# Some facts about Gaussian linear models

Michael Wilensky

February 2025

## 1 Introduction

The purpose of this document is to expose some basic facts about Gaussian linear models.

## 2 Definitions

Suppose we have some data, denoted with the vector $d$, and a linear model for the data with Gaussian noise,

$$d = Ax + n, \tag{1}$$

where $n$ is a zero-mean Gaussian noise vector, $x$ is a vector holding the parameters of the linear model (the coefficients), and $A$ is a matrix holding the predictors (basis functions). Given $x$, $A$, and the noise covariance, $N$, and some prior information $\mathcal{I}$, we have

$$d|x, A, N, \mathcal{I} \sim \mathcal{N}(Ax, N) \tag{2}$$

i.e. the data would be normally distributed about a mean $Ax$ with covariance $N$.

Generally we are interested in the case where $x$ is unknown, which is related to $d|a, A, N, \mathcal{I}$ via Bayes' theorem:

$$P(x|d, A, N, \mathcal{I}) = \frac{P(d|x, A, N, \mathcal{I})P(x|A, N, \mathcal{I}))}{P(d|A, N, \mathcal{I})}. \tag{3}$$

For the remainder of this document, we will take $A$, $N$, and $\mathcal{I}$ as fixed prior information, and omit conditioning on them in the notation.

Equation 3 implies a need for a prior distribution, $P(x)$. In theory, this prior can be anything, but in practice it encodes a particular state of knowledge and therefore the choice of prior has an effect on the outcome of the inference. We will focus on Gaussian priors, eschewing discussions of appropriateness to other works. That is to say, *a priori*,

$$x \sim \mathcal{N}(\mu, C) \tag{4}$$

# 3 Properties of the Posterior

Inferences about $x$ given $d$ are captured by the probability distribution of $x|d$. First, what is its form? By multiplying $P(d|x)$ and $P(x)$, which are two Gaussian densities, we can see by examining the exponent that the resulting density must be Gaussian. What are its parameters? The terms in the exponent are, for real vectors,[1]

$$-\frac{1}{2}\left((d-Ax)^T N^{-1}(d-Ax)+(x-\mu)^T C^{-1}(x-\mu)\right)= \\ -\frac{1}{2}\left(x^T\left(A^T N^{-1}A+C^{-1}\right)x+a(x)+b\right) \quad (5)$$

where $a(x)$ is a linear function of $x$, and $b$ is a constant. Their exact forms are not important because all we need to do is figure out what the covariance and mean of our new Gaussian are, which can be determined through a variety of methods, some more painful than others. Importantly, we can read off the posterior covariance, $C'$, from Equation 5:

$$C'^{-1} = A^T N^{-1} A + C^{-1}. \quad (6)$$

To get the mean, we take the gradient with respect to $x$ of the left side of Equation 5 to obtain linear equation that defines the location of the extremum (equivalent to the mean). I'm going to skip over formalities, but in general, one should break this down component-wise and verify the following relationship:

$$C'^{-1}\mu' = A^T N^{-1} d + C^{-1}\mu \quad (7)$$

where $\mu'$ is the posterior mean. This gives us everything we need to know analytically about the posterior. If this is the extent of the model, then the problem has a fully analytic solution and this is the end of the journey up to explorations of particular instances of the quantities involved. However, often Gaussian linear models appear in larger hierarchical models as conditional distributions of joint posteriors that are not necessarily easily understood analytically. In this case it is useful to know how to generate samples from Gaussian distributions, as part of e.g. a Gibbs sampler.

# 4 Generating Gaussian samples

In general, if $\mu'$ and $C'$ are already calculated, one may draw a standard normal sample, $\omega$, and form a sample, $s$, via

$$s = \mu' + C'^{1/2}\omega \quad (8)$$

where $C'^{1/2}$ is any matrix, $M$, satisfying $MM^T = C'$. A common choice is the Cholesky decomposition for its numerical properties but theoretically this is not strictly necessary.

[1] which is a space we can always choose to work in when faced with complex vectors – not sure what happens with e.g. quaternions

In typical sampling applications, $C$ and $N$ are easily invertible, but $C'^{-1}$ is not easily invertible to obtain $C$. Furthermore, it is also typical in Gibbs sampling applications for $A$ or $C$ to be a function of other model parameters not included in $x$, meaning an explicit inversion of $C'^{-1}$ may have to happen for every new sample to use Equation 8 naively. This is often slow and numerically unpredictable compared to various alternatives. More often, we take advantage of the fact that $C'^{-1}$ is easily calculated and instead use a fast linear solver[2] to instead solve a linear equation for $s$. For example, one could solve

$$C'^{-1}s = C'^{-1}\mu' + C'^{-1/2}\omega, \tag{9}$$

which can be written

$$\left(A^T N^{-1} A + C^{-1}\right) s = A^T N^{-1} d + C^{-1}\mu + C'^{-1/2}\omega. \tag{10}$$

Now, we can play one more trick with the fluctuation term (the one involving $\omega$), to get an equivalent set of samples. In particular, we can instead solve

$$\left(A^T N^{-1} A + C^{-1}\right) s = A^T N^{-1} d + C^{-1}\mu + A^T N^{-1/2}\omega_0 + C^{-1/2}\omega_1 \tag{11}$$

where $\omega_0$ and $\omega_1$ are standard normal random vectors. This is usually preferable because, for example, recalculating $A^T N^{-1/2}$ for each sample where $A$ changes is generally easier than recalculating $\left(A^T N^{-1} A + C^{-1}\right)^{1/2}$. To see why this works, first see that

$$C'^{-1}\langle s\rangle = C'^{-1}\mu' \tag{12}$$

i.e.

$$\langle s\rangle = \mu'. \tag{13}$$

where this expectation is over *realizations of* $\omega_0$ *and* $\omega_1$. Then see that the fluctuation terms obey

$$\mathrm{Cov}\left(A^T N^{-1/2}\omega_0 + C^{-1/2}\omega_1, A^T N^{-1/2}\omega_0 + C^{-1/2}\omega_1\right) = C'^{-1} \tag{14}$$

implying

$$\mathrm{Cov}\left(C'^{-1}s, C'^{-1}s\right) = C'^{-1}\mathrm{Cov}\left(s,s\right)C'^{-1} = C'^{-1} \tag{15}$$

further implying

$$\mathrm{Cov}\left(s,s\right) = C'. \tag{16}$$

In some instances, further speedups can be made if $C'-1$ is sparse, or can be made to be sparse with preconditioning (rumor is that this is usually done by finding an approximate inverse that is easy to compute or does not need to be recomputed ever).

[2] for example, `NUMPY`'s linear algebra package wraps `LAPACK`, which is a well-understood high-performing `FORTRAN`-based linear algebra package