Supplementary Material

Integrals over Gaussians under Linear Domain Constraints

A ALGORITHMS

Algorithm 2 Elliptical slice sampling for a linearly constrained standard normal distribution

procedure LinESS($\mathbf{A}, \mathbf{b}, N, \mathbf{x}_0$)
  ensure all($\mathbf{a}_m^\intercal \mathbf{x}_0 + b_m > 0\ \forall m$) // initial vector needs to be in domain
  $\mathbf{X} = [\ ]$ // initialize sample array
  for n = 1,..., N **do**
    $\boldsymbol{\nu} \sim \mathcal{N}(0, \mathbf{1})$
    $\mathbf{x}(\theta) = \mathbf{x}_0 \cos\theta + \boldsymbol{\nu} \sin\theta$ // construct ellipse
    $\boldsymbol{\theta} \leftarrow \textbf{sort}(\{\theta_{j,1/2}\}_{j=1}^{M})$ s.t. $\mathbf{a}_j^\intercal(\mathbf{x}_0 \cos\theta_{j,1/2} + \boldsymbol{\nu} \sin\theta_{j,1/2}) = 0$ // $2M$ intersections, Eq. (2)
    $\boldsymbol{\theta}_{\text{act}} \leftarrow \{[\theta_l^{\min}, \theta_l^{\max}]\}_{l=1}^{L}$ s.t. $\ell(x(\theta_l^{\min/\max} + d\theta)) - \ell(x(\theta_l^{\min/\max} - d\theta)) = \pm 1$ // Set brackets
    $u \sim [0, 1] \cdot \sum_l^L (\theta_l^{\max} - \theta_l^{\min})$
    $\theta_u \leftarrow$ transform u to angle in bracket
    $\mathbf{X}[n] \leftarrow \mathbf{x}(\theta_u)$ // update sample array
    $\mathbf{x}_0 \leftarrow \mathbf{x}(\theta_u)$ // set new initial vector
  end for
  return $\mathbf{X}$
end procedure

Algorithm 3 Subset simulation for linear constraints

procedure SubsetSim($\mathbf{A}, \mathbf{b}, N, \rho = \frac{1}{2}$)
  $\mathbf{X} \sim \mathcal{N}(0, \mathbf{1})$ // N initial samples
  $\gamma, \hat{\rho} =$ FindShift(ρ, $\mathbf{X}$, $\mathbf{A}, \mathbf{b}$) // find new shift value
  $\log Z = \log \hat{\rho}$ // record the integral
  while $\gamma > 0$ **do**
    $\mathbf{X} \leftarrow$ LinESS($\mathbf{A}, \mathbf{b} + \gamma, N, \mathbf{x}_0$) // draw new samples from new constrained domain
    $\gamma, \hat{\rho} \leftarrow$ FindShift(ρ, $\mathbf{X}$, $\mathbf{A}, \mathbf{b}$) // find new shift value
    $\log Z \leftarrow \log Z + \log \hat{\rho}$ // Update integral with new conditional probability
  end while
  return $\log Z$, shift sequence
end procedure

function FindShift(ρ, $\mathbf{X}$, $\mathbf{A}, \mathbf{b}$) // find shift s.t. a fraction ρ of $\mathbf{X}$ fall into the resulting domain.
  $\boldsymbol{\gamma} \leftarrow \textsc{sort}(-\min_m(\mathbf{a}_m^\intercal \mathbf{x}_n + b_m)_{n=1}^{N})$ // sort shifts in ascending order
  $\gamma \leftarrow (\boldsymbol{\gamma}[\lfloor \rho N \rfloor] + \boldsymbol{\gamma}[\lfloor \rho N \rfloor + 1])/2$ // Find shift s.t. ρN samples lie in the domain
  $\hat{\rho} \leftarrow (\#\mathbf{X}$ inside$)/N$ // true fraction could deviate from ρ
  return γ, $\hat{\rho}$
end function

B DETAILS ON EXPERIMENTS

B.1 Synthetic experiments

1000-d integrals We further consider three similar synthetic integrals over orthants of 1000-d correlated Gaussians with a fixed mean and a randomly drawn covariance matrix. Table 1 shows the mean and std. dev. of the binary logarithm of the integral estimator averaged over five runs of HDR using 2^8 samples per nesting for integration, as well as the average CPU time[1].

Table 1: Integrals of Gaussian orthants in 1000-d

#	$\langle \log_2 \hat{Z} \rangle$	std. dev.	$t_{\text{CPU}}[10^3\text{s}]$
1	-162.35	4.27	8.86
2	-160.54	2.09	7.40
3	-157.62	3.19	7.64

B.2 Bayesian optimization

Probability of minimum After having chosen N_R representer points, the approximate probability for $\mathbf{x}_i, i = 1, \dots, N_R$ to be the minimum, Eq. (6) can be rephrased in terms of Eq. (1) by writing the $N_R - 1$ linear constraints in matrix form. This $(N_R - 1) \times N_R$ matrix is a $(N_R - 1) \times (N_R - 1)$ identity matrix with a vector of $-\mathbf{1}$ added in the ith column,

$$\mathbf{M} = \begin{bmatrix} \mathbf{1}_{(i-1)\times(i-1)} & -\mathbf{1}_{i-1} & \mathbf{0}_{(i-1)\times(N_R-i)} \\ \mathbf{0}_{(N_R-i)\times(i-1)} & -\mathbf{1}_{N_R-i} & \mathbf{1}_{(N_R-i)\times(N_R-i)} \end{bmatrix}.$$

Then the objective Eq. (6) can be written as

$$\hat{p}_{\min}(\mathbf{x}_i) = \int \mathcal{N}(\mathbf{f}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \prod_{j\neq i}^{N_R} \Theta([\mathbf{Mf}]_j)\, d\mathbf{f}$$
$$= \int \mathcal{N}(\mathbf{u}, \mathbf{0}, \mathbf{1}) \prod_{j\neq i}^{N_R} \Theta\left(\left[\mathbf{M}\left(\boldsymbol{\Sigma}^{1/2}\mathbf{u} + \boldsymbol{\mu}\right)\right]_j\right)\, d\mathbf{u}$$

where we have done the substitution $\mathbf{u} = \boldsymbol{\Sigma}^{-1/2}(\mathbf{f} - \boldsymbol{\mu})$, and hence $\mathbf{f} = \boldsymbol{\Sigma}^{1/2}\mathbf{u} + \boldsymbol{\mu}$. Writing the constraints in matrix form as in Section 2, $\mathbf{A}^\intercal = \mathbf{M}\boldsymbol{\Sigma}^{1/2}$ and $\mathbf{b} = \mathbf{M}\boldsymbol{\mu}$.

Derivatives In order to compute a first-order approximation to the objective function in entropy search, we need the derivatives of $\hat{p}_{\min}$ w.r.t. the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. The algorithm requires the following derivative, where $\lambda = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$,

$$\frac{\mathrm{d}}{\mathrm{d}\,\lambda} \log p_{\min} \approx \frac{1}{\hat{p}_{\min}} \frac{\mathrm{d}\,\hat{p}_{\min}}{\mathrm{d}\,\lambda}$$
$$= \frac{1}{\hat{p}_{\min}} \int d\mathbf{f}\ \frac{\mathrm{d}\,\mathcal{N}(\mathbf{f}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\mathrm{d}\,\lambda} \prod_{j\neq i}^{N_R} \Theta([\mathbf{Mf}]_j)$$
$$= \frac{1}{\hat{p}_{\min}} \mathbb{E}\left[\frac{\mathrm{d}\log\mathcal{N}(\mathbf{f}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\mathrm{d}\,\lambda}\right],$$

using $\frac{\mathrm{d}\,\mathcal{N}(\mathbf{f},\boldsymbol{\mu},\boldsymbol{\Sigma})}{\mathrm{d}\,\lambda} = \mathcal{N}(\mathbf{f}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \frac{\mathrm{d}\log\mathcal{N}(\mathbf{f},\boldsymbol{\mu},\boldsymbol{\Sigma})}{\mathrm{d}\,\lambda}$. Hence all we need is to compute the derivatives of the log normal distribution w.r.t. its parameters, and the expected values thereof w.r.t. the integrand. The required derivatives are

$$\frac{\mathrm{d}\log\mathcal{N}(\mathbf{f}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\mathrm{d}\,\mu_i} = \left[\boldsymbol{\Sigma}^{-1}(\mathbf{f} - \boldsymbol{\mu})\right]_i,$$

$$\frac{\mathrm{d}\log\mathcal{N}(\mathbf{f}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\mathrm{d}\,\boldsymbol{\Sigma}_{ij}} = \frac{1}{2}\left[\boldsymbol{\Sigma}^{-1}(\mathbf{f} - \boldsymbol{\mu})(\mathbf{f} - \boldsymbol{\mu})^\intercal\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\right]_{ij}$$

and the second derivative

$$\frac{\mathrm{d}^2\,\mathcal{N}(\mathbf{f}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\mathrm{d}\,\mu_i\,\mathrm{d}\,\mu_j}$$
$$= \mathcal{N}(\mathbf{f}, \boldsymbol{\mu}, \boldsymbol{\Sigma})\left(\left[\boldsymbol{\Sigma}^{-1}(\mathbf{f} - \boldsymbol{\mu})(\mathbf{f} - \boldsymbol{\mu})^\intercal\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\right]_{ij}\right)$$

Hence we only need $\mathbb{E}_{p_{\min}}[(\mathbf{f} - \boldsymbol{\mu})]$ and $\mathbb{E}_{p_{\min}}[(\mathbf{f} - \boldsymbol{\mu})(\mathbf{f} - \boldsymbol{\mu})^\intercal]$ to compute the following gradients,

$$\frac{\mathrm{d}\log p_{\min}}{\mathrm{d}\,\mu_i} \approx \frac{1}{\hat{p}_{\min}}\mathbb{E}_{\hat{p}_{\min}}\left[\left[\boldsymbol{\Sigma}^{-1}(\mathbf{f} - \boldsymbol{\mu})\right]_i\right],$$

$$\frac{\mathrm{d}\log p_{\min}}{\mathrm{d}\,\boldsymbol{\Sigma}_{ij}} \approx$$
$$\frac{1}{\hat{p}_{\min}}\mathbb{E}_{\hat{p}_{\min}}\left[\frac{1}{2}\left[\boldsymbol{\Sigma}^{-1}(\mathbf{f} - \boldsymbol{\mu})(\mathbf{f} - \boldsymbol{\mu})^\intercal\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\right]_{ij}\right],$$

and the Hessian w.r.t. $\boldsymbol{\mu}$,

$$\frac{\mathrm{d}^2\log p_{\min}}{\mathrm{d}\,\mu_i\,\mathrm{d}\,\mu_j} = 2\frac{\mathrm{d}\log\hat{p}_{\min}}{\mathrm{d}\,\boldsymbol{\Sigma}_{ij}} - \frac{\mathrm{d}\log p_{\min}}{\mathrm{d}\,\mu_i}\frac{\mathrm{d}\log p_{\min}}{\mathrm{d}\,\mu_j}.$$

[1] On 6 CPUs, the wall clock time was ∼20 min per run.