子午线

学习笔记

作者：leekarry
组织：果壳
时间：June 17, 2019
版本：0.1

ElegantLaTeX Program

我很快，快到时间都会变慢。而我一慢，时间就会过得飞快。

目　录

第 I 部分 I

机器学习

第 1 章 机器学习概论

1.1 基本概念

统计学习的特点

统计学习 (statistical learning) 是关于计算机基于数据构建概率统计模型并运用模型对数据进行预测与分析的一门学科。

- 统计学习以计算机及网络为平台；
- 统计学习以数据为研究对象；
- 统计学习的目的是对数据进行预测与分析；
- 统计学习以方法为中心；
- 统计学习是概率论、统计学、信息论、计算理论、最优化理论及计算机科学等多个领域的交叉学科。

统计学习的对象

统计学习的对象是数据 (data)。它从数据出发，提取数据的特征，抽象出数据的模型，发现数据中的知识，又回到对数据的分析与预测中去。**统计学习关于数据的基本假设是同类数据具有一定的统计规律性，这是统计学习的前提**。

统计学习的目的

统计学习用于对数据进行预测与分析，特别是对未知新数据进行预测与分析。对数据的预测与分析是通过构建概率统计模型实现的。统计学习总的目标就是考虑学习什么样的模型和如何学习模型，以使模型能对数据进行准确的预测与分析，同时也要考虑尽可能地提高学习效率。

统计学习的方法

统计学习的方法是基于数据构建统计模型从而对数据进行预测与分析。

- 监督学习 (supervised learning)
- 非监督学习 (unsupervised learning)
- 半监督学习 (semi-supervised learning)
- 强化学习 (reinforcement learning)

实现统计学习方法的步骤如下：

(1) 得到一个有限的训练数据集合

(2) 确定包含所有可能的模型的假设空间，即学习模型的集合

(3) 确定模型选择的准则，即学习的策略

(4) 实现求解最优化模型的算法,即学习的算法
(5) 通过学习方法选择最优模型
(6) 利用学习的最优模型对新数据进行预测或分析

统计学习的研究

统计学习方法 (statistical learning method),旨在开发新的学习方法。

统计学习理论 (statistical learning theory),旨在探求统计学习方法的有效性与效率。

统计学习应用 (application of statistical learning),旨在将统计学习方法应用到实际问题中去,解决实际问题。

统计学习的重要性

统计学习是处理海量数据的有效方法;统计学习是计算机智能化的有效手段;统计学习是计算机科学习发展的一个重要组成部分。

1.2 统计学习三要素

(1) 模型:统计学习首先要考虑的问题是学习什么样的模型
(2) 策略:有了模型的假设空间,统计学习接着需要考虑的是按照什么样的准则学习或选择最优的模型
(3) 算法:算法是指学习模型的具体计算方法

1.3 模型评估与模型选择

统计学习的目的是使学到的模型不仅对已知数据而且对未知数据都能有很好的预测能力。不同的学习方法会给出不同的模型。当损失函数给定时,基于损失函数的模型的训练误差(training error)和模型的测试误差(test error)就自然成为学习方法评估的标准。测试误差反映了学习方法对未知的测试数据集的预测能力——泛化能力。

1.4 正则化与交叉验证

模型选择的典型方法是正则化 (regularization)。正则化是结构风险最小化策略的实现,是在经验风险上加一个正则化项 (regularizer) 和罚项 (penalty term)

$$\min_{f\in F} \quad \frac{1}{2}\sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f) \tag{1.1}$$

另一种常用的模型选择方法是交叉验证 (cross validation) 训练集用来训练模型验证集用来选择模型测试集用来对模型进行评估

1. 简单交叉验证

2. S 折交叉验证
3. 留一交叉验证

1.5 生成模型与判别模型

监督学习方法又可以分为生成方法 (generative approach) 和判别方法 (discriminative approach)。所学到的模型分别称为生成模型和判别模型。

生成方法由数据学习联合分布 $P(X,Y)$,然后求出条件概率分布 $P(Y|X)$ 作为预测的模型,即生成模型:

$$P(Y|X) = \frac{P(X,Y)}{P(X)} \tag{1.2}$$

这样的方法之所以称为生成方法,是因为模型表示了给定输入 X 产生输出 Y 的生成关系。

判别方法由数据直接学习决策函数 $f(X)$ 或者条件概率分布 $P(Y|X)$ 作为预测的模型,即判模型。判别方法关心的是对给定的输入 X,应该预测什么样的输出 Y。

生成方法的特点:生成方法可以还原出联合概率分布 P(X|Y),而判别方法则不能;生成方法的学习收敛速度更快,即当样本容量增加的时候,学到的模型可以更快地收敛于真实模型;当存在隐变量时,仍可以用生成方法学习,此时判别方法就不能用。

判别方法的特点:判别方法直接学习的是条件概率 P(Y|X) 或决策函数 $f(X)$,直接面对预测,往往学习的准确率更高;由于直接学习 P(Y|X) 或 $f(X)$,可以对数据进行各种程度上的抽象、定义特征并使用特征,因此可以简化学习问题。

1.6 频率学派和贝叶斯学派的参数估计

频率学派与贝叶斯学派的区别

简单地说,频率学派与贝叶斯学派探讨**不确定性**这件事的出发点与立足点同。频率学派从" 自然" 角度出发,试图直接为" 事件" 本身建模,即事件 A 在独立重复试验中发生的频率趋于极限 p,那么这个极限就是该事件发生的概率。贝叶斯学派并不从试图刻画" 事件" 本身,而从" 观察者" 角度出发。贝叶斯学派并不试图说" 事件本身是随机的",或者" 世界的本体带有某种随机性",而只是从" 观察者知识不完备" 这一出发点开始,构造一套在贝叶斯概率论的框架下可以对不确定知识做出推断的方法。体现在参数估计中,频率学派认为参数是客观存在,不会改变,虽然未知,但却是固定值;贝叶斯学派则认为参数是随机值,因此参数也可以有分布。

频率学派的参数估计

极大似然估计 (Maximum Likelihood Estimate,MLE),也叫最大似然估计。若总体 X 属离散型 (连续型与此类似),其分布律 $P\{X = x\} = p(x;\theta), \theta \in \Theta$ 的形式为已知,θ 为待估参数,Θ 是 θ 的取值范围,设 $X_1, X_2, \ldots, X_n$ 是来自 X 的样本,则 $X_1, X_2, \ldots, X_n$ 的联合概率分

布为

$$\prod_{i=1}^{n} p(x;\theta) \tag{1.3}$$

设 $x_1, x_2, \ldots, x_n$ 是相应的样本值，则

$$L(\theta) = L(x_1, x_2, \ldots, x_n; \hat{\theta}) = \underset{\theta \in \Theta}{argmax} \prod_{i=1}^{n} p(x;\theta) \tag{1.4}$$

贝叶斯学派的参数估计

最大后验估计 (Maximum a Posteriori estimation,MAP)，它与极大似然估计最大的区别就是，它考虑了参数本身的分布，也就是先验分布。最大后验估计是根据经验数据获得对难以观察的量的点估计。可以看作规则化的最大似然估计。假设 x 为独立同分布的采样，θ 为模型参数，p 为我们所使用的模型。那么最大似然估计可以表示为

$$\hat{\theta}_{MLE}(x) = \underset{\theta}{argmax}\, p(x|\theta) \tag{1.5}$$

现在，假设 θ 的先验分布为 g。通过贝叶斯理论，对于 θ 的后验分布如下式所示：

$$p(\theta|x) = \frac{p(x|\theta)g(\theta)}{\int_{\theta \in \Theta} p(x|\theta')g(\theta')d\theta'} \tag{1.6}$$

分母为 x 的边缘概率与 θ 无关，因此最大后验等价于使分子最大，故目标函数为

$$\hat{\theta}_{MAP}(x) = \underset{\theta}{argmax}\, p(x|\theta)g(\theta) \tag{1.7}$$

第 2 章 概率分布

概率论在解决模式识别问题时起着重要作用。现在探究一下某些特殊的概率分布的例子以及它们的性质。概率分布的一个作用是在给定有限次观测 $x_1, \ldots, x_N$ 的前提下，对随机变量 $\vec{x}$ 的概率分布 $p(\vec{x})$ 建模。这个问题被称为密度估计 (density estimation)。本章中，我们假设数据点是独立同分布的。

首先，我们考虑离散随机变量的二项分布和多项式分布，以及连续随机变量的高斯分布。这是参数分布 (paramertric distribution) 的具体例子。之所以被称为参数分布，是因为少量可调节的参数控制了整个概率分布。为了把这种模型应用到密度估计问题中，我们需要一个步骤，能够在给定观察数据集的条件下，确定参数的合适的值。在频率学家的观点中，我们通过最优化某些准则 (例如似然函数) 来确定参数的具体值。相反，在贝叶斯观点中，给定观察数据，我们引入参数的先验分布，然后使用贝叶斯定理来计算对应后验概率分布。

我们会看到，共轭先验 (conjugate prior) 有着很重要的作用，它使得后验概率分布的函数形式与先验概率相同，因此使得贝叶斯分析得到了极大的简化。指数族分布有很多重要的性质，将在本章详细讨论。

参数方法的一个限制是它假定分布有一个具体的函数形式，这对于一个具体应用来说是不合适的。另一种替代的方法是非参数 (nonparametric) 密度估计方法。这种方法中分布的形式通常依赖于数据集的规模。这些模型仍然具有参数，但是这些参数控制的是模型的复杂度而不是分布的形式。本章最后，我们会考虑三种非参数化方法，分布依赖于直方图、最近邻以及核函数。

2.1 二元分布

考虑一个二元随机变量 $x \in \{0, 1\}$。$x = 1$ 的概率被记作参数 μ，因此

$$p(x = 1|\mu) = \mu \tag{2.1}$$

其中 $0 \leqslant \mu \leqslant 1$。$x$ 的概率分布因此可以写成

$$Bern(x|\mu) = \mu^x(1 - \mu)^{1-x} \tag{2.2}$$

这被叫做伯努利分布 (Bernoulli distribution)。均值和方差为

$$\mathbb{E}[x] = \mu \tag{2.3}$$

$$var[x] = \mu(1 - \mu) \tag{2.4}$$

用最大似然估计方法求得

$$\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n = \frac{m}{N} \tag{2.5}$$

m 为数据集里 $x = 1$(正面朝上) 的观测数量。

我们也可以求解给定数据集规模 N 的条件下，$x = 1$ 的观测出现的数量 m 的概率分布。这被称谓二项分布 (binomial distribution)。

$$Bin(m|N,\mu) = \binom{N}{m}\mu^m(1-\mu)^{N-m} \tag{2.6}$$

其中

$$\binom{N}{m} \equiv \frac{N!}{(N-m)!m!} \tag{2.7}$$

均值和方差为

$$\mathbb{E}[m] \equiv \sum_{m=0}^{N} mBin(m|N,\mu) = N\mu \tag{2.8}$$

$$var[m] \equiv \sum_{m=0}^{N}(m-\mathbb{E}[m])^2 Bin(m|N,\mu) = N\mu(1-\mu) \tag{2.9}$$

Beta 分布

我们已经看到伯努利分布的参数 μ 的最大似然解。对于小规模的数据集会给出严重的过拟合结果。为了用贝叶斯的观点看待这个问题，我们需要引入一个关于 μ 的先验分布 $p(\mu)$。

在贝叶斯统计中，如果后验分布与先验分布属于同类，则先验分布与后验分布被称为**共轭分布**，而先验分布被称为似然函数的共轭先验。具体地说，就是给定贝叶斯公式，假定似然函数 $p(x|\theta)$ 是已知的，问题就是选取什么样的先验分布 $p(\theta)$ 会让后验分布与先验分布具有相同的数学形式。共轭先验的好处主要在于代数上的方便性，可以直接给出后验分布的封闭形式，否则的话只能数值计算。共轭先验也有助于获得关于似然函数如何更新先验分布的直观印象。所有指数家族的分布都有共轭先验。

因此，我们把先验分布选择为 Beta 分布，定义为

$$Beta(\mu|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)+\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1} \tag{2.10}$$

其中，$\Gamma(x)$ 定义为

$$\Gamma(x) \equiv \int_0^\infty u^{x-1}e^{-u}du \tag{2.11}$$

有以下性质

$$\Gamma(x+1) = x\Gamma(x) \tag{2.12}$$

$$\Gamma(x+1) = x! \tag{2.13}$$

Beta 分布的均值和方差为

$$\mathbb{E}[\mu] = \frac{a}{a+b} \tag{2.14}$$

$$var[\mu] = \frac{ab}{(a+b)^2(a+b+1)} \tag{2.15}$$

参数 a 和 b 被称为超参数 (hyperparameter)，因为它们控制了参数 μ 的概率分布。

μ 的后验概率分布现在可以这样得到：把 Beta 先验与二项似然函数相乘，然后归一化。我们看到后验概率分布的形式为

$$p(\mu|m,l,a,b) = \frac{\Gamma(m+a+l+b)}{\Gamma(m+a)\Gamma(l+b)}\mu^{m+a+1}(1-\mu)^{l+b+1} \tag{2.16}$$

其中 $l = N - m$，即对应于硬币“反面朝上”的样本数量。我们看到，如果一个数据集里有 m 次观测为 $x = 1$，有 l 次观测为 $x = 0$，那么从先验概率到后验概率，a 的值变大了 m, b 的值变大了 l。这让我们可以简单地把先验概率中的超参数 a 和 b 分别看成 $x = 1$ 和 $x = 0$ 的有效观测数。

另外，如果接下来观测到更多的数据，那么后验概率分布可以扮演先验概率的角色。学习过程中的顺序 (sequential) 方法可以自然而然地得出。它与先验和似然函数的选择无关，只取决于数据独立同分布的假设。

给定数据集 D 的情况下，x 的预测分布为

$$p(x=1|D) = \int_0^1 p(x=1|\mu)p(\mu|D)d\mu = \int_0^1 \mu p(\mu|D)d\mu = \mathbb{E}[\mu|D] \tag{2.17}$$

2.2 多项式变量

二元变量可以用来描述只能取两种可能值中的某一种这样的量。然而，经常会遇到可以取 K 个互斥状态中的某一种的离散变量。一种比较方便的表示方法是“1-of-K”表示法。例如，如果我们有一个能够取 K=6 种状态的变量，这个变量的某次特定的观测恰好对应于 $x_3 = 1$ 的状态，那么 x 就可以表示为

$$\boldsymbol{x} = (0,0,1,0,0,0)^T \tag{2.18}$$

这样的向量满足 $\sum_{k=1}^{K} x_k = 1$。如果我们用参数 μ_k 表示 $x_k = 1$ 的概率，那么 x 的分布就是

$$p(\boldsymbol{x}|\boldsymbol{\mu}) = p(x_1, x_2, \ldots, x_K|\mu_1, \ldots, \mu_K) = \prod_{k=1}^{K} \mu_k^{x_k} \tag{2.19}$$

参数 μ_k 要满足 $\mu_k \geqslant 0, \sum_k \mu_k = 1$。易知

$$\mathbb{E}[\boldsymbol{x}|\boldsymbol{\mu}] = \sum_{\boldsymbol{x}} p(\boldsymbol{x}|\mu)\boldsymbol{x} = (\mu_1, \mu_2, \ldots, \mu_K)^T = \boldsymbol{\mu} \tag{2.20}$$

现在考虑一个有 N 个独立观测值 $x_1, x_2, \ldots, x_N$ 的数据集 D。对应的似然函数的形式为

$$p(D|\mu) = \prod_{n=1}^{N}\prod_{k=1}^{K} \mu_k^{x_{nk}} = \prod_{k=1}^{K} \mu_k^{(\sum_n x_{nk})} = \prod_{k=1}^{K} \mu_k^{m_k} \tag{2.21}$$

其中

$$m_k = \sum_n x_{nk} \tag{2.22}$$

表示观测到 $x_k = 1$ 的次数。这被称为这个分布的充分统计量。为了找到 $\boldsymbol{\mu}$ 的最大似然解，需要关于 μ_k 最大化 $\ln p(D|\boldsymbol{\mu})$，通过拉格朗日乘数 λ 实现。

$$\mu_k^{ML} = \frac{m_k}{N} \tag{2.23}$$

我们可以考虑 $m_1, m_2, \ldots, m_K$ 在参数 μ 和观测总数 N 条件下的联合分布。这个分布的形式为

$$Mult(m_1, m_2, \ldots, m_k|\boldsymbol{\mu}, N) = \binom{N}{m_1 m_2 \ldots m_K} \prod_{k=1}^{K} \mu_k^{m_k} \tag{2.24}$$

这被称为多项式分布 (multinomial distribution)。归一化系数是把 N 个物体分成大小为 $m_1, m_2, \ldots, m_K$ 的 K 组的方案总数，定义为

$$\binom{N}{m1m2 \ldots m_K} = \frac{N!}{m_1!m_2! \ldots m_K!} \tag{2.25}$$

注意，m_k 满足下面的限制

$$\sum_{k=1}^{K} m_k = N \tag{2.26}$$

狄利克雷分布

现在介绍多项式分布的参数 $\{\mu_k\}$ 的一组先验分布。通过观察多项式分布的形式，我们看到，共轭先验为

$$p(\boldsymbol{\mu}|\alpha) \propto \prod_{k=1}^{K} \mu_k^{\alpha_k - 1} \tag{2.27}$$

其中$0 \leqslant \mu_k \leqslant 1$且$\sum_k \mu_k = 1$。这里，$\alpha_1, \alpha_2, \ldots, \alpha_K$是分布的参数，$\boldsymbol{\alpha}$表示$(\alpha_1, \alpha_2, \ldots, \alpha_K)^T$。概率的归一化形式为

$$Dir(\boldsymbol{\mu}|\boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\ldots\Gamma(\alpha_K)} \prod_{k=1}^{K} \mu_k^{\alpha_k - 1} \tag{2.28}$$

这被称为狄利克雷分布 (Dirichlet distribution)。

$$\alpha_0 = \sum_{k=1}^{K} \alpha_k \tag{2.29}$$

用似然函数乘以先验，我们得到了参数$\{\mu_k\}$的后验分布，形式为

$$p(\boldsymbol{\mu}|D, \boldsymbol{\alpha}) \propto p(D|\boldsymbol{\mu})p(\boldsymbol{\mu}|\boldsymbol{\alpha}) \propto \prod_{k=1}^{K} \mu_k^{\alpha_k + m_k - 1} \tag{2.30}$$

我们看到后验分布的形式又变成了狄利克雷分布。确定归一化系数后变成

$$p(\boldsymbol{\mu}|D, \boldsymbol{\alpha}) = Dir(\boldsymbol{\mu}|\boldsymbol{\alpha} + \boldsymbol{m}) \tag{2.31}$$

2.3 高斯分布

高斯分布，也被称为正态分布，广泛应用于连续型随机变量分布的模型中。对于一元变量x的情形，

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \tag{2.32}$$

其中μ是均值，σ^2是方差。

$$\mu_{ML} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \text{无偏} \tag{2.33}$$

$$\sigma^2_{ML} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{ML})^2 \qquad \text{有偏} \tag{2.34}$$

$$\mathbb{E}[\sigma^2_{ML}] = \frac{N-1}{N}\sigma^2 \tag{2.35}$$

$$\hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_{ML})^2 \qquad \text{无偏} \tag{2.36}$$

对于 D 维向量$\boldsymbol{x}$，多元高斯分布的形式为

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\} \tag{2.37}$$

其中，μ是一个 D 维均值向量，Σ是一个$D \times D$的协方差矩阵，$|\Sigma|$是Σ的行列式。对Σ进

行正交分解。

$$\Sigma = U\Lambda U^T \tag{2.38}$$

$$UU^T = U^TU = I \tag{2.39}$$

$$\Lambda = diag(\lambda_i) \tag{2.40}$$

$$U = (u_1, u_2, \ldots, u_p)_{p\times p} \tag{2.41}$$

考虑高斯分布的几何形式。高斯对于 $\boldsymbol{x}$ 的依赖是通过下面形式的二次型

$$\triangle^2 = (\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \tag{2.42}$$

$\triangle$ 被叫做 $\boldsymbol{\mu}$ 和 $\boldsymbol{x}$ 之间的马氏距离。协方差矩阵 Σ 可以表示成特征向量的展开的形式

$$\begin{aligned}\Sigma &= U\Lambda U^T \\ &= (u_1\ u_2\ \ldots\ u_p)\begin{pmatrix}\lambda_1 & \ldots & 0\\ \vdots & \lambda_i & \vdots\\ 0 & \ldots & \lambda_p\end{pmatrix}\begin{pmatrix}u_1^T\\ \vdots\\ u_P^T\end{pmatrix} \\ &= (u_1\lambda_1\ \ldots\ u_p\lambda_p)\begin{pmatrix}u_1^T\\ \vdots\\ u_P^T\end{pmatrix} \\ &= \sum_{i=1}^{P} u_i\lambda_i u_i^T\end{aligned} \tag{2.43}$$

于是

$$\begin{aligned}\Sigma^{-1} &= (U\Lambda U^T)^{-1} = U\Lambda^{-1}U^T \\ &= \sum_{i=1}^{P} u_i\frac{1}{\lambda_i}u_i^T\end{aligned} \tag{2.44}$$

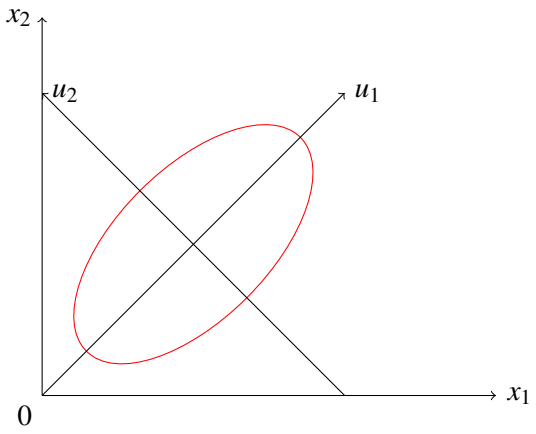
马氏距离就可以表示为

$$\begin{aligned}\triangle^2 &= (\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \\ &= \sum_{i=1}^{P}\underbrace{(\boldsymbol{x}-\boldsymbol{\mu})^Tu_i}_{y_i}\frac{1}{\lambda_i}\underbrace{u_i^T(\boldsymbol{x}-\boldsymbol{\mu})}_{y_i^T} \quad = \sum_{i=1}^{P}\frac{y_i^2}{\lambda_i}\end{aligned} \tag{2.45}$$

其中

$$y_i = (\boldsymbol{x}-\boldsymbol{\mu})^Tu_i \tag{2.46}$$

可以把 $\{y_i\}$ 表示成单位正交向量 u_i 关于原始的 x_i 坐标经过平移和旋转后形成的新的坐标系。



高斯分布的均值和方差为

$$\mathbb{E}[\boldsymbol{x}] = \boldsymbol{\mu} \tag{2.47}$$

$$var[\boldsymbol{x}] = \Sigma \tag{2.48}$$

虽然高斯分布被广泛用作概率密度模型，但是它有着一些巨大的局限性.

1. 自由参数的数量。可以进一步地把协方差矩阵限制成正比于单位矩阵，$\Sigma = \sigma^2 I$，被称为各向同性 isotropic 的协方差。尽管这样的方法限制了概率分布的自由度的数量，并且使得求协方差矩阵的逆矩阵可以更快地完成，但是这样做也极大地限制了概率密度的形式，限制了它描述模型中有趣的相关性的能力。
2. 单峰。因此不能够很好地近似多峰分布。

高斯分布一方面相当灵活，因为它有很多参数。另一方面，它又有很大的局限性，因为它不能够近似很多分布。引入潜在变量 (latent variable) 也被称为隐藏变量 (hidden variable) 或者未观察变量 (unobserved variable)，会让这两个问题都得到解决。

条件高斯分布

多元高斯分布的一个重要性质是，如果两组变量是联合高斯分布，那么以一组变量为条件，另一组变量同样是高斯分布，类似地，任何一个变量的边缘分布也是高斯分布。

首先考虑条件概率的情形，假设 $\boldsymbol{x}$ 是一个服从高斯分布 $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma)$ 的 D 维向量。我们把 $\boldsymbol{x}$ 划分成两个不相交的子集 x_a, x_b。不失一般性，我们可令 x_a 为 $\boldsymbol{x}$ 的前 M 个分量，令

x_b 为剩余的 $D-M$ 个分量，因此

$$\boldsymbol{x} = \begin{pmatrix} x_a \\ x_b \end{pmatrix} \tag{2.49}$$

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix} \tag{2.50}$$

$$\Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix} \tag{2.51}$$

$$\tag{2.52}$$

注意，协方差矩阵的对称性 $\Sigma^T = \Sigma$ 表明 Σ_{aa} 和 Σ_{bb} 也是对称的，而 $\Sigma_{ab}^T = \Sigma_{ba}$。

定理 2.1

已知 $x \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma), Y = AX + B$ 有如下结论：

1. $\mathbb{E}[Y] = A\mu + B$
2. $var[Y] = A \cdot \Sigma \cdot A^T$ ♡

x_a 可以表示为

$$x_a = \underbrace{(I_m \ o)}_{A} \underbrace{\begin{pmatrix} x_a \\ x_b \end{pmatrix}}_{x} \tag{2.53}$$

$$\mathbb{E}[x_a] = (I_m \ o)(\mu_a \ \mu_b)^T = \mu_a \tag{2.54}$$

$$var[x_a] = (I_m \ o)\begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}\begin{pmatrix} I_m \\ o \end{pmatrix} = \Sigma_{aa} \tag{2.55}$$

$$x_a \sim \mathcal{N}(\mu_a, \Sigma_{aa}) \tag{2.56}$$

同理

$$\mathbb{E}[x_b] = \mu_b \tag{2.57}$$

$$var[x_b] = \Sigma_{bb} \tag{2.58}$$

$$x_b \sim \mathcal{N}(\mu_b, \Sigma_{bb}) \tag{2.59}$$

构造

$$x_{b|a} = x_b - \Sigma_{ba}\Sigma_{aa}^{-1}x_a \tag{2.60}$$

$$\mu_{b|a} = \mu_b - \Sigma_{ba}\Sigma_{aa}^{-1}\mu_a \tag{2.61}$$

$$\Sigma_{bb|a} = \Sigma_{bb} - \Sigma_{ba}\Sigma_{aa}^{-1}\Sigma_{ab} \tag{2.62}$$

$$x_{b|a} = \underbrace{(-\Sigma_{ba}\Sigma_{aa}^{-1}\ I_n)}_{A}\underbrace{\begin{pmatrix} x_a \\ x_b \end{pmatrix}}_{x} \tag{2.63}$$

$x_{b|a}$ 的分布为

$$\mathbb{E}[x_{b|a}] = (-\Sigma_{ba}\Sigma_{aa}^{-1}\ I_n)|\begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix} = \mu_{b|a} \tag{2.64}$$

$$var[x_{b|a}] = (-\Sigma_{ba}\Sigma_{aa}^{-1}\ I_n)\begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}\begin{pmatrix} -\Sigma_{ba}\Sigma_{aa}^{-1}\ I_n \\ I \end{pmatrix} = \Sigma_{bb|a} \tag{2.65}$$

$$p(x_{b|a}) \sim \mathcal{N}(\mu_{b|a}, \Sigma_{bb|a}) \tag{2.66}$$

因为

$$x_b = x_{b|a} + \Sigma_{ba}\Sigma_{aa}^{-1}x_a \tag{2.67}$$

所以

$$\mathbb{E}[x_b|x_a] = \mu_{b|a} + \Sigma_{ba}\Sigma_{aa}^{-1}x_a \tag{2.68}$$

$$var[x_b|x_a] = var[x_{b|a}] = \Sigma_{bb|a} \tag{2.69}$$

$$p(x_b|x_a) \sim \mathcal{N}(\mu_{b|a}, \Sigma_{bb|a}) \tag{2.70}$$

高斯变量的贝叶斯定理

令边缘概率分布和条件概率分布的形式如下

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\mu, \Lambda^{-1}) \tag{2.71}$$

$$p(\boldsymbol{y}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{A}\boldsymbol{x} + b, \boldsymbol{L}^{-1}) \tag{2.72}$$

其中，μ, A 和 b 是控制均值的参数，Λ 和 L 是精度矩阵。如果 x 的维度为 M，y 的维度为 D，那么矩阵 A 的大小为 $D \times M$。

1. 求 $p(y)$。

$$y = Ax + b + \epsilon, \quad \epsilon \sim \mathcal{N}(0, L^{-1}) \tag{2.73}$$

$$\mathbb{E}[y] = \mathbb{E}(Ax + b) + \mathbb{E}(\epsilon) = A\mu + b \tag{2.74}$$

$$var[y] = var[Ax + b + \epsilon] = A\Lambda^{-1}A^T + L^{-1} \tag{2.75}$$

$$y \sim \mathcal{N}(A\mu + b, A\Lambda^{-1}A^T + L^{-1}) \tag{2.76}$$

2. 求 $p(x|y)$。

定义

$$z = \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.77}$$

$$z \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ A\mu + b \end{bmatrix}, \begin{bmatrix} \Lambda^{-1} & \triangle \\ \triangle^T & A\Lambda^{-1}A^T + L^{-1} \end{bmatrix}\right) \tag{2.78}$$

其中

$$\begin{aligned} \triangle &= Cov(x, y) \\ &= E[(x - E(x)) \cdot (y - E(y))^T] \\ &= E[(x - \mu)(y - A\mu - b)^T] \\ &= E[(x - \mu)(Ax + b + \epsilon - A\mu - b)^T] \\ &= E[(x - \mu)(Ax - A\mu)^T + (x - \mu)\epsilon^T] \\ &= E[(x - \mu)(Ax - A\mu)^T] + E[(x - \mu)\epsilon^T] \\ &= E[(x - \mu)(Ax - A\mu)^T] \\ &= E[(x - \mu)(x - \mu)^T] \cdot A^T \\ &= var[x] \cdot A^T \\ &= \Lambda^{-1}A^T \end{aligned} \tag{2.79}$$

所以

$$\mathbb{E}[x|y] = (\Lambda + A^T LA)^{-1}\{A^T L(y - b) + \Lambda\mu\} \tag{2.80}$$

$$cov[x|y] = (\Lambda + A^T LA)^{-1} \tag{2.81}$$

高斯分布的最大似然估计

给定一个数据集 $X = (x_1, x_2, \ldots, x_N)^T$，其中观测 $\{x_n\}$ 假定是独立地从多元高斯分布中抽取的。我们可以使用最大似然法估计分布的参数。

$$\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n \tag{2.82}$$

$$\Sigma_{ML} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{ML})(x_n - \mu_{ML})^T \tag{2.83}$$

$$\mathbb{E}[\mu_{ML}] = \mu \tag{2.84}$$

$$\mathbb{E}[\Sigma_{ML}] = \frac{N-1}{N}\Sigma \tag{2.85}$$

顺序估计

顺序的方法允许每次处理一个数据点，然后丢弃这个点。这对于在线应用很重要。并且当数据集相当大以至于一次处理所有数据点不可行的情况下，顺序方法也很重要。对于高斯分布

$$\begin{aligned}\mu_{ML}^{(N)} &= \frac{1}{N}\sum_{n=1}^{N} x_n \\ &= \frac{1}{N}x_N + \frac{1}{N}\sum_{n=1}^{N-1} x_n \\ &= \mu_{ML}^{(N-1)} + \frac{1}{N}(x_N - \mu_{ML}^{(N-1)})\end{aligned} \tag{2.86}$$

随着 N 的增加，后续数据点的贡献也会逐渐变小。我们不能总是能够使用这种方法推导出一个顺序的算法。因此我们要寻找一个更加通用的顺序学习的方法，这就引出了 Robbins-Monro 算法。考虑一对随机变量 θ 和 z，它们由一个联合概率分布 $p(z,\theta)$ 所控制。已知 θ 的条件下，z 的条件期望定义了一个确定的函数 $f(\theta)$，形式如下

$$f(\theta) \equiv \mathbb{E}[z|\theta] = \int z p(z|\theta) dz \tag{2.87}$$

通过这种方式定义的函数被称为回归函数 (regression function)。我们的目标是寻找根 θ^* 使得 $f(\theta^*) = 0$。如果我们有观测 z 和 θ 的一个大数据集，那么我们可以直接对回归函数建模，得到根的一个估计。但是假设我们每次观测到一个 z 的值，我们想找到一个对应的顺序估计方法来找到 θ^*。下面的解决这种问题的通用步骤由 Robbins and Monro 给出。我们假设 z 的条件方差是有穷的，因此

$$\mathbb{E}[(z-f)^2|\theta] < \infty \tag{2.88}$$

不失一般性，假设当 $\theta > \theta^*$ 时 $f(\theta) > 0$，当 $\theta < \theta^*$ 时 $f(\theta^*) < 0$。下式定义了一个根 θ^* 的顺序估计的序列

$$\theta^{(N)} = \theta^{(N-1)} - \alpha_{N-1} z(\theta^{(N-1)}) \tag{2.89}$$

$z(\theta^{(N)})$ 是当 θ 的取值为 $\theta^{(N)}$ 时 z 的观测值。系数 $\{\alpha_N\}$ 表示一个满足下列条件的正数序列

$$\lim_{N\to\infty} \alpha_N = 0 \tag{2.90}$$

$$\sum_{N=1}^{\infty} \alpha_N = \infty \tag{2.91}$$

$$\sum_{N=1}^{\infty} \alpha_N^2 < \infty \tag{2.92}$$

可以证明由公式给出的顺序估计确实以概率 1 收敛于根。第一个条件确保了后续的修正幅度会逐渐变小，从而这个过程可以收敛于一个极限值。第二个条件用来确保算法不会收

敛不到根的值。第三个条件保证了累计的噪声具有一个有限的方差，因此不会导致收敛失败。

高斯分布的贝叶斯推断

现在我们通过引入高斯分布中的参数的先验分布，介绍一种贝叶斯的方法。

1. σ^2 已知，估计 μ。
2. μ 已知，估计 σ^2。
3. μ 和 σ_2 都未知。

学生 t 分布

周期变量

混合高斯模型

虽然高斯分布有一些重要的分析性质，但是当它遇到实际数据集时，也会有巨大的局限性。通过将更基本的概率分布进行线性组合的这样的叠加方法，可以被形式化为概率模型，被称为混合模型 (mixture distributions)。通过使用足够多的高斯分布，并且调节它们的均值和方差以及线性组合的系数，几乎所有的连续概率密度都能够以任意的精度近似。考虑 K 个高斯概率密度的叠加，形式为

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \tag{2.93}$$

这被称为混合高斯 (mixture of Gaussians)。其中

$$\sum_{k=1}^{K} \pi_k = 1,\ 0 \leqslant \pi_k \leqslant 1 \tag{2.94}$$

2.4 指数族分布

指数族分布的成员有许多共同的重要性质，并且以某种程度的一般性下讨论这些性质是很有启发性的。参数为 η 的变量 x 的指数族分布定义为具有下面形式的概率分布的集合

$$p(x|\eta) = h(x)g(\eta)exp\{\eta^T u(x)\} \tag{2.95}$$

其中 x 可能是标量或者向量，可能是离散的或者是连续的。这里 η 被称为概率分布的自然参数 (natural parameters)，$u(x)$ 是 x 的某个函数。函数 $g(\eta)$ 可以被看成系数，它确保了概率分布是归一化的，因此满足

$$\int p(x|\eta)dx = 1 \tag{2.96}$$

下面看一些例子

1. 伯努利分布

$$\begin{aligned} p(x|\mu) &= Bern(x|\mu) = \mu^x(1-\mu)^{1-x} \\ &= exp\{x\ln\mu + (1-x)\ln(1-\mu)\} \\ &= (1-\mu)exp\left\{\ln\left(\frac{\mu}{1-\mu}\right)x\right\} \end{aligned} \tag{2.97}$$

与公式 2.95 比较，可以看出

$$\eta = \ln\left(\frac{\mu}{1-\mu}\right) \tag{2.98}$$

从中可以解出 η，得到 $\mu = \sigma(\eta)$，其中

$$\sigma(\eta) = \frac{1}{1+exp(-\eta)} \tag{2.99}$$

被称为 logistic sigmoid 函数。因此可以使用公式 2.95 给出的标准形式把伯努利分布写成下面的形式

$$p(x|\mu) = \sigma(-\eta)exp(\eta x) \tag{2.100}$$

2. 单一观测 x 的多项式分布

$$\begin{aligned} p(x|\mu) &= \prod_{k=1}^{M}\mu_k^{x_k} = exp\left\{\sum_{k=1}^{M}x_k\ln\mu_k\right\} \\ \mu(x) &= x \\ h(x) &= 1 \\ g(\eta) &= 1 \end{aligned} \tag{2.101}$$

注意参数 η_k 不是相互独立的，因为参数 μ_k 要满足下面的限制

$$\sum_{k=1}^{M}\mu_k = 1 \tag{2.102}$$

因此给定任意 M-1 个参数 μ_k，剩下的参数就固定了。使用这个限制，这种表达方式下多项式分布变成了

$$\begin{aligned} &exp\left\{\sum_{k=1}^{M}x_k\ln\mu_k\right\} \\ &= exp\left\{\sum_{k=1}^{M-1}x_k\ln\mu_k + \left(1-\sum_{k=1}^{M-1}x_k\right)\ln\left(1-\sum_{k=1}^{M-1}\mu_k\right)\right\} \\ &= exp\left\{\sum_{k=1}^{M-1}x_k\ln\left(\frac{\mu_k}{1-\sum_{j=1}^{M-1}\mu_k}\right) + \ln\left(1-\sum_{k=1}^{M-1}\mu_k\right)\right\} \end{aligned} \tag{2.103}$$

令

$$\ln\left(\frac{\mu_k}{1-\sum_j \mu_j}\right) = \eta_k \tag{2.104}$$

从中我们可以解出 μ_k。首先两侧对 k 求和，然后整理，回带，可得

$$\mu_k = \frac{exp(\eta_k)}{1+\sum_j exp(\eta_j)} \tag{2.105}$$

这被称为 softmax 函数，或者归一化指数 (normalized exponential)。在这个表达方式的形式下，多项式分布的形式为

$$p(x|\eta) = \left(1+\sum_{k=1}^{M-1} exp(\mu_k)\right)^{-1} exp(\mu^T x) \tag{2.106}$$

这是指数族分布的标准形式，其中参数向量 $\eta = (\eta_1, \ldots, \eta_{M-1}, 0)^T$。在这个指数族分布中

$$\mu(x) = x \tag{2.107}$$

$$h(x) = 1 \tag{2.108}$$

$$g(\eta) = \left(1+\sum_{k=1}^{M-1} exp(\eta_k)\right)^{-1} \tag{2.109}$$

3. 高斯分布

$$\begin{aligned} p(x|\mu,\sigma^2) &= \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \\ &= \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} exp\left\{-\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}\mu^2\right\} \end{aligned} \tag{2.110}$$

经过简单的推导后，它可以转化为公式 2.95 给出的标准指数族分布的形式，其中

$$\eta = \begin{pmatrix} \frac{\mu}{\sigma^2} \\ \frac{-1}{2\sigma^2} \end{pmatrix} \tag{2.111}$$

$$\mu(x) = \begin{pmatrix} x \\ x^2 \end{pmatrix} \tag{2.112}$$

$$h(x) = (2\pi)^{\frac{1}{2}} \tag{2.113}$$

$$g(\eta) = (-2\eta_2)^{\frac{1}{2}} exp\left(\frac{\eta_1^2}{4\eta_2^2}\right) \tag{2.114}$$

最大似然与充分统计量

用最大似然法估计公式 2.95 给出的一般形式的指数族分布的参数向量 μ 的问题。

$$\int h(x)g(\eta)exp\{\eta^T u(x)\} = 1 \tag{2.115}$$

对上式的两侧关于 μ 取梯度,我们有

$$\begin{aligned} &\nabla g(\eta) \underbrace{\int h(x)exp\{\eta^T u(x)\}dx}_{1/g(\eta)} + g(\eta) \int h(x)exp\{\eta^T u(x)\}u(x)dx = 0 \\ &\Rightarrow -\frac{1}{g(\eta)}\nabla g(\eta) = \int \underbrace{g(\eta)h(x)exp\{\eta^T u(x)\}}_{p(x|\eta)} u(x)dx = \mathbb{E}[\mu(x)] \\ &\Rightarrow -\nabla \ln g(\eta) = \mathbb{E}[u(x)] \end{aligned} \tag{2.116}$$

同理,$u(x)$ **的协方差,可以根据** $g(\eta)$ **的二阶导数表达,对于高阶矩的情形也类似**。因此,如果我们能够对一个来自指数族分布的概率分布进行归一化,那么我们总能够通过简单的求微分的方式找到它的矩。现在考虑一组独立同分布的数据 $X = \{x_1, \ldots, x_N\}$。对于这个数据集,似然函数为

$$p(X|\eta) = \left(\prod_{n=1}^{N} h(x_n)\right) g(\eta)^N exp\left\{\eta^T \sum_{n=1}^{N} u(x_n)\right\} \tag{2.117}$$

令 $\ln p(X|\eta)$ 关于 η 的导数等于零,我们可以得到最大似然估计 μ_{ML} 满足的条件

$$-\nabla \ln g(\eta) = \frac{1}{N}\sum_{n=1}^{N} u(x_n) \tag{2.118}$$

原则上可以通过这个方程来得到 μ_{ML}。我们看到最大似然估计的解只通过 $\sum_n u(x_n)$ 对数据产生依赖,因此这个量被称为分布的充分统计量 (sufficient statistic)。我们不需要存储整个数据集本身,只需要存储充分统计量的值即可。

共轭先验

对于指数族分布的任何成员,都存在一个共轭先验,可以写成下面的形式

$$p(\eta|\varkappa, v) = f(\varkappa, v)g(\eta)^v exp\{v\eta^T \varkappa\} \tag{2.119}$$

无信息先验

在许多情形下,我们可能对分布应该具有的形式几乎完全不知道。这时,我们可以寻找一种形式的先验分布,被称为无信息先验 (noninformative prior)。这种先验分布的目的是尽量对后验分布产生尽可能小的影响。有时被称为“让数据自己说话”。

1. 位置参数
2. 缩放参数

2.5 非参数化方法

使用一些非参数化方法进行概率密度估计。这种方法对概率分布的形式进行了很少的假设。

直方图法

标准的直方图简单地把 x 划分成不同的宽度为 $\triangle_i$ 的箱子，然后对落在第 i 个箱子中的 x 的观测数量 n_i 进行计数。为了把这种计数转换成归一化的概率密度，我们简单地把观测数量除以观测的总数 N，再除以箱子的宽度 $\triangle_i$，得到每个箱子的概率的值

$$p_i = \frac{n_i}{N\triangle_i} \tag{2.120}$$

在实际应用中，直方图方法对于快速地将一维或者二维的数据可视化很有用，但是并不适用于大多数概率密度估计的应用。一个明显的问题是估计的概率密度具有不连续性，这种不连续性是因为箱子的边缘造成的，而不是因为生成数据的概率分布本身的性质造成。直方图的另一个主要的局限性是维数放大。但是，概率密度估计的直方图方法确实告诉了我们两个重要的事情。

1. 为了估计在某个特定位置的概率密度，我们应该考虑位于那个点的某个领域内的数据点。
2. 为了获得好的结果，平滑参数的值既不能太大也不能太小。

假设观测服从 D 维空间的某个未知的概率密度分布 $p(x)$。把这个 D 维空间选择成欧几里德空间，并且我们想估计 $p(x)$ 的值。根据以前对于局部性的讨论，让我们考虑包含 x 的某个小区域 $\mathcal{R}$。这个区域的概率质量为

$$P = \int_{\mathcal{R}} p(x)dx \tag{2.121}$$

现在我们假设收集了服从 $p(x)$ 分布的 N 次观测。由于每个数据点都有一个落在区域 $\mathcal{R}$ 中的概率 P，因此位于区域 $\mathcal{R}$ 内部的数据点的总数 K 将服从二项分布

$$Bin(K|N,P) = \frac{N!}{K!(N-K)!}P^K(1-P)^{N-K} \tag{2.122}$$

落在区域内部的数据点的平均比例为 $\mathbb{E}[\frac{K}{N}] = P$。类似地，以此为均值的概率分布的方差为 $var[\frac{K}{N}] = \frac{P(1-P)}{N}$。对于大的 N 值，这个分布将会在均值附近产生尖峰，并且

$$K \simeq NP \tag{2.123}$$

$$P \simeq p(x)V \tag{2.124}$$

$$p(x) = \frac{K}{NV} \tag{2.125}$$

上式的成立依赖于两个相互矛盾的假设，即区域 $\mathcal{R}$ 要足够小，使得这个区域内的概率密度近似为常数，但是也要足够大，使得落在这个区域内的数据点的数量 K 能够足够让二项分布达到尖峰。

我们有两种方式利用这个结果。

1. 固定 K 然后从数据中确定 V 的值，这就是 K 近邻方法。
2. 固定 V 然后从数据中确定 K 的值，这就是核方法。

在极限 $N \to \infty$ 的情况下，如果 V 随着 N 而合适地收缩，并且 K 随着 N 增大，那么可以证明 K 近邻概率密度估计和核方法概率密度估计都会收敛到真实的概率密度。

核密度估计

把区域 $\mathcal{R}$ 取成以 x 为中心的小超立方体，为了统计落在这个区域内的数据点的数量 K，定义下面的函数比较方便

$$k(u) = \begin{cases} 1, & |u_i| \leqslant \frac{1}{2}, i = 1, \ldots, D \\ 0, & \text{其它情况} \end{cases} \tag{2.126}$$

这表示一个以原点为中心的单位立方体。函数 $k(u)$ 是核函数的一个例子，在这个问题中也被称为 Parzen 窗 (parzen window)。根据公式 2.126，如果数据点 x_n 位于以 x 为中心的边长为 h 的立方体中，那么量 $k(\frac{x-x_n}{h})$ 的值等于 1，否则它的值为 0. 于是

$$p(x) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h^D}k\left(\frac{x-x_n}{h}\right) \tag{2.127}$$

这个函数表述为以 N 个数据点 x_n 为中心的 N 个立方体。

核密度估计有人为带来的非连续性的问题。如果我们选择一个平滑的核函数，那么我们就可以得到一个更加光滑的模型。

$$p(x) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi h^2)^{\frac{D}{2}}}exp\left\{-\frac{\|x-x_n\|^2}{2h^2}\right\} \tag{2.128}$$

h 表示高斯分布的标准差。因此我们概率密度模型可以通过这种方式获得：令每个数据点都服从高斯分布，然后把数据集里的每个数据点的贡献相加，之后除以 N，使得概率密度正确地被归一化。

这种估计方法有一个很大的优点，即不需要进行“训练”阶段的计算，因为“训练”阶段只需要存储训练集即可。然而，这也是一个巨大的缺点，因为估计概率密度的计算代价随着数据集的规模线性增长。

近邻方法

核方法进行概率密度估计的一个困难之处是控制核宽度的参数 h 对于所有的核都是固定的。与之不同，考虑固定 K 的值然后使用数据来确定合适的 V 值。为了完成这一点，我们考虑一个以 x 为中心的小球体，然后我们想估计概率密度 $p(x)$。并且，我们允许球体的半径可以自由增长，直到它精确地包含 K 个数据点。这样，概率密度 $p(x)$ 的估计就由公式 2.126 给出，其中 V 等于最终球体的体积。这种方法被称为 K 近邻方法。

第 3 章 变分法

第 4 章 矩阵的性质

第 5 章 回归的线性模型

5.1 线性基函数模型

回归问题的最简单模型是输入变量的线性组合

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + \ldots w_D x_D$$

其中 $x = (x_1, \ldots, x_D)^T$。这通常被简单地称为**线性回归 (linear regression)**。这个模型的关键性质是它是参数 $w_0, \ldots, w_D$ 的一个线性函数。但是，它也是输入变量 x_i 的一个线性函数，这给模型带来了极大的局限性。因此，我们扩展模型的类别：将输入变量的固定的非线性函数进行线性组合，形式为

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{y=1}^{M-1} w_j \phi_j(\boldsymbol{x}) = \sum_{j=0}^{M-1} w_j \phi_j(\boldsymbol{x}) = \boldsymbol{w}^T \phi(\boldsymbol{x})$$

其中 $\phi_j(\boldsymbol{x})$ 被称为基函数 (basis function)。

1. 高斯基函数

$$\phi_j(x) = exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

2. sigmoid 基函数

$$\phi_j(x) = \frac{1}{1 + exp\left(\frac{x-\mu_j}{s}\right)}$$

3. 傅里叶基函数

最大似然与最小平方

最小化平方和误差函数可以看成高斯噪声模型的假设下的最大似然解。假设目标变量 t 由确定的函数 $y(x, w)$ 给出，这个函数被附加了高斯噪声，即

$$t = y(\boldsymbol{x}, \boldsymbol{w}) + \epsilon \tag{5.1}$$

其中 ϵ 是一个零均值的高斯随机变量，精度为 β。因此我们有

$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}(t|y(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1}) \tag{5.2}$$

最优的预测由目标变量的条件给出

$$\mathbb{E}[t|\boldsymbol{x}] = \int tp(t|\boldsymbol{x})dt = y(\boldsymbol{x}, \boldsymbol{w}) \tag{5.3}$$

现在考虑一个输入数据集 $X = \{x_1, \ldots, x_N\}$，对应的目标值为 $t_1, \ldots, t_N$。把目标向量 $\{t_n\}$ 组成一个列向量，记作 $\mathbf{t}$。这个变量的字体与多元目标值的一次观测 (记作 t) 不同。假设这些数据点是独立同分布的。那么可以得到下面的似然函数

$$p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\boldsymbol{w}^T\phi(x_n), \beta^{-1}) \tag{5.4}$$

注意，在有监督学习问题中 (例如回归问题和分类问题)，我们不是在寻找模型来对输入变量的概率分布建模。因此 x 总会出现在条件变量的位置上。为了保持记号的简洁性，不显示地写出 x。取似然函数的对数，我们有

$$\begin{aligned} \ln p(\boldsymbol{t}|\boldsymbol{w}, \beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\boldsymbol{w}^T\phi(x_n), \beta^{-1}) \\ &= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\boldsymbol{w}) \end{aligned} \tag{5.5}$$

其中平方和误差函数的定义为

$$\begin{aligned} E_D(\boldsymbol{w}) &= \frac{1}{2}\sum_{n=1}^{N}\{t_n - \boldsymbol{w}^T\phi(x_n)\}^2 \\ &= \frac{1}{2}(\boldsymbol{w}^T\phi(x_1) - t_1 \ \ldots \ \boldsymbol{w}^T\phi(x_N) - t_N)\begin{pmatrix} \boldsymbol{w}^T\phi(x_1) - t_1 \\ \vdots \\ \boldsymbol{w}^T\phi(x_N) - t_N \end{pmatrix} \\ &= \frac{1}{2}(\boldsymbol{w}^T(\phi(x_1) \ \ldots \ \phi(x_N)) - (t_1 \ \ldots \ t_N))\left(\begin{pmatrix} \boldsymbol{w}^T\phi(x_1) \\ \vdots \\ \boldsymbol{w}^T\phi(x_N) \end{pmatrix} - \begin{pmatrix} t_1 \\ \vdots \\ t_N \end{pmatrix}\right) \\ &= \frac{1}{2}(\boldsymbol{w}^T\Phi^T - \boldsymbol{t}^T)(\Phi\boldsymbol{w} - \boldsymbol{t}) \\ &= \frac{1}{2}(\boldsymbol{w}^T\Phi^T\Phi\boldsymbol{w} - 2\boldsymbol{w}^T\Phi^T\boldsymbol{t} + \boldsymbol{t}^T\boldsymbol{t}) \end{aligned} \tag{5.6}$$

使用最大似然方法确定 $\boldsymbol{w}$ 和 β。首先关于 $\boldsymbol{w}$ 求最大值。

$$\begin{aligned} \nabla \ln p(\boldsymbol{t}|\boldsymbol{w}, \beta) &= \beta\sum_{n=1}^{N}\{t_n - \boldsymbol{w}^T\phi(x_n)\}\phi(x_n)^T \\ &= \Phi^T\Phi\boldsymbol{w} - \Phi^T\boldsymbol{t} \end{aligned} \tag{5.7}$$

令梯度为零，可得

$$\boldsymbol{w} = (\Phi^T\Phi)^{-1}\Phi^T\boldsymbol{t} \tag{5.8}$$

这被称为最小平方问题的规范方程 (normal equation)。这里 Φ 是一个 $N \times M$ 的矩阵，被称为设计矩阵 (design matrix)，它的元素为 $\Phi_{nj} = \phi_j(x_n)$，即

$$\begin{pmatrix} \Phi_0(x_1) & \Phi_0(x_1) & \dots & \Phi_{M-1}(x_1) \\ \Phi_0(x_2) & \Phi_0(x_2) & \dots & \Phi_{M-1}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_0(x_N) & \Phi_0(x_N) & \dots & \Phi_{M-1}(x_N) \end{pmatrix} \tag{5.9}$$

关于噪声精度参数 β 最大化似然函数，结果为

$$\frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{n=1}^{N} \{t_n - \boldsymbol{w}_{ML}^T \phi(x_n)\}^2 \tag{5.10}$$

因此我们看到噪声精度的倒数由目标值在回归函数周围的残留方差 (residual variance) 给出。

最小平方的几何描述

顺序学习

最大似然解的求解过程涉及到一次处理整个数据集。这种批处理技术对于大规模数据集来说计算量相当大。如果数据集充分大，那么使用顺序算法 (也被称为在线算法) 可能更有价值。顺序算法中，每次只考虑一个数据点，模型的参数在每观测到一个数据点之后进行更新。顺序学习也适用于实时的应用。在实时应用中，数据观测以一个连续的流的方式持续到达，我们必须在观测到所有数据之前就做出预测。

我们可以获得一个顺序学习的算法通过考虑随机梯度下降 (stochastic gradient descent) 也被称为顺序梯度下降 (sequential gradient descent) 的方法。

正则化最小平方

为误差函数添加正则化项的思想来控制过拟合，因此需要最小化的总的误差函数的形式为

$$E_D(\boldsymbol{w}) + \lambda E_W(\boldsymbol{w}) \tag{5.11}$$

L1 正则化引起稀疏解的多种解释。

1. 用图解释：

   L2 正则相当于用圆去逼近目标，而 L1 正则相当于用菱形去逼近目标，所以更容易引起交战在坐标轴上即得到稀疏解。

2. 从导数角度解释：

   L2 正则无法将目标函数的极值点拉拢到稀疏解上，而 L1 正则因为 L1 导函数的特殊性从而可以在一定范围内将极值点直接拉拢到稀疏解上。

3. 从先验概率分布角度解释：

L2 正则相当于假设参数是服从高斯分布的，而 L1 正则相当于假设了参数是服从拉普拉斯分布的，自然拉普拉斯分布比高斯分布更集中在 0 这个点上。

多个输出

在某些应用中，我们可能想预测 $K > 1$ 个目标变量。我们把这些目标变量聚焦起来，记作目标向量 $\boldsymbol{t}$。这个问题可以这样解决：对于 $\boldsymbol{t}$ 的每个分量，引入一个不同的基函数集合，从而变成了多个独立的回归问题。但是，一个更有趣的并且更常用的方法是对目标向量的所有分量使用一组相同的基函数来建模，即

$$\boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{W}^T \boldsymbol{\phi}(\boldsymbol{x}) \tag{5.12}$$

其中 $\boldsymbol{y}$ 是一个 K 维列向量，$\boldsymbol{W}$ 是一个 $M \times K$ 的参数矩阵，$\boldsymbol{\phi}(\boldsymbol{x})$ 是一个 M 维列向量，每个元素为 $\phi_j(\boldsymbol{x})$。

5.2 偏置-方差分解

目前为止，我们对于回归的线性模型的讨论中，我们假定了基函数的形式和数量都是固定的。如果使用有限规模的数据集来训练复杂的模型，那么使用最大似然法，或者等价地使用最小平方法，会导致严重的过拟合问题。正如前面所说，过拟合现象确实是最大似然方法的一个不好的性质。但是当我们在使用贝叶斯方法对参数进行求和或者积分时，过拟合现象不会出现。从贝叶斯观点讨论模型的复杂度之前，从频率学家的观点考虑一下模型的复杂度的问题——偏置-方差折中 (bias-variance trade-off)。

最优的预测 (变分法可求出) 由条件期望 (记作 $h(x)$) 给出，即

$$h(x) = \mathbb{E}[t|x] = \int t p(t|x) dt \tag{5.13}$$

平方损失函数的期望可以写成

$$\begin{aligned} \mathbb{E}[L] &= \iint \{y(x) - h(x)\}^2 p(x,t) dx dt \\ &= \int \{y(x) - h(x)\}^2 p(x) dx + \iint \{h(x) - t\}^2 p(x,t) dx dt \end{aligned} \tag{5.14}$$

考虑第一项的被积函数，对于一个特定的数据集 D，它的形式为

$$\{y(x; D) - h(x)\}^2 \tag{5.15}$$

由于这个量与特定的数据集 D 相关，因此我们对所有的数据集取平均。如果我们在括号

内加上然后减去 $\mathbb{E}_D[y(x;D)]$，然后展开，我们有

$$\begin{aligned}&\{y(x;D)-\mathbb{E}_D[y(x;D)]+\mathbb{E}_D[y(x;D)]-h(x)\}^2\\&\quad=\{y(x;D)-\mathbb{E}_D[y(x;D)]\}^2\\&\quad+\{\mathbb{E}_D[y(x;D)]-h(x)\}^2\\&\quad+2\{y(x;D)-\mathbb{E}_D[y(x;D)]\}\{\mathbb{E}_D[y(x;D)]-h(x)\}\end{aligned}\tag{5.16}$$

现在关于 D 求期望，然后注意到**最后一项等于零**，可得

$$\begin{aligned}&\mathbb{E}_D[\{y(x;D)-h(x)\}^2]\\&\quad=\underbrace{\mathbb{E}_D[y(x;D)]-h(x)\}^2}_{(\text{偏置})^2}+\underbrace{\mathbb{E}_D[\{y(x;D)-\mathbb{E}_D[y(x;D)]\}^2]}_{\text{方差}}\end{aligned}\tag{5.17}$$

将式 5.17 代入式 5.14 中，就得到了对于期望平方损失的分解

$$\text{期望损失}=\text{偏置}^2+\text{方差}+\text{噪声}\tag{5.18}$$

其中

$$\text{偏置}^2=\int\{\mathbb{E}_D[y(x;D)]-h(x)\}^2p(x)dx\tag{5.19}$$

$$\text{方差}=\int\mathbb{E}_D[\{y(x;D)-\mathbb{E}_D[y(x;D)]\}^2]p(x)dx\tag{5.20}$$

$$\text{噪声}=\iint\{h(x)-t\}^2p(x,t)dxdt\tag{5.21}$$

我们的目标是最小化期望损失，它可以分解为 (平方) 偏置、方差和一个常数噪声项的和。对于非常灵活的模型来说，偏置较小，方差较大。对于相对固定的模型来说，偏置较大，方差较小。有着最优预测能力的模型是在偏置和方差之前取得最优的平衡的模型。

5.3 贝叶斯线性回归

使用最大似然方法设置线性回归模型的参数时，由基函数的数量控制的模型的复杂度需要根据数据集的规模进行调整。为对数似然函数增加一个正则化项意味着模型的复杂度可以通过正则化系数的值进行控制，虽然基函数的数量和形式的选择仍然对于确定模型的整体行为十分重要。

这就产生了对于特定的应用确定合适的模型复杂度的问题。这个问题不能简单地通过最大化似然函数来确定，因为这总会产生过于复杂的模型和过拟合现象。独立的额外数据能够用来确定模型的复杂度，但这需要较大的计算量，并且浪费了有价值的数据。因此我们转而考虑线性回归的贝叶斯方法，这会避免最大似然的过拟合问题，也会引出使用训练数据本身确定模型复杂度的自动化方法。

参数分布

关于线性拟合的贝叶斯方法的讨论，我们首先引入模型参数 w 的先验概率分布。把噪声精度参数 β 当做已知常数。对应的共轭先验是高斯分布。

$$p(w) \equiv \mathcal{N}(w|\boldsymbol{m}_0, \boldsymbol{S}_0) \tag{5.22}$$

均值为 $\boldsymbol{m}_0$ 协方差为 $\boldsymbol{S}_0$ 后验分布

$$p(w|\boldsymbol{t}) = \mathcal{N}(w|\boldsymbol{m}_N, \boldsymbol{S}_N) \tag{5.23}$$

其中，

$$\boldsymbol{m}_N = \boldsymbol{S}_N(\boldsymbol{S}_0^{-1} + \beta\Phi^T\boldsymbol{t}) \tag{5.24}$$

$$\boldsymbol{S}_N^{-1} = \boldsymbol{S}_0^{-1} + \beta\Phi^T\Phi \tag{5.25}$$

为了简化起见，考虑高斯先验的一个特定的形式——零均值各向同性高斯分布。这个分布由一个精度参数 α 控制，即

$$p(\boldsymbol{w}|\alpha) = \mathcal{N}(\boldsymbol{w}|0, \alpha^{-1}I) \tag{5.26}$$

后验概率分布的对数由对数似然函数与先验的对数求和的方式得到。它是 w 的函数，形式为

$$\ln p(\boldsymbol{w}|\boldsymbol{t}) - \frac{\beta}{2}\sum_{n=1}^{N}\{t_n - \boldsymbol{w}^T\phi(x_n)\}^2 - \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w} + \text{常数} \tag{5.27}$$

于是，后验分布关于 $\boldsymbol{w}$ 的最大化等价于对平方和误差函数加上一个二次正则项进行最小化。

预测分布

在实际应用中，我们通常感兴趣的不是 $\boldsymbol{w}$ 本身的值，而是对于新的 $\boldsymbol{x}$ 值预测出 t 的值。这需要计算出预测分布 (predictive distribution)，定义为

$$p(t|\boldsymbol{t}, \alpha, \beta) = \int p(t|\boldsymbol{w}, \beta)p(\boldsymbol{w}|\boldsymbol{t}, \alpha, \beta)d\boldsymbol{w} \tag{5.28}$$

其中 $\boldsymbol{t}$ 是训练数据的目标变量的值组成的向量。并且，为了简化记号，我们在右侧省略了条件概率中出现的输入向量。预测分布的形式为

$$p(t|\boldsymbol{x}, \boldsymbol{t}, \alpha, \beta) = \mathcal{N}(t|\boldsymbol{m}_N^T\phi(\boldsymbol{x}), \sigma_N^2(\boldsymbol{x})) \tag{5.29}$$

其中预测分布的方差 $\sigma_N^2(\boldsymbol{x})$ 为

$$\sigma_N^2(\boldsymbol{x}) = \frac{1}{\beta} + \phi(\boldsymbol{x})^T \boldsymbol{S}_N \phi(\boldsymbol{x}) \tag{5.30}$$

公式 5.30 的第一项表示数据中的噪声，而第二项反映了与参数 $\boldsymbol{w}$ 关联的不确定性。由于噪声和 $\boldsymbol{w}$ 的分布是相互独立的高斯分布，因此它们的值是可以相加的。注意，当额外的数据点被观测到的时候，后验概率分布会变窄。从而可以证明出 $\sigma_{N+1}^2(\boldsymbol{x}) \leqslant \sigma_N^2(\boldsymbol{x})$ 在极限 $N \to \infty$ 的情况下，公式 5.30 的第二项趋于零，从而预测分布的方差只与参数 β 控制的具有可加性的噪声有关。

等价核

把公式 5.24 代入线性基函数模型中，可以写成下面的形式

$$\begin{aligned} y(\boldsymbol{x}, \boldsymbol{m}_N) &= \boldsymbol{m}_N^T \phi(\boldsymbol{x}) \\ &= \beta \phi(\boldsymbol{x}^T \boldsymbol{S}_N \Phi^T \boldsymbol{t}) \\ &= \sum_{n=1}^{N} \underline{\beta \phi(\boldsymbol{x})^T \boldsymbol{S}_N \phi(\boldsymbol{x}_n)} t_n \\ &= \sum_{n=1}^{N} k(\boldsymbol{x}, \boldsymbol{x}_n) t_n \end{aligned} \tag{5.31}$$

可以看成在点 $\boldsymbol{x}$ 处的预测均值由训练集目标变量 t_n 的线性组合给出。函数 $k(x, x_n)$ 被称为平滑矩阵 (smoother matrix) 或者等价核 (equivalent kernel)。像这样的回归函数，通过对训练集里目标值进行线性组合做预测，被称为线性平滑 (linear smoother)。核函数 $k(x, x')$ 给出了 x 与 x' 的函数关系。可以看到，x 处的预测分布的均值 $y(x, \boldsymbol{m}_N)$ 可以通过对目标值加权组合的方式获得。距离 x 较近的数据点可以赋一个较高的权值，而距离 x 较远的数据点可以赋一个较低的权值。这种局部性不仅对于局部的高斯基函数成立，对于非局部的多项式基函数和 sigmoid 基函数也成立。

考虑 $y(\boldsymbol{x})$ 和 $y(\boldsymbol{x}')$ 的协方差

$$\begin{aligned} cov[y(\boldsymbol{x}), y(\boldsymbol{x}')] &= cov[\phi(\boldsymbol{x})^T \boldsymbol{w}, \boldsymbol{w}^T \phi(\boldsymbol{x}')] \\ &= \phi(\boldsymbol{x})^T \boldsymbol{S}_N \phi(\boldsymbol{x}') \\ &= \beta^{-1} k(\boldsymbol{x}, \boldsymbol{x}') \end{aligned} \tag{5.32}$$

根据等价核的形式，我们可以看到在附近的点处的预测均值相关性较高，而对于距离较远的点处，相关性就较低。

用核函数表示线性回归给出了解决回归问题的另一种方法。我们不引入一组基函数 (它隐式地定义了一个等价的核)，而是直接定义一个局部的核函数，然后在给定观测数据集的条件下，使用这个核函数对新的输入变量 $\boldsymbol{x}$ 做预测。这就引入了用于回归问题 (以及分类问题) 的一个很实用的框架，被称为高斯过程 (Gaussian process)。

5.4 贝叶斯模型比较

前面介绍了使用交叉验证的方法，来设置正则化参数的值，或者从多个模型中选择合适的一个。这里，从贝叶斯的角度考虑模型选择的问题。模型比较的贝叶斯观点仅仅涉及到使用概率来表示模型选择的不确定性，以及恰当地使用概率的加和规则和乘积规则。假设我们想比较 L 个模型 $\{M_i\}, i = 1, 2, \ldots, L$。这里，一个模型指的是观测数据 D 上的概率分布。我们会假设数据是由这些模型中的一个生成的，但是我们不知道究竟是哪一个。我们的不确定性通过先验概率分布 $p(M_i)$ 表示。给定一个数据集 D，我们想估计后验分布

$$p(M_i|D) \propto p(M_i)p(D|M_i) \tag{5.33}$$

先验分布让我们能够表达不同模型之间的优先级。**我们简单的假设所有的模型都有相同的先验概率**。模型证据 (model evidence)$p(D|M_i)$，表达了数据展现出的不同模型的优先级。模型证据有时也被称为边缘似然 (marginal likelihood)，因为它可以被看做在模型空间中的似然函数，在这个空间中参数已经被求和或者积分。两个模型的模型证据的比值 $\frac{p(D|M_i)}{p(D|M_j)}$ 被称为贝叶斯因子 (Bayes factor)。

一旦我们知道了模型上的后验概率分布，那么根据概率的加和规则与乘积规则，预测分布为

$$p(t|\boldsymbol{x}, D) = \sum_{i=1}^{L} p(t|\boldsymbol{x}, M_i, D)p(M_i|D) \tag{5.34}$$

这是混合分布 (mixture distribution) 的一个例子。这个公式中，整体的预测分布由下面的方式获得：对各个模型的预测分布 $p(t|\boldsymbol{x}, M_i, D)$ 求加权平均，权值为这些模型的后验概率 $p(M_i|D)$。

对模型求平均的一个简单的近似是使用最可能的一个模型做预测。这被称为模型选择 (model selection)。

对于一个由参数 $\boldsymbol{w}$ 控制的模型，根据概率的加和规则和乘积规则，模型证据为

$$p(D|M_i) = \int p(D|\boldsymbol{w}, M_i)p(\boldsymbol{w}|M_i)d\boldsymbol{w} \tag{5.35}$$

从取样的角度来看，边缘似然函数可以被看成从一个模型中生成数据集 D 的概率，这个模型的参数是从先验分布中随机取样的。同时，注意到模型证据恰好就是在估计参数的后验分布时出现在贝叶斯定理的分母中的归一化项，因为

$$p(\boldsymbol{w}|D, M_i) = \frac{p(D|\boldsymbol{w}, M_i)p(\boldsymbol{w}|M_i)}{p(D|M_i)} \tag{5.36}$$

对参数的积分进行一个简单的近似。假设后验分布在最大似然值 w_{MAP} 附近是一个尖峰，宽度为 $\triangle w_{后验}$，那么可以用被积函数的值乘以尖峰的宽度来近似这个积分。进一步假设

先验分布是平的，宽度为 $\triangle w_{先验}$，即 $p(w) = \frac{1}{\triangle w_{先验}}$，那么我们有

$$p(D) = \int p(D|w)p(w)dw \simeq p(D|w_{MAP})\frac{\triangle w_{后验}}{\triangle w_{先验}} \tag{5.37}$$

取对数可得

$$\ln p(D) \simeq \ln p(D|w_{MAP}) + \ln\left(\frac{\triangle w_{后验}}{\triangle w_{先验}}\right) \tag{5.38}$$

第一项表示拟合由最可能参数给出的数据。对于平的先验分布来说，这对应于对数似然。第二项用于根据模型的复杂度来惩罚模型。

对于一个有 M 个参数的模型，我们可以对每个参数进行类似的近似。假设所有的参数的 $\frac{\triangle w_{后验}}{\triangle w_{先验}}$ 都相同，我们有

$$\ln p(D) \simeq \ln p(D|w_{MAP}) + M\ln\left(\frac{\triangle w_{后验}}{\triangle w_{先验}}\right) \tag{5.39}$$

因此，在这种非常简单的近似下，复杂度惩罚项的大小随着模型中可调节参数 M 的数量线性增加。随着我们增加模型的复杂度，第一项通常会增大，因为一个更加复杂的模型能够更好地拟合数据，而第二项会减小，因为它依赖于 M。由最大模型证据确定的最优的模型复杂度需要在这两个相互竞争的项之间进行折中。

5.5 证据近似

在处理线性基函数模型的纯粹的贝叶斯方法中，我们会引入超参数 α 和 β 的先验分布，然后通过对超参数以及参数 w 求积分的方式做预测。但是，虽然我们可以解析地求出对 w 的积分或者求出对超参数的积分，但是对所有这些变量完整地求积分是没有解析解的。这里讨论一种近似方法。这种方法中，首先对参数 w 求积分，得到边缘似然函数 (marginal likelihood function)，然后通过最大化边缘似然函数，确定超参数的值。这个框架在统计学的文献中被称为经验贝叶斯，或者被称为第二类最大似然，或者被称为推广的最大似然。在机器学习的文献中，这种方法也被称为证据近似 (evidence approximation)。

如果引入 α 和 β 上的超先验分布，那么预测分布可以通过对 w, α, β 求积分的方法得到

$$p(t|\boldsymbol{t}) = \iiint p(t|\boldsymbol{w},\beta)p(\boldsymbol{w}|\boldsymbol{t},\alpha,\beta)p(\alpha,\beta|\boldsymbol{t})d\boldsymbol{w}\,d\alpha\,d\beta \tag{5.40}$$

如果后验分布 $p(\alpha,\beta|\boldsymbol{t})$ 在 $\hat{\alpha}$ 和 $\hat{\beta}$ 附近有尖峰，那么预测分布可以通过对 w 积分的方式简单地得到，其中 α 和 β 被固定为 $\hat{\alpha}$ 和 $\hat{\beta}$

$$p(t|\boldsymbol{t}) \simeq p(t|\boldsymbol{t},\hat{\alpha},\hat{\beta}) = \int p(t|\boldsymbol{w},\hat{\beta})p(\boldsymbol{w}|\boldsymbol{t},\hat{\alpha},\hat{\beta})d\boldsymbol{w} \tag{5.41}$$

根据贝叶斯定理，α 和 β 的后验分布为

$$p(\alpha,\beta|\boldsymbol{t}) \propto p(\boldsymbol{t}|\alpha,\beta)p(\alpha,\beta) \tag{5.42}$$

如果先验分布相对比较平，那么在证据框架中，$\hat{\alpha}$ 和 $\hat{\beta}$ 可以通过最大化边缘似然函数 $p(\boldsymbol{t}|\alpha,\beta)$ 来获得。

5.6 固定基函数的局限性

线性模型有一些重要的局限性，这使得我们要转而关注更加复杂的模型，例如支持向量机和神经网络。困难的产生主要是因为我们假设了基函数在观测到任何数据之前就被固定下来，而这正是维度灾难问题的一个表现形式。结果，基函数的数量随着输入空间的维度 D 迅速增长，通常是指数方式的增长。

幸运的是，真实数据集有两个性质，可以帮助我们缓解这个问题。

1. 数据向量 $\{x_n\}$ 通常位于一个非线性流形内部。由于输入变量之间的相关性，这个流形本身的维度小于输入空间的维度。如果我们使用局部基函数，那么我们可以让基函数只分布在输入空间中包含数据的区域。这种方法被用在径向基函数网络中，也被用在支持向量机和相关向量机当中。神经网络模型使用可调节的基函数，这些基函数有着 sigmoid 非线性的性质。神经网络可以通过调节参数，使得在输入空间的区域中基函数会按照数据流形发生变化。
2. 目标变量可能只依赖于数据流形中的少量可能的方向。利用这个性质，神经网络可以通过选择输入空间中基函数产生响应的方向。

第 6 章 分类的线性模型

分类的目标是将输入变量 x 分到 K 个离散的类别 C_k 中的某一类。所谓分类线性模型，是指决策面是输入向量 x 的线性函数，因此被定义为 D 维输入空间中的 $(D-1)$ 维超平面。

6.1 判别函数

判别函数是一个以向量 x 为输入，把它分配到 K 个类别中的某一个类别（记作 C_k）的函数。

二分类

线性判别函数的最简单的形式是输入向量的线性函数，即

$$y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0 \tag{6.1}$$

其中 $\boldsymbol{w}$ 被称为权向量 (weight vector)，w_0 被称为偏置 (bias)。对应的决策边界由 $y(\boldsymbol{x}) = 0$ 确定。对于一个输入向量 $\boldsymbol{x}$，如果 $y(\boldsymbol{x}) \geqslant 0$，那么它被分到 C_1 中，否则被分到 C_2 中。向量 $\boldsymbol{w}$ 与决策面内的任何向量都正交，从而 w 确定了决策面的方向。任意一点 x 到决策面的距离为

$$r = \frac{y(\boldsymbol{x})}{\|\boldsymbol{w}\|} \tag{6.2}$$

多分类

考虑把线性判别函数推广到 $K > 2$ 个类别。

1. 1 对其他。使用 K-1 个分类器，每个分类器用来解决一个二分类问题，把属于类别 C_k 和不属于那个类别的点分开。
2. 1 对 1。引入 $\frac{K(K-1)}{2}$ 个二元判别函数，对一对类别都设置一个判别函数。

上述两种方法都会产生输入空间无法分类的区域。通过引入 K 类判别函数可以避免这些问题。K 类判别函数由 K 个线性函数组成，形式为

$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T\boldsymbol{x} + w_{k0} \tag{6.3}$$

对于点 $\boldsymbol{x}$，如果所有的 $j \neq k$ 都有 $y_k(\boldsymbol{x}) > y_j(\boldsymbol{x})$，那么就把它分到 C_k。于是类别 C_k 和 C_j 之间的决策面为 $y_k(\boldsymbol{x}) = y_j(\boldsymbol{x})$，并且对应于一个 $(D-1)$ 维超平面，形式为

$$(\boldsymbol{w}_k - \boldsymbol{w}_j)^T\boldsymbol{x} + (w_{k0} - w_{j0}) = 0 \tag{6.4}$$

这样的判别函数的决策区域总是单连通的，并且是凸的。现在介绍三种学习线性判别函数的参数的方法，即基于最小平方的方法、Fisher 线性判别函数，以及感知器算法。

用于分类的最小平方方法

最小平方误差函数的最小化产生了参数值的简单的解析解。使用最小平方方法的一个理由是它在给定输入向量的情况下，近似了目标值的条件期望 $\mathbb{E}[t|\boldsymbol{x}]$。对于二元表示方法，条件期望由后验类概率向量给出。但不幸的是，这些概率通常很难近似。事实上，近似的过程有可能产生位于区间 $(0,1)$ 之外的值，这是因为线性模型的灵活性很受限。

使用向量记号，我们可以很容易地把这量聚焦在一起表示，即

$$\boldsymbol{y}(\boldsymbol{x}) = \tilde{W}^T\tilde{\boldsymbol{x}} = \begin{pmatrix} w_{1,0} & w_{2,0} & \cdots & w_{K,0} \\ w_{1,1} & w_{2,1} & \cdots & w_{K,1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1,D} & w_{2,D} & \cdots & w_{K,D} \end{pmatrix}^T \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_K \end{pmatrix} \tag{6.5}$$

其中 $\tilde{W}$ 是一个矩阵，第 k 列由 $D+1$ 维向量 $\tilde{\boldsymbol{w}}_k = (w_{k0}, \boldsymbol{w}_k^T)^T$ 组成，$\tilde{\boldsymbol{x}}$ 是对应的增广输入向量 $(1, \boldsymbol{x}^T)^T$，它带有一个虚输入 $x_0 = 1$。

通过最小化平方和误差函数来确定参数矩阵 $\tilde{W}$，考虑一个训练数据集 $\boldsymbol{x}_n, \boldsymbol{t}_n$，其中 $n = 1, \ldots, N$，然后定义一个矩阵 $\boldsymbol{T}$，它的第 n 行是向量 $\boldsymbol{t}_n^T$。还定义了一个矩阵 $\tilde{\boldsymbol{x}}_n^T$。这样，平方和误差函数可以写成

$$E_D(\tilde{W}) = \frac{1}{2}Tr\{(\tilde{X}\tilde{W} - \tilde{\boldsymbol{T}})^T(\tilde{X}\tilde{W} - \boldsymbol{T})\} \tag{6.6}$$

令上式关于 $\tilde{W}$ 的导数等于零，整理，可以得到 $\tilde{W}$ 的解，形式为

$$\tilde{W} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T\boldsymbol{T} = \tilde{X}^{\dagger}\boldsymbol{T} \tag{6.7}$$

这样，我们得到了判别函数，形式为

$$y(\boldsymbol{x}) = \tilde{W}^T\tilde{\boldsymbol{x}} = \boldsymbol{T}^T(\tilde{X}^{\dagger})^T\tilde{\boldsymbol{x}} \tag{6.8}$$

最小平方方法对于判别函数的参数给出了精确的解析解。但是，最小平方解对于离群点缺少鲁棒性。最小平方方法对应于高斯条件分布假设下的最大似然法，而二值目标向量的概率分布显然不是高斯分布。通过使用更恰当的概率模型，我们会得到性质比最小平方方法更好的分类方法。

Fisher 线性判别函数

Fisher 提出的思想是最大化一个函数，这个函数能够让类均值的投影分开得较大，同时让每个类别内部的方差较小，从而最小人化了类别的重叠。

首先考虑一个二分类问题，这个问题中有 C_1 类的 N_1 个点，以及 C_2 类的 N_2 个点。因此两类的均值向量为

$$\boldsymbol{m}_1 = \frac{1}{N_1}\sum_{n\in C_1}\boldsymbol{x}_n, \qquad \boldsymbol{m}_2 = \frac{1}{N_2}\sum_{n\in C_2}\boldsymbol{x}_n \tag{6.9}$$

如果投影到 $\boldsymbol{w}$ 上，那么最简单的度量类别之间分开程度的方式就是类别均值投影之后的距离。这说明，我们可以选择 $\boldsymbol{w}$ 使得下式取得最大值

$$m_2 - m_1 = \boldsymbol{w}^T(\boldsymbol{m}_2 - \boldsymbol{m}_1) \tag{6.10}$$

其中

$$m_k = \boldsymbol{w}^T\boldsymbol{m}_k \tag{6.11}$$

是来自类别 C_k 的投影数据的均值。但是，通过增大 $\boldsymbol{w}$，这个表达式可以任意大。为了解决这个问题，我们可以将 $\boldsymbol{w}$ 限制为单位长度，即 $\sum_i w_i^2 = 1$。使用拉格朗日乘数法来进行有限制条件的最大化问题的求解，我们可以发现 $\boldsymbol{w} \propto (\boldsymbol{m}_2 - \boldsymbol{m}_1)$。

来自类别 C_k 的数据经过变换后的类内方差为

$$s_k^2 = \sum_{n\in C_k}(\boldsymbol{w}^T\boldsymbol{x}_n - m_k)^2 \tag{6.12}$$

可以把整个数据集的总的方差定义为 $s_1^2 + s_2^2$。Fisher 准则根据类间方差和类内方差的比值定义，即

$$J(\boldsymbol{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \tag{6.13}$$

利用公式 6.9, 6.11

$$\begin{aligned}
(m_2 - m_1)^2 &= \left(\frac{1}{N_2}\sum_{n\in C_2}\boldsymbol{w}^T\boldsymbol{x}_n - \frac{1}{N_1}\sum_{n\in C_1}\boldsymbol{w}^T\boldsymbol{x}_n\right)^2 \\
&= \left(\boldsymbol{w}^T(\bar{x}_{C_1} - \bar{x}_{C_2})\right)^2 \\
&= \boldsymbol{w}^T(\bar{x}_{C_1} - \bar{x}_{C_2})(\bar{x}_{C_1} - \bar{x}_{C_2})^T\boldsymbol{w} \\
&= \boldsymbol{w}^T S_B \boldsymbol{w} \\
s_1^2 &= \frac{1}{N}\sum_{n\in C_1}(\boldsymbol{w}^T\boldsymbol{x}_n - \frac{1}{N}\sum_{n\in C_1}\boldsymbol{w}^T\boldsymbol{x}_n)(\boldsymbol{w}^T\boldsymbol{x}_n - \frac{1}{N}\sum_{n\in C_1}\boldsymbol{w}^T\boldsymbol{x}_n)^T \\
&= \frac{1}{N}\sum_{n\in C_1}\boldsymbol{w}^T(\boldsymbol{x}_n - \bar{x}_{C_1})(\boldsymbol{x}_n - \bar{x}_{C_1})^T\boldsymbol{w} \\
&= \boldsymbol{w}^T S_1 \boldsymbol{w} \\
s_1^2 + s_2^2 &= \boldsymbol{w}^T(S_1 + S_2)\boldsymbol{w} \\
&= \boldsymbol{w}^T S_W \boldsymbol{w}
\end{aligned} \tag{6.14}$$

显式地表达出 $J(\boldsymbol{w})$ 对 $\boldsymbol{w}$ 的依赖

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}} \tag{6.15}$$

其中 S_B 是类间 (between-class) 协方差矩阵，S_W 被称为类内 (within-class) 协方差矩阵。对该式关于 $\boldsymbol{w}$ 求导，并令其为零，有

$$\begin{aligned}
\frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} &= \frac{\partial[\boldsymbol{w}^T S_B \boldsymbol{w} \cdot (\boldsymbol{w}^T S_W \boldsymbol{w})^{-1}]}{\partial \boldsymbol{w}} \\
&= 2S_B \boldsymbol{w}(\boldsymbol{w}^T S_W \boldsymbol{w})^{-1} - 2\boldsymbol{w}^T S_B \boldsymbol{w} \cdot (\boldsymbol{w}^T S_W \boldsymbol{w})^{-2} \cdot S_W \boldsymbol{w} \\
&= 0 \\
&\Rightarrow S_B \boldsymbol{w} \cdot \boldsymbol{w}^T S_W \boldsymbol{w} = \boldsymbol{w}^T S_B \boldsymbol{w} \cdot S_W \boldsymbol{w} \\
&\Rightarrow S_W \boldsymbol{w} = \frac{\boldsymbol{w}^T S_W \boldsymbol{w}}{\boldsymbol{w}^T S_B \boldsymbol{w}} \cdot S_B \boldsymbol{w} \\
&\Rightarrow \boldsymbol{w} \propto S_W^{-1} S_B \boldsymbol{w} = S_W^{-1}(\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T \boldsymbol{w}
\end{aligned} \tag{6.16}$$

我们看到 $S_B \boldsymbol{w}$ 总是在 $\boldsymbol{m}_2 - \boldsymbol{m}_1$ 的方向上。更重要的是，我们不关心 $\boldsymbol{w}$ 的大小，只关心它的方向，因此

$$\boldsymbol{w} \propto S_W^{-1}(\boldsymbol{m}_2 - \boldsymbol{m}_1) \tag{6.17}$$

公式 6.17 的结果被称为 Fisher 线性判别函数 (Fisher linear discriminant)。如果类内协方差是各向同性的，从而 S_W 正比于单位矩阵，那么我们看到 $\boldsymbol{w}$ 正比于类均值的差。

与最小平方的关系

最小平方方法确定线性判别函数的目标是使模型的预测尽可能地与目标值接近。相反，Fisher 判别准则的目标是使输出空间的类别有最大的区分度。对于二分类问题，Fisher 准则可以看成最小平方的一个特例。

多分类的 Fisher 判别函数

感知器算法

线性判别模型的另一个例子是 Rosenblatt 提出的感知器算法。它对应于一个二分类的模型，这个模型中，输入向量 $\boldsymbol{x}$ 首先使用一个固定的非线性变换得到一个特征向量 $\phi(\boldsymbol{x})$，这个特征向量然后被用于构造一个一般的线性模型，形式为

$$y(\boldsymbol{x}) = f(\boldsymbol{w}^T \phi(\boldsymbol{x})) \tag{6.18}$$

其中非线性激活函数 $f(\cdot)$ 是一个阶梯函数，形式为

$$f(a) = \begin{cases} +1, & a \geqslant 0 \\ -1, & a < 0 \end{cases} \tag{6.19}$$

向量 $\phi(\boldsymbol{x})$ 通常包含一个偏置分量 $\phi_0(\boldsymbol{x}) = 1$。用来确定感知器的参数 $\boldsymbol{w}$ 的算法可以很容易地从误差函数最小化的思想中得到。一个自然的选择是误分类的模型的总数。但是，这样做会使得学习算法不会很简单，因为这样做会使误差函数变为 $\boldsymbol{w}$ 的分段常函数，从而当 $\boldsymbol{w}$ 的变化使得决策边界移过某个数据点时，这个函数会不连续变化。这样会不连续变化。这样做还使得使用误差函数改变 $\boldsymbol{w}$ 的方法无法使用，因为在几乎所有的地方梯度都等于零。

因此考虑一个另外的误差函数，被称为感知器准则 (perceptron criterion)。

$$E_P(\boldsymbol{w}) = -\sum_{n\in\mathcal{M}} \boldsymbol{w}^T\phi_n t_n \tag{6.20}$$

其中 $\phi_n = \phi(\boldsymbol{x}_n)$ 和 $\mathcal{M}$ 表示所有误分类模型的集合。某个特定的误分类模型对于误差函数的贡献是 $\boldsymbol{w}$ 空间中模式被误分类的区域中 $\boldsymbol{w}$ 的线性函数，而在正确分类的区域，误差函数等于零。总的误差函数因此是分段线性的。我们现在对这个误差函数使用随机梯度下降算法。这样，权向量 $\boldsymbol{w}$ 的变化为

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta\nabla E_P(\boldsymbol{w}) = \boldsymbol{w}^{(\tau)} + \eta\phi_n t_n \tag{6.21}$$

感知器收敛定理 (perceptron convergence theorem) 表明，如果存在一个精确的解 (即，如果训练数据线性可分)，那么感知器算法可以保证在有限步骤内找到一个精确解。

这里要画一个图

6.2 概率生成式模型

接下来用概率的观点考察分类问题，并且说明具有线性决策边界的模型如何通过对数据分布的简单假设得到。

首先考虑二分类的情形。类别 C_1 的后验概率可以写成

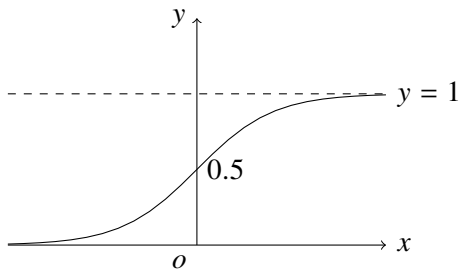
$$\begin{aligned} p(C_1|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|C_1)p(C_1)}{p(\boldsymbol{x}|C_1)+p(\boldsymbol{x}|C_2)p(C_2)} \\ &= \frac{1}{1+exp(-a)} = \sigma(a) \end{aligned} \tag{6.22}$$

其中我们定义了

$$a = \ln\frac{p(\boldsymbol{x}|C_1)p(C_1)}{p(\boldsymbol{x}|C_2)p(C_2)} \tag{6.23}$$

$\sigma(a)$ 是 logistic sigmoid 函数。“sigmoid” 的意思是 “S 形”。这种函数有时被称为 “挤压函数”，这个函数在许多分类算法中都有着重要的作用。它满足下面的对称性

$$\sigma(-a) = 1 - \sigma(a) \tag{6.24}$$



logistic sigmoid 的反函数为

$$a = \ln\left(\frac{\sigma}{1-\sigma}\right) \tag{6.25}$$

被称为 logit 函数。它表示两类的概率比值的对数 $\ln\left[\frac{p(C_1|\boldsymbol{x})}{p(C_2|\boldsymbol{x})}\right]$，也被称为 log odds 函数。

对于 K>2 个类别的情形，我们有

$$\begin{aligned} p(C_k|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|C_k)p(C_k)}{\sum_j p(\boldsymbol{x}|C_j)p(C_j)} \\ &= \frac{exp(a_k)}{\sum_j exp(a_j)} \end{aligned} \tag{6.26}$$

它被称为归一化指数 (normalized exponential)，可以被当做 logistic sigmoid 函数对于多类情况的推广。这里 a_k 被定义为

$$a_k = \ln p((\boldsymbol{x}|C_k)p(C_k)) \tag{6.27}$$

归一化指数也被称为 softmax 函数，因为它表示“max”函数的一个平滑版本。

连续输入

假设类条件概率密度是高斯分布，然后求解后验概率的形式。首先，假设所有的类别的协方差矩阵相同。这样类别 C_k 的类条件概率为

$$p(\boldsymbol{x}|C_k) = \frac{1}{(2\pi)^{\frac{D}{2}}}\frac{1}{|\Sigma|^{\frac{1}{2}}}exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right\} \tag{6.28}$$

首先考虑两类的情形。

$$
\begin{aligned}
& p(C_1|\boldsymbol{x}) = \sigma(a) \\
\Rightarrow\ & a = \ln \frac{p(\boldsymbol{x}|C_1)p(C_1)}{p(\boldsymbol{x}|C_2)p(C_2)} \\
& = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) + \ln \frac{p(C_1)}{p(C_2)} \\
& = -\frac{1}{2}\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x} + \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_1 - \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 \\
& \quad + \frac{1}{2}\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_2 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 \\
& \quad + \ln \frac{p(C_1)}{p(C_2)} \\
& = \Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 + \ln \frac{p(C_1)}{p(C_2)} \\
& = \boldsymbol{w}^T\boldsymbol{x} + w_0 \\
\Rightarrow\ & \sigma(a) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + w_0)
\end{aligned}
\tag{6.29}
$$

高斯概率密度的指数项中 $\boldsymbol{x}$ 的二次型消失了 (这是因为我们假设类概率的协方差矩阵相同),从而得到了参数为 $\boldsymbol{x}$ 的线性函数的 logistic sigmoid 函数。最终求得决策边界 $a = 0$ 为常数的决策面。先验概率密度 $p(C_k)$ 只出现在偏置参数 w_0 中,因此先验的改变的效果是平移决策边界,即平移后验概率中的常数轮廓线。

对于 K 个类别的一般情形,根据公式 6.26,有

$$a_k(\boldsymbol{x}) = \boldsymbol{w}_k^T\boldsymbol{x} + w_{k0} \tag{6.30}$$

$$w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k + \ln p(C_k) \tag{6.31}$$

其中定义了

$$\boldsymbol{w}_k = \Sigma^{-1}\boldsymbol{\mu}_k \tag{6.32}$$

如果我们不假设各个类别的协方差矩阵相同,允许每个类条件概率密度 $p(\boldsymbol{x}|C_k)$ 有自己的协方差矩阵 Σ_k,那么之前二次项消去的现象不会出现,从而我们会得到 $\boldsymbol{x}$ 的二次函数,这就引出了二次判别函数 (quadratic discriminant)。

最大似然解

一旦具体化了类条件概率密度 $p(\boldsymbol{x}|C_k)$ 的参数化的函数形式,我们就能够使用最大似然法确定参数的值,以及先验类概率 $p(C_k)$。

首先考虑两类的情形,每个类别都有一个高斯类条件概率密度,且协方差矩阵相同。假设有一个数据集 $\{x_n, t_n\}$,其中 $n = 1, \ldots, N$。先验概率记作 $p(C_1) = \pi$,从而 $p(C_2) = 1-\pi$。

对于一个来自类别 C_1 的数据点 $\boldsymbol{x}_n$，我们有 $t_n = 1$，因此

$$p(\boldsymbol{x}_n, C_1) = p(C_1)p(\boldsymbol{x}_n|C_1) = \pi\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_1, \Sigma) \tag{6.33}$$

$$p(\boldsymbol{x}_n, C_2) = p(C_2)p(\boldsymbol{x}_n|C_2) = (1-\pi)\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_2, \Sigma) \tag{6.34}$$

于是，似然函数为

$$p(\boldsymbol{t}, \boldsymbol{X}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma) = \prod_{n=1}^{N}[\pi\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_1, \Sigma)]^{t_n}[(1-\pi)\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_2, \Sigma)]^{1-t_n} \tag{6.35}$$

其中 $\boldsymbol{t} = (t_1, \ldots, t_N)^T$。对数似然函数为

$$\begin{aligned}\log p(\boldsymbol{t}, \boldsymbol{X}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma) &= \sum_{n=1}^{N}\log\left\{\pi\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_1, \Sigma)]^{t_n}[(1-\pi)\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_2, \Sigma)]^{1-t_n}\right\} \\ &= \sum_{n=1}^{N}\left\{\log\pi^{t_n}(1-\pi)^{1-t_n} + \log\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_1, \Sigma)^{t_n} + \log\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_2, \Sigma)^{1-t_n}\right\}\end{aligned} \tag{6.36}$$

最大化对数似然函数

1. 求 π

$$\begin{aligned}\frac{\partial}{\partial\pi}\left[\sum_{n=1}^{N}\log\pi^{t_n}(1-\pi)^{1-t_n}\right] &= \sum_{n=1}^{N}\frac{t_n}{\pi} - \frac{(1-t_n)}{1-\pi} = 0 \\ &= \sum_{n=1}^{N}t_n - \pi t_n - \pi + \pi t_n \\ &= \sum_{n=1}^{N}t_n - \pi \\ &= \sum_{n=1}^{N}t_n - N\pi = 0 \\ \Rightarrow \pi &= \frac{1}{N}\sum_{n=1}^{N}t_n = \frac{N_1}{N} = \frac{N_1}{N_1+N_2}\end{aligned} \tag{6.37}$$

2. 求 $\boldsymbol{\mu}$

$$
\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\mu}_1}\left[\log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_1,\Sigma)^{t_n}\right] &\Rightarrow \frac{\partial}{\partial \boldsymbol{\mu}_1}\left[\sum_{n=1}^{N} -\frac{1}{2} t_n(\boldsymbol{x}_n-\boldsymbol{\mu}_1)^T\Sigma^{-1}(\boldsymbol{x}_n-\boldsymbol{\mu}_1)\right] \\
&= \frac{\partial}{\partial \boldsymbol{\mu}_1}\left[-\frac{1}{2}\sum_{n=1}^{N} t_n(\boldsymbol{x}_n^T\Sigma^{-1}-\boldsymbol{\mu}_1^T\Sigma^{-1})(\boldsymbol{x}_n-\boldsymbol{\mu}_1)\right] \\
&= \frac{\partial}{\partial \boldsymbol{\mu}_1}\left[-\frac{1}{2}\sum_{n=1}^{N} t_n(\underbrace{\boldsymbol{x}_n^T\Sigma^{-1}\boldsymbol{x}_n}_{\text{常数}} -2\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{x}_n+\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1)\right] \\
&= \sum_{n=1}^{N} t_n(\Sigma^{-1}\boldsymbol{x}_n+\Sigma^{-1}\boldsymbol{\mu}_1)=0 \\
&\Rightarrow \sum_{n=1}^{N} t_n(\boldsymbol{\mu}_1-\boldsymbol{x}_n)=0 \\
\Rightarrow \boldsymbol{\mu}_1 &= \frac{\sum_{n=1}^{N} t_n\boldsymbol{x}_n}{\sum_{n=1}^{N} t_n} \\
&= \frac{1}{N_1}\sum_{n=1}^{N} t_n\boldsymbol{x}_n
\end{aligned}
\tag{6.38}
$$

同理

$$
\boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N}(1-t_n)\boldsymbol{x}_n \tag{6.39}
$$

3. 求 Σ

定理 6.1

$$
Tr(AB) = Tr(BA) \tag{6.40}
$$

$$
\frac{\partial Tr(AB)}{\partial A} = B^T \tag{6.41}
$$

$$
\frac{\partial |A|}{\partial A} = |A|A^{-1} \tag{6.42}
$$

$$
\frac{\partial \ln|A|}{\partial A} = A^{-1} \tag{6.43}
$$

♡

考察部分分式

$$\begin{aligned}\sum_{n=1}^{N}\log\mathcal{N}(\boldsymbol{\mu},\Sigma) &= \sum_{n=1}^{N}\log|\Sigma|^{-\frac{1}{2}} + (-\frac{1}{2}(\boldsymbol{x}_n-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}_n-\boldsymbol{\mu})) + C \\ &= -\frac{1}{2}N\log|\Sigma| - \frac{1}{2}\sum_{n=1}^{N}(\boldsymbol{x}_n-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}_n-\boldsymbol{\mu}) + C \\ &= -\frac{1}{2}N\log|\Sigma| - \frac{1}{2}NTr(S\cdot\Sigma^{-1}) + C\end{aligned} \tag{6.44}$$

其中

$$S = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n-\boldsymbol{\mu})(\boldsymbol{x}_n-\boldsymbol{\mu})^T \tag{6.45}$$

将与 Σ 相关的式子组合到一起，有，求关于 Σ 的偏导，并令其为零

$$\begin{aligned}\triangle &= \sum_{n=1}^{N}\log\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_1,\Sigma)^{t_n} + \log\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_2,\Sigma)^{1-t_n} \\ &= -\frac{1}{2}\log|\Sigma| - \frac{1}{2}N_1Tr(S_1\cdot\Sigma^{-1}) - \frac{1}{2}N_2Tr(S_2\cdot\Sigma^{-1}) + C \\ \frac{\partial\triangle}{\partial\Sigma} &= -\frac{1}{2}(N\Sigma^{-1} - N_1S_1\Sigma^{-2} - N_2S_2\Sigma^{-2}) = 0 \\ &\Rightarrow \Sigma = \frac{N_1}{N}S_1 + \frac{N_2}{N}S_2\end{aligned} \tag{6.46}$$

这个结果很容易推广到 K 类问题，得到参数的对应的最大似然解。其中我们假定每个类条件概率密度都是高斯分布，协方差矩阵相同。注意，拟合类高斯分布的方法对于离群点并不鲁棒，因为高斯的最大似然估计是不鲁棒的。

离散特征

考虑离散特征值 x_i 的情形。为了简化起见，我们首先考虑二元特征值 $x_i \in \{0,1\}$，稍后会讨论如何推广到更一般的离散特征。如果有 D 个输入，那么一般的概率分布会对应于一个大小为 2^D 的表格，包含 2^D-1 个独立变量。由于这会随着特征的数量指数增长，因此我们想寻找一个更加严格的表示方法。这里，我们做出朴素贝叶斯 (naive Bayes) 假设，这个假设中，特征值被看成相互独立的，以类别 C_k 为条件，因此。我们得到类条件分布，形式为

$$\begin{aligned}p(\boldsymbol{X}=\boldsymbol{x}|C_k) &= p(X^{(1)}=x^{(1)},\ldots,X^{(D)}=x^{(D)}|C_k) \\ &= \prod_{i=1}^{D}\mu_{k_i}^{x_i}(1-\mu_{k_i})^{1-x_i}\end{aligned} \tag{6.47}$$

其中对于每个类别,都有 D 个独立的参数。代入 softmax 函数中,有

$$\begin{aligned} a_k(\boldsymbol{x}) &= \ln p(\boldsymbol{x}|C_k)p(C_k) \\ &= \ln \prod_{i=1}^{D} \mu_{k_i}^{x_i}(1-\mu_{k_i})^{1-x_i} + \ln p(C_k) \\ &= \sum_{i=1}^{D}\{x_i \ln \mu_{k_i} + (1-x_i)\ln(1-\mu_{k_i})\} + \ln p(C_k) \end{aligned} \tag{6.48}$$

与之前一样,这是输入变量 x_i 的线性函数。对于 $K=2$ 个类别的情形,我们可以考虑另一种方法——logistic sigmoid 函数。离散变量也有类似的结果,其中,每个离散变量有 $M>2$ 种状态。

朴素贝叶斯法实际上学习到生成数据的机制,所以属于生成模型,条件独立性假设等于是说用于分类的特征在类确定的条件下都是条件独立的。这一假设使得朴素贝叶斯变得简单,但有时会牺牲一定的分类准确率。

朴素贝叶斯法分类时,对给定的输入 x,通过学习到的模型计算后验概率分布 $p(C_k|X=x)$,将后验概率最大的类作为 x 类的输出。后验概率计算根据贝叶斯定理进行。

$$y = f(\boldsymbol{x}) = arg\ \underset{c_k}{max} \frac{p(C_k)\prod_j p(X^{(j)}|C_k)}{\sum_k p(C_k)\prod_j p(X^{(j)}|C_k)} \tag{6.49}$$

注意到分母对所有 C_k 都是相同的,所以

$$y = arg\ \underset{c_k}{max}\, p(C_k)\prod_j p(X^{(j)}|C_k) \tag{6.50}$$

朴素贝叶斯法将实例分到后验概率最大的类中。这等价于期望风险最小化。假设选择 0-1 损失函数:

$$L(Y, f(X)) = \begin{cases} 1, & Y \neq f(X) \\ 0, & Y = f(X) \end{cases} \tag{6.51}$$

式中 $f(X)$ 是分类决策函数。这时,期望风险函数为

$$\begin{aligned} E_{exp}(f) &= E[L(Y, f(X))] \\ &= \int_{xXy} L(y, f(x))p(x, y)dxdy \\ &= \int_x [L(y, f(x))p(y|x)dy]p(x)dx \\ &= E_X \sum_{k=1}^{K}[L(C_k, f(X))]p(C_k|X) \end{aligned} \tag{6.52}$$

为了使期望风险最小化，只需对 $X = x$ 逐个极小化，由此得到：

$$\begin{aligned} f(x) &= arg\min_{y\in\dagger} \sum_{k=1}^{K} L(C_k, y) p(C_k|X = x) \\ &= arg\min_{y\in\dagger} \sum_{k=1}^{K} p(y \neq C_k|X = x) \\ &= arg\min_{y\in\dagger} (1 - p(y = C_k|X = x)) \\ &= arg\max_{y\in\dagger} p(y = C_k|X = x) \end{aligned} \tag{6.53}$$

下面给出朴素贝叶斯法的学习与分类算法

输入：训练数据 $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$，其中 $x_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)})^T$，$x_i^{(j)}$ 是第 i 个样本的第 j 个特征，$x_i^{(j)} \in \{a_{j1}, a_{j2}, \ldots, a_{jS_j}\}$，$a_{jl}$ 是第 j 个特征可能取的第 l 个值，$j = 1, 2, \ldots, n, l = 1, 2, \ldots, S_j, y_i \in \{c_1, c_2, \ldots, c_k\}$; 实例 x;

输出：实例 x 的分类。

1. 计算先验概率及条件概率

$$P(Y = C_k) = \frac{\sum_{i=1}^{N} I(y_i = C_k)}{N}, k = 1, 2, \ldots, K \tag{6.54}$$

$$P(X^{(j)} = a_{jl}|Y = C_k) = \frac{\sum_{i=1}^{N} I(x_i^{(j)} = a_{jl}, y_i = C_k)}{\sum_{i=1}^{N} I(y_i = C_k)} \tag{6.55}$$

$$j = 1, 2, \ldots, N, l = 1, 2, \ldots, S_j, k = 1, 2, \ldots, K \tag{6.56}$$

2. 对于给定的实例 $x = (x^{(1)}, x^{(2)}, \ldots, x^{(n)})^T$，计算

$$P(Y = C_k) \prod_{j=1}^{n} P(X^{(j)} = x^{(j)}|Y = C_k), k = 1, 2, \ldots, K \tag{6.57}$$

3. 确定实例 x 的类

$$y = arg\max_{C_k} P(Y = C_k) \prod_{j=1}^{n} P(x^{(j)} = x^{(j)}|Y = Cc_k) \tag{6.58}$$

指数族分布

正如我们已经看到的，无论是服从高斯分布的输入，还是离散的输入，后验类概率密度都是由一般的线性模型和 logistic sigmoid($K = 2$ 个类别) 或者 softmax($K \geqslant 2$ 个类别) 激活函数给出。通过假设类条件概率密度 $p(\boldsymbol{x}|C_k)$ 是指数族分布的成员，我们可以看到上述结果都是更一般的结果的特例。

6.3 概率判别式模型

对于二分类问题，我们已经看到，对于一大类的类条件概率密度 $p(\boldsymbol{x}|C_k)$ 的选择，类别 C_1 后验概率分布可以写成作用于 $\boldsymbol{x}$ 的线性函数上的 logistic sigmoid 函数的形式。类似地，对于多分类的情形，类别 C_k 的后验概率由 $\boldsymbol{x}$ 的线性函数的 softmax 变换给出。对于类条件概率密度 $p(\boldsymbol{x}|C_k)$ 的具体的选择，我们已经使用了最大似然方法估计了概率密度的参数以及类别先验 $p(C_k)$，然后使用贝叶斯定理就可以求出后验类概率。

另一种方法是显示地使用一般的线性模型的函数形式，然后使用最大似然法直接确定它的参数。

寻找一般的线性模型参数的间接方法是，分别寻找类条件概率密度和类别先验，然后使用贝叶斯定理。这是生成式建模的一个例子。在直接方法中，我们最大化由条件概率分布 $p(C_k|\boldsymbol{x})$ 定义的似然函数。这种方法代表了判别式训练的一种形式。判别式方法的一个优点是通常有更少的可调节的参数需要确定，并且预测表现会提升，尤其是当类条件概率密度的假设没有很好地近似真实的分布的时候更是如此。

固定基函数

如果首先使用一个基函数向量 $\boldsymbol{\phi}(\boldsymbol{x})$ 对输入变量进行一个固定的非线性变换，所有的这些算法仍然同样适用。

logistic 回归

首先通过二分类问题开始对于一般线性模型的讨论。类别 C_1 的后验概率可以写成作用在特征向量 $\boldsymbol{\phi}$ 的线性函数上的 logistic sigmoid 函数的形式，即

$$p(C_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma(\boldsymbol{w}^T\boldsymbol{\phi}) \tag{6.59}$$

$$p(C_2|\boldsymbol{\phi}) = 1 - p(C_1|\boldsymbol{\phi}) \tag{6.60}$$

这个模型被称为 logistic 回归，这是一个分类模型而不是回归模型。

对于一个 M 维特征空间 $\boldsymbol{\phi}$，这个模型有 M 个可调节参数。相反，如果我们使用最大似然方法调节高斯类条件概率密度，那么我们有 2M 个参数来描述均值，以及 $\frac{M(M+1)}{2}$ 个参数来描述协方差矩阵。算上类先验 $p(C_1)$，参数的总数为 $\frac{M(M+5)}{2}+1$，这随着 M 的增长而以二次的方式增长。这和 logstic 回归方法中对于参数数据 M 的线性依赖不同。对于大的 M 值，直接使用 logistic 回归模型有着很明显的优势。

现在使用最大似然方法来确定 logistic 回归模型的参数。对于一个数据集 $\boldsymbol{\phi}_n, t_n$，其中 $t_n \in \{0,1\}, \boldsymbol{\phi}_n = \boldsymbol{\phi}(\boldsymbol{x}_n), n = 1,\ldots,N$，似然函数可以写成

$$p(\boldsymbol{t}|\boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n}\{1-y_n\}^{1-t_n} \tag{6.61}$$

其中 $\boldsymbol{t} = (t_1,\ldots,t_N)^T$ 且 $y_n = p(C_1|\boldsymbol{\phi}_n)$。通过取似然函数的负对数的方式，定义一个误差

函数。这种方式产生了交叉熵 (cross-entropy) 误差函数，形式为

$$\begin{aligned} E(\boldsymbol{w}) &= -\ln p(\boldsymbol{t}|\boldsymbol{w}) \\ &= -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} \end{aligned} \tag{6.62}$$

其中 $y_n = \sigma(a_n)$ 且 $a_n = \boldsymbol{w}^T\phi_n$。两侧关于 $\boldsymbol{w}$ 取误差函数的梯度，我们有

$$\begin{aligned} \nabla E(\boldsymbol{w}) &= -\sum_{n=1}^{N}\{\frac{t_n}{y_n}y_n^{'} + \frac{(1-t_n)}{1-y_n}(1-y_n)^{'}\} \\ &= -\sum_{n=1}^{N}\frac{t_n}{\sigma}\sigma(1-\sigma)\phi_n - \frac{1-t_n}{1-\sigma}\sigma(1-\sigma)\phi_n \\ &= \sum_{n=1}^{N}(y_n - t_n)\phi_n \end{aligned} \tag{6.63}$$

推导时用到了

$$\frac{d\sigma}{da} = \sigma(1-\sigma) \tag{6.64}$$

我们看到，涉及到 logistic sigmoid 的导数的因子已经被消去，使得对数似然函数的梯度的形式十分简单。特别地，数据点 n 对梯度的贡献为目标值和模型预测值之间的“误差”与基函数向量 ϕ_n 相乘。此外它的函数形式与线性回归模型中的平方和误差函数的梯度的函数形式完全相同。

问题变成了以对数似然函数为目标函数的最优化问题。logistic 回归学习中通常采用的方法是梯度下降法及拟牛顿法。

值得注意的一点是，最大似然方法对于线性可分的数据集会产生严重的过拟合现象。最大似然方法无法区分某个解优于另一个解，并且在实际应用中哪个解被找到将会依赖于优化算法的选择和参数的初始化。只要数据是线性可分的，这个问题就会出现。通过引入先验概率，然后寻找 $\boldsymbol{w}$ 的 MAP 解，或者等价地，通过给误差函数增加一个正则化项，这种奇异性就可以被避免。

迭代重加权最小平方

对于 logistic 回归来说，不再有解析解，因为 logistic sigmoid 函数是一个非线性函数。然而，函数形式不是二次函数并不是本质的原因。精确地说，误差函数是凸函数，因此有一个唯一的最小值。此外，误差函数可以通过一种高效的迭代方法求出最小值，这种迭代方法基于 Newton-Raphson 迭代最优化框架，使用了对数似然函数的局部二次近似。为了最小化函数 $E(\boldsymbol{w})$，Newton-Raphson 对权值的更新的形式为

$$\boldsymbol{w}^{\text{新}} = \boldsymbol{w}^{\text{旧}} - H^{-1}\nabla E(\boldsymbol{w}) \tag{6.65}$$

其中 H 是一个 Hessian 矩阵。把 Newton-Raphson 方法应用到现行回归模型上，误差函数为平方和误差函数。这个误差函数的梯度和 Hessian 矩阵为

$$\nabla E[\boldsymbol{w}] = \sum_{n=1}^{N}(\boldsymbol{w}^T\phi_n - t_n)\phi_n = \Phi^T\Phi\boldsymbol{w} - \Phi^T\boldsymbol{t} \tag{6.66}$$

$$H = \nabla\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N}\phi_n\phi_n^T = \Phi^T\Phi \tag{6.67}$$

其中 Φ 是 $N \times M$ 设计矩阵，第 n 行为 ϕ_n^T。于是，Newton-Raphson 更新的形式为

$$\begin{aligned}\boldsymbol{w}^{新} &= \boldsymbol{w}^{旧} - (\Phi^T\Phi)^{-1}\{\Phi^T\Phi\boldsymbol{w}^{旧} - \Phi^T\boldsymbol{t}\} \\ &= (\Phi^T\Phi)^{-1}\Phi^T\boldsymbol{t}\end{aligned} \tag{6.68}$$

现在把 Newton-Raphson 更新应用到 logistic 回归模型的交叉熵误差函数上。

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N}(y_n - t_n)\phi_n = \Phi^T(\boldsymbol{y} - \boldsymbol{t}) \tag{6.69}$$

$$H = \nabla\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N} y_n(1 - y_n)\phi_n\phi_n^T = \Phi^T\mathcal{R}\Phi \tag{6.70}$$

推导过程中，引入了一个 $N \times N$ 的对角矩阵 $\mathcal{R}$，元素为

$$\mathcal{R}_{nn} = y_n(1 - y_n) \tag{6.71}$$

我们看到，Hessian 矩阵不再是常量，而是通过权矩阵 $\mathcal{R}$ 依赖于 $\boldsymbol{w}$。这对应于误差函数不是二次函数的事实。使用性质 $0 < y_n < 1$，我们看到对于任意向量 $\boldsymbol{u}$ 都有 $\boldsymbol{u}^T H\boldsymbol{u} > 0$，因此 Hessian 矩阵 H 是正定的。因此误差函数是 $\boldsymbol{w}$ 的一个凸函数，从而有唯一的最小值。

这样，logistic 回归模型的 Newton-Raphson 更新公式就变成了

$$\begin{aligned}\boldsymbol{w}^{新} &= \boldsymbol{w}^{旧} - (\Phi^T\mathcal{R}\Phi)^{-1}\Phi^T(\boldsymbol{y} - \boldsymbol{t}) \\ &= (\Phi^T\mathcal{R}\Phi)^{-1}\{\Phi^T\mathcal{R}\Phi\boldsymbol{w}^{旧} - \Phi^T(\boldsymbol{y} - \boldsymbol{t})\} \\ &= (\Phi^T\mathcal{R}\Phi)^{-1}\Phi^T\mathcal{R}\boldsymbol{z}\end{aligned} \tag{6.72}$$

其中 $\boldsymbol{z}$ 是一个 N 维向量，元素为

$$\boldsymbol{z} = \Phi\boldsymbol{w}^{旧} - \mathcal{R}^{-1}(\boldsymbol{y} - \boldsymbol{t}) \tag{6.73}$$

更新公式的形式为一组加权最小平方问题的规范方程。由于权矩阵 $\mathcal{R}$ 不是常量，而是依赖于参数向量 $\boldsymbol{w}$，因此我们必须迭代地应用规范方程，每次使用新的权向量 $\boldsymbol{w}$ 计算一个修正的权矩阵 $\mathcal{R}$，由于这个原因，这个算法被称为迭代重加权最小平方 (iterative reweighted least squares)，或者简称为 IRLS。对角 $\mathcal{R}$ 可以看成方差。

多类 logistic 回归

对于多分类的生成式模型，后验概率由特征变量的线性函数的 softmax 变换给出，即

$$p(C_k|\phi) = y_k(\phi) = \frac{exp(\boldsymbol{w}_k^T\phi)}{\sum_j exp(\boldsymbol{w}_k^T\phi)} \tag{6.74}$$

probit 回归

对于由指数族分布描述的一大类的类条件概率分布，最终求出的后验类概率为作用在特征变量的线性函数上的 logistic(或者 softmax) 变换。然而，不是所有的类条件概率密度都有这样简单的后验概率函数形式 (例如，如果类条件概率密度由高斯混合模型建模)。这表明研究其他类型的判别式概率模型可能会很有价值。回到二分类的情形，再次使用一般的线性模型的框架，即

$$p(t = 1|a) = f(a) \tag{6.75}$$

其中 $a = \boldsymbol{w}^T\phi$，且 $f(\cdot)$ 为激活函数。对于每个输入 ϕ_n，我们计算 $\boldsymbol{w}^T\phi_n$，然后按照下面的方式设置目标值

$$\begin{cases} t_n = 1, & \text{如果} a_n \geqslant \theta \\ t_n = 0, & \text{其他情况} \end{cases} \tag{6.76}$$

如果 θ 的值从概率密度 $p(\theta)$ 中抽取，那么对应的激活函数由累积分布函数给出

$$f(a) = \int_{-\infty}^{a} p(\theta)d\theta \tag{6.77}$$

这被称为逆 probit(inverse probit) 函数。它的形状为 sigmoid 形。许多用于计算这个函数的数值计算包都与下面的这个函数紧密相关

$$erf(a) = \frac{2}{\sqrt{\pi}} \int_0^a exp(-\theta^2)d\theta \tag{6.78}$$

它被称为 erf 函数或者被称为 error 函数。它与逆 probit 函数的关系为

$$\Phi(a) = \frac{1}{2}\left\{1 + erf\left(\frac{a}{\sqrt{2}}\right)\right\} \tag{6.79}$$

基于 probit 激活函数的一般的线性模型被称为 probit 回归。

标准链接函数

如果假设目标变量的条件分布来自于指数族分布，对应的激活函数选为标准链接函数 (canonical link function)，那么这个结果是一个一般的结果。

6.4 拉普拉斯近似

特别地，我们不能够精确地关于参数向量 $\boldsymbol{x}$ 求积分，因为后验概率分布不再是高斯分布。因此，有必要介绍某种形式的近似。后面章节中，我们会介绍一系列分析估计和数值采样的技术。

拉普拉斯近似的目标是找到定义在一组连续变量上的概率密度的高斯近似。首先考虑单一连续变量 z 的情形，假设分布 $p(z)$ 的定义为

$$p(z) = \frac{1}{Z} f(z) \tag{6.80}$$

其中 $Z = \int f(z)dz$ 是归一化系数。我们假定 Z 的值是未知的。在拉普拉斯方法中，目标是寻找一个高斯近似 $q(z)$，它的中心位于 $p(z)$ 的众数的位置。第一步是寻找 $p(z)$ 的众数，即寻找一个点 z_0 使得 $p'(z_0) = 0$，或者等价地

$$\left.\frac{df(z)}{dz}\right|_{z=z_0} = 0 \tag{6.81}$$

高斯分布有一个性质，即它的对数是变量的二次函数。于是我们考虑 $\ln f(z)$ 以众数 z_0 为中心的泰勒展开，即

$$\ln f(z) \simeq \ln f(z_0) - \frac{1}{2} A(z - z_0)^2 \tag{6.82}$$

其中

$$A = -\left.\frac{d^2}{dz^2} \ln f(z)\right|_{z=z_0} \tag{6.83}$$

注意，泰勒展开式中的一阶项没有出现，因为 z_0 是概率分布的局部最大值。两侧同时取指数，我们有

$$f(z) \simeq f(z_0) exp\left\{-\frac{A}{2}(z - z_0)^2\right\} \tag{6.84}$$

这样，使用归一化的高斯分布的标准形式，我们就可以得到归一化的概率分布 $q(z)$，即

$$q(z) = \left(\frac{A}{2\pi}\right)^{\frac{1}{2}} exp\left\{-\frac{A}{2}(z - z_0)^2\right\} \tag{6.85}$$

注意，高斯近似只在精度 A>0 时有良好的定义，换句话说，驻点 z_0 一定是一个局部最大值，使得 $f(z)$ 在驻点 z_0 处的二阶导数为负。

我们可以将拉普拉斯方法推广，使其近似多元高斯分布。在应用拉普拉斯方法时，真实概率分布的归一化常数 Z 不必事先知道。根据中心极限定理，我们可以预见模型的后验概率会随着观测数据点的增多而越来越近似于高斯分布，因此我们可以预见在数据点相对较多的情况下，拉普拉斯近似会更有用。拉普拉斯近似的一个主要缺点是，由于它是以高斯分布为基础的，因此它只能直接应用于实值变量。在其他情况下，可以将拉普拉斯近似应用于变换之后的变量上。但是，拉普拉斯框架的最严重的局限性是，它完全依赖于

真实概率分布在变量的某个具体位置上的性质，因此会无法描述一些重要的全局属性。

模型比较和 BIC

除了近似概率分布 $p(z)$，我们也可以获得对归一化常数 Z 的一个近似。

$$\begin{aligned} Z &= \int f(z)dz \\ &\simeq f(z_0)\int exp\left\{-\frac{1}{2}(z-z_0)^T A(z-z_0)\right\}dz \\ &= f(z_0)\frac{(2\pi)^{\frac{M}{2}}}{|A|^{\frac{1}{2}}} \end{aligned} \tag{6.86}$$

我们可以使用公式 6.86 的结果来获得对于模型证据的一个近似。

6.5 贝叶斯 logistic 回归

我们现在考虑 logistic 回归的贝叶斯观点。对于 logistic 回归，精确的贝叶斯推断是无法处理的。特别地，计算后验概率分布需要对先验概率分布于似然函数的乘积进行归一化，而似然函数本身由一系列 logistic sigmoid 函数的乘积组成，每个数据点都有一个 logistic sigmoid 函数。对于预测分布的计算类似地也是无法处理的。这里我们考虑使用拉普拉斯近似来处理贝叶斯 logistic 回归的问题。

拉普拉斯近似

为了获得后验概率的高斯近似，我们首先最大化后验概率分布，得到 MAP(最大后验)解 $\boldsymbol{w}_{MAP}$，它定义了高斯分布的均值。这样协方差就是负对数似然函数的二阶导数矩阵的逆矩阵，形式为

$$\boldsymbol{S}_N^{-1} = -\nabla\nabla \ln p(\boldsymbol{w}|\boldsymbol{t}) = \boldsymbol{S}_0^{-1} + \sum_{n=1}^{N} y_n(1-y_n)\phi_n\phi^T \tag{6.87}$$

于是后验概率分布的高斯近似的形式为

$$q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_{MAP}, \boldsymbol{S}_N) \tag{6.88}$$

预测分布

给定一个新的特征向量 $\phi(\boldsymbol{x})$，类别 C_1 的预测分布可以通过对后验概率 $p(\boldsymbol{w}|\boldsymbol{t})$ 积分，后验概率本身由高斯分布 $q(\boldsymbol{w})$ 近似，即

$$p(C_1|\phi,\boldsymbol{t}) = \int p(C_1|\phi,\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{t})d\boldsymbol{w} \simeq \int \sigma(\boldsymbol{w}^T\phi)q(\boldsymbol{w})d\boldsymbol{w} \tag{6.89}$$

类别 C_2 的对应的概率为 $p(C_2|\phi,\boldsymbol{t}) = 1 - p(C_1|\phi,\boldsymbol{t})$。

为了计算预测分布，我们首先注意到函数 $\sigma(\boldsymbol{w}^T\phi)$ 对于 $\boldsymbol{w}$ 的依赖只通过它在 ϕ 上的投影而实现。记 $a = \boldsymbol{w}^T\phi$，我们有

$$\sigma(\boldsymbol{w}^T\phi) = \int \delta(a - \boldsymbol{w}^T\phi)\sigma(a)da \tag{6.90}$$

其中 $\delta(\cdot)$ 是狄拉克 Delta 函数。由此我们有

$$\int \sigma(\boldsymbol{w}^T\phi)q(\boldsymbol{w})d\boldsymbol{w} = \int \sigma(a)p(a)da \tag{6.91}$$

其中

$$p(a) = \int \delta(a - \boldsymbol{w}^T\phi)q(\boldsymbol{w})d\boldsymbol{w} \tag{6.92}$$

我们可以这样计算 $p(a)$：注意到 Delta 函数给 $\boldsymbol{w}$ 施加了一个线性限制，因此在所有与 ϕ 正交的方向上积分，就得到了联合概率分布 $q(\boldsymbol{w})$ 的边缘分布。由于 $q(\boldsymbol{w})$ 是高斯分布，因此我们知道边缘概率分布也是高斯分布。我们可以通过计算各阶矩然后交换 a 和 $\boldsymbol{w}$ 的积分顺序的方式计算均值和协方差，即

$$\mu_a = \mathbb{E}[a] = \int p(a)ada = \int q(\boldsymbol{w})\boldsymbol{w}^T\phi d\boldsymbol{w} = \boldsymbol{w}_{MAP}^T\phi \tag{6.93}$$

推导时使用了 6.88 给出的后验概率分布 $q(\boldsymbol{w})$ 的结果。类似地

$$\begin{aligned}\sigma_a^2 = var[a] &= \int p(a)\{a^2 - \mathbb{E}[a]^2\}da \\ &= \int q(\boldsymbol{w})\{(\boldsymbol{w}^T\phi)^2 - (\boldsymbol{m}_N^T\phi)^2\}d\boldsymbol{w} = \phi^T\boldsymbol{S}_N\phi\end{aligned} \tag{6.94}$$

因此我们对于预测分布的近似变成了

$$p(C_1|\boldsymbol{t}) = \int \sigma(a)p(a)da = \int \sigma(a)\mathcal{N}(a|\mu_a, \sigma_a^2)da \tag{6.95}$$

关于 a 的积分表示一个高斯分布和一个 logistic sigmoid 函数的卷积，不能够解析地求值。然而，我们可以利用 logistic sigmoid 函数 $\sigma(a)$ 和逆 probit 函数 $\Phi(a)$ 的高度相似性来获得一个较好的近似。

使用逆 Probit 函数的一个优势是它与高斯的卷积可以用另一个逆 probit 解析地表示出来。特别地，我们可以证明

$$\int \Phi(\lambda a)\mathcal{N}(a|\mu, \sigma^2)da = \Phi\left(\frac{\mu}{(\lambda^{-2} + \sigma^2)^{\frac{1}{2}}}\right) \tag{6.96}$$

我们现在将逆 probit 函数的近似 $\sigma(a) \simeq \Phi(\lambda a)$ 应用于这个方程的两侧，得到下面的对于 loigstic sigmoid 函数与高斯的卷积的近似

$$\int \sigma(a)\mathcal{N}(a|\mu, \sigma^2)da \simeq \sigma(k(\sigma^2)\mu) \tag{6.97}$$

其中我们定义了

$$k(\sigma^2) = (1 + \frac{\pi\sigma^2}{8})^{-\frac{1}{2}} \tag{6.98}$$

把这个结果应用于公式中，我们得到了近似的预测分布，形式为

$$p(C_1|\phi, \boldsymbol{t}) = \sigma(k(\sigma_a^2)\mu_a) \tag{6.99}$$

第 7 章 神经网络

在上两章，我们考虑了由固定基函数的线性组合构成的回归模型和分类模型。我们看到，这些模型具有一些有用的分析性质和计算性质，但是它们的实际应用被维数灾难问题限制了。为了将这些模型应用于大规模的问题，有必要根据数据调节基函数。

支持向量机是这样解决这个问题的：首先定义以训练数据点为中心的基函数，然后在训练过程中选择一个子集。支持向量机的一个优点是，虽然训练阶段涉及到非线性优化，但是目标函数是凸函数，因此最优化问的解相对很直接，并且通常随着数据规模的增加而增多。相关向量机也选择固定基函数集合的一个子集，通常会生成一个相当稀疏的模型。与支持向量机不同，相关向量机也产生概率形式的输出，嘎然这种输出的产生会以训练阶段的非凸优化为代价。

另一种方法是事先固定基函数的数量，但是允许基函数可调节。换名话，就是使用参数形式的基函数，这些参数可以在训练阶段调节。在模式识别中，这种类型的最成功的模型是有前馈神经网络，也被称为多层感知器 (multilayer perceptron)。与具有同样泛化能力的支持向量机相比，最终的模型会相当简洁，因此计算的速度更快。这种简洁性带来的代价就是，与相关向量机一样，构成了网络训练根基的似然函数不再是模型参数的凸函数。然而，在实际应用中，考察模型在训练阶段消耗的计算资源是很有价值的，这样做会得到一个简洁的模型，它可以快速地处理新数据。

首先，我们考虑神经网络的函数形式，包括基函数的具体参数，然后我们讨论使用最大似然框架确定神经网络参数的问题，这涉及到非线性最优化问题的解。这种方法需要计算对数似然函数关于神经网络参数的导数，我们会看到这些导数可以使用误差反向传播 (error backpropagation) 的方法高效地获得。我们还会说明误差反向传播的框架如何推广到计算其他的导数，例如 Jacobian 矩阵和 Hessian 矩阵。接下来，我们讨论神经网络训练的正则化和各种方法，以及方法之间的关系。我们还会考虑神经网络模型的一些扩展。特别地，我们会描述一个通用的框架，用来对条件概率密度建模。这个框架被称为混合密度网络 (mixture density network)。最后，我们讨论神经网络的贝叶斯观点。

7.1 前馈神经网络

回归的线性模型和分类的线性模型分别在第 5 章和第 6 章讨论过了。它们基于固定非线性基函数 $\phi_j(\boldsymbol{x})$ 的线性组合，形式为

$$y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\boldsymbol{x})\right) \tag{7.1}$$

其中 $f(\cdot)$ 在分类问题中是一个非线性激活函数，在回归问题中为恒等函数。我们的目标是推广这个模型，使得基函数 $\phi_j(\boldsymbol{x})$ 依赖于参数，从而能够让这些参数以及系数 $\{w_j\}$ 能够在

训练阶段调节。

这就引出了基本的神经网络，它可以被描述为一系列的函数变换。深度前馈网络 (deep feedforward network) 也叫做前馈神经网络 (feedforward neural network) 或者多层感知机 (multilayer perceptron,MLP)，是典型的深度学习模型。与感知器相比，一个重要的区别是神经网络在隐含单元中使用连续的 sigmoid 非线性函数，而感知器使用阶梯函数这一非线性函数。这意味着神经网络函数关于神经网络是可微的，这个性质在神经网络的训练过程中起着重要的作用。前馈神经网络的目标是近似某个函数 f^*。前馈网络定义了一个映射 $y = f(x;\theta)$，并且学习参数 θ 的值，使它能够得到最佳的函数近似。

前馈神经网络之所以被称作网络，是因为它们通常用许多不同函数复合在一起来表示。该模型与一个有向无环图相关联，而图描述了函数是如何复合在一起的。例如，我们有三个函数 $f^{(1)}, f^{(2)}, f^{(3)}$ 连接在一个链上以形成 $f^{(3)}(f^{(2)}(f^{(1)}(x)))$。

在神经网络训练过程中，我们让 $f(x)$ 去匹配 $f^*(x)$ 的值。训练数据为我们提供了在不同训练点上取值的、含有噪声的 $f^*(x)$ 近似实例。每个样本 x 都伴随着一个标签 $y \approx f^*(x)$。训练样本直接指明了输出层在每一点 x 上必须做什么；它必须产生一个接近 y 的值。但是训练数据并没有直接指明其他层应该怎么做。学习算法必须决定如何使用这些层来诞生想要的输出，但是训练数据并没有说每个单独的层应该做什么。相反，学习算法必须决定如何使用这些层来最好地实现 f^* 的近似。因为训练数据并没有给出这些层中的每一层所需的输出，所以这些层被称为隐藏层 (hidden layer)。

7.2 网络训练

根据解决的问题的类型，关于输出单元激活函数和对应的误差函数，都存在一个自然的选择。

1. 对于回归问题，我们使用线性输出和平方和误差函数
2. 对于 (多类独立的) 二元分类问题，我们使用 logistic sigmoid 输出以及交叉熵误差函数
3. 对于多类分类问题，我们使用 softmax 输出以及对应的多分类交叉熵错误函数。

对于涉及到两类的分类问题，我们可以使用单一的 logistic sigmoid 输出，也可以使用神经网络，这个神经网络有两个输出，且输出激活函数为 softmax 函数。

参数最优化

下面考虑寻找能够使得选定的误差函数 $E(\boldsymbol{w})$ 达到最小值的权向量 $\boldsymbol{w}$。由于误差 $E(\boldsymbol{w})$ 是 $\boldsymbol{w}$ 的光滑连续函数，因此它的最小值出现在权空间中误差函数梯度等于零的位置上，即

$$\nabla E(\boldsymbol{w}) = 0 \tag{7.2}$$

然而，误差函数通常与权值和偏置参数的关系是高度非线性的，因此权值空间中会有很多梯度为零 (或者梯度非常小) 的点。由于显然无法找到方程 $\nabla E(\boldsymbol{w}) = 0$ 的解析解，因此我

们使用迭代的数值方法。大多数方法涉及到为权向量选择某个初始值 $\boldsymbol{w}_0$，然后在权空间中进行一系列移动，形式为

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} + \triangle w^{(\tau)} \tag{7.3}$$

其中 τ 表示迭代次数。不同的算法涉及到权向量更新 $\triangle w^{(\tau)}$ 的不同选择。许多算法使用梯度信息，为了理解梯度信息的重要性，有必要考虑误差函数基于泰勒展开的局部近似。

局部二次近似

通过讨论误差函数的局部二次近似，我们可以更深刻地认识最优化问题，以及各种解决最优化问题的方法。考虑 $E(\boldsymbol{w})$ 在权空间某点 $\hat{\boldsymbol{w}}$ 处的泰勒展开

$$E(\boldsymbol{w}) \simeq E(\hat{\boldsymbol{w}}) + (\boldsymbol{w} - \hat{\boldsymbol{w}})^T \boldsymbol{b} + \frac{1}{2}(\boldsymbol{w} - \hat{\boldsymbol{w}})^T \boldsymbol{H}(\boldsymbol{w} - \hat{\boldsymbol{w}}) \tag{7.4}$$

$\boldsymbol{b}$ 被定义为 E 的梯度在 $\hat{\boldsymbol{w}}$ 处的值。

$$\boldsymbol{b} \equiv \nabla E|_{\boldsymbol{w}=\hat{\boldsymbol{w}}} \tag{7.5}$$

Hessian 矩阵 $\boldsymbol{H} = \nabla\nabla E$ 梯度的局部近似为

$$\nabla E \simeq \boldsymbol{b} + \boldsymbol{H}(\boldsymbol{w} - \hat{\boldsymbol{w}}) \tag{7.6}$$

考虑一个特殊情况：在误差函数最小值点 $\boldsymbol{w}^*$ 附近的局部二次近似。因为 $\nabla E = 0$

$$E(\boldsymbol{w}) \simeq E(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^T \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*) \tag{7.7}$$

为了用几何的形式表示这个结果，考虑 Hessian 矩阵的特征值方程

$$\boldsymbol{H}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i \tag{7.8}$$

其中特征向量 $\boldsymbol{u}_i$ 构成了完备的单位正交集合，即

$$\boldsymbol{u}_i^T \boldsymbol{u}_j = \delta_{ij} \tag{7.9}$$

现在把 $(\boldsymbol{w} - \boldsymbol{w}^*)$ 展开成特征值的线性组合的形式

$$\boldsymbol{w} - \boldsymbol{w}^* = \sum_i \alpha_i \boldsymbol{u}_i \tag{7.10}$$

误差函数可以写成

$$E(\boldsymbol{w}) = E(\boldsymbol{w}^*) + \frac{1}{2}\sum_i \lambda_i \alpha_i^2 \tag{7.11}$$

在最小值 $\boldsymbol{w}^*$ 的领域中，误差函数可以用二次函数近似。这样，常数误差函数的轮廓线为椭圆，它的轴与 Hessian 矩阵的特征向量 $\boldsymbol{u}_i$ 给出，长度与对应的特征值 λ_i 的平方根成反比。在新的坐标第中，基向量是特征向量 $\{\boldsymbol{u}_i\}$，E 为常数的轮廓线是以原点为中心的椭圆。对应的 D 维的结论是，在 $\boldsymbol{w}^*$ 处的 Hessian 矩阵是正定矩阵。

使用梯度信息

使用误差反向传播的方法可以高效地计算误差函数的梯度。这个梯度信息的使用可以大幅度加快找到极小值点的速度。使用梯度信息构成了训练神经网络的实际算法的基础。

梯度下降最优化

最简单的使用梯度信息的方法是，每次权值更新都是在负梯度方向上的一次小的移动，即

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla E(\boldsymbol{w}^{(\tau)}) \tag{7.12}$$

这种方法被称为梯度下降法 (gradient descent) 或者最陡峭下降法 (steepest descent)。虽然这种方法在直觉上看比较合理，但是实际上可以证明它是一个很差的算法。对于批量最优化方法，存在更高效的方法，例如共轭梯度法 (conjugate gradient) 或者拟牛顿法 (quasi-Newton)。这些方法具有这样的性质：误差函数在每次迭代时总是减小的，除非权向量到达了局部的或者全局的最小值。

为了找到一个足够好的极小值，可能有必要多次运行基于梯度的算法，每次都使用一个不同的随机选择额起始点，然后在一个独立的验证集上对比最终的表现。

梯度下降法有一个在线的版本，这个版本被证明在实际应用中对于使用大规模数据集来训练神经网络的情形很有用。基于一组独立观测的最大似然函数的误差函数由一个求和式构成，求和式的每一项都对应着一个数据点

$$E(\boldsymbol{w}) = \sum_{n=1}^{N} E_n(\boldsymbol{w}) \tag{7.13}$$

在线梯度下降，也被称为顺序梯度下降 (sequential gradient descent) 或者随机梯度下降 (stochasitc gradient descent)，使得权向量的更新每次只依赖于一个数据点，即

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla E_n(\boldsymbol{w}^{(\tau)}) \tag{7.14}$$

这个更新在数据集上循环重复进行，并且即可以顺序地处理数据，也可以随机地有重复地选择数据点。当然，也有折中的方法，即每次更新依赖于数据点的一小部分。

与批处理相比，在线方法的一个优点是可以更加高效地处理数据中的冗余性。在线梯度下隆方法的另一个性质是，可以逃离局部极小值点，因为整个数据集的关于误差函数的驻点通常不会是每个数据点各自的驻点。

7.3 误差反向传播

本节中，我们的目标是寻找一种计算前馈神经网络的误差函数 $E(\boldsymbol{w})$ 的梯度的一种高效的方法。在局部信息传递的思想中，信息在神经网络中交替地向前、向后传播。这种方法被称为误差反向传播 (error backpropagation)，有时简称“反传”(backprop)。

为了不让概念发生混淆，仔细研究一下训练过程的本质是很有用的。大部分训练算法涉及到一个迭代的步骤用于误差函数的最小化，以及通过一系列的步骤进行的权值调节。在每一个迭代过程中，我们可以区分这两个不同的阶段。

1. 第一个阶段，误差函数关于权值的导数必须被计算出来。
2. 第二个阶段，导数用于计算权值的调整量。

反向传播方法的一个重要的贡献是提供了计算这些导数的一个高效的方法。由于正是这个阶段，误差通过网络进行反向传播，因此我们将专门使用反向传播这个术语来描述计算导数的过程。

误差函数导数的计算

我们现在推导适用于一般神经网络的反向传播算法。许多实际应用中使用的误差函数，例如针对一组独立同分布的数据的最大似然方法定义的误差函数，由若干的求和式组成，每一项对应于训练集的一个数据点，即

$$E(\boldsymbol{w}) = \sum_{n=1}^{N} E_n(\boldsymbol{w}) \tag{7.15}$$

这里，我们要考虑的是计算 $\nabla E_n(\boldsymbol{w})$ 的问题。这可以使用顺序优化的方法计算，或者使用批处理方法在训练集上进行累加。

首先考虑一个简单的线性模型，其中输出 y_k 是输入变量 x_i 的线性组合，即，

$$y_k = \sum_i w_{ki} x_i \tag{7.16}$$

对于一个特定的输入模式 n，误差函数的形式为

$$E_n = \frac{1}{2} \sum_k (y_{nk} - t_{nk})^2 \tag{7.17}$$

其中 $y_{nk} = y_k(\boldsymbol{x}_n, \boldsymbol{w})$。这个误差函数关于一个权值 w_{ji} 的梯度为

$$\frac{\partial E_n}{\partial w_{ji}} = (y_{nj} - t_{nj}) x_{ni} \tag{7.18}$$

它可以表示为链接 w_{ji} 的输出端相关联的“误差信号” $y_{nj} - t_{nj}$ 和与链接的输入端相关联的变量 x_{nj} 的乘积。反向传播算法可以总结如下

1. 对于网络的一个输入向量 $\boldsymbol{x}_n$，使用下列进行正向传播，找到所有隐含单元和输出单

元的激活。

$$a_j = \sum_i w_{ji} z_i \tag{7.19}$$

$$z_j = h(a_j) \tag{7.20}$$

其中 z_i 是一个单元的激活，或者是输入。它向单元 j 发送一个链接，w_{ji} 是与这个链接关联的权值。

2. 计算所有输出单元的 δ_k

$$\delta_k = y_k - t_k \tag{7.21}$$

3. 获得网络中所有隐含单元的 δ_j

$$\begin{aligned} \delta_j \equiv \frac{\partial E_n}{\partial a_j} &= \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \\ &= h'(a_j) \sum_k w_{kj} \delta_k \end{aligned} \tag{7.22}$$

其中求和式的作用对象是所有向单元 j 发送链接的单元 k。注意，单元 k 可以包含其他的隐含单元和输出单元。

4. 计算导数

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i \tag{7.23}$$

对于批处理方法，总误差函数 E 的导数可以通过下面的方式得到：对于训练集里的每个模式，重复上面的步骤，然后对所有的模式求和，即

$$\frac{\partial E}{\partial w_{ji}} = \sum_n \frac{\partial E_n}{\partial w_{ji}} \tag{7.24}$$

上面的推导中，我们隐式地假设网络中的每个隐含单元或输入单元相同的激活函数 $h(\cdot)$。

一个简单的例子

上面对于反向传播算法的推导适用于一般形式的误差函数、激活函数、以及网络拓扑结构。为了说明这个算法的应用，我们考虑一个具体的例子。具体地，我们考虑两层神经网络，误差函数为平方和误差函数，输出单元的激活函数为线性激活函数，即 $y_k = a_k$，而隐含单元的激活函数为 S 型函数，形式为

$$h(a) \equiv \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \tag{7.25}$$

这个函数的一个有用的特征是，它的导数可以表示成一个相当简单的形式

$$h'(a) = 1 - h(a)^2 \tag{7.26}$$

也考虑一个标准的平方和误差函数，即对于模式 n，误差为

$$E_n = \frac{1}{2}\sum_{k=1}^{K}(y_k - t_k)^2 \tag{7.27}$$

其中，对于一个特定的输入模式 $\boldsymbol{x}_n$，y_k 是输出单元 k 的激活，t_k 是对应的目标值。

对于训练集里的每个模式，我们首先使用下面的公式进行前向传播。

$$a_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i \tag{7.28}$$

$$z_j = \tanh(a_j) \tag{7.29}$$

$$y_k = \sum_{j=0}^{M} w_{kj}^{(2)} z_j \tag{7.30}$$

接下来，使用下面的公式计算每个输出单元的 δ 值

$$\delta_k = y_k - t_k \tag{7.31}$$

然后，将这些 δ 值反向传播，得到隐含单元的 δ 值

$$\delta_j = (1 - Z_j^2)\sum_{k=1}^{K} w_{kj}\delta_k \tag{7.32}$$

最后，关于第一层权值和第二层权值的导数为

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j x_i, \quad \frac{\partial E_n}{\partial w_{ki}^{(2)}} = \delta_k z_i \tag{7.33}$$

反向传播的效率

反向传播的一个重要的方面是它的计算效率。考察误差函数导数的计算次数与网络中权值和偏置总数 W 的关系。

Jacobian 矩阵

误差反向传播技术也可以用来计算其他类型的导数。考虑 Jacobian 矩阵的计算，它的元素的值是网络的输出关于输入的导数

$$J_{ki} \equiv \frac{\partial y_k}{\partial x_i} \tag{7.34}$$

其中，计算每个这样的导数时，其他的输入都固定。Jacobian 矩阵在由许多不同模块构建的系统中很有用。Jacobian 矩阵度量了输出对于每个输入变量的改变的敏感性，因此它也允许与输入关联的任意已知的误差 $\triangle x_i$ 在训练过的网络中传播，从而估计他们对于输出

误差 $\triangle y_k$ 的贡献。

7.4 Hessian 矩阵

反向传播也可以用来计算误差函数的二阶导数，形式为

$$\frac{\partial^2 E}{\partial w_{ji} \partial w_{lk}} \tag{7.35}$$

有时将所有的权值和偏置参数看成一个向量 (记作 $\boldsymbol{w}$) 的元素 w_i 更方便，此时二阶导数组成了 Hessian 矩阵 $\boldsymbol{H}$ 的元素 H_{ij}，其中 $i, j \in \{1, \ldots, W\}$，且 W 是权值和偏置的总数。Hessian 矩阵在神经网络计算的许多方面都有着重要的作用，包括

1. 一些用来训练神经网络的非线性最优化算法是基于误差曲面的二阶性质的，这些性质由 Hessian 矩阵控制。
2. 对于训练数据的微小改变，Hessian 矩阵构成了快速重新训练前馈网络的算法的基础。
3. Hessian 矩阵的逆矩阵用来鉴别神经网络中最不重要的权值，这是网络“剪枝”算法的一部分。
4. Hessian 矩阵是贝叶斯神经网络的拉普拉斯近似的核心。它的逆矩阵用来确定训练过的神经网络的预测分布，它的特征值确定了超参数的值，它的行列式用来计算模型证据。

计算神经网络的 Hessian 矩阵有很多近似方法，然而，使用反向传播方法的一个扩展，Hessian 矩阵可以精确地被计算出来。

对角近似

Hessian 矩阵的一些应用需要求出 Hessian 矩阵的逆矩阵，而不是 Hessian 矩阵本身。因此，我们对 Hessian 矩阵的对角化近似比较感兴趣。换句话说，就是把非对角线上的元素置为零，因此这样做之后，矩阵的逆矩阵很容易计算。Hessian 矩阵的对角线元素可以写成

$$\begin{aligned} \frac{\partial^2 E_n}{\partial w_{ji}^2} &= \frac{\partial^2 E_n}{\partial a_j^2} \frac{\partial a_j}{\partial w_{ji}} z_i + 0 \\ &= \frac{\partial^2 E_n}{\partial a_j^2} z_i^2 \end{aligned} \tag{7.36}$$

公式右侧的二阶导数可以通过递归地使用微分的链式法则的方式求出。这样，可以得到反向传播方程的形式为

$$\begin{aligned}\frac{\partial^2 E_n}{\partial a_j^2} &= \frac{\partial}{\partial a_j}\left[\sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial a_j}\right] \\ &= \frac{\partial}{\partial a_j}\left[h'(a_j)\sum_k w_{kj}\frac{\partial E_n}{\partial a_k}\right] \\ &= h'(a_j)^2\sum_k\sum_{k'} w_{kj}w_{k'j}\frac{\partial^2 E_n}{\partial a_k \partial a_{k'}} + h''(a_j)\sum_k w_{kj}\frac{\partial E_n}{\partial a_k}\end{aligned} \tag{7.37}$$

如果忽略二阶导数中非对角线无素，那么我们有

$$\frac{\partial^2 E_n}{\partial a_j^2} = h'(a_j)^2\sum_k w_{kj}^2\frac{\partial^2 E_n}{\partial a_k^2} + h''(a_j)\sum_k w_{kj}\frac{\partial E_n}{\partial a_k} \tag{7.38}$$

对角近似的主要问题是，在实际应用中 Hessian 矩阵通常是强烈非对角化的，因此为了计算方便而采取的这些近似手段必须谨慎使用。

外积近似

当神经网络应用于回归问题时，通常使用平方和误差函数。我们可以把 Hessian 矩阵写成下面的形式

$$\begin{aligned}\boldsymbol{H} = \nabla\nabla E &= \frac{\partial^2}{\partial y_n^2}\left[\frac{1}{2}\sum_{n=1}^N (y_n - t_n)^2\right] \\ &= \frac{\partial}{\partial y_n}\left[\nabla y_n \sum_{n=1}^N (y_n - t_n)\right] \\ &= \sum_{n=1}^N \nabla y_n (\nabla y_n)^T + \sum_{n=1}^N (y_n - t_n)\nabla\nabla y_n\end{aligned} \tag{7.39}$$

如果网络已经在数据集上训练过，输出 y_n 恰好非常接近 t_n，那么公式的第二项会很小，可以被忽略。我们就得到了 Levenberg-Marquardt 近似，或者称为外积近似 (outer product approximation)。形式为

$$\boldsymbol{H} \simeq \sum_{n=1}^N \boldsymbol{b}_n \boldsymbol{b}_n^T \tag{7.40}$$

其中 $\boldsymbol{b}_n \equiv \nabla a_n = \nabla y_n$，因为输出单元的激活函数就是恒等函数。Hessian 矩阵近似的计算是很容易的，因为它只涉及到误差函数的一阶导数，这可能通过使用标准的反向传播算法在 $O(W)$ 个步骤内高效地求出。需要强调的是，这种近似只在网络被恰当地训练时才成立，对于一个一般的网络映射，公式右侧的二阶导数项通常不能忽略。

在误差函数为交叉熵误差函数，输出单元激活函数为 logistic sigmoid 函数的神经网络中，对应的近似为

$$\boldsymbol{H} \simeq \sum_{n=1}^N y_n(1 - y_n)\boldsymbol{b}_n \boldsymbol{b}_n^T \tag{7.41}$$

对于输出函数为 softmax 函数的多类神经网络，可以得到类似的结果。

Hessian 矩阵的逆矩阵

使用外积近似可以提出一个计算 Hessian 矩阵的逆矩阵的高效方法。首先，我们用矩阵的记号写出外积近似，即

$$\boldsymbol{H} = \sum_{n=1}^{N} \boldsymbol{b}_n \boldsymbol{b}_n^T \tag{7.42}$$

其中，$\boldsymbol{b}_n \equiv \nabla_{\boldsymbol{w}} a_n$ 是数据点 n 产生的输出单元激活对梯度的贡献。

现上推导一个建立 Hessian 矩阵的顺序步骤，每次处理一个数据点。假设我们已经使用前 L 个数据点得到了 Hessian 矩阵的逆矩阵。通过将第 L+1 个数据点的贡献单独写出来，我们有

$$\boldsymbol{H}_{L+1} = \boldsymbol{H}_L + \boldsymbol{b}_{L+1} \boldsymbol{b}_{L+1}^T \tag{7.43}$$

为了计算 Hessian 矩阵的逆矩阵，我们考虑 Sherman-Morrison-Woodbury 公式

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I_k + V^T A^{-1} U)^{-1} V^T A^{-1} \tag{7.44}$$

其中，$A \in \mathbb{R}^{n \times n}$ 非奇异，即 A^{-1} 存在，$U, V \in \mathbb{R}^{n \times k}$。当 $k = 1$ 时的特殊形式

$$\begin{aligned} (A + uv^T)^{-1} &= A^{-1} - A^{-1}u(1 + v^T A^{-1} u)^{-1} v^T A^{-1} \\ &= A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^T A^{-1} u} \end{aligned} \tag{7.45}$$

SM 公式看似复杂，但可以通过求解以下线性方程组来推导出来：

$$(A + uv^T)x = b \tag{7.46}$$

式 7.46 两边同乘以 A^{-1}，令 $A^{-1}u = z, A^{-1}b = y$，则有

$$x + zv^T x = y \tag{7.47}$$

注意到 $V^T x$ 是标量，令 $a = v^T x$。式 7.47 两边同时乘以 v^T，得

$$a + v^T z a = v^T y \tag{7.48}$$

由于式 7.48 中 $v^T z$ 和 $v^T y$ 都是标量，从而由式 7.48 可解得

$$a = \frac{v^T y}{1 + v^T z} \tag{7.49}$$

由式 7.47 和 y, z, a 的定义可得

$$\begin{aligned} x &= y - az \\ &= A^{-1}b - A^{-1}u(1 + v^T A^{-1}u)^{-1}v^T A^{-1}b \\ &= \left[A^{-1} - A^{-1}u(1 + v^T A^{-1}u)^{-1}v^T A^{-1}\right] b \end{aligned} \tag{7.50}$$

由 7.46 和 7.50 即可得 Sherman-Morrison 公式，即 7.45。

证明： 公式 7.44 两边同乘 $(A + UV^T)$

$$\begin{aligned} I_n &= (A + UV^T)\left[A^{-1} - A^{-1}U(I_k + V^T A^{-1}U)^{-1}V^T A^{-1}\right] \\ &= I_n + UV^T A^{-1} - U(I_k + V^T A^{-1}U)^{-1}V^T A^{-1} - UV^T A^{-1}U(I_k + V^T A^{-1}U)^{-1}V^T A^{-1} \\ &= I_n + UV^T A^{-1} - U(I_k + V^T A^{-1}U)(I_k + V^T A^{-1}U)^{-1}V^T A^{-1} \\ &= I_n \end{aligned} \tag{7.51}$$

Woodbury 恒等式的主要用途是当 $n > k$ 时把一个相对较大的 n 阶矩阵求逆的问题转化为求一个相对较小的 k 阶矩阵的逆矩阵。我们可以把公式看成在已知 A^{-1} 的情况下对 $A + \triangle A$ 求逆的工具。其中 $\triangle A$ 是一个低秩扰动。

如果我们令 $\boldsymbol{H}_L = A$，且 $\boldsymbol{b}_{L+1} = \boldsymbol{v}$，我们有

$$\boldsymbol{H}_{L+1}^{-1} = \boldsymbol{H}_L^{-1} - \frac{\boldsymbol{H}_L^{-1}\boldsymbol{b}_{L+1}\boldsymbol{b}_{L+1}^T\boldsymbol{H}_L^{-1}}{1 + \boldsymbol{b}_{L+1}^T\boldsymbol{H}_L^{-1}\boldsymbol{b}_{L+1}} \tag{7.52}$$

使用这种方式，数据点可以依次使用，直到 $L + 1 = N$，整个数据集被处理完毕。于是，这个结果表示一个计算 Hessian 矩阵的逆矩阵的算法，这个算法只需对数据集扫描一次。最开始的矩阵 $\boldsymbol{H}_0$ 被选为 $\alpha\boldsymbol{I}$，其中 α 是一个较小的量，从而算法实际找的是 $\boldsymbol{H} + \aleph\boldsymbol{I}$ 的逆矩阵。结果对于 α 的精确值不是特别敏感。

有限差

与误差函数的一阶导数的形式相同，我们可以使用有限差的方法求二阶导数，精度受数值计算的精度限制。

Hessian 矩阵的精确计算

对于一个任意的前馈拓扑结构的网络，Hessian 矩阵也可以精确地计算。计算的方法是使用反向传播算法计算一阶导数的推广，同时也保留了计算一阶导数的方法的许多良好的性质，包括计算效率。这种方法可以应用于任何可微的可以表示成网络输出的函数形式的误差函数，以及任何具有可微的激活函数的神经网络。

这里我们考虑一个具体的情况，即具有两层权值的网络。这种网络中待求的方程很容易推导。我们将使用下标 i 和 i' 表示输入，用下标 j 和 j' 表示隐含单元，用下标 k 和 k' 表示输出。首先我们定义

$$\delta_k = \frac{\partial E_n}{\partial a_k}, \quad M_{kk'} \equiv \frac{\partial^2 E_n}{\partial a_k \partial a_{k'}} \tag{7.53}$$

其中 E_n 是数据点 n 对误差函数的贡献。于是，这个网络的 Hessian 矩阵可以被看成三个独立的模块，即

1. 两个权值都在第二层

$$\frac{\partial^2 E_n}{\partial w_{kj}^{(2)} \partial w_{k'j'}^{(2)}} = z_j z_{j'} M_{kk'} \tag{7.54}$$

2. 两个权值都在第一层

$$\begin{aligned}\frac{\partial^2 E_n}{\partial w_{ji}^{(1)} \partial w_{j'i'}^{(1)}} &= x_i x_{i'} h''(a_{j'}) I_{jj'} \sum_k w_{kj'}^{(2)} \delta_k \\ &+ x_i x_{i'} h'(a_{j'}) \sum_k \sum_{k'} w_{k'j'}^{(2)} w_{kj}^{(2)} M_{kk'}\end{aligned} \tag{7.55}$$

3. 每一层有一个权值

$$\frac{\partial^2 E_n}{\partial w_{ji}^{(1)} \partial w_{kj'}^{(2)}} = x_i h'(a_j) \left\{ \delta_k I_{j'j} + z^{j'} \sum_{k'} w_{k'j}^{(2)} M_{kk'} \right\} \tag{7.56}$$

这里 $I_{jj'}$ 是单位矩阵的第 j, j' 个元素。如果权值中的一个或者两个偏置项，那么只需将激活设为 1 即可得到对应的表达式。很容易将这个结果推广到允许网络包含跨层链接的情形。

Hessian 矩阵的快速乘法

对于 Hessian 矩阵的许多应用来说，我们感兴趣的不是 Hessian 本身，而是 $\boldsymbol{H}$ 与某些向量 $\boldsymbol{v}$ 的乘积。我们已经看到 Hessian 矩阵的计算需要 $O(W^2)$ 次操作，所需的存储空间也是 $O(W^2)$。但是，我们想要计算的向量 $\boldsymbol{v}^T\boldsymbol{H}$ 只有 W 个元素。因此，我们可以不把计算 Hessian 矩阵当成一个中间的步骤，而是可以尝试寻找一种只需 $O(W)$ 次操作的直接计算 $\boldsymbol{v}^T\boldsymbol{H}$ 的高效方法。

为了完成这一点，我们首先注意到

$$\boldsymbol{v}^T\boldsymbol{H} = \boldsymbol{v}^T \nabla(\nabla E) \tag{7.57}$$

其中 ∇ 表示权空间的梯度算符。然后，我们可以写下计算 ∇E 的标准正向传播和反向传播的方程，然后将公式应用于这些方程，得到一组计算 $\boldsymbol{v}^T\boldsymbol{H}$ 的正向传播和反向传播的方程。这对应于将微分算符 $\boldsymbol{v}^T\nabla$ 作用于原始的正向传播和反向传播的方程。使用记号 $\mathcal{R}\{\cdot\}$ 表示算符 $\boldsymbol{v}^T\nabla$，我们将遵从这个惯例。与之前一样，我们使用两层网络，以及线性的输出单元和平方和误差函数。我们考虑数据集里的一个模式对于误差函数的贡献。这样，我们所要求解的向量可以通过求出每个模式各自的贡献然后求和的方式得到。对于两层神经

网络,正向传播方程为

$$a_j = \sum_i w_{ji} x_i \tag{7.58}$$

$$z_j = h(a_j) \tag{7.59}$$

$$y_k = \sum_j w_{kj} z_j \tag{7.60}$$

我们现在使用 $\mathcal{R}\{\cdot\}$ 作用于这些方程上,得到一组正向传播方程,形式为

$$\mathcal{R}\{a_j\} = \sum_i v_{ji} x_i \tag{7.61}$$

$$\mathcal{R}\{z_j\} = h'(a_j)\mathcal{R}\{a_j\} \tag{7.62}$$

$$\mathcal{R}\{y_k\} = \sum_j w_{kj}\mathcal{R}\{z_j\} + \sum_j v_{kj} z_j \tag{7.63}$$

其中,v_{ji} 是向量 $\boldsymbol{v}$ 中对应于权值 w_{ji} 的元素。

考虑的平方和误差函数,我们有下面的标准的反向传播表达式

$$\delta_k = y_k - t_k \tag{7.64}$$

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k \tag{7.65}$$

与之前一样,我们将 $\mathcal{R}\{\cdot\}$ 作用于这些方程上,得到一组反向传播方程,形式为

$$\mathcal{R}\{\delta_k\} = \mathcal{R}\{y_k\} \tag{7.66}$$

$$\begin{aligned}\mathcal{R}\{\delta_j\} = {} & h''(a_j)\mathcal{R}\{a_j\} \sum_k w_{kj}\delta_k \\ & + h'(a_j) \sum_k v_{kj}\delta_k + h'(a_j) \sum_k w_{kj}\mathcal{R}\{\delta\}\end{aligned} \tag{7.67}$$

然后,我们有误差函数的一阶导数的方程

$$\frac{\partial E}{\partial w_{kj}} = \delta_k z_j \tag{7.68}$$

$$\frac{\partial E}{\partial w_{ji}} = \delta_j x_j \tag{7.69}$$

使用 $\mathcal{R}\{\cdot\}$ 作用在这些方程上,我们得到了下面的关于 $\boldsymbol{v}^T\boldsymbol{H}$ 的表达式

$$\mathcal{R}\left\{\frac{\partial E}{\partial w_{kj}}\right\} = \mathcal{R}\{\delta_k\} z_j + \delta_k \mathcal{R}\{z_j\} \tag{7.70}$$

$$\mathcal{R}\left\{\frac{\partial E}{\partial w_{ji}}\right\} = x_i \mathcal{R}\{\delta_j\} \tag{7.71}$$

算法的执行涉及到将新的变量 $\mathcal{R}\{a_j\},\mathcal{R}\{z_j\},\mathcal{R}\{\delta_j\}$ 引入到隐含单元, 将 $\mathcal{R}\{\delta_k\},\mathcal{R}\{y_k\}$ 引

入到输出单元。对于每个输入模式，这些量的值可以使用上面的结果求出，$\boldsymbol{v}^T\boldsymbol{H}$ 的值由公式 7.70 和公式 7.71 给出。这种方法的一个好处是，计算 $\boldsymbol{v}^T\boldsymbol{H}$ 的方程与标准的正向传播和反向传播的方程相同，因此将现有的神经网络计算程序扩展到能够计算这个乘积通常很容易。

7.5 神经网络的正则化

神经网络的输入单元和输出单元的数量通常由数据集的维度确定，而隐含单元的数量 M 是一个自由的参数，可以通过调节来给出最好的预测性能。M 控制了网络中参数 (权值和偏置) 的数量。然后，泛化误差与 M 的关系不是一个简单的函数关系，因为误差函数中存在局部极小值。有其他的方式控制神经网络的模型复杂度来避免过拟合。一种方法是选择一个相对大的 M 值，然后通过给误差函数增加一个正则化项，来控制模型的复杂度。最简单的正则化项是二次的，给出了正则化的误差函数，形式为

$$\tilde{E}(\boldsymbol{w}) = E(\boldsymbol{w}) + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w} \tag{7.72}$$

这个正则化项也被称为权值衰减 (weight decay)。这样模型复杂度可以通过选择正则化系数 λ 来确定。正则化项可以表示为权值 $\boldsymbol{w}$ 上的零均值高斯先验分布的负对数。

相容的高斯先验

公式 7.72 给出的简单权值衰减的一个局限性是，它与网络映射的确定缩放性质不相容。为了说明这一点，考虑一个多层感知器网络，这个网络有两层权值和线性输出单元，它给出了从输入变量集合 $\{x_i\}$ 到输出变量集合 $\{y_k\}$ 的映射。第一个隐含层的隐含单元的激活的形式为

$$z_j = h\left(\sum_i w_{ji}x_i + w_{j0}\right) \tag{7.73}$$

输出单元的激活为

$$y_k = \sum_j w_{kj}z_j + w_{k0} \tag{7.74}$$

假设我们对输入变量进行一个线性变换，形式为

$$x_i \to \tilde{x}_i = ax_i + b \tag{7.75}$$

然后我们可以根据这个映射对网络进行调整，使得网络给出的映射不变。调整的方法为，对从输入单元到隐含层单元的权值和偏置也进行一个对应的线性变换，形式为

$$w_{ji} \to \tilde{w}_{ji} = \frac{1}{a}w_{ji} \tag{7.76}$$

$$w_{j0} \to \tilde{w}_{j0} = w_{j0} - \frac{b}{a}\sum_i w_{ji} \tag{7.77}$$

类似地，网络的输出变量的线性变换

$$y_k \to \tilde{y}_k = cy_k + d \tag{7.78}$$

可以通过对第二层的权值和偏置进行线性变换的方式实现。变换的形式为

$$w_{kj} \to \tilde{w}_{kj} = cw_{kj} \tag{7.79}$$

$$w_{k0} \to \tilde{w}_{k0} = cw_{k0} + d \tag{7.80}$$

如果我们使用原始数据训练一个网络，还使用输入和 (或) 目标变换进行了上面的线性变换的数据训练一个网络，那么相容性要求这两个网络应该是等价的，差别仅在于上面给出的权值的线性变换。任何正则化项都应该与这个性质相容，否则模型就会倾向于选择某个解，而忽视某个等价的解。显然，简单的权值衰减由于把所有的权值和偏置同等对待，因此不满足这个性质。

于是我们要寻找一个正则化项，它在线性变换下具有不变性。这需要正则化项应该对于权值的重新缩放不变，对于偏置的平移不变，这样的正则化项为

$$\frac{\lambda_1}{2}\sum_{w\in\mathcal{W}_1} w^2 + \frac{\lambda_2}{2}\sum_{w\in\mathcal{W}_2} w^2 \tag{7.81}$$

其中 $\mathcal{W}_1$ 表示第一层的权值集合，$\mathcal{W}_2$ 表示第二层的权值集合，偏置未出现在求和式中。这个正则化项在权值的变换下不会发生变化，只要正则化参数进行下面的重新放缩即可：$\lambda_1 \to a^{\frac{1}{2}}\lambda_1, \lambda_2 \to c^{-\frac{1}{2}}\lambda_2$

正则化项 7.81 对应于下面形式的先验概率分布

$$p(\boldsymbol{w}|\alpha_1,\alpha_2) \propto exp\left(-\frac{\alpha_1}{2}\sum_{w\in\mathcal{W}_1} w^2 - \frac{\alpha_2}{2}\sum_{w\in\mathcal{W}_2} w^2\right) \tag{7.82}$$

注意，这种形式的先验是反常的 (improper)(不能被归一化)，因为偏置参数没有限制。使用反常先验会给正则化系数的选择造成很大的困难，也会给贝叶斯框架下的模型选择造成很大的困难，因为对应的模型证据等于零。因此，通常的做法是单独包含一个有着自己单独的一套超参数的偏置的先验 (这就破坏了平移不变性)。

早停止

另一种控制网络的复杂度的正则化方法是早停止 (early stopping)。非线性网络模型的训练对应于误差函数的迭代减小，其中误差函数是关于训练数据集定义的。对于许多用于网络训练的最优化算法，误差函数是一个关于迭代次数的不增函数。然而，在独立数据 (通常被称为验证集) 上测量的误差，通常首先减小，接下来由于模型开始过拟合而逐渐增大。于是训练过程可以在关于验证集误差最小的点停止，这样可以得到一个有着较好泛化性能的网络。

不变性

在许多模型识别的应用中，在对于输入变量进行了一个或者多个变换之后，预测不应该发生变化，或者说应用具有不变性 (invariant)。如果可以得到足够多的训练模式，那么可调节的模型 (例如神经网络) 可以学习到不变性，至少可以近似地学习到。这涉及到训练集里包含足够多的表示各种变换的效果的样本。如果训练样本数受限，或者有多个不变性 (变换的组合的数量随着变换的数量指数增长)，那么这种方法就很不实用。于是，我们要寻找另外的方法来让可调节的模型能够表述所需的不变性。这此方法大致可以分为四类。

1. 通过复制训练模式，同时根据要求的不变性进行变换，对训练集进行扩展。
2. 为误差函数加上一个正则化项，用来惩罚当输入进行变换时，输出发生的改变。这引出了切线传播方法。
3. 通过抽取在要求的变换下不发生改变的特征，不变性被整合到预处理过程中。任何后续的使用这些特征作为输入的回归或者分类系统就会具有不变性。
4. 把不变性的性质整合到神经网络的构建过程中，或者对于相关向量机的方法，整合到核函数中。一种方法是通过使用局部接收场和共享权值。如卷积神经网络。

切线传播

通过切线传播 (tangent propagation) 的方法，我们可以使用正则化来让模型对于输入的变换具有不变性。

涉及到微分几何，待整理！

用变换后的数据训练

涉及到微分几何，待整理！

卷积神经网络

另一种构造对输入变量的变换具有不变性的模型的方法是将不变性的性质融入到神经网络结构的构建中。这就是卷积神经网络 (convolutional neural network) 的基础，它被广泛地应用于图像处理领域。

考虑手写数字识别这个具体的任务。我们知道，数字的各类对于平移、缩放以及旋转具有不变性。一种简单的方法是把图像作为一个完全链接的神经网络的输入。假如数据集充分大，那么这样的网络原则上可以产生这个问题的一个较好的解，从而可以从样本中学习到恰当的不变性。然而，这种方法忽略了图像的一个关键性质，即距离较近的像素的相关性要远大于距离较远的像素的相关性。这些想法被整合到了卷积神经网络中，通过下面的三种方式：

(1) 局部接收声场
(2) 权值共享
(3) 下采样

软权值共享

降低具有大量权值参数的网络复杂度的一种方法是将权值分组，然后令分组内的权值相等。这里，我们考虑软权值共享 (soft weight sharing)。这种方法中，权值相等的硬限制被替换为一种形式的正则化，其中权值的分组倾向于取近似的值。此外，权值的分组、每组权值的均值，以及分组内的取值范围全都作为学习过程的一部分被确定。

简单的权值衰减正则化项可以被看成权值上的高斯分布的负对数。我们可以将权值分成若干组，而不是将所有权值分为一个组。分组的方法是使用高斯混合概率分布。混合分布中，每个高斯分量的均值、方差，以及混合系数，都会作为可调节的参数在学习过程中被确定。于是，我们有下面形式的概率密度

$$p(\boldsymbol{w}) = \prod_i p(w_i) \tag{7.83}$$

其中

$$p(w_i) = \sum_{j=1}^{M} \pi_j \mathcal{N}(w_i|\mu_j, \sigma_j^2) \tag{7.84}$$

π_j 为混合系数。取负对数，即可得到正则化函数，形式为

$$\Omega(\boldsymbol{w}) = -\sum_i \ln\left(\sum_{j=1}^{M} \pi_j \mathcal{N}(w_i|\mu_j, \sigma_j^2)\right) \tag{7.85}$$

从而，总的误差函数为

$$\tilde{E}(\boldsymbol{w}) = E(\boldsymbol{w}) + \lambda\Omega(\boldsymbol{w}) \tag{7.86}$$

其中，λ 是正则化系数。这个误差函数同时关于权值 w_i 和混合模型参数 $\{\pi_j, \mu_j, \sigma_j\}$ 进行最小化。如果权值是常数，那么混合模型的参数可以由 EM 算法确定。然而，权值分布本身在学习过程中是不断变化的，因此为了避免数值的不稳定性，我们同时关于权值和混合模型参数进行最优化。可以使用标准的最优化算法来完成这件事情。

为了最小化总的误差函数，难免计算出它关于各个可调节参数的参数是很有必要的。为了完成这一点，比较方便的做法是把 $\{\pi_j\}$ 当成先验概率，然后引入对应的后验概率。后验概率由贝叶斯定理给出，形式为

$$\gamma_j(w) = \frac{\pi_j \mathcal{N}(w_i|\mu_j, \sigma_j^2)}{\sum_k \pi_k \mathcal{N}(w_i|\mu_k, \sigma_k^2)} \tag{7.87}$$

这样，总的误差函数关于权值的导数为

$$\begin{aligned} \frac{\partial \tilde{E}}{\partial w_i} &= \frac{\partial E}{\partial w_i} + \lambda \frac{\partial \Omega}{\partial w_i} \\ &= \frac{\partial E}{\partial w_i} + \lambda \sum_j \gamma_j(w_i) \frac{(w_i - \mu_j)}{\sigma_j^2} \end{aligned} \tag{7.88}$$

正则化项的效果是把每个权值拉向第 j 个高斯分布的中心，拉力正比于给定权值的高斯分布的后验概率。

误差函数关于高斯分布的中心的导数

$$\frac{\partial \tilde{E}}{\partial \mu_j} = \lambda \sum_i \gamma_j(w_i) \frac{(\mu_j - w_i)}{\sigma_j^2} \tag{7.89}$$

效果是把 μ_j 拉向了权值的平均值，拉力为第 j 个高斯分量产生的权值参数的后验概率。

关于方差的导数为

$$\frac{\partial \tilde{E}}{\partial \sigma_j} = \lambda \sum_i \gamma_j(w_i) \left(\frac{1}{\sigma_j} - \frac{(w_i - \mu_j)^2}{\sigma_j^3} \right) \tag{7.90}$$

效果是将 σ_j 拉向权值在对应的中心 μ_j 附近的偏差的平方的加权平均，加权平均的权系数与之前一样，等于由第 j 个高斯分量产生的权值参数的后验概率。

关于混合系数 π_j 的导数，我们需要考虑下面的限制条件

$$\sum_j \pi_j = 1, \quad 0 \leqslant \pi_i \leqslant 1 \tag{7.91}$$

这个限制的产生，是因为我们把 π_i 看成了先验概率。可以这样做：将混合系数通过一组辅助变量 $\{\eta_j\}$ 用 softmax 函数表示，即

$$\pi_j = \frac{exp(\eta_j)}{\sum_{k=1}^{M} exp(\eta_k)} \tag{7.92}$$

这样，正则化的误差函数关于 $\{\eta_j\}$ 的导数的形式为

$$\frac{\partial \tilde{E}}{\partial \eta_j} = \sum_i \{\pi_j - \gamma_j(w_i)\} \tag{7.93}$$

效果是 π_j 被拉向第 j 个高斯分量的平均后验概率。

7.6 混合密度网络

7.7 贝叶斯神经网络

目前为止，我们对于神经网络的讨论集中于使用最大似然方法来确定网络的参数 (权值和偏置)。正则化的最大似然方法可以看成 MAP(maximum posterior) 方法，其中正则化项可以被看成先验参数分布的对数。然而，在贝叶斯方法中，为了进行预测，我们需要对参数的概率分布进行积分或求和。

在多层神经网络的情况下，网络函数对于参数值的高度非线性的性质意味着精确的贝叶斯方法不再可行，事实上，后验概率分布的对数是非凸的，对应于误差函数中的多个

局部极小值。

变分推断方法已经被用在了贝叶斯神经网络中，这种方法使用了对后验概率的分解的高斯近似，也使用了一个具有完成协方差矩阵的高斯分布。但是，最完整的贝叶斯方法是基于拉普拉斯的方法，这种方法构成了本节讨论的基础。我们会使用一个以真实后验概率的众数为中心的高斯分布来近似后验概率分布。此外，我们会假设这个高斯分布的协方差很小，从而网络函数关于参数空间的区域中的参数近似是线性关系。在参数空间中，后验概率距离概率为零的状态相当远。使用这两个近似，我们会得到与之前讨论的线性回归和线性分布的模型相类似的模型，从而我们就可以利用之前得到了结果了。这样，我们可以使用模型证据的框架来对参数进行点估计，并且比较不同的模型。

后验参数分布

考虑从输入向量 $\boldsymbol{x}$ 预测单一连续目标变量 t 的问题。我们假设条件概率分布 $p(t|\boldsymbol{x})$ 是一个高斯分布，均值与 $\boldsymbol{x}$ 有关，由神经网络模型的输出 $y(\boldsymbol{x},\boldsymbol{w})$ 确定，精度 β 为

$$p(t|\boldsymbol{x},\boldsymbol{w},\beta)=\mathcal{N}(t|y(\boldsymbol{x},\boldsymbol{w}),\beta^{-1}) \tag{7.94}$$

类似地，我们将权值 $\boldsymbol{w}$ 的先验概率分布选为高斯分布，形式为

$$p(\boldsymbol{w}|\alpha)=\mathcal{N}(\boldsymbol{w}|0,\alpha^{-1}\boldsymbol{I}) \tag{7.95}$$

对于 N 次独立同分布的观测 $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N$，对应的目标值集合 $\mathcal{D}=\{t_1,\ldots,t_N\}$，似然函数为

$$p(\mathcal{D}|\boldsymbol{w},\beta)=\prod_{n=1}^{N}\mathcal{N}(t_n|y(\boldsymbol{x}_n,\boldsymbol{w}),\beta^{-1}) \tag{7.96}$$

因此最终的后验概率为

$$p(\boldsymbol{w}|\mathcal{D},\alpha,\beta)\propto p(\boldsymbol{w}|\alpha)p(\mathcal{D}|\boldsymbol{w},\beta) \tag{7.97}$$

由于 $y(\boldsymbol{w},\boldsymbol{x})$ 与 $\boldsymbol{w}$ 的关系是非线性的，因此后验概率不是高斯分布。

使用拉普拉斯近似，我们可以找到对于后验概率分布的一个高斯近似。为了完成这一点，我们必须首先找到后验概率分布的一个 (局部) 最大值，这必须使用迭代的数值最优化算法才能找到。比较方便的做法是最大化后验概率分布的对数，它可以写成下面的形式

$$\ln p(\boldsymbol{w}|\mathcal{D})=-\frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w}-\frac{\beta}{2}\sum_{n=1}^{N}\{y(\boldsymbol{x}_n,\boldsymbol{w})-t_n\}^2+\text{常数} \tag{7.98}$$

这对应于一个正则化的平方和误差函数。假设 α 和 β 都是定值，那么我们可以通过标准的非线性最优化算法，使用误差反向传播计算所需的导数，找到后验概率的最大值。我们将最大值的位置记作 $\boldsymbol{w}_{MAP}$

找到了众数，我们就可以通过计算后验概率分布的负对数的二阶导数，建立一个局部

的高斯近似。负对数后验概率的二阶导数为

$$\boldsymbol{A} = -\nabla\nabla \ln p(\boldsymbol{w}|\mathcal{D}, \alpha, \beta) = \alpha\boldsymbol{I} + \beta\boldsymbol{H} \tag{7.99}$$

这里，$\boldsymbol{H}$ 是一个 Hessian 矩阵，由平方和误差函数关于 $\boldsymbol{w}$ 的分量组成。这样，后验概率对应的高斯近似形式为

$$q(\boldsymbol{w}|\mathcal{D}) = \mathcal{N}(\boldsymbol{w}_{MAP}, \boldsymbol{A}^{-1}) \tag{7.100}$$

类似地，预测分布可以通过将后验概率分布求积分的方式获得

$$p(t|\boldsymbol{x}, \mathcal{D}) = \int p(t|\boldsymbol{x}, \boldsymbol{w}) q(\boldsymbol{w}|\mathcal{D}) d\boldsymbol{w} \tag{7.101}$$

然而，即使对于后验分布的高斯近似，这个积分仍然无法得到解析解，因为网络函数 $y(\boldsymbol{x}, \boldsymbol{w})$ 与 $\boldsymbol{w}$ 的关系是非线性的。为了将计算过程进行下去，我们现在假设，与 $y(\boldsymbol{x}, \boldsymbol{w})$ 发生变化造成的 $\boldsymbol{w}$ 幅度相比，后验概率分布的方差较小。这使得我们可以在 $\boldsymbol{w}_{MAP}$ 附近对网络函数进行泰勒展开。只保留展开式的现行项，可得

$$y(\boldsymbol{x}, \boldsymbol{w}) \simeq y(\boldsymbol{x}, \boldsymbol{w}_{MAP}) + \boldsymbol{g}^T(\boldsymbol{w} - \boldsymbol{w}_{MAP}) \tag{7.102}$$

其中，我们定义了

$$\boldsymbol{g} = \nabla_{\boldsymbol{w}} y(\boldsymbol{w}, \boldsymbol{x})|_{\boldsymbol{w}=\boldsymbol{w}_{MAP}} \tag{7.103}$$

使用这个近似，我们现在得到了一个线性高斯模型，$p(\boldsymbol{w})$ 为高斯分布。并且，$p(t|\boldsymbol{w})$ 也是高斯分布，它的均值是 $\boldsymbol{w}$ 的线性函数，分布的形式为

$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) \simeq \mathcal{N}(t|y(\boldsymbol{x}, \boldsymbol{w}_{MAP}) + \boldsymbol{g}^T \underbrace{(\boldsymbol{w} - \boldsymbol{w}_{MAP})}_{\text{只有这里含有 } \boldsymbol{w}}, \beta^{-1}) \tag{7.104}$$

于是，我们我们可以求出边缘分布 $p(t)$

$$p(t|\boldsymbol{x}, \mathcal{D}, \alpha, \beta) = \mathcal{N}(t|y(\boldsymbol{x}, \boldsymbol{w}_{MAP}), \sigma^2(\boldsymbol{x}) \tag{7.105}$$

其中，与输入相关的方差为

$$\sigma^2(\boldsymbol{x}) = \beta^{-1} + \boldsymbol{g}^T\boldsymbol{A}^{-1}\boldsymbol{g} \tag{7.106}$$

我们看到预测分布 $p(t|\boldsymbol{x}, \mathcal{D})$ 是一个高斯分布，它的均值由网络函数 $y(\boldsymbol{x}, \boldsymbol{w}_{MAP})$ 给出，参数设置为了 MAP 值。方差由两项组成。第一项来自目标变量的固有噪声，第二项是一个与 $\boldsymbol{x}$ 相关的项，表示由于模型参数 $\boldsymbol{w}$ 的不确定性造成的内插的不确定性。

超参数最优化

用于分类的贝叶斯神经网络

第 8 章 支持向量机

支持向量机(support vector machines,SVM)是一种二类分类模型。它的基本模型是定义在特征空间上的间隔最大的线性分类器,间隔最大使它有别于感知机;支持向量机还包括核技巧,这使它成为实质上的非线性分类器。支持向量机的学习策略就是间隔最大化,可形式化为一个求解凸二次规划(convex quadratic programming)的问题,也等价于正则化的合页损失函数的最小化问题。支持向量机的学习算法是求解凸二次规划的最优化算法。

支持向量机学习方法包含构建由简至繁的模型:线性可分支持向量机(linear support vector machine in linearly separable case)、线性支持向量机(linear support vector machine)及非线性支持向量机(non-linear support vector machine)。简单模型是复杂模型的基础,也是复杂模型的特殊情况。当训练数据线性可分时,通过软件间隔最大化(hard margin maximization),学习一个线性的分类器,即线性可分支持向量机,又称为硬间隔支持向量机;当训练数据近似线性可分时,通过软件间隔最大化(soft margin maximization),也学习一个线性的分类器,即线性支持向量机,又称谓软间隔支持向量机;当训练数据线性不可分时,通过使用核技巧(kernel trick)及软间隔最大化,学习非线性支持向量机。

当输入空间为欧氏空间或离散集合、特征空间为希尔伯特空间时,核函数(kernel function)表示将输入从输入空间映射到特征空间得到的特征向量之间的内积。通过使用核函数可以学习非线性支持微量机,等价于隐式地在高维的特征空间中学习线性支持向量机。这样的方法称为核技巧。核方法(kernel method)是比支持向量机更为一般的机器学习方法。

Cortes 与 Vapnik 提出线性支持向量机,Boser、Guyon 与 Vapnik 又引入核技巧,提出非线性支持向量机。

8.1 间隔与支持向量

给定训练样本集 $D = \{(x_1, y_1), \ldots, (x_m, y_m)\}, y_i \in \{-1, +1\}$,分类学习最基本的想法是基于训练集 D 在样本空间中找到一个划分超平面,将不同类别的样本分开。如图 8.1

样本空间中任意点 $\boldsymbol{x}$ 到超平面 $(\boldsymbol{w}, b)$ 的距离可写为

$$\gamma = \frac{y(w^T x + b)}{\|w\|} = \frac{y f(x)}{\|w\|}$$

注:$y f(x)$ 相当于 $|f(x)|$。

假设超平面 $(\boldsymbol{w}, b)$ 能将训练样本正确分类,即对于 $(x_i, y_i) \in D$,若 $y_i = +1$,则有 $\boldsymbol{w}^T x_i + b > 0$;若 $y_i = -1$,则有 $\boldsymbol{w}^T x_i + b < 0$。令

$$\begin{cases} \boldsymbol{w}^T x_i + b \geq +1, & y_i = +1; \\ \boldsymbol{w}^T x_i + b \leq -1, & y_i = -1. \end{cases} \tag{8.1}$$

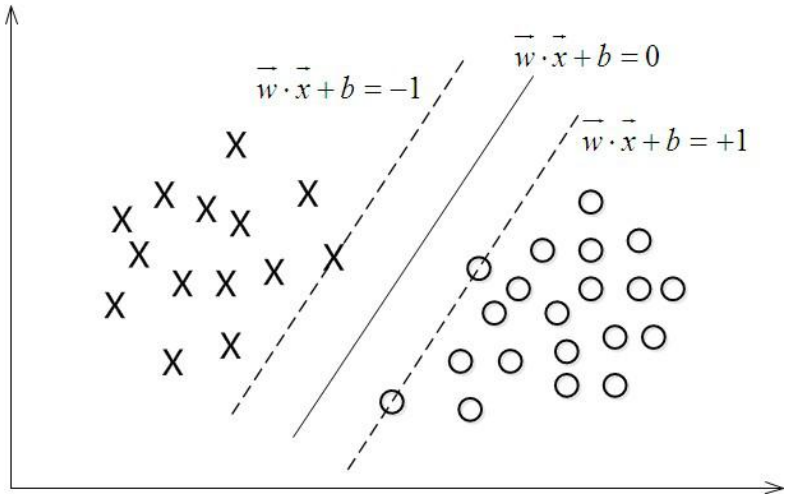


图 8.1

距离超平面最近的这几个训练样本点使式 8.1 的等号成立，它们被称为“**支持向量**(support vector)”，两个异类支持向量到超平面的距离之和为

$$\gamma = \frac{2}{\|w\|}, \tag{8.2}$$

它被称为“**间隔**”(margin)。

欲找到具有“最大间隔”的划分超平面，也就是要找到能满足式 8.1 中约束的参数 $\boldsymbol{w}$ 和 b，使得 γ 最大，即

$$\begin{aligned} min \quad & \frac{1}{2}\|w\|^2, \\ s.t. \quad & y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, m. \end{aligned} \tag{8.3}$$

这就是支持向量机的基本型。

8.2 对偶问题

注意到式 8.3 本身是一个凸二次规划问题，能直接用现成的优化计算包求解，但我们可以有更高效的办法。由于这个问题的特殊结构，还可以通过拉格朗日对偶性变换到对偶变量的优化问题，即通过求解与原问题等价的对偶问题得到原始问题的最优解，这就是线性可分条件下支持向量机的对偶算法，这样做的优点在于：

1. 对偶问题往往更容易求解；
2. 可以自然的引入核函数，进而推广到非线性分类问题。

该问题的拉格朗日函数可写为

$$L(w, b, a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} a_i(y_i(w^T x_i + b) - 1) \tag{8.4}$$

然后令

$$\theta(\boldsymbol{w}) = \max_{a_i \geq 0} L(\boldsymbol{w}, b, a) \tag{8.5}$$

具体写出来，目标函数变成了

$$\min_{\boldsymbol{w},b}\ \theta(\boldsymbol{w}) = \min_{\boldsymbol{w},b}\max_{a_i\geq 0}\ L(\boldsymbol{w},b,a) = p^* \tag{8.6}$$

这里用 p^* 表示这个问题的最优值，且和最初的问题是等价的。如果直接求解，那么一上来便得面对 w 和 b 两个参数，而 a_i 以是不等式约束，这个求解过程不好做。考虑对偶问题

$$\min_{\boldsymbol{w},b}\ \theta(\boldsymbol{w}) = \max_{a_i\geq 0}\ \min_{\boldsymbol{w},b}\ L(\boldsymbol{w},b,a) = d^* \tag{8.7}$$

原始问题通过满足 KKT 条件，已经转化成了对偶问题。而求解这个对偶问题，分为 3 个步骤

1. **让 $L(\boldsymbol{w},b,a)$ 关于 $\boldsymbol{w}$ 和 b 最小化**

   首先固定 a，要让 L 关于 w 和 b 最小化，分别对 w 和 b 求偏导，令其等于 0；

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{w}} &= \|\boldsymbol{w}\| - \sum_{i=1}^{n} a_i y_i x_i \boldsymbol{w}^T = 0 \Rightarrow \boldsymbol{w} = \sum_{i=1}^{n} a_i y_i x_i \\ \frac{\partial L}{\partial b} &= \sum_{i=1}^{n} a_i y_i = 0 \Rightarrow \sum_{i=1}^{n} a_i y_i = 0\end{aligned} \tag{8.8}$$

   将以上结果代入之前的 L，得到

$$\begin{aligned}L(\boldsymbol{w},b,a) &= \frac{1}{2}\sum_{i,j=1}^{n} a_i a_j y_i y_j x_i^T x_j - \sum_{i,j=1}^{n} a_i a_j y_i y_j x_i^T x_j - b\sum_{i=1}^{n} a_i y_i + \sum_{i=1}^{n} a_i \\ &= \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i,j=1}^{n} a_i a_j y_i y_j x_i^T x_j\end{aligned} \tag{8.9}$$

2. **求对 a 的极大**

   求对 a 的极大，即是关于对偶问题的最优化问题。经过上一个步骤的求解，得到的拉格朗日函数式子已经没有了变量 w 和 b，只有 a。从上面的式子得到

$$\begin{aligned}\max_{a}\quad & \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i,j=1}^{n} a_i a_j y_i y_j x_i^T x_j \\ s.t.\quad & a_i \geq 0, i = 1,\ldots,n \\ & \sum_{i=1}^{n} a_i y_i = 0\end{aligned} \tag{8.10}$$

   这样，求出了 a_i，从而根据

$$\begin{aligned}\boldsymbol{w}^* &= \sum_{i=1}^{n} a_i y_i x_i \\ b^* &= -\frac{max\ \boldsymbol{w}^{*T} x_i + min\ \boldsymbol{w}^{*T} x_i}{2}\end{aligned} \tag{8.11}$$

即可求出 w, b，最终得出分离超平面和分类决策函数。

3. **利用 SMO 算法求解对偶问题中的拉格朗日乘子**

在求得 $L(w, b, a)$ 关于 w 和 b 最小化和对 a 的极大之后，最后一步便是利用 SMO 算法求解对偶问题中的拉格朗日乘子

8.3 序列最小最优算法

接上一小节，讨论支持向量机学习的实现问题。讲述其中的序列最小最优化 (sequential minimal optimization,SMO) 算法。

SMO 算法是一种启发式算法，其基本思路是：**如果所有变量的解都满足此最优化问题的 KKT 条件，那么这个最优化问题的解就得到了**。因为 KKT 条件是该最优化问题的充分必要条件。否则，选择两个变量，固定其他变量，针对这两个变量构建一个二次规划问题。这个二次规划问题关于这两个变量的解应该更接近原始二次规划问题的解，因为这会使得原始二次规划问题的目标函数值变得更小。重要的是，这时子问题可以通过解析方法求解，这样就可以大大提高整个算法的计算计算速度。子问题有两个变量，一个是违反 KKT 条件最严重的那一个，另一个由约束条件自动确定。如此，SMO 算法将原问题不断分解为子问题并对子问题求解，进而达到求解原问题的目的。

SMO 算法要解如下凸二次规划的对偶问题：

$$\min_{a} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{N} a_i \tag{8.12}$$

$$s.t. \quad C \geq a_i \geq 0, \quad i = 1, \ldots, n \tag{8.13}$$

$$\sum_{i=1}^{N} a_i y_i = 0 \tag{8.14}$$

注意，子问题的两个变量中只有一个是自由变量。假设 a_1, a_2 为两个变量，$a_3, a_4, \ldots, a_N$ 固定，那么由等式约束 8.14 可知

$$a_1 = -y_1 \sum_{i=2}^{N} a_i y_i \tag{8.15}$$

如果 a_2 确定，那么 a_1 也随之确定。所以子问题中同时更新两个变量。

整个 SMO 算法包括两个部分：**求解两个变量二次规划的解析方法和选择变量的启发式方法**。

于是 SMO 的最优化问题 8.12 的子问题可以写成

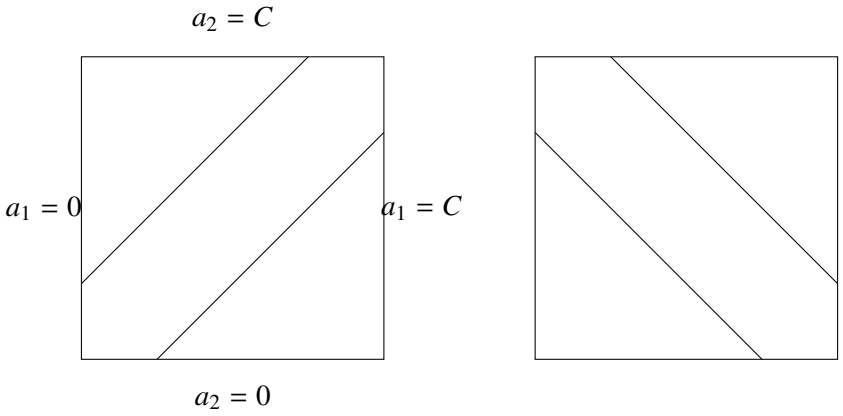
$$\min_{a_1,a_2} W(a_1,a_2)=\frac{1}{2}K_{11}a_1^2+\frac{1}{2}K_{22}a_2^2+y_1y_2K_{12}a_1a_2 -(a_1+a_2)+y_1a_1\sum_{i=3}^{N}y_ia_iK_{i1}+y_2a_2\sum_{i=3}^{N}y_ia_iK_{i2} \tag{8.16}$$

$$s.t\ a_1y_1+a_2y_2=-\sum_{i=3}^{N}y_ia_i=\varsigma \tag{8.17}$$

$$0\le a_i\le C,\ i=1,2 \tag{8.18}$$

其中，$K_{ij}=K(x_i,x_j),i,j=1,2,\ldots,N,\varsigma$ 是常数。

为了求解两个变量的二次规划问题 8.16，首先分析约束条件，然后在此约束条件下求极小。由于只有两个变量 (a_1,a_2)，约束可以用二维空间中的图形表示，如图所示



不等式约束使得 (a_1,a_2) 在盒子 $[0,C]\times[0,C]$ 内，等式约束使 (a_1,a_2) 在平行盒子的对角线的直线上。因此要求的是目标函数在一条平行于对角线线的线段上的最优值。这使得两个变量的最优化问题成为实质上的单变量的最优化问题，不妨考虑为变量 a_2 的最优化问题。

假设问题 8.16 的初始可行解为 a_1^{old},a_2^{old}，最优解为 a_1^{new},a_2^{new}，并且假设在沿着约束方向未经剪辑时 a_2 的最优解为 $a_2^{new,unc}$。

引进记号

$$g(x)=\sum_{i=1}^{N}a_iy_iK(x_i,x)+b \tag{8.19}$$

$$v_i=\sum_{j=3}^{N}y_ia_iK(x_i,x_j)=g(x_i)-\sum_{j=1}^{2}a_jy_jK(x_i,x_j)-b,\ i=1,2 \tag{8.20}$$

目标函数可写成

$$W(a_1,a_2)=\frac{1}{2}K_{11}a_1^2+\frac{1}{2}K_{22}a_2^2+y_1y_2K_{12}a_1a_2-(a_1+a_2)+y_1a_1v_1+y_2a_2v_2 \tag{8.21}$$

令

$$E_i=g(x_i)-y_i=\left(\sum_{j=1}^{N}a_jy_jK(x_j,x_i)+b\right)-y_i,\quad i=1,2 \tag{8.22}$$

当 $i=1,2$ 时，E_i 为函数 $g(x)$ 对输入 x_i 的预测值与真实输出 y_i 之差。由 $a_1y_1=\varsigma-a_2y_2$ 及 $y_i^2=1$，可将 a_1 表示为

$$a_1=(\varsigma-y_2a_2)y_1 \tag{8.23}$$

代入式 8.21，得到只是 a_2 的函数的目标函数，对 a_2 求导数

$$\frac{\partial W}{\partial a_2}=K_{11}a_2+K_{22}a_2-2K_{12}a_2-K_{11}\varsigma y_2+K_{12}\varsigma y_2+y_1y_2-1-v_1v_2+y_2v_2 \tag{8.24}$$

令其为 0，得到

$$\begin{aligned}(K_{11}+K_{22}-2K_{12})a_2=&y_2(y_2-y_1+\varsigma K_{11}-\varsigma K_{12}+v_1-v2)\\ =&y2\left[y_2-y_1+\varsigma K_{11}-\varsigma K_{12}+\left(g(x_1)-\sum_{j=1}^{2}y_ja_jK_{1j}-b\right)\right.\\ &\left.\left(g(x_2)-\sum_{j=1}^{2}y_ja_jK_{2j}-b\right)\right]\end{aligned} \tag{8.25}$$

将 $\varsigma=a_1^{old}y_1+a_2^{old}y_2$ 代入，得到

$$\begin{aligned}(K_{11}+K_{22}-2K_{12})a_2^{new,unc}&=y_2((K_{11}+K_{22}-2K_{12})a_2^{old}y_2+y_2-y_1+g(x_1)-g(x_2))\\ &=(K_{11}+K_{22}-2K_{12})a_2^{old}+y_2(E_1-E_2)\end{aligned} \tag{8.26}$$

将 $\eta=K_{11}+K_{22}-2K_{12}$ 代入，于是得到

$$a_2^{new,unc}=a_2^{old}+\frac{y_2(E_1-E_2)}{\eta} \tag{8.27}$$

要使其满足不等式的约束必须将其限制在区间 $[L,H]$ 内，从而得到 a_2^{new} 的表达式

$$a_2^{new}=\begin{cases}H, & a_2^{new,unc}>H\\ a_2^{new,unc}, & L\le a_2^{new,unc}\le H\\ L, & a_2^{new,unc}<L\end{cases} \tag{8.28}$$

如果 $y_1\neq y_2$

$$L=max(0,a_2^{old}-a_1^{old}),\quad H=min(C,C+a_2^{old}-a_1^{old})$$

如果 $y_1 = y_2$

$$L = max(0, a_2^{old} + a_1^{old} - C), \quad H = min(C, a_2^{old} + a_1^{old})$$

变量的选择方法

SMO 算法在每个子问题中选择两个变量优化，其中至少一个变量是违反 KKT 条件的。

1. 第 1 个变量的选择

   SMO 称选择第 1 个变量的过程为外层循环。外层循环在训练样本中选取违反 KKT 条件最严重的样本点，并将其对应的变量作为第 1 个变量。具体地，检验训练样本点 (x_i, y_i) 是否满足 KKT 条件，即

$$a_i = 0 \Leftrightarrow y_i g(x_i) \geq 1 \tag{8.29}$$

$$C > a_i > 0 \Leftrightarrow y_i g(x_i) = 1 \tag{8.30}$$

$$a_i = C \Leftrightarrow y_i g(x_i) \leq 1 \tag{8.31}$$

2. 第 2 个变量的选择

   SMO 称选择第 2 个变量的过程为内层循环，假设在外层循环中已经找到第 1 个变量 a_1，现在要在内层循环中找到第 2 个变量 a_2。第 2 个变量选择的标准是希望能使 a_2 有足够大的变化。由式 8.27 知，a_2^{new} 是依赖于 $|E_1 - E_2|$ 的，加了加快计算速度，一种简单的做法是选择 a_2，使其对应的 $|E_1 - E_2|$ 最大。

3. 计算阈值 b 和差值 E_i

8.4 核函数

SVM 处理线性可分的情况，而对于非线性的情况，SVM 的处理方法是选择一个核函数 $K < \cdot, \cdot >$，通过将数据映射到高维空间，来解决在原始空间中线性不可分的问题。

此外，用对偶形式表示学习器的优势在于该表示中可调参数的个数不依赖输入属性的个数，通过使用恰当的核函数来替代内积，可以隐式得将非线性的训练数据映射到高维空间，而不增加可调参数的个数。

在线性不可分的情况下，支持向量机首先在低维空间中完成计算，然后通过核函数将输入空间映射到高维牲空间，最终在高维特征空间中构造出最优分离超平面，从而把平面上本身不好分的非线性数据分开。

而在我们遇到核函数之前，如果用原始的方法，那么在用线性学习器学习一个非线性关系，需要选择一个非线性特征集，并且将数据写成新的表达形式，这等价于应用一个固定的非线性映射，将数据映射到特征空间，在特征空间中使用线性学习器，因此考虑的假设集是这种类型的函数：

$$f(x) = \sum_{i=1}^{N} w_i \phi_i(x) + b \tag{8.32}$$

这里 $\phi : X \rightarrow F$ 是从输入空间到某个特征空间的映射，这意味着建立非线性学习器分为

两步：

1. 首先使用一个非线性映射将数据变换到一个特征空间 F
2. 然后在特征空间使用线性学习器分类

而由于对偶形式就是线性学习器的一个重要性质，这意味着假设可以表达为训练点的线性组合，因此决策规则可以用测试点和训练点的内积来表示：

$$f(x) = \sum_{i=1}^{l} a_i y_i < \phi(x_i), \phi(x) > +b \tag{8.33}$$

如果有一种方式可以在特征空间中直接计算内积 $< \phi(x_i), \phi(x) >$，就像在原始输入点的函数中一样，就有可能将两个步骤融合到一起建立一个非线性的学习器，这样直接计算法的方法称为**核函数方法**

定义 8.1. Kernel

核是一个函数 K，对所有 $x, z \in X$，满足 $K(x, z) =< \phi(x), \phi(z) >$，这里 ϕ 是从 X 到内积特征空间 F 的映射。 ♣

下面举一个核函数把低维空间映射到高维空间的例子：

我们考虑核函数 $K(v_1, v_2) =< v_i, v_2 >^2$，即“内积平方”，这里 $v_1 = (x_1, y_1), v_2 = (x_2, y_2)$ 是二维空间中的两个点。这个核函数对应着一个二维空间到三维空间的映射，它的表达式是：

$$P(x, y) = (x^2, \sqrt{2}xy, y^2)$$

可以验证，

$$\begin{aligned} < P(v_1), p(v_2) > &=< (x_1^2, \sqrt{2}x_1y_1, y_1^2), (x_2^2, \sqrt{2}x_2y_2, y_2^2) > \\ &= x_1^2x_2^2 + 2x_1x_2y_1y_2 + y_1^2y_2^2 \\ &= (x_1x_2 + y_1y_2)^2 \\ &=< v_1, v_2 >^2 \\ &= K(v_1, v_2) \end{aligned}$$

上面的例子所说，核函数的作用就是隐含着一个从低维空间到高维空间的映射，而这个映射可以把低维空间中线性不可分的两类点变成线性可分的。

核函数的本质

1. 实际中，我们会经常遇到线性不可分的样例，此时，我们的常用做法是把特征映射到高维空间中去
2. 但进一步，如果凡是遇到线性不可分的样例，一律映射到高维空间，那么这个维度大小会高到可怕的。那咋办呢？
3. 此时，核函数就隆重登场了，核函数的价值在于它虽然也是讲特征进行从低维到高维的转换，但核函数绝就绝在它事先在低维上进行计算，而将实质上的分类效果表现在了高维上，也就避免了直接在高维空间中的复杂计算。

几个核函数：

- 多项式核 $K(x_1, x_2) = (< x_1, x_2 > +R)^d$
- 高斯核 $K(x_1, x_2) = exp(-\|x_1 - x_2\|^2/2\sigma^2)$
- 线性核 $K(x_1, x_2) =< x_1, x_2 >$

已知映射函数 ϕ，可以通过 $\phi(x)$ 和 $\phi(z)$ 的内积求得核函数 $K(x, z)$。不用构造映射 $\phi(x)$ 能否直接判断一个给定的函数 $K(x, z)$ 是不是核函数？或者说，函数 $K(x, z)$ 满足什么条件才能成为核函数？

已知，假设 $K(x, z)$ 是定义在 $\chi \times \chi$ 上的对称函数，并且对任意的 $x_1, \ldots, x_m \in \chi$，$K(x, z)$ 关于 $x_1, \ldots, x_m$ 的 Gram 矩阵是半正定的。可以依据函数 $K(x, z)$ 构成一个希尔伯特空间 (Hilbert space)，其步骤是：首先定义映射 ϕ 并构成向量空间 S；然后在 S 上定义内积构成内积空间；最后将 S 完备化构成希尔伯特空间。

定理 8.1. 正定核的充要条件

设 $K : \chi \times \chi \to \boldsymbol{R}$ 是对称函数，则 $K(x, z)$ 为正定核的充要条件是对任意 $x_i \in \chi, i = 1, 2, \ldots, m$，$K(x, z)$ 对应的 Gram 矩阵：

$$K = [K(x_i, x_j)]_{m \times m} \tag{8.34}$$

是半正定矩阵。 ♡

证明：

1. 必要性。由于 $K(x, z)$ 是 $\chi \times \chi$ 上的正定核，所以存在从 χ 到希尔伯特空间 $\mathcal{H}$ 的映射 ϕ，使得

$$K(x, z) = \phi(x) \cdot \phi(z) \tag{8.35}$$

于是，对任意 $x_1, \ldots, x_m$，构造 $K(x, z)$ 关于 $x_1, \ldots, x_m$ 的 Gram 矩阵

$$[K_{ij}]_{m \times m} = [K(x_i, x_j)]_{m \times m} \tag{8.36}$$

对任意 $c_1, \ldots, c_m \in \boldsymbol{R}$，有

$$\begin{aligned} \sum_{i,j=1}^{m} c_i c_j K(x_i, x_j) &= \sum_{i,j=1}^{m} c_i c_j (\phi(x_i) \cdot \phi(x_j)) \\ &= \left(\sum_{i,j=1}^{m} c_i \phi(x_i) \right) \cdot \left(\sum_{i,j=1}^{m} c_j \phi(x_j) \right) \\ &= \left\| \sum_{i} c_i \phi(x_i) \right\|^2 \geqslant 0 \end{aligned} \tag{8.37}$$

表明 $K(x, z)$ 关于 $x_1, \ldots, x_m$ 的 Gram 矩阵是半正定的。

2. 充分性根据已知，可以构造从 χ 到某个希尔伯特空间 $\mathcal{H}$ 的映射，从而表明 $K(x, z)$ 是 $\chi \times \chi$ 上的核函数。

8.5 软间隔与正则化

第 9 章 核方法

有这样一类模式识别的技术：训练数据点或者它的一个子集在预测阶段仍然保留并且被使用。许多线性参数模型可以被转化为一个等价的“对偶表示”。对偶表示中，预测的基础也是在训练数据点处计算的核函数 (kernel function) 的线性组合。对于基于固定非线性特征空间 (feature space) 映射 $\phi(\boldsymbol{x})$ 的模型来说，核函数由下面的关系给出。

$$k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}') \tag{9.1}$$

核的概念由 Aizenman 引入模型识别领域。那篇文章介绍了势函数的方法。之所以被称为势函数，是因为它类似于静电学中的概念。虽然被忽视了很多年，但是 Boser 在边缘分类器的问题中把它重新引入到了机器学习领域。那篇文章提出了支持向量机的方法。从那里起，这个话题在理论上和实用上都吸引了大家的兴趣。一个最重要的发展是把核方法进行了扩展，使其能处理符号化的物体，从而极大地扩展了这种方法能处理的问题的范围。

9.1 对偶表示

许多回归的线性模型和分类的线性模型的公式都可以使用对偶表示重写。使用对偶表示形式，核函数可以自然地产生。这里，我们考虑一个线性模型，它的参数通过最小化正则化的平方和误差函数来确定。

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - t_n\}^2 + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w} \tag{9.2}$$

其中 $\lambda \geqslant 0$。如果我们令 $J(\boldsymbol{w})$ 关于 $\boldsymbol{w}$ 的梯度等于零，那么我们看到 $\boldsymbol{w}$ 的解是向量 $\phi(\boldsymbol{w})$ 的线性组合的形式，系数是 $\boldsymbol{w}$ 的函数，形式为

$$\begin{aligned}\frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} &= \sum_{n=1}^{N}\{\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - t_n\}\phi(\boldsymbol{x}_n) + \lambda\boldsymbol{w} \\ \Rightarrow \boldsymbol{w} &= -\frac{1}{\lambda}\sum_{n=1}^{N}\{\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - t_n\}\phi(\boldsymbol{x}_n) \\ &= \sum_{n=1}^{N} a_n\phi(\boldsymbol{x}_n) = \Phi^T\boldsymbol{a}\end{aligned} \tag{9.3}$$

其中 Φ 是设计矩阵，第 n 行为 $\phi(\boldsymbol{x}_n)^T$。向量 $\boldsymbol{a} = (a_1, \ldots, a_N)^T$

$$a_n = -\frac{1}{\lambda}\{\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - t_n\} \tag{9.4}$$

我们现在不直接对参数向量 $\boldsymbol{w}$ 进行操作，而是使用参数向量 $\boldsymbol{a}$ 重新整理最小平方算法，得到一个对偶表示。如果我们将 $\boldsymbol{w} = \Phi^T \boldsymbol{a}$ 代入 $J(\boldsymbol{w})$，那么可以得到

$$\begin{aligned} J(\boldsymbol{w}) &= \frac{1}{2}(\boldsymbol{w}^T\Phi^T - \boldsymbol{t}^T)(\Phi\boldsymbol{w} - \boldsymbol{t}) + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w} \\ &= \frac{1}{2}\left[\boldsymbol{w}^T\Phi^T\Phi\boldsymbol{w} - \boldsymbol{w}^T\Phi^T\boldsymbol{t} - \boldsymbol{t}^T\Phi\boldsymbol{w} + \boldsymbol{t}^T\boldsymbol{t}\right] + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w} \\ &\Rightarrow \text{把 } \boldsymbol{a} \text{ 看作参数，代入 } \boldsymbol{w}\Phi^T\boldsymbol{a} \\ J(\boldsymbol{a}) &= \frac{1}{2}\left[\boldsymbol{a}^T\Phi\Phi^T\Phi\Phi^T\boldsymbol{a} - \boldsymbol{a}\Phi\Phi^T\boldsymbol{t} - \boldsymbol{t}^T\Phi\Phi^T\boldsymbol{a} + \boldsymbol{t}^T\boldsymbol{t}\right] + \frac{\lambda}{2}\boldsymbol{a}^T\Phi\Phi^T\boldsymbol{a} \\ &= \frac{1}{2}\boldsymbol{a}^T\Phi\Phi^T\Phi\Phi^T\boldsymbol{a} - \boldsymbol{a}\Phi\Phi^T\boldsymbol{t} + \frac{1}{2}\boldsymbol{t}^T\boldsymbol{t} + \frac{\lambda}{2}\boldsymbol{a}^T\Phi\Phi^T\boldsymbol{a} \end{aligned} \tag{9.5}$$

其中 $\boldsymbol{t} = (t_1, \ldots, t_N)^T$。定义 Gram 矩阵 $K = \Phi\Phi^T$，它是一个 $N \times N$ 的对称矩阵，元素为

$$K_{nm} = \phi(\boldsymbol{x}_n)^T\Phi(\boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m) \tag{9.6}$$

其中我们引入了公式 9.1 定义的核函数。使用 Gram 矩阵，平方和误差函数可以写成

$$J(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^T\boldsymbol{K}\boldsymbol{K}\boldsymbol{a} - \boldsymbol{a}^T\boldsymbol{K}\boldsymbol{t} + \frac{1}{2}\boldsymbol{t}^T\boldsymbol{t} + \frac{\lambda}{2}\boldsymbol{a}^T\boldsymbol{K}\boldsymbol{a} \tag{9.7}$$

由公式 9.4，求解 $\boldsymbol{a}$，我们有

$$\begin{aligned} \boldsymbol{a}^T &= -\frac{1}{\lambda}(\boldsymbol{w}^T\Phi^T - \boldsymbol{t}) \\ &\Rightarrow \lambda\boldsymbol{a}^T = \boldsymbol{t}^T - \boldsymbol{a}^T\Phi\Phi^T = \boldsymbol{t} - \boldsymbol{a}\boldsymbol{K} \\ &\Rightarrow \boldsymbol{a} = (K + \lambda\boldsymbol{I}_N)^{-1}\boldsymbol{t} \end{aligned} \tag{9.8}$$

如果我们将这个代入线性回归模型中，对于新的输入 $\boldsymbol{x}$，我们得到了下面预测

$$y(\boldsymbol{x}) = \boldsymbol{w}^T\phi(\boldsymbol{x}) = \boldsymbol{a}^T\Phi\phi(\boldsymbol{x}) = k(\boldsymbol{x})^T(\boldsymbol{K} + \lambda\boldsymbol{I}_N)^{-1}\boldsymbol{t} \tag{9.9}$$

其中我们定义了向量 $k(\boldsymbol{x})$，它的元素为 $k_n(\boldsymbol{x}) = k(\boldsymbol{x}_n, \boldsymbol{x})$。因此我们看到对偶公式使得最小平方问题的解完全通过核函数 $k(\boldsymbol{x}, \boldsymbol{x}')$ 表示。这被称为对偶公式，因为 $\boldsymbol{a}$ 的解可以被表示为 $\phi(\boldsymbol{x})$ 的线性组合，从而我们可以使用参数向量 $\boldsymbol{w}$ 恢复出原始的公式。

对偶公式的优点是，它可以完全通过核函数 $k(\boldsymbol{x}, \boldsymbol{x}')$ 来表示。于是，我们可以直接针对核函数进行计算，避免了显式地引入特征向量 $\phi(\boldsymbol{x})$，这使得我们可以隐式地使用高维特征空间，甚至无限维特征空间。

基于 Gram 矩阵的对偶表示的存在是许多线性模型的性质，包括感知器。后面，我们会研究回归的概率线性模型和高斯过程方法的对偶性。

9.2 构造核

为了利用核技巧,我们需要能够构造合法的核函数。一种方法是选择一个特征空间映射 $\phi(\boldsymbol{x})$,然后利用这个映射寻找对应的核。一维空间的核函数被定义为

$$k(x, x') = \phi(x)^T \phi(x) = \sum_{i=1}^{M} \phi_i(x)\phi_i(x') \tag{9.10}$$

其中 $\phi_i(x)$ 是基函数。

另一种方法是直接构造核函数。在这种情况下,我们必须确保我们核函数是合法的,即它对应于某个 (或能是无穷维) 特征空间的标量积。作为一个简单的例子,考虑下面的核函数

$$k(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^T \boldsymbol{z})^2 \tag{9.11}$$

如果我们取二维输入空间 $\boldsymbol{x} = (x_1, x_2)$ 的特殊情况,那么我们可以展开这一项,于是得到对应的非线性特征映射

$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{z}) &= (\boldsymbol{x}^T \boldsymbol{z})^2 = (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= (x_1^2, \sqrt{2}x_1 x_2, x_2^2)(z_1^2, \sqrt{2}z_1 z_2, z_2^2)^T \\ &= \phi(\boldsymbol{x})^T \phi(\boldsymbol{z}) \end{aligned} \tag{9.12}$$

我们看到特征映射的形式为 $\phi(\boldsymbol{x}) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)$,因此这个特征映射由所有的二阶项组成,每个二阶项有一个具体的系数。

但是,更一般地,我们需要找到一种更简单的方法检验一个函数是否是一个合法的核函数,而不需要显示地构造函数 $\phi(\boldsymbol{x})$。核函数是一个合法的核函数的充分必要条件是 Gram 矩阵在所有的集合在所有的集合 $\{\boldsymbol{x}_n\}$ 的选择下都是半正定的。

构造新的核函数的一个强大的方法是使用简单的核函数作为基本的模块来构造。可以使用下面的性质来完成这件事。给定合法的核 $k_1(\boldsymbol{x}, \boldsymbol{x}')$ 和 $k_2(\boldsymbol{x}, \boldsymbol{x}')$,下面的新核也是合

法的

$$k(\boldsymbol{x},\boldsymbol{x}') = ck_1(\boldsymbol{x},\boldsymbol{x}') \tag{9.13}$$

$$k(\boldsymbol{x},\boldsymbol{x}') = f(\boldsymbol{x})k_1(\boldsymbol{x},\boldsymbol{x}')f(\boldsymbol{x}') \tag{9.14}$$

$$k(\boldsymbol{x},\boldsymbol{x}') = q(k_1(\boldsymbol{x},\boldsymbol{x}')) \tag{9.15}$$

$$k(\boldsymbol{x},\boldsymbol{x}') = \exp(k_1(\boldsymbol{x},\boldsymbol{x}')) \tag{9.16}$$

$$k(\boldsymbol{x},\boldsymbol{x}') = k_1(\boldsymbol{x},\boldsymbol{x}') + k_2(\boldsymbol{x},\boldsymbol{x}') \tag{9.17}$$

$$k(\boldsymbol{x},\boldsymbol{x}') = k_1(\boldsymbol{x},\boldsymbol{x}')k_2(\boldsymbol{x},\boldsymbol{x}') \tag{9.18}$$

$$k(\boldsymbol{x},\boldsymbol{x}') = k_3(\phi(\boldsymbol{x},\phi(\boldsymbol{x}'))) \tag{9.19}$$

$$k(\boldsymbol{x},\boldsymbol{x}') = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}' \tag{9.20}$$

$$k(\boldsymbol{x},\boldsymbol{x}') = k_a(\boldsymbol{x}_a,\boldsymbol{x}_a') + k_b(\boldsymbol{x}_b,\boldsymbol{x}_b') \tag{9.21}$$

$$k(\boldsymbol{x},\boldsymbol{x}') = k_a(\boldsymbol{x}_a,\boldsymbol{x}_a')k_b(\boldsymbol{x}_b,\boldsymbol{x}_b') \tag{9.22}$$

其中 $c > 0$ 是一个常数，$f(\cdot)$ 是任意函数，$q(\cdot)$ 是一个系数非负的多项式，$\phi(\boldsymbol{x})$ 是一个从 $\boldsymbol{x}$ 到 $\mathbb{R}^M$ 的函数，$k_3(\cdot,\cdot)$ 是 $\mathbb{R}^M$ 中的一个合法的核，$\boldsymbol{A}$ 是一个对称半正定矩阵，$\boldsymbol{x}_a$ 和 $\boldsymbol{x}_b$ 是变量，且 $\boldsymbol{x} = (\boldsymbol{x}_a, \boldsymbol{x}_b)$。$K_a, K_b$ 是各自空间的合法的核函数。

核观点的一个重要的贡献是可以扩展到符号化的输入，而不是简单的实数向量。构造核的另一个强大的方法是从一个概率生成式模型开始构造，这使得我们可以在一个判别式的框架中使用生成式模型。生成式模型可以自然地处理缺失数据，并且在隐马尔可夫模型的情况下，可以处理长度变化的序列。相反，判别式模型在判别式的任务中通常会比生成式模型的表现更好。于是，将这两种方法结合吸收了一些人的兴趣。一种将二者结合的方法是使用一个生成式模型定义一个核，然后在判别式方法中使用这个核。

给定一个生成式模型 $p(\boldsymbol{x})$，我们可以定义一个核

$$k(\boldsymbol{x},\boldsymbol{x}') = p(\boldsymbol{x})p(\boldsymbol{x}') \tag{9.23}$$

很明显，这是一个合法的核，因为我们可以把它看成由映射 $p(\boldsymbol{x})$ 定义的一维特征空间中的一个内积。它表明，如果两个输入 $\boldsymbol{x}$ 和 $\boldsymbol{x}'$ 都具有较高的概率，那么它们就是相似的。扩展这类核的方法是考虑不同概率分布的乘积的加和，带有正的权值系数 $p(i)$，形式为

$$k(\boldsymbol{x},\boldsymbol{x}') = \sum_i p(\boldsymbol{x}|i)p(\boldsymbol{x}'|i)p(i) \tag{9.24}$$

如果不考虑一个整体的乘法常数，这个核就等价于一个混合概率密度，它可以分解成各个分量概率密度，下标 i 扮演着“潜在”变量的角色。如果两个输入 $\boldsymbol{x}$ 和 $\boldsymbol{x}'$ 在一大类的不同分量下都有较大的概率，那么这两个输入将会使核函数输出较大的值，因此就表现出相似

性。在无限求和的极限情况下，我们也可以考虑下面形式的核函数

$$k(\boldsymbol{x}, \boldsymbol{x}') = \int p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{x}'|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z} \tag{9.25}$$

其中 $\boldsymbol{z}$ 是一个连续潜在变量。

另一个使用生成式模型定义核函数的方法被称为 Fisher 核。

9.3 径向基函数网络

讨论了基于固定基函数的线性组合的回归模型，但是没有详细讨论可以取哪种形式的基函数。一种广泛使用的基函数是径向基函数 (radial basis functions)。径向基函数中，每一个基函数只依赖于样本和中心 $\boldsymbol{\mu}_j$ 之间的径向距离 (通常是欧几里德距离)，即 $\phi_j(\boldsymbol{x}) = h(\|\boldsymbol{x} - \boldsymbol{\mu}_j\|)$。历史上，径向基函数被用来进行精确的函数内插。但是，在模式识别应用中，目标值通常带有噪声，精确内插不是我们想要的，因为这对应于一个过拟合的解。

对径向基函数的展开来自正则化理论。径向基函数的另一个研究动机来源于输入变量 (而不是目标变量) 具有噪声时的内插问题。如果输入变量 $\boldsymbol{x}$ 上的噪声由一个服从分布 $v(\boldsymbol{\xi})$ 的变量 $\boldsymbol{\xi}$，那么平方和误差函数就变成了

$$E = \frac{1}{2}\sum_{n=1}^{N}\int \{y(\boldsymbol{x}_n + \boldsymbol{\xi}) - t_n\}^2 v(\boldsymbol{\xi})d\boldsymbol{\xi} \tag{9.26}$$

使用变分法，我们可以关于函数 $y(\boldsymbol{x})$ 进行最优化，得到

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} t_n h(\boldsymbol{x} - \boldsymbol{x}_n) \tag{9.27}$$

其中基函数为

$$h(\boldsymbol{x} - \boldsymbol{x}_n) = \frac{v(\boldsymbol{x} - \boldsymbol{x}_n)}{\sum_{n=1}^{N} v(\boldsymbol{x} - \boldsymbol{x}_n)} \tag{9.28}$$

我们看到这是一个以每个数据点为中心的基函数。这被称为 Nadaraya-Watson 模型。

另一个展开归一化径向基函数的情况是把核密度估计应用到回归问题。

Nadaraya-Watson 模型

对于新的输入 $\boldsymbol{x}$，线性回归模型的预测的形式为训练数据集的目标值的线性组合，组合系数由“等价核”给出，其中等价核满足加和限制。我们可以从核密度估计开始，以一个不同的角度研究该回归模型。假设我们有一个训练集 $\{\boldsymbol{x}_n, t_n\}$，我们使用 Parzen 密度估计来对联合分布 $p(\boldsymbol{x}, t)$ 进行建模，即

$$p(\boldsymbol{x}, t) = \frac{1}{N}\sum_{n=1}^{N} f(\boldsymbol{x} - \boldsymbol{x}_n, t - t_n) \tag{9.29}$$

其中 $f(\boldsymbol{x},t)$ 是分量密度函数，每个数据点都有一个以数据点为中心的这种分量。

我们现在要找到回归函数 $y(\boldsymbol{x})$ 的表达式，对应于以输入变量为条件的目标变量的条件均值，它的表达式为

$$\begin{aligned} y(\boldsymbol{x}) &= \mathbb{E}[t|\boldsymbol{x}] = \int_{-\infty}^{\infty} tp(t|\boldsymbol{x})dt \\ &= \int_{-\infty}^{\infty} t\left(\frac{p(\boldsymbol{x},t)}{p(\boldsymbol{x})}\right)dt \\ &= \frac{\int tp(\boldsymbol{x},t)dt}{\int p(\boldsymbol{x},t)dt} \\ &= \frac{\sum_n \int tf(\boldsymbol{x}-\boldsymbol{x}_n, t-t_n)dt}{\sum_m \int f(\boldsymbol{x}-\boldsymbol{x}_m, t-t_m)dt} \end{aligned} \tag{9.30}$$

简单起见，我们现在假设分量的密度函数的均值为零，即

$$\int f(\boldsymbol{x},t)tdt = 0 \tag{9.31}$$

对所有 $\boldsymbol{x}$ 都成立。使用一个简单的变量替换，我们有

$$\begin{aligned} y(\boldsymbol{x}) &= \frac{\sum_n g(\boldsymbol{x}-\boldsymbol{x}_n)t_n}{\sum_m g(\boldsymbol{x}-\boldsymbol{x}_m)} \\ &= \sum_n k(\boldsymbol{x},\boldsymbol{x}_n)t_n \end{aligned} \tag{9.32}$$

其中 $n,m = 1,\ldots,N$，且核函数 $k(\boldsymbol{x},\boldsymbol{x}_n)$ 为

$$\frac{g(\boldsymbol{x}-\boldsymbol{x}_n)}{\sum_m g(\boldsymbol{x}-\boldsymbol{x}_m)} \tag{9.33}$$

并且我们定义了

$$g(\boldsymbol{x}) = \int f(\boldsymbol{x},t)dt \tag{9.34}$$

公式 9.32 给出的结果被称为 Nadaraya-Watson 模型，或者称为核回归 (kernel regression)。对于一个局部核函数，它的性质为：给距离 $\boldsymbol{x}$ 较近的数据点 $\boldsymbol{x}_n$ 较高的权重。

这个模型的一个明显的推广是允许形式更灵活的高斯分布作为其分量，例如让输入和目标值具有不同方差。更一般地，我们可以使用高斯混合模型对联合分布 $p(t,\boldsymbol{x})$ 建模，然后找到对应的条件概率分布 $p(t|\boldsymbol{x})$。

9.4 高斯过程

通过将对偶性的概念应用于回归的非概率模型，我们引出了核的概念。这里，我们把核的角色推广到概率判别式模型中，引出了高斯过程的框架。于是，我们会看到在贝叶斯方法中，核是如何自然地被引入的。

在高斯过程的观点中，我们抛弃参数模型，直接定义函数上的先验概率分布。等价于高斯过程的模型在许多不同领域被广泛研究。例如，在统计地质学中文献中，高斯过程回

归被称为 kriging。类似地，ARMA(自动回归移动平均) 模型、Kalman 滤波以及径向基函数网络都可以被看成高斯过程模型的形式。

重新考虑线性回归问题

为了引出高斯过程的观点，我们回到线性回归的例子中，通过对函数 $y(\boldsymbol{x}\boldsymbol{w})$ 的计算，重新推导出预测分布。

考虑一个模型 M，它被定义为由向量 $\phi(\boldsymbol{x})$ 的元素给出的 M 个固定基函数的线性组合，即

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \phi(\boldsymbol{x}) \tag{9.35}$$

其中 $\boldsymbol{x}$ 是输入向量，$\boldsymbol{w}$ 是 M 维权向量。现在，考虑 $\boldsymbol{w}$ 上的一个先验概率分布，这个分布是一个各向同性的高斯分布，形式为

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|0, \alpha^{-1}\boldsymbol{I}) \tag{9.36}$$

它由一个超参数 α 控制，这个超参数表示分布的精度。对于任意给定的 $\boldsymbol{w}$，公式 9.35 定义了 $\boldsymbol{x}$ 的一个特定的函数。于是 $\boldsymbol{w}$ 上的概率分布就产生了一个函数 $y(\boldsymbol{x})$ 上的一个概率分布。在实际应用中，我们希望计算这个函数在某个具体的 $\boldsymbol{x}$ 处的函数值，例如在训练数据点 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ 处的函数值。于是我们感兴趣的是函数值 $y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_N)$ 的概率分布。我们把函数值的集合记作向量 $\boldsymbol{y}$。根据公式 9.35，这个向量等于

$$\boldsymbol{y} = \Phi \boldsymbol{w} \tag{9.37}$$

其中 Φ 是设计矩阵，元素为 $\Phi_{nk} = \phi_k(\boldsymbol{x}_n)$。我们可以用下面的方式找到 $\boldsymbol{y}$ 的概率分布。首先，我们注意到 $\boldsymbol{y}$ 是由 $\boldsymbol{w}$ 的元素给出的服从高斯分布的变量的线性组合，因此它本身是服从高斯分布。于是，我们只需要找到它的均值和方差。

$$\mathbb{E}[y] = \Phi \mathbb{E}[\boldsymbol{w}] = 0 \tag{9.38}$$

$$cov[\boldsymbol{y}] = \mathbb{E}[\boldsymbol{y}\boldsymbol{y}^T] - 0 = \Phi \mathbb{E}[\boldsymbol{w}\boldsymbol{w}^T]\Phi^T = \frac{1}{\alpha}\Phi\Phi^T = \boldsymbol{K} \tag{9.39}$$

其中，$\boldsymbol{K}$ 是 Gram 矩阵，元素为

$$K_{nm} = k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \frac{1}{\alpha}\phi(\boldsymbol{x}_n)^T \phi(\boldsymbol{x}_m) \tag{9.40}$$

$k(\boldsymbol{x}, \boldsymbol{x}')$ 是核函数。

这个模型给我们提供了高斯过程的一个具体的例子。通常来说，高斯过程被定义为函数 $y(\boldsymbol{x})$ 上的一个概率分布，使得在任意点集 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ 处计算的 $y(\boldsymbol{x})$ 的值的集合联合起来服从高斯分布。

高斯随机过程的一个关键点是 N 个变量 $y_1, \ldots, y_N$ 上的联合概率分布完全由二阶统计 (即均值和协方差) 确定。在大部分应用中，我们关于 $y(\boldsymbol{x})$ 的均值没有任何先验的知识，

因此根据对称性，我们令其等于零。这等价于基函数的观点中，令权值 $p(\boldsymbol{w}|\alpha)$ 的先验概率分布的均值等于零。之后，高斯过程的确定通过给定两个 $\boldsymbol{x}$ 处的函数值 $y(\boldsymbol{x})$ 的协方差来完成。这个协方差由核函数确定

$$\mathbb{E}[y(\boldsymbol{x}_n)y(\boldsymbol{x}_m)] = k(\boldsymbol{x}_n, \boldsymbol{x}_m) \tag{9.41}$$

我们也可以直接定义核函数，而不是间接地通过选择基函数。

用于回归的高斯过程

为了把高斯过程模型应用一回归模型，我们需要考虑观测目标值的噪声，形式为

$$t_n = y_n + \epsilon_n \tag{9.42}$$

其中 $y_n = y(\boldsymbol{x}_n), \epsilon_n$ 是一个随机噪声变量，它的值对于每个观测 n 是独立的。这里我们要考虑服从高斯分布的噪声过程，即

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1}) \tag{9.43}$$

其中 β^{-1} 是一个超参数，表示噪声的精度。由于噪声对于每个数据点是独立的，因此以 $\boldsymbol{y} = (y_1, \ldots, y_N)^T$ 为条件，目标值 $\boldsymbol{t} = (t_1, \ldots, t_N)^T$ 的联合概率分布是一个各向同性的高斯分布，形式为

$$p(\boldsymbol{t}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{y}, \beta^{-1}\boldsymbol{I}_N) \tag{9.44}$$

根据高斯过程的定义，边缘概率分布 $p(\boldsymbol{y})$ 是一个高斯分布，均值为零，协方差由 Gram 矩阵 $\boldsymbol{K}$ 定义，即

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}|0, \boldsymbol{K}) \tag{9.45}$$

确定 $\boldsymbol{K}$ 的核函数通常被选择成能够表示下面的性质：对于相似的点 $\boldsymbol{x}_n$ 和 $\boldsymbol{x}_m$，对应的值 $y(\boldsymbol{x}_n)$ 和 $y(\boldsymbol{x}_m)$ 的相关性要大于不相似的点。这里，相似性的概念取决于实际应用。

为了找到以输入值 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ 为条件的边缘概率分布 $p(\boldsymbol{t})$，我们需要对 $\boldsymbol{y}$ 积分。可以通过使用公式 2.76，我们看到 $\boldsymbol{t}$ 的边缘概率分布为

$$p(\boldsymbol{t}) = \int p(\boldsymbol{t}|\boldsymbol{y})p(\boldsymbol{y})d\boldsymbol{y} = \mathcal{N}(\boldsymbol{t}|0, \boldsymbol{C}) \tag{9.46}$$

其中协方差矩阵 $\boldsymbol{C}$ 的元素为

$$C(\boldsymbol{x}_n, \boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m) + \beta^{-1}\delta_{nm} \tag{9.47}$$

对于高斯过程回归，一个广泛使用的核函数的形式为指数项的二次型加上常数和线性项，即

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \theta_0 \exp\left\{-\frac{\theta_1}{2}\|\boldsymbol{x}_n - \boldsymbol{x}_m\|^2\right\} + \theta_2 + \theta_3 \boldsymbol{x}_n^T \boldsymbol{x}_m \tag{9.48}$$

目前为止，我们已经使用高斯过程的观点来构建数据点的集合上的联合概率分布的模型。然而，我们在回归问题中的目标是在给定一组训练数据的情况下，对新的输入变量预测目标变量的值。假设 $\boldsymbol{t}_N = (t_1, \ldots, t_N)^T$，对应于输入值 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$，组成观测训练集，并且我们的目标是对于新的输入向量 $\boldsymbol{x}_{N+1}$ 预测目标变量 t_{N+1}。这要求我们计算预测分布 $p(t_{N+1}|\boldsymbol{t}_N)$。

为了找到条件条布 $p(t_{N+1}|\boldsymbol{t})$，我们首先写下联合概率分布 $p(\boldsymbol{t}_{N+1})$。

$$p(\boldsymbol{t}_{N+1}) = \mathcal{N}(\boldsymbol{t}_{N+1}|0, \boldsymbol{C}_{N+1}) \tag{9.49}$$

其中 $\boldsymbol{C}_{N+1}$ 是一个 $(N+1)\times(N+1)$ 的协方差矩阵，我们将协方差矩阵分块如下

$$\boldsymbol{C}_{N+1} = \begin{pmatrix} \boldsymbol{C}_N & \boldsymbol{k} \\ \boldsymbol{k} & c \end{pmatrix} \tag{9.50}$$

其中 $\boldsymbol{C}_N$ 是一个 $N \times N$ 的协方差矩阵，向量 $\boldsymbol{k}$ 的元素为 $k(\boldsymbol{x}_n, \boldsymbol{x}_{N+1})$，其中 $n = 1, \ldots, N$，标量 $c = k(\boldsymbol{x}_{N+1}, \boldsymbol{x}_{N+1})$。使用公式 2.61 和公式 2.62，我们看到条件概率分布 $p(t_{N+1}|\boldsymbol{t})$ 是一个高斯分布，均值和协方差为

$$m(\boldsymbol{x}_{N+1}) = \boldsymbol{k}^T \boldsymbol{C}_N^{-1} \boldsymbol{t} \tag{9.51}$$

$$\sigma^2(\boldsymbol{x}_{N+1}) = c - \boldsymbol{k}^T \boldsymbol{C}_N^{-1} \boldsymbol{k} \tag{9.52}$$

这些是定义高斯过程回归的关键结果。注意，预测分布的均值可以写成 $\boldsymbol{x}_{N+1}$ 的函数，形式为

$$m(\boldsymbol{x}_{N+1}) = \sum_{n=1}^{N} a_n k(\boldsymbol{x}_n, \boldsymbol{x}_{N+1}) \tag{9.53}$$

其中 a_n 是 $\boldsymbol{C}_N^{-1}\boldsymbol{t}$ 的第 n 个元素。

使用高斯过程的核心计算涉及到对 $N \times N$ 的矩阵求逆。高斯过程观点的一个优点是，我们可以处理那些只能通过无穷多的基函数表达的协方差函数。但是，对于大的训练数据集，直接应用高斯过程方法就变得不可行了，因此一系列近似的方法被提出来。

学习超参数

高斯过程模型的预测部分依赖于协方差函数的选择。在实际应用中，我们不固定协方差函数，而是更喜欢使用一组带有参数的函数，然后从数据中推断参数的值。这些参数控制了相关性的长度缩放以及噪声的精度等等，对应于标准参数模型的超参数。

学习超参数的方法基于计算似然函数 $p(\boldsymbol{t}|\boldsymbol{\theta})$，其中 $\boldsymbol{\theta}$ 表示高斯过程模型的超参数。最简单的方法是通过最大化似然函数的方法进行 $\boldsymbol{\theta}$ 的点估计。由于 $\boldsymbol{\theta}$ 表示回归问题的一组超参数，因此这可以看成类似于线性回归模型的第二类最大似然步骤。可以使用高效的基于梯度的最优化算法来最大化对数似然函数。

使用多元高斯分布的标准形式，高斯过程模型的对数似然函数很容易计算。对数似然

函数的形式为

$$\ln p(\boldsymbol{t}|\boldsymbol{\theta}) = -\frac{1}{2}\ln|\boldsymbol{C}_N| - \frac{1}{2}\boldsymbol{t}^T\boldsymbol{C}_N^{-1}\boldsymbol{t} - \frac{N}{2}\ln(2\pi) \tag{9.54}$$

对于非线性最优化，我们也需要对数似然函数关于参数向量 $\boldsymbol{\theta}$ 的梯度。

$$\begin{aligned}\frac{\partial}{\partial\theta_i}\ln p(\boldsymbol{t}|\boldsymbol{\theta}) &= -\frac{1}{2}\frac{\partial[\ln|\boldsymbol{C}_N|]}{\partial\theta_i} - \frac{1}{2}\boldsymbol{t}^T\frac{\partial[\boldsymbol{C}_N^{-1}]}{\partial\theta_i}\boldsymbol{t} \\ &= -\frac{1}{2}Tr\left(\boldsymbol{C}_N^{-1}\frac{\partial\boldsymbol{C}_N}{\partial\theta_i}\right) + \frac{1}{2}\boldsymbol{t}^T\boldsymbol{C}_N^{-1}\frac{\partial\boldsymbol{C}_N}{\partial\theta_i}\boldsymbol{C}_N^{-1}\boldsymbol{t}\end{aligned} \tag{9.55}$$

由于 $\ln p(\boldsymbol{t}|\boldsymbol{\theta})$ 通常是一个非凸函数，因此它有多个极大值点。

引入一个 θ 上的先验分布然后基于梯度的方法最大化对数后验是很容易的。在一个纯粹的贝叶斯方法中，我们需要计算 $\boldsymbol{\theta}$ 的边缘概率，乘以先验概率 $p(\boldsymbol{\theta})$ 和似然函数 $p(\boldsymbol{t}|\boldsymbol{\theta})$。然而，通常精确的积分或者求和是不可行的，我们必须进行近似。

自动相关性确定

我们看到最大似然方法如何被用于确定高斯过程中的长度缩放参数的值。通过为每个输入变量整合到一个单独的参数，这种方法可以很有用地推广。正如我们将看到的那样，这样做的结果是，通过最大似然方法进行的参数最优化，能够将不同输入的相对重要性从数据中推断出来。这是高斯过程中的自动相关性确定 (automatic relecance detemination) 或者 ARD 的一个例子。

考虑二维输入空间 $\boldsymbol{x} = (x_1, x_2)$，有一个下面形式的核函数

$$k(\boldsymbol{x}, \boldsymbol{x}') = \theta_0\exp\left\{-\frac{1}{2}\sum_{i=1}^{2}\eta_i(x_i - x_i')^2\right\} \tag{9.56}$$

随着特定的 η_i 的减小，函数逐渐对对应的输入变量 x_i 不敏感。通过使用最大似然法按照数据集调整这些参数，它可以检测到对于预测分布几乎没有影响的输入变量。

ARD 框架很容易整合到指数-二次核中，得到下面形式的核函数，它对于一大类将高斯过程应用于回归问题的实际应用都很有帮助。

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \theta_0\exp\left\{-\frac{\theta_1}{2}\sum_{i=1}^{D}\eta_i(x_{ni} - x_{mi})^2\right\} + \theta_2 + \theta_3\sum_{i=1}^{D}\boldsymbol{x}_{ni}\boldsymbol{x}_{mi} \tag{9.57}$$

其中 D 是输入空间的维度。

用于分类的高斯过程

在分类的概率方法中，我们的目标是在给定一组训练数据的情况下，对于一个新的输入向量，为目标变量的后验概率建模。这些概率一定位于区间 [0, 1] 中，而一个高斯过程模型做出的预测位于整个实数轴上。然而，我们可以很容易地调整高斯过程，使基能够处理分类问题。方法为：使用一个恰当的非线性激活函数，将高斯过程的输出进行变换。

首先考虑一个二分类问题。如果我们定义函数 $a(\boldsymbol{x})$ 上的一个高斯过程，然后使用 logistic sigmoid 函数 $y = \sigma(a)$ 进行变换。那么我们就得到了函数 $y(\boldsymbol{x})$ 上的一个非高斯随机过程。一维输入空间的情况下，目标变量 t 上的概率分布是伯努利分布

$$p(t|a) = \sigma(a)^t(1 - \sigma(a))^{1-t} \tag{9.58}$$

训练集的输入记作 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$，对应的观测目标变量为 $\boldsymbol{t} = (t_1, \ldots, t_N)^T$。我们还考虑一个单一的观测数据点 $\boldsymbol{x}_{N+1}$，目标值为 t_{N+1}。我们的目标是确定预测分布 $p(t_{N+1}|\boldsymbol{t})$，其中我们没有显示地写出它对于输入变量的条件依赖。为了完成这个目标，我们引入向量 $\boldsymbol{a}_{N+1}$ 上的高斯过程先验，它的分量为 $a(\boldsymbol{x}_1), \ldots, a(\boldsymbol{x}_{N+1})$ 这反过来定义了 $\boldsymbol{t}_{N+1}$ 上的一个非高斯过程。通过以训练数据 $\boldsymbol{t}_N$ 为条件，我们得到了求解的预测分布。$\boldsymbol{a}_{N+1}$ 上的高斯过程先验的形式为

$$p(\boldsymbol{a}_{N+1}) = \mathcal{N}(\boldsymbol{a}_{N+1}|0, \boldsymbol{C}_{N+1}) \tag{9.59}$$

协方差矩阵不包含噪声项，然而，由于数值计算的原因，更方便的做法是引入一个由参数 v 控制的类似噪声的项，它确保了协方差矩阵是正定的。因此协方差矩阵 $\boldsymbol{C}_{N+1}$ 的元素为

$$C(\boldsymbol{x}_n, \boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m) + v\sigma_{nm} \tag{9.60}$$

其中 $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ 是任意的半正定核函数，v 的值通常事先固定。我们会假定核函数由参数向量 $\boldsymbol{\theta}$ 控制，稍后会讨论如何从训练数据中学习到 $\boldsymbol{\theta}$。

求解的预测分布为

$$p(t_{N+1}|\boldsymbol{t}_N) = \int p(t_{N+1} = 1|a_{N+1})p(a_{N+1}|\boldsymbol{t}_N)da_{N+1} \tag{9.61}$$

其中 $p(t_{N+1} = 1|a_{N+1}) = \sigma(\boldsymbol{a}_{N+1})$ 这个积分无法求出解析解，可以采用近似的方法。

1. 变分推断
2. 期望传播
3. 拉普拉斯近似

拉普拉斯近似

为了计算预测分布 9.61，我们寻找 a_{N+1} 的后验概率分布的高斯近似。使用贝叶斯定理，后验概率分布为

$$\begin{aligned} p(a_{N+1}|\boldsymbol{t}_N) &= \int p(a_{N+1}, \boldsymbol{a}_N|\boldsymbol{t}_N)d\boldsymbol{a}_N \\ &= \frac{1}{p(\boldsymbol{t}_N)}\int p(a_{N+1}, \boldsymbol{a}_N, \boldsymbol{t}_N)d\boldsymbol{a}_N \\ &= \frac{1}{p(\boldsymbol{t}_N)}\int p(a_{N+1}, \boldsymbol{a}_N)p(\boldsymbol{t}_N|a_{N+1}, \boldsymbol{a}_N)d\boldsymbol{a}_N \\ &= \int p(a_{N+1}|\boldsymbol{a}_N)\frac{1}{p(\boldsymbol{t}_N)}p(\boldsymbol{a}_N)p(\boldsymbol{t}_N|\boldsymbol{a}_N)d\boldsymbol{a}_N \\ &= \int p(a_{N+1}|\boldsymbol{a}_N)p(\boldsymbol{a}_N|\boldsymbol{t}_N)d\boldsymbol{a}_N \end{aligned} \tag{9.62}$$

其中，我们用到了 $p(\boldsymbol{t}_N|a_{N+1}, \boldsymbol{a}_N) = p(\boldsymbol{t}_N|\boldsymbol{a}_N)$。使用公式 9.51 和公式 9.52 给出的高斯过程回归的结果，我们可以得到条件概率分布 $p(a_{N+1}|\boldsymbol{a}_N)$，结果为

$$p(a_{N+1}|\boldsymbol{a}_N) = \mathcal{N}(a_{N+1}|\boldsymbol{k}^T\boldsymbol{C}_N^{-1}\boldsymbol{a}_N, c - \boldsymbol{k}^T\boldsymbol{C}_N^{-1}\boldsymbol{k}) \tag{9.63}$$

于是，通过找到后验概率分布 $p(\boldsymbol{a}_N|\boldsymbol{t}_N)$ 的拉普拉斯近似，然后使用两个高斯分布卷积的标准结果，我们就可以计算公式中的积分。

先验概率 $p(\boldsymbol{a}_N)$ 由一个零均值高斯过各给出，协方差矩阵为 $\boldsymbol{C}_N$，数据项 (假设数据点之间具有独立性) 为

$$p(\boldsymbol{t}_N|\boldsymbol{a}_N) = \prod_{n=1}^{N}\sigma(a_n)^{t_n}(1-\sigma(a_n))^{1-t_n} = \prod_{n=1}^{N}e^{a_n t_n}\sigma(-a_n) \tag{9.64}$$

然后通过对 $p(\boldsymbol{a}_N|\boldsymbol{t}_N)$ 的对数进行泰勒展开，就可以得到拉普拉斯近似。忽略掉一些具有可加性的常数，这个概率的对数为

$$\begin{aligned} \Psi(\boldsymbol{a}_N) &= \ln p(\boldsymbol{a}_N, \boldsymbol{t}_N) \quad \text{忽略了常数项} \\ &= \ln p(\boldsymbol{a}_N) + \ln p(\boldsymbol{t}_N|\boldsymbol{a}_N) \\ &= \underbrace{-\frac{1}{2}\boldsymbol{a}_N^T\boldsymbol{C}_N^{-1}\boldsymbol{a}_N - \frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{C}_N|}_{\boldsymbol{a}_N \text{ 服从零均值高斯分布}} \\ &\quad \underbrace{+\boldsymbol{t}_N^T\boldsymbol{a}_N - \sum_{n=1}^{N}\ln(1+e^{a_n})}_{\text{二项分布}} \end{aligned} \tag{9.65}$$

首先我们需要找到后验概率分布的众数，这需要我们计算 $\Psi(\boldsymbol{a}_N)$ 的梯度。这个梯度为

$$\nabla\Psi(\boldsymbol{a}_N) = \boldsymbol{t}_N - \boldsymbol{\sigma}_N - \boldsymbol{C}_N^{-1}\boldsymbol{a}_N \tag{9.66}$$

其中 $\boldsymbol{\sigma}_N$ 是一个元素为 $\sigma(a_n)$ 的向量。寻找众数时，我们不能简单地令这个梯度等于零，因为 $\boldsymbol{\sigma}_N$ 与 $\boldsymbol{a}_N$ 的关系是非线性的，因此我们需要使用基于 Newton-Raphson 方法的迭代的方法，它给出了一个迭代重加权最小平方 (IRLS) 算法。这需要求 $\Psi(\boldsymbol{a}_N)$ 的二阶导数，而这个二阶导数也需要进行拉普拉斯近似，结果为

$$\nabla\nabla\Psi(\boldsymbol{a}_N) = -\boldsymbol{W}_N - \boldsymbol{C}_N^{-1} \tag{9.67}$$

其中 $\boldsymbol{W}_N$ 是一个对角矩阵，元素为 $\sigma(a_n)(1-\sigma(a_n))$，并且使用了 logistic sigmoid 函数的导数的结果

$$\frac{d\sigma}{da} = \sigma(1-\sigma) \tag{9.68}$$

注意，这些对角矩阵元素位于区间 $(0, \frac{1}{4})$，因此 $\boldsymbol{W}_N$ 是一个正定矩阵。由于 $\boldsymbol{C}_N$ 被构造成正定的，并且由于两个正定矩阵的和仍然是正定矩阵，因此我们看到 Hessian 矩阵 $\boldsymbol{A} = -\nabla\nabla\Psi(\boldsymbol{a}_N)$ 是正定的，因此后验概率分布 $p(\boldsymbol{a}_N|\boldsymbol{t}_N)$ 是对数凸函数，因此有一个唯一的众数，即全局最大值。然而，后验概率不是高斯分布，因为 Hessian 矩阵是 $\boldsymbol{a}_N$ 的函数。

使用 Newton-Raphson 公式，$\boldsymbol{a}_N$ 的迭代更新方程为

$$\boldsymbol{a}_N^{\text{新}} = \boldsymbol{C}_N(\boldsymbol{I} + \boldsymbol{W}_N\boldsymbol{C}_N)^{-1}\{\boldsymbol{t}_N - \boldsymbol{\sigma}_N + \boldsymbol{W}_N\boldsymbol{a}_N\} \tag{9.69}$$

这个方程反复迭代，直到收敛于众数 (记作 $\boldsymbol{a}_N^*$)。在这个众数位置，梯度 $\nabla\Psi(\boldsymbol{a}_N)$ 为零，因此 $\boldsymbol{a}_N^*$ 满足

$$\boldsymbol{a}_N^* = \boldsymbol{C}_N(\boldsymbol{t}_N - \boldsymbol{\sigma}_N) \tag{9.70}$$

一旦我们找到了后验概率的众数 $\boldsymbol{a}_N^*$，我们就可以计算 Hessian 矩阵，结果为

$$\boldsymbol{H} = -\nabla\nabla\Psi(\boldsymbol{a}_N) = \boldsymbol{W}_N + \boldsymbol{C}_N^{-1} \tag{9.71}$$

其中 $\boldsymbol{W}_N$ 的元素使用 $\boldsymbol{a}_N^*$ 计算。这定义了我们对后验概率分布 $p(\boldsymbol{a}_N|\boldsymbol{t}_N)$ 的高斯近似，结果为

$$q(\boldsymbol{a}_N) = \mathcal{N}(\boldsymbol{a}_N|\boldsymbol{a}_N^*, \boldsymbol{H}^{-1}) \tag{9.72}$$

我们现在可以将这个结果与公式 9.63 结合，然后计算积分 9.62。因为这对应于线性高斯模型，我们可以使用一般的结果得到

$$\mathbb{E}[a_{N+1}|\boldsymbol{t}_N] = \boldsymbol{k}^T(\boldsymbol{t}_N - \boldsymbol{\sigma}_N) \tag{9.73}$$

$$var[a_{N+1}|\boldsymbol{t}_N] = c - \boldsymbol{k}^T(\boldsymbol{W}_N^{-1} + \boldsymbol{C}_N)^{-1}\boldsymbol{k} \tag{9.74}$$

现在我们有一个 $p(a_{N+1}|\boldsymbol{t}_N)$ 的高斯分布，我们可以使用 6.99 的结果近似积分 9.62。

我们还需要确定协方差函数的参数 $\boldsymbol{\theta}$。一种方法是最大化似然函数 $p(\boldsymbol{t}_N|\boldsymbol{\theta})$，此时我们需要对数似然函数和它的梯度的表达式。如果必要的话，还可以加上正则化项，产生一个正则化的最大似然解。

与神经网络的联系

在贝叶斯神经网络中，参数向量 $\boldsymbol{w}$ 上的先验分布以及网络函数 $f(\boldsymbol{x}, \boldsymbol{w})$ 产生了函数 $y(\boldsymbol{x})$ 上的先验概率分布，其中 $\boldsymbol{y}$ 是网络输出向量。在极限 $M \Rightarrow \infty$ 的情况下，对于 $\boldsymbol{w}$ 的一大类先验分布，神经网络产生的函数的分布将会趋于高斯过程。

第 10 章 混合模型和 EM 算法

10.1 一般形式的 EM 算法

EM 算法是一种迭代算法，1977 年由 Dempster 等人总结提出的，用于含有隐变量 (hidden variable) 的概率模型参数的极大似然估计，或极大后验概率估计。EM 算法的每次迭代由两步组成：E 步，求期望 (expectation)；M 步，求极大 (maximization)。所以这一算法称为期望极大算法 (expection maximization)，简称 EM 算法。

概率模型有时既含有观测变量 (observable variable)，又含有隐变量或潜在变量 (latent variable)。如果概率模型的变量都是观测变量，那么给定数据，可以直接用极大似然估计法，或贝叶斯估计法估计模型参数。但是，当模型含有隐变量时，就不能简单地使用这些估计方法。EM 算法就是含有隐变量的概率模型参数的极大似然估计法，或极大后验概率估计法。仅讨论极大似然估计，极大后验概率估计与其类似。

EM 算法的导出

为什么 EM 算法能近似实验对观测数据的极大似然估计呢？下面通过近似求解观测数据的对数似然函数的极大化问题来导出 EM 算法。

我们面对一个含有隐变量的概率模型，目标是极大化观测数据 (不完全数据)Y 关于参数 θ 的对数似然函数，即极大化

$$\begin{aligned} L(\theta) &= logP(Y|\theta) = log\sum_{Z} P(Y,Z|\theta) \\ &= log\left(\sum_{Z} P(Y|Z,\theta)P(Z|\theta)\right) \end{aligned} \tag{10.1}$$

注意到这一极大化的主要困难是式 10.1 中有未观测数据并有包含和 (或积分) 的对数。

事实上，EM 算法是通过迭代逐步极大化 $L(\theta)$ 的。假设在第 i 次迭代后 θ 的估计值是 $\theta^{(i)}$。我们希望新估计值 θ 能使 $L(\theta)$ 增加，即 $L(\theta) > L(\theta^{(i)})$，并逐步达到极大值。为此，考虑两者的差：

$$L(\theta) - L(\theta^{(i)}) = log\left(\sum_{Z} P(Y|Z,\theta)P(Z|\theta)\right) - logP(Y|\theta^{(i)}) \tag{10.2}$$

利用 Jensen 不等式[1](Jensen inequality) 得到其下界：

$$\begin{aligned} L(\theta) - L(\theta^{(i)}) &= log\left(\sum_Z P(Y|Z,\theta^{(i)})\frac{P(Y|Z,\theta)P(Z|\theta)}{P(Y|Z,\theta^{(i)})}\right) - logP(Y|\theta^{(i)}) \\ &\geqslant \sum_Z P(Y|Z,\theta^{(i)})log\frac{P(Y|Z,\theta)P(Z|\theta)}{P(Y|Z,\theta^{(i)})} - logP(Y|\theta^{(i)}) \\ &= \sum_Z P(Y|Z,\theta^{(i)})log\frac{P(Y|Z,\theta)P(Z|\theta)}{P(Y|Z,\theta^{(i)})P(Y|\theta^{(i)})} \end{aligned} \tag{10.3}$$

令

$$B(\theta,\theta^{(i)}) = L(\theta^{(i)}) + \sum_Z P(Y|Z,\theta^{(i)})log\frac{P(Y|Z,\theta)P(Z|\theta)}{P(Y|Z,\theta^{(i)})P(Y|\theta^{(i)})} \tag{10.4}$$

则

$$L(\theta) \geqslant B(\theta,\theta^{(i)}) \tag{10.5}$$

即函数 $B(\theta,\theta^{(i)})$ 是 $L(\theta)$ 的一个下界，因此**任何可以使 $B(\theta,\theta^{(i)})$ 增大的 θ，也可以使 $L(\theta)$ 增大**。为了使 $L(\theta)$ 尽可能大的增大，选择 $\theta^{(i+1)}$ 使 $B(\theta,\theta^{(i)})$ 达到极大，即

$$\theta^{(i+1)} = arg\ \max_\theta B(\theta,\theta^{(i)}) \tag{10.6}$$

现在求 $\theta^{(i+1)}$ 的表达式。省去对 θ 的极大化而言是常数的项，有

$$\begin{aligned} \theta^{(i+1)} &= arg\ \max_\theta \left(L(\theta^{(i)}) + \sum_Z P(Y|Z,\theta^{(i)})log\frac{P(Y|Z,\theta)P(Z|\theta)}{P(Y|Z,\theta^{(i)})P(Y|\theta^{(i)})}\right) \\ &= arg\ \max_\theta \left(\sum_Z P(Y|Z,\theta^{(i)})log(P(Y|Z,\theta)P(Z|\theta))\right) \\ &= arg\ \max_\theta \left(\sum_Z P(Y|Z,\theta^{(i)})log(P(Y,Z|\theta))\right) \\ &= arg\ \max_\theta \underline{Q(\theta,\theta^{(i)})} \end{aligned} \tag{10.7}$$

式 10.7 等价于 EM 算法的一次迭代，即求 Q 函数及其极大化。**EM 算法是通过不断求解下界的极大化逼近求解对数似然函数极大化的算法**。

定义 10.1. Q 函数

完全数据的对数似然函数 $logP(Y,Z|\theta)$ 关于在给定观测数据 Y 和当前参数 $\theta^{(i)}$ 下对未观测数据 Z 的条件概率分布 $P(Z|Y,\theta^{(i)})$ 的期望称为 Q 函数。即

$$Q(\theta,\theta^{(i)}) = E_Z[logP(Y,Z|\theta)|Y,\theta^{(i)}] \tag{10.8}$$

[1]这里用到的是 $log\sum_j \lambda_j y_j \geqslant \sum_j \lambda_j log y_j$

EM 算法

输入：观测变量数据 Y，隐变量数据 Z，联合分布 $P(Y,Z|\theta)$，条件分布 $P(Z|Y,\theta)$；

输出：模型参数 θ

1. 选择参数的初值 $\theta^{(0)}$，开始迭代；
2. E 步：记 $\theta^{(i)}$ 为第 i 次迭代参数 θ 的估计值，在第 i+1 次迭代的 E 步，计算

$$\begin{aligned} Q(\theta,\theta^{(i)}) &= E_Z[logP(Y,Z|\theta)|Y,\theta^{(i)}] \\ &= \sum_Z logP(Y,Z|\theta)P(Z|Y,\theta^{(i)}) \end{aligned} \tag{10.9}$$

这里，$P(Z|Y,\theta^{(i)})$ 是在给定观测数据 Y 和当前的参数估计 $\theta^{(i)}$ 下隐变量数据 Z 的条件概率分布；

3. M 步：求使 $Q(\theta,\theta^{(i)})$ 极大化的 θ，确定第 i+1 次迭代的参数的估计值 $\theta^{(i+1)}$

$$\theta^{(i+1)} = arg\ \max_\theta Q(\theta,\theta^{(i)}) \tag{10.10}$$

4. 重复第 (2)、(3) 步，直到收敛。

EM 算法的收敛性

EM 算法提供一种近似计算含有隐变量概率模型的极大似然估计的方法。EM 算法的最大优点是简单性和普适性。我们自然地要问：EM 算法得到的估计序列是否收敛？如果收敛，是否收敛到全局最大值或局部极大值？

定理 10.1

设 $P(Y|\theta)$ 为观测数据的似然函数，$\theta^{(i)}(i=1,2,\dots)$ 为 EM 算法得到的参数估计序列，$P(Y|\theta^{(i)})(i=1,2,\dots)$ 为对应的似然函数序列，则 $P(Y|\theta^{(i)})$ 是单调递增的，即

$$P(Y|\theta^{(i+1)}) \geqslant P(Y|\theta^{(i)}) \tag{10.11}$$

设 $L(\theta) = logP(Y|\theta)$ 为观测数据的对数似然函数，$L(\theta^{(i)})(i=1,2,\dots)$ 为对应的对数似然函数序列。

1. 如果 $P(Y|\theta)$ 有上界，则 $L(\theta^{(i)}) = logP(Y|\theta^{(i)})$ 收敛到某一值 L^*
2. 在函数 $Q(\theta,\theta')$ 与 $L(\theta)$ 满足一定条件下，由 EM 算法得到的参数估计序列 $\theta^{(i)}$ 的收敛值 θ^* 是 $L(\theta)$ 的稳定点 ♡

10.2 EM 的另一种观点

10.3 K 均值聚类

10.4 混合高斯

EM 算法的一个重要应用是高斯混合模型的参数估计。高斯混合模型应用广泛，在许多情况下，EM 算法是学习高斯混合模型 (Gaussian misture model) 的有效方法。

定义 10.2. 高斯混合模型

高斯混合模型是指具有如下形式的概率分布模型：

$$P(y|\theta) = \sum_{k=1}^{K} \alpha_k \phi(y|\theta_k) \tag{10.12}$$

其中，α_k 是系数，$\alpha_k \geqslant 0, \sum_{k=1}^{K} \alpha_k = 1$；$\phi(Y|\theta_k)$ 是高斯分布密度，$\theta_k = (\mu_k, \sigma_k^2)$，

$$\phi(y|\theta_k) = \frac{1}{\sqrt{2\pi}\sigma_k} exp\left(-\frac{(y-\mu_k)^2}{2\sigma_k^2}\right) \tag{10.13}$$

称为第 k 个模型。 ♣

高斯混合模型参数估计的 EM 算法

假设观测数据 $y_1, y_2, \ldots, y_N$ 由高斯混合模型生成，

$$P(y|\theta) = \sum_{k=1}^{K} \alpha_k \phi(y|\theta_k) \tag{10.14}$$

其中，$\theta = (\alpha_1, \alpha_2, \ldots, \alpha_K; \theta_1, \theta_2, \ldots, \theta_K)$。我们用 EM 算法估计高斯混合模型的参数 θ

1. 明确隐变量，写出完全数据的对数似然函数

   隐变量由 γ_{jk} 表示，其定义如下：

$$\gamma_{jk} = \begin{cases} 1, & \text{第 j 个观测来自第 k 个模型} \\ 0, & \text{否则} \end{cases} \tag{10.15}$$

   γ_{jk} 是 0-1 随机变量。那么完全数据是

$$(y_j, \gamma_{j1}, \gamma_{j2}, \ldots, \gamma_{jK}), \qquad j = 1, 2, \ldots, K$$

于是可以写出完全数据的似然函数

$$\begin{aligned}P(y,\gamma|\theta) &= \prod_{j=1}^{N} P(y_j,\gamma_{j1},\gamma_{j2},\ldots,\gamma_{jK}|\theta)\\ &= \prod_{k=1}^{K}\prod_{j=1}^{N}[\alpha_k\phi(y|\theta_k)]^{\gamma_{jk}}\\ &= \prod_{k=1}^{K}\alpha_k^{n_k}\prod_{j=1}^{N}[\phi(y|\theta_k)]^{\gamma_{jk}}\\ &= \prod_{k=1}^{K}\alpha_k^{n_k}\prod_{j=1}^{N}\left[\frac{1}{\sqrt{2\pi}\sigma_k}exp\left(-\frac{(y-\mu_k)^2}{2\sigma_k^2}\right)\right]^{\gamma_{jk}}\end{aligned}$$

式中，$n_k=\sum_{j=1}^{N}\gamma_{jk}$，$\sum_{k=1}^{K}n_k=N$。(意思是选择第 k 个高斯模型的次数)

那么，完全数据的对数似然函数为

$$logP(y,\gamma|\theta)=\sum_{k=1}^{K}\left\{n_k log\alpha_k+\sum_{j=1}^{N}\gamma_{jk}\left[log\left(\frac{1}{\sqrt{2\pi}}\right)-log\sigma_k-\frac{1}{2\sigma_k^2}(y_j-\mu_k)^2\right]\right\}$$

2. EM 算法的 E 步：确定 Q 函数

$$\begin{aligned}Q(\theta,\theta^{(i)}) &= E[logP(y,\gamma|\theta)|y,\theta^{(i)}]\\ &= E\left\{\sum_{k=1}^{K}\left\{n_k log\alpha_k+\sum_{j=1}^{N}\gamma_{jk}\left[log\left(\frac{1}{\sqrt{2\pi}}\right)-log\sigma_k-\frac{1}{2\sigma_k^2}(y_j-\mu_k)^2\right]\right\}\right\}\\ &= \sum_{k=1}^{K}\left\{\overbrace{n_k log\alpha_k}^{\text{常数}}+\sum_{j=1}^{N}E(\gamma_{jk})\left[log\left(\frac{1}{\sqrt{2\pi}}\right)-log\sigma_k-\frac{1}{2\sigma_k^2}(y_j-\mu_k)^2\right]\right\}\end{aligned}$$

这里需要计算 $E(\gamma_{jk}|y,\theta)$，记为 $\hat{\gamma}_{jk}$

$$\begin{aligned}\hat{\gamma}_{jk} &= E(\gamma_{jk}|y,\theta)=P(\gamma_{jk}=1|y,\theta)(\text{二项分布})\\ &= \frac{P(\gamma_{jk}=1|y,\theta)}{\sum_{k=1}^{K}P(\gamma_{jk}=1|y,\theta)}\\ &= \frac{P(y_j|\gamma_{jk}=1,\theta)P(\gamma_{jk}=1|\theta)}{\sum_{k=1}^{K}P(y_j|\gamma_{jk}=1,\theta)P(\gamma_{jk}=1|\theta)}\\ &= \frac{\alpha_k\phi(y_j|\theta_k)}{\sum_{k=1}^{K}\alpha_k\phi(y_j|\theta_k)},\quad j=1,2,\ldots,N;k=1,2,\ldots,K\end{aligned}$$

$\hat{\gamma}_{jk}$ 是在当前模型参数下第 j 个观测数据来自第 k 个分模型的概率，称为分模型 k 对观测数据 y_j 的响应度。代入式中得

$$Q(\theta, \theta^{(i)}) = \sum_{k=1}^{K} \left\{ n_k log\alpha_k + \sum_{j=1}^{N} \hat{\gamma}_{jk} \left[log\left(\frac{1}{\sqrt{2\pi}}\right) - log\sigma_k - \frac{1}{2\sigma_k^2}(y_j - \mu_k)^2 \right] \right\} \tag{10.16}$$

3. 确定 EM 算法的 M 步迭代的 M 步是求函数 $Q(\theta, \theta^{(i)})$ 对 θ 的极大值，即求新一轮迭代的模型参数：

$$\theta^{(i+1)} = arg\ \max_{\theta} Q(\theta, \theta^{(i)})$$

分别对 μ_k, σ_k^2 求偏导并令其为 0，即可得到 $\hat{\mu}_k, \hat{\sigma}_k$。求 $\hat{a}_k$ 是在 $\sum_{k=1}^{K} = 1$ 条件下求偏导并令其为 0 得到。

$$\hat{\mu}_k = \frac{\sum_{j=1}^{N} \hat{\gamma}_{jk} y_j}{\sum_{j=1}^{N} \hat{\gamma}_{jk}}, \quad k = 1, 2, \ldots, K \tag{10.17}$$

$$\hat{\sigma}_k^2 = \frac{\sum_{j=1}^{N} \hat{\gamma}_{jk}(y_j - \mu_k)^2}{\sum_{j=1}^{N} \hat{\gamma}_{jk}}, \quad k = 1, 2, \ldots, K \tag{10.18}$$

$$\hat{\alpha}_k = \frac{n_k}{N} = \frac{\sum_{j=1}^{N} \hat{\gamma}_{jk}}{N}, \quad k = 1, 2, \ldots, K \tag{10.19}$$

$$\tag{10.20}$$

4. 重复以上计算，直到对数似然函数值不再有明显的变化为止。

现将估计高斯混合模型参数的 EM 算法总结如下

输入：观测数据 $y_1, y_2, \ldots, y_N$，高斯混合模型；

输出：高斯混合模型参数。

(1) 取参数的初始值开始迭代

(2) E 步：依据当前模型参数，计算分模型 k 对观测数据 y_j 的响应度

$$\hat{\gamma}_{jk} = \frac{\alpha_k \phi(y_j|\theta_k)}{\sum_{k=1}^{K} \alpha_k \phi(y_j|\theta_k)}, \quad j = 1, 2, \ldots, N; k = 1, 2, \ldots, K$$

(3) M 步:计算新一轮迭代的模型参数

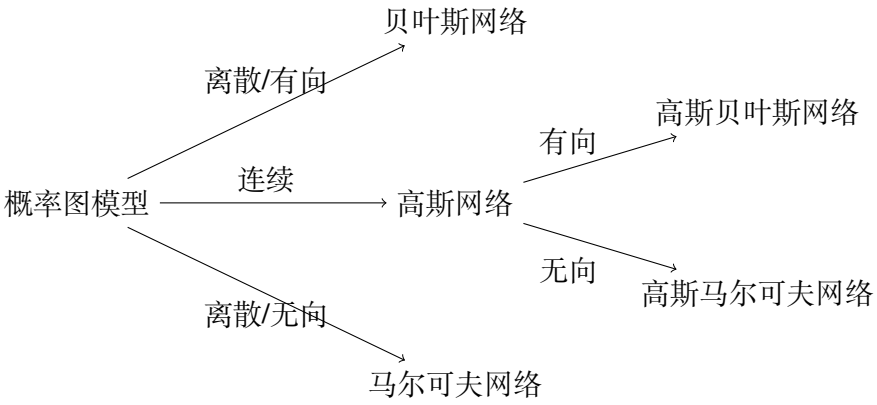
$$\hat{\mu}_k = \frac{\sum_{j=1}^{N} \hat{\gamma}_{jk} y_j}{\sum_{j=1}^{N} \hat{\gamma}_{jk}}, \quad k = 1, 2, \ldots, K$$

$$\hat{\sigma}_k^2 = \frac{\sum_{j=1}^{N} \hat{\gamma}_{jk} (y_j - \mu_k)^2}{\sum_{j=1}^{N} \hat{\gamma}_{jk}}, \quad k = 1, 2, \ldots, K$$

$$\hat{\alpha}_k = \frac{n_k}{N} = \frac{\sum_{j=1}^{N} \hat{\gamma}_{jk}}{N}, \quad k = 1, 2, \ldots, K$$

(4) 重复以上计算,直到对数似然函数值不再有明显的变化为止。

第 11 章 概率图模型



11.1 贝叶斯网络

11.2 条件独立

11.3 马尔科夫随机场

11.4 图模型中的推断

11.5 隐马尔可夫模型

隐马尔可夫模型 (hidden Markov model,HMM) 是可用于标注问题的统计学习模型,描述由隐藏的马尔可夫链随机生成观测序列的过程,属于生成模型。本节首先介绍隐马尔可夫模型的基本概念,然后分别叙述隐马尔可夫模型的概率计算算法、学习算法以及预测算法。隐马尔可夫模型在语音识别、自然语言处理、生物信息、模式识别等领域有着广泛的应用。

隐马尔可夫模型的基本概念

隐马尔可夫模型是关于时序的概率模型,描述由一个隐藏的马尔可夫链随机生成不可观测的状态随机序列,再由各个状态生成一个观测而产生观测随机序列的过程。隐藏的马尔可夫链随机生成的状态的序列,称为状态序列 (state sequence);每个状态生成一个观测,而由此产生的观测的随机序列,称为观测序列 (observation sequence)。序列的每一个位置又可以看作是一个时刻。隐马尔可夫模型由初始概率分布、状态转移概率分布以及观测概率分布确定。隐马尔可夫模型定义如下:

设 Q 是所有可能的状态的集合，V 是所有可能的观测的集合。

$$Q = \{q_1, q_2, \ldots, q_N\}, \quad V = \{v_1, v_2, \ldots, v_M\} \tag{11.1}$$

设 N 是所有可能的状态数，M 是可能的观测数。
I 是长度为 T 的状态序列，O 是对应的观测序列。

$$I = \{i_1, i_2, \ldots, i_T\}, \quad O = \{o_1, o_2, \ldots, o_T\} \tag{11.2}$$

A 是状态转移概率矩阵：

$$A = \left[a_{ij}\right]_{N\times N} \tag{11.3}$$

其中，

$$a_{ij} = P(i_{t+1} = q_j | i_t = q_i), i = 1, 2, \ldots, N; j = 1, 2, \ldots, N \tag{11.4}$$

是在时刻 t 处于状态 q_i 的条件下在时刻 $t + 1$ 转移到状态 q_j 的概率。
B 是观测概率矩阵：

$$B = \left[b_j(k)\right]_{N\times N} \tag{11.5}$$

其中，

$$b_j(k) = P(o_t = v_k | i_t = q_j), k = 1, 2, \ldots, M; j = 1, 2, \ldots, N \tag{11.6}$$

是在时刻 t 处于状态 q_j 的条件下生成观测 v_k 的概率。π 是初始状态概率向量：

$$\pi = (\pi_i) \tag{11.7}$$

其中，

$$\pi_i = P(i_1 = q_i), i = 1, 2, \ldots, N \tag{11.8}$$

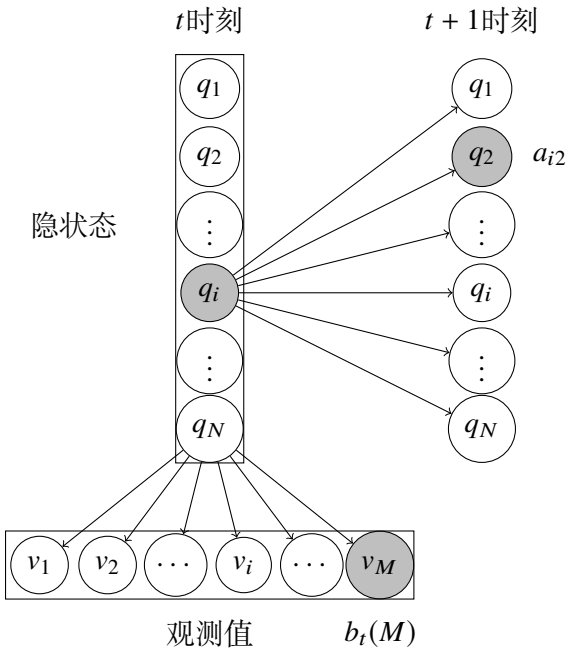
是时刻 $t = 1$ 处于状态 q_i 的概率。

隐马尔可夫模型由初始状态概率向量 π、状态转移概率矩阵 A 和观测概率矩阵 B 决定。π 和 A 决定状态序列，B 决定观测序列。因此，隐马尔可夫模型 λ 可以用三元符号表示，即

$$\lambda = (A, B, \pi) \tag{11.9}$$

A, B, π 称为隐马尔可夫模型的三要素。从定义可知，隐马尔可夫模型作了两个基本假设：

1. 齐次马尔可夫性假设，即假设隐藏的马尔可夫链在任意时刻 t 的状态只依赖于其前一时刻的状态，与其他时刻的状态及观测无关，也与时刻 t 无关。
2. 观测独立性假设，即假设任意时刻的观测只依赖于该时刻的马尔可夫链的状态，与其他观测及状态无关。



隐马尔可夫模型的 3 个基本问题

1. 概率计算问题。给定模型 $\lambda = (A, B, \pi)$ 和观测序列 $O = (o_1, o_2, \ldots, o_T)$，计算在模型 λ 下观测序列 O 出现的概率 $P(O|\lambda)$
2. 学习问题。已知观测序列 $O = (o_1, o_2, \ldots, o_T)$，估计模型 $\lambda = (A, B, \pi)$ 参数，使得在该模型下观测序列概率 $P(O|\lambda)$ 最大。即用极大似然估计的方法估计参数。
3. 预测问题。也称为解码（decoding）问题。已知模型 $\lambda = (A, B, \pi)$ 和观测序列 $O = (o_1, o_2, \ldots, o_T)$，求对给定观测序列条件概率 $P(I|\lambda)$ 最大的状态序列 $I = (i_1, i_2, \ldots, i_T)$。即给定观测序列，求最有可能的对应的状态序列。

问题一：概率计算算法

直接计算法

给定模型 $\lambda = (A, B, \pi)$ 和观测序列 $O = (o_1, o_2, \ldots, o_T)$，计算在模型 λ 下观测序列 O 出现的概率 $P(O|\lambda)$。最直接的方法是按概率公式直接计算。

$$\begin{aligned} P(O|\lambda) &= \sum_I P(O|I, \lambda) P(I|\lambda) \\ &= \sum_{i_1, i_2, \ldots, i_T} \pi_{i_1} b_{i_1}(o_1) a_{i_1 i_2} \ldots a_{i_{T-1} i_T} b_{i_T}(o_T) \end{aligned} \tag{11.10}$$

利用公式计算量大，这种算法不可行。

前向算法

首先定义前现概率

定义 11.1. 前向概率

给定隐马尔可夫模型 λ，定义到时刻 t 部分观测序列为 $O=(o_1,o_2,\ldots,o_t)$ 且状态为 q_i 的概率为前向概率，记作

$$\alpha_t(i)=P(o_1,o_2,\ldots,o_t,i_t=q_i|\lambda) \tag{11.11}$$

可以递推地求得前向概率 $\alpha_t(i)$ 及观测序列概率 $P(O|\lambda)$

输入：隐马尔可夫模型 λ，观测序列 O；

输出：观测序列概率 $P(O|\lambda)$

(1) 初值

$$\alpha_1(i)=\pi_i b_i(o_1),\qquad i=1,2,\ldots,N \tag{11.12}$$

(2) 递推，对 $t=1,2,\ldots,T-1$

$$\alpha_{t+1}(i)=\left[\sum_{j=1}^{N}\alpha_t(j)a_{ji}\right]b_i(o_{t+1}) \tag{11.13}$$

(3) 终止

$$P(O|\lambda)=\sum_{i=1}^{N}\alpha_T(i) \tag{11.14}$$

前向算法实际是基于“状态序列的路径结构”递推计算 $P(O|\lambda)$ 的算法。前身算法的高效的关键是其局部计算前向概率，然后利用路径结构将前向概率“递推”到全局，得到 $P(O|\lambda)$.

后向概率

定义 11.2. 后向算法

给定隐马尔可夫模型 λ，定义在时刻 t 状态为 q_i 的条件下，从 $t+1$ 到 T 的部分观测序列为 $o_{t+1},o_{t+2},\ldots,o_T$ 的概率为后向概率，记作

$$\beta_t(i)=P(o_{t+1},o_{t+2},\ldots,o_T|i_t=q_i,\lambda) \tag{11.15}$$

可以用递推的方法求得后向概率 $\beta_t(i)$ 及观测序列概率 $P(O|\lambda)$

输入：隐马尔可夫模型 λ，观测序列 O；

输出：观测序列概率 $P(O|\lambda)$

(1) 初值

$$\beta_T(i)=1,\qquad i=1,2,\ldots,N \tag{11.16}$$

(2) 递推，对 $t=T-1,T-2,\ldots,1$

$$\beta_t(i)=\sum_{j=1}^{N}a_{ij}b_j(o_{t+1})\beta_{t+1}(j),\quad i=1,2,\ldots,N \tag{11.17}$$

(3) 终止

$$P(O|\lambda)=\sum_{i=1}^{N}\pi_i b_i(o_1)\beta_1(i) \tag{11.18}$$

利用前向概率和后向概率的定义可以将观测序列概率 $P(O|\lambda)$ 统一写成

$$P(O|\lambda)=\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j),\quad t=1,2,\ldots,T-1 \tag{11.19}$$

一些概率与期望值的计算

1. 给定模型 λ 和观测 O,在时刻 t 处于状态 q_j 的概率。记 $\gamma_t(i)$
2. 给定模型 λ 和观测 O,在时刻 t 处于状态 q_j 且在时刻 $t+1$ 处于状态 q_j 的概率。记 $\xi_t(i,j)$
3. 一些有用的期望值

问题二:学习算法

隐马尔可夫模型的学习,根据训练数据是包括观测序列和对应的状态序列还是只有观测序列,可以分别由监督学习与非监督学习实现。

监督学习算法

假设训练数据包含 S 个长度相同的观测序列和对应的状态序列 $\{(O_1,I_1),\ldots,(O_S,I_S)\}$,那么可以利用**极大似然法**来估计隐马尔可夫模型的参数。具体方法如下。

1. 转移概率 a_{ij} 的估计

   设样本中时刻 t 处于状态 i 时刻 $t+1$ 转移到状态 j 的频数为 A_{ij},那么状态转移概率 a_{ij} 的估计是

$$\hat{a}_{ij}=\frac{A_{ij}]}{\sum_{j=1}^{N}A_{ij}},\qquad i=1,2,\ldots,N;j=1,2,\ldots,N \tag{11.20}$$

2. 观测概率 $b_j(k)$ 的估计

   设样本中状态为 j 并观测为 k 的频数是 B_{jk},那么状态为 j 观测为 k 的概率 $b_j(k)$ 的估计是

$$\hat{b}_j(k)=\frac{B_{jk}}{\sum_{k=1}^{M}B_{jk}},\qquad j=1,2,\ldots,k=1,2,\ldots,M \tag{11.21}$$

3. 初始状态概率 π_i 的估计 $\hat{\pi}_i$ 为 S 个样本中初始状态为 q_i 的频率

Baum-Welch 算法

假设训练数据包含 S 个长度相同的观测序列 $\{(O_1, I_1), \ldots, (O_S, I_S)\}$ 而没有对应的状态序列，目标是学习隐马尔可夫模型 $\lambda = (A, B, \pi)$ 的参数。我们将观测序列数据看作观测数据 O，状态序列数据看作不可观测的隐数据 I，那么隐马尔可夫模型事实上是一个含有隐变量的概率模型

$$P(O|\lambda) = \sum_I P(O|I, \lambda)P(I|\lambda) \tag{11.22}$$

它的参数学习可以由 EM 算法实现。

1. 确定完全数据的对数似然函数
   所有观测数据写成 $O = (o_1, o_2, \ldots, o_T)$，所有隐数据写成 $I = (i_1, i_2, \ldots, i_T)$ 完全数据是 $(O, I) = (o_1, o_2, \ldots, o_T, i_1, i_2, \ldots, i_T)$。完全数据的对数似然函数是 $logP(O, I|\lambda)$
2. EM 算法的 E 步：求 Q 函数 $Q(\lambda, \bar{\lambda})$

$$E_I[logP(O, I|\lambda)|O, \bar{\lambda}] = \sum_I logP(O, I|\lambda)P(I|O, \bar{\lambda}) \tag{11.23}$$

$$Q(\lambda, \bar{\lambda}) = \sum_I logP(O, I|\lambda)P(O, I|\bar{\lambda}) \tag{11.24}$$

$P(I|O, \bar{\lambda}) = P(I, O|\bar{\lambda})/P(O|\bar{\lambda})$ 省略了对 λ 而言的常数因子。

$$P(O, I|\lambda) = \pi_{i_1} b_{i_1}(o_1) a_{i_1 i_2} \ldots a_{i_{T-1} i_T} b_{i_T}(o_T) \tag{11.25}$$

于是函数 $Q(\lambda, \bar{\lambda})$ 可以写成：

$$\begin{aligned} Q(\lambda, \bar{\lambda}) = & \sum_I log\pi_{i_1} P(O, I|\bar{\lambda}) \\ & + \sum_I \left(\sum_{t=1}^{T-1} log a_{i_t i_{t+1}} \right) P(O, I|\bar{\lambda}) \\ & + \sum_I \left(\sum_{t=1}^{T} log b_{i_t}(o_t) \right) P(O, I|\bar{\lambda}) \end{aligned} \tag{11.26}$$

3. EM 算法的 M 步：极大化 $Q(\lambda, \bar{\lambda})$ 求模型参数 A, B, π
   由于要极大化的参数在式 11.26 中单独地出现在 3 个项中，所以只需对各项分别极大化。注意到

$$\sum_{i=1}^{N} \pi_i = 1, \sum_{j=1}^{N} a_{ij} = 1, \sum_{k=1}^{M} b_j(k) = 1,$$

利用拉格朗日乘子法，求得

$$\pi_i = \frac{P(O, i_1 = i|\bar{\lambda})}{P(O|\bar{\lambda})} = \gamma_1(i) \tag{11.27}$$

$$a_{ij} = \frac{\sum\limits_{t=1}^{T-1} P(O, i_t = i, i_{t+1} = j|\bar{\lambda})}{\sum\limits_{t=1}^{T-1} P(O, i_t = i|\bar{\lambda})} = \frac{\sum\limits_{t=1}^{T-1} \xi_t(i, j)}{\sum\limits_{t=1}^{T-1} \gamma_t(i)} \tag{11.28}$$

$$b_j(k) = \frac{\sum\limits_{t=1}^{T} P(O, i_t = j|\bar{\lambda}) I(o_t = v_k)}{\sum\limits_{t=1}^{T} P(O, i_t = j|\bar{\lambda})} = \frac{\sum\limits_{t=1, o_t = v_k}^{T} \gamma_t(j)}{\sum\limits_{t=1}^{T} \gamma_t(j)} \tag{11.29}$$

问题三：预测算法

近似算法

近似算法的想法是，在每个时刻 t 选择在该时刻最有可能出现的状态 i_t^*，从而得到一个状态序列 $I^* = (i_1^*, i_2^*, \ldots, i_T^*)$，将它作为预测的结果。近似算法的优点是计算简单，其缺点是不能保证预测的状态序列整体是最有可能的状态序列，因为预测的状态序列可能有实际不发生的部分。

维特比算法

维特比算法实际是用动态规划解隐马尔可夫模型预测问题，即用动态规划 (dynamic programming) 求概率最大路径 (最优路径)。这时一条路径对应着一个状态序列。

首先导入两个变量 δ 和 ψ。定义在时刻 t 状态为 i 的所有单个路径 $i_1, i_2, \ldots, i_t$ 中概率最大值为

$$\begin{aligned} \delta_{t+1} &= \max_{i_1, i_2, \ldots, i_t} P(i_t = i, i_{t-1}, \ldots, i_1, o_{t+1}, \ldots, o_1|\lambda) \\ &= \max_{1 \leqslant j \leqslant N} [\delta_t(j) a_{ji}] b_i(o_{t+1}), \quad i = 1, 2, \ldots, N \end{aligned} \tag{11.30}$$

定义在时刻 t 状态为 i 的所有单个路径 $i_1, i_2, \ldots, i_t$ 中概率最大的路径的第 $t-1$ 个结点为

$$\psi_t(i) = arg \max_{1 \leqslant j \leqslant N} [\delta_{t-1}(j) a_{ji}], \quad i = 1, 2, \ldots, N \tag{11.31}$$

(维特比算法)

输入：模型 $\lambda = (A, B, \pi)$，观测序列 $O = (o_1, o_2, \ldots, o_T)$；

输出：最优路径 $I^* = (i_1^*, i_2^*, \ldots, i_T^*)$

1. 初始化

$$\delta_1 = \pi_i b_i(o_1),$$

$$\psi_1(i) = 0, \qquad i = 1, 2, \ldots, N$$

2. 递推。对 $t = 2, 3, \ldots, T$

$$\delta_{t+1} = \max_{1 \leqslant j \leqslant N} [\delta_t(j) a_{ji}] b_i(o_{t+1})$$

$$\psi_t(i) = arg \max_{1 \leqslant j \leqslant N} [\delta_{t-1}(j) a_{ji}], \quad i = 1, 2, \ldots, N$$

3. 终止

$$P^* = \max_{1 \leqslant i \leqslant N} \delta_T(i)$$

$$i_T^* = arg \max_{1 \leqslant i \leqslant N} \delta_T(i)$$

4. 最优路径回溯。对 $t = T - 1, T - 2, \ldots, 1$

$$i_t^* = \psi_{t+1}(i_{t+1}^*)$$

求得最佳路径 $I^* = (i_1^*, i_2^*, \ldots, i_T^*)$

11.6 条件随机场

第 12 章 近似推断

第 13 章 采样方法

第 14 章 连续潜在变量

14.1 主成分分析

主成分分析,或者称为 PCA,是一种被广泛使用的技术,应用的领域包括维度降低、有损数据压缩、特征抽取「数据可视化。它也被称为 Karhunen-Loeve 变换。

有两种经常使用的 PCA 的定义,它们会给出同样的算法。PCA 可以被定义为数据在低维线性空间上的正交投影,这个线性空间被称为主空间 (principal subspace),使得投影数据的方差被最大化。等价地,它也可以被定义为使得平均投影代价最小的线性投影。平均投影代价是指数据点和它们的投影之间的平均平方距离。

考察一组观测数据集 $\{x_n\}$,其中 $n = 1, \ldots, N$,因此 x_n 是一个 D 维欧几里德空间中的变量。我们的目标是将数据投影到维度 $M < D$ 的空间中,同时最大化投影数据的方差。

样本集合

$$X = (x_1\ x_2\ \ldots x_N)^T_{N\times p} = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & x_{22} & \ldots & x_{2p} \\ \ldots & \ldots & \ldots & \ldots \\ x_{11} & x_{12} & \ldots & x_{np} \end{pmatrix}_{N\times p} \tag{14.1}$$

样本均值

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n = \frac{1}{N}(x_1\ x_2\ \ldots\ x_n)\begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}_{N\times 1} = \frac{1}{N}X^T I_N \tag{14.2}$$

样本方差

$$\begin{aligned} S &= \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T \\ &= \frac{1}{N}(x_1 - \bar{x}\ x_2 - \bar{x}\ \ldots\ x_N - \bar{x})\begin{pmatrix} (x_1 - \bar{x})^T \\ (x_2 - \bar{x})^T \\ \vdots \\ (x_N - \bar{x})^T \end{pmatrix} \\ &= \frac{1}{N}X^T\overbrace{(I_N - \frac{1}{N}I_N I_N^T)}^{H_N-\text{中心矩阵}}\cdot(I_N - \frac{1}{N}I_N I_N^T)^T \cdot X \\ &= \frac{1}{N}X^T H X \end{aligned} \tag{14.3}$$

中心矩阵 H 有以下良好的性质

$$H = I_N - \frac{1}{N} I_N I_N^T \tag{14.4}$$

$$H^T = H \tag{14.5}$$

$$H^n = H \tag{14.6}$$

最大方差形式

首先，考虑在一维空间 ($M = 1$) 上的投影。我们可心使用 D 维向量 u_1 定义这个空间的方向。不失一般性，我们假定选择一个单位向量，从而 $u_1^T u_1 = 1$(注意，我们只对 u_1 的方向感兴趣，而对 u_1 本身的大小不感兴趣)。这样，每个数据点 x_n 被投影到一个标题值 $x_1^T x_n$ 上。投影数据的均值是 $u_1^T \bar{x}$。投影数据的方差为

$$\begin{aligned} J &= \frac{1}{N} \sum_{n=1}^{N} \{u_1^T x_n - u_1^T \bar{x}\}^2 \\ &= u_1^T \underbrace{\sum_{n=1}^{N} \frac{1}{N} (x_n - \bar{x}) \cdot (x_n - \bar{x})^T}_{S} u_1 \\ &= u_1^T S u_1 \end{aligned} \tag{14.7}$$

S 是数据的协方差矩阵，优化问题变成

$$\begin{aligned} \hat{u_1} &= arg\,max\, u_1^T S u_1 \\ s.t\ \ & u_1^T u_1 = 1 \end{aligned} \tag{14.8}$$

利用拉格朗日乘子法，求偏导并令其为零，我们看到驻点满足

$$S u_1 = \lambda_1 u_1 \tag{14.9}$$

表明 u_1 一定是 S 的一个特征向量，这个特征向量被称为第一主成分。

我们可以用一种增量的方式定义额外的主成分，方法为：在所有与那些已经考虑过的方向正交的所有可能的方向中，将新的方向选择为最大化投影方差的方向。

总结一下，主成分分析涉及到计算数据集的均值 $\bar{x}$ 和协方差矩阵 S，然后寻找 S 的对应于 M 个最大特征值的 M 个特征向量。

最小误差形式

引入 D 维基向量的一个完整的单位正交集合 $\{u_i\}$，其中 $i = 1, \ldots, D$，且满足

$$u_i^T u_j = \delta_{ij} \tag{14.10}$$

由于基是完整的，因此每个数据点可以精确地表示为基向量的一个纯性组合，即

$$x_n = \sum_{i=1}^{D} a_{ni} u_i \tag{14.11}$$

其中，系数 a_{ni} 对于不同的数据点来说是不同的。这对应于将坐标系旋转到了一个由 $\{u_i\}$ 定义的新坐标系，原始的D个分量 $\{x_{n1}, \ldots, x_{nD}\}$ 被替换为一个等价的集合 $\{a_{n1}, \ldots, a_{nD}\}$。我们有 $a_{nj} = x_n^T u_j$，因此不失一般性

$$x_n = \sum_{i=1}^{D} (x_n^T u_i) u_i \tag{14.12}$$

然而，我们的目标是使用限定数量 $M < D$ 个变量的一种表示方法来近似数据点，这对应于在低维子空间上的一个投影。不失一般性，M 维线性子空间可以用前 M 个基向量表示，因此我们可以用下式来近似每个数据点 x_n

$$\widetilde{x}_n = \sum_{i=1}^{M} z_{ni} u_i + \sum_{i=M+1}^{D} b_i u_i \tag{14.13}$$

其中 $\{z_{ni}\}$ 依赖于特定的数据点，而 $\{b_i\}$ 是常数，对于所有数据点都相同。我们可以任意选择 $\{u_i\}, \{z_{ni}\}, \{b_i\}$，从而最小化由维度降低所引入的失真。作为失真的度量，我们使用原始数据点与它的近似点 $\widetilde{x}_n$ 之间的平方距离，在数据集上取平均。因此我们的目标是最小化

$$J = \frac{1}{N} \sum_{n=1}^{N} \|x_n - \widetilde{x}_n\|^2 = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=M+1}^{D} (x_n^T u_i - \bar{x}^T u_i)^2 = \sum_{i=M+1}^{D} u_i^T S u_i \tag{14.14}$$

剩下的任务是关于 $\{u_i\}$ 对 J 进行最小化。同样利用拉格朗日乘子法能够得到

$$S u_i = \lambda_i u_i \tag{14.15}$$

PCA 的应用

14.2 概率 PCA

前一节讨论的 PCA 的形式所基于的是将数据线性投影到比原始数据空间维度更低的子空间内。PCA 也可以被视为概率潜在变量模型的最大似然解。PCA 的这种形式，被称为概率 PCA(probabilistic PCA)，与传统的 PCA 相比，会带来如下几个优势。

- 概率 PCA 表示高斯分布的一个限制形式，其中自由参数的数量可以受到限制，同时仍然使得模型能够描述数据集的主要的相关关系。
- 我们可以为 PCA 推导一个 EM 算法，这个算法在只有几个主要的特征向量需要求出的情况下，计算效率比较高，并且避免了计算数据协方差的中间步骤。

- 概率模型与 EM 的结合使得我们能够处理数据集里缺失值的问题。
- 概率 PCA 混合模型可以用一种有理有据的方式进行形式化，并且可以使用 EM 算法进行训练。
- 概率 PCA 构成了 PCA 的贝叶斯方法的基础，其中主子空间的维度可以自动从数据中找到。
- 似然函数的存在使得直接与其他的概率密度模型进行对比成为可能。相反，传统的 PCA 会给接近主子空间的数据点分配一个较低的重建代价，即使这些数据点的位置距离训练数据任意远。
- 概率 PCA 可以被用来对类条件概率密度建模，因此可以应用于分类问题。
- 概率 PCA 模型可以用一种生成式的方式运行，从而可以按照某个概率分布生成样本。

最大似然 PCA

用于 PCA 的 EM 算法

14.3 核 PCA

14.4 非线性隐变量模型

第 15 章 组合模型

15.1 贝叶斯模型平均

15.2 委员会

15.3 提升方法

提升 (boosting) 方法是一种常用的统计学习方法，应用广泛且有效。在分类问题中，它通过改变训练样本的权重，学习多个分类器，并将这些分类器进行线性组合，提高分类的性能。

提升方法基于这样一种思路：对于一个复杂任务来说，将多个专家的判断进行适当的综合所得出的判断，要比其中任何一个专家单独的判断好。提升方法就是从弱学习算法出发，反复学习，得到一系列弱分类器 (又称基本分类器)，然后组合这些弱分类器，构成一个强分类器。对提升方法来说，有两个问题需要回答：

1. 在每一轮如何改变训练数据的权值或概率分布；
2. 如果将弱分类器组合成一个强分类器。

AdaBoost 的做法是，提高那些被前一轮弱分类器错误分类样本的权值，而降低那些被正确分类样本的权值。这样一来，那些没有得到正确分类的数据，由于其权值的加大而受到后一轮的弱分类器的更大关注。于是，分类问题被一系列的弱分类器“分而治之”。至于第 2 个问题，即弱分类器的组合，AdaBoost 采取加权表决的方法。具体地，加大分类误差率小的弱分类器的权值，使其在表决中起较大作用，减小分类误差率大的弱分类器的权值，使其在表决中起较小的作用。AdaBoost 的巧妙之外就在于它将这些想法自然且有效地实现在一种算法里。

AdaBoost 算法

输入：训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$；弱学习算法；

输出：最终分类器 $G(x)$

(1) 初始化训练数据的权值分布

$$D_1 = (w_{11}, \ldots, w_{1i}, \ldots, w_{1N}), w_{1i} = \frac{1}{N}, i = 1, 2, \ldots, N \tag{15.1}$$

假设训练数据集具有均匀的权值分布，即每个训练样本在基本分类器的学习中作用相同，这一假设保证第 1 步能够在原始数据上学习基本分类器 $G_1(x)$。

(2) 对 $m = 1, 2, \ldots, M$

a. 使用具有权值分布 D_m 的训练数据集学习，得到基本分类器

$$G_m(x)X \rightarrow \{-1, +1\} \tag{15.2}$$

b. 计算 $G_m(x)$ 在训练数据集上的分类误差率

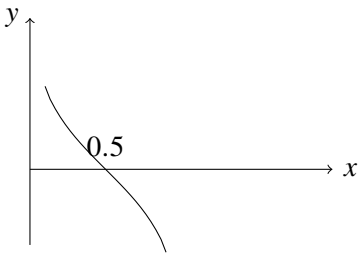
$$e_m = \sum_{i=1}^{N} P(G_m(x_i) \neq y_i) = \sum_{i=1}^{N} w_{mi} I(G_m(x_i) \neq y_i) \tag{15.3}$$

这表明，$G_m(x)$ 在加权的训练数据集上的分类误差率是被 $G_m(x)$ 误分类样本的权值之和。

c. 计算 $G_m(x)$ 的系数

$$a_m = \frac{1}{2} log \frac{1 - e_m}{e_m} \tag{15.4}$$

a_m 表示 $G_m(x)$ 在最终分类器中的重要性。



由上图可知，当 $e_m \leqslant \frac{1}{2}$ 时，$a_m \geqslant 0$，并且 a_m 随着 e_m 的减小而增大，所以**分类误差率越小的基本分类器在最终分类器中的作用越大**。

d. 更新训练数据集的权值分布

$$\begin{aligned} D_{m+1} &= (w_{m+1,1}, \ldots, m_{m+1,i}, \ldots, w_{m+1,N}) \\ w_{m+1,i} &= \frac{w_{mi}}{Z_m} exp(-a_m y_i G_m(x_i)), i = 1, 2, \ldots, N \end{aligned} \tag{15.5}$$

这里，Z_m 是规范化因子

$$Z_m = \sum_{i=1}^{N} w_{mi} exp(-a_m y_i G_m(x_i)) \tag{15.6}$$

它使 D_{m+1} 成为一个概率分布。式 15.5 可以写成

$$w_{m+1,i} = \begin{cases} \frac{w_{mi}}{Z_m} e^{-a_m}, & G_m(x_i) = y_i \\ \frac{w_{mi}}{Z_m} e^{a_m}, & G_m(x_i) \neq y_i \end{cases} \tag{15.7}$$

由此可知，**被基本分类器 $G_m(x)$ 误分类样本的权值得以扩大，而被正确分类样本的权值却得以缩小**。因此，误分类样本在下一轮学习中起更大的作用。不改变所给的训练数据，而不断改变训练数据权值的分布，使得训练数据在基本分

类器的学习中起不同的作用，这是 AdaBoost 的一个特点。

(3) 构建基本分类器的线性组合

$$f(x) = \sum_{m=1}^{M} a_m G_m(x) \tag{15.8}$$

得到最终分类器

$$G(x) = sign(f(x)) = sign\left(\sum_{m=1}^{M} a_m G_m(x)\right) \tag{15.9}$$

线性组合 $f(x)$ 实现 M 个基本分类器的加权表决。系数 a_m 表示了基本分类器 $G_m(x)$ 的重要性，这里，所有 a_m 之和并不为 1.$f(x)$ 的符号决定实例 x 的类，$f(x)$ 的绝对值表示分类的确信度。利用基本分类器的线性组合构建最终分类器是 AdaBoost 的另一个特点。

AdaBoost 算法的训练误差分析

AdaBoost 最基本的性质是它能在学习过程中不断减少训练误差，即在训练数据集上的分类误差率。

AdaBoost 算法的解释

AdaBoost 算法还有另一个解释，即可以认为 AdaBoost 算法是模型为加法模型、损失函数为指数函数、学习算法为前向分步算法时的二类分类学习方法

15.4 基于树的模型

15.5 条件混合模型

15.6 logistic 模型的混合