MCQMC 2024

An Introduction to Preconditioning

Max Hird (UCL)

Joint work with Sam Livingstone (UCL)

https://arxiv.org/abs/2312.04898

Part I: Conditioning

Conditioning

20th C Maths starts being concerned with *computability* and not simply *conceivability*:

$$\begin{array}{ll} e_1 & 1{\cdot}4x + 0{\cdot}9y = 2{\cdot}7 \\ e_2 & -0{\cdot}8x + 1{\cdot}7y = -1{\cdot}2 \end{array} \Bigg\} \iff \begin{array}{ll} 0.01 \times e_1 + e_2 & -0{\cdot}786x + 1{\cdot}709y = -1{\cdot}173 \\ e_2 & -0{\cdot}800x + 1{\cdot}700y = -1{\cdot}200 \end{array} \Bigg\}$$

well-conditioned ill-conditioned

Turing coins the *condition number* and defines it in multiple ways:

- N-condition number: $\|A\|_F \|A^{-1}\|_F$ where $\|A\|_F := \sqrt{\mathrm{Tr}(A^*A)}$
- M-condition number: $M(A)M(A^{-1})$ where $M(A) := \max_{ij} |m_{ij}|$

The condition number ≥ 1, and 1 is the best possible value

Turing [1948]

Conditioning

Nowadays the problem of matrix inversion has the condition number $\kappa(Ax = b) = \|A\|_{\text{op}}\|A^{-1}\|_{\text{op}}$

It is the worst error in the output given a noisy input: say we observe $b + \delta b$ instead of b

Relative input error: $\dfrac{\|b + \delta b - b\|}{\|b\|} = \dfrac{\|\delta b\|}{\|b\|}$

Relative output error: $\dfrac{\|A^{-1}b - A^{-1}(b + \delta b)\|}{\|A^{-1}b\|} = \dfrac{\|A^{-1}\delta b\|}{\|A^{-1}b\|}$

Worst relative output error relative to the relative input error:

$$\kappa(Ax = b) := \sup_{b, \delta b \neq 0} \left\{ \frac{\|A^{-1}\delta b\|}{\|A^{-1}b\|} \Big/ \frac{\|\delta b\|}{\|b\|} \right\}$$

Conditioning

$$\kappa(Ax=b) := \sup_{b,\delta b \neq 0} \left\{ \frac{\|A^{-1}\delta b\|}{\|A^{-1}b\|} \Big/ \frac{\|\delta b\|}{\|b\|} \right\}$$

$$= \sup_{b,\delta b \neq 0} \left\{ \frac{\|A^{-1}\delta b\|}{\|\delta b\|} \frac{\|b\|}{\|A^{-1}b\|} \right\}$$

$$= \sup_{b \neq 0} \left\{ \frac{\|b\|}{\|A^{-1}b\|} \right\} \sup_{\delta b \neq 0} \left\{ \frac{\|A^{-1}\delta b\|}{\|\delta b\|} \right\}$$

$$= sup_{c \neq 0} \left\{ \frac{\|Ac\|}{\|c\|} \right\} \|A^{-1}\|_{\mathsf{op}}$$

$$= \|A\|_{\mathsf{op}} \|A^{-1}\|_{\mathsf{op}}$$

Conditioning

$\|A\|\|A^{-1}\|$ is also important in many other scenarios:

- Matrix Multiplication
- Explicit Matrix Inversion: $\dfrac{\|A^{-1} - (A+E)^{-1}\|}{\|A^{-1}\|} / \dfrac{\|E\|}{\|A\|} \leq \|A\|\|A^{-1}\|$
- Iterative Inversion Methods: (from [Qu et al. 2022, https://arxiv.org/abs/2209.00809])

	Jacobi	Gauss-Seidel	Steepest Descent	Conjugate Gradient
linear convergence rates	$\frac{\kappa(A)-1}{\kappa(A)+1}$	$\frac{\kappa(A)-1}{\kappa(A)+1}$	$\left(\frac{\kappa(A)-1}{\kappa(A)+1}\right)^2$	$\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}$

Table 1 **Rates of linear convergence of some iterative methods for solving the system** $Ax = b$

NB $\|A\|\|A^{-1}\|$ is useful to know, but it is not the *only* way to encode difficulty

Recall Turing's initial definitions

$$\text{dist}(A, \text{Singular Matrices}) = \|A^{-1}\|^{-1} \qquad \text{[Kahan 1966]}$$

In many cases the condition number is as hard to calculate as the original problem

From Problems to Algorithms

Recall the initial motivations for the concept of conditioning

The problems $\{Ax = b, \lambda_1(A), \lambda_d(A), \dots\}$ all admit `time based' solvers/algorithms

In these contexts $\|A\| \|A^{-1}\|$ has a different meaning:

e.g. ∇-descent on $\frac{1}{2}w^T A w - b^T w$ with $A > 0$ (solution @ $w^* = A^{-1}b$)

Algorithm: $w^{k+1} = w^k - \alpha(Aw^k - b)$

Decompose along the eigenvectors of A: $x^k := Q^T(w^k - w^*)$ giving

$$x_i^{k+1} = (1 - \alpha\lambda_i)x_i^k = (1 - \alpha\lambda_i)^{k+1} x_i^0$$

From Problems to Algorithms

∇-descent on $\frac{1}{2}w^T A w - b^T w$ with $A > 0$ (solution @ $w^* = A^{-1}b$)

$$x_i^{k+1} = (1 - \alpha\lambda_i)x_i^k = (1 - \alpha\lambda_i)^{k+1}x_i^0 \qquad \left(x^k := Q^T(w^k - w^*)\right)$$

Rates of convergence are dominated by those along extremal eigenvectors

So is the choice of α

$$\left(\lambda_d = \|A^{-1}\|^{-1}\right)$$

Optimal $\alpha = \dfrac{2}{\lambda_1 + \lambda_d} = \dfrac{1}{\lambda_d}\dfrac{2}{\frac{\lambda_1}{\lambda_d} + 1} = \dfrac{2\|A^{-1}\|}{\|A\|\|A^{-1}\| + 1}$

Optimal rate $= \dfrac{\frac{\lambda_1}{\lambda_d} - 1}{\frac{\lambda_1}{\lambda_d} + 1} = \dfrac{\|A\|\|A^{-1}\| - 1}{\|A\|\|A^{-1}\| - 1}$

So both *stability and rate of convergence* are governed by $\kappa(Ax = b) = \|A\|\|A^{-1}\|$

Not only does κ describe the generic difficulty of computing a solution, it dictates performance of particular algorithms.

Condition number in Sampling

Target in the form $\pi \propto \exp(-U(x))$ on $\mathbb{R}^d$ such that $m\mathbf{I}_d \leq \nabla_x^2 U(x) \leq M\mathbf{I}_d$ for all $x \in \mathbb{R}^d$:
$U : \mathbb{R}^d \to \mathbb{R}$ is *m-strongly convex* and *M-smooth*

m-strong convexity:

- Unimodal
- m measures the curvature of $U(x)$
- e.g. posterior with concave log-likelihood, Gaussian prior

M-smoothness:

- $\nabla_x U(x)$ is M-Lipschitz
- Discretisations work nicely
- Minimum average acceptance (α_0) controlled [Andrieu et al 2022]

The condition number associated with *sampling from* π is

$$\kappa := \sup_{x \in \mathbb{R}^d} \|\nabla_x^2 U(x)\|_2 \sup_{x \in \mathbb{R}^d} \|\nabla_x^2 U(x)^{-1}\|_2$$

If $m\mathbf{I}_d \leq \nabla_x^2 U(x) \leq M\mathbf{I}_d$ is tight $\kappa = M/m$

As $\kappa \to 1$, the eigenvalues of $\nabla_x^2 U(x)$ get squeezed together, and π starts to look more like an isotropic Gaussian

Importance of the sampling condition number

	Spectral Gap	ϵ-Mixing Time
Upper bounds	$\exists$	$\forall$
	$O(\frac{\sqrt{\log d}}{\kappa\sqrt{d}})$ (on a Gaussian) Lee et al. [2021]	$O(d\kappa^2 \log\frac{1}{\epsilon})$ Dwivedi et al. [2019]
	$O(\frac{\log d}{\kappa d})$ Lee et al. [2021]	$O(d\kappa \log\frac{1}{\epsilon})$ Dwivedi et al. [2019]
	$O(\frac{\sqrt{\log d}}{\kappa\sqrt{d}})$ (on a Gaussian) Lee et al. [2021]	$O(d^{\frac{2}{3}}\kappa \log\frac{1}{\epsilon})$ Chen et al. [2019]
Lower bounds	$\forall$	$\exists$
	$O(\frac{1}{\kappa d})$ Andrieu et al. [2022]	$O(\frac{\kappa d}{\log^2 d})$ $(\epsilon = e^{-1})$ Lee et al. [2021]

Key: ● - RWM ● - MALA ● - HMC

All bounds up to logarithmic factors, mixing times in TV

Part II: Preconditioning

Preconditioning

Preconditioning is a transformation from the original problem to a new one

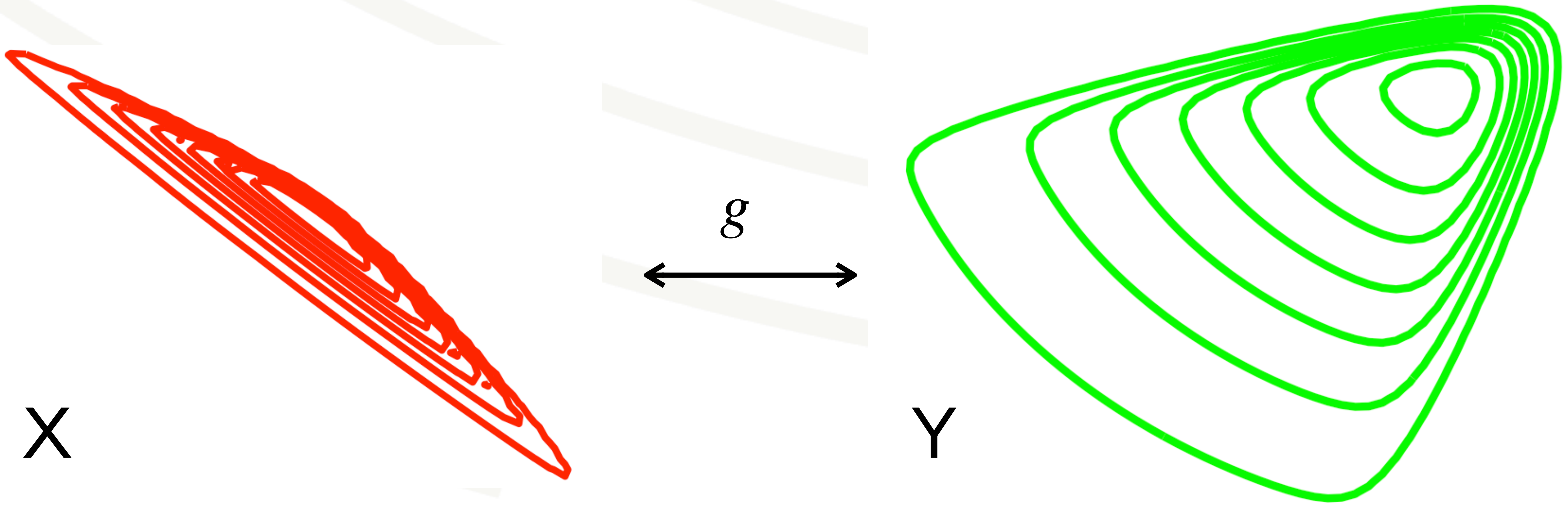
We do it to reduce the condition number:

e.g. starting with $Ax = b$ make the transformation $y = Mx, c = N^{-1}b$ to get to the problem $NAMy = c$ with condition number $\|NAM\|\|M^{-1}A^{-1}N^{-1}\|$

When $Y = g(X) = LX$ for the condition number of sampling from the distribution of Y is

$$\kappa_L := \sup_{y \in \mathbf{R}^d} \|\nabla_y^2 \tilde{U}(y)\|_2 \sup_{y \in \mathbf{R}^d} \|\nabla_y^2 \tilde{U}(y)^{-1}\|_2 = \sup_{x \in \mathbf{R}^d} \|L^{-T} \nabla_x^2 U(x) L^{-1}\|_2 \sup_{x \in \mathbf{R}^d} \|L \nabla_x^2 U(x)^{-1} L^T\|_2$$

Used in all major MCMC software packages (Stan, Tensorflow, Pyro etc.) even though theory is lacking.



Linear Preconditioning for Sampling

Intuition: set L to be the square root of some *representative* of $\nabla_x^2 U(x)$ and hope that $\kappa_L \ll \kappa$, doesn't always work:

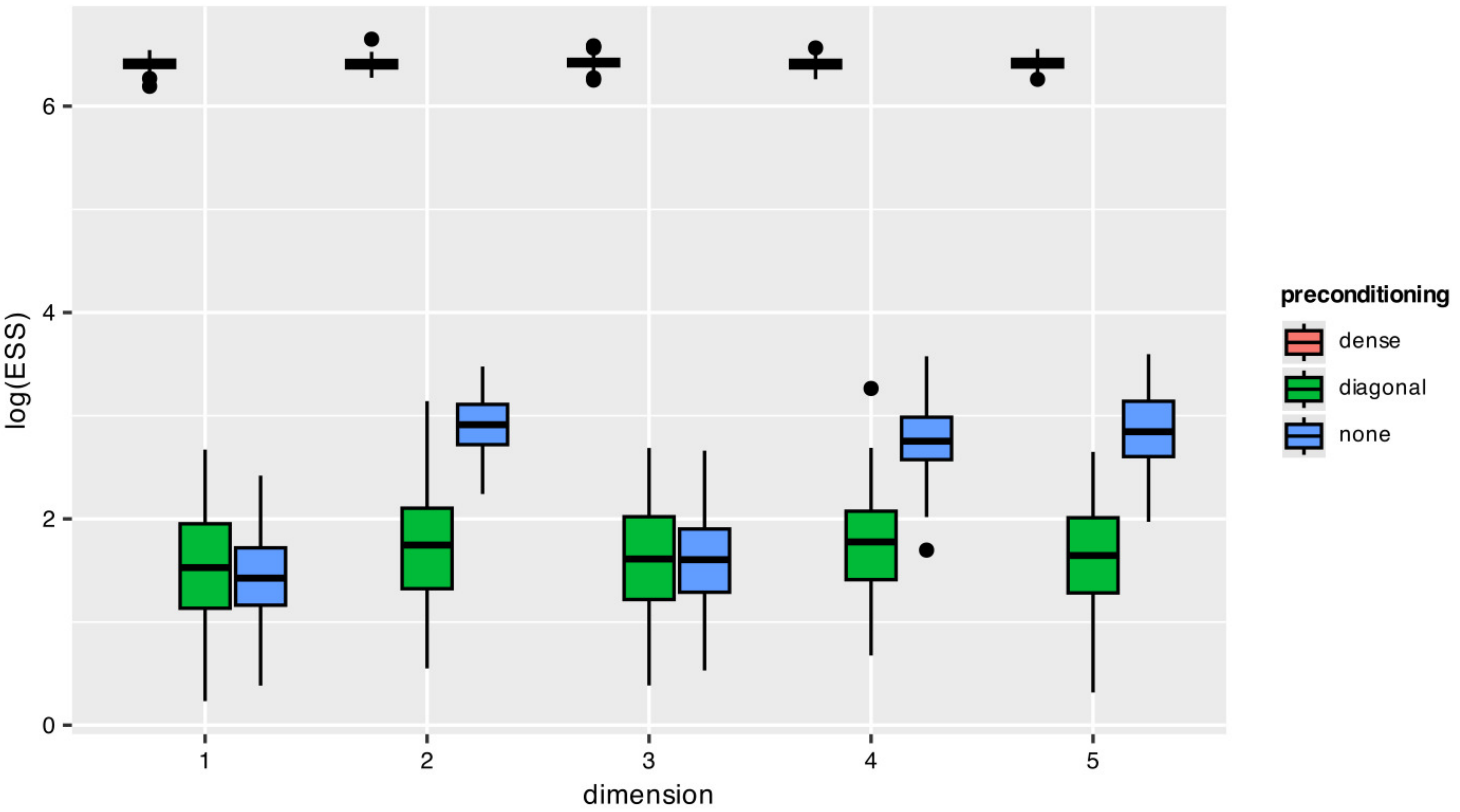
Diagonal Preconditioning: $L = \text{diag}(\Sigma_\pi)^{-\frac{1}{2}}$
Gaussian target:

$$\nabla_x^2 U(x) = \Sigma_\pi^{-1} \text{ so } \kappa_L = \|\text{diag}(\Sigma_\pi)^{\frac{1}{2}}\Sigma_\pi^{-1}\text{diag}(\Sigma_\pi)^{\frac{1}{2}}\|_2 \|\text{diag}(\Sigma_\pi)^{-\frac{1}{2}}\Sigma_\pi \text{diag}(\Sigma_\pi)^{-\frac{1}{2}}\|_2 = \|C_\pi^{-1}\|_2 \|C_\pi\|_2$$

There exist Gaussian targets for which $L = \text{diag}(\Sigma_\pi)^{-\frac{1}{2}}$ *increases the condition number*

$$\Sigma_\pi = \begin{pmatrix} 4.07, -3.90, 1.66 \\ -3.90, 3.73, -1.59 \\ 1.66, -1.59, 0.72 \end{pmatrix} \implies \kappa = 23{,}000, \kappa_L = 31{,}000$$

Diagonal Preconditioning for Sampling



Linear Preconditioning: Bounding κ_L

Theorem: For a given preconditioner $L \in GL_d(\mathbb{R})$ such that there exists $\epsilon > 0$ for which

$$\|\nabla_x^2 U(x) - LL^T\|_2 \leq m\epsilon \quad (1)$$

for all $x \in \mathbb{R}^d$, we can bound κ_L in the following way:

$$\kappa_L \leq \left(1 + \frac{m}{\sigma_d(L)^2}\epsilon\right)\left(1 + \kappa(L)^2\epsilon\right)$$

Existence of L in (1) is implied by $\|\nabla^2 U(x) - \nabla^2 U(y)\|_2 \leq m\epsilon$ for all $x, y \in \mathbb{R}^d$, therefore

Bounds inform decisions at each stage of the process: pre-check, construction, verification

Nonlinear Preconditioning

Call κ_g the condition number after general transform $g : \mathcal{X} \to \mathcal{Y}$

Proposition: It is impossible to use linear preconditioning to achieve optimality ($\kappa_g = 1$) when π is not a Gaussian

Proposition: There exist targets with *arbitrarily high condition number* that *gets worse* under *any linear preconditioning whatsoever* (excluding $L \in O(d)$)

Which g to use?

Take-aways

- Conditioning describes how well an algorithm works on a problem via a quantity known as the condition number
- Finding the condition number is often as hard as the problem itself: bounds on it are useful since...
- It is ubiquitous in the fields of numerical linear algebra and convex optimisation. It is less well known in sampling, but nonetheless important.
- Preconditioning is a transformation which lowers the condition number.
- We provide results on current preconditioning practices in sampling.
- We provide generic bounds on the condition number.

Thanks! 🐒

https://arxiv.org/abs/2312.04898