Statistics Software Lab Report - 5

Name of the Student: Shatansh Patnaik
Roll No: 20MA20067

IIT Kharagpur
Statistics Software Lab

Multivariate Normal Distribution

If we have a p x 1 random vector $\mathbf{X}$ that is distributed according to a multivariate normal distribution with a population mean vector μ and population variance-covariance matrix Σ, then this random vector $\mathbf{X}$, will have the joint density function as shown in the expression below:

$$\phi(\mathbf{x}) = \left(\frac{1}{2\pi}\right)^{p/2} |\Sigma|^{-1/2} \exp\{-\frac{1}{2}(\mathbf{x}-\mu)'\Sigma^{-1}(\mathbf{x}-\mu)\}$$

$|\Sigma|$ denotes the determinant of the variance-covariance matrix Σ and Σ^{-1} is just the inverse of the variance-covariance matrix Σ. Again, this distribution will take maximum values when the vector $\mathbf{X}$ is equal to the mean vector μ and decrease around that maximum.

If p is equal to 2, then we have a bivariate normal distribution and this will yield a bell-shaped curve in three dimensions.

The shorthand notation, similar to the univariate version above, is

$$\mathbf{X} \sim N(\mu, \Sigma)$$

We use the expression that the vector $\mathbf{X}$ 'is distributed as' multivariate normal with mean vector, μ and variance-covariance matrix Σ Some things to note about the multivariate normal distribution:

1. The following term appearing inside the exponent of the multivariate normal distribution is a quadratic form:
$$(\mathbf{x}-\mu)'\Sigma^{-1}(\mathbf{x}-\mu)$$

2. If the variables are uncorrelated then the variance-covariance matrix will be a diagonal matrix with variances of the individual variables appearing on the main diagonal of the matrix and zeros everywhere else:
$$\Sigma = \begin{pmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_p^2 \end{pmatrix}$$
In this case the multivariate normal density function simplifies to the expression below:
$$\phi(\mathbf{x}) = \prod_{j=1}^{p} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\{-\frac{1}{2\sigma_j^2}(x_j-\mu_j)^2\}$$

Generation of Multivariate Random Samples

The following is the description of algorithm that is used in the generation for multivariate random samples: This algorithm generates random samples from a multivariate normal distribution with given mean vector μ and covariance matrix σ. The steps involved include decomposing the covariance matrix into a lower triangular matrix using Cholesky decomposition, generating independent standard normal random variates, and then transforming them to the desired multivariate normal distribution. Finally, it calculates the sample mean and covariance matrix, and validates the results by computing the norm error for both the mean and the covariance matrix.

Algorithm 1 Generate Random Samples from Multivariate Normal Distribution

procedure GENERATEMULTIVARIATENORMALSAMPLES(μ, Σ, num_samples)
    $p \leftarrow$ length of μ
    $C \leftarrow$ Cholesky decomposition of Σ
    $Z \leftarrow$ matrix of standard normal random variates with dimensions num_samples $\times p$
    $X \leftarrow$ matrix of zeros with dimensions num_samples $\times p$
    for $i \leftarrow 1$ to p **do**
        $X[, i] \leftarrow \mu[i] + \sum_{j=1}^{i} C[i, j] \times Z[, j]$
    end for
    return X
end procedure

The above algorithm can be implemented by the following code in R.

```r
set.seed(67)
generateMVNSample <- function(meanVector, covarianceVector) {
  p <- length(meanVector)
  C <- chol(covarianceVector)
  Z <- matrix(rnorm(p))

  X <- meanVector + C %*% Z
  return(t(X))
}

generateRandomMVNSamples <- function(meanVector, covarianceVector, num){
  result <- numeric(0)
  for (i in 1:num){
    result <- rbind(result, generateMVNSample(meanVector, covarianceVector))
  }
  return(result)
}
```

We shall now use the above algorithm to print solutions for the given exercises:

```r
printSolutionForExercise <- function(meanVector, covarianceVector){
  num <- 5000
  samples <- generateRandomMVNSamples(meanVector, covarianceVector, num)
  sampleMean <- colMeans(samples)
  sampleCovariance <- cov(samples)

  meanNormError <- getNorm(sampleMean - meanVector)
  covarianceNormError <- getNorm(sampleCovariance - covarianceVector)

  cat("Sample Mean:", sampleMean, "\n")
  cat("Sample Covariance Matrix:", sampleCovariance,"\n")
  cat("Mean Norm Error:", meanNormError, "\n")
  cat("Covariance Matrix Norm Error:", covarianceNormError, "\n")
}

# Exercise 1
cat("Exercise 1\n")
meanVector <- matrix(c(1, -1, 2), nrow=3, ncol=1)
```

```r
covarianceVector <- matrix(c(4, 2, 2, 2, 4, 2, 2, 2, 4), nrow=3, ncol=3)
printSolutionForExercise(meanVector, covarianceVector)

# Exercise 2
cat("Exercise 2\n")
meanVector <- matrix(c(1, 1, 1), nrow=3, ncol=1)
covarianceVector <- matrix(c(2, -1, 0, -1, 2, -1, 0, -1, 2), nrow=3, ncol=3)
printSolutionForExercise(meanVector, covarianceVector)

# Exercise 3
cat("Exercise 3\n")
meanVector <- matrix(c(0, 0, 0), nrow=3, ncol=1)
covarianceVector <- matrix(c(1, -2, 0, -2, 5, 0, 0, 0, 2), nrow=3, ncol=3)
printSolutionForExercise(meanVector, covarianceVector)

# Exercise 4
cat("Exercise 4\n")
meanVector <- matrix(c(4, 3, 2, 1), nrow=4, ncol=1)
covarianceVector <- matrix(c(3, 0, 2, 2, 0, 1, 1, 0, 2, 1, 9, -2, 2, 0, -2,
    4), nrow=4, ncol=4)
printSolutionForExercise(meanVector, covarianceVector)

# Exercise 5
cat("Exercise 5\n")
meanVector <- matrix(c(2, 4, -1, 3, 0), nrow=5, ncol=1)
covarianceVector <- matrix(c(4, -1, 0.5, -0.5, 0, -1, 3, 1, -1, 0, 0.5, 1,
    6, 1, -1, -0.5, -1, 1, 4, 0, 0, 0, -1, 0, 2), nrow=5, ncol=5)
printSolutionForExercise(meanVector, covarianceVector)
```