Large Linear Gaussian Processes: Fast Multi-output GP Model Learning

PRINCETON UNIVERSITY

Vladimir Feinberg
Li-Fang Cheng
Barbara Engelhardt
Kai Li

Why multi-output GPs?

1) Create confidence intervals
2) Assign likelihoods
3) Generative sampling
4) Non-parametric modeling
5) Learn correlations between several outputs

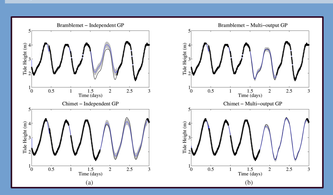
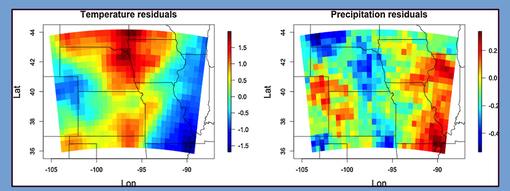


Figure 1 from [Genton and Kleiber 2015]. Authors depict residuals for average summer temperature and precipitation prediction for the year 1989, which are outputs whose regression is simultaneously learned in a multi-output GP climate model. Units in the left and right diagrams are degrees Celsius and centimeters, respectively.

Figure 3 from [Osborne 2008]. We see increase in accuracy from a single-output GP to a multi-output one in predicting tide heights over time from two sensors Bramblemet and Chimet on the south coast of England. The solid black line represents the actual held-out data, the dots represent the training data. The blue line is the expectation from the GP model and the grey shaded region is the 2 standard deviation error bar.



Goals and Scope

Only apply to *linearly coregionalization* (LMC)
Focus on time series (keep extension in mind)
Asymptotically improve learning runtime
Realize guarantees in practice
Apply to a wide range of differentiable kernels

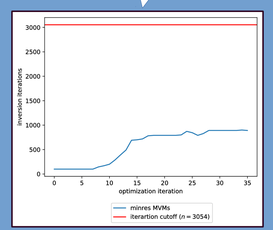
Multi-output GP Approaches

LMC Kernel

n input-output pairs
D outputs total
Q stationary kernels
Order-D matrices B_q of average rank R determine *linear cross-correlation* between all pairs of our D outputs
Use approximation of $m < n$ rank
An observation is an input $[d, \boldsymbol{x}]$, both output tag *d* (from 1 to D), and continuous, low dimensional $\boldsymbol{x}$ along with a corresponding real observation *y*.

$$K([i, \mathbf{x}_i], [j, \mathbf{x}_j]) = \sum_{q=1}^{Q} b_{ij}^{(q)} k_q(\|\mathbf{x}_i - \mathbf{x}_j\|) + \epsilon_i \delta_{ij}$$

* What is $\kappa_2^{1/2}$ in practice? Typically O(n), see (financial dataset) below.



Theoretical Complexity Comparison (∇L)
Note there are ~QRD parameters in an LMC model. Log factors hidden.

Method	Up-front Cost	Per-parameter
Exact GP	n^3	n^2
COGP [Nguyen and Bonilla 2014]*	QRm^3	nm
LLGP	$(n + \min(QR, D^2)\, m)\kappa_2^{1/2}$	n

How?

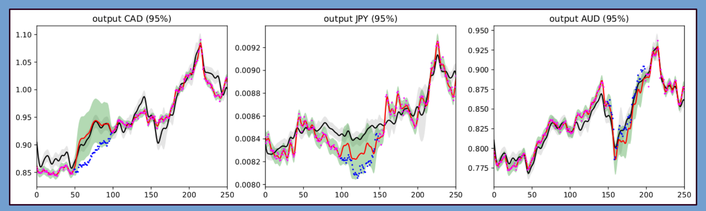
Use technique of Structured Kernel Interpolation from [Wilson et al 2015]
- Interpolate true input to a grid of *m* points for each D^2 output-output pair
- Use stationarity in stationary subkernels $k_q(\|x_{i+1} - x_{j+1}\|) = k_q(\|x_i + \Delta - x_j - \Delta\|) = k(\|x_i - x_j\|)$

Redundancy above gives rise to Toepliz matrix K_q on the grid.
Describe cross-covariances on interpolating grid U with Kronecker product!
Structured matrices → Fast Matrix Vector Products → Fast Inversions → Fast ∇L
Use MINRES and [Gibbs and Mackay 1996].

Results

Financial dataset D = 13, n = 3K, m = 238 = n / D
Indices for foreign exchange and commodities in 2007
Predict missing points (blue) from given ones (magenta)
LLGP (mean in red, confidence in green)
COGP (mean in black, confidence in grey)

Metric	LLGP	COGP
seconds	**64** (**8**)	296 (2)
SMSE	**0.21** (**0.01**)	0.26 (0.03)
NLPD	**-3.62** (**0.07**)	14.52 (3.10)



Weather dataset D = 4, n = 16K
Impute missing periods from temperature sensors on 4 points on the coast of England.

Metric	LLGP $m = 500$	LLGP $m = 1000$	COGP
seconds	**60** (**14**)	259 (62)	1380 (12)
SMSE	0.09 (0.01)	0.09 (0.01)	**0.08** (**0.00**)
NLPD	2.14 (0.58)	**1.54** (**0.03**)	98.48 (1.30)

Conclusions

Beat state-of-the art in time and accuracy by using a scalable approximation (only quadratic growth in up-front cost per gradient)

Tested on a variety of kernels for accuracy (not shown)

Future work: rigorous iteration cutoff, extend to multidimensional input.

GP Background

GP Definition [Williams and Rasmussen 1996]

Model parameters **Θ** parameterize the kernel *k*. The kernel *k* evaluated at input data points X induces a covariance matrix $K_{x,x}$ whose ij-th entry is $k(x_i, x_j)$. Then the regressor is a random variable determined by a normal with that covariance.

Use marginalization to predict distribution at query input x_*.

$$\begin{pmatrix} \mathbf{y} \\ y_* \end{pmatrix} \sim N\left(\mathbf{0}, \begin{pmatrix} K_{X,X} & K_{X,*} \\ K_{*,X} & K_{*,*} \end{pmatrix}\right)$$

prior → posterior

$$y_* | \mathbf{y} \sim N\left(K_{*,X}K_{X,X}^{-1}\mathbf{y}, K_{*,*} - K_{*,X}K_{X,X}^{-1}K_{X,*}\right)$$

Model learning task: pick **Θ** to maximize L (usually via gradient)

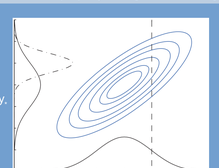
$$\mathcal{L}(\boldsymbol{\theta}) = \log p(\mathbf{f}_X | X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^\top K_{|\boldsymbol{\theta}}^{-1}\mathbf{y} - \frac{1}{2}\log\left|K_{|\boldsymbol{\theta}}\right| - \frac{n}{2}\log 2\pi$$

$$\partial_{\theta_j}\mathcal{L} = \frac{1}{2}\boldsymbol{\alpha}^\top \partial_{\theta_j} K_{|\boldsymbol{\theta}}\boldsymbol{\alpha} - \frac{1}{2}\operatorname{tr}\left(K_{|\boldsymbol{\theta}}^{-1}\partial_{\theta_j}K_{|\boldsymbol{\theta}}\right)$$

$$\boldsymbol{\alpha} \triangleq K_{|\boldsymbol{\theta}}^{-1}\mathbf{y}$$

$$\mathbf{y} \sim N(\mathbf{0}, K_{X,X})$$

Demo of GP property from [Osborne 2008]

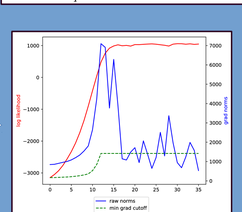


$$K_{|\boldsymbol{\theta}}\mathbf{z} \approx \tilde{K}_{|\boldsymbol{\theta}}\mathbf{z}$$

Gradient cutoff as a function of rolling gradient norm mean appropriate for maximization of L.

Demo on financial dataset.

$$K_{\mathbf{U},\mathbf{U}} = \sum_q (A_q A_q^\top + \operatorname{diag}\boldsymbol{\kappa}_q) \otimes K_q$$



Algorithm 1 The content corresponding to each line is detailed below. $\mathbf{X} \subset [D] \times \mathbb{R}$ is the set of inputs, tagged by the output that they coresond to, and limited for simplicity to the one-dimensional case. Then $\mathbf{y} \subset \mathbb{R}$ is the corresponding set of outputs. AdaDeltaUpdate is the update function for AdaDelta—we abstract away auxiliary AdaDelta variables.

1: **procedure** LLGP($\alpha, \mathbf{X}, \mathbf{y}$)
2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$ ▷ Initialization
3: $\quad g_{\max} = -\infty$
4: $\quad$**repeat**
5: $\quad\quad$Construct a linear operator $\hat{K}_{|\boldsymbol{\theta}}$ from $\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}$. ▷ MVM operator
6: $\quad\quad g \leftarrow \nabla_{\boldsymbol{\theta}}\mathcal{L}_{\hat{K}_{|\boldsymbol{\theta}}}$ ▷ Gradients from an operator, Algorithm 2
7: $\quad\quad \boldsymbol{\theta} \leftarrow \text{AdaDeltaUpdate}(\boldsymbol{\theta}, g)$
8: $\quad\quad g_{\max} = \max(\|g\|, \|g_{\max}\|)$
9: $\quad$**until** $\|g\| \le \alpha g_{\max}$ ▷ Cutoff
10: $\quad$**return** $\boldsymbol{\theta}$
11: **end procedure**

Algorithm 2 Given a linear MVM operator for a kernel, we compute the gradient of its data log likelihood. N_t is a fixed parameter for our stochastic trace approximation. The output of this procedure is $\nabla\mathcal{L}$. MINRES$(K, \mathbf{z})$ computes $K^{-1}\mathbf{z}$.

1: **procedure** Gradient($K, \mathbf{y}$)
2: $\quad R \leftarrow \{\mathbf{r}_i\}_{i=1}^{N_t}$, sampling $\mathbf{r} \sim \text{Unif}\{\pm 1\}$.
3: $\quad$**for** $\mathbf{z}$ in $\{\mathbf{y}\} \cup R$, in parallel **do**
4: $\quad\quad$Store the result of MINRES$(K, \mathbf{z})$ as $K^{-1}\mathbf{z}$.
5: $\quad$**end for**
6: $\quad$Let $\boldsymbol{\alpha} = K^{-1}\mathbf{y}$; computed before.
7: $\quad g \leftarrow \mathbf{0}$
8: $\quad$**for** θ_j in $\boldsymbol{\theta}$ **do** ▷ Compute $\partial_{\theta_j}\mathcal{L}$
9: $\quad\quad$Create the operator $L = \partial_{\theta_j} K_{|\boldsymbol{\theta}}$
10: $\quad\quad \{\mathbf{s}_i\}_{i=1}^{N_t} \leftarrow \{L(\mathbf{r}_i)\}_{i=1}^{N_t}$
11: $\quad\quad t \leftarrow \frac{1}{N_t}\sum_{i=1}^{N_t}\left(K_{|\boldsymbol{\theta}}^{-1}\mathbf{r}_i\right) \cdot \mathbf{s}_i$
12: $\quad\quad r \leftarrow \boldsymbol{\alpha} \cdot L(\boldsymbol{\alpha})$
13: $\quad\quad g_j \leftarrow \frac{1}{2}r - \frac{1}{2}t$
14: $\quad$**end for**
15: $\quad$**return** g
16: **end procedure**