PRML（Pattern Recognition And Machine Learning）读书会

第六章　Kernel Methods

主讲人 网络上的尼采

（新浪微博：@Nietzsche_复杂网络机器学习）

QQ 群 177217565

读书会微信公众平台请扫描下面的二维码

网络上的尼采(813394698) 9:16:05
今天的主要内容：Kernel 的基本知识，高斯过程。边思考边打字，有点慢，各位稍安勿躁。
机器学习里面对待训练数据有的是训练完得到参数后就可以抛弃了，比如神经网络；有的是还需要原来的训练数据比如 KNN，SVM 也需要保留一部分数据--支持向量。
很多线性参数模型都可以通过 dual representation 的形式表达为核函数的形式。所谓线性参数模型是通过非线性的基函数的线性组合来表达非线性的东西，模型还是线性的。比如线性回归模型是 y= $\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)$，$\boldsymbol{\phi}(\mathbf{x}_n)$是一组非线性基函数，我们可以通过线性的模型来表达非线性的结构。

核函数的形式： $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}').$ ，也就是映射后高维特征空间的内积可以通过原来低维的特征得到。因此 kernel methods 用途广泛。

核函数有很多种，有平移不变的 stationary kernels $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}'),$ 还有仅依赖欧氏距离的径向基核： $k(\mathbf{x}, \mathbf{x}') = k(\|\mathbf{x} - \mathbf{x}'\|)$

非线性转化为线性的形式的好处不言而喻，各种变换推导、闭式解就出来了。 下面推导下线性回归模型的 dual representation，有助于我们理解核函数的作用：

根据最小二乘，我们得到下面的目标函数 $J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$ ，加了 L2 正则。

我们对 w 求导，令 J(w)的梯度等于 0，得到以下解：

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}\boldsymbol{\phi}(\mathbf{x}_n) = \sum_{n=1}^{N} a_n\boldsymbol{\phi}(\mathbf{x}_n) = \boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a}$$

$\boldsymbol{\Phi}$ 是个由基函数构成的样本矩阵，向量 $\mathbf{a}$ 里面的元素由 $a_n = -\frac{1}{\lambda}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}$ 组成：

where $\boldsymbol{\Phi}$ is the design matrix, whose n^{th} row is given by $\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}$. Here the vector $\mathbf{a} = (a_1, \ldots, a_N)^{\mathrm{T}}$, and we have defined

$$a_n = -\frac{1}{\lambda}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}. \tag{6.4}$$

我们把 $\mathbf{w} = \boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a}$ 代入最初的 J(w)得到：

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a}$$

咱们用核矩阵 K 来替换 $\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}$，其中矩阵 K 里面的元素是 $K_{nm} = \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$

于是得到 $J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{a}.$

然后 $J(\mathbf{a})$ 对 $\mathbf{a}$ 求导，令其梯度等于 0，得到解 $\mathbf{a} = (\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}.$

所以原来的线性回归方程就变成了 $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) = \mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\phi}(\mathbf{x}) = \mathbf{k}(\mathbf{x})^{\mathrm{T}}(\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}$

K(X)的含义：we have defined the vector $\mathbf{k}(\mathbf{x})$ with elements $k_n(\mathbf{x}) = k(\mathbf{x}_n, \mathbf{x})$，上面的 DUAL 形式的含义非常明显，就是根据已知的的训练数据来做预测。至此原来线性回归方程的参数 w 消失，由核函数来表示回归方程，以上方式把基于特征的学习转换成了基于样本的学习。 这是线性回归的 DUAL 表示，svm 等很多模型都有 DUAL 表示。

80(850639048) 10:09:50

professor 核函数其实是为了求基函数的内积对吗？

网络上的尼采(813394698) 10:12:57

如果有很多基的话维度势必会很高，计算内积的花销会很大，有些是无限维的，核函数能绕过高维的内积计算，直接用核函数得到内积。

接下来看下核函数的性质及构造方法。核函数的一般形式：

$$k(x, x') = \boldsymbol{\phi}(x)^{\mathrm{T}}\boldsymbol{\phi}(x') = \sum_{i=1}^{M} \phi_i(x)\phi_i(x')$$

where $\phi_i(x)$ are the basis functions.

下面是个简单的例子说明为什么$k(\mathbf{x}, \mathbf{z}) = \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2$是个核函数：

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2 = (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= (x_1^2, \sqrt{2}x_1 x_2, x_2^2)(z_1^2, \sqrt{2}z_1 z_2, z_2^2)^{\mathrm{T}} \\ &= \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{z}). \end{aligned}$$

很明显$k(\mathbf{x}, \mathbf{z}) = \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2$是个核函数，它能写成核函数的一般形式。

核函数的一个充分必要定理也就是 mercer 定理：核矩阵是半正定的：

More generally, however, we need a simple way to test whether a function constitutes a valid kernel without having to construct the function $\boldsymbol{\phi}(\mathbf{x})$ explicitly. A necessary and sufficient condition for a function $k(\mathbf{x}, \mathbf{x}')$ to be a valid kernel (Shawe-Taylor and Cristianini, 2004) is that the Gram matrix $\mathbf{K}$, whose elements are given by $k(\mathbf{x}_n, \mathbf{x}_m)$, should be positive semidefinite for all possible choices of the set $\{\mathbf{x}_n\}$. Note that a positive semidefinite matrix is not the same thing as a matrix whose elements are nonnegative.

我们可以通过以下规则用简单的核函数来构造复杂的核函数：

Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= c k_1(\mathbf{x}, \mathbf{x}') && (6.13)\\
k(\mathbf{x}, \mathbf{x}') &= f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') && (6.14)\\
k(\mathbf{x}, \mathbf{x}') &= q\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (6.15)\\
k(\mathbf{x}, \mathbf{x}') &= \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (6.16)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') && (6.17)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}') && (6.18)\\
k(\mathbf{x}, \mathbf{x}') &= k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) && (6.19)\\
k(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^{\mathrm{T}} \mathbf{A} \mathbf{x}' && (6.20)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) && (6.21)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) k_b(\mathbf{x}_b, \mathbf{x}'_b) && (6.22)
\end{aligned}
$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

过会我们讲高斯过程时再举个核函数线性组合的例子。

介绍一个经常用到的径向基核函数，高斯核：$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2\right)$，这个核函数能把数据映射到无限维的空间：

omitted. We can see that this is a valid kernel by expanding the square

$$\|\mathbf{x} - \mathbf{x}'\|^2 = \mathbf{x}^{\mathrm{T}}\mathbf{x} + (\mathbf{x}')^{\mathrm{T}}\mathbf{x}' - 2\mathbf{x}^{\mathrm{T}}\mathbf{x}'$$

to give

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\mathbf{x}^{\mathrm{T}}\mathbf{x}/2\sigma^2\right) \exp\left(\mathbf{x}^{\mathrm{T}}\mathbf{x}'/\sigma^2\right) \exp\left(-(\mathbf{x}')^{\mathrm{T}}\mathbf{x}'/2\sigma^2\right)$$

中间$\exp\left(\mathbf{x}^{\mathrm{T}}\mathbf{x}'/\sigma^2\right)$可以展开成无限维的，然后核函数可以表示成内积的形式。

内积的含义就是表示相似性，所以核函数还有其他的用法。比如我们可以通过生成模型来构造核。

Given a generative model $p(\mathbf{x})$ we can define a kernel by

$$k(\mathbf{x}, \mathbf{x}') = p(\mathbf{x}) p(\mathbf{x}').$$

两个变量的概率都很高相似性就越大，其实这样做就是映射到一维的内积。

我们可以引入离散的隐变量：$k(\mathbf{x}, \mathbf{x}') = \sum_i p(\mathbf{x}|i) p(\mathbf{x}'|i) p(i).$

连续的隐变量：$k(\mathbf{x}, \mathbf{x}') = \int p(\mathbf{x}|\mathbf{z}) p(\mathbf{x}'|\mathbf{z}) p(\mathbf{z})\, \mathrm{d}\mathbf{z}$

举个这样做有啥用的例子，我们可以用来比较 HMM 由同一条隐马尔科夫链生成的两条序列的相似性：

Now suppose that our data consists of ordered sequences of length L so that an observation is given by $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_L\}$. A popular generative model for sequences is the hidden Markov model, which expresses the distribution $p(\mathbf{X})$ as a marginalization over a corresponding sequence of hidden states $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_L\}$. We can use this approach to define a kernel function measuring the similarity of two sequences $\mathbf{X}$ and $\mathbf{X}'$ by extending the mixture representation (6.29) to give

$$k(\mathbf{X}, \mathbf{X}') = \sum_{\mathbf{Z}} p(\mathbf{X}|\mathbf{Z}) p(\mathbf{X}'|\mathbf{Z}) p(\mathbf{Z}) \tag{6.31}$$

so that both observed sequences are generated by the same hidden sequence $\mathbf{Z}$. This model can easily be extended to allow sequences of differing length to be compared.

网络上的尼采(813394698) 10:40:34

接下来讲我们今天的重点 Gaussian Processes

牧云(1106207961) 10:41:02

数据海洋(1009129701) 10:41:14

我先再理解，理解这些。

网络上的尼采(813394698) 10:42:41

Gaussian Processes 是贝叶斯学派的一个大杀器，用处很广。不同于参数模型，Gaussian Processes 认为函数在函数空间里存在一个先验分布。

高斯过程和很多模型是等价的：ARMA (autoregressive moving average) models, Kalman filters, radial basis function networks ，还有特定情况下的神经网络。

现在我们从贝叶斯线性回归自然的引出高斯过程：

前面我们提到的线性回归的形式 $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$

贝叶斯方法为参数加了一个高斯分布 $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$

大家发现了没有，这样做直接导致了函数有个预测分布，并且也是高斯的，因为方程是线性的并且参数是高斯分布。线性的东西和高斯分布总是不分家的。

我们定义向量 $\mathbf{y}$ ：

$\mathbf{x}_1, \ldots, \mathbf{x}_N$. We are therefore interested in the joint distribution of the function values $y(\mathbf{x}_1), \ldots, y(\mathbf{x}_N)$, which we denote by the vector $\mathbf{y}$ with elements $y_n = y(\mathbf{x}_n)$ ， $\mathbf{y}$ 就是个多元的高斯分布。

它的每一维 $y(\mathbf{x}_n)$ 都是个高斯分布，这也是高斯过程的由来。

$\mathbf{y}$ 可以表示为 $\mathbf{y} = \boldsymbol{\Phi}\mathbf{w}$

where $\boldsymbol{\Phi}$ is the design matrix with elements $\Phi_{nk} = \phi_k(\mathbf{x}_n)$.

$\boldsymbol{\Phi}$ 是基函数组成的样本矩阵。

高斯过程可以由均值和协方差矩阵完全决定。由于 w 的均值是 0，所以我们也认为高斯过程的均值是 0，剩下的就是根据定义求它的协方差矩阵，很自然地就得出来了：

$$\text{cov}[\mathbf{y}] = \mathbb{E}\left[\mathbf{y}\mathbf{y}^{\mathrm{T}}\right] = \mathbf{\Phi}\mathbb{E}\left[\mathbf{w}\mathbf{w}^{\mathrm{T}}\right]\mathbf{\Phi}^{\mathrm{T}} = \frac{1}{\alpha}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}} = \mathbf{K}$$

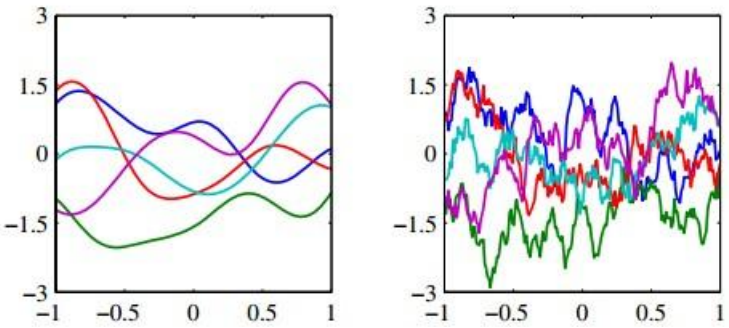
where $\mathbf{K}$ is the Gram matrix with elements

$$K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha}\phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m)$$

矩阵 K 里的元素 $K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha}\phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m)$ 都是核函数的形式。

选用什么样的核函数也是问题，下面的图是对采用高斯核和指数核的高斯过程的取样，一共取了 5 条，可以看到两者的区别：

Figure 6.4 Samples from Gaussian processes for a 'Gaussian' kernel (left) and an exponential kernel (right).



接下来我们就用 GP 来做回归 ：
我们观测的目标值是包含噪音的，噪音是高斯分布。

$$t_n = y_n + \epsilon_n$$

那么根据线性高斯模型的性质， $p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1})$ ，其中 β 是噪音的参数

对于向量 $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$ 和向量 $\mathbf{y} = (y_1, \ldots, y_N)^{\mathrm{T}}$

$$p(\mathbf{t}|\mathbf{y}) = \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}\mathbf{I}_N)$$

咱们前面说过了， $p(\mathbf{y})$ 可以表示为 $p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K})$.

所以 marginal distribution： $p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y})\,\mathrm{d}\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C})$

其中矩阵 C 的元素 $C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1}\delta_{nm}$ ， δ_{nm} 是单位矩阵的元素，其实就是把 β^{-1} 加在了矩阵 $\mathbf{K}$ 的对角线上。这个不难理解，一开始 $t_n = y_n + \epsilon_n$ ，都是高斯的，协方差是两者的相加，噪音每次取都是独立的，所以只在协方差矩阵对角线上有。

现在确定下用什么核的问题，举个例子，下面这个核函数用了高斯核，线性核，以及常数的线性组合，这样做是为了更灵活，过会再讲如何确定里面的这些超参：

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

下图是不同的超参对高斯过程的影响：

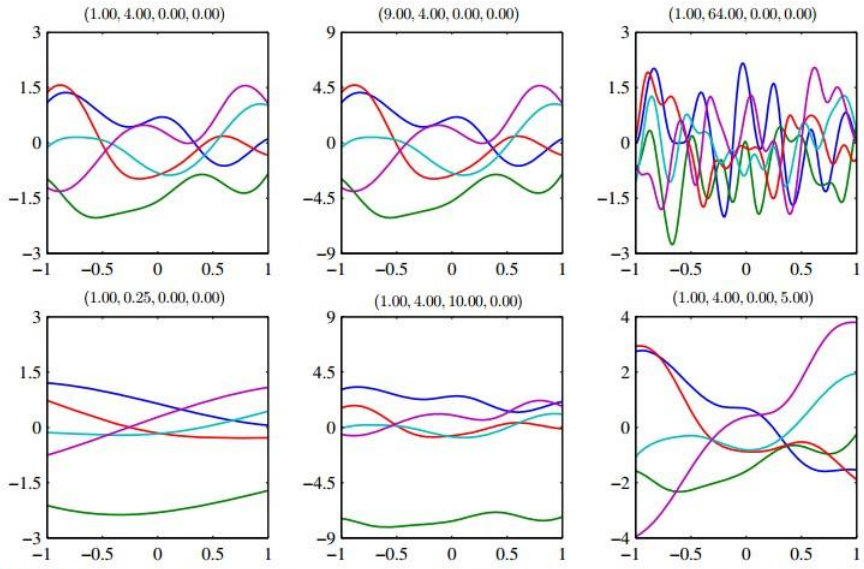


Figure 6.5 Samples from a Gaussian process prior defined by the covariance function (6.63). The title above each plot denotes $(\theta_0, \theta_1, \theta_2, \theta_3)$.

解决了核函数的问题，我们再回来，通过前面的结论，不难得出 $p(\mathbf{t}_{N+1}) = \mathcal{N}(\mathbf{t}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1})$

如何确定矩阵 $\mathbf{C}_{N+1}$ 呢，其实我们在原来矩阵 $\mathbf{C}_N$ 的基础上补上就行。

$$\mathbf{C}_{N+1} = \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^{\mathrm{T}} & c \end{pmatrix}$$

k 和 c 比较容易理解：

the vector $\mathbf{k}$ has elements $k(\mathbf{x}_n, \mathbf{x}_{N+1})$ for $n = 1, \ldots, N$,

$$c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \beta^{-1}$$

咱们的最终目标就是得 $p(t_{N+1}|\mathbf{t})$，由于这两个都是高斯分布，用第二章条件高斯分布的公式套一下，其中均值是 0：

From these we obtain the following expressions for the mean and covariance of the conditional distribution $p(\mathbf{x}_a|\mathbf{x}_b)$

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b) \quad (2.81)$$

$$\boldsymbol{\Sigma}_{a|b} = \boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba}. \quad (2.82)$$

就会得到 $p(t_{N+1}|\mathbf{t})$ 的均值和方差：

ditional distribution $p(t_{N+1}|\mathbf{t})$ is a Gaussian distribution with mean and covariance given by

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} \quad (6.66)$$

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}. \quad (6.67)$$

可以看出均值和方差都是 $\mathbf{x}_{N+1}$ 的函数，我们做预测时用均值就行了。

最后一个问题就是高斯过程是由它的协方差矩阵完全决定的，我们如何学习矩阵里面的超参呢？包括我们刚才提到的核函数里面的参数以及噪音的参数。

其实由于高斯分布的原因，我们可以方便的利用 log 最大似然：

$$\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2}\ln|\mathbf{C}_N| - \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} - \frac{N}{2}\ln(2\pi)$$

求最优解时可以用共轭梯度等方法，梯度：

$$\frac{\partial}{\partial\theta_i}\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2}\mathrm{Tr}\left(\mathbf{C}_N^{-1}\frac{\partial\mathbf{C}_N}{\partial\theta_i}\right) + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{C}_N^{-1}\frac{\partial\mathbf{C}_N}{\partial\theta_i}\mathbf{C}_N^{-1}\mathbf{t}.$$

到这里，用高斯过程做回归就结束了。

有了做回归的基础，咱们再看下如何做分类。

类似逻辑回归，加个 sigmoid 函数就能做分类了：

$$p(\mathbf{x}) = \frac{1}{1+e^{-a(\mathbf{x})}}$$

where $a(\mathbf{x})$ comes from a Gaussian process

分类与回归不同的是 $p(t|a) = \sigma(a)^t(1-\sigma(a))^{1-t}$ 是个伯努利分布。

$p(\mathbf{a}_{N+1}) = \mathcal{N}(\mathbf{a}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1})$ 这里还和前面一样。

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \nu\delta_{nm}$$

对于二分类问题，最后我们要得到是：

$$p(t_{N+1}=1|\mathbf{t}_N) = \int p(t_{N+1}=1|a_{N+1})p(a_{N+1}|\mathbf{t}_N)\,\mathrm{d}a_{N+1}$$

但是这个积分是不容易求的，$\mathbf{t}_N$ 是伯努利分布，a_{N+1} 是高斯分布，不是共轭的。求积分的方法有很多，可以用 MCMC，也可以用变分的方法。书上用的是 Laplace approximation。

今天就到这里吧，我去吃饭，各位先讨论下吧。

上面 Gaussian Processes 的公式推导虽然有点多，但都是高斯分布所以并不复杂，并且 GP 在算法实现上也不难。

另外给大家推荐一个机器学习视频的网站，

http://blog.videolectures.net/100-most-popular-machine-learning-talks-at-videolectures-net/

里面有很多牛人比如 Jordan 的 talks，第一个视频就是剑桥的 David MacKay 讲高斯过程，他的一本书 Information Theory, Inference and Learning Algorithms 也很出名。

两栖动物(9219642) 14:35:09

$\mathbf{\Phi}$ 是个由基函数构成的矩阵，向量 $\mathbf{a}$ 里面的元素由 $a_n = -\frac{1}{\lambda}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}$ 组成。

$\mathbf{\Phi}$ 的维度是基函数的个数，an 的维度是样本的个数把?

网络上的尼采(813394698) 14:36:44

对

两栖动物(9219642) 14:37:48

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}\boldsymbol{\phi}(\mathbf{x}_n) = \sum_{n=1}^{N} a_n\boldsymbol{\phi}(\mathbf{x}_n) = \mathbf{\Phi}^{\mathrm{T}}\mathbf{a}$$

哪这 2 个怎么后来乘在一起了？维度不是不一样吗？

网络上的尼采(813394698) 14:49:01

@两栖动物 $\mathbf{\Phi}$ 不是方阵，可以相乘

$\mathbf{\Phi}$ is the design matrix, whose n^{th} row is given by $\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}$. 明白了吧，另外这个由基函数表示的样本矩阵只在推导里存在。

两栖动物(9219642) 14:56:08

明白了，谢谢