`gprob` note

Sergey Fedorov
Quantop, Niels Bohr Institute
Copenhagen, Denmark
October 16, 2024

Probabilistic programs define statistical models as compositions of primitive distributions conditioned on observations. Inference of general conditional distributions is a hard task. The computational overhead of general inference methods, such as Markov chain or Hamiltonian Monte-Carlo, can make it difficult for probabilistic programs to compete with problem-specific approaches. For Gaussian random variables, however, this is not necessarily the case — as conditional distributions can be found analytically, the overhead for their computation can be minimal.

This note presents the background theory for `gprob`, a probabilistic programming package that specializes to Gaussian random variables.

1 Distributions defined via linear maps.

Any Gaussian probabilistic program [1] can be expressed as a linear map of a number of independent identically distributed (iid) normal variables e_i with zero mean and unit variance,

$$e_i \sim \mathcal{N}(0,1), \qquad i = 1, \ldots, n. \tag{1}$$

In `gprob`, all random arrays are represented in this way — as affine maps of latent variables. E.g., a random array named x indexed by a multi-dimensional index $ijk\ldots$ is represented as

$$x_{ijk\ldots} = \sum_{\lambda} e_\lambda a_{\lambda,ijk\ldots} + b_{ijk\ldots}, \tag{2}$$

where $\{e_\lambda \sim \mathcal{N}(0,1)\}$, $\lambda = 1, \ldots, n$ is the set of latent univariate normal variables, and a and b are numerical arrays with real or complex coefficients. The mean of $x_{ijk\ldots}$ equals $b_{ijk\ldots}$, and the covariance is

$$C_{ijk\ldots,rst\ldots} = \sum_{\lambda} a_{\lambda,ijk\ldots} a^*_{\lambda,rst\ldots}, \tag{3}$$

where $rst\ldots$ run over the same set of indices as $ijk\ldots$ and * denotes complex conjugation.

New latent variables are allocated during calls to `normal()`. For example, `normal(0, 1)` produces one latent variable, and `normal(size=sz)`, where `sz` is integer, produces `sz` new latent variables. A global inventory of all latent variables is maintained, but their joint distributions are only materialized within individual arrays. This has an effect on the memory footprint. Roughly speaking, if one random variable takes m bytes, then two arrays of n variables each will take $m \times (2 \times n^2)$ bytes, and not $m \times (2 \times n)^2$ bytes.

2 Conditioning.

In a probabilistic language that maintains a joint distribution over all latent variables, such as the one presented in Ref. [1], conditioning is a statement that updates the latent distribution. One can think of the conditioning in this case as a post-selection to the realizations of the latent variables that ensures that all the conditions are fulfilled. In contrast, in `gprob`, conditioning of x on $y = 0$ is an expression that produces a new random variable, whose latent space is the union of the latent spaces of x and y. In this case, one can think of the conditioning as creating a version x that is feed-forward corrected based on the measurement of y.

To show how the conditioning is implemented, we consider the one-dimensional real case. Any random variable can be transformed to this representation by flattening and, if the variable is complex, by separating its real and imaginary parts. Let the conditioned variable be $\mathbf{x}$, the condition be $\mathbf{y} = \mathbf{0}$, and the result of conditioning be $\mathbf{z} \equiv \mathbf{x}|(\mathbf{y} = \mathbf{0})$. All more general cases can be reduced to this: if a non-zero value of $\mathbf{y}$ was observed, this value can be subtracted from $\mathbf{y}$, and if there is more than one observation, they can be stacked together in one $\mathbf{y}$ vector.

The variables $\mathbf{x}$ and $\mathbf{y}$ take values in $\mathbb{R}^{d_x}$ and $\mathbb{R}^{d_y}$, and are represented by the maps of the same vector of latent variables $\mathbf{e} = (e_1, \ldots, e_n)$,

$$\mathbf{x} = \mathbf{A}_x^T \mathbf{e} + \mathbf{b}_x, \qquad \mathbf{y} = \mathbf{A}_y^T \mathbf{e} + \mathbf{b}_y, \tag{4}$$

where $\mathbf{A}_x$ and $\mathbf{A}_y$ are, respectively, $n \times d_x$ and $n \times d_y$ matrices, and $\mathbf{b}_x$ and $\mathbf{b}_y$ are d_x and d_y-size mean vectors. The rank of the condition matrix, $\mathbf{A}_y$, is assumed to be full and equal to $d_y \leq n$, which means that the conditions are neither incompatible nor tautological. If the rank of $\mathbf{A}_y$ is less than d_y, yet the conditions are compatible, some of them can be eliminated from $\mathbf{A}_y$.

Conditioning splits the latent space into two orthogonal subspaces—one in which the allowed values of the latent variables are fully determined to yield the observed value of $\mathbf{y}$, and one in which the values of the latent variables are unconstrained. The conditional mean is found by evaluating $\mathbf{x}$ at the value of the latent vector $\tilde{\mathbf{e}}$ from the first subspace, which satisfies

$$\mathbf{A}_y^T \tilde{\mathbf{e}} + \mathbf{b}_y = 0. \tag{5}$$

This value can be formally found as

$$\tilde{\mathbf{e}} = -\left(\mathbf{A}_y^+\right)^T \mathbf{b}_y, \tag{6}$$

where $\mathbf{A}_y^+$ is the Moore–Penrose pseudoinverse of $\mathbf{A}_y$. (As $\mathbf{A}_y$ is normally not a square matrix, it does not have an inverse). To obtain $\tilde{\mathbf{e}}$ in practice, it is more convenient to directly solve the system of equations in Eq. (5) in the least square sense. The mean of the conditional variable $\mathbf{z}$, $\mathbf{b}_z$, is then given by

$$\mathbf{b}_z = \mathbf{b}_x - \mathbf{A}_x^T \left(\mathbf{A}_y^+\right)^T \mathbf{b}_y. \tag{7}$$

The conditional map, $\mathbf{A}_z$, is found by left-multiplying $\mathbf{A}_x$ by the projector on the latent subspace unaffected by the conditioning, i.e. the subspace orthogonal to the columns of the constraint matrix $\mathbf{A}_y$. The operator that performs the desired projection is $\mathbf{I} - \mathbf{A}_y\mathbf{A}_y^+$, where $\mathbf{I}$ is the identity matrix, and the conditional map is

$$\mathbf{A}_z = \mathbf{A}_x - \mathbf{A}_y\mathbf{A}_y^+\mathbf{A}_x. \tag{8}$$

One can verify that the conditional mean and covariance that follow from Eqs. (7-8) agree with those obtained based on the momentum representation of $\mathbf{x}$ and $\mathbf{y}$ (given in Appendix A). To make this comparison, we can use the facts that the cross-covariance matrix between $\mathbf{x}$ and $\mathbf{y}$ is

$$\mathbf{C}_{xy} = \mathbf{A}_x^T\mathbf{A}_y, \tag{9}$$

that $\mathbf{P} = \mathbf{A}_y\mathbf{A}_y^+$ is an orthogonal projector satisfying $\mathbf{P}^2 = \mathbf{P}$ and $\mathbf{P}^T = \mathbf{P}$, and also that for a full-rank matrix $\mathbf{A}_y$ the block $\mathbf{C}_{yy} = \mathbf{A}_y^T\mathbf{A}_y$ is invertible, with its inverse being

$$\mathbf{C}_{yy}^{-1} = \mathbf{A}_y^+ \left(\mathbf{A}_y^+\right)^T. \tag{10}$$

3 Causal conditioning of time series. Wiener filtering.

Sec. 2 described the conditioning of all elements of one random vector $\mathbf{x}$ on all elements of another vector $\mathbf{y}$. There is also another, more specialized, problem that arises sometimes—the conditioning of individual elements x_i, for $i = 1, .., d_x$, on subsets $\{y_j : j \leq s_i\}$ for some non-decreasing set of integer indices s_i. This problem appears, in particular, in the causal analysis of time series via Kalman and Wiener filtering, where the elements of $\mathbf{x}$ and $\mathbf{y}$ correspond to a range of sequential moments in time. (For an introduction to filtering see, for example, Ref. [2]). The time stamps on $\mathbf{x}$ and $\mathbf{y}$ are non-decreasing, but not necessarily equally separated, and can also repeat. The task can be for each x_i to find its distribution conditioned only on those observations $\{y_j\}$ that came before x_i. We, therefore, call the corresponding problem "causal conditioning".

Causal conditioning can be performed trivially using the result of Sec. 2 by iterating over the elements of $\mathbf{x}$ and each time selecting an appropriate range from $\mathbf{y}$. This would be, roughly speaking, d_x time more computationally expensive than the simple conditioning of all x_i on all y_j. Fortunately, there is an algorithm that allows to causally condition all elements of $\mathbf{x}$ in one go.

To introduce this algorithm, we first note that in the case of regular conditioning both the conditional mean and map (given by Eq. (7) and Eq. (8), respectively), could be expressed using a $d_y \times d_x$ matrix $\mathbf{B}$,

$$\mathbf{B} = \mathbf{A}_y^+\mathbf{A}_x \tag{11}$$

as

$$\mathbf{b}_z^T = \mathbf{b}_x^T - \mathbf{b}_y^T\mathbf{B}, \qquad \mathbf{A}_z = \mathbf{A}_x - \mathbf{A}_y\mathbf{B}. \tag{12}$$

The i-th column of $\mathbf{B}$ consists of the coefficients that give the closest in Euclidean distance approximation of the i-th column of $\mathbf{A}_x$ via a linear combination of the columns of $\mathbf{A}_y$. From this point of view, it should be

clear that for vectors whose components are indexed left to right, $\mathbf{x} = (x_1, \ldots, x_{d_x})$ and $\mathbf{y} = (y_1, \ldots, y_{d_y})$, a causal structure can be enforced in conditioning by constraining the closest-approximation matrix $\mathbf{B}$ to be upper-triangular,

$$\mathbf{B} = \begin{pmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1d_x} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{s_1 1} & \beta_{s_1 2} & \cdots & \beta_{s_1 d_x} \\ 0 & \beta_{(s_1+1)2} & \cdots & \beta_{(s_1+1)d_x} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \beta_{s_2 2} & \cdots & \beta_{s_2 d_x} \\ 0 & 0 & \cdots & \beta_{(s_2+1)d_x} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta_{d_y d_x} \end{pmatrix}. \tag{13}$$

Here, the list of indices s_i is a non-decreasing sequence of integers from the range $[1, d_y]$ that is tailored to the causal structure of the problem. For example, in causal Wiener filtering of a scalar random process $x(t)$ with a scalar noisy measurement $y(t)$, discretized as x_i and y_i, we could put $s_i = i - 1$ to use all observations before the time t_i for the prediction of $x(t_i)$.

What remains is to find a way of obtaining causally-constrained matrices $\mathbf{B}$. Towards this goal, we introduce the QR decomposition of $\mathbf{A}_y$,

$$\mathbf{A}_y = \mathbf{Q}\mathbf{R}, \tag{14}$$

where $\mathbf{Q}$ is a $n \times d_y$ matrix whose columns is an orthonormal set of vectors, and $\mathbf{R}$ is a $d_y \times d_y$ non-degenerate upper triangular matrix. The left-multiplication of $\mathbf{B}$ by $\mathbf{R}$ preserves the triangular structure (meaning that the elements of $\mathbf{B}$ that are zero in Eq. (13) are also zero in the multiplication product), and, therefore, the least-square solution of the system of equations

$$\mathbf{A}_y \mathbf{B} = \mathbf{A}_x, \tag{15}$$

with the desired causal constraint on $\mathbf{B}$ can be expressed as

$$\mathbf{B} = \mathbf{R}^{-1} \mathcal{M} \left[\mathbf{Q}^T \mathbf{A}_x \right], \tag{16}$$

where $\mathcal{M}[\ldots]$ a masking operator that zeros for its argument ... the matrix elements that are zero in Eq. (13). Finally, the causally conditioned mean and map are given by

$$\mathbf{b}_z^T = \mathbf{b}_x^T - \mathbf{b}_y^T \mathbf{R}^{-1} \mathbf{Q} \, \mathcal{M} \left[\mathbf{Q}^\dagger \mathbf{A}_x \right], \qquad \mathbf{A}_z = \mathbf{A}_x - \mathbf{Q} \, \mathcal{M} \left[\mathbf{Q}^\dagger \mathbf{A}_x \right]. \tag{17}$$

A similar analysis can be performed when $\mathbf{B}$ is generalized lower-triangular, which in the Wiener filtering theory corresponds to anti-causal filtering. The main difference in this case is that $\mathbf{A}_y$ needs to be decomposed as $\mathbf{Q}'\mathbf{L}$, where $\mathbf{L}$ is lower-triangular.

Appendix A: Gaussian distributions defined via moments.

The probability density function of a d-dimensional Gaussian random variable with the mean vector $\mathbf{m} = (m_1, \ldots, m_d)$ and covariance matrix $\mathbf{C}$ to take the value $\mathbf{x} \in \mathbb{R}^d$ is given by

$$p(\mathbf{x}) = (2\pi)^{-d/2} \left(\det(\mathbf{C}) \right)^{-1/2} \exp\left(-\frac{1}{2} (\mathbf{x} - \mathbf{m})^T \mathbf{C}^{-1} (\mathbf{x} - \mathbf{m}) \right), \tag{A1}$$

where $\mathbf{C}^{-1}$ is the inverse of the covariance. If $\mathbf{C}$ is degenerate and thus non-invertible, the possible realizations of the random variable do not cover the whole space $\mathbb{R}^d$. Our convention when defining the probability density function for such variables is to assign $p(\mathbf{x}) = 0$ to impossible samples, and, when dealing with possible samples, to use the Moore–Penrose pseudoinverse $\mathbf{C}^+$ instead of $\mathbf{C}^{-1}$ and the determinant of the non-degenerate projection of $\mathbf{C}$ instead of $\det(\mathbf{C})$ in Eq. (A1). This is consistent with the way degenerate distributions are handled in `scipy.stats` [3].

If the random variable consists of two stacked vectors, $\mathbf{x}$ with the dimension d_x, and $\mathbf{y}$ with the dimension d_y, and $\mathbf{C}$ is their joint covariance matrix expressed in the block form as,

$$\mathbf{C} = \begin{pmatrix} \mathbf{C}_{xx} & \mathbf{C}_{xy} \\ \mathbf{C}_{yy} & \mathbf{C}_{yy} \end{pmatrix}, \tag{A2}$$

the conditional probability density of $\mathbf{x}$ given $\mathbf{y} = \tilde{\mathbf{y}}$ is Gaussian with the mean and covariance given by

$$\mathbf{m}_{\mathrm{c}} = \mathbf{m}_x + \mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}(\tilde{\mathbf{y}} - \mathbf{m}_y), \tag{A3}$$

$$\mathbf{C}_{\mathrm{c}} = \mathbf{C}_{xx} - \mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}\mathbf{C}_{yx}. \tag{A4}$$

Appendix B: Parametric derivatives and Fisher information.

A number of information-theoretic properties of distributions whose mean and covariance depend on some parameters θ are expressed via the logarithm of the probability density and its derivatives. For non-degenerate Gaussian distributions, they are given, respectively, by

$$\ln p(\mathbf{x}) = -\frac{1}{2}\Delta\mathbf{x}^T\mathbf{C}^{-1}\Delta\mathbf{x} - \frac{1}{2}\ln\det(\mathbf{C}), \tag{B1}$$

and

$$\frac{\partial \ln p(\mathbf{x})}{\partial\theta_i} = \frac{\partial\mathbf{m}^T}{\partial\theta_i}\mathbf{C}^{-1}\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^T\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta_i}\mathbf{C}^{-1}\Delta\mathbf{x} - \frac{1}{2}\mathrm{Tr}\left[\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta_i}\right], \tag{B2}$$

where

$$\Delta\mathbf{x} = \mathbf{x} - \mathbf{m}. \tag{B3}$$

The expression for the derivatives was obtained using $\partial\mathbf{C}^{-1}/\partial\theta = -\mathbf{C}^{-1}(\partial\mathbf{C}/\partial\theta)\mathbf{C}^{-1}$, and the fact that the matrix determinant equals the product of the eigenvalues. The components of the Fisher information matrix F_{ij} are given by (e.g. [4])

$$F_{ij} = \frac{\partial\mathbf{m}^T}{\partial\theta_i}\mathbf{C}^{-1}\frac{\partial\mathbf{m}}{\partial\theta_j} + \frac{1}{2}\mathrm{Tr}\left[\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta_i}\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta_j}\right], \tag{B4}$$

and related to the log probability density as

$$F_{ij} = \left\langle \frac{\partial \ln p(\mathbf{x})}{\partial\theta_i}\frac{\partial \ln p(\mathbf{x})}{\partial\theta_j} \right\rangle = -\left\langle \frac{\partial^2 \ln p(\mathbf{x})}{\partial\theta_i\partial\theta_j} \right\rangle. \tag{B5}$$

For completeness, here is also the expression for the second derivatives of the log probability density,

$$\begin{aligned}\frac{\partial^2 \ln p(\mathbf{x})}{\partial\theta_i\partial\theta_j} =& -F_{ij} + \frac{\partial^2\mathbf{m}^T}{\partial\theta_i\partial\theta_j}\mathbf{C}^{-1}\Delta\mathbf{x} \\ &+ \mathrm{Tr}\left[\left(\frac{1}{2}\mathbf{C}^{-1}\frac{\partial^2\mathbf{C}}{\partial\theta_i\partial\theta_j}\mathbf{C}^{-1} + \mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta_i}\frac{\partial\mathbf{C}^{-1}}{\partial\theta_j}\right)\left(\Delta\mathbf{x}\Delta\mathbf{x}^T - \mathbf{C}\right)\right].\end{aligned} \tag{B6}$$

References

[1] D. Stein and S. Staton, "Compositional Semantics for Probabilistic Programs with Exact Conditioning," in *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, Jun. 2021, pp. 1–13. [Online]. Available: https://ieeexplore.ieee.org/document/9470552

[2] R. G. Brown, *Introduction to Random Signal Analysis and Kalman Filtering.* Wiley, 1983.

[3] Scipy reference `scipy.stats.multivariate_normal`.

[4] Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi, "Theoretical foundation for CMA-ES from information geometric perspective," Jun. 2012. [Online]. Available: http://arxiv.org/abs/1206.0730