High-Dimensional Gaussian Process Inference with Derivatives

Filip de Roos[1,2], Alexandra Gessner[1,2], Philipp Hennig[1,2]
[1]Max Planck Institute for Intelligent Systems, Tübingen, [2]University of Tübingen

Background — Gradient Inference

A Gaussian process $f \sim \mathcal{GP}(\mu, k)$ is a random process defined by a mean function $\mu : \mathbb{R}^D \mapsto \mathbb{R}$ and covariance function $k : \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbb{R}$ such that f evaluated at a finite set of inputs follow a multi-variate normal distribution.

A well-known property of GPs is their closure under linear operations. For the general linear operators $\mathcal{L}$ and $\mathcal{M}$ acting on f the joint distribution of $\mathcal{L}f$ and $\mathcal{M}f$ is

$$\begin{bmatrix}\mathcal{L}f \\ \mathcal{M}f\end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix}\mathcal{L}\mu \\ \mathcal{M}\mu\end{bmatrix}, \begin{bmatrix}\mathcal{L}k\mathcal{L}' & \mathcal{L}k\mathcal{M}' \\ \mathcal{M}k\mathcal{L}' & \mathcal{M}k\mathcal{M}'\end{bmatrix}\right).$$

Given a set of input-output pairs $\{X, Y\} \in \{\mathbb{R}^{D\times N}, \mathbb{R}^N\}$ we can reason about properties of the underlying function by conditioning the GP on quantities of interest.

$$\mathcal{L}f(\cdot) \mid \mathcal{M}Y, X = \mathcal{L}\mu(\cdot) + \mathcal{L}k(\cdot, X)\mathcal{M}'(\mathcal{M}k(X,X)\mathcal{M}')^{-1}(\mathcal{M}Y - \mathcal{M}\mu(X))$$

Inference of a function $f : \mathbb{R}^D \to \mathbb{R}$ with N observations has cost

	compute	**memory**	$\mathcal{M}$
GP inference (functions)	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	Id
GP inference (**gradients**)	$\mathcal{O}((DN)^3)$	$\mathcal{O}((DN)^2)$	∇

Cost of GP inference

$\rightarrow$ 1 gradient observation $\widehat{=}$ D function evaluations

Kernel Structure — Families of Kernels

Notation

- Write a general kernel as a function of a scalar similarity measure $r \in \mathbb{R}$ as $k(\boldsymbol{x}_a, \boldsymbol{x}_b) = k(r(\boldsymbol{x}_a, \boldsymbol{x}_b)) = k_{ab}(r)$, so **subscripts indicate data** index.
- Introduce $k'_{ab} = \frac{\partial k(\boldsymbol{x}_a, \boldsymbol{x}_b)}{\partial r}$ and $k''_{ab} = \frac{\partial^2 k(\boldsymbol{x}_a, \boldsymbol{x}_b)}{\partial r^2}$.
- Define $\partial_a{}^i = \frac{\partial}{\partial \boldsymbol{x}_a{}^i}$ and similarly for $\partial_b{}^j$ so **superscripts indicate dimension** index.

The derivatives of a kernel k w.r.t. $\partial \boldsymbol{x}_a{}^i$ and $\partial \boldsymbol{x}_b{}^j$ is

$$\partial_a{}^i \partial_b{}^j k(r) = k'_{ab}(r) \cdot \partial_a{}^i \partial_b{}^j r + k''_{ab}(r) \cdot (\partial_a{}^i r)(\partial_b{}^j r).$$

Dot Product Kernels

$$r = (\boldsymbol{x}_a - \boldsymbol{c})^\top \Lambda (\boldsymbol{x}_b - \boldsymbol{c})$$
$$\partial_a{}^i \partial_b{}^j k(r) = k'_{ab}(r) \cdot \Lambda^{ij} + k''_{ab}(r) \cdot (\boldsymbol{x}_b - \boldsymbol{c})^i (\boldsymbol{x}_a - \boldsymbol{c})^j$$

Stationary Kernels

$$r = (\boldsymbol{x}_a - \boldsymbol{x}_b)^\top \Lambda (\boldsymbol{x}_a - \boldsymbol{x}_b)$$
$$\partial_a{}^i \partial_b{}^j k(r) = k'_{ab}(r) \cdot \Lambda^{ij} + (-k''_{ab}(r)) \cdot (\boldsymbol{x}_a - \boldsymbol{x}_b)^i (\boldsymbol{x}_a - \boldsymbol{x}_b)^j$$

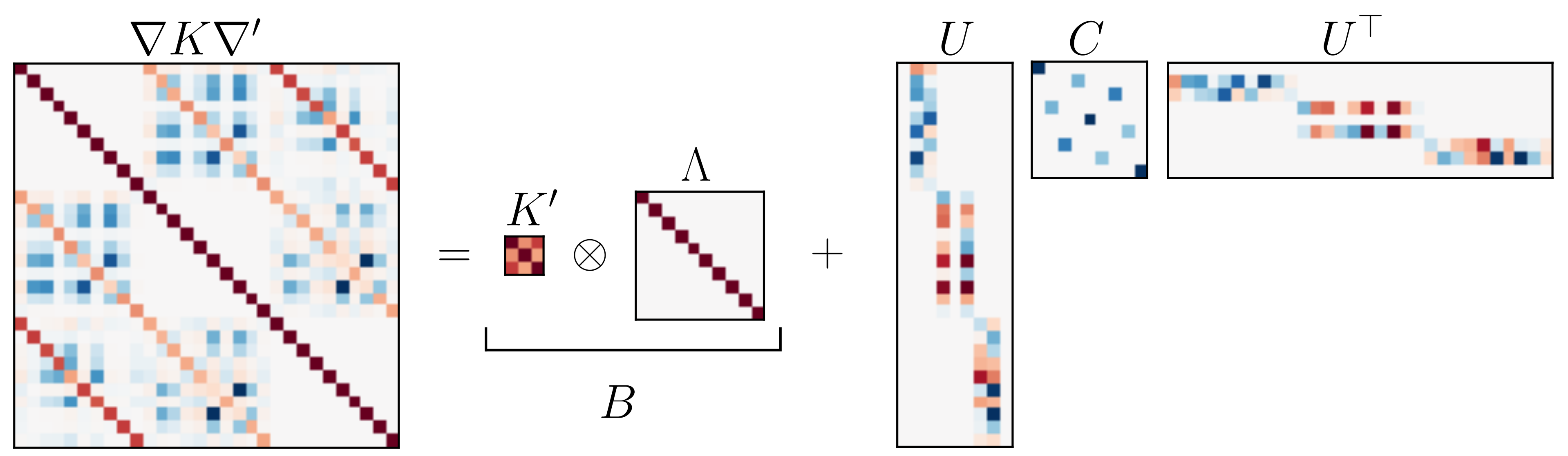
Summary

Gaussian processes is a popular framework for Bayesian modeling but it is often hindered by its $\mathcal{O}(N^3)$ computational scaling in data points (N). For inference with gradient observations the scaling further deteriorates with the dimensionality (D).

In this work we show how gradient observations can be efficiently included to speed up inference when the number of observations is lower than the dimensionality of the input ($N < D$). We show:

- The cost of **exact gradient inference** can be done in $\mathcal{O}(DN^2 + N^6)$ instead of $\mathcal{O}((DN)^3)$. This is useful for **high dimensions** and **few observations**.
- A special case for linear algebra with lower $\mathcal{O}(DN^2 + N^3)$ computational cost.
- Conceptual algorithms for **optimization** and **sampling** in high dimensions.
- **Reduced storage** from $\mathcal{O}((DN)^2)$ to $\mathcal{O}(DN + N^2)$ which can be used for implicit matrix-vector multiplication.

Kernel Decomposition — $D > N$



Kernel Gram matrix for RBF kernel with $N = 3$ gradient observations in $D = 10$ dimensions.

- Sparsity can be efficiently utilized with Kronecker algebra.

Woodbury's Matrix Inversion Lemma

$$\left(B + UCU^\top\right)^{-1} = B^{-1} - B^{-1}U\left(C^{-1} + U^\top B^{-1} U\right)^{-1} U^\top B^{-1}$$

- B is cheap to invert for small N due to Kronecker structure ($\mathcal{O}(N^3) + \mathrm{inv}(\Lambda)$).
- Lemma requires inversion of a matrix with the same size as $C \in \mathbb{R}^{N^2 \times N^2}$.

Contact @uni-tuebingen.de

filip.de-roos*

alexandra.gessner*

philipp.hennig*

Applications — High-Dimensional

Optimization

Algorithms that propose step directions for optimization.

[GP-H] Infer Hessian $H = \nabla\nabla^\top f(\boldsymbol{x})$ from $\nabla f(\boldsymbol{x})$ to determine step direction $\boldsymbol{d} = -\bar{H}(\boldsymbol{x})^{-1}\nabla f(\boldsymbol{x})$.

[GP-X] Swap inference to learn a mapping $\nabla f(\boldsymbol{x}) \to \boldsymbol{x}$ and infer local minimizer where $\nabla f = 0$.

Sampling

Hamiltonian Monte Carlo (HMC) proposes states of $P(\boldsymbol{x})$ as solutions to the system of ODEs

$$\dot{\boldsymbol{x}} = \frac{\boldsymbol{p}}{m} \quad \text{and} \quad \dot{\boldsymbol{p}} = \nabla \log P(\boldsymbol{x}).$$

with $\boldsymbol{p} \sim \mathcal{N}(0, mI)$. A lightweight surrogate alleviates large costs from repeated evaluation of $\nabla \log P(\boldsymbol{x})$ in the leapfrog integrator.

Iterative Inference

It is possible to solve $\nabla K \nabla' \operatorname{vec}(Z) = \operatorname{vec}(\nabla Y)$ with an iterative linear solver (CG) that only requires access to matrix-vector multiplications (`mvms`) of $\nabla K \nabla'$.

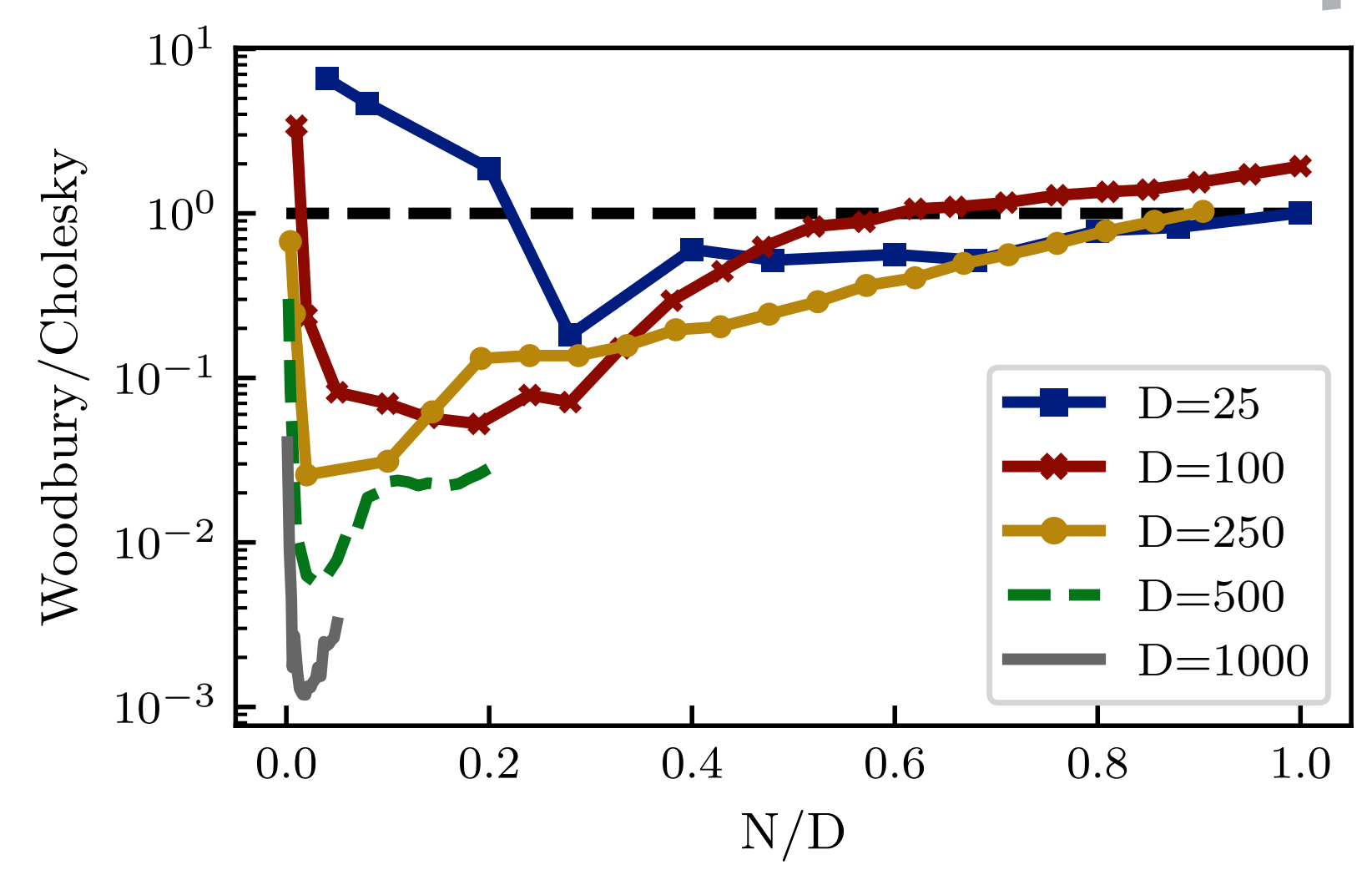
For $V \in \mathbb{R}^{D\times N}$ and $\operatorname{vec}(V) \in \mathbb{R}^{DN}$ we have

$$\nabla K \nabla' \operatorname{vec}(\mathrm{V}) = \operatorname{vec}(\Lambda V K') + \operatorname{vec}(\Lambda X (K'' \odot (X^\top \Lambda V)))$$

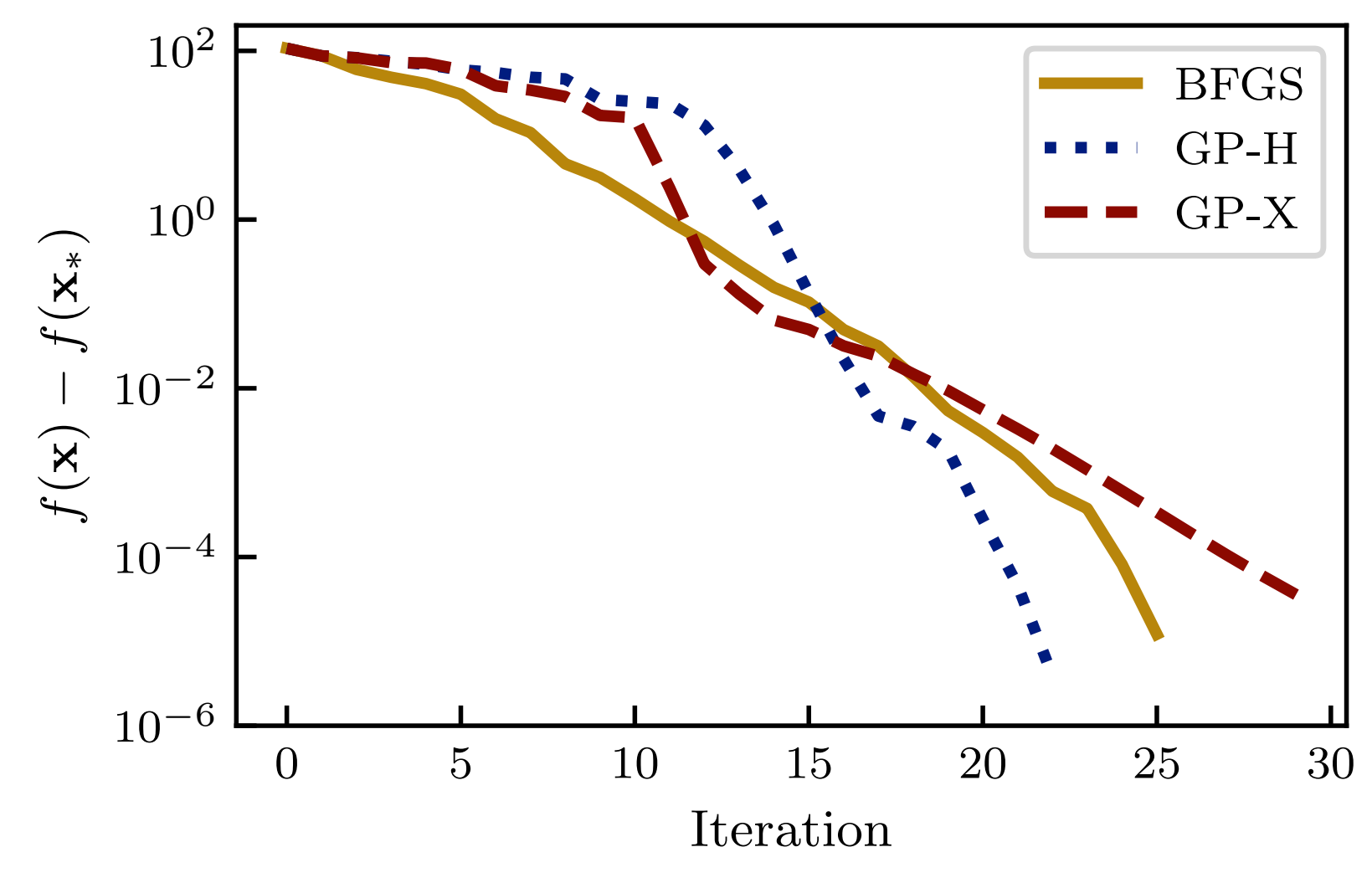
for a dot product kernel (with $\boldsymbol{c} = 0$).

Results — Proof-of-Concept

Runtime: cpu(Woodbury) divided by cpu(Cholesky) of the kernel Gram inversion for different dimensions and Gram matrices up to size $50\,000$. High D and low N can drastically speed up inversion.



Nonlinear Optimization: Convergence comparison of two proposed optimization routines with BFGS on a 100-d Rosenbrock function. All algorithms share the same line search routine.



arXiv: 2102.07542 https://github.com/fidero/gp-derivative