Scaling Gaussian Process Regression with Derivatives

Kun Dong[1] David Eriksson[1] Eric Hans Lee[2] David Bindel[2] Andrew Gordon Wilson[3]

Applied Math[1], CS[2], ORIE[3]

Cornell University

Gaussian Processes (GPs)

- Define rich distributions over functions with applications in most predictive task.
- $f(x) \sim GP(\mu(x), k(x,x'))$ with mean $\mu : \mathbb{R}^d \to \mathbb{R}$ and covariance $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$.
- Observe $y_i \sim N(f(x_i), \sigma_i^2)$

$\forall X = (x_1, \ldots, x_n),\ x_i \in \mathbb{R}^d$:

$f_X \sim N(\mu_X, K_{XX})$	where
$f_X \in \mathbb{R}^n$;	$(f_X)_i = f(x_i)$
$y_X \in \mathbb{R}^n$;	$(y_X)_i = y_i$
$\mu_X \in \mathbb{R}^n$;	$(\mu_X)_i = \mu(x_i)$
$K_{XX} \in \mathbb{R}^{n \times n}$;	$(K_{XX})_{ij} = k(x_i, x_j)$

Write K_{XX} as K when unambiguous.

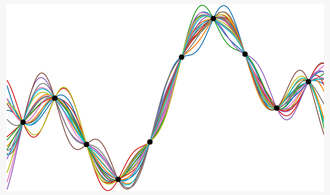


Figure: GP posterior

GP Regression with Derivatives

- Model function values and derivatives by a multi-output GP:

$$\begin{bmatrix} f_X \\ \nabla f_X \end{bmatrix} \sim N(\mu_X^\nabla, K_{XX}^\nabla), \quad \mu^\nabla(x) = \begin{bmatrix} \mu(x) \\ \nabla\mu(x) \end{bmatrix}, \quad k^\nabla(x,x') = \begin{bmatrix} k(x,x') & \nabla_{x'}k(x,x')^T \\ \nabla_x k(x,x') & \nabla^2_{xx'}k(x,x') \end{bmatrix}.$$

- Results in a larger kernel matrix $K_{XX}^\nabla \in \mathbb{R}^{n(d+1)\times n(d+1)}$.
- **GP with gradient information significantly improves the accuracy:**

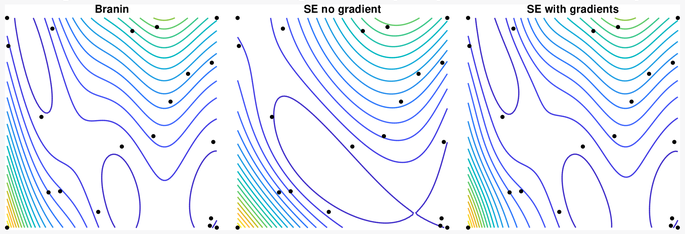


Figure: Branin function approximated by GPs with SE kernel.

Kernel Learning with Derivatives

- Train kernel hyperparameters θ by maximizing the log marginal likelihood:

$$\mathscr{L}(y^\nabla|\theta) = \mathscr{L}_{y^\nabla} + \mathscr{L}_{|K^\nabla|} - \frac{n(d+1)}{2}\log(2\pi),$$

where, if $\widetilde{K}^\nabla = K^\nabla + \Sigma$, Σ diagonal noise, $\widetilde{K}^\nabla c = (y^\nabla - \mu_X^\nabla)$,

$$\mathscr{L}_{y^\nabla} = -\frac{1}{2}(y^\nabla - \mu_X^\nabla)^T c, \qquad \frac{\partial \mathscr{L}_{y^\nabla}}{\partial \theta_i} = \frac{1}{2}c^T\left(\frac{\partial \widetilde{K}^\nabla}{\partial \theta_i}\right)c,$$

$$\mathscr{L}_{|K^\nabla|} = -\frac{1}{2}\log\det\widetilde{K}^\nabla, \qquad \frac{\partial \mathscr{L}_{|K^\nabla|}}{\partial \theta_i} = -\frac{1}{2}\mathrm{tr}\left(\widetilde{K}^\nabla \backslash \frac{\partial \widetilde{K}^\nabla}{\partial \theta_i}\right).$$

- Naive approach: Compute Cholesky factorization of $\widetilde{K}^\nabla$.
- Challenges:
  - **Direct methods lead to $O(n^3d^3)$ training and $O(nd)$ prediction.**
  - K^∇ is far more ill-conditioned than K.

Main Ideas

- **Apply structured kernel interpolation to enable fast MVMs for K^∇.**
- **Estimate $\mathscr{L}(y^\nabla|\theta)$ using iterative methods and stochastic estimators.**
- **Use a pivoted Cholesky preconditioner to accelerate convergence.**

SKI and SKIP: Structured Kernel Interpolation (with Product Kernels)

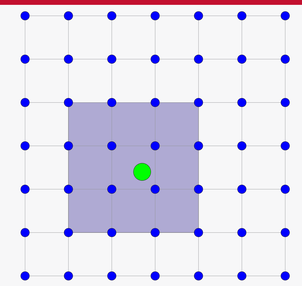


Figure: SKI (Wilson & Nickisch, ICML)

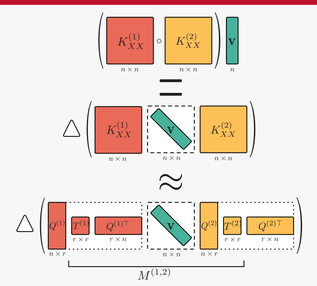


Figure: SKIP (Gardner et al., AISTATS)

D-SKI: Structured Kernel Interpolation with Derivatives

- SKI: Local polynomial interpolation on an induced grid with sparse weights.
- Differentiate interpolative weights to get a fast MVM with K^∇.

$$k(x,x') \approx \sum_i w_i(x)k(x_i,x') \to \nabla k(x,x') \approx \sum_i \nabla w_i(x) k(x_i,x').$$

$$K^\nabla \approx W^\nabla K_{UU}[W^\nabla]^T = \begin{bmatrix} W \\ \partial W \end{bmatrix} K_{UU} \begin{bmatrix} W \\ \partial W \end{bmatrix}^T.$$

- Use quintic interpolation of $k(x,x')$ for accurate interpolation of $\nabla^2_{xx'}k(x,x')$.
- MVM complexity is $O(nd6^d + m\log m)$ with m grid points.

D-SKIP: Structure Kernel Interpolation for Products with Derivatives

- SKIP: Apply D-SKI to each dimension for a product kernel:

$$K^\nabla \approx (W_1^\nabla K_1[W_1^\nabla]^T) \odot (W_2^\nabla K_2[W_2^\nabla]^T) \odot \ldots \odot (W_d^\nabla K_d[W_d^\nabla]^T),$$

- Main idea: $(A \odot B)v = \mathrm{diag}(A\,\mathrm{diag}(v)\,B^T)$ costs $O(nr^2)$ flops if A, B are rank-r.
- Divide and conquer + Lanczos to combine dimensions until:

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

- For an effective kernel rank of r at each step,
  - Construction cost: $O(d^2(n + m\log m + r^3 n \log d))$.
  - MVM cost: $O(dr^2 n)$.

D-SKI and D-SKIP Accurately Approximate the True Spectrum

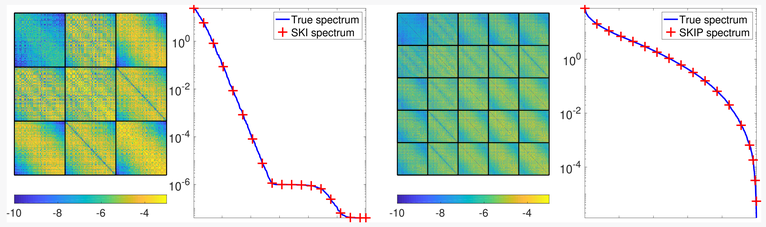


Figure: Accuracy on SE kernel (Left) D-SKI in 2 dimensions (Right) D-SKIP in 4 dimensions

Preconditioning

- (Partial) pivoted Cholesky factorization $K^\nabla \approx FF^T$ for preconditioning.
- Sherman-Morrison-Woodbury formula for $(\sigma^2 I + FF^T)^{-1}$ is unstable in practice.
- Use the stable formulation $(\sigma^2 I + FF^T)^{-1} f = \sigma^{-2}(f - Q_1(Q_1^T f))$ where

$$\begin{bmatrix} F \\ \sigma I \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R.$$

- Crucial for fast convergence of iterative solvers on K^∇.

Dimensionality Reduction via Active Subspace Learning

- Many high-dimensional problems have low-dimensional structure.
- Gradients allow us to uncover low-dimensional structure.
- λ_i is the average change in f given a perturbation along q_i:

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

- *Active subspace*: Leading $\tilde{d}$ eigenvectors describe most of the change in f.
- Fit GP with gradient information in the active subspace.

Recover Implicit Surface with D-SKI

- Noisy Stanford bunny: 25K points and noisy normals.
- Fit an implicit GP surface: $f(x_i) = 0$, $\nabla f(x_i) = n_i$

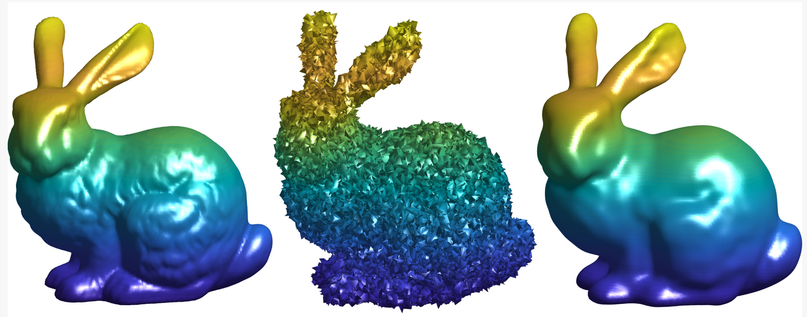
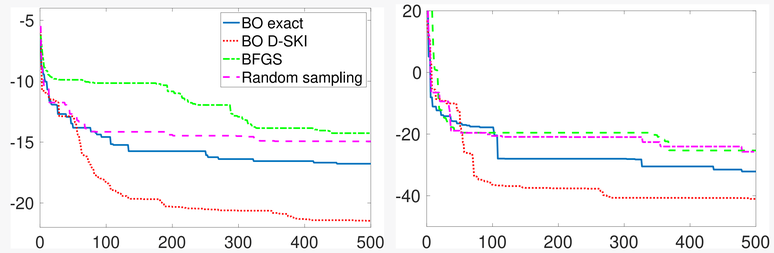


Figure: (Left) Original surface (Middle) Noisy surface (Right) D-SKI reconstruction from noisy surface

Bayesian Optimization with Derivatives and Active Subspace Learning

while Budget not exhausted **do**
  Calculate active subspace projection $P \in \mathbb{R}^{d\times\tilde{d}}$ using sampled gradients
  Fit GP with gradient information defined by kernel $k^\nabla(P^Tx, P^Tx')$
  Optimize acquisition function, $u_{n+1} = \arg\max \mathscr{A}(u)$ with $x_{n+1} = Pu_{n+1}$
  Sample point x_{n+1}, value f_{n+1}, and gradient ∇f_{n+1}
  Update data $\mathscr{D}_{i+1} = \mathscr{D}_i \cup \{x_{n+1}, f_{n+1}, \nabla f_{n+1}\}$

- We test on 5D Ackley embedded in $[-10,15]^{50}$ and 5D Rastrigin in $[-4,5]^{50}$.
- We apply D-SKI in a random 2D subspace of the estimated active subspace.



(a) BO on Ackley (b) BO on Rastrigin

Discussion

- Gradient information is valuable for GP regression, but scalability is a problem.
- Our approach: Fast MVMs, iterative methods, and dimensionality reduction.
- We achieve large-scale Bayesian optimization with derivatives.
- Implementation available at:

`https://github.com/ericlee0803/GP_Derivatives`.