SMOOTHING SPLINE DENSITY ESTIMATION: A DIMENSIONLESS AUTOMATIC ALGORITHM

by

Chong Gu
Purdue University

Technical Report #91-41

Department of Statistics
Purdue University

July 1991

◇

Smoothing Spline Density Estimation: A Dimensionless Automatic Algorithm

CHONG GU*

Abstract

As a sequel to an earlier article by Gu and Qiu (1991), this article describes and illustrates a dimensionless automatic algorithm for nonparametric probability density estimation using smoothing splines. The algorithm is designed to calculate an adaptive semiparametric solution to the penalized likelihood problem, which was shown by Gu and Qiu (1991) to share the same asymptotic convergence rates as the nonadaptive infinite dimensional solution. The smoothing parameter is updated jointly with the estimate in a performance-oriented iteration via a cross-validation performance estimate, where the performance is measured by a proxy of the symmetrized Kullback-Leibler distance between the true density and the estimate. Simulations of limited scale are conducted to examine the effectiveness of the technique. The method is also applied to some real data sets. The algorithm is implemented in a few Ratfor routines which will be included in later versions of RKPACK (Gu 1989).

KEY WORDS: Cross-validation; Density estimation; Kullback-Leibler; Penalized likelihood; Performance-oriented iteration; Smoothing splines.

*Chong Gu is Assistant Professor, Department of Statistics, Purdue University, West Lafayette, IN 47907. The author thanks Grace Wahba for the access to a DECstation-3100 in the Department of Statistics, University of Wisconsin—Madison, on which the computation was conducted. He also thanks Chunfu Qiu for helpful discussions. This research was supported by National Science Foundation under Grant DMS-9101730.

1 Introduction

Probability density estimation is a fundamental problem in theoretical and applied statistics. Given samples drawn from an unknown density f, the classical parametric estimation assumes that f belongs to a parametric family known up to a few parameters, and the common practice is to estimate the unknown parameters from the samples via the maximum likelihood (ML) method; see any standard textbook, e.g., Bickel and Doksum (1977). When prior knowledge about f is not sufficient to justify a parametric family, however, nonparametric methods have to be employed. A recent review of nonparametric density estimation can be found in Silverman (1986). See also Tapia and Thompson (1978). It appears that the existing practical nonparametric density estimators are by and large univariate, with the exception of the work by Scott (1985).

This article follows the penalized likelihood approach to density estimation pioneered by Good and Gaskins (1971). In the closely related context of nonparametric regression, penalty smoothing, or smoothing spline, has emerged as one of the most successful multivariate methods available; see, e.g., Wahba (1990) and Gu and Wahba (1991b) for a recent review. In the density estimation context, however, two intrinsic constraints, the positivity constraint that a density must be positive and the unity constraint that a density must integrate to one, have to be enforced. Once these intrinsic constraints are conveniently taken care of, one may expect a similar success of the smoothing spline technique in density estimation. In this article, I shall provide positive evidence to suggest that it is indeed a promising prospect.

The idea of Good and Gaskins (1971) is to estimate the density f by the minimizer of a penalized likelihood score $L(f) + \lambda J(f)$, where the $L(f)$ is a goodness-of-fit measure, usually taken as the minus log likelihood, the $J(f)$ is a roughness penalty, and the λ is the smoothing parameter controlling the trade-off between the two conflicting goals. Good and Gaskins (1971) took $J(f)$ as a quadratic functional in $\sqrt{f}$, which takes care of the positivity constraint but leaves the unity constraint to the numerical problem. Leonard (1978) proposed the logistic density transform $f = e^g / \int e^g$ and worked with g which is free of both constraints, but the many-to-one feature of the transform in usual function spaces introduces extra theoretical and computational inconvenience. Silverman (1982) proposed to estimate $g = \log f$ which is free of the positivity constraint, and to augment the penalized likelihood score by a term $\int e^g$ to effectively enforce the unity constraint when $J(f)$ is a quadratic functional in derivatives of $g = \log f$. O'Sullivan (1988) developed an

algorithm to calculate Silverman's (1982) estimator using B-spline approximations in one dimension. Asymptotic convergence rates are established by Klonias (1982) for the Good-Gaskins $\sqrt{f}$-based penalties and by Silverman (1982) for the Leonard-Silverman $\log f$-based penalties. In a recent article by Gu and Qiu (1991), a simple surgery on the usual function spaces was proposed to make the logistic density transform one-to-one, an asymptotic theory was developed in parallel to that of Silverman (1982), and an adaptive semiparametric estimator was proposed and justified, which paved the way for the current development. The development of Gu and Qiu (1991) is in a dimensionless generic setup and hence the current development is dimensionless.

The choice of the smoothing parameter λ has major impact on the performance of the resulting estimate. In the context of kernel density estimation, various cross-validation schemes have been developed to automatically select the smoothing parameter there, see, e.g., Scott (1986) for a review and references. Wahba (1977) developed a generalized cross-validation method for use in the context of orthogonal series density estimation. O'Sullivan (1988) adapted a certain cross-validation score in the kernel method literature to choose λ in the calculation of Silverman's (1982) estimator. In this article, I shall propose and examine a somewhat different approach to automatic smoothing parameter selection, which nevertheless is influenced by the earlier developments.

The remaining of the article is organized as follows. In Section 2, I briefly review the basic idea and a few facts in the theoretical development of Gu and Qiu (1991) and set up the numerical problem. In Section 3, a performance-oriented iteration scheme is proposed to calculate a density estimate with a proper choice of smoothing parameter, and a cross-validation performance estimate is derived for use in the iteration. In Section 4, an algorithm is proposed to carry out the computation. Simulation results of limited scale are presented in Section 5 to illustrate the effectiveness of the procedure proposed in Section 3 and to examine various practical aspects of the algorithm proposed in Section 4. A few real data examples are presented in Section 6. Section 7 concludes the article with a few remarks on the implication of the current development and possible future topics.

2 Formulation and Preliminaries

2.1 Theoretical background

Let X_i, $i = 1, \cdots, n$, be independent and identically distributed random samples from a probability density f on a finite domain $\mathcal{X}$. The goal is to estimate f from the data X_i. Assuming $f > 0$ on $\mathcal{X}$, one can make a logistic density transform (Leonard 1978) $f = e^g / \int_{\mathcal{X}} e^g$ and estimate g instead, which is free of the positivity and unity constraints. To make the transform one-to-one, Gu and Qiu (1991) propose to impose a side condition on g, such as $g(x_0) = 0$, $x_0 \in \mathcal{X}$, or $\int_{\mathcal{X}} g = 0$. A smoothing spline density estimator is then defined as the minimizer of a penalized log likelihood score

$$\frac{1}{n}\sum_{i=1}^{n} g(X_i) + \log \int_{\mathcal{X}} e^g + \frac{\lambda}{2} J(g) \tag{2.1}$$

in a function space $\mathcal{H}$, where J is a roughness penalty and λ is the smoothing parameter. The space $\mathcal{H}$ is taken as a Hilbert space in which evaluation is continuous so that the first term of (2.1) is continuous, and the members of $\mathcal{H}$ have to comply with a side condition mentioned above to make the second term of (2.1) strictly convex. The roughness penalty J is taken as a square seminorm in $\mathcal{H}$ with a null space $J_\perp$ of finite dimension M. A finite dimensional $J_\perp$ avoids interpolation and a quadratic penalty renders theoretical and numerical simplicity.

A Hilbert space in which evaluation is continuous is called a reproducing kernel Hilbert space (RKHS) possessing a reproducing kernel (RK) $R(\cdot, \cdot)$, a positive definite bivariate function on $\mathcal{X}$, such that $R(x, \cdot) = R(\cdot, x) \in \mathcal{H}$, $\forall x \in \mathcal{X}$, and $< R(x, \cdot), g(\cdot) >= g(x)$ (the reproducing property), $\forall g \in \mathcal{H}$, where $< \cdot, \cdot >$ is the inner product in $\mathcal{H}$. As a matter of fact, starting from any positive definite function $R(\cdot, \cdot)$ on the domain $\mathcal{X}$, one can construct a RKHS $\mathcal{H} = \text{span}\{R(x, \cdot), \forall x \in \mathcal{X}\}$ with an inner product satisfying $< R(x, \cdot), R(y, \cdot) >= R(x, y)$, which has $R(\cdot, \cdot)$ as its RK. The RK and the inner product in a RKHS determine each other uniquely. Details can be found in Aronszajn (1950). With J as a square seminorm, $\mathcal{H}$ can be decomposed as $\mathcal{H} = \mathcal{H}_J \oplus J_\perp$, where $\mathcal{H}_J = \{g : J(g) \in (0, \infty)\}$ is a RKHS with the square norm J. Denote the RK of $\mathcal{H}_J$ as R_J. Note that the norm in $J_\perp$ does not appear in the definition of (2.1), so the smoothing spline density estimator is solely determined by R_J, a basis of $J_\perp$, say $\{\phi_\nu\}_{\nu=1}^{M}$, and the smoothing parameter λ.

As a specific example, consider the cubic spline estimation on $\mathcal{X} = [0, 1]$ with $J(g) = \int_0^1 \ddot{g}^2$. Different side conditions lead to different R_J and $J_\perp$, although the final density estimate after the

transform $e^g/\int e^g$ remains the same. Two examples follow. If one specifies $\int g = 0$, then $J_\perp = \{(\cdot - .5)\}$ and $R_J(x,y) = k_2(x)k_2(y) - k_4(|x-y|)$, where $k_2 = (k_1^2 - 1/12)/2$, $k_4 = (k_1^4 - k_1^2/2 + 7/240)/24$, and $k_1 = (\cdot - .5)$. If one specifies $g(0) = 0$, then $J_\perp = \{(\cdot)\}$ and $R_J(x,y) = \int_0^1 (x-u)_+(y-u)_+ du$, where $(\cdot)_+$ is the positive part of $(\cdot)$. More discussion and examples can be found in Gu and Qiu (1991). See also Section 4.

2.2 Numerical problem

The space $\mathcal{H}$ is usually infinite dimensional, and the minimizer of (2.1) in $\mathcal{H}$ is in general not computable. To remedy this problem, Gu and Qiu (1991) proposed to calculate the minimizer of (2.1) in an adaptive finite dimensional space $\mathcal{H}_n = J_\perp \oplus \{R_J(X_i,\cdot), i = 1,\cdots,n\}$, and proved under certain conditions that such an adaptive solution shares the same asymptotic convergence rates as the solution in $\mathcal{H}$. Based on this result, the current development seeks to calculate the minimizer of (2.1) in $\mathcal{H}_n$ for an appropriately selected λ.

Write $\xi_i = R_J(X_i,\cdot)$. By definition, a function in $\mathcal{H}_n$ has an expression

$$g = \sum_{i=1}^{n} c_i\xi_i + \sum_{\nu=1}^{M} d_\nu\phi_\nu = \boldsymbol{\xi}^T\boldsymbol{c} + \boldsymbol{\phi}^T\boldsymbol{d}, \tag{2.2}$$

where $\boldsymbol{\xi}$ and $\boldsymbol{\phi}$ are vectors of functions and $\boldsymbol{c}$ and $\boldsymbol{d}$ are vectors of coefficients. Substituting (2.2) into (2.1), note that

$$J(g) = <\sum_{i=1}^{n} c_i\xi_i\,,\,\sum_{j=1}^{n} c_j\xi_j> = \sum_{i=1}^{n}\sum_{j=1}^{n} c_ic_jR_J(X_i,X_j), \tag{2.3}$$

the numerical problem becomes to minimize

$$A_\lambda(\boldsymbol{c},\boldsymbol{d}) = -\frac{1}{n}\mathbf{1}^T(Q\boldsymbol{c} + S\boldsymbol{d}) + \log\int_{\mathcal{X}}\exp(\boldsymbol{\xi}^T\boldsymbol{c} + \boldsymbol{\phi}^T\boldsymbol{d}) + \frac{\lambda}{2}\boldsymbol{c}^TQ\boldsymbol{c}, \tag{2.4}$$

with respect to $\boldsymbol{c}$ and $\boldsymbol{d}$, where Q is a $n\times n$ matrix with (i,j)th entry $\xi_i(X_j) = R_J(X_i,X_j)$ and S is a $n\times M$ matrix with (i,ν)th entry $\phi_\nu(X_i)$.

To minimize (2.4) with a fixed smoothing parameter λ, a standard practice is to apply the Newton iteration. Let $\mu_g(h)$ be the mean of h calculated under the density $e^g/\int e^g$, $V_g(f,h)$ be the covariance of f and h under the density $e^g/\int e^g$, and $V_g(h) = V_g(h,h)$. Write $\tilde{g} = \boldsymbol{\xi}^T\tilde{\boldsymbol{c}} + \boldsymbol{\phi}^T\tilde{\boldsymbol{d}}$ as the current iterate of g. It can be verified that $\partial A_\lambda/\partial\boldsymbol{c}|_{\tilde{g}} = -Q\mathbf{1}/n + \mu_{\tilde{g}}(\boldsymbol{\xi}) + \lambda Q\tilde{\boldsymbol{c}} = -Q\mathbf{1}/n + \mu_\xi + \lambda Q\tilde{\boldsymbol{c}}$, $\partial A_\lambda/\partial\boldsymbol{d}|_{\tilde{g}} = -S^T\mathbf{1}/n + \mu_{\tilde{g}}(\boldsymbol{\phi}) = -S^T\mathbf{1}/n + \mu_\phi$, $\partial^2 A_\lambda/\partial\boldsymbol{c}\partial\boldsymbol{c}^T|_{\tilde{g}} = V_{\tilde{g}}(\boldsymbol{\xi},\boldsymbol{\xi}^T) + \lambda Q = V_{\xi,\xi} + \lambda Q$,

$\partial^2 A_\lambda/\partial \boldsymbol{d}\partial \boldsymbol{d}^T|_{\tilde{g}} = V_{\tilde{g}}(\boldsymbol{\phi}, \boldsymbol{\phi}^T) = V_{\phi,\phi}$, and $\partial^2 A_\lambda/\partial \boldsymbol{c}\partial \boldsymbol{d}^T|_{\tilde{g}} = V_{\tilde{g}}(\boldsymbol{\xi}, \boldsymbol{\phi}^T) = V_{\xi,\phi}$. The Newton updating equation is thus

$$\begin{pmatrix} V_{\xi,\xi} + \lambda Q & V_{\xi,\phi} \\ V_{\phi,\xi} & V_{\phi,\phi} \end{pmatrix} \begin{pmatrix} \boldsymbol{c} - \tilde{\boldsymbol{c}} \\ \boldsymbol{d} - \tilde{\boldsymbol{d}} \end{pmatrix} = \begin{pmatrix} Q\mathbf{1}/n - \mu_\xi - \lambda Q\tilde{\boldsymbol{c}} \\ S^T\mathbf{1}/n - \mu_\phi \end{pmatrix}. \tag{2.5}$$

After rearranging terms, (2.5) becomes

$$\begin{pmatrix} V_{\xi,\xi} + \lambda Q & V_{\xi,\phi} \\ V_{\phi,\xi} & V_{\phi,\phi} \end{pmatrix} \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{d} \end{pmatrix} = \begin{pmatrix} Q\mathbf{1}/n - \mu_\xi + V_{\xi,g} \\ S^T\mathbf{1}/n - \mu_\phi + V_{\phi,g} \end{pmatrix}, \tag{2.6}$$

where $V_{\xi,g} = V_{\tilde{g}}(\boldsymbol{\xi}, \tilde{g})$ and $V_{\phi,g} = V_{\tilde{g}}(\boldsymbol{\phi}, \tilde{g})$.

3 Performance-Oriented Iteration

The performance of a smoothing spline estimate is largely determined by the choice of the smoothing parameter λ. In this section, I shall describe an iteration scheme which updates λ and g jointly according to a performance estimate. The numerical calculation of the quantities involved shall be described in Section 4. Its practical performance is simulated in Section 5. This procedure owes its motivation to Gu's (1990) self-voting generalized cross-validation for smoothing spline non Gaussian regression.

Let $L(g, g_0)$ be a loss function for estimating g_0 by g. Given the data drawn from $e^{g_0}/\int e^{g_0}$, our objective is to find an estimate g which delivers a small L. To calculate the minimizer of (2.4) for a fixed λ, one naturally iterates on (2.6). Now, noting that (2.6) actually defines a class of estimates with a variable λ, one may make better use of (2.6). Specifically, starting from the current iterate $\tilde{g}$, instead of calculating the next iterate based on any prespecified λ, one may choose a λ which delivers a small L among the class of estimates defined by (2.6) and calculate an update using such a λ. Such an iteration scheme tries to minimize the loss function L which is of direct interest, instead of the penalized likelihood score A_λ with fixed λ, which the Newton iteration is after. When such a performance-oriented iteration converges at λ_* and g_*, say, however, g_* is apparently the fixed point of the Newton iteration for minimizing A_{λ_*}, and hence is the minimizer of A_{λ_*}.

I choose $L(g, g_0) = \mu_{g_0}(g_0 - g) + \mu_g(g - g_0)$, the symmetrized Kullback-Leibler between g and g_0. Note that such a L is not easily computable for the class of estimates defined by (2.6) even with

a known g_0. So approximation (or modification) is necessary. By the mean value theorem, it can be shown that $L(g, g_0) = V_{g'}(g - g_0)$, where g' is a convex combination of g and g_0. As a first step, I replace $V_{g'}(g - g_0)$ by $V_{\tilde{g}}(g - g_0)$ where the density $e^{\tilde{g}} / \int e^{\tilde{g}}$ is the best current guess of the true density $e^{g_0} / \int e^{g_0}$. Since $V_g(h)$ is continuous in g, this approximation is good when g and $\tilde{g}$ are close to g_0. Now $V_{\tilde{g}}(g - g_0) = V_{\tilde{g}}(g) - 2V_{\tilde{g}}(g, \tilde{g}) + 2V_{\tilde{g}}(g, \tilde{g} - g_0) + C$, where C is a constant depending only on $\tilde{g}$ and g_0. Further, it can be shown that $\mu_{\tilde{g}}(g) - \mu_{g_0}(g) = V_{g''}(g, \tilde{g} - g_0) = V_{\tilde{g}}(g, \tilde{g} - g_0)(1 + o(1))$, where g'' is a convex combination of $\tilde{g}$ and g_0 and the $o(1)$ term is small when $\tilde{g}$ is close to g_0. Putting things together, I shall try to minimize a proxy of L,

$$L_{\tilde{g}}(g, g_0) = V_{\tilde{g}}(g)/2 - V_{\tilde{g}}(g, \tilde{g}) + \mu_{\tilde{g}}(g) - \mu_{g_0}(g),$$

in calculating an update from $\tilde{g}$ using (2.6) with a variable λ. Note that $L_{\tilde{g}}$ can also be considered as a proxy of $V_{g_0}(g - g_0)$, a weighted integrated mean square error with the true density as the weight function.

The first three terms of $L_{\tilde{g}}$ are readily computable, but the fourth term needs estimation. Assume that $V_{\phi,\phi}$ is nonsingular, which amounts to the existence of the ML estimate in $J_\perp$, and that Q is nonsingular, which is true with probability one when the domain $\mathcal{X}$ is continuous and the true density is finite. Define $H = V_{\xi,\xi} + \lambda Q$, $E = V_{\phi,\phi} - V_{\phi,\xi}H^{-1}V_{\xi,\phi}$, $\boldsymbol{u}_\xi = Q\mathbf{1}/n - \mu_\xi + V_{\xi,g}$, $\boldsymbol{u}_\phi = S^T\mathbf{1}/n - \mu_\phi + V_{\phi,g}$, $\boldsymbol{u}_{\phi|\xi} = \boldsymbol{u}_\phi - V_{\phi,\xi}H^{-1}\boldsymbol{u}_\xi$, $\boldsymbol{v}_\xi = V_{\xi,g} - \mu_\xi$, $\boldsymbol{v}_\phi = V_{\phi,g} - \mu_\phi$, and $\boldsymbol{v}_{\phi|\xi} = \boldsymbol{v}_\phi - V_{\phi,\xi}H^{-1}\boldsymbol{v}_\xi$, It can be shown that $\boldsymbol{d} = E^{-1}\boldsymbol{u}_{\phi|\xi}$ and $\boldsymbol{c} = H^{-1}(\boldsymbol{u}_\xi - V_{\xi,\phi}\boldsymbol{d})$, and that the estimate defined by (2.6) has an expression

$$\begin{aligned} g &= \boldsymbol{\xi}^T\boldsymbol{c} + \boldsymbol{\phi}^T\boldsymbol{d} \\ &= \boldsymbol{\xi}^T H^{-1}(Q\mathbf{1}/n) + \boldsymbol{\xi}^T H^{-1}\boldsymbol{v}_\xi + (\boldsymbol{\phi} - V_{\phi,\xi}H^{-1}\boldsymbol{\xi})^T E^{-1}\boldsymbol{u}_{\phi|\xi}. \end{aligned} \tag{3.1}$$

Straightforward calculation yields

$$\mu_{\tilde{g}}(g) - V_{\tilde{g}}(g, \tilde{g}) = -\boldsymbol{v}_\xi^T H^{-1}\boldsymbol{u}_\xi - \boldsymbol{v}_{\phi|\xi}^T E^{-1}\boldsymbol{u}_{\phi|\xi}. \tag{3.2}$$

Also straightforward but tedious calculation gives

$$V_{\tilde{g}}(g) = \boldsymbol{u}_\xi^T H^{-1}\boldsymbol{u}_\xi + \boldsymbol{u}_{\phi|\xi}^T E^{-1}\boldsymbol{u}_{\phi|\xi} - \lambda(\boldsymbol{u}_\xi - V_{\xi,\phi}E^{-1}\boldsymbol{u}_{\phi|\xi})^T H^{-1}QH^{-1}(\boldsymbol{u}_\xi - V_{\xi,\phi}E^{-1}\boldsymbol{u}_{\phi|\xi}). \tag{3.3}$$

To estimate $\mu_{g_0}(g)$, the only source of information is the empirical distribution of the data. For the last two terms of (3.1), sample means simply give

$$(Q\mathbf{1}/n)^T H^{-1}\boldsymbol{v}_\xi + (S^T\mathbf{1}/n - V_{\phi,\xi}H^{-1}Q\mathbf{1}/n)^T E^{-1}\boldsymbol{u}_{\phi|\xi}. \tag{3.4}$$

For the first term, care must be taken, noting that the naive sample mean is always positive. Writing

$$\mu_{g_0}(\boldsymbol{\xi}^T H^{-1} Q\mathbf{1}/n) = \frac{1}{n}\sum_{i=1}^{n} \mu_{g_0}(\boldsymbol{\xi}^T H^{-1}\boldsymbol{\xi}(X_i)),$$

it can be seen that the problem of the naive sample mean is the use of X_i itself in the estimation of $\mu_{g_0}(\boldsymbol{\xi}^T H^{-1}\boldsymbol{\xi}(X_i))$. Using the empirical distribution of the remaining $n-1$ data to estimate $\mu_{g_0}(\boldsymbol{\xi}^T H^{-1}\boldsymbol{\xi}(X_i))$ and adding them up, one obtains

$$\begin{aligned}
&\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i}\boldsymbol{\xi}(X_j)^T H^{-1}\boldsymbol{\xi}(X_i) \\
= \;& \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1}^{n}\boldsymbol{\xi}(X_j)^T H^{-1}\boldsymbol{\xi}(X_i) - \frac{1}{n(n-1)}\sum_{i=1}^{n}\boldsymbol{\xi}(X_i)^T H^{-1}\boldsymbol{\xi}(X_i) \\
= \;& \frac{n}{n-1}(Q\mathbf{1}/n)^T H^{-1}(Q\mathbf{1}/n) - \frac{1}{n(n-1)}\mathrm{trace}(QH^{-1}Q). \qquad (3.5)
\end{aligned}$$

Adding (3.2) to one-half of (3.3) and subtracting (3.4) and (3.5), the estimated $L_{\tilde{g}}(g, g_0)$ is given by

$$\begin{aligned}
\hat{L}_{\tilde{g}}(g, g_0) = \;& \frac{\mathrm{trace}(QH^{-1}Q)}{n(n-1)} - \frac{(Q\mathbf{1}/n)^T H^{-1}(Q\mathbf{1}/n)}{n-1} - \frac{\boldsymbol{u}_\xi^T H^{-1}\boldsymbol{u}_\xi + \boldsymbol{u}_{\phi|\xi}^T E^{-1}\boldsymbol{u}_{\phi|\xi}}{2} \\
& - \frac{\lambda(\boldsymbol{u}_\xi - V_{\xi,\phi}E^{-1}\boldsymbol{u}_{\phi|\xi})^T H^{-1}QH^{-1}(\boldsymbol{u}_\xi - V_{\xi,\phi}E^{-1}\boldsymbol{u}_{\phi|\xi})}{2}. \qquad (3.6)
\end{aligned}$$

One may call (3.6) a cross-validation score since (3.5) is a cross-validation estimate of $\mu_{g_0}(\boldsymbol{\xi}^T H^{-1} Q\mathbf{1}/n)$. The performance-oriented iteration can then be conducted by minimizing (3.6) in each iteration.

4 Algorithm

In this section, I derive an algorithm to carry out the performance-oriented iteration of Section 3. The key issue is the efficient evaluation of (3.6) at multiple λ values.

First assume that Q is nonsingular. Let $Q = G^T G$ be the Cholesky decomposition of Q where G is upper triangular, and $G^{-T}V_{\xi,\xi}G^{-1} = UTU^T$ be the Householder tridiagonalization of $G^{-T}V_{\xi,\xi}G^{-1}$ where U is orthogonal and T is tridiagonal. It follows that $H^{-1} = G^{-1}U(T+\lambda I)^{-1}U^T G^{-T}$. Write $F = U^T G^{-T}Q = U^T G$, $B = U^T G^{-T}V_{\xi,\phi}$, $\tilde{\boldsymbol{u}}_\xi = U^T G^{-T}\boldsymbol{u}_\xi$, and $\tilde{\boldsymbol{q}} = U^T G^{-T}Q\mathbf{1}/n = U^T G\mathbf{1}/n$. It follows from (3.6) that

$$\hat{L}_{\tilde{g}}(g, g_0) = \frac{\mathrm{trace}(F^T(T+\lambda I)^{-1}F)}{n(n-1)} - \frac{\tilde{\boldsymbol{q}}^T(T+\lambda I)^{-1}\tilde{\boldsymbol{q}}}{n-1} - \frac{\tilde{\boldsymbol{u}}_\xi^T(T+\lambda I)^{-1}\tilde{\boldsymbol{u}}_\xi + \boldsymbol{u}_{\phi|\xi}^T E^{-1}\boldsymbol{u}_{\phi|\xi}}{2}$$

$$-\frac{\lambda(\tilde{\boldsymbol{u}}_\xi - BE^{-1}\boldsymbol{u}_{\phi|\xi})^T(T+\lambda I)^{-2}(\tilde{\boldsymbol{u}}_\xi - BE^{-1}\boldsymbol{u}_{\phi|\xi})}{2}, \tag{4.1}$$

where $E = V_{\phi,\phi} - B^T(T+\lambda I)^{-1}B$ and $\boldsymbol{u}_{\phi|\xi} = \boldsymbol{u}_\phi - B^T(T+\lambda I)^{-1}\tilde{\boldsymbol{u}}_\xi$.

I propose the following algorithm.

Algorithm 4.1 *Given data X_i and functions R_J and $\{\phi_\nu\}_{\nu=1}^M$, perform the following.*

1. Initialization*:*

   (a) *Calculate $Q = G^TG$ and collect $G\mathbf{1}/n$ and $S^T\mathbf{1}/n$.*

   (b) *Calculate the solution of (2.4) for a large fixed λ as the starting estimate $\tilde{g}$.*

2. Iteration*:*

   (a) *Collect all $\tilde{g}$-specific quantities in (2.6).*

   (b) *Calculate $G^{-T}V_{\xi,\xi}G^{-1} = UTU^T$.*

   (c) *Calculate $F = U^TG$, $B = U^TG^{-T}V_{\xi,\phi}$, $\tilde{\boldsymbol{u}}_\xi = U^T(G\mathbf{1}/n + G^{-T}\boldsymbol{v}_\xi)$, and $\tilde{\boldsymbol{q}} = U^TG\mathbf{1}/n$.*

   (d) *Search for a λ to minimize (4.1).*

   (e) *Update $\tilde{g}$ via $\boldsymbol{d} = E^{-1}\boldsymbol{u}_{\phi|\xi}$ and $\boldsymbol{c} = G^{-1}U(T+\lambda I)^{-1}(\tilde{\boldsymbol{u}}_\xi - B\boldsymbol{d})$.*

   (f) *Check convergence. If converged, terminate; otherwise, goto 2.(a).*

$Q = G^TG$ in step 1.(a) takes $n^3/6$ flops. Step 1.(b) can be carried out by executing step 2 for a fixed λ, with step 2.(d) and F and $\tilde{\boldsymbol{q}}$ in step 2.(c) omitted. I take $\tilde{g} = 0$ (the uniform distribution) as the starting estimate for step 1.(b), and fix the λ at the diagonal average of T evaluated at $\tilde{g} = 0$. Such a Newton iteration in step 1.(b) usually takes less than 10 steps to converge. It is necessary to calculate a starting value using step 1.(b) before entering the iteration of step 2, because (2.6) is the Newton iteration without a line search, which may diverge for small λ's from an arbitrary starting point. Step 2.(b) consists of two separate operations. To preserve the nonnegative definiteness of $G^{-T}V_{\xi,\xi}G^{-1}$ under rounding error, I calculate a Cholesky decomposition $V_{\xi,\xi} = D^TD$, solve for $G^{-T}D^T$, and multiply back, which in general takes $(5/3)n^3$ flops in total; saving is possible when $V_{\xi,\xi}$ is computationally singular. (Direct calculation takes $2n^3$ flops.) The Householder tridiagonalization in general takes $(2/3)n^3$ flops, and saving is possible via the distributed truncation of Gu *et al.* (1989). In total, step 2.(b) takes at most $(7/3)n^3$

flops. In step 2.(c), $F = U^T G$ in general takes n^3 flops and the remaining quantities all take $O(n^2)$ flops, and any saving in the Householder tridiagonalization is passed on to this step. In step 2.(d), each evaluation of (4.1) takes $O(n^2)$ flops due to the bandedness of $(T + \lambda I)$. Step 2.(e) takes another $O(n^2)$ flops. In total, the linear algebra calculations for each iteration takes at most $(10/3)n^3 + O(n^2)$ flops. Most of these calculations can be conducted using LINPACK facilities, and the flop counts are mainly based on the LINPACK manual (Dongarra et al. 1979). Steps 2.(a) and 2.(f) will be discussed later.

Now consider the case of a singular Q. With a continuous domain $\mathcal{X}$, a prespecified side condition for g, and an infinite precision arithmetic, a singular Q only occurs with probability 0. In practical situations, however, a convenient side condition could be data specific, data recording is subject to rounding, and of course the calculation can only be carried out to a finite precision, so the handling of a singular Q is of practical importance. With a possible reordering of the data index, called pivoting in LINPACK, the Cholesky factor G of a singular Q has a form

$$G = \begin{pmatrix} G_1 & G_2 \\ O & O \end{pmatrix},$$

where G_1 is nonsingular upper triangular. Let

$$\tilde{G} = \begin{pmatrix} G_1 & G_2 \\ O & \delta I \end{pmatrix},$$

where δ is some nonzero number. With an abuse of the notation, I shall also use G to denote its nontrivial portion (G_1, G_2), which also satisfies $Q = G^T G$. Partition $\tilde{G}^{-1} = (K, L)$. It follows that $GK = I$ and $GL = O$. Further, note that $J(g)$ is a norm in $\mathcal{H}_J$ and $J(\boldsymbol{\xi}^T \boldsymbol{c}) = \boldsymbol{c}^T Q \boldsymbol{c}$ (cf. (2.3)), it follows that $\boldsymbol{\xi}^T L = 0$, and hence $L^T \mu_\xi = L^T V_{\xi,g} = 0$, $V_{\xi,\xi} L = O$, and $V_{\phi,\xi} L = O$. Now it can be shown that (2.6) is equivalent to

$$\begin{pmatrix} K^T V_{\xi,\xi} K + \lambda I & O & K^T V_{\xi,\phi} \\ O & O & O \\ V_{\phi,\xi} K & O & V_{\phi,\phi} \end{pmatrix} \begin{pmatrix} G\boldsymbol{c} \\ \delta \boldsymbol{c}_2 \\ \boldsymbol{d} \end{pmatrix} = \begin{pmatrix} G\mathbf{1}/n - K^T \mu_\xi + K^T V_{\xi,g} \\ 0 \\ S^T \mathbf{1}/n - \mu_\phi + V_{\phi,g} \end{pmatrix}, \qquad (4.2)$$

where $\boldsymbol{c}_2 = (O, I)\boldsymbol{c}$ is the tail portion of the partitioned $\boldsymbol{c}$. This is an indefinite linear system due to the linear dependency of ξ_i, in which $\boldsymbol{c}_2$ could take any value. $\boldsymbol{\xi}^T \boldsymbol{c}$ is unique, however, and all that is needed is to calculate *a* solution of (4.2), which can be accomplished by replacing the center

piece of O in the left-hand-side matrix of (4.2) by λI. Such a procedure is readily implementable by simply replacing the G^{-T} and G^{-1} in steps 2.(b), 2.(c), and 2.(e) of Algorithm 4.1 by $\tilde{G}^{-T}$ and $\tilde{G}^{-1}$.

Observational data of large (and not that large) size often come in prebinned. Let m_i be the replicate counts of X_i, $\boldsymbol{m} = (m_1, \cdots, m_n)^T$, and $N = \sum_{i=1}^n m_i$. One may consider the replicates as rounded from distinct independent observations. It is easy to see that (2.6) remains good for prebinned data with $\boldsymbol{m}/N$ replacing $\mathbf{1}/n$. Formulas (3.1) through (3.4) are also valid after such a substitution. The counterpart of (3.5), however, can be shown to be

$$\frac{N}{N-1}(Q\boldsymbol{m}/N)^T H^{-1}(Q\boldsymbol{m}/N) - \frac{1}{N(N-1)}\text{trace}(Q_m^T H^{-1} Q_m),$$

where $Q_m = Q\,\text{diag}(m_1^{1/2}, \cdots, m_n^{1/2})$. Algorithm 4.1 is directly applicable after a few adjustments in the formulas.

Let us now turn to the calculation of the integrals μ_ξ, μ_ϕ, $V_{\xi,\xi}$, $V_{\phi,\phi}$, and $V_{\xi,\phi}$, which make up the derivatives in step 2.(a) of Algorithm 4.1. To ensure the numerical stability of a derivative-based algorithm like Algorithm 4.1, the evaluation of derivatives has to be highly accurate, say to at least seven digits. The numerical evaluation of integrals practical to the current problem, however, can hardly meet such requirement, especially in high dimensions. To overcome this difficulty, I propose the following solution. Note that in the original numerical problem (2.4), the integration measure in $\int e^g$ is not specified, although it is understood as uniform on the domain $\mathcal{X}$. From a numerical perspective, however, (2.4) is well-defined with any reasonable measure, say ν, appearing in the definition of the integral. When ν is a reasonable approximation of the uniform measure ν_0 for the purpose of $\int e^g$ in a neighborhood near the sought-after solution of (2.4) under ν_0, say $\hat{g}_0$, one may expect the solution $\hat{g}_\nu$ of (2.4) under ν to be a reasonable approximation of $\hat{g}_0$. Taking a discrete measure ν on a set of points on $\mathcal{X}$, call it an integration mesh, and driving the iteration under ν, one can virtually calculate the integrals appearing in Algorithm 4.1 under ν close to the machine precision. This strategy may be perceived as using a fixed quadrature rule for all the integrals involved, but no matter how poor the quadrature rule might be for approximating any or all of the integrals under ν_0, such a consistent measure replacement ensures the numerical stability of Algorithm 4.1. In practice the robustness of $\hat{g}_\nu$ under different ν may be simulated by calculating the estimate under a few different integration meshes.

In summary, the implementation of Algorithm 4.1 takes as inputs the RK R_J, the null space basis $\{\phi_\nu\}_{\nu=1}^M$, the data X_i possibly with replicate counts m_i, and an integration mesh with possible weights. The convergence is declared when consecutive iterates of e^g on the integration mesh differs by less than a prespecified precision requirement.

5 Simulation Results

In this section, two sets of simulations, one on $[0,1]$ and one on $[0,1]^2$, are conducted to examine the effectiveness of the technique developed. Applications to some real data sets will be presented in Section 6.

For the univariate simulation, I took a density on $[0,1]$ proportional to

$$\frac{1}{3}e^{-50(x-.3)^2} + \frac{2}{3}e^{-50(x-.7)^2}, \tag{5.1}$$

which is basically a mixture of $N(.3,.01)$ and $N(.7,.01)$. I used the cubic splines with the RK $R_J(x,y) = k_2(x)k_2(y) - k_4(|x-y|)$ and the null space basis $\phi(x) = x - .5$ (cf. Subsection 2.1). Fifty replicates of data of size $n = 100$ were drawn according to (5.1) using a pseudo random number generator. The algorithm of Section 4 was applied to the fifty cases to calculate the cross-validated smoothing spline density estimates. Fixed-λ solutions of (2.4) were also calculated on a grid $\log_{10}\lambda = (-7)(.2)(-3)$ for all the fifty replicates. The symmetrized Kullback-Leibler (SKL) $\mu_{g_0}(g_0 - g) + \mu_g(g - g_0)$, the Kullback-Leibler (KL) $\mu_{g_0}(g_0 - g) - \int e^{g_0} + \int e^g$, and the mean square error (MSE) $V_{g_0}(g - g_0)$ were collected for all these estimates. An integration mesh with 300 equally spaced points on $(0,1)$ was used in all the integrations involved, including the calculation of SKL, KL, and MSE.

For all the fifty replicates, the best λ's were covered by the span of the grid, so were all but one cross-validated λ. The best SKL, KL, and MSE on the grid were identified and were compared with those of the cross-validated fits. The left frame of Figure 5.1 gives the comparison of the 50 cross-validated SKL versus the corresponding best SKL on the grid. Two cases, the best and the worst cross-validated SKL, are marked differently from the other cases. A point on the dotted line indicates a perfect performance of the cross-validation. Corresponding plots for KL and MSE look similar. The efficacy of cross-validation in all the three scores, defined by the ratio of the minimum score on the grid over the score at the cross-validated fit, is summarized in a box-plot in the right

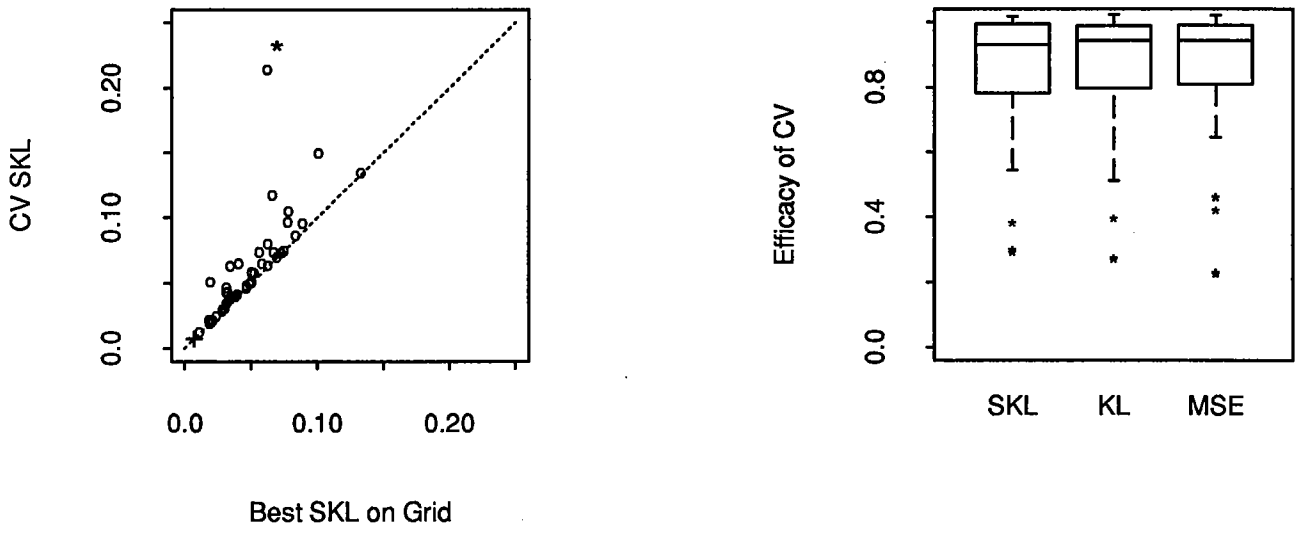


Figure 5.1: The Efficacy of Cross-Validation in the Univariate Simulation.

frame of Figure 5.1, where there are actually 3 outliers for SKL and KL and 5 for MSE with visual overlaps. The best cross-validated fit corresponding to the plus in the left frame of Figure 5.1 is plotted in the top frame of Figure 5.2 as the dashed line, superimposed with the true density as the solid line and the raw data as the finely-binned histogram in dotted lines. The worst case corresponding to the star in the left frame of Figure 5.1, which is the one with a cross-validated $\log_{10}\lambda = -7.25$, is similarly plotted in the bottom frame of Figure 5.2, where the best possible fit is also superimposed as the dashed line with long dashes.

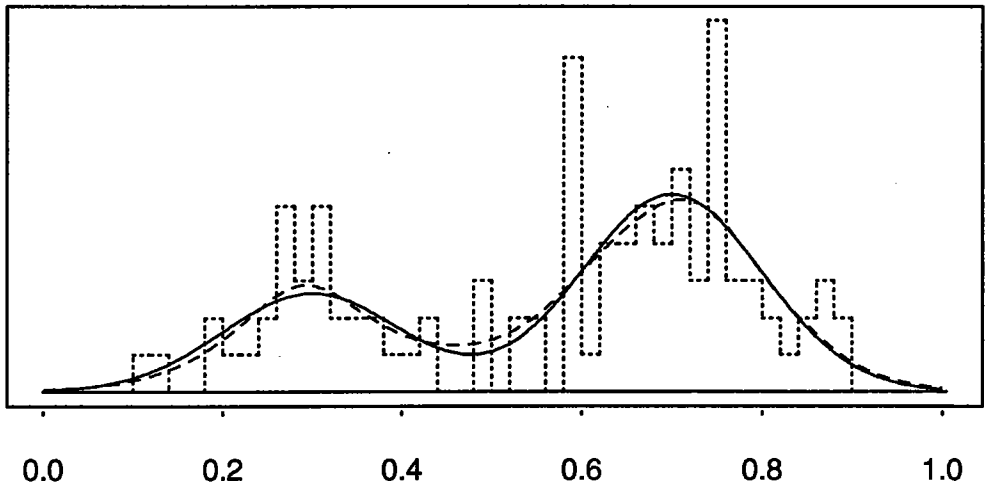
For the bivariate simulation, I took a density on $[0,1]^2$ proportional to

$$\frac{1}{2}e^{-24.5\{(x_1-.3)^2+(x_2-.5)^2\}} + \frac{1}{3}e^{-24.5\{(x_1-.7)^2+(x_2-.7)^2\}} + \frac{1}{6}e^{-24.5\{(x_1-.75)^2+(x_2-.25)^2\}}, \qquad (5.2)$$

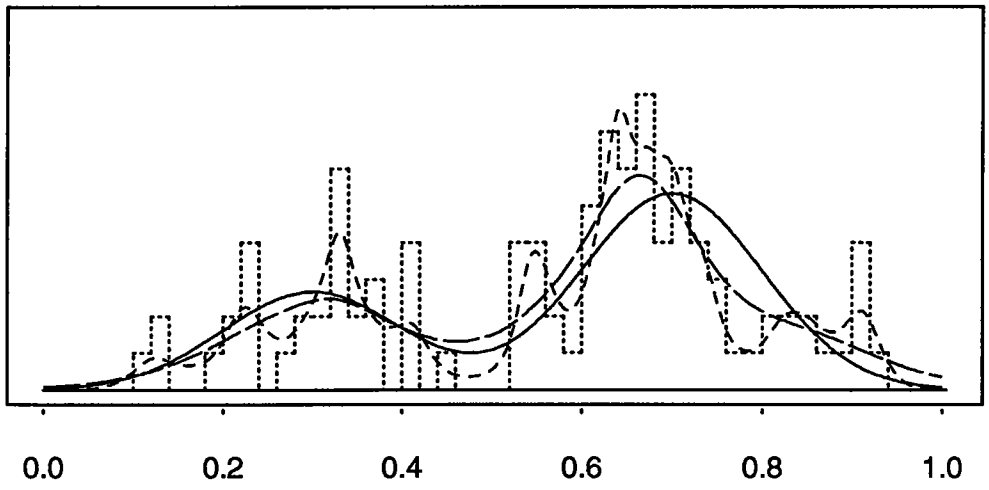
which is a mixture of $N((.3,.5)^T, I/49)$, $N((.7,.7)^T, I/49)$, and $N((.75,.25)^T, I/49)$ with the small mass out of $[0,1]^2$ chopped off. I used null space basis $\phi_1(\boldsymbol{x}) = x_1 - .5$ and $\phi_2(\boldsymbol{x}) = x_2 - .5$, and the RK

$$R_J(\boldsymbol{x},\boldsymbol{y}) = E(\boldsymbol{x},\boldsymbol{y}) - \sum_{j=1}^{4} w_j(\boldsymbol{x})E(\boldsymbol{t}_j,\boldsymbol{y}) - \sum_{k=1}^{4} w_k(\boldsymbol{y})E(\boldsymbol{x},\boldsymbol{t}_k) + \sum_{j=1}^{4}\sum_{k=1}^{4} w_j(\boldsymbol{x})w_k(\boldsymbol{y})E(\boldsymbol{t}_j,\boldsymbol{t}_k), \quad (5.3)$$

where $w_j(\boldsymbol{x}) = .25 + \sum_{\nu=1}^{2}\phi_\nu(\boldsymbol{t}_j)\phi_\nu(\boldsymbol{x})$, $E(\boldsymbol{x},\boldsymbol{y}) = \|\boldsymbol{x}-\boldsymbol{y}\|^2\log\|\boldsymbol{x}-\boldsymbol{y}\|$, and $\boldsymbol{t}_1 = (0,0)^T$, $\boldsymbol{t}_2 = (0,1)^T$, $\boldsymbol{t}_3 = (1,0)^T$, and $\boldsymbol{t}_4 = (1,1)^T$. On a domain $\mathcal{X} = (-\infty,\infty)^2$, ϕ_1, ϕ_2, and R_J as given above



An Ideal CV Fit



A Poor CV Fit

Figure 5.2: A Good Estimate and a Poor Estimate in the Univariate Simulation.

define a smoothing spline satisfying the side condition $\sum_{j=1}^{4} g(t_j) = 0$ with a so-called thin plate penalty proportional to

$$J(g) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (\ddot{g}_{1,1}^2 + 2\ddot{g}_{1,2}^2 + \ddot{g}_{2,2}^2) dx_1 dx_2, \tag{5.4}$$

see, e.g., Gu and Wahba (1990, 1991a). On a finite domain, however, the J corresponding to R_J in (5.3) can not be written explicitly as in (5.4).

Again fifty replicates of data of size 100 were generated according to (5.2) and the cross-validated fits were calculated. Solutions of (2.4) were also calculated on a grid $\log_{10}\lambda = (-5)(.2)(-1)$, which covered the best λ and the cross-validated λ in all the cases. An integration mesh of 500 uniform random points was used in all the calculations including the evaluation of SKL, KL, and MSE. The counterpart of Figure 5.1 is presented in the first row of Figure 5.3. To assess the impact of the integration mesh on the conclusions one may draw from these plots, a second random integration mesh of the same size was also used to calculate SKL, KL, and MSE of all the fits calculated under the first integration mesh. Similar plots are repeated in the second row of Figure 5.3. For the case with the maximum discrepancy between the two SKL evaluations at the cross-validated fit, the SKL as a function of λ evaluated under the two integration meshes are plotted in the left frame of Figure 5.4, where the dotted line is the self-evaluation using the fitting integration mesh and the dashed line is the cross-evaluation using the second mesh. The cross-validated fit is superimposed as the stars. The cross-evaluation agrees with the self-evaluation at the smoother end. And as the estimate gets rougher and rougher, discrepancy increases with the cross-evaluation being pessimistic and the self-evaluation being optimistic. This is typical of all the fifty cases. To obtain some information about the robustness of the cross-validation under different integration meshes, the second integration mesh was also used to calculate the cross-validated fits. The right frame of Figure 5.4 plots the cross-validated $\log_{10}\lambda$'s under the two integration meshes. The true density and the cross-validated fits corresponding to the plus and the star in the $(1,1)$st frame of Figure 5.3 are contoured in Figure 5.5 in the log scale. In the two frames for the estimates, the five solid contours are at the same levels $(1/4, 1/2, 1, 2, 4)$ as the solid contours in the true density frame, and the dotted contour in the good fit is at a higher level (8). The data are superimposed in the estimate frames as circles. A center slice is taken at the dotted lines in the contour frames and a comparison of the conditional densities at the center slice is presented in the last frame, with the true density as the solid line, the good fit as the long dash line, and the poor fit as the short

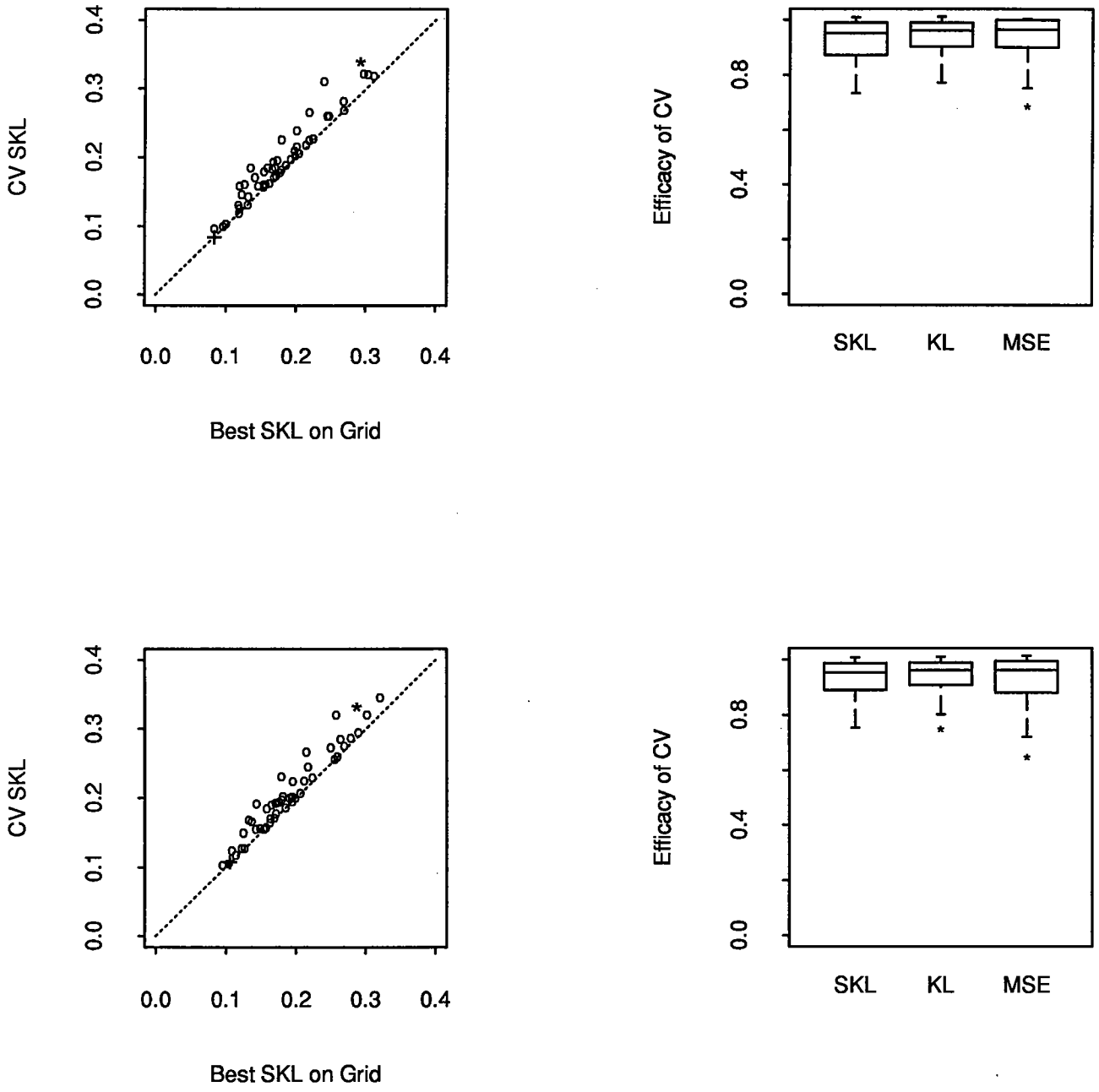


Figure 5.3: The Efficacy of Cross-Validation in the Bivariate Simulation.

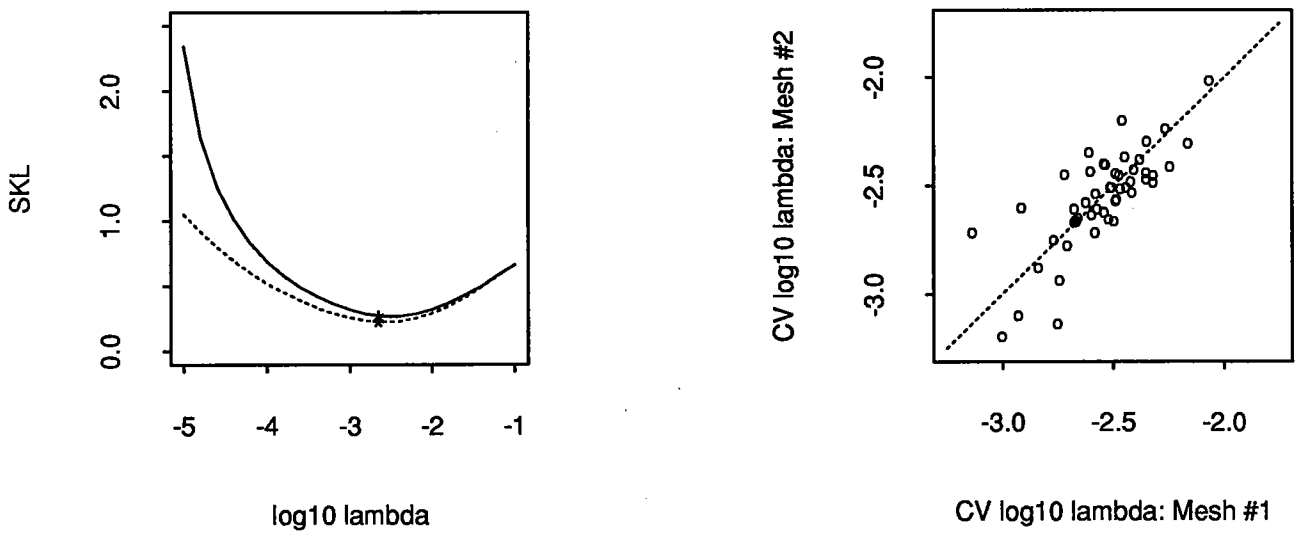


Figure 5.4: The Robustness of the Score Evaluation and the Cross-Validation under Two Integration Meshes.

dash line.

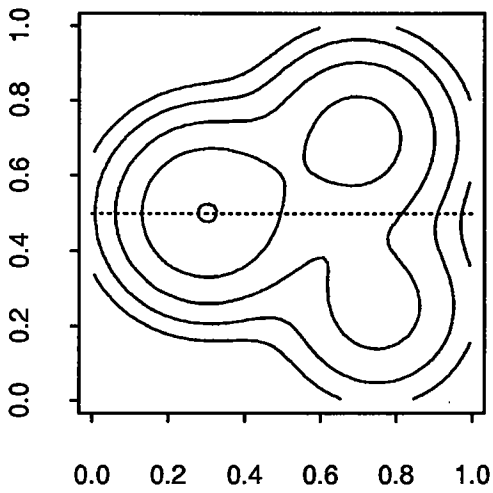
Finally, I report that the algorithm converged without incidence within 20 iterations in all the cases calculated in the two sets of simulations presented above. The number 20 was set as the maximum number of iterations allowed, and in the cases I monitored the algorithm rarely ran more than 10 iterations.

6 Examples

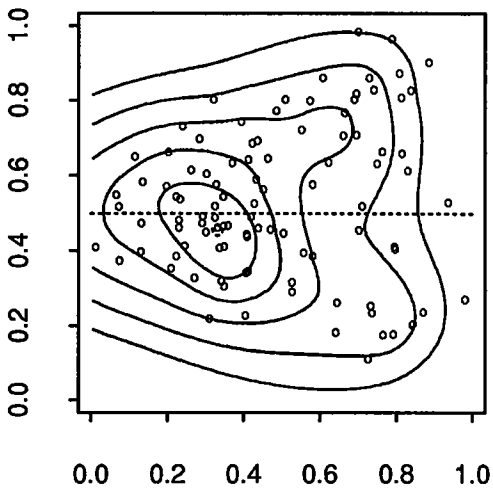
In this section, I apply the algorithm to three real data sets, of which the first two appeared in the nonparametric density estimation literature.

6.1 Buffalo snowfall data

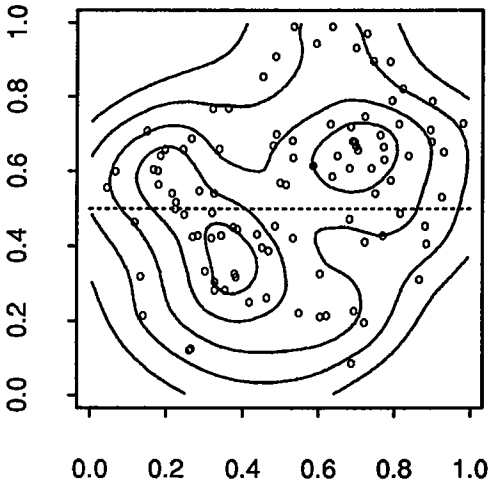
The data are 63 annual snowfall accumulations in Buffalo from 1910 to 1973 as listed in Scott (1985). I divided the raw data by 150 to map them into the $[0, 1]$ interval. I used the same R_J and ϕ as in the univariate simulations of Section 5 to calculate three cross-validated cubic spline fits using three different integration meshes. For the first fit, I used an equally spaced integration



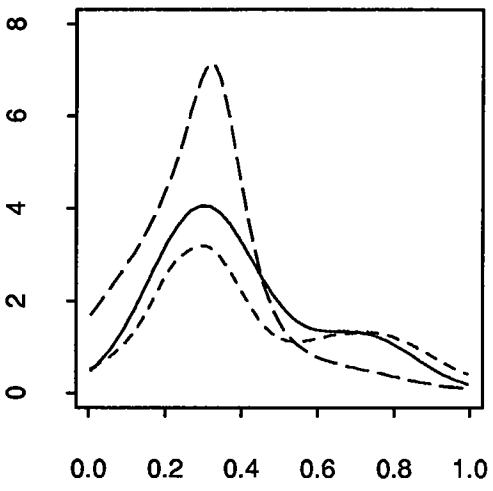
True Density



A Good CV Fit



A Poor CV Fit



A Center Slice

Figure 5.5: A Good Estimate and a Poor Estimate in the Bivariate Simulation.

mesh of 300 points on $(0,1)$, as in the univariate simulation, which amounts to assuming that the density is positive and smooth on $[0,150]$ in the original data scale. For the second fit, I dropped 20 points from each end of the integration mesh, effectively restricting the probability mass to $[10,140]$. For the third fit, 40 more points were dropped from the two ends and the effective support was $[20,130]$. The data range from 25.0 to 126.4. The three fits are plotted in Figure 6.1 as dashed lines of different dash lengths and running lengths. The raw data are superimposed as the finely-binned histogram in dotted lines. The cross-validated $\log_{10}\lambda$ are -5.18, -4.17, and -3.92 for the three fits respectively. It can be seen that as the support extends farther into the no data area, the cross-validation tries harder to take away the mass assigned to the empty space by the smoothness of the estimate, resulting in less smoothing. It can also be seen from Figure 6.1 that the estimates are less affected at the lower end because it is a very thin end (the second smallest data is 39.8). Conversely, the support change at the thin end ideally shall not affect the cross-validation by much. In fact, the cross-validated $\log_{10}\lambda$ is -5.14 with a support $[20,150]$ and is -4.26 with a support $[0,140]$ (cf. above). It is clear that the cut point at a fat end is a rather strong assumption and hence has a rather major impact on the choice of the cross-validation.

6.2 Lawrence Radiation Laboratory data

The data are listed in Good and Gaskins (1980) as a histogram of 172 bins of length 10 MeV constructed from the locations of 25752 events on a mass-spectrum. For such a huge data set in one dimension, eyeballs should be able to do as good a job as any fancy smoothing technique. Here my interest is simply to double check whether the proposed method gives a reasonable solution. A cross-validated cubic spline fit was calculated on $[250,2030]$ in the original MeV scale using an integration mesh of 500 equally spaced points, where the interval extends 3 bins on each end beyond the original histogram. Since both ends are very thin, the cross-validation is not sensitive to the cut points. The estimate is plotted in Figure 6.2 as the solid line, and the original data are superimposed as the dotted line. Visually I could see little difference between this fit and the fit presented by Good and Gaskins (1980).

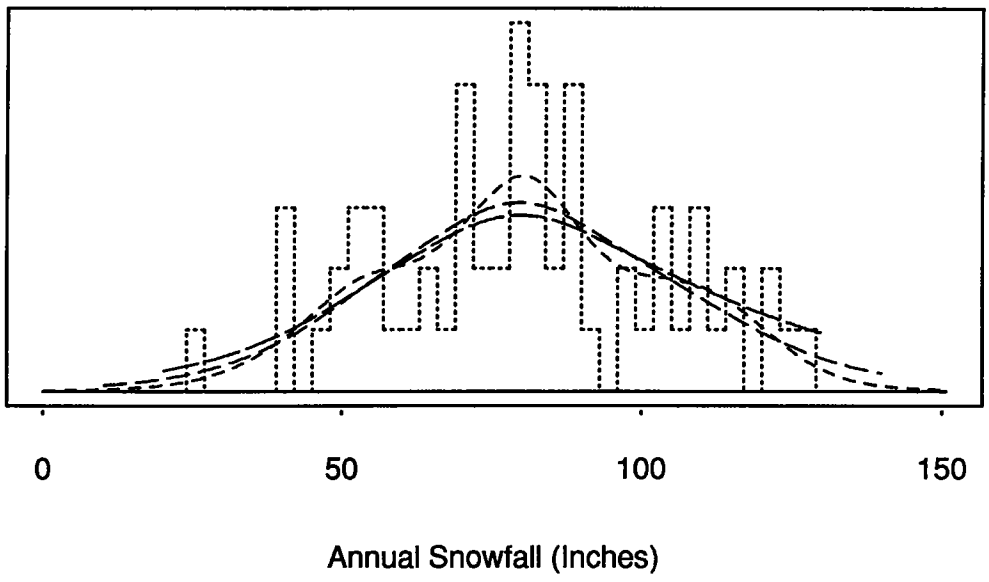


Figure 6.1: The Distribution of Buffalo Annual Snowfall.

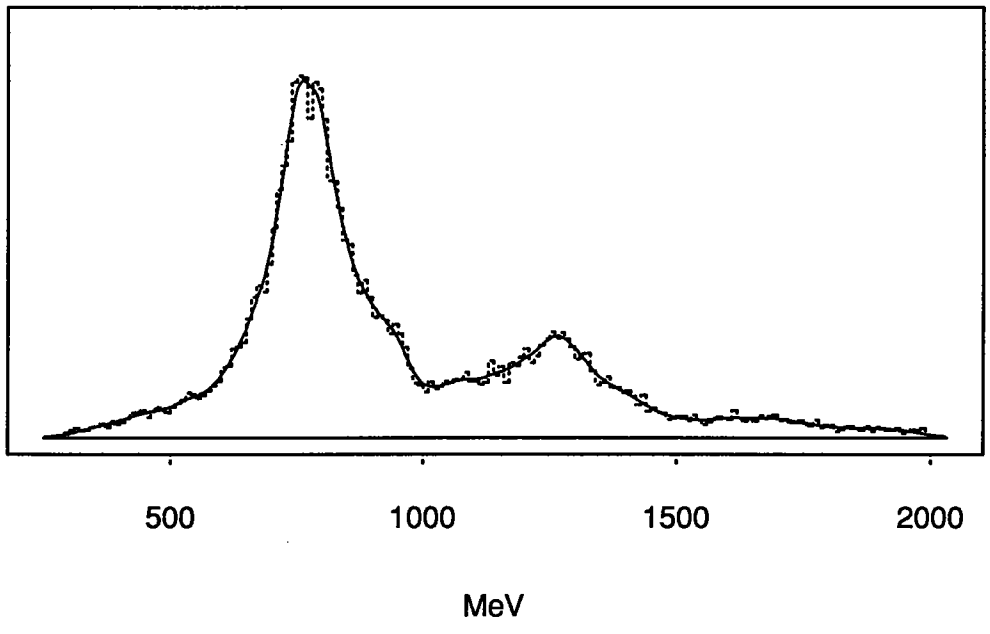


Figure 6.2: A Mass-Spectrum from Lawrence Radiation Laboratory.

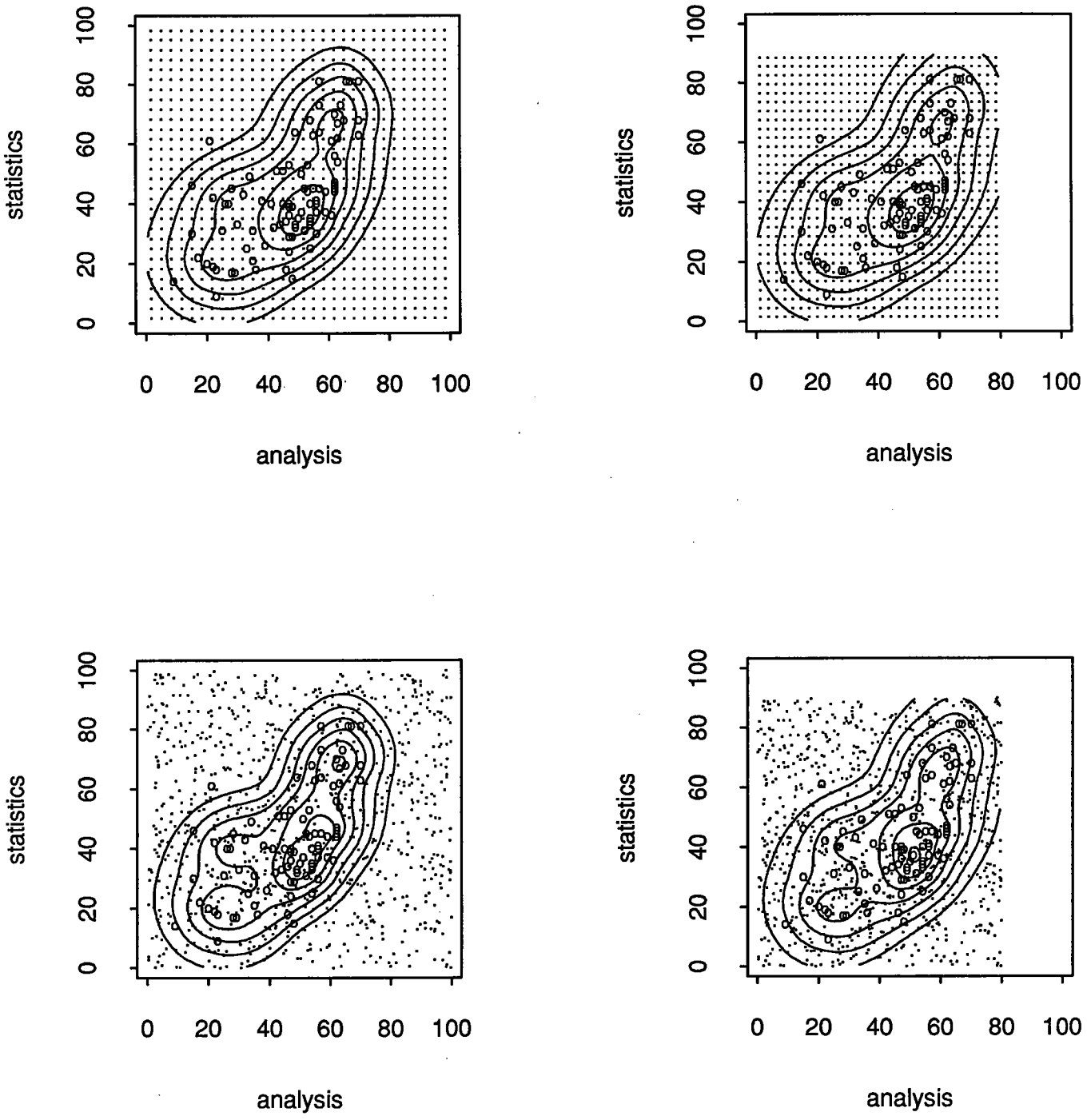


Figure 6.3: The Joint Distribution of Analysis and Statistics Marks.

6.3 Mathematics marks data

The data are listed in Whittacker (1990) and were used as a prime example there to demonstrate Gaussian graphical models. It shall be possible to conduct similar analyses nonparametrically using tensor product splines, but further computational and data analytical tools have yet to be developed. See Section 7. Here I arbitrarily pick the Analysis marks and the Statistics marks of the 88 students and try to estimate the two-dimensional joint distribution using the technique developed in this article. The marks are on the percentage scale. I transformed $[0, 100]$ to $[-2, 3]$ and used the thin plate spline of (5.3) to calculate the estimate. I chose not to map $[0, 100]^2$ onto the unit square for numerical considerations, noting that the side condition in (5.3) is $\sum_{j=1}^{4} g(t_j) = 0$ but three of the four corners of $[0, 100]^2$ are far away from the data. I used 30×30 regular integration meshes on domains $[0, 100]^2$ and $[0, 80] \times [0, 90]$ (in the original scale) respectively, and calculated the cross-validated fits. The estimates are contoured in the first row of Figure 6.3 in the log scale at levels $1/4, 1/2, 1, 2, 4, 8$, with the data superimposed as the circles and the integration meshes superimposed as the dots. Estimates were also calculated using the 80×80 meshes on the same domains, which gave contours visually identical to those calculated under the 30×30 meshes. It seems rather safe to take these fits as the solutions under the exact continuous integration measure. I also mapped two random meshes of 900 points on $[0, 1]^2$ onto the domains and tried to calculated the corresponding cross-validated fits. On the domain $[0, 100]^2$, the cross-validation drove the algorithm to interpolate the data under the first random mesh but performed normally under the second. On the domain $[0, 80] \times [0, 90]$, the algorithm performed normally under both meshes. The cross-validated fits under the second mesh are similarly plotted in the second row of Figure 6.3.

7 Discussion

In this article, I developed and illustrated a dimensionless automatic algorithm for calculating the smoothing spline density estimator of Gu and Qiu (1991). Simulation results suggest that the algorithm is quite effective in selecting a good smoothing parameter. The Ratfor implementation of the algorithm is currently available from me at `chong@stat.purdue.edu`, and will be included in later versions of RKPACK (Gu 1989) for direct public access. With the help of the software, it is hoped that the reliable routine use of nonparametric density estimation by nonexpert, even on a

multidimensional and/or irregular domain, could be made feasible.

The real beauty of the smoothing spline approach to nonparametric multivariate problems is in its structural construction of the estimators, in which tensor product splines with their built-in ANOVA decompositions play an important role; see, e.g., Gu and Wahba (1990, 1991a, b). In the density estimation context, tensor product splines may be employed to impose and/or explore various independence structures of the joint probability distribution of qualitatively different variables. Specifically, using tensor product splines, a function g of several variables, say x_α, $\alpha = 1, \cdots, d$, can be written as

$$g = C + \sum_\alpha g_\alpha(x_\alpha) + \sum_{\alpha<\beta} g_{\alpha,\beta}(x_\alpha, x_\beta) + \cdots + g_{1,\cdots,d}(x_1, \cdots, x_d), \tag{7.1}$$

where C is a constant, g_α are functions of one variable called the main effects, $g_{\alpha,\beta}$ are functions of two variables called the two-factor interactions, etc., and these terms, except the constant, satisfy certain built-in side-conditions such that the decomposition is unique. For a log likelihood g, the constant shall be dropped to maintain a one-to-one logistic density transform. The inclusion/exclusion of other terms in the right-hand-side of (7.1) may be exploited to represent various independence structures. For example, the independence of all variables is represented by the main-effect-only model, and the conditional independence of x_α and x_β given other variables can be obtained by eliminating all terms involving both x_α and x_β (cf. Whittacker 1990). The exclusion of all but two-factor interactions may partially ease the curse of dimensionality (cf. Huber 1985), and yet preserve sufficient varieties in the possible probability structures representable by the models. Actually, a family with interactions of at most two factors represents a "minimal" nonparametric generalization of the Gaussian graphical association models (cf. Whittacker 1990). In general, the use of tensor product splines shall allow the fitting of nonparametric graphical association models with both discrete and continuous variables (cf. Gu and Wahba 1991b). The current development, however, is a bit shy of materializing the full potential of the methodology, since it lacks the capability of handling multiple smoothing parameters, which are essential in the fitting of tensor product spline models.

References

Aronszajn, N. (1950), "Theory of Reproducing Kernels," *Transaction of the American Mathematical Society*, 68, 337 – 404.

Bickel, P. and Doksum, K. (1977), *Mathematical Statistics*, San Francisco: Holden-Day.

Dongarra, J. J., Moler, C. B., Bunch, J. R. and Stewart, G. W. (1979), *LINPACK Users' Guide*, Philadelphia: SIAM.

Good, I. J. and Gaskins, R. A. (1971), "Nonparametric Roughness Penalties for Probability Densities," *Biometrika*, 58, 255 – 277.

—— (1980), "Density Estimation and Bump-Hunting by the Penalized Likelihood Method Exemplified by Scattering and Meteorite Data" (with discussion), *Journal of the American Statistical Association*, 75, 42 – 73.

Gu, C. (1989), "RKPACK and Its Applications: Fitting Smoothing Spline Models," *Proceedings of Statistical Computing Section: American Statistical Association*, 42 – 51.

—— (1990), "A Note on Cross-Validating Non Gaussian Data," Technical Report 96, Dept. Statistics, University of British Columbia.

Gu, C., Bates, D. M., Chen, Z., and Wahba, G. (1989), "The Computation of GCV Functions through Householder Tridiagonalization with Applications to the Fitting of Interaction Spline Models," *SIAM Journal on Matrix Analysis and Applications*, 10, 457 – 480.

Gu, C. and Qiu, C. (1991), "Smoothing Spline Density Estimation: Theory," Technical Report 91-19, Dept. Statistics, Purdue University.

Gu, C. and Wahba, G. (1990), "Semiparametric ANOVA with Tensor Product Thin Plate Splines," Technical Report 90-61, Dept. Statistics, Purdue University.

—— (1991a), Discussion of "Multivariate adaptive regression splines" by J. Friedman, *The Annals of Statistics*, 19, 115 – 123.

—— (1991b), "Smoothing Splines and Analysis of Variance in Function Spaces," Technical Report 91-29, Dept. Statistics, Purdue University.

Huber, P. (1985), "Projection Pursuit" (with discussion), *The Annals of Statistics*, 13, 435 – 475.

Klonias, V. K. (1982), "Consistency of a Nonparametric Penalized Likelihood Estimator of the Probability Density Function," *The Annals of Statistics*, 10, 811 – 824.

Leonard, T. (1978), "Density Estimation, Stochastic Processes and Prior Information" (with discussion), *Journal of the Royal Statistical Society Ser. B*, 40, 113 – 146.

O'Sullivan, F. (1988), "Fast Computation of Fully Automated Log-Density and Log-Hazard Estimators," *SIAM Journal on Scientific and Statistical Computing*, 9, 363 – 379.

Scott, D. (1985), "Averaged Shifted Histograms: Effective Nonparametric Density Estimators in Several Dimensions," *The Annals of Statistics*, 13, 1024 – 1040.

—— (1986), "Choosing Smoothing Parameters in Density Estimators," in *Computer Science and Statistics: Proceedings of the 17th Symposium on the Interface*, ed. D.M. Allen, Amsterdam: North-Holland, pp. 225 – 229.

Silverman, B. W. (1982), "On the Estimation of a Probability Density Function by the Maximum Penalized Likelihood Method," *The Annals oof Statistics*, 10, 795 – 810.

—— (1986), *Density Estimation for Statistics and Data Analysis*, New York: Chapman and Hall.

Tapia, R. A. and Thompson, J. R. (1978), *Nonparametric Probability Density Estimation*, Baltimore: Johns Hopkins University Press.

Wahba, G. (1977), "Optimal Smoothing of Density Estimates," in *Classification and Clustering*, ed. J. van Ryzin, New York: Academic Press, pp. 423 – 458.

—— (1990), *Spline Models for Observational Data.* CBMS–NSF Regional Conference Series in Applied Mathematics, Vol. 59, Philadelphia: SIAM.

Whittacker, J. (1990), *Graphical Models in Applied Multivariate Statistics*, Chichester, U.K.: John Wiley.