We would like to thank Elsevier, the owner of the copyright, for having granted us copyright permission to redistribute the following manuscript:

Barry, Ronald, and R. Kelley Pace, "A Monte Carlo Estimator of the Log Determinant of Large Sparse Matrices," *Linear Algebra and its Applications*, Volume 289, Number 1-3, 1999, p. 41-54.

Updated contact information subsequent to publication of the manuscript:

Ronald Barry

Associate Professor of Mathematical Sciences

University of Alaska

Fairbanks, Alaska 99775-6660

(907)-474-7226

FAX: (907)-474-5394

FFRPB@aurora.alaska.edu

and

R. Kelley Pace

LREC Endowed Chair of Real Estate

E.J. Ourso College of Business

Louisiana State University

Baton Rouge, LA 70803-6308

(225)-388-6256

FAX: (225)-334-1227

kelley@spatial-statistics.com

kelley@pace.am

ELSEVIER Linear Algebra and its Applications 289 (1999) 41–54

LINEAR ALGEBRA AND ITS APPLICATIONS

Monte Carlo estimates of the log determinant of large sparse matrices

Ronald Paul Barry [a,*], R. Kelley Pace [b]

[a] *Department of Mathematical Sciences, University of Alaska Fairbanks, Fairbanks, AK 99775-6660, USA*

[b] *Department of Finance, E.J. Ourso College of Business Administration, Louisiana State University, Baton Rouge, LA 70802, USA*

Received 8 November 1996; accepted 26 June 1997

Submitted by G.P.H. Styan

Abstract

Maximum likelihood estimates of parameters of some spatial models require the computation of the log-determinant of positive-definite matrices of the form $\mathbf{I} - \alpha\mathbf{D}$, where $\mathbf{D}$ is a large, sparse matrix with eigenvalues in $[-1, 1]$ and where $0 < \alpha < 1$. With extremely large matrices the usual direct methods of obtaining the log-determinant require too much time and memory. We propose a Monte Carlo estimate of the log-determinant. This estimate is simple to program, very sparing in its use of memory, easily computed in parallel and can estimate $\log\det(\mathbf{I} - \alpha\mathbf{D})$ for many values of α simultaneously. Using this estimator, we estimate the log-determinant for a 1,000,000 $\times$ 1,000,000 matrix $\mathbf{D}$, for 100 values of α, in 23.1 min on a 133 MHz pentium with 64 MB of memory using Matlab.

Keywords: Dirichlet distribution; Eigenvalues; Maximum likelihood; Normalizing constant; Spatial autocorrelation; Spatial statistics

1. Motivation

Determinants arise out of a number of contexts in applied linear algebra. Our particular application arises out of the method of maximum likelihood

* Corresponding author. Tel.: +1 907 474 7226; fax: +1 907 474 5396; e-mail: ffrpb@aurora.uaf.edu.

0024-3795/99/$ – see front matter
PII: S0024-3795(97)10009-X

that lies at the heart of statistics. The multivariate maximum likelihood function involves the determinant of the variance–covariance matrix (or equivalently some power of the variance–covariance matrix such as $-1, 1/2$, or $-1/2$). For non-independent data, this positive-definite matrix can have zero entries in any non-diagonal position and positive elements on the main diagonal. This matrix has dimension of $n \times n$, where n represents the number of observations. Actual applications create large matrices. For example, the US Census Bureau collects data at over 250,000 census block group locations.

The canonical way of computing determinants requires the use of the Cholesky decomposition for the symmetrical variance–covariance matrix or perhaps the LU decomposition for the case of the variance–covariance matrix to the one-half power (potentially asymmetric). Fortunately, many applications involve rather sparse matrices (Barry and Pace, 1997, and Pace and Barry, 1998). However, the direct sparse matrix methods used to compute the decompositions depend upon a favorable ordering of the elements to achieve computational speed. Unfortunately, for statistical applications as n rises, it appears difficult to always achieve favorable ordering. We have observed the computation times rise at a rate substantially faster than n, a factor we attribute to the difficulty of finding one ordering appropriate for an increasingly heterogeneous collection of patterns in the data. The direct decompositions attempt to provide exact estimates of determinants. However, statistical applications may not require such exactitude. This suggests the route of finding an approximate estimate of the determinant, a route we follow in this paper.

Griffith and Sone (1995, p. 168) argued persuasively for the use of approximations of the determinant:

> ... there are three main reasons for wanting to approximate the normalizing constant term. First, finding the determinant of a large matrix may either require considerable computer resources, or not be possible, and may well be subject to numerical inaccuracies. Second, repeatedly calculating the logarithm of this determinant, as $\hat{\rho}$ [our α] changes, for each of a number of non-linear least squares iterations itself will be slow, and consumptive of computer resources. And, third, freeing spatial scientists from dealing with the complicated and awkward calculations that accompany spatial statistics will allow (1) the general, less specialized commercial software packages to be employed, and (2) much more effort to be devoted to substantive aspects of research problems.

Our application gives rise to the problem of estimating the $\log \det(\mathbf{I} - \mu\mathbf{M})$ where the eigenvalues of $\mu\mathbf{M}$ are real and in $(-1, 1)$. Without loss of generality, after a suitable rescaling, this is equivalent to estimation of $\log \det(\mathbf{I} - \alpha\mathbf{D})$ where α is in $(-1, 1)$ and $\mathbf{D}$ has real eigenvalues in $[-1, 1]$.

Ideally, we would like to obtain the log $\det(\mathbf{I}-\alpha\mathbf{D})$ for all values of α over a domain such as $(0,1)$. This greatly facilitates working with the log-likelihood function. Clearly, the direct methods require multiple evaluations to provide enough points to approximate the log $\det(\mathbf{I}-\alpha\mathbf{D})$ for all values of α. This further exacerbates the computational difficulties. Fortunately, the method introduced here can estimate log $\det(\mathbf{I}-\alpha\mathbf{D})$ simultaneously for many values of α. This Monte Carlo method provides not only an estimate of log $\det(\mathbf{I}-\alpha\mathbf{D})$, but also confidence bounds for the precision of estimation. The user can employ the algorithm interactively by conducting an initial run, examining the precision, and continuing or discontinuing computations depending upon whether the estimates yield the desired precision. Aggregating the trials from previous runs continually increases the precision of estimation. Naturally, the user can program a formal stopping rule as well. Hence, this method allows users to minimize computational time subject to their desired precision. It easily lends itself to parallel processing.

Relative to existing approximations, this one seems much more easily scaled for very large problems. As mentioned later, Martin (1993) approximation memory requirements explode as n becomes large. Griffith and Sone (1995) require computation of at least one eigenvalue and a small set of determinants for a calibration of their approximation. Hence, their approximation reduces the computational requirements substantially, but does not address the issue of how to compute log-determinants for large irregular matrices prior to the calibration of their method.

Section 2 introduces the Monte Carlo estimate of the log-determinant and the associated confidence intervals, presents the basic algorithm, and discusses issues relevant to its implementation. Section 3 estimates the log-determinant of a $1,000,000 \times 1,000,000$ matrix and demonstrates the salient advantages of the Monte Carlo estimator. Finally, Section 4 concludes with the key results.

2. The approximation

Suppose we are faced with the problem of approximating the log $\det(\mathbf{I}-\alpha\mathbf{D})$ for some $n \times n$ sparse matrix $\mathbf{D}$ that has real eigenvalues in $[-1,1]$, and where $-1 < \alpha < 1$. We first generate p independent random variables:

$$V_i = -n\sum_{k=1}^{m} \frac{\mathbf{x}_i'\mathbf{D}^k\mathbf{x}_i}{\mathbf{x}_i'\mathbf{x}_i}\,\frac{\alpha^k}{k}, \quad i = 1,\ldots,p,$$

where $\mathbf{x}_i \sim N_n(0,I)$, $\mathbf{x}_i$ independent of $\mathbf{x}_j$ if $i \neq j$.

Then we form the interval

$$(\overline{V}-F, \overline{V}+F),$$

where

$$F = \frac{n\alpha^{m+1}}{(m+1)(1-\alpha)} + 1.96\sqrt{\frac{s^2(V_1,\ldots,V_p)}{p}}.$$

The interval $(\overline{V} - F, \overline{V} + F)$ is an asymptotic 95% confidence interval for $\log\det(\mathbf{I} - \alpha\mathbf{D})$. We select the "tuning constants" m and p to give the desired degree of approximation (F).

Proof. By the triangle inequality:

$$|\overline{V} - \log\det(\mathbf{I} - \alpha\mathbf{D})| \leqslant |\overline{V} - \mathscr{E}\overline{V}| + |\mathscr{E}\overline{V} - \log\det(\mathbf{I} - \alpha\mathbf{D})|.$$

The sampling distribution of a mean of independent, finite variance random variables gives us

$$\mathscr{P}\left(|\overline{V} - \mathscr{E}\overline{V}| \leqslant 1.96\sqrt{\frac{s^2(V_1,\ldots,V_p)}{p}}\right) \approx 0.95.$$

The bound for the term $|\mathscr{E}\overline{V} - \log\det(\mathbf{I} - \alpha\mathbf{D})|$ is given in the next theorem. □

Theorem

$$|\mathscr{E}\overline{V} - \log\det(\mathbf{I} - \alpha\mathbf{D})| \leqslant \frac{n\alpha^{m+1}}{(m+1)(1-\alpha)}.$$

Proof. Start with the power series expansion of the matrix function $-\log(\mathbf{I} - \alpha\mathbf{D})$:

$$-\log(\mathbf{I} - \alpha\mathbf{D}) = \sum_{k=1}^{\infty}\frac{\mathbf{D}^k\alpha^k}{k} = \sum_{k=1}^{m}\frac{\mathbf{D}^k\alpha^k}{k} + \sum_{k=m+1}^{\infty}\frac{\mathbf{D}^k\alpha^k}{k}.$$

Now, the trace of $\log(\mathbf{I} - \alpha\mathbf{D})$ is

$$\sum_{i=1}^{n}\log(1 - \alpha\lambda_{D,i}) = \log\det(\mathbf{I} - \alpha\mathbf{D}).$$

The nice property $\mathrm{tr}(cA) + \mathrm{tr}(dB) = \mathrm{tr}(cA + dB)$ has the consequence:

$$-\log\det(\mathbf{I} - \alpha\mathbf{D}) = -\mathrm{tr}(\log(\mathbf{I} - \alpha\mathbf{D})) = \sum_{k=1}^{m}\frac{\mathrm{tr}(\mathbf{D}^k)\alpha^k}{k} + \sum_{k=m+1}^{\infty}\frac{\mathrm{tr}(\mathbf{D}^k)\alpha^k}{k}$$
$$\leqslant \sum_{k=1}^{m}\frac{\mathrm{tr}(\mathbf{D}^k)\alpha^k}{k} + \sum_{k=m+1}^{\infty}\frac{\mathrm{tr}(\mathbf{D}^k)\alpha^k}{m+1}.$$

The expansion of $\log\det(\mathbf{I} - \alpha\mathbf{D})$ in terms of the trace of $\mathbf{D}^k$ is the Martin expansion (Martin, 1993).

The first term on the right-hand side is just the expected value of V_i:

$$\mathscr{E}\overline{V} = \mathscr{E}V_i = \mathscr{E}\left(-n\sum_{k=1}^{m} \frac{\mathbf{x}_i'(\mathbf{D}^k)\mathbf{x}_i}{\mathbf{x}_i'\mathbf{x}_i}\left(\frac{\alpha^k}{k}\right)\right) + \sum_{k=1}^{m} \frac{\operatorname{tr}(\mathbf{D}^k)\alpha^k}{k}.$$

from Result 2 in Appendix A.

Finally, recognizing that $n \geqslant |\operatorname{tr}(\mathbf{D}^k)|$, we get

$$|\mathscr{E}\overline{V} - \log\det(\mathbf{I} - \alpha\mathbf{D})| = \left|\sum_{k=m+1}^{\infty} \frac{\operatorname{tr}(\mathbf{D}^k)\alpha^k}{m+1}\right| \leqslant \frac{n}{m+1}\left|\sum_{k=m+1}^{\infty} \alpha^k\right|$$
$$= \frac{n|\alpha|^{m+1}}{(m+1)(1-\alpha)}. \qquad \square$$

The Algorithm

$v \leftarrow 0$
$x \leftarrow$ random.normal (n)
$c \leftarrow x$
for $k = 1$ to m
$c \leftarrow Dc$
$v \leftarrow n\alpha^k(x'c)/k + v$
end
$v \leftarrow v/(x'x)$

This is repeated p times to obtain the V_i. If the number of non-zero entries in $\mathbf{D}$ is f, then each multiplication by a vector takes time of order f. Each realization of V_i then takes time proportional to fm, and computing the whole suite of estimators will take fmp.

If we have a sequence of increasingly large sparse matrices, with fill f increasing linearly with n, then the required time will be order nmp.

If we wish to keep the margin of error constant, we will need $m \sim \mathrm{O}(\log n)$ and $p \sim \mathrm{O}(n)$. Thus the time required will be of order $n^2 \log n$. Since the log-likelihood itself tends at $\mathrm{O}(n)$, for many applications it is enough that the margin of error (F) be $\mathrm{o}(n)$. It is sufficient that $p \sim \mathrm{O}(1)$ and $m \sim \Omega(\log n)$ to get F/n to shrink to zero. In this case the time requirement of the algorithm only grows at order $n \log n$.

Memory requirement: An advantage of this method is its extremely frugal memory requirement. The algorithm essentially only requires the storage of the sparse matrix $\mathbf{D}$, space for two $n \times 1$ vectors, and a very small amount of additional space for storing the intermediate (scalar) values. Unlike most sparse algorithms, such as the multiplication of sparse matrices, LU decomposition,

etc., this algorithm does not suffer from fill-in: the storage required does not increase as computations are performed. For instance, using the (truncated) Martin expansion directly to estimate the log determinant is problematic for large matrices, since the number of non-zero entries in $\mathbf{D}^k$ can grow explosively with k. The resulting sparse matrices can become too large to work with efficiently, and each successive matrix multiplication $\mathbf{D}\mathbf{D}^k$ becomes increasingly expensive (Griffith and Sone, 1995, p. 171).

Miscellanea: Using the Monte Carlo algorithm has some additional advantages under some circumstances:

- By using a modified algorithm, the log $\det(\mathbf{I} - \alpha\mathbf{D})$ can be computed simultaneously for a set of αs, $(\alpha_1, \ldots, \alpha_q)$. Very little additional time or memory is required. For each realization of V_i the vector $(\mathbf{x}_i'\mathbf{D}\mathbf{x}_i, \ldots, \mathbf{x}_i'\mathbf{D}^k\mathbf{x}_i)$ needs to be computed. Then it can be multiplied by a $k \times q$ matrix with i, jth entry equal to α_i^j/j. The calculation of the vector of quadratic forms takes almost all of the computational effort, thus large numbers of αs can be considered without appreciably increasing the time and memory requirements of the algorithm.
- The Monte Carlo algorithm is a perfect candidate for parallel computation. Different processors could simultaneously compute estimators V_i independently, without any need to communicate at all. The computed V_i would then be combined after all of the computations were finished. No modification of the code would be required.
- The algorithm is easy to encode in a language such as C or Fortran. The algorithm only uses one type of sparse matrix operation, requiring multiplication of a matrix by a vector on the right.

3. A test of the estimator of the log-determinant

To examine the performance of the Monte Carlo determinant estimates as well as their confidence bounds, we conducted a Monte Carlo experiment involving a matrix with known determinants. Specifically, we employed a spatial weight matrix specifying the four nearest neighbors (based on Euclidean distances) to each of 3,107 countries in the US as previously analyzed by Pace and Barry (1998) in the context of examining spatial aspects of voting. Pace and Barry computed the exact log-determinant using various direct techniques which still work well for this size matrix (but these fail to work for the huge matrices discussed later). Each row has four non-zero elements which sum to 1.

In the Monte Carlo experiment, we computed 250 trials. In each trial we used 500 iterations of 50 terms in the expansion for computing the Monte Carlo log-determinants. For each trial we computed the point estimate of the log-determinant and its 95% confidence bounds. Table 1 shows the true log-determinant, the average log-determinant across the 250 trials, the standard deviation of the estimated log-determinant across the 250 trials, and the

Table 1
True and average estimated log-determinants with empirical confidence interval coverage based on 250 trials

α	$\ln\lvert\mathbf{I}-\alpha\mathbf{D}\rvert$ (True)	$\ln\lvert\mathbf{I}-\alpha\mathbf{D}\rvert$ (Estimated)	$\hat{\sigma}_{\ln-\det}$	Empirical coverages
0.0050	−0.0082	−0.0084	0.0087	0.9400
0.0250	−0.2062	−0.2071	0.0437	0.9400
0.0450	−0.6704	−0.6721	0.0788	0.9400
0.0650	−1.4040	−1.4065	0.1141	0.9440
0.0850	−2.4105	−2.4137	0.1495	0.9440
0.1050	−3.6935	−3.6975	0.1852	0.9440
0.1250	−5.2572	−5.2620	0.2210	0.9440
0.1450	−7.1061	−7.1116	0.2570	0.9440
0.1650	−9.2452	−9.2515	0.2933	0.9440
0.1850	−11.6798	−11.6867	0.3299	0.9440
0.2050	−14.4156	−14.4233	0.3667	0.9440
0.2250	−17.4590	−17.4674	0.4038	0.9440
0.2450	−20.8168	−20.8259	0.4412	0.9440
0.2650	−24.4963	−24.5061	0.4790	0.9400
0.2850	−28.5057	−28.5162	0.5172	0.9400
0.3050	−32.8534	−32.8646	0.5557	0.9400
0.3250	−37.5491	−37.5609	0.5947	0.9480
0.3450	−42.6027	−42.6152	0.6341	0.9440
0.3650	−48.0254	−48.0385	0.6740	0.9440
0.3850	−53.8293	−53.8430	0.7144	0.9440
0.4050	−60.0273	−60.0416	0.7554	0.9440
0.4250	−66.6338	−66.6486	0.7970	0.9440
0.4450	−73.6641	−73.6795	0.8392	0.9400
0.4650	−81.1354	−81.1513	0.8822	0.9400
0.4850	−89.0669	−89.0826	0.9258	0.9400
0.5050	−97.4770	−97.4939	0.9703	0.9400
0.5250	−106.3902	−106.4075	1.0157	0.9400
0.5450	−115.8307	−115.8484	1.0619	0.9400
0.5650	−125.8261	−125.8441	1.1092	0.9400
0.5850	−136.4069	−136.4253	1.1576	0.9360
0.6050	−147.6075	−147.6260	1.2073	0.9360
0.6250	−159.4661	−159.4849	1.2582	0.9360
0.6450	−172.0261	−172.0450	1.3106	0.9360
0.6650	−185.3367	−185.3556	1.3645	0.9400
0.6850	−199.4539	−199.4728	1.4203	0.9400
0.7050	−214.4422	−214.4610	1.4780	0.9400
0.7250	−230.3766	−230.3951	1.5379	0.9400
0.7450	−247.3444	−247.3626	1.6002	0.9400
0.7650	−265.4495	−265.4671	1.6654	0.9440
0.7850	−284.8160	−284.8329	1.7338	0.9440
0.8050	−305.5950	−305.6110	1.8058	0.9440
0.8250	−327.9738	−327.9886	1.8823	0.9440
0.8450	−352.1893	−352.2021	1.9640	0.9440
0.8650	−378.5493	−378.5587	2.0519	0.9440
0.8850	−407.4682	−407.4690	2.1478	0.9840

Table 1 (continued)

α	$\ln\lvert\mathbf{I}-\alpha\mathbf{D}\rvert$ (True)	$\ln\lvert\mathbf{I}-\alpha\mathbf{D}\rvert$ (Estimated)	$\hat{\sigma}_{\ln-\det}$	Empirical coverages
0.9050	−439.5286	−439.5041	2.2538	1.0000
0.9250	−475.6031	−475.4931	2.3733	1.0000
0.9450	−517.1235	−516.7018	2.5120	1.0000
0.9650	−566.8242	−565.1713	2.6796	1.0000
0.9850	−631.8478	−624.3757	2.8945	1.0000
0.9950	−678.9991	−659.7821	3.0300	1.0000

empirical coverage of the confidence bounds of the true value of the log-determinant for varying values of α.

As Table 1 reveals, the estimated log-determinants come very close to their true values in repeated sampling. Moreover, these estimates do not show high variability. For example, the estimated log-determinant at α of 0.6050 is −147.626 with a standard error of 1.2073. The log-determinant for an $\alpha = 0.5850$ is −136.4253, a difference of 11.201 or 9.237 standard errors from the log-determinant at $\alpha = 0.6050$. Hence, these estimated log-determinants seem precise enough for most statistical purposes.

The confidence bound estimates also perform well with a minimum estimated coverage of 93.6%. The 95% lower bound of the coverage of a 95% confidence interval would be 93% given 250 trials (the 95% confidence interval for the coverage is simply $\hat{p} \pm 2\sqrt{\hat{p}(1-\hat{p})/250}$, where $\hat{p}$ is the proportion of the time that the Monte Carlo confidence interval for the determinant included the true value of the determinant). The confidence bounds become more conservative at the higher levels of α due to the error bounds employed. Generally speaking, histograms of the estimated log-determinants show a reasonably normal shape and have studentized ranges consistent with a normal distribution.

4. Estimating the log-determinant of a million by million matrix

To illustrate the utility of these techniques, we undertake the challenging task of computing the approximate determinant of a $1,000,000 \times 1,000,000$ matrix. We let $\mathbf{D} = (1/2)(\mathbf{P}_1 + \mathbf{P}_2)$ where $\mathbf{P}_1, \mathbf{P}_2$ represent arbitrary permutation matrices. The first 100,000 rows of both have randomly placed elements while the next 900,000 rows have a band structure. We adopted this structure to easily simulate the case where most of the elements are clustered around the diagonal, but some are scattered randomly in the matrix. While the resultant matrix is quite sparse, it provides quite a challenge to direct methods that depend upon a favorable ordering. The combination of two random permutation matrices creates problems for such algorithms.

We apply the methods introduced earlier to this problem. Specifically, we set p, the number of realizations of the random variable V_i, to 20 and set m, the length of the truncated series, to 20. Using the Matlab interpreted language (Gilbert et al., 1992) on a 133 MHz Pentium computer with 64 MB of memory, it took only 23.1 min to perform the computations. Note, Matlab only allows the use of the double precision data type, which slows down the computations relative to single precision. A compiled language with a single precision floating point data type should perform much faster.

Table 2 presents the estimated determinants with the lower and upper endpoints of the 95% upper confidence intervals as a function of α, the differencing parameter. In repeated trials, the random interval should cover the true value of the determinant in 95% of the trials. However, since the construction of these intervals involves an upper bound, for larger values of α these intervals should act conservatively and have greater coverage than 95%. As one would expect, the magnitudes of the estimated log-determinants on a 1,000,000 $\times$ 1,000,000 matrix become rather large in the negative direction as the matrix becomes ever closer to singularity ($\alpha = 1$).

An outstanding feature of the estimates is the non-overlap between the confidence regions of α and the estimated log-determinant for adjacent values of α over the interval $[0.005, 0.835]$. For example, the lower value of the confidence interval for $\alpha = 0.835$ is -235506.75 while the estimated log-determinant for $\alpha = 0.845$ is -237380.58. Provided the application tolerates variations in α of 0.01, the precision of the log-determinant estimation would exceed the application's requirements over the interval $[0.005, 0.835]$. For α in the range (0.4, 0.6) the fit is very good, and α values in this range are common in applications. For the statistical applications envisioned, such a small lack of precision would rarely pose a problem.

The monumental trade-off of computation time for reduced precision appears quite attractive for many applications. Moreover, this trade-off constitutes a continuous choice under the user's control. If the user wishes to obtain more precision after inspecting an initial run such as present in Table 2, adding more trials should decrease the width of the confidence intervals at the square root of the total number of trials, p. For example, adding an additional 60 trials should double the precision of estimation. If a user requires more precision for larger values of α (e.g., $\alpha > 0.9$), increasing m should help. For the larger values of α, the value of m of 20 used in Table 2 leads to large upper bounds, F, for such values. Fortunately, most empirical applications have α between 0.4 and 0.6, and few applications having $\alpha > 0.9$. Again, inspection of an initial run allows a user to calibrate the experiment to provide just the precision needed.

Finally, as Table 2 demonstrates, the algorithm provides simultaneous estimates of the log-determinant for many choices of α. Hence, the method amortizes the fixed expense of computation over a number of evaluations relative to the usual direct methods. Not only does this greatly save time, but it

Table 2
Log-determinant estimates and confidence bounds versus α

α	Lower confidence bound	Estimate of log-determinant	Upper confidence bound
0.005	−6.37	−4.12	−1.88
0.025	−144.34	−133.10	−121.86
0.045	−462.50	−442.26	−422.02
0.065	−961.42	−932.18	−902.94
0.085	−1642.00	−1603.74	−1565.48
0.105	−2505.48	−2458.19	−2410.89
0.125	−3553.44	−3497.09	−3440.73
0.145	−4787.82	−4722.37	−4656.92
0.165	−6210.91	−6136.33	−6061.76
0.185	−7825.41	−7741.67	−7657.92
0.205	−9634.41	−9541.45	−9448.49
0.225	−11641.44	−11539.20	−11436.97
0.245	−13850.46	−13738.89	−13627.32
0.265	−16265.94	−16144.97	−16024.00
0.285	−18892.87	−18762.41	−18631.96
0.305	−21736.78	−21596.75	−21456.73
0.325	−24803.82	−24654.14	−24504.46
0.345	−28100.82	−27941.39	−27781.95
0.365	−31635.33	−31466.02	−31296.71
0.385	−35415.70	−35236.39	−35057.09
0.415	−41567.70	−41373.16	−41178.61
0.425	−43751.94	−43552.24	−43352.54
0.435	−46005.31	−45800.42	−45595.53
0.445	−48329.36	−48119.24	−47909.11
0.455	−50725.70	−50510.29	−50294.88
0.465	−53196.01	−52975.28	−52754.55
0.475	−55742.09	−55515.99	−55289.88
0.485	−58365.84	−58134.31	−57902.78
0.495	−61069.24	−60832.23	−60595.22
0.505	−63854.42	−63611.87	−63369.31
0.515	−66723.59	−66475.43	−66227.27
0.525	−69679.13	−69425.29	−69171.45
0.535	−72723.53	−72463.93	−72204.32
0.545	−75859.46	−75594.00	−75328.53
0.555	−79089.74	−78818.30	−78546.85
0.565	−82417.37	−82139.81	−81862.24
0.575	−85845.56	−85561.70	−85277.83
0.585	−89377.71	−89087.33	−88796.95
0.605	−96768.83	−96464.43	−96160.04
0.625	−104623.34	−104302.83	−103982.31
0.645	−112979.30	−112638.81	−112298.32
0.665	−121881.74	−121514.10	−121146.46
0.685	−131385.27	−130977.05	−130568.82
0.705	−141557.91	−141083.95	−140610.00
0.725	−152487.51	−151900.91	−151314.31
0.745	−164292.49	−163506.07	−162719.64

Table 2 (continued)

α	Lower confidence bound	Estimate of log-determinant	Upper confidence bound
0.765	−177140.31	−175992.55	−174844.78
0.785	−191280.04	−189472.17	−187664.30
0.805	−207101.48	−204080.30	−201059.12
0.825	−225245.65	−219982.10	−214718.54
0.835	−235506.75	−228479.81	−221452.88
0.845	−246818.71	−237380.58	−227942.45
0.865	−273826.62	−256527.16	−239227.71
0.885	−310115.02	−277735.31	−245355.60
0.905	−363586.40	−301398.27	−239210.14
0.925	−452126.97	−328012.17	−203897.37
0.945	−622777.31	−358206.02	−93634.72
0.965	−1037317.51	−392780.59	251756.34
0.985	−2744759.77	−432758.93	1879241.92
0.995	−9028192.04	−455170.70	8117850.64

allows the precomputation of all the needed log-determinants prior to use at times that do not conflict with other processes. In addition, it avoids the necessity of continually switching from one computer program to another. Effectively, it allows the division of a problem into the stage of computing all the needed log-determinants and the stage of employing these in the computation of some objective of interest.

Appendix A

A series of results on the properties of Rayleigh's quotient (Strang, 1976)

$$\frac{\mathbf{x}'\mathbf{A}\mathbf{x}}{\mathbf{x}'\mathbf{x}}$$

for an $n \times n$ matrix $\mathbf{A}$ and $\mathbf{x} \sim N(0, I)$.

Result 1. *For any $n \times n$ (real) symmetric matrix $\mathbf{A}$ with eigenvalues $\lambda_{A,1}, \ldots, \lambda_{A,n}$ (these are real because $\mathbf{A}$ is symmetric),*

$$\frac{\mathbf{x}'\mathbf{A}\mathbf{x}}{\mathbf{x}'\mathbf{x}} = W_1\lambda_{A,1} + \cdots + W_n\lambda_{A,n},$$

where $W_1, \ldots, W_n$ are coeffieients with the multivariate Dirichlet$(1/2)$ distribution.

Proof. Schur's lemma applied to a real symmetric matrix tells us that we can find an orthonormal coordinate system in which $\mathbf{U}\mathbf{A}\mathbf{U}'$ is diagonal with the eigenvalues of $\mathbf{A}$ on the diagonal (Horn and Johnson, 1985, p. 82). Because the multivariate distribution $N(0,I)$ is rotationally symmetric, after rotation to this orthonormal coordinate system the distribution is still $N(0, I)$. Thus

$$\mathbf{x}'\mathbf{A}\mathbf{x} = z_1^2\lambda_{A,i} + \cdots + z_n^2\lambda_{A,i},$$

where $(z_1, \ldots, z_n)$ are $N(0, I)$. Then

$$\frac{\mathbf{x}'\mathbf{A}\mathbf{x}}{\mathbf{x}'\mathbf{x}} = \frac{z_1^2\lambda_{A,1} + \cdots + z_n^2\lambda_{A,n}}{z_1^2 + \cdots + z_n^2},$$

where $(z_1, \ldots, z_n)$ is $N(0, I)$. This fraction can be rewritten

$$\frac{z_1^2}{\sum_{i=1}^n z_i^2}\lambda_{A,1} + \frac{z_2^2}{\sum_{i=1}^n z_i^2}\lambda_{A,2} + \cdots + \frac{z_n^2}{\sum_{i=1}^n z_i^2}\lambda_{A,n}.$$

As each z_i^2 is chi-square with one degree of freedom, and z_i^2 is independent of z_j^2 for $i \neq j$, the coefficients $W_1, \ldots, W_n$ have the Dirichlet$(1/2)$ distribution (Johnson and Kotz, 1972, p. 231).

The mean, variance and covariance of n-variate Dirichlet$(1/2)$ random variables are:

$$\mathscr{E}W_i = \frac{1}{n}, \quad \mathrm{Var}(W_i) = \frac{(1/2)(n/2 - 1/2)}{(n/2)^2(n/2 + 1)} = \frac{2(n-1)}{n^2(n+2)},$$

$$\mathrm{Cov}(W_i, W_j) = \frac{-2}{n^2(n+2)}$$

(Johnson and Kotz, 1972, p. 233). For additional information on the Dirichlet distribution, see Narayanan (1990). □

Result 2. *For any real* $n \times n$ *matrix* $\mathbf{A}$ *and* $\mathbf{x} \sim \mathcal{N}_n(0, I)$

$$\mathscr{E}\frac{\mathbf{x}'\mathbf{A}\mathbf{x}}{\mathbf{x}'\mathbf{x}} = \mathrm{tr}(\mathbf{A})/n.$$

Proof. For $\mathbf{A}$ symmetric we can use Result 1 to get

$$\mathscr{E}\frac{\mathbf{x}'\mathbf{A}\mathbf{x}}{\mathbf{x}'\mathbf{x}} = \mathscr{E}W_1\lambda_{A,1} + \cdots + \mathscr{E}W_n\lambda_{\mathbf{A},n} = (\lambda_{A,1} + \cdots + \lambda_{A,n})/n = \mathrm{tr}(\mathbf{A})/n$$

If $\mathbf{A}$ is not symmetric, then consider $\mathbf{B} = (1/2)(\mathbf{A}^{\mathrm{t}} + \mathbf{A})$. Clearly $\mathbf{B}$ is symmetric and has the same trace as $\mathbf{A}$. However, for any vector $\mathbf{c}, \mathbf{c}'\mathbf{A}\mathbf{c} = \mathbf{c}'\mathbf{B}\mathbf{c}$. Thus $\mathbf{x}'\mathbf{A}\mathbf{x}/\mathbf{x}'\mathbf{x}$ must have the same distribution as $\mathbf{x}'\mathbf{B}\mathbf{x}/\mathbf{x}'\mathbf{x}$ and therefore is an unbiased estimator of $\mathrm{tr}(\mathbf{A})/n$. □

Result 3. *For real symmetric* $\mathbf{A}$,

$$\mathrm{Var}\frac{\mathbf{x}'\mathbf{A}\mathbf{x}}{\mathbf{x}'\mathbf{x}} = \frac{2\mathrm{VAR}(\lambda)}{n+2},$$

where $\mathrm{VAR}(\lambda)$ *is the population variance of the eigenvalues of* A:

$$\mathrm{VAR}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(\lambda_{A,i} - \mathrm{tr}(A)/n)^2.$$

Proof.

$$\mathrm{Var}\frac{\mathbf{x}'\mathbf{A}\mathbf{x}}{\mathbf{x}'\mathbf{x}} = \sum_{i=1}^{n}\mathrm{Var}(W_i)\lambda_{A,i}^2 + \sum\sum_{i\neq j}\mathrm{Cov}(W_i, W_j)\lambda_{A,i}\lambda_{A,j}.$$

Substituting the corresponding variance and covariance formulas gives:

$$\sum_{i=1}^{n}\frac{2n-1}{n^2(n+2)}\lambda_{A,i}^2 - \sum\sum_{i\neq j}\frac{2}{n^2(n+2)}\lambda_{A,i}\lambda_{A,j}$$

$$= \frac{2}{n+2}\left[\frac{\sum_{i=1}^{n}\lambda_{A,i}^2 - (1/n)\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_{A,i}\lambda_{A,j}}{n}\right] = \frac{2V(\lambda_A)}{n+2}. \quad \square$$

We can use this result to find the approximate variance of V_i because, for m large, $\mathrm{Var}(V_i) \sim \mathrm{Var}(n\mathbf{x}'\log(\mathbf{I}-\alpha\mathbf{D})\mathbf{x}/\mathbf{x}'\mathbf{x})$.

Result 4.

$$\mathrm{Var}\left(\frac{n\mathbf{x}'\log(\mathbf{I}-\alpha\mathbf{D})\mathbf{x}}{\mathbf{x}'\mathbf{x}}\right) \leqslant \frac{n^2(\max_i\log(1-\alpha\lambda_{D,i}) - \min_i\log(1-\alpha\lambda_{D,i}))^2}{2(n+2)}.$$

Proof. The matrix $\mathrm{Var}(n\mathbf{x}'\log(\mathbf{I}-\alpha\mathbf{D})\mathbf{x}/\mathbf{x}'\mathbf{x})$ has (real) eigenvalues $\log(1-\alpha\lambda_{D,1}), \ldots, \log(1-\alpha\lambda_{D,n})$. For any set of real numbers $a_1, \ldots, a_n$, $\mathrm{VAR}(a_1, \ldots, a_n) \leqslant (\mathrm{range}(a_1, \ldots, a_n))^2/4$. Then

$$\mathrm{Var}\left(\frac{n\mathbf{x}'\log(\mathbf{I}-\alpha\mathbf{D})\mathbf{x}}{\mathbf{x}'\mathbf{x}}\right) = \frac{2n^2\mathrm{VAR}(\log(1-\alpha\lambda_{D,i}))}{n+2}$$

$$\leqslant \frac{n^2(\max_i\log(1-\alpha\lambda_{D,i}) - \min_i\log(1-\alpha\lambda_{D,i}))^2}{2(n+2)}$$

follows from Result 3. $\square$

This implies a finite variance for the V_i when the eigenvalues of symmetric $\mathbf{D}$ are real and in the range $[-1, 1]$, and α in the range $(-1, 1)$. For the worse case, where $\mathbf{D}$ has eigenvalues at 1 and at -1, the variance of V_i is bounded above by

$$\frac{n^2(\log(1+|\alpha|) - \log(1-|\alpha|))^2}{2(n+2)}.$$

Finally, we must ensure that the matrix $\mathbf{D}$ has eigenvalues in $[-1, 1]$. One way to do so is to use row stochastic matrices, which have spectral radius one (Horn

and Johnson, 1985). Another way is to find an upper bound for the spectral radius and divide the matrix by the upper bound. For binary matrices that are the adjacency matrix of a planar graph the spectral radius cannot exceed $3 + \sqrt{8n - 15}/2$ (Boots and Royal, 1991). The spectral radius of a matrix can also be computed using iterative methods (Horn and Johnson, 1985).

References

Barry, R.P., Pace, R.K., 1997. Kriging with large data sets using sparse matrix techniques. Comm. Statist. Simulation Comput. 26 (2), 619–629.

Boots, B.N., Royal, G.F., 1991. A conjecture on the maximum value of the principal eigenvalue of a planar graph. Geographical Anal. 23, 226–282.

Gilbert, J.R., Moler, C., Schreiber, R., 1992. Sparse matrices in matlab: Design and implementation. SIAM J. Matrix Anal. 13 (1), 333–356.

Griffith, D.A., Sone, A., 1995. Trade-offs associated with normalizing constant computational simplifications for estimating spatial statistical models. J. Statist. Comput. Simulation 51, 165–183.

Horn, R.A., Johnson, C.R., 1985. Matrix Analysis. Cambridge University Press, New York.

Johnson, N.L., Kotz, S., 1972. Distributions in Statistics: Continuous Multivariate Distributions. Wiley, New York.

Martin, R.J., 1993. Approximations to the determinant term in Gaussian maximum likelihood estimation of some spatial models. Comm. Statist. Theory Methods 22 (1), 189–205.

Narayanan, A., 1990. Computational aspects of dirichlet distribution. American Statistical Association 1990 Proceedings of the Statistical Computing Section, pp. 239–243.

Pace, R.K., Barry, R.P., 1998. Quick computations of spatially autoregressive estimators, Geographical Analysis 29 (3), 232–247.

Strang, G., 1976. Linear Algebra and its Applications. Academic Press, New York.