The Pennsylvania State University

The Graduate School

Department of Statistics

ADAPTIVE SPLINES AND GENETIC ALGORITHMS FOR OPTIMAL STATISTICAL MODELING

A Thesis in

Statistics

by

Jennifer L. Pittman

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2000

We approve the thesis of Jennifer L. Pittman.

Date of Signature

C.R. Rao
Professor and Holder of the Eberly Family Chair in Statistics
Thesis Adviser and Chair of Committee

Rich Caruana
Assistant Professor of Computer Science
University of California, Los Angeles
Special Member

Mark Handcock
Associate Professor of Statistics

Zhen Luo
Assistant Professor of Statistics

Soundar R. Tirupatikumara
Professor of Industrial Engineering

Bruce Lindsay
Distinguished Professor of Statistics
Interim Head of the Department of Statistics

Abstract

Many statistical analyses and applications require the capture of a relationship between two variables X and Y that is more complex than a simple linear relationship. One approach to solving this problem is to use a modeling technique to attempt to replace the complex and/or noisy relationship which the data represent by something simple yet reasonable which captures the nature of the dependence in the data. In the case where little is known about the underlying function which relates X to Y, the modeling technique should be *flexible* or *adaptive*, i.e., able to handle a wide variety of functional shapes and behaviors. Nonparametric modeling is one such technique which has been successful in characterizing features of datasets that would not be obtainable by other means [70].

Due in part to the increased availability of computational power, spatially adaptive smoothing methods involving regression splines have become a popular and rapidly developing class of nonparametric modeling techniques. Most of these methods are based on nonlinear optimization and/or stepwise selection of basis functions. Although computationally fast and spatially adaptive, stepwise knot selection is necessarily suboptimal while determining the best model over the space of adaptive splines is a very poorly behaved nonlinear optimization problem [143]. A possible alternative is to use a genetic algorithm to perform knot selection. Genetic algorithms are stochastic search methods which, under certain design conditions, have the capability or potential of converging to the global optimum of the evaluation function of an optimization problem [14]. In other words, a genetic algorithm search is not biased from reaching the global optimum. Hence, given a variable selection criterion and a search space of possible knot locations, a genetic algorithm has

the potential to find models that are more appropriate in comparison to models selected using nonlinear optimization or stepwise methods.

In this work we explore the use of genetic algorithms for adaptive spline modeling in low dimensional settings. We introduce our work in Chapter 1 and discuss genetic algorithms in Chapter 2. Chapters 3 and 4 cover work involving optimal modeling with linear splines only while Chapter 5 involves the optimal fitting of polynomial splines. Experimental results are compared to those of Dierckx [52], Luo and Wahba [97], Donoho and Johnstone [54], Friedman [64], and Stone, Kooperberg, Hansen, and Troung [137], among others. Future research is focused on comparisons with the recent Bayesian spline methods (e.g., Denison, Mallick, and Smith [50]), extensions to higher dimensions, and explorations into other areas of study such as neural networks and fractals.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to thank my parents, Dr. and Mrs. Thomas Pittman, my sister Holly Joseph and my brother Christopher Pittman for their love and support. This work would not have been possible without them.

I would also like to acknowledge the assistance and special friendship of Dr. S. Banga.

I am indebted to Professor C. A. Murthy, Professor J. Kadane and Professor R. Caruana for their personal involvement and guidance during my studies.

I extend my best wishes and gratitude to my advisor, Professor C.R. Rao, for his patience and wisdom.

Chapter 1

Introduction

In many statistical applications, a modeling technique is needed which can capture a relationship between two variables X and Y that is more complex than a simple linear relationship. One approach to solving this problem is to attempt to replace the noisy and/or complex relationship which the data represent by something simple yet reasonable which captures the nature of the dependence in the data. In the case where little is known about the function f, the modeling technique should be *flexible*, i.e., able to handle a wide variety of functional shapes and behaviors. Nonparametric modeling is one such technique which has been successful in characterizing features of datasets that could not be obtainable by other means.

1.1 Nonparametric Modeling

The class of nonparametric modeling techniques includes methods, suggested by classical approximation theory, which involve building models from linear combinations of nonlinear functions [36]. Such linear estimators can be expressed as

$$\hat{f}(X) = \sum_{i=1}^{n} K_{\lambda}(X, x_i) y_i \tag{1.1}$$

where $K_{\lambda}(X, x_i)$ is a weighting function which depends on some parameter(s) λ [61]. Some examples include kernel estimates, series approximators (which we will not explicitly

discuss), locally weighted regression, the more recent neural network and radial basis function estimates, and splines.

1.1.1 Kernel estimators and LOESS

Kernel estimators can be expressed in the above form where the weighting function or kernel K_λ has a simple form independent of the design of $\boldsymbol{X} = (X_1, \ldots, X_N)$. Examples of such weighting functions are the uniform and triangular kernels. K_λ is a bounded function assumed to have support [-1,1], with a maximum at zero, and is usually chosen to satisfy certain moment conditions [147]. The choice of moment conditions determines the order m of the kernel estimate: conditions on higher order moments lead to higher order estimates. The parameter λ is called the *bandwidth* and determines (1) the maximum distance away from x_i a data point can be and still be included in the estimation of $f(x_i)$, and (2) the amount of emphasis placed on observations at certain distances from x_i. Note that kernel methods require the researcher to select the appropriate values for K_λ, λ, and m. Although these choices are critical to the quality of the resulting estimate, they are often made by trial-and-error or time consuming adaptive methods [111]. A popular way of choosing λ is by *cross-validation* (CV), although CV does not guarantee the selection of the optimal λ [147].

Locally weighted regression, or LOESS [40], fits an estimate $g(X)$ which uses a neighborhood of weighted observations whose values are closest to X. $g(X)$ may be written as

$$g(X) = \sum_{i=1}^{N} l_i(X) y_i \tag{1.2}$$

where the $l_i(X),\ i = 1, \ldots, N$, depend on $\{X_1, \ldots, X_N\}$, a neighborhood size q, a weighting function W, and a distance measure p. Local fitting allows the estimation of a wide class of regression surfaces, wider than the class which can be estimated by polynomials. However, LOESS does assume that $\boldsymbol{Y}$ has a normal distribution and that the estimate $g(X)$ is unbiased. Local polynomial estimators like LOESS do not suffer from the boundary bias inherent in using kernel regression estimators, i.e., because LOESS fits local polynomials at data points in the boundary region, it does not flatten out because of lack of data past the boundary region. However, this comes at the price of increased variance [132].

1.1.2 Neural Networks and Radial Basis Functions

A recent development in functional approximation is the use of *neural networks* (NN) and *radial basis functions* (RBFs). Multilayer neural networks are linear function approximators (in the sense of (1.1)) of the form, e.g.,

$$\hat{f}(X, \mathbf{W}) = \sum_{j=1}^{M} \beta_j g_j(\alpha_j X) \tag{1.3}$$

where $\{\beta_j\}_{j=1}^{M}$, are the weights connecting M hidden units to the output unit, $\{\alpha_j\}_{j=1}^{M}$, are weights connecting the input layer unit to the jth hidden layer unit, and $\{g_j\}_{j=1}^{M}$ are the hidden layer activation functions [36]. $\mathbf{W}$ represents the matrix of network weights. Because of the form of $\hat{f}$, splines and kernel estimates are sometimes viewed as special cases of NN models. In fact, NNs have been used for nonparametric regression by several authors including Chen and Jain [36] and, more recently, Lee [92].

Another special case of NN models is based on radial basis functions where the approximation is produced by passing each X_i through a set of basis functions, each

containing a RBF center, multiplying the result by a coefficient, and then summing the results. In other words,

$$\hat{f}(X_k, \mathcal{W}) = w_0 + \sum_{j=1}^{M} w_j \tau^{-p} \phi(\|X_k - c_j\|/\tau) \tag{1.4}$$

where ϕ is the radial basis function, $\{c_j\}_{j=1}^{M}$ is the set of RBF centers, τ is a scale parameter, and p is the dimension of the data. Often ϕ corresponds to a Gaussian density [38, 144]. Note that a radial basis function network (RBFN) is essentially a kernel method for regression.

Neural networks are easily programmed and, as a result, have become an almost universal optimization 'crank': simply toss in the data, add any number of parameters, and wait for gradient descent to produce the result [31]. However, there is no method for finding the best network architecture for a given problem. Once an architecture has been chosen (heuristically), the system often requires numerous adjustments, supplied by an experienced user, and do have a tendency to overfit or overparameterize the data [31]. They may also get stuck at local minima since their optimization is based on gradient descent. Chen and Jain [36] report that backward propagation can be slow and sensitive to noise. They suggest a robust modification whose parameters are the focus of further study. RBFNs have been shown to outperform multilayer perceptrons (MLPs) [144] even though the choice of centers and the curse of dimensionality can make implementation difficult. It should also be noted that both NNs and RBFNs results lack interpretability [31].

1.1.3 Splines

A function $s(X)$ defined on a finite interval $[a, b]$ is a polynomial *spline* of order m (degree $m-1$) with knots $(\tau_1, \tau_2, \ldots, \tau_{(k+1)})$, $\tau_1 < \tau_2 < \cdots < \tau_{(k+1)})$, $\tau_1 = a$, $\tau_{(k+1)} = b$, if

1. On each interval $[\tau_i, \tau_{(i+1)}]$, $i = 1, \ldots, k$, $s(X)$ is a polynomial of order at most m.

2. The first $m-2$ derivatives of $s(X)$ are continuous on $[a, b]$, i.e., $s(X) \in \mathcal{C}^{m-2}[a, b]$.

In other words, a spline is a piecewise polynomial where the pieces are tied at knots in such a way that s satisfies certain continuity properties. As such they can be viewed as an extension of polynomial regression [61]. Note that the above definition is sometimes extended so that splines with coincident knots may be considered. In this case the continuity condition must be relaxed so that if $\tau_{(i-1)} < \tau_i = \cdots = \tau_{(i+j)} = c < \tau_{(i+j+1)}$, $j \leq m-1$, then only the first $m-2-j$ derivatives of $s(X)$ must be continuous at the point c. Different classes of splines (i.e., splines which span different subspaces of piecewise polynomials) can be formed by using different basis functions, e.g., B-splines, periodic splines, etc. Splines are useful when we want an estimate which meets a quality of fit (fitness) criterion as well as a smoothness criterion. For example, we may estimate Y by choosing $\hat{f}$ to minimize

$$\frac{1}{N}\sum_{j=1}^{N}(Y_j - f(X_j))^2 + \lambda \int_a^b f^{(m)} dX \tag{1.5}$$

for $\lambda > 0$, m a positive integer, and $a \leq X_j \leq b \ \forall \ j = 1, \ldots, N$. The solution $\hat{f}$ is called a *smoothing spline* estimate, and λ is the *smoothing parameter*. λ determines the tradeoff between goodness-of-fit and smoothness. If, however, our fitness criterion is not of this form, a spline may not be the best type of estimate. It is also of note that

methods designed to find a good estimate for λ are computationally demanding [122] with cross-validation (CV)[42] estimates tending to oversmooth the data [62].

1.2 Splines and Spatially Adaptive Modeling

However, nonparametric modeling involving the use of univariate (or multivariate) spline spaces has been found to be successful in numerous statistical applications, e.g., materials science [89], computer tomography [61], and military analysis [91]. More generally, due in part to the increased availability of computational power, spatially adaptive smoothing methods have become a popular and rapidly developing class of nonparametric modeling techniques. One recent contribution to this class are the wavelet shrinkage methods of Donoho and Johnstone [53, 54] whose capabilities are supported by both theoretical and simulation results.

Traditionally, spatially adaptive smoothing methods could be classified into one of two groups. The first group consists of methods which use a locally adaptive smoothing parameter with smoothing splines or kernel techniques. A recent example is the variable bandwidth kernel method of Fan and Gijbels [63]. The second group, which we will discuss below, consists of regression spline fitting algorithms where the knot locations are chosen adaptively.

1.2.1 Nonlinear Optimization

Some of the earliest attempts at fitting adaptive or variable knot splines include those of Bellman and Roth [19] for linear splines and Hawkins' [72] method based on dynamic programming. These have been followed by algorithms which iteratively add or delete knots but do not attempt optimal knot placement, only a distribution for the knots

which is in some sense optimal. Included here is de Boor's algorithm NEWKNOT [45], the method of Agarwal and Studden [2] for linear splines, and the recent contribution of Hu [79]. In many of these methods, an optimal distribution is one in which knots are placed where the absolute value of the fourth derivative of the unknown function is largest. This is motivated by the observation that cubic splines have a piecewise constant third derivative and more knots are needed where the third derivative is rapidly changing (where the fourth derivative is largest) than where it is slowly changing. This approach is supported by Agarwal and Studden's [3] result that the minimum asymptotic integrated mean squared error (IMSE) is achieved when the density of the knots and design points is proportional to the absolute value of the fourth derivative of the true function. Unfortunately, in statistical applications usually not enough is known about the unknown function for the estimation of higher derivatives to be any easier than estimation of the function itself.

Instead of optimizing the knot distribution, there are nonlinear optimization routines which attempt to optimize knot placement with respect to least squares (LS) error for a fixed, given number of knots. de Boor and Rice [47] and Jupp [85] perform LS cubic spline approximation by beginning with an initial knot sequence and searching for the best knot placement by Gauss-Newton methods. Jupp noted that 'free' knot splines suffer from a property he called *lethargy*, i.e., if we define the knot vector $\boldsymbol{\tau}_{i,\beta} = (\tau_1, \ldots, \tau_{i-2}, c-\beta, c+\beta, \tau_{i+1}, \ldots, \tau_{k+1})$, then

$$\lim_{\beta \to 0} \partial RSS_{i,\beta}/\partial\beta = 0$$

where RSS is the residual sum of squares when a spline with knot sequence $\boldsymbol{\tau}_{i,\beta}$ is fit to the data. Hence there are many stationary points along points and ridges in the

parameter space where knots coincide. His solution was a transformation of the knots that removes redundant knots; this improves the behavior of Newton-Raphson algorithms and improves the chances of finding a global optimum. Unfortunately, failure to converge and difficulty in areas near replicate knots still occur frequently. The algorithm of Dierckx [51] is similar but employs the Fletcher/Reeves conjugate gradient method in determining the knot locations instead of using a transformation with Newton-Raphson. Schwetlick and Schütze [130] have presented a nice extension of these works based on generalized Gauss-Newton methods which allows for the optimal placement of a subset of the knot sequence. They include a knot removal strategy (see Lyche and Mørken [98]) to search for the model which approximates the data to within its estimated noise level using the fewest number of knots.

A recent approach to free knot spline modeling which employs nonlinear optimization is the algorithm of Lindstrom [95]. Approaching the choice of knot locations as a parameter estimation problem and noting the 'lethargy' property of 'free' knot splines, her algorithm attempts to find an estimate for the knot sequence which minimizes the residual sum of squares times a penalty, where the penalty increases as knots coalesce. In other words, the penalty is designed to penalize those knot sequences with replicate knots. This ameliorates lethargy and is intuitively appealing if the true, unknown function is assumed to be smooth. She proposes a new model selection criterion based on generalized cross-validation (GCV) for subset selection [42]. Generalized cross-validation is a modified version of cross-validation where the factors $(1 - \mathbf{S}_{ii})$ are replaced by their average value, $(1 - \mathrm{tr}(\mathbf{S})/N)$, where $\mathbf{S}$ is the smoothing matrix of the spline model and N is the sample size. One main reason for using GCV instead of CV is ease of computation. Lindstrom's version of GCV uses the trace of the influence matrix for both the coefficients

and the knots as an estimate of the degrees of freedom (in the original GCV criteria, tr($\mathbf{S}$) is the estimate of degrees of freedom and $\mathbf{S}$ is the influence matrix for the coefficients). Simulation results show that for several examples the penalized estimator is superior in performance to unpenalized estimators. It should be noted, however, that although these results are promising, Newton-Raphson/Levenberg-Marquardt techniques cannot escape local optima and the question of generating good starting values has been left unanswered.

1.2.2 Statistical Adaptive Spline Modeling

In the statistical literature, the knot locations of regression splines in additive modeling are restricted to the set of design points and the problem of finding the optimal set of knots is approached from the perspective of predictor subset selection instead of parameter estimation. The proper choice of knot sequence is usually determined by a stepwise addition and/or deletion procedure where only models whose number of basis functions fall within a prespecified range are considered. The appropriate model is taken as the one which minimizes a model selection criterion borrowed from the parametric literature, e.g., Mallows' C_p statistic [100] and generalized cross-validation (GCV) [42]. An explicit discussion of additive models will not be given here; see [71] for a thorough review of additive modeling.

1.2.2.1 Stepwise Selection

Applications of statistical variable selection techniques to spline fitting have developed from the seminal work of Smith [133], whose method begins with many knots and then deletes those which are least important with respect to approximation and/or prediction error. Breiman's [29] DK/CV algorithm is based on Smith's work; it uses

knot deletion combined with a cross-validation criterion for model selection. His work has shown DK/CV to be superior on small, noisy datasets to Breiman and Friedman's Alternating Conditional Expectations (ACE) [30] algorithm which uses a backfitting approach. Friedman and Silverman's algorithm TURBO [65] takes Breiman's idea one step further. It does stepwise addition of knots to get a series of models; of these, the model which minimizes a GCV criterion is chosen. Knots are then deleted from this model to achieve the final fit. In Multivariate Adaptive Regression Splines [MARS] [64], Friedman utilizes a stepwise knot addition/deletion strategy with linear splines. His GCV model selection criterion includes a 'cost' term to adjust for the adaptive nature of the knot selection. More recent contributions to additive modeling have been made by Luo and Wahba (Hybrid Adaptive Splines [HAS]) [97] and Stone, Hansen, Kooperberg, and Troung ([POLYMARS]) [90, 137]. HAS performs forward knot selection via GCV with a 'cost' term, as in MARS, but replaces backward deletion by ridge regression. POLYMARS is a multiresponse version of MARS which has been customized for computational efficiency.

The algorithms mentioned thus far in Section 1.2 are based on nonlinear optimization and/or stepwise selection. Although computationally fast and spatially adaptive, stepwise knot selection is necessarily suboptimal since there is no way to move from an undesirable local optimum. Determining the best model over the space of adaptive splines is a very poorly behaved nonlinear optimization problem [143]. As mentioned above, it has been noted by Jupp [85] that with 'free' knot LS splines, the residual sum of squares error function is nonconvex in the knot sequence $\boldsymbol{\tau}$. Hence nonlinear optimization algorithms which use generalized Gauss-Newton and/or conjugate gradient methods can converge to local minima or saddlepoints. Excellent starting values are required by such algorithms

due to the many local optima in the error surface; however, these are particularly difficult to identify by inspection of the data [95].

1.2.2.2 Bayesian Methods

The need for a more intelligent search scheme has driven recent work on Bayesian model selection and Markov chain Monte Carlo (McMC) methods for model identification [70]. Bayesian spline methods which combine both McMC and model selection include those from Smith and Kohn [134], Denison, Mallick, and Smith [50], Shively, Kohn, and Wood [131], and Biller [24]. In essence, a posterior distribution is created computationally which allows a simple Gibbs sampler to quickly search through a large number of knot locations in search of a posterior mode. Smith and Kohn and Denison, Mallick, and Smith (using reversible jump McMC; see Green [69]) consider model averaging with regression splines in the case of Gaussian errors; Shively, Kohn, and Wood's method utilizes a data-based prior, variable selection, and model averaging with McMC techniques for additive nonparametric Gaussian and binary regression modeling. Biller introduces an adaptive Bayesian regression approach to semiparametric generalized linear models which uses reversible jump McMC to estimate the number and placement of knots in B-spline models for non-Gaussian responses. These methods have proven to be quite successful, although it should be noted that if the McMC chain is started in a low density region, convergence can be slow. The sampler may effectively get stuck in a local mode if the density is multimodal, while the density of interest may also be highly correlated so mixing is slow [78]. Hence performance depends heavily upon the choice of priors - an inappropriate prior can lead to results much worse than those of greedy algorithms. However, the performance

of such methods shows that when properly designed, stochastic methods can find many more reasonable knot configurations than their greedy, deterministic competitors [70].

As noted by Hansen [70], the use of McMC for selecting the posterior mode in a Bayesian framework is essentially *simulated annealing*. Simulated annealing methods, particularly Metropolis-type (Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller [104]), are stochastic optimization methods which produce a sequence that converges in probability to the set of global minima (maxima) of the loss function as the temperature converges to zero (see Chapter 2 for more information). Simulated annealing methods have been used successfully by Stone, Hansen, Kooperberg, and Troung [137] for logspline density estimation and by Geyer and Thompson [66] to aid in the mixing of McMC chains in correlated spaces.

1.2.2.3 Genetic Algorithms

The above observations led us to consider the possibility of using *genetic algorithms* (GAs) for knot selection. Genetic algorithms are stochastic search methods which, under certain design conditions, have the ability to converge to the global optimum of the evaluation function of an optimization problem [23]. Hence, given a variable selection criterion and a search space of possible knot locations, a genetic algorithm has the potential to find models that are more appropriate in comparison to models selected using stepwise or nonlinear optimization techniques. GAs have been used successfully for optimization in similar contexts, for example, Manela, Thornhill, and Campbell [102] used a genetic algorithm to determine the appropriate spline order and roughness penalty for penalized least squares (LS) splines, while Rogers [G/SPLINES][123] incorporated a modified GA search into MARS [64].

1.3 Thesis Outline

In Chapter 2 the structure of a genetic algorithm is outlined and its basic components - selection, crossover, and mutation - are explained. The use of GAs and least squares to fit piecewise linear functions to datasets in $\mathcal{R}^2$, where the number of pieces is known but the optimal locations of the knots are unknown, is discussed in Chapter 3, along with supporting theory and experimental results. Building upon this work, in Chapter 4 a study is described in which genetic algorithms were employed to fit piecewise linear models to data sets in $\mathcal{R}^2$, where the number of pieces (or knots) as well as their placement were unknown. Instead of using least-squares for model selection, an optimization criterion is introduced which was designed to be robust in the presence of outliers. Supporting theory and experimental results are also presented and discussed.

Moving away from piecewise linear functions, in Chapter 5 a spatially adaptive modeling technique referred to as adaptive genetic splines (AGS) is introduced which combines the optimization power of a genetic algorithm with the flexibility of higher order polynomial splines. Preliminary simulation results comparing the performance of the genetic algorithm method to other current methods, such as HAS [97], SUREshrink [53, 54], and POLYMARS [90], are discussed.

The work of Chapter 5 is followed by a summary and discussion of the above works in Chapter 6. Future research with genetic algorithms and splines is also discussed here, as well as some preliminary work on a different topic, namely, a mathematical link between multilayer preceptrons and fractals based on gradient descent and contractive maps.

Chapter 2

Genetic Algorithms

2.1 Introduction

The theory of natural evolution maintains that adaptive changes to species occur through natural selection, whereby individuals better suited for survival are more likely to reproduce and hence increase the representation of their genetic material in the gene pool. The combining of genetic material from two such individuals can produce an offspring even better suited to the environment. This is how species evolve, i.e., sex is a mechanism for exchanging genetic material and hence for species to rapidly adapt to their environment.

To address the shortcomings of classical deterministic optimization algorithms, a number of algorithms based on natural selection and embedded randomness have been developed. One such class of algorithms are the population-based methods of Evolutionary Computation (EC). EC is a class of search and optimization techniques based on the Darwinian evolutionary model. The development of EC can be traced back to the 1960s and the Evolutionary Strategies of Rechenberg and Schwefel in Germany and the Genetic Algorithms (GAs) of Holland [77] in the United States (see [10] for further historical background). One of their primary features is that they search through a large population of solutions for the one which optimizes a given evaluation or fitness function. Of the three branches of evolutionary computation - evolutionary programming, evolutionary strategies, and genetic algorithms - only genetic algorithms will be discussed here; see [135] for further information on EC.

Developed by Holland [77], Genetic Algorithms (GAs) mimic the process of evolution to obtain solutions to the evaluation function of an optimization problem. In finding solutions, they are capable of searching complex, multimodal surfaces via steps which have been designed to mimic the processes of natural genetic systems. The basic idea is to maintain a population (generation) of possible solutions that evolves and improves over time through a process of competition and controlled variation. Their success comes from their *adaptability*, i.e., the ability to exploit accumulating information about the search space in order to bias subsequent searches toward promising subspaces. One important issue when using GAs is how to properly balance of the opposing goals of *exploration* and *exploitation* [78]. Note that 'hillclimbing' algorithms do not allow for any such tradeoff within the algorithm itself, thus limiting their own search capabilities. They use only local information and hence consider only modes in the search space, ignoring the possibility of other potentially optimal locations. In contrast, a GA provides a high degree of exploration through the diversity of the gene pool and the crossover operator (described in Section 2.2.1). Meanwhile, a GA uses gradient information from the search space contained implicitly in the distribution of the population at any given moment. As the search proceeds, exploration is traded for exploitation as the GA converges to a solution. The effectiveness of a GA search for optimization has been demonstrated in solving various problems from scientific fields such as scheduling, classifier systems, and pattern recognition [43].

2.2 Basic GA

Each solution in the domain space $\mathcal{D}$ is encoded as a string or chromosome S. Each string is built of elements from a finite alphabet $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_a\}$; the alphabet is determined by the encoding scheme. For example, in binary coding, $\mathcal{A} = \{0, 1\}$. Although

a GA traditionally uses a binary encoding of the solution space, there are also Gray and real encodings which may be more efficient and provide a more precise solution than the binary-coded GA [9, 84]. The length L of each string is determined by the number of parameters to be found and the desired precision. Hence each string $\boldsymbol{S}$ corresponds to a value x in $\mathcal{D}$ and may be expressed as

$$\boldsymbol{S} = (\gamma_1, \gamma_2, \ldots, \gamma_L); \quad \gamma_i \in \mathcal{A} \quad \forall \;\; i = 1, \ldots, L$$

In each iteration t, a GA starts with a population $\mathbf{P}^t$ of M strings; hence $\mathbf{P}^t = \{\boldsymbol{S}_1, \ldots, \boldsymbol{S}_M\}$. The strings in $\mathbf{P}^t$ are randomly selected from the pool of all encoded solutions. Through operations based on natural selection, the genetic algorithm seeks to optimize a specified function $fit(X),\ X \in \mathcal{D}$. If $\boldsymbol{S}_i$ represents a solution $x \in \mathcal{D}$, then the evaluation or fitness value of $\boldsymbol{S}_i$ is defined as $fit(\boldsymbol{S}_i) = fit(x)$.

2.2.1 Operations

- **Selection**: Selection gives members of the present population $\mathbf{P}^t$ with large fitness values an increased chance of being present in the next (intermediate) population $\mathbf{P}_1^t$. It dominates early performance by increasing the concentration of solutions in regions with current fitness values above the population average [94]. Most selection mechanisms use either *proportional* [77] or *ordinal* [108] selection. In proportional selection, the probability of selecting a string $\boldsymbol{S}_i$ from $\mathbf{P}^t$ for inclusion in $\mathbf{P}_1^t$ is $p_s(\boldsymbol{S}_i) = fit(S_i)/\sum_i^M fit(\boldsymbol{S}_i)$. The expected number of occurrences of string $\boldsymbol{S}_i$ in the subsequent population is $Mp_s(\boldsymbol{S}_i)$ (referred to as the *expected value* of $\boldsymbol{S}_i$ [76]). In ordinal selection, the strings are sorted according to their fitness values and $p_s(\boldsymbol{S}_i)$ is based on the rank of $\boldsymbol{S}_i$ using a nonincreasing assignment function. M

strings are selected with replacement from $\mathbf{P}^t$ using the above selection probabilities to form the intermediate population $\mathbf{P}_1^t$. Other selection rules based on fitness and threshold values are described in Adachi and Matsuo [1].

- **Crossover (recombination)**: Crossover or recombination allows pairs of strings from $\mathbf{P}_1^t$ to combine their better features to create improved strings for the next (intermediate) population $\mathbf{P}_2^t$. In this way, hyperplanes in the search space containing a region possessing a higher average fitness value can intersect, allowing good solutions to be achieved [94]. Crossover creates population diversity and thus helps prevent premature convergence. By promoting exploration, it is critical for *hillclimbing hard* (HC-hard) [68] problems where the probability of finding a point within the global basin of attraction on a given random trial τ is

$$p_\tau < 1 - (1 - \alpha)^{n_r}$$

where $\alpha > 0$ is small and n_r is the number of trials. In this case we say that the target τ is a *needle in a haystack* (NIAH). Exploration is important in solving our research problem since in higher dimensions, the regions of the search space which are occupied by the optimal knot locations are tiny fractions of the entire space. Hence finding the optimal knot locations for a spline model of fixed order is, in many cases, an HC-hard problem.

In performing crossover, all strings in $\mathbf{P}_1^t$ are paired at random in such a way that each string belongs to only one pair (hence there are $M/2$ pairs). For example, let $\boldsymbol{S}_1 = (\gamma_{11}, \ldots, \gamma_{1L})$ and $\boldsymbol{S}_2 = (\gamma_{21}, \ldots, \gamma_{2L})$ be two strings from $\mathbf{P}_1^t$ that have been selected for crossover, where for any pair of strings $\boldsymbol{S}_i, \boldsymbol{S}_j \in \mathbf{P}_1^t$ the probability

of undergoing crossover is p_c. In *simple* crossover [77], a position $i \in \{1, 2, \ldots, L-1\}$ is randomly chosen and two new chromosomes are built,

$$\dot{\boldsymbol{S}}_1 = (\gamma_{11}, \ldots, \gamma_{1i}, \gamma_{2(i+1)}, \ldots, \gamma_{2L}) \quad \text{and} \quad \dot{\boldsymbol{S}}_2 = (\gamma_{21}, \ldots, \gamma_{2i}, \gamma_{(2i+1)}, \ldots, \gamma_{2L}).$$

$\dot{\boldsymbol{S}}_1$ and $\dot{\boldsymbol{S}}_2$ are placed in $\mathbf{P}_2^t$ and $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are discarded. Each of the $M/2$ pairs undergoes crossover with probability p_c; any pair not selected for crossover is placed directly into $\mathbf{P}_2^t$. Goldberg [68] has observed that simple crossover, in combination with selection, will lead a real-coded genetic algorithm to process above-average areas of the search space in ways similar to those used by GAs based on lower cardinal alphabets.

A crossover operator is *disruptive* if the resulting strings are not members of the same subspace as the current strings. As discussed in Section 2.3, theoretical results have shown that subspaces of higher than average fitness will be allocated exponentially larger representations in the population if the effects of mutation and crossover are not too disruptive. This motivates the choice of a crossover operator with few crossover points. However, the study of the interaction of crossover and population size conducted by DeJong and Spears [49] indicates that the chance of crossover disruption is not the only factor that should be considered when choosing a crossover operator. It was determined that disruption could be advantageous late in the evolutionary process when the population is relatively homogeneous and when the population size is not large enough to allow an accurate sampling of the search space. Their simulations revealed that having a higher number of crossover points

can be beneficial when the population size is small by helping to overcome the limited information capacity of the population and the tendency for homogeneity.

- **Mutation**: Mutation gives the algorithm an opportunity to branch into previously unexplored regions of the domain space by arbitrarily altering one or more characters of a selected string. In finite populations, genetic drift occurs and hence coordinate values die due to the stochastic effects of selection. Mutation can restore lost or unexplored genetic material into the population to prevent premature convergence and ensure that the probability of reaching any point in the search space is never zero.

  Each character of every string undergoes mutation with probability p_m. In the simplest case, for each character γ_{ij} in $\mathbf{P}_2^t,\ i = 1, \ldots, M,\ j = 1, \ldots, L$, a random number r is generated from $[0, 1]$. If $r > p_m,\ \gamma_{ij}$ is unchanged; otherwise, γ_{ij} is replaced by a randomly selected member of the set $\{\mathcal{A} - \gamma_{ij}\}$. We denote the resulting population as $\mathbf{P}_3^t$. The works of various authors [9, 28, 73, 119] suggest an adaptive mutation scheme, e.g., applying different mutation densities and varying the probability of mutation during the GAs generations.

  In practice, with a finite population size, it seems preferable to set the mutation noise as large as possible to guarantee sufficient coverage of the search space and the convergence of the algorithm. For example, in high dimensional spaces the curse of dimensionality would appear to suggest that it is important to have a high mutation rate during the earlier and later iterations since most features will be NIAH [68]. However, mutation should not destroy structures, or combinations of parameter values, more quickly than selection can reproduce them; otherwise, the population

will lose structures that are needed for the optimal solution. This implies that the mutation rate has to be reduced as the dimension of the space increases in order to retain a certain survival probability for these structures [74]. Hence the probability of mutation is often proportional to $1/(\sqrt{M}L)$ as well as $e^{-\lambda t/2}$. In this way the mutation probability is proportional to the population size and the dimension of the space yet also adaptive, decreasing exponentially as the number of iterations increases.

Both crossover and mutation are recombination operations which maintain a sufficiently broad yet refined search.

The choice of p_c and p_m can be a complex nonlinear optimization problem. These values also depend critically on the specific nature of the fitness function. Despite this, some guidelines have been suggested in the literature:

1. For $M \approx 100$: p_c=0.6, p_m=0.001; for $M \approx 30$: p_c=0.9, p_m=0.01 (binary)[101]
2. $p_c \approx 0.6$, $p_m \approx 0.001$ (binary) [48]
3. $0.1 \leq p_m \leq 0.5$ decreasing exponentially (binary and gray) [9]
4. p_c=0.2, p_m=0.7 in exponential mutation with decreasing mean (integer) [28]
5. p_c=0.6, p_m=0.024 (real) [73]

These rates may vary with the choice of crossover and mutation operators, although Hesser and Männer [74] noted that previous research has indicated that p_c is not a sensitive parameter.

An additional selection strategy referred to as an *elitist* step [48], where the best individual from the present population is included in the subsequent population, is often

incorporated since the best chromosome may disappear due to crossover or mutation (see Section 2.4).

$\mathbf{P}_3^t$ is now relabeled as $\mathbf{P}^{(t+1)}$ and the cycle of operations is repeated until some termination criterion is met, at which time the best string achieved is generally taken as the solution to the optimization problem. Three popular heuristic stopping rules are:

1. Execute the process for a fixed number of iterations and report the best string found as the solution,
2. Execute the process until the fitness value does not show sufficient improvement over a fixed number of iterations, or
3. Execute the process until a predetermined fitness value is achieved.

2.3 Schema theory, Building blocks, and Coding

Why do GAs work? There are two prevailing theories on this subject: schema theory and the building block hypothesis.

2.3.1 Schema theory

Consider a problem whose solution can be encoded in a binary string of length three. A cube in $\mathcal{R}^3$ with height and width equal to 1 and origin at $(0, 0, 0)$ can be used to represent the solution space, as was done in Man, Tang, Kwong, and Halang [101]. If "$*$" denotes a wild card, then strings containing "$*$" are schemata and the front face of the cube represents the schema $1**$.

Note that each schema corresponds to a hyperplane in the search space and 3^L hyperplanes can be defined over the search space, where L is the length of the string.

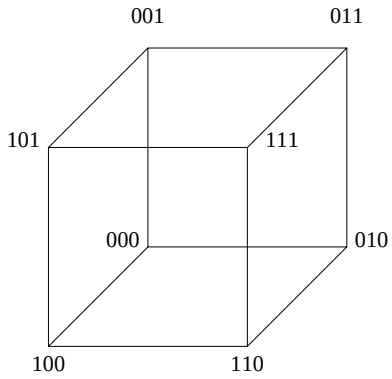


Figure 2.1 A cube representing schemata in $\mathcal{R}^3$

A GA uses a population based search and the population of sample points provides information about the hyperplanes in the search space. Numerous points should be sampled from lower order schema (*order* is the number of defined bits) and the cumulative effect of evaluating points provides statistical information about subsets of hyperplanes. Schema theory maintains that through the operations of selection, crossover, and mutation, the schemata of competing hyperplanes increase or decrease their representation in the population depending on the relative fitness values of the strings which they contain. Hence using fitness values one can estimate the expected number of strings matched by a particular schema in the next generation.

The following results can be found in Man, Tang, Kwong, and Halang [101].

Let $\lambda(\Phi, t)$ be the number of strings matching schema Φ in the current generation, generation t. Then, considering only proportional selection, the expected number of strings matching schema Φ in generation $t+1$ is

$$E(\lambda(\Phi, t+1)) = \lambda(\Phi, t)(fit(\Phi, t)/fit(t))$$

where $fit(\Phi, t)$ is the average fitness values of the strings matching schema Φ and $fit(t)$ is the average fitness value of the strings in generation t. If $fit(\Phi, t) > fit(t)$, then the number of strings matching schema Φ is expected in increase exponentially:

$$E(\lambda(\Phi, t+1)) = (\lambda(\Phi, 0)) \times (1+\epsilon)^t$$

where $\epsilon = (fit(\Phi, t) - fit(t))/fit(t)$.

In one-point crossover, a position is selected at random from the $L-1$ possible positions as the crossover point. The *defining length* of a schema Φ, denoted $\delta(\Phi)$ is defined as the distance between its outermost fixed positions (e.g., the defining length of $000*$ is two; of $1**0*$ is three). Hence the probability of schema Φ being disrupted by crossover is

$$p_d(\Phi) = \delta(\Phi)/(L-1).$$

Thus the probability of the survival of schema Φ is

$$p_s(\Phi) \geq p_c(1 - (\delta(\Phi)/(L-1)))$$

(note that a schema disrupted by crossover may still survive).

Similarly, if the order of schema Φ is $\theta(\Phi)$, then the probability of schema Φ surviving mutation is

$$p_s(\Phi) = (1-p_m)^{\theta(\Phi)}.$$

Since $p_m \ll 1$, we may approximate the above by

$$p_s(\Phi) \approx 1 - \theta(\Phi)p_m.$$

Hence the resulting schema growth equation,

$$E(\lambda(\Phi, t+1)) \geq \lambda(\Phi, t)\frac{fit(\Phi, t)}{fit(t)}[1 - p_c(1 - \frac{\delta(\Phi)}{(L-1)}) - \theta(\Phi)p_m].$$

Thus schemata with high average fitness values are expected to survive if the defining length is short and the order of the schema is low.

2.3.2 Building block hypothesis

The above observation can be restated by saying that schema theory implies that when $fit(\Phi, t) > fit(t)$, the representation of Φ in the next population is expected to increase if p_c and p_m are small. Note that this later condition is satisfied by schema of short length, since they are unlikely to be disrupted by crossover or disturbed by mutation. Hence it seems reasonable to suppose that a GA seeks near-optimal performance by juxtaposing short, low-order, highly fit schemata, or *building blocks* [106]. Hence the problem of coding a GA is essential to performance and should promote short building blocks. This idea supports the use of small alphabets since they maximize the number of schemata available for processing. For example, as presented in Goldberg [68], if the cardinality of the alphabet is α, then $\alpha + 1$ schemata are available at each position (including a wild character) and each position represents $\log_2 \alpha$ bits. Thus there are $n_s = (\alpha + 1)^{1/\log_2 \alpha}$ schemata per bit of information and the following theorem results [68]:

THEOREM 2.1 *For a given information content, strings coded with smaller alphabets are representatives of larger numbers of similarity subsets (schemata) than strings coded with larger alphabets.*

Proof: Note that $\nu = \log_2 n_s = \log_2(\alpha+1)/\log_2 \alpha = \ln(\alpha+1)/\ln \alpha$. Thus

$$\frac{d\nu}{d\alpha} = \frac{[\alpha \ln \alpha - (\alpha+1)\ln(\alpha+1)]}{\alpha(\alpha+1)\ln^2 \alpha}.$$

Since $\alpha \ln \alpha$ is monotonically increasing for $\alpha \geq 2$, $d\nu/d\alpha < 0$. exp is a monotonically increasing function and hence the result. ♠

However, there are arguments for the use of large alphabets which help explain the success of real-coded genetic algorithms. This will be discussed in more detail in Chapter 5.

Vose [141] and Vose and Liepins [142] have presented a generalized definition of schemata which includes the schema of Holland presented above. They define generalized schemata, or *predicates*, as groups of arbitrary strings and consider *invariant sublattices* of predicates, or groups of predicates partially ordered by set inclusion. Vose and Liepins argue that genetic search is directed by invariant sublattices which are closed or complete with respect to the crossover operator. An invariant sublattice $\mathcal{M}$ is *complete* if for all predicates $\mathcal{C}_1 \in \mathcal{M}$,

$$\exists\, \mathcal{C}_2 \in \mathcal{S} :\ \mathcal{C}_1 \times \mathcal{C}_2 \subset \mathcal{C}_1 \wedge \mathcal{C}_2 \ \longrightarrow \exists\, \dot{\mathcal{C}}_2 \in \mathcal{M} :\ \mathcal{C}_1 \times \dot{\mathcal{C}}_2 \subset \mathcal{C}_1 \wedge \dot{\mathcal{C}}_2$$

where $\mathcal{S}$ is the collection or lattice of all predicates and $\times$ denotes crossover. Crossover is compatable with a progression from larger to smaller predicates when $\mathcal{M}$ is complete. Note that a complete sublattice is closed only if it is maximal, i.e., a complete sublattice is closed if

$$\mathcal{M} \subset \dot{\mathcal{M}} \longrightarrow \mathcal{M} = \dot{\mathcal{M}}.$$

They define building blocks as those schemata Φ_2 of above average fitness which, when disrupted by crossover with schemata Φ_1, lead to schemata which are contained in $\Phi_2 \cap \Phi_1$. Hence the interaction of crossover and schemata determines the building blocks which guide genetic search and upon which schemata analysis should be conducted. Let us define a problem as *deceptive* if the solution or object of search is not contained in those schemata of above average fitness that are not disrupted by crossover. Vose and Liepins' theory is supported by the observation that in many deceptive problems the monotone increasing predicates ($\{\Phi_i : \Phi_i = fit^{-1}([b, +\infty)),\ b \in \mathcal{R},\ i = 1, \dots\}$) are unstable, i.e., if $p(\Phi_i)$ represents the probability that Φ_i is invariant under crossover of distinct members of Φ_i, then $p(\Phi_i) \ll 1$.

2.4 Convergence

With any optimization technique, it is important to ensure that the process will lead to the optimal solution. The formal convergence of GAs has been derived in several references, with most proofs constructed assuming a finite search space and using the powerful tools of Markov Chain theory. Eiben, Aarts, and Van Hee [59] derived a convergence in probability result for an elitist GA using such tools. Davis and Principe [44] used a Markov Chain model and ideas from simulated annealing to provide theory for the existence of the stationary distribution for the transition matrix of a GA when the mutation probability is a positive constant. Rudolph [124] examined the canonical GA and determined that it will never converge to the global optimum unless the best solution obtained is saved over the course of the search.

Nix and Vose [113] also take a Markov Chain approach to analyze the steady state distribution of a simple binary GA with a finite population. Their results shall be

summarized here. Define $\mathbf{Z}$ to be an $r \times N$ matrix whose ith column ϕ_i represents the incidence vector of population $\mathbf{P}_i$ of size M, where r is the number of possible strings and N is the number of possible populations. Each element $Z_{l,i}$ represents the number of occurrences of string l in the ith population $\mathbf{P}_i$. Nix and Vose calculate the transition probabilities by examining how the incidence vector ϕ_i describes the composition of the next generation.

The probability of producing string l in the next generation given that the current population is $\mathbf{P}_i$ is denoted as $p_i(l)$. Using this framework, the probability of producing a population $\mathbf{P}_j$ from $\mathbf{P}_i$, denoted $q_{i,j}$ follows a multinomial distribution with parameters $(M,\ p_i(0), \ldots, p_i(r-1))$:

$$q_{i,j} = M! \prod_{l=0}^{r-1} \frac{\{p_i(l)\}^{Z_{l,j}}}{Z_{l,j}!}.$$

Let $w_{i,j}$ be the probability that string l results from recombination (crossover and mutation) based on parent strings i and j and let $\mathbf{W}$ denotes the matrix having $w_{i,j}(0)$ as the (i,j)th entry. If $\oplus$ denotes exclusive-or on the integers, and permutations on $\mathcal{R}^r$ are defined as

$$\varphi_j\langle X_0, \ldots, X_{r-1}\rangle^T = \langle X_{j\oplus 0}, \ldots, X_{j\oplus(r-1)}\rangle^T,$$

then for a vector $\boldsymbol{X}$ the operator $\mathcal{M}$ is defined as

$$\mathcal{M}(\boldsymbol{X}) = \langle(\varphi_0\langle\boldsymbol{X}\rangle)^T\mathbf{W}(\varphi_0\langle\boldsymbol{X}\rangle), \ldots, (\varphi_{r-1}\langle\boldsymbol{X}\rangle)^T\mathbf{W}(\varphi_{r-1}\langle\boldsymbol{X}\rangle)\rangle^T.$$

Let fit be a positive objective function and let F be the linear operator with a diagonal matrix whose (i,i)th component is $fit(i)$. Note that $F\phi_i/|F\phi_i|$ is a vector pointing in the direction of ϕ_i with lth component equal to the probability that string l will be selected

for recombination; $\mathcal{M}(\phi_i/|F\phi_i|)$ is a vector with lth component equal to the expected proportion of string l in the next generation. Hence the transition matrix is given by

$$Q_{i,j} = M! \prod_{l=0}^{r-1} \frac{\left\{\mathcal{M}(\phi_i/|F\phi_i|)_l\right\}^{Z_{l,j}}}{Z_{l,j}!}.$$

Let $\boldsymbol{\pi}^k$ be the probability row vector having jth component equal to the probability that the kth generation is P_j. Let $\boldsymbol{\pi} = \lim_{k\to\infty} \boldsymbol{\pi}^k$ denote the steady state distribution and $\boldsymbol{\pi}^* = \lim_{M\to\infty} \boldsymbol{\pi}$. Using basic probability and Markov chain theory, the authors show that given $k > 0,\ \epsilon > 0,$ and $\gamma < 1$, there exists a population size M^* such that with probability at least γ and for all $0 \le t \le k$,

$$M > M^* \Rightarrow \| X_t - \mathcal{G}^t(X_o) \| < \epsilon$$

where X_0 is the initial state of the chain, X_t is the state of the chain at iteration t, and $\mathcal{G}^t$ is the tth iterate of $\mathcal{G}$, where $\mathcal{G}(X) = \mathcal{M}(\phi_i/|F\phi_i|)$. Let $\mathcal{F}$ be the fixed points of $\mathcal{G}$ and Λ represent the simplex formed by the states of the Markov chain, i.e.,

$$\Lambda = \{X \in \mathcal{R}^r : X > 0 \text{ and } |X| = 1\}.$$

Hence suppose that $\lim_{t\to\infty} \mathcal{G}^t \in \mathcal{F}$ for every $X \in \Lambda$. Then Vose and Nix prove that if $\boldsymbol{\pi}^*$ is a weak accumulation point of $\{\boldsymbol{\pi}\}_{M>0}$, then

$$\boldsymbol{\pi}^*(\mathcal{F}) = 1.$$

In other words, an accumulation point of $\{\boldsymbol{\pi}\}_{M>0}$ can only give positive probability to the fixed points of $\mathcal{G}(X) = \mathcal{M}(\phi_i/|F\phi_i|)$ provided that the iterates of $\mathcal{G}$ converge. Thus the probability of being away from $\mathcal{F}$ vanishes as $M \to \infty$, i.e., as the population size increases. Hence if a unique fixed point exists, then as the population size grows, the GA will spend asymptotically all of its time at the population which corresponds to that fixed point.

Although Vose and Nix do not specifically discuss mutation, from the work of Davis and Principe [44] it appears to be essential for convergence. Bhandari, Murthy, and Pal [23] note this in their analysis of the convergence of the GA with elitist model (i.e., with an elitist step). They have proven theoretically that an elitist genetic algorithm (fixed length strings) will converge to an optimal string as the number of iterations goes to infinity. The two characteristics that are necessary and sufficient for algorithm convergence are

- The optimal string from the present population has a fitness value no less than the fitness values of the optimal strings from the previous populations.
- Each string has a positive probability of going to an optimal string within any given iteration.

The outline of their proof is as follows.

Assume a genetic algorithm with finite population size M, string length L and alphabet $\mathcal{A}$ of cardinality a. Let $\mathbf{P}$ represent a collection of M strings, i.e., a possible population, and let $\mathcal{P}$ represent the class of all possible populations. The fitness value of a population, $fit(\mathbf{P})$, is defined as $fit(\mathbf{P}) = \max_{\boldsymbol{S}\in\mathbf{P}} fit(\boldsymbol{S})$ where the maximum is taken over all strings $\boldsymbol{S} \in \mathbf{P}$. $\mathcal{P}$ can be partitioned into sets where each set represents those populations having the same fitness value. In other words, let $\{fit_1, \ldots, fit_s\}$ represent

the set of possible fitness values, $s \leq a^L$, where $fit_1 > fit_2 > \cdots > fit_s$. Define

$$\mathcal{E}_i = \{\mathcal{P} : \ \mathbf{P} \in \mathcal{P} \text{ and } fit(\mathbf{P}) = fit_i\} \quad \text{for} \quad i = 1, \ldots, s.$$

Then $\mathcal{E}_i \bigcap \mathcal{E}_j = \emptyset$ and $\bigcap_{i=1}^{s} \mathcal{E}_i = \mathcal{P}$. Let $\mathbf{P}_{ij}$ be the jth population of $\mathcal{E}_i$, $i = 1, \ldots, e_i$ and $i = 1, \ldots, s$.

Denote as $p_{ij.kl}$ the probability that the genetic operators, in one generation, result in a population $\mathbf{P}_{k,l} \in \mathcal{E}_k$ from $\mathbf{P}_{i,j} \in \mathcal{E}_i$, and let $p_{ij.k} = \sum_{i=1}^{e_k} p_{ij.kl}$, $j = 1, \ldots, e_i$, $i, k = 1, \ldots, s$. Under the assumption that

$$\min_{i,j} p_{ij.1} > 0$$

it was proved that

$$\lim_{n \to \infty} p_{ij.1}^{(n)} = 1 \ \ \forall \ \ j = 1, \ldots, e_i \ \text{ and } \ i = 1, \ldots, s$$

where $p_{ij.1}^{(n)}$ denotes the probability of reaching a population in $\mathcal{E}_1$ in n generations with the starting population $\mathbf{P}_{i,j}$.

There are many results relating to the global convergence of GAs in the literature, including those given above. However, the same cannot be said for results concerning the rate of convergence. The mathematical complexity of analyzing GA convergence rates is considerable [135], hence the convergence rate results that exist are limited to fitness functions with special properties that can be exploited. In almost all cases, the fitness function is assumed to be either spherical (as in Beyer [22]) or strongly convex (see Rudolph

[125]). It is unclear what theoretical conclusions can be drawn from a broader class of problems.

It was mentioned in Chapter 1 that *simulated annealing* (SA) methods have been used successfully in several statistical applications. In their study of GA convergence, Eiben, Aarts, and Van Hee [59] describe a simulated annealing algorithm as follows:

- let A, B, and C be sets and $\alpha \in A$, $\beta \in B$, and $\gamma \in C$ be chosen by random drawings.
- S = arbitrary search space, $C = (0, 1] \subseteq \mathcal{R}$, P = set of individuals capable of producing offspring = $\{[s] \mid s \in S\}$.
- $f_c(\alpha, [s])$ = function to choose set of individuals from P for population = $\{[s]\}\ \forall\ \alpha \in A$.
- $f_p(\beta, [s])$ = function to produce children from population = $[t_\beta]\ \forall\ \beta \in B :\ t_\beta \in\ N(s)$ for a given neighborhood function N.
- $f_s(\gamma, [s, t])$ = selection function

$$= \begin{cases} [s] & \text{if } \exp\left[\frac{f(s)-f(t)}{c}\right] > \gamma \\ [t] & \text{otherwise} \end{cases}$$

where $0 < \gamma \leq 1$ by definition of C, c is the control parameter or *temperature*.

Hence simulated annealing can be viewed as a special form of a genetic algorithm with a population of size 1, a specific selection function, and children created by mutation only. During the optimization process the temperature is lowered, until only moves to lower states are accepted. The Metropolis algorithm provides a means for accepting moves

which increase the energy, hence allowing the algorithm a way to escape local optima [46]. Simulated annealing, like GAs, has been proven to achieve the global optimum of any optimization problem given an infinite amount of time [139]. Unfortunately, also like parameter selection for GAs, there are no true guidelines for choosing a suitable temperature schedule for simulated annealing - only rules of thumb based on experience.

So which performs better, evolutionary methods or SA? The answer to this question is dependent upon the problem at hand. Boseniuk and Ebeling [27] have noted in their studies involving the traveling salesman problem that SA strategies that include Darwinian or Haeckel characteristics yield better results than pure Boltzmann strategies (Haeckel strategies are evolutionary strategies which involve a period of youth when mutations are frequent and selection is seldom followed by a period of maturity when mutations are seldom and selection is frequent). They noted that thermodynamic processes, after which simulated annealing is modeled, tend to be locked in local optima surrounded by a high threshold, whereas Darwinian strategies are capable of passing high thresholds by tunneling through if the next minimum is close. However, de Groot and Würtz [46] found that whether SA or evolutionary strategies were more successful depended upon the particular problem but that overall the results of both methods were quite close. They commented that for both techniques finding the best control parameters is quite difficult and success depends upon the experience of the programmer and the amount of effort spent to solve the problem. Given the few theoretical results that are available, there is no a priori knowledge of whether evolutionary algorithms or simulated annealing techniques will yield the best results given an optimization problem.

2.5 The Next Step

We have discussed in this chapter some of the theory behind genetic algorithms and the operations that they use to solve optimization problems. We now move towards the development of a statistical application of genetic algorithms, namely, adaptive spline modeling. In Chapter 3 we take the initial step of considering the problem of fitting piecewise linear functions to datasets in $\mathcal{R}^2$.

Chapter 3

Piecewise Linear Fitting When the Number of Pieces is Known

3.1 Introduction

3.1.1 Outline of Chapters 3, 4, and 5

In Chapters 1 and 2 we have discussed the problem of interest, some of the current literature and other solution approaches, and the basics of genetic algorithms. In this chapter and in Chapter 4 the topic of GAs for piecewise linear fitting will be discussed, with supporting theory and empirical results. In reading these chapters one may notice that the specific problem situations and parameter spaces involved are not necessarily complex, nor do the empirical results show that the GA methods in these situations are anything more than competitive with existing methods. The purpose of these chapters, however, was to demonstrate some theoretical support for GA modeling and hint at the modeling potential that a GA method, or other numerical optimization methods, may have in more difficult modeling situations. We will see in Chapter 5 that, in fact, using a GA for knot placement in more challenging parameter spaces does yield better models than traditional methods. The reader interested only in the adaptive genetic splines algorithm and not the results of similar methods in simpler modeling situations or the supporting theory may wish to skip to Chapter 5.

3.1.2 Introduction to Chapter 3

As alluded to in Chapter 1, function approximation is a basic statistical tool for characterizing and analyzing some process of interest. Data or measurements, often subjected to random error, are recorded and an approximation of the process generating function is constructed from this partial information [128]. Constructing a function from a set of input-output pairs is a common problem in numerous scientific and engineering fields, including pattern recognition, computer vision, and applied mathematics [38, 147].

Genetic algorithms are recently developed search and optimization techniques from the field of artificial intelligence. They have been shown to be efficient, robust, and produce near-optimal solutions to problems in areas such as pattern recognition, machine learning, and statistical classification [14, 110, 114, 115]. In this chapter the use of genetic algorithms and least squares to fit piecewise linear functions to data sets in $\mathcal{R}^2$, where the optimal locations of the knots are unknown, is proposed. Several examples are presented with results, and areas for future research are mentioned. Current methodologies for function approximation and genetic algorithms have been discussed in Chapters 1 and 2 and will not be presented here.

3.2 Theory of Line Fitting in $\mathcal{R}^2$

3.2.1 Problem Statement

The given dataset is denoted as $(\boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$, $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)$, $(x_i, y_i) \in \mathcal{R}^2 \; \forall \; i = 1, \ldots, N, \; 1 \leq N < \infty$. Define $x_{(1)} = \min\{x_i; \; i = 1, \ldots, N\}$, $x_{(N)} = \max\{x_i; \; i = 1, \ldots, N\}$, $y_{(1)} = \min\{y_i, \; i = 1, \ldots, N\}$, and $y_{(N)} = \max\{y_i, \; i = 1, \ldots, N\}$. The random variables X_i and Y_i are related by an unknown function f

such that $Y_i = f(X_i) + \epsilon_i$, where ϵ_i is a random error. The problem is to approximate the function f by a k-piecewise linear function $\hat{f}$, k known, where the knot locations $\boldsymbol{Z} = ((Z_{11}, Z_{12}), \ldots, (Z_{(k+1)1}, Z_{(k+1)2}))$ are unknown and the least-squares error is to be minimized. No assumptions are made regarding the smoothness of $\hat{f}$ or the distributions of $(\boldsymbol{X}, \boldsymbol{Y})$ and $\boldsymbol{\epsilon}$.

3.2.2 Mathematical Formulation

Let $\mathcal{L}_k$ represent the class of all k-piecewise linear functions $L_{j\cdot}(\boldsymbol{X})$ in $\mathcal{R}^2$ that can be expressed in the following form:

$$L_{j\cdot}(\boldsymbol{X}) = \begin{cases} L_{j1}(X_i) & \text{if } Z_{(j1)1} \le X_i < Z_{(j2)1} \\ L_{j2}(X_i) & \text{if } Z_{(j2)1} \le X_i < Z_{(j3)1} \\ \vdots \qquad\qquad \vdots & \qquad \vdots \\ L_{jk}(X_i) & \text{if } Z_{(jk)1} \le X_i \le Z_{(j(k+1))1} \end{cases} \tag{3.1}$$

where the knot sequence $((Z_{(j1)1}, Z_{(j1)2}), \ldots, (Z_{(j(k+1))1}, Z_{(j(k+1))2}))$ satisfies $Z_{(j1)1} \le Z_{(j2)1} \le \cdots \le Z_{(jk)1} \le Z_{(j(k+1))1}$, $Z_{(j1)1} = X_{(1)}$, $Z_{(j(k+1))1} = X_{(N)}$, and each L_{ji}, $i = 1, \ldots, k$, can be expressed as

$$X \cos\theta_{ji} + Y \sin\theta_{ji} = d_{ji}, \quad 0 \le \theta_{ji} \le \pi, \quad d_{ji} \in \mathcal{R}$$

where θ_{ji} $(0 \le \theta_{ji} < \pi)$ is the polar angle formed by the crossing of the Y-axis and L_{ji} (when the polar axis is the Y-axis and the origin is the intersection point between the Y-axis and L_{ji}), and d_{ji} is the perpendicular distance of L_{ji} from the origin (0,0). The number of elements $L_{j\cdot} \in \mathcal{L}_k$ is uncountable but a GA can only represent a countable number

of elements. However, we can restrict the class of functions under consideration to a finite (discrete) set by restricting the values of θ and d. Consider a genetic algorithm with binary representation and for any string let l_a be the number of bits used to express θ and let l_d be the number of bits used to express d. Note that the precision of the line is determined by both l_a and l_d. θ_{ji} is restricted to the values $\{0, \frac{\pi}{2^{\mathrm{l}}a}, \frac{2\pi}{2^{\mathrm{l}}a}, \ldots, \frac{(2^{\mathrm{l}}a-1)\pi}{2^{\mathrm{l}}a}\}$. In specifying d_{ji}, the rectangle *rect* formed by the points $(X_{(1)}, Y_{(1)})$, $(X_{(N)}, Y_{(1)})$, $(X_{(1)}, Y_{(N)})$, and $(X_{(N)}, Y_{(N)})$ is utilized. Note that *rect* contains the entire data set. Let *diag* be the maximum diagonal of *rect* and let l_θ be defined as

$$l_\theta = \begin{cases} X_{(1)} \cos\theta + Y_{(1)} \cos\theta & \text{if } 0 \le \theta < \pi/2 \\ X_{(N)} \cos\theta + Y_{(1)} \cos\theta & \text{if } \pi/2 \le \theta < \pi. \end{cases} \tag{3.2}$$

Then for a given θ_{ji}, d_{ji} may only take values within the set $\{d_{ji} = l_{\theta_{ji}} + h_{ji}\delta : \ h_{ji} \in \{0, 1, \ldots, 2^{\mathrm{l}}\mathrm{d} - 1\}, \ \delta = diag/(2^{\mathrm{l}}\mathrm{d} - 1)\}$. Let $\mathcal{L}_k^r$ denote the finite set of functions in $\mathcal{L}_k$ which satisfy these restrictions. Then $\mathcal{L}_k^r$ may be expressed as

$$\mathcal{L}_k^r = \{L_{j.}(\boldsymbol{X}) : L_{j.}(\boldsymbol{X}) \in \mathcal{L}_k, \ L_{ji} \text{ is of the form}$$
$$X \cos\theta_{ji} + Y \sin\theta_{ji} = l_{\theta_{ji}} + h_{ji}\delta \ \ \forall \ \ i = 1, \ldots, k,$$
$$\theta_{ji} \in \{0, \tfrac{\pi}{2^{\mathrm{l}}a}, \tfrac{2\pi}{2^{\mathrm{l}}a}, \ldots, \tfrac{(2^{\mathrm{l}}a-1)\pi}{2^{\mathrm{l}}a}\}, \ h_{ji} \in \{0, 1, \ldots, 2^{\mathrm{l}}d - 1\},$$
$$\text{and } \delta = diag/(2^{\mathrm{l}}d - 1)\}.$$

Note 1 A line with $d = l_\theta$ intersects *rect* at the point $(X_{(1)}, Y_{(1)})$, if $0 \le \theta \le \pi/2$, and at the point $(X_{(N)}, Y_{(1)})$, if $\pi/2 \le \theta < \pi$. The parameter $h_{ji}\delta$, $0 \le h_{ji}\delta \le diag$, is sometimes referred to as the *offset* value.

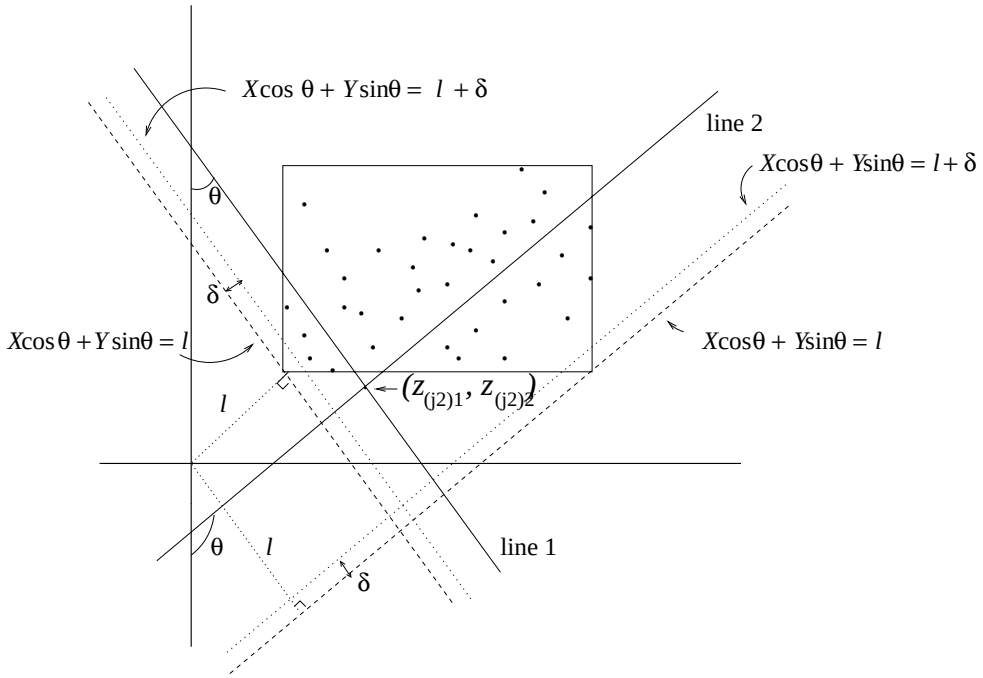


Figure 3.1 A data set with functions from $\mathcal{L}_2^r$

Note 2 For fixed k, i, and θ_{ji}, the lines $\{L_{ji} : h_{ji} = 0, \ldots, 2^{l_d} - 1\}$, are parallel and evenly spaced across the area covered by *rect*.

Note 3 If $l^*_a \geq l_a$ and $l^*_d \geq l_d$, then $\mathcal{L}_k^r \subset \mathcal{L}_k^{r*}$ where $\mathcal{L}_k^r$ corresponds to l_a and l_d and $\mathcal{L}_k^{r*}$ corresponds to l^*_a and l^*_d.

Figure 3.1 represents a sample data set with several functions from $\mathcal{L}_2^r$. For sake of clarity, $\mathcal{L}_k^r$ is henceforth specified as $\mathcal{L}_k^r(\Theta, \mathcal{H})$ where θ has Θ possible values and h has $\mathcal{H}$ possible values (*note that* $\Theta = 2^{l_a}$ *and* $\mathcal{H} = 2^{l_d}$) and specify $L_{j.}(\boldsymbol{X})$ as $L(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.})$. The goal is to use genetic algorithms to find the minimum least-squares k-piecewise linear function where k is known. This is possible if and only if

1. The search space $\mathcal{L}_k^r(\Theta, \mathcal{H})$ contains the optimal solution, i.e., a minimum least-squares function.

2. The algorithm converges to this optimal solution

as $\Theta \rightarrow \infty$ and $\mathcal{H} \rightarrow \infty$. First it is determined whether these conditions are met when $k = 1$.

3.2.3 Case $k = 1$

In the case where $k = 1$, it is desired that the optimal string represent the minimum least-squares line, i.e., the line whose fitted values $\hat{\boldsymbol{Y}}_0$ satisfy

$$(\sum_{i=1}^{N}(\hat{Y}_{0i} - Y_i)^2)^{-1} = \max_j\{(\sum_{i=1}^{N}(\hat{\hat{Y}}_{ji} - Y_i)^2)^{-1} : \hat{\hat{\boldsymbol{Y}}}_j = L(\theta_{j1}, h_{j1})|_{\boldsymbol{X}},\ L_{j1} \in \mathcal{L}_1\}. \tag{3.3}$$

Hence the fitness function fit to be maximized is $(\sum_{i=1}^{N}(\hat{Y}_{0i} - Y_i)^2)^{-1}$. The least-squares line is known as $\hat{\boldsymbol{Y}} = \hat{\beta}_1\boldsymbol{X} + \hat{\beta}_0$, where

$$\hat{\beta}_1 = \frac{\frac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X})(Y_i - \bar{Y})}{\frac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X})^2} \qquad \hat{\beta}_0 = \bar{Y} - \hat{\beta}_1\bar{X} \qquad \bar{X} = \frac{1}{N}\sum_{i=1}^{N}X_i \qquad \bar{Y} = \frac{1}{N}\sum_{i=1}^{N}Y_i$$

[146]. Note that the least-squares line intersects *rect* since it passes through the point $(\bar{X}, \bar{Y})$. Let

$$\begin{aligned}\mathcal{L}_1^r(\Theta, \mathcal{H}) = \{L(\theta_{j1}, h_{j1}) : L(\theta_{j1}, h_{j1}) \in \mathcal{L}_1,\ & L(\theta_{j1}, h_{j1}) \text{ is of the form} \\ & X\cos\theta_{j1} + Y\sin\theta_{j1} = l_{\theta_{j1}} + h_{j1}\delta, \text{ where} \\ & \theta_{j1} \text{ is one of } \Theta \text{ values, } h_{j1} \text{ is one of } \mathcal{H} \text{ values}\}\end{aligned}$$

and let

$$\begin{aligned}\mathcal{B}_1 = \{L(\theta_{m1}, h_{m1}) : L(\theta_{m1}, h_{m1}) \in \mathcal{L}_1,\ \exists\ (X_r, Y_r) \in \mathit{rect} \text{ satisfying } & L(\theta_{m1}, h_{m1}), \\ & 0 \leq \theta_{m1} < \pi,\ h_{m1} \in \mathcal{R}\}.\end{aligned}$$

Figures 3.2 and 3.3 show lines from $\mathcal{L}_1^r(\Theta, \mathcal{H})$ for a sample data set. It shall be proven

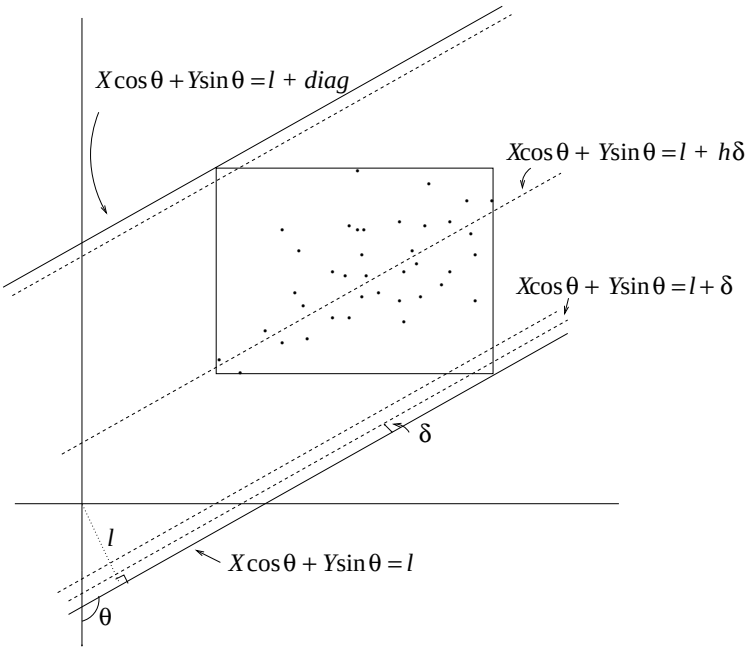


Figure 3.2 A data set with lines from $\mathcal{L}_1^r(\Theta, \mathcal{H})$, $\theta_{j1} > \pi/2$

that the class $\mathcal{L}_1^r(\Theta, \mathcal{H})$ will contain the least-squares line as $\Theta \to \infty$ and $\mathcal{H} \to \infty$.

For simplicity, let $\rho = l_\theta + h\delta$ for given θ and h.

PROPOSITION 3.1 Let $L(\theta_{m1}, h_{m1}) \in \mathcal{B}_1$. Let $\epsilon > 0$. Then $\exists\ (\Theta_\epsilon, \mathcal{H}_\epsilon) :\ \forall\ \Theta > \Theta_\epsilon$ and $\mathcal{H} > \mathcal{H}_\epsilon$, $\exists\ L(\theta, h)$*:*

1. $L(\theta, h) \in \mathcal{L}_1^r(\Theta, \mathcal{H})$
2. $\mid \theta - \theta_{m1} \mid < \epsilon/2$ and $\mid \rho - \rho_{m1} \mid < \epsilon/2$

Proof: Let $L(\theta_{m1}, h_{m1}) \in \mathcal{B}_1$ and $\epsilon > 0$ be given. Choose $\Theta_\epsilon : \pi/2^1 a = \pi/\Theta_\epsilon < \epsilon/2$. Similarly, choose $\mathcal{H}_\epsilon :\ \delta = diag/(2^1 d - 1) = diag/(\mathcal{H}_\epsilon - 1) < \epsilon/2$. Then $\exists\ L(\theta, h) \in$

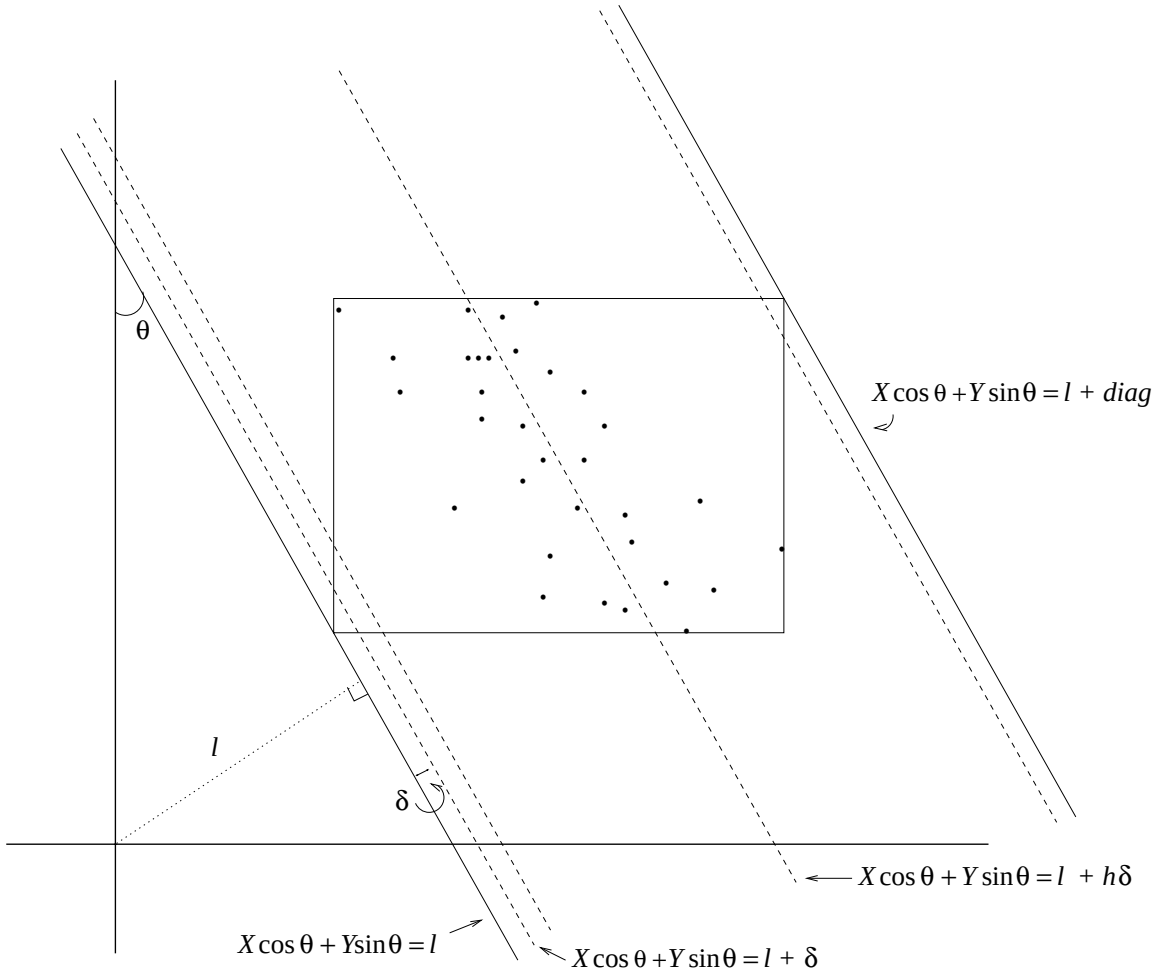


Figure 3.3 A data set with lines from $\mathcal{L}_1^r(\Theta, \mathcal{H})$, $\theta_{j1} < \pi/2$

$\mathcal{L}_1^r(\Theta_\epsilon, \mathcal{H}_\epsilon)$: *2.* is satisfied. By Note 3.2.2 above, if $L(\theta, h) \in \mathcal{L}_1^r(\Theta_\epsilon, \mathcal{H}_\epsilon)$, then $L(\theta, h) \in \mathcal{L}_1^r(\Theta, \mathcal{H})$ $\forall$ $\Theta > \Theta_\epsilon$ and $\forall$ $\mathcal{H} > \mathcal{H}_\epsilon$. Hence *1.* is satisfied. ♠

PROPOSITION 3.2 For each $\epsilon > 0$, $\exists$ $(\Theta_\epsilon, \mathcal{H}_\epsilon)$: for all $\Theta > \Theta_\epsilon$ and for all $\mathcal{H} > \mathcal{H}_\epsilon$, given any $L(\theta_{m1}, h_{m1}) \in \mathcal{B}_1$, $\exists$ $L(\theta, h)$:

1. $L(\theta, h) \in \mathcal{L}_1^r(\Theta, \mathcal{H})$

2. $| \theta - \theta_{m1} | < \epsilon/2$ and $| \rho - \rho_{m1} | < \epsilon/2$

Proof: Let $\epsilon > 0$ be given. Choose $\Theta_\epsilon : \pi/2^1 a = \pi/\Theta_\epsilon < \epsilon/2$. Then for $\theta_{\epsilon_{(i)}} \in$

$\left\{0, \dots, \frac{(\Theta_\epsilon - 1)\pi}{\Theta_\epsilon};\ \theta_{\epsilon_{(i-1)}} \le \theta_{\epsilon_{(i)}}\ \forall\ i,\ i = 1, \dots, \Theta_\epsilon\right\}$, we have $|0 - \theta_{\epsilon_{(1)}}| < \epsilon/2$,

$|\ \theta_{\epsilon_{(1)}} - \theta_{\epsilon_{(2)}}\ | < \pi/2, \dots, |\ \theta_{\epsilon_{(\Theta_\epsilon)}} - \pi\ | < \pi/2$. So given any $L(\theta_{m1}, h_{m1}) \in \mathcal{B}_1$ we can

choose Θ_ϵ so that $\exists\ \theta_{\epsilon_{m1}} \in \{0, \dots, \frac{(\Theta_\epsilon - 1)\pi}{\Theta_\epsilon}\} : |\ \theta_{m1} - \theta_{\epsilon_{m1}}\ | < \epsilon/2$.

For any angle $\theta_{\epsilon_n} \in \{[\frac{n\pi}{\Theta_\epsilon}, \frac{(n+1)\pi}{\Theta_\epsilon}],\ n = 0, \dots, \Theta_\epsilon - 2\}$, the corresponding

$\rho_{\epsilon_n} \in \{[\gamma_{\epsilon_{n_1}}, \gamma_{\epsilon_{n_2}}],\ l_{\theta_{\epsilon_n}} \le\ \gamma_{\epsilon_{n_1}} \le \gamma_{\epsilon_{n_2}} \le diag\}$. Find

$$\nu = \max_n \{\sup_{\epsilon_n} \{|\ \gamma_{\epsilon_{n_2}} - \gamma_{\epsilon_{n_1}}\ |,\ n = 0, \dots, \Theta_\epsilon - 2\}\}. \tag{3.4}$$

Choose $\mathcal{H}_\epsilon :\ \nu/\mathcal{H}_\epsilon < \epsilon/4$ and $\mathcal{H}_\epsilon = 2^y$ for some $y \in \mathcal{R}$. Then given any $L(\theta_{m1}, h_{m1}) \in \mathcal{B}_1$

we can choose $\mathcal{H}_\epsilon$ so that $\exists\ h_{\epsilon_{m1}} \in \{0, \dots, 2^{\mathcal{H}_\epsilon} - 1\} : |\ \rho_{\epsilon_{m1}} - \rho_{m1}\ | < \epsilon/2$.

Hence given any $\epsilon > 0$ and $L(\theta_{m1}, h_{m1}) \in \mathcal{B}_1$ we can find Θ_ϵ and $\mathcal{H}_\epsilon$ so that

$\exists\ L(\theta_{\epsilon_{m1}}, h_{\epsilon_{m1}}) \in \mathcal{L}_1^r(\Theta_\epsilon, \mathcal{H}_\epsilon) :|\ \theta - \theta_{m1}\ | < \epsilon/2$ and $|\ \rho - \rho_{m1}\ | < \epsilon/2$.

If $L(\theta_{\epsilon_{m1}}, h_{\epsilon_{m1}}) \in \mathcal{L}_1^r(\Theta_\epsilon, \mathcal{H}_\epsilon)$, then $L(\theta_{\epsilon_{m1}}, h_{\epsilon_{m1}}) \in \mathcal{L}_1^r(\Theta, \mathcal{H})\ \forall\ \Theta > \Theta_\epsilon$ and $\forall\ \mathcal{H} > \mathcal{H}_\epsilon$

by Note 3 of Section 3.2.2. Hence *1.* and *2.* are satisfied. ♠

Let $\{\Theta_i,\ i = 1, 2, \dots\}$ and $\{\mathcal{H}_i,\ i = 1, 2, \dots\}$ represent the possible values of Θ and $\mathcal{H}$. Let $L(\theta^*_{m1}, h^*_{m1})$ be the best line in $\mathcal{B}_1$ and let $L(\theta^*_{i1}, h^*_{i1})$ be the best line in $\mathcal{L}_1^r(\Theta_i, \mathcal{H}_i)$. We would like $\theta^*_{i1} \to \theta^*_{m1}$ and $h^*_{i1} \to h^*_{m1}$ (hence $\rho^*_{i1} \to \rho^*_{m1}$) as $i \to \infty$. To show this, we need one final result.

Define $\mathcal{C}_1 = \bigcup_{i=1}^{\infty} \mathcal{L}_1^r(\Theta_i, \mathcal{H}_i)$. Note $\mathcal{B}_1 \subseteq \mathcal{C}_1$.

THEOREM 3.1 For each $i = 1, 2, \dots,$ let $L(\theta_{n_i}, h_{n_i}) \in\ \mathcal{L}_1^r(\Theta_i, \mathcal{H}_i) :\ \theta_{n_i} \to \theta_{\lim}$ and

$h_{n_i} \to h_{\lim}$ for some $\theta_{\lim},\ 0 \le \theta_{\lim} < \pi,$ and $h_{\lim},\ 0 \le h_{\lim} < \infty$. Let

$$\mathcal{D}_1 = \{L(\theta_{\lim}, h_{\lim}) : \exists \text{ a sequence } \{L(\theta_{n_i}, h_{n_i})\}_{i=1}^{\infty},\ L(\theta_{n_i}, h_{n_i}) \in \mathcal{L}_1^r(\Theta_i, \mathcal{H}_i),$$
$$\text{such that } \theta_{n_i} \to \theta_{\lim} \text{ and } h_{n_i} \to h_{\lim}\}.$$

Then the best line in $\mathcal{B}_1$ is the best line in $\mathcal{D}_1$.

Proof: Note that $\mathcal{C}_1 \subseteq \mathcal{D}_1$ and $\mathcal{B}_1 \subseteq \mathcal{D}_1$. Since the minimal least squares line must pass through $(\bar{X}, \bar{Y})$ and $(\bar{X}, \bar{Y}) \in rect$, the best line in $\mathcal{B}_1 \equiv$ the best line in $\mathcal{D}_1$.♠

For the following claim, it is assumed that the optimal line (i.e., the line which maximizes the fitness function or, in this case, minimizes the least squares error) is unique and that the fitness function $fit : [0, \pi] \times [-M, M]$ is continuous where, for any line in $\mathcal{D}_1$, the distance of the line from the origin is less than M.

PROPOSITION 3.3 Let $L(\theta_{m1}^*, h_{m1}^*)$ be the best line in $\mathcal{D}_1$ and for each i, $i = 1, 2, \ldots$, let $L(\theta_{i1}^*, h_{i1}^*)$ be the best line in $\mathcal{L}_1^r(\Theta_i, \mathcal{H}_i)$. Then $\theta_{i1}^* \to \theta_{m1}^*$ and $h_{i1}^* \to h_{m1}^*$ as $i \to \infty$.

Proof: Let $L(\theta_{m1}^*, h_{m1}^*)$ be the best line in $\mathcal{D}_1$, i.e., if $L(\theta_{m1}, h_{m1}) \in \mathcal{D}_1$ and $L(\theta_{m1}, h_{m1}) \neq L(\theta_{m1}^*, h_{m1}^*)$ then $fit(\theta_{m1}, h_{m1}) < fit(\theta_{m1}^*, h_{m1}^*)$.

Let $\mathcal{E}_i = \{(\theta_{m1}, h_{m1}) : \mathrm{d}((\theta_{m1}, h_{m1}), (\theta_{m1}^*, h_{m1}^*)) < 1/j_i\}$ where d represents the Euclidean distance between (θ_{m1}, h_{m1}) and $(\theta_{m1}^*, h_{m1}^*)$ in the (θ, h)-plane, j_i is chosen so that if $(\theta_{i1}, h_{i1}) \in \mathcal{E}_i$ and $(\theta_{j1}, h_{j1}) \in \mathcal{E}_i^c$ then $fit(\theta_{i1}, h_{i1}) > fit(\theta_{j1}, h_{j1})$, and $j_i \to \infty$ as $i \to \infty$. Such sets $\mathcal{E}_i$ exist since the optimum is unique*.

Note that for each $i\ \exists\ \epsilon_i > 0 :\ L(\theta_{i1}^*, h_{i1}^*) \in \mathcal{L}_1^r(\Theta_{\epsilon_i}, \mathcal{H}_{\epsilon_i}),\ |\ \theta_{i1}^* - \theta_{m1}^*\ | < \epsilon_i/2,$
$|\ \rho_{i1}^* - \rho_{m1}^*\ | < \epsilon_i/2,$ and $(\theta_{i1}^*, h_{i1}^*) \in \mathcal{E}_i$. The best line in $\mathcal{L}_1^r(\Theta_{\epsilon_i}, \mathcal{H}_{\epsilon_i}) \in \mathcal{E}_i$ and the best line in $\mathcal{L}_1^r(\Theta, \mathcal{H}),\ \forall\ \Theta > \Theta_{\epsilon_i}$, and $\forall\ \mathcal{H} > \mathcal{H}_{\epsilon_i}$, will also be in $\mathcal{E}_i$ (the sets $\mathcal{E}_i,\ i = 1, 2, \ldots,$ are nested sets).

Let the best line in $\mathcal{L}_1^r(\Theta_{\epsilon_i}, \mathcal{H}_{\epsilon_i})$ be $L(\theta_{\epsilon_i}^*, h_{\epsilon_i}^*)$. Note that $\theta_{\epsilon_i}^* \to \theta_{m1}^*$ and $h_{\epsilon_i}^* \to h_{m1}^*$ as

$i \to \infty$. That is, if $L(\theta^*, h^*)$ is the best line in $\mathcal{L}_1^r(\Theta, \mathcal{H})$, then $\theta^* \to \theta^*_{m1}$ and $h^* \to h^*_{m1}$ as $\Theta \to \infty$ and $\mathcal{H} \to \infty$. ♠

*In the above proof it was stated that the sets $\mathcal{E}_i$, $i = 1, 2, \ldots$, exist because the optimum is assumed to be unique. We now prove this.

PROPOSITION 3.4 Define $\mathcal{E}_i = \{(\theta_{m1}, h_{m1}) : \mathrm{d}((\theta_{m1}, h_{m1}), (\theta^*_{m1}, h^*_{m1})) < 1/j_i\}$ where j_i is chosen so that if $(\theta_{i1}, h_{i1}) \in \mathcal{E}_i$ and $(\theta_{c1}, h_{c1}) \in \mathcal{E}_i^c$ then $fit(\theta_{i1}, k_{i1}) > fit(\theta_{c1}, k_{c1})$, and $j_i \to \infty$ as $i \to \infty$. Assume that $fit : [0, \pi] \times [-M, M]$ is continuous and has a unique. maximum. Then such sets $\mathcal{E}_i$ exist.

Proof: We prove this by contradiction. For each i, $\mathcal{E}_i$ constitutes an open disk containing $(\theta^*_{m1}, h^*_{m1})$. From topology [56], if $\mathcal{A}$ is any open set containing $(\theta^*_{m1}, h^*_{m1})$ then $fit(\mathcal{A}) \subseteq [fit(\theta^*_{m1}, h^*_{m1}) - \epsilon, fit(\theta^*_{m1}, h^*_{m1}) + \epsilon]$ for some $\epsilon > 0$.

So suppose not.

Then for all open sets $\mathcal{A}$ containing $(\theta^*_{m1}, h^*_{m1})$, if $(\theta_a, h_a) \in \mathcal{A}$ then $fit(\theta_a, h_a) \leq fit(\theta^c_a, h^c_a)$ for some point $(\theta^c_a, h^c_a) \notin \mathcal{A}$. But $fit(\mathcal{E}_i) \to fit(\theta^*_{m1}, h^*_{m1})$ as $i \to \infty$, where

$$fit(\theta^*_{m1}, h^*_{m1}) = \max_m \{fit(\theta_m, h_m); L(\theta_m, h_m) \in \mathcal{D}_1,\ m = 1, 2, \ldots\}$$

and $(\theta^*_{m1}, h^*_{m1})$ is unique. Contradiction. ♠

3.2.3.1 Remarks

1. If both Θ and $\mathcal{H}$ are large, so that $\mid \theta_i - \theta_{i-1} \mid$ and $\mid h_i - h_{i-1} \mid$ are both small, then the maximal line $L(\theta^*, h^*)$ will be close to the optimal line $L(\theta^*_{m1}, h^*_{m1})$.

2. In developing the genetic algorithm, it seems logical to start with an initial choice for $(\Theta, \mathcal{H})$ and run the algorithm for a finite number of iterations, resulting in an

approximation $L(\Theta^{*a}, \mathcal{H}^{*a})$ of $L(\Theta^*, \mathcal{H}^*)$. If Θ and/or $\mathcal{H}$ are/is small, it is possible for $L(\Theta^{*a}, \mathcal{H}^{*a})$ to be close to $L(\Theta^*, \mathcal{H}^*)$ in terms of probability but not close to $L(\Theta^*, \mathcal{H}^*)$ or $L(\theta^*_{m1}, h^*_{m1})$ in terms of Euclidean distance. Since it is unknown whether given values for Θ and $\mathcal{H}$ are 'small' or 'large', initially one is searching, given $(\Theta, \mathcal{H})$, for a $L(\Theta^{*a}, \mathcal{H}^{*a})$ that is close to $L(\Theta^*, \mathcal{H}^*)$ in terms of probability, and then chooses subsequent $(\Theta_{i}, \mathcal{H}_{i})$ so that the approximations $L(\Theta^{*a}_i, \mathcal{H}^{*a}_i)$ move closer to $L(\theta^*_{m1}, h^*_{m1})$ in terms of Euclidean distance.

The above theory is now extended to the case where the number of lines $k > 1$, k known.

3.2.4 Case $k = k^*$, k^* known, $k^* > 1$

The use of genetic algorithms to fit the minimum least-squares k-piecewise linear function to a data set $(\boldsymbol{x}, \boldsymbol{y})$, where $k = k^*$, k^* known, is considered. The fitted values $\hat{Y}_{0m}$ of the optimal function maximize the fitness function, ie.,

$$(\textstyle\sum_{i=1}^{k^*} \sum_{m=N_{0(i-1)}}^{N_{0i}-1} (\hat{Y}_{0m} - Y_m)^2)^{-1} =$$

$$\max_j \{(\sum_{i=1}^{k^*} \sum_{m=N_{j(i-1)}}^{N_{ji}-1} (\hat{\hat{Y}}_{jm} - Y_m)^2)^{-1} : \hat{\hat{\mathbf{Y}}}_j = L(\boldsymbol{\theta}_{j\cdot}, \boldsymbol{h}_{j\cdot}) \mid_{\boldsymbol{X}},\ L(\boldsymbol{\theta}_{j\cdot}, \boldsymbol{h}_{j\cdot}) \in \mathcal{L}^r_{k^*}\} \tag{3.5}$$

where

$$L(\boldsymbol{\theta}_{j\cdot}, \boldsymbol{h}_{j\cdot})|_{\boldsymbol{X}} = L(\theta_{ji}, h_{ji})|_X \quad \text{for} \quad X_{N_{j(2(i-1))}} \le X \le X_{N_{j(2i-1)}},\ X_{N_{j(0)}} = X_{(1)},$$
$$X_{N_{j(2k^*-1)}} = X_{(N)},\ X_{N_{j(i+1)}} = X_{N_{j(i)}} + 1 \quad \text{for} \quad i = 1, \ldots, k^*.$$

Note that the knot sequence $((Z_{(j1)1}, Z_{(j1)2}), \ldots, (Z_{(j(k^*+1))1}, Z_{(j(k^*+1))2}))$ satisfies $X_{N_{j(2i-1)}} \leq Z_{(j(i+1))1} \leq X_{N_{j(2i)}}$ for $i = 1, \ldots, k^* - 1$, $Z_{(j1)1} = X_{N_{j(0)}}$, $Z_{(j(k^*+1))1} = X_{N_{j(2k^*-1)}}$, and $\boldsymbol{X}_{N_j} = \{X_{N_{j(0)}}, X_{N_{j(1)}}, \ldots, X_{N_{j(2k^*-1)}}\}$ depends on the function $L(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.})$.

Our search space is

$$\mathcal{L}^r_{k^*} = \{L_{j.}(\boldsymbol{X}) : L_{j.}(\boldsymbol{X}) \in \mathcal{L}_{k^*},$$

$$L_{ji} \text{ is of the form } X\cos\theta_{ji} + Y\sin\theta_{ji} = l_{\theta_{ji}} + h_{ji}\delta \ \ \forall \ \ i = 1, \ldots, k^*,$$

$$\theta_{ji} \in \{0, \tfrac{\pi}{2^{\mathrm{l}}a}, \tfrac{2\pi}{2^{\mathrm{l}}a}, \ldots, \tfrac{(2^{\mathrm{l}}a-1)\pi}{2^{\mathrm{l}}a}\}, \ h_{ji} \in \{0, 1, \ldots, 2^{\mathrm{l}}d - 1\},$$

$$\text{and } \delta = diag/(2^{\mathrm{l}}d - 1)\}.$$

We will only consider those $L_{j.}(\boldsymbol{X}) \in \mathcal{L}_{k^*}$:

1. $-\cos\theta_{j(i)}/\sin\theta_{j(i)} \neq -\cos\theta_{j(i+1)}/\sin\theta_{j(i+1)} \ \forall\, i = 1, \ldots, k^*$ (no adjacent parallel lines).

2. If $Z_{(j1)}, \ldots, Z_{(j(k^*-1))}$ are the intersection points of $L_{j.}(\boldsymbol{X})$, then $X_{N_{j(2i-1)}} \leq Z_{(ji)} \leq X_{N_{j(2i)}}$ for $i = 1, \ldots, (k^* - 1)$.

As before, let $L_{j.}(\boldsymbol{X})$ be denoted as $L(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.})$.

The theory for the case $k = 1$ can be extended to this case. Each string can be designed to represent an k^*-piecewise function $L(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.}) \in \mathcal{L}^r_{k^*}$ satisfying the above assumptions; note that each string will resemble a combination of k^* strings from the $k = 1$ case. For example, if $k^* = 3$, $\mathrm{l}_a = 3$, and $\mathrm{l}_d = 5$, then a string may look like that represented in Figure 3.4. A similar GA optimization procedure is then employed to find the string which represents the minimal least-squares k^*-piecewise function.

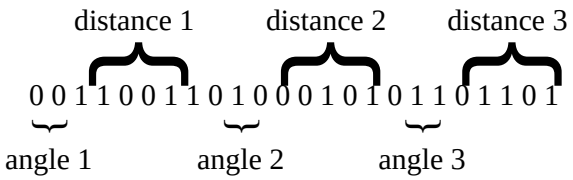


Figure 3.4 A string from $\mathcal{L}_3^r$ when $\mathrm{l}_a = 3$ and $\mathrm{l}_d = 5$

The optimization procedure can alternatively be viewed as a two-step process: for each possible choice of $\boldsymbol{X}_{N_j}$, find the optimal choice for $L(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.})$, say, $L^{\dagger}(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.})$. Then, from the set of all functions $\{L^{\dagger}(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.})\}_{j\geq 1}$, select the optimal function, say, $L^{\ddagger}(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.})$. If we let $\{\boldsymbol{X}_{N_j}\}_{j\geq 1}$ be the set of possible values for $\boldsymbol{X}_{N_j}$ and let $\{L(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.}))\}_{N_j}$ denote the set of all k^*-piecewise functions whose pieces intersect in such a way that $\boldsymbol{X}_{N_j}$ satisfies the above, then

$$fit(L^{\ddagger}(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.})) = \max_j\{\max\{fit(L(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.}));\ L(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.}) \in \{L(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.})\}_{N_j}\}\}. \tag{3.6}$$

3.2.5 Further Remarks and Discussion

3.2.5.1 Fitness Function

In Equation 3.5 the fitness function for our genetic algorithm was stated as

$$(\textstyle\sum_{i=1}^{k^*}\sum_{m=N_{0(i-1)}}^{N_{0i}-1}(\hat{Y}_{0m} - Y_m)^2)^{-1}.$$

If all of the data points fall on a line (or on several linear segments), it is possible for $\sum_{i=1}^{k^*}\sum_{m=N_{0(i-1)}}^{N_{0i}-1}(\hat{Y}_{0m} - Y_m)^2$ to equal zero. To avoid this case the above fitness function may be modified by the addition of an arbitrary positive constant ϵ, yielding

$$((\textstyle\sum_{i=1}^{k^*}\sum_{m=N_{0(i-1)}}^{N_{0i}-1}(\hat{Y}_{0m} - Y_m)^2) + \epsilon)^{-1}.$$

3.2.5.2 Assumptions

In the above theory, we have assumed that the optimal number of lines is known. However, in most cases, the optimal number of lines is unknown and must be estimated. It may be possible to generalize the above theory to this case by utilizing a genetic algorithm which allows for variable string lengths; see Chapter 4. Then, given a data set, the algorithm could select the optimal number of pieces (from an initial set of possible values) as well as the optimal piecewise function.

3.2.5.3 Curve Fitting

It is well known that piecewise linear functions can be used to approximate a curve to any degree of accuracy. Hence curve fitting can be seen as a generalization of the above problem. Suppose our interest was in fitting the optimal curve to a data set. An approach to this problem may be to apply the above theory, given a set of points, to find the piecewise linear function which best approximates the optimal curve. The quality of the approximation would be influenced by the number of pieces as well as the number of iterations.

3.2.5.4 More than 2 Dimensions

In two dimensions, our interest is in fitting a k-piecewise linear function to a data set $\{(x_1, y_1), \ldots, (x_N, y_N)\}$. A similar problem exists for data in d^* dimensions, $d^* \leq 2$; namely, fitting a k-piecewise hyperplane to a data set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, where $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{id^*})$ for $i = 1, \ldots, N$. To solve this problem using genetic algorithms, we could consider extending the methods outlined above as follows:

Each string would represent an individual solution to the problem, i.e., a k-piecewise hyperplane in d^* dimensions. From geometry we know that a hyperplane in $\mathcal{R}^{d^*}$ may be represented as

$$X_{id^*} \cos\theta_{d^*-1} + \gamma_{d^*-1} \sin\theta_{d^*-1} = c$$

where

- $(X_{i1}, \ldots, X_{id^*})$ is a point on the hyperplane
- $\gamma_{d^*-k} = X_{i(d^*-k)} \cos\theta_{d^*-(k+1)} + \gamma_{d^*-(k+1)} \sin\theta_{d^*-(k+1)}$ for $k = 1, \ldots, d^* - 1$
- θ_{d^*-k} is the angle that the projection of the normal to the $(X_1 - \cdots - X_{d^*-(k-1)})$-plane makes with the $X_{d^*-(k-1)}$ axis, $k = 2, \ldots, d^* - 1$
- θ_{d^*-1} is the angle that the projection of the normal to the hyperplane makes with the X_{d^*} axis
- θ_0 is the angle that the projection of the normal to the X_1-plane makes with the X_1 axis ($\theta_0 = 0$), and
- c is the perpendicular distance of the hyperplane from the origin.

To specify c we use the hyper-rectangle *hrect* containing the points $\{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_N, Y_N)\}$, just as we used the rectangle *rect* containing $(\boldsymbol{X}, \boldsymbol{Y})$ to specify d in $\mathcal{R}^2$. Let l_a be the number of bits used to specify an angle, and l_d be the number of bits used to specify c. Let H_{ji} represent the ith hyperplane, or piece, of the jth k-piecewise hyperplane. Then for a given set of angles $\Theta_{ji} = (\theta_{ji_0}, \ldots, \theta_{ji_{d^*-1}})$, $\theta_{ji_m} \in \{0, \ldots, \frac{(2^{\mathrm{l}_a}-1)\pi}{2^{\mathrm{l}_a}}\}$ $\forall\ m = 0, \ldots, d^*-1$, we let $c = l_{\Theta_{ji}} + h_{ji}\delta$ where $l_{\Theta_{ji}}$ is the minimum distance of the origin from one of the hyperplanes passing through a vertex of *hrect*, *diag* is the maximum diagonal

of *hrect*, $h_{ji} \in \{0, 1, \ldots, 2^{l}d - 1\}$, and $\delta = diag/(2^{l}d - 1)$. Let $\Gamma_{ji} = (\gamma_{ji_1}, \ldots, \gamma_{ji_{d^*-1}})$. Then our discrete search space $\mathcal{H}_k^r$, like $\mathcal{L}_k^r$, may be written as

$$\mathcal{H}_k^r = \{H_{jk}(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N) : H_{jk}(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N) \in \mathcal{H}_k,\ H_{j_i k} \text{ is of the form}$$

$$X_{id^*} \cos\theta_{ji_{(d^*-1)}} + \gamma_{ji_{(d^*-1)}} \sin\theta_{ji_{(d^*-1)}} = l_{\Theta_{ji}} + h_{ji}\delta \ \ \forall \ \ i = 1, \ldots, k,$$

$$\Gamma_{ji} \text{ and } \Theta_{ji} \text{ are as specified above, } \theta_{ji_m} \in \{0, \ldots, \frac{(2^{l}a - 1)\pi}{2^{l}a}\}$$

$$\forall\ m = 0, \ldots, d^* - 1,\ h_{ji} \in \{0, 1, \ldots, 2^{l}d - 1\}, \text{ and } \delta = diag/(2^{l}d - 1)\}$$

We now use GAs to search $\mathcal{H}_k^r$ for the k-piecewise hyperplane which minimizes the least-squares distance of the points $(\boldsymbol{X}_i, Y_i)$ from the hyperplane.

3.2.5.5 Pattern Classification

Recently, several applications of genetic algorithms in the field of pattern classification have been reported [14, 114, 115]. Classification is the problem of finding a decision boundary that can correctly distinguish between different classes in the feature space. Given a set of data points in $\mathcal{R}^N$, $N \geq 1$, genetic algorithms can be used to perform this task by, for example, allowing each string to represent a decision boundary formed by a set of lines or hyperplanes. A fitness function which takes larger values for smaller numbers of misclassifications is then maximized. Usually the optimal decision boundary is nonlinear so our task is to approximate the optimal boundary with a set of linear segments. The algorithm is run until a decision boundary with an acceptable number of misclassifications is found.

The application of GAs for classification is similar to the application discussed in this paper. Here, each string also represents a set of lines and the string which best approximates the optimal solution is reported as the result. Our interest, however, is

focused on finding a piecewise linear function which will minimize the squared distance of the data points from the function, and *not* on dividing the data into distinct classes.

3.3 Implementation and Results

3.3.1 Case $k = 1$

3.3.1.1 Data

The data set used is from the text *Applied Linear Regression* by S. Weisberg [146]. The set contains 17 data points that were collected in an experiment by James D. Forbes, a Scottish physicist, designed to study the relationship between atmospheric pressure (in Hg.) and boiling point (in F$^\circ$).

3.3.1.2 Genetic Algorithm

We used a fixed population size of $M = 10$ and a string length of $L = 20$, with 8 bits representing θ and 12 bits representing k; note that once l_θ and δ are known, specifying k is equivalent to specifying d. The single-point crossover probability, p, was fixed at 0.8. The mutation probability, q, varied with the iteration number over a range of [0.0015, 0.5], either increasing or decreasing depending on the value of $Nit/Nmax$, where Nit is the current iteration number and $Nmax$ is the maximum number of iterations. $Nmax$ was set at 1500, at which time the maximum fitness value attained and its corresponding string were reported. As stated previously, for a given string $\boldsymbol{S}_j$ and the fitted values $\hat{\boldsymbol{Y}}_{ji}$ of the line it represents, the fitness value is given as

$$fit(\hat{\boldsymbol{Y}}_j) = (\sum_{i=1}^{N} (\hat{Y}_{ji} - Y_i)^2)^{-1}. \tag{3.7}$$

The results of the GA were compared to the results of a simple linear regression program designed to fit the least squares line to the data.

3.3.1.3 Experimental Results

The proposed algorithm was tested on the data described in Section 3.3.1.1 . The results are shown in Table 3.1 and Figure 3.5. The results of the GA are comparable to the results of the least-squares regression program. The disparity between the results of the two methods may be the result of, for example, the algorithm failing to converge (due to an insufficient value for $Nmax$) or lack of precision in the results of the GA (due to insufficient string length).

Table 3.1 Results of Experiment 1

	Nit	function	$\max_j fit(\hat{\mathbf{y}}_j)$
Approx	1500	$\hat{f}(x) = 0.882x - 39.32$	0.450
Actual		$f(x) = 0.895x - 42.14$	0.464

3.3.2 Case $k = k^*,\ k^*$ known, $k^* > 1$

We will demonstrate this case when $k = k^* = 3$.

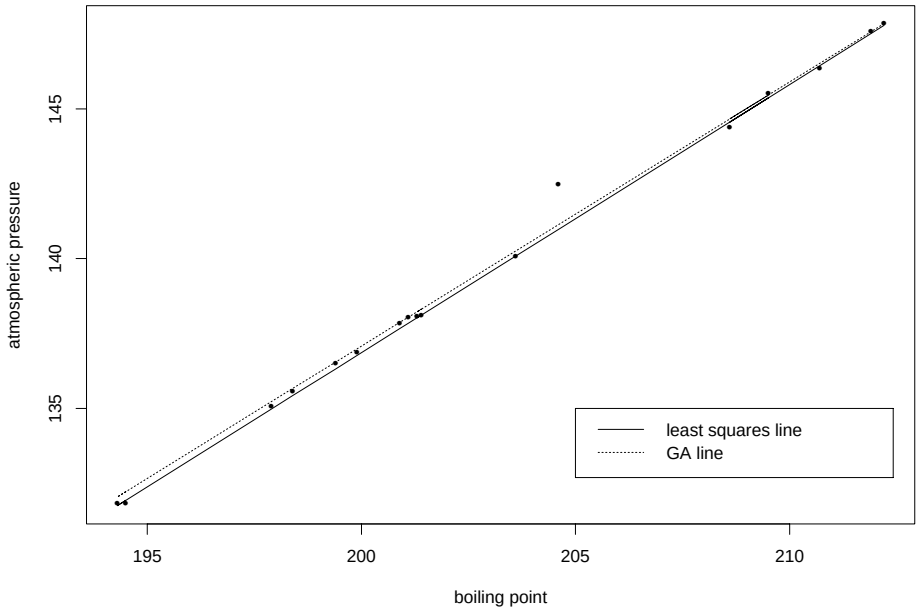


Figure 3.5 Results of Experiment 1

3.3.2.1 Data

An artificial data set was created by first selecting a 3-piecewise linear generating function $g(X)$ whose value is given by

$$g(X) = \begin{cases} 2 + X & -1 \leq X < 0 \\ 2 - X & 0 \leq X < 1 \\ X & 1 \leq X \leq 2 \end{cases} \tag{3.8}$$

where $\boldsymbol{x} = (x_1, \ldots, x_N) = (-1, -0.96, -0.92, \ldots, 1.96, 2)$. For each value $x_i \in \boldsymbol{x}$, the corresponding vector $\boldsymbol{Y}_i = (Y_{i1}, \ldots, Y_{i5})$ consists of 5 values randomly generated from a Normal $(g(x_i), 0.1)$ distribution.

3.3.2.2 Genetic Algorithm

Each string or chromosome represented a 3-piecewise linear function

$$L(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.}) = (L(\theta_{j1}, h_{j1}), L(\theta_{j2}, h_{j2}), L(\theta_{j3}, h_{j3}))$$

over the range of $\boldsymbol{x}$. Let $(Z_{(j1)1}, Z_{(j1)2})$ be the intersection point of L_{j1} and L_{j2} and let $(Z_{(j2)1}, Z_{(j2)2})$ be the intersection point of L_{j2} and L_{j3}. We chose to consider only those piecewise functions $L_{j.}$ for which

- $(Z_{(j1)1}, Z_{(j1)2})$ and $(Z_{(j2)1}, Z_{(j2)2})$ exist (no adjacent parallel pieces)
- $x_{(1)} \leq Z_{(j1)1} \leq Z_{(j2)1} \leq x_{(N)}$.

The fitness value for a given string is then

$$(\sum_{i=1}^{3} \sum_{m=N_{j(2(i-1))}}^{N_{j(2i-1)}} (\hat{Y}_{jm} - Y_m)^2)^{-1} \tag{3.9}$$

where $(N_{j0}, \ldots, N_{j5})$

- $x_{N_{j0}} = x_{(1)}$, $x_{N_{j5}} = x_{(N)}$
- $x_{N_{j1}} \leq Z_{(j1)} \leq x_{N_{j1}+1} = x_{N_{j2}}$; $x_{N_{j3}} \leq Z_{(j2)} \leq x_{N_{j3}+1} = x_{N_{j4}}$

and

$$\hat{Y}_{jm} = \begin{cases} L(\theta_{j1}, h_{j1})|_x & x_{N_{j0}} \leq x \leq x_{N_{j1}} \\ L(\theta_{j2}, h_{j2})|_x & x_{N_{j2}} \leq x \leq x_{N_{j3}} \\ L(\theta_{j3}, h_{j3})|_x & x_{N_{j4}} \leq x \leq x_{N_{j5}}. \end{cases} \tag{3.10}$$

3.3.2.3 Design Modifications

With $k^* = 3$, it became evident that if we set $\mathrm{l}_a = 8$ and $\mathrm{l}_d = 12$ as above, so that each string had length $L = 60$, the size of the population matrix and the number of iterations required for convergence would make our approach computationally expensive. As an alternative, the genetic algorithm was divided into *hierarchical loops*. The modified algorithm can be described as follows:

- Set the global parameters $M = 40$, $p = 0.8$, and Nit = number of iterations per loop = 3000.

- Loop 1

    1. Choose $\mathrm{l}_a = 2$ and $\mathrm{l}_d = 5$ ($L = 21$) so that the 4 angles $a_1, \ldots, a_4$ and 10 h values $h_1, \ldots, h_{10}$ that can be represented are evenly spaced over the ranges $[\pi/4, \pi]$ and $[0, 2^5 - 1]$, respectively.

    2. Generate the initial population $\mathbf{P}$ so that all strings represent functions which meet the above specifications for $Z_{(j1)}$ and $Z_{(j2)}$.

    3. Execute a genetic algorithm beginning with $\mathbf{P}$ to find the optimal string $\boldsymbol{S}_1 = (\boldsymbol{S}_{1a_1}, \boldsymbol{S}_{1d_1}, \ldots, \boldsymbol{S}_{1a_3}, \boldsymbol{S}_{1d_3})$.

    4. Create matrices $\mathbf{P}_{opA}$ and $\mathbf{P}_{opD}$ with three rows, where row i represents the optimal angle or optimal distance of piece i, $i = 1, \ldots, 3$. Place the appropriate sections of $\boldsymbol{S}_1$ into $\mathbf{P}_{opA}$ and $\mathbf{P}_{opD}$.

- Loop 2

    5. Generate a new matrix $\mathbf{P}^*$, also with $\mathrm{l}_a = 2$ and $\mathrm{l}_d = 5$, so given that a_i and h_i are the optimal angle and distance most recently selected for piece i, the 4

possible angles and 10 possible h values for piece i represented in $\mathbf{P}^*$ are evenly spaced over the ranges $[a_i - \pi/(4^2), a_i + 2\pi/(4^2)]$ and $[(2(h_i) - 1)/2,]h_i + 1]$.

6. Repeat step 3 to find $\boldsymbol{S}_2 = (\boldsymbol{S}_{2a_1}, \boldsymbol{S}_{2d_1}, \ldots, \boldsymbol{S}_{2a_3}, \boldsymbol{S}_{1d_3})$.
7. Place the appropriate sections of $\boldsymbol{S}_2$ into $\mathbf{P}_{opA}$ and $\mathbf{P}_{opD}$ (now $\mathbf{P}_{opA_i} = (\boldsymbol{S}_{1a_i}, \boldsymbol{S}_{2a_i})$ and $\mathbf{P}_{opD_i} = (\boldsymbol{S}_{1d_i}, \boldsymbol{S}_{2d_i})$).

- For loop j, j $\geq$ 3, repeat steps 5-7 where angle and distance values are now evenly spaced over $[a_i - \pi/(4^j), a_i + 2\pi/(4^j)]$ and $[(2(h_i) - 1)/2, h_i + 1]$.

- When the desired degree of precision has been reached, the algorithm is stopped and the matrices $\mathbf{P}_{opA}$ and $\mathbf{P}_{opD}$ contain the optimal piecewise function.

Note that the size of the population matrix remains constant regardless of the number of loops being performed. Hence the use of this modified version of a GA avoids the manipulation of large matrices, reducing the required computational resources, without adversely affecting the precision of the resulting solution.

3.3.2.4 Experimental Results

Table 3.2 shows the performance of the proposed GA based algorithm on the artificial data described in Section 3.3.2.1. The fitness value for the generating lines is stated for purpose of comparison.

After 2 loops and only 6000 iterations, the GA converged to a 3-piecewise function with a greater fitness value than the original generating lines. A graph of this function is provided in Figure 3.6.

Table 3.2 Results of Experiment 2

	Nloops	function	$\max_j f(\hat{\boldsymbol{y}}_j)$
Approx	2	$\hat{f}(x) = \begin{cases} x + 1.996 & -1 \le x < 0 \\ 1.991 - x & 0 \le x < 1 \\ x - 0.019 & 1 \le x \le 2 \end{cases}$	0.266836
Generator		$g(x) = \begin{cases} x + 2 & -1 \le x < 0 \\ 2 - x & 0 \le x < 1 \\ x & 1 \le x \le 2 \end{cases}$	0.2633

3.4 Conclusions and Future Research

We have conducted two experiments employing GAs for the fitting of piecewise linear functions to datasets in $\mathcal{R}^2$ where the number of pieces is known. Our results demonstrate that GAs can yield near optimal results at limited computational expense, results that are comparable to the results of existing methods. Although the results of the GA method are competitive, the examples cannot clearly show the advantage of a GA search since they are based on simple parameter spaces. It is not until the work of Chapter 5 that we see empirical results which demonstrate the power of numerical optimization for spline fitting. However, the empirical results of this chapter do demonstrate that developing a GA search algorithm for variable knot spline fitting in more complex parameter spaces is a goal worth pursuing.

Our experiments have involved cases where the number of lines is known. In Chapter 4 we discuss the design, performance, and supporting theory of an algorithm which determines the optimal number of lines as well as their placement. The reader interested in modeling with splines of order greater than two in more complex parameter spaces may choose to move ahead to Chapter 5.

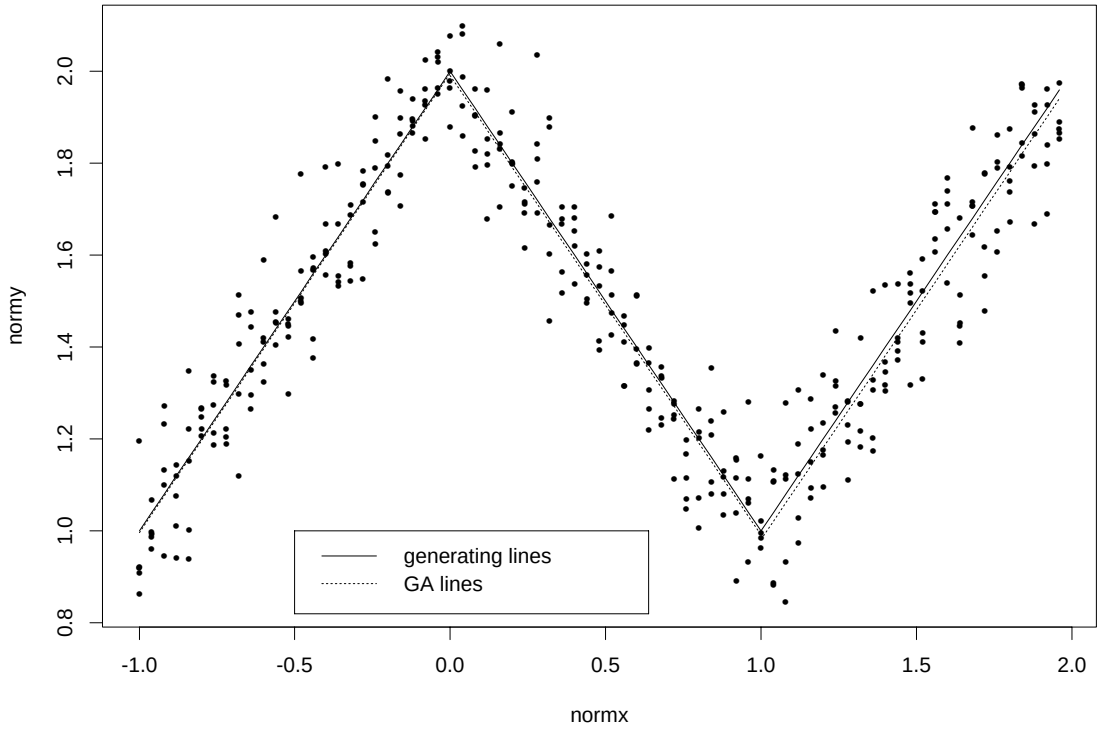


Figure 3.6 Results of Experiment 2

Chapter 4

Piecewise Linear Fitting When the Number of Pieces is Unknown

4.1 Introduction

As has previously been discussed, one of the purposes of statistical data analysis is to determine a functional relationship between some input and output variables given a dataset of noisy observations. The dataset, denoted as $(\boldsymbol{x}, \boldsymbol{y})$, is assumed to be a number of realizations of some underlying process combined with random noise, i.e.

$$\boldsymbol{Y} = f(\boldsymbol{X}) + \boldsymbol{\epsilon}$$

where $\boldsymbol{\epsilon}$ has mean zero. Often the function $f(\boldsymbol{X})$ is assumed to meet certain mathematical requirements, such as continuity or differentiability, and to be of a certain form, such as curvilinear or piecewise linear. The problem of determining $f(\boldsymbol{X})$ given a set of data points and various assumptions is relevant to many application fields including engineering, chemometrics, and materials science [87, 89].

The construction of a functional model for $f(\boldsymbol{X})$, denoted by $\hat{f}(\boldsymbol{X})$, can involve classical statistical tools such as kernel methods and regression splines [61], as well as more recent computational tools such as neural networks, radial basis functions, and genetic algorithms [38, 144, 86]. Unfortunately it is usually the choice of technique, rather than the data or prior process knowledge, that motivates the placement of artificial mathematical restrictions on the final model [35]. It should also be noted that not all techniques can guarantee convergence to a near-optimal $\hat{f}(\boldsymbol{X})$. In the case of a piecewise linear model,

we cannot assume strict differentiability and we wish to optimize the number of pieces as well as their placement in the model. Hence the optimization power of our technique and its assumptions are critical. For these reasons we utilize genetic algorithms as our primary model fitting tool.

In Chapter 3 we explored the use of GAs for piecewise linear fitting where the number of pieces is known. In this chapter we consider how GAs can be employed to fit piecewise linear models to data sets in $\mathcal{R}^2$, where the optimal number of pieces (i.e., knots) as well as their placement are unknown. In Section 4.2 we present the problem and discuss current methods for function approximation. Then in Section 4.3 we introduce genetic algorithms and detail how GAs can be used to fit optimal piecewise-linear functions. After outlining the supporting theory in Section 4.4, several examples are presented with results in Section 4.5. We finish with a brief conclusion in Section 4.6.

The main purpose of this chapter is to demonstrate that GA based methods can yield piecewise linear models, where the number of pieces are unknown, whose quality with respect to a given criterion is comparable to that of existing methods. We also wanted to provide theoretical support for the use of GAs for piecewise linear modeling and introduce a particular model selection criterion which, when combined with GA fitting, yields models that are robust in the presence of outliers. In the empirical results it is noted that traditional methods do well since the parameter space is small and the nonlinear optimization techniques can be given excellent starting values. However, in larger spaces where, for example, it is difficult to determine good starting values, the advantage of a GA search is clearly demonstrated. Hence this chapter shows primarily preparation work for Chapter 5, work of mainly theoretical and not empirical interest. The reader interested in spline fitting in larger spaces may choose to skip to Chapter 5

4.2 Problem Statement and Current Methodology

We assume a data set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, $(x_i, y_i) \in \mathcal{R}^2 \; \forall \; i = 1, \ldots, N, \; 1 \leq N < \infty$, where the values (x_i, y_i) are related by an unknown function f such that $y_i = f(x_i) + \epsilon_i$, where ϵ_i is a random error with zero mean and constant variance. The problem is to fit a model $\hat{f}$ to the data such that

- $\hat{f}$ is a k^*-piecewise linear function where k^* is an unknown positive integer representing the number of pieces, $1 \leq k^* \leq k_{\max}$.
- The knot locations $(\mathrm{Z}_{11}, \mathrm{Z}_{12}), \ldots, (\mathrm{Z}_{(\mathrm{k}^*+1)1}, \mathrm{Z}_{(\mathrm{k}^*+1)2})$ are unknown (the endpoints of $\hat{f}$ are also considered knots).
- $\hat{f}$ maximizes the quality of the fit over all such k-piecewise linear functions, $1 \leq k \leq k_{\max}$, where the quality of the fit is determined by a specified evaluation function $fit(\hat{f}(X))$.

We make no assumptions regarding the smoothness of $\hat{f}$ around its knots.

Specific algorithms for variable knot piecewise constant/linear modeling can be found in the numerical analysis literature, e.g., Baines [11], Tourigny and Baines [138], and Loach and Wathen [96]. Baines' algorithm aims for optimal discontinuous $\mathcal{L}_2$ fits to continuous functions; by concentrating on discontinuous fits it effectively linearizes the problem at hand. Tourigny and Baines [138] proved that under certain assumptions, Baines' algorithm generates an approximation mesh sequence which converges to a locally optimal mesh (for $N \in \mathcal{N}$, a *mesh* is a partition of [0,1] with N interior points $0 = X_0 < X_1 < \ldots < X_N < X_{N+1} = 1$; the approximating space is the subset of $\mathcal{L}_2[0,1]$ consisting of those functions which are polynomials of degree less than r on each subinterval (X_{j-1}, X_j)). Although the adjustment necessary to ensure a continuous fit forces the

abandonment of an optimal fit, and the algorithm can converge to a local optimum, the algorithm does have a nice extension to the two-dimensional case.

Loach and Wathen [96], on the other hand, approach the given problem by utilizing results given in Chui, Smith, and Ward [39] and Barrow, Chui, Smith, and Ward [16] to design a hybrid algorithm for computing the best $\mathcal{L}_2$ linear spline approximation to a given continuous function with a fixed number of free knots. Given its sensitivity to the choice of initial reference (i.e., knot partition), the algorithm utilizes dynamic programming to generate promising starting values. It will also, in most cases, return a solution with distinct, ordered knot points.

Variable knot models have also been fit using Bayesian estimation as opposed to GA optimization [50, 99], with promising results, although, unlike GAs, Bayesian methods cannot guarantee convergence to an optimal solution for a sufficient number of iterations [23].

GAs need only a set of parameters, a way to calculate a response, and an evaluation function (i.e., a measure of quality of fit) to construct a model for a data set. The artificial requirements of differentiability and smoothness imposed on the functional form of the model by the above methods can be discarded [35] and criteria more robust (in the sense of Section 4.4.3) to outliers in comparison to $\mathcal{L}_2$ error can be considered. In the case of piecewise linear models, the power of GA optimization allows variable knot placement as well as a variable number of knots. Related works include those of Karr [87, 86], who has applied GAs to the problem of least squares (LS) and least median squares (LMS) curve fitting (where the specific form of the curve is known (e.g., a polynomial of degree 2) and the data is noiseless), and Vankeerberghen, Smeyers-Verbeke, Leardi, Karr, and Massart [140] who have used GAs to obtain the parameters for specific, known laboratory system

models (e.g., a hyperbolic model) which are nonlinear in the parameters. We intend to utilize GAs for fitting both robust and non-robust optimal piecewise linear functions to data in $\mathcal{R}^2$.

4.3 Genetic Algorithms

As discussed in Chapter 3, *Genetic Algorithms* (GAs) are stochastic search methods which provide a near optimal solution to the evaluation function of an optimization problem [35]. They can be used to search complex, multimodal surfaces via steps which have been designed to mimic the processes of natural genetic systems.

This basic framework can be modified to specifically address a problem of interest by, e.g., the choice of population or the addition of other operators. In the following section we describe one such modified algorithm, as presented in Bandyopadhyay, Murthy, and Pal [13].

4.3.1 Variable Length Genetic Algorithm (VLGA)

Consider the problem of maximizing a function $fit(X),\ X \in \mathcal{V}$, where $\mathcal{V}$ is a finite set and $fit(X) > 0\ \forall\ X \in \mathcal{V}$. Each string $\boldsymbol{S}$, built from members of a finite alphabet $\mathcal{A} = \{\alpha_1, \ldots, \alpha_a\}$, corresponds to a value X in $\mathcal{V}$ and may be written as

$$\boldsymbol{S} = (\gamma_1, \gamma_2, \ldots, \gamma_{k*L});\ \ \gamma_i \in \mathcal{A}\ \ \forall\, i = 1, \ldots, k*L.$$

In our case, $\boldsymbol{S}$ represents a k-piecewise linear function, k unknown, $k \in \mathcal{K} = \{1, \ldots, k_{\max}\}$; L characters (i.e., bits) represent each piece. Hence the string length is not fixed but *variable* - for a given k, the string length is $k * L$. The number of different strings that are possible is $a^{1*L} + a^{2*L} + \cdots + a^{k_{\max}*L}$. A random sample of size M (M even) is

drawn from these possible strings with replacement to form the initial population $\mathbf{P}$. The evaluation or fitness value of each string $\boldsymbol{S}$ is $fit(X)$ where $X \in \mathcal{V}$ is the value represented by $\boldsymbol{S}$.

In order to maintain the same string length, each string in $\mathbf{P}$ of length less than $k_{\max} * L$ is padded with $\star$s so that its length is $k_{\max} * L$. For example, if the strings are binary, $L = 5$, $k = 2$ and $k_{\max} = 3$, then the string

$$0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0 \qquad \text{would become} \qquad 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ \star\ \star\ \star\ \star\ \star.$$

All strings in $\mathbf{P}$, although representing functions with different numbers of pieces, now have length $k_{\max} * L$. Pieces consisting only of $\star$s are defined to have the minimum fitness value.

As the first operation, *selection*, we use *stochastic sampling with replacement* [43], as described in Chapter 2. The resulting mating pool is taken as our new population $\mathbf{P}_1$.

Single point crossover, or *reproduction*, is also used as described in Chapter 2. However, suppose a string $\boldsymbol{S}_1$ with k_0 pieces and a string $\boldsymbol{S}_2$ with k_1 pieces, $k_0 < k_1$, are selected for crossover and the crossover point *pos* is such that $k_0 * L < pos < k_1 * L$, so the crossover point is located in a piece of $\boldsymbol{S}_1$ consisting only of $\star$s but in a piece of $\boldsymbol{S}_2$ consisting of 0s and 1s. The resulting strings $\dot{\boldsymbol{S}}_1$ and $\dot{\boldsymbol{S}}_2$ may be *incomplete*. A string is considered incomplete if it contains either (1) a piece consisting of characters from both {0,1} and {$\star$} or (2) a piece of only 0s and 1s located after a piece of only $\star$s. For example, suppose $h_{\max} = 3,\ L = 3$, and the pair of strings chosen for crossover is

$$0\ 1\ 0\ 1\ 1\ 1\ \#\ \#\ \# \quad \text{and} \quad 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1.$$

If the position chosen for crossover is between the seventh and eighth bits, then after crossover the resulting strings are

$$0\ 1\ 0\ 1\ 1\ 1\ \#\ 0\ 1 \quad \text{and} \quad 1\ 0\ 0\ 0\ 1\ 0\ 1\ \#\ \#.$$

Each string contains one incomplete hyperplane. Incomplete strings are adjusted for after the next operation, mutation.

Once each pair undergoes crossover, the resulting population is denoted $\mathbf{P}_2$. $\mathbf{P}_2$ has M strings, some of which may have also been elements of $\mathbf{P}_2$.

Mutation, which involves the random altering of characters in the chromosomes (strings) of $\mathbf{P}_2$, is altered slightly from that described in Chapter 2. For each character γ_i of every string $\boldsymbol{S}$, the mutation stage consists of

1. Generate a random number rnd_1 from [0,1]
2. If $rnd_1 > p_m$, γ_i is not mutated. If $rnd_1 \leq p_m$,
   (a) Generate another random number rnd_2 from [0,1]. Note that γ_i has two options for mutation, opt_1 and opt_2. For example, if $\gamma_i = 1$, then γ_i can become either 0 or $\star$.
   (b) If $rnd_2 \leq 0.5$, then γ_i becomes opt_1; else, γ_i becomes opt_2.

As explained in Chapter 2, the mutation probability may vary over iterations, e.g., initially taking a high value, then decreasing to a pre-specified minimum, then increasing again in the later stages of the algorithm [9]. The resulting population we denote as $\mathbf{P}_3$.

Note that through mutation, a given string can become any of the $a^{1*L} + \cdots + a^{k_{\max}*L}$ possible strings.

Before moving to the next stage, *elitism* [48], we adjust for any strings in $\mathbf{P}_3$ which are incomplete. If $\boldsymbol{S}_j \in \mathbf{P}_3$ is an incomplete string with pieces $\boldsymbol{S}_j^1, \ldots, \boldsymbol{S}_j^{k_{\max}}$, then for each piece $\boldsymbol{S}_j^i$ consisting of characters from both {0,1} and {$\star$}, generate a random number

rnd_1 from [0,1] and let num be the number of defined bits, i.e., the number of 0s and 1s, in $\boldsymbol{S}_j^i$.

1. If $rnd_1 \leq \frac{num}{k_{\max} * L}$ then for each $\star \in \boldsymbol{S}_j^i$, generate a random number rnd_2 from [0,1]. If $rnd_2 \leq 0.5$ then $\star$ is replaced by 0. Otherwise, $\star$ is replaced by 1.

2. If $rnd_1 > \frac{num}{k_{\max} * L}$ then each 0 or 1 in $\boldsymbol{S}_j^i$ is replaced by $\star$.

All pieces consisting only of $\star$s are then moved to the end of the string and all strings in $\mathbf{P}_3$ are complete.

At this point, *elitism* [48] is performed on $\mathbf{P}_3$ and the resulting population is taken as the new $\mathbf{P}$. The algorithm begins another iteration by returning to the first operation, *selection*.

4.3.2 Remarks

4.3.2.1 Convergence

As described in Chapter 2, Bhandari et. al. [23] have proven theoretically that an elitist genetic algorithm (fixed length strings) will converge to the optimal string as the number of iterations goes to infinity. This convergence is independent of the choice of values for the algorithm parameters (population size (M), probabilities of selection (p_s), crossover (p_c), and mutation (p_m), etc.) although these parameter values do influence the rate of convergence. Note that there is no theory to indicate the number of iterations necessary for convergence, only two popular heuristic stopping rules (see Section 2.2). However, even a compeletely random search will reach the global optimum given a sufficient number of iterations. Hence the above proof shows only that a GA search is not biased in such a way as to prevent it from reaching the global optimum.

4.3.2.2 Flexibility

GAs can be applied to a wide range of optimization problems with little adjustment - in most cases only the interpretation of each string and the fitness function need to be redefined. Thus it is possible to use the same basic algorithm to, for example, fit lines satisfying different optimization criteria to a given dataset.

4.4 Theory of Piecewise Linear Fitting in $\mathcal{R}^2$

4.4.1 Mathematical Formulation

Our stated goal is to fit piecewise linear functions to datasets in $\mathcal{R}^2$. Generally speaking, if we think of a given dataset as a realization of some random variable, we would like our set of lines to represent the center of the density of that random variable.

Let us assume that we have k^* lines, $1 \leq k^* \leq k_{\max}$, $k_{\max}$ is a positive integer. For each j, $j = 1, \ldots, k^*$, let the equation of the j^{th} line be

$$X \cos\theta_j + Y \sin\theta_j = d_j$$

for some $\theta_j \in (0, \pi]$ and $d_j \in \mathcal{R}$. Assume that the j^{th} and $(j+1)^{th}$ lines intersect at the point $(Z_{(j+1)1}, Z_{(j+1)2})$. We denote the first knot as (Z_{11}, Z_{12}) and the last knot as $(Z_{(k^*+1)1}, Z_{(k^*+1)2})$.

Let $\epsilon_0 > 0$ and let the set B_j be expressed as

$$B_j = \left\{ (X,Y) : Y \in \left[\frac{d_j - X\cos\theta_j}{\sin\theta_j} - \epsilon_0, \frac{d_j - X\cos\theta_j}{\sin\theta_j} + \epsilon_0 \right]; \right.$$
$$\left. x \in [Z_{j1}, Z_{(j+1)1}],\ j = 1, \ldots, k^*, \right\}$$

We denote $\bigcup_{j=1}^{k^*} B_j$ as B. The probability density function of (X, Y) on B is denoted by $\alpha : B \to [0, \infty)$, where

$$\alpha(X,Y) = \begin{cases} \alpha_j(X,Y) & X \in [Z_{j1}, Z_{(j+1)1}], \quad \text{for } j = 1, \ldots, k^* \\ 0 & \text{otherwise.} \end{cases}$$

We define our probability measure P as $P(A) = \int_A \alpha(X,Y)\, dX\, dY$ for all Borel $A \subseteq \mathcal{B}(B)$, the Borel σ-field of B.

Let $(X_1, Y_1), (X_2, Y_2), \ldots, (X_N, Y_N)$ be independent, identically distributed random vectors with density α, i.e., there exists a probability space $(\Omega, \mathcal{A}, Q)$ such that $(X_i, Y_i) : (\Omega, \mathcal{A}, Q) \to (B, \mathcal{B}(B), P),\ i = 1, \ldots, N$, where $Q(C) = P((X_i, Y_i)(C))\ \forall\ C \in \mathcal{A}$.

We also assume that

1. For each $j = 1, \ldots, k^*$,

$$\alpha_j(X,Y) = \begin{cases} \alpha_j\left(X, \frac{2(d_j - X\cos\theta_j)}{\sin\theta_j} - Y\right) & X \in [Z_{j1}, Z_{(j+1)1}] \\ 0 & X \notin [Z_{j1}, Z_{(j+1)1}] \end{cases} \tag{4.1}$$

   (i.e., α_j is symmetric about the line $X\cos\theta_j + Y\sin\theta_j = d_j$).

2. $\alpha_j(X,Y) > 0\ \ \forall\ (X,Y) \in B_j\ \ \forall\ j$

3. $\alpha(X,Y)$ and $\alpha_j(X,Y),\ j = 1, \ldots, k^*,$ are continuous.

4. $\int_B \alpha(X,Y)\, dX\, dY = 1$

In the above mathematical formulation, the primary assumption is the assumption of symmetry of the underlying density around a piecewise linear function (assumption

#1). The other stated assumptions are common properties of a continuous probability density function.

4.4.2 Solution Space

In order to represent our solution space, suppose we are given a data set $D = (\boldsymbol{x}, \boldsymbol{y}) = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ which is a realization of the random vectors $\{(X_i, Y_i)\}_{i=1}^{N}$, i.e., $X_i(\omega) = x_i,\ Y_i(\omega) = y_i$ for $\omega \in \Omega,\ i = 1, \ldots, N$. Define $X_{(1)} = \min\{X_i;\ i = 1, \ldots, N\},\ X_{(N)} = \max\{X_i;\ i = 1, \ldots, N\},\ Y_{(1)} = \min\{Y_i;\ i = 1, \ldots, N\},\ Y_{(N)} = \max\{Y_i;\ i = 1, \ldots, N\}$. A GA tries to find an optimal solution over a finite solution space. Thus the solution space, i.e., the collection of all k-piecewise linear functions where $k \in \mathcal{K},\ \mathcal{K} = \{1, \ldots, k_{\max}\}$, must be discretized. To formulate the problem mathematically, we proceed in the following way.

Let L_j denote the straight line

$$X \cos\theta_j + Y \sin\theta_j = d_j$$

where j belongs to an index set. Let $L_{j\cdot}^{k}$ represent a k-piecewise linear function, i.e., $L_{j\cdot}^{k} = \{L_{j1}, L_{j2}, \ldots, L_{jk}\}$ where L_{ji} denotes the i^{th} straight line among the set of k straight lines, $i = 1, \ldots, k$, and each L_{ji} satisfies the following properties:

1. L_{ji} represents the straight line $X \cos\theta_{ji} + Y \sin\theta_{ji} = d_{ji}$ where θ_{ji} $(0 < \theta_{ji} \leq \pi)$ is the polar angle formed by the crossing of the Y-axis and L_{ji} when the polar axis is taken as the Y-axis and the origin is taken as the intersection point between the Y-axis and L_{ji}; d_{ji} is the perpendicular distance of the line from the point (0,0).

2. For every $i,\ i = 1, \ldots, k-1,\ L_{ji}$ and $L_{j(i+1)}$ intersect and the point of intersection is $(Z_{(j(i+1))1}, Z_{(j(i+1))2})$.

3. $Z_{(j1)1} = X_{(1)},\ Z_{(j(k+1))1} = X_{(N)},\ Z_{(ji)1} \leq Z_{(j(i+1))1}\ \ \forall\ i = 1, \ldots, k$

4. $L^k_{j\cdot} = L_{ji}$ if $Z_{(ji)1} \leq x \leq Z_{(j(i+1))1},\ i = 1, \ldots, k.$

The knot locations $\boldsymbol{Z}_{(j\cdot)1} = (Z_{(j1)1} \ldots, (Z_{(j(k+1))1})$ of any $L^k_{j\cdot},\ 1 \leq k \leq k_{\max}$, are not restricted to the set $\{X_1, X_2, \ldots, X_N\}$.

For each $k,\ 1 \leq k \leq k_{\max}$, let $\mathcal{L}_k$ represent the class of all k-piecewise linear functions $L^k_{j\cdot}$ which satisfy the above properties. Then $\bigcup_{k\in\mathcal{K}} \mathcal{L}_k$ is the collection of functions under consideration.

Note that $\bigcup_{k\in\mathcal{K}} \mathcal{L}_k$ is uncountable. In order to obtain a 'near optimal' function from this space, we need to discretize $\bigcup_{k\in\mathcal{K}} \mathcal{L}_k$. We can achieve this by restricting the values of θ and d. Let l_a be the number of bits used to express θ and let l_d be the number of bits used to express d. We restrict θ_{ji} to the values $\{\frac{\pi}{2^{\mathrm{l}}a}, \frac{2\pi}{2^{\mathrm{l}}a}, \ldots, \frac{(2^{\mathrm{l}}a-1)\pi}{2^{\mathrm{l}}a}, \pi\}$. In specifying d_{ji}, we utilize the rectangle *rect* formed by the points $(X_{(1)}, Y_{(1)})$, $(X_{(N)}, Y_{(1)})$, $(X_{(1)}, Y_{(N)})$ and $(X_{(N)}, Y_{(N)})$. Note that *rect* contains the entire given data set. Let *diag* be the length of the diagonal of *rect* and let ℓ_θ be defined as

$$\ell_\theta = \begin{cases} X_{(1)} \cos\theta + Y_{(1)} \sin\theta & \text{if } 0 < \theta < \pi/2 \\ X_{(N)} \cos\theta + Y_{(1)} \sin\theta & \text{if } \pi/2 \leq \theta \leq \pi. \end{cases}$$

For a given $\theta_{ji},\ d_{ji}$ may only take values within the set $\{d_{ji} = \ell_{\theta_{ji}} + h_{ji}\delta :\ \ h_{ji} \in \{0, 1, \ldots, 2^{\mathrm{l}}d - 1\},\ \delta = diag/(2^{\mathrm{l}}d - 1)\}$. Note that a line with $d = l_\theta$ intersects *rect* at the point $(X_{(1)}, Y_{(1)})$, if $0 < \theta \leq \pi/2$, or the point $(X_{(N)}, Y_{(1)})$, if $\pi/2 \leq \theta < \pi$ (the parameter $h_{ji}\delta,\ 0 \leq h_{ji}\delta \leq diag$, is sometimes referred to as the *offset* value).

For each $k,\ 1 \leq k \leq k_{\max}$, let $\mathcal{L}^r_k(\Theta, \mathcal{H})$ denote the finite set of functions in $\mathcal{L}_k$ which satisfy these restrictions, where θ has Θ possible values and h has $\mathcal{H}$ possible values

(*note that* $\Theta = 2^{l}a$ *and* $\mathcal{H} = 2^{l}d$), i.e.,

$$\mathcal{L}_k^r(\Theta,\mathcal{H}) = \{L_{j\cdot}^k \in \mathcal{L}_k : L_{ji} \text{ is of the form } X\cos\theta_{ji} + Y\sin\theta_{ji} = \ell_{\theta_{ji}} + h_{ji}\delta$$
$$\forall\ \ i = 1,\dots,k,\ \theta_{ji} \in \{\tfrac{\pi}{2^{l}a}, \tfrac{2\pi}{2^{l}a},\dots,\tfrac{(2^{l}a-1)\pi}{2^{l}a},\pi\},\ h_{ji} \in \{0,1,\dots,2^{l}d-1\},$$
$$\text{and } \delta = diag/(2^{l}d-1)\}.$$

Hence $\{\mathcal{L}_k^r(\Theta,\mathcal{H}) : l_a = 1,2,\dots\}$ and $\{\mathcal{L}_k^r(\Theta,\mathcal{H}) : l_d = 1,2,\dots\}$ each represent an increasing sequence of nested sets. For sake of clarity, we will henceforth specify $L_{j\cdot}^k$ as $L(\boldsymbol{\theta}_{j\cdot}^k, \boldsymbol{h}_{j\cdot}^k)$ and denote $L_{j\cdot}^k(X)$ as $L(\boldsymbol{\theta}_{j\cdot}^k, \boldsymbol{h}_{j\cdot}^k)(X)$. Note that $\boldsymbol{\theta}_{j\cdot}^k$ and $\boldsymbol{h}_{j\cdot}^k$ represent the vectors $(\theta_{j1},\dots,\theta_{jk})$ and $(h_{j1},\dots,h_{jk})$.

For $L(\boldsymbol{\theta}_{j\cdot}^k, \boldsymbol{h}_{j\cdot}^k) \in \mathcal{L}_k^r(\Theta,\mathcal{H})$ we define

$$E_{\epsilon,L(\boldsymbol{\theta}_{j\cdot}^k,\boldsymbol{h}_{j\cdot}^k),a,b} = \{(X,Y) : Y \in (L(\boldsymbol{\theta}_{j\cdot}^k, \boldsymbol{h}_{j\cdot}^k)(X) - \epsilon, L(\boldsymbol{\theta}_{j\cdot}^k, \boldsymbol{h}_{j\cdot}^k)(X) + \epsilon),\ X \in [a,b]\}$$

for $a < b,\ \epsilon > 0$, and let $E_{\epsilon,L(\boldsymbol{\theta}_{j\cdot}^k,\boldsymbol{h}_{j\cdot}^k)} = \bigcup_b \bigcup_{a<b} E_{\epsilon,L(\boldsymbol{\theta}_{j\cdot}^k,\boldsymbol{h}_{j\cdot}^k),a,b}$.

If we recall the piecewise linear function which lies at the center of the density of (X,Y) and we denote it as ϕ, i.e.,

$$\phi(X) = \begin{cases} \frac{d_j - X\cos\theta_j}{\sin\theta_j} & x \in [Z_{(j)1}, Z_{(j+1)1}],\ \text{ for } j = 1,\dots,k^* \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

then the support B of the density is equivalent to $E_{\epsilon_0,\phi,Z_{(1)1},Z_{(k^*+1)1}}$.

Corresponding to each $E_{\epsilon,L(\boldsymbol{\theta}_{j.}^k,\boldsymbol{h}_{j.}^k)}$ we define the set of lines

$$\mathcal{L}_k^{(\epsilon)} = \{L(\boldsymbol{\theta}_{j.}^k, \boldsymbol{h}_{j.}^k) : P(E_{\epsilon,L(\boldsymbol{\theta}_{j.}^k,\boldsymbol{h}_{j.}^k)}) \geq 0.95\}$$

where $P(A) = \int_A \alpha(X,Y)\,dX\,dY$ for A Borel, $A \subseteq \mathcal{B}(B)$. Just as $\{\mathcal{L}_k^r(\Theta,\mathcal{H}) : \mathrm{l}_a = 1,2,\dots\}$ and $\{\mathcal{L}_k^r(\Theta,\mathcal{H}) : \mathrm{l}_d = 1,2,\dots\}$ represent increasing sequences of sets, so does $\{\mathcal{L}_k^{(\epsilon)} : \epsilon \in \{\epsilon_1,\epsilon_2,\dots\}; \epsilon_i < \epsilon_{i+1},\ i = 1,2,\dots\}$. These definitions involving $E_{\epsilon,L(\boldsymbol{\theta}_{j.}^k,\boldsymbol{h}_{j.}^k)}$ will be used to define our optimization criterion.

Figure 4.1 shows an example dataset where the support $B = B_1 \bigcup B_2$ of the density $\alpha(X,Y)$ of (X,Y) is centered around a 2-piece linear function. A function from $\mathcal{L}_2^r$ and *rect* are also shown.

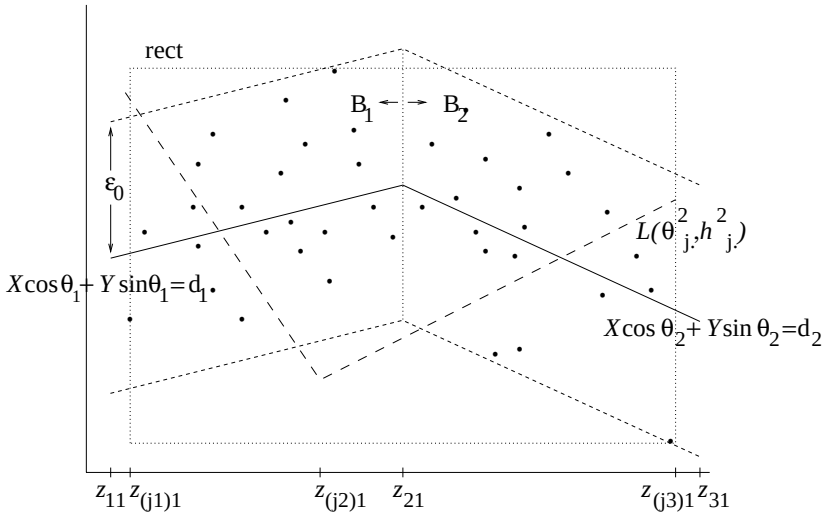


Figure 4.1 Support $B = B_1 \bigcup B_2$ with center lines; Example function from $\mathcal{L}_2^r(\Theta,\mathcal{H})$

4.4.3 Optimization Criterion

The functional model which is chosen to represent a given dataset is often that which minimizes the sum of the squared errors (i.e., the least squares function). However, a criterion based on least squares often yields a solution that is not *robust*, in the sense that outliers in the dataset can pull the least squares function away from most of the data points and hence away from ϕ [80, 140]. For this reason we choose not to use least squares as our optimization criterion. Instead, we note that our concept of a 'fitted' function is one which represents the center of a symmetric density function. If $L(\boldsymbol{\theta}_{j\cdot}^{\dot{k}}, \boldsymbol{h}_{j\cdot}^{\dot{k}})$ represents our 'fitted' function then, given a dataset D, the majority of the data points in D should fall within the region $\{L(\boldsymbol{\theta}_{j\cdot}^{\dot{k}}, \boldsymbol{h}_{j\cdot}^{\dot{k}})(x) - \epsilon, L(\boldsymbol{\theta}_{j\cdot}^{\dot{k}}, \boldsymbol{h}_{j\cdot}^{\dot{k}})(x) + \epsilon),\ X \in [Z_{(j1)1}, Z_{(j(k^*+1))1}]\}$ for some 'small' $\epsilon > 0$. This observation is the basis of our optimization criterion.

Let $\psi_{j\cdot}^{k}$ denote a k-piecewise linear function where each piece $\psi_{ji},\ i = 1, \ldots, k$, satisfies the properties listed above for L_{ji}. For any $\epsilon > 0$, define

$$\mathcal{I}_{\epsilon,\psi_{j\cdot}^{k}}(X,Y) = \begin{cases} 1 & \mid Y - \psi_{j\cdot}^{k}(X) \mid < \epsilon \\ 0 & \text{otherwise.} \end{cases}$$

Our function to be optimized (i.e., fitness function) may then be stated as

$$fit_{\epsilon,N}(\psi_{j\cdot}^{k}) = \sum_{i=1}^{N} \mathcal{I}_{\epsilon,\psi_{j\cdot}^{k}}(X_i, Y_i) + (1 - \frac{k}{k_{\max}}). \tag{4.3}$$

Our solution space $\mathcal{L}^r(\Theta, \mathcal{H}) = \bigcup_{k\in\mathcal{K}} \mathcal{L}_k^r(\Theta, \mathcal{H})$ for some Θ and $\mathcal{H}$ represents the collection of models under consideration. By the continuity of $\alpha(X,Y)$, with $\mathcal{L}^{(\epsilon)} = \bigcup_{k\in\mathcal{K}} \mathcal{L}_k^{(\epsilon)}$, we know there exists some $\epsilon^* : \mathcal{L}^{(\epsilon^*)} \neq \emptyset$ and $\mathcal{L}^{(\epsilon)} = \emptyset\ \forall\ \epsilon < \epsilon^*$ (i.e., there exists an ϵ^* such that a solution exists when $\epsilon = \epsilon^*$ but no solution exists when

$\epsilon < \epsilon^*$). To balance accuracy and robustness, we specify that at least 95% of the data points in D (not necessarily 100%) fall within ϵ^* of the optimal model - our solution belongs to $\mathcal{L}^{(\epsilon^*)}$. The choice of 95% is *heuristic* - it can be altered depending upon the characteristics of the dataset (e.g., the percentage of outliers) and the desired precision. Hence our goal is to use genetic algorithms to find an optimal k^*-piecewise linear function $L(\boldsymbol{\theta}^{k^*}_{j^*.}, \boldsymbol{h}^{k^*}_{j^*.}) \in \ \mathcal{L}^r(\Theta, \mathcal{H}) \bigcap \mathcal{L}^{(\epsilon^*)}$ such that

$$fit_{\epsilon^*,N}(L(\boldsymbol{\theta}^{k^*}_{j^*.}, \boldsymbol{h}^{k^*}_{j^*.})) = \max_{L(\boldsymbol{\theta}^{k}_{j.}, \boldsymbol{h}^{k}_{j.}) \in \mathcal{L}^r(\Theta,\mathcal{H})} fit_{\epsilon^*,N}(L(\boldsymbol{\theta}^{k}_{j.}, \boldsymbol{h}^{k}_{j.})) \tag{4.4}$$

where $k^* = \min\{k \in \mathcal{K} : \mathcal{L}^{(\epsilon^*)}_k \neq \emptyset\}$.

We observe that if we require that the entire dataset D fall within ϵ^* of the optimal function then, for Θ and $\mathcal{H}$ sufficiently large, the optimal solution will correspond to the least squares solution.

This goal can be attained if and only if

1. Our search space $\mathcal{L}^r(\Theta, \mathcal{H})$ contains an optimal solution as $\Theta \to \infty$ and $\mathcal{H} \to \infty$.

2. The algorithm converges to an optimal solution.

We will prove these statements by first assuming $k_{\max} = 1$ and then examining the case of $k_{\max} > 1,\ k^*$ unknown, $k^* \in \mathcal{K}$.

4.4.4 Case $k_{\max} = 1$

4.4.4.1 Optimal Solution in Search Space

In the case where $k_{\max} = 1$ ($k^* = 1$), we would like our optimal string (solution) to represent a line $L(\boldsymbol{\theta}^1_{j^*.}, \boldsymbol{h}^1_{j^*.}) = L(\theta_{j^*1}, h_{j^*1})$ such that

$$fit_{\epsilon^*,N}(L(\theta_{j^*1}, h_{j^*1})) = \max_{L(\theta_{j1},h_{j1})\in\mathcal{L}_1^r(\Theta,\mathcal{H})} fit_{\epsilon^*,N}(L(\theta_{j1}, h_{j1})).$$

Recall $\mathcal{L}_1^r(\Theta,\mathcal{H}) = \{L(\theta_{j1}, h_{j1}) \in \mathcal{L}_1 :\ L(\theta_{j1}, h_{j1})$ is of the form

$$X\cos\theta_{j1} + Y\sin\theta_{j1} = \ell_{\theta_{j1}} + h_{j1}\delta,$$

where θ_{j1} is one of Θ values, h_{j1} is one of $\mathcal{H}$ values$\}$.

Let $\mathcal{Q}_1 = \{L(\theta_{m1}, h_{m1}) : L(\theta_{m1}, h_{m1}) \in \mathcal{L}_1,\ \exists\ (X_r, Y_r) \in rect$ satisfying $\ L(\theta_{m1}, h_{m1}),$

$$0 < \theta_{m1} \le \pi,\ h_{m1} \in \mathcal{R}\}.$$

$\mathcal{Q}_1$ represents the set of all lines in $\mathcal{L}_1$ which intersect *rect*.

Recall that our goal in this section is to show that our search space contains an optimal solution. In order to reach this goal we will justify the following statements:

1. For any fixed $\epsilon \ge \epsilon^*$ our class $\mathcal{L}_1^r(\Theta,\mathcal{H})$ will contain an optimal line as $\Theta \to \infty$ and $\mathcal{H} \to \infty$.

2. For any fixed $\epsilon \ge \epsilon^*$ the set of optimal lines in $\mathcal{L}_1^r(\Theta,\mathcal{H})$ increases to the set of optimal lines from $\mathcal{Q}_1$ as $\Theta \to \infty$ and $\mathcal{H} \to \infty$.

   Additionally, we will show

3. For $0 < p \le 1$ let $\mathcal{A}^{(\epsilon)}(p) = \{L(\theta_{m1}, h_{m1}) \in \mathcal{Q}_1 : P(E_{\epsilon,L(\theta_{m1},h_{m1})}) \ge p\}$, and let $\epsilon_p^* : \mathcal{A}^{(\epsilon_p^*)}(p) \ne \emptyset$ and $\mathcal{A}^{(\epsilon)}(p) = \emptyset\ \ \forall\ \ \epsilon < \epsilon_p^*$. Then for $p = 1.0$ and $N \to \infty$, $\mathcal{A}^{(\epsilon_p^*)}(p)$ converges to a unique optimal line $L^{(1.0)}(\theta_{j^*1}, h_{j^*1}) \in \mathcal{Q}_1$.

As the discretization of $\mathcal{L}_1$ becomes finer (i.e., as $\Theta \to \infty$ and $\mathcal{H} \to \infty$), Statement 1 says that our solution space will contain *an* optimal solution while Statement 2 indicates

that our solution space will contain *all* optimal solutions which intersect *rect*. Statement 3 says that as we require that a larger percentage of the data points fall within ϵ^* of the 'fitted' function, the set of optimal functions intersecting *rect* decreases to a single solution. This unique solution is the least squares function.

Before we begin, note that we have restricted ourselves to considering only those functions which intersect *rect*. This assumption is validated by the following theorem:

THEOREM 4.1 For any $\epsilon \geq \epsilon^*$ and for Θ and $\mathcal{H}$ sufficiently large, there exists an optimal line $L(\theta_{j^{**}1}, h_{j^{**}1}) \in \mathcal{L}_1$ such that

$$\{(X,Y) : Y = L(\theta_{j^{**}1}, h_{j^{**}1})(X),\ X \in [Z_{11}, Z_{21}]\} \bigcap rect \neq \emptyset.$$

Proof: A proof of Theorem 4.1 and a graphical representation of the proof are provided in the Appendix.

From Theorem 4.1 we know that if an optimal function exists, then there exists an optimal function which intersects *rect*. It follows without loss of generality that given $\epsilon \geq \epsilon^*$ we can restrict ourselves to Θ and $\mathcal{H}$ sufficiently large and only those optimal lines which intersect *rect*, i.e., those optimal lines which belong to $\mathcal{Q}_1$.

We now justify Statements #1, 2, and 3 with the help of the following propositions and theorems, the proofs of which can be found in the Appendix. These justifications will prove that the search space eventually includes an optimal solution with increasingly finer discretizations of the search space.

For simplicity, let $\rho = \ell_\theta + h\delta$ for given θ and h.

Statement #1

Statement #1, that for any fixed $\epsilon \geq \epsilon^*$ our class $\mathcal{L}_1^r(\Theta, \mathcal{H})$ will contain an optimal line as $\Theta \to \infty$ and $\mathcal{H} \to \infty$, will be justified if we can show that for Θ and $\mathcal{H}$ large, given any optimal line in $\mathcal{Q}_1$ we can find a line in $\mathcal{L}_1^r(\Theta, \mathcal{H})$ which is arbitrarily close to it. We begin with Proposition 4.1 which justifies that given an optimal line in $\mathcal{Q}_1$ we can find a Θ and $\mathcal{H}$ such that there exists a line in $\mathcal{L}_1^r(\Theta, \mathcal{H})$ which is arbitrarily close to the given optimal line.

PROPOSITION 4.1 Let $L(\theta_{m1}, h_{m1}) \in \mathcal{Q}_1$. Let $\xi > 0$. Then $\exists\ (\Theta_\xi, \mathcal{H}_\xi) :\ \forall\ \Theta > \Theta_\xi$ and $\mathcal{H} > \mathcal{H}_\xi,\ \exists\ L(\theta, h)$*:*

1. $L(\theta, h) \in \mathcal{L}_1^r(\Theta, \mathcal{H})$

2. $\mid \theta - \theta_{m1} \mid < \xi/2$ and $\mid \rho - \rho_{m1} \mid < \xi/2$.

Now, given $\xi > 0$, we would like there to exist a Θ_ξ and $\mathcal{H}_\xi$ such that given *any* optimal line in $\mathcal{Q}_1$ and *any* $\Theta > \Theta_\xi,\ \mathcal{H} > \mathcal{H}_\xi$ there exists a line in $\mathcal{L}_1^r(\Theta, \mathcal{H})$ which is arbitrarily close to the given optimal line. We know such a $(\Theta_\xi, \mathcal{H}_\xi)$ exists by the following theorem.

THEOREM 4.2 For each $\xi > 0,\ \exists\ (\Theta_\xi, \mathcal{H}_\xi) :$ for all $\Theta > \Theta_\xi$ and for all $\mathcal{H} > \mathcal{H}_\xi$, given any $L(\theta_{m1}, h_{m1}) \in \mathcal{Q}_1,\ \exists\ L(\theta, h) :$

1. $L(\theta, h) \in \mathcal{L}_1^r(\Theta, \mathcal{H})$

2. $\mid \theta - \theta_{m1} \mid < \xi/2$ and $\mid \rho - \rho_{m1} \mid < \xi/2$.

Thus given Θ and $\mathcal{H}$ sufficiently large we can get a line which is arbitrarily close to an optimal line. Hence Statement #1 is justified.

Statement #2

Recall that in Statement #2 we postulated that for any fixed $\epsilon \geq \epsilon^*$ the set of optimal lines in $\mathcal{L}_1^r(\Theta, \mathcal{H})$ increases to the set of optimal lines from $\mathcal{Q}_1$ (i.e., the set of all optimal lines which intersect *rect*) as $\Theta \to \infty$ and $\mathcal{H} \to \infty$. For the purpose of justifying this statement, let $\{\Theta_i,\ i = 1, 2, \dots\}$ and $\{\mathcal{H}_i,\ i = 1, 2, \dots\}$ represent the possible values of Θ and $\mathcal{H}$. For any $\epsilon > 0$ we define

$$\mathcal{A}_{\epsilon i}(p) = \{L(\theta_{j_i 1}, h_{j_i 1}) \in \mathcal{L}_1^r(\Theta_i, \mathcal{H}_i) : P(E_{\epsilon, L(\theta_{j_i 1}, h_{j_i 1})}) \geq p\}$$

and

$$\mathcal{A}_{\epsilon}(p) = \{L(\theta_{m1}, h_{m1}) \in \mathcal{Q}_1 : P(E_{\epsilon, L(\theta_{m1}, h_{m1})}) \geq p\}$$

and let $\mathcal{A}_{\epsilon i} = \mathcal{A}_{\epsilon i}(0.95)$ and $\mathcal{A}_{\epsilon} = \mathcal{A}_{\epsilon}(0.95)$. $\mathcal{A}_{\epsilon i}$ represents the set of all optimal lines which belong to $\mathcal{L}_1^r(\Theta_i, \mathcal{H}_i)$ while $\mathcal{A}_{\epsilon}$ represents the set of all optimal lines which belong to $\mathcal{Q}_1$. We would like $\mathcal{A}_{\epsilon i} \to \mathcal{A}_{\epsilon}$ as $i \to \infty$. However, we first need that if $\Theta \to \infty$ and $\mathcal{H} \to \infty$ then the limit of any sequence of functions in $\mathcal{Q}_1$ is contained in $\mathcal{Q}_1$ (and hence belongs to our search space). This is, in fact, true, as stated in Proposition 4.2.

PROPOSITION 4.2 For each $i = 1, 2, \dots,$ let $L(\theta_{n_i 1}, h_{n_i 1}) \in \mathcal{L}_1^r(\Theta_i, \mathcal{H}_i)$: $\theta_{n_i 1} \to \theta_{\lim}$ and $h_{n_i 1} \to h_{\lim}$ for some $\theta_{\lim},\ 0 < \theta_{\lim} \leq \pi,$ and $h_{\lim}$, $0 \leq h_{\lim} < \infty,$ as $i \to \infty$. Let

$$\mathcal{T}_1 = \{L(\theta_{\lim}, h_{\lim}) :\ \exists \text{ a sequence } \{L(\theta_{n_i 1}, h_{n_i 1})\}_{i=1}^{\infty},\ L(\theta_{n_i 1}, h_{n_i 1}) \in \mathcal{L}_1^r(\Theta_i, \mathcal{H}_i) \text{ such that } \theta_{n_i 1} \to \theta_{\lim} \text{ and } h_{n_i 1} \to h_{\lim}\}.$$

Then the optimal lines in $\mathcal{Q}_1$ are the optimal lines in $\mathcal{T}_1$.

Hence $\mathcal{Q}_1$ contains any optimal function which is a limit of a sequence of functions in $\mathcal{Q}_1$. We may now justify Statement #2.

Note that the fitness function $fit_{\epsilon,N} : (0, \pi] \times [-M, M] \rightarrow [0, \infty)$ is continuous where for any line in $\mathcal{T}_1$, $\theta \in (0, \pi]$ and the distance of the line from the origin is less than M (i.e., $d \leq M$). With $fit_{\epsilon,N}$ continuous and bounded, we state Theorem 4.3.

THEOREM 4.3 Let $\mathcal{A}_{\epsilon i}$, $i = 1, 2, \ldots,$ and $\mathcal{A}_\epsilon$ be as defined above. Then $\mathcal{A}_{\epsilon i} \rightarrow \mathcal{A}_\epsilon$ as $i \rightarrow \infty$.

Hence the optimal solutions which intersect *rect* are in the search space as $\Theta \rightarrow \infty$ and $\mathcal{H} \rightarrow \infty$.

Statement #3

The above theorem holds for any $\mathcal{A}_\epsilon(p)$, $0.95 \leq p \leq 1.0$, and for any $\epsilon \in \{\epsilon_p^*\}_{p=0.95}^{1}$, where $\epsilon_p^* : \mathcal{A}_{\epsilon_p^*}(p) \neq \emptyset$ and $\mathcal{A}_\epsilon(p) = \emptyset \; \forall \; \epsilon < \epsilon_p^*$ (e.g., if we require that 96% of the data points in a given dataset fall within ϵ of our optimal function, then $\epsilon_{0.96}^*$ is the smallest $\epsilon > 0$ for which such an optimal function exists). With this in mind, we conclude with a theorem which justifies Statement #3, i.e., for $p = 1.0$ and $N \rightarrow \infty$, $\mathcal{A}^{(\epsilon_p^*)}(p)$ converges to a unique optimal line $L^{(1.0)}(\theta_{j^*1}, h_{j^*1}) \in \mathcal{Q}_1$.

THEOREM 4.4 Let $\mathcal{A}_\epsilon(p)$ and ϵ_p^* be as defined, $0.95 \leq p \leq 1.0$. Then $\mathcal{A}_{\epsilon_{1.0}^*}(1.0) \equiv L^{(1.0)}(\theta_{j^*1}, h_{j^*1})$ as $N \rightarrow \infty$.

A graphical representation of this theorem is provided in the Appendix.

Suppose we require that all of the data points fall within ϵ of the 'fitted' function, where ϵ is the smallest positive value for which such a function exists.

Then for Θ and $\mathcal{H}$ large, as $N \to \infty$ our optimal function will converge to the least squares function. This implies that our algorithm can be used to fit non-robust as well as more robust optimal functions.

We have successfully demonstrated that

1. For any fixed $\epsilon \geq \epsilon^*$ our class $\mathcal{L}_1^r(\Theta, \mathcal{H})$ will contain an optimal line as $\Theta \to \infty$ and $\mathcal{H} \to \infty$.

2. For any fixed $\epsilon \geq \epsilon^*$ the set of optimal lines in $\mathcal{L}_1^r(\Theta, \mathcal{H})$ increases to the set of optimal lines from $\mathcal{Q}_1$ as $\Theta \to \infty$ and $\mathcal{H} \to \infty$.

Additionally, we will show

3. For $0 < p \leq 1$ let $\mathcal{A}^{(\epsilon)}(p) = \{L(\theta_{m1}, h_{m1}) \in \mathcal{Q}_1 : P(E_{\epsilon, L(\theta_{m1}, h_{m1})}) \geq p\}$, and let $\epsilon_p^* : \mathcal{A}^{(\epsilon_p^*)}(p) \neq \emptyset$ and $\mathcal{A}^{(\epsilon)}(p) = \emptyset \ \ \forall \ \ \epsilon < \epsilon_p^*$. Then for $p = 1.0$ and $N \to \infty$, $\mathcal{A}^{(\epsilon_p^*)}(p)$ converges to a unique optimal line $L^{(1.0)}(\theta_{j^*1}, h_{j^*1}) \in \mathcal{Q}_1$.

By justifying these statements we have shown that our search space contains an optimal solution. It remains to be shown that the algorithm converges to an optimal solution for $k_{\max} = 1$.

4.4.4.2 Convergence to Optimum

For any $\epsilon \geq \epsilon^*$ we know from Section 4.4.4.1 that we may choose Θ and $\mathcal{H}$ sufficently large so that an optimal solution is contained in the search space $\mathcal{L}_1^r(\Theta, \mathcal{H})$. We must now show that our algorithm converges to an optimal solution.

As defined previously, let

$$\mathcal{I}_{\epsilon,L(\theta_{j1},h_{j1})}(X,Y) = \begin{cases} 1 & \mid Y - L(\theta_{j1},h_{j1})(X) \mid < \epsilon \\ 0 & \text{otherwise.} \end{cases}$$

Since k takes only one value, we may drop the term $(1 - \frac{k}{k_{\max}})$ from Equation 4.3 and allow

$$fit_{\epsilon,N}(L(\theta_{j1},h_{j1})) = \Sigma_{i=1}^{N} \mathcal{I}_{\epsilon,L(\theta_{j1},h_{j1})}(X_i,Y_i).$$

Define

$$\overline{fit}_{\epsilon,N}(L(\theta_{j1},h_{j1})) = \frac{1}{N} fit_{\epsilon,N}(L(\theta_{j1},h_{j1})).$$

Note that

$$\lim_{N\to\infty} \overline{fit}_{\epsilon,N}(L(\theta_{j1},h_{j1})) = P(\bigcup_b \bigcup_{a<b} E_{\epsilon,L(\theta_{j1},h_{j1}),a,b}) = P(E_{\epsilon,L(\theta_{j1},h_{j1})}).$$

Given the convergence of the elitist GA, we know that our GA finds an optimal $L(\theta_{j1},h_{j1})$ for a given $\mathcal{L}^r(\Theta,\mathcal{H})$, ϵ, and N. So let $L(\theta_{j^*1},h_{j^*1})_{\epsilon,N}$ be such that

$$fit_{\epsilon,N}(L(\theta_{j^*1},h_{j^*1})_{\epsilon,N}) = \max_{L(\theta_{j1},h_{j1})\in\mathcal{L}_1^r(\Theta,\mathcal{H})} fit_{\epsilon,N}(L(\theta_{j1},h_{j1}))$$

where the dependence of an optimal line on ϵ and N has been made explicit. $L(\theta_{j^*1},h_{j^*1})_{\epsilon,N}$ also maximizes $\overline{fit}_{\epsilon,N}(L(\theta_{j1},h_{j1}))$. To determine whether our algorithm converges to an optimal solution, we examine $\lim_{N\to\infty} \overline{fit}_{\epsilon,N}(L(\theta_{j^*1},h_{j^*1})_{\epsilon,N})$ for $\epsilon = \epsilon_{N_2}$ where $\overline{fit}_{\epsilon_{N_2},N}(L(\theta_{j^*1},h_{j^*1})_{\epsilon_{N_2},N}) \geq 0.95$. If $\lim_{N\to\infty} \overline{fit}_{\epsilon,N}(L(\theta_{j^*1},h_{j^*1})_{\epsilon,N})$ is at least 0.95, then our algorithm does indeed converge to an optimal solution. Theorem 4.5 states that this is true.

THEOREM 4.5 Let N be large and ϵ_{N_2} be as defined above. Then for appropriate Θ and $\mathcal{H}$

$$\liminf_{N\to\infty} \overline{fit}_{\epsilon_{N_2},N}(L(\theta_{j^*1}, h_{j^*1})_{\epsilon_{N_2},N}) \geq 0.95.$$

The proof of Theorem 4.5 is presented in the Appendix.

Hence we have established that for $k_{\max} = 1$,

1. Our search space $\mathcal{L}^r(\Theta, \mathcal{H})$ contains an optimal solution as $\Theta \to \infty$ and $\mathcal{H} \to \infty$.

2. The algorithm converges to an optimal solution.

4.4.4.3 Remarks

1. For large Θ and $\mathcal{H}$, $\mid \theta_i - \theta_{i-1} \mid$ and $\mid h_i - h_{i-1} \mid$ are both small, so the optimal line $L(\theta_{j^*1}, h_{j^*1}) \in \mathcal{L}^r(\Theta, \mathcal{H})$ will be close to the optimal line $L(\theta_{m^*1}, h_{m^*1}) \in \mathcal{Q}_1$.

2. A search procedure based on the above mathematical formulation may be designed so that Θ, $\mathcal{H}$, and ϵ are adjusted *adaptively*, i.e., in a way that is dependent upon the algorithm's result. For example, we may start with initial choices Θ_{i_0}, $\mathcal{H}_{i_0}$, and ϵ_{i_0}, and run the algorithm for a finite number of iterations. If we reach several optimal results, we may reduce ϵ_{i_0}; if we reach a single result, we may increase Θ and $\mathcal{H}$. We then repeat this process until the resulting solution reflects the center of the probability density function of the observed random variables with an acceptable level of precision. Note that if Θ or $\mathcal{H}$ is small or ϵ is large, it is possible for the algorithm's result to be close

to an optimum in terms of probability but not in terms of Euclidean distance. Since appropriate values for Θ, $\mathcal{H}$, and ϵ are unknown a priori, we need to implement an adaptive procedure.

3. Recall that our requirement that 95% of the data points be within ϵ^* of the optimal solution is *heuristic*. Depending upon the particular dataset at hand and the desired accuracy of the solution, we may alter this *critical level* to better suit the given situation.

4. In Statement #3 and its corresponding proof (Theorem 4.4) we justified that with a critical level of 1.0 (100%) the algorithm converges to a unique optimum as $N \to \infty$. Note that this unique optimum corresponds to the least squares solution. Hence our method can be used to fit both robust (using an ϵ criterion) and non-robust (using a least squares criterion) solutions.

5. It is possible for our optimization problem to have more than one solution. For example, consider Figure 4.2 which shows a cross-section of the density of the observed random variables (X, Y). Suppose we have two parallel lines lying in the (X, Y)-plane and parallel to the Y-axis, one crossing the X-axis at the star marked with a number 1 (located at the center of the interval marked with bracket 1) and one crossing the X-axis at the star marked with a number 2 (located at the center of the interval marked with bracket 2). The percent values on the graph indicate the percentage of the data points with an X value lying within the corresponding delineated interval. Using these percentages we see that 95% of the data points fall within ϵ of line #1 and 95% of the data

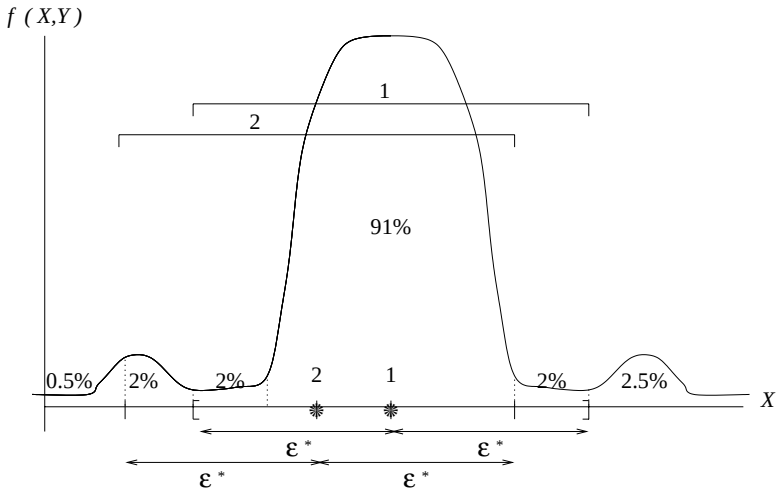


Figure 4.2 The existence of two optimal lines for a given data set

points fall within ϵ of line #2. Hence both of these lines would satisfy our optimization criteria (Equation 4.4).

6. We have assumed that the support of $\alpha_1(X,Y)$, B, is rectangular in shape. However, the support of $\alpha_1(X,Y)$ may have curved symmetric boundaries as opposed to straight lines. For example, let $\phi(X)$ be as defined in Equation 4.2, i.e.,

$$\phi(X) = \begin{cases} \frac{d_1 - X\cos\theta_1}{\sin\theta_1} & X \in [Z_{11}, Z_{21}] \\ 0 & \text{otherwise.} \end{cases}$$

Then we could have B as shown in Figure 4.3. All of the above results except Theorem 4.4 hold for such a support B as long as the symmetricity of $\alpha_1(X,Y)$ is maintained. The symmetricity is essential due to the nature of $fit_{\epsilon,N}$. For Theorem 4.4 to hold we also require that $\gamma_1 \neq \gamma_2$ (so that the center segment of ϕ has positive width).

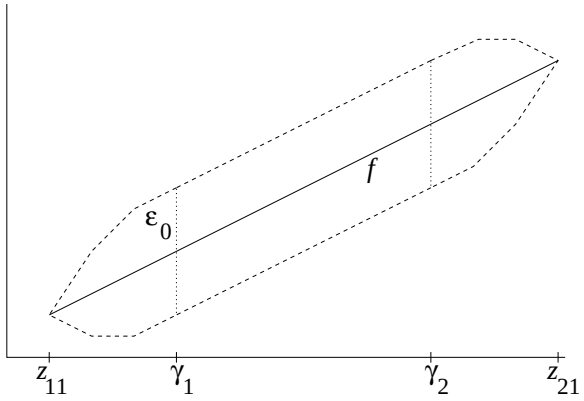


Figure 4.3 A dataset whose density has a support B with curved boundaries

4.4.5 Case $k_{\max} > 1$

Having completed the case $k_{\max} = 1$, we now consider the case of $k_{\max} > 1,\ k^*$ unknown, $k^* \in \mathcal{K}$. Recall that we need to show that

1. Our search space $\mathcal{L}^r(\Theta, \mathcal{H})$ contains an optimal solution as $\Theta \to \infty$ and $\mathcal{H} \to \infty$.

2. The algorithm converges to an optimal solution.

We extend the results of Section 4.4.4 to the case $k_{\max} > 1,\ k^*$ unknown, $k^* \in \mathcal{K}$, by utilizing the case of $k_{\max} > 1,\ k^*$ known, $k^* \in \mathcal{K}$.

4.4.5.1 Optimal Solution in Search Space

Suppose $k = k^*,\ k^* > 1,\ k^* \in \mathcal{K}$ where k^* is known. Our optimal string should represent a k^*-piecewise linear function $L(\boldsymbol{\theta}_{j^*.}^{k^*}, \boldsymbol{h}_{j^*.}^{k^*})$ such that

$$fit_{\epsilon^*,N}(L(\boldsymbol{\theta}_{j^*.}^{k^*}, \boldsymbol{h}_{j^*.}^{k^*})) = \max_{L(\boldsymbol{\theta}_{j.}^{k^*}, \boldsymbol{h}_{j.}^{k^*}) \in \mathcal{L}_{k^*}^r(\Theta,\mathcal{H})} fit_{\epsilon^*,N}(L(\boldsymbol{\theta}_{j.}^{k^*}, \boldsymbol{h}_{j.}^{k^*}))$$

where each $L(\theta_{ji}, h_{ji}),\ i = 1, \ldots, k^*$, satisfies the properties listed in Section 4.2 for L_{ji}, and

$$\mathcal{L}^r_{k^*}(\Theta, \mathcal{H}) = \{L^{k^*}_{j\cdot} \in \mathcal{L}_{k^*} : L_{ji} \text{ is of the form } X\cos\theta_{ji} + Y\sin\theta_{ji} = \ell_{\theta_{ji}} + h_{ji}\delta$$
$$\forall\ \ i = 1, \ldots, k^*,\ \theta_{ji} \in \{\tfrac{\pi}{2^l a}, \tfrac{2\pi}{2^l a}, \ldots, \tfrac{(2^l a - 1)\pi}{2^l a}, \pi\},$$
$$h_{ji} \in \{0, 1, \ldots, 2^l d - 1\},\ \text{and } \delta = diag/(2^l d - 1)\}.$$

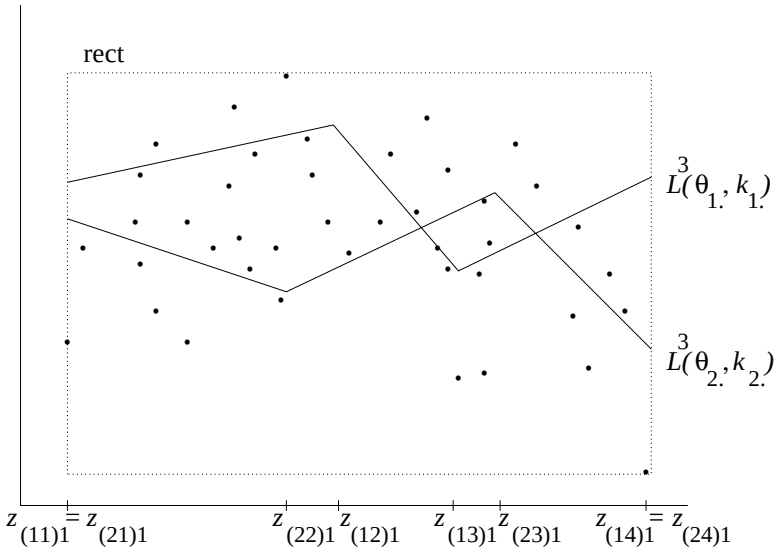


Figure 4.4 Example functions from $\mathcal{L}^r_3$

Two examples $L^3_{1\cdot}$ and $L^3_{2\cdot}$ of functions belonging to $\mathcal{L}^r_3(\Theta, \mathcal{H})$ for some choice of Θ and $\mathcal{H}$ are shown in Figure 4.4.

With respect to optimization we can approach each L_{ji} as we did each $L(\theta_{j1}, h_{j1})$ in the $k_{\max} = 1$ case, treating $\alpha_i(X, Y)$ and $[Z_{1i}, Z_{1(i+1)}]$ as we did $\alpha_1(X, Y)$ and

$[Z_{11}, Z_{21}]$, respectively. We then recognize $\mathcal{L}^r_{k^*}(\Theta, \mathcal{H})$ as simply $\bigcup_j \bigcup_{i=1}^{k^*} \mathcal{L}^r_{k^* ji}(\Theta, \mathcal{H})$ where

- $\mathcal{L}^r_{k^* ji}(\Theta, \mathcal{H})$ is $\mathcal{L}^r_1(\Theta, \mathcal{H})$ restricted to $[Z_{(ji)1}, Z_{(j(i+1))1}]$
  and
- the union is taken over all possible pieces and over all possible knot locations (any piece/knot combinations which contain pieces that do not intersect are removed).

By the results of Section 4.4.4.1, for any ji we can get arbitrarily close to any line over $[Z_{(ji)1}, Z_{(j(i+1))1}]$ intersecting *rect*, hence we can get arbitrarily close to any piecewise function with knot locations satisfying $[Z_{(j1)1}, \ldots, Z_{(j(k^*+1))1}]$ whose pieces intersect *rect*. By taking the union over all j and over all possible knot locations satisfying $[Z_{(j1)1}, \ldots, Z_{(j(k^*+1))1}]$, we see that an optimal k^*-piecewise solution is contained in $\mathcal{L}^r_{k^*}(\Theta, \mathcal{H})$ as $\Theta \to \infty$ and $\mathcal{H} \to \infty$.

By defining our search space as $\mathcal{L}^r(\Theta, \mathcal{H}) = \bigcup_{k \in \mathcal{K}} \mathcal{L}^r_k(\Theta, \mathcal{H})$ (see Section 4.4.2) we can guarantee, using the same reasoning as in Section 4.4.4.1, that an optimal k^*-piecewise solution, k^* unknown, $k^* \in \mathcal{K}$, will be contained in the search space as $\Theta \to \infty$ and $\mathcal{H} \to \infty$.

Having shown that an optimal solution is contained in the search space, we now show that the algorithm converges to an optimal solution. To show this, we first consider the case of k^* known.

4.4.5.2 Convergence to Optimum

If k^* is known, $k^* \in \mathcal{K}$, then for appropriate Θ and $\mathcal{H}$ we are searching for a function $L(\boldsymbol{\theta}^{k^*}_{j^*.}, \boldsymbol{h}^{k^*}_{j^*.}) \in \ \mathcal{L}^r_{k^*}(\Theta, \mathcal{H}) \bigcap \mathcal{L}^{(\epsilon^*)}$ which maximizes

$$fit_{\epsilon,N}(L(\boldsymbol{\theta}^{k^*}_{j.}, \boldsymbol{h}^{k^*}_{j.})) = \sum_{i=1}^{N} \mathcal{I}_{\epsilon, L(\boldsymbol{\theta}^{k^*}_{j.}, \boldsymbol{h}^{k^*}_{j.})}(X_i, Y_i) + (1 - \frac{k^*}{k_{\max}}) \tag{4.5}$$

over all $L(\boldsymbol{\theta}^{k^*}_{j.}, \boldsymbol{h}^{k^*}_{j.}) \in \mathcal{L}^r_{k^*}(\Theta, \mathcal{H})$. As was done in Section 4.4.4.2, we may drop the term $(1 - \frac{k^*}{k_{\max}})$. The convergence of the elitist GA combined with the continuity of $fit_{\epsilon,N}$ ensures that Theorem 4.5 holds, i.e.,

$$\liminf_{N\to\infty} \overline{fit}_{\epsilon_{N_2},N}(L(\boldsymbol{\theta}^{k^*}_{j^*.}, \boldsymbol{h}^{k^*}_{j^*.})_{\epsilon_{N_2},N}) \geq 0.95.$$

Since the above holds for any fixed $k = k^*$, an algorithm to find an optimal solution in the $k_{\max} > 1,\ k^* \in \mathcal{K}$ unknown case could be designed as follows:

1. Divide $\mathcal{L}^r(\Theta, \mathcal{H})$ into its component classes $\mathcal{L}^r_1(\Theta, \mathcal{H}),\ \mathcal{L}^r_2(\Theta, \mathcal{H}), \ldots, \mathcal{L}^r_{k_{\max}}(\Theta, \mathcal{H})$.

2. On each class $\mathcal{L}^r_k(\Theta, \mathcal{H}),\ k = 1, \ldots, k_{\max}$, use an elitist GA to find $L(\boldsymbol{\theta}^k_{j^*.}, \boldsymbol{h}^k_{j^*.})$ where $fit_{\epsilon,N}(L(\boldsymbol{\theta}^k_{j^*.}, \boldsymbol{h}^k_{j^*.})) = \max_{L(\boldsymbol{\theta}^k_{j.}, \boldsymbol{h}^k_{j.}) \in \mathcal{L}^r_k(\Theta,\mathcal{H})} fit_{\epsilon,N}(L(\boldsymbol{\theta}^k_{j.}, \boldsymbol{h}^k_{j.}))$.

3. Define $L_{\text{best}} = \{L(\boldsymbol{\theta}^1_{j^*.}, \boldsymbol{h}^1_{j^*.}), \ldots, L(\boldsymbol{\theta}^{k_{\max}}_{j^*.}, \boldsymbol{h}^{k_{\max}}_{j^*.})\}$. Then the solution is taken as the function $L(\boldsymbol{\theta}^{k^*}_{j^*.}, \boldsymbol{h}^{k^*}_{j^*.}) \in L_{\text{best}}$ which satisfies

$$fit_{\epsilon,N}(L(\boldsymbol{\theta}^{k^*}_{j^*.}, \boldsymbol{h}^{k^*}_{j^*.})) = \max_{L(\boldsymbol{\theta}^k_{j^*.}, \boldsymbol{h}^k_{j^*.}) \in L_{\text{best}}} fit_{\epsilon,N}(L(\boldsymbol{\theta}^k_{j^*.}, \boldsymbol{h}^k_{j^*.})).$$

This type of GA we call a *partitioned genetic algorithm*. We have noted previously that the elitist GA will converge to an optimal solution as $N \rightarrow \infty$. This proof is based on two assumptions:

1. The optimal string from the present population has a fitness value no less than the fitness values of the optimal strings from the previous populations.

2. Each string has a positive probability of going to an optimal string within any given iteration.

As these assumptions hold for the partitioned GA, it is easy to prove (although the proof will not be given here) that the proof of convergence to an optimal string holds for this algorithm as well. Hence we have shown that our algorithm converges to an optimal solution.

We have successfully proven that the GA algorithm which we have described will find an optimal k^*-piecewise linear function $L(\boldsymbol{\theta}_{j^*.}^{k^*}, \boldsymbol{h}_{j^*.}^{k^*}) \in \mathcal{L}^r(\Theta, \mathcal{H}) \bigcap \mathcal{L}^{(\epsilon^*)}$ such that

$$fit_{\epsilon^*,N}(L(\boldsymbol{\theta}_{j^*.}^{k^*}, \boldsymbol{h}_{j^*.}^{k^*})) = \max_{L(\boldsymbol{\theta}_{j.}^{k}, \boldsymbol{h}_{j.}^{k}) \in \mathcal{L}^r(\Theta,\mathcal{H})} fit_{\epsilon^*,N}(L(\boldsymbol{\theta}_{j.}^{k}, \boldsymbol{h}_{j.}^{k})) \tag{4.6}$$

where $k^* = \min\{k \in \mathcal{K} : \mathcal{L}_k^{(\epsilon^*)} \neq \emptyset\}$.

4.4.5.3 Partitioned GA vs. VLGA

Due to efficiency considerations, we plan to implement a partitioned GA instead of a variable length GA (VLGA). This choice is justified by the following argument.

Let $fit^*_{\epsilon,N}$ be the fitness value of the partitioned GA solution and let $fit^{**}_{\epsilon,N}$ be the fitness value of the VLGA solution. Note that $fit^{**}_{\epsilon,N} = fit_{\epsilon,N}(L(\boldsymbol{\theta}^{k^*}_{j^{**}.}, \boldsymbol{h}^{k^*}_{j^{**}.}))$ for some $L(\boldsymbol{\theta}^{k^*}_{j^{**}.}, \boldsymbol{h}^{k^*}_{j^{**}.}) \in \mathcal{L}^r_{k^*}(\Theta, \mathcal{H}),\ k^* \in \mathcal{K}$. We will justify that $fit^*_{\epsilon,N} = fit^{**}_{\epsilon,N}$ and use the fact that the VLGA converges to an optimal solution to conclude that the partitioned GA converges to an optimal solution.

We start with the following.

THEOREM 4.6 The partitioned GA solution and the VLGA solution have the same fitness value, i.e., $fit^*_{\epsilon,N} = fit^{**}_{\epsilon,N}$.

Proof: See Appendix.

However, as shown in [23], for fixed $\Theta,\ \mathcal{H}$, and ϵ an optimal string from the VLGA will represent a function $L(\boldsymbol{\theta}^{k^*}_{j^{**}.}, \boldsymbol{h}^{k^*}_{j^{**}.})$ such that

1. $\sum_{i=1}^N \mathcal{I}_{\epsilon, L(\boldsymbol{\theta}^{k^*}_{j^{**}.}, \boldsymbol{h}^{k^*}_{j^{**}.})}(X_i, Y_i) = \nu_{\max}$ where
$$\nu_{\max} = \max_{L(\boldsymbol{\theta}^k_{j.}, \boldsymbol{h}^k_{j.}) \in \mathcal{L}^r(\Theta,\mathcal{H})} \sum_{i=1}^N \mathcal{I}_{\epsilon, L(\boldsymbol{\theta}^k_{j.}, \boldsymbol{h}^k_{j.})}(X_i, Y_i).$$
2. $k^* = \min\{k \in \mathcal{K} : \exists\ L(\boldsymbol{\theta}^k_{j.}, \boldsymbol{h}^k_{j.}) \in \mathcal{L}^r(\Theta, \mathcal{H})$ satisfying
$$\sum_{i=1}^N \mathcal{I}_{\epsilon, L(\boldsymbol{\theta}^k_{j.}, \boldsymbol{h}^k_{j.})}(X_i, Y_i) = \nu_{\max}\}.$$

By yielding a solution with the largest fitness value and the least number of pieces, the VLGA solution maximizes our fitness function as given in Equation 4.3. Since the partitioned GA solution has the same fitness value as the VLGA solution, we conclude that the partitioned GA converges to an optimal solution.

4.4.5.4 Remarks

1. We have justified the claim that for any value of $k_{\max}$, an optimal solution is contained in the search space of our algorithm and that our algorithm will

converge to an optimal solution. We have also established that for Θ and $\mathcal{H}$ large and *critical level* = 1.0, the solution of our algorithm will converge to the least squares solution as $N \to \infty$. Hence our algorithm can be used to fit both robust and non-robust solutions.

2. As in the $k_{\max} = 1$ case, for $k_{\max} > 1$ we do not know an appropriate value for ϵ a priori. Hence to implement our algorithm we must use an adaptive procedure which will search for an appropriate value for ϵ.

The theoretical foundation of our algorithm has been established. In the next section we show the results of implementing our algorithm on several datasets and discuss how these results compare to those of similar methods.

4.5 Experimental Results

4.5.1 Methods and Implementation

4.5.1.1 Genetic Algorithm

Due to lack of computational resources the variable length genetic algorithm described above was not implemented. Instead, we applied to each dataset a partitioned GA with either $\mathcal{K} = \{2, 3, 4\}$, $\mathcal{K} = \{3, 4, 5\}$, or $\mathcal{K} = \{5, 6, 7\}$. We utilized binary coding although an alternate coding scheme could have been used. Since we chose values for Θ and $\mathcal{H}$ which were fixed but large (e.g., $\mathrm{l}_a = 8$ and $\mathrm{l}_d = 12$), the partitioned GA adaptively searched for only ϵ. We refer to such a GA as a *variable epsilon* genetic algorithm. For each value $k \in \mathcal{K}$, the variable epsilon GA can be described as follows:

1. Set the global parameters for population size M (≈ 50), crossover probability p_c ($p_c = 0.8$), number of characters for representing angle l_a ($\mathrm{l}_a \approx 8$), number of characters for representing distance or offset value l_d ($\mathrm{l}_d \approx 12$), and the maximum number of iterations $MaxNit$ ($MaxNit \approx 20000$). Our selection for M depends on the computing power of our machine while l_a and l_d depend upon the desired precision of our result. The mutation probability p_m was varied as a function of the number of iterations completed - see Section 4.3.1 and [14] for more details.

2. Choose $crit$ = critical level = percentage of data points to fall within ϵ of the final fitted piecewise linear function, and a large initial value for ϵ. The fitness value of each string (which represents a function $L(\boldsymbol{\theta}^k_{j\cdot}, \boldsymbol{h}^k_{j\cdot}) \in \mathcal{L}^r(\Theta, \mathcal{H})$) is determined by Equation 4.3.

3. For each $k \in \mathcal{K}$ run an elitist GA until either (1) $crit \leq$ (the maximum fitness value of the population)/(number of observations), or (2) the maximum number of iterations, $MaxNit$, is reached. If (1) occurs, then set $\epsilon = \epsilon - \tau$ ($\tau \approx 0.01 * \epsilon$), Nit = iteration number = 1, and restart the elitist GA using the current population as the initial population. If (2) occurs, report the function corresponding to the string with the maximum fitness value as the final result for the given value of k.

4. Compare the results across k values and select the string corresponding to the overall maximum fitness value as the optimum string.

Note that within each run of our algorithm the value of k is *fixed*. We then compare the results of each run of the algorithm for each value of $k \in \mathcal{K}$ to determine the overall optimal string.

4.5.1.2 Data

The proposed algorithm was applied to seven datasets, of which five were simulated. These datasets are described in Table 4.1.

Table 4.1 Experimental Datasets

Set	N	k^*	Generating Function	Noise
1	375	3	$g(x) = \begin{cases} -2.0x & -1.0 \le x < 0.0 \\ 0.5x & 0.0 \le x < 1.0 \\ -0.5x + 1.0 & 1.0 \le x \le 2.0 \end{cases}$	$Nor(0, 0.25)$
2	320	3	$g(x) = \begin{cases} 3.88x + 10.44 & -3.0 \le x < -1.29 \\ -1.74x + 3.14 & -1.29 \le x < 3.7 \\ 3.77x - 17.25 & 3.7 \le x \le 4.9 \end{cases}$	$Nor(0, 1)$
3	60	3	$g(x) = \begin{cases} -0.2x + 0.8 & 0.0 \le x < 1.0 \\ -1.4x + 2.0 & 1.0 \le x < 2.0 \\ 0.8x - 2.4 & 2.0 \le x \le 3.0 \end{cases}$	$Nor(0, 0.1)$
4	50	4	$g(x) = \begin{cases} 1.3x + 2 & 0.0 \le x < 0.9 \\ 0.05x + 3.125 & 0.9 \le x < 2.1 \\ -0.4x + 3.2 & 2.1 \le x < 4.3 \\ 3.2x - 11.41 & 4.3 \le x \le 5.0 \end{cases}$	$Nor(0, 0.2)$
5	128	–	$g(x) = 10(4x - 2)/(1 + (100 * ((4x - 2)^2)))$ $\quad 0 \le x < 1$	U(-0.1, 0.1)
6	49	–	titanium hat data $\quad 595 \le x \le 1075$	———
7	142	–	pezzack et. al. data $\quad 0.0 \le x \le 2.834$	———

Datasets 1, 2, 3, and 4 were generated from piecewise linear functions to have specific characteristics - Dataset 1: no outliers, equally spaced knots; Dataset 2: no outliers, unequally spaced knots; Dataset 3: outliers, equally spaced knots; and Dataset 4: outliers, unequally spaced knots. The noise was normally distributed (denoted Nor(mean,sd)) and the sample sizes range from 50 to 375 data points.

Dataset 5 was created using a generating function borrowed from Schwetlick and Schütze [130] with noise following a uniform distribution on $[-0.1, 0.1]$ (denoted U(-0.1,0.1)), as stated above.

The two non-simulated datasets are standard datasets from the literature - the titanium heat data of de Boor [45] and Pezzack, Norman, and Winter's angular displacement data [117]. These were used to demonstrate the effectiveness of the proposed method in real applications.

4.5.1.3 Comparisons

For Datasets 1, 2, 3, and 4, the results of the variable epsilon GA were compared to the results of two other piecewise linear fitting methods:

1. *Sb-spline*: a b-spline of degree 2 is fit using the functions **bs** and **lm** in the software package S-plus, version 3.4, release 1 [82]. The knot locations are equally spaced by default.

2. *least-squares GA*: a genetic algorithm which has the same global parameters as the variable epsilon GA fits a piecewise linear function by minimizing the sum of squared errors (SSE).

All methods were applied to each dataset for each value $k \in \mathcal{K}$. Then, for each method and each dataset, the results for all k values were compared and the best result was chosen as the solution proposed by the given method for the given dataset.

Note that datasets 5, 6, and 7 have abscissae values in strictly increasing order. This allows comparison with more sophisticated spline fitting algorithms. Hence for these examples we will use the following comparative methods:

1. *de Boor/Rice*: a truncated power basis spline of degree 2 is fit using the routines BSVLS, BSCPP, and BSLSQ in IMSL version 10 [83]. Given an initial set of knots, the knot locations are determined automatically by the nonlinear optimization algorithm of de Boor and Rice [47]. The number of knots is fixed by the user.

2. *Dierckx*: a b-spline of degree 2 is fit using the nonlinear optimization algorithm CURFIT of Dierckx [52]. The number of knots and their locations are chosen automatically by the algorithm; a single parameter must be specified to balance smoothness and lack of fit.

We have not attempted to compare results under neural network based methods since we are fitting straight lines and not curves. A comparison of our results with those of a least squares GA reflects the work of Karr [86, 87]; a comparison with Vankeerberghen's least median squares GA [140] may be included in future research.

All experiments were run on a Sun Sparcstation 5.

4.5.2 Selected Results

The following results clearly demonstrate that a GA based method can yield results comparable to those of existing methods, without the advantage of good starting values (unlike nonlinear optimization methods). It is also easily seen that the model selection criterion does an excellent job of allowing the GA to search for appropriate models in the presence of outliers while also permitting model fitting when outliers are not present. However, it is also shown that traditional methods can do just as well at fitting models to data without outliers as the GA method. In order for the power of the GA method to be demonstrated, one needs larger, more complicated parameter spaces. Empirical results of the GA method on problems of this nature are presented in Chapter 5.

4.5.2.1 Dataset 1

This was our 'nice' dataset - no outliers and a generating function with equally spaced knots. Since there were no outliers, we used the sum of squared errors (SSE) as a method of comparison. Our choice of critical value for the variable epsilon GA was 95%. All four methods chose the correct number of lines; the least squares GA yielded the best fit, followed by the Sb-spline and the variable epsilon GA. The run times for the Sb-spline was almost instantaneous while the GAs took longer (approx. 5 minutes).

We observed that although the variable epsilon GA did not yield the best fit, it was the only method which yielded a very nice fit (SSE = 22.80) when the number of pieces was misspecified - k was set equal to 4 although the generating function has 3

Table 4.2 Results of Dataset 1, $k = 3$

Method	SSE	Method	SSE	Method	SSE
Sb-spline	22.26	lst.sq. GA	17.89	var.eps. GA	23.73

pieces (k=3)(see Figure 4.5). The algorithm was almost able to merge 2 of the pieces into 1. Hence the variable epsilon GA was more robust against misspecification of k in relation to the other methods.

4.5.2.2 Dataset 2

This dataset had a generating function with unequally spaced knots - the middle piece was considerably longer than either of the end pieces. For this reason the fit of the Sb-spline was quite poor in comparison to the genetic algorithms' results. The best model from each method had the correct number of lines. The variable epsilon GA (with $crit$=95%) and the least squares GA yielded similar results, followed by the Sb-spline.

In some of our earlier experiments, the variable epsilon GA performed better than the least squares GA. However, further experiments revealed that the least squares GA does yield a comparable model given an equivalent number of iterations. The speed of the Sb-spline algorithm was almost instantaneous, while the GA algorithms took about 5 minutes of CPU time to yield a near optimal result.

With this dataset, all methods did merge lines for a better fit when the choice of k was too large. The Sb-spline could not match the quality of fit of the GAs because its knot locations were fixed.

Table 4.3 Results of Dataset 2, $k = 3$

Method	SSE	Method	SSE	Method	SSE
Sb-spline	610.24	lst. sq. GA	345.47	var. eps. GA	351.57

4.5.2.3 Dataset 3

The other strengths of the variable epsilon GA became evident when we considered datasets with outliers. Dataset 3 had equally spaced knots and contained a cluster of 6 outliers. The outliers made the SSE criterion useless as a measure of fit, so we based our statements about the quality of the results on visual comparisons. Figure 4.7 clearly shows that the outliers caused difficulties for all methods except the variable epsilon GA. The Sb-spline and the least squares GA showed particularly poor results. However, the parameter *crit* of the variable epsilon GA made it robust against outliers. *crit* was set at 90% to allow for the outliers - leading, in this case, to a superior fit.

When *crit* was set at 95%, as was done previously, the variable epsilon GA result was closer to that of the least squares GA (see Figure 4.8). Even if the outliers had not been clustered, the variable epsilon GA would have reached a superior solution - see Dataset 4 for an example with non-clustered outliers.

As with dataset 1, only the variable epsilon GA yielded a very nice fit when the number of lines was misspecified as $k = 4$ instead of $k = 3$. Despite solution spaces with sizes of order $2^{60} = 1152921 * 10^{12}$ for k=3, $2^{80} = 1208925 * 10^{18}$ for $k = 4$, and $2^{100} = 1267650 * 10^{24}$ for k=5, the GAs were able to yield near-optimal results in about 5 minutes.

4.5.2.4 Dataset 4

Dataset 4 combined unequally spaced knots with the presence of 4 outliers which were not clustered. We set *crit*=92% to accommodate for these outliers in our modeling procedure. From Figure 4.9 it is clear that only the variable epsilon GA provided a reasonable fit. The least squares GA was adversely affected by the outliers, while the Sb-spline failed to capture the shape of the dataset. The robustness of the variable epsilon GA proved essential for a proper fit. It should be noted, however, that the Sb-spline and the least squares GA would in all likelihood fit better models in the presence of less *influential outliers*; see Section 5.4.

We also found that when the number of lines was too large ($k = 4$ instead of $k = 3$) the variable epsilon GA was able to compensate for this by selecting pieces which almost merged.

4.5.2.5 Dataset 5

Dataset 5 has only one observation at each ordinate value and no outliers. Although our algorithm was designed with the robust case in mind, we feel that it is important for it to perform well in the non-robust case. We set *crit*=99% and $\mathcal{K} = \{5, 6, 7\}$.

Table 4.4 Results of Dataset 5

Method	SSE	Method	SSE	Method	SSE
de Boor/Rice	0.0034	Dierckx	0.5886	var. eps. GA	0.0041

It appears that the de Boor/Rice and variable epsilon GA algorithms, whose best models had $k = 5$ pieces, provided reasonable models while the Dierckx model was less adequate. We observed that with this and the following datasets, the results of the de Boor/Rice algorithm were highly dependent upon the starting values; this is alluded to in [52]. However, given reasonable starting values, the algorithm yielded the models with the lowest error.

We found the Dierckx algorithm to be difficult to implement. It was found that for an adequate fit, weights had to be supplied for the data points; balancing the weight values with the smoothing factor was non-trivial. It was also noted that the number of knots in the Dierckx model was considerably higher than the number of knots in the de Boor/Rice and variable epsilon models (in this case we could not obtain a reasonable model for $k \in \mathcal{K}$; we required $k = 12$). Further experimentation is necessary to determine whether a more experienced user could find a Dierckx model of size $k \in \mathcal{K}$ with an acceptable quality of fit.

4.5.2.6 Dataset 6

The titanium hat dataset is one of the standard test datasets in the spline fitting literature. With $crit$=99% and $\mathcal{K} = \{5, 6, 7\}$, both the de Boor/Rice algorithm and variable epsilon GA selected $k = 5$; the Dierckx method, in contrast, chose $k = 29$. In looking at Figure 8 and Table 4, we note that the de Boor/Rice spline is the superior model, followed by the variable epsilon GA spline and the Dierckx spline. Although the de Boor/Rice spline provides the best fit to the data, we again

note that the result of this algorithm was highly dependent upon the choice of initial knots. The result of the variable epsilon GA does not demonstrate this dependence.

Table 4.5 Results of Dataset 6

Method	SSE	Method	SSE	Method	SSE
de Boor/Rice	0.004	Dierckx	0.3295	var. eps. GA	0.044

4.5.2.7 Dataset 7

Our last experimental dataset is the angular displacement data of Pezzack et. al. [117], another popular dataset for testing curve fitting algorithms. Once again, the de Boor/Rice spline is the superior model followed closely by the variable epsilon GA spline. Both models selected $k = 6$ from the candidate set $\mathcal{K} = \{5, 6, 7\}$. The Dierckx model is relatively inadequate due to several spurious oscillations (this is reflected in the choice of $k = 74$). We attempted to remove these oscillations by increasing the smoothing factor but this failed to improve the fit of the model. It is possible, however, that a more experienced user would have greater success in finding the correct balance between smoothing factor, data point weights, and lack of fit.

Table 4.6 Results of Dataset 7

Method	SSE	Method	SSE	Method	SSE
de Boor/Rice	0.0147	Dierckx	1.737	var. eps. GA	0.0990

4.5.3 Conclusions

Our results demonstrate that for 'nice' datasets (no outliers) the variable epsilon GA can provide a fit comparable to that of a least squares GA, de Boor/Rice, or Dierckx spline model. Although the variable epsilon GA did not always yield the 'best' result for nice datasets, it always achieved a satisfactory fit very quickly without the benefit of a user-defined set of initial knot locations. When the number of pieces was too large, the algorithm did a nice job of decreasing the effective number of pieces by choosing lines which almost merged. By increasing the number of iterations and/or the string length, it is likely that even better results can be achieved.

For datasets with outliers, the variable epsilon GA appears capable of yielding results far superior to those of comparable methods. As with 'nice' datasets, if too many pieces have been specified the algorithm can almost merge pieces to yield a model with the appropriate number of effective knots. The variable *crit* makes it possible to adjust for outliers while the variables Θ and $\mathcal{H}$ provide some control over the precision of the final fit. By allowing ϵ to be determined adaptively, our algorithm can find a result which (1) satisfies the critical value and (2) is closest to the majority of the data points with respect to other solutions.

We conclude that the variable epsilon GA represents a valuable tool for fitting both robust and non-robust piecewise linear functions.

4.5.4 Remarks

1. It was noted in Section 5.2 that the results of methods based on least squares, such as the algorithms of de Boor/Rice and Dierckx, are adversely affected by the presence of outliers. We should note that this effect will be present most often when the outliers are *influential*. Hence it is possible to use a least squares method in the presence of such outliers if, e.g., one fits an initial model via least squares, uses influence diagnostics [146] to test for influential observations, removes such observations from the analysis, and then refits the model. This assumes, however, that such observations can be removed. It is often the case, especially in consulting situations, that the client refuses to remove any observations from the analysis. In such a situation we would recommend that the analysis be performed using a method similar to the one presented here.

2. Since the solution given by the variable epsilon GA is the result of a random process, we decided to run the variable epsilon GA described above several times on the following dataset and examine the variability of the results. Dataset 8 contained several outliers, had equally spaced knots and met the specifications in Table 4.7.

   A variable epsilon GA was executed 8 times with the global parameters set as in Section 4.5.1.1 and $crit = 95\%$. The results are shown in Table 4.8 and Figure 4.12.

Table 4.7 Results of Dataset 8

Set #	N	k	Function	Noise
5	60	3	$g(x) = \begin{cases} 2.5x & 0.0 \leq x < 1.0 \\ -4.0x + 6.5 & 1.0 \leq x < 2.0 \\ 3.5x + -8.5 & 2.0 \leq x \leq 3.0 \end{cases}$	$Nor(0, 0.1)$

Figure 4.12 does not appear to contain 8 functions because several results were identical. By examining the ranges of both the slopes and intercepts of the linear pieces, the achieved level of consistency appears to be satisfactory. Each parameter of the generating lines falls within its corresponding approximated range given in Table 4.8. For example, the slope and intercept of the first generating line are 2.5 and 0, respectively, and our experiments generated corresponding ranges of (2.41421, 3.02704) and (-0.188247, 0.180099). Further research is needed to determine a more precise measure of result variability.

Table 4.8 Dataset 8 Results; Ranges for Intercepts and Slopes

	Piece #1	Piece #2	Piece #3
Intercept	(-0.188247, 0.180099)	(6.64789, 6.91933)	(-8.73460, -7.92068)
Slope	(2.41421, 3.02704)	(-4.21080, -3.99222)	(3.29656, 3.61354)

3. For our experiments we considered datasets where k was between 3 and 7. However, we have run experiments with larger values of k (≈ 10) and found that near optimal models can be found in a reasonable amount of time (≈ 15 min.). With the increasing availability of computational resources, it seems

plausible that larger solution spaces (i.e., larger k and N) could be handled in approximately the same amount of CPU time.

4.6 Conclusions

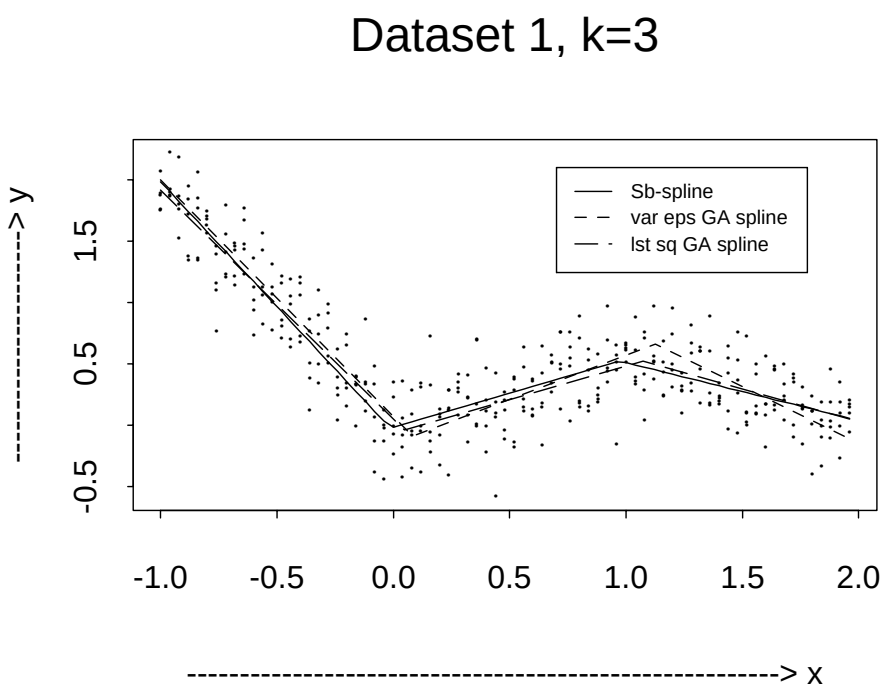
By using genetic algorithms we have devised a method for fitting piecewise linear functions to data in $\mathcal{R}^2$ which not only optimizes the number of pieces but also optimizes the knot locations. With the assumption that the probability density function of our random variables is symmetric, the above theory shows that our method will lead to a piecewise linear function which 'fits' the given dataset. However, even if we do not make this assumption (so our only assumption about the data is that the underlying probability density function is continuous), our method will still yield an optimal result for the chosen fitness function.

Our method yielded very good results in the presence of noise. The critical value makes it possible for our algorithm to reach a near-optimal result even in the presence of outliers. The convergence of genetic algorithms has been shown for practically any choice of parameter values (e.g., $M = 50$, $p_c = 0.8$, etc.) although the best choices are still a matter of contention. The formulation of an optimal stopping rule is a subject of ongoing research, although it is known that increasing the number of iterations leads to a result with better accuracy.

Note, however, that for datasets with no outliers traditional, nonlinear optimization techniques provided models competitive with those fit by the GA based method. The empirical results involve datasets with parameter spaces that are not

large nor complicated enough to demonstrate the advantage of a GA based method over traditional techniques.

In Chapter 5 we extend the use of genetic algorithms to fitting splines of higher orders and provide empirical results involving large, complex parameter spaces which leave no doubt concerning the advantage of numerical optimization techniques, such as genetic algorithms, over traditional methods in modeling with adaptive splines.



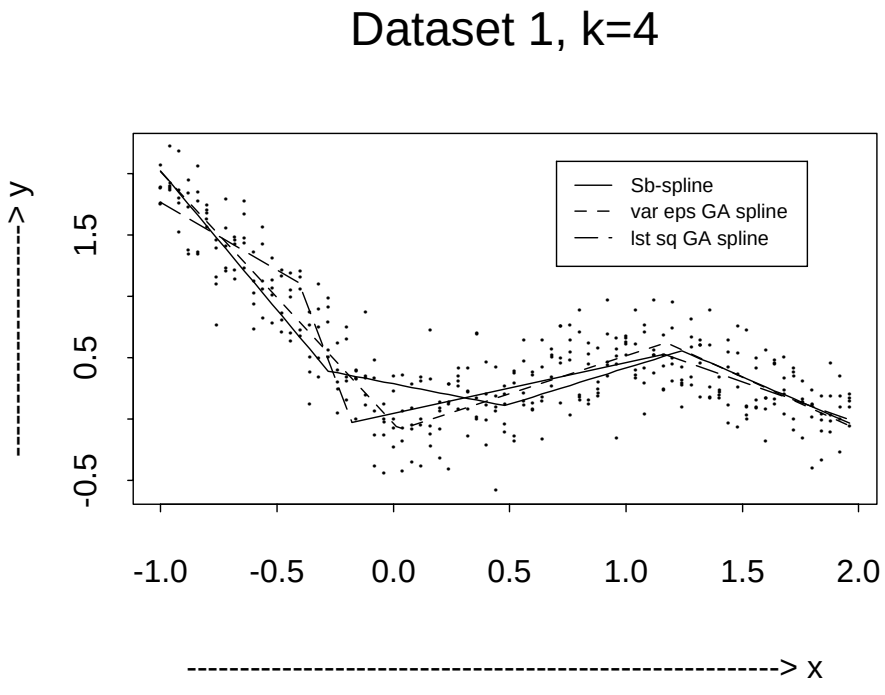


Figure 4.5 Results of Dataset 1

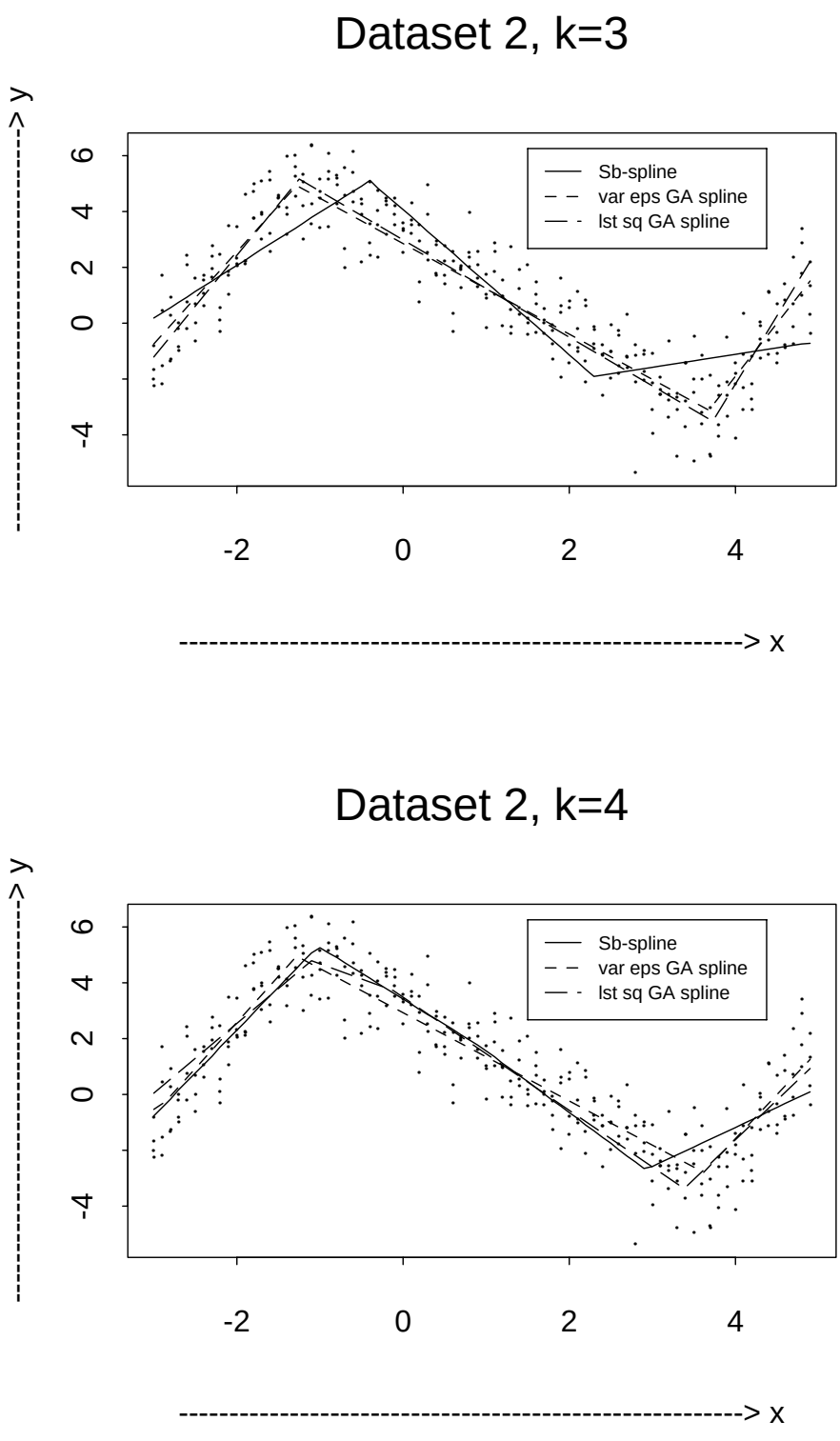
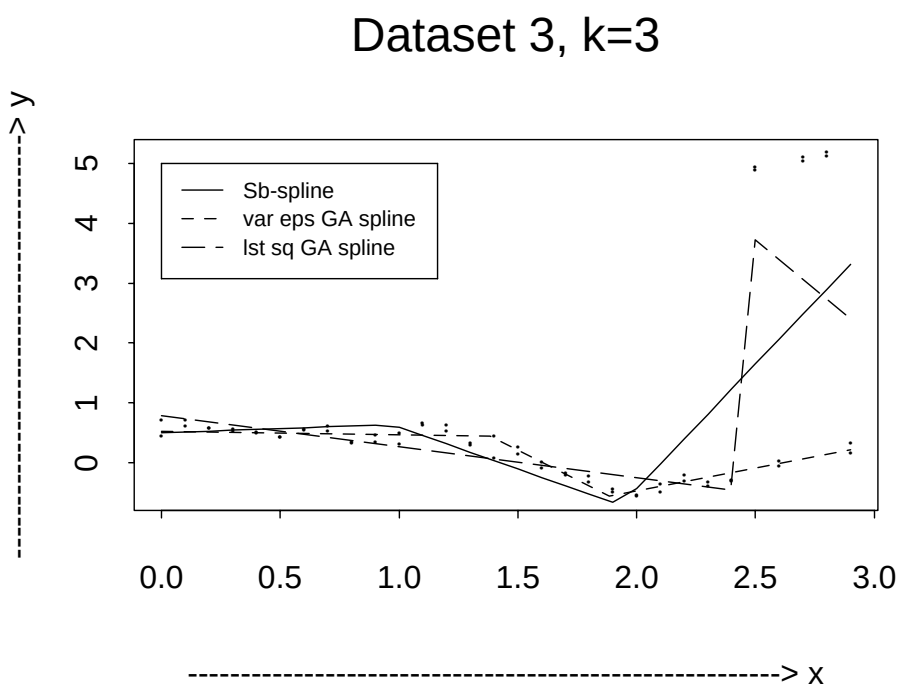


Figure 4.6 Results of Dataset 2



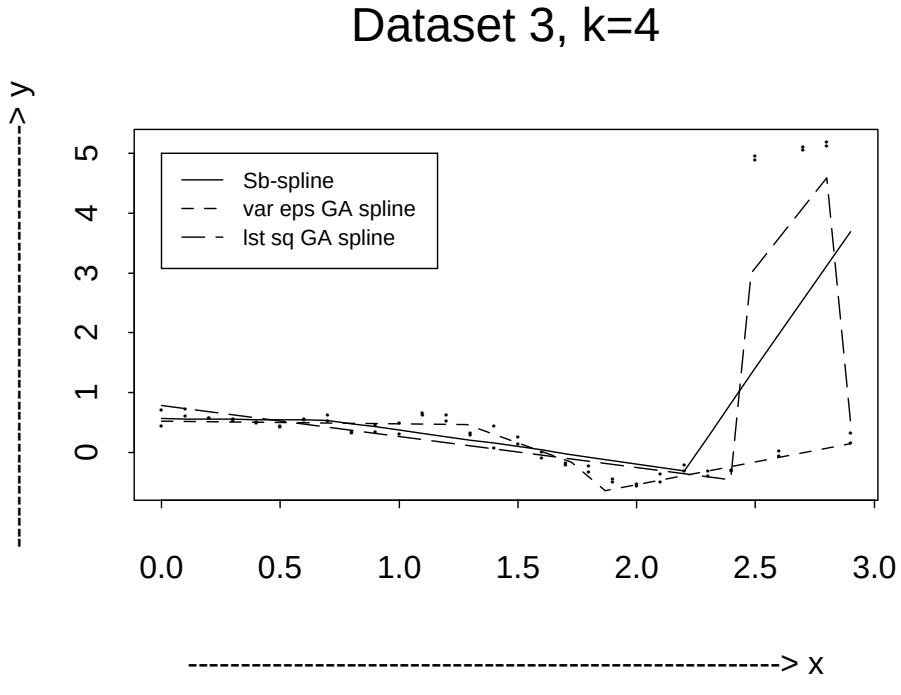


Figure 4.7 Results of Dataset 3

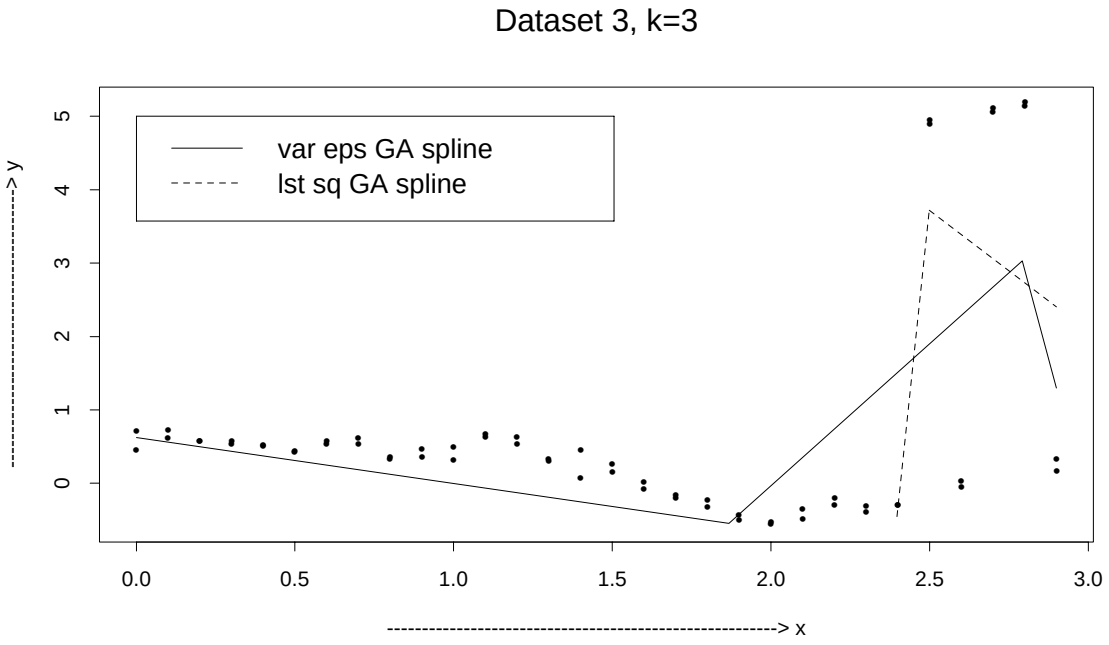


Figure 4.8 Comparison of var. eps. GA and lst. sq. GA when $crit = 0.95$

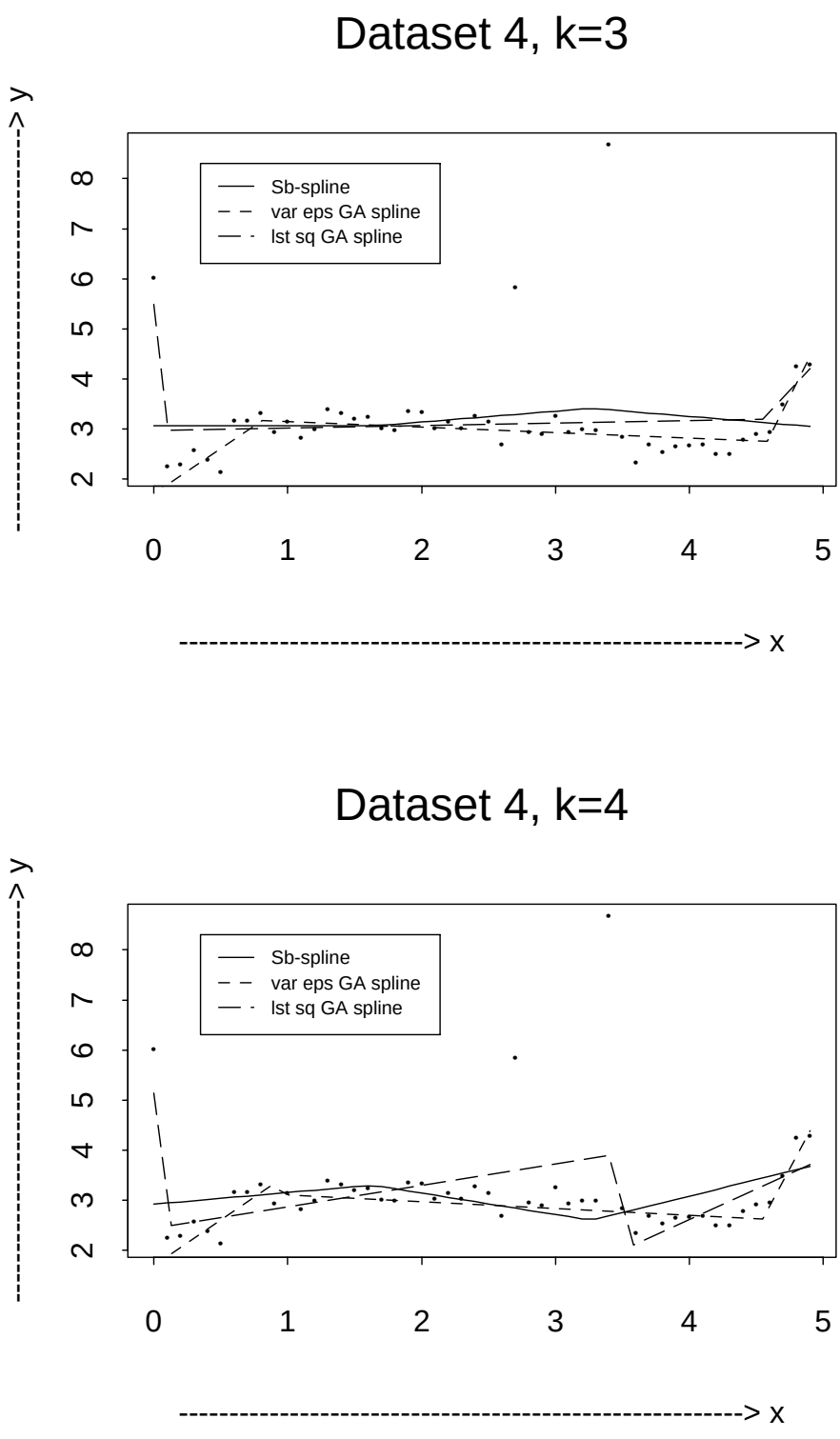


Figure 4.9 Results of Dataset 4

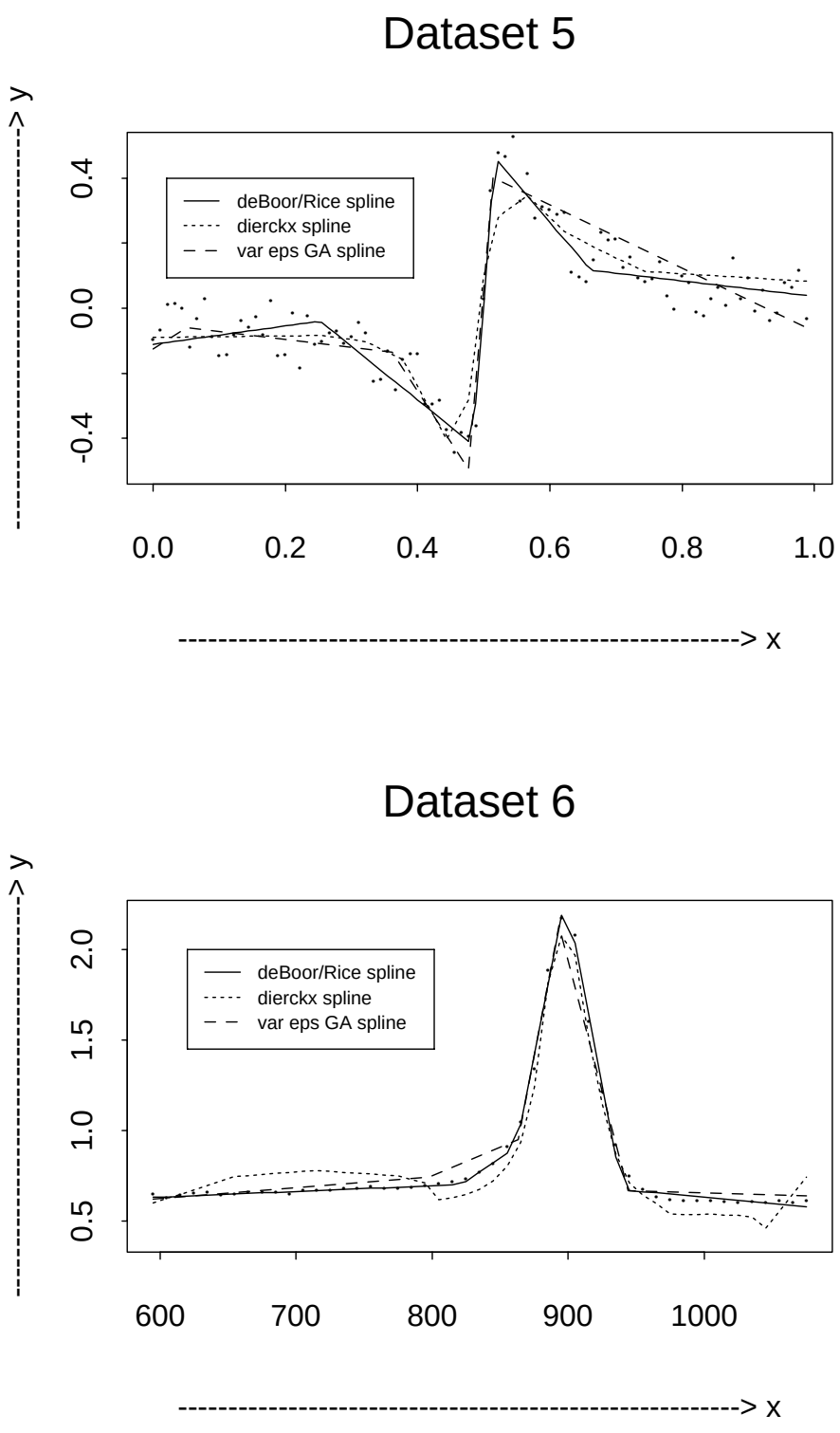


Figure 4.10 Results of Datasets 5 and 6

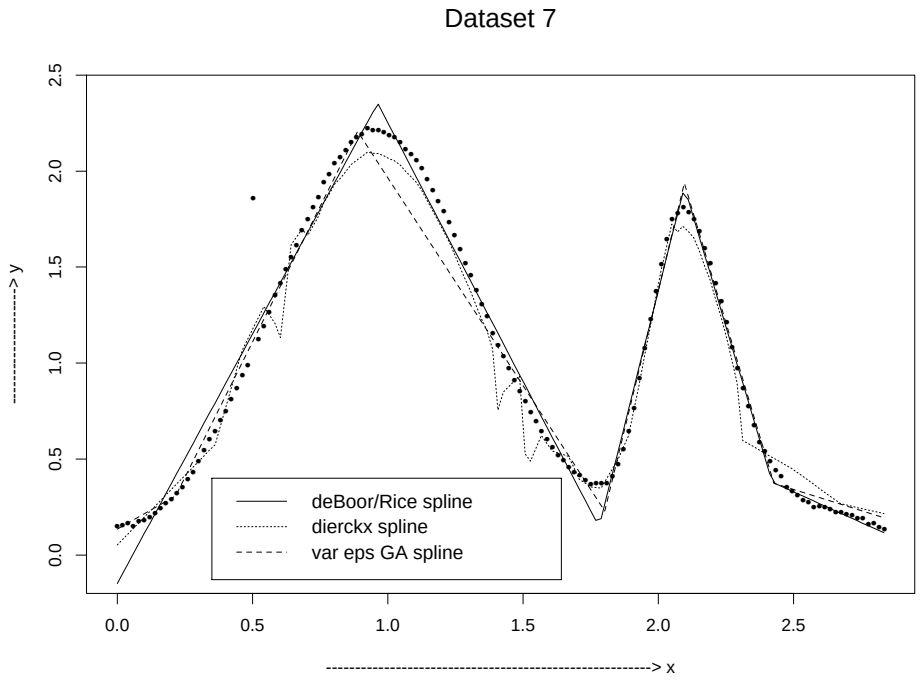


Figure 4.11 Results of Dataset 7

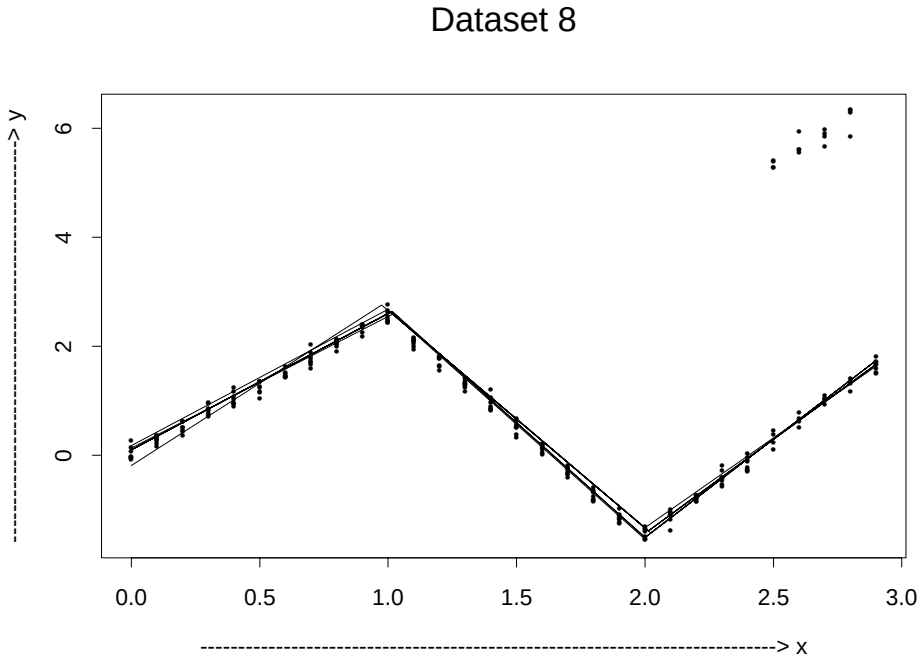


Figure 4.12 Example of Variability of Results

Chapter 5

Adaptive Genetic Splines

5.1 Introduction

Suppose we are asked to analyze a set of N measurements $\{(x_i, y_i) : i = 1, \ldots, N\}$ in $\mathcal{R}^2$, $a \leq x_1 < x_2 < \cdots < x_N \leq b$, where the system which generated the data can be described by an equation of the form $Y_i = f(X_i) + \epsilon_i$ for some unknown function f. The ϵ_is are assumed to be independent and follow some distribution with mean zero. To achieve the goal of finding a simple yet reasonable model for the dependence of Y on X, it is assumed that f can be approximated by a continuous piecewise polynomial satisfying certain continuity conditions on its lower order derivatives. In this context a rich, flexible class of models is the space of spline functions of order m with some knot sequence $\boldsymbol{Z}$, denoted $\mathcal{S}_{m,\boldsymbol{Z}}$.

As mentioned in Chapter 1, spatially adaptive smoothing methods such as adaptive spline modeling have become a popular and rapidly developing class of modeling techniques. Most of these methods are based on nonlinear optimization and/or stepwise selection. As discussed previously, the error surface relating to optimal knot selection is highly irregular, with many stationary points, and stepwise selection is suboptimal, with no way to escape local optima. A possible alternative to these methods is to use more intensive numerical optimization techniques such as genetic algorithms to perform knot selection. For example, Manela, Thornhill and Campbell [102] used a genetic algorithm to determine the appropriate order and

roughness penalty for penalized least squares (LS) splines; Rogers [G/SPLINES][123] incorporated a modified GA search into Friedman's Multivariate Adaptive Regression Splines [MARS] [64] algorithm.

In Chapters 3 and 4 the fitting of piecewise linear models via a GA based search was explored, first when the number of pieces was known (Chapter 3) and then when the number of pieces was unknown (Chapter 4). Theoretical support for modeling with GAs was provided in detail and a particular model fitting criterion was presented which allowed for model fitting with GAs in the presence of outliers. Empirical results were presented which showed that the results of GA based modeling in small parameter spaces are competitive with, although not clearly better than, those of existing methods.

In this chapter we will demonstrate that in larger parameter spaces employing a method for adaptive spline modeling which uses a genetic algorithm to optimize the choice of knot sequence, i.e., to determine the number and location of the knots, yields results which rival and are often superior to those of existing methods. Unlike the previous chapters, where the modeling techniques were restricted to the fitting of linear splines, the method presented here is capable of fitting splines of higher orders. The proposed method combines adaptive knot selection with the global optimization properties of a genetic algorithm, and hence can be characterized as a spatially adaptive smoothing technique which yields models which are near optimal with respect to the modeling criterion. We refer to this method as Adaptive Genetic Splines (AGS).

For a fixed number of knots the location of the knots is chosen to minimize the residual sum of squared errors (RSS); the appropriate number of knots is determined by selecting the model which minimizes a criterion which combines RSS and the number of 'degrees of freedom' used in the fit. The 'degrees of freedom' is a measure of model complexity based on the number of knot s or basis functions in the model and the adaptive nature of the GA search. By utilizing a criterion involving the 'degrees of freedom' of the model the hope is that a better balance between model variance and bias can be achieved, and that overparameterization, which is difficult to diagnose, may be avoided. In order to control string length, a genetic algorithm was devised which uses real valued strings as opposed to binary strings. Although we consider only model criteria based on least squares, our method can be generalized to spline fitting by more robust criteria. Simulation results show that the method's performance is comparable to and often rivals those obtained using Friedman's [64] MARS, the Hybrid Adaptive Splines [HAS] of Luo and Wahba [97], and the wavelet shrinkage techniques of Donoho and Johnstone [53, 54].

Empirical results pertaining to the contribution of the various AGS operators to the quality of the AGS model will be discussed, as well as how the AGS model develops in terms of model fit across iterations. The ability of the AGS algorithm and that of simple iterated hillclimbing to find a near optimal, parsimonious model will also be compared.

Our results clearly demonstrate that expending computational resources on the numerical optimization of the knot sequence, regardless of whether or not the optimization is based on evolutionary computation, will in most cases result in a

spline model of improved quality of fit - a quality of fit which rivals if not exceeds the quality of models fit by traditional methods.

Only a brief introduction to spline functions will be presented here; an excellent introduction to spline theory and applications can be found in de Boor [45], Wahba [42], Wegman and Wright [145], and Eubank [61]. A nice overview of algorithms for 'free' knot splines similar to the one given in Chapter 1 can be found in Schwetlick and Schütze [130].

5.2 Why Genetic Algorithms?

Genetic algorithms are stochastic search methods which, under certain design conditions, have been shown theoretically to converge to the global optimum of the evaluation function of an optimization problem [23]. Of course, since there are no results concerning the rate of convergence, except in a few limited cases, the above proof only assures us that a GA search is not biased in such a way as to prevent us from reaching the global optimum. However, as discussed in Chapter 2, given a variable selection criterion and a search space of possible knot locations, a genetic algorithm has the potential to find models that are more appropriate in comparison to models selected using stepwise or nonlinear optimization techniques.

The choice of knot sequence, in terms of both the number of knots and their placement, can be crucial for the shape as well as the quality of a spline fit [60]. In the context of the present problem, it is necessary to find not only the proper knot placement but also the proper number of knots. As discussed in Chapter 1, for a fixed number of knots, the sum of squared errors is a nonconvex function of the

knot sequence. For example, Jupp [85] notes that for the titanium heat dataset of de Boor [45], if one fits 9 variable knot B-splines of order 4 by least squares then the error surface has four stationary points - one interior optimum, two local minima, and one saddle point. Thus a traditional derivative-based approach may get stuck in a local minimum depending upon the choice of the initial knot sequence. As mentioned previously, with genetic algorithms one does have with positive probability the potential of escaping local optima. Additionally, unlike nonlinear optimization techniques, in order for convergence to a near optimal solution to occur one does not need very good initial estimates of the knot locations. This is important since good initial starting values are quite difficult to construct [95].

Determining the proper 'free' knot spline for a given dataset can be viewed as a variable selection problem: given a large set of candidate knot locations and a criterion of fit, find the subset of knots of a certain size which yields the best fit. Exhaustive enumeration is not an option due to the size of the domain space while the various stepwise and stagewise procedures popular in the literature are necessarily suboptimal [35]. Hence genetic algorithms, by performing a directed search over the model space without resorting to stepwise methods, have the potential to better address this sort of problem.

Given the above remarks a note of caution about GAs is in order. Although their convergence to the global optimum has been proven, the choice of the various algorithm parameters - population size, string length, crossover and mutation probabilities - does affect the rate of convergence. The nature of this dependence is

generally unknown so the selection of parameter values to achieve the best performance can be a difficult task. GAs are very computer intensive and hence slower than most existing methods. Finally, different runs of the same GA can lead to different results (although the results are usually reasonable solutions). Hence the nature of the current problem may motivate the development of a GA-based method, but GAs may not be an appropriate optimization tool in other modeling contexts.

5.3 Problem Statement and Solution Space

Consider a given dataset $\{(x_i, y_i) : i = 1, \ldots, N\},\ (x_i, y_i) \in \mathcal{R}^2,$ where $a \leq x_1 < x_2 < \cdots < x_N \leq b,\ a, b \in \mathcal{R}$, and the observed functional relationship between X and Y may be represented as

$$Y_i = f(X_i) + \epsilon_i \quad i = 1, \ldots, N$$

where f is a continuous function, $f \in \mathcal{C}[a, b]$, and the ϵ_is are independent and identically distributed with mean zero. The objective is to approximate the function f so that a criterion based on the (weighted) regression sum of the squared errors (RSS) (also called the sum of the squared errors (SSE)) is minimized.

To define the model space, let $\{\tau_i\}_{i=1}^{k+1}$ be a strictly increasing sequence of points in $\{X_i\}_{i=1}^{N}$ with $\tau_1 = a,\ \tau_{k+1} = b$ and k a positive integer, $k_{\min} \leq k \leq k_{\max},\ k_{\min},\ k_{\max}$ given. Let $\{P_i\}_{i=1}^{k}$ be a sequence of polynomials of order m (degree $m - 1$). Then the model space may be represented as

$$\mathcal{P}_{m,\tau} = \{g : g(X) = P_i(X) \text{ if } \tau_i < x < \tau_{i+1},\ i = 1, \ldots, k;\ \text{the first } (m-1) \text{ derivatives of } g \text{ at } \tau_i \text{ are continuous, } \{\tau_i\}_{i=1}^{k+1} \text{ and } \{P_i\}_{i=1}^{k} \text{ are given sequences, } k_{\min} \le k \le k_{\max}\}.$$

For the sake of computational efficiency and to avoid ill-conditioning, each element $g \in \mathcal{P}_{m,\tau}$ will be represented as a linear combination of normalized B-spline basis functions of order m. Hence for fixed k and given $\{\tau_i\}_{i=1}^{k+1}$ define $\boldsymbol{Z} = (Z_1, Z_2, \ldots, Z_{n+m})$ as

$$Z_1 = \cdots = Z_m = \tau_1 = a \le Z_{m+1} \le \cdots \le b = \tau_{k+1} = Z_{n+1} = \cdots = Z_{n+m},$$

$n = m + k - 1$, where each τ_i occurs once in $\boldsymbol{Z}$. Then $\mathcal{P}_{m,\tau}$ may be expressed as

$$\mathcal{S}_{m,\boldsymbol{Z}} = \{g \in \mathcal{P}_{m,\tau} :\ g = \sum_{j=1}^{n} \alpha_j B_{j,m,\boldsymbol{Z}},\ \alpha_j \in \mathcal{R},\ j = 1, \ldots, n\}$$

where $\{B_{j,m,\boldsymbol{Z}}\}_{j=1}^{n}$ represents the sequence of normalized B-splines of order m with respect to the knot sequence $\boldsymbol{Z}$. The explicit representation of $B_{j,m,\boldsymbol{Z}}$ in terms of m-th divided differences of the truncated power functions will not be given here; see de Boor [45].

Given $\boldsymbol{Z}$, the corresponding least squares spline of order m can be computed using existing algorithms [45]. The least squares solution is unique if and only if the matrix of B-spline coefficients $\mathbf{B}_{\boldsymbol{Z}}$ has full rank, i.e., if and only if $\|g\|_2 = (\sum_i (g(X_i)^2)^{1/2}$ is a norm on $\mathcal{S}_{m,\boldsymbol{Z}}$. One way to ensure that $\|\cdot\|_2$ is a norm is provided by the *Schoenberg–Whitney Theorem* [127]:

THEOREM 5.1 [*Schoenberg–Whitney*] The matrix $\mathbf{B}(\boldsymbol{Z})$ is invertible if and only if there exists a subset $\{X_{i_j}\}$ of $\{X_i\}$ such that

$$B_{j,m,\boldsymbol{Z}}(X_{i_j}) \neq 0, \ j = 1, \ldots, N$$

i.e., if and only if $Z_j < X_{i_j} < Z_{j+m} \ \forall \ j$. ♠
(of course it is assumed in the above that $X_{i_j} < X_{i_{j+1}} \quad \forall \ j = 1, \ldots, N-1$).

We enforce the above condition, which is equivalent to setting the minimum span of each basis function to one observation. It is known that the problem of best approximation always has a solution in the space of splines with multiple knots but such a solution does not always exist in the space of splines with simple knots [130]. However, for implementation reasons we have restricted ourselves to only simple knot splines, i.e., splines for which each $\tau_i, \ i = 1, \ldots, k+1$, occurs only once in $\boldsymbol{Z}$.

This choice of model space is motivated by the approximation properties of splines such as the following. Suppose that f is in the Sobolev space
$\mathcal{W}_2^q[a,b] = \{f : f^{(0)}, \ldots, f^{(q-1)}$ absolutely continuous on $[a,b], \ f^{(q)} \in \mathcal{L}^2[a,b]\}$
and the approximation g to f is defined as the minimizer of

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - g(X_i))^2 + \alpha \int_a^b (g^{(q)}(X))^2 dX$$

over all $g \in \mathcal{W}_2^q[a,b], \ \alpha > 0, \ \alpha \in \mathcal{R}$. The approximation g is a spline of order $m = 2q$ with simple knots at the predictor values $\{X_i\}$. Hence, as seen below, an adaptive genetic spline model uses a subset of the basis functions which may be used to construct traditional smoothing splines.

5.3.1 Model Criterion

A popular class of criteria for determining the appropriate model g are statistics based on generalized cross-validation (GCV)[42]. These measures of goodness-of-fit are based on the idea, borrowed from parametric linear regression, of minimizing the residual sum of squares adjusted for the amount of fitting being done by the model, or, in other words, for the increased variance associated with increased model complexity. Traditionally, the amount of fit associated with a given model is represented by the number of 'degrees of freedom' it employs. In spline fitting, the 'degrees of freedom' of a spline g is defined as a function of the smoothing matrix $\mathbf{S}$ where

$$g(\boldsymbol{Y}) = \mathbf{S}\boldsymbol{Y}.$$

Some common definitions [33] are:

1. $df = \mathrm{tr}(\mathbf{S}\mathbf{S}^t)$

   This definition is motivated by the equation for linear models

   $$\sum \mathrm{var}(\hat{Y}_i) = p\sigma^2$$

   where $df = p =$ number of parameters. The analogous definition for smoothers is $\mathrm{tr}(\mathbf{S}\mathbf{S}^t)$.

2. $df = \mathrm{tr}(2\mathbf{S} - \mathbf{S}^t\mathbf{S})$

   Note that if RSS denotes the residual sum of squared errors, then

   $$E(\mathrm{RSS}) = [N - \mathrm{tr}(2\mathbf{S} - \mathbf{S}^t\mathbf{S})]\sigma^2 + f'(\mathbf{I} - \mathbf{S})^t(\mathbf{I} - \mathbf{S})f$$

where the second term measures bias. Hence the definition $\text{tr}(2\mathbf{S} - \mathbf{S}^t\mathbf{S})$ since, once again, it is analogous to the linear regression case. If we are simply smoothing noise, so that $f = 0$, then this definition of df is the expected drop in RSS due to overfit. In the case of comparing two different fits this definition is particularly convenient since

$$E(\text{RSS}_1 - \text{RSS}_2) = [\text{tr}(2\mathbf{S}_1 - \mathbf{S}_1^t\mathbf{S}_1) - \text{tr}(2\mathbf{S}_2 - \mathbf{S}_2^t\mathbf{S}_2)]\sigma^2.$$

3. $df = \text{tr}(\mathbf{S})$

   This follows the definition of Mallows' C_p where the factor $2p\hat{\sigma}^2$ is added to RSS to make is unbiased for the predicted MSE.

Definition 3, as well as being easy to compute ($\text{tr}(\mathbf{S}) = \sum_i \lambda_i$, where λ_i is the ith eigenvalue of $\mathbf{S}$), is popular in the spline literature because, under the appropriate Bayesian assumptions, the posterior covariance of $\hat{\boldsymbol{Y}}$ is $\mathbf{S}\sigma^2$ [71]. This choice yields the original generalized cross-validation criterion (GCV)[42],

$$GCV(g) = \{RSS_g/N\}/\{(1 - \text{tr}(\mathbf{S})/N)^2\}$$

where RSS_g is the residual sum of squares from the fit of the model g to the data and $\mathbf{S}$ is the smoothing matrix corresponding to g. However, if 'degrees of freedom' is viewed as a measure of the amount of fit or the cost of the estimation process, then adaptively selected knots should employ more 'degrees of freedom' relative to nonadaptively selected knots. This, as well as the empirical observation that the flexibility of adaptive splines often leads to substantial overfitting [149],

motivates the desire to adjust the above criteria by altering the definition of 'degrees of freedom' to account for the nature of the search for basis functions. This leads to an adjusted GCV statistic of the form

$$GCV(g_n) = \{RSS_{g_n}/N\}/\{(1 - (n_1 + (n_2 * d))/N)^2\}$$

as used in MARS modeling [64] with $d = 3$, where n_2 of the n basis functions of which g is composed are adaptively selected, $n = n_1 + n_2$. Luo and Wahba [97] refer to the parameter d as the *inflated degrees of freedom* (IDF) factor and provide arguments for the use of $d = 1.2$ for reproducing kernel cubic spline bases. Although not a stepwise procedure, genetic algorithms also employ an adaptive procedure for the selection of basis functions. For this reason, and for the purpose of comparison with existing methods, a similar criterion based on RSS (or weighted RSS) will be utilized here. By utilizing a criterion based on adjusted degrees of freedom the hope is that a better balance between model variance and bias can be achieved, and that overparameterization, which is difficult to diagnose, may be avoided.

Whether adjusted GCV is an appropriate model criterion is open to further examination. If one accepts the choice of adjusted GCV, a value for d must be determined. The previously proposed values are based on arguments that depend upon the choice of basis functions and the nature of the modeling procedure [64, 97]. Thus they should not be directly applied to the proposed method. A possible solution is the concept of generalized degrees of freedom (GDF) of Ye [149] for complex modeling procedures - the techniques described by Ye could be employed

to determine the appropriate value for d in GCV for the GA procedure described here.

5.3.2 Approximation Properties

To justify the method outlined in this chapter we briefly discuss some approximation properties of spline functions. The convergence properties of genetic algorithms will not be discussed here; see Section 5.4.3 and Chapter 2.

Recall that the observed functional relationship between X and Y is assumed to have the form $Y_i = f(X_i) + \epsilon_i$ for some continuous function f where ϵ_i are $i.i.d.$ with mean zero. If our goal is to approximate f then our model class $\mathcal{S}_{m,\boldsymbol{Z}}$ should asymptotically contain elements whose distance from f is arbitrarily small. For example, Agarwal and Studden [3] have proven that if the number of (appropriately placed) knots increases at a rate of order $\mathcal{O}(N^{1/(2d+1)})$, where d is the dimension of X and N is the number of observations, then the order of the asymptotic integrated mean squared distance, or integrated mean squared error (ISME), of the estimated spline from an unknown function with m continuous derivatives is $\mathcal{O}(N^{-2d/(2d+1)})$. This is the best rate possible for a nonparametric estimator, as indicated by Stone [136]. With respect to another distance measure $\text{dist}(h, \mathcal{S}_{m,\boldsymbol{Z}})$, defined below, we will verify that the asymptotic distance of f from $\mathcal{S}_{m,\boldsymbol{Z}}$ can be arbitrarily small by stating the following theorems, whose proofs can be found in de Boor [45].

5.3.2.1 Distance from Splines

We first consider the distance of a function h from $\mathcal{S}_{m,\boldsymbol{Z}}$, where h satisfies various conditions. We defer the discussion of least squares approximation to the next section.

Define $\|h\| = \max_{a \le X \le b} |h(X)|$.

THEOREM 5.2 Let $g \in \mathcal{S}_{m,\boldsymbol{Z}}$ and $h \in \mathcal{C}[a,b]$. Let $|\boldsymbol{Z}| = \max_i \Delta Z_i$ where $\Delta Z_i = Z_{i+1} - Z_i$. Then

$$\operatorname{dist}(h, \mathcal{S}_{m,\boldsymbol{Z}}) = \min\{\|h - g\| : g \in \mathcal{S}_{m,\boldsymbol{Z}}\} \le c_m \cdot \omega(h; |\boldsymbol{Z}|)$$

where $\omega(h; |\boldsymbol{Z}|) = \max\{|h(r) - h(s)| : r, s \in [a,b], |r - s| \le |\boldsymbol{Z}|\}$ is the modulus of continuity of the function h and c_m is a constant whose value depends only on m.

♠

In the above theorem, c_m may be taken as $\lfloor (m+1)/2 \rfloor$.

Hence for any fixed order m, we can approximate h to any degree of accuracy if we are willing to use an arbitrary number of knots. In the case that h is smooth, the above bound can be improved considerably.

THEOREM 5.3 Let $\boldsymbol{Z} = \{Z_i\}_{i=1}^{n+m}$ satisfy

$$Z_1 = \cdots = Z_m = a < Z_{m+1} \le \cdots < b = Z_{n+1} = \cdots = Z_{n+m}.$$

Then for $j = 0, \ldots, m-1$ there exists $c_{m,j}$ such that for any $h \in \mathcal{C}^{(j)}[a,b]$,

$$\mathrm{dist}(h, \mathcal{S}_{m,\boldsymbol{Z}}) \leq c_{m,j} |\boldsymbol{Z}|^j \omega(h^{(j)}; |\boldsymbol{Z}|).$$

If $j = m-1$, then $\omega(h^{(m-1)}; |\boldsymbol{Z}|) \leq |\boldsymbol{Z}| \times \|h^{(m)}\|$ and the above imply

$$\mathrm{dist}(h, \mathcal{S}_{m,\boldsymbol{Z}}) \leq c_m |\boldsymbol{Z}|^m \|h^{(m)}\|. \spadesuit$$

Here $c_{m,j}$ may be defined iteratively as $c_{m,j} = c_m^{(j)} = \prod_{i=0}^{j} c_{m-i}$.

In comparison to the bound of Theorem 5.2, the distance of a smooth function from $\mathcal{S}_{m,\boldsymbol{Z}}$ goes to zero at a rate at least as fast as the jth power of the mesh size $|\boldsymbol{Z}|$.

In practice, the number of interior knots is bounded, i.e., $k \leq k_{\max}$, for some positive integer $k_{\max}$. Hence we must examine how well we can approximate f with a fixed number of knots whose placement is allowed to vary.

Let $\mathcal{S}_{m,n}$ denote all splines of order m with some knot sequence $\boldsymbol{Z}_{i=1}^{n+m}$ satisfying $Z_1 = \cdots = Z_m = a,\ Z_{n+1} = \cdots = Z_{n+m} = b,\ n$ fixed.

THEOREM 5.4 Assume that the function $h \in \mathcal{C}[a,b]$ is m times continuously differentiable at all but finitely many points and the $m-$th root of its $m-$th derivative is integrable, i.e.,

$$\int_a^b |h^{(m)}(X)|^{1/m} dX < \infty.$$

Then

$$\text{dist}(h, \mathcal{S}_{m,n}) \leq c_m n^{-m} \left(\int_a^b |h^{(m)}(X)|^{1/m} dX \right)^m . \spadesuit$$

Hence the order of approximation is n^{-m}.

This bound comes from using Theorem 5.3 and noticing that if $\tau_{i=1}^{k+1}$ is a breakpoint sequence chosen so that

$$\int_{\tau_i}^{\tau_{i+1}} |h^{(m)}(X)|^{1/m} dX = \frac{1}{k} \int_a^b |h^{(m)}(X)|^{1/m} dX, \quad i = 1, \ldots, k,$$

then

$$\text{dist}(h, \mathcal{P}_{m,\boldsymbol{\tau}} \bigcap \mathcal{C}[a,b]) \leq c_m (k)^{-m} \left(\int_a^b |h^{(m)}(X)|^{1/m} dX \right)^m .$$

Since $\mathcal{S}_{m,n} \subset \mathcal{S}_{m,\boldsymbol{Z}}$ for $k = n - m \leq k_{\max}$, the above order of approximation is a lower bound on the order of approximation attainable by functions from $\mathcal{S}_{m,\boldsymbol{Z}}$.

From the above theorems one can conclude that $\mathcal{S}_{m,\boldsymbol{Z}}$ is a reasonable approximating space for the given problem.

5.3.2.2 Least Squares

Least squares (LS) approximation is suitable when the objective is to recover information about a functional relationship from a set of noisy observations. From the above theorems we know that by allowing for many knots, our model class will contain a function whose distance from f is arbitrarily small. Hence, as $N \to \infty$, fitting by least squares will theoretically yield a model whose distance from f is

negligable. This is one consideration which motivated our use of least squares in combination with splines for model fitting.

There are some practical considerations, however. First, one can achieve a least squares model simply placing a knot at each data point and interpolating the data. Given a noisy dataset this is something to avoid; thus not only should the number of knots be less than the number of data points but the model fitting criterion should penalize models with many terms to avoid 'overfit'. For a fixed number of knots we are finding the LS solution; the penalty provides a way to determine the 'proper' number of knots. Hence an adjusted sum of squares is a reasonable choice for our fitness function. Another consideration in criterion selection is that, from a statistical standpoint, we desire that the final model be *parsimonious* - for the same amount of error, we prefer a model with few terms. Thus not only should we choose a model fitting criterion which penalizes models with many terms, but we should also select a model class containing those spline functions which maximize the amount of information contained in each knot. Variable knot splines, in combination with an optimization of knot placement, represent such a combination of modeling criterion and class.

The amount of improvement in terms of model fit that one can achieve by allowing the knots to vary is, of course, dependent upon the given dataset. On 'nice' datasets with minimal local variability, one would expect less improvement over a spline model with fixed knots or an 'optimal' knot distribution (see Chapter 1 and the discussion following Theorem 5.4), as compared to the improvement one would expect on datasets with local properties where the ability of adaptive knots

to work as local bandwidth smoothers can be more advantageous. However, given any dataset, optimizing knot placement is attractive when the number of knots in the final model is of importance. Since our objective is a parsimonious model, optimizing knot placement is of key importance.

5.4 Adaptive Genetic Splines

For each fixed k, $k_{\min} \leq k \leq k_{\max}$, a genetic algorithm can search for the model of size k which optimizes the given variable selection criterion. Each candidate model is represented as a string or chromosome where each interior knot location is represented by a fixed number of characters.

5.4.1 Nonbinary Genetic Coding

Bit encoding was initially considered for AGS, but as N and/or k increased, the string length quickly became unmanageable. Hence, in AGS, integer coding is used. Each string represents the locations of the interior knots of an adaptive spline model, where knots are restricted the set $\{x_i\}_{i=1}^{N}$. For example, the string (2 5 8 12 16 29) represents a model with 6 interior knots located at x_2, x_5, x_8, x_{12}, x_{16} and x_{29}. The use of nonbinary codings is supported by theoretical work [108, 119] as well as experimental results [43, 84]. As discussed in Chapter 2, classic schema theory suggests that small alphabets are favorable because they allow the GA to process more schemata. So why use a large alphabet? Janikow and Michalewicz [84] present several reasons, including increased stability, increased speed, and increased precision. Their experiments with binary and floating point representations support

the argument that special operators can enhance the performance of floating point representations and help compensate for the loss of the well known advantages of low-cardinality alphabets. With respect to theory, several authors have extended schema theory to real-coded genetic algorithms; see, for example, Goldberg [68] for one approach involving virtual alphabets. Two specific reasons for the use of higher cardinality alphabets will be discussed here: avoidance of Hamming cliffs and dimensionality reduction.

As noted by Goldberg [68], mutation operators for real-coded GAs usually perterb the current solution only a little about the current value and binary mutation operators usually adopt a bitwise complement operator for mutation. Hence, by creeping around, real-coded GAs can easily hill-climb the underlying solution space while with bitwise complement mutation one can get stuck on Hamming cliffs like the one shown in Figure 5.1.

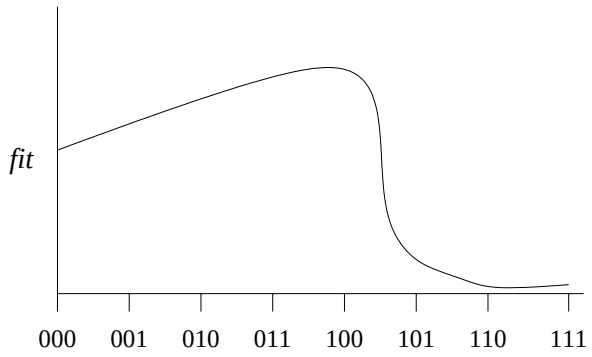


Figure 5.1 A Hamming cliff in binary space

The points to the left have above average fitness values (since $fit(0**) > fit(1**)$) so a GA is likely to quickly converge to 011. In other words, the lower

order schemata $0**$ does not contain the optimal string and hence leads the GA away from the global optimum. This string is close to 100 in decision space but not in Hamming space: to move from 011 to 100 requires changes at all three bit locations. This event is highly unlikely - of order p_m^3 - and is likely to require a long waiting time. This problem may be solved with binary codings by combining both bitwise and decision-based mutation operators.

A second reason for prefering higher cardinality alphabets is dimensionality reduction, which reduces the chance of *deception* [67] and crossover disruption. In essence, deception occurs when low-order building blocks lead in the opposite direction of high-order building blocks containing the global optimum. Dimension reduction can decrease the opportunity for deception because there are fewer low-order building blocks. Of course, this only reduces the opportunity for deception; deception can still exist. The reduction of crossover disruption can be beneficial, since it increases the chance that selection will choose good alleles. However, it also prevents low-order building blocks from directing the search toward promising, previously unvisited areas of the search space. The balance of advantage vs. disadvantage must be determined with each specific application.

5.4.2 Operators

The integer coded genetic algorithm (ICGA) which has been implemented uses a simplified linear ranking selection in combination with stochastic sampling with replacement [108]. The performance of the standard linear ranking selection of Baker [12] with stochastic universal sampling will be tested in future studies.

Stochastic universal sampling is one of the most efficient techniques and hence should reduce the computational expense of AGS.

In the initial implementation simple crossover was utilized; however, numerous crossover operators have been developed for real coded genetic algorithms (RCGAs) which have been shown to lead to improved performance [73]. Among these is BLX-α crossover, whose performance will be studied in future simulations.

The works of various authors [73, 101, 119] suggest an adaptive mutation scheme, e.g., applying different mutation densities and varying the probability of mutation during the GAs generations. In the initial implementation a triangular mutation scheme [109] was utilized, whereby a the new value of a position selected for mutation was randomly chosen from a triangular distribution centered at the current value. Nonuniform mutation [73] should also perform well.

The mutation probability p_m^t varies from 0.9 to $1/L$; this choice is supported by the work of Bäck [9]. Note that with integer strings, crossover only occurs at the boundary between parameter values and hence does not add to mutation. Hence mutation rates were set higher than usual, as in Bramlette [28]. An elitist step is included to ensure that the current best solution is retained as the algorithm progresses.

5.4.3 Convergence

Recall from Chapter 2 that the proof of Bhandari, Murthy, and Pal [23] of the convergence of a GA to the global optimum depended upon two characteristics: the optimal string in the current population having a fitness value no less

than the fitness values of the optimal strings from the previous populations, and that each string have a positive probability of going to an optimal string in one iteration. Eiben, Aarts, and Van Hee [59] refer to the first characteristic as *monotonicity* and the second characteristic as *generosity*. However, in their proof of almost sure convergence the requirement of generosity is replaced by the following: Let fit be the function to be optimized and P^0 denote the initial population. Define $\mathrm{P}^* = \{\mathrm{P} : \mathrm{P} \text{ contains an optimum of } fit\}$. Let the set of possible successors of a population P be $succ(\mathrm{P}) = \{\dot{\mathrm{P}} : \exists\ t \in \mathcal{N} : \mathcal{P}(\mathrm{P}^t = \dot{\mathrm{P}}) > 0\}$. Then

$$\exists\ n_k \in \mathcal{N} \text{ and } \epsilon_k \in (0,1]\ (k \in \mathcal{N}) \text{ such that } n_k \to \infty\ (k \to \infty) \text{ and } \prod_{k=0}^{\infty} \epsilon_k = 0$$
$$\text{and } \forall\ \dot{\mathrm{P}} \in succ(\mathrm{P}^0),\ \mathcal{P}(\mathrm{P}^{n_k+1} \cap \mathrm{P}^* = \emptyset \mid \mathrm{P}^{n_k} = \dot{\mathrm{P}}) \leq \epsilon_k$$

holds for every $k \in \mathcal{N}$.

AGS satisfies the property of monotonicity due to the elitist step of the algorithm. However, only if the mutation probabilities are chosen such that the above requirement of generosity is satisfied will the convergence proofs mentioned above (e.g., if the probabilities allow any character to change to any other character in one iteration) apply to AGS. For the simulation studies described below the probability of mutation was chosen such that the above requirements would be satisfied.

5.4.4 Simulation Studies

Simulated examples were used to examine the performance of AGS compared to the MARS [64], HAS [97], and POLYMARS [90] algorithms and the wavelet shrinkage methods of Donoho and Johnstone [53, 54]. The simulation studies performed here were modeled after those designed by Luo and Wahba for the HAS program; this was done primarily to facilitate comparison with existing methods. Table 1 gives information about the simulated datasets. Example 1 is taken from Donoho and Johnstone [54] and shows considerable spatial inhomogeneity, as does Example 2 and Example 3 from Schwetlick and Schütze [130]. Examples 4 and 5 are from Fan and Gijbels [63] and show less spatial variability. Gaussian noise was used for all examples except Example 3 where the noise was uniformly distributed. To facilitate comparison with wavelet methods, the sample sizes were all powers of 2 and the designs were equally spaced for all examples except Example 2.

Table 5.1 Simulated Examples

Ex.	f	σ	Sample size (N)	SD$(f)/\sigma$	Number of replicates
1	DJ(1994) Heavisine*2.2	1.0	2,048	6.54	31
2	$(4\sqrt{x(1-x)})(2.1\pi/(x+0.25))$	1.0	2,048	6.48	31
3	$10(4x-2)/(1+(100*((4x-2)^2)))$	0.06	128	3.33	400
4	$\sin(2(4x-2))+2\exp(-16(4x-2)^2)$	0.3	256	2.80	400
5	$(4x-2)+2\exp(-16(4x-2)^2)$	0.4	256	3.16	400

For wavelet shrinkage the SUREShrink method of Donoho and Johnstone [53] was selected with a "primary resolution level" of 5. Computations were performed

with the *wavethresh* software of Nason and Silverman [112] in S-PLUS [82], with the S-PLUS commands provided by Luo and Wahba. The family of wavelets was *DaubLeAsymm* with $filter\ number$ 8.

The maximum number of basis functions in HAS, MARS, and POLYMARS was set at 150 for Examples 1 and 2 and at 60 for Examples 3-5. The number of basis functions considered by AGS was $[18, 34]$ for Example 1, $[19, 39]$ for Example 2, and $[7, 16]$ for Examples 3-5. (The number of basis functions fit by AGS was near 21 for Example 1, 35 for Example 2, and 9 for Examples 3-5). The remaining parameters in HAS, MARS, and POLYMARS were set at their default values except for the parameter *gcv* in POLYMARS, which was set at 2.5 as in Stone et al. [137]. The IDF factor (i.e., the value d from Section 5.3.1) for AGS was set at 3 for comparison purposes.

AGS was set to fit cubic splines with a maximum of 400 generations, a crossover probability of 0.8, and the same mutation probability density (as described above) for all examples. The population size M was 50 for Examples 1 and 2 and 40 for Examples 3-5. The random number generator used in AGS was the Fortran subroutine `runi` from CMLIB [105].

For each example and each method, the MSE for each replicate was recorded (note the number of replicates for each dataset in Table 5.1). The median MSE and the difference between the 1st and 3rd quartiles of the MSE for the set of replicate results for each dataset and each method were then calculated. These results are given in Table 5.2. For Examples 1 and 3-5, the median results for all methods are

shown in Figures 5.7 and 5.9-5.11; the median results for AGS and HAS for Example 2 are shown in Figure 5.8.

Table 5.2 Median MSE (Difference between First and Third Quartiles of MSE)

Example	AGS	HAS	SUREShrink	MARS	POLYMARS
1	.0195 (.0065)	.0412 (.0085)	.0702 (.0397)	.1518 (.0134)	.2483 (.0067)
2	.0411 (.0243)	.0569 (.7784)			
3	.0004 (.0002)	.0006 (.0003)	.0013 (.0003)	.0009 (.0009)	.0016 (.0002)
4	.0052 (.0035)	.0071 (.0059)	.0178 (.0038)	.0070 (.0038)	.0087 (.0034)
5	.0098 (.0062)	.0128 (.0114)	.0464 (.0162)	.0126 (.0068)	.0104 (.0055)

The performance of AGS compares quite favorably with the results of SUREShrink and the other adaptive spline algorithms. On all examples AGS achieved the lowest median MSE, most likely due to the directed global selection of basis functions. For Examples 1, 4, and 5, the results of HAS, SUREShrink, and MARS reflect those shown in Luo and Wahba [97]. The selection of a lower "primary resolution level" for SUREShrink did not yield better performance, although the choice of another wavelet family may have led to improved models. Neither MARS nor POLYMARS gave reasonable results for Example 2 - for example, the median MSE was greater than 5 for MARS and greater than 9 for POLYMARS. However, these methods were specifically designed for high dimensional problems and not functions with extreme spatial heterogeneity. There are no wavelet results for Example 2 because the design points for this example were not equally spaced.

Since an appropriate value of d was unknown, experiments were run on each dataset with values of d ranging from 1 to 5. It was observed that the result of AGS was relatively insensitive to values of d between 2 and 4; values outside of this range lead to either undersmoothing ($d < 2$) or oversmoothing ($d > 4$).

5.4.5 Remarks

5.4.6 AGS fitness landscape

Goldberg [68] maintains that the operation of selection dominates early genetic search, essentially restricting the availalble search space to subspaces of above average fitness value. Once this reduction has been performed, the GA spends the remaining iterations exploring these remaining subspaces, searching meticulously for a global optima.

This theory implies that a graph of the fitness value of the current best solution over the iterations of the GA will show a sharp initial increase, followed by a long period of small improvements. Figure 5.2 of the fitness values for replications corresponding to the first experimental dataset (see Table 5.1) demonstrates exactly this type of pattern. The search pattern can be characterized by an early, marked increase in the fitness value of the current solution, followed by a concentrated search of above average subspaces. This concentrated search, in which the GA makes small moves and adjustments in its search for the best string, yields a series of relatively minor, intermittent improvements. It appears that the GA quickly attains a reasonable solution and then spends most of its iterations doing local searches, looking for minor improvements which may lead to the global optimum.

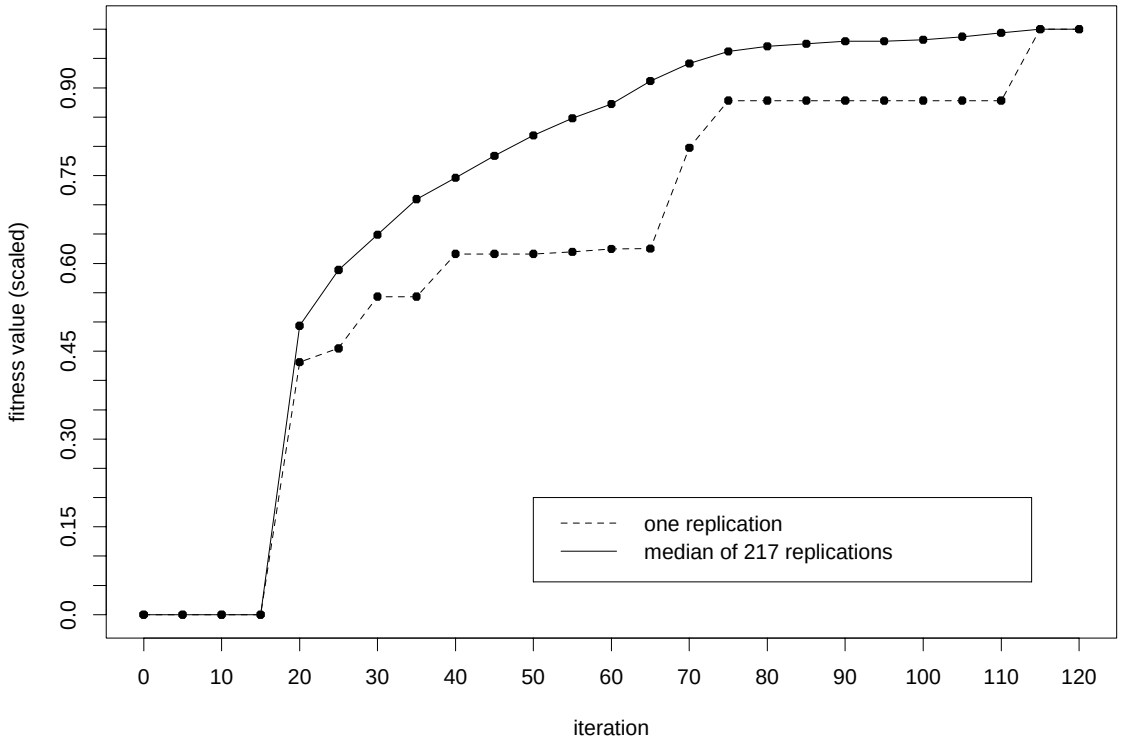


Figure 5.2 AGS fitness landscape of experiments with example 1

5.4.6.1 Crossover and mutation

Experiments with AGS were performed where either no crossover or no mutation was used. Preliminary results for Example 1 are shown in Figures 5.3 and 5.4. It is observed that without mutation AGS suffered a noticeable loss in performance (3% on average), in contrast to the loss of crossover which had little effect on the quality of the results (-0.07%, or a *gain* of 0.07% on average).

These empirical results are surprising in light of the theoretical schools of thought concerning how genetic algorithms work (see Chapter 2), which seem to indicate that crossover is the driving force behind GA optimization. Other researchers

have reported similar phenomenon [34], indicating that the prevailing theory may be suspect.

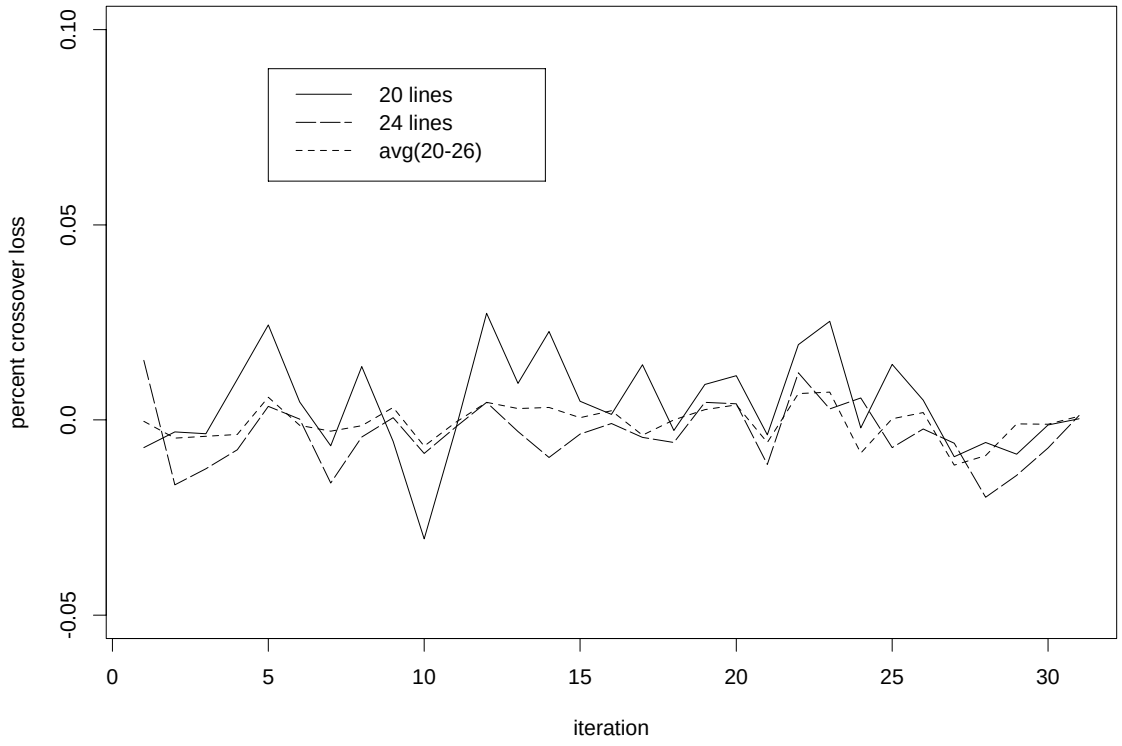


Figure 5.3 Percent loss in fitness value due to no crossover

5.4.6.2 Hillclimbing

Since the number of iterations of AGS necessary for convergence is unknown (and is unknown for genetic algorithms in general), we added a nearest-neighbor hillclimber to the end of the algorithm to see if this would improve performance. For each model size the hillclimber takes the solution provided by AGS and determines

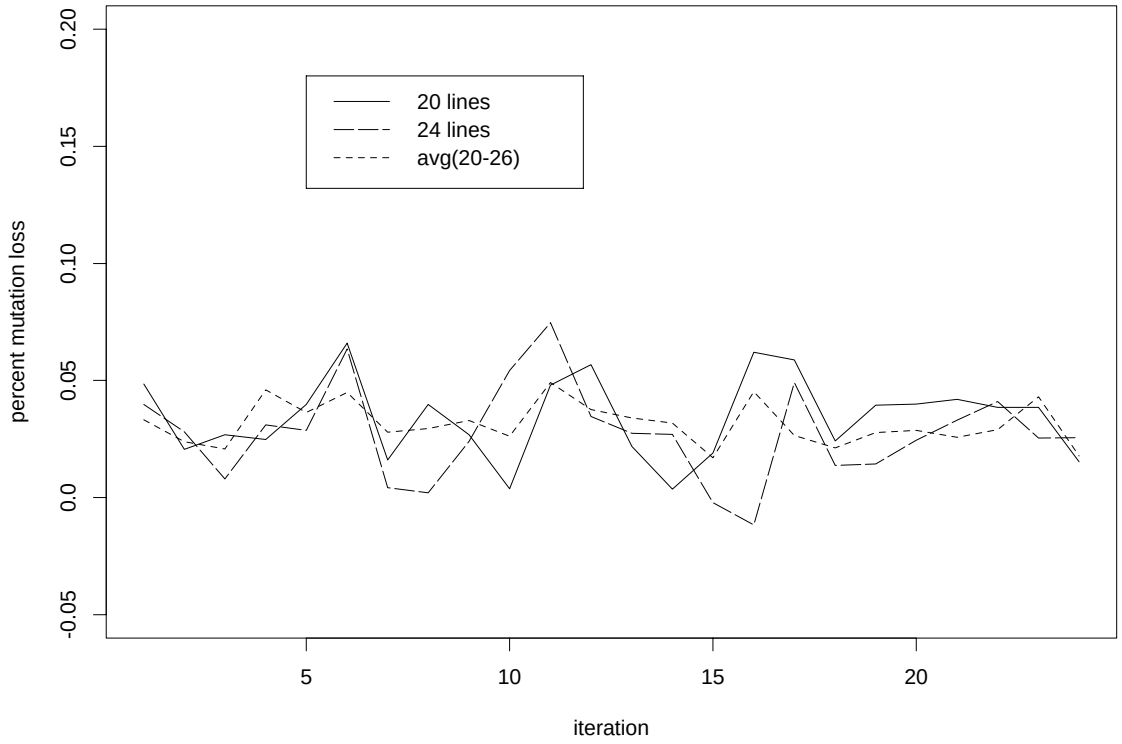


Figure 5.4 Percent loss in fitness value due to no mutation

for each knot whether or not moving the knot one place to the left or right would improve the model fit; if so, the knot is moved. The moving of knots is performed iteratively until any proposed move fails to improve upon the current model. Figure 5.5 shows some preliminary results with Example 1. The improvement in fitness value across 31 iterations for two model sizes and the average improvement over 10 model sizes achieved by using the hillclimber are shown. The hillclimber did improve the quality of the models, although only slightly (the median improvements for both model sizes and the average are 0.22%, 0.20%, and 0.21%, respectively). Although the improvement in this case is marignal, in situations where the number of iterations

is restricted, either by computational expense or availability, the hillclimber could prove to be quite valuable.

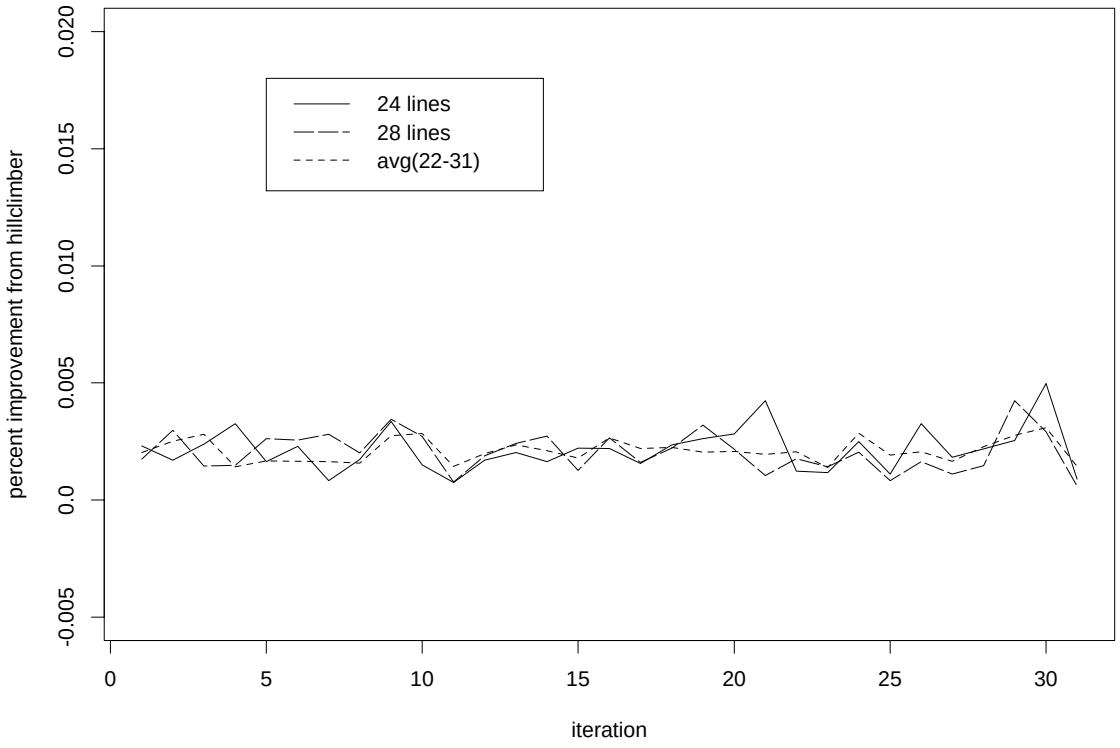


Figure 5.5 Percent improvement in fitness value due to hillclimber

5.4.6.3 Error bias

The above results show that AGS does have promise as an adaptive smoothing technique. In order to firmly establish its performance, however, additional simulation studies are necessary. As mentioned previously modeling with other families of wavelets should be done for comparison purposes and the set of test functions

should be expanded to include more spatially inhomogeneous functions. We also realize that using the same set of ordinate values for model fitting and model testing leads to biased measures of accuracy and performance. Hence in future simulations, a method which uses different ordinate values for fitting than for testing should be used. For example, once the median model for each method has been determined using the replicated datasets, a new set of replicate datsets should be generated using a new set of ordinate values and these new replicates should be used in calculating the MSE statistics shown in Table 5.2.

5.4.6.4 Implementation issues

Despite the relatively superior performance of AGS on the above examples, the method does have several implementation issues to be resolved.

- Due to the global nature of the GA search, it is relatively slow and computationally expensive (e.g., for Example 1 with sample size 2,048, each model size took approximately 90 seconds with the number of basis functions ranging from $[18, 34]$); this limits the number of model sizes which can be considered. Hopefully the efficiency of AGS can be improved by the use of more ICGA specific genetic operators (as mentioned above) and improved programming. One may also consider partitioning the dataset and fitting an AGS model to each partition separately.

- Crossover and mutation can create models with coincident knots. It is possible that this can be avoided by incorporating constraints into the GA model as in Michalewicz and Janikow [107].

5.5 Application

A study of semiconductor wafer fabrication is currently being conducted by Professor Roy Maxion of the School of Computer Science at Carnegie Mellon University. The purpose of the study is to design a computer system for process control which can detect faulty wafers and diagnose the cause of the fault during production using control data. In order to perform this task, the computer system must be able to classify wafers as either acceptable or faulty and sub-classify faulty wafers according to the cause of the fault.

AGS is being used in the initial stages of this study as a tool for modeling the process control data. The data consists of 13 variables measured for each wafer as a time series from the beginning to the end of wafer production (forming 13 *waveforms* for each wafer). The data for each variable is highly spatially inhomogeneous with many abrupt increases and decreases in value so an adaptive modeling scheme is required. As an initial approach, AGS will be used to model each waveform using linear splines and the number and location of the knots (i.e., the number of lengths of the linear segments) of the AGS model will be used for classification purposes.

An example waveform for one variable and wafer and the corresponding AGS model (with the knot locations identified) are shown in Figure 5.6. AGS takes approximately 20 seconds to fit each model size. Further research will determine whether AGS models provide the information needed to accurately classify wafers as acceptable, faulty, and, if faulty, by type of fault.

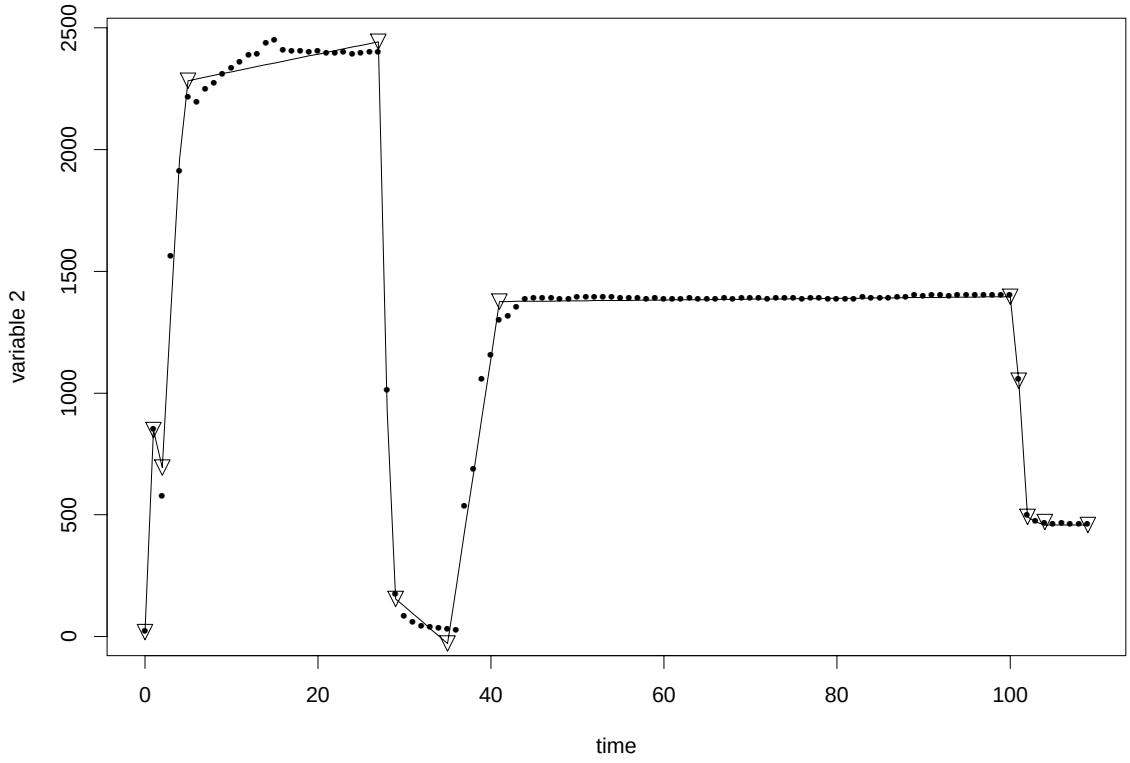


Figure 5.6 Waveform of semiconductor wafer process control data and AGS model

5.6 Summary

In this chapter a modeling technique referred to as Adaptive Genetic Splines (AGS) has been proposed for fitting adaptive splines using genetic algorithms. The basis functions are B-spline basis functions of order m ($m < 20$); in its current form the method can be used to fit adaptive splines which minimize a weighted or unweighted adjusted GCV criterion. Model fitting with other criteria is also possible simply by redefining the fitness function of AGS. For each candidate model size, AGS uses a GA to search for the optimal model by adaptively selecting the appropriate knot sequence.

The empirical results demonstrate the advantage of using intensive numerical optimization techniques such as genetic algorithms to search for the knot sequences of adaptive spline models. It has been noted by several authors that by allowing the knots of a spline model to be 'free', one can significantly improve the fit of a spline model. Traditional methods for optimizing the number and location of the knots generally use either nonlinear optimization or stepwise techniques to search for the optimal knot sequence. However, fitting adaptive splines is a very difficult nonlinear optimization problem, since the error surface contains many stationary points, and stepwise and stagewise methods are necessarily suboptimal. Unlike traditional methods, a genetic algorithm does not require good starting values and has the potential of escaping local minima in the error surface. Although there do not exist general guidelines for determining the appropriate number of iterations for a GA, it is known that a GA will converge to the global optimum of an optimization

problem given a sufficient number of iterations. In other words, a GA search is not biased in a way that would prevent the algorithm from reaching the global optimum.

In addition to the implementation issues discussed in the previous section, future research will focus on the following:

- The proof of convergence will be extended in detail to the GA designed here, to ensure theoretically that the algorithm can reach the global optimal solution.
- Although Rogers [123] reports that the performance of G/SPLINES was slightly inferior to that of MARS, it should be included in future simulation studies.
- The choice of inflation factor for the GCV criterion needs to be determined; this should be done by both simulation and the method of Ye [149]. The statistical literature contains a rich class of model selection criteria based on different measurements of model error and degree of model fit. These include statistics based on Mallows' C_p [100], AIC [81, 137], and various GCV criteria ([42, 64, 97]). Our choice of model selection criteria, although supported by statistical arguments, is mainly heuristic. Other data analysts may prefer to use different criteria; for this reason we intend to adapt our procedure for use with criteria other than adjusted GCV.

We also recognize that least squares methods have problems with outliers, so one may prefer to use a more robust model selection criterion. Here the residual sum of squared errors can be replaced by a function of the form

$$\sum_{i=1}^{N} \psi\left(\frac{Y_i - g(X_i)}{\varsigma}\right)$$

where ς is a scale measure, e.g., $\hat{\sigma}$. For a fixed set of basis functions $\{B_{j,m,\boldsymbol{Z}}\}_{j=1}^{n}$ this leads to the modified normal equations

$$\sum_{i=1}^{N} B_{j,m,\boldsymbol{Z}}(X_i)\psi(\frac{Y_i - g(X_i)}{\varsigma}) = 0 \qquad j = 1, \ldots, n.$$

Popular choices for ψ are the M-estimate [93]

$$\psi(X) = \begin{cases} X & \text{if } |X| \leq c \\ c \text{ sign}(X) & \text{otherwise} \end{cases} \tag{5.6}$$

for some constant c or Andrews' [5] sine function

$$\psi(X) = \begin{cases} a^2(1 - \cos(X/a)) & \text{if } |X| \leq \pi a \\ 2a^2 & \text{otherwise} \end{cases}$$

where a is a given constant. Lenth [93] has fit robust cubic splines with ψ as given in Equation 5.6; the implementation of a similar fitting procedure combined with GA knot selection is an interesting problem for future research.

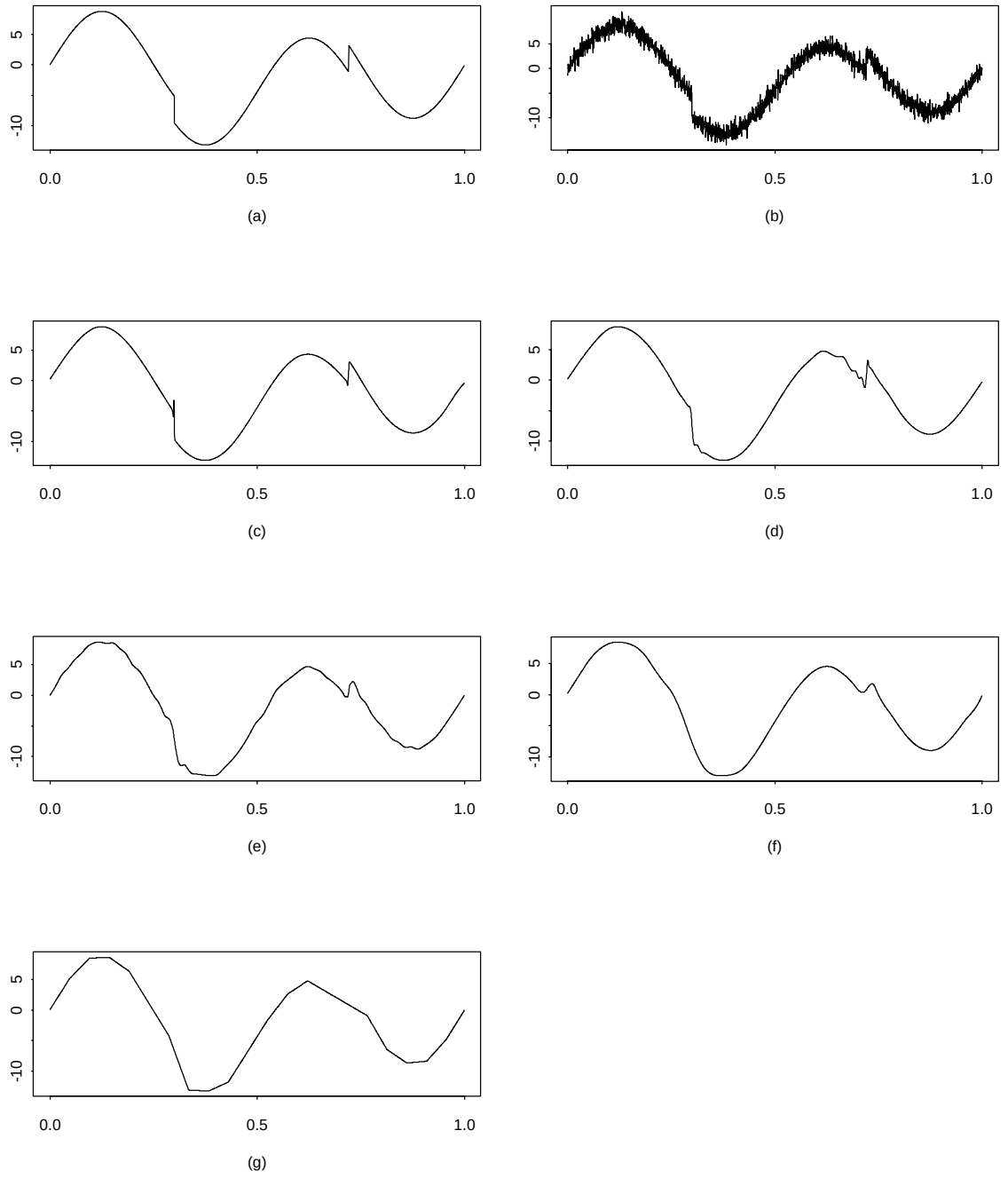


Figure 5.7 Example 1:(a) original function; (b) sample dataset of 2,048 observations from original function with added Gaussian ($\sigma = 1.0$) noise; (c) AGS fit with median MSE ($\sigma = .0195$); (d) HAS fit with median MSE ($\sigma = .0412$); (e) SUREShrink fit with median MSE ($\sigma = .0397$); (f) MARS fit with median MSE ($\sigma = .1518$); (g) POLYMARS fit with median MSE ($\sigma = .2483$)

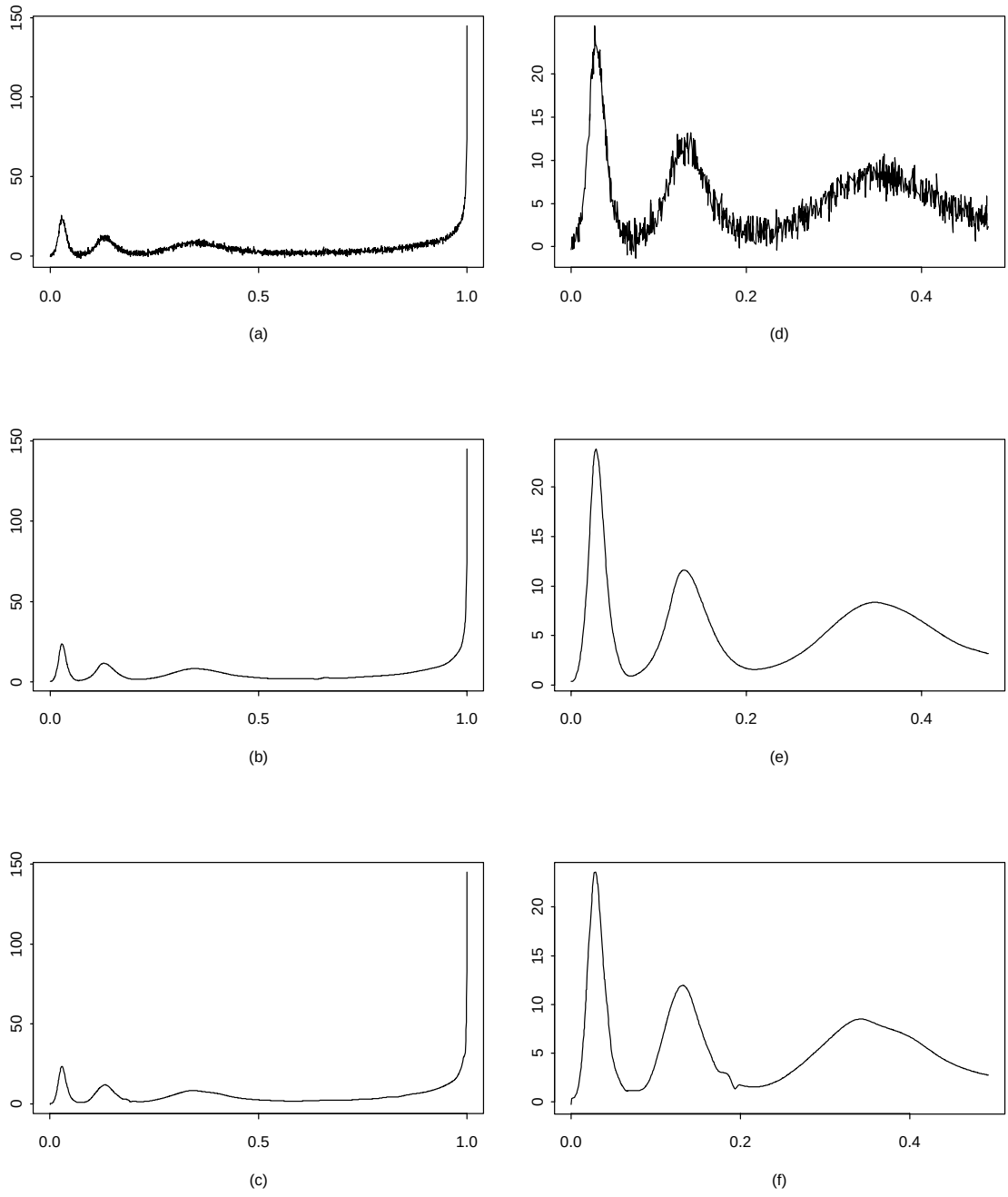


Figure 5.8 Example 2:(a) sample dataset of 2,048 observations from original function with added Uniform ($\sigma = 1.0$) noise; (b) AGS fit with median MSE ($\sigma = .0411$); (c) HAS fit with median MSE ($\sigma = .0569$); (d) sample dataset on [0,0.5]; (e) AGS fit with median MSE on [0,0.5]; (f) HAS fit with median MSE on [0,0.5]

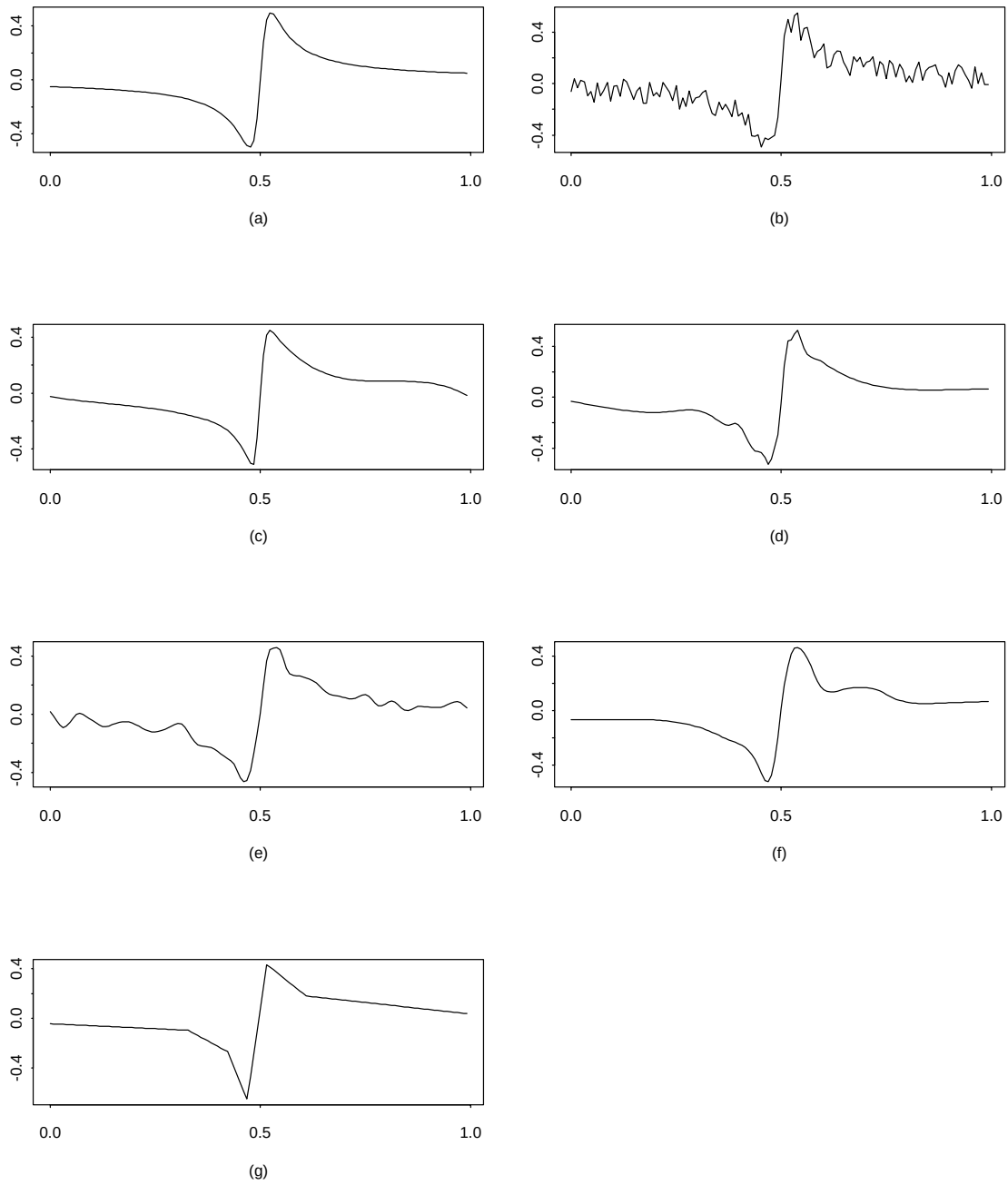


Figure 5.9 Example 3:(a) original function; (b) sample dataset of 400 observations from original function with added Gaussian ($\sigma = 0.06$) noise; (c) AGS fit with median MSE (σ = .0004); (d) HAS fit with median MSE (σ = .0006); (e) SUREShrink fit with median MSE (σ = .0013); (f) MARS fit with median MSE (σ = .0009); (g) POLYMARS fit with median MSE (σ = .0016)

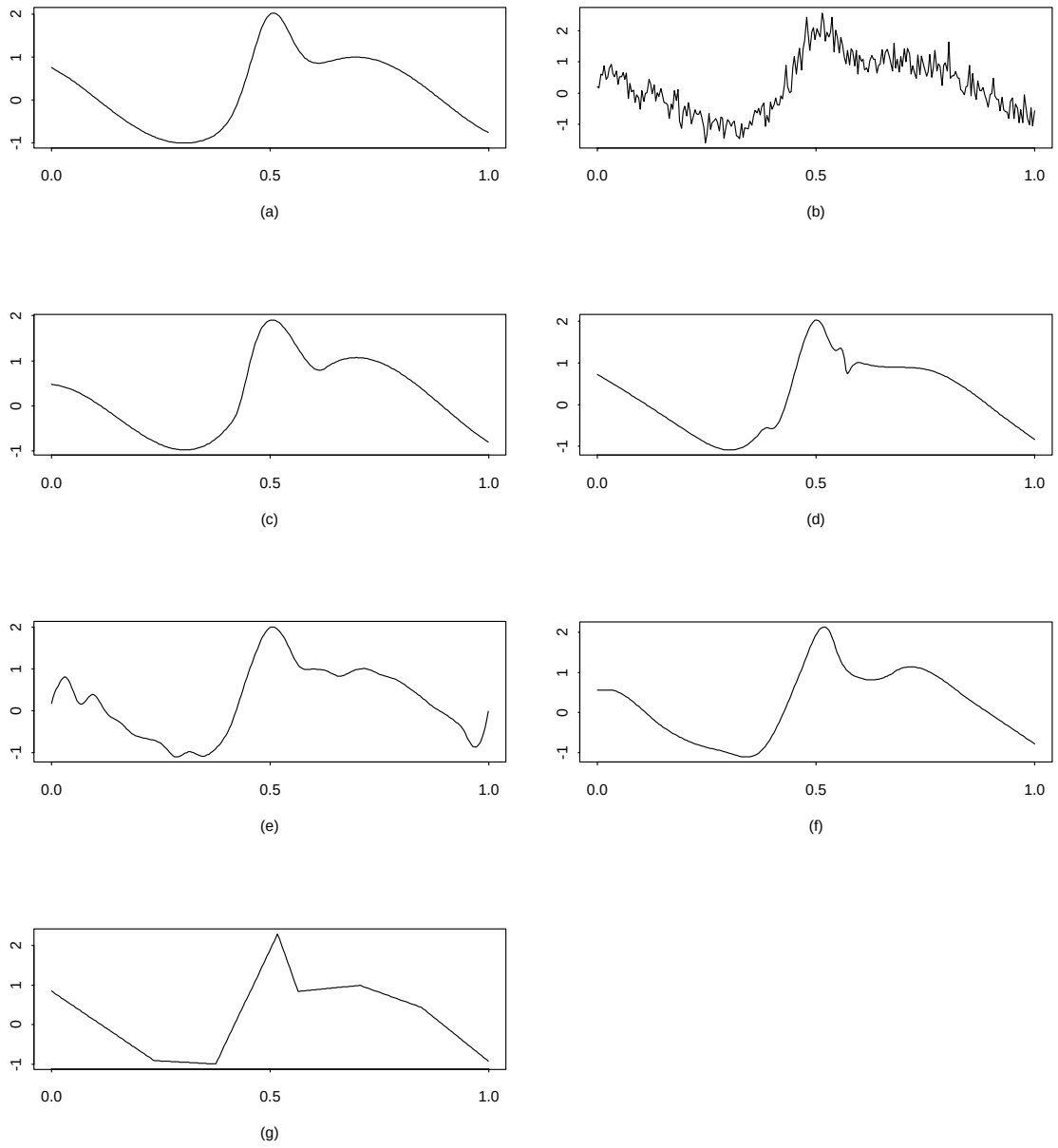


Figure 5.10 Example 4:(a) original function; (b) sample dataset of 400 observations from original function with added Gaussian ($\sigma = 0.3$) noise; (c) AGS fit with median MSE ($\sigma = .0052$); (d) HAS fit with median MSE ($\sigma = .0071$); (e) SUREShrink fit with median MSE ($\sigma = .0178$); (f) MARS fit with median MSE ($\sigma = .0070$); (g) POLYMARS fit with median MSE ($\sigma = .0087$)

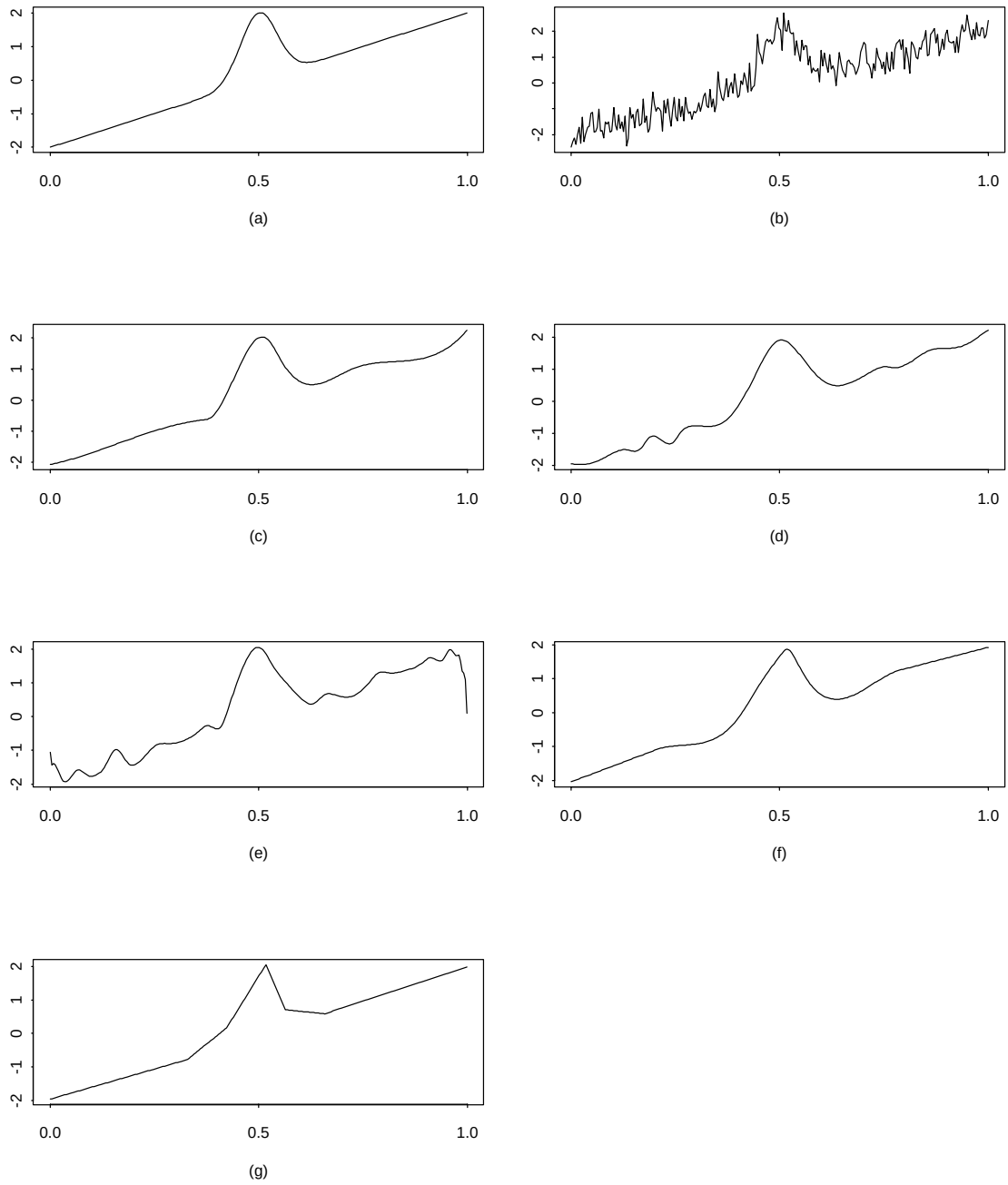


Figure 5.11 Example 5:(a) original function; (b) sample dataset of 400 observations from original function with added Gaussian ($\sigma = 0.4$) Gaussian noise; (c) AGS fit with median MSE ($\sigma = .0098$); (d) HAS fit with median MSE ($\sigma = .0128$); (e) SUREShrink fit with median MSE ($\sigma = .0464$); (f) MARS fit with median MSE ($\sigma = .0126$); (g) POLYMARS fit with median MSE ($\sigma = .0104$)

Chapter 6

Conclusion

6.1 Summary

With the existence of inexpensive, high-speed computational power, statisticians are now able to model data using new flexible and powerful techniques. The numerous assumptions that were once necessary for data analysis are no longer required, as computers provide flexibility in the use and development of modeling tools and the choice of direction in which an analysis can proceed. An area of modeling that has benefited from this flexibility is nonparametric density and regression function estimation.

The topic of interest in this work is the use of spline spaces and genetic algorithms for nonparametric adaptive function estimation and modeling. It is well known that when the number and location of the knots is optimized, the quality of a spline approximation is improved. However, optimizing the number and location of the knots is a historically difficult problem due to the non-convexity of the error surface (due to properties such as lethargy) and the size of the search space. A possible alternative to the traditional nonlinear optimization and stepwise search approaches to solving this problem is to use more intensive numerical optimization techniques such as genetic algorithms (GAs) to optimize the knot sequence. As discussed in Chapter 2, genetic algorithms require few modeling assumptions and have been shown to be a powerful optimization tool in numerous contexts. Unlike

nonlinear optimization and stepwise techniques, GAs have the potential to escape from local minima in the error surface. Given a sufficient number of iterations a GA search will lead to the global optimum in the solution space, i.e., a GA search is not biased in such a way as to prevent it from reaching the best solution.

With linear splines and a fixed number of knots, it was shown in Chapter 3 that using a GA and a criterion based on least squares does lead to an optimal model given enough iterations and a large enough search space. In the following chapter a GA was used for spline modeling where the number of knots as well as their placement were optimized. Due to the consideration of possible outliers in the data, a criterion based on least-squares was discarded in favor of a more robust criterion - robust in the sense that the result of the modeling technique based on this criterion is not affected by outliers. In this context it was shown that with few assumptions the proposed genetic algorithm would find an optimal model given a sufficient number of iterations. In both Chapter 3 and 4 a genetic algorithm was able to find models for experimental datasets that were comparable to the results of several existing spline-fitting algorithms. However, the dimension of the space was not large enough to show the advantage of using a GA search over traditional methods.

In Chapter 5, however, theoretical and experimental results were presented which support the conclusion that intensive numerical optimization of the knot sequence of a spline model, using a technique such as genetic algorithms, does lead to superior models in comparison to traditional methods. Generalizing from the case of linear splines, an algorithm referred to as Adaptive Genetic Splines (AGS) was

described in which a genetic algorithm is used to fit higher order spline models where the number and location of the knots are selected adaptively to optimize a model selection criterion. The model selection criterion in the current implementation of AGS is an adjusted generalized cross-validation statistic, where the adjustment is designed to take into account the amount of fitting that is being performed by a given model. Experimental comparisons with existing methods, including MARS [64], HAS [97], and wavelet techniques [53, 54], demonstrate that AGS models are competitive and capable of outperforming models provided by competing algorithms.

Our empirical findings indicate that in complex parameter spaces, the quality of adaptive spline models can be greatly improved by expending computational resources to optimize the knot sequence. We have shown that a genetic algorithm can be used successfully to find near-optimal adaptive spline models; whether a GA is the best optimization technique for this problem is an open question. However, it is clear that in spline modeling in complex parameter spaces, focusing computational resources on numerical optimization techniques for knot selection yields models which rival and often surpass those of more traditional spline modeling algorithms.

6.2 Future Research

The research presented here is only a beginning - there are many questions that have been left unanswered. Some of these questions have already been mentioned in previous chapters. Only a few of the many remaining questions will be mentioned here.

6.2.1 Higher Dimensions

Many interesting problems involve data sets of more than 2 dimensions; hence we would like to explore the use of GAs for fitting hyperplanes and other multidimensional surfaces. Currently the most feasible approach for extension to higher dimensional data appears to be through additive modeling, as was done by Rogers [123] with the MARS algorithm [64]. We would like to examine whether replacing stepwise techniques with genetic algorithm optimization for knot sequence selection can lead to improved fitting of other types of additive models such as HAS models [97] or the polynomial tensor product spline models of Stone, Hansen, Kooperberg, and Troung [137]. The comparison of a multivariate GA for surface fitting with projection pursuit regression [37, 118] is also worth investigating.

6.2.2 Assessing Model Uncertainty

Since AGS is designed to search the model spaces corresponding to each candidate model size k, $k_{\min} \leq k \leq k_{\max}$, it can report as standard output not only the best overall model g^* but also the minimum RSS model g_k^* for each model size k, i.e., $\{g_{k_{\min}}^*, \ldots, g_{k_{\max}}^*\}$, and the corresponding GCV values. This information gives the user the option of using his/her preferred model selection criterion (or model averaging technique) instead of GCV in choosing the final model. This flexibility of AGS with respect to model choice was considered to be a critical characteristic of the algorithm, given the many selection criteria available and the arguments for and against model selection versus model averaging. We would like to assess the uncertainty in the AGS modeling process; hence a discussion of these arguments is

pertinent. In the discussion presented below we do not discuss classical approaches to assessing model uncertainty in detail (e.g., robustness, nonparametric methods); see Draper [55] or Hoeting, Madigan, Raftery, and Volinsky [75] for information on such techniques.

6.2.2.1 Model Selection

In tackling problems in inference and prediction, in order to achieve the goal of expressing uncertainty about an unknown y in the light of a known x it is common to appeal to a model the formalizes our judgments regarding how y and x are related. In traditional settings it is usually the case that a single best model M^* (i.e., g^*) is chosen from a set of models $\mathcal{M}$ and the analysis then proceeds as if M^* were known to be correct. However, this approach fails to assess structural uncertainty, e.g., uncertainty about the distribution of residuals, the penalty for model size, etc., and hence leads to poor calibration and overconfident predictions [55]. It also does not yield a measure or estimate of the evidence provided by the data for each candidate model.

One of the classical responses to this observation is to suggest the use of a more comprehensive model or a simpler model accompanied by a sensitivity analysis [41]. This is a step in the right direction in terms of assessing uncertainty, but the problems which arise when we recognize selection as a procedure for testing model fit still remain. For example, if we are considering nested models then it is possible to have models consistent with the data but also with departures of interest, or inconsistent but in ways of no interest. If the models are not nested, then both,

either, or neither model can be consistent. These problems might be resolved by assessing the evidence for each candidate model, but the classical approach does not provide a method for doing so. Hence many statisticians prefer an alternate approach, such as a Bayesian approach, when reasonable models lead to different conclusions [120].

The Bayesian approach to model selection is based primarily on the use of Bayes factors [20, 88] to measure the amount of evidence supporting each model, i.e., to measure the move from the prior distribution to the posterior distribution. Bayes factors are constructed by viewing the unknown parameters in the model formulation as variables and the model M as a nuisance parameter. A posterior distribution for the elements of $\mathcal{M}$ is determined by integrating over the model space, i.e., integrating over the model uncertainty. The ratio of posterior probabilities for a model M_1 to a model M_2 is the Bayes factor of M_1 to M_2. Bayes factors try to answer the question of how to balance the number of model terms and the model error with the quality of prediction [88]. The key advantages of Bayes factors are their ability to analyze non-nested models and assess uncertainty; their chief limitations are sensitivity to assumptions and to the choice of prior. For more information and a discussion of Bayes factors (and their variations) and how they compare to p-values, see Berger [20], Kass and Raftery [88], and Raftery [120]. Here we will briefly mention two Bayesian approaches to model selection and assessing model uncertainty: Bayesian model averaging and model expansion.

6.2.2.2 Bayesian Model Averaging

In *Bayesian model averaging* (BMA) a weighted average of the posterior densities for the models in $\mathcal{M}$ is calculated using the posterior model probabilities as weights. BMA can be considered as a type of sensitivity analysis with respect to the structural modeling assumptions and is robust to model choice (although not to the choice of $\mathcal{M}$) [75]. The resulting models are purported to yield better out-of-sample predictions than traditional model selection techniques; the difference between good models is less than the gain that results from BMA [88]. For a discussion of various BMA techniques, see Hoeting et al. [75].

The difficulties with BMA concern the number of terms that must be averaged, the calculation of the integrals involved, and the choice of priors. There are several techniques for controlling the number of terms, e.g., Occam's razor and MC^3 [75]. The best choice of technique is problem-dependent - Occam's is faster but its results versus a full BMA (using all models) is unknown, while MC^3 is slower but yields better predictions [121]. Once the number of terms has been pruned, if necessary, the integrals involved in finding the necessary Bayes factors are often very difficult to compute and hence must be approximated. Kass and Raftery [88] have examined the use of Laplace approximations and recommend the use of such approximations if the distribution of the data is regular and the dimension is modest. The error with a Laplace approximation is at least of order $\mathcal{O}(N^{-1})$ but is usually better than the $\mathcal{O}(1)$ error which one achieves with BIC [75, 129]. BIC is easy to compute and involves no priors so it can be used as a good reference measure of the amount of evidence for the models under consideration. However, the value of

N for which BIC $\equiv$ Bayes factor is unknown and exactly what N should represent is unclear [88, 120]. BIC, like Laplace approximations, can fail if the model is not regular and there are situations, especially when N is small, in which BIC is significantly larger than the conventional level of significance (i.e., p-value). Instead of Laplace or BIC one could use iterative computational techniques to calculate the necessary integrals. Monte Carlo methods have been found to be inefficient [88]; instead, importance sampling is preferred.

The remaining difficulty with implementing BMA is the selection of priors on the parameters of all of the models and on the model space itself. Actually, Bayesians see the prior as part of the model. Unlike in estimation, however, the data do not swamp the prior [55] and rarely can one specify with any precision the nature of the dependence of the results on the priors [41]. In this sense Bayesian model selection is inherently non-robust with respect to the prior assumptions. This can be good if one has information that is useful to the analysis but otherwise can pose quite a challenge. Various authors suggest the use of different types of priors, including reference priors [21], default priors [20], data-based priors [18], elicited priors [88], and equal probabilities on the model subclasses with equal probabilities within each class [116]. Given that a Bayesian analysis is conditional on assumptions which cannot be proved or disproved by the data, the use of posterior probabilities for interpretation and as model evidence are suspect unless a sensitivity analysis has been performed and the results of the analysis have been shown to be insensitive to the choice of prior [20, 41, 116]. Hence model checking is essential.

6.2.2.3 Model Expansion

In light of this conclusion, Draper [55] has suggested *model expansion* as an alternative to model averaging. In model expansion we look for a reasonable starting model M and then expand the model on uncertain structural assumptions, e.g., different error structures. The result is a compromise between using only a single best model M^* and using the entire model set $\mathcal{M}$. By considering the key areas of uncertainty a more realistic choice of a subset of candidate models $\mathcal{M}_1 \subset \mathcal{M}$ can be utilized. We note that model expansion can be thought of as a way to select the subset $\mathcal{M}_1$ over which to perform model averaging; of course the problem of calculating the necessary integrals and selecting priors still remains.

6.2.2.4 Estimation of Variability in AGS

In the Bayesian paradigm model selection is appropriate if there exists a single model M_k for which $p(M_k \mid D) \approx 1$, where D is the available data, or the model average is dominated by models with similar values of $p(\delta \mid D, M_k)$, where δ represents a quantity of interest [88]. Otherwise model averaging or the use of a related technique to assess model uncertainty is emphasized. Using the sequence of best models $\{g^*_{k_{\min}}, \ldots, g^*_{k_{\max}}\}$ for each model size k one could adopt a model averaging approach to AGS; the open question is how to determine the probabilities to use in calculating the weighted average of the models. Buckland, Burnham, and Augustin [32] suggest, among other approaches, the use of AIC [4] to approximate the model weights and several bootstrapping methods to estimate model uncertainty.

The use of AIC with AGS is currently being explored; a discussion of the bootstrap in the context of AGS is given later in this section.

Note that because of the stochastic nature of the genetic algorithm, different applications of AGS to the same dataset will yield different results. We also have the additional variability stemming from the adaptive determination of the appropriate set of basis functions. Our estimate of model accuracy should reflect both of these sources of error.

For a fixed knot sequence, pointwise standard errors (or global confidence sets) could be calculated theoretically. If $g = \hat{\boldsymbol{\alpha}}\mathbf{B}_{\boldsymbol{Z}}$ represents the fitted model, where $\mathbf{B}_{\boldsymbol{Z}}$ is the matrix of B-spline basis functions (the dependence on the chosen knot sequence is explicit in the notation)), $\hat{\boldsymbol{\alpha}}$ is the vector of basis coefficients, and σ^2 represents the variance of $Y_i,\ i = 1, \ldots, N$, then pointwise standard errors can be derived as follows [71]:

If

$$\Sigma = \mathrm{cov}(\hat{\alpha}) = (\mathbf{B}_{\boldsymbol{Z}}^t \mathbf{B}_{\boldsymbol{Z}})^{-1}\sigma^2$$

then

$$\mathrm{cov}(g) = \mathrm{cov}(\hat{\alpha}\mathbf{B}_{\boldsymbol{Z}}) = \mathbf{B}_{\boldsymbol{Z}}\mathrm{cov}(\hat{\alpha})\mathbf{B}_{\boldsymbol{Z}}^t = \mathbf{B}_{\boldsymbol{Z}}\Sigma\mathbf{B}_{\boldsymbol{Z}}^t. \tag{6.1}$$

σ^2 can be estimated from RSS; the diagonal of $\mathrm{cov}(g)$ contains estimates of the pointwise variances so the diagonal elements of $(\mathbf{B}_{\boldsymbol{Z}}\Sigma\mathbf{B}_{\boldsymbol{Z}}^t)^{1/2}$ are estimates of the pointwise standard errors.

One can incorporate the variability due to the adaptive model search into the estimate of σ^2 by adjusting the degrees of freedom, as described in Section 5.3.1. However, the error due to the stochastic nature of the GA has yet to be considered. The nascent state of GA theory suggests that the most feasible way to attain a statistic which also captures this source of error is via simulation, e.g., by bootstrapping or other Monte Carlo methods.

The bootstrap was introduced by Efron [57] as a computational method for determining the accuracy of a parameter estimate. It is particularly useful in situations where assessing accuracy is beyond the existing theory of the estimation method or the problem is too complicated for a traditional statistical analysis. The bootstrap is one way to use computational results in lieu of theoretical analysis to effectively provide a formula for the standard error as a function of the sampling distribution of the data. The basic outline of the bootstrap technique is given below; a similar introduction can be found in Efron and Tibshirani [58].

Let $\mathbf{y} = (y_1, \ldots, y_N)$ be observed values of the random variables $Y_1, \ldots, Y_N \sim$ *i.i.d* F for some probability distribution F. The basic idea behind bootstrap standard errors is to derive $\hat{\sigma}$ from the empirical probability distribution of F. In other words, if $\sigma(F) = [\text{var}_{F,N}(\hat{\theta}(\mathbf{y}))]^{1/2}$, where $\hat{\theta}(\mathbf{y})$ represents our parameter estimate, then $\hat{\sigma} = \sigma(\hat{F})$ where $\hat{F}$ places probability $1/N$ on each observation. $\hat{\sigma}$ is evaluated by a Monte Carlo algorithm:

1. Draw a large number J of bootstrap samples $\{\mathbf{y}_j^b\}_{j=1}^J$ where each is an independent random sample drawn with replacement from the original sample.

2. For each sample $\mathbf{y}_j^b,\ j = 1, \ldots, J$, calculate $\hat{\theta}_j^b = \hat{\theta}(\mathbf{y}_j^b)$.

3. Calculate the sample standard deviation of $\{\hat{\theta^b_j}\}_{j=1}^{J}$ given by

$$\hat{\sigma}^b = \left(\frac{\sum_j (\hat{\theta^b_j} - \bar{\hat{\theta}^b})^2}{J-1} \right)^{1/2} \quad \text{where} \quad \bar{\hat{\theta}^b} = \frac{\sum_j \hat{\theta^b_j}}{J}.$$

Essentially we are evaluating a standard deviation by Monte Carlo sampling. One can of course replace standard error with some other measure of error, such as bias or prediction error, and apply the same method. In this case only step 3 in the above must be modified.

The number of bootstrap samples necessary for calculating a standard error is an open question. In general, a rough minimum sample size is $50 \leq J \leq 200$.

The above discussion suggests the following method for estimating pointwise standard errors for our model:

Let g be the AGS model fit to the original dataset (i.e., g is our estimate $\hat{\theta}$).

1. Fix the algorithm parameters (e.g., $k_{\max}$, p_c, mutation probabilities, maximum number of iterations) at those values used in determining g. Choose a value for J, $50 \leq J \leq 200$.

2. Draw J bootstrap samples $\{\mathbf{y}^b_j\}_{j=1}^{J}$ where each is an independent random sample drawn with replacement from the original sample $\mathbf{y} = (y_1, \ldots, y_N)$.

3. On each sample $\mathbf{y}^b_j$, $j = 1, \ldots, J$, apply the GA method to get an estimate $g^b_j = g^b(\mathbf{y}^b_j)$.

4. For each $g_j^b,\ j = 1, \ldots, J$, calculate the estimate $\hat{\boldsymbol{\sigma}}_j^b$ of the pointwise standard errors. $\hat{\boldsymbol{\sigma}}_j^b$ is defined as the vector whose elements are the square roots of the diagonal elements of $\mathrm{cov}(g_j^b)$ as defined in Equation 6.1, where σ^2 is estimated as

$$\hat{\sigma^2} = \frac{RSS/N}{(1 - (n_1 + (n_2 * d))/N)^2}$$

(see Chapter 5). Here $n = n_1 + n_2$ is the number of basis functions in $\mathbf{B}_{\boldsymbol{Z}_j}$ where $g_j^b = \boldsymbol{\alpha}_j \hat{\mathbf{B}}_{\boldsymbol{Z}_j}$ and d is as defined in Chapter 5.

5. Calculate the sample estimate of the pointwise standard errors, $\hat{\boldsymbol{\sigma}}^b$, given by

$$\hat{\boldsymbol{\sigma}}^b = \frac{\sum_j \hat{\boldsymbol{\sigma}}_j^b}{J}.$$

The above method, albeit heuristic, uses the bootstrap technique combined with an adjusted pointwise standard error calculation to attempt to capture the various sources of model variability.

Note that with our method we must run a GA to get each g_j^b so the computational expense of such accuracy estimates is considerable. Whether this expense is prohibitive can be determined through experimentation.

Appendix A

Proofs of Results from Chapter 4

Proof of Theorem 4.1:

Let $\epsilon \geq \epsilon^*,\ \Theta,$ and $\mathcal{H}$ be given.

Suppose $\exists\ L(\theta_{j^*1}, h_{j^*1}) :\ L(\theta_{j^*1}, h_{j^*1}) \in\ \mathcal{L}^{(\epsilon)}$ and

$$\{(X,Y) : Y = L(\theta_{j^*1}, h_{j^*1})(X),\ X \in [Z_{11}, Z_{21}]\} \bigcap\ rect\ =\ \emptyset.$$

Then either

$$\{(X,Y) : Y \in\ [L(\theta_{j^*1}, h_{j^*1})(X), L(\theta_{j^*1}, h_{j^*1})(X) + \epsilon],\ X \in [Z_{11}, Z_{21}]\} \bigcap\ rect =\ \emptyset$$

or

$$\{(X,Y) : Y \in\ [L(\theta_{j^*1}, h_{j^*1})(X) - \epsilon, L(\theta_{j^*1}, h_{j^*1})(X)],\ X \in [Z_{11}, Z_{21}]\} \bigcap\ rect =\ \emptyset.$$

Assume without loss of generality that

$$\{(X,Y) : Y \in\ [L(\theta_{j^*1}, h_{j^*1})(X) - \epsilon, L(\theta_{j^*1}, h_{j^*1})(X)],\ X \in [Z_{11}, Z_{21}]\} \bigcap\ rect =\ \emptyset.$$

This implies that $\frac{1}{N}\sum_{i=1}^{N}\varphi((X_{i},Y_{i}))\geq 0.95$, where

$$\varphi((X,Y))=\begin{cases}1 & (X,Y)\in\Big\{(X,Y):\ Y\in[L(\theta_{j^{*}1},h_{j^{*}1})(X),L(\theta_{j^{*}1},h_{j^{*}1})(X)+\epsilon],\\ & \qquad\qquad X\in[Z_{11},Z_{21}]\Big\}\\ 0 & \text{otherwise.}\end{cases}$$

Let $(\bar{\bar{X}},\bar{\bar{Y}})=(\sum_{i=1}^{N}\varphi((X_{i},Y_{i})))^{-1}\sum_{j}(X_{j},Y_{j})$ where the sum is taken over all $(X_{j},Y_{j})\in\{(X,Y):\ Y\in[L(\theta_{j^{*}1},h_{j^{*}1})(X),L(\theta_{j^{*}1},h_{j^{*}1})(X)+\epsilon],X\in[Z_{11},Z_{21}]\}$.

Define $L(\theta_{j^{**}1},h_{j^{**}1}):L(\theta_{j^{**}1},h_{j^{**}1})$ is parallel to $L(\theta_{j^{*}1},h_{j^{*}1})$ and

$$\bar{\bar{Y}}=L(\theta_{j^{**}1},h_{j^{**}1})(\bar{\bar{X}}).$$

Then

$$\{(X,Y):Y\in[L(\theta_{j^{*}1},h_{j^{*}1})(X),L(\theta_{j^{*}1},h_{j^{*}1})(X)+\epsilon],\ X\in[Z_{11},Z_{21}]\}\subseteq E_{\epsilon,L(\theta_{j^{**}1},h_{j^{**}1})}.$$

Thus $L(\theta_{j^{**}1},h_{j^{**}1})\in\mathcal{L}^{\epsilon}$ and since $\bar{\bar{Y}}=L(\theta_{j^{**}1},h_{j^{**}1})(\bar{\bar{X}})$,

$$\{(X,Y):Y=L(\theta_{j^{**}1},h_{j^{**}1})(X),\ X\in[Z_{11},Z_{21}]\}\bigcap rect\ \neq\ \emptyset.\ \spadesuit$$

For a graphical representation see Figure A.1a. Note that 95% of the datapoints fall within ϵ of $L(\theta_{j^{*}1},h_{j^{*}1})$ while 95% of the datapoints fall within $\epsilon/2$ of $L(\theta_{j^{**}1},h_{j^{**}1})$.

Proof of Proposition 4.1:

Let $L(\theta_{m1}, h_{m1}) \in \mathcal{Q}_1$ and $\xi > 0$ be given. Choose $\Theta_\xi : \pi/2^{l_a} = \pi/\Theta_\xi < \xi/2$. Similarly, choose $\mathcal{H}_\xi : \ \delta = diag/(2^{l_d} - 1) = diag/(\mathcal{H}_\xi - 1) < \xi/2$. Then $\exists\ L(\theta, h) \in \mathcal{L}_1^r(\Theta_\xi, \mathcal{H}_\xi) : 2.$ is satisfied. Since for any $k,\ k = 1, \dots, k_{\max},\ \{\mathcal{L}_k^r(\Theta, \mathcal{H}) : l_a = 1, 2, \dots\}$ and $\{\mathcal{L}_k^r(\Theta, \mathcal{H}) : l_d = 1, 2, \dots\}$ represent increasing sequences of nested sets, if $L(\theta, h) \in \mathcal{L}_1^r(\Theta_\xi, \mathcal{H}_\xi)$, then $L(\theta, h) \in \mathcal{L}_1^r(\Theta, \mathcal{H})\ \forall\ \Theta > \Theta_\xi$ and $\forall\ \mathcal{H} > \mathcal{H}_\xi$. Hence *1* is satisfied. ♠

Proof of Theorem 4.2:

Let $\xi > 0$ be given. Choose $\Theta_\xi : \pi/2^{l_a} = \pi/\Theta_\xi < \xi/2$. Then for $\theta_{\xi_{(i)}} \in \{\{\frac{\pi}{\Theta_\xi}, \dots, \pi\};\ \theta_{\xi_{(i)}} \le \theta_{\xi_{(i+1)}}\ \forall\ i,\ i = 1, \dots, \Theta_\xi - 1\}$, we have $|0 - \theta_{\xi_{(1)}}| < \xi/2$, $|\ \theta_{\xi_{(1)}} - \theta_{\xi_{(2)}}\ | < \xi/2, \dots, |\ \theta_{\xi_{(\Theta_\xi)}} - \pi\ | < \xi/2$. So given any $L(\theta_{m1}, h_{m1}) \in \mathcal{Q}_1$ we can choose Θ_ξ so that $\exists\ \theta_{\xi_{(i)}} \in \{\frac{\pi}{\Theta_\xi}, \dots, \pi\} : |\ \theta_{m1} - \theta_{\xi_{(i)}}\ | < \xi/2$.

For any angle $\theta_{\xi_{(n)}} \in [\frac{n\pi}{\Theta_\xi}, \frac{(n+1)\pi}{\Theta_\xi}],\ n = 1, \dots, \Theta_\xi - 1$, the corresponding $\rho_{\xi_{(n)}} \in [\gamma_{\xi_{(n_1)}}, \gamma_{\xi_{(n_2)}}],\ l_{\theta_{\xi_{(n)}}} \le\ \gamma_{\xi_{(n_1)}} \le \gamma_{\xi_{(n_2)}} \le diag.$ Find

$$\nu = \max_n \left\{ \sup_{\xi_{(n)}} \{|\ \gamma_{\xi_{(n_2)}} - \gamma_{\xi_{(n_1)}}\ |,\ n = 1, \dots, \Theta_\xi - 1\} \right\}$$

Choose $\mathcal{H}_\xi : \ \nu/\mathcal{H}_\xi < \xi/4$ and $\mathcal{H}_\xi = 2^u$ for some $u \in \mathcal{R}$. Then given any $L(\theta_{m1}, h_{m1}) \in \mathcal{Q}_1$ we can choose $\mathcal{H}_\xi$ so that $\exists\ k_{\xi_{(i)}} \in \{0, \dots, \mathcal{H}_\xi - 1\} : |\ \rho_{\xi_{(i)}} - \rho_{m1}\ | < \xi/2$.

Hence given any $\xi > 0$ and $L(\theta_{m1}, h_{m1}) \in \mathcal{Q}_1$ we can find Θ_ξ and $\mathcal{H}_\xi$ so that $\exists\ L(\theta_{\xi_{(i)}}, h_{\xi_{(i)}}) \in \mathcal{L}_1^r(\Theta_\xi, \mathcal{H}_\xi) :|\ \theta - \theta_{m1}\ | < \xi/2$ and $|\ \rho - \rho_{m1}\ | < \xi/2$.

If $L(\theta_{\xi_{(i)}}, h_{\xi_{(i)}}) \in \mathcal{L}_1^r(\Theta_\xi, \mathcal{H}_\xi)$, then $L(\theta_{\xi_{(i)}}, h_{\xi_{(i)}}) \in \mathcal{L}_1^r(\Theta, \mathcal{H})\ \forall\ \Theta > \Theta_\xi$ and $\forall\ \mathcal{H} > \mathcal{H}_\xi$. Hence *1.* and *2.* are satisfied.♠

Proof of Proposition 4.2:

Define $\mathcal{S}_1 = \bigcup_{i=1}^{\infty} \mathcal{L}_1^r(\Theta_i, \mathcal{H}_i)$. Note $\mathcal{Q}_1 \subseteq \mathcal{S}_1$ and $\mathcal{S}_1 \subseteq \mathcal{T}_1$ so $\mathcal{Q}_1 \subseteq \mathcal{T}_1$. Since we are only considering optimal lines which pass through *rect*, the optimal lines in $\mathcal{Q}_1 \equiv$ the optimal lines in $\mathcal{T}_1$.♠

Proof of Theorem 4.3

Note that $\mathcal{L}_1^r(\Theta_i, \mathcal{H}_i) \subset \mathcal{L}_1^r(\Theta_{\check{\imath}}, \mathcal{H}_{\check{\imath}})\ \forall\ \check{\imath} > i$. Hence $\mathcal{A}_{\epsilon i} \subseteq \mathcal{A}_{\epsilon \check{\imath}}\ \forall\ \check{\imath} > i$. Thus $\mathcal{A}_{\epsilon \lim} = \lim_{i \to \infty} \mathcal{A}_{\epsilon i}$ exists [8]. But $\mathcal{A}_{\epsilon \lim}$ is the set of optimal lines in $\mathcal{S}_1$ where $\mathcal{Q}_1 \subseteq \mathcal{S}_1 \subseteq \mathcal{T}_1$. By Proposition 4.2, $\mathcal{A}_{\epsilon \lim} = \mathcal{A}_\epsilon$. ♠

Proof of Theorem 4.4

Suppose $\exists\ L(\theta_{j^*1}, h_{j^*1}),\ L(\theta_{j^{**}1}, h_{j^{**}1}) \in \mathcal{A}_{\epsilon^*_{1.0}}(1.0)$:
$L(\theta_{j^*1}, h_{j^*1}) \neq L(\theta_{j^{**}1}, h_{j^{**}1})$. Then for all possible values of (X, Y),

$$(X,Y) \in \{(X,Y) : Y \in [L(\theta_{j^*1}, h_{j^*1})(X) - \epsilon^*_{1.0},\ L(\theta_{j^*1}, h_{j^*1})(X) + \epsilon^*_{1.0}]\}$$

and

$$(X,Y) \in \{(X,Y) : Y \in [L(\theta_{j^{**}1}, h_{j^{**}1})(X) - \epsilon^*_{1.0},\ L(\theta_{j^{**}1}, h_{j^{**}1})(X) + \epsilon^*_{1.0}]\}\ .$$

Recall that for $k^* = 1$, the support of $\alpha_1(X, Y)$ was defined as B where

$$B = \{(X,Y) : Y \in [\tfrac{d_1 - X\cos\theta_1}{\sin\theta_1} - \epsilon_0, \tfrac{d_1 - X\cos\theta_1}{\sin\theta_1} + \epsilon_0]\} \quad \text{for} \quad X \in [Z_{11}, Z_{21}].$$

As $N \to \infty$, $\epsilon^*_{1.0} \to \epsilon_0$ since $\epsilon^*_{1.0}$ is minimal and $fit_{\epsilon,N}$ is continuous for all ϵ. Hence $\{(X,Y) : Y = L(\theta_{j^*1}, h_{j^*1})(X)\} \to \{(X,Y) : X\cos\theta_1 + Y\sin\theta_1 = d_1\}$ and

$\{(X,Y) : Y = L(\theta_{j^{**}1}, h_{j^{**}1})(X)\} \rightarrow \{(X,Y) : X\cos\theta_1 + Y\sin\theta_1 = d_1\}$. Thus $L(\theta_{j^*1}, h_{j^*1}) = L(\theta_{j^{**}1}, h_{j^{**}1})$. ♠

See Figure A.1b for a graphical representation of Theorem 4.4.

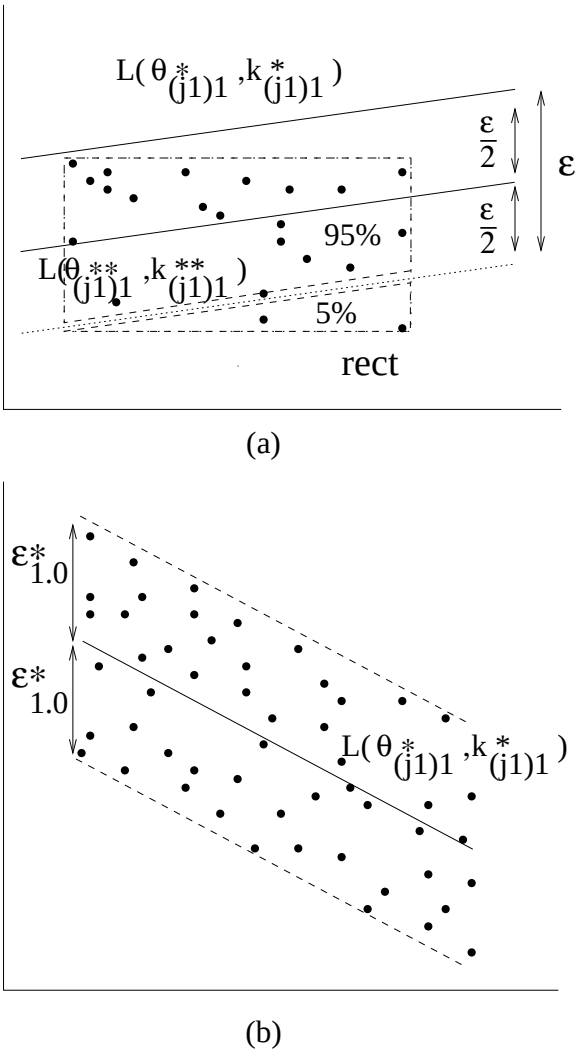


Figure A.1 Graphical representations of (a) Theorem 4.2 and (b) Theorem 4.4

Proof of Theorem 4.5:

Given the continuity of $fit_{\epsilon,N}$ we know that as $N \to \infty$, for each N large we may find $\epsilon_{N_1}, \epsilon_{N_2}, \epsilon_{N_3},\ 0 < \epsilon_{N_1} < \epsilon_{N_2} < \epsilon_{N_3}$, such that

- $\overline{fit}_{\epsilon_{N_1},N}(L(\theta_{j^*1}, h_{j^*1})_{\epsilon_{N_1},N}) < 0.95$
- $\overline{fit}_{\epsilon_{N_2},N}(L(\theta_{j^*1}, h_{j^*1})_{\epsilon_{N_2},N}) \geq 0.95$
- $\overline{fit}_{\epsilon_{N_3},N}(L(\theta_{j^*1}, h_{j^*1})_{\epsilon_{N_3},N}) \geq 0.97$

Note that $L(\theta_{j^*1}, h_{j^*1})_{\epsilon_{N_1},N} \notin \mathcal{L}^{(\epsilon_{N_1})}$ while ϵ_{N_3} is not minimal, i.e., there exists some $\epsilon < \epsilon_{N_3} : \mathcal{L}^{(\epsilon)} \neq \emptyset$. Hence for our stated goal (see Equation 4.4) we are interested only in ϵ_{N_2}. For each N there may be infinitely many such ϵ_{N_2}. For a given N and ϵ let one such ϵ_{N_2} be $\epsilon^*_{N_2}$. Then we conclude

$$\liminf_{N\to\infty} \overline{fit}_{\epsilon_{N_2},N}(L(\theta_{j^*1}, h_{j^*1})_{\epsilon_{N_2},N}) \geq 0.95$$

for appropriate Θ and $\mathcal{H}$. ♠

Proof of Theorem 4.6:

Let $fit^*_{\epsilon,N,k} = \max_{L^k(\boldsymbol{\theta}_{j\cdot}, \boldsymbol{h}_{j\cdot}) \in \mathcal{L}^r_k(\Theta,\mathcal{H})} fit_{\epsilon,N}(L^k(\boldsymbol{\theta}_{j\cdot}, \boldsymbol{h}_{j\cdot}))$, $k = 1, \ldots, k_{\max}$. Suppose $fit^*_{\epsilon,N} \neq fit^{**}_{\epsilon,N}$. Then either

1. $fit^{**}_{\epsilon,N} \neq fit^*_{\epsilon,N,k^*}$

   or

2. $fit^{**}_{\epsilon,N} = fit^*_{\epsilon,N,k^*}$ but $fit^*_{\epsilon,N,k^*} \neq fit^*_{\epsilon,N}$.

If $fit^{**}_{\epsilon,N} \neq fit^*_{\epsilon,N,k^*}$ then

$$\exists\ L^{k^*}(\boldsymbol{\theta}_{j^{**}\cdot}, \boldsymbol{h}_{j^{**}\cdot}) \in \mathcal{L}^r_{k^*}(\Theta,\mathcal{H}) :\ fit_{\epsilon,N}(L^{k^*}(\boldsymbol{\theta}_{j^{**}\cdot}, \boldsymbol{h}_{j^{**}\cdot})) > fit_{\epsilon,N}(L^{k^*}(\boldsymbol{\theta}_{j^*\cdot}, \boldsymbol{h}_{j^*\cdot})).$$

But $L^{k^*}(\boldsymbol{\theta}_{j^{**}.}, \boldsymbol{h}_{j^{**}.}) \in \mathcal{L}_{k^*} \subset \mathcal{L}_{\mathcal{K}}$ and

$$fit^{**}_{\epsilon,N} = fit_{\epsilon,N}(L^{k^*}(\boldsymbol{\theta}_{j^*.}, \boldsymbol{h}_{j^*.})) = \max_{L^k(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.}) \in \mathcal{L}_{\mathcal{K}}} fit_{\epsilon,N}(L^k(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.})).$$

Hence $fit_{\epsilon,N}(L^{k^*}(\boldsymbol{\theta}_{j^*.}, \boldsymbol{h}_{j^*.})) \geq fit_{\epsilon,N}(L^{k^*}(\boldsymbol{\theta}_{j^{**}.}, \boldsymbol{h}_{j^{**}.})) \Rightarrow fit^{**}_{\epsilon,N} = fit^{*}_{\epsilon,N,k^*}$.

However, suppose $fit^{**}_{\epsilon,N} = fit^{*}_{\epsilon,N,k^*}$ but $fit^{*}_{\epsilon,N,k^*} \neq fit^{*}_{\epsilon,N}$. If $fit^{*}_{\epsilon,N,k^*} \neq fit^{*}_{\epsilon,N}$ then

$\exists k_0 \in \mathcal{K} : fit^{*}_{\epsilon,N,k_0} > fit^{*}_{\epsilon,N,k^*}$, i.e.,

$$fit^{*}_{\epsilon,N,k^*} \neq \max_{L^k(\boldsymbol{\theta}_{j^*.}, \boldsymbol{h}_{j^*,.}) \in L_{best}} fit_{\epsilon,N}(L^k(\boldsymbol{\theta}_{j^*.}, \boldsymbol{h}_{j^*.})).$$

But $\mathcal{L}_{k_0}(\Theta, \mathcal{H}) \subset \mathcal{L}_{\mathcal{K}} \Rightarrow L^{k_0}(\boldsymbol{\theta}_{j^*.}, \boldsymbol{h}_{j^*.}) \in \mathcal{L}_{\mathcal{K}}$ while

$$fit^{**}_{\epsilon,N} = fit^{*}_{\epsilon,N,k^*} = \max_{L^k(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.}) \in \mathcal{L}_{\mathcal{K}}} fit_{\epsilon,N}(L^k(\boldsymbol{\theta}_{j.}, \boldsymbol{h}_{j.})).$$

Hence $fit^{*}_{\epsilon,N,k^*} > fit^{*}_{\epsilon,N,k_0} \Rightarrow fit^{*}_{\epsilon,N,k^*} = fit^{*}_{\epsilon,N}$. Thus $fit^{*}_{\epsilon,N} = fit^{**}_{\epsilon,N}$. ♠

Appendix B

Neural Networks and Fractals Research

In a search for techniques which may improve the performance of genetic algorithm optimization, our research has lead to interest in the properties of other optimization techniques in areas of artificial intelligence, e.g., neural networks. As previously noted, neural networks have been used successfully in nonparametric statistical modeling - see, for example, Lee [92]. Before studying NNs as a modeling tool, we have decided to focus our research on neural networks (more specifically, multilayer perceptrons (MLPs)) from a mathematical perspective by examining the relationship between the gradient descent technique and contractive maps.

B.1 Summary

The relationship of interest, i.e., between the gradient descent technique and contractive maps, is based upon the observation that the convergence of the gradient descent technique (which is used by MLPs in optimization problems) can be proved using results in fractal theory - more specifically, results concerning contractive maps - as opposed to results based on Taylor series. This proof, involving the eigenvalues of the Hessian matrix of the gradient descent technique's objective function, is presented here. A simple example is given in which steps from the aforementioned proof are used to find conditions under which a specific multilayer perceptron is guaranteed to converge. Since the gradient descent technique is used in multilayer perceptrons, and contractive maps give rise to fractals, a theoretical relationship is thus established between multilayer perceptrons

and fractals. It is possible that this connection will lead by extention to improvements in optimization techniques that are based on iterative methods, e.g., neural networks and genetic algorithms.

B.2 Introduction

Multilayer Perceptrons (MLPs) have been used in numerous applications, many of which have involved either classification of given observations or approximation of the observations' generating function [6]. To arrive at a suitable solution for such problems, MLPs use both back propagation of error [126] and the gradient descent technique to optimize an objective function.

It has been postulated whether the optimization process of MLPs and the generation of fractals are related. Fractals are self-similar mathematical objects which are usually generated by contractive maps [15]. A group of contractive maps, applied repeatedly to any collection of nonempty compact sets, will ultimately force these sets to converge to a single set, called the attractor. The attractor is considered fractal.

The proof of convergence of the gradient descent technique to a local optimum is usually given in terms of results on Taylor series [26]. However, an examination of the recent literature on fractals and contractive maps raised questions regarding whether this proof could be restated in terms of fractal theory, thus establishing a relationship between MLPs and fractals.

In the following discussion we establish such a relationship by using characteristics of fractal convergence to rewrite the proof of convergence of the gradient descent technique. This entails stating conditions under which the function representing the gradient descent process is a contractive map. Section B.3 provides some basic results on fractals which are

relevant from the point of view of MLPs. Section B.4 provides the relationship between contractive maps and the gradient descent technique. Section B.5 describes the connection between contractive maps and MLPs. Section B.6 presents some experimental results, and Section B.7 contains a final discussion. This work should encourage interaction between researchers in neural networks and those in fractal theory, hopefully leading to positive developments in both fields.

B.3 Mathematical Preliminaries

We shall describe below some of the basic results concerning fractals. The theory behind fractal generation is based on contractive maps; hence we will initially discuss such maps. Although the following results regarding contractive maps hold for any complete metric space, we will state them with respect to d-dimensional Euclidean space. Most of the results stated here can be found in Barnsley [15].

Let $\mathcal{R}$ denote the real line. Let $\mathcal{R}^d$ denote the d-dimensional Euclidean plane and let $\mathcal{A}$ denote the set of all non-empty compact subsets of $\mathcal{R}^d$. $\rho(x, y)$ will denote the Euclidean distance between two points x and y in $\mathcal{R}^d$.

DEFINITION B.1 A function f defined from $\mathcal{R}^d$ to $\mathcal{R}^d$ is said to be *contractive* if there exists $s,\ 0 \leq s < 1,$ such that

$$\rho(f(x), f(y)) \leq s \cdot \rho(x, y) \quad \forall \quad x, y \in \mathcal{R}^N.$$

s is said to be a *contractivity factor* of f. ♠

A contractive function (map) f will shrink any nonempty compact subset of $\mathcal{R}^d$. If there does not exist any $s < 1$ for which the above holds then f is not contractive.

RESULT B.1 Let f be a contractive map from $\mathcal{R}^d$ to $\mathcal{R}^d$. Then there exists a unique $x_0 \in \mathcal{R}^d$ such that

1. $f(x_0) = x_0$

   and

2. $\lim_{n\to\infty} f^n(x) = x_0 \ \ \forall \ x \in \mathcal{R}^d$ where

   - $f^1(x) = f(x)$
   - $f^n(x) = f^{n-1}(f^1(x)) \quad \forall \ n > 1$ and $\forall \ x$. ♠

x_0 is said to be the *fixed point* of f. Repeated applications of f on any nonempty compact subset of $\mathcal{R}^d$ will make it go towards a set containing only x_0.

We shall extend Result B.1 to sets by using a metric between sets, namely, the *Hausdorff metric*.

DEFINITION B.2 Let $\rho(x, A) = \inf_{y\in A} \rho(x, y)$. The *Hausdorff distance* between two sets A and B in $\mathcal{A}$ is defined as

$$D(A, B) = \max\ [\sup_{x\in A} \rho(x, B),\ \sup_{y\in B} \rho(y, A)].\ ♠$$

It can be easily shown that the above D is a metric in $\mathcal{A}$.

DEFINITION B.3 Let f be a function from $\mathcal{A}$ to itself. Then f is said to be *contractive* if there exists $s,\ 0 \le s < 1,$ such that

$$D(f(x), f(y)) \ \le \ s \cdot D(x, y) \quad \forall \ \ x, y \ \in \mathcal{A}.$$

s is said to be a *contractivity factor* of f. ♠

RESULT B.2 Let f be a contractive map from $\mathcal{A}$ to $\mathcal{A}$. Then there exists a unique $x_0 \in \mathcal{A}$ such that

1. $f(x_0) = x_0$
2. $\lim_{n\to\infty} f^n(x) = x_0 \ \forall \ x \in \mathcal{A}$ where
   - $f^1(x) = f(x)$
   - $f^n(x) = f^{n-1}(f^1(x)) \quad \forall \ n > 1$ and $\forall\, x \in \mathcal{A}$.

Here the metric under consideration is the Hausdorff metric D. ♠

RESULT B.3 Let $f_1, f_2, \cdots, f_M$ be M contractive maps defined from $\mathcal{A}$ to itself Let $s_1, s_2, \cdots, s_M$ be their respective contractivity factors. For any $C \subseteq \mathcal{A}$, let

$$F_{n+1}(C) = \bigcup_i f_i(F_n(C)) \ \forall \ n \geq 0 \quad \text{where} \quad F_0(C) = C \ \forall \ C \subseteq \mathcal{A}.$$

Then there exists $A \subseteq \mathcal{A}$ such that

1. $F_1(A) = A$
2. $\lim_{n\to\infty} F_n(C) = A \ \forall \ C \ \subseteq \mathcal{A}$.

$(\mathcal{A} : f_1, f_2, \cdots, f_M)$ is said to be an *iterated function system* and A is said to be the *attractor* of the iterated function system. ♠

The word *fractal* is defined by various authors in various ways. Barnsley considers a fractal to be a set in $\mathcal{A}$; we will also consider fractals as such. Fractals are usually generated by iterated function systems.

A definition and preliminary result from matrix algebra are stated below. These will be used in developing the relationship between the gradient descent technique and contractive maps.

DEFINITION B.4 For a real matrix $\mathbf{B}$ of order $d \times d$, the *norm* of $\mathbf{B}$, denoted by $||\mathbf{B}||$, is defined as

$$||\mathbf{B}|| = \sup_{(\boldsymbol{x}:||\boldsymbol{x}|| \neq 0)} ||\mathbf{B}\boldsymbol{x}||/||\boldsymbol{x}||.$$

where $||\boldsymbol{x}|| = \sqrt{x_1^2 + \ldots + x_d^2}$ if $\boldsymbol{x} = (x_1, \ldots, x_d)$.

RESULT B.4 If $\mathbf{B}$ is a real symmetric positive definite matrix of order $d \times d$ then its norm is $\max\{\alpha_1, \alpha_2, \cdots, \alpha_d\}$ where $\{\alpha_i\}_{i=1}^d$ are the eigenvalues of the matrix $\mathbf{B}$.

Note : If $\mathbf{B}$ is a real symmetric matrix then the norm of $\mathbf{B}$ is the maximum of the modulus of the eigenvalues of $\mathbf{B}$.

This completes the mathematical preliminaries. We shall now discuss the relationship between contractive maps and the gradient descent technique.

B.4 Gradient Descent Technique and Contractive Maps

Gradient descent is a technique used to find the minimum of a given function. It is commonly used in neural network applications to find the minimum of the given objective function [26]. We describe the gradient descent technique below.

B.4.1 Gradient Descent Technique

Let g be a continuous, twice differentiable function from $\mathcal{R}^d$ to $\mathcal{R}$. Let $\partial g(\boldsymbol{y})/\partial x_i$ denote the partial derivative with respect to x_i of the function g at the point $\boldsymbol{y}$ in $\mathcal{R}^d$ where $i = 1, 2, \cdots, d$.

Let

$$\nabla g(\boldsymbol{Y}) = (\partial g(\boldsymbol{Y})/\partial X_1, \ \partial g(\boldsymbol{Y})/\partial X_2, \cdots, \ \partial g(\boldsymbol{Y})/\partial X_d)^t$$

where t denotes the transpose. $\nabla g(\boldsymbol{Y})$ is the *gradient* of $g(\boldsymbol{Y})$.

Let f, a function from $\mathcal{R}^d$ to itself, be such that

$$f(\boldsymbol{Y}) = \boldsymbol{Y} - \eta \nabla g(\boldsymbol{Y}) \tag{B.1}$$

where $\eta > 0$ is a constant. Equation B.1 represents the process of the gradient descent technique. In other words, let

$$f^1(\boldsymbol{Y}) = f(\boldsymbol{Y}) \text{ and } f^n(\boldsymbol{Y}) = f^{n-1}(f^1(\boldsymbol{Y})) \text{ for all } n > 1 \text{ and for all } \boldsymbol{Y} \in \mathcal{R}^d.$$

The limit of $f^n(\boldsymbol{Y})$ as n goes to infinity is taken to be the solution for the minimization of g. ♠

In terms of neural networks, $g(\boldsymbol{Y})$ is usually the error function, i.e., $\sum(obs - exp)^2$, viewed as a function of the network weights. The objective is to find the choice of weights which minimizes $g(\boldsymbol{Y})$. Note, however, that the result of the gradient descent technique depends on the choice of η and the initial weight vector.

We shall find the relationship between the gradient descent technique and the contractive maps below. Initially, we shall assume that $d = 1$ and later the results will be generalized to any d.

Let $d = 1$. Then $g : \mathcal{R} \rightarrow \mathcal{R}$ and Equation B.1 can be written as

$$f(Y) = Y - \eta \frac{dg(Y)}{dY} \tag{B.2}$$

THEOREM B.1 Let g be a twice differentiable function with x_0 as a local minimum. Let $h(X) = d^2 g(X)/dX^2$ be a continuous function where $h(X) > 0$ at $X = x_0$. Let ν_2 be an open interval containing x_0 such that $dg(X)/dX \neq 0$ for all $X \neq x_0$ and $X \in \nu_2$.

Then there exists a closed interval ν around x_0 such that the function f, defined in Equation $B.2$, is a contractive map on ν and its fixed point in ν is x_0.

Proof : Note that h is continuous and $h(x_0) > 0$. Then there exists an open interval ν_1 around x_0 such that $h(X) > 0$ for all $X \in \nu_1$. Let Y_1, Y_2 be in ν_1 such that $Y_1 \neq Y_2$. Now

$$\begin{aligned}|f(Y_1) - f(Y_2)| &= |Y_1 - Y_2 - \eta(dg(Y_1)/dY - dg(Y_2)/dY)| \\ &= |Y_1 - Y_2 - \eta((dg(Y_1)/dY - dg(Y_2)/dY)/(Y_1 - Y_2))(Y_1 - Y_2)| \\ &= |Y_1 - Y_2||1 - \eta h(Y_3)|.\end{aligned}$$

(Here Y_3 is a convex combination of Y_1 and Y_2. This step follows from the Mean Value Theorem [7].)

Let ν be a closed interval such that (1) ν is contained in ν_1 and ν_2, (2) ν contains x_0, and (3) ν is bounded. Note that h is bounded on ν.

Let $a = \inf_{X \in \nu}(1/h(X))$ and

$$0 < \eta < a. \tag{B.3}$$

Then note that $0 < |1 - \eta h(Y)| < 1$ for all y in ν. Then

$$|f(Y_1) - f(Y_2)| = |Y_1 - Y_2||1 - \eta h(Y_3)| < |Y_1 - Y_2|.$$

Thus f is a contractive map on ν. Its fixed point is the point Y in ν such that $f(Y) = Y$. Now $f(Y) = Y$

$$\Leftrightarrow Y - \eta dg(Y)/dY = Y$$

$\Leftrightarrow \eta dg(Y)/dY = 0$

$\Leftrightarrow dg(Y)/dY = 0$ (since $\eta > 0$).

Note that the only point in ν for which $dg(Y)/dY = 0$ is x_0 since ν has been chosen in that way. ♠

Generally, the proof for the gradient descent technique is derived using the results on Taylor series. The above proof, however, is based on the theory of contractive maps.

Remarks

1. The above theorem indicates that f is a contractive map in the compact interval ν and if f is iterated in ν, it will eventually produce the local minimum x_0. The bound for η may also be noted in this regard. If η does not satisfy the Equation B.3, then the function f may take values outside ν for some Xs in ν. Also, the selection of η is problematic if we want to get the global optimum.

2. The results hold in a certain interval containing the local optimum and the initial point for the iteration needs to be taken in that interval in order to get that particular local optimum. Thus, if we have more than one local optimum and the initial point is taken in the respective interval of one of these local optima then, with the proper choice of η, the technique will converge to that optimum.

3. If the initial point is outside of the respective intervals of all local optima or η is not chosen properly then f may not be contractive and hence the technique may not converge. In other words, the process may diverge to $+\infty$ or -∞ or it may oscillate between $+\infty$ and -∞.

4. Let the function g posess k local optima $x_1, x_2, \ldots, x_k$ for which $h(x_i) > 0 \; \forall \; i = 1, \ldots, k$. Let ν_i be a closed disk around x_i such that $dg(Y)/dY \neq 0$ for all $Y \neq x_i$ and $Y \in \nu_i$. Let η_i be a choice of η which makes f contractive for $i = 1, \ldots, k$. Let $\lambda = \min\{\eta_1, \ldots, \eta_k\}$ and let

$$f(Y) = Y - \lambda \frac{dg(Y)}{dY} \tag{B.4}$$

Then, if Y is an element of $\bigcup_i \nu_i$, the limit of $f^n Y$ as n goes to infinity belongs to the set A where $A = \{x_1, \ldots, x_k\}$. In other words, if κ is a nonempty compact subset of A, then

$$\lim_{n \to \infty} f^n(\kappa) \; \in \; A. \; \spadesuit$$

Since neural network applications involve data of higher dimensions, the generalization to the d dimensional case is stated below.

THEOREM B.2 Let g be a function from $\mathcal{R}^d$ to $\mathcal{R}$ with a local minimum at $\boldsymbol{x}_0$. Let $b_{ij} = \partial^2 g / \partial X_i \partial X_j$ such that b_{ij} is continuous for each (ij) pair and b_{ij} $(\boldsymbol{Y})$ denotes the value of b_{ij} calculated at $\boldsymbol{Y}$. Let $\mathbf{H}$ be the Hessian matrix of g. Let ν_2 be an open disk containing $\boldsymbol{x}_0$ such that $\nabla g(\boldsymbol{X}) \neq \mathbf{0}$ for all $\boldsymbol{X} \neq \boldsymbol{x}_0$ and $\boldsymbol{X} \in \nu_2$. Let ν_3 be an open disc containing $\boldsymbol{x}_0$ such that $\mathbf{H}(\boldsymbol{c})$ is positive definite for all $\boldsymbol{c} \in \nu_3$. Then there exists a closed disc ν around $\boldsymbol{x}_0$ and a constant $\eta > 0$ such that the function f, defined in Equation $B.1$, is a contractive map on ν and its fixed point in ν is $\boldsymbol{x}_0$.

Proof : Note that for each $\boldsymbol{c} \in \nu_3$, the eigenvalues of $\mathbf{H}(\boldsymbol{c})$ are all greater than zero since $\mathbf{H}(\boldsymbol{c})$ is positive definite. Consider a closed disc ν around $\boldsymbol{x}_0$ such that $\nabla g(\boldsymbol{X}) \neq \mathbf{0}$

for all $\boldsymbol{X} \neq \boldsymbol{x}_0$, $\boldsymbol{X} \in \nu$ and $(\mathbf{H}\boldsymbol{c})$ is positive definite for all $\boldsymbol{c} \in \mathbf{H}$. Note that $\mathbf{H}$ is a subset of $\mathbf{H}_2 \bigcup \mathbf{H}_3$ and such a $\mathbf{H}$ exists.

Consider the equation

$$f(\boldsymbol{Y}) = \boldsymbol{Y} - \eta \nabla g(\boldsymbol{Y}.) \tag{B.5}$$

Let $\boldsymbol{a} \neq \boldsymbol{b}$ be two vectors in ν. Now

$$\begin{aligned} ||f(\boldsymbol{a}) - f(\boldsymbol{b})|| &= ||\boldsymbol{a} - \boldsymbol{b} - \eta(\nabla g(\boldsymbol{a}) - \nabla g(\boldsymbol{b}))|| \\ &= ||\boldsymbol{a} - \boldsymbol{b}|| * ||((\boldsymbol{a} - \boldsymbol{b})/(||\boldsymbol{a} - \boldsymbol{b}||)) - \eta((\nabla g(\boldsymbol{a}) - \nabla g(\boldsymbol{b}))/(||\boldsymbol{a} - \boldsymbol{b}||))|| \\ &= ||\boldsymbol{a} - \boldsymbol{b}|| * ||((\boldsymbol{a} - \boldsymbol{b})/(||\boldsymbol{a} - \boldsymbol{b}||)) - \eta\mathbf{H}(\boldsymbol{c})((\boldsymbol{a} - \boldsymbol{b})/(||\boldsymbol{a} - \boldsymbol{b}||))||. \end{aligned}$$

(Here $\boldsymbol{c}$ is a convex combination of $\boldsymbol{a}$ and $\boldsymbol{b}$, and $\boldsymbol{c}$ also lies in the set ν. The existence of such a $\boldsymbol{c}$ is guaranteed from the literature [103].)

Let $\mathbf{I}$ represent the d dimensional identity matrix. Then

$$\begin{aligned} ||f(\boldsymbol{a}) - f(\boldsymbol{b})|| &= ||\boldsymbol{a} - \boldsymbol{b}|| * ||(\mathbf{I} - \eta\mathbf{H}(\boldsymbol{c}))((\boldsymbol{a} - \boldsymbol{b})/(||\boldsymbol{a} - \boldsymbol{b}||))|| \\ &\leq ||\boldsymbol{a} - \boldsymbol{b}|| * ||\mathbf{I} - \eta\mathbf{H}(\boldsymbol{c})||. \end{aligned}$$

Let $\alpha(\boldsymbol{c}) = ||\mathbf{I} - \eta\mathbf{H}(\boldsymbol{c})||$. If η is selected in such a way that $\alpha < 1$ for each $\boldsymbol{c}$ in ν, then f would be a contractive map and thus its fixed point would be $\boldsymbol{x}_0$.

Let $B(\boldsymbol{c}) = \mathbf{I} - \eta\mathbf{H}(\boldsymbol{c})$. We observe that $B(\boldsymbol{c})$ is symmetric so the norm of $B(\boldsymbol{c})$ is the maximum of the modulus of its eigenvalues (from Section 2). Hence if $\mu_1(\boldsymbol{c})$, $\mu_2(\boldsymbol{c})$, $\cdots$, $\mu_N(\boldsymbol{c})$ are the eigenvalues of $\mathbf{H}(\boldsymbol{c})$ then the eigenvalues of $B(\boldsymbol{c})$ are $1 - \eta\mu_i(\boldsymbol{c})$ for $i = 1, 2, \cdots, N$. Note that (1) $\mu_i(\boldsymbol{c}) > 0 \; \forall \; i = 1, 2, \cdots, N$ and $\forall \; \boldsymbol{c} \in \nu$ since $\mathbf{H}(\boldsymbol{c})$ is positive definite, and (2) $\mu_i(\boldsymbol{c})$ is bounded for all $i = 1, 2, \cdots, N$ and for all $\boldsymbol{c}$ in v since v is closed. Hence η can be selected in such a way that

$$0 < 1 - \eta\mu_i(\boldsymbol{c}) < 1 \text{ for } i = 1, 2, \cdots, N \text{ and } \forall \; \boldsymbol{c} \in v.$$

For this specific η, $||B(\boldsymbol{c})|| < 1$. This makes f contractive in ν. Hence the theorem. ♠

Remarks

1. Different proofs of the convergence of the gradient descent technique exist in the literature; see [26]. However, this proof is unique in that it establishes a relationship between gradient descent and contractive maps.

2. The continuity of the Hessian matrix in the neighborhood of $\boldsymbol{x}_0$ is one of the assumptions on which the above proof is based. The eigenvalues of $\mathbf{H}$ for different $\boldsymbol{c}$ are bounded because of this assumption.

3. The fixed points of the function f are different for different closed discs, for different η, and for different starting vectors. Note also that for the same η, the function $f^n(\boldsymbol{Y})$ may oscillate as n goes to infinity for some $\boldsymbol{Y}$s. The values of η for which $f^n(\boldsymbol{Y})$ oscillates depends on $\boldsymbol{Y}$ and the particular problem under consideration.

4. The selection of η is crucial to the performance of a neural network and its final result. The current methods for choosing η are heuristic, not theoretical. One such method is to choose a value of η which is very low. However, which values of η are considered low is the subjective choice of the researcher, and may in fact be quite large relative to the problem of interest.

5. Let the function g possess k local optima $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_k$ for which the corresponding Hessian matrices are positive definite. Let ν_i be a closed disc around $\boldsymbol{x}_i$ such that the corresponding Hessian matrices are positive definite for all $\boldsymbol{Y} \in \nu_i$ and for all $i = 1, 2, \cdots, k$. Let η_i be a choice of η which makes f contractive for $i = 1, 2, \cdots, k$. Set $\lambda = \min[\eta_1, \eta_2, \cdots, \eta_k]$. Let

$$f(\boldsymbol{Y}) = \boldsymbol{Y} - \lambda \nabla g(\boldsymbol{Y}) \tag{B.6}$$

Then the limit of $f^n \boldsymbol{Y}$ as n goes to infinity belongs to the set A where $A = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k\}$ if $\boldsymbol{Y}$ is an element of $\bigcup_i \nu_i$. In other words, if C is a nonempty compact subset of A, then

$$\lim_{n \to \infty} f^n(C) \in A. \spadesuit$$

In the next section, the connection between MLP and the above theorems is described.

B.5 MLP and Contractive Maps

Multilayer Perceptrons is a neural network model which is commonly used in supervised pattern classification. The connection weights in the network model are updated with the help of back propagation [126]. There are two ways of implementing MLP - batch

mode and on line. The following arguments don't depend upon the mode of implementation. However, some authors have considered adding a momentum term to the equation for updating the connection weights in MLP. The following discussion is confined to the MLP as described in [126], and as such does not include models with the momentum term modification.

The connection weights in the MLP are modified with the help of Equation B.5. The vector $\boldsymbol{Y}$ represents the weight vector while $\eta \nabla g(\boldsymbol{Y})$ represents the change in the weight vector (as noted previously, $g(\boldsymbol{Y})$ is usually the error function [148]. In the batch mode learning algorithm, the connection weights are changed after **all** the vectors in the training set have been fed to the input layer whereas in the online learning algorithm, the weights are changed after **each** vector is fed to the input layer. In the back propagation algorithm, initially, the connection weights in the topmost layer are modified. Then the connection weights in the next lower layer are modified, and so on, until the weights in the bottom-most layer are modified. The modification of connection weights in any layer is done using Equation B.5. It is expected that the back propagation algorithm will provide a local optimal solution. It is not difficult to reach a stable value for g by making a few trials with η.

Theorem B.2 establishes the relationship between MLP and fractals. It specifies that the gradient descent technique employed by MLP converges for certain values of η and certain starting values. For these values f is a contractive map. Systems of contractive maps are often used to generate fractals. Note that the choice of η depends upon the eigenvalues of the Hessian matrix. For this reason, given a real life problem, it may be very difficult to use Theorem B.2 (see Section B.6). However, recent developments

in neural network theory regarding the computation of the Hessian matrix for certain network models [25, 148] may ease its implementation.

B.6 Example

B.6.1 Preliminaries

We determined initially that constructing a hypothetical example involving the use of a MLP to build a mathematical model for minimizing an expression of the form $\sum(obs - exp)^2$ would be extremely difficult [148]. This is due primarily to two facts: first, that such an example would require the creation of a function $g : \mathcal{R}^w \rightarrow \mathcal{R}$ where w is the number of weights in the MLP and a choice of training sample points such that the expected results were known and could be compared to the experimental results. Second, the function $\sum(obs - exp)^2$ must have a positive definite Hessian matrix, where the derivatives are taken with respect to the network weights. However, (1) this is often not the case in multilayer perceptron learning [17, 148]), hence the existence of modified Newton methods, and (2) determining whether the Hessian is positive definite requires the calculation of its eigenvalues, a computationally intensive task if the dimension of the Hessian is large. The extreme difficulties in constructing a realistically complex example is highlighted by the lack of such an example in the existing literature.

For these reasons we chose instead to demonstrate the use of the above theory to minimize a function using a neural network of the form shown in Figure B.1 where (1) $X_1, \ldots, X_n$ are the network weights as well as the function variables, (2) $h_1(\boldsymbol{X}), \ldots, h_n(\boldsymbol{X})$ are factors of $g(\boldsymbol{X})$ (i.e., $g(\boldsymbol{X}) = h_1(\boldsymbol{X}) * \cdots * h_n(\boldsymbol{X})$), and (3) no transfer function is used. Nevertheless, if one has a function to be minimized which

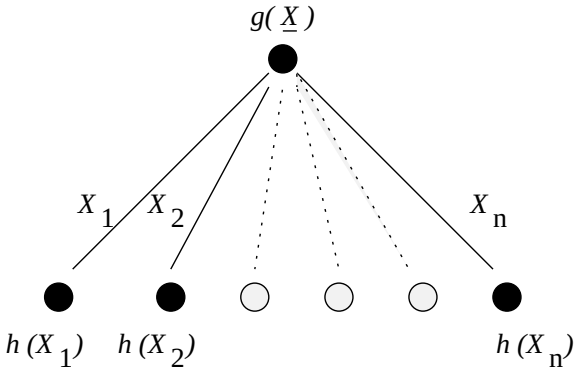


Figure B.1 Neural network model

meets the assumptions of the above theorems, then our methodology will work for solving problems of the type outlined in the previous section.

B.6.2 Construction

Let $\boldsymbol{X} = (X_1, \ldots, X_n)$. The function $g(\boldsymbol{X})$ to be minimized was required to have a positive definite Hessian matrix, i.e., $\mu_i > 0 \; \forall \; i = 1, \ldots, n$, where $(\mu_1, \ldots, \mu_n)$ are the eigenvalues of

$$H_g = \begin{bmatrix} \frac{\partial^2}{\partial^2 X_1} g(\boldsymbol{X}) & \frac{\partial^2}{\partial X_1 \partial X_2} g(\boldsymbol{X}) & \cdots & \cdots & \frac{\partial^2}{\partial X_1 \partial X_n} g(\boldsymbol{X}) \\ \frac{\partial^2}{\partial X_2 \partial X_1} g(\boldsymbol{X}) & \frac{\partial^2}{\partial^2 X_2} g(\boldsymbol{X}) & \cdots & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial X_1 \partial X_n} g(\boldsymbol{X}) & \cdots & \cdots & \cdots & \frac{\partial^2}{\partial^2 X_n} g(\boldsymbol{X}) \end{bmatrix}.$$

We chose to minimize the function $g(X_1, X_2) = 5X_1^2 + 8X_1X_2 + 5X_2^2$. In this case, the Hessian matrix is

$$H = \begin{bmatrix} 10 & 8 \\ 8 & 10 \end{bmatrix}$$

with eigenvalues $\mu_1 = 2$ and $\mu_2 = 18$.

Our neural network may be diagrammed as in Figure B.2.

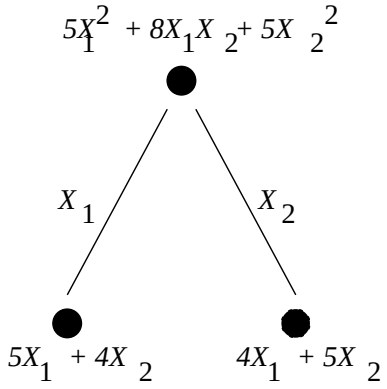


Figure B.2 Example neural network

Note that the X_1 and X_2 are the input values as well as the network weights. Given an initial input/weight vector (X_1^0, X_2^0) and a learning parameter η, the network will search for the value (X_1^*, X_2^*) which minimizes $g(X_1, X_2)$ by iteratively updating the set of weights. The updating equations are given by

$$\begin{pmatrix} X_1^{i+1} \\ X_2^{i+1} \end{pmatrix} = \begin{pmatrix} X_1^i \\ X_2^i \end{pmatrix} - \eta \begin{pmatrix} \frac{\partial g((X_1,X_2))}{\partial X_1} \\ \frac{\partial g((X_1,X_2))}{\partial X_2} \end{pmatrix}_{(X_1^i, X_2^i)} = \begin{pmatrix} X_1^i \\ X_2^i \end{pmatrix} - \eta \begin{pmatrix} 10X_1^i + 8X_2^i \\ 8X_1^i + 10X_2^i \end{pmatrix}$$

where (X_1^i, X_2^i) denotes the value of (X_1, X_2) after the i^{th} iteration and η is chosen such that

$$0 < 1 - \eta\mu_{_i} < 1 \text{ for } i = 1, 2,$$

i.e.,

$$0 < 1 - \eta(2) < 1 \text{ and } 0 < 1 - \eta(18) < 1 \quad \Rightarrow \quad 0 < \eta < 1/18.$$

Note that

$$\begin{aligned} g(X_{_1}, X_{_2}) &= 5X_1^2 + 8X_{_1}X_{_2} + 5X_2^2 \\ &= 5X_1^2 + 2 * \sqrt{5} * \tfrac{4}{\sqrt{5}} * X_{_1}X_{_2} + \tfrac{16}{5}X_2^2 + (5 - \tfrac{16}{5})X_2^2 \\ &= (5X_{_1} + \tfrac{4}{\sqrt{5}}X_{_2})^2 + \tfrac{9}{5}X_2^2 \geq 0 \end{aligned}$$

so $(X_1^*, X_2^*) = (0, 0)$. Hence we wish to choose (X_1^0, X_2^0) as a value which lies within an open disk around (0,0).

Given that $g(X_{_1}, X_{_2})$ has a positive definite Hessian matrix, if we (1) choose η such that $0 < \eta < 1/18$, (2) choose (X_1^0, X_2^0) within an open disk around (0,0), and (3) update the network weights using the given formulas, then the above theory ensures that

$$(X_1^n, X_2^n) \rightarrow (X_1^*, X_2^*) \text{ as } n \rightarrow \infty.$$

B.6.3 Results

For our experiments, we chose three initial values for $(X_{_1}, X_{_2})$ and three values for η. The value $\eta = 0.2$ is outside of (0,1/18) and was chosen for comparison purposes. The program was written so that the algorithm was considered to have converged as soon as

the updating values in the weight updating equations dropped below a tolerence value, i.e., if τ denoted the tolerence value (supplied by the user), the program would stop when

$$10X_1^i + 8X_2^i < \tau \text{ and } 8X_1^i + 10X_2^i < \tau.$$

Our experimental results are shown below. Note that n = number of iterations for updating values to fall below tolerence (i.e., until the algorithm had reached desired convergence) and D denotes scientific notation, e.g., 8.05D-2 $= 8.05 * 10^{-2}$. No tolerence value was set for $\eta = 0.2$.

Our method did converge to the minimum function value for the given starting values and the values of $\eta : 0 < \eta < 1/18$. Note that for $\eta = 0.20$, the algorithm diverged instead of converging to the minimum. This is consistent with the above theory.

Table B.1 Experimental Results

(X_1^0, X_2^0)	η	X_1^n	X_2^n	$g(X_1^n, X_2^n)$	τ	n
(1,1)	0.05	9.99D-26	9.99D-26	1.8D-49	0.05D-5	< 25
	0.10	2.037D-10	2.037D-10	7.47D-19	0.05D-5	< 100
	0.20	3.14D+41	3.14D+41	1.778D+84	$* * * * *$	< 100
(2,-3)	0.05	6.64D-05	-6.64D-05	8.818D-09	0.05D-3	< 100
	0.10	4.07D-10	-6.11D-10	7.05D-19	0.05D-6	< 100
	0.20	-1.57D+41	-1.57D+41	4.44D+83	$* * * * *$	< 100
(-0.5,-1)	0.05	6.64D-06	-6.64D-06	8.81D-11	0.05D-4	< 100
	0.10	-1.01D-10	-2.03D-10	4.25D-19	0.05D-6	< 100
	0.20	-2.35D+41	-2.35D+41	1.00D+84	$* * * * *$	< 100

B.6.4 Comments

1. At first glance, it may appear quite easy to find a function with a positive definite Hessian matrix. However, many simple, 'nice' functions do not have positive definite Hessian matrices. For example, the function $f(X_1, X_2) = X_1^2 X_2^2$ has Hessian matrix

$$\begin{bmatrix} 2X_2^2 & 4X_1X_2 \\ 4X_1X_2 & 2X_1^2 \end{bmatrix}$$

which is **not** positive definite. Note that with this function it is not possible to choose a starting value (X_1^0, X_2^0) which lies within an open disk of the minimum value. This is because when $X_1 = 0$ ($X_2 = 0$) there are infinitely many values of X_2 (X_1) at which the function reaches its minimum. One must be careful when applying the above theory to any problem.

2. For a given starting value (X_1^0, X_2^0), the method outlined above may still lead to the minimum value of a function $g(\boldsymbol{X})$ even when the learning parameter η does not meet the above specifications. Using the same function as above and a starting value of $(X_1^0, X_2^0) = (2, -2)$, we attained the results displayed in Table 2. The algorithm converged when $\eta = 0.20$ even though this value is outside of our specifications. However, the algorithm is not guaranteed to converge for such values; only if η is chosen appropriately does the above theory guarantee convergence.

Table B.2 Example with η outside specifications

(X_1^0, X_2^0)	η	X_1^n	X_2^n	$g(X_1^n, X_2^n)$	τ	n
(2,-2)	0.05	5.31D-05	-5.31D-05	5.644D-09	0.05D-3	< 100
	0.10	4.07D-10	-4.07D-10	3.319D-19	0.05D-7	< 100
	0.20	1.306D-22	-1.306D-22	3.414D-44	0.05D-20	< 100

B.7 Discussion

A relationship between the gradient descent technique and contractive maps has been derived. Noting that gradient descent is used in multilayer perceptron learning and the application of contractive maps gives rise to fractals, the above relationship is a connection between MLPs and fractals. Given the amount of research and attention being devoted to both neural networks and fractals, we hope that this connection will attract the attention of researchers and lead to advancements in both subjects.

Now that a connection has been made between these two subjects, this connection can be examined with future research. The similarity between gradient descent and fractal generation may lead to improvements in the gradient descent algorithm (hence enhancing MLP performance), as well as improvements in other optimization algorithms which use iterative techniques (such as genetic algorithms).

The utility of Theorem B.2 in modifying the connection weights for real life problems needs to be explored. We would also like to examine what Theorem B.2 can tell us about the selection of MLP parameters such as η (to avoid oscillation or divergence) and how recent advances in Hessian computation may ease implementation.

References

[1] N. Adachi and K. Matsuo. Ecological dynamics under different selection rules in distributed and iterated prisoner's dilemma game. In H.-P. Schwefel and R. Männer, editors, *Lecture notes in computer science: Parallel problem solving from nature*, pages 388–394. Springer-Verlag: Berlin and Heidelberg, 1991.

[2] G. Agarwal and W. Studden. Asymptotic design and estimation using linear splines. *Communications in Statistics B. Simulation and Computation*, 7:309–320, 1978.

[3] G. Agarwal and W. Studden. Asymptotic integrated mean square error using least squares and bias minimizing splines. *Annals of Statistics*, 8(6):1307–1325, 1980.

[4] H. Akaike. Information theory and an extension of the maximum likelihood principle. In B. Petrox and F. Caski, editors, *Second International Symposium on Information Theory*, page 267. Akadémiai Kiadó: Budapest, 1973.

[5] D. Andrews. A robust method for multiple linear regression. *Technometrics*, 16:523–531, 1974.

[6] P. Antognetti and V. Milutinovic, editors. *Neural Networks : Concepts, applications and implementations Vols. 1, 2, 3 and 4.* Prentice Hall: NJ, 1991.

[7] T. Apostol. *Mathematical Analysis, 2nd Edition.* Addison-Wesley: Reading, MA, 1974.

[8] R. Ash. *Real Analysis and Probability.* Academic Press: New York, 1972.

[9] T. Bäck. Optimal mutation rates in genetic search. In *Proceedings of the Fifth International Conference on Genetic Algorithms*. Morgan Kaufmann: San Mateo, CA, 1993.

[10] T. Bäck, F. Hoffmeister, and H.-P. Schwefel. A survey of Evolution Strategies. In R. Belew and L. Booker, editors, *Proceedings of the fourth international conference on genetic algorithms*, pages 2–9. Morgan Kaufmann: San Mateo, CA, 1991.

[11] M. Baines. Algorithms for optimal discontinuous piecewise linear and constant $\mathcal{L}_2$ fits to continuous functions with adjustable nodes in one and two dimensions. *Math. Comput.*, 62(206):645–669, 1994.

[12] J. Baker. Adaptive selection methods for genetic algorithms. In L. Erlbaum Associates, editor, *Proceedings of an International Conference on Genetic Algorithms*, pages 101–111, 1985. Hillsdale, MA.

[13] S. Bandyopadhyay, C. Murthy, and S. Pal. Pattern classification using genetic algorithms: Determination of H. *Pattern Recognition Letters*, 19(13):1171–1181, 1998.

[14] S. Bandyopadhyay, C. Murthy, and S. Pal. Theoretical performance of genetic pattern classifier. *Journal of the Franklin Institute*, 336(3):408–413, 1999.

[15] M. Barnsley. *Fractals Everywhere*. Academic Press: New York, 1993.

[16] D. Barrow, C. Chui, P. Smith, and J. Ward. Unicity of best mean approximation by second order splines with variable knots. *Math. Comput.*, 32(144):1131–1143, 1978.

[17] R. Battiti. First- and second-order methods for learning: Between steepest descent and Newton's method. *Neural Computation*, 4(2):141–166, 1992.

[18] M. Bayarri. Discussion to Cox, D., 'the relation between theory and application in statistics'. *Test*, 4(2):228–233, 1995.

[19] R. Bellman and R. Roth. Curve fitting by segmented string lines. *Journal of the American Statistical Association*, 64:1079–1094, 1969.

[20] J. Berger and L. Pericchi. The intrinsic Bayes factor for model selection and prediction. *Journal of the American Statistical Association*, 91(433):109–122, 1996.

[21] J. Bernardo. Discussion to Cox, D., 'the relation between theory and application in statistics'. *Test*, 4(2):237–239, 1995.

[22] H.-G. Beyer. Toward a theory of evolution strategies: On the benefits of sex - the $(\mu/\mu, \lambda)$ theory. *Evolutionary Computation*, 3:81–111, 1995.

[23] D. Bhandari, C. Murthy, and S. Pal. Genetic algorithm with elitist model and its convergence. *International Journal of Pattern Recognition and Artificial Intelligence*, 10(6):731–747, 1996.

[24] C. Biller. Adaptive bayesian regression splines in semiparametric generalized linear models. *Journal of Computational and Graphical Statistics*, 1999. to be published.

[25] C. Bishop. Exact calculation of the hessian matrix for the multilayer preceptron. *Neural Computation*, 4(4):494–501, 1992.

[26] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press: Oxford, 1995.

[27] T. Boseniuk and W. Ebeling. Boltzmann-, Darwin-, and Haeckel-strategies in optimization problems. In H.-P. Schwefel and R. Männer, editors, *Lecture notes in computer science: Parallel problem solving from nature*, pages 430–444. Springer-Verlag: Berlin and Heidelberg, 1991.

[28] M. Bramlette. Initialization, mutation and selection methods in genetic algorithms for function optimization. In R. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 100–107. Morgan Kaufmann: San Mateo, CA, 1991.

[29] C. Breiman. Fitting additive models to data. *Computational Statistics and Data Analysis*, 15:13–46, 1993.

[30] C. Breiman and J. Friedman. Estimating optimal transformations for multiple regression and correlation (w/ discussion). *Journal of the American Statistical Association*, 80:580–619, 1985.

[31] L. Breiman. Comment to B. cheng and D. M. titterington 'neural networks: A review from a statistical perspective'. *Statistical Science*, 9(2):38, 1994.

[32] S. Buckland, K. Burnham, and N. Augustin. Model selection: An integral part of inference. *Biometrics*, 53:275–290, 1997.

[33] A. Buja, T. Hastie, and R. Tibshirani. Linear smoothers and additive models (with discussion). *Annals of Statistics*, 17:453–456, 1989.

[34] R. Caruana. personal communication, 2000.

[35] S. Chatterjee, M. Laudato, and L. Lynch. Genetic algorithms and their statistical applications: An introduction. *Computational Statistics and Data Analysis*, 22(6):633–651, 1996.

[36] D. Chen and R. Jain. A robust back propagation learning algorithm for function approximation. *IEEE Transactions on Neural Networks*, 5(3):467–479, 1994.

[37] H. Chen. Estimation of a projection-pursuit type regression model. *Annals of Statistics*, 19(1):142, 1991.

[38] B. Cheng and D. Titterington. Neural networks: A review from a statistical perspective (with discussion). *Statistical Science*, 9(2):2–30, 1994.

[39] C. Chui, P. Smith, and J. Ward. On the smoothness of best $\mathcal{L}_2$ approximants from nonlinear spline manifolds. *Math. Comput.*, 31(137):17–23, 1977.

[40] W. Cleveland and S. Devlin. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403):596–610, 1988.

[41] D. Cox. The relation between theory and application in statistics. *Test*, 4(2):207–261, 1995.

[42] P. Craven and G. Wahba. Smoothing noisy data with spline functions. *Numerische Mathematik*, 31(4):377–403, 1979.

[43] L. Davis, editor. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold: New York, 1991.

[44] T. Davis and C. Principe. A simulated annealing like convergence theory for the simple genetic algorithm. In R. Belew and L. Booker, editors, *Proceedings of the fourth international conference on genetic algorithms*, pages 174–181. Morgan Kaufmann: San Mateo, CA, 1991.

[45] C. De Boor. *A practical guide to splines*. Springer-Verlag: New York, 1978.

[46] C. de Groot and D. Würtz. Optimizing complex problems by nature's algorithms: Simulated annealing and evolution strategy - a comparative study. In H.-P. Schwefel and R. Männer, editors, *Lecture notes in computer science: Parallel problem solving from nature*, pages 445–454. Springer-Verlag: Berlin and Heidelberg, 1991.

[47] C. deBoor and J. Rice. Least squares cubic spline approximation II - variable knots. Technical Report 21, Department of Computer Science, Purdue University, 1968.

[48] K. DeJong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Ann Arbor, MI, 1975. PhD thesis.

[49] K. DeJong and W. Spears. An analysis of the interacting roles of population size and crossover in genetic algorithms. In H.-P. Schwefel and R. Männer, editors, *Lecture notes in computer science: Parallel problem solving from nature*, pages 38–47. Springer-Verlag: Berlin and Heidelberg, 1991.

[50] D. Denison, B. Mallick, and A. Smith. Automatic bayesian curve fitting. *R. Stat. Soc. Ser. B Stat. Methodol.*, 60(2):333–350, 1998.

[51] P. Dierckx. *Het aanpassen van krommen en oppervlakken aan meetpunten met behulp van spline funkties*. PhD thesis, Katholieke Universiteit Leuven, 1979.

[52] P. Dierckx. *Curve and surface fitting with splines*. Clarendon: New York, 1993.

[53] D. Donoho and I. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.

[54] D. Donoho and I. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224, 1995.

[55] D. Draper. Assessment and propagation of model uncertainty. *Journal of the Royal Statistical Society, Series B*, 57(1):45–97, 1995.

[56] J. Dugundji. *Topology*. Allyn and Bacon: Boston, MA, 1966.

[57] B. Efron. Bootstrap methods: Another look at the Jackknife. *Annals of Statistics*, 7:1–26, 1979.

[58] B. Efron and R. Tibshirani. Boostrap for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science*, 1:54–77, 1986.

[59] A. Eiben, E. Aarts, and K. van Hee. Global convergence of genetic algorithms: A Markov Chain analysis. In *Parallel problem solving from nature*. Springer-Verlag: Berlin and Heidelberg, 1990.

[60] R. Eubank. Discussion to Ramsay, J., 'monotone regression splines in action'. *Statistical Science*, 3(4):425–462, 1988.

[61] R. Eubank. *Spline Smoothing and Nonparametric Regression*. Marcel Dekker: New York, 1988.

[62] R. Eubank and P. Speckman. Curve fitting by polynomial-trigonometric regression. *Biometrika*, 77(1):1–9, 1990.

[63] J. Fan and I. Gijbels. Data-driven bandwidth selection in local polynomial fitting: Variable bandwidth and spatial adaptation. *Journal of the Royal Statistical Society, Ser. B.*, 57(2):371–394, 1995.

[64] J. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19(1):1–141, 1991.

[65] J. Friedman and B. Silverman. Flexible parsimonious smoothing and additive modeling (with discussion). *Technometrics*, 31:3–39, 1989.

[66] C. Geyer and E. Thompson. Annealing markov chain monte carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90:909–920, 1995.

[67] D. Goldberg. Simple genetic algorithms and the minimal deceptive problem. In L. Davis, editor, *Genetic algorithms and simulated annealing*, pages 74–88. Morgan Kaufmann: Los Altos, CA, 1987.

[68] D. Goldberg. Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems*, 5(2):139–167, 1991.

[69] P. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732, 1995.

[70] M. Hansen and C. Kooperberg. Spline adaptation in extended linear models. *Statistical Science*, 1999. submitted.

[71] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall: London, 1990.

[72] D. Hawkins. On the choice of segments in piecewise approximation. *Journal of the Institute of Mathematics and its Applications*, 9:250–256, 1972.

[73] F. Herrera, M. Lozano, and J. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioral analysis. *Artificial Intelligence Review*, 12(4):265–319, 1998.

[74] J. Hesser and R. Männer. Towards an optimal mutation probability for genetic algorithms. In H.-P. Schwefel and R. Männer, editors, *Lecture notes in computer science: Parallel problem solving from nature*, pages 23–32. Springer-Verlag: Berlin and Heidelberg, 1991.

[75] J. Hoeting, D. Madigan, A. Raftery, and C. Volinsky. Bayesian model averaging: A tutorial (with discussion). *Statistical Science*, 14(4), 1999.

[76] F. Hoffmeister and T. Bäck. Genetic algorithms and evolution strategies: Similarities and differences. In H.-P. Schwefel and R. Männer, editors, *Lecture notes in computer science: Parallel problem solving from nature*, pages 455–469. Springer-Verlag: Berlin and Heidelberg, 1991.

[77] J. Holland. *Adaptation in natural and artificial systems.* The University of Michigan Press: Ann Arbor, MI, 1975.

[78] C. Holmes and B. Mallick. Parallel chain MCMC and genetic type algorithms in bayesian computation. *Statistics and Computing*, 1998. submitted.

[79] Y. Hu. An algorithm for data reduction using splines with free knots. *IMA Journal of Numeical Analysis*, 13:328–343, 1993.

[80] P. Huber. *Robust Statistics*. John Wiley & Sons: New York, 1981.

[81] C. Hurvich, J. Simonoff, and C. Tsai. Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. *Journal of the Royal Statistical Society, Series B*, 60(2):271–293, 1998.

[82] Mathsoft Inc. *Splus 3.4, Release 1.* Cambridge, MA., 1996.

[83] Visual Numerics Inc. *IMSL Fortran Numerical Libraries version 10.*

[84] C. Janikow and Z. Michalewicz. An experimental comparison of binary and floating point representation in ga. In R. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 31–36. Morgan Kaufmann Publishers: San Mateo, CA, 1991.

[85] D. Jupp. Approximation to data by splines with free knots. *SIAM Journal on Numerical Analysis*, 15(2):328–343, 1978.

[86] C. Karr, D. Stanley, and B. Scheiner. Genetic algorithm applied to least squares curve fitting. Technical Report 9339, U.S. Dept. of the Interior, Bureau of Mines, 1991.

[87] C. Karr and B. Weck. Improved computer modelling using least median squares curve fitting and genetic algorithms. *Fluid/Practice Separation Journal*, 8(2):117–124, 1995.

[88] R. Kass and A. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90:773–795, 1995.

[89] P. Kokkonis and V. Leute. Least squares splines approximation applied to multi-component diffusion data. *Computational Materials Science*, 6(1):103–111, 1996.

[90] C. Kooperberg, S. Bose, and C. Stone. Polychotomous regression. *Journal of the American Statistical Association*, 92(437):117–127, 1997.

[91] H. Larson. Least squares estimation of linear splines with unknown knot locations. *Computational Statistics and Data Analysis*, 13(1):1–8, 1992.

[92] H. Lee. A framework for nonparametric regression using neural networks. *Journal of the American Statistical Association*, 1999. submitted.

[93] R. Lenth. Robust splines. *Communications in Statistics: Theory and Methods*, 6:847–854, 1977.

[94] T-H. Li, C. Lucasius, and G. Katerman. Optimization of calibration data with the dynamic genetic algorithm. *Analytica Chimica Acta*, 2768:123–134, 1992.

[95] M. Lindstrom. Penalized estimation of free-knot splines. *Journal of Computational and Graphical Statistics*, 8(2):333–352, 1999.

[96] P. Loach and A. Wathen. On the best least squares approximation of continuous functions using linear splines with free knots. *IMA J. Numer. Anal.*, 11(3):393–409, 1991.

[97] Z. Luo and G. Wahba. Hybrid adaptive splines. *Journal of the American Statistical Association*, 92(437):107–115, 1997.

[98] T. Lyche and K. Mørken. A data reduction strategy for splines with applications to the approximation of functions and data. *IMA Journal of Numerical Analysis*, 8:185–208, 1988.

[99] B. Mallick. Bayesian curve estimation by polynomials of random order. Preprint, April 1996.

[100] C. Mallows. Some comments on C_p. *Technometrics*, 15:661–675, 1973.

[101] K. Man, K. Tang, S. Kwong, and W. Halang. *Genetic algorithms for control and signal processing*. Springer-Verlag: New York, 1997.

[102] M. Manela, N. Thornhill, and J. Campbell. Fitting spline functions to noisy data using a ga. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA, 1993.

[103] J. Marsden and A. Tromba. *Vector Calculus*. W. H. Freeman and Company: New York, 1988.

[104] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Computational Physics*, 21:1087–1092, 1953.

[105] M. Meyer, editor. *CMLIB*. Carnegie Mellon University, Pittsburgh, PA., 1994.

[106] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs*. Springer-Verlag: New York, 1996.

[107] Z. Michalewicz and C. Janikow. Handling constraints in genetic algorithms. In L.B. Booker, editor, *Proc. of Fourth Int. Conf. on Genetic Algorithms*, pages 151–157. Morgan Kaufmann: San Mateo, CA, 1991.

[108] B. Miller and D. Goldberg. Genetic algorithms, selection schemes and the varying effects of noise. Technical Report 95009, Illinois Genetic Algorithms Laboratory, Univeristy of Illinois at Urbana-Champaign, 1995.

[109] C. Murthy. personal communication, 1998.

[110] C. Murthy, S. Bandyopadhyay, and S. Pal. Genetic algorithm-based pattern classification: Relationship with bayes classifier. In S.K. Pal and P.P Wang, editors, *Genetic Algorithms for Pattern Recognition*. CRC Press: Boca Raton, FL, 1996.

[111] C. Murthy and J. Pittman. Optimal line fitting using genetic algorithms. Technical Report 99-02, Department of Statistics, The Pennsylvania State University, July 1997.

[112] G. Nason and B. Silverman. The discrete wavelet transform in s. *Journal of Computational and Graphical Statistics*, 3:163–191, 1994.

[113] A. Nix and M. Vose. Modeling genetic algorithms with Markov Chains. *Annals of Mathematics and Artificial Intelligence*, 5(1):79–88, 1992.

[114] S. Pal, S. Bandyopadhyay, and C. Murthy. Genetic algorithms for generation of class boundaries. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 28(6):816–828, 1998.

[115] S. Pal and P. Wang, editors. *Genetic Algorithms for Pattern Recognition*. CRC Press: Boca Raton, FL, 1996.

[116] L. Pericchi. Discussion to Cox, D., 'the relation between theory and application in statistics'. *Test*, 4(2):245–248, 1995.

[117] J. Pezzack, R. Norman, and D. Winter. An assessment of derivative determining techniques used for motion analysis. *Biomechanics*, 10:377–382, 1977.

[118] C. Posse. Projection pursuit exploratory data analysis. *Computational Statistics and Data Analysis*, 20(6):669, 1995.

[119] X. Qi and F. Palmieri. Theoretical analysis of evolutionary algorithms with an inifinite population size in continuous space part i: Basic properties of selection and mutation. *IEEE Transactions on Neural Networks*, 5(1):102–119, 1994.

[120] A. Raftery. Bayesian model selection in social research (with discussion). In P. Marsden, editor, *Sociological Methodology 1995*, pages 111–196. Blackwell: Malden, MA, 1995.

[121] A. Raftery, D. Madigan, and J. Hoeting. Bayesian model averaging for linear regression models. *Journal of the American Statistical Association*, 92(437):179–191, 1997.

[122] B. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press: Cambridge, MA, 1996.

[123] D. Rogers. G/SPLINES: A hybrid of friedman's 'multivariate adaptive regression splines'. In L.B. Booker, editor, *Proc. of Fourth Int. Conf. on Genetic Algorithms*. Morgan Kaufmann: San Mateo, CA, 1991.

[124] G. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101, 1994.

[125] G. Rudolph. Convergence rates of evolutionary algorithms for a class of convex objective functions. *Control and Cybernetics*, 26:375–390, 1997.

[126] D. Rumelhart, J. McClelland, and the PDP Research Group., editors. *Parallel Distributed Processing, Vol. 1.* MIT Press: Cambridge, MA, 1986.

[127] I. Schoenberg and A. Whitney. On Pólya frequency functions III: The positivity of translation determinants with an application to the interpolation problem by spline curves. *Transactions of the American Mathematical Society*, 74:246–259, 1953.

[128] L. Schumaker. *Spline Functions: Basic Theory.* John Wiley & Sons: New York, 1981.

[129] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[130] H. Schwetlick and T. Schütze. Least squares approximation by splines with free knots. *BIT*, 35(3):361–384, 1995.

[131] T. Shively, R. Kohn, and S. Wood. Variable selection and function estimation in additive nonparametric regression using a data-based prior. *Journal of the American Statistical Association*, 94(447):777–807, 1999.

[132] J. Simonoff. *Smoothing methods in statistics*. Springer-Verlag: New York, 1996.

[133] P. Smith. Curve fitting and modeling with splines using statistical variable selection techniques. Technical Report 166034, NASA Langley Research Center, 1982.

[134] P. Smith and R. Kohn. Nonparametric regression using Bayesian variable selection. *J. Econometrics*, 75:317–344, 1996.

[135] J. Spall, S. Hill, and D. Stark. Some theoretical comparisons of Evolutionary Computation and other optimization approaches. In *Proceedings of the Congress on Evolutionary Computation*, pages 1398–1405, 1999.

[136] C. Stone. Optimal global rates of convergence for nonparametric regression. *Annals of Statistics*, 10:1040–1053, 1982.

[137] C. Stone, M. Hansen, C. Kooperberg, and Y. Troung. Polynomial splines and their tensor products in extended linear modeling, with discussion. *Annals of Statistics*, 25(4):1371–1470, 1997.

[138] Y. Tourigny and M. Baines. Analysis of an algorithm for generating locally optimal meshes for $\mathcal{L}_2$ approximation by discontinuous piecewise polynomials. *Math. Comput.*, 66(218):623–650, 1997.

[139] P. van Laarhoven and E. Aarts. *Simulated annealing: theory and applications*. Dordrecht: Reidel, 1987.

[140] P. Vankeerberghen, J. Smeyers-Verbeke, R. Leardi, C. Karr, and D. Massart. Robust regression and outlier detection for non-linear models using genetic algorithms. *Chemometrics and Intelligent Laboratory Systems*, 28(1):73–87, 1995.

[141] M. Vose. Generalizing the notion of schema in genetic algorithms. *Artificial Intelligence*, 50:385–396, 1991.

[142] M. Vose and G. Liepins. Schema disruption. In R. Belew and L. Booker, editors, *Proceedings of the fourth international conference on genetic algorithms*, pages 237–242. Morgan Kaufmann: San Mateo, CA, 1991.

[143] G. Wahba. Discussion to Ramsay, J., 'monotone regression splines in action'. *Statistical Science*, 3(4):425–462, 1988.

[144] K. Warwick and R. Craddock. An introduction to radial basis functions for system identification a comparison with other neural network methods. In *Proceedings of the IEEE Conference on Decision and Control*, volume 1, pages 464–469, 1996.

[145] E. Wegman and I. Wright. Splines in statistics. *Journal of the American Statistical Association*, 78:351–365, 1983.

[146] S. Weisberg. *Applied Linear Regression, 2nd Edition*. John Wiley & Sons: New York, 1985.

[147] M. Werman and Z. Geyzel. Fitting a second degree curve in the presence of error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):207–211, 1995.

[148] J. Wille. On the structure of the hessian matrix in feedforward networks and second derivative methods. In *Proceedings of the ICNN*, pages 1851–1855, 1997.

[149] J. Ye. On measuring and correcting the effects of data mining and model selection. *Journal of the American Statistical Association*, 93(441):120–131, 1998.

Vita

Jennifer L. Pittman

Education

Ph.D. in Statistics **May 2000**
The Pennsylvania State University, University Park, PA
Thesis title: *Adaptive Splines and Genetic Algorithms for Optimal Statistical Modeling*
Advisor, Professor C.R. Rao

M.S. in Statistics **May 1995**
Carnegie Mellon University, Pittsburgh, PA

B.A. in Mathematics **May 1993**
B.A. in Psychology
Skidmore College, Saratoga Springs, NY

Selected Professional Experience

Statistical Consulting Center **5/99 to present**
The Pennsylvania State University, University Park, PA
Co-Director and Senior Graduate Consultant

Department of Statistics **8/95 to 5/99**
The Pennsylvania State University, University Park, PA
Research Assistant, Course Instructor, Teaching Assistant

Department of Statistics **8/93 to 5/95**
Carnegie Mellon University, Pittsburgh, PA
Research Assistant, Course Instructor, Teaching Assistant

Selected Publications

Pittman, J. (1999). Adaptive Splines and Genetic Algorithms. *Journal of Computational and Graphical Statistics*, accepted for publication.

Murthy, C.A. and **Pittman, J.** (1998). Multilayer Perceptrons and Fractals. *Information Sciences*, 112, **1**, 137-150.

Pittman, J. and Murthy, C.A. (1997). Fitting Optimal Piecewise Linear Functions Using Genetic Algorithms. *IEEE Pattern Analysis and Machine Intelligence*, accepted for publication.

Personal website: *http://www.stat.psu.edu/∼pittman*