. Indeed, the association between $p.T$ and $q.T$ induces some grouping (see Figure 5.3.b and 5.3.c) in the tuples of $q.T$, and thus selects some tuples of $p.T$. A *present* arc a from p to q selects those tuples which both belong to the *join* between $p.T$ and $q.T$ according to the foreign key constraint $a.fk$ and match the involved list of conditions ($p.C \wedge q.C$). Conversely, an *absent* arc a from p to q selects those tuples of $p.T$ which both satisfy the conditions in $p.C$ and have no joined tuples in $q.T$ that satisfy conditions in $q.C$. This suggests that the target objects explained (covered) by a regression selection graph G are those tuples $t_i \in n_0.T$ such that t_i satisfies the conjunction of conditions stored in $n_0.C$ and for each present (absent) arc $a_j \in G.A$ outgoing from the target node ($a_j.p = n_0$) there exists (or does not exist) some tuple $s_i \in a_j.q.T$ such that s_i is foreign key associated to t_i according to foreign key association instantiated by a_j and s_i covers the sub-graph rooted in $a_j.q$.

Regression selection graphs are typically considered more intuitive than expressions in SQL or Prolog [KBSV99], since they *graphically* reflect the structure of database schema (i.e. relational data model). Furthermore, whenever they are employed to represent multi-relational patterns associated with each node of a multi-relational SMOTI tree, refinement operators to perform either a splitting test or a regression test can be simply defined in terms of adding or updating arcs and/or nodes of the regression selection graph G in question. Finally, in the case no recursive relationships is modeled by G, it can be straightforwardly translated into an SQL query $Q(G)$.

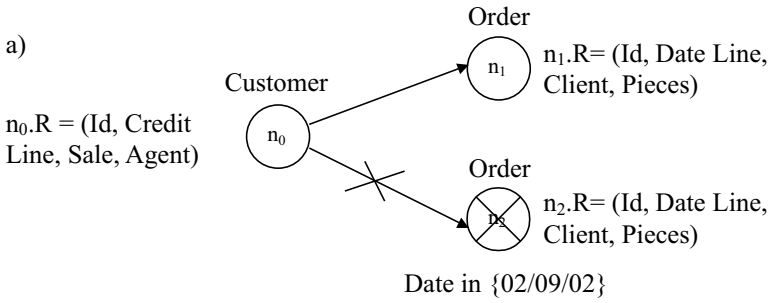
The top-level description of the translation of a regression selection graph into an SQL query is reported in Algorithm 5.1 that extends the procedure presented in [KBSV99] in order to translate regression steps too. The procedure *TranlateIntoSQL* produces a list of tables (*tablesList*), attributes (*attributeList*), join conditions (*joinList*) and conditions (*conditionList*), and combine these to produce an SQL query. More precisely, the translation procedure produces a join for all tables connected in a *present* arc path that does not involve any regression selection node belonging to some sub-graph of G that is the root in a closed regression selection node and pointed by an absent arc

By adopting this procedure, the regression selection graph in Figure 5.4 can be translated into an SQL query, where each sub-graph rooted in a closed node and pointed by an absent arc is translated into a negated inner sub-query.

Algorithm 5.1 Translation of a regression selection graph into an SQL query

`PROCEDURE TranslateIntoSQL(`G, Q`)`

a)



b)

SELECT n0.ID N0_Id, n0.Sale N0_Sale, n0.CreditLine N0_CreditLine, n0Agent N0_Agent, n1.ID n1_ID, n1.Date N1_Date, n1.Client N1_Client, n1.Pieces N1_Pieces
FROM Customer n0, Order n1
WHERE n0.ID=n1.Client and g0.ID **not in** (**SELECT** g2.Client **FROM** Order g2 **WHERE** g2.Date **in** {02/09/02}));

c)

← customer(N0_Id, N0_Sale, N0_CreditLine, N0_Agent),
order(N1_ID, N1_Date, N0_ID, N1_Pieces),
¬ (order(N2_ID, N2_Date, N0_ID, N2_Pieces), N2_Date = '02/09/02').

FIGURE 5.4: The regression selection graph corresponding with the set of customers which have performed no order on the 2nd September 2002 and its translation into both a) an SQL statement and b) a first order logic expression.

```
INPUT:
   G:  regression selection graph;
OUTPUT:
   Q:  SQL query;
BEGIN
   tableList = ∅; attributeList = ∅; joinList = ∅; conditionList = ∅;
   FOR each arc a in G.A DO
      IF (a.e = "present" and not_in_closed_subgraph(a.p, G)) THEN
         IF (a.right_node().s = "open") THEN joinList.add(a.fk);
         ELSE joinList.add(a.fk.left_join_attribute()+ " not in " +
               TranslateSubGraph(subgraph(G, a.right_node()),
               a.fk.right_join_attribute());
         END IF;
      END IF;
   END FOR;
   FOR each node n in G.N DO
      IF (n.s = "open" and not_in_closed_subgraph(n, G)) THEN
         tableList.add(n.T + alias(n.T));
         FOR each condition c in n.C DO
            conditionList.add(c);
         END FOR;
```

```
            FOR each attribute X in n.R DO
               attributeList.add(X + alias(n.R.X));
            END FOR;
         END IF;
      END FOR;
      Q = "SELECT " + attributeList + " FROM " + tableList + " WHERE " +
           joinList + " AND " + condList;
   END PROCEDURE;

   PROCEDURE TranslateSubGraph(G, K)
   INPUT:
      G: regression selection graph;
      K: attribute;
   OUTPUT:
      Q: SQL query;
   BEGIN
      tableList = ∅; joinList = ∅; conditionList = ∅;
      FOR each node n in G.N DO
         tableList.add (n.T + alias(g.T));
         FOR each condition c in n.C DO
            conditionList.add(c);
         END FOR;
      END FOR;
      FOR each arc a in G.A DO
         joinList.add (a.fk);
      END FOR;
      Q = "SELECT " + K + " FROM " + tableList + " WHERE " + joinList +
           " AND " + condList;
   END PROCEDURE;
```

5.2.3 Refinements of regression selection graphs

When regression selection graphs are adopted to represent multi-relational patterns associated with each (splitting, regression or leaf) node of a multi-relational model tree τ, growing τ corresponds with refining the regression selection graph associated with the current node t. This means that when a split is introduced in τ, we are in fact refining the regression selection graph associated with t in two ways (i.e. by introducing both a left child t_L and a right child t_R) in order to partition the current set of individuals falling in t according to a binary test condition. Conversely, if a regression step is performed, we are refining the current regression selection graph in order to remove the effect of performed regression according to the stepwise procedure.

In the case of multi-relational splitting tests, two splitting refinement operators of a regression selection graph G can be defined, namely "add condition" refinement and "add present arc and open node" refinement. Both these refinements can involve

only open regression selection nodes of G which do not belong to any sub-graph of G that is rooted into a closed regression selection node and pointed by an absent arc. We denote by $G.N_O$ the set of these regression selection nodes in G.

The *add condition* refinement of G adds a condition c to a regression selection node $n_i \in G.N_O$ without actually changing the structure of G. The condition c is a boolean test which involves a variable (or residual) $X \in n_i.R$ that does not correspond to neither a primary key nor a foreign key of the table $n_i.T$ in D. In continuous case, c is in the form $X \leq \alpha$, while in discrete case c is in the form $X \in \{x_1, \ldots, x_s\}$.

Since the add condition refinement is a splitting refinement, it has to be introduced together with its complementary refinement. We denote by G_L and G_R the refinement and complementary refinement respectively of a regression selection graph G corresponding with testing a binary condition c. The refinement G_R is the complement of G_L in the sense that it covers those individuals covered by the original regression selection graph G which are not covered by G_L: for each individual covered by G, (i.e. the query $Q(G)$ associated with G succeeds), it is covered by exactly one of both graphs G_L and G_R (i.e. exactly one of both queries $G(Q_L)$ and $G(Q_R)$ succeeds).

Knobbe and his colleagues [KSV99] have proposed a complementary refinement named *add negative condition* that should solve the problem of mutual exclusion between an add condition refinement and its complement. When the node to be refined is the target node $n_0 \in G.N$, the add negative condition refinement simply refines the regression selection graph G by adding the negated condition $\neg c$ to the condition list $n_0.C$. Conversely, when the regression selection node $n_i \in G.N_O$ to be refined is not the target node ($n_i \neq n_0$), the add negative condition refinement G_R is built from G by introducing an absent arc from the parent of n_i to the clone of the entire sub-graph of G that is rooted in n_i. The introduced sub-graph has a root (a clone of the node to be refined) which is a closed node updated with the refinement condition that is not negated. In this way, the complementary operation builds a regression selection graph that negates an entire inner sub-query and not simply a condition. This is an important difference compared with the propositional case in which a test condition and its simple negation generate a partitioning of the training data.

However, this complementary refinement may fail when the node to be refined by adding the condition c is *not directly* connected with target node.

Example 5.1 *Let us consider the regression selection graph G in Figure 5.5.a that can be easily translated into the SQL query $Q(G)$ as shown in Figure 5.5.b. G is covered by all customers which have performed almost one order with one or more details ("object('C124')" and "object('C501')" in Figure 5.5c). We may decide to refine G by adding the condition "Quantity $\leq$ 22" on the list of conditions of the open node n_2, where $n_2.T$ = "Detail". Note that n_2 is not directly connected with target node n_0 in G.*

The resulting add condition refined regression selection graph G_L (see Figure 5.6) covers all customers which have performed almost one order with one or more de-

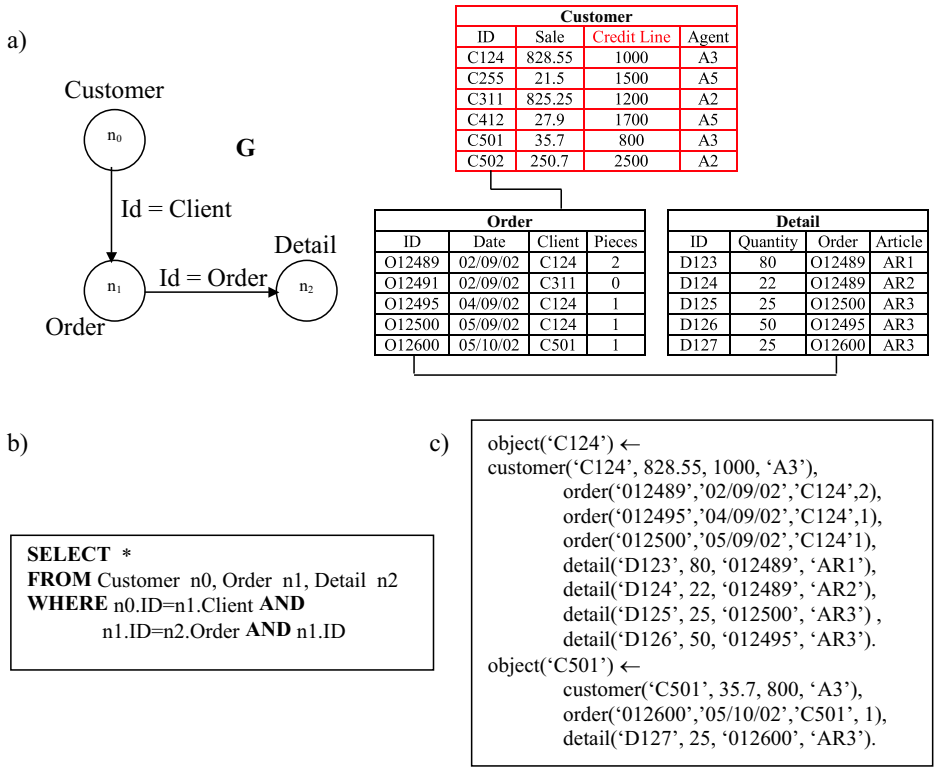


FIGURE 5.5: An example of (a) a regression section graph G, (b) its translation into SQL and (c) the set of objects covered by G.

tails having quantity greater than twenty-two (only "object('C124')" in Figure 5.6). Intuitively, the complementary regression selection graph G_R should cover all customers which have performed almost one order with one or more details, but no one of these orders has some detail with quantity less than twenty-two ("object('C501')"). Anyway, this does not occur if G_R is properly built by following the procedure proposed by Knobbe and his colleagues that suggest to complement G_L by introducing in G an absent arc from the parent (n_1) of the regression selection node in question (n_2) to a new closed node n_3 that is the clone of n_2. The condition list of n_2 is then copied into n_3 and extended by the new condition "Quantity $\leq$ 22". Adopting this construction, the complementary refinement G_R is covered by some individuals already covered by G_L (in Figure 5.7 G_R is covered by "object('C501')" but also by "object('C124')").

♦

The Example 5.1 confirms the intuition of Leiva [Lei02] by proving that the add negative condition mechanism proposed by Knobbe and his colleagues may lead to a regression selection graph G_L and a complementary regression selection graph G_R which are not mutually exclusive. To guarantee the mutual exclusion also when G_L is obtained by adding the condition c to a regression selection node of G that is not directly connected with target node, G_R must be actually built from G by adding an absent arc from the target regression selection node n_0 to the clone of

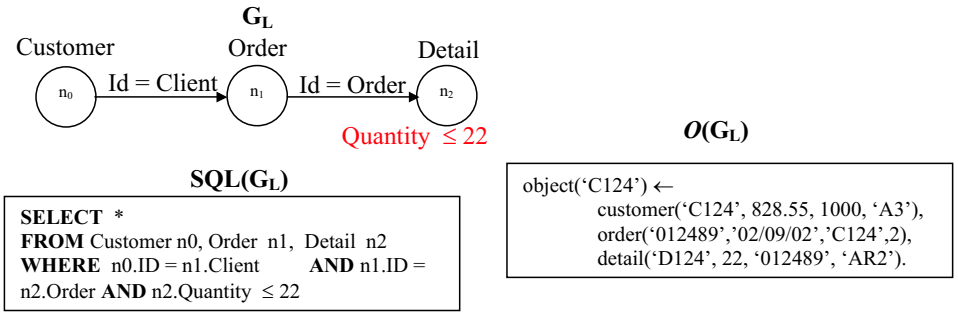


FIGURE 5.6: An example of add condition refinement for the regression selection graph G described in Figure 5.5.a.

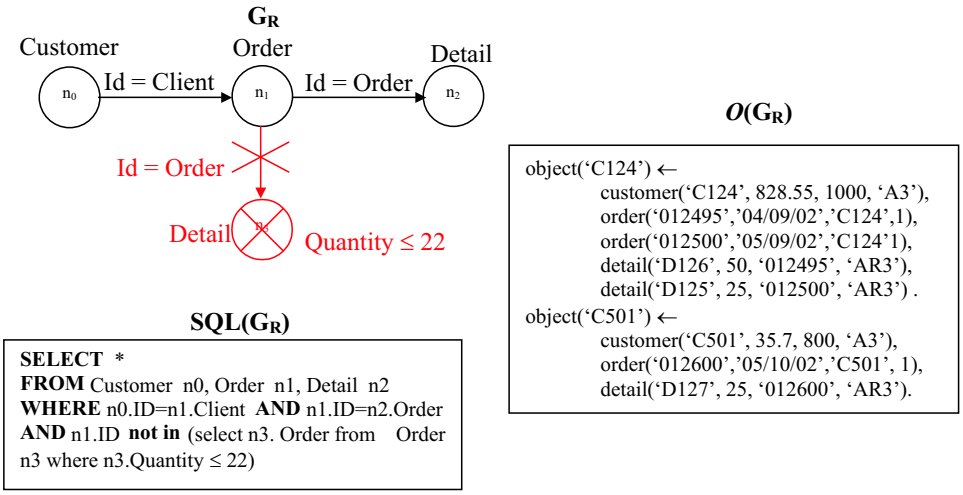
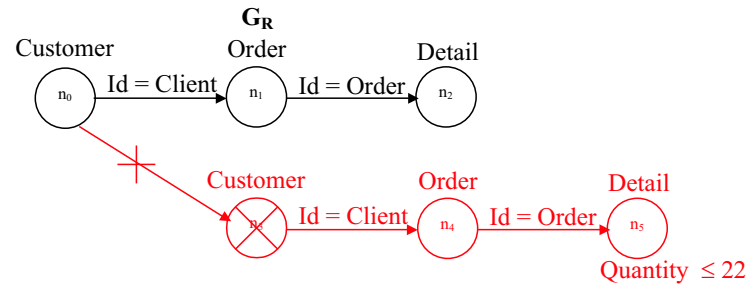


FIGURE 5.7: The add negative condition refinement for the regression selection graph G described in Figure 5.5.a. G_R is here built following the procedure proposed by Knobbe and his colleagues that could not satisfy the mutual exclusion requirement in the case the regression selection node in question is not directly connected with target node.

the minimal sub-graph in G containing the foreign key path from the target node to the node to be refined. This is coherent with the semantics of a first-order splitting test formalized by [Blo98] [BD98] for first-order decision trees. The introduced sub-graph has a root (a clone of n_0) that is a *closed* node and it is updated with the refinement condition that is not negated. A new *absent* arc is also introduced between the target node and its closed clone. This arc is an instance of the implicit relationship between the primary key of the target table and the own itself (see Figure 5.8).

The *add present arc and open node* refinement instantiates a foreign key association in form of a present arc and then add it to G together with its corresponding table in a new open regression selection node. The complementary refinement introduces an absent arc together with its corresponding table. This is the *add absent arc and closed node* refinement proposed in [KBSV99]. However, also for this refinement we have to deal with the special case of a node to be added that is not



SQL(G_R)

```
SELECT *
FROM Customer n0, Order n1, Detail n2
WHERE n0.ID=n1.Client AND
      n1.ID=g2.Order AND
      n0.ID not in ( select n3.ID
      from Customer n3, Order n4, Detail n5
      where n3.ID=n4.Client and
      n4.ID=n5.Order and n5.Quantity ≤ 22)
```

***O*(G_R)**

```
object('C501') ←
        customer('C501', 35.7, 800, 'A3'),
        order('012600','05/10/02','C501', 1),
        detail('D127', 25, '012600', 'AR3').
```

FIGURE 5.8: The correct add negative condition refinement for the regression selection graph G described in Figure 5.5.a. This complementary refinement satisfies the mutual exclusion requirement.

directly connected with the target node. Similar considerations to those we have drawn above for complementing the add condition refinement are valid here. This means that when n_j, the node to be added in forming G_L, has to be directly connected with a node $n_i \in G.N_O$ (i.e. there will exist an arc $a \in G_L.A$ such that $a.p = n_i$ and $a.q = n_j$) such that n_i is not target node in G ($n_i \neq n_0$), the complementary refinement G_R is obtained from G by adding an absent arc from the target node n_0 to a new sub-graph rooted in a closed node that is the clone of the minimal sub-graph in G_L connecting the target node with the added node (see Figure 5.9).

In addition to splitting refinements, we define a *regression refinement* of a regression selection graph G that corresponds with performing a regression step on the tuples generated by running the SQL query $Q(G)$ associated with G. Let us consider a generic regression step that is a straight-line regression in the form:

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X,$$

where Y is the residual target attribute as it appears in $n_0.R$ of G, while X is the residual of a continuous attribute (neither primary key nor foreign key in data model) not yet included in previous regression refinement to obtain G (i.e. it is not involved in forming the current Y residual). The residual X belongs to the list of residuals R of a regression selection node $n_i \in G.N_O$. Since the stepwise construction of a multiple regression model imposes that, for each regression selection node $n_j \in G.N_O$, the regression step removes the effect of performed regression on X from all residual $X_i \in n_j.R$ ($X_i \neq X$), we consider a further set of straight-line regressions in the form:

$$\hat{X}_i = \hat{\beta}_{ij_0} + \hat{\beta}_{ij_1} X,$$

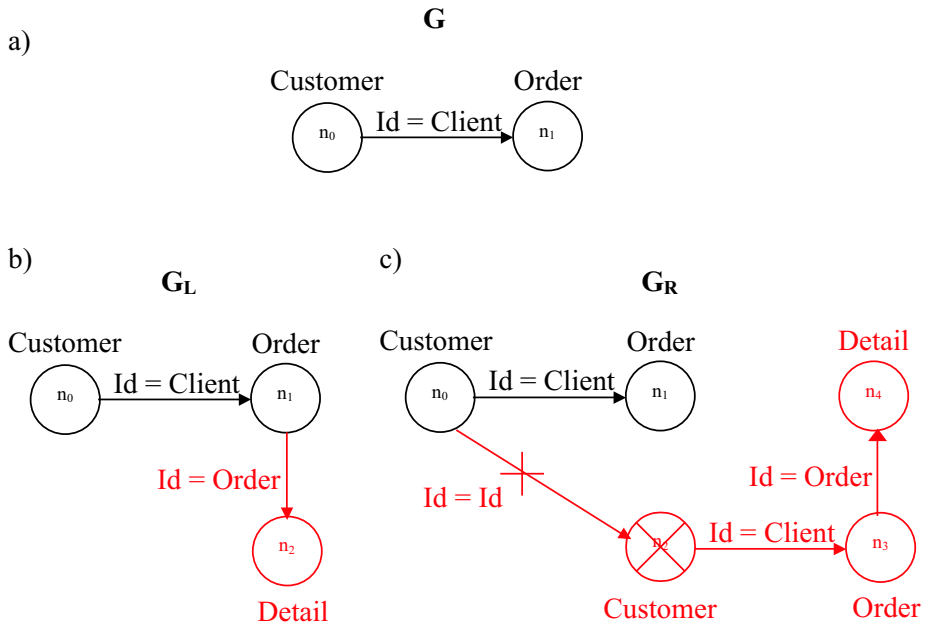


FIGURE 5.9: An example of a) a regression selection graph G, b) an add present arc and open node refinement G_L of G and c) its complementary refinement G_R.

where X_i corresponds with a continuous attribute (neither primary key nor foreign key in data model) of $n_j.T$ not yet involved in a regression refinement to obtain G.

The regression refinement of G corresponding with a regression step on a continuous residual X returns a regression selection graph G_R obtained by cloning G and updating the list of residuals for each node $n_j \in G.N_O$ by substituting X_i with $X_i - \beta_{ij_0} - \beta_{ij_1} X$. This means that the residual $Y \in n_0.R$ of the target attribute $n_0.T.Y$ is replaced with $Y - \hat{\beta}_0 + \hat{\beta}_1 X$, while the residual of discrete attributes, primary/foreign keys as well as continuous attributes involved in regression refinement already applied to obtain G, are simply replaced with the own itself (see Figure 5.10).

5.3 Mr-SMOTI

Mr-SMOTI stepwise mines multi-relational model trees with both splitting and regression nodes over data that resides in multiple tables of a tightly-coupled relational database. It represents the upgrade of the propositional miner SMOTI toward multi-relational according to a methodology similar to the one described in [VD01] to upgrade propositional miners toward first-order logic. This is coherent with the idea of developing multi-relational miners having approximately the existing propositional system as special case.

The first-order upgrading methodology represents the most important lesson learned during the development of some well known ILP systems such as TILDE [Blo98] to induce first-order decision trees, ICL [DW95] to learn first-order rules,

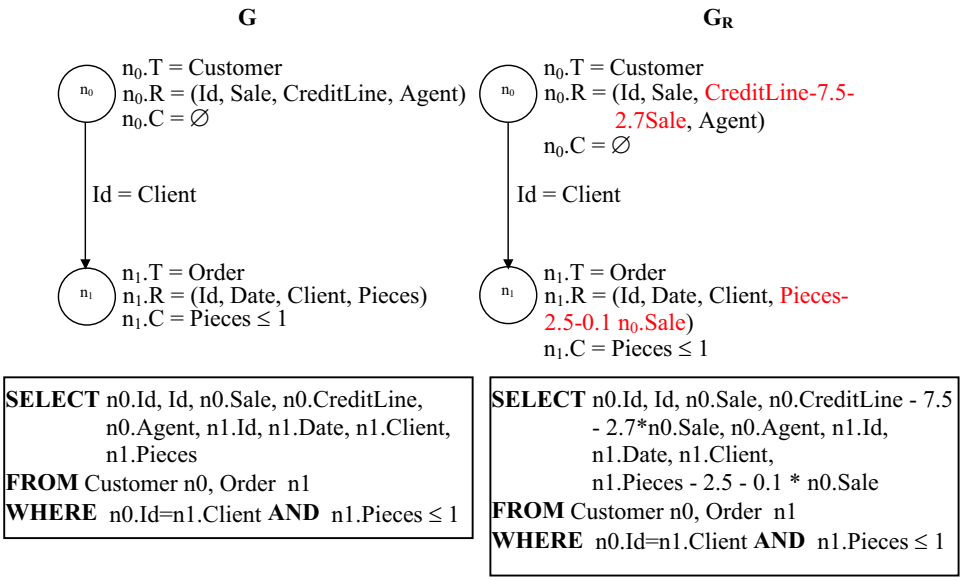


FIGURE 5.10: An example of a regression refinement G_R of a regression selection graph G corresponding with the regression step on $Credit\ line = 7.5 + 2.7Sale$ and $Pieces = 2.5 + 0.1Sale$.

CLAUDIEN [DD97] for clausal discovery and WARMR [De 97] to mine association rules in multiple relations. In all these cases, the first-order upgrading starts from an existing propositional miner and provides a recipe for upgrading it toward the use of first-order logic. This recipe involves the mining from units of analysis which are complex training individuals (examples) corresponding to sets of ground facts (interpretations), the adaptation of the representation of the hypotheses toward Prolog, the employment of a search operator (e.g. θ-substitution) to structure and explore the space of hypotheses as well as the introduction of a declarative bias. Moreover, it recycles as much as possible from the propositional system.

Starting from this methodology, we have extended it in order to take into account units of analysis which are described with tuples scattered over several tables of a relational database rather than ground atoms stored in a Prolog program. This suggests that multi-relational no recursive hypotheses can be naturally expressed as SQL statements rather than Prolog programs. The search operators to explore the space of hypotheses should correspond to specialization or generalization of SQL statement.

In the remaining of this Section, we present Mr-SMOTI algorithm describing details about the computation of the set of both multi-relational splitting and regression steps at each node as well as stopping criteria. Finally, the complexity analysis of Mr-SMOTI algorithm is evaluated.

5.3.1 The algorithm

Mr-SMOTI mining algorithm is based on a *divide-and-conquer* strategy starting with a root node t_0 that contains the entire set of $\boldsymbol{O}$ individuals stored in a relational database D. At root level t_0, each individual $\boldsymbol{o} \in \boldsymbol{O}(t_0)$ is simply described by the

corresponding tuple in the target table T. This means that in the graphical language of regression selection graphs, t_0 is associated with the regression selection graph G_0 that contains only the target node. The relational structure of D is then adequately exploited to choose from three different possibilities:

- growing τ by performing a splitting test on the current node t and introducing the nodes t_L (left child of t) and t_R (right child of t),
- growing τ by performing a regression step on the current node t and introducing its unique child t_R,
- stopping the tree's growth at the current node t.

Similarly to SMOTI, the validity of either a splitting test or a regression step at node t is based on two different heuristic functions $\sigma(t)$ and $\rho(t)$, respectively.

If t_L and t_R are the left and right child of a splitting node t, then:

$$\sigma(t) = \frac{n(t_L)}{n(t_L)+n(t_R)} R(t_L) + \frac{n(t_R)}{n(t_L)+n(t_R)} R(t_R), \tag{5.1}$$

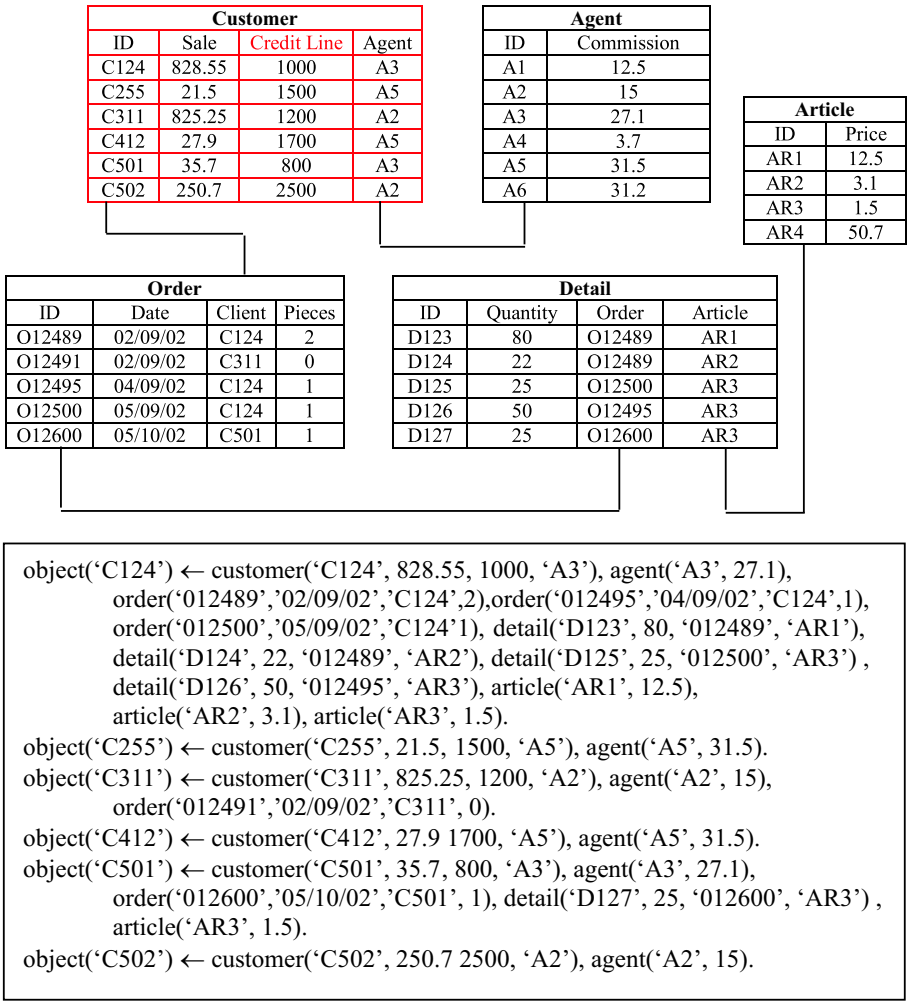
where $n(t_L)$ ($n(t_R)$) is the number of tuples obtained by molding the (multi-)relational description of training individuals falling in $\boldsymbol{O}(t_L)$ ($\boldsymbol{O}(t_R)$) in a single table representation $\boldsymbol{O}_P(t_L)$ ($\boldsymbol{O}_P$ (t_R)) derived by joining the portion of D actually involved in describing $\boldsymbol{O}(t_L)$ ($\boldsymbol{O}(t_R)$). $R(t_L)$ ($R(t_R)$) is the resubstitution error on the left (right) child, computed as follows:

$$R(t_L) = \sqrt{\frac{1}{n(t_L)} \sum_{i=1\ldots n(t_L)} (y_i - \widehat{y}_i)^2} (R(t_R) = \sqrt{\frac{1}{n(t_R)} \sum_{i=1\ldots n(t_R)} (y_i - \widehat{y}_i)^2}), \tag{5.2}$$

such that $\widehat{Y}$ is the (multi-)relational regression model built by combining the best straight-line regression associated to t_L (t_R), with all straight-line regressions introduced along the path from the root to t_L (t_R).

Example 5.2 *Let D in figure 5.11 be an instance of the relational database "CustomerDB", which represents a collection $\boldsymbol{O}$ of six customers. Each customer is described by properties (e.g. "Sale" and "Credit Line") which are stored in the target table "Customer" as well as properties stored in the target relevant tables "Agent", "Order", "Detail" and "Article". Relationships between each target object and corresponding target relevant objects are implicitly modeled by foreign key path associations in CustomerDB.*

We consider the case of a splitting node t at root level of a multi-relational tree τ such that t partitions customers according to the existence (or not) of any tuple in Order that is foreign key path associated with the target object stored in Customer (see Figure 5.12.a). Training individuals that satisfy the test (i.e. customers which have performed almost one order) are passed down to the left child t_L and they are described by considering tuples in both Customer table and Order table. Training individuals that do not satisfy the test (i.e. customers which have not performed



Customer

ID	Sale	Credit Line	Agent
C124	828.55	1000	A3
C255	21.5	1500	A5
C311	825.25	1200	A2
C412	27.9	1700	A5
C501	35.7	800	A3
C502	250.7	2500	A2

Agent

ID	Commission
A1	12.5
A2	15
A3	27.1
A4	3.7
A5	31.5
A6	31.2

Article

ID	Price
AR1	12.5
AR2	3.1
AR3	1.5
AR4	50.7

Order

ID	Date	Client	Pieces
O12489	02/09/02	C124	2
O12491	02/09/02	C311	0
O12495	04/09/02	C124	1
O12500	05/09/02	C124	1
O12600	05/10/02	C501	1

Detail

ID	Quantity	Order	Article
D123	80	O12489	AR1
D124	22	O12489	AR2
D125	25	O12500	AR3
D126	50	O12495	AR3
D127	25	O12600	AR3

object(‘C124’) ← customer(‘C124’, 828.55, 1000, ‘A3’), agent(‘A3’, 27.1),
order(‘012489’,’02/09/02’,’C124’,2),order(‘012495’,’04/09/02’,’C124’,1),
order(‘012500’,’05/09/02’,’C124’1), detail(‘D123’, 80, ‘012489’, ‘AR1’),
detail(‘D124’, 22, ‘012489’, ‘AR2’), detail(‘D125’, 25, ‘012500’, ‘AR3’) ,
detail(‘D126’, 50, ‘012495’, ‘AR3’), article(‘AR1’, 12.5),
article(‘AR2’, 3.1), article(‘AR3’, 1.5).
object(‘C255’) ← customer(‘C255’, 21.5, 1500, ‘A5’), agent(‘A5’, 31.5).
object(‘C311’) ← customer(‘C311’, 825.25, 1200, ‘A2’), agent(‘A2’, 15),
order(‘012491’,’02/09/02’,’C311’, 0).
object(‘C412’) ← customer(‘C412’, 27.9 1700, ‘A5’), agent(‘A5’, 31.5).
object(‘C501’) ← customer(‘C501’, 35.7, 800, ‘A3’), agent(‘A3’, 27.1),
order(‘012600’,’05/10/02’,’C501’, 1), detail(‘D127’, 25, ‘012600’, ‘AR3’) ,
article(‘AR3’, 1.5).
object(‘C502’) ← customer(‘C502’, 250.7 2500, ‘A2’), agent(‘A2’, 15).

FIGURE 5.11: An instance of CustomerDB database containing six complex individuals (units of analysis) forming the training set in a multi-relational regression problem of predicting the Credit Line for a customer. Customer represents the target table, while Order, Detail, Article and Agent represent the target relevant tables.

any order) are passed down to the right child t_R and they continue to be described only with respect to attributes in Customer table.

To evaluate this split test, we transform the (relational) description of training individuals as they fall in t_L and t_R respectively into attribute-value format, and then compute the best straight-line regression on each side according to transformed attribute-value individuals (see Figure 5.12.b). The transformation is performed by joining tuples from the portion of tables involved in relationally describing individuals at current node. This join operation takes into account the foreign key constraint conditions involved along the path from the root to the current node.

When each tree node is associated to a regression selection graph, this transformation of relational individuals falling in t_L (t_R) corresponds with the set of attribute-value tuples returned by running on D the SQL query $Q(G_{t_L})$ ($Q(G_{t_R})$) translating the regression selection graph G_{t_L} (G_{t_R})

In this case, three customers ($\boldsymbol{O}(t_L)$) are passed down to t_L and they are transformed into five attribute-value tuples ($\boldsymbol{O}_P(t_L)$), while remaining three customers ($\boldsymbol{O}(t_R)$) are passed down to t_R and they are transformed in three attribute-value tuples ($\boldsymbol{O}_P(t_R)$). The best straight-line regression on left (right) side is then computed on the basis of $\boldsymbol{O}_P(t_L)$ ($\boldsymbol{O}_P(t_R)$) attribute-value tuples: for each candidate regression attribute, intercept and slope are computed according to least square methodology and resulting straight-line regression is evaluated according to resubstitution error. In this way, we obtain $R(t_L) = 77.86$ when $Credit\ Line = 0.314\ Sale + 789.39$, while $R(t_L) = 109.54$ when $Credit\ Line = -1004\ Pieces + 1100$, and then return $Credit\ Line = 0.314\ Sale + 789.39$ as best straight-line regression on t_L. Similarly, we can identify $Credit\ Line = 3.99\ Sale + 1500.899$ as best straight-line regression on t_R with $R(t_R) = 70.50$. Finally, by applying Equation 5.1, we obtain:

$$\sigma(t) = \tfrac{5}{5+3}77.86 + \tfrac{3}{5+3}70.50 = 75.09.$$

♦

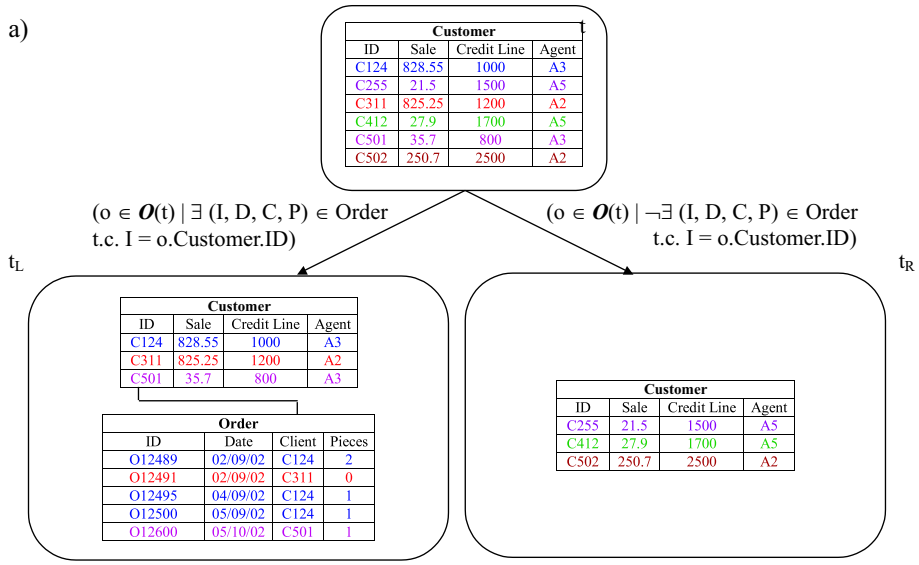
The evaluation of a regression step $\widehat{Y} = \widehat{\beta}_0 + \widehat{\beta}_1 X_i$ at node t requires to grow the (multi-)relational model tree at a further level in order to base the computation of $\rho(t)$ on the best splitting test, after that the current regression step has been performed. This look-ahead in evaluating regression steps aims at emulating the foresight of the splitting test evaluation measure σ which is actually based on resubstitution error in the best multiple linear regressions *after* the split is performed. Therefore, $\rho(t)$ is defined as follows:

$$\rho(t) = \min\{R(t), \sigma(t')\}. \tag{5.3}$$

where t' is the best multi-relational splitting node following the regression step in t.

In this way the criterion for selecting the optimal node is fully characterized coherently with the propositional case, but it also takes into account the relational structure of training data. Indeed, both splitting and regression nodes may involve single-valued attributes in the target table as well as eventually multi-valued attributes in target relevant tables. Consequently, both σ and ρ are based on a resubstitution error measure that is computed on the *attribute-value transformation* of the current relational description of the portion of training individuals falling in the partitions to be examined. The attribute-value transformation is performed by joining tables according to the multi-relational pattern currently generated.

The top-level algorithmic description of Mr-SMOTI is shown in Algorithm 5.2. Each node (splitting, regression or leaf) of the output tree is associated with a *multi-relational pattern* expressed as a regression selection graph and then translated into an SQL query. This permits representing the tree-based regression model as a set of SQL queries stored in XML format that can, in turn, be the object of a query. In particular, SQL queries associated with each leaf can be applied to new individuals stored in the relational database in order to predict an estimate of the corresponding (unknown) target attribute.



b)

$O_p(t_L)$

Customer_Order						
ID	Sale	Credit Line	Agent	ID_Order	Date	Pieces
C124	828.55	1000	A3	O12489	02/09/02	2
C124	828.55	1000	A3	O12495	04/09/02	1
C124	828.55	1000	A3	O12500	05/09/02	1
C311	825.25	1200	A2	O12491	02/09/02	0
C501	35.7	800	A3	O12600	05/10/02	1

Credit Line = 0.314 Sale + 789.39 (MSE(t_L)= 77.86)
Credit Line = -100 Pieces + 1100(MSE(t_L)= 109.54)

$O_p(t_L)$

Customer			
ID	Sale	Credit Line	Agent
C255	21.5	1500	A5
C412	27.9	1700	A5
C502	250.7	2500	A2

Credit Line = 3.99 Sale +500.899
(MSE(t_R)=70.50)

$$\sigma(t)=\frac{5}{5+3}77.86+\frac{3}{5+3}70.50$$

FIGURE 5.12: Computing the evaluation function according to an attribute-value transformation of the relational training individuals described in Figure 5.11 passed down to the left (right) child of a node t.

Algorithm 5.2 Mr-SMOTI Algorithm

`PROCEDURE build-Mr-SMOTI-tree(`D, T, Y, Q, R, F, τ`)`

`INPUT:`

D: `a relational database;`

T: `the target table in` D`;`

Y: `the target attribute in` T`;`

Q: `the SQL query describing a subset of individuals in` D`;`

R: `a subset of variables associated with attributes included in the SELECT clause of` Q (Q_{SELECT}) `not yet included in the regression model currently built;`

F: `a subset of foreign key constraints in` D `concerning at least one table included in the FROM clause of` Q `but not yet involved into a join condition of` Q`;`

`OUTPUT:`

```
    τ:  a (multi-)relational model tree with regression and splitting
        nodes built on (D, Y);
BEGIN
    Best − ρ = ∞;
    Best − t_R = a node whose model is the estimated mean Ȳ;
    FOR each continuous attribute X_i ∈ R DO
        Compute the best regression node t_R with attribute X_i of R;
        IF ρ(t_R) ≤ Best − ρ THEN
            Best − ρ = ρ(t_R); Best − t_R = t_R;
        END IF;
    END FOR;
    IF stopping criteria THEN τ = leaf(Best − t_R);
    ELSE
        Best − σ = ∞; Best − t_S = nil;
        FOR each attribute X_i ∈ Q_SELECT/{Y} and X_i is not a foreign
            key or primary key in D DO
            Compute the best splitting node t_S with attribute X_i;
            Compute the evaluation measure σ(t_S) for t_S;
            IF σ(t_S) ≤ Best − σ THEN
                Best − σ = σ(t_S); Best − t_S = t_S;
            END IF;
        END FOR
        FOR each foreign key constraint FK_i ∈ F DO
            Compute the best splitting node t_S involving the foreign key
            constraint FK_i;
            Compute the evaluation measure σ(t_S);
            IF σ(t_S) ≤ Best − σ THEN
                Best − σ = σ(t_S); Best − t_S = t_S;
            END IF;
        END FOR;
        IF Best − σ > Best − ρ THEN
            IF Best−t_S is a splitting node involving a foreign key THEN
                Best − FK_S is the foreign key involved in Best − t_S;
            ELSE
                Best − FK_S = ∅;
            END IF;
            Q_L = the query describing individuals passed down to the
            left child of Best − t_S;
            Q_R = the query describing individuals passed down to the
            right child of Best − t_R;
            build-Mr-SMOTI-tree(D, T, Y, Q_L, R, F/Best − FK_S, τ_L);
            build-Mr-SMOTI-tree(D, T, Y, Q_R, R, F, τ_R);
            τ = tree with root Best−t_S and left (right) branch τ_L (τ_R);
        ELSE
            Let Best − X the regression attribute in Best − t_R;
```

```
        Residual − R = ∅;
        FOR each X_i ∈ R/Best − X DO
            Residual − R = Residual − R ∪ {X_i − β_0i − β_1i Best − X};
        END FOR;
        Q_R = the query describing individuals passed down to the
        unique child of Best − t_R where each X_i ∈ R/Best − X is
        replaced with the corresponding residuals from residual−R;
        build-SMOTI-tree(D, T, Y, Q_R, residual − R, F, τ_R);
        τ = tree with root in Best − t_R and child τ_R;
      END IF;
    END IF;
  END PROCEDURE
```

5.3.2 Computing the set of splitting tests for a node

One of the main differences between Mr-SMOTI and the propositional SMOTI tree miner is the computation of the set of splitting tests to be considered at a node t. While SMOTI partitions a training space described by the m+1-dimensional feature vector $X_1, \ldots, X_m, Y$ according to binary tests on either a continuous variable (i.e. $X_k \leq \alpha$) or a discrete variable ($X_k \in \{x_{k_1}, \ldots, x_{k_s}\}$), Mr-SMOTI exploits the relational structure of data stored in D also checking the conditions which involve tuples not stored in the target table. By adopting the graphical language of regression selection graphs, this means that Mr-SMOTI starts with a root node t_0 that is associated with the regression selection graph G_0 containing only the target node and at each step, whenever a splitting node t is introduced into the model tree, Mr-SMOTI is in fact refining the regression selection graph G_t associated with the current tree node t, by applying either an add condition refinement or an add present arc and open node refinement. The set of candidate splitting test to be evaluated at t node according to the evaluation measure $\sigma(t)$ includes all possible splits involving a regression selection node $n_j \in G_t.N_O$.

More precisely, for each $n_j \in G_t.N_O$, Mr-SMOTI evaluates the candidate add condition refinements of G_t which are obtained for each residual $X \in n_j.R$, where X does not correspond to neither the target attribute nor a primary/foreign key attribute in D data model. When X corresponds with a continuous attribute in $n_j.T$, the condition c to be added to $n_j.C$ is in the form $X \leq \alpha$ with α one of the cut points found by an equal-frequency discretization [Cat91] of the ordered values of $n_j.R.X$ obtained by running $Q_X(G_t)$ (i.e., the projection on X of the SQL query $Q(G_t)$ associated with G_t) on D. The number of bins to be returned is fixed to the square root of the number of training tuples returned by running $Q_X(G_t)$. Conversely, when X corresponds with a discrete attribute in $n_j.T$, the condition c is in the form $X \in U$, with U a subset of the range of $n_j.R.X$ in the result set obtained by running $Q_X(G_t)$ on D. As in propositional SMOTI, the greedy strategy suggested in [MAR96] is employed to identify U. Initially, $U = \emptyset$ is considered, the possible refinement is then obtained by moving one discrete value from the range set computed for X to U, such that the move results in a better

splitting according to the heuristic measure σ.

Similarly, for each $n_j \in G_t.N_O$, Mr-SMOTI evaluates the best add present arc and open node refinement of the regression selection graph G_t, which is obtained by instantiating a new foreign key constraint of D data model involving $n_j.T$ as source table or destination table. Knowledge about the nature and multiplicity is used to guide and optimize the search. Since the investigated associations are foreign key constraints not already included in G_t, the proposed refinement can have two directions: *backward* or *forward*. The former express *many-to-one* associations, while the latter describe *one-to-many* associations in the data model. It is noteworthy that a many-to-one foreign key association leads to an add present arc and open node refinement whose complement cannot cover any training individuals. Therefore, a backward refinement of the regression selection graph G_t does not actually partition the set of individuals covered by G_t, but simply extends their descriptions by also considering the joined tuples in the table introduced with a new regression selection node foreign key associated with some regression selection node of $G_t.N_O$. Whenever the table in the added regression selection node contains continuous attributes which are not primary keys or foreign keys, the residuals of these variables are computed according to the regression steps already performed in τ along the path from the root to the tree node t associated with G_t. In this case, regression coefficients are estimated with the least square regression methodology on attribute-value set of joined tuples of D that satisfy the refinement (i.e. the set of tuples obtained by running the SQL query $Q(G_t)$ on D). Conversely, when no regression step has been performed, the list R associated with the added regression selection node contains exactly the list of attributes, without any modification, as they appear in the corresponding table of D.

It is important to note that only through the "add arc" refinements the exploration of all the relations in D is carried out. Indeed, it is possible to consider the "add condition" on some attribute (or residual) from some table only after the arc to this table has been added into the current regression selection graph.

5.3.3 Computing the set of regression steps for a node

Mr-SMOTI, evaluates all candidate regression steps associated with the current node t by considering each straight-line regression:

$$\widehat{Y} = \widehat{\beta}_0 + \widehat{\beta}_1 X. \tag{5.4}$$

between the target (residual) attribute Y ($Y \in n_0.R$) and the candidate regression (residual) continuous attribute X ($X \in n_i.R$ and $n_i \in G_t.N_O$) that is neither primary/foreign key in data model of D nor a regression attribute already involved in any regression step along the path from the root to the current node. Both slope ($\widehat{\beta}_0$) and intercept ($\widehat{\beta}_1$) are estimated according to least square regression methodology on the attribute-value tuples obtained by running $Q_{XY}(G_t)$ (i.e. the projection on X and Y of the SQL query $Q(G_t)$) on the training relational data in D. A similar approach is followed in determining the pairs of slope and intercept $(\beta_{ij_0}, \beta_{ij_1})$ to remove the effect of performed regression on X from all remaining

(residual) attribute $X_i \in n_j.R$ ($X_i \neq X$ and $n_j \in G_t.N_O$), where X_i does not correspond with any continuous attribute of $n_j.T$ that is either a primary/foreign key in D data model or a regression attribute already involved in a regression step along the path from the root to t.

Since the best regression at the node t is evaluated on the basis of the evaluation measure $\rho(t)$ that looks-ahead the best split after the current regression is performed, look-ahead refinements, which are sequence of several refinement, are used for dealing with this situation. More precisely, when a regression refinement is evaluated the best split refinement among the set of next candidate splits is also considered as refinements of G_t.

5.3.4 Stopping criteria

In Mr-SMOTI, three different stopping criteria are implemented. The first requires that the number of units of analysis covered by the regression selection graph associated with each node be greater than a minimum user-defined threshold. The second stops the induction process at the current node t and transforms it into a leaf node when all continuous attributes describing $\boldsymbol{O}(t)$ (i.e. continuous attributes of tables associated with each regression selection node $n_i \in G_t.N_O$) are involved in regression steps along the path from the root t_0 of the tree τ to t and there is no add open node and present arc refinement of G_t that extends the description of $\boldsymbol{O}(t)$ by including new continuous attributes. This is due to the fact that a continuous attribute chosen for a regression step is no longer considered for regression purposes, so that it can appear only once in a regression node along a path from the root to a leaf. Finally, the third stops the induction process when the coefficient of determination R^2[4] is greater than a minimum user-defined threshold.

It is noteworthy that a leaf node t that is introduced in τ typically corresponds with a regression step (without any look-ahead). The regression function associated with the leaf node t combines all straight-line regressions along the path from the root t_0 of τ to t with the best straight-line regression computed for the corresponding residual of the target attribute Y on the training subset $\boldsymbol{O}(t)$ as it appears described by the corresponding regression selection graph G_t. When all continuous attributes have been already involved in the regression model, the straight-line regression for the residual of Y to be associated to t is the function of constant zero value. Conversely, when no regression step has been performed and $\boldsymbol{O}(n)$ is only described by discrete variables, then the regression model corresponds properly to a constant function that is the mean of Y values for all tuples of D covered by the regression selection graph G_t.

Finally, the regression model $f_t : D \rightarrow \Re$ built at leaf node is expressed as an SQL query that predicts Y by grouping together all the (joined) tuples which describe a same multi-relational individual $\boldsymbol{o}$ according to the regression selection graph G_t and then averaging the returned prediction for Y (see Figure 5.13).

[4] R^2 is computed on the attribute-value tuples obtained by running $Q(G_t)$ on D

5.3.5 Complexity analysis

The computational complexity of adding a splitting node t into Mr-SMOTI tree depends on the complexity of performing a splitting refinement of the regression selection graph G_t associated with t and the complexity of the best regression step selection in the children regression selection graphs G_{t_L} and G_{t_R} respectively. On the contrary, the computational complexity of adding a regression node t depends on the complexity of performing a regression refinement of G_t and the complexity of the best splitting refinement in its refined regression selection graph G_{t_R}.

A splitting refinement of G_t can be either an "add condition" refinement or an "add present arc and open node" refinement. In the former case, the condition c to be added into a regression selection node $n_i \in G_t.N_O$ involves some residual $X \in n_i.R$ that represents either a continuous attribute or a discrete attribute of $n_i.T$. In continuous case, a threshold α has to be selected. Let D the training database, which consists of a set S of v tables $T_1, \ldots, T_v$ and a set FK of w foreign key associations $FK_1, \ldots, FK_w$. Each table $T_i \in S$ contains N_i tuples which are described according to M_i attributes, while each foreign key association $fk_i \in FK$ involves a pair of tables (T_i, T_j) from S. We denote by N the maximum number of tuples stored in a table $T_i \in S$ ($N = \max_{i=1,\ldots,v} N_i$) and G_{MAX} a regression selection graph obtained starting from G_0 (i.e. the regression selection graph which contains only the target node) and applying $h \leq w$ "add present arc and open node" refinements such that each foreign key path of D that connects the target table T with a table relevant table $T_i \in S$ ($T_i \neq T$) is included in the sub-graph of G_{MAX} corresponding with $G_{MAX}.N_O$. Hence, we may denote with M the total number of residuals in each regression selection node $n_i \in G_{MAX}.N_O$. In worst case, G_t includes exactly M continuous residuals and running $Q(G_t)$ on D returns N attribute-value tuples. Thresholds for X are determined by extracting X in D according to the projection on X of $Q(G_t)$ that is the query:

SELECT X FROM *TableList* WHERE *ConditionList* ORDER BY X.

where *TableList* and *ConditionList* translate G_t, and then equal-frequency discretizing these values and returning at worst $\sqrt{N}-1$ cut-points as candidate thresholds. It is noteworthy that the cost complexity of extracting X values strongly depends on performing the list of joins on *TableList* as well as the lists of joins involved in d ($d \geq 0$) sub-queries eventually included in the *ConditionList*. In worst case, each join list concerns all foreign key association in FK leading to a cost complexity $\boldsymbol{O}(N^{w+1})$. Moreover, the ordering of these values requires $\boldsymbol{O}(N \lg N)$ when an optimal algorithm is used to sort the values. Finally, the equal-frequency discretization requires a $\boldsymbol{O}$(N) cost complexity. Since M is the maximum number of residuals occurring in G_t, for each of the $M \times (\sqrt{N}-1)$ thresholds, Mr-SMOTI finds the best straight-line regression at both children, which has a complexity of $M \times N^{w+1}$ in the worst case. Hence, the add continuous condition refinement has a complexity $\boldsymbol{O}(M \times (N^{w+1} + N \log N + N) + M \times (\sqrt{N}-1)(M \times N^{w+1}))$. Similarly, for discrete case, the worst case complexity is $\boldsymbol{O}(M \times (N^{w+1} + k^2))$ where k is the maximum number of distinct values of a discrete variable. The selection of the best

discrete splitting test has a complexity $\boldsymbol{O}(M \times (N^{w+1}) + M \times k^2 \times (M \times N^{w+1}))$. Therefore, finding the best add condition refinement (either continuous or discrete) has a cost complexity $\boldsymbol{O}(M \times (N^{w+1} + N \log N + N) + M \times (\sqrt{N} - 1)(M \times N^{w+1}) + M \times (N^{w+1}) + M \times k^2 \times (M \times N^{w+1}))$, and under the reasonable assumption that $k^2 \leq N$ (i.e. the number of distinct values of the a discrete variable is less then the number of cases) and $\sqrt{N} \leq N$, the worst case complexity is $\boldsymbol{O}(M^2 \times N^{w+2})$. Conversely, an "add present arc and open node" refinement of G_t instantiates a foreign key association in FK that is not yet included in an arc of G_t already connecting a pair of nodes of $G_t.N_O$. In worst case, the selection of best "add present arc and open node" refinement has a cost complexity $\boldsymbol{O}(w \times M \times N^{w+1})$ Therefore, under the reasonable assumption that $w \leq M \times N$, finding the best splitting node (by either "add condition" refinement or "add present arc and open node" refinement) has a worst case complexity $\boldsymbol{O}(M^2 \times N^{w+2})$.

The selection of the best regression refinement G_{t_R} requires the computation, for each of the M residuals (worst case), of M straight-line regressions (one for the regression node plus $M - 1$ to remove the effect of the regressed variable) and the updating of residuals in each $n_i.R$ for $n_i \in G_t.N_O$. This takes time $\boldsymbol{O}(M^2 \times (N^{W+1} + N))$, since the complexity of the computation of a generic straight-line regression between X_1 and X_2 depends on extracting X_1 and X_2 values that is linear in N^{W+1} and computing both intercept and slope that is linear in N. Moreover, for each straight-line regression, a splitting test is required, which has a worst case complexity of $\boldsymbol{O}(M^2 \times N^{w+2})$. Therefore, the selection of the best regression step has a complexity $\boldsymbol{O}(M^2 \times N^{W+1} + M^3 \times N^{w+2})$, that is, $\boldsymbol{O}(M^3 \times N^{w+2})$.

The above results lead to an $\boldsymbol{O}(M^3 \times N^{w+2})$ worst case complexity for the selection of any node (splitting or regression).

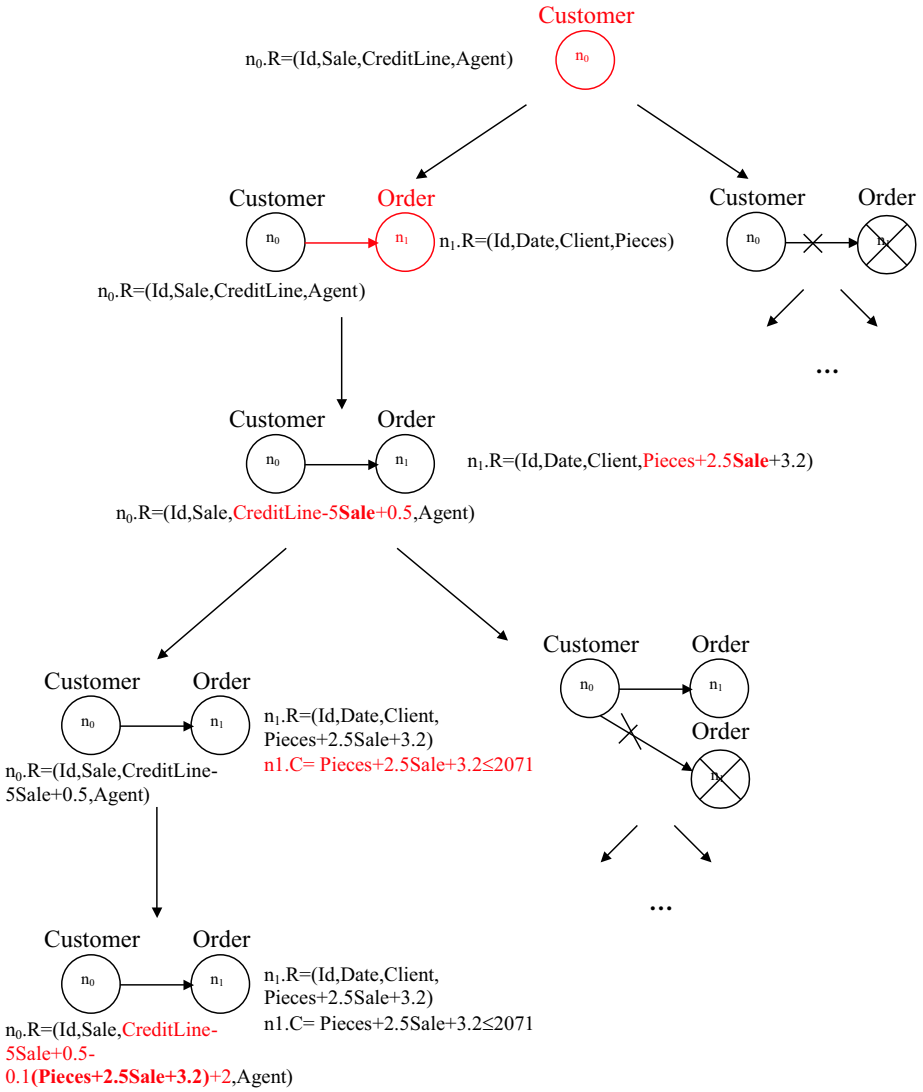
5.4 Conclusions

In this chapter we have presented a novel multi-relational TDIMT method, namely Mr-SMOTI, that upgrades SMOTI algorithm to multi-relational representations. Mr-SMOTI has been implemented as a module of the system MURENA (MUlti-RElatioanl ANAlizer)) that tightly couples an Oracle® 9*i* relational database. Consequently, it is able to exploit information about relational data model embedded in the database schema (relational data model) in order to both guide the tree growing and reduce the search space of multi-relational patterns.

Mr-SMOTI mines multi-relational model trees by integrating both the predictive step and splitting partitioning from data stored in multiple tables of a relational database. This means that, similarly to the propositional SMOTI, Mr-SMOTI is able to mine model trees with two types of internal nodes: regression nodes and splitting nodes. The regression model at each leaf t is effectively built stepwise by combining the best straight-line regression computed on t with all straight-line regressions along the path from the root to the current node. This potentially solves the problem of modeling multi-relational phenomena, where some attributes have a global effect while others have only a local effect. Moreover, differently from the

propositional case, patterns associated with each tree node are effectively multi-relational patterns in the sense they may involve multiple attributes from several tables of the relational data schema.

These multi-relational patterns are expressed in the graphical language of regression selection graphs which are extensions of the classical selection graphs already proposed to represent the multi-relational patterns associated with both splitting nodes and leaves of a multi-relational decision trees. Both splitting and regression refinements of regression selection graphs have been formally introduced such that Mr-SMOTI tree growing can be equivalently expressed in terms of splitting/regression refining the regression selection graph associated with the current node under analysis. Each regression selection graph is easily translated into SQL, or equivalently into first order logic expressions such that the entire tree-based regression model can be expressed as a set of SQL queries stored in XML format.




```
SELECT n0.Id, AVG (5*n0.Sale-0.5+0.1*(n1.Pieces+2.5*n0.Sale+3.2)-2)
FROM Customer n0, Order  n1
WHERE n0.Id=n1.Client AND n1.Pieces+2.5*n0.Sale+3.2≤2071
GROUP BY n0.Id
```

FIGURE 5.13: Representation in form of SQL query of the (multi-)relational multiple regression function at leaf node of an hypothetical model tree stepwise built by Mr-SMOTI on an instance of CustomerDB.

Chapter 6

Applications

Multi-relational regression has several applications in social sciences, physical and biological sciences, business and technologies as well as humanities. In this chapter, we present three real-world applications of multi-relational regression. The first one involves the analysis of geo-referenced census data provided by United Kingdom (UK) 1991 census to investigate the migration phenomenon in Stockport enumeration districts (EDs). The last two applications are related to bioinformatics and concern the quantitative interpretation of structure-activity relationships of chemical compounds (QSARs) in two multi-relational datasets, namely Mutagenesis and Biodegradability. Mutagenesis concerns the prediction of molecule mutagenic activity while Biodegradability involves the prediction of biodegradability in an aqueous environment under aerobic conditions.

For each application, units of analysis $\boldsymbol{O}$ are finally stored in multiple tables of a relational database D that is analyzed by means of a 10-fold cross-validation. This means that D is firstly divided into ten blocks $\{D_1, \ldots, D_{10}\}$ that is, the target table in D is first divided into ten blocks of near-equal size and distribution of target values, and then, for every block, a subset of target relevant tuples in tables of D foreign key path associated with tuples in the target table block are extracted. In this way, ten databases are created. Mr-SMOTI is trained on nine databases and tested on the hold-out database.

The predictive accuracy of the method is evaluated on the basis of the average square radix mean square error ($Avg.RMSE$) computed as follows:

$$Avg.RMSE = \frac{1}{10} \sum_{D_i \in 10-CV(D)} \sqrt{\frac{1}{\#D_i} \sum_{j=1,\ldots,\#D_i} (y_j - \hat{y}_j)^2} \tag{6.1}$$

where $\#D_i$ is the number of units of analysis (multi-relational individuals) in D_i (i.e. number of target units of observation stored as single-tuples in the corresponding target table T in D_i), and $\hat{y}_j$ is the value predicted for the j-th testing individual of D_i by the regression model built on $D \backslash D_i$. Obviously, for each fold D_i, a testing individual $\boldsymbol{o}_j \in D_i$ is described according to the same relational data model of D (i.e. a single tuple in the target table of D_i, and zero, one or more tuples in foreign key path associated target relevant tables).

The complexity of learned regression models is evaluated according to both the average number leaves and the average number of regression nodes.

For each application, we analyze results obtained with the use of both Mr-SMOTI and TILDE-RT on the multi-relational representation of data and compare these methods with some propositional TDIMT methods that is SMOTI and M5'. The empirical comparison with S-CART that is the other multi-relational TDIMT system able to label leaves with multiple functions was not possible since the system is not publicly available.

Propositional regression methods are applied to two different attribute-value data representation derived by transforming each problem from a multi-relational setting to a propositional one. The first transformation (P1) creates a single table by performing some *left outer join* operations[1] that include all attributes forming the schema of the target table and derive remaining attributes from the target relevant tables which are foreign key path associated with the target table. This data transformation may cause a natural change of the units of analysis since a single multi-relational individual is transformed into one or *more* attribute-value individuals in presence of one-to-many associations between tables in question. Conversely, the second attribute-value representation (P2) is derived by properly propositionalizing the multi-relational dataset. The propositionalization is here intended as the process of transforming a multi-relational dataset, containing relational individuals, into a propositional dataset with derived attribute-value features, describing the structural properties of individuals. The process can thus be thought as summarizing data stored in multiple table containing *one tuple* for each individual [KHS01]. The traditional way to summarize each relationship in statistics and OLAP is through aggregates which are based on histograms (e.g. count, sum, min, max and avg). In this context, we extend the schema of target table with aggregated information found in the target relevant tables which are foreign key path associated with target table. Continuous attributes are aggregated by average while discrete attributes are aggregated by mode.

For pairwise comparison of methods, the non-parametric Wilcoxon two-sample paired signed rank test is used, since the number of folds (or independent trials) is relatively low and does not justify the application of parametric tests, such as the t-test. In all experiments reported in this empirical study, the significance level used in the test is set at 0.05.

The thresholds for stopping criteria are fixed as follows: the minimum number of individuals falling in each internal node must be greater than the square root of the number of individuals in the entire training set, while the coefficient of determination R in each internal node must be below 0.80.

6.1 Applications to geo-referenced census data

One of the application domain that benefits from multi-relational setting for regression is regression in spatial databases where the unit of representation is spatial

[1] Left outer join operation includes all of the tuples from the first (left) of two tables, even if there are no matching values for tuples in the second (right) table.

object, i.e. an entity having an associated value (attribute) of a spatial data type (point, line or polygon in 2D), a-spatial properties as well as space-depending attribute characteristics [Mar99]. Both spatial objects and space-depending attributes are associated with location in space in which they exist or have constant values respectively. It is worthwhile to emphasize that space-depending attributes are characteristics of the embedding space and indirectly become properties of the spatial objects via their location in space.

Example 6.1 *A land parcel has the vegetation rate (i.e. percentage of soil covered by vegetation) as attribute. While space-depending attribute such as vegetation rate exists over the space, spatial objects such as land parcels exist in certain locations and inherit those values of attributes referring to their exact location. So, although some applications may view the vegetation rate as a property of the land parcel, it is clear that (i) the attribute vegetation is defined whether or not the land parcel boundary exists at that location space, and (ii) when the land parcel changes shape, the attribute vegetation rate inherits new values from the new location.*

♦

The location of spatial objects in a space defines implicitly spatial relationships of different nature, such as geometrical (e.g. distance), directional (e.g. north of) and topological (e.g. adjacent, inside or disjoint) relationships.

In the last two decades spatial data have attracted a great deal of attention. This is due to the rapidly expanding amount of spatial data gathered by collection tools, such as satellite systems or remote sensing systems which have paved the way for advances in spatial data structures [Güt94], spatial reasoning [Ege91] and computational geometry [PS85] to serve multiple tasks including storage and sophisticated treatment of real-world geometry in a spatial database.

An important application area where spatial database is emerged as central is offered by urban planning where advances in geo-referencing have caused a growing demand for more powerful exploratory spatial data analysis to acquire the necessary predictive or descriptive knowledge by analyzing jointly socio-economic data and topographic maps both stored in spatial databases. This has been the practice in urban planning environments for centuries. For instance, population and economic census data overlapped to geographic data can be the key indicator of level of deprivation, thus supporting a good public policy. In these studies, geo-referencing has enabled the spatial representation of socio-economic phenomena as spatial objects. In UK for instance the geo-referencing units for population census data are the areal objects ED (enumeration district), ward, district, and county, of which ED is the smallest unit for which data is published.

These advances in geo-referencing have provided an added impetus to the development of GISs (Geographical Information Systems) which are increasingly used to store, manipulate and analyze physical, social and economic data of a geographic area in order to provide the information necessary for effective decision-making in urban planning [HK89]. As a toolbox, a GIS allows planners to perform spatial analysis using geo-processing functions such as map overlay or connectivity measurements. Yet GISs do not adequately support the spatial decision process because

they lack of appropriate modeling capabilities [Den91] [Kee98]. In this case, spatial data mining adds some data analysis capabilities to GISs to assist the work of public and private planners in knowledge-intensive activities such as urban planning [MEL+03].

In this section, we strike the right balance between fundamental concepts of spatial databases and special requirements of spatial data mining. In particular, we focus on regression that is a fundamental task in spatial data mining, where the goal is to mine a spatial regression model on the basis of the interaction of two or more spatially-referenced objects or space-dependent attributes, according to a particular spacing or set of arrangements. Finally, we discuss an application of multi-relational regression to some real-world geo-referenced census data from Stockport area.

6.1.1 Spatial databases

A spatial database is a full-fledged database system with additional capabilities for handling both spatial data types in its data model and spatial query language and supporting them in their implementation by providing at least spatial indexing and efficient algorithms for spatial join [RSV02]. These spatial requirements justify the need of integrating the representation and sophisticated treatment of real-world geometry with traditional data at the logical level and providing an efficient support to model, store and query spatial data at the physical level.

It is clear that this spatial data treatment cannot be addressed with a traditional relational data model since the underlying data representation, the query language and access methods are designed to deal with simple types such as integer and string. Really, information concerning a spatial object should be spread over many relations and many join operations have to be performed to recreate a typical complex, structured spatial object. Furthermore, no appropriate indexing and retrieval operations for spatial data are provided.

Conversely, the object-relational (OR) paradigm appears to be perfectly suited to manage spatial data that requires complex structures to describe the geometry of spatial objects and model the relationships among them [SM96]. The geometry attribute is a structured data type defined to represent a geometry in terms of both Euclidean spatial dimensions (e.g. set of x-y coordinates) and additional data dimensions including depth, time and elevation: different types of spatial objects (e.g. wards or roads) are modeled in layers located in single relations, sharing a common coordinate system. Each layer can have its own set of a-spatial attributes $A_1, \ldots, A_n$, named thematic data, and at most one geometry attribute S. Thereby, an object-relational spatial database D is a set of relations $T_1, \ldots, T_n$, where each relation $T_i \in D$ has either a geometry attribute S_i or an attribute A_i, such that T_i can be linked (joined) to a relation $T_j \in D$ having a geometry attribute S_j. Points, lines and/or polygons are stored in a single geometry field within a relation. A spatial query combines spatial information with attribute data describing objects located in the space and include spatial join operators which are defined to query multiple layers on the base of geometrical, directional or topological relationships.

To increase the efficiency of spatial join operations, spatial indexes, such as KD-tree and/or Quadtree [Sam90], can be built on geometry attributes.

In alternative, the object-oriented (OO) paradigm may be suited in modeling spatial objects and managing huge quantities of spatial data. For instance, in [MEL+03], Malerba and his colleagues present a protypical GIS named INGENS (Inductive Geographic Information System) that couples a collection of maps stored in an object-oriented DBMS (ObjectStore 5.0 by Object Design, Inc.), so that full use is made of a well-developed, technologically mature a-spatial DBMS. Each map is treated as a grid of cells with respect to a hybrid tessellation-topological model that follows the usual topographic practice of superimposing a regular grid on a map to simplify the localization process: a map is divided into square cells of the same size and one-to-one associations (e.g. left, right, up, down, ...) among cells allow map-reading from a cell to one of its neighbors in the map. For each cell both the raster image in GIF format is kept in addition to its vector coordinates and component spatial objects which are described according to two structural categories of objects: geometric and thematic. The former corresponds to a physical hierarchy, while the latter corresponds to a logical hierarchy. The physical hierarchy describes the spatial objects according to the most appropriate geometric entity, that is: point, line or polygon. In different maps of the same geographical area, the same object may have different physical representations. For instance, a road can be represented as a line on a small-scale map, or as a region on a large-scale map. Points are described by their spatial coordinates, while (broken) lines are characterized by the list of line vertexes and regions are represented by their boundary lines. Similarly, the logical hierarchy describes the nature of each spatial object. For instance, an administrative boundary must be put into one of the following classes: city, province, county or state. Both physical and logical hierarchies may be easily modeled in OO data model by means of features like polymorphisms and inheritance. The limitation is that neither spatial indexing nor spatial join operation are embedded in classical OODBMS, but the object-oriented technology facilitates the extension of the OODBMS to accommodate the management of spatial objects. Indeed, INGENS architecture includes the Map Descriptor module that is the application enabler responsible for the automated generation of first-order logic descriptions of spatial and a-spatial properties or relationships involving geographical objects stored in the coupled OO database [LML+02].

6.1.2 A spatial data mining framework

Spatial data mining investigates how implicit knowledge, spatial relationships, or other patterns not explicitly stored in spatial data can be extracted [Kop99]. A spatial pattern is a pattern showing the interaction of two or more spatially-referred objects or space-dependent attributes, according to a particular spacing or set of arrangements [DeM00]. Although many data mining tools deal at least implicitly, with spatial data, they essentially ignore the spatial dimension of data treating them as non-spatial.

Spatial data mining demands for the development of specific techniques which,

differently from traditional data mining techniques, take this spatial dimension of the data into account when exploring the pattern space. This leads to one of the main degree of complexity in mining spatial data that is the implicit definition of spatial properties and relationships due to the geometrical representation and positioning of spatial objects in the space. Modeling these spatial relationships is a key challenge both in descriptive tasks (e.g. association rules discovery) [MELA02] and predictive tasks (e.g. classification and regression) [SSV+02] that arise in spatial domains.

The problem of regression in spatial domains has been already investigated by some researchers [KVCS96] [PB97] that exploit principles of spatial statistics [Cre93] to perform regression on spatial objects belonging to a single layer (e.g. land parcel layer) by taking into account the spatial correlation among them (e.g. land parcels forming the neighborhood of a specific parcel). In the special case a regression model is built on area referenced data, the target variable is associated with an area. The units of observation for an area can be descriptive of one or more primary units, possibly of different type, within the area. In addition to attributes that relate to primary units or areas, there are attributes that refer to relationships between primary units (e.g. contact frequencies between households) and between areal units (e.g., migration rates). This relational information may affect the spatial variation and it must be taken into account in modeling phase. For instance, when the target variable Y_i measures the proportion of people suffering from respiratory disease in area i and if X_{ki} measures levels of atmospheric pollution in area i, the Y_i may not be only a function of X_{ki} but also levels of X_k in areas neighboring i. Such definition reflects both the reinforcing effects of extensive tracts of high (low) levels of pollution but also the fact that people move across areas over the course of time and their exposure to this risk factor is not only a function of their local environment. In this case, we are interested in mining a regression model with spatially lagged explanatory variables [Hai90].

Whittle [Whi54] and Mead [Mea67] have argued that a further aspect of spatial regression is that the effect of target variable at i site may operate as an explanatory (predictor) variable at another site. This leads to build a spatial regression model with spatially lagged target variable [Hai90] that can be used to describe the spatial correlation among the values assumed by the target variable in a specific site and its neighborhood. For instance, when the target variable Y_i measures the level of atmospheric pollution in an area, its value may be influenced by the spatial variation of level of atmospheric pollution in the neighborhood.

In general, when the spatial heterogeneity of response is anticipated, data analyst may allow either the constant or one (or more) of the other regression parameters to vary spatially, thanks to the introduction of dummy variables identifying different sub-areas of area under analysis. This comprises the benefits of automated methods that look for regional segmentation (e.g. tree structured models) according to spatial heterogeneity and build a spatial regression model for each segment, i.e., piecewise spatial regression model.

All these issues are clearly coped by Mr-SMOTI that fully exploits the multi-relational approach to build regression model in form of trees including spatially

lagged explanatory (eventually target) variables. In this case, multi-relational approach perfectly fits the need of representing spatial objects, spatial relationships among them as well as spatial dependent attributes and exploits the fact that the effect of an explanatory variable at any target spatial object is not limited to the properties of the specified object during tree induction step. Moreover, differently from classical spatial data mining methods that underlie single table assumption, Mr-SMOTI is able to process units of observations collected for spatial objects belonging not necessarily to the same layer (e.g. EDs, road crossing EDs, urban areas overlapping EDs, etc.).

As a consequence discovered patterns are model trees with regression and splitting nodes that may involve attributes from several spatially related layers. Splitting nodes contribute to segment area under analysis according to spatial heterogeneity in spatial structure of units of analysis. Regression nodes allow to detect spatial and a-spatial factors which have either a global or local effect for prediction.

Since Mr-SMOTI is tight-coupled with a relational database, this requires a complex data transformation to make spatial relationships explicitly stored in tables of a relational database. This requirement has been already coped for spatial association rules discovery tasks in [ACL+03], where a spatial data mining framework named ARES (Spatial Association Rules Extractor) is described. ARES couples an object-relational spatial database (Oracle® Spatial Cartridge 9*i*) and integrates an algorithm named SPADA (Spatial Pattern Discovery Algorithm) [LM04] to mine multi-level spatial association rules involving relationships among spatial objects stored in a spatial database. In this case, the access to geo-referenced data for making explicit spatial properties and/or relationships among them is accomplished through a middle layer including two modules namely RUDE (Relative Unsupervised DiscrEtization) and FEATEX (FEATure EXtraction). RUDE is designed for relative unsupervised discretization [LW00] of numerical features, while FEATEX is implemented as a PL/SQL Oracle package of procedures and functions, each of which computes a different feature [ACL+03]. Hence, FEATEX functions can be used in SQL queries (see Example 6.2). According to their nature, features extracted by FEATEX can be distinguished as geometrical, directional and topological features [LML+02]. Geometrical features (e.g. area, length) are based on principles of Euclidean geometry, directional features (e.g. north, south, north-east, north-west) regard relative spatial orientation in 2D, while topological features (e.g. crosses, on top) are relations preserving themselves under topological transformations such as translation, rotation, and scaling. In addition, hybrid features (e.g. roughly parallel), which merge properties of two or more feature categories, can be also extracted by FEATEX.

Example 6.2 *The query:*

SELECT FEATEX.RELATE(X.GEOM, Y.GEOM) from roads X, parks Y

analyzes the geometries of each pair of spatial objects in the layers of interest (roads and parks) and returns the name of the topological binary relationship (e.g. disjoint, crosses) between them. Topological relationships are computed according to the 9-intersection model [Ege91] that is based on the consideration that for each spatial

object X it is possible to distinguish three parts: its interior (A^O), its boundary (δA) and its exterior (A^-). Therefore, binary topological relationships between two objects can be described in terms of part intersections.The 9-intersection model is concisely represented as a 3×3 matrix:

$$R(X,Y) = \begin{pmatrix} X^0 \cap X^0 & X^0 \cap \partial Y & X^0 \cap Y^- \\ \partial X \cap Y^0 & \partial X \cap \partial Y & \partial X \cap Y^- \\ X^- \cap Y^0 & X^- \cap \partial Y & X^- \cap Y^- \end{pmatrix}$$

Not all the matrix configurations correspond to physically feasible relations between two spatial objects.

♦

In [CAM04], Ceci and his colleagues focus on spatial classification tasks and present an extension of ARES framework that includes a spatial associative classifier method based on a multi-relational approach that is able to take spatial relationships into account. Classification is driven from spatial association rules and it is performed by Mr-SBC (Multi-Relational Structural Bayesian Classifier) [CAM03b] that is an extension of naïve Bayes classifier [DP97] to multi-relational setting. Similarly to Mr-SMOTI, Mr-SBC is currently implemented as a module of the system MURENA and therefore tightly-couples an Oracle® 9*i* relational database. This means that features of spatial objects are extracted by FEATEX and stored into relational tables. Foreign key associations between tables are then adequately modeled.

Following this suggestion, we integrate Mr-SMOTI into this spatial data mining framework (see Figure 6.1) to support data miners in regression tasks occurring in spatial domains when spatial relationships and properties are explicitly computed by FEATEX and stored into multiple relational tables.

It is noteworthy that, spatial regression performed within this framework improves classical spatial regression methods since it makes data miners able to detect not only spatial correlation among areal units but also among spatial objects of different type (e.g. the proportion of people suffering from respiratory diseases in a site depends on the high/low level of pollution of sites where people daily move but it may also depends on the traffic value of main roads crossing the area).

6.1.3 Mining Stockport geo-referenced census data

In this section, we present a real-world application concerning the mining of a multi-relational regression model for a quantitative interpretation of geo-referenced census data. We consider both 1991 census and digital map data provided in the context of the European project SPIN! (Spatial Mining for Data of Public Interest) [May00]. This data concerns Stockport, one of the ten metropolitan districts in Greater Manchester, UK which is divided into twenty-two wards for a total of 589 census EDs. Spatial analysis is enabled by the availability of vectorized boundaries for 578 Stockport EDs as well as by other Ordnance Survey digital maps of UK, where several interesting layers are found, namely shopping areas, housing areas, and employment areas (see Figure 6.2). Census data collected on 1991 is available

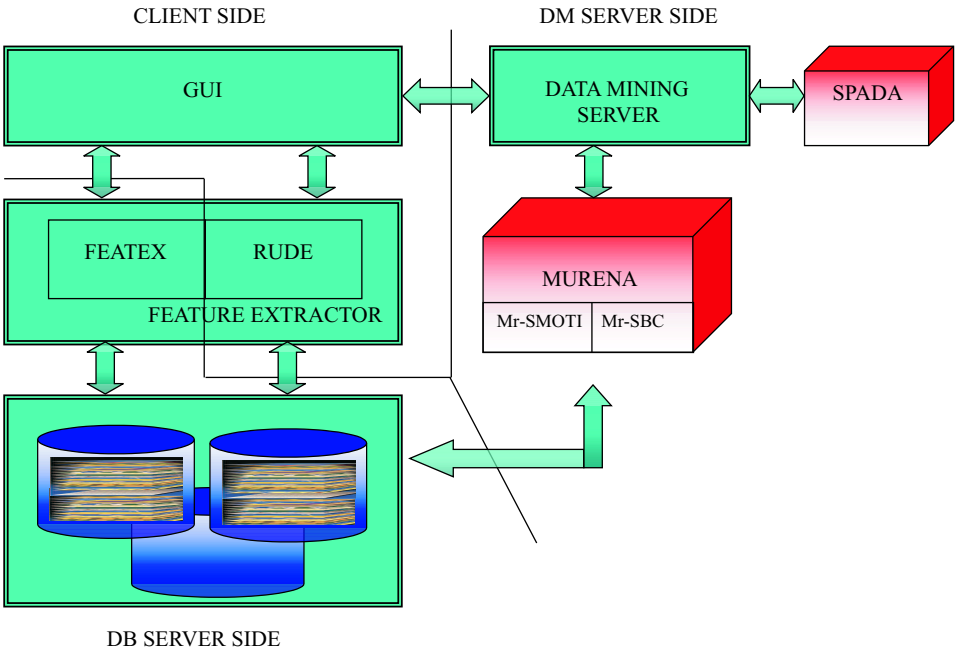


FIGURE 6.1: A spatial data mining framework.

at ED level and provides socio-economic statistics (e.g. number of migrants, number of communal establishments, etc).

Both census and map data are stored in an Oracle® Spatial Cartridge 9*i* database according to an object-relational data model.

The application presented in this study investigates the number of migrants in Stockport EDs according to socio-economic factors represented in census data as well as geographical factors (spatial layers) represented in topographic maps. This is a spatial regression problem where units of analysis involve several units of observation representing spatial objects belonging to several geographical layers and spatial (topological) relationships among them. Target objects of analysis are Stockport EDs. For each target ED, we determine by FEATEX relate function the list of adjacent EDs in Stockport map forming the neighborhood of the ED as well as the list of shopping areas, housing areas and employment areas whose boundary (partially) overlaps the ED in question. In this way, units of analysis are obtained by making topological relationships explicit and then they are stored in multiple tables of an Oracle® 9*i* relational database (see Figure 6.3). We extract census data concerning the number of migrants (i.e. the target attribute), the number of migrants moving within the ward and the number of communal establishments in each ED. Census data also contains information about the number of employees extracted on a 10% sample of ED population. Finally, for each pair formed by a shopping (housing or region) area and overlapped ED, we compute the portion of ED that is covered by the overlapping shopping (housing or region) area.

We define three experimental setting. The first one (BK_1) is obtained by ignoring data on neighboring EDs. This is an example of intra-ED analysis where spatial

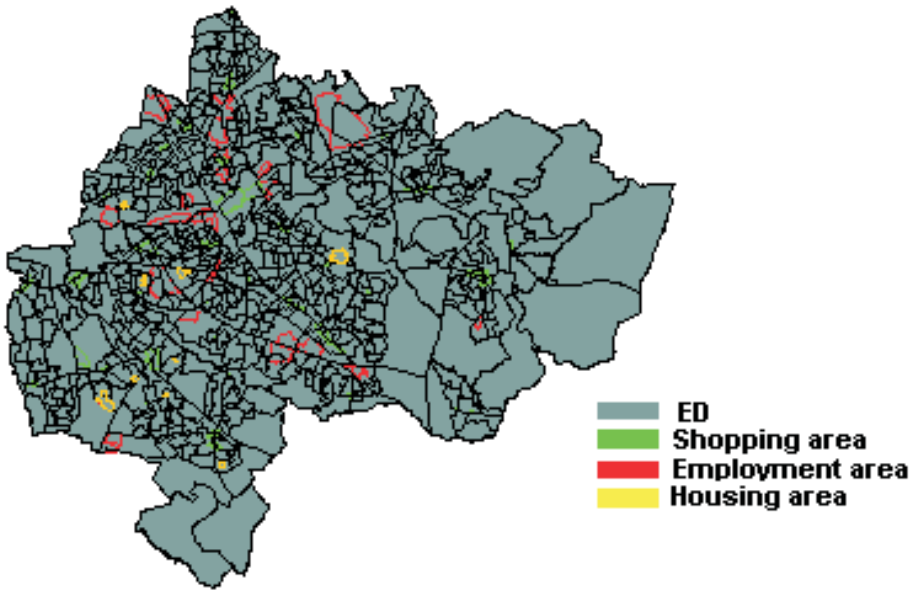


FIGURE 6.2: Geographic layers extracted from the topographic map of Stockport.

features concern geographic objects enclosed within the boundaries of an ED, while a-spatial features are aggregated census data concerning a single ED. Intra-ED analysis is generally adopted in the characterization of a site (residential, industrial area, and so on) for land allocation purposes. The second setting (BK_2) is obtained by processing neighboring census data with exception of number of migrants. This is an inter-ED analysis, where topological relationships (e.g. adjacent) between two regions are used to express spatial dependencies between EDs in addition to topological relationships (e.g. overlap) between EDs and shopping areas, housing areas or employment areas. The third setting (BK_3) is obtained by extending BK_2 data with number migrants on EDs forming the neighborhood of each target ED. In this case we are interested in modeling spatial migration phenomenon in Stockport by also exploiting the self-correlation on spatially lagged target variable over the neighborhood of each ED. This corresponds with formulating the hypothesis that a systematic spatial variation of number of migrants in Stockport EDs may arise not only from the effect of one or more explanatory (predictor) variables over the target ED, neighboring EDs or shopping (housing or employment) areas overlapping the ED in question but it is also self-induced in the neighborhood. This auto-correlation can be exploited both in the automated spatial segmentation (partitioning step) of Stockport EDs as well as in forming regression function (regression step) to predict number of migrants in each segment.

Each node of the tree mined by Mr-SMOTI in each BK_i setting is translated into an SQL query and stored in XML format as shown in Figure 6.4. SQL queries associated with regression selection graphs corresponding to leaves of the tree have multiple linear functions associated with them. Functions at leaves may involve variables from two or more spatially-referenced objects. They are built stepwise, therefore the effect of some variable that is introduced at higher level of tree is

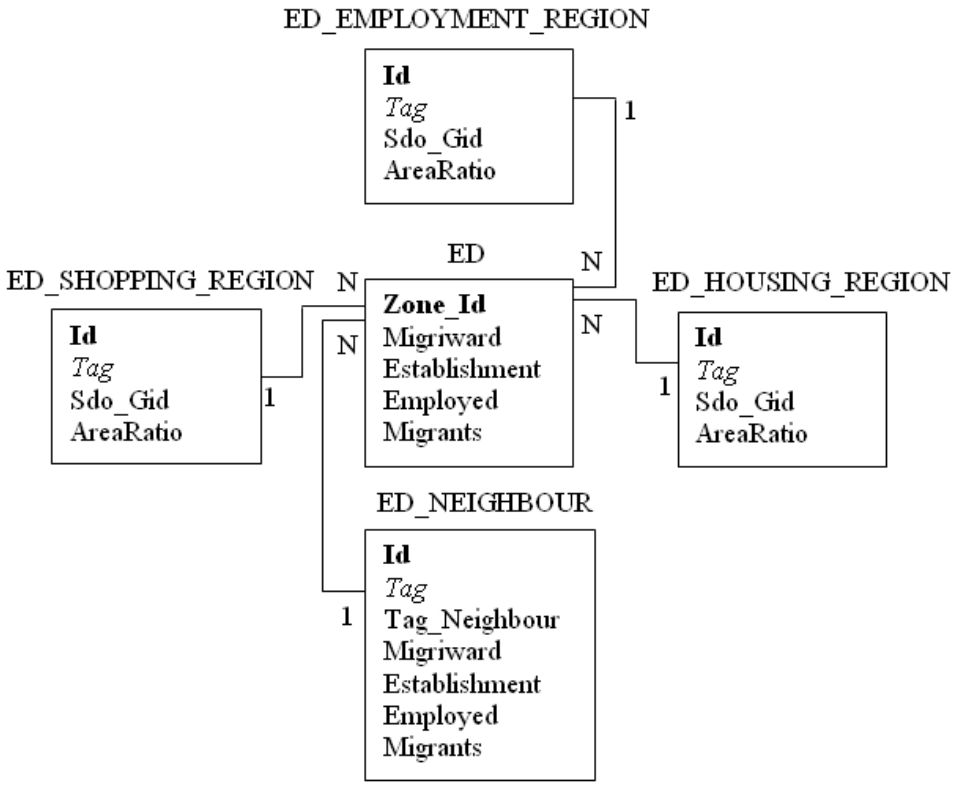


FIGURE 6.3: Multi-relational representation of geo-referenced census data extracted from Stockport data.

shared by several functions associated with different leaves. This allows us to interpret discovered patterns in order to identify variables which have a global or local influence on the trend of phenomenon to be modeled also according to a particular set of spatial arrangements.

Each target ED satisfies exactly one query associated with a leaf of mined tree. The complementary nature of different branches of a splitting node ensures that a given ED instance is not assigned conflicting prediction. Therefore, the set of queries associated with leaves may be run on a new database structured according to the same relational model of Stockport database in order to predict the unknown values for the number of migrants.

At BK_1 level, Mr-SMOTI mines a multi-relation model tree in 317 secs partitioning Stockport data into 51 leaves. The tree is built by fully exploiting spatial relationships (i.e. overlap) between EDs and shopping areas (housing areas or employment areas). The model tree is equivalently expressed by the set of mutually exclusive SQL (or Prolog) queries associated with each leaf of the output tree structure. An example of extracted regression rule is the following:

```
SELECT T0.ZONE_ID T0ZONE_ID,
    AVG((16.06+4.22*T0.EMPLOYED)+(1.4266667+-58.791576*
    (T1.AREARATIO -(0.42323083+-0.14083165*T0.EMPLOYED))))
```

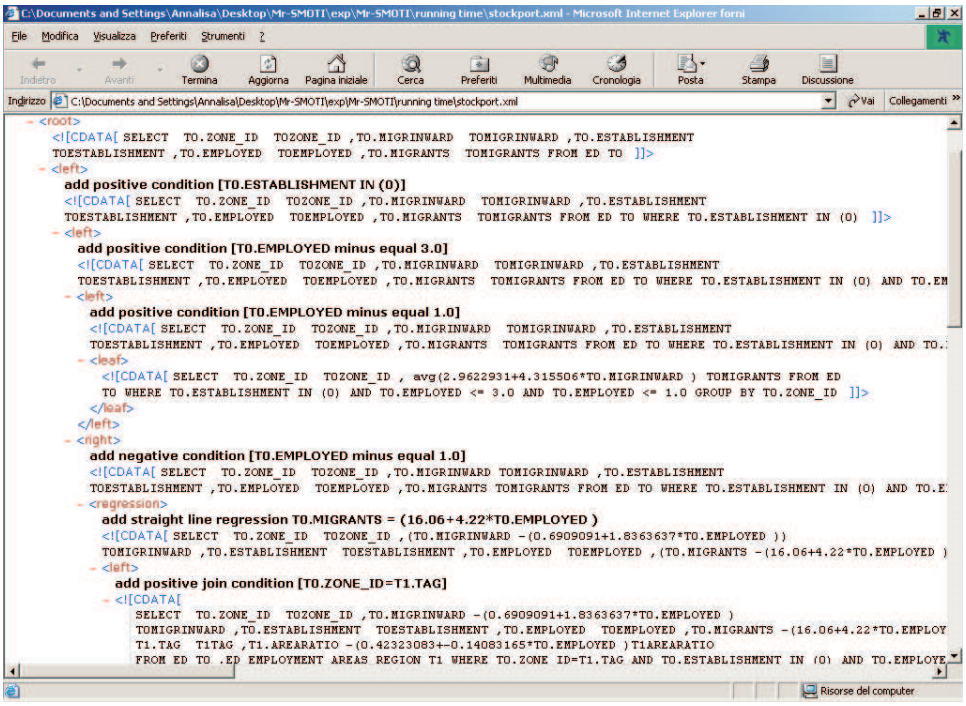


FIGURE 6.4: A portion of the model tree built by Mr-SMOTI to predict the number of migrants from Stockport data at BK_1 level.

FROM ED T0, ED_EMPLOYMENT_REGION T1
WHERE T0.ZONE_ID=T1.TAG AND T0.ESTABLISHMENT IN (0) AND
T0.EMPLOYED <= 3.0 AND T0.EMPLOYED > 1.0.
GROUP BY T0.ZONE_ID

The variable T0.EMPLOYED has here a global effect since it is estimated by considering a set of examples that includes as a proper set the training examples effectively falling in the leaf node. This means that the effect of a straight-line regression on T0.EMPLOYED is shared by all leaves of the sub-tree rooted in the corresponding regression node. In this special case, straight-line regression on the number of employees on 10% sample population of target ED (T0.EMPLOYED) is estimated on the entire set of target EDs (T0) having both no communal establishments (T0.ESTABLISHMENT IN (0)) and number of employees in the 10% sample population that is greater than one but less or equal to three (T0.EMPLOYED<= 3.0 AND T0.EMPLOYED>1.0) independently by the presence or absence of some employment area overlapping the target ED in question. In contrast, straight-line regression on the residual of the percentage of ED area overlapped by a single emplyment area (T1.AREARATIO-0.42323083+0.14083165*T0.EMPLOYED) is evaluated on a proper subset of the target EDs used to estimate the straigh-line regression on T0.EMPLOYED that is the EDs (partially) overlapped by at least one employment area.

When Stockport database is translated into a set of Prolog facts, the regression rule above can be equivalently expressed as a Prolog program in the form:

```
query(X,Y):-findall(ZONE_ID,ed(ZONE_ID,_,_,_,_),L),predictMigrants(L,Y).

predictMigrants([ ],[ ]).
predictMigrants([H|T],W):- migrants(H,M), W =[[H,M]|V], predictMigrants(T,V).
predictMigrants([H|T],v):- predictMigrants(T,V).

migrants(ZONE_ID, MIGRANTS) :- findall([EMPLOYED, AREARATIO,
    MIGRANTS], rule(ZONE_ID,MIGRINWARD,ESTABLISHMENT,
    EMPLOYED,MIGRANTS,AREARATIO), [ ]),!, fail.
migrants(ZONE_ID, MIGRANTS) :- findall([EMPLOYED, AREARATIO,
    MIGRANTS], rule(ZONE_ID, MIGRINWARD, ESTABLISHMENT,
    EMPLOYED, MIGRANTS, AREARATIO), EDs), predict(EDs, MIGRANTS),
    !.

rule(X, MIGRINWARD, ESTABLISHMENT, EMPLOYED, MIGRANTS,
    AREARATIO):- ed(X, MIGRINWARD, ESTABLISHMENT, EMPLOYED,
    MIGRANTS), ESTABLISHMENT=0, EMPLOYED =< 3,
    EMPLOYED > 1, ed_employment_region(X, SDO_GID,
    AREARATIO).

predict(EDs,MIGRANTS):- length(EDs, L), reg_function(EDs, V),
    MIGRANTS is V/L.

reg_function([ ], 0).
reg_function([[EMPLOYED, AREARATIO, MIGRANTS]|T],V):-
    reg_function(T, W), V is W+41.29-4.16*EMPLOYED-
    58.79*AREARATIO.
```

At BK_2 level, Mr-SMOTI mines a multi-relation model tree in 17120 secs partitioning Stockport data into 54 leaves. This confirms that processing census data in the neighborhood strongly increases the time-complexity of mining process but not the size-complexity of mined tree expressed as number of leaves.

An example of regression rule extracted is the following:

```
SELECT T0.ZONE_ID T0ZONE_ID ,
       AVG((14.660273+5.168677*T0.EMPLOYED )+ (6.724149+
           1.4513543*(T1.EMPLOYED -(7.7872725+-0.0508446*
           T0.EMPLOYED )))+ (-8.056492+-1.8275621*((T1.MIGRINWARD
           -(11.272183+-1.2915368*T0.EMPLOYED ))-(-4.5321145+
           0.23710525*(T1.EMPLOYED -(7.7872725+-0.0508446*
           T0.EMPLOYED )))))) T0MIGRANTS ,
FROM ED T0, ED_NEIGHBOUR T1
WHERE T0.ZONE_ID=T1.TAG AND
       T0.ESTABLISHMENT IN (0) AND
```

```
        T0.EMPLOYED <= 3.0 AND
        (T0.MIGRINWARD -(2.6219103+1.1807395*T0.EMPLOYED ))<=
            11.016611 AND
        ((T0.MIGRINWARD -(2.6219103+1.1807395*T0.EMPLOYED ))-
            (-2.5749257+-0.4340402*(T1.EMPLOYED -(7.7872725+-0.0508446*
            T0.EMPLOYED ))))<= 1.8331742 AND
        (T1.MIGRINWARD -(11.272183+-1.2915368*T0.EMPLOYED ))<=
            -0.6891094 AND
        (T1.EMPLOYED -(7.7872725+-0.0508446*T0.EMPLOYED ))<=
            -2.685583 AND
        ((T1.MIGRINWARD -(11.272183+-1.2915368*T0.EMPLOYED ))-
            (-4.5321145+0.23710525*(T1.EMPLOYED -(7.7872725+-0.0508446*
            T0.EMPLOYED ))))<= 1.9208084
    GROUP BY T0.ZONE_ID
```

This regression rule identifies Stockport EDs where census data in neighborhood seems to affect the prediction of number of migrants. The prediction function in SELECT clause is equivalent to "AVG(11.06+ 2.91*T0.EMPLOYED+1.86*T1. EMPLOYED-1.82*T1.MIGRINWARD)". This multiple function is obtained by combining the effect of three straight-line regression: the first straight-line regression involves the number of employees on 10% sample population of each target ED while the latter two regressions involves the number of 10% sample employees and the number of migrants within the ward of some neighboring EDs respectively. In this case T0.EMPLOYED has a global effect. Indeed, looking at corresponding output tree, straight-line regression on T0.EMPLOYED is estimated on the set of Stockport EDs in T0 having no communal establishment and number of employees on 10% sample population that is greater than 3 (T0.ESTABLISHMENT IN (0) AND T0.EMPLOYED <= 3.0) without considering EDs forming the neighborhood. Straight-line regression on T1.EMPLOYED is then performed on a subset of the EDs already mined to estimate regression on T0.EMPLOYED. This subset corresponds to Stockport EDs in T0 having both "T0.ESTABLISHMENT IN (0) AND T0.EMPLOYED <= 3.0 AND (T0.MIGRINWARD -(2.6219103+1.1807395*T0. EMPLOYED))<=11.016611" and one or more neighboring EDs in T1 with "T1. MIGRINWARD-(11.272183+-1.2915368*T0. EMPLOYED))<=-0.6891094 AND (T1.EMPLOYED -(7.7872725+-0.0508446*T0.EMPLOYED))<= -2.685583". Finally, straight-line regression on the residual of T1.MIGRINWARD has an effect that is local to the portion of training data actually falling in the corresponding leaf. It is noteworthy that all continuous splitting tests following a regression step are oblique splits (e.g. T0.MIGRINWARD -(2.6219103+1.1807395* T0. EMPLOYED))<= 11.016611) since they involve continuous residual variables where the effect of performed regressions has been removed.

At BK_3 level, Mr-SMOTI mines a multi-relation model tree in 18425 secs partitioning Stockport data into 49 leaves. An example of regression rule discovered by Mr-SMOTI is the following:

```
SELECT T0.ZONE_ID T0ZONE_ID ,
AVG((14.660273+5.168677*T0.EMPLOYED)+
   (-1.2258624+-0.0458495*(T1.MIGRANTS -(42.826923+-3.1153846*
   T0.EMPLOYED)))) T0MIGRANTS
FROM NEIGHBOUR T1, ED T0
WHERE T0.ZONE_ID=T1.TAG AND T0.ESTABLISHMENT IN (0) AND
   T0.EMPLOYED <= 3.0 AND
   (T1.MIGRINWARD -(11.272183+-1.2915368*T0.EMPLOYED ))
   <= -0.6891094 AND
   T0.ZONE_ID NOT IN (SELECT T1.TAG FROM NEIGHBOUR T1
      (WHERE T1.MIGRINWARD-(11.272183+-1.2915368*
      T0.EMPLOYED))<=-0.6891094 AND (T1.EMPLOYED -(7.7872725
      -0.050844677*T0.EMPLOYED ))<= -2.685583 ) AND
   (T0.MIGRINWARD -(2.6219103+1.1807395*T0.EMPLOYED ))
      <= 11.016611
GROUP BY T0.ZONE_ID
```

This rule models the migration phenomenon on 20 Stockport EDs by exploiting correlation on spatially lagged target variable (i.e. number of migrants) over the neighborhood of each ED. Regression function in SELECT clause is equivalently written as "AVG(5.02*T0.EMPLOYED-0.045*T1.MIGRANTS+15.36)". This function is obtained stepwise by combining two straight-line regressions. The first straight-line regression concerns the number of employees in the 10% sample population of target EDs (T0.EMPLOYED). Looking at tree, we dicover that slope (14.660273) and intercept (5.168677) of this straight-line regression are estimated on the set of 66 target EDs (T0) having no communal establishment and low number of employees on 10% sample population (T0.ESTABLISHMENT in (0) AND T0.EMPLOYED<=3). Regression on T0.EMPLOYED has clearly a global effect and its contribution is equally shared by all regression functions associated with leaves of the sub-tree rooted in the corresponding regression node. The second straigth-line regression concerns the residual variable associated with number of migrants (T1.MIGRANTS-42.826923+3.1153846*T0.EMPLOYED) in neighbouring EDs (T1) having the residual value of number of migrants within ward less than -0.6891094 (T1.MIGRINWARD-11.272183+1.2915368*T0.EMPLOYED<= -0.6891094). It is computed on a proper subset of the EDs previously mined to estimate the straight-line regression involving the number of sample employees in target ED as predictor variable. This means that number of migrants in neighbourhood has only a local effect.

Predictive accuracy and size complexity of multi-relational model trees mined with Mr-SMOTI are estimated according to a 10-fold cross validation of Stockport data by varying the level BK_i. Both accuracy and complexity are compared with corresponding values obtained with multi-relational model trees mined by TILDE-RT on Prolog facts representation of same data. We also evaluate predictive accuracy and complexity of model trees mined with SMOTI and M5' on attribute-value

representation of Stockport data obtained according to P1 and P2. All these results are reported in Table 6.1 while results of the Wilcoxon test on the predictive accuracy performed by Mr-SMOTI with respect to the other regression methods in this study are presented in Table 6.2.

Three main conclusions can be drawn from these experimental results. First, Mr-SMOTI appears to be able to detect both local and global effects represented in (multi-)relational data and it is more accurate than TILDE-RT which approximates the model at leaf with a simple constant value. Second, Mr-SMOTI is able to exploit the relational structure of data generally resulting in a better accuracy results with respect to the propositional method SMOTI. Moreover M5' combined with the P1 and P2 attribute-value data transformation sometime results in regression models which have accuracy approximately equivalent to Mr-SMOTI predictive accuracy, although model trees mined with M5' are significantly more complex than the corresponding (multi-)relational model trees. Third the propositional SMOTI builds model trees with a greater number of regression nodes than Mr-SMOTI. This can be explained by the fact that SMOTI transforms splitting nodes in regression nodes when two lines associated with the children of a splitting node are equal with respect to a statistical test [MECA04]. In Mr-SMOTI the (multi-)relational structure of data does not permit performing the statistical test for coincident regression lines [Wei85], hence more splitting tests should be expected.

TABLE 6.1: Performance of predicting the number of migrants from Stockport data. Results of Mr-SMOTI and TILDE-RT are obtained by running the methods on (multi-)relational data, while results of SMOTI and M5' are obtained by running the methods on the attribute-value repreresentation of same data obtained according to P1 and P2.

BK	Results	Multi-relational setting		Propositional setting			
				P1		P2	
		Mr-SMOTI	TILDE-RT	SMOTI	M5'	SMOTI	M5'
BK_1	Avg.MSE	15.39	16.71	18.50	15.24	15.36	21.31
	Avg.Leaves	46.5	2.9	48.3	185.3	48.3	178.5
	Avg.Regnodes	3.4	-	5.3	-	4.8	-
BK_2	Avg.MSE	15.33	16.59	24.50	17.38	16.32	18.75
	Avg.Leaves	48.5	2	108.8	939.5	50.7	226.2
	Avg.Regnodes	5.3	-	14.4	-	8.6	-
BK_3	Avg.MSE	15.436	16.59	22.06	17.38	19.49	16.26
	Avg.Leaves	44.4	2	120.8	989.4	48.6	227.1
	Avg.Regnodes	7.2	-	14.1	-	9.2	-

6.2 Applications to Bioinformatics

Multi-relational regression has some interesting applications within bioinformatic domains, where ILP methods have been already applied on drug designing, pre-

TABLE 6.2: Mr-SMOTI versus TILDE-RT, SMOTI and M5': results of Wilcoxon rank test on the accuracy of regression models induced from Stockport data.

BK	Method	Wilcoxon Test		
		p	W_+	W_-
BK_1	TILDE-RT	0.1055	11	44
	P1-SMOTI	0.1309	12	43
	P1-M5'	0.9219	26	29
	P2-SMOTI	0.0019	0	55
	P2-M5'	0.9219	29	26
BK_2	TILDE-RT	0.0371	7	48
	P1-SMOTI	0.0097	3	52
	P1-M5'	0.0136	4	51
	P2-SMOTI	0.0644	9	46
	P2-M5'	0.4116	19	36
BK_3	TILDE-RT	0.01137	4	51
	P1-SMOTI	0.03711	7	48
	P1-M5'	0.0039	1	54
	P2-SMOTI	0.03711	7	48
	P2-M5'	0.08398	10	45

dicting mutagenicity and carcinogenicity as well as predicting protein structure and function including genome scale prediction of protein functional class [Dŏ1]. In this Section, we discuss two applications in the area of quantitative structure activity relationships (QSARs), namely Mutagenesis and Biodegradability. Both applications involve molecular databases where the structure of molecules is mined to understand complex molecular properties (i.e. mutagenecity in nitro-aromatic compounds and biodegradability activity in chemicals compounds of water).

Graph mining methods are adopted to mine this type of data. They typically derive interesting frequent, patterns and then use these as features to build predictive models. Several approaches have been suggested [YH02] [DKK03] [IK03] [KD04] for the task of identifying fragments which can be used to build such models. The earliest approaches to compute such fragments are based on techniques from ILP [Deh98] which is here theoretically appealing because of the use of expressive representation languages.

This data has been also adopted as ILP benchmark datasets for multi-relational regression and some results are discussed in [Blo98] [Kra96] [KWPd01].

6.2.1 Experiments on Mutagenesis

Mutagenesis data [SMKS94] consists of structural descriptions of 230 molecules of aromatic and heteroaromatic nitro compounds. The regression problem here is to predict the mutagenic activity of these molecules, that is measured by a real number. Since mutagenecity is the ability to cause DNA to mutate, predicting the mutagenic activity of molecules is a relevant issue in the understanding and prediction of

carcinogenesis. In this study, the data to be analyzed are based on results reported in [DLD+91] where the mutagenic activity of molecules is estimated according to the Ames test using Styphimurium TA98. The original study of Debnath and his colleagues recognizes two subsets of data: 188 (regression friendly) molecules for which linear regression yields good results and 42 molecules that are regression-unfriendly.

In our experiments, we analyze relational descriptions of the 188 regression friendly molecules consisting of atoms and their bond connectivities stored in three different tables of a relational database that is Molecule (target table), Atom and Bond. According to a recent study on this data [SKM99], we consider three level of experimental setting for mutagenesis, which can provide richer descriptions of molecules. The first setting (BK_1) consists of those data derived with the molecular modeling package QUANTA. For each molecule it obtains the atoms, bonds, bond types, atom types, and partial charges on atoms. The second setting (BK_2) adds a boolean attribute identifying compounds with 3 or more benzyl rings (termed indicator variable Ind1) and a boolean attribute identifying a sub-class of compounds termed acenthryles (termed indicator variable IndA). The third setting (BK_3) adds information on the hydrophobicity of molecules (termed logP) and the energy level of the lowest unoccupied molecular orbital (termed LUMO). The greater the BK_i (i = 1, 2, 3) the more complex the regression problem with $BK_1 \subset BK_2 \subset BK_3$.

By running Mr-SMOTI on the entire set of data stored in an Oracle $9i$ database, it builds the output multi-relational model tree in 1653 secs at BK_1 level, 1126 secs at BK_2 level and 45 secs at BK_3 level. This suggests that at higher levels, namely BK_2 and BK_3, we add information that is strongly relevant for the regression task and permits a more rapidly converge toward the output tree.

TABLE 6.3: Performance of predicting mutagenesis level from Mutagenesis data: results of Mr-SMOTI and TILDE-RT are obtained by running the methods on multi-relational data, while results of SMOTI and M5' are obtained by running the methods on the attribute-value representations of same data obtained according to P1 and P2.

BK	Results	Multi-relational setting		Propositional setting			
				P1		P2	
		Mr-SMOTI	TILDE-RT	SMOTI	M5'	SMOTI	M5'
BK_1	Avg.MSE	1.79	1.51	71.55	1.72	3.17	1.61
	Avg.Leaves	18.2	9	134.4	654.3	20.3	67.7
	Avg.Regnodes	1.50	-	18.2	-	4.9	-
BK_2	Avg.MSE	1.14	1.19	8.97	1.28	1.23	1.15
	Avg.Leaves	15.4	11.70	137.3	692.7	19.8	67.7
	Avg.Regnodes	1.00	-	14.8	-	4.3	-
BK_3	Avg.MSE	0.88	1.19	1.39	1.02	1.28	1.03
	Avg.Leaves	4.80	14.90	80.9	140.3	15.5	71.4
	Avg.Regnodes	2.00	-	17.5	-	8.8	-

TABLE 6.4: Mr-SMOTI versus TILDE-RT, SMOTI and M5': results of Wilcoxon rank test on the accuracy of regression models induced from Mutagenesis data.

BK	Method	Wilcoxon Test		
		p	W_+	W_-
BK_1	TILDE-RT	0.2324	40	15
	P1-SMOTI	0.0371	7	48
	P1-M5'	0.6953	32	23
	P2-SMOTI	0.8457	25	30
	P2-M5'	0.4922	35	20
BK_2	TILDE-RT	0.4922	20	35
	P1-SMOTI	0.0039	1	54
	P1-M5'	0.0371	7	48
	P2-SMOTI	0.3750	18	37
	P2-M5'	0.9219	26	29
BK_3	TILDE-RT	0.0039	1	54
	P1-SMOTI	4.0000	4	51
	P1-M5'	0.0839	10	45
	P2-SMOTI	0.0371	7	48
	P2-M5'	0.0839	10	45

Predictive accuracy and complexity of multi-relational model trees built by Mr-SMOTI are estimated according to a 10-fold cross validation of Mutagenesis data by varying the level BK_i. These results as well as the corresponding results obtained by running TILDE-RT, SMOTI and M5' are reported in Table 6.3. Results of the Wilcoxon test on the predictive accuracy performed by Mr-SMOTI with respect to the other regression methods in this study are presented in Table 6.4.

These results show that at BK_1 level, Mr-SMOTI predictive accuracy is statistically equivalent to the accuracy of regression models built by both TILDE-RT and M5', while it is significantly better than the accuracy performed by SMOTI in both attribute-value setting (P1 and P2). In addition, except for TILDE-RT, regression models built by Mr-SMOTI are generally simpler than the corresponding models built by propositional methods. At BK_2 level, results on average square radix *MSE* prove that Mr-SMOTI can perform better than TILDE-RT that labels each leaf of a tree-based predictor with a constant value and the propositional methods SMOTI and M5' that do not exploit the relational structure of training data. These conclusions are significantly confirmed by results collected at BK_3 level.

6.2.2 Experiments on Biodegradability

This dataset consists of 328 chemical molecules whose structural description in terms of atoms and bonds is derived from the chemicals SMILES encodings. The SMILES notation contains information on the two-dimensional structure of each molecule. Therefore, an atom-bond representation similar to representation adopted in experiments to predict mutagenecity is generated with SMILES encoding of a

chemical molecule. Note that this atom-bond representation is quite different from the QUANTA-derived representation that comprises atom charges in addition to atom types and bond type. Conversely, SMILES-derived representation includes only bond type and atom element. Biodegradability dataset also contains information about the number of times some FG bounds occurs in a molecule (e.g. ssCount(moleculeId, fgName, count)) as well as the number of times certain small substructures occur in a molecule (e.g. p2count(moleculeId, substructureId, count)).

This data structure can be naturally expressed by means of five tables of an Oracle 9*i* relational database that is Molecule (the target table), Atom, Bond, SSCount and P2Count while associations among them are modeled by means of foreign key constraints. Data concerning molecule activity are stored in the target table. Other global features of a molecule are mWeight (i.e. the molecular weight) and logP (i.e. the logarithm of the compound's octanol/water partition coefficient). In particular, logP is a measure of hydrophobicity that is used in the mutagenecity study.

According to data described above, the aim of this study is to predict the biodegradability of the chemical compound in water [DBK+99]. Generally, the biodegradability of a molecule is described by different degradation rates that are available in literature [HBJ+91] in the form of half-life times (HLTs) for overall, biotic and a-biotic degradation in four environmental comportments that is soil, air, surface water and ground water. In this study, we focus on predicting the surface water biodegradation HLT's in aerobic conditions. The target variable for this regression problem is the natural logarithm of the arithmetic mean of the low and high estimate of the HLT for aqueous biodegradation in aerobic conditions, measured in hours.

Similarly to Mutagenesis dataset, we may consider four level of experimental setting for biodegradability data which provide richer descriptions of molecules. The first setting (BK_1) consists of those data derived with SMILES without any global feature on molecule. The second setting (BK_2) adds the numerical attributes mWeight and logP. The third setting (BK_3) extends BK_1 by adding the indicator on molecular activity, while the fourth setting (BK_4) includes all global features describing the molecules.

By running Mr-SMOTI on the entire dataset, it builds the output multi-relational model tree in 8475 secs at BK_1 level, 18221 secs at BK_2 level, 367 secs at BK_3 level and 1746 at BK_4 level.

Predictive accuracy as well as complexity of regression models built by Mr-SMOTI, TILDE-RT, SMOTI and M5' are reported in Table 6.5 by varying the level BK_i, while results of Wilcoxon test on the predictive accuracy performed by Mr-SMOTI with respect to the other regression methods in this study are presented in Table 6.6.

In this case, results on predictive accuracy show that at BK_1 level, Mr-SMOTI is statistically equivalent to TILDE-RT, while at BK_2 level Mr-SMOTI looses against TILDE-RT. Interestingly both methods outperform SMOTI and M5'. Conversely, at BK_3 and BK_4 levels, Mr-SMOTI outperforms TILDE-RT, but its accuracy is significantly worse than corresponding results obtained with propositional regression methods.

Results at BK_1 and BK_2 levels confirm some consideration already suggested by Mutagenesis data: a constant piecewise multi-relational model can be more accurate than the corresponding multiple linear piecewise multi-relational model when no continuous attribute effectively significant for the regression problem in question is included in training data. Conversely, BK_3 and BK_4 levels confirm that the stepwise construction of a regression model may lead to significantly more accurate regression model both in multi-relational and propositional setting. In this particular case, propositionalization by aggregation combined with SMOTI miner outperforms Mr-SMOTI, but the propositional model tree is significantly more complex (number of regression nodes and number of leaves) than the corresponding multi-relational model trees.

TABLE 6.5: Performance of predicting degradation level from Biodegradability data: results of Mr-SMOTI and TILDE-RT are obtained by running the methods on multi-relational data, while results of SMOTI and M5' are obtained by running the methods on the attribute-value representation of same data obtained according to P2. The transformation of Biodegradability data with P1 method returns 1506060 attribute-value individuals that is quite hard to be processed with SMOTI or M5'.

BK	Results	Multi-relational setting		Propositional setting	
				P2	
		Mr-SMOTI	TILDE-RT	SMOTI	M5'
BK_1	Avg.MSE	1.38	1.31	2.24	1.44
	Avg.Leaves	38.6	7	34.1	108
	Avg.Regnodes	0.3	-	1.1	-
BK_2	Avg.MSE	1.42	1.23	4.27	1.54
	Avg.Leaves	29.3	7.8	32.5	112.3
	Avg.Regnodes	4.6	-	5	-
BK_3	Avg.MSE	0.37	0.58	0.05	0.17
	Avg.Leaves	2.3	4.6	13.1	38.6
	Avg.Regnodes	0	-	1.4	-
BK_4	Avg.MSE	0.38	0.58	0.04	0.174
	Avg.Leaves	2.7	4.7	13.3	39.5
	Avg.Regnodes	0	-	2	-

6.3 Conclusions

In this chapter, we have discussed three real-world applications of multi-relational regression.

The first one concerns the analysis of geo-referenced census data provided by United Kingdom 1991 census to investigate the migration phenomenon in Stockport enumeration districts (EDs) according to socio-economic factors represented in census data as well as geographical factors (spatial layers) represented in topographic maps. In this case, multi-relational regression extends classical spatial regression by supporting data miners in modeling and processing spatial relationships among spatial objects of different nature too. The goal is to predict a continuous property

TABLE 6.6: Mr-SMOTI versus TILDE-RT, SMOTI and M5': results of Wilcoxon rank test on the accuracy of regression models induced from Biodegradability data.

BK	Method	Wilcoxon Test		
		p	W_+	W_-
BK_1	TILDE-RT	0.4922	35	20
	P2-SMOTI	0.0488	8	47
	P2-M5'	0.1934	14	41
BK_2	TILDE-RT	0.0097	52	3
	P2-SMOTI	0.0019	0	55
	P2-M5'	0.6953	23	32
BK_3	TILDE-RT	0.0058	2	53
	P2-SMOTI	0.0019	55	0
	P2-M5'	0.0039	54	1
BK_4	TILDE-RT	0.0058	2	53
	P2-SMOTI	0.0019	55	0
	P2-M5'	0.0019	55	0

on the basis of the spatial interaction of two or more spatially-referenced objects not necessarily belonging to the same layer or space-dependent attributes according to a particular spacing or set of arrangements. In particular, regression models mined by Mr-SMOTI are model trees with regression and splitting nodes that may involve attributes from several spatially related layers. This means that both spatial and a-spatial factors which have either a global or local effect for prediction are easily detected by exploiting the stepwise construction of the multi-relational regression model.

The last two applications are related to bioinformatics and concern the quantitative interpretation of structure-activity relationships of chemical compounds (QSARs) in two well-known ILP benchmark datasets that is Mutagenesis and Biodegradability.

The comparison with the ILP method TILDE-RT and the two propositional TDIMT methods SMOTI and M5' on different (multi-)relational datasets demonstrate that Mr-SMOTI is generally competitive with respect to existing regression methods and its advantages (e.g. the solution of (multi-)relational regression problem in its original representation and the detection of global and local effect of variables to also increase the comprehensibility of regression model) are not at the expense of predictive accuracy or model complexity.

Chapter 7

Conclusions

In this dissertation we have presented both a new TDIMT method, named SMOTI, for the data-driven construction of model trees from attribute-value data and its further extension, Mr-SMOTI, to multi-relational setting. We conclude with an overall summary of our claimed contributions and discussion of future work.

7.1 Summary

Model trees are an extension of regression trees that associate leaves with multiple regression models. In this dissertation we have revised the current state of top-down induction of model trees (TDIMT) in order to improve interpretability and accuracy of these regression models as well as extend their applicability to practical problems where data is naturally stored in multiple tables of a relational database.

We have started from the strengths and weaknesses of well known data-driven approaches for mining model trees and proposed a new method named Stepwise Model Tree Induction (SMOTI). Its main characteristic is the induction of trees with two types of nodes: regression nodes, which perform only straight-line regression, and splitting nodes, which partition the training space. The multiple linear model associated with each leaf is then built stepwise by combining straight-line regressions reported along the path from the root to the leaf. In this way, SMOTI solves the problem of modeling phenomena where some variables have a global effect while others have only a local effect. Internal regression nodes contribute to the definition of multiple models and have a global effect, while straight-line regressions at leaves have only local effect. This stepwise construction allows SMOTI, at no additional cost, to define a heuristic evaluation function, which is coherent with the linear models at the leaves. On the other hand, only a subset of continuous attributes may be involved in multiple linear models associated with the leaves, thus solving problems due to collinearity.

To keep the overfitting problem under control, we have proposed to simplify model trees mined by SMOTI with two alternative methods, namely Reduced Error Pruning (REP) and Reduced Error Grafting (REG).

Experimental results on artificially generated datasets show that SMOTI outper-

forms two state of art model tree induction systems, M5' and RETIS, in accuracy. Results on benchmark datasets used for studies on both regression and model trees show that SMOTI performs better than RETIS in accuracy, while it is not possible to draw statistically significant conclusions on the comparison with M5'. In any case, model trees induced by SMOTI are generally simple and easily interpretable, and their analysis often reveals interesting patterns which are not evident in trees generated by the other systems.

We have also investigated the effect of simplification on benchmark datasets and results are in favor of simplified trees in most cases.

Nevertheless, SMOTI suffers from an important limitation, that is, the restriction to propositional representation (i.e. single table assumption). In other words, training data must be described by a fixed set of attributes, each of which can have only a single, primitive value. Consequently, SMOTI does not appear able to directly mine model trees over data that reside in multiple tables. This implies that aspects of the internal structure of training data cannot be processed and mined trees cannot refer to such a structural property. This might comprise the application of model trees in domains where the internal structure of individual to be mined is of principal importance (e.g. geo-referenced data analysis, chemistry or biology).

To deal with data scattered over many tables, we have explored the idea of combining the stepwise construction supported by SMOTI with achievements of relational data mining in order to overcome limitations due to single table assumption. Hence, we have illustrated how to upgrade the propositional SMOTI to the multi-relational miner named Multi-Relational Stepwise Model Tree Induction (Mr-SMOTI) and mine multi-relational model trees directly from data, which resides in multiple tables of a tightly integrated Oracle $9i$ relational database. Patterns associated with each node of the tree structure are (multi-)relational patterns since they may involve multiple tables from the training relational database. They are represented in the graphical language of regression selection graphs, which can be translated into SQL, or equivalently into first order expressions.

Finally, we have discussed the application of Mr-SMOTI into some multi relational regression problems occurring in geo-referenced census data analysis and bioinformatics. The former involves data provided by the United Kingdom 1991 census where the goal is to investigate the migration phenomena (i.e. number of migrants) in Stockport enumeration districts (EDs) according to socio-economic factors represented in census data as well as geographical factors (spatial layers) represented in topographic maps. In this case, multi-relational regression well copes with the main issue of spatial regression, that is, the need to represent both units of observation and their relationships. The former are spatial objects sometime belonging to different layers, while the latter are useful to predict a continuous property. The bioinformatics application focuses on two multi-relational benchmark datasets that is Mutagenesis and Biodegradability, which have been extensively used in ILP. Mutagensis concerns the problem of predicting mutagenic activity of molecules while Biodegradability concerns the problem of investigating biodegradability in an aqueous environment under aerobic conditions.

7.2 Future Work

As future work, we plan to use SQL primitives and parallel database servers to speed up the stepwise construction of (multi-)relational model trees from data stored in large database. In addition, following the mainstream of our research on data mining query languages for spatial databases with an object-oriented logical model [MAC03], we intend to pursue the investigation of defining a data mining query language appropriate to support both the discovery and the query of model trees.

Similar to many TDIMT algorithms, Mr-SMOTI may generate model trees that over-fit training data. Therefore, another future research direction is also the *a posteriori* simplification of (multi-)relational model trees with both regression nodes and splitting nodes. We plan to investigate simplification methods based on both pruning and grafting operators.

We are also interested in applying (multi-)relational model trees mined by Mr-SMOTI to the problem of predicting ordinal classes that is discrete classes with a linear ordering as suggested by Kramer et al. [KWPd01].

Finally, applications to spatial domains suggest a further extension of Mr-SMOTI in order to take advantage from a tight coupling with an object-relational spatial database where both target objects as well as target relevant objects are spatial objects belonging to several spatial layers stored in spatial tables. This corresponds to modify the space of patterns explored during the tree construction by taking into account spatial nature of data and explore spatial properties (e.g. area or length) as well as spatial topological relationships (e.g. adjacent or overlap) implicitly defined by the location of spatial objects in a space.

Bibliography

[ACL+03] A. Appice, M. Ceci, A. Lanza, F. A. Lisi, and D.Malerba. Discovery of spatial association rules in georeferenced census data: A relational mining approach. *Intelligent Data Analysis*, 7(6):541–566, 2003.

[ACM02] A. Appice, M. Ceci, and D. Malerba. KDB2000: An integrated knowledge discovery tool. In A. Zanasi, C. A. Brebbia, N.F.F. Ebecken, and P. Melli, editors, *Data Mining III*, volume 6 of *Management Information Systems*, pages 531–540. Wessex Institute of Technology, 2002.

[ACM03] A. Appice, M. Ceci, and D. Malerba. Mining model trees: A multi-relational approach. In T. Horvath and A. Yamamoto, editors, *Inductive Logic Programming, 13th International Conference, ILP 2003*, volume 2835 of *LNAI*, pages 4–21. Springer-Verlag, 2003.

[ADED04] A. Appice, C. D'Amato, F. Esposito, and D.Malerba. k-nearest neighbour classification of symbolic objects. In P. Brito and M. Noirhomme-Fraiture, editors, *Proceedings of Workshop on Symbolic and Spatial Data analysis: Mining Complex Data Structures in conjunction with the 15th European Conference on Machine Learning, ECML 2004, and the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 2004*, pages 19–30, Pisa, Italy, 2004.

[Aha97] D. W. Aha. *Lazy Learning*. Kluwer Academic Publisher, 1997.

[AKA91] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[ALH03] A. Atramentov, H. Leiva, and V. Honovar. A multi-relational decision tree learning algorithm. In T. Horvath and A. Yamamoto, editors, *Inductive Logic Programming, 13th International Conference, ILP 2003*, volume 2835 of *LNAI*, pages 38–56. Springer-Verlag, 2003.

[AMS97] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted leraning. *Artificial Intelligence Review, Special Issue on Lazy Learning*, 11:11–73, 1997.

[BB03] J. Bi and K. P. Bennett. Regression error characteristic curves. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning, ICML 2003*, pages 43–50, Washington, USA, 2003. AAAI Press.

[BD96] H. Blockeel and L. De Raedt. Relational knowledge discovery in databases. In T. B. Pfahringer and J. Fuernkranz, editors, *Proceedings of MLnet Familiarization Workshop on Data Mining with Inductive Logic Programming In conjunction with the 13th International Conference on Machine Learning*, pages 111–124, Bari, Italy, 1996.

[BD98] H. Blockeel and L. De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.

[Ben75] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[BFOS84] L. Breiman, J. Friedman, R. Olshen, and J. Stone. *Classification and regression tree*. Wadsworth & Brooks, 1984.

[BKM99] J.F. Boulicot, M. Klemettinen, and H. Mannila. Modeling KDD processes within the indutive database framework. In *Data Warehouse and Knowledge Discovery , 1th International Conference*, volume 1676 of *LNAI*, pages 293–302. Springer-Verlag, 1999.

[BL01] A. Buja and Y.S. Lee. Data mining criteria for tree-based regression and classification. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2001*, pages 27–36, San Francisco, USA, 2001.

[Blo98] H. Blockeel. *Top-down induction of first order logical decision trees*. PhD thesis, Department of Computer Science, Katholieke Universiteit, Leuven, Belgium, 1998.

[Bos95] H. Bostrom. Covering vs divide-and-conquer for top-down induction of logic programs. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI 1995*, volume 2, pages 1194–1200, Montréal, Québec, Canada, 1995. Morgan Kaufmann.

[Bre96] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[BT90] P. Brazdil and L. Torgo. Knowledge acquisition via knowledge integration. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber, and M. van Someren, editors, *Current Trends in Knowledge Acquisition*, volume 8 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 1990.

[CAM03a] M. Ceci, A. Appice, and D. Malerba. Comparing simplification methods for model trees with regression and splitting nodes. In N.Zong, Z.W. Ras, S. Tsumoto, and E. Suzuki, editors, *Foundations of Intelligent Systems, 14th International Symposium, ISMIS 2003*, volume 2871 of *LNAI*, pages 49–56. Springer-Verlag, 2003.

[CAM03b] M. Ceci, A. Appice, and D. Malerba. Mr-SBC: a multi-relational naive bayes classifier. In N. Lavrac, D. Gamberger, L. Todorovski, and H. Blockeel, editors, *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 2003*, volume 2838 of *LNAI*, pages 95–106. Springer-Verlag, 2003.

[CAM03c] M. Ceci, A. Appice, and D. Malerba. Simplification methods for model trees with regression and splitting nodes. In P. Perner and A. Rosenfeld, editors, *IAPR International Conference on Machine Learning and Data Mining for Pattern Recognition, MLDM 2003*, volume 2734 of *LNAI*, pages 53–64. Springer-Verlag, 2003.

[CAM04] M. Ceci, A. Appice, and D. Malerba. Spatial associative classification at different levels of granularity: A probabilistic approach. In J. F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 2004*, volume 3202 of *LNAI*, pages 99–111. Springer-Verlag, 2004.

[Cat91] J. Catlett. On changing continuous attributes into ordered discrete attributes. In *Proceedings of the Fifth European Working Session on Learning*, pages 164–178, 1991.

[CB91] B. Cestnik and I. Bratko. On estimating probabilities in tree pruning. In *Proceedings of the Fifth European Working Session on Learning*, pages 151–163, 1991.

[Ces90] B. Cestnik. Estimating probabilities: A crucial task in machine learning. In J. P. Martins and M. Reinfrank, editors, *Proceedings of the 9th European Conference on Artificial Intelligence, ECAI 1990*, volume 515 of *LNAI*, pages 147–149. Springer-Verlag, 1990.

[CHLY94] P. Chaudhuri, M. C. Huang, W. Y. Loh, and R. Yao. Piecewise-polynomial regression trees. *Statistica Sinica*, 4:143–167, 1994.

[CL96] W. Cleveland and C. Loader. *Statistical theory and computational aspects of smoothing*, chapter Smoothing by Local Regression: Principles and Methods, pages 10–49. Physica-Verlag, 1996.

[CN89] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.

[Cod72] E. F. Codd. Relational completeness for data base languages. *Data Base Systems*, 6:65–98, 1972.

[Cod93] E.F. Codd. *Providing OLAP (On-Line Analytical Processing) to User-Analysts*. An IT Mandate. E.F. Codd and Associates, 1993.

[Cre93] N. A. C. Cressie. *Statistics for spatial data*. Wiley, New York, 1993.

[CU87] M. Connel and P. Utgoff. Learning to control a dynamical physical system. In *Proceedings of the 6th National Conference on Artificial Intelligence*. Morgan Kaufmann, 1987.

[Das91] B. V. Dasarathy. *Nearest Neighbor(NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1991.

[DBK+99] S. Džeroski, H. Blockeel, S. Kramer, B. Kompare, B. Pfahringer, and W. Van Laer. Experiments in predicting biodegradability. In S. Džeroski and P. Flach, editors, *International Workshop on Inductive Logic Programming ILP 1999*, volume 1634 of *LNAI*, pages 80–91. Springer-Verlag, 1999.

[DD97] L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26:99–146, 1997.

[De 96] L. De Raedt. Induction in logic. In R. S. Michalski and J. Wnek, editors, *Proceedings of the 3rd International Workshop on Multistrategy Learning*, pages 29–38, 1996.

[De 97] L. De Raedt. Mining association rules in multiple relations. In N. Lavrač and S. Džeroski, editors, *Inductive Logic Programming, 7th International Workshop, ILP 1997*, volume 1297 of *LNAI*, pages 125–132. Springer-Verlag, 1997.

[De 02] L. De Raedt. A perspective on inductive databases. *ACM SIGKDD Explorations Newsletter*, 4(2):69–77, 2002.

[Deh98] L. Dehaspe. *Frequent Pattern Discovery in First-Order Logic*. PhD thesis, Department of Computer Science, Katholieke Universiteit, Leuven, Belgium, 1998.

[DeM00] M. N. DeMers. *Fundamentals of Geographic Information Systems*. John Wiley & Sons, New York, second edition edition, 2000.

[Den91] P. Densham. Spatial decision support systems. *Geographical Information Systems: principles and applications*, pages 403–412, 1991.

[DG02] A. Dobra and J. E. Gehrke. SECRET: A scalable linear regression tree algorithm. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, 2002.

[DKK03] M. Deshpande, M. Kuramochi, and G. Karypis. Frequent sub-structure-based approaches for classifying chemical compounds. In *Proceedings of the 3rd IEEE International Conference on Data Mining, ICDM 2003*, pages 35–42, Melbourne, Florida, USA, 2003.

[DL01a] S. Džeroski and N. Lavrač. *Relational Data Mining*. Springer-Verlag, 2001.

[DL01b] S. Džeroski and N. Lavrač. *Relational Data Mining*, chapter An introduction to Inductive Logic Programming, pages 48–73. LNAI. Springer-Verlag, Berlin Heidelberg Germany, 2001.

[DLD+91] A.K. Debnath, R.L. Lopez de Compadre, G. Debnath, A.J. Shusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *J. Med. Chem*, 34:786–797, 1991.

[DM95] K. Deng and A. W. Moore. Multiresolution instance-based learning. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI 1995*, volume 2, pages 1233–1242. Morgan Kaufmann, 1995.

[Dom99] P. Domingos. The role of Occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3(4):409–425, 1999.

[DP97] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zeo-ones loss. *Machine Learning*, 28(2-3):103–130, 1997.

[DS82] N. R. Draper and H. Smith. *Applied regression analysis*. John Wiley & Sons, 1982.

[DT95] S. Džeroski and L. Todorovski. Discovering dynamics: from inductive logic programming to machine discovery. *Journal of Intelligent Information Systems*, 4:89–108, 1995.

[DTU95] S. Džeroski, L. Todoroski, and T. Urbancic. Handling real numbers in inductive logic programming: A step towards better behavioural clones. In N. Lavrač and S. Wrobel, editors, *European Conference of Machine Learning, ECML 1995*, volume 912 of *LNAI*, pages 283–286. Springer-Verlag, 1995.

[Dž95] S. Džeroski. *Numerical Constraints and Learnability in Inductive Logic Programming*. PhD thesis, Faculty of Electrical Engineering and Computer Science, Ljubljana, Slovenia, 1995.

[Dž01] S. Džeroski. *Relational Data Mining*, chapter Relational data mining applications: an overview, pages 339–36. LNAI. Springer-Verlag, Berlin Heidelberg Germany, 2001.

[DW95] L. De Raedt and W Wan Laer. Inductive constraint logic. In K. P. Jantke, T. Shinohara, and T. Zeugmann, editors, *Proceedings of the 6th International Conference on Algorithmic Learning Theory, ALT 1995*, volume 997 of *LNAI*, pages 80–94. Springer-Verlag, 1995.

[EF84] P. Ein Dor and J. Feldmesser. Attributes of the performance of central processing units: A relative performance prediction model. *Communications of the ACM*, 30(87):308–317, 1984.

[Efr79] B. Efron. Bootstrap methods: another look at the jacknife. *Annals of Statistics*, 7(1):1–26, 1979.

[Ege91] M. J. Egenhofer. Reasoning about binary topological relations. In *Proceedings of the 2nd Symposium on Large Spatial Databases, VLDB 1991*, pages 143–160, Zurich, Switzerland, 1991.

[EK01] T. Elomaa and Kääriäinen. An analysis of reduced error pruning. *Journal of Artificial Intelligence Research*, 15:163–187, 2001.

[EMS97] F. Esposito, D. Malerba, and G. Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):476–491, 1997.

[ESF01] M. G. Elfeky, A. A. Saad, and S. A. Fouad. Odmql: Object data mining query language. In *Proceedings of Symposium on Objects and Databases*, Sophia Antipolis, France, 2001.

[ET93] B. Efron and R. Tibshirani. *An introduction to the bootstrap*. Chapman & Hall, 1993.

[Eub88] R. L. Eubank. *Spline smoothing and nonparametric regression*. Marcel Dekker, 1988.

[Fan95] J. Fan. Local modelling. *Encyclopedia of Statistics Science*, 1995.

[FF03] J. Fürnkranz and P. A. Flach. An analysis of rule evaluation metrics. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning, ICML 2003*, pages 202–209, Washington, USA, 2003. AAAI Press.

[FL96] A. A. Freitas and S. H. Lavington. Using SQL primitives and parallel db servers to speed up knowledge discovery in large relational databases. In R. Trappl, editor, *Proceedings of the 13th European Meeting on Cybernetics and Systems Research, Cybernetics and Systems 1996*, pages 955–960, Vienna, Austria, 1996.

[FP99] T. Fawcett and F.J. Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining, KDD 1999*, pages 53–62, San Diego, USA, 1999.

[FPM91] W. Frawley, G. Piatesky-Shapiro, and C. Matheus. *Knowledge Discovery in databases*, chapter Knowledge discovery in dtabases: an overview, pages 1–27. MIT Press, Cambridge, MA, 1991.

[FPS96] U. Fayyad, G. Piatesky-Shapiro, and P. Smyth. Knowledge discovery and data mining: Towards a unifying framework. In E. Simoudis, J. Han, and U. Fayyad, editors, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, KDD 1996*, pages 82–88, Portland, Oregon, 1996.

[FPSSU96] U. Fayyad, G. Piatesky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. Mit Press, 1996.

[Fri91] J. H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19(1):1–141, 1991.

[FS81] J. H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical*, 76(376):817–823, 1981.

[FWI+98] E. Frank, Y. Wang., S. Inglis., G. Holmes, and I.H. Witten. Using model trees for classification. *Machine Learning*, 32:63–76, 1998.

[Gam04] J. Gama. Functional trees. *Machine Learning*, 55(3):219–250, 2004.

[Get01] L. Getoor. Multi-relational data mining using probabilistic relational models: research summary. In A. Knobbe and D. M. G. Van der Wallen, editors, *Proceedings of the 1st Workshop in Multi-relational Data Mining*, Freiburg, Germany, 2001.

[Goo65] I.J. Good. *The Estimation of Probabilities.* MIT press, 1965.

[GRM03] J. Gama, R. Rocha, and P. Medas. Accurate decision trees for mining high-speed data streams. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutso, editors, *Proceedings of the 9th ACM SIGKDD International Conference in Knowledge Discovery and Data Mining, KDD 2003*, pages 517–522, Washington, USA, 2003. ACM Press.

[GS94] P. J. Green and B. W. Silverman. *Nonparametric regression and generalized linear models.* Chapman and Hall, London, UK, 1994.

[Güt94] R.H. Güting. An introduction to spatial database systems. *VLDB Journal*, 4(3):357–399, 1994.

[GW02] G. Gora and A. Wojna. RIONA: A classifier combining rule induction and k-NN method with automated selection of optimal neighbourhood. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *European Conference on Machine Learning, ECML 2002*, volume 2430 of *LNAI*, pages 111–123. Springer-Verlag, 2002.

[GWJ96] Z. Ghahramani, D. M. Wolpert, and M.I. Jordan. Generalization to local remapping of the visuo-motor coordinate transformation. *Journal of Neuroscience*, 1996.

[Hai90] R. Haining. *Spatial data analysis in the social and environmental sciences.* Cambridge University Press, 1990.

[Har90] W. Hardle. *Applied nonparametric Regression.* Cambridge University Press, 1990.

[HBJ+91] P. H. Howard, R. S. Boethling, W. F. Jarvis, W. M. Meylan, and E. M. Michalenko. *Handbook of Environomental Degradation Rates.* Lewis Publishers, 1991.

[HFW+96] J. Han, Y. Fu, W. Wang, K. Koperski, and O. R. Zaïane. DMQL: a data mining query language for relational databases. In *Proceedings of the Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 27–34, Montreal, Canada, 1996.

[Hig62] W. H. Highleyman. The design and analysis of pattern recognition experiments. *Bell Systems Technical Journal*, 41:723–744, 1962.

[HK89] S. Y. Han and T. J. Kim. Can expert systems help with planning? *Journal of the American Planning Association*, 55:296–308, 1989.

[HKS94] J. D. Hurst, R. D. King, and M. J. E. Sternberg. Quantitative structure-activity relationships by neural networks and inductive logic programming. the inhibition of dihydrofolate reductase by pyrimidines. *Journal of Computer Aided Molecular Design*, 8:421–432, 1994.

[HT90] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman & Hall, 1990.

[IK03] A. Inokuchi and H. Kashima. Mining significant pairs of patterns from graph structures with class labels. In *Proceedings of the 3rd IEEE International Conference on Data Mining, ICDM 2003*, pages 83–90, Melbourne, Florida, USA, 2003.

[IM96] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communication of the ACM*, 39(11):58–64, 1996.

[IV99] T. Imielinski and A. Virmani. MSQL: A query language for database mining. *Data Mining and Knowledge Discovery*, 3(4):373–408, 1999.

[Iye99] V. S. Iyengar. HOT: Heuristics for oblique trees. In *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, pages 91–98, 1999.

[JJ94] M. I. Jordan and R. A. Jacobs. Hierarchical mixture of experts and the em algorithms. neural computation. *Neural Computation*, 6:181–214, 1994.

[KAA89] D. Kibler, D. Aha, and M. Albert. Instance-based prediction of real-valued attributes. *Computational Intelligence*, 5:51–57, 1989.

[Kar84] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4(4):373–381, 1984.

[Kar92] A. Karalic. Linear regression in regression tree leaves. In *Proceedings of International School for Synthesis of Expert Knowledge, ISSEK 1992*, pages 151–163, Bled, Slovenia, 1992.

[Kar95] A. Karalic. *First Order regression*. PhD thesis, Faculty of Electrical Engineering and Computer Science, Ljubljana, Slovenia, 1995.

[KB97] A. Karalic and I. Bratko. First order regression. *Machine Learning*, 26:147–176, 1997.

[KBSV99] J. Knobbe, H. Blockeel, A. Siebes, and D. M. G. Van der Wallen. Multi-relational data mining. In *Proceedings of the Benelearn 1999*, 1999.

[KC91] A. Karalic and B. Cestnik. The bayesian approach to tree-structured regression. In *Proceedings of Information Technology Interfaces, ITI 1991*, pages 155–160, Cavtat, Croatia, 1991.

[KD04] A. Karwath and L. De Raedt. Predictive graph mining. In J. N. Kok and T. Washio, editors, *Proceedings of the 2nd International Workshop on Mining Graphs, Trees and Sequences in conjunction with the 15th European Conference on Machine Learning, ECML 2004, and the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 2004*, Pisa, Italy, 2004.

[Kee98] P. Keenan. Spatial decision support systems for vehicle routing. *Decision Support Systems*, 22:65–71, 1998.

[KHS94] R.D. King, J.D. Hurst, and M.J.E. Sternberg. A comparison of artificial inteligence methods for modelling qsars. *Applied Artificial Intelligence*, 1994.

[KHS01] J. Knobbe, M. Haas, and A. Siebes. Propositionalisation and aggregates. In L. De Raedt and A. Siebes, editors, *European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 2001*, volume 2168 of *LNAI*, pages 277–288. Springer-Verlag, 2001.

[KLF01] S. Kramer, N. Lavrač, and P. Flach. *Relational Data Mining*, chapter Propositionalization Approaches to Relational Data Mining, pages 262–291. LNAI. Springer-Verlag, Berlin Heidelberg Germany, 2001.

[KM02] W. Klosgen and M. May. Spatial subgroup mining integrated in an object-relational spatial database. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 2002*, volume 2431 of *LNAI*, pages 275–286. Springer-Verlag, 2002.

[KMLS92] R.D. King, S. Muggleton, R.A. Lewis, and M.J.E. Sternberg. Drug design by machine learning: the use of inductive logic programming to model the structure-activity relationship of trimephoprim analogues binding to dihydrofolate reductase. In *Proceedings of the National Academy Science*, volume 89, pages 11322–11326, USA, 1992.

[Kop99] K. Koperski. *Progressive Refinement Approach to Spatial Data Mining.* PhD thesis, Computing Science, Simon Fraser University, British Columbia, Canada, 1999.

[Kow88] R. A. Kowalski. The early years of logic programming. *Communications of the ACM*, 31(1):38–43, 1988.

[Kra96] S. Kramer. Structural regression trees. In *Proceedings of the National Conference on Artificial Intelligence*, 1996.

[Kra99] S. Kramer. *Relational Learning vs. Propositionalization: Investigations in Inductive Logic Programming and Propositional Machine Learning.* PhD thesis, Vienna University of Technology, Vienna, Austria, 1999.

[KSV99] J. Knobbe, A. Siebes, and D. M. G. Van der Wallen. Multi-relational decision tree induction. In J. M. Zytkow and J. Rauch, editors, *European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 1999*, volume 1704 of *LNAI*, pages 378–383. Springer-Verlag, 1999.

[KVCS96] S. P. Kaluzny, S. C. Vega, T. P. Cardoso, and A. A. Shelly. *S+SPATIALSTATS users manual version 1.0.* MathSoft Inc., Seattle, 1996.

[KW01] S. Kramer and G. Widmer. *Relational Data Mining*, chapter Inducing Classification and Regression Trees in First Order Logic, pages 140–156. LNAI. Springer-Verlag, Berlin Heidelberg Germany, 2001.

[KWH01] M. Kirsten, S. Wrobel, and T. Horváth. *Relational Data Mining*, chapter Distance Based approach to Relational Learning and Clustering, pages 213–232. LNAI. Springer-Verlag, Berlin Heidelberg Germany, 2001.

[KWPd01] S. Kramer, G. Widmer, B. Pfahringer, and M. de Groeve. Prediction of ordinal classes using regression trees. *Fundamenta Informaticae*, 47:1001–1013, 2001.

[Lan96] P. Langley. *Elements of Machine Learning.* Morgan Kaufmann, 1996.

[LD94] N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, Chichester, UK, 1994.

[Lei02] H. A. Leiva. MRDTL: A multi-relational decision tree learning algorithm. Master's thesis, University of Iowa, USA, 2002.

[LM04] F. A. Lisi and D. Malerba. Inducing multi-level association rules from multiple relations. *Machine Learning*, 55:175–210, 2004.

[LML+02] A. Lanza, D. Malerba, F.A. Lisi, A. Appice, and M. Ceci. Generating logic descriptions for the automated interpretation of topographic maps. In D. Blostein and Y. B. Kwon, editors, *Graphics Recognition: Algorithms and Applications*, volume 2390 of *LNCS*, pages 200–210. Springer-Verlag, 2002.

[Loh02] W. Y. Loh. Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica*, 12:361–386, 2002.

[Lub94] D. Lubinsky. Tree structured interpretable regression. In D. Fisher and H.J. Lenz, editors, *Learning from Data*, volume 112 of *LNCS*, pages 387–398. Springer-Verlag, 1994.

[LW00] M. C. Ludl and G. Widmer. Relative unsupervised discretization for association rule mining. In D. A. Zighed, H. J. Komorowski, and J. M. Zytkow, editors, *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, PKDD 2000*, volume 1910 of *LNCS*, pages 148–158. Springer-Verlag, 2000.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical statistics and probability*, volume 1, pages 281–297. University of California Press, 1967.

[MAC03] D. Malerba, A. Appice, and M. Ceci. *Database Support for Data Mining Applications*, chapter A Data Mining Query Language for Knowledge Discovery in a Geographical Information System, pages 95–116. Number 2682 in LNCS. Springer-Verlag, Berlin Heidelberg Germany, 2003.

[MACM02] D. Malerba, A. Appice, M. Ceci, and M. Monopoli. Trading-off local versus global effects of regression nodes in model trees. In H. S. Hacid, Z.W. Ras, D.A. Zighed, and Y. Kodratoff, editors, *Foundations of Intelligent Systems, 13th International Symposium, ISMIS'2002*, volume 2366 of *LNAI*, pages 393–402. Springer-Verlag, 2002.

[Mal03] D. Malerba. Learning recursive theories in the normal ilp setting. *Fundamenta Informaticae*, 57(1):39–77, 2003.

[Man91] A. Mansfield. Comparison of perceptron training by linear programming and by the perceptron convergence procedure. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pages 25–30, 1991.

[Man97] H. Mannila. Inductive databases and condensed representations for data mining. In J. Maluszynski, editor, *Proceedings of the International Logic Programming Symposium*. MIT Press, 1997.

[MAR96] M. Mehta, R. Agrawal, and J. Rissanen. SLIQ: A fast scalable classifier for data mining. In *Proceedings of the 5th International Conference on Extending Database Technology*, pages 18–32, 1996.

[Mar99] D. Martin. *Geographical Information Systems*, volume 1, pages 71–80. John Wiley and Sons, 2nd edition edition, 1999.

[May00] M. May. Spatial knowledge discovery: The spin! system. In K. Fullerton, editor, *Proceedings of the EC-GIS Workshop*, 2000.

[Mea67] R. Mead. A mathematical model for the estimation of interplant competition. *Biometrics*, 23:189–205, 1967.

[MECA04] D. Malerba, F. Esposito, M. Ceci, and A. Appice. Top down induction of model trees with regression and splitting nodes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):612–625, 2004.

[MEL+03] D. Malerba, F. Esposito, A. Lanza, F. A. Lisi, and A. Appice. Empowering a gis with inductive learning capabilities: The case of ingens. *Journal of Computers, Environment and Urban Systems, Elsevier Science*, 27:265–281, 2003.

[MELA02] D. Malerba, F. Esposito, F. A. Lisi, and A. Appice. Mining spatial association rules in census data. *Research in Official Statistics*, 5(1):19–44, 2002.

[Mit97] T. Mitchell. *Machine Learning*. McGraw Hill, New York, USA, 1997.

[MM00] O. L. Mangasarian and D. R. Musicant. Robust linear and support vector regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):950–955, 2000.

[MMHL86] R. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The multi-purpose incremental learning system aq15 and its testing application to three medical domains. In *Proceedings of the AAAI-86*, pages 1041–1045, 1986.

[MP43] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[MP69] M. Minsky and S. Papert. *Perceptrons : an introduction to computational geometry*. MIT Press, 1969.

[MS63] J.N. Morgan and J.A. Sonquist. Problems in the analysis of survey data and a proposal. *American Statistical Association Journal*, pages 415–434, 1963.

[MS03] K. Morik and M. Scholz. The miningmart approach to knowledge discovery in databases. *Handbook of Intelligent IT*, 2003.

[MSD97] A. Moore, J. Schneider, and K. Deng. Efficient locally weighted polynomial regression predictions. In D. Fisher, editor, *Proceedings of the 14th International Conference on Machine Learning, ICML 1997*, pages 236–244. Morgan Kaufmann, 1997.

[MW63] F. Mosteller and D.L. Wallace. Inference in an authorship problem. *Journal of the American Statistical Association*, 58:275–309, 1963.

[Nad64] E.A. Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 9:141–142, 1964.

[NB86] T. Niblett and I. Bratko. Learning decision rules in noisy domains. In M.A. Bramer, editor, *Research and Development in Expert Systems*, volume 3, pages 25–34. Cambridge University Press, Cambridge, UK, 1986.

[ND97] S. H. Nienhuys-Cheng and R. De Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *LNAI*. Springer-Verlag, 1997.

[OC00] C. Ordonez and P. Cereghini. SQLEM: Fast clustering in sql using the em algorithm. In W. Chen, J. Naughton, and P. Bernstein, editors, *Proceedings of the ACM SIGMOD 2000*, volume 29, 2000.

[OD90] M. Orkin and R. Drogin. *Vital Statistics*. McGraw Hill, New York, USA, 1990.

[Par62] E. Parzen. On estimation of a probability density function and model. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[PB97] R. K. Pace and R. Barry. Performing large-scale spatial autoregressions. *Economics Letters*, 54:-291, 1997.

[PM94] G. Piatesky-Shapiro and C. Matheus. The interestingness of deviations. In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings of the AAAI Workshop of Knowledge Discovery in Databases, KDD 1994*, pages 25–36, Seattle, Washington, 1994. AAAI Press.

[Pot04a] D. Potts. Fast incremental learning of linear model trees. In C. Zhang, H. W. Guesgen, and W. K. Yeap, editors, *Proceedings of the 8th Pacific Rim International Conference of Artificial Intelligence*, volume 3157 of *LNAI*. Springer-Verlag, 2004.

[Pot04b] D. Potts. Incremental learning of linear model trees. In C. E. Brodley, editor, *Proceedings of the 21th International Conference on Machine Learning, ICML 2004*. Morgan Kaufmann Publishers, 2004.

[PS85] F. Preparata and M. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

[Que00] J. R. Quevedo Pérez. *SAFE: Sistema de aprendizaje de funciones a partir de ejemplos*. PhD thesis, Departamento de Informática, Universidad de Oviedo, Leuven, Belgium, 2000.

[Qui87] J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[Qui90] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 3(5):239–266, 1990.

[Qui92] J. R. Quinlan. Learning with continuous classes. In Adams and Sterling, editors, *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, pages 343–348. World Scientific, 1992.

[Qui93a] J. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers, 1993.

[Qui93b] J. Quinlan. Combining instance-based and model-based learning. In *Proceedings of the 10th International Conference on Machine Learning, ICML 1993*, pages 236–243, 1993.

[Qui96] J. Quinlan. Learning first-order definitions of functions. *Journal of Artificial Intelligence Research*, 5:139–161, 1996.

[Rip96] B. D. Ripley. *Pattern Recognition and Neural Network*. Cambridge University Press, 1996.

[Ris82] J. Rissanen. A universal prior for integers and estimation by the minimum description length. *Annals of Statistics*, 11:416–431, 1982.

[RK98] M. Robnik-Šikonja and I. Kononenko. Pruning regression trees with mdl. In H. Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence*, pages 455–459, Chichester, UK, 1998. J. Wiley.

[Rob97] M. Robnik-Šikonja. Core - a system that predicts continuous variables. In *Proceedings of Electrotechnical and Computer Science Conference*, pages 455–459, Portorož, Slovenia, 1997.

[Ros56] M. Rosenblatt. Remarks on some non parametric estimates of a density function. *Annals Mathematical Statistics*, 27:832–837, 1956.

[Ros58] M. Rosenblatt. The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:368–408, 1958.

[RSV02] P. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases with Application to GIS*. Morgan Kaufmann Publishers, 2002.

[RWL94] D. Rumelhart, B. Widrow, and M. Lehr. The basic ideas in neural networks. *Communications of the ACM*, 37(3):87–92, 1994.

[Sam90] H. Samet. *Applications of spatial data structures*. Addison-Wesley longman, 1990.

[Sch93] C. Schaffer. Overfitting avoidance as bias. *Machine Learning*, 10(2):153–178, 1993.

[SD01] K. Sattler and O. Dunemann. Sql database primitives for decision tree classifiers. In *Proceedings of the 10th ACM International Conference on Information and Knowledge Management, ACM CIKM 2001*, Atlanta, USA, 2001.

[She68] D. Shepard. A two-dimensional function for irregularly spaced data. In *Proceedings of ACM National Conference*, pages 517–524, 1968.

[SKM99] A. Srinivasan, R. D. King, and S. Muggleton. The role of background knowledge: using a problem from chemistry to examine the performance of an ILP program. In *Technical Report PRG-TR-08-99*. Oxford University ComputingLaboratory, 1999.

[SM94] R. Siciliano and F. Mola. Modelling for recursive partitioning in variable selection. In R. Dutter and W. Grossman, editors, *Proceedings of COMPSTAT 1994*, pages 172–177. Physica-Verlag, 1994.

[SM96] M. Stonebraker and D. Moore. *Object-Relational DBMSs: The Next Great Wave*. Morgan Kaufmann, 1996.

[SMKS94] A. Srinivasan, S. Muggleton, R. D. King, and M. J. E. Sternberg. Mutagenesis: Ilp experiments in a non-determinate biological domain. In S. Wrobel, editor, *Proceedings of the 4th Inductive Logic Programming Workshop*, pages 217–232. GMD-Studien, 1994.

[SN02] K. Saito and R. Nakano. Extracting regression rules from neural networks. *Neural Networks*, 15(10):1279–1288, 2002.

[SP91] G. Silverstein and M. J. Pazzani. Relational clichés: Constraining constructive induction during relational learning. In L. Birnbaum and G. Collins, editors, *Proceedings of the International Workshop on Machine Learning*, pages 203–207. Morgan Kaufmann, 1991.

[SSV+02] S. Shekhar, P. R. Schrater, R.R. Vatsavai, W. Wu, and S. Chawla. Spatial contextual classification and prediction models for mining geospatial data. *IEEE Transactions on Multimedia*, 4(2):174–188, 2002.

[ST95] A. Silberschatz and A. Tuzhilin. On subjective measures of interestingness in knowledge discovery. In R. Uthurusamy U. M. Fayyad, editor, *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining, KDD 1995*, pages 275–281, Montreal, Canada, 1995. AAAI Press.

[STA98] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating mining with relational database systems: Alternatives andimplications. In L. Haas and A. Tiwary, editors, *Proceedings of ACM SIGMOD 1998*, Seattle, USA, 1998.

[Sto74] M. Stone. Cross-validatory choice and assesment of statistical predictions. *Journal of the Royal Statistical Society*, 36:111–147, 1974.

[Sto85] C. J. Stone. Additive regression and other nonparametric models. *Annals of Statistics*, 13:689–705, 1985.

[SYM04] N. Subramanian, A. Yajnik, and R. S. R. Murthy. Artificial neural network as an alternative to multiple regression analysis in optimizing formulation parameters of cytarabine liposomes. *AAPS PharmSciTech*, 5(1), 2004.

[TG96] L. Torgo and J. Gama. Regression by classification. In D. Borges and C. Kaestner, editors, *Proceedings of the 13th Brazilian Symosium on Artificial Intelligence, SBIA 1996,*, volume 1159 of *LNAI*, pages 51–60. Springer-Verlag, 1996.

[TLD04] L. Todorovski, P. Ljubič, and S. Džeroski. Inducing polynomial equations for regression. In J. F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Proceedings of the 15th European Conference on Machine Learning, ECML 2004*, volume 3201 of *LNAI*, pages 441–452. Springer-Verlag, 2004.

[Tor95] L. Torgo. Data fitting with rule-based regression. In J. Zizka and P. Brazdil, editors, *Proceedings of the 2nd International Workshop on Artificial Intelligence Techniques, AIT 1995*, Brno, Czech Republic, 1995.

[Tor97] L. Torgo. Functional models for regression tree leaves. In D. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning, ICML 1997*, pages 385–393, Nashville, Tennessee, 1997.

[Tor99] L. Torgo. *Inductive Learning of Tree-based Regression Models.* PhD thesis, Department of Computer Science, Faculty of Sciences, University of Porto, Porto, Portugal, 1999.

[Tor00] L. Torgo. Partial linear trees. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning, ICML 2000*, page 1007–1014. Morgan Kaufmann Publishers, 2000.

[Tor02] L. Torgo. Computationally efficient linear regression trees. In K. Jajuga and al., editors, *Classification, Clustering and Data Analysis: recent advances and applications (Proceedings of IFCS 2002)*. Springer-Verlag, 2002.

[TP00] L. Torgo and J. Pinto da Costa. Clustered partial regression. In R. López de Mántaras and E. Plaza, editors, *Conference on Machine Learning, 11th European Conference, ECML 2000*, volume 1810 of *LNAI*, pages 426–436. Springer-Verlag, 2000.

[TR03] L. Torgo and R. Ribeiro. Predicting outliers. In N. Lavrač, D. Gamberger, H. Blockeel, and L. Todorovski, editors, *Principles and Practice of Knowledge Discovery in Databases, 7th European Conference, PKDD 2003*, volume 2838 of *LNAI*, pages 447–458. Springer-Verlag, 2003.

[UBC97] P. E. Utgoff, N. C. Berkman, and J. A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29:5–44, 1997.

[Ull88] J. Ullman. *Principles of Database and Knowledge Base Systems*, volume 1. Computer Science Press, 1988.

[Van79] C.J. Van Rijsbergen. *Information Retrieval.* Butterworths, London, second edition edition, 1979.

[VD01] W. Van Laer and L. De Raedt. *Relational Data Mining*, chapter How to Upgrade Propositional Learners to First Order logic: A Case Study, pages 235–261. LNAI. Springer-Verlag, Berlin Heidelberg Germany, 2001.

[Wah90] G. Wahba. *Spline Models for Observational Data*, volume 59. SIAM, 1990.

[Wat64] G. S. Watson. Smooth regression analysis. *Sankhya: The Indian Journal of Statistics*, 26:359–372, 1964.

[WD94] D. Wettschereck and T. Dietterich. Locally adaptive nearest neighbor algorithms. *Advances in Neural Information Processing Systems*, 6:184–191, 1994.

[Wei85] S. Weisberg. *Applied regression analysis.* Wiley, New York, USA, second edition, 1985.

[Wer75] P. Werbos. *Beyond regression – New tools for prediction and analysis in the behavioral sciences.* PhD thesis, Harvard University, Cambridge, USA, 1975.

[Wer96] P. Werbos. *The roots of backpropagation – from observed derivatives to neural networks and political forecasting.* J. Wiley & Sons, New York, USA, 1996.

[WF00] I. Witten and E. Frank. *Data mining: practical machine learning tools and techniques with Java implementations.* Morgan Kaufmann, San Francisco, 2000.

[Whi54] P. Whittle. On stationary processes in the plane. *Biometrika*, 49:434–449, 1954.

[WI93a] S. Weiss and N. Indurkhya. Optimized rule induction. *IEEE Expert*, 8(6):61–69, 1993.

[WI93b] S. Weiss and N. Indurkhya. Rule-base regression. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1072–1078, 1993.

[WI95] S. Weiss and N. Indurkhya. Rule-based machine learning methods for functional prediction. *Journal Of Artificial Intelligence Research*, 3:383–403, 1995.

[WI98] S. M. Weiss and N. Indurkhya. *Predictive Data Mining. A Practical Guide.* Morgan Kaufmann, San Francisco, Canada, 1998.

[Wol92] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[Wro01] S. Wrobel. *Relational Data Mining*, chapter Inductive logic programming for knowledge discovery in databases, pages 74–101. LNAI. Springer-Verlag, Berlin Heidelberg Germany, 2001.

[WW97] Y. Wang and I.H. Witten. Inducing model trees for continuous classes. In M. Van Someren and G. Widmer, editors, *Proceedings of the 9th European Conference on Machine Learning, ECML 1997*, pages 128–137, Prague, Czech Republic, 1997.

[WZ00] X. Wu and Y. Zhu. Neural network regression model for relative dose computation. *Physics in Medicine and Biology*, 45:913–922, 2000.

[YH02] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the 2nd IEEE International Conference on Data Mining, ICDM 2002*, pages 721–724, Maebashi City, Japan, 2002. IEEE Computer Society.