sparseMVN: An R Package for Multivariate Normal Distributions with Sparse Covariance or Precision Matrices

Michael Braun
SMU Cox School of Business
Southern Methodist University

November 4, 2013

Abstract

The **sparseMVN** package provides functions to sample from a multivariate normal distribution, and compute its density, when the covariance or precision matrix is sparse. By exploiting this sparsity, we can handle high-dimensional distributions quickly, with lower memory usage.

1 Introduction

The R package **mvtnorm** (Genz *et al.* 2012) includes functions for sampling from a multivariate normal (MVN) distribution (`rmvnorm`), and for computing the density of an MVN sample (`dmvnorm`). To use these functions, the user must provide the covariance matrix as a standard, base R matrix. This approach can be inefficient when most of the entries in the covariance matrix are zero. Not only will R store these zeros explicitly, but linear algebra functions (e.g., matrix multiplication, `solve`) will treat the zeros like any other number. Additionally, `rmvnorm` computes a factorization (e.g., eigenvalue, singular value, or Cholesky) of this dense matrix every time the function is called.

The **sparseMVN** provides analogous functions to `rmvnorm` and `dmvnorm` that can be used when either the covariance or precision matrix is sparse. Our method uses the classes and methods from the **Matrix** package (Bates and Maechler 2013), which is now one of the "recommended" packages in R. In contrast to the **mvtnorm** functions, the user supplies the Cholesky decomposition of either the covariance or precision (inverse covariance) matrix. By separating the Cholesky decomposition from the sampling or density functions, the user can call these functions repeatedly, while performing the factorization step only once. Having the option to work with the precision matrix instead of the covariance avoids the need to explicitly invert the precision matrix when the latter is more readily available. This option is useful, for example, when the precision of an MVN distribution is determined by the Hessian at the optimum of a target distribution, as in Laplace approximation of posterior densities. More importantly, by using the sparse matrix algorithms in the **Matrix** package, we dramatically reduce the computational expense of sampling from, and computing densities of, MVN variates of very high dimension.

In the next section, we describe how **sparseMVN** works. We then compare the functions in **sparseMVN** to those in **mvtnorm**, in terms of execution time. We do not provide alternatives to any of the other functions in **mvtnorm**, such as those that estimate cumulative probabilities of the MVN, or those that deal with the multivariate t distribution. In addition, we do not address the case for which the covariance matrix is not positive definite.

2 Algorithmic details

Let X be a random variable, with k dimensions, that is distributed MVN with mean μ and covariance Σ. Let L be a matrix root of Σ, such that $\Sigma = LL'$, and L is lower triangular. To generate a sample realization x, we sample k independent standard normal random variates (call that vector z), and let $x = \mu + Lz$. The matrix factor L could be generated via an eigenvalue, singular value, or Cholesky decomposition. The Cholesky factor of a symmetric, positive definite matrix is the unique factor L for which all of the diagonal elements are positive. For the rest of this paper, we will use that definition for L.

The density of the MVN distribution is

$$f(x) = (2\pi)^{-\frac{k}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(x-\mu)' \Sigma^{-1} (x-\mu)\right] \tag{1}$$

The determinant of Σ is the square of the product of the diagonal elements of L. In addition, $\Sigma^{-1} = (LL')^{-1} = L'^{-1}L^{-1}$. If we define $y = L^{-1}(x-\mu)$, we can write the log density of x as

$$\log f(x) = -\frac{k}{2}\log(2\pi) - \log|L| - \frac{1}{2}y'y \tag{2}$$

Since L is triangular, we can solve $Ly = x - \mu$ for y quickly, avoiding the need to invert Σ explicitly.

In some cases, the precision matrix Σ^{-1} is more readily available than Σ. Let $\Sigma^{-1} = \Lambda\Lambda'$ represent the Cholesky decomposition of Σ^{-1}. To sample x, we sample z as before, solve $\Lambda' x = z$, and then add μ. Since $E(z) = 0$ and $E(zz') = I_k$, we have $E(x) = \mu$, and $\mathrm{cov}(xx') = E(\Lambda'^{-1}zz'\Lambda^{-1}) = \Lambda'^{-1}\Lambda^{-1} = (\Lambda\Lambda')^{-1} = \Sigma$. Then, if we let $y = \Lambda'(x-\mu)$, the log density is

$$\log f(x) = -\frac{k}{2}\log(2\pi) + |\Lambda| - \frac{1}{2}y'y \tag{3}$$

Regardless of whether the user starts with Σ or Σ^{-1}, either random sampling or density calculation will involve solving a triangular system. Which matrix to use depends mostly on convenience and the task at hand.

Nothing in this section so far is special for sparse Σ or Σ^{-1}. The efficiency gains in **sparseMVN** come from storing Σ or Σ^{-1} in a compressed format without explicit zeros, and applying linear algebra routines that are optimized for those sparse matrix structures. The **Matrix** package calls sparse linear algebra routines that are implemented in the **CHOLMOD** library (Chen *et al.* 2008; Davis and Hager 1999, 2009); more information about these routines is available there.

3 Using the package

There are two functions in **sparseMVN**: `rmvn.sparse` and `dmvn.sparse`. The former samples from an MVN, and the latter computes the log density. The signatures are

```
rmvn.sparse(N, mu, CH, prec=TRUE)
dmvn.sparse(x, mu, CH, prec=TRUE)
```

`N` is the number of samples to collect; each sample will be returned in a row in a standard base R matrix. Each row in the matrix `x` is a sample for which we want to compute the log density. In both functions, `mu` is the mean (a vector), and `CH` is the Cholesky decomposition either the covariance or precision matrix. The argument `prec` flags from which type of matrix `CH` was decomposed.

`CH` must be an object of the class `dCHMsimpl` or `dCHMsuper`, as returned by the `Cholesky` function in the **Matrix** package. The first argument to `Cholesky` must be a sparse symmetric matrix, stored as an object of class `dsCMatrix`. As far as we know, there is no particular need to deviate from the defaults of the remaining arguments. If `Cholesky` uses a fill-reducing permutation to compute `CH`, the **sparseMVN** functions will handle that directly, with no additional user intervention required.

A demonstration of how to use the package is in the file `demo/sparseMVN.R`, and can be run with default arguments by calling `demo(sparseMVN)`. Much of the time in this script is spent constructing the covariance or precision matrix, and not in the sampling or computation routines.

4 Timing comparison

Next, we present the results of a timing comparison that demonstrates how much faster `rmvn.sparse` and `dmvn.sparse` can be than their **mvtnorm** counterparts. In this example, we consider an MVN distribution for which either the covariance or precision matrix has a "band-arrow" structure. To construct this matrix, we start with Q_1, a $p \times p$ dense, lower triangular matrix, where p is a relatively small

integer. Then, let $Q_2 = Q_1 \otimes I_m$, where m can be large. Note that Q_2 is also lower triangular. Then, we augment Q_2 by appending k additional rows, all of which are filled by random non-zero values. Call this $(pm + k) \times (pm + k)$ lower triangular matrix Q_3. Finally, we compute $Q = Q_3 Q_3'$, which is symmetric, positive definite, and sparse.

We can visualize the sparsity pattern of Q as a $p \times p$ tiling of m-dimensional diagonal matrices, with dense margins on the bottom and the left. Although Q itself is $(pm + k) \times (pm + k)$, there are only $p^2m + 2kpm + k^2$ non-zero elements. For example, if $p = 3$, $m = 1000$, and $k = 20$, Q has 9,120,400, elements, but only 129,400 of them (1.4%) are non-zero. As m increases, the matrix becomes more sparse.

The timing comparison consists of generating values for the covariance matrix, simulating 200 MVN draws, and computing the log densities, all for different values of p and m. We replicate this study 30 times, and report the mean run times for random sampling and log density computation. Then, we do the whole thing again, but treating the input matrix as a precision matrix instead. Results are in Table 1. Computation was done on a 2010-vintage Mac Pro with 12 processing cores, each running at 2.93GHz, and with 32GB of RAM.

input			density eval		200 mvn draws	
matrix	p	m	dense	sparse	dense	sparse
cov	2	25	0.01	0.01	0.01	0.02
cov	2	250	0.73	0.02	0.31	0.04
cov	2	500	1.62	0.03	0.45	0.08
cov	5	25	0.11	0.01	0.01	0.02
cov	5	250	3.63	0.05	1.15	0.12
cov	5	500	19.88	0.13	5.41	0.31
prec	2	25	0.01	0.02	0.06	0.01
prec	2	250	0.84	0.02	0.25	0.03
prec	2	500	2.43	0.05	1.30	0.07
prec	5	25	0.12	0.01	0.02	0.01
prec	5	250	4.18	0.06	2.04	0.12
prec	5	500	21.44	0.13	7.52	0.27

Table 1: Time, in seconds, for evaluating the density of, and generating 200 random samples from, a multivariate normal distribution. The input matrix is either a covariance or precision, and is either dense or sparse. If dense, routines from **mvtnorm** are used. Otherwise, results are from **sparseMVN**. For all examples here, $k = 15$.

For small m, the MVN routines in **sparseMVN**, actually take longer than those in **mvtnorm**. But as either the precision or covariance matrices become more sparse, we see a dramatic speed-up in computation time.

Table 1 was generated from the file `inst/tables.R` in the package source code. Users can generate one replication of the comparison by calling `demo(sparseMVN)`.

References

Bates D, Maechler M (2013). *Matrix: Sparse and Dense Matrix Classes and Methods*. R package version 1.1-0, URL `http://CRAN.R-project.org/package=Matrix`.

Chen Y, Davis TA, Hager WW, Rajamanickam S (2008). "Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate." *ACM Transactions on Mathematical Software*, **35**(3), 1–14.

Davis TA, Hager WW (1999). "Modifying a Sparse Cholesky Factorization." *SIAM Journal on Matrix Analysis and Applications*, **20**(3), 606–627.

Davis TA, Hager WW (2009). "Dynamic Supernodes in Sparse Cholesky Update/Downdate and Triangular Solves." *ACM Transactions on Mathematical Software*, **35**(4), 1–23.

Genz A, Bretz F, Miwa T, Mi X, Leisch F, Scheipl F, Hothorn T (2012). *mvtnorm: Multivariate Normal and t Distributions*. R package version 0.9-9994, URL `http://CRAN.R-project.org/package=mvtnorm`.