Scalable First-Order Bayesian Optimization via Structured Automatic Differentiation

Sebastian Ament and Carla Gomes

Cornell University

Bayesian Optimization

is designed to *globally* optimize functions that are

Bayesian Optimization

is designed to *globally* optimize functions that are

- expensive to evaluate $

Bayesian Optimization

is designed to *globally* optimize functions that are

- expensive to evaluate 
- non-convex 

Bayesian Optimization

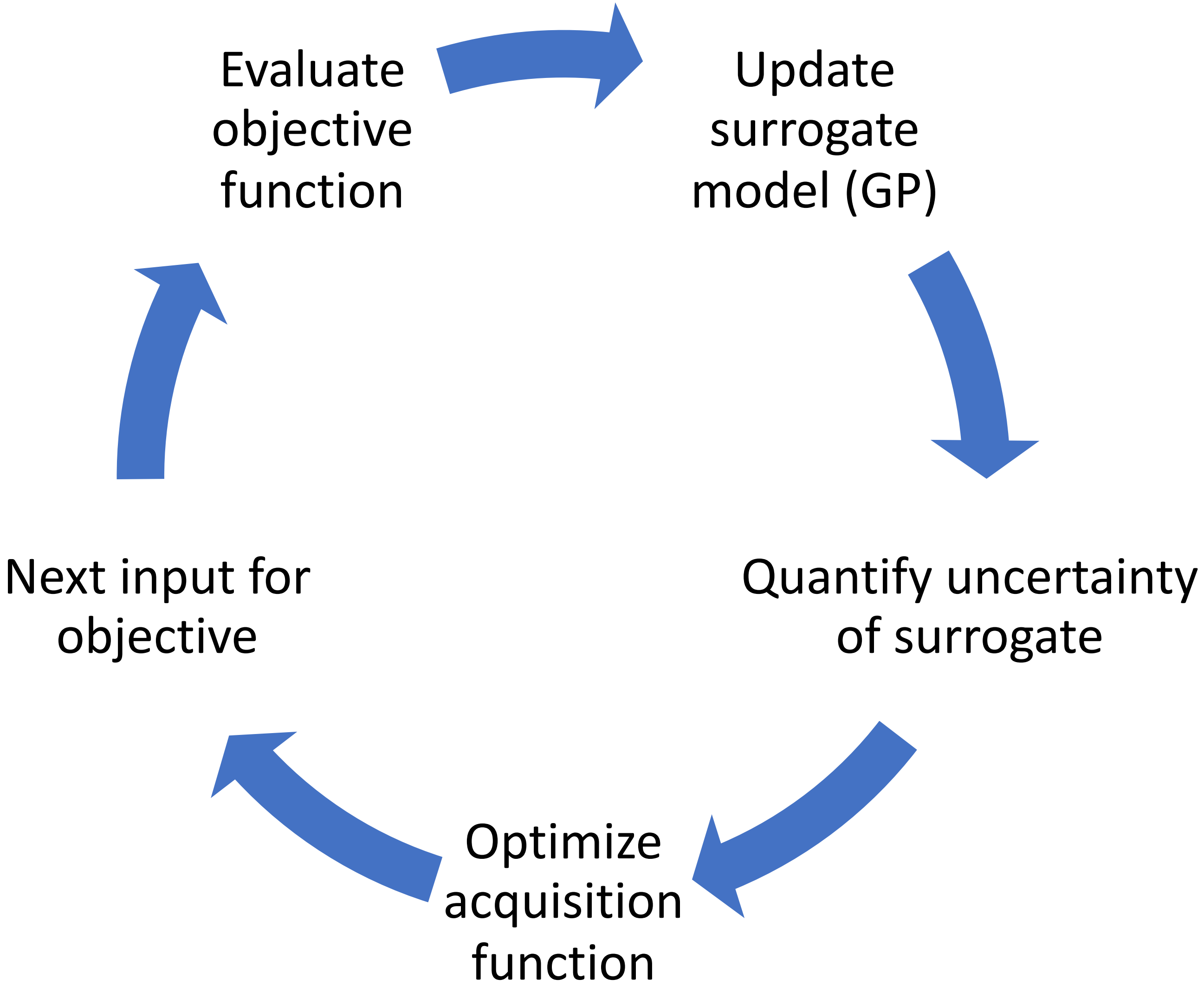
is designed to *globally* optimize functions that are

- expensive to evaluate

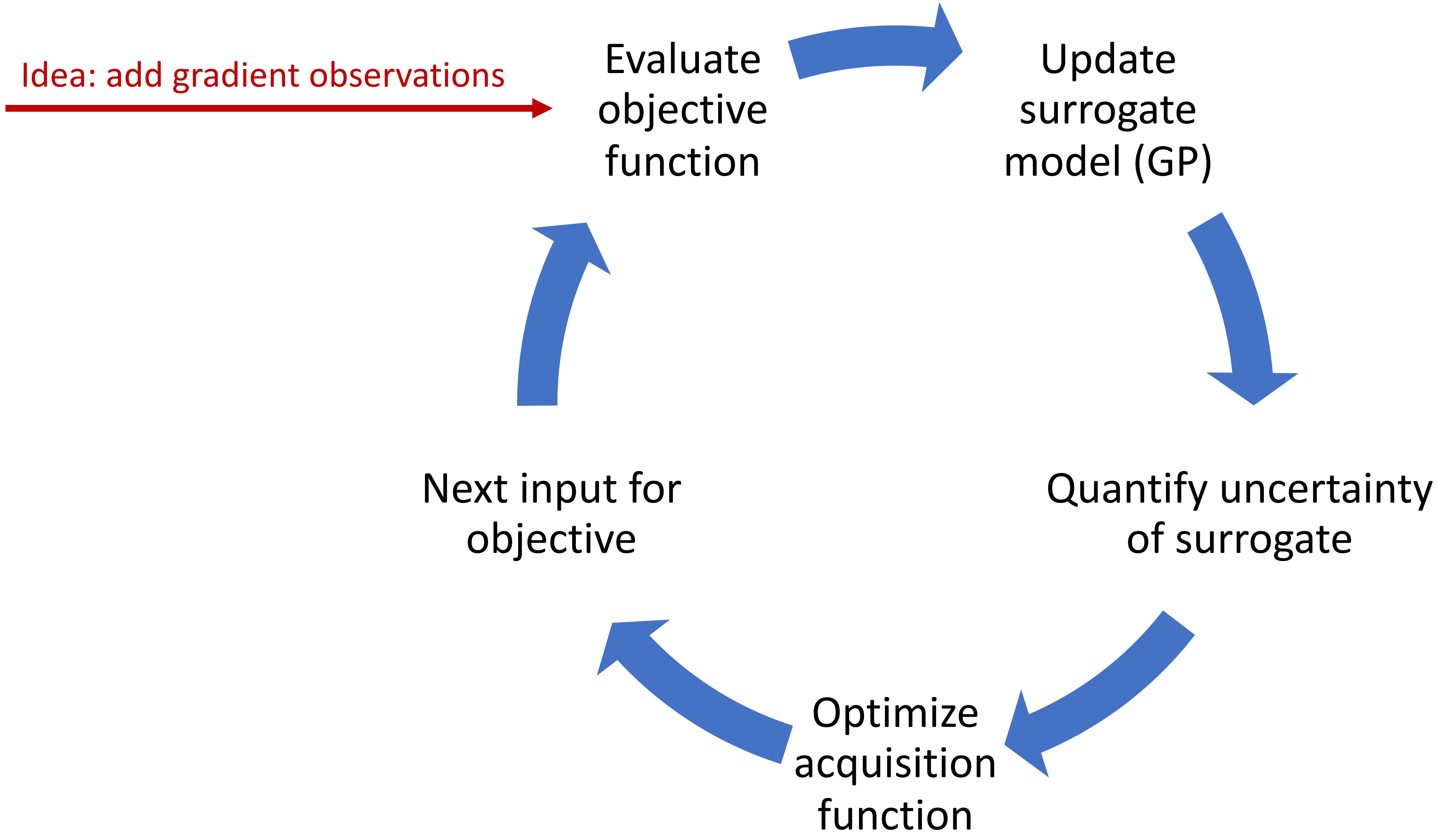
- non-convex

- black boxes

The Bayesian Optimization Loop



The Bayesian Optimization Loop



The Problem: A Large Matrix

- n – number of evaluations of the objective
- d – number of parameters of the objective

The Problem: A Large Matrix

- n – number of evaluations of the objective
- d – number of parameters of the objective
- Including gradient information into a GP surrogate involves
  - nd by nd matrices

The Problem: A Large Matrix

- n – number of evaluations of the objective
- d – number of parameters of the objective
- Including gradient information into a GP surrogate involves
  - nd by nd matrices
  - $O(n^2 d^2)$ operations for matrix-vector multiplications (MVMs)

The Problem: A Large Matrix

- n – number of evaluations of the objective
- d – number of parameters of the objective
- Including gradient information into a GP surrogate involves
  - nd by nd matrices
  - $O(n^2 d^2)$ operations for matrix-vector multiplications (MVMs)
  - $O(n^3 d^3)$ operations for matrix inversion

The Problem: A Large Matrix

- n – number of evaluations of the objective
- d – number of parameters of the objective
- Including gradient information into a GP surrogate involves
  - nd by nd matrices
  - $O(n^2 d^2)$ operations for matrix-vector multiplications (MVMs)
  - $O(n^3 d^3)$ operations for matrix inversion
- Our work reduces this to $O(n^2 d)$ for MVMs

The Problem: A Large Matrix

- n – number of evaluations of the objective
- d – number of parameters of the objective
- Including gradient information into a GP surrogate involves
  - nd by nd matrices
  - $O(n^2 d^2)$ operations for matrix-vector multiplications (MVMs)
  - $O(n^3 d^3)$ operations for matrix inversion
- Our work reduces this to $O(n^2 d)$ for MVMs
- Use iterative solvers for solves

The Problem: A Large Matrix

- $k(x, y)$ – covariance function of two inputs x and y
- Covariance function of *gradients* is given by $G[k]$, where

$$G_{ij} = \partial_{x_i} \partial_{y_j}$$

- $G[k]$ – is d by d
- We show how to compute MVMs with $G[k]$ in $O(d)$.

Scalable First-Order Bayesian Optimization via Structured Automatic Differentiation, Ament and Gomes, *ICML 2022*

The Problem: A Large Matrix

- If we have an $O(d)$ MVM with $G[k]$, we have an MVM with K^∇ in $O(n^2 d)$.

- K^∇ – covariance matrix between gradients of *all* points (nd by nd)

$$K_{ij}^\nabla = G[k](x_i, y_j)$$

Scalable First-Order Bayesian Optimization via Structured Automatic Differentiation, Ament and Gomes, *ICML 2022*

The Solution: Structure-Aware AD

Many kernel can be written as

$$k(\mathbf{x}, \mathbf{y}) = f(\text{proto}(\mathbf{x}, \mathbf{y})),$$

$$\text{where proto}(\mathbf{x}, \mathbf{y}) = (\mathbf{r} \cdot \mathbf{r}),\ (\mathbf{c} \cdot \mathbf{r}),\ \text{or}\ (\mathbf{x} \cdot \mathbf{y})$$

For these choices, we have

$$\mathbf{G}[\mathbf{r} \cdot \mathbf{r}] = -\mathbf{I}_d,\ \ \mathbf{G}[\mathbf{c} \cdot \mathbf{r}] = \mathbf{0}_{d \times d},\ \ \text{and}\ \ \mathbf{G}[\mathbf{x} \cdot \mathbf{y}] = \mathbf{I}_d.$$

Scalable First-Order Bayesian Optimization via Structured Automatic Differentiation, Ament and Gomes, *ICML 2022*

The Solution: Structure-Aware AD

Many kernel can be written as

$$k(\mathbf{x}, \mathbf{y}) = f(\text{proto}(\mathbf{x}, \mathbf{y})),$$

$$\text{where proto}(\mathbf{x}, \mathbf{y}) = (\mathbf{r} \cdot \mathbf{r}),\ (\mathbf{c} \cdot \mathbf{r}),\ \text{or}\ (\mathbf{x} \cdot \mathbf{y})$$

For these choices, we have

$$\mathbf{G}[\mathbf{r} \cdot \mathbf{r}] = -\mathbf{I}_d,\ \ \mathbf{G}[\mathbf{c} \cdot \mathbf{r}] = \mathbf{0}_{d \times d},\ \ \text{and}\ \ \mathbf{G}[\mathbf{x} \cdot \mathbf{y}] = \mathbf{I}_d.$$

→ $O(d)$ MVM with G

Scalable First-Order Bayesian Optimization via Structured Automatic Differentiation, Ament and Gomes, *ICML 2022*

The Solution: Structure-Aware AD

A Chain Rule Many kernels can be expressed as $k = f \circ g$ where g is scalar-valued. For these types of kernels, we have

$$\mathbf{G}[f \circ g] = (f' \circ g)\, \mathbf{G}[g] + (f'' \circ g)\, \nabla_{\mathbf{x}}[g] \nabla_{\mathbf{y}}[g]^{\top}.$$

The Solution: Structure-Aware AD

A Chain Rule Many kernels can be expressed as $k = f \circ g$ where g is scalar-valued. For these types of kernels, we have

$$\mathbf{G}[f \circ g] = (f' \circ g)\, \mathbf{G}[g] + (f'' \circ g)\, \nabla_{\mathbf{x}}[g] \nabla_{\mathbf{y}}[g]^\top.$$

⟶ $O(d)$ MVM with G

The Solution: Structure-Aware AD

Sums and Products of kernels: $k = \prod_i^r k_i$

$$\mathbf{G}[k] = \sum_{i=1}^{r} \mathbf{G}[k_i]p_i + \mathbf{J}_{\mathbf{x}}[\mathbf{k}]^\top \, \mathbf{P} \, \mathbf{J}_{\mathbf{y}}[\mathbf{k}],$$

Direct sums and products:

$$[\mathbf{G}k_i]_{ii} = [\partial_{x_i}\partial_{y_i}k_i] \prod_{j \neq i} k_j, \quad \text{and} \quad [\mathbf{J}_{\mathbf{x}}\mathbf{k}]_{ii} = \partial_{x_i}k_i.$$

⟶ $O(d)$ MVM with G

Scalable First-Order Bayesian Optimization via Structured Automatic Differentiation, Ament and Gomes, *ICML 2022*

The Solution: Structure-Aware AD

And more

Rescaling

$$\mathbf{G}[k](\mathbf{x},\mathbf{y}) = f(\mathbf{x})\mathbf{G}[h](\mathbf{x},\mathbf{y})f(\mathbf{y}) + \nabla_{\mathbf{x}} \begin{bmatrix} f(\mathbf{x}) & k(\mathbf{x},\mathbf{y}) \end{bmatrix} \begin{bmatrix} h(\mathbf{x},\mathbf{y}) & f(\mathbf{y}) \\ f(\mathbf{x}) & 0 \end{bmatrix} \nabla_{\mathbf{y}} \begin{bmatrix} f(\mathbf{y}) & k(\mathbf{x},\mathbf{y}) \end{bmatrix}^{\top}$$

Warping

$$\mathbf{K}^{\nabla} = \mathrm{diag}(\mathbf{J}[\mathbf{u}](\mathbf{X}))^{\top}\, \mathbf{H}^{\nabla}\, \mathrm{diag}(\mathbf{J}[\mathbf{u}](\mathbf{X})).$$

Scalable First-Order Bayesian Optimization via Structured Automatic Differentiation, Ament and Gomes, *ICML 2022*

The Solution: Structure-Aware AD

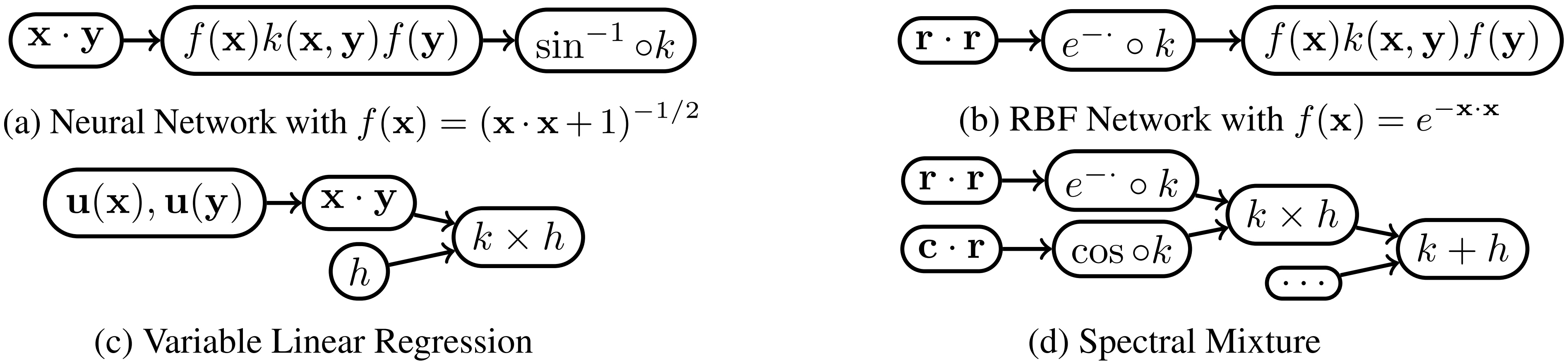


Figure 1: Computational graphs of composite kernels whose gradient kernel matrix can be expressed with the data-sparse structured expressions derived in Section 3.2. Inside a node, k and h refer to kernels computed by previous nodes.

$\longrightarrow$ $O(d)$ MVM with G

Scalable First-Order Bayesian Optimization via Structured Automatic Differentiation, Ament and Gomes, *ICML 2022*

Yet more: Hessian observations

$O(d^4)$ MVM with H

Scalable First-Order Bayesian Optimization via Structured Automatic Differentiation, Ament and Gomes, *ICML 2022*

Yet more: Hessian observations

C. Hessian Structure

Note that for arbitrary vectors $\mathbf{a}, \mathbf{b}$, not necessarily of the same length, $\mathbf{a} \otimes \mathbf{b} = \text{vec}(\mathbf{b}\mathbf{a}^\top)$. This will come in handy to simplify certain expressions in the following.

Dot-Product Kernels First, note that

$$\nabla_{\mathbf{y}}^\top \text{vec}(\mathbf{y}\mathbf{y}^\top) = \mathbf{I}_d \otimes \mathbf{y} + \mathbf{y} \otimes \mathbf{I}_d \qquad \nabla_{\mathbf{y}} \nabla_{\mathbf{y}}^\top \text{vec}(\mathbf{y}\mathbf{y}^\top) = \mathbf{S}_{dd} + \mathbf{I}_{d^2}.$$

Where $\mathbf{S}_{dd}$ is a "shuffle" matrix such that $\mathbf{S}_{dd}\text{vec}(\mathbf{A}) = \text{vec}(\mathbf{A}^\top)$, and for square matrices $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{B} \in \mathbb{R}^{m \times m}$, the Kronecker sum is defined as $\mathbf{A} \oplus \mathbf{B} \stackrel{\text{def}}{=} \mathbf{A} \otimes \mathbf{I}_m + \mathbf{I}_n \otimes \mathbf{B}$. Then for dot-product kernels, we have

$$[\mathbf{h}_{\mathbf{x}} k](\mathbf{x}, \mathbf{y}) = f''(r)\text{vec}(\mathbf{y}\mathbf{y}^\top).$$

$$[\mathbf{h}_{\mathbf{x}} \nabla_{\mathbf{y}}^\top k](\mathbf{x}, \mathbf{y}) = f''(r)(\mathbf{I}_d \otimes \mathbf{y} + \mathbf{y} \otimes \mathbf{I}_d) + f'''(r)\text{vec}(\mathbf{y}\mathbf{y}^\top)\mathbf{x}^\top.$$

$$[\mathbf{h}_{\mathbf{y}}^\top \mathbf{h}_{\mathbf{x}} k](\mathbf{x}, \mathbf{y}) = (\mathbf{I}_{d^2} + \mathbf{S}_{dd})[f''(r)\mathbf{I}_{d^2} + f'''(r)(\mathbf{y}\mathbf{x}^\top \oplus \mathbf{y}\mathbf{x}^\top)] + f''''(r)\text{vec}(\mathbf{y}\mathbf{y}^\top)\text{vec}(\mathbf{x}\mathbf{x}^\top)^\top.$$

Isotropic Kernels Then for isotropic product kernels with $r = \|\mathbf{r}\|_2^2$, we have

$$\mathbf{J}_{\mathbf{x}}\text{vec}(\mathbf{r}\mathbf{r}^\top) = \mathbf{I}_d \otimes \mathbf{r} + \mathbf{r} \otimes \mathbf{I}_d \qquad \mathbf{H}_{\mathbf{y}}\text{vec}(\mathbf{r}\mathbf{r}^\top) = \mathbf{S}_{dd} + \mathbf{I}_{d^2}.$$

Which implies

$$[\mathbf{h}_{\mathbf{x}} k](\mathbf{x}, \mathbf{y}) = f'(r)\text{vec}(\mathbf{I}_d) + f''(r)\text{vec}(\mathbf{r}\mathbf{r}^\top).$$

$$[\mathbf{h}_{\mathbf{x}} \nabla_{\mathbf{y}}^\top k](\mathbf{x}, \mathbf{y}) = -f''(r)(\mathbf{I}_d \otimes \mathbf{r} + \mathbf{r} \otimes \mathbf{I}_d) - [f''(r)\text{vec}(\mathbf{I}_d) + f'''(r)\text{vec}(\mathbf{r}\mathbf{r}^\top)]\mathbf{r}^\top.$$

$$\mathbf{h}_{\mathbf{y}}^\top \mathbf{h}_{\mathbf{x}} k(\mathbf{x}, \mathbf{y}) = (\mathbf{I}_{d^2} + \mathbf{S}_{dd})[f''(r)\mathbf{I}_{d^2} + f'''(r)(\mathbf{r}\mathbf{r}^\top \oplus \mathbf{r}\mathbf{r}^\top)] + \begin{bmatrix}\text{vec}(\mathbf{I}_d) & \text{vec}(\mathbf{r}\mathbf{r}^\top)\end{bmatrix} \begin{bmatrix} f''(r) & f'''(r) \\ f'''(r) & f''''(r) \end{bmatrix} \begin{bmatrix}\text{vec}(\mathbf{I}_d) & \text{vec}(\mathbf{r}\mathbf{r}^\top)\end{bmatrix}^\top.$$

A Chain Rule $k(\mathbf{x}, \mathbf{y}) = (f \circ g)(\mathbf{x}, \mathbf{y})$.

$$[\mathbf{h}_{\mathbf{x}} k](\mathbf{x}, \mathbf{y}) = f'(r)\mathbf{h}_{\mathbf{x}}[g] + f''(r)\text{vec}(\nabla_{\mathbf{x}} g \nabla_{\mathbf{x}} g^\top).$$

$$[\mathbf{h}_{\mathbf{x}} \nabla_{\mathbf{y}}^\top k](\mathbf{x}, \mathbf{y}) = f''(r)(\mathbf{H}_{\mathbf{x}} g \otimes \nabla_{\mathbf{y}} g + \nabla_{\mathbf{y}} g \otimes \mathbf{H}_{\mathbf{x}} g) + [f''(r)\mathbf{h}_{\mathbf{x}}[g]) + f'''(r)\text{vec}(\nabla_{\mathbf{x}} g \nabla_{\mathbf{x}} g^\top)]\nabla_{\mathbf{y}} g^\top.$$

$$\mathbf{h}_{\mathbf{x}} \mathbf{h}_{\mathbf{y}}^\top k(\mathbf{x}, \mathbf{y}) = (\mathbf{I}_{d^2} + \mathbf{S}_{dd})[f''(r)\mathbf{I}_{d^2} + f'''(r)(\nabla_{\mathbf{x}} g \nabla_{\mathbf{x}} g^\top \oplus \nabla_{\mathbf{y}} g \nabla_{\mathbf{y}} g^\top)] + \begin{bmatrix}\mathbf{h}_{\mathbf{x}} g & \text{vec}(\nabla_{\mathbf{x}} g \nabla_{\mathbf{x}} g^\top)\end{bmatrix} \begin{bmatrix} f''(r) & f'''(r) \\ f'''(r) & f''''(r) \end{bmatrix} \begin{bmatrix}\mathbf{h}_{\mathbf{y}} g & \text{vec}(\nabla_{\mathbf{y}} g \nabla_{\mathbf{y}} g^\top)\end{bmatrix}^\top.$$

Vertical Scaling $k(\mathbf{x}, \mathbf{y}) = f(\mathbf{x})h(\mathbf{x}, \mathbf{y})f(\mathbf{y})$ for a scalar-valued f, then

$$\begin{aligned}
\mathbf{h}_{\mathbf{x}} k(\mathbf{x}, \mathbf{y}) &= \mathbf{h}_{\mathbf{x}}[f(\mathbf{x})h(\mathbf{x}, \mathbf{y})]f(\mathbf{y}) \\
&= [f(\mathbf{x})\mathbf{h}_{\mathbf{x}}[h](\mathbf{x}, \mathbf{y}) \\
&\quad + \mathbf{h}[f](\mathbf{x})h(\mathbf{x}, \mathbf{y}) \\
&\quad + \nabla_{\mathbf{x}}[h](\mathbf{x}, \mathbf{y}) \otimes \nabla[f](\mathbf{x}) \\
&\quad + \nabla[f](\mathbf{x}) \otimes \nabla_{\mathbf{x}}[h](\mathbf{x}, \mathbf{y})]\ f(\mathbf{y}) \\
[\mathbf{h}_{\mathbf{x}} \nabla_{\mathbf{y}}^\top k](\mathbf{x}, \mathbf{y}) &= [f(\mathbf{x})[\mathbf{h}_{\mathbf{x}} \nabla_{\mathbf{y}}^\top h](\mathbf{x}, \mathbf{y}) \\
&\quad + \mathbf{h}[f](\mathbf{x})[\nabla_{\mathbf{y}}^\top h](\mathbf{x}, \mathbf{y}) \\
&\quad + \mathbf{G}[h](\mathbf{x}, \mathbf{y}) \otimes \nabla[f](\mathbf{x}) \\
&\quad + \nabla[f](\mathbf{x}) \otimes \mathbf{G}[h](\mathbf{x}, \mathbf{y})]\ f(\mathbf{y}) \\
&\quad + \mathbf{h}_{\mathbf{x}}[f(\mathbf{x})h(\mathbf{x}, \mathbf{y})]\nabla_{\mathbf{y}}^\top\ f(\mathbf{y}) \\
[\mathbf{h}_{\mathbf{x}} \mathbf{h}_{\mathbf{y}}^\top k](\mathbf{x}, \mathbf{y}) &= [f(\mathbf{x})[\mathbf{h}_{\mathbf{x}} \mathbf{h}_{\mathbf{y}}^\top h](\mathbf{x}, \mathbf{y}) \\
&\quad + \mathbf{h}[f](\mathbf{x})[\mathbf{h}_{\mathbf{y}}^\top h](\mathbf{x}, \mathbf{y}) \\
&\quad + \mathbf{G}[h](\mathbf{x}, \mathbf{y}) \otimes \nabla[f](\mathbf{x})\nabla^\top[f](\mathbf{y}) \\
&\quad + \nabla[f](\mathbf{x})\nabla^\top[f](\mathbf{y}) \otimes \mathbf{G}[h](\mathbf{x}, \mathbf{y})]\ f(\mathbf{y}) \\
&\quad + \mathbf{h}_{\mathbf{x}}[f(\mathbf{x})h(\mathbf{x}, \mathbf{y})]\mathbf{h}_{\mathbf{y}}^\top\ f(\mathbf{y})
\end{aligned}$$

Again, we observe a structured representation of the Hessian-kernel elements which permit a multiply in $\mathcal{O}(d^2)$ operations.

Warping $k(\mathbf{x}, \mathbf{y}) = h(\mathbf{u}(\mathbf{x}), \mathbf{u}(\mathbf{y}))$,

$$\begin{aligned}
\mathbf{h}_{\mathbf{x}} k(\mathbf{x}, \mathbf{y}) &= (\mathbf{J} \otimes \mathbf{J})^\top[\mathbf{u}](\mathbf{x})\ [\mathbf{h}_{\mathbf{x}} h](\mathbf{u}(\mathbf{x}), \mathbf{u}(\mathbf{y})) \\
[\mathbf{h}_{\mathbf{x}} \nabla_{\mathbf{y}}^\top k](\mathbf{x}, \mathbf{y}) &= (\mathbf{J} \otimes \mathbf{J})^\top[\mathbf{u}](\mathbf{x})\ [\mathbf{h}_{\mathbf{x}} \nabla_{\mathbf{y}}^\top h](\mathbf{u}(\mathbf{x}), \mathbf{u}(\mathbf{y}))\ \mathbf{J}[\mathbf{u}](\mathbf{y}) \\
[\mathbf{h}_{\mathbf{x}} \mathbf{h}_{\mathbf{y}}^\top k](\mathbf{x}, \mathbf{y}) &= (\mathbf{J} \otimes \mathbf{J})^\top[\mathbf{u}](\mathbf{x})\ [\mathbf{h}_{\mathbf{x}} \mathbf{h}_{\mathbf{y}}^\top h](\mathbf{u}(\mathbf{x}), \mathbf{u}(\mathbf{y}))\ (\mathbf{J} \otimes \mathbf{J})[\mathbf{u}](\mathbf{y}).
\end{aligned}$$

We therefore see that $\mathbf{K}^{\mathbf{H}} = \mathbf{h}_{\mathbf{x}} \mathbf{h}_{\mathbf{y}}^\top k(\mathbf{X}) = \mathbf{D}_{\mathbf{J}}[\mathbf{h}_{\mathbf{x}} \mathbf{h}_{\mathbf{y}}^\top h](\mathbf{X})\mathbf{D}_{\mathbf{J}}$, where $\mathbf{D}_{\mathbf{J}}$ is the block-diagonal matrix whose i^{th} block is equal to $(\mathbf{J} \otimes \mathbf{J})[\mathbf{u}](\mathbf{x}_i) = \mathbf{J}[\mathbf{u}](\mathbf{x}_i) \otimes \mathbf{J}[\mathbf{u}](\mathbf{x}_i)$. Note that for linearly warped kernels for which $\mathbf{u}(\mathbf{x}) = \mathbf{U}\mathbf{x}$, where $\mathbf{U} \in \mathbb{R}^{r \times d}$, we have $(\mathbf{J} \otimes \mathbf{J})[\mathbf{u}](\mathbf{x}_i) = \mathbf{U} \otimes \mathbf{U}$ so that we can multiply with the kernel matrix $\mathbf{K}^{\mathbf{H}}$ in $\mathcal{O}(n^2 r^2 + n(d^2 r + r^2 d))$. The complexity is due to the following property of Kronecker product:

$$(\mathbf{U} \otimes \mathbf{U})\text{vec}(\mathbf{H}) = \text{vec}(\mathbf{U}\mathbf{H}\mathbf{U}^\top),$$

which can be computed in $\mathcal{O}(d^2 r + r^2 d)$ for every of the n Hessian observations.

$O(d^4)$ MVM with H

Scalable First-Order Bayesian Optimization via Structured Automatic Differentiation, Ament and Gomes, *ICML 2022*

Yet more: Hessian observations

C. Hessian Structure

Note that for arbitrary vectors $\mathbf{a}, \mathbf{b}$, not necessarily of the same length, $\mathbf{a} \otimes \mathbf{b} = \text{vec}(\mathbf{b}\mathbf{a}^\top)$. This will come in handy to simplify certain expressions in the following.

Dot-Product Kernels First, note that

$$\nabla_{\mathbf{y}}^\top \text{vec}(\mathbf{y}\mathbf{y}^\top) = \mathbf{I}_d \otimes \mathbf{y} + \mathbf{y} \otimes \mathbf{I}_d \qquad \nabla_{\mathbf{y}} \nabla_{\mathbf{y}}^\top \text{vec}(\mathbf{y}\mathbf{y}^\top) = \mathbf{S}_{dd} + \mathbf{I}_{d^2}.$$

Where $\mathbf{S}_{dd}$ is a "shuffle" matrix such that $\mathbf{S}_{dd}\text{vec}(\mathbf{A}) = \text{vec}(\mathbf{A}^\top)$, and for square matrices $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{B} \in \mathbb{R}^{m \times m}$, the Kronecker sum is defined as $\mathbf{A} \oplus \mathbf{B} \stackrel{\text{def}}{=} \mathbf{A} \otimes \mathbf{I}_m + \mathbf{I}_n \otimes \mathbf{B}$. Then for dot-product kernels, we have

$$[\mathbf{h}_{\mathbf{x}} k](\mathbf{x}, \mathbf{y}) = f''(r)\text{vec}(\mathbf{y}\mathbf{y}^\top).$$

$$[\mathbf{h}_{\mathbf{x}} \nabla_{\mathbf{y}}^\top k](\mathbf{x}, \mathbf{y}) = f''(r)(\mathbf{I}_d \otimes \mathbf{y} + \mathbf{y} \otimes \mathbf{I}_d) + f'''(r)\text{vec}(\mathbf{y}\mathbf{y}^\top)\mathbf{x}^\top.$$

$$[\mathbf{h}_{\mathbf{y}}^\top \mathbf{h}_{\mathbf{x}} k](\mathbf{x}, \mathbf{y}) = (\mathbf{I}_{d^2} + \mathbf{S}_{dd})[f''(r)\mathbf{I}_{d^2} + f'''(r)(\mathbf{y}\mathbf{x}^\top \oplus \mathbf{y}\mathbf{x}^\top)] + f''''(r)\text{vec}(\mathbf{y}\mathbf{y}^\top)\text{vec}(\mathbf{x}\mathbf{x}^\top)^\top.$$

Isotropic Kernels Then for isotropic product kernels with $r = \|\mathbf{r}\|_2^2$, we have

$$\mathbf{J}_{\mathbf{x}}\text{vec}(\mathbf{r}\mathbf{r}^\top) = \mathbf{I}_d \otimes \mathbf{r} + \mathbf{r} \otimes \mathbf{I}_d \qquad \mathbf{H}_{\mathbf{y}}\text{vec}(\mathbf{r}\mathbf{r}^\top) = \mathbf{S}_{dd} + \mathbf{I}_{d^2}.$$

Which implies

$$[\mathbf{h}_{\mathbf{x}} k](\mathbf{x}, \mathbf{y}) = f'(r)\text{vec}(\mathbf{I}_d) + f''(r)\text{vec}(\mathbf{r}\mathbf{r}^\top).$$

$$[\mathbf{h}_{\mathbf{x}} \nabla_{\mathbf{y}}^\top k](\mathbf{x}, \mathbf{y}) = -f''(r)(\mathbf{I}_d \otimes \mathbf{r} + \mathbf{r} \otimes \mathbf{I}_d) - [f''(r)\text{vec}(\mathbf{I}_d) + f'''(r)\text{vec}(\mathbf{r}\mathbf{r}^\top)]\mathbf{r}^\top.$$

$$\begin{aligned}\mathbf{h}_{\mathbf{y}}^\top \mathbf{h}_{\mathbf{x}} k(\mathbf{x}, \mathbf{y}) &= (\mathbf{I}_{d^2} + \mathbf{S}_{dd})[f''(r)\mathbf{I}_{d^2} + f'''(r)(\mathbf{r}\mathbf{r}^\top \oplus \mathbf{r}\mathbf{r}^\top)] \\ &+ \begin{bmatrix}\text{vec}(\mathbf{I}_d) & \text{vec}(\mathbf{r}\mathbf{r}^\top)\end{bmatrix} \begin{bmatrix} f''(r) & f'''(r) \\ f'''(r) & f''''(r)\end{bmatrix} \begin{bmatrix}\text{vec}(\mathbf{I}_d) & \text{vec}(\mathbf{r}\mathbf{r}^\top)\end{bmatrix}^\top.\end{aligned}$$

A Chain Rule $k(\mathbf{x}, \mathbf{y}) = (f \circ g)(\mathbf{x}, \mathbf{y})$.

$$[\mathbf{h}_{\mathbf{x}} k](\mathbf{x}, \mathbf{y}) = f'(r)\mathbf{h}_{\mathbf{x}}[g] + f''(r)\text{vec}(\nabla_{\mathbf{x}} g \nabla_{\mathbf{x}} g^\top).$$

$$[\mathbf{h}_{\mathbf{x}} \nabla_{\mathbf{y}}^\top k](\mathbf{x}, \mathbf{y}) = f''(r)(\mathbf{H}_{\mathbf{x}} g \otimes \nabla_{\mathbf{y}} g + \nabla_{\mathbf{y}} g \otimes \mathbf{H}_{\mathbf{x}} g) + [f''(r)\mathbf{h}_{\mathbf{x}}[g]) + f'''(r)\text{vec}(\nabla_{\mathbf{x}} g \nabla_{\mathbf{x}} g^\top)]\nabla_{\mathbf{y}} g^\top.$$

$$\begin{aligned}\mathbf{h}_{\mathbf{x}} \mathbf{h}_{\mathbf{y}}^\top k(\mathbf{x}, \mathbf{y}) &= (\mathbf{I}_{d^2} + \mathbf{S}_{dd})[f''(r)\mathbf{I}_{d^2} + f'''(r)(\nabla_{\mathbf{x}} g \nabla_{\mathbf{x}} g^\top \oplus \nabla_{\mathbf{y}} g \nabla_{\mathbf{y}} g^\top)] \\ &+ \begin{bmatrix}\mathbf{h}_{\mathbf{x}} g & \text{vec}(\nabla_{\mathbf{x}} g \nabla_{\mathbf{x}} g^\top)\end{bmatrix} \begin{bmatrix} f''(r) & f'''(r) \\ f'''(r) & f''''(r)\end{bmatrix} \begin{bmatrix}\mathbf{h}_{\mathbf{y}} g & \text{vec}(\nabla_{\mathbf{y}} g \nabla_{\mathbf{y}} g^\top)\end{bmatrix}^\top.\end{aligned}$$

Vertical Scaling $k(\mathbf{x}, \mathbf{y}) = f(\mathbf{x})h(\mathbf{x}, \mathbf{y})f(\mathbf{y})$ for a scalar-valued f, then

$$\begin{aligned}\mathbf{h}_{\mathbf{x}} k(\mathbf{x}, \mathbf{y}) =& \mathbf{h}_{\mathbf{x}}[f(\mathbf{x})h(\mathbf{x}, \mathbf{y})]f(\mathbf{y}) \\ =& [f(\mathbf{x})\mathbf{h}_{\mathbf{x}}[h](\mathbf{x}, \mathbf{y}) \\ &+ \mathbf{h}[f](\mathbf{x})h(\mathbf{x}, \mathbf{y}) \\ &+ \nabla_{\mathbf{x}}[h](\mathbf{x}, \mathbf{y}) \otimes \nabla[f](\mathbf{x}) \\ &+ \nabla[f](\mathbf{x}) \otimes \nabla_{\mathbf{x}}[h](\mathbf{x}, \mathbf{y})]\ f(\mathbf{y}) \\ [\mathbf{h}_{\mathbf{x}} \nabla_{\mathbf{y}}^\top k](\mathbf{x}, \mathbf{y}) =& [f(\mathbf{x})[\mathbf{h}_{\mathbf{x}} \nabla_{\mathbf{y}}^\top h](\mathbf{x}, \mathbf{y}) \\ &+ \mathbf{h}[f](\mathbf{x})[\nabla_{\mathbf{y}}^\top h](\mathbf{x}, \mathbf{y}) \\ &+ \mathbf{G}[h](\mathbf{x}, \mathbf{y}) \otimes \nabla[f](\mathbf{x}) \\ &+ \nabla[f](\mathbf{x}) \otimes \mathbf{G}[h](\mathbf{x}, \mathbf{y})]\ f(\mathbf{y}) \\ &+ \mathbf{h}_{\mathbf{x}}[f(\mathbf{x})h(\mathbf{x}, \mathbf{y})]\nabla_{\mathbf{y}}^\top\ f(\mathbf{y}) \\ [\mathbf{h}_{\mathbf{x}} \mathbf{h}_{\mathbf{y}}^\top k](\mathbf{x}, \mathbf{y}) =& [f(\mathbf{x})[\mathbf{h}_{\mathbf{x}} \mathbf{h}_{\mathbf{y}}^\top h](\mathbf{x}, \mathbf{y}) \\ &+ \mathbf{h}[f](\mathbf{x})[\mathbf{h}_{\mathbf{y}}^\top h](\mathbf{x}, \mathbf{y}) \\ &+ \mathbf{G}[h](\mathbf{x}, \mathbf{y}) \otimes \nabla[f](\mathbf{x})\nabla^\top[f](\mathbf{y}) \\ &+ \nabla[f](\mathbf{x})\nabla^\top[f](\mathbf{y}) \otimes \mathbf{G}[h](\mathbf{x}, \mathbf{y})]\ f(\mathbf{y}) \\ &+ \mathbf{h}_{\mathbf{x}}[f(\mathbf{x})h(\mathbf{x}, \mathbf{y})]\mathbf{h}_{\mathbf{y}}^\top\ f(\mathbf{y})\end{aligned}$$

$O(d^2)$ MVM with H

Again, we observe a structured representation of the Hessian-kernel elements which permit a multiply in $\mathcal{O}(d^2)$ operations.

Warping $k(\mathbf{x}, \mathbf{y}) = h(\mathbf{u}(\mathbf{x}), \mathbf{u}(\mathbf{y}))$,

$$\begin{aligned}\mathbf{h}_{\mathbf{x}} k(\mathbf{x}, \mathbf{y}) &= (\mathbf{J} \otimes \mathbf{J})^\top[\mathbf{u}](\mathbf{x})\ [\mathbf{h}_{\mathbf{x}} h](\mathbf{u}(\mathbf{x}), \mathbf{u}(\mathbf{y})) \\ [\mathbf{h}_{\mathbf{x}} \nabla_{\mathbf{y}}^\top k](\mathbf{x}, \mathbf{y}) &= (\mathbf{J} \otimes \mathbf{J})^\top[\mathbf{u}](\mathbf{x})\ [\mathbf{h}_{\mathbf{x}} \nabla_{\mathbf{y}}^\top h](\mathbf{u}(\mathbf{x}), \mathbf{u}(\mathbf{y}))\ \mathbf{J}[\mathbf{u}](\mathbf{y}) \\ [\mathbf{h}_{\mathbf{x}} \mathbf{h}_{\mathbf{y}}^\top k](\mathbf{x}, \mathbf{y}) &= (\mathbf{J} \otimes \mathbf{J})^\top[\mathbf{u}](\mathbf{x})\ [\mathbf{h}_{\mathbf{x}} \mathbf{h}_{\mathbf{y}}^\top h](\mathbf{u}(\mathbf{x}), \mathbf{u}(\mathbf{y}))\ (\mathbf{J} \otimes \mathbf{J})[\mathbf{u}](\mathbf{y}).\end{aligned}$$

We therefore see that $\mathbf{K}^{\mathbf{H}} = \mathbf{h}_{\mathbf{x}} \mathbf{h}_{\mathbf{y}}^\top k(\mathbf{X}) = \mathbf{D}_{\mathbf{J}}[\mathbf{h}_{\mathbf{x}} \mathbf{h}_{\mathbf{y}}^\top h](\mathbf{X})\mathbf{D}_{\mathbf{J}}$, where $\mathbf{D}_{\mathbf{J}}$ is the block-diagonal matrix whose i^{th} block is equal to $(\mathbf{J} \otimes \mathbf{J})[\mathbf{u}](\mathbf{x}_i) = \mathbf{J}[\mathbf{u}](\mathbf{x}_i) \otimes \mathbf{J}[\mathbf{u}](\mathbf{x}_i)$. Note that for linearly warped kernels for which $\mathbf{u}(\mathbf{x}) = \mathbf{U}\mathbf{x}$, where $\mathbf{U} \in \mathbb{R}^{r \times d}$, we have $(\mathbf{J} \otimes \mathbf{J})[\mathbf{u}](\mathbf{x}_i) = \mathbf{U} \otimes \mathbf{U}$ so that we can multiply with the kernel matrix $\mathbf{K}^{\mathbf{H}}$ in $\mathcal{O}(n^2 r^2 + n(d^2 r + r^2 d))$. The complexity is due to the following property of Kronecker product:

$$(\mathbf{U} \otimes \mathbf{U})\text{vec}(\mathbf{H}) = \text{vec}(\mathbf{U}\mathbf{H}\mathbf{U}^\top),$$

which can be computed in $\mathcal{O}(d^2 r + r^2 d)$ for every of the n Hessian observations.

Scalable First-Order Bayesian Optimization via Structured Automatic Differentiation, Ament and Gomes, *ICML 2022*

Performance Comparison to Prior Work

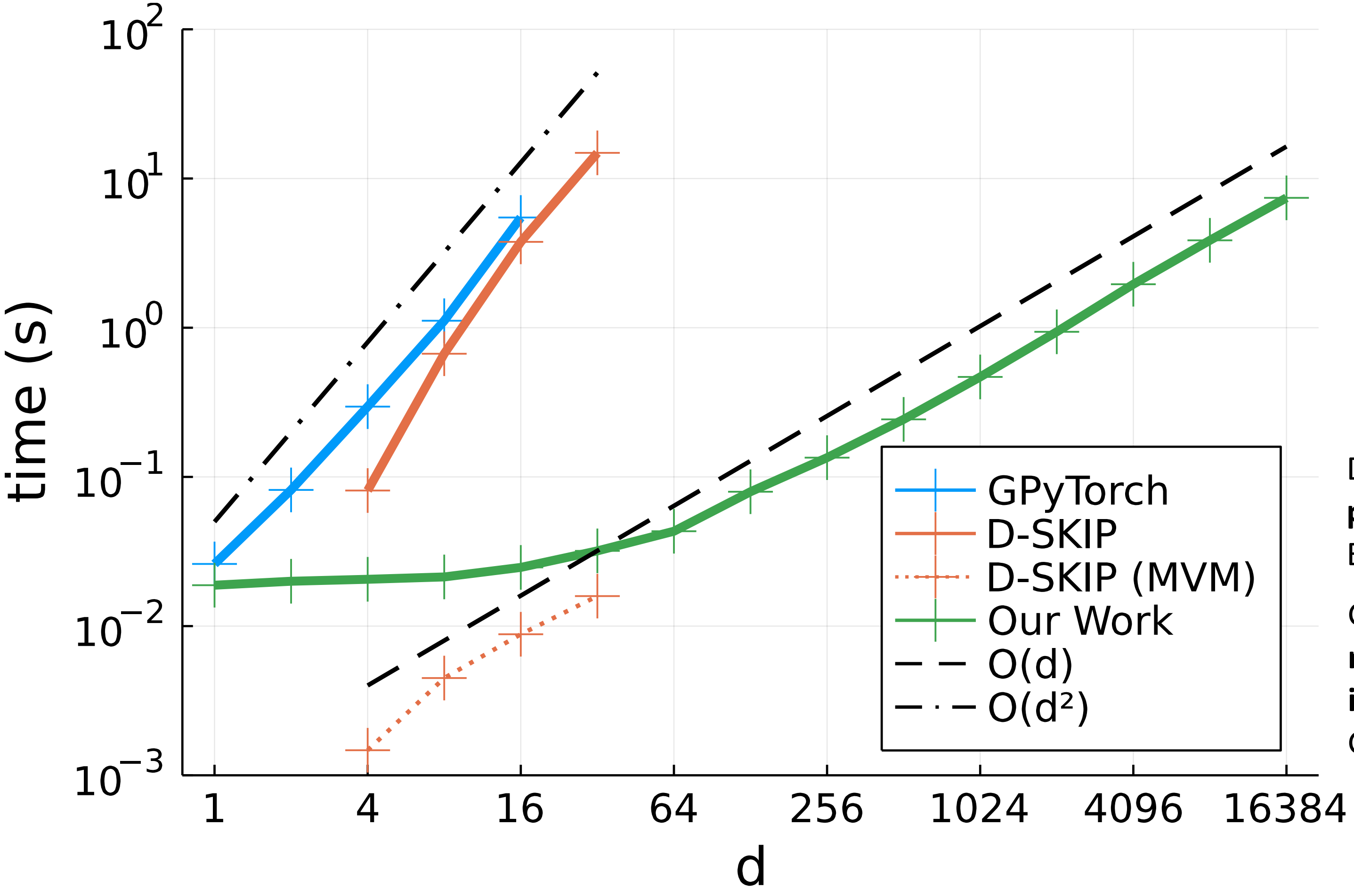
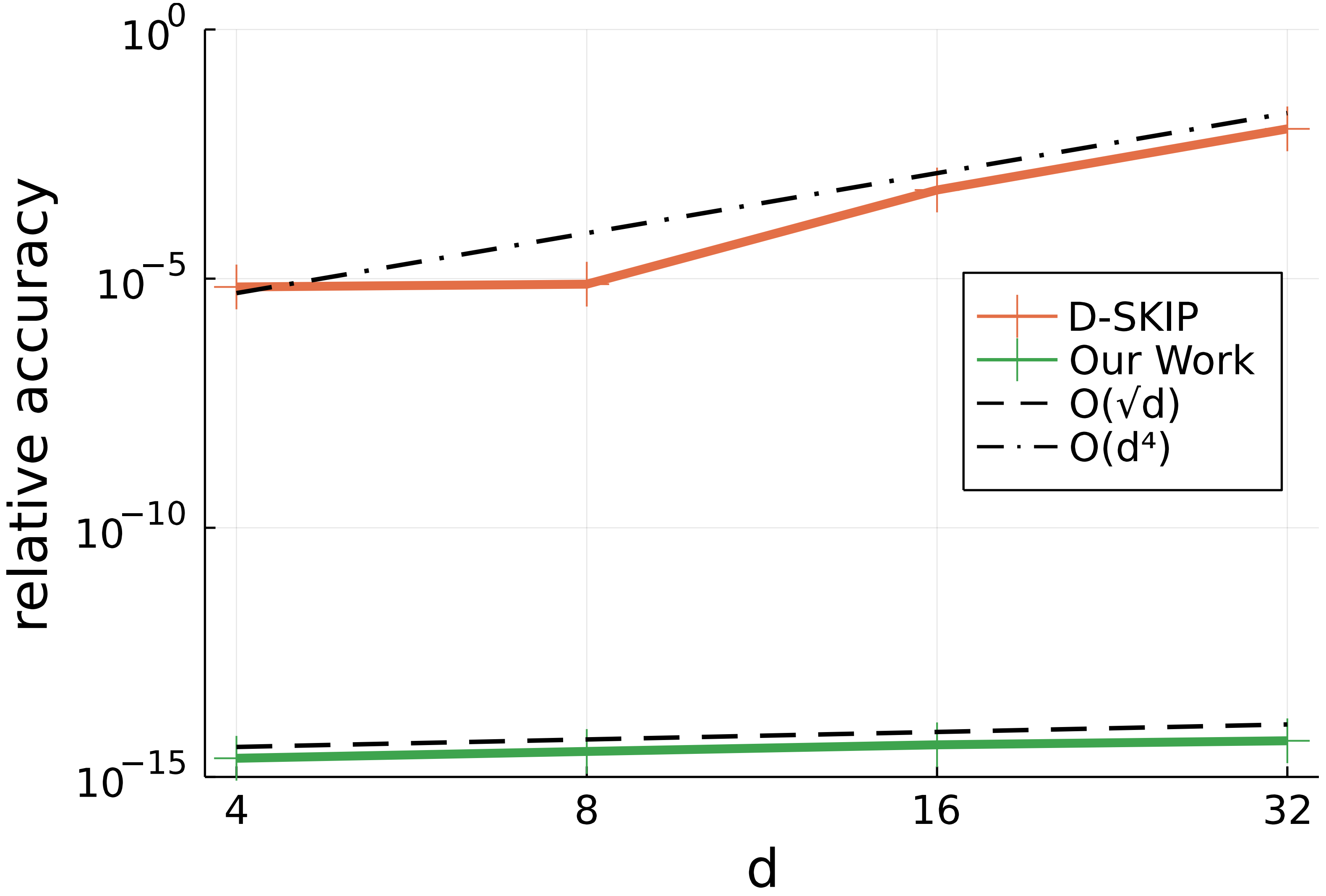


Figure 4: Time to first MVM of GPyTorch, D-SKIP, and our work for RBF gradient kernel matrices with $n = 1024$.

D-SKIP from “**Scaling Gaussian process regression with derivatives.**” Eriksson et al. *NeurIPS 2018*.

GPyTorch from "**Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration.**” Gardner et al. *NeurIPS 2018*.

Accuracy Comparison to Prior Work



D-SKIP from “**Scaling Gaussian process regression with derivatives.**” Eriksson et al. *NeurIPS 2018*.

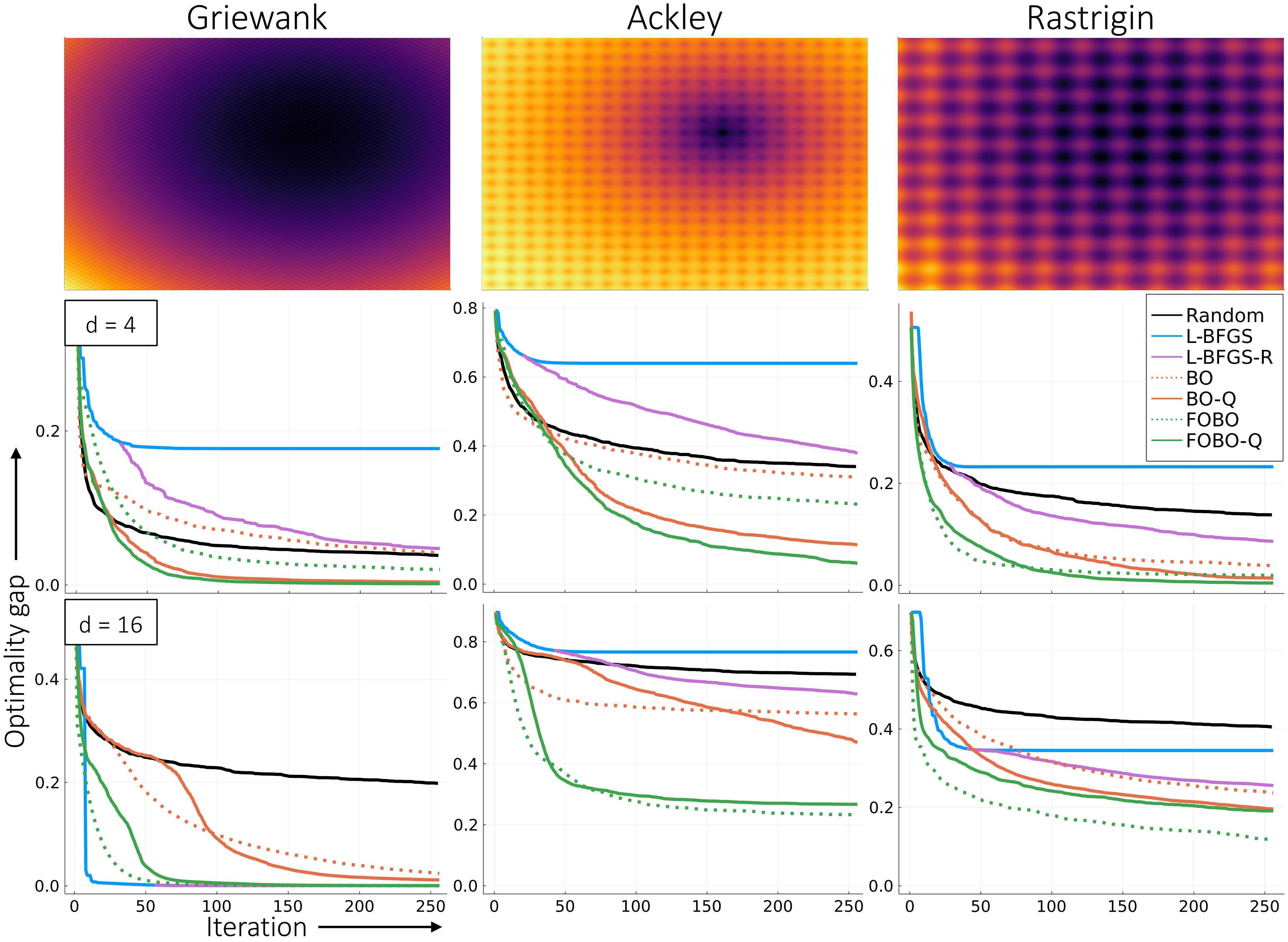
Bayesian Optimization Benchmarks

Comparing against

- Random sampling
- Convex optimization (L-BFGS)
- Convex optimization with restarts (L-BFGS-R)
- Bayesian Optimization (BO)

- BO with quadratic mixture kernel (BO-Q)
- First-order BO (FOBO)
- FOBO with quadratic mixture kernel (FOBO-Q)

} Proposed / scaled by our work

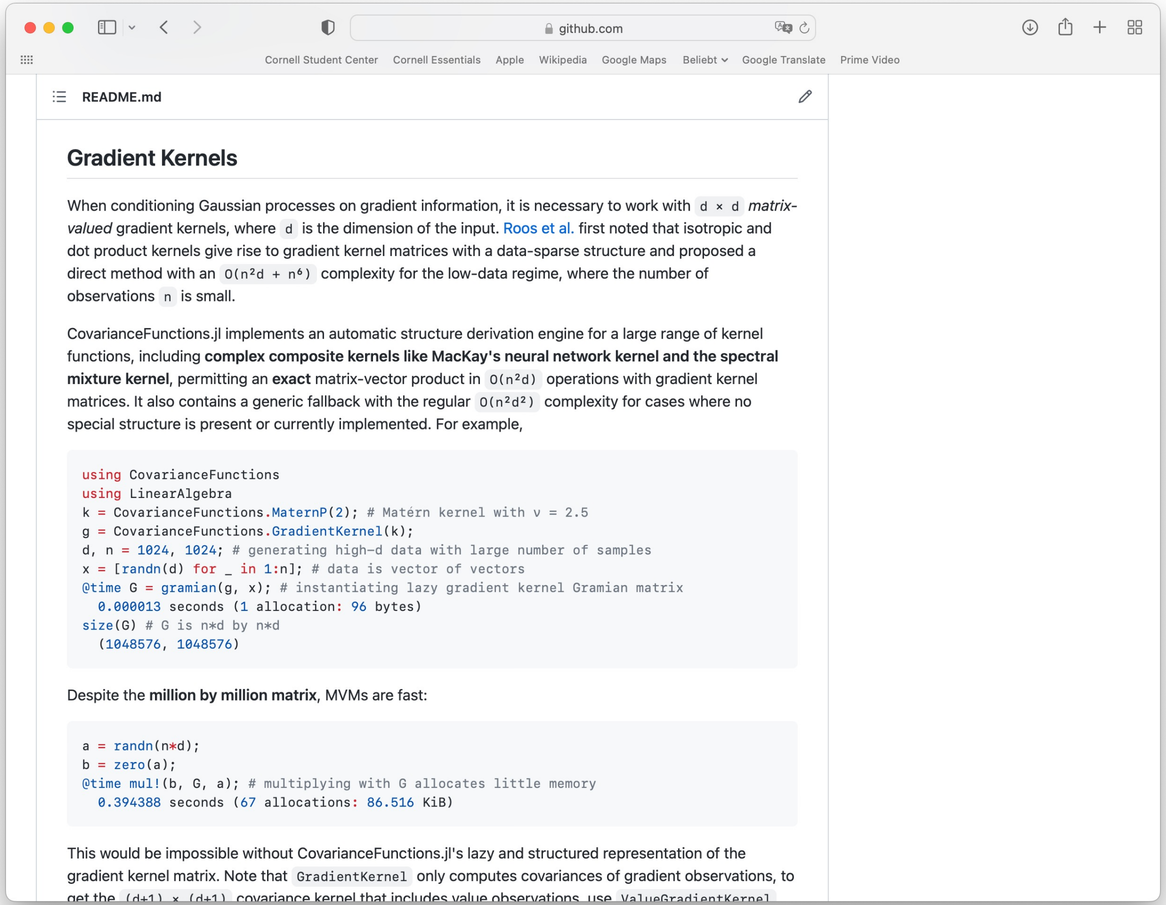


Griewank
Ackley
Rastrigin
d = 4
d = 16
Optimality gap
Iteration
Random
L-BFGS
L-BFGS-R
BO
BO-Q
FOBO
FOBO-Q
0.0
0.2
0.4
0.6
0.8
0
50
100
150
200
250

CovarianceFunctions.jl

Our methods are now available and open source at:

github.com/SebastianAment/CovarianceFunctions.jl



Gradient Kernels

When conditioning Gaussian processes on gradient information, it is necessary to work with `d × d` *matrix-valued* gradient kernels, where `d` is the dimension of the input. Roos et al. first noted that isotropic and dot product kernels give rise to gradient kernel matrices with a data-sparse structure and proposed a direct method with an `O(n²d + n⁶)` complexity for the low-data regime, where the number of observations `n` is small.

CovarianceFunctions.jl implements an automatic structure derivation engine for a large range of kernel functions, including **complex composite kernels like MacKay's neural network kernel and the spectral mixture kernel**, permitting an **exact** matrix-vector product in `O(n²d)` operations with gradient kernel matrices. It also contains a generic fallback with the regular `O(n²d²)` complexity for cases where no special structure is present or currently implemented. For example,

```
using CovarianceFunctions
using LinearAlgebra
k = CovarianceFunctions.MaternP(2); # Matérn kernel with ν = 2.5
g = CovarianceFunctions.GradientKernel(k);
d, n = 1024, 1024; # generating high-d data with large number of samples
x = [randn(d) for _ in 1:n]; # data is vector of vectors
@time G = gramian(g, x); # instantiating lazy gradient kernel Gramian matrix
  0.000013 seconds (1 allocation: 96 bytes)
size(G) # G is n*d by n*d
  (1048576, 1048576)
```

Despite the **million by million matrix**, MVMs are fast:

```
a = randn(n*d);
b = zero(a);
@time mul!(b, G, a); # multiplying with G allocates little memory
  0.394388 seconds (67 allocations: 86.516 KiB)
```

This would be impossible without CovarianceFunctions.jl's lazy and structured representation of the gradient kernel matrix. Note that `GradientKernel` only computes covariances of gradient observations, to get the `(d+1) × (d+1)` covariance kernel that includes value observations, use `ValueGradientKernel`

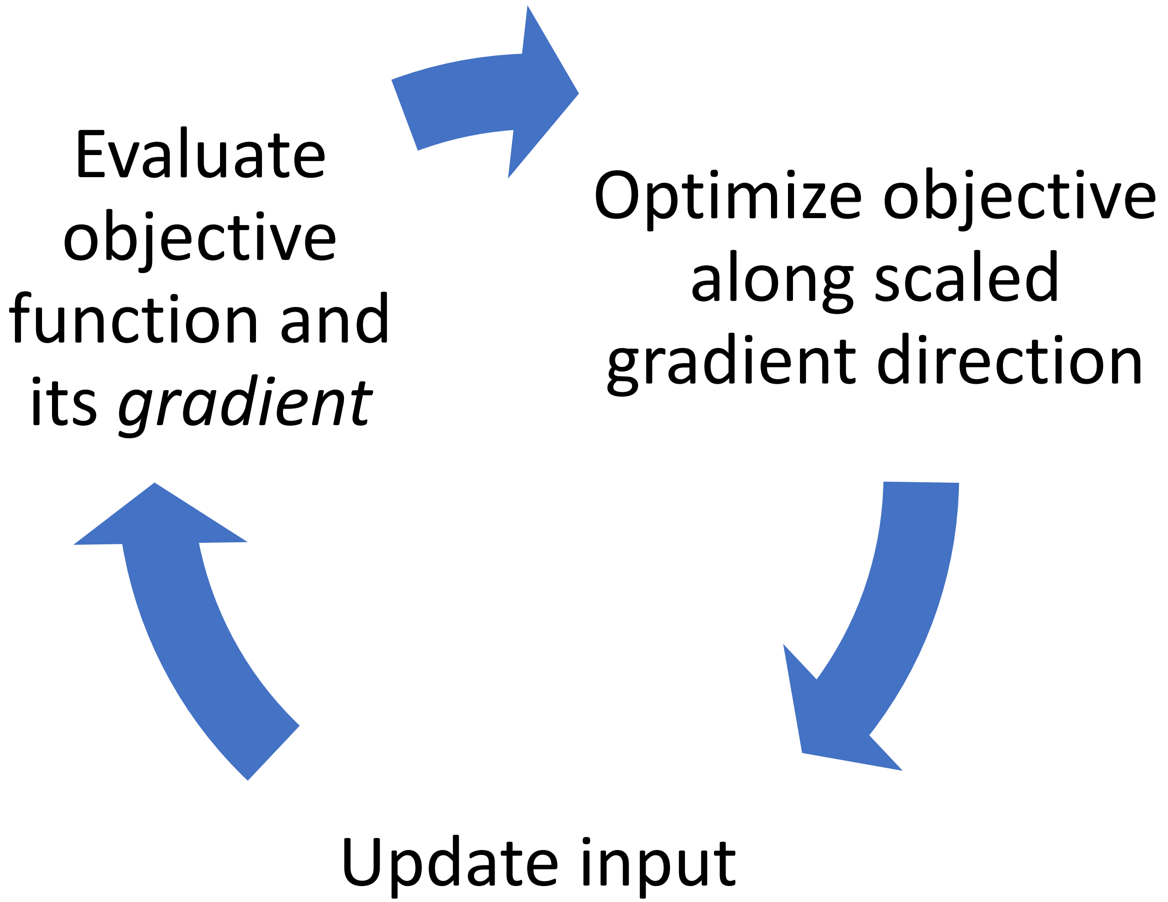
Thank you for listening!

Sebastian Ament and Carla Gomes

Cornell University

First-Order Optimization

locally optimizes a function by



Focus of Our Work

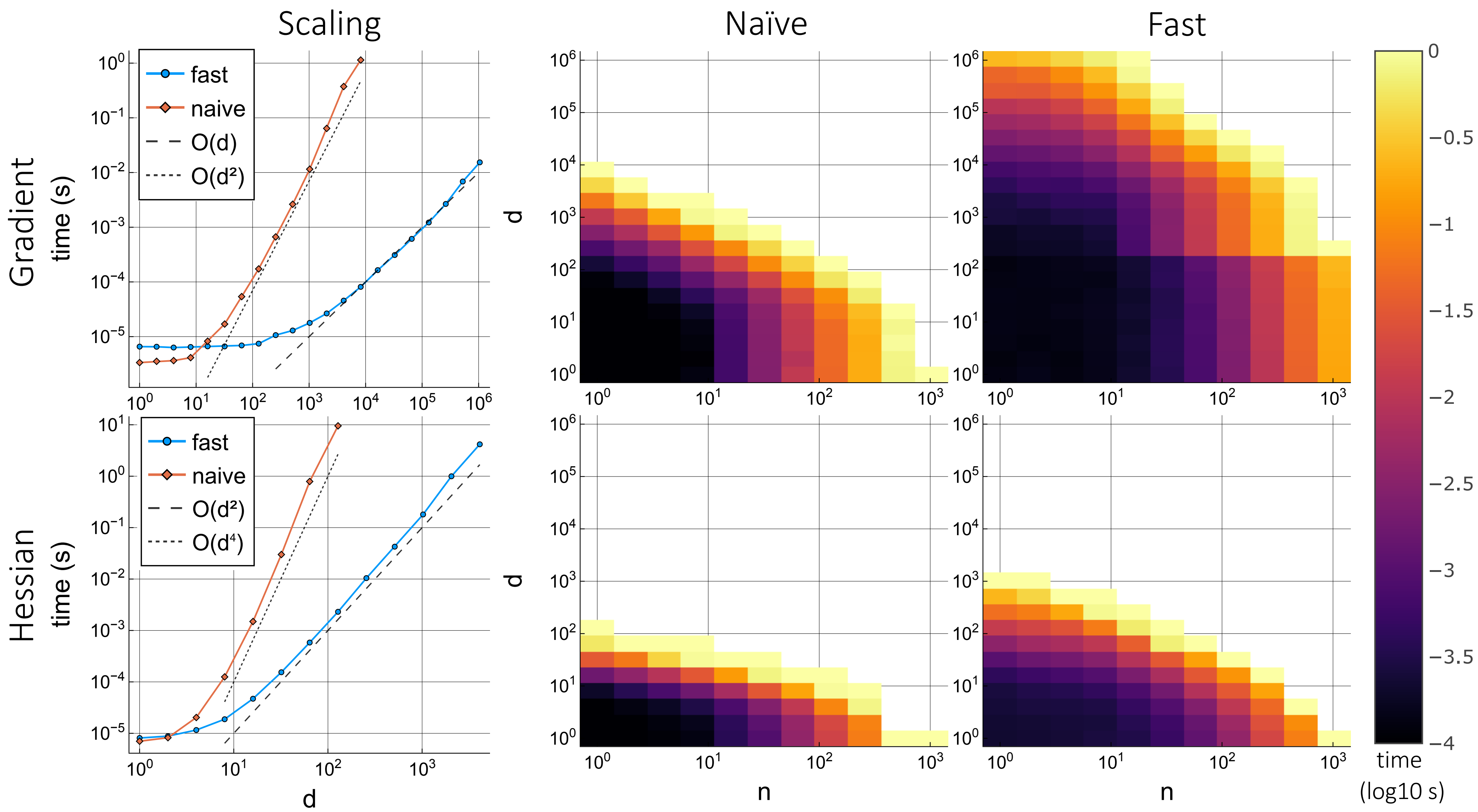
- Use iterative solvers based on $O(n^2 d)$ MVM
  - Does not have low-data restriction
  - Allows easy combining of value and derivative observations
- Increase scope of structured representations
  - Automatic derivation of structure for vast class of kernels
  - Structured Hessian kernel representations
- First-order Bayesian optimization

Combining Orders

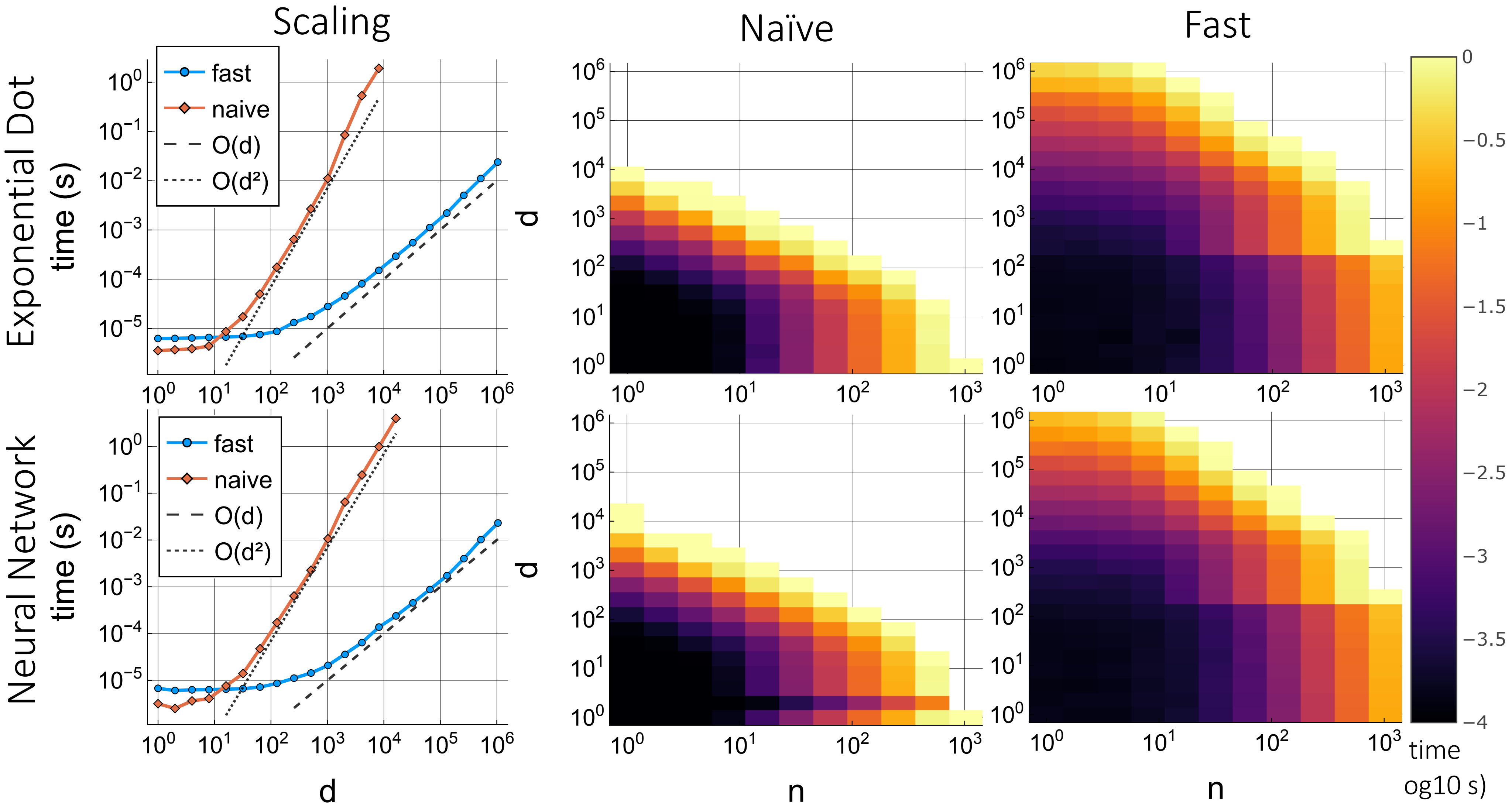
- We can combine value, gradient, and Hessian observations
- Include the relevant cross covariances

$$\begin{bmatrix} k & \nabla_{\mathbf{y}}[k]^\top & \mathbf{h}_{\mathbf{y}}[k]^\top \\ \nabla_{\mathbf{x}}[k] & \mathbf{G}[k] & \mathbf{J}_{\mathbf{x}}[\mathbf{h}_{\mathbf{y}}[k]] \\ \mathbf{h}_{\mathbf{x}}[k] & \mathbf{J}_{\mathbf{y}}[\mathbf{h}_{\mathbf{x}}[k]] & \mathbf{H}[k] \end{bmatrix}$$

Gradient and Hessian MVM Benchmarks



Composite Kernels MVM Benchmarks



Scope Comparison to Prior Work

Table 1: MVM complexity with select gradient kernel matrices. SM = spectral mixture kernel, NN = neural network kernel. *See the discussion on the right about D-SKIP's complexity.

	RBF	SM	NN
GPFlow / SKLearn	✗	✗	✗
GPyTorch	$\mathcal{O}(n^2d^2)$	✗	✗
(Eriksson et al., 2018)	$\mathcal{O}(nd^2)^*$	✗	✗
(De Roos et al., 2021)	$\mathcal{O}(n^2d)$	✗	✗
Our work	$\mathcal{O}(n^2d)$	$\mathcal{O}(n^2d)$	$\mathcal{O}(n^2d)$