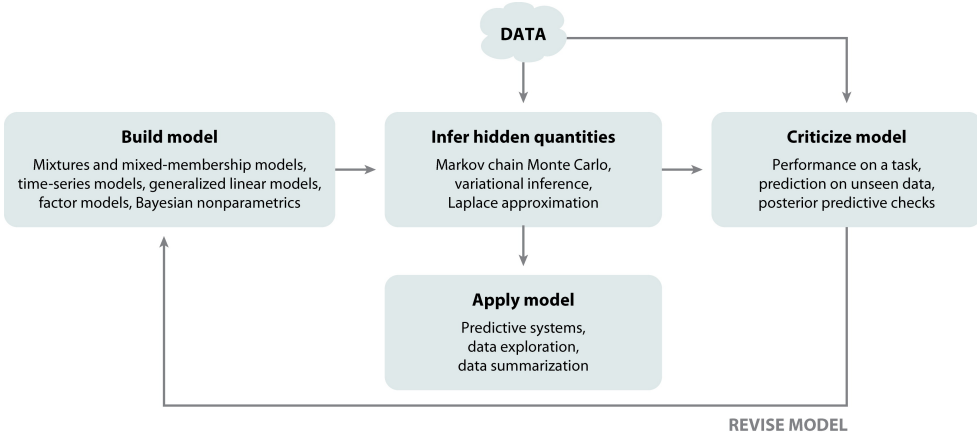
STATS271/371: Applied Bayesian Statistics

Gaussian processes, elliptical slice sampling, and Bayesian optimization

Scott Linderman

May 24, 2021

Box's Loop



Blei DM. 2014.
Annu. Rev. Stat. Appl. 1:203–32

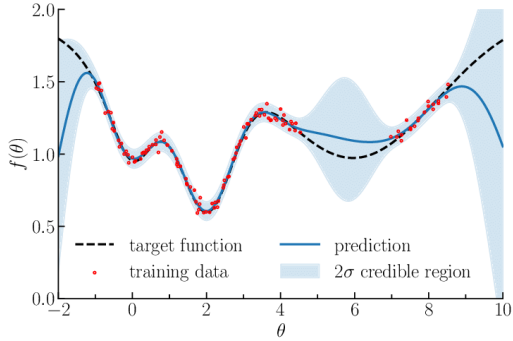
Blei, *Ann. Rev. Stat. App.* 2014.

Lap 8: Gaussian processes, elliptical slice sampling, and Bayesian optimization

- **Model:** Gaussian processes
- **Algorithm:** Elliptical slice sampling
- **Application:** Bayesian optimization

Gaussian processes

- Gaussian processes are distributions on functions $f : \mathbb{R}^D \to \mathbb{R}$. (We can generalize to other domains as well.)
- Equivalently, a GP is a continuous set of r.v.'s $\{f(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^D\}$; i.e. a stochastic process.



https://www.researchgate.net/figure/Illustration-of-Gaussian-process-regressio fig1_327613136

Gaussian processes II

We say $f \sim \mathrm{GP}(\mu(\cdot), K(\cdot,\cdot))$ if

$$\begin{bmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \vdots \\ \mu(\boldsymbol{x}_N) \end{bmatrix}, \begin{bmatrix} K(\boldsymbol{x}_1,\boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_1,\boldsymbol{x}_N) \\ \vdots & & \vdots \\ K(\boldsymbol{x}_N,\boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_N,\boldsymbol{x}_N) \end{bmatrix} \right) \tag{1}$$

for all finite subsets of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^D$.

Here, $\mu : \mathbb{R}^D \to \mathbb{R}$ is the **mean function** and $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is the **covariance function**, or **kernel**.

The covariance matrix obtained by applying the covariance function to each pair of data points above is called the **Gram matrix**.

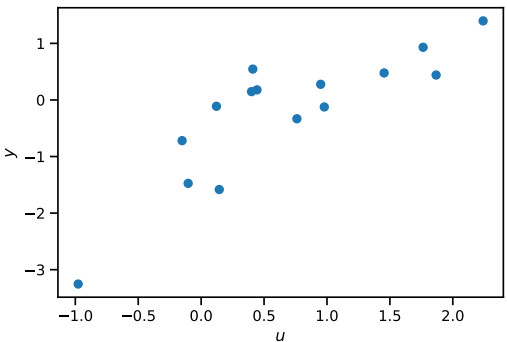
The covariance function must be positive definite; i.e. the Gram matrix must be positive definite for any subset of points.

From linear regression to GPs

- Think back to Lap 1: Bayesian Linear Regression.
- Our motivating example was approximating a $D = 1$ dimensional function $y(x) : \mathbb{R} \to \mathbb{R}$ given noisy observations $\{y_n, x_n\}_{n=1}^N$.
- We cast polynomial regression as linear regression by encoding the inputs x_n with feature vectors,

$$\boldsymbol{\phi}(x_n) = (x_n^0, x_n^1, \ldots, x_n^{P-1}) \in \mathbb{R}^P. \qquad (2)$$

- (I've changed notation slightly to match the slides above.)



From linear regression to GPs II

More generally, let $\boldsymbol{\phi}(x) = (\phi_1(x), \ldots, \phi_P(x)) \in \mathbb{R}^P$ be a function that encodes x in a P-dimensional feature space.

With these features, our linear model was,

$$\mathbb{E}[y_n \mid x_n] = \sum_{p=1}^{P} w_p \phi_p(x_n) = \boldsymbol{\phi}(x_n)^\top \boldsymbol{w} \triangleq f(x_n). \tag{3}$$

Now assume a Gaussian prior, $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \frac{\lambda}{P}\boldsymbol{I})$. Then for any $\{x_1, \ldots, x_N\} \subset \mathbb{R}$,

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix} = \begin{bmatrix} \phi_1(x_1) & \cdots & \phi_P(x_1) \\ \vdots & & \vdots \\ \phi_1(x_N) & \cdots & \phi_P(x_N) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_P \end{bmatrix} = \boldsymbol{\Phi}\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \tfrac{\lambda}{P}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top) \tag{4}$$

The function $f(\cdot)$ is a GP! It has kernel $K(x_i, x_j) = \frac{\lambda}{P}\boldsymbol{\phi}(x_i)^\top \boldsymbol{\phi}(x_j)$

Example: radial basis functions

Instead of polynomial features, let's use **radial basis functions**,

$$\phi_p(x) = e^{-\frac{1}{2\ell^2}(x-c_p)^2}. \tag{5}$$

These are un-normalized Gaussian bumps of width ℓ located at $c_1, \ldots, c_P$.

With this feature encoding, the kernel is,

$$K(x_i, x_j) = \tfrac{\lambda}{P}\boldsymbol{\phi}(x_i)^\top \boldsymbol{\phi}(x_j) = \frac{\lambda}{P}\sum_{p=1}^{P} e^{-\frac{1}{2\ell^2}(x_i-c_p)^2} e^{-\frac{1}{2\ell^2}(x_j-c_p)^2} \tag{6}$$

Now take the limit of having $P \to \infty$ equally spaced centers. Then,

$$\lim_{P\to\infty} K(x_i, x_j) = \lambda \int_{-\infty}^{\infty} e^{-\frac{1}{2\ell^2}(x_i-c)^2} e^{-\frac{1}{2\ell^2}(x_j-c)^2}\, \mathrm{d}c = \sqrt{\pi}\ell\lambda e^{-\frac{1}{2(\sqrt{2}\ell)^2}(x_i-x_j)^2} \tag{7}$$

This is called a **squared exponential kernel** with length scale $\sqrt{2}\ell$ and variance $\sqrt{\pi}\ell\lambda$.

Demo: sampling GPs with squared exponential kernels

https://colab.research.google.com/drive/1PFF5EkD-6g4Ta_4oU3y5ErdzQKh0T1lX?usp=sharing

The Matérn family of kernels

The Matérn family of kernels is defined by

$$K(x_i, x_j) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\Delta_{ij}}{\ell} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}\Delta_{ij}}{\ell} \right), \tag{8}$$

where

- $\Delta_{ij} = x_i - x_j$ is the difference of the points,
- ℓ is a positive length scale,
- the positive parameter ν controls the smoothness of the function, and
- and K_{ν} (in a potentially confusing overloading of notation) denotes the modified Bessel function.

When $\nu \to \infty$, the Matérn kernel converges to the squared exponential.

When $\nu = \frac{1}{2}$ the kernel is $K(x_i, x_j) = e^{-\frac{\Delta_{ij}}{\ell}}$, the covariance function of the **Ornstein-Uhlenbeck (OU) process**, a continuous-time AR(1) process.

When $\nu = p - \frac{1}{2}$, the kernel is the covariance of a continuous-time AR(p) process.

The Matérn family of kernels II

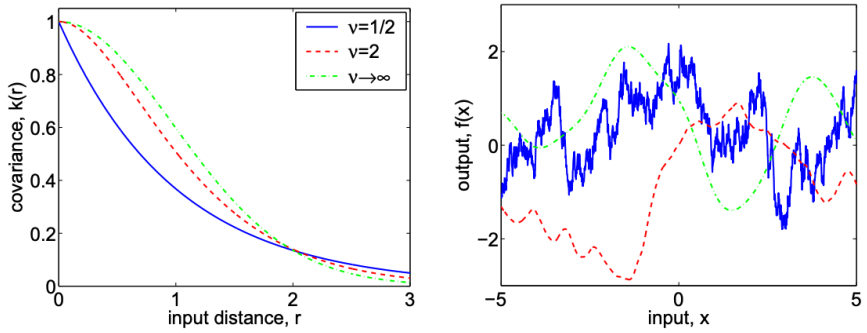


Figure 4.1: Panel (a): covariance functions, and (b): random functions drawn from Gaussian processes with Matérn covariance functions, eq. (4.14), for different values of ν, with $\ell = 1$. The sample functions on the right were obtained using a discretization of the x-axis of 2000 equally-spaced points.

Stationary kernels and Bochner's theorem

Note that both the squared exponential and the Matérn kernels are **stationary** in that $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ only depends on $\boldsymbol{\Delta}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$.

Stationary kernels are particularly interesting because they can be defined by their power spectrum.

Theorem (Bochner's Theorem)

A complex-valued function $K(\boldsymbol{\Delta})$ on $\mathbb{R}^D$ is the covariance function of a weakly stationary mean square continuous complex valued random process on $\mathbb{R}^D$ if and only if it can be represented as

$$K(\boldsymbol{\Delta}) = \int_{\mathbb{R}^D} e^{2\pi i \boldsymbol{s}\cdot\boldsymbol{\Delta}} \, \mathrm{d}\mu(\boldsymbol{s}) \tag{9}$$

where μ is a positive finite measure.

If μ has a density $S(\boldsymbol{s})$ then S is known as the **spectral density** or **power spectrum** corresponding to $K(\boldsymbol{\Delta})$. (This is Theorem 4.1 of Williams and Rasmussen [1996].)

Power spectrum intuition

Think of $\mu(\boldsymbol{s})$ as the power put into frequency $\boldsymbol{s}$. The constraint that it be positive is akin to requiring that the covariance be positive definite.

The **Wiener-Khintchine theorem** says that covariance function and the spectral density (assuming it exists) are Fourier duals of one another,

$$K(\boldsymbol{\Delta}) = \int S(\boldsymbol{s}) e^{2\pi i \boldsymbol{s}\cdot\boldsymbol{\Delta}} \,\mathrm{d}\boldsymbol{s}, \tag{10}$$

$$S(\boldsymbol{s}) = \int K(\boldsymbol{\Delta}) e^{-2\pi i \boldsymbol{s}\cdot\boldsymbol{\Delta}} \,\mathrm{d}\boldsymbol{\Delta} \tag{11}$$

The variance of the process is $K(\mathbf{0}) = \int S(\boldsymbol{s}) \,\mathrm{d}\boldsymbol{s}$ so the spectral density must be integrable (it must decay sufficiently fast as $|\boldsymbol{s}| \to \infty$) to define a valid process.

One nice consequence: the Gram matrix of a stationary kernel evaluated at evenly spaced points $x_1, \ldots, x_N$ in 1D is a **Toeplitz matrix**, which can be inverted in only $O(N \log N)$ time using the Fourier transform [Storkey, 1999, Cunningham et al., 2008]

Common kernels

covariance function	expression	S	ND
constant	σ_0^2	$\surd$	
linear	$\sum_{d=1}^{D} \sigma_d^2 x_d x'_d$		
polynomial	$(\mathbf{x} \cdot \mathbf{x}' + \sigma_0^2)^p$		
squared exponential	$\exp(-\frac{r^2}{2\ell^2})$	$\surd$	$\surd$
Matérn	$\frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{\ell} r\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{\ell} r\right)$	$\surd$	$\surd$
exponential	$\exp(-\frac{r}{\ell})$	$\surd$	$\surd$
γ-exponential	$\exp\left(-(\frac{r}{\ell})^\gamma\right)$	$\surd$	$\surd$
rational quadratic	$(1 + \frac{r^2}{2\alpha\ell^2})^{-\alpha}$	$\surd$	$\surd$
neural network	$\sin^{-1}\left(\frac{2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}}'}{\sqrt{(1+2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}})(1+2\tilde{\mathbf{x}}'^\top \Sigma \tilde{\mathbf{x}}')}}\right)$		$\surd$

Table 4.1: Summary of several commonly-used covariance functions. The covariances are written either as a function of $\mathbf{x}$ and $\mathbf{x}'$, or as a function of $r = |\mathbf{x} - \mathbf{x}'|$. Two columns marked 'S' and 'ND' indicate whether the covariance functions are stationary and nondegenerate respectively. Degenerate covariance functions have finite rank, see section 4.3 for more discussion of this issue.

From Williams and Rasmussen [1996]

The Kernel Cookbook:

Advice on Covariance functions

by David Duvenaud

https://www.cs.toronto.edu/~duvenaud/cookbook/

The "Automated Statistician"

Duvenaud et al. [2013] proposed a method to search over compositions of kernels to best fit the data. The idea is very cool, but unfortunately they came up with the *worst name ever*.

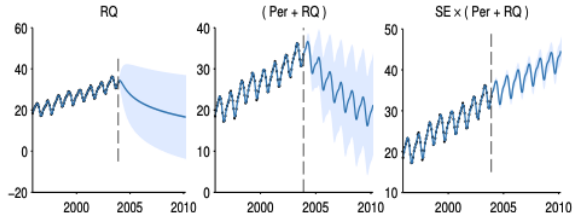


Figure 3. Posterior mean and variance for different depths of kernel search. The dashed line marks the extent of the dataset. In the first column, the function is only modeled as a locally smooth function, and the extrapolation is poor. Next, a periodic component is added, and the extrapolation improves. At depth 3, the kernel can capture most of the relevant structure, and is able to extrapolate reasonably.

Gaussian process regression

Now that we have a prior distribution on functions, suppose we observe $(y_n, \boldsymbol{x}_n)$ pairs. Assume,

$$f \sim \mathrm{GP}(\mu(\cdot), K(\cdot,\cdot)) \qquad\qquad y_n \sim \mathcal{N}(f(\boldsymbol{x}_n), \sigma^2) \tag{12}$$

independently for $n = 1, \ldots, N$.

Let $\boldsymbol{y} = [y_1, \ldots, y_N]^\top$, $\boldsymbol{\mu} = [\mu(\boldsymbol{x}_1), \ldots, \mu(\boldsymbol{x}_N)]^\top$, $\boldsymbol{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]^\top$, and $\boldsymbol{K}$ denote the Gram matrix of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$. Under the GP prior,

$$p(\boldsymbol{f} \mid \{\boldsymbol{x}_n, y_n\}_{n=1}^N) \propto \left[\prod_{n=1}^N p(y_n \mid f(\boldsymbol{x}_n))\right] p(\boldsymbol{f})\, \mathrm{d}\boldsymbol{f} \tag{13}$$

$$= \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{f}, \sigma^2 \boldsymbol{I})\, \mathcal{N}(\boldsymbol{f} \mid \boldsymbol{\mu}, \boldsymbol{K})\, \mathrm{d}\boldsymbol{f} \tag{14}$$

$$\propto \mathcal{N}(\boldsymbol{f} \mid \boldsymbol{\mu}', \boldsymbol{K}'), \tag{15}$$

where

$$\boldsymbol{K}' = (\boldsymbol{K}^{-1} + \sigma^{-2}\boldsymbol{I})^{-1} \qquad\qquad \boldsymbol{\mu}' = \boldsymbol{K}'(\boldsymbol{K}^{-1}\boldsymbol{\mu} + \sigma^{-2}\boldsymbol{y}) \tag{16}$$

Marginal distribution

Likewise, we can integrate over the random function to obtain the marginal distribution,

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y} \mid \boldsymbol{f})\, p(\boldsymbol{f})\, \mathrm{d}\boldsymbol{f} \tag{17}$$

$$= \int \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{f}, \sigma^2 \boldsymbol{I})\, \mathcal{N}(\boldsymbol{f} \mid \boldsymbol{\mu}, \boldsymbol{K})\, \mathrm{d}\boldsymbol{f} \tag{18}$$

$$= \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{\mu}, \boldsymbol{K} + \sigma^2 \boldsymbol{I}). \tag{19}$$

This allows us to compute the marginal likelihood of the data exactly.

Posterior predictive distribution

What is the posterior predictive distribution $p(y_{N+1} \mid \boldsymbol{x}_{N+1}, \{\boldsymbol{x}_n, y_n\}_{n=1}^N)$?

It's one big Gaussian model,

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \\ y_{N+1} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \vdots \\ \mu(\boldsymbol{x}_N) \\ \mu(\boldsymbol{x}_{N+1}) \end{bmatrix}, \begin{bmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_N) & K(\boldsymbol{x}_1, \boldsymbol{x}_{N+1}) \\ \vdots & & \vdots & \\ K(\boldsymbol{x}_N, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_N, \boldsymbol{x}_N) & K(\boldsymbol{x}_N, \boldsymbol{x}_{N+1}) \\ K(\boldsymbol{x}_{N+1}, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_{N+1}, \boldsymbol{x}_N) & K(\boldsymbol{x}_{N+1}, \boldsymbol{x}_{N+1}) \end{bmatrix} + \sigma^2 \boldsymbol{I} \right) \tag{20}$$

We obtain the predictive likelihood via a **Schur complement**,

$$y_{N+1} \mid \boldsymbol{x}_{N+1}, \{\boldsymbol{x}_n, y_n\}_{n=1}^N \sim \mathcal{N}(m_{N+1}, v_{N+1}) \tag{21}$$

$$m_{N+1} = \mu(\boldsymbol{x}_{N+1}) + \boldsymbol{k}^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - \boldsymbol{\mu}) \tag{22}$$

$$v_{N+1} = K(\boldsymbol{x}_{N+1}, \boldsymbol{x}_{N+1}) - \boldsymbol{k}^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k} + \sigma^2. \tag{23}$$

where $\boldsymbol{k} = [K(\boldsymbol{x}_1, \boldsymbol{x}_{N+1}), \ldots, K(\boldsymbol{x}_N, \boldsymbol{x}_{N+1})]^\top \in \mathbb{R}^N$.

GPs are linear predictors

Assume that $\mu(\boldsymbol{x}) = 0$ everywhere. Then we can write the predictive mean as,

$$m_{N+1} = \sum_{n=1}^{N} \alpha_n y_n, \tag{24}$$

where

$$\alpha_n = [\boldsymbol{K}^{-1}\boldsymbol{k}]_n \tag{25}$$

Question: what is the complexity of computing the predictive mean?

Recall the posterior predictive distribution in Bayesian linear regression

In Bayesian linear regression, we first compute the posterior mean of the weights.

In our notation from Slide 7, the posterior mean (under an uninformative prior) was,

$$\mathbb{E}[\boldsymbol{w} \mid \{\boldsymbol{x}_n, y_n\}_{n=1}^N] = \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{I}\right)^{-1} \left(\boldsymbol{\Phi}^\top \boldsymbol{y}\right) \tag{26}$$

so the prediction would be

$$\mathbb{E}[y_{N+1} \mid \boldsymbol{x}_{N+1}, \{\boldsymbol{x}_n, y_n\}_{n=1}^N] = \boldsymbol{\phi}(\boldsymbol{x}_{N+1})^\top \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{I}\right)^{-1} \left(\boldsymbol{\Phi}^\top \boldsymbol{y}\right) \tag{27}$$

Question: what is the complexity of computing the predictive mean in Bayesian linear regression?

Question: When is it more efficient to work with kernels rather than weights?

GP Classification

We've seen how Gaussian processes can be used for regression problems with $y_n \in \mathbb{R}$. What if we have binary observations under the following model,

$$f \sim \mathrm{GP}(\mu(\cdot), K(\cdot,\cdot)) \qquad\qquad y_n \sim \mathrm{Bern}(\sigma(f(\boldsymbol{x}_n))) \tag{28}$$

independently for $n = 1, \ldots, N$.

This is the GP analog of logistic regression.

As in logistic regression, the posterior is *not* available in closed form. All we know is,

$$p(\boldsymbol{f} \mid \{\boldsymbol{x}_n, y_n\}_{n=1}^N) \propto \left[\prod_{n=1}^N \mathrm{Bern}(y_n \mid \sigma(f(\boldsymbol{x}_n)))\right] \mathcal{N}(\boldsymbol{f} \mid \boldsymbol{\mu}, \boldsymbol{K}) \tag{29}$$

Also like logistic regression, we can use a Laplace approximation. But can we do better?

Lap 8: Gaussian processes, elliptical slice sampling, and Bayesian optimization

- **Model:** Gaussian processes
- **Algorithm: Elliptical slice sampling**
- **Application:** Bayesian optimization

Slice sampling intuition

Slice sampling

Slice sampling [Neal, 2003] is a MCMC algorithm for distributions with intractable normalization constants.

Suppose we want to sample a posterior distribution,

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}) = \frac{p(\boldsymbol{\theta}, \boldsymbol{y})}{p(\boldsymbol{y})}. \tag{30}$$

Assume we can evaluate the joint probability (numerator) but not the marginal (denominator).

Slice sampling works by introducing an **auxiliary variable** $u \in \mathbb{R}$ so that,

$$p(u, \boldsymbol{\theta} \mid \boldsymbol{y}) = \begin{cases} \frac{1}{p(\boldsymbol{y})} & \text{if } 0 \leq u \leq p(\boldsymbol{\theta}, \boldsymbol{y}) \\ 0 & \text{o.w.} \end{cases} \tag{31}$$

Note that marginalizing over u recovers the posterior,

$$\int p(u, \boldsymbol{\theta} \mid \boldsymbol{y}) \, \mathrm{d}u = \int_0^{p(\boldsymbol{\theta}, \boldsymbol{y})} \frac{1}{p(\boldsymbol{y})} \, \mathrm{d}u = \frac{p(\boldsymbol{\theta}, \boldsymbol{y})}{p(\boldsymbol{y})} = p(\boldsymbol{\theta} \mid \boldsymbol{y}). \tag{32}$$

If we can sample $p(u, \boldsymbol{\theta} \mid \boldsymbol{y})$, we can simply throw away the u's to get samples from $p(\boldsymbol{\theta} \mid \boldsymbol{y})$.

Slice sampling II

Gibbs sampling is often easy in the augmented space of $(u, \boldsymbol{\theta})$:

Given $\boldsymbol{\theta}$, the auxiliary variable u follows a uniform distribution,

$$p(u \mid \boldsymbol{\theta}, \boldsymbol{y}) \propto p(u, \boldsymbol{\theta} \mid \boldsymbol{y}) = \frac{1}{p(\boldsymbol{y})} \mathbb{I}[0 \leq u \leq p(\boldsymbol{\theta}, \boldsymbol{y})] \propto \text{Unif}(u; [0, p(\boldsymbol{\theta}, \boldsymbol{y})]). \tag{33}$$

Given u, the variable of interest $\boldsymbol{\theta}$ is uniformly distributed as well,

$$p(\boldsymbol{\theta} \mid u, \boldsymbol{y}) \propto p(u, \boldsymbol{\theta} \mid \boldsymbol{y}) = \frac{1}{p(\boldsymbol{y})} \mathbb{I}[p(\boldsymbol{\theta}, \boldsymbol{y}) \geq u] \propto \text{Unif}(\boldsymbol{\theta}; \boldsymbol{\Theta}(u)), \tag{34}$$

where $\boldsymbol{\Theta}(u) = \{\boldsymbol{\theta}' : p(\boldsymbol{\theta}', \boldsymbol{y}) \geq u\}$ is a **slice** of parameter space with joint probability at least u.

If we can find that slice and sample uniformly from it, we're in business!

Slice sampling III

$\Theta(u)$ is generally a complex subset that is hard to sample uniformly.

Instead, we typically sampling one coordinate of θ at a time, holding the rest of fixed. (This is also a valid Gibbs update.)

Still, we need some way of finding the slice. Neal [2003] proposes a "stepping out" procedure to compute a 1D slice.

Slice sampling for GPs

Consider a GP classification with 2 data points ($\boldsymbol{x}_1, \boldsymbol{x}_2$). Assume they are close relative to the length-scale of a squared exponential kernel.

Accounting for the correlated Gaussian prior

Single variable slice sampling, like all Gibbs sampling methods, suffers when the coordinates are correlated.

For Gaussian processes, the prior induces correlations between the function values, and this can serious impair such coordinate-wise update algorithms.

Can we reparameterize the problem so that our 1D slice is better aligned with the prior?

A strange way to sample a Gaussian...

Suppose $\boldsymbol{f} \sim \mathcal{N}(\mathbf{0}, \Sigma)$. A strange but interesting way to sample $\boldsymbol{f}$ is via the following program:

$$\boldsymbol{v}_0 \sim \mathcal{N}(\mathbf{0}, \Sigma) \tag{35}$$
$$\boldsymbol{v}_1 \sim \mathcal{N}(\mathbf{0}, \Sigma) \tag{36}$$
$$\theta \sim \mathrm{Unif}([0, 2\pi)) \tag{37}$$
$$\boldsymbol{f} = \boldsymbol{v}_0 \sin\theta + \boldsymbol{v}_1 \cos\theta \tag{38}$$

To see this, define $\boldsymbol{v}' = \begin{bmatrix} \boldsymbol{v}_0 & \boldsymbol{v}_1 \end{bmatrix}^\top$ and note that for any $\theta \in [0, 2\pi)$

$$\boldsymbol{v}' \sim \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\Sigma & \mathbf{0}\\ \mathbf{0} & \Sigma\end{bmatrix}\right), \tag{39}$$

$$\boldsymbol{f} = \begin{bmatrix}\sin\theta \boldsymbol{I} & \cos\theta \boldsymbol{I}\end{bmatrix} \boldsymbol{v}' \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\sin\theta \boldsymbol{I} & \cos\theta \boldsymbol{I}\end{bmatrix} \begin{bmatrix}\Sigma & \mathbf{0}\\ \mathbf{0} & \Sigma\end{bmatrix} \begin{bmatrix}\sin\theta \boldsymbol{I}\\ \cos\theta \boldsymbol{I}\end{bmatrix}\right) \tag{40}$$

$$\sim \mathcal{N}(\mathbf{0}, (\sin^2\theta + \cos^2\theta)\Sigma) = \mathcal{N}(\mathbf{0}, \Sigma). \tag{41}$$

Elliptical slice sampling

With this sampling procedure in mind, Murray et al. [2010] proposed **elliptical slice sampling**.

Reparameterize $\boldsymbol{f}$ in terms of $\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \theta$ and sample the posterior,

$$p(\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \theta \mid \boldsymbol{y}) \propto \mathcal{N}(\boldsymbol{\nu}_0 \mid \mathbf{0}, \Sigma)\, \mathcal{N}(\boldsymbol{\nu}_1 \mid \mathbf{0}, \Sigma)\, \mathrm{Unif}(\theta \mid [0, 2\pi))\, p(\boldsymbol{y} \mid \boldsymbol{f}(\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \theta)) \tag{42}$$

Their sampling algorithm consists of two steps:

1. Sample $\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \theta \mid \boldsymbol{y}, \boldsymbol{f} = \boldsymbol{\nu}_0 \sin\theta + \boldsymbol{\nu}_1 \cos\theta$
   - 1.1 Sample $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \Sigma)$ and $\theta \sim \mathrm{Unif}([0, 2\pi))$.
   - 1.2 Set $\boldsymbol{\nu}_0 = \boldsymbol{f} \sin\theta + \boldsymbol{\nu} \cos\theta$ and $\boldsymbol{\nu}_1 = \boldsymbol{f} \cos\theta - \boldsymbol{\nu}' \sin\theta$.
2. Sample $\theta \mid \boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \boldsymbol{y}$ by slice sampling its conditional $p(\theta \mid \boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \boldsymbol{y}) \propto p(\boldsymbol{y} \mid \boldsymbol{\nu}_0 \cos\theta + \boldsymbol{\nu}_1 \sin\theta)$.

We can think of step 2 as slice sampling on an elliptical slice, hence the name.

Elliptical slice sampling II

This formulation is a little complicated. There's no need to actually instantiate ν_0 and ν_1. Instead, we can just work with ν and θ.

Input: current state $\mathbf{f}$, a routine that samples from $\mathcal{N}(0, \Sigma)$, log-likelihood function $\log L$.

Output: a new state $\mathbf{f}'$. When $\mathbf{f}$ is drawn from $p^\star(\mathbf{f}) \propto \mathcal{N}(\mathbf{f}; 0, \Sigma)\, L(\mathbf{f})$, the marginal distribution of $\mathbf{f}'$ is also $p^\star$.

1. Choose ellipse: $\boldsymbol{\nu} \sim \mathcal{N}(0, \Sigma)$
2. Log-likelihood threshold:
$$u \sim \text{Uniform}[0, 1]$$
$$\log y \leftarrow \log L(\mathbf{f}) + \log u$$
3. Draw an initial proposal, also defining a bracket:
$$\theta \sim \text{Uniform}[0, 2\pi]$$
$$[\theta_{\min}, \theta_{\max}] \leftarrow [\theta - 2\pi,\ \theta]$$
4. $\mathbf{f}' \leftarrow \mathbf{f} \cos\theta + \boldsymbol{\nu} \sin\theta$
5. **if** $\log L(\mathbf{f}') > \log y$ **then:**
6. Accept: **return** $\mathbf{f}'$
7. **else:**
Shrink the bracket and try a new point:
8. **if** $\theta < 0$ **then:** $\theta_{\min} \leftarrow \theta$ **else:** $\theta_{\max} \leftarrow \theta$
9. $\theta \sim \text{Uniform}[\theta_{\min}, \theta_{\max}]$
10. **GoTo** 4.

Elliptical slice sampling III

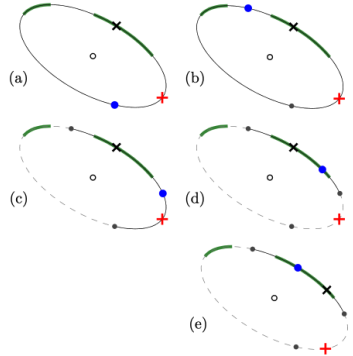


Figure 3: **(a)** The algorithm receives $\mathbf{f}$=✕ as input. Step 1 draws auxiliary variate $\boldsymbol{\nu}$=+, defining an ellipse centred at the origin (○). Step 2: a likelihood threshold defines the 'slice' (—). Step 3: an initial proposal ● is drawn, in this case not on the slice. **(b)** The first proposal defined both edges of the $[\theta_{\min}, \theta_{\max}]$ bracket; the second proposal (●) is also drawn from the whole range. **(c)** One edge of the bracket (—) is moved to the last rejected point such that ✕ is still included. Proposals are made with this shrinking rule until one lands on the slice. **(d)** The proposal here (●) is on the slice and is returned as $\mathbf{f}'$. **(e)** Shows the reverse configuration discussed in Section 2.3: ✕ is the input $\mathbf{f}'$, which with auxiliary $\boldsymbol{\nu}'$=+ defines the same ellipse. The brackets and first three proposals (•) are the same. The final proposal (●) is accepted, a move back to $\mathbf{f}$.

Computational complexity

Question: What is the computational complexity of one elliptical slice sampling update? Assume conditionally independent Bernoulli observations, as in GP classification.

Lap 8: Gaussian processes, elliptical slice sampling, and Bayesian optimization

- **Model:** Gaussian processes
- **Algorithm:** Elliptical slice sampling
- **Algorithm:** Variational Inference with Sparse GPs
- **Application:** Bayesian optimization

Inducing point methods

The key limitation of GPs is the cubic complexity of inference (with quadratic memory complexity).

One way of circumventing this complexity is via **inducing points** [**???**]. These form the basis of an approximate model called **Sparse GPs**.

We follow the presentation by Hensman et al. [2013].

Sparse GPs and Inducing Points

Consider a Gaussian process regression with inputs $\{\boldsymbol{x}_n\}_{n=1}^N \subset \mathbb{R}^D$ and observations $\{y_n\}_{n=1}^N \subset \mathbb{R}$. As above, let $\boldsymbol{y} = [y_1, \ldots, y_N]^\top$ and $\boldsymbol{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]^\top$

Introduce a set of **inducing points** $\{\boldsymbol{z}_m\}_{m=1}^M \subset \mathbb{R}^D$ with corresponding function values $u_m = f(\boldsymbol{z}_m)$.

Using the GP predictive distribution, we can write,

$$p(\boldsymbol{y} \mid \boldsymbol{f}) = \mathcal{N}(\boldsymbol{y} \mid \sigma^2 \boldsymbol{I}) \tag{43}$$

$$p(\boldsymbol{f} \mid \boldsymbol{u}) = \mathcal{N}(\boldsymbol{f} \mid \boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{u}, \tilde{\boldsymbol{K}} \tag{44}$$

where $\boldsymbol{K}_{nm}$ is the Gram matrix of kernel evaluations for each $(\boldsymbol{x}_n, \boldsymbol{z}_m)$ pair, $\boldsymbol{K}_{mm}$ is the Gram matrix for each $(\boldsymbol{z}_m, \boldsymbol{z}_m)$ pair, and

$$\tilde{\boldsymbol{K}} = \boldsymbol{K}_{nn} - \boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{mn}. \tag{45}$$

Note that $\tilde{\boldsymbol{K}} \in \mathbb{R}^{N \times N}$, so we're still looking at $O(N^3)$ complexity.

Variational inference for Sparse GPs

Now we will lower bound the marginal log likelihood using Jensen's inequality,

$$\log p(\boldsymbol{y} \mid \boldsymbol{u}) = \log \int p(\boldsymbol{y}, \boldsymbol{f} \mid \boldsymbol{u}) \, \mathrm{d}\boldsymbol{f} \tag{46}$$

$$= \log \mathbb{E}_{p(\boldsymbol{f} \mid \boldsymbol{u})}[p(\boldsymbol{y} \mid \boldsymbol{f})] \tag{47}$$

$$\geq \mathbb{E}_{p(\boldsymbol{f} \mid \boldsymbol{u})}[\log p(\boldsymbol{y} \mid \boldsymbol{f})] \tag{48}$$

$$\triangleq \mathscr{L}_1. \tag{49}$$

Assume $p(\boldsymbol{y} \mid \boldsymbol{f}) = \prod_{n=1}^{N} p(y_n \mid f(\boldsymbol{x}_n))$; i.e. the observations are conditionally independent.

Question: what is the complexity of evaluating the bound in eq. (48)?

Question: when is the bound maximized?

Variational inference for Sparse GPs II

Using the bound from above, we can obtain a lower bound on the marginal likelihood of the data, integrating out the inducing variables [Titsias, 2009],

$$\log p(\boldsymbol{y} \mid \boldsymbol{X}) = \log \int p(\boldsymbol{y} \mid \boldsymbol{u})\, p(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u} \tag{50}$$

$$\geq \log \int \exp\{\mathscr{L}_1\}\, p(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u} \tag{51}$$

$$\triangleq \mathscr{L}_2. \tag{52}$$

After some algebra, this bound equals,

$$\mathscr{L}_2 = \log \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{0}, \boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{mn} + \sigma^2 \boldsymbol{I}) - \frac{1}{2\sigma^2}\mathrm{Tr}(\tilde{\boldsymbol{K}}). \tag{53}$$

We can view this bound as arising from a variational approximation $q(\boldsymbol{u}) = \mathcal{N}(\hat{\boldsymbol{u}}, \boldsymbol{\Lambda}^{-1})$ where

$$\boldsymbol{\Lambda} = \sigma^{-2}\boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{mn}\boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1} + \boldsymbol{K}_{mm}^{-1} \tag{54}$$

$$\hat{\boldsymbol{u}} = \sigma^{-2}\boldsymbol{\Lambda}^{-1}\boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{mn}\boldsymbol{y}. \tag{55}$$

Question: what is the complexity of computing $\mathscr{L}_2$?

Stochastic variational inference for Sparse GPs

Marginalizing over $\boldsymbol{u}$ coupled the observations $\boldsymbol{y}$ so that the likelihood was a multivariate Gaussian (albeit with a low rank plus diagonal covariance).

As we briefly discussed in Lap 5: Mixed Membership Models, if the likelihood factors into a product over data points we can use **stochastic variational inference**, operating on mini-batches of data instead.

Can we do something similar here?

Stochastic variational inference for Sparse GPs II

Hensman et al. [2013] showed that if we explicitly represent $q(\boldsymbol{u})$, the likelihood $p(\boldsymbol{y} \mid \boldsymbol{u})$ factors into a product over the N data points.

Define a new lower bound,

$$\begin{aligned} \log p(\boldsymbol{y} \mid \boldsymbol{X}) &\geq \mathbb{E}_{q(\boldsymbol{u})}\left[\log p(y \mid \boldsymbol{u}) + \log p(\boldsymbol{u}) - \log q(\boldsymbol{u})\right] && (56) \\ &\geq \mathbb{E}_{q(\boldsymbol{u})}\left[\mathscr{L}_1 + \log p(\boldsymbol{u}) - \log q(\boldsymbol{u})\right] && (57) \\ &\triangleq \mathscr{L}_3, && (58) \end{aligned}$$

and assume $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{m}, \boldsymbol{S})$. Then the bound simplifies to,

$$\mathscr{L}_3 = \sum_{n=1}^{N}\left[\log \mathcal{N}(y_n \mid \boldsymbol{k}_n^\top \boldsymbol{K}_{mm}^{-1}\boldsymbol{m},\, \sigma^2) - \frac{1}{2\sigma^2}\tilde{K}_{n,n} - \frac{1}{2}\mathrm{Tr}(\boldsymbol{S}\boldsymbol{\Lambda}_n\right] - D_{\mathrm{KL}}\left(q(\boldsymbol{u}) \,\|\, p(\boldsymbol{u})\right) \tag{59}$$

where $\boldsymbol{k}_n$ is the k-th column of $\boldsymbol{K}_{mn}$ and $\Lambda_n = \sigma^{-1}\boldsymbol{K}_{mm}^{-1}\boldsymbol{k}_n\boldsymbol{k}_n^\top\boldsymbol{K}_{mm}^{-1}$.

This lower bound and its gradients wrt $\boldsymbol{m}$ and $\boldsymbol{S}$ can be approximated with Monte Carlo using mini-batches. Hensman et al. [2013] used this trick to fit sparse GPs to millions of data points.

Choosing the inducing points

Question: how should we obtain the inducing points?

References I

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for regression*. MIT Press, 1996.

Amos J Storkey. Truncated covariance matrices and toeplitz methods in gaussian processes. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, pages 55–60 vol.1. unknown, February 1999.

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, New York, NY, USA, July 2008. Association for Computing Machinery.

David Duvenaud, James Lloyd, Roger Grosse, Joshua Tenenbaum, and Ghahramani Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, pages 1166–1174. PMLR, 2013.

Radford M Neal. Slice sampling. *Annals of Statistics*, 31(3):705–767, June 2003.

References II

Iain Murray, Ryan Adams, and David MacKay. Elliptical slice sampling. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 541–548. jmlr.org, March 2010.

James Hensman, Nicolò Fusi, and Neil D Lawrence. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pages 282–290, Arlington, Virginia, USA, August 2013. AUAI Press.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 2009. PMLR.