# 第1章 绪 论

## 1.1 计算智能及其相关的基本理论

### 1.1.1 计算智能的发展历程

目前，人工智能（Artificial intelligence，简称 AI）尚无统一的定义。人类的自然智能无处不在，人类的诸多活动，例如驾驶、竞技、解算题、编程序、骑车等，都涉及人类的“智能”。若机器能执行类似人类的这种任务，则认为机器已具有某种性质的“人工智能”[1]。

1956 年，在美国达特茅斯（Dartmouth）大学举办了一场人工智能研讨会，会上首次使用了人工智能这一术语，标志着人工智能科学的诞生，具有重要的历史意义。1969 年，第一届国际人工智能联合会议（International joint conference on AI）召开。1970 年，《人工智能》国际杂志（International journal of AI）创刊。专门的学术会议和刊物对人工智能的交流、促进人工智能的研究和发展起到积极作用。

判断机器是否具有智能的方法为阿兰·图灵提出的“图灵测试（Turing Test）”。图灵建议与其提出一个长长的而可能有争议的清单来列举智能所需要的能力，不如采用基于人类这种无可置疑的智能实体的辨别能力的测试。如果人类询问者，即裁判（Judge）在提出一些书面问题后，无法判断答案是否由人或是机器写出，那么该机器被认为通过了测试[2]。作者有幸于 2008 年参加在英国雷丁大学举办的图灵测试活动并担任裁判，并于会后提供了 2 条裁判经验[3]。2012 年，在英国布莱切利园（Bletchley park）举行纪念图灵 100 周年诞辰暨图灵测试活动，仍未有机器通过该测试。

关于生物智能（Biological intelligence，简称 BI）、人工智能（AI）和计算智能（CI）之间的关系，很多学者提出了各自的看法。1994 年，James C.Bezdek 在 IEEE WCCI 会议上阐述他的观点，即智能分为三个层次：①生物智能，是由人脑的物理化学过程反映出来的，人脑是有机物，它是智能的基础。②人工智能是非生物的、人造的，常用符号来表示，AI 的来源是人类知识的精华。③计算智能是通过数学方法和计算机实现的，CI 的来源是数值计算的传感器。这三个层次从复杂程度来看，BI>AI>CI，CI 与 AI 的差距要比 AI 与 BI 的差距小得多。从所属关系来看，CI 是 AI 的一个子集，而 AI 不是 BI 的子集，但 BI 通常用来指导 AI 模型，同样也指导 CI 模型。AI 是 CI 到 BI 的过渡，因为 AI 中除计算方法外，还包括符号表示及数值信息处理。

Lotfi Zadeh 认为，传统的硬件计算是强调人工智能的计算模式，而计算智能的基础是软件计算，即模糊逻辑、神经网络和进化计算。他认为 CI 和 AI 的本质区别在于使用的推理类型不同，AI 使用的是易脆逻辑，而 CI 使用的是模糊逻辑和规则。

David Fogel 在 1995 年发表的评论中指出智能就是对环境的适应能力，他认为 CI 高于 AI，CI

包含 AI。

Eberhart、Dobbins 和 Simpson 关于 CI 的理解是，将智能系统置于一个环境中，智能行为的标准是改变或作用于环境的能力。而 CI 只是智能系统的一个内部节点，适应性只是 CI 的衡量指标。CI 是一种方法论，是通过计算实现适应和处理新形势的能力，具有推理的属性，能得到预测或决定的结果。

另外，也有些学者认为 CI 和 AI 仅有部分重合。他们认为 AI 是符号主义，知识、规则和推理，相当于人的左脑；CI 属于连接主义，基于数据、学习和记忆，相当于人的右脑。

此外，由于 AI 应用领域的不断扩大，导致人工神经网络（Artificial neural networks，ANN）、进化计算（Evolutionary computation）、模糊逻辑（Fuzzy logic）等模拟生物智能的研究重新获得繁荣，这就促使了计算智能的兴起。计算智能又被称为“软计算”[4]。

计算智能的定义为借助计算机模拟人的智能机制、生命的演化过程、人或其他生物的智能行为而进行信息获取、处理、利用的理论和方法[4, 5]。1994 年，首届 IEEE 世界计算智能大会（World congress on computational intelligence，WCCI）召开，大会覆盖了模糊系统、神经网络和进化计算三个领域，标志着融合计算智能学科的建立。目前计算智能领域主要存在着三种观点，即模拟生命生成过程和智能进化过程的演化性观点、模拟智能产生与作用赖以存在的结构性观点以及模拟智能表现与功能行为的逻辑观点。与这些观点相对应，计算智能形成了以进化计算、人工神经网络与模糊逻辑为代表的三个典型分支[4, 5]，虽说计算智能是一个新的研究方向，但是其三个典型分支却有了几十年的发展。同时，一些近年来发展的新兴方法如群智能、人工免疫算法、DNA 计算等也被纳入到计算智能中来，体现了理论发展的与时俱进。

计算智能的方法很多，本书首先简要概述其三个分支中的进化计算和人工神经网络，而后针对近年来新发展的人工蜂群算法、发展较快的微粒群算法、应用广泛的多层前馈神经网络（MLPNN）和径向基神经网络（RBFNN），介绍其基本理论以及应用研究现状。

### 1.1.2 进化计算概述

传统的进化计算主要由三个分支组成，即遗传算法（Genetic algorithms，GA）、进化规划（Evolutionary programming，EP）和进化策略（Evolutionary strategy，ES）。进化计算的方法统一称为进化算法（Evolutionary algorithms，EA），传统算法中以遗传算法影响最大，应用最广。有关传统进化计算的研究主要集中在：编码方式的研究、遗传算子的研究、控制参数的优化以及选择机制等方面[5]。

进化算法的优越性为其在搜索的过程中不易陷入局部极值，即使适应度函数不连续或者有噪声的情况，其也能以很大的概率找到全局最优解。虽然进化算法的理论研究仍不完善，但其大量的成功应用使其成为优化算法研究中令人瞩目的领域之一。

得益于 20 世纪 60 年代创立的仿生学，研究者们从模仿生物的角度出发，创造了很多含有新功能的实用优化工具。其中蚁群、蜂群、鸟群等的自组织行为引起了广泛关注，许多学者对这种行为进行数学建模并对其进行仿真，群智能（Swarm intelligence，SI）由此产生。社会性动物的奇妙之处在于个体行为简单、能力有限，然而当它们一起协同工作时，却表现出非凡的行为特征，而这种复杂的行为特征并不是简单个体能力的累加。表 1.1 列举了迄今为止几乎所有的智能优化算法。

表 1.1　智能优化算法列表

| 序号 | 算法名称 | 提出者 | 提出时间 | 主要应用领域 |
|---|---|---|---|---|
| 1 | 模拟退火算法[6] Simulate annealing | Metroplis | 1953 | 组合优化，函数优化 |
| 2 | 进化策略[7] Evolutionary strategy | Rechenberg | 1965 | 连续优化 |
| 3 | 进化规划[8] Evolutionary programming | Fogel，et al. | 1966 | 连续优化 |
| 4 | 遗传算法[9] Genetic algorithms | Holland | 1975 | 组合优化，连续优化 |
| 5 | 禁忌搜索[10] Tabu search | Glover | 1986 | 组合优化，函数优化 |
| 6 | 蚁群算法[11, 12] Ant colony optimization | Dorigo，et al. | 1991 | 组合优化 |
| 7 | 文化算法[13] Cultural algorithms | Reynolds | 1994 | 连续优化 |
| 8 | 差分进化算法[14-16] Differential evolution | Storn and Price | 1995 | 连续优化 |
| 9 | 微粒群算法[17] Particle swarm optimization | Eberhart and Kennedy | 1995 | 连续优化，组合优化 |
| 10 | 和声搜索[18] Harmony search | Geem | 2001 | 连续优化 |
| 11 | 人工鱼群算法[19] Artificial fish swarm | 李晓磊，等 | 2002 | 连续优化 |
| 12 | 细菌趋药性算法[20] bacterial chemotaxis | Müller，et al. | 2002 | 连续优化 |
| 13 | 自由搜索[21] Free search | Penev and Littlefair | 2005 | 连续优化 |
| 14 | 人工蜂群算法[22] Artificial bee colony | Karaboga | 2005 | 连续优化 |
| 15 | 生物地理优化算法[23] Biogeography-based optimization algorithm | Dan Simon | 2008 | 连续优化 |

在群智能优化算法出现之前，研究最深入和应用最广泛的是进化类算法。即便是现在，进化类算法仍然是优化计算领域一个重要且快速发展的分支。群智能优化算法和进化类算法都具有内在的并行性是它们的共同之处，此外它们也有一些差别。首先，进化算法模拟的是生物进化系统，其基本单位是基因，在代与代之间相传。而群智能算法模拟的是社会性动物的群体行为，其基本

单位是敏因，它们在群体中的个体之间共享。其次，进化算法强调“适者生存、优胜劣汰”的自然法则，适应度是其核心，同时也是群体进化内在的驱动力，优质的个体通过产生更多的后代来传播自己的基因。而群智能优化算法强调的是“群体协作”思想，通过该思想，劣质的个体向优质的个体学习以此来完善自己，个体则通过吸引同伴来传播自己的敏因。此外，群智能中个体之间只需一些简单的信息交互便能实现全局搜索目的，而进化算法则需较为复杂的遗传操作；在整个搜索过程中，个体是一直存在的，而进化算法需有新旧个体的更迭。这些独特的优点，使得群智能优化算法相比进化算法在近几年得到了更加广泛的关注。

值得一提的是，1997 年，Wolpert 和 Lilacready 发表了论文[24]，提出优化领域著名的“无免费午餐理论”（No free lunch theory）。该理论指出，对于有限搜索空间下的优化问题组成的整体来说，所有优化算法的平均表现是一样的，本质上并无优劣之分。打个比方说，如果算法 A 在某些问题上的求解性能优于 B，其必定在另外一些问题的求解性能上不如 B。这里所指的性能为综合性能，即包括优化精度、时间复杂度和空间复杂度等方面。“无免费午餐理论”间接指出了研究优化算法应该遵循的原则：寻找某个万能的优化方法是浪费时间的，对于研究者只需找出在某个特定问题上哪些算法具有更好的性能便可。换句话说，即对于某一个具体的算法来说，找出它适合解决哪类问题具有更大意义。

### 1.1.3 人工神经网络概述

人工神经网络是对人脑的模拟和抽象，它是通过数学方法来实现基本特征刻画的，其本质是一种模仿人脑结构及其功能的非线性信息处理系统。

人工神经元是对生物神经元的简化和模拟，它是人工神经网络的基本处理单元。大量的神经元相互连接即组成人工神经网络。一个典型的神经元是由输入、网络权值和阈值、激活单元、激活函数和输出组成。

人工神经网络的类型多种多样，从不同角度考虑，可以分成不同的类型，具体见表 1.2。

**表 1.2 人工神经网络分类**

| 分类角度 | 典型的神经网络 |
| --- | --- |
| 功能学习特性 | 线性神经网络、BP 网络、径向基神经网络、自组织网络 |
| 网络拓扑结构 | 前向网络、反馈网络和互联网络 |
| 网络输入类别 | 静态网络和动态网络（如自适应滤波线性神经网络和 Hopfield 网络） |

人工神经网络的应用范围很广，之所以被广泛应用是由于其具有以下优点：①具有充分逼近任意复杂非线性函数的能力；②具有学习与适应不确定系统的动态特性；③神经元节点可以存储信息，可用于多输入多输出系统；④具有进行快速大量运算的能力。

文献[25]指出目前包括进化计算和人工神经网络等智能算法的研究趋势：一是对经典智能算法的改进和广泛应用，以及对理论的深入、广泛研究；二是现代智能优化算法的发展，即开发新的智能工业，拓宽应用领域，寻求理论基础；三是经典智能算法与现代智能算法的结合建立混合智能算法。

### 1.1.4　人工蜂群算法

近年来，在优化算法领域兴起了一种称为蜂群算法的随机型优化搜索算法[26]，该类算法主要受蜜蜂采蜜启发，蜂群算法有很多种[26]，本书感兴趣的是人工蜂群算法（ABC，artificial bee colony algorithm），它是由 Dervis Kababoga 于 2005 年提出的一种随机型优化搜索算法[22]。

人工蜂群算法自 2005 年被 Dervis Kababoga 提出以来，应用领域得到了很大的扩展，其 2007 年发表于 Journal of Global Optimization 上的文章[27]已被引用近百次。数十篇相关论文发表于期刊和会议上，具体列于人工蜂群算法的主页上[28]，可以看到目前人工蜂群算法的应用情况为滤波器设计[29]，神经网络训练[30]，混凝土坝参数反演[31]，声发射信号分类[32]，低空飞行物的目标识别[33]。人工蜂群算法自身的改进主要有文献[34]中的将其与量子进化算法融合以及文献[35]中的与混沌理论融合。同时可以看到，目前用于求解离散问题的二进制人工蜂群算法及其应用未见报道。

人工蜂群包含 3 个组成部分：采蜜蜂，跟随蜂，侦察蜂。蜂群中的一半为采蜜蜂，另一半为跟随蜂。每一处蜜源仅对应一个采蜜蜂，也就是说蜜源数和采蜜蜂的数目相等。人工蜂群的搜索过程可以概括如下：采蜜蜂根据它们记忆中的蜜源位置在其邻域内确定另一个蜜源；采蜜蜂在蜂巢内将它们的信息通过舞蹈共享给跟随蜂，跟随蜂选择一个蜜源；跟随蜂根据所选择的蜜源在其邻域内搜索另一个蜜源；放弃所采蜜源的采蜜蜂将成为侦察蜂并搜索一个新的随机蜜源。蜂群采蜜的工作过程如图 1.1 所示。

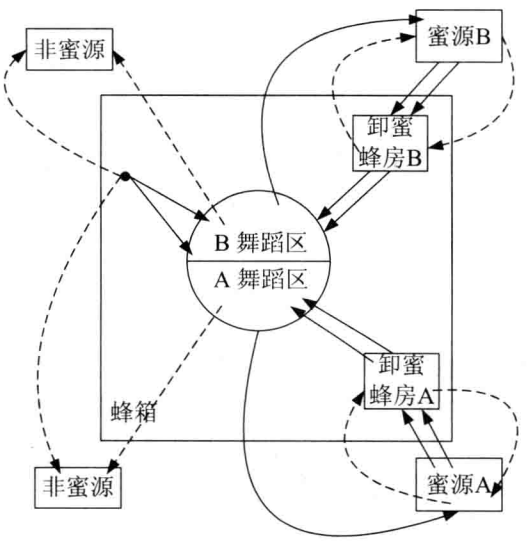



图 1.1　蜂群采蜜工作示意

在 ABC 算法中，每一处蜜源的位置即代表优化问题的一个可能的解，蜜源的花蜜数量代表解的优劣（适应度）。首先，ABC 随机产生含 $n_e$ 个初始解的蜂群，$n_e$ 为采蜜蜂的数目也等于蜜源数目。每个解 $\mathbf{x}_i$ 是一个 $d$ 维向量，其中 $i \in \{1,2,\cdots,n_e\}$，$d$ 为所要优化参数的个数。以这些初始解为基础，采蜜蜂、跟随蜂和侦察蜂开始进行循环搜索。采蜜蜂根据它记忆中的局部信息产生一个变化的位置并检查新位置的花蜜量即适应度值，如果新位置优于原位置，则该蜜蜂记住新位置并放弃原位置。所有的采蜜蜂完成搜索过程后，它们将所得到的蜜源信息通过舞蹈区与跟随蜂共享。跟随蜂根据从采蜜蜂处得到的蜜源信息，按照与花蜜量相关的概率选择一个蜜源位置，并且像采

蜜蜂那样对记忆中的位置做一定的改变，且检查新候选位置的花蜜量，若新位置优于记忆中的位置，则采用新位置替换原先记忆中的蜜源位置；否则保留原记忆中的位置。换句话说，贪婪选择机制被用于选择原位置和候选的新位置。

一个跟随蜂选择某个蜜源的概率为

$$p_i = \frac{F(\theta_i)}{\sum_{i=1}^{s} F(\theta_i)} \tag{1.1}$$

其中 $F(\theta_i)$ 是第 $i$ 个解的适应度。

采蜜蜂和跟随蜂按照下式进行邻域搜索

$$v_{ij} = x_{ij} + \varphi(x_{ij} - x_{kj}) \tag{1.2}$$

其中 $k \in \{1,2,\cdots,n_e\}$，$j \in \{1,2,\cdots,d\}$，$k$、$j$ 均为随机选取，且 $k$ 的取值需要满足 $k \neq i$。$\varphi$ 为参数，有文献称其为搜索因子[5]，且其为[-1，1]范围内的随机数，此参数控制了 $x_{ij}$ 邻域内新解的产生。同时从式（1.2）可以看出，随着优化搜索过程的推进，$x_{ij}$ 与 $x_{kj}$ 之间的差距将越来越小，步长会自适应地缩小，使得该算法具有自适应收敛的特性。

假如 $x_i$ 在经过用户设定的外循环次数极限之后仍然不能够被改进，则该位置将被抛弃。该位置的采蜜蜂蜕变为侦察蜂，假如被抛弃的解为 $x_i$，那么将由下式产生新解代替原 $x_i$：

$$x_i^j = x_{\min}^j + \text{rand}(0,1)(x_{\max}^j - x_{\min}^j) \tag{1.3}$$

以上就是基本人工蜂群算法的计算流程。此外，对于任何一个优化算法，其收敛性都至关重要。由于人工蜂群算法是一个较新的优化方法，其收敛性证明的文献很少，据作者所知，仅文献[33]给出了人工蜂群算法的收敛性证明，在此不再赘述。

### 1.1.5 微粒群算法

微粒群（PSO）[17]是由 Eberhart 和 Kennedy 提出的一种进化计算方法。其思想来源于人工生命和演化计算理论，是基于对鸟群飞行行为的研究而提出的群智能算法。PSO 通过群体中粒子间的合作与竞争产生群体智能，从而指导优化搜索。相比进化类算法，PSO 保留了基于种群的全局搜索策略，采用简单的速度和位移模型，既避免了复杂的遗传操作，又具有较强的全局收敛能力和鲁棒性，且不需要借助问题的特征信息。

微粒群算法中，微粒的速度更新公式如下：

$$v_{id}^{t+1} = \omega v_{id}^t + c_1 r_1 (p_{id}^t - x_{id}^t) + c_2 r_2 (p_{gd}^t - x_{id}^t) \tag{1.4}$$

$i = 1,2,\cdots,m$；$d = 1,2,\cdots,D$

位置更新公式如下：

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \tag{1.5}$$

式中：$t$——当前群的迭代数；

$v_{id}$——微粒 $i$ 的第 $d$ 维的速度分量；

$x_{id}$——微粒 $i$ 的第 $d$ 维的位置分量；

$\omega$——惯性权重；

$c_1$、$c_2$——加速因子；

$p_{id}$——微粒 $i$ 自身最好位置的第 $d$ 维分量；

$p_{gd}$——微粒群中最好位置的第 $d$ 维分量；

$D$——粒子维数；

$m$——种群微粒数量；

$r_1$、$r_2$——在区间[0，1]均匀分布的随机数。

微粒群微粒的速度更新由三部分组成，第一部分是微粒的先前速度，说明微粒目前的状态，在搜索初期较大的 $\omega$ 值有利于跳出局部极小点，搜索后期较小的 $\omega$ 值有利于算法收敛；第二部分是个体的认知部分，这部分使微粒有较强的全局搜索能力，避免陷入局部极小；第三部分是社会共享信息，这部分将使微粒从其他优秀微粒中汲取经验，加强搜索能力。

PSO 的基本算法步骤如下：

（1）初始化微粒群，即随机设定各微粒的初始位置和初始速度；

（2）计算每个微粒的适应度值；

（3）对每个微粒比较它的适应度值和它经历过的最好位置的适应度值，若更好，则更新当前最优位置 $p_{id}$；

（4）对每个微粒，比较它的适应度值和整个群体所经历的最好位置的适应度值，若更好，则更新全局最优位置 $p_{gd}$；

（5）根据式（1.4）和式（1.5）更新微粒速度和位置；

（6）如果满足设定结束条件，则结束，否则返回步骤（2）。

用流程图表示，则如图 1.2 所示。

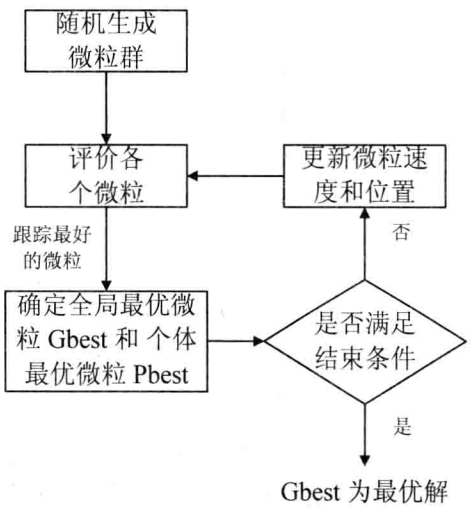



图 1.2　微粒群算法流程

关于 PSO 算法的收敛性问题，尽管公式（1.4）中的 $v_i$、$x_i$ 为多维变量，但是其每维均相互独立，因此对微粒群算法的分析可以简化到一维进行。同时，按照文献[254]的假设，$\omega$、$p_i(t)$、$p_g(t)$ 时不变，且对于种群的最优微粒有等式 $p_i(t)=p_g(t)$ 成立，又记 $p=p_i(t)=p_g(t)$，$\varphi_1(t)=c_1r_1(t)$，$\varphi_2(t)=c_2r_2(t)$，$\varphi(t)=\varphi_1(t)+\varphi_1(t)$，将迭代次数挪到下标位置，则公式（1.4）可以改写为如下形式

$$\begin{bmatrix} v_{t+1} \\ x_{t+1} \end{bmatrix}=\begin{bmatrix} \omega & -\phi_t \\ \omega & 1-\phi_t \end{bmatrix}\begin{bmatrix} v_t \\ x_t \end{bmatrix}+\begin{bmatrix} \phi_t p \\ \phi_t p \end{bmatrix} \tag{1.6}$$

记

$$\mathbf{A}=\begin{bmatrix} \omega & -\phi_t \\ \omega & 1-\phi_t \end{bmatrix}$$

若将 $\varphi_t$ 看作常量，则微粒群算法的动态特性可简化为线性时不变二阶系统。

$\boldsymbol{A}$ 阵的特征多项式为

$$\lambda^2+(\varphi_t-1-\omega)+\omega$$

对应的特征值为 $\lambda_1$、$\lambda_2$，微粒轨迹收敛的条件是 $\max(\|\lambda_1\|,\|\lambda_2\|)\le 1$。

文献[249-255]分析了算法的收敛性，均基于上述结论，即均基于以下认识：只要系统矩阵的特征根在单位圆内，即能保证系统稳定，也即保证算法收敛。然而，该结论只是线性时不变离散系统渐进稳定的充分必要条件，对于微粒群算法而言，算法所对应的系数矩阵 $\boldsymbol{A}$ 是一个带有随机变量的时变系数矩阵，而不是常矩阵，其谱半径在单位圆内是微粒群算法稳定的既非充分也非必要条件。为此，文献[255]给出了基于随机过程的 PSO 算法收敛性分析，在满足式（1.4）的假设条件下，若式（1.7）和式（1.8）成立，则 PSO 算法的微粒轨迹均方收敛于 $p$。

$$\begin{cases} 0<\mu<2(\omega+1) \\ |\omega|<1 \end{cases} \tag{1.7}$$

$$\begin{cases} 1+a_2+a_1+a_0>0 \\ 1-a_2+a_1-a_0>0 \\ 1>|a_0| \\ 1-a_0^2>|a_1-a_2a_0| \end{cases} \tag{1.8}$$

其中，

$$\begin{cases} a_2=\omega-\mu^2-\sigma^2 \\ a_1=\mu^2\omega-\sigma^2\omega-\omega^2 \\ a_0=-\omega^3 \end{cases} \tag{1.9}$$

微粒群算法的研究现状可归结为 5 个部分[36]，即算法本身的改进、算法参数的改进、算法拓扑结构的改进、算法的融合和算法的应用。本书感兴趣的是算法应用。在应用方面主要有：

（1）优化问题的求解。用于具有约束条件的优化、多目标优化、离散优化以及动态跟踪优化的求解等[36]。

（2）自动控制。用于控制器的设计，提高系统响应效率[37]。

（3）图像处理。在图像分割和配准、图像融合、图像识别、图像压缩等方面得到成功应用[36]。

（4）神经网络训练。用于人工神经网络连接权值的训练、结构设计、学习规则调整、输入特征选取、连接权值的初始化等[36]。

（5）电力系统设计及优化。将电力企业中的某些问题转化为函数的最小值问题，使用 PSO 算法进行优化求解。实验结果证明了 PSO 算法在解决该问题的优势[38]。

（6）其他。如半导体器件综合、生物信号识别、航运优化、系统辨识等[36]。

### 1.1.6　多层前馈神经网络

多层前馈神经网络（MLPNN）通常指的是 BP 网络。它由输入层、中间层和输出层组成。中间层即隐层，可以为一层或多层。一般情况下，隐层数量选为一层。MLPNN 属于静态网络，它是最重要的神经网络之一，被研究的最多，应用得也最广泛。

1986 年，Rumelhart 等提出了 BP 算法，系统地解决了多层神经元网络中隐单元层接连权的学习问题，并在数学上给出了完整的推导。由于 BP 算法克服了简单感知器不能解决的 XOR 及其他一些问题，所以，BP 模型已成为神经网络的重要模型之一，并得以广泛使用。

MLPNN 各隐节点的激活函数使用 Sigmoid 函数，其输出节点的激活函数根据应用的不同而异：如果多层感知器用于分类，则输出层节点一般用 Sigmoid 函数或硬极限函数；如果多层感知器用于函数逼近，则输出层节点应该用线性函数。MLPNN 采用多层结构，包括输入层、各个隐层、输出层，各层之间实现全连接。图 1.3 给出了隐节点和输出节点都使用 Sigmoid 函数的 BP 神经网络结构，图中各神经元的阈值没有画出。

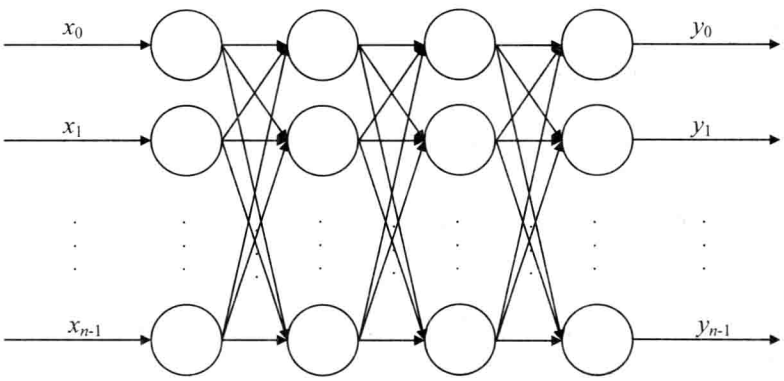



图 1.3　MLPNN 结构

对于 MLPNN 神经网络的各个计算节点，有

$$u_j = \sum_{i=1}^{n} w_i x_i - \theta_j \tag{1.10}$$

$$y_j = f\left(u_j\right) = \frac{1}{1+\exp\left(-\lambda u_j\right)} \tag{1.11}$$

式（1.11）给出的是单极 S 函数，双极 S 函数亦可，实际上，只要是连续可微单调上升函数即可。

$$y_j^{'}=f^{'}\left(u_j\right)=\frac{\lambda\exp\left(-\lambda u_j\right)}{1+\exp\left(-\lambda u_j\right)}\frac{1}{\times 1+\exp\left(-\lambda u_j\right)}=\lambda\left[1-f\left(u_j\right)\right]f\left(u_j\right) \tag{1.12}$$

假设 MLPNN 神经网络的输入矢量为 $x\in R^n, x=\left(x_0,x_1,\cdots,x_{n-1}\right)^{\mathrm{T}}$，第一隐层有 $n_1$ 个神经元，它们的输出为 $x^{'}\in R^{n_1}, x^{'}=\left(x_0^{'},x_1^{'},\cdots,x_{n-1}^{'}\right)^{\mathrm{T}}$，第二隐层 $n_2$ 有个神经元，它们的输出为 $x^{''}\in R^{n_2}$, $x^{''}=\left(x_0^{''},x_1^{''},\cdots,x_{n-1}^{''}\right)^{\mathrm{T}}$，输出层有 $m$ 个神经元，输出 $y\in R^m, y=\left(y_0,y_1,\cdots,y_{n-1}\right)^{\mathrm{T}}$。设输入层到第一隐层的权为 $\omega_{ij}$，阈值为 $\theta_j$，第一隐层到第二隐层的权为 $\omega_{jk}^{'}$，阈值为 $\theta_k^{'}$，第二隐层到输出层的权为 $\omega_{kl}^{''}$，阈值为 $\theta_l^{''}$。于是，各层神经元的输出为

$$\begin{cases} x_j^{'}=f\left(\sum\limits_{i=0}^{n-1}\omega_{ij}x_i-\theta_j\right), j=0,1,2,\cdots,n_1-1 \\ x_k^{''}=f\left(\sum\limits_{i=0}^{n_1-1}\omega_{jk}^{'}x_j^{'}-\theta_k^{'}\right), k=0,1,2,\cdots,n_2-1 \\ y_l=f\left(\sum\limits_{i=0}^{n_2-1}\omega_{kl}^{''}x_k^{''}-\theta_l^{''}\right), l=0,1,2,\cdots,m-1 \end{cases} \tag{1.13}$$

显然，它完成了 $n$ 维空间到 $m$ 空间的映射。

下面给出所有训练样本同时用于网络权值调节（即批处理）的 BP 算法。设有 $P$ 个学习样本矢量，对应的输出期望为 $\boldsymbol{d}^{(1)},\boldsymbol{d}^{(2)},\cdots,\boldsymbol{d}^{(p)}$，学习是通过误差校正权值使各 $\boldsymbol{y}^{(p)}$ 接近 $\boldsymbol{d}^{(p)}$。为简化推导，把各计算节点的阈值并入权矢量，即设 $\theta_l^{''}=\omega_{n_2 l}^{''}, \theta_k^{'}=\omega_{n_1 k}^{'}$ $\theta_j=\omega_{nj}, x_{n_2}^{''}=x_{n_1}^{'}=x_n=-1$，则式（1.13）中相应的矢量 $w$，$w^{'}$，$w^{''}$，$x$，$x^{'}$，$x^{''}$ 的维数均增加 1。

该算法的学习规则是基于最小均方误差准则。当一个样本（设为第 $p$ 个样本）输入网络并产生输出时，均方误差应为各输出单元误差平方之和，即

$$E^{(P)}=\frac{1}{2}\sum_{i=0}^{m-1}\left(d_l^{(p)}-y_l^{(p)}\right)^2 \tag{1.14}$$

当所有样本都输入一次后，总误差为

$$E_T=\sum_{p=1}^{P}E^{(p)}=\frac{1}{2}\sum_{p=1}^{P}\sum_{i=0}^{m-1}\left(d_l^{(p)}-y_l^{(p)}\right)^2 \tag{1.15}$$

设 $w_{sp}$ 为网络中的一个连接权值，则根据梯度下降法，批处理方式下的权值修正量应为

$$\Delta w_{sp} = -\frac{\partial E_T}{\partial w_{sp}} \tag{1.16}$$

增量方式下的权值修正量应为

$$\Delta w_{sp} = -\frac{\partial E^{(p)}}{\partial w_{sp}} \tag{1.17}$$

下面以批处理为例加以讨论，并假设S函数的增益 $\lambda = 1$。

对输出层而言，

$$\omega''_{kl}(t+1) = \omega''_{kl}(t) - \eta \frac{\partial E_T}{\partial \omega''_{kl}} \tag{1.18}$$

式中，t为迭代次数。$\frac{\partial E_T}{\partial \omega''_{kl}}$ 求解过程采用 $\delta$ 学习规则，即

$$\begin{aligned}
\frac{\partial E_T}{\partial \omega''_{kl}} &= \sum_{p=1}^{P} \frac{\partial E_T}{\partial \omega''_{kl}} \\
&= \sum_{p=1}^{P} \frac{\partial E^{(p)}}{\partial y_l^{(p)}} \frac{\partial y_l^{(p)}}{\partial u_l''^{(p)}} \frac{\partial u_l''^{(p)}}{\partial \omega''_{kl}} \\
&= -\sum_{p=1}^{P} \left(d_l^{(p)} - y_l^{(p)}\right) f'\left(u_l''^{(p)}\right) x_k''^{(p)} \\
&= -\sum_{p=1}^{P} \left(d_l^{(p)} - y_l^{(p)}\right) y_l^{(p)} \left(1 - y_l^{(p)}\right) x_k''^{(p)} \\
&= -\sum_{p=1}^{P} \delta_{kl}^{(p)} x_k''^{(p)}
\end{aligned}$$

式中，$\delta_{kl}^{(p)} = \left(d_l^{(p)} - y_l^{(p)}\right) y_l^{(p)} \left(1 - y_l^{(p)}\right)$。所以有

$$\omega''_{kl}(t+1) = \omega''_{kl}(t) - \eta \sum_{p=1}^{P} \delta_{kl}^{(p)} x_k''^{(p)} \tag{1.19}$$

注意：在推导过程中，$E^{(P)}$ 是 $y_0, y_1, y_2, \cdots, y_{m-1}$ 的函数，但 $\omega''_{kl}$ 只影响 $y_l$；$\partial u_l''^{(p)} = \sum_{k=0}^{n_2} \omega''_{kl} x_k''^{(p)}$；$y_l^{(p)} = f\left(u_l''^{(p)}\right)$；$f'\left(u_l''^{(p)}\right) = y_l^{(p)}\left(1 - y_l^{(p)}\right)$。

对中间隐层而言，

$$\omega'_{jk}(t+1) = \omega'_{jk}(t) - \eta \frac{\partial E_T}{\partial \omega'_{jk}} \tag{1.20}$$

$\frac{\partial E_T}{\partial \omega'_{jk}}$ 求解过程采用 $\delta$ 学习规则，即

$$\frac{\partial E_T}{\partial \omega'_{jk}}=\sum_{p=1}^{P}\frac{\partial E^{(p)}}{\partial \omega'_{jk}}$$

$$=\sum_{p=1}^{P}\sum_{i=0}^{m-1}\frac{\partial E^{(p)}}{\partial y_l^{(p)}}\frac{\partial y_l^{(p)}}{\partial u_l''^{(p)}}\frac{\partial u_l''^{(p)}}{\partial x_k''^{(p)}}\frac{\partial x_k''^{(p)}}{\partial u_k'^{(p)}}\frac{\partial u_l'^{(p)}}{\partial \omega'_{jk}}$$

$$=-\sum_{p=1}^{P}\sum_{i=0}^{m-1}\left(d_l^{(p)}-y_l^{(p)}\right)f'\left(u_l''^{(p)}\right)\omega''_{kl}x_k''^{(p)}\left(1-x_k''^{(p)}\right)x_j'^{(p)}$$

$$=-\sum_{p=1}^{P}\sum_{i=0}^{m-1}\delta_{kl}^{(p)}w''_{kl}x_{kl}''^{(p)}\left(1-x_k''^{(p)}\right)x_j'^{(p)}$$

$$=-\sum_{p=1}^{P}\delta_{kl}^{(p)}x_j'^{(p)}$$

式中，$\delta_{kl}^{(p)}=\sum_{i=0}^{m-1}\delta_{kl}^{(p)}w''_{kl}x_k''^{(p)}\left(1-x_k''^{(p)}\right)$。所以有

$$\omega'_{jk}(t+1)=\omega'_{jk}(t)+\eta\sum_{p=1}^{P}\delta_{jk}^{(p)}x_j'^{(p)} \tag{1.21}$$

注意：在推导过程中，$E^{(P)}$ 是 $y_0,y_1,y_2,\cdots,y_{m-1}$ 的函数，每个 $y_l$ 都受 $\omega'_{jk}$ 的影响。

同理，可以得到第一隐层的权值修正公式为

$$\omega_{ij}(t+1)=\omega_{ij}(t)+\eta\sum_{p=1}^{P}\delta_{ij}^{(p)}x_i^{(p)} \tag{1.22}$$

式中，$\delta_{kl}^{(p)}=\sum_{k=0}^{n_2}\delta_{kl}^{(p)}w'_{jk}x_j'^{(p)}\left(1-x_j'^{(p)}\right)$。

对于增量式修正，上面各式中各权值的修正量是一项，而不是 1~$p$ 的求和。

显然，学习分两个阶段：由前向后向计算各隐层和输出层的输出，由后向前误差反向传播用于权值修正。

综合以上，可以总结 BP 算法步骤如下：

（1）权值初始化。$w_{sp}$ =random（.），$sp$ 为 $ij$、jk 或 $kl$。

（2）依次输入 $P$ 个学习样本，设当前输入为第 $p$ 个样本。

（3）依次计算各层的输出。$x'_j$、$x''_k$ 及 $y_l$，$j$=0,1,2，…，$n_1$；$k$=0,1,2，…，$n_2$；$l$=0，1,2，…,$m$−1。

（4）求各层的反传误差。

$$\delta_{kl}^{(p)}=\left(d_l^{(p)}-y_l^{(p)}\right)y_l^{(p)}\left(1-y_l^{(p)}\right),\qquad l=0,1,2,\cdots,m-1$$

$$\delta_{jk}^{(p)}=\sum_{i=0}^{m-1}\delta_{kl}^{(p)}w''_{kl}x_k''^{(p)}\left(1-x_k''^{(p)}\right),\qquad k=0,1,2,\cdots,n_2$$

$$\delta_{ij}^{(p)}=\sum_{k=0}^{n_2}\delta_{jk}^{(p)}w'_{jk}x_j'^{(p)}\left(1-x_j'^{(p)}\right),\qquad j=0,1,2,\cdots,n_1$$

并记下各个 $x_k''^{(p)}$，$x_j'^{(p)}$ 及 $x_i^{(p)}$ 的值。

（5）记录已经学习过的样本个数 $p$。如果 $p<P$，则转到步骤（2）继续计算，如果 $p=P$，则转到步骤（6）。

（6）按权值修正公式修正各层的权值。

（7）按照新的权值再计算 $x_j^{'}$、$x_k^{''}$、$y_l$ 和 $E_T$，若对每个 $p$ 和 $l$ 都满足 $\left|(d_l^{(p)}-y_l^{(p)})\right|<\varepsilon$（或 $E_T<\varepsilon$）达到最大学习次数，则终止学习，否则，转到步骤（2）继续新一轮的学习。

虽然 BP 算法得到广泛的应用，但它也存在不足，其主要表现在训练过程不确定上，具体如下：

（1）训练时间较长。对于某些特殊的问题，运行时间可能需要几个小时甚至更长，这主要是因为学习率太小导致，可以采用自适应学习率加以改进。

（2）完全不能训练。训练时由于权值调整过大使激活函数达到饱和，从而使网络权值的调节几乎停滞。为避免这种情况，一是选取较小的初始权值，二是采用较小的学习率。

（3）易陷入局部极小值。BP 算法可以使网络权值收敛到一个最终解，但它并不能保证所求为误差超平面的全局最优解，也可能是一个局部极小值。这主要是因为 BP 算法所采用的是梯度下降法，训练是从某一起始点开始沿误差函数的斜面逐渐达到误差的最小值，故不同的起始点可能导致不同的极小值产生，即得到不同的最优解。如果训练结果未达到预定精度，常常采用多层网络和较多的神经元，以使训练结果的精度进一步提高，但与此同时也增加了网络的复杂性与训练时间。

（4）“喜新厌旧”。训练过程中，学习新样本时有遗忘就样本的趋势。

在提高收敛速度和避免局部最小方面，已有许多学者进行了研究并提出了很多方法，如附加动量法、自适应学习率、竞争 BP 算法、牛顿法、弹性 BP 算法、改进误差函数等。

### 1.1.7 径向基神经网络

1985 年，Powell 提出了多变量插值的 RBF 方法，1988 年，Broomhead 和 Lowe 首先将 RBF 应用于神经网络设计，构成了 RBF 神经网络。RBF 神经网络一般为 3 层结构，如图 1.4 所示。

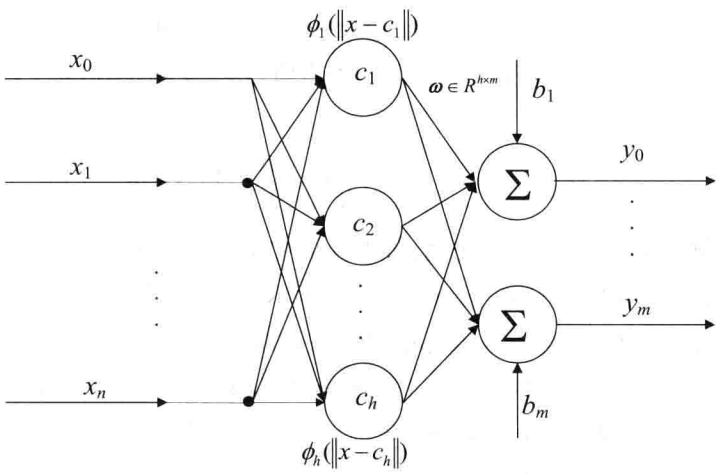



图 1.4 RBFNN 网络结构

图 1.4 所示的 RBF 神经网络结构为 $n$—$h$—$m$，即网络具有 $n$ 个输入、$h$ 个隐节点、$m$ 个输

出，其中，$\boldsymbol{x}=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}\in R^n$ 为网络输入矢量，$\boldsymbol{\omega}\in R^{h\times m}$ 为输出权矩阵，$\boldsymbol{b}=[b_1,b_2,\cdots,b_m]^{\mathrm{T}}$ 为输出单元偏移，$\boldsymbol{y}=[y_1,y_2,\cdots,y_m]^{\mathrm{T}}$ 为网络输出，$\phi(\bullet)$ 为第 $i$ 个隐节点的激活函数。图中输出层节点中的Σ表示输出层神经元采用线性激活函数（输出神经元也可以采用其他非线性激活函数，如 Sigmoid 函数）。多层感知器神经网络的隐节点基函数采用线性函数，激活函数采用 Sigmoid 函数或硬极限函数，而 RBF 神经网络最显著的特点是隐节点采用距离函数（如欧氏距离），并使用 RBF（如高斯函数）作为激活函数。RBF 关于 $n$ 维空间的一个中心点具有径向对称性，而且神经元的输入离该中心点越远，神经元的激活程度就越低，隐节点的这个特性常被称为“局部特性”。因此，RBF 神经网络的每个隐节点都具有一个数据中心，如图 1.4 中 $c_i$ 就是网络中第 i 个隐节点的数据中心值，$\|\bullet\|$ 则表示欧氏距离。

RBF $\phi(\bullet)$ 可以取多种形式，如式（1.23）~式（1.25）所示。

（1）高斯函数。

$$\phi(u)=\mathrm{e}^{\frac{u^2}{\delta^2}} \tag{1.23}$$

（2）反射 Sigmoid 函数。

$$\phi(u)=\frac{1}{1+\mathrm{e}^{\frac{u^2}{\delta^2}}} \tag{1.24}$$

（3）逆多二次函数。

$$\phi(u)=\frac{1}{\left(u^2+\delta^2\right)^{\frac{1}{2}}} \tag{1.25}$$

式（1.23）~式（1.25）中的 $\delta>0$ 称为该基函数的扩展常熟或宽度。显然，$\delta$ 越小，RBF 的宽度就越小，基函数就越具有选择性，扩展常数取值为 1。于是在图 1.4 中，RBF 神经网络的第 j 个输出可表示为

$$y_j=\sum_{i=1}^{h}\omega_{ij}\phi\left(\|x-c_i\|\right),\quad 1\le j\le m \tag{1.26}$$

下面以二输入单输出函数逼近为例，简要介绍 RBF 神经网络的工作原理。输出节点相连的隐层第 $i$ 个隐节点的所有参数可用三元组 $(c_i,\delta_i,\omega_i)$ 表示。由于每个隐层神经元都对输入产生响应，且响应特性呈径向对称（只要输入模式离数据中心的距离相等，节点输出就相等，即是一个个同心圆）。假设输入区域中有 6 个神经元，每个神经元都对输入 $x$ 产生一个响应 $\phi_i\left(\|x-c_i\|\right)$，而神经网络的输出则是所有这些响应的加权和由于每个神经元具有局部特性，最终整个 RBF 神经网络也呈现“局部映射”特性，即 RBF 神经网络是一种局部响应神经网络，这意味着如果神经网络有较大的输出，必定激活了一个或多个隐节点。

假定共有 $N$ 个学习样本，其输入为 $\boldsymbol{s}=[X_1,X_2,\cdots,X_N]$，相应的样本输出即教师信号（单输出）为 $\boldsymbol{t}=[y_1,y_2,\cdots,y_N]$。所谓的多变量内插问题是指寻找函数，使之满足以下的内插条件：

$$y_i=F(X_i) \tag{1.27}$$

这是一个数学问题，可有多种解决方案，采用 RBF 神经网络也可解决这个问题的学习问题。由式（1.27）可知，使用 RBF 神经网络前必须确定其隐节点数据中心（包括数据中心的数目、数据中心值和扩展常数值）及相应的一组权值。RBF 神经网络解决内插问题时，一种方案是使用 $N$ 个隐节点，并把所有的样本输入选为 RBF 神经网络的数据中心，且各基函数取相同的扩展常数，于是，RBF 神经网络从输入层到隐层的输出便是确定的，然后确定网络的 $N$ 个输出权值 $w=[w_1,w_2,\cdots,w_N]^{\mathrm{T}}$（待定）。只要把所有的样本再输入一遍，便可解出各 $\omega_i$ 的值。假定当输入为 $X_i(i=1,2,\cdots,N)$ 时，第 $j$ 个隐节点的输出为

$$h_{ij}=\phi_j\left(\left\|X_i-c_j\right\|\right) \tag{1.28}$$

式中，$\phi_j(\bullet)$ 为该隐节点的激活函数；$c_j=X_j$ 为该隐节点 RBF 的数据中心。于是，可定义 RBF 神经网络的隐层输出阵为 $\boldsymbol{H}=\left[h_{ij}\right],\boldsymbol{H}\in R^{N\times N}$。此时，RBF 神经网络的输出为

$$F(X_i)=\sum_{j=1}^{N}h_{ij}w_j=\sum_{j=1}^{N}w_j\phi_j\left(\left\|X_i-c_j\right\|\right) \tag{1.29}$$

令 $y_i=F(X_i),\boldsymbol{y}=t^{\mathrm{T}}=[y_1,y_2,\cdots,y_N]^{\mathrm{T}},\boldsymbol{w}=[w_1,w_2,\cdots,w_N]^{\mathrm{T}}$，便得

$$\boldsymbol{y}=\boldsymbol{H}\boldsymbol{w} \tag{1.30}$$

如果 $\boldsymbol{H}$ 可逆，即其列向量构成 $R^N$ 中的一组基，则输出权矢量为

$$\boldsymbol{w}=\boldsymbol{H}^{-1}\boldsymbol{y} \tag{1.31}$$

根据 Micchelli 定理，如果隐节点激活函数采用上述的 RBF，且 $X_1,X_2,\cdots,X_N$ 各不相同，则隐层输出阵 $\boldsymbol{H}$ 的可逆性是可以保证的。因此，如果把全部样本输出作为 RBF 神经网络的数据中心，网络在样本输入点的输出就等于教师信号，此时，网络对样本实现了完全内插，即对所有样本误差为 0。但式（1.31）内插方案存在以下问题：

（1）通常情况下，样本数据较多，即 $N$ 数值较大，上述方案中隐层输出阵 $\boldsymbol{H}$ 的条件数可能过大，求逆时可能导致不稳定。

（2）如果样本输出含有噪声，由于存在过学习的问题，做完全内插是不合适的，而对数据做有限精度逼近可能更合理。

为了解决这些问题，可以采用正则化网络。正则化网络具有以下特点：

（1）具有万能逼近能力，即只要有足够的隐节点，正则化网络能逼近紧急上的任意连续函数。

（2）具有最佳逼近特性，即任给未知的非线性函数，总可以找到一组权系数，在该组系数下正则化网络对该函数的逼近优于其他系数。

（3）正则化网络得到的解是最优的，即通过最小化式，得到同时满足对样本的逼近误差和逼近曲线平滑性的最优解。

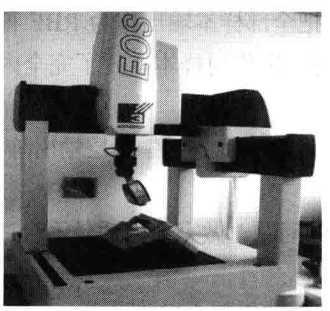
对于某一 RBF 神经网络，如果给定了训练样本，那么，该网络的学习算法应该解决以下问题：结构设计（即如何确定网络隐节点数 $h$）、确定各 RBF 的数据中心 $c_i$ 及扩展常数 $\delta_i$ 、输出权值修正。一般情况下，如果知道了网络的隐节点数、数据中心和扩展常数，RBF 神经网络从输入到输出就成了一个线性方程组，此时，权值学习可采用最小二乘法求解。

在 MLPNN 和 RBFNN 的应用方面，已有一些学者对它们的特定应用领域作了综述。例如，文献[39]总结了 1988 年到 1995 年，文献[40]总结了 1992 年到 1998 年，包括 MLPNN 和 RBFNN 在内的 ANN 在商业上的应用情况，并作了对比分析；文献[41]给出了它们在制造领域的应用综述；文献[42]总结分析了神经网络在金融领域的应用；文献[43]则对神经网络在管理科学上的应用作了综述。最近的一篇文献[44]给出了人工神经网络截至 2007 年的几乎所有的应用情况。

MLPNN 与 RBFNN 都是通用的逼近器，万能逼近定理[45]保证了它们均能实现非线性映射。二者在对生物系统的模拟、网络结构与表达能力、训练算法和逼近方式等方面有所差异。MLPNN 一般采用 BP 训练算法，该算法存在训练速度慢，易陷入局部最优等问题，本书在第六章中，应用 MLPNN 进行曲面重构时，提出了一种改进的 BP 训练算法。RBFNN 结构简单，训练速度快，采用局部逼近方式，表达能力强，然而 RBFNN 径向基函数中心的选择仍是一个没有得到很好解决的问题，本书第五章引入微粒群算法在初始聚类获得径向基函数中心的基础上解决该问题。

## 1.2 三维表面扫描技术应用现状

随着时代的进步和技术的发展，三维表面扫描技术得到了迅猛发展。三维表面扫描设备即三维扫描仪，如雨后春笋般涌入市场。三维扫描仪可以分为接触式与非接触式两大类，后者又可分为两类，一种是“激光式”（即主动测量模式），另一种是“照相式”（即被动测量模式）。目前常见的典型“激光式”三维扫描仪见图 1.5。“照相式”扫描仪，其工作过程类似于照相过程，扫描物体的时候一次性扫描一个测量面，因此得名。



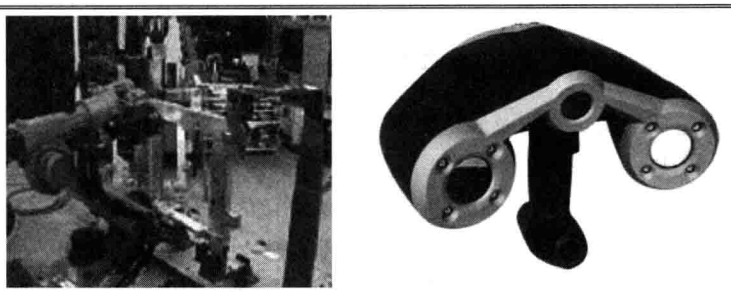



图 1.5 典型的“激光式”三维扫描仪

三维视觉传感器是三维扫描仪中的关键部件，图 1.6 左图是法国 Kreon 公司的 ZEPHYR-KZ50 三维视觉传感器。图 1.6 右图是法国 Kreon 公司 ZEPHYR-KZ100 三维视觉传感器。ZEPHYR-KZ100 具有传统接触式扫描和现代非接触式扫描的双重功能，用户可选择视觉调整获取更高精度。

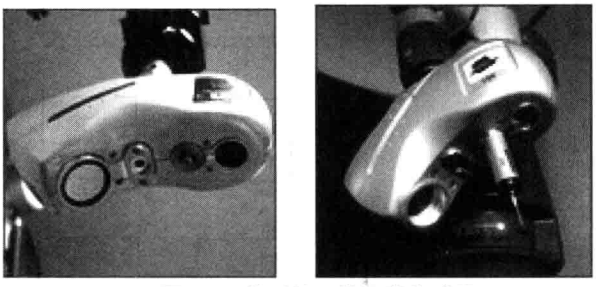


图 1.6 典型的三维视觉传感器

三维表面扫描技术是一项非接触、自动化的在线质量控制技术，可以对物体的几何尺寸等进行无人为干涉的主动测量，在工业、航空航天、军事、能源等各个领域都具有非常广阔的应用。其主要的具体应用领域如下：

（1）工业。工业生产中几乎所有行业（汽车制造、钢铁、木材、纺织、制药、食品饮料等）产品的生产和装配、外形检验、包装印刷检验等方面[46-49]。

（2）航空航天。航天器等大尺寸物体检测及装配，地形、地貌识别和理解等[50-52]。

（3）军事。现代军事设备，如舰、船、艇等制造与装配过程的数字化测量等[53]。

（4）能源。能源设备，如涡轮机、蒸汽轮机叶片几何尺寸、形面检测，输油管道表面缺陷检测等[54, 55]。

（5）其他。刑侦领域用于提取现场痕迹；动漫领域用于三维动画建模[56]等。

## 1.3 三维表面扫描机器人系统

对于具有占地面积小、数据获取速度快等优点的三维表面扫描机器人系统，不仅能够充分发挥机器人运动灵活的特点，而且可以随时变换程序即能够满足产品多品种、多系列的三维扫描需求，特别适合在生产线上对大型零部件进行非接触、快速、精确在线测量，具有不可替代的优势。其应用领域亦十分广泛，下文给予具体介绍。