第 5 章
典型的非线性分类器

5.1 引言

很多实际情况下，类别之间的分类边界并不是线性的。例如，在第 2 章里我们看到，即使样本都是正态分布的，通常情况下最小错误率的分类器是二次函数。在更复杂的分布情况下，需要更复杂的非线性判别函数来分类。

与线性判别函数不同，非线性判别函数并不是明确的一类函数，而是除线性函数外的各种函数的集合。因此，非线性判别函数和非线性学习机器的设计方法就更多种多样，难以一般性地讨论。本章中，我们选取几种经典的有代表性的非线性分类方法进行介绍，在第 6 章中将要介绍的近邻法、决策树，实现的也都是非线性函数。

5.2 分段线性判别函数

我们知道，一个非线性函数可以用多段线性函数来逼近。分段线性判别函数（piecewise linear discriminant functions）就是采用了这种思想，用多个线性分类器片段来实现非线性分类，如图 5-1 所示。由于每一段分类面都是线性的超平面，可以采用第 4 章讲述的一些线性分类器设计方法进行设计；同时，多段超平面组合可以逼近各种形状的超曲面，能够适应各种复杂的数据分布情况。分段线性判别函

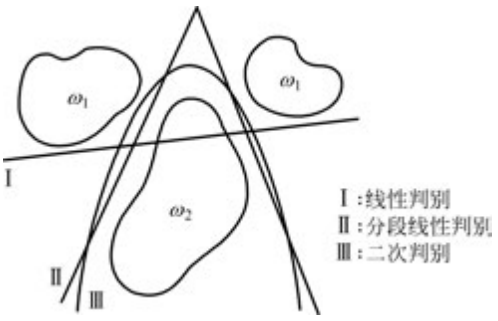


图 5-1 分段线性分类器示意图

数不但能逼近任意已知形式的非线性判别函数，当实际情况下类别之间的划分并不能用解析形式表示时，非线性判别函数仍能很好地对判别函数进行逼近。

实际上，第 4 章中介绍的多类线性判别函数在特征空间里就构成了一组分段线性的决策面。因此，求解两类之间的分段线性判别函数，基本的做法就是把各类划分成适当的子类，在两类的多个子类之间构建线性判别函数，然后把它们分段合并成分段线性判别函数。下面就从最简单的分段线性距离分类器开始介绍设计分段线性判别函数的基本做法。

5.2.1 分段线性距离分类器

在第 2 章我们看到，当两类的类条件概率密度为正态分布，两类先验概率相等，而且各维特征独立且方差相等时，最小错误率贝叶斯决策就是直观的最小距离分类器：以两类各自的均值为中心点，新样本离哪类的中心点近就决策为哪一类。即若有 $\omega_1,\omega_2,\cdots,\omega_c$ 个类别，各类的均值分别是 $\boldsymbol{\mu}_i(i=1,2,\cdots,c)$，对样本 $\boldsymbol{x}$，如果 $\|\boldsymbol{x}-\boldsymbol{\mu}_k\|^2=\min\limits_{i=1,\cdots,c}\|\boldsymbol{x}-\boldsymbol{\mu}_i\|^2$，则决策 $\boldsymbol{x}$ 属于 ω_k 类。两类情况下，最小距离分类器就是两类均值之间连线的垂直平分面(超平面)，如图 5-2 所示。

最小距离分类器虽然是在正态分布的特殊条件下推出来的，但是在很多情况下，只要每一类数据的分布是单峰的、在各维上的分布基本对称且各类先验概率基本相同，则最小距离分类器都不失为一种简单有效的分类方法。实际上，我们可以把类均值看作该类的代表点，或者模板，最小距离分类器就是模板匹配：新样本与哪一类的模板更相似则归为哪一类。

沿着这种思路，在各类的数据分布是多峰的情况下，我们可以把每类划分成若干子类，使每个子类是单峰分布且尽可能在各维上对称。每个子类取均值作为模板，这样每个类就有多个模板，一个两类问题就可以用多类的最小距离分类器来解决，即对一个待分类样本，比较它到各个子类均值的距离，把它分到距离最近的子类所属于的类。这样所得到的分类面就是由多段超平面组成的，如图 5-3 所示。这种分类器称作分段线性距离分类器。这种做法对多类同样适用。

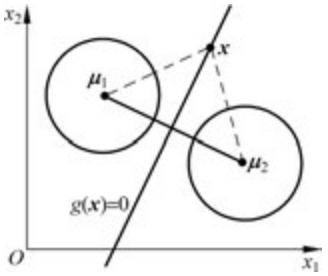


图 5-2 两类的最小距离分类器

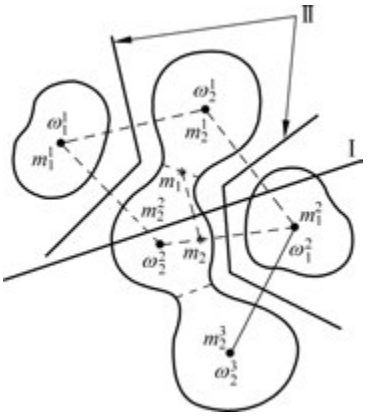


图 5-3 分段线性距离分类器示例

用数学语言来描述，分段线性距离分类器可以表示为：把属于 $\omega_i(i=1,2,\cdots,c)$ 类的样本区域 R_i 划分为 l_i 个子区域 $R_i^l(l=1,2,\cdots,l_i)$，每个子类的均值是 $\boldsymbol{m}_i^l$，对样本 $\boldsymbol{x}$，ω_i 类的判别函数定义为

$$g_i(\boldsymbol{x})=\min_{l=1,\cdots,l_i}\|\boldsymbol{x}-\boldsymbol{m}_i^l\| \tag{5-1a}$$

即本类中离该样本最近的子类均值到样本的距离。决策规则是

$$\text{若 } g_k(\boldsymbol{x})=\min_{i=1,\cdots,c} g_i(\boldsymbol{x}),\text{则决策 } \boldsymbol{x}\in\omega_k \tag{5-1b}$$

5.2.2 一般的分段线性判别函数

5.2.1 节介绍的分段线性距离分类器是分段线性判别函数的特殊情况，适用于各子类在各维分布基本对称的情形。一般情况下，可以对每个子类建立更一般形式的线性判别函数，即把每个类别划分成 l_i 个子类

$$\omega_i=\{\omega_i^1,\omega_i^2,\cdots,\omega_i^{l_i}\},\quad i=1,2,\cdots,c \tag{5-2}$$

对每个子类定义一个线性判别函数

$$g_i^l(\boldsymbol{x})=\boldsymbol{w}_i^l\cdot\boldsymbol{x}+w_{i0}^l,\quad l=1,2,\cdots,l_i,\quad i=1,2,\cdots,c \tag{5-3a}$$

其中 $\boldsymbol{w}_i^l$ 和 w_{i0}^l 分别是对应子类 ω_i^l 的权向量和阈值。当然，这些判别函数也可以用增广的形式表示，即

$$g_i^l(\boldsymbol{y})=\boldsymbol{\alpha}_i^l\cdot\boldsymbol{y},\quad l=1,2,\cdots,l_i,\quad i=1,2,\cdots,c \tag{5-3b}$$

类 ω_i 的分段线性判别函数就定义为

$$g_i(\boldsymbol{x})=\max_{l=1,2,\cdots,l_i} g_i^l(\boldsymbol{x}),\quad i=1,2,\cdots,c \tag{5-4}$$

决策规则是

$$\text{若 } g_k(\boldsymbol{x})=\max_{i=1,2,\cdots,c} g_i(\boldsymbol{x}),\text{则决策 } \boldsymbol{x}\in\omega_k \tag{5-5}$$

两个相邻的类之间的决策面方程就是两个判别函数相等，即

$$g_i(\boldsymbol{x})=g_j(\boldsymbol{x}) \tag{5-6}$$

由于 $g_i(\boldsymbol{x})$ 和 $g_j(\boldsymbol{x})$ 都是由式(5-4)定义的分段线性判别函数，这个决策面也是由多个分段的超平面组成的，其中的一段是一类中的某个子类和另一类中的相邻子类之间的分类面。

在确定了子类划分之后，分段线性判别函数的设计就等同于多类分类器的设计。因此，在分段线性判别函数的设计中所遇到的新问题是子类的划分。可以分为三种情况考虑。

第一种情况，根据问题的领域知识和对数据分布的了解，人工确定子类的划分方案。例如在字符识别中，一种字符作为一个类，而同一个字符又有不同的字体，可以把一种字体作为一个子类。在某些医学研究中，可以把同一种疾病的病人按照性别、年龄、地域或遗传学特征等分成子类。有些情况下还可以尝试多种不同的划分方案。在第 9 章将要介绍非监督学习方法，也可以用其中的方法对同一类的样本进行聚类分析，得到子类的划分。

第二种情况，已知或者可以假定各类的子类数目，但是不知道子类的划分，可以用下面的错误修正法在设计分类器的同时确定出子类的划分。这里用增广的线性判别函数形式来描述这个算法。

条件：已知共有 c 个类别 ω_i，$i=1,2,\cdots,c$，并且已知 ω_i 类应该划分成 l_i 个子类。每

个类都有一定数量的训练样本。

(1) 初始化。任意给定各类各子类的权值$\boldsymbol{\alpha}_i^l(0)$,$l=1,2,\cdots,l_i$,$i=1,2,\cdots,c$,通常可以选用小的随机数。

(2) 在时刻 t,当前权值为$\boldsymbol{\alpha}_i^l(t)$,$l=1,2,\cdots,l_i$,$i=1,2,\cdots,c$,考虑某个训练样本 $\boldsymbol{y}_k\in\omega_j$,找出 ω_j 类的各子类中判别函数最大的子类,记为 m,即

$$\boldsymbol{\alpha}_j^m(t)^{\mathrm{T}}\boldsymbol{y}_k=\max_{l=1,2,\cdots,l_j}\{\boldsymbol{\alpha}_j^l(t)^{\mathrm{T}}\boldsymbol{y}_k\} \tag{5-7}$$

考查当前权值对样本 $\boldsymbol{y}_k$ 的分类情况:

① 若$\boldsymbol{\alpha}_j^m(t)^{\mathrm{T}}\boldsymbol{y}_k>\boldsymbol{\alpha}_i^l(t)^{\mathrm{T}}\boldsymbol{y}_k$,$\forall i=1,\cdots,c$,$i\neq j$,$l=1,\cdots,l_i$,即 $\boldsymbol{y}_k$ 分类正确,则所有$\boldsymbol{\alpha}_i^l(t)$均不变:$\boldsymbol{\alpha}_i^l(t+1)=\boldsymbol{\alpha}_i^l(t)$,$l=1,2,\cdots,l_i$,$i=1,2,\cdots,c$;

② 若对某个 $i\neq j$,存在子类 l 使得$\boldsymbol{\alpha}_j^m(t)^{\mathrm{T}}\boldsymbol{y}_k\leqslant\boldsymbol{\alpha}_i^l(t)^{\mathrm{T}}\boldsymbol{y}_k$,即 $\boldsymbol{y}_k$ 被当前权值错分,则选取$\boldsymbol{\alpha}_i^l(t)^{\mathrm{T}}\boldsymbol{y}_k$中最大的子类(不妨记作 ω_i 类的第 n 个子类),对权值进行如下修正:

$$\boldsymbol{\alpha}_j^m(t+1)=\boldsymbol{\alpha}_j^m(t)+\rho_t\boldsymbol{y}_k \tag{5-8a}$$

$$\boldsymbol{\alpha}_i^n(t+1)=\boldsymbol{\alpha}_i^n(t)-\rho_t\boldsymbol{y}_k \tag{5-8b}$$

其余权值不变。

(3) $t=t+1$,考查下一个样本,回到步骤(2)。如此迭代,直到算法收敛。

可以看出,这个算法与 4.9.2 节介绍的多类线性判别函数的逐步修正法很相像,这里的子类相当于 4.9.2 节中考虑的多类中的一类。所不同的是,这里的分类器设计过程实际上也是子类的划分过程,而考查权值是否需要修正时并不是考查样本是否被分到某个特定的子类,而是只需要判断样本是否被分到它所属的类别的几个子类中的一个。

算法的终止条件是算法收敛,即对所有训练样本都分类正确,在一轮循环中不再对权值进行修正。从第 4 章关于感知器和多类线性判别函数的逐步修正法的讨论可以知道,这种算法只有在不同类别的各个子类之间都是线性可分的情况下才能保证收敛,对某些数据并不一定能实现,在指定子类数目时更是如此。与第 4 章中讨论的方法类似,如果算法不能收敛,人们通常可以用逐步缩小训练步长 ρ_t 的方法强制算法收敛。

当然,在实际应用中,子类数目很多情况下并无法严格地指定。人们可以通过采用不同的子类数目进行一些试验来确定子类数目。

第三种情况是子类数目无法事先确定。虽然可以用不同的子类数目尝试上面的算法,但是如果没有一个参考数字,盲目地试凑各种可能的子类数目所需要的运算量是巨大的。另一种方法是可以采用分类树的思想来分级划分子类和设计分段线性判别函数。下面我们用一个二维的示例来说明这种方法的基本思想。

对于如图 5-4 所示的两类情况,我们可先用两类线性判别函数算法找一个权向量$\boldsymbol{\alpha}_1$,它所对应的超平面 H_1 把整个样本集分成两部分,我们称之为样本子集。由于样本集不是线性可分的,因而每一部分仍然包含两类样本。

接着,再利用算法找出第二个权向量$\boldsymbol{\alpha}_2$、第三个权向量$\boldsymbol{\alpha}_3$,超平面 H_2、H_3 分别把相应的样本子集分成两部分。若某一部分仍然包含两类样本,则继续上述过程,直到某一权向量(如图中$\boldsymbol{\alpha}_4$)把两类样本完全分开为止。

这样得到的分类器显然也是分段线性的,基决策面如图 5-4 中粗线所示。"→"表示权向量$\boldsymbol{\alpha}_i$ 的方向,它指向超平面 H_i 的正侧。它的识别过程是一个树状结构,如图 5-5 所示。

图中用虚线显示了对未知样本 $\boldsymbol{y}$ 的决策过程。经过三步，判断 $\boldsymbol{y}\in\omega_1$。

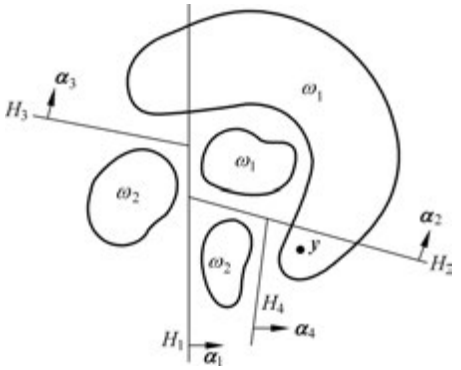


图 5-4 树状分段性分类器举例

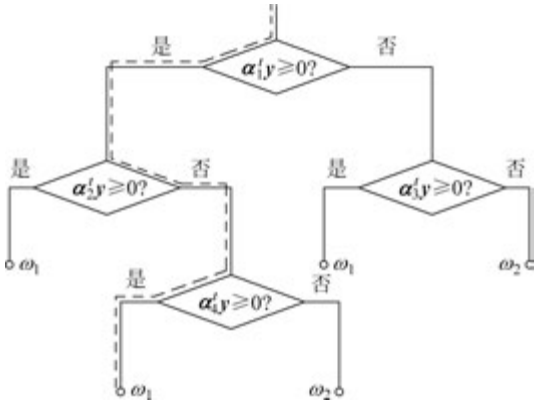


图 5-5 与图 5-4 对应的树状决策过程

需要指出，这种方法对初始权向量的选择很敏感，其结果随初始权向量的不同而大不相同。此外，在每个节点上所用的寻找权向量$\boldsymbol{\alpha}_i$ 的方法不同，结果也将各异。通常可以选择分属两类的欧氏距离最小的一对样本，取其垂直平分面的法向量作为$\boldsymbol{\alpha}_1$ 的初始值，然后求得局部最优解$\boldsymbol{\alpha}_1^*$ 作为第一段超平面的法向量。对包含两类样本的各子类的划分也可以采用同样的方法。

5.3 二次判别函数

在第 2 章已经看到，在一般的正态分布情况下，贝叶斯决策面是二次函数。二次判别函数(quadratic discriminant)也是一种比较常用的固定函数类型的分类方法，它的一般形式是

$$\begin{aligned} g(\boldsymbol{x}) &= \boldsymbol{x}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{x}+\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0 \\ &= \sum_{k=1}^{d} w_{kk}x_k^2 + 2\sum_{j=1}^{d-1}\sum_{k=j+1}^{d} w_{jk}x_jx_k + \sum_{j=1}^{d} w_jx_j + w_0 \end{aligned} \tag{5-9}$$

其中，$\boldsymbol{W}$ 是 $d\times d$ 实对称矩阵，$\boldsymbol{w}$ 为 d 维向量。

不难看出，这个判别函数中包含$\dfrac{1}{2}d(d+3)+1$ 个参数，因此如果像线性判别函数那样，直接根据一定的规则从数据去学习这些参数，计算起来会非常复杂，而且在样本数不足够多时估计如此多的参数，结果的可靠性和推广能力很难保证。

实际中，人们在应用二次判别函数时，往往采用参数化的方法来估计二次判别函数。例如，人们往往假定每一类数据都是正态分布，这时每一类可以定义如下的二次判别函数：

$$g_i(\boldsymbol{x}) = K_i^2 - (\boldsymbol{x}-\boldsymbol{m}_i)^{\mathrm{T}}\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{m}_i) \tag{5-10}$$

其中 $\boldsymbol{m}_i$ 是 ω_i 类的均值，$\boldsymbol{\Sigma}_i$ 是 ω_i 类的协方差矩阵，K_i^2 是一个阈值项，它受协方差矩阵和先验概率的影响。式(5-10)的判别函数就是样本到均值的 Mahalanobis 距离的平方与固定阈值的比较，样本的均值和方差可以用下面的估计：

$$\hat{\boldsymbol{m}}_i = \frac{1}{N_i}\sum_{j=1}^{N_i}\boldsymbol{x}_j$$

$$\hat{\boldsymbol{\Sigma}}_i = \frac{1}{N_i - 1}\sum_{j=1}^{N_i}(\boldsymbol{x}_j - \hat{\boldsymbol{m}}_i)(\boldsymbol{x}_j - \hat{\boldsymbol{m}}_i)^{\mathrm{T}}$$

当两类都近似服从正态分布时，可以对每一类估计式(5-10)的判别函数，两类间的决策面方程就是

$$g_1(\boldsymbol{x}) - g_2(\boldsymbol{x}) = 0$$

代入式(5-10)并整理，可得

$$-\boldsymbol{x}^{\mathrm{T}}(\hat{\boldsymbol{\Sigma}}_1^{-1} - \hat{\boldsymbol{\Sigma}}_2^{-1})\boldsymbol{x} + 2(\hat{\boldsymbol{m}}_1^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_1^{-1} - \hat{\boldsymbol{m}}_2^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_2^{-1})\boldsymbol{x} - (\hat{\boldsymbol{m}}_1^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_1^{-1}\hat{\boldsymbol{m}}_1 - \hat{\boldsymbol{m}}_2^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_2^{-1}\hat{\boldsymbol{m}}_2) + (K_1^2 - K_2^2) = 0$$

决策规则是

$$\text{若 } g_1(\boldsymbol{x}) - g_2(\boldsymbol{x}) \gtrless 0, \text{则 } \boldsymbol{x} \in \begin{cases}\omega_1 \\ \omega_2\end{cases} \tag{5-11}$$

其中，可以通过调整两类的阈值 K_1^2 和 K_2^2 来调整两类错误率情况。

另一种情况是，两类中一类 ω_1 分布比较成团(近似正态分布)，另一类 ω_2 则比较均匀地分布在第一类附近，这种情况下只要对第一类求解其二次判别函数即可，即

$$g(\boldsymbol{x}) = K^2 - (\boldsymbol{x} - \hat{\boldsymbol{m}}_1)^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_1^{-1}(\boldsymbol{x} - \hat{\boldsymbol{m}}_1) \tag{5-12}$$

决策规则是

$$\text{若 } g(\boldsymbol{x}) \gtrless 0, \text{则 } \boldsymbol{x} \in \begin{cases}\omega_1 \\ \omega_2\end{cases} \tag{5-13}$$

同样，可以用 K^2 来调整决策的偏向。直观解释是，当样本到 ω_1 类均值的 Mahalanobis 距离的平方小于 K^2 时则决策为 ω_1 类，否则决策为 ω_2 类。

5.4 多层感知器神经网络

在第 1 章中曾经讨论过，模式识别是一种基本的智能活动，对模式识别方法的研究是机器智能研究的一个重要方面。人们对机器智能的研究有两个主要的出发点，一是通过试图对人类(和其他高级动物)的自然智能建立一定的数学模型，来帮助理解智能活动的奥秘；二是利用各种数学手段，以计算机为工具建立具备一定智能的机器。本书前几章介绍的模式识别方法，除感知器外，其他大部分都是直接从数学角度来分析数据，建立线性或非线性的判别函数，并没有直接与“智能”相连。

从 20 世纪 40 年代开始，科学家们开始了对 Cybernetics 的系统研究。Cybernetics 中文翻译为“控制论”或“生物控制论”，其核心思想是，对于很多不管是生物的还是人造的系统，它们可以通过对其中信息处理和传递的建模得到更好的理解，而不是对能量传递的建模。当时引领这一研究方向的科学家包括 Alan Turing、Warren McCulloch、Claude Shannon、Norbert Wiener、John von Neumann 和 Kenneth Craik 等，其中，N. Wiener(维纳)在 1948 年出版的《控制论》一书把 Cybernetics 定义为动物和机器中的控制与通信过程。1943 年，W. S. McCulloch 和 W. H. Pitts 提出了对神经细胞信息加工的数学模型，这就是第 4 章中介绍的感知器方法所采用的模型。

第 4 章中介绍的感知器方法和 ADALINE 方法在建立能够从数据进行自动学习的机器上取得了重要进展，是人工神经网络(artificial neural networks)研究的开端，但由于技术

和人为的原因，这一研究方向从 20 世纪 60 年代开始不再受到当时的主流研究者的重视，直到 20 世纪 80 年代才再一次得到较大发展。

我们先来看人工神经网络的基本思想。根据对自然神经系统构造和机理的认识，神经系统是由大量的神经细胞（神经元）构成的复杂的网络，人们对这一网络建立一定的数学模型和算法，设法使它能够实现诸如基于数据的模式识别、函数映射等带有“智能”的功能，这种网络就是人工神经网络。

采用不同的数学模型就得到不同的神经网络方法，其中最有影响的模型应该是多层感知器（multi-layer perceptron，MLP）模型，它具有从训练数据中学习任意复杂的非线性映射的能力，也包括实现复杂的非线性分类判别函数。从模式识别角度，多层感知器方法可以看作一种通用的非线性分类器设计方法。

从“多层感知器”这一名字即可看出，它与第 4 章介绍的感知器算法有紧密的联系。本节介绍多层感知器方法，也顺便引出人工神经网络的一些基本概念。

5.4.1 神经元与感知器

一个神经元（neuron）就是一个神经细胞，它是神经系统的基本组成单位。根据目前的认识，一个典型的神经元由以下几部分组成（如图 5-6 所示）：

（1）细胞体（cell body），是神经细胞的主体，内有细胞核和细胞质，除了实现细胞生存的各种基本功能外，这里是神经细胞进行信息加工的主要场所。

（2）树突（dendrites），是细胞体外围的大量微小分支，是细胞的“触角”，一个神经元的树突可达 10^3 数量级，多数长度很短，主要担负着从外界（其他细胞或体液环境）接收信息的功能。

（3）轴突（axon），是细胞的输出装置，负责把信号传递给另外的神经细胞，通常每个神经元有一个轴突，有的轴突会很长，例如人体四肢的某些神经细胞的轴突可以长达 1m 以上。

（4）突触（synapse），是一个神经元的轴突与另一个神经元的树突相“连接”的部位，这种连接并不是物理上的直接接触，而是二者的细胞膜充分靠近，通过之间的微小缝隙传递带电离子。

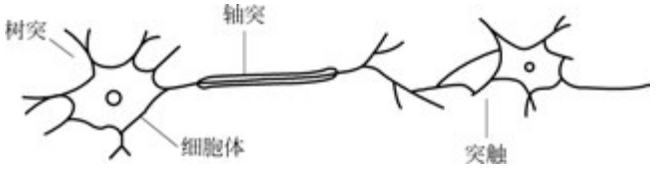


图 5-6 典型的神经元构成示意图

神经系统中的信号是电化学信号，是靠带电离子在细胞膜内外的浓度差来形成和维持的，这种信号可以以脉冲的形式沿着轴突传播，并经由突触把电荷传递给下一个神经元。突触的不同状态可以影响信号传递的效率，可以称之为突触的连接强度，同时，信号的传递效率也可以受到细胞所在的体液环境中相关离子浓度等的影响。一个神经系统就是由大量神经元组成的，人的神经系统中各种神经元的总数可达 $10^{10}\sim10^{11}$。神经元之间通过突触连接，构成了复杂的神经网络系统。

一个典型的、简化的神经元工作过程是这样的：来自外界（环境或其他细胞）的电信号通过突触传递给神经元，当细胞收到的信号总和超过一定的阈值后，细胞被激活，通过轴突向下一个细胞发送电信号，完成对外界信息的加工。这一过程可以用如图 5-7 所示的数学

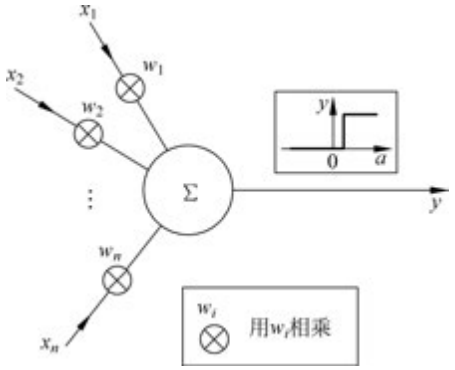


图 5-7 阈值逻辑单元：McCulloch-Pitts 神经元模型

模型表示出来，称作 McCulloch-Pitts 模型，是由 W. S. McCulloch 和 W. H. Pitts 在 1943 年提出的[①]。图 5-7 中，$x_1,\cdots,x_n$ 表示神经元的多个树突接收到的信号，n 是向量 $\boldsymbol{x}$ 的维数，$w_1,\cdots,w_n$ 称作权值，反映了各个输入信号的作用强度。神经元的作用是将这些信号加权求和，当求和超过一定的阈值后神经元即进入激活状态，输出值 $y=1$；否则神经元处于抑制状态，输出值为 0。

这个模型可以用下面的公式表示：

$$y=\theta\left(\sum_{i=1}^{n}w_ix_i+w_0\right) \tag{5-14}$$

也称作阈值逻辑单元(threshold logic unit，TLU)。其中，$\theta(\cdot)$为单位阶跃函数(当自变量为正时函数取值为 1，否则取值为 0)。在某些情况下，也可以用符号函数 sgn(·)替代 $\theta(\cdot)$，这时输出 y 的取值就是 1 或−1。在这个神经元模型中，x 称作神经元的输入，w 称作神经元的权值，$\theta(\cdot)$或 sgn(·)函数称作神经元的传递函数，y 称作神经元的输出。

从几何上说，感知器神经元就是用平面(在高维空间中是超平面)

$$\sum_{i=1}^{n}w_ix_i+w_0=0$$

把特征空间分成两个区域，一个区域内 $y=1$，另一个区域内 $y=-1$(或 0)。

当然，这个神经元的模型是极度简化的，实际神经系统中的神经元的活动要复杂得多。有人甚至比喻单个神经元的复杂程度就相当于一台数字计算机。但是，作为一种高度的抽象，这一简单的模型反映了神经元最典型的也是最关键的特性，是人工神经网络的基础。

回顾第 4 章的内容，可以很容易发现，如果把神经元的两个可能的输出值看作两类，式(5-14)所描述的神经元的传递函数实际就是一个线性分类器。特别地，如果分别用 $y=0$ 和 $y=1$ 来表示要区分的两类，用 $d(\boldsymbol{x})$代表训练样本 $\boldsymbol{x}$ 的正确分类，式(5-14)就是第 4 章讲过的感知器判别函数，其中的权值可以按照以下公式根据每个训练样本进行迭代的训练：

$$\boldsymbol{w}(t+1)=\boldsymbol{w}(t)+\eta(t)\{d(\boldsymbol{x}(t))-y(t)\}\boldsymbol{x}(t) \tag{5-15}$$

其中，t 为迭代次数记数，$\boldsymbol{x}(t)$是当前时刻考查的样本，$\eta(t)$是训练步长。(作为练习，读者可以自己分析一下为什么式(5-15)的学习算法与第 4 章讨论的感知器算法是等价的。)

可以证明，当两类数据线性可分时，式(5-15)的神经元权值迭代算法能够在有限步内收敛到一个使所有训练样本都正确分类的解。

5.4.2 用多个感知器实现非线性分类

正如在第 4 章中看到的，单个感知器神经元能够完成线性可分数据的分类问题，是一种

① McCulloch W S，Pitts W H. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*，1943，5：115-133.

最简单的可学习机器。但是，它无法解决非线性问题。例如，在图 5-8 的四个点中，如果(1,1)点和(－1,－1)点同属于第一类，而(1,－1)点和(－1,1)点同属于第二类，这一问题在逻辑学里称作异或(XOR)问题。对于这样的问题，单个感知器神经元是无法正确分类的。

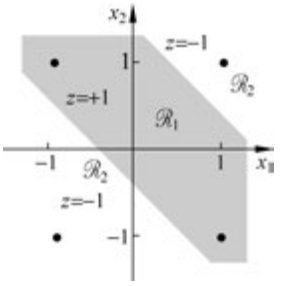


图 5-8 异或问题

单个阈值逻辑单元作为分类器的这一局限早在 1969 年 Minsky 和 Papert 的专著《感知器》里就进行了透彻的分析[①]。他们证明了感知器只能解决所谓一阶谓词逻辑问题，如与(AND)、或(OR)等，而不能解决异或(XOR)之类的高阶谓词逻辑问题。

很快人们就发现，虽然一个神经元无法实现诸如异或这样的高阶谓词逻辑问题，但是可以通过将多个神经元分层组合起来实现复杂的空间形状的分割。例如，如果按图 5-9 所示，将两层神经元按照一定的结构和系数进行组合，用第一层神经元分别实现两个线性分类器，把特征空间分割，而在这两个神经元节点的输出之上再加一层感知器节点，就可以实现异或运算。

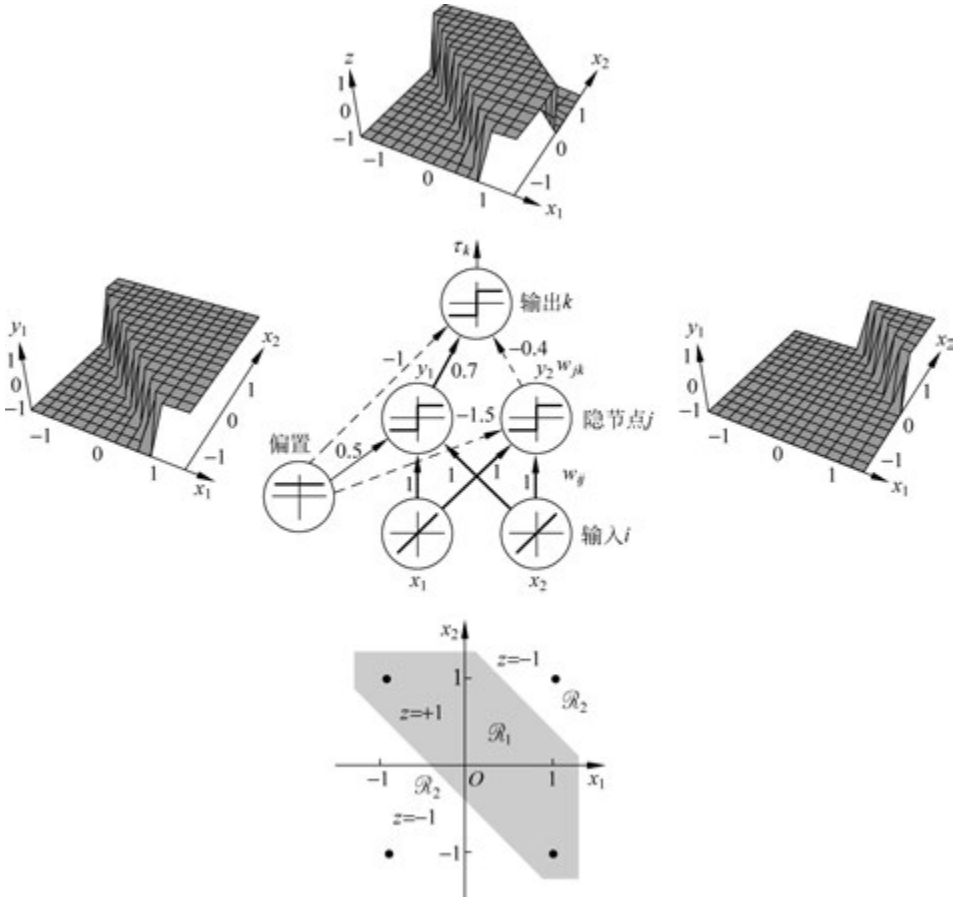


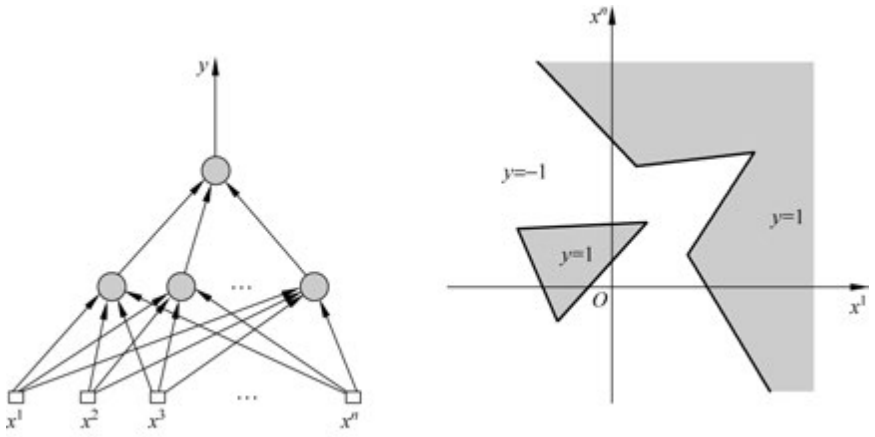
图 5-9 用两层神经元实现异或形式的分类(取自文献[4])

① Minsky M, Papert S. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

实际上，可以想象，正如可以用分段线性判别函数来实现复杂的非线性分类面一样，对于任意复杂形状的分类区域，总可以用多个神经元组成一定的层次结构来实现分类，如图 5-10 所示。也就是，可以用多个感知器的组合

$$y=\theta\left\{\sum_{j=1}^{n} v_j\theta\left(\sum_{i=1}^{m} w_{ij}x_i+w_{0j}\right)+v_0\right\} \tag{5-16}$$

来实现非线性分类面，其中的 $\theta(\cdot)$ 是阶跃函数或符号函数。



(a) 感知器模型　　(b) 输入空间中的两个区域

图 5-10　用多个感知器组成两层结构实现分段线性分类

在认识到单个感知器的局限后，Rosenblatt 提出了这种多层的学习模型：前一层神经元的输出是后一层神经元的输入，最后一层只有一个神经元，它接收来自前一层的 n 个输入，给出作为决策的一个输出。

遗憾的是，在 20 世纪 60 年代，人们发现感知器学习算法无法直接应用到这种多层模型的参数学习上，因此，Rosenblatt 提出了这样的方案[①]：除了最后一个神经元之外，事先固定其他所有神经元的权值，学习过程只是用感知器学习算法来寻找最后一个神经元的权系数。实际上，这样做就相当于通过第一层神经元把原始的特征空间变换到了一个新的空间，第一层的每个神经元构成新空间的一维，每一维取值都为二值（{0,1}或{−1,1}），然后在这个新空间里用感知器学习算法构造一个线性分类器。显然，由于第一层神经元的权值是需要人为给定的，模型的性能很大程度上取决于能否设计出恰当的第一层神经元模型，而这又取决于对所面临的数据和问题的了解。人们当时没有找到能够针对任意问题求解第一层神经元参数的方法，所以这方面没有进一步进展，人们对感知器的研究就此停滞了大约 25 年。

5.4.3　反向传播算法

回顾第 4 章介绍的感知器学习算法，其核心思想是梯度下降法，即以训练样本被错分的

① Rosenblatt F. *Principles of Neurodinamics: Preceptron and Theory of Brain Mechanisms*. DC: Spartan Books, 1962.

程度为目标函数，训练中每次出现错误时便使权系数朝着目标函数相对于权系数的负梯度方向更新，直到目标函数取得极小值即没有训练样本被错分。在采用多层的阈值逻辑单元时，由于神经元的传递函数是阶跃函数，输出端的误差只能对最后一个神经元的权系数求梯度，无法对其他神经元的权系数求梯度，所以无法使用这种梯度下降法训练其他神经元的权系数。

1986 年，在人们无法解决感知器多层模型的参数学习问题长达 25 年之后，几项几乎同时发表的研究工作给出了求解这些参数的一种有效算法，这就是所谓反向传播（back-propagation 或 BP）算法[①②]。这一算法上的突破，主要来源于用所谓 Sigmoid 函数代替感知器中的阈值函数来构造神经元网络。

阶跃函数 $\theta(\alpha)$ 的曲线如图 5-11 所示，它在 $\alpha=0$ 处不可导。考查图 5-12 的函数，当 α 远离 0 时，该函数与阶跃函数相同，所不同的是函数取值从 0 到 1 的变化不是突然完成的，而是平滑地完成的。这种函数看上去像是字母 S 的变形，因此称作 Sigmoid 函数（即“S 形”函数）。

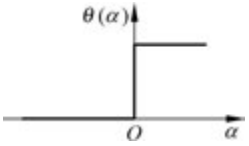


图 5-11 阶跃函数 $\theta(\alpha)$ 的曲线

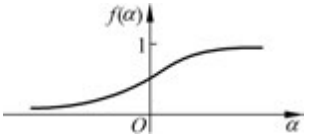


图 5-12 Sigmoid（S 形）函数

Sigmoid 函数通常可以写成

$$f(\alpha)=\frac{1}{1+\mathrm{e}^{-\alpha}} \tag{5-17}$$

其取值范围在(0,1)，它可以看作对阶跃函数的一种逼近。如果要逼近符号函数，可以用下面的双曲正切形式：

$$f(\alpha)=\mathrm{th}(\alpha)=\frac{\mathrm{e}^{\alpha}-\mathrm{e}^{-\alpha}}{\mathrm{e}^{\alpha}+\mathrm{e}^{-\alpha}}=\frac{2}{1+\mathrm{e}^{-2\alpha}}-1 \tag{5-18}$$

其取值范围是(−1,1)。

如果用 Sigmoid 函数替代阶跃函数作为神经元的传递函数，则得到

---

① Rumelhart D E, Hinton G E, William R J. Learning representations by back-propagating errors, Nature, 323: 533-536, 1986; Rumelhart D, Hinton G and Williams R. Learning internal representations by error propagation. In: *Parallel Distributed Processing*, *Chapter* 8, MIT Press, Cambridge, MA, 1986, pp. 318-362; Parker D. *Learning Logic*. Technical Report TR-87, Cambridge, MA: Center for Computational Research in Economics and Management Science, MIT, 1985; LeCun Y. Learning processes in an asymmetric threshold network. In: Bienenstock E, Fogelman-Smith F, Weisbuch G. (eds). *Disordered Systems and Biological Organization*, NATO ASI Series, F20. Berlin: Springer-Verlag, 1986.

② 实际上，反向传播算法的思想最早是由 P. J. Werbos 于 1974 年在其哈佛大学的博士学位论文 *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences* 中提出的，但是当时并没有引起人们太多注意，也未与人工神经网络联系起来。在更早的一部著作 A. E. Bryson and Y-C. Ho, *Applied Optimal Control*. New York: Blaisdell, 1969 中也描述了相似的思路。但是，直到 1986 年几项新的独立的工作在神经网络的领域中先后发表出来，才使得这一方法得到了迅速的普及和推广应用。

$$y=f(\boldsymbol{x})=\frac{1}{1+\mathrm{e}^{-\sum_{i=1}^{n}w_i x_i-w_0}} \tag{5-19a}$$

为了方便分析，通常人们把常数项 w_0 作为一个固定输入 1 的权值合并到加权求和项中，把上式简化成

$$y=f(\boldsymbol{x})=\frac{1}{1+\mathrm{e}^{-\sum_{i=1}^{n}w_i x_i}} \tag{5-19b}$$

这里，分母的指数项中的求和应该比式(5-19a)中多一项，但为了书写方便我们仍写为从 $i=1$ 到 n，即假定 $x_i(i=1,2,\cdots,n)$ 中已经包含一个对应常数输入的项。

可以看到，Sigmoid 函数是单调递增的非线性函数，无限次可微，而且当权值较大时可以逼近阈值函数，当权值很小时则逼近线性函数。

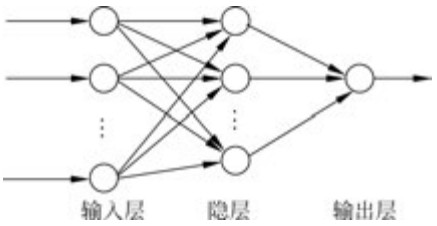


图 5-13 带有一个隐层的多层感知器神经网络的例子

人们通常把由多个计算神经元相互连接组成的系统称作人工神经网络(简称神经网络)，而把由多个感知器组成的神经网络称作多层感知器(multi-layer perceptron，MLP)网络，如图 5-13 所示。由于采用 Sigmoid 函数作为神经元的传递函数，不管网络的结构多么复杂，总可以通过计算梯度来考查各个参数对网络输出的影响，通过梯度下降法调整各个参数。这就是多层感知器反向传播算法的基本思想。人工神经网络的研究从 20 世纪 80 年代中期开始得到了迅猛的发展，多层感知器反向传播算法的突破是其中一个重要的原因。从那时起，“人工神经网络”一词开始成为机器学习领域被使用最多的词汇之一，而多层感知器则成为人工神经网络的典型代表。在一些情况下，很多人甚至说人工神经网络就专指多层感知器。当然，这种说法很不严格，因为还有其他类型的人工神经网络。

多层感知器实现的是从 d 维输入 $\boldsymbol{x}$ 到输出 $\boldsymbol{y}$(一维或多维)的一个映射。它由多个采用 Sigmoid 传递函数的神经元节点连接而成，这些神经元节点分层排列，每一层的神经元接收来自前一层的信号，经过加工后又传递给后一层。

多层感知器的第一层是输入层(input layer)，每个节点对应于 $\boldsymbol{x}$ 的每一维，节点本身并不完成任何处理，只是把每一维的信号“分发”到后一层的每个节点。最后一层是输出层(output layer)，如果 y 是一维，则输出层只有一个节点。在输入层和输出层之间的各层都被称作“隐层”(hidden layer)或中间层，其中的节点都被称作“隐节点”(hidden nodes)，因为它们是“隐藏”在输入层和输出层之间的。

类似这种形式的神经网络被称作是前馈型神经网络(feedforward neural networks)，是神经网络的主要结构形式之一。前馈型神经网络中，信号沿着从输入层到输出层的方向单向流动，输入层把信号传递给隐层，隐层再把信号传递给下一个隐层(如果有多个隐层)，最后一个隐层把信号传递给输出层。这种神经网络实现的是从输入层到输出层的映射。把一个样本特征向量的各分量分别输入网络输入层的各个对应节点上，经过在网络上从前向后一系列加工运算，在输出端得到相应的输出值(或输出向量)。

人们通常把包含输入层、输出层和一个隐层的多层感知器叫作一个三层的多层感知器网络或三层前馈神经网络。如果有两个隐层，则叫作四层网络。图 5-13 就是一个三层前馈神经网络的示意图。需要说明的是，这只是多数人约定俗成的叫法，也有文献在考虑神经网络的层数时不计入第一层(输入层)，那样的话，图 5-13 中的例子就算是一个两层前馈网络，而有两个隐层的网络则叫作三层网络。在文献里，这两种说法可能都有人采用，需要特别注意。比较确切的描述方法是用隐层的个数来描述神经网络的结构，这样就不容易产生歧义，例如图 5-13 就是带有一个隐层的多层感知器神经网络模型。

一个输出是一维的三层感知器所实现的从 $\boldsymbol{x}$ 到 y 的映射可以用下面的函数来描述：

$$y = g(\boldsymbol{x}) = f\left(\sum_{j=1}^{n_H} w_{jk} f\left(\sum_{i=1}^{d} w_{ij} x_i\right)\right) \tag{5-20}$$

其中，$f()$是 Sigmoid 函数(或其他传递函数)，d 是输入 $\boldsymbol{x}$ 的维数，即输入层节点的数目，w_{ij} 是从输入层第 i 个节点到隐层第 j 个节点之间的连接强度(权值)，w_{jk} 是从隐层第 j 个节点到输出层第 k 个节点(这里只有一个输出节点，$k=1$)之间的权值。

在讨论多层感知器学习算法之前，先来看这种多层感知器类型的前馈网络能够逼近什么函数。Kolmogorov 曾经证明[①]，在单位超立方体 $\boldsymbol{I}^n$ ($\boldsymbol{I}=[0,1]$，$n>2$)内的任意连续函数 $g(\boldsymbol{x})$，都可以通过选择适当的 $\Xi()$和 $\psi()$表示成以下两级函数求和的问题：

$$g(\boldsymbol{x}) = \sum_{j=1}^{2n+1} \Xi_j \left(\sum_{i=1}^{d} \psi_{ij}(x_i)\right) \tag{5-21}$$

类似地，人们研究发现，任意一个从 $\boldsymbol{x}$ 到 y 的非线性映射，都存在一个适当结构的三层前馈神经网络能够以任意的精度来逼近它。人们分别从 Kolmogorov 定理的角度和函数的傅里叶展开的思想来研究了这一性质[②]，感兴趣的读者可以参考有关人工神经网络的专门教材来进一步学习对这一性质的有关研究。

这一结论说明，多层感知器神经网络是一种可普遍适用的非线性学习机器，能够实现任意复杂的函数映射。但是，这一结论是存在性的结论，并没有指出如何才能得到能够实现所需映射的网络结构和参数。

不同的神经网络结构、不同的神经元传递函数和不同的权值决定方法就构成了不同类型的神经网络。通常，在实际应用中，神经元传递函数是确定的(对于前馈网络来说一般都是 Sigmoid 函数)，神经网络的结构也是事先设定的，而网络中各个神经元的权值是通过训练样本进行学习的。训练样本就是一组 $\boldsymbol{x}$ 和 y 都是已知的样本，学习算法根据这些样本来调整神经网络中的权值，目标是使神经网络能够最好地逼近 y 和 $\boldsymbol{x}$ 之间的函数关系，即让训练以后的神经网络“学会”所需的函数映射。

---

① Kolmogorov A N. On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. *Doklady Akademiia Nauk SSSR*, 1957,114(5): 953-956.

② George Cybenko, Approximation by superpositions of a sigmoidal function. *Mathematical Control Signals Systems*,1989,2: 303-314.

Robert Hecht-Nielsen. Theory of the backpropagation neural network. In: *Proceedings of the International Joint Conference on Neural Networks* (*IJCNN*), v. 1, pp. 593-605, New York: IEEE, 1989.

下面就来介绍用于在给定多层感知器结构的情况下训练其权值的反向传播算法(BP 算法),关于如何确定神经网络结构的问题,放在后面的小节里讨论。为了讨论方便,我们参照图 5-14 所示对神经网络的各个变量重新约定如下:

假设输入向量为 n 维,$\boldsymbol{x}=[x_1,\cdots,x_n]^{\mathrm{T}}$。用上标 l 代表神经元节点所在的层,输入层记 $l=0$,第一个隐层记 $l=1$,以此类推。记总层数为 L。如果是一个有两个隐层的四层网络,则输出层记为 $l=L-1=3$。第 l 层第 i 个神经元的输出记作 x_i^l,对输入层 $x_i^0=x_i$,$i=1,2,\cdots,n$。设输出层节点的个数为 m,即网络有 m 维输出 $\boldsymbol{y}=[y_1,\cdots,y_m]^{\mathrm{T}}$,第 l 个隐层的神经元个数为 n_l。第 l 层的权值都用 w_{ij}^l 表示,其中上标 l 表示所在层,下标 ij 表示所连接的两个节点,w_{ij}^l 就表示第 $l-1$ 层的节点 i 连接到第 l 层的节点 j 的权值。

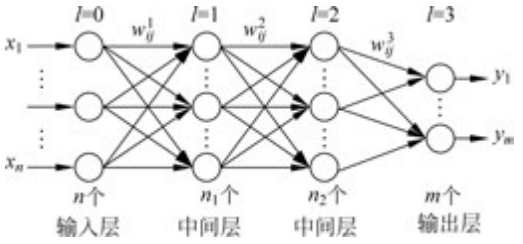


图 5-14 多层感知器的例子和符号约定

BP 算法是迭代计算各个权值,在下面的算法描述中,用 $w_{ij}^l(t)$ 表示在 t 时刻(第 t 步迭代)时权值 w_{ij}^l 的取值。

BP 算法的目标函数是神经网络在所有训练样本上的预测输出与期望输出的均方误差,采用梯度下降法通过调整各层的权值求目标函数最小化。这里只介绍算法的思路和步骤。算法的推导过程可作为作业供读者练习,也可查阅几乎任何一本神经网络方面的教材。

BP 算法的基本做法是,在训练开始之前,随机地赋予各权值一定的初值。训练过程中,轮流对网络施加各个训练样本。当某个训练样本作用于神经网络输入端后,利用当前权值计算神经网络的输出,这是一个信号从输入隐层再到输出的过程,称作前向过程。考查所得到的输出与训练样本的已知正确输出之间的误差,根据误差对输出层权值的偏导数修正输出层的权值;把误差反向传递到倒数第二层的各节点上,根据误差对这些节点权值的偏导数修正这些权值,以此类推,直到把各层的权值都修正一次。然后,从训练集中抽出另外一个样本进行同样的训练过程。如此不断进行下去,直到在一轮训练中总的误差水平达到预先设定的阈值,或者训练时间达到了预定的上限。

在这个学习过程中,误差反向传播到各隐层节点是能够对中间各层的权值进行学习的关键。在早期采用阶跃函数作为感知器的传递函数时,只能根据误差对最后一个感知器的权值进行训练,无法对前面的权值进行调整。由于采用了 Sigmoid 传递函数,误差可以分别对所有权值项计算梯度,才可以把误差反向传递到各个节点上。因此,人们把这种神经网络的权值学习算法称作反向传播算法(BP 算法,亦译后向传播算法)。由于反向传播算法是多层感知器神经网络的标准学习算法,因此也有人干脆把这种前馈型的多层感知器神经网络称作 BP 网络。

下面给出 BP 算法的具体步骤。

(1) 确定神经网络的结构,用小随机数进行权值初始化,设训练时间 $t=0$。

(2) 从训练集中得到一个训练样本 $\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}\in\mathbf{R}^n$,记它的期望输出是 $\boldsymbol{D}=[d_1,d_2,\cdots,d_m]^{\mathrm{T}}\in\mathbf{R}^m$。样本通常按照随机的或任意的顺序从训练集中选取。

(3) 计算在 $\boldsymbol{x}$ 输入下当前神经网络的实际输出,即

$$y_r=f\left(\sum_{s=1}^{n_{L-2}}w_{sr}^{l=L-1}\cdots f\left(\sum_{j=1}^{n_1}w_{jk}^{l=2}f\left(\sum_{i=1}^{n}w_{ij}^{l=1}x_i\right)\right)\right),\quad r=1,\cdots,m \tag{5-22}$$

其中,$f(\cdot)$是 Sigmoid 函数,即

$$f(\alpha)=\frac{1}{1+\mathrm{e}^{-\alpha}} \tag{5-23}$$

(4) 从输出层开始调整权值,做法如下。

对第 l 层,用下面的公式修正权值:

$$w_{ij}^{l}(t+1)=w_{ij}^{l}(t)+\Delta w_{ij}^{l}(t),j=1,\cdots,n_l,\quad i=1,\cdots,n_{l-1} \tag{5-24}$$

其中,$\Delta w_{ij}^{l}(t)$为权值修正项,即

$$\Delta w_{ij}^{l}(t)=-\eta\delta_j^l x_i^{l-1} \tag{5-25}$$

η 是学习步长,需在学习之前事先给定(必要时在算法中可变化)。

对输出层($l=L-1$),δ_j^l 是当前输出与期望输出之误差对权值的导数,即

$$\delta_j^l=-y_j(1-y_j)(d_j-y_j),\quad j=1,\cdots,m \tag{5-26}$$

而对中间层,δ_j^l 是输出误差反向传播到该层的误差对权值的导数,即

$$\delta_j^l=x_j^l(1-x_j^l)\sum_{k=1}^{n_{l+1}}\delta_k^{l+1}w_{jk}^{l+1}(t),\quad j=1,\cdots,n_l \tag{5-27}$$

(5) 在更新全部权值后对所有训练样本重新计算输出,计算更新后的网络输出与期望输出的误差。检查算法终止条件,如果条件已达到则停止,否则置 $t=t+1$,返回步骤(2)。算法的终止条件通常是在最近一轮训练中网络实际输出与期望输出之间的总误差(均方误差)小于某一阈值,或者是在最近一轮的训练中所有权值的变化都小于一定阈值,或者算法达到了事先约定的总训练次数上限。

需要注意,这里给出的 BP 算法是针对神经元节点为式(5-23)的 Sigmoid 函数的。这时 $f(\alpha)$的梯度函数是

$$f'(\alpha)=f(\alpha)(1-f(\alpha)) \tag{5-28}$$

如果采用其他形式的 Sigmoid 函数或其他函数,则梯度形式就不同,需要根据其梯度函数修正算法步骤(4)中的式(5-26)、式(5-27)。

还需要说明的是,虽然采用 Sigmoid 函数后可以用梯度下降算法计算各层节点的权值,但是算法有可能会陷入目标函数的局部极小点,不能保证收敛到全局最优点。这是因为,通常情况下,目标函数是权值的复杂的非线性函数,往往存在多个局部极小点,梯度下降算法如果收敛到某一个局部极小点,梯度就会等于或接近 0,无法进一步改进目标函数,导致学习过程无法收敛到全局最优解。

研究 BP 算法的误差收敛过程对于掌握神经网络的学习情况是很重要的。为了直观地考查算法的收敛情况,可以画出如图 5-15 所示的误差曲线,直观考查误差随训练时间变化的情况。较理想的情况是,误差能够在较短时间内趋于一个较小的值或 0,如图 5-15 中实线所

示；而图中的两条虚线给出的误差曲线则分别反映了算法收敛很慢和学习过程出现很大振荡的情形。

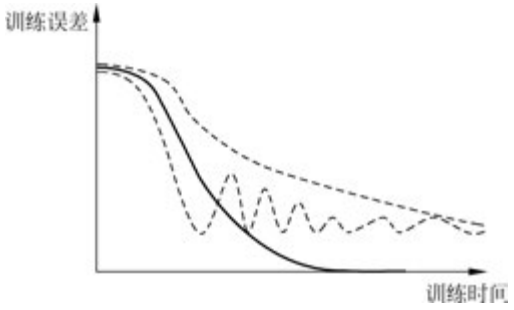


图 5-15 多层感知器训练的误差曲线示例

BP 算法的最终收敛结果有时受初始权值的影响很大。对于多层感知器神经网络(三层或三层以上),各个初始权值不能为 0,也不能都相同,而是应该采用较小的随机数(例如±0.3 内的随机数)作为初始权值。在实际应用中,如果算法很难收敛,可以尝试改变初值重新试算。

一个影响算法收敛性质的重要参数是学习步长 η。通常,如果步长太大,收敛速度可能一开始会较快,但可能会容易导致算法出现振荡而不能收敛或收敛很慢;如果步长太小,则权值调整可能会非常慢,导致算法收敛太慢,而且一旦陷于局部极小点就容易停在那里。为了选取适当的学习步长,往往需要在具体问题上进行试探和摸索,例如有些情况下步长可在 0.1～3 内选择,但针对不同的数据和神经网络结构,步长选择可能会很不同。试算过程中观察不同步长下得到的误差收敛曲线有助于找到针对特定问题的较合理的步长。

为了兼顾训练过程和训练的精度,人们有时采用变步长的办法,例如开始时采用较大的步长,而随着学习的不断进行,逐步减小步长。

为了使 BP 算法有更好的收敛性能,人们提出了很多改进方案。其中,一种有代表性的思想是在权值更新过程中引入“记忆项”或“惯性项”,使本次权值修改的方向不是完全由当前样本下的误差梯度方向决定,而是采用上一次权值修改方向与本次负梯度方向的组合,在某些情况下这样可以避免过早地收敛到局部极小点。具体做法是,把 BP 算法中式(5-25)定义的权值更新项改为

$$\Delta w_{ij}^{l}(t)=\alpha\Delta w_{ij}^{l}(t-1)+\eta\delta_{j}^{l}x_{i}^{l-1} \tag{5-29}$$

这里介绍的是最基本的 BP 算法。针对不同的具体问题,人们先后发展了很多改进的 BP 算法。BP 算法也是现在普遍关注的深度神经网络学习的最基本方法。

5.4.4 多层感知器网络用于模式识别

由于多层感知器神经网络具有通用非线性函数逼近器的性质,它在模式识别问题中得到了广泛的应用。尤其是对于非线性的模式识别问题,传统方法中需要设定特殊的非线性判别函数的形式才能设计分类器,或者需要设计分段线性的分类器,而如果用神经网络就显得非常方便:它具有“黑盒子”的特点,只要事先确定了神经网络结构,那么只需要用 BP 算法来训练神经网络,并不需要关心网络最后实现的分类器的具体形式。从应用上,神经网络

也在很多领域应用中都取得了很好的效果。而且，由于它看起来不需要使用者对数学模型有很多的了解，其基本思想很快就被各个领域所接受，从而极大地推动了模式识别概念的普及并激发了各行各业对模式识别技术的应用。

对于一个模式识别问题，我们的任务是根据样本的特征向量 $\boldsymbol{x}$ 来预测样本的分类。在上面的介绍中，多层感知器的输出 y 是连续变量，要用它来实现分类，就需要用神经网络输出对类别进行适当的编码。下面我们来看看用这种神经网络解决模式识别问题的一般做法。

1. 两类问题

模式识别中研究最多的是两类问题。对于两类问题，最常见的做法是，神经网络采用一个输出节点，在训练阶段把其中一类样本的期望输出指定为 0，另一类的期望输出指定为 1。在对新样本进行分类决策时，根据某一阈值（如 0.5）来判断类别，如果大于阈值则决策为 1 一类，反之决策为 0 一类。当然，根据所面对的具体问题，也可以采用其他的阈值，以调整对两类错误率的偏重或者对两类先验概率的认识，还可以用 2.4 节介绍的 ROC 曲线来考查阈值变化对两类错误率及灵敏度和特异性的影响。在某些应用中，也可以引入一定的拒绝区域，例如只有当输出 y 大于某一阈值（如 0.7）时决策为 1 一类，小于另一阈值（如 0.3）时决策为 0 一类，中间取值时拒绝决策。

2. 多类问题

可以用有多个输出节点的多层感知器网络来实现多类分类。常见的做法是：对于 c 类问题，设计有 c 个输出节点的神经网络，使每个节点对应一类。在训练阶段，对于属于第 i 类的样本，设定第 i 个输出节点的期望输出为 1，其余节点为 0。在对新样本进行识别时，考查各个输出节点，以输出值最大的节点所对应的类别作为对该样本的类别决策。

与两类情况下类似，有时也可以设定一个阈值，要求最大的输出值与其他节点的输出值之差必须大于该阈值才能有把握地决策为这一类，否则不做决策。

这种用 c 个输出节点代表 c 类的做法有时被称作“C 中取一”（1-of-C）编码，因为这实际上是把 c 个类别编码为 c 维的向量，各个类别分别是 $[1,0,\cdots,0]^{\mathrm{T}}$，$[0,1,\cdots,0]^{\mathrm{T}}$，…以此类推。为了使在多类中正确的一类能够尽快地“脱颖而出”，有人针对这种多类问题提出对 BP 算法的一些修改，例如引入约束条件，要求 c 个输出节点的取值之和为 1，在某些应用中可以加快训练速度和减小拒绝率。

多类分类问题还可以有其他的编码方式。例如，可以把每个输出节点看作一个 0、1 二值变量，用 m 个输出节点来编码 c 类，如 3 个节点即可编码 8 类：000，001，010，011，100，101，110，111。类似这种编码可以更节省输出节点数目，但是有可能导致目标函数更复杂，使神经网络训练更加困难。

如果可以在一个网络中同时实现多类分类，在应用中当然有很大方便，但由于各类之间可能会发生相互影响，有可能导致神经网络结构更复杂或导致训练过程更加困难。因此，很多人更喜欢只用神经网络来解决两类问题，而对多类问题则采用在类似 4.9 节中介绍的思路，用多个两类的神经网络来实现。

3. 特征预处理

在应用各种神经网络进行模式识别时，有一点细节需要特别注意，这就是：神经网络的

节点采取的传递函数对特征的取值范围有一定的要求。例如，多层感知器中的 Sigmoid 函数的值域为[0,1]或[−1,1](根据不同的函数形式)，虽然其自变量的取值范围是(−∞,+∞)，但自变量过大或过小会导致函数饱和，不能区分自变量中的变化。例如，如果某特征的物理性质决定了其取值远大于 1，例如在 100 的量级上，若直接把这样的特征作为神经网络的输入，则不论这个特征的取值是 100 还是 200，对神经元来说其值都近似为无穷大，经过一个神经元节点后可能输出值很接近，该特征在分类中就起不到作用。虽然这种情况可能通过采用很小的权值纠正过来，但是如果各权值间差距太大，或者权值本身太小或太大，都不利于学习过程的收敛。

因此，为避免这种情况，人们经常需要把特征进行标准化，通过调整特征的尺度和平移特征的均值，使各特征的取值都基本在 Sigmoid 函数较灵敏的自变量取值范围内。这是一个初学者很容易忽视的问题，经常有刚刚接触神经网络的人因为没有意识到这一问题而无法在应用上得到合理的结果。

特征标准化的具体做法有很多，例如把训练样本中各特征的取值范围都归一化到最小为 0(或−1)、最大为 1，或者把各特征都归一化成固定的均值和标准差，等等，可以根据实际问题灵活选择。

5.4.5 神经网络结构的选择

前面介绍了在给定多层感知器结构的情况下如何根据数据训练各个节点权值的算法，这里我们来讨论如何选择神经网络结构的问题。

人工神经网络有三个要素：神经元的传递函数、网络结构(神经元的数目和相互间的连接形式)和连接权值的学习算法。这三个因素的不同就定义了不同的神经网络模型。多层感知器的结构特点是多个采用 Sigmoid 传递函数的神经元分层排列，网络的具体结构，包括采用几层节点、每层采用多少节点等，都需要根据实际问题来决定。

在前面的讨论中已经看到，对一个任意复杂的非线性函数，总存在适当的多层感知器神经网络能够以任意的精度逼近它。因此，多层感知器几乎可以被看作一种“万能”的分类器，这是它能够得到广泛重视的一个重要原因。但是，这样的理论只是存在性的理论，对于如何针对具体问题找到适当的神经网络结构却没有给出提示。

在基于数据的机器学习中，我们面对的只是训练样本，如何设计适当的神经网络结构才能得到更好的效果，这个问题从神经网络方法诞生之日起就成为人们关注的一个重要问题。到目前为止，人们还没有很好的方法能对这个问题给出一般性的答案，只是在大量的理论和应用研究中积累了一些观察和经验。

一般来说，采用三层神经网络(一个隐层)就可以比较满意地完成各种常见任务。输入层节点数目就是样本特征的维数，输出层节点数目也是根据问题确定的(见上一小节)，因此最主要的不确定因素是中间层的节点数目。

通常，隐层节点数目越大则神经网络的“学习能力”就越强，在训练集上会更容易收敛到一个训练误差较小的解。但是，在样本数目有限的情况下，小的训练误差并不一定能保证在预测未知样本的分类时也有高的准确性，这是所谓推广能力或推广性的问题。过强的学习能力可能会导致神经网络推广能力弱，即出现虽然训练误差很快收敛到很小、但在新的独立

样本上的测试误差却很大的情况，这种情况称作“过学习”或“过适应”(over-fitting)。这就如同小学生学习，如果一个同学很善于死记硬背公式，他可能很快就能把老师教的例题学会，但如果考试时遇到新问题则可能一筹莫展；而另一个同学则善于理解公式的原理，他可能学习得慢一点，但却在解决新问题上有更强的能力。

因此，一味地增加多层感知器的隐层节点数目是不可取的。另外，如果隐层节点数目过少，则神经网络的能力就较小，无法构成复杂的非线性分类面，对于复杂的数据很难得到小的训练误差，当然在测试样本上也无法得到满意的表现，这种情况被称作“欠学习”(under-fitting)。

因此，可以说，多层感知器结构的选择，实际上就是通过选择适当的隐层节点数来取得在过学习和欠学习之间的平衡。如果采用多个隐层，在决定隐层数目和各隐层的节点方面面临的是同样的问题。在统计学习理论中，以第 4 章介绍过的 VC 维概念为核心，对在有限训练样本下得到的经验风险与学习机器的推广能力方面有系统的理论研究，有兴趣的读者可以参阅相关专著和文献。

人们通常有三种做法来选择多层感知器网络的隐层节点数目(和隐层个数)。

第一种基本的做法是根据具体问题进行试探选择。虽然神经网络结构选择缺乏理论指导，但是对于很多问题来说，只要经过几次试算就可能找到比较恰当的隐层节点数目，而且这个数目的一些不大的变化并不会严重影响网络的性能。试算时可以选择几个不同的隐层节点数目，分别对训练样本集进行试验，采用留一法或其他方法交叉验证，根据交叉验证的错误率来选择较好的节点数目。人们也总结出了一些不成文的基本经验，例如通常隐层节点数目应该小于输入维数，当训练样本数较小时应该适当采用少的隐层节点，有人建议采用输入节点数的一半左右，等等。这些经验性的建议可以帮助确定试算的候选值。

第二种做法是根据对问题的先验知识去精心地设计隐层节点的层数和节点数目，例如有人设计了多层的神经网络来进行手写体数字识别，其中的一些隐层是专门为考虑数字的旋转不变性和某些变形不变性而设计的。

第三种做法是试图用算法来确定隐层节点数目。其中，比较有代表性也是比较成功的方法是裁减方法。其基本做法是：初始时采用较多的隐层节点，在采用 BP 算法进行权值学习时增加一条额外的目标，就是要求所有权值的绝对值和或平方和尽可能小。这样，一部分多余的隐层节点的权值会逐渐减小。在学习到一定阶段时，检查各个隐层节点的权值，将权值过小的隐层节点删除，对剩余的神经网络重新进行学习。这一裁减过程可以进行多次，最后得到一个比较合理的网络结构。

与这种方法对应的方法还有从一个较小的网络结构开始、根据学习进展情况逐渐增加隐层节点的做法。

需要清楚的是，不论是哪种方法，都是带有试探性的，需要根据具体问题具体调整网络结构。不能企望有一种方法能够完全自动地确定出神经网络的结构。对于一些复杂的问题，如何更有效地运用神经网络算法有时会成为一件带有“技巧性”的工作，需要对所研究的问题、所面临的数据和所采用的神经网络算法特性有充分的认识才能选择出比较恰当的结构。

5.4.6 前馈神经网络与传统模式识别方法的关系

神经网络与传统的统计模式识别在很多方面是相联系的，它们在某些方面具有一定的等价关系。例如我们已经看到，单层的感知器模型实际上就是一种采用感知准则函数的线

性判别函数，多层感知器则可看作它的非线性推广和发展。人们对前馈型神经网络与统计模式识别的关系开展了大量的研究，这里只对其中一个有代表性的结论进行简要介绍，有兴趣的读者可以通过文献进行深入的学习和研究。

20世纪90年代以来发表的一些理论分析和实验结果表明，很多情况下，多层感知器的输出可以看作对贝叶斯后验概率的估计[①]。例如可以证明，当网络输出采用“C 中取1”的类别编码，并且采用最小均方误差作为训练目标时，多层感知器的输出就是对贝叶斯后验概率的估计。估计的精度受网络的复杂程度、训练样本数、训练样本反映真实分布的程度及类先验概率等多种因素影响。这里仅对两类情况进行讨论。

设网络有 n 个输入节点，输入向量 $\boldsymbol{x}\in\mathbf{R}^n$；对于两类情况，网络只有一个输出节点，记其输出为 $f(\boldsymbol{x},\boldsymbol{w})$，其中 $\boldsymbol{w}$ 表示网络的所有权值。两个类别分别记作 ω_1 和 ω_2，设输出编码为：样本如果属于 ω_1，则期望输出 $d=1$；如果属于 ω_2，则期望输出 $d=0$。设所有训练样本的集合为 $\mathcal{X}$，其中属于 ω_1 类和 ω_2 类的样本的集合分别为 $\mathcal{X}_1$ 和 $\mathcal{X}_2$，则训练的均方误差为

$$E_{\mathrm{s}}(\boldsymbol{w})=\sum_{\boldsymbol{x}\in\mathcal{X}}[f(\boldsymbol{x},\boldsymbol{w})-d(\boldsymbol{x})]^2=\sum_{\boldsymbol{x}\in\mathcal{X}_1}[f(\boldsymbol{x},\boldsymbol{w})-1]^2+\sum_{\boldsymbol{x}\in\mathcal{X}_2}[f(\boldsymbol{x},\boldsymbol{w})]^2 \tag{5-30}$$

把样本 $\boldsymbol{x}$ 看作随机变量，其概率密度函数为 $p(\boldsymbol{x})$，设两类的先验概率分别为 $P(\omega_1)$和 $P(\omega_2)$，$p(\boldsymbol{x}|\omega_i)$，$i=1,2$ 是两类样本的类条件概率密度，$P(\omega_i|\boldsymbol{x})$是样本 $\boldsymbol{x}$ 属于 ω_i 的后验概率。设训练样本数为无穷大，且它们的分布反映真实的概率分布，则式(5-30)的均方误差函数就成为

$$E_{\mathrm{a}}(\boldsymbol{w})=P(\omega_1)\int[f(\boldsymbol{x},\boldsymbol{w})-1)]^2p(\boldsymbol{x}|\omega_1)\mathrm{d}\boldsymbol{x}+P(\omega_2)\int[f(\boldsymbol{x},\boldsymbol{w})]^2p(\boldsymbol{x}|\omega_2)\mathrm{d}\boldsymbol{x} \tag{5-31}$$

利用贝叶斯公式

$$P(\boldsymbol{x}|\omega_1)=\frac{p(\omega_1|\boldsymbol{x})p(\boldsymbol{x})}{P(\omega_1)}$$

和

$$P(\boldsymbol{x})=p(\boldsymbol{x}|\omega_1)P(\omega_1)+p(\boldsymbol{x}|\omega_2)P(\omega_2)$$

式(5-31)可以转化为

$$E_{\mathrm{a}}(\boldsymbol{w})=e^2(\boldsymbol{w})+\int P(\omega_1|\boldsymbol{x})(1-P(\omega_1|\boldsymbol{x}))p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \tag{5-32}$$

其中

$$e^2(\boldsymbol{w})=\int[f(\boldsymbol{x},\boldsymbol{w})-P(\omega_1|\boldsymbol{x})]^2p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \tag{5-33}$$

由于式(5-32)中的后一项与权值 $\boldsymbol{w}$ 无关，因此最小化式(5-32)的均方误差等价于最小化式(5-33)，它是网络实际输出与样本后验概率之间的平方误差的数学期望。因此可以得出

---

① 此部分内容的主要参考文献如下：

Ruck D W, et al. The multilayer perceptron as an approximator to a Bayes optimal discriminate function. *IEEE Trans. on NN*, 1990, 1: 296-298.

Richard M D and Lippmann R P. Neural network classifiers estimate Bayesian a posteriori Probabilities. *Neural Computation*, 1991, 3: 461-483.

Ken-ichi Funahashi. Multilayer neural networks and Bayes decision theory. *Neural Networks*, 1998, 11: 209-213.

结论：当训练样本无穷多时，BP 算法的目标函数等价于神经网络输出与样本后验概率的均方误差，最小化这样的目标函数得到的网络输出就是对样本后验概率的最小均方误差估计。

需要说明，这里得到的最小均方误差估计是在给定的多层感知器结构下的最小，也就是说，是在由确定的神经网络结构所定义的函数集上得到对后验概率均方误差最小的函数，而且也是在当样本无穷多时才成立。在一个实际问题中，这种估计的准确度取决于很多因素，包括样本情况、神经网络结构、学习过程中的收敛情况等。尽管如此，这一结论为我们认识神经网络“黑盒子”的机理提供了重要线索，也为在多类情况下通过比较输出层各个节点的输出值进行分类决策提供了依据。

5.4.7 人工神经网络的一般知识

多层感知器是一种有代表性的人工神经网络模型，还有很多其他类型的人工神经网络。

一般来讲，人工神经网络可以看作由大量简单计算单元（神经元节点）经过相互连接而构成的学习机器，网络中的某些因素，如连接强度（权值）、节点计算特性、网络结构等，可以按照一定的规则或算法根据样本数据进行调整（即训练或学习），最终使网络实现一定的功能。

根据神经网络的结构特点，人们通常把神经网络模型分成三种类型：前馈型神经网络（feedforward network）、反馈型神经网络（feedback network）和竞争学习神经网络（competitive learning network）。

1. 前馈型神经网络

前馈型神经网络的基本特点是，节点按照一定的层次排列，信号按照单一的方向从一层节点传递到下一层节点，网络连接是单向的。多层感知器就是最典型的前馈型神经网络。在这种分层的神经网络中，也可以把每一层看作对特征进行一次加工或变换。如果节点传递函数是线性函数则这种变换就是线性变换，如果是非线性函数则是非线性变换。经过一系列变换后，由网络的最后一层节点来进行判别决策。特别地，如果一个多层感知器的最后一层的节点采用阈值逻辑函数，那么多层感知器实际上就是通过隐层节点对样本特征进行非线性变换，然后在变换空间中采用感知准则函数构建线性分类器。

还有一种较常见的前馈型神经网络，就是径向基函数（RBF）网络。径向函数（radial function）是一种取值只依赖于样本到原点（或到其他中心点）的距离的函数，即 $\varphi(\boldsymbol{x})=\varphi(\|\boldsymbol{x}\|)$，$\|\cdot\|$ 通常用欧氏距离。径向基函数（radial basis function）就是用一组径向函数的加权和来实现某种函数逼近，即

$$y(\boldsymbol{x})=\sum_{i=1}^{N} w_i \varphi(\|\boldsymbol{x}-\boldsymbol{c}_i\|) \tag{5-34}$$

最常用的 RBF 函数是高斯函数，即

$$\varphi(\boldsymbol{x})=\exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{c})^2}{r^2}\right) \tag{5-35}$$

它由其中心点 $\boldsymbol{c}$ 和宽度参数 r 决定。径向基函数神经网络就是用一个三层神经网络的形式实现径向基函数逼近，如图 5-16 所示。

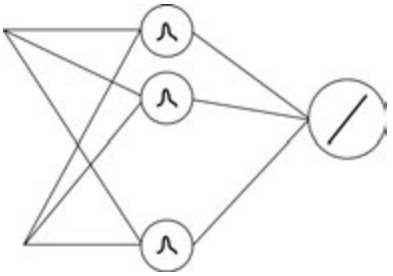


图 5-16 径向基函数神经网络示意图

在径向基函数神经网络中，每一个输入特征都以一定的权值连接到中间层的节点，每个

中间层节点是一个径向基函数，所有径向基函数又通过一定的权值连接到输出节点。对用于函数逼近的径向基网络，输出节点通常是线性函数；而对用于模式识别的径向基网络，输出节点就可以是阈值逻辑函数，如果各个径向基输出的加权和超过一定阈值则决策为第一类，否则决策为第二类。

径向基网络中可以调整的因素主要是径向基函数的个数、每个径向基函数的中心、宽度和各个连接权值。人们可以根据先验知识事先确定径向基函数个数、中心、宽度等参数，也可以采用聚类分析（见第 7 章）等方法来帮助确定。权值可以采用梯度下降法学习。

2. 反馈型神经网络

反馈型神经网络以 Hopfield 网络为代表，如图 5-17 所示。这种神经网络的特点是，输入信号作用于神经元节点上后，各个节点的输出又作为输入反馈到各节点，形成一个动态系统，当系统稳定后读取其输出。Hopfield 网络在函数优化等领域有较多应用，在模式识别领域中可以用于联想记忆、模板匹配等。感兴趣的读者可以学习有关神经网络的教材。

由于 Hopfield 网络和在此基础上发展出的限制性波尔兹曼机（RBM）网络对机器学习领域的奠基性贡献，John Hopfield 和 Geoffrey Hinton 在 2024 年获得了诺贝尔物理学奖。这两种神经网络的基本原理见 10.4 节，在我们的新版《模式识别与机器学习》一书中将对这部分内容进行更详细介绍。

3. 竞争学习神经网络

竞争学习神经网络中，神经元节点通常排列在同一个层次上，没有反馈连接，但是神经网络之间有横向的连接或相互影响，在学习时通过神经元之间的竞争实现特定的映射。典型的竞争学习网络是自组织映射（self-organizing map，SOM）神经网络，如图 5-18 所示。自组织映射神经网络的学习过程是非监督学习，在监督模式识别和非监督模式识别中都有应用，我们将在第 9 章进行介绍。

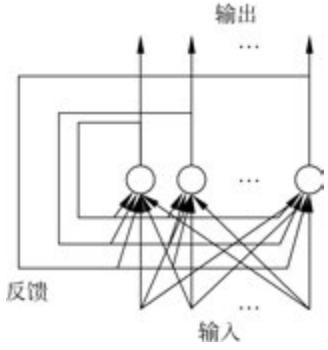


图 5-17 Hopfield 网络示意图

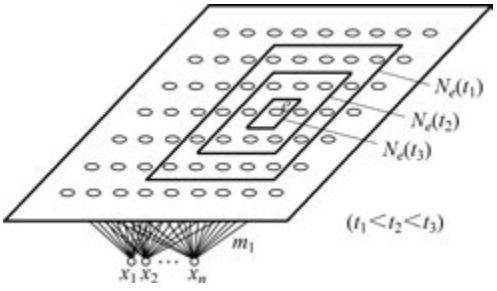


图 5-18 自组织映射神经网络

除了这里提到的典型的神经网络模型，还有很多其他形式的神经网络，它们有些是这些网络的变化，有些是把人工神经网络与其他技术结合起来，例如混合神经网络、模糊神经网络、概率网络等，读者可以参考很多神经网络的专门教材。

以多层感知器为代表的神经网络方法在进入 21 世纪后得到了突飞猛进的发展，发展出了多种深度神经网络模型，为机器学习和模式识别增添了“深度学习”这个极具活力的新方向。

5.5 支持向量机

在第 4 章我们学习了最优分类超平面，即线性的支持向量机，现在我们来讨论如何构造非线性的支持向量机。可以从传统的广义线性判别函数入手来讨论这一问题。

5.5.1 广义线性判别函数

假定有一个如图 5-19 所示的两类问题，样本的特征 x 是一维的，决策规则：如果 $x<b$ 或 $x>a$，则 x 属于 ω_1 类；如果 $b<x<a$ 则 x 属于 ω_2 类。显然，这样的决策无法用线性判别函数来实现，需要设计非线性分类器。

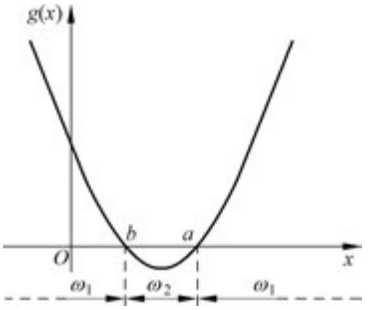


图 5-19 二次判别函数举例

在这个例子中，可以建立一个二次判别函数

$$g(x)=(x-a)(x-b) \tag{5-36}$$

来很好地实现所需的分类决策，决策规则是

$$\text{若 } g(x)\gtrless 0,\quad \text{则 } x\in\begin{cases}\omega_1\\ \omega_2\end{cases}$$

一般来讲，二次判别函数可以写成如下形式：

$$g(x)=c_0+c_1x+c_2x^2 \tag{5-37}$$

如果适当选择 $x\rightarrow\boldsymbol{y}$ 的映射，则可把二次判别函数化为 $\boldsymbol{y}$ 的线性函数，即

$$g(x)=\boldsymbol{a}^{\mathrm{T}}\boldsymbol{y}=\sum_{i=1}^{s}a_iy_i \tag{5-38}$$

式中

$$\boldsymbol{y}=\begin{bmatrix}y_1\\y_2\\y_3\end{bmatrix}=\begin{bmatrix}1\\x\\x^2\end{bmatrix},\quad \boldsymbol{a}=\begin{bmatrix}a_1\\a_2\\a_3\end{bmatrix}=\begin{bmatrix}c_0\\c_1\\c_2\end{bmatrix} \tag{5-39}$$

$g(x)=\boldsymbol{a}^{\mathrm{T}}\boldsymbol{y}$ 称为广义线性判别函数，$\boldsymbol{a}$ 叫作广义权向量。一般来说，对于任意高次判别函数 $g(x)$（这时的 $g(x)$可看作对任意判别函数进行级数展开，然后取其截尾后的逼近），都可以通过适当的变换，化为广义线性判别函数来处理。$\boldsymbol{a}^{\mathrm{T}}\boldsymbol{y}$ 不是 x 的线性函数，但却是 $\boldsymbol{y}$ 的线性函数。$\boldsymbol{a}^{\mathrm{T}}\boldsymbol{y}=0$ 在 $\boldsymbol{Y}$ 空间确定了一个通过原点的超平面。这样，就可以利用线性判别函数的简单性来解决复杂的问题。

遗憾的是，经过这种变换，维数大大增加了。这将使问题很快陷入所谓的“维数灾难”(the curse of dimensionality)，一方面使这种计算变得非常复杂而不可行，另一方面将样本变换到很高维空间中以后，由于样本数目并未增加，在很高维空间中就变得很稀疏，很多算法会因为病态矩阵等问题而无法实现。

例如，如果原特征空间维数是 n，要构造一个广义线性判别函数来实现二阶多项式判别

函数，那么变换后新的特征空间维数将是 $N=n(n+3)/2$，其中包括以下新特征：

$$z^1=x^1,\cdots,z^n=x^n \qquad n \text{ 个特征}$$

$$z^{n+1}=(x^1)^2,\cdots,z^{2n}=(x^n)^2 \qquad n \text{ 个特征}$$

$$z^{2n+1}=x^1x^2,\cdots,z^N=x^nx^{n-1} \qquad n(n-1)/2 \text{ 个特征}$$

如果原始特征的维数更高，或者非线性的阶数更高，则变换后的空间维数会变得非常高，例如在 200 维的原始特征上构造四阶或五阶的多项式，变换后空间的维数将在 10^9 以上。

但是，如果有办法处理维数灾难问题，对特征进行变换，通过在新特征空间里求线性分类器来实现原空间里的非线性分类器的思路仍然是十分有效的。

5.5.2 核函数变换与支持向量机

支持向量机就是采用引入特征变换来将原空间中的非线性问题转化成新空间中的线性问题的，如图 5-20 所示。但是，支持向量机并没有直接计算这种复杂的非线性变换，而是采用了一种巧妙的迂回方法来间接实现这种变换。

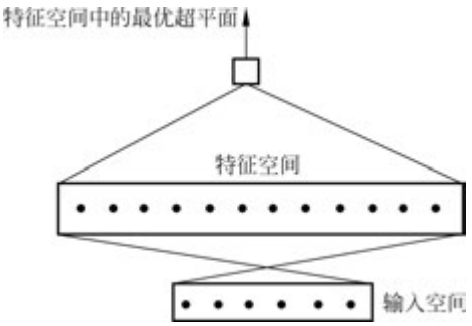


图 5-20 通过非线性变换实现非线性分类器

再次考查第 4 章已经得到的线性支持向量机，它求解的分类器是

$$f(\boldsymbol{x})=\mathrm{sgn}(\boldsymbol{w}\cdot\boldsymbol{x}+b)=\mathrm{sgn}\left(\sum_{i=1}^{n}\alpha_i y_i(\boldsymbol{x}_i\cdot\boldsymbol{x})+b\right) \tag{5-40}$$

其中的 $\alpha_i, i=1,\cdots,n$ 是下列二次优化问题的解：

$$\begin{aligned}&\max_{\boldsymbol{\alpha}} \quad Q(\boldsymbol{\alpha})=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i\cdot\boldsymbol{x}_j)\\ &\text{s.t.} \quad \sum_{i=1}^{n}y_i\alpha_i=0\\ &\qquad 0\leqslant\alpha_i\leqslant C,\quad i=1,\cdots,n\end{aligned} \tag{5-41}$$

其中，b 通过使

$$y_j\left(\sum_{i=1}^{n}\alpha_i(\boldsymbol{x}_i\cdot\boldsymbol{x}_j)+b\right)-1=0 \tag{5-42}$$

成立的样本 $\boldsymbol{x}_j$（即支持向量）求得。

如果我们对特征 $\boldsymbol{x}$ 进行非线性变换，记新特征为 $\boldsymbol{z}=\varphi(\boldsymbol{x})$，则新特征空间里构造的支持向量机决策函数是

$$f(\boldsymbol{x})=\operatorname{sgn}(\boldsymbol{w}^{\varphi}\cdot\boldsymbol{z}+b)=\operatorname{sgn}\left(\sum_{i=1}^{n}\alpha_i y_i(\varphi(\boldsymbol{x}_i)\cdot\varphi(\boldsymbol{x}))+b\right) \tag{5-43}$$

而相应的优化问题变成

$$\begin{aligned}\max_{\boldsymbol{\alpha}}\quad & Q(\boldsymbol{\alpha})=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j(\varphi(\boldsymbol{x}_i)\cdot\varphi(\boldsymbol{x}_j))\\ \text{s.t.}\quad & \sum_{i=1}^{n}y_i\alpha_i=0\\ & 0\leqslant\alpha_i\leqslant C,\quad i=1,\cdots,n\end{aligned} \tag{5-44}$$

定义支持向量的等式成为

$$y_j\left(\sum_{i=1}^{n}\alpha_i y_i(\varphi(\boldsymbol{x}_i)\cdot\varphi(\boldsymbol{x}_j))+b\right)-1=0 \tag{5-45}$$

仔细观察这些公式会发现，在进行变换后，无论变换的具体形式如何，变换对支持向量机的影响是把两个样本在原特征空间中的内积 $(\boldsymbol{x}_i\cdot\boldsymbol{x}_j)$ 变成了在新空间中的内积 $(\varphi(\boldsymbol{x}_i)\cdot\varphi(\boldsymbol{x}_j))$。新空间中的内积也是原特征的函数，可以记作

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j)\xlongequal{\text{def}}(\varphi(\boldsymbol{x}_i)\cdot\varphi(\boldsymbol{x}_j)) \tag{5-46}$$

把它称作核函数。这样，变换空间里的支持向量机就可以写成

$$f(\boldsymbol{x})=\operatorname{sgn}\left(\sum_{i=1}^{n}\alpha_i y_i K(\boldsymbol{x}_i,\boldsymbol{x})+b\right) \tag{5-47}$$

其中，系数 α 是下列优化问题的解：

$$\begin{aligned}\max_{\boldsymbol{\alpha}}\quad & Q(\boldsymbol{\alpha})=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i\cdot\boldsymbol{x}_j)\\ \text{s.t.}\quad & \sum_{i=1}^{n}y_i\alpha_i=0\\ & 0\leqslant\alpha_i\leqslant C,\quad i=1,\cdots,n\end{aligned} \tag{5-48}$$

其中，b 通过满足下式的样本(支持向量)求得：

$$y_j\left(\sum_{i=1}^{n}\alpha_i y_i K(\boldsymbol{x}_i\cdot\boldsymbol{x}_j)+b\right)-1=0 \tag{5-49}$$

对比第 4 章讨论的线性支持向量机，这里通过内积实现的非线性支持向量机中的差别是，原来的内积运算 $(\boldsymbol{x}_i\cdot\boldsymbol{x}_j)$ 变成了核函数 $K(\boldsymbol{x}_i,\boldsymbol{x}_j)$。

从计算角度，不论 $\varphi(\boldsymbol{x})$ 所生成的变换空间维数有多高，这个空间里的线性支持向量机求解都可以在原空间通过核函数 $K(\boldsymbol{x}_i,\boldsymbol{x}_j)$ 进行，这样就避免了高维空间里的计算，而且计算核函数 $K(\boldsymbol{x}_i,\boldsymbol{x}_j)$ 的复杂度与计算内积并没有实质性的增加。

进一步分析就很容易发现，只要知道了核函数 $K(\boldsymbol{x}_i,\boldsymbol{x}_j)$，实际上甚至没有必要知道 $\varphi(\boldsymbol{x})$ 的实际形式。那么，如果要通过设计非线性变换来求解非线性的支持向量机，能否直接设计核函数 $K(\boldsymbol{x}_i,\boldsymbol{x}_j)$ 而不用设计变换 $\varphi(\boldsymbol{x})$ 呢？

泛函空间的有关理论告诉我们，这样做是完全可行的，条件是需要找到能够构成某一变换空间里的内积的核函数。Mercer 条件给出了这一条件：

定理（Mercer 条件） 对于任意的对称函数 $K(\boldsymbol{x},\boldsymbol{x}')$，它是某个特征空间中的内积运算的充分必要条件：对于任意的 $\varphi \neq 0$ 且 $\int \varphi^2(\boldsymbol{x})\mathrm{d}\boldsymbol{x} < \infty$，有

$$\iint K(\boldsymbol{x},\boldsymbol{x}')\varphi(\boldsymbol{x})\varphi(\boldsymbol{x}')\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{x}' > 0 \tag{5-50}$$

因此，选择一个满足 Mercer 条件的核函数，就可以构建非线性的支持向量机。进一步可以证明，这个条件还可以放松为满足如下条件的正定核（positive definite kernels）：$K(\boldsymbol{x}_i,\boldsymbol{x}_j)$ 是定义在空间 X 上的对称函数，且对任意的训练数据 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_m \in X$ 和任意的实系数 $a_1,\cdots,a_m \in \mathbf{R}$，都有

$$\sum_{i,j} a_i a_j K(\boldsymbol{x}_i,\boldsymbol{x}_j) \geqslant 0 \tag{5-51}$$

对于满足正定条件的核函数，肯定存在一个从 X 空间到内积空间 H 的变换 $\varphi(\boldsymbol{x})$，使得

$$K(\boldsymbol{x},\boldsymbol{x}') = (\varphi(\boldsymbol{x}) \cdot \varphi(\boldsymbol{x}')) \tag{5-52}$$

这样构成的空间是在泛函中定义的所谓的可再生核希尔伯特空间（reproducing kernel Hilbert space，RKHS）。更多关于 RKHS 的知识可以参考相关泛函教材或 Bernhard Schölkopf 与 Alexander Smola 的专著 *Learning with Kernels*（MIT Press，Cambridge，MA，2002）。

1. 常用的核函数形式

采用不同的核函数就得到不同形式的非线性支持向量机。目前较常用的核函数主要有三种类型。

第一种是多项式核函数：

$$K(\boldsymbol{x},\boldsymbol{x}') = ((\boldsymbol{x} \cdot \boldsymbol{x}') + 1)^q \tag{5-53}$$

采用这种核函数的支持向量机实现的是 q 阶的多项式判别函数。

第二种是径向基（RBF）核函数：

$$K(\boldsymbol{x},\boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{\sigma^2}\right) \tag{5-54}$$

采用它的支持向量机实现与径向基网络形式相同的决策函数。

第三种是 Sigmoid 函数：

$$K(\boldsymbol{x},\boldsymbol{x}') = \tanh(v(\boldsymbol{x} \cdot \boldsymbol{x}') + c) \tag{5-55}$$

采用这种核函数的支持向量机在 v 和 c 满足一定取值条件的情况下等价于包含一个隐层的多层感知器神经网络。

在采用径向基核函数时，支持向量机能够实现一个径向基函数神经网络的功能，但是二者有很大不同。径向基函数神经网络通常需要靠启发式的经验或规则来选择径向基的个数、每个径向基的中心位置、径向基函数的宽度等，只有权系数是通过学习算法得到的；而在支持向量机中，每一个支持向量构成一个径向基函数的中心，其位置、宽度、个数以及连接权值都可以通过训练过程来确定。

对于采用 Sigmoid 核函数的支持向量机，实现的是一个三层神经网络，隐层节点个数就是支持向量的个数，所以，支持向量机等价地实现了对神经网络隐层节点数目的自动选择。

2. 核函数及其参数的选择

支持向量机通过选择不同的核函数实现不同形式的非线性分类器。当核函数选为线性内积时就是线性支持向量机。

针对一个具体的应用问题，应该采用什么样的核函数呢？人们对这个问题进行了很多尝试，但是目前仍没有一个满意的答案。其实这也很自然，具体问题具体分析是辩证唯物主义的一个基本原理，对于模式识别问题也一样，核函数也需要针对具体问题来具体选择，很难有一个一般性的准则。同时，有实验表明，在一些实际数据上，从分类错误率和所选择出的支持向量角度来看，不同类型的核函数有可能达到同样的效果，但在同一类型的核函数中要选择适当的参数。对于多项式核函数，参数就是多项式核的阶数 q；对 RBF 核函数，参数是核函数的宽度 σ；对 Sigmoid 核函数，参数是 v 和 c。

统计学习理论中，根据一系列关于推广性的理论，给出了针对具体的样本集选择核函数参数的方法，但是在实际应用中，这些方法不容易实现，因此，人们采用更多的是启发式方法或者累试的方法来选择核函数参数。例如，最流行的支持向量机软件之一 LibSVM(见 5.5.4 节)提供了一种功能，按照规定的网格自动用各种参数取值来进行试验，根据每个参数取值下的留一法交叉验证结果选择最佳的参数。

通常，对于很多应用来说，核函数参数的选择并不是十分困难，人们往往手工尝试几种选择便会找出比较合适的参数。一条基本的经验是，应该首先尝试简单的选择，例如首先尝试线性核，当结果不满意时才考虑非线性核；如果选择 RBF 核函数，则首先应该选用宽度比较大的核，即 σ 比较大，宽度越大越接近线性，然后再尝试减小宽度，增加非线性程度。

3. 核函数与相似性度量

支持向量机的基本思想可以概括为，首先通过非线性变换将输入空间变换到一个高维空间，然后在这个新空间中求最优分类面即最大间隔分类面，而这种非线性变换是通过定义适当的内积核函数实现的。

支持向量机求得的分类函数，形式上类似于一个神经网络，其输出是若干中间层节点的线性组合，而每一个中间层节点对应于输入样本与一个支持向量的内积，因此早期也被叫作支持向量网络，如图 5-21 所示。

支持向量机的决策过程也可以看作一种相似性比较的过程。首先，输入样本与一系列模板样本进行相似性比较，模板样本就是训练过程中决定的支持向量，而采用的相似性度量就是核函数。样本与各支持向量比较后的得分进行加权后求和，权值就是训练时得到的各支持向量的系数 α 与类别标号的乘积。最后根据加权求和值的大小来进行决策。

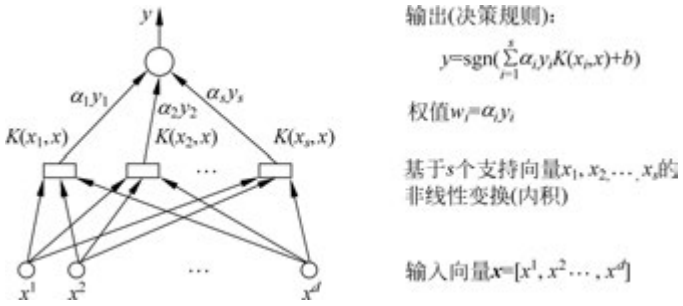


图 5-21 支持向量机的决策函数

采用不同的核函数,可以看作选择不同的相似性度量,线性支持向量机就是采用欧氏空间中的内积作为相似性度量。根据这一思想,人们除了可以选择上面介绍的常用的核函数形式外,在一些实际问题中还可以根据相关领域的专门知识定义一些特殊的核函数。例如,人们在用支持向量机进行生物序列的分类时,可以根据专门的生物知识定义序列样本间的相似性度量,例如采用编辑距离作为相似性度量,这样构造的支持向量机能够更好地与专业知识相结合,往往可以取得更好的效果①。

需要注意的是,当选用新的核函数或自己定义核函数时,需要考虑所定义的核函数是否满足 Mercer 条件。如果不满足,则可能导致支持向量机的目标函数不再是凸函数,导致解不唯一。有些支持向量机程序可以对这种情况报错,或者不能正常结束程序。对于某些核函数,从理论上证明是否满足 Mercer 条件可能会有困难,此时可以考虑用一些仿真数据检查正定核的条件。在很多情况下,即使所采用核函数可能不严格满足正定条件,如果它能较好地反映应用问题中的专业知识,仍然可以取得不错的结果。

把核函数看作相似性度量还有另外一个好处,就是可以结合专业知识来对非数值特征进行编码。再拿 DNA 序列的分类作为例子。一个 DNA 序列样本是一个由 A、C、G、T 字母组成的字符串,要对它进行分类,通常就需要把这个字符串编码成一个数字取值的向量,以便可以使用本书前面介绍的各种方法进行分类。最常见的编码方式是所谓正交编码,即用一个四维向量代表序列的一个位置,A 用 1000 代表、C 用 0100 代表、G 用 0010 代表、T 用 0001 代表。这样,如果序列的长度是 n,我们就得到一个维数为 $4n$ 的特征向量,然后就可以用各种模式识别方法来进行分类。显然,这种方法看上去比较"笨拙",没有体现生物序列的特殊含义。如果用支持向量机对它们进行分类,由于在支持向量机训练和决策阶段中都不需要直接使用样本特征,只要能够计算两两样本间的核函数就可以,可以根据生物序列的含义定义序列片段间的相似性度量,从而避免了采用没有生物意义的编码。对于蛋白质序列,有些氨基酸之间有一定程度的可替代性,这样定义的核函数就更能反映问题背后的内在生物学原理。

4. 维数与推广能力

支持向量机通过采用核函数作为内积,间接地实现了对特征的非线性变换,因此避开了

① 例如,这样的一个例子可以见 Li H and Jiang T. A class of edit kernels for SVMs to predict translation initiation sites in Eukaryotic mRNAs. *Journal of Computational Biology*, 2005,12(6): 702-718.

在高维空间进行计算。然而，即使不直接地进行非线性变换，核函数的作用仍然是把样本的特征映射到高维空间，例如，如果用 RBF 核函数，映射后的空间实际是无穷维。这样，利用有限的样本在很高维甚至无穷维的空间里构造分类器，其推广能力仍然是一个很大的问题。

在第 4 章中讲到，支持向量机通过最大化分类间隔来控制函数集的 VC 维，使得在高维空间里的函数集的 VC 维可以大大低于空间的维数，从而保证好的推广能力。正因为有这一性质，才使得支持向量机可在采用核函数内积后仍然可以有好的推广能力。

5.5.3 支持向量机早期应用举例

为了帮助读者直观理解非线性支持向量机的解的情况，这里给出几个二维空间里用不同核函数取得的支持向量机解的图例。图 5-22 是对两组数据用二阶多项式核函数取得的结果，图中两类样本分别用小圆圈和黑点表示，虚线标出了支持向量机的决策面，用大圆圈套住的样本是支持向量，打叉的样本是分错的样本。图 5-23 给出的是一个用高斯径向基核函数的例子，除了分类面，图中还标出了决策函数取 1 和－1 的边界线，并用灰度值表示出了决策函数绝对值的大小。

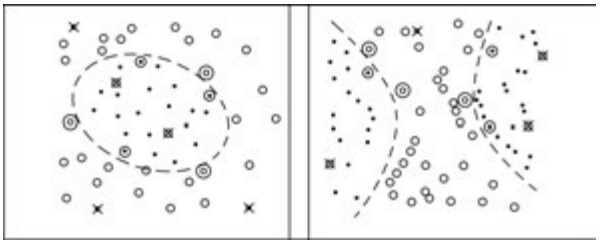


图 5-22 二阶多项式核函数支持向量机的结果举例（引自文献[5]）

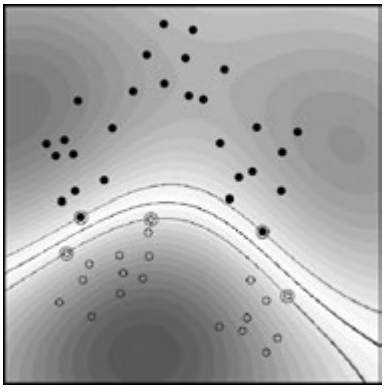


图 5-23 径向基核函数支持向量机的结果举例

支持向量机最早的实际应用是由其创始人 Vapnik 带领的 AT&T 实验室研究小组进行的手写数字识别的实验[①]。当时所用的数据是美国邮政 USPS 手写数字数据库。这个数据库包含 7300 个训练样本和 2000 个测试样本，每个样本都是 16×16 的点阵，构成 256 维原始特征。数据的可识别性比较差，图 5-24 给出了其中一些样本的例子。表 5-1 给出了当时传统方法在这组数据上报道的最好结果，其中的两层神经网络的结果是在多种两层神经网络（一个隐层）中的最好者，而 LeNet1 是一个专门针对这组数据设计的五层神经网络，其中利用了一些字符的旋转、平移不变性信息。

① Cortes C and Vapnik V N. Support vector networks. *Machine Learning*, 1995, 20(3): 273-297.

图 5-24 USPS 手写数字样本举例(引自文献[5])

表 5-1 传统方法在 USPS 上的实验性能

分 类 器	测试错误率	分 类 器	测试错误率
人工分类	2.5%	两层神经网络(一个隐层)	5.9%
决策树方法	16.2%	LeNet1 五层神经网络	5.1%

Vapnik 等在这组数据上对支持向量机的性能进行了系统的实验。他们选用了三组核函数,分别如下:

多项式核函数: $K(\boldsymbol{x},\boldsymbol{x}')=\left(\frac{1}{256}(\boldsymbol{x}\cdot\boldsymbol{x}')\right)^q$;

径向基核函数: $K(\boldsymbol{x},\boldsymbol{x}')=\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{256\sigma^2}\right)$;

Sigmoid 核函数：$K(\boldsymbol{x},\boldsymbol{x}')=\tanh\left(\frac{b(\boldsymbol{x}\cdot\boldsymbol{x}')}{256}-c\right)$；

并对每一种核函数试用了多套参数，最后得到的最好结果及相应的参数见表 5-2。

表 5-2 支持向量机在 USPS 数据上的实验结果

支持向量机类型	内积函数中的参数	支持向量个数	测试错误率
多项式内积	$q=3$	274	4.0%
径向基函数内积	$\sigma^2=0.3$	291	4.1%
Sigmoid 内积	$b=2,c=1$	254	4.2%

可以看到，支持向量机比其他方法在分类性能上表现出了明显的优势。同时还看到，采用三种不同的核函数取得的效果非常接近，而且他们的实验还发现，三种核函数下得到的支持向量样本 80%以上都是重合的。这些现象提示，支持向量机对于核函数的选择具有一定的不敏感性。在研究用 Parzen 窗法估计概率密度函数时，人们看到，窗函数形式的选择对结果的影响不如窗宽参数的影响大，在支持向量机中也看到类似的现象，只要核函数的参数选择恰当，不同的核函数可以达到同样的效果。这些现象目前还只是部分实验中的观察，其背后的理论性质仍然在研究中。

支持向量机方法的最初发表是在 1992 年和 1995 年，当时并没有立即引起很大的效应。但是，从 20 世纪 90 年代末开始，机器学习和模式识别界迅速掀起了一个支持向量机研究和应用的热潮。较典型的应用是在字符识别、人脸图像识别、文本识别、基因表达数据分析等方面，在其他各个领域也有很多的应用。与传统模式识别方法以及人工神经网络方法相比，支持向量机的最主要特点是它能够在样本数相对较少、特征维数高的情况下仍然取得很好的推广能力。这种情况在图像、文本、基因表达等领域的应用中最常见，因此也是在这些领域中最容易体现出支持向量机方法的优势。

5.5.4 支持向量机的实现算法

支持向量机需要求解的是式(5-48)定义的关于$\boldsymbol{\alpha}$ 的二次优化函数。这是一个有线性约束的二次优化问题，有唯一的最优解，这与多层感知器神经网络相比是一个优势。而且，问题的计算复杂度是由样本数目决定的，计算复杂度不取决于样本的特征维数和所采用的核函数形式。

一般来讲，条件极值问题可以用单纯形法或罚函数法求解。例如，我们可以通过引入惩罚项将约束条件构造到无约束的罚函数中，对于二次优化问题可用共轭梯度法求解。

但是，对样本数目 n，式(5-48)的目标函数中涉及 $n\times n$ 矩阵的运算，当 n 较大时，需要计算和存储的矩阵就很大，常规的优化方法很难工作。为了解决这一问题，人们研究了多种策略，主要思想是将大的优化问题分解为若干子问题，按照某种迭代策略反复求解子问题，最终使结果收敛到原问题的最优解。不同的划分子问题的方法和迭代求解的方法就产生了不同的支持向量机实现算法。

值得高兴的是，支持向量机的诞生正值以 GNU 计划为代表的开放源码运动蓬勃发展的时期，正当很多人自己尝试编写支持向量机程序但效果不理想的时候，一些优秀的优化问

题研究者开发了多种支持向量机实现软件，并通过因特网向广大研究者免费发布。在这些软件中，比较有影响的是 SVM^{light}、SVMTorch 和 LibSVM①。这些软件都经历了多个版本的演化，现在已经非常完善。除了基本的支持向量机算法外，很多软件还包含一些扩展的算法和功能。

例如，LibSVM 中包含了单类支持向量机(One-class SVM)②和用支持向量机实现多类分类的功能，还包含了交叉验证、用网格试算方法选择核函数参数的功能。LibSVM 的软件和文档可以从 http://www.csie.ntu.edu.tw/~cjlin/libsvm/网站获得。

SVM^{light} 的网址是 http://svmlight.joachims.org/，SVMTorch 的网址是 http://bengio.abracadoudou.com/projects/SVMTorch.html。除了这些软件以外，还有多种运行在 MATLAB、R 等通用计算平台上的支持向量机软件，其中很多软件都可以在 http://www.kernel-machines.org/software 网页上找到其链接。

5.5.5 多类支持向量机

在第 4 章里曾经介绍了实现多类分类的两种做法：一是用多个两类分类器实现多类分类，二是直接设计多类分类器。利用支持向量机进行多类分类也同样有这样两种做法。

利用多个支持向量机来进行多类分类是比较常用的做法，具体做法在 4.9.1 节已经介绍。这里讨论一种直接设计多类 SVM 分类器的方法，称作多类支持向量机(multicategory SVM)③。

首先，支持向量机可以用正则化(regularization)的框架来重新表述如下：

设有训练样本集$\{(\boldsymbol{x}_i, y_i), i=1,\cdots,n\}$，$\boldsymbol{x}_i \in \mathbf{R}^d$ 是样本的特征，$y_i=\{1,-1\}$是样本的类别标号。待求函数 $f(\boldsymbol{x})=h(\boldsymbol{x})+b$，$h \in H_K$，$H_K$ 是由核函数 K 定义的可再生希尔伯特空间。决策规则是 $g(\boldsymbol{x})=\mathrm{sgn}(f(\boldsymbol{x}))$。支持向量机求解的是这样的 f，它最小化以下的目标函数：

$$\frac{1}{n}\sum_{i=1}^{n}(1-y_i f(\boldsymbol{x}_i))_+ + \lambda \| h \|^2_{H_K} \tag{5-56}$$

其中，第一项是支持向量机原来的目标函数式(4-114)中的松弛因子项，第二项是对函数复杂性的惩罚，对应于式(4-114)中间隔最大化的项，λ 的作用相当于式(4-114)中的 C，它调节

---

① 这些方法的主要参考文献分别如下。

SVM^{light}：Thorsten Joachims. Making large-scale SVM learning practical. In: Schoekopf B et al., eds. *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

SVMTorch：Ronan Collobert and Samy Bengio. SVMTorch: support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 2001, 1: 143-160.

LibSVM：Fan R E, Chen P H, and Lin C J. Working set selection using the second order information for training SVM. *Journal of Machine Learning Research*, 2005, 6: 1889-1918.

② Schölkopf B, Platt J, Shawe-Taylor J, Smola A J, and Williamson R C. Estimating the support of a high-dimensional distribution. *Neural Computation*, 2001, 13: 1443-1471.

③ Yoonkyung Lee, Yi Lin and Grace Wahba. Multicategory Support Vector Machines. *Technical Report* No. 1043, *Department of Statistics*, *University of Wisconsin*, *Madison*, Sept. 29, 2001.

在函数复杂性和在训练样本上的分类精度之间的平衡。

如果样本的类别标号 y 和待求的函数 $f(\boldsymbol{x})$都从标量变为向量，则上述表述就可以用于多类分类问题。

对于 k 类问题，$\boldsymbol{y}_i$ 是一个 k 维向量，如果样本 $\boldsymbol{x}_i$ 属于第 j 类，则 $\boldsymbol{y}_i$ 的第 j 个分量为 1，其余分量为 $-\dfrac{1}{k-1}$，这样，$\boldsymbol{y}_i$ 的各分量值总和为 0。举例来说，如果 $k=3$，则

$$\boldsymbol{y}_i=\begin{cases}(1,-1/2,-1/2), & \boldsymbol{x}_i\in\omega_1\\(-1/2,1,-1/2), & \boldsymbol{x}_i\in\omega_2\\(-1/2,-1/2,1), & \boldsymbol{x}_i\in\omega_3\end{cases}$$

待求函数为 $f(\boldsymbol{x})=(f_1(\boldsymbol{x}),\cdots,f_k(\boldsymbol{x}))$，它的各分量之和须为 0，即 $\sum_{j=1}^{k}f_j(\boldsymbol{x})=0$，且每一个分量都定义在核函数可再生希尔伯特空间中：

$$f_j(\boldsymbol{x})=h_j(\boldsymbol{x})+b_j,\quad h_j\in H_K$$

把多个类别编码成这样的向量标签后，多类支持向量机就是求 $f(\boldsymbol{x})=(f_1(\boldsymbol{x}),\cdots,f_k(\boldsymbol{x}))$，使下列目标函数达到最小：

$$\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{L}(\boldsymbol{y}_i)\cdot(f(x_i)-\boldsymbol{y}_i)_+ +\frac{\lambda}{2}\sum_{j=1}^{k}\|h_j\|_{H_K}^2 \tag{5-57}$$

其中，$\boldsymbol{L}(\boldsymbol{y}_i)$是损失矩阵 $\boldsymbol{C}$ 与样本类别 $\boldsymbol{y}_i$ 相对的行向量。损失矩阵 $\boldsymbol{C}$ 是一个 $k\times k$ 矩阵，它的对角线上是 0，其余元素均为 1，例如 $k=3$ 情况下的损失矩阵是

$$\boldsymbol{C}=\begin{pmatrix}0&1&1\\1&0&1\\1&1&0\end{pmatrix}$$

得到函数 $f(\boldsymbol{x})$后，类别决策规则是 $g(\boldsymbol{x})=\underset{j}{\operatorname{argmax}}\,f_j(\boldsymbol{x})$，即决策为 $f(\boldsymbol{x})$各分量中取值最大的分量对应的类别（“argmax”表示取得后面函数最大值的下标）。

可以证明，前面讨论的两类的支持向量机可以看作这种多类支持向量机在 $k=2$ 情况下的特例。有兴趣的读者可以作为练习自己证明。

5.5.6 用于函数拟合的支持向量机——支持向量回归

支持向量机最初是作为一个分类机器提出来的，但很快就被推广到用于实函数的拟合问题上。虽然这已经超出了狭义的模式识别研究的范畴，但为了使读者对支持向量机有一个比较全面的认识，这里也简要地介绍一下用于函数估计的支持向量机，也有人称作支持向量回归（support vector regression，SVR），相应地把用于分类的支持向量机称作支持向量分类（support vector classification，SVC）。

在讨论用于分类的支持向量机时，首先是从线性支持向量机即最优分类面开始的。类似地，考查用线性回归函数

$$f(\boldsymbol{x})=\boldsymbol{w}\cdot\boldsymbol{x}+b \tag{5-58}$$

来拟合训练数据$\{(\boldsymbol{x}_i,y_i),i=1,\cdots,n\}$，$\boldsymbol{x}_i\in\mathbf{R}^d$，$y_i\in\mathbf{R}$。与分类时首先考虑样本线性可分

的情况类似，这里也首先考虑所有样本都可以在一定的精度 ε 范围内用线性函数拟合的情况，即

$$\begin{cases} y_i - \boldsymbol{w} \cdot \boldsymbol{x}_i - b \leqslant \varepsilon, \\ \boldsymbol{w} \cdot \boldsymbol{x}_i + b - y_i \leqslant \varepsilon, \end{cases} \quad i = 1, 2, \cdots, n \tag{5-59}$$

这里，因为拟合的误差可能是两个方向的，所以对每个样本有两个不等式。

与支持向量分类时控制最大化分类间隔类似，这里也要求最小化 $\frac{1}{2}\| \boldsymbol{w} \|^2$，它对应的是要求回归函数最平坦。这样，就有了用于回归的支持向量机的原问题

$$\min_{\boldsymbol{w}, b} \frac{1}{2} \| \boldsymbol{w} \|^2$$

$$\text{s. t.} \begin{cases} y_i - \boldsymbol{w} \cdot \boldsymbol{x}_i - b \leqslant \varepsilon, \\ \boldsymbol{w} \cdot \boldsymbol{x}_i + b - y_i \leqslant \varepsilon, \end{cases} \quad i = 1, 2, \cdots, n \tag{5-60}$$

如果允许拟合误差超过 ε，则可以与分类时类似地引入松弛因子，只是这里需要对上、下两个方向分别引入一个松弛因子，使约束条件变成

$$\begin{cases} y_i - \boldsymbol{w} \cdot \boldsymbol{x}_i - b \leqslant \varepsilon + \xi_i^*, \\ \boldsymbol{w} \cdot \boldsymbol{x}_i + b - y_i \leqslant \varepsilon + \xi_i, \end{cases} \quad i = 1, 2, \cdots, n \tag{5-61}$$

$$\xi_i \geqslant 0, \xi_i^* \geqslant 0, i = 1, \cdots, n \tag{5-62}$$

目标函数变成

$$\min_{\boldsymbol{w}, b} \frac{1}{2} \| \boldsymbol{w} \|^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) \tag{5-63}$$

其中，常数 C 控制着对超出误差限样本的惩罚与函数的平坦性之间的折中。这样的目标函数，实际是对拟合误差采用了如图 5-25 所示的 ε-不敏感损失函数

$$|y - f(\boldsymbol{x}, \alpha)|_\varepsilon = \begin{cases} 0, & |y - f(\boldsymbol{x}, \alpha)| \leqslant \varepsilon \\ |y - f(\boldsymbol{x}, \alpha)| - \varepsilon, & \text{其他} \end{cases} \tag{5-64}$$

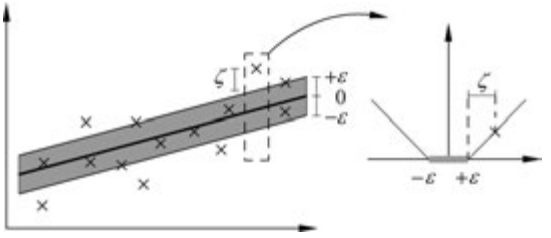


图 5-25 ε-不敏感损失函数

经过与支持向量分类中类似的推导，可以得到由式(5-63)和式(5-61)、式(5-62)构成的优化问题的对偶问题，即

$$\max_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*} W(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = -\varepsilon \sum_{i=1}^{l} (\alpha_i^* + \alpha_i) + \sum_{i=1}^{l} y_i (\alpha_i^* - \alpha_i) - \frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)(\boldsymbol{x}_i \cdot \boldsymbol{x}_j) \tag{5-65}$$

$$\text{s.t.} \quad \sum_{i=1}^{l}\alpha_i^* = \sum_{i=1}^{l}\alpha_i$$
$$0 \leqslant \alpha_i^* \leqslant C, \quad i=1,2,\cdots,l$$
$$0 \leqslant \alpha_i \leqslant C, \quad i=1,2,\cdots,l$$

回归函数的权值与对偶问题中的系数的关系是

$$\boldsymbol{w} = \sum_{i=1}^{l}(\alpha_i^* - \alpha_i)\boldsymbol{x}_i \tag{5-66}$$

得到的回归函数是

$$f(\boldsymbol{x}) = \boldsymbol{w}\cdot\boldsymbol{x} + b = \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)(\boldsymbol{x}_i\cdot\boldsymbol{x}) + b^* \tag{5-67}$$

与分类情况下类似，这里的多数 α_i 和 α_i^* 都为 0，非零的 α_i 或 α_i^*（二者不可能同时非零）对应的样本是支持向量，它们或者落在离回归函数距离恰为 ε 的“ε-管道”上，相当于分类情况下离分类面距离最近的样本，或者落在“ε-管道”之外，相当于分类情况下的错分样本，且这些样本对应的 α_i 或 α_i^* 等于 C。

与分类支持向量机相同，也可以通过核函数间接进行非线性变换来实现非线性的支持向量机函数拟合。得到的拟合函数的形式如

$$f(\boldsymbol{x}) = \sum_{i=1}^{l}\beta_i K(\boldsymbol{x},\boldsymbol{x}_i) + b \tag{5-68}$$

其中，$K(\cdot,\cdot)$是核函数，系数 $\beta_i \overset{\text{def}}{=\!=} \alpha_i^* - \alpha_i$，$i=1,\cdots,l$ 是以下优化问题的解：

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} W(\boldsymbol{\alpha},\boldsymbol{\alpha}^*) = &-\varepsilon\sum_{i=1}^{l}(\alpha_i^* + \alpha_i) + \sum_{i=1}^{l}y_i(\alpha_i^* - \alpha_i) - \\ &\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(\boldsymbol{x}_i,\boldsymbol{x}_j)\\ &\text{s.t.} \quad \sum_{i=1}^{l}\alpha_i^* = \sum_{i=1}^{l}\alpha_i\\ &0 \leqslant \alpha_i^* \leqslant C, \quad i=1,2,\cdots,l\\ &0 \leqslant \alpha_i \leqslant C, \quad i=1,2,\cdots,l\end{aligned} \tag{5-69}$$

由于支持向量回归的优化问题与支持向量分类基本相同，目前的多数支持向量机软件，如 SVM$^{\text{light}}$、SVMTorch 和 LibSVM 等都包含了支持向量回归的功能。

5.6 核函数机器

5.6.1 大间隔机器与核函数机器

支持向量机有两个核心的思想，一是通过最大化分类间隔来保证最好的推广能力，二是通过核函数定义的内积函数来间接地实现对特征的非线性变换，用变换空间中的线性问题来求解原空间中的非线性问题。这两个思想给了人们很大的启示。

正如在 5.5.1 节中所讨论的，实际上，对于任何的线性方法，如果把特征进行适当的变换，就可以得到相应的非线性方法。

但是，这种变换带来两个层面上的问题。

一是计算层面上的问题：多数情况下，要通过变换把非线性问题线性化，样本的特征维数必然会升高，很多情况下是随着样本原特征维数的增加以及非线性程度的增加而呈指数增加，此时很多算法将无法运行。

二是概念层面上的问题：样本的维数升高了，但是样本数并没有增加，所要求解的决策函数的复杂度却大大增加。一个基本的常识是，在有限的样本下，所要确定或估计的参数越多，这种估计的可信性就越低。在一个很高维的空间里，可能会很容易地找到一个能够把有限的训练样本分开的解，但是这样的解在面对新样本时可能推广能力很差。

支持向量机通过其“大间隔”和“核函数”的思想有效地解决了这两个问题：通过采用核函数，运算不需要在高维空间里进行，避免了计算上的困难；通过控制最大化分类间隔，使它即使在很高维的空间里仍能保持最好的推广能力。

借用这两个主要思想，人们对一系列传统的线性方法进行了发展。基本做法是，如果原方法能表述成只涉及样本的内积计算的形式，那么，就可以通过采用核函数内积实现非线性变换，而通过引入适当的间隔约束来控制非线性机器的推广能力。人们把这些方法统称作大间隔方法(large-margin method)或核函数方法(kernel method)，现在一般称作核函数方法或核方法。

这里只介绍最有代表性的核 Fisher 判别方法，在第 8 章再介绍用于特征变换的核主成分分析(KPCA)方法。

5.6.2 核 Fisher 判别

在 4.4 节已经介绍了 Fisher 线性判别方法，这是一种经典的线性分类方法，方法的原理和实现都很简单，在很多实际问题上都有很好的应用效果，因此在新方法层出不穷的今天仍然是一种很受重视的方法。为了把这种方法推广到非线性情况，人们发展了核 Fisher 判别(kernel Fisher's discriminant，KFD)方法[①]。

首先回顾 Fisher 线性判别的原理。Fisher 线性判别就是寻找最优的投影方向，使下面的准则最大化

$$\max_{\boldsymbol{w}} \quad J(\boldsymbol{w}) = \frac{\boldsymbol{w}^{\mathrm{T}} \boldsymbol{S}_{\mathrm{B}} \boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}} \boldsymbol{S}_{\mathrm{W}} \boldsymbol{w}} \tag{5-70}$$

其中，$\boldsymbol{S}_{\mathrm{B}}$ 和 $\boldsymbol{S}_{\mathrm{W}}$ 分别是如下定义的类间离散度矩阵和类内离散度矩阵：

$$\boldsymbol{S}_{\mathrm{B}} = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^{\mathrm{T}} \tag{5-71}$$

$$\boldsymbol{S}_{\mathrm{W}} = \sum_{q=1,2} \sum_{\boldsymbol{x}_i \in \omega_q} (\boldsymbol{x}_i - \boldsymbol{m}_q)(\boldsymbol{x}_i - \boldsymbol{m}_q)^{\mathrm{T}} \tag{5-72}$$

其中，$\boldsymbol{m}_1$、$\boldsymbol{m}_2$ 分别是两类的样本均值向量。如果类内离散度矩阵可逆，则 Fisher 线性判别

① Mika S, Ratsch G, Weston J, Schölkopf B and Muller K R. Fisher discriminant analysis with kernels. *Neural Networks for Signal Processing* IX, pp. 41-48, IEEE, 1999.

的解是

$$\boldsymbol{w}=\boldsymbol{S}_{\mathrm{W}}^{-1}(\boldsymbol{m}_1-\boldsymbol{m}_2) \tag{5-73}$$

下面考虑如何通过非线性变换来设计非线性的 Fisher 判别，基本思想就是采用广义线性判别函数的思想，首先将样本映射到高维空间，然后在新空间里求解 Fisher 线性判别。

对样本 $\boldsymbol{x}$ 进行非线性变换 $\boldsymbol{x}\rightarrow\Phi(\boldsymbol{x})\in F$。在变换后的空间 F 中，Fisher 线性判别的准则为

$$J(\boldsymbol{w})=\frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{B}}^{\Phi}\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{W}}^{\Phi}\boldsymbol{w}} \tag{5-74}$$

这里的 $\boldsymbol{w}$ 是 F 空间里的权值向量，$\boldsymbol{S}_{\mathrm{B}}^{\Phi}$ 和 $\boldsymbol{S}_{\mathrm{W}}^{\Phi}$ 分别是 F 空间里的类间离散度矩阵和类内离散度矩阵，即

$$\boldsymbol{S}_{\mathrm{B}}^{\Phi}=(\boldsymbol{m}_1^{\Phi}-\boldsymbol{m}_2^{\Phi})(\boldsymbol{m}_1^{\Phi}-\boldsymbol{m}_2^{\Phi})^{\mathrm{T}} \tag{5-75}$$

$$\boldsymbol{S}_{\mathrm{W}}^{\Phi}=\sum_{i=1,2}\sum_{x\in\omega_i}(\boldsymbol{\Phi}(\boldsymbol{x})-\boldsymbol{m}_i^{\Phi})(\boldsymbol{\Phi}(\boldsymbol{x})-\boldsymbol{m}_i^{\Phi})^{\mathrm{T}} \tag{5-76}$$

$\boldsymbol{m}_i^{\Phi}$ 是 F 空间里各类样本的均值，即

$$\boldsymbol{m}_i^{\Phi}=\frac{1}{l_i}\sum_{j=1}^{l_i}\boldsymbol{\Phi}(\boldsymbol{x}_j^i) \tag{5-77}$$

其中，l_i 是第 i 类样本数，l 是总样本数。

如果直接在变换空间里求解 Fisher 线性判别，由于变换复杂、维数高，这种做法没有优势，很多情况下甚至不可行。下面把这个问题转化成通过核函数求解的形式。

根据可再生核希尔伯特空间的有关理论可以知道，上述问题的任何解 $\boldsymbol{w}\in F$ 都处在 F 空间中所有训练样本张成的子空间中，即

$$\boldsymbol{w}=\sum_{j=1}^{l}\alpha_j\boldsymbol{\Phi}(\boldsymbol{x}_j) \tag{5-78}$$

因此，可以推出

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{m}_i^{\Phi}=\frac{1}{l_i}\sum_{j=1}^{l}\sum_{k=1}^{l_i}\alpha_j k(\boldsymbol{x}_j,\boldsymbol{x}_k^i)=\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{M}_i \tag{5-79}$$

其中，核函数 $k(\boldsymbol{x}_j,\boldsymbol{x}_k^i):=(\boldsymbol{\Phi}(\boldsymbol{x}_j),\boldsymbol{\Phi}(\boldsymbol{x}_k^i))$，$\boldsymbol{M}_i$ 定义为 $(\boldsymbol{M}_i)_j:=\frac{1}{l_i}\sum_{k=1}^{l_i}k(\boldsymbol{x}_j,\boldsymbol{x}_k^i)$。

考查目标函数式(5-74)的分子部分，记

$$\boldsymbol{M}:=(\boldsymbol{M}_1-\boldsymbol{M}_2)(\boldsymbol{M}_1-\boldsymbol{M}_2)^{\mathrm{T}} \tag{5-80}$$

考虑到类间离散度矩阵的定义式(5-75)，可以得到

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{B}}^{\Phi}\boldsymbol{w}=\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\alpha} \tag{5-81}$$

其中，$\boldsymbol{\alpha}$ 是有所有 l 个 α_j 组成的向量。

再考查目标函数式(5-74)的分母部分，利用式(5-78)和式(5-76)、式(5-77)，可以得到

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{W}}^{\Phi}\boldsymbol{w}=\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{N}\boldsymbol{\alpha} \tag{5-82}$$

其中

$$\boldsymbol{N}:=\sum_{j=1,2}\boldsymbol{K}_j(\boldsymbol{I}-\boldsymbol{1}_{l_j})\boldsymbol{K}_j^{\mathrm{T}} \tag{5-83}$$

其中，$\boldsymbol{K}_j$ 是 $l\times l_j$ 矩阵，$(\boldsymbol{K}_j)_{nm}:=k(\boldsymbol{x}_n,\boldsymbol{x}_m^j)$，称作第 j 类的核函数矩阵；$\boldsymbol{I}$ 是单位矩阵，

$\mathbf{1}_{l_j}$ 是所有元素都为$\frac{1}{l_j}$的矩阵。

通过式(5-81)和式(5-82),变换空间里的 Fisher 线性判别的目标函数成为

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{a}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\alpha}}{\boldsymbol{a}^{\mathrm{T}}\boldsymbol{N}\boldsymbol{\alpha}} \tag{5-84}$$

经过与 Fisher 线性判别类似的推导,可以得到,最大化式(5-84)的解是 $\boldsymbol{N}^{-1}\boldsymbol{M}$ 的最大本征值对应的本征向量,并且可以得出,最优解的方向是

$$\boldsymbol{\alpha} \propto \boldsymbol{N}^{-1}(\boldsymbol{M}_1 - \boldsymbol{M}_2) \tag{5-85}$$

如果需要求出变换空间里的投影方向 $\boldsymbol{w}$,则需要利用式(5-78)且需要显式地计算变换 $\boldsymbol{\Phi}(\boldsymbol{x})$,这就失去了核函数方法的优势,在很多情况下变得不可行。然而,求解 Fisher 判别的目的并不是求出投影方向的显式表达,而是实现对原空间任意一个样本到 Fisher 判别的方向上的投影,即只需要计算

$$<\boldsymbol{w},\boldsymbol{\Phi}(\boldsymbol{x})>=\sum_{i=1}^{l}\boldsymbol{\alpha}_i k(\boldsymbol{x}_i,\boldsymbol{x}) \tag{5-86}$$

这又是通过核函数来完成的。

通常,上述问题可能是病态的,因为矩阵 $\boldsymbol{N}$ 可能非正定,这是由于变换后样本维数升高导致的。一种简单的补偿办法是,引入一个新的矩阵

$$\boldsymbol{N}_{\mu} := \boldsymbol{N} + \mu\boldsymbol{I} \tag{5-87}$$

来代替原来的矩阵(μ 是一个常数),使矩阵正定。这样做同时还实现了对 $\|\boldsymbol{\alpha}\|^2$ 的正则化控制,类似于支持向量机中控制间隔的作用。需要说明的是,这只是定性的分析,其理论依据还需要进一步研究。

核 Fisher 判别(KFD)方法在一些实际数据上取得了很好的效果,在 S. Mika 等发表的实验中,KFD 的测试错误率在多个数据上好于 SVM 或与 SVM 相当。之所以 KFD 能够在一些数据上性能胜过 SVM,一个可能的原因是,KFD 中应用了所有训练样本中的分类信息,带有一定的平均的性质,对数据中的噪声不十分敏感;而 SVM 的分类面只取决于两类边界处的样本和错分的样本,当数据中噪声较强或数据分布很不均匀时容易出现偏差。

除了 Fisher 线性判别方法,还有很多经典的线性方法,例如 4.6 节介绍的最小平方误差(MSE)方法,它能够处理线性不可分问题,在一定条件下可以等价于 Fisher 线性判别方法,而且可以最优地逼近贝叶斯分类器。借鉴支持向量机中的思想和 KFD 中成功的例子,可以用核函数方法构造非线性的 MSE 方法,即核最小平方误差方法或称 KMSE(kernel MSE)方法[①]。很多其他的线性方法也可以类似地改造为核函数方法。

5.6.3 中心支持向量机

支持向量机的核心思想是通过控制分类间隔实现对学习机器函数集复杂性的有效控

① Jianhua Xu, Xuegong Zhang, Yanda Li. Kernel MSE algorithm: a unified framework for KFD, LS-SVM and KRR. *Proceedings of IJCNN'01*, 2001, pp. 1486-1491.

许建华,李衍达,张学工. 最小平方误差算法的正则化形式. 自动化学报,2004,30(1): 27-36.

制，从而实现在小样本情况下好的推广能力。分类间隔是通过处在两类边界上的支持向量来定义的，它们代表了分类中最重要的信息。

但是，这种定义也使得支持向量机方法对于样本中的噪声和偏离数据分布的野值非常敏感。例如在图 5-26 的例子中，数据中在两类边界附近有一个被标为第二类的点，它远离本类其他样本点，可以判断为野值。图 5-26(b)中，是否在训练样本中包括这个野值点，所得的 SVM 分类线变化很大，这是因为这个野值点处在边界附近，被算法识别为支持向量。而在图 5-26(a)中用最小平方误差 MSE 方法进行分类，分类线基本不受野值点影响。从这个例子可以看出，如果出现人为标错训练样本或个别训练样本的数据严重偏离分布的情况，支持向量机的结果可能会偏离最优。

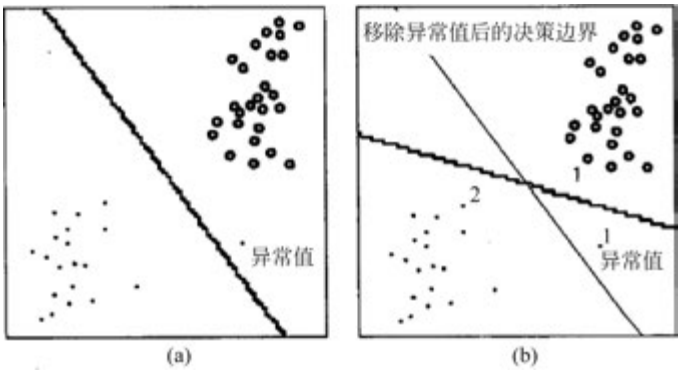


图 5-26 野值对最小平方误差(MSE)方法和支持向量机(SVM)方法的影响。其中，(a)为 MSE 方法，野值点未对 MSE 分类线造成影响；(b)为 SVM 方法，纳入野值点后的 SVM 分类线变动很大

即使是样本中没有明显野值，但在样本数目非常少的情况下，样本采样的偶然性增加，分类器过分依赖个别样本在实际应用中也可能带来很大不确定性。统计学习理论中涉及支持向量机的理论性质，但其中的结论都是在概率意义下成立的，并假定训练样本都是独立同分布，当实际数据可能存在野值时，数据分布方差很大而样本数又过小时，一些理论上的结论就不再成立。

与此相对比，最小平方误差、Fisher 线性判别等线性方法和人工神经网络等非线性方法，运算中包含对全部样本进行平均，这在一定意义上避免了方法对个别样本的过度敏感。根据这一思想，我们提出了中心支持向量机(central support vector machine，CSVM)方法[①]，通过用中心间隔代替支持向量机中的边界间隔，来综合基于均值样本方法的优势和基于边界样本方法的优势，使得在极少样本或含野值样本下能够得到更可靠的分类器。

对于线性判别函数 $y=\boldsymbol{w}\cdot\boldsymbol{x}+b$，设有训练样本集$(\boldsymbol{x}_i,y_i)$，$\boldsymbol{x}_i\in\mathbf{R}^d$，$y_i\in\{+1,-1\}$，$i=1,\cdots,n$。首先考虑所有训练样本都线性可分的情况，即

$$y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)>0,i=1,\cdots,n \tag{5-88}$$

不失一般性，我们引入一个小的常数 $\varepsilon>0$，要求所有样本都满足

① Xuegong Zhang, Using class-center vectors to build support vector machines, *IEEE NNSP*, 1999, 3-11.

$$y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)\geqslant\varepsilon>0,i=1,\cdots,n \tag{5-89}$$

即所有样本都至少离分类超平面有一个小间隔。

计算两类训练样本的中心，分别记为 $\boldsymbol{x}^+$ 和 $\boldsymbol{x}^-$，计算它们到分类超平面的距离：

$$d^+=\frac{|\boldsymbol{w}\cdot\boldsymbol{x}^++b|}{\|\boldsymbol{w}\|}=\frac{y^+(\boldsymbol{w}\cdot\boldsymbol{x}^++b)}{\|\boldsymbol{w}\|}$$
$$d^-=\frac{|\boldsymbol{w}\cdot\boldsymbol{x}^-+b|}{\|\boldsymbol{w}\|}=\frac{y^-(\boldsymbol{w}\cdot\boldsymbol{x}^-+b)}{\|\boldsymbol{w}\|} \tag{5-90}$$

其中，$y^+=1$，$y^-=-1$。若训练样本集中两类样本数分别为 n^+ 和 n^-，则两个类中心到分类面的距离之和可以写为

$$d=d^++d^-=\frac{\sum_{i=1}^{n}l_iy_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)}{\|\boldsymbol{w}\|} \tag{5-91}$$

其中，对第一类样本 $l_i=1/n^+$，对第二类样本 $l_i=1/n^-$。我们把这个距离定义为分类超平面的中心分离间隔，如图 5-27 所示。与支持向量机的情况类似，如果要最大化这个间隔，也存在尺度不确定的问题，为此，引入约束条件

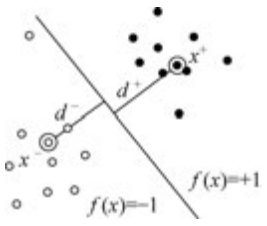


图 5-27 中心分类间隔

$$\sum_{i=1}^{n}l_iy_i(\boldsymbol{w}\cdot\boldsymbol{x}_i)=1 \tag{5-92}$$

在这个约束下，最大化式(5-91)定义的中心分离间隔等价于最小化 $\|\boldsymbol{w}\|$，因此得到下面的优化问题：

$$\min\frac{1}{2}\|\boldsymbol{w}\|^2 \tag{5-93}$$

约束条件是式(5-92)和所有样本都满足式(5-89)。我们把优化这个问题的方法称作中心支持向量机。

容易看出，对于线性可分的情况，如果所有样本分类正确的要求用式(5-88)的条件来保障，则最小化式(5-93)得到的解一定是两类样本中心的垂直平分线方向，即第 2 章提到的最小距离分类器。但当我们通过式(5-89)的条件约束离分类面最近的样本与分类面的间隔最小不能小于 $\varepsilon>0$ 后，实际上是追求最大化中心分离间隔的同时，保证了离分类面最近的样本也有足够的分类间隔。也就是说，中心支持向量机实际上是通过引入 ε 和中心分离间隔，实现了支持向量机与基于均值的分类器的折中。

当训练样本不是线性可分时，条件式(5-89)无法对所有样本同时满足。与支持向量机方法相同，我们引入松弛因子 $\xi_i>0$，使下列不等式对所有样本都满足：

$$y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)+\xi_i\geqslant\varepsilon>0,i=1,\cdots,n \tag{5-94}$$

并把优化的目标函数修正为

$$\min\frac{1}{2}\|\boldsymbol{w}\|^2+C\left(\sum_{i=1}^{n}\xi_i\right) \tag{5-95}$$

其中，C 与支持向量机中一样是控制对错分样本惩罚程度的参数。

采用与支持向量机相同的拉格朗日优化方法，可以得到中心支持向量机的对偶问题：

$$\max\quad Q(\alpha,\beta)=\sum_{i=1}^{n}\varepsilon\alpha_i+\beta-\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i+\beta l_i)(\alpha_j+\beta l_j)y_iy_j(\boldsymbol{x}_i\cdot\boldsymbol{x}_j) \tag{5-96}$$

约束条件是

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

和

$$0 \leqslant \alpha_i \leqslant C, \quad i = 1, \cdots, n$$

可以看到，这个对偶问题形式上仍然与支持向量机的对偶问题相同，可以采用同样的算法进行优化，得到的最优解将满足

$$\boldsymbol{w}^* = \sum_{i=1}^{n} (\alpha_i^* + \beta^* l_i) y_i \boldsymbol{x}_i = \sum_{i=1}^{n} \alpha_i^* y_i \boldsymbol{x}_i + \beta^* (\boldsymbol{x}^+ - \boldsymbol{x}^-) \tag{5-97}$$

从直观上看，式(5-97)的解也由两部分组成，一部分对应着支持向量机的解，另一部分对应着最小距离分类器，这两部分之间的折中由 β^* 决定，它受到 ε 参数设置的影响。利用这一性质，在实际应用中，我们可以不对中心支持向量机设计新的优化算法，而是首先求解标准的支持向量机问题，然后用下面的方法直接显式地规定支持向量机权值与最小距离分类器权值之间的折中：

$$\boldsymbol{w}^{\mathrm{CSVM}} = (1 - \lambda) \boldsymbol{w}^{\mathrm{SVM}} + \lambda (\boldsymbol{x}^+ - \boldsymbol{x}^-) \tag{5-98}$$

折中系数 λ 可以根据训练样本数目和对其中噪声和数据分散程度的认识进行设置，避免了对参数 ε 的设置。

可以看到，中心支持向量机的优化问题和最后解的形式与标准的支持向量机相同，因此也可以采用同样的核函数技巧实现非线性的中心支持向量机。但需要注意的是，如果不通过式(5-97)的对偶问题求解中心支持向量机，而是通过式(5-98)的方式等价地求解，那么 $\boldsymbol{x}^+$、$\boldsymbol{x}^-$ 需要在核函数变换后的空间里计算，即新样本需要与两类的每个训练样本计算核函数内积后再分别求均值。

实验证明，在训练样本数非常少或样本中存在较严重噪声的情况下，中心支持向量机可以比标准的支持向量机得到更稳定的结果，不会因为个别样本的变动而带来解的巨大变化。

5.7 统计学习理论与正则化理论简介

5.7.1 统计学习理论简介

基于数据的机器学习，可以看作从数据中进行判断和提取规律的统计学方法的一种延伸和扩展。早在 200 多年前，人们就开始了对从数据中总结规律的方法的研究，诞生了线性回归方法。统计学是人们研究数据中规律的学科，是机器学习与模式识别的重要基础。传统统计学所关注的是渐近理论，即当样本数目趋向于无穷大时的极限特性，统计学中各种关于估计的理论，包括一致性、无偏性和估计方差的界等，以及前面讨论的关于分类错误率的诸多结论，都属于这种渐近特性。模式识别和机器学习的研究在很大程度上沿袭了统计学的基本思想，在方法性能进行理论分析时，大都采用了渐近分析的思想。这些分析的结论在样本充分多时有效指导了方法的设计，当样本有限时则面临很多问题。

在样本数有限时，机器学习面临的最突出问题之一是过学习。所谓“过学习”，英文是over-fitting，也译为“过适应”，是指学习机器在训练样本上的表现明显好于在未来测试样本上或在实际应用上的表现，比如在训练样本上得到较小的错误率，但在独立的测试样本上错误率却远大于训练错误率。人们把在一定样本上训练的模型或算法在未来新样本上的表现称作学习机器的推广能力(generalization ability)，也有人译为“泛化能力”。学习机器在某个实验中出现了过学习现象，意味着它的推广能力差；反之，一个推广能力差的学习机器，在应用中更容易出现过学习现象。

推广能力是学习机器(包括其模型和算法)的性质，是否出现过学习是学习机器在具体数据上表现的现象。什么样的机器在什么情况下容易出现过学习？这是模式识别与机器学习领域一个重要问题，但很多此类研究带有试错性和启发式的特点。20世纪60年代开始，苏联科学家Vladimir Vapnik和Alexey Chervonenkis等开始系统研究有限样本情况下的机器学习问题①。这些研究最初发表在俄文文章和著作中，其中有些陆续被翻译成德文和英文。由于当时的研究尚不完善，结论趋于保守，数学上比较艰涩，而且在20世纪90年代以前并没有能将理论付诸实现的有效方法，加之八九十年代人工神经网络研究吸引了人们主要的注意力，这些理论研究很长时间没有得到充分重视。直到90年代中期，有限样本情况下的机器学习理论研究逐渐成熟起来，形成了一个较完善的理论体系——统计学习理论(statistical learning theory，SLT)，并诞生了在4.8节和5.5节中介绍的支持向量机方法。与此同时，人工神经网络方法的研究在90年代进展趋缓。在这种情况下，统计学习理论逐步得到重视，并成为20世纪90年代中后期和21世纪开始十年中机器学习研究的主要关注点。

统计学习理论是一套比较完整的理论体系，包含内容很丰富，也涉及比较多的数学知识。4.8.2节对统计学习理论关于支持向量机推广能力的结论进行了简要介绍，限于本书的定位，本节尝试用很短的篇幅扼要介绍该理论体系的主要逻辑框架和基本结论，感兴趣的读者可以学习相关的统计学习理论专著②。

5.7.2 关于VC维与推广性界的核心结论

统计学习理论把机器学习问题表述为利用数据进行函数估计的问题。如图5-28所示，已知变量y与输入x之间存在一定的未知依赖关系，即存在一个未知的联合概率密度函数$F(x,y)$，机器学习就是根据l个独立同分布观测样本

$$(x_1,y_1),(x_2,y_2),\cdots,(x_l,y_l) \tag{5-99}$$

① V. N. Vapnik & A. Ja. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities. *Doklady Akademii Nauk USSR*, 1968, 181(4).

V. N. Vapnik & A. Ja. Chervonenkis. *Theory of Pattern Recognition (in Russian)*. Nauka, Mascow, 1974 (German translation 1979, Akademia, Berlin).

V. N. Vapnik. *Estimation of Dependencies Based on Empirical Data (in Russian)*. Nauka, Moscow, 1979 (English translation, 1982, Springer, New York).

② 如V. Vapnik著，张学工译，《统计学习理论的本质》，清华大学出版社，2000；V. Vapnik著，许建华、张学工译，《统计学习理论》，清华大学出版社，2003.

在一个函数集$\{f(x,\alpha),\alpha\in\Lambda\}$中求一个最优的函数$f(x,\alpha_0)$，使它给出的预测的期望风险

$$R(\alpha)=\int L(y,f(x,\alpha))\mathrm{d}F(x,y) \tag{5-100}$$

最小。其中，$\{f(x,\alpha),\alpha\in\Lambda\}$是候选函数集，$\alpha\in\Lambda$为函数的广义参数，故$\{f(x,\alpha)\}$可以表示任何函数集。$L(y,f(x,\alpha))$为由于用$f(x,\alpha)$对$y$进行预测而造成的损失，称作损失函数。$R(\alpha)$是函数$f(x,\alpha)$的函数，故称作期望风险泛函(expected risk functional)。不同类型的学习问题有不同形式的损失函数。模式识别问题、回归问题和概率密度函数估计问题，都可以看作这个机器学习框架下的特例。

图 5-28 统计学习理论表述的机器学习问题基本框架

实际问题中，我们并不知道联合概率密度函数$F(x,y)$，因此只能用训练样本上得到的经验风险(empirical risk)作为对期望风险的估计：

$$R_{\mathrm{emp}}(\alpha)=\frac{1}{l}\sum_{i=1}^{l}L(y_i,f(x_i,\alpha)) \tag{5-101}$$

统计学习理论把这种做法称为经验风险最小化(empirical risk minimization，ERM)原则下的机器学习。可以看到，我们前面介绍的除支持向量机外的各种线性和非线性模式识别方法，都可以看作ERM原则机器学习的具体实践。

在有限样本下机器学习推广性的问题，就是学习机器的函数集中用经验风险最小化得到的最优函数，是否为对期望风险最小化函数的逼近。

统计学习理论研究发现，对这个问题的回答并不总是肯定的，而取决于学习机器所实现的函数集的性质。统计学习理论提出了度量函数集性质的一系列指标，其中最重要的是函数集的VC维(Vapnik-Chervonenkis dimension)。

统计学习理论把实现两类分类的判别函数称为指示函数。指示函数集的VC维可以直观地定义为：对这个指示函数集，如果存在至少一组h个样本能被函数集中的函数按所有可能的2^h种可能的方案分开，则称函数集把h个样本“打散”，函数集的VC维就是能被函数集打散的最大样本数。VC维反映了函数集的容量或能力，函数集的VC维越高，则能力越强。统计学习理论的一个主要结论是：对于有限的训练样本来说，学习机器所实现的函数集的能力不能过强，否则有可能出现过学习。从直观上认识，学习机器如果在有限样本下从一个能力极强的函数集里得到一个使经费风险很小的函数，我们并无法确认是因为函数集容量大才轻易得到了这样一个函数，还是真正找到了能反映数据内在规律的函数。

统计学习理论的一个核心结论是：对于有l个训练样本的两类分类问题，学习机器的期望风险至少以概率$1-\eta$满足如下的上界：

$$R(\alpha)\leqslant R_{\mathrm{emp}}(\alpha)+\sqrt{\frac{h\left(\ln\frac{2l}{h}+1\right)-\ln\frac{\eta}{4}}{l}} \tag{5-102}$$

或者定性简写为

$$R(\alpha) \leqslant R_{\text{emp}}(\alpha) + \Phi\left(\frac{h}{l}\right) \tag{5-103}$$

其中，$\Phi(l/h)$是样本数 l 的单调减函数、VC 维 h 的单调增函数。也就是说，在有限样本下，期望风险可能会大于经验风险，超出部分的最大上界是 $\Phi(h/l)$。在统计学习理论的文献中，超出部分的上界被称作置信范围(confidence interval)①。

要在有限样本下保证学习机器有好的推广能力，不能只最小化经验风险，还需要同时最小化置信范围。图 5-29 示意了期望风险与经验风险和置信范围的关系。统计学习理论提出了所谓“结构风险最小化”(structural risk minimization，SRM)原则，即要把学习机器的函数集划分为多个嵌套的子集，在最优的子集中选择经验风险最小的函数。

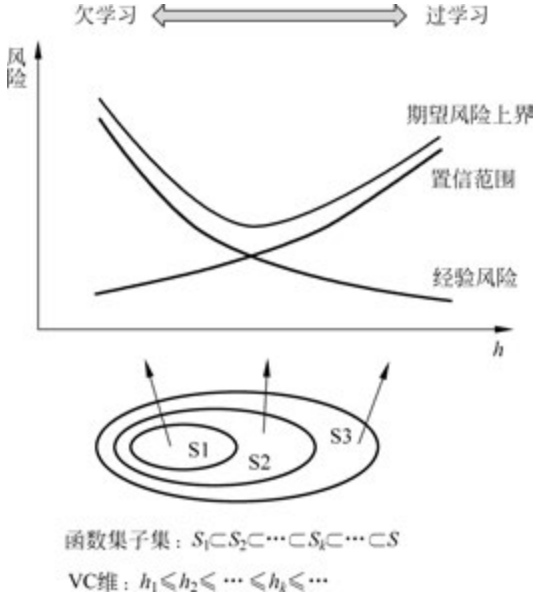


图 5-29 结构风险最小化(SRM)原则示意图

统计学习理论的研究证明，支持向量机中最大化分类间隔，就是最小化函数集 VC 维的上界。在高维空间中，尤其是经过核函数变换后的高维空间中，空间维数很大甚至是无穷大，但通过控制分类间隔可以有效控制函数子集的 VC 维，从而保证在函数子集中求得经验风险最小的解具有好的推广能力。同时也证明，支持向量机在有限样本训练后得到的支持向量数目在训练样本中所占的比例，可以体现学习机器的推广能力，比例越小，则期望的测试错误率上界越小。

统计学习理论为研究小样本机器学习问题提供了一套比较完整的理论体系，其主要结论对在有限样本下如何避免过学习提供了重要指导或参考原则。但统计学习理论本身也存在一定的局限：一方面，VC 维对大部分机器学习模型都难以估计，使得理论对很多模型的设计和选择难以提供精准指导；另一方面，统计学习理论采取的是最坏情况估计的策略，对

① 这里的 confidence interval 与传统统计学中置信区间定义不同，所以我们译作“置信范围”，也有人把它称作 VC Confidence，可译作“VC 置信”。

于很多实际情况来说偏保守，具体反映在理论中的很多估计的界都比较松。如何拓展统计学习理论，发展能适用于更多场景的机器学习理论，是值得深入探讨的一个未来研究方向。

5.7.3 不适定问题和正则化方法简介

“不适定问题”(ill-posed problems)是指在求解一个算子方程时，观测值的微小扰动或噪声可能会导致求解结果很大变化的情况。模式识别与机器学习中的过学习问题，可以看作不适定问题的特例。

不适定问题是早在 20 世纪初由 Hadamard 发现的：在很多情况下，求解算子方程

$$Af=F,\quad f\in\mathcal{F} \tag{5-104}$$

的问题是不适定的。即，即使方程存在唯一解，方程右边的微小扰动 $\|F-F_\delta\|<\delta$ 也会带来解的很大变化。这种情况下，在无法得到准确的观测 F 的情况下，对带有噪声的观测 F_δ 用常见的最小化下面的目标泛函：

$$R(f)=\|Af-F_\delta\|^2 \tag{5-105}$$

的方法无法得到对解 f 的好的估计，即使扰动 δ 趋向于零也如此。

20 世纪 60 年代，以 Tikhonov、Ivanov、Phillips 等为代表的学者提出求解不适定问题的正则化(regularization)方法[①]。他们发现，不适定问题不能通过最小化式(5-104)定义的目标泛函来求解，而应该最小化下面的正则化泛函(regularized functional)：

$$R^*(f)=\|Af-F_\delta\|^2+\lambda(\delta)\Omega(f) \tag{5-106}$$

其中，$\Omega(f)$是度量解 f 的某种性质的泛函，$\lambda(\delta)$是与观测噪声水平有关的需适当选取的常数。对这个正则化目标进行最小化，就能保证得到的解在噪声趋向于零时收敛到理想的解。

对照统计学习理论中的主要结论，我们可以看到，式(5-105)的目标类似于经验风险最小化，而式(5-106)的目标类似于结构风险最小化。在结构风险最小化要优化的目标 $R_{\text{emp}}(\alpha)+\Phi(h/l)$里，置信范围 $\Phi(h/l)$可以看作对解函数的某种正则化目标。

有些学者把支持向量机用正则化的框架表述如下：设待求函数为 $f(x)=h(x)+b$，其中 h 是由核函数 K 确定的可再生希尔伯特空间$\mathcal{H}_K$ 中的函数，待求的分类器是对 $f(x)$取符号，即 $\phi(x)=\text{sgn}(f(x))$，支持向量机就是在 l 个训练样本对(x_i,y_i)，$i=1,\cdots,l$ 下最小化下述目标函数：

$$\min_f \frac{1}{l}\sum_{i=1}^{l}(1-y_i f(x_i))_+ +\lambda\|h\|^2_{\mathcal{H}_K} \tag{5-107}$$

其中，目标函数的第一项对应着支持向量机原问题中用松弛因子表示的分类错误惩罚，第二项对应着支持向量机原问题中最大化间隔的项，两项之间折中的系数变成了这里的正则化系数 λ。

采用这个框架，Lee. Lin 和 Wahba 发展了一种直接解决多类分类问题的多类支持向量

---

① A. N. Tikhonov. On the stability of inverse problem. *Dokl Acad. Nauk USSR*, 1963, 39: 5(in Russian).
V. V. Ivanov. On linear problems which are not well-posed. *Soviet Math. Dokl.*, 1962, 3(4): 981-983.
D. Z. Phillips. A technique for numerical solution of certain integral equation of the first kind. *J. Assoc. Comput. Mach.*, 1962, 9: 84-96.

机方法(详见4.5.5节)。

在式(5-106)的正则化方法框架下,选取不同的正则化项$\Omega(f)$,就产生了不同的正则化方法,它们在模型和算法性质上各有不同的特点。支持向量机中最大化分类间隔,等价于是用参数向量模的平方作为正则化项,与Tikhonov正则化的原理是一致的。根据正则化项对参数空间采用的度量不同,在线性回归基础上出现了一系列正则化方法,它们大致可以分L_0正则化、L_1正则化、L_2正则化、L_q正则化和弹性网(elastic net)方法,它们分别采用L_0、L_1、L_2、L_q范数和混合的范数对解函数进行正则化约束。我们用$\boldsymbol{\beta}$来表示回归函数中的参数向量,$V(y_j,\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}_j)$表示回归误差的某种度量(如绝对值误差或平方误差),以下介绍各种正则化方法代表性的目标函数。

L_0正则化:

$$\min_{\beta}\frac{1}{l}\sum_{i=1}^{l}V(y_j,\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}_j)+\lambda\parallel\boldsymbol{\beta}\parallel_0 \tag{5-108}$$

L_1正则化(Lasso或基追踪算法):

$$\min_{\beta}\frac{1}{l}\sum_{i=1}^{l}(y_j-\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}_j)^2+\lambda\parallel\boldsymbol{\beta}\parallel_1 \tag{5-109}$$

L_2正则化(Tikhonov正则化):

$$\min_{\beta}\frac{1}{l}\sum_{i=1}^{l}V(y_j,\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}_j)+\lambda\parallel\boldsymbol{\beta}\parallel^2 \tag{5-110}$$

L_q正则化:

$$\min_{\beta}\frac{1}{l}\sum_{i=1}^{l}V(y_j,\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}_j)+\lambda\sum_{j}\mid\beta_j^q\mid^{\frac{1}{q}} \tag{5-111}$$

弹性网(Elastic-Net,亦称混合正则化):

$$\min_{\beta}\frac{1}{l}\sum_{i=1}^{l}(y_j-\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}_j)^2+\lambda(\alpha\parallel\boldsymbol{\beta}\parallel_1+(1-\alpha)\parallel\boldsymbol{\beta}\parallel^2) \tag{5-112}$$

这些不同的正则化方法都是用不同的范数来对解函数进行约束,会带来不同的效果。比如,L_0范数就是对参数向量中非零参数个数的计数,把它放到目标函数中进行最小化,就是在要求经验风险最小化的同时,希望函数中非零参数的个数尽可能少,实现在减小训练误差的同时实现特征选择的功能,也就是常说的学习对样本特征的稀疏表示,这也是“压缩感知”的基本思想。但L_0范数的优化计算很难,L_1范数即参数向量各元素的绝对值之和也可以用来作为对非零参数个数的一种惩罚,所以比较广泛地被采用。

L_2范数由于采用了平方和,在计算上有很大的方便性,也是最早提出正则化方法时采用的范数。L_2范数能够有效地防止参数变得过大,可以较有效地避免过拟合,但平方惩罚对于强制小的参数变成0的作用不大。采用L_2范数的线性回归方法也称作岭回归(Ridge regression)。支持向量机中的最大化分类间隔就等价于采用L_2范数作为正则化项,统计学习理论为采用这种正则化对提高推广能力的作用提供了理论依据。与L_2正则化回归方法不同,支持向量机中对错误的度量不是采用平方误差函数,而是对分类错误采用了线性的惩罚。

弹性网方法则是采用了L_1范数与L_2范数相结合的方式,它既发挥L_2范数的作用防止参数值过大带来的过学习风险,也利用L_1范数有效减少非零参数个数,两个目标通过人

为确定的常数来进行权衡。

不同的正则化项对学习性能具有不同的理论性质，同时也在优化算法上有不同的优势或劣势，在统计学和机器学习领域有很多研究，限于本教材范围，这里不展开深入讨论。

在本节的所有讨论中，我们都忽略了正则化系数 λ 的选择问题，除了在开始提到了它应该与噪声水平有关。λ 的选择是机器学习模型选择中的一个重要问题，它应该反映我们对数据内存在的规律和噪声的认识。但遗憾的是，对于复杂的机器学习问题，我们很难事先对数据有足够的了解，所以很大程度上需要凭经验进行选择，或者通过一定的方法进行试算后选择。从理论上，正则化系数的选择属于模型选择的一个问题，人们在贝叶斯的框架下开展了很多研究，限于本书范围也无法展开讨论。与支持向量机中的核函数及其参数选择、折中参数 C 的选择、神经网络结构的选择等类似，正则化方法中正则化项和正则化系数的选择也属于机器学习中所谓“超参数”，需要在理论、经验和试算共同指导下进行。

5.8 讨论

本章讨论了几种非线性分类器的概念和设计方法，包括经典的分段线性分类器、二次判别函数和新近发展的多层感知器神经网络方法及支持向量机方法。与线性方法相比，非线性方法的种类更多，这里只是讨论了其中有代表性的几种。第 6 章将要介绍的一些方法，包括近邻法、分类树与随机森林方法等，大多也是实现某种非线性分类，但是由于它们不是从设计判别函数的角度引出的，我们把它们作为单独的一章来介绍。

由于客观世界的复杂性和观测数据的局限性，人们面临的很多或者大多数问题严格来说都是非线性的，从这个意义上说，非线性方法具有更广的适用性。但同时，多数实际问题中的样本又往往是有限的和不准确的，而且我们一般并不知道问题内在的规律符合什么样的非线性模型，这种情况下，如果盲目地选用复杂的非线性方法不一定取得好的效果，相反，线性方法却有可能成为对未知非线性规律的更好的逼近。

本章花大量篇幅介绍的多层感知器方法和支持向量机方法是两种适用性很广的非线性方法，它们并没有直接对非线性模型进行假设，而是通过多个隐节点作用的组合或样本与多个支持向量的核函数内积加权和来实现各种非线性分类面。在多层感知器中，分类面的复杂度取决于网络结构的设计和训练样本的分布；在支持向量机中，分类面的复杂度取决于核函数的选取及训练样本的分布。神经网络结构的设计和支持向量机核函数的选择问题都可看作模型选择问题。对于有限数目的训练样本来说，在决定神经网络结构和选择支持向量机核函数及其参数时，应该充分考虑到分类器的推广能力问题，使模型和参数选择与样本和问题的复杂度相适应，必要时可以通过交叉验证等方法对可能的选择进行比较。如果能在神经网络结构设计或支持向量机核函数选择等方面充分结合应用领域的专门知识，则模式识别效果将会比单纯采用“黑箱”式的算法更好。

在训练样本有限时，如何保障较好的推广能力是机器学习模型与方法设计的一个重要问题。统计学习理论为研究有限样本下学习机器的推广能力提供了理论框架，正则化方法则通过对学习机器的性质进行约束来提供不适定问题的解决方案。