Journal of Statistical Software

April 2012, Volume 47, Issue 6. *http://www.jstatsoft.org/*

A Greedy Algorithm for Unimodal Kernel Density Estimation by Data Sharpening

Mark A. Wolters
University of Western Ontario

Abstract

We consider the problem of nonparametric density estimation where estimates are constrained to be unimodal. Though several methods have been proposed to achieve this end, each of them has its own drawbacks and none of them have readily-available computer codes. The approach of Braun and Hall (2001), where a kernel density estimator is modified by data sharpening, is one of the most promising options, but optimization difficulties make it hard to use in practice. This paper presents a new algorithm and MATLAB code for finding good unimodal density estimates under the Braun and Hall scheme. The algorithm uses a greedy, feasibility-preserving strategy to ensure that it always returns a unimodal solution. Compared to the incumbent method of optimization, the greedy method is easier to use, runs faster, and produces solutions of comparable quality. It can also be extended to the bivariate case.

Keywords: constrained nonparametric estimation, shape constraints, optimization heuristics, MATLAB.

1. Introduction

Nonparametric density estimators are the usual recourse when a researcher is not prepared to assume a specific functional form for a density to be estimated. The advantage of nonparametric estimators is their flexibility: based on minimal assumptions, they can take shapes that are highly data-driven. This shape flexibility can be problematic, however, in cases where one knows (or strongly suspects) that the true density is unimodal. Spurious peaks can appear in unlikely locations, such as in the tails of the density.

When one believes that a density is unimodal, there are two good reasons to add a unimodality constraint to the estimate. First, making use of the extra shape information about the density can be expected to improve estimation accuracy. Second, incorporating the constraint will

eliminate spurious modes that may reduce the effectiveness of the density estimate as an exploratory tool and communication aid.

The contribution of the present paper is an algorithm and code, in the MATLAB language (The MathWorks, Inc. 2007), which improves the computational performance and ease-of-use of a particular unimodal density estimator: the sharpened kernel density estimator of Braun and Hall (2001). A brief introduction to unimodal density estimation is presented below, with emphasis on the sharpened kernel density estimator. The new algorithm is then described in Section 2, followed by simulation results in Section 3 that measure the relative performance of the algorithm. Section 4 contains examples which demonstrate the use of the code in practice.

1.1. Unimodal density estimation

The class of unimodal densities includes monotone densities as a special case, for which the mode is located at either the left or right edge of the density's support. Early work by Grenander (1956) developed the nonparametric maximum likelihood estimator for the monotone case. For nonincreasing densities, it is the derivative of the least concave majorant of the empirical cumulative distribution function (ECDF); for nondecreasing densities, it is the derivative of the greatest convex minorant of the ECDF.

Later research attempted to extend the Grenander estimator to any unimodal density. The common premise was to combine a nondecreasing Grenander estimate to the left of the mode with a nonincreasing one to the right. The crucial problem is determining where to locate the mode. Wegman (1972) considered specifying a modal interval, Bickel and Fan (1996) plugged in a consistent point estimate of the mode location, and e (1997) chose the mode to minimize the distance between the estimate and the ECDF. Like the Grenander estimator itself, all of these methods produce a step-function estimate (though Bickel and Fan 1996 did propose methods of smoothing the density after estimation).

Alternative approaches to estimation have been proposed to produce a smooth unimodal density estimate directly. Fougeres (1997) uses a monotone rearrangement to transform a multimodal density into a unimodal one, though under the restrictive assumption that the mode location is known. Cheng, Gasser, and Hall (1999) start with a unimodal "template" density and then iteratively apply monotone transformations (possibly with intermediate smoothing steps) to construct an estimate.

1.2. The sharpened kernel density estimator

All of the aforementioned approaches require unconventional estimators to handle the unimodality constraint. It would be preferable if unimodality could be achieved by adding a conceptually simple modification to a standard nonparametric estimator. Data sharpening, as advanced by Braun and Hall (2001), is one approach that operates in this way.

The premise of data sharpening is that the characteristics of an estimator can be improved by shifting the data points in a controlled way prior to estimation. That is, if the original *unsharpened* data vector is $\mathbf{x} = [x_1 \cdots x_n]^\top$, we simply replace it by a new *sharpened* data vector $\mathbf{y}$ before using our estimator of choice. The critical matter is the selection of $\mathbf{y}$—it must be chosen to endow the estimator with the desired properties (in this case, unimodality), while maintaining maximum fidelity to the observed $\mathbf{x}$.

The attractive feature of data sharpening is its potential generality: it can in principle be

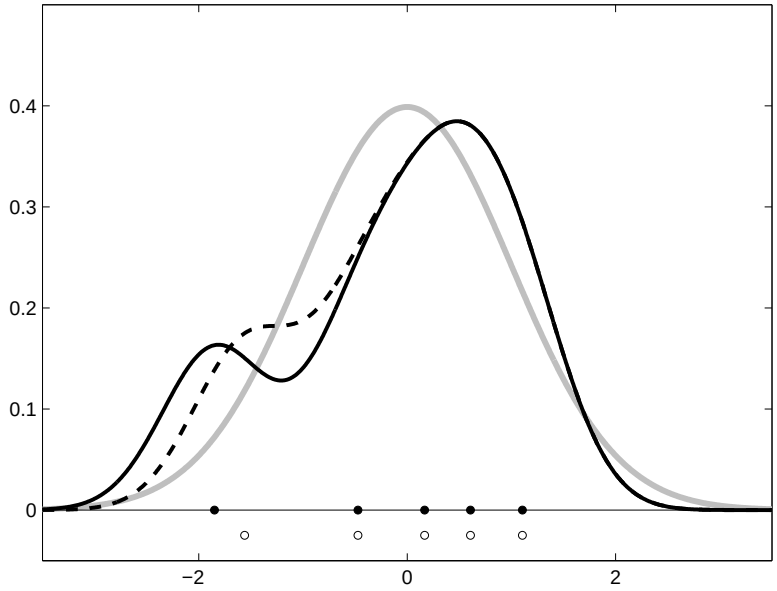

Figure 1: A small example illustrating the premise of data sharpening. The grey curve is the true distribution, the solid curve is the KDE based on the original data (filled circles), and the dashed curve is the KDE based on the sharpened data (open circles).

applied to any nonparametric estimator, with any shape constraint, and even in higher dimensions. Using data sharpening, Braun and Hall (2001) showed that enforcing constraints can improve the mean squared error performance of a number of estimators, including unimodal density estimators.

The present work follows Braun and Hall (2001) in applying data sharpening to a kernel density estimator (KDE) to obtain unimodal estimates (see, e.g., Wand and Jones 1995 for background on standard kernel density estimation). The problem may be set up as follows. Let the sharpened KDE be

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

where $K_h(\cdot)$ is a symmetric, zero-centered kernel function (usually a density function) with scale parameter h. The parameter h is called the bandwidth of the KDE, and it is assumed throughout this work that the bandwidth is given (though see the remark below, and the Appendix). Estimator (1) is algebraically the same as the standard kernel density estimator; only a subscript has been added to $\hat{f}$, indicating which data vector is used to produce the estimate. The usual KDE is $\hat{f}_{\mathbf{x}}$, that is, it arises when $\mathbf{y} = \mathbf{x}$.

The sharpened data are chosen such that they produce a unimodal estimate while being as close as possible to $\mathbf{x}$. Figure 1 illustrates the idea through a small example. The data are a sample of size five from a standard normal distribution. A standard KDE with Gaussian kernel and bandwidth $h = 0.5$ produces a bimodal estimate. In this case the sharpened estimate can be constructed by shifting only one of the five points, and this point is shifted as little as possible to render the estimate unimodal.

A measure of the closeness of $\mathbf{y}$ and $\mathbf{x}$ is required to define what is the best data sharpening

estimator. A natural choice, used by Braun and Hall (2001) and Hall and Kang (2005), is to base the measurement on a norm of the difference $\mathbf{y} - \mathbf{x}$, defining

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

Using the L_α distance for some value of α, the best sharpened data vector $\mathbf{y}^*$ can be defined as a solution to a constrained optimization problem:

$$\mathbf{y}^* = \operatorname*{argmin}_{\mathbf{y} \in \mathcal{C}} \; L_\alpha(\mathbf{y}, \mathbf{x}), \tag{3}$$

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

$$\mathcal{C} = \left\{ \mathbf{y} : \exists m \;\text{ such that } \begin{array}{ll} \hat{f}'_{\mathbf{y}}(x) \geq 0 & \text{if } x \leq m \\ \hat{f}'_{\mathbf{y}}(x) \leq 0 & \text{if } x \geq m \end{array} \right\}. \tag{4}$$

The constraint set $\mathcal{C}$ formalizes the definition of unimodality: there must be some m, the location of the mode, for which the density estimate is nondecreasing to the left and nonincreasing to the right.

The remainder of the article focuses on ways of finding good solutions to problem (3).

Remark on bandwidth selection

The simplest way to choose h in data sharpening is to proceed as if no sharpening will be done. Apply a standard bandwidth selector (such as direct-plug-in) to the original data $\mathbf{x}$, and then hold its value constant when determining $\mathbf{y}$. When the unimodality assumption is valid, this strategy can be justified asymptotically: as n increases, the sharpened estimator modifies the KDE only in the ever-smaller regions where the constraint is violated (Hall and Kang 2005). For finite samples, empirical results (Braun and Hall 2001; Hall and Kang 2005) suggest that sharpened estimators do have different optimal bandwidths than their unsharpened counterparts, but also that sharpened estimators are less sensitive to bandwidth choice than unsharpened ones. For these reasons the simple approach to bandwidth selection is used here. An alternative method is proposed in the Appendix, but a thorough study of optimal bandwidth selection for shape-constrained estimators remains an area for future research.

1.3. Optimization considerations

The incumbent method of optimization, to which the new greedy algorithm will be compared, is sequential quadratic programming (SQP; see Nocedal and Wright 1999 for a thorough discussion). SQP has been used by Braun and Hall (2001) and Hall and Kang (2005) to find sharpened KDEs, and by others (Hall and Huang 2002; Racine and Parmeter 2008) to perform other forms of constrained nonparametric estimation.

One inconvenient feature of SQP is its poor suitability when the mode m is unknown. As with other mathematical programming methods, SQP's standard formulation requires the problem's constraints to be expressed as a set of fixed inequalities in $\mathbf{y}$. This cannot be done with the problem as written in (3) and (4), because m 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. Taking m as fixed, then, and enforcing the constraints at a grid of points $\mathbf{g} = (g_1, \ldots, g_G)$ covering the support of the estimate, the optimization problem is a search for the value

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmin}} \; L_\alpha(\mathbf{x}, \mathbf{y}) \quad \text{subject to} \quad \begin{cases} \hat{f}'_{\mathbf{y}}(g_i) \geq 0, & i = 1, \ldots, k \\ \hat{f}'_{\mathbf{y}}(g_i) \leq 0, & i = k+1, \ldots, G, \end{cases} \tag{5}$$

where m falls between gridpoints k and $k+1$.

Solving a sequence of problems like (5) for different values of m makes it possible to use SQP to find the sharpened estimate. Note, however, that finding the best m is not trivial, since the optimal L_α value is not necessarily a convex function of m. An exhaustive grid search over some interval is a reasonable approach to find a good m value, but this comes with a considerable computational cost.

A second problem associated with SQP arises when setting $\alpha = 1$ in the L_α objective function. Braun and Hall (2001) and Hall and Kang (2005) found some evidence that the sharpening estimator performs best at or near $\alpha = 1$, but at this setting the objective function is not differentiable in its i-th dimension at the point $x_i = y_i$. The L_1 norm is also not strictly convex, so solutions might not be unique. These features of the objective can cause numerical problems that prevent the algorithm from converging.

The most serious problem when using SQP to find sharpened KDEs, however, is the non-convexity of the constraints. For SQP to find a global optimum, the constraint functions $\hat{f}'_{\mathbf{y}}$ in (5) should be convex functions of $\mathbf{y}$; in reality, they are not. The constraint set $\mathcal{C}$ is not convex, and may not even be a contiguous set (for an illustration of this, see Wolters 2009). The result is a problem with many local optima. In the best case, SQP will converge to one of these local optima in the region near its starting point; in the worst case it will fail to converge, returning an error or a nonsensical solution.

The end result of these problems is that SQP is difficult to implement reliably in practice. Experience has shown that solution quality and the chance of non-convergence depends not only on the chosen value of α, but also on the magnitude of departures from unimodality in the sample, and on the quality of the initial solution provided to the algorithm. As an example, we will see in the simulation study of Section 3 that failure to converge can occur more than 10% of the time.

Faced with a problem of this nature, it is prudent to consider a heuristic optimization approach that may be able to find better solutions. The greedy heuristic discussed in the next section has some advantages: it never fails to return a unimodal solution, it does not need the mode location to be pre-specified, and it runs more quickly than SQP. Like SQP, the results of the greedy algorithm depend on its starting point; but there is a natural choice of starting value that gives good solutions in practice.

Remark on the use of sub-optimal solutions

It is worthwhile to underscore at this point that neither SQP nor the new algorithm can guarantee that they will find a global optimum of problem (3) or (5). The current work is based on the belief that i) an easy-to-use nonparametric unimodal density estimator is of value for practical purposes even if global optimality cannot be proven, and ii) the simplicity and

potential generality of data sharpening justifies the search for a suitable optimization routine, despite the difficulty of the problem. The convergent nature of mathematical programming methods should not be misconstrued to imply that these methods tend to find superior solutions. If a heuristic method can find a solution with a better objective function value, then that solution is to be preferred even if the method is not convergent in the classical sense.

2. The algorithm

The greedy data sharpening algorithm is described below in three stages. The new algorithm is first described at a high level to identify its main features. Afterwards, computational details related to the computer implementation are described. Extension of the method to bivariate data is discussed at the end of the section.

2.1. High-level description

The steps in the proposed algorithm are given in the flowchart of Figure 2. The algorithm starts with a user-supplied initial guess solution, $\mathbf{v}$, that is feasible, and improves this guess by moving its points closer to the original data. From the initial solution $\mathbf{y} = \mathbf{v}$, each sharpened data point y_i is moved to be closer to its corresponding unsharpened data point x_i. Every such move will reduce the objective function $L_\alpha(\mathbf{y}, \mathbf{x})$, but moves are only made if the constraint is not violated. No point may be moved in a way that causes the density estimate to gain a second mode, and so feasibility is guaranteed throughout. The algorithm cycles through the points repeatedly as long as improvements can still be made without violating the constraint. The algorithm is greedy in the sense that each point is moved individually to improve the objective function, without consideration of how the current move will impact future moves of other points.

Step one in the search is initialization. The original data $\mathbf{x}$ is sorted in ascending order, and the solution is initialized to $\mathbf{y} = \mathbf{v}$. The initial solution may be a simplistic choice, but it must satisfy the constraint. The default starting point is to put all of the data points at the location of the highest mode in the unconstrained estimate (in other words, if m_0 is the location of this mode, we set $\mathbf{v} = m_0\mathbf{1}$, where $\mathbf{1}$ is an n-vector of ones). This starting point has been found to provide good solutions in most circumstances.

The second step is to prepare for moving the $\mathbf{y}$ points. Each x_i, $i = 1, \ldots, n$ is called the *home* or *target* position for the corresponding y_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*.

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

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

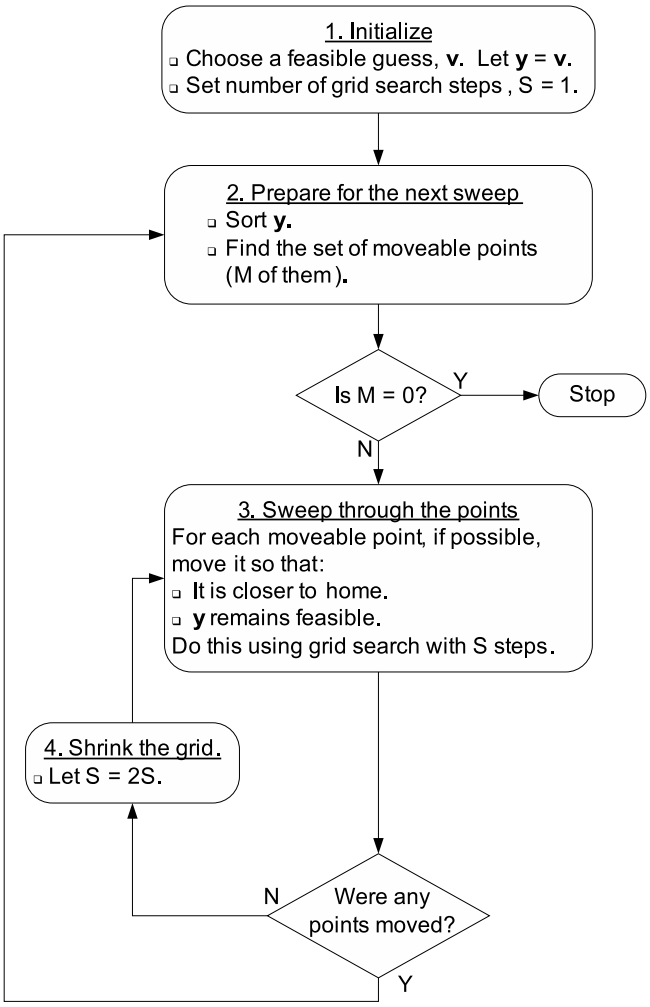

Figure 2: A high-level depiction of the greedy algorithm.

y and re-counting the number of moveable points. If a complete sweep results in no moved points, the value of S is doubled before the next pass, permitting smaller moves to be made on a finer grid.

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

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

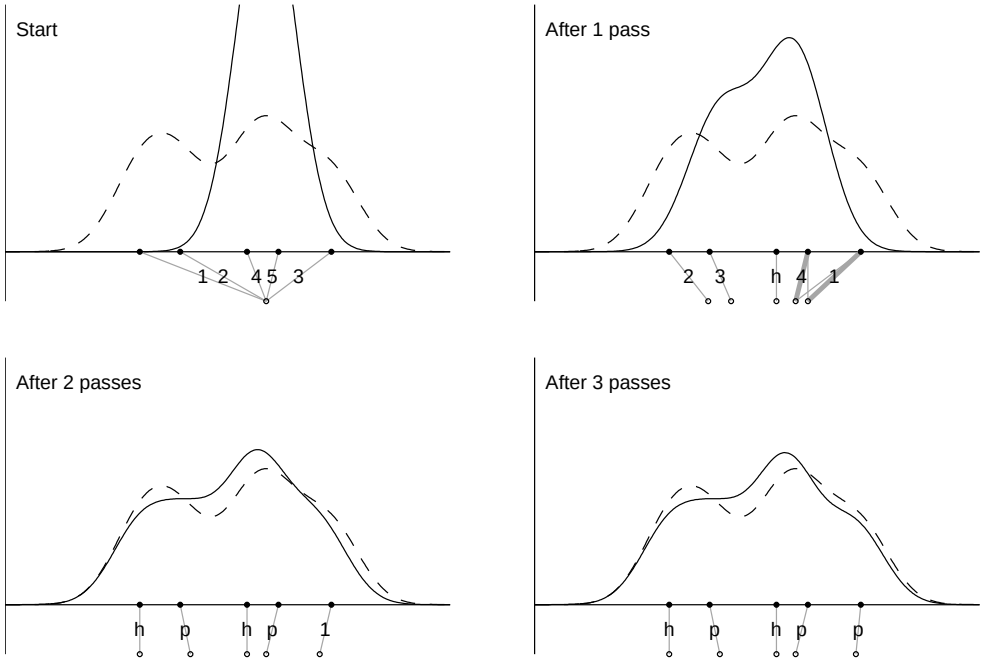

Figure 3: A small example illustrating the greedy sharpening method. Solid/dashed lines show the sharpened/unsharpened estimates. Open/filled circles show the sharpened/unsharpened data. Grey lines join each unsharpened point to its target and indicate the status of the point.

The ideas of this section are illustrated in Figure 3, which shows how the solution develops over three passes for a small example with only five data points. The intermediate positions of the sharpened points are shown after each pass, and a line joins each point to its target. Each line is labeled to show the status of its corresponding sharpened point. Lines labeled with numbers correspond to moveable points, and the numbers indicate the order in which points are to be moved. Lines labeled with h correspond to points at home, while those labeled with p correspond to points that are pinned. After the first pass (the upper right plot in the figure), the sorting step has caused two points to switch targets. The thick grey lines indicate the points' new targets after re-matching. In this example the search terminated after three passes, with three points pinned and two points at home.

2.2. Implementation details

The steps just described omitted several details that are important when implementing the algorithm on a computer. These details will be briefly discussed here.

Initial solution

Just like SQP, the greedy algorithm is sensitive to its starting point $\mathbf{v}$. Supplying an inappropriate starting value will result in premature termination of the algorithm at a poor solution. The recommended initial value (all points at the unsharpened mode) is pragmatic because it typically allows many points to move directly to their home position in the early stages of

the search, with the estimate slowly moving outward toward the tails as search progresses. Nevertheless, to reduce the risk of initialization dependence, one could try multiple starting points and keep only the best solution found—perhaps by using $\mathbf{v} = c\mathbf{1}$ and letting c vary over the range of $\mathbf{x}$. Such possibilities are not considered here because of the good general performance of the default starting choice.

Sweep order

Each sweep through the data is done in descending order of distance from home, i.e., the points with the greatest value of $|y_i - x_i|$ are moved first. If all points are started near the center of the distribution, this has the effect of moving points toward the tails first, and then moving interior points that are closer to home. Experience has shown that this sweep order provides improved performance and speed.

Feasibility checking

Frequent feasibility checks (verifying that $\mathbf{y} \in \mathcal{C}$) account for most of the computational burden of the method. A practical means of evaluating whether a given solution produces a unimodal estimate is to calculate exact or approximate values of $\hat{f}'_{\mathbf{y}}$ at a grid of points $\mathbf{g}$ covering the support of the estimate, and then to count the number of times $\hat{f}'_{\mathbf{y}}(g_i)$ changes sign as g_i increases. A monotone estimate will have no sign changes, a peaked unimodal estimate will have one, and a multimodal estimate will have more than one. The values of $\hat{f}'_{\mathbf{y}}(g_i)$ are calculated using a binned kernel density estimator approximation (Wand and Jones 1995, Appendix D.2).

Determining the status of each point

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

$$y_i \text{ is home} \quad \Leftrightarrow \quad |x_i - y_i| \leq \tau \tag{6}$$

$$y_i \text{ is pinned} \quad \Leftrightarrow \quad \text{setting } \; y_i := y_i + \text{sign}(x_i - y_i)\tau \; \text{ renders } \mathbf{y} \text{ infeasible} \tag{7}$$

$$y_i \text{ is moveable} \quad \Leftrightarrow \quad \begin{cases} |x_i - y_i| > \tau, \quad \text{and} \\ \mathbf{y} \text{ remains feasible when } \; y_i := y_i + \text{sign}(x_i - y_i)\tau. \end{cases} \tag{8}$$

Statements (6) through (8) mean, respectively, that a point is home when it is within τ of its target; it is pinned when a move of size τ toward home causes a constraint violation; and it is moveable when it is neither pinned nor at home.

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

Design of the grid search

The goal of each grid search step is to move the current point y_i closer to its target x_i without violating the constraint. There are S candidate points along the interval $[y_i, x_i]$. Rather than

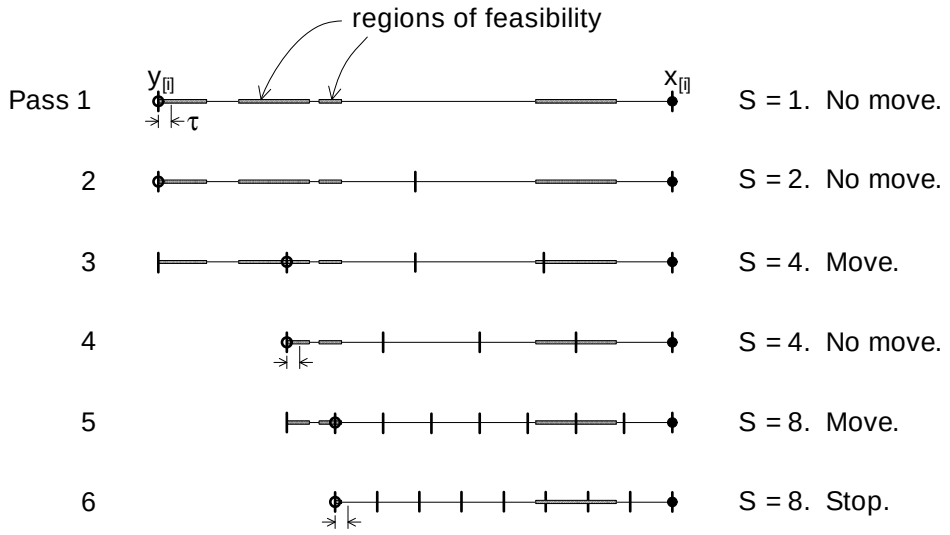

Figure 4: A schematic illustration of the grid search for a single point.

searching all S steps, the grid search is conducted by stepping out from y_i along the grid, until feasibility is lost. After this the last feasible point is chosen. This procedure will not always find the feasible grid point closest to x_i, but it makes the overall search much more efficient by eliminating many needless constraint checks.

The nature of this shrinking-grid search is illustrated in Figure 4 by looking at a single point over six passes. For each pass, the interval $[y_i, x_i]$ is shown with a grid of S steps superimposed. When the point is moveable, but cannot step out along the grid, the grid is made more fine by doubling S. As long as the point remains moveable, each pass results in either a successful step out or a doubling of the grid. Search terminates when the point ceases to be moveable. Two facts are not clearly depicted in the figure: i) the grid steps are doubled only when none of the n points move; and ii) the feasible region or target x might change between passes, since they can be changed by the movement of other points.

Sorting memory

On rare occasions the configuration of the points could lead to cyclic behavior caused by the sorting step. For example, the two moveable points could exchange targets repeatedly and thereby never reach a pinned state. To prevent this, a list of all previous orderings is kept in memory, and new orderings are only accepted if they have not been visited previously. The memory requirements for this control are not problematic, since the number of passes used is typically small (on the order of 100 passes for moderate-sized data sets).

2.3. The bivariate case

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

The only aspect of the greedy algorithm that does not translate directly from univariate to

bivariate problems is the sorting step that occurs between passes through the data (in step 2 of Figure 2). In higher dimensions it is not clear how to choose the best matching of sharpened to unsharpened points. The points cannot be "sorted," since a total ordering property can no longer be exploited.

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

An example of the greedy algorithm applied to a bivariate data set is given in Section 4.3.

3. Simulation results

A simulation study was performed to compare the performance of three optimization options: (i) the greedy algorithm, (ii) SQP using NAG routine `e04wd` (Numerical Algorithms Group 2009), and (iii) a combined optimizer, where the greedy solution is used as the starting point for an SQP search.

3.1. Study design

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

- Target densities: the t distribution with three degrees of freedom, and a three-component normal mixture distribution. See Figure 5.
- Sample sizes: 25, 50, and 100.
- Bandwidths: $0.75h_{SJ}$ and h_{SJ}, where h_{SJ} is the Sheather-Jones direct plug-in bandwidth (Sheather and Jones 1991).

The target densities correspond to test densities 2 and 4 of Hall and Huang (2002). The t_3 distribution is challenging because its heavy tails result in outliers, and corresponding spurious modes, in many samples. The mixture distribution has a large, nearly flat shoulder to the right of its mode, which produces a variety of multimodal shapes in samples. The two bandwidth levels were chosen to influence the problem difficulty rather than to achieve optimal estimation performance. Setting $h = h_{SJ}$ produces very smooth estimates that are easier to sharpen, while $h = 0.75h_{SJ}$ produces more separated peaks and reduces the size of the feasible set $\mathcal{C}$, making optimization harder.

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

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

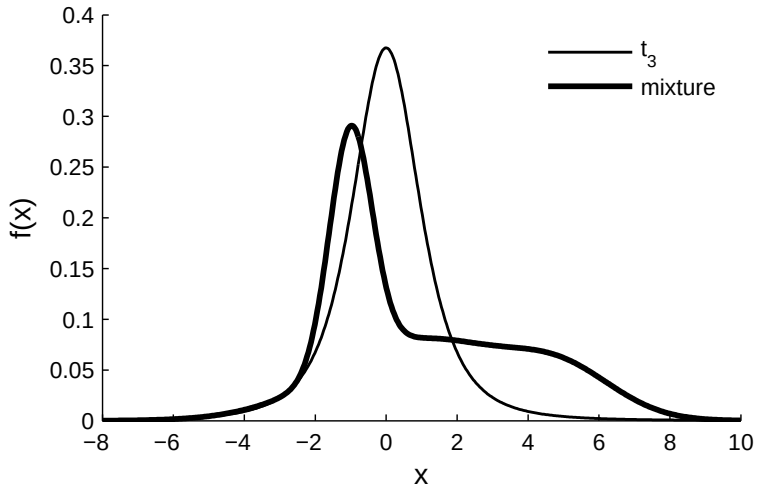

Figure 5: The simulation target densities. The mixture density is composed of $N(-1, 0.6^2)$, $N(1, 2.5^2)$, and $N(5, 1.5^2)$ components in proportions 0.35, 0.5, 0.15.

The use of SQP required certain implementation decisions. First, because SQP requires a fixed mode value, the optimizer was run 20 times with different mode locations, evenly spaced between the first and third quartiles of the data. The best of these 20 solutions was returned as the final result. Second, the unsharpened data, $\mathbf{x}$, was used as the starting point for the SQP search. Finally, the $L_1(\mathbf{y}, \mathbf{x})$ distance caused problems for the NAG routine when used as the objective, so the the following twice-differentiable piecewise function, termed the rounded-corners objective, was used:

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

where $d_i = |y_i - x_i|$ and $\mathbf{I}$ is the indicator function. The summand of (9) is a convex cubic polynomial in the interval $|y_i - x_i| \leq \gamma$ and a line with unit slope (just like L_1) outside this interval. The central interval is effectively a curved, differentiable patch that replaces the corner in the usual L_1 objective. The constant γ determines the width of this interval; smaller values of γ more closely approximate L_1.

In the simulation's SQP runs, the $RC_{0.01}$ function was used as the objective. By setting a small value $\gamma = 0.01$, behaviour very similar to L_1 was obtained with significantly improved numerical stability. Because the $RC_{0.01}$ and L_1 curves are so similar, the terms "L_1 distance" or "sharpening distance" will continue to be used to refer to objective function values in the following discussion.

3.2. Convergence and run time

For the present discussion an optimizer is defined to have converged if it reaches any of its normal stop conditions and returns a feasible solution. Table 1 shows the proportion of runs converging, and the median run time, for all three methods across simulation cases.

The greedy algorithm converged for all simulation runs, because it is designed to always return a feasible solution. Sequential quadratic programming had some failures to converge in seven

			Proportion converging			Median run time (s)		
Density	Bandwidth	n	Greedy	SQP	Combined	Greedy	SQP	Combined
t_3	$0.75h_{SJ}$	25	1	0.956	1	0.061	9.6	2.0
t_3	$0.75h_{SJ}$	50	1	0.880	1	0.130	19.0	4.8
t_3	$0.75h_{SJ}$	100	1	0.844	0.992	0.310	41.0	16.0
t_3	h_{SJ}	25	1	0.964	1	0.044	3.9	2.0
t_3	h_{SJ}	50	1	0.936	0.996	0.092	15.0	4.7
t_3	h_{SJ}	100	1	0.916	0.996	0.210	38.0	15.0
Mixture	$0.75h_{SJ}$	25	1	1	1	0.120	4.4	3.6
Mixture	$0.75h_{SJ}$	50	1	1	1	0.280	9.7	8.4
Mixture	$0.75h_{SJ}$	100	1	0.992	1	0.740	31.0	28.0
Mixture	h_{SJ}	25	1	1	1	0.060	3.3	3.1
Mixture	h_{SJ}	50	1	1	1	0.140	8.2	7.7
Mixture	h_{SJ}	100	1	1	1	0.380	28.0	26.0

Table 1: Convergence and run time results.

of the twelve test conditions, including all six cases based on the t_3 distribution. In those cases the proportion of runs converging varied from 84 to 96 percent, with with larger sample sizes having lower percentages. Note that for SQP to record a failure, the algorithm must fail to return a feasible solution at all 20 candidate mode points attempted.

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

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

3.3. Optimization performance

The three optimization methods were run on the same data sets, so each method's sharpening distances can be directly compared. Figure 6 shows the objective function values for the greedy and SQP methods plotted against each other, for the 2872 pairs of optimizations where SQP converged. Cases based on each of the two target distributions are plotted with different markers. The 1:1 line is also shown on the plot. Points below the line represent runs where the greedy method outperformed SQP, and points above the line represent runs where SQP found the better solution.

The figure suggests that the greedy method had good relative performance. While most of the runs had similar results for the two methods, there were also a large number of runs where the SQP L_1 value greatly exceeded the greedy value. These are runs where SQP stopped at a particularly poor local minimum. Interestingly, most of these poor SQP results arose in

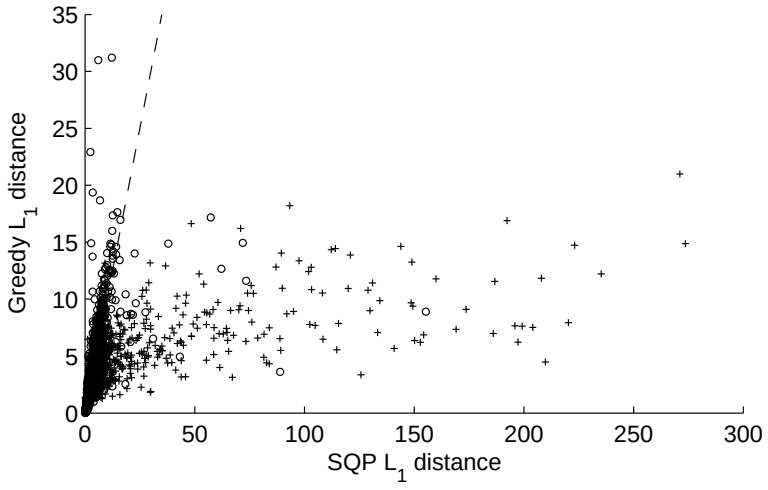

Figure 6: Scatter plot of L_1 sharpening distances across all simulation runs. Circles and crosses denote cases based on the t_3 and mixture distributions, respectively.

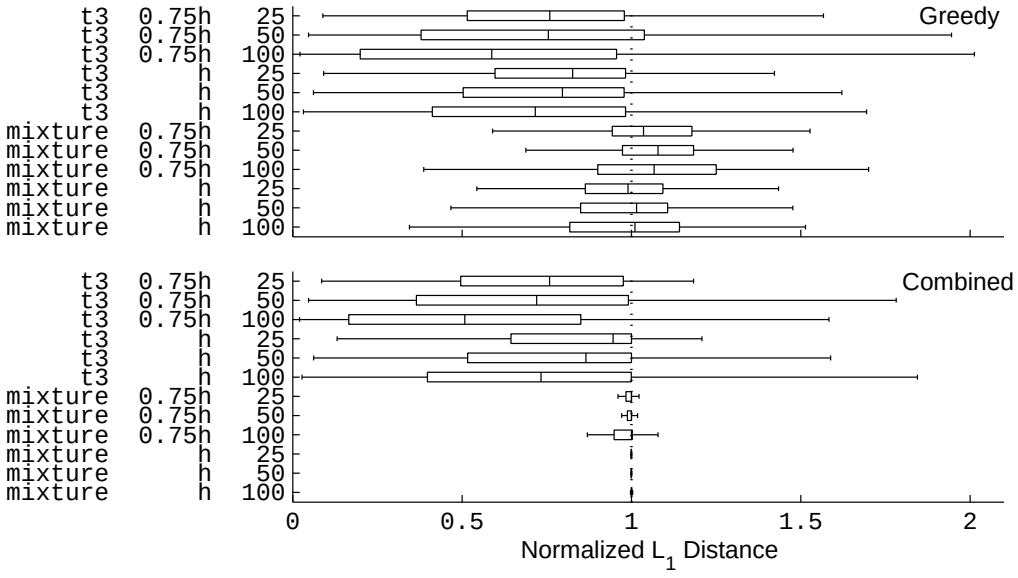

Figure 7: Box plots of normalized L_1 distance, for both the greedy and combined search methods. The labels at left give the simulation case.

problems based on the mixture distribution, where convergence was not a problem. There were also some data sets where SQP greatly outperformed greedy, but such cases were much less frequent.

To facilitate a more detailed comparison, objective function values for the greedy and combined methods were normalized relative to the SQP sharpening distance for the same sample and bandwidth. The normalized sharpening distance is the ratio of that method's L_1 distance to the SQP value. Figure 7 shows box plots of the normalized sharpening distance for both

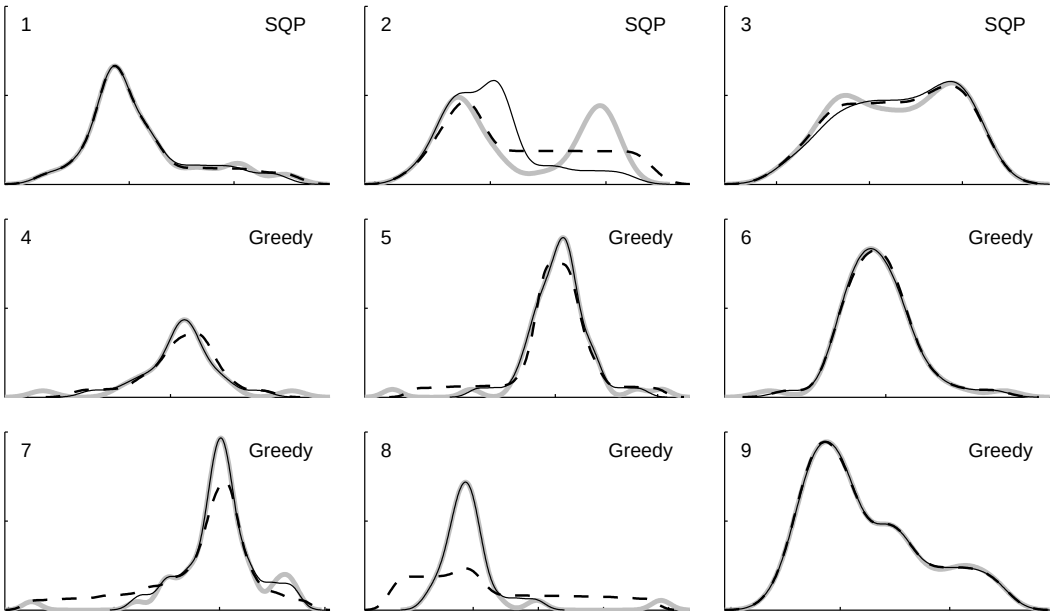

Figure 8: Comparing the unsharpened estimate (thick grey line), greedy estimate (thin solid line), and SQP estimate (dashed line) for nine simulation data sets. Plot labels indicates which method had a smaller L_1 sharpening distance.

the greedy and combined methods, for all 12 simulation cases. Boxes show locations of the first, second, and third quartiles, and whiskers extend to the most extreme values differing from the median by less than 1.5 times the interquartile range.

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

Looking at the combined-method cases in Figure 7, it is clear that the combined method performed better than the default SQP with $\mathbf{x}$ as its starting point. Starting at the greedy optimum had a pronounced effect for the more difficult t_3 cases, but only a negligible effect on the mixture cases. Note that using the greedy starting point does not always improve the performance of SQP. The best starting point for SQP is sample-dependent and one could not expect any rule to provide the best start for all cases.

As a further illustration of the performance of the methods, Figure 8 gives plots of the density estimates for nine randomly-selected simulation data sets. Each plot gives the unsharpened density as well as the sharpened density based on both the SQP and greedy search. Plots 1–3 show cases where SQP found the better result, and plots 4–9 show cases where the greedy algorithm found the better result. These examples were sampled from only those cases where the relative difference in sharpening distance was large (the worse method's sharpening distance being at least 50% larger than the better method's).

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

Plot 2 in Figure 8 is something of a special case. The data happened to arise in such a way that the original estimate consisted of two modes of nearly equal height. In this case neither method could estimate the density well, and the results of either method would be highly sensitive to the initial solution provided.

3.4. Estimation performance

The simulation results can also be used to compare the statistical performance of the density estimators involved. The measure of estimation quality used here is the total variation (TV) distance between each estimate and the truth:

$$TV(\hat{f}_{\mathbf{y}}, f) = \frac{1}{2} \int_{-\infty}^{\infty} |\hat{f}_{\mathbf{y}}(t) - f(t)| dt. \tag{10}$$

Devroye and Lugosi (2001) examined different distance measures in a density estimation context and concluded that the TV distance has several theoretical advantages over more common measures like integrated squared error. In particular, TV admits a probability interpretation: $TV(\hat{f}_{\mathbf{y}}, f)$ is the maximum possible difference attainable when the same probability is calculated with the two densities $\hat{f}_{\mathbf{y}}$ and f. That is, for any Borel set B, $|P_{\hat{f}_{\mathbf{y}}}(B) - P_f(B)| \leq TV(\hat{f}_{\mathbf{y}}, f)$.

Table 2 summarizes the results. For each combination of true density and sample size, the average TV distance is shown for seven different estimators. The pairs of columns labeled KDE, SQP, and Greedy give the results for the unsharpened (multimodal) KDE, the unimodal SQP estimate, and the unimodal greedy estimate, respectively; the subscripts on the labels indicate the bandwidth (h_{SJ} or $0.75h_{SJ}$). The column labeled Reboul gives the TV distance for the unimodal estimator described by Reboul (2005), which is an extension of the work of e (1997). This estimator is a histogram with automatically-determined variable bin widths. While it is not smooth, its performance can be used as a reference point, since an upper bound on its L_1 risk has been established (Reboul 2005).

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

		Bandwidth $h = h_{SJ}$			Bandwidth $h = 0.75h_{SJ}$		
Case	Reboul	KDE	SQP	Greedy	KDE	SQP	Greedy
t_3, $n = 25$	0.258	0.156	0.166	0.149	0.167	0.187	0.151
t_3, $n = 50$	0.189	0.120	0.132	0.114	0.127	0.149	0.116
t_3, $n = 100$	0.147	0.095	0.119	0.091	0.101	0.133	0.093
Mixture, $n = 25$	0.242	0.183	0.175	0.173	0.179	0.162	0.162
Mixture, $n = 50$	0.182	0.148	0.141	0.140	0.142	0.129	0.126
Mixture, $n = 100$	0.139	0.118	0.113	0.111	0.111	0.106	0.098

Table 2: Sample mean of TV distances from the truth across 250 replications, for seven estimators and six test cases. The SQP results exclude runs that failed to converge. The standard error of the estimate is less than 0.0036 for all table entries.

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

4. MATLAB code and usage examples

This paper includes code for running the new greedy algorithm in MATLAB, including associated functions for calculating KDEs and checking the unimodality constraint. Below we will review the included code and then illustrate its use on one univariate and one bivariate example.

4.1. The included code

The code consists of ten files: seven functions, one script, and two data sets. Table 3 lists the file name, typical syntax, and a brief description for each. All of the files in the table must be in MATLAB's current directory or on the search path to use the greedy method or to run the examples that follow.

The general usage of the code is as follows. The main function `improve(v, x, confun)` is used to perform data sharpening via the greedy algorithm. Its arguments are `v`, the initial unimodal guess solution, `x`, the unsharpened data, and `confun`, a handle to a function that checks whether a proposed solution is unimodal. The main function has been named `improve` to reflect the fact that the objective function value of `v` will always be improved upon by moving it toward `x`, as long as `v` has at least one moveable point.

The constraint-checking function `confun` must be constructed by the user before `improve` can be called. The functions `bkde` and `isuni` are used to do this: `bkde` evaluates the KDE at a grid of points, and `isuni` checks whether the function values form a unimodal curve. So the typical construction is `confun = @(y) isuni(bkde(y, h, G))`, where `h` is the bandwidth and `G` is the number of grid points.

This brief description and the examples that follow should be sufficient to demonstrate how the functions are used. The interested user is advised to consult the help text and comments

File name	Type	Typical usage	Description
`improve.m`	Function	`y = improve(v,x,confun)`	Executes the greedy algorithm.
`isuni.m`	Function	`tf = isuni(fhat)`	Checks whether a density estimate is unimodal.
`isuni2D.m`	Function	`tf = isuni2D(fhat)`	Two-dimensional unimodality-checking function.
`bkde.m`	Function	`f = bkde(y,h,G)`	Calculates the binned kernel density estimator approximation at points specified by `G`.
`bkde2D.m`	Function	`f = bkde2D(X,h,G)`	Two-dimensional binned kernel density estimator approximation.
`match.m`	Function	`ynew = match(y, x)`	For matching the sharpened data `y` to the unsharpened data `x`. Called by `improve`.
`SJbandwidth.m`	Function	`h = SJbandwidth(x)`	To calculate the Sheather-Jones plug-in bandwidth. Used in the examples.
`examples.m`	Script	`edit examples`	Examples of this section.
`windspeed.mat`	Dataset	`load windspeed`	Data for the univariate example.
`kidney.mat`	Dataset	`load kidney`	Data for the bivariate example.

Table 3: List of files included with the paper.

within each file to learn the details of each function, including optional input and output arguments that are not discussed here.

Code for running the examples is found in the script file `examples.m`.

4.2. Univariate example: Wind speed data

This example will proceed line-by-line through a univariate problem, to show the sequence of commands involved in a typical analysis.

Alibrandi and Ricciardi (2008) reported data on 57 wind speed measurements made at each of five different elevations in Italy's Messina Strait region. The dataset array `windspeed`, stored in `windspeed.mat`, contains these data. The present analysis will compare the distributions of the measurements made at the lowest and highest positions (10 and 128 meters), by producing a plot with regular kernel density estimates and their sharpened unimodal counterparts.

The first step in the analysis is to load the data and perform some preliminary operations.

```
load windspeed
x1 = double(windspeed.h10);
x2 = double(windspeed.h128);
n = 57;
figure(1)
set(gcf, 'Units', 'pixels', 'Position', [100 100 800 400])
```

Two vectors, `x1` and `x2`, have been created to hold the speed values at the two heights, and a figure window has been prepared. Next, the original (unsharpened) density estimates will be added to the plot. Doing so illustrates how the function `bkde` works.

```
h1 = SJbandwidth(x1);
h2 = SJbandwidth(x2);
```

```
[f1 g1] = bkde(x1, h1, 250);
[f2 g2] = bkde(x2, h2, 250);
plot(g1, f1, 'k-');
hold on
plot(g2, f2, 'k--');
```

The function `SJbandwidth` has been used to choose a bandwidth for each dataset, using the plug-in method of Sheather and Jones (1991). The binned kernel density estimates were calculated using, e.g., `[f1 g1] = bkde(x1, h1, 250)`. This calculates the estimate at a grid of 250 points extending beyond the range of the data, using `x1` as the data and `h1` as the bandwidth. The grid values and function values are returned in `g1` and `f1`, respectively, making it easy to plot the estimate.

Moving on to find the shape-constrained estimate, it is necessary to set up the constraint checking function `confun` and to find the initial guess `v`. Then, `improve` can be called.

```
confun1 = @(y) isuni(bkde(y, h1, 250));
v1 = g1(find(f1 == max(f1), 1, 'first')) * ones(n, 1);
y1 = improve(v1, x1, confun1);
confun2 = @(y) isuni(bkde(y, h2, 250));
v2 = g2(find(f2 == max(f2), 1, 'first')) * ones(n, 1);
y2 = improve(v2, x2, confun2);
```

The code consists of three identical lines for each case; let us consider the first set of lines. First, the anonymous function `confun1` is constructed. It calls the binned kernel density estimator, submitting the output of that function to the unimodality-checking function `isuni`. The result is a logical value of `true` if the supplied data vector produces a unimodal estimate.

Second, the starting value `v1` is created. The location of the highest mode of `f1` is found using `g1(find(f1 == max(f1), 1, 'first'))`, and this value is multiplied by a vector of ones to create the starting solution.

Third, the `improve` function is called to run the greedy algorithm. Its output is the sharpened data vector. Note that there are no density estimation steps native to the `improve` function. All of the estimation steps are done during calls to `confun`. The main function is only moving points and checking for feasibility. Having generated the sharpened data vectors, the unimodal estimates can be added to the figure:

```
[f1u g1u] = bkde(y1, h1, 250);
[f2u g2u] = bkde(y2, h2, 250);
plot(g1u, f1u, 'k-', 'linewidth', 2)
plot(g2u, f2u, 'k--', 'linewidth', 2)
xlabel('Speed')
ylabel('f(speed)')
```

The final result is shown in Figure 9. Considering the original estimates, both curves have three modes: a main peak, a smaller peak in the shoulder of the distribution, and an extra mode in the right tail. The sharpened estimates match the unsharpened ones away from the extra modes, but shift the curve as necessary near those modes to satisfy the unimodality constraint.

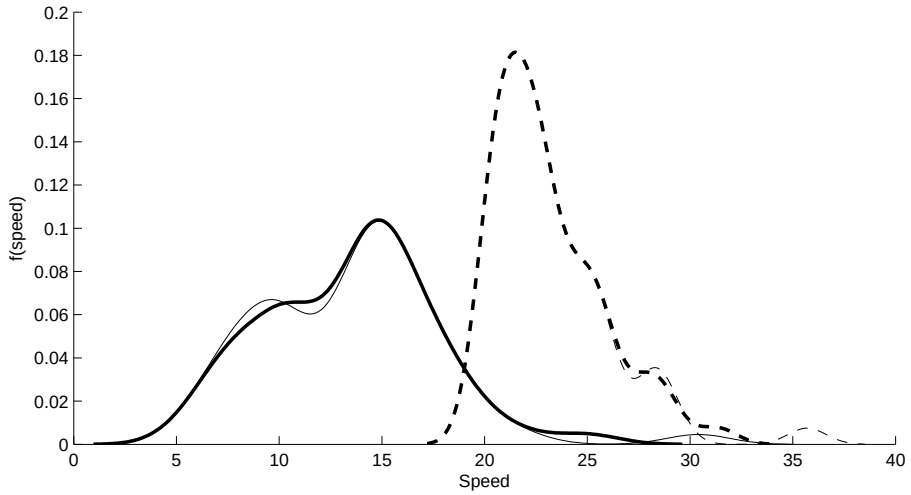

Figure 9: Unsharpened and sharpened KDEs for the windspeed data, at 10 meters (left pair of curves) and 128 meters (right pair of curves). In both cases the dark curve is the unimodal sharpened estimate.

4.3. Bivariate example: Kidney data

The file `kidney.mat` contains data arising from a study of the association between birth weight and kidney length for newborns (Fitzsimons 1983). The data consists of these two measurements taken from a sample of 102 babies. The detailed code for this example is not given here. Rather, a description of the important points will be provided, and the output will be examined. The interested reader can replicate the analysis by running the script in the file `examples.m`.

The function `improve` (as well as `match`, the function it calls to perform the matching step) has been written to work in d dimensions; it does not need modification for the bivariate case. In the call `y = improve(v, x, confun)`, `y`, `v`, and `x` will be n-by-1 vectors for univariate data, and n-by-d matrices for d-dimensional data (here, $d = 2$). The parts of the analysis that do need modification, however, are the density estimation and constraint-checking functions.

Kernel density estimation in two dimensions can be carried out using the function `bkde2D`. This function implements the Gaussian product kernel (the kernel functions are bivariate normal distributions with no correlation), with bandwidths h_1 and h_2. As in the univariate case, a binned approximation (Wand 1994) is used to speed up computation. For the present problem, the bandwidth in each component direction is set to 90% of the value chosen by the normal-reference bandwidth selection rule (Silverman 1986, pp. 86–87).

Bivariate constraint checking is carried out using the function `isuni2D`. The input to this function is a matrix of function values evaluated at a uniformly-spaced k-by-k grid defined on the plane covering the range of the data. Such a matrix is output by `bkde2D`. With these function values, the `isuni2D` function can check for unimodality—that is, the presence of only one local maximum and no local minima in the function values. Two other types of unimodality check can also be applied. Unimodality of all conditional distributions can be checked by applying a univariate unimodality check to the rows and columns of the matrix,

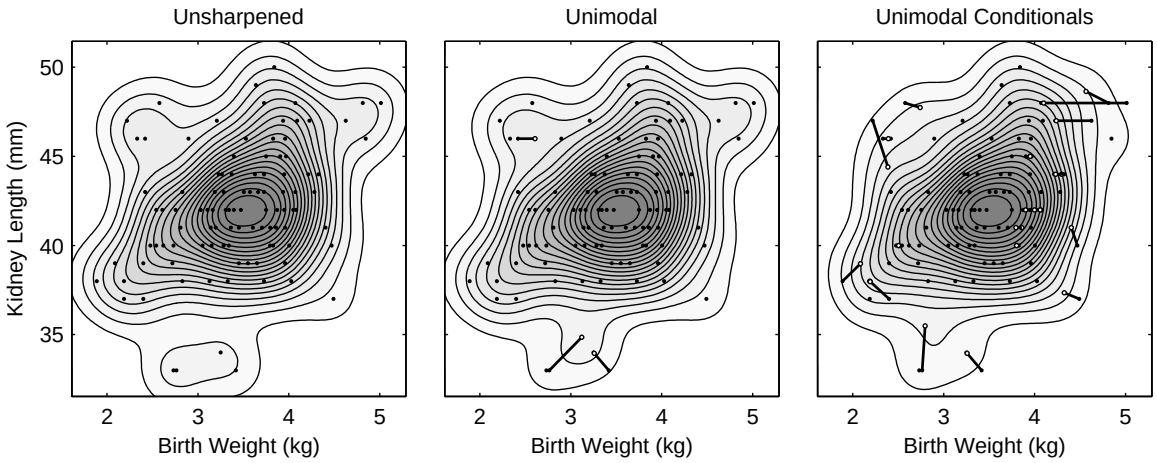

Figure 10: Density estimates for the kidney data. Lines connect any pinned sharpened points to their final targets. Darker shading corresponds to regions of higher density.

and unimodality of marginal distributions can be checked by numerically integrating across each matrix dimension and then checking the results for univariate unimodality. These checks can also be used in combination.

Returning to the kidney data, the unsharpened density estimate is shown in the leftmost plot of Figure 10. It has two spurious modes: one at the bottom of the plot and one at the upper left. The unimodal sharpened estimate is shown in the middle plot. Lines in the plot connect each sharpened data point that is not at home, to its target point. Only three points have been moved to eliminate the two small spurious modes.

The rightmost plot of Figure 10 shows the sharpening result when the density is constrained to have unimodal conditionals. This constraint would be useful if, for example, the researchers felt that babies of any fixed birth weight should have a unimodal distribution of kidney lengths. In this more restrictive case, more points have been moved.

5. Discussion

The simulation study and examples just presented show the potential of the greedy method. Although it is a relatively simple heuristic, it has some distinct advantages over mathematical programming when applied to unimodal kernel density estimation problems.

Foremost among the advantages of the greedy algorithm is its speed and reliability relative to SQP. The practical advantages of this are considerable. In an exploratory analysis, a researcher may wish to view density estimates under a variety of possible scenarios or bandwidth values. Using SQP or a similar method, this will be a slow process and could include failures to converge at certain points. Though it was not considered here, `improve` could also be used unchanged with a different nonparametric density estimator. Making such a change with SQP requires a considerable effort in calculation and coding to update the derivative information and constraint functions for the new estimator.

The advantage of the heuristic approach becomes more pronounced as the constraints get

more complicated and more highly nonlinear. The kidney example with unimodal conditionals (Figure 10) provides an excellent example of this. The constraint-checking function for this case involves evaluating the density at a rectangular grid of points, and then checking whether the first difference along every horizontal and vertical grid line changes sign only once. To cast this constraint in a mathematical programming framework is very difficult unless the mode along each grid line is known in advance (in which case each grid line produces a set of inequalities like Equation 5).

There are some statistical questions related to the greedy estimator (and to data sharpening in general) that have not been addressed here. These include how to generate pointwise interval estimates for the sharpened density, and how one could test or evaluate whether the constraint is appropriate for a given set of data. A particularly relevant question is how bandwidth selection procedures could be modified to account for the data sharpening step. The appendix contains one proposal, but others could doubtless be brought forward.

One area of particular concern with data sharpening density estimators is the behavior of the estimators in the tails of the distribution. Examining the wind speed data (Figure 9) for example, both of the distributions had an outlying point causing a spurious mode in the right tail. Such a data point is fundamentally problematic for unimodal density estimation with a fixed-bandwidth estimator. A standard KDE can only be made unimodal in this case by oversmoothing the density; data sharpening improves on this by allowing the point to move inward, but is still subject to a trade-off between oversmoothing the density and shifting the point too far. Simulation studies focused on bandwidth selection and tail behavior for unimodal estimators would be of value here.

Future work in this area could consider all of the above questions, but also could involve improved optimization techniques to more closely approach the unknown global optimum of problem (3). One could devise iterative or population-based versions of the greedy algorithm, or different heuristics altogether, to improve performance. Ideally a data sharpening optimizer could be applied to other shape-constrained estimation problems as well, not just unimodal density estimation. The `improve` function is considered a first step toward a robust optimizer for this type of problem.

Acknowledgments

The author thanks Professor W. John Braun for many productive discussions related to this research, and an anonymous referee for suggesting the Reboul estimator as a benchmark. Funding from the National Sciences and Engineering Research Council of Canada and the Ontario Student Assistance Program are also gratefully acknowledged.

References

Alibrandi U, Ricciardi G (2008). "Efficient Evaluation of the PDF of a Random Variable Through the Kernel Density Maximum Entropy Approach." *International Journal for Numerical Methods in Engineering*, **75**, 1511–1548.

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

Braun WJ, Hall P (2001). "Data Sharpening for Nonparametric Inference Subject to Constraints." *Journal of Computational and Graphical Statistics*, **10**(4), 786–806.

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

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

e LB (1997). "Estimation of Unimodal Densities without Smoothness Assumptions." *The Annals of Statistics*, **25**(3), 970–981.

Fitzsimons RB (1983). "Kidney Length in the Newborn Measured by Ultrasound." *Acta Paediatrica Scandinavica*, **72**, 885–887.

Fougeres AL (1997). "Estimation de Densites Unimodales." *The Canadian Journal of Statistics*, **25**(3), 375–387.

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

Hall P, Huang LS (2002). "Unimodal Density Estimation Using Kernel Methods." *Statistica Sinica*, **12**, 965–990.

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

Nocedal J, Wright SJ (1999). *Numerical Optimization*. Springer-Verlag.

Numerical Algorithms Group (2009). ***NAG*** *Toolbox for* MATLAB, *Mark 21E*. Numerical Algorithms Group, Oxford. URL http://www.nag.co.uk/.

Racine JS, Parmeter CF (2008). "Constrained Nonparametric Kernel Regression: Estimation and Inference." URL http://www.maxwell.syr.edu/uploadedFiles/econ/kernel_cons.pdf.

Reboul L (2005). "Estimation of a Function Under Shape Restrictions. Applications to Reliability." *The Annals of Statistics*, **33**(3), pp. 1330–1356.

Sheather SJ, Jones MC (1991). "A Reliable Data-Based Bandwidth Selection Method for Kernel Density Estimation." *Journal of the Royal Statistical Society B*, **53**, 683–690.

Silverman BW (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall.

The MathWorks, Inc (2007). MATLAB *– The Language of Technical Computing, Version 7.4.0*. The MathWorks, Inc., Natick, Massachusetts. URL http://www.mathworks.com/products/matlab/.

Wand MP (1994). "Fast Computation of Multivariate Kernel Estimators." *Journal of Computational and Graphical Statistics*, **3**(4), 433–445.

Wand MP, Jones MC (1995). *Kernel Smoothing*. Chapman & Hall, London.

Wegman EJ (1972). "Nonparametric Probability Density Estimation: I. A Summary of Available Methods." *Technometrics*, **14**(3), 533–546.

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

A. A proposed bandwidth selector

As mentioned in Section 1.3, there is some justification for the simple approach of selecting the bandwidth prior to data sharpening, and holding it fixed thereafter. Still, a bandwidth selection procedure designed for data sharpening estimators would be welcome.

Optimal bandwidth selection for data sharpened density estimators is an open problem, beyond the scope of the present work. Rather than try to solve the problem here, we propose an option for bandwidth selection that has some intuitive appeal for unimodal density estimation.

Consider first the likelihood cross-validation bandwidth selector for unconstrained kernel density estimation (Silverman 1986, p. 52-55). Let $\hat{f}$ be the standard KDE and $\hat{f}_{-i}$ be the KDE with x_i withheld from the data set. Then likelihood cross-validation selects

$$h^* = \operatorname*{argmax}_h \prod_{i=1}^{n} \hat{f}_{-i}(x_i; h) \tag{11}$$

as the optimal bandwidth. It is necessary to withhold x_i to ensure that h^* is nonzero (if x_i is not left out, the product $\prod_{i=1}^{n} \hat{f}(x_i; h)$ approaches infinity as $h \to 0$). Finding h^* involves conducting a line search over the possible values of h.

Directly applying likelihood cross-validation to the data sharpening case would yield the bandwidth

$$h_{LCV} = \operatorname*{argmax}_h \prod_{i=1}^{n} \hat{f}_{\mathbf{y}_{-i}}(x_i; h), \tag{12}$$

where the notation $\mathbf{y}_{-i}$ indicates that i-th point is withheld from the sample before data sharpening. Implementing (12) is computationally intensive, as it requires n data sharpening runs per candidate h value (compared to n KDE evaluations per h value to obtain h^*).

The existence of the unimodality constraint makes it unnecessary to withhold points to obtain a reasonable bandwidth, however. When the constraint is enforced, the product $\prod_{i=1}^{n} \hat{f}_{\mathbf{y}}(x_i; h)$ approaches zero, not infinity, as $h \to 0$. So rather than using h_{LCV}, it is proposed to use

$$h_{ML} = \operatorname*{argmax}_h \prod_{i=1}^{n} \hat{f}_{\mathbf{y}}(x_i; h) \tag{13}$$

to choose the bandwidth of the unimodal data sharpened density estimator.

The product in the right hand side of (13) is the likelihood of $\mathbf{x}$ under the density $\hat{f}_{\mathbf{y}}$, if we take $\mathbf{y}$ to be a fixed vector (rather than what it truly is, a function of $\mathbf{x}$ and h). This resemblance to a maximum likelihood estimate motivates the notation h_{ML}. The proposed bandwidth has intuitive appeal because, just like true maximum likelihood estimators, it favors h choices that put higher probability mass on the observed sample. Also, only one data sharpening run is performed for each candidate h value, an approximately n-fold reduction in computational effort compared to h_{LCV}.

Table 4 compares the mean TV distance from the truth for the simulated data sets of Section 3, using h_{SJ}, h_{ML}, and h_{LCV} as the bandwidth in the greedy algorithm. Bandwidth h_{ML} had better performance than h_{LCV} in all cases. Relative to h_{SJ}, it showed improved performance for the mixture distribution cases, but reduced performance for the t_3 cases. The reduced performance in the t_3 case is caused by sensitivity to outliers. Any estimate that puts a negligible mass on an outlying point will have a likelihood near zero, and as a consequence h_{ML}

Case	h_{SJ}	h_{ML}	h_{LCV}
t_3, $n = 25$	0.149	0.173	0.209
t_3, $n = 50$	0.114	0.140	0.165
t_3, $n = 100$	0.091	0.120	0.140
Mixture, $n = 25$	0.173	0.156	0.175
Mixture, $n = 50$	0.140	0.124	0.139
Mixture, $n = 100$	0.111	0.097	0.111

Table 4: Mean TV distance from the truth for the greedy algorithm with three different bandwidth choices.

will favor larger bandwidths when there are outlying observations. The poor performance of h_{LCV} is also explained by outlier sensitivity, which is only exacerbated by withholding points from the estimate.

The tendency of h_{ML} to oversmooth in the presence of outliers could be viewed in either a positive or negative light. It has the benefit of avoiding density estimates that place negligible probability mass on points that were actually observed; but at the same time, outlier-induced oversmoothing near the mode will degrade overall measures of statistical quality. This conflict between generating plausible individual estimates and ensuring good average performance is a manifestation of the problem mentioned in the discussion: a fixed-bandwidth estimator has fundamental difficulties simultaneously estimating the tails and enforcing the unimodality constraint. A possible direction for future work is to sharpen the bandwidths, rather than the kernel centers; maximum likelihood could again be used to choose the optimal bandwidths in such an estimator.

Affiliation:

Mark A. Wolters
Department of Statistical and Actuarial Sciences
University of Western Ontario
London, Ontario N6A 5B7, Canada
E-mail: mwolters@uwo.ca

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 47, Issue 6 *Submitted:* 2010-08-23
April 2012 *Accepted:* 2012-04-05