

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Figure 15.1 A Gaussian process for 2 training points and 1 testing point, represented as a mixed directed and undirected graphical model representing  $p(y, f|x) = N(f|0, K(x)) \prod p(y_i|f_i)$ . The hidden nodes  $f_i = f(x_i)$  represent the value of the function at each of the data points. These hidden nodes are fully interconnected by undirected edges, forming a Gaussian graphical model; the edge strengths represent the covariance terms  $\Sigma_{ij} = \kappa(x_i, x_j)$ . If the test point  $x_*$  is similar to the training points  $x_1$  and  $x_2$ , then the predicted output  $y_*$  will be similar to  $y_1$  and  $y_2$ .

## 15 Gaussian processes

### 15.1 Introduction

In supervised learning, we observe some inputs  $x_i$  and some outputs  $y_i$ . We assume that  $y_i = f(x_i)$ , for some unknown function  $f$ , possibly corrupted by noise. The optimal approach is to infer a distribution over functions given the data,  $p(f|X, y)$ , and then to use this to make predictions given new inputs, i.e., to compute

$$p(y_*|x_*, X, y) = p(y_*|f, x_*)p(f|X, y)df \quad (15.1)$$

Up until now, we have focussed on parametric representations for the function  $f$ , so that instead of inferring  $p(f|D)$ , we infer  $p(\theta|D)$ . In this chapter, we discuss a way to perform Bayesian inference over functions themselves.

Our approach will be based on Gaussian processes or GPs. A GP defines a prior over functions, which can be converted into a posterior over functions once we have

seen some data. Although it might seem difficult to represent a distribution over a function, it turns out that we only need to be able to define a distribution over the function's values at a finite, but arbitrary, set of points, say  $x_1, \dots, x_N$ . A GP assumes that  $p(f(x_1), \dots, f(x_N))$  is jointly Gaussian, with some mean  $\mu(x)$  and covariance  $\Sigma(x)$  given by  $\Sigma_{ij} = \kappa(x_i, x_j)$ , where  $\kappa$  is a positive definite kernel function (see Section 14.2 information on kernels). The key idea is that if  $x_i$  and  $x_j$  are deemed by the kernel to be similar, then we expect the output of the function at those points to be similar, too. See Figure 15.1 for an illustration. It turns out that, in the regression setting, all these computations can be done in closed form, in  $O(N^3)$  time. (We discuss faster approximations in Section 15.6.) In the classification setting, we must use approximations, such as the Gaussian approximation, since the posterior is no longer exactly Gaussian.

GPs can be thought of as a Bayesian alternative to the kernel methods we discussed in Chapter 14, including L1VM, RVM and SVM. Although those methods are sparser and therefore faster, they do not give well-calibrated probabilistic outputs (see Section 15.4.4 for further discussion). Having properly tuned probabilistic output is important in certain applications, such as online tracking for vision and robotics (Ko and Fox 2009), reinforcement learning and optimal control (Engel et al. 2005; Deisenroth et al. 2009), global optimization of non-convex functions (Mockus et al. 1996; Lizotte 2008; Brochu et al. 2009), experiment design (Santner et al. 2003), etc. Our presentation is closely based on (Rasmussen and Williams 2006), which should be consulted for further details. See also (Diggle and Ribeiro 2007), which discusses the related approach known as kriging, which is widely used in the spatial statistics literature.

已思考 33s

## 翻译 (逐句不遗漏)

图 15.1 两个训练点与一个测试点的高斯过程 (Gaussian process)。它被表示为一个“有向一无向混合”的图模型，用来刻画

$$p(\mathbf{y}, \mathbf{f} \mid \mathbf{x}) = N(\mathbf{f} \mid \mathbf{0}, \mathbf{K}(\mathbf{x})) \prod_i p(y_i \mid f_i).$$

隐藏节点  $f_i = f(x_i)$  表示函数在每个数据点处的取值。这些隐藏节点通过无向边完全相互

连接，构成一个高斯图模型；边的强度代表协方差项  $\Sigma_{ij} = \kappa(x_i, x_j)$ 。如果测试点  $x^{*}$  与训练点  $x_1$  和  $x_2$  相似，那么预测输出  $y^{*}$  将与  $y_1$  和  $y_2$  相似。

## 15 高斯过程

### 15.1 引言

在监督学习中，我们观测到一些输入  $x_i$  和一些输出  $y_i$ 。我们假设  $y_i = f(x_i)$ ，其中  $f$  是某个未知函数（可能再加上噪声）。最优的方法是：在给定数据的情况下，对函数的**分布**进行推断，即求  $p(f | X, y)$ ，然后用它在新输入下进行预测，也就是计算

$$p(y^{*} | x^{*}, X, y) = \int p(y^{*} | f, x^{*}) p(f | X, y) df \quad (15.1)$$

到目前为止，我们一直专注于函数  $f$  的**参数化**表示，因此并不是推断  $p(f | D)$ ，而是推断  $p(\theta | D)$ 。本章讨论一种直接对**函数本身**进行贝叶斯推断的方法。

我们的方法基于**高斯过程 (GP)**。GP 为函数定义了一个**先验**，在看到数据之后可以把它转化为对函数的**后验**。虽然看起来“给一个函数定义分布”很难，但事实是我们只需要能在某个有限（但可任意选取）的点集  $x_1, \dots, x_N$  上，定义该函数取值的联合分布即可。GP 假设  $(f(x_1), \dots, f(x_N))$  服从**联合高斯分布**，其均值为  $\mu(x)$ ，协方差  $\Sigma(x)$  由  $\Sigma_{ij} = \kappa(x_i, x_j)$  给出，其中  $\kappa$  是**正定核函数**（有关核的内容见第 14.2 节）。关键思想是：如果核认为  $x_i$  与  $x_j$  相似，那么函数在这两个点的输出也应当相似。参见图 15.1 的示意。

事实证明，在**回归**情形下，所有这些计算都可以**解析**完成，时间复杂度为  $O(N^3)$ 。（更快的近似算法见 15.6 节。）在**分类**情形下，由于后验不再是精确的高斯分布，我们必须使用近似方法，例如**高斯近似**。

可以把 GP 看作第 14 章介绍的核方法（包括 L1VM、RVM 和 SVM）的**贝叶斯替代方案**。虽然这些方法更稀疏、因此更快，但它们不能给出**校准良好的**概率输出（更多讨论见 15.4.4 节）。在一些应用中，得到调得合适的概率输出非常重要，例如视觉与机器人中的在线跟踪（Ko 和 Fox, 2009）、强化学习与最优控制（Engel 等, 2005; Deisenroth 等, 2009）、非凸函数的全局优化（Mockus 等, 1996; Lizotte, 2008; Brochu 等, 2009）、实验设计（Santner 等, 2003）等。

本章的讲解主要参考 Rasmussen 与 Williams (2006)，欲知更多细节请参阅其著作。另见 Diggle 与 Ribeiro (2007)，他们讨论了一种相关方法**克里金 (kriging)**，该方法在空间统计文献中被广泛使用。

## 通俗解释

## 1) 高斯过程到底是什么?

把一个函数想象成“在所有输入点上的无限长的向量”。GP 的核心做法是：不用直接处理这条“无限向量”，而是说——**只要你挑出任意有限个输入点  $x_1, \dots, x_N$** ，我就认为这些点上的函数值 ( $f(x_1), \dots, f(x_N)$ ) 共同服从一个**多元高斯分布**。这个分布由两部分决定：

- **均值函数  $\mu(x)$** : 通常取 0, 表示先验上函数围绕 0 摆动；
- **协方差 (核) 函数  $\kappa(x, x')$** : 衡量两个输入的**相似度**，相似度越高，两个位置上的函数值越容易接近。

## 2) 图 15.1 在表达什么?

- 每个白色圆圈  $f_i$  是“函数在输入  $x_i$  处的真实值”；灰色圆圈  $y_i$  是我们观测到的输出（可带噪声）。
- 竖直的**有向边**  $f_i \rightarrow y_i$  表示“观测由函数值生成”（即  $p(y_i | f_i)$ ）。
- 各个  $f_i$  之间有**无向边并且完全相连**，这在概率图里表示“它们联合服从高斯分布”，边的强度就是协方差  $\Sigma_{ij} = \kappa(x_i, x_j)$ 。
- 测试点  $x^{*}$  也接入同样的网络：如果它和  $x_1, x_2$  在核意义上很相似，那么它的函数值  $f^{*}$  以及最终的预测  $y^{*}$  就会与  $y_1, y_2$  相似。直观地说：**相似的输入→相似的输出**。

## 3) 预测时在做什么积分 (式 15.1) ?

我们先根据训练数据求出“函数整体的后验分布”  $p(f | X, y)$ 。预测  $y^{*}$  时，把所有“可能的函数”都考慮到：对每个可能的  $f$ ，它会给出一个  $p(y^{*} | f, x^{*})$ ；然后对这些可能性做加权平均（积分）。这就是式 (15.1)。

## 4) 为什么回归能“闭式”算、复杂度 $O(N^3)$ ?

回归通常假设高斯噪声，此时“先验是高斯 + 似然是高斯 → 后验仍是高斯”，所有量都能写成矩阵公式。计算里要对  $N \times N$  的协方差矩阵做**矩阵分解/求逆**，所以复杂度是  $O(N^3)$ 。当样本很多时，这会变慢，于是有各种近似（15.6 节）。

## 5) 分类为什么要近似？

分类的观测  $y_i$  常是离散标签（用逻辑回归/Probit 等链接函数），这时  $p(y_i | f_i)$  不是高斯，导致后验不再是高斯，没法直接写解析式，只能用拉普拉斯近似、期望传播、变分法等“把它近似成高斯”的办法。

## 6) 和 SVM/RVM 等核方法的关系？

它们都用核来度量相似度，但：

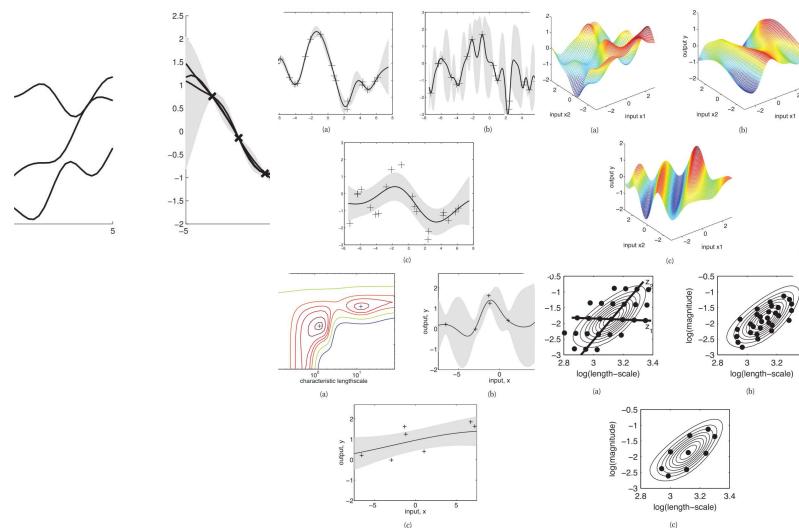
- SVM/RVM 更稀疏、预测更快；
- GP 是全贝叶斯：直接产出**带不确定性的概率预测**，而且这些概率通常**校准更好**（说 70% 的概率就真的差不多发生 70% 次）。在需要可靠概率的场景（在线跟踪、强化学习、全局优化、试验设计）中，GP 很吃香。

## 7) “先验/后验 over 函数”到底意味着什么？

不是在给某几个参数设先验，而是直接规定“函数在任意点集上的取值联合分布”。这让我们能在没有固定参数维度的情况下做推断，属于**非参数贝叶斯方法**。

# 要点总结

- GP 定义：**对函数建立高斯先验；在任意有限点集上，函数值联合高斯，协方差由核  $\kappa(x, x')$  给出。
- 直觉：**输入越相似（核越大），输出越相似；预测点会“参考”相似训练点的输出。
- 图模型视角：**  $f_i$  两两无向相连表示“联合高斯”， $f_i \rightarrow y_i$  的有向边表示“由函数值生成观测”。整体联合分布  $N(\mathbf{f} \mid 0, \mathbf{K}) \prod_i p(y_i \mid f_i)$ 。
- 回归：**高斯噪声  $\rightarrow$  一切闭式可解，复杂度  $O(N^3)$ （主要来自对协方差矩阵的分解/求逆）。
- 分类：**似然非高斯  $\rightarrow$  需要近似（拉普拉斯、EP、变分等）。
- 与核方法比较：** SVM/RVM 更快更稀疏；GP 提供**校准良好**的概率输出，适合需要不确定性评估的任务。
- 应用：**跟踪、强化学习与控制、全局优化、实验设计等。
- 参考：**主要依据 Rasmussen & Williams (2006)；相关领域“克里金”在空间统计中广泛使用。



翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

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  $E[f(x)] \pm 2\text{std}(f(x))$ . Based on Figure 2.2 of (Rasmussen and Williams 2006)

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 ( $\sigma_f, \sigma_y$ ) are as follows: (a) (1, 1, 0.1) (b) (0.3, 0.1, 0.00005), (c) (3.0, 1.16, 0.89). Based on Figure 2.5 of (Rasmussen and Williams 2006).

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)  $M = I$ , (b)  $M = \text{diag}(1, 3) - 2$ , (c)  $M = (1, -1; -1, 1) + \text{diag}(6, 6) - 2$ . Based on Figure 5.1 of (Rasmussen and Williams 2006).

Figure 15.5 Illustration of local minima in the marginal likelihood surface. (a) We plot the log marginal likelihood vs  $\sigma^2_y$  and  $\sigma^2_f$ , for fixed  $\sigma^2_f = 1$ , using the 7 data points shown in panels b and c. (b) The function corresponding to the lower left local minimum,  $(\sigma^2_y, \sigma^2_f) \approx (1, 0.2)$ . This is quite “wiggly” and has low noise. (c) The function corresponding to the top right local minimum,  $(\sigma^2_y, \sigma^2_f) \approx (10, 0.8)$ . This is quite smooth and has high noise. The data was generated using  $(\sigma^2_y, \sigma^2_f) = (1, 0.1)$ .

Figure 15.6 Three different approximations to the posterior over hyper-parameters: grid-based, Monte Carlo, and central composite design.

## 15.2 GPs for regression

In this section, we discuss GPs for regression. Let the prior on the regression function be a GP, denoted by  $f(x) \sim \text{GP}(m(x), \kappa(x, x))$  (15.2)

where  $m(x)$  is the mean function and  $\kappa(x, x')$  is the kernel or covariance function, i.e.,

$$m(x) = E[f(x)] \quad (15.3)$$

$$\kappa(x, x') = E[(f(x) - m(x))(f(x') - m(x'))^T] \quad (15.4)$$

We obviously require that  $\kappa()$  be a positive definite kernel.

For any finite set of points, this process defines a joint Gaussian:

$$p(f|X) = N(f|\mu, K) \quad (15.5)$$

where  $K_{ij} = \kappa(x_i, x_j)$  and  $\mu = (m(x_1), \dots, m(x_N))$ . Note that it is common to use a mean function of  $m(x) = 0$ , since the GP

is flexible enough to model the mean arbitrarily well, as we will see below. However, in Section 15.2.6 we will consider parametric models for the mean function, so the GP just has to model the residual errors. This semi-parametric approach combines the interpretability of parametric models with the accuracy of non-parametric models.

### 15.2.1 Predictions using noise-free observations

Suppose we observe a training set  $D = \{(x_i, f_i), i = 1: N\}$ , where  $f_i = f(x_i)$  is the noise-free observation of the function evaluated at  $x_i$ . Given a test set  $X^*$  of size  $N^* \times D$ , we want to predict the function outputs  $f^*$ . If we ask the GP to predict  $f(x)$  for a value of  $x$  that it has already seen, we want the GP to return the answer  $f(x)$  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

$$f, f^* \sim N(\mu, K) \quad (15.6)$$

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

$$p(f^* | X^*, X, f) = N(f^* | \mu^*, \Sigma^*) \quad (15.7)$$

$$\mu^* = \mu(X^*) + K^* K^{-1} (f - \mu(X)) \quad (15.8)$$

$$\Sigma^* = K^{**} - K^* K^{-1} K^* \quad (15.9)$$

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

$$\kappa(x, x') = \sigma^2 f \exp(-\frac{1}{2} \frac{2}{l^2} (x - x')^2) \quad (15.10)$$

Here  $l$  controls the horizontal length scale over which the function varies, and  $\sigma^2 f$  controls the vertical variation. (We discuss how to estimate such kernel parameters below.) On the right we show samples from the posterior,  $p(f^* | X^*, X, f)$ . 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,  $y = f(x) + \epsilon$ , where  $\epsilon \sim N(0, \sigma^2)$ . 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

$$\text{cov}[y_p, y_q] = \kappa(x_p, x_q) + \sigma^2 \delta_{pq} \quad (15.11)$$

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

$$\text{cov}[y|X] = K + \sigma^2 I_N = Ky \quad (15.12)$$

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

$$y | f_* \sim N(0, Ky K^T * K^{**}) \quad (15.13)$$

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

$$p(f_* | X, y) = N(f_* | \mu_*, \Sigma_*) \quad (15.14)$$

$$\mu_* = K^T * K^{-1} y \quad (15.15)$$

$$\Sigma_* = K^{**} - K^T * K^{-1} y K_* \quad (15.16)$$

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

$$p(f_* | x_*, X, y) = N(f_* | k^T * K^{-1} y, k^{**} - k^T * K^{-1} y k_*) \quad (15.17)$$

where  $k_* = [\kappa(x_*, x_1), \dots, \kappa(x_*, x_N)]$  and  $k^{**} = \kappa(x_*, x_*)$ .

Another way to write the posterior mean is as follows:

$$f_* = k^T * K^{-1} y = \sum_{i=1}^n \alpha_i \kappa(x_i, x_*) \quad (15.18)$$

where  $\alpha = K^{-1} y$ . We will revisit this expression later.

### 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

$$\kappa(x_p, x_q) = \sigma^2 f \exp(-\frac{1}{2} (x_p - x_q)^2) + \sigma^2 \delta_{pq} \quad (15.19)$$

Here  $l$  is the horizontal scale over which the function changes,  $\sigma_f^2$  controls the vertical scale of the function, and  $\sigma_y^2$  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.

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

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

We can define the matrix  $M$  in several ways. The simplest is to use an isotropic matrix,  $M_1 = -2\ell I$ . See Figure 15.4(a) for an example. We can also endow each dimension with its own characteristic length scale,  $M_2 = \text{diag}(\ell_1, \dots, \ell_d) - 2I$ . 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  $M = M_2$  with  $\ell = (1, 3)$ , so the function changes faster along the  $x_1$  direction than the  $x_2$  direction. We can also create a matrix of the form  $M_3 = \Lambda \Lambda^T + \text{diag}(\ell) - 2I$ , where  $\Lambda$  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  $\Lambda$  correspond to relevant directions in input space. In Figure 15.4(c), we use  $\ell = (6, 6)$  and  $\Lambda = (1, -1)$ , 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

$$p(y|X) = p(y|f, X)p(f|X)df \quad (15.21)$$

Since  $p(f|X) = N(f|0, K)$ , and  $p(y|f) = \prod_i N(y_i|f_i, \sigma^2_y)$ , the marginal likelihood is given by

$$\log p(y|X) = \log N(y|0, K_y) = -\frac{1}{2} y^T K^{-1} y - \frac{1}{2} \log |K_y| - \frac{N}{2} \log(2\pi) \quad (15.22)$$

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  $\sigma^2_y$  fixed. Let  $J(l) = -\log p(y|X, l)$ . For short length scales, the fit will be good, so  $y^T K^{-1} y$  will be small. However, the model complexity will be high:  $K$  will be almost diagonal (as in Figure 14.3, top right), since most points will not be considered “near” any others, so the  $\log |K_y|$  will be large. For long length scales, the fit will be poor but the model complexity will be low:  $K$  will be almost all 1’s (as in Figure 14.3, bottom right), so  $\log |K_y|$  will be small.

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

$$\partial \partial \theta_j \log p(y|X) = \frac{1}{2} y^T K^{-1} y \partial K_j \partial \theta_j K^{-1} y - \frac{1}{2} \text{tr}(K^{-1} y \partial K_j \partial \theta_j) \quad (15.23)$$

$$= \frac{1}{2} \text{tr}(\alpha \alpha^T - K^{-1} y) \partial K_j \partial \theta_j \quad (15.24)$$

where  $\alpha = K^{-1} y$ . It takes  $O(N^3)$  time to compute  $K^{-1} y$ , and then  $O(N^2)$  time per hyperparameter to compute the gradient. The form of  $\partial K_j \partial \theta_j$  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  $\sigma^2_y \geq 0$ . In this case, we can define  $\theta = \log(\sigma^2_y)$ , 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  $\sigma^2_f = 1$ , and plot  $\log p(y|X, \theta, \sigma^2_y)$  (where  $X$

and  $y$  are the 7 data points shown in panels b and c) as we vary  $\ell$  and  $\sigma^2 y$ . The two 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  $\sigma^2 y \approx 1$  (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  $\approx 0.5$  (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.

#### 15.2.4.2 Bayesian inference for the hyper-parameters

An alternative to computing a point estimate of the hyper-parameters is to compute their posterior. Let  $\theta$  represent all the kernel parameters, as well as  $\sigma^2 y$ . If the dimensionality of  $\theta$  is small, we can compute a discrete grid of possible values, centered on the MAP estimate  $\hat{\theta}$  (computed as above). We can then approximate the posterior over the latent variables using

$$p(f|D) \propto \sum_{s=1} S p(f|D, \theta_s) p(\theta_s|D) \delta_s \quad (15.25)$$

where  $\delta_s$  denotes the weight for grid point  $s$ . In higher dimensions, a regular grid suffers from the curse of dimensionality. An obvious alternative is Monte Carlo, but this can be slow. Another approach is to use a form of quasiMonte Carlo, whereby we place grid points at the mode, and at a distance  $\pm 1\text{sd}$  from the mode along each dimension, for a total of  $2|\theta| + 1$  points. This is called a central composite design (Rue et al. 2009). (This is also used in the unscented Kalman filter, see Section 18.5.2.) To make this Gaussian-like approximation more reasonable, we often log-transform the hyper-parameters. See Figure 15.6 for an illustration.

### 15.2.4.3 Multiple kernel learning

A quite different approach to optimizing kernel parameters known as multiple kernel learning. The idea is to define the kernel as a weighted sum of base kernels,  $\kappa(x, x_i) = \sum w_j \kappa_j(x, x_i)$ , and then to optimize the weights  $w_j$  instead of the kernel parameters themselves. This is particularly useful if we have different kinds of data which we wish to fuse together. See e.g., (Rakotomamonjy et al. 2008) for an approach based on risk-minimization and convex optimization, and (Girolami and Rogers 2005) for an approach based on variational Bayes.

### 15.2.5 Computational and numerical issues

The predictive mean is given by  $f_* = k^T K^{-1} y$ . For reasons of numerical stability, it is unwise to directly invert  $Ky$ . A more robust alternative is to compute a Cholesky decomposition,  $Ky = LL^T$ . We can then compute the predictive mean and variance, and the log marginal likelihood, as shown in the pseudo-code in Algorithm 6 (based on (Rasmussen and Williams 2006, p19)). It takes  $O(N^3)$  time to compute the Cholesky decomposition, and  $O(N^2)$  time to solve for  $\alpha = K^{-1} y$   $y = L^T L^{-1} y$ . We can then compute the mean using  $k^T \alpha$  in  $O(N)$  time and the variance using  $k^T k - k^T L^T L^{-1} k$  in  $O(N^2)$  time for each test case. An alternative to Cholesky decomposition is to solve the linear system  $Ky = y$  using conjugate gradients (CG). If we terminate this algorithm after  $k$  iterations, it takes  $O(kN^2)$  time. If we run for  $k = N$ , it gives the exact solution in  $O(N^3)$  time. Another approach is to approximate the matrix-vector multiplies needed by CG using the fast Gauss transform. (Yang et al. 2005); however, this doesn't scale to high-dimensional inputs. See also Section 15.6 for a discussion of other speedup techniques.

### 15.2.6 Semi-parametric GPs

Sometimes it is useful to use a linear model for the mean of the process, as follows:

$$f(x) = \beta^T \varphi(x) + r(x) \quad (15.26)$$

where  $r(x) \sim GP(0, \kappa(x, x_i))$  models the residuals. This combines a parametric and a nonparametric model, and is known as a semi-parametric model. If we assume  $\beta \sim N(b, B)$ , we can integrate these parameters out to get a new GP (O'Hagan 1978):

$$f(x) \sim GP(\varphi(x)^T b, \kappa(x, x_i) + \varphi(x)^T B \varphi(x_i)) \quad (15.27)$$

Integrating out  $\beta$ , the corresponding predictive distribution for test inputs  $X_*$  has the following form (Rasmussen and Williams 2006, p28):

$$p(f_*|X_*, X, y) = N(f_*, \text{cov}[f_*]) \quad (15.28)$$

$$f_* = \Phi T * \beta + K T * K^{-1} y (y - \Phi \beta) \quad (15.29)$$

$$\beta = (\Phi T K^{-1} y \Phi + B - 1)^{-1} (\Phi K^{-1} y y + B - 1 b) \quad (15.30)$$

$$\text{cov}[f_*] = K_{**} - K T * K^{-1} y K * + R(T(B - 1 + \Phi K^{-1} y \Phi T) - 1) R \quad (15.31)$$

$$R = \Phi * - \Phi K^{-1} y \Phi * \quad (15.32)$$

The predictive mean is the output of the linear model plus a correction term due to the GP, and the predictive covariance is the usual GP covariance plus an extra term due to the uncertainty in  $\beta$ .

已思考 8s

下面先给出逐句完整翻译（包含公式编号），随后是通俗解释与要点总结。我会尽量把数学讲清楚，用直观比喻帮助理解。

## 翻译（不遗漏）

**图 15.2 左：**使用 SE 核（平方指数核）从 GP 先验中抽样得到的一些函数。右：在对 5 个无噪声观测进行条件化之后，从 GP 后验中抽样得到的一些函数。灰色带表示  $E[f(x)] \pm 2 \text{ std}(f(x))$ 。基于 (Rasmussen and Williams 2006) 图 2.2 改绘。

**图 15.3 一些一维 GP：**都使用 SE 核，但超参数不同，并对 20 个含噪观测进行拟合。核的形式见式 (15.19)。超参数  $(\ell, \sigma_f, \sigma_y)$  分别为：(a)  $(1, 1, 0.1)$ ; (b)  $(0.3, 1.08, 0.00005)$ ; (c)  $(3.0, 1.16, 0.89)$ 。基于 (Rasmussen and Williams 2006) 图 2.5 改绘。

**图 15.4 使用 SE 核但超参数不同，从 GP 中抽样得到的一些二维函数。**核的形式见式 (15.20)，其中 (a)  $M = I$ ; (b)  $M = \text{diag}(1, 3)^{-2}$ ; (c)  $M = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} + \text{diag}(6, 6)^{-2}$ 。基于 (Rasmussen and Williams 2006) 图 5.1 改绘。

**图 15.5 边际似然曲面中局部极小值的示意。**(a) 在固定  $\sigma_f^2 = 1$  的情况下，利用面板 (b)、(c) 所示 7 个数据点，作出 log 边际似然关于  $\sigma_y^2$  与  $\ell$  的图。(b) 对应左下局部极小值  $(\ell, \sigma_n^2) \approx (1, 0.2)$  的函数；它比较“抖”（变化快）且噪声低。(c) 对应右上局部极小值  $(\ell, \sigma_n^2) \approx (10, 0.8)$  的函数；它很平滑且噪声高。数据实际是用  $(\ell, \sigma_n^2) = (1, 0.1)$  生成的。

**图 15.6** 三种不同的“超参数后验”的近似方法：基于网格、蒙特卡洛、以及中心复合设计 (central composite design)。

## 15.2 回归中的高斯过程 (GPs for regression)

在本节中，我们讨论用于回归的 GP。令回归函数的先验为一个 GP，记为

$$f(x) \sim \text{GP}(m(x), \kappa(x, x')) \quad (15.2)$$

其中  $m(x)$  为均值函数， $\kappa(x, x')$  为核函数或协方差函数，即

$$m(x) = \mathbb{E}[f(x)] \quad (15.3)$$

$$\kappa(x, x') = \mathbb{E}((f(x) - m(x))(f(x') - m(x'))^T) \quad (15.4)$$

显然我们要求  $\kappa(\cdot)$  是正定核。对于任意有限点集，该过程定义了一个联合高斯分布：

$$p(\mathbf{f} | X) = \mathcal{N}(\mathbf{f} | \boldsymbol{\mu}, K) \quad (15.5)$$

其中  $K_{ij} = \kappa(x_i, x_j)$ ,  $\boldsymbol{\mu} = (m(x_1), \dots, m(x_N))$ 。需要说明的是，常见做法是取零均值  $m(x) = 0$ ，因为 GP 的灵活性足以很好地刻画均值（下面会看到）。不过在 15.2.6 节我们会考虑对均值函数使用参数化模型，此时 GP 只需拟合残差。这种半参数方法兼具参数模型的可解释性与非参数模型的精度。

### 15.2.1 使用无噪声观测的预测

假设我们观测到训练集  $D = \{(x_i, f_i), i = 1:N\}$ ，其中  $f_i = f(x_i)$  是在  $x_i$  处的无噪声函数值。给定大小为  $N_{\text{*}} \times D$  的测试集  $X_{\text{*}}$ ，我们希望预测函数输出  $\mathbf{f}_{\text{*}}$ 。如果让 GP 对已经见过的输入  $x$  进行预测，我们希望它在该点返回精确的  $f(x)$  且没有不确定性。换言之，它应当对训练数据进行插值。这只有在假设观测无噪声时才会发生。下面回到预测问题。由 GP 的定义，联合分布为

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

其中  $K = \kappa(X, X) \in \mathbb{R}^{N \times N}$ ,  $K_{\text{*}} = \kappa(X, X_{\text{*}}) \in \mathbb{R}^{N \times N_{\text{*}}}$ ,  $K_{\text{*}}^T = \kappa(X_{\text{*}}, X_{\text{*}}) \in \mathbb{R}^{N_{\text{*}} \times N_{\text{*}}}$ 。按高斯条件分布的标准公式（第 4.3 节），后验为

$$p(\mathbf{f}_{\text{*}} | X_{\text{*}}, X, \mathbf{f}) = \mathcal{N}(\mathbf{f}_{\text{*}} | \boldsymbol{\mu}_{\text{*}}, \Sigma_{\text{*}}) \quad (15.7)$$

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

$$\Sigma_{\text{*}} = K_{\text{*}}^T K^{-1} K_{\text{*}} \quad (15.9)$$

这一过程如图 15.2 所示。左图展示了先验  $p(\mathbf{f} \mid X)$  的一些样本，我们使用的是**平方指数核**（也叫高斯核或 RBF 核）。在一维时，该核为

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

其中  $\ell$  控制函数**横向**变化的长度尺度， $\sigma_f^2$  控制**纵向**变化幅度。（如何估计这些核参数将于下文讨论。）右图展示了后验  $p(\mathbf{f}_* \mid X_*, X, \mathbf{f})$  的样本。可以看到模型对训练数据**完全插值**，并且当远离观测区域时预测不确定性会增大。无噪声 GP 回归的一个应用是：把它作为复杂仿真器（如天气预报程序）行为的**廉价代理**。若仿真器是随机的，我们可以把  $f$  定义为其**平均输出**（注意此处仍无观测噪声）。随后就能通过考察仿真参数改变对 GP 预测的影响，来估计真实仿真器的变化，而无需多次运行昂贵的仿真。这类策略称为 **DACE** (design and analysis of computer experiments, 计算机试验的设计与分析) (Santner 等, 2003)。

### 15.2.2 使用含噪观测的预测

现在考虑我们观测到的是带噪声的函数版本，

$$y = f(x) + \varepsilon, \quad \varepsilon \sim N(0, \sigma_y^2).$$

这种情况下，模型不再要求插值数据，但必须与观测“**接近**”。带噪响应的协方差为

$$\text{cov}[y_p, y_q] = \kappa(x_p, x_q) + \sigma_y^2 \delta_{pq} \quad (15.11)$$

其中  $\delta_{pq} = I(p = q)$ 。也即

$$\text{cov}[\mathbf{y} \mid X] = K + \sigma_y^2 I_N = K_y \quad (15.12)$$

第二个矩阵是对角阵，因为我们假设对每个观测独立地加了噪声。观测数据与测试点处的潜在（无噪声）函数的联合分布为

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim N(\mathbf{0}, \begin{bmatrix} K_y & K_* \\ K_*^T & K_{**} \end{bmatrix}) \quad (15.13)$$

（为书写简洁，这里假设均值为零。）因此后验预测分布为

$$p(\mathbf{f}_* \mid X_*, X, \mathbf{y}) = N(\mathbf{f}_* \mid \boldsymbol{\mu}_*, \Sigma_*) \quad (15.14)$$

$$\boldsymbol{\mu}_* = K_*^T K_y^{-1} \mathbf{y} \quad (15.15)$$

$$\Sigma_* = K_{**} - K_*^T K_y^{-1} K_* \quad (15.16)$$

若只有一个测试输入，进一步化简为

$$p(f_{\setminus *} \mid x_{\setminus *}, X, \mathbf{y}) = N(f_{\setminus *} \mid \mathbf{k}_{\setminus *}^T K_y^{-1} \mathbf{y}, k_{\setminus * \setminus *} - \mathbf{k}_{\setminus *}^T K_y^{-1} \mathbf{k}_{\setminus *}) \quad (15.17)$$

其中  $\mathbf{k}_{\setminus *} = [\kappa(x_{\setminus *}, x_1), \dots, \kappa(x_{\setminus *}, x_N)]$ ,  $k_{\setminus * \setminus *} = \kappa(x_{\setminus *}, x_{\setminus *})$ 。后验均值也可写成

$$f_{\setminus *} = \mathbf{k}_{\setminus *}^T K_y^{-1} \mathbf{y} = \sum_{i=1}^N \alpha_i \kappa(x_i, x_{\setminus *}) \quad (15.18)$$

其中  $\alpha = K_y^{-1} \mathbf{y}$ 。我们稍后会回到这个表达式。

### 15.2.3 核参数的影响

GP 的预测性能完全取决于所选核是否合适。假设我们为含噪观测选用如下**平方指数 (SE) 核**

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

其中  $\ell$  是函数变化的**水平尺度**,  $\sigma_f^2$  控制函数的**垂直尺度**,  $\sigma_y^2$  为噪声方差。图 15.3 展示了改变这些参数的效果。我们先用  $(\ell, \sigma_f, \sigma_y) = (1, 1, 0.1)$  从 SE 核采样出 20 个带噪数据, 然后在给定数据的条件下, 用不同参数做预测。图 15.3(a) 仍用  $(1, 1, 0.1)$ , 拟合良好。图 15.3(b) 把长度尺度降到  $\ell = 0.3$  (其它参数用最大 (边际) 似然优化; 见下文), 此时函数更**起伏** (更“抖”)。同时不确定性上升得更快, 因为与训练点的“有效距离”增长更快。图 15.3(c) 把长度尺度增至  $\ell = 3$ , 函数变得更**平滑**。

我们可把 SE 核扩展到多维:

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

矩阵  $M$  可多种设定。最简单是**各向同性**矩阵  $M_1 = \ell^{-2} I$  (图 15.4(a))。也可以给每个维度设置自己的长度尺度,  $M_2 = \text{diag}(\ell)^{-2}$ 。若某些长度尺度很大, 则对应特征维度被视为**不相关**, 与 ARD (13.7 节) 类似。图 15.4(b) 使用  $M = M_2$  且  $\ell = (1, 3)$ , 因此  $x_1$  方向的变化比  $x_2$  方向更快。还可以构造

$M_3 = \Lambda \Lambda^T + \text{diag}(\ell)^{-2}$ , 其中  $\Lambda \in \mathbb{R}^{D \times K}$ ,  $K < D$ 。 (Rasmussen 和 Williams 2006, 第 107 页) 把它称为“**因子分析距离函数**”, 类比于因子分析 (12.1 节) 把协方差近似为低秩矩阵加对角矩阵。矩阵  $\Lambda$  的列对应输入空间中的“相关方向”。图 15.4(c) 取  $\ell = (6, 6)$ ,  $\Lambda = (1, -1)^T$ , 所以函数主要沿着与  $(1, 1)$  垂直的方向变化最快。

### 15.2.4 估计核参数

一种办法是对离散网格做穷举搜索, 以验证损失为目标, 但这很慢 (SVM 调核多用此法)。这里我们采用**经验贝叶斯**, 可用**连续优化**, 更快。具体是最大化**边际似然**

$$p(\mathbf{y} \mid X) = \int p(\mathbf{y} \mid \mathbf{f}, X) p(\mathbf{f} \mid X) d\mathbf{f} \quad (15.21)$$

由于  $p(\mathbf{f} | X) = \mathcal{N}(\mathbf{f} | 0, K)$ , 且  $p(\mathbf{y} | \mathbf{f}) = \prod_i \mathcal{N}(y_i | f_i, \sigma_y^2)$ , 因此边际似然为

$$\log p(\mathbf{y} | X) = \log \mathcal{N}(\mathbf{y} | 0, K_y) = -\frac{1}{2} \mathbf{y}^T K_y^{-1} \mathbf{y} - \frac{1}{2} \log |K_y| - \frac{N}{2} \log(2\pi) \quad (5.22)$$

第一项是**数据拟合项**, 第二项是**模型复杂度项**, 第三项是常数。为理解前两项的权衡, 考虑一维 SE 核, 改变长度尺度  $\ell$  而固定  $\sigma_y^2$ 。记  $J(\ell) = -\log p(\mathbf{y} | X, \ell)$ 。当  $\ell$  很短时, 拟合好 ( $\mathbf{y}^T K_y^{-1} \mathbf{y}$  小), 但复杂度高: 由于大多数点彼此“不近”,  $K$  几乎成对角 (见图 14.3 右上), 所以  $\log |K_y|$  很大。相反, 当  $\ell$  很长时, 拟合差但复杂度低:  $K$  近似全 1 (见图 14.3 右下),  $\log |K_y|$  很小。

下面讨论如何最大化边际似然。令核参数 (亦称超参数) 为  $\theta$ 。可得

$$\frac{\partial}{\partial \theta_j} \log p(\mathbf{y} | X) = \frac{1}{2} \mathbf{y}^T K_y^{-1} \frac{\partial K_y}{\partial \theta_j} K_y^{-1} \mathbf{y} - \frac{1}{2} \text{tr}(K_y^{-1} \frac{\partial K_y}{\partial \theta_j}) \quad (15.23)$$

$$= \frac{1}{2} \text{tr}((\boldsymbol{\alpha} \boldsymbol{\alpha}^T - K_y^{-1}) \frac{\partial K_y}{\partial \theta_j}), \quad \boldsymbol{\alpha} = K_y^{-1} \mathbf{y} \quad (15.24)$$

计算  $K_y^{-1}$  需  $O(N^3)$  时间; 之后每个超参数的梯度为  $O(N^2)$ 。 $\partial K_y / \partial \theta_j$  的形式取决于核及参数。超参数常有约束 (如  $\sigma_y^2 \geq 0$ ), 此时可设  $\theta = \log \sigma_y^2$ , 用链式法则求导。有了  $\log$  边际似然与其导数, 就能用任意标准的**基于梯度的优化器**估计核参数。但目标函数**非凸**, 会有**局部极小/极大**, 如下例所示。

#### 15.2.4.1 例子

见图 15.5。使用式 (15.19) 的 SE 核, 取  $\sigma_f^2 = 1$ , 对  $\ell, \sigma_y^2$  变化绘制  $\log p(\mathbf{y} | X, \ell, \sigma_y^2)$  ( $X, \mathbf{y}$  即 (b)(c) 中 7 个点)。两个局部最优用“+”标出。左下最优对应**低噪声、短长度尺度** (面板 b), 右上最优对应**高噪声、长长度尺度** (面板 c)。仅凭 7 个点, 证据不足以自信地选择哪一个更合理, 尽管更复杂的模型 (b) 的边际似然约比简单模型 (c) 高 60%。随着数据增多, MAP 估计会更稳定。图 15.5 还展示了其他典型现象: 当  $\sigma_y^2 \approx 1$  (面板 a 顶部) 噪声极高,  $\log$  边际似然对  $\ell$  不敏感 (等高线水平), 因为数据几乎都被解释成噪声; 当  $\ell \approx 0.5$  (面板 a 左侧) 长度尺度极短时,  $\log$  边际似然对噪声水平不敏感, 因为数据被**完美插值**。好的优化器都不会落在这些区域。

#### 15.2.4.2 超参数的贝叶斯推断

与其求超参数的点估计, 不如求它们的**后验**。令  $\theta$  表示全部核参数以及  $\sigma_y^2$ 。若  $\theta$  维度较小, 可在 MAP 估计  $\hat{\theta}$  (按上法得到) 附近构建**离散网格**。随后用

$$p(\mathbf{f} | D) \propto \sum_{s=1}^S p(\mathbf{f} | D, \theta_s) p(\theta_s | D) \delta_s \quad (15.25)$$

来近似潜变量的后验, 其中  $\delta_s$  是网格点  $s$  的权重。高维情况下, 规则网格遭遇**维度灾难**。显然可用**蒙特卡洛**, 但可能较慢。另一种是**准蒙特卡洛/中心复合设计 (CCD)**: 把网格点

放在众数上，并在每个维度方向上离众数  $\pm 1$  个标准差处各放一点，共  $2|\theta| + 1$  个点 (Rue 等, 2009)；这也用于无迹卡尔曼滤波 (18.5.2 节)。为使该“类高斯”近似更合理，常对超参数做对数变换。见图 15.6。

#### 15.2.4.3 多核学习 (Multiple kernel learning)

另一类优化核参数的方法是多核学习。思想是把核写成基核的加权和，

$$\kappa(x, x') = \sum_j w_j \kappa_j(x, x'),$$

然后优化权重  $w_j$  而不是每个核自身的参数。若我们要融合不同类型的数据，这尤其有用。参见基于风险最小化与凸优化的方法 (Rakotomamonjy 等, 2008)，以及基于变分贝叶斯的方法 (Girolami 和 Rogers, 2005)。

#### 15.2.5 计算与数值问题

预测均值为  $f_{\text{pred}} = \mathbf{k}_{\text{pred}}^T K_y^{-1} \mathbf{y}$ 。出于数值稳定性考虑，不要直接求逆  $K_y^{-1}$ 。更稳健的做法是做 Cholesky 分解： $K_y = LL^T$ 。随后可按算法 6 (基于 Rasmussen & Williams 2006, 第 19 页) 中的伪代码计算预测均值与方差以及 log 边际似然。计算  $L$  需  $O(N^3)$ ；解  $\alpha = K_y^{-1} \mathbf{y} = L^T L^{-1} \mathbf{y}$  需  $O(N^2)$ 。之后每个测试点，均值用  $\mathbf{k}_{\text{pred}}^T \alpha$  ( $O(N)$ )，方差用  $k_{\text{pred}}^T L^T L^{-1} \mathbf{k}_{\text{pred}}$  ( $O(N^2)$ )。另一种替代 Cholesky 的办法是用共轭梯度 (CG) 求解线性方程  $K_y \alpha = \mathbf{y}$ 。若在  $k$  次迭代时停止，耗时  $O(kN^2)$ ；若取  $k = N$  则得到精确解，耗时  $O(N^3)$ 。还可以用快速高斯变换近似 CG 所需的矩阵-向量乘 (Yang 等, 2005)，但它不适合高维输入。更多加速方法见 15.6 节。

#### 15.2.6 半参数 GP (Semi-parametric GPs)

有时把过程的均值设为线性模型会更有用：

$$f(x) = \boldsymbol{\beta}^T \phi(x) + r(x) \quad (15.26)$$

其中  $r(x) \sim \text{GP}(0, \kappa(x, x'))$  用来刻画残差。这是参数与非参数的结合，称为半参数模型。若假设  $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{b}, B)$ ，可将这些参数积分掉，得到一个新的 GP (O'Hagan 1978)：

$$f(x) \sim \text{GP}(\phi(x)^T \mathbf{b}, \kappa(x, x') + \phi(x)^T B \phi(x')) \quad (15.27)$$

积分掉  $\boldsymbol{\beta}$  后，测试输入  $X_{\text{pred}}$  的对应预测分布为 (Rasmussen & Williams 2006, 第 28 页)：

$$p(\mathbf{f}_{\text{pred}} | X_{\text{pred}}, X, \mathbf{y}) = \mathcal{N}(\mathbf{f}_{\text{pred}}, \text{cov}[\mathbf{f}_{\text{pred}}]) \quad (15.28)$$

$$\mathbf{f}_{\text{pred}} = \Phi_{\text{pred}}^T \boldsymbol{\beta} + K_{\text{pred}}^T K_y^{-1} (\mathbf{y} - \Phi \boldsymbol{\beta}) \quad (15.29)$$

$$\boldsymbol{\beta} = (\Phi^T K_y^{-1} \Phi + B^{-1})^{-1} (\Phi^T K_y^{-1} \mathbf{y} + B^{-1} \mathbf{b}) \quad (15.30)$$

$$\text{cov}[\mathbf{f}_{\setminus *}] = K_{\setminus * \setminus *} - K_{\setminus *}^T K_y^{-1} K_{\setminus *} + \mathbf{R}^T (B^{-1} + \Phi K_y^{-1} \Phi^T)^{-1} \mathbf{R} \quad (15.31)$$

$$\mathbf{R} = \Phi_{\setminus *} - \Phi K_y^{-1} \Phi^T \quad (15.32)$$

预测均值等于线性模型输出加上 GP 的修正项；预测协方差等于常规 GP 协方差再加上由于  $\beta$  不确定性引入的额外项。

---

## 通俗解释

### 1) 先验样本 vs 后验样本 (图 15.2)

- 先验 = “在没看数据前，我觉得函数长什么样”的分布。SE 核意味着函数是平滑的；从先验里随机抽几条曲线，就是左图。
- 条件化（看到 5 个无噪点）后，后验就“被数据拉住”：曲线必定穿过黑叉（插值），并且在数据附近灰带（不确定性）很窄，远离数据就变宽。

### 2) 含噪 vs 无噪

- 无噪：模型必须精确穿过训练点。
- 含噪：不必穿过，只需“靠近”。数学上就是把训练协方差  $K$  加上噪声方差  $\sigma_y^2 I$ 。

### 3) 核与超参数的物理意义 (图 15.3)

- $\ell$ （长度尺度）小：函数抖、变化快；远离训练点时“相似度掉得快”，所以不确定性涨得快。
- $\ell$  大：函数更平滑。
- $\sigma_f$ ：函数幅度（上下摆动范围）。
- $\sigma_y$ ：观测噪声（灰带厚度的一部分）。

### 4) 多维核矩阵 $M$ (图 15.4)

- $M = \ell^{-2} I$ ：各方向一样的“平滑度”。
- $M = \text{diag}(\ell_1, \ell_2)^{-2}$ ：每个方向自己的长度尺度 (ARD)：某方向  $\ell$  很大  $\rightarrow$  该维度几乎无关。
- $M = \Lambda \Lambda^T + \text{diag}(\ell)^{-2}$ ：先选若干重要方向 ( $\Lambda$  的列)，再加各向异性尺度，能表示“沿某条斜线方向变化快”。

### 5) 为什么用“边际似然”找超参数 (15.2.4) ?

- 直观：它综合了“拟合好不好”（第一项）与“模型多复杂”（第二项）。
- $\ell$  很小：拟合极好但复杂度大（容易过拟合）。
- $\ell$  很大：复杂度低但拟合差（欠拟合）。
- 求导公式 (15.23)(15.24) 允许用梯度法调超参。注意目标非凸，会有局部极值（图 15.5 两个“+”）。

## 6) 图 15.5 的启示

- 数据少时，短 $\ell$ +低噪与长 $\ell$ +高噪都能解释数据；二者对应两种**不同解释**：要么“真函数很抖但噪声小”，要么“真函数很平滑但噪声大”。
- 在“噪声极大”或“长度极短”这两端，边际似然对另一参数不敏感，优化容易**迷路**，所以初始化与正则很重要。

## 7) 超参数的贝叶斯处理（图 15.6）

- 不只给出一个最优点，而是给超参数一个**后验分布**。
- 低维：小网格就够；高维：规则网格爆炸  $\rightarrow$  用 MC 或**中心复合设计 (CCD)**：在众数附近、每个维度的  $\pm 1$  标准差放点，像“无迹变换”。
- 常对超参数取**对数**再近似，使后验更接近高斯。

## 8) 数值计算 (15.2.5)

- 千万别直接算  $K_y^{-1}$ ！用 Cholesky： $K_y = LL^T$ 。
- 复杂度：分解  $O(N^3)$ ；求  $\alpha = L^{-T}L^{-1}\mathbf{y}$  是  $O(N^2)$ ；每个新点的均值  $O(N)$ 、方差  $O(N^2)$ 。
- 可用**共轭梯度**近似求解，迭代  $k$  次  $\rightarrow O(kN^2)$ 。

## 9) 半参数 GP (15.2.6)

- 把“已知结构”（如线性或多项式部分）放入**均值里**： $f(x) = \beta^T \phi(x) + r(x)$ 。
- $\beta$  给高斯先验并**积分掉**，得到“新核 = 原核 + 线性部分诱导的协方差”。
- 预测 = “线性模型的输出 + GP 的修正”，方差 = “普通 GP 方差 +  $\beta$  的不确定性项”。
- 好处：既可**解释** ( $\beta$  代表可读参数)，又**灵活**（剩下由 GP 捕获）。

## 要点总结

- **GP 回归**：任意点集上的函数值联合高斯；核  $\kappa$  决定相似度与平滑性。
- **无噪 vs 含噪**：无噪  $\rightarrow$  完全插值；含噪  $\rightarrow$  用  $K_y = K + \sigma_y^2 I$ ，不必穿点。
- **预测公式**：均值  $K_{\backslash * \backslash *}^T K_y^{-1} \mathbf{y}$ ；方差  $K_{\backslash * \backslash *}^T K_y^{-1} K_{\backslash * \backslash *} - K_{\backslash * \backslash *}^T K_y^{-1} K_{\backslash * \backslash *}^*$ 。单点版本见式 (15.17)。
- **核超参数**： $\ell$  控制平滑度， $\sigma_f$  控制幅度， $\sigma_y$  为噪声。多维用矩阵  $M$  表示各向异性/重要方向 (ARD、因子形式)。
- **选超参**：最大化**边际似然**（拟合 vs 复杂度权衡），梯度见 (15.23)(15.24)；注意**非凸与局部解**（图 15.5）。
- **贝叶斯超参**：用网格、MC 或 CCD 近似超参数后验（图 15.6），常对超参数取对数。
- **数值实现**：用 Cholesky，避免显式求逆；复杂度主导为  $O(N^3)$ 。CG 可作近似。

- **半参数 GP**: 均值用线性模型，核中自动加入该结构的协方差；预测 = 线性 + GP 修正，方差也多一项。

如果你想，我可以把这些公式换成简短伪代码/步骤清单，或用你手头的数据做一个小型 GP 回归演示（比如可视化不同  $\ell$  对预测带的影响）。

这一小节主要在讲什么？我不太懂

已思考 14s

明白！这段内容其实就是在讲：**怎么用高斯过程 (GP) 来做回归**，以及在实操中要注意什么。给你按“故事线”捋一遍——

---

## 这小节在讲啥（一句话）

把“函数”当作随机对象来建模：先用核函数规定“相似输入→相似输出”的程度（先验），再结合数据得到后验，用它来**预测数值和不确定性**；同时讲清**无噪/含噪两种情形、核的超参数**怎么影响结果、**怎么学这些超参数、如何高效计算**，以及一个**半参数扩展**。

---

## 核心思路（通俗版）

### 1. 先验（不看数据时对函数的想法）

设定一个均值函数（通常取 0）和一个核函数  $\kappa(x, x')$ 。核越大=两个输入越相似→我们相信这两个位置的函数值越接近。

### 2. 看到数据后的后验

- **无噪情形**：模型必须**穿过训练点**（插值），远离数据处不确定性变大。（图 15.2 右侧的灰色带）
- **含噪情形**：允许不完全穿点，只要接近即可；协方差里多了噪声项  $\sigma_y^2 I$ 。

### 3. 核与超参数的作用（图 15.3、15.4）

- $\ell$ : **长度尺度**。小 → 曲线更“抖”、不确定性涨得快；大 → 更平滑。
- $\sigma_f$ : **幅度**（上下波动有多大）。
- $\sigma_y$ : **观测噪声**。
- 多维里用矩阵  $M$  表示各向同性/各向异性/沿某些方向更快变化（ARD、因子形式）。

#### 4. 怎么学到“好”的超参数?

用**边际似然** (既看拟合好不好, 又惩罚模型复杂) 做最大化。它**非凸**, 会有**局部极值**:

- 一种解释: 短  $\ell$ +**低噪** → 真函数很“抖”。
- 另一种: 长  $\ell$ +**高噪** → 真函数很平滑但噪大。

数据少时两者都能解释 (图 15.5 的两个“+”)。

若不想只给一个点解, 也可以对超参数做**贝叶斯后验近似**: 小网格、MC 或**中心复合设计** (图 15.6)。

#### 5. 计算实现要点

别直接求逆  $K_y^{-1}$ ! 用 **Cholesky 分解**更稳; 复杂度主项  $O(N^3)$ 。也可用**共轭梯度近似**加速。

#### 6. 半参数 GP

把“线性/多项式趋势”写进均值  $f(x) = \beta^\top \phi(x) + \text{GP残差}$ 。把  $\beta$  积分掉后, 预测=线性部分 + GP 修正, 方差也多一个“ $\beta$  不确定性”项。既可解释又灵活。

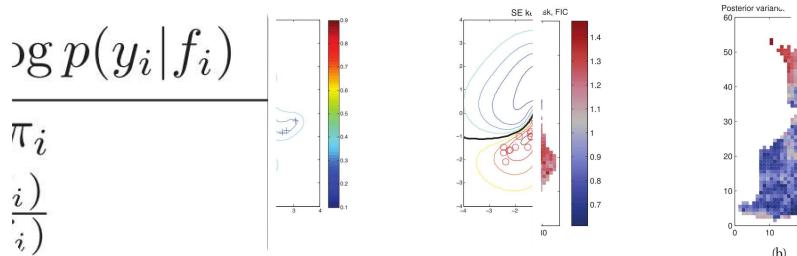
### 图里在说什么 (快速对照)

- 图 15.2: 左=先验样本 (没看数据的函数形状); 右=后验样本 (被 5 个无噪点“拉住”), 灰带=均值  $\pm 2$  标准差。
- 图 15.3: 同一数据, 不同  $(\ell, \sigma_f, \sigma_y)$  下的拟合对比:  $\ell$  小更“抖”,  $\ell$  大更平滑。
- 图 15.4: 二维里不同  $M$  的效果: 各向同性、各向异性、沿特定方向变化更快。
- 图 15.5: 边际似然的**两个局部解**: 短  $\ell$ +低噪 vs 长  $\ell$ +高噪; 数据少时都说得通。
- 图 15.6: 超参数后验的三种近似: 网格、MC、中心复合设计。

### 一页纸要点

- GP 回归 = “核 + 高斯”框架: 给出**预测均值和不确定性**。
- **无噪**: 精确插值; **含噪**: 不必穿点, 协方差多  $\sigma_y^2 I$ 。
- 预测公式本质: 训练点到测试点的**相似度加权和**。
- 超参数  $\ell, \sigma_f, \sigma_y$  决定**平滑度/幅度/噪声**; 用**边际似然**学它们, 但要防**局部极值**。
- 计算上用 Cholesky (稳) 或 CG (近似快)。
- 半参数版 = 线性趋势 + GP 残差: **可解释又拟合强**。

如果你愿意, 我可以用一个小例子 (比如 1D 人工数据) 画出不同  $\ell$  的预测曲线和灰色不确定带, 让直觉更强。



翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Table 15.1 Likelihood, gradient and Hessian for binary logistic/ probit GP regression. We assume  $y_i \in \{-1, +1\}$  and define  $\pi_i = (y_i + 1)/2 \in \{0, 1\}$  and  $\pi_i = \text{sigm}(f_i)$  for logistic regression, and  $\pi_i = \Phi(f_i)$  for probit regression. Also,  $\varphi$  and  $\Phi$  are the pdf and cdf of  $N(0, 1)$ .

Figure 15.7 Contours of the posterior predictive probability for the red circle class generated by a GP with an SE kernel. Thick black line is the decision boundary if we threshold at a probability of 0.5. (a) Manual parameters, short length scale. (b) Learned parameters, long length scale.

Figure 15.8 We show the relative risk of heart disease in Finland using a Poisson GP. Left: posterior mean. Right: posterior variance.

### 15.3 GPs meet GLMs

In this section, we extend GPs to the GLM setting, focussing on the classification case. As with Bayesian logistic regression, the main difficulty is that the Gaussian prior is not conjugate to the bernoulli/ multinoulli likelihood. There are several approximations one can adopt: Gaussian approximation (Section 8.4.3), expectation propagation (Kuss and Rasmussen 2005; Nickisch and Rasmussen 2008), variational (Girolami and Rogers 2006; Opper and Archambeau 2009), MCMC (Neal 1997; Christensen et al. 2006), etc. Here we focus on the Gaussian approximation, since it is the simplest and fastest.

#### 15.3.1 Binary classification

In the binary case, we define the model as  $p(y_i|x_i) = \sigma(y_i f(x_i))$ , where, following (Rasmussen and Williams 2006),

we assume  $y_i \in \{-1, +1\}$ , and we let  $\sigma(z) = \text{sigm}(z)$  (logistic regression) or  $\sigma(z) = \Phi(z)$  (probit regression). As for GP regression, we assume  $f \sim GP(0, \kappa)$ .

### 15.3.1.1 Computing the posterior

Define the log of the unnormalized posterior as follows:

$$(f) = \log p(y|f) + \log p(f|X) = \log p(y|f) - \frac{1}{2} f^T K^{-1} f - \frac{1}{2} \log |K| - N/2 \log 2\pi \quad (15.33)$$

Let  $J(f) = -(f)$  be the function we want to minimize.

The gradient and Hessian of this are given by

$$g = -\nabla \log p(y|f) + K^{-1} f \quad (15.34)$$

$$H = -\nabla \nabla \log p(y|f) + K^{-1} = W + K^{-1} \quad (15.35)$$

Note that  $W = -\nabla \nabla \log p(y|f)$  is a diagonal matrix because the data are iid (conditional on  $f$ ). Expressions for the gradient and Hessian of the log likelihood for the logit and probit case are given in Sections 8.3.1 and 9.4.1, and summarized in Table 15.1. We can use IRLS to find the MAP estimate. The update has the form

$$\begin{aligned} f_{\text{new}} &= f - H^{-1} g = f + (K^{-1} + W)^{-1} (\nabla \log p(y|f) - K^{-1} f) \\ &\quad (15.36) \end{aligned}$$

$$= (K^{-1} + W)^{-1} (Wf + \nabla \log p(y|f)) \quad (15.37)$$

At convergence, the Gaussian approximation of the posterior takes the following form:

$$p(f|X, y) \approx N(\hat{f}, (K^{-1} + W)^{-1}) \quad (15.38)$$

### 15.3.1.2 Computing the posterior predictive

We now compute the posterior predictive. First we predict the latent function at the test case  $x^*$ . For the mean we have

$$E[f^*|x^*, X, y] = E[f^*|f, x^*, X, y] p(f|X, y) df \quad (15.39)$$

$$= kT * K^{-1} f p(f|X, y) df \quad (15.40)$$

$$= kT * K^{-1} E[f|X, y] \approx kT * K^{-1} \hat{f} \quad (15.41)$$

where we used Equation 15.8 to get the mean of  $f^*$  given noise-free  $f$ . To compute the predictive variance, we use the rule of iterated variance:

$$\text{var}[f^*] = E[\text{var}[f^*|f]] + \text{var}[E[f^*|f]] \quad (15.42)$$

where all probabilities are conditioned on  $x^*, X, y$ . From Equation 15.9 we have

$$\begin{aligned} E[\text{var}[f^*|f]] &= E[k^{**} - kT * K^{-1} k^*] = k^{**} - kT * K^{-1} k^* \\ &\quad (15.43) \end{aligned}$$

From Equation 15.9 we have

$$\text{var}[E[f^*|f]] = \text{var}[k^* K^{-1} f] = kT * K^{-1} \text{cov}[f] K^{-1} k^* \quad (15.44)$$

Combining these we get

$$\text{var}[f^*] = k^{**} - kT * (K^{-1} - K^{-1} \text{cov}[f] K^{-1}) k^* \quad (15.45)$$

From Equation 15.38 we have  $\text{cov}[f] \approx (K^{-1} + W)^{-1}$ . Using

the matrix inversion lemma we get

$$\text{var}[f^*] \approx k^{**} - kT * K^{-1}k* + kT * K^{-1}(K-1 + W) \\ - 1K^{-1}k* \quad (15.46)$$

$$= k^{**} - kT * (K+W-1) - 1k* \quad (15.47)$$

So in summary we have

$$p(f^*|x^*, X, y) = N(E[f^*], \text{var}[f^*]) \quad (15.48)$$

To convert this in to a predictive distribution for binary responses, we use

$$\pi^* = p(y^* = 1|x^*, X, y) \approx \sigma(f^*) p(f^*|x^*, X, y) df^* \quad (15.49)$$

This can be approximated using any of the methods discussed in Section 8.4.4, where we discussed Bayesian logistic regression. For example, using the probit approximation of Section 8.4.4.2, we have  $\pi^* \approx \text{sigm}(\kappa(v)E[f^*])$ , where  $v = \text{var}[f^*]$  and  $\kappa(v) = (1 + \pi v/8) - 1$ .

### 15.3.1.3 Computing the marginal likelihood

We need the marginal likelihood in order to optimize the kernel parameters. Using the Laplace approximation in Equation 8.54 we have

$$\log p(y|X) \approx (\hat{f}) - 1/2 \log |H| + \text{const} \quad (15.50)$$

Hence

$$\log p(y|X) \approx \log p(y|\hat{f}) - 1/2 \hat{f}^T K^{-1} \hat{f} - 1/2 \log |K| - 1/2 \log |K-1+W| \quad (15.51)$$

Computing the derivatives  $\partial \log p(y|X, \theta) / \partial \theta_j$  is more complex than in the regression case, since  $\hat{f}$  and  $W$ , as well as  $K$ , depend on  $\theta$ . Details can be found in (Rasmussen and Williams 2006, p125).

### 15.3.1.4 Numerically stable computation

To implement the above equations in a numerically stable way, it is best to avoid inverting  $K$  or  $W$ . (Rasmussen and Williams 2006, p45) suggest defining

$$B = I + W^{-1/2} K W^{-1/2} \quad (15.52)$$

which has eigenvalues bounded below by 1 (because of the  $I$ ) and above by  $1 + N^{-1} \max_{ij} K_{ij}$  (because  $w_{ii} = \pi_i(1 - \pi_i) \leq 0.25$ ), and hence can be safely inverted. One can use the matrix inversion lemma to show

$$(K-1+W)^{-1} = K^{-1} - K^{-1}W^{-1}K \quad (15.53)$$

Hence the IRLS update becomes

$$f_{\text{new}} = (K-1+W)^{-1} (Wf + \nabla \log p(y|f)) \quad (15.54)$$

$$= K(I - W^{-1/2} B^{-1} W^{-1/2}) b \quad (15.55)$$

$$= K(b - W^{-1/2} L^{-1} (L^{-1} (W^{-1/2} Kb))) \quad (15.56)$$

where  $B = LL^T$  is a Cholesky decomposition of  $B$ . The fitting algorithm takes in  $O(TN^3)$  time and  $O(N^2)$  space, where  $T$  is the number of Newton iterations. At

convergence we have  $a = K^{-1}f$ , so we can evaluate the log marginal likelihood (Equation 15.51) using

$$\log p(y|X) = \log p(y|f) - \frac{1}{2} a^T f - \frac{1}{2} \log |K| (15.57)$$

where we exploited the fact that

$$|B| = |K||K^{-1} + W| = |I_N + W| (15.58)$$

We now compute the predictive distribution. Rather than using  $E[f^*] = k^T * K^{-1}f$ , we exploit the fact that at the mode,  $\nabla f = 0$ , so  $f^* = K(\nabla \log p(y|f))$ . Hence we can rewrite the predictive mean as follows:

$$E[f^*] = k^T * \nabla \log p(y|f) (15.59)$$

To compute the predictive variance, we exploit the fact that

$$(K+W-1)^{-1} = W^{-1} (K+W-1)^{-1} W^{-1} = W^{-1} (15.60)$$

to get

$$\text{var}[f^*] = k^*k^T - k^T * W^{-1} (L^T L) - 1 W^{-1} k^* = k^*k^T - v^T v (15.61)$$

where  $v = L \backslash (W^{-1} k^*)$ . We can then compute  $\pi^*$ . The whole algorithm is summarized in Algorithm 16, based on (Rasmussen and Williams 2006, p46). Fitting takes  $O(N^3)$  time, and prediction takes  $O(N^2 N^*)$  time, where  $N^*$  is the number of test cases.

### 15.3.1.5 Example

In Figure 15.7, we show a synthetic binary classification problem in 2d. We use an SE kernel. On the left, we show predictions using hyper-parameters set by hand; we use a short length scale, hence the very sharp turns in the decision boundary. On the right, we show the predictions using the learned hyper-parameters; the model favors a more parsimonious explanation of the data.

### 15.3.2 Multi-class classification

In this section, we consider a model of the form

$p(y_i|x_i) = \text{Cat}(y_i|S(f_i))$ , where  $f_i = (f_{i1}, \dots, f_{iC})$ , and we assume  $f_{ic} \sim \text{GP}(0, \kappa_c)$ . Thus we have one latent function per class, which are a priori independent, and which may use different kernels. As before, we will use a Gaussian approximation to the posterior. (A similar model, but using the multinomial probit function instead of the multinomial logit, is described in (Girolami and Rogers 2006).)

#### 15.3.2.1 Computing the posterior

The unnormalized log posterior is given by

$$l(f) = -\frac{1}{2} f^T K^{-1} f + y^T f - \frac{1}{2} \sum_{i=1}^N \log C_c \exp f_{ic} (15.62)$$

$$\log |K| - CN \log 2\pi (15.62)$$

where

$$f = (f_{11}, \dots, f_{N1}, f_{12}, \dots, f_{N2}, \dots, f_{1C}, \dots, f_{NC})^T (15.63)$$

and  $y$  is a dummy encoding of the  $y_i$ 's which has the same layout as  $f$ . Also,  $K$  is a block diagonal matrix containing  $K_c$ , where  $K_c = [k_c(x_i, x_j)]$  models the correlation of the  $c$ 'th latent function. The gradient and Hessian are given by

$$\nabla l = -K^{-1}f + y - \pi (15.64)$$

$$\nabla \nabla l = -K^{-1} - W (15.65)$$

where  $W = \text{diag}(\pi) - \Pi \Pi^T$ , where  $\Pi$  is a  $CN \times N$  matrix obtained by stacking  $\text{diag}(\pi:c)$  vertically. (Compare these expressions to standard logistic regression in Section 8.3.7.) We can use IRLS to compute the mode. The Newton step has the form

$$f_{\text{new}} = (K^{-1} + W)^{-1}(Wf + y - \pi) (15.66)$$

Naively implementing this would take  $O(C^3N^3)$  time.

However, we can reduce this to  $O(CN^3)$ , as shown in (Rasmussen and Williams 2006, p52).

### 15.3.2.2 Computing the posterior predictive

We can compute the posterior predictive in a manner analogous to Section 15.3.1.2. For the mean of the latent response we have

$$E[f_*] = k_c(x_*)^T K^{-1} c = k_c(x_*)^T (\bar{y}_c - \hat{\pi}_c) (15.67)$$

We can put this in vector form by writing

$$E[f_*] = Q^* \bar{y} - \hat{\pi} (15.68)$$

where

$$Q^* = (|k_1(x_*) \dots 0 \dots 0 \dots k_C(x_*)|) (15.69)$$

Using a similar argument to Equation 15.47, we can show that the covariance of the latent response is given by

$$\text{cov}[f_*] = \Sigma + Q^* T^* K^{-1} (K^{-1} + W)^{-1} K^{-1} Q^* (15.70)$$

$$= \text{diag}(k(x_*, x_*)) - Q^* T^* (K + W - 1) - 1 Q^* (15.71)$$

where  $\Sigma$  is a  $C \times C$  diagonal matrix with  $\Sigma_{cc} = k_c(x_*, x_*) - k_c(x_*) K^{-1} c k_c(x_*)$ , and  $k(x_*, x_*) = [k_c(x_*, x_*)]$ . To compute the posterior predictive for the visible response, we need to use

$$p(y|x_*, X, y) \approx \text{Cat}(y|S(f_*)) N(f_* | E[f_*], \text{cov}[f_*]) df_* (15.72)$$

We can use any of deterministic approximations to the softmax function discussed in Section 21.8.1.1 to compute this. Alternatively, we can just use Monte Carlo.

### 15.3.2.3 Computing the marginal likelihood

Using arguments similar to the binary case, we can show that

$$\log p(y|X) \approx -\frac{1}{2} f^T K^{-1} f + y^T f - N \sum_{c=1}^C \log C$$

$\hat{f}_{ic} = 1/2 \log |ICN + W|^{1/2} KW^{1/2} (15.73)|$

This can be optimized numerically in the usual way.

#### 15.3.2.4 Numerical and computational issues

One can implement model fitting in  $O(TCN^3)$  time and  $O(CN^2)$  space, where  $T$  is the number of Newton iterations, using the techniques described in (Rasmussen and Williams 2006, p50). Prediction takes  $O(CN^3 + CN^2N^*)$  time, where  $N^*$  is the number of test cases.

#### 15.3.3 GPs for Poisson regression

In this section, we illustrate GPs for Poisson regression. An interesting application of this is to spatial disease mapping. For example, (Vanhatalo et al. 2010) discuss the problem of modeling the relative risk of heart attack in different regions in Finland. The data consists of the heart attacks in Finland from 1996-2000 aggregated into 20km x 20km lattice cells. The model has the following form:

$$y_i \sim Poi(e_i r_i) \quad (15.74)$$

where  $e_i$  is the known expected number of deaths (related to the population of cell  $i$  and the overall death rate), and  $r_i$  is the relative risk of cell  $i$  which we want to infer. Since the data counts are small, we regularize the problem by sharing information with spatial neighbors. Hence we assume  $f = \log(r) \sim GP(0, \kappa)$ , where we use a Matern kernel with  $v = 3/2$ , and a length scale and magnitude that are estimated from data. Figure 15.8 gives an example of the kind of output one can obtain from this method, based on data from 911 locations. On the left we plot the posterior mean relative risk (RR), and on the right, the posterior variance. We see that the RR is higher in Eastern Finland, which is consistent with other studies. We also see that the variance in the North is higher, since there are fewer people living there.

已思考 32s

下面先把**正文与图表标题逐句翻译**（含表格中的导数/二阶导数），再用**通俗解释**讲清要点，最后给**总结清单**。

## 翻译 (不遗漏)

**表 15.1** 二元 logistic / probit GP 回归的似然、梯度与 Hessian。我们假设  $y_i \in \{-1, +1\}$ ，并定义

$t_i = (y_i + 1)/2 \in \{0, 1\}$ , 且在 logistic 回归中  $\pi_i = \text{sigm}(f_i)$ , 在 probit 回归中  $\pi_i = \Phi(f_i)$ 。另外,  $\phi$  与  $\Phi$  分别是  $N(0, 1)$  的概率密度函数与累积分布函数。

$$\log p(y_i | f_i) \quad \frac{\partial}{\partial f_i} \log p(y_i | f_i) \quad \frac{\partial^2}{\partial f_i^2} \log p(y_i | f_i)$$


---

logistic:  $t_i - \pi_i$   $-\pi_i(1 - \pi_i)$   
 $\log \text{sigm}(y_i f_i)$

---

probit:  $\log \Phi(y_i f_i)$   $\frac{y_i \phi(f_i)}{\Phi(y_i f_i)}$   $-\frac{\phi(f_i)^2}{\Phi(y_i f_i)^2} - \frac{y_i f_i \phi(f_i)}{\Phi(y_i f_i)}$

图 15.7 用 SE 核的 GP 为“红色圆点”这一类生成的后验预测概率等高线。若把阈值设为 0.5, 粗黑线是决策边界。(a) 人工设定的超参数, 短长度尺度。(b) 学到的超参数, 长长度尺度。

图 15.8 使用 Poisson GP 给出芬兰心脏病的相对风险图。左: 后验均值。右: 后验方差。

## 15.3 GPs 与广义线性模型 (GLMs)

本节把 GP 扩展到 GLM 场景, 重点是分类。和贝叶斯 logistic 回归一样, 主要困难在于: 高斯先验与 Bernoulli / Multinoulli 似然不共轭。可用多种近似: 高斯近似 (8.4.3 节)、期望传播 (Kuss & Rasmussen 2005; Nickisch & Rasmussen 2008)、变分法 (Girolami & Rogers 2006; Opper & Archambeau 2009)、MCMC (Neal 1997; Christensen 等 2006) 等。此处我们聚焦高斯近似, 因其最简单、最快。

### 15.3.1 二分类

二分类里, 模型为  $p(y_i | x_i) = \sigma(y_i f(x_i))$ 。按 (Rasmussen & Williams 2006), 令  $y_i \in \{-1, +1\}$ , 并取  $\sigma(z) = \text{sigm}(z)$  (logistic) 或  $\sigma(z) = \Phi(z)$  (probit)。像 GP 回归一样, 设  $f \sim \text{GP}(0, \kappa)$ 。

#### 15.3.1.1 计算后验

未归一化的对数后验:

$$\ell(f) = \log p(y | f) + \log p(f | X) = \log p(y | f) - \frac{1}{2} f^\top K^{-1} f - \frac{1}{2} \log |K| - \frac{N}{2} \ln(2\pi)$$

令  $J(f) = -\ell(f)$  为需最小化的函数。其梯度与 Hessian:

$$g = -\nabla \log p(y | f) + K^{-1} f \quad (15.34)$$

$$H = -\nabla \nabla \log p(y | f) + K^{-1} = W + K^{-1} \quad (15.35)$$

这里  $W = -\nabla \nabla \log p(y | f)$  为对角矩阵 (因为在给定  $f$  时各样本独立)。logit/probit 情形的梯度与 Hessian 见 8.3.1 与 9.4.1 节, 亦汇总于表 15.1。可用 IRLS (迭代加权最小二

乘) 求 MAP:

$$f_{\text{new}} = f - H^{-1}g = f + (K^{-1} + W)^{-1}(\nabla \log p(y | f) - K^{-1}f) \quad (15.36)$$

$$= (K^{-1} + W)^{-1}(Wf + \nabla \log p(y | f)) \quad (15.37)$$

收敛后, 后验的高斯近似为

$$p(f | X, y) \approx N(\hat{f}, (K^{-1} + W)^{-1}). \quad (15.38)$$

### 15.3.1.2 计算后验预测

先预测测试点  $x_{\backslash *}$  处的潜变量  $f_{\backslash *}$  的均值:

$$E[f_{\backslash *} | x_{\backslash *}, X, y] = \int E[f_{\backslash *} | f, x_{\backslash *}, X, y] p(f | X, y) df \quad (15.39)$$

$$= \int k_{\backslash *}^T K^{-1} f p(f | X, y) df = k_{\backslash *}^T K^{-1} E[f | X, y] \approx k_{\backslash *}^T K^{-1} \hat{f} \quad (15.40-15.41)$$

方差用“方差迭代公式”:

$$\text{var}[f_{\backslash *}] = E[\text{var}[f_{\backslash *} | f]] + \text{var}[E[f_{\backslash *} | f]] \quad (15.42)$$

由 (15.9) 得

$$E[\text{var}[f_{\backslash *} | f]] = k_{\backslash * \backslash *} - k_{\backslash *}^T K^{-1} k_{\backslash *} \quad (15.43)$$

$$\text{var}[E[f_{\backslash *} | f]] = k_{\backslash *}^T K^{-1} \text{cov}[f] K^{-1} k_{\backslash *} \quad (15.44)$$

合并得

$$\text{var}[f_{\backslash *}] = k_{\backslash * \backslash *} - k_{\backslash *}^T (K^{-1} - K^{-1} \text{cov}[f] K^{-1}) k_{\backslash *}. \quad (15.45)$$

由 (15.38)  $\text{cov}[f] \approx (K^{-1} + W)^{-1}$ 。用矩阵求逆引理化简:

$$\text{var}[f_{\backslash *}] \approx k_{\backslash * \backslash *} - k_{\backslash *}^T K^{-1} k_{\backslash *} + k_{\backslash *}^T K^{-1} (K^{-1} + W)^{-1} K^{-1} k_{\backslash *} \quad (15.46)$$

$$= k_{\backslash * \backslash *} - k_{\backslash *}^T (K + W^{-1})^{-1} k_{\backslash *}. \quad (15.47)$$

因此

$$p(f_{\backslash *} | x_{\backslash *}, X, y) = N(E[f_{\backslash *}], \text{var}[f_{\backslash *}]). \quad (15.48)$$

把它转成类别 1 的预测概率:

$$\pi_{\backslash *} = p(y_{\backslash *} = 1 | x_{\backslash *}, X, y) \approx \int \sigma(f_{\backslash *}) p(f_{\backslash *} | x_{\backslash *}, X, y) df_{\backslash *}. \quad (15.49)$$

可用 8.4.4 节讨论的近似方法。比如 **probit 近似** (8.4.4.2) :  $\pi_{\text{approx}} \approx \text{sigm}(\kappa(v) E[f_{\text{approx}}])$ , 其中  $v = \text{var}[f_{\text{approx}}]$ ,  $\kappa^2(v) = (1 + \pi v / 8)^{-1}$ 。

### 15.3.1.3 计算边际似然

为优化核参数需边际似然。用 **Laplace 近似**(8.54):

$$\log p(y | X) \approx \ell(\hat{f}) - \frac{1}{2} \log |H| + \text{const.} \quad (15.50)$$

即

$$\log p(y | X) \approx \log p(y | \hat{f}) - \frac{1}{2} \hat{f}^\top K^{-1} \hat{f} - \frac{1}{2} \log |K| - \frac{1}{2} \log |K^{-1} + W|. \quad (15.51)$$

求  $\partial \log p(y | X, \theta) / \partial \theta_j$  比回归更复杂, 因为  $\hat{f}$ 、 $W$ 、 $K$  都依赖  $\theta$ 。细节见 (Rasmussen & Williams 2006, 第 125 页)。

### 15.3.1.4 数值稳定的计算

为数值稳定, 应避免直接求  $K$  或  $W$  的逆。 (Rasmussen & Williams 2006, 第 45 页) 建议定义

$$B = I_N + W^{1/2} K W^{1/2}, \quad (15.52)$$

它的特征值下界为 1 (由  $I$ ) , 上界为  $1 + \frac{N}{4} \max_{ij} K_{ij}$  (因  $w_{ii} = \pi_i(1 - \pi_i) \leq 0.25$ ) , 因此可放心求逆。用矩阵求逆引理可得

$$(K^{-1} + W)^{-1} = K - K W^{1/2} B^{-1} W^{1/2} K. \quad (15.53)$$

于是 IRLS 更新

$$f_{\text{new}} = (K^{-1} + W)^{-1} b, \quad b = W f + \nabla \log p(y | f) \quad (15.54)$$

$$= K(I - W^{1/2} B^{-1} W^{1/2} K)b \quad (15.55)$$

$$= K(b - W^{1/2} L^\top (L^{-1}(W^{1/2} K b))), \quad B = LL^\top \quad (\text{Cholesky}) . \quad (15.56)$$

整体拟合耗时  $O(TN^3)$ 、用时空  $O(N^2)$ , 其中  $T$  是牛顿迭代次数。收敛时令  $a = K^{-1} \hat{f}$ , 则 (15.51) 可写为

$$\log p(y | X) = \log p(y | \hat{f}) - \frac{1}{2} a^\top \hat{f} - \sum_i \log L_{ii}, \quad (15.57)$$

这里用到了

$$|B| = |K| |K^{-1} + W| = |I_N + W^{1/2} K W^{1/2}|. \quad (15.58)$$

预测分布方面, 不用  $E[f_{\text{approx}}] = k_{\text{approx}}^\top K^{-1} \hat{f}$ 。注意在众数处  $\nabla \ell = 0 \Rightarrow \hat{f} = K(\nabla \log p(y | \hat{f}))$ , 故

$$\mathbb{E}[f_*] = k_{\cdot *}^\top \nabla \log p(y | f). \quad (15.59)$$

方差利用

$$(K + W^{-1})^{-1} = W^{1/2}(LL^\top)^{-1}W^{1/2} \quad (15.60)$$

得到

$$\text{var}[f_*] = k_{\cdot * *} - k_{\cdot *}^\top W^{1/2}(LL^\top)^{-1}W^{1/2}k_{\cdot *} = k_{\cdot * *} - v^\top v, \quad v = L^{-1}(W^{1/2}k_{\cdot *}) \quad (15.61)$$

随后即可算  $\pi_{\cdot *}$ 。完整算法见算法 16（基于 Rasmussen & Williams 2006，第 46 页）。拟合  $O(N^3)$ ，预测  $O(N^2 N_{\cdot *})$ 。

### 15.3.1.5 例子

图 15.7 给出一个二维的模拟二分类问题，SE 核。左图：用手工设定的超参数（长度尺度短），因此决策边界转折很尖锐。右图：用学习到的超参数，模型倾向于更简洁的解释。

### 15.3.2 多分类

设  $p(y_i | x_i) = \text{Cat}(y_i | S(f_i))$ ，其中

$f_i = (f_{i1}, \dots, f_{iC})$ ，并假设每个类别各有一条潜在函数  $f_{ic} \sim \text{GP}(0, \kappa_c)$ 。因此每类一个潜在 GP，**先验独立**，核也可不同。仍用高斯近似做推断。（一个类似模型使用多项 probit 而非多项 logit，见 Girolami & Rogers 2006。）

#### 15.3.2.1 计算后验

未归一化对数后验：

$$\ell(f) = -\frac{1}{2}f^\top K^{-1}f + y^\top f - \sum_{i=1}^N \log \sum_{c=1}^C e^{f_{ic}} - \frac{1}{2} \log |K| - \frac{CN}{2} \log 2\pi \quad (15.62)$$

其中

$$f = (f_{11}, \dots, f_{N1}, f_{12}, \dots, f_{N2}, \dots, f_{1C}, \dots, f_{NC})^\top, \quad (15.63)$$

$y$  为与  $f$  同形状的**one-hot**编码。 $K$  为**分块对角**，每个块  $K_c = [\kappa_c(x_i, x_j)]$  表示第  $c$  个潜在函数的相关性。梯度与 Hessian：

$$\nabla \ell = -K^{-1}f + y - \pi, \quad \nabla \nabla \ell = -K^{-1} - W \quad (15.64-65)$$

其中  $W = \text{diag}(\pi) - \Pi \Pi^\top$ ， $\Pi$  为把  $\text{diag}(\pi_{\cdot c})$  纵向堆叠得到的  $CN \times N$  矩阵。（可与 8.3.7 节的标准多项 logistic 回归对比。）用 IRLS 计算众数，牛顿步为

$$f_{\text{new}} = (K^{-1} + W)^{-1}(Wf + y - \pi). \quad (15.66)$$

朴素实现是  $O(C^3N^3)$ , 但可降到  $O(CN^3)$  (见 Rasmussen & Williams 2006, 第 52 页)。

### 15.3.2.2 计算后验预测

做法与 15.3.1.2 类似。潜变量均值:

$$\mathbb{E}[f_{\setminus c}^*] = k_c(x_{\setminus c}^*)^\top K_c^{-1} \hat{f}_{\setminus c} = k_c(x_{\setminus c}^*)^\top (y_c - \hat{\pi}_c). \quad (15.67)$$

向量式:

$$\mathbb{E}[f_{\setminus c}^*] = Q_{\setminus c}^\top (y - \hat{\pi}), \quad (15.68)$$

其中

$$Q_{\setminus c} = \begin{pmatrix} k_1(x_{\setminus c}) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & k_C(x_{\setminus c}) \end{pmatrix}. \quad (15.69)$$

协方差 (类比 15.47) :

$$\text{cov}[f_{\setminus c}^*] = \Sigma + Q_{\setminus c}^\top K^{-1} (K^{-1} + W)^{-1} K^{-1} Q_{\setminus c} \quad (15.70)$$

$$= \text{diag}(k(x_{\setminus c}, x_{\setminus c})) - Q_{\setminus c}^\top (K + W^{-1})^{-1} Q_{\setminus c}, \quad (15.71)$$

其中  $\Sigma$  为  $C \times C$  对角阵,  $\Sigma_{cc} = k_c(x_{\setminus c}, x_{\setminus c}) - k_c(x_{\setminus c})^\top K_c^{-1} k_c(x_{\setminus c})$ , 而  $k(x_{\setminus c}, x_{\setminus c}) = [k_c(x_{\setminus c}, x_{\setminus c})]$ 。可得可见响应的后验预测:

$$p(y | x_{\setminus c}, X, y) \approx \int \text{Cat}(y | S(f_{\setminus c}^*)) N(f_{\setminus c}^* | \mathbb{E}[f_{\setminus c}^*], \text{cov}[f_{\setminus c}^*]) df_{\setminus c}. \quad (15.72)$$

可用 21.8.1.1 节里对 softmax 的确定性近似, 或直接做 Monte Carlo。

### 15.3.2.3 计算边际似然

$$\log p(y | X) \approx -\frac{1}{2} \hat{f}^\top K^{-1} \hat{f} + y^\top \hat{f} - \sum_{i=1}^N \log \sum_{c=1}^C e^{\hat{f}_{ic}} - \frac{1}{2} \log |I_{CN} + W^{1/2} K W^{1/2}| \quad (15.73)$$

可按常规数值方法优化。

### 15.3.2.4 数值与计算

按 (Rasmussen & Williams 2006, 第 50 页) 的方法, 拟合的时间/空间为  $O(TCN^3)/O(CN^2)$ ; 预测为  $O(CN^3 + CN^2 N_{\setminus c})$ 。

### 15.3.3 用于 Poisson 回归的 GP

本节用 GP 做 Poisson 回归。一个有趣应用是**空间疾病制图**。例如 (Vanhatalo 等, 2010) 建模芬兰不同区域的心梗**相对风险**。数据为 1996–2000 年芬兰心梗病例, 按 20km × 20km 网格聚合。模型:

$$y_i \sim \text{Poi}(e_i r_i) \quad (15.74)$$

其中  $e_i$  是已知的期望死亡人数 (与第  $i$  个网格的人口和整体死亡率相关),  $r_i$  是我们要推断的**相对风险**。由于计数小, 我们通过在空间邻域间**共享信息**来正则化: 设  $f = \log r \sim \text{GP}(0, \kappa)$ , 核取 Matérn ( $\nu = 3/2$ ), 其长度尺度与幅度从数据估计。图 15.8 展示了 911 个位置数据得到的输出: 左图为相对风险的后验**均值**, 右图为后验**方差**。可见东部芬兰 RR 较高 (与其他研究一致); 北部人口稀少, 因此方差更大。

---

## 通俗解释

### 这节在做什么?

把“**分类/计数**”这些 GLM 里的任务也交给 GP: 仍然对**潜在函数** $f(x)$  放 GP 先验, 但用**链接函数**把  $f$  变成概率:

- 二分类:  $\sigma(f)$  (logistic 或 probit) ;
- 多分类: 对每类有一条潜在函数, 用 softmax;
- 计数 (Poisson) :  $\lambda = e r = \exp(f) \cdot e$ 。

### 为什么要近似?

因为“高斯先验 + 非高斯似然”→ 后验不再是高斯。我们用**高斯近似 (拉普拉斯近似/IRLS)**: 找到众数  $\hat{f}$ , 把后验近似成  $N(\hat{f}, (K^{-1} + W)^{-1})$ 。

### 表 15.1 的意思

- logistic:  $\log p(y_i|f_i) = \log \sigma(y_i f_i)$ 。梯度  $t_i - \pi_i$  (“标签-预测概率”), Hessian  $-\pi_i(1 - \pi_i)$  (不确定性权重)。
- probit: 把  $\sigma$  换成  $\Phi$ ; 梯度/二阶导如表, 数值上与 logit 类似。

### 怎么预测?

先得到  $f_{\backslash*}$  的高斯近似  $N(\mu_{\backslash*}, \nu_{\backslash*})$  (式 15.48); 再把它通过**链接函数积分** (式 15.49) 变成类别概率  $\pi_{\backslash*}$ ; 该积分用简单近似即可 (比如 probit 近似)。

### 数值上如何稳定?

- 不要直接“求逆”。引入  $B = I + W^{1/2} K W^{1/2}$ , 对它做 Cholesky  $B = LL^T$ , 再用三角求解“\”完成更新与方差计算 (式 15.56、15.61)。

- 复杂度：训练约  $O(N^3)$ , 预测  $O(N^2 N_{\text{**}})$ 。

### 图 15.7 的含义

长度尺度短 → 决策边界很“抖”；学习到的长长度尺度 → 更平滑、更简洁。黑粗线：0.5 概率阈值的边界。

### 图 15.8 的含义

Poisson GP 在地图上给出每个网格的“相对风险”均值与不确定性：东部风险偏高；北部样本少，所以不确定性大。

## 要点总结

- GP + GLM**：把 GP 放在潜在函数上，用链接函数连接到 Bernoulli/softmax/Poisson 似然。
- 后验近似**：用拉普拉斯/IRLS 得到  $p(f \mid X, y) \approx \mathcal{N}(\hat{f}, (K^{-1} + W)^{-1})$ 。
- 预测**：先得  $\hat{f}_{\text{**}}$  的高斯分布，再对链接函数做一维/低维积分近似得到类别/计数的预测。
- 表 15.1** 给出了 logistic/probit 的梯度与 Hessian：实现 IRLS 的关键。
- 数值技巧**：用  $B = I + W^{1/2} K W^{1/2}$  和 Cholesky，避免显式求逆；训练  $O(N^3)$ ，预测  $O(N^2 N_{\text{**}})$ 。
- 超参数影响**：长度尺度短 → 边界复杂；长 → 边界平滑（图 15.7）。
- Poisson GP 应用**：空间相对风险制图，能同时给出均值与不确定性（图 15.8）。

这一小节主要在讲什么？我不太懂

已思考 16s

这小节讲的是：怎样把高斯过程（GP）用到广义线性模型（GLM）里，尤其是分类和计数（Poisson）问题，以及实际该怎么做近似推断、预测和调参。

## 核心思路（一句话）

把不可直接观测的潜在函数  $f(x)$  放 GP 先验；再用链接函数把  $f(x)$  变成概率/强度（logistic、probit、softmax、Poisson 的 exp）；由于“高斯先验 + 非高斯似然”后验不再是高斯，所以用\*\*拉普拉斯/高斯近似（IRLS）\*\*来做。

## 主要内容拆解

## 1. 二分类 (logistic / probit)

- 模型:  $p(y|x) = \sigma(yf(x)); f \sim \text{GP}(0, \kappa)$ 。
- 用 IRLS 找后验众数  $\hat{f}$ , 把后验近似为  $N(\hat{f}, (K^{-1} + W)^{-1})$ 。
- 表 15.1 给出了 logit/probit 的梯度和 Hessian, 是 IRLS 的配方。
- 预测: 先得到  $f^*$  的高斯分布 (均值、方差), 再把它通过  $\sigma(\cdot)$  做一维积分近似得到类概率。
- 边际似然: 用拉普拉斯近似来估计, 用于学习核的超参数。
- 数值稳定: 避免显式求逆; 构造  $B = I + W^{1/2} K W^{1/2}$  做 Cholesky, 更稳更快。
- 图 15.7: 长度尺度短→边界“抖”; 学到的较长长度尺度→边界更平滑。

## 2. 多分类

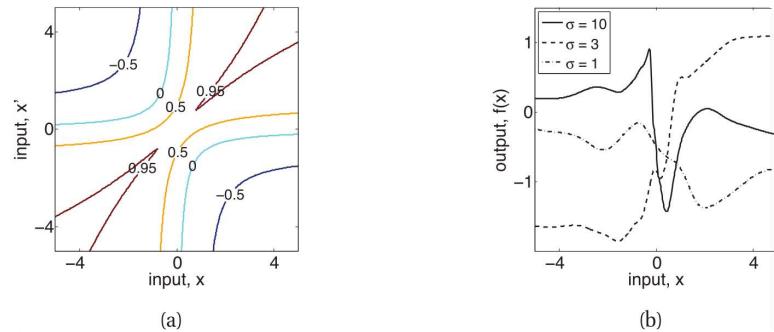
- 每个类别一条潜在函数  $f_c \sim \text{GP}$ , 用 softmax 连接到类别概率。
- 仍用高斯近似; 给出后验、预测的均值/协方差公式, 并说明计算复杂度如何降到  $O(CN^3)$ 。

## 3. Poisson 回归 (空间疾病制图示例)

- 计数模型:  $y_i \sim \text{Poi}(e_i r_i)$ , 令  $f = \log r \sim \text{GP}$  (常用 Matérn 核)。
- 产出每个格网的相对风险均值与不确定性; 图 15.8 展示芬兰心脏病的空间分布: 东部风险更高、人口稀少北部不确定性更大。

## 你需要记住的要点

- GP + 链接函数 = 适配 GLM (分类/计数)。
- 因为不共轭, 后验要近似: 最常用的是拉普拉斯/IRLS 的高斯近似。
- 预测 = 先算  $f^*$  的高斯分布, 再通过链接函数把它变成概率/强度。
- \*\*边际似然 (拉普拉斯近似) \*\*用来选核超参数。
- 数值实现用 Cholesky 和  $B = I + W^{1/2} K W^{1/2}$ , 避免求逆、稳又快。
- 多分类: 每类一个 GP; Poisson: 对对数强度放 GP, 可做空间风险图并给不确定性。



翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Figure 15.9 (a) Covariance function  $\kappa_{NN}(x, x')$  for  $\sigma_0 = 10$ ,  $\sigma = 10$ . (b) Samples from a GP with this kernel, using various values of  $\sigma$ .

#### 15.4 Connection with other methods

There are variety of other methods in statistics and machine learning that are closely related to GP regression/ classification. We give a brief review of some of these below.

##### 15.4.1 Linear models compared to GPs

Consider Bayesian linear regression for D-dimensional features, where the prior on the weights is  $p(w) = N(0, \Sigma)$ . The posterior predictive distribution is given by the following;

$$p(f^* | x^*, X, y) = N(\mu, \sigma^2) \quad (15.75)$$

$$\mu = 1/\sigma^2 y^T X^T * A^{-1} X y \quad (15.76)$$

$$\sigma^2 = x^T * A^{-1} x^* \quad (15.77)$$

where  $A = \sigma^{-2} y^T X^T X y + \Sigma^{-1}$ . One can show that we can rewrite the above distribution as follows

$$\mu = x^T * \Sigma X^T (K + \sigma^2 I)^{-1} y \quad (15.78)$$

$$\sigma^2 = x^T * \Sigma x^* - x^T * \Sigma X^T (K + \sigma^2 I)^{-1} X \Sigma x^* \quad (15.79)$$

where we have defined  $K = X \Sigma X^T$ , which is of size  $N \times N$ . Since the features only ever appear in the form  $X \Sigma X^T$ ,  $x^T * \Sigma X^T$  or  $x^T * \Sigma x^*$ , we can kernelize the above expression by defining  $\kappa(x, x') = x^T \Sigma x'$ . Thus we see that Bayesian linear regression is equivalent to a GP with covariance function  $\kappa(x, x') = x^T \Sigma x'$ . Note, however, that this is a degenerate covariance function, since it has at most D non-zero eigenvalues. Intuitively this reflects the fact that the model can only represent a limited number of functions. This can result in underfitting, since the model

is not flexible enough to capture the data. What is perhaps worse, it can result in overconfidence, since the model's prior is so impoverished that its posterior will become too concentrated. So not only is the model wrong, it thinks it's right!

### 15.4.2 Linear smoothers compared to GPs

A linear smoother is a regression function which is a linear function of the training outputs:

$$\hat{f}(x^*) = \sum_i w_i(x^*) y_i \quad (15.80)$$

where  $w_i(x^*)$  is called the weight function (Silverman 1984). (Do not confuse this with a linear model, where the output is a linear function of the input vector.) There are a variety of linear smoothers, such as kernel regression (Section 14.7.4), locally weighted regression (Section 14.7.5), smoothing splines (Section 15.4.6), and GP regression. To see that GP regression is a linear smoother, note that the mean of the posterior predictive distribution of a GP is given by

$$\hat{f}(x^*) = K^T (K + \sigma^2 I)^{-1} y = \sum_i w_i(x^*) y_i \quad (15.81)$$

where  $w_i(x^*) = [(K + \sigma^2 I)^{-1} k^*]_i$ .

In kernel regression, we derive the weight function from a smoothing kernel rather than a Mercer kernel, so it is clear that the weight function will then have local support. In the case of a GP, things are not as clear, since the weight function depends on the inverse of  $K$ . For certain GP kernel functions, we can analytically derive the form of  $w_i(x)$ ; this is known as the equivalent kernel (Silverman 1984). One can show that  $\sum_i w_i(x^*) = 1$ , although we may have  $w_i(x^*) < 0$ , so we are computing a linear combination but not a convex combination of the  $y_i$ 's. More interestingly,  $w_i(x^*)$  is a local function, even if the original kernel used by the GP is not local. Furthermore the effective bandwidth of the equivalent kernel of a GP automatically decreases as the sample size  $N$  increases, whereas in kernel smoothing, the bandwidth  $h$  needs to be set by hand to adapt to  $N$ . See e.g., (Rasmussen and Williams 2006, Sec 2.6, Sec 7.1) for details.

#### 15.4.2.1 Degrees of freedom of linear smoothers

It is clear why this method is called "linear", but why is it called a "smoother"? This is best explained in terms of GPs. Consider the prediction on the training set:

$$\hat{f} = K(K + \sigma^2 I)^{-1} y \quad (15.82)$$

Now let  $K$  have the eigendecomposition  $K = \sum_i \lambda_i u_i u_i^T$

. Since K is real and symmetric positive definite, the eigenvalues  $\lambda_i$  are real and non-negative, and the eigenvectors  $u_i$  are orthonormal. Now let  $y = N \sum_{i=1}^n y_i u_i$ , where  $y_i = u_i^T y$ . Then we can rewrite the above equation as follows:

$$f = N \sum_{i=1}^n y_i \lambda_i + \sigma^2 y u_i \quad (15.83)$$

This is the same as Equation 7.47, except we are working with the eigenvectors of the Gram matrix K instead of the data matrix X. In any case, the interpretation is similar: if  $\lambda_i \approx 0$ , then the corresponding basis function  $u_i$  will not have much influence. Consequently the highfrequency components in  $y$  are smoothed out. The effective degrees of freedom of the linear smoother is defined as

$$dof = \text{tr}(K(K + \sigma^2 I)^{-1}) - 1 = N \sum_{i=1}^n \lambda_i / (\lambda_i + \sigma^2) \quad (15.84)$$

This specifies how "wiggly" the curve is.

#### 15.4.3 SVMs compared to GPs

We saw in Section 14.5.2 that the SVM objective for binary classification is given by Equation 14.57

$$J(w) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n (1 - y_i w_i) \quad (15.85)$$

We also know from Equation 14.59 that the optimal solution has the form  $w = \sum_i \alpha_i x_i$ , so  $\|w\|^2 = \sum_{i,j} \alpha_i \alpha_j x_i^T x_j$ . Kernelizing we get  $\|w\|^2 = \alpha K \alpha$ . From Equation 14.61, and absorbing the  $w_0$  term into one of the kernels, we have  $f = K \alpha$ , so  $\|w\|^2 = f^T K f - 1$ . Hence the SVM objective can be rewritten as

$$J(f) = \frac{1}{2} f^T f + C \sum_{i=1}^n (1 - y_i f_i) \quad (15.86)$$

Compare this to MAP estimation for GP classifier:

$$J(f) = \frac{1}{2} f^T f - \sum_{i=1}^n \log p(y_i | f_i) \quad (15.87)$$

It is tempting to think that we can "convert" an SVM into a GP by figuring out what likelihood would be equivalent to the hinge loss. However, it turns out there is no such likelihood (Sollich 2002), although there is a pseudo-likelihood that matches the SVM (see Section 14.5.5).

From Figure 6.7 we saw that the hinge loss and the logistic loss (as well as the probit loss) are quite similar to each other. The main difference is that the hinge loss is strictly 0 for errors larger than 1. This gives rise to a sparse solution. In Section 14.3.2, we discussed other ways to derive sparse kernel machines. We discuss the connection between these methods and GPs below.

#### 15.4.4 L1VM and RVMs compared to GPs

Sparse kernel machines are just linear models with basis function expansion of the form  $\varphi(x) = [\kappa(x, x_1), \dots, \kappa(x, x_N)]$ .

From Section 15.4.1, we know that this is equivalent to a GP with the following kernel:

$$\kappa(x, x') = \sum_{j=1}^D \alpha_j \varphi_j(x) \varphi_j(x') \quad (15.88)$$

where  $\varphi_j(w) = N(0, \text{diag}(\alpha_j - 1))$ . This kernel function has two interesting properties. First, it is degenerate, meaning it has at most  $N$  non-zero eigenvalues, so the joint distribution  $p(f, f^*)$  will be highly constrained. Second, the kernel depends on the training data. This can cause the model to be overconfident when extrapolating beyond the training data. To see this, consider a point  $x^*$  far outside the convex hull of the data. All the basis functions will have values close to 0, so the prediction will back off to the mean of the GP. More worryingly, the variance will back off to the noise variance. By contrast, when using a non-degenerate kernel function, the predictive variance increases as we move away from the training data, as desired. See (Rasmussen and Quiñonero-Candela 2005) for further discussion.

#### 15.4.5 Neural networks compared to GPs

In Section 16.5, we will discuss neural networks, which are a nonlinear generalization of GLMs. In the binary classification case, a neural network is defined by a logistic regression model applied to a logistic regression model:

$$p(y|x, \theta) = \text{Ber}(y|\text{sigm}(w^T \text{sigm}(Vx))) \quad (15.89)$$

It turns out there is an interesting connection between neural networks and Gaussian processes, as first pointed out by (Neal 1996). To explain the connection, we follow the presentation of (Rasmussen and Williams 2006, p91). Consider a neural network for regression with one hidden layer. This has the form

$$p(y|x, \theta) = N(y|f(x; \theta), \sigma^2) \quad (15.90)$$

where

$$f(x) = b + \sum_{j=1}^H v_j g(x; u_j) \quad (15.91)$$

where  $b$  is the offset or bias term,  $v_j$  is the output weight from hidden unit  $j$  to the response  $y$ ,  $u_j$  are the inputs weights to unit  $j$  from the input  $x$ , and  $g()$  is the hidden unit activation function. This is typically the sigmoid or tanh function, but can be any smooth function. Let us use the following priors on the weights: where  $b \sim N(0, \sigma_b^2)$   $v_j \sim N(v_j|0, \sigma_v^2)$ ,  $u_j \sim N(u_j|0, \sigma_u^2)$  for some unspecified  $p(u_j)$ .

Denoting all the weights by  $\theta$  we have

$$E_\theta [f(x)] = 0 \quad (15.92)$$

$$E_\theta [f(x)f(x')] = \sigma_b^2 b^2 + \sum_j \sigma_v^2 v_j^2 E_v [g(x; u_j)g(x'; u_j)] \quad (15.93)$$

$$= \sigma^2 b + H\sigma^2 v \mathbb{E}_u [g(x; u)g(x'; u)] \quad (15.94)$$

where the last equality follows since the  $H$  hidden units are iid. If we let  $\sigma^2 v$  scale as  $\omega^2/H$  (since more hidden units will increase the input to the final node, so we should scale down the magnitude of the weights), then the last term becomes  $\omega^2 \mathbb{E}_u [g(x; u)g(x'; u)]$ . This is a sum over  $H$  iid random variables. Assuming that  $g$  is bounded, we can apply the central limit theorem. The result is that as  $H \rightarrow \infty$ , we get a Gaussian process.

If we use as activation / transfer function  $g(x; u) = \text{erf}(u_0 + D \sum_{j=1}^D u_j x_j)$ , where  $\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$ , and we choose  $u \sim N(0, \Sigma)$ , then (Williams 1998) showed that the covariance kernel has the form

$$\kappa_{NN}(x, x') = 2 \pi \sin -1 2 \tilde{x}^T \Sigma \tilde{x} (1 + 2 \tilde{x}^T \Sigma \tilde{x}) (1 + 2 (\tilde{x}^T \Sigma \tilde{x})) \quad (15.95)$$

where  $\tilde{x} = (1, x_1, \dots, x_D)$ . This is a true "neural network" kernel, unlike the "sigmoid" kernel  $\kappa(x, x') = \tanh(a + bx^T x')$ , which is not positive definite. Figure 15.9(a) illustrates this kernel when  $D = 2$  and  $\Sigma = \text{diag}(\sigma^2_0, \sigma^2_2)$ . Figure 15.9(b) shows some functions sampled from the corresponding GP. These are equivalent to functions which are superpositions of  $\text{erf}(u_0 + ux)$  where  $u_0$  and  $u$  are random. As  $\sigma^2$  increases, the variance of  $u$  increases, so the function varies more quickly. Unlike the RBF kernel, functions sampled from this kernel do not tend to 0 away from the data, but rather they tend to remain at the same value they had at the "edge" of the data. Now suppose we use an RBF network, which is equivalent to a hidden unit activation function of the form  $g(x; u) = \exp(-|x - u|^2/(2\sigma^2 g))$ . If  $u \sim N(0, \sigma^2 u)$ , one can show that the corresponding kernel is equivalent to the RBF or SE kernel.

#### 15.4.6 Smoothing splines compared to GPs

Smoothing splines are a widely used non-parametric method for smoothly interpolating data (Green and Silverman 1994). They are a special case of GPs, as we will see. They are usually used when the input is 1 or 2 dimensional.

##### 15.4.6.1 Univariate splines

The basic idea is to fit a function  $f$  by minimizing the discrepancy to the data plus a smoothing term that penalizes functions that are "too wiggly". If we penalize the  $m$ 'th derivative of the function, the objective becomes

$$J(f) = \sum_i (f(x_i) - y_i)^2 + \lambda (\int |f^{(m)}(x)|^2 dx) \quad (15.96)$$

One can show (Green and Silverman 1994) that the solution is a piecewise polynomial where the polynomials have order  $2m-1$  in the interior bins  $[x_{i-1}, x_i]$  (denoted  $I$ ), and order  $m-1$  in the two outermost intervals  $(-\infty, x_1]$  and  $[x_N, \infty)$ :

$$f(x) = \sum_{j=0}^{m-1} \beta_j x^j + \sum_{i=1}^N \alpha_i (x - x_i)^{2m-1} + \sum_{i=1}^N \alpha_i (x - x_i)^{m-1} \quad (15.97)$$

For example, if  $m=2$ , we get the (natural) cubic spline  $f(x) = \beta_0 + \beta_1 x + \sum_{i=1}^N \alpha_i (x - x_i)^3 + \sum_{i=1}^N \alpha_i (x - x_i)^2$  (15.98) which is a series of truncated cubic polynomials, whose left hand sides are located at each of the  $N$  training points. (The fact that the model is linear on the edges prevents it from extrapolating too wildly beyond the range of the data; if we drop this requirement, we get an “unrestricted” spline.) We can clearly fit this model using ridge regression:  $\hat{w} = (\Phi^T \Phi + \lambda I N)^{-1} \Phi^T y$ , where the columns of  $\Phi$  are 1,  $x_i$  and  $(x - x_i)^3 +$  for  $i=2:N-1$  and  $(x - x_i)^2$  for  $i=1$  or  $i=N$ . However, we can also derive an  $O(N)$  time method (Green and Silverman 1994, Sec 2.3.3).

#### 15.4.6.2 Regression splines

In general, we can place the polynomials at a fixed set of  $K$  locations known as knots, denoted  $\xi_k$ . The result is called a regression spline. This is a parametric model, which uses basis function expansion of the following form (where we drop the interior/ exterior distinction for simplicity):

$$f(x) = \beta_0 + \beta_1 x + \sum_{k=1}^K \alpha_j (x - \xi_k)^3 \quad (15.99)$$

Choosing the number and locations of the knots is just like choosing the number and values of the support vectors in Section 14.3.2. If we impose an  $L_2$  regularizer on the regression coefficients  $\alpha_j$ , the method is known as penalized splines. See Section 9.6.1 for a practical example of penalized splines.

#### 15.4.6.3 The connection with GPs

One can show (Rasmussen and Williams 2006, p139) that the cubic spline is the MAP estimate of the following function

$$f(x) = \beta_0 + \beta_1 x + r(x) \quad (15.100)$$

where  $p(\beta_j) \propto 1$  (so that we don’t penalize the zero’th and first derivatives of  $f$ ), and  $r(x) \sim GP(0, \sigma^2 k_{sp}(x, x))$ , where  $k_{sp}(x, x) = 1 - \frac{1}{2} |x - x'| + \frac{1}{2} |x - x'|^2$  (15.101)

Note that the kernel in Equation 15.101 is rather

unnatural, and indeed posterior samples from the resulting GP are rather unsMOOTH. However, the posterior mode/mean is smooth. This shows that regularizers don't always make good priors.

#### 15.4.6.4 2d input (thin-plate splines)

One can generalize cubic splines to 2d input by defining a regularizer of the following form:

$$\partial^2 f(x) \partial x_2 1 2 + 2 \partial^2 f(x) \partial x_1 \partial x_2 2 + \partial^2 f(x) \partial x_2 2 2 dx_1 dx_2 \quad (15.102)$$

One can show that the solution has the form

$$f(x) = \beta_0 + \beta^T 1 x + \sum_{i=1}^N \alpha_i \varphi_i(x) \quad (15.103)$$

where  $\varphi_i(x) = \eta(\|x - x_i\|)$ , and  $\eta(z) = z^2 \log z$ . This is known as a thin plate spline. This is equivalent to MAP estimation with a GP whose kernel is defined in (Williams and Fitzgibbon 2006).

#### 15.4.6.5 Higher-dimensional inputs

It is hard to analytically solve for the form of the optimal solution when using higher-order inputs. However, in the parametric regression spline setting, where we forego the regularizer on  $f$ , we have more freedom in defining our basis functions. One way to handle multiple inputs is to use a tensor product basis, defined as the cross product of 1d basis functions. For example, for 2d input, we can define

$$f(x_1, x_2) = \beta_0 + m \beta_1 m(x_1 - \xi_1 m) + m \beta_2 m(x_2 - \xi_2 m) + \quad (15.104)$$

$$+ m \beta_{12} m(x_1 - \xi_1 m)(x_2 - \xi_2 m) \quad (15.105)$$

It is clear that for high-dimensional data, we cannot allow higher-order interactions, because there will be too many parameters to fit. One approach to this problem is to use a search procedure to look for useful interaction terms.

This is known as MARS, which stands for "multivariate adaptive regression splines". See Section 16.3.3 for details.

#### 15.4.7 RKHS methods compared to GPs

We can generalize the idea of penalizing derivatives of functions, as used in smoothing splines, to fit functions with a more general notion of smoothness. Recall from Section 14.2.3 that Mercer's theorem says that any positive definite kernel function can be represented in terms of eigenfunctions:

$$\kappa(x, x') = \sum_{i=1}^{\infty} \lambda_i \varphi_i(x) \varphi_i(x') \quad (15.106)$$

The  $\varphi_i$  form an orthonormal basis for a function space:

$$H_k = \{f : f(x) = \sum_{i=1}^{\infty} f_i \varphi_i(x), \sum_{i=1}^{\infty} f_i^2 / \lambda_i < \infty\} \quad (15.107)$$

Now define the inner product between two functions  $f(x) = \sum_{i=1}^{\infty} f_i \varphi_i(x)$  and  $g(x) = \sum_{i=1}^{\infty} g_i \varphi_i(x)$  in this space as follows:

$$\langle f, g \rangle_H = \sum_{i=1}^{\infty} f_i g_i \lambda_i \quad (15.108)$$

In Exercise 15.1, we show that this definition implies that  $\langle \kappa(x_1, \cdot), \kappa(x_2, \cdot) \rangle_H = \kappa(x_1, x_2)$  (15.109)

This is called the reproducing property, and the space of functions  $H_k$  is called a reproducing kernel Hilbert space or RKHS. Now consider an optimization problem of the form

$$J(f) = \frac{1}{2\sigma^2} \sum_{i=1}^N (y_i - f(x_i))^2 + \frac{1}{2} \|f\|_H^2 \quad (15.110)$$

where  $\|f\|_H$  is the norm of a function:

$$\|f\|_H^2 = \langle f, f \rangle_H = \sum_{i=1}^{\infty} f_i^2 \lambda_i \quad (15.111)$$

The intuition is that functions that are complex wrt the kernel will have large norms, because they will need many eigenfunctions to represent them. We want to pick a simple function that provides a good fit to the data. One can show (see e.g., (Schoelkopf and Smola 2002)) that the solution must have the form

$$f(x) = \sum_{i=1}^N \alpha_i \kappa(x, x_i) \quad (15.112)$$

This is known as the representer theorem, and holds for other convex loss functions besides squared error. We can solve for the  $\alpha$  by substituting in  $f(x) = \sum_{i=1}^N \alpha_i \kappa(x, x_i)$  and using the reproducing property to get

$$J(\alpha) = \frac{1}{2\sigma^2} \sum_{i=1}^N |y_i - \sum_{j=1}^N \alpha_j \kappa(x_i, x_j)|^2 + \frac{1}{2} \alpha^T K \alpha \quad (15.113)$$

Minimizing wrt  $\alpha$  we find

$$\hat{\alpha} = (K + \sigma^2 I)^{-1} y \quad (15.114)$$

and hence

$$\hat{f}(x^*) = \sum_{i=1}^N \hat{\alpha}_i \kappa(x^*, x_i) = K^T (K + \sigma^2 I)^{-1} y \quad (15.115)$$

This is identical to Equation 15.18, the posterior mean of a GP predictive distribution. Indeed, since the mean and mode of a Gaussian are the same, we can see that linear regression with an RKHS regularizer is equivalent to MAP estimation with a GP. An analogous statement holds for the GP logistic regression case, which also uses a convex likelihood / loss function.

下面先逐句完整翻译（含公式和图注），再给通俗解释，最后做要点总结。

## 翻译（不遗漏）

## 图 15.9

- (a) 当  $\sigma_0 = 10$ ,  $\sigma = 10$  时的协方差函数  $\kappa_{NN}(x, x')$ 。
  - (b) 使用该核从 GP 中抽样得到的若干函数, 分别采用不同的  $\sigma$  值。
- 

## 15.4 与其他方法的联系

在统计与机器学习中, 有很多与 GP 回归/分类密切相关的方法。下面简要回顾其中一些。

### 15.4.1 线性模型与 GPs 的比较

考虑  $D$  维特征的贝叶斯线性回归, 权重的先验为  $p(w) = N(0, \Sigma)$ 。其后验预测分布为

$$p(f_{\setminus *} | x_{\setminus *}, X, y) = N(\mu, \sigma^2) \quad (15.75)$$

$$\mu = \frac{1}{\sigma_y^2} x_{\setminus *}^\top A^{-1} X^\top y \quad (15.76)$$

$$\sigma^2 = x_{\setminus *}^\top A^{-1} x_{\setminus *} \quad (15.77)$$

其中  $A = \sigma_y^{-2} X^\top X + \Sigma^{-1}$ 。可以把上式改写为

$$\mu = x_{\setminus *}^\top \Sigma X^\top (K + \sigma_y^2 I)^{-1} y \quad (15.78)$$

$$\sigma^2 = x_{\setminus *}^\top \Sigma x_{\setminus *} - x_{\setminus *}^\top \Sigma X^\top (K + \sigma_y^2 I)^{-1} X \Sigma x_{\setminus *} \quad (15.79)$$

这里定义  $K = X \Sigma X^\top \in \mathbb{R}^{N \times N}$ 。注意特征只以  $X \Sigma X^\top$ 、 $x_{\setminus *}^\top \Sigma X^\top$  或  $x_{\setminus *}^\top \Sigma x_{\setminus *}$  的形式出现, 因此令

$\kappa(x, x') = x^\top \Sigma x'$  就可以把上式核化。于是可见: **贝叶斯线性回归等价于协方差函数为  $\kappa(x, x') = x^\top \Sigma x'$  的 GP**。

但要注意该核是退化的: 其非零特征值至多  $D$  个。直观上, 这意味着模型能表达的函数数量有限, 可能造成欠拟合。更糟的是, 由于先验太贫乏, 后验可能过于自信——模型既错, 还“很确定自己是对的”。

### 15.4.2 线性平滑器与 GPs 的比较

线性平滑器指预测函数是训练输出的线性组合:

$$\hat{f}(x_{\setminus *}) = \sum_i w_i(x_{\setminus *}) y_i \quad (15.80)$$

其中  $w_i(x_{\setminus *})$  为**权重函数** (Silverman 1984)。这不同于“线性模型”(输出是输入的线性函数)。常见线性平滑器包括核回归 (14.7.4)、局部加权回归 (14.7.5)、平滑样条 (15.4.6) 以及 GP 回归。

GP 是线性平滑器, 因为其后验预测均值

$$f(x_{\setminus *}) = k_{\setminus *}^\top (K + \sigma_y^2 I_N)^{-1} y = \sum_{i=1}^N y_i w_i(x_{\setminus *}) \quad (15.81)$$

其中  $w_i(x_{\setminus *}) = [(K + \sigma_y^2 I_N)^{-1} k_{\setminus *}]_i$

在核回归里，权重来自“平滑核”（非 Mercer 核），因此权重**局部支撑**是显然的。对于 GP，情况不那么直观，因为权重依赖于  $K^{-1}$ 。对某些 GP 核，可以解析推导  $w_i(x)$  的形式，这称为**等价核** (equivalent kernel, Silverman 1984)。可证明  $\sum_i w_i(x_{\setminus *}) = 1$ ，但  $w_i(x_{\setminus *})$  可为负值——因此是线性组合而非凸组合。更有趣的是，即使 GP 使用的原核不是**局部的**， $w_i(x_{\setminus *})$  也往往是**局部函数**。此外，GP 等价核的**有效带宽**会随样本量  $N$  增大而**自动缩小**；而在核平滑中，带宽  $h$  需要手工随  $N$  调整。详见 (Rasmussen & Williams 2006, 2.6、7.1 节)。

#### 15.4.2.1 线性平滑器的自由度

为何称它为“平滑器”？用 GP 最好解释。训练集上的预测：

$$f = K(K + \sigma_y^2 I)^{-1} y \quad (15.82)$$

令  $K = \sum_{i=1}^N \lambda_i u_i u_i^\top$  (特征分解； $\lambda_i \geq 0$ ,  $u_i$  正交归一)，再写  $y = \sum_i \gamma_i u_i$ ,  $\gamma_i = u_i^\top y$ 。则

$$f = \sum_{i=1}^N \gamma_i \frac{\lambda_i}{\lambda_i + \sigma_y^2} u_i \quad (15.83)$$

含义与式 7.47 类似：若  $\frac{\lambda_i}{\lambda_i + \sigma_y^2}$  很小，则对应“高频”基函数  $u_i$  影响弱，于是把  $y$  的高频成分**平滑掉**。线性平滑器的**有效自由度**定义为

$$\text{dof} = \text{tr}(K(K + \sigma_y^2 I)^{-1}) = \sum_{i=1}^N \frac{\lambda_i}{\lambda_i + \sigma_y^2} \quad (15.84)$$

它刻画曲线“有多抖”。

#### 15.4.3 SVM 与 GPs 的比较

二分类的 SVM 目标 (14.57)：

$$J(w) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N (1 - y_i f_i)_+ \quad (15.85)$$

由 14.59 得最优  $w = \sum_i \alpha_i x_i \Rightarrow \|w\|^2 = \sum_{i,j} \alpha_i \alpha_j x_i^\top x_j$ 。核化得  $\|w\|^2 = \alpha^\top K \alpha$ 。由 14.61 (把  $\hat{w}_0$  吸收进核) 有  $f = K \alpha \Rightarrow \|w\|^2 = f^\top K^{-1} f$ 。故

$$J(f) = \frac{1}{2} f^\top f + C \sum_i (1 - y_i f_i)_+ \quad (15.86)$$

而 GP 分类的 MAP 目标为

$$J(f) = \frac{1}{2} f^T f - \sum_i \log p(y_i | f_i) \quad (15.87)$$

看似可把铰链损失当成某个似然的负对数，但事实上**不存在**与铰链损失等价的真正似然 (Sollich 2002)，只有伪似然（见 14.5.5）。铰链损失与 logistic/probit 损失很像，主要差别在：当误差超过 1 时铰链损失**严格为 0**，因此产生**稀疏解**。14.3.2 节还介绍了其它稀疏核机方法，下面会谈它们与 GP 的关系。

#### 15.4.4 L1VM 与 RVM 与 GPs 的比较

稀疏核机本质上是形如  $\phi(x) = [\kappa(x, x_1), \dots, \kappa(x, x_N)]$  的**线性模型**。由 15.4.1 可知，这等价于核

$$\kappa(x, x') = \sum_{j=1}^D \frac{1}{\alpha_j} \phi_j(x) \phi_j(x') \quad (15.88)$$

并对权重给先验  $p(w) = N(0, \text{diag}(\alpha_j^{-1}))$ 。此核有两点：

- (1) **退化**：非零特征值最多  $N$ ，因而联合分布  $p(f, f_{\setminus *})$  受强约束；
- (2) **依赖训练数据**：外推时可能**过度自信**。若  $x_{\setminus *}$  远离数据的凸包，所有基函数近 0，预测会退回 GP 的均值，更糟的是方差会退回**噪声方差**。相反，用**非退化核**时，离开训练数据越远，**预测方差越大**（理想特性）。详见 Rasmussen & Quiñonero-Candela (2005)。

#### 15.4.5 神经网络与 GPs 的比较

第 16.5 节将讲神经网络（GLM 的非线性推广）。二分类时，神经网络可写成“对 logistic 的 logistic”：

$$p(y | x, \theta) = \text{Ber}(y | \text{sigm}(w^T \text{sigm}(Vx))) \quad (15.89)$$

Neal (1996) 首先指出 NN 与 GP 的联系。按 Rasmussen & Williams (2006, p91) 的表述，考虑一层隐层的**回归**网络：

$$p(y | x, \theta) = N(y | f(x; \theta), \sigma^2) \quad (15.90)$$

$$f(x) = b + \sum_{j=1}^H v_j g(x; u_j) \quad (15.91)$$

其中  $b$  为偏置； $v_j$  为隐层到输出的权重； $u_j$  为输入到第  $j$  个隐单元的权重； $g$  是激活函数（sigmoid/tanh 等平滑函数）。令先验

$b \sim N(0, \sigma_b^2)$ ,  $v_j \sim N(0, \sigma_v^2)$ ,  $u_j \sim p(u_j)$ 。记所有权重为  $\theta$ ，则

$$E_\theta[f(x)] = 0 \quad (15.92)$$

$$\mathbb{E}_\theta[f(x)f(x')] = \sigma_b^2 + \sum_j \sigma_v^2 \mathbb{E}_v[g(x; u_j)g(x'; u_j)] = \sigma_b^2 + H\sigma_v^2 \mathbb{E}_u[g(x; u)g(x'; u)] \quad (15.94)$$

若令  $\sigma_v^2 = \omega^2/H$  (隐单元多则需缩放)，最后一项变为  $\omega^2 \mathbb{E}_u[g(x; u)g(x'; u)]$ 。这是  $H$  个 iid 随机变量之和；若  $g$  有界，用中心极限定理得：当  $H \rightarrow \infty$ ，得到一个**高斯过程**。

若  $g(x; u) = \text{erf}(u_0 + \sum_{j=1}^D u_j x_j)$ ,  $\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$ , 并取  $u \sim N(0, \Sigma)$ , 则 (Williams 1998) 得到核

$$\kappa_{NN}(x, x') = \frac{2}{\pi} \sin^{-1} \left( \frac{2 \tilde{x}^\top \Sigma \tilde{x}'}{\sqrt{(1 + 2 \tilde{x}^\top \Sigma \tilde{x})(1 + 2 \tilde{x}'^\top \Sigma \tilde{x}')}} \right), \quad \tilde{x} = (1, x_1, \dots, x_D) \quad (15.95)$$

这是真正的“神经网络核”，不同于**并非正定**的“sigmoid 核” $\kappa(x, x') = \tanh(a + b x^\top x')$ 。图 15.9(a) 展示了  $D = 2$ ,  $\Sigma = \text{diag}(\sigma_0^2, \sigma^2)$  时该核；图 15.9(b) 展示相应 GP 的函数样本：可看作若干  $\text{erf}(u_0 + u^\top x)$  的叠加，其中  $u_0, u$  随机。随着  $\sigma^2$  增大， $u$  的方差增大，函数变化**更快**。不同于 RBF 核，采样函数不会随远离数据而趋向 0，而是趋向于在“数据边缘”的值附近保持。若使用 RBF 网络（激活  $g(x; u) = \exp(-\|x - u\|^2/(2\sigma_g^2))$  且  $u \sim N(0, \sigma_u^2 I)$ ），可证明其对应核等价于 RBF/SE 核。

## 15.4.6 平滑样条与 GPs 的比较

平滑样条是常用的非参数平滑插值方法 (Green & Silverman 1994)，是 GPs 的一个特例，常用于 1-2 维输入。

### 15.4.6.1 一元样条

基本思想：拟合函数  $f$ ，既要贴近数据，也要用“光滑项”惩罚过于“抖”的函数。若惩罚  $m$  阶导数，目标为

$$J(f) = \sum_{i=1}^N (f(x_i) - y_i)^2 + \lambda \int \left( \frac{d^m}{dx^m} f(x) \right)^2 dx \quad (15.96)$$

可证 (Green & Silverman 1994) 解是**分段多项式**：在内部区间  $[x_{i-1}, x_i]$  (记作  $I$ ) 为**阶数**  $2m-1$  的多项式，在两端外区间  $(-\infty, x_1]$ 、 $[x_N, \infty)$  为**阶数**  $m-1$ ：

$$f(x) = \sum_{j=0}^{m-1} \beta_j x^j + \mathbf{1}(x \in I) \sum_{i=1}^N \alpha_i (x - x_i)_+^{2m-1} + \mathbf{1}(x \in I) \sum_{i=1}^N \alpha_i (x - x_i)_+^{m-1} \quad (15.97)$$

例如  $m = 2$  得\*\* (自然) 三次样条\*\*：

$$f(x) = \beta_0 + \beta_1 x + \mathbf{1}(x \in I) \sum_{i=1}^N \alpha_i (x - x_i)_+^3 + \mathbf{1}(x \in I) \sum_{i=1}^N \alpha_i (x - x_i)_+ \quad (15.98)$$

它是若干**截断三次多项式**之和，折点在每个训练点处。（边缘为线性，避免外推过度；若取消此限制即“非自然”样条。）可用岭回归求解： $\hat{w} = (\Phi^\top \Phi + \lambda I)^{-1} \Phi^\top y$ ， $\Phi$  的列包含常数、 $x_i$ 、 $(x - x_i)_+^3$  ( $i = 2:N - 1$ ) 与  $(x - x_i)_+$  ( $i = 1$  或  $N$ )。也可用  $O(N)$  算法（Green & Silverman 2.3.3 节）。

#### 15.4.6.2 回归样条

更一般地，在固定的  $K$  个结点  $\zeta_k$  放多项式，得到**回归样条**（参数模型）：

$$f(x) = \beta_0 + \beta_1 x + \sum_{k=1}^K \alpha_k (x - \zeta_k)_+^3 \quad (15.99)$$

选结点的数量与位置，类似 14.3.2 节中选择支持向量的数量与位置。若对  $\alpha_k$  加  $L_2$  正则，称**惩罚样条**（9.6.1 有实例）。

#### 15.4.6.3 与 GPs 的联系

可证（Rasmussen & Williams 2006, p139）：**三次样条是**下式的 MAP 估计：

$$f(x) = \beta_0 + \beta_1 x + r(x) \quad (15.100)$$

其中  $p(\beta_j) \propto 1$ （不惩罚 0、1 阶导），且  $r(x) \sim \text{GP}(0, \sigma_f^2 \kappa_{\text{sp}}(x, x'))$ ，

$$\kappa_{\text{sp}}(x, x') = \int_0^1 (x - u)_+ (x' - u)_+ du \quad (15.101)$$

注意 15.101 的核比较“反直觉”，其后验**样本**并不很平滑，但后验**众数/均值**是光滑的。这表明：**正则项不一定是好先验**。

#### 15.4.6.4 二维输入（薄板样条）

把三次样条推广到 2D，可定义如下正则：

$$\iint \left( \frac{\partial^2 f}{\partial x_1^2} \right)^2 + 2 \left( \frac{\partial^2 f}{\partial x_1 \partial x_2} \right)^2 + \left( \frac{\partial^2 f}{\partial x_2^2} \right)^2 dx_1 dx_2 \quad (15.102)$$

解的形式为

$$f(x) = \beta_0 + \beta_1^\top x + \sum_{i=1}^N \alpha_i \phi_i(x) \quad (15.103)$$

其中  $\phi_i(x) = \eta(\|x - x_i\|)$ ， $\eta(z) = z^2 \log z^2$ 。这就是**薄板样条**。它等价于某个 GP（其核见 Williams & Fitzgibbon 2006）的 MAP 估计。

#### 15.4.6.5 更高维输入

高维时解析解困难。在**回归样条**（放弃对 $f$ 的全局正则）设定下，可自由设定基函数。多输入的一种办法是用**张量积基**（1D 基函数的笛卡儿积）。例如二维：

$$f(x_1, x_2) = \beta_0 + \sum_m \beta_{1m} (x_1 - \xi_{1m})_+ + \sum_m \beta_{2m} (x_2 - \xi_{2m})_+ \quad (15.104)$$

$$+ \sum_m \beta_{12m} (x_1 - \xi_{1m})_+ (x_2 - \xi_{2m})_+ \quad (15.105)$$

高维下若允许高阶交互，参数会爆炸；可用**MARS**（多元自适应回归样条）用搜索方式挑选有用的交互（见 16.3.3）。

### 15.4.7 RKHS 方法与 GPs 的比较

把“惩罚导数”的思想推广，用更一般的“光滑度”拟合函数。回忆 14.2.3：Mercer 定理说任意正定核可写为

$$\kappa(x, x') = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(x') \quad (15.106)$$

$\{\phi_i\}$  构成函数空间的正交基：

$$H_{\kappa} = \{f : f(x) = \sum_{i=1}^{\infty} f_i \phi_i(x), \sum_i \frac{f_i^2}{\lambda_i} < \infty\} \quad (15.107)$$

在此空间中， $f = \sum f_i \phi_i$ ,  $g = \sum g_i \phi_i$  的内积定义为

$$\langle f, g \rangle_H = \sum_{i=1}^{\infty} \frac{f_i g_i}{\lambda_i} \quad (15.108)$$

练习 15.1 可得

$$\langle \kappa(x_1, \cdot), \kappa(x_2, \cdot) \rangle_H = \kappa(x_1, x_2) \quad (15.109)$$

这称为**再生性**，该空间称**再生核希尔伯特空间 (RKHS)**。

考虑优化问题

$$J(f) = \frac{1}{2\sigma_y^2} \sum_{i=1}^N (y_i - f(x_i))^2 + \frac{1}{2} \|f\|_H^2 \quad (15.110)$$

其中函数范数

$$\|f\|_H^2 = \langle f, f \rangle_H = \sum_{i=1}^{\infty} \frac{f_i^2}{\lambda_i} \quad (15.111)$$

直觉：相对核而言“复杂”的函数需要很多特征函数来表示，范数就大。我们希望选一个简单且拟合好的数据。可证 (Schoelkopf & Smola 2002)，解必具有

$$f(x) = \sum_{i=1}^N \alpha_i \kappa(x, x_i) \quad (15.112)$$

这就是**表征定理** (representer theorem)，在其它凸损失下也成立。把 (15.112) 代入，并用再生性，有

$$J(\alpha) = \frac{1}{2\sigma_y^2} \|y - K\alpha\|^2 + \frac{1}{2} \alpha^\top K \alpha \quad (15.113)$$

对  $\alpha$  最小化得

$$\hat{\alpha} = (K + \sigma_y^2 I)^{-1} y \quad (15.114)$$

因此

$$\hat{f}(x_{*}) = \sum_i \hat{\alpha}_i \kappa(x_{*}, x_i) = k_{*}^\top (K + \sigma_y^2 I)^{-1} y \quad (15.115)$$

这与式 (15.18) (GP 预测的**后验均值**) 完全一样。由于高斯的均值=众数，**带 RKHS 正则的线性回归等价于 GP 的 MAP**。对 GP logistic 回归也有类似结论 (同为凸损失)。

## 通俗解释

- 线性模型是 GP 的特例：**把核取成“线性核” $\kappa(x, x') = x^\top \Sigma x'$ ，贝叶斯线性回归  $\leftrightarrow$  GP。缺点是表达能力受限（退化核），容易欠拟合、且不确定性会被**低估**。
- GP 是一种“加权平均器”：**预测是训练标签的加权和（权重自动学出、会随  $N$  自适应缩小作用范围）。“有效自由度”告诉你曲线有多“抖”。
- SVM vs GP：**SVM 最小化“铰链损失 +  $L_2$  正则”（稀疏、判别式）；GP 做的是“（负）对数似然 + 先验”的**贝叶斯**版本（能给概率与不确定性）。铰链损失不是任何真正似然的对数。
- L1VM/RVM (稀疏核机) 的问题：**对应的核既退化又**依赖训练点**，外推时常“信心过头”；好的 GP 核会在离数据远处**增大方差**。
- 神经网络  $\leftrightarrow$  GP：**一层宽度  $H \rightarrow \infty$  的 NN，在合适的权重先验与缩放下，其函数分布  $\Rightarrow$  GP；不同激活给出不同“NN 核”。erf 激活产生的  $\kappa_{NN}$  如图 15.9； $\sigma$  越大，函数越“抖”。RBF 网络对应的就是 SE 核。
- 平滑样条  $\leftrightarrow$  GP 的 MAP：**三次样条等价于某个奇特核的 GP 的**众数**；这也提醒：常用的**正则不一定就是“好先验”**。

- **RKHS 与 GP:** 在 RKHS 里做“平方误差 + 函数范数”最小化的点估计，与 GP 预测均值相同（表征定理）。区别：GP 还给出了完整不确定性。

## 要点总结

- 线性回归 = 核为线性核的 GP (退化, 表达力有限)。
- GP 预测均值是**线性平滑器**；权重局部、会随样本数自适应。
- “有效自由度” $\sum \lambda_i / (\lambda_i + \sigma_y^2)$  衡量平滑程度。
- **SVM 与 GP:** 损失/似然不同；SVM 稀疏快，GP 给概率和不确定性更好。
- **稀疏核机 (L1VM/RVM)** \*\*的核依赖数据、外推易过自信；非退化 GP 核会远离数据时增大方差。
- **NN 与 GP:** 无限宽 NN  $\Rightarrow$  GP；erf 激活产生  $\kappa_{NN}$  (图 15.9)， $\sigma$  越大越“抖”；RBF 网络对应 SE 核。
- **平滑样条**是某 GP 的 MAP；二维对应**薄板样条**；高维可用**张量积基**或 MARS。
- **RKHS:** 正则化回归的解  $f(x) = \sum_i \alpha_i \kappa(x, x_i)$ ，与 GP 后验均值一致 (但 GP 还给方差/置信度)。

这一小节主要在讲什么？我不太懂

已思考 12s

这小节其实是在**把高斯过程 (GP) 放到“大地图”里，说明它跟常见方法的关系与区别**。你可以把它当成“亲戚关系表”。核心点如下：

### 一句话

GP = 一个能给出预测+不确定性的核方法；很多常见方法都可看成它的特例、近亲或对照组。

### 主要关系图 (通俗版)

- **贝叶斯线性回归  $\leftrightarrow$  GP 特例**  
取线性核  $\kappa(x, x') = x^\top \Sigma x'$  时，贝叶斯线性回归就等价于 GP。缺点：这个核是**退化的** (表达力有限)，容易**欠拟合**且**过自信**。
- **线性平滑器  $\leftrightarrow$  GP 后验均值**  
GP 的预测均值是训练标签的**加权和** (线性平滑)。权重总体上是**局部**的且会随样本数 N 自动收缩带宽；还能用特征值给出“**自由度**” $\sum \lambda_i / (\lambda_i + \sigma_y^2)$  衡量曲线有多“抖”。

- **SVM vs GP**

SVM 最小化“铰链损失+ $L_2$  正则”（产生稀疏解），不是任何真正似然；

GP 做的是概率模型（似然+先验），能输出校准的概率与不确定性。两者目标函数看起来像，但不等价。

- **稀疏核机 (L1VM/RVM) vs GP**

本质是用训练点做基函数的线性模型，对应一个退化且依赖数据的核：外推时常过度自信；而好的 GP 核会在远离数据时方差变大（更靠谱）。

- **神经网络  $\leftrightarrow$  GP (无限宽极限)**

隐层数固定、宽度  $H \rightarrow \infty$  时，在合适先验与缩放下，NN 的函数分布  $\Rightarrow$  GP。

用 erf 激活可得到神经网络核  $\kappa_{\text{NN}}$ （图 15.9）； $\sigma$  越大，函数变化越快。RBF 网络对应的就是 SE/RBF 核。

- **平滑样条  $\leftrightarrow$  GP 的 MAP**

三次样条等价于某个特殊核的 GP 的众数解；二维是薄板样条。提示：常用的正则项不一定是好先验，但 MAP 解会很平滑。

- **RKHS 正则化  $\leftrightarrow$  GP**

在 RKHS 里做“平方误差 + 函数范数”最小化，解必形如

$f(x) = \sum_i \alpha_i \kappa(x, x_i)$ （表征定理），得到的  $\hat{f}$  与 GP 预测均值完全一致；但 GP 还给出方差/不确定性。

## 记住这些就够了

1. **GP 统一视角：**线性回归、核平滑、样条、RKHS、无限宽 NN 都能在 GP 框架下被解释。
2. **差别关键：**GP 是贝叶斯概率模型，天然给不确定性；很多替代方法更快/更稀疏，但不提供或不够好的概率。
3. **退化核要小心：**表达力有限、外推易过自信；选非退化、合适的核更安全。
4. **图 15.9：**展示神经网络核的形状以及用它采样到的函数， $\sigma$  越大曲线越“抖”。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Figure 15.10 2d representation of 12 dimensional oil flow data. The different colors/symbols represent the 3 phases of oil flow. (a) Kernel PCA with Gaussian kernel. (b) GP-LVM with Gaussian kernel. The shading represents the

precision of the posterior, where lighter pixels have higher precision. From Figure 1 of (Lawrence 2005).

### 15.5 GP latent variable model

In Section 14.4.4, we discussed kernel PCA, which applies the kernel trick to regular PCA. In this section, we discuss a different way to combine kernels with probabilistic PCA. The resulting method is known as the GP-LVM, which stands for “Gaussian process latent variable model” (Lawrence 2005). To explain the method, we start with PPCA. Recall from Section 12.2.4 that the PPCA model is as follows:

$$p(z_i) = N(z_i | 0, I) \quad (15.116)$$

$$p(y_i | z_i, \theta) = N(y_i | Wz_i, \sigma^2 I) \quad (15.117)$$

We can fit this model by maximum likelihood, by integrating out the  $z_i$  and maximizing  $W$  (and  $\sigma^2$ ). The objective is given by

$$p(Y|W, \sigma^2) = (2\pi)^{-DN/2} |C|^{-N/2} \exp \left[ -\frac{1}{2} \text{tr}(C^{-1} Y T Y) \right] \quad (15.118)$$

where  $C = WWT + \sigma^2 I$ . As we showed in Theorem 12.2.2, the MLE for this can be computed in terms of the eigenvectors of  $Y T Y$ . Now we consider the dual problem, whereby we maximize  $Z$  and integrate out  $W$ . We will use a prior of the form  $p(W) = \prod_j N(w_j | 0, I)$ . The corresponding likelihood becomes

$$\begin{aligned} p(Y|Z, \sigma^2) &= \prod_{d=1}^D N(y_d | 0, Z Z^T + \sigma^2 I) \quad (15.119) \\ &= (2\pi)^{-DN/2} |K_z|^{-D/2} \exp \left[ -\frac{1}{2} \text{tr}(K_z^{-1} Y T Y) \right] \quad (15.120) \end{aligned}$$

where  $K_z = Z Z^T + \sigma^2 I$ . Based on our discussion of the connection between the eigenvalues of  $Y T Y$  and of  $Y T Y$  in Section 14.4.4, it should come as no surprise that we can also solve the dual problem using eigenvalue methods (see (Lawrence 2005) for the details). If we use a linear kernel, we recover PCA. But we can also use a more general kernel:  $K_z = K + \sigma^2 I$ , where  $K$  is the Gram matrix for  $Z$ . The MLE for  $Z$  will no longer be available via eigenvalue methods; instead we must use gradient-based optimization. The objective is given by

$$I = -\frac{1}{2} \log |K_z| - \frac{1}{2} \text{tr}(K_z^{-1} Y T Y) \quad (15.121)$$

and the gradient is given by

$$\partial \partial Z_{ij} = \partial \partial K_z \partial K_z \partial Z_{ij} \quad (15.122)$$

where

$$\partial \partial K_z = K_z^{-1} Y T Y K_z^{-1} - D K_z^{-1} \quad (15.123)$$

The form of  $\partial K_z \partial Z_{ij}$  will of course depend on the kernel used. (For example, with a linear kernel, where  $K_z = ZZ^T + \sigma^2 I$ , we have  $\partial K_z \partial Z = Z$ .) We can then pass this gradient to any standard optimizer, such as conjugate gradient descent. Let us now compare GP-LVM to kernel PCA. In kPCA, we learn a kernelized mapping from the observed space to the latent space, whereas in GP-LVM, we learn a kernelized mapping from the latent space to the observed space. Figure 15.10 illustrates the results of applying kPCA and GP-LVM to visualize the 12 dimensional oil flow data shown in Figure 14.9(a). We see that the embedding produced by GP-LVM is far better. If we perform nearest neighbor classification in the latent space, GP-LVM makes 4 errors, while kernel PCA (with the same kernel but separately optimized hyper-parameters) makes 13 errors, and regular PCA makes 20 errors. GP-LVM inherits the usual advantages of probabilistic generative models, such as the ability to handle missing data and data of different types, the ability to use gradient-based methods (instead of grid search) to tune the kernel parameters, the ability to handle prior information, etc. For a discussion of some other probabilistic methods for (spectral) dimensionality reduction, see (Lawrence 2012).

### 15.6 Approximation methods for large datasets

The principal drawback of GPs is that they take  $O(N^3)$  time to use. This is because of the need to invert (or compute the Cholesky decomposition of) the  $N \times N$  kernel matrix  $K$ . A variety of approximation methods have been devised which take  $O(M^2N)$  time, where  $M$  is a user-specifiable parameter. For details, see (Quinonero-Candela et al. 2007).

下面先把原文完整翻译（含图注与公式），再用通俗解释讲清核心思想，最后给要点总结。

## 翻译（不遗漏）

**图 15.10** 12 维油流数据的二维表示。不同的颜色/符号表示油流的 3 个相。

- (a) 采用高斯核（RBF 核）的核 PCA。
  - (b) 采用高斯核的 GP-LVM。阴影表示后验的精度（precision），颜色越浅表示精度越高。
- 摘自 (Lawrence 2005) 图 1。

## 15.5 GP 潜变量模型 (GP-LVM)

在 14.4.4 节我们讨论了核 PCA，它把核技巧用在普通的 PCA 上。本节讨论把核与\*\*概率 PCA (PPCA) \*\*结合的另一种方式。得到的方法称为 GP-LVM (Gaussian process latent variable model, 高斯过程潜变量模型) (Lawrence 2005)。为说明该方法，我们从 PPCA 开始。回忆 12.2.4 节，PPCA 的模型为

$$p(z_i) = \mathcal{N}(z_i | 0, I) \quad (15.116)$$

$$p(y_i | z_i, \theta) = \mathcal{N}(y_i | Wz_i, \sigma^2 I) \quad (15.117)$$

我们可以通过极大似然来拟合该模型：积分掉  $z_i$  并对  $W$  (以及  $\sigma^2$ ) 做最大化。其目标函数为

$$p(Y | W, \sigma^2) = (2\pi)^{-DN/2} |C|^{-N/2} \exp(-\frac{1}{2} \text{tr}(C^{-1} Y^\top Y)), \quad (15.118)$$

其中  $C = WW^\top + \sigma^2 I$ 。正如定理 12.2.2 所示，该式的极大似然解可以用  $Y^\top Y$  的特征向量来计算。

现在我们考虑对偶问题：最大化  $Z$ ，并积分掉  $W$ 。对  $W$  使用先验  $p(W) = \prod_j \mathcal{N}(w_j | 0, I)$ 。对应的似然变为

$$p(Y | Z, \sigma^2) = \prod_{d=1}^D \mathcal{N}(y_{:,d} | 0, ZZ^\top + \sigma^2 I) \quad (15.119)$$

$$= (2\pi)^{-DN/2} |K_z|^{-D/2} \exp(-\frac{1}{2} \text{tr}(K_z^{-1} Y^\top Y)), \quad (15.120)$$

其中  $K_z = ZZ^\top + \sigma^2 I$ 。结合 14.4.4 节关于  $Y^\top Y$  与  $YY^\top$  特征值关系的讨论，这里我们也可以用特征值方法来解对偶问题 (细节见 Lawrence 2005)。如果使用线性核，我们能恢复 PCA。

但我们也可能使用更一般的核：令  $K_z = K + \sigma^2 I$ ，其中  $K$  是对  $Z$  做 Gram 矩阵得到的核矩阵。此时  $\hat{Z}$  的 MLE 不再能用特征值方法求出；必须用基于梯度的优化。目标函数为

$$\ell = -\frac{D}{2} \log |K_z| - \frac{1}{2} \text{tr}(K_z^{-1} Y^\top Y) \quad (15.121)$$

其梯度为

$$\frac{\partial \ell}{\partial Z_{ij}} = \frac{\partial \ell}{\partial K_z} \frac{\partial K_z}{\partial Z_{ij}}, \quad (15.122)$$

其中

$$\frac{\partial \ell}{\partial K_z} = K_z^{-1} Y^\top Y - D K_z^{-1}. \quad (15.123)$$

$\partial K_z / \partial Z_{ij}$  的具体形式显然取决于所用的核。（例如用线性核、且  $K_z = ZZ^T + \sigma^2 I$  时，有  $\partial K_z / \partial Z = Z$ 。）随后把这个梯度交给任意标准优化器，如共轭梯度法。

现在比较 GP-LVM 与 核 PCA：在 kPCA 中，我们学习的是从观测空间到潜在空间的核映射；而在 GP-LVM 中，我们学习的是从潜在空间到观测空间的核映射。图 15.10 展示了把 kPCA 与 GP-LVM 用于可视化 12 维油流数据（见图 14.9(a)）的结果。可以看到，GP-LVM 得到的嵌入要好得多。若在潜在空间里做最近邻分类，GP-LVM 的错误为 4 次；核 PCA（使用相同核但单独优化超参数）错误为 13 次；普通 PCA 错误为 20 次。

GP-LVM 继承了概率生成模型的常见优点：能处理缺失数据和不同类型的数据；可用基于梯度的方法（而非网格搜索）来调核的超参数；能纳入先验信息，等等。关于其他（谱）降维的概率方法，参见 (Lawrence 2012)。

---

## 15.6 大数据集的近似方法

GP 的主要缺点是计算量为  $O(N^3)$ 。这是因为需要对  $N \times N$  的核矩阵  $K$  求逆（或做 Cholesky 分解）。已经提出了多种近似方法，其时间复杂度为  $O(M^2N)$ ，其中  $M$  是用户可设的参数。详见 (Quinonero-Candela et al., 2007)。

---

## 通俗解释

### 图 15.10 在说什么？

- 我们有 12 维的“油流”数据，想在 2D 里可视化并把 3 个流动“相”分开。
- 核 PCA (a)：把“输入→潜在”的映射做成核化版本。
- GP-LVM (b)：反过来建一个生成模型，“潜在→观测”，并用 GP 来描述这个生成映射。
- 结果：GP-LVM 的二维图把三类分得更清楚；图中的浅色区域表示后验精度高（即不确定性小）。

### GP-LVM 是什么？

- 在 PPCA 中： $z_i$  是低维潜变量， $y_i$  是高维观测，线性关系  $y_i = Wz_i + \text{噪声}$ 。
- GP-LVM 把这个“线性投影  $W$ ”积分掉，转而只优化“潜在位置”  $Z = [z_1, \dots, z_N]$ ；同时把“线性关系”升级成非线性，用核（或直接用 GP）来表达“潜在 → 观测”的映射。
- 换句话说：我们让每一维观测  $y_{:,d}$  都来自一个以  $Z$  为输入的高斯过程，因此似然是  $N(0, K_z)$  的乘积（式 15.119–15.120）。
- 目标函数  $\ell$ （式 15.121）只有两个部分：
  1.  $-\frac{D}{2} \log |K_z|$ ：惩罚“复杂的” $K_z$ ；

2.  $-\frac{1}{2} \text{tr}(K_z^{-1} Y Y^\top)$ : 要求  $K_z$  能解释数据的相关结构。

- 用链式法则得到对  $Z$  的梯度 (式 15.122–15.123)，然后就能用常见的**梯度优化**求解 (比核 PCA 的“只调核参数、 $Z$  固定在特征向量上”更灵活)。

## 它为何通常比核 PCA 好?

- kPCA 只学“如何把数据**拉到潜在空间**”；
- GP-LVM 学的是“潜在空间**如何生成数据**”。作为生成模型，它能：
  - 给**不确定性** (图里阴影就是后验精度)；
  - 用**概率方式**处理**缺失与异质数据**；
  - 直接用**梯度**调核的超参数 (不必笨重的网格搜索)；
  - 可以把**先验** (比如几何/时间顺序) 注入到  $Z$  上。
- 实验上，对同一油流数据，kPCA 近邻分类错 13 次，GP-LVM 只错 4 次，PCA 错 20 次。

## 大数据时怎么加速 GP?

- 直接用 GP 需要  $O(N^3)$  (对  $K$  做 Cholesky)。
- 思路：用一个规模为  $M$  的**近似表示** (比如“诱导点/低秩近似”)，把计算降到  $O(M^2N)$ ； $M$  由你设定，平衡**速度与精度**。 (细节见 Quinonero-Candela 等 2007。)

## 要点总结

- GP-LVM = PPCA 的非线性概率版**：把“潜在  $\rightarrow$  观测”的映射从线性改为**GP/核**；最大化  $Z$ ，积分掉  $W$ 。
- 目标函数**： $\ell = -\frac{D}{2} \log |K_z| - \frac{1}{2} \text{tr}(K_z^{-1} Y Y^\top)$ ，对  $Z$  用梯度法优化。
- 与核 PCA 的区别**：kPCA 学“输入  $\rightarrow$  潜在”；GP-LVM 学“潜在  $\rightarrow$  观测”(生成式)，因此能给出**不确定性**、处理**缺失/异质数据**、用**梯度**调参，并常常得到更好的嵌入 (图 15.10)。
- 性能对比 (油流数据)**：PCA 20 错，kPCA 13 错，GP-LVM 4 错 (近邻分类)。
- 大数据近似**：原始 GP  $O(N^3) \rightarrow$  采用近似 (如诱导点/低秩) 到  $O(M^2N)$ ， $M$  由用户控制，用以折中速度与精度。

这一小节主要在讲什么？我不太懂

已思考 14s

这小节主要讲两件事：

## 一句话版

- 15.5: GP-LVM —— 用高斯过程把“潜在→观测”的关系建成一个**非线性、带不确定性的生成模型**, 做降维/可视化, 比核PCA更灵活、效果常更好。
  - 15.6: **大数据上的加速** —— 标准GP要  $O(N^3)$  太慢, 需用\*\*近似 (如诱导点/低秩) \*\*把复杂度降到  $O(M^2N)$ 。
- 

## 通俗解释

### 1) GP-LVM 是啥? 怎么来的?

- 从 PPCA 出发: 高维观测  $y_i$  由**低维潜变量**  $z_i$  线性生成  $y_i = Wz_i + \text{噪声}$ 。
- **GP-LVM 的想法:**
  1. **把线性  $W$  积分掉;**
  2. 把“线性生成”升级为**非线性生成**: 让每一维观测都是以  $Z = [z_i]$  为输入的**高斯过程**。  
于是观测的似然只通过**核矩阵**  $K_z = K(Z, Z) + \sigma^2 I$  表达。
- **要做的事:** 选择一组低维坐标  $Z$  让模型最能解释数据。优化的目标

$$\ell(Z) = -\frac{D}{2} \log |K_z| - \frac{1}{2} \text{tr}(K_z^{-1} Y Y^\top)$$

用链式法则求梯度, 对  $Z$  做**梯度优化**即可 (共轭梯度等)。

### 2) 和核PCA的区别

- **核PCA:** 学的是**观测→潜在**的核映射 (投影), 是判别式的。
- **GP-LVM:** 学的是**潜在→观测**的核映射 (生成式)。因此它:
  - 给**不确定性** (图里浅色=精度高) ;
  - 能处理**缺失/多类型数据**;
  - 核超参可用**梯度调** (不必网格搜索) ;
  - 可加入**先验约束** (顺序、几何等)。
- 在油流数据的二维可视化里: **GP-LVM 更好分类** (近邻错 4 次), 核PCA错 13 次, 普通PCA错 20 次。

### 3) 图 15.10 看什么

- (a) 核PCA 的嵌入; (b) GP-LVM 的嵌入; 浅色区域表示模型更自信。

- 直观理解：GP-LVM是在二维上“摆放”每个点  $z_i$ ，让一个平滑的GP能“画回”原来的高维数据；摆得好，类就分得清。

#### 4) 大数据时为什么要近似 (15.6)

- 用 GP 必须对  $N \times N$  的核矩阵做 Cholesky/求逆  $\Rightarrow O(N^3)$ 。
  - 近似思路：**用  $M$  个**诱导点/低秩表示**把计算改成  $O(M^2N)$  ( $M \ll N$ )，在速度与精度之间折中。
- 

### 要点总结

- GP-LVM = PPCA 的非线性、概率、生成式降维：**学习潜在坐标  $Z$  与核超参，使 GP 从低维能“生成”高维数据。
- 目标函数只依赖核矩阵  $K_z$ ，对  $Z$  可直接做梯度优化。**
- 优于核PCA：**能给不确定性、处理缺失/异质数据、可纳入先验，实际可视化/分类效果常更好。
- 计算瓶颈：**标准GP是  $O(N^3)$ ；用**诱导点/稀疏近似**降到  $O(M^2N)$ 。

基于本对话里我贴过的整章原文，输出：  
①知识点总结  
(分点+关键词+用途) ②子章节关联 (依赖/对比/递进+关系句，附Mermaid可选) ③本章总结。

已思考 14s

太棒了！我把本章（第15章：Gaussian Processes）按你要的格式提炼好了——

---

### ① 知识点总结 (分点 + 关键词 + 用途)

- GP 基本概念**
  - 关键词：先验/后验、核函数  $k(x,x')$ 、均值函数  $m(x)$ 、协方差  $K$ 、相似即相关
  - 用途：用“相似输入→相似输出”的规则，对未知函数整条曲线做**贝叶斯建模**，给出**预测均值 + 不确定性**。
- 无噪/含噪回归预测 (15.2.1–15.2.2)**
  - 关键词：插值 (无噪)、噪声方差  $\sigma_y^2$ 、后验均值  $k^{*T}(K+\sigma_y^2I)^{-1}y$ 、后验方差

- 用途：在任意新点给出**点预测与置信带**；作为复杂仿真的替代（DACE/仿真实验设计）。
- **核与超参数** (15.2.3)
  - 关键词：*SE/RBF 核、长度尺度  $\ell$ 、幅度  $\sigma_f$ 、ARD/各向异性、多维  $M$  矩阵*
  - 用途：控制函数**平滑度与变化方向**；做**特征相关性/自动相关性判别**。
- **超参数学习** (15.2.4)
  - 关键词：边际似然  $\log p(y|X)$ 、数据拟合 vs 复杂度、梯度、局部极小、贝叶斯超参 (网格/MC/CCD)
  - 用途：自动、可微地调参；用近似贝叶斯方法**表达不确定性并减轻过拟合**。
- **数值与计算** (15.2.5)
  - 关键词：*Cholesky 分解、避免求逆、共轭梯度、 $O(N^3)$ 、 $O(N^2)$  预测*
  - 用途：**稳定高效**实现 GP 训练/预测。
- **半参数 GP** (15.2.6)
  - 关键词：线性/多项式均值  $\beta^\top \varphi(x) + GP$  残差、把  $\beta$  积分掉
  - 用途：兼顾**可解释性与灵活拟合**（趋势+非线性余项）。
- **GP + GLM (分类/Poisson)** (15.3)
  - 关键词：*logistic/probit 链接、拉普拉斯/IRLS 高斯近似、W 对角、预测把  $f^*$  的高斯分布过链接函数积分*
  - 用途：**二分类/多分类/计数任务**，输出**良好校准的概率与不确定性**；空间疾病制图 (Poisson GP)。
- **与其他方法的关系** (15.4)
  - 关键词：
    - 贝叶斯线性回归  $\leftrightarrow$  线性核的 GP (退化核)
    - 线性平滑器/等价核、有效自由度  $\sum \lambda_i / (\lambda_i + \sigma^2)$
    - SVM (铰链损失, 稀疏, 但非概率)  $\leftrightarrow$  GP (概率、给不确定性)
    - RVM/L1VM：数据依赖、退化核，外推过自信
    - 无限宽神经网络  $\leftrightarrow$  GP (NN 核、RBF 网络  $\leftrightarrow$  SE 核)
    - 平滑样条  $\leftrightarrow$  某 GP 的 MAP；薄板样条 (2D)
    - RKHS 正则回归  $\leftrightarrow$  GP 后验均值 (表征定理)
  - 用途：统一视角理解方法差异，**选型与互换**。
- **GP-LVM 非线性降维** (15.5)
  - 关键词：*PPCA 的对偶、潜在→观测用 GP、最大化 Z、目标  $-(D/2)\log|Kz| - \frac{1}{2} \text{tr}(Kz^{-1}YY^\top)$*

- 用途：**生成式**降维/可视化（给不确定性、可处理缺失/异质数据、可加先验），常优于核PCA。
  - **大数据近似** (15.6)
    - 关键词： $O(N^3) \rightarrow$  诱导点/低秩近似  $O(M^2N)$ 、可调  $M$
    - 用途：把 GP 扩展到**大规模数据**，在**速度-精度**间折中。
- 

## ② 子章节关联 (依赖 / 对比 / 递进)

- 15.1 引言 → 15.2 回归：从“函数的分布”理念过渡到**闭式**的回归推断与预测公式。  
**(依赖)**
- 15.2.3–15.2.4：核的形状决定能力，**边际似然**学习其超参，存在**局部最优**；随后  
(15.2.5) 给出**稳定计算**。**(递进)**
- 15.2.6 半参数与15.2 回归：在回归的基础上把趋势显式分离，提高**可解释性**。**(扩展)**
- 15.3 GPs meet GLMs：把 15.2 的思路迁移到**非高斯似然**（分类/计数），用**拉普拉斯/IRLS**近似解决不共轭。**(迁移递进)**
- 15.4 关系：把前面方法与SVM/NN/RKHS/**样条/线性**模型放在同一坐标系里比较，强  
调 GP 的**概率与不确定性**优势、退化核风险。**(对比/统摄)**
- 15.5 GP-LVM：由 PPCA 出发引入**生成式非线性降维**，与核PCA对照：输入→潜在（判  
别） vs 潜在→观测（生成）。**(对比+升级)**
- 15.6 近似：针对全章的计算瓶颈提出**实用可扩展方案**。**(工程支撑)**

可选 Mermaid 关系图：

```

graph TD
    A[15.1 引言: 函数的分布/核先验] --> B[15.2 回归: 闭式预测 & 核学习]
    B --> B1[15.2.3 核与超参]
    B1 --> B2[15.2.4 边际似然学习/局部极小]
    B2 --> B3[15.2.5 数值稳定: Cholesky/CG]
    B3 --> B4[15.2.6 半参数GP]
    B4 --> C[15.3 GP + GLM: 分类/Poisson (拉普拉斯/IRLS)]
    A --> D[15.4 与其他方法关系: SVM/NN/RKHS/样条/线性]
    D -.-> E[15.5 GP-LVM: 生成式降维]
    E -.-> F[15.6 近似: 诱导点/低秩 O(M^2N)]
    C -.-> F
  
```

## ③ 本章总结 (一屏速记)

- **本质**：高斯过程把“未知函数”当作随机对象，用核函数编码“相似输入→相似输出”。在回归中给出**闭式**的预测均值与方差；在分类/计数中通过**拉普拉斯/IRLS**做近似，输出**校准概率**。

- **建模三板斧：**选核（形状/ARD）、学超参（边际似然 + 梯度，注意局部极小）、稳计算（Cholesky/CG； $O(N^3)$ ）。
- **能力边界：**标准 GP 计算随样本量立方增长；需采用**诱导点/低秩**等近似把复杂度降到  $O(M^2N)$ 。
- **方法地图：**
  - 线性回归、RKHS、样条、无限宽 NN 都可在 GP 视角下解释（有的为特例/极限）。
  - SVM/RVM 等判别式或稀疏方法更快，但通常不提供**良好不确定性**；GP 的强项正是**概率与置信度**。
- **扩展应用：**半参数 GP 兼顾可解释与灵活；GP-LVM 把 GP 用到**非线性降维/可视化**（生成式、可处理缺失/异质数据），往往优于核PCA。

一句话记忆：GP = 核 + 高斯 + 贝叶斯 → 给“函数”做概率推断，既能拟合又能量化不确定性；借助近似与工程技巧，既能做回归/分类，也能做降维与空间建模。

请用通俗易懂的话总结本章，使我能够理解

好～把第 15 章用“人话”快速讲明白：

## 这章到底在干嘛？

把未知函数当成“会抖动的一根橡皮筋”，用概率的方式来拟合它。

高斯过程（GP）做的事是：给出“在每个  $x$  处，这根橡皮筋大概在哪儿，以及它有多不确定”。

核心靠一个核函数（也叫协方差函数）：它把“输入像不像”变成“输出会不会相似”。像，就一起动；不像，就各走各的。

## 回归：先从最常见的曲线拟合说起

- **无噪声：**GP 会精确穿过训练点（插值），离数据远了，不确定性变大。
- **有噪声：**不用穿点，只要靠近；模型会自己权衡“跟数据贴多紧”与“整体要多平滑”。
- **你需要知道的三个旋钮**（超参数）
  - 长度尺度  $\ell$ ：控制“曲线多抖”。小=抖、大=平滑。
  - 幅度  $\sigma_f$ ：曲线上下摆动有多大。

- 噪声  $\sigma_y$ : 观测里掺了多少随机抖动。
  - 怎么自动调旋钮? 用“边际似然”: 既看拟合好不好, 又惩罚太复杂的曲线 (防过拟合)。它不是凸的, 会有局部最优, 要小心初始化。
  - 算得稳: 别直接求逆, Cholesky 分解最稳; 复杂度主因是要处理  $N \times N$  的核矩阵 (训练大约是  $O(N^3)$ )。
- 

## 分类 & 计数: 不只画曲线, 还要给概率

- 二分类/多分类: 把潜在函数丢进 sigmoid/softmax, 得到“属于某类的概率”。但这时“高斯先验 + 非高斯似然”不好直接解 → 用拉普拉斯近似/IRLS把后验“捏回”成高斯, 算得快还挺准。
  - Poisson (计数): 对“强度的对数”放 GP, 能做空间风险图: 哪里风险高、哪里不确定性大, 一目了然。
- 

## 它和别的常见方法啥关系?

- 贝叶斯线性回归: 相当于用了“**线性核**”的 GP, 是 GP 的一个特例 (但表达力有限, 容易过度自信)。
  - SVM: 优化“铰链损失+正则”, 稀疏、快; 但不是概率模型, 不给/不好给置信度。GP 的强项恰是**不确定性**。
  - RKHS 正则化/样条: 在函数空间里“惩罚不平滑”得到的解, 和 GP 的**预测均值**等价 (但 GP 还多给一个“方差”)。
  - 神经网络: 一层网络在宽度  $\rightarrow \infty$  时会收敛成 GP (不同激活函数对应不同“NN 核”)。RBF 网络  $\leftrightarrow$  RBF (SE) 核。
- 

## 非线性降维: GP-LVM

- 和核 PCA 相比: 核 PCA 学“**把数据拉到低维**”; GP-LVM 学“**低维怎么生成高维**”(**生成式**)。
  - 好处: 能给**不确定性**、能处理**缺失/异质数据**、核参数能用梯度调、还能塞先验。
  - 实验里 (油流数据) 可视化/分类更清晰、错误更少。
- 

## 规模化: 大数据怎么办?

标准 GP 要  $O(N^3)$  (太贵)。可用**诱导点/低秩近似**把计算降到  $O(M^2N)$  ( $M \ll N$ )，自己选  $M$  来权衡速度 vs 精度。

---

## 什么时候用 GP? 怎么落地?

- 你需要靠谱的不确定性 (置信带/概率) 时。
  - 数据量中等 (几千到几万以内) 或你能接受**诱导点近似**时。
  - 先选一个合理核 (RBF/ARD 是稳妥缺省)，**标准化**输入输出；
  - 用**边际似然**学超参，注意别卡在明显极端 ( $\ell$  特别小/特别大)；
  - 看**残差和不确定性**是否合情理：离数据远处，方差应该**变大** (这点比很多“只会给点预测”的方法靠谱)。
- 

## 一页速记

- 本质：**GP = 核 (相似性) + 高斯 (可计算) + 贝叶斯 (不确定性)。
- 能力：**回归有闭式；分类/计数用近似；还可做降维 (GP-LVM)。
- 优点：**概率与不确定性、小样本友好、能融入先验、解释性强。
- 缺点：**原始复杂度高；需用近似做大规模。
- 和别人比：**SVM/稀疏方法更快更硬，但**不给概率**；NN 很灵活，但要调；GP 在“需要置信度的中等规模问题”里**很香**。

**一句话：**把“拟合一条曲线/决策边界”升级为“给整条函数一个分布”，既能预测“在哪里”，也能说清“有多确定”。