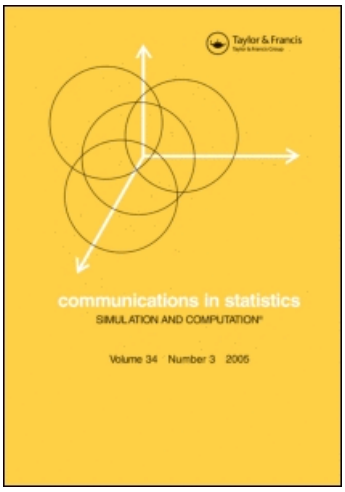

Communications in Statistics - Simulation and Computation
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713597237

An Efficient Algorithm for the Least-Squares Cross-Validation with Symmetric and Polynomial Kernels

Byung Gook Lee [a]; Byung Chun Kim [a]
[a] Korea Advanced Institute of Science and Technology, Seoul, Korea

Online Publication Date: 01 January 1990

To cite this Article Lee, Byung Gook and Kim, Byung Chun(1990)'An Efficient Algorithm for the Least-Squares Cross-Validation with Symmetric and Polynomial Kernels',Communications in Statistics - Simulation and Computation,19:4,1513 — 1522
To link to this Article: DOI: 10.1080/03610919008812933
URL: http://dx.doi.org/10.1080/03610919008812933

COMMUN. STATIST.-SIMULA., 19(4), 1513-1522 (1990)

AN EFFICIENT ALGORITHM FOR THE LEAST-SQUARES CROSS-VALIDATION WITH SYMMETRIC AND POLYNOMIAL KERNELS

Byung Gook Lee and Byung Chun Kim
Korea Advanced Institute of Science and Technology
Seoul, Korea

Key Words and Phrases: probability density estimation; least-squares cross-validation; convolution; symmetric and polynomial kernels.

ABSTRACT

The least-squares cross-validation is a completely automatic method for choosing the smoothing parameter in probability density estimation but this method consume large amounts of computer time. This article concerns an efficient computational algorithm for this method when the kernel is symmetric and polynomial functions.

1. INTRODUCTION

It is assumed that we have a sample $X_1, X_2, \ldots, X_n$ of independent, identically distributed observations from a continuous univariate distribution with probability density function f, which we are trying to estimate.

Rosenblatt (1956) introduced the kernel estimator with kernel K is defined by

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - X_i}{h}) \tag{1}$$

where h is the smoothing parameter.

When applying the method in practice it is of course necessary to choose a kernel and a smoothing parameter. The choice of kernel was considered by Epanechnikov (1969) who showed that there is in some sense an optimal kernel, which is part of a parabola, but that any reasonable kernel gives almost optimal results. Therefore the choice of kernel is not as important a problem in practice as might be supposed. The problem of choosing how much to smooth is of crucial importance in probability density estimation.

The most widely studied smoothing parameter selector is the least squares cross-validation, proposed by Rudemo (1982) and Bowman (1984). This method has been shown to have the attractive asymptotic property of given an answer that converges to the optimum under weak conditions (see Stone (1984)).

The least-squares cross-validated smoothing parameter h is the minimizer of the cross-validation score function

$$M_0(h) = \int \hat{f}^2 - 2n^{-1} \sum_{i=1}^{n} \hat{f}_{-i}(X_i) \tag{2}$$

where $\hat{f}_{-i}$ denotes the leave-one-out kernel estimator constructed from the data X_i deleted.

To express the score function M_0 in a form which is more suitable for computation, define $K^{(2)}$ to be a convolution of the kernel with itself and assuming K is symmetric. Then the least-squares cross-validation score function is given by

$$M_0(h) = \frac{1}{n^2 h}\sum_{i=1}^{n}\sum_{j=1}^{n} K^{(2)}\left(\frac{X_i - X_j}{h}\right)$$

$$-\frac{2}{n(n-1)h}\sum_{i=1}^{n}\sum_{j=1}^{n} K\left(\frac{X_i - X_j}{h}\right) + \frac{2}{(n-1)h}K(0). \tag{3}$$

See Silverman (1986) for details. But the direct use for computation of formular (3) for the kernel estimate is highly inefficient because it appears that $n(n-1)/2$ evaluations of the function $K^{(2)}$ and K are needed to compute each value of the score function M_0; since it is then necessary to minimize over h.

In order to compute score function M_0, Silverman (1982) presents an algorithm for the efficient computation of kernel density estimate method by Fourier transform method and the improvements to this algorithm suggested by Jones and Lotwick (1984).

Now we shall see below that the least-squares cross-validation score function M_0 can be found quickly when the kernel K is symmetric and polynomial functions. The basic idea of the algorithm we shall develop in this article is to extract h from K and $K^{(2)}$, and not evaluate the sum of K and $K^{(2)}$ at each h.

2. THE PROPOSED ALGORITHM

In this section we present an efficient algorithm to calculate the score function when the kernel K is symmetric and polynomial functions, for example the rectangular, triangular, biweight and Epanechnikov kernel. Before beginning the algorithm, it is necessary to calculate the absolute difference $X_i - X_j$ as

$$\begin{array}{lll} d'_1 = |X_1 - X_2|, & d'_2 = |X_1 - X_3|, & \cdots \quad d'_{n-1} = |X_1 - X_n| \\ & d'_n = |X_2 - X_3|, & \cdots \quad d'_{2n-3} = |X_2 - X_n| \\ & & \ddots \qquad \vdots \\ & & d'_{n(n-1)/2} = |X_{n-1} - X_n|. \end{array}$$

Since the kernel K is symmetric function, it is clear that $K((X_i - X_j)/h) = K((X_j - X_i)/h) = K(|X_i - X_j|/h)$ for all i, j.

The vector $\{d_1, d_2, \ldots, d_{n(n-1)/2}\}$ is the sorted array in ascending order for the array $\{d'_1, d'_2, \ldots, d'_{n(n-1)/2}\}$ and we need to construct, for each $i = 1, 2, \ldots, n(n-1)/2$, the vector $S_i = \sum_{k=1}^{i} d_k$.

Then we will check each polynomial kernels.

Algorithm 1.

If function K is the rectangular kernel

$$K(t) = \begin{cases} \frac{1}{2}, & \text{if } |t| < 1; \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

then

$$K^{(2)}(t) = \begin{cases} \frac{1}{2} - \frac{1}{4}|t|, & \text{if } |t| < 2; \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

If we define $t = (X_i - X_j)/h$ and I_A denotes the indicator function of set A, then from (3) the least-squares cross-validation score is given by

$$M_0(h) = \frac{1}{n^2 h}\{\sum_{i=1}^{n}\sum_{j=1}^{n}(\frac{1}{2} + \frac{t}{4})I_{(-2,0]}(t) + \sum_{i=1}^{n}\sum_{j=1}^{n}(\frac{1}{2} - \frac{t}{4})I_{(0,2)}(t)\}$$

$$-\frac{1}{n(n-1)h}\sum_{i=1}^{n}\sum_{j=1}^{n}I_{(-1,1)}(t) + \frac{1}{(n-1)h}. \tag{6}$$

Let m be the number of pairs $i < j$ for which $X_i = X_j$, then

$$M_0(h) = \frac{1}{n^2 h}\{\frac{1}{2}(2\sum_{i=1}^{n}\sum_{j=1}^{n}I_{(0,2)}(t) + 2m + n) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}tI_{(0,2)}(t)\}$$

$$-\frac{1}{n(n-1)h}(2\sum_{i=1}^{n}\sum_{j=1}^{n}I_{(0,1)}(t) + 2m + n) + \frac{1}{(n-1)h} \tag{7}$$

$$= \frac{1}{n^2 h}\{(\sum_{i=1}^{n}\sum_{j=1}^{n} I_{(0,2)}(t) + m) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} tI_{(0,2)}(t) + \frac{n}{2}\}$$

$$-\frac{2}{n(n-1)h}(\sum_{i=1}^{n}\sum_{j=1}^{n} I_{(0,1)}(t) + m). \tag{8}$$

The above formular (8) can be phrased as an algorithm.

Step 1: Set h.
Step 2: Find the largest number n_1, n_2 such that $d_{n_1} < h$, $d_{n_2} < 2h$.
Step 3: $M_0(h) =$

$$\frac{1}{n^2 h}(n_2 - \frac{1}{2h}S_{n_2} + \frac{n}{2}) - \frac{2n_1}{n(n-1)h}$$

Algorithm 2.

If function K is the triangular kernel

$$K(t) = \begin{cases} 1 - |t|, & \text{if } |t| < 1; \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

then

$$K^{(2)}(t) = \begin{cases} \frac{1}{2}|t|^3 - t^2 + \frac{2}{3}, & \text{if } |t| < 1; \\ -\frac{1}{6}|t|^3 + t^2 - 2|t| + \frac{4}{3}, & \text{if } 1 \le |t| < 2; \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

By the same method of Algorithm1, we can easily formulate as follows.

Step 1: Set h.
Step 2: Find the largest number n_1, n_2 such that $d_{n_1} < h$, $d_{n_2} < 2h$.
Step 3: $M_0(h) =$

$$\frac{1}{n^2 h}\{\frac{1}{3h^3}(-S_{n_2}^3 + 4S_{n_1}^3) + \frac{2}{h^2}(S_{n_2}^2 - 2S_{n_1}^2) - \frac{4}{h}(S_{n_2} - S_{n_1})$$

$$+\frac{4}{3}(2n_2 - n_1) + \frac{2}{3}n\} - \frac{4}{n(n-1)h}(-\frac{1}{h}S_{n_1} + n_1)$$

where $S_i^j = \sum_{k=1}^{i} d_k^j$.

Algorithm 3.

If function K is the Epanechnikov kernel

$$K(t) = \begin{cases} \frac{3}{4\sqrt{5}}(1 - \frac{1}{5}t^2), & \text{if } |t| < \sqrt{5}; \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

then

$$K^{(2)}(t) = \begin{cases} \frac{3}{40}(-\frac{1}{500}|t|^5 + \frac{1}{5}|t|^3 - \frac{2\sqrt{5}}{5}t^2 + \frac{8\sqrt{5}}{5}), & \text{if } |t| < 2\sqrt{5}; \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

By the same method of Algorithm1, we can easily formulate as follows.

Step 1: Set h.

Step 2: Find the largest number n_1, n_2
such that $d_{n_1} < \sqrt{5}h$, $d_{n_2} < 2\sqrt{5}h$.

Step 3: $M_0(h) =$

$$\frac{3}{100n^2h}\{-\frac{1}{100h^5}S_{n_2}^5 + \frac{1}{h^3}S_{n_2}^3 - \frac{2\sqrt{5}}{h^2}S_{n_2}^2 + 4\sqrt{5}(2n_2 + n)\}$$

$$-\frac{3}{\sqrt{5}n(n-1)h}(-\frac{1}{5h^2}S_{n_1}^2 + n_1)$$

3. COMPUTATIONAL EXPERIENCE

A computational study was conducted to compare new algorithm with a routine calculating direct from (3) in each symmetric and polynomial kernel. The observations were generating using the standard normal distribution and the sample sizes used were 50, 100, 150 and 200. Results were based on 10 replications. These were calculated from the range of smoothing parameter $(0, 2)$ and step size=0.01. IBM PC 80386 microcomputer was used in this study, and the sample results are given in Table I. The results that minimizes the score function of the two method were exactly equivalent.

TABLE I
Computational Results. Mean Time (Sec.)

Sample Size		Rectangular	Triangular	Epanechnikov
50	New Alg	.66	.99	1.05
	Direct	45.65	90.08	201.68
100	New Alg	3.02	4.28	4.40
	Direct	182.25	359.71	812.02
150	New Alg	7.31	10.25	10.38
	Direct	410.01	813.45	1828.58
200	New Alg	13.78	18.95	19.45
	Direct	729.41	1442.07	3250.21

4. CONCLUSIONS

This article has considered the problem of calculating the least-squares cross-validation score function for a kernel density estimation. From the computational experience it appears that the new method is very efficient algorithm in terms of computer time but requires more storage. The number of observations may not exceed 245. The user can easily extend these limitations by changing the dimensions on the appropriate working arrays in the program.

BIBLIOGRAPHY

Bowman,A.W.(1984). An alternative method of cross-validation for the smoothing of density estimates. *Biometrika* **71**,353-360.

Epanechnikov,V.A.(1969). Nonparametric estimation of a multidimensional probability density. *Theory Prob.Appl.* **14**,153-158.

Jones,M.C. and Lotwick,H.W.(1984). A remark on Algorithm AS 176. Kernel density estimation using the fast Fourier transform. Remark AS R50. *Appl.Statist.* **33**,120-122.

Rosenblatt,M.(1956). Remarks on some nonparametric estimates of a density function. *Ann.Math.Statist.* **27**,832-837.

Rudemo,M.(1982). Empirical choice of histograms and kernel density estimators. *Scand.J.Statist.* **9**,65-78.

Silverman,B.W.(1982). Kernel density estimation using the fast Fourier transform. Statistical Algorithm AS 176. *Appl.Statist.* **31**,93-97.

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

Stone,C.J.(1984). An asymptotically optimal window selection rule for kernel density estimates. *Ann.Statist.* **12**,1285-1297.

APPENDIX

```
      SUBROUTINE EPASCORE(DT, NDT, LSP, USP, STEP, SCORE, SP)

C     FIND MINIMUM SCORE M0 OVER THE RANGE OF SMOOTHING
C     PARAMETER (LSP, USP) BY THE LEAST-SQUARES CROSS-
C     VALIDATION WITH EPANECHNIKOV KERNEL
C
C INPUT:
C
C DT   = Real array(NDT) contains the raw data values.
C NDT  = Number of observations.
C LSP  = The lower bound of the range of smoothing parameter.
C USP  = The upper bound of the range of smoothing parameter.
C STEP = Step size.
C
C OUTPUT:
C
C SCORE= Minimum score M0 value.
C SP   = The smoothing parameter value.
C
      INTEGER    NDT, COUNT
      REAL       H, LSP, USP, STEP, SCORE, SP, SQ5
      REAL       DT, DIST, DIS2, DIS3, DIS5
C
C     The number of observation may not exceed 200(NDT).
C     N = 20000 (NDT*(NDT-1)/2).
C
      PARAMETER  (N=20000)
      DIMENSION  DT(*), DIST(N), DIS2(N), DIS3(N), DIS5(N)
C
C     Set up initial data values.
C     Make absolute difference Xi-Xj vector.
C
      CALL MAKDIST(NDT, DT, DIS2, COUNT)
C
C     Sort absolute difference vector.
C     SVRGN : Sort a real array by algebraic value.
C     IMSL STAT/LIBRARY FORTRAN SUBROUTINE
```

```
C
      CALL SVRGN(COUNT, DIS2, DIST)
C
C     Make sum of sorted absolute difference vector.
C
      CALL SUMDIST(COUNT, DIST, 2., DIS2)
      CALL SUMDIST(COUNT, DIST, 3., DIS3)
      CALL SUMDIST(COUNT, DIST, 5., DIS5)
C
C     N1 = The largest number such that d_n1 <   SQ5*H.
C     N2 = The largest number such that d_n2 < 2*SQ5*H.
C
      N1=1
      N2=1
      SQ5=SQRT(5.)
      SCORE=999.
      DO 10 H = LSP, USP, STEP
 11     IF((DIST(N1) .LT. SQ5*H) .AND. (N1 .LE. COUNT)) THEN
          N1=N1+1
          GOTO 11
        ENDIF
 12     IF((DIST(N2) .LT. 2.*SQ5*H) .AND. (N2 .LE. COUNT)) THEN
          N2=N2+1
          GOTO 12
        ENDIF
        SUM1=-DIS5(N2)/(100.*H**5.)+DIS3(N2)/H**3.
     *    -2.*SQ5*DIS2(N2)/H**2.+4.*SQ5*(2.*(N2-1.)+NDT)
        SUM2=N1-1.-DIS2(N1)/(5.*H**2.)
        SUM =3.*(SUM1/(100.*NDT*NDT*H)-SUM2/(SQ5*NDT*(NDT-1.)*H))
        IF(SUM .LT. SCORE) THEN
          SCORE=SUM
          SP=H
        ENDIF
 10   CONTINUE
      RETURN
      END



      SUBROUTINE MAKEDIST(N, X, D, K)
C
C     MAKE ABSOLUTE DIFFERENCE X_i-X_j VECTOR
C
C INPUT:
C
C N    = Number of obervations.
C X    = Real array(N) contains the raw data values.
C
C OUTPUT:
C
C D    = Absolute difference vector.
C K    = Size of D vector (N*(N-1)/2).
C
```

```
      DIMENSION  X(*), D(*)

      K=0
      DO 10 I = 1, N
        DO 10 J = I+1, N
          K=K+1
          D(K)=ABS(X(I)-X(J))
 10   CONTINUE
      RETURN
      END



      SUBROUTINE SUMDIST(N, X, P, D)
C
C     MAKE SUM OF SORTED ABSOLUTE DIFFERENCE DATA VALUES
C
C INPUT:
C
C N    = Size of vector X.
C X    = Real array(N) contains the sorted absolute
C        difference data values.
C P    = P_th power value.
C
C OUTPUT:
C
C D    = Sum of sorted absolute difference data values
C        with P_th power.
C        ( D(i) = X(1)**P + X(2)**P + . . . + X(i-1)**P )
C
      DIMENSION  R(*), D(*)

      D(1)=0.
      D(2)=X(1)**P
      DO 10 I = 3, N+1
         D(I)=D(I-1)+X(I-1)**P
 10   CONTINUE
      RETURN
      END
```