Surrogates 7020

Chapter 9: GP Fidelity and Scale

Dr. Alex Bledar Konomi

Department of Mathematical Sciences
University of Cincinnati

Gaussian Process

- Flops in $O(n^3)$ for matrix decompositions, furnishing determinants and inverses, is severe.
- Many practitioners point out that storage, which is in $O(n^2)$, is the real bottleneck, at least on modern laptops and workstations.
- Even if you're fine with waiting hours for MVN density calculation for a single likelihood evaluation, chances are you wouldn't have enough high-speed memory (RAM) to store the $n \times n$ matrix Σ_n, let alone its inverse too and any auxiliary space required.

Compactly supported kernels

- A kernel $k_{r_{max}}(r)$ is said to have compact support if $k_{r_{max}}(r) = 0$ when $r > rmax$. Recall from 5.3.3 that $r = |x - x'|$ for a stationary covariance.
- We may still proceed component-wise with $rj = |x_j - x'_j|$ and rj, max for a separable compactly supported kernel, augment with scales for amplitude adjustments, nuggets for noisy data and embellish with smoothness parameters (Matern), etc.
- Rate of decay of correlation can be managed by lengthscale hyperparameters.
- A compactly supported kernel (CSK) introduces zeros into the covariance matrix, so sparse matrix methods may be deployed to aid in computations, both in terms of economizing on storage and more efficient decomposition for inverses and determinants.

Two families of CSKs

Two families of CSKs, Bohman and truncated power, offer decent approximations to the power exponential family (5.3.3), of which the Gaussian (power $\alpha = 2$) is a special case. These kernels are zero for $r > r_{max}$, and for $r \leq r_{max}$:

$$k^B_{r_{max}}(r) = (1 - \frac{r}{r_{max}}) \cos(\frac{\pi r}{r_{max}}) + \frac{1}{\pi} \sin(\frac{\pi r}{r_{max}})$$

$$k^{tp}_{r_{max}}(r; \alpha, \nu) = [1 - (r/r_{max})^\alpha]^\nu,$$

where $0 < \alpha < 2$ and $\nu \geq \nu_m(\alpha)$. The function $\nu_m(\alpha)$ in the definition of the truncated power kernel represents a restriction necessary to ensure a valid correlation in m dimensions.

Working with CSKs

```
 kB <- function(r, rmax) {
  rnorm <- r/rmax
  k <- (1 - rnorm)*cos(pi*rnorm) + sin(pi*rnorm)/pi
  k <- k*(r < rmax)}
###
library(plgp)
X <- matrix(seq(0, 10, length=2000), ncol=1)
D <- distance(X)
##
 eps <- sqrt(.Machine$double.eps)  ## numerical stability
 K <- exp(-D) + diag(eps, nrow(D))
 K2 <- kB(sqrt(D), 2)
 K1 <- kB(sqrt(D), 1)
 K025 <- kB(sqrt(D), 0.25)
 c(mean(K > 0), mean(K2 > 0), mean(K1 > 0), mean(K025 > 0))
```

Invert these matrixes in R

Investigating the extent to which those levels of sparsity translate into computational savings requires investing in a sparse matrix library e.g., spam (Furrer 2018) or Matrix (Bates and Maechler 2019).

```
library(Matrix)
c(system.time(chol(K))[3],
system.time(chol(Matrix(K2, sparse=TRUE)))[3],
system.time(chol(Matrix(K1, sparse=TRUE)))[3],
system.time(chol(Matrix(K025, sparse=TRUE)))[3])
```

Discusion

- As you can see, small r_{max} holds the potential for more than an order of magnitude speedup.
- Further improvements may be possible if the matrix can be built natively in sparse representation.
- We want to encourage sparsity because that means speed, but getting enough sparsity requires lots of zeros, and that means sacrificing long range spatial correlation.
- If local modeling is sufficient, then why bother with a global model?

Bad results for CSK

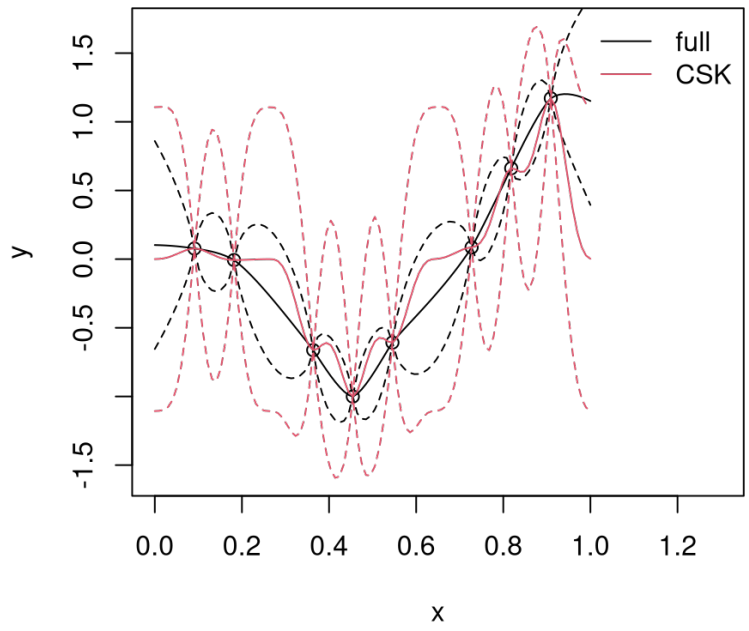


Figure: Predictions under CSK compared to the ideal full GP.

Impruvments exist but not neccessary streightover.

Tree grpah

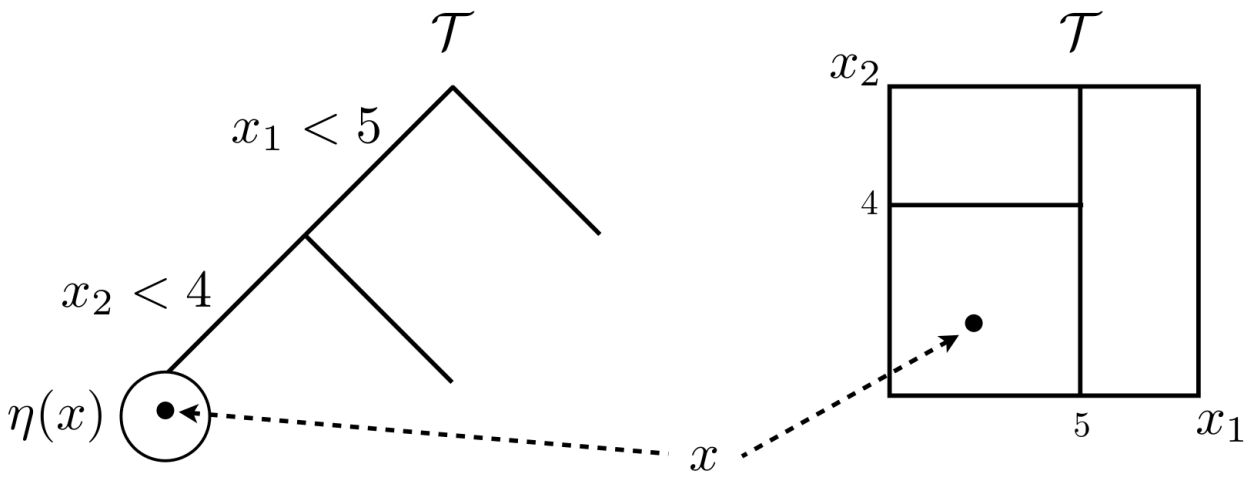


Figure: Tree graph (left) and partition of a 2d input space (right). Borrowed from H. Chipman et al. (2013) with many similar variations elsewhere; used with permission from Wiley.

Random-walk proposals in tree

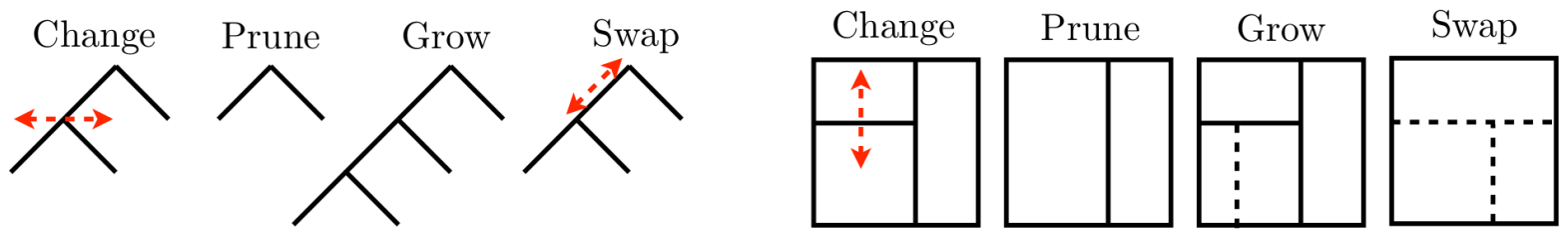


Figure: Random-walk proposals in tree space graphically (left four) and as partitions (right four). Borrowed from H. Chipman et al. (2013) with many similar variations elsewhere; used with permission from Wiley.

laGP

- A local approximate Gaussian process (LAGP), which the author has been so excited to tell you about, has aspects in common with partition based schemes.
- In the sense that it creates sparsity in the covariance structure in a geographically local way.
- In fact, LAGP is a partitioning scheme in a limiting sense, although delving too deeply into that connection is counterproductive because the approach is quite different from partitioning in spirit.
- The core LAGP innovation is reminiscent of what Cressie (1992, 131–34) called "ad hoc local kriging neighborhoods".
- **LAGP lies squarely in prediction, which is the primary goal in computer experiments and ML applications.**

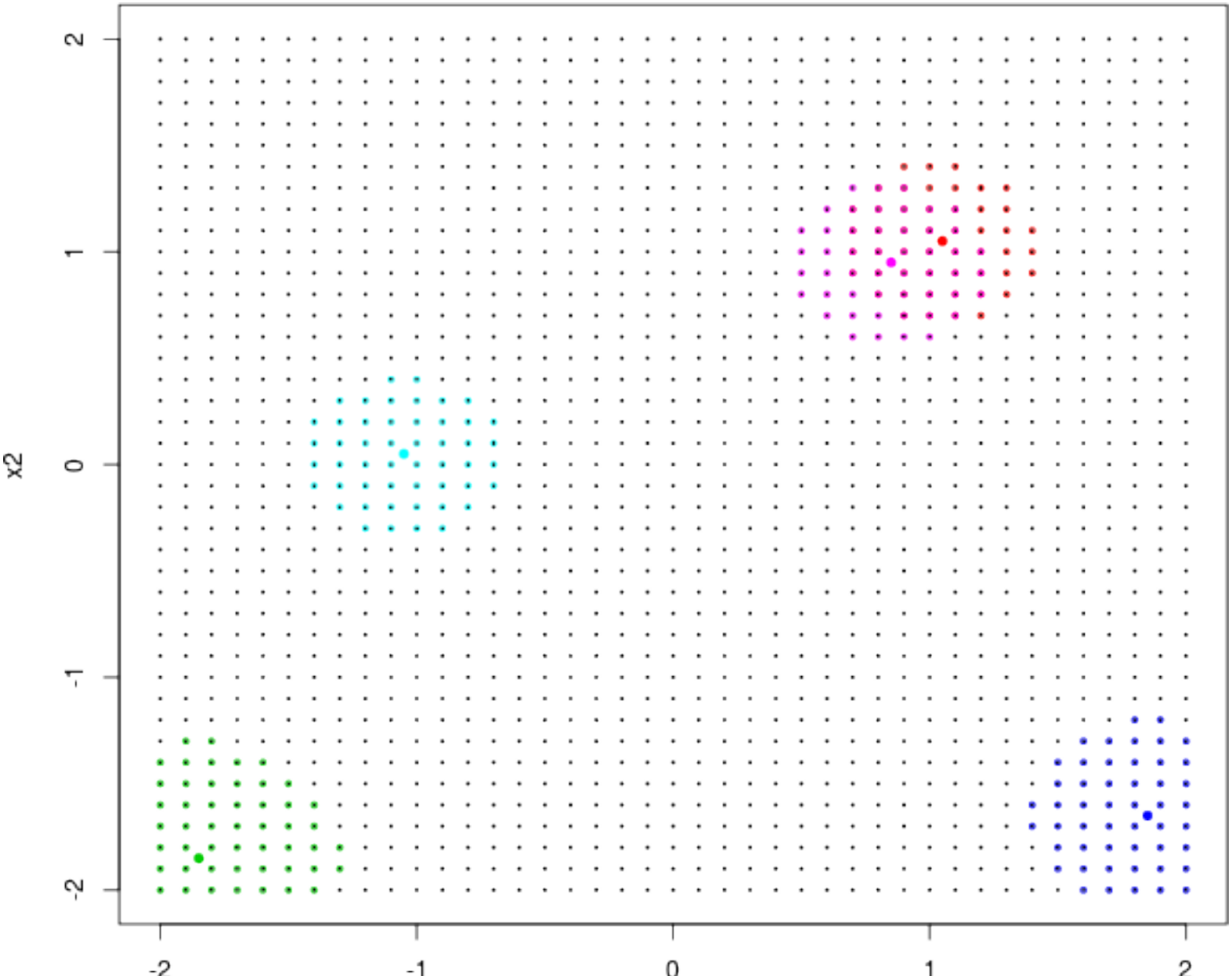
laGP

- For the next little bit, focus on prediction at a single testing location x. Coordinates encoded by x are arbitrary; it's only important that it be a single location in the input space X.
- Let's think about the properties of a GP surrogate at x. Training data far from x have vanishingly small influence on GP predictions, especially when correlation is measured as an inverse of exponentiated Euclidean distances.
- This is what motivates a CSK approach to inducing sparsity, but the difference here is that we're thinking about a particular x, not the entire spatial field.
- The crux of LAGP is a search for the most useful training data points – a subdesign relative to x – for predicting at x, without considering/handling large matrices.

Computaitonal Complexity

- One option is a nearest neighbor (NN) subset. Specifically, fill $X_n(x) \subset X_N$ with local$-n << full - N$ closest locations to x.
- big N represent the size of a potentially enormous training set, unwieldy for conventional GPs, and now little n denotes a much smaller, more manageable size.
- Derive GP predictive equations under $Y(x)D_n(x)$ where $D_n(x) = (X_n, Y_n)$, pretending that no other data exist. The best reference for this idea is Emery (2009).
- Prediction costs are in $O(n^3)$ and $O(n^2 + N)$ for decomposition(s) and storage, respectively; and NNs can be found in $O(nlogN)$ time with k-d trees after an up-front $O(NlogN)$ build cost.

Local neighborhoods



Topology

- Notice how topology of the global design X_N impacts the shape of local designs $X_n(x)$. When two predictive locations are nearby, as illustrated in pink and red, training data sites may be shared by subdesigns.
- There are no hard boundaries whereby adjacent, arbitrarily close predictive locations might be trained on totally disjoint data subsets.
- It's even possible to have two very close predictive locations $x \neq x'$ with the same subdesign $X_n(x) = X_n(x')$ when they share the same NN sets
- It can also be shown, again under some regularity conditions, that $V(x)D_n >> V(x)D_N$, reflecting uncertainties inflated by the smaller design, where $\sigma^2(x) = \tau^2 V(x)$.

Optimal Choice

- Finding the optimal n of N, of which there are (N_n) alternatives, could be a combinatorially huge undertaking.
- Can we do better than NN (in terms of prediction accuracy) without much extra effort (in terms of computational cost)? More precisely, n-NN GP prediction requires computation in $O(n^3)$.
- The answer to that question is a qualified "Yes!", with a greedy/forward stepwise scheme.

Greedy/forward stepwise scheme

For $j = n_0, \ldots, n$:

1. given $D_j(x)$, choose x_{j+1} according to some criterion;
2. augment the design $D_{j+1}(x) = D_j(x) \bigcup (x_{j+1}, y(x_{j+1}))$ and update the GP approximation.

Optimizing the criterion (1), and updating the GP (2), must not exceed $O(j^2)$ so the total scheme remains in $O(n^3)$.
Initialize with a small $D_{n_0}(x)$ comprised of NNs.

Criteria

Gramacy and Apley (2015), G&A below, proposed the following criterion for sequential subdesign. Given $D_j(x)$ for particular x, search for $x_{j+1} \in X_N \backslash X_n(x)$ considering its impact on predictive variance $V_j(x) \equiv V(x)|D_j(x)$, while taking into account uncertainty in hyperparameters θ, by minimizing empirical Bayes mean-squared prediction error:

$$\begin{aligned} J(x_{j+1}, x) &= E\{[Y(x) - \mu_{j+1}(x; \hat{\theta}_{j+1})]^2 | D_j(x)\} \\ &\approx V_j(x|x_{j+1}; \hat{\theta}_j) + \left(\frac{\partial \mu_j(x;\theta)}{\partial \theta}\bigg|_{\theta=\hat{\theta}_j} \right)^2 / \mathcal{G}_{j+1}(\hat{\theta}_j). \end{aligned}$$

Derivation

$$
\begin{aligned}
J(x_{j+1}, x) &= E\{[Y(x) - \mu_{j+1}(x;\theta_{j+1}) + \mu_{j+1}(x;\theta_{j+1}) - \mu_{j+1}(x;\hat{\theta}_{j+1})]^2 | D_j(x)\} \\
&= E\{[Y(x) - \mu_{j+1}(x;\theta_{j+1})]^2 | D_j(x)\} + E\{[\mu_{j+1}(x;\theta_{j+1}) - \mu_{j+1}(x;\hat{\theta}_{j+1})]^2 | D_j(\\
&+ 2E\{[Y(x) - \mu_{j+1}(x;\theta_{j+1})][\mu_{j+1}(x;\theta_{j+1}) - \mu_{j+1}(x;\hat{\theta}_{j+1})] | D_j(x)\} \\
&= V_j(x | x_{j+1}; \hat{\theta}_j) + E\{[\mu_{j+1}(x;\theta_{j+1}) - \mu_{j+1}(x;\hat{\theta}_{j+1})]^2 | D_j(x)\}.
\end{aligned}
$$

$$
\begin{aligned}
\mu_{j+1}(x;\theta_{j+1}) - \mu_{j+1}(x;\hat{\theta}_{j+1}) &\approx \left(\frac{\partial \mu_j(x;\theta)}{\partial \theta}\bigg|_{\theta=\hat{\theta}_{j+1}} \right) (\theta - \hat{\theta}_{j+1}) \\
&\approx \left(\frac{\partial \mu_j(x;\theta)}{\partial \theta}\bigg|_{\theta=\hat{\theta}_{j}} \right) (\theta - \hat{\theta}_{j+1})
\end{aligned}
$$

Derivation

$$E(\mu_{j+1}(x;\theta_{j+1}) - \mu_j(x;\hat{\theta}_{j+1}))^2 \approx \left(\frac{\partial \mu_j(x;\theta)}{\partial \theta}\bigg|_{\theta=\hat{\theta}_j} \right)^2 E(\theta - \hat{\theta}_{j+1})^2$$

Remember the connection of the variance with the second derivative of the likelihood (or the Fisher information matrix).

How to find the Variance

Sequential updating of the Fisher information leverages a recursive expression of the log likelihood which follows trivially from a cascading conditional representation of the joint probability of the responses given the parameters:

$$\begin{aligned} l_{j+1}(\theta) &= \log p(Y_{j+1}|\theta) \\ &= \log p(y_{j+1}|Y_j, \theta) + \log p(Y_j|\theta) \\ &= l_j(\theta) + l_j(y_{j+1}; \theta) \end{aligned}$$

where the final term represents the conditional log- likelihood for y_{j+1} given Y_j and θ.

Fisher information Matrix

Taking (negative) second derivatives yields the following updating equations:

$$\{F_{j+1}(\theta)\}_{kl} = -\frac{\partial^2 l_j(Y_j|\theta)}{\partial\theta_k\partial\theta_l} - \frac{\partial^2 l_j(y_{j+1}|\theta)}{\partial\theta_k\partial\theta_l}$$

Remember the connection of the variance with the second derivative of the likelihood (or the Fisher information matrix).

$$\{G_{j+1}(\theta)\} = \{F_{j+1}(\theta)\} - E\{\frac{\partial^2 l_j(y_{j+1}|\theta)}{\partial\theta_k\partial\theta_l}\}$$

Another Approximation

Unfortunately, the Student-t predictive equations preclude a tractable analytic expectation calculation (original paper the author is giving 2 pages of to find the second derivative. Which they never uses!). Therefore, we approximate by employing Gaussian surrogate equations with matched moments.

$$l_j(y_{j+1};\theta) = l_j(y_{j+1}|Y_j,\theta) \approx -\frac{1}{2}log(2\pi) - \frac{1}{2}log(V_j) - \frac{(y_{n+1}-\mu_j)^2}{2}$$

We take the first derivative, we take the second derivative. We find the expected value of the negative second derivative and finally we go back.

Intuition

- Let's break down elements of the MSPE criterion $J(x_{j+1}, x)$. Apparently it combines **variance** and **rate of change of the mean** at x.
- G&A's presentation, and indeed the original laGP package implementation (Gramacy and Sun 2018), emphasized isotropic lengthscale parameters θ.
- Our summary here follows that simplified setup. For extensions to vectorized θ for separable, coordinate-wise, lengthscales see the appendix to the original paper. A subsequently updated version of laGP supports separable lengthscales, as detailed by authors empirical work.