Scalable Gaussian processes with a twist of Probabilistic Numerics

Kurt Cutajar

EURECOM, Sophia Antipolis, France

Data Science Meetup - October 30th 2017

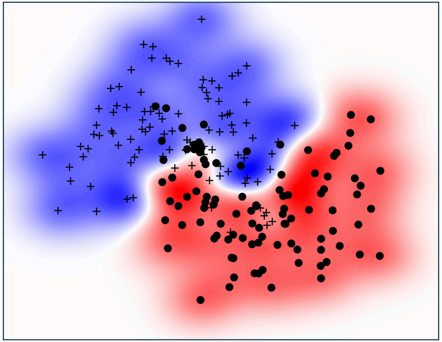
Agenda

- Kernel Methods
- Scalable Gaussian Processes (using Preconditioning)
- Probabilistic Numerics

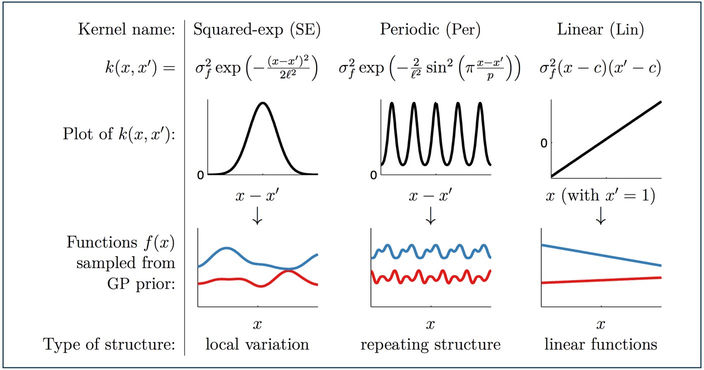
Kernel Methods

- Operate in a high-dimensional, implicit feature space
- Rely on the construction of an $n \times n$ Gram matrix K
- E.g. RBF : $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sigma^2 \exp\left(-\frac{1}{2} d^2\right)$

  where $d^2 = \left(\mathbf{x}_i - \mathbf{x}_j\right)^\top \Lambda \left(\mathbf{x}_i - \mathbf{x}_j\right)$

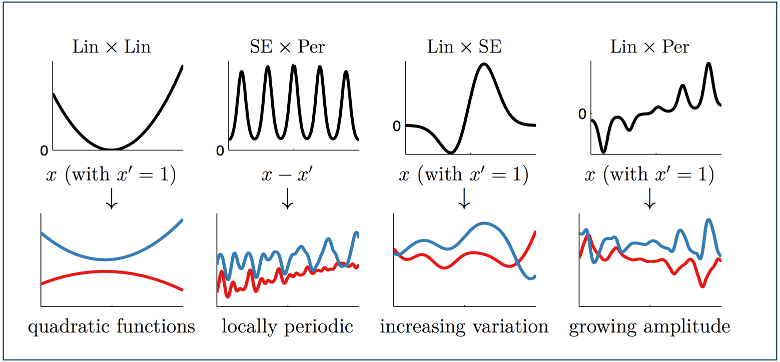


- Wide variety of kernel functions available



Taken from David Duvenaud's PhD Thesis

- Choice is not always straightforward!



Taken from David Duvenaud's PhD Thesis

All About that ~~Bass~~ Bayes

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}$$

$$p\left(\mathrm{par} \middle| X, \mathbf{y}\right) = \frac{p\left(\mathbf{y} \middle| X, \mathrm{par}\right) \times p\left(\mathrm{par}\right)}{p\left(\mathbf{y} \middle| X\right)}$$

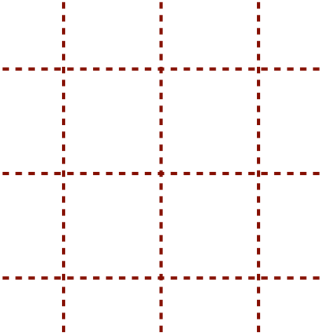
- We average over all possible parameter values, weighted by their posterior probability

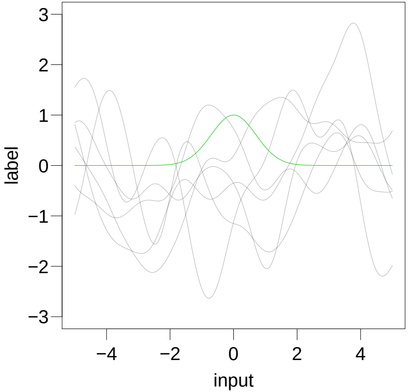
$$p\left(\mathbf{y}^*|\mathbf{x}^*, X, \mathbf{y}\right) = \int p\left(\mathbf{y}^*|\mathbf{x}^*, \mathrm{par}\right) p\left(\mathrm{par}|X, \mathbf{y}\right)\ \mathrm{d}\mathrm{par}$$
$$= \mathcal{N}\left(\mathbb{E}\left[\mathbf{y}^*\right], \mathbb{V}\left[\mathbf{y}^*\right]\right)$$

Gaussian Processes

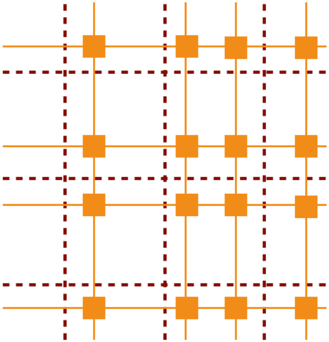
Gaussian Processes - Prior Distribution over Functions

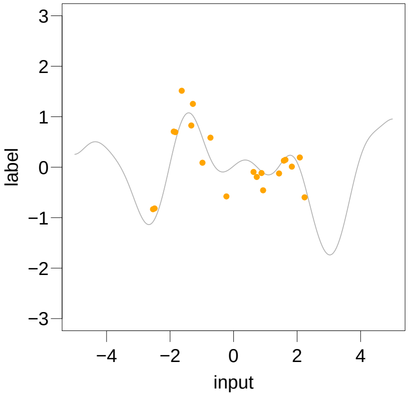
$K_\infty =$



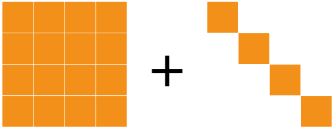


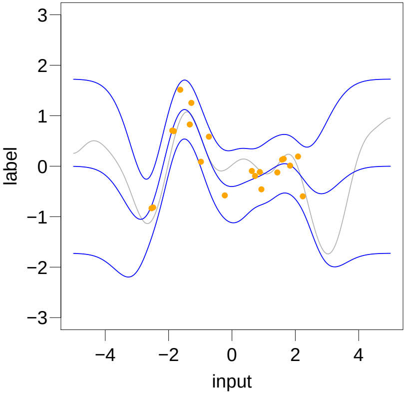
Gaussian Processes - Conditioned on Observations

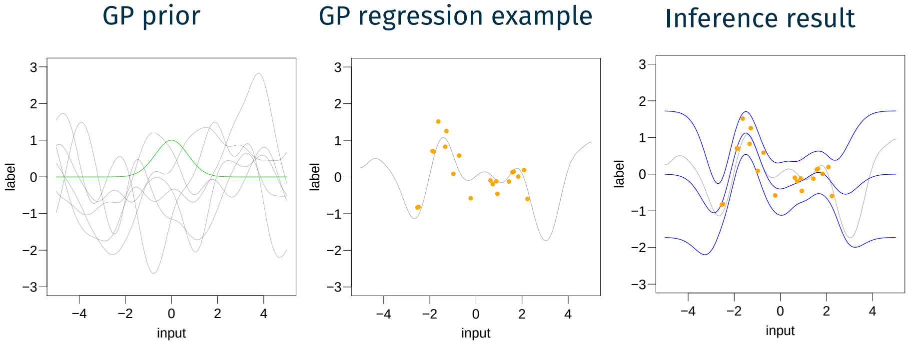
$K_\infty =$ 



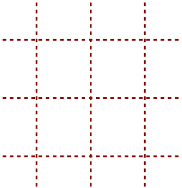
Gaussian Processes - Posterior Distribution over Functions

$K_{\mathbf{y}} =$ 

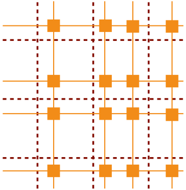




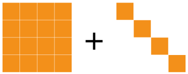
$K_\infty =$



$K_\infty =$



$K_y =$



Bayesian Learning vs Deep Learning

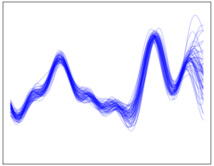
- Deep Learning
  - \+ Scalable to very large datasets
  - \+ Increased model flexibility/capacity
  - \- Frequentist approaches make only point estimates
  - \- Less robust to overfitting
- Bayesian Learning
  - \+ Incorporates uncertainty in predictions
  - \+ Works well with smaller datasets
  - \- Lack of conjugacy necessitates approximation
  - \- Expensive computational and storage requirements

Bayesian Learning vs Deep Learning - Deep Gaussian Processes

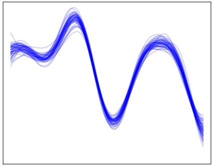
- Deep probabilistic models
- Composition of functions

$$\mathsf{f}(\mathsf{x}) = \left(\mathsf{h}^{(N_\mathrm{h}-1)}\left(\boldsymbol{\theta}^{(N_\mathrm{h}-1)}\right) \circ \ldots \circ \mathsf{h}^{(0)}\left(\boldsymbol{\theta}^{(0)}\right)\right)(\mathsf{x})$$

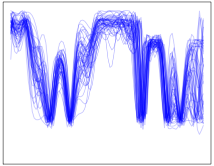
$\mathsf{h}^{(0)}(\mathsf{x})$



$\mathsf{h}^{(1)}(\mathsf{x})$



$\mathsf{h}^{(1)}\left(\mathsf{h}^{(0)}\left(\mathsf{x}\right)\right)$



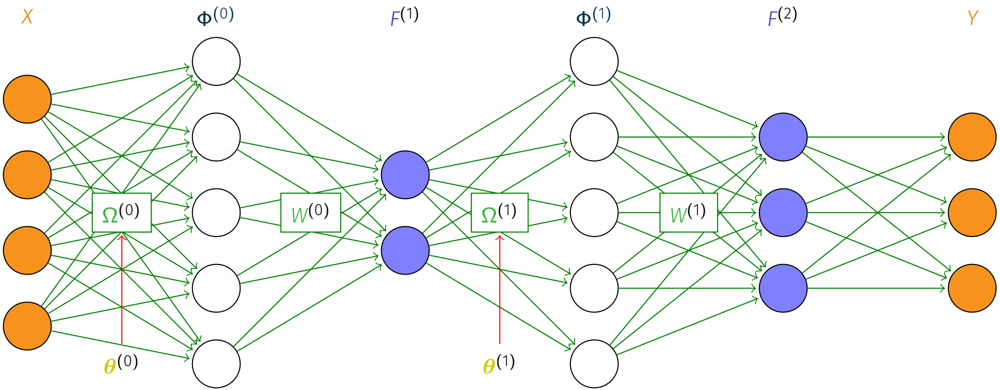
Bayesian Learning vs Deep Learning - Deep Gaussian Processes

- Inference requires calculating the marginal likelihood:

$$
\begin{aligned}
p(Y|X,\boldsymbol{\theta}) \quad = \quad & \int p\left(Y|F^{(N_\mathrm{h})},\boldsymbol{\theta}^{(N_\mathrm{h})}\right) \times \\
& p\left(F^{(N_\mathrm{h})}|F^{(N_\mathrm{h}-1)},\boldsymbol{\theta}^{(N_\mathrm{h}-1)}\right) \times \ldots \times \\
& p\left(F^{(1)}|X,\boldsymbol{\theta}^{(0)}\right) dF^{(N_\mathrm{h})} \ldots dF^{(1)}
\end{aligned}
$$

- Very challenging!

Bayesian Learning vs Deep Learning - Deep Gaussian Processes



Cutajar et al., *Random Feature Expansions for Deep Gaussian Processes*, ICML 2017
Yarin Gal, *Bayesian Deep Learning*, PhD Thesis

Scalable Gaussian Processes

Gaussian Processes

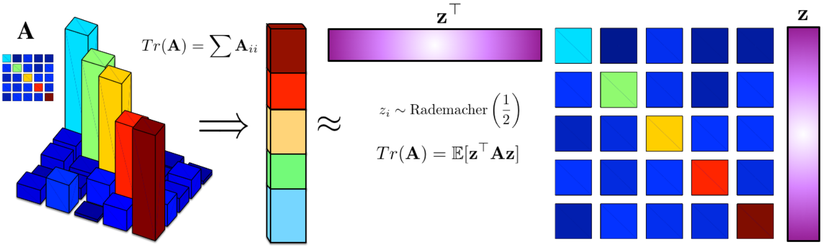
- Marginal likelihood

$$\log[p(\mathbf{y}|\mathrm{par})] = -\frac{1}{2}\log|K_{\mathbf{y}}| - \frac{1}{2}\mathbf{y}^{\mathrm{T}}K_{\mathbf{y}}^{-1}\mathbf{y} + \mathrm{const.}$$

- Derivatives wrt par

$$\frac{\partial\log[p(\mathbf{y}|\mathrm{par})]}{\partial\mathrm{par}_i} = -\frac{1}{2}\mathrm{Tr}\left(K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial\mathrm{par}_i}\right) + \frac{1}{2}\mathbf{y}^{\mathrm{T}}K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial\mathrm{par}_i}K_{\mathbf{y}}^{-1}\mathbf{y}$$

Gaussian Processes - Stochastic Trace Estimation



Taken from Shakir Mohamed's Machine Learning Blog

Gaussian Processes - Stochastic Gradients

- Stochastic estimate of the trace - assuming $\mathrm{E}[\mathbf{r}\mathbf{r}^{\mathrm{T}}] = I$, then

$$\mathrm{Tr}\left(K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial \mathrm{par}_i}\right) = \mathrm{Tr}\left(K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial \mathrm{par}_i}\mathrm{E}[\mathbf{r}\mathbf{r}^{\mathrm{T}}]\right) = \mathrm{E}\left[\mathbf{r}^{\mathrm{T}}K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial \mathrm{par}_i}\mathbf{r}\right]$$

- Stochastic gradient

$$-\frac{1}{2N_{\mathbf{r}}}\sum_{i=1}^{N_{\mathbf{r}}}\mathbf{r}^{(i)\mathrm{T}}K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial \mathrm{par}_i}\mathbf{r}^{(i)} + \frac{1}{2}\mathbf{y}^{\mathrm{T}}K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial \mathrm{par}_i}K_{\mathbf{y}}^{-1}\mathbf{y}$$

- Stochastic estimate of the trace - assuming $\mathrm{E}[\mathbf{r}\mathbf{r}^{\mathrm{T}}] = I$, then

$$\mathrm{Tr}\left(K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial \mathrm{par}_i}\right) = \mathrm{Tr}\left(K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial \mathrm{par}_i}\mathrm{E}[\mathbf{r}\mathbf{r}^{\mathrm{T}}]\right) = \mathrm{E}\left[\mathbf{r}^{\mathrm{T}}K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial \mathrm{par}_i}\mathbf{r}\right]$$

- Stochastic gradient

$$-\frac{1}{2N_{\mathbf{r}}}\sum_{i=1}^{N_{\mathbf{r}}}\mathbf{r}^{(i)\mathrm{T}}K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial \mathrm{par}_i}\mathbf{r}^{(i)} + \frac{1}{2}\mathbf{y}^{\mathrm{T}}K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial \mathrm{par}_i}K_{\mathbf{y}}^{-1}\mathbf{y}$$

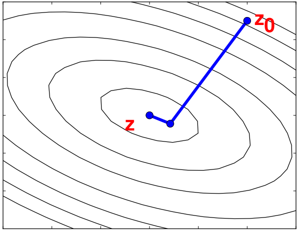
Linear systems only!

Solving Linear Systems

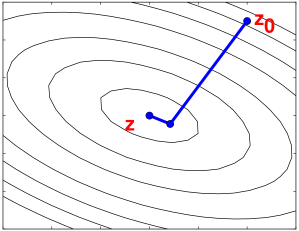
- Involve the solution of linear systems $K\mathbf{z} = \mathbf{v}$
- **Cholesky Decomposition**
    - K must be stored in memory!
    - $\mathcal{O}(n^2)$ space and $\mathcal{O}(n^3)$ time - unfeasible for large n

Solving Linear Systems

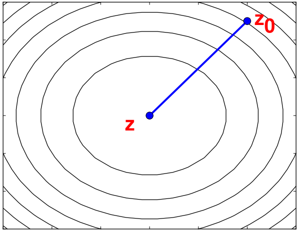
- Involve the solution of linear systems $K\mathbf{z} = \mathbf{v}$
- **Cholesky Decomposition**
  - K must be stored in memory!
  - $\mathcal{O}(n^2)$ space and $\mathcal{O}(n^3)$ time - unfeasible for large n
- **Conjugate Gradient**
  - Numerical solution of linear systems
  - $\mathcal{O}(tn^2)$ for t CG iterations - in theory $t = n$ (possibly worse!)



- **Preconditioned Conjugate Gradient** (henceforth **PCG**)
- Transforms linear system to be better conditioned, improving convergence
- Yields a new linear system of the form $P^{-1}K\mathbf{z} = P^{-1}\mathbf{v}$
- $\mathcal{O}(tn^2)$ for t PCG iterations - in practice $t \ll n$



CG

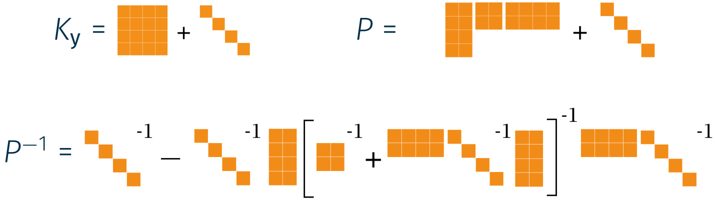


PCG

Preconditioning Approaches

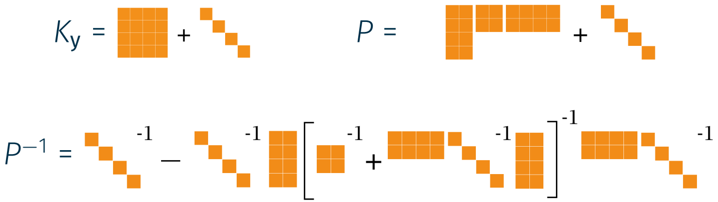
- Suppose we want to precondition $K_{\mathbf{y}} = K + \lambda I$
- Our choice of preconditioner, P, should:
  - Approximate $K_{\mathbf{y}}$ as closely as possible
  - Be easy to invert

- Suppose we want to precondition $K_{\mathbf{y}} = K + \lambda I$
- Our choice of preconditioner, P, should:
  - Approximate $K_{\mathbf{y}}$ as closely as possible
  - Be easy to invert
- For low-rank preconditioners we employ the Woodbury inversion lemma:



Preconditioning Approaches

- Suppose we want to precondition $K_{\mathbf{y}} = K + \lambda I$
- Our choice of preconditioner, P, should:
    - Approximate $K_{\mathbf{y}}$ as closely as possible
    - Be easy to invert
- For low-rank preconditioners we employ the Woodbury inversion lemma:

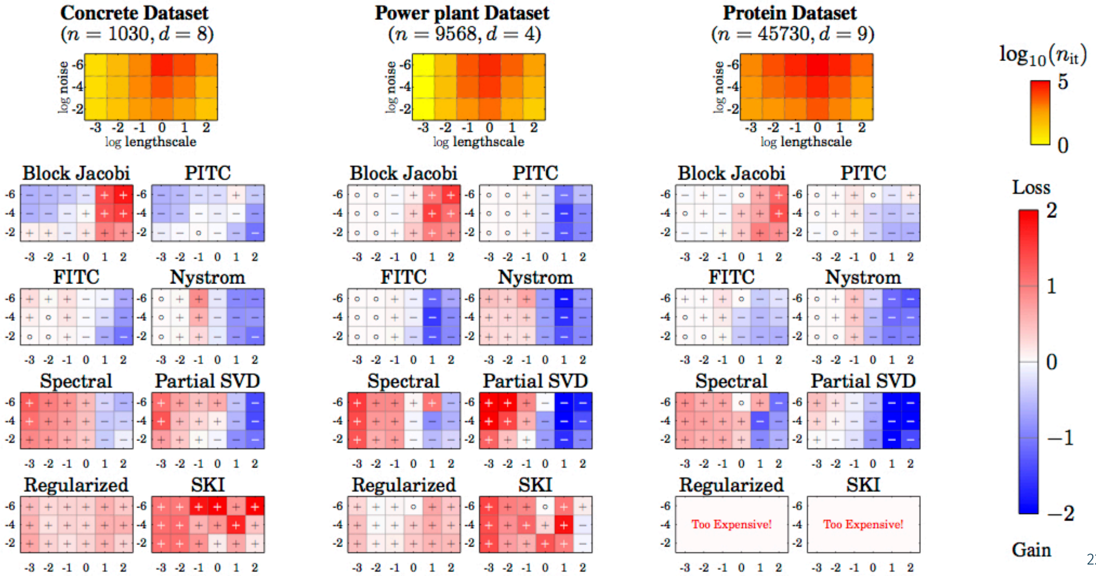


- For other preconditioners we solve inner linear systems once again using CG!

Preconditioning Approaches

Nyström	$P = K_{XU} K_{UU}^{-1} K_{UX} + \lambda I$ where $U \subset X$
FITC	$P = K_{XU} K_{UU}^{-1} K_{UX} + \mathrm{diag}\left(K - K_{XU} K_{UU}^{-1} K_{UX}\right) + \lambda I$
PITC	$P = K_{XU} K_{UU}^{-1} K_{UX} + \mathrm{bldiag}\left(K - K_{XU} K_{UU}^{-1} K_{UX}\right) + \lambda I$
Spectral	$P_{ij} = \frac{\sigma^2}{m} \sum_{r=1}^{m} \cos\left[2\pi \mathbf{s}_r^\top \left(\mathbf{x}_i - \mathbf{x}_j\right)\right] + \lambda I_{ij}$
Partial SVD	$K = A \Lambda A^\top \quad \Rightarrow \quad P = A_{[\cdot,1:m]} \Lambda_{[1:m,1:m]} A_{[1:m,\cdot]}^\top + \lambda I$
Block Jacobi	$P = \mathrm{bldiag}\left(K\right) + \lambda I$
SKI	$P = W K_{UU} W^\top + \lambda I$ where K_{UU} is Kronecker
Regularization	$P = K + \lambda I + \delta I$

Comparison of Preconditioners vs CG

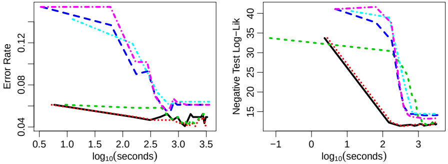


Experimental Setup - GP Kernel Parameter Optimization

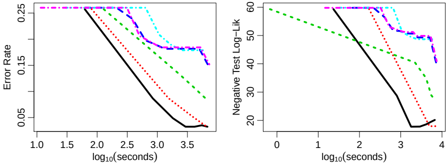
- Exact gradient-based optimization using **Cholesky** decomposition (CHOL)
- Stochastic gradient-based optimization
  - Linear systems solved with **CG** and **PCG**
- GP Approximations
  - Variational learning of inducing variables (**VAR**)
  - Fully Independent Training Conditional (**FITC**)
  - Partially Independent Training Conditional (**PITC**)

Classification

Spam (n = 4061, d=57)

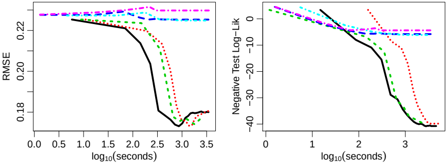


EEG (n = 14979, d=14)

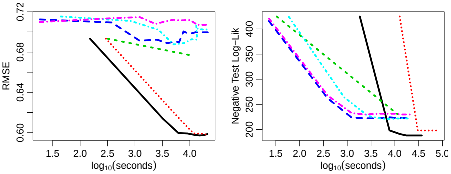


Regression

Power plant (n = 9568, d=4)



Protein (n = 45730, d=9)



PCG CG CHOL FITC PITC VAR

Follow-up Work

- Faster Kernel Ridge Regression Using Sketching and Preconditioning
  Avron et al. (2017)
- FALKON: An Optimal Large Scale Kernel Method
  Rosasco et al. (2017)
- Large Linear Multi-output Gaussian Process Learning for Time Series
  Feinberg et al. (2017)
- Scaling up the Automatic Statistician: Scalable Structure Discovery using Gaussian Processes
  Kim et al. (2017)

Follow-up work

Follow-up work

… but what's left to do now?

Probabilistic Numerics

Probabilistic Numerics - Mission Statement

We deliver a call to arms for *probabilistic numerical methods*: algorithms for numerical tasks, including linear algebra, integration, optimization and solving differential equations, that return uncertainties in their calculations.

Such uncertainties, arising from the loss of precision induced by numerical calculation with limited time or hardware, are important for much contemporary science and industry.

Hennig et al., 2015

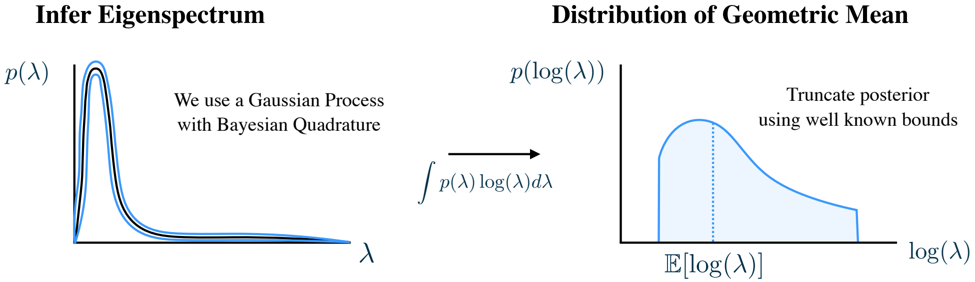
Probabilistic Numerics - Applications

- Quadrature
- Linear Algebra
- Optimization
- Ordinary Differential Equations
- Partial Differential Equations
- Monte Carlo Sampling

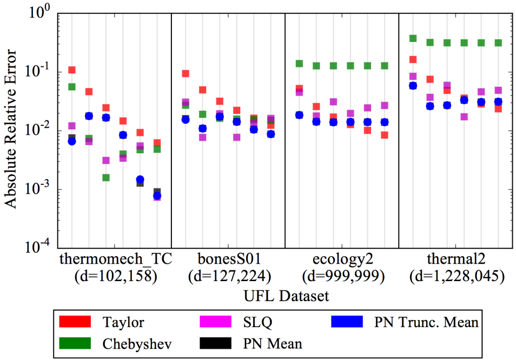
- Quadrature
- Linear Algebra
- Optimization
- Ordinary Differential Equations
- Partial Differential Equations
- Monte Carlo Sampling

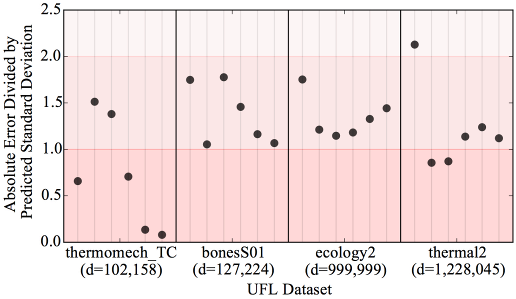
Bayesian Inference of Log Determinants

- Standard computation of $\log |A|$ is $\mathcal{O}(n^3)$
- Existing approximations do not give uncertainty estimates



Bayesian Inference of Log Determinants

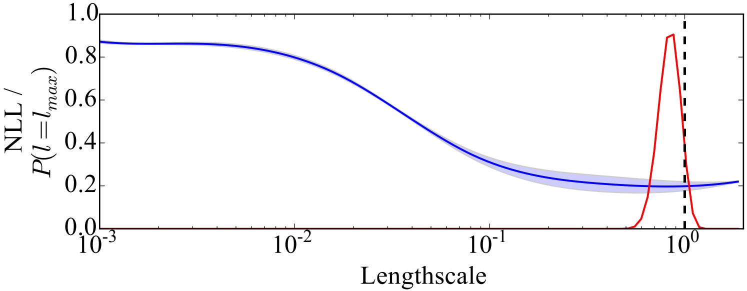




- Methods compared on a variety of UFL Sparse Datasets. For each dataset, the matrix was approximately raised up to the power of 5, 10, 15, 20, 25 and 30 (left to right) using stochastic trace estimation.

Bayesian Inference of Log Determinants - Application to DPPs

$$\log p\left(Z\right) = \log \frac{|L_J|}{|L+I|} \quad \forall J \subset X$$



Outlook

Can we include this computational uncertainty within the full pipeline of GP inference?

Outlook

Can we include this computational uncertainty within the full pipeline of GP inference?

- Better calibrated uncertainties
- Performance tunable to computational budget
- Applications to Bayesian optimisation

Other Work

- Preconditioning Kernel Matrices (ICML 2016)
  Kurt Cutajar, John Cunningham, Michael Osborne, Maurizio Filippone
- Bayesian Inference of Log Determinants (UAI 2017)
  Jack Fitzsimons, Kurt Cutajar, Michael Osborne, Stephen Roberts, Maurizio Filippone
- Random Feature Expansions for Deep Gaussian Processes (ICML 2017)
  Kurt Cutajar, Edwin V. Bonilla, Pietro Michiardi, Maurizio Filippone
- AutoGP: Exploring the capabilities and limitations of Gaussian processes (UAI 2017)
  Karl Krauth, Edwin V. Bonilla, Kurt Cutajar, Maurizio Filippone
- Entropic Trace Estimates for Log Determinants (ECML/PKDD 2017)
  Jack Fitzsimons, Diego Granziol, Kurt Cutajar, Maurizio Filippone, Michael Osborne, Stephen Roberts

Thank you!

Thank you!

https://github.com/mauriziofilippone

kurt.cutajar@eurecom.fr