Algorithm AS 105

Fitting a Covariance Selection Model to a Matrix

By Nanny Wermuth and Eberhard Scheidt

Institut für Medizinische Statistik und Dokumentation,
Johannes-Gutenberg-Universität, Mainz, Federal Republic of Germany

Keywords: COVARIANCE SELECTION; PATTERN OF ASSOCIATION; PARTIAL CORRELATION

Language

ISO Fortran

Description and Purpose

We propose a cyclic fitting algorithm to fit any given covariance selection model to a symmetric, positive definite matrix, generally to an observed covariance matrix. The theory of covariance selection had been proposed (Dempster, 1972) as a means of parameter reduction when the covariance structure of a multivariate normal distribution is to be estimated. It has been shown (Wermuth, 1976b) how a subclass of covariance selection models and multiplicative models for contingency tables (Goodman, 1970; Bishop, 1971) may both be used in a similar manner to study simple patterns of assocation (see also Wermuth *et al.*, 1976).

A covariance selection model may be characterized by the sequence of variable pairs that have zero concentrations (Dempster, 1972) or, equivalently, by the sequence of variable pairs with zero partial correlations (Wermuth, 1976a). If exactly one concentration is to be restricted to zero, then the matrix $\mathbf{S} = (s_{rt})$ and its inverse $\mathbf{S}^{-1} = (s^{rt})$ are changed to $\mathbf{M}$ and to $\mathbf{M}^{-1}$, respectively, in the following way. Let (i, j) be the indices of the variable pair that is to have zero concentration, then $\mathbf{M}$ coincides with S except for the elements m_{ij} and m_{ji}:

$$m_{ij} = s_{ij} + s^{ij}/D \quad \text{with} \quad D = s^{ii}s^{jj} - (s^{ij})^2. \tag{1}$$

Furthermore, $\mathbf{M}^{-1}$ is defined by

$$\left.\begin{aligned} m^{ij} &= 0 \\ m^{ii} &= D/s^{jj}, \\ m^{jj} &= D/s^{ii}, \\ m^{ik} &= s^{ik} - s^{ij}s^{jk}/s^{jj}, \\ m^{jk} &= s^{jk} - s^{ij}s^{ik}/s^{ii}, \\ m^{kl} &= s^{kl} - (s^{ij}/D)\{s^{ik}(s^{jl} - s^{ij}s^{il}/s^{ii}) + s^{jk}(s^{il} - s^{ij}s^{jl}/s^{jj})\}, \\ &\text{for all } k \text{ and } l \text{ not equal to } i \text{ or } j. \end{aligned}\right\} \tag{2}$$

If S is the covariance matrix of a multivariate normal distribution, then (2) gives the closed form of the maximum-likelihood estimate for the inverse covariance structure with one zero concentration. We call (2) the *INVEST*-operator.

For a typical covariance selection model several concentrations have to be forced to zero. Then, the *INVEST*-operator may be applied repeatedly to the prespecified or selected variable pairs. The cycling ends when all these concentrations are close enough to zero.

The subroutine *INVEST* modifies and operates on only the upper triangular part of its input matrix. Since the input matrix for *INVEST* is the inverse of some initial matrix, we assume that its positive-definiteness has been checked before entering into *INVEST*. Therefore, there is only one failure indication.

Structure

SUBROUTINE INVEST (MAT, NDIM, I, J, NVAR, IFAULT)

Formal parameters

MAT	Real array (*NDIM*, *NVAR*)	input:	any symmetric, positive definite matrix, S^{-1}, generally the inverse of a covariance matrix
		output:	the upper triangular part of *MAT* is modified for variable pair (*I*,*J*) as specified in equation (2)
NDIM	Integer	input:	the maximum number of rows in the matrix *MAT*, as specified in the main program
I	Integer	input:	the smaller index of a selected variable pair
J	Integer	input:	the larger index of a selected variable pair
NVAR	Integer	input:	the actual number of rows in the matrix *MAT*
IFAULT	Integer	output:	indicates failure to specify *I*,*J* correctly

$$IFAULT = \begin{cases} 0 & \text{if } 1 \leqslant I < J \leqslant NVAR \\ 1 & \text{otherwise} \end{cases}$$

Time and Accuracy

We tested the algorithm with a calling program on a number of simple matrices. In this calling program the *NPAIR* concentrations were not fitted in a prespecified order, instead—at each call of *INVEST*—the largest of the *NPAIR* concentrations was set to zero. The iterations

Table 1

Number of calls to INVEST and computing time (sec) on a CDC 3300

		Correlation matrices					
		$r = 0{\cdot}2$		$r = 0{\cdot}5$		$r = 0{\cdot}8$	
		DELTA		*DELTA*		*DELTA*	
NVAR	*NPAIR*†	10^{-4}	10^{-6}	10^{-4}	10^{-6}	10^{-4}	10^{-6}
4	3	10 (0·1)	15 (0·1)	23 (0·1)	35 (0·2)	40 (0·2)	60 (0·3)
9	3	7 (0·2)	10 (0·2)	10 (0·2)	15 (0·2)	10 (0·2)	15 (0·2)
9	9	37 (0·7)	55 (1·1)	54 (0·9)	79 (1·4)	66 (1·2)	94 (1·8)
18	3	6 (0·4)	8 (0·5)	6 (0·5)	9 (0·6)	7 (0·6)	10 (0·6)
18	9	26 (1·8)	39 (2·7)	30 (2·3)	43 (2·9)	34 (2·3)	48 (3·5)
18	18	66 (4·3)	98 (6·4)	77 (5·3)	111 (8·4)	89 (5·2)	122 (7·7)

† *NPAIR* = 3: (1, 2) (1, 3) (2, 4)
9: the previous 3 plus (5, 6) (6, 8) (7, 8) (2, 5) (3, 5) (4, 6)
18: the previous 9 plus (9, 11) (10, 11) (10, 17) (2, 9)
(3, 11) (3, 17) (4, 10) (5, 17) (6, 11).

ceased when the sum of the absolute values of the *NPAIR* concentrations was less than some prespecified value, *DELTA*. We chose correlation matrices with differing degrees of multicollinearity by taking $r_{ij} = 0{\cdot}2$, $0{\cdot}5$ and $0{\cdot}8$ for all $i \neq j$. Furthermore, we varied the size of the matrices (*NVAR* = 4, 9 and 18), the number of selected variable pairs with zero concentrations (*NPAIR* = 3, 9 and 18), and the desired precision in the zero concentrations (*DELTA* = 10^{-4} and 10^{-6}). Table 1 shows the number of iterations and the computing time in seconds needed on a CDC 3300 with word length 24 bits. For instance, it took 1·1 sec or 55 calls for *INVEST* to fit nine zero concentrations to the 9×9 correlation matrix with $r_{ij} = 0{\cdot}2$. Generally, the number of iterations increased—other things being equal—with an increase in *NPAIR*, and in the degree of multicollinearity, but it decreased with an increase in *DELTA* and in *NVAR*, the size of the matrix.

Another calling program for *INVEST* that fits the *NPAIR* variable pairs always in unchanged order and in full cycles of size *NPAIR* yielded comparable results in iterations and time for the 36 test cases. In situations with a very high degree of multicollinearity though, convergence was reached much later. Then, we recommend use of a double precision version of *INVEST* to allow the proper inversion of the fitted matrix. On the CDC 3300 computing time was increased by a factor of 10–15 for double precision.

Together with an inversion algorithm, that operates only on a triangular part of a matrix, *INVEST* allows the fitting of a covariance selection model to a correlation matrix by using only one storage array of dimension *NDIM*, *NVAR*.

Acknowledgement

We thank a referee for suggestions for improving the Fortran subroutine.

References

Bishop, Y. M. M. (1971). Effects of collapsing multi-dimensional contingency tables. *Biometrics*, **27**, 545–562.

Dempster, A. P. (1972). Covariance selection. *Biometrics*, **28**, 157–175.

Goodman, L. A. (1970). The multivariate analysis of qualitative data: interactions among multiple classifications. *J. Amer. Statist. Ass.*, **65**, 225–256.

Wermuth, N. (1976a). Analogies between multiplicative models in contingency tables and covariance selection. *Biometrics*, **32**, 95–108.

—— (1976b). Model search among multiplicative models. *Biometrics*, **32**, 253–263.

Wermuth, N., Wehner, T. and Gönner, H. (1976). Finding condensed descriptions for multi-dimensional data. *Computer Programs in Biomedicine*, **6**, 23–38.

```
      SUBROUTINE INVEST(MAT, NDIM, I, J, NVAR, IFAULT)
C
C        ALGORITHM AS 105  APPL. STATIST. (1977), VOL.26, NO.1
C
C        INVEST IS A SUBROUTINE FOR FITTING A COVARIANCE SELECTION
C        MODEL TO A MATRIX. IT MODIFIES THE UPPER TRIANGULAR PART
C        OF AN INVERSE MATRIX SO THAT ONE PRESPECIFIED OFF-DIAGONAL
C        ELEMENT EQUALS ZERO AND ONLY THE CORRESPONDING ELEMENT
C        IN THE ORIGINAL MATRIX IS ALTERED
C
C
      REAL MAT(NDIM, NVAR), D, SAVE1, SAVE2, MII, MIJ, MJJ
C
C        CHECK PARAMETERS
C
      IFAULT = 0
      IF(I .LT. 1 .OR. J .LE. I .OR. J .GT. NVAR) GOTO 13
C
C        SET COUNTERS
C
      I1 = I - 1
      I2 = I + 1
      J1 = J - 1
      J2 = J + 1
```

```
C        STORE VALUES
C
      MII = MAT(I, I)
      MJJ = MAT(J, J)
      MIJ = MAT(I, J)
      D = MII * MJJ - MIJ ** 2
C
C        POSITIONS (I, I), (J, J), (I, J)
C
      MAT(I, I) = D / MJJ
      MAT(J, J) = D / MII
      MAT(I, J) = 0.0
C
C        RESET VALUES
C
      MII = MIJ / MII
      MJJ = MIJ / MJJ
      MIJ = -MIJ / D
C
C        POSITIONS WITH  K AND L LESS THAN I
C
      IF(I .EQ. 1) GOTO 6
      DO 1 K = 1, I1
      SAVE1 = MAT(K, I)
      SAVE2 = MAT(K, J)
      MAT(K, I) = SAVE1 - MJJ * SAVE2
      MAT(K, J) = SAVE2 - MII * SAVE1
      MAT(K, K) = MAT(K, K) +
     *  MIJ * (MAT(K, I) * SAVE2 + MAT(K, J) * SAVE1)
      IF (K .EQ. I1) GOTO 2
      K1 = K + 1
      DO 1 L = K1, I1
      MAT(K, L) = MAT(K, L) +
     *  MIJ * (MAT(K, I) * MAT(L, J) + MAT(K, J) * MAT(L, I))
    1 CONTINUE
C
C        POSITIONS WITH  K LESS THAN I AND L BETWEEN I AND J
C
    2 IF (I2 .EQ. J) GOTO 4
      DO 3 K = 1, I1
      DO 3 L = I2, J1
      MAT(K, L) = MAT(K, L) +
     *  MIJ * (MAT(K, I) * MAT(L, J) + MAT(K, J) * MAT(I, L))
    3 CONTINUE
C
C        POSITIONS WITH  K LESS THAN I AND L GREATER THAN J
C
    4 IF (J .EQ. NVAR) GOTO 6
      DO 5 K = 1, I1
      DO 5 L = J2, NVAR
      MAT(K, L) = MAT(K, L) +
     *  MIJ * (MAT(K, I) * MAT(J, L) + MAT(K, J) * MAT(I, L))
    5 CONTINUE
C
C        POSITIONS WITH  K AND L BETWEEN I AND J
C
    6 IF (I2 .EQ. J) GOTO 10
      DO 7 K = I2, J1
      SAVE1 = MAT(I, K)
      SAVE2 = MAT(K, J)
      MAT(I, K) = SAVE1 - MJJ * SAVE2
      MAT(K, J) = SAVE2 - MII * SAVE1
      MAT(K, K) = MAT(K, K) +
     *  MIJ * (MAT(I, K) * SAVE2 + MAT(K, J) * SAVE1)
      IF (K .EQ. J1) GOTO 8
      K1 = K + 1
      DO 7 L = K1, J1
      MAT(K, L) = MAT(K, L) +
     *  MIJ * (MAT(I, K) * MAT(L, J) + MAT(K, J) * MAT(I, L))
    7 CONTINUE
C
C        POSITIONS WITH  K BETWEEN I AND J AND L GREATER THAN J
```

```
    8 IF (J .EQ. NVAR) GOTO 10
      DO 9 K = I2, J1
      DO 9 L = J2, NVAR
      MAT(K, L) = MAT(K, L) +
     *  MIJ * (MAT(I, K) * MAT(J, L) + MAT(K, J) * MAT(I, L))
    9 CONTINUE
C
C         POSITIONS WITH  K AND L GREATER THAN J
C
   10 IF (J .EQ. NVAR) GOTO 12
      DO 11 K = J2, NVAR
      SAVE1 = MAT(I, K)
      SAVE2 = MAT(J, K)
      MAT(I, K) = SAVE1 - MJJ * SAVE2
      MAT(J, K) = SAVE2 - MII * SAVE1
      MAT(K, K) = MAT(K, K) +
     *  MIJ * (MAT(I, K) * SAVE2 + MAT(J, K) * SAVE1)
      IF (K .EQ. NVAR) GOTO 12
      K1 = K + 1
      DO 11 L = K1, NVAR
      MAT(K, L) = MAT(K, L) +
     *  MIJ * (MAT(I, K) * MAT(J, L) + MAT(J, K) * MAT(I, L))
   11 CONTINUE
   12 RETURN
C
   13 IFAULT = 1
      RETURN
      END
```

Algorithm AS 106

The Distribution of Non-negative Quadratic Forms in Normal Variables

By J. Sheil[†] and I. O'Muircheartaigh

University College, Galway, Ireland

Keywords: QUADRATIC FORM; DISTRIBUTION; NON-NEGATIVE DEFINITE

Language

ISO Fortran

Description and Purpose

Given that the n-dimensional vector $\mathbf{z}$ has a multivariate normal distribution with expected value vector $\boldsymbol{\mu}$ and non-singular covariance matrix $\mathbf{V}$, this algorithm computes the distribution function of the quadratic form $(\mathbf{z}+\mathbf{a})^T \mathbf{C}(\mathbf{z}+\mathbf{a})$ for a fixed vector $\mathbf{a}$ and symmetric positive definite, or positive semi-definite, matrix $\mathbf{C}$. The value of the density is also presented in the output. The quadratic form is expressed as an infinite series in central χ^2 distribution functions: both the distribution functions and the series coefficients are evaluated recursively.

Theory and Numerical Method

n = dimensionality of $\mathbf{z}$.

By making the linear transformations

$$\mathbf{z}-\boldsymbol{\mu} = \mathbf{L}^T\mathbf{R}\mathbf{x}, \quad \mathbf{a}+\boldsymbol{\mu} = \mathbf{L}^T\mathbf{R}\mathbf{b},$$

[†] Now at University of Newcastle-upon-Tyne.