2009

A Greedy Algorithm for Unimodal Kernel Density Estimation by Data Sharpening

Mark A. Wolters
The University of Western Ontario, mwolter@uwo.ca

A Greedy Algorithm for Unimodal Kernel Density Estimation By Data Sharpening

Mark A. Wolters

March 4, 2009

Contents

Chapter 1

Introduction

Nonparametric methods for smoothing, regression, and density estimation produce estimators with great shape flexibility. Although this flexibility is an advantage, the practical value of nonparametric methods would be increased if *qualitative constraints*—natural-language shape restrictions—could also be imposed on the estimator. In density estimation, the most common such constraints are monotonicity (the density must be nondecreasing or nonincreasing) and unimodality (the density must have only one peak).

The work presented here takes unimodal kernel density estimation as a representative problem in constrained nonparametric estimation. The method proposed for handling the constraint is *data sharpening*. The following section provides a brief review of the unimodal kernel density estimation problem and the data sharpening approach to solving it. Afterwards, the goals and scope of the current work are discussed in more detail.

1.1 Unimodal Kernel Density Estimation

Let $\mathbf{x}$ be a set of n independent observations from a distribution with pdf f. The kernel density estimator (KDE), denoted by $\hat{f}_{\mathbf{x}}$, is

$$\hat{f}_{\mathbf{x}}(u) = \frac{1}{n}\sum_{i=1}^{n} K_h(u - x_i), \tag{1.1}$$

where K_h is a kernel function with bandwidth h. The Gaussian kernel will be used exclusively here, so K_h is taken to be a N(0,h^2) density. It is assumed that h is known, or chosen by some automatic rule. For more information on kernel density estimation, see Wand and Jones (1995).

The density estimate (1.1) with Gaussian kernel is an evenly-weighted mixture of n normal densities, with one mixture component centered at each data point, and each com-

ponent having variance h^2. If h is sufficiently small, the estimate $\hat{f}_{\mathbf{y}}$ will have n modes; 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.

The problem of interest here is to obtain a kernel density estimate that is forced to be unimodal. A few possibilities exist. The weight on the i^{th} point ($i = 1, 2, \ldots, n$) could be changed from $1/n$ to some probability p_i that ensures a single mode (Hall and Heckman 2002). Alternatively, a different bandwidth could be used at each point; the resulting bandwidth vector $\mathbf{h}$ could be constrained to enforce unimodality[1]. The monotone rearrangement method of Fougeres (1997) could be used; it starts with a standard KDE and uses transformations to produce a unimodal density. There are, in addition, several other non-kernel-based methods for unimodal or monotone density estimation and the related problem of isotonic regression. See Racine and Parmeter (2008), Silvapulle and Sen (2005), and Cheng et al. (1999) for further information and references.

1.2 Data Sharpening

Data sharpening (Braun and Hall 2001; Hall and Kang 2005) is the method of constraint handling to be pursued here. This method modifies the data, rather than modifying the estimator, to accommodate the constraint. A new set of "sharpened" data, $\mathbf{y}$, is formed by finding a minimal perturbation of $\mathbf{x}$ that results in a unimodal KDE. The sharpened estimate of f is then just $\hat{f}_{\mathbf{y}}$—estimator (1.1) is used without modification on the sharpened data $\mathbf{y}$.

Figure 1.2 gives an example of how sharpening can be used to make the third case of Figure 1.1 ($h = 0.5$) unimodal. A unimodal estimate can be achieved with only a small shift of the leftmost point toward the other four points. In this case the best way to shift the points was obvious, but in real problems it will be unclear which perturbation of the data yields the best unimodal KDE. The present research deals with the problem of designing an automatic procedure to make the data perturbations in a sensible way.

The data sharpening concept does not depend on the statistical method being used. As such, sharpening is potentially applicable to a wide range of nonparametric estimators and constraint types. Extension to higher-dimensional data is also conceptually straightforward. This potential generality of data sharpening is its major advantage over other constraint-

[1]It appears that this method of *adaptive kernels* has not been applied to handling constraints.

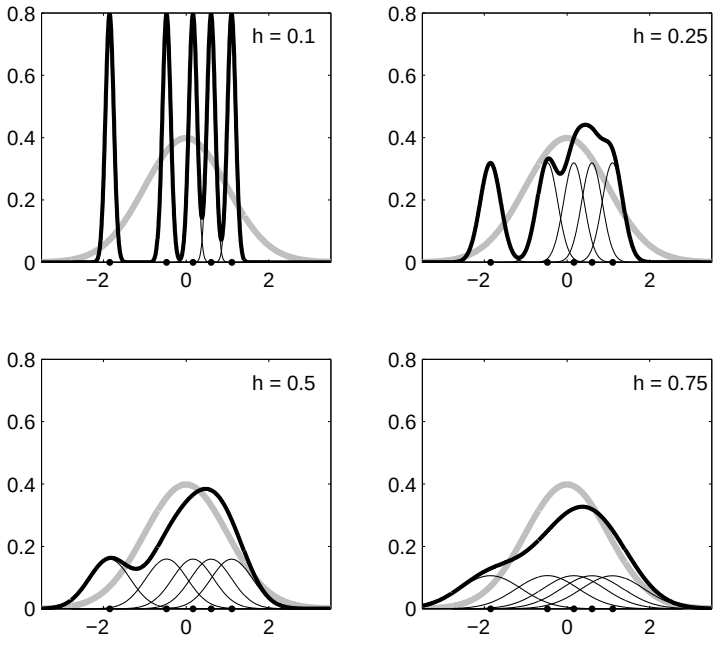

Figure 1.1: Kernel density estimates for a small sample using four different bandwidths. The data are sampled from a standard normal distribution (thick grey curve). The dark lines are the density estimates. The kernel functions for each point are also shown.

handling methods. Other methods are typically very difficult to extend beyond their original setting.

1.3 Goals and Scope

The difficulty in achieving the potential of data sharpening is in actually finding $\mathbf{y}$, the sharpened data set. The difficulty is twofold: first, how does one define which $\mathbf{y}$ is best; and second, how does one actually find this solution? The remainder of this report constitutes a first step toward answering these questions. The unimodal KDE problem is used as a vehicle for development, but the ultimate goal is to develop procedures that can be used more generally. For this reason, every effort will be made to avoid dependence on the characteristics of kernel density estimators or the unimodality constraint.

This report has the following three goals:

1. Introduce the unimodal KDE data sharpening problem, and thoroughly discuss the issues involved in data sharpening optimization. Justify the use of heuristic optimiza-

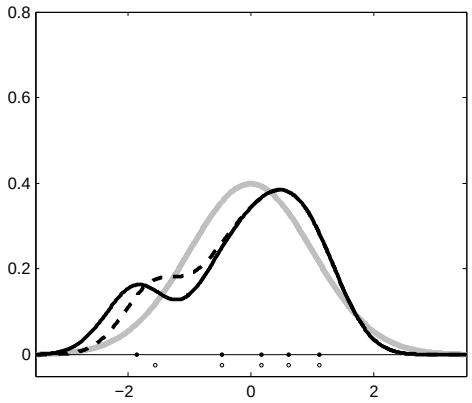

Figure 1.2: The third KDE shown in figure 1.1 ($h = 0.5$), rendered unimodal by data sharpening. The dark solid line is the original KDE, based on the filled circles. The dashed line is sharpened estimate, based on the open circles.

tion methods for the general data sharpening problem.

2. Describe in detail the `improve` function that has been developed to perform data sharpening. This function uses a greedy optimization heuristic. While it may be useful in its own right, it is primarily envisioned as a building block for more sophisticated algorithms in the future.

3. Lay the foundation for future developments, such as sharpening with other estimators, other constraints, higher dimensions, and so on.

It should be noted that while the generality of data sharpening has been stressed, the work presented here will be narrow in focus. Only kernel density estimation will be considered; the only constraint tested will be unimodality; and the development will only consider scalar data. The problem of bandwidth selection will not be addressed, nor will the question of the statistical properties of sharpened estimators. All such extensions and related topics are deferred to future work to allow a focus on the core optimization problem.

Chapter 2

The Sharpening Optimization Problem

The goal of data sharpening is to find the "best" solution that satisfies a chosen qualitative constraint—in this case, unimodality of a KDE. The sharpening optimization problem will now be defined and explored in detail. Particular emphasis will be placed on understanding the sources of difficulty in setting up and solving the problem.

2.1 Problem Definition

As in the introduction, let $\mathbf{x}$ be the observed vector of n data points, and $\mathbf{y}$ be any proposed sharpened data vector. The kernel density estimates based on $\mathbf{x}$ and $\mathbf{y}$ are $\hat{f}_{\mathbf{x}}$ and $\hat{f}_{\mathbf{y}}$, respectively. Let $\delta(\mathbf{y}, \mathbf{x})$ be a smaller-the-better objective function defining which $\mathbf{y}$ are best, and let $\mathbf{y}^*$ be the minimizer of this function.

2.1.1 Known Mode

If the estimator is constrained to have its mode at a known location m, the optimization problem is a search for $\mathbf{y}^*$, where

$$\mathbf{y}^* = \text{argmin } \delta(\mathbf{y}, \mathbf{x}) \quad \text{subject to } \begin{cases} \hat{f}'_{\mathbf{y}}(x) \geq 0 & \text{if } x \leq m \\ \hat{f}'_{\mathbf{y}}(x) \leq 0 & \text{if } x \geq m, \end{cases} \tag{2.1}$$

which makes it clear that unimodality is a derivative constraint on the density estimator.

In a practical implementation, the constraints of (2.1) will be enforced only at a grid of

points $\mathbf{g} = (g_1, \ldots, g_G)$ covering the support of the estimate[1]. If m falls between gridpoints k and $k+1$, the problem may be formulated as

$$\mathbf{y}^* = \text{argmin } \delta(\mathbf{y}, \mathbf{x}) \quad \text{subject to } \begin{cases} \hat{f}'_{\mathbf{y}}(g_i) \geq 0, & i = 1, \ldots, k \\ \hat{f}'_{\mathbf{y}}(g_i) \leq 0, & i = k+1, \ldots, G. \end{cases} \tag{2.2}$$

This formulation converts the unimodality constraints into a set of G explicit inequalities involving $\mathbf{y}$.

Expressed as (2.2), the sharpening problem looks like a standard problem in constrained nonlinear optimization, for which there is extensive theory and numerous algorithms available. The algorithm of primary interest here is sequential quadratic programming (SQP), which has been used by others (Braun and Hall 2001; Hall and Huang 2002; Hall and Kang 2005; Racine and Parmeter 2008) to perform nonparametric inference subject to constraints. Nocedal and Wright (1999) provide a thorough discussion of SQP and related methods.

2.1.2 Unknown Mode

In the more realistic case where the mode is unknown, the problem is more complicated because m (or k) depends on $\mathbf{y}$. One way to proceed is to treat m as fixed, solve the sharpening problem, and then repeat the process according to a 1-D optimization scheme to find the best value of m. This allows the constraints to be handled explicitly as in (2.2), but with a corresponding computational cost.

It is desirable (in terms of generality, and possibly performance and speed as well) to design an algorithm capable of performing data sharpening directly. The algorithm would simply search for the best solution satisfying the constraint. The optimization problem is then the search for

$$\mathbf{y}^* = \underset{\mathbf{y} \in \mathcal{C}}{\text{argmin}} \ \delta(\mathbf{y}, \mathbf{x}), \tag{2.3}$$

where $\mathcal{C}$ is the set of all *feasible* solutions (those giving unimodal KDEs):

$$\mathcal{C} = \left\{ \mathbf{y} : \exists m \ \text{ such that } \begin{array}{ll} \hat{f}'_{\mathbf{y}}(x) \geq 0 & \text{if } x \leq m \\ \hat{f}'_{\mathbf{y}}(x) \leq 0 & \text{if } x \geq m \end{array} \right\}. \tag{2.4}$$

For computer implementation one would use a discretized version of (2.4) in terms of g_i and k, as in (2.2).

In this general formulation, it is not possible to express the unimodality constraint as a set of inequalities in $\mathbf{y}$. Though this makes it impossible to use standard deterministic

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

algorithms such as SQP, there are many heuristic search techniques that are potentially well suited to this problem. These heuristics require only a feasibility check or test function that evaluates whether or not a candidate solution $\mathbf{y}$ satisfies the constraint.

A practical way to test for feasibility is to get exact or approximate derivative values of $\hat{f}_{\mathbf{y}}$ at the grid points $\mathbf{g}$, and then count the number of modes based on the number of sign changes. An algorithm that only requires this feasibility check could simultaneously choose the mode and perform data sharpening.

2.2 Possible Objective Functions

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

2.2.1 Objectives Based on Data Vectors

Because the sharpened data are considered perturbations of the original data, it is natural to take the objective to be a measure of distance between vectors. One option is to base $\delta(\mathbf{y}, \mathbf{x})$ on a norm of the difference $\mathbf{x} - \mathbf{y}$, defining

$$L_\alpha(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{n} |y_i - x_i|^\alpha, \qquad 1 \leq \alpha \leq 2. \tag{2.5}$$

The parameter α in (2.5) 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), 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 i^{th} dimension at $y_i = x_i$.

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

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

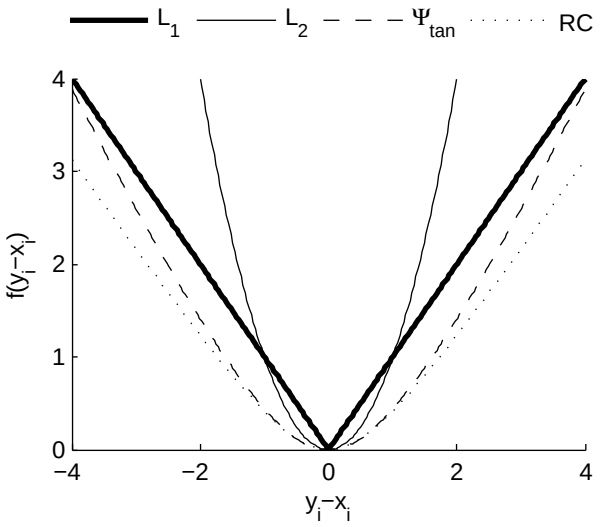

Figure 2.1: Summands of four objective functions.

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 the present work, the following rounded-corners objective will be used instead:

$$RC(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{n} \sqrt{1 + (y_i - x_i)^2} - 1. \tag{2.6}$$

The summand of this function is a hyperbola in $y_i - x_i$. Figure 2.1 compares the summands of L_1, L_2, $\Psi_{\tan}$, and RC. The RC objective achieves the same aims as $\Psi_{\tan}$, but without the need for integration[2].

The goal of data sharpening problem is to find a good density estimate that satisfies the constraint, but the optimization problem is posed in terms of data vectors, not density estimates. The set of solutions $\{\mathbf{y}\}$ has a many-to-one mapping onto the set of density estimates $\{\hat{f}_{\mathbf{y}}\}$, because the KDE is invariant to permutations of the sharpened data points 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 (2.5) or (2.6) 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.

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

[2]Ideally one would like to have a controllable amount of curvature at the vertex, but neither $\Psi_{\tan}$ nor RC have this property.

start with $\mathbf{x}$ sorted in ascending order and then sort any proposed $\mathbf{y}$ before calculating the objective function. A sorted L_α objective can then be defined as:

$$L^s_\alpha(\mathbf{y}, \mathbf{x}) = L_\alpha(\text{sort}(\mathbf{y}), \mathbf{x}) = \sum_{i=1}^{n} |y_{(i)} - x_{(i)}|^\alpha, \qquad 1 \leq \alpha \leq 2, \tag{2.7}$$

where $y_{(i)}$ represents the i^{th} largest point in $\mathbf{y}$. The sorted version of RC can be similarly defined and denoted $RC^s(\mathbf{y}, \mathbf{x})$.

An optimization algorithm 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 $\mathbf{x}$. 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.

2.2.2 Objectives Based on Density Estimates

Another natural approach to choosing an objective function is to use a metric based on the sharpened and unsharpened density estimates, $\hat{f}_\mathbf{y}$ and $\hat{f}_\mathbf{x}$. 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

$$ISE(\mathbf{y}, \mathbf{x}) = \int_{-\infty}^{\infty} (\hat{f}_\mathbf{x}(t) - \hat{f}_\mathbf{y}(t))^2 dt, \tag{2.8}$$

$$KL(\mathbf{y}, \mathbf{x}) = \int_{-\infty}^{\infty} \hat{f}_\mathbf{y}(t) \ln \left(\frac{\hat{f}_\mathbf{y}(t)}{\hat{f}_\mathbf{x}(t)} \right) dt, \tag{2.9}$$

$$TV(\mathbf{y}, \mathbf{x}) = \frac{1}{2} \int_{-\infty}^{\infty} |\hat{f}_\mathbf{x}(t) - \hat{f}_\mathbf{y}(t)| dt. \tag{2.10}$$

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 $KL(\mathbf{y}, \mathbf{x}) \neq KL(\mathbf{x}, \mathbf{y})$.

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, $|P_{\hat{f}_\mathbf{x}}(B) - P_{\hat{f}_\mathbf{y}}(B)| \leq TV(\mathbf{y}, \mathbf{x})$. That is, $TV(\mathbf{y}, \mathbf{x})$ is the maximum possible difference attainable when the same probability is calculated with the two density estimates $\hat{f}_\mathbf{y}$ and $\hat{f}_\mathbf{x}$.

Objectives based directly on the estimates have the advantage of being insensitive to the ordering of $\mathbf{y}$ and $\mathbf{x}$. On the other hand, the density-based objectives are specific to the

density-estimation context, and would not apply if, for example, the sharpening methods were used in a monotone regression problem.

2.2.3 A Likelihood Objective

A final objective function to be considered is based on the negative log-likelihood of the original data $\mathbf{x}$ under the sharpened density $\hat{f}_{\mathbf{y}}$. That is, define

$$LIK(\mathbf{y}, \mathbf{x}) = -\sum_{i=1}^{n} \ln \hat{f}_{\mathbf{y}}(x_i). \tag{2.11}$$

The LIK measure, like KL, is not commutative. It is justified on the basis of the principle of maximum likelihood. Using this objective, the goal of sharpening is to find the unimodal KDE that assigns greatest likelihood to the observed data.

The motivation for using likelihood as an objective is to avoid the awkward situation where the sharpened estimator puts negligible density over an observed data point. This could happen if there is an outlying observation and the estimator (using, say, an L_1 objective) pulls this outlier into the main group of points. The likelihood objective will strongly penalize such solutions[3].

One potential drawback of LIK as an objective is that its unconstrained minimum value (the maximum likelihood) might be achieved at some point other than the unsharpened data $\mathbf{x}$. In this case the objective of minimizing $LIK(\mathbf{y}, \mathbf{x})$ will conflict with the goal of moving $\mathbf{y}$ toward $\mathbf{x}$, and data sharpening becomes more like fitting a normal mixture model than like performing a modification to a KDE[4].

2.3 Visualizing the Problem

The most important aspects of the sharpening optimization problem will now be demonstrated using two examples. In both examples, sharpening is done by moving only two of the data points, so that the objective and constraint can be visualized on the plane. The major sources of difficulty will be made clear, as motivation for the new search heuristic to follow.

[3]It is an open question, however, which type of tail behaviour is actually most desirable when doing density estimation by data sharpening.

[4]Choosing the sharpened data to minimize (2.11) is equivalent to maximum likelihood fitting of an equally-weighted, constant-variance normal mixture with unknown means. It can be shown by counterexample that the unsharpened KDE $\hat{f}_{\mathbf{x}}$ is not, in general, the maximum likelihood estimate for this situation.

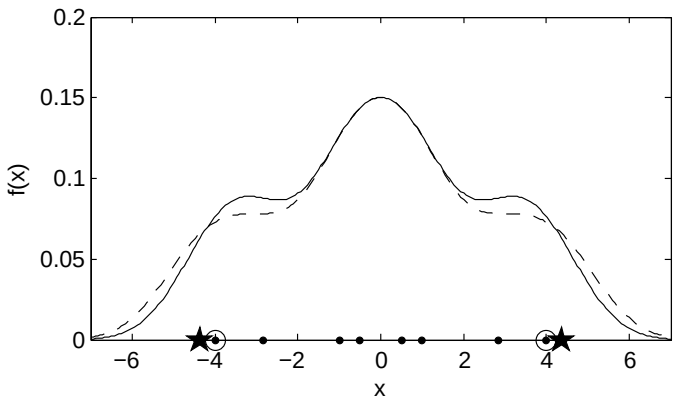

Figure 2.2: An example of data sharpening by moving two points. Filled circles and solid line show the unsharpened data and density estimate, respectively. Circles indicate the two moveable points. Stars show the L_1-best solution. The dashed line is the corresponding density estimate.

2.3.1 Example 1: Non-Convexity

The first example illustrates the characteristics of the objective functions and the constraint set. 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 points. The problem has solutions of the form (y_1, y_8).

The unsharpened estimate for this example is symmetric with three modes, as shown in Figure 2.2. The L_1-optimal solution is also shown on the figure. It is found by shifting each of the two moveable points slightly away from the center of the distribution.

Figure 2.3 shows this problem's objective function contours for the eight objectives defined in (2.5) through (2.11). 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. 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, reflecting the symmetry of the original problem.

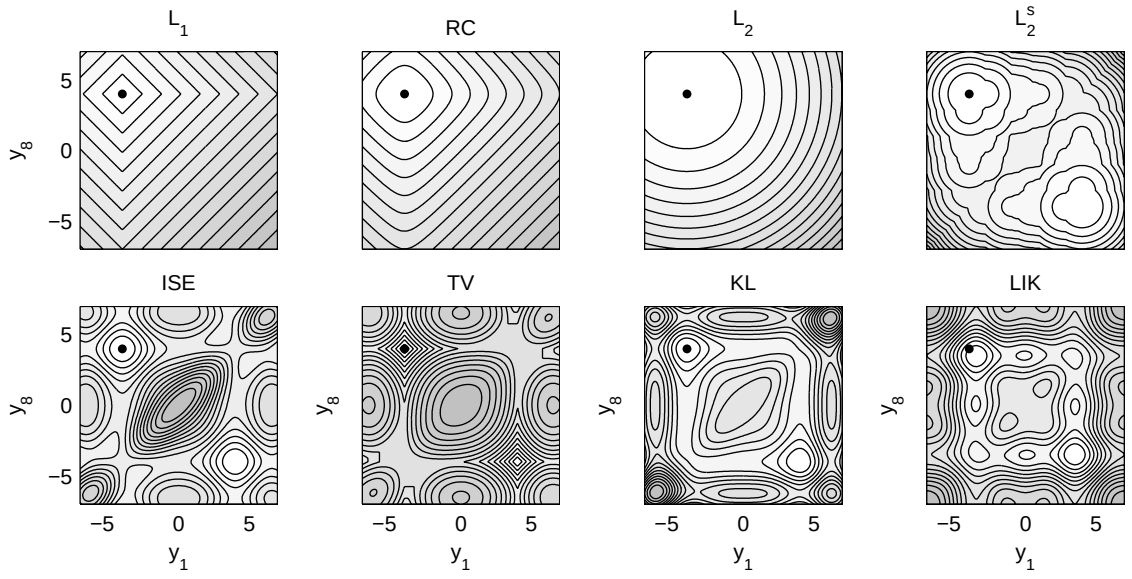

Figure 2.3: Contour plots of eight objective functions over the search space for the example of Figure 2.2. Higher (worse) regions are shaded in gray. The dots show the unsharpened solution $(-4, 4)$. All axes have the same scaling.

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.

Whether or not the chosen objective function is convex, the most significant difficulties in sharpening optimization will be due to the shape of the feasible set, $\mathcal{C}$. In the present example, $\mathcal{C}$ can be considered a region in the solution space. Figure 2.4 shows the constraint region superimposed on the L_1 and L_1^s contours. The constraint region for this case is not only non-convex, it is not even contiguous. Finding the best solution on the constraint boundary is a very difficult problem, even with only two points to optimize.

The plots above each set of contours show the unsharpened KDE and the best sharpened estimates for the two objectives (the best estimates are the same for both objectives). Note that when the sorted objective is used the objective function presents a more truthful picture of solution space. Both of the minima on the L_1^s surface are global minima, and they both correspond to the same density estimate.

The figure also shows the performance of a sequential quadratic programming algorithm as it searches for an optimum from different starting points. The starting points are indi-

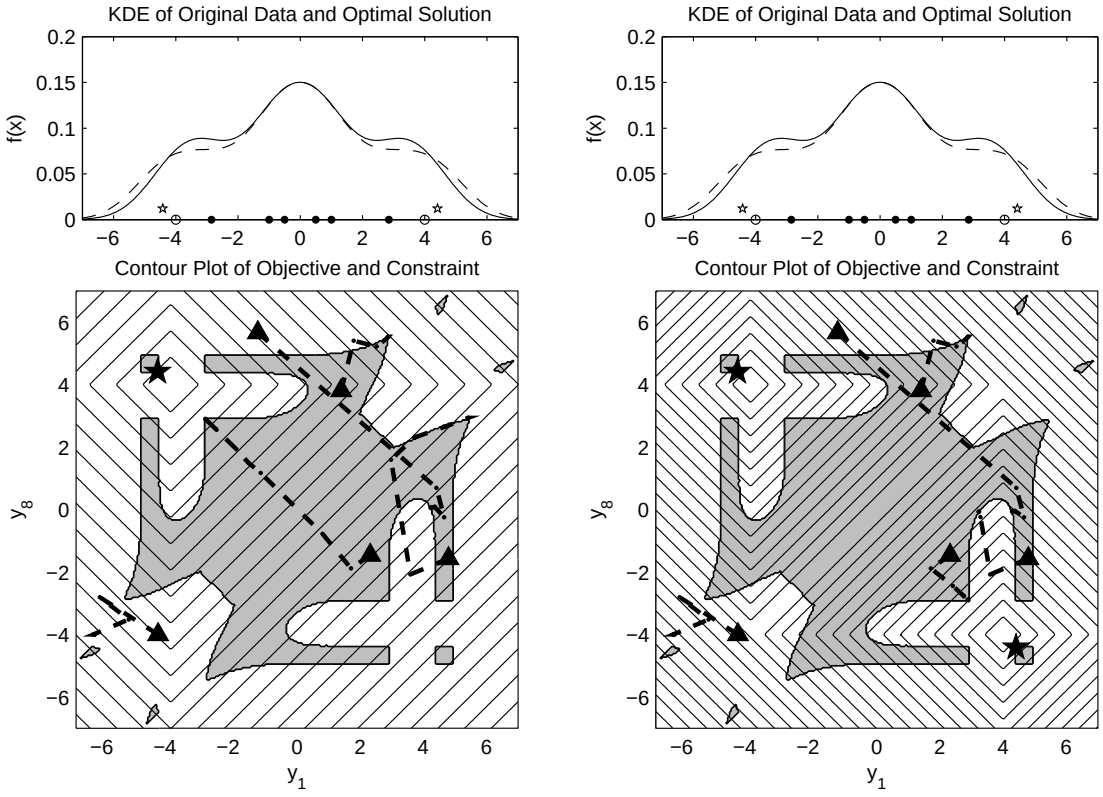

Figure 2.4: A Unimodal KDE problem by data sharpening, with only two points being moved. The top two graphs show the original data (dots), the two moveable points x_1 and x_8 (circles), and the L_1-optimal solution (stars). The bottom two graphs show the solution space, with the feasible region $\mathcal{C}$ overlaid on the contours of the objective function. The left plot is for the usual L_1 objective, and the right is for the sorted version, L_1^s. The dashed lines show solution progress for optimization by SQP, for several starting values (triangles).

cated 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 complex constraint region, and was not able to find the true optimum from any of the cases. In general, SQP was not successful unless the initial value was very close to the optimum.

2.3.2 Example 2: Multiple Solutions

A second example will show that the sharpening problem can have a multiplicity of solutions in two different senses. First, there may be more than one solution with optimal or near-optimal objective values. Second, the solution can change considerably when different objective functions are used.

The unsharpened and sharpened data vectors in this new example are:

$$\begin{aligned} \mathbf{x} &= [\ -2.25 \quad -1.25 \quad 1.25 \quad 2.25\ ]^T \\ \mathbf{y} &= [\ \quad y_1 \quad -1.25 \quad 1.25 \quad y_4\ ]^T. \end{aligned}$$

Figure 2.5 repeats the plots of Figure 2.4 for this example, this time using objective functions L_1^s and L_2^s. The unsharpened estimate is symmetric and bimodal. The constraint region is again non-convex, and the sorted objective functions are non-convex as well.

The optimal solution is not unique for either objective function. In both cases there are four optimal (y_1, y_4) pairs, corresponding to two best estimates that are mirror images of each other. The best estimates also differ between the two objective functions, suggesting the need for a principled way to decide which objective function makes a best overall choice.

All of the optimal solutions in this example lie on the constraint boundary. In larger problems, optimal solutions will often involve many points sitting just on the boundary. This adds complexity to the optimization, since the high-dimensional constraint boundary is generally only discoverable through trial and error.

2.4 Summary

The classical formulation of constrained optimization is

$$\text{minimize } f(\mathbf{y}) \quad \text{subject to } \begin{cases} u_i(\mathbf{y}) = 0, & i = 1, \ldots, I \\ v_j(\mathbf{y}) \geq 0, & j = 1, \ldots, J. \end{cases} \tag{2.12}$$

The functions $u_i(\mathbf{y})$ and $v_j(\mathbf{y})$ define equality and inequality constraints that must be satisfied by the optimal solution.

Problems of the form (2.12) are classified according to the characteristics of the objective and constraint functions. So-called *linear programming* problems arise when both the objective function and the constraint are linear in the unknown variables; *nonlinear programming* refers to all problems where the objective function is not linear. Among nonlinear programming problems, those with quadratic objectives and linear constraints are called *quadratic programming* problems, while those with general convex objectives and general convex constraints are termed *convex programming* problems (Hillier and Lieberman 1980, ch. 19). Well-established global optimization algorithms exist for each of these problem classes.

The unimodal KDE data sharpening problem only fits into the standard form (2.12) when the mode is pre-specified, as in (2.2). Even in that case, the problem does not fall

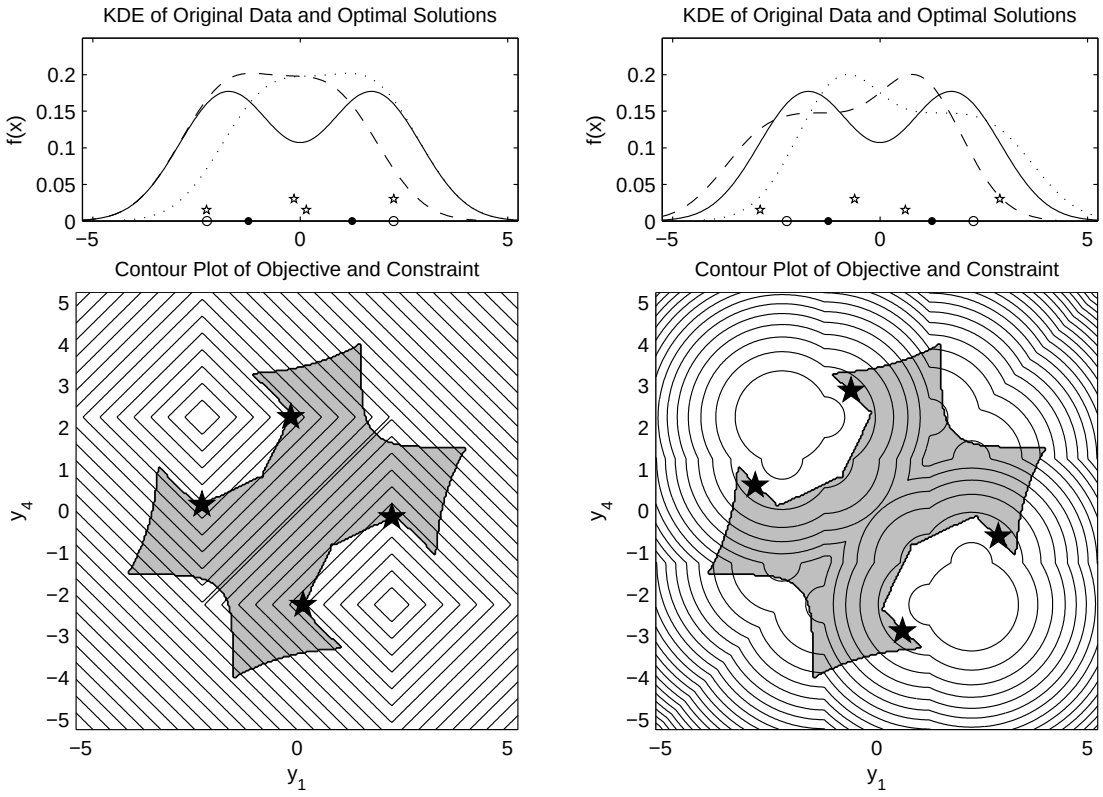

Figure 2.5: A second example of unimodal KDE by moving two points (see figure 2.4 for a description of the layout). The left case uses the L_1^s objective, and the right uses L_2^s. In both cases there are two optimal estimates, and these optima are different for the different objectives.

into any of the aforementioned standard categories. It has non-convex constraints and, depending on the choice of $\delta(\mathbf{y}, \mathbf{x})$, it may have a non-convex objective as well. The high dimensionality of the problem (dimension n) and the potential for multiple optima only exacerbate optimization difficulties. There are no general algorithms that can guarantee achievement of a global optimum for problems of this type. The best that can be hoped for is convergence to a local optimum.

Recent research on constrained nonparametric estimation (Hall and Kang 2005; Racine and Parmeter 2008) has focused on convex objective functions and used SQP to find solutions when the constraints are not convex. But SQP is not totally suitable for problems with non-convex constraints: it may find the global minimum, it could return a poorer local optimum, or it could even fail to converge. The outcome depends on the quality of the initial guess and the complexity of the constraint.

When a problem is not suitable for convergent deterministic optimization, it is preferable to use one of the various heuristic methods available for such problems. Heuristic approaches are usually more flexible, and therefore more capable of getting good approximate solutions across a variety of difficult problems. The remainder of this report presents a greedy heuristic that could form a part of data sharpening algorithms with greater reliability and generality than SQP.

Chapter 3

A Greedy Data Sharpening Algorithm

The term *heuristic* is used to describe optimization methods that are defined primarily through their search algorithms. They are usually applied when exhaustive search or standard function optimization approaches do not work.

The simplest way to develop an optimization heuristic is often the *greedy* approach. Greedy algorithms break the overall optimization problem, which may be complicated, into a series of simple steps. At each step, the solution is improved as much as possible. The advantage of greedy algorithms is their simplicity, but it is usually the case that greed in early decisions forces the search to make poor moves later on—resulting in sub-optimal solutions.

Below, a greedy approach is presented for data sharpening in the unimodal KDE problem. While the limitations of greedy methods are acknowledged, the new algorithm has one major advantage: it will always provide a solution satisfying the unimodality constraint. One of the most challenging and problem-dependent aspects of algorithm design is constraint handling, so a simple method with guaranteed constraint-satisfaction could be a very useful component of more sophisticated heuristics in the future[1].

The greedy method for unimodal kernel density estimation will now be described, first as a general approach, and then in two specific implementations.

[1]There are many broad categories of more sophisticated heuristics. See Michalewicz and Fogel (2004) for a good review of evolutionary algorithms and many strategies for constraint handling; see Engelbrecht (2005) for particle swarm algorithms. Other approaches include simulated annealing and tabu search.

3.1 Structure of the Algorithm

The premise of the proposed algorithm is to start with an initial guess solution that is feasible, and to improve this guess by moving its points closer to the original data. Points are moved one at a time, and the algorithm may cycle through the points repeatedly as long as the solution keeps improving. A critical requirement is that feasibility must be maintained throughout. No point may be moved in a way that causes the density estimate to gain a second mode.

The sharpened solution vector, to be modified throughout the search, is denoted $\mathbf{y}$. The initial solution is $\mathbf{v}$, and the original ("unsharpened") data is $\mathbf{x}$. The vectors $\mathbf{y}$ and $\mathbf{x}$ are assumed to be matched elementwise, and each x_i, $i = 1, \ldots, n$ is called the *target* or *home* position for the corresponding y_i. This terminology reflects the goal of sharpening, which is to achieve unimodality while keeping the y_i as close as possible to the x_i.

The solution is improved during the algorithm by moving each y_i toward home. When the unimodality constraint prevents a point from moving closer to home, the point is said to be *pinned*. A point is considered *moveable* if it is neither pinned nor at home.

The flow chart in Figure 3.1 shows the structure of the algorithm. The central step is a *sweep* or *pass* through all moveable points in $\mathbf{y}$. In each pass, every moveable point is moved closer to its target position. Starting with the initial guess, the algorithm sweeps through the data repeatedly. Between sweeps, the sharpened data are re-ordered to match the ordering of the original data, and the sequence in which the points will next be processed is determined. The search terminates when there are no moveable points, that is, when all sharpened points are either at home or are pinned by the constraint boundary.

3.2 Implementation Issues

Figure 3.1 also indicates six implementation issues that must be addressed to produce working code. These issues are discussed below, starting from the last and proceeding to the first.

Choice of Tolerance

Issue. Any numerical optimization routine will rely on a tolerance specification to determine convergence. The tolerance is an adjustable parameter in the algorithm. Its setting will influence the computation time and, more importantly, could also affect the quality of solutions found.

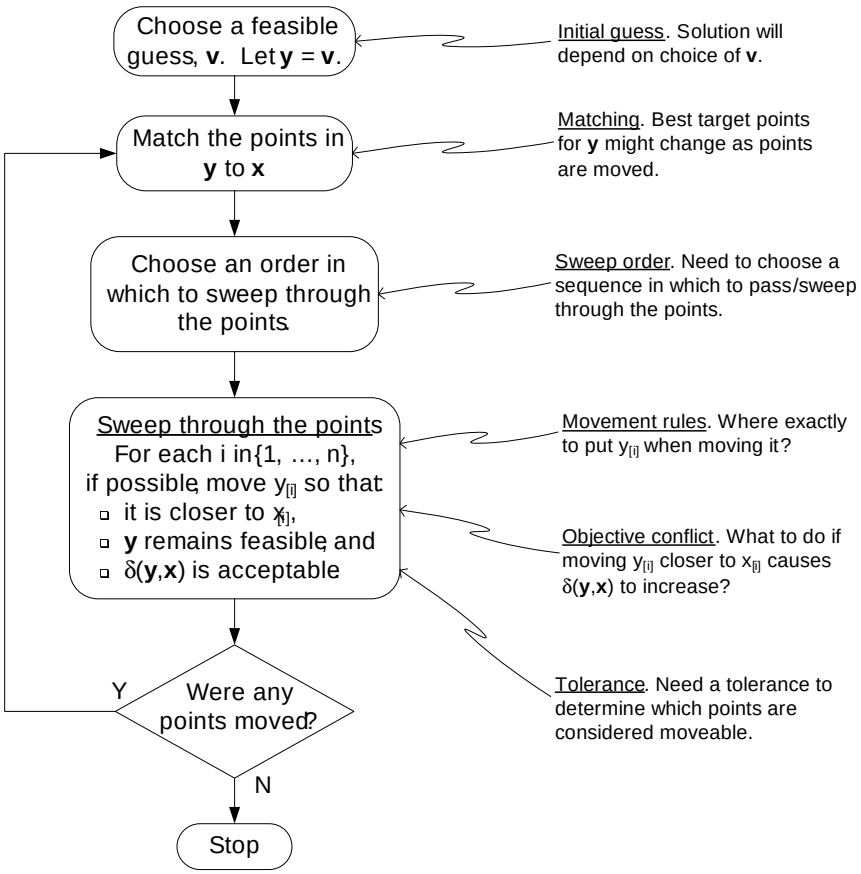

Figure 3.1: A high-level description of the greedy algorithm, indicating issues that need to be resolved to create practical code.

Resolution. A single tolerance τ will be used where necessary to control the precision of calculations. The tolerance helps to determine when points have reached home:

$$y_i \text{ is home} \quad \Leftrightarrow \quad |x_i - y_i| \leq \tau. \tag{3.1}$$

It also is used to determine when the constraint is active. A point is considered pinned if shifting it by a distance τ causes a constraint violation:

$$y_i \text{ is pinned} \quad \Leftrightarrow \quad \text{setting } \; y_i := y_i + \text{sign}(x_i - y_i)\tau \; \text{ renders } \mathbf{y} \text{ infeasible.} \tag{3.2}$$

In section 3.4, two versions of the algorithm will be introduced, based on two different line search tactics: bisection and grid search. Both line searches will use the tolerance τ to assess convergence. It will be shown that insensitivity to τ is one of the main advantages of the grid search version of the algorithm.

Objective Conflict

Issue. Depending on the objective function used, it is possible that the goal of moving $\mathbf{y}$ closer to $\mathbf{x}$ may conflict with the overall goal of minimizing $\delta(\mathbf{y}, \mathbf{x})$. This cannot happen with objectives of the L_α type, but it can be a problem with objective functions based on the density estimate and not on the solution vector $\mathbf{y}$.

Resolution. A solution to the problem is to include a reduction in the objective function as a part of the constraint. If $\mathbf{v}$ is the initial solution and $\mathbf{y}'$ is the intermediate solution formed by moving $y_{[i]}$, the notion of feasibility is slightly modified:

$$y_i \text{ is feasible} \quad \Leftrightarrow \quad \mathbf{y} \in \mathcal{C} \; \text{ and } \; \delta(\mathbf{y}', \mathbf{x}) \leq \delta(\mathbf{v}, \mathbf{x}). \tag{3.3}$$

That is, each move is accepted as long as the solution is unimodal and the objective is no greater than the starting objective value. With this restriction the algorithm will always return a solution vector that is no further from $\mathbf{x}$ than the initial guess and a density estimate that is no worse, in terms of the objective function, than the initial estimate. It should be reiterated that definition (3.3) results in no change when the objective is based on the L_α or RC metrics.

Rules for Moving a Point

Issue. The proposed algorithm is greedy in the sense that each time a single point is moved, its movement is optimized without considering other points or the future progress of the

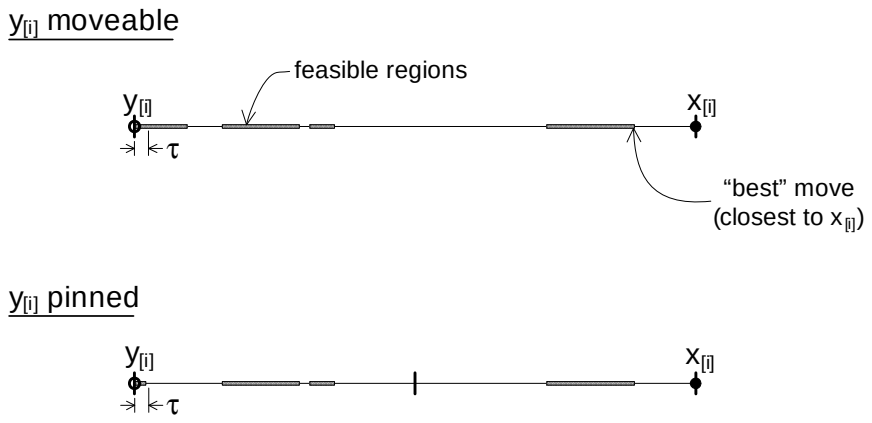

Figure 3.2: An illustration of the 1-D movement problem for the case of a moveable point (top) and a pinned point (bottom). A point is pinned if a shift of length τ toward the target causes a constraint violation.

search. In principle the algorithm will work with any movement rule that causes points to be moved closer to home at each step. The particular choice of movement rule will, however, have considerable impact on the performance of the algorithm and on the details of its implementation.

Consider the movement of a sharpened point from its original location y_i to some new value y_i'. For simplicity assume that the point is moving to the right, which means $y_i < x_i$. The set of possible values for y_i' is the interval $[y_i, x_i]$. Setting $y_i' := y_i$ means no movement, and setting $y_i' := x_i$ means movement all the way home.

Typically only a subset of values in $[y_i, x_i]$ will yield feasible solutions. The set of feasible values is not known and does not have to be contiguous. Each move is a 1-D optimization problem with the goal of finding the largest value in the feasible subset of $[y_i, x_i]$. The problem is illustrated in Figure 3.2.

Resolution. First note that not every point is moveable. To be moveable, a point must be neither pinned nor at home. Considering equations 3.1 and 3.2, this means

$$y_i \text{ is moveable} \quad \Leftrightarrow \quad \begin{cases} \mathbf{y} \text{ remains feasible when } \; y_i := y_i + \text{sign}(x_i - y_i)\tau \\ |x_i - y_i| > \tau. \end{cases} \tag{3.4}$$

Figure 3.2 illustrates the difference between moveable and pinned points. Note that because the feasible set of solutions need not be contiguous, a point may be pinned despite having a feasible solution closer to the target. Such a point is still not considered moveable, however, because the existence of better solutions is only discoverable by inspection.

Since a 1-D optimization problem needs to be solved for each moveable point in each pass through the data, it is important to have an efficient routine for finding good solutions. The bisection and grid search versions of the greedy algorithm resolve this problem in different ways. They will be discussed in sections 3.4.1 and 3.4.2, respectively.

Sweep Order

Issue. The order in which the points are to be moved must be determined before each pass through the data. The choice of sweep order will influence the performance of the algorithm, because moving any one point can influence the ability of other points to move. It is not sensible to sweep through the points in sequence from y_1 to y_n, as this could introduce a positional bias in the results. Another rule for sweeping must be devised.

Several possible rules could be constructed. For example, the points could be moved in random order, or starting in the tails, or starting in the center of the distribution. The rule that was found to work most reliably, however, is based on the distances between the points and their home positions.

Resolution. The standard rule is to move the points in descending order of their distance from home: the points with the greatest value of $|y_i - x_i|$ are moved first. If all points start in the central part of the distribution, this rule has the effect of moving points toward the tails first, and then moving interior points that are closer to home.

When discussing the movement of individual points during a sweep, the notation $y_{[i]}$ will be used to denote the moveable point i^{th} farthest from home; $x_{[i]}$ will denote its matching element of $\mathbf{x}$. This notation was also used in Figure 3.1.

Matching of Sharpened and Unsharpened Data

Issue. The idea of matching elements of $\mathbf{y}$ to those in $\mathbf{x}$ was introduced in section 2.2 as a way to make the search invariant to permutations of $\mathbf{y}$. In the context of algorithm development, matching is also important to ensure that the elements of $\mathbf{y}$ are being moved toward sensible targets. After a single pass through the data, some points may have crossed over others, meaning that the point-to-target distances can be reduced by reordering $\mathbf{y}$. For this reason a re-matching step between passes can improve performance.

Resolution. A natural way to match points in one-dimensional problems is to arrange $\mathbf{y}$ so that its j^{th} largest element is in the same position as the j^{th} largest element of $\mathbf{x}$, that is, match $y_{(j)}$ to $x_{(j)}$. If the original data starts out in sorted order, this amounts to simply

sorting $\mathbf{y}$ between passes[2].

Adding a sorting step to the algorithm introduces the possibility that the search could take backward steps or even cycle endlessly by revisiting previous arrangements of $\mathbf{y}$. To prevent this, the search is limited to only accepting new rearrangements of the points—the matching step may never result in an ordering of $\mathbf{y}$ that was previously used[3]. Implementing this restriction on matching requires the algorithm to store a record of previous matching arrangements.

Dependence on Initial Guess

Issue. The only requirement on the starting solution $\mathbf{v}$ is that it must produce a unimodal kernel density estimate. The algorithm will produce a solution from any feasible starting point, but the solution will depend on the particular choice of $\mathbf{v}$. The best choice of $\mathbf{v}$ is likely to be problem-specific, but it is desirable to have a simple rule for generating a starting solution regardless of the particular data at hand.

Resolution. The default way to form a starting solution is to put all of the sharpened data points at the location of the highest mode of the unsharpened KDE. If m_0 is this location, the initial guess is

$$\underset{n\times 1}{\mathbf{v}} = (m_0, \ldots, m_0)^T. \tag{3.5}$$

Experience has shown that this simple method works well and prevents points from getting pinned too early in the search[4]. The location of the highest mode can be easily approximated by evaluating the unsharpened kernel density estimator at a grid of points.

3.3 An Illustration

A small example can illustrate the main features of the greedy method. Five data points were generated from a standard normal distribution and used to construct a kernel density

[2]The question of best matching for multivariate data is much more difficult, because meaningful sorting or ordering operations are harder to define for vectors. This problem will not be addressed here.

[3]It has been observed on test problems that preventing reversion to old matchings does more than just preventing cycles; it also improves performance of the greedy algorithm (especially the grid search version of section 3.4.2).

[4]Another simple option is to start with the sharpened data points evenly spaced out around the mean, median, or mode of the original data, producing a flat mound as the initial density estimate. This option performs poorly, since interior points are nearly pinned. Small movements of these points will introduce extra modes in the estimate.

estimate: $\mathbf{x} = [-0.7885 \quad -0.4245 \quad 0.1745 \quad 0.4587 \quad 0.9341]^T$. To encourage a multimodal estimate, the bandwidth was chosen to be $h = 0.264$, which is half of the value chosen by the normal scale bandwidth selection rule (Wand and Jones 1995, p.60). The greedy algorithm just described was used on these five points to produce a unimodal estimate[5].

The algorithm completed after only three passes through the data. The initial state and the results after each pass are shown in Figure 3.3. The sharpened solution and its corresponding density estimate are shown as open circles and a solid line, respectively. The unsharpened data and solution are also shown on the figure. At each stage of the algorithm, lines are drawn to connect each sharpened point to its target $\mathbf{x}$ value. A text label on each line indicates the status of the corresponding point. For moveable points, the label is a number indicating the order in which that point is to be moved on the next pass. For non-moveable points, the label is either an h if the point is at home, or a p if the point is pinned.

The upper left figure shows the initial state of the algorithm. All sharpened points are placed at the location of the highest unsharpened mode. All points are moveable and are to be moved in descending order of their distance from their targets.

The upper right figure shows the results of the first pass. One point has arrived home, and the others have all been moved closer to their targets. Note that the two rightmost points have exchanged their targets during the re-matching stage (the thin grey lines show the matching before adjustment, and the thick grey lines show the matching after adjustment). This is because the point moved third was not able to move very far, while the point moved fifth was able to cross over it and move all the way home. The matching step puts the sharpened vector in a more natural order and facilitates further improvement.

The bottom two figures show the results after passes two and three. No further matching adjustments were required in these steps. During the second pass, another point was able to reach home, and two others reached the constraint boundary, leaving only one moveable point. During the third step, this point was also moved as far as possible toward home, leaving all points in the home or pinned states and causing the algorithm to terminate.

Note that throughout the algorithm, the sharpened solution maintained feasibility. The reason the three pinned points are not moveable after three passes is the plateau to the left of the mode. Moving any of the pinned points would introduce a second mode by causing this plateau to become a trough.

[5]This example uses the bisection method to be described in section 3.4.1

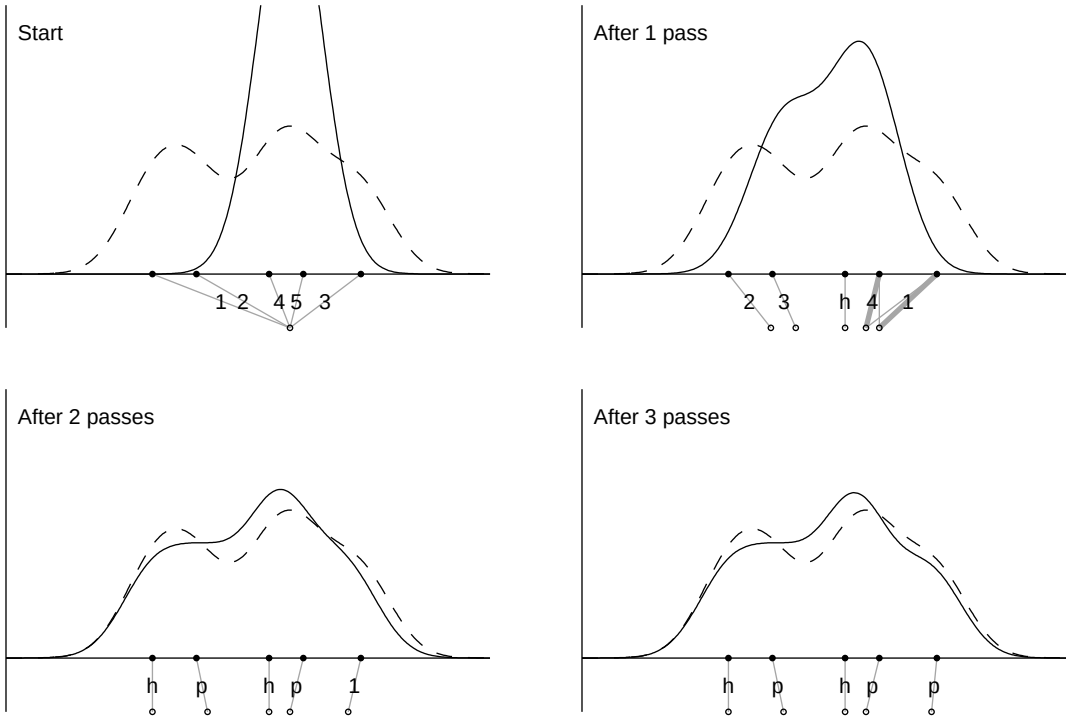

Figure 3.3: A small example illustrating the greedy sharpening method. Solid/dashed lines show the sharpened/unsharpened estimates. Open/filled circles show the sharpened/unsharpened data. Grey lines join each unsharpened point to its target and indicate the status of the point. Note in the upper right that two points have switched targets during the matching step (thick grey lines indicate targets after re-matching). All figures have equal axis scaling.

3.4 The `improve` Function

The ideas of the preceding sections have been put together in a function called `improve`. The function is so named because when $\mathbf{y} = \texttt{improve}(\mathbf{v}, \mathbf{x})$ is called, the result $\mathbf{y}$ will have $L_\alpha(\mathbf{y}, \mathbf{x}) \leq L_\alpha(\mathbf{v}, \mathbf{x})$. Any unimodal starting solution will be transformed into another unimodal solution with an improved objective function value[6].

The main detail not specified in the preceding sections was the form of the line search involved in the movement of each point. In the following three sections, two versions of `improve` are discussed and then compared. The first version uses bisection to do the line search, and the second uses a type of grid search.

3.4.1 Initial Approach: Bisection

The point-moving problem depicted in Figure 3.2 can be solved using bisection. Starting with an interval known to contain the solution, the interval is repeatedly divided in half. At each division the half not containing the solution is discarded, and the process continues until the interval is sufficiently small.

Assuming the point is moving to the right as in the figure, the search interval is $[y_i, x_i]$. The solution to be found is a location where the point y_i' shifts from being feasible to being infeasible. The point y_i is moveable, so by definition (see equation 3.4) there must be at least one such point in the interval. The midpoint $(x_i + y_i)/2$ can be evaluated for feasibility. If it is feasible, search focuses on the right half of the interval; if not, search focuses on the left half. This process continues until the search interval has width less than τ. Note that, because the feasible region does not need to be contiguous, there is no guarantee that the best solution will be found.

A short pseudocode describing the bisection version of `improve` is given in Table 3.1. A more detailed pseudocode, including details of the bisection step, is shown in Table A.1 in the Appendix.

A discussion of the performance of the bisection approach is deferred to section 3.4.3, where the bisection and grid-search approaches are compared. In that section, it will be shown that using bisection gives the `improve` function both poor performance and high sensitivity to the tolerance τ.

[6]Strictly speaking, there must be at least one moveable point in order for the function to return an improvement.

Table 3.1. The `improve` *Function—Bisection Approach.*

```
Set up the initial guess and match it to x.
Find the set of moveable points.
Set the sweep order.
WHILE there are still moveable points:
    FOR each moveable point:
        Use bisection to move the point closer to home.
    END FOR
    Re-match the updated solution to x.
    Find the set of moveable points.
    Re-set the sweep order.
END WHILE
```

3.4.2 Improved Approach: Shrinking Grid

Grid search is a line search method even more basic than bisection. A simple grid search could be used for the point-moving problem of Figure 3.2 as follows: test solutions y_i' at a sequence of locations in the interval $[y_i, x_i]$, and then choose the largest y_i' that gives a feasible density estimate.

This simplistic grid search is computationally intensive and thus not desirable for the `improve` function. Instead, the search is modified in two ways:

1. The grid search will proceed from y_i toward x_i and will terminate as soon as the first infeasible y_i' is found. That is, the search will "step out" from the sharpened point toward home, and not all grid points will be tested.

2. The number of grid steps, S, will be modified globally throughout the search. The algorithm will start with $S = 1$, meaning that the search tries to move points directly to home. After a complete pass through the data, if no points were able to move, the number of steps will be doubled. This process continues, with the number of grid steps successively doubling whenever no points can be moved.

These two modifications greatly improve the efficiency of grid search. They also allow data sharpening to start with large moves first, and then to move on to smaller and smaller refinements of the estimate as the search continues. This helps to overcome the tolerance sensitivity and early-pinning problems of the bisection approach.

The nature of this shrinking-grid search is illustrated in Figure 3.4 by looking at a single point. For each of six passes through the data, 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

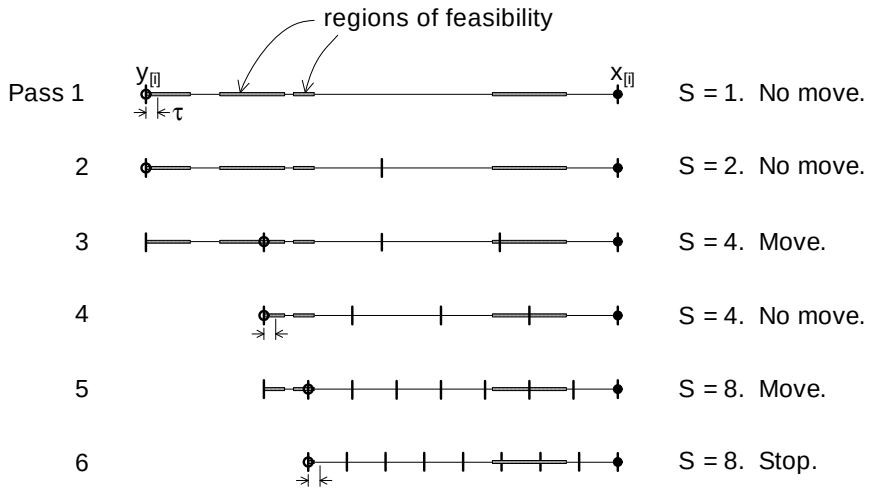

Figure 3.4: A schematic illustration of the grid search for a single point. Two facts are not clearly depicted here: i) the grid steps are doubled only when none of the n points move; and ii) the feasible region or target x might change between passes, since they can be changed by the movement of other points.

results in either a successful step out or a doubling of the grid. Search terminates when the point ceases to be moveable.

An important fact that is not properly depicted in Figure 3.4 is that the grid-doubling step happens only when *none* of the moveable points are able to step out at the current value of S. The algorithm tries to move points as much as possible at the current coarse grid setting, and only shrinks the grid when no more improvement can be made. This is shown more clearly in the pseudocode of Table 3.2 (and in the more detailed pseudocode found in Table A.2). Comparison with the bisection code (Table 3.1) shows that the two algorithms are structurally very similar. As long as points are moving, the grid-search algorithm proceeds in exactly the same way as the bisection algorithm; but once points stop moving, the process is interrupted to double the resolution of the grid. The ultimate termination of the algorithm does not depend on the value of S; it occurs when no more points are considered moveable based on the tolerance τ.

3.4.3 Comparing the Two Approaches

Performance of the two versions of `improve` was evaluated using a test case from Braun and Hall (2001). The true density in this example is a mixture of $N(-1, 0.6^2)$, $N(1, 2.5^2)$, and $N(5, 1.5^2)$ densities, with mixing proportions 0.35, 0.5, and 0.15, respectively. This density was constructed to be unimodal, but with a long, flat shoulder so that kernel density

Table 3.2. The `improve` *Function—Grid Search Approach.*

```
Set up the initial guess and match it to x.
Find the set of moveable points.
Set the sweep order.
Set the number of grid search steps to S := 1.
WHILE there are still moveable points:
    FOR each moveable point:
        Use grid search with S steps to move the point closer to home.
    END FOR
    IF at least one point has moved:
        Re-match the updated solution to x.
        Find the set of moveable points.
        Re-set the sweep order.
    ELSE
        Double the number of grid-search steps: S := 2S.
    END IF
END WHILE
```

estimates will usually be multimodal. The density is shown in Figure 3.5.

Braun and Hall (2001) performed data sharpening on this example, with a sample size of $n = 25$, using an $L_{1.375}(\mathbf{x}, \mathbf{y})$ objective function and a Gaussian kernel density estimator. They used an off-the-shelf sequential quadratic programming routine to find the solution (with a separate optimization step to determine the best mode location), and found that a bandwidth of approximately $h = 0.6$ was optimal in terms of mean integrated squared error (MISE) between the sharpened estimate and the truth. The MISE at this optimal bandwidth was about 0.0155.

To permit comparison with these earlier results, the bandwidth was fixed at 0.6 throughout the present simulations. One thousand samples were drawn from the test distribution, and both versions of the `improve` algorithm were run on each sample at seven different tolerance levels: $\tau = 10^{-k}, k = 2, 3, \ldots, 8$.

Figure 3.6 shows the MISE performance of the bisection and grid search methods as a function of tolerance. The MISE of the unsharpened (and usually not unimodal) estimates is shown for reference. The grid search outperforms bisection at all values of τ and also shows much less sensitivity to τ. Grid search also gave MISE values better than the unsharpened estimate, and approaching the SQP results of the Braun and Hall paper (MISE = 0.0155).

The tolerance sensitivity of bisection visible in figure 3.6 is actually flattering to the bisection method because of averaging across the 1000 simulation runs. Inspection of individual runs shows that the bisection method does not actually converge, in the sense that the resulting density estimates should stop changing as the tolerance is decreased. Figure

Table 3.3. Convergence of solutions.

Tolerance		Mean $\|\|\mathbf{y}_{\tau_2} - \mathbf{y}_{\tau_1}\|\|$	
τ_1	τ_2	bisection	grid search
10^{-2}	10^{-3}	1.29	0.186
10^{-3}	10^{-4}	1.01	0.0501
10^{-4}	10^{-5}	1.49	0.0145
10^{-5}	10^{-6}	2.19	0.00515
10^{-6}	10^{-7}	2.47	0.00102
10^{-7}	10^{-8}	2.13	0.000409

3.7 shows typical results from a single simulation run. The `improve` function with bisection produces seven different density estimates at the seven different tolerance levels, while the density estimates hardly change when the grid search is used instead.

The convergence problem is further illustrated in Table 3.4.3, where the effect of shrinking the tolerance is shown in terms of the average change in the Euclidean distance between solution vectors. The grid search behaves in a convergent manner: the solutions change less and less as the tolerance is successively shrunk. Bisection, on the other hand, shows no sign of convergence. Solutions based on tolerances of 10^{-7} and 10^{-8} are just as different as solutions based on tolerances of 10^{-2} and 10^{-3}.

The poor performance of bisection as the line search is due to the search working too hard to optimize individual moves. The movement of one point has strong interactions on the ability of other points to move. Moving a particular point very close to the constraint boundary can cause other points to become pinned prematurely. Enforcing a small tolerance early in the search is also computationally wasteful. At the smallest tolerance of 10^{-8} the average run time was roughly 70 percent longer for bisection than for grid search[7].

The shrinking-grid approach avoids the problem of over-searching by moving all points at a coarse level first. The search only works hard to get close to the constraint boundary later in the search, when the points are already close to their final positions. In this sense the grid search is less greedy than the bisection search.

Based on these results, the shrinking-grid version of `improve` will be taken as the default function in subsequent discussion and future work.

[7]It is possible to adaptively shrink the tolerance of the bisection as the algorithm proceeds, and this does improve the performance of bisection. But the method then becomes sensitive to the choice of initial tolerance, and the best choice depends on the scale of the problem. The shrinking-grid method avoids this concern by making the line search independent of problem scale.

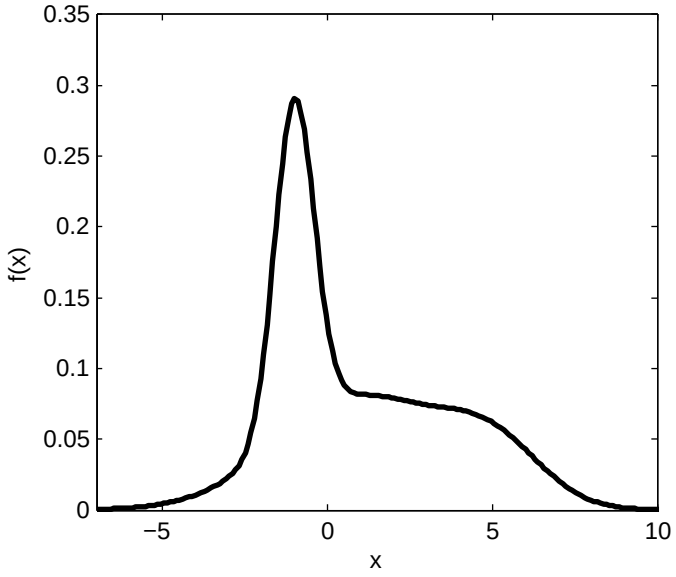

Figure 3.5: True distribution for performance evaluation of the bisection and grid-search methods.

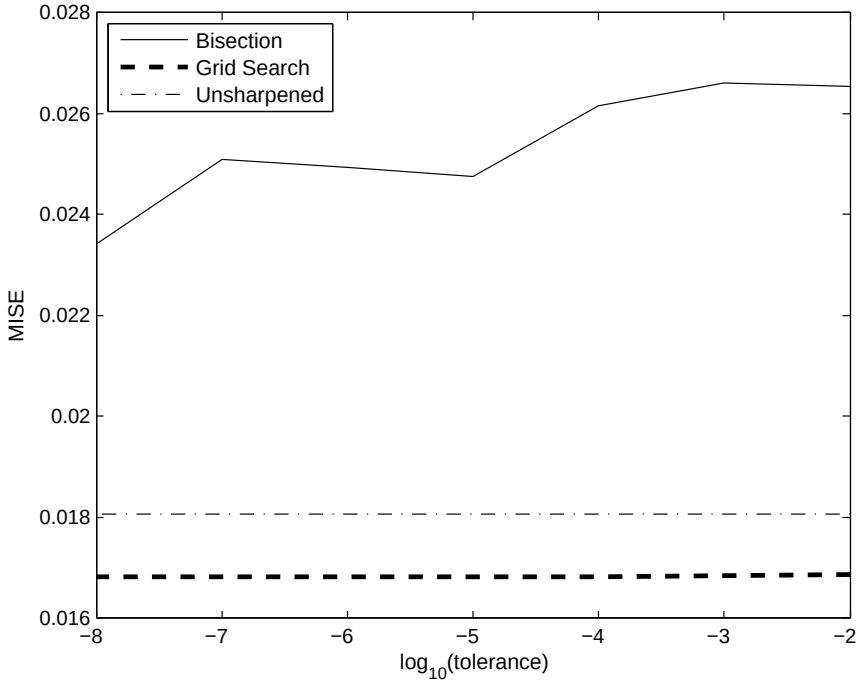

Figure 3.6: MISE versus log tolerance for both versions of `improve` and for the raw data.

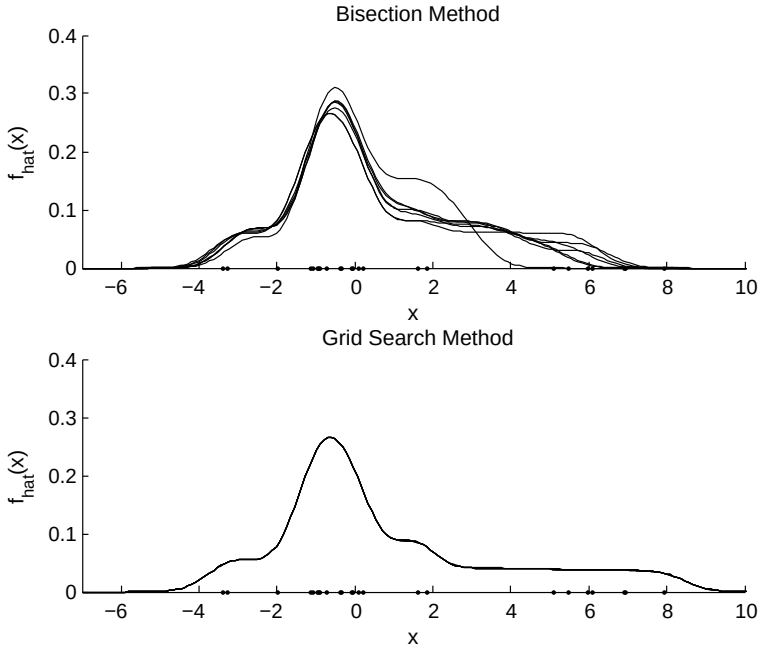

Figure 3.7: Variation among solutions at different tolerance levels for a typical simulation run. Top: using bisection, there is high variability in the density estimates as the tolerance is changed. Bottom: using grid search, the effect of tolerance is barely discernable. In both plots, the points show the unsharpened data.

Chapter 4

Performance Evaluations

The relative merits of sequential quadratic programming and the new `improve` function were explored through simulation, using two challenging unimodal KDE problems.

4.1 Study Design

The premise of the study was to generate multiple data vectors from test cases and then to perform data sharpening on each data set using different combinations of bandwidth, optimization method, and objective function. All estimation was done using the Gaussian KDE.

A test case is defined by a true density and a sample size. Due to time constraints, only two densities and a single sample size were used:

True density. Density 1 is the t distribution with three degrees of freedom, and density 2 is the three-component normal mixture introduced in section 3.4.3 (Figure 3.5)[1].

Sample size. The sample size was $n = 25$. The original intent was to include sample size 100 in the design, but as discussed in the following section, optimization by SQP was prohibitively slow at the larger sample size.

Two hundred random samples of size 25 were generated from each of the true distributions. To avoid trivial cases where the unsharpened estimate is already unimodal, a rejection step was included when generating samples. Any sample producing a unimodal estimate (using the normal-scale bandwidth selector) was replaced with a new sample until a multimodal estimate was obtained.

[1] These two densities correspond to test densities 2 and 4 from Hall and Kang (2005).

For each of the data vectors, sharpening was performed using two different bandwidths, three different optimization methods, and up to four different objective functions.

Bandwidth. Let h_N be the bandwidth selected by the normal-scale rule for a particular data set. Optimization was performed using bandwidths h_N and $0.5h_N$.

Optimization method. The three methods were:

Greedy. The greedy algorithm, using the `improve` function.

SQP. Sequential quadratic programming using the unsharpened data $\mathbf{x}$ as a starting value, with outliers corrected[2].

Combined. SQP using the greedy solution as its starting point.

Objective function. The RC, RC^s, TV, and LIK objectives were used.

For each $\mathbf{x}$, sharpening was done for the 18 combinations of these three factors shown in Table 4.1, and the sharpened data set was recorded along with the run time for later analysis. The greedy method was not run with all four objective functions, since the objective function will play no role in the search when `improve` is started from a naïve guess.

The simulation consisted of 7200 optimizations (200 samples from each of two distributions, with 18 optimizations per sample). In the analysis to follow, appropriate subsets of the simulation runs will be used to answer questions of interest.

Remark 1. Sequential quadratic programming was implemented using the `fmincon` function in version 3.1.1 of the MATLAB optimization toolbox. SQP requires the mode to be pre-specified, so the optimizer was run at 20 candidate mode points evenly spaced between the first and third quartiles of the data. The solution with the best objective function value was returned. The constraint was enforced at 100 evenly-spaced points covering the range of the data.

Remark 2. It was originally intended to include an annealing algorithm such as those in Braun and Hall (2001) and Hall and Kang (2005) as a third optimizer, but attempts to produce a sufficiently automatic version of the algorithm were not successful[3].

[2] Any observations with distance from the median more than 1.5 times the interquartile range (IQR) were shifted so that they were exactly 1.5IQR away from the median.

[3] The references cited did not provide automatic rules for setting the tuning constants, and the present attempt was not successful in developing such rules. The annealing approach can work, but requires problem-specific adjustments.

Table 4.1. List of Optimizations for Each $\mathbf{x}$.

Method	Objective	Bandwidth
greedy	RC	h_N
greedy	RC	$0.5h_N$
SQP	RC	h_N
SQP	RC^s	h_N
SQP	TV	h_N
SQP	LIK	h_N
SQP	RC	$0.5h_N$
SQP	RC^s	$0.5h_N$
SQP	TV	$0.5h_N$
SQP	LIK	$0.5h_N$
combined	RC	h_N
combined	RC^s	h_N
combined	TV	h_N
combined	LIK	h_N
combined	RC	$0.5h_N$
combined	RC^s	$0.5h_N$
combined	TV	$0.5h_N$
combined	LIK	$0.5h_N$

4.2 Convergence, Ease of Use, and Run time

The greedy algorithm, by design, has guaranteed convergence[4], and requires minimal tuning regardless of problem type or starting point. In contrast, getting SQP to work reliably across simulation runs proved to be problematic. It is worthwhile to discuss SQP implementation challenges before reviewing the simulation results proper.

To be useful and fair in a simulation, the SQP code should perform as well as possible, run quickly, and require no case-specific user intervention. These goals naturally conflict to some extent. Reconciling them required judicious choices of starting solutions and iteration limits.

Performance of SQP was very sensitive to the starting solution. The algorithm would often fail to converge when using the unsharpened data as the initial solution. Other attempted starting points (all points at the unsharpened mode, evenly-spaced points, etc.) also performed poorly.

Eventually it was observed that convergence problems were often associated with outliers. The best way to promote convergence was to use $\mathbf{x}$ as the starting solution, but with outlying observations shifted toward the main body of points. This was achieved either by

[4]The term "convergence" here is used in the loosest sense, meaning that the algorithm reached its pre-specified stop conditions with a feasible solution.

simply shifting the outliers inward, or by using the greedy solution as a starting point.

When SQP did converge, run times were usually reasonable (a few minutes or less); conversely, when it failed to converge, run times could be unacceptably long (hours). It was ultimately discovered that this problem was due to an inability to find feasible solutions.

As its name implies, SQP involves the solution of a series of quadratic programming (QP) subproblems. At each subproblem, there is a search for a feasible solution. When no such solution can be found, the search will run the maximum allowed time, only to fail at the end.

To alleviate this problem, the QP subroutine was modified to reduce the maximum duration of search for a feasible solution. This change had little effect on successful runs, which usually found feasible points quickly anyway, but it made failed runs shorter. A sample size of 100 was still not feasible in the simulation, however. Even after the change to the QP subroutine, individual runs could take one or two hours to complete. The only way to further reduce run time is to put more stringent limits on the number of function evaluations in the search, but this results in premature termination of the algorithm and correspondingly poor solutions.

With the above practical difficulties noted, consider the simulation results. The proportion of converged solutions and the median run times observed in the simulation are shown in Tables 4.2 and 4.3. Median run time is used rather than average because the distribution of run time was considerably right-skewed for all cases. The main observations from these tables are given below in point form.

- The greedy method always gives a feasible solution, thus its convergence proportion is always one.
- Greedy search also runs much more quickly than SQP in all cases. Median run times were less than 0.5 seconds for `improve`, and range from several seconds to about one and a half minutes for SQP.
- In terms of convergence, the t_3 problem is harder than the mixture density problem, and problems with smaller bandwidth are harder than those with larger bandwidth. This is related to the prevalence of outliers and the difficulty finding feasible solutions. For the t_3 problem with bandwidth $0.5h_N$, the majority of SQP attempts failed to converge.
- As measured by run time, problem difficulty for a chosen objective function is dominated by bandwidth. Run time differences between different objective functions are

Table 4.2. Proportion of 200 runs converging.

Problem	Bandwidth	Objective	Greedy	SQP	Combined
t_3	$0.5h_N$	RC	1	0.25	0.23
t_3	$0.5h_N$	RC^s	—	0.22	0.24
t_3	$0.5h_N$	TV	—	0.81	0.72
t_3	$0.5h_N$	LIK	—	0.17	0.24
t_3	h_N	RC	1	0.89	0.90
t_3	h_N	RC^s	—	0.91	0.91
t_3	h_N	TV	—	1	0.99
t_3	h_N	LIK	—	0.88	0.90
mixture	$0.5h_N$	RC	1	0.90	0.91
mixture	$0.5h_N$	RC^s	—	0.88	0.90
mixture	$0.5h_N$	TV	—	0.95	0.98
mixture	$0.5h_N$	LIK	—	0.83	0.89
mixture	h_N	RC	1	1	1
mixture	h_N	RC^s	—	1	1
mixture	h_N	TV	—	1	1
mixture	h_N	LIK	—	1	1

hard to compare because the objectives themselves have different computational costs (TV and LIK being much more expensive to evaluate than RC).

- The proportion of converged runs is almost the same for SQP and for the combined method. Using the greedy solution as the starting point in SQP does not seem to improve the chance of converging.

- Supplying the greedy solution as the start value in SQP does reduce run time, however. Average run time was lower for the combined method in almost all cases.

- SQP was most likely to converge when using the TV objective. It is not clear why this is true. Total variation runs also took the longest time, because of the computational cost involved with calculating $TV(\mathbf{y}, \mathbf{x})$.

4.3 Solution Quality

In the preceding results, the greedy method shows several advantages over SQP. It always gives a useable result, it does so quickly, and it is insensitive to the details of the problem or the data. Not surprisingly, these advantages come with the trade-off that the solutions are usually inferior to the results of a converged SQP run.

Table 4.4 compares the performance of the greedy algorithm to that of SQP, using the RC objective as a basis for comparison. The table considers only runs for which SQP

Table 4.3. Median run time in seconds (converged runs only).

Problem	Bandwidth	Objective	Greedy	SQP	Combined
t_3	$0.5h_N$	RC	0.40	19	9
t_3	$0.5h_N$	RC^s	–	21	11
t_3	$0.5h_N$	TV	–	83	76
t_3	$0.5h_N$	LIK	–	75	52
t_3	h_N	RC	0.12	5	4
t_3	h_N	RC^s	–	12	4
t_3	h_N	TV	–	59	63
t_3	h_N	LIK	–	38	20
mixture	$0.5h_N$	RC	0.46	32	16
mixture	$0.5h_N$	RC^s	–	29	13
mixture	$0.5h_N$	TV	–	86	80
mixture	$0.5h_N$	LIK	–	57	47
mixture	h_N	RC	0.11	5	4
mixture	h_N	RC^s	–	7	5
mixture	h_N	TV	–	55	57
mixture	h_N	LIK	–	21	19

converged. It shows both the median $RC(\mathbf{y}, \mathbf{x})$ value and the proportion of times each method found the better solution. As with run time, the median is shown rather than the average because the distributions of objective values were right-skewed.

The table makes it clear that SQP is generally preferable to greedy—assuming SQP is able to converge. The median objective values were considerably smaller for SQP, and the SQP solution was better than the greedy solution more than 90% of the time for each test case. Nevertheless, the fact that SQP is not *always* better than `improve` underscores the point that convergence of SQP does not imply the attainment of a global minimum.

Figure 4.1 shows sharpening results from a random sample of runs with the RC objective function. Each graph in the figure shows the unsharpened estimate, the SQP-sharpened estimate, and the greedy-sharpened estimate. The difference between greedy and SQP solutions in the plots is not great in a qualitative sense. When the unsharpened estimate is nearly unimodal (cases 3, 5, and 8 in the figure), the two methods agree closely. If the unsharpened data has more strongly separated modes or more outliers (cases 1, 2, 6, 7, and 9), the SQP estimate tends to reach somewhat farther into the tails, while the greedy solution may retain higher fidelity near the unsharpened mode. Case 4 shows a result very similar to the small scale example of Figure 2.5: the unsharpened estimate is bimodal and nearly symmetric, and the two competing sharpened estimates are nearly mirror images of one another.

While it does not outperform SQP on its own, one can make use of the greedy method to further improve SQP performance. Table 4.5 compares the performance of the combined

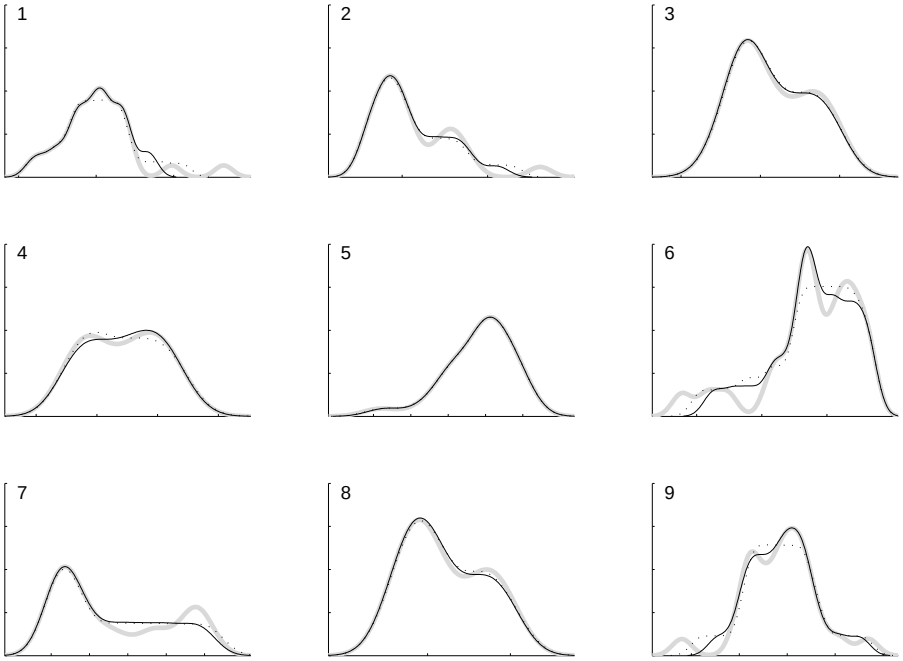

Figure 4.1: Comparing unsharpened estimate (thick grey line), greedy estimate (thin solid line), and SQP estimate (thin dashed line), for nine cases selected at random.

Table 4.4. Performance of greedy and SQP algorithms.

		Median $RC(\mathbf{y}, \mathbf{x})$		Proportion better	
Problem	Bandwidth	Greedy	SQP	Greedy	SQP
t_3	$0.5h_N$	2.3	0.66	0.08	0.92
t_3	h_N	0.71	0.33	0.04	0.96
mixture	$0.5h_N$	1.8	0.89	0.06	0.94
mixture	h_N	0.30	0.10	0.005	0.995

Table 4.5. Performance of SQP and combined algorithms (converged runs only).

			Median objective		Proportion better	
Problem	Bandwidth	Objective	SQP	Combined	SQP	Combined
t_3	$0.5h_N$	RC	0.66	0.56	0.41	0.59
t_3	$0.5h_N$	RC^s	0.53	0.58	0.36	0.64
t_3	$0.5h_N$	TV	0.10	0.07	0.27	0.73
t_3	$0.5h_N$	LIK	43.6	46.0	0.65	0.35
t_3	h_N	RC	0.33	0.33	0.38	0.62
t_3	h_N	RC^s	0.34	0.34	0.25	0.75
t_3	h_N	TV	0.042	0.033	0.07	0.93
t_3	h_N	LIK	42.3	42.2	0.43	0.57
mixture	$0.5h_N$	RC	0.89	0.87	0.37	0.63
mixture	$0.5h_N$	RC^s	0.87	0.87	0.39	0.61
mixture	$0.5h_N$	TV	0.078	0.057	0.29	0.71
mixture	$0.5h_N$	LIK	53.3	53.0	0.35	0.65
mixture	h_N	RC	0.098	0.098	0.39	0.61
mixture	h_N	RC^s	0.14	0.12	0.35	0.65
mixture	h_N	TV	0.020	0.010	0.22	0.79
mixture	h_N	LIK	54.8	54.8	0.41	0.59

method (SQP with greedy start) to the standard SQP method (using $\mathbf{x}$ with outliers shifted). In all but two cases, the median objective value was either the same or lower for the combined method than for SQP with default starting value. In all but one case, the combined method achieved a better solution than SQP in the majority of replications.

4.4 Comparison of Different Objective Functions

A final topic of interest in the simulation is comparing the characteristics of the different objective functions (using SQP as the optimizer). It has already been seen that the objectives differ in terms of computation time, and that TV has an advantage in terms of promoting convergence, but it would be beneficial to have a comparison of the statistical properties of the estimators arising from each objective.

Table 4.6 shows the estimated MISE for each objective function for each of the four combinations of problem and bandwidth. The MISE calculation excluded non-converged runs. There appears to be no major or systematic difference among the estimates.

Table 4.6. MISE estimates for each objective function.

Problem	Bandwidth	RC	RC^s	TV	LIK
t_3	$0.5h_N$	0.022	0.022	0.024	0.033
t_3	h_N	0.015	0.016	0.016	0.021
mixture	$0.5h_N$	0.079	0.080	0.080	0.079
mixture	h_N	0.080	0.080	0.080	0.069

Another means of comparison is a plot of MSE versus position for each problem/bandwidth combination. These plots (not shown here) did not reveal large or consistent differences between objectives. In particular, the MSE differences in the tails were not large enough to have practical significance.

The preliminary investigation of different objectives here is not deep enough to be conclusive. Improved study of the different objectives would be easier after the optimization step has been sufficiently improved. At that point it would be more appropriate to design a systematic study of the effect of these objective functions.

Chapter 5

Discussion

Data sharpening has the potential to become a general approach to constraint handling in nonparametric estimation. One of the major barriers to the realization of this potential is optimization. The optimization difficulties have been explained at length in the preceding chapters, using unimodal kernel density estimation as a representative problem.

The greedy algorithm developed in Chapter 3, implemented in the `improve` function, is a tool to help improve data sharpening optimization. The following sections review the characteristics of the greedy method, discuss how it might be used, and describe the next stages in this line of research.

5.1 Advantages and Disadvantages of the Greedy Algorithm

The greedy method has several desirable properties. It is fast; it is guaranteed to produce a feasible solution; its only adjustable parameter is a tolerance, to which the end result is not sensitive; and the algorithm's design is not specific to a particular problem or constraint. As a result of these properties, the `improve` function is reliable and easy to use.

When applied to unimodal kernel density estimation with its default starting solution, `improve` tends to return solutions that match the unsharpened estimate at points away from constraint violations. Modifications to the data are typically restricted to the troughs and tails of the unsharpened density estimate (a desirable property, as discussed in Hall and Kang 2005).

The limitations of the greedy method follow directly from its design, and are typical of greedy algorithms in general. While a feasible solution is guaranteed, the solution will normally be sub-optimal; if other solution methods (like SQP) are able to converge, they will often perform better. In addition, the greedy method does not directly consider the

chosen objective function—it just makes the sharpened points move toward their targets at each pass through the data. This movement rule hard-wires the algorithm to distance measures like L_α and RC, for which the objective always decreases as $\mathbf{y}$ moves closer to $\mathbf{x}$. There is no natural way to adapt `improve` to objectives not having this property, such as TV and LIK.

5.2 Applications

The performance limitations of `improve` limit its applicability as a stand-alone data sharpening method. Nevertheless, its advantages suggest it could still be useful in at least three ways.

1. As a starting value for SQP. The simulations of the previous chapter suggest that using the greedy solution as the starting point for SQP improves both run time and solution quality. The greedy algorithm being much faster than SQP, this strategy provides a free gain for SQP. The fact that such a gain is possible, however, reflects the inadequacy of SQP as an optimizer in this context. Furthermore, the improved starting point does not improve the likelihood of convergence. This means that SQP still remains severely limited in sufficiently challenging sharpening problems.

2. As a repair method in a more advanced heuristic. One general strategy for constraint handling in heuristic optimization is to use a "repair" operator to convert infeasible solutions into feasible ones. A promising application of `improve` is as such an operator in a more sophisticated algorithm. If $\mathbf{y}_F$ and $\mathbf{y}_I$ are feasible and infeasible solutions, respectively, then $\mathbf{y}_R = \texttt{improve}(\mathbf{y}_F, \mathbf{y}_I)$ will return a feasible repaired solution. The repair is performed by treating $\mathbf{y}_I$ as the unsharpened target, and moving the initial solution $\mathbf{y}_F$ toward it.

3. As an optimizer in other constrained estimation scenarios. The greedy method (either by itself or as part of a larger heuristic) could ultimately be applied to problems beyond the unimodal KDE setting. The first such expansion could be to density estimation with other shape constraints, like k-modality, symmetry, or bell shape[1]. Another level of expansion could be to higher dimensions, starting with unimodal bivariate kernel density estimation. Finally, the methods begun here could be moved to different nonparametric estimators altogether, to tackle problems like shape-constrained regression.

[1]Constraining the density to be unimodal with only two inflection points would restrict second derivative changes and eliminate shoulders and waves in the density estimate.

5.3 Future Work

The possibilities for further research can be divided into three mutually-supporting categories: work toward applications, theoretical work, and work on the statistical properties of sharpened estimators.

Work Toward Applications

This branch of work is focused on further improvements in optimization, using the `improve` function as a part of a larger optimization heuristic. Existing classes of heuristic that could be suitable include genetic algorithms, particle swarm approaches, and simulated annealing[2].

As mentioned above, the greedy algorithm could be incorporated in one of these standard methods as a way to handle constraints. Given the importance of the constraints in data sharpening, however, it is possible that best results will be achieved by a built-to-suit hybrid of ideas from various heuristic approaches. Preliminary work to this end suggests that a promising approach is to maintain a population of solutions while repeatedly perturbing them, repairing them, and discarding the worst candidates.

Many application possibilities open up if a robust optimizer with good performance can be developed around the greedy algorithm. For example, the optimizer could be used to try unimodal estimation using adaptive kernels (optimize the bandwidths individually per point) or by adjusting points' weights (as in Racine and Parmeter 2008). This could potentially be done by setting the target to a uniform vector of h in the first case or $1/n$ in the second. Beyond this, the optimizer could be applied to a variety of constraints or estimators, as mentioned in the previous subsection.

Adaptation of the greedy method to higher dimensions is reasonably straightforward. The movement rules have natural multi-dimensional analogs. The primary sources of difficulty in a higher-dimensional greedy algorithm will be in the matching step and in constraint checking. Defining how unsharpened points should be matched to sharpened ones is much harder in higher dimensions than it is on the line. The problem of constraint checking is more computational: as dimension increases, constraint-checking over a grid of points becomes increasingly burdensome.

[2]Hall and Ooi (2004) use an annealing algorithm different from the one unsuccessfully attempted here. Their algorithm uses excess mass as a measure of distance from unimodality.

Theoretical Work

The heuristic nature of the present developments make it difficult to establish properties of the method theoretically. Still, some theoretical considerations would be useful to develop further in the future.

The computational complexity of the greedy algorithm (or its subsequent heuristics) is one area where it should be possible to make progress. It would be beneficial to be able to make statements about how the algorithm will scale with sample size, for example, without having to rely wholly on simulation studies.

Another area where theoretical insights would be welcome is the choice of objective function. Without input from theory, only extensive simulations can guide the choice of objective. For example, does the likelihood objective—which amounts to fitting a normal mixture to the data—have any advantageous properties? Or can any of the objectives be shown to encourage desirable tail behaviour in terms of MISE or MSE?

Statistical Properties

One very important area for future work is in the development of confidence bands for sharpened estimators. It is felt that resampling methods should be suitable to generate bands, but how exactly to do this has not been explored.

Another development that would be very welcome is a test of the validity of the constraint. This would be particularly valuable if it could be developed in a way that matched the generality of the data sharpening approach. The combination of data sharpening and a test for constraint validity would be a very powerful data analysis tool.

There are in addition to intervals and tests, other statistical considerations commonly addressed in kernel density estimation. These include bandwidth selection, asymptotic performance, and boundary issues. Bandwidth selection in particular can always be done by an empirical method such as cross-validation. As for other considerations, the hope thus far is that by using standard estimators and only perturbing the data, the favourable properties of the estimator will be retained.

Appendix A

Detailed Pseudocode

Table A.1. Pseudocode for the Improve Function—Bisection Approach.

Inputs:

v a feasible starting solution.
x the observed data, sorted in ascending order.
τ bisection tolerance.

Output:

y a new feasible solution, having each $y_i = v_i + \rho_i(x_i - v_i)$, with $\rho_i \in [0, 1]$.

Algorithm:

```
Perform initial sorting and ordering
Set y := v.
Sort y in ascending order.
Find the set of moveable points (M of them).
Set the sweep order: Let y_[j] be the moveable point that is j^th farthest from home.
Loop until no points are moveable
WHILE M > 0:
    Sweep through the moveable points
    FOR j = 1 to M:
        Set bounds for bisection: A := y_[j], B := x_[j].
        Store the best feasible solution: y_best := A + τ·sign(B − A).
        Use bisection to move the point
        WHILE |B − A| > τ:
            Set y_[j] := (A + B)/2.
            IF y is feasible
                A := y_[j]
                y_best := A
            ELSE
                B := y_[j]
            END IF
        END WHILE
        Set y_[j] := y_best
    END FOR
    Revise the sorting and ordering
    Sort y in ascending order, unless doing so repeats a previous ordering.
    Find the set of moveable points (M of them).
    Set the sweep order: Let y_[j] be the moveable point that is j^th farthest from home.
END WHILE
```

Table A.2. Pseudocode for the Improve Function—Grid Search Approach.

Inputs

v a feasible starting solution.
x the observed data, sorted in ascending order.
τ search tolerance.

Output

y a new feasible solution, having each $y_i = v_i + \rho_i(x_i - v_i)$, with $\rho_i \in [0, 1]$.

Algorithm

Set $\mathbf{y} := \mathbf{v}$.
Set the number of grid search steps, $S := 1$
Sort **y** in ascending order.
Find the set of moveable points (M of them).
Set the sweep order: Let $y_{[j]}$ be the moveable point that is j^{th} farthest from home.
Loop until no points are moveable
WHILE $M > 0$:
    Sweep through the moveable points
    FOR $j = 1$ to M:
        Set the bounds for search: $A := y_{[j]}$, $B := x_{[j]}$.
        Set the step size, $\Delta := \max(|B - A|/S, \tau)$.
        Reset the number of successful steps, $k := 0$.
        Use grid search to move the point
        WHILE **y** is feasible and $(k + 1)\Delta \leq |B - A|$:
            Set $y_{[j]} := A + \text{sign}(B - A) \cdot (k + 1)\Delta$.
            IF **y** is feasible:
                $k = k + 1$
            END IF
        END WHILE
        Set $y_{[j]} := A + \text{sign}(B - A) \cdot k\Delta$
    END FOR
    Revise the sorting and ordering, or adjust number of steps
    IF at least one point has moved:
        Sort **y** in ascending order, unless doing so repeats a previous ordering.
        Find the set of moveable points (M of them).
        Set the sweep order: Let $y_{[j]}$ be the moveable point that is j^{th} farthest from home.
    ELSE
        $S = 2S$
    END IF
END WHILE

Bibliography

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

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.

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

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

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

Hall, P. and Heckman, N. E. (2002), "Estimating and Depicting the Structure of a Distribution of Random Functions," *Biometrika*, 89, 145–158.

Hall, P. and Huang, L.-S. (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 Ooi, H. (2004), "Attributing a Probability to the Shape of a Probability Density," *The Annals of Statistics*, 32, 2098–2123.

Hillier, F. S. and Lieberman, G. J. (1980), *Introduction to Operations Research*, 3rd ed., Holden-Day.

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.

Racine, J. S. and Parmeter, C. F. (2008), "Constrained Nonparametric Kernel Regression: Estimation and Inference," Web.uvic.ca/econ/racine.pdf.

Silvapulle, M. J. and Sen, P. K. (2005), *Constrained Statistical Inference*, Wiley.

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