Scalable kernel methods and their use in black-box optimization

David Eriksson

Center for Applied Mathematics
Cornell University

dme65@cornell.edu

November 9, 2018

Main projects

1. Scaling Gaussian process regression **[Today]**

   Collaborators: David Bindel, Kun Dong, Hannes Nickisch, Andrew Wilson

2. Scaling Gaussian process regression with derivatives **[Today]**

   Collaborators: David Bindel, Kun Dong, Eric Lee, Andrew Wilson

3. Energy bound optimization **[A-exam]**

   Collaborators: David Bindel

4. Asynchronous surrogate optimization **[A-exam]**

   Collaborators: David Bindel, Christine Shoemaker

5. Khatri-Rao systems of equations **[Another time]**

   Collaborators: Alex Townsend, Charles Van Loan

Outline

Section 1

Kernel methods

Scattered data interpolation

- Given:
  - Pairwise distinct points: $X = \{x_i\}_{i=1}^n \subset \Omega \subset \mathbb{R}^d$
  - Function values: $f_X = [f(x_1), \ldots, f(x_n)]^T$
- Goal: Find continuous function $s_{f,X}$ s.t.

$$s_{f,X}(x_i) = f(x_i), \qquad i = 1, \ldots, n$$

- Can use linear combination of continuous basis functions

$$s_{f,X}(x) = \sum_{i=1}^{n} \lambda_i b_i(x)$$

- Need to solve $A_X \lambda = f_X$, where $(A_X)_{ij} = b_j(x_i)$
- Well-posed if A_X is non-singular. When is this the case?

Basis functions

$(d = 1)$: Can choose basis functions independent of data

- Example: Polynomial interpolation with the monomial basis

$$\det A_X = \prod_{1 \leq i < j \leq n} (x_j - x_i) \neq 0$$

- Always non-singular if X are pairwise distinct

$(d \geq 2)$: Famous negative result:

- **Mairhuber-Curtis:** In order for $\det A_X \neq 0$ for all pairwise distinct $X \subset \Omega$, the basis functions must depend on X

Positive definite kernels

- Characterizing all data dependent basis functions challenging
- Common restriction: Require that A_X is always s.p.d.
- Achieved by using an s.p.d. kernel: $b_i(x) = k(x, x_i)$

Definition (Positive definite kernel)

A (continuous) symmetric function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is called a positive definite kernel if for all X, λ s.t.

1. The points in X are pairwise distinct,
2. $\lambda \neq 0$,

$$\implies \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j k(x_i, x_j) > 0.$$

Popular positive definite kernels

- White noise: $k(x, y) = \sigma^2 \delta_{xy}$
- Gaussian (SE): $k(x, y) = s^2 \exp\left(-\frac{\|x-y\|^2}{2\ell^2}\right)$
- Matérn $1/2$: $k(x, y) = s^2 \exp\left(-\frac{\|x-y\|}{\ell}\right)$
- Matérn $3/2$: $k(x, y) = s^2 \left(1 + \frac{\sqrt{3}\|x-y\|}{\ell}\right) \exp\left(-\frac{\sqrt{3}\|x-y\|}{\ell}\right)$
- Matérn $5/2$:
  $k(x, y) = s^2 \left(1 + \frac{\sqrt{5}\|x-y\|}{\ell} + \frac{5\|x-y\|^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}\|x-y\|}{\ell}\right)$
- Rational quadratic: $k(x, y) = s^2 \left(1 + \frac{\|x-y\|^2}{2\alpha\ell^2}\right)^{-\alpha}$

Polynomial precision

- Example: Gaussian kernel cannot reproduce $f(x) = \text{constant}$
- Desirable: $s_{f,X}$ exact for low-degree polynomials
- Often referred to as polynomial precision
- Mairhuber-Curtis $\implies$ Need additional assumptions on X

Definition

A set of points X are ν-unisolvent if the only polynomial of degree at most ν interpolating zero data on X is the zero polynomial.

Three collinear points in $\mathbb{R}^2$

The points $(0,0)$, $(1,1)$, $(2,2)$ are not 1-unisolvent

Kernel methods and polynomial precision

- Assume: The points X are ν-unisolvent
- $\{\pi_i\}_{i=1}^m$ basis for $p(x) \in \Pi_\nu^d$ (polynomials of degree $\leq \nu$)
- Look for

$$s_{f,X}(x) = \sum_{i=1}^{n} \lambda_i k(x, x_i) + \sum_{i=1}^{m} \mu_i \pi_i(x)$$

- We now have n equations and $m + n$ unknowns
- Add the m discrete orthogonality conditions:

$$\sum_{i=1}^{n} \lambda_j \pi_i(x_i) = 0, \quad j = 1, \ldots, m$$

- Allows us to use a larger family of kernels!

Kernel methods and polynomial precision

- Letting $(K_{XX})_{ij} = k(x_i, x_j)$ and $(P_X)_{ij} = \pi_j(x_i)$:

$$\begin{bmatrix} K_{XX} & P_X \\ P_X^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} f_X \\ 0 \end{bmatrix}$$

- Need: X $(\nu-1)$-unisolvent, $p \in \Pi^d_{\nu-1}$, k c.p.d of order ν

Definition (Conditionally positive definite kernel)

A (continuous) symmetric function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is called a conditionally positive definite kernel of order ν if for all X, λ s.t.

1. The points in X are pairwise distinct,
2. $\lambda \neq 0$ and $\sum_{i=1}^n \lambda_i q(x_i) = 0, \qquad \forall q \in \Pi^d_{\nu-1}$,

$$\implies \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j k(x_i, x_j) > 0.$$

Radial basis functions

- Important special case: $\varphi(r) = k(x, y)$ where $r = \|x - y\|$
- Cubic ($\varphi(r) = r^3$), Thin-plate spline ($\varphi(r) = r^2 \log r$)
- Semi-norm: $|s_{f,X}|^2 = \langle s, s \rangle = \lambda^T \Phi_{XX} \lambda$
- Native space: $|f|_{\mathcal{N}_\varphi} = \sup_{X \subset \Omega, |X| < \infty} |s_{f,X}|$
- Generic error estimate:

$$|f(x) - s_{f,X}(x)| \leq P_{\varphi,X}(x) \sqrt{|f|^2_{\mathcal{N}_\varphi} - |s_{f,X}|^2}$$

- Power function:

$$[P_{\varphi,X}(x)]^2 = \varphi(0) - \begin{bmatrix} \Phi_{Xx} \\ P_x^T \end{bmatrix}^T \begin{bmatrix} \Phi_{XX} & P_X \\ P_X^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} \Phi_{Xx} \\ P_x^T \end{bmatrix}.$$

- The power function is a Schur complement after adding x

Section 2

Scalable Gaussian processes

Gaussian processes interpolation

- Defines a distribution over functions:

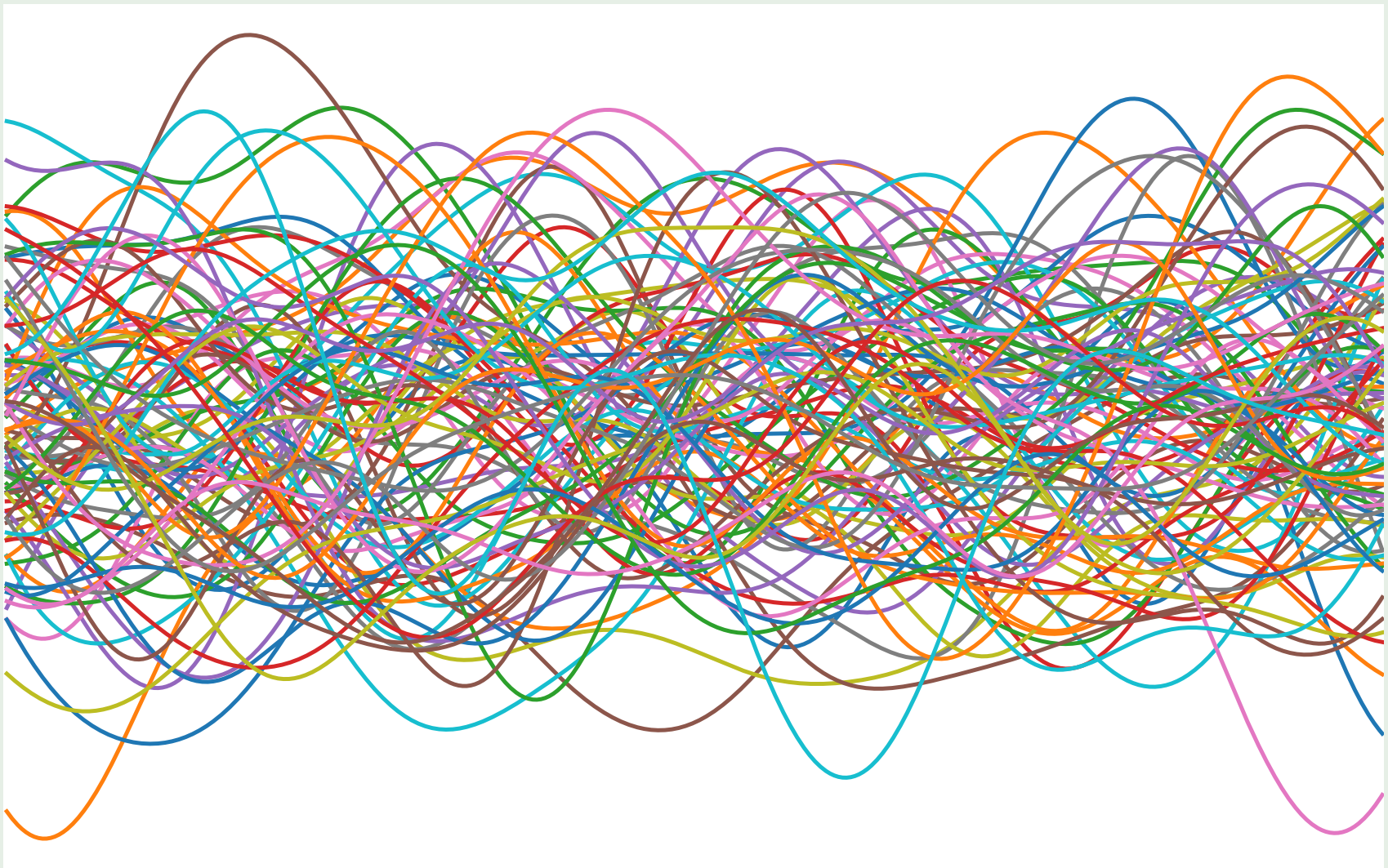
$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x'))$$

- Mean function: $\mu : \mathbb{R}^d \to \mathbb{R}$, often low-degree polynomial
- Covariance function: $\mathrm{cov}(f(x_i), f(x_j)) = k(x_i, x_j)$ s.p.d kernel
- Posterior mean and variance at x:

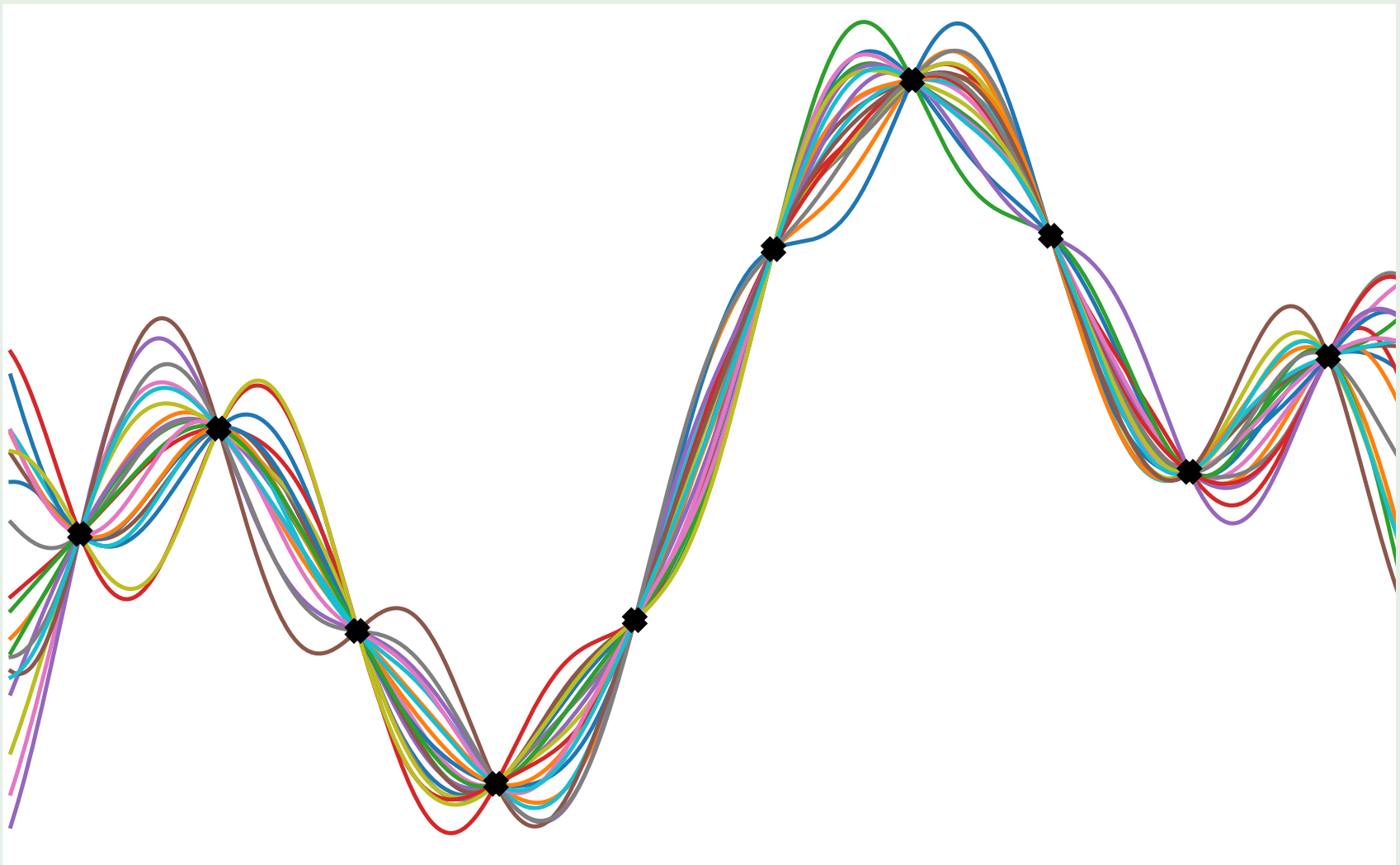
$$\mathbb{E}[f(x)] = K_{xX} K_{XX}^{-1} (y_X - \mu_X),$$
$$\mathbb{V}[f(x)] = K_{xx} - K_{xX} K_{XX}^{-1} K_{Xx},$$

- Compared to RBFs, $\mathbb{V}[f(x)]$ tells us about the average case

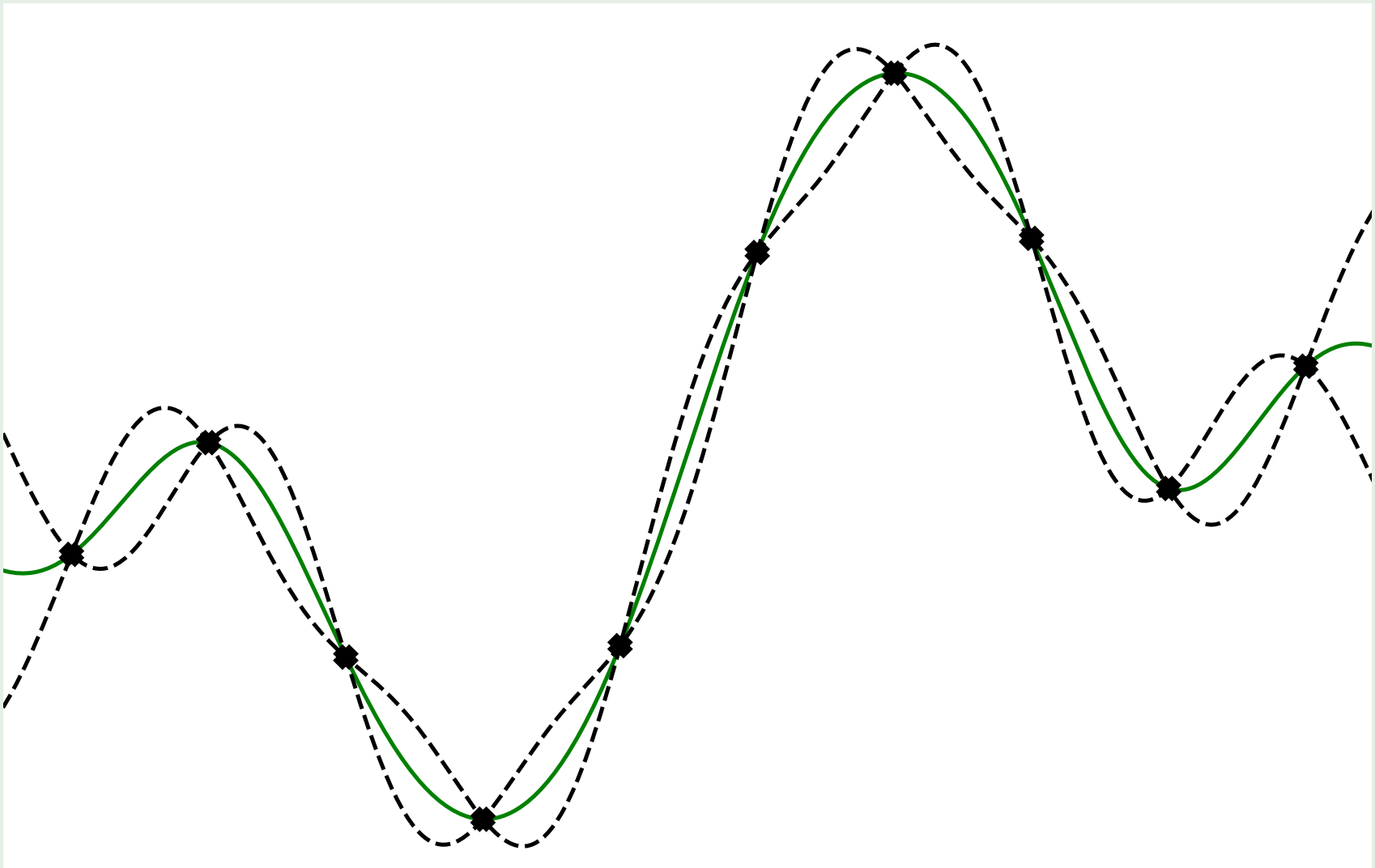
Draws from GP prior with zero mean



Draws from GP posterior



Posterior mean and variance



Gaussian processes regression

- Assume we observe $f_X \sim y_X + \epsilon,\ \epsilon \in \mathcal{N}(0, \sigma^2 I)$
- Add white noise kernel:

$$\tilde{k}(x, y) = k(x, y) + \sigma^2 \delta_{xy}$$

- We often do this even in the case of no noise
- Weyl: $\varphi(r) \in C^\nu \implies |\lambda_n| = o\left(n^{-\nu-1/2}\right)$
- Example: $|\lambda_n|$ decays exponentially for Gaussian (SE) kernel
- Adding $\sigma^2 \delta_{xy}$ guarantees $|\lambda_n| \geq \sigma^2$
- Gershgorin:

$$\kappa(\Phi_{XX} + \sigma^2 I) \leq \frac{n\,\varphi(0)}{\sigma^2}$$

- Example: $\kappa(\Phi_{XX} + \sigma^2 I) \leq n\left(\frac{s}{\sigma}\right)^2$ for Gaussian (SE) kernel

Kernel hyper-parameters

- How do we learn the optimal kernel hyperparameters θ?
- Bayesian approach is expensive, often do MLE
- Log marginal likelihood:

$$\log p(\theta \mid y_X) = \mathcal{L}_y + \mathcal{L}_{|K|} - \frac{n}{2} \log 2\pi$$

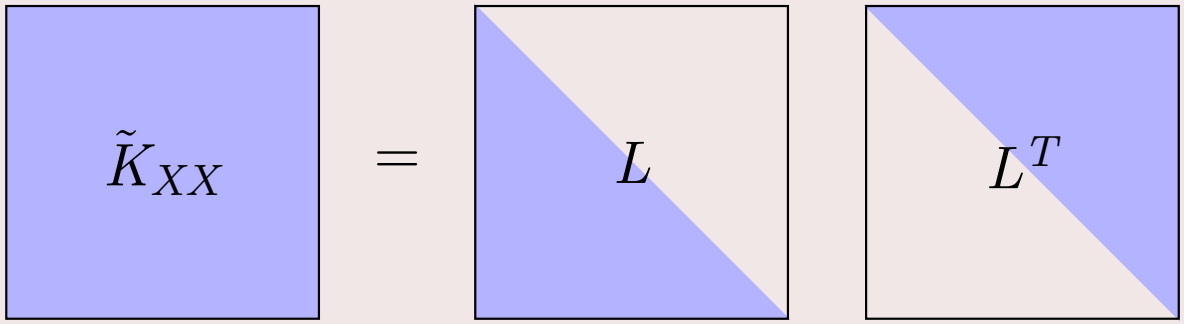
- Need to compute:

$$\mathcal{L}_y = -\frac{1}{2}(y_X - \mu_X)^T c, \qquad \frac{\partial \mathcal{L}_y}{\partial \theta_i} = \frac{1}{2} c^T \left(\frac{\partial \tilde{K}_{XX}}{\partial \theta_i} \right) c$$

$$\mathcal{L}_{|K|} = -\frac{1}{2} \log \det \tilde{K}_{XX}, \qquad \frac{\partial \mathcal{L}_{|K|}}{\partial \theta_i} = -\frac{1}{2} \operatorname{tr} \left(\tilde{K}_{XX}^{-1} \frac{\partial K_{XX}}{\partial \theta_i} \right)$$

where $c = \tilde{K}_{XX}^{-1}(y_X - \mu_X)$.

Exact kernel learning



- Compute dense Cholesky factorization: $\mathcal{O}(n^3)$ flops
- Solves and logdet computations with $\tilde{K}_{XX}$ are now trivial:

$$\tilde{K}_{XX} \backslash c = L^T \backslash (L \backslash c)$$

$$\log\det \tilde{K}_{XX} = 2 \sum_{i=1}^{n} \log L_{ii}$$

- Works for small n, but dense LA is not scalable!

Iterative methods

- **Assumption:** We have access to a fast MVM with $\tilde{K}_{XX}$
- Use a Krylov method to solve linear systems with $\tilde{K}_{XX}$

$$\mathcal{K}_k(A, b) = \text{span}\{b, Ab, \ldots, A^{k-1}b\}$$

- $\tilde{K}_{XX}$ is s.p.d $\implies$ use the conjugate gradient (CG) method
- Only interacts with $\tilde{K}_{XX}$ via MVMs
- Converges in n iterations in exact arithmetic
- A few iterations are enough for many kernels
- Small ℓ: K_{XX} almost diagonal $\implies$ fast convergence
- Large ℓ: Pivoted Cholesky preconditioner, $K_{XX} \approx P(LL^T)P^T$

Stochastic trace estimation

- How do we approximate $\log\det \tilde{K}_{XX}$ using fast MVMs?
- Note that $\log\det \tilde{K}_{XX} = \mathrm{tr}(\log \tilde{K}_{XX})$
- Estimate trace of a matrix $\implies$ Stochastic trace estimation
- If z has independent random entries, $\mathbb{E}[z_i] = 0$, $\mathbb{E}[z_i^2] = 1$:

$$\mathbb{E}[z^T A z] = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\,\mathbb{E}[z_i z_j] = \mathrm{tr}(A)$$

- Common choices of probe vector z:
  - Hutchinson: $z_i = \pm 1$ with probability 0.5
  - Gaussian: $z_i \sim \mathcal{N}(0,1)$
- This requires fast computation of $\log(\tilde{K}_{XX})z$:
- Function application with Hermitian matrix $\implies$ Lanczos

Lanczos

- Lanczos computes factorization: $\tilde{K}_{XX} Q = Q T$
  - Q orthogonal, T tridiagonal
- Elegant three term recursion with one MVM per iteration
- Converges in $k \leq p$ steps if $\tilde{K}_{XX}$ has p distinct eigenvalues
- Function application starting at $z/\|z\|$:

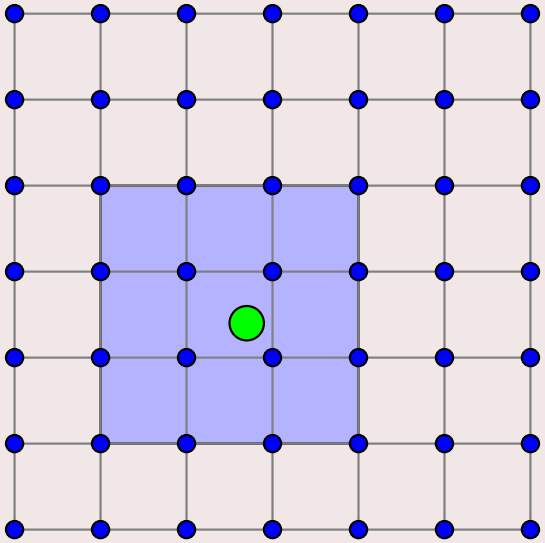
$$f(\tilde{K}_{XX}) z = Q f(T) Q^T z = \|z\| Q f(T) e_1$$

- Truncate after $k \ll n$ steps:

$$f(\tilde{K}_{XX}) z \approx \|z\| Q_k f(T_k) e_1$$

- N.B: CG is a special case of Lanczos

Fast MVMs: SKI



- Structured kernel interpolation (SKI):
  - $K_{XX} \approx W^T K_{UU} W$
  - U is a structured grid with m points
  - K_{UU} is BTTB (with Kronecker structure for product kernel)
  - W sparse matrix with interpolation weights
- Can apply MVM with K_{XX} in $\mathcal{O}(m \log m)$ time using FFT
- Grid structure limited to ≈ 5 dimensions

SKI for Product kernels (SKIP)

- Main idea: $(A \odot B)x = \mathrm{diag}(A\ \mathrm{diag}(x)\ B^T)$
- Cost for an MVM: $\mathcal{O}(nr^2)$ flops if A, B have rank r
- Assume tensor product structure: $k(x, y) = \prod_{i=1}^{d} k_i(x_i, y_i)$
- Many popular kernels (e.g., SE) have tensor product structure
- Use SKI in each dimension:

$$K \approx (W_1 K_1 W_1^T) \odot \ldots \odot (W_d K_d W_d^T)$$

- Divide and conquer + truncated rank-r Lanczos factorizations:

$$K \approx (Q_1 T_1 Q_1^T) \odot (Q_2 T_2 Q_2^T)$$

- Constructing SKIP kernel: $\mathcal{O}(n + m \log m + r^3 n \log d)$ flops
- Often achieve high accuracy for $r \ll n$

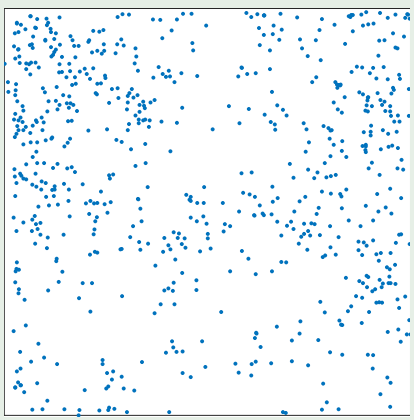
Rainfall

Method	n	m	MSE	Time [min]
Lanczos	528k	3M	0.613	14.3
Scaled eigenvalues	528k	3M	0.621	15.9
Exact	12k	-	0.903	11.8

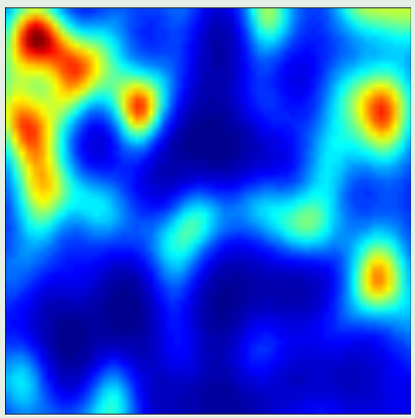
- Data: Hourly precipitation data at 5500 weather stations
- Aggregate into daily precipitation
- Total data: $628k$ entries
- Train on $528k$ data points, test on remainder
- Use SKI with 100 points per spatial dim, 300 in time
- Reference comparison: exact computation ($12k$ entries)

Hickory Data Set

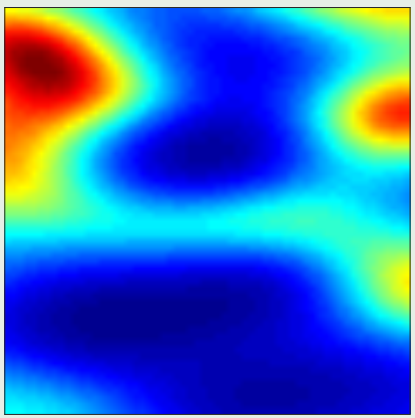
- Our approach can be used for non-Gaussian likelihoods
- Example: Log-Gaussian Cox process
  - Count data for Hickory trees in Michigan
  - Area discretized using a 60×60 grid
  - Use the Poisson likelihood with the SE kernel
  - Laplace approximation for posterior
- The scaled eigenvalue method uses the Fiedler bound



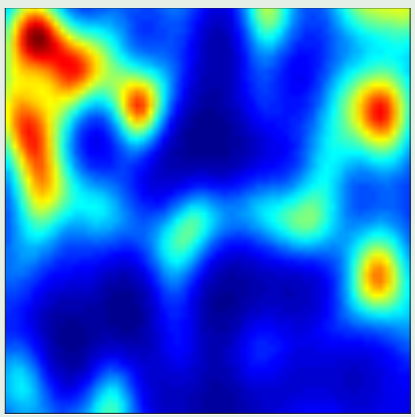
(a) Count data



(b) Exact



(c) Scaled eigs



(d) Lanczos

Section 3

Scalable Gaussian processes with derivatives

Gaussian process with derivatives

- Assume we observe both $f(x)$ and $\nabla f(x)$
- Let $f(x) \sim \mathcal{GP}(\mu(x), k(x, x'))$
- Differentiation is a linear operator:

$$\mu^{\nabla}(x) = \begin{bmatrix} \mu(x) \\ \nabla\mu(x) \end{bmatrix}, \quad k^{\nabla}(x, x') = \begin{bmatrix} k(x, x') & (\nabla_{x'} k(x, x'))^T \\ \nabla_x k(x, x') & \nabla^2 k(x, x') \end{bmatrix}$$

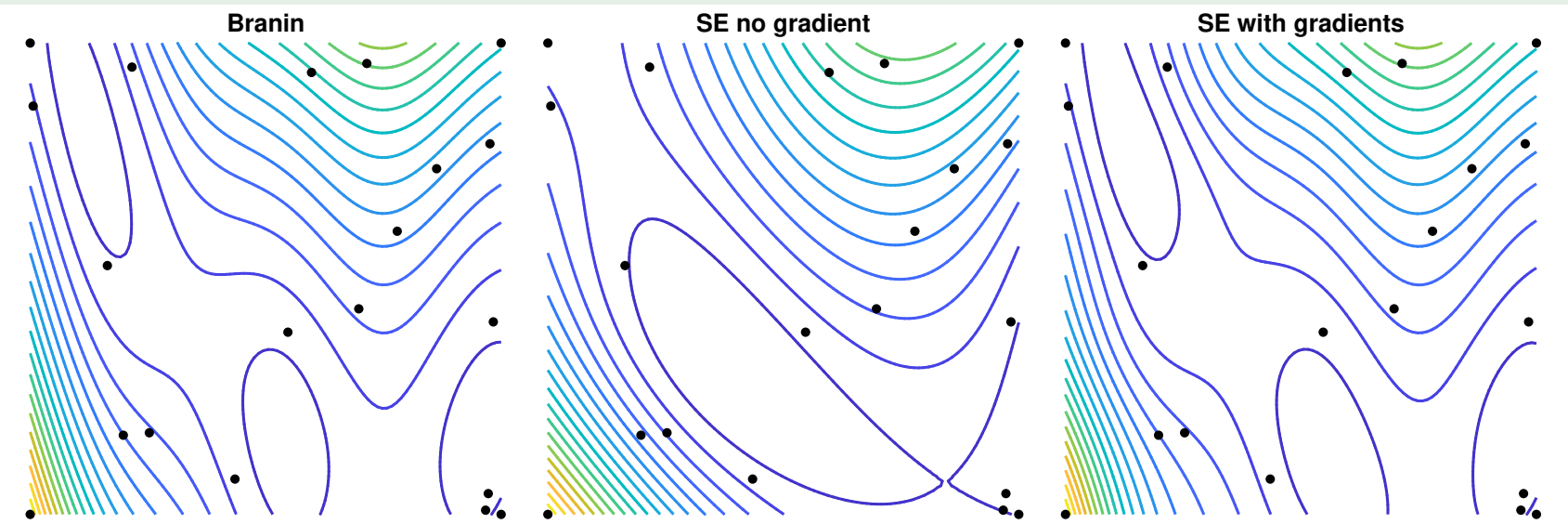
- Multi-output GP:

$$\begin{bmatrix} f(x) \\ \nabla f(x) \end{bmatrix} \sim \mathcal{GP}\left(\mu^{\nabla}(x), k^{\nabla}(x, x')\right)$$

- Exact kernel learning and inference is now $\mathcal{O}(n^3 d^3)$ flops
- Involves kernel matrix of size $n(d+1) \times n(d+1)$

Example: Branin function

- Gradient information can make the GP model more accurate
- (Left) True function
- (Middle) GP without derivatives
- (Right) GP with derivatives



Extending SKI and SKIP

- Differentiate the approximation scheme
- D-SKI: $k(x, x') \approx \sum_i w_i(x) k(u_i, x') \rightarrow \nabla k(x, x') \approx \sum_i \nabla w_i(x) k(u_i, x')$
- D-SKIP: Differentiate each Hadamard product

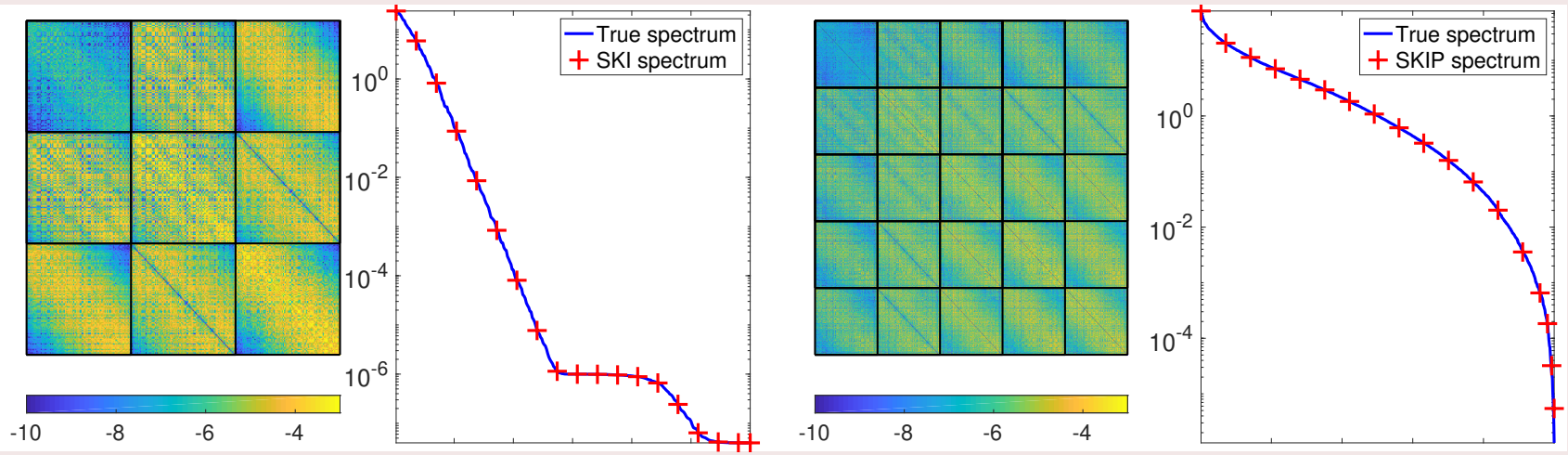


Figure: (Left) $\log_{10}$ error in D-SKI approximation and comparison to the exact spectrum. (Right) $\log_{10}$ error in D-SKIP approximation and comparison to the exact spectrum.

Active subspaces

- Can estimate active subspace from gradients:

$$C = \int_\Omega \nabla f(x) \nabla f(x)^T \, dx \approx Q \Lambda Q^T$$

- λ_i measures the average change in f along q_i
- Optimal $\tilde{d}$-dimensional subspace P: First $\tilde{d}$ columns of Q
- Active subspace approximation: $f(x) \approx f(PP^T x)$
- Can work with kernel $\tilde{k}(x, x') = k(P^T x, P^T x')$
- We estimate C using Monte Carlo integration:

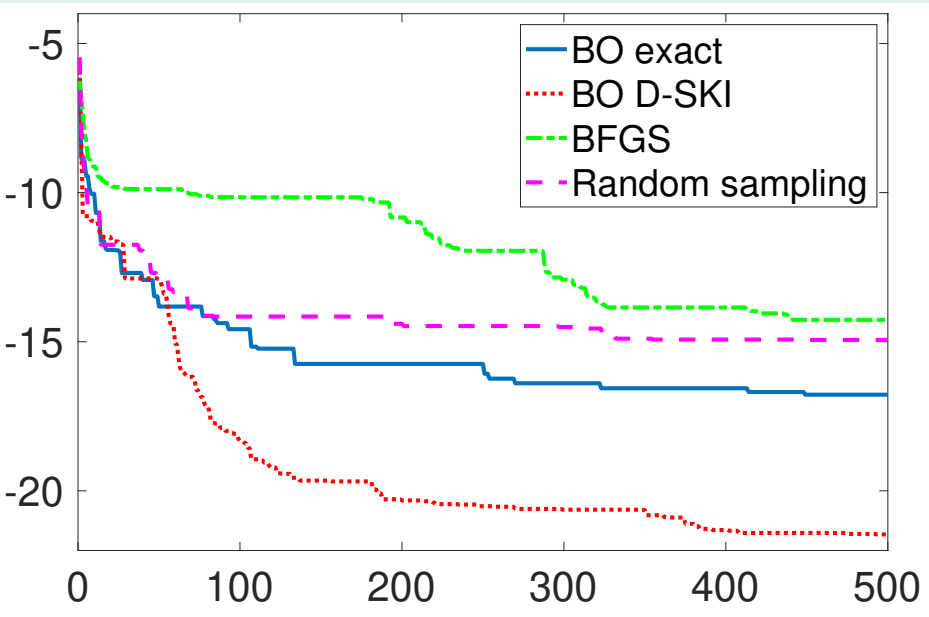
$$C \approx \frac{1}{n} \sum_{i=1}^{n} \nabla f(x_i) \nabla f(x_i)^T$$

Bayesian optimization with active subspace learning

1: Generate experimental design
2: Evaluate experimental design
3: **while** Budget not exhausted **do**
4: Calculate active subspace P using sampled gradients
5: Fit GP with derivatives using $k(P^T x, P^T x')$
6: Optimize $u_{n+1} = \arg\max \mathcal{A}(u)$ with $x_{n+1} = P u_{n+1}$
7: Sample point x_{n+1}, value f_{n+1}, and gradient ∇f_{n+1}
8: Update data $\mathcal{D}_{i+1} = \mathcal{D}_i \cup \{x_{n+1}, f_{n+1}, \nabla f_{n+1}\}$
9: **end**

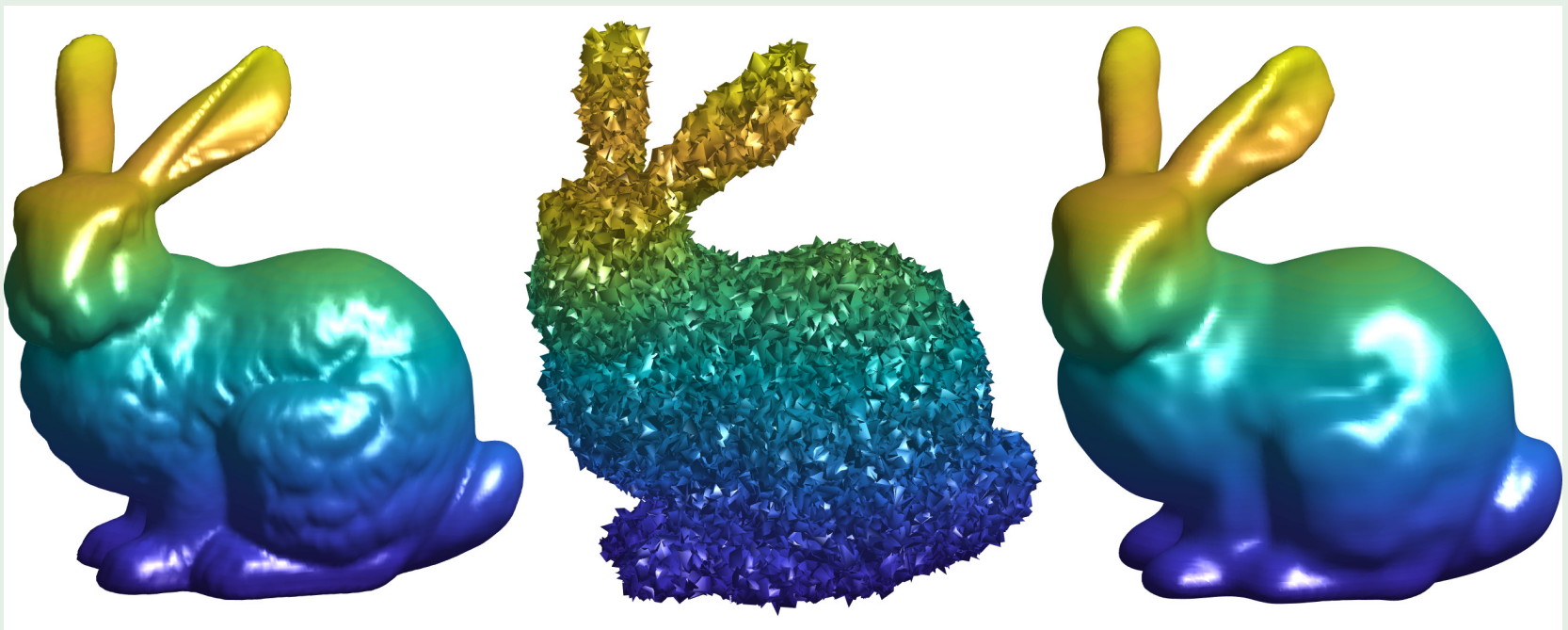
Bayesian optimization with EI

- 5-dimensional Ackley randomly embedded in 50 dimensions
- Observe noisy values and noisy gradients
- Use active subspace learning from sampled gradients
- Use D-SKI in the active subspace for fast kernel learning
- Active subspace learning improves the performance of BO



Stanford bunny

- Recovering the Stanford bunny from 25k noisy normals
- Spline kernel: $k(x, y) = s^2(\|x - y\|^3 + a\|x - y\|^2 + b)$
- Fit an implicit GP surface: $f(x_i) = 0$, $\nabla f(x_i) = n_i$



Section 4

Questions

Thank you for your attention!

Questions?