Scalable algorithms for kernel-based surrogates in prediction and optimization

David Bindel
1 Nov 2017
Surrogate optimization idea:

- Goal: minimize $f$
- Know some information (e.g. $f(x_1), \ldots, f(x_n)$)
- Fit an approximation $\hat{f}$ to guide next sample

See:

https://github.com/dme65/pySOT
https://github.com/dbindel/POAP

... and that is all I will say about surrogate optimization!
• Formal training: numerical analysis at CS/math border
• Research “home base”: numerical linear algebra
  • With applications from engineering and CS
  • Projections into optimization, approximation theory, HPC
• Most weeks, this means:
  CG = Conjugate Gradients
  DFT = Discrete Fourier Transform
  LDA = Latent Dirichlet Allocation
Compact modeling: PDE device model → small ODE model

- May involve reduced theory
  - Solid → beam and plate theory
  - Reaction-diffusion → CSTR
  - Maxwell → basic circuit elements
- Or automated model reduction (computer driven)
- Or phenomenological (e.g. most transistor models)
What makes a good reduced model depends on application:

- High accuracy
  - May not require uniform error bounds
  - May want an error indicator
- Good numerical stability
  - Careful bias/variance tradeoff
  - Regularization matters!
- Low computational expense
  - Set up fit in reasonable time?
  - More stringent demands on evaluation time
- Composability (structural constraints)
- Parameterized behavior

Goal today: Address first few points for kernel methods
Let $u = (u_1, u_2)$. Given $u_1$, what is $u_2$?

We need an assumption! Two different standard takes.
Let $U = (U_1, U_2) \sim N(0, K)$. Given $U_1 = u_1$, what is $U_2$?

Posterior distribution: $(U_2|U_1 = u_1) \sim N(w, S)$ where

$$w = K_{21}K_{11}^{-1}u_1$$

$$S = K_{22} - K_{21}K_{11}^{-1}K_{12}$$
Being bounded

Let $u = (u_1, u_2)$ s.t. $\|u\|_{K^{-1}}^2 \leq 1$. Given $u_1$, what is $u_2$?

Optimal recovery: $\|u_2 - w\|_{S^{-1}}^2 \leq 1 - \|u_1\|_{(K_{11})^{-1}}^2$

$$w = K_{21}K_{11}^{-1}u_1$$

$$S = K_{22} - K_{21}K_{11}^{-1}K_{12}$$
Both cases: $K$ plays a similar fundamental role

- Predictor minimizes $\|u\|_{K^{-1}}^2$ subject to data
- Extends beyond knowing values – OK if data is any $l^*u$
- Schur complement central to error estimates

But error interpretation is very different!
Basic ingredients

Gaussian Processes (GPs)

Our favorite continuous distributions over

\( \mathbb{R} \) : Normal(\( \mu, \sigma^2 \)), \( \mu, \sigma^2 \in \mathbb{R} \)

\( \mathbb{R}^n \) : Normal(\( \mu, C \)), \( \mu \in \mathbb{R}^n, C \in \mathbb{R}^{n \times n} \)

\( \mathbb{R}^d \rightarrow \mathbb{R} \) : GP(\( \mu, k \)), \( \mu : \mathbb{R}^d \rightarrow \mathbb{R}, k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R} \)

More technically, define GPs by looking at finite sets of points:

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

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

\( f_X \in \mathbb{R}^n, \) \( (f_X)_i \equiv f(x_i) \)

\( \mu_X \in \mathbb{R}^n, \) \( (\mu_X)_i \equiv \mu(x_i) \)

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

When \( X \) is unambiguous, we will sometimes just write \( K \).
Bayesian inference

Prior: $f \sim \text{GP}(\mu, k)$, noisy measurements

\[
f_X \sim y + \epsilon, \quad \epsilon \sim \mathcal{N}(0, W), \quad \text{typically } W = \sigma^2 I
\]

Posterior: $f \sim \text{GP}(\mu', k')$ with

\[
\mu'(x) = \mu(x) + K_{xx}c
\]
\[
k'(x, x') = K_{xx'} - K_{xx} \tilde{K}^{-1} K_{xx'}
\]
\[
\tilde{K} = K_{xx} + W
\]
\[
c = \tilde{K}^{-1}(y - \mu_X)
\]
Cubic splines

Minimize bending energy

\[ \mathcal{E}(u) = \int_{\Omega} |u''|^2 \, dx \]

subject to constraints. Write solution as

\[ s(x) = \sum_{j=1}^{n} c_j |x - x_j|^3 + a_1 x + a_2 \]

where

\[
\begin{bmatrix}
K & P \\
PT & 0
\end{bmatrix}
\begin{bmatrix}
c \\
a
\end{bmatrix} =
\begin{bmatrix}
f_x \\
0
\end{bmatrix}
\]

with \( K_{ij} = |x_i - x_j|^3 \), \( P_{1i} = x_i \), \( P_{2i} = 1 \).

**Interpret:** \( f_x \) is displacements, \( c \) is forces, \( c^TKc \) as energy.

“Native space” \( \mathcal{H}^2(\Omega) \) of functions where energy makes sense.
Beyond cubic splines

- Define a native space (a RKHS) via kernel
  - Start with kernel $k(x, \cdot)$
  - Interpolants are linear combinations of kernels
  - Native space is closure of space of all interpolants

- Interpolation: minimize $|s|^2$ s.t. $s_X = y$
- Smoothing: minimize $\|s_X - y\|^2 + \lambda |s|^2$
- Smoothing spline $\equiv$ noisy GP
What is a kernel function $k(x, y)$?

- A useful basis function for interpolation
- A Green’s function for a PDE (for polyharmonic splines)
- An inner product in a feature space: $k(x, y) = \langle \phi(x), \phi(y) \rangle$
- A covariance function
- A reproducing kernel in a certain RKHS

Choose whichever makes you happy...
Kernel properties

For variational interp, need:

- **Pos def**: need $K_{xx}$ positive definite or
- **Conditional pos def**: $c^T K_{xx} c > 0$ for $c \neq 0$ and $P^T c = 0$

Often desirable:

- **Stationary**: $k(x, y)$ depends only on $x - y$
- **Isotropic**: $k(x, y)$ depends only on $x$ and $\|x - y\|$

Radial basis function is both (sloppy notation: $k = k(r)$).
Matérn and SE kernels

- Matérn 1/2
- Matérn 3/2
- Matérn 5/2
- Squared exp
A few ways to let physics inform kernel approximation:

- Less smooth kernels for less regular functions.
- Can always symmetrize kernel; for symmetry group $G$:
  
  $$k_{\text{sym}}(x, y) = \frac{1}{|G|} \sum_{Q \in G} k(Qx, Qy)$$

- Can include known singularities in a tail term:
  
  $$f(x) \approx \sum_{j=1}^{n} c_k k(x, x_j) + B(x)a$$

  where $B$ is a basis for an extra space (polynomial or other).
- Can symmetrize the tail as well.
Observations on kernel matrices

Kernel is *chosen by modeler*

- Choose Matérn / SE for regularity and simplicity
- Rarely have the intuition to pick the “right” kernel
- Common choices are *universal* — can recover anything
  - ... though with less data for a “good” choice
- Universality may not be needed for all kernels...

Properties of kernel matrices:

- Positive definite by design, but not well conditioned!
- Weyl: $k(r) \in C^{\nu} \implies |\lambda_n| = o(n^{-\nu-1/2})$
- SE case: eigenvalues decay exponentially
- Adding $\sigma^2 I$ “wipes out” small eigenvalues
Hyper-parameter MLE

How to estimate hyper-parameters (i.e. $\ell$ and $s_f$, $\sigma^2$)? Recall

$$f(y) = \frac{1}{\sqrt{\text{det}(2\pi \tilde{K}_{xx})}} \exp \left( -\frac{1}{2} u^T \tilde{K}_{xx}^{-1} u \right)$$

Log-likelihood function for kernel hypers $\theta$

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

where (again with $c = \tilde{K}^{-1}(y - \mu_x)$)

$$\mathcal{L}_y = -\frac{1}{2} (y - \mu_x)^T c,$$

$$\frac{\partial \mathcal{L}_y}{\partial \theta_i} = \frac{1}{2} c^T \left( \frac{\partial \tilde{K}}{\partial \theta_i} \right) c$$

$$\mathcal{L}_{|K|} = -\frac{1}{2} \log \text{det} \tilde{K},$$

$$\frac{\partial \mathcal{L}_{|K|}}{\partial \theta_i} = -\frac{1}{2} \text{tr} \left( \tilde{K}^{-1} \frac{\partial \tilde{K}}{\partial \theta_i} \right)$$
Consider $n$ data points

- Straightforward regression: factor $\tilde{K}$ at $O(n^3)$ cost
- Kernel hyper MLE requires multiple $O(n^3)$ ops
  - To compute $\log \det \tilde{K}$ is $O(n^3)$ per step
  - To compute $\text{tr} \left( \tilde{K}^{-1} \frac{\partial \tilde{K}}{\partial \theta_i} \right)$ is $O(n^3)$ per hyper per step
- GCV has similar costs (see Golub, Heath, Wahba 1979)

Two possible work-arounds

- Data-sparse factorization methods
- Methods that avoid factorization (e.g. iterative solvers)
  - Q: how to handle determinants and traces?

Today: The second approach.
Basic ingredients

- Fast MVMs with kernel matrices
- Krylov methods for linear solves and matrix functions
- Stochastic estimators: trace, diagonal, and other
Kernel approximations

Goal: Fast matrix-vector multiplication

- Low-rank approximation
  - Often phrased via *inducing points*
  - Non-smooth kernels, small length scales $\implies$ large rank
  - Only semi-definite

- Sparse approximation
  - OK with SE kernels and short length scales
  - Less good with heavy tails or long length scales
  - May again lose definiteness

- More sophisticated: fast multipole, Fourier transforms
  - Same picture as in integral eq world (FMM, PFFT)
  - Main restriction: low dimensional spaces (2-3D)
  - But... this really means “near” vs “far”

- Kernel a model choice — how does approx affect results?
Write $K_{XX} \approx W^T K_{UU} W$ where

- $U$ is a uniform mesh of $m$ points ($m$ not always small)
- Sparse $W$ interpolates values from $X$ to $U$
- $K_{UU}$ has Toeplitz or block Toeplitz structure

Apply $K_{UU}$ via FFTs in $O(m \log m)$ time.
The power of fast MVMs

Fast MVM with symmetric $\tilde{K} \implies$ try Lanczos!

- Incrementally computes $\tilde{K}Q = QT$ where
  - $Q$ has orthonormal columns
  - Leading $k$ columns span $k$-dim Krylov space
  - $T$ is tridiagonal

- Building block for
  - Solving linear systems (CG)
  - Approximating eigenvalues
  - Approximating matrix functions: $f(\tilde{K})b$
  - Quadrature vs spectral measure for $\tilde{K}$

- Fast (three-term recurrence) and elegant...
- ... but not forward stable in finite precision
A computational kernel: \( f(\tilde{K})b \)

- Run Lanczos from starting vector \( b/\|b\| \)
- At \( n \) steps in exact arithmetic,
  \[
  f(\tilde{K})b = Qf(T)Q^Tb = \|b\|Qf(T)e_1
  \]
- Truncate at \( k \ll n \) steps, use
  \[
  f(\tilde{K})b \approx \|b\|Q_1f(T_{11})e_1
  \]
- Error analysis hinges on quality of poly approx
  \[
  \min_{f \in P_k} \max_{\lambda \in \Lambda(\tilde{K})} |f(\lambda) - \hat{f}(\lambda)|
  \]
- Compare: Chebyshev methods just use \([\lambda_{\min}, \lambda_{\max}]\)

CG is a special case corresponding to \( f(z) = z^{-1} \).
Tractable traces

CG solves systems with $\tilde{\mathbf{K}}$; problem terms are

$$\mathcal{L}_{|\mathbf{K}|} = -\frac{1}{2} \text{tr} \left( \log \tilde{\mathbf{K}} \right) \quad \frac{\partial \mathcal{L}_{|\mathbf{K}|}}{\partial \theta_i} = -\frac{1}{2} \text{tr} \left( \tilde{\mathbf{K}}^{-1} \frac{\partial \tilde{\mathbf{K}}}{\partial \theta_i} \right)$$

Q: How do we parley fast MVMs into trace computations?
Tractable traces

Stochastic trace estimation trick:

• \( z \in \mathbb{R}^n \) has independent random entries
• \( \mathbb{E}[z_i] = 0 \) and \( \mathbb{E}[z_i^2] = 1 \)

Then

\[
\mathbb{E}[z^T A z] = \sum_{i,j} a_{ij} \mathbb{E}[z_i z_j] = \text{tr}(A).
\]

NB: \( \mathbb{E}[z \odot A z] = \text{diag}(A) \).

Standard choices for the probe vector \( z \):

• Hutchinson: \( z_i = \pm 1 \) with probability 0.5
• Gaussian: \( z_i \sim N(0, 1) \)

See Avron and Toledo review, JACM 2011.
Putting it together

For each probe vector $z$ until error bars small enough:

- Run Lanczos from $z/\|z\|$.
- Use Lanczos to estimate $\tilde{K}^{-1} z$ and $\log(\tilde{K}) z$.
- Dot products yield estimators:

$$L_{|K|} = -\frac{1}{2} \mathbb{E} \left[ z^T \log(\tilde{K}) z \right]$$

$$\frac{\partial L_{|K|}}{\partial \theta_i} = -\frac{1}{2} \mathbb{E} \left[ (\tilde{K}^{-1} z)^T \left( \frac{\partial \tilde{K}}{\partial \theta_i} z \right) \right]$$

Cost per probe:

- One Lanczos process.
- One matvec per parameter with derivative.

This is quite effective in practice!
Hessian estimators

• For Hessian of $\mathcal{L}_y$, exploit $\mathbb{E}[zz^T] = I$:

$$
\frac{\partial^2 \mathcal{L}_y}{\partial \theta_i \partial \theta_j} = \frac{1}{2} c^T \left( \frac{\partial^2 K}{\partial \theta_i \partial \theta_j} - 2 \frac{\partial K}{\partial \theta_i} \frac{\partial K}{\partial \theta_j} \right) c
$$

$$
= \frac{1}{2} \mathbb{E} \left[ c^T \left( \frac{\partial^2 K}{\partial \theta_i \partial \theta_j} - 2 \frac{\partial K}{\partial \theta_i} zz^T \tilde{K}^{-1} \frac{\partial K}{\partial \theta_j} \right) c \right]
$$

• Tackle Hessian of $\mathcal{L}_{|K|}$ with independent probe $\tilde{z}$:

$$
\frac{\partial^2 \mathcal{L}_{|K|}}{\partial \theta_i \partial \theta_j} = \frac{1}{2} \text{tr} \left( \tilde{K}^{-1} \frac{\partial K}{\partial \theta_i} \tilde{K}^{-1} \frac{\partial K}{\partial \theta_j} - \tilde{K}^{-1} \frac{\partial^2 K}{\partial \theta_i \partial \theta_j} \right)
$$

$$
= \frac{1}{2} \mathbb{E} \left[ z^T \left( \tilde{K}^{-1} \frac{\partial K}{\partial \theta_i} \tilde{K}^{-1} \frac{\partial K}{\partial \theta_j} - \tilde{K}^{-1} \frac{\partial^2 K}{\partial \theta_i \partial \theta_j} \right) z \right]
$$

• Too much variance to be useful without help
Control variates

If unsatisfied with estimator, use *control variates*:

\[
\mathbb{E}[X] \text{ desired}
\]

\[
\mathbb{E}[Y] = 0
\]

\[
\mathbb{E}[X - \alpha Y] = E[X]
\]

\[
\text{Var}[X - \alpha Y] = \text{Var}[X] - 2\alpha \text{Cov}[X, Y] + \alpha^2 \text{Var}[Y]
\]

Optimal choice is

\[
\alpha_* = \frac{\text{Cov}[X, Y]}{\text{Var}[Y]}, \quad \text{Var}[X - \alpha_* Y] = \text{Var}[X] - \frac{\text{Cov}[X, Y]^2}{\text{Var}[Y]}
\]

Idea: Crude kernel approximants to construct control variates.
So where are we now?
Have you ever seen the rain?
Have you ever seen the rain?

- Data: Hourly precipitation data at 5500 weather stations
- Aggregate into daily precipitation
- Total data: 628K entries
- Train on 100K data points, test on remainder
- Use SKI with 100 points per spatial dim, 300 in time
- Reference comparisons:
  - Scaled eigenvalue approximation for log det
  - Smaller exact computation (12K entries)
<table>
<thead>
<tr>
<th>Method</th>
<th>$n$</th>
<th>$m$</th>
<th>MSE</th>
<th>Time [min]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lanczos</td>
<td>528k</td>
<td>3M</td>
<td>0.613</td>
<td>14.3</td>
</tr>
<tr>
<td>Scaled eigenvalues</td>
<td>528k</td>
<td>3M</td>
<td>0.621</td>
<td>15.9</td>
</tr>
<tr>
<td>Exact</td>
<td>12k</td>
<td>-</td>
<td>0.903</td>
<td>11.8</td>
</tr>
</tbody>
</table>

So should we just stick to scaled eigs?
Hickory data

- Log-Gaussian Cox process model
  - Poisson conditional on intensity function
  - Log intensity drawn from a GP
- Laplace approximation for posterior
- Data set is point pattern of 703 hickory trees in Michigan
### Table 1: Hyper-parameters recovered by different methods

| Method     | $s_f$  | $\ell_1$ | $\ell_2$ | $-\log p(y|\theta)$ | Time [s] |
|------------|--------|----------|----------|-----------------------|----------|
| Exact      | 0.696  | 0.063    | 0.085    | 1827.56               | 465.9    |
| Lanczos    | 0.693  | 0.066    | 0.096    | 1828.07               | 21.4     |
| Scaled eigs| 0.543  | 0.237    | 0.112    | 1851.69               | 2.5      |
Figure 1: Prediction by different methods on the Hickory dataset.
Conclusions (?)

“Scalable Log Determinants for GP Kernel Learning”
NIPS 2017 (and will appear on arXiV r.s.n.)

Still pursuing connections!

• Variance reduction (esp. for Hessian info)
• Fast large-scale posterior variance
• Connections to Bayesian optimization

Would love to add some chemical applications to the list.