Stochastic LA for Scalable GPs

David Bindel

4 Jun 2018

Collaborators

Work presented:

- Andrew Wilson (Cornell ORIE)
- Hannes Nickisch (Phillips Research)
- Kun Dong (Cornell CAM)
- David Eriksson (Cornell CAM)

Work hinted at:

- Kilian Weinberer (Cornell CS)
- Jake Gardner (Cornell CS)
- Geoff Pleiss (Cornell CS)
- Eric Lee (Cornell CS)

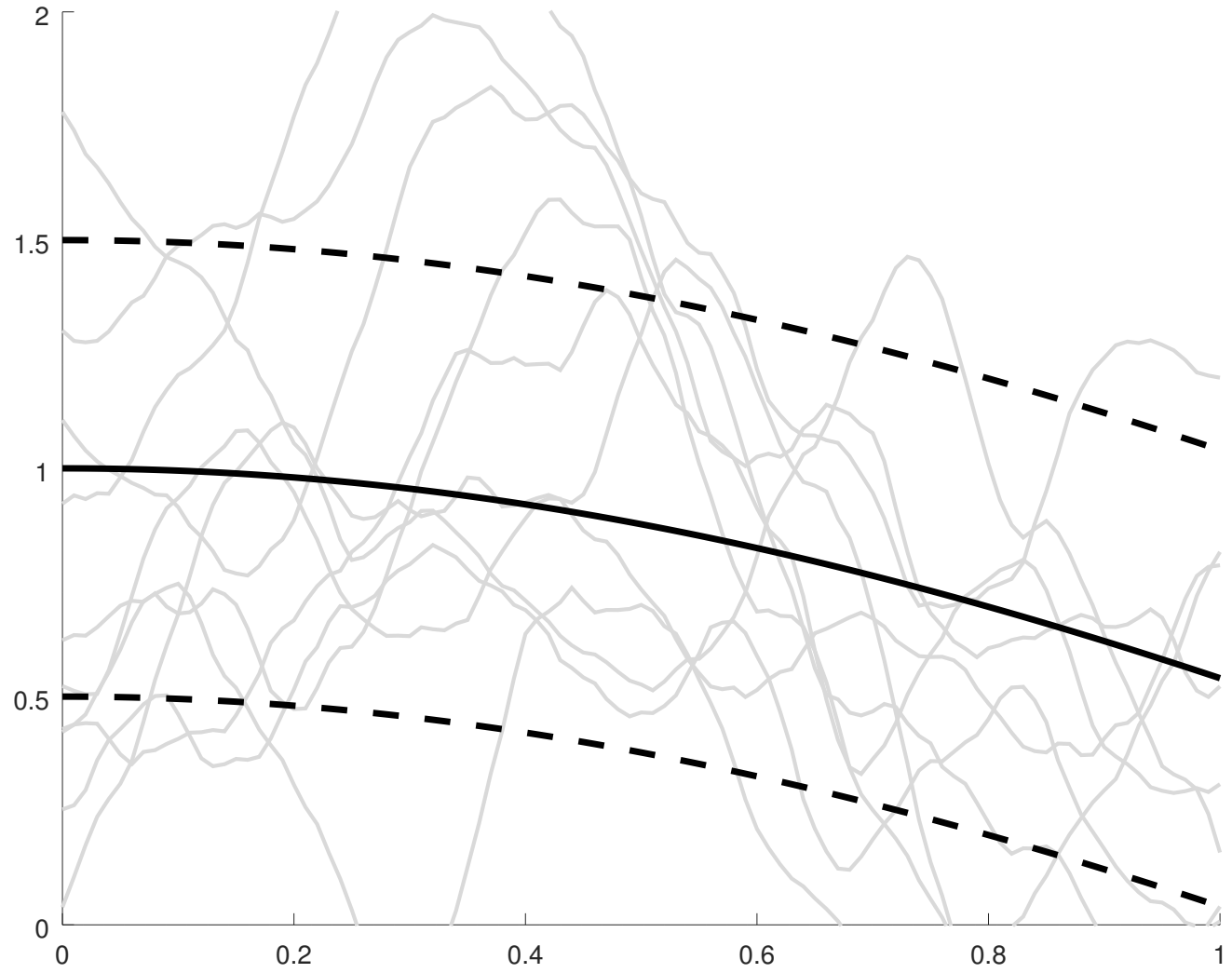
The Big Picture

Gaussian processes (GPs) are

- Key building block for ML and spatio-temporal statistics
- Tightly connected to integral equations, kernel regression
- Straightforward to reason about (just linear algebra)
- But hard to scale to big data (because of dense LA)

Goal today: How to make these methods scale!

Basic ingredient: Gaussian Processes (GPs)



Basic ingredient: Gaussian Processes (GPs)

Our favorite continuous distributions over

$$\begin{array}{lll} \mathbb{R}: & \text{Normal}(\mu, \sigma^2), & \mu, \sigma^2 \in \mathbb{R} \\ \mathbb{R}^n: & \text{Normal}(\mu, C), & \mu \in \mathbb{R}^n, C \in \mathbb{R}^{n \times n} \\ \mathbb{R}^d \to \mathbb{R}: & \text{GP}(\mu, k), & \mu : \mathbb{R}^d \to \mathbb{R},\ k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R} \end{array}$$

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

$$\begin{aligned} \forall X = (x_1, \ldots, x_n),\ x_i \in \mathbb{R}^d, & \\ \text{have } f_X \sim N(\mu_X, K_{XX}), & \text{ where} \\ f_X \in \mathbb{R}^n, \quad & (f_X)_i \equiv f(x_i) \\ \mu_X \in \mathbb{R}^n, \quad & (\mu_X)_i \equiv \mu(x_i) \\ K_{XX} \in \mathbb{R}^{n \times n}, \quad & (K_{XX})_{ij} \equiv k(x_i, x_j) \end{aligned}$$

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

Basic ingredient: Kernel functions

Kernels for function approximation three ways:

- Feature maps: $\phi : \mathbb{R}^d \to \mathcal{H}$, $k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$.
  Approx scheme: $f(x) \approx w^* \phi(x)$ with $\|w\|_2$ minimal.
- Shape functions: $f(x) \approx \sum_j c_j k(x, x_j)$.
  Equivalent to feature map picture ("kernel trick").
- Covariance for Gaussian process.

RBF / kernel ridge regression / GP differ mainly in regularization, interpretation of error analysis.

Basic ingredient: Kernel functions

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

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

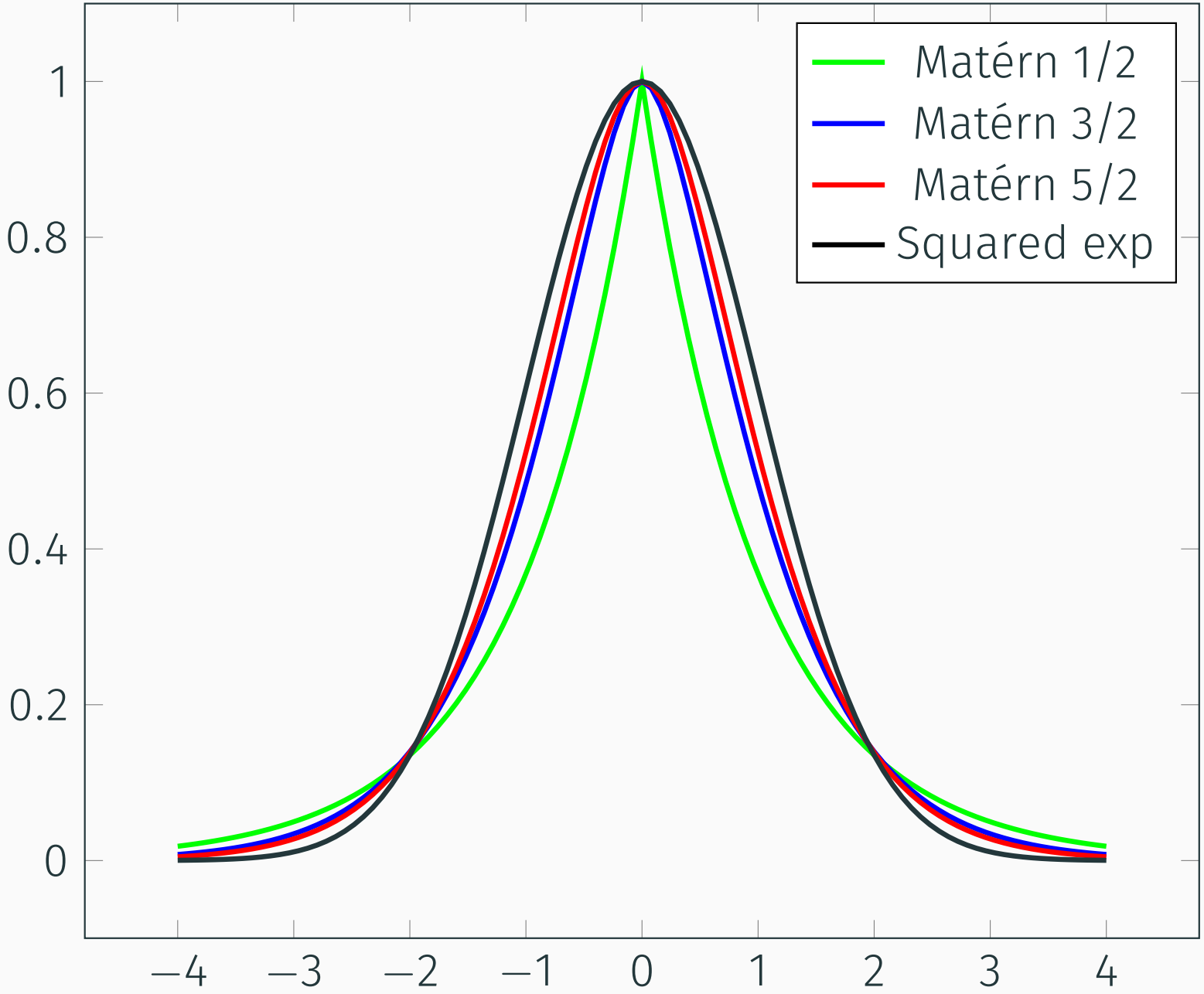
Often desirable:

- **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: $k = k(r)$).

Common examples (e.g. Matérn, SE) also depend on *hyper-parameters* θ — suppressed in notation unless needed.

Matérn and SE kernels



Observations on kernel matrices

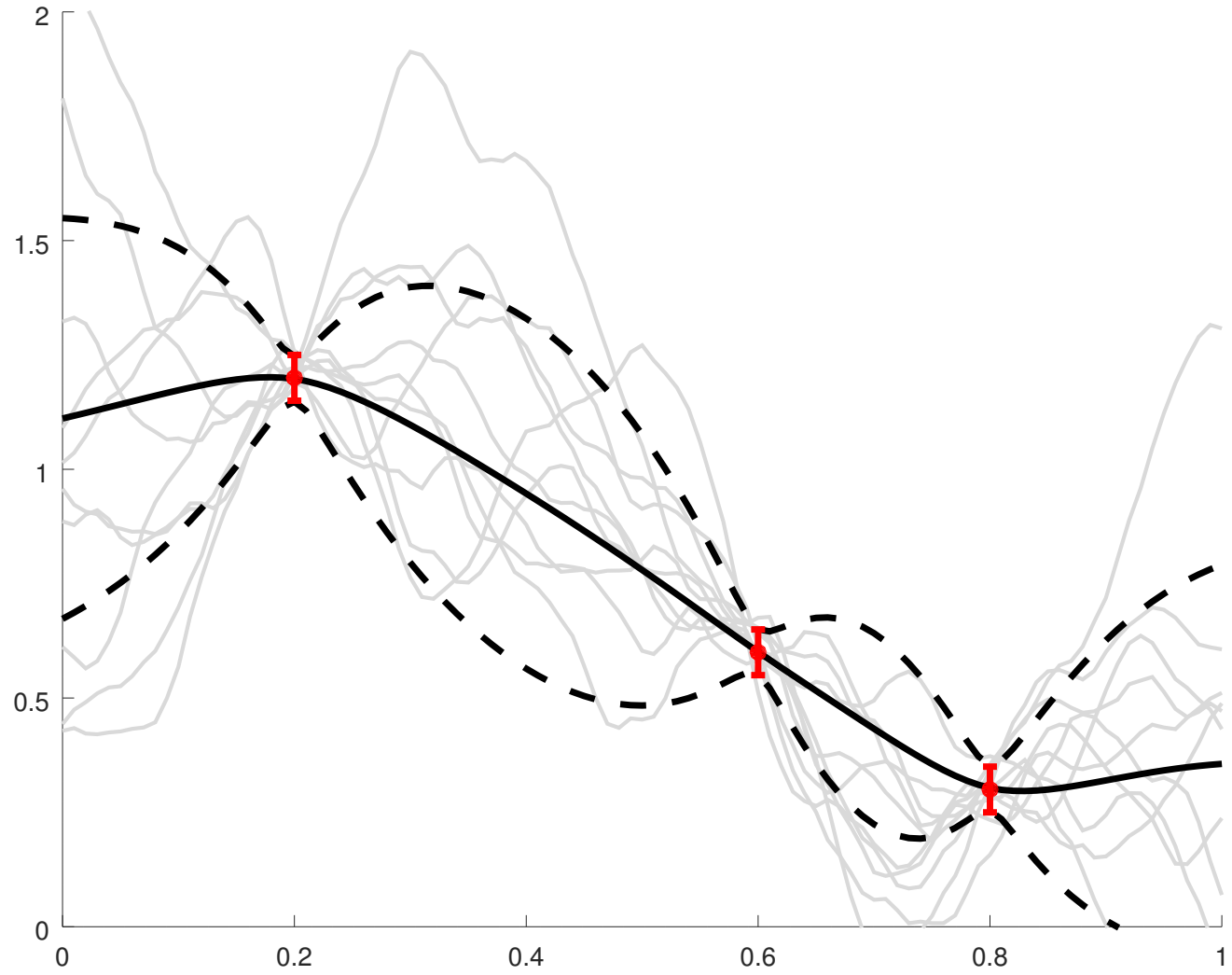
Kernel is *chosen by modeler*

- Choose Matérn / SE for regularity and simplicity
- Rarely have the intuition to pick the "right" kernel
- Common choices are *universal* — can recover anything
  - ... with less data for "good" choice (inductive bias)
- Can combine with DNNs ("deep kernel learning")

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 (super)exponentially
- Adding $\sigma^2 I$ "wipes out" small eigenvalues

Being Bayesian



Being Bayesian

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

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

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

$$\mu'(x) = \mu(x) + K_{xX}c \qquad \tilde{K} = K_{XX} + W$$
$$k'(x, x') = K_{xx'} - K_{xX}\tilde{K}^{-1}K_{Xx'} \qquad c = \tilde{K}^{-1}(y - \mu_X)$$

The expensive bit: solves with $\tilde{K}$.

Kernel hyper-parameters

How to estimate *hyper-parameters* θ?

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

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

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

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 $\operatorname{tr}\left(\tilde{K}^{-1}\frac{\partial \tilde{K}}{\partial \theta_i}\right)$ 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

- Low-rank approximation (via *inducing variables*)
  - 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)
- 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

$$f(\tilde{K})b \approx \|b\|Q_1 f(T_{11})e_1$$

- 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 $[\lambda_{\mathsf{min}}, \lambda_{\mathsf{max}}]$

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

Tractable traces

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

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

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

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

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

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

Quite effective in practice! And amenable to preconditioning.

Control variates

If unsatisfied with estimator, use *control variates*:

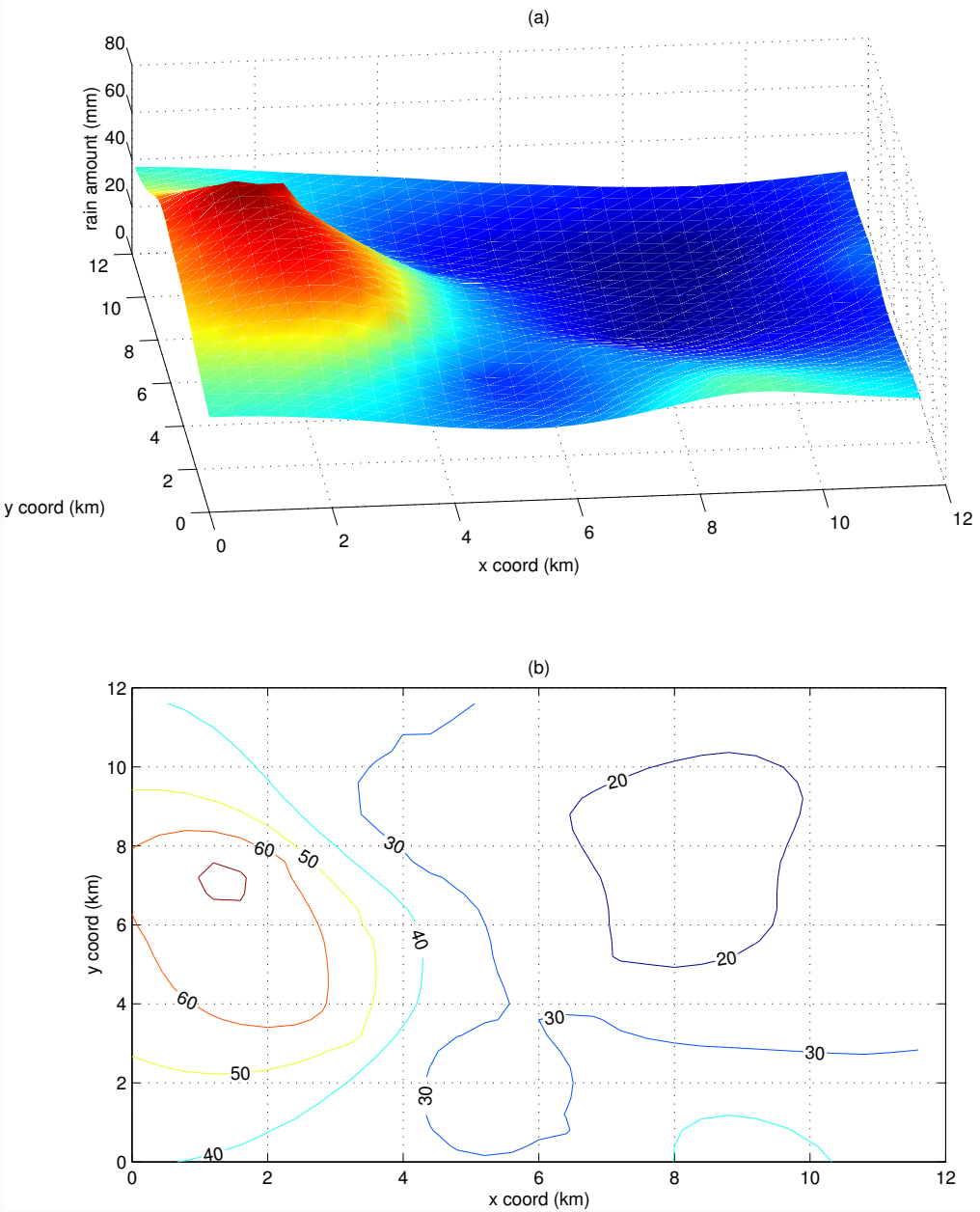
$$\begin{aligned} \mathbb{E}[X] &\text{ desired} \\ \mathbb{E}[Y] &= 0 \\ \mathbb{E}[X - \alpha Y] &= E[X] \\ \mathsf{Var}[X - \alpha Y] &= \mathsf{Var}[X] - 2\alpha\,\mathsf{Cov}[X, Y] + \alpha^2\,\mathsf{Var}[Y] \end{aligned}$$

Optimal choice is

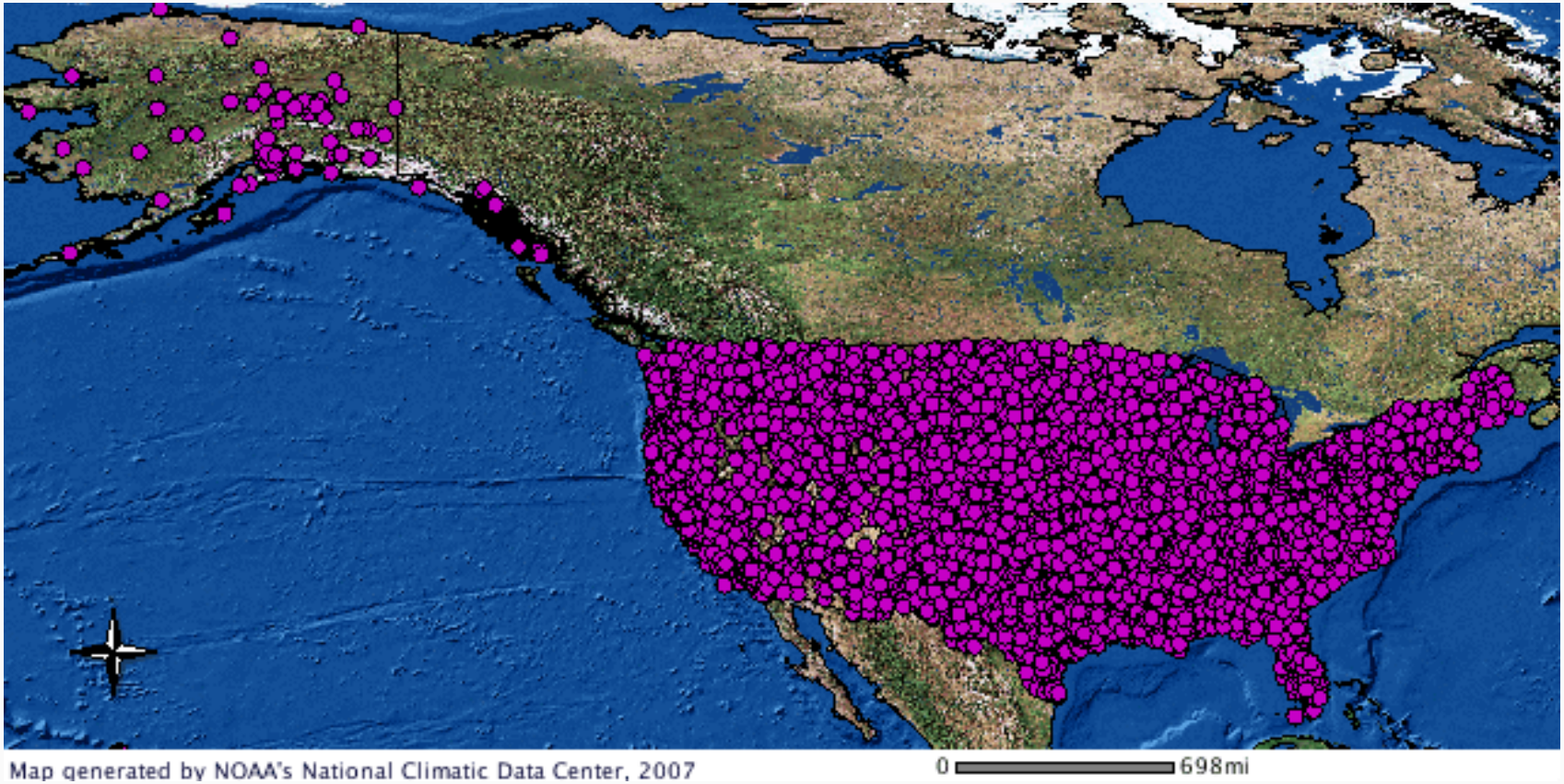
$$\alpha_* = \mathsf{Cov}[X, Y]/\,\mathsf{Var}[Y], \quad \mathsf{Var}[X - \alpha_* Y] = \mathsf{Var}[X] - \frac{\mathsf{Cov}[X, Y]^2}{\mathsf{Var}[Y]}.$$

Idea: Crude kernel approximants to construct control variates.

Example: Rainfall



Example: Rainfall



Example: 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

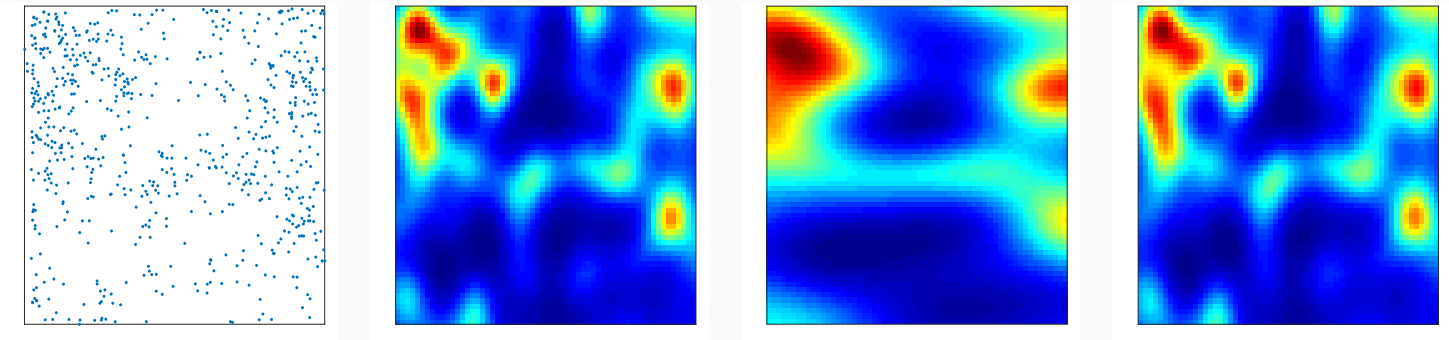
- Data: Hourly precipitation data at 5500 weather stations
- Aggregate into daily precipitation
- Total data: $628K$ entries
- Train on $100K$ data points, test on remainder
- Use SKI with 100 points per spatial dim, 300 in time
- Comparison: scaled eigenvalues approx, exact solve

Example: Hickory data

Can build other stochastic processes via GPs

- Example: Log-Gaussian Cox process model
  - Models count data (e.g. events in spatial bins)
  - 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

Example: Hickory data



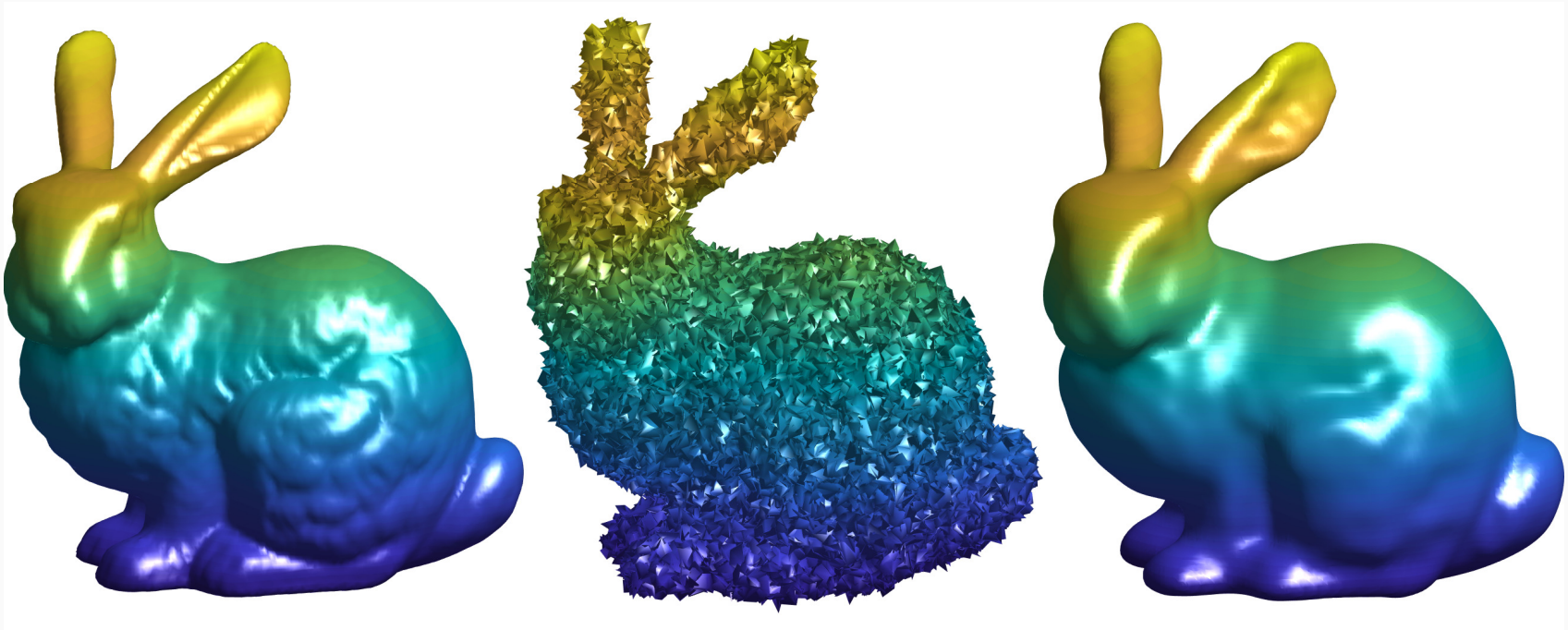
(a) Points (b) Exact (c) Scaled eigs (d) Lanczos

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

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

Teaser example: Applications with derivatives



Recovering the Stanford bunny model from 25K noisy normals.

Conclusions

"Scalable Log Determinants for GP Kernel Learning"
K. Dong, D. Eriksson, H. Nickisch, D. Bindel, A. G. Wilson
NIPS 2017

Still pursuing connections!

- Improved variance reduction (esp. for Hessian info)
- More efficient methods in high-dimensional spaces
- Fast large-scale posterior variance
- Connections to Bayesian optimization