

Stochastic LA for Scaling GPs

David Bindel

25 Sep 2017

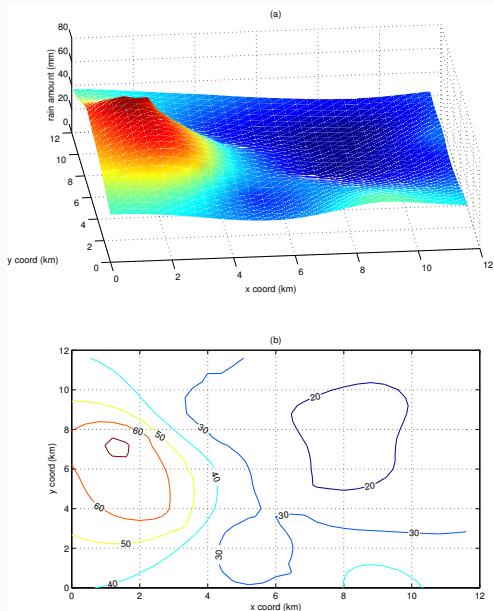
Today's talk is:

- Somewhat related to Nick T's CAM talk
- Also related to my NA reading group talks (Oct 11+18)
- Work with Kun Dong, David Eriksson, Andrew Wilson, Hannes Nickisch
 - And Kun and David will speak in SCAN later this semester!
- Also related to work by an undergrad I used to know...

TkBTG (c. 1997)



The most remarkable thing about the rain



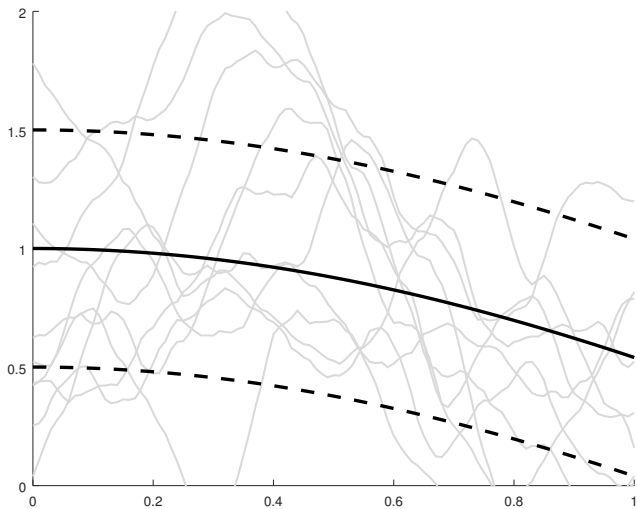
Fast forward two decades (minus a bit)

ML is the hot new reason to do computational stats

- Spring 2016: Meet Andrew Wilson on interview
- Examples still look like spatio-temporal stats
- But now chatting about GPs with $10^6 - 10^9$ data points
- NB: Rain gauge data from 1997 pics was at 24 sites

Scale presents some interesting new challenges!

Basic ingredient: Gaussian Processes (GPs)



Basic ingredient: Gaussian Processes (GPs)

Our favorite continuous distributions over

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

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

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

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

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

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

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

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

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

When X is unambiguous, we will sometimes just write K .

Kernel functions

Call the *kernel* (or *covariance*) function k . Required property:

- **Pos def:** K_{XX} is always positive definite

Often desirable:

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

Often want both (sloppy notation: $k = k(r)$).

Matérn and RBF kernels

Let $d = r/\ell$. Two very popular stationary and isotropic choices:

$$k_{\text{Mat},\nu} = s_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} (\sqrt{2\nu}d)^\nu K_\nu(\sqrt{2\nu}d)$$

$$k_{\text{RBF}}(r) = s_f^2 \exp(-d^2/2)$$

For integer-plus-half ν , get polynomial times exponential:

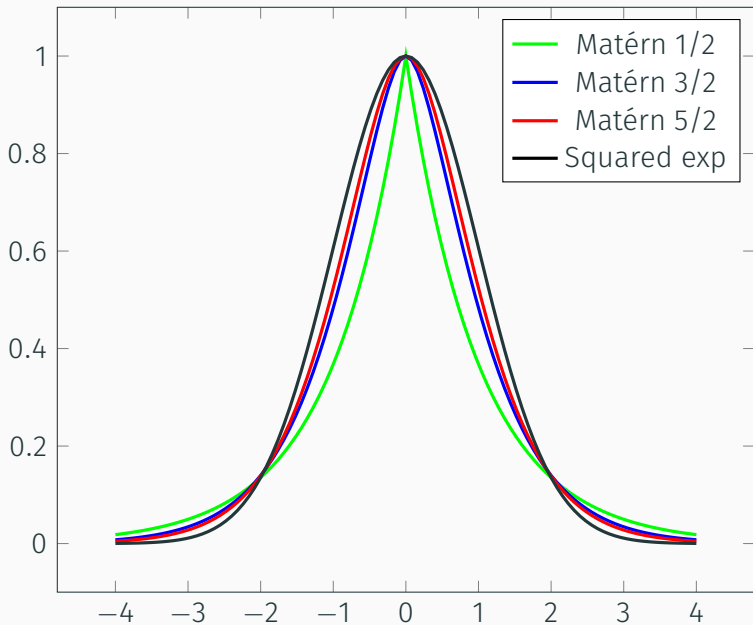
$$k_{\text{Mat},1/2}(r) = s_f^2 \exp(-d)$$

$$k_{\text{Mat},3/2}(r) = s_f^2 (1 + \sqrt{3}d) \exp(-\sqrt{3}d)$$

$$k_{\text{Mat},5/2}(r) = s_f^2 \left(1 + \sqrt{5}d + \frac{5}{3}d^2\right) \exp(-\sqrt{5}d)$$

Bigger ν gives smoother functions; converges to RBF in limit.

Matérn and RBF kernels



Observations on kernel matrices

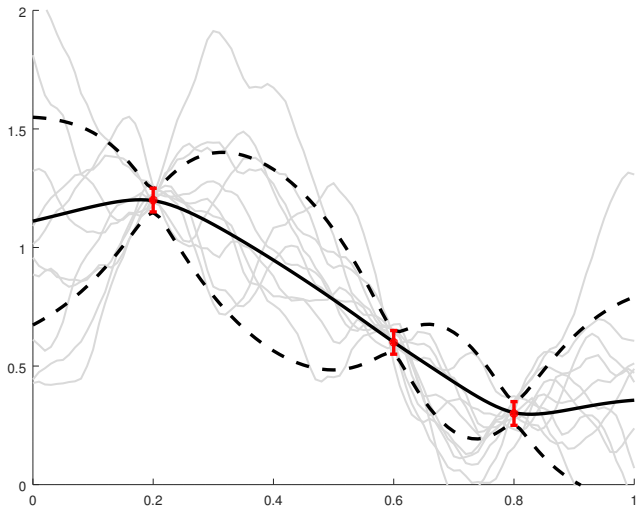
Kernel is *chosen by modeler*

- Choose Matérn / RBF 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

Properties of kernel matrices:

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

Being Bayesian



Bayesian approach to inference:

- Prior distribution (say on X and Y) represents initial belief.
- Experiments reveal $X = x$
- Update to posterior distribution $f_{Y|X}(y|x) = f_{X,Y}(x,y)/f_X(x)$.

Being Bayesian

Consider $U = (X, Y) \sim N(0, C)$, i.e.

$$f_U(u) = \frac{1}{\sqrt{\det(2\pi C)}} \exp\left(-\frac{1}{2}u^T C^{-1}u\right)$$

Partial Gaussian elimination involving C yields:

$$u^T C^{-1}u = (y - w)^T S^{-1}(y - w) + q(x), \text{ where}$$

$$S = C_{22} - C_{21}C_{11}^{-1}C_{12}$$

$$w = C_{21}C_{11}^{-1}x$$

$$q(x) = \text{quadratic in } x \text{ only}$$

Thus

$$Y|X \sim N(w, S)$$

Now consider prior of $f \sim \text{GP}(\mu, k)$, noisy measurements

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

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

$$\begin{aligned} \mu'(x) &= \mu(x) + K_{xx}c & \tilde{K} &= K_{xx} + W \\ k'(x, x') &= K_{xx'} - K_{xx}\tilde{K}^{-1}K_{xx'} & c &= \tilde{K}^{-1}(y - \mu_X) \end{aligned}$$

Kernel hyper-parameters

How to estimate *hyper-parameters* (i.e. ℓ and s_f)?

- Bayesian approach? Expensive...
- Usually just do maximum likelihood estimation (MLE)

Log-likelihood function for kernel hypers θ

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

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

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

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

Scalability bottlenecks

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

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 RBF 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)
- Kernel a model choice — how does approx affect results?

Example: Structured Kernel Interpolation (SKI)

Write $K_{XX} \approx W^T K_{UU} W$ where

- U is a uniform mesh of m points
- K_{UU} has Toeplitz or block Toeplitz structure
- Sparse W interpolates values from X to U

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

Function application via Lanczos

A computational kernel: $f(\tilde{K})b$

- Run Lanczos from starting vector $b/\|b\|$
- At n steps in exact arithmetic,

$$f(\tilde{K})b = Qf(T)Q^T b = \|b\|Qf(T)e_1$$

- 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 $f(z) = z^{-1}$.

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

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

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:

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

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 $\mathbb{E}[zz^T] = I$:

$$\begin{aligned}\frac{\partial^2 \mathcal{L}_y}{\partial \theta_i \partial \theta_j} &= \frac{1}{2} c^T \left(\frac{\partial^2 K}{\partial \theta_i \partial \theta_j} - 2 \frac{\partial K}{\partial \theta_i} \tilde{K}^{-1} \frac{\partial K}{\partial \theta_j} \right) c \\ &= \frac{1}{2} \mathbb{E} \left[c^T \left(\frac{\partial^2 K}{\partial \theta_i \partial \theta_j} - 2 \frac{\partial K}{\partial \theta_i} z z^T \tilde{K}^{-1} \frac{\partial K}{\partial \theta_j} \right) c \right]\end{aligned}$$

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

$$\begin{aligned}\frac{\partial^2 \mathcal{L}_{|K|}}{\partial \theta_i \partial \theta_j} &= \frac{1}{2} \text{tr} \left(\tilde{K}^{-1} \frac{\partial K}{\partial \theta_i} \tilde{K}^{-1} \frac{\partial K}{\partial \theta_j} - \tilde{K}^{-1} \frac{\partial^2 K}{\partial \theta_i \partial \theta_j} \right) \\ &= \frac{1}{2} \mathbb{E} \left[z^T \left(\tilde{K}^{-1} \frac{\partial K}{\partial \theta_i} \check{z} \check{z}^T \tilde{K}^{-1} \frac{\partial K}{\partial \theta_j} - \tilde{K}^{-1} \frac{\partial^2 K}{\partial \theta_i \partial \theta_j} \right) z \right]\end{aligned}$$

- Too much variance to be useful without help

Control variates

If unsatisfied with estimator, use *control variates*:

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

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

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

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

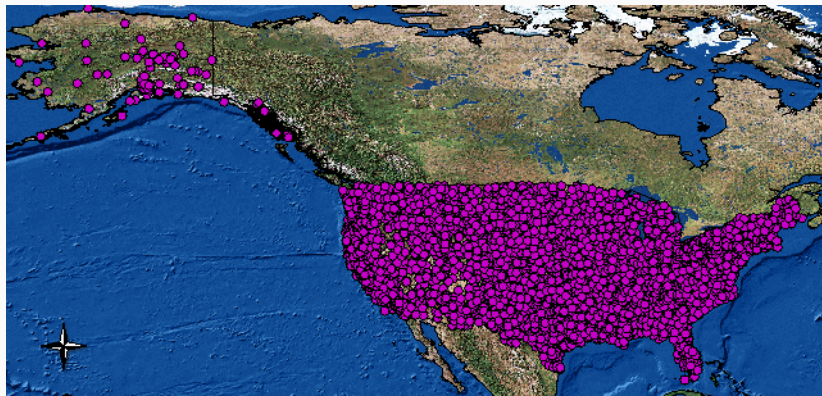
Optimal choice is

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

Idea: Crude kernel approximants to construct control variates.

So where are we now?

The rain, revisited



Map generated by NOAA's National Climatic Data Center, 2007

0 698mi

The rain, revisited

- 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)

The rain, revisited

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

So should we just stick to scaled eigs?

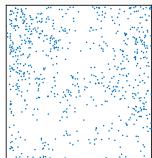
- Log-Gaussian Cox process model
 - Poisson conditional on intensity function
 - Log intensity drawn from a GP
- Laplace approximation for posterior
- Data set is point pattern of 703 hickory trees in Michigan

Hickory data

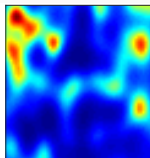
Method	s_f	l_1	l_2	$-\log p(y \theta)$	Time [s]
Exact	0.696	0.063	0.085	1827.56	465.9
Lanczos	0.693	0.066	0.096	1828.07	21.4
Scaled eigs	0.543	0.237	0.112	1851.69	2.5

Table 1: Hyper-parameters recovered by different methods

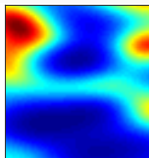
Hickory data



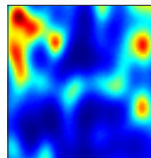
(a) Points



(b) Exact



(c) Scaled eigs



(d) Lanczos

Figure 1: Prediction by different methods on the Hickory dataset.

Conclusions (?)

“Scalable Log Determinants for GP Kernel Learning”
K. Dong, D. Eriksson, H. Nickisch, D. Bindel, A. G. Wilson
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