2022-01-10

> 本文是对《PRML》中6、7两章内容的梳理，细节请参考原文

6 Kernel Methods

6.1 Dual Representations

推导了 `L2` 正则化的线性回归模型的 dual representation，从而引出了核方法。

相比 the original parameter space formulation, the advantage of the dual formulation, as we shall see, is that it is expressed entirely in terms of the kernel function $k(\mathrm{x}, \mathrm{x}')$. We can therefore work directly in terms of kernels and avoid the explicit introduction of the feature vector $\boldsymbol{\phi}(\mathrm{x})$, which allows us implicitly to use feature spaces of high, even infinite, dimensionality.

缺点是一般而言计算量会更大，the original parameter space formulation 在 M 维的参数空间中进行计算，而 the dual formulation 在 N 维的样本点空间中进行计算，而样本容量 N 一般大于基函数的个数 M。

6.2 Constructing Kernels

核函数的充要条件：A necessary and sufficient condition for a function $k(\mathrm{x}, \mathrm{x}')$ to be a valid kernel (Shawe-Taylor and Cristianini, 2004) is that the Gram matrix K, whose elements are given by $k(\mathrm{x}_n, \mathrm{x}_m)$, should be positive semidefinite for all possible choices of the set $\{\mathrm{x}_n\}$.

给出了核函数的相关性质，由此可以根据已知核函数构造新的合法的核函数。一般地，We require that the kernel $k(\mathrm{x}, \mathrm{x}')$ be symmetric and positive semidefinite and that it expresses the appropriate form of similarity between x and x' according to the intended application.

最后介绍了几个典型的核函数：

- polynomial kernel

  $k(\mathrm{x}, \mathrm{x}') = (\mathrm{x}^{\mathrm{T}}\mathrm{x}')^M$ ：contains all monomials of order M；更一般地，
  $k(\mathrm{x}, \mathrm{x}') = (\mathrm{x}^{\mathrm{T}}\mathrm{x}' + c)^M$, with $c > 0$：include all terms up to degree M
- Gaussian kernel

  $$\begin{aligned} k(\mathrm{x}, \mathrm{x}') &= \exp(-\|\mathrm{x} - \mathrm{x}'\|^2/2\sigma^2) \\ &= \exp\left\{-\frac{\mathrm{x}^{\mathrm{T}}\mathrm{x} + (\mathrm{x}')^{\mathrm{T}}\mathrm{x}' - 2\mathrm{x}^{\mathrm{T}}\mathrm{x}'}{2\sigma^2}\right\} \end{aligned}$$

  Note that the feature vector that corresponds to the Gaussian kernel has infinite dimensionality.

The Gaussian kernel is not restricted to the use of Euclidean distance. Replace $\mathrm{x}^{\mathrm{T}}\mathrm{x}'$ with a nonlinear kernel $\kappa(\mathrm{x},\mathrm{x}')$, we obtain

$$k(\mathrm{x},\mathrm{x}') = \exp\left\{-\frac{\kappa(\mathrm{x},\mathrm{x}) + \kappa(\mathrm{x}',\mathrm{x}') - 2\kappa(\mathrm{x},\mathrm{x}')}{2\sigma^2}\right\}$$

- An important contribution to arise from the kernel viewpoint has been the extension to inputs that are symbolic, rather than simply vectors of real numbers. Kernel functions can be defined over objects as diverse as graphs, sets, strings, and text documents. 也就是说，除了实数向量，核函数还可以定义在图、集合、字符串、文本等符号对象上。实际上也很容易理解，毕竟存在映射 $\boldsymbol{\phi}(\cdot)$ 可以将这些符号对象映射回实向量空间。

  一个例子：Consider a fixed set and define a nonvectorial space consisting of all possible subsets of this set. If A_1 and A_2 are two such subsets then one simplec hoice of kernel would be：

  $$k(A_1, A_2) = 2^{|A_1 \bigcap A_2|}$$

  where $A_1 \bigcap A_2$ denotes the intersection of sets A_1 and A_2, and $|A|$ denotes then umber of subsets in A. 上述定义在集合空间中的函数是一个合法的核函数，因为我们可以构造如下的特征空间 (也就是构造映射 $\boldsymbol{\phi}(\cdot)$)，在该特征空间中，上式可以表示为特征向量的内积：Since A may be equal to D (the subset relation was not defined to be strict), $\boldsymbol{\phi}(D)$ must be defined. This will map to a vector of $2|D|$ 1s, one for each possible subseto f D, including D itself as well as the empty set. For $A \subset D$, $\boldsymbol{\phi}(A)$ will have 1s in all positions that correspond to subsets of A and 0s in all other positions. Therefore, $\boldsymbol{\phi}(A_1)^{\mathrm{T}}\boldsymbol{\phi}(A_2)$ will count the number of subsets shared by A_1 and A_2. However, this can just as well be obtained by counting the number of elements in the intersection of A_1 and A_2, and then raising 2 to this number, which is exactly what (6.27) does.

- use a generative model to define a kernel:

  1. Given a generative model $p(\mathrm{x})$ we can define a kernel by $k(\mathrm{x},\mathrm{x}') = p(\mathrm{x})p(\mathrm{x}')$. we can interpret it as an inner product
     in the one-dimensional feature space defined by the mapping $p(\mathrm{x})$. It says that two inputs x and x' are similar if they both have high probabilities.

  2. 进一步，we can extend this class of kernels by considering sums over products of different probability distributions, with positive weighting coefficients $p(i)$, of the form $k(\mathrm{x},\mathrm{x}') = \sum_i p(\mathrm{x}|i)p(\mathrm{x}'|i)p(i)$. Two inputs x and x' will give a large value for the kernel function, and hence appear similar, if they have significant probability under a range of different components.

     Taking the limit of an infinite sum, we can also consider kernels of the form

     $$k(\mathrm{x},\mathrm{x}') = \int p(\mathrm{x}|\mathrm{z})p(\mathrm{x}'|\mathrm{z})p(\mathrm{z})\mathrm{d}\mathrm{z}$$

     where z is a continuous latent variable.

- An alternative technique for using generative models to define kernel functions is known as the Fisher kernel. Consider a parametric generative model $p(\mathrm{x}|\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ denotes the vector of parameters. The goal is to find a kernel that measures the similarity of two input vectors x and x' induced by the generative model.

  the Fisher Score: $\boldsymbol{g}(\boldsymbol{\theta},\mathrm{x}) = \grad_{\boldsymbol{\theta}} \ln p(\mathrm{x}|\boldsymbol{\theta})$; the Fisher information matrix: $\mathrm{F} = \mathbb{E}_{\mathrm{x}}[\boldsymbol{g}(\boldsymbol{\theta},\mathrm{x})\boldsymbol{g}(\boldsymbol{\theta},\mathrm{x})^{\mathrm{T}}]$. Fisher kernel is defined by:

  $$k(\mathrm{x},\mathrm{x}') = \boldsymbol{g}(\boldsymbol{\theta},\mathrm{x})^{\mathrm{T}}\mathrm{F}^{-1}\boldsymbol{g}(\boldsymbol{\theta},\mathrm{x}')$$

Note that the presence of the Fisher information matrix causes this kernel to be invariant under a nonlinear re-parameterization of the density model $\theta \to \psi(\theta)$, where the function $\psi(\cdot)$ is invertible and differentiable.

In practice, it is often infeasible to evaluate the Fisher information matrix. One approach is simply to replace the expectation in the definition of the Fisher information with the sample average, giving

$$\mathrm{F} \approx \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{g}(\theta, \mathrm{x}_n) \boldsymbol{g}(\theta, \mathrm{x}_n)^{\mathrm{T}}$$

This is the covariance matrix of the Fisher scores, and so the Fisher kernel corresponds to a whitening of these scores.

More simply, we can just omit the Fisher information matrix altogether and use the noninvariant kernel:

$$k(\mathrm{x}, \mathrm{x}') = \boldsymbol{g}(\theta, \mathrm{x})^{\mathrm{T}} \boldsymbol{g}(\theta, \mathrm{x}')$$

- the sigmoidal kernel: $k(\mathrm{x}, \mathrm{x}') = \tanh(a\mathrm{x}^{\mathrm{T}}\mathrm{x}' + b)$, whose Gram matrix in general is not positive semidefinite. This form of kernel has, however, been used in practice (Vapnik, 1995), possibly because it gives kernel expansions such as the support vector machine a superficial resemblance to neural network models. As we shall see, in the limit of an infinite number of basis functions, a Bayesian neural network with an appropriate prior reduces to a Gaussian process, thereby providing a deeper link between neural networks and kernel methods.

6.3 Radial Basis Function Networks

线性回归模型的基函数选择径向基函数 (radial basis functions) 时的情况，which have the property that each basis function depends only on the radial distance (typically Euclidean) from a centre $\boldsymbol{\mu}_j$, so that $\phi_j(\mathrm{x}) = h(\|\mathrm{x} - \boldsymbol{\mu}_j\|)$。

首先介绍了径向基函数的几种起源：

1. Historically, radial basis functions were introduced for the purpose of exact function interpolation.
2. Expansions in radial basis functions also arise from regularization theory.
3. Another motivation for radial basis functions comes from a consideration of the interpolation problem when the input (rather than the target) variables are noisy.
4. Another situation in which expansions in normalized radial basis functions arise is in the application of kernel density estimation to the problem of regression, as we shall discuss in Section 6.3.1.

Typically, the number of basis functions, and the locations $\boldsymbol{\mu}_j$ of their centres, are determined based on the input data $\{\mathrm{x}_n\}$ alone. The basis functions are then kept fixed and the coefficients w_i are determined by least squares by solving the usual set of linear equations, as discussed in Section 3.1.1. 接着，介绍了三种基函数中心点的选取方法：random、orthogonal least squares、Clustering algorithms such as K-means。

6.3.1 Nadaraya-Watson model

第三章介绍 Bayesian Linear Regression 时，通过对结果的整理引入核方法，得到了核回归。这里的 Nadaraya-Watson model 则从另外一个角度推导出了类似形式的核回归，We can motivate the kernel regression model (3.61) from a different perspective, starting with kernel density estimation (Section 2.5.1).

Suppose we have a training set $\{\mathrm{x}_n, t_n\}$ and we use a Parzen density estimator to model the joint distribution $p(\mathrm{x}, t)$, so that

$$p(\mathrm{x}, t) = \frac{1}{N} \sum_{n=1}^{N} f(\mathrm{x} - \mathrm{x}_n, t - t_n)$$

where $f(\mathrm{x}, t)$ is the component density function, and there is one such component centred on each data point. We now assume for simplicity that the component density functions have zero mean so that

$$\int_{-\infty}^{+\infty} f(\mathrm{x}, t) t \mathrm{d}t = 0$$

for all values of x. 可以看到，概率模型的构建直接基于样本集$\{\mathrm{x}_n, t_n\}$ (kernel density estimation, 类似的无参数模型还有K-nearest-neighbour density estimator)，而没有引入参数 w.

接下来的处理就很简单了，基于构建出来的概率模型，计算回归方程即可:

$$y(\mathrm{x}) = \mathbb{E}[t|\mathrm{x}]$$

由此，便可得到 Nadaraya-Watson model, or kernel regression:

$$\begin{gathered} y(\mathrm{x}) = \sum_n k(\mathrm{x}, \mathrm{x}_n) t_n \\ \text{with} \\ k(\mathrm{x}, \mathrm{x}_n) = \frac{g(\mathrm{x} - \mathrm{x}_n)}{\sum_m g(\mathrm{x} - \mathrm{x}_m)} \\ g(\mathrm{x}) = \int_{-\infty}^{+\infty} f(\mathrm{x}, t) \mathrm{d}t \end{gathered}$$

此外，也可以计算条件分布$p(t|\mathrm{x})$.

可以看到，Bayesian Linear Regression 和 Nadaraya-Watson model 对概率分布建模时选择了不同的路径，前者基于参数空间，后者基于样本空间。

6.4 Gaussian Processes

In Section 6.1, we introduced kernels by applying the concept of duality to a nonprobabilistic model for regression. Here we extend the role of kernels to probabilistic discriminative models, leading to the framework of Gaussian processes. We shall thereby see how kernels arise naturally in a Bayesian setting. 相比 Section 6.1 通过非概率回归模型的对偶表示引入核函数，这里我们将基于概率判别模型，在贝叶斯的框架下自然而然地引入核函数，并由此得到高斯过程。

都是基于贝叶斯理论，这里与第三章贝叶斯线性回归处理上的不同：

- In Chapter 3, we considered linear regression models of the form $y(\mathrm{x}, \mathrm{w}) = \mathrm{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathrm{x})$ in which w is a vector of parameters and $\boldsymbol{\phi}(\mathrm{x})$ is a vector of fixed nonlinear basis functions that depend on the input vector x. We showed that a prior distribution over w induced a corresponding prior distribution over functions $y(\mathrm{x}, \mathrm{w})$. Given a training data set, we then evaluated the posterior distribution over w and thereby obtained the corresponding posterior distribution over regression functions, which in turn (with the addition of noise) implies a predictive distribution $p(t|\mathrm{x})$ for new input vectors x.

- In the Gaussian process viewpoint, we dispense with the parametric model and instead define a prior probability distribution over functions directly. 也就是说，我们不在**参数空间**中进行计算，而直接在**函数空间**中进行计算。
  - At first sight, it might seem difficult to work with a distribution over the uncountably infinite space of functions. However, as we shall see, for a finite training set we only need to consider the values of the function at the discrete set of input values x_n corresponding to the training set and test set data points, and so in practice we can work in a finite space.

Gaussian Process的定义：In general, a Gaussian process is defined as a probability distribution over functions $y(\mathrm{x})$ such that the set of values of $y(\mathrm{x})$ evaluated at an arbitrary set of points $\mathrm{x}_1, \cdots, \mathrm{x}_N$ jointly have a Gaussian distribution.

- In cases where the input vector x is two dimensional, this may also be known as a Gaussian random field.
- More generally, a stochastic process $y(\mathrm{x})$ is specified by giving the joint probability distribution for any finite set of values $y(\mathrm{x}_1), \cdots, y(\mathrm{x}_N)$ in a consistent manner.

也就是说，随机过程是定义在函数上的概率分布：x 的每一种取值，都对应一个随机变量$y(\mathrm{x})$，因此存在无数个随机变量，随机过程研究的是它们的联合概率分布。以高斯过程为例，记 $\mathbf{y}_N = \langle y(\mathrm{x}_1), \cdots, y(\mathrm{x}_N)\rangle^{\mathrm{T}} = \langle y_1, \cdots, y_N\rangle^{\mathrm{T}}$，则 $\forall \mathbf{y}_N$ (包括任意的数据点个数 N)，$\mathbf{y}_N$ 均服从高斯分布。

首先，**6.4.1 Linear regression revisited** 小节给出了高斯分布的一个例子：

$$y(\mathrm{x}) = \mathrm{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathrm{x})$$
$$p(\mathrm{w}) = \mathcal{N}(\mathrm{w}|\mathbf{0}, \alpha^{-1}\mathrm{I})$$

注意，虽然该小节名为 Linear regression revisited，但上述模型少了noise项，并不具有regression功能。The probability distribution over w induces a probability distribution (Gaussian) over functions $y(\mathrm{x})$。进一步，可得 $\mathbf{y}_N$ 的概率分布 (conditioned on $\mathrm{x}_1, \cdots, \mathrm{x}_N$，为简便起见，我们不显示地写出这些条件)

$$\begin{aligned} p(\mathbf{y}_N) &= \mathcal{N}(\mathbf{y}_N|\mathbf{0}, \frac{1}{\alpha}\Phi\Phi^{\mathrm{T}}) \\ &= \mathcal{N}(\mathbf{y}_N|\mathbf{0}, \mathrm{K}) \end{aligned}$$

为高斯分布，其中，Φ 为 design matrix，$\Phi_{nk} = \phi_k(\mathrm{x}_n)$；可见，$y(\mathrm{x})$ 是一个高斯过程。此外，注意到此高斯过程的协方差矩阵为 Gram matrix (记为 K，相应的核函数为 $k(\mathrm{x}_n, \mathrm{x}_m) = \frac{1}{\alpha}\boldsymbol{\phi}(\mathrm{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathrm{x}_{\mathrm{m}})$)。由此，便自然而然地引入了核函数，且该高斯过程由核函数唯一确定。

关于核函数的选取：The kernel function that determines K is typically chosen to express the property that, for points x_n and x_m that are similar, the corresponding values $y(\mathrm{x}_n)$ and $y(\mathrm{x}_m)$ will be more strongly correlated than for dissimilar points. Here the notion of similarity will depend on the application.

> 随机过程的采样：

> 对随机过程的一次采样得到的应该是一个函数，但我们无法得到无穷个随机变量的联合概率分布，因此无法得到该函数的解析表达式，而只能选择布有限的 N 个点 $\mathrm{x}_1, \cdots, \mathrm{x}_N$，计算相应的 N 个随机变量 $\mathbf{y} = \langle y_1, \cdots, y_N \rangle^{\mathrm{T}}$ 的联合概率分布 $p(\mathbf{y})$，对其进行采样。由此，便可根据采样的数据 $\{\mathrm{x}_n, y_n\}_{n=1}^N$ 画出函数图像。
>
> 此外，在一次采样中，若已采样 N 组数据 $\{\mathrm{x}_n, y_n\}_{n=1}^N$，并想进一步对 x_{N+1} 处的随机变量 y_{N+1} 进行采样，这是一个预测问题，需计算关于 y_{N+1} 的后验概率，再对其采样。

接着，**6.4.2 Gaussian processes for regression** 小节在 6.4.1 小节模型的基础上添加了 noise 项 $t_n = y_n + \epsilon_n$，where ϵ_n is a random noise variable whose value is chosen independently for each observation n, $\epsilon \sim \mathcal{N}(0, \beta^{-1})$.

前面我们说模型 y 无法用于regression，但此时模型 t 可用于regression，且它也是一个高斯过程，我们称其为高斯过程回归 (GPR)。自然地，我们需要求解该高斯过程的概率分布，并在此基础上解决回归问题，即推导用于预测的条件分布：

1. 该高斯过程的概率分布：In order to find the marginal distribution $p(\mathbf{t}_N)$, conditioned on the input values $\mathrm{x}_1, \cdots, \mathrm{x}_N$, we need to integrate over $\mathbf{y}_N$.

$$\begin{aligned} p(\mathbf{t}_N) &= \int p(\mathbf{t}_N|\mathbf{y}_N)p(\mathbf{y}_N)\mathrm{d}\mathbf{y}_N \\ &= \mathcal{N}(\mathbf{t}_N|\mathbf{0}, \mathrm{C}) \end{aligned}$$

   where the covariance matrix C has elements

$$C(\mathrm{x}_n, \mathrm{x}_m) = k(\mathrm{x}_n, \mathrm{x}_m) + \beta^{-1}\delta_{nm}$$

   可见，该高斯过程由协方差矩阵 C 唯一确定。

2. Our goal in regression, however, is to make predictions of the target variables for new inputs, given a set of training data.

   Let us suppose that $\mathbf{t}_N = (t_1, \cdots, t_N)^{\mathrm{T}}$, corresponding to input values $\mathrm{x}_1, \cdots, \mathrm{x}_N$, comprise the observed training set, and our goal is to predict the target variable t_{N+1} for a new input vector x_{N+1}. This requires that we evaluate the predictive distribution $p(t_{N+1}|\mathbf{t}_N)$. Note that this distribution is conditioned also on the variables $\mathrm{x}_1, \cdots, \mathrm{x}_N$ and x_{N+1}. However, to keep the notation simple we will not show these conditioning variables explicitly.

   通过计算可得 $p(t_{N+1}|\mathbf{t}_N)$ 为高斯分布，其均值和方差为：

$$\begin{aligned} m(\mathrm{x}_{N+1}) &= \mathrm{k}^{\mathrm{T}}\mathrm{C}_N^{-1}\mathbf{t}_N \\ \sigma^2(\mathrm{x}_{N+1}) &= c - \mathrm{k}^{\mathrm{T}}\mathrm{C}_N^{-1}\mathrm{k} \end{aligned}$$

   其中，$\mathrm{C_N}$ 为前N个样本点对应的协方差矩阵，the vector k has elements $k(\mathrm{x}_n, \mathrm{x}_{N+1})$ for $n = 1, \cdots, N$, and the scalar $c = k(\mathrm{x}_{N+1}, \mathrm{x}_{N+1}) + \beta^{-1}$. we see that the predictive distribution is a Gaussian whose mean and variance both depend on x_{N+1}.

高斯过程回归(GPR)与贝叶斯线性回归(BLR)的比较：

- GPR 在函数空间中计算 (By working directly with the covariance function we have implicitly marginalized over the distribution of weights)，BLR在参数空间中计算：If the number M of basis functions is smaller than the number N of data points, it will be computationally more efficient to work in the basis function framework;
- However, an advantage of a Gaussian processes viewpoint is that we can consider covariance functions that can only be expressed in terms of an infinite number of basis functions.

6.4.3 Learning the hyperparameters 和 **6.4.4 Automatic relevance determination** 小节则在 6.4.2 小节的基础上进一步讨论超参数 (包括核函数中的超参数和 β 两部分，不妨将所有超参数统记为θ) 的设置与优化问题。

由 6.4.2 小节，高斯过程回归由协方差矩阵 C 唯一确定，而 C 又依赖于核函数 $k(\mathrm{x}_n, \mathrm{x}_m)$ 和 β 的选取：

- 与 BLR 对应的核函数为：

$$k(\mathrm{x}_n, \mathrm{x}_m) = \frac{1}{\alpha}\boldsymbol{\phi}(\mathrm{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathrm{x_m})$$

  此时超参数为 α 和 β。和 BLR 章节中介绍的那样，对于超参数的设置，可以引入它们的先验，采用 fully Bayesian treatment；也可以简单一点，采用 type 2 maximum likelihood procedure，即 make a point estimate of θ by maximizing the model evidence $p(\mathbf{t}_N|\theta)$。事实上，model evidence 也 conditioned on 基函数向量 $\boldsymbol{\phi}(\mathrm{x}_n)$，即基函数也是一种超参数，只不过它无法自动优化，只能提前给定，为了简便起见，这里不显示地在条件中写出。

  我们选择采用最大化 model evidence 方式。可以看到，与 BLR 中计算 model evidence 需要积掉基函数系数 w 不同，即 $p(\mathbf{t}_N|\theta) = \int p(\mathbf{t}_N, \mathrm{w}|\theta)\mathrm{dw} = \int p(\mathbf{t}_N|\mathrm{w})p(\mathrm{w}|\theta)\mathrm{d}\theta$，GPR 的 model evidence 可由高斯过程的概率分布直接给出，即 $p(\mathbf{t}_N|\theta) = \mathcal{N}(\mathbf{t}_N|\mathbf{0}, \mathrm{C})$，这是因为 By working directly with the covariance function we have implicitly marginalized over the distribution of weights。接着，采用梯度下降法，计算梯度、优化即可。

- 进一步，既然使用了核方法，那么核函数的选取就不一定要拘泥于与 BLR 对应的 $\frac{1}{\alpha}\boldsymbol{\phi}(\mathrm{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathrm{x_m})$ 这种形式。虽然核函数都可以写成特征向量内积的形式 $k(\mathrm{x}_n, \mathrm{x}_n) = \boldsymbol{\psi}(\mathrm{x}_n)^{\mathrm{T}}\boldsymbol{\psi}(\mathrm{x_m})$，但特征向量 $\boldsymbol{\psi}(\mathrm{x})$ 可以是无穷维的；此外，特征映射 $\boldsymbol{\psi}(\cdot)$ 还可以是带参数的函数，而不一定要是某个确定函数，比如 exponential-quadratic kernel 式 (6.63)：

$$k(\mathrm{x}_n, \mathrm{x}_m) = \theta_0 \exp\left\{-\frac{\theta_1}{2}\|\mathrm{x}_n - \mathrm{x}_m\|^2\right\} + \theta_2 + \theta_3 \mathrm{x}_n^{\mathrm{T}}\mathrm{x}_m$$

  将其与 BLR 中的情况相对应：

$$k(\mathrm{x}_n, \mathrm{x}_m) = \frac{1}{\alpha}\left(\alpha\theta_0 \exp\left\{-\frac{\theta_1}{2}\|\mathrm{x}_n - \mathrm{x}_m\|^2\right\} + \alpha\theta_2 + \alpha\theta_3 \mathrm{x}_n^{\mathrm{T}}\mathrm{x}_m\right)$$

  可以看到，在 BLR 的角度，核函数中含有 $\exp\left\{-\frac{\theta_1}{2}\|\mathrm{x}_n - \mathrm{x}_m\|^2\right\}$ 项，这意味着基函数向量 $\boldsymbol{\phi}(\mathrm{x})$ 是无穷维的；而 $\theta_0, \theta_1, \theta_3$ 等超参数的存在意味着基函数向量 $\boldsymbol{\phi}(\mathrm{x})$ 是一个参数模型。那么，基于上述核函数，通过 max model evidence 优化超参数 θ (核函数中的超参数 + β)，在一定程度上就实现了对基函数向量 $\boldsymbol{\phi}(\mathrm{x})$ 的学习。因此，相比 BLR，GPR 的优势在于基函数可以无穷维 + 基函数可以实现某种程度上的自动学习两点。

  > 类比多层神经网络，最后一层隐藏层的输出可以视为基函数，该隐藏层与输出层间的权重则为基函数的系数。可以看到，DNN 的基函数并不是固定的，也含有参数，可通过学习得到，这一学习最优基函数的过程就是表示学习。

- 6.4.4 小节介绍可以通过选择特定形式的核函数，通过像上面那样优化核函数的超参数，输入向量 x 中那些与拟合不太相关的输入分量可被剔除出去，从而实现稀疏解的效果，这类方法也被称为 automatic relevance determination (ARD)。为了实现上述目标，ARD 要求核函数 incorporate a separate parameter for each input variable，比如 Consider a Gaussian process with a two-dimensional input space $\mathrm{x} = (x_1, x_2)$, having a kernel function of the form：

$$k(\mathrm{x}, \mathrm{x}') = \theta_0 \exp\left\{-\frac{1}{2}\sum_{i=1}^{2}\eta_i(x_i - x_i')^2\right\}$$

其中，超参数 η_i 就可表征输入分量与回归目标的相关性。若通过超参数的优化，某一维度的 η 较小或为 0 则表示该输入分量与问题不太相关，从而可以剔除该输入分量。再比如，7.2节介绍的 Relevance Vector Machines，只不过7.2节没有采用高斯过程的形式，而是采用参数模型的形式介绍的。

我们还可以对上一小点介绍的 exponential-quadratic kernel 进行类似的修改：

$$k(\mathrm{x}_n, \mathrm{x}_m) = \theta_0 \exp\left\{-\frac{1}{2}\sum_{i=1}^{D}\eta_i(x_{ni} - x'_{mi})^2\right\} + \theta_2 + \theta_3\sum_{i=1}^{D}x_{ni}x_{mi}$$

where D is the dimensionality of the input space.

6.4.5 Gaussian processes for classification 和 **6.4.6 Laplace approximation** 小节讨论的是 Gaussian process 用于 classification 的问题，其思路和regression时类似，不同之处在于高斯过程无法直接用于分类问题的建模：In a probabilistic approach to classification, our goal is to model the posterior probabilities of the target variable for a new input vector, given a set of training data. These probabilities must lie in the interval (0, 1), whereas a Gaussian process model makes predictions that lie on the entire real axis. 为此，我们可以 define a Gaussian process over a function $a(\mathrm{x})$ and then transform the function using a logistic sigmoid (用于二分类问题) or softmax activation function (用于多分类问题).

构建完 GPC 模型后，和 regression 类似，我们还需要解决预测分布的推导，以及超参数的优化这两个问题。文中以二分类问题为例，对上述内容的相关公式进行了推导。而无论是预测分布，还是超参数优化时的 model evidence $\ln p(\mathbf{t}_N|\theta)$，它们均无法解析地计算，需使用近似算法。这里我们选择采用 Lapalace approximation，这就是 6.4.6 小节中的内容。

> Lapalace approximation 假设概率分布 $p(\mathrm{x}|\theta)$ 能用一个高斯分布近似，近似高斯分布的均值和协方差矩阵并不是取原分布的均值和协方差矩阵，而是取原分布的 MLE x^* 为均值，原分布对数的 Hessian matirx 在 MLE 处的值再取负号作为协方差矩阵，即 $\left[-\text{\grad}_{\mathrm{x}}^2 \ln p(\mathrm{x}|\theta)\right]_{\mathrm{x}=\mathrm{x}^*}$。可以看到，实际上就是在对 $\ln p(\mathrm{x}|\theta)$ 在 MLE 处作二阶泰勒近似。

最后一小节 **6.4.7 Connection to neural networks** 并不是在讨论 GP 与神经网络的联系，比如它们之间的相同点、不同点等，而是在讨论一个贝叶斯神经网络是否能成为一个高斯过程，即 In a Bayesian neural network, the prior distribution over the parameter vector w, in conjunction with the network function $f(\mathrm{x}, \mathrm{w})$, produces a prior distribution over functions from $y(\mathrm{x})$，那么 $y(\mathrm{x})$ 是否能成为一个高斯过程呢？答案是肯定的。研究表明，for a broad class of prior distributions over w, the distribution of functions generated by a neural network will tend to a Gaussian process in the limit $M \to \infty$，其中 M 表示隐藏单元的数量。

接着，和前面的思路类似，本小节又讨论了基于神经网络的高斯过程的核函数的特点，核函数超参数的学习等。

---

7 Sparse Kernel Machines

In the previous chapter, we explored a variety of learning algorithms based on nonlinear kernels. One of the significant limitations of many such algorithms is that the kernel function $k(\mathrm{x}_n, \mathrm{x}_m)$ must be evaluated for all possible pairs x_n and x_m of training points, which can be computationally infeasible during training and can lead to excessive computation times when making predictions for new data points.
In this chapter we shall look at kernel-based algorithms that have sparse solutions, so that

predictions for new inputs depend only on the kernel function evaluated at a subset of the training data points.

本章介绍了 support vector machine (SVM) 和 relevance vector machine (RVM) 两个模型。The SVM is a decision machine and so does not provide posterior probabilities, whereas the RVM is based on a Bayesian formulation and provides posterior probabilistic outputs, as well as having typically much sparser solutions than the SVM.

7.1 Maximum Margin Classifiers

实际上就是在讲支持向量机，不再赘述。

7.2 Relevance Vector Machines

SVMs 的问题：

1. the outputs of an SVM represent decisions rather than posterior probabilities.
2. Also, the SVM was originally formulated for two classes, and the extension to $K > 2$ classes is problematic.
3. There is a complexity parameter C, or ν (as well as a parameter ϵ in the case of regression), that must be found using a hold-out method such as cross-validation.
4. Finally, predictions are expressed as linear combinations of kernel functions that are centred on training data points and that are required to be positive definite (合法的核函数只要求是半正定的，而 SVM 要求使用正定的核函数；另外，说核函数半正定是指对应的Gram matrix半正定，核函数本身的输出是可正可负的).

The relevance vector machine or RVM is a Bayesian sparse kernel technique for regression and classification that shares many of the characteristics of the SVM whilst avoiding its principal limitations. Additionally, it typically leads to much sparser models resulting in correspondingly faster performance on test data whilst maintaining comparable generalization error.

这里为了便于区分，引入**广义相关向量机**和**狭义相关向量机。**

广义相关向量机的模型如下：

$$\begin{gathered}p(t|\mathrm{x},\mathrm{w},\beta)=\mathcal{N}(t|y(\mathrm{x}),\beta^{-1})\\ \text{with}\\ y(\mathrm{x})=\sum_{i=1}^{M}w_i\phi_i(\mathrm{x})=\mathrm{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathrm{x})\\ p(\mathrm{w}|\boldsymbol{\alpha})=\prod_{i=1}^{M}\mathcal{N}(w_i|0,\alpha_i^{-1})\end{gathered}$$

可以看到，相比第三章介绍的贝叶斯线性回归，区别在于参数w的先验分布采用了上述特殊的形式 (上述先验也称 ARD prior)，the key difference in the RVM is that we introduce a separate hyperparameter α_i for each of the weight parameters w_i instead of a single shared hyperparameter. 由此，通过优化超参数 $\boldsymbol{\alpha},\beta$，最终会学到 sparse model，即 We shall see that, when we maximize the evidence with respec to these hyperparameters, a significant proportion of them go to infinity (下文会推导这一情况发生的条件，即 $q_i^2 < s_i$), and the corresponding weight parameters have posterior distributions that are concentrated at zero. The basis functions associated with these parameters therefore play no role in the predictions made by the model and so are effectively pruned out, resulting in a sparse model.

可以看到，广义RVM就是高斯过程6.4.4 Automatic relevance determination的一个例子，只不过这里并没有采用高斯过程的形式，而是采用参数模型的形式介绍的。相应地，广义RVM的高斯过程的协方差矩阵为：

$$\begin{gathered}\mathrm{C} = \beta^{-1}\mathrm{I} + \mathrm{K} \\ \text{with} \\ \mathrm{K} = \Phi \mathrm{A}^{-1} \Phi^{\mathrm{T}} \\ \mathrm{A} = \mathrm{diag}(\alpha_i)\end{gathered}$$

可见，与贝叶斯线性回归的唯一区别是矩阵 A 由一个标量 α 代替。

类似地，**7.2.1 RVM for regression** 小节还介绍了超参数的优化和预测分布的计算。为了优化超参数，需要求参数 w 的后验分布 (是一个高斯分布)，因此，参数 w 的学习问题也顺道解决了；但在做预测时，我们可以进行fully Bayesian treatment，即不采用代入w 的最大后验估计到模型，再使用模型进行预测的方式，而是直接计算预测分布，即$p(t|\mathrm{x}, \mathrm{X}, \mathbf{t}, \boldsymbol{\alpha}^*, \beta^*) = \int p(t|\mathrm{x}, \mathrm{w}, \beta^*) p(\mathrm{w}|\mathrm{X}, \mathbf{t}, \boldsymbol{\alpha}^*, \beta^*) \mathrm{dw}$，此积分为两个高斯分布的积分，可得解析解 (也是一个高斯分布)，$\boldsymbol{\alpha}^*, \beta^*$ 为经过超参数优化找到的最优超参数。超参数的优化 (即 max model evidence) 无法推得解的显示表达式，需借助迭代算法，其中，α_i 的迭代公式形如：$\alpha_i \leftarrow f(\boldsymbol{\alpha}, \beta)$，即式(7.87)。而 **7.2.2 Analysis of sparsity** 小节则基于如下想法对迭代公式进行了优化：We first note that, in the result (7.87) for re-estimating the parameter α_i, the terms on the right-hand side are themselves also functions of α_i. These results therefore represent implicit solutions, and iteration would be required even to determine a single α_i with all other α_j for $j \neq i$ fixed；因此，我们可以尝试 make explicit all of the dependence of the marginal likelihood (7.85) (也就是model evidence) on a particular α_i and then determine its stationary points explicitly. 由此，我们可得如下的改进后的迭代公式：

$$\alpha_i = \begin{cases} \infty, & q_i^2 < s_i \\ \frac{s_i^2}{q_i^2 - s_i}, & q_i^2 > s_i \end{cases}$$

其中，q_i 被称为相应基向量的quality，s_i 被称为sparsity，它们的计算公式中均不再含有 α_i。可以看到，the relative size of the quality and sparsity terms determines whether a particular basis vector will be pruned from the model or not：若 $q_i^2 < s_i$，则 $\alpha_i = \infty$，如前文所说，相应的基函数会被pruned out。由此，便从理论上解释了稀疏性的由来。基于上述迭代公式，7.2.2 小节最后还给出了比7.2.1小节更高效的超参数迭代优化算法。

7.2.3 RVM for classification 小节讨论的则是分类问题。类似地，它和第四章介绍的贝叶斯线性分类的唯一区别在于参数 w 取 ARD prior。和回归中的思路一样，本小节讨论了超参数的优化问题，即max model evidence。此时，model evidence计算公式中的积分无法像回归中那样得到解析解，需借助Laplace approximation进行近似计算；而在计算model evidence的过程中需要计算 w 的最大后验估计(实际上是对 w 的后验分布作了Laplace approximation，将其近似为均值为最大后验估计的高斯分布)，因此对参数的学习也就顺带完成了。然而，这里并没有像回归中那样进一步推导预测分布。预测分布虽然也可以求，但比较麻烦，需作近似。对于预测问题更简单的做法是直接使用 w 的点估计，即代入 w 的最大后验估计到模型中，再使用模型进行预测。此外，类似回归中的情形，通过改进超参数优化的迭代计算公式，我们也可以得到稀疏性的理论解释和更高效的超参数迭代优化算法。最后，RVM 对多分类的推广也是直接的，不像 SVM 那样不容易扩展到多分类问题中。

狭义的 RVM 则限制 $y(\mathrm{x})$ 为如下的形式 (SVM-like form)：

$$y(\mathrm{x}) = \sum_{i=1}^{N} w_i k(\mathrm{x}, \mathrm{x}_i) + b$$

也就是基函数 $\phi_i(\mathrm{x})$ 限制为核函数 $k(\mathrm{x},\mathrm{x}_i)$ 的形式(待定偏置 b 也包含在内)；x_i由人为选定，是已知的。实际上就是借用了 SVM 判定边界的形式：

$$y(\mathrm{x})=\sum_{i=1}^{N}\alpha_i k(\mathrm{x},\mathrm{x}_i)+b$$

其中，N 表示训练样本总数，x_i 为第 i 个训练样本，α_i 为第 i 个训练样本的拉格朗日乘子；若$\alpha_i\neq 0$，则表示该样本点为支持向量。但狭义RVM中基函数的个数N可以任意选定，不表示训练样本总数；基函数$k(\mathrm{x},\mathrm{x}_i)$中的$\mathrm{x}_i$也可以任意选定，不表示训练样本。因此狭义RVM只是借用了SVM的形式而已。

显然，前文对广义RVM的叙述和推导对狭义RVM都成立。通过超参数的学习，狭义RVM中很多w_i都会变为0，产生稀疏的效果。类似SVM，那些 $w_i\neq 0$ 的项中的x_i被我们称为相关向量。

虽然狭义RVM中的基函数个数N、核函数形式$k(\cdot,\cdot)$和点 x_i的位置可任意设置，但若是为了比较狭义RVM与SVM的学习效果，我们会将狭义RVM设置得和SVM一样，即和SVM采用一样的核函数，N设置为训练样本总数，x_i也设置为训练样本，也就是使用不同的方法训练相同的模型以比较这两种方法的效果。通过实验可以发现，狭义RVM会产生更稀疏的解，泛化能力也不弱，算是全面碾压SVM了。