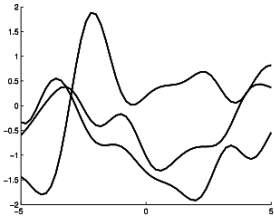
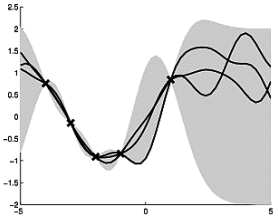


(a)



(b)

Figure 15.2 Left: some functions sampled from a GP prior with SE kernel. Right: some samples from a GP posterior, after conditioning on 5 noise-free observations. The shaded area represents $\mathbb{E}[f(\mathbf{x})] \pm 2\text{std}(f(\mathbf{x}))$. Based on Figure 2.2 of (Rasmussen and Williams 2006). Figure generated by `gprDemoNoiseFree`.

15.2.1 Predictions using noise-free observations

Suppose we observe a training set $\mathcal{D} = \{(\mathbf{x}_i, f_i), i = 1 : N\}$, where $f_i = f(\mathbf{x}_i)$ is the noise-free observation of the function evaluated at $\mathbf{x}_i$. Given a test set $\mathbf{X}_*$ of size $N_* \times D$, we want to predict the function outputs $\mathbf{f}_*$.

If we ask the GP to predict $f(\mathbf{x})$ for a value of $\mathbf{x}$ that it has already seen, we want the GP to return the answer $f(\mathbf{x})$ with no uncertainty. In other words, it should act as an **interpolator** of the training data. This will only happen if we assume the observations are noiseless. We will consider the case of noisy observations below.

Now we return to the prediction problem. By definition of the GP, the joint distribution has the following form

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix} \right) \tag{15.6}$$

where $\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X})$ is $N \times N$, $\mathbf{K}_* = \kappa(\mathbf{X}, \mathbf{X}_*)$ is N

$\times N_*$, and $\mathbf{K}_{**} = \kappa(\mathbf{X}_*, \mathbf{X}_*)$ is $N_* \times N_*$.
By the standard rules for conditioning Gaussians (Section 4.3), the posterior has the following form

$$p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{f}) = \mathcal{N}(\mathbf{f}_*|\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \tag{15.7}$$

$$\boldsymbol{\mu}_* = \boldsymbol{\mu}(\mathbf{X}_*) + \mathbf{K}_*^T\mathbf{K}^{-1}(\mathbf{f} - \boldsymbol{\mu}(\mathbf{X})) \tag{15.8}$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^T\mathbf{K}^{-1}\mathbf{K}_* \tag{15.9}$$

This process is illustrated in Figure 15.2. On the left we show sample samples from the prior, $p(\mathbf{f}|\mathbf{X})$, where we use a **squared exponential kernel**, aka Gaussian kernel or RBF kernel. In 1d, this is given by

$$\kappa(x, x') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}(x - x')^2) \tag{15.10}$$

Here ℓ controls the horizontal length scale over which the function varies, and σ_f^2 controls the vertical variation. (We discuss how to estimate such kernel parameters below.) On the right we

show samples from the posterior, $p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{f})$. We see that the model perfectly interpolates the training data, and that the predictive uncertainty increases as we move further away from the observed data.

One application of noise-free GP regression is as a computationally cheap proxy for the behavior of a complex simulator, such as a weather forecasting program. (If the simulator is stochastic, we can define f to be its mean output; note that there is still no observation noise.) One can then estimate the effect of changing simulator parameters by examining their effect on the GP's predictions, rather than having to run the simulator many times, which may be prohibitively slow. This strategy is known as DACE, which stands for design and analysis of computer experiments (Santner et al. 2003).

15.2.2 Predictions using noisy observations

Now let us consider the case where what we observe is a noisy version of the underlying function, $y = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_y^2)$. In this case, the model is not required to interpolate the data, but it must come "close" to the observed data. The covariance of the observed noisy responses is

$$\text{cov}\left[y_p, y_q\right] = \kappa(\mathbf{x}_p, \mathbf{x}_q) + \sigma_y^2 \delta_{pq} \tag{15.11}$$

where $\delta_{pq} = \mathbb{I}(p = q)$. In other words

$$\text{cov}\left[\mathbf{y}|\mathbf{X}\right] = \mathbf{K} + \sigma_y^2 \mathbf{I}_N \triangleq \mathbf{K}_y \tag{15.12}$$

The second matrix is diagonal because we assumed the noise terms were independently added to each observation.

The joint density of the observed data and the latent, noise-free function on the test points is given by

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K}_y & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix}\right) \tag{15.13}$$

where we are assuming the mean is zero, for notational simplicity. Hence the posterior predictive

density is

$$p(\mathbf{f}_*|\mathbf{X}_*, \quad \mathbf{X}, \mathbf{y}) \quad = \quad \mathcal{N}(\mathbf{f}_*|\boldsymbol{\mu}_*, \quad \boldsymbol{\Sigma}_*)$$

(15.14)

$$\boldsymbol{\mu}_* \quad = \quad \mathbf{K}_*^T \mathbf{K}_y^{-1} \mathbf{y}$$

(15.15)

$$\boldsymbol{\Sigma}_* \quad = \quad \mathbf{K}_{**} \quad - \mathbf{K}_*^T \mathbf{K}_y^{-1} \mathbf{K}_*$$

(15.16) In the case of a single test input, this simplifies as follows

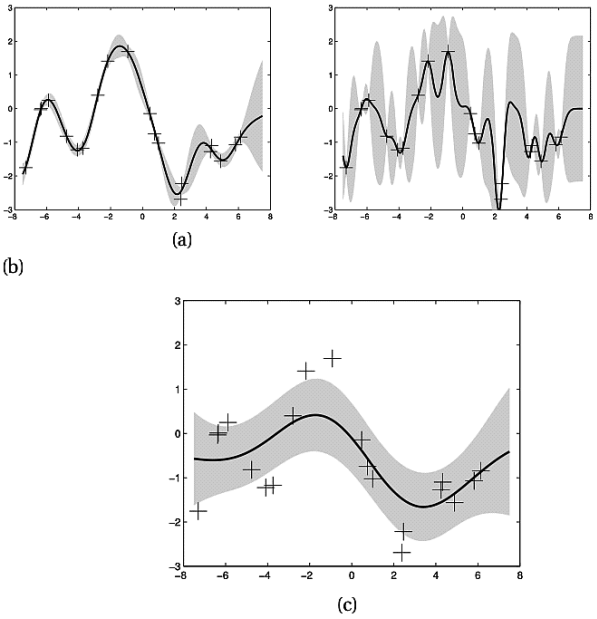
$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) \quad = \quad \mathcal{N}(f_*|\mathbf{k}_*^T \mathbf{K}_y^{-1} \mathbf{y}, k_{**} - \mathbf{k}_*^T \mathbf{K}_y^{-1} \mathbf{k}_*)$$

(15.17)

where $\mathbf{k}_* = [\kappa(\mathbf{x}_*, \mathbf{x}_1), \ldots, \kappa(\mathbf{x}_*, \mathbf{x}_N)]$ and $k_{**} = \kappa(\mathbf{x}_*, \mathbf{x}_*)$. Another way to write the posterior mean is as follows:

$$\overline{f}_* = \mathbf{k}_*^T \mathbf{K}_y^{-1} \mathbf{y} = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_*)$$

(15.18) where $\boldsymbol{\alpha} = \mathbf{K}_y^{-1} \mathbf{y}$. We will revisit this expression later.



(a)

(b)

(c)

Figure 15.3 Some 1d GPs with SE kernels but different hyper-parameters fit to 20 noisy observations. The kernel has the form in Equation 15.19. The hyper-parameters $(\ell, \sigma_f, \sigma_y)$ are as follows: (a) (1,1,0.1) (b) (0.3, 0.1.08, 0.00005), (c) (3.0, 1.16, 0.89). Based on Figure 2.5 of (Rasmussen and Williams 2006). Figure generated by `gprDemoChangeHparams`, written by Carl Rasmussen.

15.2.3 Effect of the kernel parameters

The predictive performance of GPs depends exclusively on the suitability of the chosen kernel. Suppose we choose the following squared-exponential (SE) kernel for the noisy observations

$$\kappa_y(x_p, x_q) = \sigma_f^2 \exp(-\frac{1}{2\ell^2}(x_p - x_q)^2) + \sigma_y^2 \delta_{pq} \quad (15.19)$$

Here ℓ is the horizontal scale over which the function changes, σ_f^2 controls the vertical scale of the function, and σ_y^2 is the noise variance. Figure 15.3 illustrates the effects of changing these parameters. We sampled

20 noisy data points from the SE kernel using $(\ell, \sigma_f, \sigma_y) = (1, 1, 0.1)$, and then made predictions various parameters, conditional on the data. In Figure 15.3(a), we use $(\ell, \sigma_f, \sigma_y) = (1, 1, 0.1)$, and the result is a good fit. In Figure 15.3(b), we reduce the length scale to $\ell = 0.3$ (the other parameters were optimized by maximum (marginal) likelihood, a technique we discuss below); now the function looks more "wiggly". Also, the uncertainty goes up faster, since the effective distance from the training points increases more rapidly. In Figure 15.3(c), we increase the length scale to $\ell = 3$; now the function looks smoother.

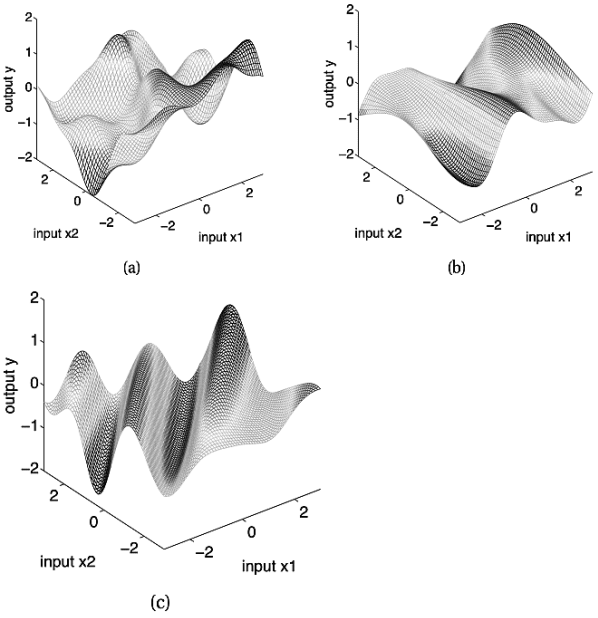


Figure 15.4 Some 2d functions sampled from a GP with an SE kernel but different hyper-parameters. The kernel has the form in Equation 15.20 where (a) $\mathbf{M} = \mathbf{I}$, (b) $\mathbf{M} = \text{diag}(1, 3)^{-2}$, (c) $\mathbf{M} = (1, -1; -1, 1) + \text{diag}(6, 6)^{-2}$. Based on Figure 5.1 of (Rasmussen and Williams 2006). Figure generated by `gprDemoArd`, written by Carl Rasmussen.

We can extend the SE kernel to multiple dimensions as follows:

$$\kappa_y(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^T \mathbf{M}(\mathbf{x}_p - \mathbf{x}_q)) + \sigma_y^2 \delta_{pq} \quad (15.20)$$

We can define the matrix $\mathbf{M}$ in several ways. The simplest is to use an isotropic matrix, $\mathbf{M}_1 = \ell^{-2}\mathbf{I}$. See Figure 15.4(a) for an example. We can also endow each dimension with its own characteristic length scale, $\mathbf{M}_2 = \text{diag}(\boldsymbol{\ell})^{-2}$. If any of these length scales become large, the corresponding feature dimension is deemed "irrelevant", just as in ARD (Section 13.7). In Figure 15.4(b), we use $\mathbf{M} = \mathbf{M}_2$ with $\boldsymbol{\ell} = (1, 3)$, so the

function changes faster along the x_1 direction than the x_2 direction.

We can also create a matrix of the form $\mathbf{M}_3 = \mathbf{\Lambda}\mathbf{\Lambda}^T + \text{diag}(\boldsymbol{\ell})^{-2}$, where $\mathbf{\Lambda}$ is a $D \times K$ matrix, where $K < D$. (Rasmussen and Williams 2006, p107) calls this the **factor analysis distance** function, by analogy to the fact that factor analysis (Section 12.1) approximates a covariance matrix as a low rank matrix plus a diagonal matrix. The columns of $\mathbf{\Lambda}$ correspond to relevant directions in input space. In Figure 15.4(c), we use $\boldsymbol{\ell} = (6; 6)$ and $\mathbf{\Lambda} = (1; -1)$, so the function changes mostly rapidly in the direction which is perpendicular to (1,1).

15.2.4 Estimating the kernel parameters

To estimate the kernel parameters, we could use exhaustive search over a discrete grid of values, with validation loss as an objective, but this can be quite slow. (This is the approach used to tune kernels used by SVMs.) Here we consider an empirical Bayes approach, which will allow us to use continuous optimization methods, which are much faster. In particular, we will maximize the marginal likelihood[1]

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{X})p(\mathbf{f}|\mathbf{X})d\mathbf{f}$$

(15.21) Since $p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$, and $p(\mathbf{y}|\mathbf{f}) = \prod_i \mathcal{N}(y_i|f_i, \sigma_y^2)$, the marginal likelihood is given by

$$\log p(\mathbf{y}|\mathbf{X}) = \log \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_y) = -\frac{1}{2}\mathbf{y}\mathbf{K}_y^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_y| - \frac{N}{2}\log(2\pi)$$

(15.22) The first term is a data fit term, the second term is a model complexity term, and the third term is just a constant. To understand the tradeoff between the first two terms, consider a SE kernel in 1D, as we vary the length scale ℓ and hold σ_y^2 fixed. Let $J(\ell) = -\log p(\mathbf{y}|\mathbf{X}, \ell)$. For short length scales, the fit will be good, so $\mathbf{y}^T\mathbf{K}_y^{-1}\mathbf{y}$ will be small. However, the model complexity will be high: $\mathbf{K}$ will be almost diagonal (as in Figure 14.3, top right), since most points will not be considered "near" any others, so the $\log|\mathbf{K}_y|$ will be large. For long length scales, the fit will be poor but the model complexity will be low: $\mathbf{K}$ will be almost all 1's (as in Figure 14.3, bottom right), so $\log|\mathbf{K}_y|$ will be small.

We now discuss how to maximize the marginal likelhood. Let the kernel parameters (also called hyper-parameters) be denoted by $\boldsymbol{\theta}$. One can show that

$$\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|\mathbf{X}) = \frac{1}{2}\mathbf{y}^T\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_j}\mathbf{K}_y^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}(\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_j})$$

(15.23)

$$= \frac{1}{2}\text{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \mathbf{K}_y^{-1})\frac{\partial \mathbf{K}_y}{\partial \theta_j}\right) \quad (15.24)$$

where $\boldsymbol{\alpha} = \mathbf{K}_y^{-1}\mathbf{y}$. It takes $O(N^3)$ time to compute $\mathbf{K}_y^{-1}$, and then $O(N^2)$ time per hyperparameter to compute the gradient.

The form of $\frac{\partial \mathbf{K}_y}{\partial \theta_j}$ depends on the form of the kernel, and which parameter we are taking derivatives with respect to. Often we have constraints on the hyper-parameters, such as $\sigma_y^2 \geq 0$. In this case, we can define $\theta = \log(\sigma_y^2)$, and then use the chain rule.

Given an expression for the log marginal likelihood and its derivative, we can estimate the kernel parameters using any standard gradient-based optimizer. However, since the objective is not convex, local minima can be a problem, as we illustrate below.

15.2.4.1 Example

Consider Figure 15.5. We use the SE kernel in Equation 15.19 with $\sigma_f^2 = 1$, and plot $\log p(\mathbf{y}|\mathbf{X}, \ell, \sigma_y^2)$ (where $\mathbf{X}$ and $\mathbf{y}$ are the 7 data points shown in panels b and c) as we vary ℓ and σ_y^2. The two

1. The reason it is called the marginal likelihood, rather than just likelihood, is because we have marginalized out the latent Gaussian vector $\mathbf{f}$. This moves us up one level of the Bayesian hierarchy, and reduces the chances of overfitting (the number of kernel parameters is usually fairly small compared to a standard parametric model).

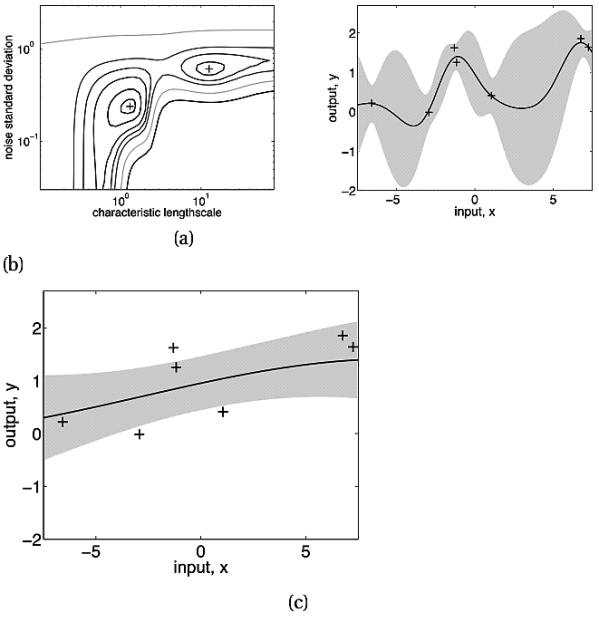


Figure 15.5 Illustration of local minima in the marginal likelihood surface. (a) We plot the log marginal likelihood vs σ_y^2 and ℓ, for fixed $\sigma_f^2 = 1$, using the 7 data points shown in panels b and c. (b) The function corresponding to the lower left local minimum, $(\ell, \sigma_n^2) \approx (1, 0.2)$. This is quite "wiggly" and has low noise. (c) The function corresponding to the top right local minimum, $(\ell, \sigma_n^2) \approx (10, 0.8)$. This is quite smooth and has high noise. The data was generated using $(\ell, \sigma_n^2) = (1, 0.1)$. Source: Figure 5.5 of (Rasmussen and Williams 2006). Figure generated by `gprDemoMarglik`, written by Carl Rasmussen.

local optima are indicated by +. The bottom left optimum corresponds to a low-noise, shortlength scale solution (shown in panel b). The top right optimum corresponds to a high-noise, long-length scale solution (shown in panel c). With only 7 data points, there is not enough evidence to confidently decide which is more reasonable, although the more complex model (panel b) has a marginal likelihood that is about 60% higher than the simpler model (panel c). With more data, the MAP estimate should come to dominate.

Figure 15.5 illustrates some other interesting (and typical) features. The region where $\sigma_y^2 \approx 1$ (top of panel a) corresponds to the case where the noise is very high; in this regime, the marginal likelihood is insensitive to the length scale (indicated by the horizontal contours), since all the data is explained as noise. The region where $\ell \approx 0.5$ (left hand side of panel a) corresponds to the case where the length scale is very short; in this regime, the marginal likelihood is insensitive to the noise level, since the data is perfectly interpolated. Neither of these regions would be chosen by a good optimizer.