第 七 章　　人工神经网络的优化

7.1 人工神经网络（Artificial Neural Network，简称 ANN）

早在 19 世纪末，人类就发现大自然赋予自身的头脑具有许多绝妙之处。准确地说，大脑是由大量的神经元经过复杂的相互连接而形成的一种高度复杂、非线性、并行处理信息的系统。它使得人类能够快速地从外界环境中摄取大量的信息，并加以处理、存储，及时地对环境的变化做出各种响应，并不断向环境学习，从而提高人类的适应能力。而这一切均有赖于大脑的物质基础—神经网络。

从那时起，人类就梦想着能够从模仿人脑智能的角度出发，去探寻新的信息表示、存储、处理方式，从而构建一种全新的、接近人类智能的信息处理模型。1943 年，McCulloch 和 Pitts 根据心理学家 James 所描述的神经网络的基本原理[James W 1890]，建立了第一个人工神经网络模型（后被扩展为“认知模型”）[McCulloch and Pitts 1943]，可用来解决简单的分类问题。

1969 年，Minsky 和 Papert 在《认识论（Perceptrons）》一书中指出，McCulloch 和 Pitts 所提出的认知模型无法解决经典的异或（XOR-exclusive-or）问题。这个结论曾一度使人工神经网络的研究陷入危机。实际上这一结论是非常片面的，因为 Minsky 和 Rumelhart 主要研究的是单隐含层的认知网络模型，而简单的线性感知器功能是有限的，这一结论不应该对人工神经网络进行全面否定。

20 世纪 80 年代，Hopfield 将人工神经网络成功地应用于组合优化问题上[Hopfield 1985，1986]，McClelland 和 Rumelhart 构造的多层反馈学习算法成功地解决了单隐含层认知网络的“异或问题”及其他的识别问题[McClelland 1988]，这些突破重新掀起了人工神经网络的研究热潮。

由于人工神经网络具有较强的自适应性、学习能力和大规模并行计算能力，目前已被广泛应用于各种研究及实际工程领域中，如模式识别、信号处理、控制与优化、预测建模、通信等领域。

虽然人工神经网络已具有较成熟的模型、数学理论基础，但依然有许多因素制约着其发展。下面首先介绍人工神经网络的基本知识及研究内容，以期明确神经网络优化过程中的任务和难点。

7.1.1　人工神经网络的基本概念

人工神经网络模型是基于生物学中神经网络的基本原理而建立的一种模仿人脑工作方式的计算模型，可以被视为一种具有大量连接的并行分布式处理器，它可以通过学习获取知识并解决问题，并且将知识分布存储在连接权中（对应生物神经元的突触）。

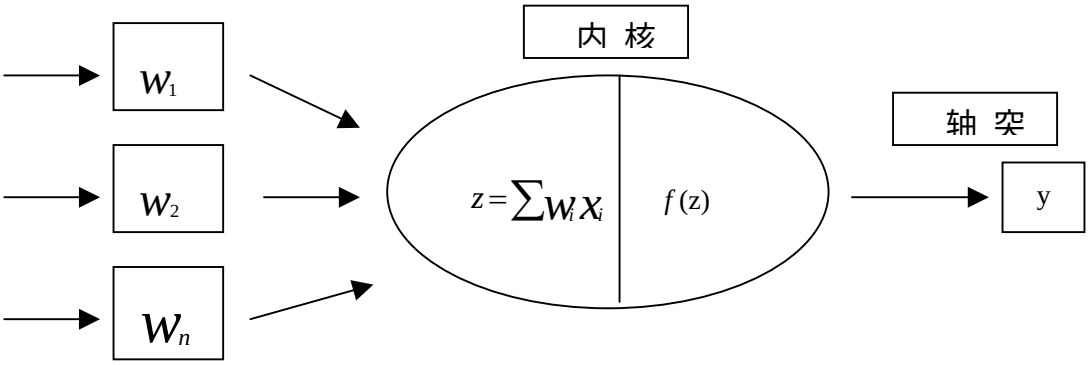


图 7.1　McCulloch — Pitts 认知网络

图 7.1 表示了 McCulloch 和 Pitts 所提出的认知网络，又称 MP 模型。这一认知网络实际上模拟了神经网络中一个神经元加工处理信息的过程。在这个神经元模型中存在三个基本要素：

1．$W_i \cdots W_n$,一组连接权，设神经元收到几个信息，则 W_i 表示神经元对接收到第 i 个信息的感知能力。

2．一个求和单元，用于求取各输入信息的线性加权和。

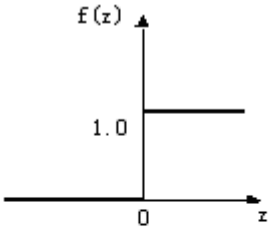
3．一个非线性激励函数 *f*(z)，起非线性映射作用并限制神经元的输出幅度在一定范围内（一般在[0，1]或[-1，1]）。

MP 模型中的激励函数采用的是阈值函数(图一)，故输出函数定义如下：

$$y = f(z) = \mathrm{sgn}(\sum_{i=1}^{n} w_i x_i - \theta)$$

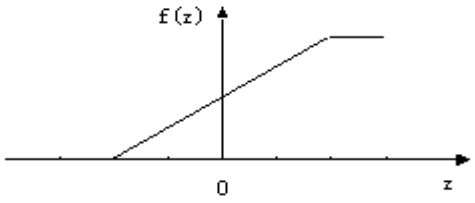
其中θ称为阈值。

除了阈值函数以外，神经网络中常采用的激励函数还有分段线性函数与 sigmoid 函数，其中 Sigmoid 函数具有平滑、单调和渐近性，经常用于不同函数的的非线性映射问题中。



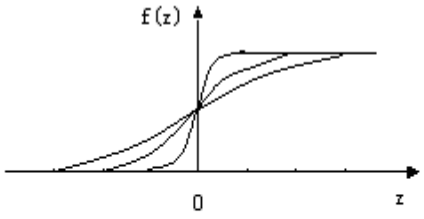
图一 阈值函数

$$f(z) = \begin{cases} 1, z \geq 0 \\ 0, z < 0 \end{cases}$$



图二 分段线性函数

$$f(z) = \begin{cases} 1, & z \geq 1 \\ \frac{1}{2}(1+z), & -1 < v < 1 \\ 0, & v \leq -1 \end{cases}$$



图三 Sigmoid 函数

$$f(z) = \frac{1}{1 + e \times p(-\alpha z)}$$

图 7.2 几种常见的激励函数

对于特定的神经网络，当权值为固定值时，给定一组输入，则很容易计算得出输出值，但通常对于神经网络的期望是：对任意给定的输入，网络的输出尽可能的与实际值相吻合。因此，在解决实际问题中，应该建立什么样的网络模型，如何确定适当的权值，采用何种学习方法，才能使神经网络具有高度的智能性而正确地去求解问题，这些才是我们首要考虑的问题。

通常从连接方式来看，人工神经网络有两类主要模型，前馈型网络和反馈型网络。目前神经网络的优化研究主要集中在前馈网络上，故在以下内容中将详细讨论前馈型神经网络的工作机理。

7.1.2 前馈神经网络（feed forward neural networks）

前面所介绍的 MP 模型实际上可以视为一个线性阈值单元，其激励函数采用的是阈值函数。研究表明，线性阈值单元可以实现任一布尔乘积或任一布尔求和项，由若干个线性阈值单元所构成的单层前馈网络（这类网络主要用在识别问题中，又称感知器，仅有一层计算单元）只能实现线性可分函数。要想增加神经网络对复杂问题的分类能力，唯一的方法是采用多层前馈网络，即在输入、输出层之间增加隐层以构成多层前馈网络（Multi-layer feed forward neural networks）。由于任何布尔函数都可以化为析取范式，因此均可用一个三层的前馈型神经网络实现；如果隐层的作用函数采用连续函数（如 Sigmoid 函数），则网络输出可以逼近一个连续函数。具体地说，假设网络有 P 个输入、Q 个输出，则其作用可看作是由 P 维欧氏空间到 Q 维欧氏空间的一个非线性映射。许多研究表明，只要隐层的神经元个数足够多，则这种映射可以逼近任何连续函数。也就是说，只含一个隐层的前馈网络是一种通用函数逼近器，为逼近一个连续函数，一个隐层就足够了，当然这并不意味着从网络结构、学习速度等方面看，一个隐层是最好的。

由于多层前馈网络具有很强的分类能力，不仅能够解决单层感知器所不能解决的经典异或问题，而且能够实现任意连续函数的逼近问题，因此对多层前馈网络的研究具有很大的实际意义。图 7.3 所示的是一个多层前馈神经网络的拓扑结构。

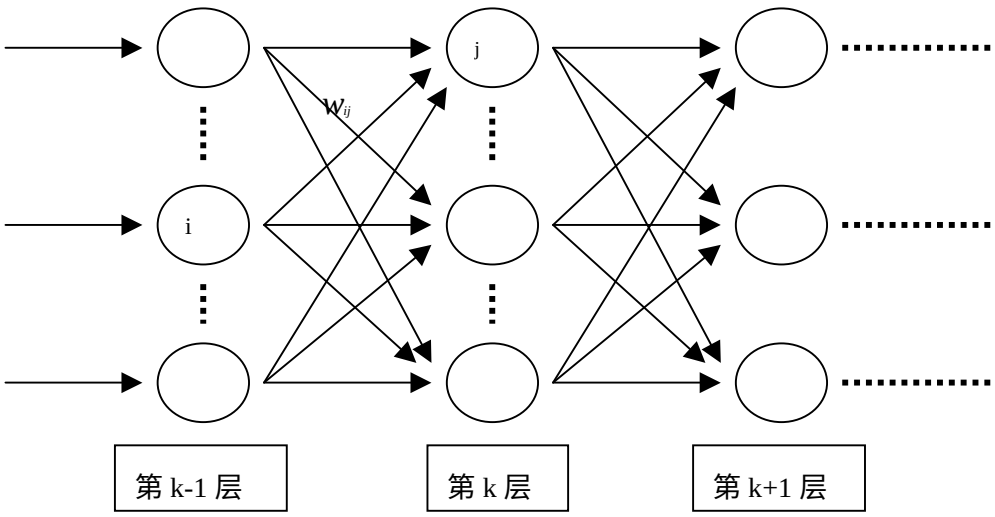


图 7.3　　多层前馈神经网络

在多层前馈网络中，同一层内的神经元之间没有任何联系，各层中的神经元接受前一层的输入，并输出给下一层。除输入层和输出层之外的均称为隐层，输入层记为第 0 层，输出层所在的层数为神经网络的层数。

通过向环境学习获取知识并改进自身性能是神经网络的一个重要特点，在一般情况下，性能的改善是按某种预定的度量通过调节自身参数，通过一定时间的训练而逐步达到的。人工神经网络的计算通常分为两个阶段，第一为学习阶段，这个阶段的主要工作是确定并调节权值；第二为应用阶段，在已确定权值的基础上，用确定权值的神经网络去解决实际问题。

网络学习（或称训练）通常有两种方式：监督学习(有教师学习)与无监督学习（无教师指导学习）。前者需要外界存在一个“ 教师 ”可对给定网络的一组输入提供应有的正确输出，这组已知的输入—输出数据称为训练样本集，而学习网络可以根据已知输出与实际输出之间的差值来调节系统参数。无监督学习方式中不存在外部“ 教师 ”，系统完全按照环境提供数据的某些统计规律自发调节自身参数或结构，以表示出外部输入的某种固有特性。相比较而言，监督学习方式研究的最为充分，因此在讨论多层前馈神经网络时，都采用监督学习方式。

既然多层前馈网络可以解决非线性可分问题，采用 Sigmold 函数（S 形函数）作为激励，可

以逼近任意一个连续函数，我们可以采用监督学习方式，利用误差最小原则对其进行反复训练，以求其具有较强的自适应求解能力。除此之外，我们还要具体考虑网络的学习规则问题，即采用何种方法去调整、确定权值参数。对于一个结构复杂的网络来说，学习训练过程是一个非常复杂的过程，尤其是多层前馈网络，隐层的增加，虽然提高了网络的分类能力，但是却增加了学习的难度。反向传播算法的出现使这一问题迎刃而解。

图 7.4 表示的是采用反向传播算法（Back Propagation，简称 BP 算法）的多层前馈网络中的信号流图。

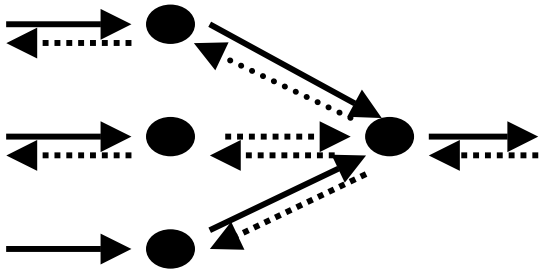


图 7.4　　采用 BP 算法前馈网络中的信号流图

其中有两种信号在流通：一是工作信号（实线），沿输入端到输出端方向向前传播的信号，它是输入信号与权值的函数。二是误差信号（虚线），网络实际输出与应有输出间的差值即为误差，它由输出端开始逐层向后反向传播

假设在第 n 次迭代中输出端的第 j 个单元的输出 $y_j(n)$,则该单元的误差信号为：

$$e_i(n) = d_i(n) - y_i(n) \tag{7.1}$$

设其平均误差为 $\frac{1}{2}e_j^2(n)$,则输出端总的平方误差瞬时值为：

$$\varepsilon(n) = \frac{1}{2}\sum_{j\in m} e_j^2(n) \quad (\text{m 为输出单元个数}) \tag{7.2}$$

假设训练样本总数为 N 个，则平均误差的均值为：

$$\varepsilon_{AV} = \frac{1}{n}\sum_{n=1}^{N}\varepsilon_{(n)} = \frac{1}{2N}\sum_{n=1}^{N}\sum e_j^2(n) \tag{7.3}$$

这就是多层前馈网络学习的目标函数，学习的目的是使均方误差达到最小。这一目标是通过反复学习、不断调整权值而实现的。传统的 BP 学习算法中采用基于梯度信息调整权值的学习规则，学习过程中权值可按如下公式进行调整。

设 w_{ji} 为单元 i 到单元 j 的连接权，则：

$$\Delta w_{ji} = -\eta\,\delta_j(n)\,y_i(n) \tag{7.4}$$

其中，η 为学习步长，$y_i(n)$ 为单元 i 向单元 j 的输入信号，$\delta_j(n)$ 为局部梯度，负号表示权值的修正沿梯度下降方向进行。

从上述分析中可知，BP 算法为多层前馈网络的训练提供了有效的学习方法，但是如果采用基于梯度信息调整连接权的学习规则，则在训练求解复杂问题的神经网络时，往往存在着计算复杂、学习速度慢且易于陷入局部极值等缺陷。

7.1.3 网络的泛化能力

通常，我们最关心的是所设计的神经网络是否有较强的泛化能力(Generalization Ability)。所谓网络的泛化能力，是指经训练后的网络对未在训练集中出现的（但来自同一分布的）样本做出正确反应的能力。也就是说，网络的学习不是单纯的记忆已学过的输入，而是通过训练样本学习到隐含在样本中的有关环境本身的内在规律性，从而对未来出现的输入也能给出正确的反应。

影响网络泛化能力的因素主要有三种：

1. 训练样本的质量和数量
2. 网络结构
3. 问题本身的复杂程度

显而易见，求解问题的复杂程度实际上是不可控制的，因此网络泛化能力的大小主要取决于前两个因素。针对这两个因素，我们需要讨论两个问题：

<一>.当网络结构一定（根据求解问题由经验选定），为达到较好的泛化能力，需要多少训练样本。

<二>.训练样本量一定时，如何确定网络规模以保证其具有较好的泛化能力。

为了保证网络具有一定的泛化能力，这就要求在学习过程中选择适当的训练样本集，选定的训练样本集应当包含能够反映环境变化内在规模的所有信息。研究表明：当训练样本趋于无穷时，通过训练样本学到的权值参数在概率上收敛于真正要求的权值。但实际上，训练样本集总是有限的，因此，在网络训练过程中，就不可避免会出现“过拟合现象”(Over fitting)。

也就是说，在神经网络的学习过程中如果过分追求训练集内误差最小，就会使得训练后的网络因学习过多的特殊样本，只记住了个别特例以至某些噪声，从而未能学到真正的规律，以至于遇到未出现过的输入而难以给出正确反应，即不具备泛化能力或泛化能力较差。因此，选取训练样本时，一定要考虑训练样本的数量。一般说来，当网络结构一定时（权数一定），则训练样本量应当比可调参数大几倍，这样学习结果才是可靠的。

为避免出现“过拟合现象”，在检验网络性能时，应采用训练集以外的样本。通常把给定问题的样本分成两组，一组作为训练集用于训练网络，另一组作为检验学习结果，称作测试集。实际中，应合理划分样本，使得既充分利用有限的样本去训练网络，又能较好地检验训练结果。

7.1.4 神经网络结构设计

神经网络拓扑结构是网络设计的一个重要内容，它直接影响着网络的泛化能力，当结构规模过大，会造成过度拟合现象。通常，网络结构一定时，其中待学习的参数是有限的，故参数学习的搜索空间也是有限的。当缺乏先验知识进行网络结构设计时，由于结构的搜索空间非常大，对应于每个结构又有许多可能的参数组合，因而这个过程要比参数学习复杂困难的多，故可以认为神经网络结构的设计，实际上是一个复杂的优化问题。

目前，网络结构设计倘无确定可循的理论方法，一个公认的指导原则是：在没有其它先验知识时，能与给定样本相符合的最简单的网络就是最佳选择。具体的方法有逐步增长法、逐步修剪法以及正规化约束法（regularization）[Reed 1993]。

逐步增长法是先从一个较简单的网络开始，逐步增加隐单元数目直到满足要求为止。

逐步修剪发则相反，从一个较复杂的网络开始，逐步删除隐单元数目直至满足要求为止。

正规化约束法是在目标函数中，增加对模型复杂性的约束。

从前述内容的讨论中可以看出，针对实际工程问题设计神经网络的过程，实际上可以归结为权植参数的学习与合理的网络结构的选择这两类典型的优化问题。由于神经网络的训练是一个极其复杂的过程，传统的学习算法往往需要很长时间（当优化采用 BP 算法的多层前馈网络时，还要用到梯度），才能在一定精度范围内找到符合要求的网格结构与参数。因此，为提高网络的设计

速度及泛化能力，各种全局优化算法越来越受到相关研究人员的青睐。

7.2 进化计算用于神经网络的优化

近年来，随着进化计算研究热潮的兴起，人们逐渐将进化计算与人工神经网络相结合，利用各种进化方法去训练神经网络。由于进化算法具有较强的全局收敛能力和较强的鲁棒性，且不需要借助问题的特征信息，如导数等梯度信息。因此，将两者相结合，不仅能发挥神经网络的泛化映射能力，而且能够提高神经网络的收敛速度及学习能力。进化计算用于神经网络优化主要有两个方面：一是用于网络训练，既优化网络各层之间的连接权值；二是优化网络的拓扑结构。具体的研究方法有如下三种：

1 通过优化连接权值来训练神经网络

针对特定的神经网络，列出所有的神经元，并将所有神经元可能存在的连接权值编码成二进制码串或是实数码串表示的个体，随机生成这些码串的群体，按照常规的方法执行进化操作。将新形成的码串解码构成神经网络，计算所有训练样本通过此神经网络产生的平均误差，以此来确定每一个体的适应度。这种方法思路简单，但是用于优化的计算量较大，尤其是在优化解决复杂问题的大规模神经网络时，随着神经元数目的增多，连接权的总数也随之增多，即而造成进化计算的搜索空间急剧增大。

2 优化神经网络的结构和学习规则

利用进化计算优化设计神经网络的结构，同时包括网络的学习规则以及与之关联的参数。这种方法直接将未经训练的神经网络的结构模式和学习规则编码成码串表示的个体，再执行进化操作。与前一种方法相比，其搜索空间相对较小，但是也存在着一定的缺陷。在优化过程中，每个被选择的个体都必须解码成未经训练的神经网络，再经过传统的训练以确定其连接权值，因此收敛速度较慢。

3 同时优化神经网络的结构和连接权值

同时对神经网络的结构和连接权值进行编码以进行优化。Vittorio 曾提出粒度编码方法来提高连接权值的优化精度，但是粒度控制同时会加剧个体适应度的不连续变化，从而影响到算法的收敛速度。

采用进化计算去优化神经网络，比起基于梯度的 BP 学习算法，无论是精度还是速度上，均有了很大的提高。然而，作为一种仿生的随机算法，进化计算本身又具有不可克服的缺陷。比如进化计算中研究最为充分的遗传算法，虽然它可以用来求解各类复杂问题，但总是难以克服过早收敛的缺点，同时在采用遗传操作进化时，需要的控制参数过多，尤其是在优化神经网络时候，优化过程总是难以控制。因此，为神经网络的优化寻求更简单、更有效的全局优化算法，是优化领域中的一个研究热点。

7.3 用 PSO 优化神经网络

作为一种简单、有效的随机搜索算法，PSO 同样可用来优化神经网络。尽管这一方面的研究尚处于初期阶段，但是已有的研究成果表明 PSO 在优化神经网络方面具有很大的潜力。

7.3.1 训练神经网络的 PSO 算法的设计

1、问题的描述

采用 PSO 训练神经网络时，首先应将特定结构中所有神经元间的连接权值编码成实数码串表示的个体。假设网络中包含 M 个优化权值（包括阈值在内），则每个个体将由 M 个权值参数组成的一个 M 维向量来表示。

例如：给定如下结构的神经网络，其中包括一个阈值，一维输入，两个隐层单元，一维输出，

从图中可知其中包括 6 个连接权，分别是 $\{W_{11},W_{12},W_{21},W_{22},W_{Y_1},W_{Y_2}\}$，令 $X_1=W_{11},X_2=W_{12},X_3=W_{21},X_4=W_{22},X_5=W_{Y_1},X_6=W_{Y_2}$，则微粒群中的个体可用一个 6 维向量来表示，即 $Indv=\{X_1,X_2,X_3,X_4,X_5,X_6\}$。此时，个体结构中的每一个元素，即代表神经网络中的一个权值。

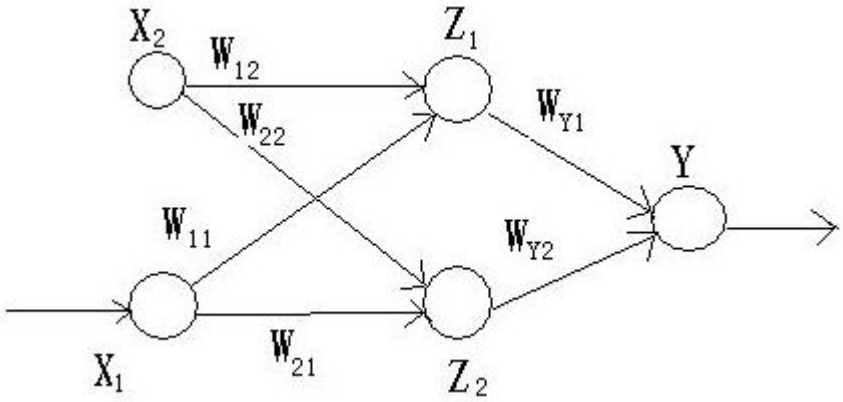


图 7.5　一个多层前馈神经网络

2、初始化微粒群

根据微粒群规模，按照上述个体结构随机产生一定数目的个体（微粒）组成种群，其中不同的个体代表神经网络的一组不同权值。同时初始化 g_{best}，l_{best}。

3、神经网络的训练及微粒的评价

将微粒群中每一个体的分量映射为网络中的权值，从而构成一个神经网络。对每一个体对应的神经网络，输入训练样本进行训练。网络权值的优化过程是一个反复叠代的过程。通常为了保证所训练的神经网络具有较强的泛化能力，在网络的训练过程中，往往将给定的样本空间分为两部分，一部分用作训练样本，称为训练集，一部分作为测试样本，称为测试集。而在权值优化过程中，每进行一次训练，都要对给定的样本集进行分类，以保证每次训练时采用的训练集均不相同。

计算每一个网络在训练集上产生的均方误差，并以此作为目标函数，并构造如下的适应度函数，用来计算个体的适应度。

$$E(X_p)=\frac{1}{2n}\sum_{p=1}^{n}\sum_{k=0}^{c}\left(Y_{K,P}(X_P)-t_{k,p}\right) \tag{7.5}$$

其中 $t_{k,p}$ 指训练样本 P 在 K 输出端的给定输出，则适应度函数定义如下：

$$f(x)=\frac{1}{1+E(X)} \tag{7.6}$$

4、PSO 模型计算

评价微粒群中的所有个体（每一个体视为可飞行的微粒），从中找到最佳个体用来判断是否需要更新微粒的 Gbest 与 Lbest。之后，按照 PSO 模型更新每一个体不同分量上的飞行速度，并以此产生新的个体微粒。

5、算法的终止条件

当目标函数值（即均方误差）小于给定的 $\varepsilon \to 0$ 时，算法终止。

用 PSO 训练神经网络算法的具体流程如下：

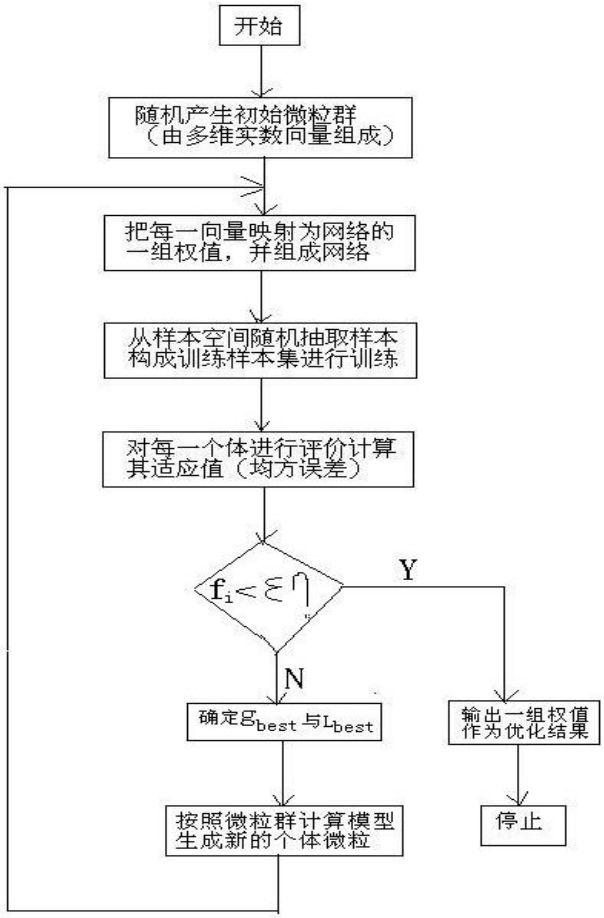


图 7.6　PSO 优化神经网络流程图

7.3.2 算法的评价及分析

评价 PSO 算法在神经网络训练中的优化能力，通常可通过其训练求解特定问题的神经网络的性能好坏来判断。对于前馈神经网络的优化，采用的测试问题主要有分类问题及函数逼近问题。对于优化算法在网络训练中的性能指标通常有四种：

一、训练集中分类错误率

设训练中样本数目为 M_T，训练过程中分类失败的样本数目为 m_T，则

$$\varepsilon_T = \frac{m_T}{M_T} \times 100\% \tag{7.7}$$

定义为训练集上的分类误差。

二、测试集上的分类错误率

设测试集中的测试样本总数为 M_G，对训练后的网络进行测试时，分类失败的样本数目为 m_G，则定义测试集上的分类错误率如下：

$$\varepsilon_G = \frac{m_G}{M_G} \times 100\% \tag{7.7}$$

三、训练集上的均方误差

$$MSE_T = \frac{1}{2M_G} \sum_{P=1}^{M_G} \sum_{K=0}^{C} \left(Y_{k,p}\left(X_p\right) - t_{k,p} \right)^2 \tag{7.8}$$

四、测试集上的均方误差

$$MSE_G = \frac{1}{2M_G} \sum_{P=1}^{M_G} \sum_{k=0}^{C} \left(Y_{K,p}\left(X_p\right) - t_{k,p} \right)^2 \tag{7.9}$$

用多层前馈网络求解分类问题时，四个性能指标都可选用，而对于函数逼近和拟合问题中，只能利用 MSE_T 与 MSE_G 对训练结果进行评估。

7.4 协同 PSO 算法优化神经网络

7.4.1 协同 PSO 算法

所谓的协同进化，是指将解空间中的群体划分为若干子群体，每个子群体代表求解问题的一个子目标，所有子群体在独立进化的同时，基于信息迁移与知识共享，共同进化。

协同进化算法中最常见的协同模型是“ 孤岛模型 ”(island model) 与“ 邻域模型 ”(Neighbourhood Model）[H.Mulenbein 1989]。在这两种模型中，直接将群体中的个体划分为若干子群体，每一子群体代表解空间中的一个子区域（子空间），其中的每一个体均代表问题的一个解。所有子群体并行展开局部搜索，所搜索到的优良个体将在不同子群体间进行迁移，作为共享信息指导进化的进行，从而有效的提高算法的全局收敛效率。

除此之外，Potter 曾提出了另外一种协同进化模型（Cooperative Coevolutionary Genetic Algorithm，简称 CCGA[J.Pearsall 1999; Mitchell A. Potter 1994]）。在 CCGA 算法中，子群体的构成采用一种截然不同的划分方式。假设一个待求解问题的解空间被映射为一个包含 m 个个体的群体，而其中的每一个体均由一个 n 维向量表示，则 CCGA 将群体中的所有个体划分为 n 个一维向量个体，然后同一分量方向上的 m 个一维向量相互组合，从而形成 n 个子群体。显而易见，此时每个子群体并不能独立求解优化问题，问题的可行解必须由来自 n 个不同子群体中的 n 个个体共同组合而构成，因此在优化过程中，所有子群体必须进行相互协调，共同进化以求取问题的最优解。

此处即将介绍的协同 PSO 算法（CPSO）[Van den Bergh 与 Engelrecht 2000,2001]，其思想类似于 CCGA 算法的协同模型，同样采用沿不同分量划分子群体的原则。但是，这种算法在优化过程中各分量是独立调整计算的，因此往往会出现这样一种现象：在某一更新代后，尽管局部范围内适应函数值有所增长，但这一结果却有可能是由微粒在某些分量上背向而行所造成的，这种现象势必削弱了算法的全局收敛性，而且使搜索易于陷入局部极值。这就是所谓的“ 两步向前、一步向后 ” 现象。

CPSO 算法以一定的计算费用为代价有效地解决了这一问题。在每一操作代中，问题的解是按照每一分量依次进行更新的。当某一子群体中的微粒沿着某一分量方向飞行一次后，随即便与其它子群体搜索到的当前最优相组合，构成的新解用来对其进行适应度评价，以确定此次飞行的成败。在这之后，下一个子群体的微粒再进行飞行。依次类推，直至所有子群体循环完毕。可以看出，上述操作使得 CPSO 能够以较大的机率收敛于全局最优解，但同时却增加了许多计算量。为了平衡这种矛盾，Van den Bergh 与 Engelrecht 在 CPSO 算法中，引入了一种灵活的子群体划分

方案，将群体中的所有向量（尽量按照分量间的相关性大小）划分为 K 个部分（K<n，称为分裂因子），从而形成 K 个子群体。在每一操作代中，问题的解是基于 K 个子群体依次进行更新的，从而有效的限制了 CPSO 为克服“ 两步向前、一步向后 ”现象而增加的计算代价。上述算法被称为带有分裂因子的 CPSO，简称 CPSO-S 算法。其中分裂因子的取值直接影响着优化算法的性能。

7.4.2 协同 PSO 算法优化神经网络应用实例

Van den Bergh 与 Engelrecht 将 CPSO 算法用于神经网络的训练[Van den Bergh 与 Engelrecht 2001]，并通过大量的分类问题与函数逼近问题，对其所训练的神经网络进行了详细的性能分析。下面的内容将详细介绍相关的结果。

1、实验中选用的神经网络及测试问题

Frans Van Den Bergh 在实验中选用了两类典型的前馈网络，加和神经网络（Summation-unit Networks，SUN）与倍乘神经网络（Product-unit Networks, PUN）。这两类网络均具有良好的非线性映射能力，可用来求解各种分类问题及函数逼近问题。我们将基于这两种网络的优化对 PSO 算法展开讨论。

图 7.7 表示的是一个两层加和神经网络，又可称为多层感知器，它只包含一个隐层。

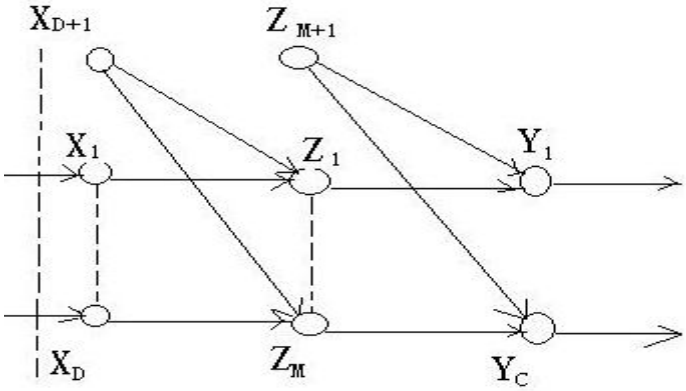


图 7.7　加和神经网络

此神经网络具有 D 个输入单元外加一个阈值单元，C 个输出，隐层包含 M 个单元和一个阈值，隐层采用 Sigmoid 函数作为激励函数。网络输出如下：

$$Y_k(X)=\sum_{j=1}^{M+1}W_{kj}\,g\left(\sum_{i=1}^{D+1}W_{ji}X_i\right) \tag{7.10}$$

其中激励函数为

$$g(a)=\frac{1}{1+\exp(-a)} \tag{7.11}$$

W_{kj} 指的是隐层单元 j 与输出单元 k 之间的连接权，而 W_{ji} 指的是输入单元 i 与隐层单元 j 之间的连接权，$1\le i\le D+1,1\le j\le M+1,1\le k\le c$。

由 Sigmoid 函数可知，当 $|a|>10$，$g(a)\approx 0$ 或 $g(a)\approx 1$，类似阶梯函数。如果输入层与隐层单元间的连接权很小，如 $|a|<2$，则 $g(a)$ 近乎是线性的，因此可实现相当平滑的网络映射；当隐层单元数目足够多时，加和神经网络能够以任意精度逼近任何连续函数。

图 7.8 表示的是一个两层倍乘神经网络[Durbin,Rumelhart 1989]。

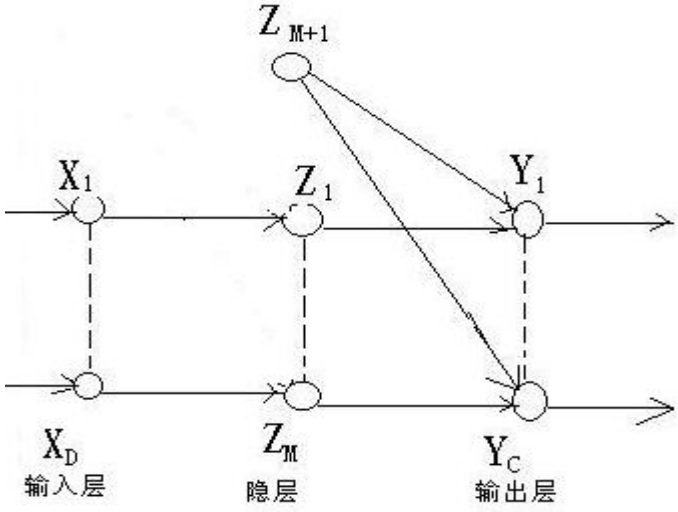


图 7.8　倍乘神经网络

其中含有 D 维输入，M 个隐层单元，C 维输出。隐层作为乘积单元，输出层则采用线性加和的输出方式，此时网络的输出如下：

$$Y_k = \sum_{j=0}^{M+1} w_{kj} \prod_{i=1}^{D} x_{i,p}^{w_{ji}} \tag{7.12}$$

其中，w_{kj} 为输出层单元 k 与隐层单元 j 间的连接权，w_{ji} 为隐层单元 j 与输入层单元 i 之间的连接权。

两类网络均采用监督学习方式，优化过程中利用如下的均方误差作为目标函数：

$$E = \frac{1}{2n}\sum_{p=1}^{n}\sum_{k=0}^{c}\left(y_{k,p}(X_p) - t_{k,p}\right)^2 \tag{7.13}$$

其中$t_{k,p}$指训练样本 p 在 k 输出端的给定输出。

Van den Bergh 与 Engelrecht 基于多种分类问题及函数逼近问题，利用 PSO 算法分别对加和网络和倍乘网络进行优化。在以下所有的试验中，加和网络和倍乘网络都给予一指定的简单结构[Y.Lecun, J.S.Denker and S.A. Solla. 1990]算法的优化结果）。具体信息见下表：

表 7.1　求解不同问题的神经网络

	优化问题	样本数目	输入	输出	指定结构	权值数	
						SUN	PUN
分类问题	Iris	150	4	3	4-4-3	35	31
	BreastCancer	600	9	1	9-8-1	89	81
	Wine	178	13	3	13-5-3	88	83
	Diabetes	700	8	1	8-16-1	161	145
	Hepatitis	154	19	2	19-9-1	190	181
函数拟合	HenonMap	100	2	1	2-5-1	21	16
	CubicFunction	100	1	1	1-2-1	7	5

从上表可以看出，在给定结构、样本量有限的情况下，求解同样的分类与函数拟和问题，倍乘网络比加和网络所需的连接权数少，这也说明了前者比后者具有更强的存储能力。但是，当利用传统的采用基于梯度的训练方法对网络进行学习时，后者的训练难度要比前者大的多，因为其中的乘积项会造成很多扰动误差，相比之下，遗传算法、微粒群等全局优化算法，则更适合于倍乘网络的优化。

下面的研究内容基于上表所列的多种分类问题及函数逼近问题，利用 PSO 算法，分别对加和网络和倍乘网络进行优化。对于给定的神经网络结构，只需对连接权进行编码，将其映射为码串表示的个体，同时将训练中产生的均方误差作为评价个体的目标函数，此时神经网络的训练问题就可转化为寻找一组使均方误差最小的最佳连接权值的优化问题。

2、实验参数

实验中采用带有惯性权重因子的 PSO 计算模型，其中 $\omega = 0.72, c_1 = c_2 = 1.49$，微粒群规模为 10。每一实验中算法均反复运行 50 次，每运行一次，神经网络要经过 50 个不同的训练集进行训练。

对于复杂神经网络来说，由于其包含的权系数特别多，因此个体所对应的向量往往是高维的。为提高训练效率与精度，实验中采用 CPSO-S 协同算法训练网络，因此还必须为算法的分裂因子 K 确定适当的取值。Van den Bergh 与 Engelrecht 通过大量的实验发现了一个有趣的结果：CPSO-S 算法优化网络时采用的最佳分裂因子取值与网络连接权的数目有直接的关系，当两者近似符合下述关系式时, CPSO-S 算法所训练的神经网络具有较理想的性能。

$$K = a\sqrt{W} \tag{7.14}$$

其中 W 代表网络的连接权数目，a 是一个经验常数（对于加和网络取 a=1.3；对于倍乘网络取 a=0.5）。

3、实验结果

实验中 Van den Bergh 与 Engelrecht 利用表 7.1 中各种分类问题及函数逼近问题，采用 CPSO 算法训练神经网络，并基于训练结果对 CPSO 与其它算法的优化能力进行了比较。下面摘录了部分实验结果。

实验一：利用加和与倍乘两种前馈网络对下面的立方函数进行逼近:

$$f(Z) = Z^3 - 0.04Z \tag{7.15}$$

其中权值的训练利用 CPSO-S 算法。实验中两种网络均采用 1-2-1 的简单结构，CPSO 算法的各参数取值如下：$W = 0.72, C_1 = C_2 = 1.49$，微粒群规模为 10，分裂因子 K 分别为 2（SNN）与 3(PNN)。实验结果见下表：

表 7.2　用 CPSO 算法训练 SNN 求解立方函数逼近问题的两种均方误差

Algorithm	MSE_T	MSE_G
GD	$1.00e-04 \pm 3.61e-06$	$2.19e-04 \pm 1.82e-05$
SCG	$1.06e-04 \pm 1.57e-06$	$2.21e-04 \pm 1.70e-05$
CCGA	$1.06e-04 \pm 5.70e-06$	$2.29e-04 \pm 2.51e-05$
GA	$1.17e-05 \pm 5.66e-06$	$2.38e-05 \pm 1.16e-05$
$CPSO-S_2$	$9.45e-06 \pm 7.56e-06$	$2.39e-05 \pm 2.06e-05$

表 7.3　用 CPSO 算法训练 PNN 求解立方函数逼近问题的两种均方误差

Algorithm	MSE_T	MSE_G
GD	$2.74e-01\pm4.05e-01$	$5.38e-01\pm6.85e-01$
SCG	$1.69e-01\pm1.77e-01$	$1.05e-01\pm1.14e-01$
CCGA	$4.29e+140\pm8.62e+140$	$1.84e+103\pm3.71e+103$
GA	$5.99e-04\pm2.77e-04$	$9.84e-04\pm3.34e-04$
$CPSO-S_3$	$3.35e-05\pm3.04e-05$	$8.12e-05\pm7.77e-05$

实验二：用ＣＰＳＯ算法训练神经网络求解爱尔兰综合症分类问题：

给定样本１５０个，根据４种特征输入来区分三种典型的爱尔兰综合症。实验中，两种网络均采用４-４-３的结构。其中，加和网络具有３５个权值参数，最佳的分裂因子为７；倍乘网络具有３１个连接权，最佳的分裂为６。ＣＰＳＯ其它参数与实验一相同。

表 7.4 用 CPSO 算法训练 SNN 求解 Iris 分类问题的两种均方误差

Algorithm	MSE_T	MSE_G
GD	1.16 ± 0.32	5.10 ± 1.15
SCG	1.56 ± 0.34	4.80 ± 0.90
CCGA	21.53 ± 1.41	28.07 ± 1.75
GA	25.22 ± 1.78	30.17 ± 1.85
$CPSO-S_3$	0.62 ± 0.24	5.80 ± 091

表 7.5　用 CPSO 算法训练 PNN 求解 Iris 分类问题的两种均方误差

Algorithm	MSE_T	MSE_G
GD	90.44 ± 3.02	91.33 ± 3.01
SCG	90.93 ± 2.84	91.33 ± 2.63
CCGA	25.89 ± 4.07	32.10 ± 3.82
GA	37.29 ± 3.62	42.93 ± 3.78
$CPSO-S_6$	14.16 ± 2.66	25.50 ± 3.33

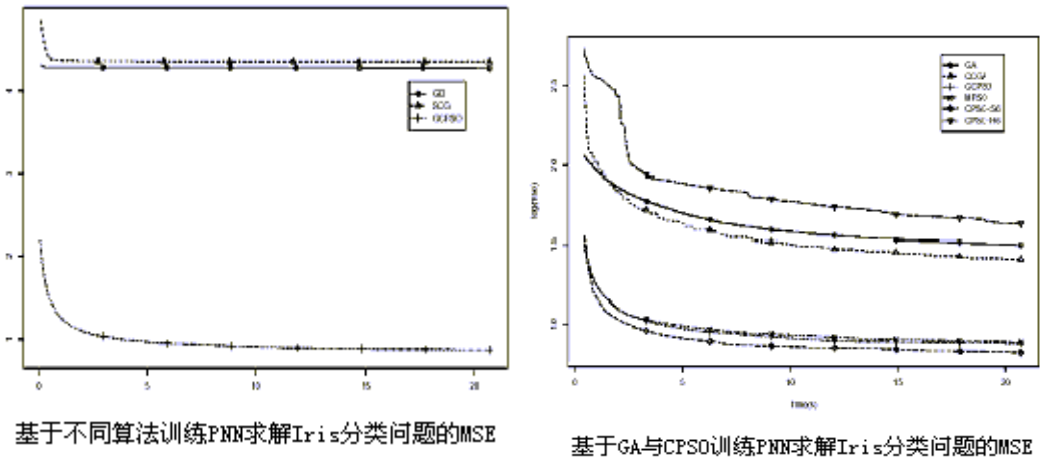


图 7.9 不同算法训练 PNN 网络的求解结果

在上述图表中，GD（Gradient Descent）与 SCG(A Scaled Conjugate Gradient Descent Algorithm)是两种基于梯度的训练算法；CCGA [Potter 1994]是指协同遗传算法。

上述图表以训练集与测试集上的两种均方误差作为性能指标，对比了不同算法所训练的神经网络的优劣。四张表中的数据表明，CPSO 算法所训练的网络无论是在训练集还是测试集中，所得到的均方误差都最小，这说明 CPSO 算法训练网络的效果最好，其所训练的网络具有最强的泛化能力。同时图 7.9 中的不同均方误差曲线图显示，为达到某一给定的均方误差，CPSO 算法所需的时间最短，这又说明了采用 CPSO 算法训练网络具有较快的速度。

故根据实验结果可以得到如下结论：在训练求解分类与函数逼近问题的神经网络时，采用 PSO 算法所训练的网络所达到的精度及泛化能力要优于基于梯度的学习方法及遗传算法。

本部分讨论的内容并没有重点考虑神经网络训练中的过度拟合问题，只是在实验中采用较小的网络结构来避免，因而没有涉及网络结构的优化。从实验结果来看，所训练的网络一定程度上均存在过度拟合问题，因为所有网络在测试集上的误差均大于其在训练集上的均方误差。因此，为 PSO 寻找一种有效的学习机制解决结构优化问题，从而使 PSO 所训练的神经网络具有更强的泛化能力，并有效克服过度拟合问题，这也是 PSO 应用领域中待研究的一个重点内容。