STAT G8325
Gaussian Processes and Kernel Methods
Lecture Notes §06: Speed and Scaling Part 2

John P. Cunningham

Department of Statistics
Columbia University

Outline

Administrative interlude

Practical realities of kernel methods

Reminder: conjugate gradients

Generic special structures

Kronecker structures in multiple dimensions [GSC15]

Markov structured gp and SDEs

Model selection for structured gp

Outline

Progress...

Week	Lectures	Content
5	Oct 12,14	Speed and scaling part 1: reduced-rank processes
6	Oct 19, 21	Speed and scaling part 2: special structure • [GSC15]; [RW06, ch 4.3.2]
7	Oct 21, 26	Bayesian optimization and active learning • [SLA12]; [GSW$^+$15]; [HHGL11]
8		Special GP topics: dynamical systems, quadrature, ode solvers, etc.

- I have met or scheduled with almost everyone.
- If we are not scheduled, email me immediately after class today.
- HW3 to be posted, but will be outlining your project.

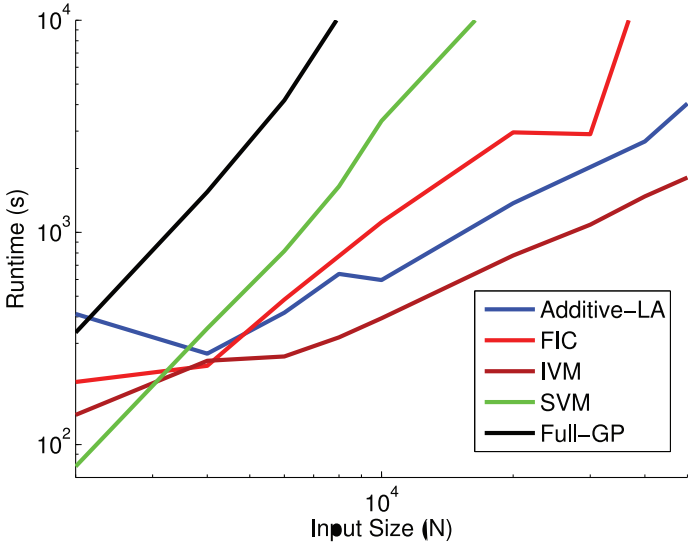
Outline

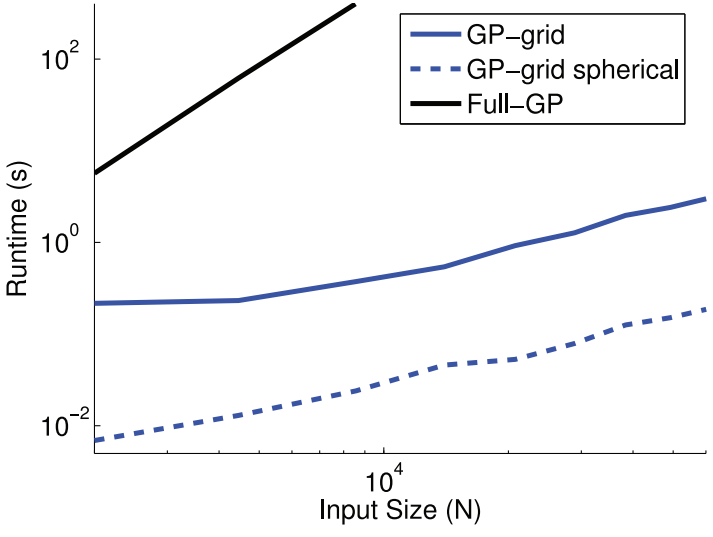
Fundamental fact about nonparametric techniques

- The number of parameters grows with the amount of data.
- In gp (and mostly in kernel methods):

$$y^*|y \sim \mathcal{N}\left(K_{y^*y}K_{yy}^{-1}(y - m_y) \quad , \quad K_{y^*y^*} - K_{y^*y}K_{yy}^{-1}K_{y^*y}^{\top}\right)$$

- Storing and inverting $K_{yy} \in \mathbb{R}^{n \times n}$ costs $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$.
- Practically speaking, these operations become impossible fast:





Outline

Solving linear systems

- GP inference boils down to solving a few linear systems in $K_{yy} \in \mathbb{S}_{++}$:

$$y^*|y \sim \mathcal{N}\left(K_{y^*y}K_{yy}^{-1}y \quad , \quad K_{y^*y^*} - K_{y^*y}K_{yy}^{-1}K_{y^*y}^\top\right)$$

- There are four ways to solve $K^{-1}y$ (equivalently, $K^{-1}k(X, x^*)$):
  - invert K and store $K^{-1} \rightarrow \quad \mathcal{O}(n^3)$... almost always the wrong choice.
  - directly factorize full K (e.g., cholesky) and solve $\rightarrow \quad \mathcal{O}(n^3)$
  - (directly factorize if K sparse and solve $\rightarrow \quad < \mathcal{O}(n^3)$)
  - iteratively solve if K has special structure $\rightarrow \quad \ll \mathcal{O}(n^3)$.
- Iterative solutions typically rely on the conjugate gradients method.
- Essential idea: Note that $K^{-1}y$ is the solution to:

$$\arg\min_z \phi(z) = \frac{1}{2}z^\top Kz - y^\top z + w.$$

- with gradient $\nabla_z \phi = Kz - y = 0 \rightarrow z = K^{-1}y$.
- Gradient steps ($z^{(r+1)} = z^{(r)} - \alpha^{(r)}\nabla_z\phi$) involve *forward* multiplications Kz.

Conjugate gradients

- Gradient descent is very slow.
- Newton's method is fast but requires Hessian inversion.

Why is Newton's nonsensical here?

- Conjugate gradient takes better gradient steps in ϕ by noticing:

$$\begin{aligned}\phi(z) - \phi(z^*) &= \frac{1}{2} z^\top K z - y^\top z + w - \frac{1}{2} z^{*\top} K z^* + y^\top z^* - w \\ &= \frac{1}{2}(z - z^*)^\top K (z - z^*) \\ &= \frac{1}{2} ||z - z^*||_K^2,\end{aligned}$$

...second line follows from $y = K z^*$.

- and thus takes "conjugate" gradient steps in this Mahalanobis distance.
- Further details are rather dense, but what results is a black box...

Conjugate gradients

Algorithm 1 The celebrated conjugate gradients (simplest form)

Input: $K \succ 0, y$, initial value z
$r = y - Kz$
$v = r$
$\delta_{new} = r^\top r$
while not converged
$q = Kv$
$\alpha = \frac{\delta_{new}}{v^\top q}$
    $z = z + \alpha v$
    $r = r - \alpha q$
    $\delta_{old} = \delta_{new}$
    $\delta_{new} = r^\top r$
    $v = r + \frac{\delta_{new}}{\delta_{old}} v$

...from [She94].

- Very important: only *forward* multiplications $q = Kv$!

Conjugate gradients

- Very important: only *forward* multiplications $q = Kv$!
- In theory this converges in n steps for $K \in \mathbb{R}^{n \times n} \to \mathcal{O}(n^3)$

Why?

- However, CG often converges in much fewer than n steps:
  - condition number of matrix $\kappa(K)$,
  - number of distinct eigenvalues (or clustered),
  - tolerance of convergence.
- More important: if K has special structure such that products Kv are fast...
- Then total cost of $K^{-1}y$ can be $\mathcal{O}(n^2), \mathcal{O}(n \log n), ...$
- CG is one fundamental approach to scaling gp with special structure.

Outline

Common special structures that exist

- Generally, special structure in K can take on many forms:
  - sparse ...random sparsity is very uncommon in kernel methods
  - banded ...think triangle kernel (not square)
  - low rank plus diagonal ....think degenerate kernels
  - RBF ...surprisingly enough
  - Toeplitz ...think evenly spaced inputs
- Note: these are kernel choices and/or input point choices.
- Then, the actual cost of a forward multiply Kv is:
  - (sparse: $\#\{(i,j) : K_{ij} \neq 0\}$)
  - banded: $\mathcal{O}(nk)$
  - low rank plus diagonal: $\mathcal{O}(nm)$ (cf. inducing points!)
  - RBF: (complicated, uses multipole methods)
  - Toeplitz ...let's discuss this very common case
- CG is general but sometimes overkill.

...e.g. rank one plus diagonal $\rightarrow$ Woodbury formula is easier.

Toeplitz matrices

- *Toeplitz* $\rightarrow$ hugely important and well studied matrices of the form:

$$K = \begin{bmatrix} k(0) & k(-1) & k(-2) & \dots & k(-(n-1)) \\ k(1) & k(0) & k(-1) & & k(-(n-2)) \\ k(2) & k(1) & k(0) & & \vdots \\ \vdots & & & \ddots & \\ k(n-1) & & & & k(0) \end{bmatrix}$$

- Colloquially, Toeplitz matrices are striped (equal along diagonals).
- What would a gp need to have a Toeplitz K?
- ...a stationary kernel k, and evenly spaced input points $x_1, ..., x_n \in \mathbb{R}$.
- Is this limited? ...yes, but time series with uniform measurements abound.

Exploiting Toeplitz structure

$$K = \begin{bmatrix} k(0) & k(-1) & k(-2) & \dots & k(-(n-1)) \\ k(1) & k(0) & k(-1) & & k(-(n-2)) \\ k(2) & k(1) & k(0) & & \vdots \\ \vdots & & & \ddots & \\ k(n-1) & & & & k(0) \end{bmatrix}$$

- Note K has only $\mathcal{O}(n)$ distinct elements $\rightarrow$ storage $\ll \mathcal{O}(n^2)$.
- By symmetry we know $K_{ij} = k(i-j) = k(j-i) = K_{ji}$, so we have a vector $k \in \mathbb{R}^n$ such that:

$$K = \begin{bmatrix} k_0 & k_1 & k_2 & \dots & k_{n-1} \\ k_1 & k_0 & k_1 & & k_{n-2} \\ k_2 & k_1 & k_0 & & \vdots \\ \vdots & & & \ddots & \\ k_{n-1} & & & & k_0 \end{bmatrix}$$

- But what about runtime?

Exploiting Toeplitz structure

$$K = \begin{bmatrix} k_0 & k_1 & k_2 & \dots & k_{n-1} \\ k_1 & k_0 & k_1 & & k_{n-2} \\ k_2 & k_1 & k_0 & & \vdots \\ \vdots & & & \ddots & \\ k_{n-1} & & & & k_0 \end{bmatrix}$$

- Notice K above looks almost like a (discrete) convolution:

$$(c * v)_j = \sum_{i=-n}^{n} c_{j-i} v_i$$

- But without the wraparound... $K_{i,1} \neq K_{i-1,n-1}$, aka $k_{n-i} \neq k_i$.
- Idea: embed K into a circulant matrix C...

Exploiting Toeplitz structure

- Idea: embed K into a circulant matrix C...

$$C = \begin{bmatrix} K & 0 & \text{flip}(K) \\ 0 & 0 & 0 \\ \text{flip}(K) & 0 & K \end{bmatrix},$$

- where C is determined by the first row:

$$\begin{bmatrix} k & 0 & \text{flip}(k) \end{bmatrix} = \begin{bmatrix} k_0 & k_1 & \dots & k_{n-1} & \dots & 0 & \dots & k_{n-1} & \dots & k_2 & k_1 \end{bmatrix}$$

- $C \in \mathbb{R}^{(2n-1)\times(2n-1)}$ is a *circulant* matrix.
- Now we do have a discrete convolution:

$$\begin{bmatrix} Kv \\ (\text{junk}) \end{bmatrix} \quad = \quad C \begin{bmatrix} v \\ 0 \end{bmatrix} \quad = \quad Cv' \quad = \quad (c * v')_j = \sum_{i=-n}^{n} c_{j-i} v'_i.$$

- Thus, $Kv =$ the first n elements of $Cv' = \mathcal{F}^{-1}\left(\mathcal{F}(c) \cdot \mathcal{F}(v')\right)$.
- ...which is $\mathcal{O}(n \log n)$ time, by using the fast fourier transform!
- In practice clever zero padding (0 above) can greatly improve the scaling.

Exploiting Toeplitz structure... popping back up the stack

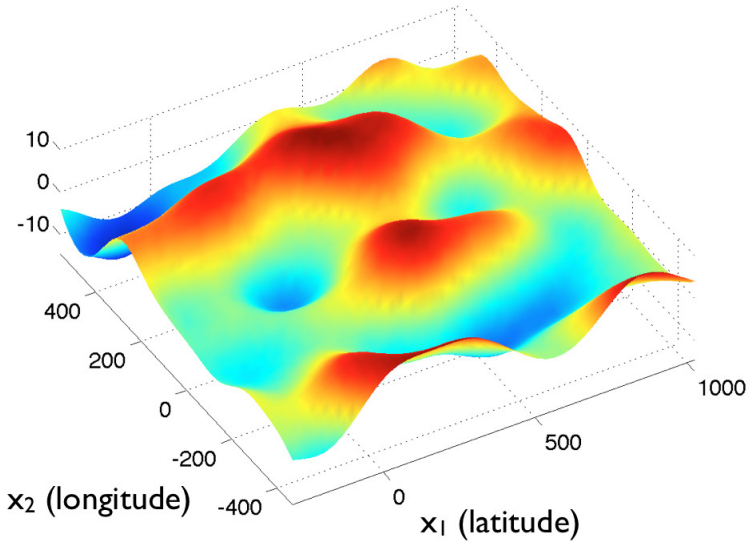
- CG turns $K^{-1}y$ into an often small number of products Kv (say m of them).
- Toeplitz structure reduces the inherent complexity of K to a vector k.
- Embedding K into a circulant matrix C makes Kv (part of) a convolution.
- The fft accomplishes this product in $\mathcal{O}(n \log n)$ time.
- Result:
  - Storage complexity: $\mathcal{O}(n^2) \to \mathcal{O}(n)$.
  - Runtime complexity: $\mathcal{O}(n^3) \to \mathcal{O}(mn \log n)$.
- Note: this method is **exact** (cf §05)... and widely used [CSS08].

Outline

GP in multiple dimensions

- Recall multidimensional gp and their common kernels:

$$k = \sigma_f^2 \exp\left\{ -\sum_{d=1}^{D} \frac{1}{2\ell_d^2} (t_i^d - t_j^d)^2 \right\}$$



- Notice this is a product structure $k = \prod_d k_d$.

Implied Kronecker structure

- When all points X lie on a grid, product kernels factorize:

$$k = \prod_d k_d \quad \leftrightarrow \quad K = K_1 \otimes K_2 \otimes ... \otimes K_D \quad \leftrightarrow \quad K = \otimes_d K_d.$$

- Say each dimension of the grid has $m = n^{\frac{1}{D}}$ points.
- Then $n = 10^6$ points is only three matrices $K_d \in \mathbb{R}^{100 \times 100}$.
- So what? (aka some facts about Kronecker matrices):
  - $K^{-1} = \otimes_d K_d^{-1}$.
  - $K = Q \Lambda Q^\top \quad \leftrightarrow \quad Q = \otimes_d Q_d$ (and same for Λ).
  - and others...

Implied Kronecker structure

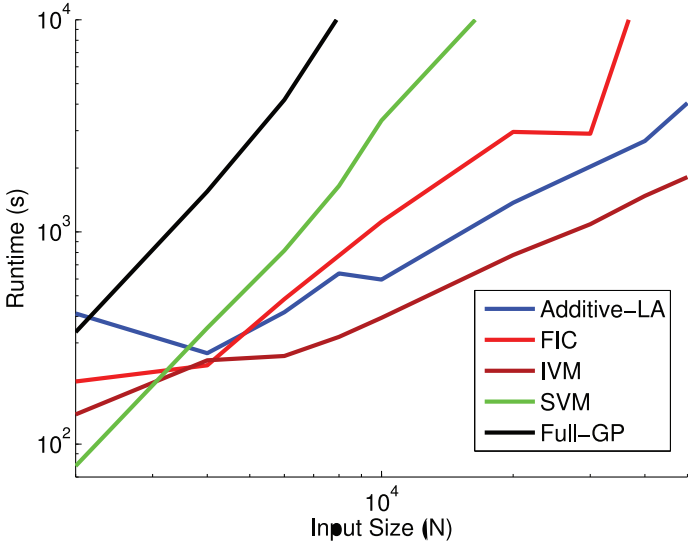
- Most importantly (the "Kronecker trick"):

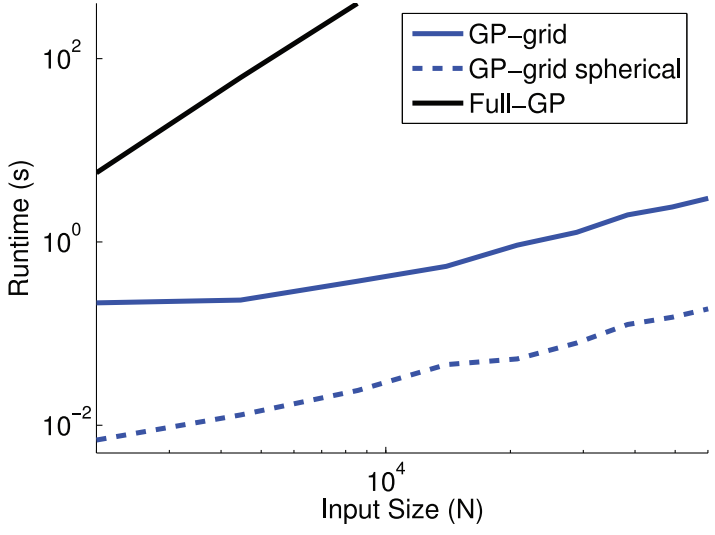
$$\begin{aligned} Kv &= \left(\otimes_d K_d\right) v \\ &= \text{vec}\left(K_D V \left(\otimes_{d=1}^{D-1} K_d\right)^\top\right) \\ &= \text{vec}\left(\left[K_1 \ldots \left[A_{D-1}\left[K_D B V\right]^\top\right]^\top\right]^\top\right). \end{aligned}$$

- Here $V \in \mathbb{R}^{n^{\frac{1}{D}} \times n^{D-1}{}_D}$ is such that $v = \text{vec}(V)$,
- ...and $[KV]^\top = \text{reshape}\left((KV)^\top\right)$.
- Result:
  - Storage complexity: $\mathcal{O}(n^2) \to \mathcal{O}(Dn^2 D)$.
  - Runtime complexity: $\mathcal{O}(n^3) \to \mathcal{O}(mn^{D+1} D)$.
- Again this method is **exact**! (cf §05)

Empirical results

- Reminder: gridded data are not unusual (images, movies...)
- Empirical results of this grid kronecker method:





- Note some additional fun is had for an incomplete grid; see [GSC15].

Outline

Some things we know

- Many of us will have seen discrete time linear dynamical systems:

$$\begin{aligned} f_{t+1} &= Af_t + \xi_t \quad \text{where} \quad f_t \in \mathbb{R}^p \;,\; \xi_t \sim \mathcal{N}(0, \Xi). \\ y_t &= Cf_t + \epsilon_t \quad \text{where} \quad y_t \in \mathbb{R}^q \;,\; \epsilon_t \sim \mathcal{N}(0, \Psi). \end{aligned}$$

- for $t = 1, ..., n$. To make this feel like a gp, set $p = q = 1, C = I$.
- The point: this has simplifying Markov structure.
- Kalman filter (message passing, etc.) infers $p(f|y)$ in $\mathcal{O}(n) \ll \mathcal{O}(n^3)$.
- ...but this model is all jointly linear gaussian, so it is (almost) a gp.
- The continous time (one-dimensional) case:

$$\frac{df(t)}{dt} = \alpha f_t + \epsilon_t.$$

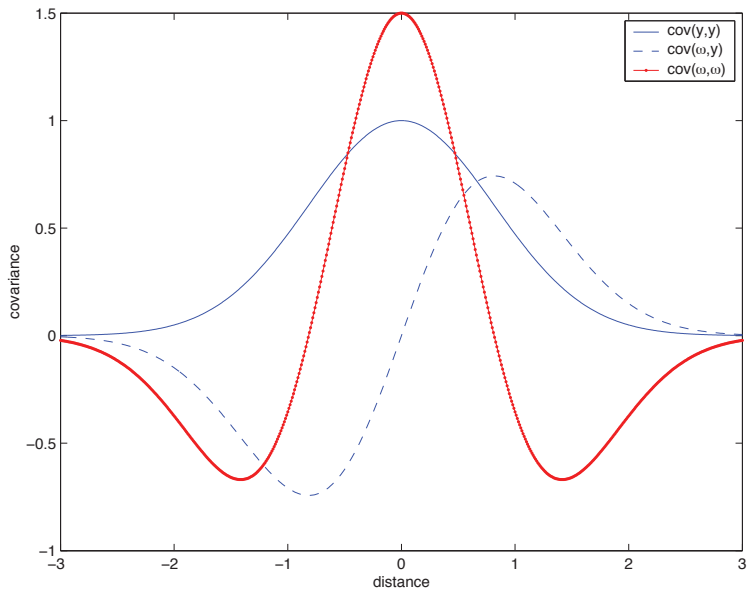
- This is a gp...

Derivatives of gp

- Differentiation is a linear operator: $\frac{d}{dt}(af(t) + bg(t)) = a\frac{d}{dt}f(t) + b\frac{d}{dt}g(t)$.
- Thus $f'(t) = \frac{d}{dt}f(t)$ is also gaussian $\mathcal{N}(0, \frac{d^2}{dt^2}k(t,t))$, leading to:

$$E\left(f'f'^{\top}\right) = E\left(\frac{d}{dt}f\left(\frac{d}{dt}f\right)^{\top}\right) = \frac{d}{dt}K\frac{d}{dt}^{\top}$$

- Thus f' is also a gp, and is jointly gaussian with f. Consider SE kernel:



[SMSL+03]

Gauss-markov processes

- Using this joint gaussianity, we return to our lds:

$$\frac{df(t)}{dt} = \alpha f_t + \epsilon_t.$$

- or more generally, the m^{th} *order gauss-markov process*:

$$\frac{d^m f(t)}{dt^m} + \alpha_{m-1}\frac{d^{m-1} f(t)}{dt^{m-1}} + \ldots + \alpha_1 \frac{df(t)}{dt} + \alpha_0 f_t = \epsilon_t.$$

- Here ϵ_t is a white noise gp (i.e. $k_\epsilon = \epsilon\delta(t_i, t_j)$).
- Taking the fourier transform:

$$\sum_{k=0}^{m} \alpha_k \frac{d^k f(t)}{dt^k} = \epsilon_t. \quad \Leftrightarrow \quad \sum_{k=0}^{m} \alpha_k (2\pi i\omega)^k F(\omega) = \epsilon$$

- Thus, f has with fourier transform:

$$F(\omega) = \frac{\epsilon}{\sum_{k=0}^{m} \alpha_k (2\pi i\omega)^k}$$

Gauss-markov processes

- Thus, f has with fourier transform:

$$F(\omega) = \frac{\epsilon}{\sum_{k=0}^{m} \alpha_k (2\pi i \omega)^k}$$

- Put differently, the draw $f(t)$ is now a filtered white noise draw:

$$f(t) = \epsilon(t) * h(t) \qquad \text{where} \qquad h(t) = \mathcal{F}^{-1}\left(\frac{1}{\sum_{k=0}^{m} \alpha_k (2\pi i \omega)^k}\right)(t),$$

- which implies that $f \sim \mathcal{GP}(0, k)$, with:

$$k(\tau) = h(\tau) * k_\epsilon(\tau) * h(-\tau) \qquad \Leftrightarrow \qquad S(\omega) = \frac{\epsilon}{\left|\sum_{k=0}^{m} \alpha_k (2\pi i \omega)^k\right|^2}$$

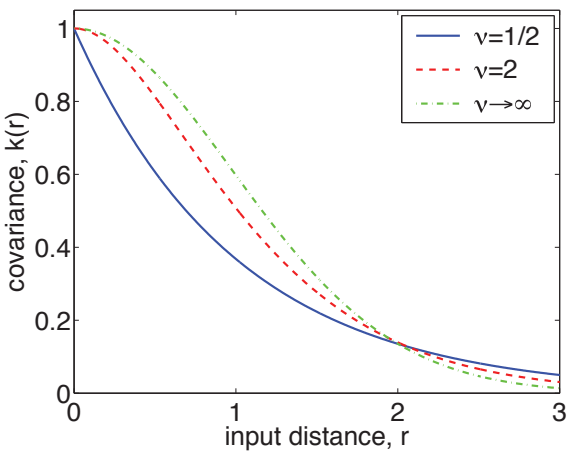
- Thus, any kernel with $S(\omega)$ (above) corresponds to a markov gp!
- So what: we can use message passing to infer $p(f|y)$ in linear time...

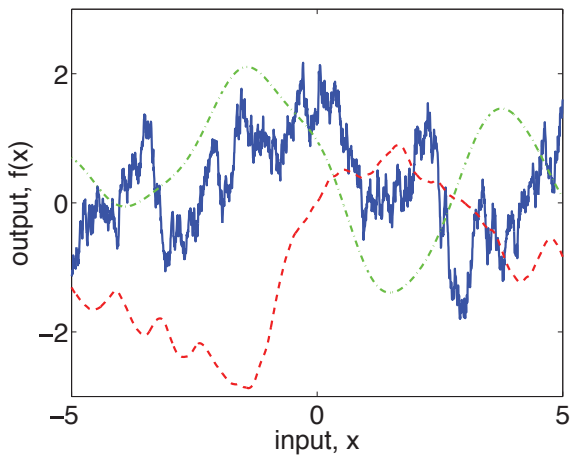
Generality of gauss-markov processes

- Remember Matérn kernels:

$$k(r) = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{\ell} \right)^{\nu} B_\nu \left(\frac{\sqrt{2\nu} r}{\ell} \right)$$

- $\nu, \ell > 0$, modified Bessel function B_ν.





- The spectral density of a Matérn kernel:

$$S(\omega) = \frac{\sigma_f^2 2\sqrt{\pi}\Gamma(\nu + \frac{1}{2})(2\nu)^\nu}{\Gamma(\nu)\ell^{2\nu}} \frac{1}{\left(\frac{2\nu}{\ell^2} + \omega^2\right)^{\nu+\frac{1}{2}}}.$$

- which is a constant times an inverse squared polynomial $\rightarrow$ a gmp!

Generality of gauss-markov processes

- The spectral density of a Matérn kernel:

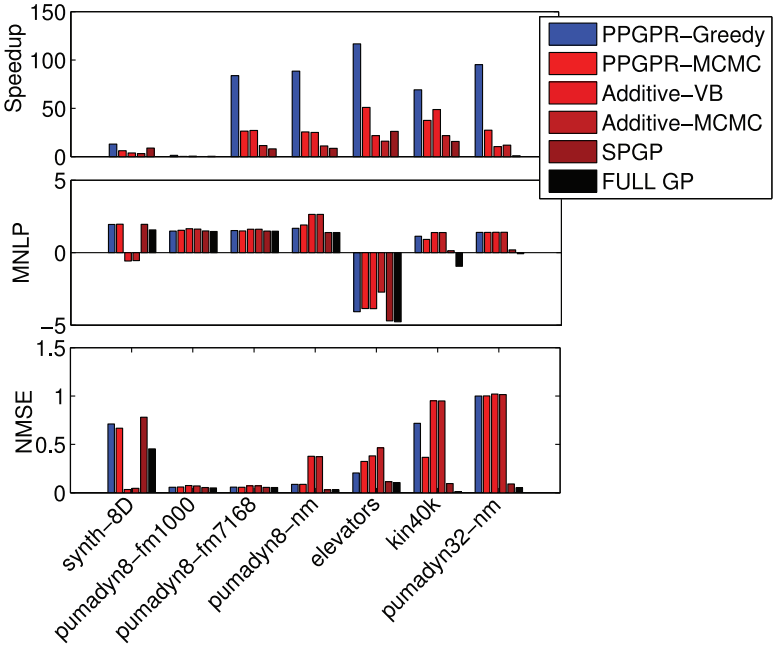
$$\begin{aligned} S(\omega) &= \frac{\sigma_f^2 2\sqrt{\pi}\Gamma(\nu+\frac{1}{2})(2\nu)^\nu}{\Gamma(\nu)\ell^{2\nu}} \frac{1}{\left(\frac{2\nu}{\ell^2}+\omega^2\right)^{\nu+\frac{1}{2}}}. \\ &= \epsilon \frac{1}{\left(\frac{\sqrt{2\nu}}{\ell}+i\omega\right)^{-(\nu+\frac{1}{2})}\left(\frac{\sqrt{2\nu}}{\ell}-i\omega\right)^{-(\nu+\frac{1}{2})}} \\ &= \epsilon \frac{1}{H(\omega)H(\omega)^*} \end{aligned}$$

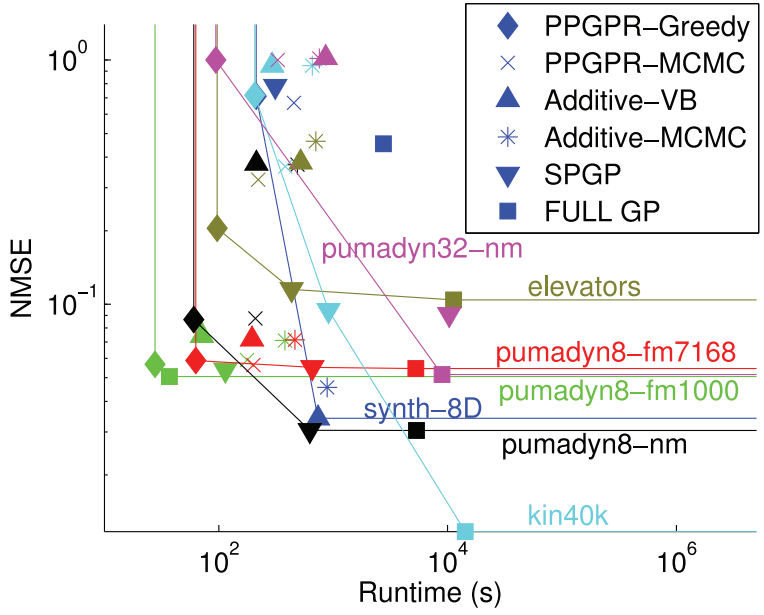
- from this for of $H(\omega)$ one can extract the values α_k in the original gmp.
- $\mathcal{O}(n)$ approximations result from m^{th} order gmp approximation kernels...

more on this, and a SE example in [Saa12].

Empirical results

- Empirical results of this gmp approximation method:





- Note some extra steps with backfitting for additive kernels; see [GSC15].

Outline

Model selection

- Note we have exclusively focused on $K^{-1}y$:

$$y^*|y \sim \mathcal{N}\left(K_{y^*y}K_{yy}^{-1}(y-m_y) \quad , \quad K_{y^*y^*} - K_{y^*y}K_{yy}^{-1}K_{y^*y}^\top\right)$$

- What about model selection? Recall:

$$\log\left(p(y|\theta)\right) = -\frac{1}{2}(y-m)^\top K_\theta^{-1}(y-m) - \frac{1}{2}\log|K_\theta| - \frac{n}{2}\log(2\pi).$$

- We haven't dealt with $\frac{1}{2}\log|K_\theta|$ (nor its gradients).
- Many interesting log determinant approximations. See [GSC15].
- (also an interesting project; see project idea sheet for further references).

References

[CSS08] John P Cunningham, Krishna V Shenoy, and Maneesh Sahani.
Fast gaussian process methods for point process intensity estimation.
In *Proceedings of the 25th international conference on Machine learning*, pages 192–199. ACM, 2008.

[GSC15] Elad Gilboa, Yunus Saatçi, and John P Cunningham.
Scaling multidimensional inference for structured gaussian processes.
Pattern Analysis and Machine Intelligence, IEEE Transactions on, 37(2):424–436, 2015.

[GSW+15] Jacob R Gardner, Xinyu Song, Kilian Q Weinberger, Dennis Barbour, and John P Cunningham.
Psychophysical detection testing with bayesian active learning.
UAI, 2015.

[HHGL11] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel.
Bayesian active learning for classification and preference learning.
arXiv preprint arXiv:1112.5745, 2011.

[RW06] C. E. Rasmussen and C.K.I. Williams.
Gaussian Processes for Machine Learning.
MIT Press, Cambridge, 2006.

[Saa12] Yunus Saatçi.
Scalable inference for structured Gaussian process models.
PhD thesis, University of Cambridge, 2012.

[She94] Jonathan Richard Shewchuk.
An introduction to the conjugate gradient method without the agonizing pain, 1994.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams.
Practical bayesian optimization of machine learning algorithms.
In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[SMSL+03] Ercan Solak, Roderick Murray-Smith, William E Leithead, Douglas J Leith, and Carl Edward Rasmussen.
Derivative observations in gaussian process models of dynamic systems.
2003.