Computation of the Information Matrix for Models of Spatial Interaction

Oleg Smirnov

Regional Economics Applications Laboratory

University of Illinois at Urbana-Champaign

email: osmirnov@uiuc.edu

Abstract. I provide algebraic and computational considerations to alleviate the burden of computing the information matrix for models of spatial interaction. Of prime importance, I compute the information matrix without storing an inverse of the spatial differencing operator, a large sparse matrix. To accomplish this efficiently, I developed a version of the conjugate gradient method for the case of sparse matrices and vectors. Performance is demonstrated for various data sets and major types of models of spatial interaction. Its advantages include modest computing resource requirements, suitability for multiprocessing environments, and easily controlled precision. Convergence is guaranteed for all feasible values of the maximum likelihood estimates with the rate of convergence depending on the coefficient of spatial correlation. The efficiency of the proposed method allows complete maximum likelihood estimation of models with very large spatial data sets (one million observations or more).

Keywords: spatial statistics, maximum likelihood estimation, sparse conjugate gradient method

1. Introduction

Since as early as Whittle (1954), the method of maximum likelihood (ML) has been considered the preferred for the estimation of models of spatial interaction. Unlike least squares estimation that could be biased, inefficient, or both, exact maximum likelihood estimation is consistent and asymptotically efficient. The complete procedure of ML estimation involves two steps: obtaining ML estimates and computing statistical inference. Most efforts have been directed towards the first step, the computation of values of the parameters that maximize log-likelihood function (Griffith, 2002; Pace and Barry, 1997; Pace and LeSage, 2002; Smirnov and Anselin, 2001, among others). Mathematical and computational complexity of log-likelihood function is so overwhelming, that little attention has been paid to the second step—statistical inference. The second step, the computation of the information matrix and its inverse, however, is as important and highly desirable as the first step because it complements ML estimates of the parameters with information on their asymptotic variance, thence, statistical significance of the coefficients, testing model specification, etc.

While there are several practical computational methods of obtaining ML estimates, the computational problem of statistical inference for the models of spatial interaction has never been methodically investigated. Recent advances in computational methods enable to obtain ML estimates for models of spatial interaction for very large data sets (one million observations or more) using personal workstations. Those techniques, however, are useless for computing information matrix, because information matrix has little in common with log-likelihood. To date, no practical solutions have been offered for the computation of the information matrix.[1]

1. Study of Driscoll and Kraay (1998) demonstrates importance of consistent covariance matrix estimation and suggests use of Monte Carlo method in the context of generalized method of moments estimator.

Computation of the information matrix involves computation of inverse of a large sparse matrix. The dimension of the matrix is determined by the size of the sample, so computing its inverse becomes increasingly strenuous for large spatial data sets. Inverting a large matrix becomes impractical if not possible on personal workstations for matrices as little as a few thousand. In addition, straightforward utilization of existing formulas (Anselin, 1988; Anselin, Bera, 1998) for the information matrix suggest non-symmetry of the matrix to be inverted.

I offer a few considerations that substantially alleviate the computation of the information matrix. Similarity transformations applied to the spatial differencing operator replace non-symmetric matrix operations with symmetric ones; symmetry of the matrices allows use of computationally more efficient algorithms. Since spatial differencing operator is much more sparse than its inverse, its use in computational method is potentially more efficient. Finally, I develop a procedure for computing all the necessary quantities needed for the information matrix without storing the inverse of the spatial differencing operator. The procedure utilizes the conjugate gradient method, adopted for the case of sparse matrices and vectors. This method imposes fewer demands on computational environment than any other.

Performance of the method for computing information matrix is demonstrated for various data sets and major types of models of spatial interaction. The convergence of the method is guaranteed for all feasible values of the ML estimates, but the rate of convergence varies depending on the value of the coefficient of spatial correlation.

Major advantages of the proposed method include modest memory requirements, accommodation of sparse matrices of unusual structure without imposing additional demands on the computing resources, suitability for a multiprocessing environment, easily controlled accuracy.

Computational efficiency of the proposed method for computing the information matrix is critical for it allows to perform computations for sparse matrices up to a million of observation and more. The method also proves to be useful for improving the computational accuracy of the initial ML estimates (especially those obtained by an approximation technique).

2. Structure of the information matrix for SAR models

Ord (1975) originally provided information matrix for the spatial autoregressive model $y = \rho Wy + \varepsilon$. In practice, a full specification of the model with spatial interaction contains explanatory variables and a spatial autoregressive term in either ρWy (spatial lag model) or $\rho W\varepsilon$ (spatial error model) form, where ρ is a spatial autoregressive parameter, W is a $n \times n$ matrix of spatial weights, and y, ε are $n \times 1$ vectors of observations on the dependant variable and unobservable error terms. Benirshka and Binkley (1994) provided the information matrix for spatial error model; Anselin and Bera specified information matrices for both models.

Formally, spatial lag model is presented as

$$y = \rho Wy + X\beta + \varepsilon, \tag{1}$$

where W is a spatial weights matrix, $\varepsilon \sim N(0, \sigma^2 I)$ is a vector of random error terms. The log-likelihood function for the model is

$$L(\rho, \beta, \sigma^2) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{e'e}{\sigma^2} + \ln|I - \rho W|, \tag{2}$$

where $e = (I - \rho W)y - X\beta$ and the information matrix for the model is

$$I(\rho, \beta, \sigma^2) = \begin{bmatrix} tr(AA) + tr(A'A) + \dfrac{\beta' X' A' A X \beta}{\sigma^2} & \dfrac{\beta' X' A' X}{\sigma^2} & \dfrac{tr(A)}{\sigma^2} \\ \dfrac{X' A X \beta}{\sigma^2} & \dfrac{X'X}{\sigma^2} & 0 \\ \dfrac{tr(A)}{\sigma^2} & 0 & \dfrac{N}{2\sigma^2} \end{bmatrix}, \tag{3}$$

where $A = W(I - \rho W)^{-1}$.

Asymptotic variance matrix is an important component of statistical inference. Many statistics for models with spatial dependence such as asymptotic *t*-test for spatial and non-spatial coefficients require knowledge of the entire asymptotic variance matrix or some of its elements (for examples, see Anselin, Bera, 1998)[2]. Asymptotic variance can be obtained in the two-step procedure. In the first step, the information matrix is computed, and in the second, it is inverted. The latter is a routine computational task that can easily be performed for the entire information matrix or its blocks. Key issue is to obtain the information matrix, because it involves computation of several traces: *tr(A)*, *tr(AA)*, and $tr(A'A)$. In addition, element $I_{\beta\rho}$ contains matrix inverse.

Spatial error model is typically presented as

$$\begin{aligned} y &= X\beta + \varepsilon \\ \varepsilon &= \lambda W\varepsilon + \mu \end{aligned}, \tag{4}$$

where $\mu \sim N(0, \sigma^2 I)$ is a vector of error terms. Log-likelihood function for this model is

$$L(\lambda, \beta, \sigma^2) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{e_d{}'e_d}{2\sigma^2} + \ln|I - \lambda W|, \tag{5}$$

where $e_d = (I - \lambda W)(y - X\beta)$ and the information matrix is

$$I(\lambda, \sigma^2, \beta) = \begin{bmatrix} tr(AA) + tr(A'A) & \frac{tr(A)}{\sigma^2} & 0 \\ \frac{tr(A)}{\sigma^2} & \frac{N}{2\sigma^2} & 0 \\ 0 & 0 & \sigma^2[X_d'X_d] \end{bmatrix}, \tag{6}$$

where $A = W(I - \lambda W)^{-1}$.

The information matrix (6) is block-diagonal, so the terms containing traces of matrix inverses are needed only in the block corresponding to λ and σ^2. The information matrix for spatial lag

2. A rare exception is likelihood ratio statistic, which is computed using values of log-likelihood function only

model is somewhat more challenging because it is not diagonal, hence must be always computed in full, and contains matrix inverse in element $I_{\beta\rho}$.

3. Approaches for computing the information matrix

The straightforward computation of inverse of a large matrix is a well-studied problem of linear algebra (Golub, Van Loan, 1996). It is an $O(n^3)$ operation, that imposes burdensome requirements on fast-access memory. In practice, the size of data set for which this operation can be effectively computed on a typical workstation is limited to a few thousand observations. Since the computational complexity and memory requirements for the problem are highly non-linear, it cannot be effectively solved for those data sets that exceed in size a level specific for each computational system. For large data sets (hundred thousand observations or more), it is impractical to solve on a personal workstation if possible at all. While advances in modern computational techniques and progress in computing environments do alleviate this problem, the needs of spatial statistical analysis and handling of large-sized data grow even faster (see Pace and LeSage, 2002; Smirnov and Anselin, 2001 for examples of large spatial datasets).

An obvious solution to reduce the computational burden and memory requirements for computing traces of inverse of a sparse matrix is to exploit its sparsity. Common feature of sparse matrix factorization algorithms is a reliance on ordering of rows and columns to reduce the fill of the resulting factor matrix[3]. Ideally, the resulting factor matrix would have no fill while preserving the numerical characteristics of the original matrix (see, George and Liu, 1981; Gilbert, Moler, and Schreiber, 1992). Situation of the no fill is possible, for example, for a banded matrix,

3. Good ordering routines even can reduce the complexity of the problem for a typical spatial weights matrix from $O(n^3)$ down to $O(n^2)$ (Gilbert, Moler, and Schreiber, 1992; MathSoft, 1996; Ng and Raghavan, 1999).

when all non-zero elements are concentrated along main diagonal[4]. Such matrices represent a very specific case of spatial layout, which rarely found in practical applications. For example, spatial weights matrix that corresponds to a regular grid of $n \times n$ would have a band approximately of size *n*. The size of a band would increase with the size of the matrix, and so would the fill and the computational complexity of the Cholesky factorization and the matrix inverse.

The fill of the matrix depends not as much on the number of neighbors in the spatial weights matrix, as on how local they are. Optimal ordering for regular grids produces fill of order $O(n \ln n)$ and total number of operations to perform factorization is $O(n\sqrt{n})$. In comparison, fill for the banded matrix is zero, and its factorization is a linear complexity operation. In sum, typical spatial weights matrices tend to produce larger fill and their factorization requires more operations than sparse matrices with a band or other no-fill and small-fill structure. Therefore, sparse matrix factorization routines should be expected slower for spatial matrices and memory requirement are a factor that makes it impossible to perform factorization of large spatial weights matrices.

Most computational solutions for sparse matrix factorization rely heavily on various strategies to reduce the fill and, consequently, the complexity of sparse matrix factorization. These methods, however, do not give an information on the memory requirements for storing the Cholesky factor of a sparse matrix until after the ordering or symbolic factorization is actually performed, lest the number of operations. The knowledge of the memory prerequisites is critical for determining if the factor matrix fits into the constraints of the particular computational environment. Such uncer-

4. For spatial weights matrix *W* that has a band with width *b*, i.e. $w_{ij} = 0$ for all $i > j + b$ and $i < j - b$, Cholesky factor of matrix $I - \rho W$ is also a banded matrix with width *b*.

tainty places factorization methods of arbitrary sparse matrices in the category of methods that might produce a result and might fail.

4. A Conjugate Gradient Method

4.1. Principle

An alternative approach is to compute column-vector $(I-\rho W)^{-1}X\beta$ using implicit matrix inverse. The idea is to find a vector z that satisfies the following condition:

$$(I-\rho W)z = X\beta. \tag{7}$$

Once such vector is found, the element $I_{\rho\beta}$ of the information matrix is computed using only sparse matrix-vector multiplication $I_{\rho\beta} = \frac{1}{\sigma^2}X'Wz$.

Solving system of linear equations (7) without explicitly computing matrix inverse is a well-known linear algebra problem (see Golub, Van Loan, 1996, chapter 10; Watkins, 2002, chapter 7). Key advantage of iterative methods over the direct Cholesky factorization is that computing the inverse is unnecessary and memory requirements are very modest. For the methods to be computationally efficient, the only requirement is efficient implementation of sparse matrix-vector multiplication. Row and column re-ordering or any form of symbolic computation which is a major time-consuming operation in sparse matrix factorization also is not required.

The fastest best-known iterative methods for solving the system of linear equations are descend and conjugate gradient methods. Both methods are iterative. The former is easier to implement, but the latter has better convergence rates and is guaranteed to converge to the solution within a finite number of iterations. For these reasons, I focus on conjugate gradient method. Its complete description and proves of the properties can be found in (Golub, Van Loan, 1996:527-8; Watkins, 2002, 577-8).

Conjugate gradient method is an iterative procedure when an initial value of z is sequentially updated until vector $(I - \rho W)z$ becomes close enough or computationally indistinguishable from the desired target, i.e., $X\beta$. Update vector d is chosen in a such a manner that it is conjugate to the previous update vectors. This ensures that the iterative process never proceeds in the repetitive directions and that the method stops after a finite number of iterations. Each iteration involves computation of $(I - \rho W)d$, which is a sparse matrix-vector operation and can be implemented efficiently. Other operations include computing vector norm, multiplication by a scalar and vector subtraction which are even less computationally demanding. The convergence is achieved when the update vector $(I - \rho W)d$ becomes too small, i.e., has a small norm and further updates do not have any impact on the solution vector z. Intermediate results are a few vectors and no other memory is needed.

The convergence rates of the conjugate gradient method depend on the properties of matrix $I - \rho W$. The upper bound of the rate is determined by the condition number (ratio of the largest eigenvalue to the smallest one) of matrix $I - \rho W$ (Watkins, 2002). The higher the condition number, the higher the convergence rates and the fewer the iterations. In practice, in the case of spatial weights matrices, the convergence is much faster and inversely related to the condition number, i.e. the value of coefficient ρ, actually, is the only factor[5] that determines the rate of convergence. Values of ρ close to zero produce a matrix with condition number close to one, its smallest value, but it takes only a few iterations, normally five or size, to compute vector z. Larger values of ρ lead to the matrices with higher condition numbers, but slower convergence rates. For example, in the presence of strong spatial association, when ρ is equal to 0.5, it requires typically 30 to 40 iterations to arrive at the accurate solution. Higher values of ρ imply larger number of iterations.

5. The size of the matrix $I - \rho W$ does not affect the convergence rates beside providing an upper bound on the number of iterations.

Since higher values of ρ are extremely rare for the correctly specified models with spatial interaction, normally it would take 50 or fewer iterations to achieve the convergence.

4.2. Computing information matrix for originally symmetric spatial weights matrix

Consider cases where matrix W is symmetric or obtained by row-standardizing a symmetric matrix W_{orig}. In the latter event, row-standardization can be expressed as $W = D^{-1}W_{orig}$, where D is a diagonal matrix with diagonal elements equal to the sum of the elements in the corresponding row of the original spatial weights matrix. Although row-standardization of the matrix W_{orig} yields asymmetric matrix, the latter can be easily transformed into a symmetric one by a similarity transformation $W_s = D^{1/2}WD^{-1/2}$. It can be readily demonstrated that the traces of the matrix W are the same as those of the matrix W_s. In this respect, symmetry can be utilized to make the computations of trace terms more efficient as

$trW_s = trW$, $trW_s(I - \rho W_s)^{-1} = trW(I - \rho W)^{-1}$, and

$trW_s(I - \rho W_s)(I - \rho W_s)'W_s' = trW(I - \rho W)(I - \rho W)'W'$.

Note that computing trace of matrix S is equivalent to computing the sum of scalars, which can be done without explicit matrix inverse

$$trA = trS = trSI = trS\sum_{i=1} e_i e_i' = \sum_{i=1} e_i'Se_i\ , \qquad (8)$$

where $S = W_s(I - \rho W_s)^{-1}$, e_i is an *i*-th ord, i.e., a column-vector with 1 at *i*-th row and zeros elsewhere. Formally, computing $e_i'Se_i = e_i'W_s(I - \rho W_s)^{-1}e_i$ is equivalent to computing $e_i'W_sz_i$, where z_i is the solution of the system of linear equations $(I - \rho W_s)z_i = e_i$. The system is free from an explicit matrix inverse and can be effectively solved by conjugate gradient or another iterative method. Computing $e_i'Se_i$ is essentially reduced to the computation of the *i*-th component of vector $v_i = Se_i$. The advantage of conjugate gradient methods is that vector v_i can also be used in computing other trace terms.

Follow symmetry consideration as in the (8), the treatment of the term $trAA$ is also significantly simplified

$$trAA = trSS = trSSI = trSS\sum_{i=1} e_i e_i' = \sum_{i=1} e_i'SSe_i = \sum_{i=1} \|Se_i\|^2, \qquad (9)$$

where $\|x\|$ denotes Euclidean norm of a vector. Computing (9) is further simplified using vector $v_i = Se_i$ already computed in (8). The only necessity is to compute the square of its norm.

Term $trAA'$ represents trace of a symmetric matrix AA', which is not equal to $trSS'$. Use of $A = W(I - \rho W)$ and $W = D^{-1/2}W_s D^{1/2}$ leads to

$trAA' = trD^{-1}W_s(I - \rho W_s)^{-1}D(I - \rho W_s')^{-1}W_s' = trD^{-1}SDS$.

Similarly, using symmetry of matrix *S*, $trSS'$ can be presented as

$$trD^{-1}SDS = trD^{-1}SIDS = tr\sum_{i=1} D^{-1}Se_i e_i' DS = \sum_{i=1} e_i' DSD^{-1}Se_i. \qquad (10)$$

Since matrix *D* is diagonal and e_i' has all zeros except at the *i*-th component, the use of $e_i'D = d_i e_i'$ further simplifies (10):

$$\sum_{i=1} e_i' DSD^{-1}Se_i = \sum_{i=1} e_i' Sd_i D^{-1}Se_i. \qquad (11)$$

Re-use vector $v_i = Se_i$ from (8), and (11) becomes

$$trAA' = \sum_{i=1}\sum_{j=1} \frac{d_i}{d_j} v_{ij}^2. \qquad (12)$$

The outlined approach for computing *trA*, *trAA*, and $trAA'$ uses conjugate gradient method and does not require storing of the inverse of matrix $I - \rho W$. The main advantages of the approach are low computer memory requirements on all stages, no need for symbolic computation or reordering of rows and columns of matrix *W* and suitability for multiprocessing computing environments. The major computational task of the method is computing vector

$v_i = W_s(I - \rho W_s)^{-1} e_i$, $i = 1, \ldots, N$. The fact that this task must be performed for each i, emphasizes the need for finding more efficient ways for computing vector v_i.

4.3. Computing information matrix for originally non-symmetric spatial weights matrix

Non-symmetry of the original spatial weights matrix requires additional manipulations to reduce the problem of computing traces. The simplest solution is to replace the non-symmetric system of linear equations $(I - \rho W)z_i = e_i$ with the equivalent symmetric system $(I - \rho W')(I - \rho W)z_i = (I - \rho W')e_i$. Although he latter uses symmetric matrix $(I - \rho W')(I - \rho W)$, it would take two matrix-vector multiplications on each iteration to compute solution update.

Once the system of linear equations $(I - \rho W')(I - \rho W)z_i = (I - \rho W')e_i$ is solved using conjugate gradient method, equation

$$trA = \sum_{i=1}^{N} e_i' A e_i \tag{13}$$

becomes a simple matter of extracting i-th component from vector z_i:

$trA = \sum_{i=1}^{N} e_i' z_i$.

Computation of *trAA* requires additional step of solving $(I - \rho W)(I - \rho W')z_i^w = (I - \rho W)W' e_i$. Further, z_i and z_i^w are used to compute the desired trace term:

$$trAA = \sum_{i=1}^{N} z_i' W' z_i^w . \tag{14}$$

Term $trAA'$ can be computed as

$$trAA' = \sum_{i=1}^{N} z_i' W' W z_i = \sum_{i=1}^{N} \|Wz_i\| . \tag{15}$$

Equations (13)-(15) constitute the basis for computing the elements of the information matrix if the spatial weights matrix is not symmetric. Since they involve more computations than (8), (9), and (12), it is always advisable to use symmetric cases whenever possible.

5. Sparse Conjugate Gradient Method

For a given matrix $I - \rho W$, computing vector $v_i = W_s(I - \rho W_s)^{-1} e_i$ is a linear computational complexity process, i.e., the larger the dataset, the higher the computational burden. Since there are *N* vectors to be sequentially computed, the total computational complexity of computing traces *trA*, *trAA*, and $trAA'$ becomes $O(N^2)$ which is comparable to the computational complexity of performing sparse Cholesky decomposition for a typical spatial weights matrix. Sparse conjugate gradient method allows to reduce the computational complexity of computing traces from a quadratic to linear. This is achieved by utilizing the specifics of computing the trace terms -- sparsity of vectors e_i.

The starting point of the sparse conjugate gradient method is that vectors *e* are sparse, i.e. they contain only single non-zero component. Initial value for vector *z* is *e*. Vector of residuals $r^0 = \rho We$ is also sparse. At each iteration, the direction d^k for updating vector *z* is chosen as $d^k = r^{k-1} + \alpha d^{k-1}$, where α is the ratio of squared norms of vectors r^{k-1} andr^{k-2}. Consequently, vector of residuals r^0 is updated by adding a multiple of vector $(I - \rho W_s)d^k$.

The iterative nature of conjugate gradient method contributes to the phenomenon, that the number of non-zero entries in vectors *z*, *r*, and *d* increases. The set of non-zero entries is expanded by adding elements in their immediate neighborhood at each iteration. Thus, if vector d^0 contains only one non-zero entry, which corresponds to location *i*, vector d^1 will have first order neighbors to *i*, vector d^2 will have second order neighbors to *i* and so on. Adding spatial neighbors to the sparse vector does not alter its sparsity, because most first order neighbors to non-zero entries have been already included.

In a typical spatial weights matrix, the number of non-zero entries in the higher order neighbors is not dependent on the size of the data set; it rather depends on the criteria used to define locality and neighborhood conditions. For example, in a banded matrix with bandwidth *b*, the number of non-zero entries in vector d^k is equal to $b(k+1)+1$. For a spatial weights matrix computed over a regular grid with rook criteria of contiguity, the number of non-zero entries in vector d^k is no more than $2(k+1)^2+1$. While this number increases with the number of iterations, it remains negligibly small for large data sets. Thus, the significant improvement to the performance of the conjugate gradient method can be done by replacing sparse matrix—dense vector multiplication $(I-\rho W_s)d^k$ requiring *bN* arithmetic operations with a sparse matrix—sparse vector application requiring *bp* arithmetic operations, where *b* is the average number of non-zero elements in matrix $I-\rho W$ and p is the total number of non-zero elements in vector d^k. For example, for a spatial weights matrix originating from a regular grid with the rook criteria of contiguity and size of the data set N=100,000, the original implementation requires 400,000 operations and the improved - less than 20,004 operations for $k = 49$. The difference becomes more dramatic for larger data sets, because the latter does not change significantly with the size of the data set.

The efficient implementation of the sparse conjugate gradient method depends on the efficient implementation of sparse vector operations. The major gain in performance of sparse operations over the dense vector operations is attributed to elimination of unnecessary operations with components that are known to be zeros. Thus, sparse vector operations have to keep track of the non-zero components, i.e., to identify which components differ from zero and what are the values in these components. Vector operations of addition and multiplication are carried out only with the non-zero components.

Data structures for efficient sparse vector operations should be efficient yet flexible enough to accommodate the dynamic nature of most sparse vectors used in the sparse conjugate gradient method. Intermediate sparse vectors change form iteration to iteration and the number of non-zero entries in these vectors typically increases. The number of non-zero elements in a vector is unlikely to reach the upper limit, *N*. The best data structure should contain the list of non-zero entries in the vector, so that the major vector operations would involve only non-zero vector components. In addition, it should provide the efficient vector update, i.e., insertion of new non-zero elements or updating the value of the existing non-zero entry.

I use regular one-dimensional array to store the values of elements of the vector which is accompanied by the list of indices of non-zero elements in the vector. While such combination is not the most efficient from point of view of memory requirement, it is the most computationally efficient, because it provides a constant computational complexity for operations of inserting, accessing and updating single element and enables to list all non-zero elements bypassing zero components.

6. Numerical experiments

The main purpose of the numerical experiments is to test viability of the sparse gradient method for computing the elements of the information matrix for the models with spatial interaction.

First, I want to establish if the method is practical for computing the information matrix for large data sets in the regular workstation environment. This issue is wider than just confirming that a particular computational problem can be solved given the limitations on computational resources of a specific workstation. After all, the notion of *modern* workstation has being constantly upgraded and advances in computing techniques relax the limitations on the size and complexity of the problems that can be solved. However, the notions of a *typical* or *large* spatial data sets are inflating even faster. For these reasons, the practicality of the method depends on how an increase in the size of the data set affects the method s performance.

The experiment that aim to establish how the data set size affects the performance of the sparse conjugate gradient method has to include several spatial weights matrices that preferably differ in the size and not other characteristics (average number of neighbors, presence or absence of spatial clusters, connectivity, etc.). Spatial weights matrices generated over the regular grids give the best fit for the purpose. I used rook criteria of contiguity[6] to define spatial neighbors. This criterion results in a symmetric spatial weights matrix with 2, 3, or 4 neighbors per observation. The size of regular grid determines the number of observations and, consequently, the size of the data sets and the dimension of spatial weights matrix.

The series of regular grids presented in Table 1 cover a range of spatial data sets. The smallest data set has 1,024 and the largest has 3,001,556 observations. The spatial weights matrices are originally symmetric binary sparse matrices. Explanatory and dependent variables are generated in such a way that the model to be estimated had a given true value of the coefficient of spatial association.

According to Table 1, the timing of the method increases with the size of the matrix[7]. It is increasing a bit faster than the theoretically established linear complexity of the method suggests,

6. i.e., two cells on the regular rectangular grid are neighbors if and only if they have a common boundary
7. All computations have been performed on a Macintosh computer equipped with PowerPC G4 processor rated at 800 MHz and 512 MB installed memory, operating system - Mac OS X. Programming language is C++.

Table 1: Performance of sparse conjugate gradient method for computing the Information matrix for various spatial layouts, $\rho = 0.25$

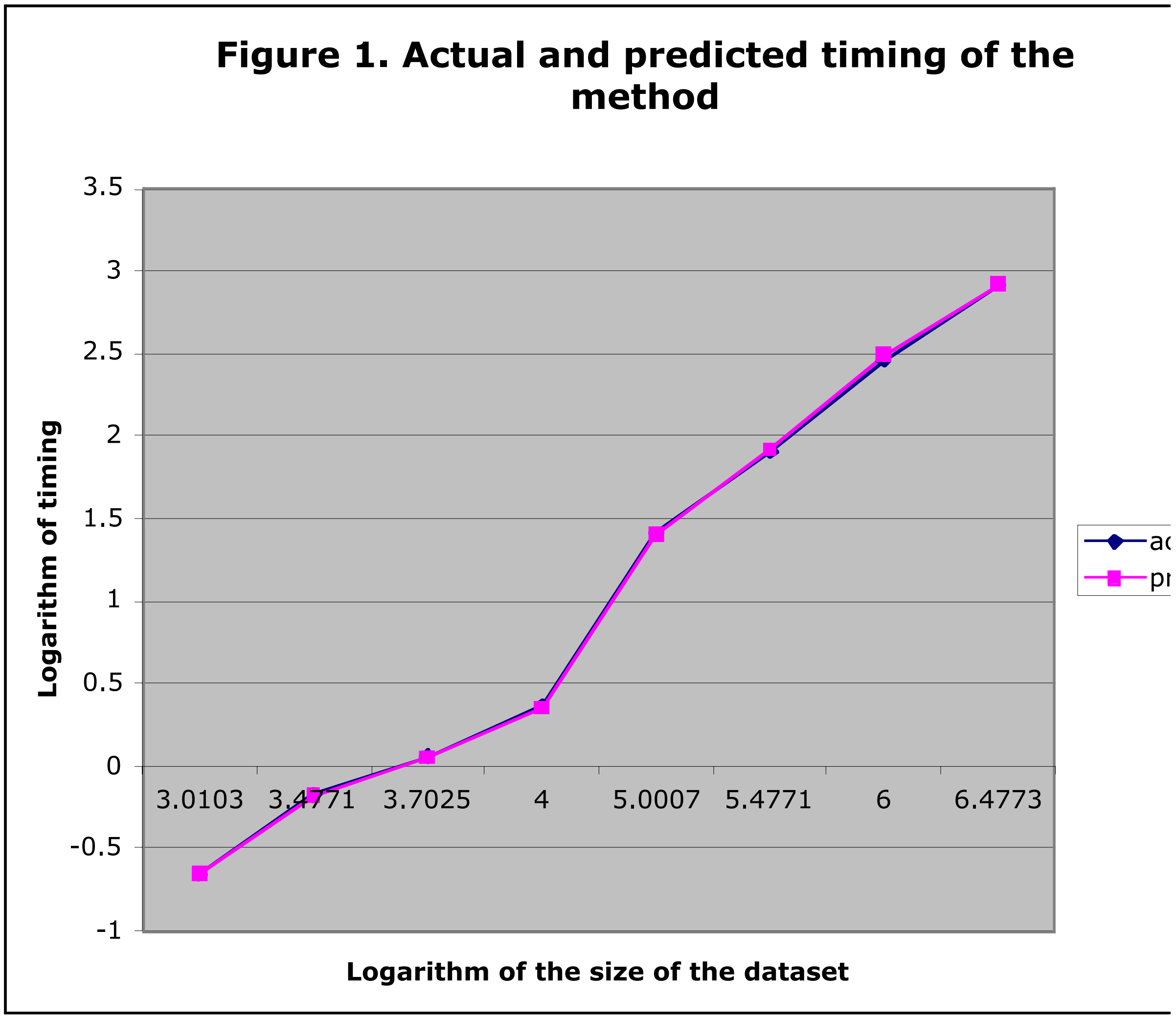
Regular grid	Data set size	Non-zero elements in the spatial weights matrix	Time, seconds	Performance, rows/sec.
32 x 32	1,024	3,968	0.22	4,655
50 x 60	3,000	11,780	0.67	4,478
71 x 71	5,041	19,880	1.15	4,383
100 x 100	10,000	39,600	2.32	4,310
316 x 317	100,172	399,422	25.46	3,935
500 x 600	300,000	1,197,800	80.88	3,709
1,000x1,000	1,000,000	3,996,000	291.08	3,435
1,732x1,733	3,001,556	11,999,294	835.50	3,593[a]

a. Computations have been performed over night; fewer running applications result in lower CPU load and memory segmentation.

but slower than the rates of $O(n \ln n)$ or $O(n \ln \ln n)$. It appears that the best curve to fit the timing for the method in Table 1 is $O(n\sqrt{\ln \ln n})$. The actual timing and predicted timing are depicted in Figure 1. Both horizontal and vertical axes in Figure 1are correspondingly the logarithms of the size of the data sets and timing. Visually, the predicted values for $O(n\sqrt{\ln \ln n})$ are practically identical to the actual values of logarithm of timing.

The evidence in Table 1 and Figure 1 indicates that actual timing of the sparse conjugate gradient method is $O(n\sqrt{\ln \ln n})$ rather than a theoretical $O(n)$. This raises two important questions: why does this occur and what factors affect practical performance of the sparse conjugate gradient method?

Theoretical linear computational complexity of the method relies on the assumption that the memory access operations are scale-independent. In practice, common hierarchical structure of random access memory (via one or more levels of cache) in modern computers makes smaller



tasks disproportionately easier to accomplish than the larger ones. The larger tasks require more frequent access to slower layers of memory, which increases the execution time even if the floating point operations are not affected. Nonetheless, the overhead associated with memory access is relatively modest; as Table 1 indicates, tenfold increase in the size of the data set causes the performance of the method to slow down by less than 15 percent.

The answer on the second question depends on the size of the problem. For smaller problems, when the spatial weights matrix and most of the intermediate vectors fit into faster access cache, the computational power of the processor is important and RAM architecture and bus speed are of lesser significance. For larger problems, the computational power of the processor becomes less important because the memory access becomes the limiting factor and the processor is mostly idling. Thus, the systems with slower processor but faster bus and better architecture can outperform those with faster processor and slower memory access.

Other factors that affect the performance of the sparse conjugate gradient method include the number of non-zero elements in the spatial weights matrix and the value of the coefficient of spatial association. Computing spatial lag of a sparse vector is the major operation of the method. The fewer non-zero entries in the spatial weights matrix, the faster the computation of the spatial lag.

The importance of the value of the coefficient of spatial association for the convergence of the spatial conjugate gradient method is demonstrated in Table 2. The timing data in Table 2 are obtained by computing the information matrix for the model with 100,172 observations and various values of the coefficient of spatial association. While I compute the information matrix for all the values listed in Table 2, it should be noted, that in practice the accurate computation of the information matrix is desirable mostly for lower values of the coefficient (less than 0.5), because higher values of coefficient are rare in correctly specified models.

It takes approximately 56 seconds to obtain ML estimates using characteristic polynomial method described in (Smirnov, Anselin, 2001). Using this timing as a benchmark, the sparse conjugate gradient method is reasonably fast for vast majority of practical applications when spatial coefficient is not exceeding 0.7. As Table 2 indicates, the performance of sparse conjugate gradient method dramatically deteriorates for extremely high values of the coefficient of spatial correlation (0.7 and higher). The simplest way to reduce the timing for such cases is to find a reasonable compromise between the timing and the accuracy. The iterative nature of the method

Table 2: Performance of the method for various values of spatial correlation coefficient[a]

Coefficient	Time, seconds
0.05	8.66
0.10	13.02
0.15	18.39
0.25	25.46
0.35	43.96
0.50	87.73
0.70	245.20
0.90	1,290.86

a. The spatial weights matrix is obtained from a regular grid with 316 row and 317 columns using rook criterion of contiguity.

provides a base for such a compromise; the iterations can be terminated much earlier than in the exact.version of the method. The termination condition in the exact method stipulates that the squared norm of the correction vector is less than 1×10^{-16}. This ensures exactness of the information matrix. However, if the exact values are not needed and a good approximation suffices, less stringent termination condition reduces the number of iterations and decreases timing. For example, terminating value of 1×10^{-8} reduces timing from 1,290.86 for the exact method with correlation 0.9 to just 152.83 seconds while ensuring four significant digits in the results. Terminating value 1×10^{-6} further reduces timing to 70.32 seconds and provides only three significant digits in the result. Memory requirements for exact and approximate versions are the same.

7. Conclusion

In this paper, I identify the computation of trace terms of the inverse of spatial difference operator as the key issue for computing the information matrix for either spatial lag and spatial error models. While the formulas needed for computing the information matrix are well-known, they cannot be easily applied for arbitrary large spatial problems. The conventional methods for computing such terms impose substantial requirements on the computational resources, CPU and memory. Excessive demands for the resources preclude efficient application of existing methods for solving models with large spatial data sets. In addition, the requirements for sparse matrix factorization methods are unknown in advance, thus introducing additional inconveniencies in the practice of estimating models with spatial interaction.

The major contribution of this paper is in devising and analyzing an innovative approach for computing the information matrix for the models with spatial interaction. The core of the approach is build on sparse conjugate gradient method for computing necessary trace terms and other quantities. Unlike sparse or dense matrix inverse methods, it does not explicitly store and compute the inverse of a large matrix. Small memory requirements of the method enable computation of the information matrix for large spatial datasets, up to 1 million observations and more. Computational efficiency and theoretical linear computational complexity of the sparse conjugate gradient method allow to equally effectively deal with small and large models. In addition, the method can be deployed in the multiprocessing or distributed computing environments.

Numerical experiments have been conducted to test practicality of the sparse conjugate gradient method on a series of spatial data sets ranging in size from 1,024 to 3,001,556 observations. It has been established that the actual timing of the method is close to linear complexity, with performance declining less than 15 percent when the size of the problem rises tenfold. The actual timing is described by $O(n\sqrt{\ln \ln n})$ function, which can be explained by the hierarchical structure of memory and includes a modest penalty for accessing slower layers of memory.

Sparse conjugate gradient method is an effective solution that is modest in computational requirements and applicable to small and large scale problems. The only limitation of the exact version of the method is associated with extremely high (and rare high) values of the coefficient of spatial association (greater than 0.7). The timing, but not memory requirement or accuracy of the method, substantially increase in such cases. When the value of the coefficient is known and a good approximation of the information matrix is sufficient, the approximate version of the method can be utilized. Flexibility of the sparse conjugate gradient method allows to control the accuracy of the method and to compute approximate information matrix. Less precise results can be computed faster.

In sum, unlike sparse matrix inverse methods, sparse conjugate gradient method is predictable, modest in requirements, has linear computational complexity (i.e., very fast), allows for control of the accuracy of the results, and is easy to implement in single- or multiprocessing environments.

References

Anselin, Luc, 1988. *Spatial Econometrics: Methods and Models*. Kluwer Academic, Dordrecht, Netherlands.

Anselin, Luc, 1992. *SpaceStat, a Software Program for the Analysis of Spatial Data*. National Center for Geographic Information and Analysis. University of California, Santa Barbara, CA.

Anselin, Luc and Anil K. Bera. 1998. Spatial dependence in linear regression models with an introduction to spatial econometrics. In: Ullah, A., Giles, D.E. (Eds.), *Handbook of Applied Economic Statistics*, Marcel Dekker, New York, pp. 237-289.

Benirschka, M. and J. K. Binkley. 1994. Land Price Volatility in a Geographically Dispersed Market, *American Journal of Agricultural Economics*, 75, 185-195.

Cressie, Noel, 1993. *Statistics for Spatial Data*. *Wiley*, New York, NY.

Driscoll, J. C. and A. C. Kraay. 1998. Consistent covariance matrix estimation with spatially dependent panel data, *The Review of Economics and Statistics*, Vol. 80, pp. 549-560.

George, A. and J. W. H. Liu. 1981. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, NJ.

Gilbert, John R., Cleve Moler, and Robert Schreiber, 1992. Sparse matrices in MATLAB: design and implementation, *SIAM Journal on Matrix Analysis and Applications*, Vol. 13, No. 1:333-356.

Golub, Gene H. and Charles F. Van Loan, 1996. *Matrix Computations*. Third Edition. John Hopkins University Press.

Griffith, Daniel A., 2000. Eigenfunction properties and approximations of selected incidence matrices employed in spatial analyses, *Linear Algebra and its Applications*, Vol. 321, Issues 1-3: 95-112.

Haining, Robert, 1990. *Spatial data analysis in the social and environmental sciences*. Cambridge University Press, Cambridge, United Kingdom.

Mathsoft, 1996. *S+SpatialStats Users's Manual for Windows and Unix*. Mathsoft, Seattle, WA.

Ng, G. Esmond and Padma Raghavan. 1999. Performance of greedy ordering heuristics for sparse Cholesky factorization, *SIAM Journal on Matrix Analysis and Applications*, Vol. 20, No. 4: 902-914.

Ord, Keith. 1975. Estimation methods for models of spatial interaction, *Journal of the American Statistical Association*, Volume 70, Number. 349: 120-126.

Pace, R. Kelley and Ronald Barry. 1997. Sparse spatial autoregressions, *Statistics and Probability Letters*, Vol. 33, Issue 3: 291-297.

Pace, R. Kelly and James P. LeSage. 2002. Semiparametric maximum likelihood estimates of spatial dependence, *Geographical Analysis*, 34:76-90.

Smirnov, Oleg and Luc Anselin. 2001. Fast maximum likelihood estimation of very large spatial autoregressive models: a characteristic polynomial approach, *Computational Statistics and Data Analysis*, Vol. 35, Issue 3:301-319.

Watkins, David S., 2002. *Fundamentals of Matrix Computations*. Second Edition. Wiley.

Whittle, P., 1954. On stationary processes in the plane. *Biometrik*a, Vol. 41: 434-49.