补充材料

刘家锋

目录

第2章　概率密度函数的参数估计

2.1　矩阵和矢量的导数

令$f(x_1, \cdots, x_d)$是一个d元标量函数，如果这d个自变量刚好是矢量$\mathbf{x} = (x_1, \cdots, x_d)^t$的元素，那么这个函数可以表示为以矢量$\mathbf{x}$为自变量的函数$f(\mathbf{x})$。多元函数$f(\mathbf{x})$可以对它的每一个自变量求偏导数$\partial f/\partial x_i$，所有这些偏导数可以表示为一个$d$维的列矢量，称为函数$f(\mathbf{x})$关于矢量$\mathbf{x}$的导数，也称为函数$f(\mathbf{x})$的梯度矢量：

$$\nabla f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \\ \vdots \\ \partial f/\partial x_d \end{pmatrix} \tag{2.1}$$

同样，如果nm元函数$f(w_{11}, \cdots, w_{1m}, w_{21}, \cdots w_{nm})$的自变量是$n \times m$矩阵$W$的元素，那么也可以表示为以矩阵$W$为自变量的函数$f(W)$，函数关于矩阵$W$的导数可以表示为一个$n \times m$的矩阵：

$$\frac{\partial f(W)}{\partial W} = \begin{pmatrix} \partial f/\partial w_{11} & \partial f/\partial w_{12} & \cdots & \partial f/\partial w_{1m} \\ \partial f/\partial w_{21} & \partial f/\partial w_{22} & \cdots & \partial f/\partial w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \partial f/\partial w_{n1} & \partial f/\partial w_{n2} & \cdots & \partial f/\partial w_{nm} \end{pmatrix} \tag{2.2}$$

再来看更复杂一些的情况，令$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}, \cdots, f_n(\mathbf{x}))^t$是一个定义在矢量$\mathbf{x}$上的矢量函数，也就是说$\mathbf{f}(\mathbf{x})$的每一维元素都是以矢量$\mathbf{x}$为自变量的标量函数。$\mathbf{f}(\mathbf{x})$通常也称为由$d$维空间到$n$维空间的映射，记作$\mathbf{f}(\mathbf{x}) : R^d \to R^n$ 。$\mathbf{f}(\mathbf{x})$关于自变量$\mathbf{x}$的导数是一个$n \times d$的矩阵，称为雅克比（Jacobi）

矩阵：

$$J(\mathbf{x}) = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \partial f_1/\partial x_1 & \partial f_1/\partial x_2 & \cdots & \partial f_1/\partial x_d \\ \partial f_2/\partial x_1 & \partial f_2/\partial x_2 & \cdots & \partial f_2/\partial x_d \\ \vdots & \vdots & \ddots & \vdots \\ \partial f_n/\partial x_1 & \partial f_n/\partial x_2 & \cdots & \partial f_n/\partial x_d \end{pmatrix} \tag{2.3}$$

观察公式(2.1)我们会发现，标量函数$f(\mathbf{x})$关于$\mathbf{x}$的梯度$\nabla f(\mathbf{x})$就是一个定义在矢量$\mathbf{x}$上的d维矢量函数，梯度矢量关于$\mathbf{x}$的导数是一个$d \times d$的矩阵，一般称为Hessian矩阵：

$$H(\mathbf{x}) = \frac{\partial \nabla f(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \partial^2 f/\partial x_1^2 & \partial^2 f/\partial x_1 \partial x_2 & \cdots & \partial^2 f/\partial x_1 \partial x_d \\ \partial^2 f/\partial x_2 \partial x_1 & \partial^2 f/\partial x_2^2 & \cdots & \partial^2 f/\partial x_2 \partial x_d \\ \vdots & \vdots & \ddots & \vdots \\ \partial^2 f/\partial x_d \partial x_1 & \partial^2 f/\partial x_d \partial x_2 & \cdots & \partial^2 f/\partial x_d^2 \end{pmatrix} \tag{2.4}$$

因为$\partial^2 f/\partial x_i \partial x_j = \partial^2 f/\partial x_j \partial x_i$，所以海森矩阵$H(\mathbf{x})$是对称矩阵。

下面给出一些常用的关于矢量和矩阵导数的公式，其中a为标量，$\mathbf{x}$和$\mathbf{y}$为d维矢量，A和W为$d \times d$的矩阵，$\mathbf{1}$为元素均为1的d维矢量：

$$f(\mathbf{x}) = \mathbf{x}^t\mathbf{y} = \mathbf{y}^t\mathbf{x}: \qquad \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{y} \tag{2.5}$$

$$f(\mathbf{x}) = \mathbf{x}^t A\mathbf{x}: \qquad \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \left(A + A^t\right)\mathbf{x} \tag{2.6}$$

$$f(W) = \mathbf{x}^t W\mathbf{y}: \qquad \frac{\partial f(W)}{\partial W} = \mathbf{x}\mathbf{y}^t \tag{2.7}$$

$$f(\mathbf{x}) = g(\mathbf{x})h(\mathbf{x}): \qquad \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}}h(\mathbf{x}) + \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}g(\mathbf{x}) \tag{2.8}$$

$$f(\mathbf{x}) = g(u), u = h(\mathbf{x}): \qquad \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = g'(h(\mathbf{x}))\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} \tag{2.9}$$

2.2　高斯分布参数的极大似然估计

样本集合$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$独立抽样自均值为$\boldsymbol{\mu}$，协方差矩阵为$\Sigma$的高斯分布。建立对数自然函数：

$$l(\boldsymbol{\mu}, \Sigma) = \sum_{i=1}^{n} \ln p(\mathbf{x}_i|\boldsymbol{\mu}, \Sigma) \tag{2.10}$$

其中：

$$\ln p(\mathbf{x}_i|\boldsymbol{\mu},\Sigma) = \ln\left[\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu})^t\Sigma^{-1}(\mathbf{x}_i-\boldsymbol{\mu})\right)\right] \tag{2.11}$$

$$= -\frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma| - \frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu})^t\Sigma^{-1}(\mathbf{x}_i-\boldsymbol{\mu}) \tag{2.12}$$

因此，对数似然函数为：

$$l(\boldsymbol{\mu},\Sigma) = \sum_{i=1}^{n}\left[-\frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma| - \frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu})^t\Sigma^{-1}(\mathbf{x}_i-\boldsymbol{\mu})\right] \tag{2.13}$$

首先来推导均值矢量$\boldsymbol{\mu}$的最大似然估计，对数似然函数对均值矢量$\boldsymbol{\mu}$求偏导数及极值：

$$\frac{\partial l(\boldsymbol{\mu},\Sigma)}{\partial\boldsymbol{\mu}} = \sum_{i=1}^{n}\Sigma^{-1}(\mathbf{x}_i-\boldsymbol{\mu}) = \Sigma^{-1}\left[\sum_{i=1}^{n}(\mathbf{x}_i-\boldsymbol{\mu})\right] = \mathbf{0} \tag{2.14}$$

这里利用了协方差矩阵为对称矩阵的事实，上式两边左乘Σ：

$$\sum_{i=1}^{n}(\mathbf{x}_i-\boldsymbol{\mu}) = \sum_{i=1}^{n}\mathbf{x}_i - n\boldsymbol{\mu} = \mathbf{0} \tag{2.15}$$

这样就得到了均值矢量$\boldsymbol{\mu}$的最大似然估计：

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \tag{2.16}$$

协方差矩阵Σ的最大似然估计推导要复杂一些，首先给出需要用到的几个关于$d\times d$维方阵A的基本性质：

1. 逆矩阵的行列式值等于行列式值的倒数：

$$\left|A^{-1}\right| = \frac{1}{|A|} \tag{2.17}$$

2. 令$f(A)=|A|$，则矩阵A的行列式值对矩阵的导数：

$$\frac{\partial f(A)}{\partial A} = \frac{\partial\left(|A|\right)}{\partial A} = |A|A^{-1} \tag{2.18}$$

3. 令$g(A)=\mathbf{x}^tA\mathbf{y}$，$\mathbf{x}$和$\mathbf{y}$为$d$维列矢量，函数$g(A)$对矩阵$A$的导数：

$$\frac{\partial g(A)}{\partial A} = \mathbf{x}\mathbf{y}^t \tag{2.19}$$

根据性质1，公式(2.13)的对数似然函数可以写成：

$$l(\boldsymbol{\mu},\Sigma)=\sum_{i=1}^{n}\left[-\frac{d}{2}\ln 2\pi+\frac{1}{2}\ln|\Sigma^{-1}|-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu})^t\Sigma^{-1}(\mathbf{x}_i-\boldsymbol{\mu})\right] \tag{2.20}$$

$l(\boldsymbol{\mu},\Sigma)$对协方差矩阵的逆阵$\Sigma^{-1}$求偏导数及极值：

$$\frac{\partial l(\boldsymbol{\mu})}{\partial\Sigma^{-1}}=\sum_{i=1}^{n}\left\{\frac{1}{2}\frac{\partial\left(\ln|\Sigma^{-1}|\right)}{\partial\Sigma^{-1}}-\frac{1}{2}\frac{\partial}{\partial\Sigma^{-1}}\left[(\mathbf{x}_i-\boldsymbol{\mu})^t\Sigma^{-1}(\mathbf{x}_i-\boldsymbol{\mu})\right]\right\} \tag{2.21}$$

$$=\sum_{i=1}^{n}\left[\frac{1}{2}\frac{1}{|\Sigma^{-1}|}\frac{\partial|\Sigma^{-1}|}{\partial\Sigma^{-1}}-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu})(\mathbf{x}_i-\boldsymbol{\mu})^t\right] \tag{2.22}$$

$$=\sum_{i=1}^{n}\left[\frac{1}{2}\frac{1}{|\Sigma^{-1}|}|\Sigma^{-1}|\Sigma-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu})(\mathbf{x}_i-\boldsymbol{\mu})^t\right] \tag{2.23}$$

$$=\frac{1}{2}\sum_{i=1}^{n}\left[\Sigma-(\mathbf{x}_i-\boldsymbol{\mu})(\mathbf{x}_i-\boldsymbol{\mu})^t\right] \tag{2.24}$$

$$=\frac{n}{2}\Sigma-\frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i-\boldsymbol{\mu})(\mathbf{x}_i-\boldsymbol{\mu})^t \tag{2.25}$$

$$=0 \tag{2.26}$$

其中第2行和第3行分别用到了性质3和性质2，由此可以得到：

$$\Sigma=\frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i-\boldsymbol{\mu})(\mathbf{x}_i-\boldsymbol{\mu})^t$$

2.3　高斯混合模型EM 算法的迭代公式推导

我们首先来推导一般混合密度模型参数估计的EM算法迭代公式，然后再将一般的混合密度模型具体化为高斯混合模型。

2.3.1　混合密度模型

假设样本集$X=\{\mathbf{x}_1,\cdots,\mathbf{x}_n\}$中的样本相互独立，并且按照如下的过程产生：

1. 样本是依据概率由M个分布中的一个产生的，分布的概率密度函数为$p(\mathbf{x}|\boldsymbol{\theta}_k)$，$k=1,\cdots,M$，$\boldsymbol{\theta}_k$为分布的参数；
2. 由第k个分布产生样本的先验概率为α_k；
3. 先验概率$\boldsymbol{\alpha}=(\alpha_1,\cdots,\alpha_M)^t$，以及分布的参数$\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_M$均未知。

我们称样本集X来自于一个“混合密度模型”，混合密度模型的概率密度函数为：

$$p(\mathbf{x}|\boldsymbol{\Theta}) = \sum_{k=1}^{M} \alpha_k p(\mathbf{x}|\boldsymbol{\theta}_k) \tag{2.27}$$

其中$\boldsymbol{\Theta} = (\alpha, \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_M)$为模型的参数，每个$p(\mathbf{x}|\boldsymbol{\theta}_k)$称为一个分量密度。

2.3.2　混合密度模型参数估计的EM迭代公式

混合密度模型的参数估计中，由于样本是由哪个分量密度所产生的信息$Y = \{y_1, \cdots, y_n\}$是未知的，因此需要将其视作“丢失”信息，使用EM算法进行估计。EM算法中E步和M步的迭代公式：

$$\text{E步：} Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}^g) = E_Y \left[\ln p(X, Y|\boldsymbol{\Theta})|X, \boldsymbol{\Theta}^g\right] \tag{2.28}$$

$$\text{M步：} \boldsymbol{\Theta}^* = \arg\max_{\boldsymbol{\Theta}} Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}^g) \tag{2.29}$$

其中$\boldsymbol{\Theta}^g$是对参数$\boldsymbol{\Theta}$的一个猜测值。E步计算的是在已知样本集X和参数猜测值$\boldsymbol{\Theta}^g$条件下的期望对数似然函数；而M步则是对$Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}^g)$的优化。更新参数的猜测值设置：$\boldsymbol{\Theta}^g = \boldsymbol{\Theta}^*$，进入下一轮迭代。

E步期望对数似然函数$Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}^g)$的推导：

训练样本$\mathbf{x}_i$是由第y_i个分量密度函数产生的，$y_i = 1, \cdots, M$，这两个随机事件的联合概率密度：

$$p(\mathbf{x}_i, y_i|\boldsymbol{\Theta}) = \alpha_{y_i} p(\mathbf{x}_i|\boldsymbol{\theta}_{y_i}) \tag{2.30}$$

因此，关于完整数据集$D = \{X, Y\}$的对数似然函数为：

$$l(\boldsymbol{\Theta} = \ln p(X, Y|\boldsymbol{\Theta}) = \sum_{i=1}^{n} \ln \left[\alpha_{y_i} p(\mathbf{x}_i|\boldsymbol{\theta}_{y_i})\right] \tag{2.31}$$

另外根据贝叶斯公式，在已知参数$\boldsymbol{\Theta}^g = (\alpha_1^g, \cdots, \alpha_M^g, \boldsymbol{\theta}_1^g, \cdots, \boldsymbol{\theta}_M^g)$ 和样本$\mathbf{x}_i$的条件下，$\mathbf{x}_i$由第y_i个分量产生的概率为：

$$P(y_i|\mathbf{x}_i, \boldsymbol{\Theta}^g) = \frac{p(\mathbf{x}_i, y_i|\boldsymbol{\Theta}^g)}{p(\mathbf{x}_i|\boldsymbol{\Theta}^g)} = \frac{\alpha_{y_i}^g p(\mathbf{x}_i|\boldsymbol{\theta}_{y_i}^g)}{\sum_{k=1}^{M} \alpha_k^g p(\mathbf{x}_i|\boldsymbol{\theta}_k^g)} \tag{2.32}$$

考虑到样本的独立同分布性，y_i只与$\mathbf{x}_i$有关，独立于其它的$\mathbf{x}_j$和y_j，$j \neq i$，因此：

$$P(Y|X, \boldsymbol{\Theta}^g) = P(y_1, \cdots, y_n|\mathbf{x}_1, \cdots, \mathbf{x}_n, \boldsymbol{\Theta}^g) \tag{2.33}$$

$$= \prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \boldsymbol{\Theta}^g) \tag{2.34}$$

将公式(2.31)、(2.34)代入到公式(2.28)E步的期望对数似然函数，同时考虑到每一个y_i是离散的，只取$1,\cdots,M$中的某一个值，对Y 的数学期望可以由如下的求和式计算：

$$Q(\boldsymbol{\Theta};\boldsymbol{\Theta}^g)=\sum_{y_1=1}^{M}\cdots\sum_{y_n=1}^{M}\ln p(X,Y|\boldsymbol{\Theta})P(Y|X,\boldsymbol{\Theta}^g) \tag{2.35}$$

$$=\sum_{y_1=1}^{M}\cdots\sum_{y_n=1}^{M}\left\{\sum_{i=1}^{n}\ln\left[\alpha_{y_i}p(\mathbf{x}_i|\boldsymbol{\theta}_{y_i})\right]\right\}\prod_{i=1}^{n}P(y_i|\mathbf{x}_i,\boldsymbol{\Theta}^g) \tag{2.36}$$

$$=\sum_{y_1=1}^{M}\cdots\sum_{y_n=1}^{M}\sum_{i=1}^{n}\sum_{l=1}^{M}\left\{\delta_{l,y_i}\ln\left[\alpha_{y_i}p(\mathbf{x}_i|\boldsymbol{\theta}_{y_i})\right]\prod_{j=1}^{n}P(y_j|\mathbf{x}_j,\boldsymbol{\Theta}^g)\right\} \tag{2.37}$$

$$=\sum_{i=1}^{n}\sum_{l=1}^{M}\left\{\ln\left[\alpha_l p(\mathbf{x}_i|\boldsymbol{\theta}_l)\right]\left\{\sum_{y_1=1}^{M}\cdots\sum_{y_n=1}^{M}\left[\delta_{l,y_i}\prod_{j=1}^{n}P(y_j|\mathbf{x}_j,\boldsymbol{\Theta}^g)\right]\right\}\right\} \tag{2.38}$$

其中：

$$\delta_{l,y_i}=\begin{cases}1, & l=y_i\\ 0, & l\neq y_i\end{cases}$$

由于$\sum_{j=1}^{M}P(y_j|\mathbf{x}_j,\boldsymbol{\Theta}^g)=1$，因此公式(2.38)内层大括号中的内容可以简化为：

$$\sum_{y_1=1}^{M}\cdots\sum_{y_n=1}^{M}\left[\delta_{l,y_i}\prod_{j=1}^{n}P(y_j|\mathbf{x}_j,\boldsymbol{\Theta}^g)\right]=\left[\sum_{y_1=1}^{M}\cdots\sum_{y_n=1}^{M}\prod_{\substack{j=1\\j\neq i}}^{n}P(y_j|\mathbf{x}_j,\boldsymbol{\Theta}^g)\right]p(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)$$

$$=\prod_{\substack{j=1\\j\neq i}}^{n}\left(\sum_{y_j=1}^{n}P(y_j|\mathbf{x}_j,\boldsymbol{\Theta}^g)\right)p(l|\mathbf{x}_i,\boldsymbol{\Theta}^g) \tag{2.39}$$

$$=P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g) \tag{2.40}$$

公式(2.39)使用了乘法的分配率，代入公式(2.38)可得：

$$Q(\boldsymbol{\Theta};\boldsymbol{\Theta}^g)=\sum_{i=1}^{n}\sum_{l=1}^{M}\ln\left[\alpha_l p(\mathbf{x}_i|\boldsymbol{\theta}_l)\right]P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g) \tag{2.41}$$

$$=\sum_{i=1}^{n}\sum_{l=1}^{M}\ln\alpha_l P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)+\sum_{i=1}^{n}\sum_{l=1}^{M}\ln p(\mathbf{x}_i|\boldsymbol{\theta}_l)P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g) \tag{2.42}$$

上式中的期望对数似然函数$Q(\boldsymbol{\Theta};\boldsymbol{\Theta}^g)$只是参数$\boldsymbol{\Theta}$的函数，$\mathbf{x}_1,\cdots,\mathbf{x}_n$以及$\boldsymbol{\Theta}^g$均为已知。

M步期望对数似然函数$Q(\boldsymbol{\Theta};\boldsymbol{\Theta}^g)$的优化：

下面来求解公式(2.29)M 步的优化问题，需要注意的是参数$\boldsymbol{\alpha} = (\alpha_1,\cdots,\alpha_M)^t$存在约束$\sum_{k=1}^{M}\alpha_k = 1$，因此构造Lagrange 函数：

$$L(\boldsymbol{\Theta},\lambda) = Q(\boldsymbol{\Theta};\boldsymbol{\Theta}^g) + \lambda\left(\sum_{k=1}^{M}\alpha_k - 1\right) \tag{2.43}$$

$$= \sum_{i=1}^{n}\sum_{l=1}^{M}[\ln\alpha_l P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)] + \sum_{i=1}^{n}\sum_{l=1}^{M}[\ln p(\mathbf{x}_i|\boldsymbol{\theta}_l)P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)] + \lambda\left(\sum_{k=1}^{M}\alpha_k - 1\right) \tag{2.44}$$

Lagrange函数对α_l求偏导数及极值：

$$\frac{\partial L(\boldsymbol{\Theta},\lambda)}{\partial\alpha_l} = \sum_{i=1}^{n}\frac{1}{\alpha_i}P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g) + \lambda = 0 \tag{2.45}$$

因此有：

$$\alpha_i\lambda + \sum_{i=1}^{n}P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g) = 0 \tag{2.46}$$

等式对l求和：

$$\sum_{l=1}^{M}\left[\alpha_i\lambda + \sum_{i=1}^{n}P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)\right] = \lambda\sum_{l=1}^{M}\alpha_l + \sum_{i=1}^{n}\sum_{l=1}^{M}P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g) = \lambda + n = 0 \tag{2.47}$$

因此Lagrange系数$\lambda = n$，代入公式(2.46)式得到关于混合密度组合系数α_l的估计公式：

$$\alpha_l = \frac{1}{n}\sum_{i=1}^{n}P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g) \tag{2.48}$$

其中$P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)$可以由公式(2.32)计算。

2.3.3　高斯混合模型参数估计的EM迭代公式

对于每一个分量密度函数参数的估计，需要考虑具体的分量密度函数形式，下面推导高斯混合模型中分量高斯的均值矢量$\boldsymbol{\mu}_l$和协方差矩阵Σ_l的估计公式。

高斯混合模型中，第l个分量密度函数是一个高斯函数：

$$p_l(\mathbf{x}|\boldsymbol{\theta}_l) = \frac{1}{(2\pi)^{d/2}|\Sigma_l|^{1/2}}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_l)^t\Sigma_l^{-1}(\mathbf{x}-\boldsymbol{\mu}_l)\right] \tag{2.49}$$

考虑到公式(2.44)Lagrange函数中第1项和第3 项与均值矢量$\boldsymbol{\mu}_l$和协方差矩阵Σ_l无关，在优化时不起作用，为了书写简单可以将其省略。将高斯函数代入公式(2.44)：

$$L(\boldsymbol{\Theta},\lambda)=\sum_{l=1}^{M}\sum_{i=1}^{n}[\ln p(\mathbf{x}_i|\boldsymbol{\theta}_l)P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)] \tag{2.50}$$

$$=\sum_{l=1}^{M}\sum_{i=1}^{n}\left[\left(-\frac{d}{2}\ln 2\pi-\frac{1}{2}\ln|\Sigma_l|-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_l)^t\Sigma_l^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_l)\right)P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)\right] \tag{2.51}$$

首先对$\boldsymbol{\mu}_l$求偏导数及极值：

$$\frac{\partial L(\boldsymbol{\Theta},\lambda)}{\partial\boldsymbol{\mu}_l}=\sum_{i=1}^{n}\left[\Sigma_l^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_l)P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)\right] \tag{2.52}$$

$$=\Sigma_l^{-1}\left[\sum_{i=1}^{n}\mathbf{x}_iP(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)-\boldsymbol{\mu}_l\sum_{i=1}^{n}P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)\right] \tag{2.53}$$

$$=\mathbf{0} \tag{2.54}$$

两边左乘Σ_l，可以得到均值矢量$\boldsymbol{\mu}_l$的估计公式：

$$\boldsymbol{\mu}_l=\frac{\sum_{i=1}^{n}\mathbf{x}_iP(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)}{\sum_{i=1}^{n}P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)} \tag{2.55}$$

公式(2.51)对Σ^{-1}求偏导数及极值（具体过程参见高斯分布参数最大似然估计的推导过程）：

$$\frac{\partial L(\boldsymbol{\Theta},\lambda)}{\partial\Sigma_l^{-1}}=\sum_{i=1}^{n}\left[\left(\frac{1}{2}\frac{1}{|\Sigma_l^{-1}|}|\Sigma_l^{-1}|\Sigma_l-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_l)(\mathbf{x}_i-\boldsymbol{\mu}_l)^t\right)P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)\right] \tag{2.56}$$

$$=\frac{1}{2}\left\{\Sigma_l\left[\sum_{i=1}^{n}P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)\right]-\sum_{i=1}^{n}P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)(\mathbf{x}_i-\boldsymbol{\mu}_l)(\mathbf{x}_i-\boldsymbol{\mu}_l)^t\right\} \tag{2.57}$$

$$=0 \tag{2.58}$$

由此得到协方差矩阵Σ_l的估计公式：

$$\Sigma_l=\frac{\sum_{i=1}^{n}P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)(\mathbf{x}_i-\boldsymbol{\mu}_l)(\mathbf{x}_i-\boldsymbol{\mu}_l)^t}{\sum_{i=1}^{n}P(l|\mathbf{x}_i,\boldsymbol{\Theta}^g)} \tag{2.59}$$

总结公式(2.32)、(2.48)、(2.55)和(2.59)，得到高斯混合模型参数估计EM算法第j轮迭代公式：

$$P(l|\mathbf{x}_i,\boldsymbol{\Theta}^{j-1})=\frac{\alpha_l^{j-1}p(\mathbf{x}_i|\boldsymbol{\theta}_l^{j-1})}{\sum_{k=1}^{M}\alpha_k^{j-1}p(\mathbf{x}_i|\boldsymbol{\theta}_k^{j-1})} \tag{2.60}$$

其中$p(\mathbf{x}_i|\boldsymbol{\theta}_l^{j-1})$为高斯函数；

$$\alpha_l^j = \frac{1}{n}\sum_{i=1}^{n} P(l|\mathbf{x}_i, \boldsymbol{\Theta}^{j-1}) \tag{2.61}$$

$$\boldsymbol{\mu}_l^j = \frac{\sum_{i=1}^{n} \mathbf{x}_i P(l|\mathbf{x}_i, \boldsymbol{\Theta}^{j-1})}{\sum_{i=1}^{n} P(l|\mathbf{x}_i, \boldsymbol{\Theta}^{j-1})} \tag{2.62}$$

$$\Sigma_l^j = \frac{\sum_{i=1}^{n} P(l|\mathbf{x}_i, \boldsymbol{\Theta}^{j-1})(\mathbf{x}_i - \boldsymbol{\mu}_l^j)(\mathbf{x}_i - \boldsymbol{\mu}_l^j)^t}{\sum_{i=1}^{n} P(l|\mathbf{x}_i, \boldsymbol{\Theta}^{j-1})} \tag{2.63}$$

2.4　103 页，例2 的详细推导过程

二维空间中4个样本，其中的一个样本丢失1个特征：

$$D = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\} = \left\{ \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} * \\ 4 \end{pmatrix} \right\} \tag{2.64}$$

假设样本满足正态分布，协方差矩阵为对角阵：

$$\boldsymbol{\theta} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \sigma_1 \\ \sigma_2 \end{pmatrix}, \qquad \text{初始参数：} \boldsymbol{\theta}_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \tag{2.65}$$

EM算法估计正态分布的参数

E步：计算已知参数$\boldsymbol{\theta}_0$的条件下，参数$\boldsymbol{\theta}$的对数似然函数

$$Q(\boldsymbol{\theta}; \boldsymbol{\theta}_0) = E_{x_{41}}\left[\ln p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4|\boldsymbol{\theta})|\boldsymbol{\theta}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, x_{42}\right] \tag{2.66}$$

$$= \int_{-\infty}^{+\infty} \ln p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4|\boldsymbol{\theta}) p(x_{41}|\boldsymbol{\theta}_0, x_{42}) dx_{41} \tag{2.67}$$

其中：

$$\ln p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4|\boldsymbol{\theta}) = \sum_{i=1}^{3} \ln p(\mathbf{x}_i|\boldsymbol{\theta}) + \ln p(x_{41}, x_{42}|\boldsymbol{\theta}) \tag{2.68}$$

$$p(x_{41}|\boldsymbol{\theta}_0, x_{42}) = \frac{p(x_{41}, x_{42}|\boldsymbol{\theta}_0)}{p(x_{42}|\boldsymbol{\theta}_0)} + \frac{p(x_{41}, x_{42}|\boldsymbol{\theta}_0)}{\int_{-\infty}^{+\infty} p(x_{41}, x_{42}|\boldsymbol{\theta}_0) dx_{41}} \tag{2.69}$$

$$= \frac{1}{K} p(x_{41}, x_{42}|\boldsymbol{\theta}_0) \tag{2.70}$$

K为归一化因子，可以计算得到：

$$K = \int_{-\infty}^{+\infty} p(x_{41}, x_{42}|\boldsymbol{\theta}_0)dx_{41} \tag{2.71}$$

$$= \int_{-\infty}^{+\infty} \frac{1}{2\pi\sigma_{10}\sigma_{20}} \exp\left[-\frac{(x_{41}-\mu_{10})^2}{2\sigma_{10}^2} - \frac{(x_{42}-\mu_{20})^2}{2\sigma_{20}^2}\right] dx_{41} \tag{2.72}$$

$$= \frac{1}{\sqrt{2\pi}\sigma_{20}} \exp\left[-\frac{(x_{42}-\mu_{20})^2}{2\sigma_{20}^2}\right] \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma_{10}} \exp\left[-\frac{(x_{41}-\mu_{10})^2}{2\sigma_{10}^2}\right] dx_{41} \tag{2.73}$$

$$= \frac{1}{\sqrt{2\pi}\sigma_{20}} \exp\left[-\frac{(x_{42}-\mu_{20})^2}{2\sigma_{20}^2}\right] = \frac{e^{-8}}{\sqrt{2\pi}} \tag{2.74}$$

将公式(2.68)和(2.70)带入(2.67)：

$$\begin{aligned} Q(\boldsymbol{\theta};\boldsymbol{\theta}_0) = &\int_{-\infty}^{+\infty} \left[\sum_{i=1}^{3} \ln p(\mathbf{x}_i|\boldsymbol{\theta})\right] p(x_{41}|\boldsymbol{\theta}_0, x_{42})dx_{41} \\ &+ \int_{-\infty}^{+\infty} \ln p(x_{41}, x_{42}|\boldsymbol{\theta}) p(x_{41}|x_{42}, \boldsymbol{\theta}_0)dx_{41} \end{aligned} \tag{2.75}$$

$$= \sum_{i=1}^{3} \ln p(\mathbf{x}_i|\boldsymbol{\theta}) + \frac{1}{K} \int_{-\infty}^{+\infty} \ln p(x_{41}, x_{42}|\boldsymbol{\theta}) p(x_{41}|x_{42}, \boldsymbol{\theta}_0)dx_{41} \tag{2.76}$$

其中：

$$p(x_{41}|x_{42}, \boldsymbol{\theta}_0) = \ln\left\{\frac{1}{2\pi\sigma_1\sigma_2} \exp\left[-\frac{(x_{41}-\mu_1)^2}{2\sigma_1^2} - \frac{(x_{42}-\mu_2)^2}{2\sigma_2^2}\right]\right\} \tag{2.77}$$

$$= -\ln 2\pi\sigma_1\sigma_2 - \frac{(x_{41}-\mu_1)^2}{2\sigma_1^2} - \frac{(x_{42}-\mu_2)^2}{2\sigma_2^2} \tag{2.78}$$

将公式(2.78)代入(2.76)

$$\begin{aligned} Q(\boldsymbol{\theta};\boldsymbol{\theta}_0) = &\sum_{i=1}^{3} \ln p(\mathbf{x}_i|\boldsymbol{\theta}) - \frac{1}{K} \int_{-\infty}^{+\infty} \frac{(x_{41}-\mu_1)^2}{2\sigma_1^2} p(x_{41}|x_{42}, \boldsymbol{\theta}_0)dx_{41} \\ &- \frac{1}{K} \int_{-\infty}^{+\infty} \left[\ln(2\pi\sigma_1\sigma_2) + \frac{(x_{42}-\mu_2)^2}{2\sigma_2^2}\right] p(x_{41}|x_{42}, \boldsymbol{\theta}_0)dx_{41} \end{aligned} \tag{2.79}$$

$$\begin{aligned} = &\sum_{i=1}^{3} \ln p(\mathbf{x}_i|\boldsymbol{\theta}) - \ln(2\pi\sigma_1\sigma_2) - \frac{(x_{42}-\mu_2)^2}{2\sigma_2^2} \\ &- \frac{1}{K} \int_{-\infty}^{+\infty} \frac{(x_{41}-\mu_1)^2}{2\sigma_1^2} p(x_{41}|x_{42}, \boldsymbol{\theta}_0)dx_{41} \end{aligned} \tag{2.80}$$

计算积分：

$$\begin{aligned}
&\int_{-\infty}^{+\infty} \frac{(x_{41}-\mu_1)^2}{2\sigma_1^2} p(x_{41}|x_{42},\boldsymbol{\theta}_0)dx_{41} \\
&\quad = \frac{1}{2\sigma_1^2 \times 2\pi\sigma_{10}\sigma_{20}} \exp\left[-\frac{(x_{42}-\mu_2 0)^2}{2\sigma_{20}^2}\right] \\
&\qquad \times \int_{-\infty}^{+\infty} (x_{41}-\mu_1)^2 \exp\left[-\frac{(x_{41}-\mu_{10})^2}{2\sigma_{10}^2}\right] dx_{41} \qquad (2.81) \\
&\quad = \frac{e^{-8}}{4\pi\sigma_1^2} \int_{-\infty}^{+\infty} (x_{41}^2 - 2\mu_1 x_{41} + \mu_1^2) e^{-\frac{x_{41}^2}{2}} dx_{41} \qquad (2.82)
\end{aligned}$$

利用积分公式：

$$\int_{-\infty}^{+\infty} x_{41}^2 e^{-\frac{x_{41}^2}{2}} dx_{41} = \sqrt{2\pi} \qquad (2.83)$$

$$\int_{-\infty}^{+\infty} -2\mu_1 x_{41} e^{-\frac{x_{41}^2}{2}} dx_{41} = 0 \qquad (2.84)$$

$$\int_{-\infty}^{+\infty} \mu_1^2 e^{-\frac{x_{41}^2}{2}} dx_{41} = \mu_1^2\sqrt{2\pi} \qquad (2.85)$$

得到：

$$\begin{aligned}
&\frac{1}{K}\int_{-\infty}^{+\infty} \frac{(x_{41}-\mu_1)^2}{2\sigma_1^2} p(x_{41}|x_{42},\boldsymbol{\theta}_0)dx_{41} \\
&\quad = \frac{1}{K} \times \frac{e^{-8}}{4\pi\sigma_1^2} \times (\sqrt{2\pi} + \mu_1^2\sqrt{2\pi}) \qquad (2.86) \\
&\quad = \frac{\sqrt{2\pi}}{e^{-8}} \times \frac{\sqrt{2\pi}e^{-8}}{4\pi\sigma_1^2} \times (1+\mu_1^2) \qquad (2.87) \\
&\quad = \frac{1}{2\sigma_1^2}(1+\mu_1^2) \qquad (2.88)
\end{aligned}$$

将(2.88)代入(2.80)：

$$\begin{aligned}
Q(\boldsymbol{\theta};\boldsymbol{\theta}_0) &= \sum_{i=1}^{3} \ln p(\mathbf{x}_i|\boldsymbol{\theta}) - \ln(2\pi\sigma_1\sigma_2) - \frac{(x_{42}-\mu_2)^2}{2\sigma_2^2} - \frac{1+\mu_1^2}{2\sigma_1^2} \qquad (2.89) \\
&= \sum_{i=1}^{3} \left[-\ln(2\pi\sigma_1\sigma_2) - \frac{(x_{i1}-\mu_1)^2}{2\sigma_1^2} - \frac{(x_{i2}-\mu_1)^2}{2\sigma_2^2}\right] \\
&\quad - \ln(2\pi\sigma_1\sigma_2) - \frac{(x_{42}-\mu_2)^2}{2\sigma_2^2} - \frac{1+\mu_1^2}{2\sigma_1^2} \qquad (2.90)
\end{aligned}$$

M步：计算$Q(\boldsymbol{\theta};\boldsymbol{\theta}_0)$关于参数$\boldsymbol{\theta}$的极值

关于μ_1的极值：

$$\frac{\partial Q}{\partial \mu_1} = \sum_{i=1}^{3} \frac{(x_{i1} - \mu_1)}{\sigma_1^2} - \frac{\mu_1}{\sigma_1^2} = 0 \tag{2.91}$$

得到：

$$\mu_1 = \frac{1}{4} \sum_{i=1}^{3} x_{i1} = \frac{3}{4} \tag{2.92}$$

关于μ_2的极值：

$$\frac{\partial Q}{\partial \mu_2} = \sum_{i=1}^{3} \frac{(x_{i2} - \mu_2)}{\sigma_2^2} + \frac{(x_{42} - \mu_2)}{\sigma_2^2} = 0 \tag{2.93}$$

得到：

$$\mu_2 = \frac{1}{4} \left(\sum_{i=1}^{3} x_{i2} + x_{42} \right) = \frac{8}{4} \tag{2.94}$$

关于σ_1的极值：

$$\frac{\partial Q}{\partial \sigma_1} = \sum_{i=1}^{3} \left[-\frac{1}{\sigma_1} + \frac{(x_{i1} - \mu_1)^2}{\sigma_1^3} \right] - \frac{1}{\sigma_1} + \frac{(1 + \mu_1^2)}{\sigma_1^3} = 0 \tag{2.95}$$

得到：

$$\sigma_1^2 = \frac{1}{4} \left(\sum_{i=1}^{3} (x_{i1} - \mu_1)^2 + (1 + \mu_1^2) \right) = \frac{3.75}{4} \tag{2.96}$$

关于σ_2的极值：

$$\frac{\partial Q}{\partial \sigma_2} = \sum_{i=1}^{3} \left[-\frac{1}{\sigma_2} + \frac{(x_{i2} - \mu_2)^2}{\sigma_2^3} \right] - \frac{1}{\sigma_2} + \frac{(x_{42} - \mu_2)^2}{\sigma_2^3} = 0 \tag{2.97}$$

得到：

$$\sigma_2^2 = \frac{1}{4} \left(\sum_{i=1}^{3} (x_{i2} - \mu_2)^2 + (x_{42} - \mu_2)^2 \right) = \frac{8}{4} \tag{2.98}$$

2.5　一维高斯分布均值的贝叶斯估计

样本集$D = \{x_1, \cdots, x_n\}$来自于1维高斯分布$N(\mu, \sigma^2)$，其中方差σ^2是已知的，计算均值μ的贝叶斯估计。假设均值的先验$p(\mu) \sim N(\mu_0, \sigma_0^2)$是以$\mu_0$为均值，$\sigma_0^2$为方差的高斯分布。

首先计算μ的后验概率密度$p(\mu|D)$，根据贝叶斯公式：

$$p(\mu|D) = \frac{p(D|\mu)p(\mu)}{p(D)} = \frac{p(D|\mu)p(\mu)}{\int p(D|\mu)p(\mu)d\mu} \tag{2.99}$$

由于$p(D) = \int p(D|\mu)p(\mu)d\mu$是与$\mu$及x无关的常数，因此令：

$$\alpha = \frac{1}{p(D)} = \frac{1}{\int p(D|\mu)p(\mu)d\mu} \tag{2.100}$$

样本集$D = \{x_1, \cdots, x_n\}$是独立同分布的，因此：

$$p(\mu|D) = \alpha p(D|\mu)p(\mu) \tag{2.101}$$

$$= \alpha \prod_{i=1}^{n} p(x_i|\mu)p(\mu) \tag{2.102}$$

$$= \alpha \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x_i-\mu)^2}{2\sigma^2}\right] \times \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left[-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}\right] \tag{2.103}$$

$$= \frac{\alpha}{\left(\sqrt{2\pi}\sigma\right)^n \sqrt{2\pi}\sigma_0} \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2 - \frac{1}{2\sigma_0^2}(\mu-\mu_0)^2\right] \tag{2.104}$$

$$= \alpha' \exp\left[-\frac{1}{2}\left(\frac{1}{\sigma^2}\sum_{i=1}^{n}x_i^2 - \frac{2\mu}{\sigma^2}\sum_{i=1}^{n}x_i + \frac{n\mu^2}{\sigma_0^2} + \frac{\mu^2}{\sigma_0^2} - \frac{2\mu_0\mu}{\sigma_0^2} + \frac{\mu_0^2}{\sigma_0^2}\right)\right] \tag{2.105}$$

$$= \alpha' \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}x_i^2 + \frac{\mu_0^2}{\sigma_0^2}\right] \exp\left\{-\frac{1}{2}\left(\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}\right)\mu^2 - \left(\frac{1}{\sigma^2}\sum_{i=1}^{n}x_i + \frac{\mu_0}{\sigma_0^2}\right)\mu\right\}$$

$$= \alpha'' \exp\left\{-\frac{1}{2}\left[\left(\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}\right)\mu^2 - 2\left(\frac{1}{\sigma^2}\sum_{i=1}^{n}x_i + \frac{\mu_0}{\sigma_0^2}\right)\mu\right]\right\} \tag{2.106}$$

上述过程中分2次对与μ无关项进行了归并，其中：

$$a' = \frac{\alpha}{(\sqrt{2\pi}\sigma)^n\sqrt{2\pi}\sigma_0}, \qquad a'' = a' \exp\left[-\frac{1}{2}\left(\frac{1}{\sigma^2}\sum_{i=1}^{n}x_i^2 + \frac{1}{\sigma_0^2}\mu_0^2\right)\right] \tag{2.107}$$

由上面的推导结果可以看出，$p(\mu|D)$的指数部分是关于μ的二次函数，由此可以断定$p(\mu|D)$服从高斯分布：$p(\mu|D) \sim N(\mu_n, \sigma_n^2)$。

$$p(\mu|D) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_n}{\sigma_n}\right)^2\right] \tag{2.108}$$

$$= \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left[-\frac{1}{2}\left(\frac{1}{\sigma_n^2}\mu^2 - \frac{2\mu_n}{\sigma_n^2}\mu + \frac{\mu_n^2}{\sigma_n^2}\right)\right] \tag{2.109}$$

对比公式(2.106)和(2.109)，可以得到：

$$\frac{1}{\sigma_n^2} = \frac{n}{\sigma^2} + \frac{1}{\sigma_0^2} \tag{2.110}$$

$$\frac{\mu_n}{\sigma_n^2} = \frac{1}{\sigma^2}\sum_{i=1}^{n}x_i + \frac{\mu_0}{\sigma_0^2} \tag{2.111}$$

因此：

$$\sigma_n^2 = \frac{\sigma^2\sigma_0^2}{n\sigma_0^2+\sigma^2} \tag{2.112}$$

$$\mu_n = \left(\frac{1}{\sigma^2}\sum_{i=1}^{n} x_i + \frac{\mu_0}{\sigma_0^2}\right)\frac{\sigma^2\sigma_0^2}{n\sigma_0^2+\sigma^2} \tag{2.113}$$

$$= \frac{\sigma_0^2}{n\sigma_0^2+\sigma^2}\sum_{i=1}^{n} x_i + \frac{\sigma^2\mu_0}{n\sigma_0^2+\sigma^2} \tag{2.114}$$

简化符号，令：$\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} x_i$，则有：

$$p(\mu|D) \sim N(\frac{\sigma_0^2}{n\sigma_0^2+\sigma^2}\hat{\mu}_n + \frac{\sigma^2\mu_0}{n\sigma_0^2+\sigma^2}, \frac{\sigma^2\sigma_0^2}{n\sigma_0^2+\sigma^2}) \tag{2.115}$$

这就是1维高斯分布均值μ的贝叶斯估计后验概率密度。有了分布参数μ的后验概率$p(\mu|D)$，下面来计算待识样本x的后验概率：

$$p(x|D) = \int p(x|\mu)p(\mu|D)d\mu \tag{2.116}$$

$$= \int \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]\frac{1}{\sqrt{2\pi}\sigma_n}\exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_n}{\sigma_n}\right)^2\right]d\mu \tag{2.117}$$

$$= \frac{1}{2\pi\sigma\sigma_n}\int \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 - \frac{1}{2}\left(\frac{\mu-\mu_n}{\sigma_n}\right)^2\right]d\mu \tag{2.118}$$

$$= \frac{1}{2\pi\sigma\sigma_n}\int \exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma^2} - \frac{2x\mu}{\sigma^2} + \frac{\mu^2}{\sigma^2} + \frac{\mu^2}{\sigma_n^2} - \frac{2\mu\mu_n}{\sigma_n^2} + \frac{\mu_n^2}{\sigma_n^2}\right)\right]d\mu \tag{2.119}$$

$$= \frac{1}{2\pi\sigma\sigma_n}\int \exp\left[-\frac{(\sigma_n^2+\sigma^2)\mu^2 - 2(\sigma_n^2x+\sigma^2\mu_n)\mu + (\sigma_n^2x^2+\sigma^2\mu_n^2)}{2\sigma^2\sigma_n^2}\right]d\mu \tag{2.120}$$

$$= \frac{1}{2\pi\sigma\sigma_n}\exp\left[\frac{(\sigma_n^2x+\sigma^2\mu_n)^2}{2\sigma^2\sigma_n^2(\sigma_n^2+\sigma^2)} - \frac{(\sigma_n^2x^2+\sigma^2\mu_n^2)}{2\sigma^2\sigma_n^2}\right]$$
$$\times \int \exp\left[-\frac{\sigma_n^2+\sigma^2}{2\sigma^2\sigma_n^2}\left(\mu - \frac{\sigma_n^2x+\sigma^2\mu_n}{\sigma_n^2+\sigma^2}\right)\right]d\mu \tag{2.121}$$

$$= \frac{f(\sigma,\sigma_n)}{2\pi\sigma\sigma_n}\exp\left[\frac{\sigma_n^2x^2 + 2\sigma_n^2\sigma^2\mu_nx + \sigma^4\mu_n^2 - \sigma_n^4x^2 - \sigma^2\sigma_n^2x^2 - \sigma_n^2\sigma^2\mu_n^2 - \sigma^4\mu_n^2}{2\sigma^2\sigma_n^2(\sigma_n^2+\sigma^2)}\right] \tag{2.122}$$

$$= \frac{f(\sigma,\sigma_n)}{2\pi\sigma\sigma_n}\exp\left[\frac{-\sigma^2\sigma_n^2x^2 + 2\sigma^2\sigma_n^2\mu_nx - \sigma^2\sigma_n^2\mu_n^2}{2\sigma^2\sigma_n^2(\sigma_n^2+\sigma^2)}\right] \tag{2.123}$$

$$= \frac{f(\sigma,\sigma_n)}{2\pi\sigma\sigma_n}\exp\left[-\frac{1}{2}\frac{x^2 - 2x\mu_n + \mu_n^2}{\sigma_n^2+\sigma^2}\right] \tag{2.124}$$

$$= \frac{f(\sigma,\sigma_n)}{2\pi\sigma\sigma_n}\exp\left[-\frac{1}{2}\frac{(x-\mu_n)^2}{\sigma_n^2+\sigma^2}\right] \tag{2.125}$$

其中公式(2.121)中的积分项简记为关于σ和σ_n的函数形式：

$$f(\sigma,\sigma_n) = \int \exp\left[-\frac{\sigma_n^2+\sigma^2}{2\sigma^2\sigma_n^2}\left(\mu - \frac{\sigma_n^2x+\sigma^2\mu_n}{\sigma_n^2+\sigma^2}\right)\right]d\mu \tag{2.126}$$

注意到被积函数是关于μ的二次指数函数，因此是一个高斯函数，而（3）式为高斯积分，其值的大小只与σ和σ_n有关，与μ，x和μ_n无关。根据上面的推导结果可以看出，样本x的后验概率密度$p(x|D)$服从高斯分布：

$$p(x|D) \sim N(\mu_n, \sigma_n^2 + \sigma^2) \tag{2.127}$$

$f(\sigma, \sigma_n)$只是一个归一化因子，不需要计算积分即可得到：

$$f(\sigma, \sigma_n) = \frac{\sqrt{2\pi}\sigma\sigma_n}{\sqrt{\sigma_n^2 + \sigma^2}} \tag{2.128}$$

第5章　线性判别函数

5.1　支持向量机的学习

给定两个类别的训练样本集$D = \{(\mathbf{x}_1, z_1), \cdots, (\mathbf{x}_n, z_n)\}$，$\mathbf{x}_i$为训练样本的特征矢量，$z_i$是样本的类别标识：

$$z_i = \begin{cases} +1, & \mathbf{x}_i \in \omega_1 \\ -1, & \mathbf{x}_i \in \omega_2 \end{cases} \tag{5.1}$$

5.1.1　线性可分的情况

我们先来讨论样本集D是线性可分的情况。作为最优分类超平面来说，首先需要能够对样本集D正确分类，亦即需要满足：

$$z_i(\mathbf{w}^t\mathbf{x}_i + w_0) > 0, \qquad \forall i = 1, \cdots, n \tag{5.2}$$

由于：

$$z_i(\mathbf{w}^t\mathbf{x}_i + w_0) \geq \min_{1 \leq j \leq n} \left[z_i(\mathbf{w}^t\mathbf{x}_j + w_0) \right] = b_{min} > 0 \tag{5.3}$$

b_{min}是训练样本集中距离分类超平面最近样本的函数间隔。当权值矢量$\mathbf{w}$和偏置w_0同时乘上一个正数$1/b_{min}$时，对应超平面的位置是不会发生变化的，因此公式(5.2)的条件可以重写为：

$$z_i(\mathbf{w}^t\mathbf{x}_i + w_0) \geq 1, \qquad \forall i = 1, \cdots, n \tag{5.4}$$

这样做的好处是通过适当调整权值矢量和偏置，使得最优超平面到样本集的函数间隔b变为了1，亦即支持面与分类界面之间的函数间隔为1。

最优分类超平面的第2个条件是要使得与样本集之间的几何间隔γ最大，在函数间隔为1的条件下有：

$$\gamma = \frac{1}{\|\mathbf{w}\|} \tag{5.5}$$

这样我们就得到了训练样本集线性可分条件下学习最优分类超平面的优化准则和约束条件：

原始优化问题

$$\min_{\mathbf{w},w_0} \quad \frac{1}{2}\|\mathbf{w}\|^2 \tag{5.6}$$

subject to

$$z_i(\mathbf{w}^t\mathbf{x}_i + w_0) \geq 1, \qquad i = 1, \cdots, n$$

这是一个典型的线性不等式约束条件下的二次优化问题，优化目标函数为$J_{SVM}(\mathbf{w}, w_0) = \frac{1}{2}\|\mathbf{w}\|^2$。在支持向量机的学习算法中，一般并不是直接求解这个原始问题，而是转而求解与其等价的对偶问题。

在说明对偶问题之前，先来看定义在矢量$\mathbf{u}$和$\mathbf{v}$上的函数$f(\mathbf{u}, \mathbf{v})$的min-max和max-min问题。

$$\text{min-max问题：} \begin{cases} f^*(\mathbf{u}) = \max_{\mathbf{v}} f(\mathbf{u}, \mathbf{v}) \\ \min_{\mathbf{u}} f^*(\mathbf{u}) = \min_{\mathbf{u}} \max_{\mathbf{v}} f(\mathbf{u}, \mathbf{v}) \end{cases} \tag{5.7}$$

$$\text{max-min问题：} \begin{cases} f^*(\mathbf{v}) = \min_{\mathbf{u}} f(\mathbf{u}, \mathbf{v}) \\ \max_{\mathbf{v}} f^*(\mathbf{v}) = \max_{\mathbf{v}} \min_{\mathbf{u}} f(\mathbf{u}, \mathbf{v}) \end{cases} \tag{5.8}$$

冯.诺依曼证明上述两个问题如果有解存在的话，必在同一点取得最优解，亦即：

$$\min_{\mathbf{u}} \max_{\mathbf{v}} f(\mathbf{u}, \mathbf{v}) = \max_{\mathbf{v}} \min_{\mathbf{u}} f(\mathbf{u}, \mathbf{v}) = f(\mathbf{u}^*, \mathbf{v}^*) \tag{5.9}$$

也就是说函数$f(\mathbf{u}, \mathbf{v})$先对$\mathbf{u}$取最小值再对$\mathbf{v}$取最大值，还是先对$\mathbf{v}$取最大值再对$\mathbf{u}$取最小值的结果是一样的，两个优化问题的求解顺序可以颠倒。

下面根据公式(5.6)的原始优化问题构造Lagrange函数：

$$L(\mathbf{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[z_i(\mathbf{w}^t\mathbf{x}_i + w_0) - 1 \right] \tag{5.10}$$

其中$\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n)^t$，$\alpha_i \geq 0$是针对公式(5.6)优化问题中每个约束不等式引入的Lagrange系数。

考虑Lagrange函数关于矢量$\boldsymbol{\alpha}$的最大化问题：$\max_{\boldsymbol{\alpha} \geq \mathbf{0}} L(\mathbf{w}, w_0, \boldsymbol{\alpha})$，当$z_i(\mathbf{w}^t\mathbf{x}_i + w_0) > 1$时，Lagrange函数在$\alpha_i = 0$处取得最大值；当$z_i(\mathbf{w}^t\mathbf{x}_i +$

$w_0) = 1$时，α_i可以大于0。总之当Lagrange函数取得最大值时，公式(5.10)求和式中的两个乘积项α_i和$z_i(\mathbf{w}^t\mathbf{x}_i + w_0) - 1$必有一项为0，因此：

$$\max_{\boldsymbol{\alpha}} L(\mathbf{w}, w_0, \boldsymbol{\alpha}) = J_{SVM}(\mathbf{w}, w_0) = \frac{1}{2}\|\mathbf{w}\|^2 \tag{5.11}$$

这样公式(5.6)的原始优化问题就等价于一个min-max问题。考虑到公式(5.9)式，这个问题也等价于一个max-min问题：

$$\min_{\mathbf{w},w_0} J_{SVM}(\mathbf{w}, w_0) = \min_{\mathbf{w},w_0} \max_{\boldsymbol{\alpha}} L(\mathbf{w}, w_0, \boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha}} \min_{\mathbf{w},w_0} L(\mathbf{w}, w_0, \boldsymbol{\alpha}) \tag{5.12}$$

首先计算Lagrange函数针对$\mathbf{w}$和w_0的最小化问题：

$$\frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i z_i \mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i z_i \mathbf{x}_i \tag{5.13}$$

$$\frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha})}{\partial w_0} = -\sum_{i=1}^{n} \alpha_i z_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i z_i = 0 \tag{5.14}$$

将公式(5.13)和(5.14)重新代入Lagrange函数：

$$L(\mathbf{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[z_i(\mathbf{w}^t\mathbf{x}_i + w_0) - 1 \right] \tag{5.15}$$

$$= \frac{1}{2} \left(\sum_{i=1}^{n} \alpha_i z_i \mathbf{x}_i \right)^t \left(\sum_{i=1}^{n} \alpha_i z_i \mathbf{x}_i \right) - \sum_{i=1}^{n} \left\{ \alpha_i z_i \left(\sum_{j=1}^{n} \alpha_j z_j \mathbf{x}_j \right)^t \mathbf{x}_i + \alpha_i z_i w_0 - \alpha_i \right\} \tag{5.16}$$

$$= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j \mathbf{x}_i^t \mathbf{x}_j - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j \mathbf{x}_i^t \mathbf{x}_j - w_0 \sum_{i=1}^{n} \alpha_i z_i + \sum_{i=1}^{n} \alpha_i \tag{5.17}$$

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j \mathbf{x}_i^t \mathbf{x}_j \tag{5.18}$$

此时，Lagrange函数只与优化矢量$\boldsymbol{\alpha}$有关，而与$\mathbf{w}, w_0$无关。因此，可以由Lagrange函数针对$\boldsymbol{\alpha}$的最大化，同时考虑公式(5.14)的约束，得到原始问题的对偶优化问题：

对偶优化问题

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j z_i z_j \mathbf{x}_i^t\mathbf{x}_j \tag{5.19}$$

subject to

$$\alpha_i \geq 0, \qquad i = 1, \cdots, n$$

$$\sum_{i=1}^{n}\alpha_i z_i = 0$$

原始优化问题和对偶优化问题都是典型的线性不等式约束条件下的二次优化问题，求解两者中的任何一个都是等价的。但SVM算法一般求解的是对偶问题，因为它有如下两个特点：

1. 对偶问题不直接优化权值矢量$\mathbf{w}$，因此与样本的特征维数d无关，只与样本的数量n有关。当样本的特征维数很高时，对偶问题更容易求解；

2. 对偶优化问题中，训练样本只以任意两个矢量内积的形式出现，因此只要能够计算矢量之间的内积，而不需要知道样本的每一维特征就可以进行优化求解；

以上两个特点使得我们可以很容易地将“核函数”引入到算法中，实现非线性的SVM分类。

5.1.2　线性不可分的情况

下面来看一下样本集D是线性不可分的情况。重新考察公式(5.6)的优化问题，当样本集线性不可分时，不存在任何一个权值矢量$\mathbf{w}$和偏置w_0能够使得作为约束的n个不等式都得到满足。通过在每个不等式上引入一个非负的“松弛变量”ξ，使得不等式变为：

$$z_i(\mathbf{w}^t\mathbf{x}_i + w_0) \geq 1 - \xi_i, \quad \xi_i \geq 0 \tag{5.20}$$

只要选择一系列适合的松弛变量ξ，不等式约束条件总是可以得到满足的。然而，即使训练样本集是线性不可分的，我们也希望学习得到的分类器能够正确识别尽量多的训练样本，换句话说就是希望尽量多的松弛变量$xi_i = 0$。因此目标函数就需要同时考虑两方面因素的优化：与分类界面和样本集之间的几何间隔相关的$\|w\|^2$，以及不为0的松弛变量的数量。

直接优化松弛变量的数量存在一定的难度，一般是转而优化一个相关的目标：$\sum_{i=1}^{n}\xi_i$。在一个优化问题中无法同时优化两个目标，需要引入一个大于0的常数C来协调对两个优化目标的关注程度。C值越大表示我们希望更少的训练样本被错误识别，C值越小表示我们希望分类界面与训练样本集的间隔更大。这样，我们就得到了在样本集线性不可分情况下的原始优化问题：

原始优化问题

$$\min_{\mathbf{w},w_0} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{5.21}$$

subject to

$$\begin{aligned} & z_i(\mathbf{w}^t\mathbf{x}_i + w_0) \geq 1 - \xi_i, \qquad i = 1, \cdots, n \\ & \xi_i \geq 0, \qquad i = 1, \cdots, n \end{aligned}$$

类似于线性可分情况，针对两组不等式约束分别引入Lagrange系数$\boldsymbol{\alpha}$和$\boldsymbol{\beta}$，建立Lagrange函数：

$$\begin{aligned} L(\mathbf{w}, w_0, \boldsymbol{\alpha}, \boldsymbol{\beta}) = & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ & - \sum_{i=1}^{n}\alpha_i\left[z_i(\mathbf{w}^t\mathbf{x}_i + w_0) - 1 + \xi_i\right] - \sum_{i=1}^{n}\beta_i\xi_i \end{aligned} \tag{5.22}$$

同样道理，原始问题的优化等价于Lagrange函数首先对$\mathbf{w}, w_0$和$\boldsymbol{\xi}$进行最小值优化，然后对$\boldsymbol{\alpha}, \boldsymbol{\beta}$在非负的约束下进行最大值优化：

$$\frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n}\alpha_i z_i \mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n}\alpha_i z_i \mathbf{x}_i \tag{5.23}$$

$$\frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial w_0} = -\sum_{i=1}^{n}\alpha_i z_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{n}\alpha_i z_i = 0 \tag{5.24}$$

$$\frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \tag{5.25}$$

将上述3式重新代入Lagrange函数：

$$L(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[z_i(\mathbf{w}^t\mathbf{x}_i + w_0) - 1 + \xi_i\right] - \sum_{i=1}^{n}\beta_i\xi_i \tag{5.26}$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i z_i \mathbf{x}_i\right)^t\left(\sum_{i=1}^{n}\alpha_i z_i \mathbf{x}_i\right) + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\beta_i\xi_i - \sum_{i=1}^{n}\left\{\alpha_i z_i\left(\sum_{j=1}^{n}\alpha_j z_j \mathbf{x}_j\right)^t \mathbf{x}_i + \alpha_i z_i w_0 - \alpha_i + \alpha_i\xi_i\right\} \tag{5.27}$$

$$= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j z_i z_j \mathbf{x}_i^t\mathbf{x}_j - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j z_i z_j \mathbf{x}_i^t\mathbf{x}_j - w_0\sum_{i=1}^{n}\alpha_i z_i + \sum_{i=1}^{n}\alpha_i + \sum_{i=1}^{n}(C - \alpha_i - \beta_i)\xi_i \tag{5.28}$$

$$= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j z_i z_j \mathbf{x}_i^t\mathbf{x}_j \tag{5.29}$$

可以看出，重写的Lagrange函数与线性可分情况是完全相同的，与$\mathbf{w}$，w_0，松弛矢量$\boldsymbol{\xi}$无关，也与引入的第2组Lagrange系数$\boldsymbol{\beta}$无关。对偶优化问题与线性可分情况的唯一不同点是由公式(5.25)式引入的：$\alpha_i = C - \beta_i$，考虑到$\beta_i \geq 0$，因此需要增加约束$\alpha_i \leq C$。

对偶优化问题

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j z_i z_j \mathbf{x}_i^t\mathbf{x}_j \tag{5.30}$$

subject to

$$C \geq \alpha_i \geq 0, \qquad i = 1, \cdots, n$$

$$\sum_{i=1}^{n}\alpha_i z_i = 0$$

在线性不可分的情况下，对偶问题相比于原始优化问题要简单。线性SVM分类器的学习，就是采用二次规划算法对公式(5.30)优化问题的求解。最优化方法的研究已经证明此类问题属于凸规划问题，存在唯一的最优解，并且可以由相关算法计算求解。常用的二次规划算法包括：内点法、有效集法、椭球算法等等，而且经过研究已经找到了专门针对SVM学习的有效算法：序列最小化算法（SMO, Sequential Minimal Optimization）。

通过对偶问题的优化，可以得到与每个训练样本相关的一组最优Lagrange系数$\boldsymbol{\alpha}$。构造线性判别函数需要的是权值矢量$\mathbf{w}$和偏置w_0，因此下面需要考虑如何由系数$\boldsymbol{\alpha}$计算$\mathbf{w}$和w_0。在此之前我们首先来看一下$\boldsymbol{\alpha}$中元素的含义。

从前面针对$\boldsymbol{\alpha}$优化的分析中可以看到，α_i是与公式(5.6)优化问题的第i个约束$z_i(\mathbf{w}^t\mathbf{x}+w_0)\geq 1$相关的Lagrange系数。当约束不等式以大于1的方式得到满足时，相应的Lagrange系数$\alpha_i=0$；而当约束以等于1的方式得到满足时，系数α_i可以大于0。同样道理，线性不可分情况下优化问题(5.21)中，由于有公式(5.25)中$\alpha_i=C-\beta_i$关系存在，因此当$\xi_i>0$时，Lagrange系数$\beta_i=0$，而$\alpha_i=C$；当$\xi_i=0$时，β_i可以大于0，相应的α_i可以小于C 。

更严格地说，依据最优化方法中的Kuhn-Tucker定理可以证明有如下关系存在：

$$\begin{cases} z_i(\mathbf{w}^t\mathbf{x}_i+w_0)>1, & \alpha_i=0 \\ z_i(\mathbf{w}^t\mathbf{x}_i+w_0)=1, & C>\alpha_i>0 \\ z_i(\mathbf{w}^t\mathbf{x}_i+w_0)<1, & \alpha_i=C \end{cases} \tag{5.31}$$

在建立学习优化问题的过程中，我们通过适当调整$\mathbf{w}$和w_0，使得距离最优分类界面最近的样本到分类超平面的函数间隔变为了1，亦即两个类别的支持面与分类超平面之间的函数间隔为1。因此，由图5.1可以看出，依据对偶优化问题的解，完全可以确定每个训练样本相对于最优分类超平面以及两个支持面之间的位置关系。$\alpha_i=0$对应的训练样本处于各自类别支持面之外；$C>\alpha_i>0$对应的训练样本处于支持面之上；$\alpha_i=C$对应的训练样本则处于各自类别支持面与分类超平面之间，甚至是分类界面的反方向区域（图5.1中黑色方框中的样本）。所有对应$\alpha_i>0$的训练样本称为支持向量。

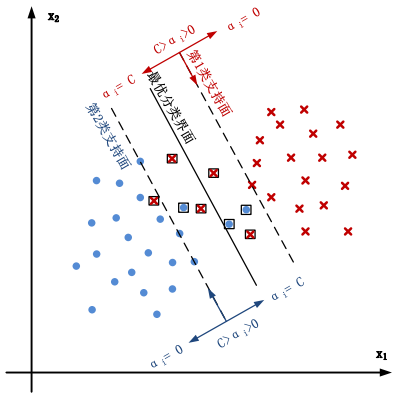

图 5.1: 支持向量与Lagrange系数

借助于公式(5.23)，可以将判别函数的权值$\mathbf{w}$表示为训练样本由相应Lagrange系数加权求和的形式：

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i z_i \mathbf{x}_i \tag{5.32}$$

因此，由对偶优化问题的解可以直接得到判别函数的权值矢量。这里需要注意的是实际上只有支持向量参与了求和式的计算，非支持向量的系数α_i为0，对$\mathbf{w}$的计算没有贡献。

任意一个处于支持面上的支持向量与分类界面之间的函数间隔为1，因此偏置w_0可以利用任意一个对应于$C > \alpha_i > 0$的支持向量$\mathbf{x}_i$由下述方程求得：

$$z_i(\mathbf{w}^t\mathbf{x}_i + w_0) = 1 \quad \Rightarrow \quad w_0 = z_i - \mathbf{w}^t\mathbf{x}_i \tag{5.33}$$

这样，我们就可以通过求解对偶优化问题得到一组Lagrange系数$\boldsymbol{\alpha}$，进而根据公式(5.32)和(5.33)计算线性判别函数的权值矢量$\mathbf{w}$和偏置w_0，得到最优的线性判别函数。

5.1.3　非线性支持向量机

将核函数引入支持向量机，将其转化为非线性分类器。支持向量机的学习过程主要是求解优化问题(5.30)，注意到其中只涉及到任意两个训练样本的内积计算，因此可以引入核函数κ将其转化为(5.34)式进行优化，实现首先将每个训练样本由某个非线性映射Φ映射到特征空间，然后在特征空间中求解最大间隔超平面的目的，在输入空间中对应于非线性分类界面。

非线性SVM 的优化问题

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j z_i z_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \tag{5.34}$$

subject to

$$C \geq \alpha_i \geq 0, \qquad i = 1, \cdots, n$$

$$\sum_{i=1}^{n} \alpha_i z_i = 0$$

通过(5.34)优化问题的求解，可以得到每个训练样本对应的Lagrange系数$\boldsymbol{\alpha}$，而要构造判别函数需要计算权值矢量$\mathbf{w}$和偏置w_0。注意到权值矢

量$\mathbf{w}$是一个经过Φ映射之后特征空间中的矢量，可以由公式(5.32)计算，只不过每个训练样本需要由映射之后的矢量$\Phi(\mathbf{x})$来代替：

$$\mathbf{w}^t\Phi(\mathbf{x}) + w_0 = \left[\sum_{i=1}^{n} \alpha_i z_i \Phi(\mathbf{x}_i)\right]^t \Phi(\mathbf{x}) + w_0 \tag{5.35}$$

$$= \sum_{i=1}^{n} \alpha_i z_i \kappa(\mathbf{x}, \mathbf{x}_i) + w_0 \tag{5.36}$$

可以看出，输入空间中的非线性SVM判别函数只需要利用核函数计算测试样本$\mathbf{x}$和训练样本$\mathbf{x}_i$在特征空间中的内积$\kappa(\mathbf{x}, \mathbf{x}_i)$：

$$g(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i z_i \kappa(\mathbf{x}, \mathbf{x}_i) + w_0 \tag{5.37}$$

偏置w_0同样可以由某个满足$C > \alpha_j > 0$的支持向量$\mathbf{x}_j$计算：

$$w_0 = z_j - \sum_{i=1}^{n} \alpha_i z_i \kappa(\mathbf{x}_j, \mathbf{x}_i) \tag{5.38}$$

这样我们看到，通过引入核函数可以实现非线性的支持向量机分类。所付出的代价是无法像线性SVM一样直接计算出权值矢量$\mathbf{w}$，而是需要在识别的时候，采用公式(5.37)利用核函数计算测试样本与训练样本在特征空间中的内积，从而得到判别函数的输出。由于非支持向量的Lagrange系数α为0，因此算法只需要保存和计算所有的支持向量即可。

5.1.4　多类别支持向量机

支持向量机可以采用“一对多方式”和“一对一方式”，将多类别问题转化为多个两类别问题来解决，但是这两种方式都存在着无法辨别的区域。在支持向量机中还可以采用一种特殊的方式进行多类别分类。

这种特殊的多类别分类方式是在原来“一对一方式”的基础之上，增加了一个“投票”的机制。首先，在学习过程中利用任意两个类别的样本学习出$c(c-1)/2$个区分两个类别的支持向量机分类器。在识别过程中，计算每一个支持向量机判别函数的输出，根据输出的正负向相关类别的“投票箱”中投入一票。例如，如果判别函数$g_{ij}(\mathbf{x}) > 0$，则在第i类的投票箱中投入一票，反之则在第j类的票箱中投入一票。最后，统计c个类别的得票

数，判别$\mathbf{x}$属于得票最多的类别。这个过程可以形式化的表示为：

$$v_i(\mathbf{x}) = \sum_{j=1, j\neq i}^{c} I(g_{ij} > 0) \tag{5.39}$$

$$\text{if} \quad k = \arg\max_{1\leq i\leq c} v_i(\mathbf{x}), \qquad \text{then} \quad \mathbf{x} \in \omega_k \tag{5.40}$$

其中I为示性函数：

$$I(a) = \begin{cases} 1, & a \text{ is true} \\ 0, & a \text{ is false} \end{cases} \tag{5.41}$$

5.1.5　支持向量机的最优性

结构风险最小化原则(SRM, Structural Risk Minimization)：把函数集$S = \{f(\mathbf{x}, \mathbf{w}), \mathbf{w} \in \Omega\}$分解为一个函数子集序列：

$$S_1 \subset S_2 \subset \cdots \subset S_k \subset \cdots \subset S \tag{5.42}$$

各个子集按照VC维的大小排序：

$$h_1 \leq h_2 \leq \cdots \leq h_k \leq \cdots \tag{5.43}$$

在子集序列中寻找经验风险与置信范围之和最小的子集，这个子集中使经验风险最小的函数就是所求的最优函数。

定理：在d维空间中，假设所有n个样本都在一个超球范围之内，超球的半径为R，那么γ-间隔分类超平面集合的VC维h满足如下不等式：

$$h \leq \min\left(\left[\frac{R^2}{\gamma^2}\right], d\right) + 1 \tag{5.44}$$

而间隔$\gamma = 1/\|\mathbf{w}\|$，因此根据SRM的原则，只需在保证经验风险为0的条件下（超平面能够正确分类全部训练样本），最小化权值矢量的长度$\|\mathbf{w}\|$。

第6章　第6章多层神经网络

6.1　BP 算法的推导

多层感知器学习的目的是要调整网络中每个神经元的权值矢量和偏置，使得对于训练样本集合D的识别误差最小。同线性的两层感知网络一样，多层感知器的权值学习也要转化为对误差平方和的优化问题求解。

令训练样本集包含n个样本$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$，根据每个样本的类别属性设定相应的期望输出矢量$\{\mathbf{t}_1, \cdots, \mathbf{t}_n\}$。如果将网络所有神经元的权值和偏置表示为一个统一的参数矢量$\mathbf{w}$，对应所有训练样本的网络实际输出为$\{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$，需要优化的平方误差函数为：

$$J(\mathbf{w}) = \sum_{i=1}^{n} \mathcal{E}_i(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{n} \|\mathbf{t}_i - \mathbf{z}_i\|^2 \tag{6.1}$$

在线性网络的学习中，我们优化的是同样的平方误差函数。由于线性网络的实际输出$\mathbf{z}_i$可以很容易地表示为网络参数$\mathbf{w}$的简洁表达式，通过微分运算就能够得到优化函数的极值点；而在多层感知器网络中，由于隐含层神经元的存在，实际输出$\mathbf{z}_i$与参数矢量$\mathbf{w}$之间的关系比较复杂，很难由简单的微分求解公式(6.1)平方误差函数的最小值点，需要采用梯度法进行优化，由初始的参数矢量$\mathbf{w}_0$开始，迭代优化直到收敛：

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \left. \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} \right|_{\mathbf{w}=\mathbf{w}_{t-1}} \tag{6.2}$$

考虑公式(6.1)，由于：

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \sum_{i=1}^{n} \frac{\partial \mathcal{E}_i(\mathbf{w})}{\partial \mathbf{w}} \tag{6.3}$$

因此算法的关键是要计算输入训练样本$\mathbf{x}_i$时的平方误差$\mathcal{E}_i(\mathbf{w}) = \frac{1}{2}\|\mathbf{t}_i - \mathbf{z}_i\|^2$关于每个网络参数的偏导数。下面我们以包含一个隐含层的三层感知器

网络为例，推导各个网络参数的迭代公式，包含多个隐含层的情况可以依此类推。

6.1.1　输出层

首先，考虑包含n_H个神经元的隐含层第j个神经元与包含L个神经元的输出层第k个神经元之间的连接权值w_{kj}以及偏置b_k（如图6.1）。令隐含层神经元的输出为$\{y_1, \cdots, y_{n_H}\}$，输出层神经元的净输入为$\{net_1, \cdots, net_L\}$，输出层和隐含层神经元的激活函数分别为$f_o(u)$和$f_h(u)$，应用隐函数链式求导法则有：

$$\frac{\partial \mathcal{E}}{\partial w_{kj}} = \frac{\partial \mathcal{E}}{\partial z_k} \times \frac{\partial z_k}{\partial net_k} \times \frac{\partial net_k}{\partial w_{kj}} \tag{6.4}$$

$$\frac{\partial \mathcal{E}}{\partial b_k} = \frac{\partial \mathcal{E}}{\partial z_k} \times \frac{\partial z_k}{\partial net_k} \times \frac{\partial net_k}{\partial b_k} \tag{6.5}$$

其中，$\mathcal{E}$为输入$\mathbf{x_i}$时网络输出的平方误差，为简洁起见省略了符号的下标i。

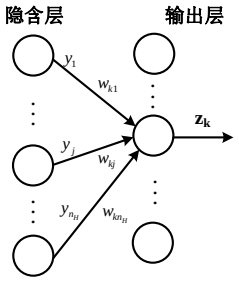

图 6.1: 输出层神经元权值的学习

由于：

$$\mathcal{E} = \frac{1}{2}\|\mathbf{t} - \mathbf{z}\|^2 = \frac{1}{2}\sum_{l=1}^{L}(t_l - z_l)^2 \quad \Rightarrow \quad \frac{\partial \mathcal{E}}{\partial z_k} = -(t_k - z_k) \tag{6.6}$$

$$z_k = f_o(net_k) \quad \Rightarrow \quad \frac{\partial z_k}{\partial net_k} = f_o'(net_k) \tag{6.7}$$

$$net_k = \sum_{l=1}^{L} w_{kl} y_l + b_k \quad \Rightarrow \quad \frac{\partial net_k}{\partial w_{kj}} = y_j, \quad \frac{\partial net_k}{\partial b_k} = 1 \tag{6.8}$$

代入公式(6.4)和(6.5)：

$$\frac{\partial \mathcal{E}}{\partial w_{kj}} = -(t_k - z_k) f_o'(net_k) y_j \tag{6.9}$$

$$\frac{\partial \mathcal{E}}{\partial b_k} = -(t_k - z_k) f_o'(net_k) \tag{6.10}$$

定义$\delta_k = (t_k - z_k) f_o'(net_k)$，则可以得到平方误差关于输出层神经元参数$w_{kj}$和$k_b$的梯度：

$$\frac{\partial \mathcal{E}}{\partial w_{kj}} = -\delta_k y_j, \qquad \frac{\partial \mathcal{E}}{\partial b_k} = -\delta_k \tag{6.11}$$

6.1.2　隐含层

现在考虑隐含层第j个节点与输入层第i个节点之间的连接权值w_{ji}以及偏置b_j（如图6.2）。与输出层类似，应用隐函数求导规则：

$$\frac{\partial \mathcal{E}}{\partial w_{ji}} = \frac{\partial \mathcal{E}}{\partial y_j} \times \frac{\partial y_j}{\partial net_j} \times \frac{\partial net_j}{\partial w_{ji}} \tag{6.12}$$

$$\frac{\partial \mathcal{E}}{\partial b_j} = \frac{\partial \mathcal{E}}{\partial y_j} \times \frac{\partial y_j}{\partial net_j} \times \frac{\partial net_j}{\partial b_j} \tag{6.13}$$

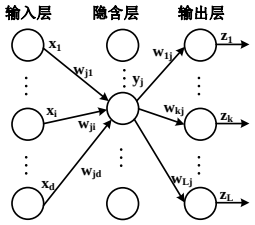

图 6.2: 隐含层神经元权值的学习

其中：

$$y_j = f_h(net_j) \qquad \Rightarrow \quad \frac{\partial y_j}{\partial net_j} = f_h'(net_j) \tag{6.14}$$

$$net_j = \sum_{k=1}^{d} w_{jk} x_k + b_j \qquad \Rightarrow \quad \frac{\partial net_j}{\partial w_{ji}} = x_i, \quad \frac{\partial net_j}{\partial b_j} = 1 \tag{6.15}$$

$\mathcal{E}$不能直接表示成y_j的函数，但可以表示成输出层$\{z_1, \cdots, z_L\}$的函数，而每一个输出z_k都是y_j的函数，$\partial \mathcal{E}/\partial y_j$的计算相对于输出层要复杂一些。

由于：

$$\mathcal{E} = \frac{1}{2}\sum_{k=1}^{L}(t_k - z_k)^2, \qquad z_k = f_o(net_k) \tag{6.16}$$

$$net_k = \sum_{l=1}^{n_H} w_{kl} y_l + b_k \tag{6.17}$$

因此：

$$\frac{\partial \mathcal{E}}{\partial y_j} = -\sum_{k=1}^{L}(t_k - z_k)\frac{z_k}{y_j} \tag{6.18}$$

$$= -\sum_{k=1}^{L}(t_k - z_k)\frac{z_k}{net_k}\frac{\partial net_k}{\partial y_j} \tag{6.19}$$

$$= -\sum_{k=1}^{L}(t_k - z_k) f_o'(net_k) w_{kj} \tag{6.20}$$

$$= -\sum_{k=1}^{L}\delta_k w_{kj} \tag{6.21}$$

将公式(6.14)、(6.15)和(6.21)代入(6.12)、(6.13)：

$$\frac{\partial \mathcal{E}}{\partial w_{jm}} = -\left[\sum_{k=1}^{L}\delta_k w_{kj}\right] f_h'(net_j) x_m \tag{6.22}$$

$$\frac{\partial \mathcal{E}}{\partial b_j} = -\left[\sum_{k=1}^{L}\delta_k w_{kj}\right] f_h'(net_j) \tag{6.23}$$

定义

$$\delta_j = f_h'(net_j)\sum_{k=1}^{L}\delta_k w_{kj} \tag{6.24}$$

则可以得到平方误差关于隐含层神经元参数w_{jm}和b_j的梯度：

$$\frac{\partial \mathcal{E}}{\partial w_{jm}} = -\delta_j x_m \tag{6.25}$$

$$\frac{\partial \mathcal{E}}{\partial b_j} = -\delta_j \tag{6.26}$$

由公式(6.11)和(6.25)、(6.26)，我们得到了平方误差函数$\mathcal{E}$关于输出层和隐含层参数的梯度。注意到输出层需要计算的主要是每个节点的$\delta_k = (t_k - z_k) f_o'(net_k)$，某种程度上这可以看作是输出节点$k$上的误差；而隐含

层每个节点计算$\delta_j = f_h'(net_j)\sum_{k=1}^{L}\delta_k w_{kj}$时需要用到所有输出层节点的误差$\delta_k$，这也可以看作是隐含层节点$j$的误差。由于隐含层节点的误差需要由输出层节点的误差反向计算得到，因此多层感知器网络参数的学习算法也被称为误差反向传播算法（Backpropagation Alogrithm，BP 算法）。

6.2　Levenberg-Marquardt 算法（LM 算法）

LM算法是一种专门适合于平方误差优化函数的方法。首先，我们需要将全部n个训练样本在L个网络输出神经元上的期望输出和实际输出写成矢量形式：

$$\mathbf{t} = (t_{11}, \cdots, t_{1L}, t_{21}, \cdots, t_{nL})^t \tag{6.27}$$

$$\mathbf{z} = (z_{11}, \cdots, z_{1L}z_{21}, \cdots, z_{nL})^t \tag{6.28}$$

其中t_{ij}和z_{ij}分别为第i个训练样本在第j个输出神经元上的期望输出和实际输出。定义$n \times L$维矢量：$\mathbf{v}(\mathbf{w}) = \mathbf{t} - \mathbf{z}$，对比公式(6.1)可以看出，平方误差优化函数可以表示为：

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{v}^t(\mathbf{w})\mathbf{v}(\mathbf{w}) \tag{6.29}$$

平方误差函数的梯度：

$$\nabla J(\mathbf{w}) = \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \left(\frac{\partial \mathbf{v}(\mathbf{w})}{\partial \mathbf{w}}\right)^t \mathbf{v}(\mathbf{w}) = \tilde{J}^t(\mathbf{w})\mathbf{v}(\mathbf{w}) \tag{6.30}$$

其中$\tilde{J}(\mathbf{w})$为矢量$\mathbf{v}$关于$\mathbf{w}$的Jacobian矩阵，令N为网络权值数：

$$\tilde{J}(\mathbf{w}) = \frac{\partial \mathbf{v}(\mathbf{w})}{\partial \mathbf{w}} = \begin{pmatrix} \frac{\partial v_1(\mathbf{w})}{\partial w_1} & \frac{\partial v_1(\mathbf{w})}{\partial w_2} & \cdots & \frac{\partial v_1(\mathbf{w})}{\partial w_N} \\ \frac{\partial v_2(\mathbf{w})}{\partial w_1} & \frac{\partial v_2(\mathbf{w})}{\partial w_2} & \cdots & \frac{\partial v_2(\mathbf{w})}{\partial w_N} \\ \vdots & vdots & \ddots & \vdots \\ \frac{\partial v_{nL}(\mathbf{w})}{\partial w_1} & \frac{\partial v_{nL}(\mathbf{w})}{\partial w_2} & \cdots & \frac{\partial v_{nL}(\mathbf{w})}{\partial w_N} \end{pmatrix} \tag{6.31}$$

进一步，计算优化函数$J(\mathbf{w})$的Hessian 矩阵：

$$H = \frac{\partial(\nabla J(\mathbf{w}))}{\partial \mathbf{w}} = \tilde{J}^t(\mathbf{w})\frac{\partial \mathbf{v}(\mathbf{w})}{\partial \mathbf{w}} + S(\mathbf{w}) \tag{6.32}$$

其中$S(\mathbf{w})$由Jacobian矩阵关于$\mathbf{w}$的微分和$\mathbf{v}(\mathbf{w})$计算，如果假设$S(\mathbf{w}$很小，则可以得到Hessian矩阵的近似：

$$H \approx \tilde{J}^t(\mathbf{w})\tilde{J}(\mathbf{w}) \tag{6.33}$$

由此可以看出，Hessian矩阵可以由Jacobian矩阵近似计算。将公式(6.30)和(6.33)式带入到牛顿法的权值增量计算公式：

$$\Delta \mathbf{w} = -H^{-1}\left(\frac{\partial J}{\partial \mathbf{w}}\right) = -\left[\tilde{J}^t(\mathbf{w})\tilde{J}(\mathbf{w})\right]^{-1}\tilde{J}^t(\mathbf{w})\mathbf{v}(\mathbf{w}) \tag{6.34}$$

按照这种方式计算的优点是无需计算二阶导数矩阵H，但可以近似得到二阶优化的效果，这种方法一般称为高斯-牛顿法。

在实际应用中，矩阵$\tilde{J}^t(\mathbf{w})\tilde{J}(\mathbf{w})$可能是奇异矩阵，不可逆的，一般采用如下方式计算权值矢量的增量：

$$\Delta \mathbf{w} = -\left[\tilde{J}^t(\mathbf{w})\tilde{J}(\mathbf{w}) + \mu I\right]^{-1}\tilde{J}^t(\mathbf{w})\mathbf{v}(\mathbf{w}) \tag{6.35}$$

其中I是单位矩阵，$\mu > 0$。算法每一轮迭代中尝试不同的μ，在保证矩阵可逆的条件下使得μ尽量小，这种方法一般称为Levenberg-Marquardt算法。

Jacobian矩阵$\tilde{J}(\mathbf{w})$的维数为$nL \times N$，当样本很多时维数会很大。但在算法实现时可以直接计算$N \times N$维矩阵$\tilde{J}^t(\mathbf{w})\tilde{J}(\mathbf{w})$和$N$维矢量$\tilde{J}^t(\mathbf{w})\mathbf{v}(\mathbf{w})$，因此存储复杂度只是$O(N^2)$。

Jacobian矩阵中每个元素的计算类似于BP算法对w中每个元素导数的计算，也需要一个由输出层到各个隐含层的反馈过程，由于只是一次项，因此还要简单一些。

LM算法避免了直接计算Hessian矩阵，但仍然需要计算$N \times N$维矩阵的逆阵，每一轮迭代的计算复杂度比BP算法要高得多。但由于近似实现了二阶优化技术，迭代次数会远远少于BP算法，所以学习过程的收敛速度一般明显快于梯度下降的BP算法。LM算法的存储复杂度和计算复杂度都与N相关，因此一般适用于网络规模不大，参数数量适中的问题。

6.3　快速传播算法(Quickprop)的推导

MLP训练中，假设各个权值相互独立的，可以单独进行训练，而且误差函数为权值的2次函数：

$$J(w) = aw^2 + bw + c \tag{6.36}$$

误差函数$J(\mathbf{w})$对权值$\mathbf{w}$求导数：

$$\frac{dJ(w)}{dw} = 2aw + b \tag{6.37}$$

令w_{m-1}, w_m, w_{m+1}分别表示第$m-1, m$和$m+1$步迭代时的权值，$m-1$及m步迭代时的权值增量为$\Delta w(m), \Delta w(m-1)$，则：

$$w_m = w_{m-1} + \Delta w(m) \tag{6.38}$$

$$w_{m+1} = w_m + \Delta w(m+1) \tag{6.39}$$

计算第m步的最优权值增量$\Delta w(m+1)$，希望在第m步迭代之后，权值由w_m变为w_{m+1}，而w_{m+1}到达误差函数$J(w)$的最小值点，因此有：

$$\left.\frac{dJ(w)}{dw}\right|_{w=w_{m+1}} = 2aw_{m+1} + b = 0 \tag{6.40}$$

将公式(6.39)式代入：

$$2a\left[w_m + \Delta w(m+1)\right] + b = 0 \tag{6.41}$$

因此：

$$\Delta w(m+1) = -\frac{2aw_m + b}{2a} = -\frac{2aw_m + b}{2a\Delta w(m)}\Delta w(m) \tag{6.42}$$

由公式(6.37)式，还有：

$$\left.\frac{dJ(w)}{dw}\right|_{w=w_m} = 2aw_m + b \tag{6.43}$$

$$\left.\frac{dJ(w)}{dw}\right|_{w=w_{m-1}} = 2aw_{m-1} + b \tag{6.44}$$

因此：

$$\left.\frac{dJ(w)}{dw}\right|_{w=w_{m-1}} - \left.\frac{dJ(w)}{dw}\right|_{w=w_m} = 2aw_{m-1} - 2aw_m \tag{6.45}$$

$$= 2aw_{m-1} - 2a\left[w_{m-1} + \Delta w(m)\right] \tag{6.46}$$

$$= -2a\Delta w(m) \tag{6.47}$$

将公式(6.43)和(6.47)代入(6.42)：

$$\Delta w(m+1) = -\frac{2aw_m + b}{2a\Delta w(m)}\Delta w(m) \tag{6.48}$$

$$= \frac{\left.\frac{dJ(w)}{dw}\right|_{w=w_m}}{\left.\frac{dJ(w)}{dw}\right|_{w=w_{m-1}} - \left.\frac{dJ(w)}{dw}\right|_{w=w_m}} \tag{6.49}$$

第8章　第8章成分分析与核函数

8.1　矢量与坐标系

特征矢量一般都是以坐标的形式表示的，矢量的每一个分量是它在相应坐标轴上的投影。坐标系可以用坐标原点O和一组代表各个坐标轴的“基矢量”$\{\mathbf{e}_1, \cdots, \mathbf{e}_d\}$来表示，如果基矢量之间满足如下关系：

$$\mathbf{e}_i^t \mathbf{e}_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{8.1}$$

则称这组基矢量构成了一个标准正交系，所谓“标准”是指每个矢量的长度均为1，所谓“正交”是指任意两个矢量之间是正交的。这样构成的坐标系就是我们所熟悉的直角坐标系。

如图8.1所示，在一个直角坐标系下，矢量$\mathbf{x} = (x_1, \cdots, x_d)^t$可以表示为基矢量的线性组合：

$$\mathbf{x} = x_1\mathbf{e}_1 + \cdots + x_d\mathbf{e}_d = \sum_{i=1}^{d} x_i\mathbf{e}_i \tag{8.2}$$

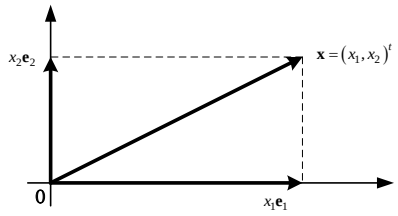

图 8.1: 矢量和直角坐标系

很显然，根据公式(8.1)和(8.2)，我们可以通过计算矢量$\mathbf{x}$与基矢量$\mathbf{e}_j$的内积来得到相应的坐标分量：

$$\mathbf{x}^t\mathbf{e}_j = \sum_{i=1}^{d} x_i\mathbf{e}_i^t\mathbf{e}_j = x_j \tag{8.3}$$

8.2　主成分分析算法的推导

我们知道，样本集合中的每一个样本都对应着特征空间中的一个点，同样的一个样本点在不同坐标系下对应着不同的矢量，例如图8.2中某个样本对应着特征空间中的点A，在以O为坐标原点，$\{\mathbf{v}_1, \mathbf{v}_2\}$为基矢量的原坐标系下，$A$点对应的矢量是$\mathbf{x}$；而在$O.$为原点，$\{\mathbf{e}_1, \mathbf{e}_2\}$为基矢量的新坐标系下，对应的矢量是$\mathbf{x}'$。如果新坐标系的原点$O'$在原坐标系下对应的矢量是$\boldsymbol{\mu}$，显然有如下关系：

$$\mathbf{x} = \boldsymbol{\mu} + \sum_{i=1}^{d} a_i\mathbf{e}_i \tag{8.4}$$

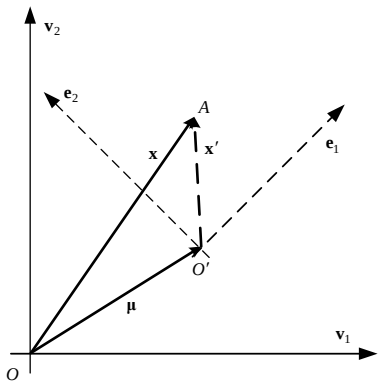

图 8.2: 同一个样本在不同坐标系下的表示

在新坐标系下，矢量$\mathbf{x}$的元素可以由原坐标系下的矢量$\mathbf{x}$以及$\boldsymbol{\mu}$和基矢量$\{\mathbf{e}_1, \cdots, \mathbf{e}_d\}$计算得到：

$$a_i = \mathbf{e}_i^t(\mathbf{x} - \boldsymbol{\mu}), \qquad i = 1, \cdots, d \tag{8.5}$$

同样也可以根据公式(8.4)由新坐标下的矢量$\mathbf{x}'$来恢复原矢量$\mathbf{x}$，不会存在任何误差。然而如果我们只保留新坐标系下$d' < d$个元素，然后用保留

的d'个元素来恢复原坐标系下的d维矢量：

$$\hat{\mathbf{x}} = \boldsymbol{\mu} + \sum_{i=1}^{d'} a_i \mathbf{e}_i \tag{8.6}$$

显然$\hat{\mathbf{x}}$只是对$\mathbf{x}$的近似，用$\hat{\mathbf{x}}$来代替$\mathbf{x}$就会出现一定的误差，误差的大小同新坐标系的位置、基矢量的方向以及保留哪些特征有关。

在主成分分析方法中，新坐标系的原点选择在训练样本集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$的均值矢量$\boldsymbol{\mu}$上，然后寻找一组最优的基矢量$\{\mathbf{e}_1, \cdots, \mathbf{e}_d\}$，使得在只保留前$d'$个元素的条件下，由新的坐标根据公式(8.6)恢复样本集D的均方误差最小，即求解如下的优化问题：

$$\min_{\mathbf{e}_1, \cdots, \mathbf{e}_d} \frac{1}{n} \sum_{k=1}^{n} \|\mathbf{x}_k - \hat{\mathbf{x}}_k\|^2 \tag{8.7}$$

其中$\hat{\mathbf{x}}_k$是根据公式(8.5)将$\mathbf{x}_k$由原坐标系变换到新坐标系下，然后再根据公式(8.6)只使用前d'个特征恢复的近似矢量。如果用a_{ki}表示第k个样本在新坐标系下的第i维特征，由公式(8.4)和(8.6)式可以得到：

$$\mathbf{x}_k - \hat{\mathbf{x}}_k = \sum_{i=1d'+1}^{d} a_{ki} \mathbf{e}_i \tag{8.8}$$

代入到公式(8.7)：

$$\frac{1}{n} \sum_{k=1}^{n} \|\mathbf{x}_k - \hat{\mathbf{x}}_k\|^2 = \frac{1}{n} \sum_{k=1}^{n} \left\| \sum_{i=d'+1}^{d} a_{ki} \mathbf{e}_i \right\|^2 \tag{8.9}$$

$$= \frac{1}{n} \sum_{k=1}^{n} \left(\sum_{i=d'+1}^{d} a_{ki} \mathbf{e}_i \right)^t \left(\sum_{i=d'+1}^{d} a_{ki} \mathbf{e}_i \right) \tag{8.10}$$

$$= \frac{1}{n} \sum_{k=1}^{n} \sum_{i=d'+1}^{d} a_{ki}^2 \tag{8.11}$$

$$= \frac{1}{n} \sum_{k=1}^{n} \sum_{i=d'+1}^{d} \left[\mathbf{e}_i^t (\mathbf{x}_k - \boldsymbol{\mu}) \right] \left[\mathbf{e}_i^t (\mathbf{x}_k - \boldsymbol{\mu}) \right]^t \tag{8.12}$$

$$= \sum_{i=d'+1}^{d} \mathbf{e}_i^t \left[\frac{1}{n} \sum_{k=1}^{n} (\mathbf{x}_k - \boldsymbol{\mu})(\mathbf{x}_k - \boldsymbol{\mu})^t \right] \mathbf{e}_i \tag{8.13}$$

其中(8.10)到(8.11)利用了$\{\mathbf{e}_1,\cdots,\mathbf{e}_d\}$是新坐标系的基矢量，因此构成了一个标准正交系：

$$\mathbf{e}_i^t\mathbf{e}_j=\begin{cases}1, & i=j\\ 0, & i\neq j\end{cases} \tag{8.14}$$

而(8.11)到(8.12)则是基于如下事实：a_{ki}是一个标量，它的转置与其自身相等，并且有公式(8.5)成立，因此：

$$a_{ki}=\mathbf{e}_i^t(\mathbf{x}_k-\boldsymbol{\mu})=\left[\mathbf{e}_i^t(\mathbf{x}_k-\boldsymbol{\mu})\right]^t \tag{8.15}$$

如果定义矩阵：

$$\Sigma=\frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k-\boldsymbol{\mu})(\mathbf{x}_k-\boldsymbol{\mu})^t \tag{8.16}$$

恰好是样本集D的协方差矩阵，则公式(8.7)的优化问题变为：

$$\min_{\mathbf{e}_1,\cdots,\mathbf{e}_d}\sum_{i=d'+1}^{d}\mathbf{e}_i^t\Sigma\mathbf{e}_i \tag{8.17}$$

仔细观察公式(8.17)会发现，直接求解这个优化问题是没有意义的。由于Σ是半正定矩阵，因此当$\mathbf{e}_1,\cdots,\mathbf{e}_d=\mathbf{0}$时取得最小值0，显然零矢量并不能作为基矢量，导致这样结果的原因在于优化(8.17)时没有约束$\mathbf{e}_1,\cdots,\mathbf{e}_d$的长度。主成分分析在求解新坐标系的基矢量时，优化的是如下的约束问题：

$$\min_{\mathbf{e}_1,\cdots,\mathbf{e}_d}\sum_{i=d'+1}^{d}\mathbf{e}_i^t\Sigma\mathbf{e}_i \tag{8.18}$$

$$\text{subject to} \tag{8.19}$$

$$\|\mathbf{e}_i\|^2=1,\quad i=1,\cdots,d \tag{8.20}$$

有约束优化问题可以通过构造Lagrange函数转化为无约束问题（参见最优化方法中的“约束优化”）：

$$L(\mathbf{e}_1,\cdots,\mathbf{e}_d,\lambda_1,\cdots,\lambda_d)=\sum_{i=d'+1}^{d}\mathbf{e}_i^t\Sigma\mathbf{e}_i-\sum_{i=1}^{d}\lambda_i(\mathbf{e}_i^t\mathbf{e}_i-1) \tag{8.21}$$

对每一个基矢量$\mathbf{e}_j$求导数：

$$\frac{\partial L(\mathbf{e}_1,\cdots,\mathbf{e}_d,\lambda_1,\cdots,\lambda_d)}{\partial\mathbf{e}_j}=2\Sigma\mathbf{e}_j-2\lambda_j\mathbf{e}_j=\mathbf{0} \tag{8.22}$$

其中利用到了Σ为对称矩阵的事实。由此得到优化问题(8.18)的解应满足：

$$\Sigma \mathbf{e}_j = \lambda_j \mathbf{e}_j \tag{8.23}$$

显然，使得上式成立的λ_j和$\mathbf{e}_j$分别为矩阵Σ的特征值和对应的特征矢量。由此我们可以得到这样的结论：如果我们希望将一个样本集合D中的d维特征矢量在一个新的坐标系下只用d'个特征表示，那么应该将新坐标系的坐标原点放在D的均值$\boldsymbol{\mu}$的位置，而以集合D的协方差矩阵的特征矢量$\mathbf{e}_1, \cdots, \mathbf{e}_d$作为基矢量，这样可以保证只用保留的$d'$维特征恢复原矢量时均方误差最小。

通过这样的方式可以得到一个最优的新坐标系，注意到Σ是一个$d \times d$的矩阵，存在d个特征值和特征矢量，现在的问题是我们只希望保留新坐标系中的d'个坐标，应该保留哪些坐标才能够保证恢复出的d维特征矢量的均方误差最小？回到优化问题(8.18)，将公式(8.23)代入优化函数：

$$\sum_{i=d'+1}^{d} \mathbf{e}_i^t \Sigma \mathbf{e}_i = \sum_{i=d'+1}^{d} \lambda_i \mathbf{e}_i^t \mathbf{e}_i = \sum_{i=d'+1}^{d} \lambda_i \tag{8.24}$$

要使得优化目标最小，只需要选择$\lambda_{d'+1}, \cdots, \lambda_d$是$\Sigma$最小的$d - d'$个特征值。这里需要注意一点，在整个推导过程中我们约定的是要保留新坐标系下前d'个特征，而放弃掉后面的$d - d'$个特征，因此在新的坐标系下应该选择保留的是Σ最大的d'个特征值对应的特征矢量作为新坐标系的基矢量。

8.3　线性判别分析算法的推导

先从一种简单的情况入手来研究这个问题，我们将两个类别的样本向一条通过坐标原点的直线上投影，也就是用1维特征来表示d维矢量，希望在1维空间中两类样本的可分性最大。从图(8.3)可以看出在不同方向的直线上，两类样本的可分性是不同的，如果想要找到一个最优的投影直线方向，首先需要对1维空间中样本的可分性进行度量。

假设两类问题的样本集为：

$$D_1 = \left\{ \mathbf{x}_1^{(1)}, \cdots, \mathbf{x}_{n_1}^{(1)} \right\}, \quad D_2 = \left\{ \mathbf{x}_1^{(2)}, \cdots, \mathbf{x}_{n_2}^{(2)} \right\} \tag{8.25}$$

投影直线的单位矢量为$\mathbf{w}$，d维空间的矢量$\mathbf{x}$在这条直线上的投影为一个标量：

$$y = \mathbf{w}^t \mathbf{x} \tag{8.26}$$

图 8.3: 二维模式在一维空间的投影

两类样本集经过投影之后成为标量集：

$$D_1 \longrightarrow \mathcal{Y}_1 = \{y_1^{(1)}, \cdots, y_{n_1}^{(1)}\}, \quad D_2 \longrightarrow \mathcal{Y}_2 = \{y_1^{(2)}, \cdots, y_{n_2}^{(2)}\} \tag{8.27}$$

不同类样本的分散程度越大，同类样本的聚集程度越高则类别之间的可分性越强。在1维空间中可以用两个类别样本均值之差的平方$(\tilde{\mu}_1 - \tilde{\mu}_2)^2$来度量两类样本的分散程度；而样本类内的离散程度可以用样本的方差之和来度量$\tilde{s}_1^2 + \tilde{s}_2^2$。综合考虑类内的聚集程度和类间的分散程度，可以建立如下的Fisher准则：

$$J(\mathbf{w}) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2} \tag{8.28}$$

Fisher准则函数的值越大，类别的可分性则越强。下面写出准则函数$J(\mathbf{w})$关于投影直线方向矢量$\mathbf{w}$的显式表达式，首先计算投影之后类别的均值$\tilde{\mu}$：

$$\tilde{\mu}_i = \frac{1}{n_i} \sum_{y \in \mathcal{Y}_i} y = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{w}^t \mathbf{x} = \mathbf{w}^t \boldsymbol{\mu}_i, \quad i = 1, 2 \tag{8.29}$$

投影之后两类均值之差的平方可以表示为：

$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = (\mathbf{w}^t \boldsymbol{\mu}_1 - \mathbf{w}^t \boldsymbol{\mu}_2)^2 \tag{8.30}$$

$$= \mathbf{w}^t (\mu_1 - \boldsymbol{\mu}_2)(\mu_1 - \boldsymbol{\mu}_2)^t \mathbf{w} \tag{8.31}$$

$$= \mathbf{w}^t S_b \mathbf{w} \tag{8.32}$$

其中$S_b = (\mu_1 - \boldsymbol{\mu}_2)(\mu_1 - \boldsymbol{\mu}_2)^t$是类间散布矩阵（假设两类的先验概率相

等)。类似的：

$$\tilde{s}_i^2 = \sum_{y \in \mathcal{Y}_i} (y - \tilde{\mu}_i)^2 \tag{8.33}$$

$$= \sum_{\mathbf{x} \in D_i} (\mathbf{w}^t \mathbf{x} - \mathbf{w}^t \boldsymbol{\mu}_i)^2 \tag{8.34}$$

$$= \sum_{\mathbf{x} \in D_i} \mathbf{w}^t (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^t \mathbf{w} \tag{8.35}$$

$$= \mathbf{w}^t S_i \mathbf{w} \tag{8.36}$$

其中

$$S_i = \sum_{\mathbf{x} \in D_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^t \tag{8.37}$$

称为第i类的类内散度矩阵。总的方差：

$$\tilde{s}_1^2 + \tilde{s}_2^2 = \mathbf{w}^t S_1 \mathbf{w} + \mathbf{w}^t S_2 \mathbf{w} \tag{8.38}$$

$$= \mathbf{w}^t (S_1 + S_2) \mathbf{w} \tag{8.39}$$

$$= \mathbf{w}^t S_w \mathbf{w} \tag{8.40}$$

将公式(8.32)和(8.40)代入Fisher准则，可以得到如下优化问题：

$$\max_{\mathbf{w}} \frac{\mathbf{w}^t S_b \mathbf{w}}{\mathbf{w}^t S_w \mathbf{w}} \tag{8.41}$$

上式也被称为是Rayleigh商的优化问题。实际上这个问题存在无穷多个解，因为如果$\mathbf{w}^*$是一个最优解，对于任意的$a \neq 0$，$a\mathbf{w}^*$同样是最优解。我们真正关心的是投影矢量$\mathbf{w}$的方向，而不关心它的长度（可以规格化为单位矢量），因此可以通过适当调整$\mathbf{w}$使得Fisher准则的分母$\mathbf{w}^t S_w \mathbf{w}$等于一个常数$C$。这样我们就得到了一个有约束的优化问题：

$$\max_{\mathbf{w}} \mathbf{w}^t S_b \mathbf{w} \tag{8.42}$$

$$\text{subject to} \tag{8.43}$$

$$\mathbf{w}^t S_w \mathbf{w} = C \tag{8.44}$$

构造Lagrange函数转化为无约束优化：

$$L(\mathbf{w}, \lambda) = \mathbf{w}^t S_b \mathbf{w} - \lambda(\mathbf{w}^t S_w \mathbf{w} - C) \tag{8.45}$$

对$\mathbf{w}$求导及极值：

$$\frac{\partial L(\mathbf{w},\lambda)}{\partial \mathbf{w}} = 2S_b\mathbf{w} - 2\lambda S_w\mathbf{w} = \mathbf{0} \tag{8.46}$$

因此有：

$$S_b\mathbf{w} = \lambda S_w\mathbf{w} \tag{8.47}$$

满足上式的λ和$\mathbf{w}$称为关于S_b和S_w的广义特征值和特征矢量，如果S_w的逆矩阵存在的话，有：

$$S_w^{-1}S_b\mathbf{w} = \lambda\mathbf{w} \tag{8.48}$$

λ和$\mathbf{w}$分别是矩阵$S_w^{-1}S_b$的特征值和对应的特征矢量。与主成分分析一样，$S_w^{-1}S_b$是一个$d \times d$的方阵，存在d个特征值和d个特征矢量，哪一个特征矢量使得Fisher 准则取得最大值？将满足公式(8.47)的一个特征矢量$\mathbf{w}_i$代入(8.41)：

$$\frac{\mathbf{w}_i^tS_b\mathbf{w}_i}{\mathbf{w}_i^tS_w\mathbf{w}_i} = \frac{\lambda_i\mathbf{w}_i^tS_w\mathbf{w}_i}{\mathbf{w}_i^tS_w\mathbf{w}_i} = \lambda_i \tag{8.49}$$

由此可见，最大特征值对应的特征矢量是使得Fisher准则取得最大值的方向矢量。通过这样一个推导过程我们可以得出如下结论：两个类别的样本向一条直线上投影，当直线的方向矢量$\mathbf{w}$为矩阵$S_w^{-1}S_b$最大特征值对应的特征矢量时，可以使得投影之后样本在1维空间中具有最大的可分性。

下面将这个问题推广到c个类别的样本向d'个方向上的投影，即将d维特征降维为d'个特征，使得降维之后的样本具有最大的可分性。首先定义矩阵：

$$S_w = \sum_{i=1}^{c} S_i, \quad S_b = \sum_{i=1}^{c} n_i(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^t \tag{8.50}$$

其中$\boldsymbol{\mu}$为所有样本的均值矢量。可以证明使得Fisher准则最大的d'个投影矢量是对应于矩阵$S_w^{-1}S_b$最大d'个特征值的特征矢量。

8.4　KPCA 方法的推导

首先证明样本集合协方差矩阵的特征向量处于样本所张成的空间。

定理：令Σ为样本集合$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$的协方差矩阵，$\mathbf{v}$为矩阵$\Sigma$的一个特征矢量。$\mathbf{v}$处于由样本集$D$所张成的空间，即：$\mathbf{v} \in span\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$。

证明：为了方便起见，令D为中心化的样本集，即：

$$\sum_{i=1}^{n} \mathbf{x}_i = \mathbf{0} \tag{8.51}$$

样本集合D的协方差矩阵：

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^t \tag{8.52}$$

令λ为Σ的特征值，$\mathbf{v}$为对应特征矢量，则有：

$$\lambda\mathbf{v} = \Sigma\mathbf{v} \tag{8.53}$$

$$= \left(\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^t\right)\mathbf{v} \tag{8.54}$$

$$= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^t\mathbf{v})\mathbf{x}_i \tag{8.55}$$

即：

$$\mathbf{v} = \frac{1}{m\lambda}\sum_{i=1}^{n}(\mathbf{x}_i^t\mathbf{v})\mathbf{x}_i \tag{8.56}$$

因此：$\mathbf{v} \in span\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$，定理得证。■

下面来推导特征空间中协方差矩阵特征值和特征向量的求解方法。设有样本集合：$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$，取非线性映射：$\phi : R^d \to \mathcal{F}$，$R^d$为样本所处的$d$维欧氏空间，称为输入空间，$\mathcal{F}$为一个Hilbert空间，称为特征空间，样本在特征空间中的内积可以用一个核函数来计算：

$$\kappa(\mathbf{x}, \mathbf{y}) = \phi^t(\mathbf{x})\phi(\mathbf{y}) \tag{8.57}$$

假设样本集合D在特征空间中的均值为$\mathbf{0}$，即：

$$\sum_{i=1}^{n}\phi(\mathbf{x}_i) = \mathbf{0} \tag{8.58}$$

样本在特征空间中的协方差矩阵为：

$$\tilde{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}\phi(\mathbf{x}_i)\phi^t(\mathbf{x}_i) \tag{8.59}$$

令$\tilde{\lambda}$为$\tilde{\Sigma}$的特征值，$\tilde{\mathbf{v}}$为对应的特征矢量，则有：

$$\tilde{\lambda}\tilde{\mathbf{v}} = \tilde{\Sigma}\tilde{\mathbf{v}} \tag{8.60}$$

根据上述定理，特征矢量$\tilde{\mathbf{v}} \in span\{\phi(\mathbf{x}_1, \cdots, \phi(\mathbf{x}_n)\}$，即存在一组系数$\alpha_1, \cdots, \alpha_n$，使得：

$$\tilde{\mathbf{v}} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i) \tag{8.61}$$

公式(8.60)左乘$\phi^t(\mathbf{x}_j)$，则有：

$$\lambda \left[\phi^t(\mathbf{x}_j)\tilde{\mathbf{v}}\right] = \phi^t(\mathbf{x}_j)\tilde{\Sigma}\tilde{\mathbf{v}}, \quad j = 1, \cdots, n \tag{8.62}$$

将公式(8.61)代入(8.62)的左端：

$$\lambda \left[\phi^t(\mathbf{x}_j)\tilde{\mathbf{v}}\right] = \lambda \sum_{i=1}^{n} \alpha_i \phi^t(\mathbf{x}_j)\phi(\mathbf{x}_i) \tag{8.63}$$

$$= \lambda \sum_{i=1}^{n} \alpha_i \kappa(\mathbf{x}_j, \mathbf{x}_i) \tag{8.64}$$

$$= \lambda (K\boldsymbol{\alpha})_j \tag{8.65}$$

其中，K为核矩阵：

$$K = \begin{pmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_n, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \tag{8.66}$$

$\boldsymbol{\alpha}$为组合系数矢量：

$$\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n)^t \tag{8.67}$$

$(K\boldsymbol{\alpha})_j$表示矢量$K\boldsymbol{\alpha}$的第j个元素。

将(8.59)和(8.61)代入公式(8.62)的右端：

$$\phi^(\mathbf{x}_j)\tilde{\Sigma}\tilde{\mathbf{v}} = \phi^t(\mathbf{x}_j) \cdot \frac{1}{n} \sum_{l=1}^{n} \phi(\mathbf{x}_l)\phi^t(\mathbf{x}_l) \cdot \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i) \tag{8.68}$$

$$= \frac{1}{n} \sum_{l=1}^{n} \sum_{i=1}^{n} \alpha_i \left[\phi^t(\mathbf{x}_j)\phi(\mathbf{x}_l) \cdot \phi^t(\mathbf{x}_l)\phi(\mathbf{x}_i)\right] \tag{8.69}$$

$$= \frac{1}{n} \sum_{l=1}^{n} \sum_{i=1}^{n} \alpha_i \kappa(\mathbf{x}_j, \mathbf{x}_l)\kappa(\mathbf{x}_l, \mathbf{x}_i) \tag{8.70}$$

$$= \frac{1}{n} \left(K^2\boldsymbol{\alpha}\right)_j \tag{8.71}$$

(8.65)和(8.71)相等，则有：

$$\lambda K\boldsymbol{\alpha} = \frac{1}{n} K^2\boldsymbol{\alpha} \tag{8.72}$$

因此，求取$\tilde{\Sigma}$的特征值和特征向量的问题可以转化为如下特征值问题：

$$m\lambda\boldsymbol{\alpha} = K\boldsymbol{\alpha} \tag{8.73}$$

用λ代替$m\lambda$，即为：

$$\lambda\boldsymbol{\alpha} = K\boldsymbol{\alpha} \tag{8.74}$$

令：$\lambda^1, \cdots, \lambda^n$为矩阵$K$的特征值，$\boldsymbol{\alpha}^1, \cdots, \boldsymbol{\alpha}^n$为特征矢量，则对应的$\tilde{\Sigma}$的第$i$个特征值和特征矢量为：

$$\lambda_i = \frac{\lambda^i}{n}, \qquad \tilde{\mathbf{v}}_i = \sum_{j=1}^{n} \alpha_j^i \phi(\mathbf{x}_j) \tag{8.75}$$

其中α_j^i为$\boldsymbol{\alpha}^i$的第j个元素，将$\tilde{\mathbf{v}}_i$规范化为单位矢量：

$$\tilde{\mathbf{v}}_i^t \tilde{\mathbf{v}}_i = \sum_{j=1}^{n}\sum_{l=1}^{n} \alpha_j^i \alpha_l^i \phi^t(\mathbf{x}_j)\phi(\mathbf{x}_l) \tag{8.76}$$

$$= \sum_{j=1}^{n}\sum_{l=1}^{n} \alpha_j^i \alpha_l^i \kappa(\mathbf{x}_j, \mathbf{x}_l) \tag{8.77}$$

$$= \left(\boldsymbol{\alpha}^i\right)^t K\boldsymbol{\alpha}^i \tag{8.78}$$

$$= \lambda^i \left(\boldsymbol{\alpha}^i\right)^t \boldsymbol{\alpha}^i \tag{8.79}$$

$$= \lambda^i \|\boldsymbol{\alpha}^i\|^2 \tag{8.80}$$

因此只须使：

$$\|\boldsymbol{\alpha}^i\|^2 = \frac{1}{\lambda^i} \tag{8.81}$$

即可使$\tilde{\mathbf{v}}^i$规范化。计算样本$\mathbf{x}$在特征空间中第j个轴上的投影，利用公式(8.75)，有：

$$\phi^t(\mathbf{x})\tilde{\mathbf{v}}_j = \sum_{i=1}^{n} \alpha_i^j \phi^t(\mathbf{x})\phi(\mathbf{x}_i) \tag{8.82}$$

$$= \sum_{i=1}^{n} \alpha_i^j \kappa(\mathbf{x}, \mathbf{x}_i) \tag{8.83}$$

KPCA算法：

1. 计算核矩阵K：$k_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j), \quad i, j = 1, \cdots, n$

2. 计算K的特征值和特征向量：$\lambda^1, \cdots, \lambda^n$，$\boldsymbol{\alpha}^1, \cdots, \boldsymbol{\alpha}^n$

3. 取最大的前d'个特征值：$\lambda^1, \cdots, \lambda^{d'}$

4. 规范化特征向量，使得：$\|\boldsymbol{\alpha}^i\|^2 = 1/\lambda^i$

5. 由(8.83)式计算样本$\mathbf{x}$的第j个核主成分。

附录：最优化方法

在模式识别的学习和训练过程中，需要根据训练样本来确定一组与分类器模型相关的参数。学习过程往往首先定义某个准则函数，用以描述参数的“适合性”，然后寻找一组“适合性”最大的参数作为学习的结果，也就是说将模式识别的学习问题转化为针对某个准则函数的优化问题。

假设准则函数$f(\mathbf{x})$针对$\mathbf{x}$的每个分量都可微，下面介绍几种常用的函数优化方法。

梯度法

根据数学分析的知识我们知道，m维矢量$\mathbf{x}^*$是函数$f(\mathbf{x})$极值点的必要条件是：$\partial f/\partial x_i^* = 0$，对任意的$1 \leq i \leq m$成立。如果把所有的偏导数写成矢量的形式，则函数$f(\mathbf{x})$的极值点可以通过求解矢量方程得到：

$$\frac{f(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \partial f/\partial x_1 \\ \vdots \\ \partial f/\partial x_m \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} = \mathbf{0} \tag{1}$$

上述方程的解可能是极大值点，可能是极小值点，也可能不是极值点，具体情况还需要根据二阶导数来判断。如果我们希望求取的是$f(\mathbf{x})$的最大值或最小值点，可以通过比较所有的极大值或极小值点得到。

对于简单的纯凸或纯凹函数（如二次函数），由于只存在唯一的极值点，极值点即为最大值或最小值点，因此可以直接通过求解矢量方程(1)得到$f(\mathbf{x})$ 的优化解。

对于复杂的函数来说，直接求解方程(1)得到优化函数的极值点往往是很困难的。在这种情况下，可以考虑采用迭代的方法从某个初始值开始逐渐逼近极值点，梯度法就是一种迭代求解函数极值点的方法。

考虑多元函数$f(\mathbf{x})$在点$\mathbf{x}$附近的一阶泰勒级数展开式：

$$f(\mathbf{x}+\Delta\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{m} \frac{\partial f}{\partial x_i}\Delta x_i + r(\mathbf{x},\Delta\mathbf{x}) \tag{2}$$

其中$\Delta\mathbf{x}$是增量矢量，Δx_i为其第i维元素，$r(\mathbf{x},\Delta\mathbf{x})$为展开式的余项。注意到第二项的求和式，实际上是$f(\mathbf{x}$关于$\mathbf{x}$的梯度矢量与$\Delta\mathbf{x}$之间的内积，同时当$\|\mathbf{x}\|$很小时忽略余项$r(\mathbf{x},\Delta\mathbf{x})$，可以得到一阶近似展开式：

$$f(\mathbf{x}+\Delta\mathbf{x}) \approx f(\mathbf{x}) + (\nabla f(\mathbf{x}))^t \Delta\mathbf{x} = f(\mathbf{x}) + \left(\frac{\partial f}{\partial \mathbf{x}}\right)^t \Delta\mathbf{x} \tag{3}$$

如果我们要求取$f(\mathbf{x})$的极小值$\mathbf{x}^*$，可以从某个初始点$\mathbf{x}_0$开始搜索，每次增加一个增量$\Delta\mathbf{x}$。虽然不能保证$\mathbf{x}+\Delta\mathbf{x}$直接到达极小值点，但如果能够保证每次迭代过程中函数值逐渐减小，$f(\mathbf{x}+\Delta\mathbf{x}) < f(\mathbf{x})$，那么经过一定的迭代步数之后，就能够逐渐地接近$\mathbf{x}^*$，这是一个函数值逐渐下降的过程。更进一步，我们总是希望函数值下降的过程越快越好，这样可以用尽量少的迭代次数，达到对$\mathbf{x}^*$更高精度的逼近，因此这种方法也被称作“最速下降法”。

根据公式(3)，要使得函数值下降得最快，就是要寻找一个增量矢量$\Delta\mathbf{x}$使得$(\nabla f(\mathbf{x}))^t\Delta\mathbf{x}$最小。注意到公式(3)只是在点$\mathbf{x}$附近的一阶近似，当$\|\Delta\mathbf{x}\|$过大时，近似的精度会很差，因此不能直接寻找增量矢量，而是应该寻找使得函数值下降最快的方向。也就是在约束$\|\Delta\mathbf{x}\|=1$的条件下，寻找使得$(\nabla f(\mathbf{x}))^t\Delta\mathbf{x}$最小的增量矢量。找到最速下降的方向之后，再来确定此方向上合适的增量矢量长度。

根据Cauchy - Schwarz不等式，两个矢量内积的绝对值小于等于两个矢量长度的乘积：

$$\left|(\nabla f(\mathbf{x}))^t\Delta\mathbf{x}\right| \le \|\nabla f(\mathbf{x})\| \cdot \|\Delta\mathbf{x}\| \tag{4}$$

有：

$$(\nabla f(\mathbf{x}))^t\Delta\mathbf{x} \ge -\|\nabla f(\mathbf{x})\| \cdot \|\Delta\mathbf{x}\| = -\|\nabla f(\mathbf{x})\| \tag{5}$$

如果令：$\Delta\mathbf{x} = -\nabla f(\mathbf{x})/\|\nabla f(\mathbf{x})\|$，则有：

$$(\nabla f(\mathbf{x}))^t \Delta\mathbf{x} = (\nabla f(\mathbf{x}))^t \left[-\frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}\right] \tag{6}$$

$$= -\frac{(\nabla f(\mathbf{x}))^t \Delta f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|} \tag{7}$$

$$= -\frac{\|\nabla f(\mathbf{x})\|^2}{\|\nabla f(\mathbf{x})\|} \tag{8}$$

$$= -\|\nabla f(\mathbf{x})\| \tag{9}$$

由此我们知道，当$\Delta\mathbf{x}$为负的梯度方向时，不等式(5)中的等号成立，也就是说$(\nabla f(\mathbf{x}))^t\Delta\mathbf{x}$取得最小值。因此，增量矢量的方向为负的梯度方向时，函数值下降得最快。最速下降法中每一轮应该按照下式进行迭代：

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x} = \mathbf{x} - \eta\nabla f(\mathbf{x}) \tag{10}$$

其中参数η控制增量矢量的长度，在模式识别的算法中一般被称作“学习率”。与此类似，如果优化问题需要寻找的是极大值点，每次迭代中增量矢量应该沿着正的梯度方向。

梯度下降算法的过程是从一个随机的初始点$\mathbf{x}_0$开始，每一轮迭代中计算当前点处的梯度矢量，然后根据公式(10)修正当前的优化点$\mathbf{x}$。由于极值点处的梯度是$\mathbf{0}$矢量，因此算法的收敛条件是判断当前点处梯度矢量的长度是否足够小，当达到一定的收敛精度后可以停止迭代。

Algorithm 1 梯度下降算法

Input: 优化函数$f(\mathbf{x})$，学习率η，收敛精度θ
Output: 函数$f(\mathbf{x})$的极小值点$\mathbf{x}^*$

```
procedure GradientDescent
    随机初始化x_0，i = 0；
    repeat
        计算当前点x_i的梯度矢量：∇f(x)|_{x=x_i}
        更新优化解：x_{i+1} = x_i − η∇f(x)|_{x=x_i}
        i ← i + 1
    until ||η∇f(x)|_{x=x_i}|| < θ
return x* = x_i
end procedure
```

参数θ为收敛精度，值越小输出的解越接近于极小值点。梯度下降算法的优点是：算法简单，只要能够计算任意一点的梯度矢量就可以进行迭代优化；在设置合适的学习率η的条件下，算法具有收敛性，能够收敛于一个极小值点。

同样，梯度下降算法也存在着自身的缺点。首先是收敛速度慢，特别是在一些梯度值较小的区域表现得尤为明显；收敛性依赖于适合的学习率η的设置，而与初始点$\mathbf{x}_0$的选择无关，但对于一个具体问题来说还没有能够直接确定η的方法，一般需要进行一定的尝试；梯度法只能够保证收敛于一个极值点，而无法一次计算出所有的极值点，具体收敛于哪一个极值点决定于初始点$\mathbf{x}_0$；同时，算法收敛的极值点不能保证是优化函数的最小值点，往往需要进行多次尝试，从得到的多个极值点中找出一个最小值点，但由于尝试的次数有限，因此也不能保证找到优化函数的最小值点。

牛顿法

采用梯度法对一个复杂的函数进行优化，迭代的收敛速度往往很慢。这主要是由于，梯度法是利用一阶泰勒级数展开式在$\mathbf{x}$附近用一个线性函数来近似优化$f(\mathbf{x})$，当$f(\mathbf{x})$是一个复杂的非线性函数时，近似的精度很低。二阶技术就是用二次函数来近似优化函数，降低近似误差，从而达到提高优化迭代效率的目的。下面，首先来看一下函数的二阶泰勒级数展开式：

$$f(\mathbf{x}+\Delta\mathbf{x}) \approx f(\mathbf{x}) + \left(\frac{\partial f}{\partial \mathbf{x}}\right)^t \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^t H \Delta\mathbf{x} \tag{11}$$

其中H是f的二阶导数Hessian矩阵：

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_m} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_m \partial x_1} & \frac{\partial^2 f}{\partial x_m \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_m \partial x_m} \end{pmatrix} \tag{12}$$

为了寻找使得$f(\mathbf{x}+\Delta\mathbf{x})$最小的权值增量$\Delta\mathbf{x}$，公式(11)对$\Delta\mathbf{x}$微分求取极值点，同时考虑到$H$是对称矩阵：

$$\frac{\partial f}{\partial \mathbf{x}} + H\Delta\mathbf{x} = \mathbf{0} \quad \Rightarrow \quad \Delta\mathbf{x} = -H^{-1}\left(\frac{\partial f}{\partial \mathbf{x}}\right) \tag{13}$$

这样，我们就得到了在二阶泰勒级数近似条件下的最优权值增量，此方法一般被称为牛顿法。

牛顿法虽然形式上很简单，但在实际问题中往往无法直接应用。首先，当矢量$\mathbf{x}$的维数m很大时，$m \times m$的Hessian矩阵H无论是计算、存储还是求逆的复杂度都很高；更严重的问题是函数f并不是一个二次函数，而牛顿法是建立在二阶近似基础之上的，这就导致了直接使用牛顿法并不能保证算法的收敛。牛顿法虽然无法直接使用，但受此启发人们提出了多种近似的二阶优化技术。

拟牛顿法

在牛顿法中存在着计算函数$f(\mathbf{x})$的二阶导数矩阵H的困难，拟牛顿法利用函数的一阶导数（梯度）来近似递推矩阵H。

类似于公式(11)，函数在$\mathbf{x}_{k+1}$附近的二阶泰勒级数展开为：

$$f(\mathbf{x}) \approx f(\mathbf{x}_{k+1}) + \mathbf{g}_{k+1}^t(\mathbf{x}-\mathbf{x}_{k+1}) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_{k+1})^t H_{k+1}(\mathbf{x}-\mathbf{x}_{k+1}) \quad (14)$$

其中$\mathbf{g}_{k+1} = (\partial f/\partial \mathbf{x})_{\mathbf{x}=\mathbf{x}_{k+1}}$是函数$f(\mathbf{x})$在$\mathbf{x}_{k+1}$处的梯度，$H_{k+1}$是函数在$\mathbf{x}_{k+1}$处的Hessian矩阵。公式(14)两边对$\mathbf{x}$求导：

$$\mathbf{g}(\mathbf{x}) = \frac{\partial f}{\partial \mathbf{x}} \approx \mathbf{g}_{k+1} + H_{k+1}(\mathbf{x}-\mathbf{x}_{k+1}) \quad (15)$$

将$\mathbf{x} = \mathbf{x}_k$代入，为了表示方便，令$\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$，$\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$，可得到拟牛顿方程：

$$H_{k+1}\mathbf{s}_k = \mathbf{y}_k \quad (16)$$

拟牛顿法的关键是要用上一步的Hessian矩阵H_k来递推当前的H_{k+1}，因此需要对H_k进行校正。假设H_{k+1}可以通过对H_k的秩二校正得到，即：

$$H_{k+1} = H_k + a\mathbf{u}\mathbf{u}^t + b\mathbf{v}\mathbf{v}^t \quad (17)$$

将公式(17)代入拟牛顿方程(16)，则有：

$$H_{k+1}\mathbf{s}_k = H_k\mathbf{s}_k + a\mathbf{u}\mathbf{u}^t\mathbf{s}_k + b\mathbf{v}\mathbf{v}^t\mathbf{s}_k = \mathbf{y}_k \quad (18)$$

满足等式(18)式的$\mathbf{u}, \mathbf{v}$和a, b是不唯一的，可以选择：$\mathbf{u} = \mathbf{y}_k, \mathbf{v} = H_k\mathbf{s}_k$，代入(18)：

$$\mathbf{v} + a\mathbf{u}\mathbf{u}^t\mathbf{s}_k + b\mathbf{v}\mathbf{v}^t\mathbf{s}_k - \mathbf{u} = (a\mathbf{u}^t\mathbf{s}_k - 1)\mathbf{u} + (b\mathbf{v}^t\mathbf{s}_k + 1)\mathbf{v} = \mathbf{0} \quad (19)$$

当$\mathbf{u}$和$\mathbf{v}$不为$\mathbf{0}$矢量时有：

$$a\mathbf{u}^t\mathbf{s}_k = 1, \qquad b\mathbf{v}^t\mathbf{s}_k = -1 \tag{20}$$

因此：

$$a = \frac{1}{\mathbf{u}^t\mathbf{s}_k} = \frac{1}{\mathbf{y}_k^t\mathbf{s}_k}, \quad b = -\frac{1}{\mathbf{v}^t\mathbf{s}_k} = -\frac{1}{\mathbf{s}_k^tH_k\mathbf{s}_k} \tag{21}$$

代入公式(17)，得到递推公式：

$$H_{k+1} = H_k + \frac{\mathbf{y}_k\mathbf{y}_k^t}{\mathbf{y}_k^t\mathbf{s}_k} - \frac{H_k\mathbf{s}_k\mathbf{s}_k^tH_k}{\mathbf{s}_k^tH_k\mathbf{s}_k} \tag{22}$$

这样，我们就可以由上一步的Hessian矩阵H_k、位置差$\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$以及梯度差$\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$来近似地递推计算当前的Hessian矩阵$H_{k+1}$，而不需要计算函数$f(\mathbf{x})$的二阶导数。在牛顿法中需要计算的是Hessian矩阵的逆矩阵（见公式(13)），而由公式(22)迭代得到的近似矩阵存在奇异阵的可能性，同时逆矩阵的计算也相对比较复杂，因此常用的方法是直接递推Hessian矩阵的逆矩阵。由公式(22)进一步的推导可以得到逆矩阵的递推公式：

$$H_{k+1}^{-1} = \left(I - \frac{\mathbf{s}_k\mathbf{y}_k^t}{\mathbf{s}_k^t\mathbf{y}_k}\right)H_k^{-1}\left(I - \frac{\mathbf{y}_k\mathbf{s}_k^t}{\mathbf{y}_k^t\mathbf{s}_k}\right) + \frac{\mathbf{s}_k\mathbf{s}_k^t}{\mathbf{s}_k^t\mathbf{s}_k} \tag{23}$$

(22)和(23)一般称为BFGS 拟牛顿法递推公式。

Algorithm 2 BFGS拟牛顿算法

Input: 优化函数$f(\mathbf{x})$，收敛精度θ

Output: 函数$f(\mathbf{x})$的极小值点$\mathbf{x}^*$

procedure QuasiNewton
    随机初始化$\mathbf{x}_0$，$H_0 = I$，$k = 0$；
    计算$\mathbf{x}_0$处的梯度矢量：$\mathbf{g}_0 = \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0}$
    repeat
        计算优化方向：$\mathbf{d}_k = -H_k^{-1}\mathbf{g}_k$
        沿方向$\mathbf{d}_k$进行1维搜索，使得$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k d_k$为此方向上的极小值点，$\alpha_k > 0$；
        计算$\mathbf{x}_{k+1}$处的梯度矢量：$\mathbf{g}_{k+1} = \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_{k+1}}$
        公式(23)递推计算H_{k+1}^{-1}；
        $k = k + 1$
    until $\mathbf{g}_{k+1} < \theta$
return $\mathbf{x}^* = \mathbf{x}_k$
end procedure

共轭梯度法

拟牛顿法解决了二阶导数矩阵H的计算问题，但当矢量$\mathbf{x}$的维数m较大时，需要的存储量和矩阵乘法的计算量仍然较大。共轭梯度法也是一种近似的二阶方法，只需计算一阶导数，不需要计算和存储二阶导数矩阵。

对于$m \times m$矩阵H来说，如果对于任意的两个矢量$\mathbf{d}_i, \mathbf{d}_j$满足$\mathbf{d}_i^t H \mathbf{d}_j$，其中$i \neq j$，$i, j = 0, 1, \cdots, m-1$，则称$\mathbf{d}_0, \cdots, \mathbf{d}_{m-1}$是关于矩阵$H$的一组共轭矢量。显然，当$H$为单位矩阵$I$时，这组共轭矢量之间是相互正交的。

下面来看一下优化函数$f(\mathbf{x})$为二次正定函数的情况：$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t H \mathbf{x} + \mathbf{u}^t \mathbf{x} + c$，其中$H$为$n \times n$ 的正定对称矩阵。我们先来证明这样一个事实：如果采用共轭方向算法从任意的起始点$\mathbf{x}_0$出发，每一轮迭代都是沿着矩阵H的一组共轭矢量方向$\mathbf{d}_0, \cdots, \mathbf{d}_{m-1}$做1维搜索，找到在一个共轭方向上的最小值点，那么只需要m轮迭代就可以找到函数$f(\mathbf{x})$的最小值点。

共轭方向算法：

1. 初始化起始点$\mathbf{x}_0$，一组关于矩阵H的共轭矢量$\mathbf{d}_0, \cdots, \mathbf{d}_{m-1}$，$k = 0$；

2. 计算α_k和$\mathbf{x}_{k+1}$，使得：

$$f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) = \min_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{d}_k) \tag{24}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k \tag{25}$$

3. 转到2，直到$k = m - 1$为止。

上述事实可以由如下定理的证明得到：

定理：对于正定二次优化函数$f(\mathbf{x})$，如果按照共轭方向进行搜索，至多经过m步精确的线性搜索可以终止；并且每一个$\mathbf{x}_{i+1}$都是在$\mathbf{x}_0$和方向$\mathbf{d}_0, \cdots, \mathbf{d}_i$所张成的线性流形$\left\{\mathbf{x} | \mathbf{x} = \mathbf{x}_0 + \sum_{j=0}^{i} \alpha_j \mathbf{d}_j\right\}$ 中的极值点。

证明：令$\mathbf{g}_i$为第i步的梯度，即：$\mathbf{g}_i = \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_i}$。上述定理实际上只需证明，对$\forall j \leq i$，$\mathbf{g}_{i+1}^t \mathbf{d}_j = 0$即可。因为$\mathbf{g}_{i+1}$正交于$\mathbf{d}_0, \cdots, \mathbf{d}_i$，则$\mathbf{g}_{i+1}$正交于它们所张成的线性流形，$\mathbf{x} = \mathbf{x}_0 + \sum_{j=0}^{i} \alpha_j \mathbf{d}_j$包含在此线性流形中，因此在此线性流形中$f(\mathbf{x})$的梯度为$\mathbf{0}$，即$\mathbf{x}_{i+1}$为在线性流形上的极值点。当$i + 1 = m$时，$\mathbf{d}_0, \cdots, \mathbf{d}_{m-1}$所张成的线性流形即为整个$m$维空间$R^m$，只有当$\mathbf{g}_m = 0$时，才有$\mathbf{g}_m^t \mathbf{d}_j = 0$成立，因此$\mathbf{x}_m$为极值点。

梯度$\mathbf{g}=\nabla f(\mathbf{x})=H\mathbf{x}+\mathbf{u}$，因此两次迭代之间梯度的差值矢量为：

$$\mathbf{g}_{k+1}-\mathbf{g}_k=H(\mathbf{x}_{k+1}-\mathbf{x}_k)=\alpha_k H\mathbf{d}_k \tag{26}$$

对于$\forall j<i$：

$$\mathbf{g}_{i+1}^t\mathbf{d}_j=\mathbf{g}_{i+1}^t\mathbf{d}_j-\mathbf{g}_i^t\mathbf{d}_j+\mathbf{g}_i^t\mathbf{d}_j-\mathbf{g}_{i-1}^t\mathbf{d}_j+\cdots+\mathbf{g}_{j+1}^t\mathbf{d}_j \tag{27}$$

$$=\mathbf{g}_{i+1}^t\mathbf{d}_j+\sum_{k=j+1}^{i}(\mathbf{g}_{k+1}-\mathbf{g}_k)^t\mathbf{d}_j \tag{28}$$

$$=\mathbf{g}_{i+1}^t\mathbf{d}_j+\sum_{k=j+1}^{i}\alpha_k\mathbf{d}_k^tH\mathbf{d}_j \tag{29}$$

$\mathbf{x}$是沿着$\mathbf{d}_j$方向搜索的极值点，因此$\mathbf{g}_{j+1}^t\mathbf{d}_j=0$，而$\mathbf{d}_0,\cdots,\mathbf{d}_i$互为共轭，所以有$\sum_{k=j+1}^{i}\alpha_k\mathbf{d}_k^tH\mathbf{d}_j=0$，因此：

$$\mathbf{g}_{i+1}^t\mathbf{d}_j=0 \tag{30}$$

上述定理得证。■

由此可以看出，当优化函数$f(\mathbf{x})$是二次函数时，共轭方向法只需经过m轮迭代就可以收敛于函数的最小值点。如果函数$f(\mathbf{x})$不是二次函数，我们仍然可以在每一轮迭代中沿着Hessian矩阵H的共轭方向搜索到极小值点，只不过不能保证算法经过m轮迭代收敛，一般需要更多的迭代次数。

使用这种方法还需要解决的一个问题是，如何得到关于优化函数$f(\mathbf{x})$的Hessian矩阵H的一组共轭方向矢量，而不需要计算出矩阵H。实际上任意给定一个初始的方向$\mathbf{d}_0$就可以确定一组关于矩阵H的共轭方向矢量，在共轭梯度法中一般选择初始点$\mathbf{x}_0$处的负梯度方向作为初始方向：

$$\mathbf{d}_0=-\mathbf{g}_0=-\left(\frac{\partial f}{\partial\mathbf{x}}\right)_{\mathbf{x}=\mathbf{x}_0} \tag{31}$$

而第$k+1$步的共轭方向矢量$\mathbf{d}_{k+1}$，由第$k+1$步的负梯度方向$-\mathbf{g}_{k+1}$与第k步的共轭方向矢量$\mathbf{d}_k$的线性组合得到：

$$\mathbf{d}_{k+1}=-\mathbf{g}_{k+1}+\beta_k\mathbf{d}_k \tag{32}$$

组合系数β_k的确定有多种方式，常用的包括：

Crowder-Wolfe公式：
$$\beta_k = \frac{\mathbf{g}_{k+1}^t(\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{d}_k^t(\mathbf{g}_{k+1} - \mathbf{g}_k)} \tag{33}$$

Fletcher-Reeves 公式：
$$\beta_k = \frac{\mathbf{g}_{k+1}^t\mathbf{g}_{k+1}}{\mathbf{g}_k^t\mathbf{g}_k)} \tag{34}$$

Polak-Ribiere-Polyak 公式：
$$\beta_k = \frac{\mathbf{g}_{k+1}^t(\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{g}_k^t\mathbf{g}_k} \tag{35}$$

Algorithm 3 共轭梯度算法

Input: 优化函数$f(\mathbf{x})$，收敛精度θ

Output: 函数$f(\mathbf{x})$的极小值点$\mathbf{x}^*$

procedure ConjugateDescent
　随机初始化$\mathbf{x}_0$，$k = 0$；
　计算$\mathbf{x}_0$处的负梯度矢量作为初始的搜索方向：$\mathbf{d}_0 = -\mathbf{g}_0 = -\nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0}$
　repeat
　　沿着共轭方向$\mathbf{d}_k$搜索局部最小值点，即求解优化问题：

$$f(\mathbf{x}_k + \alpha_k\mathbf{f}_k) = \min_{\alpha} f(\mathbf{x}_k + \alpha\mathbf{d}_k)$$

　　更新优化解：$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\mathbf{d}_k$
　　计算$\mathbf{x}_{k+1}$处的梯度矢量：$\mathbf{g}_{k+1} = \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_{k+1}}$
　　由公式(31)以及(33–35)中的一个计算下一个搜索方向$\mathbf{d}_{k+1}$；
　　$k \leftarrow k + 1$
　until $\|\mathbf{g}_k\| < \theta$
return $\mathbf{x}^* = \mathbf{x}_k$
end procedure

约束优化

前面介绍的几种方法，解决的都是直接针对函数$f(\mathbf{x}$的优化问题，解矢量$\mathbf{x}^*$可以是R^m空间中的任意矢量，一般称为“无约束优化”。在模式识别中，经常还会遇到另外一类“约束优化”问题，要求解矢量满足一定的约

束条件。约束优化问题的一般形式可以表示为：

$$\min_{\mathbf{x}\in R^m} f(\mathbf{x}) \tag{36}$$

subject to

$$c_i(\mathbf{x}) = 0, \quad i = 1, \cdots, l$$
$$c_i(\mathbf{x}) \leq 0, \quad i = l+1, \cdots, k$$

其中前l个称为等式约束，后$k-l$个称为不等式约束。对于约束优化问题的严格证明比较复杂，需要的数学知识超出了本书的范畴，下面简单介绍两种课程中需要用到的约束问题求解方法。

线性等式约束优化问题

在这类问题中只包含等式约束，并且函数$c_i(\mathbf{x})$均为线性函数。先看一个二维矢量和一个线性约束的简单例子：

$$\min_{\mathbf{x}\in R^2} x_1^2 + x_2^2 \tag{37}$$

subject to

$$\mathbf{a}^t\mathbf{x} = b$$

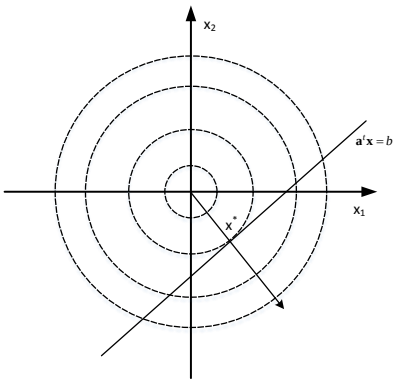

图 8.4: 极值点处优化函数的梯度与线性约束的权值矢量共线

图8.4中虚线表示的是优化函数$f(\mathbf{x})$的等值线，而实线是满足约束条件的点所在的直线，优化问题的解$\mathbf{x}^*$应该在这条直线上。显然这条直线上函

数值最小的点处于直线与$f(\mathbf{x})$等值线相切的位置，即在$\mathbf{x}^*$处优化函数的梯度矢量$f(\mathbf{x}^*)$与直线正交，与直线的权值矢量$\mathbf{a}$共线，因此$\mathbf{x}^*$是如下方程组的解：

$$\begin{cases} \nabla f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \lambda \mathbf{a} \\ \mathbf{a}^t \mathbf{x} = b \end{cases} \tag{38}$$

如果我们构造一个函数：

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda(\mathbf{a}^t \mathbf{x} - b) \tag{39}$$

那么就会发现：

$$\frac{\partial L(\mathbf{x}, \lambda)}{\partial \mathbf{x}} = \frac{f(\mathbf{x})}{\partial \mathbf{x}} - \lambda \mathbf{a} = \mathbf{0} \tag{40}$$

$$\frac{\partial L(\mathbf{x}, \lambda)}{\partial \lambda} = -\mathbf{a}^t \mathbf{x} + b = 0 \tag{41}$$

刚好得到了公式(38)的方程组。换句话说就是，公式(37)的约束优化问题可以转化为求解函数$L(\mathbf{x}, \lambda)$的无约束优化极值问题。函数$L(\mathbf{x}, \lambda)$称为Lagrange函数，而λ称为Lagrange系数。

对于m维矢量的优化，以及多个线性等式约束的问题也有同样的结果。将所有的线性约束写成矩阵形式：

$$\min_{\mathbf{x} \in R^m} f(\mathbf{x}) \tag{42}$$

subject to

$$A\mathbf{x} = \mathbf{b}$$

可以证明，上述约束优化问题可以通过构造Lagrange函数转化为无约束问题求解：

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\lambda}^t (A\mathbf{x} - \mathbf{b}) = f(\mathbf{x}) - \sum_{i=1}^{l} \lambda_i (\mathbf{a}_i \mathbf{x} - b_i) \tag{43}$$

Lagrange函数的极值点满足方程：

$$\begin{cases} \frac{L(\mathbf{x}, \boldsymbol{\lambda})}{\partial \mathbf{x}} = \frac{f(\mathbf{x})}{\partial \mathbf{x}} - A^t \boldsymbol{\lambda} = \mathbf{0} \\ \frac{L(\mathbf{x}, \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} = A\mathbf{x} - \mathbf{b} = \mathbf{0} \end{cases} \tag{44}$$

不等式约束优化问题

不等式约束优化问题的一般形式是：

$$\min_{\mathbf{x}\in R^m} f(\mathbf{x}) \tag{45}$$

subject to

$$c_i(\mathbf{x}) \leq 0, \quad i = 1, \cdots, k$$

类似于等式约束优化，引入Lagrange函数：

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{i=1}^{k} \lambda_i c_i(\mathbf{x}) \tag{46}$$

可以证明优化问题(45)的最优解$\mathbf{x}^*$，满足如下一组必要条件，称为Karush-Kuhn-Tucker(KKT)条件：

1. $\partial L(\mathbf{x}^*, \boldsymbol{\lambda})/\partial \mathbf{x} = \mathbf{0}$
2. $\lambda_i \geq 0, \quad i = 1, \cdots, k$
3. $\lambda_i c_i(\mathbf{x}^*) = 0, \quad i = 1, \cdots, k$

因此，不等式约束优化问题可以通过求解上述三式得到极值点。观察条件2和3可以得出这样一个关于约束不等式的结论：当某一个不等式在极值点$\mathbf{x}^*$上是以$c_i(\mathbf{x}) < 0$的方式得到满足时，对应的Lagrange系数$\lambda_i = 0$；当以$c_i(\mathbf{x}) = 0$的形式满足时，$\lambda_i > 0$。