Statistical Learning: Gaussian process regression

Jakob Söhl

Delft University of Technology

Introduction

Gaussian process regression

Advanced topics (not part of the exam)

Hyperparameter estimation by empirical Bayes

Gaussian process classification (RG section 8.3)

Gibbs sampling for GP classification (RG section 9.2)

Introduction

Some historical background

- Long history in geostatistics/spatial statistics: kriging.
- Popularised in machine learning 1995–
- Influential papers by Radford Neal.
- Well known book: *Gaussian Processes for Machine Learning*, Carl Edward Rasmussen and Chris Williams, the MIT Press, 2006. `https://gaussianprocess.org/gpml/`
- Builds upon theory on stochastic processes in continuous time.

Remark on notation

- I try to avoid boldface, bars, etc.
- So a vector is written like $y \in \mathbb{R}^n$.
- If there is risk to confusion I use boldface to denote a vector. Example: $\boldsymbol{x}_i = \begin{bmatrix} 1 & x_i \end{bmatrix}^T$.

Gaussian process regression

Revisiting the Bayesian linear model

The model is given by

$$\begin{aligned} y \mid \theta &\sim N(X\theta, \Sigma) \\ \theta &\sim N(0, \Sigma_0) \end{aligned}$$

Assume the noise covariance Σ is known.

Claim: $y \sim N(0, \mathcal{K})$ with $\mathcal{K} = \Sigma + X\Sigma_0 X^T$.

Checking that the mean and covariance are as stated:

1. $\mathbb{E}[y] = \mathbb{E}\,\mathbb{E}[y \mid \theta] = \mathbb{E}[X\theta] = X\mathbb{E}\theta = 0$.
2. Use the law of total covariance:

$$\begin{aligned} \operatorname{Cov} y &= \mathbb{E} \operatorname{Cov}(y \mid \theta) + \operatorname{Cov} \mathbb{E}[y \mid \theta] \\ &= \mathbb{E}\Sigma + \operatorname{Cov}(X\theta) = \Sigma + X\Sigma_0 X^T. \end{aligned}$$

Implied covariance structure

Assume $\Sigma = \sigma^2 I_n$, $\Sigma_0 = \sigma_0^2 I_p$.

$$\text{Cov } y = \Sigma + X\Sigma_0 X^T = \sigma^2 I_n + \sigma_0^2 X X^T.$$

This implies

$$\text{Cov}(y_i, y_j) = \sigma^2 \delta_{ij} + \sigma_0^2 \, x_i^T x_j,$$

where

$$\delta_{ij} = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases}$$

and x_i^T is the i-th row of X. Let

$$\mathcal{K}_{ij} = \sigma^2 \delta_{ij} + \sigma_0^2 x_i^T x_j.$$

If we define

$$K(x, \tilde{x}) = \sigma^2 \mathbf{1}_{\{x = \tilde{x}\}} + \sigma_0^2 x^T \tilde{x}$$

then $\mathcal{K}_{ij} = K(x_i, x_j)$.

Example: linear regression with M basis functions (1/2)

Assume observation pairs (u_i, y_i) and the model

$$y_i = \sum_{k=1}^{p} \theta_k \varphi_k(u_i) + \varepsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2).$$

Thus, **feature vector**

$$x_i = \begin{bmatrix} \varphi_1(u_i) & \varphi_2(u_i) & \dots & \varphi_p(u_i) \end{bmatrix}^T$$

and **parameter vector** $\theta = \begin{bmatrix} \theta_1 & \cdots & \theta_p \end{bmatrix}^T$.

Bayesian analysis: suppose prior

$$\{\theta_k\} \overset{\text{ind}}{\sim} N(0, \lambda_k), \qquad k = 1, \dots, p$$

so $\boldsymbol{\theta} \sim N_p(0, \Sigma_0)$ with $\Sigma_0 = \operatorname{diag}(\lambda_1, \dots, \lambda_p)$.

Example: linear regression with M basis functions (2/2)

Recall

$$y \sim N(0, \mathcal{K}), \qquad \mathcal{K} = X\Sigma_0 X^T + \Sigma.$$

This implies

$$\mathcal{K}_{ij} = \sum_{k=1}^{p} \lambda_k \varphi_k(u_i)\varphi_k(u_j) + \delta_{ij}\sigma^2.$$

Gaussian Process Regression (GPR):

- Consider $p \to \infty$ (implies conditions on decay of λ_k, as $k \to \infty$).
- Key idea: start from the kernel K.

Gaussian processes in case of noiseless regression

Main idea: nonparametric regression by assuming

$$y_i = f(x_i) + \underbrace{\epsilon_i}_{\text{assume 0 (just for now)}}.$$

Denote $f_i \equiv f(x_i)$. So that $y = If$ (with I the identity matrix), a linear model without noise.

Define the prior

$$\begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \sim N_n \left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \dots & K(x_1, x_n) \\ \vdots & \vdots & \dots & \vdots \\ K(x_n, x_1) & K(x_n, x_2) & \dots & K(x_n, x_n) \end{bmatrix} \right),$$

where the **kernel function** $K(x, y)$ needs to be such that the covariance matrix is symmetric and positive semi definite (psd).

What is a Gaussian process?

Definition. Let x_i, $i = 1, \ldots, n$ be feature vectors, say in $\mathbb{R}^p$. Let

- $\mu : \mathbb{R}^p \to \mathbb{R}$ be the mean function
- $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ be the (covariance) kernel function.

Then $x \mapsto f_x$ is a Gaussian process with mean function μ and kernel K if for *any* finite set $\{x_1, \ldots, x_n\}$, with $x_i \in \mathbb{R}^p$ for all i, we have

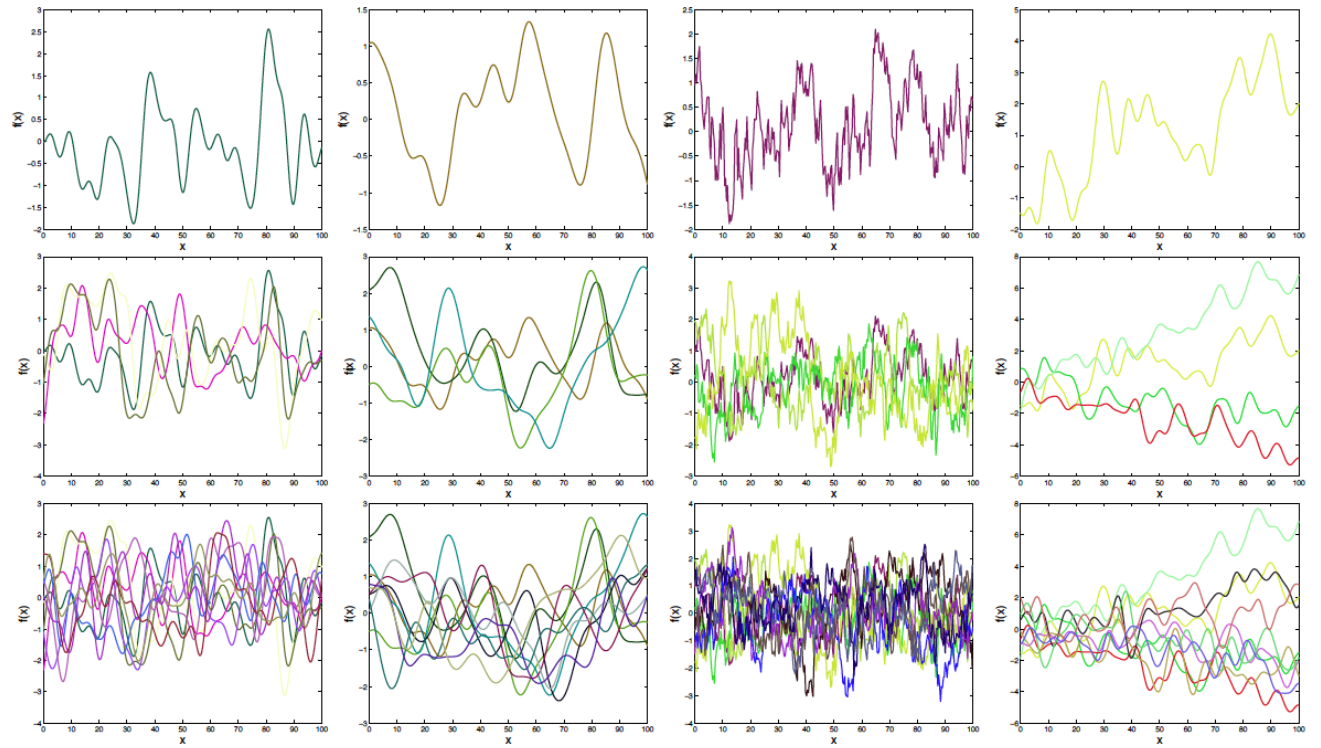
$$\begin{bmatrix} f_{x_1} \\ f_{x_2} \\ \vdots \\ f_{x_n} \end{bmatrix} \sim N_n \left(\begin{bmatrix} \mu(x_1) \\ \mu(x_2) \\ \vdots \\ \mu(x_n) \end{bmatrix}, \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \ldots & K(x_1, x_n) \\ K(x_2, x_1) & K(x_2, x_2) & \ldots & K(x_2, x_n) \\ \vdots & \vdots & \ldots & \vdots \\ K(x_n, x_1) & K(x_n, x_2) & \ldots & K(x_n, x_n) \end{bmatrix} \right).$$

We write $f \sim GP(\mu, K)$. The matrix with elements $K_{ij} \equiv K(x_i, x_j)$ is called the **Gram**-matrix.

It can be shown that such processes exist, with realisations that are continuous functions (under certain conditions).

Some realisations of Gaussian processes

Samples from GPs with different $K(x, x')$



Kernels

- K needs to be such that the covariance matrix of $(Z_{t_1}, \ldots, Z_{t_n})$ is positive definite (advanced: Bochner's theorem gives conditions for stationary covariance functions).
- A covariance function is **stationary** if $K(x, y)$ is a function of $x - y$. This yields invariance with respect to translations in feature (input) space.
- A covariance function is **isotropic** if $K(x, y)$ is a function of $\|x - y\|$. This yields invariance with respect to rigid motions in input space.

Kernels

- Popular choice **Square exponential kernel**

$$K(x, \tilde{x}) = v_0 \exp\left(-\frac{\|x - \tilde{x}\|^2}{2\ell^2}\right),$$

  with

  - v_0 signal variance
  - ℓ characteristic length scale

- **Polynomial kernels**

$$K(x, \tilde{x}) = \alpha(1 + x^T \tilde{x})^u.$$

`https://distill.pub/2019/visual-exploration-gaussian-processes/`

Bayesian nonparametric model

Gaussian Process regression assumes the following model:

$$y_i \mid f \overset{\text{ind}}{\sim} N(f(x_i), \sigma^2)$$
$$f \sim GP(0, K)$$

Note that x_i may be vector valued.

Key result: the posterior is a Gaussian process.

In the following we assume σ is known and fixed.

Finding the posterior: case $\sigma = 0$

$\sigma = 0$ implies $y_i = f_i$, with $f_i = f(x_i)$. Set

$$\boldsymbol{f} = \begin{bmatrix} f_1 & \dots & f_n \end{bmatrix}^T.$$

Suppose we are interested in $f_1^*, \dots, f_L^*$, where $f_k^* = f(x_k^*)$. Set

$$\boldsymbol{f}^* = \begin{bmatrix} f_1^* & \dots & f_L^* \end{bmatrix}^T.$$

By definition of the Gaussian process:

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}^* \end{bmatrix} \sim N\left(\begin{bmatrix} \mu \\ \mu^* \end{bmatrix}, \begin{bmatrix} \mathcal{K} & R \\ R^T & \mathcal{K}^* \end{bmatrix} \right)$$

$$\mu_i = \mu(x_i) \qquad \mu^*(x_k) = \mu(x_k)$$

$$\mathcal{K}_{ij} = K(x_i, x_j) \qquad \mathcal{K}^*_{k\ell} = K(x_k^*, x_\ell^*) \qquad R_{ik} = K(x_i, x_k^*).$$

Note that the matrix K is denoted by C in RG.

Conditionals of the Gaussian distribution are Gaussian:

$$\boldsymbol{f}^* \mid \boldsymbol{f} \sim N\left(\mu^*_{\text{post}}, \Sigma^*_{\text{post}}\right)$$

with

$$\mu^*_{\text{post}} = \mu^* + R^T \mathcal{K}^{-1}(\boldsymbol{f} - \mu) \qquad \Sigma^*_{\text{post}} = \mathcal{K}^* - R^T \mathcal{K}^{-1} R$$

Note that the formulas in RG consider the special case where $\mu = \mu^* = 0$.

Case $\sigma \neq 0$

Basic idea

$$\begin{aligned} p(\boldsymbol{f}^* \mid \boldsymbol{y}) &= \int p(\boldsymbol{f}^*, \boldsymbol{f} \mid \boldsymbol{y}) \, \mathrm{d}\boldsymbol{f} \\ &= \int p(\boldsymbol{f}^* \mid \boldsymbol{f}) p(\boldsymbol{f} \mid \boldsymbol{y}) \, \mathrm{d}\boldsymbol{f} \end{aligned}$$

combined with

$$p(\boldsymbol{f} \mid \boldsymbol{y}) \propto p(\boldsymbol{y} \mid \boldsymbol{f}) p(\boldsymbol{f}).$$

All densities appearing are Gaussian.

Lengthy calculation (Section 8.2.5 in RG) gives

$$\boldsymbol{f}^* \mid \boldsymbol{y} \sim N\left(\mu^*_{\text{post}}, \Sigma^*_{\text{post}}\right)$$

with

$$\mu^*_{\text{post}} = \mu^* + R^T(\mathcal{K} + \sigma^2 I)^{-1}(\boldsymbol{y} - \mu) \qquad \Sigma^*_{\text{post}} = \mathcal{K}^* - R^T(\mathcal{K} + \sigma^2 I)^{-1} R$$

Remarks

- Incorporating noise can be done directly in the kernel.
- Formulas look somewhat unwieldy. Need careful numerical linear algebra.
- Computationally intensive part: inversion of $n \times n$ matrix $\mathcal{K}$. This is $\mathcal{O}(n^3)$.
- Parameters in the kernel can be dealt with by either
  - use empirical Bayes to estimate these;
  - employ a prior on these and do MCMC.

Want to do this in Python? See for example `https://juanitorduz.github.io/gaussian_process_reg/` and `https://andrewcharlesjones.github.io/posts/2020/11/gaussian-processes/`

Advanced topics (not part of the exam)

Periodic kernel

Map 1-dimensional variable x to $u(x) := (\cos(x), \sin(x))$ and use the squared exponential kernel to get

$$\begin{aligned} K(x, \tilde{x}) &= v_0 \exp\left(-\frac{\|u(x) - u(\tilde{x})\|^2}{2\ell^2}\right) \\ &= v_0 \exp\left(\frac{2\sin^2((x-\tilde{x})/2)}{\ell^2}\right). \end{aligned}$$

This follows from

$$\|u(x) - u(\tilde{x})\|^2 = 4\sin^2\left(\frac{x-\tilde{x}}{2}\right).$$

Building kernels from simpler kernels

Slightly out of scope for this course, but

- sum of kernels $k_1(x, \tilde{x}) + k_2(x, \tilde{x})$ gives a kernel
- product of kernels $k_1(x, \tilde{x})k_2(x, \tilde{x})$ gives a kernel
- direct sums of kernels $k(x, \tilde{x}) = k_1(x_1, \tilde{x}_1) + k_2(x_2, \tilde{x}_2)$ gives a kernel
- tensor product $k(x, \tilde{x}) = k_1(x_1, \tilde{x}_1)k_2(x_2, \tilde{x}_2)$ gives a kernel

A Special kernel: Automatic Relevance Detection (ARD)

We can adjust the square exponential kernel to the **non-isotropic** case.

Let M be PSD and set

$$K(x, \tilde{x}) = v_0 \exp\left(-\frac{(x-\tilde{x})^T M (x-\tilde{x})}{2\ell^2}\right).$$

In particular, if $M = \mathrm{diag}(\gamma_1, \ldots, \gamma_p)$ then (taking $\ell = 1$)

$$K(x, \tilde{x}) = v_0 \exp\left(-\sum_{j=1}^{p} \gamma_j (x_j - \tilde{x}_j)^2\right).$$

Different dimensions in x get different weights! Idea goes back to MacKay and Neal (1993).

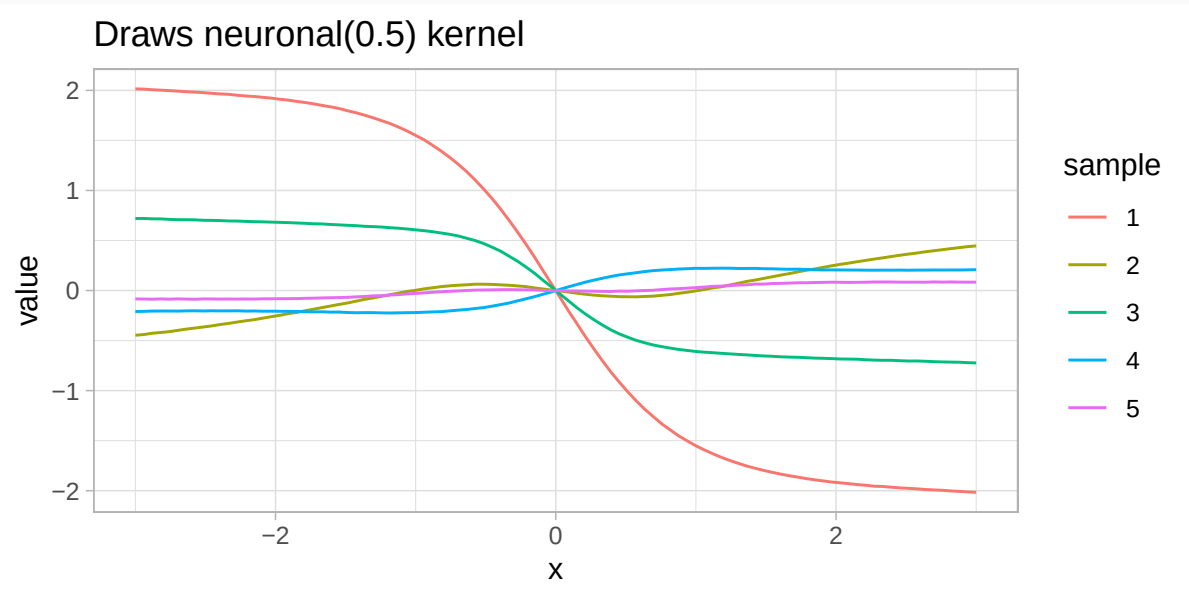
Nice Idea: include $\gamma_1, \ldots, \gamma_p$ as parameter and estimate these.

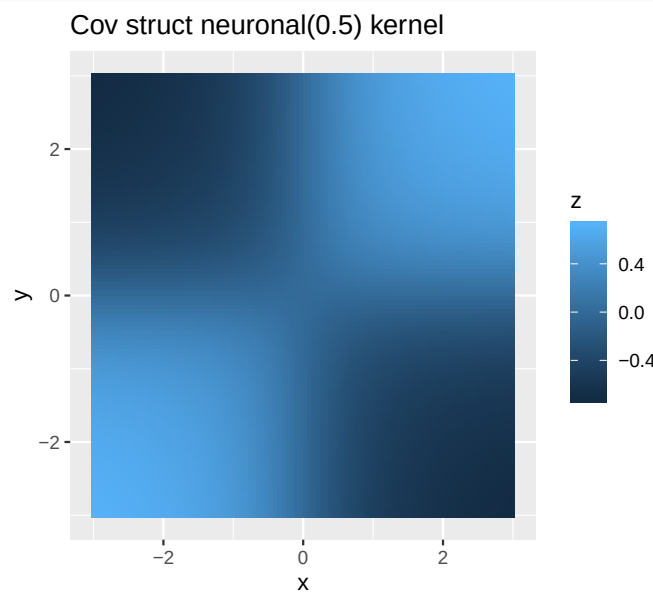
A Special kernel: neuronal kernel

$$k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{2}{\pi} \arcsin\left(\frac{a(\boldsymbol{x}, \tilde{\boldsymbol{x}})}{\sqrt{(1 + a(\boldsymbol{x}, \boldsymbol{x}))(1 + a(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}))}} \right)$$

where

$$a(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = 2\tilde{\boldsymbol{x}}^T \Sigma \boldsymbol{x}.$$





One layer perceptron network: neuronal kernel. RG section 8.2.6.3.

Assume $y \in \mathbb{R}$ and

$$\begin{aligned} y &= f(\boldsymbol{x}) \\ f(x) &= \epsilon + \sum_{j=1}^{N} v_j h(\boldsymbol{x}; u_j), \qquad h(x,u) = \operatorname{erf}(\boldsymbol{x}^T u) \\ u_1, \ldots, u_N &\overset{\text{ind}}{\sim} N(0, \Sigma), \end{aligned}$$

where $\operatorname{erf}(z) = 1/\sqrt{\pi} \int_{-z}^{z} e^{-t^2} \, \mathrm{d}t$.

- v_j: hidden-to-output weights
- $h(\boldsymbol{x}, u_j)$: hidden unit transfer functions

One layer perceptron network

Assume ϵ, $\{v_j\}$ are independent, zeromean, with variances σ_ϵ^2 and σ_v^2 respectively.

$$\mathbb{E}f(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x}) \mid \{u_j\}] = 0.$$

$$\begin{aligned}
\operatorname{Cov}(f(\boldsymbol{x}), f(\tilde{\boldsymbol{x}})) &= \mathbb{E}\left[f(\boldsymbol{x})f(\tilde{\boldsymbol{x}})\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[f(\boldsymbol{x})f(\tilde{\boldsymbol{x}}) \mid \{u_j\}_j\right]\right] \\
&= \sigma_\epsilon^2 + N\sigma_v^2\, \mathbb{E}_u\left[h(\boldsymbol{x}, u)h(\tilde{\boldsymbol{x}}, u)\right] \\
&= \sigma_\epsilon^2 + \omega^2\, \mathbb{E}_u\left[h(\boldsymbol{x}, u)h(\tilde{\boldsymbol{x}}, u)\right],
\end{aligned}$$

if $N\sigma_v^2 = \omega^2$.

Kernel for one layer perceptron network

Remains to compute

$$k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) := \mathbb{E}_u \left[h(\boldsymbol{x}, u) h(\tilde{\boldsymbol{x}}, u) \right]$$

with $h(\boldsymbol{x}, u) = \operatorname{erf}(\boldsymbol{x}^T u)$ and $u \sim N(0, \Sigma)$

Then

$$k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{2}{\pi} \arcsin \left(\frac{a(\boldsymbol{x}, \tilde{\boldsymbol{x}})}{\sqrt{(1 + a(\boldsymbol{x}, \boldsymbol{x}))(1 + a(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}))}} \right)$$

where

$$a(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = 2 \tilde{\boldsymbol{x}}^T \Sigma \boldsymbol{x}.$$

Something cool, *advanced*

- A popular class of kernels are called Matérn covariance functions:

$$K(x, y) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\,\frac{\tau}{\ell}\right)^{\nu} \mathcal{K}_\nu \left(\frac{\tau}{\ell}\right),$$

  where $\mathcal{K}_\nu$ is the modified Bessel function, $\tau = \|x - y\|$, and ν, σ, ℓ are the smoothness, magnitude and length scale parameters.
- Realisations can be obtained also as solution to a Stochastic Differential Equation (SDE).
- For example, if $\nu = 3/2$ then

$$\mathrm{d}f(t) = \begin{bmatrix} 0 & 1 \\ -\lambda^2 & -2\lambda \end{bmatrix} f(t)\,\mathrm{d}t + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathrm{d}W_t,$$

  with $\lambda = \sqrt{2\nu}/\ell$.
- Use Kalman smoothing to solve the interpolation problem: much lower computational cost! Not specific to Matérn covariance functions.

Hyperparameter estimation by empirical Bayes

Different ways to determine hyperparameters

1. Fully Baysian: Put a hyperprior on the hyperparameters.
2. Empirical Bayes: Compute the probability of the model given the data.
3. Compute the generalisation error (the average error on unseen test examples).
4. Bound the generalisation error.

Compute the probability of the model given the data

Assume σ fixed. As all is conditional on X, drop it from the notation.

$$\begin{aligned} y \mid f &\sim N(f, \sigma^2 I) \\ f &\sim N(0, \mathcal{K}) \end{aligned}$$

Then $y \sim N\left(0, \sigma^2 I + \mathcal{K}\right)$.

"Check":

$$\begin{aligned} \mathbb{E}\, y &= \mathbb{E}\, \mathbb{E}\left[y \mid f\right] = \mathbb{E}\, f = 0 \\ \operatorname{Cov} y &= \mathbb{E}\, \operatorname{Cov}\left(y \mid f\right) + \operatorname{Cov}\, \mathbb{E}\left[y \mid f\right] = \sigma^2 I + \mathcal{K}. \end{aligned}$$

Empirical Bayes

Marginal likelihood:

$$p(y;\sigma^2,\eta) = (2\pi)^{-\frac{n}{2}} |\det(\mathcal{K}_y)|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} y^T \mathcal{K}_y^{-1} y\right),$$

$$\log p(y;\sigma^2,\eta) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\det(\mathcal{K}_y)| - \frac{1}{2} y^T \mathcal{K}_y^{-1} y$$

with $\mathcal{K}_y = \sigma^2 I + \mathcal{K}$ and η the parameter vector of the kernel K.

Gradient methods can be used to optimise with respect to (σ^2, η). This is an **empirical Bayes** method.

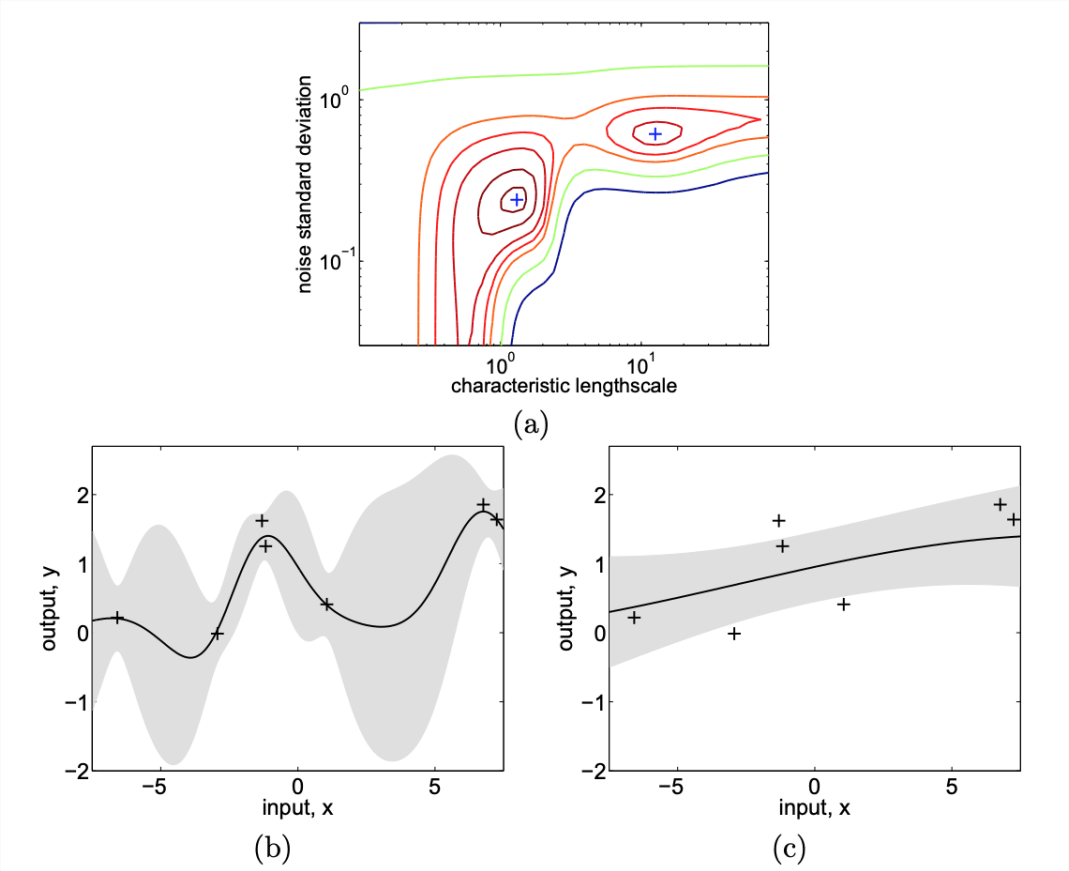


Figure 5.5: Panel (a) shows the marginal likelihood as a function of the hyperparameters ℓ (length-scale) and σ_n^2 (noise standard deviation), where $\sigma_f^2 = 1$ (signal standard deviation) for a data set of 7 observations (seen in panels (b) and (c)). There are two local optima, indicated with '+': the global optimum has low noise and a short length-scale; the local optimum has a high noise and a long length scale. In (b) and (c) the inferred underlying functions (and 95% confidence intervals) are shown for each of the two solutions. In fact, the data points were generated by a Gaussian process with $(\ell, \sigma_f^2, \sigma_n^2) = (1, 1, 0.1)$ in eq. (5.1).

Figure taken Rasmussen & Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006.

Gaussian process classification (RG section 8.3)

Gaussian process classification

- Let ψ denote the logistic function. Assume data $\{(x_i, y_i),\, i = 1, \ldots, n\}$ with $y_i \in \{0, 1\}$.
- Let X denote the design matrix.

Assume

$$\begin{aligned} y_1, \ldots, y_n \mid f, X &\overset{\text{ind}}{\sim} Ber\left(\psi\left(f(x_i)\right)\right) \\ f &\sim GP(0, K) \end{aligned}$$

(note that K is the kernel function).

Prediction under Gaussian process classification

Suppose new data $x_{\text{new}} = x^*$ and we wish to predict $y_{\text{new}} = y^*$.

Let $\boldsymbol{f} = (f_1, \ldots, f_n)$ with $f_i = f(x_i)$, $y = (y_1, \ldots, y_n)$ and $f^* = f(x^*)$.

Decompostion 1.

$$p(y^* \mid x^*, X, y) = \int p(y^* \mid f^*, x^*, X, y) p(f^* \mid x^*, X, y) \, \mathrm{d}f^*.$$

Decomposition 2.

$$p(f^* \mid x^*, X, y) = \int p(f^* \mid \boldsymbol{f}, x^*, X, y) p(\boldsymbol{f} \mid x^*, X, y) \, \mathrm{d}\boldsymbol{f}$$

(blue things can be "erased" by conditional independence). To sample from the predictive density:

(A) Sample $\boldsymbol{f}$ from the posterior $p(\boldsymbol{f} \mid X, y)$.
(B) Sample f^* from $p(f^* \mid \boldsymbol{f}, x^*)$.
(C) Sample $y^* \sim Ber\left(\psi\left(f^*\right)\right)$.

Step A: Sample f from the posterior $p(f \mid X, y)$

The likelihood is

$$p(y \mid X, f) = \prod_{i=1}^{n} \psi(f_i)^{y_i} (1 - \psi(f_i))^{1-y_i}$$

The prior is

$$\boldsymbol{f} \sim N_n(0, \mathcal{K}) \qquad \text{with} \qquad \mathcal{K}_{ij} = K(x_i, x_j).$$

The structure of the problem is exactly as in "ordinary" logistic regression; only the prior covariance is more sophisticated.

In ordinary logistic regression we would have $f_i = \theta^T x_i$ and prior $\theta \sim N_p(0, \Sigma)$.

Step B: Sample f^* from $p(f^* \mid \boldsymbol{f}, x^*)$

Note that (notation is somewhat sloppy here)

$$\begin{bmatrix} \boldsymbol{f} \\ f^* \end{bmatrix} \sim N\left(\begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} (K(x_i, x_j))_{i,j} & (K(x_i, x^*))_i \\ (K(x_i, x^*))_i & K(x^*, x^*) \end{bmatrix} \right).$$

This implies $f^* \mid \boldsymbol{f}, x^*$ is normally distributed.

See Comment 8.2 in RG. *This is why the multivariate normal distribution is so convenient.*

Step C: Sample $Y^* \sim Ber\left(\psi\left(f^*\right)\right)$

This is really simple: once we have f^* we compute $\psi(f^*)$ and draw from the Bernoulli distribution.

Remarks on computational effort

- Step A cannot be done in closed form. For Bayesian logistic regression one option is to use the MH-algorithm.
- Note that $\boldsymbol{f} \in \mathbb{R}^n$, so we need the MH-algorithm in dimension n, rather than dimension p (if $\theta \in \mathbb{R}^p$ for ordinary logistic regression).
- MCMC in high dimension is a very active topic of research.
- For step B, we need to invert the $n \times n$ matrix $(K(x_i, x_j))_{i,j}$. The computational cost is $\mathcal{O}(n^3)$.

Gibbs sampling for GP classification (RG section 9.2)

Gibbs sampling for GP classification (RG section 9.2)

Assume the **probit-regression** model:

$$\begin{aligned} y_1, \ldots, y_n \mid f, X &\overset{\text{ind}}{\sim} Ber\left(\Phi\left(f(x_i)\right)\right) \\ f &\sim GP(0, K) \end{aligned}$$

with

$$\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \, \mathrm{d}x.$$

With Φ as *inverse-link* function we can avoid tuning MH-steps needed in step A. Instead: use Gibbs sampling.

Turning the model in parametric form

Rewrite the model, using $f_i := f(x_i)$:

$$\begin{aligned} y_1, \ldots, y_n \mid f, X &\overset{\text{ind}}{\sim} Ber\left(\Phi\left(f_i\right)\right) \\ f &\sim N\left(0, \mathcal{K}\right) \end{aligned}$$

with $\mathcal{K}_{i,j} = K(x_i, x_j)$.

First objective is to sample from $f \mid y$, where $y = (y_1, \ldots, y_n)$.

Introducing auxiliary variables

Trick: introduce auxiliary variables $z_1, \ldots, z_n$ such that

$$\begin{aligned} y_i &= \mathbf{1}\{z_i \geq 0\} \\ z_i \mid f_i &\overset{\text{ind}}{\sim} N(f_i, 1). \end{aligned}$$

This gives the correct Bernoulli distribution for $y_i \mid f_i$ since

$$\mathrm{P}(y_i = 1 \mid f_i) = \mathrm{P}(z_i \geq 0 \mid f_i) = \mathrm{P}\Big(\underbrace{z_i - f_i}_{N(0,1)} \geq -f_i \mid f_i \Big) = \Phi(f_i).$$

[Use that for $Z \sim N(0,1)$ we have $\mathrm{P}(Z \geq -z) = \mathrm{P}(Z \leq z) = \Phi(z)$.]

In the following let $z = (z_1, \ldots, z_n)$ and let $\varphi_n(x; \mu, \Sigma)$ denote the density of the $N(\mu, \Sigma)$-distribution, evaluated at x.

Gibbs sampler (1/2)

Structure:

$$f_i \longrightarrow z_i \longrightarrow y_i.$$

Gibbs sampling for

$$\begin{aligned} p(z, f \mid y) &\propto p(y, z, f) = p(y \mid z)p(z \mid f)p(f) \\ &= p(f)p(z \mid f) \prod_{i=1}^{n} p(y_i \mid z_i). \end{aligned}$$

- $p(f) = \varphi_n(f; 0, \mathcal{K})$, where $\mathcal{K}_{ij} = K(x_i, x_j)$ (K is kernel from Gaussian process),
- $p(z \mid f) = \prod_{i=1}^{n} \varphi(z_i; f_i, 1) = \varphi_n(z; f, I_n)$,
- $p(y_i \mid z_i) = \alpha_i^{y_i}(1 - \alpha_i)^{1-y_i}$ with $\alpha_i = \mathbf{1}_{\{z_i \geq 0\}}$ (because $y_i \sim Ber(\alpha_i)$).

Gibbs sampler (2/2)

- Updating f:

$$p(f \mid y, z) \propto \varphi_n(f; 0, \mathcal{K})\varphi_n(z; f, I_n).$$

  Conjugate step: amounts to drawing from the multivariate normal distribution.[1]

- Updating z_i can be done independently:

$$p(z_i \mid y, f) \propto \alpha_i^{y_i}(1-\alpha_i)^{1-y_i}\varphi(z_i; f_i, 1)$$

  This means

  - If $y_i = 1$, draw z_i from $N(f_i, 1)$-distribution, conditioned to be positive.
  - If $y_i = 0$, draw z_i from $N(f_i, 1)$-distribution, conditioned to be negative.

[1] In fact, conditional on z, y plays no role and $f \mid y, z \sim N(\mu_f, \Sigma_f)$, where $\mu_f = \Sigma_f z$ and $\Sigma_f = (\mathcal{K}^{-1} + I)^{-1}$.