The Fast RODEO for Local Polynomial Regression

Daniel V. Samarov

An open challenge in nonparametric regression is finding fast, computationally efficient approaches to estimating local bandwidths for large datasets, in particular in two or more dimensions. In the work presented here, we introduce a novel local bandwidth estimation procedure for local polynomial regression, which combines the greedy search of the regularization of the derivative expectation operator (RODEO) algorithm with linear binning. The result is a fast, computationally efficient algorithm, which we refer to as the fast RODEO. We motivate the development of our algorithm by using a novel scale-space approach to derive the RODEO. We conclude with a toy example and a real-world example using data from the Cloud-Aerosol Lidar and Infrared Pathfinder Satellite Observation (CALIPSO) satellite validation study, where we show the fast RODEO’s improvement in accuracy and computational speed over two other standard approaches.

Key Words: Nonparametric regression; LIDAR; Local polynomial regression.

1. INTRODUCTION

Multivariate nonparametric regression is an invaluable tool for exploratory data analysis and inference, particularly local polynomial regression (LPR; Fan and Gijbels 1996), which has found widespread use. A key component of LPR and other nonparametric smoothers is the estimation of the bandwidth (i.e., smoothing) parameter or, in the multivariate case, the bandwidth matrix. With large amounts of data, for example, in image analysis and remote sensing, bandwidth estimation can be a computationally prohibitive process, in particular local bandwidth estimation, which we focus on here.

While considerable work has gone into the development of methods for global bandwidth selection (see Jones, Marron, and Sheather 1996 for a review of univariate methods), the development of computationally tractable methods for local bandwidth selection, especially in the multivariate setting, is an open and challenging problem. A powerful approach to locally adaptive smoothing that should be mentioned is the propagation separation approach to nonparametric smoothing (formally introduced as adaptive weighted smoothing, Polzehl

Daniel V. Samarov is Mathematical Statistician, Statistical Engineering Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899 (E-mail: daniel.samarov@nist.gov).

In the Public Domain
Journal of Computational and Graphical Statistics, Volume 24, Number 4, Pages 1034–1052
DOI: 10.1080/10618600.2014.949724
Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/r/jcgs.

1034
and Spokoiny 2000, 2006). The main idea behind the latter is to find the greatest possible neighborhood about a point in which the local parametric assumptions about the data are justified. While this method has many theoretically appealing properties and has been shown to have strong performance in many practical applications, the focus is not on computational speed and scalability and as such is not as well suited to larger scale problems.

One promising new approach for fast local (and global) multivariate bandwidth selection is the regularization of the derivative expectation operator algorithm (RODEO; Lafferty and Wasserman 2008). The RODEO algorithm was developed as a general nonparametric method for bandwidth and variable selection in high-dimensional data. Here, we focus on bandwidth estimation using the RODEO. The principal behind the algorithm is to continually decrease the bandwidth by infinitesimal increments and determine whether these decreases lead to a significant change in the estimator. Given a particular value of the bandwidth, this is equivalent to testing whether the derivative of the estimator as a function of the bandwidth is significant; if it is significant the algorithm proceeds in a greedy manner to a smaller bandwidth but otherwise it stops. This procedure is repeated independently for each observation along each dimension, the final output is a collection of bandwidth estimates for each observation.

Herein, we introduce the fast RODEO, a novel approach to bandwidth estimation that combines the greedy selection search of the RODEO with linear binning (Fan and Marron 1994; Wand 1994). The integration of these methodologies results in faster, more efficient, and scalable local bandwidth estimation as compared to standard approaches. We also give an example of a scale-space-based interpretation of the standard and fast RODEO algorithms, providing new insights into their behavior.

The application we focus on here is high spectral resolution light detection and ranging (HSRL). HSRL is a remote sensing technology that is commonly used to characterize clouds and small particles in the atmosphere called aerosols. The measurements collected by this system are backscattered light or photons from these clouds and particles. Thus, a single measurement at a given spatial location is the aggregation of measured light along the path from that particular point to the HSRL system. To obtain an actual estimate of cloud density or aerosol concentration at any given spatial location, the derivative of these raw measurements must be taken.

The dataset we consider here include HSRL measurements of aerosols collected as part of the Cloud-Aerosol Lidar and Infrared Pathfinder Satellite Observation (CALIPSO) satellite validation study (http://science.larc.nasa.gov/hsrl/CALIPSO-VAL.html). These approximately 300 vertical by 20,000 horizontal measurements were collected from an aircraft flying at an altitude of approximately 9 kilometers (km) over a 4-h time period, with data collected at 0.5 sec intervals. Figure 1, taken from Rogers et al. (2009), shows a plot of the flight path and Figure 2 the corresponding “unfolded” raw measurements. As mentioned above, we refer to these as the raw measurements as we are interested in the derivative (which must be estimated) along the Altitude-axis which will give us the desired measurement, that is, aerosol concentrations. Also note that the x-axis, labeled as “Time (hrs)” corresponds to a time stamp of the spatial location of the plane taken from the beginning of the flight (as shown in Figure 1). These measurements have been preprocessed/corrected to adjust for movement of the plane, the effects of wind, and other environmental factors. While there is no ground truth available for the CALIPSO data to confirm that a local
Figure 1. Flight path in the CALIPSO validation study.

bandwidth is optimal, the distribution of measurements shown in Figure 2 suggests that there is considerable variation in the scale of local features. As a result, if a global bandwidth was used, it may smooth out features that are present and/or undersmooth those that are not.

In Section 2, we provide an overview of the RODEO algorithm, the principles of binning in the context of LPR, and how these methods can be interpreted in scale-space. In Section 3, we describe the fast RODEO procedure, and in Section 4 we provide results on both a toy and real world example.

Figure 2. The “unfolded” measurements shown in Figure 1.
2. BACKGROUND

2.1 THE RODEO

In this section, we provide an overview of the RODEO algorithm. Let $Y_i \in \mathbb{R}$ and $X_i = (X_{i,1}, \ldots, X_{i,d}) \in \mathbb{R}^d, i = 1, \ldots, n$ denote the set of observed sample pairs. In the HSRL example, $Y_i$ would be the measured aerosol extinction and $X_i = (X_{i,1}, X_{i,2})$ the latitude and time stamp of that measurement. Next, define $m(x)$ to be the true, unknown function so that $Y_i = m(x) + v(x)^{1/2}\epsilon(x)$. Here, the smooth function $v(x)$ models the local variation in the data (assumed to be equal to a constant $\sigma$ for the purposes of the work presented here) and $\epsilon(x)$ is the iid random error with mean 0 and variance 1. Let $\hat{m}_H(x)$ denote the LPR estimate with diagonal bandwidth matrix $H = \text{diag}(h_1, \ldots, h_d)$, and $x = (x_1, \ldots, x_d)$ some target point. It can be shown that the LPR solution has the form

$$\hat{m}_H(x) = e_1^T (X^T X)_{1,1}^{-1} X^T Y,$$

(1)

Here $e_1 = (1, 0, \ldots, 0)^T$, $X_x$ is a $p$th-order, $d$-dimensional polynomial basis, $W_{X,H} = \text{diag}(K_H(X_1 - x), \ldots, K_H(X_n - x))$, and $K_H(x) = |H|^{-1/2}K(H^{-1/2}x)$, where $K(\cdot)$ is assumed to be a product kernel. To illustrate the form of $X_x$ suppose $p = 1$, which corresponds to local linear regression (LLR), then

$$X_x = \begin{pmatrix} 1 & X_1 - x & \vdots & \vdots & 1 & X_n - x \end{pmatrix} \in \mathbb{R}^{n \times (d+1)}.$$

Following Lafferty and Wasserman (2008), we express the “effective kernel” as

$$G(u, v, H) = e_1^T (X^T W_{X,H} X)_{1,1}^{-1} (1 (u - v)^T) K_H(u - v),$$

(2)

so that $\hat{m}_H(x) = \sum_{i=1}^n G(X_i, x, H)Y_i$.

Recall that the RODEO algorithm looks to determine whether an infinitesimal decrease in the bandwidth will lead to significant changes in the estimator. This is done by testing whether the derivative of the bandwidth in the direction of the $j$th dimension defined as

$$Z_j = \frac{\partial \hat{m}_H(x)}{\partial h_j} = \sum_{i=1}^n G_j(X_i, x, H)Y_i,$$

(3)

with

$$G_j(u, v, H) = \frac{\partial G(u, v, H)}{\partial h_j}$$

(4)

is significant. Significance of $Z_j$ is determined by testing whether $|Z_j| \geq \sqrt{2 \log n V_j^2}$, where

$$V_j^2 = \text{var}(Z_j|X_1, \ldots, X_n) = \sigma^2 \sum_{i=1}^n G_j(X_i, x, H)^2.$$  

(5)

Several methods for estimating $\sigma$ are given in Lafferty and Wasserman (2008), the details of which are also provided in the supplemental material (Section 1 and additional discussion at the end of Section 2).
1. Select a constant $0 < \beta < 1$ and initial bandwidth $h_0^j = c_0^j / \log \log n$, where $c_0^j$ is a sufficiently large constant so that the smooth produced by $h_0^j$ is effectively linear.

2. Initialize the bandwidths, and activate all covariates:
   (a) $h_j = h_0^j$, $j = 1, \ldots, d$.
   (b) $A = \{1, \ldots, d\}$.

3. While $A$ is nonempty, do for each $j \in A$:
   (a) Calculate derivative estimate $Z_j$ in (3) and $V_j$ in (5).
   (b) Compute the threshold $\lambda_j = V_j \sqrt{2 \log n}$.
   (c) If $|Z_j| > \lambda_j$, then set $h_j \leftarrow \beta h_j$; otherwise remove $j$ from $A$.

4. Output bandwidths $H^* = \langle h_1, \ldots, h_d \rangle$ and estimator $\hat{m}_{H^*}(x)$.

To summarize: for each point $x \in \mathbb{R}^d$, the RODEO algorithm starts with a $d \times d$ bandwidth matrix $H = \text{diag}(h_0^1, \ldots, h_0^d)$, for $h_0^j(>0)$ sufficiently large (so that we are essentially performing standard linear regression), $j = 1, \ldots, d$. Then for each $j$, we calculate $Z_j$ and $V_j$. If $|Z_j| > \lambda_j$, then set $h_j \leftarrow \beta h_j$; otherwise remove $j$ from $A$. This process is repeated for each observation. The algorithm is summarized in Figure 3. The final output is a set of local bandwidth estimates $H^*$ for each $x$.

**Example 2.1.** To provide a more concrete illustration of what the RODEO solution path will look like, consider the example in Figure 4. The data, shown in the $x, y$-plane are sampled from $f(x) = \sin(5/x) + \epsilon$, where $\epsilon \sim N(0, 1/2)$ and $x \in (1/4, \pi)$; along the $z$-axis is a sequence of standard, fixed bandwidth LPR smooths of the data, placed in order from smallest (“furthest” from us) to largest (“closest” to us) bandwidth. Collectively, this series of smooths creates a “scale-space surface” (Lindeberg 1994; Chaudhuri and Marron 2000; Godtliebsen, Marron, and Chaudhuri 2002; Holmström 2010). Note, in general any type of smoothing method can be used, not just LPR.

The line along the scale-space surface is the RODEO solution path whose coordinates are given by $(x, \hat{m}_h, h)$, where the value of $h$ for each $x$ is determined by the RODEO (outlined in Figure 3). Visualized in this way the RODEO solution path can be thought of as being the derivative along the $z$-axis in Figure 4, followed by computing the test statistics, $Z$ and $V$ and proceeding as previously discussed.

As expected the size of the bandwidths is directly related to the fineness/coarseness of the local features in the data. In those regions with finer features, the RODEO moves further down the surface (toward smaller bandwidths) and in regions with coarser features it does not (remaining at larger bandwidths). Additional discussion on the relationship between the RODEO and scale-space can be found in the supplemental materials.
Figure 4. The line along the scale-space surface shows the final RODEO solution path.

2.2 MULTIVARIATE BINNING

Binning has been successfully applied in the nonparametric regression setting to improve computational speed and efficiency in both the univariate (Fan and Marron 1994) and multivariate cases (Wand 1994) for fixed global bandwidths. We will show how the same principles can be applied to dramatically improve the computational speed of the RODEO algorithm and local bandwidth estimation. First, we provide an overview of multivariate binning in the context of LPR.

The principle behind binning is the following: assuming that two points located near one another will tend to be close in value, many of the kernel evaluations will be quite similar. Thus, the number of necessary evaluations can be greatly reduced without sacrificing much in the way of accuracy by performing these computations at a reduced number of representative locations. In the context of binning, this is achieved by first constructing an equally spaced grid spanning the range of the $X_i$'s. The data are then approximated by “binning” each $X_i$ with its nearest grid point and fitting the LPR model to this grid of values. This has the effect of both reducing the number of calculations (assuming that the number of grid points is less than the original number of observations) and allowing all the kernel evaluations to be precomputed as the distance between the points on the grid is now fixed.

We now provide details needed for developing the binned implementation of the RODEO algorithm. First, define the matrices

$$T_1 = X_x^T W_{x,H} Y$$
$$S_1 = X_x^T W_{x,H}.$$  \hspace{1cm} (6)

which are the same matrices that appear in (1). For the remainder of this article, we suppress the subscripts $x$ and $H$ from the matrices $X_x$ and $W_{x,H}$. Following the notational convention
of Wand (1994), let

\[ \hat{t}_{1,k} = T_1(k) = \sum_{i=1}^{n} (X_i - x)^k K_H(X_i - x) Y_i \]

\[ \hat{s}_{1,k} = S_1(k) = \sum_{i=1}^{n} (X_i - x)^k K_H(X_i - x), \tag{7} \]

where \( k = (k_1, \ldots, k_d) \), and for a \( d \)-vector \( u = (u_1, \ldots, u_d) \), \( u^k = \prod_{i=1}^d u_i^{k_i} \). So in the case of LLR, using the notation in (6) and (7) we have

\[ \hat{\theta}(X) = \mathbf{e}_1^T \mathbf{S}_1^{-1} \mathbf{T}_1 = \mathbf{e}_1^T \left( \begin{array}{c} \hat{s}_{1,1,0} \\ \hat{s}_{1,1,1} \\ \hat{s}_{1,1,2} \end{array} \right)^{-1} \left( \begin{array}{c} \hat{t}_{1,1,0} \\ \hat{t}_{1,1,1} \\ \hat{t}_{1,1,2} \end{array} \right). \tag{8} \]

Next, let \( g_{j1}, \ldots, g_{jM_j} \) denote the equally spaced grid with \( M_j \) points along the \( j \)th coordinate axis such that the interval \([g_{j1}, g_{jM_j}]\) contains the range of values of \( X_i, i = 1, \ldots, n \). Define \( g_j = (g_{j1}, \ldots, g_{jM_j}) \), \( 1 \leq j \leq M \), \( i = 1, \ldots, d \) to be a point on the grid with index \( j = (j_1, \ldots, j_d) \) and let the \( j \)th bin width be denoted by \( \delta_j = (g_{jM_j} - g_{j1})/(M_j - 1) \).

From here on we assume that linear binning is being used (see Hall and Wand 1994 for a discussion on some of the advantages of linear binning over other binning schemes). Once an appropriate grid has been constructed, each data point must then be appropriately apportioned to its surrounding grid points. Figure 5 illustrates how linear binning works. Here the point marked “X” will have the top-right proportion of the shaded square assigned to the grid point \( g_1 \), top-left to \( g_2 \), bottom-right to \( g_3 \), and bottom-left to \( g_4 \). This process is repeated for every observation resulting in the set of grid counts \((c_j, d_j)\) that represent the amount of \((X, Y)\) data near each grid point. Specifically, if \( \omega(X) \) is the weight that \( X \) assigns to the grid point \( g_1 \) as described above, then

\[ c_1 = \sum_{i=1}^{n} \omega(X_i) \text{ and } d_1 = \sum_{i=1}^{n} \omega(X_i)Y_i. \]

The binned approximations of the quantities in (7) are

\[ \tilde{t}_{1,k}(g_j) = \sum_{i_1=1}^{M_{j_1}} \cdots \sum_{i_d=1}^{M_{j_d}} (g_{j1} - g_i)^k K_H(g_{j1} - g_i)d_{j1}, \]

\[ \tilde{s}_{1,k}(g_j) = \sum_{i_1=1}^{M_{j_1}} \cdots \sum_{i_d=1}^{M_{j_d}} (g_{j1} - g_i)^k K_H(g_{j1} - g_i)c_{j1}, \]

from which it can be shown that

\[ \tilde{t}_{1,k}(g_j) = \sum_{l_1=-L_1}^{L_1} \cdots \sum_{l_d=-L_d}^{L_d} d_j \kappa_{k,1}^{-1} \tilde{s}_{1,k}(g_j) = \sum_{l_1=-L_1}^{L_1} \cdots \sum_{l_d=-L_d}^{L_d} c_j \kappa_{k,1}^{-1}, \tag{9} \]

where \( \kappa_{k,1} = \prod_{j=1}^{d} K_h(l_j \delta_j)(l_j \delta_j)^k \), \( L_i = \min([\tau h_i/\delta_i], M_i - 1) \) and \([x]\) denotes the greatest integer less than or equal to \( x \). The parameter \( \tau \) corresponds to the interval
Illustration of Linear Binning

\[ -\tau, \tau \] outside of which the kernel function \( K \) is exactly or effectively 0. If \( K \) has infinite support, then one could replace \( K \) by \( K \mathbf{1}_{[-\tau, \tau]} \), where \( \mathbf{1}_I \) denotes the indicator function over the interval \( I \). The parameter \( \tau \) should be chosen such that it has a negligible effect on the estimation procedure. For example, in the case of a Gaussian kernel a reasonable choice would be \( \tau \approx 4 \).

The key observation is that the quantities in (9) are simply convolutions of \( \kappa_{k,1}^1 \) with the counts \( d_{j-1} \) and \( c_{j-1} \), respectively, and thus can be rapidly calculated using the fast Fourier transform (FFT). Wand (1995) showed that there is a considerable speed advantage using the FFT over a direct approach.

3. FAST RODEO

3.1 BINNING THE RODEO

Before proceeding with the details of the fast RODEO, let us briefly return to Figure 4 that provided an illustration of the RODEO solution path from the scale-space perspective. Recall the scale-space surface shown here is simply a sequence of standard, fixed bandwidth LPR smooths of the data, placed in order from smallest to largest bandwidth. From this perspective, the test statistics \( Z(3) \) and \( V(5) \) are derived from the partial derivative along the bandwidth-axis.

In a similar fashion, the fast RODEO approach uses binning to rapidly compute the scale-space surface and the corresponding test statistics \( Z \) and \( V \).
With this in mind, we now provide details on the derivation of the fast RODEO. We begin by introducing some notation; let

\[ T_{2,j} = X^T W_{L_j} Y, \quad S_2 = X^T W^2 X, \quad S_{3,j} = X^T W L_{j} X, \quad S_{4,j} = X^T W^2 L_{j} X, \quad S_{5,j} = X^T W^2 L_{j}^2 X. \] (10)

It can be shown that \( Z_j \) in (3) can be written as

\[ Z_j = e_1^T S_1^{-1} X^T \frac{\partial W}{\partial h_j} (I - X S_1^{-1} X^T W) Y \]
\[ = e_1^T S_1^{-1} X^T W L_j (I - X S_1^{-1} X^T W) Y \]
\[ = G_j^T Y, \] (11)

where \( \frac{\partial W}{\partial h_j} = W_{L_j} \), \( L_j = \text{diag}(\frac{\partial \log K((X_{1,j} - x_j)/h_j)}{\partial h_j}, \ldots, \frac{\partial \log K((X_{n,j} - x_j)/h_j)}{\partial h_j}) \), and \( G_j = (G_j(X_1, x, H), \ldots, G_j(X_n, x, H))^T \). From (11), the variance \( V_j^2 \) in (5) is calculated as

\[ V_j^2 = \text{var}(G_j^T Y) \]
\[ = G_j^T \{ E[\epsilon \epsilon^T] \} G_j \]
\[ = \sigma^2 G_j^T G_j. \] (12)

Expanding (11) and (12) and using the definitions of the matrices in (10) we have

\[ Z_j = e_1^T S_1^{-1} (T_{2,j} - S_{2,j} S_1^{-1} T_1), \]
\[ V_j^2 = \sigma^2 e_1^T S_1^{-1} (2S_{4,j} S_1^{-1} S_{3,j} - S_{5,j} - S_{3,j} S_1^{-1} S_2 S_1^{-1} S_{3,j}) S_1^{-1} e_1. \] (13)

Of particular interest is that the elements of the matrices in (13) can be represented in a similar form as \( \hat{t}_{1,k} \) and \( \hat{s}_{1,k} \) in (7). Specifically,

\[ \hat{t}_{2,k,j} = T_{2,j}(k) = \sum_{i=1}^{n} (X_i - x)^{k+2e_j} K_H(X_i - x) Y_i, \]
\[ \hat{s}_{2,k} = S_2(k) = \sum_{i=1}^{n} (X_i - x)^k K_H(X_i - x)^2, \]
\[ \hat{s}_{3,k,j} = S_{3,j}(k) = \sum_{i=1}^{n} (X_i - x)^{k+2e_j} K_H(X_i - x), \]
\[ \hat{s}_{4,k,j} = S_{4,j}(k) = \sum_{i=1}^{n} (X_i - x)^{k+2e_j} K_H(X_i - x)^2, \]
\[ \hat{s}_{5,k,j} = S_{5,j}(k) = \sum_{i=1}^{n} (X_i - x)^{k+4e_j} K_H(X_i - x)^2. \] (14)
Here, \( e_j \) is a \( d \)-dimensional vector with the \( j \)th element equal to 1 and 0’s elsewhere. Using the notation in (14), for LLR the matrices in (10) are expressed as

\[
\mathbf{T}_{2,:j} = \left( \tilde{t}_{2,0,0}, \tilde{t}_{2,1,0}, \tilde{t}_{2,0,1} \right), \quad \mathbf{S}_2 = \begin{pmatrix} \hat{s}_{2,0,0} & \hat{s}_{2,1,0} & \hat{s}_{2,0,1} \\ \hat{s}_{2,1,0} & \hat{s}_{2,2,0} & \hat{s}_{2,1,1} \\ \hat{s}_{2,0,1} & \hat{s}_{2,1,1} & \hat{s}_{2,0,2} \end{pmatrix}, \quad \mathbf{S}_l,j = \begin{pmatrix} \hat{s}_{l,(0,0),j} & \hat{s}_{l,(1,0),j} & \hat{s}_{l,(0,1),j} \\ \hat{s}_{l,(1,0),j} & \hat{s}_{l,(2,0),j} & \hat{s}_{l,(1,1),j} \\ \hat{s}_{l,(0,1),j} & \hat{s}_{l,(1,1),j} & \hat{s}_{l,(0,2),j} \end{pmatrix},
\]

for \( j = 1, \ldots, d \) and \( l = 3, 4, 5 \).

Following the discussion in Section 2.2, the quantities in (14) can be expressed in terms of their binned approximations

\[
\tilde{t}_{2,:j} = \tilde{\mathbf{T}}_{2,:j}(\mathbf{k}) = \sum_{l_i=-L_1}^{L_1} \cdots \sum_{l_d=-L_d}^{L_d} d_{l_{-1}} \kappa_{k,1,l,j}^2, \quad \tilde{s}_{2,:j} = \tilde{\mathbf{S}}_{2,:j}(\mathbf{k}) = \sum_{l_i=-L_1}^{L_1} \cdots \sum_{l_d=-L_d}^{L_d} c_{j-1} \kappa_{k,1,j}^3,
\]

\[
\tilde{t}_{3,:j} = \tilde{\mathbf{T}}_{3,:j}(\mathbf{k}) = \sum_{l_i=-L_1}^{L_1} \cdots \sum_{l_d=-L_d}^{L_d} \sum_{l_j=-L_d}^{L_d} c_{j-1} \kappa_{k,1,j}^4, \quad \tilde{s}_{3,:j} = \tilde{\mathbf{S}}_{3,:j}(\mathbf{k}) = \sum_{l_i=-L_1}^{L_1} \cdots \sum_{l_d=-L_d}^{L_d} \sum_{l_j=-L_d}^{L_d} \sum_{l_j=-L_d}^{L_d} c_{j-1} \kappa_{k,1,j}^5,
\]

(15)

where

\[
\kappa_{k,1,j}^2 = \prod_{i=1}^{d} K_{h_l}(l_i \delta_i)(l_i \delta_i)^{k_i+2 e_i}, \quad \kappa_{k,1,j}^3 = \prod_{i=1}^{d} K_{h_l}(l_i \delta_i)(l_i \delta_i)^{k_i},
\]

\[
\kappa_{k,1,j}^4 = \prod_{i=1}^{d} K_{h_l}(l_i \delta_i)(l_i \delta_i)^{k_i+2 e_i}, \quad \kappa_{k,1,j}^5 = \prod_{i=1}^{d} K_{h_l}(l_i \delta_i)(l_i \delta_i)^{k_i+4 e_i},
\]

and \( e_{ij} \) is the \( i \)th element of the vector \( e_j \). Letting \( \tilde{\mathbf{T}}_{2,:j}, \tilde{\mathbf{S}}_{2,:j}, \text{ and } \tilde{\mathbf{S}}_{l,:j} \), \( j = 1, \ldots, d, l = 3, 4, 5 \), be the binned approximations of the matrices in (10) and plugging these into (13), we get the binned approximations of \( Z_{j} \) and \( V_{j} \), \( \tilde{Z}_{j} \), and \( \tilde{V}_{j} \). As was discussed in Section 2.2, the key observation here is that the quantities in (15) are simply the convolution of the counts \( d_{l_{-1}} \) and \( c_{j-1} \) with their respective kernels, \( \kappa_{k,1} \) and \( \kappa_{k,1,j} \), which can be quickly and efficiently computed using FFTs.

**Remark 3.1.** It is important to highlight the relationship between binning and the dimensionality of the data. While the framework we provide generalizes to an arbitrary number of dimensions, it must be kept in mind that as the number of dimensions increases, the number of grid points increases exponentially. Since the number of grid points will quickly become greater than the number of observations (more often than not), binning will no longer be a computationally efficient approach.

It is worth mentioning that the largest number of dimensions we have considered in practice is five. While not yet developed, as part of future work we are planning on building a primarily C++ driven version of the fast RODEO, which is to be implemented on five-dimensional image data, that is, spatial \((x, y, z)\), spectral \((\lambda)\), and temporal \((t)\). As these data are densely sampled, the binning-based fast RODEO remains a sensible and computationally efficient approach.
Remark 3.2. An additional point to note is the interplay between the scale of features in the data, the bandwidth, and grid size. As with any implementation of binning, careful considerations need to be made regarding grid size. If the grid is too coarse, we run the risk of masking features potentially present in the data and no choice of bandwidth will help in this situation; if the grid is too fine we increase the number of computations, reducing the efficiency of the algorithm.

In our implementations of the fast RODEO, the choice of grid size is primarily dictated by the features we believe to be present and the size of the dataset. In the HSRL example in Section 4.2, we used bins that were roughly \(5 \times 12\) (recall the dataset itself is approximately \(300 \times 20,000\)) since we did not believe that any of the features of interest would be smaller than this. In general, it is recommended that several grid sizes be considered and the results visualized before moving forward with any analysis.

As far as the effect of spatially nonuniform data, we have found that binning, in general, is not affected by any more or less than nonbinning-based methods (e.g., kriging). For an example of the use of the fast RODEO for interpolation (as well as kriging) on spatially nonuniform data, see Cambaliza et al. (in press).

Interesting work presented by Raykar, Duraiswami, and Zhao (2010) studies the problem of grid size selection and provides \(\epsilon\) exact estimates of the difference between binned and unbinned kernel estimators. This \(\epsilon\) error measure is then used as a criteria for selecting grid size. While not implemented here, in principle similar ideas could be extended to the fast RODEO.

3.2 The Fast RODEO Algorithm

In terms of implementation there are two primary differences between the RODEO algorithm as described in Section 2.1 and what we propose here: first, the range of candidate bandwidths, from largest to smallest is selected in advance and second, the number of bandwidths across this range over which to search, call this \(n_c\), must be specified. Once these values have been selected what follows is the same as in the standard RODEO algorithm. That is, for every \(t \in \{0, \ldots, n_c - 1\}\) and corresponding \(h_j(t) = \beta^t h_0^j, j = 1, \ldots, d\) along the solution path, we compute the binned approximation of the RODEO statistics \(\tilde{Z}_j(t)\) and \(\tilde{V}_j(t)\) for each point \(g\) on the grid for the entire solution path. The algorithm then proceeds as usual using these precomputed values. This is outlined in Figure 6.

In general, the following guidelines are suggested when selecting values for the parameters \(h_0^j, \beta,\) and \(n_c\). First, the initial bandwidth, \(h_0^j\) is selected so that the number of points in the neighborhood of the kernel function is less than or equal to the hypercube formed by the \(M_1 \times \cdots \times M_d\) grid, that is, (near) the entire space so that we are effectively performing a (global) linear regression. Next, the parameters \(\beta\) and \(n_c\) are selected so that the effective neighborhood of the kernel function associated with the smallest bandwidth \(\beta^{n_c-1} h_0\) in the RODEO solution path is greater than or equal to a single (or a small number of) point(s) on the grid.

The cost of computing the solution path of the fast RODEO is equal to \(O(5d n_c M_1 \log M_1 \times \cdots \times M_d \log M_d)\). This can be seen by recalling that the computational complexity of performing an FFT over an \(M_1 \times \cdots \times M_d\) grid is \(O(M_1 \log M_1 \times \cdots \times M_d \log M_d)\). For each of the five matrices in (10) and each dimension \(d\), the FFT of
1. Select a constant $0 < \beta < 1$, number of bandwidth points, $n_c$ to search over and initial bandwidth $h^0_j = c^0_j / \log \log n$.

2. Compute binned approximations $\hat{Z}^{(t)}_j$ and $\hat{V}^{(t)}_j$ for all $t \in \{0, \ldots, n_c - 1\}$.

3. Initialize the bandwidths, and activate all covariates:
   (a) $h_j = h^0_j$, $j = 1, \ldots, d$.
   (b) $\mathcal{A} = \{1, \ldots, d\}$.

4. While $\mathcal{A}$ is nonempty, do for each $j \in \mathcal{A}$:
   (a) Compute the threshold $\lambda_j = V_j \sqrt{2 \log n}$.
   (b) If $|\hat{Z}_j| > \lambda_j$, then set $h_j \leftarrow \beta h_j$; otherwise remove $j$ from $\mathcal{A}$.

5. Output bandwidths $\mathbf{H}^* = (h_1, \ldots, h_d)$ and estimator $\hat{m}_{\mathbf{H}^*}(x)$.

Figure 6. The fast RODEO algorithm.

the corresponding grids must be calculated. Putting this together, the resulting complexity for a fixed $t$ and bandwidth $h(t) = \beta^t h_0$ is $O((5dM_1 \log M_1 \times \cdots \times M_d \log M_d))$. Repeating this calculation for each $t \in \{1, \ldots, n_c\}$ gives the above complexity. Note that this is only an order $5dn_c$ larger than the corresponding binned implementation of LPR with a global bandwidth.

4. EXAMPLES

4.1 TOY EXAMPLE

The toy example we consider here provides a simple simulation of HSRL measurements in a two-dimensional setting (similar to those shown in Figures 1 and 2). One sample dataset from our simulation is shown in Figure 7. The plot on the left shows the ground truth, with each “hotspot” corresponding to a region of high concentration of a gas or aerosol. The plot on the right is the normalized running sum, going from top to bottom of the plot on the left with the addition of Gaussian noise (details below). Recall from Section 1, that the observed HSRL measurements correspond to the plot on the right, with the goal being to estimate the partial derivative of these data (along the $y$-direction) to retrieve the concentrations, shown in the plot on the left. The purpose of these simulations is to test the fast RODEO’s speed and accuracy as compared to a standard bandwidth selection approach (here we use the R package locfit; Loader 2012, which uses generalized cross-validation), and an approach that uses simple differencing (commonly used in the remote sensing community, see remark) across datasets of varying size and feature complexity.

Remark 4.1. The simple differencing approach to aerosol concentration estimation consists of two steps. To start, let $y_{i,j}$ be the raw value, with $i = 1, \ldots, n$ indexing the time stamp (i.e., the spatial location as discussed in Section 1 and illustrated in Figure 1), and $j = 1, \ldots, m$ the index of the current altitude. The first step is then to calculate $D_{ij} = (y_{i,j} - y_{i,j-1})$ (change in altitude), for $i = 1, \ldots, n$, $j = 2, \ldots, m$, where
Figure 7. An illustration of one sample from our simulation study. The figure on the left shows the ground truth “concentration” measurements, and the figure on the right shows the observed measurements.

\( D_{11} = D_{12} \). The second step is then to perform a LOWESS smooth (Cleveland 1981) on the \( D_{ij} \)’s across altitudes, that is, for each \( i \) smooth \( D_{ij}, j = 1, \ldots, m \). The output from this second step is the estimate of the aerosol concentration.

The model that we use for generating the simulated examples is a mixture of Gaussian densities, this is meant to mimic the “hot spots” often observed in real HSRL data. To begin, we define the covariance matrix, \( \Sigma \) used in this study. Letting \( \Sigma_1 = I_2, \Sigma_2 = 0.25I_2, \Sigma_3 = 0.5I_2, \Sigma_4 = \begin{pmatrix} 0.5 & 1 \\ 1 & 0.5 \end{pmatrix}, \Sigma_5 = \begin{pmatrix} -0.9 & 1 \\ 1 & -0.9 \end{pmatrix} \), we have

\[
\Sigma = \sum_{i=1}^{5} \pi_i \Sigma_i,
\]

where \( (\pi_1, \ldots, \pi_5) \sim \text{Mult}(1/5, 1/5, \ldots, 1/5) \), so that \( \Sigma \) is randomly selected from \( \Sigma_1, \ldots, \Sigma_5 \). These choices for the covariance matrix allow for hot spots of varying size and directionality.

Next, letting \( u_1, u_2 \sim \text{Unif}[-7, 7] \), the mean, that is, the location of the hot spot, is sampled as \( \mu = (u_1, u_2) \). Define \( G = [-8 + \frac{16}{n_g^2 - 1}i, -8 + \frac{16}{n_g^2 - 1}j]_{i,j=0}^{n_g^2 - 1} \) to be a regularly spaced, \( n_g \times n_g \) grid with \( n_g \) taken to be 400 or 800. We had also considered smaller and larger grid sizes, however, we found with grid sizes larger than 800 \( \times \) 800 that \texttt{locfit} began to run into memory issues, so we did not report those results (fast RODEO’s computational time remains approximately the same for 1200 \( \times \) 1200 grids).

Let \( f(x; \mu, \Sigma) = \text{Norm}(x; \mu, \Sigma) \) (\( \text{Norm}(x; \mu, \Sigma) \)) denotes the Gaussian density, evaluated at \( x \), with mean \( \mu \) and covariance \( \Sigma \), \( f_{ij} = f(G_{ij}; \mu, \Sigma) \) the evaluation of \( f \) at the \( ij \)th point in \( G \) and \( F = \{ f_{ij} \}_{i,j=1}^{n_g^2} \) the associated grid. If \( F_k \) is one realization of \( F \) for a given \( \mu, \Sigma, \) and \( n_g, \) and \( s_k \sim \text{Unif}[10, 20], \) iid (where \( \text{Unif}[a, b] \) denotes the uniform distribution of \([a, b]\)), then \( \mathcal{F} = \sum_{k=1}^{n_f} s_k F_k \) corresponds to one random sample from our simulation, an example of which is shown in the left plot of Figure 7 (i.e., these are the true, unobserved concentrations we are trying to estimate). The parameter \( n_f \) is the number
of “features,” taken to be 20 or 40, allowing for varying levels of complexity and $s_k$ is the relative magnitude of the sampled feature. An example with $n_f = 3$ is shown in Figure 8 ($s_k$ set to 1).

Note, each $F$ has associated with it an $n_g$, $n_f$ pair. As the calculations are effectively the same regardless of the choice of these parameters, we do not include them in the notation.

Once $F$ has been sampled, the normalized running sum is calculated, from top to bottom as $Y_{ij} = \frac{16}{n_g-1} \sum_{k=1}^{t} F_{n_g+1-k,j}$, for $t, j = 1, \ldots, n_g$, with $Y = \{Y_{ij}\}_{t,j=1}^{n_g}$. Putting this all together, the points that make up the observed data in our simulations are $Y^* = Y + \mathcal{E}$, where $\mathcal{E} = \{\epsilon_{ij}\}_{t,j=1}^{n_g}$, $\epsilon \sim \text{Norm}(0, \sigma)$, $\sigma = 1/2$ (similar experiments were run using 0.25 and 1, yielding similar results).

To measure algorithm performance, we calculate the mean square error (MSE) between the actual values, $F_{ij}$ and the predicted values, $\hat{m}_F^{(y)}(G_{ij})$, $\hat{m}_S^{(y)}(G_{ij})$, and $\hat{m}_D^{(y)}(G_{ij})$, where $F$ is the fast RODEO, $S$ is the plug-in bandwidth approach, and $D$ is simple differencing. The superscript $(y)$ denotes the partial derivative with respect to $y$ (as this is the quantity of interest in HSRL studies). The estimated fits from one of the simulations (the corresponding

Figure 8. An illustration of one sampling of $F$. 
raw data are shown in the right-hand plot of Figure 7) are shown in Figure 9. The MSE for a single realization from our simulation is then calculated as \( \text{MSE}_M = \frac{1}{n_g} \sum_{i,j=1}^{n_g} (F_{ij} - m_M(G_{ij}))^2 \), where \( M \) denotes the method used. For a given number of features, \( n_f \) and grid size \( n_g \), we generate 10 examples and for each example we randomly sample the Gaussian error matrix \( \mathcal{E} \), 10 times. Thus, for a given \( n_f, n_g \) pair, we have 100 simulated samples. Overall performance is then calculated as \( \text{MSE}_{\text{total}} = \frac{1}{100} \sum_{k=1}^{100} \text{MSE}_k \), where the \( \text{MSE}_k \)'s correspond to the MSEs for a given \( n_f, n_g, \) and \( G^* \). The standard errors across the \( \text{MSE}_k \)'s were also calculated. For algorithm speed, we computed the average time and standard deviation, in seconds for each \( n_f \) and \( n_g \). The results of the simulations are shown in Figures 10 and 11. In each figure, the points correspond to the average MSE across the simulations and the whiskers two standard deviations.

Note that in practice, to obtain better estimates with simple differencing it is common to take a running average of the HSRL measurements over 1 min time intervals (corresponding to 120 measurements) and perform the differencing on the average. The purpose of doing this is to average out noise, which may be present in a single measurement. For this toy example, we found the optimal time range to average over that minimized MSE was 10 and 20 for \( n_g \) that is equal to 400 and 800, respectively. Of course in practice ground truth would (most likely) not be available so this represents approximately the best possible performance for this method.

In each case, we see that the fast RODEO performs better, in terms of MSE than standard LPR, and both generally outperform simple differencing. Additionally, the standard error of MSEs is much smaller for the fast RODEO than both other approaches, showing its consistent performance across simulations. In terms of speed as we go from \( n_g = 400 \) to \( n_g = 800 \), the fast RODEO has over a 5-fold speed up over standard LPR (with simple differencing outperforming both). It is worth noting that the fast RODEO code is almost exclusively implemented in R whereas the standard LPR code is primarily written in C. As mentioned earlier, memory issues arose with matrices over \( 800 \times 800 \) for standard LPR, however, for the successful runs we did have at \( 1200 \times 1200 \) the exponential increase in compute time continued for the standard LPR while the fast RODEO remained about the same.

4.2 APPLICATION TO THE CALIPSO HSRL MEASUREMENTS

Next we apply the fast RODEO and simple differencing to real HSRL measurements collected in the CALIPSO study discussed in Section 1. We did not include results for
Figure 10. MSE results from the toy example for each $n_f$ and $n_g$ pair.

Figure 11. Speed results from the toy example for each $n_f$ and $n_g$ pair.
standard LPR due to memory issues. It is important to emphasize that an absolute ground truth is not available for these results, so the discussion here is largely qualitative. The results using the fast RODEO and differencing are shown in Figures 12 and 13.

In terms of overall trend, both approaches yield similar predictions of aerosol concentration. The major difference is that the estimates from the fast RODEO suggest that many small-scale features may be present, while the differencing approach tends to smooth these
out. As mentioned above, while we have no way to absolutely confirm the presence of these regions, from the simulation results in Figure 9 we see that the differencing approach (middle plot) does have a tendency to smooth out finer features actually present in the image (compare to ground truth in the left-hand plot of Figure 7).

The potential “loss” of features is illustrated in Figure 14; here we have zoomed in on the region falling between 500 and 1200 m and 5.6 and 5.9 h for both the fast RODEO (top) and simple differencing (bottom). In the fast RODEO, we can see a distinct series of hot spots, while in the simple differencing approach these features are largely smoothed out. Here, the advantage of using the fast RODEO is that if these hot spots are real, they will be completely missed by simple differencing.

Of course, the number of time points being averaged over in the differencing approach may be causing some of this feature loss. The challenge is that if we reduce the number of points being averaged over, then we risk increasing the variability of the estimates in regions where no discernible features are present. The effect is that rather than drawing the scientists’ eye to regions more likely to contain signal of interest, they now have many (possibly) irrelevant regions to look through.

**SUPPLEMENTARY MATERIALS**

The supplementary material contains details on the selection of the variance parameter $\sigma$ as well as a scale-space based interpretation of the fast RODEO algorithm.

[Received January 2013. Revised June 2014.]
REFERENCES

Cambaliza, M. O., Shepson, P. B., Caulton, D., Stirm, B., Samarov, D., Gurney, K., Turnbull, J., Davis, K. J., Possolo, A., Karion, A., Sweeney, C., Moser, B., Hendricks, A., Lauvaux, T., Mays, K., Whetstone, J., Huang, J., Razlivanov, I., Miles, N. L., and Richardson, S. J. (2014), “Assessment of Uncertainties of an Aircraft-Based Mass Balance Approach for Quantifying Urban Greenhouse Gas Emissions,” *Atmospheric Chemistry and Physics*, 14, 9029–9050. [1044]

Chaudhuri, P., and Marron, J. (2000), “Scale Space View of Curve Estimation,” *The Annals of Statistics*, 28, 408–428. [1038]

Cleveland, W. (1981), “Lowess: A Program for Smoothing Scatterplots by Robust Locally Weighted Regression,” *The American Statistician*, 35, 54. [1046]

Fan, J., and Gijbels, I. (1996), *Local Polynomial Modelling and Its Applications*, London: Chapman & Hall. [1034]

Fan, J., and Marron, J. (1994), “Fast Implementation of Nonparametric Curve Estimators,” *Journal of Computational and Graphical Statistics*, 3, 35–56. [1035,1039]

Godtliebsen, F., Marron, J., and Chaudhuri, P. (2002), “Significance in Scale Space for Bivariate Density Estimation,” *Journal of Computational and Graphical Statistics*, 11, 1–21. [1038]

Hall, P., and Wand, M. P. (1996), “On the Accuracy of Binned Kernel Density Estimators,” *Journal of Multivariate Analysis*, 56, 165–184. [1040]

Holmström, L. (2010), “Scale Space Methods,” *WIREs Computational Statistics*, 2, 150–159. [1038]

Jones, M., Marron, J., and Sheather, S. (1996), “A Brief Survey of Bandwidth Selection for Density Estimation,” *Journal of the American Statistical Association*, 91, 401–407. [1034]

Lafferty, J., and Wasserman, L. (2008), “Rodeo: Sparse, Greedy Nonparametric Regression,” *The Annals of Statistics*, 36, 28–63. [1035,1037]

Lindeberg, T. (1994), *Scale-Space Theory in Computer Vision*, Norwell, MA: Kluwer Academic Publishers. [1038]

Loader, C. (2012), *locfit: Local Regression, Likelihood and Density Estimation*, R package version 1.5-7. Available at [http://CRAN.R-project.org/package=locfit]. [1045]

Polzehl, J., and Spokoiny, V. (2000), “Adaptive Weights Smoothing With Applications to Image Restoration,” *Journal of the Royal Statistical Society, Series B*, 60, 335–354. [1035]

——— (2006), “Propagation-Separation Approach for Local Likelihood Estimation,” *Probability Theory Related Fields*, 135, 335–363. [1035]

Raykar, V., Duraiswami, R., and Zhao, L. (2010), “Fast Computation of Kernel Estimators,” *Journal of Computational and Graphical Statistics*, 19, 205–220. [1044]

Rogers, R., Hair, J., Hostetler, C., Ferrare, R., Obland, M., Cook, A., Harper, D., Burton, S., Shinozuka, Y., McNaughton, C., Clarke, A., Redemann, J., Russell, P., Livingston, J., and Kleinman, L. (2009), “Nasa Larc Airborne High Spectral Resolution Lidar Aerosol Measurements During Milagro: Observations and Validation,” *Atmospheric Chemistry and Physics*, 9, 4811–4826. [1035]

Wand, M. (1994), “Fast Computation of Multivariate Kernel Estimators,” *Journal of Computational and Graphical Statistics*, 3, 433–445. [1035,1039]