REGRESSION BY SELECTING BEST FEATURE(S)

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

by

Tolga Aydın

September, 2000

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Halil Altay Güvenir (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Özgür Ulusoy

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. İlyas Çiçekli

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Science

ABSTRACT

REGRESSION BY SELECTING BEST FEATURE(S)

Tolga Aydın
M.S. in Computer Engineering
Supervisor: Assoc. Prof. Halil Altay Güvenir
September, 2000

Two new machine learning methods, Regression by Selecting Best Feature Projections (RSBFP) and Regression by Selecting Best Features (RSBF), are presented for regression problems. These methods heavily make use of least squares regression to induce eager, parametric and context-sensitive models. Famous regression approaches of machine learning and statistics literature such as DART, MARS, RULE and kNN can not construct models that are both predictive and having reasonable training and/or querying time durations. We developed RSBFP and RSBF to fill the gap in the literature for a regression method having higher predictive accuracy and faster training and querying time durations. RSBFP constructs a decision list consisting of simple linear regression lines belonging to linear features and/or categorical feature segments. RSBF is the extended version of RSBFP such that the decision list consists of both simple, belonging to categorical feature segments, and/or multiple, belonging to linear features, linear regression lines. A relevancy heuristic has been developed to determine the features involving in the multiple regression lines. It is shown that the proposed methods are robust to irrelevant features, missing feature values and target feature noise, which make them suitable prediction tools for real-world databases. In terms of robustness, RSBFP and RSBF give better results when compared to other famous regression methods.

Keywords: Regression, function approximation, feature projections.

ÖZET

EN İYİ ÖZNİTELİKLERİ SEÇME İLE REGRESYON

Tolga Aydın
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Doç. Dr. Halil Altay Güvenir
Eylül, 2000

Regresyon problemleri için, En İyi Öznitelik İzdüşümlerini Seçerek Regresyon (RSBFP) ve En İyi Öznitelikleri Seçerek Regresyon (RSBF) adında iki yeni makine öğrenmesi metodu sunulmuştur. Bu metodlar minimum kareler regresyonunun ağırlıklı kullanımı ile çalışkan, parametrik ve adaptif modeller oluştururlar. Makine öğrenmesi ve istatistik literatürünün DART, MARS, RULE ve kNN gibi ünlü metodları hem tahmin gücü yüksek, hem de hızlı öğrenme ve/veya sorgulama yapan modeller üretememektedirler. RSBFP ve RSBF, literatürdeki bu boşluğu doldurmak için geliştirilmiştir. RSBFP, lineer özniteliklere ve/veya kategorik öznitelik parçalarına ait olan basit lineer regresyon doğrularından bir karar listesi (model) oluşturur. RSBF, RSBFP'nin gelişmiş versiyonu olup, karar listesi hem lineer özniteliklere ait çoklu hem de kategorik öznitelik parçalarına ait basit lineer regresyon doğrularından oluşur. Çoklu regresyon doğrularında yer alan öznitelikler geliştirilen bir uygunluk sezgisi ile bulunur. RSBFP ve RSBF'in, gereksiz özniteliklere, bilinmeyen değerlere ve gürültülü hedef öznitelik değerlerine karşı dayanıklı bir performansa sahip olduğu gösterilmiştir. Böyle durumlarda diğer metodlara nazaran daha iyi sonuçlar vermeleri, gerçek veri kümeleri için uygun birer tahmin aracı olduklarını gösterir.

Anahtar Sözcükler: Regresyon, fonksiyon yaklaştırımı, öznitelik izdüşümü.

Anneme ve Babama.

ACKNOWLEDGMENTS

I would like to express my gratitude to Dr. H. Altay Güvenir, from whom I have learned a lot, due to his supervision, suggestions, and support during this research.

I am also indebted to Dr. Özgür Ulusoy and Dr. İlyas Çiçekli for showing keen interest to the subject matter and accepting to read and review this thesis.

I would like to thank to İlhan Uysal for his morale support and for many things.

This thesis was supported, in part, by TUBITAK (Scientific and Technical Research Council of Turkey) under Grant 198E015.

Contents

1 Introduction 1

1.1 Parametric versus Non-Parametric Learning 2

1.2 Eager versus Lazy Learning 3

1.3 Regression by Selecting Best Feature(s) 4

1.4 Outline of the Thesis 6

2 Overview of Regression Techniques 7

2.1 k Nearest Neighbor Regression 7

2.2 Locally Weighted Regression 9

2.2.1 Nonlinear Local Models 9

2.2.2 Linear Local Models 10

2.3 Regression by Decision Rule Induction 11

2.4 Regression by Decision Tree Induction 14

2.4.1 CART 16

2.4.2 RETIS 18

2.4.3 M5 20

2.5 Multivariate Adaptive Regression Splines 21

3 Regression by Selecting Best Feature Projections 25

3.1 The RSBFP Algorithm 25

3.1.1 Training 26

3.1.2 Querying 29

3.1.3 Target Noise Elimination 31

3.2 Properties of RSBFP 33

3.2.1 Eager Learning 33

3.2.2 Context-sensitive (Adaptive) Learning 34

3.2.3 Different Feature Types 34

3.2.4 Curse of Dimensionality 35

3.2.5 Normalization of Features 35

3.2.6 Irrelevant Features 36

3.2.7 Redundant Features 36

3.2.8 Missing Feature Values 37

3.2.9 Noise 37

3.2.10 Bias-variance Trade-off 38

3.2.11 Model Complexity and Occam's Razor 38

3.2.12 Interpretation 39

3.2.13 Interactions and Lack of Many Additive Terms 39

3.3 Complexity Analysis 40

4 Regression by Selecting Best Features **42**

4.1 The RSBF Algorithm 42

4.1.1 Training 43

4.1.2 Querying 47

4.1.3 Target Noise Elimination 48

4.2 Properties of RSBF 49

4.2.1 Eager Learning 50

4.2.2 Context-sensitive (Adaptive) Learning 51

4.2.3 Different Feature Types 51

4.2.4 Curse of Dimensionality 51

4.2.5 Normalization of Features 51

4.2.6 Irrelevant Features 52

4.2.7 Missing Feature Values 52

4.2.8 Noise 52

4.2.9 Bias-variance Trade-off 53

4.2.10 Model Complexity and Occam's Razor 53

4.2.11 Interpretation 53

4.2.12 Interactions and Lack of Combinations 53

4.3 Complexity Analysis 54

5 Empirical Evaluations **56**

5.1 Performance Measure 57

5.2 Other Regression Methods Used in Comparisons 58

5.2.1 RULE 58

5.2.2 KNN 58

5.2.3 DART 59

5.2.4 MARS 59

5.3 Real Data Sets 59

5.4 Accuracy 59

5.5 Robustness to Irrelevant Features 61

5.6 Robustness to Missing Feature Values 63

5.7 Robustness to Target Noise 63

5.8 Computation Times 68

6 Conclusion and Future Work **73**

List of Figures

2.1 The k Nearest Neighbor Regression 8

2.2 Swap-1 Algorithm 11

2.3 A sample output 12

2.4 Constructing Pseudo-Classes (P-Class) 14

2.5 Overview of Method for Decision Rule Induction in Regression Problems 15

2.6 An Example Regression Tree 17

2.7 Construction process of the example regression tree. Predictor features x_1 and x_2 construct three leaf nodes. 17

2.8 An example region consisting of only one predictor , with large variance, which is not suitable for splitting 19

2.9 MARS Algorithm 24

2.10 An example for the regions of MARS algorithm 24

3.1 An example training set projected to four features: f_1, f_2, f_3 and f_4. 26

3.2 The decision list of simple linear regression lines (the model learned). 29

3.3 Model Construction in RSBFP 30

3.4 An example for querying phase 30

3.5 Querying Phase in RSBFP 31

3.6 Determination of the optimal value of k for the RSBFP method. 33

4.1 An example training set projected to five features: f_1, f_2, f_3, f_4 and f_5. 43

4.2 Normal Equations Method 45

4.3 Cholesky Factorization 45

4.4 The ordered list of simple linear regression lines of linear features. 46

4.5 The induced model 47

4.6 Model Construction in RSBF 48

4.7 Querying Phase in RSBF 49

4.8 Determination of the optimal value of k for RSBF method . . . 50

5.1 Relative errors of methods with increasing irrelevant features . . 64

5.2 Relative errors of methods with increasing missing feature values 66

5.3 Relative errors of methods with increasing target noise 69

List of Tables

2.1 Example of swapping and adding components. 13

5.1 Properties of the data sets used in the experiments. L: Linear, C: Categorical. 60

5.2 Relative errors of regression methods. Best results are typed with bold font. 62

5.3 Relative errors of regression methods, where 30 irrelevant features are added to real data sets. If the result is not available due to singular variance/covariance matrix, it is shown with (*). Best results are typed with bold font. 65

5.4 Relative errors of regression methods, where 20% of values of real data sets are removed. If the result is not available due to singular variance/covariance matrix, it is shown with (*). Best results are typed with bold font. 67

5.5 Relative errors of regression methods, where 20% target noise is added to real data sets. Best results are typed with bold font. . 70

5.6 Training time durations of methods in milliseconds. Best results are typed with bold font. 71

5.7 Querying time durations of methods in milliseconds. Best results are typed with bold font. 72

List of Symbols and Abbreviations

B	: Basis Function
β	: Parameter set
CART	: Classification and Regression Trees
d	: Distance function
D	: Training set
DART	: Regression Tree Induction Algorithm
DMSK	: Data Miner Software Kit
DNF	: Disjunctive Normal Form
f	: Approximated function
I	: Impurity measure
i	: Instance
IBL	: Instance-Based Learning
K	: Kernel Function
k	: Number of neighbor instances
KMEANS	: Partitioning clustering algorithm
KNN	: K Nearest Neighbor
KDD	: Knowledge Discovery in Databases
L	: Loss function
log	: Logarithm in base 2
m	: Number of predictor features
MAD	: Mean Absolute Distance
MARS	: Multivariate Adaptive Regression Splines
M5	: Regression tree induction algorithm
n	: Number of training instances
p	: Number of parameters or features
$\mathbf{x_q}$	: Query instance
R	: Region
R_k	: Rule set
RE	: Relative Error
RETIS	: Regression tree induction algorithm
RSBF	: Regression by Selecting Best Features
RSBFP	: Regression by Selecting Best Feature Projections

RULE	: Rule-based regression algorithm
r	: Rule
t	: A test example
T	: Number of test instances
$\mathbf{X}$	: Instance matrix
$\mathbf{x}$	: Instance vector
$\mathbf{x}_i$	: Value vector of i^{th} instance
$\mathbf{y}$	: Target vector
$\bar{y}$	: Estimated target

Chapter 1

Introduction

Prediction is the most common problem researched in machine learning and data mining. Predicting values of categorical or nominal features is called *classification*, whereas predicting values of numeric or linear features is called *regression* in the literature. In machine learning, much research has been performed on classification. But recently, researchers began to deal with regression since many real-world problems can be modeled as regression problems. Dynamic control problems can be considered as real-world problems. For instance, learning to catch a ball moving in a three-dimensional space, is a dynamic control problem that is mainly researched by robotics community. Aha and Salzberg proposed several variants of k-nearest, an instance-based algorithm, algorithms to increase the ability of the robot in catching the ball moving in three dimensions [2]. Furthermore, learning the city-cycle fuel consumption of cars in miles per gallon is another interesting real-world problem that can be modeled as a regression problem [20]. Fuel consumption is determined by many factors including the horsepower, weight, acceleration and the number of cylinders. A regression algorithm can be employed to model the relationship between the fuel consumption and the factors mentioned above. Although *regression* term is mainly used, there are other names given to it, such as *functional prediction*, *function approximation* and *continuous class learning*.

Databases can store large amounts of data belonging to many different domains. And since database management systems enable only deductive querying, different experts are required for each different domain to discover knowledge in databases. In some cases, a domain expert may not be available or the knowledge of the experts may be implicit [1, 29]. Therefore, the use of automatic methods such as induction becomes helpful for knowledge discovery.

Many induction techniques have been developed in machine learning to discover knowledge from databases. The idea of using induction techniques is widely accepted by Knowledge Discovery in Databases (KDD) discipline, which incorporates researchers from various areas. Knowledge engineers and database experts cooperate to create a database management system that not only enables deductive querying, but also provides an inductive component for automatic knowledge discovery.

The term "knowledge" means two types of information. One is the information used for prediction of a new case, given training cases; the other is the information used for extracting new rules about the domain by interpreting the induced models. The induced models reviewed and developed in this thesis can be employed in cases, when the underlying problem is formalized as a prediction of a linear target feature.

1.1 Parametric versus Non-Parametric Learning

Parametric learning methods try to fit the data to a global parametric function. On the other hand, non-parametric learning methods make no assumption about the structure of the function. Parametric learning methods perform well when the assumed structure of the function is close to the function that generated the data. However, non-parametric learning methods can be preferred if finding a general structure rich enough to model a large portion of all possible functions is of concern, or no information about the structure of the function is available.

The classical *linear regression* model is a well-known example of parametric learning. This model consists of a dependent (target) feature y and independent (predictor) features (x's). The relationship between y and each x is assumed to be linear. That is, target feature's value changes at a constant rate as the value of any predictor feature changes.

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i \tag{1.1}$$

The subscript i denotes the observations, the second subscript p denotes the index of independent features. There are $p+1$ parameters, $\beta_j, j = 0, \ldots, p$, to be estimated. In the parametric model, the structure of the function is given, and the procedure estimates the values of the parameters, β_j, according to a fitting criterion. This criterion is generally a minimization of an error function for all data points in a training set. Very often this is the *least squares* criterion, which minimizes the sum of the squares of the prediction errors of the estimated linear function for all instances. The error term, ε_i , denotes the error of estimation for each instance i, and it is assumed to be normally distributed.

1.2 Eager versus Lazy Learning

Learning methods can also be grouped as *eager* versus *lazy* methods. Lazy methods do not construct models since the model is the training data itself. On the other hand, eager methods construct rigorous models. These two type of methods can be compared in many aspects.

The major task of both methods is prediction. Although they both perform well in this major task, only eager methods address the induction of concept description that enables interpretation of the data. Eager methods induce models such as decision trees and decision rules that enable us to interpret the underlying data. Furthermore, lazy methods store the whole data in memory to process them in the prediction phase rather than the training phase. This may cause some storage problems when the size of the data is too big to fit into

the memory. Eager methods are very fast in the prediction phase. On the other hand lazy methods suffer at this point since all the computations are performed at this phase. Finally, although most eager methods are *adaptive (context-sensitive)*, most of lazy methods do not hold this property. An adaptive method can determine the relevant or important regions of the instance space. That is, it does not simply label predictor features as relevant or irrelevant; instead, it determines the relevant regions of each predictor feature.

Although eager methods are preferable in terms of less storage requirements, fast prediction phase and interpretation of the underlying data, they are outperformed by lazy methods in terms of some criteria. Lazy methods do not generalize the data by constructing global models. Therefore, their training phase is very simple and fast since it involves only storing the training data. In the prediction phase, they make predictions according to the local position of the query instances. This brings out the fact that lazy methods can form complex decision boundaries around the query instance even in the existence of little information.

A powerful regression method is the one that has small training and prediction time requirements, while preserving a comparable predictive accuracy. In this thesis, we have developed two eager regression methods holding these desired properties. As being an eager method, the proposed methods are fast in the prediction phase. In the training phase, they are much more faster than other popular eager approaches. This is achieved by the simplicity of our approach in constructing the models. Our proposed methods are not only fast and predictive, but also interpretable. So they will especially be preferable in modeling large databases.

1.3 Regression by Selecting Best Feature(s)

This thesis describes two new machine learning methods based on selecting best feature(s). They are *Regression by Selecting Best Feature Projections* (RSBFP) and *Regression by Selecting Best Features* (RSBF). Training in RSBFP aims to find the predictive power of each predictor feature by constructing simple

linear regression lines, one per each linear predictor feature and number of categories per each categorical predictor feature. Although the predictive power of a linear predictor feature is constant, it varies for each distinct value of categorical predictor features. At the end of the training part, these simple linear regression lines are sorted according to their predictive power to induce the final model. Training in RSBF consists of two phases: The first phase is similar to the training part of RSBFP. We construct simple linear regression lines, one per each linear predictor feature and number of categories per each categorical predictor feature. The second phase constructs multiple linear regression lines among linear predictor features, each time excluding the worst predictor feature among the current set. Finally, these multiple linear regression lines and categorical predictor features' simple linear regression lines are sorted according to their predictive power to induce the final model. These two phases are together called the training part of the RSBF method. In the prediction part of learning, the best (simple or multiple) linear regression line in the case of RSBF, and the best simple linear regression line in the case of RSBFP are selected to make predictions for the query instances.

Both RSBFP and RSBF are robust to irrelevant features. They select the regression line consisting of best feature(s) to predict the target feature value of a query instance. They are also flexible methods since the best feature(s) may differ for each query instance. They handle missing feature values naturally, without filling them with estimated values. The experimental results show that they achieve the highest accuracy values when there are many missing values, irrelevant features and target noise.

RSBFP and RSBF are eager regression methods since they construct a global model in the training phase. The form of the global model is parametric, the model is actually a decision list consisting of parametric, explicitly linear, regression lines. The high predictive power of the proposed methods is important since parametric, especially linear, models are not expected to fit into the real-world databases. Important properties of RSBFP and RSBF, also a detailed comparison of them with other famous lazy and eager methods are described in the following chapters.

1.4 Outline of the Thesis

In the next chapter, we make an overview of existing important regression methods in the literature. In Chapter 3 and Chapter 4, we describe RSBFP and RSBF, respectively. The detailed description of characteristic properties of our methods are given in these chapters. Empirical evaluations of RSBFP and RSBF are shown in Chapter 5, and we conclude the thesis with Chapter 6.

Chapter 2

Overview of Regression Techniques

In this chapter, some regression techniques developed in machine learning and statistics community are reviewed. In the first two sections, we review two lazy approaches for regression: k nearest-neighbor regression, and locally weighted regression. In the subsequent sections, from Section 2.3 to Section 2.5, three eager approaches for regression are reviewed. These are namely rule-based regression, tree-based regression and multivariate adaptive regression splines.

2.1 k Nearest Neighbor Regression

k Nearest neighbor regression is an instance-based learning (IBL) algorithm. IBL algorithms are well known due to their computationally simple training (learning) phases [3, 11]. In the k nearest neighbor regression, as in many other IBL algorithms, training is performed by simply storing the instances in the memory.

Each training instance is represented as a set of feature-value pairs. Predictor feature values may be of categorical (nominal) or linear (ordered) type, whereas target feature values are of only linear type. In the training phase, each training instance is stored in memory. The querying phase of the k nearest

neighbor regression tries to predict the target feature value of a query instance as a function of most similar instances' target feature values. The k value is selected as the number of nearest (most similar) neighbors that will be taken into account in the querying phase.

Training:
[1] $\forall \mathbf{x}_t \in$ Training Set
[2] Store $\mathbf{x}_t$ in memory

Querying:
[1] $\forall \mathbf{x}_q \in$ Query Set
[2] $\forall \mathbf{x}_t \{\mathbf{x}_t \neq \mathbf{x}_q\}$: Calculate $Similarity(\mathbf{x}_q, \mathbf{x}_t)$
[3] Let $Similars$ be set of k most similar instances to $\mathbf{x}_q$ in Training Set
[4] Let $Sum = \sum_{\mathbf{x}_t \in Similars} Similarity(\mathbf{x}_q, \mathbf{x}_t)$
[5] Then $\bar{y}_q = \sum_{\mathbf{x}_t \in Similars} \frac{Similarity(\mathbf{x}_q, \mathbf{x}_t)}{Sum} \mathbf{y}_t$

Figure 2.1. The k Nearest Neighbor Regression

There is a variety of k nearest neighbor regression approaches in the literature. The algorithm, shown in Figure 2.1, is the simplest k nearest neighbor regression approach. For a given query instance, k nearest (similar) training instances are determined by using the *Similarity* function. The similarity between the query instance x_q and a training instance x_t is determined as,

$$Similarity(\mathbf{x}_q, \mathbf{x}_t) = \sqrt{\sum_{i=1}^{p} Sim(x_{qi}, x_{ti})} \tag{2.1}$$

where $Sim(x_{qi}, x_{ti}) = (\frac{x_{qi} - x_{ti}}{range\ (i)})^2$ where i is the feature dimension.

Finally, the weighted sum of the target values of the k nearest neighbors of x_q is used as the predicted target value, $\bar{y}_q$, of the query instance x_q.

The k nearest neighbor algorithm assumes that all the predictor features are equally relevant. However, the prediction accuracy of the model can be improved if the predictor features are assigned proper weights to denote their relevancy in the prediction process [39]. These weight values can be either obtained from database experts or automatically determined by some feature weight learning algorithms [10, 26]. In terms of interpretability, the k nearest

neighbor algorithm is very poor, since it is a lazy approach. It does not induce models that enable interpretation of the underlying data set [28]

2.2 Locally Weighted Regression

Locally weighted regression is very similar to k nearest neighbor regression. It is also an instance based algorithm. In the training phase, it just stores the training instances in the memory. The main work is done in the querying phase, where it makes use of k nearest neighbors of the query point, and gives importance to the nearby instances proportional to their similarity to the query instance. Although k nearest neighbor regression approach takes the weighted average of the target values of the nearby training instances, locally weighted regression approach constructs a local linear or non-linear model by using these nearest neighbors. As in k nearest neighbor regression approach, nearby instances have more weight on the construction of the local parametric model, whereas distant instances have less weight on the model construction process. The local models are each specific to the query point. That is, a different model is constructed for each different query instance. A detailed information about the structure of the linear or non-linear models constructed for query instances can be found in [6].

2.2.1 Nonlinear Local Models

A non-linear local model can be constructed by modifying the non-linear global model. A general global model can be trained to minimize the following unweighted training criterion:

$$C = \sum_i L(f(\mathbf{x}_i, \boldsymbol{\beta}), y_i) \tag{2.2}$$

where the y_i is the output value corresponding to the input vector $\mathbf{x}_i$, $\boldsymbol{\beta}$ is the parameter vector for the nonlinear model $\bar{y}_i = f(\mathbf{x}_i, \beta)$, and $L(\bar{y}_i, y_i)$ is a general loss function for predicting $\bar{y}_i$ when the training data is y_i. Often the

least squares criterion is used for the loss function $L(\bar{y}_i, y_i) = (\bar{y}_i - y_i)^2$, leading to the training criterion:

$$C = \sum_i (f(\mathbf{x}_i, \boldsymbol{\beta}) - y_i)^2 \tag{2.3}$$

The parameters of a global model may not provide a good approximation of the true function. In this case, there are two approaches to solve the problem. First, we can use a larger, more complex global model and hope that it can approximate the data sufficiently. The second approach, which is the main concern of this subsection, is to fit the model to local patches instead of the whole region.

The training data set can be tailored to the query point by emphasizing nearby points in the regression. This can be accomplished by weighting the training criterion:

$$C(\mathbf{q}) = \sum_i [L(f(\mathbf{x}_i, \boldsymbol{\beta}), y_i)) K(d(\mathbf{x}_i, \mathbf{q}))] \tag{2.4}$$

where K is the weighting or *kernel* function and $d(\mathbf{x}_i, \mathbf{q})$ is the distance between the data point $\mathbf{x}_i$ and the query $\mathbf{q}$. Using this training criterion, $f(\mathbf{x}, \beta(\mathbf{q}))$ becomes a local model, and can have a different set of parameters $\beta(\mathbf{q})$ for each query point $\mathbf{q}$.

2.2.2 Linear Local Models

Given that we are using local models, it seems advantageous to keep them simple, and to keep the training criterion simple as well. This leads us to explore local models that are linear in the unknown parameters, and to use the least squares criterion:

$$C(\mathbf{q}) = \sum_i [(\mathbf{x}_i \boldsymbol{\beta}) - y_i)^2 K(d(\mathbf{x}_i, \mathbf{q}))]. \tag{2.5}$$

There are many variations of distance (d) and weighting (K) functions for

local models [6]. These functions lead us to construct many types of linear and non-linear locally weighted regression methods.

2.3 Regression by Decision Rule Induction

Learning decision rules in Disjunctive Normal Form (DNF) from a given training set is also popular in machine learning. Weiss and Indurkhya developed a rule-based classification algorithm [40], called Swap1, and then adapted it for regression [41, 42].

[1] Input: D, a set of training cases
[2] Initialize $R_1 \leftarrow$ empty set, $k \leftarrow 1$, and $C_1 \leftarrow D$

[3] repeat
[4] create a rule B with a randomly chosen feature as its left-hand side
[5] while (B is not 100-percent predictive) do
[6] make single best swap for any component of B, including deletion of the component, using cases in C_k
[7] If no swap is found, add the single best component to B
[8] endwhile
[9] $P_k \leftarrow$ rule B that is now 100-percent predictive
[10] $E_k \leftarrow$ cases in C that satisfy the single-best-rule P_k
[11] $R_{k+1} \leftarrow R_k \cup \{P_k\}$
[12] $C_{k+1} \leftarrow C_k - \{E_k\}$
[13] $k \leftarrow k + 1$
[14] until (C_k is empty)
[15] find rule r in R_k that can be deleted without affecting performance on cases in training set D
[16] while (r can be found)
[17] $R_{k+1} \leftarrow R_k - \{r\}$
[18] $k \leftarrow k + 1$
[19] endwhile
[20] output R_k and halt.

Figure 2.2. Swap-1 Algorithm

Learning decision tree induction models is similar to learning decision rule induction models in the sense that they can be converted into DNF models. In decision tree induction models, there is exactly one path from the root to

a leaf that is satisfied for a query instance. Therefore, if each of these paths is regarded as a rule, the fact that these rules are mutually exclusive is easily observed. This leads to the restriction that decision tree induction models are not compact. On the other hand, decision rule induction models, like Swap1, are compact since the rules are not mutually exclusive. This may cause a query instance to satisfy more than one rule. For example, Swap1 algorithm may assign more than one class for a query instance. This problem can be resolved by assigning ordering to the rules according to their extraction order. The first rule, according to this ordering, that satisfies the query instance determines the class of the query instance. The Swap-1 decision rule induction model [40] and a sample output are shown in Figure 2.2 and Figure 2.3, respectively.

$$
\begin{array}{lll}
X > 0.2 \text{ And } Y > 2.5 & \leftarrow & Class = 1 \\
Z > 4.5 & \leftarrow & Class = 1 \\
[True] & \leftarrow & Class = 2
\end{array}
$$

Figure 2.3. A sample output

In constructing a new rule, Swap-1 constantly searches all the conjunctive components it has already formed, and tries all the swapping combinations between the components already held in the rule and the components lying outside the rule. If some of the swapping combinations improve the rule, then the best swap, the swap leading to best predictive value of the rule, is selected among them. It may be the case that none of the swapping combinations improves the rule. In that case, the best component lying outside is inserted into the rule. The best component is the one that increases the predictive value of the rule most. In the existence of more than one best components to be added, maximum instance coverage is used as the second criterion. The swappings and additions end when the rule reaches 100% predictive value. Any conjunctive component that is swapped out need not necessarily stay out, it can swap in during the next swappings and additions provided that its addition increases the predictive value of the rule.

Table 2.1 shows a sample decision rule induction. After the rule reaches 100% predictive value, the instances covered by that rule are removed from the instance space. The remaining instances are now ready for the next decision rule induction steps.

STEP	PREDICTIVE VALUE (%)	RULE
1	31	p3
2	36	p6
3	48	p6 & p1
4	49	p4 & p1
5	69	p4 & p1 & p2
6	80	p4 & p1 & p2 & p5
7	100	p3 & p1 & p2 & p5

Table 2.1. Example of swapping and adding components.

Upon the construction of all rules, a pruning and an optimization procedure can be employed [40]. Pruning aims to decrease the number of rules. If the removal of any rule does not affect the predictive value on training set, then that rule need not be stored anymore.

Many decision rule induction models, such as Swap-1, are intended to predict categorical target features. By employing a preprocessing step, the linear target features can be transformed to categorical target features. This preprocessing step enables us to use decision rule induction models in regression problems.

Figure 2.4 shows the P-class algorithm [42] that transforms the linear target features to categorical target features. The underlying idea of P-class algorithm is to make y values within one class most similar and y values across classes most dissimilar. Assignment of y values is performed in such a way that the distance between each y_i and its class mean should be minimized.

P-Class algorithm is in fact a variation of the famous $KMEANS$ clustering algorithm [14, 24]. It is very simple when compared to the $KMEANS$, since it produces only one-dimensional clustering of training data. On the other hand, $KMEANS$ can produce multi-dimensional clustering of training data by swapping the instances between the clusters according to a clustering criterion. The drawback of both $KMEANS$ and P-Class algorithms is their unability to determine the number of clusters and classes, respectively.

The use of P-Class algorithm in the preprocessing step lets us to use Swap-1

[1]Input: $\{y\}$ a set of output values
[2] Initialize n = number of cases, k = number of classes

[3] repeat for each $Class_i$
[4] $Class_i$ = next n/k cases from list of sorted y values
[5] end

[6] repeat for each $Class_i$ (until no change for any class)
[7] repeat for each case j in $Class_i$
[8] 1. Move $Case_{ij}$ to $Class_{i-1}$, compute Err_{new}
[9] If $Err_{new} > Err_{old}$ return $Case_{ij}$ to C_i
[10] 2. Move $Case_{ij}$ to $Class_{i+1}$, compute Err_{new}
[11] If $Err_{new} > Err_{old}$ return $Case_{ij}$ to C_i
[12] next $Case_j$ in $Class_i$
[13] Next $Class_i$

[14] repeat for each $Class_i$ (until no change for any class)
[15] If Mean($Class_i$) = Mean($Class_j$) then
[16] Combine $Class_i$ and $Class_j$
[17] end

Figure 2.4. Constructing Pseudo-Classes (P-Class)

in regression problems. An overview of the whole procedure for decision rule induction in regression problems is shown in Figure 2.5.

There are different approaches to predict the target feature of the query instance. Mean or median value of the class can be assigned for the prediction of the target feature of the query instance. However, some parametric (such as linear least squares regression) or non-parametric (such as k nearest neighbor regression) models can also be employed. Weiss obtained significant improvements by employing nearest neighbor regression instead of simply using median or mean of the class [42].

2.4 Regression by Decision Tree Induction

Decision tree induction models determine the nodes in the tree and the tests associated with nonterminal nodes. They rely on recursive partitioning of

[1] Generate a set of Pseudo-classes using the P-Class algorithm.
[2] Generate a covering rule-set for the transformed classification problem using a rule induction method such as Swap-1.
[3] Initialize the current rule set to be the covering rule set and save it.
[4] If the current rule set can be pruned, iteratively do the following:
a) Prune the current rule set.
b) Optimize the pruned rule set and save it.
c) Make this pruned rule set the new current rule set.
[5] Use test instances or cross-validation to pick the best of the rule sets.

Figure 2.5. Overview of Method for Decision Rule Induction in Regression Problems

the data by picking the single best feature to separate the data and repeat the process on the subdivisions of the data. The terminal nodes are assigned the majority class of the training instances found in the terminal node for classification. In the case of regression, terminal nodes are generally assigned the mean or median value of the training instances found in the terminal node. Decision tree induction models may also employ some pruning strategies to avoid overfitting and to obtain simpler decision trees.

Decision tree induction models are well suited for high dimensional applications, since they employ dynamic feature selection. They have the ability to exploit low local dimensionality of functions. In local regions of the training data set, a few predictor features may have the highest influence on the predicted target feature. The decision tree induction models enable us to handle such cases. Another advantage of these models is the fact that they allow interpretation of the underlying training data set. In terms of accuracy of the predictions and time complexity, they are also comparable to many other models.

On the other hand, these models have some drawbacks. For instance, there is no continuity between the terminal nodes (regions). As a consequence of this situation, they can not approximate even simple continuous functions such as linear functions. By just analyzing the structure of the decision tree, it is not possible to understand the structure of the function (e.g. linear or additive).

The following subsections describe three different decision tree induction models: CART, RETIS and M5. Although they all have the characteristics mentioned above, they differ in some of properties and measures such as the measure used to select the single best feature and its partitioning value.

2.4.1 CART

CART is the first decision tree induction model developed by the statistical research community [9]. It is suitable for both classification and regression. CART induces decision trees in the following manner. It begins with the whole training data set and stores them in the root of the tree. Then it searches for the best feature and feature value of any instance to split the root node. Splitting the root node yields two nodes, and the original training data is now divided among them. These nodes represent two disjoint regions in the training data set, and one of these regions is selected for further splitting. Again the best feature and feature value of any instance lying in the splitted region is searched for splitting process. At any step of the decision tree induction, one of the disjoint regions that has not yet been splitted is chosen for further splitting. The decision tree induction process ends when a predefined number of disjoint regions is reached.

The selection of the region to be splitted among non-splitted disjoint regions, the feature and the splitting feature value play a key role at each step of the induction process. CART uses an error criterion to produce optimum disjoint leaf nodes. The optimum value of this error criterion, Equation 2.8, is used at every step of the induction process to select the best disjoint region, feature and splitting feature value.

At each leaf node, the variance of the output values of the training instances lying in that node is used as the impurity measure.

$$Variance = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{2.6}$$

where n is the number of instances in the disjoint region.

$$Splitting\ Error = \frac{1}{n}\left\{ \sum_{\mathbf{x}_i \in \mathbf{X}_{left}} (y_i - \bar{y}_{left})^2 + \sum_{\mathbf{x}_j \in \mathbf{X}_{right}} (y_j - \bar{y}_{right})^2 \right\} \quad (2.7)$$

Then at each leaf node, the splitting error is computed for each possible feature and splitting feature value pair. The disjoint region, the feature and the splitting feature value that maximizes the C in Equation 2.8 are used for the current splitting step of the decision tree induction process.

$$C = Variance - Splitting\ Error \quad (2.8)$$

An example regression tree and its construction process are shown in Figure 2.6 and Figure 2.7, respectively.

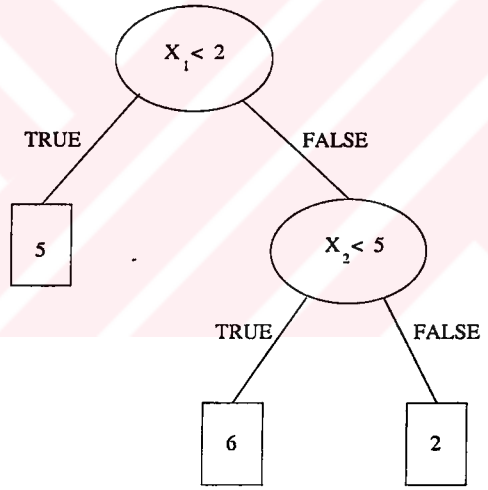

Figure 2.6. An Example Regression Tree

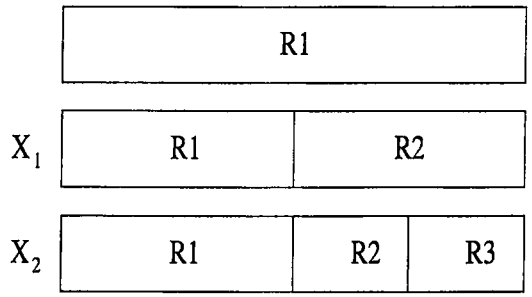

Figure 2.7. Construction process of the example regression tree. Predictor features x_1 and x_2 construct three leaf nodes.

When the decision tree induction process is completed, each leaf (non-splitted) node determines a constant value to be used in predicting the target feature value of a query instance. In the case of classification, each leaf node is assigned the majority class of the training instances found in that leaf node. In the case of regression, each leaf node is assigned the mean or median value of the training instances found in that leaf node. Any query instance follows a unique path from the root node to the leaf node covering its location. The value found in the covering leaf node is used as the predicted target feature value of that query instance.

Decision tree induction models like CART may produce a tree consisting of many disjoint regions. If the regions are too small, then it is very probable that the classification or the regression tree will overfit the training data. To overcome this problem, a pruning strategy can be employed.

One strategy may be to remove some of the leaf nodes of the tree. But at this time, when a query instance follows the path from the root node to one of these removed nodes, the tree will not determine a prediction value for that query instance. This problem can be avoided by removing the sibling leaf nodes together, and by merging them to a single disjoint region [9].

2.4.2 RETIS

RETIS (Regression Tree Induction System) decision tree induction model is different from CART in that it uses a different error criterion to maximize and it is suitable only for regression problems. Therefore, RETIS [22, 23] is actually a regression tree induction model. In CART, the leaf (terminal) nodes use $\bar{y}$ of the instances lying in their disjoint region as a prediction value of the query instances. In the case of RETIS, a multiple linear regression line is constructed at every leaf node. The use of linear regression at the leaf nodes of the regression tree is called *local linear regression* [22].

Since RETIS employs local linear regression at the leaf nodes, this technique also affects the measure of the error criterion used in the model. In CART, the Equation 2.8 is used as the error criterion. The disjoint region, the feature and

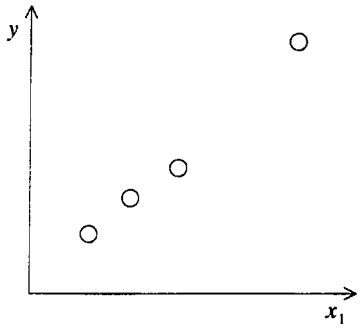

Figure 2.8. An example region consisting of only one predictor , with large variance, which is not suitable for splitting

the splitting feature value that maximizes the expected variance reduction are selected for further splitting. However, expected variance reduction may not be suitable for RETIS. In a region, if the relationship between the predictor features and the target feature is linear, it will not be suitable to further split that region even in the existence of high variance as shown in Figure 2.8. In such a case, the use of C in Equation 2.8 as an error criterion may cause this region to be selected for further splitting. Therefore, RETIS uses the error criterion C given in Equation 2.9.

$$C = I(\mathbf{X}) - \textit{Splitting Error} \tag{2.9}$$

I (*impurity measure*) is defined as the following:

$$I(\mathbf{X}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 \tag{2.10}$$

where n is the number of instances, f is the linear function that best fits the instances of the region. Furthermore, the splitting error is defined as in Equation 2.11.

$$\textit{Splitting Error} = \frac{1}{n}[n_{left}I_{left} + n_{right}I_{right}] \tag{2.11}$$

The disjoint region, the feature and the splitting feature value whose split gives the maximum C are selected for the current step of splitting process. In the querying phase of learning, the multiple linear regression line of the leaf

node is used to predict the target feature value of the query instance falling in that leaf node.

A pruning strategy may be employed after the construction of the regression tree to overcome overfitting the training data. RETIS computes two error measures: *static error* and the *backed-up error* at each node. The static error of a node is the error that will be faced if the node was converted to a leaf node. On the other hand, backed-up error of a node is the error that will be faced if the node's subtree was not pruned. The subtree is pruned at that node if the static error is less than or equal to the backed-up error.

2.4.3 M5

M5 decision tree induction model is actually a regression tree induction model, since it is suitable only for regression problems. It is similar to both CART and RETIS. The error criterion used in M5 [33] is the expected standard deviation reduction (Equation 2.12). The disjoint region, the feature and the splitting feature value that maximize the expected standard deviation reduction are chosen for the current step of the splitting process. This error criterion is similar to the one used in CART (the expected variance reduction).

$$\textit{Expected StdDev Reduction} = \sigma(\mathbf{X}) - \sum_i \frac{|\mathbf{X}_i|}{|\mathbf{X}|}\sigma(\mathbf{X}_i). \qquad (2.12)$$

where σ is standard deviation and i is the number of subregions of a region whose instances are denoted by $\mathbf{X}$.

M5 is also similar to RETIS, in that it employs linear regression models on the nodes [30]. Although RETIS employs linear regression models just after each split process, M5 employs those models after the regression tree was constructed. Another important difference between M5 and RETIS is the fact that any node of M5 restricts itself to the predictor features that are referenced by tests or linear regression models somewhere in the subtree at this node. Therefore, it can not use all the predictor features in constructing its own linear regression model.

The linear regression models may underestimate the error on query instances. This usually happens if the linear regression model involves many parameters and was constructed from small number of cases. Therefore, the error on any query instance is multiplied by $(n + v)/(n - v)$, where n is the number of instances and v is the number of parameters in the linear regression model.

After constructing the regression tree and linear regression models, M5 eliminates the parameters of its linear regression models to minimize the error on query instances. Even though the elimination of parameters generally causes the error on query instances to increase, it also reduces the multiplicative factor $(n + v)/(n - v)$. Therefore, the multiplied error value decreases. M5 uses a greedy search to remove parameters that contribute little to the model; in some cases, M5 removes all of the parameters, leaving only a constant [22].

Finally, a pruning strategy, which is the same as that of RETIS, can be employed. A nonterminal node is pruned if its linear regression model gives less prediction error than its subtree. Pruning strategy is employed by starting near the bottom.

2.5 Multivariate Adaptive Regression Splines

The CART decision tree induction model's major drawback is the lack of continuity. Piecewise constant values are assigned to the subregions, and sharply discontinuous patterns are formed at subregion boundaries. The second drawback of CART is its inability to produce good approximations to some functions, including very simple linear functions. MARS (Multivariate Adaptive Regression Splines) was developed to overcome these drawbacks [15]. It gives better accuracy when compared to the CART decision tree induction model.

MARS is a flexible regression modeling of high dimensional data. It takes the form of an expansion in product spline basis functions, where the number of product spline basis functions as well as the parameters associated with each one (product degree and knot locations) are automatically determined by

the data [15]. Actually, the procedure is implemented by constructing a set of globally defined product spline basis functions that span the space of qth order spline approximations and by fitting the coefficients of the expansion to the data by ordinary least-squares. MARS is motivated by the recursive partitioning approach used in regression tree induction models. However. MARS is different from those models in that it produces continuous models in the subregions. It has more power to model relationships that are nearly additive or involve interactions in at most a few predictor features. In addition. it can be represented in a form that separately identifies additive and interaction contributions.

Each product spline basis function is a low order polynomial belonging to a different subregion of the training data set. The actual function is approximated as an expansion of these product spline basis functions. This is called as piecewise parametric fitting of the data set.

A product spline basis function is univariate, if it is in the following form:

$$[\pm(x-t)]_+^q \tag{2.13}$$

where t is the knot location, q is the order of the spline, and the subscript indicates the positive part of the argument. That is, the subscript indicates a value of zero for negative values of the argument. For $q > 0$, the spline approximation is continuous.

Although any $q > 0$ guarantees the spline approximation to be continuous, MARS selects the value of 1 for q for a simple implementation. The use of splines handles the lack of continuity problem. A general review of splines is given in [12].

The use of splines causes subregions to involve functions having high order interactions among predictor features. At each split of a region, the parametric function of that region is removed, and two new parametric functions are constructed for two child regions. These new functions involve one more variable than the parent region's function. The interaction order among predictor features increases by 1. As a consequence of having such complex parametric

functions, having high order interactions, it becomes difficult to approximate some functions, including very simple linear functions.

MARS (shown in Figure 2.9) also handles this situation. It does not delete the lower order parametric function of the parent region after splitting it. Therefore, many such splits can be performed on the same parent. By employing this strategy, MARS does not increase the depth of the model and simple functions such as linear ones are well approximated since permitting a parent region to be splitted more than once gives an additive property to the model.

A product spline basis function is multivariate, if it is in the following form:

$$B_m^{(q)}(\mathbf{x}) = \prod_{k=1}^{K_m} [s_{km} \cdot (x_{v(k,m)} - t_{km})]_+^q \tag{2.14}$$

where the quantity K_m is the number of splits that gave rise to B_m, and The quantity s_{km} takes $(+/-)1$ values indicating the right/left portions, $v(k,m)$ label the predictor features, and t_{km} represent values on the corresponding predictor features. The discussion about the selection of q is given in [15].

Multivariate spline basis functions may involve the same predictor feature more than once. For $q > 0$, higher orders than q may be produced on such predictor features. MARS handles this problem by restricting the multivariate spline basis functions to involve distinct features as shown in line 4 of Figure 2.9.

Finally, a pruning strategy can also be employed in MARS. The child regions need not be deleted in pairs as in CART. Because, the parent region is not deleted in MARS, there will not be any holes left in the model.

[1] $B_1(\mathbf{x}) \leftarrow 1$; $M = 2$
[2] Loop until $M > M_{max}$: $lof^* \leftarrow \infty$
[3] For $m = 1$ to $M - 1$ do :
[4] For $v \notin \{v(k,m)|1 \leq k \leq K_m\}$
[5] For $t \in \{x_{vj}|B_m(x_j) > 0\}$
[6] $g \leftarrow \sum_{i=1}^{M-1} a_i B_i(\mathbf{x}) + a_M B_m(\mathbf{x})[+(x_v - t)]_+$
$+a_{M+1} B_m(\mathbf{x})[-(x_v - t)]_+$
[7] $lof \leftarrow min_{a_1 \ldots a_{M-1}} LOF(g)$
[8] if $lof < lof^*$, then $lof^* \leftarrow lof$; $m^* \leftarrow m$; $v^* \leftarrow v$; $t^* \leftarrow t$ end if
[9] end for
[10] end for
[11] end for
[12] $B_M(\mathbf{x}) \leftarrow B_{m^*}(\mathbf{x})[+(x_{v^*} - t^*)]_+$
[13] $B_{M+1}(\mathbf{x}) \leftarrow B_{m^*}(\mathbf{x})[-(x_{v^*} - t^*)]_+$
[14] $M \leftarrow M + 2$
[15] end loop
[16] end algorithm

Figure 2.9. MARS Algorithm

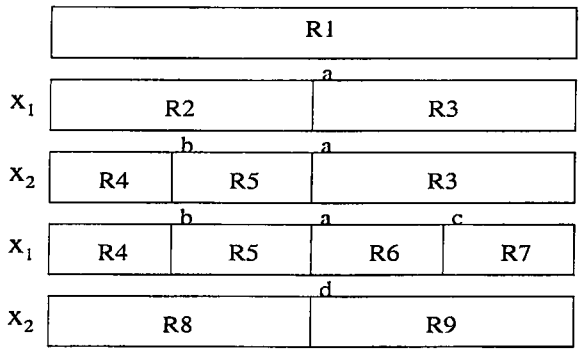

Figure 2.10. An example for the regions of MARS algorithm

Chapter 3

Regression by Selecting Best Feature Projections

In this chapter we describe the new regression method called Regression by Selecting Best Feature Projections (RSBFP). RSBFP is an eager, parametric and adaptive method which makes use of feature projections and least squares regression. All such properties of RSBFP will be described and discussed in detail in the chapter.

3.1 The RSBFP Algorithm

RSBFP constructs simple linear regression lines for each feature by using the projections of the training instances on each feature dimension separately. In the case of linear (ordered) valued features, exactly one simple linear regression line is constructed. On the other hand, in the case of categorical (unordered) valued features, exactly one simple linear regression line per each distinct value of a categorical feature is constructed. All of these simple linear regression lines are then sorted according to their predictive power, and the sorted list determines the induced parametric model. A target noise elimination procedure is employed to increase the predictive power of the model before construction of the regression lines.

All query instances use the same induced model, which makes RSBFP an eager method. That is, a different model for each query instance is not constructed as in lazy methods. Although there is exactly one induced model, the choice of the appropriate regression line in the model differs for each query instance, which gives RSBFP flexibility in terms of selecting the appropriate regression line.

3.1.1 Training

Training begins by storing the training instances as their projections on each feature dimension separately. A copy of target value is associated with each feature dimension. In the existence of missing feature values, the training instance is stored only on feature dimensions whose values are known. That is, the training instance is not simply ignored when it has some missing feature values. An example training set with four features and ten training instances projected to these features is shown in Figure 3.1.

f_1	:	9	8	7	10	6	7	4	5	8	4
TARGET	:	7	6.9	8	10	7.4	11	6.9	18	7.11	2
f_2	:	8	7	6	6	5	8	4	2	7	3
TARGET	:	7	6.9	8	10	7.4	11	6.9	18	7.11	2
f_3	:	*A*	*B*	*B*	*B*	*B*	*B*	*A*	*B*	*A*	*B*
TARGET	:	7	6.9	8	10	7.4	11	6.9	18	7.11	2
f_4	:	*X*	*X*	*Y*	*Y*	*Z*	*Z*	*Z*	*Z*	*Y*	*Y*
TARGET	:	7	6.9	8	10	7.4	11	6.9	18	7.11	2

Figure 3.1. An example training set projected to four features: f_1, f_2, f_3 and f_4.

After storing the training instances as their projections on the feature dimensions, simple linear regression lines are constructed at each feature dimension and sorted according to their predictive power to induce a parametric model. The use of simple linear least squares regression and the construction

of the model will be described in the next two sections.

3.1.1.1 Simple Linear Least Squares Regression

Simple linear least squares regression can be applied when the parametric form of the model is assumed to be linear, and consists of a single feature. The parametric form is given in Equation 3.1, and the task is to approximate coefficients of this equation using the least squares error criterion in Equation 3.2.

$$\hat{y}_{qf} = \beta_{0f} + \beta_{1f} x_{qf} \tag{3.1}$$

here, x_q is the query point, x_{qf} is the f^{th} feature value of the query, β_{0f} and β_{1f} are the two parameters of the linear function and $\hat{y}_{qf}$ is the approximation for query instance at feature f.

$$E_f = \sum_{i=1}^{n} (y_i - \hat{y}_{if})^2 \tag{3.2}$$

where n is the number of training instances, $\hat{y}_{if}$ is the approximation for training instance at feature f, and y_i is the actual target value of the training instance.

The parameters of (3.1), β_{0f} and β_{1f} for each feature f are computed as the following:

By taking the derivatives of (3.3) to minimize the error E_f, the parameters β_{0f} and β_{1f} are determined for linear least squares approximation.

$$E_f = \sum_{i=1}^{n} (y_i - \beta_{0f} - \beta_{1f} x_{if})^2 \tag{3.3}$$

From $\frac{\partial E}{\partial \beta_{0f}} = 0$

$$n\beta_{0f} + \beta_{1f} \sum_{i=1}^{n} x_{if} = \sum_{i=1}^{n} y_i \tag{3.4}$$

From $\frac{\partial E}{\partial \beta_{1f}} = 0$

$$\beta_{0f} \sum_{i=1}^{n} x_{if} + \beta_{1f} \sum_{i=1}^{n} x_{if}^2 = \sum_{i=1}^{n} x_{if} y_i \tag{3.5}$$

By solving the above equations, β_{0f} and β_{1f} are found as follows.

$$\beta_{0f} = \frac{\sum_{i=1}^{n} y_i - \beta_{1f} \sum_{i=1}^{n} x_{if}}{n} \tag{3.6}$$

$$\beta_{1f} = \frac{SP_f}{SSx_f} \tag{3.7}$$

where

$$SP_f = \sum_{i=1}^{n} x_{if} y_i - \frac{(\sum_{i=1}^{n} x_{if})(\sum_{i=1}^{n} y_i)}{n} \tag{3.8}$$

and

$$SSx_f = \sum_{i=1}^{n} x_{if}^2 - \frac{(\sum_{i=1}^{n} x_{if})^2}{n} \tag{3.9}$$

3.1.1.2 Model Construction

Simple linear least squares regression is applied to obtain the simple linear regression lines. For each linear (ordered) valued feature, a unique regression line is constructed. But if all the training instances have the same linear value for a particular feature dimension, the slope of the simple linear regression line will be infinity. This situation can be determined by looking at the value of SSx_f in Equation 3.9. If $SSx_f = 0$, it will not be possible to apply the linear least squares approximation.

Although linear features sometimes encounter this problem, categorical features always encounter the same problem. The number of regression lines is equal to the number of distinct values for each categorical (unordered) valued feature. And $SSx_f = 0$ for any C value of any categorical feature f, again leading to *division by* 0 situation. Those problematic situations can be handled by taking β_{1f} parameter in Equation 3.7 as zero. It is also observed that β_{0f} parameter always gives the mean target value of the training instances in such problematic situations.

Upon the construction of the simple linear regression lines, model construction phase continues by sorting those lines according to their predictive power

to induce the model. The relative error measure, RE, in Equation 3.10 is employed to determine the predictive power of any regression line. The smaller the relative error, the stronger the predictive power of the corresponding regression line.

$$RE = \frac{\sum_{i=1}^{n} |t(q_i) - \hat{t}(q_i)|}{\sum_{i=1}^{n} |t(q_i) - \bar{t}|} \tag{3.10}$$

where n is the number of training instances used to construct the simple linear regression line, $\bar{t}$ is the median of the target values of n training instances, $t(q_i)$ is the actual target value of i^{th} training instance and $\hat{t}(q_i)$ is the predicted target value of the i^{th} training instance.

To illustrate the model construction phase, the training set given in Figure 3.1 is used. The training set consists of four features, and the set will construct a total of seven simple linear regression lines. The decision list of these lines according to their associated relative errors is shown in Figure 3.2. The induced model shows that the predictive power of any categorical feature may vary among its values. In the given example, f_4 is very powerful for X value, although it is very poor on the remaining values, Y and Z.

[1]	$Y = 6.950$	$if\ f_4 = X,$	$RE = 1.000$
[2]	$Y = 7.003$	$if\ f_3 = A,$	$RE = 1.016$
[3]	$Y = 6.249f_1 + 0.243$	$if\ 4 \leq f_1 \leq 10,$	$RE = 1.043$
[4]	$Y = 5.882f_2 + 0.338$	$if\ 2 \leq f_2 \leq 8,$	$RE = 1.081$
[5]	$Y = 8.660$	$if\ f_3 = B,$	$RE = 1.098$
[6]	$Y = 8.370$	$if\ f_4 = Y,$	$RE = 1.128$
[7]	$Y = 8.433$	$if\ f_4 = Z,$	$RE = 1.252$

Figure 3.2. The decision list of simple linear regression lines (the model learned).

The model construction process in RSBFP is summarized in Figure 3.3.

3.1.2 Querying

In the querying phase of the RSBFP, for a query instance the most predictive simple linear regression line is tried first. However, it is sometimes not possible

[1] For $f = 1$ to p
[2] if f is a linear feature
[3] Construct a simple linear regression line of the form $\hat{y}_f = \beta_{0f} + \beta_{1f}.x_f$
[4] end if
[5] else (f is a categorical feature)
[6] For each distinct value c of f
[7] Construct a simple linear regression line of the form $\hat{y}_{fc} = \beta_{0fc}$
[8] end for
[9] end else
[10] end for
[11] Sort the simple linear regression lines according to their relative errors
[12] Store the ordered list of $\hat{y}_f$'s

Figure 3.3. Model Construction in RSBFP

to use the most predictive line due to different feature value, missing feature value and out of range problems. Therefore, the search for the next best regression line continues until a suitable one is found. The querying phase can be explained through an example. We will again refer to the training set given in Figure 3.1.

Step	Regression Line Tested		Result	Reason
[1]	$Y = 6.950$	*if* $f_4 = X$	**not suitable**	$f_4(= Y) \neq X$
[2]	$Y = 7.003$	*if* $f_3 = A$	**not suitable**	$f_3(=?) \neq A$
[3]	$Y = 6.249f_1 + 0.243$	*if* $4 \leq f_1 \leq 10$	**not suitable**	$f_1(= 12)$ *is not in the range* $[4, 10]$
[4]	$Y = 5.882f_2 + 0.338$	*if* $2 \leq f_2 \leq 8$	**suitable**	$f_2(= 5)$ *is in the range* $[2, 8]$

Prediction : $Y = (5.582 * 5) + 0.338$

Figure 3.4. An example for querying phase

Figure 3.4 shows an example for querying phase. The query instance used in the example is $Q :< 12, 5, ?, Y >$. Q uses the 4^{th} best regression line of the model to approximate its target feature value (Y).

The querying phase in RSBFP is summarized in Figure 3.5.

```
[1]  Prediction ← 0; v ← 1; Suitable-Regression-Line-Found = FALSE
[2]  u ← Number of simple linear regression lines
[3]  Simple linear regression lines are in sorted order

[4]  While Suitable-Regression-Line-Found = FALSE and v < u + 1
[5]     f ← Predictor feature of v^th simple linear regression line
[6]     if x_qf is known then
[7]        if f is a linear feature then
[8]           if x_qf is in the range of x_f values of the training data then
[9]              Prediction ← Prediction + ŷ_f(x_qf)
[10]             Suitable-Regression-Line-Found = TRUE
[11]          end if
[12]       end if
[13]       else (f is a categorical feature)
[14]          if x_qf = v's categorical value for f
[15]             Prediction ← Prediction + ŷ_f(x_qf)
[16]             Suitable-Regression-Line-Found = TRUE
[17]          end if
[18]       end else
[19]    end if
[20]    else
[21]       v ← v + 1
[22]    end else
[23] end while
```

Figure 3.5. Querying Phase in RSBFP

3.1.3 Target Noise Elimination

The distribution of the target values of the instances determines the success of the model induced by RSBFP. The induced model will be more predictive if the target values do not deviate so much from their mean value. However, in real life databases, it is generally not possible to find such a smooth distribution of the target values. Therefore, a target noise elimination procedure can be employed as a preprocessing step of the model induction process.

In this preprocessing step, training instances whose target values are within

k standard deviations of the mean target value are selected as non-noisy training instances to be used in the training phase of RSBFP method. The remaining instances whose target values are outside the k standard deviations of the mean target value are regarded as noisy training instances. The k value maximizing the predictive accuracy of the RSBFP method on the data sets. which were used in our emperiments, was empirically determined to be $\sqrt{2}$. as shown in Figure 3.6. For this k value, the mean relative error of the RSBFP method on our data sets is minimized, and therefore the predictive accuracy is maximized.

The target noise elimination procedure causes RSBFP to use less number of training instances to induce its parametric model. The selection of $\sqrt{2}$ guarantees that at least 50% of the training instances will be used in the training phase according to *Chebyshev's Result.*

There is a remarkable result discovered by the Russian mathematician Chebyshev that uses the standard deviation to determine the proportion of values in a population that is within a specified distance from the mean. *Chebyshev's Result* [18] is as the following:

1. *At least* 75% of the values in any population of numbers are within 2 standard deviations of the mean (that is, at least 75% of the population values have Z-scores between -2 and 2 inclusive).

2. *At least* 88% of the values in any population of numbers are within 3 standard deviations of the mean (that is, at least 88% of the population values have Z-scores between -3 and 3 inclusive).

3. For any positive number k, *at least* $(1 - 1/k^2) * 100\%$ of the values are within k standard deviations of the mean (that is, at least $(1 - 1/k^2) * 100\%$ of the population values have Z-scores between $-k$ and k inclusive).

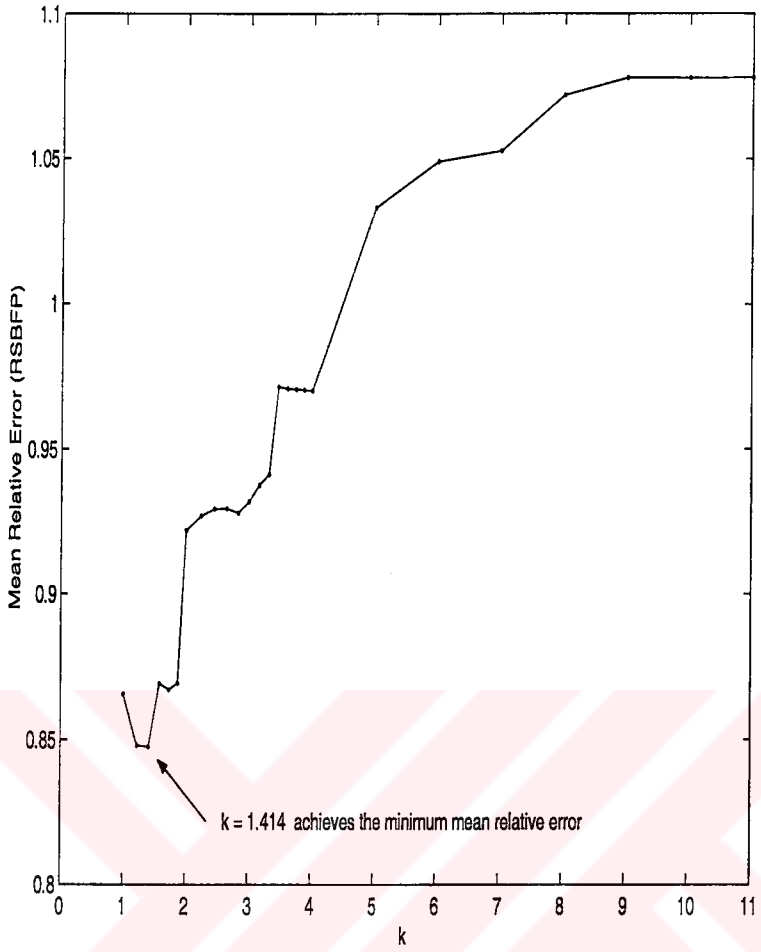

Figure 3.6. Determination of the optimal value of k for the RSBFP method.

3.2 Properties of RSBFP

In this section, we describe properties and problems encountered in regression algorithms and evaluate RSBFP in terms of these topics.

3.2.1 Eager Learning

Eager learning methods complete most of the processing in the training phase by inducing a model. This global model is used to fit all the training instances. It has both advantages and disadvantages over *lazy learning* methods. Eager learning methods enable interpretation of the underlying data by human beings. They generally give more accurate approximations, since they generalize

the whole data set. They are also very fast in the prediction phase. Query instances just use the induced model, rather than inducing models in the prediction phase. However, eager methods are not suitable for cases where the training data changes dynamically. RSBFP, being an eager approach, has the properties of eager methods explained above.

3.2.2 Context-sensitive (Adaptive) Learning

A regression method has a context-sensitive nature, if the contribution of the features varies in different locations of the instance space. This property is achieved in the categorical features in RSBFP. The contribution of a categorical feature may differ among its values. Although the feature's predictive power is large in some values, the same feature may be the worst feature among other values in terms of its predictive power.

3.2.3 Different Feature Types

Regression methods accept two type of features, *categorical* or *linear*. Categorical features take unordered values, whereas linear features take ordered (linear) values. Regression methods differ at handling these two type of features. Any one or both of these two types should be handled. If the regression method handles only linear features, then the categorical ones can be replaced with a unique linear one. If the method handles only categorical features, then a clustering procedure is employed to transform linear features to categorical ones. In most of the cases, each cluster is given a unique categorical value, and the linear feature values are replaced with their corresponding cluster's categorical value.

These transformation procedures are not only time consuming but also open to erroneous situations. RSBFP does not employ any transformation strategy. It is designed to handle both type of features.

3.2.4 Curse of Dimensionality

Many regression methods mentioned in Chapter 2 suffer from sparsity for very large feature dimensions and with moderate number of training instances. In other words, much more training instances are required to make better approximations, as the feature dimension increases. This problem can be explained through an example. Consider an input space consisting of exactly one feature. Also let the training instances be uniformly distributed among this feature. having values ranging from 1 to 2. In this scenario, a random choice of half of the feature dimension will contain half of the training instances. If we add one more feature having the same properties to the previous one, using random choices of halves of each feature dimension will contain 1/4th of the training instances. A further addition of a new feature will cause this ratio to decrease to 1/8. That is, increasing the feature dimension will lead much sparse instance spaces.

In machine learning community, this problem is called as *curse of dimensionality.* In RSBFP, linear regression lines are constructed on each feature dimension separately. As the number of feature dimensions increases, the density of training instances at any feature projection does not change. This shows that RSBFP is suitable for data sets with large number of features.

3.2.5 Normalization of Features

Normalization procedure is important for regression methods that make use of Euclidean distance measure. For instance, kNN regression method heavily uses Euclidean distance measure to determine the k nearest neighbors. The values of all the predictor features need normalization to ensure that all features contribute equally in the computation of distance. The lack of normalization procedure may lead to erroneous situations. For example, a linear predictor feature whose unit is miles may lead to different nearest neighbors than the same linear predictor feature whose unit is kilometers. Therefore, normalization causes regression methods to operate independent of the domain knowledge.

Although RSBFP does not employ any normalization procedure, it can also operate independent of the domain knowledge. Because, it makes use of feature projections that naturally handle the normalization problem.

3.2.6 Irrelevant Features

Eliminating irrelevant features is also an important concept in machine learning. Lazy methods generaly suffer from irrelevant features. For instance, in kNN regression method, nearest k neighbours are determined according to the Euclidean distance measure. While computing Euclidean distances of training instances to the query instance, all features are given equal weight. The existence of irrelevant features may cause beneficial training instances to be thrown away.

On the other hand, eager regression methods are successful in elimination of the irrelevant features. For instance, in regression tree induction methods, the partitioning begins from the most significant feature and continues with less relevant features in the subsequent partitionings.

The interpretation of the model induced by RSBFP shows that it is somewhat similar to the regression tree induction methods. Because, it places the most relevant feature and its simple linear regression line on top of the model. The other features, along with their equations, come next in the model hierarchy.

3.2.7 Redundant Features

There are two types of redundant features. The first one is due to the existence of same feature more than once in the database. The second one is due to the existence of functional dependencies between features. That is, a feature may actually be a combination of some of the other predictor features.

RSBFP is suitable for both type of redundant features. If a feature is repeated more than once in the database, then the simple linear regression

line of that feature will be duplicated in the model. This will not affect the querying phase of the method, the model will just have a massy appearance. Furthermore, if a feature is a combination of the other ones, then this situation will not effect the querying phase, either.

The redundant feature situation is a problem in the regression methods especially making use of feature projections. Because the contribution of the features in the querying phase may be doubled, tripled etc.

3.2.8 Missing Feature Values

In real-life databases, some feature values may be unknown for some instances or tuples. And it will not be suitable to completely discard an instance having missing feature values in regression methods. There are different approaches to handle missing feature value problem. The most common approach is to fill those places with some constant values. These constant values may be the mean of the known values or the mostly encountered value of the corresponding feature. However, this may cause distortion of the data set. Many learning methods can not prevent this distortion caused by filling missing feature values [31, 32, 34]. If missing feature values are very frequently seen on some instances or features, removing these instances or features can also be employed.

RSBFP leaves those places empty to provide a natural solution. By this way, the contents of the original data set is not distorted. RSBFP uses the known values of each feature projection to construct the simple linear regression lines.

3.2.9 Noise

In data sets, two types of noise can occur, *predictor feature noise* and *target feature noise*. Although most of the regression methods are robust to predictor feature noise, they are not so robust in the case of target feature noise.

For target feature noise problem, in RSBFP, the training instances whose target values are not in some predefined range are avoided in the model construction phase. But at least 50% of the training instances are employed in the training phase. The empirical results show that robustness to target feature noise in RSBFP is better than some other well known regression methods.

3.2.10 Bias-variance Trade-off

There are two error types, *bias* and *variance*, that affect the success of the regression algorithms. *Bias* is a consequence of underfitting the training data, whereas *variance* is a consequence of overfitting the training data. It is generally not possible to decrease both of these errors simultaneously. Because, a decrease in *bias* leads to an increase in *variance*. As an example, kNN regression method causes a large *bias* error if a large k is chosen. On the other hand, kNN will lead to a large *variance* error if a small k is chosen.

In regression algorithms, this trade-off always exists. RSBFP chooses to minimize the *variance* error. It makes strong assumptions about the training data. A detailed information about this trade-off concept is presented in [17].

3.2.11 Model Complexity and Occam's Razor

William of Occam's Razor principle states that "Entities should not be multiplied beyond necessity" [13]. This principle has been approved by machine learning community as in many other communities. Given two learning algorithms with the same accuracy, the simpler one must be selected. This is especially the case if our concern is interpretation of the induced model.

The model constructed by RSBFP is a simple decision list. Therefore, RSBFP follows the Occam's Razor principle.

3.2.12 Interpretation

The interpretation of the induced model by human beings is an important trait of a machine learning algorithm. Eager regression methods' induced models are usually easy to interpret, whereas lazy methods' ones are hard to interpret. RSBFP, as a parametric eager approach, constructs global models. By analyzing the model, the importance of the predictor features and the important segments of the categorical predictor features can be determined.

3.2.13 Interactions and Lack of Many Additive Terms

RSBFP has two main limitations. Firstly, it is not suitable for domains having interactions among its features. Furthermore, the linear regression lines of the model are simple rather than multiple. That is, the linear regression lines consist of at most one additive term. This generally causes large bias error in the induced model. Nevertheless, the empirical evaluations indicate that RSBFP achieves comparable accuracy values in spite of these limitations.

If the effect of any predictor feature on the target feature is dependent on some of the other predictor features, then this indicates the existence of interactions in the data set. For example, we can not determine the increase in the area of a rectangle when the width of the rectangle is increased. The increase amount depends on the particular value of the height of the rectangle. The area calculation formula involves, actually consists of just, one interaction term which is *width * height.*

On the other hand, we can determine the increase in the perimeter of a rectangle when the width of the rectangle is increased. The increase amount does not depend on the particular value of the height of the rectangle. The perimeter calculation formula involves, actually consists of, exactly two additive terms which are *width* and *height.*

The induced model of RSBFP does not have any interaction terms, and involves at most one additive term. The second drawback is handled in RSBF

(Regression by Selecting Best Features) method that will be explained throughout the Chapter 4. But the interaction problem still remains in the induced model of RSBF.

3.3 Complexity Analysis

RSBFP is an eager method, and stores the simple linear regression lines in the memory. Given a data set with n instances and m features, where m_l of the features are linear and $(m - m_l)$ are categorical, we let c to denote the maximum number of values of a categorical feature. Then, in the worst case, the number of simple regression lines will be $m_l + (m - m_l).c$. Therefore, the space complexity of RSBF method will be $O(m_l + (m - m_l).c)$.

The computational complexity of the method differs for the training and the querying phase. In the training phase, determination of the training instances that are free of target noise requires an $O(n)$ time complexity. Storing and sorting the non-noisy training instances at each feature separately requires $O(m.n)$ and $O(m.n.\log n)$, respectively. Construction of the simple linear regression lines at each feature takes a total of $O(m_l + (m - m_l).c)$ time. The training phase ends by sorting the simple linear regression lines, which brings a cost of $O((m_l + (m - m_l).c).\log(m_l + (m - m_l).c))$. The computation of the mean and the standard deviation of the target values is also computed in the training phase, which causes an extra $O(n)$ time complexity.

In the training phase, the computational complexity is the sum of the components described above; $O(n) + O(m.n) + O(m.n.\log n) + O(m_l + (m - m_l).c)$ $+ O((m_l + (m - m_l).c).\log(m_l + (m - m_l).c)) + O(n)$. So the complexity of the training phase is $O((m_l + (m - m_l).c).\log(m_l + (m - m_l).c) + m.n.\log n)$.

In the querying phase, a query instance will have to search all the linear regression lines of the model to find a suitable regression line, in the worst case. Since $m_l + (m - m_l).c$ is the upper bound for the number of regression lines, the computational complexity of the querying phase for a single query instance is $O(m_l + (m - m_l).c)$.

The empirical results show that RSBFP is fast both in training and querying phase, while preserving a comparable predictive accuracy.

Chapter 4

Regression by Selecting Best Features

In this chapter we describe another regression method called Regression by Selecting Best Features (RSBF). RSBFP, described in the previous chapter, method was incapable of inducing models involving additive terms. RSBF was developed to override this limitation. It is also an eager, parametric and adaptive method which makes use of feature projections and least squares regression. All such properties of RSBF will be described and discussed in detail in the chapter.

4.1 The RSBF Algorithm

RSBF's induced model is similar to that of RSBFP. The categorical features are handled in the same way. Exactly one simple linear regression line per each distinct value of a categorical feature is constructed. However, linear features are handled in a different way. Multiple linear regression lines are constructed among linear features. The number of multiple regression lines is equal to the number of linear features. The linear features to be involved in each multiple regression line are determined by using a relevancy heuristics. The induced model consists of an ordered list of multiple and simple linear regression lines

according to their predictive power. A target noise elimination procedure is also employed to increase the predictive power of the model before construction of the regression lines.

RSBF is an eager method, since all query instances use the same induced model. A different model per query instance is not constructed, as in lazy regression methods. RSBF is also a flexible method since the choice of the appropriate regression line differs for each query instance.

4.1.1 Training

Training phase again begins by storing the training instances as their projections on each feature dimension separately. A copy of the target value is associated with each feature dimension. The existence of missing feature values is handled in a natural way. RSBF stores an instance only on feature dimensions whose values are known. An example training set with five features and ten training instances projected to these features is shown in Figure 4.1.

f_1	:	3	5	4	7	10	−11	6	0	−12	20
TARGET	:	15	14.5	13.6	15.2	17.3	15	13.4	13.9	18.7	15
f_2	:	4	−10	5	−4	−3	−5	9	12	−4	−12
TARGET	:	15	14.5	13.6	15.2	17.3	15	13.4	13.9	18.7	15
f_3	:	1	7	−5	12	2	10	15	−3	6	9
TARGET	:	15	14.5	13.6	15.2	17.3	15	13.4	13.9	18.7	15
f_4	:	*A*	*B*	*B*	*B*	*B*	*A*	*B*	*B*	*B*	*A*
TARGET	:	15	14.5	13.6	15.2	17.3	15	13.4	13.9	18.7	15
f_5	:	*Z*	*Y*	*Y*	*Z*	*Y*	*Y*	*X*	*X*	*Z*	*Z*
TARGET	:	15	14.5	13.6	15.2	17.3	15	13.4	13.9	18.7	15

Figure 4.1. An example training set projected to five features: f_1, f_2, f_3, f_4 and f_5.

After storing the training instances as their projections on the feature dimensions, the linear regression lines are constructed. Multiple linear regression lines belonging to the linear features and simple linear regression lines belonging to categorical features are used to induce the model. The use of simple least squares regression was explained in Chapter 3. The next two sections describe the multiple linear least squares regression and the construction of the model.

4.1.1.1 Multiple Linear Least Squares Regression

Multiple linear least squares regression can be applied when the parametric form of the model is assumed to be linear, and consists of multiple features. The parametric form is given in Equation 4.1, and the task is to approximate the coefficients of this equation by using the least squares error criterion in Equation 3.2. Simple linear least squares regression is the primitive form of the multiple version since it employs exactly one feature.

$$\hat{y}_q = \sum_{j=1}^{p} \beta_j . x_{qj} + \beta_0 \tag{4.1}$$

here, p is the number of features, x_q is the query point, x_{qj} is the j^{th} feature value of the query, β_j is the j^{th} parameter of the function and $\hat{y}_q$ is the approximated value of the function for the query point x_q.

For a training data set consisting of n instances and m_l linear features, matrix $A_{n\text{x}(m_l+1)}$ is used to store the feature values of the instances. Each a_{ij} entry of A denotes the j^{th} feature value of the i^{th} instance. Furthermore, the target feature values of the instances are stored in $y_{n\text{x}1}$ vector. Multiple linear least squares regression finds a least squares solution to $A\beta = y$ by employing the method of *Normal Equations* given in Figure 4.2. The least squares solution vector, β, includes the parameters of (4.1).

Normal Equations method makes use of *Cholesky Factorization* that is explained in Figure 4.3. Cholesky Factorization overwrites the lower triangular portion of C matrix, and the overwritten lower triangular portion is assigned to the G matrix.

[1] Multiply both sides of $A\beta = y$ by A^T
[2] $\rightarrow A^T A\beta = A^T y$
[3] $C = A^T A$ and $d = A^T y$
[4] Compute the *Cholesky Factorization* of C
[5] $\rightarrow C = GG^T$ where G is a lower triangular matrix
[6] $z = G^T\beta$
[7] Solve $Gz = d$ and $G^T\beta = z$

Figure 4.2. Normal Equations Method

[1] For $k = 1$ to m do
[2] if $C(k,k) = 0$ then
[3] goto [1]
[4] end if
[5] $C(k,k) = \sqrt{C(k,k)}$
[6] For $j = (k+1)$ to m do
[7] $C(j,k) = C(j,k)/C(k,k)$
[8] end for
[9] For $j = (k+1)$ to m do
[10] For $i = j$ to m do
[11] $C(i,j) = C(i,j) - C(i,k)C(j,k)$
[12] end for
[13] end for
[14] end for

Figure 4.3. Cholesky Factorization

4.1.1.2 Model Construction

For each distinct value, C, of categorical feature f, the simple linear regression line will consist of one parameter, which is the mean target value of the training instances having the categorical value of C for f. In the case of linear features, exactly one simple linear regression line is constructed. Equation 3.10 is used to find the relative error, and therefore the predictive power, of the lines. Then, the regression lines belonging to linear features are sorted according to their predictive power. This sorting procedure is used as our relevancy heuristics in constructing multiple linear regression lines among linear features. The first

multiple linear regression line will consist of all the linear features. The second one will exclude the worst linear feature and use the remaining linear features. From this point, each incoming multiple linear regression line will exclude the next worst linear feature along with the previously discarded features. Therefore, the number of multiple linear regression lines will exactly be equal to the number of linear features.

Upon the completion of multiple regression line construction phase. the categorical features' simple and linear features' multiple linear regression lines will be sorted by using again the Equation 3.10. This last procedure will result in the induced model of the RSBF method, which is a decision list.

To illustrate the model construction phase, we will use the training set given in Figure 4.1. The training set consists of five features and the set will construct a total of five simple (two for f_4, three for f_5) and three multiple linear regression lines that will construct the model. To determine which of the linear features will participate in which of the multiple linear regression lines, we construct simple linear regression lines by using the linear features and measure how each one is successful by using only that feature for prediction purposes. The sorted order of these regression lines according to their associated relative errors are shown in Figure 4.4:

[1]	$Y = -0.068f_2 + 14.736$	$RE = 0.869$
[2]	$Y = 0.003f_3 + 14.753$	$RE = 1.027$
[3]	$Y = 0.031f_1 + 14.613$	$RE = 1.060$

Figure 4.4. The ordered list of simple linear regression lines of linear features.

Figure 4.4 shows that f_2 is the most relevant (predictive) linear feature, whereas f_1 is the least one. In constructing the first multiple linear regression line, none of the linear features are discarded. The second multiple regression line will discard the least relevant linear feature f_1, and the third one will discard the next least relevant linear feature f_3 along with the previously discarded linear feature f_1. This shows that the third multiple linear regression line is actually a simple linear regression line consisting of only the most relevant linear feature f_2. Therefore, the parametric form of the third multiple linear regression line is the same as that of f_2's simple linear regression line.

Model construction is completed by sorting these multiple linear regression lines along with the simple linear regression lines belonging to the categorical features. Figure 4.5 shows the induced model by using our example data set. The smaller the relative error, the larger the predictive power of the regression line in the model.

The model construction in RSBF is summarized in Figure 4.6.

[1]	$Y = 15$	$if\ f_4 = A,$	$RE = 0.000$
[2]	$Y = -0.08f_2 - 0.036f_3 + 14.92$		$RE = 0.863$
[3]	$Y = -0.068f_2 + 14.736$		$RE = 0.869$
[4]	$Y = 0.008f_1 - 0.077f_2 - 0.035f_3 + 14.884$		$RE = 0.890$
[5]	$Y = 13.650$	$if\ f_5 = X,$	$RE = 1.000$
[6]	$Y = 15.010$	$if\ f_5 = Y,$	$RE = 1.048$
[7]	$Y = 14.650$	$if\ f_4 = B,$	$RE = 1.049$
[8]	$Y = 15.067$	$if\ f_5 = Z,$	$RE = 1.333$

Figure 4.5. The induced model

4.1.2 Querying

In the querying phase of the RSBF, for a query instance the most predictive linear regression line is tried first. However, it is sometimes not possible to use the most predictive line due to different feature value and missing feature value problems. Therefore, the search for the next best regression line continues until a suitable one is found.

The querying phase can be better explained through an example. We will again refer to the training set given in Figure 4.1. If we let our query instance be $Q :< 2, 10, ?, B, Z >$, the best regression line of the model will not be suitable for this query instance since $f_4 = B$ rather than A in Q. The second best regression line, which is a multiple regression line in this case, will also be unsuitable since the value of f_3 is missing. Finally, the third regression line is suitable for Q, and this regression line of the model will be used to predict the target value of Q. The search will stop here and the remaining regression lines of the model need not be dealed anymore. That is, Q uses the 3^{th} best regression line of the model, and the value of Y for the query instance Q is

[1] For $f = 1$ to p
[2] if f is a linear feature
[3] Construct a simple linear regression line of the form $\hat{y}_f = \beta_{0f} + \beta_{1f}.x_f$
[4] end if
[5] else (f is a categorical feature)
[6] For each distinct value c of f
[7] Construct a simple linear regression line of the form $\hat{y}_{fc} = \beta_{0fc}$
[8] end for
[9] end else
[10] end for
[11] Let p_l be the number of linear features
[12] Sort the simple linear regression lines belonging to linear features according to their relative errors
[13] $s \leftarrow p_l$
[14] While $s > 0$
[15] Construct a multiple linear regression line by using s linear features of the form $\hat{y} = (\sum_{i=1}^{s} \beta_i x_i) + \beta_0$
[16] Exclude the current worst linear feature
[17] $s \leftarrow s - 1$
[18] end while
[19] Sort p_l multiple linear regression lines belonging to linear features and simple linear regression lines belonging to categorical features altogether
[20] Store the ordered list of (multiple and simple) linear regression lines

Figure 4.6. Model Construction in RSBF

predicted as $(-0.068 * 10) + 14.736$.

The querying phase in RSBF is summarized in Figure 4.7.

4.1.3 Target Noise Elimination

Target noise elimination procedure employed as a preprocessing step of the model induction process in RSBFP is also used in the case of RSBF. The k value maximizing the predictive accuracy of RSBF method on all the data sets was empirically determined to be $\sqrt{2}$, as shown in Figure 4.8. It can be observed that both RSBF and RSBFP has the same optimal value of k, which

```
[1]  Prediction ← 0; v ← 1; Suitable-Regression-Line-Found = FALSE
[2]  u ← Number of (multiple and simple) linear regression lines
[3]  Linear regression lines are in sorted order

[4]  While Suitable-Regression-Line-Found = FALSE and v < u + 1
[5]     if v is a simple linear regression line
[6]        f ← Feature of v^th simple linear regression line
[7]        if x_qf is known then
[8]           if x_qf = v's categorical value for f
[9]              Prediction ← Prediction + ŷ_f(x_qf)
[10]             Suitable-Regression-Line-Found = TRUE
[11]          end if
[12]       end if
[13]    end if
[14]    else (v is a multiple linear regression line)
[15]       Let v consist of j linear features
[16]       if x_q has known values for all of these j linear features
[17]          Prediction ← Prediction + ŷ(x_q)
[18]          Suitable-Regression-Line-Found = TRUE
[18]       end if
[19]    end else
[20]    v ← v + 1
[21] end while
```

Figure 4.7. Querying Phase in RSBF

is $\sqrt{2}$.

4.2 Properties of RSBF

In this section, we describe properties of and limitations encountered in RSBF regression method.

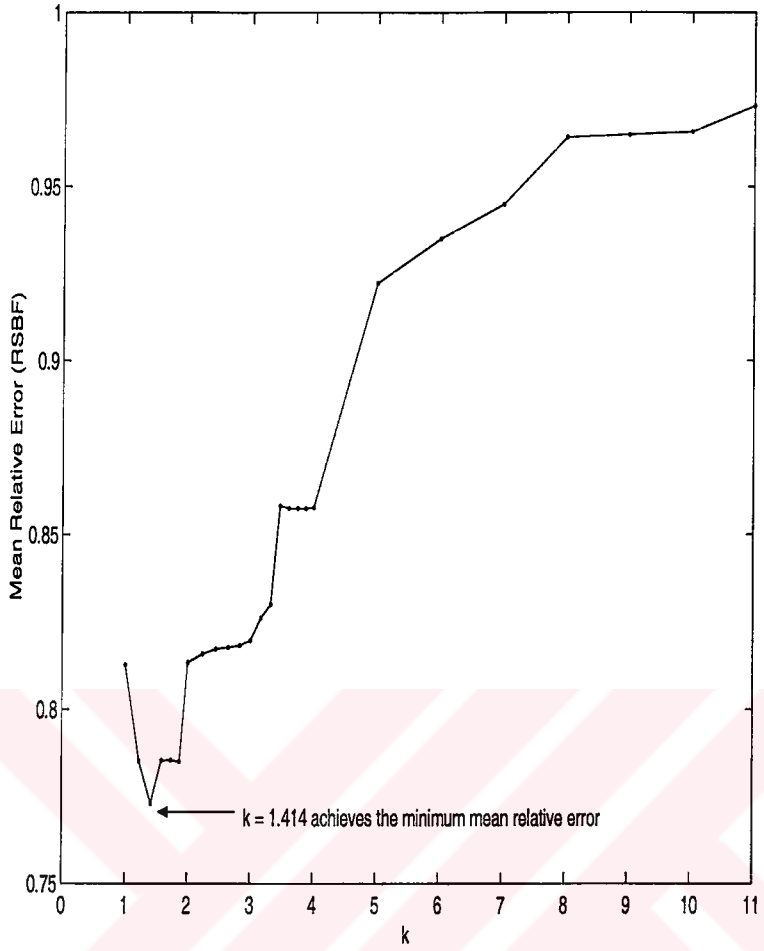

Figure 4.8. Determination of the optimal value of k for RSBF method

4.2.1 Eager Learning

RSBF is an example of eager learning methods. It induces a model in the training phase, and query instances use this model in the querying phase rather than constructing their own model around their query location. Being an eager method, it enables the interpretation of the training instances. In terms of time complexity, it is very fast in the querying phase while preserving a slower execution time in the training phase (Section 4.4).

4.2.2 Context-sensitive (Adaptive) Learning

Context-sensitive property of RSBF is satisfied by categorical features, as in RSBFP. The contribution of a categorical feature may differ among its values. It may be more predictive in some values, although it would be poor in the remaining values. The predictive segments of the categorical features are placed at top levels of the model, whereas the other segments stay at bottom levels.

4.2.3 Different Feature Types

RSBF method accepts both *categorical* and *linear* features. It combines these two type of features to induce its model. There is no need to transform any linear feature to a categorical one, or vice versa. No need for transformation ensures that erroneous situations will not be encountered.

4.2.4 Curse of Dimensionality

RSBF does not follow a partitioning strategy as in regression tree induction methods, which will cause the number of instances to decrease in proportional to the increase in the feature dimension. It always uses the whole training data set to construct simple and multiple linear regression lines. Therefore, RSBF is suitable for data sets with large number of features. There is not a *curse of dimensionality* problem in our method.

4.2.5 Normalization of Features

RSBF does not employ any normalization procedure. Although it does not cause any problems in the case of simple linear regression lines, the multiple linear regression lines may suffer if the range of the values greatly differs among linear features.

4.2.6 Irrelevant Features

The main motivation behind RSBF was to select the best subset of features and construct a linear regression line by using those features. However, RSBF does not limit itself to exactly one linear regression line. It constructs many linear regression lines to induce its model. The linear regression line consisting of the most predictive, relevant features is placed at the top of the model. The other lines come next in the model hierarchy. A linear feature may exist in more than one linear regression line, indicating that the linear regression lines are not isolated from each other.

4.2.7 Missing Feature Values

Missing (absent) feature value problem is commonly encountered in real-life databases. RSBF leaves these places empty, rather than assigning an arbitrary value to them. This guarantees that the original data set will not be distorted. The RSBF method uses the known values while projecting the training instances on each feature dimension. Also the regression functions involving a feature for which the corresponding value is missing in a query instance are not used.

4.2.8 Noise

The RSBF is robust to especially *target feature noise*. Training phase begins by avoiding instances whose target values are not in some predefined range. In spite of this elimination procedure, at least 50% of the training instances are employed in the training phase. The empirical results show that RSBF is very robust to target feature noise when compared to the other regression methods. (See Section 3.1.3 for details of the elimination of noisy instances)

4.2.9 Bias-variance Trade-off

There is always a trade-off between bias and variance. RSBF behaves similarly to RSBFP in that respect, and chooses to minimize the *variance* error. It makes strong assumptions about the underlying data set and tries to fit the data set into an ordered list of linear (multiple or simple) regression lines. As a consequence, RSBF generally underfits, and avoids overfitting the training data set.

4.2.10 Model Complexity and Occam's Razor

RSBF is not a complex regression method when compared to other eager and lazy methods in the literature. However, it achieves better accuracy values in addition to its simplicity and interpretability. The performance results of RSBF on real data sets confirm its compliance with the Occam's Razor principle, which states that given the two learning algorithms having the same accuracy, the simpler one should be selected. The decision list model constructed by RSBF is very simple compared to other regression techniques.

4.2.11 Interpretation

Interpretation of an induced model by human experts is important in machine learning. RSBF induces a global model that is easy to understand and interpret. By analyzing the linear regression lines of the model, one can determine the best subset of features and the predictive segments of the categorical predictor features.

4.2.12 Interactions and Lack of Combinations

RSBF has two main limitations. Firstly, it is not suitable for domains having interactions among its features. Furthermore, although the linear regression lines of the model are multiple, they do not try all the combinations of the

linear predictor features. This causes the induced model to include a bias error.

The induced model of RSBF does not include any interaction terms that were mentioned in Chapter 3. However, it is empirically shown that usually the real-world data sets do not contain interacting features [21, 19]. Besides this, while constructing the multiple linear regression lines, RSBF uses a relevancy heuristics. That is, once a linear feature is selected to be the next worst linear feature among the current set, it can not be reused in the construction of the further multiple linear regression lines. There is a need for having a greedy relevancy heuristics, because it is computationally infeasible to try each possible combination of the linear features to determine the best subset of features in terms of predictive accuracy.

4.3 Complexity Analysis

RSBF is an eager method, and stores the simple and multiple linear regression lines in the memory. Given a data set with n instances and m features, where m_l of the features are linear and $(m - m_l)$ are categorical, we let c to denote the maximum number of values of a categorical feature. Then, in the worst case, the number of simple and multiple regression lines will be $(m - m_l).c$ and m_l, respectively. Therefore, the space complexity of RSBF method will be $O(m_l + (m - m_l).c)$.

The computational complexity of the method differs for the training and the querying phase. In the training phase, determination of the training instances that are free of target noise requires an $O(n)$ time complexity. Storing and sorting the non-noisy training instances at each feature separately requires $O(m.n)$ and $O(m.n.\log n)$, respectively. Construction of the simple linear regression lines at each feature takes a total of $O(m_l + (m - m_l).c)$ time. The simple linear regression lines belonging to the linear features need sorting to be used in the construction of the multiple linear regression lines. This causes a $O(m_l.\log m_l)$ time complexity. Construction of the m_l multiple linear regression lines, by making use of *Normal Equations* method, takes $O(n.m_l^3 + m_l^4)$ [5].

The training phase ends by sorting the simple linear regression lines of categorical features and multiple linear regression lines of linear features altogether, which brings a cost of $O((m_l + (m - m_l).c).\log(m_l + (m - m_l).c))$. The computation of the mean and the standard deviation of the target values is also computed in the training phase, which causes an extra $O(n)$ time complexity.

In the training phase, the computational complexity is the sum of the components described above; $O(n) + O(m.n) + O(m.n.\log n) + O(m_l + (m - m_l).c) + O(m_l.\log m_l) + O(n.m_l^3 + m_l^4) + O((m_l + (m - m_l).c).\log(m_l + (m - m_l).c)) + O(n)$. Therefore, the overall computational complexity of the training phase is $O((m_l + (m - m_l).c).\log(m_l + (m - m_l).c) + n.m_l^3 + m_l^4 + m.n.\log n)$.

In the querying phase, a query instance will have to search all the linear regression lines of the model to find a suitable regression line, in the worst case. Since $m_l + (m - m_l).c$ is the upper bound for the number of regression lines, the computational complexity of the querying phase for a single query instance is $O(m_l + (m - m_l).c)$.

The empirical results show that RSBF is fast both in training and querying phases, while preserving a comparable predictive accuracy. Although querying time complexity of RSBF is exactly the same as that of RSBFP, training time complexity of RSBF is bigger than RSBFP. These results are also empirically verified on real-world data sets. The empirical evaluation of RSBFP and RSBF is given in the next chapter.

Chapter 5

Empirical Evaluations

This chapter is devoted to the empirical evaluations of RSBFP, RSBF and other regression methods mentioned in Chapter 2. A large number of real data sets, available in the Bilkent University Function Approximation Repository [20], are used to compare the predictive power and computational complexity of those methods.

The regression methods selected to compare with RSBFP and RSBF are successful representatives of rule-based learning, regression tree induction, spline-based regression and instance-based learning. RULE, KNN, DART and MARS are used as representatives of rule-based learning, instance-based learning, regression tree induction and spline-based regression, respectively. All of these methods are the most recent versions of their categories and outperform the previously implemented versions. The source code of these methods are obtained from publicly available resources.

The organization of the chapter is as follows: We first define the performance measure that will be used to compare the methods in terms of predictive accuracy. Then, a brief explanation of the particular implementation of the previously developed methods is given. The properties of the real data sets used in the experiments are also mentioned in this chapter. Finally, the empirical results including the predictive power of the methods, robustness to

the irrelevant features, missing feature values and target feature noise are discussed. The chapter concludes by computational complexity comparison of the methods in terms of training time and querying time requirements.

5.1 Performance Measure

The performance measure is used to determine the predictive power of the methods. The predictive power of a method is large if the actual target values of the query instances are close to the target values predicted by the proposed method. There are two commonly used performance measures: *mean absolute distance* and *relative error*. The second performance measure is exactly the same as the one used for calculating the predictive power of simple and multiple linear regression lines.

Mean absolute distance (MAD) [42, 43] is computed as the following:

$$MAD = \frac{\sum_{i=1}^{Q} |y_i - \hat{y}_i|}{Q} \tag{5.1}$$

where Q is the number of query instances.

However, MAD is not an appropriate performance measure. It depends on the range of the target values of the query instances. MAD will be large for domains having large target values, and small for domains having small target values [39]. Therefore, *relative error* (RE) [42, 43] is employed as the modified version of MAD. RE is an appropriate performance measure since it normalizes the MAD by the mean absolute distance from the median target value. RE is computed as the following:

$$RE = \frac{MAD}{\frac{1}{Q}\sum_{i=1}^{Q} |y_i - median(y)|} \tag{5.2}$$

10-fold cross validation technique [19] is employed on the experiments. Therefore, the prediction error of a method on any data set is computed as the average of 10 runs in each of which a disjoint set of 1/10 of the data set is

used in the querying and the remaining 9/10 in the training phase.

5.2 Other Regression Methods Used in Comparisons

In this chapter, we will briefly explain the properties of the implementations of the other regression algorithms used in experiments.

5.2.1 RULE

The most recent rule-based regression implementation, written by Weiss and Indurkhya [42, 43] is used in the experiments. The source code of the program is available in the data mining software kit (DMSK), attached to [43].

5.2.2 KNN

The version of KNN implemented by Bilkent University Machine Learning Group is used in our experiments. It makes use of Euclidean distance to determine the similarities of the training instances to the query point. A normalization procedure is also employed in order to obtain values ranging between 0 and 1.

Missing feature values are handled differently for categorical and linear features. If the feature is of linear type, mean values of the feature is used to fill the empty places. In the case of categorical features, the most frequent categorical value is selected to fill the empty places. K parameter is set to 10 in all the experiments.

5.2.3 DART

The most recent regression-tree induction implementation, written by Friedman [16] is used in the experiments. The regression-tree induction methods generally suffer from disjoint partitioning of the regions. However, DART avoids this problem by permitting to construct overlapping regions.

5.2.4 MARS

The most recent version of MARS, mars3.6, written by Friedman [15] is used in the experiments. Linear spline approximation produces better results than cubic spline approximations. So, *linear spline approximation* choice is set in our experiments. Also, the highest possible interaction level is set to figure out the interactions in our real data sets.

5.3 Real Data Sets

Twenty nine data sets, collected mainly from three sources [27, 8, 35], were used in the experiments. It was not easy to collect much more data sets, since most of the data sets are produced to be used in classification tasks. Table 5.1 shows the properties of these data sets. Further and detailed information about those data sets can be found in Bilkent University Function Approximation Repository [20].

5.4 Accuracy

Table 5.2 shows the relative errors *(RE)* of the regression methods on 29 real data sets. For each data set, the smallest relative error indicating the best result is typed in boldface. RSBFP and RSBF achieve the best results in 6 and 8 of these data sets, respectively. The other methods, DART, MARS, KNN and RULE, achieve the best results in 5, 5, 4, and 2 of these data sets,

Original Name	Abbrev.	Instances	Features (L+C)	Missing Values	Target Feature
Abalone	AB	4177	8 (7+1)	None	Rings
Airport	AP	135	4 (4+0)	None	Tons of mail
Auto	AU	398	7 (6+1)	6	Gas consumption
Baseball	BA	337	16 (16+0)	None	Salary
Buying	BU	100	39 (39+0)	27	Husbands buy video
Comp.Hard.	CH	209	7 (6+1)	None	CPU performance
Country	CN	122	20 (20+0)	34	Population
College	CO	236	25 (25+0)	381	Competitiveness
Education	ED	1500	43 (43+0)	2918	Income in 1991
Electric	EL	240	12 (10+2)	58	Serum 58
Fat	FA	252	17 (17+0)	None	Body height
Fishcatch	FC	158	7 (6+1)	87	Fish weight
Fruitfly	FF	125	4 (3+1)	None	Sleep time
Housing	HO	506	13 (12+1)	None	House prices
Homerunrace	HR	163	19 (19+0)	None	Run race score
Northridge	NE	2929	10 (10+0)	None	Earthquake magnit.
Normtemp	NT	130	2 (2+0)	None	Heart rate
Plastic	PL	1650	2 (2+0)	None	Pressure
Poverty	PV	97	6 (5+1)	6	Death rate
Read	RE	681	25 (24+1)	1097	Reader satisfaction
SolarFlare	S2	1066	10 (0+10)	None	Flare production
Schools	SC	62	19 (19+0)	1	Reading score
Servo	SE	167	4 (0+4)	None	Rise time of a servo
Stock	SP	950	9 (9+0)	None	Stock price
Television	TV	40	4 (4+0)	None	People per TV
Usnews	US	1269	31 (31+0)	7624	Rate of Ph.D.'s
Village	VL	766	32 (29+3)	3986	Number of sheep
WeatherAnkara	WA	1609	9 (9+0)	None	Mean temperature
WeatherIzmir	WI	1461	9 (9+0)	None	Mean temperature

Table 5.1. Properties of the data sets used in the experiments. L: Linear, C: Categorical.

respectively. Although these results indicate the high predictive power of the newly developed regression methods, RSBFP and RSBF, mean value of relative errors on 29 data sets can also be used as an indicator of predictive power for a regression method. RSBFP, having a mean relative error of 0.740, becomes the most predictive regression method. On the other hand, RSBF becomes the third most predictive regression method by having a mean relative error of 0.805 which is very close to the DART's mean relative error, 0.789.

For any regression method, the distribution of the relative errors for different data sets is also important [39]. In Table 5.2, the standard deviation of these relative errors for each different regression method were computed. RSBFP and RSBF achieved the lowest standard deviation values, 0.316 and 0.389. This shows the domain independent characteristics of our methods. They can be applied on many real life databases regardless of any prior knowledge about those domains.

The last column of Table 5.2 shows the standard deviation of the relative errors achieved by the regression methods for each data set. These standard deviation values are used to determine a small portion of data sets to be used for further comparison of regression methods in terms of robustness to irrelevant features, missing feature values and target noise. We have chosen six data sets (AB, AU, HO, SC, WA and WI) having the minimum standard deviation values, since it would be suitable to compare the regression methods, in terms of irrelevant features, missing feature values and target noise, for data sets having initially similar relative error values. These standard deviation values are also typed in boldface in Table 5.2.

5.5 Robustness to Irrelevant Features

The predictive performance of regression methods on selected data sets (AB, AU, HO, SC, WA and WI) by adding new irrelevant features are shown in Figure 5.1. The performance of RSBFP is not effected from new irrelevant features in all data sets. On the other hand, RSBF's performance degrades only on SC data set. Besides RSBFP and RSBF, the regression methods

Data Set	RSBFP	RSBF	KNN	RULE	MARS	DART	StdDev
AB	0.729	0.678	**0.661**	0.899	0.683	0.678	**0.082**
AP	0.550	**0.532**	0.612	0.744	0.720	0.546	0.085
AU	0.489	0.413	**0.321**	0.451	0.333	0.346	**0.063**
BA	0.768	0.570	**0.443**	0.666	0.493	0.508	0.111
BU	**0.678**	0.732	0.961	0.946	0.947	0.896	0.113
CH	0.781	0.606	0.944	0.678	0.735	**0.510**	0.618
CN	**1.429**	1.469	1.642	6.307	5.110	1.695	1.989
CO	0.514	1.554	0.764	0.290	1.854	**0.252**	0.137
ED	0.668	0.461	0.654	**0.218**	0.359	0.410	0.159
EL	**1.003**	1.020	1.194	1.528	1.066	1.118	0.179
FA	0.725	**0.177**	0.785	0.820	0.305	0.638	0.246
FC	0.578	0.638	0.697	0.355	**0.214**	0.415	0.169
FF	1.016	1.013	1.201	1.558	**1.012**	1.077	0.196
HO	0.698	0.589	0.600	0.641	0.526	**0.522**	**0.062**
HR	0.890	**0.707**	0.907	0.890	0.769	0.986	0.093
NE	0.969	0.938	1.034	1.217	0.928	**0.873**	0.114
NT	**0.976**	0.977	1.232	1.250	1.012	1.112	0.111
PL	0.887	0.444	0.475	0.477	**0.404**	0.432	0.166
PV	0.921	0.715	0.796	0.916	1.251	**0.691**	0.187
RE	**0.997**	1.001	1.062	1.352	1.045	1.189	0.126
S2	**1.434**	**1.434**	2.307	1.792	1.556	1.695	0.300
SC	0.376	**0.175**	0.388	0.341	0.223	0.352	**0.081**
SE	0.868	0.868	0.619	**0.229**	0.432	0.337	0.248
SP	1.416	1.101	**0.599**	0.906	0.781	0.754	0.267
TV	1.176	**1.175**	1.895	4.195	7.203	2.690	2.123
US	0.402	**0.385**	0.480	0.550	0.412	0.623	0.087
VL	0.940	**0.930**	1.017	1.267	1.138	1.355	0.161
WA	0.255	0.096	0.113	0.116	**0.073**	0.095	**0.060**
WI	0.209	0.071	0.098	0.100	**0.064**	0.082	**0.049**
Mean	0.805	**0.740**	0.845	1.093	1.091	0.789	
StdDev	**0.316**	0.389	0.483	1.246	1.467	0.548	

Table 5.2. Relative errors of regression methods. Best results are typed with bold font.

RULE and MARS are also noted to be robust to irrelevant features.

Table 5.3 shows the comparison of regression methods on all data sets where 30 irrelevant features are added to each of them. RSBFP and RSBF achieve the two lowest mean relative error values. RSBFP is the best in 8 data sets. whereas RSBF is the best in 7 data sets. MARS regression method, having the lowest relative error values in 10 data sets, is also noted to be a successful method.

5.6 Robustness to Missing Feature Values

The predictive performance of regression methods on selected data sets (AB, AU, HO, SC, WA and WI) by increasing missing feature values are shown in Figure 5.2. The performance of RSBFP and RSBF is slightly affected from increasing missing values in all data sets, except WI (RSBFP), AU (RSBF) and HO (RSBF). However, RSBFP and RSBF's response to increasing missing values is not worse than the other methods in WI, AU, and HO. They are comparable to other regression methods. Therefore, our proposed methods can be regarded as robust to missing feature values. From Figure 5.2, it is noted that DART is also a robust method to missing feature values.

Table 5.4 shows the comparison of regression methods on all data sets where 20% of the values of real data sets are removed. RSBF and RSBFP achieve the two lowest mean relative error values. RSBFP is the best in 2 data sets, whereas RSBF is the best in 12 data sets. DART regression method, having the lowest relative error values in 8 data sets, is also noted to be a successful method.

5.7 Robustness to Target Noise

The predictive performance of regression methods on selected data sets (AB, AU, HO, SC, WA and WI) by increasing target noise are shown in Figure 5.3. The performance of RSBFP and RSBF is not affected from increasing noise in

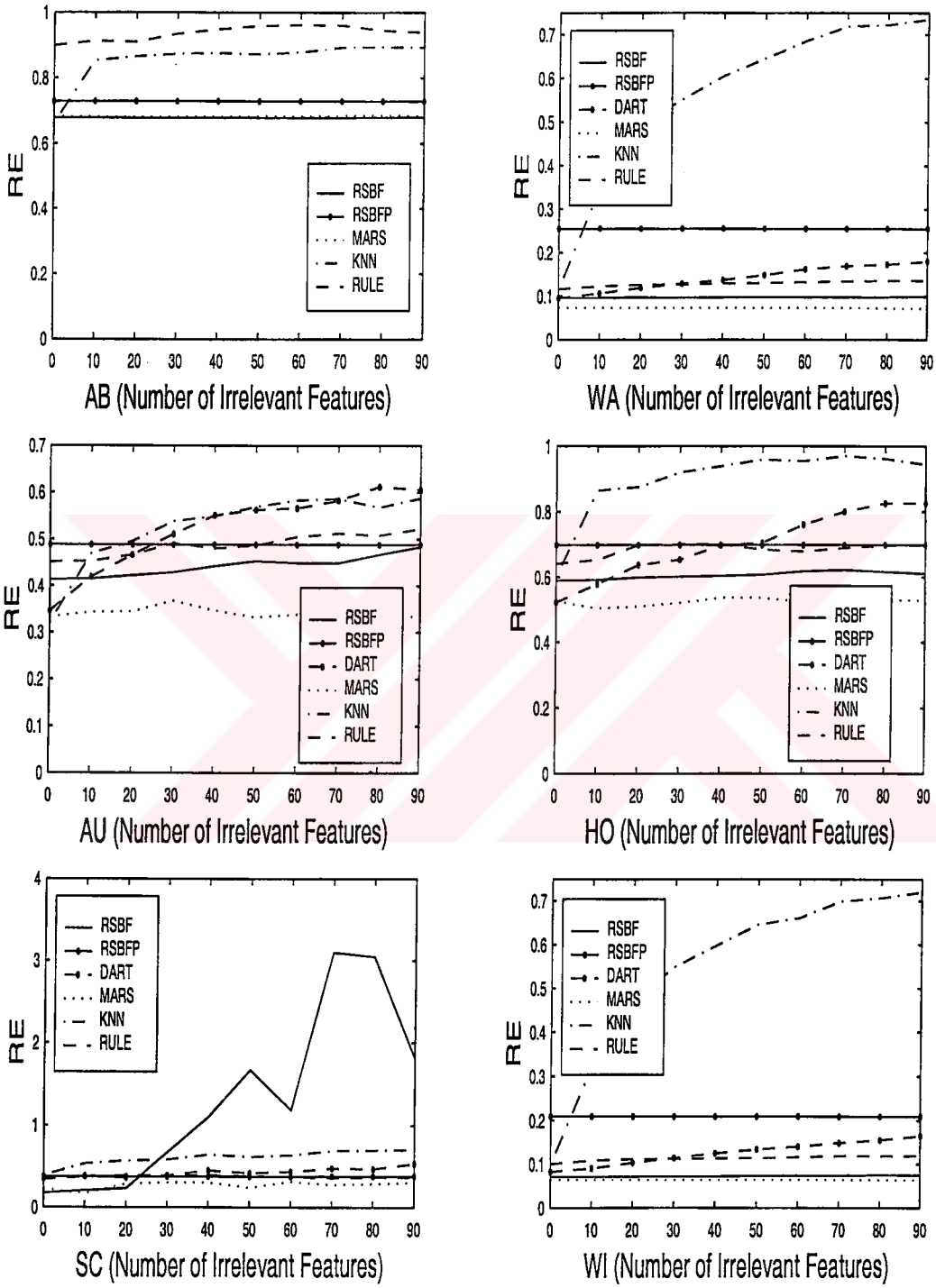

Figure 5.1. Relative errors of methods with increasing irrelevant features

Data Set	RSBFP	RSBF	KNN	RULE	MARS	DART
AB	0.728	**0.677**	0.873	0.934	0.682	*
AP	**0.555**	0.794	1.514	0.723	0.682	0.657
AU	0.488	0.429	0.538	0.491	**0.368**	0.511
BA	0.768	0.603	0.568	0.574	**0.536**	0.628
BU	**0.678**	1.325	0.968	1.073	0.877	0.969
CH	0.781	0.676	1.107	0.753	**0.613**	0.668
CN	**1.425**	2.119	2.854	1.794	4.126	1.662
CO	0.514	1.111	1.162	**0.284**	2.195	0.306
ED	0.668	0.461	0.802	**0.268**	0.404	0.573
EL	**1.006**	1.010	1.037	1.367	1.134	1.236
FA	0.725	**0.204**	1.026	1.039	0.249	0.877
FC	0.578	0.694	0.917	0.456	**0.247**	0.420
FF	**1.030**	1.096	1.063	1.513	1.777	1.430
HO	0.698	0.601	0.920	0.701	**0.521**	0.653
HR	0.890	**0.800**	0.932	1.049	0.847	1.165
NE	0.969	0.938	1.076	1.284	**0.916**	*
NT	**1.000**	1.070	1.079	1.484	1.370	1.156
PL	0.887	0.450	0.961	0.575	**0.407**	0.734
PV	0.966	**0.838**	0.855	0.934	1.005	1.013
RE	**0.998**	1.014	1.045	1.380	1.042	1.311
S2	1.433	**1.429**	1.454	1.765	1.629	1.490
SC	0.376	0.672	0.582	0.386	**0.305**	0.391
SE	0.926	1.036	0.835	**0.471**	0.798	0.641
SP	1.416	1.104	1.188	0.914	0.817	**0.756**
TV	**1.220**	2.222	3.241	5.572	5.614	2.709
US	0.402	**0.385**	0.757	0.557	0.394	0.906
VL	0.939	**0.930**	1.050	1.454	1.257	1.307
WA	0.255	0.097	0.552	0.127	**0.073**	0.129
WI	0.209	0.072	0.550	0.112	**0.064**	0.114
Mean	**0.811**	0.857	1.086	1.036	1.067	0.909
StdDev	**0.319**	0.489	0.586	0.979	1.164	0.536

Table 5.3. Relative errors of regression methods, where 30 irrelevant features are added to real data sets. If the result is not available due to singular variance/covariance matrix, it is shown with (*). Best results are typed with bold font.

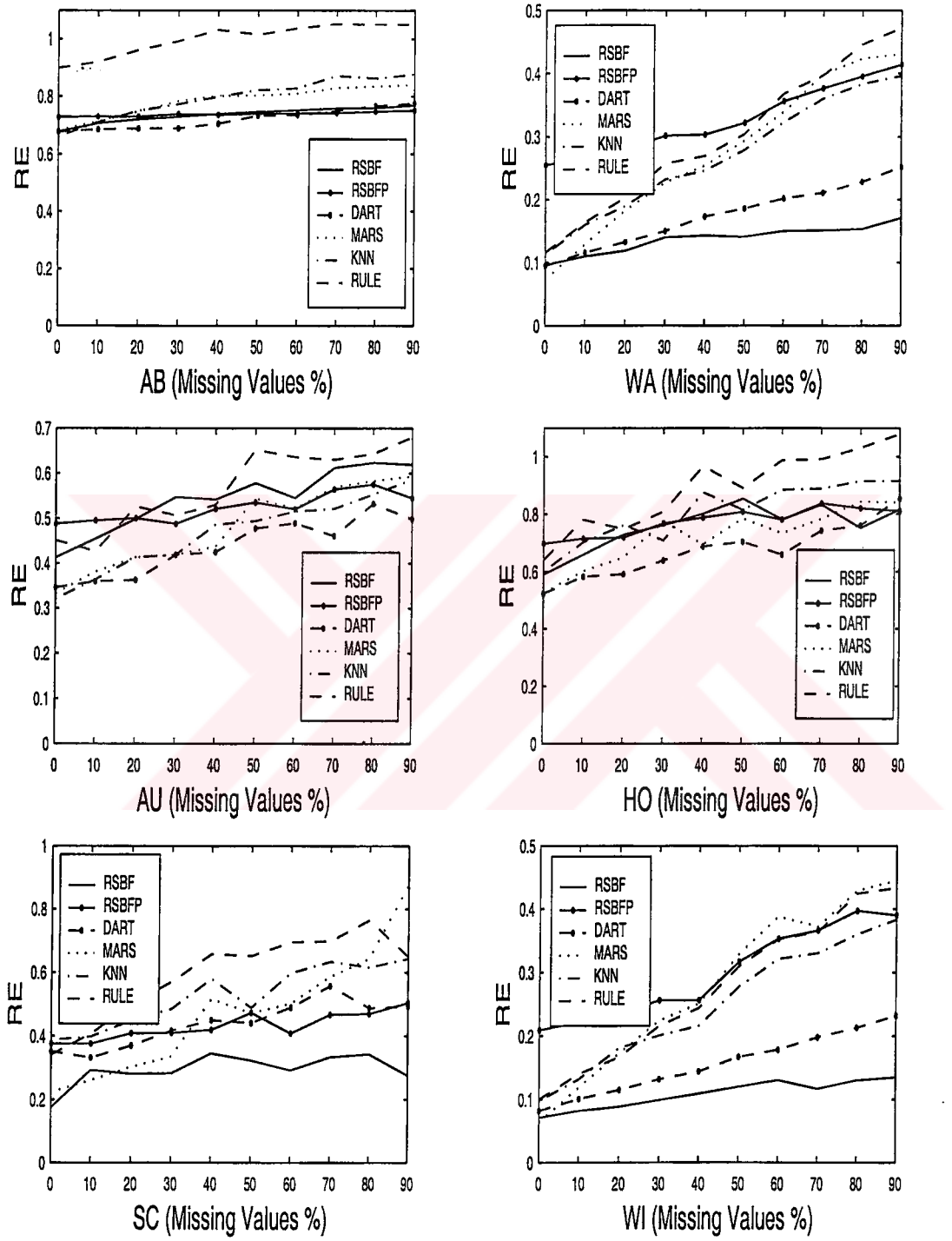

Figure 5.2. Relative errors of methods with increasing missing feature values

Data Set	RSBFP	RSBF	KNN	RULE	MARS	DART
AB	0.729	0.720	0.750	0.961	0.748	**0.688**
AP	0.562	**0.496**	0.726	0.676	0.798	0.546
AU	0.500	0.499	**0.414**	0.526	**0.414**	0.363
BA	0.785	0.714	**0.553**	0.833	0.637	0.576
BU	0.785	**0.682**	0.951	0.878	0.862	1.026
CH	0.746	0.719	0.922	0.832	0.747	**0.608**
CN	1.480	**1.399**	1.856	3.698	3.733	2.377
CO	0.583	0.622	0.942	**0.399**	0.801	0.435
ED	0.685	0.572	0.743	**0.497**	0.595	0.536
EL	**1.005**	1.019	1.097	1.537	1.073	1.191
FA	0.749	0.739	0.849	0.948	**0.731**	0.735
FC	0.570	0.631	0.675	0.543	0.537	**0.401**
FF	1.019	1.034	1.711	1.557	**1.012**	1.347
HO	0.718	0.729	0.761	0.748	0.649	**0.590**
HR	0.899	**0.725**	0.910	1.040	0.836	0.974
NE	0.974	**0.951**	1.072	1.272	0.972	*
NT	1.020	1.006	1.229	1.363	**0.989**	1.222
PL	0.903	0.515	0.733	0.686	0.679	**0.420**
PV	0.920	**0.767**	0.976	1.189	1.026	0.792
RE	0.996	**0,995**	1.059	1.364	1.048	1.229
S2	1.429	1.429	1.851	1.751	1.557	**1.421**
SC	0.409	**0.281**	0.449	0.500	0.303	0.370
SE	0.879	0.879	0.921	0.849	0.746	**0.495**
SP	1.430	1.228	0.744	0.904	0.930	**0.707**
TV	**1.272**	1.408	4.398	3.645	16.502	2.512
US	0.460	**0.388**	0.558	0.620	0.497	0.844
VL	0.949	**0.947**	1.056	1.410	1.090	*
WA	0.265	**0.119**	0.190	0.203	0.183	0.133
WI	0.221	**0.089**	0.181	0.167	0.173	0.115
Mean	0.826	**0.769**	1.010	1.090	1.409	0.843
StdDev	**0.316**	0.340	0.754	0.807	2.917	0.570

Table 5.4. Relative errors of regression methods, where 20% of values of real data sets are removed. If the result is not available due to singular variance/covariance matrix, it is shown with (*). Best results are typed with bold font.

AB, AU, HO and WA until a 30% noise level is reached. Their performance is not affected also in WI until a 20% noise level is reached. In cases where the performance of RSBFP and RSBF is affected from increasing noise level, that performance are comparable to other methods. Therefore, our proposed methods can be regarded as robust to target noise.

Table 5.5 shows the comparison of regression methods on all data sets where 20% target noise is added to the data sets. RSBF and RSBFP achieve the two lowest mean relative error values. RSBFP is the best in 10 data sets, whereas RSBF is the best in 16 data sets. The other methods perform far below RSBFP and RSBF in the existence of 20% target noise.

5.8 Computation Times

The computation times of the regression methods are measured in terms of training and querying times. Table 5.6 shows that RSBF and RSBFP are very fast with respect to other eager methods, in the training phase. Since kNN is a lazy regression method, it does not do much work in the training phase. Hence, RSBFP and RSBF are not faster than kNN in the training phase. In the querying phase, RSBFP and RSBF are very fast with respect to kNN and RULE, whereas they have very close querying times when compared to MARS and DART. Table 5.7 shows the querying times of the regression methods.

When comparing a lazy regression method to an eager one in terms of computational complexity, it will be more fair to compare one's training time to the other's querying time. In this aspect, the querying times of kNN are noted to be much more longer than the training times of RSBFP and RSBF. So it can be easily stated that the proposed regression methods, RSBFP and RSBF, are not only best in terms of predictive accuracy, but also best in terms of computational complexity.

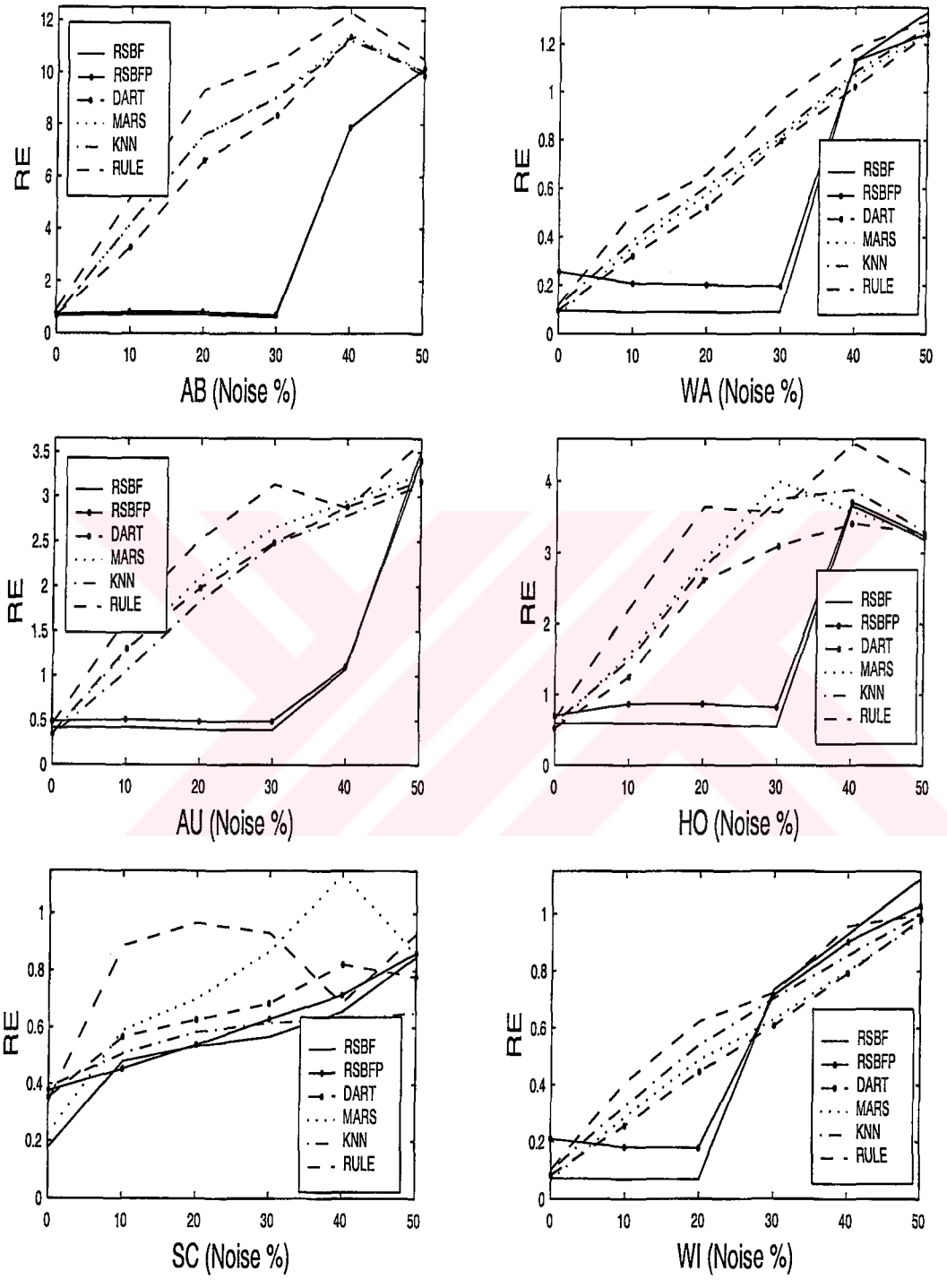

Figure 5.3. Relative errors of methods with increasing target noise

Data Set	RSBFP	RSBF	KNN	RULE	MARS	DART
AB	0.819	**0.726**	7.592	9.301	7.602	6.603
AP	0.952	0.906	0.807	1.122	0.856	**0.785**
AU	0.488	**0.398**	1.832	2.531	2.107	1.981
BA	0.813	0.675	**0.457**	0.712	0.537	0.556
BU	**0.597**	0.935	12.660	12.920	13.300	10.670
CH	0.815	0.720	0.930	0.782	0.745	**0.636**
CN	**1.516**	1.702	1.676	3.102	5.874	2.040
CO	**0.468**	0.834	8.283	11.237	9.393	6.127
ED	0.653	**0.430**	2.166	2.384	2.164	2.276
EL	**0.978**	0.995	1.465	1.899	1.148	1.431
FA	0.684	**0.170**	2.525	3.208	2.447	2.058
FC	0.544	0.653	0.710	0.528	0.501	**0.387**
FF	**1.030**	1.036	2.394	3.247	1.710	2.089
HO	0.865	**0.575**	2.801	3.635	2.893	2.611
HR	0.863	**0.754**	7.853	11.530	10.290	6.115
NE	0.986	**0.947**	38.840	42.320	37.660	31.540
NT	0.951	**0.909**	1.403	2.220	1.037	1.196
PL	0.852	**0.411**	5.492	5.777	4.921	5.107
PV	0.829	**0,692**	9.429	9.456	4.213	6.038
RE	**0.952**	0.958	6.597	10.33	6.759	7.108
S2	**2.366**	**2.366**	73.890	77.210	70.900	71.400
SC	0.538	**0.533**	0.583	0.968	0.700	0.627
SE	**0.697**	**0.697**	21.290	27.770	22.010	21.720
SP	1.183	**0.646**	1.921	3.887	1.966	1.871
TV	**1.468**	1.747	2.087	4.569	7.267	2.671
US	0.643	0.634	0.636	0.865	**0.541**	0.764
VL	**0.973**	0.976	1.030	1.513	0.977	1.518
WA	0.200	**0.088**	0.606	0.657	0.573	0.521
WI	0.176	**0.067**	0.540	0.623	0.487	0.444
Mean	0.858	**0.799**	7.534	8.839	7.640	6.859
StdDev	**0.411**	0.474	14.779	15.656	14.236	13.889

Table 5.5. Relative errors of regression methods, where 20% target noise is added to real data sets. Best results are typed with bold font.

Data Set	RSBFP	RSBF	KNN	RULE	MARS	DART
AB	148	211.5	**8.9**	3219	10270	477775
AP	1.1	2	**0**	90.8	159.2	62
AU	8.9	12.5	**0.6**	248.9	570.5	1890.1
BA	19	35.3	**0**	181.8	915.1	3171.1
BU	10.5	45.5	**0**	67.1	761.7	794.4
CH	4.1	6.3	**0**	52.7	575.3	286
CN	8	17.9	**0.1**	108.6	475.3	481
CO	15.9	34.1	**0.5**	148.2	1274.3	717.6
ED	278.2	691.1	**13.5**	862.8	10143.9	27266
EL	8.1	13.3	**0.2**	69.5	407.5	1017
FA	15.8	36.4	**0**	161.1	985	1773.9
FC	2.1	4.2	**0**	47.8	240.2	201.4
FF	1.1	1	**0**	34.1	99.5	45.9
HO	21.2	36.3	**1**	264.9	1413.9	8119.7
HR	8.2	19	**0**	57.5	616.3	893.9
NE	130.5	189.9	**7.4**	3493	5709.9	87815
NT	**0**	0.2	**0**	30.6	69.3	18.9
PL	10	13.7	**0.2**	175.3	824.8	10024.4
PV	1	2.1	**0**	40.9	127.3	44
RE	52	104.3	**3**	196	2744.6	33044.6
S2	36.1	40.8	**3.5**	108.8	667.2	971.4
SC	3	8.1	**0**	45.3	260.8	84.4
SE	1.8	2.2	**0**	37	116.4	83.4
SP	28.5	57.1	**1.4**	365.1	2281.4	17346.4
TV	**0**	0.2	**0**	30.9	31.1	3.1
US	136.4	245	**7.4**	2547.1	8435.2	168169
VL	74.6	136.8	**4.4**	513.6	3597.8	23405
WA	130.8	181.2	**10.4**	1288.6	3201.6	44525.6
WI	114.6	159.6	**3.2**	1085.5	2662.4	36524.1
Mean	43.776	79.57	**2.265**	536.98	2056.46	32639.8

Table 5.6. Training time durations of methods in milliseconds. Best results are typed with bold font.

Data Set	RSBFP	RSBF	KNN	RULE	MARS	DART
AB	23.3	21.3	6547	14433	7.9	**6.1**
AP	1	1	3.4	141.7	**0**	**0**
AU	3	2.1	64.5	462.2	**0**	**0**
BA	2.1	2.2	54.6	244.8	**0**	**0**
BU	**0**	**0**	11.6	32.1	**0**	**0**
CH	1	1	11.6	87.3	**0**	**0**
CN	1	0.3	8.4	98.4	**0**	0.1
CO	1.1	1	38.2	40.3	1	**0**
ED	9	8.1	2699.7	312.3	2.7	**1.7**
EL	1.6	1.6	21	117.5	**0**	**0**
FA	2	1.4	33.1	96.4	**0**	**0**
FC	1.1	1.2	7.9	48.8	**0**	**0**
FF	0.9	0.1	2	45.4	**0**	**0**
HO	3.2	3	107.8	410.5	**0**	**0**
HR	0.6	0.6	13.3	43	**0**	**0**
NE	14	12.6	3399.4	11327	4.7	**1.75**
NT	1	**0**	1.9	30.8	**0**	**0**
PL	16.7	9.5	571.9	2192.7	**0.2**	1.2
PV	**0**	**0**	2.2	37.1	**0**	**0**
RE	3	3.2	265.6	627.2	**0**	1
S2	4	4.2	407.8	223.6	0.4	**0**
SC	**0**	0.3	2	27.8	3.7	**0**
SE	0.1	0.1	4.2	49.1	**0**	**0**
SP	6.2	5.4	303.2	1090.9	0.1	**0**
TV	**0**	**0**	**0**	24	**0**	**0**
US	8	7.3	1383.2	1877.3	7	**2**
VL	6	5.8	439	1118.2	0.3	**0**
WA	2.1	**0.2**	1794.6	6606.7	2.1	1.1
WI	**0**	**0**	1462.4	5267.5	2	1.1
Mean	3.862	3.224	677.983	1624.60	1.107	**0.553**

Table 5.7. Querying time durations of methods in milliseconds. Best results are typed with bold font.

Chapter 6

Conclusion and Future Work

In this thesis, we have presented two new regression methods called RSBFP and RSBF. They are eager, parametric, linear and context-sensitive methods based on feature projections. They achieve higher accuracy results and faster execution times when compared to important eager and lazy methods of both machine learning and statistics community. The common drawback of RSBFP and RSBF is the lack of handling interactions among predictor features. RSBFP does not allow the regression lines of the induced model to include more than one feature. Although this limitation is handled in RSBF, it does not use all the possible combinations of the predictor features because of the infeasibility of the approach. Therefore, we used a relevancy heuristics to determine the best subset of the predictor features.

Besides these drawbacks, RSBFP and RSBF are powerful in the existence of missing feature values, target noise and irrelevant features. These three factors heavily exist in real life databases, and it is important for a learning method to give promising results in the presence of those factors. The robustness to irrelevant features was our main motivation to induce RSBFP and RSBF. RSBFP method orders the features in terms of their predictive power, that is, in terms of their relevancy. On the other hand, RSBF produces a decision list of multiple linear regression lines. The regression line on the top of the list is claimed to include the most predictive, relevant predictor features. Furthermore, our methods are context-sensitive since the predictive power of

any categorical feature may vary among its values.

The computational complexity of a learning method is also important in machine learning. The success of a learning method is not only measured by its predictive power, but also by its training and querying time requirements. RSBFP and RSBF are very fast both in training and querying phases. Although kNN is the fastest regression method in training phase, it suffers in querying phase and also gives higher prediction errors when compared to RSBFP and RSBF. On the other hand, DART is as successful as RSBFP and RSBF in making correct predictions. But its training phase takes a lot of time. In fact, it is the slowest method in training phase. RULE and MARS are the weakest methods since they neither make promising predictions, nor have small training or querying times. The proposed methods, RSBFP and RSBF, can be regarded as the best regression methods since they both make better predictions and execute faster than the other approaches.

The advantages and limitations of RSBFP and RSBF are described in detail in previous chapters. Future work can be directed to overcome these limitations, such as allowing regression lines include interaction terms, and to incorporate different properties to our methods, such as incorporating an expert knowledge or using a different relevancy heuristic in the elimination process of irrelevant features.

Machine learning methods were developed to extract knowledge without an expert, since databases come from a large number of domains. However, when we deal with data belonging to a single domain where a domain expert is available, the expert knowledge can be incorporated to our methods to increase the accuracy significantly.

As a final word, *regression by selecting best feature(s)* is a successful technique in regression. RSBFP and RSBF can compete with the famous and successful methods of machine learning and statistics community. Some important properties of our methods, which are missing in many other methods, such as robustness, domain independence and handling missing feature values naturally enable them to be invaluable tools for knowledge extraction and prediction systems.

Bibliography

[1] Adeli, H., Knowledge Engineering Vol.1, *McGraw Hill*, 1990.

[2] Aha D. W., Salzberg, S. L., Learning to Catch: Applying Nearest Neighbor Algorithms to Dynamic Control Tasks, *Selecting Models from Data: Artificial Intelligence and Statistics IV.*, New York, NY: Springer-Verlag. 1994.

[3] Aha, D., Kibler, D., and Albert, M., Instance-based Learning Algorithms. *Machine Learning*, 6, 37-66, 1991.

[4] Akkuş, A. and Güvenir,H. A., k Nearest Neighbor Classification on Feature Projections, In *Proceedings of the* 13^{th} *International Conference on Machine Learning.* Lorenza Saitta (Ed.), Bari, Italy: Morgan Kaufmann. 12–19, 1996.

[5] Ari, B., Clustered Linear Regression, *Senior Project*, Bilkent University, May 2000.

[6] Atkenson, G. C., Moore, A. W., Schaal, S., Locally Weighted Learning. *Artificial Intelligence Review*, Vol.11, No.1/5, 11-73, February 1997.

[7] Aydin, T., Güvenir, H. A., Regression by Selecting Appropriate Features, *Proceedings of TAINN'2000*, Izmir, June 21-23, 2000, 73-82.

[8] Blake, C., Keogh, E. and Merz, C. J., UCI Repository of Machine Learning Databases, [http://www.ics.uci.edu/mlearn/MLRepository.html], Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[9] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. Classification and Regression Trees. *Wadsworth*, Belmont, Calif., USA, 1984.

[10] Cost, S., Salzberg, S., A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features, *Machine Learning*, 10(1), 57-78, 1993.

[11] Dasarathy, B. V. (Ed.), Nearest Neighbor(NN) Norms: NN Pattern Classification Techniques, *CA:IEEE Computer Society Press*, Los Alamitos. 1991.

[12] De Boor, C., A Practical Guide to Splines, *Springer*, New York, 1978.

[13] Domingos, P., Occam's Two Razors, The Sharp and the Blunt, *Proceedings of KDD'98*, 1998.

[14] Duda, R. and Hart,P.E., Pattern Classification and Scene Analysis, *Wiley*, 1973.

[15] Friedman, J. H., Multivariate Adaptive Regression Splines, *The Annals of Statistics*, 19 (1) 1-141, 1991.

[16] Friedman, J. H., Local Learning Based on Recursive Covering, [ftp:// stat.stanford.edu/pub/friedman/dart.ps.Z], 1996.

[17] Friedman, J. H., On Bias, Variance, 0/1-loss and the Curse of Dimensionality, *Data Mining and Knowledge Discovery*, 1, 55, 1997.

[18] Graybill, F.A., Iyer, H.K., Burdick, R.K., Applied Statistics - A First Course in Inference, *Prentice Hall*, 5, 105, 1998.

[19] Güvenir, H. A., Şirin, İ., Classification by Feature Partitioning, *Machine Learning*, Vol.23, No:1, 47-67, 1996.

[20] Güvenir, H. A. and Uysal, İ., Bilkent Function Approximation Repository, [http://funapp.cs.bilkent.edu.tr/], 1999.

[21] Holte, R. C., Very Simple Classification Rules Perform Well on Most Commonly Used Datasets, *Machine Learning*, 11,63-91, 1993.

[22] Karalic, A., Employing Linear Regression in Regression Tree Leaves, *In Proceedings of ECAI'92 (European Congress on Artificial Intelligence)*, Vienna, Austria, Bernd Newmann (Ed.),440-441, 1992.

[23] Karalic, A., Linear Regression in Regression Tree Leaves, *In Proceedings of ISSEK'92 (International School for Synthesis of Expert Knowledge) Workshop*, Bled, Slovenia, 1992.

[24] Kaufman, L. and Rousseeuw, P.J., Finding Groups in Data - An Introduction to Cluster Analysis, *Wiley Series in Probability and Mathematical Statistics*, 1990.

[25] Kibler, D., Aha D. W., Albert, M. K., Instance-based Prediction of Real-valued Attributes, *Comput. Intell.*, 5, 51-57, 1989.

[26] Kohavi, R., Langley, P., Yun, Y., The Utility of Feature Weighting in Nearest-Neighbor Algorithms, *ECML-97*

[27] Lock, R. H. and Arnold, T., Datasets and Stories: Introduction and Guidelines, *Journal of Statistics Education* [Online], 1 (1), [http://www.amstat.org/publications/jse/v1n1/datasets.html], 1993.

[28] Mitchell, T.M., Machine Learning, *McGraw Hill*, 1997.

[29] Peterson, M., Lundberg, D., Ek, A., Knowledge Engineering System Design in Diffuse Domains, *Studentlitteratur, Chartwell-Bratt*, 1990.

[30] Press, W. H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., Numerical Recipes in C, *Cambridge University Press*, 1988.

[31] Quinlan, J. R., Induction of Decision Trees, *Machine Learning*, 1, 81-106, 1986.

[32] Quinlan, J. R., Unknown Attribute Values in Induction, In A. Segre (Ed.), In *Proceedings of the 16th International Workshop on Machine Learning*, 164-168, San Mateo, CA:Morgan Kaufmann, 1989.

[33] Quinlan, J. R., Learning with Continuous Classes, *In Proceedings AI 92 (Adams and Sterling Eds.)*, 343-348, Singapore, 1992.

[34] Quinlan, J. R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann, California, 1993.

[35] SPSS Sample Data Sets, [ftp://ftp.spss.com/pub/spss/sample/datasets/], 1999.

[36] Uysal, İ. and Güvenir, H. A. An Overview of Regression Techniques for Knowledge Discovery, *Knowledge Engineering Review*, Cambridge University Press, Vol.14, No.4, 319-340, 1999.

[37] Uysal, İ. and Güvenir, H. A., Regression by Feature Projections, *Proceedings of Third Eurepean Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'99)*, Springer-Verlag, LNAI 1704, Jan. M. Zytkow and Jan Rauch (Eds.), Prague, Czech Republic, 568-573, 1999.

[38] Uysal, İ, Güvenir, H. A., Regression on Feature Projections, *Knowledge-Based Systems*, Vol.13, No:4, 207-214, 2000.

[39] Uysal, İ., Regression by Partitioning Feature Projections, *M.S.Thesis*, Computer Engineering, Bilkent University, January, 2000.

[40] Weiss, S. and Indurkhya, N., Optimized Rule Induction. *IEEE Expert*, 8(6), 61-69, 1993.

[41] Weiss, S. and Indurkhya, N., Rule-based Regression, *In Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pp1072-1078, 1993.

[42] Weiss, S. and Indurkhya, N., Rule-based Machine Learning Methods for Functional Prediction, *Journal of Artificial Intelligence Research*, 3 383-403, 1995.

[43] Weiss, S. and Indurkhya, N., Predictive Data Mining: A Practical Guide, *Morgan Kaufmann*, San Francisco, 1998.