Methods for Shape-Constrained Kernel Density Estimation

Mark A. Wolters
The University of Western Ontario

Supervisor
W. John Braun
The University of Western Ontario

METHODS FOR SHAPE-CONSTRAINED KERNEL DENSITY ESTIMATION

(Thesis format: Monograph)

by

Mark <u>Wolters</u>

Graduate Program in Statistical and Actuarial Sciences

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO
School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Supervisor	Examiners
Dr. W. John Braun	Dr. Duncan Murdoch
	Dr. Reg Kulperger
	Dr. Craig Miller
	Dr. Hanna K. Jankowski

The thesis by

Mark Anthony Wolters

entitled:

Methods for Shape-Constrained Kernel Density Estimation

is accepted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Date	Chair of the Thesis Examination Board

Abstract

Nonparametric density estimators are used to estimate an unknown probability density while making minimal assumptions about its functional form. Although the low reliance of nonparametric estimators on modelling assumptions is a benefit, their performance will be improved if auxiliary information about the density's shape is incorporated into the estimate. Auxiliary information can take the form of shape constraints, such as unimodality or symmetry, that the estimate must satisfy. Finding the constrained estimate is usually a difficult optimization problem, however, and a consistent framework for finding estimates across a variety of problems is lacking.

It is proposed to find shape-constrained density estimates by starting with a pilot estimate obtained by standard methods, and subsequently adjusting its shape until the constraints are satisfied. This strategy is part of a general approach, in which a constrained estimation problem is defined by an estimator, a method of shape adjustment, a constraint, and an objective function. Optimization methods are developed to suit this approach, with a focus on kernel density estimation under a variety of constraints. Two methods of shape adjustment are examined in detail. The first is data sharpening, for which two optimization algorithms are proposed: a greedy algorithm that runs quickly but can handle a limited set of constraints, and a particle swarm algorithm that is suitable for a wider range of problems. The second is the method of adjustment curves, for which it is often possible to use quadratic programming to find optimal estimates.

The methods presented here can be used for univariate or higher-dimensional kernel density estimation with shape constraints. They can also be extended to other estimators, in both the density estimation and regression settings. As such they constitute a step toward a truly general optimizer, that can be used on arbitrary combinations of estimator and constraint.

Keywords: Kernel density estimation, nonparametric statistics, shape-constrained estimation, heuristic optimization, particle swarm optimization, unimodal density estimation.

Dedicated to the memory of Lanying Ma

Acknowledgements

I would like to thank Dr. John Braun for his support and guidance over the course of my studies. In his role as supervisor he was always focused on what was best for me, both as a student and as a person, and for that I am grateful. My appreciation is also extended to the rest of the faculty and staff at the Department of Statistical and Actuarial Sciences, especially to Jennifer Dungavell and Jane Bai for their cheerful patience in administrative matters.

As my years of study have accumulated, I have developed a deeper appreciation of the impact scholarship funding has had on my career and my life. I thank the Natural Sciences and Engineering Research Council of Canada and the Ontario Student Assistance Program for their generous support.

I am also happy to have the opportunity to express my gratitude to my parents, Lawrence and Elizabeth, and to my brother John and his family, for the many good influences they have had on me; and to my wife Na for her constant love and companionship.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

An estimator of an unknown probability density, regression curve, or other function of interest is *shape constrained* if it is restricted to produce estimates having some desired qualitative features. Qualitative features that might be of interest include monotonicity, unimodality, or convexity, for example. Parametric estimators may be considered shape constrained to a high degree, as their qualitative characteristics are pre-established. Nonparametric function estimators, conversely, have a low degree of shape restriction, their qualitative features being determined primarily by the data.

This thesis concerns a middle ground between the parametric and nonparametric alternatives, where qualitative shape controls are added to a standard nonparametric estimator. There are three main advantages to this estimation approach:

1. The data analyst can choose the shape of the estimate in a way that best matches the available subject-matter knowledge, allowing the modelling assumptions to be tailored to more closely match reality. The result is better estimation than a purely nonparametric option, with less risk of model error than a purely parametric option.

2. Enforcing constraints ensures that estimates have the desired shape characteristics for all samples, not just on average or asymptotically. This provides considerable benefit in exploratory data analysis and when communicating results to others.

3. Adding a shape restriction to a smooth nonparametric estimator usually makes its performance less sensitive to the value of its smoothing parameters. This makes the estimator easier to use in practice.

The barrier to realizing these advantages, and the main concern of this thesis, is the difficult optimization problem that typically arises when building the estimate. The focus of this work is density estimation, but a general constrained estimation framework is kept in mind throughout. The concepts and methods introduced have application in other settings, including regression.

1.1 Background on Shape-Constrained Estimation

Shape-restricted estimation is more common in regression than in density estimation. The simplest constraint encountered in regression is monotonicity. The method of isotonic regression (Brunk, 1955; Barlow et al., 1972), which produces monotonic step functions, was the earliest proposal for this constraint. Smooth monotonic regression estimators have also been proposed, for example using splines (Ramsay, 1988). Extensions have also allowed other simple constraints such as concavity or convexity to be enforced (Meyer, 2008). Henderson and Parmeter (2009) summarize shape-constrained regression in much more detail.

In density estimation, the qualitative constraints that have received the most attention in the literature are monotonicity (the density must be nondecreasing or nonincreasing) and unimodality (the density must have only one peak). Note that the class of unimodal densities includes monotone densities as a special case, for which the mode is located at either the left or right edge of the density's support. Grenander (1956) developed the nonparametric maximum likelihood estimator (NPMLE) for the monotone case. For nonincreasing densities, it is the derivative of the least concave majorant of the empirical cumulative distribution function (ECDF); for nondecreasing densities, it is the derivative of the greatest convex minorant of the ECDF. The pool adjacent violators algorithm (described in Barlow et al., 1972) provides a simple means of finding the required majorant or minorant.

Later research attempted to extend the Grenander estimator to any unimodal density. The common premise was to combine a nondecreasing Grenander estimate to the left of the mode with a nonincreasing one to the right. When the location of the mode is known, this estimator is the NPMLE; otherwise the NPMLE does not exist, and other means must be used to situate the mode. Wegman (1972) considered specifying a modal interval, Bickel and Fan (1996) plugged in a consistent point estimate of the mode location, and Birgé (1997) chose the mode to minimize the distance between the estimate and the ECDF. Reboul (2005) extended the work of Birgé to a general unimodal or U-shaped function estimation setting. Like the Grenander estimator itself, all of these methods produce a step-function estimate (though Bickel and Fan did propose methods of smoothing the density after estimation).

Alternative approaches to estimation have been proposed to produce smooth density estimates directly, under the unimodality constraint or other simple constraints. Fougères (1997) uses a monotone rearrangement to transform a multimodal density estimate into a unimodal one, though under the restrictive assumption that the final mode location is known. Cheng et al. (1999) start with a unimodal template density and then iteratively apply monotone transformations (possibly with intermediate smoothing steps) to construct a more suitable unimodal estimate. The method of rearrangements has also been used to find monotone, convex, or log-concave estimates (Birke 2009; see references therein for additional alternatives with these constraints).

When a shape-restricted estimation problem is stated formally, it typically leads to a constrained optimization problem. All of the methods just summarized ease the burden of constrained optimization in one of two ways: they either define the estimator such that it satisfies the constraint by construction, or they limit attention to certain constraints for which the optimization is straightforward (usually cases where the constraints can be stated as a system of linear inequalities). As a result there are numerous estimators, each applicable to a narrow range of problems and each using its own unique methodology. Furthermore, most of the constrained estimators discussed so far are have little connection to standard unrestricted estimators that may be familiar to practitioners. These factors form a barrier to adoption of the methods. A data analyst wishing to explore three different shape restrictions, for example, may be required to learn and implement three different estimators, none of which resembles the estimator the analyst would choose in the absence of constraints.

Recent research has begun to address this problem by developing more general approaches that can apply a variety of different constraints to familiar nonparametric estimators in a consistent way. Braun and Hall (2001) and Hall and Kang (2005), for example, used data sharpening (shifting the data points) to satisfy a variety of qualitative constraints in both density estimation and regression; and Du et al. (2010), expanding on Hall and Huang (2001), used weights on the data points to enforce a broad class of derivative constraints on kernel regression estimates. The principle at work in each case is the application of a generic method of shape adjustment to enforce constraints on a standard nonparametric estimator. Because approaches like shifting or re-weighting data points are so general, they can work with any estimator, and can in principle handle arbitrary constraints or high-dimensional problems. Such methods, and their application specifically to shape-constrained kernel density estimation, are discussed further in Section 1.3.

1.2 The Problem Defined in General

As mentioned above, many previous attempts at constrained estimation have sacrificed generality for the sake of easier optimization problems. The spirit of the present work is to reverse this idea: to retain a general estimation methodology, at the cost of more difficult constrained optimization. The primary goal of this thesis is to present heuristic optimization techniques that make such an approach feasible. First, we will consider a general description of the constrained estimation problem.

Let $\mathbf{x}$ be a random sample of size n. Our preferred nonparametric estimator for the function of interest—the one we would use in the absence of any constraints—is the *pilot* estimator, denoted $\hat{f}^{\circ}(x)$. It is assumed that the pilot estimate fails to meet the desired shape restrictions (otherwise the case is trivial).

To be able to handle constraints, the algebraic form of $\hat{f}^{\circ}$ must admit some means of shape adjustment. Let $\boldsymbol{q}$ be a vector of adjustable values in the formula for $\hat{f}^{\circ}$ that allows its shape to be altered. Let $\hat{f}_{\boldsymbol{q}}$ represent the shape-adjusted estimate at a particular value of $\boldsymbol{q}$. Further, define $\mathbf{t}$ to be the target value of $\boldsymbol{q}$—the value that reproduces the pilot estimate (that is, $\hat{f}_{\mathbf{t}} = \hat{f}^{\circ}$). Note that the pilot estimator may contain other parameters, for example a bandwidth. It is assumed for the moment that such parameters are chosen independently (see Section 2.4 for more on this topic).

Define the functional $\mathcal{I}$ to indicate when an estimate satisfies all desired shape constraints: $\mathcal{I}(\hat{f}_{\boldsymbol{q}}) = 1$ when the constraints are satisfied, and $\mathcal{I}(\hat{f}_{\boldsymbol{q}}) = 0$ otherwise. Let $\mathcal{C}$ be the set of $\boldsymbol{q}$ vectors that produce constraint-respecting estimates:

$$\mathcal{C} = \{\boldsymbol{q} : \mathcal{I}(\hat{f}_{\boldsymbol{q}}) = 1\}. \tag{1.1}$$

Any $\boldsymbol{q}$ in $\mathcal{C}$ is called a feasible solution.

The optimal shape-constrained estimate can now be identified. It is defined to be $\hat{f}_{\boldsymbol{q}^*}$, where $\boldsymbol{q}^*$ is the solution to the constrained optimization problem

$$\boldsymbol{q}^* = \underset{\boldsymbol{q} \in \mathcal{C}}{\operatorname{argmin}} \; \delta(\boldsymbol{q}, \mathbf{t}), \tag{1.2}$$

and $\delta(\boldsymbol{q}, \mathbf{t})$ is an objective function measuring the closeness of $\boldsymbol{q}$ and $\mathbf{t}$.

Note from the above that four elements are required to define a constrained estimation problem:

1. A pilot estimator. This could be a standard nonparametric estimator, for example a kernel density estimator (as considered in the following chapters), or a local regression estimator, a smoothing spline, and so on.

2. A method of adjusting the estimate, achieved by varying a set of values. Four adjustment methods suitable for kernel density estimators are introduced in Section 1.3.

3. A set of shape constraints. Many types of shape constraints suitable for density estimation are discussed in Section 2.2.

4. An objective function, such as the squared-error distance. The choice of objective function is addressed in Section 2.3.

The difficulty of problem (1.2) depends on the specifics of these four components in the problem at hand. The shape constraints are particularly important from the optimization standpoint. In some cases the constraint $\boldsymbol{q} \in \mathcal{C}$ can be translated into a set of inequalities in $\boldsymbol{q}$. If the inequalities take a simple form (especially if they are linear), it may be possible to use standard methods of mathematical programming to find the optimal solution. If the inequality constraints are nonlinear or nonconvex, the problem will be challenging, and could have many local optima. In other cases it may not even be possible or practical to express the constraint as explicit inequalities in $\boldsymbol{q}$. Furthermore, a change to any of the four elements of the problem creates a new mathematical programming problem that must be worked out afresh.

If one abandons the mathematical programming approach and uses heuristic optimization techniques to find solutions, it is possible to leave the constraints in their *black box* form, and use the constraint-checking functional $\mathcal{I}(\hat{f}_{\boldsymbol{q}})$ in the search. This allows a much wider array of constraints to be considered. It is also more flexible in a computer implementation, because it is usually easy to write functions to check the validity of different constraints. This is the approach explored in the current work. Of course, when one uses heuristic search algorithms to solve optimization problems it is often necessary to give up any guarantees of convergence or optimality. This is of little consequence in problems with nonlinear or nonconvex constraints, however, because the mathematical programming methods also cannot guarantee global optimality. The purpose of defining the objective function $\delta(\boldsymbol{q}, \mathbf{t})$ is to define which solutions are better than others; so whichever method provides the solution with the smallest objective value is to be preferred.

The above ideas are made more concrete in the next section, where they are applied to kernel density estimators.

1.3 The Case of Kernel Density Estimation

The constraint handling approach followed here involves first constructing a preliminary or pilot estimate, and subsequently adjusting its shape in some manner to satisfy the required constraints. Henceforth we will be concerned only with density estimation, and will exclusively use kernel estimates as pilot densities. The kernel density estimator (KDE) is defined below, and four methods of adjusting its shape are subsequently introduced. The first three methods are natural extensions of the functional form of the KDE, while the fourth approach is a new proposal.

1.3.1 The Kernel Density Estimator

Let $\mathbf{x}$ be a set of n independent observations from a distribution with probability density function (pdf) f. The kernel density estimator of f is

$$\hat{f}\,(u) = \frac{1}{n}\sum_{i=1}^{n} K_h(u - x_i), \tag{1.3}$$

where K_h is a kernel function with scale parameter (or bandwidth) h. The kernel function integrates to 1. It is usually symmetric around zero and often nonnegative (in which case it is a pdf). If f is uniformly continuous, K_h satisfies very mild regularity conditions, and $h \to 0$ at an appropriate rate as $n \to \infty$, the KDE is a uniformly consistent estimator. The texts by Silverman (1986) and Wand and Jones (1995) provide more information on these and other aspects of kernel density estimation.

The Gaussian kernel is used exclusively here, so K_h is taken to be a N$(0,h^2)$ density. The Gaussian KDE may be written

$$\hat{f}^{\circ}(u) = \frac{1}{nh}\sum_{i=1}^{n} \phi\left(\frac{u - x_i}{h}\right), \tag{1.4}$$

where $\phi(\cdot)$ is the standard normal density function. The notation $\hat{f}^{\circ}$ will be used throughout to denote the pilot (unadjusted) estimator. For the moment it is assumed that h is known, or chosen by some automatic rule. More will be said about bandwidth

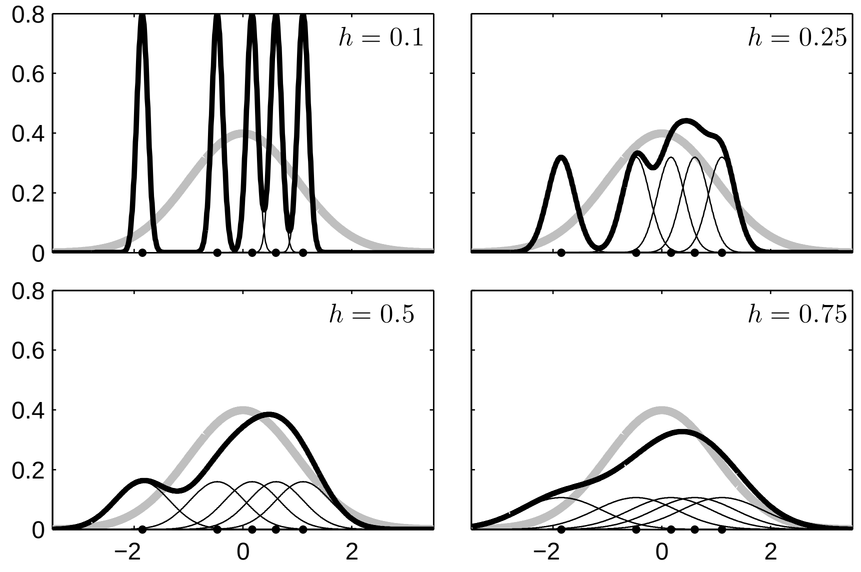

Figure 1.1: Kernel density estimates for a small sample using four different bandwidths. The data are sampled from a standard normal distribution (thick grey curve). The dark lines are the density estimates. The kernel functions for each point are also shown.

selection in Section 2.4.

Density estimate (1.4) is an evenly-weighted mixture of n normal densities, with one mixture component centered at each data point, and each component having variance h^2. Assuming no duplicate observations, $\hat{f}$ will have n modes if h is sufficiently small; if it is sufficiently large, there will be only one mode. This is illustrated in Figure 1.1. For an intermediate h value, the density estimate may have spurious or unwanted extra modes, especially in the tail regions where there are few points.

Thinking of the KDE as a mixture density suggests that it may be generalized by allowing the locations, scales, and weights of its components to vary. This more general estimator is

$$\hat{f}\,(u) = \sum_{i=1}^{n} \frac{w_i}{b_i} \phi \left(\frac{u - m_i}{b_i} \right), \tag{1.5}$$

where the ith component has location m_i, standard deviation (bandwidth) b_i, and weight w_i. The parameters of the mixture are $\mathbf{m} = [m_1 \dots m_n]^T$, $\mathbf{b} = [b_1 \dots b_n]^T$ (with $b_i > 0$, $\forall i$), and $\mathbf{w} = [w_1 \dots w_n]^T$ (with $w_i \geq 0$ and $\sum w_i = 1$). Note that the standard KDE (1.4) is reproduced when

$$\begin{aligned} \mathbf{m} &= \mathbf{x} \\ \mathbf{w} &= \frac{1}{n}\mathbf{1} \\ \mathbf{b} &= h\mathbf{1}, \end{aligned} \tag{1.6}$$

where $\mathbf{1}$ is an n-vector of ones. We refer to the right hand sides of equations (1.6) as the *target* values for $\mathbf{m}$, $\mathbf{w}$, and $\mathbf{b}$.

Jones and Henderson (2005, 2009) use this n-component Gaussian mixture as a density estimator with no shape constraints. They allow some or all of the triple $\{\mathbf{m}, \mathbf{w}, \mathbf{b}\}$ to vary (fixing the others at their target values), and fit the mixture to data using maximum likelihood. When $\mathbf{b}$ is allowed to vary, maximum likelihood fitting is made possible by the restriction that the geometric mean of the bandwidths $\left(\prod_{i=1}^{n} b_i\right)^{1/n}$ must equal some overall bandwidth h.

We will follow this example in considering the Gaussian KDE to be a normal mixture, but rather than finding maximum likelihood estimates, we will treat the pilot estimate $\hat{f}^{\circ}$ as our preferred estimator, and focus on adjusting it as little as possible to satisfy shape constraints. The first three approaches to shape adjustment are to vary $\mathbf{m}$, $\mathbf{w}$, or $\mathbf{b}$. The fourth is a new proposal. The four options are qualitatively summarized below; mathematical details are given as needed in later chapters.

1.3.2 Varying the Locations (Data Sharpening)

The pilot estimate (1.4), thought of as a mixture distribution, uses the observations $\mathbf{x}$ as its location parameters. So varying the locations of the estimator to accommodate shape constraints amounts to modifying the observed data. As a general strategy for improving the performance of statistical estimators, this practice is known as data sharpening. Let $\mathbf{x}$ represent the original or *unsharpened* data, and $\mathbf{y}$ represent the modified or *sharpened* data (that is, we use $\mathbf{m} = \mathbf{x}$ in the pilot estimate, and $\mathbf{m} = \mathbf{y}$ in the constrained estimate).

The original motivation for introducing data sharpening into density estimation was to reduce the asymptotic bias of kernel density estimators (Choi and Hall, 1999; Hall and Minnotte, 2002). The sharpening done by these original methods had the effect of partially clustering the data to make the modes of the estimate more peaked. In the method of Choi and Hall, $\mathbf{y}$ is found by performing local constant regression of $\mathbf{x}$ on $\mathbf{x}$. This operation tends to move points toward local modes. If iterated to convergence (as done by Woolford and Braun 2007), it will identify one or more points in the vicinity of the density's peaks, making it a mode-finding algorithm rather than a density estimator.

The use of data sharpening to accommodate constraints was outlined by Braun and Hall (2001), in both the density estimation and regression contexts. Additional theoretical and practical details were added by Hall and Kang (2005), for the case of unimodal kernel density estimation. In this form of sharpening, $\mathbf{y}$ is determined through an optimization step, rather than by a clustering procedure. The intent is to choose a $\mathbf{y}$ that causes the constraints to be satisfied, while perturbing $\mathbf{x}$ as little as possible. Figure 1.2 illustrates the principle, using the $n = 5$ example introduced in Figure 1.1. Assume that we start with $\hat{f}^{\circ}$ having bandwidth $h = 0.5$, and we seek a unimodal estimate. The pilot estimate is bimodal, but in this case the sharpened estimate can be constructed by shifting only one of the five points. The leftmost point is shifted the minimum distance necessary to render the estimate unimodal.

In keeping with the notation introduced previously, the data sharpening estimator with location values $\mathbf{y}$ will be denoted $\hat{f}^{M}_{\mathbf{y}}(x)$. The superscript M is included to remind us that the locations m_i are being adjusted, and to distinguish this shape-adjusted estimator from the alternative ones introduced below.

The objective function $\delta(\mathbf{y}, \mathbf{x})$, once specified, determines which sharpened data set is best. A natural choice is to base the objective on a norm of the difference $\mathbf{y} - \mathbf{x}$, defining

$$L_\alpha(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{n} |y_i - x_i|^\alpha, \qquad 1 \le \alpha \le 2. \tag{1.7}$$

Both Braun and Hall (2001) and Hall and Kang (2005) found some evidence that the L_1 distance had better statistical performance than the more standard L_2, but

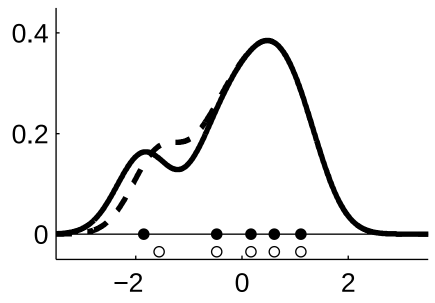

Figure 1.2: A small example illustrating the premise of data sharpening. The solid curve is the pilot KDE based on $\mathbf{x}$ (filled circles), and the dashed curve is the sharpened KDE based on $\mathbf{y}$ (open circles).

also that optimization suffered from numerical difficulties when $\alpha = 1$. They used sequential quadratic programming (SQP) to carry out the optimization (for detail on SQP, see Nocedal and Wright 1999, ch. 18, or Antoniou and Lu 2007, ch. 15; further comments on SQP are also found in Section 2.2.1).

1.3.3 Varying the Weights

A second way to accommodate constraints in the KDE is to change the mixture weights $\mathbf{w}$ from the standard n^{-1} weight on each point, to a non-uniform weighting. This is the approach used, in the regression context, by Hall and Huang (2001). They use a probability vector of weights to render a kernel regression estimator monotone, by minimizing a distance measure between $\mathbf{w}$ and the target $\frac{1}{n}\mathbf{1}$. This approach was significantly extended by Du et al. (2010), who applied the same adjustment principle to multivariate kernel regression. Their method can handle a wider range of constraints, namely those expressible as linear inequality constraints on the derivatives of the regression function. In their formulation the weights are allowed to take negative values, and the L_2 distance is used to measure the deviation of $\mathbf{w}$ from the target. Sequential quadratic programming was again the optimizer of choice.

The same adjustment method can be readily applied to density estimation, and the weight-adjusted KDE will be denoted by $\hat{f}^W_{\mathbf{w}}(x)$. Figure 1.3 provides a demonstration, using the same unimodality-constrained example of the previous two figures.

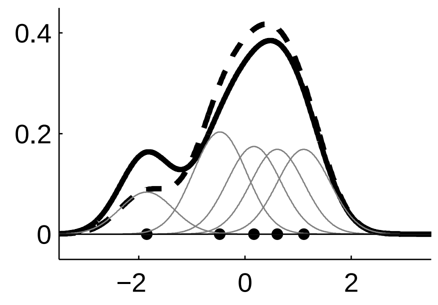

Figure 1.3: A small example illustrating the premise of varying the weights. The solid curve is the pilot KDE, and the dashed curve is the unimodal adjusted KDE. The weighted kernel functions used to produce the unimodal curve are also shown in grey.

It shows that a unimodal estimate can be achieved by reducing the weight on the left-most point, and increasing the weight on the second point (this solution was found using the L_2 objective). This example demonstrates that when using weights to enforce a constraint locally, a global change in the estimate can result, because of the requirement that $\sum w_i = 1$. This effect becomes less noticeable as the sample size increases, however. Note also that if the nonnegativity requirement on $\mathbf{w}$ is lifted as in Du et al. (2010), it is necessary to add a further shape constraint that $\hat{f}^W_{\mathbf{w}}(x) \geq 0, \forall x$, to eliminate the chance of obtaining a density estimate with negative function values.

1.3.4 Varying the Bandwidths

A KDE with a different bandwidth value b_i for each data point is known as a *variable KDE* (Wand and Jones, 1995, p. 42). The attraction of the variable KDE is that it permits varying degrees of smoothing in different parts of the density. Silverman (1986, p. 21), for example, suggests letting b_i equal the distance from x_i to its kth nearest neighbour to achieve more smoothing in data-sparse regions and less smoothing in data-rich regions.

The idea of using a variable KDE to satisfy constraints does not appear to be documented in the literature. Figure 1.4 shows how an adjustment of $\mathbf{b}$ can be used to achieve unimodality, using the same example considered previously ($n = 5$, target bandwidth 0.5, L_2 objective). The spurious mode in the pilot estimate is eliminated

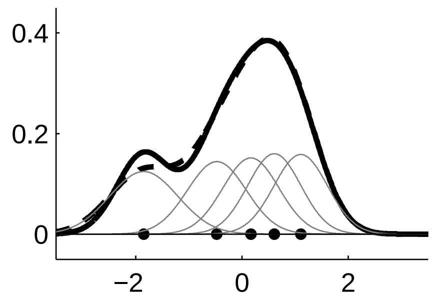

Figure 1.4: A small example illustrating the premise of varying the bandwidths. The solid curve is the pilot KDE, and the dashed curve is the unimodal adjusted KDE. The variable-bandwidth kernel functions used to produce the unimodal curve are also shown in grey.

by increasing the bandwidths used by the first three points. Other constraints could be handled in a similar manner. The notation $\hat{f}^{B}_{\mathbf{b}}(x)$ will be used to refer to the shape-constrained variable KDE.

Remark

The variable KDE approach has particular appeal because it potentially obviates the need to independently choose a pilot bandwidth. Adjusting the bandwidths to minimize a distance measure between $\mathbf{b}$ and $h_0\mathbf{1}$ still involves the problem of selecting h_0, but there are two possible strategies to eliminate this step. The first is to set h_0 to an artificially small value, or even to zero. In this case the optimization will attempt to move each b_i toward smaller values, but the shape constraint will prevent individual bandwidths from becoming too small. The second is to drop the notion of a target $\mathbf{b}$ value, and instead to maximize the likelihood of $\mathbf{x}$ under $\hat{f}$, subject to the shape constraint. This latter approach is similar to the method of Jones and Henderson (2005, 2009), who found the maximum likelihood estimate of $\mathbf{b}$ in the absence of shape constraints on $\hat{f}$. Their procedure requires a constraint on the geometric mean of the b_i, but with a sufficiently strict shape constraint, this is not necessary. Despite the promise of these ideas, they are not pursued further here because they are beyond the scope of the present work.

1.3.5 Using an Adjustment Curve

The final method of shape adjustment considered here is a new proposal. A simple option for modifying the shape of a pilot KDE is to add a function to it, that can annihilate any unwanted features of $\hat{f}^{\circ}$, or add any desired features that are not present. This added function will be referred to as an *adjustment curve*. Figure 1.5 continues the small example of the previous pages, showing how an adjustment curve can be used to achieve unimodality. To ensure that the final estimate is still a density, the adjustment curve must integrate to zero, and must not introduce any negative density values. As seen in the figure, the adjustment curve is zero in locations distant from constraint violations, and takes nonzero values only where the constraint is being violated.

The crucial concerns, of course, are how the adjustment curve is constructed and how the optimal adjustment curve is determined. These topics are addressed in Chapter 5. There, it is proposed to let the adjustment curve be a linear combination of appropriately chosen density functions ψ_i, making the estimator

$$\hat{f}^{A}_{\boldsymbol{a}}(x) = \hat{f}^{\circ}(x) + \sum_{i=1}^{k} a_i \psi_i(x). \tag{1.8}$$

The coefficients $\boldsymbol{a} = [a_1 \cdots a_k]^T$ of this linear combination are the adjustable parameters of the method, and they can be chosen to minimize an appropriate objective function. In this formulation, many common constraints can be handled in a quadratic programming framework.

1.3.6 Shape Adjustment in Higher Dimensions

The generalized Gaussian KDE of equation (1.5) can be extended to d dimensions as the d-variate normal mixture

$$\hat{f}\,(\mathbf{u}) = \sum_{i=1}^{n} w_i \mathrm{N_d}(\mathbf{u}; \mathbf{m}_i, \mathbf{B}_i), \tag{1.9}$$

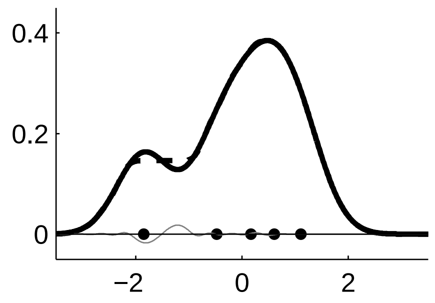

Figure 1.5: A small example illustrating the premise of adjustment curves. The solid curve is the pilot KDE, and the dashed curve is the unimodal adjusted KDE. The adjustment curve used to produce the unimodal estimate is also shown in grey.

where $\mathrm{N_d}(\,\cdot\,;\mathbf{m}_i,\mathbf{B}_i)$ is the d-variate normal density with mean $\mathbf{m}_i$ and covariance matrix $\mathbf{B}_i$. We may consider how each of the shape adjustment methods just introduced can be extended to this multivariate situation.

Data sharpening

In the univariate case, data sharpening involves minimally perturbing n scalar points to enforce a constraint. The extension to d dimensions is conceptually straightforward. Each $\mathbf{m}_i$ is now a d-vector with target $\mathbf{x}_i$, and the set of n such vectors must be perturbed to accommodate the constraint. The objective function used in the optimization must measure a distance between the sets of vectors $\{\mathbf{m}_i\}$ and $\{\mathbf{x}_i\}$, and the optimizer itself must operate on such sets of vectors. A simple option, for example, is to stack the $\{\mathbf{m}_i\}$ and $\{\mathbf{x}_i\}$ vectors into column vectors of length nd and use an L_2 distance.

Weight adjustment

A practical advantage of the weight adjustment approach is that it does not depend on the dimension of the data. There is a single n-vector $\mathbf{w}$ of adjustable parameters, regardless of the value of d in (1.9). So no particular difficulties arise with multivariate data.

Bandwidth adjustment

The complexity of constraint handling by bandwidth adjustment depends on how the matrices $\{\mathbf{B}_i\}$ in (1.9) are defined. Three possibilities are, in increasing order of complexity,

$$\mathbf{B}_i = b_i\mathbf{I}, \tag{1.10}$$

$$\mathbf{B}_i = \mathrm{diag}(\mathbf{b}_i), \tag{1.11}$$

and

$$\mathbf{B}_i = \text{an arbitrary covariance matrix.} \tag{1.12}$$

Here $\mathbf{I}$ is the identity matrix and $\mathrm{diag}(\mathbf{v})$ is the diagonal matrix with jth diagonal element v_j. The first option (1.10) implies a radially symmetric standard normal kernel function, and requires only one bandwidth to be specified per point, n bandwidths in all. This option does not require any extra effort when moving from $d = 1$ to higher dimensionality. The second option (1.11) implies a product-kernel structure for the density estimate. It requires a d-vector of parameters for each data point (nd bandwidths in all). The final, fully general option (1.12) requires a nonnegative definite matrix with $d(d+1)/2$ unique elements to be specified for each data point. This option is typically not practical except possibly for $d = 2$.

Adjustment curves

In d dimensions the pilot density $\hat{f}^\circ$ is a (hyper) surface, and so what was an adjustment curve in one dimension must become an adjustment surface. In the univariate case, the adjustment curve is defined as a linear combination of univariate density functions. To expand this to higher dimensions, in principle it is only necessary to let the adjustment densities ψ_i be d-dimensional. Practical difficulties arise, however, because the number of adjustment densities required to construct a good adjustment curve rises rapidly with d. This problem is discussed in Chapter 5, which also includes an example of bivariate adjustment.

1.4 Overview of the Thesis

Shape-constrained nonparametric estimation is a large topic, and the range of problems expressible in the manner of Section 1.2 is broad. Some limitations on the scope of the project are necessary. These limitations are reviewed next, and the layout of the remainder of the thesis is described afterwards.

1.4.1 Scope of the Present Work

It has already been mentioned that this work focuses exclusively on kernel density estimation; although the methods developed here can be extended to other density estimators or regression estimators, such extensions are only mentioned in passing. Similarly, while four methods of shape adjustment have just been listed, only two of them (data sharpening and the method of adjustment curves) will be considered in subsequent chapters. Having thus restricted attention to a single estimator and two adjustment methods, the work will focus on different algorithms for handling a variety of constraints.

The language and notation of the thesis will often imply univariate data, but as described in Section 1.3.6, the adjustment methods can be extended to higher dimensions. The new optimization heuristics are also designed with the $d > 1$ case in mind. To make this more clear, each heuristic will be demonstrated on both a univariate and a bivariate data set. Where data or adjustable values ($\mathbf{x}$ or $\mathbf{y}$, for example) are expressed as vectors, the multivariate extension will require these symbols to be thought of as collections of vectors (or as a matrix). Additional complications arising from the jump to higher dimensions will be addressed as they arise.

1.4.2 Plan of the Thesis

Five chapters follow this one. Chapter 2 explores shape constraints that are suitable for density estimation. A variety of useful constraints are proposed. A new idea in this chapter is to base shape constraints on the number of inflection points of the density or its derivatives. Different possibilities for the objective function $\delta(\cdot, \cdot)$ are briefly discussed as well. The problem of pilot bandwidth selection is also addressed in this chapter, and a new likelihood-based bandwidth selector is proposed.

Chapters 3 and 4 describe two new constrained estimation optimizers for data sharpening. A greedy algorithm is the subject of Chapter 3. This algorithm has many practical advantages, but it is limited in some respects because of its greedy design. It works well when the constraint is unimodality, but it is not suitable for more difficult constraints. The limitations of the algorithm are alleviated somewhat by incorporating it as part of a metaheuristic known as iterated local search. A more generally applicable algorithm is given in Chapter 4. This algorithm is based on particle swarm optimization, but with several unique features designed to let the search handle the difficult constraints arising in constrained estimation problems. The swarm-based algorithm is more computationally intensive, but can handle a much wider range of problems.

The method of adjustment curves is described in Chapter 5. While heuristic optimization methods are advocated throughout the thesis, the adjustment curve is constructed in such a way that globally optimal solutions for many important constraints can be found by quadratic programming (QP). For this reason the QP framework is the main focus of the chapter. Adjustment curves for constraints not fitting the QP structure can still be found using heuristic optimizers (such as the algorithm of Chapter 4).

Chapter 6 is a concluding chapter that identifies areas of this research that would benefit from further study.

Chapter 2

Defining Constraints and Finding Estimates

The preceding chapter was primarily concerned with nonparametric estimators and methods of shape adjustment. The remaining two elements of a constrained estimation problem—the shape constraint and the objective function—are discussed below in some detail. The difficult question of how to choose a bandwidth for the pilot density is also addressed. Two data sets, that will be used to illustrate the ideas of this and subsequent chapters, are first introduced.

2.1 Two Illustrative Examples

One univariate data set, the wind speed data, and one bivariate data set, the heart disease data, have been chosen as examples.

2.1.1 Wind Speed Data

Alibrandi and Ricciardi (2008) reported data on 57 wind speed measurements made at each of five different elevations in Italy's Messina Strait region. We will consider only the measurements made at the lowest elevation, 10 meters. The minimum and maximum speeds measured were 5.6 and 30.4, respectively (units of measurement were not reported). The quartiles of the data were $(Q_1, Q_2, Q_3) = (10.1, 14.3, 15.8)$.

Figure 2.1 shows kernel density estimates based on this data set, for six different bandwidth choices in the range $1 \leq h \leq 3.5$. The estimate with $h = 1$ has three modes. The central peak is the highest, while the mode on the right side is attributable to a single outlying point. As the bandwidth is increased, the mode on the left shrinks and becomes a shoulder in the main peak before $h = 2$ (a shoulder in an estimate is here taken to mean a change in concavity that does not produce a mode). Further increases in bandwidth cause further smoothing of the central mode. It takes a bandwidth greater than about $h = 3.4$ to finally cover the outlying point and render the estimate unimodal.

This data set demonstrates why it might be desirable to have shape control on a density estimate, beyond what is possible through bandwidth selection. As a point of reference, a popular automatic bandwidth selection rule (the Sheather-Jones bandwidth, discussed in Section 2.4) chooses $h = 1.55$ for these data. Although this selector generally works well, in this case it results in an estimate with three modes. Given the small sample size and the nature of the quantity being measured, however, it might be reasonable to require that the density estimate be unimodal, or at least that it have smooth tails. It is also natural to require that the density estimate have negligible probability mass for speeds less than zero. It is clear from Figure 2.1 that these requirements on the density cannot be satisfied by an unconstrained KDE with any value of h.

2.1.2 Heart Disease Data

The bivariate example is based on a South African study of risk factors for coronary heart disease (Hastie and Tibshirani, 1987; Hastie et al., 2009). Two variables from a larger data set are considered: systolic blood pressure (SBP) and concentration of low density lipoprotein (LDL). Only the measurements of the $n = 160$ diseased patients in the study are included. For convenience, both variables have been standardized to have mean zero and unit standard deviation.

Figure 2.2 shows the heart disease data graphically. The scatter plot of LDL

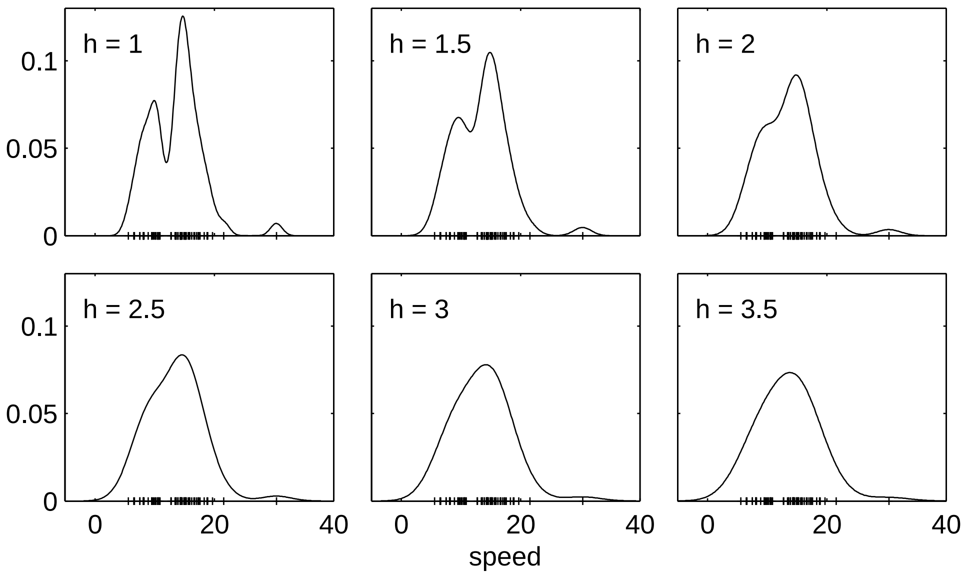

Figure 2.1: Kernel density estimates for the wind speed data set, with six different bandwidths.

against SBP is repeated four times, with different density estimates shown as contour plots. Contour lines on each graph are drawn to enclose, from outermost to innermost, probability mass of 0.95, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, and 0.05 (a convention that will be followed on all subsequent contour plots as well). The first three plots show bivariate KDEs using the Gaussian product kernel, with three different bandwidths (for simplicity, and because the data are not highly correlated, the same bandwidth has been used for both variables). The fourth plot shows the best bivariate normal fit to these data.

Visual inspection of the fourth plot suggests that the bivariate normal model does not capture all of the meaningful features visible in the scatter plot. While it is possible to consider data transformations or alternative parametric models, the simplicity and data-driven nature of kernel methods are attractive. The addition of shape constraints offers the potential to maintain the data-driven nature of the estimate, while smoothing out some of the irregularities visible in the first three plots of Figure 2.2. These irregularities are likely caused by sampling variation rather than underlying data structure.

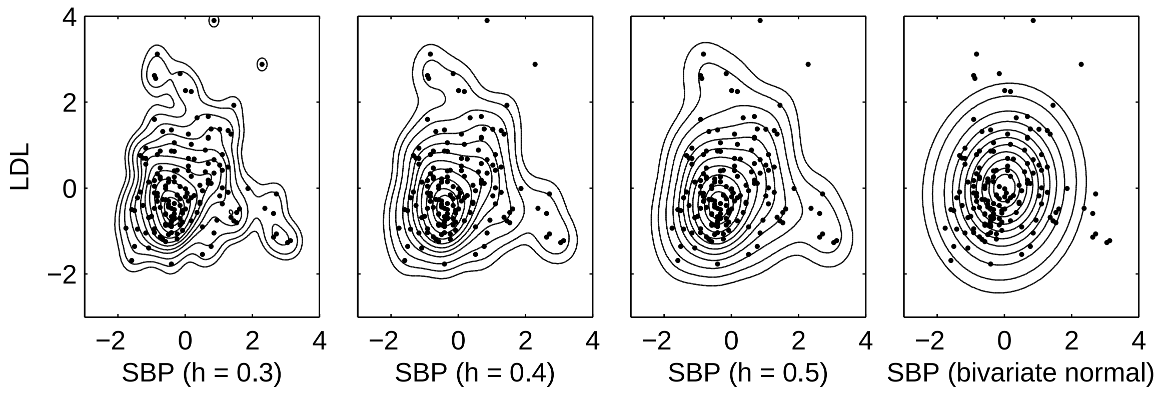

Figure 2.2: Bivariate density estimates for the heart disease data. The first three estimates are kernel density estimates with different bandwidths; the last estimate is the maximum likelihood bivariate normal fit.

2.2 A Suite of Useful Shape Constraints

The crucial step of deciding which shape constraints to apply in a given situation has not yet been addressed. Little general advice can be given, since this is a problem-specific question. In some cases there may be theoretical reasons to expect a density to have particular characteristics like monotonicity or unimodality. In small-sample situations, shape restrictions provide a way to impose smoothness on an estimate, and to eliminate spurious modes that typically arise in the density's tails. Even when n is larger, shape constraints offer an auxiliary way to realize the smoothness assumption that is made in kernel density estimation (hopefully reducing sensitivity to bandwidth choice in the process). Finally, it could be beneficial in the exploratory stage of data analysis to consider a variety of shape restrictions. This section presents a number of potential shape constraints that could be useful in any of these situations.

Remark

The definitions and constraints to follow make reference to a density function f. It is assumed that the density in question has the characteristics typical of kernel density estimators, particularly continuity and differentiability up to the necessary order. We will begin by considering constraints applicable to univariate densities, move on to bivariate constraints afterwards.

2.2.1 Constraints on the number of Modes

The most obvious constraint one can apply to a KDE is a restriction on the number of modes. Some elementary definitions are required first to eliminate confusion.

Definition 2.1 (mode) *A point m is a mode of a density f if f is increasing in a neighbourhood to the left of m and decreasing in a neighbourhood to the right (i.e., it is a local maximum of f). An interval (m_1, m_2) where f is constant and these conditions are satisfied is also called a mode.*

Definition 2.2 (two-tailed density) *A density with support $[a, b]$ is called two-tailed if neither a nor b is a mode.*

Definition 2.3 (sign change) *Let u be a zero of a function g. Then g has a sign change at u if there exists an $\epsilon > 0$ such that $g(u - \omega)g(u + \omega) < 0$ for all $\omega \in (0, \epsilon)$. If g is zero on an interval (u_1, u_2), we say that g has a sign change over this interval if $g(u_1 - \omega)g(u_2 + \omega) < 0$ for all $\omega \in (0, \epsilon)$.*

Definition 2.1 is the usual definition of a mode applied to continuous density estimates, but it is written here to distinguish our usage of the term from the one typically used in the discrete case (where the mode occurs at the support point(s) with the global maximum relative frequency). Also, note that a plateau in the density, where the function value is constant, is counted as a single mode if it is higher than its neighbouring points on either side.

Definition 2.2 is included to allow distinction between monotone densities (which are also unimodal) and other unimodal densities. Because we are considering the KDE with Gaussian kernel, we will concern ourselves primarily with constraints suitable for two-tailed densities defined on the whole real line.

The definition of a sign change is also the natural one, and is also extended to include the case of an interval over which the function g is zero. To count the sign changes in a function, we may ignore any zeros of the function and observe how many times the signum function $\text{sgn}(g)$ changes value. Counting sign changes is a convenient basis for implementing constraint-checking functions on a computer. Algorithm 2.1

describes the function `signchanges` that performs this task using a set of function values evaluated at a grid of points.

Algorithm 2.1: Counting sign changes (`signchanges`).

```
Input: A vector of $r$ function values, $\mathbf{f}$, evaluated at increasing abscissa values.
Output: $c$, the number of sign changes in $\mathbf{f}$.
Set $\mathbf{s} \leftarrow \text{sgn}(\mathbf{f})$
Delete any elements of $\mathbf{s}$ that are zero.
Let $\mathbf{d}$ be the first difference of $\mathbf{s}$ (that is, $d_i = s_{i+1} - s_i$).
$c \leftarrow$ the number of nonzero elements of $\mathbf{d}$.
```

Definitions 2.1 and 2.3 were careful to treat constant intervals where f is a maximum or $g = 0$ appropriately, counting each interval as only a single mode or sign change. This is important because such regions can in fact arise (at least to numerical tolerances) in KDEs adjusted by data sharpening or adjustment curves. Figure 1.5 in the previous chapter illustrated this: the shape-adjusted density estimate had a plateau across what was originally the unwanted second mode. In the constraints to follow it is important that such plateaus are counted only as a single feature of the density.

A formal definition of k-modality for a univariate density can now be presented. Because the unimodal ($k = 1$) case is so important, it is given as a separate constraint.

Constraint 1 (unimodality) *A two-tailed density $f(x)$ is unimodal if there exists an m such that f is nondecreasing to the left of m and nonincreasing to the right. That is,*

$$\begin{aligned} f'(x) &\geq 0 \quad \text{if } x \leq m \\ f'(x) &\leq 0 \quad \text{if } x \geq m. \end{aligned}$$

Equivalently, the density is unimodal if $f'(x)$ has exactly one sign change.

Constraint 2 (k-modality) *A two-tailed density $f(x)$ is k-modal ($k \geq 1$) if its first derivative has exactly $2k - 1$ sign changes.*

Consider how the problem of finding a unimodal KDE may be set up as a mathematical programming problem. Let $\boldsymbol{q}$ be the vector of adjustable parameters and $\mathbf{t}$ be its target value. Let the constraint be enforced at a grid of points $\mathbf{g} = (g_1, \ldots, g_G)$ covering the support of the estimate[1]. Take the mode location m as fixed. If m falls between grid points p and $p+1$, the problem may be formulated as

$$\boldsymbol{q}^* = \operatorname*{argmin}_{\boldsymbol{q}} \delta(\boldsymbol{q}, \mathbf{t}) \quad \text{subject to} \quad \begin{cases} \hat{f}'_{\boldsymbol{q}}(g_i) \geq 0, & i = 1, \ldots, p \\ \hat{f}'_{\boldsymbol{q}}(g_i) \leq 0, & i = p+1, \ldots, G, \end{cases} \tag{2.1}$$

where $\hat{f}_{\boldsymbol{q}}$ is the constrained estimate and $\boldsymbol{q}^*$ is the optimal solution. The unimodality constraint has been converted into a set of G inequalities involving $\boldsymbol{q}$. Expressed this way, the problem looks like a standard problem in constrained nonlinear optimization, and it is possible to use numerical optimization routines (SQP, for example) in an attempt to solve it. The dependence of (2.1) on m is manifested in the value of p, which changes as m varies. It is therefore necessary to run the SQP solver inside an additional 1-D optimization scheme to determine the best value of m. This is not trivial, since argmin $\delta(\boldsymbol{q}, \mathbf{t})$ is not necessarily a convex function of m. An exhaustive search over a range of possible of m values is a reasonable strategy.

The requirement of fixed m for running SQP in the unimodal case is manageable, but the situation quickly becomes more cumbersome for the k-modal case. Taking bimodality ($k = 2$) as an example, there are three important points at which the derivative changes sign: the left mode of the estimate falls at some point m_1, the right mode at m_3, and the minimum value between the modes falls at m_2. Let p_1, p_2, and p_3 be the elements of $\mathbf{g}$ just to the left of m_1,m_2, and m_3, respectively. Then the

[1]The elements of $\mathbf{g}$ are assumed to be monotonically increasing. For kernels with unbounded support like the Gaussian, it is sufficient to let the grid extend beyond the smallest and largest observations.

problem is a search for

$$\boldsymbol{q}^* = \underset{\boldsymbol{q}}{\operatorname{argmin}}\ \delta(\boldsymbol{q}, \mathbf{t}) \quad \text{subject to} \quad \begin{cases} \hat{f}'_{\boldsymbol{q}}(g_i) \geq 0, & i = 1, \ldots, p_1 \\ \hat{f}'_{\boldsymbol{q}}(g_i) \leq 0, & i = p_1 + 1, \ldots, p_2 \\ \hat{f}'_{\boldsymbol{q}}(g_i) \geq 0, & i = p_2 + 1, \ldots, p_3 \\ \hat{f}'_{\boldsymbol{q}}(g_i) \leq 0, & i = p_3 + 1, \ldots, G. \end{cases} \tag{2.2}$$

There is no extra difficulty in solving problem (2.2) for a single choice of $\{p_1, p_2, p_3\}$, but as these quantities are not known beforehand, SQP will need to run inside a three-dimensional optimization routine in order to find the best estimate.

The mathematical programming approach to setting up the problem can be contrasted with the heuristic optimization strategy using a black-box constraint, where constraint validity only needs to be checked using the indicator functional $\mathcal{I}(\hat{f}_{\boldsymbol{q}})$. In this case the constraint checking function `haskmodes(`$\hat{f}_{\boldsymbol{q}}$`,`k`)` (Algorithm 2.2) can take the role of $\mathcal{I}(\hat{f}_{\boldsymbol{q}})$, and it can be used for any choice of k. It is easy to evaluate and introduces no extra difficulties. Solutions for a variety of k values can be found without extra effort, as long as the heuristic optimizer is capable of finding good solutions.

Algorithm 2.2: Checking for k modes in a density estimate (`haskmodes`).

```
Input: $\hat{f}$, a density estimator; $k$, the number of modes to check for.
Output: TF, a logical variable (true if the constraint is satisfied).
Set $\mathbf{g} \leftarrow [g_1 \cdots g_G]^T$, a vector of increasing values covering the data range.
Set $\mathbf{f} \leftarrow [\hat{f}'(g_1) \cdots \hat{f}'(g_G)]^T$
if signchanges(f) $= 2k - 1$                         *Use Algorithm 2.1*
    TF $\leftarrow$ true
else
    TF $\leftarrow$ false
```

Furthermore, there is also no guarantee that the mathematical programming approach will find better solutions than a heuristic optimizer, even for the unimodal case. Chapter 3 will show in more detail how a greedy heuristic method can be competitive with SQP for the unimodal estimation problem, but for the moment a small example can illustrate why this is so.

The example is a constructed problem where a KDE with $h = 1$ is to be rendered unimodal through data sharpening, by moving only two of the data points. Let the original and sharpened data vectors be

$$\begin{aligned} \mathbf{x} &= [\, -4 \quad -2.85 \quad -1 \quad -0.5 \quad 0.5 \quad 1 \quad 2.85 \quad 4 \,]^T \\ \mathbf{y} &= [\, y_1 \quad -2.85 \quad -1 \quad -0.5 \quad 0.5 \quad 1 \quad 2.85 \quad y_8 \,]^T. \end{aligned}$$

That is, the sharpening is to be performed by moving only the first and last of the eight data points. The problem has solutions of the form (y_1, y_8), allowing the objective function and the feasible region to be visualized on the plane.

The unsharpened estimate for this example is symmetric with three modes, as shown in the top plot of Figure 2.3. The L_1-optimal solution is also shown on the plot. It is found by shifting each of the two moveable points slightly away from the center of the distribution.

The bottom plot in Figure 2.3 shows the feasible region $\mathcal{C}$ superimposed on the contours of the L_1 objective. The feasible region for this case is not only non-convex, it is not even a contiguous region. Finding the best solution on the constraint boundary is a difficult problem, even with only two points to optimize.

The performance of a sequential quadratic programming algorithm is also shown on the plot. SQP was started from five random starting points, and the solution progress from each location is shown on the plot. The starting points are indicated by triangles, and the search paths are plotted as dashed lines. The true location of the mode ($m = 0$) was given to allow the algorithm to handle the unimodality constraint. Even so, SQP had difficulty dealing with the complicated constraint region, and was not able to find the true optimum from any of the starting points. In additional repetitions, SQP was not successful unless it was supplied initial values close to the true optimum.

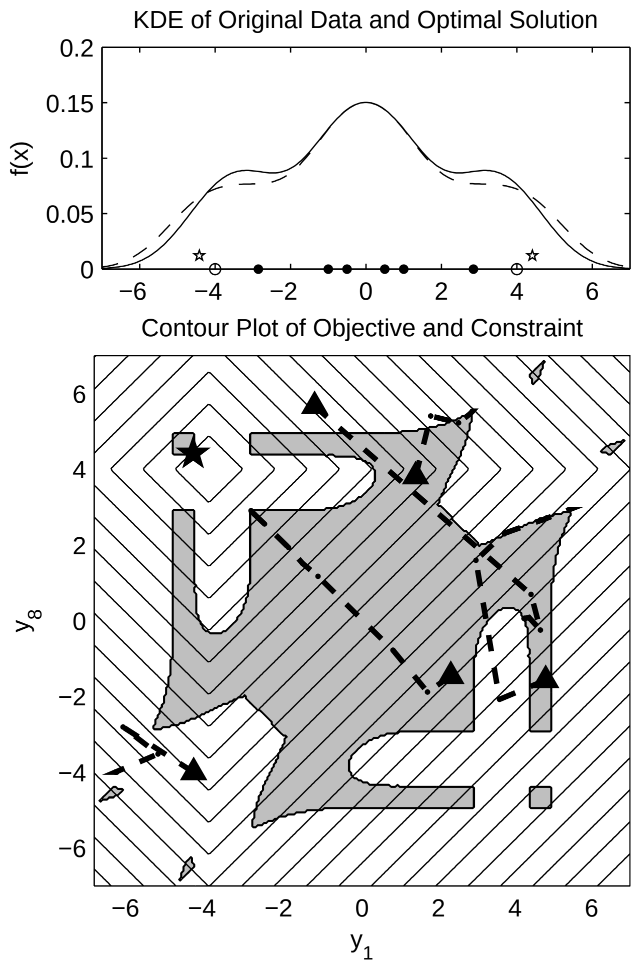

Figure 2.3: The feasible set for an example with only two moveable points. The top graph shows the original data (dots), the two moveable points x_1 and x_8 (circles), and the L_1-optimal solution (stars). The bottom graph shows the solution space, with the feasible region $\mathcal{C}$ overlaid on the contours of the L_1 objective function. The dashed lines show solution progress for optimization by SQP, for several starting values (triangles).

2.2.2 Smoother Unimodal Constraints

Mode constraints are restrictions on the number of sign changes in an estimate's first derivative. A greater degree of shape control and smoothness can be achieved by considering the sign changes of higher derivatives. This section contains several new proposals for constraints of this type, that operate on f'' and f'''.

Constraint 3 puts a restriction on the number of sign changes of the second derivative, that is, the number of inflection points of the density.

Constraint 3 (b inflections) *A two-tailed density $f(x)$ has b inflection points if its second derivative has b sign changes.*

The usual interpretation of an inflection point is of a transition between convex and concave regions of a curve. Because our definition of sign changes includes the possibility that the curve is zero over an interval, we also count as an inflection any constant region of the density that separates convex and concave portions of the curve.

The validity of this constraint can be numerically verified using the `hasbinflections` function provided in Algorithm 2.3. In this algorithm, line A involves calculating the second derivative of the density estimate at a grid of points. This can be done exactly by differentiating the kernel function, or approximately by using numerical differentiation techniques. The number of sign changes in the vector of derivative values is checked in line B. Constraining a density to have b inflections will prevent excessive waviness in the estimate[2].

The qualitative difference between constraining modes and constraining inflections is demonstrated in Figure 2.4. The figure shows a trimodal pilot estimate, with constrained estimates using the unimodal, bimodal, and four-inflections constraints. The plot is a cartoon; the actual shape of the constrained estimates will depend on the particular adjustment method and objective function used. Constraints on the number of modes do not impart a great degree of qualitative smoothness, as they

[2]It might be more sensible to constrain the estimate to have b *or fewer* sign changes, so as not to preclude very smooth estimates when such estimates are supported by the data.

Algorithm 2.3: Checking for b inflections in an estimate (`hasbinflections`).

```
    Input: $\hat{f}$, a density estimator; $b$, the number of inflections to check for.
    Output: TF, a logical variable (true if the constraint is satisfied).
    Set $\mathbf{g} \leftarrow [g_1 \cdots g_G]^T$, a vector of increasing values covering the data range.
A   Set $\mathbf{f} \leftarrow [\hat{f}''(g_1) \cdots \hat{f}''(g_G)]^T$          *Use exact or approximate derivatives*
B   if signchanges(f)=$b$                                    *Use Algorithm 2.1*
        TF ← true
    else
        TF ← false
```

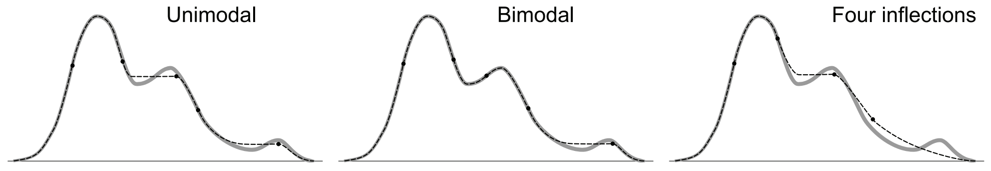

Figure 2.4: Hypothetical example of three different constraints applied to a trimodal density (thick grey curve). The dashed line shows the constrained density for each case and the inflection points on each curve are indicated by dots.

typically convert modes into plateaus in the estimate. Restrictions on inflections have a stronger smoothing effect, since reducing the number of inflections requires that the waves or shoulders in the density be eliminated.

For the important case of unimodal densities, it is possible to achieve smoother estimates by combining mode and inflection constraints. Constraints 4 and 5 are two possibilities. Numerical algorithms for checking these two constraints are not provided, as they are straightforward extensions of the previous algorithms.

Constraint 4 (two shoulders) *A two-tailed density satisfies the two shoulders constraint if it is unimodal with no more than 6 inflections.*

Constraint 5 (Bell shaped, type 1) *A two-tailed density is bell shaped, type 1, if it has exactly two inflections. This is a special case of Constraint 3, with $b = 2$.*

The two shoulders constraint could be used when a unimodal estimate is desired, but a controlled amount of data-driven waviness is permissible. The first and third constrained estimates in Figure 2.4 satisfy the two shoulders constraint.

Constraint 5 introduces the term bell shaped, which is reserved for constrained estimates that are unimodal with a degree of smoothness approaching typical parametric forms[3]. The type 1 bell shape constraint requires only that the number of inflections be exactly two. This implies that the KDE is unimodal with no shoulders.

Constraints 6 and 7 extend the family of bell shaped constraints. They impose higher degrees of smoothness by controlling the number of inflections of the first derivative of f, which is equivalent to restricting the number of sign changes of its third derivative. This may seem excessive, but there are visually discernable differences among these options.

Constraint 6 (Bell shaped, type 2) *Let f be a two-tailed density with inflection points at L and R, and call $[L, R]$ the modal interval. Then f is bell shaped, type 2 if it is concave ($f'' \leq 0$) on the modal interval and f' has exactly one inflection (f''' has one sign change) on either side of the interval.*

Constraint 7 (Bell shaped, type 3) *A two-tailed density f is bell shaped, type 3 if f' has three inflections (f''' has three sign changes).*

The three classes of bell shaped densities just defined are nested inside one another: type 2 is a subclass of type 1, and type 3 is a subclass of type 2. Figure 2.5 clarifies the differences between the types. It shows an example of a density of each type, with the first two derivatives of each density also shown. All three types consist of a unimodal density with two inflection points, at L and R. It is convenient to think of the density as separated into three segments: a modal region $[L, R]$, and two tail regions, one on either side of the modal interval. The type 1 class has no further restrictions on its shape. In the example in the figure, the type 1 density has a bend in it (we may call it a kink, or a knee, in the curve) in the modal region and one in the right tail. While the density is certainly smooth, such kinks are not characteristic of parametric densities that we may take as our ideal of qualitative smoothness.

[3]Note that the term "bell shaped" as used here only refers to the smoothness of the density as defined in Constraints 5, 6, and 7; in particular, the present definitions do not require symmetry.

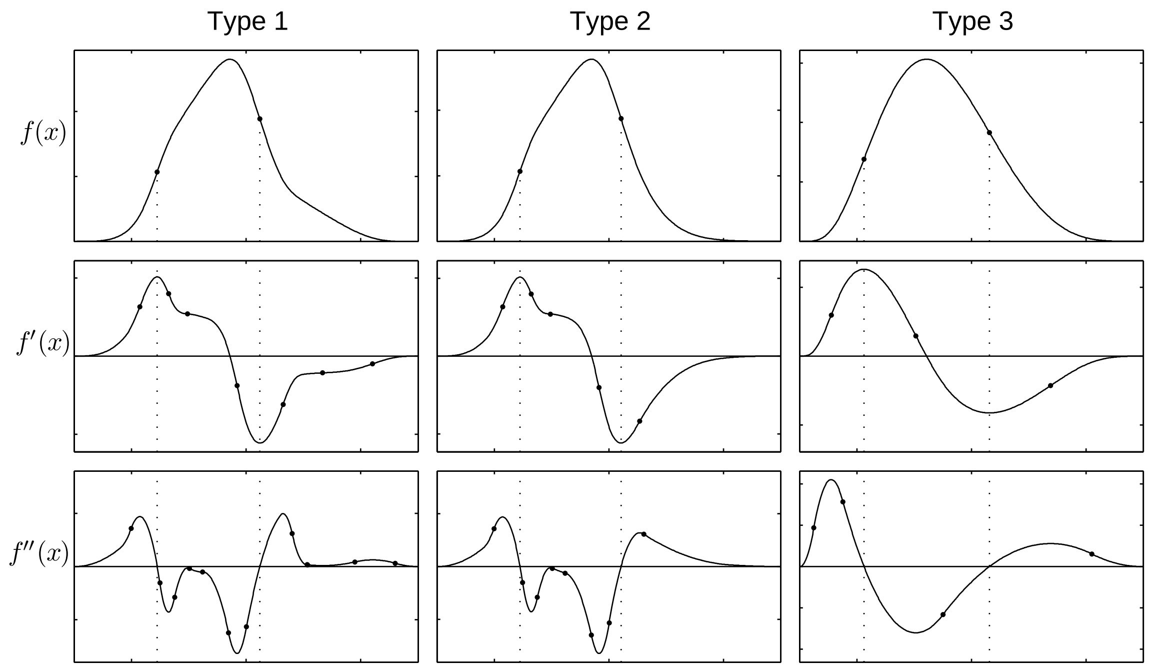

Figure 2.5: Shape differences among bell-shaped densities. Three density curves are shown, with their first two derivatives plotted underneath. Dashed lines demarcate the boundaries between the modal region and the tails. The inflection points on each curve are indicated by dots.

The kinks or knees in f can be eliminated by controlling the number of inflections f'. The type 2 bell-shaped density restricts f' to have only one inflection point in each tail region, but does not restrict the derivative in the modal region. As seen in the middle column of Figure 2.5, this type has very smooth tails but can still have knees in the modal interval. The type 3 constraint removes the possibility of knees in the modal region by requiring that f' have exactly three inflections—one in each region of the density.

The bell shaped constraints, in particular type 3, should be of considerable practical interest if they can be implemented reliably. When a data analyst uses non-parametric density estimation, it is usually because of unwillingness to commit to a particular parametric form. This does not necessarily mean that the analyst wishes to abandon the qualitative characteristics of parametric densities altogether. The bell shaped constraints capture the qualitative characteristics of parametric forms to a greater degree than simpler constraints like unimodality. Type 3 bell shape can be considered a constrained nonparametric estimate with a parametric appearance because it eliminates extra inflections in f' that standard parametric densities do not have (the normal distribution, for example, has two inflections, and its jth derivative has j+2 inflections).

2.2.3 More Univariate Possibilities

A number of other univariate constraints could be useful in different situations. They are introduced below.

Constraint 8 (monotonicity) *A density f is monotonic if its derivative has zero sign changes over its support. That is, $f'(x) \geq 0, \forall x \in \mathcal{S}$ or $f'(x) \leq 0, \forall x \in \mathcal{S}$ where $\mathcal{S}$ is the support of the density.*

Monotonicity is an important constraint for densities with bounded support, particularly densities defined on the positive half-line. This constraint is included for completeness, since it is not possible to achieve a monotonic estimate using a Gaussian KDE[4].

Constraint 9 (Log-concavity) *A density f is log-concave if* $\ln(f)$ *is a concave function.*

The log-concavity constraint has received considerable interest in the literature, partly because it is mathematically convenient. Dümbgen and Rufibach (2009) and Cule et al. (2010) summarize the theoretical and applied properties of log-concave density estimators, and supply algorithms for obtaining the unique maximum likelihood log-concave density estimate for data of any dimension. This estimate is not smooth, but it does not depend on a bandwidth parameter, which is particularly advantageous in higher dimensions. They also discuss a smoothed version of the estimate. Birke (2009) has developed a different smooth log-concave estimator, using kernel methods with a monotone rearrangement.

In one dimension, log-concave estimates are unimodal and do not have plateaus, but they may still possess kinks similar to those demonstrated in the type 1 bell-shaped densities. As well, log-concavity does not tolerate heavy tails. For example, the normal distribution is log-concave, but the t distributions are not.

Constraint 10 (nonnegative support) *Let $\hat{f}$ be a Gaussian KDE, and let X be a random variable with $\hat{f}$ as its density function. Then $\hat{f}$ has a nonnegative support, up to tolerance ϵ, if* $\mathrm{P}(X < 0) \leq \epsilon$.

Constraint 10 is intended to allow the Gaussian KDE to be used to estimate densities defined on the half-line. While the Gaussian KDE is supported on the whole line, enforcing the nonnegative support constraint with, for example, $\epsilon = 0.01$, will allow the estimator to produce practically useful estimates. Considered another way, this constraint can be used to prevent unrealistically large h values from arising when performing bandwidth selection. The wind speed example of Figure 2.1 illustrated the need for such a constraint.

[4]Monotonic estimates on the half-line can be achieved with the Gaussian KDE using special techniques, for example reflecting the data around zero and enforcing a unimodality constraint. Such techniques are not explored further here.

Constraint 11 (Symmetry) *A density f is symmetric around the point M, up to tolerance ϵ, if $|f(M-d) - f(M+d)| \leq \epsilon$, $\forall d > 0$.*

The symmetry constraint is potentially useful on its own, or in conjunction with other constraints (symmetric and unimodal, or symmetric and bell-shaped, for example). The definition in Constraint 11 includes a tolerance because it looks toward numerical implementations. Any constraint-checking function will require a tolerance to effectively evaluate whether the constraint is satisfied.

The final univariate constraint considered here is a new proposal. It is intended for cases where one believes the density should be close to a certain parametric form.

Constraint 12 (Nearly parametric) *Let $\hat{f}$ be a nonparametric density estimate, and let $\hat{g}$ be the density from a chosen parametric family that is closest to $\hat{f}$ in some sense. Then $\hat{f}$ is said to be kth order nearly equal to the parametric family, with tolerance ξ, if*

$$\int_{-\infty}^{\infty} |\hat{g}^{(k)}(x) - \hat{f}^{(k)}(x)| dx \ \leq \ \xi \int_{-\infty}^{\infty} |\hat{g}^{(k)}(x)| dx, \tag{2.3}$$

where $f^{(k)}$ denotes the kth derivative of f.

In this definition, $\hat{g}$ is the member of the parametric family that best matches $\hat{f}$; it is not necessarily the one that best matches the data (for example, it is not the maximum likelihood estimate). The nearly parametric constraint serves to define a new family of densities that, while not parametric, are within a certain distance of a chosen parametric form. Fidelity to the data is handled at a higher level, by finding the particular member of the nearly-parametric family that is closest to the pilot density estimate[5].

Two additional points are worth noting in Constraint 12. First, the definition allows the closeness of $\hat{g}$ and $\hat{f}$ to be measured based on either the densities directly ($k = 0$), or on any of their derivatives. It is expected that setting k to 1, 2, or 3 should

[5]One could define a constraint based on the distance to a fixed curve, such as a maximum likelihood density estimate or its derivative. The drawback of doing this is that, depending on the bandwidth and the means of adjusting the shape of the KDE, a feasible solution might not exist.

produce progressively smoother estimates. Second, the measure of closeness in (2.3) is only one possibility. Using this measure restricts the integrated distance between the two curves, and allows the tolerance ξ to be interpreted as a fraction of the area under $|\hat{g}^{(k)}(x)|$. An alternative is to restrict the pointwise difference between the two curves, but this has two drawbacks: i) no distinction is made between the tails and the peaks in the curve, and ii) the value of ξ is harder to interpret and choose.

The practical value of Constraint 12 is still to be determined. The ability to constrain a KDE to be close to a parametric form is desirable, but doing so requires introducing two new parameters, k and ξ, that have to be set in addition to the bandwidth. A first attempt at using this constraint is shown in Chapter 4.

2.2.4 The Bivariate Case

Defining constraints in higher dimensions is more difficult than in the univariate case, and it is also harder to implement efficient functions for checking constraint validity. Several constraints applicable to two-dimensional data are considered in this section, to explore some of the possibilities.

The first constraint is bivariate unimodality. There are several non-equivalent definitions of unimodality that can be applied to multivariate densities, such as star unimodality, level set unimodality, and α-unimodality, among others (see, for example, Gupta, 1976; Klemelä, 2009, p. 38–39). Star unimodality finds application in Chapter 5. Constraint 13 provides one way of defining it.

Constraint 13 (Star unimodality) *A multivariate density f is star unimodal with mode $\mathbf{m}$ if it is a decreasing function along all rays emanating from $\mathbf{m}$.*

For present purposes, however, the following definition of unimodality is sufficient, and acts as an easily-verified constraint in the bivariate case.

Constraint 14 (Bivariate unimodality) *A bivariate density f is unimodal with mode $\mathbf{m}$ if it has only one local maximum (at $\mathbf{m}$), and no unique local minima.*

The stipulation that the density have no local minima is required because a bivariate density with only one mode can have one or more local minima as well. Any such

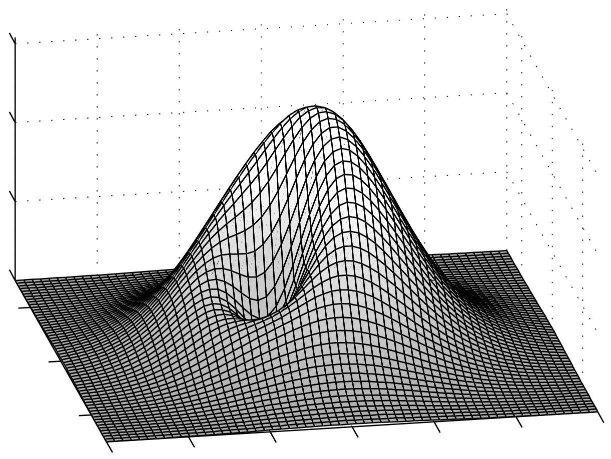

Figure 2.6: A unimodal bivariate density with a local minimum.

minimum will have the appearance of a dimple in the downward-sloping side of the density (see Figure 2.6).

To check these constraints and the ones to follow numerically, it is necessary to evaluate the density function at a rectangular grid of points. Let $\mathbf{v} = [v_1 \cdots v_{m_2}]^T$ and $\mathbf{z} = [z_1 \cdots z_{m_1}]^T$ be two regularly-spaced, increasing vectors of grid points covering the range of the data in dimensions 1 and 2, respectively. Then we may use the notation `grid(v,z)` to represent the $m_1 \times m_2$ grid with points at coordinates (v_i, z_j).

A simple means of checking the validity of Constraint 14 is given in the function `isuni2D` (Algorithm 2.4). The function evaluates the density over a rectangular grid, producing a matrix of function values. An element of this matrix is taken to correspond to a local maximum if its value is higher than those of its eight neighbouring points. Note that Algorithm 2.4 does not require the mode location as an input; this would certainly be required if setting up this constraint using mathematical programming.

Another option is to base the constraints on the density's marginal or conditional distributions. In Constraints 15 and 16, these distributions are required to be unimodal.

Constraint 15 (Unimodal marginals) *A bivariate density f satisfies the unimodal marginals constraint if both of its marginal distributions are unimodal (they satisfy*

Algorithm 2.4: Checking for 2-D unimodality (`isuni2D`).

```
Input: $\hat{f}$, a bivariate density estimator; $\mathbf{v}$ and $\mathbf{z}$, vectors of grid points in each
       dimension.
Output: TF, a logical variable (true if the constraint is satisfied).
Set $\mathbf{f} \leftarrow$ a matrix of $\hat{f}$ values evaluated at grid(v,z).
Set $n_{max} \leftarrow$ the number of local max. in $\mathbf{f}$.   *Compare points to 8 neighbours*
Set $n_{min} \leftarrow$ the number of local max. in $-\mathbf{f}$.                    *Counts minima*
if $n_{max} = 1$ and $n_{min} = 0$
    TF $\leftarrow$ true
else
    TF $\leftarrow$ false
```

Constraint 1).

Constraint 16 (Unimodal conditionals) *A bivariate density f satisfies the unimodal conditionals constraint if all of its conditional densities, in either variable, are unimodal (they satisfy Constraint 1).*

The unimodal marginals constraint is a relatively weak shape restriction, but it could be of interest if there is reason to believe each of the variables in question should have unimodal densities. Other marginal constraints (bell-shape, for example) could be used instead.

Algorithm 2.5: Checking for unimodal marginals (`unimarg`).

```
Input: $\hat{f}$, a bivariate density estimator; $\mathbf{v}$ and $\mathbf{z}$, vectors of grid points in each
       dimension.
Output: TF, a logical variable (true if the constraint is satisfied).
Set $\mathbf{f} \leftarrow$ a matrix of $\hat{f}$ values evaluated at grid(v,z).
Set $\mathbf{f_1} \leftarrow$ a vector with $j$th element $\sum_i \mathbf{f}_{ij}$.
Set $\mathbf{f_2} \leftarrow$ a vector with $i$th element $\sum_j \mathbf{f}_{ij}$.
if $\mathbf{f_1}$ and $\mathbf{f_2}$ are both unimodal                    *Use Algorithm 2.2*
    TF $\leftarrow$ true
else
    TF $\leftarrow$ false
```

The unimodal conditionals constraint, on the other hand, is a strong shape requirement, and should enforce a higher degree of smoothness on f. In practice, Constraint

15 and Constraint 16 can both be checked using a similar approach, as shown in Algorithms 2.5 and 2.6. In both cases the density is evaluated at a rectangular grid of points, and the resulting matrix **f** is used to approximate the appropriate marginal or conditional densities. The `haskmodes` function is then employed to check for unimodality. Note that checking for unimodal conditionals is more computationally intensive, because it requires checking for the unimodality of every row and column of **f**.

Algorithm 2.6: Checking for unimodal conditionals (`unicond`).

```
Input: $\hat{f}$, a bivariate density estimator; $\mathbf{v}$ and $\mathbf{z}$, vectors of grid points in each
        dimension.
Output: TF, a logical variable (true if the constraint is satisfied).
Set $\mathbf{f} \leftarrow$ an $m_1 \times m_2$ matrix of $\hat{f}$ values evaluated at grid(v,z).
for $i = 1$ to $m_1$
    Set $\mathbf{f_i} \leftarrow$ the $i$th row of $\mathbf{f}$.
    if $\mathbf{f_i}$ is NOT unimodal                                    *Use Algorithm 2.2*
        TF $\leftarrow$ false
        Terminate the algorithm.
for $i = 1$ to $m_2$
    Set $\mathbf{f_j} \leftarrow$ the $j$th column of $\mathbf{f}$.
    if $\mathbf{f_j}$ is NOT unimodal                                    *Use Algorithm 2.2*
        TF $\leftarrow$ false
        Terminate the algorithm.
TF $\leftarrow$ true
```

The last two constraints to be considered involve level sets of a bivariate density function. It is common to use contour plots when visualizing a bivariate density, and each contour in the plot is a level set enclosing a certain probability mass. Often sampling variability makes the outer contours in such a plot convoluted and discontiguous. The effect of sampling variability on these contours can be reduced by requiring that certain level sets be connected, or enclose a convex region.

Constraint 17 (Contiguous contour) *Let $\mathcal{A}$ be a level set of a bivariate density f, defined either directly by the function value, or by the probability mass it encloses. Then f satisfies the constraint if $\mathcal{A}$ is a connected set. In this case $\mathcal{A}$ forms a contiguous contour.*

Constraint 18 (Convex contour) *Let $\mathcal{A}$ be the level set of a bivariate density $f(x)$, with level c. Then f satisfies the constraint if $\{x : f(x) \geq c\}$ is a convex set. In this case $\mathcal{A}$ forms a convex contour.*

Making certain level sets connected is a way of selectively smoothing certain regions of a contour plot. Requiring that the outermost contours be drawn as single lines allows the low-density regions of the plot to be smoothed without affecting the estimate in higher-density areas. See for example the heart disease data (Figure 2.2). Applying Constraint 17 to the outermost three contours would improve the qualitative appearance of the estimate with $h = 0.3$. A higher degree of smoothing can be imposed by requiring the contours to be not only contiguous, but convex as well.

The best way to implement constraints 17 and 18 will depend on the system being used to produce the contour plots. For this reason no algorithms for doing so are presented here. The plots in this thesis were produced using MATLAB (The Mathworks, Inc., 2007), and in this environment the contour plotting function returns a data structure that makes it easy to inspect individual contours to see if they are contiguous or convex[6].

Constraints 17 and 18 again demonstrate the utility of defining constraints in a black-box manner. Computer implementation of these constraints, in the form of a binary constraint-checking function, is easy. Expressing the constraint in a form suitable for mathematical programming, on the other hand, would be a daunting task.

2.3 Choice of Objective Function

The choice of objective function will influence both the nature of the optimization problem and the qualitative behaviour of the resulting density estimates. Several possibilities for $\delta(\boldsymbol{q}, \mathbf{t})$ are discussed below.

[6]The plotting function returns a set of points that define each contour. Convexity, for example, can be checked by using built-in functions to compare the area enclosed by these points to the area of the points' convex hull.

2.3.1 Objectives Based on the Adjustable Values

The adjustable values $\boldsymbol{q}$ can be considered perturbations of the target vector $\mathbf{t}$. It is natural, then, to take the objective to be a measure of distance between vectors. One option is the L_α distance defined in equation (1.7), which takes $\delta(\boldsymbol{q}, \mathbf{t})$ to be a norm of the difference $\boldsymbol{q} - \mathbf{t}$.

The choice of α can have important consequences on the performance of an estimator when the L_α distance is used. In data sharpening, for instance, α can be interpreted as controlling the tendency to sharpen by moving single points or groups of points. Setting $\alpha = 2$ discourages movement of single points through large distances, while setting $\alpha = 1$ makes the optimizer indifferent to the number of points moved. The value of α can particularly affect behaviour in the tails of the distribution, where there are few data points.

The L_α distance was used by Braun and Hall (2001) and Hall and Kang (2005) to perform data sharpening with SQP as the optimizer. Those studies found that $\alpha = 1$ gave better mean integrated squared error ($MISE$) performance in test problems, but caused problems with numerical stability, occasionally leading to non-convergence. Failure to converge was attributed to differentiability: $L_1(\mathbf{y}, \mathbf{x})$ is not differentiable in its ith dimension at $y_i = x_i$.

To improve the numerical stability of optimization, Hall and Kang (2005) proposed using a metric defined as

$$\Psi_{\tan}(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{n} \int_0^{d_i} \arctan(t)dt,$$

where $d_i = |y_i - x_i|$. The reason for using this function was to mimic the linear behaviour of L_1 away from $d_i = 0$ while maintaining differentiability at zero. In Chapter 3, a new alternative, the rounded-corners objective, will be used instead:

$$RC_\gamma(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{n} \left[\left(\frac{2}{3\gamma} d_i^2 - \frac{1}{9\gamma^2} d_i^3 \right) \mathbf{I}(d_i \leq \gamma) + \left(d - \frac{4}{9}\gamma \right) \mathbf{I}(d_i > \gamma) \right], \tag{2.4}$$

where $d_i = |y_i - x_i|$ and $\mathbf{I}$ is the indicator function. The RC objective is a twice-

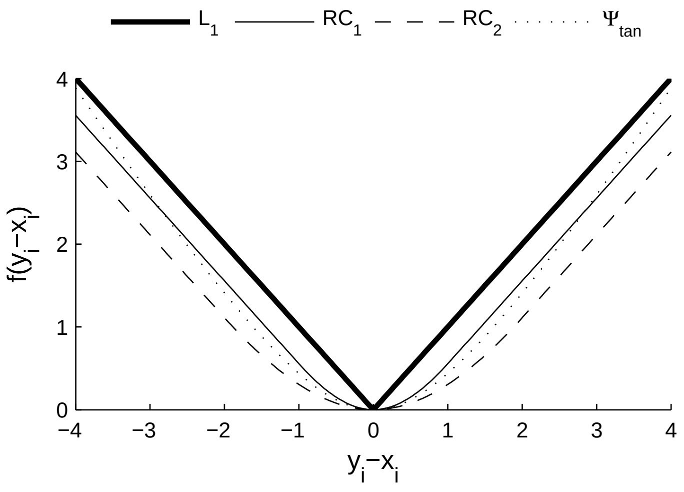

Figure 2.7: Summands of four objective functions.

differentiable continuous piecewise function of d_i. The summand of (2.4) is a convex cubic polynomial in the interval $|y_i - x_i| \leq \gamma$ and a line with unit slope (just like L_1) outside this interval. The central interval is effectively a curved, differentiable patch that replaces the corner in the usual L_1 objective. The constant γ determines the width of this interval; smaller values of γ more closely approximate L_1.

Figure 2.7 compares the summands of L_1, RC_1, RC_2, and $\Psi_{\tan}$. The RC objective achieves the same aims as $\Psi_{\tan}$, but without the need for integration. It also allows the amount of curvature at the vertex to be controlled by changing the value of γ.

The goal of shape-constrained estimation is to find a good density estimate that satisfies the constraint. If the problem is posed in terms of the adjustment vector $\boldsymbol{q}$, rather than the density estimate itself, then in some situations the set of solutions $\{\boldsymbol{q}\}$ can have a many-to-one mapping onto the set of density estimates $\{\hat{f}_{\mathbf{y}}\}$. Data sharpening has this property, for example, because the KDE is invariant to permutations of $\mathbf{y}$ while the L_α and RC objectives are not. If two solution vectors $\mathbf{y}_1$ and $\mathbf{y}_2$ are permutations of each other then they are practically equivalent; nonetheless, numerical optimization routines using objective functions (1.7) or (2.4) will consider them to be different because $\delta(\mathbf{y}_1, \mathbf{x}) \neq \delta(\mathbf{y}_2, \mathbf{x})$ in general for those objectives.

Where appropriate, the objective function can be modified to include a matching step to enforce permutation invariance on the solutions. The simplest way to match points in 1-D problems is to start with $\mathbf{t}$ sorted in ascending order and then sort any proposed $\boldsymbol{q}$ before calculating the objective function. A sorted L_α objective can then be defined as:

$$L^s_\alpha(\boldsymbol{q}, \mathbf{t}) = L_\alpha(\text{sort}(\boldsymbol{q}), \mathbf{t}) = \sum_{i=1}^{n} |q_{(i)} - t_{(i)}|^\alpha, \qquad 1 \le \alpha \le 2, \tag{2.5}$$

where $q_{(i)}$ represents the ith largest point in $\boldsymbol{q}$. The sorted version of RC can be similarly defined and denoted $RC^s(\boldsymbol{q}, \mathbf{t})$.

An optimization heuristic using only the un-sorted objective function will have no way of knowing whether a solution's objective value could be improved by re-matching its points to $\mathbf{t}$. The algorithm may fail to use promising solution paths, or may converge to sub-optimal solutions when points "cross over" each other into an un-matched state. Performing matching before evaluating the solution might improve performance and reliability of solution methods.

2.3.2 Objectives Based on Density Estimates

Another approach to choosing an objective function is to use a metric based on the constrained and pilot density estimates, $\hat{f}_{\boldsymbol{q}}$ and $\hat{f}_{\mathbf{t}}$. There are a number of suitable distance or discrepancy measures available, including integrated squared error (ISE), Kullback-Leibler divergence (KL), and total variation (TV), respectively defined as

$$ISE(\boldsymbol{q}, \mathbf{t}) = \int_{-\infty}^{\infty} (\hat{f}_{\mathbf{t}}(t) - \hat{f}_{\boldsymbol{q}}(t))^2 dt, \tag{2.6}$$

$$KL(\boldsymbol{q}, \mathbf{t}) = \int_{-\infty}^{\infty} \hat{f}_{\boldsymbol{q}}(t) \ln\left(\frac{\hat{f}_{\boldsymbol{q}}(t)}{\hat{f}_{\mathbf{t}}(t)}\right) dt, \tag{2.7}$$

$$TV(\boldsymbol{q}, \mathbf{t}) = \frac{1}{2}\int_{-\infty}^{\infty} |\hat{f}_{\mathbf{t}}(t) - \hat{f}_{\boldsymbol{q}}(t)| dt. \tag{2.8}$$

ISE is an integrated L_2 distance between estimates, while TV is an integrated L_1 distance. Note that KL (also known as relative entropy) is not a true distance, because $KL(\boldsymbol{q}, \mathbf{t}) \neq KL(\mathbf{t}, \boldsymbol{q})$.

Devroye and Lugosi (2001) examined different distance measures in a density estimation context and concluded that the TV distance has several theoretical advantages. In particular, it admits a probability interpretation: for any Borel set B, $|P_{\hat{f}_\mathbf{t}}(B) - P_{\hat{f}_\boldsymbol{q}}(B)| \leq TV(\boldsymbol{q}, \mathbf{t})$. That is, $TV(\boldsymbol{q}, \mathbf{t})$ is the maximum possible difference attainable when the same probability is calculated with the two density estimates $\hat{f}_\boldsymbol{q}$ and $\hat{f}_\mathbf{t}$.

Objectives based directly on the estimates have the advantage of being insensitive to the ordering of $\boldsymbol{q}$ and $\mathbf{t}$. On the other hand, the density-based objectives are specific to the density-estimation context, and would not apply if, for example, the constraint handling methods were used in a monotone regression problem.

2.3.3 A Likelihood Objective

A final objective function is based on the negative log-likelihood of the data under the constrained density $\hat{f}_\boldsymbol{q}$. Using this objective, the goal is to find the shape-restricted KDE that assigns greatest likelihood to the observed data. The objective function can be written as

$$LIK(\boldsymbol{q}, \mathrm{x}) = -\sum_{i=1}^{n} \ln \hat{f}_\boldsymbol{q}(x_i). \tag{2.9}$$

The LIK objective has $\boldsymbol{q}$ and $\mathbf{x}$ as its arguments, rather than $\boldsymbol{q}$ and $\mathbf{t}$ in the general framework. As such it does not strictly fit into the general framework previously proposed (except in the case of data sharpening, where $\mathbf{x} = \mathbf{t}$). The algorithms presented in later chapters all take $\mathbf{t}$ or $\hat{f}_\mathbf{t}$ as their reference point–the goal is to make the adjusted estimate as close to the pilot estimate as possible. If instead we use likelihood as the measure of success, the pilot estimate might not be the best point of reference and the algorithms might not work as well.

Despite these complications, the LIK objective is given here because of its intuitive appeal. Likelihood is also used in Section 2.4 to motivate a bandwidth selector suitable for shape-constrained estimates.

2.3.4 Visualizing the Objective Functions

Section 2.2.1 included a small example where two of eight points were moved to achieve unimodality (Figure 2.3). The same example can be used to visualize the different objective functions in two dimensions. Figure 2.8 shows this problem's objective function contours for the eight objectives defined above. Each graph in the figure shows the solution space for the problem, with each point (y_1, y_8) in the graphs representing a potential solution.

Seven of the objective functions have their minimum value at the observed data. LIK is the exception, reaching its minimum when both y_1 and y_8 are shifted slightly inward from their corresponding x values.

The L_1, L_2, and RC objectives are all convex functions of (y_1, y_8). The sorted objective L_2^s and the four density-based objectives (ISE, TV, KL, and LIK) are not—they exhibit many local optima, ridges, and plateaus. They are all symmetric around the $y_1 = y_8$ line, consistent with the symmetry of the original problem.

From an optimization standpoint, the convexity of the first three functions is attractive. All common deterministic optimization routines require a convex objective function in order to reliably find a global optimum. Nevertheless, it would be better if the choice of objective was not motivated by optimization convenience, and in any given situation it may be desirable to use one of the non-convex objectives for theoretical or practical reasons.

2.4 Bandwidth Selection

The proposed method for handling shape constraints involves constructing a pilot estimate and subsequently adjusting its shape. An important question is how the bandwidth of the pilot estimator should be chosen. Bandwidth selection is a difficult problem even in the absence of shape constraints, and there is a large literature on the subject, with many proposed selection rules. No single rule dominates all the others. The best bandwidth selector for a given case depends on the true shape of the density being estimated, and on how one measures estimation quality.

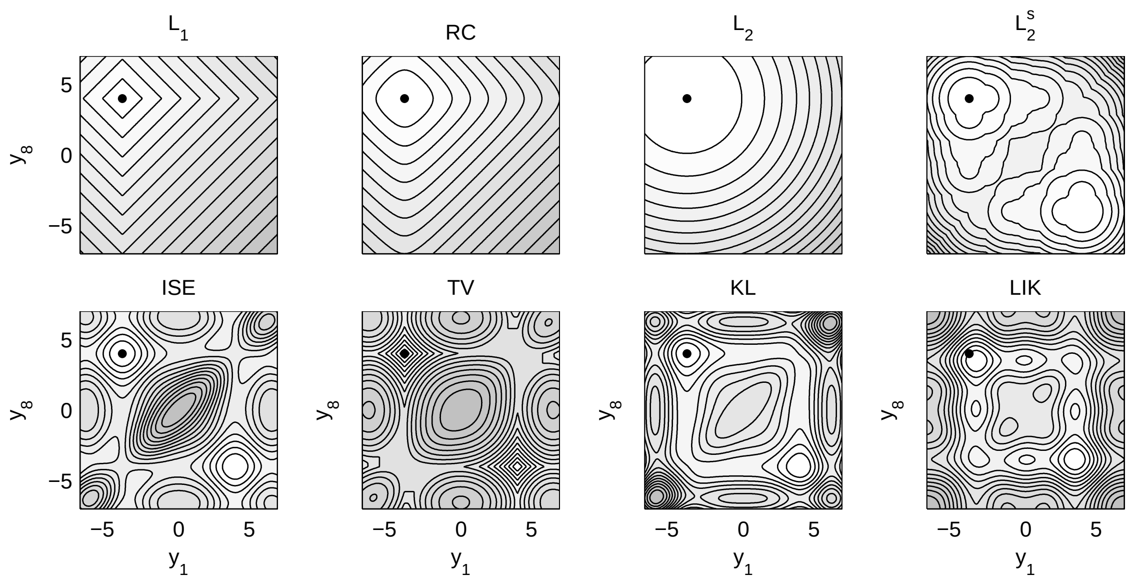

Figure 2.8: Contour plots of eight objective functions over the search space for the example of Figure 2.3. Higher (worse) regions are shaded in gray. The dots show the unsharpened solution $(-4, 4)$. All axes have the same scaling.

Optimal bandwidth selection becomes even more difficult when constructing a shape constrained estimate, because the bandwidth that is best for an unconstrained KDE is not necessarily best when constraints are added. The complexity of the adjustment process makes theoretical development of an optimal shape-constrained bandwidth selector a difficult task. Rather than attempting a theoretical treatment here, two practical solutions are proposed. The first is to use a standard selection rule for the pilot estimator. Justification for this approach, and a review of some common selection rules, are given in Section 2.4.1. The second solution is to choose h such that a type of likelihood is maximized after shape adjustment. This option is described in Section 2.4.2, where it is also compared to the method of likelihood cross validation. A simulation study comparing the bandwidth selection methods is deferred to Appendix A.

2.4.1 Using a Standard Pilot Bandwidth

The simplest way to choose the bandwidth in shape-constrained estimation is to ignore any shape restrictions and choose an h that yields a good unconstrained density estimate. Shape adjustment can then be applied to this good pilot density to enforce constraints. Although this strategy does not account for the adjustment process, it can be justified in two ways.

The first justification is an asymptotic argument that holds when the assumed constraints are valid for the true density. If the KDE with the chosen bandwidth is a consistent estimator and the true density satisfies the constraints, then it should become unnecessary to do any adjustment for n sufficiently large. In that case a bandwidth that is suitable for the pilot estimate should be suitable for the adjusted one as well (since the two estimates are the same). Hall and Kang (2005) showed this rigorously for the data sharpening case: as n increases, the sharpened estimator modifies the KDE only in the ever-smaller regions where the constraint is violated.

A second justification of the simple approach to bandwidth selection, one that applies to finite samples, is that the statistical properties of shape-constrained estimators should be less sensitive to bandwidth choice than unconstrained ones. This tendency has been observed in empirical studies (Braun and Hall, 2001; Hall and Kang, 2005) where it was found that data sharpened estimators do have different optimal bandwidths than their unsharpened counterparts, but also that a wide range of bandwidths near the optimum give good results. It appears that restricting the density estimate to a particular class of shapes reduces the impact of sample-to-sample variation and thereby makes it less critical to choose an ideal value for h.

If we accept this approach to bandwidth selection, there are still many available bandwidth selection rules from which to choose. Several of the most important options are briefly reviewed here. For more details on the results below, see Wand and Jones (1995, ch. 3), Silverman (1986, sec. 3.4), Scott (1992), or Wasserman (2006, ch. 6).

The normal-scale bandwidth is an estimate of the optimal bandwidth in the asymptotic mean integrated squared error (AMISE) sense if the data-generating density is $N(\mu, \sigma^2)$. For the Gaussian KDE, this bandwidth choice is

$$h_{NS} = 1.06 \frac{\hat{\sigma}}{n^{1/5}}, \tag{2.10}$$

where $\hat{\sigma}$ is an estimate of σ (such as the sample standard deviation). Another AMISE-motivated choice of h is the oversmoothed bandwidth selector, which is

$$h_{OS} = 1.14 \frac{\hat{\sigma}}{n^{1/5}} \tag{2.11}$$

for the Gaussian KDE. It is meant more as a point of reference than as a good bandwidth choice in itself, since it estimates an upper bound on the AMISE-optimal h. As its name implies, h_{OS} will typically produce overly smooth density estimates. Comparison of (2.10) and (2.11) shows that h_{NS} is about 93% of the upper bound estimate, so h_{NS} can also be expected to oversmooth the KDE unless the true density is nearly as smooth as the normal pdf.

A more sophisticated selection method, with better performance across different true densities, is the "two-stage plug-in" selector of Sheather and Jones (1991). For the Gaussian KDE, this bandwidth is

$$h_{SJ} = \left(\frac{0.282}{\hat{R}_4 n} \right)^{1/5}, \tag{2.12}$$

where $\hat{R}_j$ is an estimate of $R_j = \mathrm{E}[f^{(j)}(X)]$ and $f^{(j)}$ is the jth derivative of the unknown density f. The complexity of this method lies in determining $\hat{R}_4$. The asymptotically optimal bandwidth for estimating R_j depends on R_{j+2}. To obtain h_{SJ}, a crude estimate of R_8 is found using the normal-scale bandwidth. This estimate is then used to find $\hat{R}_6$ and finally $\hat{R}_4$. The procedure is described in more detail by Wand and Jones (1995, p. 72).

Cross-validation methods can also be used to choose a bandwidth in kernel density estimation. One approach, known as least squares cross-validation, selects

$$h_{LSCV} = \underset{h \geq 0}{\text{argmin}} \int \hat{f}(x;h)^2 dx - \frac{2}{n} \sum_{i=1}^{n} \hat{f}_{-i}(x_i;h) \tag{2.13}$$

as the optimal bandwidth, where $\hat{f}(x;h)$ is the standard KDE and $\hat{f}_{-i}$ is the *leave-one-out* estimator, the KDE formed using all of the data except x_i. It can be shown that, up to a constant free of h, the quantity being minimized in (2.13) is an unbiased estimator of the mean integrated squared error between $\hat{f}$ and f.

Another type of leave-one-out cross-validation is likelihood cross-validation. The bandwidth selected by this method is

$$h_{LCV} = \underset{h \geq 0}{\text{argmax}} \prod_{i=1}^{n} \hat{f}_{-i}(x_i;h). \tag{2.14}$$

The ith term in the product on the right hand side of (2.14) is the likelihood of the KDE with x_i left out, evaluated at x_i. The leave-one-out approach is necessary to ensure that h_{LCV} is nonzero. If the x_i were not left out, the product $\prod_{i=1}^{n} \hat{f}(x_i;h)$ would approach infinity as $h \to 0$.

2.4.2 Maximizing a Pseudo-Likelihood

Despite the preceding justification for selecting the bandwidth prior to shape adjustment, a bandwidth selection procedure designed specifically for shape-constrained estimators would be welcome. Optimal bandwidth selection for this situation is an open problem. Rather than try to solve the problem rigorously here, a bandwidth choice with some some intuitive and practical appeal is proposed. The new bandwidth is denoted h_{ML}, and is the maximizer of a quantity resembling a likelihood.

The proposal may be motivated by starting with the likelihood cross-validation bandwidth (2.14). There are two attributes of h_{LCV} that are particularly important:

1. It promotes density estimates that place higher density on the observed data points.

2. It severely penalizes any density estimate that places small probability mass on an observed data point, because any h value yielding a negligible density over a single point will drive the product in equation (2.14) close to zero.

Attribute 1 is reasonable, and is one of the motivations behind maximum likelihood estimation in general. Attribute 2 has both positive and negative consequences. Its positive consequence is preventing any density estimates that place negligible probability mass near an observed value. Its negative consequence is sensitivity to outliers (Scott and Factor, 1981). Because the density estimate must not be too close to zero at any data point, outliers will have disproportionate influence on h_{LCV}, tending to cause larger h values to be selected.

If one were to apply likelihood cross-validation to the shape-adjusted density estimator $\hat{f}_{\boldsymbol{q}}$, the following bandwidth selector would suggest itself:

$$h_{LCVa} = \underset{h \geq 0}{\operatorname{argmax}} \prod_{i=1}^{n} \hat{f}_{\boldsymbol{q}_{-i}}(x_i; h), \tag{2.15}$$

where the notation $\boldsymbol{q}_{-i}$ indicates that the ith data point is withheld before determining the adjustment. Implementing (2.15) would be computationally intensive. In a line search over the possible values of h, the adjustment procedure would need to be carried out n times for each candidate h. Outlier sensitivity similar to h_{LCV} could be expected, since a larger h value would still be required for the case when the outlying x value is left out.

The proposed bandwidth selector for shape-adjusted KDEs attempts to retain the desirable characteristics of h_{LCV}, with reduced outlier sensitivity and reduced computational burden relative to h_{LCVa}. The proposed selector is

$$h_{ML} = \underset{h \geq 0}{\operatorname{argmax}} \prod_{i=1}^{n} \hat{f}_{\boldsymbol{q}}(x_i; h), \tag{2.16}$$

where $\hat{f}_{\boldsymbol{q}}(x_i; h)$ is the shape constrained estimator with bandwidth h. The product in the right hand side of (2.16) is the likelihood of $\mathbf{x}$ under the density $\hat{f}_{\boldsymbol{q}}$, if we take $\boldsymbol{q}$ to be a fixed vector (rather than what it truly is, a function of $\mathbf{x}$ and h). This resemblance to a maximum likelihood estimate motivates the notation h_{ML}.

The product in (2.16) does not involve the leave-one-out approach; the estimate $\hat{f}_{\boldsymbol{q}}$ only needs to be worked out once per candidate h value. The existence of the shape constraint makes it unnecessary to withhold points to obtain a reasonable bandwidth, because for most constraints of practical interest, the product $\prod_{i=1}^{n} \hat{f}_{\boldsymbol{q}}(x_i; h)$ approaches zero, not infinity, as $h \to 0$ when the constraint is enforced. Eliminating the cross-validation element from the selector causes an approximately n-fold reduction in computation versus h_{LCVa}, and should also reduce outlier sensitivity because the outlying points never need to be "left out."

Appendix A provides details of a simulation study that compares h_{ML} to h_{SJ} and h_{LCVa}. The results suggest that h_{ML} provides a reasonable bandwidth choice. While it still has some sensitivity to outliers, this sensitivity is considerably reduced relative to likelihood cross-validation.

Chapter 3

A Greedy Algorithm for Data Sharpening

Heuristic optimizers operate by iteratively updating one or more candidate solutions. Each update is a move that shifts a solution from one location to another in the solution space. A major task of algorithm design is to define the set of possible moves a candidate solution can make at any stage of the search, and a means of selecting one move over the others. An algorithm is called *greedy* if, at each iteration, the move that causes maximal improvement in the objective function is selected.

Greedy algorithms are a convenient first choice when developing heuristics, because they are often conceptually simple and computationally fast. The use of locally optimal moves at each iteration maximizes the short-term improvement of the search, but also makes the search prone to be trapped in local optima. It is usually possible to improve the overall performance of a greedy heuristic by searching less aggressively for good solutions at each move.

This chapter introduces a greedy algorithm for shape-constrained density estimation by data sharpening. It is a deterministic algorithm that executes quickly, but owing to its greedy design it only works well for less stringent constraints like unimodality. The algorithm is described below, and its properties are examined through examples and simulations. Afterwards it is shown how it can be incorporated into a metaheuristic known as iterated local search (ILS), that adds randomness to the search and should make the algorithm capable of solving more complex problems.

3.1 The `improve` Algorithm

The new algorithm described in this chapter applies to data sharpening problems, where the target is the data $\mathbf{x}$, and the candidate solution is a sharpened data vector $\mathbf{y}$. It carries out moves of the solution $\mathbf{y}$ (which are points in the n-dimensional solution space) by sequentially moving its elements y_i (which are scalar points in the data space). Each move of a sharpened data point y_i is done in a greedy manner. The algorithm is called `improve`, because it takes a feasible guess solution as input and returns another feasible solution with improved objective function as output.

3.1.1 Algorithm Description

The proposed procedure is listed as pseudocode in Algorithm 3.1. Search starts from a user-supplied initial guess solution, $\mathbf{v}$, that is feasible. From the initial solution $\mathbf{y} = \mathbf{v}$, each y_i is moved to be closer to its corresponding unsharpened (target) data point x_i. Every such move will reduce the $L_\alpha(\mathbf{y}, \mathbf{x})$ objective function, but no point may be moved in a way that causes constraint violations. In this way feasibility is guaranteed throughout. The algorithm cycles through the elements of $\mathbf{y}$ for as long as feasibility-preserving improvements can be made. The procedure is greedy in the sense that each sharpened data point is moved individually to improve the objective function as much as possible, without consideration of how the current move will impact future moves of other points.

Step one in the search is initialization. The original data $\mathbf{x}$ is sorted in ascending order, and the solution is initialized to $\mathbf{y} = \mathbf{v}$. The initial solution may be a simplistic choice, but it must satisfy the constraint. If the kernel function itself satisfies the operative shape constraints, an easy way to initialize is to let $\mathbf{v}$ have all of its data points at the same location. This will cause the KDE to have the same shape as the kernel function. When using this initialization strategy, the default choice for the kernels' location is the location of the highest mode in the unconstrained estimate. In other words, if m_0 is the location of the highest mode, we set $\mathbf{v} = m_0\mathbf{1}$. This starting solution has been found to perform adequately in most circumstances.

The second step is to prepare for moving the y_i. The target values x_i, $i = 1, \ldots, n$ will also be called the *home* positions for their corresponding y_i values. The solution is improved during the algorithm by moving each y_i toward home. If a point reaches home, it stops moving. If the constraint prevents a point from moving closer to home, that point is said to be *pinned*.

In preparation for moving the points, $\mathbf{y}$ is first sorted, to produce a sensible matching to $\mathbf{x}$. After this, each point is examined to determine whether or not it is *moveable*. A point is considered moveable if it is neither pinned nor at home. The total number of moveable points is M. The algorithm terminates when $M = 0$; at this point no further moves can be made without either worsening the solution or violating the constraint.

Step three in the algorithm is the core of the method—a *sweep* or *pass* through all M moveable points in $\mathbf{y}$. In each pass, every moveable point is moved closer to its target position, or left in place if no feasible move is found. The movement of each point is done by grid search over the interval $[y_i, x_i]$. Grid search is performed by dividing the search interval into S steps. If any moves are made in a pass, S is left unmodified and another pass begins after re-sorting $\mathbf{y}$ and re-counting the number of moveable points. If a complete sweep results in no moved points, the value of S is doubled before the next pass, permitting smaller moves to be made on a finer grid.

An important feature of the algorithm is that S is initialized to 1. This means that during the first sequence of passes through the data, there is an attempt to move points all the way home directly in one step. Doing so saves computation time since in many cases a large portion of the points can move home immediately without violating the constraint. By successively doubling S only when moves cannot be made, more thorough searches are deferred until the later stages, when a small number of points are being moved up against the constraint boundary. This strategy reduces the greediness of the method, preventing points from becoming pinned too soon and thereby conferring a considerable performance improvement.

Algorithm 3.1: A greedy data sharpening algorithm (`improve`).

```
Input: A feasible initial guess, $\mathbf{v}$; the data, $\mathbf{x}$; a bandwidth, $h$
Output: A feasible solution $\mathbf{y}$ with $L_\alpha(\mathbf{y},\mathbf{x}) \leq L_\alpha(\mathbf{v},\mathbf{x})$
Initialize
Set $\mathbf{y} \leftarrow \mathbf{v}$.
Let $S$ be the number of grid search steps. Set $S \leftarrow 1$.
Prepare for the first sweep
Sort $\mathbf{y}$.
Find the set of moveable points ($M$ of them).
while $M > 0$
    Sweep through the points
    for each moveable point
        Use grid search with $S$ steps to move the point closer to home, while
        maintaining feasibility.
    Prepare for the next sweep
    if at least one point has moved
        Sort $\mathbf{y}$.
        Find the set of moveable points ($M$ of them)
    else
        Set $S \leftarrow 2S$
```

Note also that the sorting step is performed before every pass through the data. Re-sorting the points at each step improves the performance of the algorithm because sometimes points cross over one another, in which case both will be closer to home, and the objective function will be decreased, if they switch target points.

These ideas are illustrated in Figure 3.1, which shows how the solution develops over three passes for a small example with only five data points. The constraint in this example is unimodality. The intermediate positions of the sharpened points are shown after each pass, and a line joins each point to its target. Each line is labeled to show the status of its corresponding sharpened point. Lines labeled with numbers correspond to moveable points, and the numbers indicate the order in which points are to be moved. Lines labeled with `h` correspond to points at home, while those labeled with `p` correspond to points that are pinned. After the first pass (the upper right plot in the figure), the sorting step has caused two points to switch targets. The thick grey lines indicate the points' new targets after re-matching. In this example

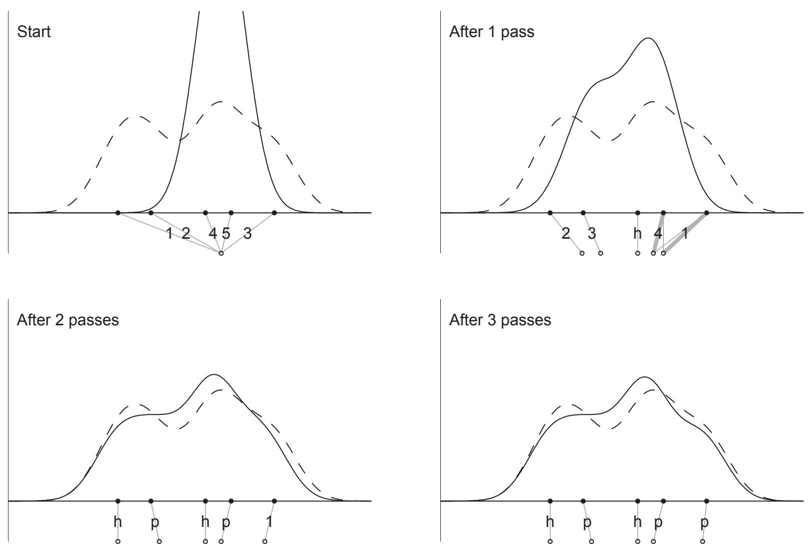

Figure 3.1: A small example illustrating the greedy sharpening method. Solid/dashed lines show the sharpened/unsharpened estimates. Open/filled circles show the sharpened/unsharpened data. Grey lines join each unsharpened point to its target and indicate the status of the point.

the search terminates after three passes, with three points pinned and two points at home.

3.1.2 Implementation Details

The steps just described omitted several details that are important when implementing the algorithm on a computer. These details are briefly discussed below. More information on the development of the algorithm is reported elsewhere (Wolters, 2009).

Initial Solution

Just like SQP, the greedy algorithm is sensitive to its starting point $\mathbf{v}$. Supplying a poorly-chosen starting value will result in premature termination of the algorithm at a low-quality solution. The recommended initial value (all points at the highest unsharpened mode) is pragmatic because it typically allows many points to move directly to their home position in the early stages of the search, with the estimate slowly moving outward toward the tails as search progresses. Nevertheless, to reduce the risk of initialization dependence, one could try multiple starting points and keep only the best solution found—perhaps by using $\mathbf{v} = c\mathbf{1}$ and letting c vary over the range of $\mathbf{x}$. Such possibilities are not considered here because of the good general performance of the default starting choice.

Sweep Order

Each sweep through the data is done in descending order of distance from home; that is, the points with the greatest value of $|y_i - x_i|$ are moved first. If all points are started near the center of the distribution, this has the effect of moving points toward the tails first, and then moving interior points that are closer to home. During algorithm development this sweep order was found to have performance and speed advantages over alternative orderings.

Feasibility Checking

The feasibility-preserving nature of the algorithm makes it necessary to perform a large number of feasibility checks (to verify that $\mathbf{y} \in \mathcal{C}$, or in the general notation, that $\mathcal{I}(\hat{f}^M_{\mathbf{y}}) = 1$). Feasibility must be checked at every step in the grid search, for every moveable point at each iteration. Evaluating $\mathcal{I}(\hat{f}^M_{\mathbf{y}})$ therefore accounts for most of the computational cost of the method. The `improve` function will execute quickly only if the estimator itself can be evaluated quickly. Fortunately, the KDE can be evaluated with high speed and accuracy using a binned kernel density approximation (Wand and Jones, 1995, Appendix D.2).

Determining the Status of Each Point

Before each pass, every point is evaluated to determine whether it is home, pinned, or moveable. Each of these states is defined computationally using a numerical tolerance, τ:

$$\begin{aligned}
y_i \text{ is home} \quad &\Leftrightarrow \quad |x_i - y_i| \le \tau && (3.1)\\
y_i \text{ is pinned} \quad &\Leftrightarrow \quad \text{setting } \; y_i := y_i + \operatorname{sgn}(x_i - y_i)\tau \; \text{ renders } \mathbf{y} \text{ infeasible} && (3.2)\\
y_i \text{ is moveable} \quad &\Leftrightarrow \quad \begin{cases} |x_i - y_i| > \tau, \quad \text{and} \\ \mathbf{y} \text{ remains feasible when } \; y_i := y_i + \operatorname{sgn}(x_i - y_i)\tau. \end{cases} && (3.3)
\end{aligned}$$

Statements (3.1) through (3.3) mean, respectively, that a point is home when it is within τ of its target; it is pinned when a move of size τ toward home causes a constraint violation; and it is moveable when it is neither pinned nor at home.

The default value of τ is 10^{-4}, though it should be adjusted to be suitable for the scale of the data. Setting τ to be 4 or 5 orders of magnitude smaller than the range of $\mathbf{x}$ is sufficient. Making τ too small will increase run time, though the final density estimates will not be noticeably affected. Making τ too large will cause the algorithm to terminate too soon, degrading the performance of the estimator.

Design of the grid search

The goal of each grid search step is to move the current point y_i closer to its target x_i without violating the constraint. There are S candidate points along the interval $[y_i, x_i]$. Rather than searching all S steps, the grid search is conducted by stepping out from y_i along the grid, until feasibility is lost. After this the last feasible point is chosen. This procedure will not always find the feasible grid point closest to x_i, but it makes the overall search much more efficient by eliminating many fruitless constraint checks.

The nature of the shrinking-grid search is illustrated in Figure 3.2 by looking at a single point, y_i, over six passes. For each pass, the interval $[y_i, x_i]$ is shown with a grid of S steps superimposed. When the point is moveable, but cannot step out along the grid, the grid is made more fine by doubling S. As long as the point remains moveable, each pass results in either a successful step out or a doubling of the grid. Search terminates when the point ceases to be moveable. Two facts are not clearly depicted in the figure: i) the grid steps are doubled only when none of the n points

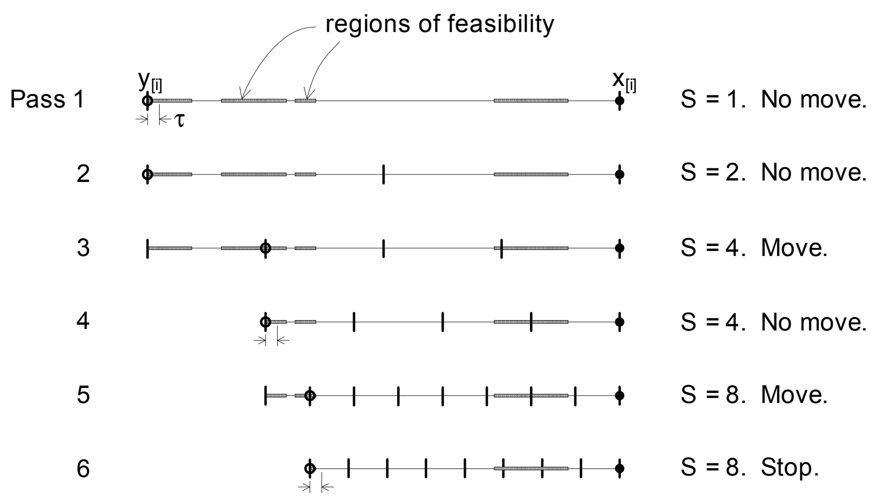

Figure 3.2: A schematic illustration of the grid search for a single point.

move; and ii) the feasible region or target x might change between passes, since they can be changed by the movement of other points.

Sorting memory

On rare occasions the configuration of the points could lead to cyclic behaviour caused by the sorting step. For example, two moveable points could exchange targets repeatedly and thereby never reach a pinned state. To prevent this, a list of all previous orderings is kept in memory, and new orderings are only accepted if they have not been visited previously. The memory requirements for this control are not problematic, since the number of passes used is typically small (on the order of 100 passes for moderate-sized data sets). Eliminating cycling of the orders ensures that a final order will eventually be reached. Once the ordering is fixed, the algorithm is guaranteed to terminate (M will eventually equal zero), because every point will move toward its target until it is either pinned or at home.

The bivariate case

The greedy algorithm involves moving points in turn toward their targets. These steps can be applied to two-dimensional points, so the same algorithm can be applied almost unchanged to bivariate problems (and, in principle, to higher dimensions as well).

The only aspect of the greedy algorithm that does not translate directly from univariate to bivariate problems is the sorting step that occurs between passes through the data. In higher dimensions it is not clear how to choose the best matching of sharpened to unsharpened points. The points cannot be "sorted," since a total ordering property can no longer be exploited.

The following matching procedure is proposed for bivariate data. The sharpened and unsharpened data are both given a location and scale transformation such that their componentwise means and variances are zero and one, respectively. Then, the points are matched to each other in a greedy way, by recursively letting those two unmatched points separated by the smallest Euclidean distance be the next matched pair.

3.2 Examples

The `improve` function was applied to both of the illustrative examples introduced in Section 2.1. The wind speed data is used to compare the greedy algorithm to optimization by SQP, and also to explore the use of the proposed h_{ML} bandwidth (2.16). The heart disease data is used for a simple demonstration of bivariate unimodality. Additional examples are reported by Wolters (2012).

3.2.1 Wind Speed Data

The simplest constraint to consider for the wind speed data is unimodality. As mentioned in Chapter 2, the unimodality constraint is amenable to optimization by sequential quadratic programming, so these data provide an opportunity to compare estimates obtained using SQP with those obtained using `improve`.

Certain implementation decisions had to be made to run SQP. Because it requires a mode location to be specified, the optimizer was run 20 times with different mode choices, evenly spaced between the first and third quartiles of the data. The best of these 20 solutions was used as the final estimate. Each SQP run was carried out using NAG routine e04wd (Numerical Algorithms Group, 2009). The unsharpened

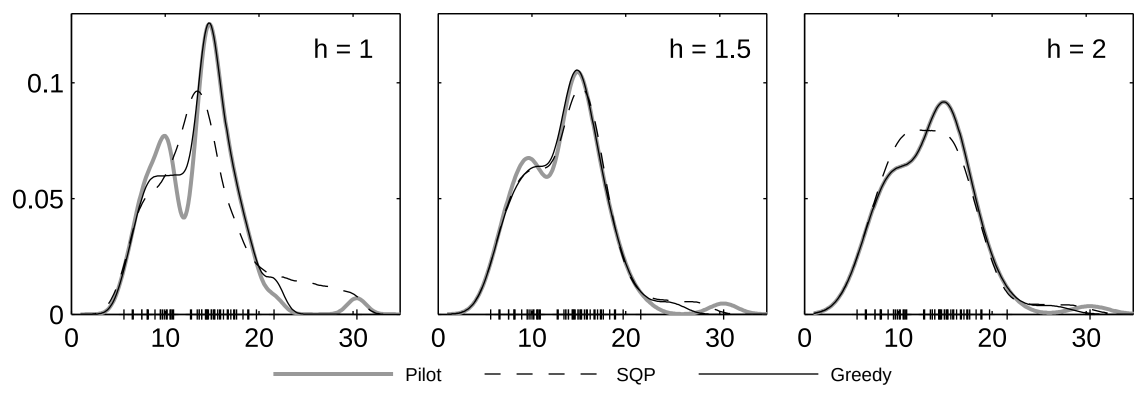

Figure 3.3: Results of unimodal density estimation on the wind speed data, using SQP and the greedy algorithm.

data, $\mathbf{x}$, was used as the starting point for the SQP search. The $L_1(\mathbf{y}, \mathbf{x})$ objective function caused numerical problems for the NAG routine, so the $RC_{0.01}$ objective function (equation 2.4) was used instead. By setting a small value $\gamma = 0.01$ in the objective, behaviour nearly equivalent to L_1 was obtained with significantly improved numerical stability. Because the $RC_{0.01}$ and L_1 curves are so similar, no distinction between the two objectives will be made in the following discussion.

Figure 3.3 shows results of both the greedy method and SQP for bandwidths $h =$ 1, 1.5, and 2. In each plot of the figure, the pilot estimate is shown with the SQP and greedy estimates superimposed. The $h = 1$ and $h = 2$ cases provide instances of undersmoothing and oversmoothing, respectively, while the $h = 1.5$ case represents a reasonable bandwidth choice (for reference, $h_{SJ} = 1.55$ for these data).

The undersmoothed case is the most difficult from the optimization standpoint, because the feasible region of the search space is smallest. The greedy algorithm essentially ignored the outlying point in the right tail. This happened because the starting solution puts all points at the highest unsharpened mode, causing the points to move outward as search progresses. It was therefore not possible for the greedy moves to shift a point far into the tail without violating the constraint. The SQP solution, on the other hand, covers the outlying point, but in order to do so many points had to be moved to fill in the gap between the outlier and the main part of the density. The solution is clearly a poor local optimum. The objective values for the greedy and SQP solutions are 15.1 and 84.2, respectively.

The two methods perform more comparably in the $h = 1.5$ case. Both estimates are equal to the pilot estimate over most of the density's support. The SQP estimate stretches farther into the tail, however, with a slightly lowered main peak to compensate. The objective values in this case are 9.3 for `improve`, and 19.6 for SQP. Again, the heuristic method found the preferable solution.

Looking finally at the oversmoothed $h = 2$ case, we see that SQP again had difficulty duplicating $\hat{f}^{\circ}$ at its main peak. This is surprising, because the pilot estimate itself is almost unimodal at this bandwidth, which should make the problem easier. Nevertheless SQP was unable to find a good optimum. The objective values were 4.1 for `improve`, and 18.6 for SQP.

Figure 3.4 shows how the two methods compare when using the h_{ML} bandwidth of Section 2.4.2. Recall that this selector requires performing optimization over a range of possible h values. Because each optimizer could find different solutions for any particular bandwidth, it is necessary to determine h_{ML} separately for each method. It happens that in this case SQP and `improve` choose nearly equal values for h_{ML}: 2.06 and 2.03, respectively. These values are large, and probably oversmooth the density estimate in the central portion of the range. The large choice of h_{ML} can be attributed to the outlier sensitivity inherent in the selection method (as discussed in Section 2.4 and Appendix A). The two unimodal estimates have nearly the same shape, but as with the other bandwidths, the SQP estimate extends farther into the right tail, while the greedy solution matches the pilot estimate more closely everywhere else. The objective function values—which are only comparable because the bandwidths are so similar—are 4.0 for `improve` and 8.3 for SQP.

The greedy algorithm obtained reasonable unimodal density estimates at all bandwidths considered, suggesting it is a suitable method for unimodal density estimation (further evidence for this claim is given in the simulations of the next section). The limitations of the method become more apparent when considering more strict constraints, however.

Figure 3.5 shows the estimates obtained by `improve` for the type 1 bell-shaped

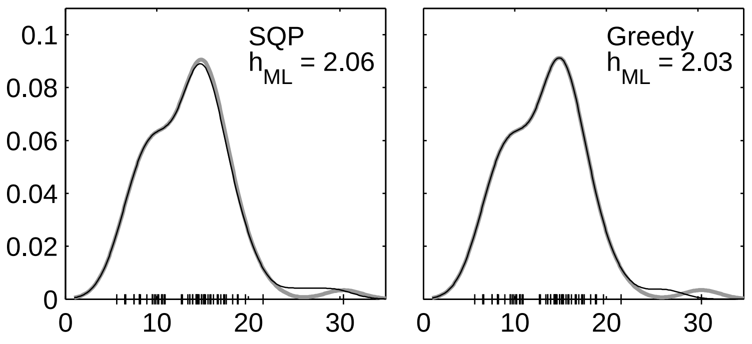

Figure 3.4: Results of unimodal density estimation on the wind speed data, using SQP and the greedy algorithm with the h_{ML} bandwidth. In both graphs, the thick grey line is the pilot estimate, and the thin black line is the constrained estimate.

constraint (Constraint 5), for four different bandwidths (1, 1,5, 2, and h_{ML})[1]. For the two smallest bandwidths in particular, the constrained estimates fail to provide a good match to the pilot estimate. In all cases, the bell-shaped estimate has a higher main peak than the pilot density. This is because the default starting solution places all sharpened points at the same location. With the more difficult bell-shaped constraint, it is not possible for individual points to move out into the tails without violating the constraint. To find a feasible solution closer to the target $\mathbf{x}$, it would be necessary to move multiple points at once, or to move points outside the search interval $[y_i, x_i]$. This is a limitation of the greedy design of the algorithm.

The pseudo-likelihood bandwidth selection procedure chose a value of $h_{ML} = 2.39$ with this constraint, a bandwidth even larger than was chosen for unimodality. In this instance the large bandwidth can be attributed not only to outlier sensitivity, but also to the poor performance of the optimizer. If the optimizer cannot find good solutions at smaller h values, then larger bandwidths will be required to achieve a maximum likelihood. The objective function values for the four cases in Figure 3.5 are, from left to right, 46.1, 26.8, 20.4, and 19.4. It will be shown in the next chapter that better solutions do in fact exist for each case.

[1]The SQP solutions are not compared for this constraint because the time required to set up and solve the mathematical programming problem is prohibitive; it involves an outer optimization loop to find the best locations for both of the density's inflection points.

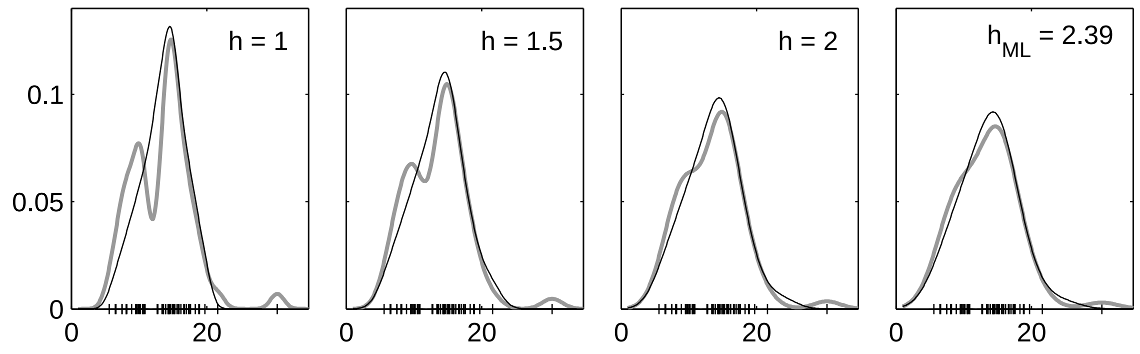

Figure 3.5: Results of bell-shaped (type 1) density estimation on the wind speed data, using the greedy algorithm with four bandwidths. In all graphs, the thick grey line is the pilot estimate, and the thin black line is the constrained estimate.

3.2.2 Heart Disease Data

The heart disease data can be used to demonstrate data sharpening in two dimensions. Bivariate unimodality (Constraint 14) is taken as the operative shape restriction.

The pilot density and its unimodal adjustment are plotted in Figure 3.6. For this example, the Gaussian product kernel was used, with the normal-scale bandwidth choice (equation 2.10). Because the data have been standardized, this selector chooses the same bandwidth of $h_{NS} = 0.43$ for both variables. The estimates were calculated using a bivariate binned KDE approximation at a 200×200 grid of points, and the same grid was used to check the validity of the constraint (as described in Algorithm 2.4).

The pilot estimate has five local maxima and one local minimum. All but the maximum that occurs at the central mode of the density are small variations occurring in low-density regions. As seen in the right plot of the figure, the greedy algorithm is able to eliminate these spurious maxima and minima by shifting several points. The high-density regions of the estimate are not altered.

3.3 Simulation Studies

A simulation study was performed to compare the performance of three optimization options: the greedy algorithm, SQP, and a combined optimizer, where the greedy

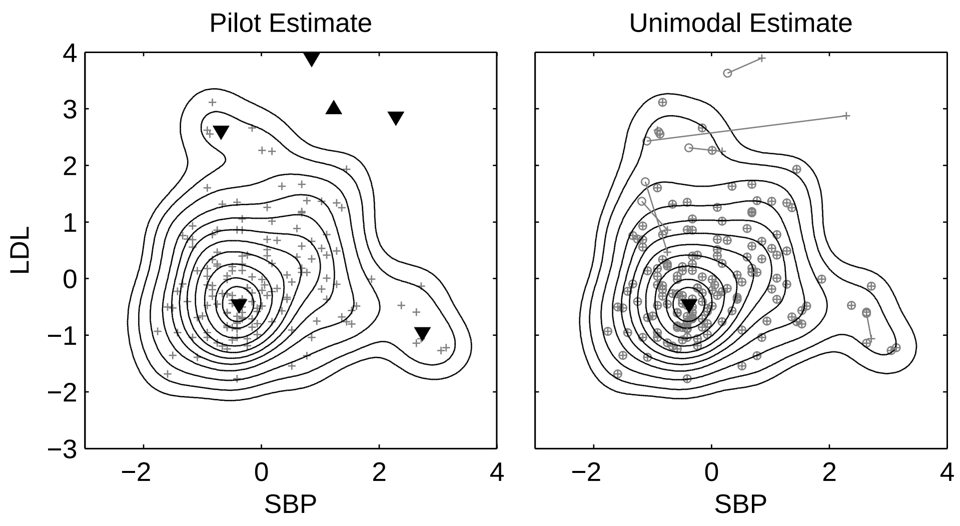

Figure 3.6: Pilot estimate and unimodal estimate for the heart disease data, using the greedy algorithm to perform data sharpening. The original and sharpened data are shown by crosses and circles respectively. Lines join the sharpened points to their target locations. Local maxima are indicated by ▼ and local minima by ▲.

solution is used as the starting point for an SQP search. All SQP runs were done in the same manner as the wind speed estimates of the previous section.

3.3.1 Study Design

The three optimization methods were compared across 12 different test cases. The test cases consisted of all combinations of two target densities, three sample sizes, and two bandwidths.

Target densities: the t distribution with three degrees of freedom, and a three-component normal mixture distribution. See Figure 3.7.

Sample sizes: 25, 50, and 100.

Bandwidths: $0.75h_{SJ}$ and h_{SJ}, where h_{SJ} is the Sheather-Jones direct plug-in bandwidth defined in (2.12).

The constraint of interest for all simulation runs was unimodality.

The target densities correspond to test densities 2 and 4 of Hall and Huang (2002). The t_3 distribution is challenging because its heavy tails result in outliers, and corresponding spurious modes, in many samples. The mixture distribution has a large,

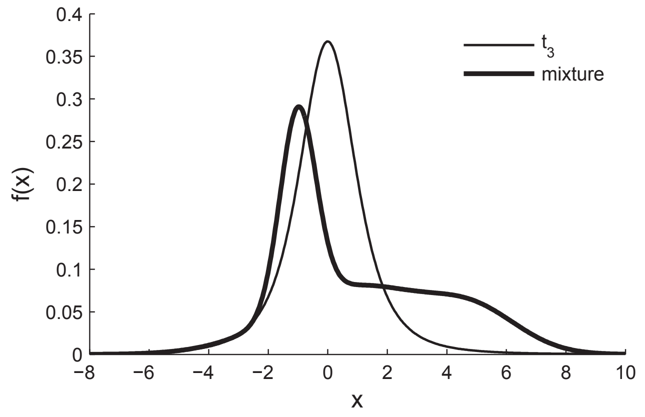

Figure 3.7: True densities used in the simulation. The mixture density is composed of $N(-1, 0.6^2)$, $N(1, 2.5^2)$, and $N(5, 1.5^2)$ components in proportions 0.35, 0.5, 0.15.

nearly flat shoulder to the right of its mode, which produces a variety of multimodal shapes in samples. The two bandwidth levels were chosen to influence the problem difficulty rather than to achieve optimal estimation performance. Setting $h = h_{SJ}$ produces smoother estimates that are easier to sharpen, while $h = 0.75h_{SJ}$ produces more separated peaks and reduces the size of the feasible set $\mathcal{C}$, making optimization harder.

For each target density, 250 data vectors of each sample size were drawn from the target distribution. To avoid trivial cases where no sharpening was necessary, a rejection step was included when generating samples. Any sample producing a unimodal unsharpened estimate was replaced with a new sample until a multimodal estimate was obtained.

For each generated $\mathbf{x}$, all three optimizers were run on the same data using both bandwidths. All runs used the data sharpened Gaussian KDE $\hat{f}^M_{\mathbf{y}}(x)$ as the estimator, and the response of interest was the $L_1(\mathbf{y}, \mathbf{x})$ objective (referred to as the *sharpening distance*)[2]. In all, 9000 optimizations were performed (12 cases, three optimizers, and 250 replicates).

[2]As with the wind speed examples, the $RC_{0.01}$ objective was used with SQP, but given its similarity to L_1, we will treat all runs as if they used the same objective function, for ease of exposition.

3.3.2 Convergence and Run Time

For the present discussion an optimizer is defined to have converged if it reaches any of its normal stop conditions and returns a feasible solution. Table 3.1 shows the proportion of runs converging, and the median run time, for all three methods across simulation cases.

The greedy algorithm converged for all simulation runs, because it is designed to always return a feasible solution. Sequential quadratic programming had some failures to converge in seven of the twelve test conditions, including all six cases based on the t_3 distribution. In those cases the proportion of runs converging varied from 84 to 96 percent, with with larger sample sizes having lower percentages[3]. Note that for SQP to record a failure, the algorithm must fail to return a feasible solution at all 20 candidate mode points attempted.

The combined algorithm (where SQP was started from the greedy solution) had much improved convergence proportions compared to SQP. Only three of the test cases had any failures, and even in these three cases only one or two of the 250 replicates failed to converge. This indicates the importance of choosing a good starting solution, and suggests that the greedy algorithm could at least be used as a way to generate starting points for SQP.

The run time results show a clear advantage of greedy over SQP, with greedy runs taking a fraction of a second while the median SQP run time ranged from four to 40 seconds depending on the case. For all cases, using the greedy solution as a starting point (the combined method) caused a reduction in SQP run time. The improvement was most pronounced for the t_3 problems, all of which had a dramatic decrease in median run time.

3.3.3 Optimization Performance

The three optimization methods were run on the same data sets, so each method's sharpening distances can be directly compared. Figure 3.8 shows the objective func-

[3]The sample size effect could be caused by greater inherent difficulty in the n-dimensional optimization, or by the presence of more distant outliers in the larger t_3 samples.

Table 3.1: Convergence and run time results.

			Proportion converging			Median run time (s)		
Density	Bandwidth	n	Greedy	SQP	Combined	Greedy	SQP	Combined
t_3	$0.75h_{SJ}$	25	1	0.956	1	0.061	9.6	2.0
t_3	$0.75h_{SJ}$	50	1	0.880	1	0.13	19	4.8
t_3	$0.75h_{SJ}$	100	1	0.844	0.992	0.31	41	16
t_3	h_{SJ}	25	1	0.964	1	0.044	3.9	2.0
t_3	h_{SJ}	50	1	0.936	0.996	0.092	15	4.7
t_3	h_{SJ}	100	1	0.916	0.996	0.21	38	15
mixture	$0.75h_{SJ}$	25	1	1	1	0.12	4.4	3.6
mixture	$0.75h_{SJ}$	50	1	1	1	0.28	9.7	8.4
mixture	$0.75h_{SJ}$	100	1	0.992	1	0.74	31	28
mixture	h_{SJ}	25	1	1	1	0.060	3.3	3.1
mixture	h_{SJ}	50	1	1	1	0.14	8.2	7.7
mixture	h_{SJ}	100	1	1	1	0.38	28	26

tion values for the greedy and SQP methods plotted against each other, for the 2872 pairs of optimizations where SQP converged. Cases based on each of the two target distributions are plotted with different markers. The 1:1 line is also shown on the plot. Points below the line represent runs where the greedy method outperformed SQP, and points above the line represent runs where SQP found the better solution.

The figure suggests that the greedy method had good relative performance. While most of the runs had similar results for the two methods, there were also a large number of runs where the SQP objective value greatly exceeded the greedy value. These are runs where SQP stopped at a particularly poor local minimum. Interestingly, most of these poor SQP results arose in problems based on the mixture distribution, where convergence was not a problem. There were also some data sets where SQP greatly outperformed `improve`, but such cases were much less frequent.

To facilitate a more detailed comparison, objective function values for the greedy and combined methods were normalized relative to the SQP sharpening distance for the same sample and bandwidth. The normalized sharpening distance is the ratio of that method's L_1 distance to the SQP value. Figure 3.9 shows box plots of the normalized sharpening distance for both the greedy and combined methods, for all 12 simulation cases. Boxes show locations of the first, second, and third quartiles, and whiskers extend to the most extreme values differing from the median by less than

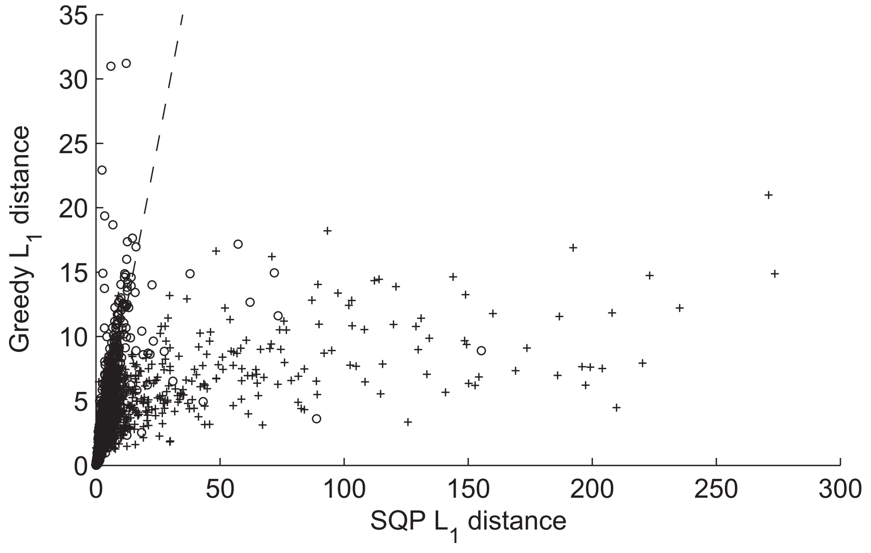

Figure 3.8: Scatter plot of sharpening distances across all simulation runs. Circles and crosses denote cases based on the t_3 and mixture distributions, respectively.

1.5 times the interquartile range.

Normalized objective function values less than 1 indicate performance better than SQP. The boxplots for the greedy method show that it strongly outperformed SQP on the t_3 problems, and was roughly equivalent to SQP on the mixture problems. For the t_3 cases, all but one of the cases have their third quartiles less than one, indicating that the greedy result was better than the SQP result more than 75% of the time. The improvement over SQP is also more pronounced for larger sample sizes. In the mixture cases, SQP outperformed greedy when the bandwidth was smaller, while neither method was clearly superior for the larger bandwidth.

Looking at the combined-method cases in Figure 3.9, it is clear that the combined method performed better than the default SQP with $\mathbf{x}$ as its starting point. Starting at the greedy optimum had a pronounced effect for the more difficult t_3 cases, but only a negligible effect on the mixture cases. Note that using the greedy starting point does not always improve the performance of SQP. The best starting point for SQP is sample-dependent and one could not expect any rule to provide the best start for all cases.

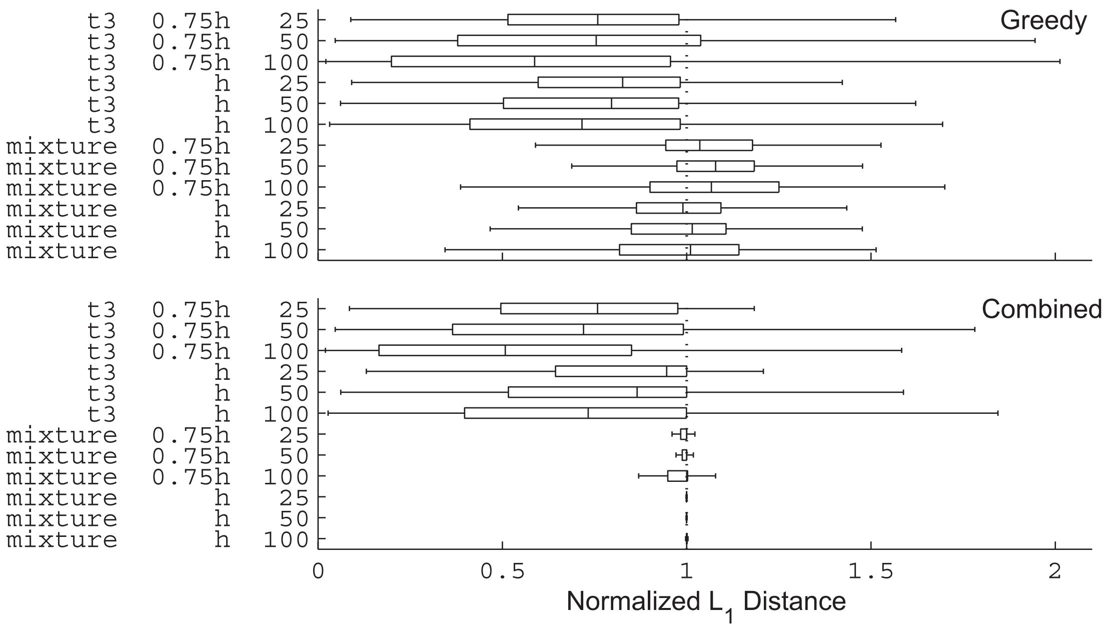

Figure 3.9: Box plots of normalized L_1 sharpening distance, for both the greedy and combined search methods. The labels at left give the simulation case.

As a further illustration of the performance of the methods, Figure 3.10 gives plots of the density estimates for nine randomly-selected simulation data sets. Each plot gives the unsharpened density as well as the sharpened density based on both the SQP and greedy search. Plots 1–3 show cases where SQP found the better result, and plots 4–9 show cases where the greedy algorithm found the better result. These examples were sampled from only those cases where the relative difference in sharpening distance was large (the worse method's sharpening distance being at least 50% larger than the better method's).

The examples show that for cases when the unsharpened estimate is nearly unimodal (as in plots 1, 3, 6, and 9), there is little qualitative difference between the greedy and SQP solutions despite the large relative difference in $L_1(\mathbf{y}, \mathbf{x})$. When the original estimate does have outliers or other large deviations from unimodality, the differences in the estimate are more pronounced, and typically the SQP estimate is inferior (as in plots 4, 5, 7, and 8). The greedy estimate matches the unsharpened curve exactly at points away from the unwanted modes, while the SQP estimate may

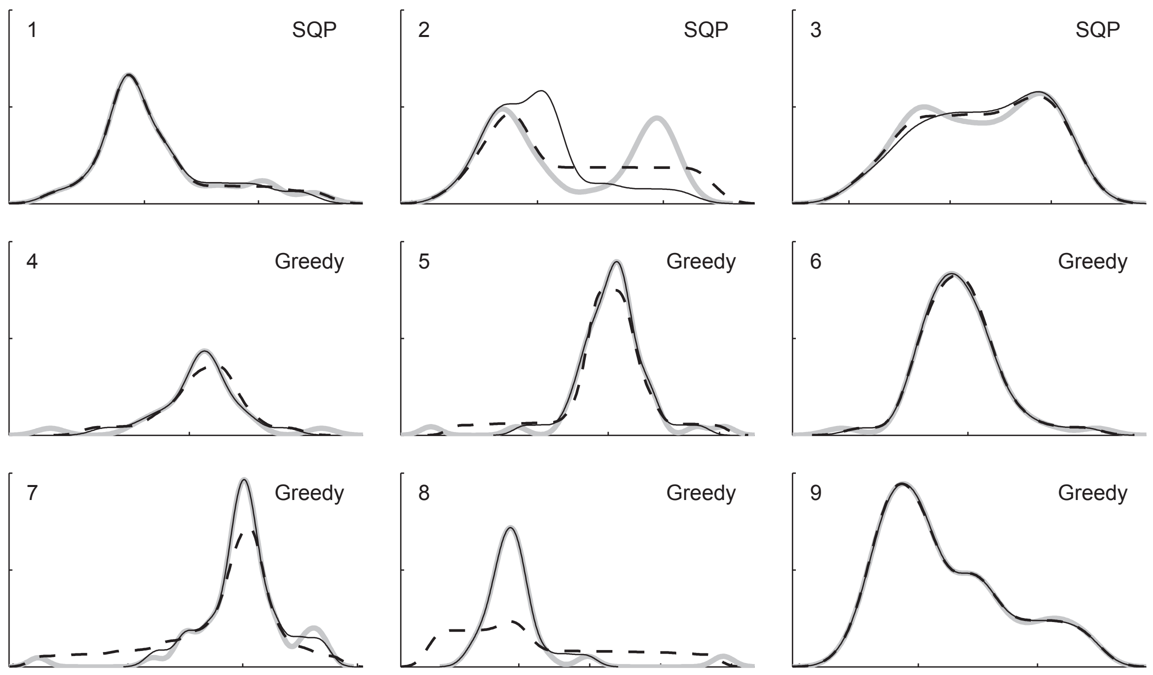

Figure 3.10: Comparing the unsharpened estimate (thick grey line), greedy estimate (thin solid line), and SQP estimate (dashed line) for nine simulation data sets. Plot labels in the upper right indicate which method had a smaller L_1 sharpening distance.

be poor everywhere if the algorithm converges to a low-quality local optimum.

Plot 2 in Figure 3.10 is something of a special case. The data happened to arise in such a way that the original estimate consisted of two modes of nearly equal height. In this case neither method could estimate the density well, and the results of either method would be highly sensitive to the initial solution provided.

Figure 3.11 provides some further justification for the claim that the greedy algorithm produces reasonable density estimates, by comparing the greedy- and SQP-based estimates, across all the generated data sets for which SQP was able to converge. The plot shows the ECDF of the total variation distance (equation 2.8) between the greedy- and SQP-based estimates, based on two groups of cases: the 1117 data sets for which SQP was better than greedy (in the $L_1(\mathbf{y}, \mathbf{x})$ sense), and the 1755 data sets for which it was worse.

The figure shows that for those cases where SQP was better than the greedy algorithm in sharpening-distance terms, the density estimates did not differ by much.

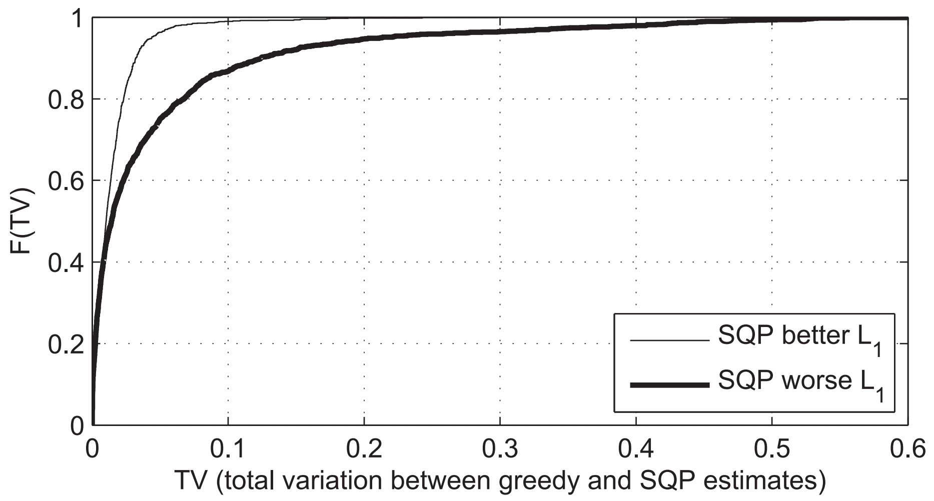

Figure 3.11: Empirical CDFs of the total variation distances between SQP and greedy density estimates.

Over 95% of those cases had $TV(\mathbf{y}_{\text{sqp}}, \mathbf{y}_{\text{greedy}}) < 0.05$, and only about 1% of them had $TV > 0.1$. Conversely, when SQP performed worse than the greedy method, the estimates had more pronounced differences. Only about 70% of the SQP-worse cases had $TV < 0.05$, and about 13% of the runs had $TV > 0.1$. In other words, when the greedy estimate loses, it does not lose by much, but when it wins, it can win by a wide margin. This is in agreement with the observations made from the sample of results illustrated in Figure 3.10.

3.3.4 Estimation Performance

The simulation results can also be used to compare the statistical performance of the density estimators involved. Table 3.2 summarizes the results. For each combination of true density and sample size, the average TV distance from the truth is shown for seven different estimators. The columns labeled KDE, SQP, and Greedy give the results for the unsharpened (multimodal) KDE, the unimodal SQP estimate, and the unimodal greedy estimate, respectively; and one group of columns is given for each bandwidth (h_{SJ} or $0.75h_{SJ}$). The column labeled Reboul gives the TV distance for the unimodal estimator described by Reboul (2005), which is an extension of the

Table 3.2: Sample mean of TV distances from the truth across 250 replications.

		$h = h_{SJ}$			$h = 0.75h_{SJ}$		
Case	Reboul	KDE	SQP	Greedy	KDE	SQP	Greedy
t_3, $n = 25$	.258	.156	.166	.149	.167	.187	.151
t_3, $n = 50$	.189	.120	.132	.114	.127	.149	.116
t_3, $n = 100$	.147	.095	.119	.091	.101	.133	.093
mixture, $n = 25$	.242	.183	.175	.173	.179	.162	.162
mixture, $n = 50$	.182	.148	.141	.140	.142	.129	.126
mixture, $n = 100$	.139	.118	.113	.111	.111	.106	.098

Note: The SQP results exclude runs that failed to converge. The standard error of the estimate is less than 0.0036 for all table entries.

work of Birgé (1997). This estimator is a histogram with automatically-determined variable bin widths. While it is not smooth, its performance can be used as a reference point, since an upper bound on its L_1 risk has been established (Reboul, 2005).

Considering first the results for the mixture distribution, both unimodal estimates demonstrate a slight improvement in TV distance over the unsharpened KDE. The greedy results are never larger than the SQP values, but the differences between the two sharpening methods are very small. For the t_3 cases, SQP performs worse than either the greedy algorithm or the unsharpened estimate. This is because SQP occasionally converged to local optima far from the best solution, causing the distribution of TV values to be right-skewed (the median TV values for the t_3 cases follow a pattern similar to the mixture cases). Finally, all of the smooth estimates, including the unconstrained KDEs, were markedly better than the Reboul estimator. This is not surprising since the densities being estimated were in fact smooth, and the sample sizes considered were relatively small.

The results of this brief study agree with the intuition that when $\mathbf{x}$ is sampled from a unimodal density, estimation can be improved by adding a unimodality constraint. Braun and Hall (2001) and Hall and Kang (2005) have also demonstrated this from a squared error perspective. Naturally one must find a good data sharpening solution to achieve this improvement in practice. In this respect the greedy algorithm showed an advantage over SQP, particulary in the t_3 examples.

3.4 An Iterated Greedy Algorithm

The performance of the greedy algorithm can be improved by running it in an iterative manner, with random perturbations introduced between iterates. The perturbations allow the search to escape local optima, making it feasible, at least in principle, to use the greedy algorithm on more difficult problems.

3.4.1 Iterated Local Search

The term *metaheuristic* is used to describe high-level search strategies that are not specific to a particular problem instance, but that may be used to guide the design of algorithms for particular situations. Iterated local search (Lourenço et al., 2010; Talbi, 2009, sec. 2.6) is a conceptually simple metaheuristic that can be used to enhance the performance of any local optimizer. It has been applied primarily to combinatorial optimization problems, but its structure does not preclude its use in continuous problems like the ones considered here.

The generic ILS algorithm is given in Algorithm 3.2. The metaheuristic assumes that the local optimizer, called `locsearch` in the pseudocode, is a black box routine capable of finding optimal solutions in the neighbourhood of its starting point. ILS attempts to improve the performance of `locsearch` by embedding it in an iterative procedure.

Let $\mathtt{s}'$ be the locally-optimal solution found by `locsearch`, starting from some user-supplied initial guess. This solution is used to start the three-step iteration that is the core of the ILS algorithm, and is given in lines A, C, and D in the psuedocode. The first step is to jump from $\mathtt{s}'$ to another point $\mathtt{s}$ in the search space (using the function `perturb` to do so). The local search is then started from $\mathtt{s}$ to find a new optimum $\mathtt{s}'_{new}$. In the final step, the `accept` routine is used to determine which of $\mathtt{s}'$ and $\mathtt{s}'_{new}$ are retained to be used in the next iteration. This process is continued until some stopping criterion is satisfied.

The operational characteristics of ILS depend on the details of the `perturb` and `accept` steps. The goal in perturbing a locally-optimal solution is to find a point

Algorithm 3.2: Iterated local search.

```
Input: A starting solution, $\mathbf{s}_0$.
Output: Final solution $\mathbf{s}^*$.
Set $\mathbf{s}' \leftarrow$ locsearch($\mathbf{s}_0$)          *Find the first local optimum*
repeat
A   Set $\mathbf{s} \leftarrow$ perturb($\mathbf{s}'$)          *Produce a new solution*
C   Set $\mathbf{s}'_{new} \leftarrow$ locsearch($\mathbf{s}$)          *Find another local optimum*
D   if accept($\mathbf{s}', \mathbf{s}'_{new}$)          *Check acceptance criterion*
        Set $\mathbf{s}' \leftarrow \mathbf{s}'_{new}$
until stop condition satisfied
Set $\mathbf{s}^* \leftarrow \mathbf{s}'$
```

close enough to the current optimum to stay in the vicinity of good solutions, but far enough from the optimum that the search can escape from poorer local optima and move toward better ones. The `perturb` routine usually involves a stochastic element so that the search does not cycle between solutions. Even so, if the perturbations are too small the search will keep re-visiting the same parts of the search space and the algorithm will terminate prematurely. If they are too large, the search will resemble a random-restarts approach and will fail to exploit any structure that is present in the objective function.

The acceptance criterion will affect the efficiency of the search in a similar way. If only improving moves are accepted, for example, exploration of the search space will be inhibited and it will be harder to find distant solutions that are better than the current optimum. If we accept too many solutions that worsen the objective function, however, the search might spend too much time in regions of the search space that are not interesting. This is the classical trade-off between exploitation (moving from a good solution to better ones in the same neigbourhood) and exploration (moving to other neighbourhoods in the hopes of finding better solutions), that arises in all heuristic optimizers.

3.4.2 Incorporating the Greedy Algorithm in an ILS Scheme

The speed and guaranteed convergence of the greedy algorithm make it well suited as the local search component of an ILS scheme. A crucial element that is lacking in the standard ILS prescription, however, is the ability to handle constraints. In a problem with constraints, perturbing a feasible solution randomly will often result in an infeasible point being selected. A local search based on a feasibility-preserving method (like `improve`) cannot handle infeasible starting points.

One way to use ILS in constrained optimization problems is to add a *repair* step between the perturbation and optimization steps (lines A and C) in Algorithm 3.2. The repair step acts to take an infeasible solution and move it back into the feasible part of the search space.

It happens that `improve` can also be used as a repair method, leading to the proposed iterative method `ILSimprove`, presented in Algorithm 3.3. The algorithm starts by running the greedy search as usual. The resulting feasible solution is used to start the ILS iterations (with repair) shown in lines A through D. In line A, the current solution vector is perturbed by the addition of Gaussian noise with standard deviation σ. The perturbed solution $\mathbf{y}_\epsilon$ will usually not satisfy the constraint, but it can be made feasible by using `improve` as a repair method in line B. This step (using the previous best solution as the feasible starting point and $\mathbf{y}_\epsilon$ as the target) produces a new feasible point from which to start the optimization again in line C. Finally the better of the new and old feasible solutions is retained in line D. Repeating this perturb-repair-improve cycle typically yields better solutions than just running the greedy algorithm once.

3.4.3 Performance of `ILSimprove`

Algorithm 3.3 was applied to the simulation problems of section 3.3. To run the algorithm, a value for the noise standard deviation σ must be chosen, and a stopping rule must be defined. For the noise level, it was decided to set σ to equal the KDE's bandwidth, h. This is reasonable, because h is a measure of the scale that one

Algorithm 3.3: Iterated greedy algorithm for data sharpening (`ILSimprove`).

```
    Input: The data, $\mathbf{x}$; a feasible starting solution, $\mathbf{y}_0$; a noise level, $\sigma$.
    Output: Final solution $\mathbf{y}^*$.
    Set $\mathbf{y}' \leftarrow$ improve($\mathbf{y}_0$,$\mathbf{x}$)                  *Use Algorithm 3.1*
    repeat
        Let $\boldsymbol{\epsilon}$ be a vector of iid N(0,$\sigma^2$) variates.
A       Set $\mathbf{y}_\epsilon \leftarrow \mathbf{y}' + \boldsymbol{\epsilon}$,               *Perturb: $\mathbf{y}_\epsilon$ is probably not feasible*
B       Set $\mathbf{y}_\mathrm{f} \leftarrow$ improve($\mathbf{y}'$,$\mathbf{y}_\epsilon$)    *Repair: data sharpening with $\mathbf{y}_\epsilon$ as target*
C       Set $\mathbf{y}'_{new} \leftarrow$ improve($\mathbf{y}_\mathrm{f}$,$\mathbf{x}$)   *Improve: data sharpening with $\mathbf{x}$ as target*
D       if $\mathbf{y}'_{new}$ is better than $\mathbf{y}'$           *Keep the better of two solutions*
            Set $\mathbf{y}' \leftarrow \mathbf{y}'_{new}$
    until stop condition satisfied
    Set $\mathbf{y}^* \leftarrow \mathbf{y}'$
```

defines as local for a particular data set. Adding N(0,h^2) noise in the perturbation step should allow perturbed density estimates to cover the neighbourhood of a given solution without straying too far away. For the stopping rule, the search was limited to only 50 iterations, because this made `ILSimprove` have approximately the same average run time as SQP.

Across all simulation runs, the iterated algorithm was able to improve upon the single-run solution 95% of the time. Consequently its advantage over SQP was also greater: while the greedy algorithm outperformed SQP in 61% of runs overall, the iterated version did so in 84% of runs. Figure 3.12 illustrates this case-by-case, using boxplots constructed the same way as those of Figure 3.9. Comparison of the two figures shows that relative to a single greedy run, the average performance is improved (the distributions of sharpening distances are shifted left) and fewer runs compare poorly to SQP (the right tails are much lighter).

3.5 Limitations and Extensions

The `improve` algorithm compares favorably to SQP in the important special case of univariate, unimodal density estimation. While this is promising, there are a few limitations inherent in the design of the algorithm:

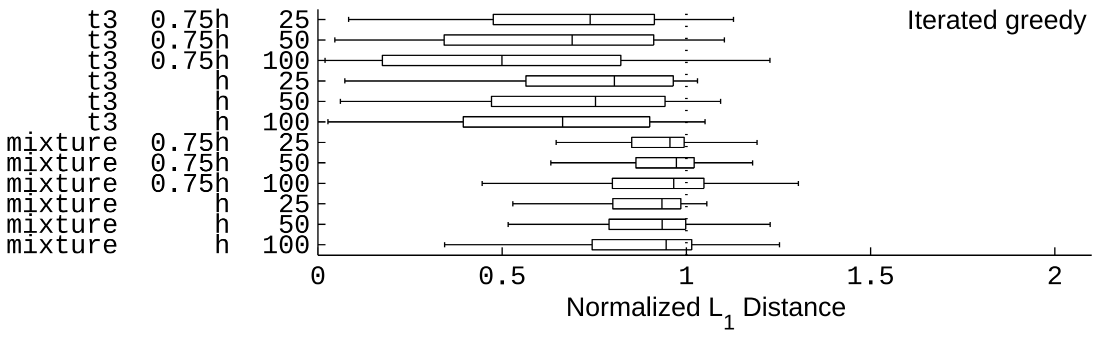

Figure 3.12: Box plots of normalized L_1 distance for the simulation data sets, using the iterated greedy algorithm. Compare to Figure 3.9.

1. It performs well in the unimodal problems because the default starting point ($\mathbf{y} = m_0\mathbf{1}$, where m_0 is the location of the highest unsharpened mode) is well suited to this constraint. The algorithm causes the points in $\mathbf{y}$ to spread out incrementally toward $\mathbf{x}$, which often yields a good estimate. For other constraints, it will be harder to find good starting solutions (consider bimodal estimation, for example).

2. The algorithm requires a large number of constraint checks. In cases where checking constraint validity is computationally intensive, the speed advantage of the greedy approach could be lost[4].

3. The number of constraint checks is proportional to n, meaning that the method might not scale well to large sample sizes. In well-behaved cases many of the n points will be moved home early in the search, reducing the computational burden. But the magnitude of this effect will be problem-specific.

The ILS variant of `improve` is one possibility for overcoming the first limitation. The proposal given in Algorithm 3.3 is a starting point, and could itself be improved. In its present form it at least provides a way of generating a variety of feasible starting points, by using `improve` as a repair mechanism. More sophisticated features like

[4]Any method that uses a black-box constraint will require frequent checks of constraint validity, but this problem is exacerbated in the case of `improve`, because it moves one element of $\mathbf{y}$ at a time.

stochastic acceptance rules or adaptive perturbation could be implemented to increase the performance of the algorithm.

Finally, the idea of using `improve` as a repair function is itself potentially useful in other constrained optimization settings. Repair of infeasible solutions is a general strategy in adapting heuristics to handle constraints (Michalewicz and Fogel, 2004, ch. 9), and designing an appropriate repair method can be challenging. Given a feasible solution $\mathbf{y}_\mathrm{f}$ and an infeasible one $\mathbf{y}$, `improve(`$\mathbf{y}_\mathrm{f}$`,`$\mathbf{y}$`)` will return a feasible solution that is not on the straight-line path between the two points (because it acts on the solutions elementwise). Such a repair method could be beneficial in population-based heuristics that may track sub-groups of feasible and infeasible solutions.

Chapter 4

A Particle Swarm Algorithm for Data Sharpening

The greedy algorithm of the previous chapter is a single-solution, deterministic search method. The algorithm presently to be described, by contrast, is a multiple-solution search with stochastic movement rules. It is called constrained estimation particle swarm optimization (CEPSO). Search is carried out by two populations of solutions that collaboratively explore the solution space using the principles of particle swarm optimization (PSO). The algorithm is more complex and more computationally intensive than the `improve` algorithm, but it can handle a wider range of problems. It will be defined in a generic way suitable for any of the adjustment methods of Section 1.3, but demonstrations and evaluations of the method will use the data sharpening estimator, $\hat{f}^M$.

A review of PSO is provided below. The new algorithm is then described, followed by demonstrations on the example data sets, and then by simulation results.

4.1 Particle Swarm Optimization

Particle swarm optimization is a population-based optimization heuristic originally proposed by Kennedy and Eberhart (1995). It is one of many heuristics that seek to solve problems by mimicking the behaviour of biological systems (flocks of birds, ant colonies, swarms of bees, and so on). An overview of this natural computing approach to optimization is provided by de Castro (2006). A brief summary of the standard PSO algorithm is given below. For further background, see Kennedy et al. (2001), Engelbrecht (2005), or Poli et al. (2007).

Consider the problem of minimizing a scalar objective function of n variables. Particle swarm optimization conceives of potential solutions as objects (particles) flying through the n-dimensional solution space, in discrete-time steps. A swarm is a collection of P particles. The pth particle has position $\mathbf{s}_p$ and velocity $\mathbf{v}_p$, both of which are n-vectors. Particle movement is governed by simple rules that allow a limited form of memory and inter-particle communication, and encourage the swarm to move toward better solutions over time.

The quality or fitness of a particle's current position is determined by its objective function value. Each particle is aware of its current fitness, and also retains a memory of the best location it has visited in the past—its *personal best* solution, $\dot{\mathbf{s}}_p$. Particles are also able to see the personal best solutions of other particles, and use this information to determine their own *local best* solution. The pth particle's local best solution, $\ddot{\mathbf{s}}_p$, is the best of the personal best solutions it is allowed to observe. The set of particles used to determine the local best is called the particle's *neighbourhood*. There are various ways to define particle neighbourhoods, but in the present work the simple "lbest(k)" scheme (Kennedy and Mendes, 2002) is used, in which neighbourhoods are established following the sequence of particle indices. In this scheme, particle p has k neighbors, with indices $p + i - P\lfloor \frac{p+i}{P+1} \rfloor$ for $i = 1, \ldots, k$ (here $k \leq P - 1$, and $\lfloor\rfloor$ indicates the floor function). Figure 4.1 provides two examples of how neighbours are assigned. The use of local bests is an important feature of PSO, as it allows information about promising solutions to be transmitted through the swarm over time.

Each iteration of a PSO algorithm involves i) updating $\dot{\mathbf{s}}$ and $\ddot{\mathbf{s}}$ for each particle, ii) determining new particle velocities, and iii) updating the particle positions. The

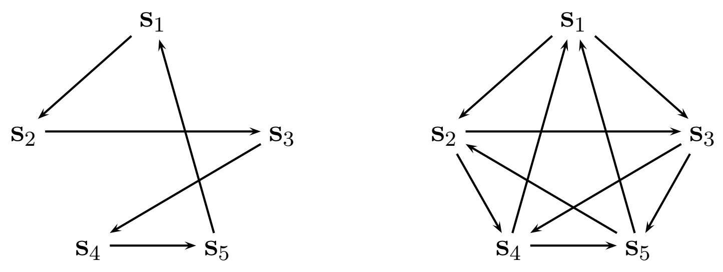

Figure 4.1: Illustrating the lbest(k) neighbourhood structure for a swarm with $P = 5$ and $k = 1$ (left) or $k = 2$ (right). Arrows point from each particle to its neighbours.

velocity update equation is

$$\mathbf{v}_p \leftarrow w\mathbf{v}_p + c_1\mathbf{r}_1 \circ (\dot{\mathbf{s}}_p - \mathbf{s}_p) + c_2\mathbf{r}_2 \circ (\ddot{\mathbf{s}}_p - \mathbf{s}_p), \tag{4.1}$$

and the position update is

$$\mathbf{s}_p \leftarrow \mathbf{s}_p + \mathbf{v}_p. \tag{4.2}$$

In the velocity update, $w \geq 0$ is a scalar *inertia weight*, $c_1 \geq 0$ and $c_2 \geq 0$ are scalar *acceleration coefficients*, $\mathbf{r}_1$ and $\mathbf{r}_2$ are independent vectors of independent $U(0, 1)$ variates (re-generated for each p at each iteration), and $\circ$ is used to represent elementwise multiplication. The updated velocity is a weighted sum of three components: the current velocity, a component in the direction of the personal best, and a component in the direction of the local best.

The second and third terms in update (4.1) represent attraction to $\dot{\mathbf{s}}_p$ and $\ddot{\mathbf{s}}_p$, respectively. These attractors tend to keep the particle near solutions that are known from past iterations to be of high quality. This tendency is counterbalanced by the first term, a self-velocity term, that gives the particle impetus to explore new directions. The random vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ make the search stochastic. Note that the random vectors are multiplied elementwise, which ensures that attraction does not act directly along a line from the current position to the attractor. Rather, each coordinate of the solution is attracted a random amount, independently. Each coordinate in the updated position is the sum of two (location-scale transformed) uniform random variables. Figure 4.2 shows the marginal and joint distributions of

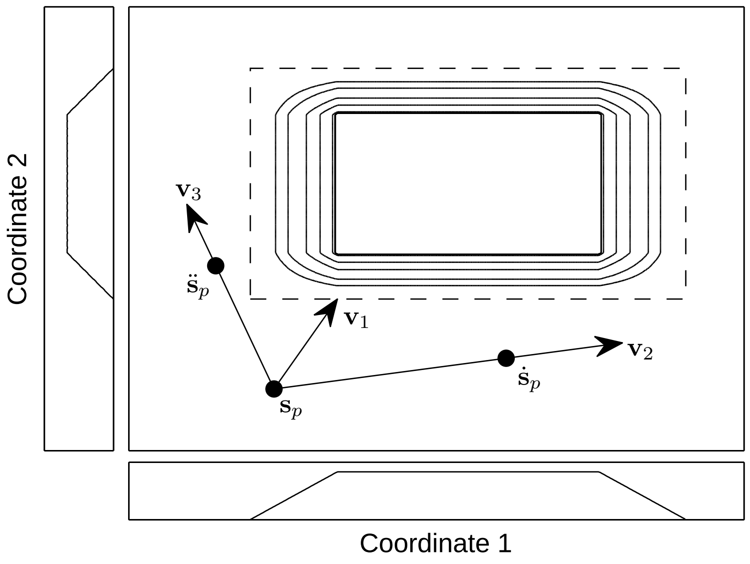

Figure 4.2: Joint and marginal distributions of possible moves for a single particle $\mathbf{s}_p$, in the canonical PSO. Contours of the joint density are shown inside the dashed box (which is the density's support). The vectors $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ represent the three terms in velocity update (4.1).

the new position for a hypothetical two-dimensional particle using update rules (4.1) and (4.2).

PSO search continues using these particle movement rules until some stopping criterion is satisfied. At any point in the search, the personal best solution with the best fitness is called the *global best* solution, represented by $\tilde{\mathbf{s}}$. When the search is terminated, $\tilde{\mathbf{s}}$ is returned as the selected optimum. With appropriate choice of the algorithm parameters and sufficiently long run time, the swarm will converge—all particles will be within some small distance of $\tilde{\mathbf{s}}$, and particle velocities will be nearly zero.

The values of w, c_1, and c_2 determine the convergence behaviour and the long-run trade-off between exploration (greater swarm diversity, less chance of being trapped near local optima) and exploitation (reduced diversity, faster convergence). Values of w greater than unity cause the swarm to diverge due to increasing velocities, while setting $0 < w < 1$ promotes convergence. At the same time, choosing larger values of c_1, c_2 promotes greater exploration by increasing the average step size. It is customary, but not necessary, to set $c_1 = c_2$. The neighbourhood size k also plays a role. Larger k hastens the spread of information through the swarm, encouraging convergence; smaller k slows communication and promotes search diversity.

Clerc and Kennedy (2002) developed the so-called constriction coefficient approach to setting the three constants in the velocity update, such that the swarm is guaranteed to avoid divergent velocities, and ultimately to converge to a single solution. The constriction method, with typical settings, is equivalent to employing the standard update equation (4.1) with $w = 0.730$ and $c_1 = c_2 = 1.496$ (see Poli et al., 2007, p. 37–38). This choice of coefficients is taken to be the canonical PSO, on which the methods of this chapter are based. It is recorded in Algorithm 4.1.

Algorithm 4.1: The canonical PSO algorithm.

```
Input: Population size, $P$; neighbourhood size, $k$.
Output: Best solution found, $\mathbf{s}^*$.
Initialize particle positions and velocities.
repeat
    for $i = 1, \ldots, P$
        Compute $\dot{\mathbf{s}}_i$ and $\ddot{\mathbf{s}}_i$.
        Set $\mathbf{v}_i \leftarrow 0.73\mathbf{v}_i + 1.5\mathbf{r}_1 \circ (\dot{\mathbf{s}}_i - \mathbf{s}_i) + 1.5\mathbf{r}_2 \circ (\ddot{\mathbf{s}}_i - \mathbf{s}_i)$
        Set $\mathbf{s}_i \leftarrow \mathbf{s}_i + \mathbf{v}_i$
until stop condition is met
Set $\mathbf{s}^* \leftarrow \tilde{\mathbf{s}}$
```

4.2 Constrained Estimation PSO

The canonical PSO is applicable to unconstrained (or at most, bound constrained) optimization problems. CEPSO is an attempt to incorporate the standard PSO into a more sophisticated heuristic capable of handling the difficult constraints that arise in shape-restricted estimation problems.

4.2.1 Algorithm Description

The proposed algorithm is a cooperative search conducted by two swarms of P particles each. The first swarm, which will be referred to as Swarm 1 or the *exploiter* swarm, is primarily responsible for finding improved feasible solutions in the vicinity of the best known solution. It is this swarm that will ultimately provide the final result. The second swarm, called Swarm 2 or the *explorer* swarm, is responsible for covering the remainder of the search space, looking for either i) new feasible solutions in distant areas, or ii) promising infeasible solutions that could lead to better search paths in Swarm 1. At each iteration, after both swarms have updated their particle positions, the swarms have the opportunity to trade particles in an *exchange* step. Swarm 1 receives any feasible or promising particles from Swarm 2, in turn releasing its worst particles to join the exploration group.

In this system, the set of personal best solutions, $\{\dot{\mathbf{s}}_p\}$, is held in common between the two swarms. The explore/exploit/exchange steps are continued until either the values of $\{\dot{\mathbf{s}}_p\}$ converge, or the value of the global best $\tilde{\mathbf{s}}$ stops changing. Note that while particles in either swarm may visit infeasible portions of the search space, only feasible solutions are permitted to become personal bests. This guarantees that $\tilde{\mathbf{s}}$ will always be feasible, and therefore the final solution will be as well.

The two swarms use different velocity update equations to perform their different functions. For the exploiter swarm, the update equation is

$$\mathbf{v}_p \leftarrow w\mathbf{v}_p + 1.5\mathbf{r}_1 \circ (\dot{\mathbf{s}}_p - \mathbf{s}_p) + 1.5\mathbf{r}_2 \circ (\ddot{\mathbf{s}}_p - \mathbf{s}_p), \tag{4.3}$$

which is the same as the canonical PSO update equation (4.1), with c_1 and c_2 set to the values required by the constriction coefficient approach. The value of w is left unspecified for the moment; it is used to control swarm dynamics as described in the next section. As with the canonical update, $\dot{\mathbf{s}}_p$ and $\ddot{\mathbf{s}}_p$ act as attractors for the particle. In this case, however, the attractors may only be feasible points, causing Swarm 1 to stay close to known feasible regions.

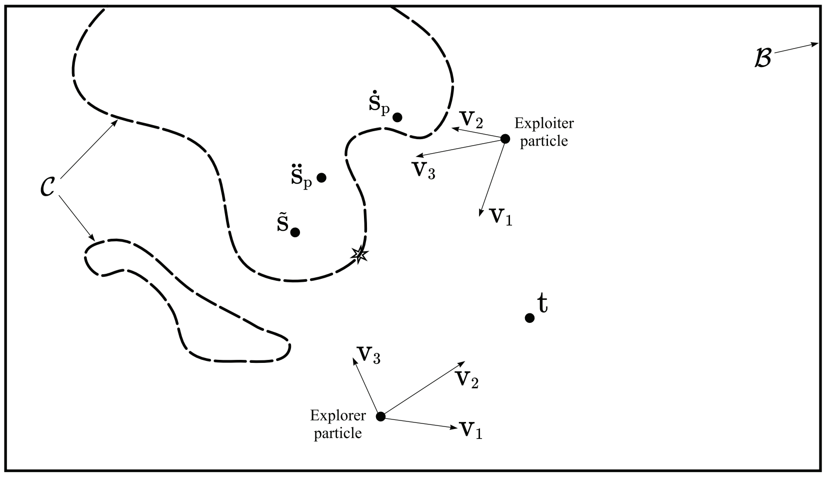

Figure 4.3: The velocity components for particles in either swarm. For clarity, components $\mathbf{v}_2$ and $\mathbf{v}_3$ are shown to point directly toward their attractors. The actual move will be random, as shown in Figure 4.2.

For the explorer particles, the proposed velocity update is

$$\mathbf{v}_p \leftarrow w\mathbf{v}_p + 1.5\mathbf{r}_1 \circ (\mathbf{t} - \mathbf{s}_p) + 1.5\mathbf{r}_2 \circ (\tilde{\mathbf{s}} - \mathbf{s}_p). \tag{4.4}$$

Swarm 2 particles are attracted to the target or pilot solution $\mathbf{t}$ and the global best $\tilde{\mathbf{s}}$, instead of their personal or local bests. Because the local bests are not involved, the particles of Swarm 2 do not influence each other. Every particle independently searches a larger part of the solution space around the global best (where good solutions are known to exist) and the target (where we would like to find new solutions).

Velocity updates (4.3) and (4.4) are illustrated pictorially in Figure 4.3. The figure shows a hypothetical two-dimensional optimization, with feasible set $\mathcal{C}$. A star indicates the optimal solution, which is the feasible solution closest to $\mathbf{t}$. One particle from each swarm is shown, with arrows $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ representing the three terms in the respective update equations. Optimization by CEPSO is done with bound constraints, to prevent particles from moving too far afield, and to permit bound restrictions on solutions. The hyper-rectangular search space defined by the bounds is denoted $\mathcal{B}$.

The example of Figure 4.3 is extended in Figure 4.4 to illustrate how “promising”

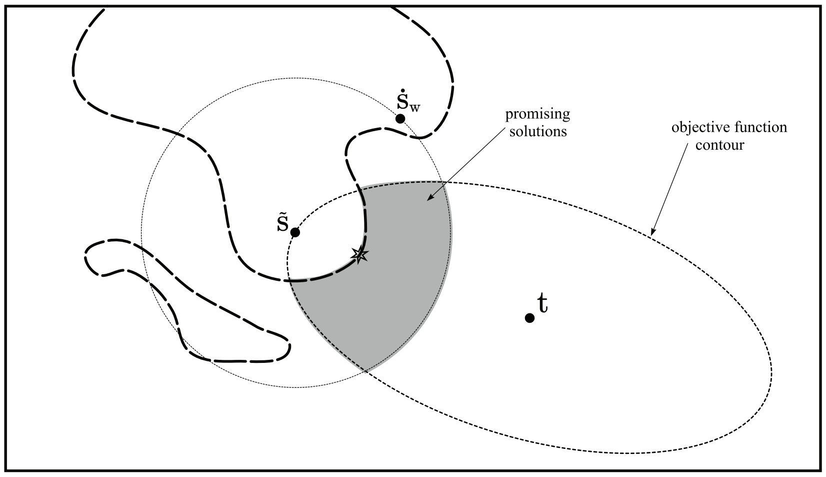

Figure 4.4: The region of promising solutions.

solutions are defined. A solution is considered promising if it is in the set

$$\{\mathbf{s} : \mathbf{s} \notin \mathcal{C},\ \|\mathbf{s} - \tilde{\mathbf{s}}\| \leq \|\dot{\mathbf{s}}_w - \tilde{\mathbf{s}}\|,\ \delta(\mathbf{s}, \mathbf{t}) \leq \delta(\tilde{\mathbf{s}}, \mathbf{t})\}\,, \tag{4.5}$$

where $\dot{\mathbf{s}}_w$ is the worst personal best (the one farthest from $\tilde{\mathbf{s}}$), and $\|\cdot\|$ represents Euclidean distance. This definition is motivated by the desire to find points near the boundary of feasibility, in the general direction of the target. Such particles are moved into the exploiter swarm during the exchange step. The hope is that subsequent exploitation moves will cause the promising particles to cross into the feasible region and provide improved solutions.

Note that the size of the promising region will shrink throughout the search as $\tilde{\mathbf{s}}$ moves closer to $\mathbf{t}$ and $\dot{\mathbf{s}}_w$ approaches $\tilde{\mathbf{s}}$. If Swarm 1 converges to the global optimum as hoped, then continuing to run Explore and Exchange steps late in the search is computationally wasteful. Nonetheless, the full Explore/Exploit/Exchange cycle is run throughout the search, since there is no way to have final assurance that Swarm 1 is actually in the vicinity of the global optimum.

4.2.2 Controlling Swarm Dynamics

In the canonical PSO, the trade-off between global exploration and local exploitation is determined by the choice of w, c_1, and c_2. In the proposed approach, exploration and exploitation duties are explicitly assigned to different swarms. For each swarm to carry out its function, its spatial extent must be controlled as the search progresses. At each iteration, this is done by first comparing each swarm to a *shrinkage/expansion boundary*, and then adjusting the inertia weight w accordingly.

Figure 4.5 continues the previous two illustrations, this time showing the shrinkage/expansion boundaries used for the two swarms. Both boundaries are level sets of the objective function. For Swarm 1, the boundary is $\{\mathbf{s} : \delta(\mathbf{s}, \mathbf{t}) = \delta(\dot{\mathbf{s}}_w, \mathbf{t})\}$, and for Swarm 2 it is $\{\mathbf{s} : \delta(\mathbf{s}, \mathbf{t}) = \delta(\tilde{\mathbf{s}}, \mathbf{t})\}$. The value of the inertia weight is chosen for each swarm as follows. Let ρ be the proportion of particles that are inside the swarm's shrinkage/expansion boundary. Then set

$$w = \begin{cases} 0.5 & \text{if } \rho < 0.9 \\ 0.73 & \text{if } 0.9 \leq \rho < 1 \\ 1.0 & \text{if } \rho = 1. \end{cases} \tag{4.6}$$

The swarm is considered to be too diffuse if less than 90% of its particles are inside the boundary, and too concentrated if 100% of its particles are inside. Applying rule (4.6) leaves the inertia weight at its default of 0.73 when the swarm is neither too diffuse nor too concentrated. If it is too diffuse, a smaller value of $w = 0.5$ is used to encourage contraction of the swarm, and if it is too concentrated, a larger value of $w = 1$ is used to encourage expansion.

Though this means of control is crude, it is sufficient to prevent velocities from becoming too large or too small, without unduly interfering in the natural swarm dynamics. It also helps each swarm to more quickly adjust to the new search environment when the values of $\dot{\mathbf{s}}_w$ or $\tilde{\mathbf{s}}$ change. This is also illustrated in Figure 4.5. As the search progresses, $\tilde{\mathbf{s}}$ will move toward $\mathbf{t}$, and $||\dot{\mathbf{s}}_w - \tilde{\mathbf{s}}||$ will get smaller. The shrinkage/expansion boundaries will therefore also contract; this will result in a value

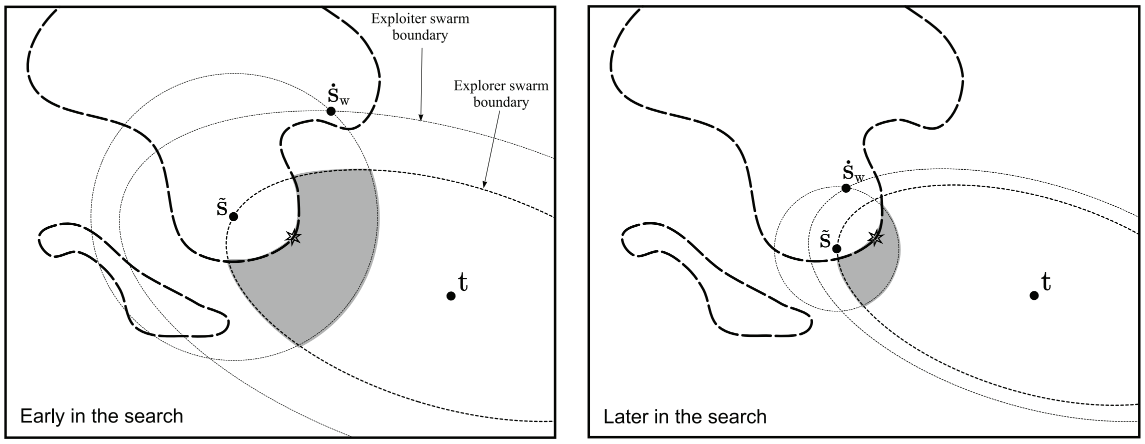

Figure 4.5: The shrinkage/expansion boundary for each swarm, at two stages during the search.

of $w = 0.5$ to be used for a number of iterations until the swarms have adapted themselves to the new boundaries.

Note that although the boundary for Swarm 1 is outside that for Swarm 2, this does not imply that the exploiter swarm is more spread out than the explorer swarm. Swarm 1 will stay concentrated near the locations of good solutions, because exploiter particles are attracted to $\dot{\mathbf{s}}_p$ and $\ddot{\mathbf{s}}_p$, which are always in $\mathcal{C}$.

4.3 Implementation Details

The preceding description of CEPSO explained the general approach of the algorithm. More detailed information, including pseudocode and the means handling of several special cases, is provided below.

4.3.1 The Main Function

The CEPSO algorithm is summarized at a high level in Algorithm 4.2. Inputs to the algorithm include all of the elements required for specification of the problem, as well as a bounding box $\mathcal{B}$ and the population parameters, P and k. A final, optional, input is a set of starting solutions. The main loop of the algorithm consists of three groups of operations: updates for Swarm 2 (the explore step), updates for Swarm 1 (the exploit step), and trading of particles (the exchange step). Every in-bounds particle must have its feasibility checked and its objective re-calculated after it is updated, meaning that there are at most $2P$ calls to the objective and constraint-checking functions per iteration.

Three topics of practical importance have not yet been addressed: how to initialize the swarms, how to handle any particles that move outside of $\mathcal{B}$, and how to define stopping conditions. After considering these questions, the core explore, exploit, and exchange steps are described in more detail.

Algorithm 4.2: Constrained estimation PSO (`CEPSO`).

```
Input: For the problem definition: $\hat{f}_{\mathbf{s}}(\cdot)$, $\mathbf{t}$, $\delta(\cdot,\cdot)$, $\mathcal{I}(\cdot)$, $\mathcal{B}$.
       For the algorithm: $P$, $k$, and starting solutions.
Output: Best feasible solution found, $\mathbf{s}^*$.

Initialize Swarms.
Update personal and local bests.
Exchange particles if possible.
repeat
    Explore step
    Select $w$ for Swarm 2 using (4.6)
    Update Swarm 2 velocities using (4.4)
    Update Swarm 2 positions using (4.2)
    Exploit step
    Select $w$ for Swarm 1 using (4.6)
    Update Swarm 1 velocities using (4.3)
    Update Swarm 1 positions using (4.2)
    Exchange step
    Update personal/local bests
    Exchange particles if possible
until stop conditions are met
Output $\mathbf{s}^* \leftarrow \tilde{\mathbf{s}}$
```

Initializing the swarms.

The CEPSO code is designed to accept P or fewer user-supplied initial solutions, that are used to populate Swarm 1. The remaining particles of Swarm 1, and all of Swarm 2, are randomly initialized by uniform sampling inside $\mathcal{B}$. Some special cases arise if too few of the initial solutions are feasible. If none of the initial particles are feasible, then no personal, local, or global best solutions exist. In this case, both swarms are run as explorer swarms using a random attractor in the place of the global best solution (in a manner made explicit in the next section). If only a few feasible solutions are given or discovered, then exploration moves can occur as usual, but exploitation moves are still problematic, since $\dot{\mathbf{s}}_p$ and $\ddot{\mathbf{s}}_p$ will not exist for all p. In this case, any velocity terms in the exploiter update (4.3) that cannot be computed are set to zero.

The algorithm will converge faster, and to better solutions, if good feasible starting points are supplied. The means of finding feasible starting particles is problem-specific and non-trivial. Additional comments on this topic are given in the conclusion to this chapter.

Out-of-bounds particles.

The problem's bounding box $\mathcal{B}$ is sometimes purely a convenience, but in other cases (as when $\mathbf{s}$ is a vector of probability weights), the estimator $\hat{f}_s$ does not exist for $\mathbf{s} \notin \mathcal{B}$. To handle this, and to prevent particles from spending too much time out of bounds, the self-velocity term $w\mathbf{v}_p$ in (4.3) and (4.4) is set to zero for any particle that is out of bounds. Since all of the other terms in both the Explore and Exploit velocity updates involve attraction to points inside $\mathcal{B}$, setting $w = 0$ will naturally bring particles back in bounds.

Stop conditions.

Search proceeds until some stop conditions are met. Some possible stopping criteria are:

1. The maximum absolute componentwise distance between any two personal best solutions is less than some tolerance:

$$\max_{i,j} \|\dot{\mathbf{s}}_i - \dot{\mathbf{s}}_j\|_\infty < \epsilon. \tag{4.7}$$

2. The global best solution has not changed for g iterations.

3. The search has run for G iterations.

Stop condition 1 indicates that the exploiter swarm has converged, but such convergence may occur slowly, since it requires all particles to visit a feasible point sufficiently close to $\tilde{\mathbf{s}}$. Condition 2 provides a practical stopping criterion based on lack of progress. Condition 3 simply provides an upper bound on the run time by limiting the search to a maximum of G iterations.

For multi-modal problems like those of interest here, choosing termination conditions is inherently difficult, as there is always a chance that better solutions exist somewhere else in the search space. Even if Swarm 1 converges, there is a chance that a new, better region of the search space will be found by Swarm 2 if the search is continued. So the explorer swarm provides some insurance against premature convergence if the search is continued long enough.

4.3.2 Explore and Exploit

The Explore and Exploit steps carry out the PSO moves for Swarm 1 and Swarm 2, respectively. Both types of move are built around the canonical PSO, and both follow the same general steps. First the extent of the swarm is examined to determine whether the inertia weight should be set to a higher or lower value for this iteration. Then the particle velocities are updated. Each velocity is composed of three terms: one term based on the previous velocity, and two more terms that define centers of attraction for the swarm. After being updated, the new velocities are used to update the particles' positions.

Algorithm 4.3 gives pseudocode for both types of move. The steps unique to each type of move are shown separately, in two boxes. Performing the velocity update in the first box will result in an exploration move, while using the steps in the second box will result in an exploitation move. This pseudocode is consistent with the description in Section 4.2.1, but also includes expressions for handling out-of-bounds points, and dealing with situations where $\dot{\mathbf{s}}_p$, $\ddot{\mathbf{s}}_p$, or $\tilde{\mathbf{s}}$ do not yet exist.

The velocity update for Explore is a standard PSO velocity update with three

Algorithm 4.3: Detailed code for the Explore and Exploit steps.

```
Input: A swarm of particles.
Output: An updated swarm.

Adjust the inertia weight
if swarm is too large, then set $w \leftarrow 0.5$; elseif swarm is too small, set $w \leftarrow 1$;
else set $w \leftarrow 0.73$.

Update velocities—EXPLORE method
for each $p$
    if $\mathbf{s}_p$ is in bounds, then set $\mathbf{v}_1 \leftarrow \mathbf{v}_p$; else set $\mathbf{v}_1 \leftarrow \mathbf{0}$.
    Set $\mathbf{v}_2 \leftarrow \mathbf{r}_1 \circ (\mathbf{t} - \mathbf{s}_p)$
    if $\tilde{\mathbf{s}}$ exists, then set $\mathbf{v}_3 \leftarrow \mathbf{r}_2 \circ (\tilde{\mathbf{s}} - \mathbf{s}_p)$; else set $\mathbf{v}_3 \leftarrow \mathbf{r}_2 \circ (\mathbf{R} - \mathbf{s}_p)$.
    Set $\mathbf{v}_p \leftarrow w\mathbf{v}_1 + 1.5\mathbf{v}_2 + 1.5\mathbf{v}_3$

Update velocities—EXPLOIT method
for each $p$
    if $\mathbf{s}_p$ is in bounds, then $\mathbf{v}_1 \leftarrow \mathbf{v}_p$; else $\mathbf{v}_1 \leftarrow \mathbf{0}$.
    if neither $\dot{\mathbf{s}}_p$ nor $\ddot{\mathbf{s}}_p$ exist
        Set $\mathbf{v}_2 \leftarrow \mathbf{0}$
        Set $\mathbf{v}_3 \leftarrow \mathbf{r}_2 \circ (\tilde{\mathbf{s}} - \mathbf{s}_p)$
    else
        if $\dot{\mathbf{s}}_p$ exists, then set $\mathbf{v}_2 \leftarrow \mathbf{r}_1 \circ (\dot{\mathbf{s}}_p - \mathbf{s}_p)$; else set $\mathbf{v}_2 \leftarrow \mathbf{0}$.
        if $\ddot{\mathbf{s}}_p$ exists, then set $\mathbf{v}_3 \leftarrow \mathbf{r}_2 \circ (\ddot{\mathbf{s}}_p - \mathbf{s}_p)$; else set $\mathbf{v}_3 \leftarrow \mathbf{0}$.
    Set $\mathbf{v}_p \leftarrow w\mathbf{v}_1 + 1.5\mathbf{v}_2 + 1.5\mathbf{v}_3$

Update positions
for each $p$
    Set $\mathbf{s}_p \leftarrow \mathbf{s}_p + \mathbf{v}_p$
```

terms, but modified to make it into a non-convergent exploratory search. The three terms are:

1. The self-velocity term. This is set to $w\mathbf{v}_p$, as in the standard PSO. If the particle is out of bounds, however, $\mathbf{v}_p$ is set to zero to encourage it to return to the allowable search domain.

2. An attraction to the target solution, $\mathbf{t}$. This replaces the personal-best term in the standard PSO. This term remains the same in all circumstances.

3. An attraction to the global best, $\tilde{\mathbf{s}}$. This replaces the local-best term in the standard PSO. In the event that $\tilde{\mathbf{s}}$ does not exist (because no feasible solutions have yet been found), the attractor is a randomly-generated point that is the same distance from $\mathbf{t}$ as $\mathbf{s}_p$ (this point is denoted $\mathbf{R}$ in Algorithm 4.3).

The Exploit velocity update matches the canonical PSO update even more closely. For particles that are in bounds and for which $\dot{\mathbf{s}}_p$ and $\ddot{\mathbf{s}}_p$ exist, in fact, the update is identical to the standard form. Modifications are only applied in special cases that are most likely to arise in the early part of the search. First, as with Explore, particles that are outside the search bounds have their self-velocity terms set to zero. Second, the velocity terms for attraction to the personal or local bests are set to zero if the corresponding best solution does not exist (as can happen before many feasible solutions have been encountered). If it happens that neither $\dot{\mathbf{s}}_p$ nor $\ddot{\mathbf{s}}_p$ exist, then a new velocity term in the direction of $\tilde{\mathbf{s}}$ is added in their place.

4.3.3 Exchange

The exchange of particles requires first updating the objective function value of all in-bounds points, and then revising each particle's personal and local best positions. During this process, the exchange step also handles the sorting of solutions, in the same way (and for the same reason) as the `improve` algorithm. When $\hat{f}_{\mathbf{s}}$ is invariant to permutations of $\mathbf{s}$, but $\delta(\mathbf{s}, \mathbf{t})$ is *not* permutation-invariant, it is advantageous to start with the elements of $\mathbf{t}$ ordered, and to sort any feasible solutions that are found.

This will result in a reduction of $\delta(\mathbf{s},\mathbf{t})$ without a change in $\hat{f}_{\mathbf{s}}$. The main example of this is when data sharpening is used to adjust the shape of a kernel density estimator. The shape of the density estimate does not depend on the ordering of $\mathbf{s}$, but if the objective function is, for example, a norm of $\mathbf{t}-\mathbf{s}$, then its value *does* depend on order. In such a case, $\mathbf{s}$ can be improved directly by putting $\mathbf{s}$ and $\mathbf{t}$ in the same sort order.

When sorting of solutions is desirable, the Exchange step will include an initial sorting of any feasible solutions that are found. This ensures that the personal bests (and thus the local and global bests, as well) are all in sorted order. Infeasible solutions are not sorted, to reduce computational burden and to maintain swarm diversity. As long as the best solutions are in sorted order, the rest of the swarm will tend to approach a correct ordering as well.

After these preliminary activities, the Exchange step carries out two functions. First, the two swarms' lists of best solutions are brought back into agreement, keeping the combined best P personal bests from both swarms. Second, the actual exchange takes place: any feasible or promising particles from Swarm 2 are transmitted to Swarm 1, in return for that swarm's worst solutions (which are usually infeasible).

The explorer swarm is most likely to find new feasible solutions in the early stages of the search, where $\tilde{\mathbf{s}}$ is far from the global optimum and there is much room for improvement. If any feasible solutions are found, transferring them to Swarm 1 can result in a rapid decrease in the best objective function value. At later stages, Explore will become less and less able to find new feasible solutions, especially those that improve upon $\tilde{\mathbf{s}}$. Nonetheless, Explore can help Swarm 1 to avoid stagnation, by providing promising points that may move into better solution regions in subsequent steps.

The design of the algorithm is such that the number of points transferred in the Exchange step will necessarily drop to zero as Swarm 1 converges (because $\dot{\mathbf{s}}_w$ will become arbitrarily close to $\tilde{\mathbf{s}}$, meaning that no particles can be labeled as promising). As mentioned elsewhere, Explore steps are continued regardless, in the hope that improved solutions can still be found.

4.4 Examples

The following two sections return to the wind speed data and the heart disease data. Density estimation is performed with a variety of constraints chosen to illustrate the flexibility of CEPSO, as well as to demonstrate some of the characteristics of the algorithm. A population of size $P = 50$ and a neighbourhood size of $k = 10$ was used for both swarms in all of the examples. Demonstrations with other data sets have also been reported previously (Wolters, 2011).

4.4.1 Wind Speed Data

As a first demonstration of CEPSO, three different constraints were imposed on a kernel estimate of the wind speed distribution. The constraints were unimodality, type 1 bell shape, and type 3 bell shape. Shape adjustment was done by data sharpening, with the L_2 distance as objective function. Repeated runs were conducted at different bandwidths, and the bandwidth that maximised the pseudo-likelihood (h_{ML}) criterion was chosen[1]. A constraint of nonnegative support was also added to each estimate, to ensure that the results were physically meaningful even at larger bandwidths. The area under each constrained estimate to the left of zero was restricted to be less than 10^{-4}.

Figure 4.6 shows pilot and constrained estimates for each case. In the density plots, the thick grey curve is $\hat{f}^{\circ}$, and the thin black curve is $\hat{f}_{\mathbf{s}}^{M}$. As expected, the constrained density estimates become smoother as one moves from unimodality, to type 1 bell shape, to type 3 bell shape. The bandwidth is also indicated on each plot. Its value increases considerably as the constraints become more restrictive—an effect caused by the sensitivity of h_{ML} to the outlying point at a wind speed of 30.4. Looking at the left tail of the bell shaped density estimates, we can see that the constraint of nonnegative support had a noticeable impact on the shape of the estimate.

[1]Optimizations runs of 500 iterations were carried out at 20 bandwidths between $0.5h_{OS}$ and h_{OS} to find an approximate value for h_{ML}. The chosen h_{ML} value was then used in a longer run of 1500 iterations to obtain the final solution.

The figure also includes summaries of the search progress over the 1500 explore-exploit-exchange iterations. The plots in the right half of the figure have two vertical axes. The logarithmic axis on the left is used to measure the L_2 objective function value of the global best solution, and the value of a convergence diagnostic (the one based on the distance between personal best solutions, equation 4.7). The linear axis on the right is used to count the number of particle swaps made during the exchange step, which are drawn as black bars on the plots. In all three cases, solution progress follows a similar pattern. In the early stages of the search, the objective function drops rapidly and there are many particle swaps. After the first 100 iterations or so, there is little improvement made on the objective, and few particle exchanges occur. Meanwhile the set of personal best solutions converges slowly toward the vicinity of the global best. These later iterations correspond to the situation illustrated in Figure 4.5, where the promising region of the solution space has become too small to generate many particle exchanges, and there is little opportunity for improvement.

The convergence metric is not monotonically decreasing, because it is based on an elementwise maximum distance. Any new personal best that is discovered must have a lower the objective value than its predecessor, but could still increase the distance between itself and the other personal bests. If a better solution is found in a distant portion of the search space, for example, the personal bests will become spread out for a time while the swarm migrates toward the new best part of the search space. As a general rule, waiting for the exploiter swarm to converge to a single value is not a practical stopping strategy. In an unconstrained problem using the canonical PSO, the swarm will indeed converge or coalesce to a single point over time; but when constraints are present, this could take a prohibitively long time. The swarm can only converge after all of the personal best solutions have converged, and this requires each of P particles to, by chance, move to a feasible point sufficiently close to the global best solution.

In Section 3.2.1, an attempt was made to use the `improve` algorithm to find type 1 bell shaped KDEs from the wind speed data. Four bandwidths were used: 1, 1.5, 2, and 2.39. At each bandwidth, the greedy algorithm returned estimates that appeared

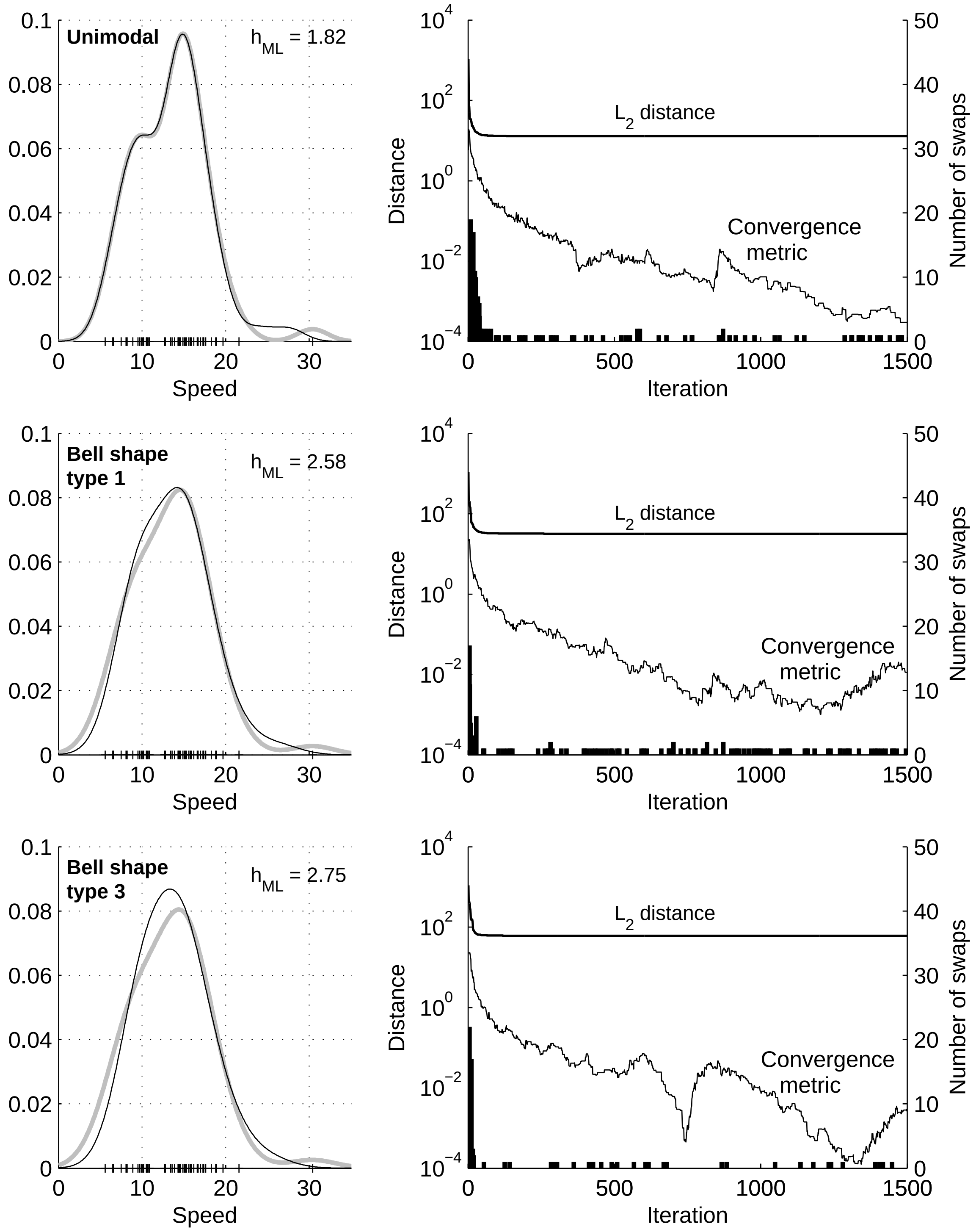

Figure 4.6: CEPSO results and solution progress for three constraints on the wind speed data.

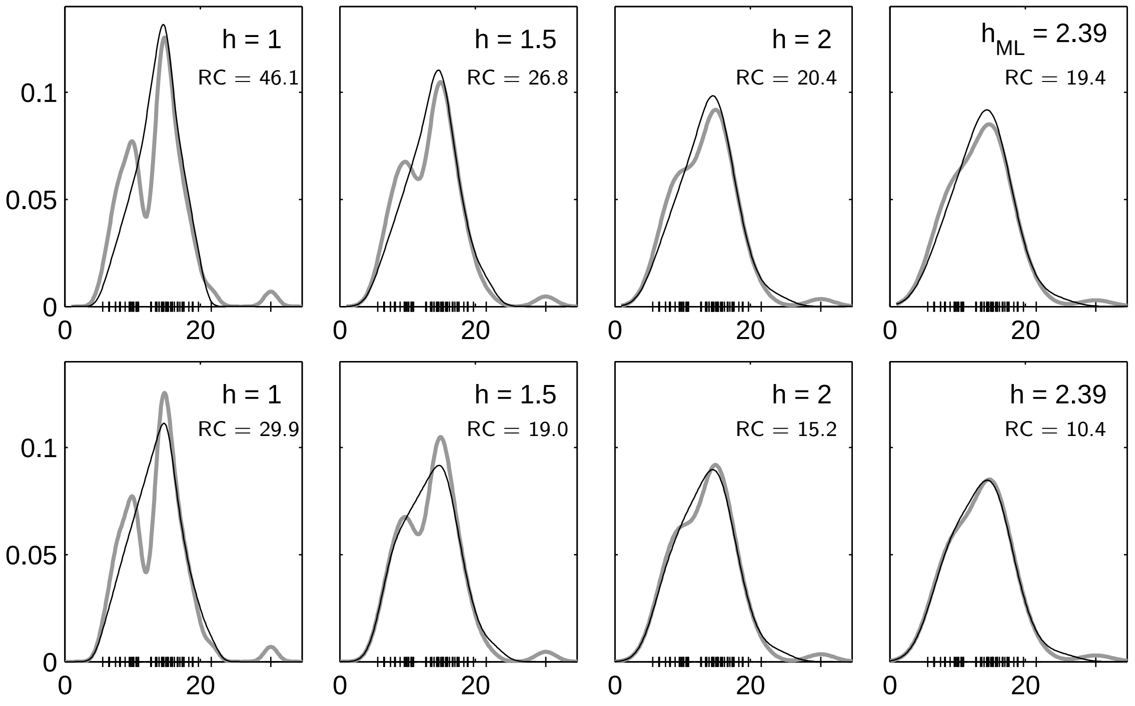

Figure 4.7: Comparing results of `improve` (top row, repeated from Figure 3.5) and `CEPSO` (bottom row) on the wind speed data with the type 1 bell shape constraint.

to be far from the best possible solution. To confirm that these results could in fact be improved upon, CEPSO was run on the same four problems. The two methods are compared in Figure 4.7. The top row of plots shows the estimates found using `improve` (reproduced from Figure 3.5). The bottom row shows the estimates found by CEPSO. The value of the $RC_{0.01}$ objective function is also given on each plot. CEPSO finds better solutions for all four bandwidths, and its bell-shaped estimates match the shape of the pilot estimates more closely.

The nearly-parametric constraint (Constraint 12) can be used as a final demonstration with the wind speed data. The normal distribution family was chosen as the desired parametric form, and the L_2 distance was used as the objective function for data sharpening. To make the problem more challenging, a pilot bandwidth of $0.75h_{SJ} = 1.16$ was used. Constrained estimates were found using CEPSO with the tolerance in equation (2.3) set to one of three different values ($\xi = 0.2$, 0.1 or 0.05),

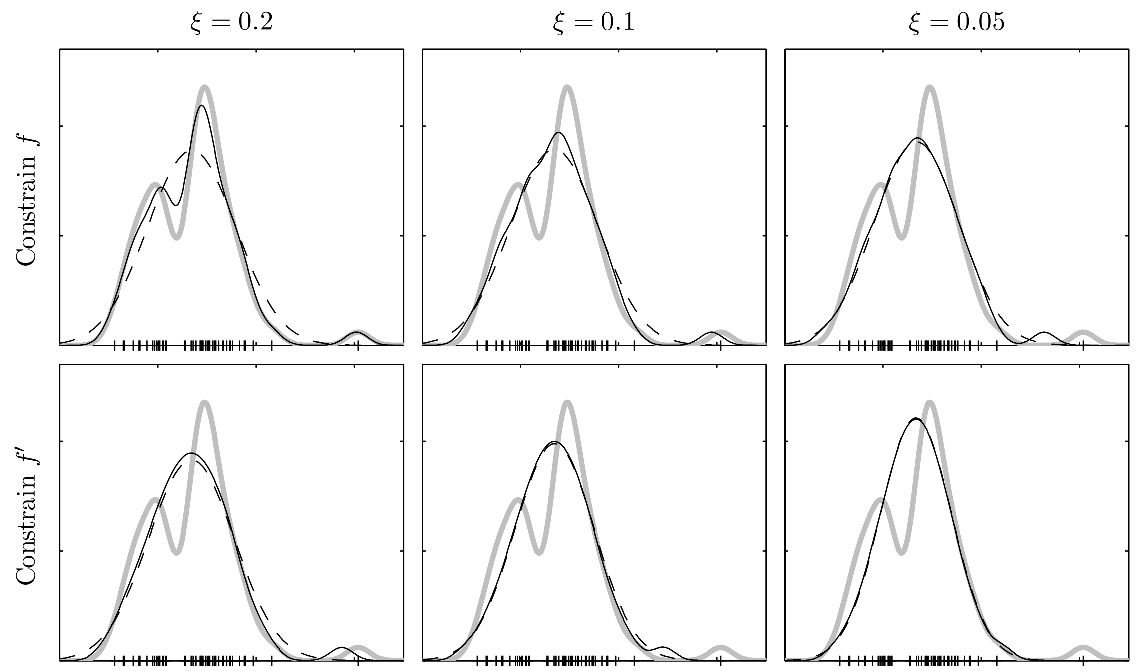

Figure 4.8: Density estimates for the wind speed data with a nearly normal constraint. In each plot the grey curve is $\hat{f}^{\circ}(x)$, the solid black curve is $\hat{f}^{M}_{\mathbf{s}}(x)$, and the dashed curve is the normal density implicit in the constraint. All axes have the same scaling.

and the constraint being based either directly on the density or its first derivative. Figure 4.8 shows the six estimates obtained, with the corresponding normal densities also plotted.

As expected, the estimates appear smoother when ξ is smaller, or when the constraint is applied to the derivative rather than to the density itself. The derivative-constrained cases in particular have an appealing qualitative similarity of shape with the normal distribution. Despite their overall smoothness, however, five of the six estimates have two or more modes. This is possible because small modes in the tail of the density only make a small contribution to the discrepancy between the KDE and the matching normal distribution. An interesting possibility is to combine the nearly normal constraint with a unimodal constraint, to build an estimator that is unimodal, but less likely to have plateaus in its estimates. It is still questionable, however, how useful such an estimator would be relative to a bell-shaped constraint that achieves similar aims without introducing the new tuning parameter ξ.

4.4.2 Heart Disease Data

For the analysis of the bivariate data set, the estimator was set up in the same way described previously for the `improve` runs of Section 3.2.2: the product-normal kernel function was used, with the normal-scale bandwidth $h_{NS} = 0.43$ used in both directions. The previous calculations were done using a binned kernel density estimator for speed, and the estimate was calculated over a 200×200 grid. For this set of trials the kernel functions were evaluated directly using the bivariate normal density function, as this was found to reduce numerical problems with the relevant constraint-checking functions. The higher accuracy of the direct calculation allowed the set of constraint-checking points to be thinned out to a 90×90 grid of locations. With this configuration, one set of KDE evaluations could be done in approximately the same time as the former binned KDE.

CEPSO was used to enforce three different constraints by data sharpening, using the L_2 objective function. The constraints were unimodality (Constraint 14, checked by applying Algorithm 2.4), unimodal conditional distributions (Constraint 16, checked with Algorithm 2.6), and convex contours (Constraint 18). The convex contours constraint was checked by inspecting each of the default contour lines (those enclosing probability mass of 0.95, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, and 0.05). The first two cases were run for 1000 CEPSO iterations, while the last one, which involves a more strict constraint, was run for 2000 iterations.

The pilot estimate and each of the constrained estimates are shown in Figure 4.9. The plots show both the sharpened and unsharpened data values, as well as the locations of any local minima or maxima present in the estimate. The unimodal and unimodal conditionals constraints were satisfied with limited movement of data points, because the pilot estimate does not contain large violations of either of these constraints. The convex contours constraint, however, could only be satisfied with larger perturbations of the data. This constraint also allowed the three spurious local optima in the high-LDL portion of the unsharpened density to persist, since these points existed outside the 0.95 contour. Each of the three constrained estimates convey the same general structure present in the pilot estimate, but with increasing degrees of smoothing in their outer contour lines.

These bivariate runs provide another opportunity to observe the convergence behaviour of the CEPSO algorithm. Figure 4.10 shows the objective function values and the values of the convergence diagnostic (4.7) as a function of iteration number, for all three estimates. In these examples, the objective function drops more gradually than in the univariate examples of Figure 4.6. It appears, in fact, that further improvements in the convex contours estimator could have been made if the search had been run for a longer time. The comments made about the convergence metric in the univariate case apply here as well—while the value of this quantity might provide some information about solution progress, it is likely impractical to use it as a stopping criterion.

The run time required to obtain these bivariate estimates was much longer than that required for the univariate problems. In each case the CEPSO search took about four hours per 1000 iterations. A similar number of iterations could be done in a few minutes with the wind speed data. This is partly due to the sample size ($n = 160$ for the heart disease data, versus 57 for the wind speed data), but is primarily a result of the extra time required to compute the bivariate estimates. Because of the prohibitive run time, no investigation of optimal bandwidth selection was performed for the heart disease data.

4.5 Simulation Studies

CEPSO uses stochastic moves, so it will not necessarily return the same solution in repeated runs, even for identical initial conditions. The repeatability of CEPSO results is the subject of two simulation studies reported below. The first study evaluates the degree of run-to-run variation in CEPSO solutions, and the sensitivity of results to the population parameters P and k. The second study repeats the first, but at different values of the PSO control parameters w, c_1, and c_2. Both of the studies are based on repeated optimizations with the same single sample of size 50, drawn from

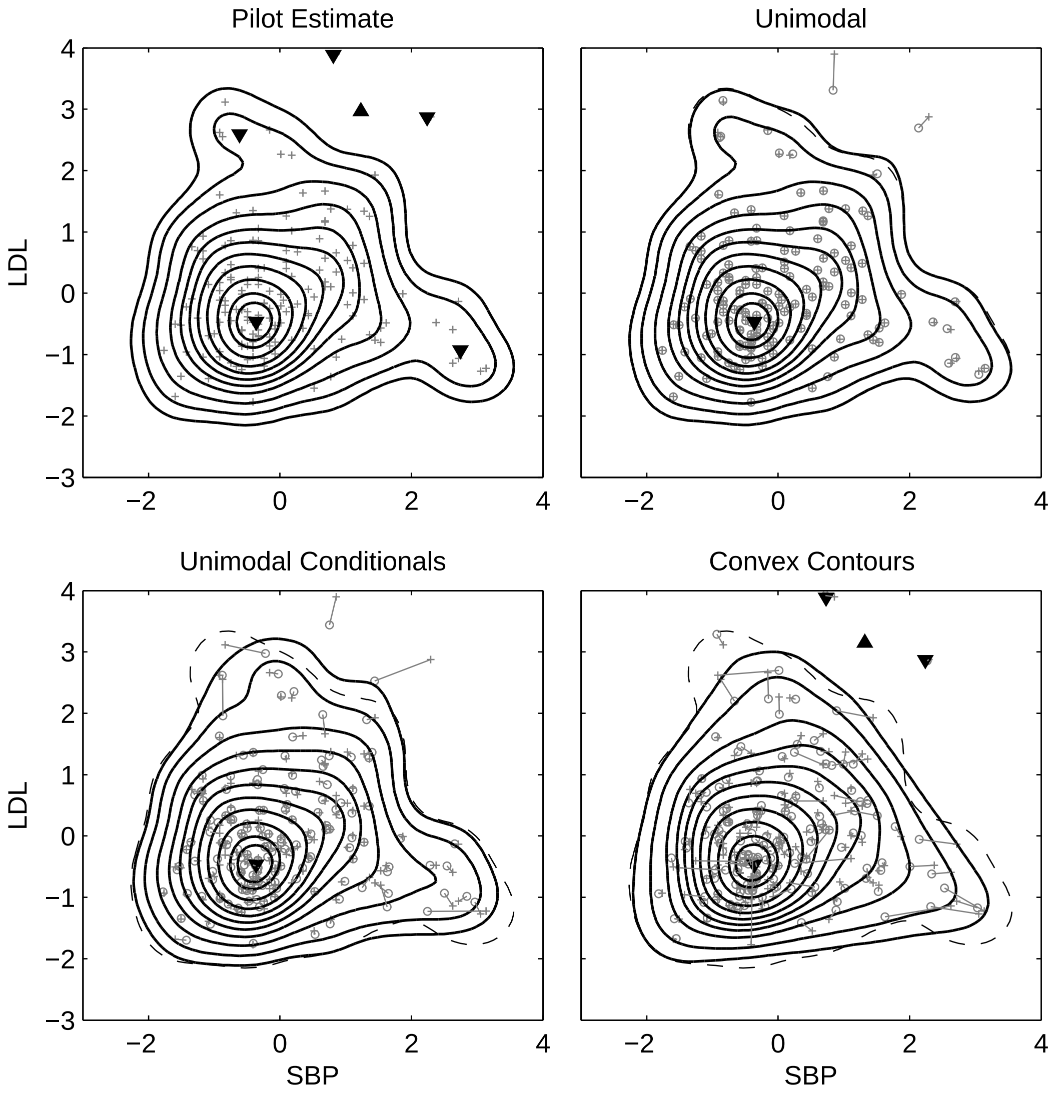

Figure 4.9: Pilot estimate and three shape-constrained estimates for the heart disease data, using CEPSO to perform data sharpening. The original and sharpened data are shown by crosses and circles respectively. Lines join the sharpened points to their target locations. Local maxima are indicated by ▼ and local minima by ▲. The outermost contour of the pilot estimate is reproduced in each plot (dashed line) to facilitate comparison.

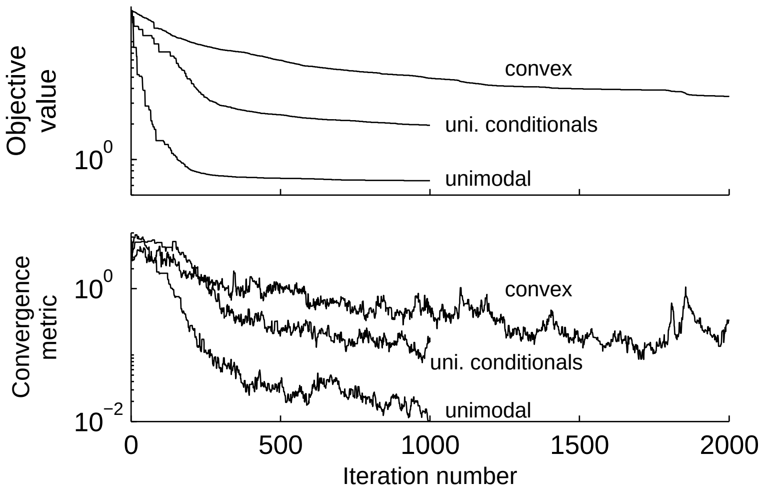

Figure 4.10: Information on CEPSO solution progress for the heart disease examples.

the t distribution with three degrees of freedom. The problem being solved in both cases is specified by the following four elements.

Estimator: the Gaussian kernel density estimator, with bandwidth $h = 0.75h_{SJ}$, where h_{SJ} is the Sheather-Jones bandwidth.

Adjustment method: data sharpening, where $\mathbf{y}$ is the sharpened data vector and the target value is $\mathbf{x}$.

Constraint: unimodality.

Objective function: the L_2 distance.

4.5.1 Run-to-Run Variability

The thick grey line in Figure 4.11 shows the pilot density estimate for the data set. These data are particularly challenging for unimodal density estimation, because there are outlying observations in both tails, as well as an extra mode (or shoulder, depending on the bandwidth) near the main peak. The relatively small bandwidth of $0.75h_{SJ}$ was chosen to accentuate the difficulty of the problem.

To examine the performance of CEPSO, optimization was repeated 100 times at all four combinations of two population sizes ($P = 25$ and $P = 50$) and two

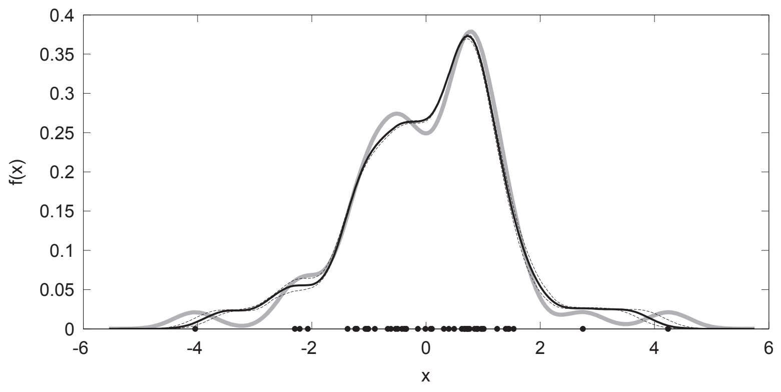

Figure 4.11: The unconstrained estimate (thick grey curve) for a t_3 data set and the best unimodal adjustment found (black curve). The dashed lines indicate the run-to-run variation in best solution found.

neighbourhood sizes ($k = \frac{1}{5}P$ and $k = \frac{2}{5}P$). The optimizations with $P = 25$ and $P = 50$ were run for 4000 and 2000 iterations, respectively, to make them roughly equivalent in terms of computational effort. Randomly-generated starting solutions were used, since the exploration moves were able to find feasible solutions quite quickly for this case.

The solid black curve in Figure 4.11 is the best unimodal sharpened KDE found over all 400 optimizations. The unimodality constraint has been satisfied by shifting the data points to turn the extra modes into shoulders and plateaus. The dashed lines around the best solution give the pointwise 2.5% and 97.5% percentiles of the set of 400 solutions. They show that the results in the central part of the distribution are almost identical for all runs, with greater run-to-run variation in the tails, where the solution depends on the precise location of only a few sharpened points.

Table 4.1 provides information on the variability of the best objective function values found at each combination of P and k. The objective function value for the overall best solution was 1.131, and the table shows that the algorithm was able to consistently find solutions in the vicinity of this overall best, despite not converging to exactly the same solution each time. As illustrated in Figure 4.11, the variation among solutions does not translate into an appreciable difference in the density estimates, especially considering the uncertainty in the estimate due to bandwidth selection and

Table 4.1: Mean, standard deviation, extremes, and quartiles of L_2 objective function values for repeated runs on a t_3 data set, at four different (P, k) combinations.

(P, k)	mean	SD	min	Q1	Q2	Q3	max
(25,5)	1.200	0.057	1.136	1.160	1.184	1.219	1.403
(25,10)	1.220	0.134	1.134	1.160	1.184	1.225	2.164
(50,10)	1.179	0.046	1.132	1.145	1.163	1.204	1.358
(50,20)	1.201	0.169	1.131	1.146	1.164	1.196	2.583

sampling variation.

The runs with larger neighbourhoods ($P = 25, k = 10$ and $P = 50, k = 20$) were more likely to prematurely converge to poor solutions, and this is reflected in the larger maximum objective values and standard deviations for those two cases in Table 4.1. This is consistent with the expected behaviour of particle swarms, where greater information-sharing among particles tends to promote convergence. Nevertheless, typical solutions found by all four combinations of P and k were close enough to the overall best to be practically useful. This small study was the basis for choosing the values $P = 50$ and $k = 10$ as the default inputs to the algorithm.

4.5.2 Sensitivity to Swarm Control Parameters

The CEPSO algorithm considers the swarm control parameters w, c_1, and c_2 to be fixed quantities. Their values ($w = 0.73$ and $c_1 = c_2 = 1.5$) are based on the canonical PSO. While it is natural to wonder how changing w, c_1, and c_2 might influence CEPSO performance, there are a few reasons why it is inadvisable (or at any rate, difficult) to modify the default scheme:

1. In the standard PSO, the relative sizes of c_1 and c_2 determine the trade-off between exploration and exploitation. In CEPSO, exploration and exploitation are explicitly assigned to the two swarms. So there is little motivation to explore parameter settings with $c_1 \neq c_2$.

2. The optimal parameter settings will be problem dependent, especially when considering cases with $c_1 \neq c_2$.

3. There is theoretical and empirical evidence that the default parameter settings (which are consistent with the constriction coefficient approach) will not lead to divergent particle velocities (Poli et al., 2007, p. 37). For this reason, CEPSO does not include *velocity clamping* (upper bounds on particle speeds), a device that appears in other PSO variants. Other parameter settings might cause the swarms to diverge if velocity clamping is not in place. Note that the introduction of clamping would effectively introduce a new adjustable parameter, the maximum speed in each coordinate.

4. CEPSO includes some adaptation of the parameters, where the value of w is set to a lower or higher value (0.5 or 1.0) depending on the spatial extent of the swarm. It is not clear how one should modify this adaptation scheme when exploring different combinations of w, c_1 and c_2.

Attempts to run CEPSO with arbitrary parameter combinations would, then, require additional modifications to the algorithm, and any proposed settings would have to be tested on a wide range of problems. Additionally, the three control parameters could in principle be varied independently for both swarms. A thorough examination of these possibilities is beyond the scope of this work.

While it may not be advisable to run CEPSO with arbitrary w, c_1, and c_2 levels, some theoretical results can be used to suggest reasonable combinations of the parameters. Poli et al. (2007, section 5) and Engelbrecht (2005, chapter 13) summarize a number of theoretical investigations into swarm dynamics. One useful result (Engelbrecht, 2005, p. 158) gives the following condition for a convergent swarm, based on non-stochastic theoretical analyses of swarm dynamics:

$$0 \leq \frac{1}{2}(c_1 + c_2) - 1 < w < 1.$$

Setting $w = 0.73$ to avoid changing the CEPSO adaptation, and maintaining $c_1 = c_2$ as in the default case, this leads to

$$c_1 = c_2 \in [1, 1.73]$$

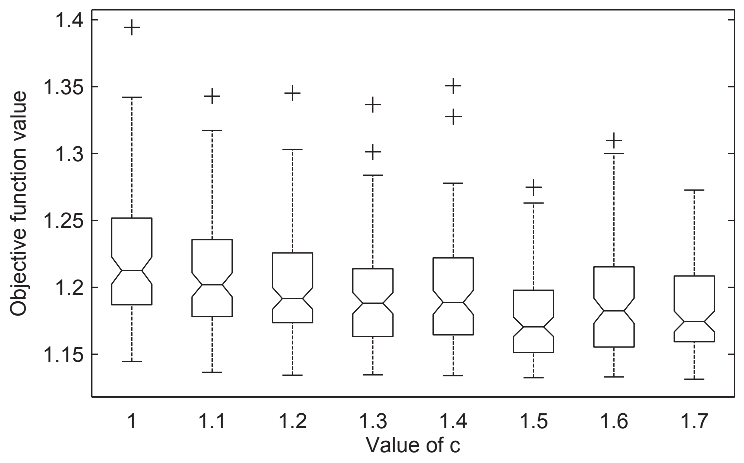

Figure 4.12: Boxplots of final objective function value for 100 CEPSO runs using the t_3 example, for eight values of c.

as a range of plausible $c \equiv c_1 = c_2$ values that should be compatible with the existing CEPSO framework.

To explore how sensitive CEPSO is to changes in c, we can use the same unimodal t_3 example of the preceding section, keeping $P = 50$ and $k = 10$ and varying c this time. The optimization was repeated 100 times each for eight levels of c in the suggested range: 1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, and 1.7. CEPSO was run for 1000 iterations each time.

Figure 4.12 shows, for each level of c, a boxplot of the objective function values returned by CEPSO. The method appears reasonably robust to changes in c, though with a slight shift toward worse results at the lowest c values. Figure 4.13 helps to interpret the practical significance of the differences observed in the box plots. It shows the best estimate and the pointwise 2.5% and 97.5% percentiles of the sharpened estimates for the worst-performing case, $c = 1$. Even at a less favorable value of c, the variation among the estimates is small.

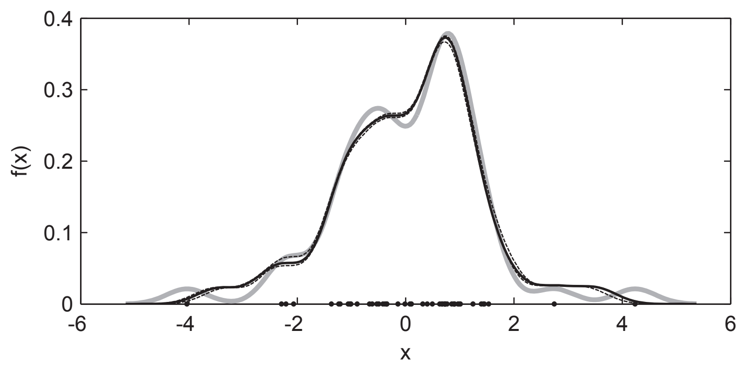

Figure 4.13: Best estimate found, and pointwise variation among the estimates, for the t_3 data set with c=1.

4.6 Limitations and Extensions

This chapter presents one way of implementing a particle swarm search for shape-constrained nonparametric estimation. The main advantage of the proposed algorithm is its broad applicability. Though the demonstrations of the method were limited to match the scope of this work, CEPSO can be applied not only to KDEs with different constraints, but also to different estimators, adjustment methods, and objective functions. The algorithm has only two adjustable parameters, the population size and the neighbourhood size, and it can find solutions for constraints that would be intractable for traditional methods of optimization.

The algorithm is not without limitations, however, and in its present form it should be viewed as a proof of concept, demonstrating that a general heuristic for constrained nonparametric estimation is possible. We will now review the current limitations of the method, together with some possible remedies.

One challenge that limits the number of estimators CEPSO can handle arises when the estimator includes an equality constraint on the adjustable values. Sum constraints (of the form $\sum_{i=1}^{n} s_i = \theta$) can be particularly important. The weights in the estimator $\hat{f}^W_{\mathbf{w}}$, for example, must sum to unity; similarly the coefficients in the adjustment-curve estimator $\hat{f}^A_{\boldsymbol{a}}$ (as defined in the next chapter) must sum to zero. CEPSO in its present form does not work in these cases, because the feasible set $\mathcal{C}$ has zero volume, being of smaller dimension than the search space. Normal CEPSO moves will never be able to find feasible solutions.

Two possible solutions for this problem have met with some success in limited testing. An approach used by Paquet and Engelbrecht (2003) involves replacing the vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ in the velocity updates by scalar values r_1 and r_2. Doing this ensures that all subsequent particle positions will satisfy the sum constraint, as long as all initial solutions satisfy it as well. Alternatively, each solution can be projected orthogonally onto the subspace defined by $\sum s_i = 0$, and then θ/n can be added to each element of $\mathbf{s}$ to achieve appropriate sum. This approach permits the normal swarm dynamics to take place at each move, and maintains the sum constraint with only a single added step.

Finding feasible starting solutions is another potential difficulty. In some cases simple ways of constructing feasible solutions will be readily apparent. Kernel density estimates, for example, can be made to satisfy constraints by putting all points at the same location (for data sharpening), by making the bandwidths sufficiently large (for variable bandwidths), or by setting a sufficient number of weights to zero (for variable weights)—as long as the kernel function itself satisfies the constraints. Few recommendations for finding feasible solutions can be made in general, however, since appropriate means of achieving feasibility will vary from problem to problem.

The algorithm does not in principle require feasible solutions to run. In the absence of feasible starting points both swarms will perform Explore moves until a feasible particle arises. The wait for this to happen may be impractically long, however, when the volume of $\mathcal{C}$ is small compared to the overall search space.

Another limitation is the potential for inefficiency when the global best solution or the target are close to the search boundaries. This problem arises, for example, when some of the optimal weights in a variable-weight estimate are nearly zero. In such a case, particles in both swarms will spend a high proportion of their time out of bounds, and better solutions may never be found, or the solution may be improved only slowly.

Adding penalty functions to the objective could alleviate problems caused by an excess of infeasible or out-of-bounds particles. Penalty functions would be used to evaluate approximately how far a solution is from $\mathcal{C}$, allowing infeasible candidates to be ranked and handled accordingly. The movement of particles in Swarm 2, for example, could be biased to favor directions of decreasing penalty, making the Explore moves much more likely to discover feasible solutions. This would also ameliorate the problem that the Explore moves become increasingly futile as the final optimum is approached. Another alternative is to include a repair function (one that takes infeasible solutions and makes them feasible, or takes out-of-bounds particles and returns them to admissible space). This would also reduce the proportion of moves wasted on inappropriate solutions.

It may not be possible, of course, to define penalty functions or repair functions for all scenarios, which is why they have not been considered in the development of CEPSO to this point. A logical next step in development would be to make such functions optional user-supplied inputs, and write the swarm movement rules to take advantage of this extra information when it is available. Alternatively, one could attempt to solve difficult problems by running CEPSO on a sequence of problems with successively tightening constraints.

Computational efficiency could be further improved by accelerating convergence in the later stages of search. A local search step could be added, whereby the personal best solutions attempt to improve themselves (for example, by moving some coordinates in the direction of $\mathbf{t}$ until the constraint boundary is found). It is possible that the greedy algorithm of Chapter 3 could be useful either as this type of local search, or as a repair function. Various forms of adaptation could also be considered to better control the exploration/exploitation trade-off. The neighbourhood size, for instance, could be changed based on the number of particles that are feasible; the relative sizes of Swarm 1 and Swarm 2 could be modified; or a different means of adapting the swarm parameters w, c_1, and c_2 could used.

A final remark is reserved for the topic of run time. The execution time of the PSO runs reported here varied from minutes (for the t_3 data) to several hours (for the heart disease data) on a laptop computer. The run time is dominated by the time required to evaluate the estimator, however. Efficient implementations of the pilot estimator will result in more efficient constrained estimates. All of the estimators used here (and the PSO code itself) were written in the interpreted MATLAB language (The Mathworks, Inc., 2007). Run times could be significantly improved if the estimators were written in a low-level compiled language.

Chapter 5

Optimal Adjustment Curves by Quadratic Programming

The shape-constrained density estimates developed in this chapter differ from those of the previous chapters in two important ways. First, the method of shape adjustment is different: this chapter uses adjustment curves, while the preceding two chapters used data sharpening. Second, the optimization problems of this chapter are solved by mathematical programming, rather than by heuristic methods. The adjustment curve and the objective function are formulated in such a way that several important constraints can be expressed as quadratic programs, for which globally optimal solutions can be found.

5.1 The Method

Let $\Psi(x)$ be the adjustment curve. It is a function used in an additive manner to correct any constraint violations in the pilot estimate $\hat{f}^{\circ}(x)$. The proposed estimator is

$$\hat{f}^{A}(x) = \hat{f}^{\circ}(x) + \Psi(x). \tag{5.1}$$

The goal in constructing $\Psi(x)$ is to bring the estimate into conformance with the constraints, with minimal modification of the pilot density. For $\hat{f}^{A}(x)$ to be a density, the adjustment curve must preserve the non-negativity and unit area properties of the pilot estimate. That is,

$$\Psi(x) \geq -\hat{f}^{\circ}(x), \quad \forall x \tag{5.2}$$

for non-negativity, and

$$\int_{-\infty}^{\infty} \Psi(x) dx = 0 \tag{5.3}$$

for integration to unity.

The new method is founded on the idea of letting $\Psi(x)$ be a linear combination of k density functions, $\psi_i(x)$, $i = 1, \ldots, k$. The $\psi_i(x)$ are called *adjustment densities.* Using this construction, the constrained estimator is

$$\begin{aligned} \hat{f}_{\boldsymbol{a}}^{A}(x) &= \hat{f}^{\circ}(x) + a_1\psi_1(x) + a_2\psi_2(x) + \cdots + a_k\psi_k(x) \\ &= \hat{f}^{\circ}(x) + \boldsymbol{a}^T\boldsymbol{\psi}(x), \end{aligned} \tag{5.4}$$

where $\boldsymbol{a} = [a_1 \cdots a_k]^T$ are the coefficients of the combination, and the notation $\boldsymbol{\psi}(x) = [\psi_1(x) \cdots \psi_k(x)]^T$ allows the estimator to be expressed in convenient vector form. Estimator (5.4) fits into the general constrained estimation framework of Section 1.2. The vector of adjustable values to be optimized is $\boldsymbol{a}$, and the target value, at which $\Psi(x) = 0$ everywhere and the pilot estimate is reproduced, is $\boldsymbol{a} = \mathbf{0}$.

The value of k, and the location and scale of each ψ_i, determine which $\Psi(x)$ curves are possible. A proposal for setting up the ψ_i is given in Section 5.1.3. For the moment it is sufficient to take the ψ_i as given, and assume only that they are arranged such that a solution to the constrained estimation problem exists. The next section shows how, for an appropriately chosen objective function and constraint, the globally optimal $\boldsymbol{a}$ can be found using quadratic programming.

5.1.1 A Quadratic Objective and Linear Constraints

A number of possible objective functions could be considered to determine the best adjustment curve from among all possibilities. A natural choice is to use the L_2

distance between $\boldsymbol{a}$ and its target:

$$L_2(\boldsymbol{a}, \mathbf{t}) = L_2(\boldsymbol{a}) = \boldsymbol{a}^T \boldsymbol{a}, \tag{5.5}$$

which takes this particularly simple form since the target is the zero vector. An alternative is to use the integrated squared error between $\hat{f}^A_{\boldsymbol{a}}$ and $\hat{f}^\circ$ as the objective:

$$\text{ISE}(\boldsymbol{a}) = \int_{-\infty}^{\infty} (\hat{f}^A(x) - \hat{f}^\circ(x))^2 dx = \int_{-\infty}^{\infty} \boldsymbol{a}^T \boldsymbol{\psi}(x) \boldsymbol{\psi}(x)^T \boldsymbol{a} \; dx. \tag{5.6}$$

Either of these two objectives are suitable for quadratic programming, as both are quadratic forms in $\boldsymbol{a}$. For reasons to be explained in Section 5.1.3, the L_2 objective is used henceforth.

The optimal set of coefficients $\boldsymbol{a}^*$ can now be defined as the solution to the optimization problem

$$\boldsymbol{a}^* = \underset{\boldsymbol{a} \in \mathcal{C}}{\operatorname{argmin}} \;\; \boldsymbol{a}^T \boldsymbol{a} \quad \text{subject to} \begin{cases} \sum_{i=1}^k a_i = 0 \\ \hat{f}^\circ(g) + \boldsymbol{a}^T \boldsymbol{\psi}(g) \geq 0, \qquad g \in \{g_1, \ldots g_G\}, \end{cases} \tag{5.7}$$

where as before, $\mathcal{C}$ is the set of coefficients for which the adjusted estimate satisfies the shape constraints. Two constraints on $\boldsymbol{a}$ are included in (5.7) to ensure that $\hat{f}^A_{\boldsymbol{a}}$ is a *bona fide* density. The first is a sum constraint on the a_i, ensuring that $\Psi(x)$ satisfies (5.3) by integrating to zero. The second is a set of G inequalities, effective at the grid points in $\mathbf{g} = [g_1 \cdots g_G]^T$, to ensure non-negativity as in (5.2).

Except for the requirement that $\boldsymbol{a} \in \mathcal{C}$, problem (5.7) is a quadratic program—it has a quadratic form as its objective function, and constraints that are linear in $\boldsymbol{a}$. Quadratic programs can be readily solved in most statistical computing environments. In the present case the quadratic objective is positive definite, so quadratic programming (QP) should return the globally optimal solution.

It is shown below that, for several important constraints, the $\boldsymbol{a} \in \mathcal{C}$ restriction can also be expressed as a system of linear equalities and inequalities in $\boldsymbol{a}$, preserving the favorable QP structure.

5.1.2 Constraints Fitting the QP Framework

Shape constraints are to be imposed by enforcing them at $\mathbf{g}$, a vector of constraint-checking points, in the same manner as the non-negativity constraint in (5.7). A reasonable choice is to let $\mathbf{g}$ be an evenly-spaced grid of values extending beyond the minimum and maximum values of $\mathbf{x}$, and consisting of $G = 100$ points or more (a default rule for setting G is given in the next section).

The rth derivative of the adjusted estimate, evaluated at g_i, is

$$\hat{f}_{\boldsymbol{a}}^{A(r)}(g_i) = \hat{f}^{\circ(r)}(g_i) + \boldsymbol{a}\boldsymbol{\psi}^{(r)}(g_i),$$

which is linear in $\boldsymbol{a}$. Thus any shape constraints involving only linear restrictions on $\hat{f}_{\boldsymbol{a}}^{A}$ or its derivatives will be linear in $\boldsymbol{a}$, and so expressible in a form suitable for QP. Enforcing the constraints at all points in $\mathbf{g}$ will produce a system of G equalities or inequalities, each involving k coefficients. Of the univariate constraints given in Section 2.2, the following can be expressed in this manner.

- M modes, with modes at $m_1, \ldots, m_M$, and inter-mode minima at $u_1, \ldots, u_{M-1}$. Unimodality is a special case, requiring the only the single point m_1 to be specified.

- b inflections, at $v_1, \ldots, v_b$. Bell shape (type 1) is a special case of this constraint with $b = 2$.

- Two shoulders, with mode at m and inflection points at $v_1, \ldots, v_6$.

- Bell shape (type 2), with inflections at L and R and inflections of f' at $v_1 < L$ and $v_2 > R$.

- Bell shape (type 3), with inflections of f' at v_1, v_2, and v_3.

- Monotonic increasing or decreasing on the interval (g_1, g_G).

- nonnegative support with tolerance ϵ.

- Symmetry with known point of symmetry S and tolerance ϵ.

Appendix B gives the details of setting up such constraints in a manner suitable for QP software. Also note that it is not difficult to apply multiple constraints from the above list simultaneously, for example to achieve a symmetric and unimodal estimate.

Most of the above constraints only satisfy the QP structure if one or more important points such as the mode, point of symmetry, or inflection points are known. This is exactly the situation encountered when using SQP for data sharpening, as discussed previously. The need to search for the best combination of the important points adds complexity to the problem and destroys any guarantee of global optimality in practical application. Quadratic programming problems possess the advantage, however, that they may be solved more quickly and with fewer numerical difficulties than SQP problems. Consequently more effort can be directed toward finding the best values of the important points, and good constrained estimates can be found as long as the number of important points is not too large.

The following approach is recommended. Let the number of important points be r, and label the points from left to right in ordered sequence $v_1 \leq v_2 \leq \ldots \leq v_r$. Let v_0 and v_{r+1} be lower and upper bounds for the search, respectively. When $r = 1$, the best estimate may be found by performing a one-dimensional minimization of the QP objective as a function of v_1, over the interval (v_0, v_2). For $r > 1$, a good solution can be found by iteratively optimizing each v_i over (v_{i-1}, v_{i+1}), and stopping when no improvement can be made. This one-at-a-time approach is summarized in Algorithm 5.1. Any one-dimensional, gradient-free minimizer (such as golden section search) can be used for the minimization step at line A; each evaluation of $\mathtt{OF}(v_i;\boldsymbol{v})$ at that step requires the quadratic program to be solved for a particular value of v_i.

5.1.3 Choosing the Adjustment Densities

The estimator $\hat{f}^A_{\boldsymbol{a}}(x)$ can be constructed in a manner suitable for solution by QP, but the quality of the solution obtained (and the existence of a solution in the first place) depends on the number of adjustment densities used, and on the location and scale of each. To perform its function well, the adjustment curve should be smooth, but

Algorithm 5.1: A minimizer for selecting r important points (`findpoints`).

```
Input: Objective function OF($\boldsymbol{v}$); Initial guess $\boldsymbol{v}_0$; bounds $v_0$ and $v_{r+1}$.
Output: Solution $\boldsymbol{v}^*$.
Notation: Let OF($v_i$;$\boldsymbol{v}$) be the objective viewed as a function of $v_i$ only, with
other elements of $\boldsymbol{v}$ fixed.

Set $\boldsymbol{v} \leftarrow \boldsymbol{v}_0$
repeat
    for $i = 1, \dots, r$
        Let $v_i \leftarrow$ the minimizer of OF($v_i$;$\boldsymbol{v}$) over bounds $(v_{i-1}, v_{i+1})$      A
until The loop has completed with no changes to $\boldsymbol{v}$.
Set $\boldsymbol{v}^* \leftarrow \boldsymbol{v}$
```

still have a high degree of shape flexibility over the support of the density—enough that $\hat{f}^A_{\boldsymbol{a}}$ can take shapes ranging from sharp peaks to completely flat sections. One way of achieving this is to let the ith adjustment density be a $N(\mu_i, \sigma_i^2)$ density,

$$\psi_i(u) = \frac{1}{\sigma_i}\phi\left(\frac{u-\mu_i}{\sigma_i}\right), \tag{5.8}$$

which is a particularly convenient choice when $\hat{f}^\circ$ is a Gaussian KDE. Good performance of $\Psi(x)$ can then be ensured by appropriate choices of (μ_i, σ_i), $i = 1, \dots, k$. Two options appear most natural.

Option 1. Make the ψ_i equal to the kernel functions used to produce the pilot KDE. In this option, $k = n$ and the ith adjustment density has parameters $\mu_i = x_i$ and $\sigma_i = h$. In effect, each adjustment density is assigned to one data point and serves to increase or decrease the contribution of the kernel at that point. With the $\{\psi_i\}$ matched to the KDE in this way, $\hat{f}^A_{\boldsymbol{a}}(x)$ is

$$\begin{aligned}\hat{f}^A_a(u) &= \hat{f}^\circ(u) + \sum_{i=1}^n a_i\psi_i(u) \\ &= \frac{1}{nh}\sum_{i=1}^n \phi\left(\frac{u-x_i}{h}\right) + \sum_{i=1}^n \frac{a_i}{h}\phi\left(\frac{u-x_i}{h}\right) \\ &= \frac{1}{h}\sum_{i=1}^n \left(\frac{1}{n}+a_i\right)\phi\left(\frac{u-x_i}{h}\right),\end{aligned} \tag{5.9}$$

which is equivalent to the variable-weight estimator $\hat{f}_{\mathbf{w}}^{W}(x)$, with $w_i = \frac{1}{n} + a_i$. This shows that the variable weight estimator can also be found using QP for the constraints listed in Section 5.1.2.

Figure 5.1 shows what the estimate looks like using this arrangement of adjustment densities. The data in the figure are a random sample of size 50 from a lognormal distribution. The pilot estimate (with $h = 0.75h_{SJ}$) is trimodal with an outlying point. Optimal unimodal estimates are shown for both the ISE and L_2 objective functions.

The figure illustrates the advantages of constructing $\Psi(x)$ in this way. The weight interpretation of $\boldsymbol{a}$ is an advantage in itself. Also, the adjustment curve is able to perfectly annihilate any unwanted features of the pilot density (as with the outlying mode in this example), because the adjustment densities are equal to the kernel functions. Simplicity is another advantage, since k and $\{\mu_i\}$ are fixed by the data, and choosing the pilot bandwidth determines $\{\sigma_i\}$.

Several important disadvantages of this construction are also apparent in the estimates. First, in some circumstances it may be necessary to give points zero weight ($a_i = -\frac{1}{n}$) in order to find a feasible solution. This is the case for the outlying point in Figure 5.1. It is not possible for the constrained estimator to extend its right tail all the way out to this outlier. Second, this method inherits the problem of variable-weight estimators, that a local adjustment in one region of the curve may require compensatory adjustment in a distant region. In the figure, this effect is more obvious when the $\boldsymbol{a}^T\boldsymbol{a}$ objective is used. The fact that the two objective functions produce such different estimates is also discouraging, as both options should promote selection of solutions that are close to the target (pilot) density. Finally, the adjustment densities may become unnecessarily concentrated in the high-density regions of the curve. This becomes increasingly inefficient as n grows, and could cause ill-conditioning of the coefficient matrices used by the QP solver.

Option 2. Set the ψ_i to be identical, overlapping densities on a grid.

The second natural choice is to let all the adjustment densities have the same standard

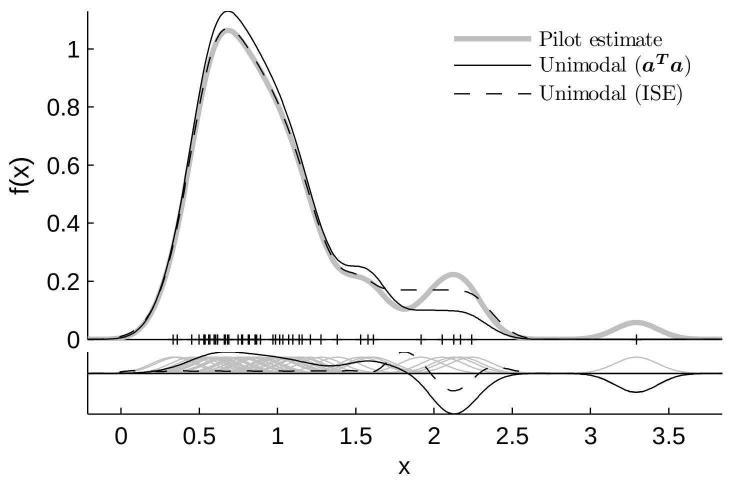

Figure 5.1: A density adjusted to satisfy the unimodality constraint, with the adjustment densities located at the data points. The top plot shows the pilot estimate and the unimodal estimates. The bottom plot shows the set of adjustment densities (scaled down to fit on the plot), with the adjustment curves superimposed.

deviation σ, and locate them on an evenly-spaced grid. Let l and u be lower and upper bounds for the grid, selected so that (l, u) extends beyond the data in either direction[1]. Then the set of densities is fixed by specifying k and σ. As a rule of thumb, it is proposed to use

$$k = \left\lceil \frac{2(u-l)}{h} \right\rceil \qquad \text{and} \qquad \sigma = \frac{u-l}{k-1} \equiv \Delta, \tag{5.10}$$

where $\lceil\ \rceil$ represents the ceiling function and Δ is the grid spacing. With this rule, the adjustment densities are centered at $\mu_i = l + (i-1)\Delta$, $i = 1, \ldots, k$.

The logic behind recommendation (5.10) is as follows. Take l and u as given. The set of adjustment densities must be able to reproduce the pilot pdf to within some tolerance, otherwise $\Psi(x)$ would not be able to eliminate unwanted features of the density. So the grid must be dense enough that every data point is close to a grid point μ_i. The bandwidth h can be taken as a measure of closeness, so a grid spacing of approximately $\frac{h}{2}$ should be sufficient. The grid spacing is $\Delta = \frac{u-l}{k-1}$, so ideally one

[1]Setting $l = x_{(1)} - 4h$ and $u = x_{(n)} + 4h$ would seem reasonable.

would choose

$$\frac{u-l}{k-1} = \frac{h}{2} \qquad \Rightarrow \qquad k = \frac{2(u-l)}{h} + 1.$$

The value suggested in (5.10) results by noting that $2(u-l)/h \gg 1$ and that k must be an integer.

With the values of k and Δ thus determined, we set $\sigma = \Delta$ to ensure that the $\psi_i(x)$ overlap to an appropriate degree. A trade-off exists in the choice of σ. If it is made too large, the adjustment densities will overlap too much, and the adjustment curve will be too smooth—unable to make rapid local changes of shape. The numerical performance and speed of the QP solver is also adversely affected in this case. If σ is too small, on the other hand, the adjustment curve (or its derivatives, which are used in the constraints) will be insufficiently smooth, and the solver might not be able to find a solution. Experience has shown that setting $\sigma = \Delta$ provides a good compromise between these two extremes.

Figure 5.2 demonstrates the results of this construction of $\Psi(x)$ on the lognormal-data example of Figure 5.1. In this case the adjustments to the pilot density are confined to those regions near the constraint violations, and the adjusted estimate does extend out to the outlying point. Also, the two different objective functions return nearly indistinguishable solutions. This is a consequence of defining the adjustment densities in this way, and the agreement between $ISE(\boldsymbol{a})$ and $L_2(\boldsymbol{a})$ improves as n or k grow (see Appendix B). Given these appealing characteristics, the grid construction for $\Psi(x)$ with rule of thumb (5.10) is used from this point forward.

When using this rule of thumb for setting up the adjustment densities, it is also important to ensure that G, the number of constraint checking points, is sufficiently large. If G is too small, then some adjustment densities might fall between points in $\mathbf{g}$, and the corresponding a values will have no influence on the constraints inside the QP solver. This can lead to solutions with unintended constraint violations. A default setting of $G = 2k$ is recommended to avoid this problem. This default is used in all of the examples and simulations to follow.

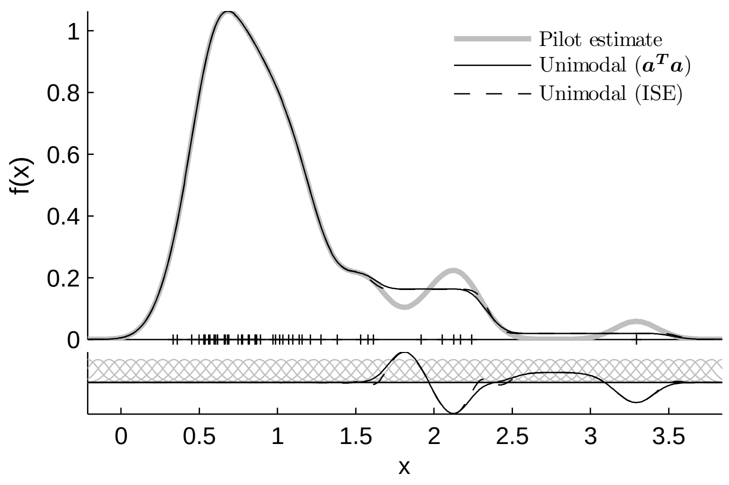

Figure 5.2: The example of Figure 5.1, but with the adjustment densities located on a grid. The rule of thumb (5.10) chose $k = 60$ for these data.

5.2 Examples

The characteristics of $\hat{f}^A_{\boldsymbol{a}}$ may be further explored using the wind speed data set. The estimator can also be used on bivariate data, as demonstrated on the heart disease data.

5.2.1 Wind Speed Data

Figure 5.3 shows the pilot and shape-adjusted estimates for the wind speed data at four different bandwidths. The operative constraints in the figure are unimodality and nonnegative support (the density was restricted to be less than 10^{-6} for $x < 0$). The constrained estimates' mode locations were chosen to minimize the $\boldsymbol{a}^T\boldsymbol{a}$ objective; in all cases the constrained mode matched the location of the highest mode in the pilot estimate.

For each pilot bandwidth, the estimator achieves unimodality by flattening out the density across any constraint violations. The estimate looks increasingly like a step function as h gets smaller and the number of constraint violations grow. This illustrates how $\hat{f}^A$ does not necessarily inherit the smoothness of the pilot KDE, because the adjustment curve operates over the whole line, and not just at the data points. Such behaviour is in contrast with the data sharpening estimator $\hat{f}^M$, which

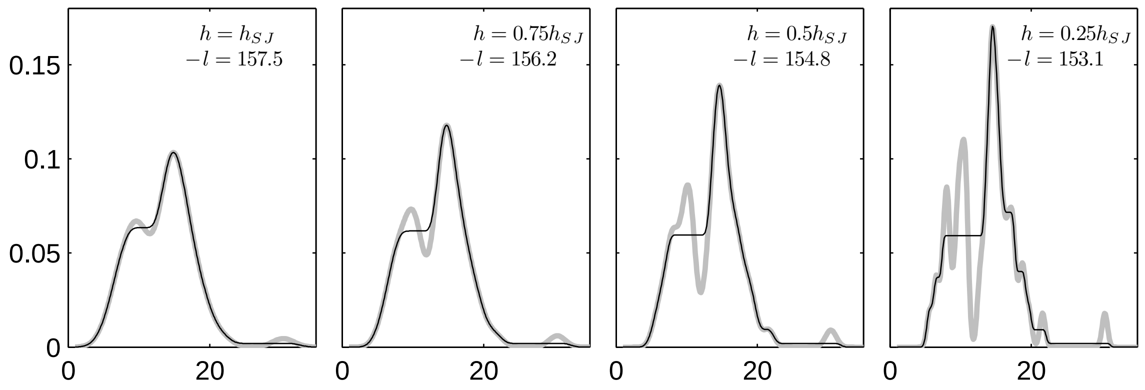

Figure 5.3: Unimodal estimates for the wind speed data at different bandwidths, using the method of adjustment curves. Each plot shows the pilot estimate (grey) and the adjusted estimate (black). Labels on the plots give the bandwidth and the negative log likelihood of the data under the estimate.

is just as smooth as the pilot estimate because it uses the same bandwidth. When using an adjustment curve, then, we can expect the choice of constraints to play a bigger role in determining the qualitative smoothness of the estimate than it would with another approach to shape adjustment.

This example also demonstrates a case where the proposed h_{ML} bandwidth will not work well. As h gets smaller, the adjustment curve gets increasingly more shape flexible, allowing the constrained estimate to have sharp spikes. Consequently the pseudo-likelihood used to choose h_{ML} keeps growing as h is decreased (the negative log-likelihood values are shown in the figure). This problem could be solved either by using a more restrictive shape constraint or by changing the way $\Psi(x)$ is constructed—putting an upper bound on k, or a lower bound on σ, or using the weighted KDE arrangement, for example.

The bell-shaped constraints are an example of more restrictive criteria that should produce smoother estimates. Figure 5.4 shows the bell-shaped estimates of type 1, 2, and 3, each with pilot bandwidth h_{ML}. For each estimate, the optimal bandwidth was determined by line search over a the range $[0.2h_{OS}, h_{OS}]$, where h_{OS} is the over-smoothed bandwidth (equation 2.11). At each candidate bandwidth the best choices of inflection points for each estimate were found using Algorithm 5.1. Interestingly, the three estimates are nearly identical, despite the differences in their pilot band-

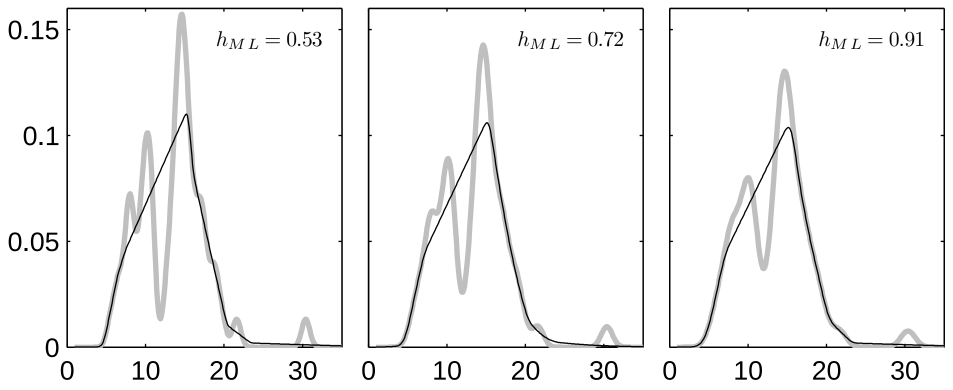

Figure 5.4: Bell shaped estimates for the wind speed data, with h_{ML} bandwidths. Estimates are type 1, 2, and 3 bell shaped, from left to right. Each plot shows the pilot estimate (grey) and the adjusted estimate (black).

widths. The main discernible changes among them are slight differences in the shape of the right tail.

5.2.2 Heart Disease Data

The estimator $\hat{f}^A_{\boldsymbol{a}}$ is in principle easily extended to higher dimensions. If a d-dimensional estimate is required, one only needs to define the k adjustment densities ψ_i as d-variate functions. The constrained estimator is still linear in $\boldsymbol{a}$, and $\boldsymbol{a}$ is still a $k \times 1$ vector. Practical implementation of the method in d dimensions involves two significant complications, however.

The first difficulty is the potential explosion in the number of adjustment densities and constraint-checking points required as d increases. In the univariate case, it was recommended to create a grid of k adjustment densities with a second grid of $G = 2k$ points used to evaluate the constraints. The size of the system of inequalities in the QP problem will quickly become unmanageable if this strategy is expanded to placing the ψ_i and $\mathbf{g}_i$ on d-dimensional rectangular meshes. The number of adjustment densities can be reduced to n (with a trade-off in estimator flexibility) by reverting to the weighted-KDE arrangement, but there is little that can be done about the number of constraint-checking points, unless moderate constraint violations can be accepted.

The size of the system of inequalities is a problem of computational capacity, but a more fundamental problem is whether higher-dimensional shape constraints can be expressed as linear inequalities in $\boldsymbol{a}$. Simple univariate constraints like unimodality or bell shape do not necessarily translate easily to higher dimensions, and more complex restrictions like unimodal conditional distributions are difficult to express mathematically without assuming that many important points are pre-specified.

Despite these difficulties, some progress can be made. The heart disease data is bivariate, and for $d = 2$ it is still possible to put the adjustment densities and constraint-checking points on a mesh without exceeding the capacities of a typical personal computer. Also, one multivariate constraint that can be implemented using QP is star unimodality (Constraint 13). This constraint specifies that the density is decreasing along all rays emanating from the mode location $\mathbf{m}$. When $\mathbf{m}$ is taken as known, the directional derivative of $\hat{f}^{A}_{\boldsymbol{a}}(x)$ along the ray from $\mathbf{m}$ to $\mathbf{g}_i$ can be expressed as a function that is linear in $\boldsymbol{a}$ (see Appendix B). The constraint can be implemented by establishing a set of constraint-enforcement points $\{\mathbf{g}_i\}$, and requiring the directional derivative to be negative at all elements of the set.

Figure 5.5 shows the star unimodal estimator. The adjustment surface was constructed using a 20×20 grid of independent bivariate normal distributions, with component standard deviations equal to the grid spacing. The constraint was enforced at a 35×35 grid of points. As with previous estimates on this normalized data set, the kernel function for the pilot was an uncorrelated bivariate normal density with covariance matrix $h^2\mathbf{I}$. The bandwidth was set to $h = 0.23$, which maximized the pseudo-likelihood criterion of section 2.4.2. Applying the constraint does improve the qualitative smoothness of the estimate, though the constrained estimate becomes star-shaped, as the name of the constraint implies. The adjusted estimate has one visible violation of the constraint (noted by an arrow in the figure). Increasing the density of the grid would eliminate such artifacts, at the cost of longer run time. The estimate in Figure 5.5 was obtained in approximately 30 seconds on a laptop computer.

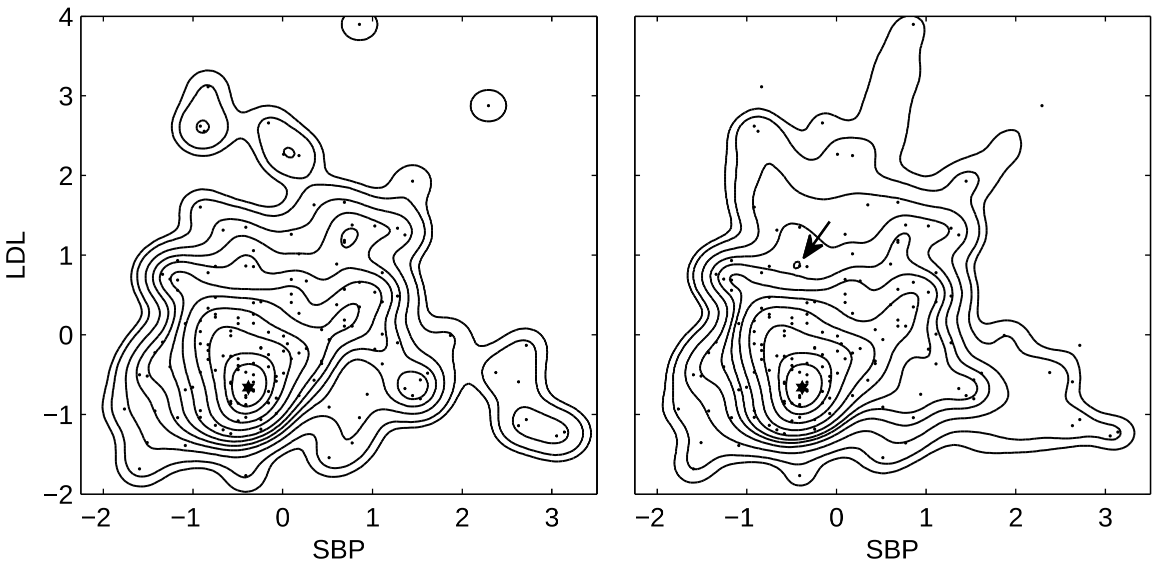

Figure 5.5: Pilot density estimate (left) and star unimodal estimate (right) for the heart disease data. The bandwidth used was $h = h_{ML} = 0.23$, and the highest mode of the pilot density (labelled by a star) was used as the mode for the adjusted estimate.

5.3 Simulation Studies

A simulation study was performed to observe how the addition of different shape constraints influence the quality of estimation afforded by the univariate Gaussian KDE. Data sets for the simulation were drawn from the t distribution with 3 degrees of freedom, with $n = 50$. For each of 260 draws, the $\hat{f}^A_{\boldsymbol{a}}$ estimator was calculated using the following five constraints:

1) no constraint,
2) unimodal,
3) unimodal and symmetric,
4) type 1 bell shape,
5) type 1 bell shape and symmetric around zero.

Each constraint was enforced using 10 different pilot bandwidths, evenly spaced between 0.2 and 0.8. In total, 13000 estimates were calculated (all combinations of 260 data sets, five constraints, and 10 bandwidths).

Note that unimodality, bell shape, symmetry, and symmetry around zero are all true characteristics of the t densities, so each of the constraints introduces valid auxiliary information that should enhance estimation performance. The main goal of the study was to observe whether the different constraints, which include different amounts of auxiliary information, produce appreciable differences in mean estimation quality and bandwidth sensitivity. The method of adjustment curves is well suited to this goal because it can handle a range of possible constraints (unlike the greedy algorithm) and still executes quickly (unlike the CEPSO algorithm).

The results of the study are summarized in Figure 5.6, which shows the mean values of the TV and ISE distances between the estimates and the truth, as a function of h, for each constraint. The horizontal dashed line on each plot shows the mean value of the appropriate distance when each unconstrained KDE was computed with an *oracle*[2] bandwidth selector—the bandwidth that actually minimizes the distance between the estimate and the truth. Performance with the oracle bandwidth represents the best possible performance of an unconstrained KDE, and provides a useful reference point.

The results suggest that adding constraints does improve performance and reduce bandwidth sensitivity. The constraints involving more qualitative information yield greater improvements. The symmetric and bell shaped estimator performed particularly well, likely because the correct point of symmetry (zero) was supplied to this estimator. It should also be noted that the optimal bandwidth is largest for the unconstrained estimate, and becomes smaller as better constrained estimators are used.

One way to understand these observations is in terms of the partitioning of pointwise mean squared error (MSE) into bias and variance terms. Using a smaller bandwidth reduces bias, but increases the variance of the pilot estimate. Enforcing the constraint should allow the adjusted estimator to damp out much of this increased variance, with a resulting improvement in MSE. This line of reasoning corroborates the idea that $MISE$-optimal bandwidths may be smaller when constraints are imposed than in the unconstrained case.

[2]A term used in model selection studies for selectors that operate with knowledge of the true model. See, e.g., Fan and Li (2001).

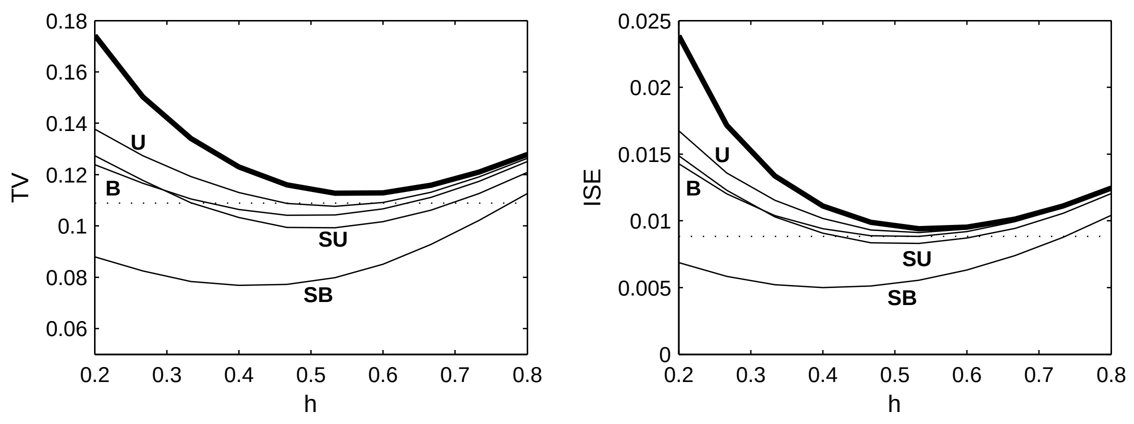

Figure 5.6: Statistical performance of constrained estimates using $\hat{f}^A_{\boldsymbol{a}}$. The thick line is the result for the pilot estimator. Labels on the other four lines indicate the operative constraints: U for unimodality, B for bell shape, and S for symmetry. The dotted horizontal lines give the performance of the unconstrained estimator with oracle bandwidth. Typical standard errors for the points in the plot are 0.002 for TV, and 0.0004 for ISE.

The simulation also provides information on typical run times required to obtain constrained estimates. Figure 5.7 plots the median run time as a function of h and the constraint type. The run times in the plot reflect the combined effects of two factors: the size of the system of inequalities necessary to enforce the constraints, and the repetitions required to find the best inflection or mode points. The system of inequalities becomes larger as h gets smaller (a consequence of the default construction of the adjustment curve), and also becomes larger when the symmetry constraint is added. The bell-shaped constraint requires two fixed points to be selected, while unimodality only requires one. Figure 5.6 suggests optimal bandwidths fall in the range $(0.4, 0.6)$. In this range, estimates can typically be obtained in 30 seconds or less even for the symmetric and bell-shaped estimator.

5.4 Limitations and Extensions

The method of adjustment curves has some attractive features, foremost of which is the ability to use fast and reliable quadratic programming routines to obtain certain types of constrained estimates. In addition, the shape adjustment may be designed by

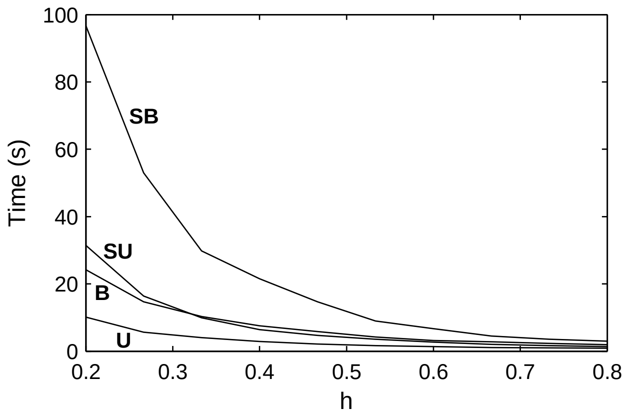

Figure 5.7: Median run times for the adjustment curve estimates. Line labels are the same as in Figure 5.6. The unconstrained estimate is not shown because its run times are too close to zero (typically 0.007s).

the user, and is not coupled to the form of the estimator, as it is for $\hat{f}^M$, $\hat{f}^W$, and $\hat{f}^B$. This offers potentially greater flexibility in determining the constrained estimator's characteristics, and opens up a number of avenues for refinement or expansion of the method. Several such ideas are summarized here.

- The method of constructing the adjustment curve is open to alteration. One obvious change is to use adjustment densities that are compactly supported. Alternatively, it may be possible to define a_i to be the height of the adjustment curve at point μ_i, and to define $\psi(x)$ as a curve that interpolates these points. Such changes might simplify the construction of $\Psi(x)$ or improve numerical performance.

- Two options for placement of the adjustment densities were proposed in this chapter: putting them at the data locations, or putting them on a grid. An adaptive or hybrid method of locating the adjustment densities could be proposed, that combines the advantages of both options by putting more densities in data-rich regions, and a regular grid of densities in data-poor regions.

- In addition to strict shape constraints, penalty terms can be added to the objective function of (5.7) to further control the shape of $\hat{f}^A(x)$. Similar to penalties in functional data analysis, these terms could be used to penalize roughness or to encourage the estimate to move toward a certain parametric form. Importantly, the objective function is still a quadratic form with such penalties, so QP can still be used to find the estimates. This is a strength of the proposed method, since similar function estimation approaches involve more complicated optimizations not so routinely solved (e.g. Ramsay and Silverman, 2005, sec. 6.6). Taking the unimodal estimates of Figure 5.3 as an example, a roughness penalty could be used to resolve the problem of spikes in the constrained estimates (which would also allow h_{ML} to be used). Appendix B gives an example of how a roughness penalty can be set up.

- Many shape constraints are actually restrictions on the derivatives of the estimate. It may be possible to apply the adjustment curve to the appropriate derivative of the KDE rather than to the KDE itself. This approach could be expected to give better numerical stability and smoother density estimates, but more sophisticated optimization might be required.

- Quadratically constrained quadratic programming (QCQP) is a different convex programming method that allows the constraints, not just the objective function, to be quadratic forms. If QCQP solvers are available, additional constraints could be handled without resorting to heuristic methods. In particular, the nearly parametric constraint (with distance measured by integrated squared error) could be solved.

- Exploring better numerical methods could lead to improvements in the ability to handle higher-dimensional problems. The optimization problem has a sparseness property, since kernel functions and adjustment densities distant from any x will not influence the estimate at x. Methods that work locally near a given x might be able to expand the range of problems that are practicable.

- It should be possible to adapt the adjustment curve approach to constrained nonparametric regression problems. Because the adjustment curve does not directly depend on the data, it may be easier to find an optimal adjustment than to constrain the regression estimator directly.

This chapter has focused solely on the use of adjustment curves with problems for which QP can be used to find solutions. It is important to note that $\hat{f}_{\boldsymbol{a}}^{A}$ is not limited to these cases, however. As long as a sufficiently effective optimizer is available, other constraints could be satisfied by this adjustment method. To that end, a potentially fruitful option is to use the CEPSO optimizer of Chapter 4 to find good values of $\boldsymbol{a}$ for constraints that do not meet the QP requirements.

Chapter 6

Conclusions and Further Work

A number of new contributions were introduced in the preceding chapters. Heuristic optimization algorithms `improve`, `ILSimprove`, and `CEPSO` were proposed for solving difficult shape-constrained estimation problems. These heuristics were successfully demonstrated on data sharpening problems, though they have the potential to work with other forms of shape adjustment as well. A new method of shape adjustment, the additive adjustment curve, was also developed. This form of adjustment has considerable appeal because globally optimal solutions can be found for many constraints when it is used. Other new ideas appeared alongside the optimization methods. Several new constraints were proposed, a new distance function (RC_γ) was used to improve the numerical performance of SQP, and a new quantity h_{ML} was introduced as a workable bandwidth selector that accounts for shape restrictions.

The limitations and possible extensions of the new methods have already been discussed in Chapters 3, 4, and 5. Rather than reiterating them here, we will consider the results of this work as a whole. The possibilities for application of the tools in their present form are discussed first, followed by thoughts on the most important areas for future work.

6.1 Why, When, and How to Use the Methods

The advantages of shape-restricted nonparametric estimation have been demonstrated throughout the thesis. When constraints are expressed as black box functions, modelling assumptions can be tailored to more closely reflect the particular circumstances of an analysis. Constraints can be used when there are theoretically-motivated reasons to do so, but even in the absence of a theoretical basis, constrained estimation can augment or improve the way a nonparametric estimator achieves its smoothness. An appropriately-chosen constraint can influence the amount of smoothing that takes place in different parts of the estimate, or reduce an estimator's sensitivity to the values of its smoothing parameters.

In their current state of development, the optimization algorithms of chapters 3, 4, and 5 allow pointwise estimation of shape-constrained densities in one or two dimensions (though higher-dimensional estimation may be feasible in some cases). Interval estimation has not been considered. Because of this the methods are most likely to be useful in exploratory data analysis and data visualization, where low-dimensional density plotting and qualitative interpretation of results are important activities. Fortunately this is a significant application of density estimation in practice. Constraints are also particularly useful when the sample size is small. In small-n problems, the effect of sampling variation can be large enough to drastically influence the qualitative characteristics of an estimate. Shape constraints provide a way to use auxiliary information to reduce this effect.

The three main optimizers developed in previous chapters are `improve`, CEPSO, and the adjustment-curve method using quadratic programming. A formal comparison of these optimizers' performance has not been conducted. The main reason for this is the lack of overlap in the typical use for each approach. Given a data set and a constraint to impose on a density estimate, the best optimization method can be chosen in the following manner:

1. If the adjustment-curve estimator $\hat{f}^A_{\boldsymbol{a}}(x)$ is acceptable and the constraint is among those suitable for quadratic programming, use the methods of Chapter 5.

In this combination of circumstances the calculation will be fast and the global optimum solution will be found[1], so there is no reason to consider alternative methods.

2. If the data sharpening estimator $\hat{f}^{M}_{\mathbf{y}}(x)$ is preferred, the objective function is the L_α type, and the constraint is univariate unimodality, either `improve` or CEPSO can be used. Though both methods will run quickly, the greedy algorithm will usually be faster, so it would be preferred when run time is a priority (e.g., in a simulation study).

3. In all other situations, use CEPSO.

The `ILSimprove` function has not been included in this list. It would be an alternative for any problem otherwise solved by CEPSO, but insufficient testing has been performed to determine whether its performance is actually competitive with CEPSO. Also, if the $\hat{f}^{W}_{\mathbf{w}}(x)$ or $\hat{f}^{A}_{\boldsymbol{a}}(x)$ estimators are to be used with CEPSO, additional modifications would be required to handle the sum constraints on the adjustable parameters (as discussed in Section 4.6).

6.2 Areas for Further Improvement

The most important open tasks suggested by this work can be grouped into three areas: general improvements to the methodology, improvements specific to density estimation, and extension of the methods to regression problems.

Methodological improvements

The ultimate goal toward which this thesis is working is a general-purpose constrained estimation optimizer that can find good solutions for problems with any combination of estimator, constraint, adjustment method, and objective function. The `improve` algorithm is not sufficiently adaptable to meet this goal, owing to its greedy design and its implicit use of the L_α objective function. The quadratic programming methods

[1]As described in Section 5.1.2, global optimality is contingent on finding the optimal locations of certain important points (mode, inflection points, etc.) required as inputs to the QP solver.

of Chapter 5 are inherently limited to problems using adjustment curves (or variable weights, in the cases where these two adjustment methods are equivalent). So it is the CEPSO algorithm that holds the most promise for becoming a truly general optimizer for constrained estimation.

Even in its present form, CEPSO can be applied in a wider range of situations than those considered in Chapter 4. This includes problems involving variable KDEs, weighted KDEs, and regression estimators (see Wolters, 2011, for examples). Still, the extensions proposed at the end of Chapter 4 are important to improve the algorithm's performance and expand its applicability. Finalizing a means of handling sum-constrained adjustable values, for example, will make it a routine matter to use different adjustment methods (including all four adjustments proposed for KDEs: $\hat{f}^M$, $\hat{f}^W$, $\hat{f}^B$, and $\hat{f}^A$). Other performance enhancements such as repair functions, penalty functions, or local improvement steps will likely be required to achieve reliable performance over the widest range of problems.

Two outstanding concerns of a statistical nature form an avenue for improvement completely separate from optimization questions. The first concern is uncertainty assessment (interval estimation) for shape-constrained estimators. This is an especially important topic in regression problems, where one is more likely to have inference as a goal of the analysis. The second is establishing a means for testing the validity of a shape constraint. Having a sufficiently powerful test for the truth of an arbitrary shape constraint would help build confidence in constrained estimation. It could also open up a new range of applications for constrained estimation, since the validity of the constraint is often a question of considerable interest in its own right (as when testing for unimodality of a density, for example). There are prior results in this area upon which to build. Du et al. (2010) and Cule et al. (2010), for example, suggest resampling-based methods for the constraints that they treat in their work.

Density estimation problems

The variable-bandwidth estimator $\hat{f}^B$ was not examined in the preceding chapters, and it does not appear to have received attention in the constrained estimation literature. Nevertheless, it offers some enticing advantages that give it considerable potential in the continued development of shape-constrained density estimation. Foremost among these is the possibility of eliminating bandwidth selection from the estimation process. Another advantage is the ability to adapt the density's smoothness locally, which could be especially beneficial for eliminating spurious modes in a density's tails. It is not clear whether these advantages would correspond to improved estimation in the $MISE$ sense (Farmen and Marron, 1999, studied variable bandwidth KDEs in the unconstrained case and found little improvement over h_{SJ}), but the possibility is worth further study.

Regression problems

The methods of this thesis could open new opportunities in shape constrained regression problems. The number of inflection points in a nonparametric regression estimate, for example, could be used as a type of smoothing parameter. An estimate with a controlled number of inflections would have a restricted number of peaks and valleys, but at the local scale any of its peaks or valleys could have high curvature if the data demanded it. Or a model of the probability of success in a binary-response problem could be constrained to have exactly one inflection point, as a more flexible alternative to parametric options like logistic regression.

While there is no immediate barrier to applying CEPSO to regression problems, more work must be done to explore its capabilities and to identify problem-specific difficulties. The problem of finding feasible initial solutions, for example, is more challenging in regression than in the density estimation case. In regression, the strategy of reproducing the kernel function by putting all points at the same location is not available. For some constraints it could be difficult to find even a single feasible starting point. A reliable means of finding feasible solutions would therefore be of great benefit.

Another important question for future work is whether and how the adjustment curve approach can be extended to the regression case. Changes to the construction of the adjustment curve would probably be sensible, since a linear combination of density functions might not be the best choice outside of the density estimation context. Different constraints would have to be considered in conjunction with the adjustment curve, to see whether any interesting problems could be solved by quadratic programming. The only major differences between regression and density estimation are the non-negativity and unit-integral restrictions, however, and these can easily be removed from the adjustment curve. So there is reason to believe that the method could be fruitfully implemented.

Appendix A

Bandwidth Selection Simulation Study

A simulation study was performed to investigate the performance of different bandwidth selectors in shape-constrained density estimation. The same data sets generated for the simulation of Section 3.3 were used: 250 replications for each combination of two densities (t_3 or mixture, as shown in Figure 3.7) and three sample sizes (25, 50, or 100). Unimodality was taken as the operative constraint, and data sharpening was used as the method of shape adjustment. The greedy algorithm (Algorithm 3.1) was used to find the unimodal estimates.

Three bandwidth selectors were considered:

The proposed pseudo-likelihood bandwidth, h_{ML} (equation 2.16).

The Sheather-Jones bandwidth h_{SJ} (equation 2.12).

The likelihood cross-validation bandwidth h_{LCVa} (equation 2.15).

Bandwidth h_{SJ} was calculated from the unsharpened data prior to sharpening. This bandwidth, well established in the unconstrained case, can serve as a reference value for the other two bandwidths. For h_{ML}, the bandwidth was selected by line search over the possible h values, with sharpening performed at each candidate h. Line search was also used for h_{LCVa}, with cross-validation carried out separately at each candidate h. Both line searches were carried out over the interval $[0.01, 1.5]h_{OS}$, using golden section search (see, e.g. Lange, 2010, p. 67).

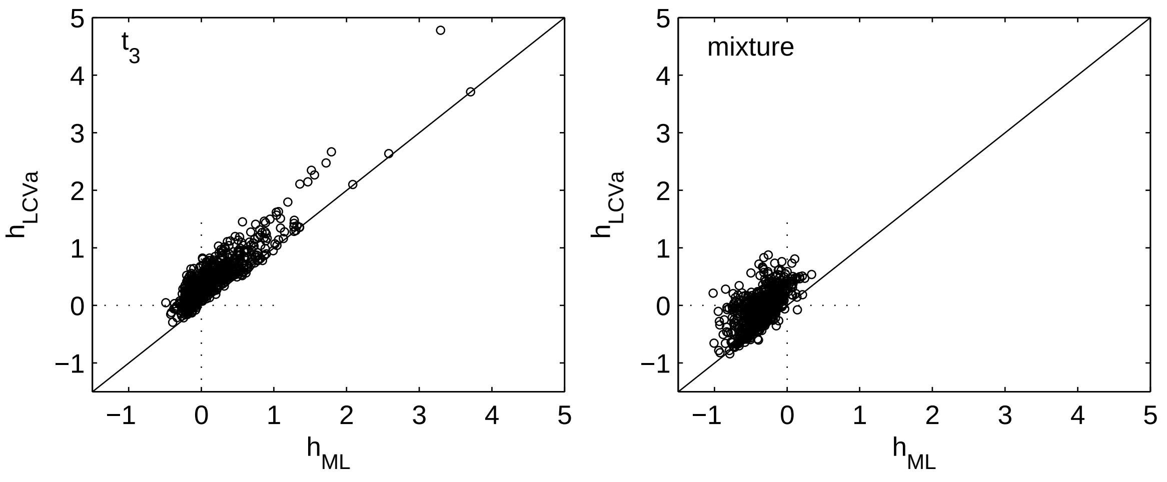

Figure A.1: Scatter plot of h_{LCVa} versus h_{ML} for the simulation runs. The value of h_{SJ} has been subtracted from each bandwidth. The left and right plots show the t_3 and mixture distribution cases, respectively. The 1:1 lines are also shown.

A summary of the results is shown in Figure A.1. The values of $(h_{LCVa} - h_{SJ})$ and $(h_{ML} - h_{SJ})$ are plotted against one another in two scatter plots, one for each true density. The relative magnitudes of h_{LCVa} and h_{ML} may be compared by observing which side of the 1:1 line the points fall on. Points above the line have $h_{LCVa} > h_{ML}$, and points below it have $h_{ML} > h_{LCVa}$. To compare either of the other bandwidths to h_{SJ}, the marginal distribution of points can be observed. Points close to zero represent cases where the selected bandwidth is close to h_{SJ}.

Applying these rules of interpretation to the two plots shows that likelihood cross-validation typically smoothed the density estimates more than the pseudo-likelihood method: h_{LCVa} was greater than or equal to h_{ML} in almost all cases, for both true densities. For samples from the heavier-tailed t_3 density, both h_{LCVa} and h_{ML} usually selected bandwidths larger than h_{SJ}. For the mixture cases, h_{LCVa} selected a bandwidth smaller than h_{SJ} about half of the time, while h_{ML} was less than h_{SJ} in most of the samples.

Figure A.2 shows the unimodal density estimates constructed using the three competing bandwidths, for nine randomly-selected simulation runs. The plots illustrate that h_{LCVa} is most sensitive to outliers, while h_{ML} has this property to a lesser extent.

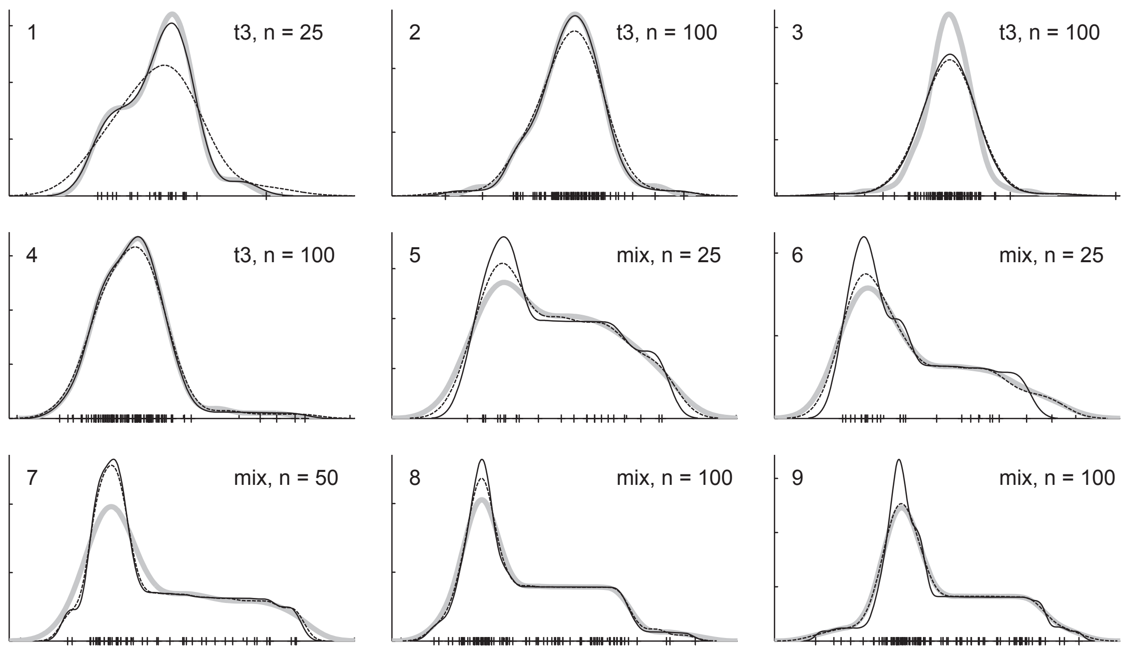

Figure A.2: Unimodal density estimates using h_{ML} (solid line), h_{LCVa} (dashed line), and h_{SJ} (thick grey line), for nine randomly selected simulation runs. The labels on each plot indicate the true density and sample size. The unsharpened data are plotted on the horizontal axis.

When there are not extreme outliers, h_{ML} tends to select smaller bandwidths that create sharper peaks over regions of high density.

The overall performance of each bandwidth can be evaluated by integrated measures of estimation accuracy. Table A.1 gives simulation averages of two such measures, the total variation (equation 2.8) and the integrated squared error (equation 2.6), for the six combinations of density and sample size. The conclusions are the same for both TV and ISE. The pseudo-likelihood and cross-validation selectors each performed worse than h_{SJ} on the t_3 cases, with h_{LCVa} showing particularly poor results. For the mixture cases, h_{ML} was the best bandwidth, with h_{SJ} and h_{LCVa} approximately equal. The relative standing of the selectors was unaffected by sample size, though the estimation accuracy naturally improved as n increased.

The absolute performance of each bandwidth can be evaluated using the performance of the oracle bandwidth (the best bandwidth possible, given knowledge of the true density, as used in Section 5.3) as a benchmark. Table A.1 also includes average

Table A.1: Sample mean of TV and ISE distances from the truth.

	TV					$10 \times ISE$				
Case	h_{SJ}	h_{ML}	h_{LCVa}	h_U^{Or}	h_C^{Or}	h_{SJ}	h_{ML}	h_{LCVa}	h_U^{Or}	h_C^{Or}
t_3, $n = 25$	.149	.173	.209	.131	.119	.165	.231	.319	.128	.116
t_3, $n = 50$	.114	.140	.165	.102	.095	.100	.158	.214	.078	.073
t_3, $n = 100$	.091	.120	.140	.084	.079	.063	.123	.160	.054	.051
mixture, $n = 25$	.173	.156	.175	.168	.143	.182	.143	.179	.159	.124
mixture, $n = 50$	.140	.124	.139	.136	.114	.125	.093	.123	.104	.080
mixture, $n = 100$	.111	.097	.111	.108	.090	.080	.055	.081	.063	.052

Note: the standard error of the estimate is less than 0.0057 for all TV entries and less than 0.0152 for all $10 \times ISE$ entries.

TV and ISE values for two oracle estimators. The column labelled h_U^{Or} gives results for the unconstrained KDE with the oracle bandwidth selector—the best possible pilot estimator. The column labelled h_C^{Or} shows results for the constrained estimate with oracle bandwidth. The values in this column are the best results achievable with any bandwidth selector when the `improve` algorithm is used to perform data sharpening. Oracle bandwidth values were found using golden section search in the same way as the other bandwidths.

The performance measures for h_U^{Or} can be thought of as reasonable targets for the performance of a constrained estimator. If the constraint confers an accuracy advantage, it is reasonable to hope to estimate the truth as well as the best possible unconstrained estimate. At the same time, the h_C^{Or} results define a bound on the amount of improvement that can be achieved. For the t_3 problems, for example, h_U^{Or} and h_C^{Or} give nearly equal performance, especially for the two largest sample sizes. In this case we can not expect constrained estimation to have much effect on these summary measures, regardless of bandwidth selector. It is perhaps not surprising, then, that none of the three competing bandwidth selectors was able to outperform h_U^{Or}. Other constraints or adjustment methods could perhaps offer greater improvements.

For the mixture problems, the difference between unconstrained and constrained oracle estimates was larger, suggesting that there is greater potential for gains in estimation performance. Still, neither h_{SJ} nor h_{LCVa} was able to outperform the unconstrained oracle estimator. Only h_{ML} showed some improvement. Estimates based on h_{ML} had average TV and ISE performance approximately midway between that of h_U^{Or} and h_C^{Or} for all sample sizes.

Figure A.3 provides another view of statistical performance by showing the pointwise bias-variance decomposition of the density estimates obtained using each bandwidth. The figure contains one sub-plot for each simulation case. Each plot shows, as a function of x, three envelopes, one for each bandwidth option. Each envelope consists of two lines. The upper line is the mean squared error, and the lower line is the squared bias. The gap between the two lines shows the magnitude of the variance. The information in this figure corroborates the results already discussed. In the t_3 samples, outlier-induced oversmoothing has led h_{ML} and h_{LCVa} to poorly estimate both the peak (underestimation) and the tails (overestimation) of the density, relative to h_{SJ}. Likelihood cross-validation shows a particularly high bias at the peak. The pseudo-likelihood bandwidth does a much better job of estimating the peak in the mixture cases than either of the other bandwidths.

The results presented here underscore the difficulty in making a general recommendation for bandwidth selection. It appears, on one hand, that reasonable performance can be achieved when using an established bandwidth selector like h_{SJ} for shape-adjusted estimation. On the other hand, it is also clear that there is potential for performance improvement using a bandwidth selector that accounts for the shape adjustment. The simulation suggests that the pseudo-likelihood bandwidth h_{ML} is a promising choice, particularly when the density to be estimated is not heavy-tailed.

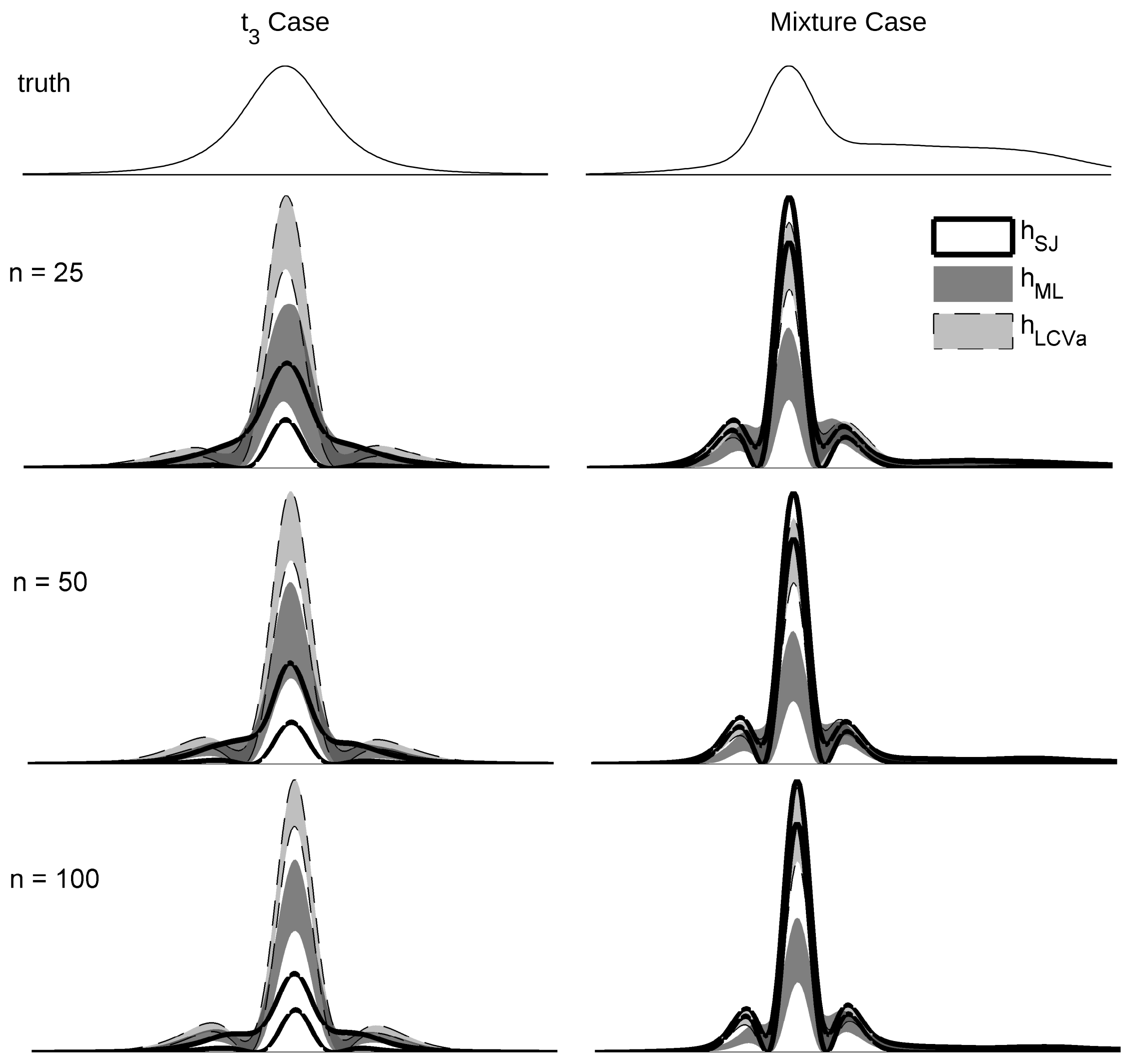

Figure A.3: Pointwise decomposition of MSE for the six simulation cases. The two top plots show the densities being estimated. The rest of the plots show one envelope for each bandwidth choice. The lower bound of each envelope is the squared bias, and the upper bound is the MSE. Each plot has been scaled independently to fill its axes.

Appendix B

Example Quadratic Programs

The general form of a quadratic program is as follows. Minimize the quadratic objective function

$$\boldsymbol{a}^T \mathbf{H} \boldsymbol{a} + \mathbf{v}^T \boldsymbol{a}, \tag{B.1}$$

subject to linear equality and inequality constraints

$$\begin{aligned} \mathbf{A}\boldsymbol{a} &\leq \mathbf{b} & \text{(B.2)} \\ \mathbf{A}_{eq}\boldsymbol{a} &= \mathbf{b}_{eq}. & \text{(B.3)} \end{aligned}$$

Here $\boldsymbol{a}$ is the k-vector of adjustment coefficients in the estimator $\hat{f}^A_{\boldsymbol{a}}(x) = \hat{f}^{\circ}(x) + \boldsymbol{a}^T \boldsymbol{\psi}(x)$. Recall that $\boldsymbol{\psi}(x) = [\psi_1(x) \ \cdots \ \psi_k(x)]^T$ is the vector collecting the values of all adjustment densities at x. The other quantities $\mathbf{H}$, $\mathbf{v}$, $\mathbf{A}$, $\mathbf{b}$, $\mathbf{A}_{eq}$, and $\mathbf{b}_{eq}$ are appropriately-sized matrices and vectors of constants that depend on the pilot estimator, the chosen constraints and the way $\Psi(x)$ is defined. This appendix demonstrates how to determine these quantities for three instances:

1. A problem with symmetry and unimodality constraints.

2. The case where a penalty is added to the objective function to control the roughness of the final estimate.

3. The bivariate star unimodality case.

The other constraints listed in Chapter 5 can be set up in a manner similar to these.

A Symmetric and Unimodal Estimator

We will consider how to construct the objective function, the equality constraints, and the inequality constraints. The system of inequalities (B.2) must include three shape restrictions: non-negativity, unimodality, and symmetry. So the matrix $\mathbf{A}$ and vector $\mathbf{b}$ are each partitioned into three parts,

$$\mathbf{A} = \begin{bmatrix} \mathbf{A_1} \\ \mathbf{A_2} \\ \mathbf{A_3} \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} \mathbf{b_1} \\ \mathbf{b_2} \\ \mathbf{b_3} \end{bmatrix}, \tag{B.4}$$

with each submatrix/subvector handling one constraint.

The Objective Function

Two possible objectives were mentioned in Chapter 5: the L_2 objective $\boldsymbol{a}^T\boldsymbol{a}$, and the ISE objective $\int_{-\infty}^{\infty} \boldsymbol{a}^T\boldsymbol{\psi}(x)\boldsymbol{\psi}(x)^T\boldsymbol{a}\ dx$. The L_2 objective leads to a simple form of (B.1):

L_2 Objective

Minimize $\boldsymbol{a}^T\mathbf{H}\boldsymbol{a} + \mathbf{v}^T\boldsymbol{a}$,

where $\mathbf{H} = \mathbf{I_k}$, $\mathbf{v} = \mathbf{0}$, and $\mathbf{I_k}$ is the $k \times k$ identity matrix.

The ISE objective can be approximated using the trapezoidal rule with the function evaluated at the constraint-checking points $\mathbf{g} = [g_1 \ldots g_G]^T$, yielding

$$ISE(\boldsymbol{a}) \approx \frac{g_G - g_1}{2(G-1)} \left[\boldsymbol{a}^T \left(\mathbf{D}_1 + \mathbf{D}_G + 2\sum_{l=2}^{G-1} \mathbf{D}_l \right) \boldsymbol{a} \right] \propto \boldsymbol{a}^T\mathbf{D}\boldsymbol{a}, \tag{B.5}$$

where $\mathbf{D}_l = \boldsymbol{\psi}(g_l)\boldsymbol{\psi}(g_l)^T$ is a $k \times k$ matrix. Thus the ISE objective may be expressed as follows.

ISE Objective

Minimize $\boldsymbol{a}^T\mathbf{H}\boldsymbol{a} + \mathbf{v}^T\boldsymbol{a}$,

where $\mathbf{H} = \mathbf{D}$ as defined in (B.5), and $\mathbf{v} = \mathbf{0}$.

The form of the ISE objective allows us to see why the two objectives give such similar results when the $\{\psi_i\}$ and $\mathbf{g}$ are chosen by the default method described in Section 5.1.3. With the ISE objective, the matrix $\mathbf{H}$ is a sum of G matrices of the form $\mathbf{D}_l = \boldsymbol{\psi}(g_l)\boldsymbol{\psi}(g_l)^T$, and the (i, j)th element of $\mathbf{D}_l$ is $\psi_i(g_l)\psi_j(g_l)$. This product will only be non-negligible if ψ_i and ψ_j are near each other; consequently $\mathbf{D}_l$ (and $\mathbf{H}$) will take large values only on the main diagonal and the first few sub- and super-diagonals, regardless of k. When k is sufficiently large, $\mathbf{H}$ behaves for the purposes of optimization much like an identity matrix, and the two objectives are nearly equivalent.

Constraints to Ensure the Estimate is a Density

The constraint that the density estimate integrate to one leads to a restriction that the a_i must sum to zero.

Unit area restriction

Require $\mathbf{A}_{eq}\boldsymbol{a} = \mathbf{b}_{eq}$, where $\mathbf{A}_{eq} = \mathbf{1}_k^T$, $\mathbf{b}_{eq} = \mathbf{0}$, and $\mathbf{1}_k$ is a k-vector of ones.

The non-negativity constraint is enforced at the points in $\mathbf{g}$ and results in a system of G inequalities. At the point g_l, the inequality is $\hat{f}^\circ(g_l) + \boldsymbol{\psi}(g_l)^T\boldsymbol{a} \geq 0$, and this leads to the following system.

Non-negativity constraint

Require $\mathbf{A_1}\boldsymbol{a} \leq \mathbf{b_1}$, where

$$\underset{G\times k}{\mathbf{A_1}} = -\begin{bmatrix} \boldsymbol{\psi}(g_1)^T \\ \vdots \\ \boldsymbol{\psi}(g_G)^T \end{bmatrix} \qquad \text{and} \qquad \mathbf{b_1} = \begin{bmatrix} \hat{f}^{\circ}(g_1) \\ \vdots \\ \hat{f}^{\circ}(g_G) \end{bmatrix}.$$

Shape Constraints

The first shape constraint is unimodality with mode m (which is taken as known and fixed). Considering the constraint-checking points, we require the first derivative to satisfy

$$\hat{f}^{\circ\prime}(g_l) + \boldsymbol{\psi}'(g_l)^T\boldsymbol{a} \begin{cases} \geq 0, & g_l \leq m \\ \leq 0, & g_l \geq m \end{cases}, \tag{B.6}$$

or, equivalently,

$$\begin{aligned} -\boldsymbol{\psi}'(g_l)^T\boldsymbol{a} &\leq \hat{f}^{\circ\prime}(g_l) && \text{when } g_l \leq m \\ \boldsymbol{\psi}'(g_l)^T\boldsymbol{a} &\leq -\hat{f}^{\circ\prime}(g_l) && \text{when } g_l \geq m. \end{aligned}$$

The two inequalities above differ only in their signs. The signum function can be used to write the system of constraints in a unified way.

Unimodality constraint

Require $\mathbf{A_2}\boldsymbol{a} \leq \mathbf{b_2}$, where

$$\underset{G\times k}{\mathbf{A_2}} = -\begin{bmatrix} \operatorname{sgn}(g_1 - m)\boldsymbol{\psi}'(g_1)^T \\ \vdots \\ \operatorname{sgn}(g_G - m)\boldsymbol{\psi}'(g_G)^T \end{bmatrix} \qquad \text{and} \qquad \mathbf{b_2} = \begin{bmatrix} \operatorname{sgn}(m - g_1)\hat{f}^{\circ\prime}(g_1) \\ \vdots \\ \operatorname{sgn}(m - g_G)\hat{f}^{\circ\prime}(g_G) \end{bmatrix}.$$

Moving on to the symmetry constraint, note first that if the estimate is unimodal with mode m, then m must be its point of symmetry as well. For simplicity, let $m = (g_1 + g_G)/2$, so the estimate is to be symmetric around the midpoint of the constraint-checking grid. The constraint is to be enforced at r pairs of points equidistant from m. If G is odd, $m = g_{(G-1)/2}$ and we will have $r = (G-1)/2$; if G is even, $r = G/2$. Strict symmetry constraints are equalities, however there may be numerical difficulties enforcing them as such. For example, if the grid of ψ_i densities is not aligned to $\mathbf{g}$, or if k is too small, it may not be possible to get exact reflection around m. So it is more effective to enforce near-symmetry through inequality constraints with a tolerance, ϵ. For $l = 1, \ldots, r$, the constraint is $|\hat{f}^A_{\boldsymbol{a}}(g_l) - \hat{f}^A_{\boldsymbol{a}}(g_{G-l+1})| \leq \epsilon$, or

$$\begin{aligned} \hat{f}^A_{\boldsymbol{a}}(g_l) - \hat{f}^A_{\boldsymbol{a}}(g_{G-l+1}) &\geq -\epsilon \\ \hat{f}^A_{\boldsymbol{a}}(g_l) - \hat{f}^A_{\boldsymbol{a}}(g_{G-l+1}) &\leq \epsilon. \end{aligned}$$

So each of r symmetry checks produces two inequalities that must be satisfied. Writing them in terms of the adjustment densities and their coefficients $\boldsymbol{a}$ produces

$$\begin{aligned} (\boldsymbol{\psi}(g_l) - \boldsymbol{\psi}(g_{G-l+1}))^T \boldsymbol{a} &\leq \hat{f}^\circ(g_{G-l+1}) - \hat{f}^\circ(g_l) + \epsilon \\ (\boldsymbol{\psi}(g_{G-l+1}) - \boldsymbol{\psi}(g_l))^T \boldsymbol{a} &\leq \hat{f}^\circ(g_l) - \hat{f}^\circ(g_{G-l+1}) + \epsilon, \end{aligned}$$

which may be combined in matrix-vector form as shown below.

Symmetry constraint

Require $\mathbf{A_3}\boldsymbol{a} \leq \mathbf{b_3}$, where $\underset{2r\times k}{\mathbf{A_3}} = \begin{bmatrix} \mathbf{M} \\ -\mathbf{M} \end{bmatrix}$, $\mathbf{b}_3 = \begin{bmatrix} \mathbf{w} + \epsilon\mathbf{1} \\ -\mathbf{w} + \epsilon\mathbf{1} \end{bmatrix}$, and

$$\underset{r\times k}{\mathbf{M}} = \begin{bmatrix} (\boldsymbol{\psi}(g_1) - \boldsymbol{\psi}(g_G))^T \\ \vdots \\ (\boldsymbol{\psi}(g_l) - \boldsymbol{\psi}(g_{G-l+1}))^T \\ \vdots \\ (\boldsymbol{\psi}(g_r) - \boldsymbol{\psi}(g_{G-r+1}))^T \end{bmatrix} \quad \text{and} \quad \mathbf{w} = \begin{bmatrix} \hat{f}^\circ(g_G) - \hat{f}^\circ(g_1) \\ \vdots \\ \hat{f}^\circ(g_{G-l+1}) - \hat{f}^\circ(g_l) \\ \vdots \\ \hat{f}^\circ(g_{G-r+1}) - \hat{f}^\circ(g_r) \end{bmatrix}.$$

Adding a roughness penalty to the objective

Consider a slightly more general form of the objective function (B.1),

$$\boldsymbol{a}^T(\mathbf{H} + \lambda\mathbf{S})\boldsymbol{a} + \mathbf{v}^T\boldsymbol{a}, \tag{B.7}$$

where the matrix of the quadratic form has been expressed as a sum of two parts. The matrix $\mathbf{H}$ is unchanged from the preceding calculations: it measures the L_2 or ISE distance between the estimate and the pilot density. The matrix $\mathbf{S}$ measures the roughness of the estimate, in a manner to be described presently. The nonnegative scalar λ is a tuning parameter that determines the degree to which roughness is taken into account. Since the goal is to minimize the objective function, the $\lambda\mathbf{S}$ term may be viewed as a penalty that discourages less smooth solutions.

A common way of measuring the overall lack of smoothness of a function is the integral of its squared second derivative. If we consider this quantity for $\hat{f}^A_{\boldsymbol{a}}$, we find

$$\begin{aligned} \int_{-\infty}^{\infty} \left(\hat{f}^{A\prime\prime}_{\boldsymbol{a}}(x)\right)^2 dx &= \int_{-\infty}^{\infty} \left(\hat{f}^{\circ\prime\prime}(x) + \boldsymbol{\psi}''(x)^T\boldsymbol{a}\right)^2 dx \\ &= \int_{-\infty}^{\infty} \left((\hat{f}^{\circ\prime\prime}(x))^2 + 2\hat{f}^{\circ\prime\prime}(x)\boldsymbol{\psi}''(x)^T\boldsymbol{a} + \boldsymbol{a}^T\boldsymbol{\psi}''(x)\boldsymbol{\psi}''(x)^T\boldsymbol{a}\right)^2 dx, \end{aligned}$$

which may be approximated using the trapezoidal rule over the points in $\mathbf{g}$ in the same manner as (B.5). Doing so and ignoring $\boldsymbol{a}$-free terms we arrive at the quantity

$$\text{Pen}(\boldsymbol{a}) = \frac{g_G - g_1}{2(G-1)} \left[\boldsymbol{a}^T \left(2\hat{f}^{\circ\prime\prime}(g_1)\boldsymbol{\psi}''(g_1) + 2\hat{f}^{\circ\prime\prime}(g_G)\boldsymbol{\psi}''(g_G) + 4\sum_{l=2}^{G-1} \hat{f}^{\circ\prime\prime}(g_l)\boldsymbol{\psi}''(g_l) \right) \right.$$
$$\left. + \, \boldsymbol{a}^T \left(\boldsymbol{\psi}''(g_1)\boldsymbol{\psi}''(g_1)^T + \boldsymbol{\psi}''(g_G)\boldsymbol{\psi}''(g_G)^T + 2\sum_{l=2}^{G-1} \boldsymbol{\psi}''(g_l)\boldsymbol{\psi}''(g_l)^T \right) \boldsymbol{a} \right],$$

and the final penalty is found by dropping the leading proportionality constant.

Roughness penalty

In the objective function $\boldsymbol{a}^T(\mathbf{H} + \lambda\mathbf{S})\boldsymbol{a} + \mathbf{v}^T\boldsymbol{a}$, define $\mathbf{S}$ and $\mathbf{v}$ as

$$\mathbf{S} = \mathbf{S}_1 + \mathbf{S}_G + 2\sum_{l=2}^{G-1} \mathbf{S}_l \qquad \text{and} \qquad \mathbf{v} = \mathbf{v}_1 + \mathbf{v}_G + 2\sum_{l=2}^{G-1} \mathbf{v}_l,$$

where $\mathbf{S}_l = \boldsymbol{\psi}''(g_l)\boldsymbol{\psi}''(g_l)^T$ and $\mathbf{v}_l = \hat{f}^{\circ\prime\prime}(g_l)\boldsymbol{\psi}''(g_l)$.

The above penalty is not the only such quantity that could be derived. Other integrated squared functions would have a similar form. For example one could penalize only on the roughness of the adjustment rather than on the roughness of the final estimate; or penalize on the integrated squared distance from a parametric density.

The Star Unimodal Constraint

The constraint of star unimodality with mode at $\mathbf{m}$ applies to d-dimensional densities. It requires that $\hat{f}^A_{\boldsymbol{a}}(\mathbf{x})$ is decreasing along any ray emanating from $\mathbf{m}$. We will consider the $d = 2$ case. Where previously the constraints were enforced at a collection of G scalar points, they are now enforced at $\{\mathbf{g}_l\}$, $l = 1, \ldots, G$, a set of points in two-dimensional space. The arrangement of these points over the support of the density is practically important, but does not affect how the QP problem is set up.

Star unimodality can be checked by confirming that the directional derivative of the density along the appropriate ray is negative. Let $\mathbf{u}_l$ represent the unit vector in the direction of $\mathbf{g}_l$ from $\mathbf{m}$, that is

$$\mathbf{u}_l = \frac{\mathbf{g}_l - \mathbf{m}}{\|\mathbf{g}_l - \mathbf{m}\|}.$$

The directional derivative at $\mathbf{g}_l$ in the direction of $\mathbf{u}_l$ is $(\nabla \hat{f}^A_{\boldsymbol{a}}(\mathbf{g}_l))^T \mathbf{u}_l$, where $\nabla \hat{f}^A_{\boldsymbol{a}}$ is the gradient of the estimate:

$$\begin{aligned} \nabla \hat{f}^A_{\boldsymbol{a}}(\mathbf{g}_l) &= \nabla \hat{f}^\circ(\mathbf{g}_l) + \nabla \left(\boldsymbol{a}^T \boldsymbol{\psi}(\mathbf{g}_l)\right) \\ &= \nabla \hat{f}^\circ(\mathbf{g}_l) + \begin{bmatrix} \boldsymbol{a}^T \boldsymbol{\psi}_1'(\mathbf{g}_l) \\ \boldsymbol{a}^T \boldsymbol{\psi}_2'(\mathbf{g}_l) \end{bmatrix}, \end{aligned}$$

where $\boldsymbol{\psi}_i'(\mathbf{y})$ is the derivative of $\boldsymbol{\psi}$ with respect to y_i. So, letting $\mathbf{u}_l = [u_1^l \ u_2^l]^T$, the constraint is

$$\begin{aligned} (\nabla \hat{f}^A_{\boldsymbol{a}}(\mathbf{g}_l))^T \mathbf{u}_l &= (\nabla \hat{f}^\circ(\mathbf{g}_l))^T \mathbf{u}_l + \begin{bmatrix} \boldsymbol{a}^T \boldsymbol{\psi}_1'(\mathbf{g}_l) & \boldsymbol{a}^T \boldsymbol{\psi}_2'(\mathbf{g}_l) \end{bmatrix} \begin{bmatrix} u_1^l \\ u_2^l \end{bmatrix} \\ &= (\nabla \hat{f}^\circ(\mathbf{g}_l))^T \mathbf{u}_l + \boldsymbol{a}^T \left(u_1^l \boldsymbol{\psi}_1'(\mathbf{g}_l) + u_2^l \boldsymbol{\psi}_2'(\mathbf{g}_l)\right). \end{aligned}$$

Because this quantity must be less than or equal to zero, the constraint at point $\mathbf{g}_l$ is

$$\left(u_1^l \boldsymbol{\psi}_1'(\mathbf{g}_l) + u_2^l \boldsymbol{\psi}_2'(\mathbf{g}_l)\right)^T \boldsymbol{a} \quad \leq \quad -(\nabla \hat{f}^\circ(\mathbf{g}_l))^T \mathbf{u}_l.$$

Combining the constraints at all $\mathbf{g}_l$ produces the system of inequalities.

Star unimodality constraint

Require $\mathbf{A_4}\boldsymbol{a} \leq \mathbf{b_4}$, where

$$\underset{G\times k}{\mathbf{A_4}} = -\begin{bmatrix} u_1^1 \boldsymbol{\psi}_1'(\mathbf{g}_l)^T + u_2^1 \boldsymbol{\psi}_2'(\mathbf{g}_1)^T \\ \vdots \\ u_1^G \boldsymbol{\psi}_1'(\mathbf{g}_G)^T + u_2^G \boldsymbol{\psi}_2'(\mathbf{g}_G)^T \end{bmatrix} \quad \text{and} \quad \mathbf{b_4} = -\begin{bmatrix} (\nabla \hat{f}^\circ(\mathbf{g}_1))^T \mathbf{u}_1 \\ \vdots \\ (\nabla \hat{f}^\circ(\mathbf{g}_G))^T \mathbf{u}_G \end{bmatrix}.$$

Bibliography

Alibrandi, U. and Ricciardi, G. (2008), "Efficient evaluation of the pdf of a random variable through the kernel density maximum entropy approach," *International Journal for Numerical Methods in Engineering*, 75, 1511–1548.

Antoniou, A. and Lu, W. (2007), *Practical optimization: algorithms and engineering applications*, Springer-Verlag New York Inc.

Barlow, R. E., Bartholomew, R. J., Bremner, J. M., and Brunk, H. D. (1972), *Statistical Inference Under Order Restrictions*, London: John Wiley & Sons.

Bickel, P. J. and Fan, J. (1996), "Some Problems on the Estimation of Unimodal Densities," *Statistica Sinica*, 6, 23–45.

Birgé, L. (1997), "Estimation of Unimodal Densities Without Smoothness Assumptions," *The Annals of Statistics*, 25, 970–981.

Birke, M. (2009), "Shape Constrained Kernel Density Estimation," *Journal of Statistical Planning and Inference*, 139, 2851–2862.

Braun, W. J. and Hall, P. (2001), "Data Sharpening for Nonparametric Inference Subject to Constraints," *Journal of Computational and Graphical Statistics*, 10, 786–806.

Brunk, H. D. (1955), "Maximum Likelihood Estimates of Monotone Parameters," *The Annals of Mathematical Statistics*, 26, pp. 607–616.

Cheng, M.-Y., Gasser, T., and Hall, P. (1999), "Nonparametric Density Estimation under Unimodality and Monotonicity Constraints," *Journal of Computational and Graphical Statistics*, 8, 1–21.

Choi, E. and Hall, P. (1999), "Data Sharpening as a Prelude to Density Estimation," *Biometrika*, 86, 941–947.

Clerc, M. and Kennedy, J. (2002), "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *Evolutionary Computation, IEEE Transactions on*, 6, 58 –73.

Cule, M., Samworth, R., and Stewart, M. (2010), "Maximum likelihood estimation of a multi-dimensional log-concave density," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72, 545–607.

de Castro, L. N. (2006), *Fundamentals of Natural Computing*, Chapman & Hall.

Devroye, L. and Lugosi, G. (2001), *Combinatorial Methods in Density Estimation*, Springer.

Du, P., Parmeter, C. F., and Racine, J. S. (2010), "Nonparametric Kernel Regression with Multiple Predictors and Multiple Shape Constraints," `http://www.stat.vt.edu/facstaff/pangdu/papers/2010_kern.pdf`.

Dümbgen, L. and Rufibach, K. (2009), "Maximum likelihood estimation of a log-concave density and its distribution function: Basic properties and uniform consistency," *Bernoulli*, 15, 40–68.

Engelbrecht, A. P. (2005), *Fundamentals of Computational Swarm Intelligence*, Wiley.

Fan, J. and Li, R. (2001), "Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties," *Journal of the American Statistical Association*, 96, 1348–1360.

Farmen, M. and Marron, J. (1999), "An assessment of finite sample performance of adaptive methods in density estimation," *Computational statistics & data analysis*, 30, 143–168.

Fougères, A.-L. (1997), "Estimation de densites unimodales," *The Canadian Journal of Statistics / La Revue Canadienne de Statistique*, 25, 375–387.

Grenander, U. (1956), "On the Theory of Mortality Measurement, Part II," *Skandinavisk Aktuarietidskrift*, 39, 125–153.

Gupta, S. D. (1976), "S-Unimodal Function: Related Inequalities and Statistical Applications," *Sankhya: The Indian Journal of Statistics, Series B (1960-2002)*, 38, pp. 301–314.

Hall, P. and Huang, L.-S. (2001), "Nonparametric Kernel Regression Subject to Monotonicity Constraints," *The Annals of Statistics*, 29, pp. 624–647.

— (2002), "Unimodal Density Estimation Using Kernel Methods," *Statistica Sinica*, 12, 965–990.

Hall, P. and Kang, K.-H. (2005), "Unimodal Kernel Density Estimation by Data Sharpening," *Statistica Sinica*, 15, 73–98.

Hall, P. and Minnotte, M. C. (2002), "High Order Data Sharpening for Density Estimation," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 64, 141–157.

Hastie, T. and Tibshirani, R. (1987), "Non-Parametric Logistic and Proportional Odds Regression," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 36, 260–267.

Hastie, T., Tibshirani, R., and Friedman, J. (2009), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., Springer.

Henderson, D. J. and Parmeter, C. F. (2009), "Imposing Economic Constraints in Nonparametric Regression: Survey, Implementation, and Extension," *Advances in Econometrics*, 25, 433–469.

Jones, M. and Henderson, D. (2005), "Maximum Likelihood Kernel Density Estimation," Technical Report 01/05, The Open University, UK.

Jones, M. C. and Henderson, D. A. (2009), "Maximum likelihood kernel density estimation: On the potential of convolution sieves," *Computational Statistics and Data Analysis*, 53, 3726–3733.

Kennedy, J. and Eberhart, R. (1995), "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, pp. 1942 –1948 vol.4.

Kennedy, J., Eberhart, R. C., and Shi, Y. (2001), *Swarm Intelligence*, Morgan Kaufmann.

Kennedy, J. and Mendes, R. (2002), "Population structure and particle swarm performance," in *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, vol. 2, pp. 1671 –1676.

Klemelä, J. (2009), *Smoothing of Multivariate Data*, Wiley.

Lange, K. (2010), *Numerical Analysis for Statisticians*, 2nd ed., Springer.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2010), "Iterated Local Search: Framework and Applications," in *Handbook of Metaheuristics*, eds. M. Gendreau and J.-Y. Potvin, Springer, pp. 363–397.

Meyer, M. (2008), "Inference using shape-restricted regression splines," *The Annals of Applied Statistics*, 2, 1013–1033.

Michalewicz, Z. and Fogel, D. B. (2004), *How to Solve it: Modern Heuristics*, 2nd ed., Springer-Verlag.

Nocedal, J. and Wright, S. J. (1999), *Numerical Optimization*, Springer.

Numerical Algorithms Group (2009), *NAG Toolbox for Matlab, Mark 21E*, Numerical Algorithms Group, Oxford, UK.

Paquet, U. and Engelbrecht, A. (2003), "A new particle swarm optimiser for linearly constrained optimisation," in *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, vol. 1, pp. 227 – 233 Vol.1.

Poli, R., Kennedy, J., and Blackwell, T. (2007), "Particle Swarm Optimization: An Overview," *Swarm Intelligence*, 1, 33–57.

Ramsay, J. and Silverman, B. (2005), *Functional Data Analysis*, 2nd ed., Springer.

Ramsay, J. O. (1988), "Monotone Regression Splines in Action," *Statistical Science*, 3, pp. 425–441.

Reboul, L. (2005), "Estimation of a Function under Shape Restrictions. Applications to Reliability," *The Annals of Statistics*, 33, pp. 1330–1356.

Scott, D. W. (1992), *Multivariate Density Estimation: Theory, Practice, and Visualization*, John Wiley and Sons.

Scott, D. W. and Factor, L. E. (1981), "Monte Carlo Study of Three Data-Based Nonparametric Probability Density Estimators," *Journal of the American Statistical Association*, 76, pp. 9–15.

Sheather, S. J. and Jones, M. C. (1991), "A reliable data-based bandwidth selection method for kernel density estimation," *Journal of the Royal Statistical Society, Series B*, 53, 683–690.

Silverman, B. W. (1986), *Density Estimation for Statistics and Data Analysis*, Chapman and Hall.

Talbi, E.-G. (2009), *Metaheuristics: From Design to Implementation*, Wiley.

The Mathworks, Inc. (2007), *MATLAB Version 7.4.0*, Natick, Massachusetts.

Wand, M. and Jones, M. (1995), *Kernel Smoothing*, London: Chapman & Hall.

Wasserman, L. (2006), *All of Nonparametric Statistics*, Springer.

Wegman, E. J. (1972), "Nonparametric Probability Density Estimation: I. A Summary of Available Methods," *Technometrics*, 14, 533–546.

Wolters, M. A. (2009), "A Greedy Algorithm for Unimodal Kernel Density Estimation by Data Sharpening," Technical Report TR-09-01, Department of Statistical and Actuarial Sciences, University of Western Ontario, `http://ir.lib.uwo.ca/statspub/2/`.

— (2011), "A particle swarm algorithm with broad applicability in shape-constrained estimation," *Computational Statistics & Data Analysis*, in press, DOI: 10.1016/j.csda.2011.11.009.

— (2012), "A Greedy Algorithm for Unimodal Kernel Density Estimation by Data Sharpening," *Journal of Statistical Software*, 47, 1–26.

Woolford, D. G. and Braun, W. J. (2007), "Convergent data sharpening for the identification and tracking of spatial temporal centers of lightning activity," *Environmetrics*, 18, 461–479.

Curriculum Vitae

Name:	Mark A. Wolters
Education:	Ph.D. (statistics), University of Western Ontario, 2012
	M.Sc. (statistics), Simon Fraser University, 2007
	B.Sc. (metals and materials engineering), University of British Columbia, 1996
Awards:	NSERC Canada Graduate Scholarship D, 2009–2010
	Ontario Graduate Scholarship, 2008, 2009
	Best Student Poster Award, Statistical Society of Canada meetings, 2009
	SFU Graduate Fellowship Award, 2006
	ASQ Statistics Division's Ellis R. Ott Scholarship in Applied Statistics, 2006
	NSERC Industrial Postgraduate Scholarship 1, 2004–2005
Work Experience:	Instructor, Statistical Science 1023A (Statistical Concepts), fall 2009
	Instructor, Statistical Science 2024A (Introduction to Statistics), fall 2008 (13 lectures)
	Product development engineer, Ballard Power Systems, Burnaby, BC, 1998–2004
Peer-Reviewed Publications:	Wolters, M. A. and Bingham, D. (2011), "Simulated Annealing Model Search for Subset Selection in Screening Experiments," *Technometrics*, 53, 225–237.
	Wolters, M. A. (2011), "A Particle Swarm Algorithm with Broad Applicability in Shape-Constrained Estimation," *Computational Statistics and Data Analysis*, DOI: 10.1016/j.csda.2011.11.009.
	Wolters, M. A. (2012), "A Greedy Algorithm for Unimodal Kernel Density Estimation by Data Sharpening," *Journal of Statistical Software*, 47(6), 1–26.

Other Publications:

Wolters, M. A. (2009), “A Greedy Algorithm for Unimodal Kernel Density Estimation by Data Sharpening,” Technical Report TR-09-01, Department of Statistical and Actuarial Sciences, University of Western Ontario, http://ir.lib.uwo.ca/statspub/2/

Wolters, M. A. (2008), “A Critical Look at Risk Measures: Why Statistics Don’t Always Mean What they Say,” *Fraser Forum*, Feb. 2008.

Mark Wolters (2007), “Real Risks: Statistical Thinking and Risk Perception,” *Fraser Institute Digital Publications*, Dec. 2007.