EMPRR: A High-dimensional EM-Based Piecewise Regression Algorithm

by

Manimozhiyan Arumugam

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Stephen D. Scott

Lincoln, Nebraska

August, 2003

EMPRR: A High-dimensional EM-Based Piecewise Regression Algorithm

Manimozhiyan Arumugam, M.S.
University of Nebraska, 2003

Adviser: Stephen D. Scott

We propose a novel general piecewise surface regression model that allows for arbitrary functions to be used in each piece, and arbitrary boundary surfaces between pieces. We also give an EM-based algorithm for this model, `EMPRR`, that scales to high dimensions. We compare `EMPRR`'s performance with those of model trees and functional trees, two regression tree learning methods, on synthetic piecewise data and benchmark data sets. Our results show that `EMPRR` outperforms the other two methods on the synthetic data sets and performs competitively on the benchmark data sets while generating accurate and much more compact models.

Acknowledgements

I would like to thank my advisor Prof. Stephen Scott first of all for making me a computer scientist. He took a risk of hiring a student with biotechnology background as a research assistant in computer science and I hope I have justified his decision in the past two and a half years. He has been more a friend to me than an advisor, making sure that things are fine for me during my stay in Lincoln. I would also like to thank him for his invaluable guidance throughout my research and when I was writing research papers and this thesis. Finally, I am grateful to him, for his reference played a major role in my job offers.

I thank Prof. Jitender Deogun, Prof. Ashok Samal and Prof. Andrew Benson for being on my committee and for their valuable input on my work and this report.

Thanks are due to Dr. Deogun and Prof. Byrav Ramamurthy for providing me references in my job search.

I also thank Dr. João Gama for providing the implementation of functional trees and helpful instructions on running the implementation, and Prof. Tapabrata Maiti for his helpful discussions during the early development of this work.

I thank my friends and acquaintances in Lincoln, who made my stay a very pleasant one.

I thank my parents for bringing me to this world and making me what I am today.

I thank God Almighty for his blessings throughout my life, through the good and bad times.

My work was funded in part by NSF grants CCR-9877080 (with matching funds from UNL-CCIS), CCR-0092761, and EPS-0091900, and a grant from the University of Nebraska Foundation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Piecewise functions have many applications in science. From signal and control theory where they are used in controllers that change the flow rate from $3ft^3/sec$ to $5ft^3/sec$ at a certain time point, to biology where the bacterial growth curve is characterized typically by four different growth phases, piecewise functions occur as natural and man-made phenomena. Modeling such phenomena requires nonlinear n-way piecewise regression, which the currently available methods for piecewise regression do not provide. Some of them (McGee & Carleton, 1970; Quinlan, 1992; Wang & Witten, 1997; Yu et al., 2001; Gama, 2002) are limited to linear regression, some are limited to only two regimes (Quandt, 1958), and some need prior information about the underlying piecewise hypothesis (Tishler & Zang, 1981).

This thesis develops a model for general piecewise surface regression. This model allows for arbitrary functions to be used in each regime, can enforce continuities or allow discontinuities between regimes, and allows for arbitrary boundary surfaces between regimes. However, the large number of parameters needed to represent a piecewise surface in this model can grow quickly, making direct nonlinear optimization difficult. Thus we also present *EM-based Piecewise Regressor Algorithm* (`EMPRR`), an EM-based algorithm for our model that scales more easily to high dimensions.

We perform the regression by minimizing the squared error between a hypothesis (model) and the sample data. We can use any nonlinear optimization technique to perform this optimization step. In our study, we use the nonlinear optimization technique of Marquardt (1963). This model does not require any prior information about the underlying piecewise hypothesis beyond knowledge of the number of regimes and the general form of the function in each regime (e.g. cubic, exponential).

Under a few assumptions, regression to a piecewise function with one independent variable is straightforward if we use dynamic programming. But it is unlikely that dynamic programming can be used in the case of multiple independent variables since the boundaries of the regimes become higher-dimensional. Thus we developed a heuristic model that divides the regression problem into two independent, simpler regression problems. This model performs two independent regression steps, each of which optimize a distinct set of parameters, switching back and forth between them until it converges. This is analogous to the Expectation Maximization (EM) algorithm (Dempster et al., 1977) commonly used in machine learning, which alternates between two steps, the Expectation step (E-step) that estimates the expected values of the hidden parameters from the current hypothesis and the Maximization step (M-step) that finds a new hypothesis that maximizes the expectation from the E-step.

In this thesis, we evaluate `EMPRR` by devising experiments to test its performance. We generated synthetic piecewise data using different sets of parameters and tried to fit them using `EMPRR`, and also using `M5′` (Wang & Witten, 1997) and functional trees (Gama, 2002), two regression tree-learning methods mentioned in Chapter 3. Results show that `EMPRR` outperforms the other two methods in the data sets that we used, by generating accurate and compact models. We also used a set of benchmark regression problems to compare the performance of `EMPRR` with `M5′` and functional trees. Our results show that `EMPRR` is competitive against the two methods and constructs a model with a much more compact representation. Being a regression technique however, it fits the data so well that it performs slightly worse on unseen data, but the differences are not statistically significant most of the time.

The rest of the thesis is organized as follows. Chapter 2 provides the necessary background information on piecewise functions and the theory behind data modeling. Section 2.2 introduces the concept of data modeling. Section 2.3 explains various methods to measure correlation between variables during data modeling. Chapter 3 lists the currently available methods from the literature and their shortcomings.

Chapter 4 explains our piecewise surface regression algorithm. Section 4.1 explains the steps in evaluating piecewise function with one independent variable. Section 4.2 explains the piecewise surface regression model and the steps involved in EMPRR. Section 4.3 discusses the variations to the basic algorithm and Section 4.4 talks about our implementation.

In Chapter 5 we present some interesting results from comparing EMPRR with M5$'$ and functional trees. Section 5.1 talks about the synthetic data sets that we generated and gives results obtained from running EMPRR on these data sets. Section 5.2 talks about the set of benchmark regression data sets and summarizes results from analyzing data sets. Finally, Chapter 6 draws conclusions and also briefs work in progress and future work.

Chapter 2

Background

This chapter provides necessary background information about data modeling techniques. The topics discussed here are either directly used in our algorithm or relevant to the understanding of the implementation of the algorithm. But the reader can skip Section 2.2 onwards without loss of continuity, if understanding the implementation is not desired. Most statistical books will provide the information given here. A portion of the following discussion given here has been adopted from Press et al. (1992), which provides a compact summary of the theory behind data modeling. Section 2.1 introduces piecewise functions and Section 2.2 gives details that are necessary to understand data modeling. Section 2.3 talks about measuring the association between two variables.

2.1 Piecewise Functions

Let X and Y be two sets. A function f from X to Y is a relation between X and Y such that for each $x \in X$ there is one and only one associated $y \in Y$. The set X is known as the *domain* of the function and Y is known as the *range* of the function. A function is generally denoted by $y = f(x)$, indicating a relation $\{x, f(x)\}$. x can be viewed as an *input* to function f and y as the *output* of function f.

A piecewise function is a function whose definition of the relation between X and

Y depends on the value of the input. For example, the following function

$$f(x) = \begin{cases} 4x^2 + 2x + 5 & \text{if } x < 8 \\ \sin x & \text{if } 8 \leq x < 15 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

uses a different relation to evaluate $y = f(x)$ for $x = 1$, $x = 10$ and $x = 20$. A *transition point* (also referred to as a *join point* or *knot* in the literature) of a piecewise function is defined as the point at which it makes a transition from one form to another. For example, for the function defined by Equation (2.1), the transition points are 8 and 15. A piecewise function may be continuous or discontinuous depending on whether or not the functions flanking the transition point meet at the transition point.

Piecewise functions occur commonly in nature. For example, the bacterial growth curve contains four different phases among which three are distinct and well-defined. They are exponential phase, stationary phase and death phase. Each phase is characterized by a different equation. The three-phase bacterial growth curve can be defined by

$$S(t) = \begin{cases} S_0 e^{\eta t} & \text{if } 0 \leq t < t_s \\ S_s & \text{if } t_s \leq t < t_d \\ S_s + M(t - t_d) & \text{if } t_d \leq t \end{cases}, \tag{2.2}$$

where

S_0 is the initial cell number (number of cells) in the population,

S_s is the cell number during stationary phase,

t_s is the time at which stationary phase starts,

t_d is the time at which death phase starts,

η is the exponential growth rate constant,

M is the rate at which cells disappear during death phase.

Modeling the bacterial growth curve and estimating the growth parameters is a common procedure in biology.

Modeling piecewise functions is more complex than modeling ordinary functions since it does not have a unique representation of the relation between the domain and the range.

2.2 Modeling of Data

Fitting data to a model that depends on adjustable parameters is known as 'modeling of data'. This is conventionally done by defining a *figure of merit function* that relates the data to the model. The properties of the relation and the relation itself are then used in fitting the data as close to the model as possible. The parameters that define the model closest to the data in terms of the figure of merit function are known as best-fit parameters. Traditional figure of merit functions are inversely proportional to the quality of the fit, i.e., the lower the value of the function, the better the fit.

But data modeling does not necessarily stop with finding the best-fit parameters. Measured data are generally subject to measurement errors. Thus typical data may not exactly fit the model that is being used, even when that model is correct. Moreover, if the class of model is incorrectly chosen, there may not exist a set of parameters to even fit the data very well.

2.2.1 Least Squares and χ-Square Fitting

Suppose that we are fitting N data points (x_i, y_i), $i = 1, \ldots, N$, to a model that has M adjustable parameters a_j, $j = 1, \ldots, M$. The model predicts a functional relationship between the measured *independent variable* (or *independent attribute*) x and *dependent variable* (or the *target variable* or *dependent attribute*) y as,

$$y(x) = y(x; a_1 \ldots a_M) \ , \tag{2.3}$$

where the right-hand side indicates that y is dependent also on the adjustable parameters. (We use $y(x)$ and $y(x; a)$ interchangeably – the latter only reiterates the definition of the model using a.) We need a measure to minimize so as to get fitted values for a_j's.

For any given data set of x_i's and y_i's, we can say that some parameter sets $a_1 \ldots a_M$ are very likely to have generated them and some are unlikely. To precisely quantify this behavior, we cannot ask "What is the probability that a particular set of fitted parameters $a_1 \ldots a_M$ is correct?" (Press et al., 1992). This question would be appropriate if there is a statistical universe of models from which the model is drawn. On the contrary, there is only one correct model (that generated the data) and a statistical universe of data sets (including the given data set) that are drawn from it. A valid question would then be, "Given a particular set of parameters, what is the probability that this data set could have occurred (Press et al., 1992), allowing a deviation of some fixed Δy on each data point?" If this probability is infinitesimally small, then we can say that the set of parameters are unlikely to be right. Conversely, this probability *should not* be very low for the correct choice of parameters. This probability is the *likelihood* of parameters given the data.

Suppose that there is a measurement error associated with each data point y_i and that it is independently random and normally distributed around the true model $y(x)$. Suppose also that the standard deviations σ of these normal distributions are the same for all points. We can calculate the probability P of the data set as the product of the probabilities of the individual data points, using

$$P \propto \prod_{i=1}^{N} \left\{ \exp\left[-\frac{1}{2} \left(\frac{y_i - y(x_i)}{\sigma} \right)^2 \right] \Delta y \right\} \ . \tag{2.4}$$

To obtain the best-fit for the data points, we only need to find the parameters $a_1 \ldots a_M$ that maximize the probability of the data set. Now maximizing Equation (2.4) is equivalent to minimizing the negative of its logarithm,

$$\left[\sum_{i=1}^{N} \frac{[y_i - y(x_i)]^2}{2\sigma^2} \right] - N \log \Delta y \ .$$

which is equivalent to minimizing

$$\sum_{i=1}^{N} [y_i - y(x_i; \hat{a})]^2 \ , \tag{2.5}$$

since N, σ and Δy are all constants. This is known as the *least-squares fit.*

If each data point (x_i, y_i) has its own normal distribution of its measurement error around the true model, with standard deviation σ_i, then Equation (2.4) is modified to reflect this behavior. Thus the maximum likelihood estimate of the model parameters is obtained by minimizing the quantity

$$\chi^2 \equiv \sum_{i=1}^{N} \left(\frac{y_i - y(x_i; \hat{a})}{\sigma_i} \right)^2 \quad , \tag{2.6}$$

called the *chi-square*.

If we are measuring a single variable y as a function of more than one variable — say, a *vector* of variables $\hat{x}$, then our basis functions will be functions of a vector, $X_1(\hat{x}), \ldots, X_M(\hat{x})$. The χ^2 merit function is now

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - y(\hat{x}_i; \hat{a})}{\sigma_i} \right)^2 \quad . \tag{2.7}$$

2.2.2 Linear and Nonlinear Models

To fit a set of data to a straight line, we can use linear regression, which guarantees the best-fit model parameters $a_1 \ldots a_M$ based on the least squares criterion. This can also be generalized to a model that is a linear combination of any M specified functions of $\hat{x}$, like a polynomial of degree $M-1$ or a harmonic series containing sines and cosines of $\hat{x}$. The general form of this kind of model is

$$y(\hat{x}) = \sum_{k=1}^{M} a_k X_k(\hat{x}) \tag{2.8}$$

where $X_1(\hat{x}), \ldots, X_M(\hat{x})$ are arbitrary fixed functions of $\hat{x}$. Using χ^2 as the merit function we can use several different techniques to pick best parameters that minimize χ^2. Two such methods are *solution by normal equations* and *solution by singular value decomposition*. These methods determine the best-fit parameters by minimizing χ^2 in one step. With nonlinear dependencies, however, the minimization must proceed iteratively. Let us take some point $\hat{P}$ as the origin of the coordinate system with coordinates $\hat{a}$. Then χ^2 can be approximated by its Taylor's series

$$\begin{aligned} \chi^2(\hat{a}) &= \chi^2(\hat{P}) + \sum_i \frac{\partial \chi^2(\hat{P})}{\partial a_i} a_i + \frac{1}{2} \sum_{i,j} \frac{\partial^2 \chi^2(\hat{P})}{\partial a_i \partial a_j} a_i a_j + \cdots \\ &\approx \gamma - \hat{d} \cdot \hat{a} + \frac{1}{2} \hat{a} \cdot \hat{D} \cdot \hat{a} \quad , \end{aligned} \tag{2.9}$$

where

$\gamma \equiv \chi^2(\hat{P})$, the value of χ^2 at $\hat{P}$,

$\hat{d} \equiv -\nabla\chi^2(\hat{P})$, the negative gradient vector (w.r.t. $\hat{a}$) of χ^2 evaluated at $\hat{P}$, and

$\hat{D}_{ij} \equiv \frac{\partial^2\chi^2(\hat{P})}{\partial a_i \partial a_j}$, the second order gradient vector of χ^2 evaluated at $\hat{P}$.

The matrix $\hat{D}$ whose components are the second partial derivatives of the function is called the *Hessian matrix* of the function at $\hat{P}$. Near the current point $\hat{a}_{cur}$, we can approximate the value of χ^2 up to second order, as

$$\chi^2(\hat{a}) \approx \chi^2(\hat{a}_{cur}) + (\hat{a} - \hat{a}_{cur}) \cdot \hat{d} + \frac{1}{2}(\hat{a} - \hat{a}_{cur}) \cdot \hat{D} \cdot (\hat{a} - \hat{a}_{cur}) \ , \tag{2.10}$$

so

$$\nabla\chi^2(\hat{a}) \approx \nabla\chi^2(\hat{a}_{cur}) + \hat{D} \cdot (\hat{a} - \hat{a}_{cur}) \ . \tag{2.11}$$

If the approximation in Equation (2.9) is good, using Newton's method we set $\nabla\chi^2(\hat{a}) = 0$ and determine the next iteration point using

$$\hat{a}_{next} = \hat{a}_{cur} - \hat{D}^{-1} \cdot \nabla\chi^2(\hat{a}_{cur}) \ . \tag{2.12}$$

On the contrary, if Equation (2.9) is a poor local approximation to χ^2 at $\hat{a}_{cur}$, we can take a step down the gradient, as in steepest descent methods,

$$\hat{a}_{next} = \hat{a}_{cur} - \text{constant} \times \nabla\chi^2(\hat{a}_{cur}) \ , \tag{2.13}$$

where the constant is chosen such that it does not exhaust the entire downhill direction. To use Equations (2.12) and (2.13), we must be able to compute the gradient of χ^2 at any set of parameters $\hat{a}$. To use Equation (2.12), we also need the Hessian matrix $\hat{D}$ at any $\hat{a}$. The choice between these two equations depends primarily on the availability of the Hessian matrix. In this case, since we know the exact form of χ^2, the Hessian matrix is known to us. Therefore we can use Equation (2.12) whenever we want. The only reason to choose Equation (2.13) will be failure of Equation (2.12) to improve the fit, which tells us that Equation (2.9) is not a good local approximation.

2.2.3 Calculating the Gradient and Hessian Matrix

The model to be fitted is

$$y = y(\hat{x}; \hat{a}) \tag{2.14}$$

and the χ^2 merit function is Equation (2.7). To obtain the parameters $\hat{a}$ at the χ^2 minimum, we need to set the gradient of χ^2 with respect to $\hat{a}$ to zero. This gradient is given by

$$\frac{\partial \chi^2}{\partial a_k} = -2 \sum_{i=1}^{N} \frac{[y_i - y(\hat{x}_i; \hat{a})]}{\sigma_i} \frac{\partial y(\hat{x}_i; \hat{a})}{\partial a_k}, k = 1, 2, \ldots, M \ . \tag{2.15}$$

Taking an additional partial derivative gives

$$\frac{\partial^2 \chi^2}{\partial a_k \partial a_l} = 2 \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[\frac{\partial y(\hat{x}_i; \hat{a})}{\partial a_k} \frac{\partial y(\hat{x}_i; \hat{a})}{\partial a_l} - [y_i - y(\hat{x}_i; \hat{a})] \frac{\partial^2 y(\hat{x}_i; \hat{a})}{\partial a_l \partial a_k} \right] \ . \tag{2.16}$$

If we remove factors of 2 by defining

$$\beta_k \equiv -\frac{1}{2} \frac{\partial \chi^2}{\partial a_k} \text{ and } \alpha_{kl} \equiv \frac{1}{2} \frac{\partial^2 \chi^2}{\partial a_k \partial a_l} \ , \tag{2.17}$$

and making $[\alpha] = \frac{1}{2}\hat{D}$ in Equation (2.12), we can rewrite it as the set of linear equations

$$\sum_{l=1}^{M} \alpha_{kl} \delta a_l = \beta_k \ . \tag{2.18}$$

This set is solved for the increments δa_l, which is the distance to the next approximation in the positive direction. In the context of least-squares, the matrix $[\alpha]$, equal to one-half times the Hessian matrix, is usually called the *curvature matrix.*

Equation (2.13), the steepest descent formula, translates to

$$\delta a_l = \text{constant} \times \beta_l \ . \tag{2.19}$$

The components α_{kl} of the Hessian matrix in Equation (2.16) depend both on the first and second derivatives of the basis functions with respect to their parameters. Second derivatives occur because the gradient in Equation (2.15) already has a dependence on $\frac{\partial y}{\partial a_k}$, so the next derivative contains terms involving $\frac{\partial^2 y}{\partial a_l \partial a_k}$. The second derivative term can be dismissed when it is zero (as in a linear function), or small enough to be negligible when compared to the term involving the first derivative. In practice, this is quite often ignorably small. If we look at Equation (2.16) carefully, the second derivative is multiplied by $[y_i - y(\hat{x}_i; \hat{a})]$. For a successful model, this term should just be the random measurement error of each point. This error can have either sign, and should in general be uncorrelated with the model. Therefore, the second derivative terms tend to cancel out when summed over i.

Inclusion of the second-derivative term can in fact be destabilizing if the model fits badly or is contaminated by outlier points that are unlikely to be offset by compensating points of opposite sign. Therefore the definition of α_{kl} becomes

$$\sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[\frac{\partial y(\hat{x}_i;\hat{a})}{\partial a_k} \frac{\partial y(\hat{x}_i;\hat{a})}{\partial a_l} \right] . \tag{2.20}$$

2.2.4 Marquardt's Method

Most algorithms for the least-squares estimation of nonlinear parameters have centered about either of the two approaches introduced in this section. Both methods not infrequently run aground: the Taylor series method because of its divergence of the successive iterates, and the steepest-descent (or gradient) methods because of agonizingly slow convergence after the first few iterations. Marquardt (1963) developed an elegant *maximum neighborhood* method, related to an earlier suggestion of Levenberg (1944), which, in effect, performs an optimum interpolation between the Taylor series method (2.18) and the steepest-descent method (2.19), the interpolation being based upon the maximum neighborhood in which the truncated Taylor series gives an adequate representation of the nonlinear model. Taylor series method is used far from the minimum, switching continuously to the steepest-descent method as the minimum is approached. This *Marquardt method* works very well in practice and has become the standard of nonlinear least-squares routines (Press et al., 1992).

The Marquardt method is based on two important insights. There is no information about the scale or the value of the "constant" in Equation (2.19) in the gradient. Marquardt surmised that the components of the Hessian matrix must hold some information about the order-of-magnitude scale of the problem. Equation (2.7) shows that χ^2 is nondimensional. But β_k has the dimensions of $\frac{1}{a_k}$, which may be dimensional. The constant of proportionality between β_k and δa_k must therefore have the dimensions of a_k^2. Among the components of $[\alpha]$ there is only one quantity with these dimensions: $\frac{1}{\alpha_{kk}}$, the reciprocal of the diagonal element. So he suggested that $\frac{1}{\alpha_{kk}}$ must set the scale of the constant. To avoid the scale becoming too large, it is divided by a nondimensional fudge factor λ. The step can be cut down drastically by setting $\lambda \gg 1$. Equation (2.19) is then replaced by

$$\delta a_l = \frac{1}{\lambda a_{ll}} \beta_l \text{ or } \lambda a_{ll} \delta a_l = \beta_l \ . \tag{2.21}$$

It is necessary that a_{ll} be positive, but this is guaranteed by definition from Equation (2.20).

Marquardt's second insight is that Equations (2.21) and (2.18) can be combined if we define a new matrix α' by the following prescription

$$\begin{aligned} \alpha'_{jj} &\equiv \alpha_{jj}(1+\lambda) \\ \alpha'_{jk} &\equiv \alpha_{jk},\ \forall j \neq k \end{aligned} \tag{2.22}$$

and replace both Equations (2.21) and (2.18) by

$$\sum_{l=1}^{M} \alpha'_{kl} \delta a_l = \beta_k \ . \tag{2.23}$$

When λ is very large, the matrix α' is forced into being *diagonally dominant*, so equation (2.23) goes over to be identical to Equation (2.21).

For notational convenience, let us denote the value of χ^2 at iteration $r-1$ as $\Phi^{(r-1)}$. Let $\lambda^{(r-1)}$ denote the value of λ from the previous iteration. Initially let $\lambda^{(0)} = 10^{-2}$, say. Marquardt's strategy to find optimal choice for $\lambda^{(r)}$ is as follows:

Let $\nu > 1$
Compute $\Phi(\lambda^{(r-1)})$ and $\Phi(\frac{\lambda^{(r-1)}}{\nu})$.

1. If $\Phi(\frac{\lambda^{(r-1)}}{\nu}) \leq \Phi^{(r)}$, let $\lambda^{(r)} = \frac{\lambda^{(r-1)}}{\nu}$.
2. If $\Phi(\frac{\lambda^{(r-1)}}{\nu}) > \Phi^{(r)}$ and $\Phi(\lambda^{(r-1)}) \leq \Phi^{(r)}$, let $\lambda^{(r)} = \lambda^{(r-1)}$.
3. If $\Phi(\frac{\lambda^{(r-1)}}{\nu}) > \Phi^{(r)}$, and $\Phi(\lambda^{(r-1)}) > \Phi^{(r)}$, increase λ by successive multiplication by ν until for some smallest w, $\Phi(\lambda^{(r-1)}\nu^w) \leq \Phi^{(r)}$. Let $\lambda^{(r)} = \lambda^{(r-1)}\nu^w$.

Given an initial guess for the set of fitted parameters $\hat{a}$, Press et al. (1992) implement the Marquardt method as follows:

1. Compute $\chi^2(\hat{a})$.
2. Pick a modest value of $\lambda = 0.001$.
3. Solve linear equations in Equation (2.23) for δa and evaluate $\chi^2(\hat{a} + \delta\hat{a})$.
4. If $\chi^2(\hat{a} + \delta\hat{a}) \geq \chi^2(\hat{a})$, increase λ by a factor of 10 and go back to Step 3.
5. If $\chi^2(\hat{a} + \delta\hat{a}) < \chi^2(\hat{a})$, decrease λ by a factor of 10, update the trial solution $\hat{a} \leftarrow \hat{a} + \delta\hat{a}$, and go back to Step 3.

Given the iterative procedure, we also need a condition for stopping the iterations. It is not necessary to continue until convergence (to machine accuracy), since a change in the parameters that changes χ^2 by an amount $\ll$1 is never statistically meaningful (Press et al., 1992). Parameters are often found wandering around near the minimum in a flat valley of complicated topography. Therefore, in practice, one might as well stop iterating on the first or second occasion that χ^2 decreases by a negligible amount, either absolutely or some fractional amount. But iterations should not be stopped after a step where χ^2 increases: That only shows that λ has not yet adjusted itself optimally.

2.2.5 Problems Associated with Nonlinear Optimization Techniques

One potential problem with nonlinear optimization techniques is that finding a global minimum of χ^2 is not guaranteed. A greater problem faced by almost every nonlinear optimization technique is non-convergence, which is convergence to a solution that is relatively very far (in terms of χ^2) from a global optimum. Non-convergence increases with the complexity of the function to be optimized and the number of parameters. These apply to Marquardt's method also.

Since the Marquardt method also uses Taylor series iteration, another problem is a singular Hessian matrix for which we need to find an inverse. This happens due to various reasons, such as one parameter being *co-dependent* on another parameter and the value of one parameter *smothering* another parameter (Sherrod, 2001). To illustrate the former case, consider the function $y = p_0 + p_1 p_2 x$. This does not have a unique solution since p_2 can take almost any value if a corresponding value for p_1 is chosen. To illustrate the latter case, consider the function $y = p_0 + p_1 x^{p_2}$. If during the solution process, p_1 takes on the value 0, then varying the value of p_2 has no effect on the equation, and the method cannot figure out which way to change the value of p_2 to move toward convergence.

This phenomenon can be generalized as follows: Equations (2.12) and (2.18) can be viewed as linear mapping from the vector space δa to the vector space $\nabla \chi^2$ by $\hat{D}$. If the matrix $\hat{D}$ is nonsingular, it maps vector space δa into the vector space $\nabla \chi^2$ of the same dimension, and there is only one possible value for δa. If it is singular, it maps δa into a vector of lower dimensionality, and for a given $\hat{D}$ there are more than one possible vectors that can be mapped to $\nabla \chi^2$. Taylor series methods do not provide a solution in such cases, since the inverse of the Hessian matrix does not exist.

2.3 Correlation between attributes

In certain cases, the dependent variable may not depend on one or more independent variables. In such cases, it is better to remove such variables, since they will only mislead the least-square fit. There are several techniques to evaluate the correlation or dependence between two variables. These can be used to find uncorrelated variables and remove them from the model so that the data can be fit to a more realistic model.

2.3.1 Linear Correlation

Linear correlation is a measure of association between continuous variables. For pairs of quantities (x_i, y_i), $i = 1, \ldots, N$, the linear *correlation coefficient*, r, is given by the formula

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2}\sqrt{\sum_i (y_i - \bar{y})^2}} \quad ,$$

where $\bar{x}$ is the mean of the x_i's and $\bar{y}$ is the mean of y_i's.

The value of r lies in $[-1, 1]$. When all data points lie on a straight line with a positive slope (y increasing as x increases), r becomes 1. This is known as a *complete positive correlation*. If, on the other hand, all data points lie on a straight line with a negative slope (y decreasing as x increases), we call that *complete negative correlation*, with r becoming -1. If r takes on a value very close to zero, it indicates that x and y are uncorrelated. r can be used to summarize a correlation if the correlation itself is significant. But it cannot tell us whether that correlation is *statistically* significant. Since r does not take into account the individual distributions of x and y, there is no universal way to compute its distribution in case of null hypothesis.

2.3.2 Non-parametric Correlation

The uncertainty in interpreting the significance of the linear correlation coefficient r leads us to the important concepts of *nonparametric* or *rank correlation*. It obviates the difficulties that arise from lack of information on the probability distribution function from which the x_i's or y_i's were drawn.

If we replace the value of a point x_i by its rank among all the x_i's in the sample, we would get a list of numbers uniformly drawn from a known distribution – integers between 1 and N, inclusive. If all the x_i's are distinct, then each integer will occur exactly once. If some x_i's have the same values, then the "ties" are assigned the mean of the ranks that they would have had if their values had been slightly different (without affecting the ranks of points other than the ties.) This midrank will sometimes be an integer, sometimes a half-integer. But the sum of all the assigned ranks will be equal to the sum of all integers from 1 to N, which is $\frac{1}{2}N(N+1)$. We repeat the same procedure for y_i's, replacing each value by its rank among all y_i's in the sample.

Now the job is to detect correlation between uniform sets of integers between 1 and N (with the possibility of ties in the ranks). Of course, we lose some information when we replace the value of a point with its rank, but the objective is different here. We are trying to correlate the relative ordering of the points, which is nonparametric, in x and y. If we find a correlation using the ranks, then we can be more confident that there is really a correlation between x and y thus avoiding potential spurious correlations, i.e., resulting from points with measurement errors. In this manner, nonparametric correlation is more robust than parametric correlation. This can be compared to the idea of the median being more robust that the mean. Two particular choices of a statistic are the Spearman rank-order correlation coefficient (r_s), and Kendalls tau (τ).

Spearman Rank-Order Correlation Coefficient

Let R_i be the rank of x_i among the other x's, S_i be the rank of y_i among the other y's, ties being assigned the appropriate midrank. Then the rank-order correlation coefficient is defined to be the linear correlation coefficient of the ranks, namely,

$$r_s = \frac{\sum_i (R_i - \bar{R})(S_i - \bar{S})}{\sqrt{\sum_i (R_i - \bar{R})^2}\sqrt{\sum_S (S_i - \bar{S})^2}} \quad .$$

The significance of a nonzero value of r_s is tested by computing

$$t = r_s\sqrt{\frac{N-2}{1-r_s^2}}.$$

Kendall's τ Coefficient

While the rank order coefficient uses the numerical difference of ranks, Kendall's τ uses only the relative ordering of these ranks: higher in rank, lower in rank, or in the same rank. In that sense, τ us more nonparametric than r_s. But τ and r_s are very strongly correlated, and in most applications, are effectively the same test.

Consider all $\frac{1}{2}N(N-1)$ pairs of data points, where a data point cannot be paired with itself, and where the points in either order count as one pair. We call a pair *concordant* if the relative ordering of the ranks of the two x's is the same as the relative ordering of the ranks of the two y's. We call a pair *discordant* if the relative ordering of the ranks of the two x's is opposite from the relative ordering of the ranks of the two y's. If there is a tie in either the ranks of the two x's or the ranks of the two y's, then we do not call the pair either concordant or discordant. If the tie is in the x's, we will call the pair an "extra y pair." If the tie is in the y's, we will call the pair an "extra x pair." If the tie is in both the x's and the y's, we do not call the pair anything at all. Kendall's τ is now the combination of these various counts:

$$\tau = \frac{\text{concordant} - \text{discordant}}{\sqrt{\text{concordant} + \text{discordant} + \text{extra-}y}\sqrt{\text{concordant} + \text{discordant} + \text{extra-}x}} \ .$$

This must lie between 1 and -1, and takes on the extreme values only for complete rank agreement or complete rank reversal, respectively. In the null hypothesis of no association between x and y, τ is approximately normally distributed, with zero expectation and a variance of

$$Var(\tau) = \frac{4N + 10}{9N(N-1)} \ .$$

Chapter 3

Related Work

Various attempts have been made to fit data to a piecewise hypothesis consisting of many sub-hypotheses, each of which is active only in a single region of space. Though each of these attempts has exploited different characteristics of the problem, they can be broadly categorized, among many, into two approaches – regression approaches and decision tree approaches.

3.1 Regression Approaches

These approaches solve the piecewise regression problem by formulating either a new regression problem or multiple independent regression problems whose solution presumably gives a near-optimal solution to the original problem.

Quandt (1958) estimated the parameters of a linear regression system obeying at most two separate regimes. Hudson (1966) reported a method to find the overall least squares solution when a complete curve to be fit consists of two or more submodels (though his main results are based on models consisting only of two submodels). But an important constraint in his approach is that the overall model is continuous at each join point. McGee and Carleton (1970) outlined an approach that uses hierarchical clustering to cluster the points into segments that represent the individual regimes of the piecewise function and perform standard linear regression on them.

Tishler and Zang (1981) formulated a unified function from the piecewise linear function using `min` and `max` operators and use a nonlinear optimization technique to minimize mean squared error between the sample data and the hypothesis. The choice of a sub-hypothesis in a regime is expressed as a combination of `min` and `max` operators on all the sub-hypotheses. Choosing the appropriate combination of these two operators requires prior knowledge of the underlying sub-hypotheses (at least their ordering in each regime). They also reported that replacing linear functions by non-linear functions could be a possible extension. Yu et al. (2001) proposed a general fuzzy piecewise regression method that formulates the problem as a mixed-integer programming problem which restricts the function in each regime to be linear and/or restricts the overall model to be a linear combination of individual functions.

3.2 Decision Tree Approaches

Decision tree learning is a widely used and practical method for inductive inference of data. It approximates discrete-valued functions, is robust to noisy data and is capable of learning disjunctive expressions. Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute. Decision tree approaches can naturally handle *nominal* attributes (or *enumerated* attributes) – attributes that can only take (possibly non-numeric) values from a set of predefined entries.

Most algorithms that have been developed for learning decision trees are variations on a core algorithm that employs a top-down, greedy search through the space of possible decision trees. Classification and Regression Trees (CART) system, introduced by Breiman et al. (1984), is a tree learning technique that assigns constant values at the leaves of the tree. Thus it can fit piecewise constant data very well and piecewise linear data with some error. The top-down approach is exemplified in C4.5 (Quinlan, 1993) which was independently derived but similar to CART. C4.5 included methods for dealing with numerical attributes, missing values, noisy data and generating rules from trees.

Motivated by these recursive partitioning approaches to regression, Friedman (1991) presented a new method, Multiple Adaptive Regression Splines (MARS), for flexible regression modeling of high-dimensional data. The model takes the form of an expansion in product spline basis functions, where the number of basis functions as well as the parameters associated with each one are automatically determined by the data. This method produces continuous models with continuous derivatives. The simplest model learns a linear combination of nonlinear functions of individual inputs. It can generate rectilinear partitions of the sample space by choosing knot points in each dimension (in the attribute space) and fit these functions within each partition. Each knot point lies on an axis of the attribute space. MARS also allows for interactions between attributes by analyzing the dependencies using two-way analysis of variance. The use of knot points in MARS is analogous to the use boundary surfaces in `EMPRR`. More information on this comparison is given in Section 4.2.3, where we explain the boundary surfaces in `EMPRR`.

Quinlan (1992) developed a new system, M5, that combines conventional decision trees with linear regression functions at the leaves. Like CART, M5 builds tree-based models; but, whereas regression trees have constant values at their leaves, the trees constructed by M5 can have multivariate linear models. These *model trees* are analogous to piecewise linear functions.

M5 and CART learn efficiently and can build accurate models of high-dimensional data — up to hundreds of attributes. This ability sets M5 and CART apart from MARS, whose computational requirements grow very rapidly with dimension, effectively limiting its applicability to tasks with no more than 20 or so attributes. The advantage of M5 over CART is that model trees are generally much smaller than regression trees.

Wang and Witten (1997) described a new implementation of a model-tree inducer (`M5′`) based on Quinlan's work. They adopted a method from CART for dealing with enumerated attributes, and adapted a method for treating missing values. They reported that `M5′` performs somewhat better than M5 on the standard data sets for which results have been published.

Gama (2002) presented a new algorithm that allows combination of attributes both at decision and leaf nodes of what are called *functional trees*. The algorithm combines a univariate regression tree with a linear regression function by means of constructive induction. This approach can be seen as a hybrid model that combines linear regression (known to have low variance) with a regression tree (known to have low bias). Results from his work show that functional trees are competitive against the state-of-the-art in model trees.

These models (except MARS) scale up easily to higher-dimensional data and their performances are not affected much by the dimensionality of data. But none of these regression/optimization methods (except MARS) solves the problem of fitting data to a general piecewise nonlinear function, e.g. the bacterial growth curve defined by Equation (2.2). But, as we mentioned earlier, MARS can only generate rectilinear partitions in the sample space.

Chapter 4

Piecewise Surface Regression Algorithm

We begin the discussion of our model with the case of a single independent variable. Then we generalize this model to higher dimensions (surface regression). We also discuss the variations to our basic surface regression algorithm and the implementation of our algorithm.

4.1 Single-Dimensional Piecewise Regression

In our single-dimensional model, we first transform a piecewise function to a single, continuous function using the Heaviside function (also known as the Unit Step function), defined as

$$U(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} .$$

We can perform this transformation by switching on and switching off the appropriate functions at the right time. For example, the function

$$f(x) = \begin{cases} 4x^2 + 2x + 5 & \text{if } x < 8 \\ \sin x & \text{if } 8 \leq x < 15 \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

can be transformed using the Heaviside function to

$$\begin{aligned} f(x) &= (4x^2 + 2x + 5) - (4x^2 + 2x + 5) \cdot U(x-8) \\ &\quad + U(x-8) \cdot \sin x - U(x-15) \cdot \sin x \end{aligned} \quad . \tag{4.2}$$

We then use the Marquardt non-linear optimization method[1] (Marquardt, 1963), discussed at length in Section 2.2.4, to minimize the sum of the squared errors (SSE) between the expected and observed values. The Marquardt method is one of the most widely-used non-linear curve fitting methods. It is very useful for finding solutions to complex fitting problems. It varies smoothly between the extremes of the Taylor Series method and the steepest descent method for finding the next step size. The latter method is used far from the minimum, switching continuously to the former as the minimum is approached. This technique requires differentiation of the function to be optimized. A differentiable approximation for the Heaviside function is the sigmoid function[2], defined as

$$\sigma(x) = \frac{1}{1 + e^{-ax}} \quad ,$$

and Equation (4.2) can now be rewritten as

$$\begin{aligned} f(x) &= (4x^2 + 2x + 5) - (4x^2 + 2x + 5) \cdot \sigma(x-8) \\ &\quad + \sigma(x-8) \cdot \sin x - \sigma(x-15) \cdot \sin x \end{aligned} \quad .$$

Our initial model assumes that there is exactly one function between two contiguous transition points and that we know the general form of that function. Theoretically, any function that can be regressed (e.g. polynomial, exponential) can occur in a regime. If the optimization technique is robust, it will fit the data well.

Although the functions flanking a transition point might meet at that point, it is not necessary in our model because this constraint is not imposed, unlike in Hudson (1966). While this is a useful feature for some application areas, discontinuous regimes reveal a minor difficulty in our model: They could cause a nonzero minimum error even if there exists a piecewise model that fits the data with zero error. This is

[1] Any nonlinear optimization method can be used, preferably a robust method.
[2] We used $a = 10$ in all our experiments.

because the sigmoid function is only an approximation of the Heaviside function: It does not change from 0 to 1 in the ordinate exactly at 0 in the abscissa (in fact, $\sigma(0) = 0.5$). Therefore exactly at a transition point t_p, the value of the unified function is the average of the values of the two functions, say $f(t_p)$ and $g(t_p)$, in the flanking regimes, whereas the discontinuous piecewise function jumps to g at t_p. Therefore non-zero error in our solution is unavoidable if there is extensive sampling near the transition point. Note that in many cases, we can write the unified function to require continuity across regimes simply by defining the start point of regime i to be a function of the end point of regime $i-1$.

There is a straightforward efficient dynamic programming solution to this one-dimensional problem, so nonlinear optimization is in fact unnecessary. What is more interesting is the surface regression problem, for which there is no obvious efficient optimal algorithm.

4.2 Piecewise Surface Regression Model

It is possible to extend the model of Section 4.1 to higher dimensions by generalizing the curves of each regime to surfaces, and generalizing the transition points to surfaces as well. In a higher-dimensional piecewise function (piecewise surface function), we assume that the individual functions are multidimensional surfaces of multiple independent variables. The boundaries of regimes are also multidimensional, e.g., for a piecewise surface function with D independent variables, the boundary surface is D-dimensional. We partition the independent variables into two sets x and $\hat{z}$, where x is a single independent variable and $\hat{z}$ is a vector of all the other independent variables. We assume that the boundary surfaces can be written as $x = b(\hat{z})$, since a piecewise function has to be piecewise in some dimension. The piecewise surface function can be written as

$$F(x,\hat{z}) = \begin{cases} f_1(x,\hat{z}) & \text{if } x < b_1(\hat{z}) \\ f_2(x,\hat{z}) & \text{if } b_1(\hat{z}) \le x < b_2(\hat{z}) \\ & \vdots \\ f_n(x,\hat{z}) & \text{if } b_{n-1}(\hat{z}) \le x \end{cases}, \tag{4.3}$$

where

f_i is the surface in regime R_i, and

b_i is the boundary surface between regimes R_i and R_{i+1}.

Then a natural single-equation representation of the piecewise surface can be formed as

$$\begin{aligned} \hat{Y}(x,\hat{z}) &= f_1(x,\hat{z}) - f_1(x,\hat{z}).U(x - b_1(\hat{z})) \\ &\quad + f_2(x,\hat{z}).U(x - b_1(\hat{z})) - f_2(x,\hat{z}).U(x - b_2(\hat{z})) \\ &\quad \vdots \\ &\quad + f_n(x,\hat{z}).U(x - b_{n-1}(\hat{z})) \end{aligned}, \tag{4.4}$$

and nonlinear optimization can be applied as in Section 4.1. However, this generalization increases the dimension of the search space significantly. For example, for linear f and b with n regimes, the number of parameters is $(2n-1)|\hat{z}|+3n-1$, where $|\hat{z}|$ is the number of independent variables in $\hat{z}$. Even for reasonably high values of n and $|\hat{z}|$, this greatly increases the chance of the nonlinear optimization method getting trapped in a local minimum. Therefore, we formulate our model with two disjoint sets of parameters: one for the surfaces of the regimes (α) and one for the boundary surfaces between regimes (β). We can now *fix* one set of parameters (e.g. the boundary surfaces) while optimizing the other (e.g. the piecewise surfaces). We then alternate in an *EM-like fashion* until little improvement is seen.

With the partitioned parameter set, the piecewise surface function can be written as

$$F(x,\hat{z},\alpha,\beta) = \begin{cases} f_1(x,\hat{z},\alpha_1) & \text{if } x < b_1(\hat{z},\beta_1) \\ f_2(x,\hat{z},\alpha_2) & \text{if } b_1(\hat{z},\beta_1) \le x < b_2(\hat{z},\beta_2) \\ & \vdots \\ f_n(x,\hat{z},\alpha_n) & \text{if } b_{n-1}(\hat{z},\beta_{n-1}) \le x \end{cases}, \tag{4.5}$$

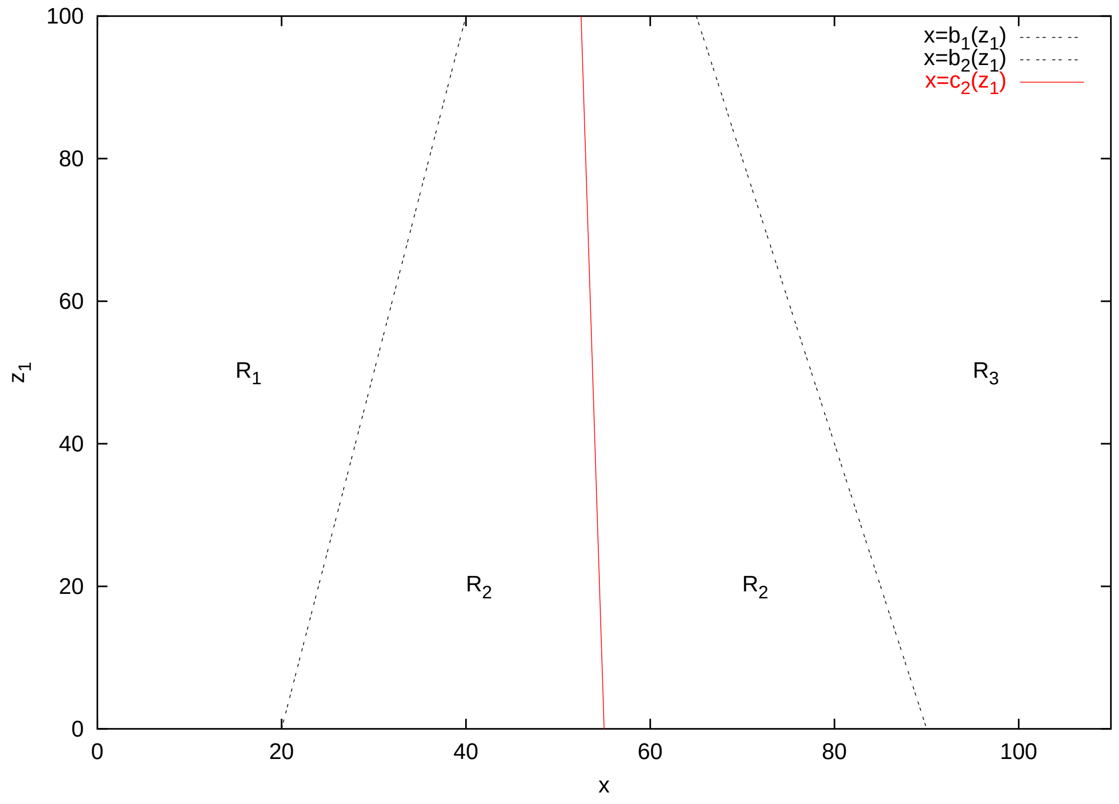

Figure 4.1: Sample boundary surface in case of 2 independent variables. Primary independent variable x is dependent on z_1 on the boundary surface. (Third dimension of the dependent variable not shown.)

where

α_i is the set of parameters that characterize f_i,

β_i is the set of parameters that characterize b_i,

$\alpha = \{\alpha_i | 1 \leq i \leq n\}$, and

$\beta = \{\beta_i | 1 \leq i \leq n-1\}$.

Note that in Equations (4.3) and (4.5), we assume that the regimes do not overlap each other within the bounding box of the data points. Currently our model assumes that the regimes do not overlap and we develop our algorithm under this assumption.

1. Make an initial guess for β
2. Regress individual regimes for α
3. Fix α for all regimes
4. Solve for β by fitting data to the model defined by Equation (4.10)
5. If convergence criterion not met go to Step 2

Figure 4.2: Algorithm `EMPRR`.

As an example, consider a piecewise surface function defined by

$$F(x,\hat{z}) = \begin{cases} 2 + 3x - 4z_1 & \text{if } x < 0.2z_1 + 20 \\ 1 - 8x + 2z_1 & \text{if } 0.2z_1 + 20 \leq x < -0.25z_1 + 90 \\ -5 - 3x + 4z_1 & \text{if } -0.25z_1 + 90 \leq x \end{cases}, \tag{4.6}$$

The 2-D projection of this sample 3-D piecewise function with 3 regimes is shown in Figure 4.1. The boundary surfaces (black dotted lines in Figure 4.1) are defined by:

$$\begin{array}{rcl} b_1(z_1) & = & 0.2z_1 + 20 \\ b_2(z_1) & = & -0.25z_1 + 90 \end{array}. \tag{4.7}$$

We then employ `EMPRR` which will optimize the parameters α and β independently of each other in two subsequent stages during each iteration. We start with an initial guess of β and individually regress the points that lie between two adjacent boundary surfaces. Thus we obtain a set of values for parameters in α. Fixing the values of α, we now solve for the optimal values of parameters in β. This iterative procedure is repeated until a convergence criterion is met. The algorithm is given in Figure 4.2.

4.2.1 Optimizing Regimes

Solving for α during the first stage of the iterative process is simple. During Step 2 in Figure 4.2, for each regime R_i, we pass the relevant points (between the boundary surfaces b_{i-1} and b_i) to the Marquardt method and obtain the optimal values of α_i that characterize that regime. α can now be obtained by combining all α_i. Recall our assumption in Section 4.2 that the boundary surfaces do not overlap. This assumption makes the assignment of a regime to a point very simple. In the case of $n-1$ overlapping boundary surfaces, there will be (perhaps exponentially) more than n regimes. Then assignment of regimes becomes complex and time complexity increases significantly.

Attributes are also tested for their significance and only those attributes that are significant are used in the regression. For all the points in R_i, the mean and standard deviation for each attribute is calculated. If the standard deviation of an attribute (within that regime) is infinitesimal relative to its mean, the attribute is considered insignificant. Moreover, if the values of the attributes themselves are infinitesimal, they are considered insignificant too. This is done for each regime independently. More often than not, such attributes lead to singularities during optimization.

4.2.2 Optimizing Boundaries

Solving for β becomes more complex for general boundary surfaces. This is because normally, sum of squared error (when the piecewise surfaces defined by α are fixed) only changes when data points move from one regime to another. This discretizes the search space, making application of Marquardt's method more difficult. Hence we need a continuous function in terms of α and β to fit the data. For the boundary surface to be effectively varied during the optimization, this function should be sensitive to the interaction between α and β. Thus interaction between α_j and β_k, $j \neq k$, should be explicitly incorporated in this function. But we will keep α constant since we need to only solve for β for the values of α obtained from the previous step. Therefore we decided to evaluate the expected value of $y(x_m, \hat{z}_m)$ for a point $p = (x_m, \hat{z}_m, y_m)$ in regime R_i as a weighted sum of f_{i-1}, f_i and f_{i+1}, the weights being determined by a non-trivial subset of β. This would effectively *share* points between 3 regimes.

A natural choice for the weights would be a Normal distribution centered at R_i with a standard deviation of half the width of R_i — the weight being maximum at the centre and monotonically decreasing with respect to the distance from the centre. But the weight at the centre is not uniform over all regimes if they are not all of equal widths. This is a result of normalizing the distribution by σ, the standard deviation of the distribution. Therefore we chose not to normalize the distribution. Thus we define our M-function (a pseudo- distribution since the area under the function is not necessarily 1) as:

$$M(\mu, \sigma^2)|_x = \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \quad . \tag{4.8}$$

where μ is the mean of the pseudo-distribution. This function has the following characteristics:

$$\begin{aligned} M(\mu, \sigma^2)|_x &= 1 \text{ when } x = \mu \ , \\ M(\mu, \sigma^2)|_x &= 0 \text{ when } |x - \mu| = \infty \ . \end{aligned} \tag{4.9}$$

For any point that belongs to regime R_i, we define the following model to fit the data while solving for β:

$$\begin{aligned} y(x_m, \hat{z}_m; \beta) \ = \ & f_i(x_m, \hat{z}_m, \alpha_i) \cdot [1 - \delta_{i+1}(p) - \delta_{i-1}(p)] \\ & + f_{i+1}(x_m, \hat{z}_m, \alpha_{i+1}) \cdot \delta_{i+1}(p) \qquad , \\ & + f_{i-1}(x_m, \hat{z}_m, \alpha_{i-1}) \cdot \delta_{i-1}(p) \end{aligned} \tag{4.10}$$

where

$\delta_j(p)$ is the value of $M(c_j, {w_j}^2)$ at $|x_m - c_j|$,

c_j is the centre of regime R_j (defined as the mean of b_{j-1} and b_j), and

w_j is the half-width of R_j (defined as the distance between c_j and b_j).

Note that if b_j is not constant, the values c_j and w_j are going to be different for each instance of $\hat{z}$. Thus the functions $M(c_j, {w_j}^2)$ and $\delta_j(p)$ are all specific to a particular instance of $\hat{z}$. Unless the boundary surfaces are perfectly *normal* to the x-axis, they would evaluate to be different at each instance of $\hat{z}$.

This model induces the expected interaction between α and β — Equation (4.10) involves α_{i-1}, α_i, α_{i+1}, β_{i-2}, β_{i-1}, β_i and β_{i+1}. Note that the fractional weight for f_i is determined by $\delta_{i+1}(p)$ and $\delta_{i-1}(p)$. There are two motives for this choice: to make the fractional weights sum up to 1, and to increase effective sharing of each point. This model lets the points closer to the centre of the regime influence the parameters of its

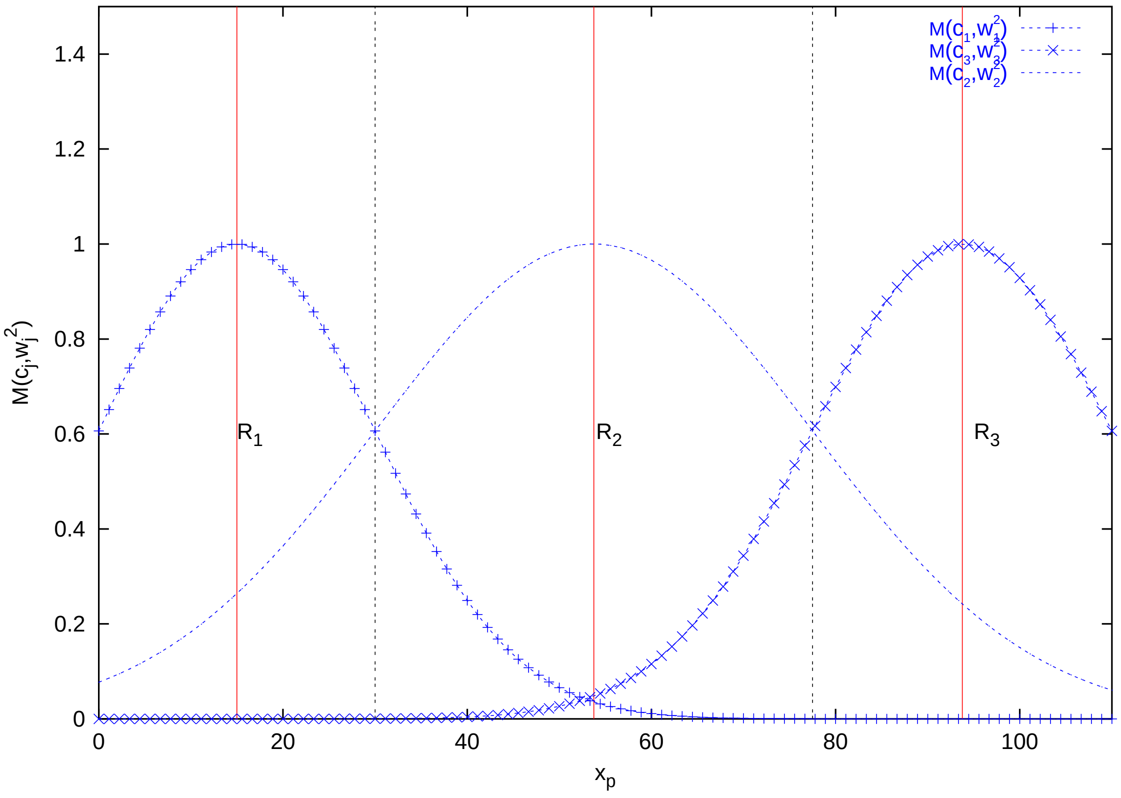

Figure 4.3: M-functions for the example from Figure 4.1, at $z_1 = 50$.

boundary surfaces more. Another important characteristic is that the M-functions centered at c_i and c_{i+1} meet at b_i. Points farther from c_j have little or no influence over b_{j-1} or b_j. We now minimize the sum of $[y_m - y(x_m, \hat{z}_m)]^2$ using the Marquardt method.

Figure 4.1 shows the centre c_2 (red solid line) for R_2 for the example mentioned in Equation (4.6). M-functions centered at R_j's for that example and the fractional weights of f_1, f_2 and f_3 for a point in R_2 for $z_1 = 50$ are shown in Figures 4.3 and 4.4, respectively. Notice that $\hat{z}$ ($= z_1$ in this case) is not visible in these figures. As mentioned earlier in this section, the M-functions are specific to a particular instance of $\hat{z}$, and thus the boundary surfaces that were not necessarily axis parallel in the $\hat{z}$-axis in Figure 4.1 are now axis parallel, but in a different axis (δ in this case). In

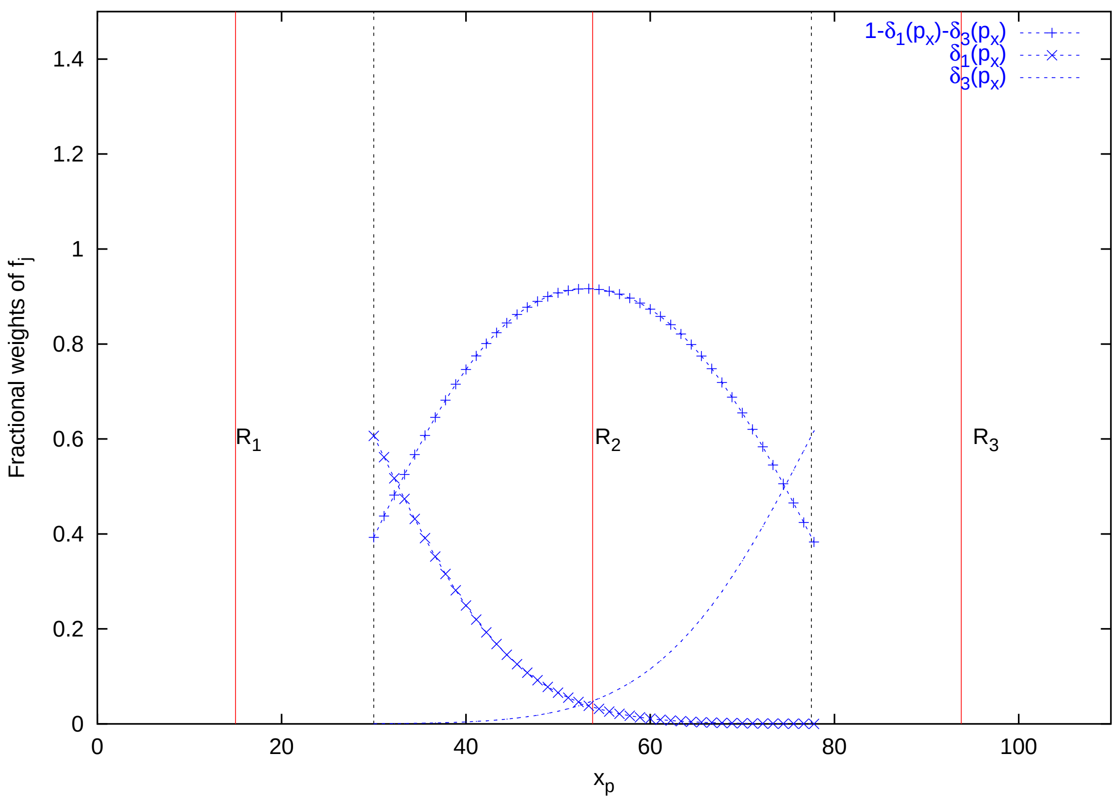

Figure 4.4: Fractional weights of f_1, f_2 and f_3 for a point in R_2 at $z_1 = 50$ for the example from Figure 4.1.

Figure 4.3, M-functions centrad at R_1,R_2 and R_3 are plotted against x_p. Notice that the M-functions meet at the boundary surfaces. If a point lies in regime R_i, the M-function of R_j, $j \neq i$, does not evaluate higher for that point than the M-function of R_i. In Figure 4.4 though, the fractional weights of f_j on a point belonging to R_i ($j \neq i$) can be higher than the fractional weight of f_i. The curve marked with '+' is the fractional weight of f_2 on points within R_2. Notice that near the boundary the fractional weight of f_2 is slightly less than the fractional weight of the function across the boundary.

4.2.3 Comparison to MARS

As we mentioned in Section 3, MARS's estimation of the target function, which was not known to us during `EMPRR`'s development, is analogous to (but not the same as) Equation (4.8). The following simplified discussion of Friedman (1991) has been adopted from Moore (2001). For notational simplification, let us redefine the notations for the independent attributes for this section (Section 4.2.3) only. Each data point $\hat{x}$ is the vector of size m containing individual scalars x_k, $k = 1, \ldots m$ corresponding to the m attributes. Consider a linear combination of m nonlinear functions, one for each attribute, in the form of

$$y(\hat{x}) = \sum_{k=1}^{m} g_k(x_k) = \sum_{k=1}^{m} \sum_{j=1}^{N_k} h_j^k f_j^k(x_k). \tag{4.11}$$

where

$$f_j^k(\hat{x}) = \begin{cases} 1 - \frac{|x_k - q_j^k|}{w_k} & \text{if } |x_k - q_j^k| < w_k \\ 0 & \text{otherwise} \end{cases} ,$$

and

N_k is the number of knots in the kth dimension,
q_k^j is the location of the jth knot in the kth dimension,
h_k^j is the regressed height of the jth knot in the kth dimension, and
w_k is the spacing between knots in the kth dimension.

Fitting data to Equation (4.11) generates a decision tree model, where a subset of the subspaces is linear. Here, a point $\hat{x}$ is shared between 2 knot points in each dimension, and the sharing across dimensions are combined linearly. MARS also allows two-way interactions between attributes. Consider

$$y(\hat{x}) = \sum_{k=1}^{m} g_k(x_k) + \sum_{k=1}^{m} \sum_{t=k+1}^{m} g_{kt}(x_k, x_t). \tag{4.12}$$

Equations (4.11) and (4.12) can be expressed as a linear combination of basis functions and can be learned by linear regression (Section 2.2.2).

The effective sharing technique used in MARS is thus comparable to our technique. But MARS considers each dimension independently whereas `EMPRR` performs a multidimensional decision at boundary surfaces. Also, MARS first creates a large number of knot points (up to a predefined threshold) and then prunes them back. The following quote is taken from MARS's user guide (Salford-Systems, 2001):

> In data mining problems, complex interactions must be permitted; thus, it is reasonable to allow literally hundreds of basis functions in a forward search. A very quick way to get a ball park estimate is to first run a CART model (given CART will run faster than MARS) and then allow twice as many basis functions as terminal nodes in the optimal CART tree. In our data mining experience we have rarely allowed for more than 250 basis functions.

4.2.4 Initial Guess for β

A good initial guess helps in converging towards the global minimum. Without prior knowledge about the underlying model, it is generally not possible to come up with a very good guess. We tried a few heuristic methods to choose start points. One way is to partition the feature space into n equal regimes in the x-dimension, using the surfaces $x = x_{min} + k$, $x = x_{min} + 2k, \ldots, x = x_{min} + (n-1)k$ normal to x-axis. Another way is to come up with n partitions, $x = k_1$, $x = k_2, \ldots, x = k_{n-1}$, each containing equal points. The former would be a good guess if the sampling of data is uniform in at least the x-dimension and the latter would be if the sampling is not uniform. In all our experiments, we used equally populated partitions.

4.2.5 Time Complexity

Since `EMPRR` is a nonlinear optimization technique, an absolute bound on the time complexity is not feasible since there is no way of knowing exactly how long it takes for the method to converge. Therefore, to obtain a reasonable approximation, let us make a few assumptions. First let us assume that the number of iterations within the Marquardt optimization method is bounded by I_M and the number of iterations within `EMPRR` (Figure 4.2) is bounded by I_E.

Each call to the Marquardt method first evaluates the function and its derivatives with respect to the model parameters at each point. This takes $O(mN)$ steps where

m is the number of model parameters and N is the number of data points. Then the inverse of the Hessian matrix is calculated using Gauss-Jordan elimination and that takes $O(m^3)$ steps. This is repeated up to I_M iterations. Thus each call to the Marquardt method takes $O\left(m(m^2+N)I_M\right)$ steps, given m parameters to fit on N points.

During each iteration in `EMPRR`, $n+1$ calls are made to the Marquardt method, n for regression of n regimes and one for regression of the boundary surface. Let us assume that all the individual functions f_i are expressed using the same number of parameters, m_R. If N_i is the number of points in R_i, then the regressing R_i takes $O\left((m_R N_i + m_R^3)I_M\right)$ steps. Summing this up over all regimes, we get $O\left((m_R N + m_R^3 n)I_M\right)$ steps for n calls. Now, let the number of parameters in Equation (4.10) be m_B. Then the optimization of boundary surfaces takes $O\left((m_B N + m_B^3)I_M\right)$ steps. Thus each iteration in `EMPRR` takes $O\left((m_R N + m_R^3 n + m_B N + m_B^3)I_M\right)$ steps. Since the number of iterations in `EMPRR` is bounded by I_E, the total time complexity T of `EMPRR` under the aforementioned assumptions is given by

$$O\left((m_R N + m_R^3 n + m_B N + m_B^3)I_M I_E\right) \quad . \tag{4.13}$$

As an example of applying Equation (4.13), consider a sample data set with D attributes and a model with linear functions in regimes and linear boundary surfaces. Then $m_R = D+1$ and $m_B = (n-1)D$. Then the time complexity becomes

$$O\left(((D+1)N + (D+1)^3 n + (n-1)DN + (n-1)^3 D^3)I_M I_E\right).$$

In general, the $(n-1)DN$ term will tend to dominate over $(D+1)N$ term and $(n-1)^3 D^3$ term will tend to dominate over $(D+1)^3 n$. Thus we can simplify the complexity as $O\left(((n-1)DN + (n-1)^3 D^3)I_M I_E\right)$. Now the true complexity depends on which of these is higher – N or $(n-1)^2 D^2$). For a regression problem with a large data set and reasonably low number of regimes and attributes, the $(n-1)DN$ term is going to dominate the time complexity. If the data set is small and the dimensionality and/or the number of regimes are large, then the $(n-1)^3 D^3$ term is going to dominate the time complexity.

A typical population dynamics analysis containing multiple populations each characterized by one instance of Equation (2.2) would have $D = 3$, $n = 3$ and $m_R \leq 2$. If we assume linear boundary surfaces, then $m_B = 6$. Therefore the time complexity becomes

$$O\left((2N + 24 + 6N + 6^3)I_M I_E\right) = O\left(N I_M I_E\right) \ .$$

4.3 Variations

We developed a few variations to the algorithm of Section 4.2. Each variation changes how the optimization procedure performs.

4.3.1 Unscaled and Scaled M-function

Normally, the variance of $M(c_j, {w_j}^2)$ is w_j^2. We denote this as an unscaled M-function. In a scaled M-function, the variance used is $\left(\frac{w_j}{S}\right)^2$ where S is the scaling factor. Regime R_j has more and less influence on its adjacent regimes when $S < 1$ and $S > 1$ respectively. M-functions centered at R_j's for the example in Figure 4.1 and the fractional weights of f_1, f_2 and f_3 for a point in R_2 for $z_1 = 50$ and S=5 are shown in Figures 4.5 and 4.6 respectively. Comparing these two figures with Figures 4.3 and 4.4 (where S=1) clearly shows that the influence of adjacent regimes decreases with an increase in S.

4.3.2 Uniform and Weighted M-function

In a scaled M-function, S is constant across all regimes. We denote this as uniform M-function. In a weighted M-function, the variance scaling factor S_j for R_j depends on R_j's contribution to the total sum of squared error. If e_j is the current SSE in regime R_j, and e is the total SSE for all regimes, we define S_j as

$$S_j = 10^k S \sqrt{\left(\frac{e_j}{e} + \frac{1}{10^{4k}}\right)} \ , \tag{4.14}$$

where k is a user-defined parameter. This is a parabolic relation between e_j and S_j. $S_j \approx 10^k S$ if the total error is contributed by R_j and $S_j = 10^{-k} S$ if R_j contributes no

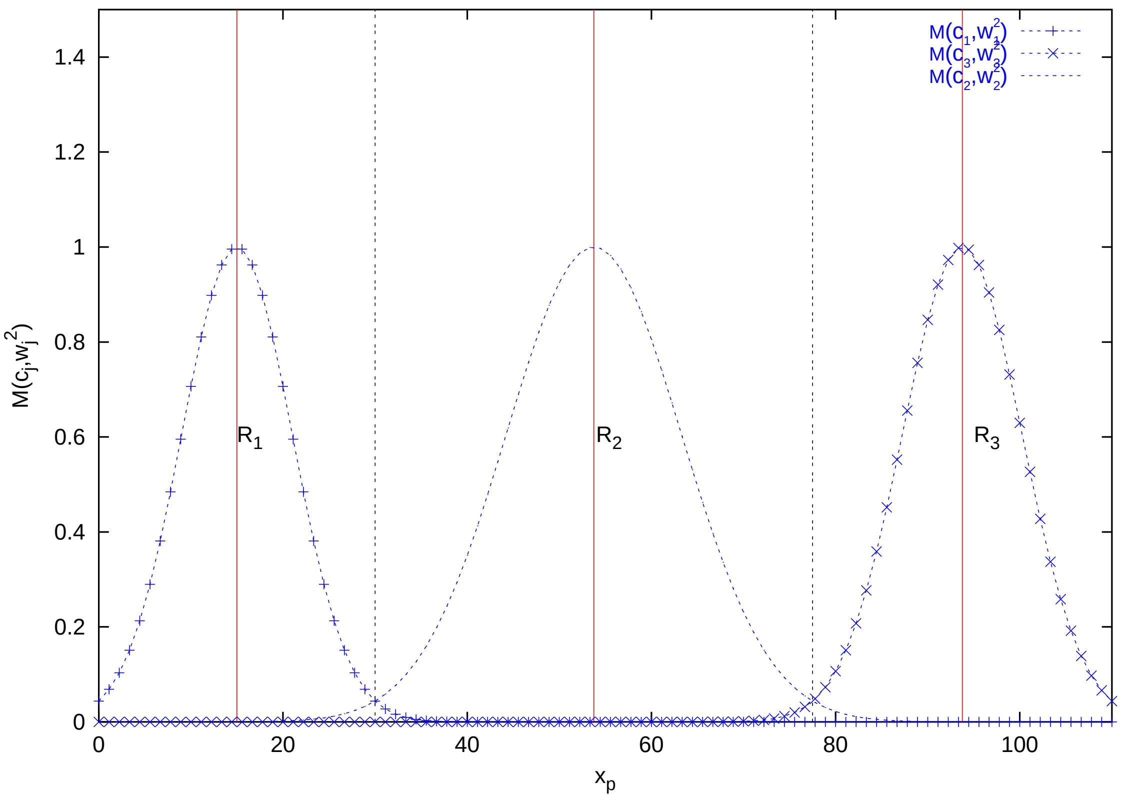

Figure 4.5: M-functions for the example from Figure 4.1 for $z_1 = 50$ and S=5.

error. Thus, a regime with low error influences its adjacent regimes more and regime with high error influences its adjacent regimes less.

4.3.3 Individual and Weighted Regression in Regimes

Marquardt's method allows for specifying the uncertainty σ_i associated with the measurement of each data point p_i. In a weighted regression variation, *all* points are passed to the regression algorithm in each regime R_j; each point p_i has a certainty measure of belonging to R_j defined by

$$1 - \delta_{j+1}(p_i) - \delta_{j-1}(p_i) \ ,$$

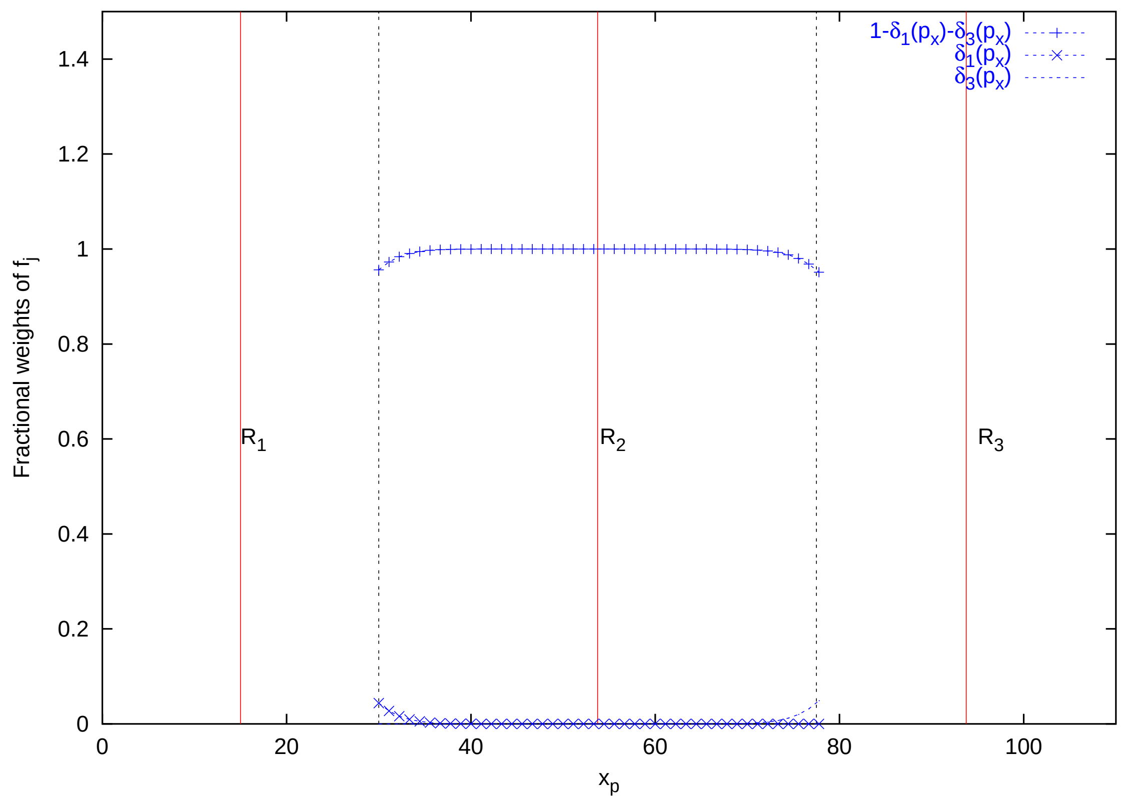

Figure 4.6: Fractional weights of f_1, f_2 and f_3 for a point in R_2 for the example from Figure 4.1 for $z_1 = 50$ and S=5.

which ranges between 0 and 1. We can take the inverse of this to calculate the uncertainty of p_i belonging to R_j as

$$\sigma_i = \frac{1}{1 - \delta_{j+1}(p) - \delta_{j-1}(p)} \quad .$$

Points that are closer to c_j have less uncertainty (close to 1) when they are passed for optimizing R_j, and the farther they go away from c_j, the more uncertain it is that they belong to model f_j.

4.3.4 Headstart from `M5′`

The initial guess greatly influences the performance of the optimization process. In addition to the heuristics described in Section 4.2.4, we used the output of `M5′`

to obtain good initial guesses. Before we begin the optimization process, we need to make three important decisions: choosing the attribute x which is the primary independent attribute, choosing the number of regimes n, and choosing the initial boundary surfaces. We can choose all three of these from the model tree that `M5′` generates. To determine the decision-making capability of an attribute in the model tree, we assign the following *decision-making score* to attribute i:

$$\theta_i = \sum_{j=1}^{\ell} c_{ij} 2^j \ , \tag{4.15}$$

where ℓ is the depth of the tree and c_{ij} is the number of decisions attribute i makes at level j. A decision taken at level j receives twice the score of a decision at level $j-1$. Thus this scoring scheme gives more importance to decisions at higher levels and also an opportunity for decisions at lower levels to outscore a higher level if the number of decisions are more. We then choose for x the attribute that has the highest decision-making score using

$$x = \arg\max_i \{\theta_i\} \ .$$

The total number of unique decisions on this attribute is set as the number of regimes n and the (sorted) values against which `M5′` makes those decisions are chosen as initial guesses for boundary surfaces (see also Section 4.2.4). We denote this method as `EMPRR-H`.

Notice that we use headstart only to select the number of regimes, the primary independent attribute x and the values against which `M5′` makes decisions. We are not directly using a solution from `M5′`. (Compare this with the quote from Salford-Systems (2001) in Section 4.2.3 where they recommend using the number of rules generated by CART as the threshold.)

Headstart from Functional trees could not be performed because the current executable obtained from João Gama does not give the multivariate decisions made at interior nodes. Once that facility is available, it would be interesting to devise a scoring scheme to choose x and the exploit the multivariate boundary surfaces.

4.3.5 Choice of Primary Independent Attribute x

Section 4.3.4 discussed a method to find the primary independent attribute x from the model tree structure obtained from M5′. In the absence of such information, there are several possible strategies to choose x. We can calculate the correlation between each independent attribute and the target variable using the methods discussed in Section 2.3. We can use this correlation as an indicator of the attribute's decision making capability.

Alternatively, EMPRR can be run $(1 + |\hat{z}|)$ times, each time choosing a different attribute as x. The best fit among these can be chosen as the final model. But this increases the time complexity by a factor of $|\hat{z}|$.

4.4 Implementation

EMPRR was implemented in C. The implementations of the following methods were adopted from Press et al. (1992):

1. Marquardt's method,
2. Spearman's rank coefficient calculation,
3. Kendall's τ coefficient calculation.

Several standard modes for the regime functions have been defined. They are: constant, linear, simple quadratic (only the constant, x_i and x_i^2 terms) and complex quadratic (all terms in a quadratic). Support for user defined functions is provided in our implementation. Any user-defined function can be used in a regime – all the user needs to provide are definitions of the function and its first order partial derivative with respect to each model parameter. This definition can be readily plugged into the main module. Moreover, each regime can have a different family of functions. For example, f_1 can be a linear function whereas f_2 can be a polynomial. Boundary surfaces are limited to linear functions in the current implementation.

The following arguments are required to be entered while invoking EMPRR:

- number of regimes,
- number of secondary independent attributes,
- name of file containing training data.

The files can be either comma-delimited or in ARFF format (Witten & Frank, 1999). The following arguments are optional during invocation:

- name of file containing test data,
- mode for regime functions,
  available options – constant, linear (default), simple quadratic, complex quadratic, user-defined
- mode for boundary surfaces,
  available options – constant, linear (default)
- specify initial values (default - see Section 4.2.4),
- use weighted M-function (default - true),
- use weighted regression in regimes (default - false),
- index of primary independent attribute x (default - first attribute),
- index of target variable (default - last attribute).

The following additional options are also available:

- finding the highest correlated attribute for use as x,
- filtering the data to remove attributes that are not at all correlated to the target variable,
- choice between Spearman's r_s and Kendall's τ for calculating correlations.

4.5 Problems and Exceptions

Our model suffers the disadvantages of nonlinear optimization techniques, some of which have been briefly mentioned in Section 2.2.5. Problems related to smothering and co-dependence are specific to the model at hand. Since we used linear models in all our experiments, we did not face these issues. But they are still potential problems in this approach when user-defined functions are used as f_i's. Apart from these, there are a few other practical problems and exceptions, which we explain in this section.

4.5.1 Insufficient Points in Regime

Situations may arise when the number of points n in a regime is less than the dimension of the attribute space. Then there exist multiple models that optimally fit the data in that regime. This also leads to a singular matrix (Section 2.2.5).

There are several ways to deal with this complication. We use Spearman's rank coefficient or Kendall's τ coefficient (see Section 2.3) to eliminate the independent attributes that are not highly correlated with the target variable. We then bring the dimensionality of the attribute space down to the number of points, so that we can obtain a unique model for that regime. This model may not generalize well since it can tend to fit the idiosyncrasies of the points in that regime. If, at the end of optimization, $n = 0$ for R_j, the regime is not regressed and b_j is set to b_{j-1} or b_{j+1} so that no points are assigned to R_j in future, e.g. from the test set.

Chapter 5

Experiments and Results

To evaluate `EMPRR`, we generated synthetic high-dimensional data that is piecewise-defined. The models that we used are given in Section 5.1.1. In Section 5.1.2, we report the results from comparing the performance of `EMPRR`'s variations described in Section 4.3 on these data sets. We also compared the performance of `EMPRR` with those of `M5`$'$ and `Rgtree`, an implementation of the functional tree algorithm from Gama (2002), on these data sets. We evaluated the performance of `EMPRR` using a set of benchmark problems (described in Section 5.2.1) and compared with `M5`$'$ and `Rgtree`. Section 5.2.2 summarizes the results.

5.1 Synthetic Data Sets

5.1.1 Models Used

To assess the influence of `EMPRR`'s variations, we generated 4 data sets using 4 piecewise models given below.

Model1

$$F(x,\hat{z}) = \begin{cases} 3 + x + z & \text{if } x < 19 + z \\ 39 - x + 3z & \text{if } 19 + z \le x < 32 - z \\ -81 + 3x - z & \text{if } 32 - z \le x \end{cases}, \tag{5.1}$$

2 continuous attributes and 150 samples corresponding to $x = \{1, 2, \ldots, 50\} \times z = \{1, 2, 3\}$. Samples are noise-free. The individual surfaces meet at the boundary surface.

Model2

$$F(x, \hat{z}) = \begin{cases} 2 + 5x + 2z & \text{if } x < 10 + 2z \\ 32 + 2x + 8z & \text{if } 10 + 2z \leq x < 30 - z \\ 181 - 3x + 3z & \text{if } 30 - z \leq x \end{cases}, \tag{5.2}$$

2 continuous attributes and 400 samples corresponding to $x = \{0.5, 1, 1.5, \ldots, 50\} \times z = \{0, 1, 2, 3\}$. Samples are noise-free. The individual surfaces meet at the boundary surface.

Model3

$$F(x, \hat{z}) = \begin{cases} 4 + 8x + 3z_2 - 8z_4 - 3z_6 + 4z_7 & \text{if } x < b_1(\hat{z}) \\ 5x - 2z_1 + 3z_2 + 8z_4 - 2z_5 + 2z_7 + 8z_8 & \text{if } b_1(\hat{z}) \leq x < b_2(\hat{z}) \\ 14 + 8x - 12z_1 + 4z_3 - 38z_4 + 3z_6 + 8z_8 & \text{if } b_2(\hat{z}) \leq x < b_3(\hat{z}) \\ 24 + 8x - 32z_1 + 4z_3 - 2z_5 + 15z_6 + 7z_7 + 6z_8 & \text{if } b_3(\hat{z}) \leq x < b_4(\hat{z}) \\ 4 + 9x - 12z_1 + 7z_3 + 8z_4 - 7z_5 + 4z_7 & \text{if } b_4(\hat{z}) \leq x \end{cases}, \tag{5.3}$$

where the boundaries are defined by

$$\begin{aligned} b_1(\hat{z}) &= 4 - 2z_2 + 2z_4 \\ b_2(\hat{z}) &= 8 + 1z_2 + 3z_4 \\ b_3(\hat{z}) &= 13 + 4z_2 + 7z_4 \\ b_4(\hat{z}) &= 18 + 6z_2 + 8z_4 \end{aligned}. \tag{5.4}$$

10 continuous attributes and 1000 samples with random sampling from $x \in [-50, 50]$ and $z_i \in [0, 10] \; \forall \, 1 \leq i \leq 9$. We also introduced normally distributed noise, with zero mean and unit variance, in the target variable y. The individual surfaces do not meet at the boundary surface.

Model4

$$F(x,\hat{z}) = \begin{cases} 4+8x+3z_2-8z_4-3z_6+4z_7 & \text{if } x < b_1(\hat{z}) \\ 5x-2z_1+3z_2+8z_4-2z_5+2z_7+8z_8 & \text{if } b_1(\hat{z}) \leq x < b_2(\hat{z}) \\ 14+8x-12z_1+4z_3-38z_4+3z_6+8z_8 & \text{if } b_2(\hat{z}) \leq x < b_3(\hat{z}) \\ 24+8x-32z_1+4z_3-2z_5+15z_6+7z_7+6z_8 & \text{if } b_3(\hat{z}) \leq x < b_4(\hat{z}) \\ 4+9x-12z_1+7z_3+8z_4-7z_5+4z_7 & \text{if } b_4(\hat{z}) \leq x \end{cases}, \tag{5.5}$$

where the boundaries are defined by

$$\begin{aligned} b_1(\hat{z}) &= 4-2z_2+2z_4 \\ b_2(\hat{z}) &= 8+1z_2+3z_4 \\ b_3(\hat{z}) &= 13+4z_2+7z_4 \\ b_4(\hat{z}) &= 4-2z_2+4z_4 \end{aligned}. \tag{5.6}$$

10 continuous attributes and 10000 samples with random sampling from $x \in [-50, 50]$ and $z_i \in [0, 10]$, $\forall\, 1 \leq i \leq 9$. Notice that *model4* differs from *model3* in number of samples and $b_4(\hat{z})$. $b_4(\hat{z})$ in *model4* is deliberately assigned so that it overlaps with $b_3(\hat{z})$ and $b_2(\hat{z})$. We also introduced normally distributed noise, with zero mean and unit variance, in the target variable y. The individual surfaces do not meet at the boundary surface.

5.1.2 Preliminary Investigation Experiments on Synthetic Data Sets

The simplest model of `EMPRR` is using an unscaled M-function. Our initial investigations showed that scaled M-functions (see Section 4.3.1) performed much better than unscaled M-functions. But no particular value of S worked for all data sets. For each data set, the algorithm found very good solutions with a different value of S. Therefore we modified `EMPRR` such that it tries a range of values for S for each data set. Currently the values are $\{100, 10, 1, 0.1, 0.01, 0.001\}$.

For 10-fold cross validation, we randomly divide the data set into 10 equal parts. Each part is held out in turn and the learning method is trained on the remaining nine-tenths. The model generated from this training is evaluated on the part that was held out. Thus we can calculate the training and test root mean square error (RMSE) values for each fold. To ensure that the same parts are used in 10-fold cross validation across all variations of `EMPRR`, we divided the data set into 10 equal parts and marked them as 10 test sets. The complement of each test set is its corresponding training set. Each variation is run on a training set to calculate the training RMSE and on the corresponding test set to calculate the test RMSE. These values are used to estimate the confidence intervals of the expected RMSE on unseen data from that variation.

We then tried all combinations of the other 3 variations of `EMPRR`, namely, uniform and weighted M-function, individual and weighted regression in regimes, and `EMPRR-H` and default start points. For comparison purposes, we also included two benchmark problems. The results from these experiments are given in Table 5.1. RMSE values are reported with 68% confidence intervals (one standard deviation from the mean over the 10 folds, denoted by "mean $\pm$ standard deviation") in Table 5.1.

These results show that using uniform M-function and weighted regression in regimes works best for synthetic data sets. In synthetic data sets, transition from weighted M-function to uniform M-function reduces RMSE and so does transition from individual regression to weighted regression. `EMPRR-H` seems to perform worse on the training data, but generalizes the model to better fit test data. But none of these differences is statistically significant. Almost all the 68% confidence intervals overlap with each other. Using uniform M-function with individual regimes appears to be a good choice for the 2 benchmark data sets, though the advantage is not statistically significant. Weighted regression does not perform better than the rest on benchmark data sets. This could be because the synthetic data sets were generated by a piecewise model, whereas the underlying models for the benchmark data sets are unknown. Therefore we chose uniform M-function with individual regimes for the benchmark data sets in experiments of Section 5.2.2. Also, headstart from `M5`$'$ provided the best choice for x in all the synthetic data sets. So we used headstart from `M5`$'$ to choose x for later experiments.

Table 5.1: 68% confidence intervals of RMSE values from variations of `EMPRR` on 2 synthetic and 2 benchmark data sets.

Weighted M-function, individual regression

model	training set RMSE	test set RMSE
model1H	0.9461 ± 0.3246	1.0914 ± 0.7057
model1	0.2117 ± 0.164	0.2556 ± 0.3874
model2H	1.1818 ± 0.6293	1.1526 ± 0.5633
model2	3.2821 ± 0.2055	3.3598 ± 0.6842
machineH	32.4743 ± 2.1893	60.9612 ± 36.2107
machine	32.5461 ± 3.6599	62.1526 ± 46.5792
priceH	1355.7167 ± 229.9051	4387.7929 ± 4584.774
price	1423.9444 ± 179.1155	3785.0005 ± 3582.9012

Uniform M-function, individual regression

model	training set RMSE	test set RMSE
model1H	0.6927 ± 0.3977	0.8495 ± 0.6506
model1	0.5077 ± 0.1975	0.6292 ± 0.5339
model2H	0.9563 ± 0.2941	1.0505 ± 0.5814
model2	0.7346 ± 0.8399	0.7120 ± 1.3189
machineH	33.8419 ± 2.1439	49.9162 ± 30.1046
machine	33.6002 ± 5.03	978.1572 ± 2919.7835
priceH	1336.9474 ± 192.6269	4718.5608 ± 4616.5388
price	1482.1872 ± 162.0408	8456.2149 ± 10991.0192

Uniform M-function, weighted regression

model	training set RMSE	test set RMSE
model1H	0.5931 ± 0.3596	0.8985 ± 0.7385
model1	0.5077 ± 0.1975	0.6288 ± 0.5335
model2H	0.7506 ± 0.4096	1.0294 ± 0.7127
model2	0.2964 ± 0.3043	0.4581 ± 0.6466
machineH	41.3861 ± 3.6763	64.8823 ± 37.5157
machine	43.1138 ± 5.6562	74.3383 ± 45.0165
priceH	1652.7911 ± 101.8976	3580.9441 ± 1470.9248
price	1701.7738 ± 115.6675	3631.4039 ± 1962.8494

A model name followed by 'H' denotes `EMPRR-H`.

5.1.3 Performance Comparison Experiments on Synthetic Data Sets

Since we are comparing our results to those from `M5′` and `Rgtree`, a brief discussion of their *modus operandi* is in order.

`M5′` is an implementation of the model tree technique in Java. The implementation along with the source is freely available from the Weka Machine Learning Project's web page[1]. `M5′` takes as arguments the names of the files containing the training data and test data. The specific formats that `M5′` can recognize are discussed in Witten and Frank (1999). If test data is not provided, it uses the training data to perform a 10-fold cross validation. The data is divided randomly into ten parts. Each part is held out in turn and the learning scheme trained on the remaining nine-tenths; then its error rate is calculated on the holdout set. Finally the ten error estimates are averaged to yield an overall error estimate. It also has an optional argument called "pruning factor" (default is set to 2) for specifying the pruning level of the final model tree. The higher the value is, the more the tree is pruned after being built. Increasing the pruning factor generally (but not necessarily) reduces the size of the final tree. Nevertheless, there is no direct relation between the pruning factor and the number of leaves. Hence it is not necessarily possible to obtain a model tree with a *specific* number of leaves. The impact of this behavior will become clear later in Section 5.2.2.

`Rgtree` is a Linux executable obtained from João Gama and takes as argument the name of the model. For a model named *name*, it assumes that there exist 3 files: `name.domain`, `name.data` and `name.test`. The `.domain` file provides details about the attributes and `.data` and `.test` files contain training and test data samples, respectively. The data sample files contain one sample per line with comma-delimited entries for attributes.

We ran `Rgtree` and `M5′` on the synthetic data sets. Model tree structures obtained from `M5′` were then analyzed to find start parameters in terms of primary independent attribute, number of regimes and good guess for initial boundary surfaces (see Section 4.3.4). These start parameters were then fed to `EMPRR` for the corresponding

[1]`http://www.cs.waikato.ac.nz/~ml/weka/index.html`

Table 5.2: 68% confidence intervals of RMSE values from performance comparison experiments on synthetic data sets.

Model	**M5D**		**RGD**		**EMH**	
Training data						
model1	2.8359	± 0.0913	2.4667	± 0.1578	**0.6927**	**± 0.3977**
model2	4.3626	± 0.0863	5.9767	± 0.3403	**0.9564**	**± 0.2942**
model3	37.6909	± 3.0437	33.3751	± 1.5461	33.1963	± 3.1723
model4	18.0527	± 0.5899	17.7178	± 0.3802	*23.1932*	± *2.8689*
Test data						
model1	3.1364	± 0.4939	2.4666	± 0.1578	**0.8849**	± **0.6212**
model2	4.4266	± 0.4994	5.9767	± 0.3403	**0.9735**	**± 0.6118**
model3	44.8143	± 5.0992	44.6639	± 6.1463	43.2325	± 9.2488
model4	21.8027	± 4.1490	24.4420	± 3.4887	23.9805	± 3.8475
Avg. Rules	30.3750		134.9750		9.6667	

M5D - `M5′`, RGD - `Rgtree`, EMH - `EMPRR-H`.

data sets. We used linear mode in both boundary surfaces and regime functions. The results are presented in Table 5.2. We also report the average number of rules generated by each method for a data set. In `EMPRR` this is the number of regimes, and in `M5′` and `Rgtree` this is the number of leaves of the final tree. We can clearly see that `EMPRR` outperforms `M5′` and `Rgtree` on these data sets both on the training examples and the test examples. Moreover, it uses only a fraction of the rules that `M5′` or `Rgtree` uses. Thus we conclude that `EMPRR` can generate compact, accurate models of these data sets. An interesting observation is that `Rgtree` uses significantly more number of rules to represent linear data that are linearly separable. Both `Rgtree` and `EMPRR` fail to fit the exact underlying model for *model3* which has 10 independent attributes. `EMPRR` fails to fit *model4* well since the boundary surfaces overlap in the underlying model.

5.2 Benchmark Problems

5.2.1 Data Sets Used

Data sets for the benchmark problems, some of which have been used in Quinlan (1992); Wang and Witten (1997); Gama (2002), were obtained from the *Repository of Regression Problems at LIACC*[2]. Though this repository is only a collection of data from other sources, we chose this because the data sets have been pruned according to our specifications – nominal attributes and samples with missing attributes have been removed. Except the 2-D planes data set, which was generated using a piecewise hypothesis, models defining these data sets are not necessarily piecewise. The following description of these data sets is quoted from the repository. The properties of these data sets are also summarized in Table 5.3.

Ailerons This data set addresses a control problem, namely flying an F16 aircraft. The attributes describe the status of the aeroplane, while the goal is to predict the control action on the ailerons of the aircraft. 13750 cases with 40 continuous attributes.

Delta Ailerons This data set is also obtained from the task of controlling the ailerons of an F16 aircraft, although the target variable and attributes are different from the ailerons domain. The target variable here is a variation instead of an absolute value, and there is some pre-selection of the attributes. 7129 cases with 6 continuous attributes.

Elevators This data set is also obtained from the task of controlling an F16 aircraft, although the target variable and attributes are different from the ailerons domain. In this case the target variable is related to an action taken on the elevators of the aircraft. 16559 cases with 18 continuous attributes.

Delta Elevators This data set is also obtained from the task of controlling the elevators of an F16 aircraft, although the target variable and attributes are

[2]`http://www.liacc.up.pt/~ltorgo/Regression/DataSets.html`

Table 5.3: Summary of the benchmark data sets.

Data Set	**Code**	**Examples**	**Attributes**
Ailerons	ailerons	13750	40
Delta Ailerons	delta-ailerons	7129	6
Elevators	elevators	16599	18
Delta Elevators	delta-elevators	9517	6
2-D Planes	cart-delve	40768	10
Auto MPG	auto	8192	7
Auto Price	price	159	15
Bank Domain	bank8FM	8192	8
Boston Housing	housing	506	13
California Housing	cal-housing	20640	8
Census	house-8L	22784	8
Computer Activity	cpu-small	8192	8
Diabetes	diabetes	43	2
Kinematics	kin8nm	8192	8
Machine CPU	machine	209	6
Pumadyn Dynamics	puma8NH	8192	8
Pyrimidines	pyrim	74	27
Stocks	stock	950	10
Triazines	triazines	186	60
Wisconsin	r-wpbc	194	32

different from the elevators domain. The target variable here is a variation instead of an absolute value, and there is some pre-selection of the attributes. 9517 cases with 6 continuous attributes.

2-D planes This is an artificial data set similar (but not exactly equal) to the one described in Breiman et al., (1984, page 238). The cases are generated using the following method: Generate the values of the 10 attributes independently using probabilities

$$\begin{aligned} P(X_1 = -1) = P(X_1 = 1) &= \frac{1}{2} \\ P(X_m = -1) = P(X_m = 0) = P(X_m = 1) &= \frac{1}{3}, m = 2, \ldots, 10. \end{aligned}$$

Obtain the value of the target variable Y using the rule

$$\begin{aligned} Y &= 3 + 3X_2 + 2X_3 + X_4 + \sigma(0,1) \text{ if } X_1 = 1 \\ Y &= -3 + 3X_5 + 2X_6 + X_7 + \sigma(0,1) \text{ if } X_1 = -1, \end{aligned}$$

where $\sigma(0,1)$ is a normal deviation with zero mean and unit variance. 40768 cases with 10 continuous attributes.

Auto Price This data set consists of three types of entities

- the specification of an auto in terms of various characteristics
- its assigned insurance risk rating
- its normalized losses in use as compared to other cars.

The second rating corresponds to the degree to which the auto is more risky than its price indicates. Cars are initially assigned a risk factor symbol associated with its price. Then, if it is more risky (or less), this symbol is adjusted by moving it up (or down) the scale. A value of +3 indicates that the auto is risky, −3 that it is probably pretty safe. The third factor is the relative average loss payment per insured vehicle year. This value is normalized for all autos within a particular size classification, and represents the average loss per car per year. 159 cases with 14 continuous attributes and 1 nominal attribute[3].

Bank domain A family of data sets synthetically generated from a simulation of how bank-customers choose their banks. Tasks are based on predicting the fraction of bank customers who leave the bank because of full queues. The bank family of data sets are generated from a simplistic simulator, which simulates the queues in a series of banks. Customers come from several residential areas, choose their preferred bank depending on distances and have tasks of varying complexity, and various levels of patience. Each bank has several queues, that open and close according to demand. The tellers have various efficiencies, and customers

[3]We considered the nominal attribute of this data set as a continuous attribute since its enumerated values were integral.

may change queue, if their patience *expires*. The objective is to predict the rate of rejections, i.e. the fraction of customers that are turned away from the bank because all the open tellers have full queues. 8192 cases with 8 continuous attributes.

Boston Housing This data set concerns the task of predicting housing values in areas of Boston. The attributes are:

CRIM – per capita crime rate by town

ZN – proportion of residential land zoned for lots over 25,000 sq.ft.

INDUS – proportion of non-retail business acres per town

CHAS – Charles River dummy variable (1 if tract bounds river; 0 otherwise)

NOX – nitric oxides concentration (parts per 10 million)

RM – average number of rooms per dwelling

AGE – proportion of owner-occupied units built prior to 1940

DIS – weighted distances to five Boston employment centres

RAD – index of accessibility to radial highways

TAX – full-value property-tax rate per \$10,000

PTRATIO – pupil-teacher ratio by town

B – $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town

LSTAT – lower status of the population

MEDV – Median value of owner-occupied homes in \$1000's.

506 cases with 13 continuous attributes.

California Housing This data set contains information on block groups in California from the 1990 Census. The target variable is median house value. Independent attributes are median income, housing median age, total rooms, total bedrooms, population, households, latitude, and longitude. 20640 cases with 8 continuous attributes.

Census This database was designed on the basis of data provided by US Census Bureau. The data were collected as part of the 1990 US census. These concern with predicting the median price of the house in the region based on demographic composition and a state of housing market in the region. 22784 cases with 8 continuous attributes.

Computer Activity The Computer Activity databases are a collection of computer systems activity measures. The data was collected from a Sun Sparcstation 20/712 with 128 Mbytes of memory running in a multi-user university department. Users would typically be doing a large variety of tasks ranging from accessing the internet, editing files or running very CPU-bound programs. The data was collected continuously on two separate occasions. On both occasions, system activity was gathered every 5 seconds. The final data set is taken from both occasions with equal numbers of observations coming from each collection epoch. The goal is to predict `usr`, the portion of time that CPUs run in user mode, using a restricted number of attributes from the original data. 8192 cases with 8 continuous attributes.

Diabetes This data set concerns the study of the factors affecting patterns of *insulin-dependent diabetes mellitus* in children. The objective is to investigate the dependence of the level of serum C-peptide on various other factors in order to understand the patterns of residual insulin secretion. The response measurement is the logarithm of C-peptide concentration ($pmol/ml$) at the diagnosis, and the predictor measurements are age and base deficit, a measure of acidity. 43 cases with 2 continuous attributes.

Kinematics of Robot Arm This data set is concerned with the forward kinematics of an 8 link robot arm. Among the existing variants of kinematics data set, this is variant $8nm$, which is known to be highly non-linear and medium noisy. 8192 cases with 8 continuous attributes.

Machine-CPU The problem concerns Relative CPU Performance Data. The attributes are

MYCT - machine cycle time in nanoseconds (integer)

MMIN - minimum main memory in kilobytes (integer)

MMAX - maximum main memory in kilobytes (integer)

CACH - cache memory in kilobytes (integer)

CHMIN - minimum channels in units (integer)

CHMAX - maximum channels in units (integer)

PRP - published relative performance (integer) (target variable).

209 cases with 6 continuous attributes.

Pumadyn Dynamics This is a family of data sets synthetically generated from a realistic simulation of the dynamics of a Unimation Puma 560 robot arm. The task in this data set is to predict the angular acceleration of one of the robot arm's links. The inputs include angular positions, velocities and torques of the robot arm. The family has been specifically generated for the delve environment and so the individual data sets span the corners of a cube whose dimensions represent

- number of inputs (8 or 32).
- degree of non-linearity (fairly linear or non-linear)
- amount of noise in the output (moderate or high).

8192 cases with 8 continuous attributes.

Pyrimidines The task consists of learning Quantitative Structure Activity Relationships (QSARs). The data are described in King et al. (1992). 74 cases with 27 continuous attributes.

Stocks Daily stock prices from January 1988 through October 1991, for ten aerospace companies. 950 cases with 10 continuous attributes.

Triazines The problem is to learn a regression equation/rule/tree to predict the activity from the descriptive structural attributes. The data and methodology is described in detail in King et al. (1995); King et al. (1994). 186 cases with 60 continuous attributes.

Wisconsin Breast Cancer Database Each record represents follow-up data for one breast cancer case. These are consecutive patients seen by a particular physician since 1984, and include only those cases exhibiting invasive breast cancer and no evidence of distant metastases at the time of diagnosis. The goal is to predict the time to recur.

5.2.2 Performance Comparison Experiments on Benchmark Data Sets

We performed 10-fold cross validation experiments on the benchmark data sets using the same procedure as on synthetic data sets. Since our headstart strategy determines the one attribute with maximum decision making capability, the number of regimes that is fed to `EMPRR` is generally less than the number of rules (leaves) that `M5`$'$ generates. In certain cases, it is less by more than an order of magnitude. (The number of leaves in a functional tree generated for the data was generally higher than the number of rules from `M5`$'$.) In such cases, `EMPRR` is restricted by the number of regimes. Therefore we included two more experiments for each data set. In the first additional experiment, we modeled the data using `EMPRR`, with a large number of regimes. We call this unrestricted `EMPRR` (`EMPRR-U`). The initial guesses for boundary surfaces are assigned by dividing the x-dimension into partitions containing an equal number of points (see Section 4.2.4). We started with a small number of regimes and tried incremental values. We stopped when `EMPRR` performed significantly better than the other methods, or the number of regimes became too high[4] for `EMPRR`, or when the number of regimes was larger than the number of rules from `M5`$'$ as well as

[4]For a very large number of regimes, the number of parameters in Equation (4.10) becomes very large. Being a nonlinear optimization technique, `EMPRR` takes a long time to perform Step 3 in Figure 4.2. Since the overall procedure makes numerous calls to this step, it takes a very long time for `EMPRR` to converge. Moreover, increasing the number of parameters increases the chance of getting trapped in a local minimum (Section 2.2.5). We can reduce the running time if we can reduce the number of parameters by independently optimizing each boundary's parameters.

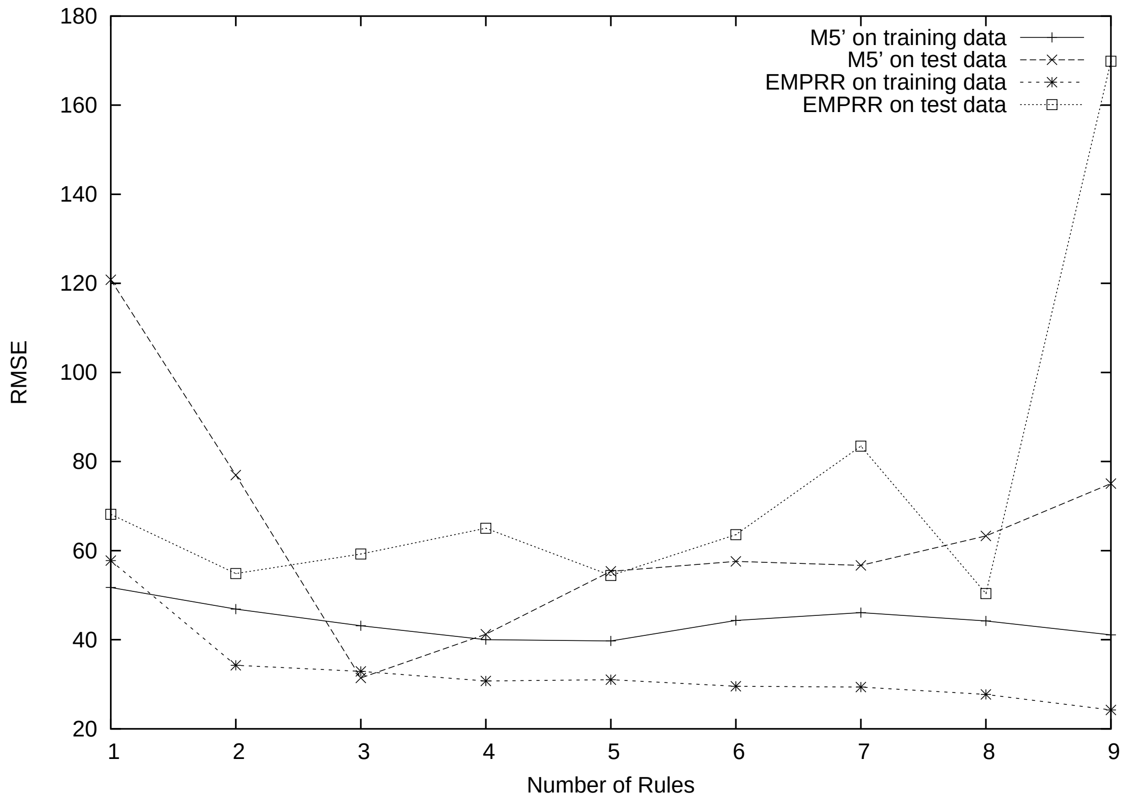

Figure 5.1: Learning curves for `M5′` and `EMPRR` on the *machine* data set. Each point in the curves is averaged over the 10 folds.

from `Rgtree`. In most of the benchmark data sets, increasing the number of regimes resulted in better fits, up to a point where `EMPRR` overfits. A sample learning curve of `EMPRR` on the *machine* data set with respect to the number of regimes is given in Figure 5.1. For comparison purposes, the learning curve of `M5′` on the same data set is also given. Each point in the curves is averaged over the 10 folds. In the second additional experiment, we restricted the number of rules generated by `M5′` by modifying the pruning factor[5]. We tried to bring the rules down to the number of regimes that was used by `EMPRR-H`. Let this number be r. As mentioned earlier in this section, there is no direct relation between the pruning factor and the number of

[5]As mentioned in Section 4.3.4, this cannot be performed for the current implementation of `Rgtree`.

rules, and getting a specific number of rules was not always possible. When it was not possible to generate a model with r rules, we took two trees with sizes $r + \epsilon_1$ and $r - \epsilon_2$ for some minimum $\epsilon_1 > 0$ and $\epsilon_2 > 0$ and averaged the statistics. We call this restricted M5′.

Table 5.4 lists the RMSE values obtained from the 10-fold cross validation experiments of running M5′, restricted M5′, Rgtree and EMPRR on the training sets of 20 synthetic data sets. Table 5.5 lists the RMSE values for the test sets. RMSE values are provided with 68% confidence intervals (one standard deviation from the arithmetic mean over the 10 folds).

Unrestricted EMPRR (EMPRR-U) was run only on 14 data sets since the number of rules in the rest were too large (see Footnote 4). Table 5.6 summarizes the statistics of the results from running M5′, restricted M5′, Rgtree and EMPRR on all 20 benchmark data sets. Table 5.7 summarizes the statistics of running M5′, restricted M5′, Rgtree, EMPRR and EMPRR-U on 14 data sets.

Each column in Tables 5.4, 5.5, 5.6 and 5.7 represents results from one method. Each data set has two corresponding rows in each of Tables 5.4 and 5.5. The first row lists the RMSE values with 68% confidence intervals and the second row reports the average number of rules generated by the method on the data set. The first six rows summarize the results from Tables 5.4 and 5.5. The first and fourth rows list the geometric mean RMSE value for training and test data from each method, respectively. For example, geometric mean RMSE for training data from M5′ is the geometric mean of the mean RMSE values in column 2 (under M5D) in Table 5.4. Geometric mean was chosen over the arithmetic mean because the individual RMSE values differ from each other by several orders of magnitude (ranging from ≈ 32980 to 0.000235). Arithmetic mean will be highly sensitive to higher RMSE values and highly insensitive to lower RMSE values, and hence will not represent the overall goodness of the method. The second and fifth rows list the average rank of each method on the training and test data, respectively. For example, average rank of M5′ on training data is calculated as follows: For each data set, the rank of M5′'s mean RMSE among all methods (an integer between 1 and the number of methods, inclusive) is calculated for the training set. The average of these individual ranks is

Table 5.4: 68% confidence intervals of training RMSE values for benchmark data sets.

Model	**M5D**		**M5R**		**RGD**		**EMH**		**EMU**	
ailerons	5.5e-4	± 8.357e-4	2.35e-4	± 8.182e-5	1.428e-4	± 2.7e-6	4.373e-4	± 2.865e-4	N/A	
	10.6		3.15		168.5		3.1		N/A	
auto	2.6033	± 0.1323	2.7849	± 0.0534	2.3112	± 0.0701	2.6859	± 0.0564	**1.7724**	**± 0.0927**
	6.8		3.3		20.8		2.9		20	
bank8FM	0.0292	± 2.789e-4	0.0314	± 2.846e-4	**0.0283**	**± 4.555e-4**	0.0307	± 7.432e-4	N/A	
	32.3		8.9		76.4		8.9		N/A	
cal-housing	4.901e+4	± 7.934e+2	5.919e+4	± 4.248e+2	**4.27e+4**	**± 3.577e+2**	*6.357e+4*	± *4.165e+2*	N/A	
	236		25.6		1631.3		25.5		N/A	
cart-delve	0.9968	± 0.0010	0.9968	± 0.0010	*1.0012*	± *0.0024*	0.9967	± 0.0010	0.9967	± 0.0010
	2		2		25.1		2		2	
cpu-small	2.8531	± 0.0429	3.1177	± 0.0536	2.8639	± 0.0396	3.638	± 0.9603	N/A	
	55.5		7.5		287.6		8		N/A	
delta-ailerons	2.0e-4	± 0.0	2.0e-4	± 0.0	1.667e-4	± 1.16e-6	1.64e-4	± 1.826e-6	**1.6e-4**	**± 1.054e-6**
	31.2		7		3		8.1		30	
delta-elevators	0.0014	± 0.0	0.0014	± 0.0	0.0014	± 0.0	0.0014	± 0.0	0.0014	± 0.0
	6.2		2		2		2.8		6	
diabetes	0.5404	± 0.0522	0.5404	± 0.0522	*0.7441*	± *0.0321*	0.5122	± 0.0825	**0.3815**	**± 0.0238**
	1.6		1.6		4.5		1.6		4	
elevators	0.0020	± 5.164e-5	0.0023	± 5.986e-5	0.0021	± 2.402e-5	0.0022	± 1.075e-4	0.0024	± 3.162e-5
	41.3		5.85		35.5		5.8		35	
house-8L	2.888e+4	± 2.703e+2	3.182e+4	± 2.791e+2	**2.537e+4**	**± 5.23e+2**	*3.298e+4*	± *4.576e+2*	N/A	
	142.2		20.5		2033.1		20.1		N/A	
housing	2.7056	± 0.1236	3.3455	± 0.4428	2.5585	± 0.1259	3.0161	± 0.245	**0.9466**	**± 0.4457**
	14.2		4.55		32.5		4.3		32	
kin8nm	0.1422	± 0.0037	0.1816	± 0.0030	0.118	± 0.0022	0.1671	± 0.0106	0.1543	± 0.0045
	131.7		11.55		438.3		10.7		50	
machine	40.1394	± 2.6886	48.2297	± 4.6172	36.0979	± 1.6725	33.2264	± 1.5714	32.6421	± 2.0185
	3.7		2		11.3		2.1		11	
price	1.687e+3	± 89.4147	1.975e+3	± 1.073e+2	1.431e+3	± 1.184e+2	1.337e+3	± 1.926e+2	**5.913e+2**	**± 1.631e+2**
	7.1		3.6		7.4		3.6		7	
puma8NH	3.1171	± 0.014	3.2318	± 0.0488	2.5822	± 0.0399	3.3223	± 0.0906	N/A	
	30.7		13.4		718.2		13.5		N/A	
pyrim	0.0621	± 0.0108	0.0687	± 0.0124	0.0452	± 0.0046	**0.0208**	**± 0.0072**	**0.0162**	**± 0.0029**
	2.8		1.9		3		1.9		3	
r-wpbc	29.0826	± 1.4089	28.672	± 1.4969	21.7628	± 1.3455	23.6915	± 2.8478	17.2268	± 21.4757
	2.8		1.7		36.3		1.7		6	
stock	0.7124	± 0.0155	0.8827	± 0.0652	0.7015	± 0.0197	0.8499	± 0.0559	0.5489	± 0.0963
	43.9		12.85		48		12.6		48	
triazines	0.1028	± 0.0072	0.1236	± 0.0125	0.1128	± 0.0068	**0.0753**	**± 0.0097**	**0.0736**	**± 0.0101**
	3.8		2.15		4.1		2.1		4	

M5D - `M5′`, M5R - restricted `M5′`, RGD - `Rgtree`, EMH - `EMPRR` with headstart from `M5′`, EMU - unrestricted `EMPRR` with default starting points.

Table 5.5: 68% confidence intervals of test RMSE values for benchmark data sets.

Model	**M5D**		**M5R**		**RGD**		**EMH**		**EMU**	
ailerons	5.0e-4	± 7.149e-4	2.4e-4	± 9.661e-5	1.686e-4	± 9.252e-6	4.36e-4	± 2.797e-4	N/A	
	10.6		3.15		168.5		3.1		N/A	
auto	2.9423	± 0.3861	2.9356	± 0.4221	2.9314	± 0.5381	3.1294	± 0.5474	4.0832	± 1.7035
	6.8		3.3		20.8		2.9		20	
bank8FM	0.0304	± 4.625e-4	0.0321	± 8.099e-4	0.0302	± 8.395e-4	0.0315	± 9.052e-4	N/A	
	32.3		8.9		76.4		8.9		N/A	
cal-housing	5.502e+4	± 2.56e+3	5.818e+4	± 3.304e+3	5.535e+4	± 2.144e+3	2.376e+5	± 3.889e+5	N/A	
	236		25.6		1631.3		25.5		N/A	
cart-delve	0.997	± 0.0092	0.7961	± 8.822e-4	1.0075	± 0.0088	0.9972	± 0.0091	0.9972	± 0.0091
	2		2		25.1		2		2	
cpu-small	3.1019	± 0.2239	3.1994	± 0.2054	3.0268	± 0.1972	3.692	± 1.2343	N/A	
	55.5		7.5		287.6		8		N/A	
delta-ailerons	2.0e-4	± 0.0	2.0e-4	± 0.0	1.678e-4	± 8.954e-6	1.689e-4	± 8.386e-6	1.753e-4	± 1.476e-5
	31.2		7		3		8.1		30	
delta-elevators	0.0014	± 4.83e-5	0.0014	± 5.164e-5	0.0014	± 5.164e-5	0.0014	± 5.27e-5	0.0014	± 5.164e-5
	6.2		2		2		2.8		6	
diabetes	0.6452	± 0.2237	0.4262	± 0.0447	1.0053	± 0.1394	0.6055	± 0.186	0.7164	± 0.3707
	1.6		1.6		4.5		1.6		4	
elevators	0.0021	± 8.233e-5	0.0023	± 7.379e-5	0.0022	± 7.859e-5	0.0022	± 1.418e-4	0.0024	± 9.487e-5
	41.3		5.85		35.5		5.8		35	
house-8L	3.18e+4	± 1.445e+3	3.412e+4	± 1.261e+3	3.179e+4	± 1.678e+3	3.477e+4	± 2.857e+3	N/A	
	142.2		20.5		2033.1		20.1		N/A	
housing	3.508	± 0.8161	4.3232	± 0.5151	3.9048	± 0.8367	4.3113	± 1.2948	*53.3326*	*± 55.8897*
	14.2		4.55		32.5		4.3		32	
kin8nm	0.16	± 0.0038	0.1864	± 0.0043	0.1522	± 0.0046	0.1685	± 0.0080	0.1665	± 0.0099
	131.7		11.55		438.3		10.7		50	
machine	55.7781	± 40.8016	67.2026	± 37.045	56.1074	± 34.2541	52.873	± 27.0995	53.3255	± 24.5848
	3.7		2		11.3		2.1		11	
price	2.297e+3	± 8.821e+2	2.465e+3	± 7.998e+2	2.432e+3	± 7.918e+2	4.719e+3	± 4.617e+3	*1.968e+4*	*± 2.213e+4*
	7.1		3.6		7.4		3.6		7	
puma8NH	3.1982	± 0.0686	3.2361	± 0.1403	3.3582	± 0.0542	3.3595	± 0.1079	N/A	
	30.7		13.4		718.2		13.5		N/A	
pyrim	0.0807	± 0.0779	0.085	± 0.0749	0.083	± 0.0749	0.1329	± 0.0921	*2.6204*	*± 5.9952*
	2.8		1.9		3		1.9		3	
r-wpbc	36.8174	± 7.3609	36.0938	± 6.988	35.8112	± 5.4599	41.6148	± 14.2049	*2.705e+2*	*± 1.17e+2*
	2.8		1.7		36.3		1.7		6	
stock	0.8613	± 0.1258	1.0003	± 0.1539	1.0266	± 0.2543	1.1309	± 0.1966	1.8497	± 0.293
	43.9		12.85		48		12.6		48	
triazines	0.1248	± 0.032	0.1405	± 0.0327	0.147	± 0.0348	0.2705	± 0.2579	0.3036	± 0.1111
	3.8		2.15		4.1		2.1		4	

M5D - `M5′`, M5R - restricted `M5′`, RGD - `Rgtree`, EMH - `EMPRR` with headstart from `M5′`, EMU - unrestricted `EMPRR` with default starting points.

Table 5.6: Performance comparison on 20 benchmark data sets (without unrestricted `EMPRR`).

	Statistic	**M5D**	**M5R**	**RGD**	**EMH**
Training Data	Geo. Mean	0.9713	1.0275	0.8451	0.9157
	Avg. Rank	2.2500	3.2500	1.7000	2.3500
	Avg. Ratio	1.1174	1.1973	0.9837	1.0000
Test Data	Geo. Mean	1.1033	1.0954	1.0879	1.3580
	Avg. Rank	1.8500	2.700	2.000	3.3500
	Avg. Ratio	0.8601	0.8548	0.8612	1.0000
Both	Avg. Rules	40.3200	6.4948	279.3450	7.0650

M5D - `M5′`, M5R - restricted `M5′`, RGD - `Rgtree`, EMH - `EMPRR` with headstart from `M5′`.

Table 5.7: Performance comparison on 14 benchmark data sets (including unrestricted `EMPRR`).

	Statistic	**M5D**	**M5R**	**RGD**	**EMH**	**EMU**
Training Data	Geo. Mean	0.4008	0.4458	0.3745	0.3550	0.2711
	Avg. Rank	3.0714	4.2143	2.7143	2.5714	1.4286
	Avg. Ratio	1.1980	1.3327	1.1015	1.0000	0.8026
Test Data	Geo. Mean	0.4453	0.4661	0.4725	0.5389	1.1584
	Avg. Rank	1.9231	3.0000	2.2308	3.3846	4.1538
	Avg. Ratio	0.8579	0.8979	0.9154	1.0000	4.0858
Both	Avg. Rules	21.3643	4.4321	47.9857	4.4429	18.4286

M5D - `M5′`, M5R - restricted `M5′`, RGD - `Rgtree`, EMH - `EMPRR` with headstart from `M5′`.

the average rank of `M5′` over all training data. The third and sixth rows list the average ratio of each method's RMSE against `EMPRR-H`. For example the average RMSE ratio of `M5′` on training data is the average of the individual ratios of `M5′`'s mean RMSE and `EMPRR-H`'s mean RMSE on each data set. This quantifies how much better or worse the RMSE values from a method are, when compared to `EMPRR-H`. Finally, the seventh row lists the average number of rules across all data sets.

On the training data, there does not seem to be statistically significant difference between `M5′` and restricted `M5′` on most of the data sets (with 68% confidence level, they are not significantly different on almost all data sets). `Rgtree` has a statistically significant better fit (with 68% confidence level) on *bank8FM*, *cal-housing* and *house-8L* data sets. `EMPRR-U` has a statistically significant better fit (with 68% confidence level) on *auto*, *delta-ailerons*, *diabetes*, *housing*, *price*, *pyrim* and *triazines*. EMPRR-H itself has a better fit than M5′, restricted M5′ and Rgtree on *delta-ailerons*, *pyrim* and *triazines*. The entries in bold type and italicized type in Tables 5.4 and 5.5 are statistically significant better and worse fits than all the other methods, respectively, at 68% confidence level. On the test data, M5′, restricted M5′ and Rgtree do not seem to have statistically significant difference on almost all data sets. Restricted M5′ performs significantly better on *diabetes* and so does M5′ on *puma8NH*. Rgtree performs better on *ailerons* and *delta-ailerons*, two closely related data sets. EMPRR-U performs unpredictably on *housing*, *price*, *pyrim* and *r_wpbc*. It is interesting to note that EMPRR-U performed significantly better on the corresponding training data sets.

While comparing methods excluding EMPRR-U on 20 data sets (Table 5.6), Rgtree gives the best results on training sets in terms of geometric mean RMSE, average rank and average RMSE ratio. EMPRR-H stands third in terms of average rank. M5′, which stands second in terms of average rank, leads only by a value of 0.1. But EMPRR-H has a better average RMSE ratio and geometric mean RMSE than M5′ which shows that it finds better solutions (in terms of RMSE) than M5′ in general. Nevertheless, EMPRR-H uses only a fraction of the number of rules used by M5′ or Rgtree. This reiterates our earlier inference that EMPRR generates accurate and compact models. On test data, M5′, restricted M5′ and Rgtree perform almost equally well and also better than EMPRR-H. EMPRR-H, which performed better than restricted M5′ on training sets, performs worse than restricted M5′ and fails to generalize the model for unseen data.

While comparing methods including EMPRR-U on 14 data sets (Table 5.7), EMPRR-U seems to outperform all the other methods on the training sets in terms of geometric mean RMSE, average rank and average RMSE ratio. Note also that EMPRR-H performs better than M5′ and almost as well as Rgtree in terms of average rank. When we compare the RMSE ratios, EMPRR-H performs better than both M5′ and Rgtree. The reason could be that the 6 data sets that were removed in this comparison had a large number of rules, as mentioned earlier. This can also be seen by comparing the average rules for each method in Tables 5.6 and 5.7. The average rules used by M5′ and Rgtree are reduced drastically by removing these 6 data sets. These results show that EMPRR can produce a compact representation for data that *can* be compactly represented. We suspect that it may not be possible to compactly represent the 6 data sets that were removed (using linear models) due to the fact that M5′ and Rgtree found too many rules for those 6 data sets and EMPRR failed to find a good compact representation. It would be interesting to try to fit them using quadratic or higher-order models. On test data, M5′ seems to be the best performer, followed by restricted M5′ and Rgtree. EMPRR and EMPRR-H perform worse on test data when compared to the other methods. It is interesting to note that Rgtree performs better than EMPRR-H on test data, even though it uses more rules, which generally is a sign of over-fitting the training data.

Being a regression approach, EMPRR is designed to fit the training data as well as possible by minimizing the sum of squared errors. Under the assumption that the observed training values are generated by adding random noise to the true target value, where this random noise is drawn independently for each example from a Normal distribution with zero mean, the maximum likelihood hypothesis is the one that minimizes the sum of the squared errors between the observed training values and the model predictions (Mitchell, 1997; Press et al., 1992). EMPRR performs that task very well, but does not necessarily generate a model that fits unseen data as well as M5′ and Rgtree do, especially if the training data contains noise. For example, in Tables 5.1 and 5.2 EMPRR performed equally well (compared to Tables 5.6 and 5.7) on both training and test sets for the synthetic data sets.

The running-time for EMPRR depends on the characteristics of the data set, the model that we are trying to fit to and the number of regimes. For example, running EMPRR on *house-8L* data set with approximately 20 regimes took 5 to 13 hours. Running it on auto (3 regimes), *diabetes* (2 regimes) and *machine* (2 regimes) took less than 1 minute, on *delta-ailerons* (8 regimes) and *delta-elevators* (3 regimes) took 7 to 15 minutes, on *cpu-small* (8 regimes) and *bank8FM* (9 regimes) took 30 to 50 minutes, on *elevators* (6 regimes) took 2 to 3 hours.

Chapter 6

Conclusions and Future Work

We developed a novel EM-based algorithm, `EMPRR`, that can perform piecewise regression on high-dimensional data. We implemented the algorithm in `C` and evaluated its performance on synthetic high-dimensional piecewise data that we generated. We also compared its performance with two regression tree learning methods, model tree and functional tree, on 20 benchmark data sets. We found that `EMPRR` performs competitively with these two methods on these data sets. It finds better and compact models to fit the data. It fits them so well that it performs slightly worse on unseen data. The results given here are specific to the data sets we used and the choice of Marquardt's method as the nonlinear optimization technique in the implementation.

`EMPRR` is in an earlier stage and several enhancements are possible. In terms of handling data, we observe that `EMPRR` cannot handle nominal attributes and missing values. CART (Breiman et al., 1984) and `M5`$'$ handle nominal attributes by replacing an enumerated attribute with k values by $k-1$ synthetic binary attributes. CART uses "surrogate splitting" to handle situations when an attribute that is involved in a split is missing. It essentially finds another attribute to split in place of the original attribute. `M5`$'$ uses the class value as the surrogate attribute. These techniques can be extended to `EMPRR` also, probably with a few modifications.

Though the current implementation of `EMPRR` can handle nonlinear and user-defined functions in regimes, our experiments did not use this facility. This was to facilitate direct comparison between the performances of `EMPRR`, `M5`$'$ and `Rgtree`.

Further experiments using non-linear and more complex functions would be interesting, especially on the same data sets for which we tried linear models.

Note that in Equation (4.13) the value of m_B is generally large since Equation (4.8) is defined in terms of β_i, $1 \leq i \leq n-1$, and we need to fit all the parameters in all boundary surfaces. Optimizing all the boundary surfaces together thus makes it slower when n and D increase. As we mentioned in Section 5.2.2, for some data sets with many attributes and data points, running EMPRR with too many regimes took very long, and some did not finish even after a couple of days. If we can develop an EM formulation that optimizes the boundary surfaces independently, thus requiring only to fit β_i during one call to the Marquardt method, the time complexity can be improved significantly.

In Section 2.2.5, we discussed problems associated with nonlinear optimization techniques. One such difficulty was encountering a singular Hessian matrix. Inverse Hessian methods such as Marquardt's method will fail to solve Equation (2.18) and they cannot diagnose the problem. There are a set of techniques, known as *singular value decomposition*, which will diagnose the problem and also give a numerical answer that satisfies the simultaneous equations. There are several robust variations of the basic technique. MARS (Friedman, 1991) uses one such technique known as the *Cholesky decomposition*. Using such methods in the optimization procedure might improve the performance of EMPRR, especially when user-defined functions are used. Even in linear mode, the search for the optimal solution may stop if the current solution leads to a singular Hessian matrix. Changing λ will help in moving to some other solution, but in extreme cases the search may stop there.

An interesting extension would be to develop a proper EM formulation where one step calculates the expected parameters and the other maximizes the likelihood. Though our algorithm is analogous to the EM-algorithm in general, a proper formulation will provide specific convergence properties.

As we mentioned in Section 5.1.2, different values of S work well for different data sets. We suspect that there should be some information about the value of S in the data. We can compare this to Marquardt's insight in assigning the value for the "constant" in Equation (2.19). If there is indeed a relation between S and the data, it would only make `EMPRR` run faster, since it does not have to try multiple values for S. Also, if there is an optimal value for S that lies outside the interval that `EMPRR` tests, `EMPRR`'s solution may be suboptimal. Therefore further investigation in this direction could be useful.

Bibliography

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, *39*, 1–38.

Friedman, J. H. (1991). Multivariate Adaptive Regression Splines. *Annals of Statistics*, *19*, 1–141.

Gama, J. (2002). An analysis of functional trees. *Proceedings of the Nineteenth International Conference on Machine Learning* (pp. 155–162). Morgan Kauffman.

Hudson, D. J. (1966). Fitting segmented curves whose join points have to be estimated. *Journal of the American Statistical Association*, *61*, 1097–1129.

King, R., Hirst, J., & Sternberg, M. (1994). Quantitative Structure-Activity Relationships by neural networks and inductive logic programming II . The inhibition of Dihydrofolate Reductase by triazines. *Journal of Computer Aided Molecular Design*, *8*, 421–432.

King, R., Hirst, J., & Sternberg, M. (1995). Comparison of artificial intelligence methods for modeling pharmaceutical QSARS. *Applied Artificial Intelligence*, *9*, 213–234.

King, R., Muggleton, S., Lewis, R., & Sternberg, M. (1992). Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to Dihydrofolate Reductase. *Proceedings of the National Academy of Sciences of the USA*, *89*, 11322–11326.

Levenberg, K. (1944). A method for the solution of certain non-linear problems in least-squares. *Quarterly of Applied Mathematics*, *2*, 164–168.

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, *11*, 431–441.

McGee, V. E., & Carleton, W. T. (1970). Piecewise regression. *Journal of the American Statistical Association*, *65*, 1109–1124.

Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.

Moore, A. W. (2001). Eight more classic machine learning algorithms. `http://www.cs.cmu.edu/~awm/tutorials`.

Press, W. H., Teukolsky, S. A., Vellering, W. T., & Flannery, B. P. (1992). *Numerical recipes in C*. New Delhi: Cambridge University Press.

Quandt, R. E. (1958). The estimation of the parameters of a linear regression system obeying two separate regimes. *Journal of the American Statistical Association*, *53*, 873–880.

Quinlan, J. (1992). Learning with continuous classes. *Proceedings of the Second Australian Conference on Artificial Intelligence* (pp. 343–348). World Scientific.

Quinlan, J. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kauffman.

Salford-Systems (2001). *MARS User Guide*. Salford Systems, San Diego, CA.

Sherrod, P. H. (2001). NLREG*: Nonlinear regression analysis program*. Reference Manual for NLREG v5.3.

Tishler, A., & Zang, I. (1981). A maximum likelihood method for piecewise regression models with a continuous dependent variable. *Applied Statistics*, *30*, 116–124.

Wang, Y., & Witten, I. H. (1997). Inducing model trees for continuous classes. *Proceedings of the Ninth European Conference on Machine Learning Poster Papers* (pp. 128–137).

Witten, I. H., & Frank, E. (1999). *Data mining: Practical machine learning tools and techniques with Java implementations.* Morgan Kaufmann.

Yu, J.-R., Tzeng, G.-H., & Li, H.-L. (2001). General fuzzy piecewise regression analysis with automatic change-point detection. *Fuzzy Sets and Systems*, *119*, 247–257.

Index