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)

- Multivariate normals are distributions over vectors.
- Gaussian processes are distributions over functions with a mean field $\mu : \mathbb{R}^d \to \mathbb{R}$ and a covariance kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$.
- $f \sim GP(\mu, k)$ means for any $X = (x_1, \ldots, x_n)$, $x_i \in \mathbb{R}^d$:

$f_X \sim N(\mu_X, K_{XX})$ where

$f_X \in \mathbb{R}^n$; $(f_X)_i = f(x_i)$

$\mu_X \in \mathbb{R}^n$; $(\mu_X)_i = \mu(x_i)$

$K_{XX} \in \mathbb{R}^{n \times n}$; $(K_{XX})_{ij} = k(x_i, x_j)$

We write K_{XX} as K when unambiguous.

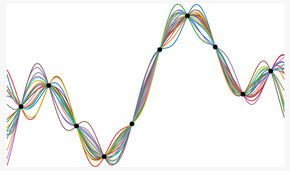


Figure: GP posterior

GP Regression with Derivatives

- Function values and derivatives can be modeled by a multi-output GP:

$$\begin{bmatrix} f_X \\ \partial_X f_X \end{bmatrix} \sim N(\mu^\nabla_X, K^\nabla_{XX}), \quad \mu^\nabla(x) = \begin{bmatrix} \mu(x) \\ \partial_x \mu(x) \end{bmatrix}, \quad k^\nabla(x, x') = \begin{bmatrix} k(x,x') & \partial_{x'} k(x,x')^T \\ \partial_x k(x,x') & \partial^2_x k(x,x') \end{bmatrix}.$$

- With derivatives, we get a larger kernel matrix $K^\nabla_{XX} \in \mathbb{R}^{n(d+1) \times n(d+1)}$.
- Gradient information significantly improves regression accuracy.

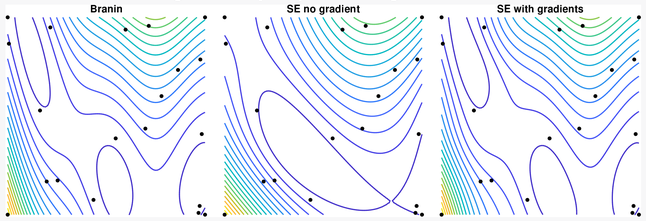


Figure: Branin function approximated by GPs with SE kernel.

Kernel Learning with Derivatives

- Kernel depends on hyperparameters θ, which are trained by maximizing the log-likelihood:

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

where (assuming $K^\nabla c = (y^\nabla - \mu^\nabla_X)$)

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

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

- Naive approach computes the Cholesky factorization of K^∇.
- Challenges:
  - Direct methods lead to $O(n^3 d^3)$ 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^∇.**
- **Combine iterative methods and stochastic estimators to avoid direct computation, with preconditioning to improve convergence.**
- **Use active subspace learning to overcome curse of dimensionality.**

D-SKI: Structured Kernel Interpolation with Derivatives

- Differentiate the interpolation weights to guarantee positive-semidefiniteness

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

- Use local quintic interpolation to get a sparse weight matrix and grid-structured kernel

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

- Due to sparsity of $W, \partial W$ and fast MVMs with K_{UU} via FFTs, overall MVM complexity is $O(nd6^d + m \log m)$.

D-SKIP: Structure Kernel Interpolation for Products with Derivatives

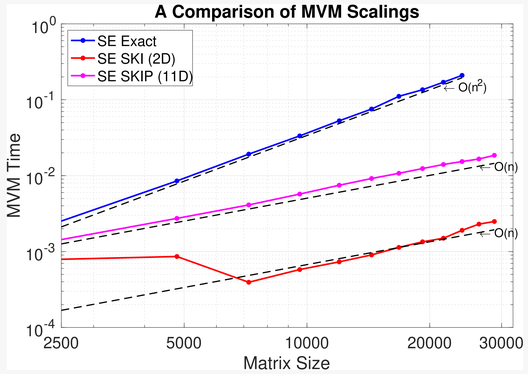
- SKIP yields the following approximations for separable kernels:

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

- We differentiate and iteratively apply the Lanczos decomposition to obtain:

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

- 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: $O(dr^2 n)$.



Preconditioning

- (Partial) pivoted Cholesky factorization $K^\nabla \approx FF^T$ provides a cheap and effective preconditioner $M = \sigma^2 I + FF^T$.
- Sherman-Morrison-Woodbury formula for M^{-1} is unstable in practice.
- Instead, we use $M^{-1} f = \sigma^{-2}(f - Q_1(Q_1^T f))$ where

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

- Crucial for the convergence of iterative solvers on K^∇

Dimensionality Reduction via Active Subspace Learning

- Many high-dimensional problems are low-dimensional in practice.
- The active subspace spans the dominant eigenvectors of the cov. matrix:

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

- The function can be well approximated by a GP on the reduced space using the kernel $k(P^T x, P^T x')$, where P projects onto the active subspace.

Experiment: Reconstructing Implicit Surfaces from Normals

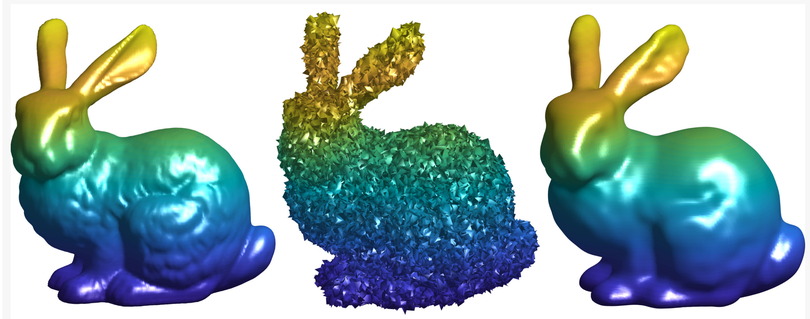


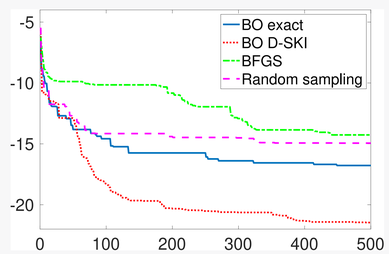
Figure: (Left) Original surface (Middle) Noisy surface (Right) SKI reconstruction from noisy surface

- The Stanford bunny is a data set of 25000 points and associated normals.
- We heavily pollute the data points and normals with noise, and accurately recover the underlying implicit surface from the noisy data.
- We use SKI with an induced grid of size 30 in each dimension.

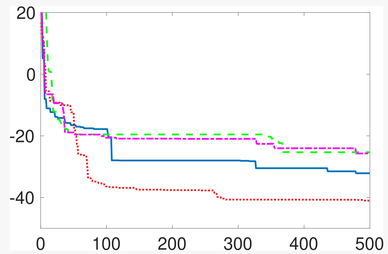
Bayesian Optimization with Derivatives and Active Subspace Learning

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

- We test on 5D Ackley embedded in $[-10, 15]^{50}$ and 5D Rastrigin in $[-4, 5]^{50}$.
- We pick a 2D active subspace at each iteration, using the D-SKI kernel and expected improvement acquisition function in this lower-dimensional space.



(a) BO on Ackley



(b) BO on Rastrigin

Discussion

- Gradient information is valuable for GP regression, but scalability is a problem.
- Our methods make computation of GP with derivatives scalable through fast MVMs, preconditioning, and dimensionality reduction.
- Our approach builds upon structured interpolation, but extends to any differentiable MVM.
- BO with derivatives unifies global optimization and gradient-based local optimization.
- Implementation available at:

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

https://people.cam.cornell.edu/~kd383/ kd383@cornell.edu