A Fast Algorithm for Computing Multivariate Normal Densities using Matérn-like Precision Matrices with Unit Marginal Variances

Brynjólfur Gauti Guðrúnar Jónsson

# Introduction

Let be a multivariate random vector with marginal distribution functions for . The joint distribution function of can be written as:

where is the Gaussian copula defined by the GMRF precision matrix .

The Gaussian copula is given by:

where is the joint cumulative distribution function of a multivariate normal distribution with mean vector and precision matrix , and is the inverse of the standard normal cumulative distribution function.

It is imperative that the precision matrix governing the GMRF Copula, , has marginal variance equal to 1 so that is it on the same scale as the transformed data, . This can be troublesome because GMRFs are defined in terms of their precision matrices, , which more often than not have marginal variances that are different from 1.

This paper presents an fast and efficient algorithm for creating a Matérn-like precision matrix, Q, with unit marginal variance, and computing the multivariate Gaussian copula density of where .

The matrix, Q, is defined as

where is the precision matrix of a standardized one-dimensional AR(1) process and is the kronecker product.

The method leverages the special structure of and the use of a kronecker sum in the definitionto avoid explicit formation and inversion of the larger matrix, Q, making it particularly suitable for high-dimensional spatial data.

# Methods

## Theory

## One-Dimensional AR(1) Matrix

The core of our approach is based on the eigendecomposition of an AR(1) precision matrix, which forms the building block of our Matérn-like precision structure. For a one-dimensional AR(1) process with parameter , the precision matrix has a tridiagonal structure:

## Matérn-like Precision Matrix

For two-dimensional spatial fields, we construct a Matérn-like precision matrix using Kronecker products:

where is the identity matrix and denotes the Kronecker product The eigenvalues and eigenvectors of can be calculated numerically and used to evaluate the multivariate Gaussian density, letting us skip out on forming entirely. This is due to the following theorem:

## Eigendecomposition of Kronecker Sums

### Theorem

Let have eigenvalues , , and let have eigenvalues , . Then the Kronecker sum has eigenvalues , .

Moreover, if are linearly independent right eigenvectors of corresponding to (), and are linearly independent right eigenvectors of corresponding to (), then are linearly independent right eigenvectors of corresponding to , .

### Discussion

This theorem provides a crucial insight that allows us to efficiently construct the eigendecomposition of the full precision matrix for two-dimensional spatial fields by leveraging the eigendecomposition of the one-dimensional precision matrix , avoiding the computationally intensive process of explicitly forming and inverting the large matrix .

Furthermore, we can compute the log-density of an observation x directly by using just the eigenvalues and eigenvectors og , thereby enabling efficient computation of the log-density of the multivariate normal distribution even for large spatial fields.

## Example: Calculating Eigenvalues and Eigenvectors for Using

To illustrate the application of Theorem 13.16, consider the one-dimensional precision matrix with known eigendecomposition. Suppose has eigenvalues and corresponding eigenvectors . For a two-dimensional spatial field, we construct the Matérn-like precision matrix using Kronecker products as follows:

where is the identity matrix and denotes the Kronecker product.

### Eigenvalues

The eigenvalues of can be determined from the eigenvalues of . If has eigenvalues for , then the eigenvalues of are given by:

### Eigenvectors

Similarly, the eigenvectors of can be constructed from the eigenvectors of . If and are eigenvectors of corresponding to eigenvalues and respectively, then the eigenvectors of are given by:

Here, is the Kronecker product of and .

These relationships allow us to efficiently compute the eigendecomposition of the full precision matrix using the eigendecomposition of the smaller matrix , significantly reducing the computational complexity.

## Efficient Density Calculation Using Eigendecomposition

Given the eigendecomposition of Q1, we can efficiently compute the multivariate normal density with respect to the precision matrix Q. Let and be the eigenvalues and eigenvectors of Q1, respectively. The log-density of a multivariate normal distribution with precision matrix Q is given by:

where is the dimension of , is the determinant of Q, and is the quadratic form.

### Log-Determinant Calculation

The log-determinant of Q can be computed efficiently using the eigenvalues of Q1:

where is the dimension of Q1.

### Quadratic Form Calculation

First, we define the product of the eigenvector and the data vector as:

where and are the eigenvectors of and denotes the Kronecker product.

Next, we define the eigenvalue sum as:

where and are the eigenvalues of . Using these definitions, the quadratic form can be expressed as:

In this way, we calculate the quadratic form without having to form the matrix Q or its full set of eigenvectors or values.

### Scaling the Input x

The input vector has zero mean and unit marginal variance. Instead of scaling the precision matrix to have , we calculate the marginal standard deviations, , implied by using the eigenstructure of , then scale the input vector using those standard deviations. First we compute the marginal standard deviations:

where is the -th element of the Kronecker product .

## Algorithm Implementation

The density calculation is implemented as follows:

* Compute the eigendecomposition of .
* Calculate the marginal standard deviations .
* For each observation :
  1. Standardize by element-wise multiplication with .
  2. Compute the log-determinant and quadratic form using the formulas above.
  3. Combine the terms to get the log-density.

This approach avoids explicit formation and inversion of the full precision matrix Q, allowing for efficient computation even for large spatial fields. This method has been implemented in C++ code that can perform a full set of computations for a spatial grid (i.e.  would be ) in just over one second.