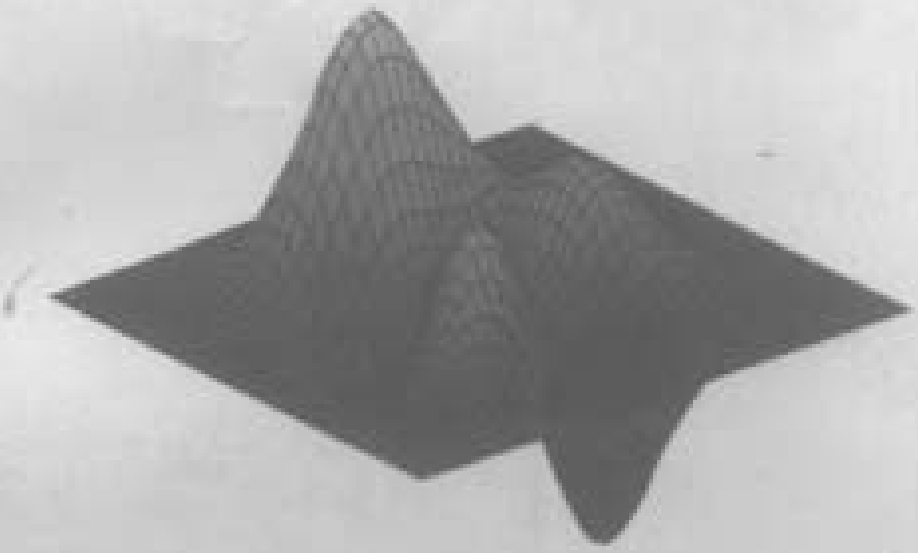
智能故障诊断与专家系统

吴今培　肖健华 著



Intelligent FD and ES

科学出版社

智能故障诊断与专家系统

吴今培　肖健华　著

科学出版社

1997

内 容 简 介

本书主要从知识处理的角度系统地论述了现代机器设备智能故障诊断的原理和方法，模糊逻辑、人工神经网络和专家系统在复杂设备状态监测及故障诊断中的应用.

全书共分六章，内容包括：诊断知识处理，诊断征兆获取和诊断问题求解，传统故障诊断专家系统，神经网络故障诊断专家系统，模糊神经网络故障诊断专家系统，智能故障诊断技术的发展和展望. 为便于读者应用智能诊断理论解决工程实际问题，书中还附有各种智能故障诊断系统的应用程序. 这些程序经过多次检验，证明行之有效，可运行在各类微机上.

本书可供航空、航天、冶金、化工、机械、电子、电力、交通等领域从事设备工况监视、故障诊断、可靠性和维修性工程的技术人员、科研与开发人员学习参考，也可作为高等学校计算机、机电工程、信息处理、人工智能等有关专业高年级大学生或研究生的教学参考书.

图书在版编目(CIP)数据

智能故障诊断与专家系统/吴今培，肖健华著. —北京：科学出版社，1997

ISBN 7-03-005848-8

Ⅰ.智… Ⅱ.①吴… ②肖… Ⅲ.故障诊断-专家系统诊断法 Ⅳ.TP277

中国版本图书馆 CIP 数据核字(97)第 10154 号

科学出版社出版

北京东黄城根北街 16 号

邮政编码：100717

北京双青印刷厂 印刷

新华书店北京发行所发行 各地新华书店经售

*

1997 年 9 月第 一 版 开本：850×1168 1/32

1997 年 9 月第一次印刷 印张：11 1/4

印数：1—5 000 字数：292 000

定价：20.00 元

序

我怀着十分高兴的心情，阅读了吴今培教授所著的《智能故障诊断与专家系统》手稿．该书对近年来故障诊断专家系统技术的发展做了较为系统的总结，同时也将作者多年来的诸多研究成果融入其中，较为全面地反映了这一领域的国内外研究现状．

智能诊断技术自80年代中期在国内开展以来，经过广大科技工作者十多年的不懈努力，不仅在理论研究方面取得了许多重要的研究成果，而且在工程应用方面也积累了宝贵的实践经验．与此同时，作者也培养了一大批从事这一研究工作的专门人才，作者所在的研究集体也做了许多卓有成效的研究工作．但目前还比较缺乏专门论述这一研究领域的学术著作，因此，该书的出版对于总结这一领域现有的研究成果，促进这一领域工作的进一步深入发展，具有重要意义．

该书对各类典型专家系统进行了深入浅出的阐述，不仅内容新，而且重实用．书中提供的智能诊断系统实例，对于广大工程技术人员从事这方面的研究和应用工作具有重要的参考价值．同时，该书适合于各种层次读者的需要，既是一本很好的教学用书，又是一本很好的工程研究参考书．我非常赞成出版这部书，希望该书的出版能够在促进这一领域的深入研究和推广应用这一领域的成果方面起到积极的作用．

值本书出版之际，谨为作序，并祝愿本书出版成功，也祝我国的故障诊断事业更加繁荣昌盛．

中国科学院院士
华中理工大学校长
杨叔子

1997年4月

前　言

众所周知，给人诊病的科学是医学，给机器设备诊病的科学被称为“故障诊断学”. 与历史悠久和成果辉煌的医学相比，故障诊断学在世界上还是一门新兴学科.

所谓故障诊断，就是鉴别机器设备的技术状态是否正常，确定故障的性质，发现故障的部位，寻找故障起因，提出排除故障的相应措施. 早在本世纪 60 年代，在美国、英国和日本等少数工业发达国家，就掀起了现代设备故障诊断研究的热潮，并在工程应用中发挥了重要作用，取得了显著的社会经济效益.

故障诊断技术发展至今已经历了三个阶段：第一阶段由于机器设备比较简单，故障诊断主要依靠专家或维修人员的感觉器官、个人经验及简单仪表就能胜任故障的诊断与排除工作；传感器技术、动态测试技术及信号分析技术的发展使得诊断技术进入了第二阶段，并且在维修工程和可靠性工程中得到了广泛的应用；80 年代以来，由于机器设备日趋复杂化、智能化及光机电一体化，传统的诊断技术已不能适应了，随着计算机技术、人工智能技术特别是专家系统的发展，诊断技术进入了它的第三个发展阶段——智能化阶段.

今日，对复杂系统进行智能诊断已成为智能技术研究的前沿课题和热点. 在专家系统研究已有较深厚基础的国家中，故障诊断专家系统已基本完成了研究和试验阶段，开始进入了广泛应用的阶段. 如 1985 年 Regenie 等人研制的飞行器控制系统监视器(EEFSM)和 1987 年 Malin 研制的汽车故障诊断系统(FIXER)以及美国宇航局 Langley 研究中心研制的飞行器故障诊断专家系统(Fault-finder)等都已达到了实际应用水平，并投入使用. 尤其在航空航天领域中，可靠性和维修性工程越来越受到重视，迫切需要

利用自动故障诊断和维修的专家系统来保障火箭、卫星和导弹试验以及基地飞行试验的安全.现已研制出一些智能诊断系统,像火箭发动机故障诊断专家系统(REFDES)、卫星控制系统地面实时故障诊断专家系统等等.自80年代中期以来,国内在智能诊断方面的研究已蓬勃展开,并取得了一定的研究成果.可以预料,现代机器智能诊断这一新兴学科的发展将充满生机,具有广泛的前景.

本书共分六章,内容包括:诊断知识处理,诊断征兆获取与诊断问题求解,传统故障诊断专家系统,神经网络故障诊断专家系统,模糊神经网络故障诊断专家系统,智能故障诊断技术的发展和展望.全书形成了智能诊断的理论框架,并将模糊逻辑、神经网络与知识工程融合起来,提出了智能诊断系统的原理及设计,开发了智能诊断系统的计算机软件,并结合作者的研究成果和国内外应用实例深入浅出地介绍了智能诊断领域内的新课题,这使得人们对智能诊断的本质问题及未来智能诊断系统是什么样的等问题给予了更多的思考.

作者试图在下述方面形成本书的特点:

首先,内容新.本书的内容体现了现代机器故障诊断学正朝着计算机辅助的故障模式识别、故障诊断决策支持系统及故障诊断专家系统的方向发展,系统总结了智能诊断的理论和方法,为复杂设备的多故障、多过程和突发性故障的自动监测诊断开辟了新的途径.

其次,重实用.本书在结构体系上将理论、方法介绍与具体计算机程序相结合,并落实到典型设备的智能诊断软件上.因此,有利于初次涉足于智能诊断领域的读者掌握书中的内容.本书提供的各种实用程序均经过多次检验,可运行在各类微机上,广大工程技术人员,通过实例学习,了解应用,上手快,有利于智能诊断技术的应用及推广.

应该指出,复杂系统的智能诊断涉及广泛的学科领域,许多问题尚有待于进一步研究和探索,对其进行完整和系统的研究是一项艰巨而又困难的工作,因此本书仅仅在某种程度上起到抛砖引

玉的作用.尤其希望有更多的科技工作者参加到它的研究和开发行列中来,以推动它的进一步发展.

本书的问世,不只是作者的努力结果,而且还凝结了许多人的成果.本书参考和引用了国内外有关学者的论著,从中受到莫大的启迪.作者在此向各位学者表示衷心的感谢.

本书获得广东省高校重点学科项目和广东省自然科学基金项目的资助,书中反映的研究成果就是在他们的有力支持下取得的.

中国科学院院士、华中理工大学校长杨叔子教授在百忙之中为本书作序,并提出了许多宝贵的意见,在此向他表示深深的谢忱!

鉴于作者学识水平所限,书中难免存在错误和不足,殷切希望读者不吝赐教.

吴今培

1996 年 11 月

目　录

第一章　诊断知识处理

1.1　概　　述

1.1.1　故障诊断的必要性

诊断学(diagnostics)一词源于希腊文，意指鉴别、确定．最早的诊断莫过于医疗诊断，随着人类文明的进步和科技的发展，人们在越来越多的领域里开展了诊断活动，比如企业诊断和环境诊断等．在工程技术领域中，自从机器问世以来，人们就非常关心它的健康——能否正常工作．对机械设备的运行状态进行诊断的技术至今已有很长的历史，可以说几乎是与机器的发明同时产生的．最初，机械设备较为简单，维修人员主要靠感觉器官、简单仪表和个人经验就能胜任故障的诊断和排除工作，我们把它叫做传统的诊断技术．

随着现代工业及科学技术的迅速发展，生产设备日趋大型化、高速化、自动化和智能化，传统的诊断技术已远远不能适应了．现代化工业生产，一旦因故障停机，损失巨大．近年来，因关键设备故障而引起的灾难性事故时有发生，例如：1984 年 12 月位于印度博帕尔市的美国碳化物公司农药厂，发生毒气泄漏事件，造成 2000 多人死亡，20 多万人受害，成为世界工业史上最大的恶性事故．1986 年 1 月 28 日，美国航天飞机“挑战者”号的空中爆炸事件，导致 7 名宇航员全部遇难，总计损失达 12 亿美元．1986 年 4 月 27 日，前苏联切尔诺贝利核电站的大量放射性元素外泄事件，2000 余人死亡，几万居民撤离，损失达 30 亿美元，并且污染波及周边各国．我国机械设备发生故障引起的损失也十分惊人，如 1985 年大同电厂和 1988 年秦岭电厂的 200MW 汽轮发电机组的严重断轴毁机事件；进口的大型万吨货轮航行中的主机突然损坏遇风暴沉

入海底等.这些严重的、或灾难性的事件不断发生,迫使人们在设备诊断方面进行了大量的研究,形成了机器设备、工程结构和工艺过程的故障诊断这一新兴的研究领域.

国内外许多资料表明,开展故障诊断的经济效益是明显的.据日本统计,在采用诊断技术后,事故率减少75%,维修费降低25%—50%,英国对2000个国营工厂的调查表明,采用诊断技术后每年节省维修费3亿英镑,用于诊断技术的费用仅为0.5亿英镑,净获利2.5亿英镑.据有关部门统计,我国每年用于设备维修的费用仅冶金部就达250亿元,如果将故障诊断这项技术推广,每年可减少事故50%—70%,节约维修费用10%—30%,效益相当可观.

1.1.2 故障诊断的基本概念

一般来说,诊断对象可以是一个比较复杂的大系统(甚至是巨系统),也可以是一个简单的元件或部件(子系统).严格地说,从不同的层次观察可对诊断对象有不同的划分.但无特别说明的话,诊断对象都被视为系统,其诊断也就看作系统诊断.

所谓系统的故障,是指系统的运行处于不正常状态(劣化状态),并可导致系统相应的功能失调,即导致系统相应的行为(输出)超过允许范围,使系统的功能低于规定的水平,这种劣化状态称为故障.

该定义具有以下特点:

1) 强调系统的输出(行为),从而有利于给出系统状态的测量途径.

2) 强调系统故障评判的多样性,即从系统的所有行为表现都可以进行评判.系统的输出是以多种方式表现:一些能直接表现系统的功能行为(称功能输出);一些则是实现系统功能时所附带产生的行为(称附加输出).如发动机发出功率,缸体发热都是发动机的输出,前者是其功能输出,后者则是附加输出.

3) 强调对故障认识的主观性,即系统的状态"超过允许范围"

是人参与的体现.

4）该定义也不排斥客观性，即“超过允许范围”是一个诊断标准问题，一旦标准确立，则具有客观性与公正性.

对故障分类目前还缺乏系统的方法，现在从不同角度给出以下几种分类方法：

1）按故障发生的时间历程分，有突发性故障和渐进性故障.突发性故障是发生故障前，不能提前测试与预测，这种故障表现出随机性；渐进性故障是由系统参数的逐步劣化产生的，这种故障能够在一定程度上早期预测，一般正常使用下在其有效寿命的后期才表现出来.

2）按故障存在的时间历程分，有间歇性故障和永久性故障.间歇故障是系统功能输出或附加输出在短时间内超过规定界限的现象，如发动机高温下工作时个别气缸由于供油系气阻而间歇不发火；永久性故障是系统功能输出或附加输出持续超过界限的现象，如发动机气缸磨损后引起异响等.

3）按故障的显现状况来分，有潜在故障和功能故障.潜在故障是系统功能输出并未超过允许范围，但其附加输出已有明显的表现；功能故障则是系统的功能输出超过规定范围，一般是子系统的功能降低，严重的情况是零部件的损坏.

4）按故障原因分，有内在故障和环境故障.内在故障是系统内部各部分结构关系不协调或结构劣化引起，环境故障是系统的输入异常引起.如由于设计、制造和装配以及零件变形或子系统异常等引起的故障是内在故障，而系统的环境故障是指其从环境中获得的能量、物质和信息异常，例如，发动机输入的燃料不足、空气稀薄、司机操纵失误等引起的发动机输出超界是环境故障.

故障的主要性质表现如下：

1）层次性：从系统论的观点看，可以认为系统是由“元素”按一定的规律聚合而成的.当然，系统的“元素”可以是子系统，子系统的“元素”还可以是更深层次的子系统，如此类推，直到元素是物理元件为止.显然，系统是有层次的.故障的产生对应于系统的不

同层次而表现出层次性.

2）时间性：系统故障的产生与表现常常与时间有关，以及由其运行的动态性所决定，如渐进性故障、间歇故障等.

3）相关性：复杂系统（如汽车发动机）是若干相互联系的子系统组成的整体，某些子系统的故障常常是由于与之相关子系统或下一级子系统故障传播所致，从而表现出相关性.

4）模糊性：系统运行状态中的模糊性，以及人们在状态监测和技术诊断中存在着许多模糊的概念及方法.

5）随机性：故障的发生常常与时间紧密相关的随机过程有关.

6）未确知性：它既不是由于故障描述的模糊性引起，也不是因随机性而产生，而是由于人为主观上因条件的限制，在系统故障已产生后，不能准确说明其发生的部位与原因，而它又确实已经存在，只是因条件不足我们不能完全感知.

7）相对性：系统故障与一定的条件和环境有关，不同条件和环境下的故障表现以及对其描述与划分存在不一致性，如不同的描述方法故障的程度就不同等.

何谓故障诊断？

故障诊断是指系统在一定工作环境下查明导致系统某种功能失调的原因或性质，判断劣化状态发生的部位或部件，以及预测状态劣化的发展趋势等. 犹如医疗诊断，是指医生借助各种手段对人体进行检查、化验，然后根据医学理论确定诊断对象是否患病和患有何种疾病的过程. 诊断就是由现象判断本质，由当前预测未来，由局部推测整体的过程. 在工程技术领域，也需要根据设备各种可测量的物理现象和技术参数的检测来推断设备是否正常运转，判断发生故障的原因和部件，预测潜在故障的发生等等. 借用了医疗方面的术语，将给机器的故障诊断的过程称为故障诊断.

故障诊断的基本思想一般可以这样表述：设被检测对象全部可能发生的状态（包括正常和故障状态）组成状态空间 S，它的可观测量特征的取值范围全体构成特征空间 Y，当系统处于某一状

态 s 时，系统具有确定的特征 y，即存在映射 g：

$$g:S\to Y$$

反之，一定的特征也对应确定的状态，即存在映射 f：

$$f:Y\to S$$

状态空间与特征空间的关系可用图 1.1 表示.

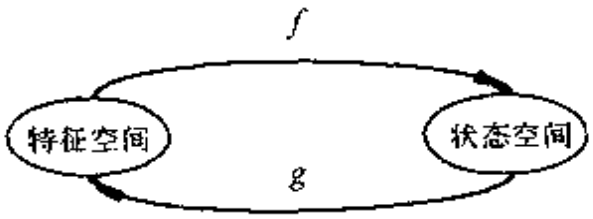


图 1.1　故障诊断表述

如果 f 和 g 是双射函数，即特征空间和状态空间存在一对一的单满射，则由特征向量可唯一确定系统的状态，反之亦然. 故障诊断的目的在于根据可测量的特征向量来判断系统处于何种状态，也就是找出映射 f.

若系统可能发生的状态是有限的，例如可能发生 n 种故障，这时把正常系统所处的状态称为 s_0，把存在不同故障的系统所处的不同状态称为 $s_1,s_2,\cdots,s_n$. 当系统处于状态 s_i 时，对应的可测量特征向量为 $Y_i=(y_{i1},\cdots,y_{im})$. 故障诊断是由特征向量 $y=(y_1,\cdots,y_m)$，求出它所对应的状态 s 的过程. 因为一般故障状态并非绝对清晰的，有一定模糊性. 因此，它所对应的特征值也在一定范围内变动，在这种情况下，故障诊断就成为按特征向量对被测系统进行分类的问题或去对特征向量进行状态的模式识别问题. 因此，故障诊断实质上是一类模式分类问题.

何谓故障诊断学?

故障诊断学是以可靠性理论、信息论、控制论和系统论为理论基础，以现代测试仪器和计算机为技术手段，结合各种诊断对象（系统、设备、机器、装置、工程结构、工艺过程等）的特殊规律而逐步形成的一门新兴学科. 它大体上由三大部分组成. 第一部分为故障诊断物理、化学过程的研究. 例如电气、机械部件失效的腐蚀、蠕

变、疲劳、氧化、绝缘击穿、断裂、磨损等理化原因的研究.第二部分为故障信息学的研究,它主要研究故障信号的采集、选择、处理与分析过程.例如通过传感器采集设备运行中的信号(如振动、转速),再经过时域和频域上的分析处理来识别和评价设备所处的状态或故障,上述过程均属于故障信息学范畴.第三部分为诊断逻辑与数学原理方面的研究,主要是通过逻辑方法、模型方法、推理方法及人工智能等方法,根据可观测的设备故障表征来确定下一步的检测部位,最终分析判断故障发生的部位和产生故障的原因.

1.1.3 故障诊断的过程

故障诊断的过程有三个主要步骤:第一步是检测设备状态的特征信号;第二步是从所检测到的特征信号中提取征兆;第三步是根据征兆和其它诊断信息来识别设备的状态,从而完成故障诊断.故障诊断的过程如图 1.2 所示.

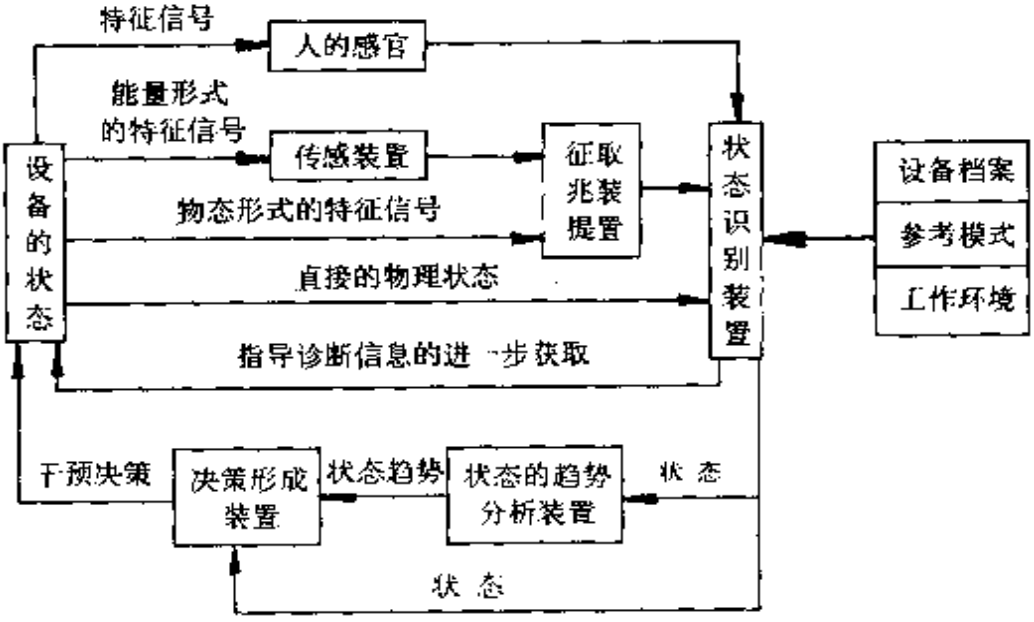


图 1.2 故障诊断过程

检测设备状态的特征信号,一般来说,它具有两种表现形式,一种是以能量方式表现出来的特征信号,如振动、噪声、温度、电压、电流、磁场、射线、弹性波等;另一种是以物态形式表现出来的

特征信号，如设备产生或排出的烟雾、油液等以及可直接观测到的锈蚀、裂纹等．检测能量方式所表现出的特征信号，如果不使用人的感官，则必须使用传感装置，因为检测这类信号是通过能量交换来完成的；而提取物态形式的特征信号一般不采用传感装置，只采用特定的收集装置或直接观测．

从所检测的特征信号中提取征兆．如输入征兆提取装置的特征信号是能量形式的，则可在时域中提取征兆，也可以在频域、幅值域或相位域中提取征兆．对于物态形式的特征信号，如油液、烟雾等，其征兆提取方法一般是通过特定的物理或化学方法，得到诸如铁谱、光谱、浓度、粘度以及化学成分等征兆．对于直接观测到的锈蚀、裂纹等信息，可以直接作为征兆来使用．

从征兆提取装置输出的征兆即可馈入状态识别装置来识别系统的状态，这是整个诊断过程的核心．一般来说，这一步是将实际上已存在的参考模式（标准模式）与现有的由征兆按不同方式组成的相应的待检模式进行对比，而决定待检模式应划为哪一类参考模式，即对系统的当前状态进行模式识别．在智能技术引入诊断领域之前，状态识别实际上是由领域专家来完成的．随着人工智能技术特别是专家系统技术在诊断领域的应用，产生了基于知识的诊断推理这一发展方向，它模拟领域专家来完成状态识别任务，这也是智能诊断技术同一般诊断技术之间最主要的差别．状态识别过程是一个由粗到精，由高层到低层直至找到满意的诊断解为止的逐层诊断过程．当然，对于整个诊断过程，自然还应形成最后的干预决策，并付诸实施．

综上所述，诊断过程包括三个主要步骤，即信号测取、征兆提取和状态识别．由以上诊断步骤，决定了设备诊断技术的主要内容为

1）采用合适的特征信号及相应的观测方式（包括合适的传感装置、人的感官），在设备合适的部位，测取设备有关状态的特征信号．

2）采用合适的征兆提取方法与装置，从特征信号中提取设备

有关状态的征兆.

3) 采用合适的状态识别方法与装置,依据征兆进行推理而识别出设备的有关状态.显然,有关状态包括正常的与不正常的有关状态,不正常状态时,即为故障诊断.

4) 采用合适的状态趋势分析方法与装置,依据征兆与状态进行推理而识别出有关状态的发展趋势,这里包括故障的早期诊断与预测.

5) 采用合适的决策形成方法与装置,从有关状态及其趋势形成正确的干预决策:或者深入系统的下一层次,继续诊断;或者已达指定的系统层次,作出调整、控制、自诊治、维修等某类决策.

在设备诊断中,毫无疑问,应要求在每一步骤,花费尽可能少,有关状态的信息获取尽可能多,结果尽可能好.然而,各诊断步骤是彼此相互联系与相互影响的,各局部最优并不能保证全局最优,因此,还必须从全局出发,全面考虑整个诊断过程,从宏观上制定尽可能好的诊断策略.

1.1.4 故障诊断的智能化

近年来,由于计算机技术、现代测试技术和信号处理技术的迅速发展,设备故障诊断技术取得了很大的进展.人们已开发和研究了一些较成熟的诊断技术及理论方法,如铁谱分析、声发射、红外测温、油液分析和各种无损监测等诊断技术,以及信号处理、模式识别、模糊推理等理论方法,人们可对在多种工作环境条件及运行状态下的机器或工程系统的许多故障模式进行监测、识别、诊断和排除.然而,在工程实际中存在着大量的多故障、多过程,突发性故障及需要对庞大机器或工程系统进行的监测和诊断,上述技术手段和理论方法往往显示出较大的局限性,主要表现在:

1) 不能有效地利用专家的知识和经验;

2) 缺乏推理能力,只能向前推理,不能像专家一样既能向前推理,又能向后推理;

3) 不具备学习机制;

4）对测试诊断结果缺乏解释，测试诊断程序的修改和维护性差.

随着人工智能技术的迅速发展，特别是知识工程、专家系统和人工神经网络在诊断领域中的进一步应用，迫使人们对智能诊断问题进行更加深入与系统的研究. 所谓诊断系统的智能就是它可有效地获取、传递、处理、再生和利用诊断信息，从而具有对给定环境下的诊断对象进行成功状态识别和状态预测的能力. 但是诊断系统的智能并不意味着完全代替人的智力活动，将人排斥于诊断系统之外. 实践证明，任何人工智能系统的研究，都不能完全摆脱人脑对系统的参与，只能是“人帮机”和“机帮人”. 人是智能系统的重要组成部分. 由此，我们可以这样来定义智能诊断系统：它是由人（尤其是领域专家）、当代模拟脑功能的硬件及其必要的外部设备、物理器件以及支持这些硬件的软件所组成的系统. 该系统以对诊断对象进行状态识别与状态预测为目的. 显然，该定义下的智能诊断系统具有以下特点：

1）它是一个开放的系统，其智能水平处于一个动态变化之中，且具备自我提高的潜能.

2）它是由计算机硬件与软件组成的系统，但又不同于常规的计算机程序系统，不具有确定的算法和程序途径. 智能诊断系统是根据诊断过程的需要搜索和利用领域专家的知识及经验来达到诊断的目的.

3）它既是一个人工智能系统，离不开模拟人脑功能的硬件设备及软件，另一方面又不排斥人的作用，同时对硬件并不仅仅限制为今天的冯·诺依曼式传统计算机. 这种计算机的固有缺陷是局域式信息储存、串行程序式的符号处理等. 近年来人工神经网络的兴起，发展了基于神经网络的智能诊断系统，将利用神经网络的学习功能、联想记忆功能、分布式并行信息处理功能，解决诊断系统的知识表示、获得和并行推理等问题.

智能诊断系统中的知识处理包括了三个主要步骤：知识获取、知识存储及推理. 根据智能诊断系统在知识处理途径上的不同，智

能诊断方法可以分为两大类：一类是基于符号推理的智能诊断方法，如传统人工智能；一类是基于数值计算的智能诊断方法，如人工神经网络，它是当今智能诊断的一个重要研究领域.

在基于符号推理的知识处理系统中，知识是按一定的规则用特定的描述符加以表示、存储和处理的. 知识的获取过程就是对事件型知识或从领域专家那里获得的功能型知识加以描述的过程. 按一定的规则存储这些知识，所得的知识体系就称之为知识库. 得到这样的知识库之后，知识处理系统就能根据输入数据及一定的推理机制和推理策略进行逻辑推理，最后，得出所要求或希望的结果. 在这里，知识获取程序、知识库及推理机之间的联系有时往往不是很密切，即它们各自具有相对的独立性. 同时，由于知识获取的“瓶颈”和逻辑推理的“组合爆炸”等问题，已使基于符号推理的智能诊断受到了一定程度的限制.

在基于数值计算的知识处理系统中，知识是通过系统的权系数矩阵来加以存储的，即知识是表示在系统的权系数矩阵之中的. 知识的获取过程就是按一定的学习规则通过学习逐步改变其权系数矩阵的过程. 神经网络能进行联想和记忆推理，因而具有很大的容错性. 对于不精确的、矛盾的和错误的数据，它都能进行推理，并能得出好的结果. 在神经网络的知识处理系统中，知识获取、知识存储及推理之间的联系非常密切，即具有很大的交融性. 同时，神经网络的知识处理系统不存在知识获取的“瓶颈”和推理的“组合爆炸”等问题. 因而，使其应用范围更加宽广.

表 1.1 列出了智能诊断系统中两种知识处理方式的比较.

表 1.1　两种知识处理方式的比较

处理方式	符号推理	数值计算
知识表示	规则式表示体系	权系数矩阵隐式内部表示形式
知识获取	知识库形式 存在知识获取“瓶颈”问题	自适应学习方法 便于克服知识获取“瓶颈”问题
推　理	搜索求解和无穷递归 存在“组合爆炸”等问题	联想、记忆推理 便于克服“组合爆炸”等问题

机器故障智能诊断的目的在于保证机器在允许的工作条件下和规定的时间内，完成预期的功能.对此智能诊断系统应当有效地获取、传递、处理、再生和利用机器诊断信息，从而有效地识别机器工作状态、找出机器可能存在的故障或导致这些故障的原因，甚至还要预测机器状态的发展趋势.智能诊断系统与诊断对象的关系如图 1.3 所示.

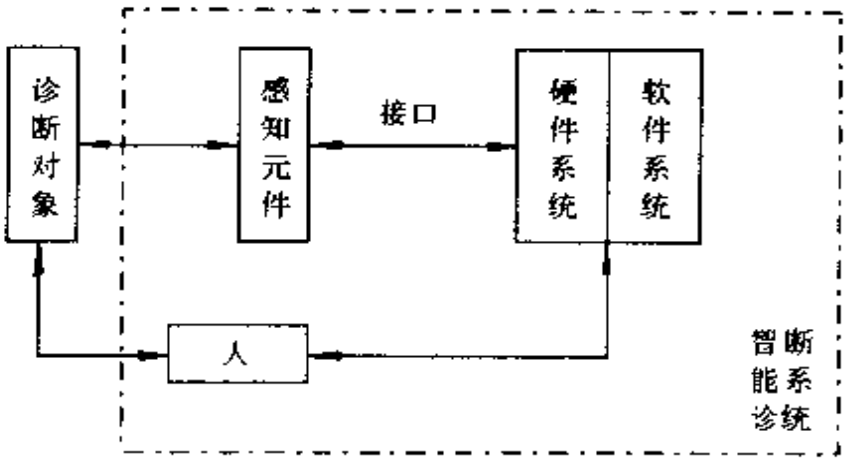


图 1.3 智能诊断系统与诊断对象的关系

智能诊断技术是当今世界发达国家的研究热点之一.在专家系统已有较深厚基础的国家中，机械、电子设备的故障诊断专家系统已基本完成了研究和试验的阶段，开始进入广泛应用的阶段.如 1985 年 Regenie 等人研制的飞行器控制系统监视器(EEFSM)和 1987 年 Malin 研制的汽车故障诊断系统(FIXER)以及美国宇航局 Langley 研究中心研制的飞行器故障诊断专家系统(Fault-finder)等都已达到实际应用水平，并投入使用.尤其在航空航天领域中，可靠性和维修性工程越来越受到重视，迫切需要利用自动故障诊断和维修的专家系统来保障火箭、卫星和导弹试验以及基地飞行试验的安全.现已研制出一些智能诊断系统，像火箭发动机专家系统(REFDES)、航天器故障诊断试验专家系统(ATFDES)、卫星控制系统地面实时故障诊断专家系统等等.

自 80 年代以来，我国不少教学科研院所先后开展了故障诊断专家系统的研究工作，并取得了一定的研究成果，有一些系统已投

人了实际运行.如华中理工大学设计的基于知识的汽车发动机的诊断专家系统,运用了深层和浅层知识,进行功能、症状和特性的三种分析法,较成功地对一复杂系统进行故障诊断.解放军军械工程学院在军械装备故障诊断专家系统的设计中,采用规则、语义网络、框架等多种知识表示形式,在浅层推理失败后,可进行基于军械装备物理功能模型的深层推理.以上两个例子已说明我国的故障诊断专家系统开始走向第二代专家系统.比起基于经验规则推理的第一代专家系统来讲,第二代专家系统的知识集更完备.

故障智能诊断理论和方法的研究虽然已取得较大的成就,但在实时性、机器学习等方面仍面临一些问题,有待于进一步解决.

在实际应用中,许多机械、电子设备要求诊断系统对故障能进行实时检测、实时诊断,不允许得到解的时间很长.当前,国外已出现一些实时的故障诊断专家系统."实时"依靠的技术主要有:

1)依靠智能软件的并行;

2)依靠分级进程推理;

3)依靠信息的高度精炼,即机器在相当长的一段时间内要自行运行并不断排除垃圾.

所谓机器学习,其实质就是知识获取.随着专家系统的发展,知识获取已成为建造专家系统的"瓶颈".故障诊断专家系在获取知识的过程中,最初是将专家提供的知识存人知识库中,这是一种机械记忆的学习过程.当解题中遇到无法解决的问题时,可通过提问,由外界提供信息,形成新知识并存人知识库中,继续推理.还可通过示例学习,要求系统能够从特定的示例中归纳出一般性的规则.机器学习则要求系统能在实际工作中不断地总结、归纳成功和失败的经验教训,对知识库中的知识自动进行调整和修改,以丰富、完善系统的知识.机器学习从内在行为看,是从未知到已知的过程,是知识增加的过程;从外在表现看,是系统的某些适应性改变,使得系统能完成原来不能完成的任务或把原来的任务做得更好.但由于学习问题的多样性与复杂性以及目前计算机工作原理带来的制约和限制,机器学习问题至今尚无突破性进展.

1.2 从传统诊断到智能诊断

目前，用于系统故障诊断的方法可分为两大类：一类是完全基于检测数据处理的诊断方法，如用得较多的对比诊断法、函数诊断法、振动诊断法、模型识别法、统计诊断法以及模糊诊断法等，它们是通过对故障检测信号的处理而较早地发现故障，以至预报故障；另一类则是主要基于专家经验及知识处理的专家系统诊断方法，它是模仿人类专家在进行故障诊断时，首先观察机器的症状，然后依所观察到的症状，利用自己所具有的知识来推断故障的原因. 由于专家系统方法是最近几年才开始用于系统故障诊断的，所以，我们称第一类完全基于数据处理的诊断方法为传统诊断方法，而基于知识处理的诊断方法为智能诊断方法.

传统诊断方法，尽管可以通过检测信号的处理，实现机器工况监视与故障诊断，但当诊断对象变得庞大而复杂时，为了能把故障比较细致的区分出来，一方面需要增加检测手段，另一方面需大大增加计算量，从而使得诊断的时间延长. 当调试完毕后，用传统诊断方法编制的软件系统的功能就确定下来了，不易更改，只局限于某一具体系统的诊断，很难应用于不同的诊断对象，因而推广较难.

人类专家在进行故障诊断时是怎么做的呢？领域专家往往可以直接凭借系统发生故障时，用视觉、听觉、嗅觉或触觉得到的一些难以由数据描述的事实以及专家对系统发生故障的历史和系统的结构等作出判断，从而可能很快地找到故障源，这种专家经验的应用，对于复杂大系统的故障诊断尤其有效，在多数情况下，可能做到比用传统诊断方法判断故障要快得多. 而完全基于检测数据处理的传统诊断方法的共同局限性恰恰在于根本没有利用人类领域专家的丰富经验和知识在故障诊断中的快速而有效的作用.

对智能诊断方法研究的目的之一就是试图以计算机模拟人类专家对复杂系统进行故障诊断. 做到既能充分发挥领域专家在诊断中根据各种感觉得到的事实及专家经验进行快速推理，又能很

方便地推广应用于各种不同的诊断对象，这正是智能故障诊断方法不同于传统诊断方法的显著优点.

为了对智能诊断的概念有深入的理解，先考察一下人的智力活动过程是很有必要的. 图 1.4 是人智力活动过程的一般模型.

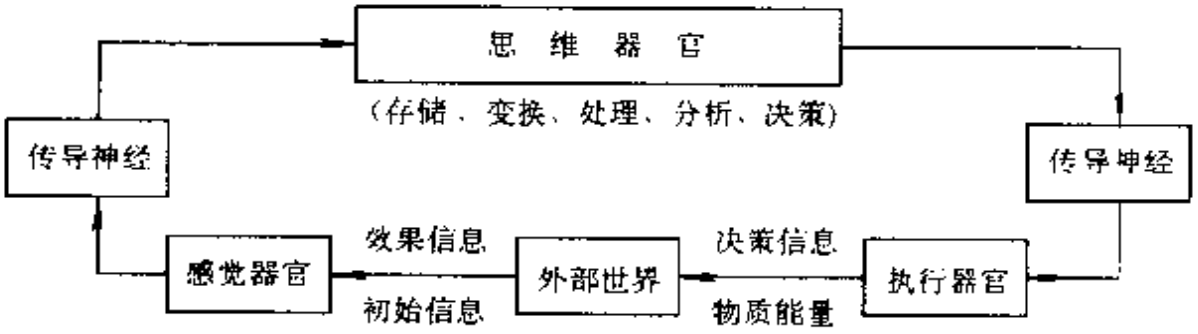


图 1.4 人的智力活动过程模型

人为了认识世界，首先通过自己的感觉器官来感受外部世界环境中各种事物的运动状态和方式，然后，通过导入神经系统把感受到的外部世界信息传递给思维器官(大脑)，对这些信息进行存储、变换、处理、分析和判断，去除各种干扰，提取有用的信息. 在此基础上形成初步的判断，获得相应的认识，并在思维器官中形成决策信息，发出指令，再通过导出神经系统来指挥执行器官完成相应的动作，作用于外部世界. 这是人的活动的一个最基本的循环. 实际上，人的活动是一个连续的周而复始的过程. 在第一个最基本的循环之后，为了检验“认识”和“动作”的效果，感觉器官还要把指令信息作用于外部世界的效果，作为一种效果信息反馈给思维器官，根据这个反馈信息，思维器官再作出补充决策，调整原来的指令信息，修正执行器官的行动策略，并进而有效控制执行器官对外部世界的后续作用. 这样不断循环、修正，直到人与外部世界之间达成某种相互协调的条件为止.

另一方面，我们还可以将人的活动过程模型从信息角度加以描述，如图 1.5 所示. 图 1.4 模型中的感觉器官的功能主要是为了从外部世界中获取信息，感知外部世界事物的运动状态及其变化方式. 这些信息是经过神经系统送到思维器官，思维器官通过分析

计算产生出有利于自身发展的策略信息.可以认为思维器官的主要功能是处理信息和再生信息.执行器官的功能则主要是按照策略信息,对外部世界的事物产生相应的反作用.改变或维持该事物运动状态和状态的变化方式,即执行器官的主要功能就是使策略信息发生效用.

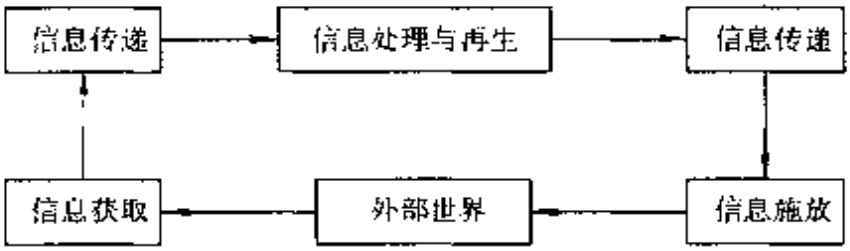


图 1.5 人的智能活动的信息模型

从信息角度出发,智能的关键是获取、处理、再生和利用信息的能力.对于同样的环境和目的,谁具有更强的获取、传递、处理、再生和利用信息的能力,谁就能更成功地达到目的,就可以认为谁具有更高的智能水平.由此,对于一个故障诊断系统的智能可以作如下定义.

定义 1 诊断系统的智能就是指可有效地获取、传递、处理、再生和利用诊断信息,从而具有对给定环境下的诊断对象进行成功状态识别和状态预测的能力.

智能的核心是思维,思维的器官在大脑.在人类漫长的进化过程中,已使人脑成为无与伦比的智慧载体.因此,我们在研究智能诊断系统时就不应摆脱人脑对诊断过程的参与,人是智能诊断系统的重要组成部分.由此,对于智能诊断系统可以作如下定义.

定义 2 智能诊断系统是由人(尤其是领域专家)、当代模拟脑功能的硬件及其必要的外部设备、物理器件,以及支持这些硬件的软件所组成的系统.该系统以对诊断对象进行状态识别与状态预测为目的.

智能诊断的优越性在于它实现了人-机联合诊断功能,它综合多个专家的最佳经验,其功能水平可以达到超过专家,至少具有专

家的水平，实现多故障、多过程、突发性故障的快速分析诊断.如果说传统诊断系统是由计算机硬件与软件组成的话，那么智能诊断系统既离不开模拟人脑功能的计算机硬件及软件，又不排斥人的作用，而是集传统诊断方法优点和专家经验于一体，实现了人-机联合诊断功能.

必须指出，对于一个领域、一类设备非常有效的诊断方法，对于另一个领域、另一类设备则可能完全不适用，这就说明不同诊断领域、不同类型设备所需的诊断理论和方法具有明显的特殊性，这显然是由于对于不同的诊断领域，在智能诊断问题的描述、诊断知识的使用与组织、诊断知识的学习、诊断信息的类型与获取等诸方面的不同因素所引起的.因此，通向智能诊断的途径不是唯一的.目前，智能诊断方法主要是依据知识处理方法来区分.知识处理方法可归结为两种：以符号处理为核心的方法和以网络联接为主的联接机制方法.

以符号处理为核心的方法又称为自上而下的方法，它是当前智能诊断中普遍采用的传统知识处理方法.其主要特征有：

1）适合用于模拟人的逻辑思维过程，解决需要进行逻辑推理的复杂问题.

2）知识可用显式的符号表示，在已知基本规则的情况下，就无需输入大量的细节知识.

3）便于模块化，当个别事实发生变化时易于修改.

4）能与传统的符号数据库进行接口.

5）能解释自己的推理过程，并能解释结论是如何获得的.

但是，人们并非仅仅依靠逻辑推理来解决问题，有时非逻辑推理在解决问题中也起着重要的作用，甚至是决定性的作用.人的感知过程主要是形象思维，这是单靠逻辑思维做不到的，因而无法用符号方法进行模拟.人们的某些只能意会而不可言传的经验性知识，也不能用任何符号系统表示.另外，用符号表示概念时，其有效性在很大程度上取决于符号表示的正确性，当把有关信息转换成推理机构能进行处理的符号时，将会丢失一些重要的信息.它对带

噪声的信息以及不完整的信息也难以进行处理.而这些正是联接机制方法的长处.

以网络联接为主的联接机制方法属于非符号处理范畴,又称为自下而上的方法.这种方法把注意力转向人脑的信息处理模式.近年来正在兴起的人工神经网络的研究就是为了探明人脑的信息处理方式,建立脑的模型,进一步阐明并行信息处理的基本原理,并从应用角度寻求其工程实现方法.利用人工神经网络的学习能力、联想记忆功能、分布式并行信息处理功能,解决智能诊断中的知识表示、知识获取和并行推理等问题,因此上述方法又称之为神经网络知识处理方法.

上面讨论的两种方法各有所长,也各有所短.符号处理方法善于模拟人的逻辑思维,但其机制固定于回收知识与推理,有一定局限性;网络联接机制方法适合于模拟人的形象思维,但由于固定的体系结构与组成方案,使得所构成的系统达不到开发多种多样知识要求.在这种情况下,如果能把两种方法有机地结合起来,构成一个混合系统,使它们在知识获取、知识表示、推理及解释等方面进行合作,从而有效地模拟人的逻辑思维与形象思维,那将是当今智能科学发展的历史选择.

在混合系统中,关键的问题是如何把两种方法有机地结合起来.目前采用的方法一般有两种:一种是结合,即两者都分别保持原来的结构,但密切合作,任何一方都把自己不能解决的问题转化给另一方;另一种方法是统一,即把两者自然地统一在一个系统中,既有逻辑思维功能,又有形象思维功能.为了建立混合系统,国内外学者都开展了相应的研究.

1.3 从数据处理、建模处理到知识处理

故障诊断技术发展至今已经历了三个阶段.第一阶段,诊断结果在很大程度上取决于领域专家的感官和专业经验,对诊断信息只作简单的数据处理.第二阶段是以传感器技术和动态测试技术

为手段，以信号处理和建模处理为基础的现代诊断技术，在工程中已得到了广泛的应用. 近年来，为了满足复杂系统的诊断要求，随着计算机及人工智能的发展，诊断技术进入以知识处理为核心，信号处理、建模处理与知识处理相融合的第三发展阶段——智能诊断技术阶段.

诊断数据处理(或信号处理)的主要内容是统计分析、相关分析、频谱分析、小波分析和模态分析等，其理论基础是数理统计与随机过程. 诊断建模处理的主要内容是参数估计、系统辨识、模式识别等，其理论基础是系统论、信息论和控制论. 诊断知识处理的主要内容是知识表示、知识获取和知识运用，其学科领域是人工智能和知识工程.

基于信号处理和建模处理的现代设备故障诊断技术，经过近30年的发展与应用，虽然取得了显著的社会与经济效益，但进一步的理论研究与应用结果表明，它本身存在着以下几个方面的局限性：

1) 各种信息检测手段和诊断方法都未将诊断对象看成是一个有机的整体，大多是利用诊断对象所表现出的特定信号(特征信号)来诊断特定类型的故障，未能有效地考虑多故障同时发生和各种故障之间可能存在的相互联系及影响.

2) 这种设备故障诊断技术只是种类繁多的检测手段和多种具体的诊断方法在某种程度上的“堆积”，缺乏统一的概念体系和系统化的理论.

3) 利用信号处理和建模处理，仅仅在一定程度上弥补了人类在数值处理上的不足. 然而大量的理论研究与应用结果表明，为了进一步提高诊断效率与水平，几乎在每个主要环节上都需要领域专家的知识处理问题的方法，尤其是辩证思维和符号处理能力.

4) 基于信号处理和建模处理构造的诊断系统专用性非常强. 一旦完成，它们的诊断能力在很大程度上也就确定了，其功能难以扩充或修改，并且人-机接口“柔性”很差.

近十几年来，人工智能特别是专家系统、知识工程发展迅速，

设备诊断技术正向着智能化阶段迈进.在这一阶段,领域专家的知识将得到充分的重视,诊断问题的研究将致力于模拟专家的推理过程、控制和运用各种诊断知识的能力.目前基于信号处理和建模处理的设备诊断技术正发展为基于知识处理的设备诊断技术,在知识层次上,实现辩证逻辑与数理逻辑的集成、符号处理与数值处理的统一、推理过程与算法过程的统一、知识库与数据库的交互等.

什么是知识处理?

本质上说知识处理也是一类信息处理,主要是符号的处理,包括符号表示、符号推理和搜索.虽然这种符号处理可以看成传统数据处理的延伸和发展,但是,符号的内涵不再局限于数据计算和数据处理中的数据和一般的信息,而主要表示人类推理所需要的各类知识.除了一些通用知识,例如数学、物理学、逻辑学等知识之外,更需要应用大量与所解问题领域密切相关的知识,即所谓领域知识.

什么是领域知识?

它是专家在长期的领域研究和处理各种领域问题的过程中,实践经验的概括和总结,它来源于专家的实践又指导着专家的实践.为了把领域知识和经验从专家的头脑中和书本中抽取出来,因而研究各种获取知识的方法和途径成了知识处理中第一个需要解决的问题.然而,知识是一种抽象的东西,要把它告诉计算机或者在其间进行传递,还必须把它们以某种形式逻辑地表现出来,并最终编码到计算机中去.这就是知识处理中要研究的"知识的表示"问题.正如记数法是数据处理的基础一样,知识的表示也是知识处理的基础.因为只当有了知识的表示以后才谈得上对它进行处理.而且不同的知识需要用不同的形式和方法来表示.它既应能表示事物间结构关系的静态知识,又应能表示如何对事物进行各种处理的动态知识,它既要能表示各种各样客观存在着的事实,又要能表示各种客观规律和处理规则;它既要能表示各种精确的、确定的和完全的知识,还应能表示更加复杂的、模糊的、不确定的和不完全的知识,因此一个问题能否有合适的知识表示往往成为知识处

理成败的关键.当然,获取知识和表示知识的最终目的还是为了运用知识来解决各种实际问题.如果只把获取的知识存放在机器中,而不加以运用,那么这仅仅是一堆死知识,它既不会再生知识,也不会产生任何效益.为了使已有的知识产生各种效益,使它对外部世界产生影响和作用,必须研究如何运用知识的各种技术和方法问题.综上所述"知识的获取"、"知识的表示"以及"知识的运用"也就成为知识处理学的三大要素或主要研究内容.

运用符号表示的知识或经验规则,而不是运用算法或数据处理方法来执行任务的任何系统,都可以叫做基于知识的系统,它具有符号推理、联想、学习和解释的能力,能够帮助人们进行判断、决策,开拓未知的领域和获取新的知识.如果说建模处理主要用于那些能够完全用数学精确描述的系统,那么知识处理主要用于那些没有精确数学模型或很难建立精确数学模型的复杂系统.

建模处理采用强方法来求解实际问题.这里所谓的强方法是指:

1) 把问题转化为数学模型;

2) 模型用精确的算法来描述;

3) 算法映射成计算机语言;

4) 计算机按算法指定的路径运行并得到结果.

强方法求解保证了在精确性科学计算与确定性过程控制应用中取得了巨大成功.

知识处理是采用弱方法进行搜索求解,求解问题时既不存在可资使用的精确模型,更没有现成的算法.这里所谓的弱方法是指:

1) 它对给定信息的要求比较弱,即问题的已知信息可能是不精确的、不完整的或模糊的;

2) 使用的知识本身属经验的、不严格的或人类尚未完全掌握的;

3) 求解问题需要经过反复的试探或搜索,求解的过程就是搜索的过程.搜索包括对知识块的匹配、选择和解释,匹配和解释的

结果往往引起再搜索.如此往复,直到达到目的,作出决策.

1.4 诊断知识的获取

1.4.1 诊断知识的分类

在诊断专家系统中,诊断知识来源于两个方面:1)自然语言文献,包括专业书籍、期刊、产品出厂文件、设计施工总结、安装调试记录以及设备运行历史资料等;2)领域专家的经验,包括领域专家在问题求解过程中所利用的结构知识、因果知识、行为知识等.

为完成一个实际的诊断任务所使用的有关具体诊断对象的知识,一般可分为以下几种:

(1) 结构与功能知识

表示诊断对象的结构和性能的描述知识,这种知识又称为深知识(deep knowledge),主要包括设备的结构层次、功能层次和相互间的输入、输出行为关系以及设备本身的工作原理等.

(2) 专家启发式经验知识

表示故障、征兆和原因等直接相联系的专家启发式经验知识,这种知识通常称为浅知识(shallow knowledge).它是领域专家的经验总结,常以规则的形式存在于专家的头脑而被专家非常灵活地应用.由于对这种知识常常缺乏本质性的认识,在很多情况下,即使专家本人也难以清楚地表达出来.因此,这种知识很难获取.经验知识作为诊断知识的一个重要组成部分,虽然缺乏充分的理论依据,但在解决复杂的实际问题时往往十分有效,开发故障诊断知识库的一个重要任务就是挖掘诊断领域专家的这种知识.

(3) 工况状态知识

表示诊断对象在某一工况下正常工作时应具备的性能参数指标,如发动机在额定转速下的功率、扭矩、油耗、怠速稳定转速等.这类知识反映当前发动机的整体性能.

(4) 信号特征知识

在复杂设备的诊断中,由于要求进行不解体在线诊断,许多参

数是无法检测的(如汽缸内压力、气门间隙等),因此,大量应用的是设备运行过程中产生的外部直接可观测的信号(如振动、噪声、温度等),通过对这些特征参数信号的分析能比较准确地诊断设备故障.

(5) 环境知识

表示与诊断对象有关的外围知识.如设备使用时间、大修间隔与大修总次数、工作环境与故障频率及故障类型分布等.

(6) 控制知识

表示对领域知识的运用起指导作用的知识.如引导规则的选择、控制推理路径;检查推理是否正常结束以及推理结果是否能够确诊故障等.

以上是根据故障诊断领域知识的特点来进行分类的,但从不同角度进行划分,可得到不同的分类方法.比如,若就知识的确定性来划分,知识可分为:确定性知识和不确定性知识.前者是指可指出其"真"或"假"的知识;后者是泛指不完全、不精确及模糊的知识.

若就知识的结构及表现形式来划分,知识可分为:逻辑型知识和形象性知识.逻辑型知识是反映人类逻辑思维过程的知识,例如人类的经验性知识等.在下面将要讨论的知识表示方法中,逻辑表示法、产生式表示法等就是用来表示这一类知识的.人类的思维过程除了逻辑思维方式之外,还有一种称为形象思维的方式,凡是通过事物的形象建立起来的知识称为形象性知识.目前,人们正在研究用神经网络来表示这类知识.

若撇开知识涉及领域的具体特点,从抽象的、整体的观点来划分,知识可分为:常识性知识、领域知识和元知识.常识性知识是指问题领域的事实、定理、方法和操作等常识性知识及原理性知识;领域知识是面向某个具体领域的专业性知识,例如领域专家的启发性经验知识;元知识是指如何运用以上两种知识的知识.元知识又可具体地划分为两类:一类是关于我们所知道些什么知识的元知识,这些元知识刻划了领域知识的内容和结构的一般特征,如知

识的产生背景、范围、可信程度等等；另一类是关于如何运用我们所知道的知识的元知识，例如问题求解中的推理策略、搜索策略等等.

一般来说，诊断知识表现出以下几种性质：

1）层次性：它表现为诊断知识在智能诊断系统中所起的作用不同，和对诊断对象的各子系统故障与征兆对应关系描述精确程度的不同，即知识的深与浅的不同.

2）模糊性：系统内部征兆与故障对应关系描述的不精确性.

3）相关性：来自于系统本身各子系统之间及其故障之间的相互联系.

4）多样性：从不同角度可以给出多种诊断知识，从而使诊断知识表现出多样性.

5）冗余性：它是由诊断知识的相关性与多样性决定的，从不同的角度给出的诊断知识常常存在包容.

6）动态性：它是由智能诊断系统对诊断知识的更新决定的.

1.4.2 诊断知识的获取方式

拥有知识是智能系统或专家系统有别于其它计算机软件系统的重要标志，而知识的质量和数量又是决定智能系统性能的关键因素.如何将专业领域的大量概念、事实、关系和方法，包括专家处理问题的各种启发性知识，从专家头脑中或其它知识源（如文献资料）中提取出来，并按照一种合适的知识表示形式将它们转移到计算机中去，就成为专家系统开发研究的一个重要课题.知识从外部知识源（人类专家、书籍文献等）到计算机内部的转换过程通常称为知识获取.

目前，对知识的获取有三种类型：自动型知识获取、非自动型知识获取和基于神经网络的知识获取.

1. 非自动知识获取

就目前的研究水平而言，非自动知识获取仍是成功的传统专

家系统所采用的，其工作方式如图 1.6 所示.

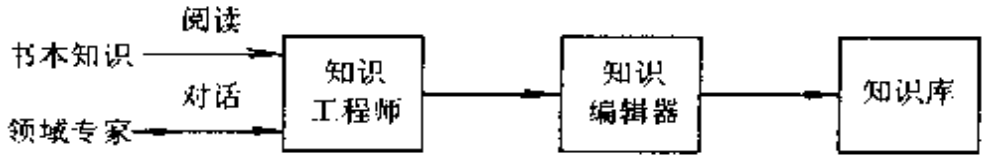


图 1.6 非自动知识获取

从外部知识源获取知识需要一个中间媒介，这个角色是由被称之为知识工程师的、经过专门训练的人来担任.知识工程师通过大量查阅文献、同领域专家反复交谈等，深入地分析领域专家处理问题的思想方法和思维特点，将从领域专家及有关文献中获得的专家系统所需要的知识，用合适的知识表示模式或语言表示出来，交给知识编辑器进行编辑输入.知识编辑器是一种用于知识输入的软件工具，它把知识工程师用某种语言表示的知识转换成计算机可接受的内部形式，并输入到知识库中.

2. 全自动知识获取

近年来，研究者们试图建立一种不需要知识工程师介入的专门人机交互系统来完成知识工程师的工作，这是未来专家系统知识获取的发展方向.一个理想的人机交互系统应具备以下功能：1)自动地将领域专家知识转换成为专家系统推理机可执行的知识代码；2)自动地将可执行的知识代码按一定的规律组织起来，支持推理且便于维护.要实现全自动的知识获取，一般要解决机器感知、机器识别和机器学习的问题.图 1.7 是一个全自动知识获取模型框图.

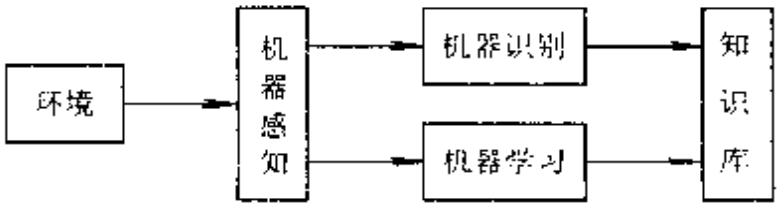


图 1.7 全自动知识获取模型

3. 基于神经网络的知识获取

诊断专家在进行机器故障诊断时，首先是利用机器所表现的症状(或征兆)，按自己以往所积累的诊断经验判断出机器故障的原因及类型.诊断的关键是领域专家的经验知识，这种知识是领域专家在其诊断生涯中从大量有意义的故障实例中抽象出来的，有关故障、征兆和原因之间相互关系的知识.也就是说，领域专家每当遇到一个新的实例，他首先是由相似性而联想到过去的某一实例，并与之相比较，然后作结论.这是一种按相似进行分类的方法.然而传统诊断专家系统广泛采用的是基于规则表示领域专家知识的方法，只能对分类作出明确表示，很难表示出故障实例与实例相对比所表现出来的相似性.另外，知识工程师提取规则不仅工作量大、周期长、效率低，而且对某些经验知识也很难用明确的规则表达出来.因此，知识获取就成为传统专家系统研制中的瓶颈问题.而基于神经网络的知识获取并不需知识工程师从领域专家的经验中提取规则，它只是对领域专家提供的大量故障实例进行学习，自动从领域专家的实例中提取知识，知识也是隐含地分布存储在神经网络中，并不像规则那样显式地表示出来，这种知识的获取方式是自动的，它在一定程度上缓解或克服了传统专家系统研制中存在的知识获取瓶颈问题.

基于大脑神经系统结构和功能模拟基础上的神经网络，可以通过对故障实例的不断学习而提高神经网络中所存储知识的数量和质量，特别是它可以提取类似实例之间的相似性和不同类别实例之间的差异，并体现在神经网络中神经元之间连接权值调整过程中.另外，神经网络的求解能力也较强，当环境信息不十分完全时，它仍然可以通过计算而得出一个比较令人满意的解答.因而如果我们用神经网络方法来构造故障诊断专家系统不仅可以在一定程度上克服知识获取的瓶颈问题，而且也将提高系统的强壮性，为诊断专家系统的研制开辟一条新途径.

可以认为，神经网络获取知识至少有以下两条途径：

1）直接从数值化的实例学习.根据领域专家给出的一些诊断实例偶对(如故障征兆与原因等),通过神经网络学习算法自动从这些数值化的偶对所表示的诊断经验中获取知识,并将这些知识分布存储在神经网络中.

2）将传统专家系统已获取的知识特例化为神经网络的分布式存储,且由神经网络完成并行推理.

神经网络可以采用以下策略来获取前面介绍的几类诊断知识:

1）结构与功能知识的获取:采用领域专家人工方法归纳给出,交由神经网络学习.

2）工况状态知识获取:此类知识也主要是由人工给出.它的获得比较容易,也比较单一化.对于此类知识的描述与定义,可依赖于系统程序化的模块完成.

3）环境知识获取:采用提供样本由系统自动学习方式获取.

4）信号特征知识获取:以样本学习为主要知识获取途径.

5）专家启发式经验知识获取:专家提供样本,系统格式化后由神经网络学习获得.

6）诊断控制知识获取:领域专家定义,通过人工给出.

1.5 诊断知识的表示

众所周知,计算机的计算,历来以数据作为处理对象,但只有当数据用二进制来表示时才得以在计算机中进行存储和运算.任何需要进行交流、处理的对象都需要用适当的形式表示出来才能被应用,对于知识当然也是这样.人工智能研究的目的是要建立一个能模拟人类智能行为的系统.为达到这个目的,就必须研究人类智能行为在计算机上的表示形式,只有这样才能把知识存储到计算机中去,供求解实际问题使用.所谓知识的表示就是一种描述,一种计算机可接受的对人类智能行为的描述.

当前知识表示的方式多种多样,但由于对人类的知识结构及机制尚不完全清楚,因此关于知识表示的理论及规范尚未系统建立起来.

一般来说，在选择知识表示模式时，应从以下几方面进行考虑：

1. 充分表示领域知识

确定一个知识表示模式时，首先应该考虑的是它能否充分地表示领域知识. 为此，需要深入地了解领域知识的特点以及每一种表示模式的特征，以便做到“对症下药”. 例如，在故障诊断领域中，其浅知识一般具有经验性、因果性的特点，适用于产生式表示模式；而深知识是对诊断对象的结构与功能关系进行描述，此时用产生式表示就不能反映出知识间的结构关系，而需要用框架表示模式才合适.

2. 有利于对知识的利用

把知识表示出来并存储到计算机中去的目的是为了利用这些知识进行推理，以便求解现实问题. 所谓推理是指根据问题的已知事实，利用存储在计算机中的知识推出新的事实或者执行某个操作过程. 推理与知识表示有着密切的关系，如果一种表示模式的数据结构过于复杂或者难以理解及实现，则必然会影响到系统的推理效率，从而降低系统求解问题的能力.

3. 具有合适的数据结构

一种知识表示模式应符合人们的思维习惯，便于人们理解. 另外，在一个诊断系统初步建成后，经过对一定数量实例的运行可能会发现其知识在质量或性能方面存在某些问题，此时或者需要增补一些新知识，或者需要修改甚至删除某些已有的知识. 在确定知识的表示模式时，应充分考虑维护与管理的方便性.

目前用得较多的表示方法主要有以下几种：

1）传统的知识表示法，包括逻辑表示法，语义网络表示法，产生式表示法，框架表示法等；

2）面向对象的知识表示法；

3）神经网络的知识表示法；

4) 不确定性知识的表示法.

1.5.1 谓词逻辑表示法

谓词逻辑适合于表示事物的状态、属性、概念等事实性的知识,也可方便地表示事物间的因果关系,即规则.

对于简单的事实,如:这辆小车是福特牌可表示为

is_a (car ford)

即事实由单个关系 is_a 和客体 car, ford 组成,我们称关系名为谓词,客体为变元. 客体可以是常量,也可以是变量,在上例中的两个客体都是常量,再如:有些车是福特牌这一事实可表示为

is_a (X ford)

在以后的推理过程中,变量可被某个常量填入,称为实例化.

客体的顺序必须事先给定,不同的定义方式可将同一个事实理解成完全相反的意思,如下面这个例子:

teach (smith tom)

在不同的顺序定义中可以理解为 smith 教 tom 和 tom 教 smith 两种,但顺序一给定,意思也就明确了.

上面所列的都是两个客体,称为二元谓词. 而 is_a(dog)为一元谓词,同样,可定义三元谓词,四元谓词……像 works(smith ibm computer-science)为三元谓词.

我们可以通过逻辑词将简单的谓词公式联结起来构成复合的谓词公式,用来表达复杂的内容. 这些联结词有否定词、合取词、析取词、条件词、双条件词等,在这里我们不作过多的介绍.

现在假设有如下一条经验知识:如果小轿车的发动机冷却水温度过高,发动机油压过低,同时发动机声音异常,且有油糊味,那么,这辆小轿车的发动机内有烧结物. 对于这条经验知识,我们可用下面的方法表示:

rule(1,"发动机内有烧结物","发动机冷却水温过高","发动机油压过低","有油糊味","发动机声音异常"),rule 是谓词,数字 1 是知识编号.

在更多的情况下，为了方便，利用表的形式将知识表示出来，如上面这条知识可以按照下面这种方式表示出来：

rule(1,“发动机内有烧结物”,“小轿车”,[1,2,3,4]).

cond(1,“发动机冷却水温过高”).

cond(2,“发动机油压过低”).

cond(3,“有油糊味”).

cond(4,“发动机声音异常”).

第一条中的“发动机内有烧结物”是小轿车的故障原因，[1,2,3,4]是故障症状表. 下面的四条是用来表示事实的，cond 是谓词，前面的数字是故障症状编号，后面的中文是故障症状. 显然，在知识库中拥有大量知识的情况下，使用这种表示方法，可以使得知识库简洁易读，且利于扩充. 另外，谓词逻辑是一种接近于自然语言的形式语言，人们比较容易接受，用它表示的知识便于理解. 但是，谓语逻辑表示法只能表示精确性的知识，不能表示不精确性的知识，而在人类的知识中大多都是不精确及模糊的知识，这就使得谓词逻辑表示知识的范围受到了限制.

1.5.2 语义网络表示法

语义网络是一种采用网络形式表示人类知识的方法. 形式上，语义网络是由一组节点和一组连接节点的弧构成，节点表示问题领域中的物体、概念、动作、状态、属性等，弧表示各种语义联系，指明所连接节点间的某种关系. 节点和弧都必须带有标记，以便区分各种不同对象以及对象间的各种不同的语义联系. 每个节点可以带有若干属性，一般用框架表示. 另外，节点还可以是一个语义子网络，形成一个多层次的嵌套结构.

语义网络可以描述事物间复杂的语义联系，常见的主要有如下几种：

1）实例联系：用于表示类节点与所属实例节点之间的联系，通常标识为 ISA(is_a). 例如“泄漏是一种故障”，可以表示为如图 1.8 所示，其中，“泄漏”与“故障”是两个节点，它们之间的弧线及

其上面的标识“ISA”是这两个节点之间的语义联系，它具体地指出“泄漏”是“故障”中的一种，两者之间存在类属关系.

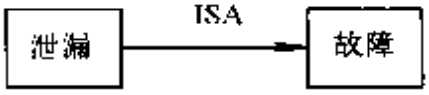


图 1.8 ISA 联系

ISA 是在人工智能中用得最多的一种语义联系，其含义很广泛，可以认为它既能表示概念与概念之间、类属与类属之间的关系，也可以表示类属与个体之间的关系. 通常把它理解为“是一个”、“是一种”、“是一条”等等.

2) 泛化联系：用于表示一种节点(如旋转机械)与更抽象的类节点(如机械)之间的联系，通常用 AKO(A Kind of)表示. AKO 是一个偏序联系，通过 AKO 可以将问题领域中的所有类节点组织成一个 AKO 层网络. 图 1.9 给出机械设备分类系统中的部分概念类型之间的 AKO 联系描述.

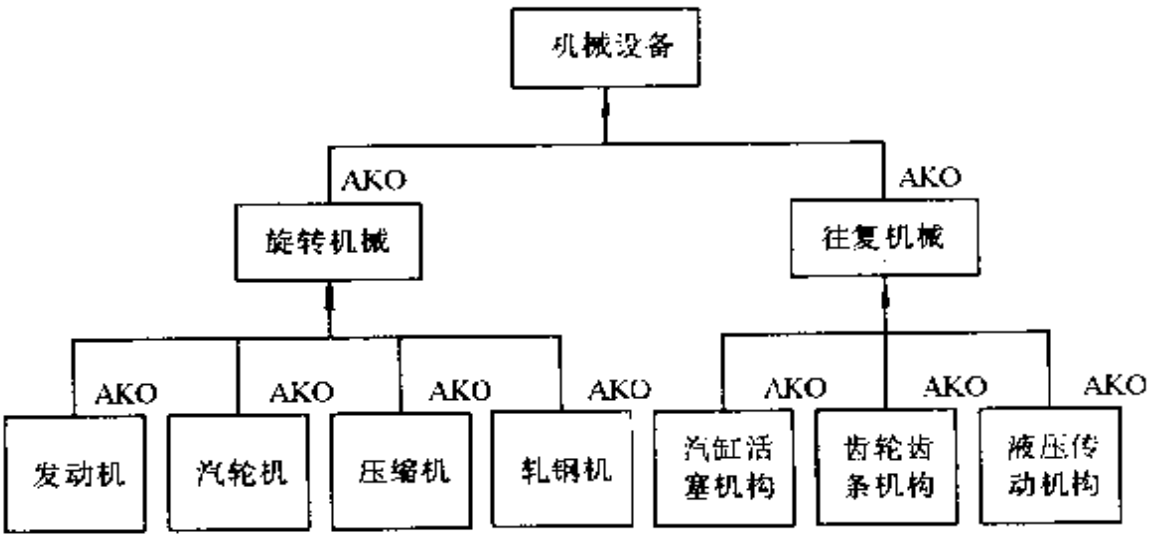


图 1.9 AKO 联系

3) 聚焦联系：用于表示部分(或部件)与整体之间的联系，通常用 Part_of 表示. 例如“开关是台灯的一个部件”可用图 1.10 表示. 聚集联系基于概念的分解性，将高层概念分解为若干低层概念

的集合.

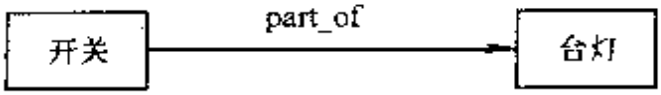


图 1.10 Part-of 联系

4) 属性联系:用于表示个体属性及其取值之间的联系,通常用有向弧表示属性,用这些弧所指向的节点表示各自的值.如图 1.11 所示,约翰的性别是男性,年龄为 30 岁等.

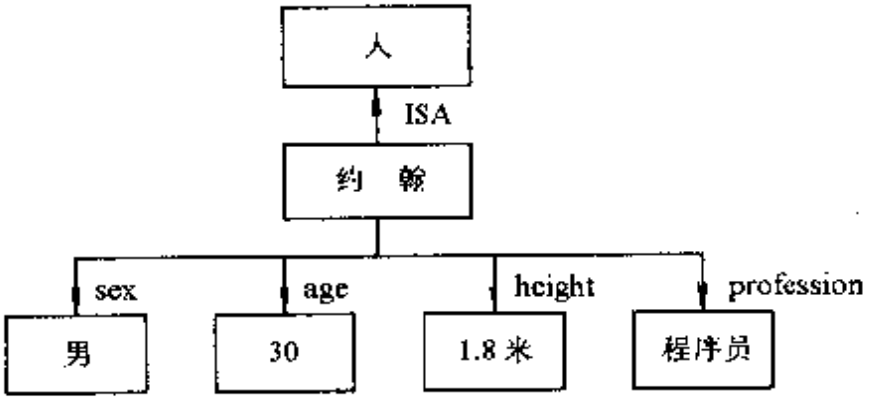


图 1.11 属性联系

图 1.12 是关于自行车描述的语义网络,其中包含有实例、泛化、聚集和属性四种联系.

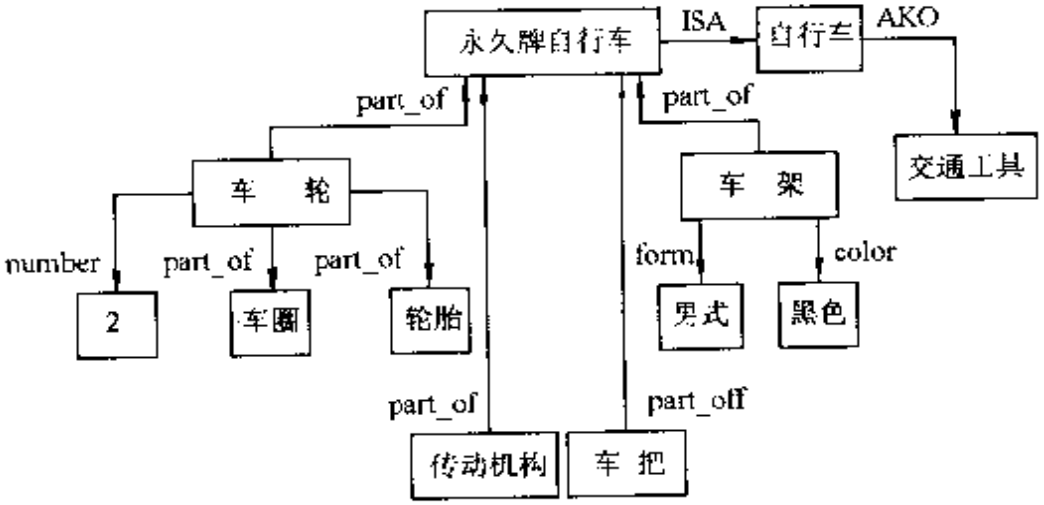


图 1.12 描述自行车的语义网络

语义网络知识表示法的优点是能把各种事物有机地联系起来，知识表示简洁、直观，且求解问题时可以通过网络的连接关系推导有关对象和概念，而不必遍历整个庞大的知识库，因而在专家系统等领域得到了广泛的应用. 它的缺点是不便于表达深知识，如与时间因素有关的动态知识. 另外，知识表达的内容受到节点联系的限制，如增加节点之间的联系将会大大增加网络的复杂程度，给知识的存储、修改及管理维护带来困难.

1.5.3 产生式表示法

产生式(production rules)表示法又称为规则表示法. 产生式通常用于表示具有因果关系的知识，其基本形式是

$$P \rightarrow Q$$

或者

IF P TNEN Q

其中，P 代表条件，如前提、状态、原因等；Q 代表结果，如结论、动作、后果等. 其含意是：如果前提 P 被满足，则可推出结论 Q 或执行 Q 所规定的动作. 典型的产生式的表示模式是：

IF[Premisers]THEN[action(s)]ELSE[action(s)](如果[前提]则[结果]否则[结果]).

我们举一条发动机诊断领域的产生式表示的实例：

IF {(Fuel-Consume-Enlarge), ANd (Black-Smoke), ANd (Exhaust-Pipe-Blowout)}

THEN{(Ignition-Ahead-Time-Small)}.

(如果燃料消耗过大，而且发动机冒黑烟，以及排气管发出爆破声，则发动机点火提前时间小).

把一组产生式放在一起，让它们互相配合，协同作用，一个产生式生成的结论可以供另一个产生式作为前提使用，以这种方式求得问题的解决，这样的系统就称为产生式系统，也称之为基于规则的系统.

一个产生式系统由三个基本部分组成：

1. 规则库

由产生式所组成的集合称为规则库.规则库反映了领域知识,其内容是否完整、一致将直接影响到系统的功能和性能.规则库中的每一条规则都有一个编号,系统运行时通过编号标识每一条规则.

2. 综合数据库

综合数据库又称为事实库、上下文、黑板等.它是一个类似缓冲器的数据结构,用于存放问题求解过程中的各种当前信息.例如问题的初始状态、推理时得到的中间结论及最终结论等.当规则库中某条产生式的前提可与综合数据库中某些事实匹配时,该产生式就被激活,并把其结论放入到综合数据库中,所以综合数据库的内容是在不断变化的,是动态的.

3. 推理结构

这是一组程序,负责整个产生式系统的运行.粗略地说,它要做以下几项工作:

1) 按一定的策略从规则库中选择规则与综合数据库中的已知事实进行匹配.所谓匹配就是把综合数据库中的事实与规则的前提进行比较,如果二者一致,称为匹配成功,相应的规则称为可用的;否则称为匹配不成功,相应的规则称为不可用的.

2) 匹配成功的规则可能不止一条,此时称为发生了冲突,推理机构必须有相应的解决冲突的策略,以便从中选出一条执行.

3) 在执行某一条规则时,如果规则的右部是一个或多个结论,就把这些结论加入到综合数据库中;如果规则的右部是一个或多个操作,则执行这些操作.

4) 随时掌握结束产生式系统运行的时机,以便在适当的时候终止问题的求解.

产生式表示法的优点在于接近人的思维方式,知识表示直观、

自然，又便于推理.产生式有固定的格式，任何一个产生式都由前提与结论这两部分组成，这种统一格式易于设计、控制和检测.规则表示模块化，它为知识的增、删、改带来了方便；为规则库的建立与扩展提供了可管理性.但是，产生式表示的刚性太强，对层次的表达力很弱，在推理过程中不能省略事先确定的相继关系，必须一步步前后匹配，从而降低了推理效率.另外，产生式适合表示具有因果关系的知识，不能表达具有结构性的知识.因此，在近几年的发展中，大型复杂的专家系统已转向采用框架式表示法.

1.5.4 框架式表示法

一个复杂系统在结构上总是由若干个子系统组成的.在系统的工作过程中，这些子系统相互之间密切配合，按照一定的规律工作，并能完成一定的功能，从而可实现整个系统的功能.同样，各个低层次的子系统亦是如此.对系统的这种结构与功能关系进行定性描述的一种有效方式.即是框架(frame)式结构.

框架式知识表示方法的发展是这样一个认识过程，那就是人类具有根据以前相类似状态的经验来解释当前碰上的新的状态的能力，这个能力使我们能在每一件经历过的事件中积累知识而不是碰上一件新事物，即从头开始.我们每个人都有许多知识和经验在大脑里，并用它们去分析和解决问题.框架理论认为人们对现实世界中各种事物的认识都是以某种类似于框架的结构存储在大脑之中，当面临新事物时，就从脑中取出一个相近的框架来进行匹配，如能匹配成功，就得到了对此新事物的认识；如果匹配不成功，则寻找原因，重新取一个更能与新事物匹配的框架，或者根据实际情况对最相近的框架进行修改、补充，从而形成新的认识.例如，当人们一提到“教室”这一概念时，立即就会想到它有四面墙，有天花板和地板，有课桌、坐凳以及黑板、门窗等.这就是一个关于教室的框架.当你走进一个房间时，如能看到这些东西，经与脑中关于教室的框架匹配，就会得出“这是个教室”的结论.另外，如果你希望把一个房间作为教室，经过与教室框架的匹配，就会得知这个房间

应该配置课桌、坐凳、黑板等物，至于教室中每面墙的尺寸、黑板的位置、桌凳的数量及颜色等，各个具体的教室都可能不同，每个教室的一组数据构成了“教室”框架的一个事例，称为事例框架.

框架的形式为

框架名

槽名 1：槽值 1

槽名 2：槽值 2

……

槽名 i ：侧面 1：槽值 $i1$

侧面 2：槽值 $i2$

……

侧面 m：槽值 im

槽名 n：槽值 n

框架中的槽可以有一个或多个描述体，每个描述体各有对应的槽值.

表 1.2 是框架式表示方法的一个例子，表中通过型号、制造者、重量、马达、加速等槽来描述一种轿车. 在这些槽中，当确切知道有关事物的槽值时就可填入. 如果缺少有关事物的信息，则有的槽包含有缺省值，如车门数. 如果在问题求解过程中，当前状态相应知识与事实没有得到，就用缺省值推理，这是框架式借用过去经验的一种方式. 另一种槽含有“与过程有关”的说明语，表示这些槽可进一步分解，并追加相应的描述体和槽值. 在表中，有一个槽指向另一个框架，这样能表达知识单元间的关系和层次，也可根据需求来确定知识表示的不同详尽程度，不同抽象层次. 参见式的槽可能保留几个最必要的描述体，如果推理过程不需要那么详尽的知识，就可以不进行参见槽的匹配.

框架表示法的优点是：

1）它善于表达结构性的知识，能够把知识间的结构关系充分表示出来，因此它是一种经过组织的结构化的知识表示方法. 这一特点是产生式表示法所不具备的. 另外，它与语义网络所表示的结

构性又不完全相同，语义网络通常用于表达知识间的宏观结构关系，而框架表示法适于表示固定的、典型的概念、事件和行为. 在讨论语义网络的概念时曾经指出，如果我们把一种事物、概念、情况、属性等作为一个节点，节点间的语义关系通过语义网络表示出来，则每个节点可用框架表示其内部的结构关系.

表 1.2　框架式表示的例子

Automobile Frame	汽车框架
Class of:Transfortation	上位类:交通
Name of Manufacturer:Audi	制造者:奥迪
Origin of Manufacturer:Germany	制造者所在地:德国
Model:5000Turbo	型号:涡轮 5000
Type of Car:Sedan	汽车类型:轿车
Weight:3300 lb	重量:3300lb
Wheel base:105. 8 inches	车轮架:105. 8in
Number of Doors:4(default)	车门数:4(缺省)
Engine:(Reference Engine Frame)	马达:(参见马达框架)
Type:In-line,Overhead Cam	类型:直列式,上置凸轮
Number of Cylinders:5	汽缸数:5
Transmission:3-Speed automatic	变速器:3 速自动
Acceleration(Procedural attachment)	加速(与过程有关)
0—60:10. 4seconds	0—60:10. 4s
Quarter Mile:17. 1 seconds,85mph	四分之一英里:17. 1s,85mph
Gas Mileage:22mpg average(Procedual attachment)	耗油量:平均 22mpg(与过程有关)
Engine Frame	马达框架
Cylinder bore:3. 19 inches	汽缸内径:3. 19in
Cylinder Stroke:3. 4 inches	冲程:3. 4in
Compression ratio:7. 8 to 1	压缩比:7. 8 比 1
Fuel System:Injection with turbochanger	燃油系统:涡轮增压器喷射
Horsepower:140hp	马力:140hp
Torque:160ft/LB	转矩:160ft/LB

2) 框架之间可以形成层次的或更复杂的关系，组成一种框架网络，代表整块的知识结构，可以表示复杂的知识内容. 在框架网络中，下层框架可以继承上层框架的槽值，也可以进行补充和修改，这样不仅能把知识充分地表达出来，而且减少了知识的冗余，

较好地保持了知识的一致性.

3) 框架表示法体现了人们在观察事物时的思维活动，当遇到新的事物时，就从已有的记忆中调用类似事物的框架，将其中某些细节加以修改、补充，形成对当前事物的认识.

框架表示法除了上述优点之外，其主要的不足之处是不善于表达过程性的知识. 因此，框架表示法经常与产生式表示法结合起来使用，这样可以取得互补的效果.

目前，人们已经对框架式表示法做了相当多的研究及应用工作，例如美国 Intellicorp 公司研制了基于框架，基于产生式规则，面向过程、面向对象的通用专家系统工具 KEE(Knowledge Engineering Environment).

1.5.5 面向对象的表示法

面向对象的方法学的第一个基本观点是：认为世界是由各种"对象"组成的. 任何事物都是对象，是某对象类的元素；复杂的对象可由相对比较简单的对象以某种方式组成. 甚至整个世界也可以一些最原始的对象开始，经过层层组合而成. 从这个意义讲，整个世界可认为是一个最复杂的对象. 根据这种观点，为了要认识一个复杂对象必须首先去认识构成该复杂对象的各子对象.

面向对象的方法学的第二基本观点是：所有对象被分成各种对象类，每个对象类都相应地定义了一组所谓"方法"，实际上可视它们为允许作用于该类对象上的各种操作. 对该类中的对象的操作都可通过应用相应的"方法"于该对象来实现. 这种操作在面向对象的方法学中被称为"送一个消息给某对象".

面向对象的方法学的第三个基本观点是：对象之间除了互递消息的联系之外，不再有其它联系. 因此，对象之间的界面很清楚.

面向对象的方法学的第四个基本观点是：一切局限于对象的信息和"方法"的具体实现等都被封装在相应对象类的定义之中，在外面是不可见的，这即所谓"封装"的概念，所以对象类的定义非常模块化. 它们具有类间联系少和类中凝聚力大的优点，这是完全

符合软件工程的基本原则的，而且，容易通过“封装”来实现“数据抽象”.

面向对象的方法学的第五个基本观点是：对象类将按“类”，“子类”与“超类”的概念构成一种层次关系（或树形结构）.在这种层次结构中，上一层对象具有的一些属性或特征可被下一层对象继承，除非在下一层对象中对应的属性作了重新描述（这时以新的属性为准），从而避免了描述中的信息冗余.这称为对象类之间的属性继承关系.

按照面向对象方法学的观点，一个对象的形式定义可以用如下四元组表示：

对象::=(ID,DS,MS,MI)

对象的标识符 ID(IDentifier)又称对象名，用以标识一个特定的对象，正如一个人有人名，一所学校有校名，一支军队有番号，每一特定事物（如机械故障）有特定的标记.

对象的数据结构 DS(Data Structure)描述了对象当前的内部状态或所具有的静态属性，常用一组〈属性名 属性值〉表示.例如，一支军队当前所在的地理位置、各级首长的情况、当前的兵力配备和武器装备状况等都可以作为该支军队（对象）的内部状态.

对象的方法集合 MS(Method Set)用以说明对象所具有的内部处理方法或对受理的消息的操作过程，它反映了对象自身的智能行为.例如，军人以服从命令为天职，对于上级的一个命令或决定，如何具体贯彻执行，需要制订有关措施；一个军事指挥员，根据上级意图、我情、敌情和天时地理，对作战方案应作出合理的决策.在面向对象的系统中，这些措施和决策都要通过方法描述.

对象的消息接口 MI(Message Interface)是对象接收外部信息和驱动有关内部方法的唯一对外接口.这里的外部信息称为消息.例如，一支军队接收的消息可能有上级的命令，友邻部队的请求，敌军的撤退或进攻的信息，以及其它各种军事情报等.发送消息的对象称为发送者，接收消息的对象称为接收者.消息接口以消息模式集的形式给出，每一消息模式有一消息名，通常还包含必要

的参数表. 当接收者从它的消息接口受理发送者的某一消息时，首先要判断该消息属哪一消息模式，找出与之匹配的内部方法，然后，执行与该消息相联的方法，进行相应的消息处理或回答某些信息.

在面向对象的系统中，问题求解或程序执行是依靠对象间传递消息完成的. 最初的消息通常来自用户的输入，某一对象在处理相应的消息时，如果需要，又可以通过传递消息去请求其它对象完成某些处理工作或回答某些信息，其它对象在执行所要求的处理时同样可以通过传递消息与别的对象联系，至此下去，直至得到问题的解.

消息流统一了数据流和控制流，它是实现对象之间联系的唯一途径. 消息中只包含发送者给出的信息，这些信息往往表示对接收者的某种要求，但仅仅告诉接收者需要做什么，并不指示接收者如何去完成所需的处理. 消息完全由接收者解释，接收者可以独立决定以何种方式或通过什么样的操作过程去完成相应的工作. 同样的消息可以传递给不同的对象，不同的对象可以对同样的消息做出不同的反应，正如不同的部队对收到的同一军事情报会做出不同反应一样，而且同一对象可以接收多个对象传来的不同消息，对传来的消息可以返回相应的回答信息，也可以不予回答. 可以看出，面向对象系统中的消息传递与传统的子程序调用和返回有着明显的差别.

消息模式不仅定义了该对象所能受理的消息，而且还规定了该对象的固有处理能力. 每个对象的一种消息模式都对应于该对象内部的方法. 方法是对象固有处理能力的具体实现，软件中通常用一个可执行的代码段表示. 通过消息模式及消息引用，对应方法的代码段执行，相应的处理能力也就表现出来了. 在方法实施期间，通常还需要引用自己的内部状态，对有关数据进行操作，必要时可以修改自己的内部状态，还往往要发送一批消息依次请求其它对象做事. 每个对象的内部状态不允许其它对象直接引用和修改. 对象的内部状态和方法对外界是隐蔽的. 因此，只要给出对象的所有消息模式，包括相应于每个消息的处理能力，也就定义了

个对象的外部特性.于是,每个对象就像集成电路的芯片一样被封装在一明确的范围内,对外接口是它的消息模式集,受"黑盒"保护的内部实现由它的状态和操作细节组成.这就是面向对象方法的学习封装性.

如前所述,一个复杂对象常由若干相对简单的对象组成.简单对象所提供的某些消息有时可能仅供复杂对象内部使用,复杂对象的这种不向外界公开的消息称为该复杂对象的私有消息.在日常生活中,私有消息的例子比比皆是.例如,一个人或一个集团有各种各样的能力,虽然许多能力是可以公开告知外界的,但也不乏纯属内部或不愿对外公开和不提供对外服务的能力.这些不对外公开的能力所对应的消息就是私有消息.相对地,对象向外界公开提供的消息称为该对象的公有消息.在面向对象的语言中,对象的外部接口是以对象协议或规格说明的形式提供的.协议是一个对象对外服务的说明,它告知该对象可以为外界做些什么.外界对象能够并且只能够向该对象发送协议中所包含的消息,也就是说,请求对象进行操作的唯一途径是通过该对象协议中所包含的消息进行.从私有消息和公有消息上看,协议是一个对象所能接受的所有公有消息(模式)的集合.可以说,一个对象就是在它的协议下封装起来的.

封装是一种信息隐蔽技术,它使对象的设计者与对象的使用者分开,使用者无需知道对象行为的实现细节,而只需通过对象协议中的消息便可访问该对象.显式地把对象的外部定义和对象的内部实现分开是面向对象系统的一大特色.封装性本身就是模块性,模块的定义和实现分开,使面向对象的软件系统便于维护和修改,这也是软件工程所追求的目标之一.

面向对象的知识表达方法以领域对象为中心,以对象为基本单位表示知识,将多种单一的知识表示方法如规则、框架等表示方法按面向对象的原则组成一种混合的知识表达形式,即以对象的属性、动态的行为特征、相关领域的知识和数据处理方法等有关知识封装在表达对象的结构中.对象类是对一类对象的抽象描述,而

对象的实例则是表达具体的对象.类、实例和对象是三个不同的概念.对象之间除了通过消息传递之外,不再有其它任何联系,实现了信息的封装.

面向对象的知识表示方法将对象抽象成类,将类实例化为对象.这种知识表示方法与人的认知习惯相近;在结构上具有层次分明、模块性强、知识单位独立性强等优点.通过继承可以减少知识表达上的冗余,知识库的修改、增删以及使用维护都十分方便,对一个知识单元进行修改不会影响其它单位,每一知识单元中所包含的知识规则有限,推理空间小,从而提高了推理效率.

1.5.6 神经网络表示法

在人类的知识中,有一些知识是需要通过一系列的例子才能总结出来的,如果把它们都穷举编码,就可能引起知识组合爆炸,这就需要我们用别的办法来表达这样的知识.如果在表达一个概念时,不是像语义网络那样用一个节点表示一个概念,而是把表示这个概念的有关信息分布在许多单元上,并将信息的某种分布方式加以表达,那么不仅可以避免组合爆炸,而且当某个单元上的信息发生畸变失真时,也不会使所表达的概念属性发生重大的变化.另外,用这种方式表示知识时,还可使一些类似的概念分布在共同的单元上.这就是神经网络表示知识的基本思想.也许,人脑中的知识就是这样表示与存储的.

在目前神经网络的研究中,较有代表性的工作是"并行信息分布处理"模型.这种模型假设信息处理是通过大量称为"单元"的简单处理元件交互进行的,每个单元都对上层的单元发出激励或抑制信号.这里所说的"并行性"是指网络是针对全局的,所有的目标都同时进行处理;这里所说的"分布性"是指信息分布在整个网络内部,每个节点及其连线上只表达部分信息,而不是一个完整的概念.

传统的知识表示,不管是产生式规则,还是语义网络,都可以看做一种显式表示,而神经网络知识表示是一种隐式表示.产生式系

统中知识独立表示为规则，在神经网络中将同一问题的知识表示在同一网络中．如下列规则可以用图 1.13 所示的神经网络来表示．

if $(x_1=0)$ and $(x_2=0)$ then $(y=0)$
if $(x_1=0)$ and $(x_2=1)$ then $(y=1)$
if $(x_1=1)$ and $(x_2=0)$ then $(y=1)$
if $(x_1=1)$ and $(x_2=1)$ then $(y=0)$

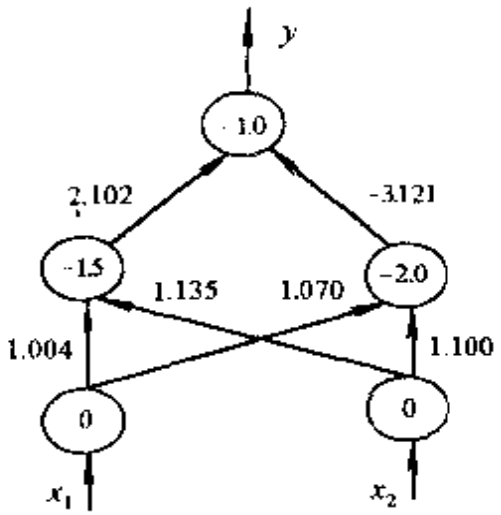


图 1.13　神经网络表示法

神经网络的知识表示法有如下优点：

1）以分布方式表示信息．任何知识规则都可以变换成数的形式，因而便于知识库的组织与管理，且通用性强．

2）可以拥有大量的知识，如果神经网络拥有 N 个输入单元，且输入模型是二值逻辑，则可提供表达知识的样本数为 2^N．

3）便于实行并行联想推理和自适应推理，因而在模式识别、图象信息压缩、故障智能诊断方面的应用上取得了较大的进展．

4）在一定程度上模拟了专家凭直觉解决不确定性问题的过程．能够表示事物的复杂关系，如模糊因果关系等．

但是神经网络对于给定的输入，用户只能得到一个结果，不清楚整个推理过程，因此解释困难．

1.5.7　不精确知识表示法

前面介绍的几种知识表示方法都假定所表示的事实、状态、前提等不是真就是假，但在现实世界中并非如此．在诊断的环境中，不确定的问题占多数，比如，一个患者有发烧、咳嗽的症状，可是他到底得的是感冒？肺炎？或者是其它疾病？仅根据这两个症状是确定不下来的．实际上，必须需要人类专家处理的问题常常多是与缺少“严格性、精确性和条理性”相联系的，所以在专家系统中，如何表示和处理各种不精确知识就成为一个重要的课题．本节将针对前

面讨论的几种基本知识表示方法讨论不精确知识的表示问题.

1. 基于产生式的不精确知识表示法

(1) 可信度方法

人们在长期的实践活动中,对客观世界的认识积累了大量的经验.当面临一个新情况时,可用这些经验对问题的真假或为真的程度作出判断.人们对一个事物或现象为真的相信程度称为可信度.

显然,可信度带有较大的主观性及经验性,其准确性难以把握,但由于机器故障的信息环境多是一个不确定性的环境,不能像数学那样具有严密性和精确性,因此,用可信度来表示不精确知识不失为一种可行的方法.另外,领域专家都是所在领域的行家里手,有丰富的专业知识及实践经验,对领域内的知识也不难给出其可信度.

规则的一般形式是

IF E THEN H (CF(H,E))

其中,E 为前提,它既可以是一个简单条件,也可以是由多个简单条件构成的逻辑组合,例如 $E=E_1$ AND E_2 AND E_3;H 是结论,它也可以是一个或多个结论;CF(H,E)是该规则的可信度,称为规则强度,它表示当条件 E 为真时,则结论 H 有 CF(H,E)大小的可信度.CF 在[−1,1]上取值,值越大表示相应的知识越为真.当 CF 的值为 1 时,表示相应的知识为真;当 CF 值为 −1 时,表示相应的知识为假.例如有这样一条产生式:

规则 R_1:

IF(如果):(汽缸上下温差超过允许值 30℃)和

(机组处于热态启动过程中)

THEN(则):(汽缸工况处于热态不平衡) CF0.95

它表示当列出的各个前提都得到满足时,结论有 0.95 的可信度.

(2) 概率方法

概率论是一门研究和处理随机现象的学科.在随机现象中事

件本身的含意是明确的，只是由于发生的条件不充分使得条件与事件之间不能出现决定性的因果关系，从而使事件的出现与否表现出不确定性，这种不确定性称为随机性.

对于具有随机性的知识可用概率论的有关方法进行表示和处理. 但由于大容量的样本不易获得，使得纯概率的方法受到了限制. 因此，在应用时通常用一些改进的理论模型和经验公式来处理知识的不确定性，主观 Bayes 方法就是其中的一种. 在该方法中，每条规则的表示形式是

IF E THEN (LS,LN) H ($P(H)$)

其中

1) E 是规则的前提，它可以是用 AND 或 OR 连接起来的复合条件.

2) H 是结论；$P(H)$是先验概率，它指出在没有任何专门证据的情况下，结论 H 为真的概率. $P(H)$的值由领域专家给出.

3) LS 称为充分性量度，用于反映 E 对 H 的支持程度. 它的取值范围为$[0,+\infty)$，值越大表示 E 越支持 H. LS 的值由领域专家给出.

4) LN 称为必要性量度，用于反映非 E 对 H 的支持程度，即当 E 所对应的证据不存在时，对 H 为真的支持程度. LN 的取值范围为$[0,+\infty)$，值越大表示非 E 越支持 H. LN 的值由领域专家给出.

下面给出两个用主观 Bayes 方法表示不精确知识的例子：

IF E_1 THEN (2,0.01) H_1(0.1)

IF E_2 THEN (100,0.001) H_2(0.05)

(3) 模糊逻辑方法

可信度方法是用人们主观上对事物的相信程度来描述事物的不确定性，概率方法是用随机性来描述事物的不确定性，两者虽然都可以作为不精确知识的表示方法，但都没有把事物本身所具有的模糊性反映出来，也不能对其客观存在的模糊性进行有效的处理. 扎德提出的模糊子集概念及其可能性理论弥补了这一缺憾，在

模糊知识的表示和处理方面得到了应用.

模糊产生规则:人类领域专家的直觉、经验、窍门和启发式知识往往缺乏明确的逻辑联系,有时就是领域专家本人也很难把这些知识以及它们之间的关系表述得十分清楚,因而领域专家经常使用一些模糊语言来描述他们的这些知识,如"马力不足"、"排气温度偏高"、"振动大"等,并用模糊逻辑来进行推理.为了更好地描述和模拟领域专家的模糊思维和推理行为,我们在模糊集合论的基础上,提出了一种模糊产生式规则.

模糊语言的含糊性主要是由两方面引起的,其一是由于语句中含有不精确的语言量词所致;其二是由于词汇本身的模糊性所产生的.例如"这是一批新机器,不太可能出故障",这里的"不太可能"是模糊量词,而"新机器"是模糊词汇.为了在产生式规则中描述模糊语句的模糊性,就需要进行量化处理.对于模糊量词,我们可以给它赋给[0,1]中的任意两个实数所构成的一个区间中的值来定量地表示它,表 1.3 中给出了常见的一类模糊量词的区间值,而词汇本身的模糊性所引起的不确定性可以采用隶属函数来表示.

表 1.3 常见模糊量词的区间值

模糊量词	数值区间
总是	[1.00,1.00]
很难	[0.93,0.99]
强	[0.80,0.92]
或多或少强	[0.65,0.79]
中等	[0.45,0.64]
或多或少弱	[0.30,0.44]
弱	[0.10,0.29]
很弱	[0.01,0.09]
无	[0.00,0.00]

对于模糊语句进行量化处理之后,我们即可在传统的产生式规则中表示模糊知识,为清楚起见,我们将经过改造的这种规则称为模糊产生式规则,其形式如下:

IF(p_1,t_1) and (p_2,t_2) and…and (p_n,t_n)

THEN Q CF x

其中，规则前件中的 $p_1, p_2, \cdots, p_n$ 分别表示断言，对每个断言可按照前面所描述的方法赋给它一个相应的隶属值 $t_1, t_2, \cdots, t_n$，用来表示断言的确定性程度；Q 表示规则的后件，x 为可信度 CF 的值，它表示规则的信任程度. 当规则前件中的断言 $p_1, p_2, \cdots, p_n$ 均是确定性的断言时，那么各断言的隶属值 $t_1, t_2, \cdots, t_n$ 就均为 1，此时模糊产生式规则就变为普通的产生式规则了.

2. 基于框架的不精确知识表示法

框架也可以表示不精确知识，具体为

(1) 槽值允许是模糊数据、模糊操作或模糊过程

这使得很多模糊现象可以表示在框架中，从而大大丰富了框架的表达力. 模糊数据可以是各种模糊数、模糊语言值、模糊集、模糊关系、各种模糊逻辑公式等，它们所能表示的意义是十分广泛的. 模糊操作或模糊过程可以是一个简单的模糊动作，一个(或一组)产生式规则，也可以是一个用模糊程序设计语言编写的一段程序(或一个过程)等.

(2) 框架与框架之间的各种关联可以模糊化

使得框架之间的关联并非不是有就是无，可以允许有各种中间状态. 框架间模糊的关联采用一种关联强度来描述它们之间的关联程度. 连接强度可用[0,1]间的数、各种模糊数或模糊语言值等来描述. 然后采用图论中计算一条路径两端结点间的关联强度的办法来计算间接关联强度. 例如有框架关联关系如图 1.14 所示.

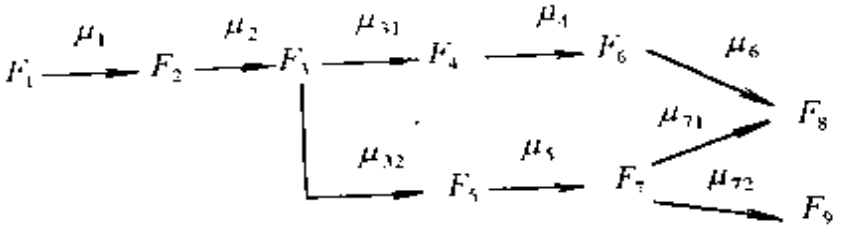


图 1.14 框架关联关系

F_1 与 F_9 之间的关联强度为

$$S(F_1,F_9)=\min(\mu_1,\mu_2,\mu_{32},\mu_5,\mu_{72})$$

而 F_1 与 F_8 之间的关联强度为

$$S(F_1,F_8)=\min(\mu_1,\mu_2,\mu_{31},\mu_4,\mu_6)$$

特别指出的是为使继承关系模糊化，可在任意两个具有直接继承关系的框架之间，设一个“继承因子”或“继承强度”，然后设计一种计算间接继承强度的公式（例如，采用连乘积或取极小等办法）. 当间接继承强度小于某设定的阈值（$0<\tau<1$）时，就认为它们之间不再有继承关系. 框架间的关联模糊化是很有实际意义的，因为现实生活中事物之间的联系很多都是模糊不清的，根本就不可能用精确的模型来描述，模糊框架在这方面可以较好地满足要求.

（3）框架中的约束条件亦可模糊化

允许用各种模糊逻辑公式来表示判断条件，从而条件是否满足也得使用一个阈值来控制，条件真值大于等于该阈值时算满足，否则算不满足.

3. 基于语义网络的不精确知识表示法

用语义网络表示不精确知识时，可以从以下两个方面着手进行：

1）节点的内容可用模糊数据或不精确框架表示.

2）节点间的语义联系可通过建立联系强度使其不精确化，其方法与框架中继承强度类似.

1.6 基于知识的诊断推理

前面我们讨论了知识获取及知识表示的有关问题，这样就可把问题领域中的知识表示出来，并以某种内部形式存储到计算机中，形成知识库。但是，正如一个人只有知识而没有运用知识求解问题的能力仍然算不上“聪明”一样，对一个智能系统来说，不但应使它具有问题领域的知识，还应该使它具有运用知识求解问题的

能力。运用知识的过程是一个思维过程，即推理过程。

1.6.1 推理的基本概念

推理是从一个或几个判断中得出一个新判断的思维形式. 任何推理都有这样两个组成部分，即推理所依据的判断(已知判断)以及推出的新判断. 前者叫做前提，它包括知识库中的领域知识及问题的初始证据；后者叫做结论，是由已知判断推出的新判断. 在智能系统中，推理是由计算机程序实现的，称为推理机.

例如，在医疗诊断专家系统中，专家的经验及医学常识以某种表示形式存储于知识库中，当用它为病人诊治疾病时，推理机从病人的症状及化验结果等初始证据出发，按某种搜索策略在知识库中搜寻可与之匹配的知识，从而推出某些中间结论，然后再以这些中间结论为证据，推出进一步的中间结论，如此反复进行，直到推出最终的结论，即病人的病因与治疗方案. 像这样不断运用知识库中的知识，逐步推出结论的过程就是推理.

人类的智能活动有多种推理方式，人工智能作为对人类智能的模拟，相应地也有多种推理方式，下面分别从不同角度简要地介绍一下这些推理方式.

1. 演绎推理、归纳推理、类比推理

若根据思维进程中从一般到特殊、从特殊到一般、从特殊到特殊的区别，推理可分为演绎推理、归纳推理和类比推理. 演绎推理是从一般到特殊的推理. 归纳推理是从特殊到一般的推理. 类比推理是从特殊到特殊的推理.

(1) 演绎推理

演绎推理是从已知的判断出发，通过演绎推出结论的一种推理方式，其结论就蕴含在已知的判断中，所以演绎推理是一种由一般到个别的推理. 由于结论是蕴含在已知判断中的，因而只要已知判断正确，则通过演绎推理推出的结论也必然正确.

演绎推理是一种必然性推理. 因为推理的前提是一般，推出的

结论是个别，一般中概括了个别，凡是一类事物所共有的属性，其中的每一个别事物必然具有，所以从一般中必然能够推出个别.然而，推出的结论是否正确，这要取决于推理的前提是否正确，以及推理的形式是否符合逻辑规则.

综上所述，演绎推理具有以下特点：

1）在演绎推理中，前提与结论之间有必然联系.这也就是说，当我们用任何具体内容代入前提与结论时，如果前提是真的，结论也是真的.这种必然的联系，有时叫做蕴函关系.

2）演绎推理，一般说来，是由一般（普遍）到个别（特殊），前提是普遍性判断而结论是个别性判断.

3）演绎推理的结论所断定的，没有超出前提所断定的范围.或者说，演绎是从一般性较大的前提导出一般性较小的结论的推理.与此相反，归纳是从一般性较小的前提导出一般性较大的结论的推理.

（2）归纳推理

归纳推理通常是指由个别性知识的前提推出一般性知识的结论的推理.

归纳推理是人类思维中最基本、最常用的一种推理形式.人们在由个别、特殊到一般的思维过程中经常要运用这种推理.

例如，人们早已知道，某些生物的活动是按时间的变化（昼夜交替或四季变更）来进行的，像鸡叫一遍天亮，牵牛花破晓开放，青蛙冬眠春晓，大雁春来秋往等都具有周期性的节律.现代科学的研究表明，某些生物测量时间的准确度是很高的，因而通常把生物这种测量时间的本领，称作“生物钟”.从充塞雨滴的微生物到高植物乃至人类，都可以找到这种无声无息的生物钟.生物钟已被概括为有机体的一个特征.这就是根据个别种类的生物体活动具有周期性节律而概括出的一个一般性结论：凡生物体的活动都具有时间上的周期性节律.这是一个归纳推理的过程.

归纳推理是从足够多的事例中归纳出一般性知识的推理，是从个别到一般（或普遍）.就是说，前提是个别性的判断而结论是普

遍性的判断，恰好与演绎推理相反．在归纳推理中，前提与结论之间，没有必然性的联系，而只是一种或然性联系．也就是说，当我们用某些具体内容代入前提与结论时，前提是真的，结论也是真的；但是，用另一些具体内容代入前提与结论时，前提是真的，但结论却是假的．

(3) 类比推理

类比推理是根据两个对象在一系列属性上是相同的，而且已知其中的一个对象还具有其它的属性，由此推出另一个对象也具有同样的其它属性的结论．

类比推理在人们认识客观世界和改造客观世界的活动中，具有非常重要的意义．科学上的许多重要理论和发现都是通过类比推理提出的．例如，惠更斯提出光的波动学说，是受到水波、声波的启发，英国医生詹纳发现“种牛痘”可以预防天花，则是受到挤牛奶女工感染了牛痘而不患天花的启发．

类比推理是依据下述方式进行的：

A 对象具有属性 a,b,c,d

B 对象具有属性 a,b,c

所以，B 对象也具有属性 d

这种推理方式的客观依据，是因为事物的各个属性并不是孤立存在的，而是互相联系和相互制约的．然而，类比推理的结论是或然的．这是因为客观上存在着以下两种情况：

1) 对象之间不仅存在着相似性，而且存在着差异性．

2) 对象中并存的许多属性，有些是对象的固有属性，有些是对象的偶有属性．如果把某对象的偶有属性类推到其它对象，那就犯了“机械类比”的错误．

如何提高类比推理结论的可靠性呢？

1) 前提中确认的相同属性愈多，那么结论的可靠性程度也就愈大．因为两个对象的相同属性愈多，意味着它们在自然领域中的地位也较为接近．这样，类推的属性也就有较大的可能是两个对象所共同的．

2）前提中确认的相同属性愈是本质的，相同属性与类推的属性之间愈是相关的，那么结论的可靠性程度也就愈大．因为本质的东西是对象的内在规定，对象的其它属性大多是由对象的本质决定的．因而，两个对象的相同属性如果是本质的，那么，它们就有其它一系列属性是相似的．

2. 精确推理、不精确推理

若根据推理时所用知识的确定性来划分，推理可分为精确推理与不精确推理．

计算机是在精确科学的沃土中培育起来的一朵奇葩．计算机解决问题的高速度和高精度是人脑望尘莫及的．有了计算机，精确方法的可行性大大提高．但也正是在使用计算机的实践中，使人们养成了追求严格、崇尚精确的习惯．然而，现实世界中的事物和现象大都是不严格、不精确的，许多概念是模糊的，没有明确的类属界限，很难用精确的数学模型来表示和处理．也就是说，大量未解决的重要问题往往需要运用专家的经验，而这样的问题是难以建立精确数学模型的，也不宜用常规的传统程序来求解．在此情况下，若仍用经典逻辑做精确处理，势必要人为地在本来没有明确界限的事物间划定界限，从而舍弃了事物固有的模糊性，失去了真实性．这就是为什么近年来，各种不精确推理迅速崛起，成为人工智能重要研究课题的原因．

不精确推理的主要理论基础是概率论．由于现实世界中，不易获得大容量的样本及其它一些原因，使得纯概率论方法受到了限制．为此，智能系统的建造者们提出了许多改进的理论模型和经验公式来处理不确定性．其中有代表性的不精确推理有下面五种方法：

1）MYCIN 的不精确推理模型；

2）主观 BAYES 方法；

3）模糊推理；

4）证据理论；

5）发生率计算.

关于不精确推理的论述较多，这里不能多引，需要深入研究的读者，可参阅有关论著.

3. 单调推理、非单调推理

所谓推理的单调性是指随着推理的向前推进及新知识的加入，推出的结论是否越来越接近最终目标.若以这一标准来划分，推理可分为单调推理及非单调推理.

建立在谓词逻辑基础上的传统推理系统是单调的，其意思是：已知为真的命题数目随着时间而严格增加.那是由于新的命题可加入系统，新的定理可证明，但这种加入和证明决不会导致前面已知为真或已证明的命题变成无效.这种单调推理有以下优点：

1）当加入新的命题时，不必检查新命题与原有知识间的不相容性.

2）对每一个已证明的命题，不必保留一个命题表，它的证明是以该命题表中的命题为依据.因为不存在那些命题会被取消的危险.

遗憾的是，人类的思维推理本质上不是单调的.人们对周围世界中各种事物的认识、信念和看法是处于不断地调整之中.获得了新的知识就可能要修正，甚至抛弃原有的观念或看法.在这种情况下结论并不随着新知识的增多而增加，有时不但不会增强已推出的结论，反而要撤消某些由不正确假设所推出的结论.这种性质与传统逻辑具有的特性不同，这就是人们所说的非单调推理.

举个例子来说，假设要去朋友家做客，并经过路旁的卖花店时，对于“这位朋友喜欢花吗?”这样的一个问题，可能没有任何具体信息可作为回答问题的依据.但若利用一般的规则——因为大多数人喜欢花，假定这个具体的人也喜欢，除非有相反的证据如对花过敏等，那么，这个问题便可以做得很好.这类推理是非单调的，即加进一条信息就可能迫使取消另一条信息，因为用这种方式推导出来的命题是依赖于在某个别的命题中缺少某种信念.即如果

前面那些缺少的命题一旦加入系统，就必须消除用非单调推理产生的命题. 因此，如果一个人拿着花走到朋友家门口时，见到这位朋友立刻打喷嚏，就应该取消以前的信念——这位朋友喜欢花. 当然，也必须取消建立在已被取消的信念基础上的任何信念.

人工智能界非单调推理研究的四个代表性的理论或系统是：

1) Reiter 的缺省理论；

2) Mcdermott 的非单调逻辑；

3) Mccarthy 的界限理论；

4) Doyle 的正确性维持系统.

它们代表了非单调推理的主要方面，后面的工作大多是在它们的基础上进行的. 读者如需要深入研究这些内容，可以阅读有关文献.

4. 正向推理、反向推理、混合推理

若根据推理的方向划分，推理可分为正向推理、反向推理及正向-反向混合推理.

(1) 正向推理

正向推理又称为数据驱动控制策略或前件推理. 其基本思想是：从问题已有的事实(初始证据)出发，正向使用规则，当规则的条件部分与已有的事实匹配时，就把该规则作为可用规则放入候选规则队列中，然后通过冲突消解，在候选队列中选择一条规则作为启用规则进行推理，并将其结论放入数据库中，作为下一步推理时的证据. 如此重复这个过程，直到再无可用规则可被选用或者求得了所要求的解为止. 这一过程可用如下算法描述：

Procedure data-driven

　　扫描知识库，形成可用规则集 S

while S 非空且问题未得到最终解

begin

　　调用冲突消解算法，从 S 中选出启用规则 R；

　　执行 R，将其结论部分放入数据库；

扫描知识库，形成新的可用规则集 S

end

正向推理的优点是比较直观，允许用户主动提供有用的事实信息，适合于诸如设计、预测、监控、诊断等类问题的求解. 主要缺点是推理时无明确的目标，求解问题时可能要执行许多与解无关的操作，导致推理的效率较低.

(2)反向推理

反向推理又称为目标驱动控制策略或自顶向下推理、目标推理、后件推理等. 其推理过程刚好与正向推理相反，它是首先提出某个假设，然后寻找支持该假设的证据，若所需的证据都能找到，说明原假设是正确的；若无论如何都找不到所需要的证据，则说明原假设不成立，此时需要另作新的假设. 具体地说，其推理过程是：

1) 首先提出假设 G.

2) 扫描知识库，找出那些其后件可与该假设匹配的规则，构成规则集 S.

3) 若 S 为空，即知识库中不存在可与该假设匹配的规则，则向用户询问此假设是否为真.

4) 若 S 不空，则从 S 中选出一条规则，并将其条件部分的每一个子条件都作为新的假设，且对每一个这样的新假设都重复 2) 至 4)的过程.

5) 如果一条规则的条件部分是多个子条件的合取，则只有当每一个子条件都被满足时，才说明相应假设是成立的. 若条件部分是多个子条件的析取，则只要有一个子条件得到满足就说明假设是成立的.

6) 如果不能证明某假设成立，则需另外提出假设，重复上述步骤. 该过程可用如下算法描述：

Precedure goal-driven(G)

扫描知识库，找出能导出假设 G 的规则集 S；

if S 空

then 向用户询问关于 G 的信息

else while G 未知且 S 非空 do

begin

从 S 中选出某一个规则 R；

把 R 的条件部分→G'；

if G' 未知

then 调用 goal-driven(G')；

if G' 为真

then 执行 R 的结论部分，并从 S 中删去 R

end

反向推理的主要优点是不必使用与总目标无关的规则，且有利于向用户提供解释．其主要缺点是要求提出的假设要尽量符合实际，否则就要多次提出假设，也会影响求解的效率．

(3) 混合推理

正向推理的主要缺点是推理具有盲目性，效率较低，推理过程中可能要推出许多与问题无关的子目标；反向推理的主要缺点是若提出的假设具有盲目性，也会降低问题求解的效率．为解决这些问题，可使用正向、反向混合推理．另外，在下述几种情况下，通常也需要运用混合推理：

1) 已知的事实不充分．数据库中的已知事实不够充分，若用这些事实与规则的条件部分进行匹配，可能没有一条规则可匹配成功，这就会使得推理无法进行下去．此时，可把其条件部分不能完全匹配的规则都找出来，并把这些规则的结论作为假设，然后分别对这些假设进行反向推理．由于在反向推理中可以向用户询问有关的论证，这就有可能使推理进行下去．在此推理过程中，先通过正向推理形成假设，然后通过反向推理证实假设的真假，这就是一种正向、反向混合推理．

2) 由正向推理推出的结论可信度不高．用正向推理进行推理时，虽然提出了结论，但可信度可能不高，甚至低于规定的阈值．此时可选择几个可信度相对较高的结论作为假设，然后进行反向推理，通过向用户询问进一步的信息，有可能得出可信度较高的

结论.

3）希望得出更高的结论. 在反向推理中，由于要与用户进行对话，这就会获得许多原来不掌握的信息，这些信息不仅可用于证实要证明的假设，同时，还可能推出其它结论. 这时可通过使用正向推理，充分利用这些新获得的证据推出另外一些结论. 例如在故障诊断中，先用反向推理证实了设备有某种故障，然后利用反向推理中获取的信息再进行正向推理，有可能推出该设备还有别的什么故障.

4）希望从正、反两个方向同时进行推理. 有时希望从正、反两个方向同时进行推理，即根据问题的初始证据进行正向推理，同时由假设的结论进行反向推理，当两个方向的推理在某处“碰头”时，则推理结束. 此时原先的假设就是问题的解. 这里所谓“碰头”是指由正向推理推出的中间结论恰好是反向推理到这一步时所需要的证据. 用这种方式进行推理时，困难的是“碰头”的判断问题，其时机不易掌握.

正向、反向混合推理的思想可大致地描述如下：

Procedure alternate

repeat

调用 date-driven，根据用户提供的初始证据推出部分目标；

根据这些目标做出对总目标的假设 G；

调用 goal-driven，确定 G 的真假

until 问题被求解

end

1.6.2 基于知识的诊断推理

非智能化的传统诊断技术在实际应用中表现了如下方面的不足：

1）非智能化诊断的多数方法是采用单一信息来源，而放弃了其它来源的信息，这对于复杂系统或并发性的故障诊断是十分不利的.

2）非智能化诊断方法不具备表达和综合处理多层次、多因素、多形式以及定性与定量等复杂知识体系的能力.

3）数据、知识和控制不能相互分离，新的经验、成果不能灵活地加入，知识的继承性和发展受到限制.

4）对诊断结果的解释性、交互性不佳.

基于知识的诊断推理克服了非智能诊断中知识表示处理、继承与扩充的局限性以及诊断系统的弱解释性，使设备诊断进入了一个崭新的智能化诊断发展阶段.

基于知识的诊断推理，可分为三类，即利用浅知识的诊断推理；利用深知识的诊断推理和利用深、浅知识的诊断推理.

1. 基于浅知识的诊断推理

基于浅知识的诊断推理实际上是这样一个问题，即已知一组征兆，要求对产生这组征兆的原因作出解释. 这类问题的求解，需要用到两类知识：一类是表示系统故障是如何引起各种征兆的因果性知识，另一类是反映因果关系的成立程度(模糊强度)和可能性(概率强度)方面的知识. 利用浅知识诊断方法的特点是：

1）浅知识通常以 IF-THEN 类型的规则形式表现，这是一种很容易读的形式，也是一种即使不熟悉计算机的人也容易理解的记述方法. 另外，知识的变更非常容易.

2）诊断的推理方法比较简单，而且可以利用医疗诊断领域开发的丰富的技术.

3）其诊断能力主要取决于所使用知识的质和量，一般要收集处理所有异常的知识是困难的.

以前的大多数故障诊断专家系统(属于第一代专家系统)都是采用浅知识推理，也就是根据已知的征兆，通过故障与征兆之间的因果关系，采用外延推理求取最为可能的故障假设. 由于第一代故障诊断专家系统的知识库容易构造和管理，推理效率高，抛开了客观世界内部的许多因果联系，集中体现了专家经验性知识等优点，因而赢得了众多专家系统使用者的青睐. 但是，由于浅知识模型的

限制，诊断精度也常常受到限制.诊断时往往是根据某些表面上出现的征兆，因此诊断知识的获取也受到限制.同时，由于只使用浅知识这一层次的知识，无法得到更为深刻的诊断结果和解释过程.还有从专家那里获取经验较难，知识集不完备，对没有考虑到的问题，诊断系统容易陷入困境.

2. 基于深知识的诊断推理

为了克服第一代专家系统基于浅知识推理的缺点，人们在专家系统中引入深知识推理的概念.所谓深知识目前还没有一个明确的定义，大致可概括为

1）结构知识：这是诊断知识中最低层次的知识，它反映的是诊断对象的各级组成元素及它们之间的相互关系.

2）功能知识：这种知识主要描述诊断对象的功能单元及各功能单元之间的功能关系.

3）因果知识：这种知识用因果网络描述诊断对象各单元之间故障及征兆之间的因果关系和传播途径.

4）诊断对象的数学或模拟模型以及支配这些模型的数学或物理定律：这类知识一般包括定性知识和定量知识两个方面.

由于深知识刻划了专门领域内原理性和功能性的知识，更深刻地了解领域对象和对象之间的相互作用，所以在遇到未料到的新情况时，能根据丰富的定性知识来解决问题，至少不至于使推理完全失败.另外，它能给人们提供更为令人信服的解释.

基于深知识的诊断推理属于第二代诊断专家系统，其特点是：

1）利用深知识的诊断方法收集的知识比较容易，诊断能力也高.如果给机械设备装置以适当的记述形式，它可以比规则法有更充分的知识表现，从而增加了知识的利用深度.

2）如果给出诊断对象的构造记述，立即就可以进行故障的诊断，这很适合机械设备种类多样化的特点，机械设备种类繁多，但其构造大多数是可以知道的，而要获得规则性知识则要积累很多经验，特别是对于新的机械设备，缺乏经验性知识，只利用深知识

的方法也可以进行故障诊断.

3）利用深知识诊断方法的缺点是：其推理处理复杂，搜索空间大，处理速度慢.

3. 基于深浅知识的诊断推理

人类专家在进行故障诊断时，不仅使用经验性知识，同时也利用机器的构造和性能描述知识，这就是深、浅知识相结合的诊断方法.综合利用深、浅知识诊断时，有可能达到人类专家诊断过程相近的诊断，从而使专家系统的智能化水平也前进了一步.

首先采用浅知识推理形成诊断焦点，再使用深知识进行确认，然后产生精确的解释.浅知识是基于对象和规则表示专家经验，而深知识是以因果网络的方式来深刻描述系统的结构、性能等知识.两层知识互相协作，弥补彼此的不足，获得效益和可靠性的统一.图 1.15 描述了深浅知识混合的专家系统的结构框图.

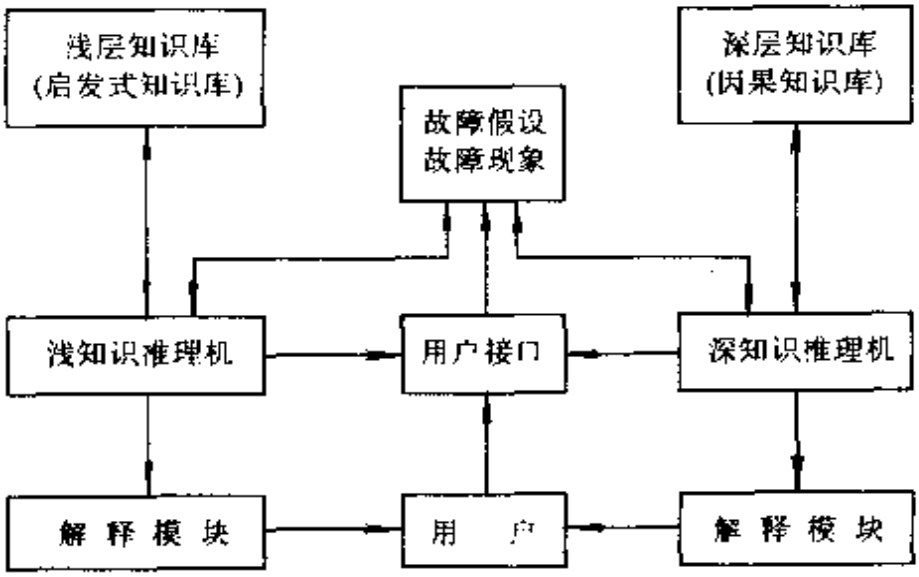


图 1.15 混合知识专家系统结构框图

由图可以看出，该专家系统事实上是两部分组成的，浅知识部分和深知识部分，它们各有自己的知识库和推理机，是可以进行独立工作的.在两者之间有一个叫做黑板的公共数据交换区.在该交换区中，保存着各种表征故障设备工作状态的数据和一些故障

现象以及故障假设，这些可被两部分共享. 浅知识层中每个故障假设与深层因果网络中的某个故障假设结点是相对应的，即每个表示故障假设的对象在因果网络中都有一个结点与之对应.

在推理过程中，首先启动浅知识进行推理，如果成功地找出故障源，则给出解释并停机，否则产生一些最有可能的故障假设. 然后启动因果网络层的知识进行深知识推理来确认这些假设并产生解释.

深知识推理，首先从故障假设结点开始，对因果网络的有关部分进行搜索. 在搜索过程中，搜索到相应的结点，该结点首先被例化成包括实际数据的对象. 如果一个假设被确认，首先产生一个完整精确的解释. 对于未被考虑的数据和未预料到的数据给出建议、假设，这些假设或者送回浅知识层继续推理，或者输出给用户来决定. 如果最初假设被否定，则产生另外可选择的故障假设，送回浅知识层继续推理，并把控制权交回给浅知识层推理机，最后直到成功或失败.

简言之，混合知识诊断专家系统的诊断过程首先由浅层推理产生初始诊断假设，再由深层诊断进行确认和解释. 两层之间的通讯是通过浅层中的假设对象与深层中的一个假设结点相对应，当浅层中产生一个故障假设对象后，深层推理则与之相对应网络结点开始推理. 也就是浅层知识(基于启发式的知识)推理用于产生诊断焦点，而深层知识则用于对诊断假设进行确认，具体化以及提出精确的解释或者推翻故障假设.

总之，混合知识诊断专家系统使得基于专家经验的故障诊断和基于诊断对象结构、功能的诊断解释有机地结合起来，取长补短，相得益彰，从而大大提高了诊断系统的工作效率和诊断的可靠性.

第二章　征兆获取和诊断求解

2.1　故障征兆的自动获取

诊断问题可以这样来描述：当诊断对象发生某种或某些故障时，其输出或行为将与正常状态时不同，诊断的任务就是寻找引起这些异常征兆的可能原因，即查明诊断对象可能发生了什么故障. 然而，征兆的自动获取是所有的智能化故障诊断系统必须解决的首要问题，只有解决了这一问题，才能真正实现诊断的智能化、自动化. 传统的基于专家系统的故障诊断在实际应用中所遇到的最大障碍就是诊断征兆必须通过人机交互的方式获取，为了获得最终的诊断结果，操作者需要回答几十个甚至上百个专业性极强的问题，这对操作者的知识水平提出了极高的要求. 因为有些问题普通的操作者根本不知如何回答. 这种人机交互的征兆获取方式已成为推广智能化诊断系统的一个不可忽略的问题.

对于故障诊断系统而言，征兆的形式主要表现为以下三种：一是数值型征兆，如频谱能量分布、油温、油压等；二是语义型征兆，如“振动增大”、“转子偏心变化不大”等；三是图形征兆，如“轴心轨迹呈椭圆形”、“波形畸变”等. 对于数值型征兆而言，可以直接根据传感器的信息自动获取. 而对于语义型征兆和图形征兆，自动获取的难度就相当大，有些征兆甚至不能做到自动获取. 尽管如此，尽量减少不能自动获取的征兆数目，对于智能诊断系统的推广应用仍然具有重要意义.

目前，征兆自动获取还没有一个系统化的方法，尤其是对于语义型征兆和图形征兆的自动获取更是难点. 现有的诊断方法，通常都需要已知的标准故障样本，这些样本的来源主要是通过实验室中的实验台进行模拟获得. 然而现代化生产设备，尤其是大型机械

设备，结构和动力特性极其复杂且制造费用高昂，很难做出一台与真实设备类同的实验台，而只能模拟真实设备的某一关键部件，如转子实验台等，其环境、装置以及运行参数均经过了简化. 由于机械故障的信息环境在很多情况下基本上是一个不确定环境，即使对同一类乃至同一设备，在不同的生产环境、不同的安装条件下，其同一故障所反映出的信息也存在着差异，征兆与故障之间没有一一对应的因果关系. 而且严格地讲，实验室中获取的数据与真实机组运行中获取的数据不属于同一母体，不同母体的样本可比性差，用实验室获得的结果对工程实际的机械设备进行诊断，其可靠性必然会降低. 另外，实际中有许多故障也不能通过实验台进行模拟，因此，有必要在实际生产环境中针对机器设备征兆的自动获取问题进行研究，这对于促进智能故障诊断技术走向实用化具有重要意义.

2.1.1 数值型征兆的自动获取

数值型征兆直接来源于诊断系统的各类传感器所提供的数值信息. 从本质上讲，所有的故障征兆都可由传感器获得，但是由于在实际应用场合，传感器的安装不便或者考虑到减小诊断系统的规模和复杂性，目前直接由传感器采集的故障征兆还仅仅局限于有限的振动信息和部分工艺参数信息. 数值型征兆可分为时域征兆和频域征兆两种，下面分别进行讨论.

1. 时域征兆的自动获取

由于原始检测信号是随机过程，很难直接用于设备的工况监视和故障诊断，必须将原始信号转换为能反映故障征兆的特征参量，包括信号的均值、方差、自相关函数、峭度、散度以及由时间序列分析方法所得的模型参数、模型残差等. 时域征兆通常对故障不十分敏感，但是其计算已有固定的公式和快速算法，因此常用于一些实时性较强的工况识别任务，通过这些时域征兆参数迅速地判别当前设备所处的工况正常还是异常.

用时间序列分析方法研究机器系统的状态监测和故障诊断问题，只需提取反映系统当前时刻的总体平均势态信息的参数和当前时刻瞬时状态信息的变化率参数，就可对系统运行状态作出合理的表征.

表征系统总体平均势态的信息，主要是信号的能量、信息距离、散度、方差、相关矩阵、信号所建时序模型的参数和残差以及它们的有效组合等，定义为$\Sigma(k)$. $\Sigma(k)$反映系统运行状态相对初始设定状态的偏离程度. 这种偏离越大，$\Sigma(k)$也越大；超出一定范围，则认为系统当前状态已不属于初始设定的系统状态. 如果系统运行状态平稳，$\Sigma(k)$也相对稳定，变化不大. 因此，对缓变性状态或故障一直存在的过程，在$\Sigma(k)$的值的大小上能客观反映，而对冲击性、阶跃性的随机故障的起始点则反映迟钝且相对滞后.

反映系统瞬时状态信息变化参数，实质上是以前一时刻的状态信息来检测后一时刻的状态变化情况. 设$\Phi(k)$，$\Phi(k-1)$分别为k时刻、$(k-1)$时刻反映系统运行状态的状态信息参数，则$\Delta\Phi(k)=\Phi(k)-\Phi(k-1)$为$k$时刻的状态信息变化程度或变化率. 相对状态变化越大，$\Delta\Phi(k)$也越大，超过一定的变化阈值，则认为状态发生显著变化. 显然，这一参数不受工况变化的影响. 从$\Delta\Phi(k)$可知，对状态平稳的正常过程，$\Phi(k)$很小，接近于0，对冲击性变化，$\Delta\Phi(k)$迅速增大后很快变小. 因为系统很快又恢复了原来的状态；对于阶跃性状态变化，$\Delta\Phi(k)$增大后，很快衰减；而对状态变化缓慢过程以及一开始就处于异常的过程状态，$\Delta\Phi(k)$始终保持很小的值. 因而$\Delta\Phi(k)$对冲击性、阶跃性等随机变化的起始点反映灵敏，能迅速反映系统状态的突变过程，而对缓变过程、一直存在的异常状态和阶跃变化后的状态，不能正确客观地反映.

综上所述，只需提取这样性质的两类特征参数，就能全面表征系统的运行状态，对系统实行状态监测和故障诊断.

如何提取$\Delta\Phi(k)$和$\Sigma(k)$呢？

我们提取机器运行时的振动信号去建立时序模型AR(n)，其参数特征向量$(\varphi_1,\varphi_2,\cdots,\varphi_n,\sigma_a^2)^T$凝聚了动态信号的主要信息，能

体现系统的运行状态.因此,选择 AR(n)模型参数估计的变化率作为 $\Delta\Phi(k)$,而选择动态信号的方差作为 $\Sigma(k)$就能对系统运行状态的好坏作出评价.

AR(n)模型为

$$x_k=X^T(k)\Phi(k)+\alpha_k \tag{2.1.1}$$

式中

$\Phi(k)=(\varphi_1,\varphi_2,\cdots,\varphi_n)^T$ 为模型参数;

$X(k)=(x_{k-1},x_{k-2},\cdots,x_{k-n})^T$ 为样本序列;

α_k 为均值等于零的白噪声.

利用递推最小二乘法可得到 AR(n)模型的参数估计:

$$\left.\begin{aligned}\Phi(k)&=\Phi(k-1)+K(k)[x_k-X^T(k)\Phi(k-1)]\\K(k)&=P(k-1)X(k)[1+X^T(k)P(k-1)X(k)]^{-1}\\P(k)&=P(k-1)-K(k)X^T(k)P(k-1)\end{aligned}\right\} \tag{2.1.2}$$

式(2.1.2)具有深刻的意义,它表示模型参数的新估计 $\Phi(k)$等于原估计 $\Phi(k-1)$予以校正.校正项为 $K(k)[x_k-X^T(k)\Phi(k-1)]$,它是新数据 x_k 与新数据的估计 $X^T(k)\Phi(k-1)$之差的加权处理,加权系数为 $K(k)$,$K(k)$又称校正系数.

对 $\Delta\Phi(k)$可选取:

$$\Delta\Phi(k)=\|\Phi(k)-\Phi(k-1)\| \tag{2.1.3}$$

$$\Delta\Phi(k)=\|P(k)-P(k-1)\| \tag{2.1.4}$$

$$\Delta\Phi(k)=\Phi^T(k)P(k)\Phi(k)-\Phi^T(k-1)P(k-1)\Phi(k-1) \tag{2.1.5}$$

等等.其中 $\|\cdot\|$ 表示范数概念,可有多种算式.当然,也可构造或选取其它能反映系统性态的变化率的参数.

$\Sigma(k)$特征选择振动信号的方差为

$$\Sigma(k)=\sigma_k^2=\frac{1}{N}\sum_{k=1}^{N}(x_{k-1}-\hat{m})(x_{k-1}-\hat{m})^T \tag{2.1.6}$$

式中

$$\hat{m} = \frac{1}{N}\sum_{k=1}^{N} x_k \qquad (\hat{m} \text{ 为均值}) \tag{2.1.7}$$

方差的变化率 $\Delta\Sigma(k)$ 也可作为 $\Delta\Phi(k)$ 的特征参数. 这里选取不同类型参数是为了更有效、更多形式地表征系统性态，起到多参数评判的效果.

以 $\Delta\Phi(k)$ 和 $\Sigma(k)$ 来检验当前系统的状态，正常时 $\Delta\Sigma(k)$ 很小，接近于 0；$\Sigma(k)$ 较平稳，在保证此条件的基础上，采集振动信号进行适应性学习训练，确定 $\Sigma(k)$ 和 $\Delta\Phi(k)$ 的阈值：

$$\min\{\overline{\Sigma}(k) - 3\sigma_\Sigma, 0.8\Sigma(k)_{\min}\} \leqslant \Sigma(k) \leqslant \max\{\overline{\Sigma}(k) + 3\sigma_\Sigma, 1.2\Sigma(k)_{\max}\} \tag{2.1.8}$$

$$|\Delta\Phi(k)| \leqslant \max\{\overline{\Delta\Phi}(k) + 3\sigma_{\Delta\Phi}, 1.2\Delta\Phi(k)_{\max}\} \tag{2.1.9}$$

式中 $\overline{\Sigma}(k)$，$\overline{\Delta\Phi}(k)$ 表示 $\Sigma(k)$，$\Delta\Phi(k)$ 参数的均值，$\Sigma(k)_{\max}$，$\Sigma(k)_{\min}$，$\Delta\Phi(k)_{\max}$ 为 $\Sigma(k)$，$\Delta\Phi(k)$ 参数的最大值和最小值；σ_Σ，$\sigma_{\Delta\Phi}$ 为 $\Sigma(k)$，$\Delta\Phi(k)$ 参数的均方差；$\overline{\Sigma}(k) \pm 3\sigma_\Sigma$，$\overline{\Delta\Phi}(k) + 3\sigma_{\Delta\Phi}$ 表示 97.5% 的置信区间；0.8，1.2 的加权表示 $\Sigma(k)$，$\Delta\Phi(k)$ 的最小、最大值的 80% 与 120%.

在一般情况下，对机器系统的运行状态监测，只需在线（也可离线）计算 $\Sigma(k)$，$\Delta\Phi(k)$，与假定的阈值比较，若满足式(2.1.8)和式(2.1.9)，则表明运行正常，反之，则有异常或故障出现.

本方法获取时域征兆参数的优点是所选的两类特征参数能迅速、有效地反映机器运行过程的状态变化和状态异常，并满足机器运行状态监测的需要.

2. 频域征兆的自动获取

频域征兆是故障诊断中应用最为广泛的一类故障特征，尤其是在对旋转机械的监测和诊断方面，频谱分析技术的应用更加广泛而有效，这主要表现在两个方面：一方面理论工作者对旋转机械的振动机理做了越来越深入的研究，为更好地运用频谱分析技术提供了理论基础；另一方面工程技术专家在实践中不断总结经验，将各种频谱图与相应的故障原因及可能出现的频率分布进行统

计、归纳、对比，为后人分析频谱图提供了较为有效的手段. 因为机械设备的运行状态及故障有其特定的征兆，反映在振动功率谱中，则是某些特定的谱峰. 如果将大量典型的振动信号频谱值和故障信号频谱值以一定的表格形式存放在计算机中，构成诊断用频谱数据库. 那么通过谱峰的寻找对比，由其高度变化和各种故障原因可能出现的频率分布概率，便可得出相应的诊断结论. 但要得到数据库的对应关系，则有赖于进行大量的模拟实验. 这往往需要付出很大的代价，有时甚至是不可能做到的. 因此，提出另外一种做法，就是在机器正常运行状态下采集一组时域信号，通过傅里叶变换成频域信号，给出正常运行状态的功率谱的极限指标，一旦超过此极限指标时，则将机器判定为异常运行状态. 整个步骤如下：

1) 将传感器采集的时域信号 $\{x_t\}(t=0,1,\cdots,N-1)$ 经过傅里叶变换或进行建模得到傅里叶功率谱和时序模型谱，傅里叶功率谱由下式求出：

$$S_x(\omega)=\frac{1}{2\pi N}\left|\sum_{t=0}^{N-1}x_te^{-j\omega t}\right|^2=\frac{1}{2\pi N}|X(\omega)|^2,\quad -\pi\leqslant\omega\leqslant\pi \tag{2.1.10}$$

其中 $X(\omega)$ 是信号序列 $\{x_t\}$ 的快速傅里叶变换(FFT).

时序模型谱由下式求出：

$$S_x(\omega)=\frac{\sigma_a^2\Delta}{\left|1-\sum_{k=1}^{n}\varphi_ke^{-j\omega k\Delta}\right|^2},\quad -\frac{\pi}{\Delta}\leqslant\omega\leqslant\frac{\pi}{\Delta} \tag{2.1.11}$$

其中 $\varphi_1,\varphi_2,\cdots,\varphi_n$ 及 σ_a^2 为自回归模型 AR(n) 的参数估计值及方差；Δ 为采样间隔.

2) 将功率谱等分为 n 个区域，其中心频率为 $\omega_1,\omega_2,\cdots,\omega_n$，相应的每一矩形面积内的平均功率为 $p_1,p_2,\cdots,p_n$，这样，这张功率谱可以用一个 n 维向量表示，如图 2.1 所示.

$$p=(p_1,p_2,\cdots,p_n)^T \tag{2.1.12}$$

3) 以上划分的结果，往往需要用 50—100 维的向量表示一张功率谱，并且向量中的每个元素都不是线性独立的. 为了简化计算

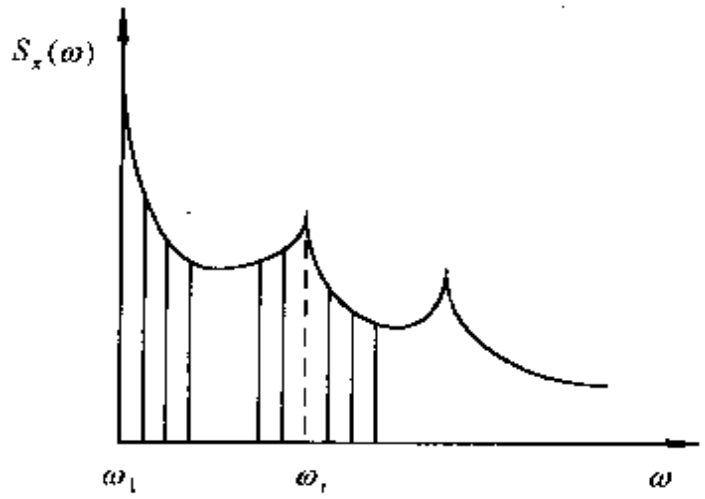


图 2.1　用 n 维向量表示功率谱

机运算和监视过程，需要进行特征抽取，同时将原来的 n 个元素，用新的 p 个元素组成的向量来代替：

$$y = (y_1, y_2, \cdots, y_p),\ p \ll n \tag{2.1.13}$$

特征抽取的基本方法是主成分分析.

4）决定极限指标 a_i, b_i. 设向量 y 的元素 y_i 具有概率密度 $p_i(y_i)$，则有

$$\left.\begin{aligned} p(y_i \leqslant a_i) &= \int_{-\infty}^{a_i} p_i(y_i)dy_i \\ p(y_i \geqslant b_i) &= \int_{b_i}^{\infty} p_i(y_i)dy_i \end{aligned}\right\} \tag{2.1.14}$$

如果 y_i 的任一观察值超过 (a_i, b_i) 的界限之外，则认为机组是处在不正常状态. 给定 $p(y_i \leqslant a_i)$ 和 $p(y_i \geqslant b_i)$ 之值，用数值积分的方法对式(2.1.14)求解 a_i 和 b_i 的值，即是要求的极限指标.

5）概率密度函数 $p_i(y_i)$ 的估计. 由于 $p_i(y_i)$ 是单独进行估计的，我们用 $p(y)$ 来表示任意向量元素 y_i 的概率密度函数. 设 y_i 的 N 个观察值在 0 与 y_{max} 间变化，则可将 $(0, y_{max})$ 区间分为 I 个等分，作出图 2.2 所示的直方图，极限指标 a 与 b 可直接由直方图确定. 这种方法称为直方图近似法.

另外还有一些提取频域征兆的简捷方法，比如在振动功率谱

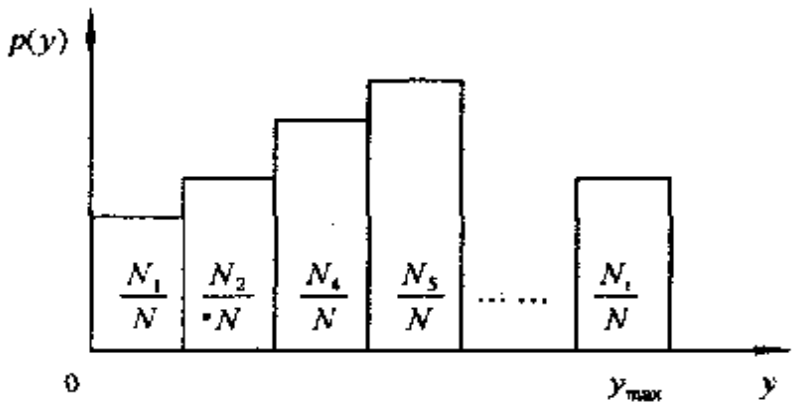


图 2.2　直方图近似法

或幅值谱图上，设立若干个表征设备故障特性的谱窗(频段)，以这些窗内的振动功率或最大谱峰值占全部频段的总功率或全部频段谱峰值之和的比值作为特征参数.从而，频谱特征的自动获取可以由下面的公式实现：

$$\mu(x_i)=\frac{x_i}{\sum_{j=1}^{n}x_j},\quad i=1,2,\cdots,n \tag{2.1.15}$$

式中 x_i,x_j 分别为第 i 个和第 j 个特征频段的振动功率或最大振动幅值，n 为特征频段数，$\mu(x_i)$为第 i 个特征频段的特征值.

3. 趋势征兆的自动获取

常见的趋势征兆，如振幅增大、急剧增大、瓦温升高等等，由于采用了自然语言进行描述，使得其具有很大的模糊性，与此同时，这些实际测量所得的征兆量又具有随机性的特点，往往表现为一个确定性的趋势与随机性变化的相互叠加，为此，这里采用了时间序列分析中的趋势项提取技术以及模糊隶属度的概念来获取趋势征兆.

设一趋势征兆量为某一测点的振幅、温度等的时间序列$\{x_t,t=1,2,\cdots,N\}$，则

$$x_t=d_t+\varepsilon_t \tag{2.1.16}$$

式中$\{d_t\}$为趋势项，它是随时间 t 变化的某一确定函数，如线性函

数、指数函数、周期函数等；$\{\varepsilon_t\}$为随机项，它反映了从$\{x_t\}$中提取$\{d_t\}$后剩下的随机波动成分.

通常故障诊断中应用的趋势征兆主要是线性趋势，因此这里仅介绍线性趋势征兆的自动获取方法，此时趋势项d_t为时间t的线性函数：

$$d_t = \beta_0 + \beta_1 t \tag{2.1.17}$$

它反映了时间序列$\{x_t\}$在截距为β_0，斜率为β_1的直线附近随机变化，显然，当$\beta_1 > 0$时，x_t随时间t逐渐增大，β_1的大小反映了征兆增大的快慢程度.

事实上式(2.1.17)即为数理统计中的一元线性回归问题，只不过此时的自变量为时间t，在时间序列$\{x_t\}$中，t取整数值$(1,2,\cdots,N)$，为避免当数据量N很大时，在回归方程t中的取值过大，可取

$$t_0 = 1, \quad t_i = 1 + i/N, \quad i = 1,2,\cdots,N \tag{2.1.18}$$

即

$$d_t = \beta_0 + \beta_1(1 + t/N) \tag{2.1.19}$$

则根据最小二乘法，可求得β_0,β_1的最小二乘估计式为

$$\begin{cases} \hat{\beta}_0 = \mu_x - \hat{\beta}_1 \bar{t} \\ \hat{\beta}_1 = \dfrac{\sum\limits_{i=1}^{N}(x_i - \mu_x)(t_i - \bar{t})}{\sum\limits_{i=1}^{N}(t_i - \bar{t})^2} \end{cases} \tag{2.1.20}$$

式中$\mu_x = \dfrac{1}{N}\sum\limits_{i=1}^{N} x_i$，$\bar{t} = \dfrac{1}{N}\sum\limits_{i=1}^{N} t_i = \dfrac{1}{2}\left(3 + \dfrac{1}{N}\right)$分别为$\{x_i\}$和$\{t_i\}$的均值，从而去除趋势项后，$\{x_t\}$中的随机项为

$$\varepsilon_t = x_t - d_t = \hat{\beta}_0 - \hat{\beta}_1(1 + t/N), \quad i = 1,2,\cdots,N \tag{2.1.21}$$

获得趋势项和随机项后就可以根据下面的方法提取趋势征兆：

对斜率进行分段，令$0 < k_1 < k_2 < k_3 < k_4 < +\infty$，表示征兆基本不变的斜率范围为$[0,k_1]$，表示征兆有增大趋势的斜率范围为

$[k_2, k_3]$，表示趋势急剧增大的斜率范围为$[k_4, +\infty]$，则对于实际估计所得的斜率β_1可以分别按下式求得属于上述三个模糊语义论域的隶属度为

$$基本不变=\begin{cases}1, & \beta_1 \leqslant k_1\\ \dfrac{\beta_1-k_2}{k_1-k_2}, & k \leqslant \beta_1 \leqslant k_2\end{cases} \tag{2.1.22}$$

$$增大=\begin{cases}\dfrac{\beta_1-k_1}{k_1-k_2}, & k_1 \leqslant \beta_1 \leqslant k_2\\ 1, & k_2 \leqslant \beta_1 \leqslant k_3\\ \dfrac{\beta_1-k_4}{k_3-k_4}, & k_3 \leqslant \beta_1 \leqslant k_4\end{cases} \tag{2.1.23}$$

$$急剧增大=\begin{cases}\dfrac{\beta_1-k_3}{k_4-k_3}, & k_3 \leqslant \beta_1 \leqslant k_4\\ 1, & \beta_1 \geqslant k_4\end{cases} \tag{2.1.24}$$

上述各式的隶属函数图形如图 2.3 所示.

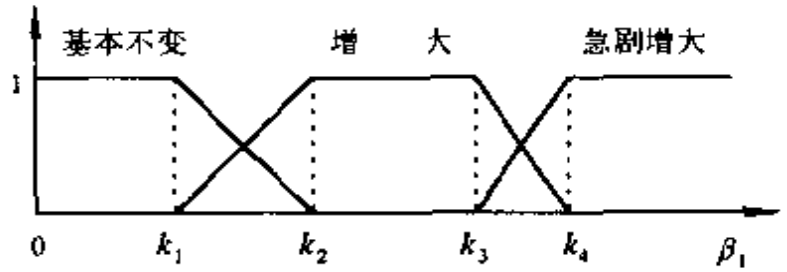


图 2.3 趋势征兆隶属函数

2.1.2 语义型征兆的自动获取

语义型征兆是指那些直接由人类自然语言进行描述的征兆信息. 语义型征兆产生的原因主要是由于机械设备的复杂性使得其故障形成原因和征兆的因果关系错综复杂，难以用测试手段将不同故障的信息分离开来，征兆与故障之间根本无法建立明确的数学模型，而只能采用诸如设备运行时“振动太大”、“转速不稳”等模糊不清的自然语言来说明故障的特征. 语义型征兆是机械故障诊断中一类重要的征兆信息，事实上，即使是能够通过传感器精确测

量的数值征兆，当它应用于诊断时，仍然必须根据征兆与故障的模糊语义关系转变为数值化的语义信息，如用模糊隶属度表示“振动太大”的程度.

语义型征兆的自动获取是指对具有具体量值的征兆，根据其测量值，转化为对应语义论域上的模糊语义量. 对于纯粹的没有量值的语义征兆或虽有量值但不能通过传感器获得其测量值的语义征兆，其获取只能通过人机交互的方式. 语义征兆获取事实上是把自然语义值表示为[0,1]上的数值形式以供计算机处理.

对于可测得量值的语义征兆，如温度、压力过高或过低等，若其具有三类语义量：过高、过低、正常，且该征兆的可能量值落在区间[min,max]，正常量值落在区间$[a,b]$，($[a,b]\subset[\min,\max]$)，则各个语义量的隶属函数可通过图 2.3 所示的梯形隶属函数表示，即

$$\text{过高}=\begin{cases}1, & x\geqslant \max\\ \dfrac{x-b}{\max-b}, & b\leqslant x\leqslant \max\end{cases} \tag{2.1.25}$$

$$\text{正常}=\begin{cases}\dfrac{x-\max}{b-\max}, & b\leqslant x\leqslant \max\\ 1, & a\leqslant x\leqslant b\\ \dfrac{x-\min}{a-\min}, & \min\leqslant x\leqslant a\end{cases} \tag{2.1.26}$$

$$\text{过低}=\begin{cases}\dfrac{x-a}{\min-a}, & \min\leqslant x\leqslant a\\ 1, & x\leqslant \min\end{cases} \tag{2.1.27}$$

如果语义特征量没有具体的量值，则其隶属度可按如下方式量化：

$$\begin{aligned}&\text{过高}=\{0.05/\text{过低},\ 0.5/\text{正常},0.9/\text{过高}\}\\ &\text{正常}=\{0.5/\text{过低},\ 0.9/\text{正常},0.5/\text{过高}\}\\ &\text{过低}=\{0.9/\text{过低},\ 0.5/\text{正常},0.05/\text{过高}\}\end{aligned} \tag{2.1.28}$$

2.1.3 图形征兆的自动获取

图形征兆是指那些以图形方式表示出来的故障特征. 比如轴

心轨迹的形状就是故障诊断中最为重要的一类图形征兆.虽然图形征兆所蕴含的诊断信息从理论上讲与构造这幅图形所用的数据信息是冗余的,但图形征兆能够更加有效地反映出故障特征,因而在诊断系统中得到广泛的应用.

下面着重讨论应用神经网络对图形征兆进行自动识别的基本原理,其主要步骤是:

1) 提取图形的不变特征.这种不变性有三层含义,即旋转不变性、平移不变性及刻度因子不变性.具体实现是先将图形规格化以获得平移不变性和刻度因子不变性,再对规格化后的图形提取 Zernike 矩特征.Zernike 矩特征具有旋转不变性.

2) 将所提取的特征归一化后输入到神经网络中去学习,该网络是一个多层感知器,其中包含一个隐含层,学习中使用的是误差反向传播算法.

3) 待测图形的分类识别.先提取待测图形的不变性特征并归一化,然后作为输入,送至训练好的神经网络进行分类识别.

1. Zernike 矩特征的提取方法

(1) Zernike 矩概念

Zernike 矩是建立在单位圆 $x^2+y^2\leqslant 1$ 内部的一个完备正交集,我们设$\{V_{nm}(x,y)\}$表示阶数为 n,重复数为 m 的 Zernike 多项式,则其表达形式为

$$V_{nm}(x,y)=V_{nm}(\rho,\theta)=R_{nm}(\rho)\exp(jm\theta) \quad (2.1.29)$$

其中 n 为正整数或零;m 为正负整数,且需满足$n-|m|$为偶数和$|m|\leqslant n$的条件限制;ρ 为原点到点(x,y)的矢量长度;θ 是 x 轴与 ρ 矢量逆时针方向的夹角.

$R_{nm}(\rho)$表示点(x,y)的径向多项式,定义如下:

$$R_{nm}(\rho)=\sum_{s=0}^{(n-|m|)/2}\frac{(-1)^s[(n-s)!]\rho^{n-2s}}{s!\left(\frac{n+|m|}{2}-s\right)!\left(\frac{n-|m|}{2}-s\right)!} \quad (2.1.30)$$

注意到有：$R_{n(-m)}(\rho)=R_{nm}(\rho)$.

Zernike 多项式 $V_{nm}(x,y)$是单位圆内的正交多项式，它们满足以下关系式：

$$\iint_{x^2+y^2\leqslant 1}[V_{nm}^*(x,y)]V_{pq}(x,y)dxdy=\frac{\pi}{n+1}\delta_{np}\delta_{nq} \quad (2.1.31)$$

式中

$$\delta_{ab}=\begin{cases}1, & a=b\\ 0, & \text{其它}\end{cases}$$

Zernike 矩实质是图像函数正交的一个映射. 图像函数 $f(x,y)$的 n 阶重复数为 m 的 Zernike 矩 A_{nm}定义如下：

$$A_{nm}=\frac{n+1}{n}\iint_{x^2+y^2\leqslant 1}f(x,y)V_{nm}^*(\rho,\theta)dxdy \quad (2.1.32)$$

对于离散的数字化图像函数，则用求和代替积分：

$$A_{nm}=\frac{n+1}{n}\sum_x\sum_y f(x,y)V_{nm}^*(\rho,\theta) \quad (2.1.33)$$

计算 Zernike 矩时，是把图像矩心当作原点，而且将图像的象素坐标映射到单位圆 $x^2+y^2\leqslant 1$ 范围之内，而落在单位圆外的象素在计算中则不起作用. 这里注意到

$$A_{nm}^*=A_{n(-m)}$$

Zernike 矩具有简单的旋转特性，若 A_{nm}和 A_{nm}^*分别表示图像 $f(x,y)$和它逆时针旋转 φ 角度图像的 Zernike 矩，则它们之间有如下关系：

$$A'_{nm}=A_{nm}\exp(-jm\varphi) \quad (2.1.34)$$

由式(2.1.34)可知：旋转后图像 Zernike 矩仅产生一个相位移动$(-m\varphi)$角，而矩的模值保持不变.

(2) Zernike 矩特征的提取

由以上分析可知，我们可以提取图像的 Zernike 矩模值$|A_{nm}|$作为其旋转不变特征，而且由于 $A_{n(-m)}=A_{nm}^*$，可以不考虑 $m<0$ 的情形.

为了满足特征的另外两个不变性，即平移和比例因子不变性，

应先将图像规格化.规格化过程分两步进行:首先平移变换图像,使其一阶几何矩 m_{01} 和 m_{10} 都为零.具体变换是将 $f(x,y)$ 变为 $g_1(x,y)$,公式如下:

$$g_1(x,y) = f(x + x_1, y + y_1) \tag{2.1.35}$$

其中 (x_1,y_1) 指原图像的矩心坐标,计算方法是:

$$x_1 = \frac{m_{10}}{m_{00}}, \quad y_1 = \frac{m_{01}}{m_{00}} \tag{2.1.36}$$

其中 m_{00},m_{10},m_{01} 分别是原图像 $f(x,y)$ 的零阶及一阶几何矩.$f(x,y)$ 的 $p+q$ 阶的几何矩定义式为

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y) \tag{2.1.37}$$

平移完成后,便对图像 $g_1(x,y)$ 进行比例变换 $g_1(x,y)$ 至 $g(x,y)$.事先确定一个目标象素总数 β(二维图像下的零阶几何矩实质即为图像中目标象素的总数),比例变换公式为

$$g(x,y) = g_1(x/a, y/a) \tag{2.1.38}$$

其中 a 为比例变换因子,计算公式为 $a = \left(\frac{\rho}{m_{00}}\right)^{1/2}$,$m_{00}$ 是图像 $g_1(x,y)$ 的零阶几何矩.

综上所述,图像 $f(x,y)$ 的不变性特征提取有以下几个步骤:

1) 将 $f(x,y)$ 映射到正方形区域 $x \in [-1,+1], y \in [-1,+1]$ 内;

2) 原图像规格化为重心在原点,目标象素总数为常数 β 的标准图像.原图像函数 $f(x,y)$ 经规格化后图像函数为 $g(x,y)$,它们之间的映射关系为

$$g(x,y) = f\left(\bar{x} + \frac{x}{a}, \bar{y} + \frac{y}{a}\right) \tag{2.1.39}$$

3) 对规格化后图像函数提取 Zernike 矩,取其模值为图像的特征.必须注意的是 Zernike 矩所有的计算都限于单位圆内,而且矩从二阶开始取,因为规格化之后的 A_{00} 及 A_{11} 都为常数.$\left(A_{00} = \frac{1}{\pi} m_{00} = \frac{1}{\pi}\beta,\ A_{11} = 0\right)$ 不能作为表征图像信息的特征.

2. 神经网络图形分类器的构成原理

(1) 神经网络图形分类器的结构

应用神经网络对图形进行分类或识别之前，首先有一个学习(训练)的过程，然后网络才能根据所学习的有关知识进行具体的识别. 这里采用的学习网络是多层感知器神经网络(见图 4.6)，包括输入层、隐含层和输出层. 如果我们输入是从图形中提取的 Zernike 不变性矩特征，例如为 12 阶矩，它有 47 个矩特征，则输入节点数为 47. 如我们要识别的图形为 26 个英文字母，则输出层节点数为 26. 在输入、隐含、输出各层中每一个节点都由一定权值连接着高一层的所有节点. 训练网络的过程即为相对于所有连线寻找合适的权值，使得对于相应输入得到的各节点输出尽量接近理想的输出.

(2) 神经网络的训练

神经网络训练采用传统的 BP 算法，即误差反向传播法. 其本质是梯度下降法，以实际输出与要求输出之差构成的误差函数对加权空间的连接强度加以修正，使误差函数不断下降. 算法的主要步骤如下：

1) 初始化. 预置所有连接权 w_{ij} 为[−1，+1]之间的随机值. w_{ij} 是节点 j 与低一层节点 i 之间的加权值.

2) 考虑第 m 类型的输入并规定理想的输入如下：相应的第 m 个节点的理想输出为 1，其余各节点期望输出为 0.

3) 利用当前的 w_{ij} 值计算所有节点的实际输出，用 y_j 表示节点 j 的输出值，它是一个关于所有输入的非线性函数：

$$y_j = \frac{1}{1 + \exp\left(-\sum_i y_i w_{ij}\right)}$$

此特定的非线性函数为 S 形函数.

4) 对于所有节点寻找误差 δ_j. 如果用 d_j, y_j 分别表示节点 j 的期望输出和理想输出，则对于一个输出节点 j，有

$$\delta_j = (d_j - y_j) y_j (1 - y_j)$$

对于一个隐含层节点 j，有

$$\delta_j = y_j(1 - y_j)\sum_k \delta_k w_{jk}$$

其中 k 是对于节点 j 的高一层中的所有节点求和.

5) 用下式修正权值

$$w_{ij}(n+1) = w_{ij}(n) + \alpha\delta_j y_j + \zeta[w_{ij}(n) - w_{ij}(n-1)]$$

其中$(n+1)$、(n)和$(n-1)$分别代表下一次、当前及上一次迭代权值的下标. 参量 α 是一个学习速率，类似于梯度搜索法的步长. ζ 是介于 0 和 1 之间的一个常数，它决定了在加权空间中上一次的加权变化对现在权值调整方向的影响，给出了一个惯性矩. 这样就有效地滤去了误差表面的高频变化.

6) 取另一个模式输入并回到步骤 2). 所有的训练输入模式是周期性提出，反复迭代训练直到权值收敛.

总之，以上算法是一个权值空间的梯度下降迭代过程. 最后是要使所有输出点的理想输出与实际输出的差值的平方和最小，以达到最好的分类效果.

3. 图形征兆的自动获取

根据上面论述的图形不变性特征的提取方法以及应用 BP 神经网络作为图形分类器的原理，文献[10]提出了一种基于 Zernike 矩特征的神经网络轴心轨迹图形的自动识别方法.

轴心轨迹的形状是旋转机械故障诊断中最为重要的一类图形征兆. 转子在轴承中高速度旋转时并不是只围绕自身中心旋转，而且还环绕某一中心作涡流运动. 产生涡流运动的原因可能是转子不平衡、对中不良、动静摩擦等，这种涡流运动的轨迹称为轴心轨迹，如图 2.4 所示.

轴心轨迹的获取是利用相互垂直的两个非接触式传感器分别安置于轴的某一截面上，同时刻采集或进入示波器，也称为李莎育图形. 通过分析轴心轨迹的运动方向与转轴的旋转方向，可以确定转轴的进动方向(正进动和反进动). 轴心轨迹在故障诊断中可

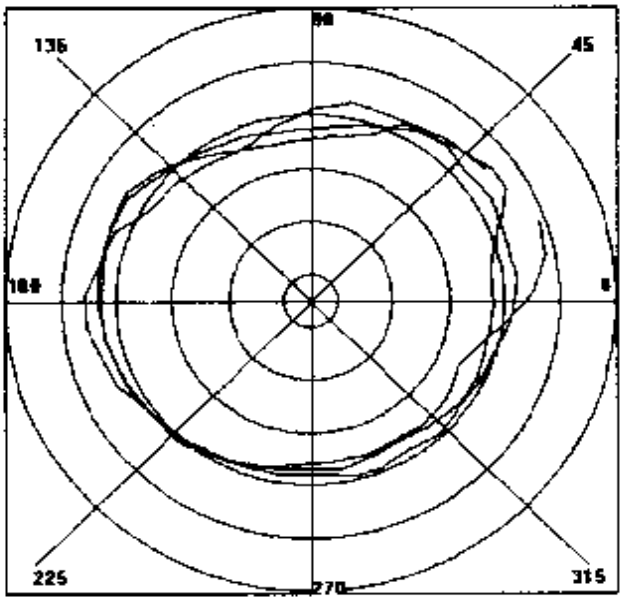


图 2.4　轴心轨迹图

用来确定转子的临界转速、空间振型曲线及部分故障，如不对中、摩擦、油膜振荡等.

轴心轨迹图形的自动识别方法包括实际设备各种轴心轨迹图形的采集；图形的二值化；图形 Zernike 矩特征的提取及归一化；BP 神经网络分类器的训练. 神经网络训练达到要求后便成为一个较为理想的分类器. 对待测图形分类前，也必须先将其二值化并提取不变矩特征，归一化后作为神经网络的输入，求出相应的输出，并寻找具有最大值的输出节点号赋予图形类别.

2.2　多故障诊断问题的求解

在第一章中所提到的故障诊断问题实际上是由征兆空间到故障空间的一种映射关系，这里所说的映射是一种多对多的复杂映射：多种征兆对应于一个故障，一种征兆同时也对应着多个故障. 例如在汽车发动机的故障诊断中，征兆表现为 13 种，如发动机冷却水温过高、发动机冷却水温过低、发动机油压过低、有焦糊味、有油糊味、发动机声音异常……. 故障表现为 9 种，如发动机过热、发

动机过冷、发动机内有烧结物、离合器打滑、发动机发滞……. 图 2.5表明了它们之间的对应关系.

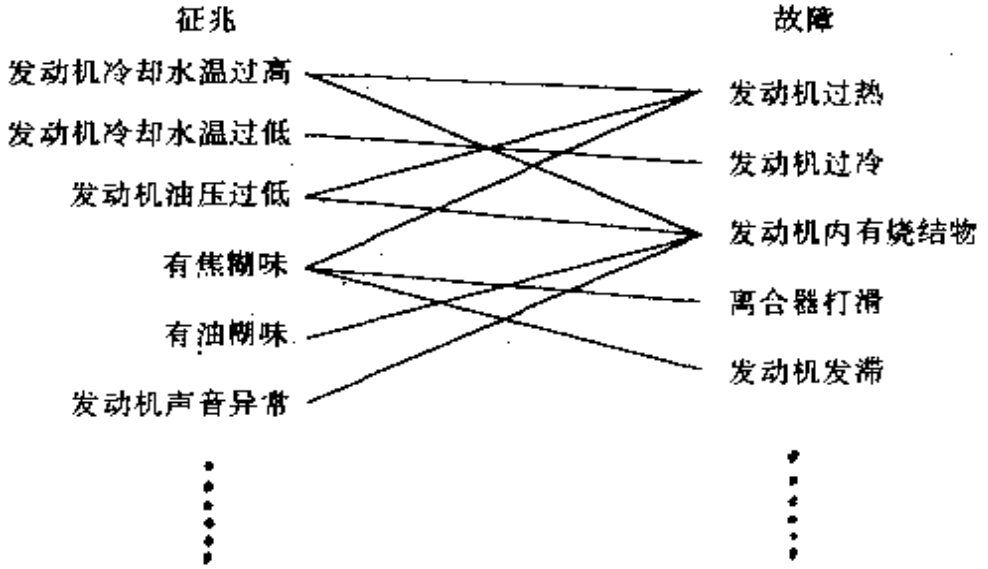


图 2.5 故障与征兆关系图

图 2.5 中某一对故障与征兆之间有直线连接表明这一对故障与征兆之间有联系,没有直线连接表明该故障是否出现与这个征兆是否出现无关. 这种故障与征兆的复杂关系常常作为经验知识保存在专家的头脑中,它们的具体表现形式从连接的方式上可以分为三类:1) 故障与征兆之间有着明确的 0,1 关系. 2) 故障与征兆之间关系明确,但是不是 0,1 关系,即存在一定的系数关系. 3) 故障与征兆之间存在错综复杂的非线性关系.

从获取征兆的次序来说,故障诊断可分为序贯性故障诊断和并发性故障诊断. 所谓序贯性故障诊断,是指诊断对象的征兆信息不是一次获得的,而是在诊断过程中逐步得到的;所谓并发性故障诊断是指诊断初期就已经获得诊断对象所表现出来的所有征兆信息.

2.2.1 并发性故障诊断问题求解

并发性故障诊断问题,也称为同时性故障诊断问题. 由于征兆在开始时已经全部获取,因此对于复杂设备的同时性故障诊断问

题，就是一次提供给诊断系统初始信息，系统按一定的策略寻求并给出合理的诊断解．求解过程可以概括为图 2.6 所示．

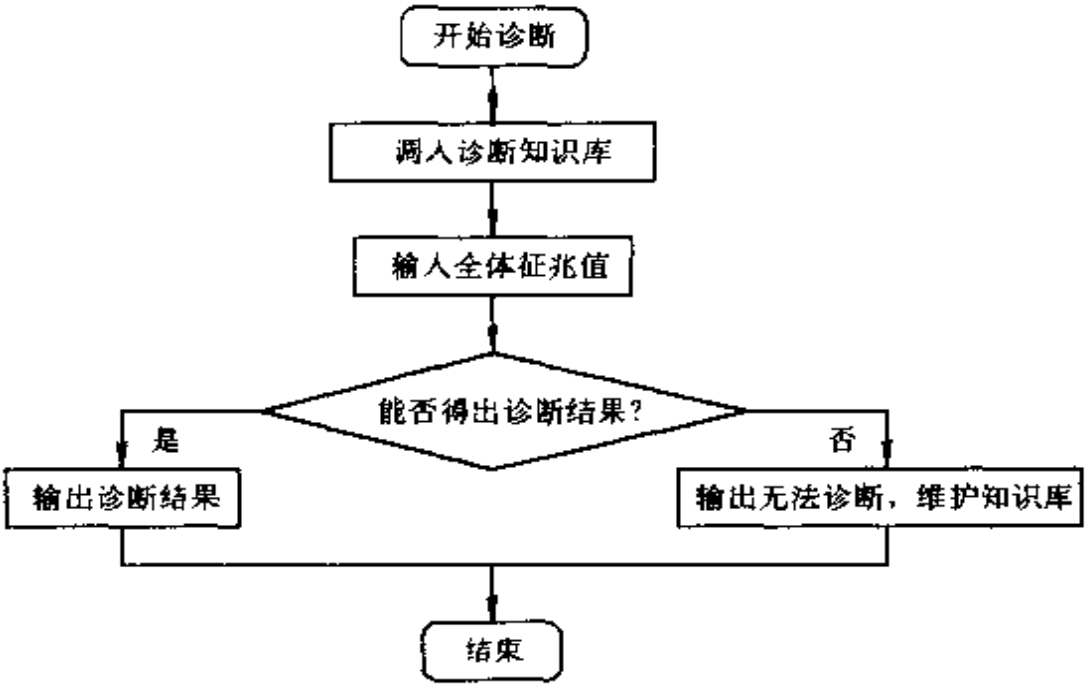


图 2.6　并发诊断求解示意图

在并发性故障诊断中，采用正向推理是显而易见的．同时，由于在诊断前必须获取全部的征兆信息，因此所付出的代价也就较高．但是，当诊断对象的故障与征兆具有第三种连接方式时，采用这种方式一般来说又是必须的．

在并发性故障诊断问题中，通常可以采用神经网络诊断方法和模糊诊断方法．

神经网络应用于诊断中主要是通过与诊断专家系统的结合来实现的．这主要是考虑到它们各自的优点．神经网络具有以下几个优点：求解问题可用连接模型表示；网络的连接权值可通过训练获得；连接模型具有鲁棒性；网络具有很强的容错能力；连接模型适合于硬件实现．而专家系统也存在许多诱人的特点：系统能在不确定、不完备的领域知识的条件下进行推理；系统可从外部世界获取知识，也可从内部推理过程中机械地学习知识；系统能对问题求解过程给出解释，使推理得出的结论令人信服．然而，它们也各自存

在缺陷，如神经网络存在网络连接模型表达的复杂性(如神经元个数选取、模型的可分性等)、网络训练过程的稳定性、训练时间过长、并行分布技术带来识别结果难于理解和解释等等，专家系统存在知识表示、知识获取、知识验证等问题. 因此，有必要构造基于神经网络的诊断专家系统，使之既能保持专家系统的原有特色，又兼有神经网络的特点，基于神经网络的故障诊断方法将在第四章中重点讨论.

模糊诊断方法是处理并发性故障诊断问题的另一种有效方法. 此时诊断对象的征兆和故障都表现出一定的隶属度，模糊诊断方法正是通过这些征兆的隶属度来求出各种故障的隶属度. 模糊诊断方法将在第五章中重点讨论.

2.2.2 序贯性故障诊断问题求解

对于实际的诊断问题，在大多数情况下不是并发性的，而是序贯形式的. 在对序贯性诊断问题进行求解过程中，人们首先可能只知道诊断对象的表层征兆信息，例如：医疗诊断中病人的脸色、体温、心跳次数等，至于稍微里面一些的征兆信息如血液中红、白细胞的数量等往往是随着诊断的进行来确定是否需要输入.

在序贯性诊断问题中，由于不知道全部的征兆信息，而是根据诊断的需要来讨论征兆量，因而在诊断中经常采用反向推理和混合双向推理. 这里结合具体的实例分别采用这两种推理方法来讨论序贯性诊断问题.

1. 采用反向推理的序贯性诊断问题求解

采用反向推理的基于规则的序贯性诊断大致可用流程图 2.7 来表示.

设某液压传动系统的换向阀故障诊断专家系统的知识库中有如下 17 条规则：

规则 1

如果　油缸完全不移动

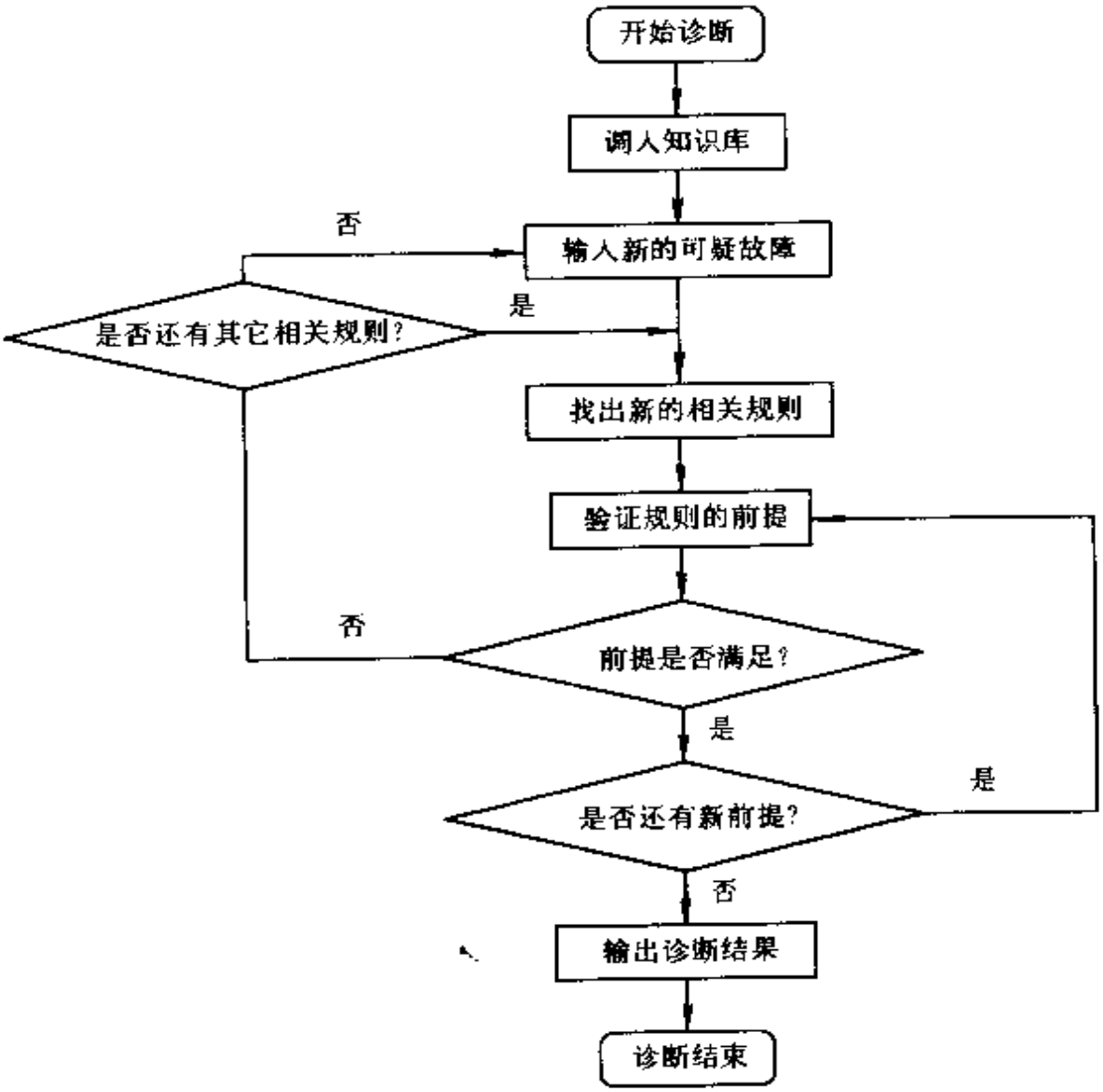


图 2.7　采用反向推理的基于规则的序贯诊断

则　　阀芯不移动

规则 2　　如果　油缸仅不能换向

则　　阀芯不能返回中心位置

规则 3

如果　弹簧是新换的

则　　弹簧应该不会是坏的

规则 4

如果　换向阀第一次使用

则　电磁铁应该不会是坏的

规则 5

如果　阀芯不能返回中心位置

则　失效原因在于弹簧

规则 6

如果　换向阀失效原因在于弹簧

换向阀一直工作正常

则　弹簧可能折断了

规则 7

如果　换向阀失效原因在于弹簧

弹簧处于正常工作位置

弹簧应该不会是坏的

则　弹簧刚度不够

规则 8

如果　换向阀失效原因在于弹簧

弹簧刚度足够

弹簧不会是坏的

则　弹簧可能漏装了

规则 9

如果　阀芯不移动

在油中发现有磨损的颗粒

则　阀芯可能被卡住了

规则 10

如果　阀芯不移动

在油中发现有其它杂质的颗粒

则　油源可能被污染了

规则 11

如果　阀芯不移动

电磁铁应该不会是坏的

工作中油温上升

油源是干净的

则　阀芯被卡住可能由于配合间隙小

规则 12

如果　阀芯不移动

电磁铁应该不会是坏的

工作中油温正常

油源是干净的

则　阀芯被卡住是由于液压卡紧力

规则 13

如果　阀芯不移动

弹簧应该不会是坏的

阀芯不能被卡住

该阀是电磁阀

则　阀芯失效原因在于电磁铁

规则 14

如果　阀芯失效原因在于电磁铁

电磁铁线圈有电

电磁铁应该不会是坏的

则　电磁铁推力不够

规则 15

如果　阀芯失效原因在于电磁铁

电磁铁线圈有电

电磁铁推力一定是足够的

则　电磁铁线圈可能损坏了

规则 16

如果　阀芯失效原因在于电磁铁

行程开关工作正常

电磁铁线圈没电

则　电磁铁通电线路可能短路

规则 17

如果　阀芯失效原因在于电磁铁
　　电磁铁通电线路正常
　　电磁铁线圈没电
则　　换向阀失效原因在行程开关失灵

设现在假设故障原因为通电线路短路，其反向推理过程为

1）输入怀疑故障：电磁铁通电线路短路；

2）调用规则 16，验证阀芯失效原因在于电磁铁.

3）调用规则 4，询问：换向阀是否第一次使用？接受用户响应：不是. 规则 4 匹配失败，返回规则 16.

4）调用规则 13，验证阀芯不移动.

5）调用规则 1，询问：油缸完全不移动？ 接受用户响应：是. 规则 1 匹配成功. 返回规则 13.

6）验证弹簧应该不会是坏的. 调用规则 3，询问：弹簧是否新换的？ 接受用户响应：是. 规则 3 匹配成功. 返回规则 13.

7）验证阀芯不能被卡住. 调用规则 9，询问：在油中是否有磨损的颗粒？ 接受用户响应：没有. 规则 9 匹配失败，从而，阀芯不能被卡住成立. 返回规则 13.

8）询问：该阀是否是电磁阀？接受用户响应：是. 至此，规则 13 匹配成功，得出阀芯失效原因在于电磁铁. 返回规则 16.

9）验证行程开关工作正常. 询问：行程开关是否正常？接受用户响应：正常.

10）验证电磁铁线圈没电. 询问：电磁铁线圈是否没电？ 接受用户响应：是.

11）至此，规则 16 匹配成功. 输出诊断结果：该换向阀的故障是：电磁铁通电线路短路.

2. 采用混合双向推理的序贯性诊断问题求解

混合双向推理是一种典型的序贯性诊断，其过程可分为三步：第一步，初诊阶段. 诊断系统获得诊断对象的部分征兆，并根据这些征兆进行诊断，提出可能的故障；第二步，逼近和排除阶段. 诊断

系统对第一步提出的可能故障逐一进行审查，如果诊断对象不具备某故障的其它征兆，就对其实施排除，如果诊断对象具有某故障的其它征兆，就认为该故障得到进一步的证实；第三步，确诊．不断地进行排除和逼近，直到某些故障的全部征兆都出现在诊断对象上，才认为诊断对象确实具有该故障．

根据上面的描述，可以得出序贯性诊断结构图 2.8：

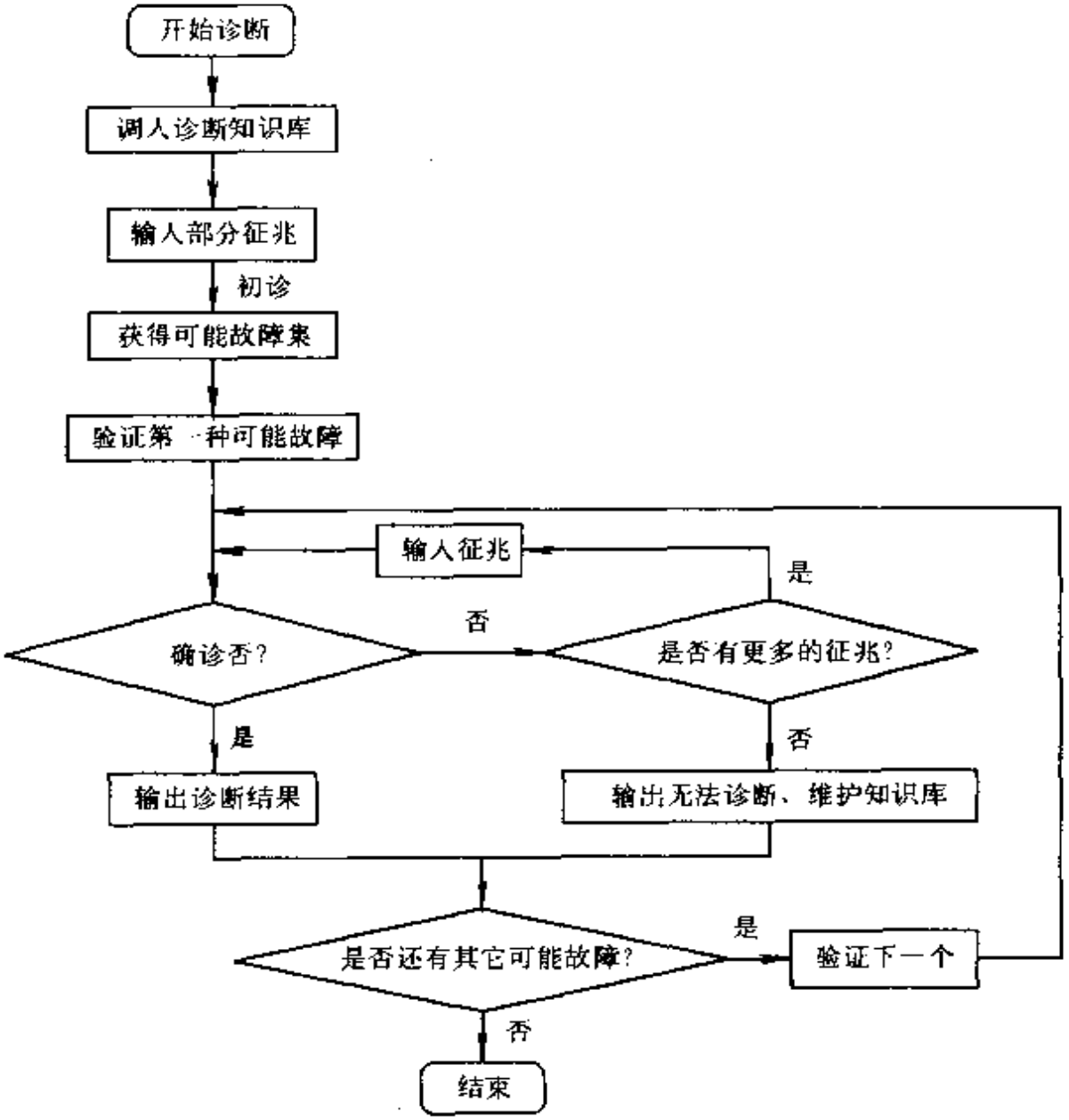


图 2.8　序贯性诊断结构图

在序贯性诊断问题中，由于不知道全部的征兆，为了说明这种诊断方法的可靠性，先探讨它的数学基础.

设某诊断对象的故障空间和征兆空间分别用集合 U 和 V 表示，即 U，V 分别表示故障库和征兆库. $P(U)$和 $Q(V)$分别表示 U 和 V 的任意子集的集合. A 为任意故障集，即 $A\in P(U)$，B 为任意征兆集，即 $B\in Q(V)$，$f(A)$表示故障集 A 的全体征兆集，$g(B)$表示征兆集 B 所对应的故障集. 即存在：

征兆映射　f：$P(U)\to Q(V)$

故障映射　g：$Q(V)\to P(U)$

领域专家在进行诊断时，一般这样认为：一个诊断对象如果存在某种故障，则这个诊断对象必然出现该故障所具有的全部征兆；一个有征兆的诊断对象，必然具有某些故障. 如果某些故障的征兆全部出现在诊断对象上，那么就认为该诊断对象确有这些故障.

这种诊断思维可用下面的数学语言进行描述.

定义 1　如果故障集 A 的征兆 $f(A)$都包含在诊断对象的征兆集 B 中，那么就认为诊断对象具有故障集 $g(B)$包含的故障集 A.

定义 1 可用式(2.2.1)表示：

$$\forall\ A\in P(U),\ \forall\ B\in Q(V),\ f(A)\subset B \text{ 当且仅当 } g(B)\supset A \tag{2.2.1}$$

在部分征兆的情况下，定义 1 可描述成定义 2 的形式.

定义 2　V 是征兆空间，$H\subset V$，称 H 为征兆子空间. 如果征兆映射 f_H：$P(U)\to Q(H)$，故障映射 g_H：$Q(H)\to P(U)$满足式(2.2.2)，即，$\forall\ A\in P(U)$，$\forall\ B\in Q(H)$

$$f_H(A)\subset B \text{ 当且仅当 } g_H(B)\supset A \tag{2.2.2}$$

则称 f_H，g_H 为部分征兆的诊断思维的数学模型.

我们可以证明下面的几个定理：

定理 1　f 和 g 满足式(2.2.1)，H 是征兆子空间，则 f_H，g_H 满足式(2.2.2)，其中，$\forall\ A\in P(U)$，$\forall\ B\in Q(H)$

$$f_H(A)\triangleq f(A)\cap H \tag{2.2.3}$$

$$g_H(B)\triangleq\bigcup\{A\mid f_H(A)\subset B\subset H,\forall\ A\in P(U)\} \tag{2.2.4}$$

定理 2 在定理 1 的条件下，$\forall\ B\in Q(V)$，则

$$g(B)\subset g_H(B\cap H) \tag{2.2.5}$$

定理 2 说明用部分征兆先进行诊断，真正的故障集 $g(B)$ 不会被遗漏.

定理 3 对于 H_1 满足 $H\subset H_1\subset V$，$\forall\ B\in Q(V)$，则

$$g(B)\subset g_{H_1}(B\cap H_1)\subset g_H(B\cup H) \tag{2.2.6}$$

定理 3 说明随着部分征兆集 H 扩展到 H_1，其对应的故障集 $g_{H_1}(B\cap H_1)$ 将更逼近真正的故障集 $g(B)$.

根据上面的定义和定理可总结出具体的诊断算法如下：

设 $U=\{u_1,u_2,\cdots,u_n\}$，$V=\{v_1,v_2,\cdots,v_m\}$ 为有 n 种故障和 m 种征兆的某一诊断对象的诊断模型. 已知故障 u_i 的征兆集 $f(u_i)\subset V$，$i=1,2,\cdots,n$.

依照集合与二值逻辑的关系，U 中任意子集 A 可用向量 $A=\{x_1,x_2,\cdots,x_n\}$ 来表示，其中 $x_i\in\{0,1\}$，那么 $f(A)=f(x_1,x_2,\cdots,x_n)$. 设故障 u_i 的征兆集为

$$f(u_i)=f(0,\cdots,0,1,0,\cdots,0)=(b_{i1},b_{i2},\cdots,b_{im})$$

其中 $b_{ij}\in\{0,1\}$ 为已知，$b_{ij}=1$ 表示故障 u_i 有征兆 v_j，否则 $b_{ij}=0$.

根据定义 1，我们可以得到下式：

$$f(x_1,x_2,\cdots,x_n)=\left(\bigvee_{i=1}^{n}(x_i\wedge b_{i1}),\cdots,\bigvee_{i=1}^{n}(x_i\wedge b_{im})\right) \tag{2.2.7}$$

式中 $\vee$ 是析取算子，满足 $\bigvee\limits_{i=1}^{n}a_i=\begin{cases}0, & a_i=0,\ i=1,2,\cdots,n\\ 1, & \text{其它}\end{cases}$

$\wedge$ 是合取算子，满足 $a\wedge b=\begin{cases}1, & a=b=1\\ 0, & \text{其它}\end{cases}$

令 $B=(y_1,y_2,\cdots,y_m)$，$y_j\in\{0,1\}$ 是任意征兆集，同时根据定义 1，可以推导下式：

$$g(y_1,y_2,\cdots,y_m)=\left\{\bigwedge_{j=1}^{m}(b_{1j}\rightarrow y_j),\cdots,\bigwedge_{j=1}^{m}(b_{nj}\rightarrow y_j)\right\} \tag{2.2.8}$$

式中→为蕴含算子，满足 $a\to b=\begin{cases}1, & a=0 \text{ 或 } b=1\\ 0, & \text{其它}\end{cases}$

式(2.2.8)给出了由征兆集求故障集的公式，在部分征兆集 $H\subset V$ 下，设 $I_H=\{j\mid j=1,2,\cdots,m,v_j\in H\}$. 那么

$$g_H(y_1,y_2,\cdots,y_m\mid H)=\left(\bigwedge_{j\in I_H}(b_{1j}\to y_j),\cdots,\bigwedge_{j\in I_H}(b_{nj}\to y_j)\right) \tag{2.2.9}$$

式(2.2.7)，(2.2.8)，(2.2.9)给出的是二值逻辑下的征兆与故障关系. 在实际情况下，征兆与故障都是模糊量，例如对某一种故障可分为很严重、较严重、一般等级别，更为普遍的情况是故障和征兆的状态采用[0,1]来代替{0,1}. 设 $U=\{u_1,u_2,\cdots,u_n\}$，$V=\{v_1,v_2,\cdots,v_m\}$，令 $A=(a_1,a_2,\cdots,a_n)$，$B=(b_1,b_2,\cdots,b_m)$，$a_i,b_j\in[0,1]$，分别表示诊断对象的故障集和征兆集. 此时式(2.2.1)转换为

$$\begin{aligned}&f(a_1,a_2,\cdots,a_n)\leqslant(b_1,b_2,\cdots,b_n)\qquad \text{当且仅当}\\ &g(b_1,b_2,\cdots,b_m)\geqslant(a_1,a_2,\cdots,a_n)\end{aligned} \tag{2.2.10}$$

在模糊逻辑的情形下，f,g 的表达式如下：

$$f(x_1,x_2,\cdots,x_n)=(\sup_i(x_i\wedge b_{i1}),\cdots,\sup_i(x_i\wedge b_{im})) \tag{2.2.11}$$

$$g(y_1,y_2,\cdots,y_m)=(\inf_j(b_{1j}\to y_j),\cdots,\inf_j(b_{nj}\to y_j)) \tag{2.2.12}$$

其中 $i=1,2,\cdots,n$；$j=1,2,\cdots,m$；sup，inf 分别为上下确界；∧，→分别为模糊逻辑中的合取和蕴含算子，表 2.1 列出了三种常用的算子.

表 2.1　三对常用的合取和蕴含算子

编号	$a\wedge b=$	$a\to b=$
1	$\min(a,b)$	$\begin{cases}1, & a\leqslant b\\ b, & a>b\end{cases}$
2	$\begin{cases}a\cdot b, & a>0\\ 0, & a=0\end{cases}$	$\begin{cases}\min\left(\frac{b}{a},1\right), & a>0\\ 1, & a=0\end{cases}$
3	$\max(a+b-1,0)$	$\min(1-a+b,1)$

为了进一步说明这种使用混合双向推理的序贯性诊断模型，这里举两个例子.

例 2.1 设某诊断对象有 8 种征兆，4 种故障，具体的知识库如表 2.2 所示.

表 2.2 某诊断对象的诊断知识库

征兆 / 关系 / 故障	征兆 v_1	征兆 v_2	征兆 v_3	征兆 v_4	征兆 v_5	征兆 v_6	征兆 v_7	征兆 v_8
故障 u_1	1	1	0	0	1	0	1	1
故障 u_2	0	1	1	0	1	1	1	0
故障 u_3	0	1	1	0	1	0	1	1
故障 u_4	1	0	1	1	0	1	1	0

已知部分征兆为 $B=(y_1,y_2,y_3,y_4,y_5,y_6,y_7,y_8)=(0,1,x,x,1,x,x,x)$，其中 x 表示未知的征兆，从部分征兆集 $H=(1,1,0,0,1,0,0,0)$ 开始诊断.

诊断开始，先由式(2.2.9)可以得出：

$$g_H(B\cap H)=\begin{pmatrix}(1\to 0)\wedge(1\to 1)\wedge(1\to 1)\\(0\to 0)\wedge(1\to 1)\wedge(1\to 1)\\(0\to 0)\wedge(1\to 1)\wedge(1\to 1)\\(1\to 0)\wedge(0\to 1)\wedge(0\to 1)\end{pmatrix}^T=(0,1,1,0)$$

因此得到诊断对象在已知部分征兆(v_1,v_2,v_5)下的初诊结果：可能存在故障为 u_2,u_3. 现在对可能的故障集 u_2 和 u_3 进行逼近或排除. 先考虑 u_2，由于 $v_3\in f(u_2)\quad H$，令 $H_1=H\cup\{v_3\}$，即得到 $H_1=(1,1,1,0,1,0,0,0)$. 提问：诊断对象是否存在故障 v_3？响应：存在. 从而有 $y_3=1$ 即 $B_1=(0,1,1,x,1,x,x,x)$. 仍根据式(2.2.9)，通过计算得 $g_{H_1}(B_1)=(0,1,1,0)$，即 u_2 仍为可疑故障. 由于 $v_6\in f(u_2)-H_1$，令 $H_2=H_1\cup\{v_6\}=(1,1,1,0,1,1,0,0)$. 提问：诊断对象是否存在故障 v_6？响应：不存在. 从而有 $y_6=0$，则 $B_2=(0,1,1,x,1,0,x,x)$，仍然依据式(2.2.9)，计算 $g_{H_2}(B_2)=(0,0,1,0)$，于是排除了故障 u_2，现在验证最后的可能故障 u_3，由于 $v_7,v_8\in$

$f(u_3)-H_2$，令 $H_3=\{v_7,v_8\}\cup H_2=(1,1,1,0,1,1,1,1)$，提问：诊断对象是否存在故障 v_7,v_8？响应：存在，则 $B=(0,1,1,x,1,0,1,1)$，计算 $g_{H_3}(B_3)=(0,0,1,0)$，由于 $f(u_3)-H_3=\varphi$，从而确诊了该诊断对象存在故障 u_3.

例 2.2 造成某种柴油机“负荷转速不足”有 5 个主要原因，表现出来的征兆有 6 个. 根据这种柴油机的经验资料和机理分析，可获得如下的关系.

表 2.3 柴油机故障诊断知识表

故障 j / 关系 / 征兆 i	气门弹簧断 y_1	喷油头积碳堵孔 y_2	机油管破裂 y_3	喷油过迟 y_4	喷油泵驱动键滚键 y_5
排气过热 x_1	0.6	0.4	0	0.98	0
振动 x_2	0.8	0.98	0.3	0	0
扭矩急转 x_3	0.95	0	0.8	0.3	0.98
机油压过低 x_4	0	0	0.98	0	0
机油耗量大 x_5	0	0	0.9	0	0
转速上不去 x_6	0.3	0.6	0.9	0.98	0.95

设现在已知部分征兆信息 $B=(b_1,b_2,b_3,b_4,b_5,b_6)=(1,0.15,x,x,x,0.85)$，设诊断过程中采用表 2.1 中的第三组合取和蕴含算子. 在诊断过程中，采取如下的判别原则：若某一故障的模糊隶属度大于 0.5，则说它是可能存在的故障.

诊断开始，由 $H=(1,1,0,0,0,1)$ 及 $B_0=B=(1,0.15,x,x,x,0.85)$ 得到

$$g_H(B\cap H)=\begin{Bmatrix}(0.6\to 1)\wedge(0.8\to 0.15)\wedge(0.3\to 0.85)\\(0.4\to 1)\wedge(0.98\to 0.15)\wedge(0.6\to 0.85)\\(0\to 1)\wedge(0.3\to 0.15)\wedge(0.9\to 0.85)\\(0.98\to 1)\wedge(0\to 0.15)\wedge(0.98\to 0.85)\\(0\to 1)\wedge(0\to 0.15)\wedge(0.95\to 0.85)\end{Bmatrix}^T$$

$$=(0.35,0.17,0.85,0.87,0.9).$$

由此得到初步诊断结果：存在可能故障 y_3, y_4, y_5，现在对可能故障 y_3, y_4, y_5 进行逼近和排除，询问：扭矩急转的隶属度是多少？响应：0.25. 结果计算 $0.85 \wedge (0.8 \rightarrow 0.25) = 0.45, 0.87 \wedge (0.3 \rightarrow 0.25) = 0.87, 0.9 \wedge (0.98 \rightarrow 0.25) = 0.27$，即在部分征兆 $H_1 = (1,1,1,0,0,1)$ 下，可能故障只剩下 y_4，由于 $f(y_4) - H_1 = \varphi$，从而也就确诊了诊断对象存在故障 y_4.

第三章　传统故障诊断专家系统

传统的故障诊断方法如故障树法、故障字典法等对于简单的诊断对象较容易实现，对于复杂的诊断对象则实现难度大且效果不好．人工智能的引进，为故障诊断领域开拓了一片新天地，采用专家系统为复杂对象进行故障诊断已成为一种趋势．

自从 Standford 大学于 1968 年开发第一个专家系统 DENDRAL 以来，专家系统由于其广泛的应用范围和能产生巨大的经济效益而得到了迅速的发展，现已成为人工智能的三大研究前沿（其余两个为模式识别和智能机器人）之一．诊断专家系统作为专家系统中的一个分支，其研究也得到了各国的高度重视，并相继在各行业中开发出了一些诊断专家系统，如 Bell 实验室于 1983 年开发的 ACE（用于电话电缆故障诊断与维护）系统、EGG 公司于 1982 年开发的 REACTOR（用于核反应堆故障诊断与处理）系统等等．

在这一章里，我们将首先简要介绍传统专家系统的基本结构及其工作原理，然后通过一个汽车故障诊断例子，分节讨论专家系统的各个组成部分以及如何采用 Turbo Prolog 语言来具体实现，最后简单介绍不精确的知识表示和推理、搜索策略以及专家系统的各种类型的开发工具．

3.1　传统诊断专家系统设计的基本组成

根据 Standford 大学 Feighbaum 教授（即 DENDRAL 的开发者）于 1982 年给出的定义：专家系统（Expert System 简称 ES）是一种智能的计算机程序，这种计算机程序使用知识与推理过程，求解那些需要杰出人物的专家知识才能求解的高难度问题．根据这

个定义我们可以知道：专家系统本质上是一个(或一组)计算机程序，它能借助人类的知识采取一定的搜索策略并通过推理的手段去解决某一特定领域的困难问题. 这一章所说的所谓传统故障诊断专家系统是相对于后面将要讨论的神经网络故障诊断专家系统(Fault Diagnosis Expert System Based on Neural Network 简称为 NN-FD-ES)和模糊神经网络故障诊断专家系统(Fault Diagnosis Expert System Based on Fuzzy Neural Network 简称为 FNN-FD-ES)而言，并无其它更深的含义.

为了完成专家系统最基本的功能，一个专家系统起码要包含三个组成部分：知识库、推理机及人机接口，其结构如图 3.1 所示.

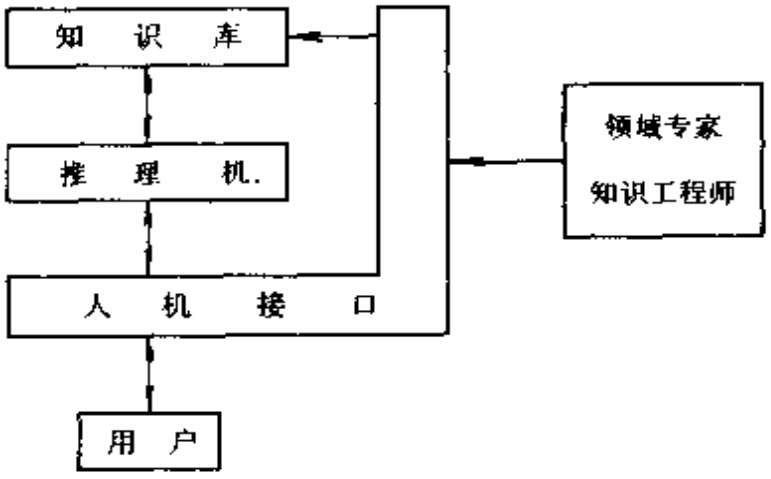


图 3.1 专家系统的最基本结构

上图反映了专家系统最简单的工作原理：在知识库创建和维护阶段，领域专家与知识工程师合作通过人机接口对知识库进行操作；在诊断阶段，用户也是通过人机接口将征兆信息传送给推理机，推理机根据诊断过程的需要，检索知识库中的各条知识或继续向用户要征兆信息，诊断结果也通过人机接口返回给用户.

一个专家系统应具有以下三个特性：

启发性：一个专家系统的知识库中不仅要有逻辑性知识，还要求能包含启发性知识. 所谓逻辑性知识是指能确保其准确无误的知识，通常是一些常识性知识；而启发性知识是指领域专家所掌握

的一些专业知识，它们通常没有严谨的理论依据，很难保证其普遍正确性．正是因为使用了启发性知识，也就使得专家系统在工作时会出错．

透明性：能向用户解释它的推理过程，还能回答用户提出的一些关于它自身的问题．一个专家系统所具有的解释能力是衡量一个专家系统水平的重要因素．

灵活性：专家系统知识库中的知识应便于修改和补充，正如知识的获取是设计一个专家系统时的“瓶颈”问题，该特性的实现也是一个难点．

上面的三个特性实际上就是设计专家系统时的三条要求，因此对于一个成熟的专家系统来说，为了实现这三条要求，还必须在上面的三个基本组成部分上增加另外三个组成部分：全局数据库、知识获取部分和解释部分，如图 3.2 所示，图中箭头表示数据流向．

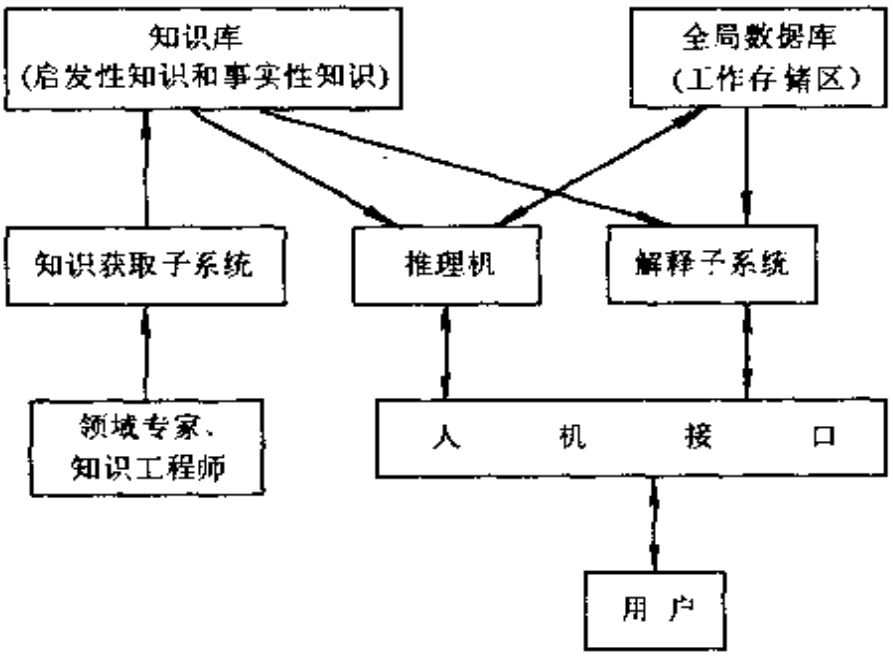


图 3.2　专家系统的一般结构

图 3.2 给出了专家系统六个基本组成部分：知识库、推理机、人机接口、知识获取子系统、解释子系统、全局数据库．在一个专家

系统中，它的知识库包含所要解决问题领域中的大量事实和规则．即知识库是领域知识及该专家系统工作时所需的一般常识性知识的集合．这些知识可以用一种或几种知识表示方法来表示．知识表示方法决定着知识库的组织结构并直接影响整个系统的工作效率．

知识库是一个独立的实体，它内存的知识通过程序来提取和管理．知识库中应易于存入新的知识而且不和已有的知识互相发生干扰．

推理机是专家系统的组织控制机构，它根据当前的输入数据（如机器设备运行时的各种征兆），运用知识库中的知识，按一定的策略进行推理，以达到要求的目标．在推理机的作用下，一般用户能够如同领域专家一样解决某一领域的困难问题．

全局数据库又称为工作存储器或动态数据库，是用于储存所诊断问题领域内原始特征数据的信息、推理过程中得到的各种中间信息和解决问题后输出结果信息的储存器．

知识获取子系统是专家系统和领域专家、知识工程师的接口．通过它与领域专家和知识工程师的交互，使知识库不仅可获得知识，而且可使知识库中的知识得到不断的修改、充实和提炼，从而使系统的性能得到不断的改善．

解释子系统能够对推理过程作出解释，可以解释推理的路线和为什么需要询问那些特征信息数据，而且还可以解释推理得到的确定性结论．在专家系统中设置解释子系统是专家系统与传统的计算机程序系统不同的一个重要特色，其目的是使用户更容易接受系统的整个推理过程和所得出的结论，同时也为系统的维护和专家经验知识的传授提供方便．

也有些人不把人机接口作为专家系统中的第六个组成部分，即他们把图 3.2 所示的专家系统看作由五个部分组成．人机接口有时又被称为用户界面，是专家系统和用户之间进行信息交换的媒介．它常常以用户熟悉的手段（如自然语言、图形、表格等）与用户进行交互；把用户输入的信息转换成系统的内部表示形式，然后

由相应的部件去处理；把系统内部的信息显示给用户．友善的用户界面是专家系统的重要组成部分．

如同图 3.1 所示的最简单的专家系统，具有一般结构的专家系统的工作原理大致为：在知识库创建和维护阶段，知识获取子系统在领域专家和知识工程师（在知识自动获取的情况下，可以脱离他们，然而到目前为止，专家系统的知识自动获取能力是很弱的）的指导下，将专家知识、诊断对象的结构知识等存放于知识库中或对知识库进行维护（增加、删除和修改）；在诊断阶段，用户通过人机接口将诊断对象的征兆信息传送给推理机，推理机根据诊断过程的需要，对知识库中的各条知识及全局数据库中的各项事实进行搜索或继续向用户索要征兆信息．最后，诊断结果也通过人机接口返回给用户；如需要，解释子系统可调用知识库中的知识和全局数据库中的事实对诊断结果和诊断过程中用户提出的问题作出合理的解释．

1983 年，Hayes Roth、Waterman 和 Lenat 等人提出了一种专家系统的理想结构模型．图 3.3 为这种理想结构模型的示意图．在这种模型中有一个在用户与专家系统之间进行通讯的语言处理模块，负责用户与系统之间的信息交流和转换，为用户提供与系统直接对话的能力；一个记录中间结果的黑板，它负责记录系统在求解

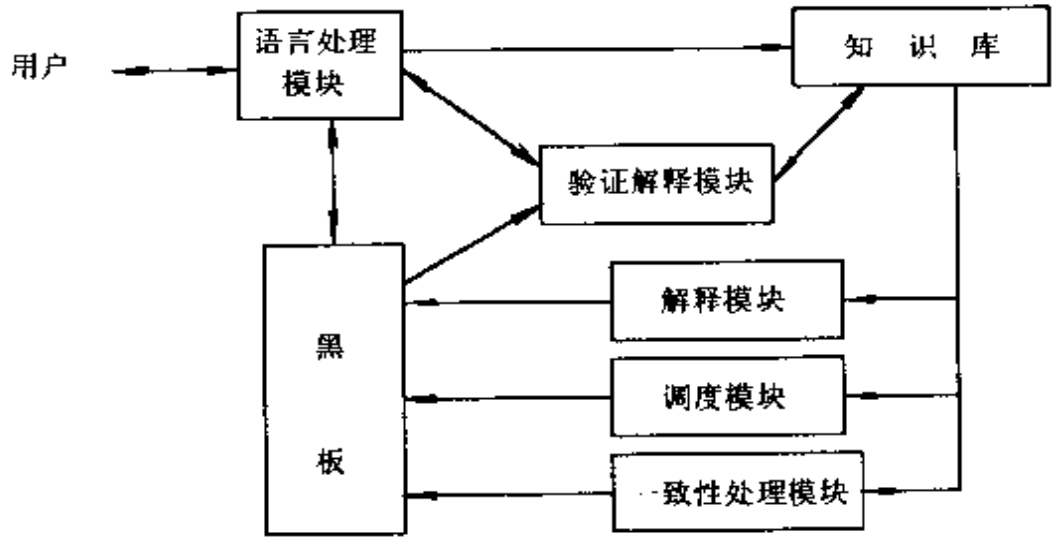


图 3.3 专家系统的理想结构示意图

过程中所产生的中间假设和结果，是沟通系统中各个部件的全局工作区；一个应用规则的解释模块，负责回答用户的提问，它能从黑板中找出对回答用户的问题有意义的信息；一个由事实和启发式规则以及问题求解规则组成的知识库，在知识库中，知识被划分为规则、事实、问题三类；一个控制规则处理顺序的调度模块，它负责管理控制议程，决定下一步做哪一工作；一个用于维护系统所得出的结果具有一致表示形式的一致性处理模块，它负责确保得出可能的结论，同时避免出现不一致的结论；一个用于向用户解释系统行为的验证解释模块.

然而到目前为止，还没有一个专家系统能包括所有这些组成部分，每一个实际的专家系统都是根据任务要求的不同而只包含其中的一个或多个组成部分.

3.2 知识库的建立和维护

3.2.1 知识获取

专家系统的核心是知识，故有时专家系统又被称为基于知识的系统. 知识库中拥有知识的多少及知识的质量决定了一个专家系统所具有解决问题的能力. 因此，建造一个专家系统首先便是要获取专业领域中的大量概念、事实、关系和方法，包括人类专家处理实际问题时的各种启发性知识，以构造一个内容丰富的知识库.

对于传统专家系统来说，知识库的建立更是整个系统设计的“瓶颈”，它的质量直接决定整个专家系统的质量.

一般来说，组建一个知识库需经历两个阶段：访问专家阶段和机器学习阶段. 在访问专家阶段，知识工程师通过对专家实际工作时如何求解问题进行观察、与专家进行长时间的交谈等手段获取知识，然后对这些知识进行精化、检查和验证等处理，最后将这些处理过的知识作为机器学习的材料. 在机器学习阶段实现将知识工程师提供的各种知识储存到知识库中，在传统专家系统中，机器

学习的方式由低到高大致可分为六个级别：机械学习、提问指导学习、实例学习、类比学习、通过书本资料学习和归纳总结学习.

机械学习又称死学习，是最简单的学习方式.在学习过程中，不需要作任何的假设，是从特殊到特殊的学习过程，训练时只需把特殊的知识教给对方，不作任何处理，系统所要做的就是记住所有的知识.例如某诊断专家系统的功能是确定一种映射关系，输入为(征兆 1，征兆 2，…，征兆 m)，输出为(原因 1，原因 2，…，原因 n).机械学习就是简单地把关系对[(征兆 1，征兆 2，…，征兆 m)，(原因 1，原因 2，…，原因 n)]存入知识库中.

通过指导学习类似于学生向老师学习的过程.这种学习方法的特点是外界提供的信息是知识库中所需要的信息，但它的表现形式很抽象或太一般，即训练者只给出一般的指示或建议，还不能直接为推理机所利用，学习环节必须把它们具体化为细节知识或特殊规则，成为可执行的形式，并与知识库中已有知识有机地联结在一起.总之，这是一个由一般到特殊的过程.这种方法的优点在于：它既能避免系统自己分析、归纳和发现知识的困难，又无需要求提供知识的领域专家了解系统内部表示和组织知识的具体细节.目前，专家系统中采用这种学习方法的较多.

通过实例学习是一种从特殊到一般的学习过程，即从特定的示例中归纳出一般性的规则.在这种学习方法中，外界提供的是专门、具体的信息，学习环节必须从中概括出事物的特性和规律性的知识，这是一种高级的学习行为.例如，让专家系统学习狗的概念，可以先提供给专家系统各种动物的知识，并告诉哪些是狗，哪些不是，那么这个系统学习以后会概括出一般性的规则和狗的概念模型，这样就可作为在动物世界中识别狗的分类的准则.

在通过类比学习情况下，外界只提供与某一个类似的执行任务有关的信息，因此，学习环节必须发现其类似性并假设出当前执行任务所需的类似规则，即从特殊事例概括出类比关系和转换规则，通常是获取新概念或新技巧的一种学习方法.这种类比的学习

方法在教学、管理、经济、法律、医学等方面普遍存在. 例如，在教学过程中，老师有时先教学生学习例题(即老师提供先例)，然后留习题给学生练习，学生一方面练习，一方面要去发现能应用于普遍情况下的若干原理. 即学生们必须在先例和习题之间找出对应关系，应用先例中的知识去处理习题中的问题，进行一般化的归纳，建立起原理，并对这些原理进行索引编排，以便以后检索、使用.

书本知识是专家知识的书面表述. 为了使系统能从书本中抽取自己所需的知识，通过书本资料学习方法首先要求建立一个资料库，以比较自然的方式将书本资料存放在库中，然后系统以资料库作为知识源进行学习.

通过归纳总结学习要求系统能在实际工作(运行)过程中不断地进行总结，归纳出成功和失败的经验教训，对知识库中的知识进行自调整和修改，以丰富系统的知识. 可见，归纳总结学习实质上是要求专家系统能够进行自学习. 为实现这种自学习过程，学习系统就必须有一个反馈装置，用来将实际的结果与所需结果进行比较，再利用反馈信息去修改知识库.

随着神经网络的重新兴起和遗传算法的提出，人们又提出了神经网络学习和通过遗传算法学习. 神经网络学习将在第四章介绍.

学习后的知识是以一定的方式储存在知识库中的，此即所谓的知识表示，如第一章介绍知识的表示方法有多种形式，比如：规则表示法、谓词逻辑表示法、框架表示法、语义网络表示法等等.

3.2.2 知识库的建立

现在我们来讨论如何具体建立一个汽车故障诊断专家系统的知识库. 设知识工程师根据与汽车维修专家(即领域专家)的交谈和对他实际操作过程的观察后得到图 3.4 所示的汽车故障征兆-原因结构图.

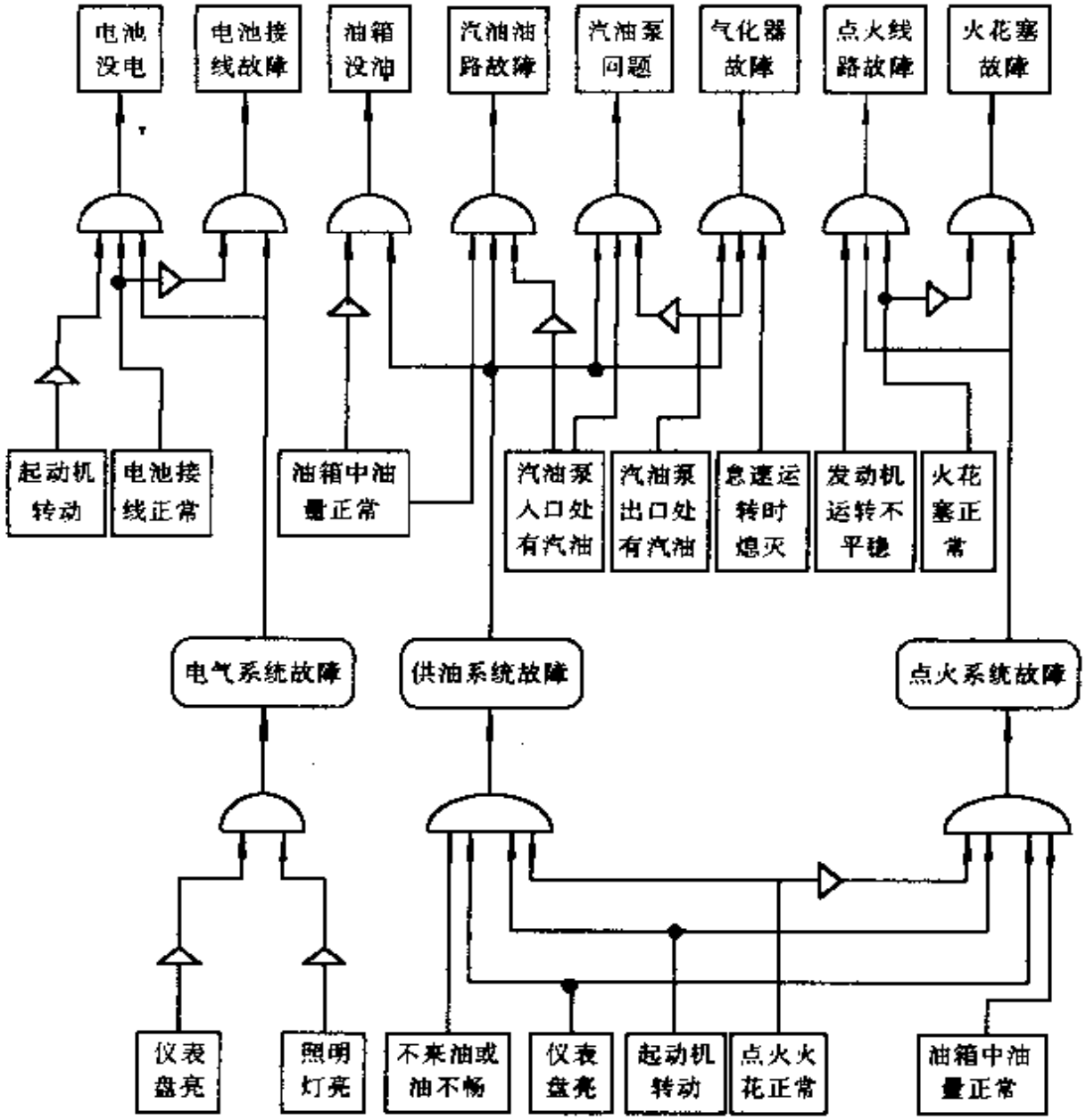


图 3.4　汽车故障征兆-原因结构图

图中—▷——表示对某一事实的否定.

由上图我们可以得出 11 条规则,它们的自然语言描述是:

规则 1

如果　电气系统故障

　且　起动机不转动

　且　电池接线正常

则　　电池没电

规则 2

如果　电气系统故障

　且　电池接线不正常

则　　电池接线故障

规则 3

如果　供油系统故障

　且　油箱中油量不正常

则　　油箱没油

规则 4

如果　供油系统故障

　且　油箱中油量正常

　且　汽油泵入口处没有汽油

则　　汽油油路故障

规则 5

如果　供油系统故障

　且　汽油泵入口处有汽油

　且　汽油泵出口处没有汽油

则　　汽油泵问题

规则 6

如果　供油系统故障

　且　汽油泵出口处有汽油

　且　怠速运转时熄灭

则　　气化器故障

规则 7

如果　点火系统故障

　且　发动机运转不平稳

　且　火花塞正常

则　　点火线路故障

规则 8

如果　点火系统故障

且　火花塞不正常
则　火花塞故障

规则 9

如果　仪表盘不亮
且　照明灯不亮
则　电气系统故障

规则 10

如果　不来油或油不畅
且　仪表盘亮
且　起动机起动
且　点火火花正常
则　供油系统故障

规则 11

如果　点火火花不正常
且　起动机起动
且　仪表盘亮
则　点火系统故障

值得注意的是，在建立这些产生式规则时要遵守下面两条基本原则：

1）尽可能用最小一组充分条件来定义一条产生式规则，避免不必要的冗余；

2）避免任何两条产生式规则发生冲突.

通过对上面这些以自然语言形式描述的产生式规则的理解，我们可以轻而易举地将其翻译成为 Turbo-prolog 语言的表示形式，然后，将其储存于知识库中. 如以 diagnose_is(no1.，故障 A) 表示汽车具有代号为 no1.，中文表示为故障 A 的故障；positive (has，征兆 B，no2)表示汽车具有中文表示为征兆 B 的症状，在规则中加入 no1，no2 是为了实现对推理过程的跟踪，为以后的解释程序的设计提供方便. not 表示对某一事实的否定，这样每条规则的具体形式可表示如下：

```
diagnose is(1,"电池没电"):-
    diagnose is(9,"电气系统故障"),
    not (positive(has,"起动机转动",1)),
    positive(has,"电池接线正常",1).
diagnose.is(2,"电池接线故障"):-
    diagnose.is(9,"电气系统故障"),
    not (positive(has,"电池接线正常",2).
diagnose is(3,"油箱没油"):-
    diagnose.is (10,"供油系统故障"),
    not (positive(has,"油箱中油量正常",3)).
diagnose.is(4,"汽油油路故障"):-
    diagnose.is (10,"供油系统故障"),
    not(positive (has,"汽油泵入口处有汽油",
        4)),
    positive (has,"油箱中油量正常",4).
diagnose.is(5,"汽油泵故障"):-
    diagnose is (10,"供油系统故障" ),
    positive (has,"汽油泵入口处有汽油",5 ),
    not (positive(has,"汽油泵出口处没有汽
        油",5)).
diagnose.is(6,"气化器故障"):-
    diagnose is (10,"供油系统故障"),
    positive (has,"汽油泵出口处有汽油",6),
    positive (has,"怠速运转时熄灭",6).
diagnose is(7,"点火线路故障"):-
    diagnose is (11,"点火系统故障"),
    positive (has,"火花塞正常",7),
    positive (has,"发动机运转不平稳",7).
diagnose is(8,"火花塞故障"):-
    diagnose.is (11,"点火系统故障"),
    not(positive (has,"火花塞正常",8)).
diagnose is(9,"电气系统故障"):-
    not (positive(has,"仪表盘亮",9)),
```

规则 1
调用规则 9

规则 2
调用规则 9

规则 3
调用规则 10

规则 4
调用规则 10

规则 5
调用规则 10

规则 6
调用规则 10

规则 7
调用规则 11

规则 8
调用规则 11

规则 9

```
    not (positive(has,"照明灯亮",9)).
diagnose.is(10,"供油系统故障"):-                    规则 10
    positive(has,"仪表盘亮",10),
    positive (has,"起动机转动",10),
    positive (has,"点火火花正常",10),
    positive (has,"不来油或油不畅",10).
diagnose.is(11,"点火系统故障"):-                    规则 11
    positive(has,"仪表盘亮",11),
    positive (has,"起动机转动",11),
    positive (has,"油箱中油量正常",11 ),
    not (positive(has,"点火火花正常",11)).
```

这样，我们便形成了一个由 11 条规则组成的知识库，在建立这个知识库的过程中所采取的机器学习方法是机械学习，即将这些规则不作任何处理，直接将其输入到一个专家系统中去. 通常每一个传统专家系统的学习过程都必须包含机械学习.

由图 3.4 可得出故障征兆-原因表 3.1.

表 3.1　故障征兆-原因表

故障 / 关系 / 征兆	电气系统故障	点火系统故障	供油系统故障	电池没电	电池接线故障	油箱没油	汽油路故障	汽油泵问题	气化器故障	火花塞故障	点火线路故障
起动机不转动	0	0	0	1	0	0	0	0	0	0	0
电池接线不正常	0	0	0	0	1	0	0	0	0	0	0
油箱中油量不正常	0	0	0	0	0	1	0	0	0	0	0
汽油泵入口处没有汽油	0	0	0	0	0	0	1	0	0	0	0
汽油泵出口处没有汽油	0	0	0	0	0	0	0	1	0	0	0
火花塞不正常	0	0	0	0	0	0	0	0	0	1	0
仪表盘不亮	1	0	0	0	0	0	0	0	0	0	0
照明灯不亮	1	0	0	0	0	0	0	0	0	0	0
点火火花不正常	0	1	0	0	0	0	0	0	0	0	0
不来油或油不畅	0	0	1	0	0	0	0	0	0	0	0
怠速运转时熄灭	0	0	0	0	0	0	0	0	1	0	0
发动机运转不平稳	0	0	0	0	0	0	0	0	0	0	1
电气系统故障	0	0	0	1	1	0	0	0	0	0	0
点火系统故障	0	0	0	0	0	0	0	0	0	1	1
供油系统故障	0	0	0	0	0	1	1	1	1	0	0

有了上表我们便可以采用谓词逻辑知识表示法来表示汽车维修知识. 在下面的谓词逻辑表示中,rule 表示规则,symptom 表示征兆,topic 表示主题. 规则 rule(no1,故障 1,故障 2,L)表示:规则号为 no1,该规则的上级故障为故障 1,本级故障为故障 2,具有征兆表 L;symptom(no2,征兆 A)表示编号为 no2 的征兆为征兆 A.

topic("汽车")
topic("电气系统故障")
topic("点火系统故障")
topic("供油系统故障")
rule(1,"汽车","电气系统故障",[7,8])
rule(2,"汽车","点火系统故障",[9])
rule(3,"汽车","供油系统故障",[10])
rule(4,"电气系统故障","电池没电",[1])
rule(5,"电气系统故障","电池接线故障",[2])
rule(6,"供油系统故障","油箱没油",[3])
rule(7,"供油系统故障","汽油油路故障",[4])
rule(8,"供油系统故障","汽油泵问题",[5])
rule(9,"供油系统故障","气化器故障",[11])
rule(10,"点火系统故障","火花塞故障",[6])
rule(11,"点火系统故障","点火线路故障",[12])
symptom(1,"起动机不转动")
symptom(2,"电池接线不正常")
symptom(3,"油箱中油量不正常")
symptom(4,"汽油泵入口处没有汽油")
symptom(5,"汽油泵出口处没有汽油")
symptom(6,"火花塞不正常")
symptom(7,"仪表盘不亮")
symptom(8,"照明灯不亮")
symptom(9,"点火火花不正常")
symptom(10,"不来油或油不畅")

symptom(11,"怠速运转时熄灭")

symptom(12,"发动机运转不平稳")

这样我们便建立了一个基于谓词逻辑的知识库.从上面可以看出,在基于逻辑的专家系统中,知识库是由一些说明事实的谓词逻辑子句组成,将这些事实和子句存在名字为 car.dba 的数据文件中,即为独立于推理机制的知识库.与基于规则的专家系统相似,知识库也必须有一个清晰的逻辑组织,并要做到冗余数据最少.

3.2.3 知识库的维护

知识库的维护实际上也是知识的获取,与建立知识库相比,它所采用的是高一级的机器学习方法:通过指导学习,而非机械学习.

通常,对知识库的维护包括三种操作:扩展知识库、修改知识库和删除知识库.在简单的专家系统中,修改知识库的操作可由扩展知识库和删除知识库的组合来完成:先删除要修改的记录,然后加入修改后的记录.因此,对于修改知识库就不再作详细的讨论.

在用 Turbo-prolog 语言编制的专家系统中,知识库的维护是针对用谓词逻辑表示法表示的知识库.由于这类知识库中通常包括三类知识:故障主题、故障规则和征兆事实.这样对于知识库的维护应包括六种操作:扩展故障主题、扩展故障规则、扩展征兆事实、删除故障主题、删除故障规则和删除征兆事实.同时,在进行维护操作时,为方便用户应尽可能采用菜单选择输入方法和数字输入方法.

在维护过程中,知识库是放在内存中的,因此在退出维护前应将其存入到磁盘中.

汽车故障诊断专家系统的知识库的维护是按以下方法进行:

```
.....
consult("car.dba"),                    /* 调出知识库 */
show_menu2,
```

```
.....
show_menu2 : -clearwindow,nl,nl,nl,nl,
     write("   扩展故障知识库"),nl,nl,
     write("   1   扩展故障主题"),nl,
     write("   2   扩展故障规则"),nl,
     write("   3   扩展征兆事实"),nl,
     write("   4   删除故障主题"),nl,
     write("   5   删除故障规则"),nl,
     write("   6   删除征兆事实"),nl,
     write("   7   退出系统"),nl,nl,
     write("·····请选择 1--7"),nl,
     write("您的选择是"),nl,
     readint(Choice),
     process1(Choice),
     save("car.dba"),
     show_menu2.
process(7) : -show_menu2.
process1(1) : -nl,write("请输入故障主题"),
     readln(Topic_name),
     assertz(topic(Topic_name)).
process1(2) : -nl,write("请输入规则序号:"),
     readint(Rule_no),
     nl,write("请输入上一级故障描述:"),
     readln(Rule_fa1),
     nl,write("请输入本级故障描述:"),
     readln(Rule_fa2),
     nl,write("请输入故障征兆约束表"),
     nl,write("输入数字,用空格间隔,回车结束"),
     readln(Sentence),
     convers(Sentence,List),
```

```
        convert(List,COND),
        assertz(rule(Rule_no,Rule_fa1,Rule_fa2,COND)).
process1(3):-nl,write("请输入征兆序号:"),
        readint(Cond_no),
        nl,write("请输入征兆描述:"),
        readln(Cond_sym),
        assert(symptom(Cond_no,Cond_sym)).
process1(4):-nl,write("目前数据库中的主题集"),nl,nl,
        display1,
        nl,write("请输入要删除的故障主题:"),
        readln(Topic_name),
        retract(topic(Topic_name)).
process1(5):-nl,clearwindow,
        write("目前数据库中的规则集"),nl,
        write("规则号  上级故障  本级故障  征兆表"),nl,
        display2,
        nl,write("请输入要删除的规则号:"),
        readint(Rule_no),
        retract(rule(Rule_no,_,_,_)).
process1(6):-nl,write("目前数据库中的征兆集"),nl,nl,
        write("征兆序号  征兆描述"),nl,
        display3,
        nl,write("请输入要删除的征兆序号"),
        readint(Cond_no),
        retract(symptom(Cond_no,_)).
process1(_):-show_menu2.

display1:-topic(Topic_name1),
        nl,write("        ",Topic_name1),
        fail.
```

```
display1.

display2 :-rule (RNO1, CATEGORY1, CATEGORY2,
        CONDITIONS1),
    nl,write("     ",RNO1,"     "),
    writef("%20s",CATEGORY1),
    writef("%20s",CATEGORY2),
    write("     ",CONDITIONS1),
    fail.
display2.

display3 :-symptom(BNO1,STRING1),
    nl,write("     ",BNO1,"     "),
    write(STRING1),
    fail.
display3.

convers(-,[ ]).
convers (Str, [Head | Tail]) :-fronktoken (Str, Head,
        Str1),!,
    convers(Str1,Tail).

convert([ ],[ ]).
convert([Head|Tail],[Head1|Tail1]) :-
    str_int(Head,Head1),
    convert(Tail,Tail1).
```

3.3 全局数据库的设计与操作

全局数据库主要用于存放专家系统运行过程中产生的一些数

据记录及诊断问题领域内的原始特征数据.由于在 Turbo-Prolog 数据库中,数据记录是以子句的方式储存的,因此在使用全局数据库之前,有必要对子句谓词进行定义.

如同一般的谓词的定义方式,数据库谓词的定义方式很简单,如在附录中的系统 AUTO-FWFDES 中,只要在数据库中存放对故障征兆询问作出的回答,则可如下定义:

```
database
    xpositive(symbol,symbol)
    xnegative(symbol,symbol)
```

在系统 AUTO-BWFDES 中,由于需要将知识库调进数据库中,因而,还需在数据库中定义知识库谓词.具体方式如下:

```
database
    rule(RNO, CATEGORY, CATEGORY, CONDI-
        TIONS)
    symptom(BNO,string)
    topic(symbol)
    yes(BNO)
    no(BNO)
    diag_list(integer,symbol)
```

谓词 diag_list(integer,symbol)用于列出故障编号与故障含义的对应关系,以方便以后用户对知识库的维护.

对数据库的操作主要为增加记录和删除记录两种.如前面介绍的知识库的维护实质上就是对数据库进行增加记录与删除记录的操作.Turbo-Prolog 中几个与数据库操作相关的内部谓词简要介绍如下:

assert	在内部数据库尾插入一事实
asserta	在内部数据库头插入一事实
assertz	在内部数据库尾插入一事实
retract	删除内部数据库的一个事实
retractall	删除内部数据库所有匹配事实

在系统 AUTO-FWFDES 中，我们用如下的语句来实现对数据库的操作：

```
remember(X,Y,1) : -asserta(xpositive(X,Y)).
                        /* 增加记录 */
remember(X,Y,0) : -asserta(xnegative(X,Y)), fail.
cleat_facts : -retract(Xpositive(_, )), fail.
                        /* 清除记录 */
clear_facts : -retract(Xnegative(_, )), fail.
```

在系统 AUTO-BWFDES 中，数据库的操作为

```
do_answer(BNO,1) : -assert(yes(BNO)).
                        /* 增加记录 */
clear_facts : -retract(yes(_)), fail.
                        /* 清除记录 */
process1(1) : - ……,
    assertz(topic(Topic_name)).
                        /* 在数据库末尾增加主题 */
process1(2) : - ……,
    assertz(diag_list(Rule_no,Rule_fa2)),
                        /* 在数据库末尾增加记录 */
    ……,
    assertz(rule(Rule_no,Rule_fa1,Rule_fa2,COND)).
                        /* 在数据库末尾增加该条规则 */
process1(3) : - ……,
    assertz(symptom(Cond_no,Cond_sym)).
                        /* 在数据库末尾增加该征兆 */
process1(4) : - ……,
    retract(topic(Topic_name)).
                        /* 删除该主题 */
process1(5) : - ……,
    retract(diag_list(Rule_no,_)),
```

```
                    /* 删除该记录 */
    retract(rule(Rule_no,_,_,_)).
                    /* 删除编号为 Rule_no 的规则 */
process1(6) :- ……,
    retract(symptom(Cond_no,_)).
                    /* 删除编号为 Cond_no 的征兆 */
```

3.4 推 理 机

在前面的分析中，我们知道：推理机作为专家系统的组织控制机构，能通过运用由用户提供的征兆数据，从知识库中选取相关的知识并按照一定的推理策略进行推理，直到得出相应的结论。在设计推理机时应考虑推理方法、推理方向和搜索策略三个方面，我们将在 3.8 节再对搜索策略进行讨论。

3.4.1 推理方法

推理方法分为精确推理和不精确推理两种。

(1) 精确推理

所谓精确推理就是把领域知识表示为必然的因果关系，推理的前提和推理的结论或者是肯定的，或者是否定的，不存在第三种可能。在这种推理中，一条规则被激活的条件是它的所有前提都必须为真。现在结合 Turbo Prolog 语言举一个人物识别例子说明。

例 3.1 某基于规则的专家系统的目标函数是 name_is(*X*)，现在已建立的知识库中有下面所示的四条规则：

```
name_is ("tom") :-
        sex(male),
        positive(has,"tall figure"),
        positive(has,"long haired").
name_is("mary") :-
        sex(female),
```

```
        positive(has,"tall figure"),
        positive(has,"short haired").
name_is("judy"):-
        sex(female),
        positive(has,"short figure"),
        positive(has,"long haired").
name_is("smith"):-
        sex(male),
        positive(has,"tall figure"),
        positive(has,"short haired").
```

为推出目标即所求者的姓名，系统通过人机接口进行询问，不断获取所需信息即特征，同时不断匹配合适的规则，具体的推理过程如图 3.5 所示：

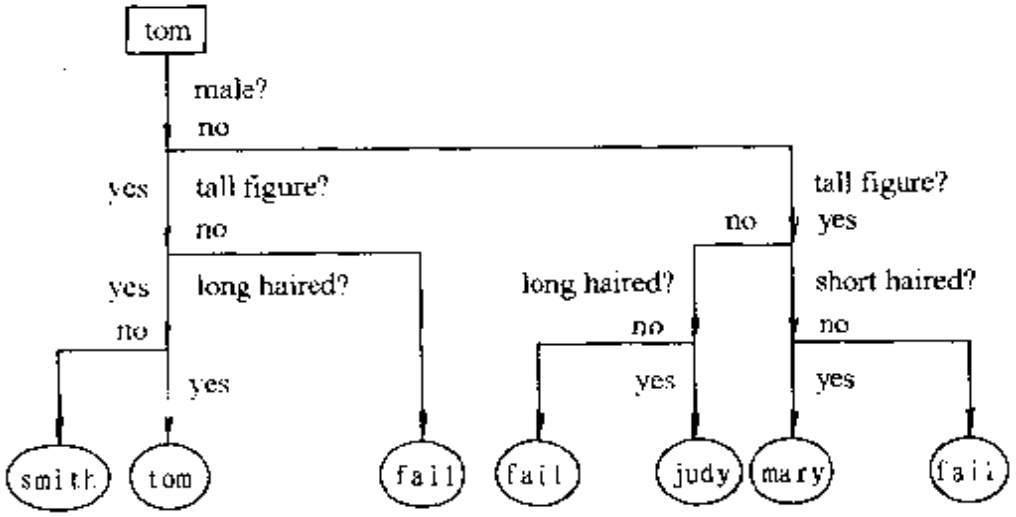


图 3.5　基于规则的精确推理图

图中的 fail 表示在知识库中找不到合适的规则，匹配失败。基于谓词逻辑的专家系统有类似的精确推理过程。

(2) 不精确推理

由于在现实实际中，事物的特征并不总是表现出明显的是与非，同时还可能存在着其它原因，如概念模糊、知识本身存在着可信度问题等，因而使得在专家系统中往往要使用不精确推理方法，

不精确推理又称为似然推理，是专家系统中常用的推理方法，它比精确推理要复杂得多．关于不精确知识表示和推理的讨论将在3.7节进行．

在附录的系统 AUTO-FWFDES 和系统 AUTO-BWFDES 中，我们采用的推理方法都是精确推理．

3.4.2 推理方向

推理方向有三种：正向（或向前）推理（forward chaining）、反向（或向后）推理（backward chaining）及正反向混合推理（forward and backward chaining）．

（1）正向推理

正向推理是指从已知的事实出发，向结论方向进行推导，直到推出正确的结论．这种方式又称为事实驱动方式，它的大体过程是：系统根据用户提供的原始信息与规则库中的规则的前提条件进行匹配，若匹配成功，则将该知识块的结论部分作为中间结果，利用这个中间结果继续与知识库中的规则进行匹配，直到得出最后的结论．

设知识库中共有 K 条规则，其中第 I 条（$1\leqslant I\leqslant K$）的前提部分的事实又有 J 个，对于单故障诊断推理来说，可用图 3.6 来表示它的推理机．

由于 Turbo Prolog 语言在设计上是以回溯、递归等手段来实现匹配的，因此，它在设计正向推理机时显得特别的简洁有效．如在系统 AUTO-FWFDES 中的正向推理机是：

```
.....
process(1),
.....
process (1) : -clearwindow,
        diagnose_is(N,X),!,
        write("您的车的故障是",X),nl,
        clear_facts.
```

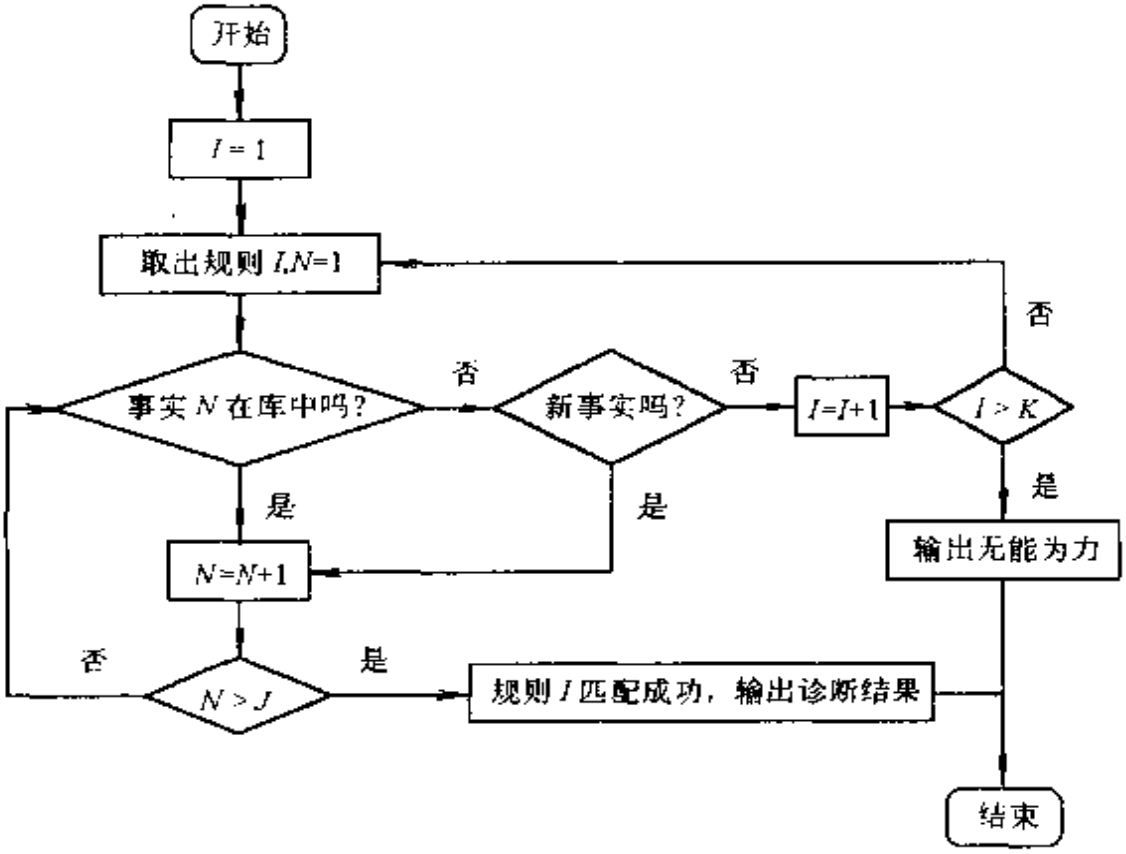


图 3.6　正向推理示意图

```
process(1) :-nl,write("对不起,爱莫能助"),
        clear_facts.
positive(X,Y) :-xpositive(X,Y),!.
positive(X,Y) :-not(negative(X,Y)),!,
        ask(X,Y).
negative(X,Y) :-xnegative(X,Y).
ask(X,Y) :-write("提问:-您的车是否有……",Y,"?"),
        readint(Reply),
        remember(X,Y,Reply).
remember(X,Y,1) :-asserta(xpositive(X,Y)).
remember(X,Y,0) :-asserta(xnegative(X,Y)),
        fail.
```

在介绍精确推理时,所举的人物识别例子也属于正向推理这

一方式.

与其它推理方式相比，正向推理简单，容易实现，但在推理过程中常常要用到回溯，从而推理速度较慢，且目的性不强，不能反推.

(2) 反向推理

所谓反向推理是指先从知识库中选择一种故障作为假设，然后寻找支持假设的证据或事实来验证这种假设的真假性，当用户提供的数据与系统所需要的证据完全匹配成功时，则推理成功，所作的假设也就得到了证实. 这种推理方式又称为目标驱动方式. 与正向推理相比，反向推理具有很强的目的性.

反向推理一般用于验证某一特定规则是否成立. 比如，在汽车诊断专家系统中，就可采用反向推理，先假设诊断对象存在某种可能的故障原因，然后一一去验证相关的故障征兆.

在医疗诊断专家系统中常常采用反向推理.

在系统 AUTO-BWFDES 中，我们采用了反向推理机制，其示意图如下：

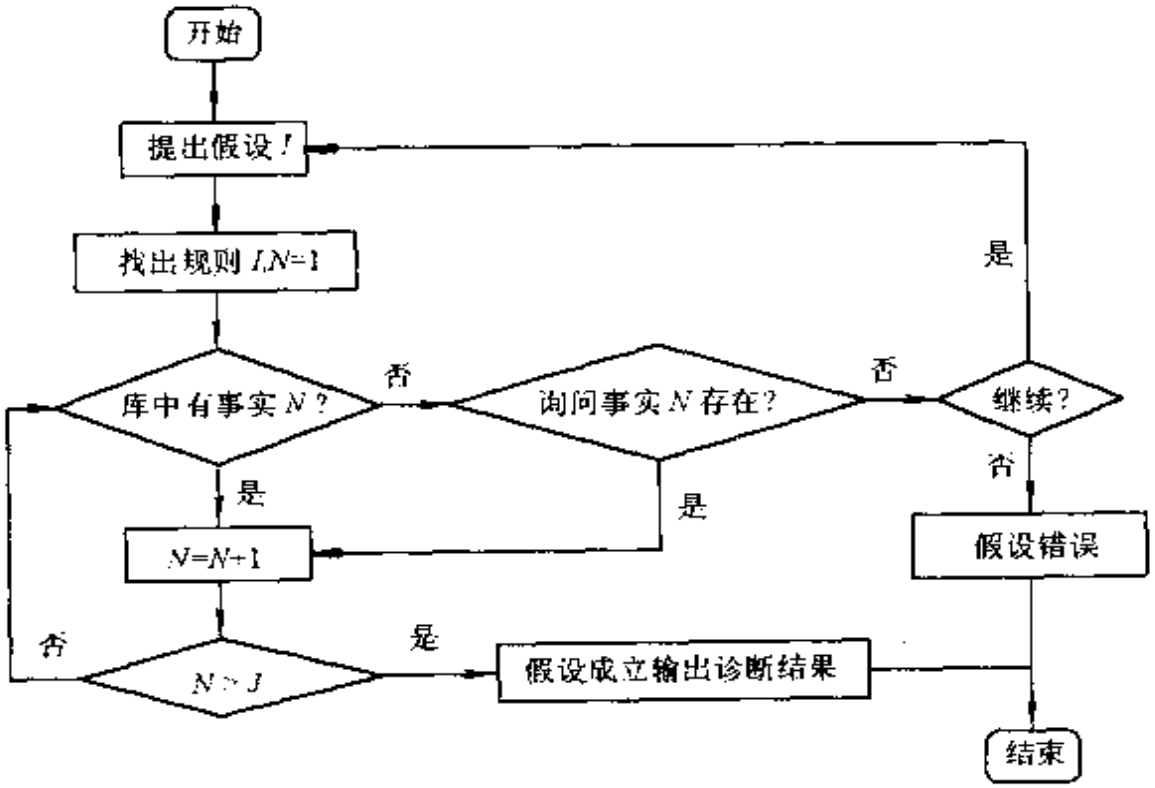


图 3.7 反向推理机制示意图

具体的语言实现为

```
do_diagnosing : -show_menu1.
    show_menu1 : -clearwindow,nl,nl,
        write("请输入您怀疑的故障代号"),nl,
        write("  1  电池没电"),nl,
        write("  2  电池接线故障"),nl,
        write("  3  油箱没油"),nl,
        write("  4  汽油油路故障"),nl,
        write("  5  汽油泵故障"),nl,
        write("  6  汽化器故障"),nl,
        write("  7  点火电极故障"),nl,
        write("  8  火花塞故障"),nl,
        write("  9  退出诊断"),nl,
        write("您的选择是："),
        readint(I),
        verify(I),
        nl,write("**********************"),
        nl,write("您的猜测完全正确"),
        nl,write("任一键进行下一次诊断"),
        nl,write("**********************"),
        clear_facts,
        readchar(_),
        show_menu1.

    verify(9) : -exit.
    verify(I) : -diag_list(I,X),
        rule(RNO1,Y,X,COND1),
        rule(RNO2,_,Y,COND2),
        prove(COND2),
        prove(COND1).
```

```
prove([ ]).
prove([Head|Tail]) :-prove1(Head),
    prove(Tail).

prove1(BNO) :-yes(BNO).
prove1(BNO) :-symptom(BNO,Z),
    ask(BNO,Z).

ask(BNO,Z) :-nl,write("问题:-",Z,"吗?"),
    makewindow (2,15,-7,"用户响应窗口",16,40,4,
        32),
    write("肯定,否定——键入 1,0"),nl,
    readint(Response),
    do_answer(BNO,Response),
    clearwindow,
    shiftwindow(1).

do_answer(BNO,1) :-assert(yes(BNO)).
do_answer(BNO,0) :-shiftwindow(1),
    nl,write("您的猜测错误,请按任意键返回选择"),
    readchar(_),
    show_menu.

clear_facts :-retract(yes(_)),
    fail.
clear_facts.

diag_list(1, "电池没电").
diag_list(2, "电池接线故障").
diag_list(3, "油箱没油").
```

diag_list(4, "汽油油路故障").

diag_list(5, "汽油泵故障").

diag_list(6, "汽化器故障").

diag_list(7, "点火电极故障").

diag_list(8, "火花塞故障").

(3) 正反向混合推理

所谓正反向混合推理是指先根据给定的原始数据或证据(这些数据或证据往往是不充分的)向前推理,得出可能成立的诊断结论,然后,以这些结论为假设,进行反向推理,寻找支持这些假设的事实或证据.

正反向双向推理一般用于以下几种情形:已知条件不足,用正向推理不能激发如何一条规则;正向推理所得的结果可信度不高,用反向推理来求解更确切的答案;由已知条件查看是否还有其它结论存在.

双向推理机制的结构图如下:

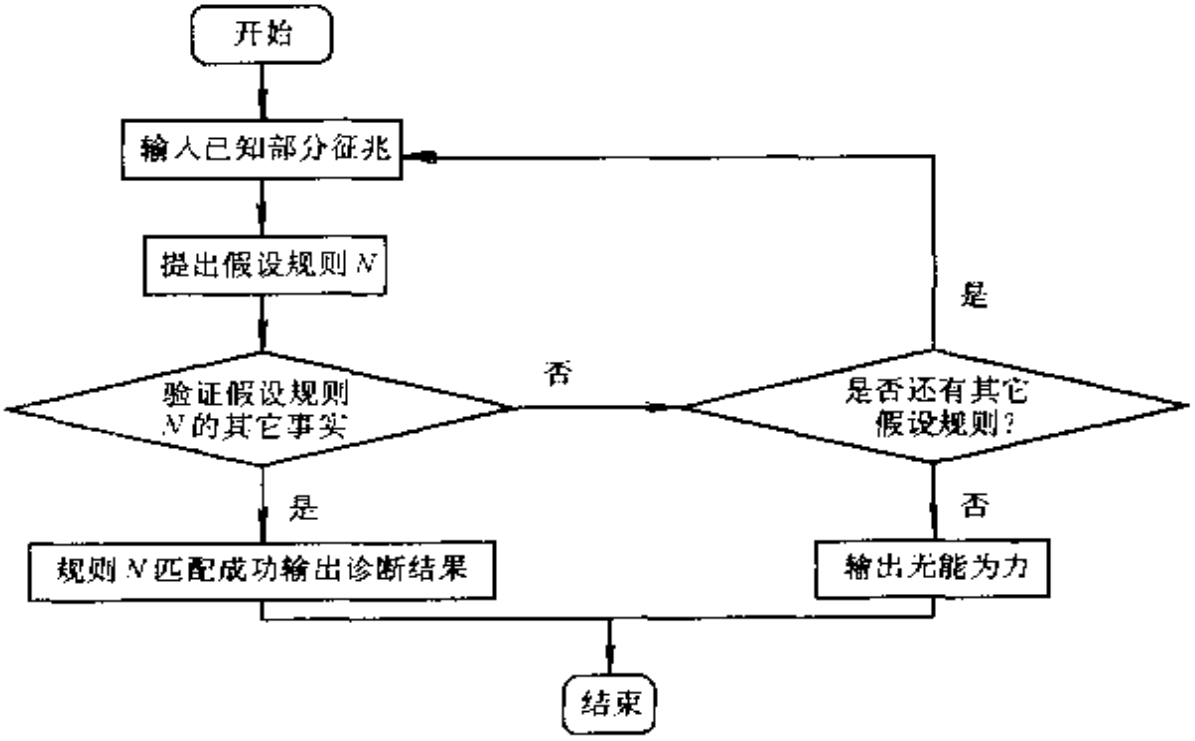


图 3.8 双向推理机制示意图

正反向双向推理集中了正向推理和反向推理的优点,更类似

于人们日常进行决策时的思维模式，求解过程也更为人们所理解，但控制策略较前面两种更为复杂，这种方法经常用来实现复杂问题的求解.

3.5 解释程序的设计

解释程序负责回答用户可能提出的各种问题，包括与系统运行有关的问题和与运行无关的关于系统自身的一些问题. 在专家系统的组成部分中，它不是一个必不可少的部分，但它是实现系统透明性的主要部件，是一个专家系统区别于其它计算机程序系统的一个重要特征. 对于一个完善的专家系统来说，不仅要求它能够以专家级的水平去解决问题，而且还要求它能对问题的求解过程和求解结果给出合理的解释. 只有这样，系统给出的结论才是令人信服的，专家系统本身才是有效的.

一些著名的专家系统都有自己独特的解释机制. 例如，由斯坦福国际研究所开发的系统 PROSPECTOR 的解释程序系统可以为用户提供几种不同类型的解释. 最简单的一种解释是允许系统在咨询的任何时刻检查推理网络中某个语义空间的后验概率，其次解释系统可以向用户显示推断某一结论所用的规则. 用户还可以检查某一数据对推理网络中任一特定空间概率的影响. 这种解释可以为用户提供两种很有意义的信息. 首先，系统可以通过这种解释能力告诉用户，它所采集的数据中哪些是最有意义的；其次，系统可以提示用户需要采集的有用的数据是什么.

总之，解释程序为用户提供了关于系统的一个认识窗口，使用户理解程序正在做什么和为什么这样做以及为什么得出这样的结论. 专家系统之所以受到人们的欢迎，且能够蓬勃地发展起来，同它有理智的解释是分不开的.

为了实现对各种询问的回答，解释机制一般都使用几个比较通用的问题规划. 例如：在回答“为什么”得到某一结论的询问时，系统通常需要反向跟踪全局数据库中保存的解链或推理路径，并

把它翻译成用户能接受的自然语言表达方式.在回答“为什么不”之类的询问时,系统一般要使用有关解释技术的启发式方法.

在设计一个解释程序时,应注意以下几个方面:

1) 能够对专家系统知识库中所具有的每一种故障原因给出合理的解释,能够在诊断过程中对用户的每一个 Why 作出响应.

2) 每一次的解释都要求做到完整,且易于理解.

3) 充分考虑使用该专家系统的具体用户情况,不同的用户对解释程序有不同的侧重面要求.

下面简单介绍几种解释方法.

1. 预置文本与路径跟踪法

预置文本法是最简单的解释方法,具体的方法是将每一个问题求解方式的解释框架采用自然语言或其它易于被用户理解的形式事先组织好,插入程序段或相应的数据库中.在执行目标的过程中,同时生成解释信息,其中的模糊量或语言变量通常都要转化为合适的修饰词.一旦用户询问,只需把相应的解释信息填入解释框架,并组织成合适的文本方式提交给用户即可.

这种解释方法简单直观,知识工程师在编制相应解释的预置文本时,可以针对不同用户的要求随意编制不同的解释文本,其缺点在于对每一个问题都要考虑其解释内容,大大增加了系统开发时的工作量.大型专家系统的解释机制不可能采用这种方法,只能用于小型专用系统.

克服预置文本法缺陷的一条自然途径是对程序的执行过程进行跟踪,在问题求解的同时,将问题求解所使用的知识自动记录下来.当用户提出相应的问题时,解释机制向用户显示问题的求解过程,这就是路径跟踪法.

MYCIN 的解释机制就是采用预置文本法和路径跟踪法.

2. 策略解释法

在专家系统的许多实际应用中,用户往往不满足于系统简单

地告诉他们是怎样一步步得到问题的结论的，而要求系统给出求解问题所采用的其它方法和手段. 策略解释法就是依照这种要求来设计的.

策略解释法向用户解释的是与问题求解策略有关的规划和方法，从策略的抽象表示及其使用过程中产生关于问题求解的解释.

3. 自动程序员方法

前面的解释只回答了“Why”这一询问也即仅仅解释了系统的行为，而没有论证其行为的合理性. 为了解决这一问题，Swartout 在 XPLAIN 系统的设计中提出了自动程序员解释方法.

自动程序员方法的基本思想是：在设计一个专家咨询程序的过程中，对领域模型和领域原理进行描述的同时，将自动程序员嵌入其中，通过自动程序员将描述性知识转化成一个可执行的程序，附带产生有关程序行为的合理性说明，从而向用户提供一个非常有力的解释机制.

1985 年，Neches 等人提出了一种新的解释系统设计方法. 这种方法的基本思想是将专家系统的设计与解释机制的设计结合在一起进行全盘考虑，从而给出一种新的解释方法. 这种方法既可以减轻领域专家和知识工程师建立知识库的工作量，又可为用户提供关于系统行为合理性的非常有力的说明.

在系统 AUTO-FWFDES 中，我们采用的解释方案是预置文本与路径跟踪法，为了达到跟踪的目的，在规则中加入了规则编号. 其解释内容包括对诊断结果的解释和对推理过程的解释两种.

3.5.1 对诊断结果的解释

为了实现对诊断结果的解释，AUTO-FWFDES 做了三步相关的工作：

1) 对每一条规则预置各自的解释文本；

2) 使用谓词 append_list 对诊断过程进行跟踪，产生表 L，L 中包括了所有匹配成功的规则编号.

3) 使用谓词 explain_list 对表 L 进行解释.

具体的程序编码为

```
.....
append_list(L1,L2,L),          /* 路径跟踪,形成解释表 */
explain_list(L),
.....
explain_list([ ]).             /* 对表进行解释 */
explain_list([Head|Tail]):-explain(Head),
        explain_list(Tail).
.....
        append_list([ ],L,L).
append_list([N1|L2],L3,[N1|L4]):-append_list(L2,L3,
        L4).

/* 预置文本 */
explain(1):-nl,
    write("因为您的车存在电气系统故障,"),nl,
    write("且发动机不转动,"),nl,
    write("且电池接线正常,"),nl,
    write("因此,根据规则 1 得出您车上的电池没电."),nl.
explain(2):-nl,
    write("因为您的车存在电气系统故障,"),nl,
    write("且电池接线不正常,"),nl,
    write("因此,根据规则 2 得出,您的车的电池接线存在
        问题."),nl.
explain(3):-nl,
    write("因为您的车存在供油系统故障,"),nl,
    write("且油箱中油量不正常,"),nl,
    write("因此,根据规则 3 得出,您车上的油箱中没油."),
        nl.
```

```
explain(4) :-nl,
        write("因为您的车存在供油系统故障,"),nl,
        write("且汽油泵入口处没有汽油,"),nl,
        write("且油箱中油量正常,"),nl,
        write("因此,根据规则 4 得出,您车上的汽油油路存在
            问题."),nl.
explain(5) :-nl,
        write("因为您的车存在供油系统故障,"),nl,
        write("且汽油泵入口处有汽油,"),nl,
        write("且汽油泵出口处没有汽油,"),nl,
        write("因此,根据规则 5 得出,您车上的汽油泵存在问
            题."),nl.
explain(6) :-nl,
        write("因为您的车存在供油系统故障,"),nl,
        write("且汽油泵入口处有汽油,"),nl,
        write("且怠速运转时熄灭,"),nl,
        write("因此,根据规则 6 得出,您的车的毛病是汽化器
            故障."),nl.
explain(7) :-nl,
        write("因为您的车存在点火系统故障,"),nl,
        write("且火化塞正常,"),nl,
        write("且发动机运转不平稳,"),nl,
        write("因此,根据规则 7 得出,您的车的点火线路有问
            题."),nl.
explain(8) :-nl,
        write("因为您的车存在点火系统故障,"),nl,
        write("且火化塞不正常,"),nl,
        write("因此,根据规则 8 得出,您的车的火花塞有问
            题."),nl.
explain(9) :-nl,
```

```
        write("因为您的车仪表盘不亮,"),nl,
        write("且照明灯不亮,"),nl,
        write("因此,根据规则 9 得出,电气系统有故障."),nl.
explain(10) :-nl,
        write("因为您的车仪表盘亮,"),nl,
        write("且发动机转动,"),nl,
        write("且点火火花正常,"),nl,
        write("且不来油或油不畅,"),nl,
        write("因此,根据规则 10 得出,供油系统故障."),nl.
explain(11) :-nl,
        write("因为您的仪表盘亮,"),nl,
        write("且发动机转动,"),nl,
        write("且油箱中油量正常,"),nl,
        write("且点火火花不正常,"),nl,
        write("因此,根据规则 11 得出,点火系统故障."),nl.
```

3.5.2 对推理过程的解释

在 AUTO-FWFDES 系统中,对推理过程的解释,主要是跟踪系统目前正企图匹配的规则,由这条规则产生解释语句.具体的程序编码为

```
positive(X,Y,M) :-not(negative(X,Y)),!,
        ask(X,Y,M).
ask(X,Y,N) :-write("提问:-您的车是否有……",Y,"?"),
        readint(Reply),
        Reply()2,!,
        remember(X,Y,Reply).
ask(X,Y,N) :-diag_list(N,Fault),
        write("您的车可能有故障—",Fault),nl,
        write("而是否存在征兆—",Y),
        write("直接关系到是否具有故障—",Fault),nl,
```

```
write("因而必须回答,是否有此征兆 1/0? 响应:"),
readint(Reply),
remember(X,Y,Reply).
```

3.6 用户界面设计

在早期的传统诊断专家系统如 MYCIN 中,用户界面是专家系统与用户进行信息交换的唯一窗口,它的设计应首先考虑到如何最大限度地方便用户,使用户易于操作,即为用户提供友善的人机接口.比如说设计出各种各样的菜单,使用户在选择系统功能时只要敲入数字键或用箭头甚至鼠标就可以选择执行系统所具有的各项功能,在对系统的询问进行响应时,用户用 Y/N 或 1/0 来表示是/否具有该征兆等;其次,用户应能识别并处理各种输入错误,例如当需要输入 Y/N 时,用户却键入了 1/0 等;最后,一个友好的用户界面应能支持用户与专家系统之间的流畅对话,系统在对话的基础上往往还需要给出一些恰如其分的解释.这里,只举一个简单的例子来说明前两点.

在系统 AUTO-BWFDES 中有这么一段程序:

```
repeat.
repeat : -repeat.
show_menu : -repeat,
    clear_facts,
    clearwindow,nl,nl,nl,
    write("   ****************************"),nl,
    write("   *      请选择系统功能      *"),nl,
    write("   *                          *"),nl,
    write("   *        1  故障诊断       *"),nl,
    write("   *        2  知识库维护     *"),nl,
    write("   *        3  退出系统       *"),nl,
    write("   ****************************"),nl,nl,
```

```
    write("  请选择  1,2,3,...  "),
    readint(Choice),
    Choice<4,
    Choice>0,
    process(Choice),
    Choice=3,!.
process(3) :-removewindow,
    exit.
process(1) :-clearwindow,
    consult("car.dba"),
    do_diagnosing.
process(2) :-clearwindow,
    consult("car.dba"),
    show_menu2.

ask(BNO,Z) :-nl,write("问题:-",Z,"吗?"),
    makewindow(2,15,-7,"用户响应窗口",16,40,4,32),
    write("肯定,否定一键入 1,0"),nl,
    readint(Response),
    do_answer(BNO,Response),
    clearwindow,
    shiftwindow(1).

show_menu2 :-clearwindow,nl,nl,nl,nl,
    write("  扩展故障知识库"),nl,nl,
    write("  1  扩展故障主题"),nl,
    write("  2  扩展故障规则"),nl,
    write("  3  扩展征兆事实"),nl,
    write("  4  删除故障主题"),nl,
    write("  5  删除故障规则"),nl,
```

```
    write("   6   删除征兆事实"),nl,
    write("   7   退出系统"),nl,nl,
    write("       请选择 1 - 7"),
    nl,write("您的选择是:      "),
    readint(Choice),
    process1(Choice),
    save("car.dba"),
    show_menu2.
.....
process1(_) :-show_menu2.
```

在上面的程序片段中,前面和后面的语句都是为了设计菜单,可以利用数字键来选择执行各种不同的功能,repeat 是为了重复执行系统的功能,对变量 Choice 的判断和谓词 process(_)为出错处理. 谓词 ask 为接受用户的征兆输入.

这样,进入上面的程序段后,系统执行主菜单,会出现下面的窗口:

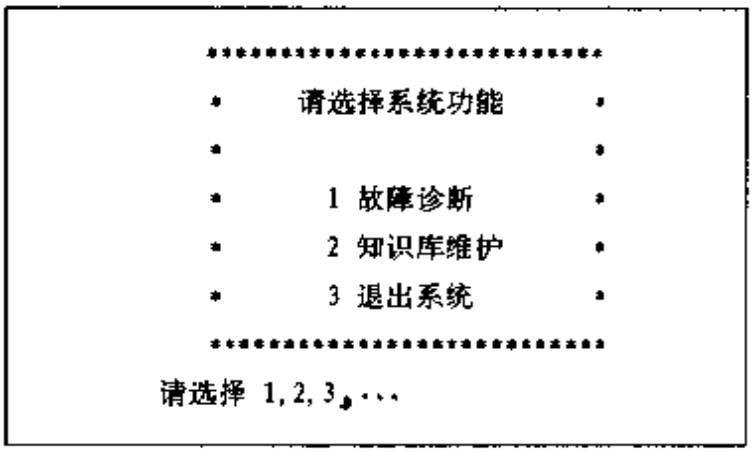


图 3.9 系统 AUTO-BWFDES 的主菜单

假设现在用户选择功能 2,系统执行维护知识库的子菜单,出现下面的窗口:

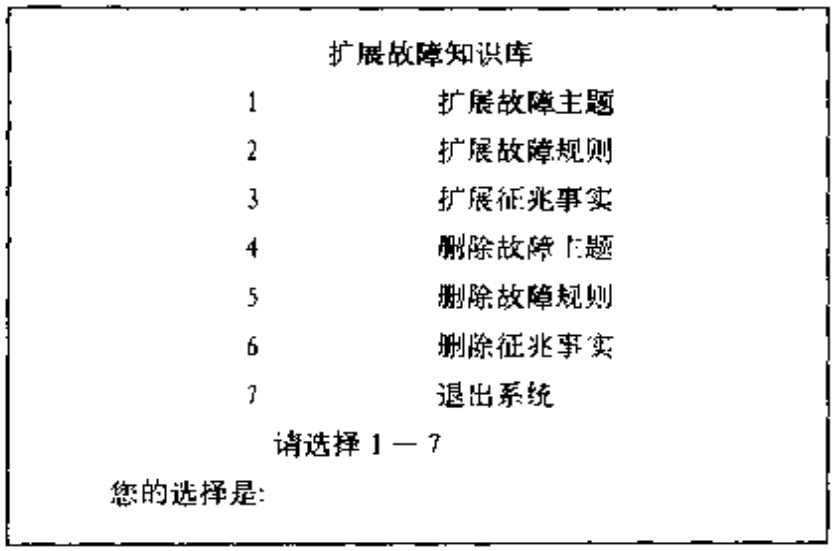


图 3.10 系统 AUTO-BWFDES 维护知识库的子菜单

在接受用户的征兆输入时，系统出现如下的窗口：

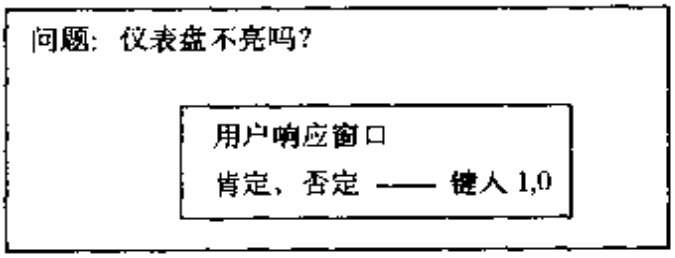


图 3.11 系统 AUTO-BWFDES 接受用户输入的窗口

现在，随着专家系统的发展和数据采集水平的提高，在一些实用的专家系统中越来越多的征兆信息是通过数据采集仪来获取，这样做有几个优点：

1) 方便了用户. 对于一个复杂的诊断对象，征兆信息相当多，如果每一个征兆数据都是通过人机接口的方式来获取，这对于用户来讲无疑是一件痛苦的事情，而使用数据采集仪，诊断系统可以根据诊断推理的需要，自动去采集数据，并经过一定的数据处理方式处理后经全局数据库送往推理机.

2) 加快了诊断速度. 在诊断过程中，通过人机接口获取诊断数据要耗费大量的时间，而通过数据采集仪可以将这部分时间降

低很多.随着数据采集仪的大量使用,在线故障诊断也就成为可能.

3) 提高了诊断精度.纯碎的人机接口会带来许多人为的误差因素,如:由于疲劳或错误操作而导致的误输入,各种测量仪表读取时的错误等等;与之相反的是,数据采集设备可以不知疲倦地准确工作.

当然,在诊断专家系统中,即使对于征兆数据的输入,也不能完全不使用人机接口,这至少有两方面的原因:数据采集仪可能不能采集所有的征兆数据(也可能是为了避免系统代价过高或防止系统太复杂的原因);数据采集仪特别是传感器部分可能会损坏.

3.7 基于规则的专家系统中的不精确推理

3.7.1 不精确推理理论

在前面几节的讨论中,诊断系统中的征兆、规则和推理都是精确表示的.例如:在基于规则的专家系统中,一个征兆要么存在,要么不存在;当规则右边的前提是绝对存在时,则左边的结论也是绝对存在的等.

然而在现实实际中,事物的特征并不总是表现出明显的是与非,同时由于还存在着其它原因,如概念模糊、知识本身存在着可信度等等,使得在专家系统中往往要使用不精确推理方法.不精确推理又称为似然推理,就是根据证据的不确定性和知识的不确定性来求出结论的不确定性的一种推理方法.它是专家系统中常用的推理方法,比精确推理要复杂得多.在这一节中,只讨论如何在基于规则的专家系统中实现不精确推理.

在一个诊断专家系统中,不确定性主要表现在以下几个方面:

(1) 征兆的不确定性

征兆的不确定性表现在三个主要方面:概念的模糊性,在故障诊断中,某种征兆的概念往往具有模糊性,比如:汽车发动机诊断中的尾气颜色是否正常便带有模糊性;测量的不精确性,这主要是

指测量值与真实值之间的差别，它往往由测量仪表的准确性和一些人为因素造成；随机性，对于一个诊断对象，它的某一种征兆可能会在这次诊断中出现，而在下一次诊断中不出现.

(2) 规则的不确定性

规则的不确定性是指在规则的前提成立的条件下，规则结论成立的不确定性. 例如：在汽车诊断专家系统中，可以使用规则

battery(dead,0.6) : -ignition(wont_start).

表示在不能点火的情况下，故障原因为电池损坏的可能性是 0.6. 说明即使点火失败，也不能肯定是由电池损坏所致，而只能有60%的把握.

(3) 推理的不确定性

推理的不确定性是指在推理过程中，知识不确定性的动态积累和传播. 例如：现有规则和征兆事实：

If A and B then C with PB1.	如果征兆 A 和 B 存在，则故障为 C 的可能性为 PB1
A(PB2)	征兆 A 存在的可能性为 PB2
B(PB3)	征兆 B 存在的可能性为 PB3

现在，如何就目前 A，B 存在的可能性和规则中的不确定性因子求出 C 存在的可能性.

由这三种不确定性可以知道：为了实现不精确推理，必须完成以下的三步工作：

1) 确定征兆的不确定性. 以 $C(E)$ 表示征兆的不确定性，表示征兆 E 为真的程度.

2) 确定规则的不确定性. 以 $f(H,E)$ 表示规则的不确定性，它表示在征兆 E 以程度 1 存在的前提下，结论 H 存在的程度. $f(H,E)$ 又称为规则强度.

3) 确定推理不确定性的各种算法. 在基于规则的专家系统中，每一条规则内部只存在“and”运算，规则与规则之间只存在

"or"运算，推理中的不确定性算法只有两种.

a) 根据规则前提 E_i 的不确定性 $C(E_i)$ 和规则强度 $f(H,E_1,E_2,\cdots)$ 求得结论 H 的不确定性 $C(H)$，即定义函数 g_1，使之满足：

$$C(H)=g_1[C(E_1),C(E_2),\cdots,f(H,E_1,E_2,\cdots)] \quad (3.7.1)$$

这种运算即是"and"运算.

b) 在专家系统中，如果对于同一个结论的几条规则同时被激活，那么这时计算结论成立的可能性因子必须考虑所有被激活规则的可能性因子. 即需要根据每条规则的不确定性因子 $C_1(H)$，$C_2(H)$，…，求出它们的组合所导致的结论 H 的不确定性 $C(H)$，即定义函数 g_2，使之满足：

$$C(H)=g_2[C_1(H),C_2(H),\cdots] \quad (3.7.2)$$

这种运算即是"or"运算.

例如：在汽车诊断专家系统中，如果存在事实：

```
ignition(wont_start,1.0)
radio(wont_play,1.0)
```

那么规则

```
battery(dead,0.2) :-ignition(wont_start,1.0)
battery(dead,0.5) :-radio(wont_play,1.0)
```

同时被激活，这时必须利用上面两条规则的每一个可能性因子(0.2,0.5)来计算电池损坏的可能性因子. 直观告诉我们，这个因子应该不小于 0.5.

3.7.2 基于规则的专家系统中不精确推理的 PROLOG 语言的实现

在专家系统中实现不精确推理，其实质是用 PROLOG 语言来表达征兆、规则和推理的不确定性.

(1) 征兆不确定性的表示

我们可以通过在用 PROLOG 表示的每一个征兆后面增加一个概率因子来表示这个征兆存在的不确定性，如：

```
battery(dead,0.03)
```

表示电池坏的可能性为 0.03.

(2) 规则的不确定性表示

与征兆不确定性表示相似,规则的不确定性表示也是只需要在规则的前提部分和结论部分各增加一个概率因子,如

battery(dead,0.2) : -ignition(wont_start,1.0)

表示规则:如果不能点火,则电池坏的可能性是 0.2. 当然对于不同的征兆引起同一种故障的概率是不同的,如我们还可以有下面的规则

battery(dead,0.5) : -radio(wont_play,1.0)

表示:当收音机不响时,电池坏的可能性为 0.5.

(3) 推理的不确定性表示

在讨论推理的不确定性之前,有必要先简单介绍一下概率的相关性. 对于两个事件 A 和 B,设它们独立发生的概率为 $P(A)$和 $P(B)$,它们的相关性有三种情况:

最大相关:两个事件同时发生的概率最大,只要概率小的事件发生,则概率大的事件必定发生. 满足:

$$P(A \text{ and } B)=\min(P(A),P(B))$$

$$P(A \text{ or } B)=\max(P(A),P(B))$$

最小相关:两个事件同时发生的概率尽量小,如果两个事件发生的概率之和小于 1,则两个事件同时发生的概率为 0. 满足:

$$P(A \text{ and } B)=\max(0,P(A)+P(B)-1)$$

$$P(A \text{ or } B)=\min(1,P(A)+P(B))$$

相互独立:事件 A 的发生和事件 B 的发生互不相关. 满足:

$$P(A \text{ and } B)=P(A)P(B)$$

$$P(A \text{ or } B)=P(A)+P(B)-P(A)P(B)$$

图 3.12 反映了这三种情况下事件 A 和 B 的相关性.

为方便起见,这里仅讨论互不相关的情况(这在故障诊断中是极为普遍的情况). 在这种情况下,上面的概率公式可推广到多个事件的情况.

$$P(A \text{ and } B \text{ and } C \text{ and } \cdots) = P(A)P(B)P(C)\cdots$$

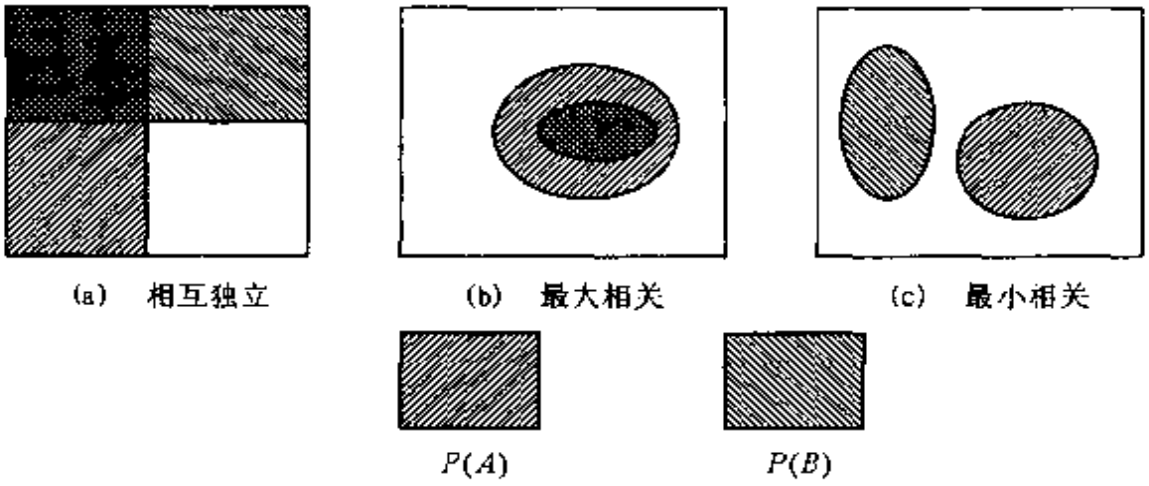


图 3.12 两个事件的概率相关图

$$P(A \text{ or } B \text{ or } C) = P(A) + P(B) + P(C) - P(A)P(B) - P(A)P(C) - P(B)P(C) + P(A)P(B)P(C)$$

$$P(A \text{ or } B \text{ or } C \text{ or } \cdots) = 1 - [(1 - P(A))(1 - P(B))(1 - P(C))\cdots]$$

有了上面的讨论，我们可以确定上一小节中的两个函数 g_i，$i=1,2$，并可将式(3.7.1)改写为

$$C(H) = C(E_1)C(E_2)\cdots f(H, E_1, \cdots) \qquad (3.7.3)$$

将式(3.7.2)改写为

$$C(H) = 1 - [(1 - C_1(H))(1 - C_2(H))\cdots] \qquad (3.7.4)$$

1) "and"运算的 PROLOG 实现. 可以在规则的后面增加一个计算概率因子的公式即可. 如

```
battery(dead, P) : -electrial_problem(P2)
                    light(dim, P3)
                    P = P2 * P3 * 0.5
```

0.5 为规则强度.

2) "or"运算的 PROLOG 实现. 用 PROLOG 语言实现"or"运算包括两步：

a) 将激活规则的概率因子存入一个表中；这只要在每一条规

则的后面增加一条语句，该语句实现将概率因子存入表中. 如：

```
rule :- …,
    append_list(L,P)
```

b) 对这个表进行运算，计算最后的结果. 如：

```
or([P],P)
or([P|PL],Ptotal) :-or(PL,P2)
    Ptotal=1-((1-P)*(1-P2))
```

3.8 搜索策略

专家系统的推理机在进行匹配操作时，会出现三种可能的结果：

只有一条规则匹配成功，这是最为理想的情形，余下的事情只需验证这条规则的其它前提是否成立，若都成立，则这条规则最终被“激活”.

没有一条规则匹配成功，造成这种情形的原因有多种，可能是由于在前面执行问题求解过程时选择的路径不对，因而需要重新回溯，也有可能是因为知识库中的知识不全没有包含目前这一种情形等原因.

第三种情形是有两条以上的规则匹配成功，成为竞选规则. 这时需要建立一种原则，对这些规则进行排序，以选取其中一条来执行，这一过程被称为问题求解过程(或冲突消解过程)，所建立的原则称为问题求解策略(或冲突消解策略).

从广义上讲，问题求解包含全部计算机科学，因为任何一个计算任务都可被看作一个要解答的问题. 不过，在人工智能研究中，不把例行的计算方法包含在问题求解过程中.

研究专家系统中的问题求解方法是人工智能的重要课题之一. 问题求解的目标是寻找最好的搜索技术，它能生成一条有效的解题途径. 搜索策略分为两大类：盲目搜索和启发式搜索. 盲目搜索又称为弱搜索，这类搜索方法不使用智能决策. 在搜索过程中不

需要前后相关的或有关问题域的专门信息.如果盲目搜索是在一个大的状态空间中进行,机时的开销将很大.

启发式搜索需要分析问题域的专门信息(即启发式知识),并因此而缩小了搜索空间.这些前后相关的信息或有关问题域的信息用于:

1) 确定下一步搜索哪一条路径;

2) 确定后续搜索路径(节点);

3) 将全部搜索空间中的部分节点标上无需搜索的记号.

启发式搜索规则并不能总是有效地缩小状态空间,在不能缩小状态空间的情况下,它也就变为盲目搜索.

下面讨论几种主要的盲目搜索和启发式搜索方法,本节所有的搜索图中 A 为搜索的起始点,G 为搜索的目标点,虚线为搜索路径.

3.8.1 盲目搜索

盲目搜索的常用方法有穷尽式搜索、宽度优先搜索和深度优先搜索三种.

穷尽式搜索方法是沿着决策网络中的每一条可能的途径进行搜索.由于它的时间开销很大,这种方法只适用于那些将精确性放在最重要的位置而不考虑时间的问题中.

宽度优先搜索方法(breadth-first search)如图 3.13 所示,它按照一层一层的步骤来搜索即首先搜索与起始节点直接相连的下一层节点,再搜索所有与下层节点直接相连的更下层的节点,依次类推.

在宽度优先搜索法中,由于每一步要搜索一层,因此需要保存好已搜索到的这一层各个节点的状态,议程(agenda)就是用来保存这些状态的.议程中的每一个状态节点的下续节点还没有找到,每个都表示下一步要做的工作.

宽度优先搜索法实际上也考虑了搜索中可能出现的各种情形,因此只要问题有解,采取该方法就一定能以最短的路径搜索到

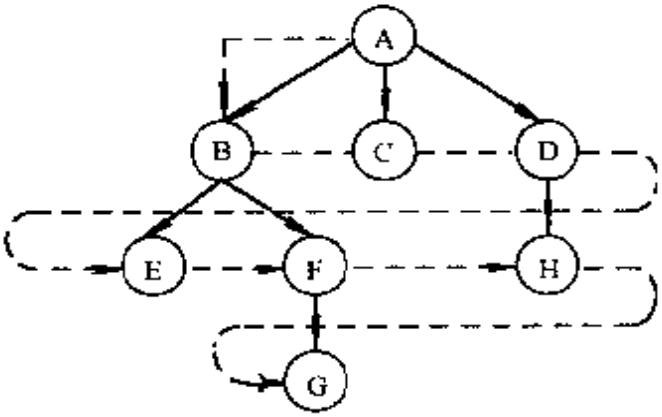


图 3.13　宽度优先搜索法

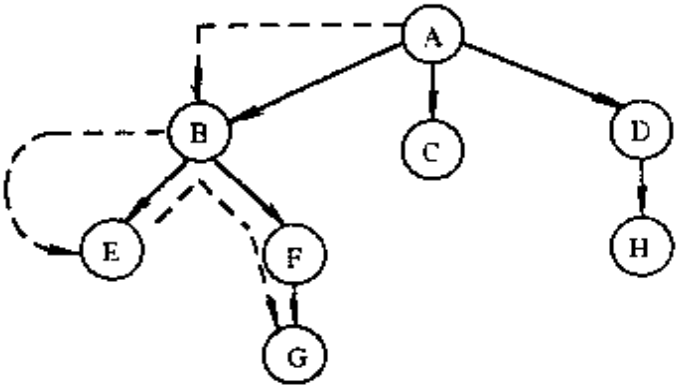


图 3.14　深度优先搜索法

这个解.

宽度优先搜索法的缺点是随着搜索深度的增加，下一步的搜索节点可能会指数增长，因而所耗费的时间将是巨大的.

为了克服宽度优先搜索的弱点，深度优先搜索方法(depth-first search)采用如图 3.14 所示的搜索方向，它每次总是搜索深度大的节点，即沿着一个节点的分支纵向搜索，直到找到目标结点，如没找到，再搜索另一个节点的分枝，依次类推.

深度优先搜索方法主要存在两个缺点：有可能会出现无穷递归的情况，从而搜索不到需要的解；即使搜索到也极可能不是最短路径.

系统 AUTO-FWFDES 和系统 AUTO-BWFDES 都是采用深

度优先搜索.

3.8.2 启发式搜索

在介绍各种具体的启发式搜索方法之前,先介绍两个用于启发式搜索的函数:代价函数和估价函数.代价函数(cost function)是指从搜索起点到当前点所耗费的代价;估价函数(evalution function)是指从当前点到搜索终点所将要耗费的代价的预测函数.代价函数和估价函数构成启发式搜索的"启发式信息".从本质上说,启发式搜索就是寻找一种搜索方法,使得代价函数或估价函数为最小或者它们的和为最小.

启发式搜索大都是在深度优先搜索和宽度优先搜索的基础上发展起来的.启发式搜索的方式比较多,较常用的有爬山法(hill-climbling 又称 optimization)、最好优先法(best-fist)、分枝界限法(branch-and-bound)、A^*算法(A^*algorithm)、生成测试法(generate-and-test)等.

爬山法是在深度优先搜索法的基础上,采用估价函数而产生的启发式搜索方法.它不是绝对按照一个次序穷尽搜索全部路径,而是依照下面步骤代价的高低来选择最可能接近目标的路径.

最好优先搜索法是在宽度优先搜索法的基础上,采用估价函数而形成的启发式搜索方法.如同宽度优先搜索方法,它也需要一个议程.对于每一个状态节点,下一步执行的是议程中具有最小估价函数的节点.

分枝界限法如同最好优先搜索法,只不过它采用的是最小代价,而不是最小估价.它也具有一个议程.对于每一个状态节点,下一步执行的是议程中与初始节点具有最小代价函数的节点.

A^*算法是最好优先搜索法与分枝界限法的结合.它也具有一个议程.对于每一个状态节点,下一步执行的是议程中具有最小代价函数与估价函数之和的节点.

表 3.2 列出了这六种方法的比较.

表 3.2 六种搜索方法比较表

搜索算法名	议程	估价函数	代价函数	搜索的下一个节点
深度优先搜索法	无	无	无	当前状态的下一节点或上一状态的下一节点
宽度优先搜索法	有	无	无	议程中的最前面的节点
爬山法	无	有	无	当前状态的下一节点中估价函数最低者
最好优先搜索法	有	有	无	议程中具有最低估价函数的节点
分枝界限搜索法	有	无	有	议程中具有最低代价函数的节点
A*算法	有	有	有	议程中具有最低估价和代价和的节点

3.9 专家系统开发工具

在早期的专家系统开发研制中，开发者们在开发过程中发现了一些问题：

1）研制时间很长. 为了开发一个专家系统，往往要耗费多人多年的时间，如 MYCIN 系统的开发用了几十人年的时间(始于 1972 年，基本完成于 1974 年).

2）重复性劳动多. 即使对于相近的系统，开发者们也得做很多重复性的工作.

3）需要许多人尤其是领域专家与知识工程师的长期合作，从而使得开发工作变得极其复杂，开发出来的专家系统的质量也可能受到影响.

为了缩短专家系统的开发周期，降低专家系统的开发成本，提高专家系统的性能，研究者们在系统与知识的相互独立、推理机制、开发效率、系统通用性和系统质量等方面做了很多的研究，研制出了不少的开发工具，在开发工具上开发专家系统可大大降低难度. 如：在骨架系统 EMYCIN 基础上开发 SACON(用于结构分析)仅用了四人月工作量，开发 CLOT(用于血液凝结病诊断)更是只用了 60 个小时.

开发工具按其功能可分为四类：通用程序设计语言、骨架系

统、通用知识工程语言和专家系统开发环境.

通用程序设计语言是开发专家系统的最基本的也是最早的工具,典型的通用程序设计语言有两种:符号处理语言 LISP 和逻辑语言 PROLOG,用这两种语言可以极为方便地表示专家系统的知识和设计推理机制,从而大大地减少设计专家系统的工作量.此外,其它的一些计算机语言如面向对象的 C^{++},传统的 C,Pascal 等都可以用来开发专家系统.

由于通用程序设计语言在其它范畴也广为应用,因而在此就不做更详细的介绍,下面讨论其它的三种开发工具.

3.9.1 骨架系统

骨架系统是将一个被实践证明为行之有效的专家系统删去其特定的领域知识而留下它的系统框架所形成的系统.骨架系统完全继承了原专家系统中所具有的知识表示方法、推理机制、知识库结构以及全部辅助工具.因此,利用一个骨架系统来建造新的专家系统,只需要把新专家系统所工作的领域中的知识按照骨架系统要求的知识表示方式输入到知识库中即可.之所以能从专家系统中分离出骨架系统就是因为专家系统中推理机与知识库的完全分离所决定的.下面是两个经典的骨架系统:

EMYCIN 系统是 MYCIN 系统的开发者们将 MYCIN 系统中的与感染病有关的内容剔除后再结合 TEIRSIAS 系统的部分成果而形成的框架系统.

KAS 是一个知识获取系统(knowledge acquisition system),它是由 PROSPECTOR 除去领域知识形成的框架系统,它为知识工程师提供了便于开发调试维护知识库的工具.

骨架系统虽然可以大大简化建造另一个专家系统的过程,但是它缺乏通用性和灵活性,限制了专家系统的构造者的设计选择,因此,某一个骨架系统只适合于某一类特定的问题领域,而不能为其它领域采用.

3.9.2 通用知识工程语言

通用知识工程语言是专门用于构造和调试专家系统的通用设计语言，它能处理不同领域和不同类型的问题，并能提供各种控制结构，用通用知识工程语言设计推理机和知识库，比用一般的人工智能程序设计语言如 LISP 和 PROLOG 等更为方便．由于它并不和特定的结构和方法紧密联系，因而比骨架系统更加灵活．

为说明通用知识工程语言的优势，现以 OPS-5 的特点为例加以描述：

1）OPS-5将通用的表示和控制结合起来，提供了专家系统中

表 3.3 通用知识工程语言

名称	特　征	实现语言	研制者	时间
OPS-5	基于规则表示、正向链	FRANZ LISP	Carnegie Mellon	1977
UNITS	框架	InterLisp	Standford	1979
RLL	框架	InterLisp	Standford	1980
ROSIE	规则，面向过程，正向链，类英语语法	InterLisp，C	Rand	1981
ROSS	面向对象、规则	FRANZLISP	Rand	1982
MRS	规则、逻辑	LISP	Standford	1982
LOOPS	面向对象，规则，正向过程，面向访问	InterLisp	Xeror	1983
SRL+	框架，逻辑，规则，面向对象	LISP	Carnegie 公司	1983
S1	规则，框架，面向过程，反向链，黑板	InterLisp	Tekowledge	1984
M.1	规则，反向链，类英语语法，S1子集	PROLOG	Tekowledge	1984
KEE	规则，框架，面向过程，正，反，双向链	InterLisp	Intellicorp	1984
ART	规则，框架，面向过程，正，反，双向链	Lisp，C	Interfere 公司	1984
CLIPS	规则，面向过程，正向链	C	NASA	1985

所需的基本机制，并不偏向某些特定的问题求解策略或知识表示模式.

2) OPS-5 允许程序设计者使用符号表示并表达符号之间的关系.但并不事先定义符号与关系之间的含义，这些含义完全由程序设计者设计的产生式规则确定.

3) OPS-5 解释程序的控制机制是一个称作认识动作(recognize-act)周期的简单循环，这一循环可按用户的意愿进行尽情的发挥.

4) OPS-5 提供了一种单一的全局数据库，也就是工作存储区.

从上面的四条可以得出：用通用知识工程语言构造一个专家系统所需的工作量要比用通用程序设计语言构造同一个专家系统所需的工作量要小，比骨架系统所需的工作量大.但它比骨架系统灵活，并具有更大的通用性，而不界限于某一特定的领域.

表 3.3 列出了常用的通用知识工程语言.

3.9.3 专家系统开发环境

专家系统开发环境是以一种或多种工具和方法为核心，加上与之配套的各种辅助工具和界面环境形成的完整的集成系统，这种环境可提供多种类型的推理机制和多种知识表示方法，帮助专家系统建造者选择结构，设计规则语言和使用各种组件，使之成为一个完整的专家系统. Standford 大学的 AGE 是一种典型的代表.

AGE 是一种在一定新概念和新技术基础上为用户提供一套设计、构造和测试多种类型专家系统的预构件系统，它在一个主要方面与骨架系统不同，其目标是提供一种环境，使用户可以从中选择并指定多种知识表示和处理方式.

AGE 为用户提供了一个通用的专家系统结构的框架，并将该框架分解为许多在功能和结构上较为独立的组成部件，这些组件已预先编制成标准模块存放在系统中.用户可通过两条途径构造

自己的系统：

1）用户使用 AGE 中已编制好的各种组件作为构造材料，非常方便地来组合设计自己所需的系统.

2）用户通过 AGE 所提供的工具界面，定义和设计各种所需的组成部件，以构造自己的专家系统.

为了指导用户使用 AGE 提供的结构框架，简便地构造专家系统，AGE 设立了一个智能辅助系统，其本身也是一个基于知识的程序系统. 它通过与用户交互的方式指导用户如何使用 AGE 中提供的各种工具和构造材料去构造专家系统.

为此，AGE 的开发可分为两部分：1）构造通用的专家系统结构框架；2）建立指导使用这个框架的智能机制.

附录　传统故障诊断专家系统 TURBO PROLOG 语言程序集

在本附录中共有两个故障诊断专家系统：AUTO-FWFDES（采用正向推理的汽车故障诊断专家系统）和 AUTO-BWFDES（采用反向推理的汽车故障诊断专家系统）. 有三个文件，其中 FWFDES. PRO 就是 AUTO-FWFDES 系统；文件 CAR. DBA 是汽车诊断的知识库，它和文件 BWFDES. PRO 一起组成 AUTO-BWFDES 系统. 在 AUTO-FWFDES 系统的知识库中采用规则的知识表示方法，推理方向为正向，系统可以对诊断结果和诊断过程作出解释；在 AUTO-BWFDES 系统的知识库中采用谓词逻辑的知识表示方法，推理方向为反向，系统可以对知识库进行维护.

```
/**********************************************/
/**  汽车故障诊断专家系统 AUTO-FWFDES  **/
/**              文件名 FWFDES.PRO          **/
/**              采用正向推理方法           **/
/**          本程序具有诊断和解释功能       **/
/**          用 TURBO PROLOG 2.0 编制       **/
/**********************************************/

domains
/* 定义域 */
    diag_list=number *
    number=integer
    N=integer
    FAULT=symbol
database
/* 数据库谓词 */
```

```
    xpositive(symbol,symbol)
    xnegative(symbol,symbol)
predicates
/*谓词说明*/
    diagnose_is(N,FAULT)
    explain(integer)
    start
    show_menu
    positive(symbol,symbol,integer)
    negative(symbol,symbol)
    ask(symbol,symbol,integer)
    remember(symbol,symbol,integer)
    process(integer)
    clear_facts
    explain_list(diag_list)
    append_list(diag_list,diag_list,diag_list)
    diag_list(integer,symbol)
goal
    start.
clauses
/*子句*/
    start:-makewindow(1,30,-4,"汽车故障诊断专家系统",0,0,25,80),
        nl,nl,nl,nl,
        write(" 欢迎进入本专家系统,本系统能"),nl,
        write(" 采用正向推理方法进行故障诊断"),nl,
        write(" 在进行诊断询问时,请用 1 表示肯"),nl,
        write(" 定的回答,用 0 表示否定的回答"),nl,
        write(" 用 2 表示询问为什么要输入 "),nl,
        write(" * * * * 任意键开始"),
        readchar(_),
        show_menu.

    show_menu:-clearwindow,nl,nl,nl,
```

```
    write(" ***************************** "),nl,
    write(" *      请选择系统功能          * "),nl,
    write(" *                              * "),nl,
    write(" *        1 故障诊断            * "),nl,
    write(" *        2 退出系统            * "),nl,
    write(" ***************************** "),nl,nl,
    write(" 请选择 1,2,... "),
    readint(Choice),
    Choice<3,
    Choice>0,
    process(Choice).
process(2):-removewindow,
    exit,!.
process(1):-clearwindow,
    diagnose_is(N,FAULT),!,
    nl,nl,write("从上面的两条可以知道:"),nl,
    write(" 您的车的故障是",FAULT),nl,
    clear_facts.

process(1):-nl,write("对不起,爱莫能助"),
    clear_facts.

explain_list([ ]).
explain_list([Head|Tail]):-explain(Head),
    explain_list(Tail).
ask(X,Y,N):-write("提问:-您的车是否有.....",Y,"?"),
    readint(Reply),
    Reply<>2,!,
    remember(X,Y,Reply).
ask(X,Y,N):-diag_list(N,Fault),
    write("您的车可能有故障    -",Fault),nl,
    write("而是否存在征兆——",Y),
    write("直接关系到是否具有故障——",Fault),nl,
```

```
        write("因而必须回答,是否有此征兆 1/0? 响应:"),
        readint(Reply),
        remember(X,Y,Reply).

    append_list([],L,L).
    append_list([N1|L2],L3,[N1|L4]):-append_list(L2,L3,L4).

    /* 正向推理机 */

    positive(X,Y,M):-xpositive(X,Y),!.

    positive(X,Y,M):-not(negative(X,Y)),!,
        ask(X,Y,M).

    negative(X,Y):-xnegative(X,Y),!.

    remember(X,Y,1):-asserta(xpositive(X,Y)).
    remember(X,Y,0):-asserta(xnegative(X,Y)),
        fail.

    clear_facts:-retract(xpositive(_,_)),
        fail.
    clear_facts:-retract(xnegative(_,_)),
        fail.
/*************************************************************/
/*** 简易正向推理汽车故障诊断专家系统知识库及解释内容  ***/
/*************************************************************/

diagnose_is(1,"电池没电"):-
    diagnose_is(9,"电气系统故障"),
    not(positive(has,"发动机转动",1)),
    positive(has,"电池接线正常",1),
    append_list([9],[1],L),
```

```
    explain_list(L).
diagnose_is(2,"电池接线故障"):-
    diagnose_is(9,"电气系统故障"),
    not(positive(has,"电池接线正常",2)),
    append_list([9],[2],L),
    explain_list(L).
diagnose_is(3,"油箱没油"):-
    diagnose_is(10,"供油系统故障"),
    not(positive(has,"油箱中油量正常",3)),
    append_list([10],[3],L),
    explain_list(L).
diagnose_is(4,"汽油油路故障"):-
    diagnose_is(10,"供油系统故障"),
    not(positive(has,"汽油泵入口处有汽油",4)),
    positive(has,"油箱中油量正常",4),
    append_list([10],[4],L),
    explain_list(L).
diagnose_is(5,"汽油泵故障"):-
    diagnose_is(10,"供油系统故障"),
    positive(has,"汽油泵入口处有汽油",5),
    not(positive(has,"汽油泵出口处有汽油",5)),
    append_list([10],[5],L),
    explain_list(L).
diagnose_is(6,"汽化器故障"):-
    diagnose_is(10,"供油系统故障"),
    positive(has,"汽油泵入口处有汽油",6),
    positive(has,"怠速运转时熄灭",6),
    append_list([10],[6],L),
    explain_list(L).
diagnose_is(7,"点火线路故障"):-
    diagnose_is(11,"点火系统故障"),
    positive(has,"火花塞正常",7),
```

```
    positive(has,"发动机运转不平稳",7),
    append_list([11],[7],L),
    explain_list(L).
diagnose_is(8,"火花塞故障"):-
    diagnose_is(11,"点火系统故障"),
    not(positive(has,"火花塞正常",8)),
    append_list([11],[8],L),
    explain_list(L).
diagnose_is(9,"电气系统故障"):-
    not(positive(has,"仪表盘亮",9)),
    not(positive(has,"照明灯亮",9)).
diagnose_is(10,"供油系统故障"):-
    positive(has,"仪表盘亮",10),
    positive(has,"发动机转动",10),
    positive(has,"点火火花正常",10),
    positive(has,"不来油或油路不畅",10).
diagnose_is(11,"点火系统故障"):-
    positive(has,"仪表盘亮",11),
    positive(has,"发动机转动",11),
    positive(has,"油箱中油量正常",11),
    not(positive(has,"点火火花正常",11)).
explain(1):-nl,
    write(" 因为您的车存在电气系统故障,"),nl,
    write(" 且发动机不转动,"),nl,
    write(" 且电池接线正常"),nl,
    write(" 因此,根据规则 1 得出您车上的电池没电."),nl.
explain(2):-nl,
    write(" 因为您的车存在电气系统故障,"),nl,
    write(" 且电池接线不正常,"),nl,
    write(" 因此,根据规则 2 得出,您的车的电池接线存在问题."),nl.
explain(3):-nl,
    write(" 因为您的车存在供油系统故障,"),nl,
    write(" 且油箱中油量不正常,"),nl,
```

```
    write(" 因此,根据规则 3 得出,您车上的油箱中没油."),nl.
explain(4):-nl,
    write(" 因为您的车存在供油系统故障,"),nl,
    write(" 且汽油泵入口处没有汽油,"),nl,
    write(" 且油箱中油量正常,"),nl,
    write(" 因此,根据规则 4 得出,您车上的汽油油路存在问题."),nl.
explain(5):-nl,
    write(" 因为您的车存在供油系统故障,"),nl,
    write(" 且汽油泵入口处有汽油,"),nl,
    write(" 且汽油泵出口处没有汽油,"),nl,
    write(" 因此,根据规则 5 得出,您车上的汽油泵存在问题."),nl.
explain(6):-nl,
    write(" 因为您的车存在供油系统故障,"),nl,
    write(" 且汽油泵入口处有汽油."),nl,
    write(" 且怠速运转时熄灭"),nl,
    write(" 因此,根据规则 6 得出,您的车的毛病是汽化器故障."),nl.
explain(7):-nl,
    write(" 因为您的车存在点火系统故障,"),nl,
    write(" 且火化塞正常,"),nl,
    write(" 且发动机运转不平稳"),nl,
    write(" 因此,根据规则 7 得出,您的车的点火线路有问题."),nl.
explain(8):-nl,
    write(" 因为您的车存在点火系统故障,"),nl,
    write(" 且火化塞不正常,"),nl,
    write(" 因此,根据规则 8 得出,您的车的火花塞有问题."),nl.
explain(9):-nl,
    write(" 因为您的车仪表盘不亮,"),nl,
    write(" 且照明灯不亮,"),nl,
    write(" 因此,根据规则 9 得出,电气系统故障."),nl.
explain(10):-nl,
    write(" 因为您的车仪表盘亮,"),nl,
    write(" 且发动机转动,"),nl,
    write(" 且点火火花正常,"),nl,
```

```
    write(" 且不来油或油不畅"),nl,
    write(" 因此,根据规则 10 得出,供油系统故障."),nl.
explain(11):-nl,
    write(" 因为您的仪表盘亮,"),nl,
    write(" 且发动机转动,"),nl,
    write(" 且油箱中油量正常,"),nl,
    write(" 且点火火花不正常,"),nl,
    write(" 因此,根据规则 11 得出,点火系统故障."),nl.
diag_list(1,"电池没电").
diag_list(2,"电池接线故障").
diag_list(3,"油箱没油").
diag_list(4,"汽油油路故障").
diag_list(5,"汽油泵故障").
diag_list(6,"汽化器故障").
diag_list(7,"点火电极故障").
diag_list(8,"火花塞故障").
diag_list(9,"电气系统故障").
diag_list(10,"供油系统故障").
diag_list(11,"点火系统故障").
```

==========

```
/*****************************************/
/** 汽车故障诊断专家系统 AUTO-BWFDES   **/
/**        文件名     BWFDES.PRO       **/
/**            采用反向推理方法         **/
/**              知识库基于逻辑         **/
/**   本程序具有诊断和维护知识库的功能  **/
/**      用 TURBO-PROLOG 2.0 编制      **/
/*****************************************/
domains
/*定义域*/
    CONDITIONS=BNO*
```

```
    RNO,BNO,FND=integer
    CATEGORY=symbol
    STR_LIST=string*
database
/*数据库谓词*/
    rule(RNO,CATEGORY,CATEGORY,CONDITIONS)
    symptom(BNO,string)
    topic(symbol)
    yes(BNO)
    no(BNO)
    diag_list(integer,symbol)

predicates
/*谓词说明*/
    /*用户接口谓词*/
    repeat
    start
    show_menu
    do_diagnosing
    show_menu1
    show_menu2
    process(integer)
    process1(integer)
    display1
    display2
    display3
    /*推理机谓词*/
    verify(integer)
    prove(CONDITIONS)
    prove1(integer)
    do_answer(integer,integer)
    clear_facts
    convers(string,STR_LIST)
```

```
    convert(STR_LIST,CONDITIONS)
    ask(integer,symbol)

goal
    start.

clauses
/* 子句 */
    repeat.
    repeat:-repeat.
    start:-makewindow(1,30,-4,"汽车故障诊断专家系统",0,0,25,80),
        nl,nl,nl,nl,
        write(" 欢迎进入本专家系统,本系统能"),nl,
        write(" 采用反向推理方法进行故障诊断"),nl,
        write(" 并能对数据库进行维护,在进行"),nl,
        write(" 诊断询问时,请用 1 表示肯定的"),nl,
        write(" 回答,用 0 表示否定的回答."),nl,
        write(" * * * * 任意键开始."),nl,
        readchar(_),
        show_menu.

    show_menu:-repeat,
        clear_facts,
        clearwindow,nl,nl,nl,
        write(" ****************************"),nl,
        write(" *      请选择系统功能      *"),nl,
        write(" *                          *"),nl,
        write(" *        1  故障诊断       *"),nl,
        write(" *        2  知识库维护     *"),nl,
        write(" *        3  退出系统       *"),nl,
        write(" ****************************"),nl,nl,
        write(" 请选择 1,2,3,... "),
        readint(Choice),
```

```
        Choice<4,
        Choice>0,
        process(Choice).
        Choice=3,!.
process(3):-removewindow,
        exit.
process(1):-clearwindow,
        consult("car.dba"),
        do_diagnosing.
process(2):-clearwindow,
        consult("car.dba"),
        show_menu2.

do_diagnosing:-show_menu1.
show_menu1:-clearwindow,nl,nl,
        write("请输入您怀疑的故障代号"),nl,
        write("  1  电池没电"),nl,
        write("  2  电池接线故障"),nl,
        write("  3  油箱没油"),nl,
        write("  4  汽油油路故障"),nl,
        write("  5  汽油泵故障"),nl,
        write("  6  汽化器故障"),nl,
        write("  7  点火电极故障"),nl,
        write("  8  火花塞故障"),nl,
        write("  9  退出诊断"),nl,
        write("您的选择是:"),
        readint(I),
        verify(I),
        nl,write("**********************"),
        nl,write("您的猜测完全正确"),
        nl,write("任一键进行下一次诊断"),
        nl,write("**********************"),
        clear_facts.
```

```
        readchar(_),
        show_menu1.

verify(9):-exit.
verify(I):-diag_list(I,X),
        rule(RNO1,Y,X,COND1),
        rule(RNO2,_,Y,COND2),
        prove(COND2),
        prove(COND1).
prove([ ]).
prove([Head|Tail]):-prove1(Head),
        prove(Tail).

prove1(BNO):-yes(BNO).
prove1(BNO):-symptom(BNO,Z),
        ask(BNO,Z).

ask(BNO,Z):-nl,write("问题:-",Z,"吗?"),
        makewindow(2,15,-7,"用户响应窗口",16,40,4,32),
        write("肯定,否定 —— 键入 1,0"),nl,
        readint(Response),
        do_answer(BNO,Response),
        clearwindow,
        shiftwindow(1).

do_answer(BNO,1):-assert(yes(BNO)).
do_answer(BNO,0):-shiftwindow(1),
        nl,write("您的猜测错误,请按任意键返回选择"),
        readchar(_),
        fail.

clear_facts:-retract(yes(_)),
        fail.
```

```
clear_facts.

/*************************************************/

show_menu2:-clearwindow,nl,nl,nl,nl,
    write("   扩展故障知识库"),nl,nl,
    write("   1   扩展故障主题"),nl,
    write("   2   扩展故障规则"),nl,
    write("   3   扩展征兆事实"),nl,
    write("   4   删除故障主题"),nl,
    write("   5   删除故障规则"),nl,
    write("   6   删除征兆事实"),nl,
    write("   7   退出系统"),nl,nl,
    write("      请选择 1--7"),
    nl,write("您的选择是：      "),
    readint(Choice),
    process1(Choice),
    save("car.dba"),
    show_menu2.

process1(7):-show_menu.
process1(1):-nl,write(" 请输入故障主题:"),
    readln(Topic_name),
    assertz(topic(Topic_name)).
process1(2):-nl,write(" 请输入规则序号:"),
    readint(Rule_no),
    nl,write(" 请输入上一级故障描述:"),
    readln(Rule_fa1),
    nl,write(" 请输入本级故障描述:"),
    readln(Rule_fa2),
    assertz(diag_list(Rule_no,Rule_fa2)),
    nl,write(" 请输入故障-征兆约束表"),
    nl,write(" 输入数字,用空格间隔,回车结束"),
```

```
        readln(Sentence),
        convers(Sentence,List),
        convert(List,COND),
        assertz(rule(Rule_no,Rule_fa1,Rule_fa2,COND)).
process1(3):-nl,write(" 请输入征兆序号:"),
        readint(Cond_no),
        nl,write(" 请输入征兆描述:"),
        readln(Cond_sym),
        assert(symptom(Cond_no,Cond_sym)).
process1(4):-nl,write(" 目前数据库中的主题集"),nl,nl,
        display1,
        nl,write(" 请输入要删除的故障主题:"),
        readln(Topic_name),
        retract(topic(Topic_name)).
process1(5):-nl,clearwindow,
        write(" 目前数据库中的规则集"),nl,nl,
        write(" 规则号  上级故障  本级故障  征兆表"),nl,
        display2,
        nl,write(" 请输入要删除的规则号:"),
        readint(Rule_no),
        retract(diag_list(Rule_no,_)),
        retract(rule(Rule_no,_,_,_)).
process1(6):-nl,write(" 目前数据库中的征兆集"),nl,nl,
        write(" 征兆序号  征兆描述 "),nl,
        display3,
        nl,write(" 请输入要删除的征兆序号:"),
        readint(Cond_no),
        retract(symptom(Cond_no,_)).
process1(_):-show_menu2.

display1:-topic(Topic_name1),
        nl,write("          ",Topic_name1),
        fail.
```

```
    display1.

    display2:-rule  (RNO1,  CATEGORY1,  CATEGORY2,  CONDI-
            TIONS1),
        nl,write("     ",RNO1,"  "),
        writef("%20s",CATEGORY1),
        writef("%20s",CATEGORY2),
        write("     ",CONDITIONS1),
        fail.
    display2.

    display3:-symptom(BNO1,STRING1),
        nl,write("     ",BNO1,"     "),
        write(STRING1),
        fail.
    display3.

    convers(_,[]).
    convers(Str,[Head|Tail]):-fronttoken(Str,Head,Str1),!,
        convers(Str1,Tail).

    convert([],[]).
    convert([Head|Tail],[Head1|Tail1]):-
        str_int(Head,Head1),
        convert(Tail,Tail1).
```

```
/**********************************************************/
/** 服务于系统 AUTO-BWFDES 的汽车故障诊断知识库  **/
/**                  文件名     CAR.DBA                  **/
/**********************************************************/

rule(1,"汽车","电气系统故障",[7,8])
```

```
rule(2,"汽车","点火系统故障",[9])
rule(3,"汽车","供油系统故障",[10])
rule(4,"电气系统故障","电池没电",[1])
rule(5,"电气系统故障","电池接线故障",[2])
rule(6,"供油系统故障","油箱没油",[3])
rule(7,"供油系统故障","汽油油路故障",[4])
rule(8,"供油系统故障","汽油泵问题",[5])
rule(9,"供油系统故障","气化器故障",[11])
rule(10,"点火系统故障","火花塞故障",[6])
rule(11,"点火系统故障","点火线路故障",[12])
symptom(1,"发动机不转动")
symptom(2,"电池接线不正常")
symptom(3,"油箱中油量不正常")
symptom(4,"汽油泵入口处没有汽油")
symptom(5,"汽油泵出口处没有汽油")
symptom(6,"火花塞不正常")
symptom(7,"仪表盘不亮")
symptom(8,"照明灯不亮")
symptom(9,"点火火花不正常")
symptom(10,"不来油或油不畅")
symptom(11,"怠速运转时熄灭")
symptom(12,"发动机运转不平稳")
topic("汽车")
topic("电气系统故障")
topic("点火系统故障")
topic("供油系统故障")
diag_list(1,"电池没电")
diag_list(2,"电池接线故障")
diag_list(3,"油箱没油")
diag_list(4,"汽油油路故障")
diag_list(5,"汽油泵故障")
```

```
diag_list(6,"气化器故障")
diag_list(7,"点火电极故障")
diag_list(8,"火花塞故障")
```

第四章　神经网络故障诊断专家系统

本章首先讨论传统专家系统存在的局限，用以说明引入神经网络故障诊断专家系统的必要性并给出其总体框架；接着简要地介绍一些神经网络的基本知识，在此基础上以一些实际例子来介绍如何利用前馈多层神经网络具体组建基于神经网络的故障诊断专家系统．同时由于传统专家系统和神经网络专家系统各有优劣，谁也不能取代谁，这样它们之间的关系也就有必要进行讨论；最后，扼要地介绍其它神经网络结构在故障诊断领域中的应用．在本章的末尾附有文中诊断例子的C语言程序．

4.1　从传统专家系统到神经网络专家系统

虽然传统的诊断专家系统在很多领域(如机械、医疗、计算机、航空等)中得到了广泛的应用且取得了不少成果，并且显示出了相当出色的工作能力，在某些方面达到甚至超过了人类专家的工作水平，然而在其开发研制过程中也碰到了不少问题，这些问题至少在目前是难以克服的．诸如：

1) 知识获取的“瓶颈”问题：由前面的讨论我们知道，传统专家系统主要通过两种方法来获取知识．一种是领域知识先由知识工程师从领域专家这儿获得(通过这两者的反复交谈或者前者对后者实际操作的观察)，再由知识工程师输入到知识库中．这种方法造成知识失真的可能性有两方面：一方面，领域专家自己也很难描述自己所拥有的知识，对于具体故障，他们往往只知道如何去解决，却说不出采用这种解决方法的理由，而且，有时他们的知识也有错误成分；另一方面，不同的领域专家的知识可能不一样，甚至互相矛盾，在这种情况下，知识工程师往往显得束手无策．第二种

知识获取的方法是机器学习，但是直到目前为止，机器学习能力是很弱的.

2）知识难以维护：现有的传统专家系统知识库中的知识大都是依靠知识工程师人为输入，在知识库里面往往是简单地堆放在一起.这样，当这个专家系统的知识库中的知识达到成千上万条时，维护与管理就显得十分困难.

3）知识窄台阶：专家系统的工作范围很狭窄，只能对专家知识领域范围内的问题以专家级的水平去解决，但对于专家知识领域范围以外的问题则显得无能为力而不能正确处理，只能得出一些稀奇古怪的结论，即传统专家系统没有联想、记忆、类比等形象思维能力，致命的问题是，系统本身无法判断自己是否工作在专家领域知识范围内.

4）推理能力弱：传统专家系统推理方法一般较为简单，控制策略不灵活，容易出现诸如“匹配冲突”、“组合爆炸”以及“无穷递归”等问题，推理速度慢，效率低.

5）实用性差：由于上面的这些严重缺陷，使得一些专家系统很难进入实用阶段.同时，由于推理速度慢，导致一般的传统专家系统难以适应在线工作要求，只能在离线、非实时条件下工作.

6）不精确推理不适合解决模糊问题：知识和故障征兆往往有一定的模糊性，采集的数据也有模糊性（如测量误差、元件参数的容差等），而传统专家系统的不精确推理是基于概率的，与“模糊”是两个不同的概念.

人工神经网络简称为神经网络（Neural Network，简称 NN），是由大量简单的处理单元（称为神经元）广泛地互相连接而形成的复杂网络，它是对人脑神经网络的某种程度上的简化、抽象和模拟，而不是真实写照.目前研究开发的神经网络模型具有人脑功能的基本特征：学习、记忆和归纳，从而解决了过去基于逻辑与符号处理的人工智能研究中的某些局限性，为人工智能的研究开辟了崭新的途径.

我们知道所谓的故障诊断就是对诊断对象的故障模式进行分类和识别或根据现有的知识和一定的推理机制推断出其故障的所在部位和严重程度.由于神经网络具有处理复杂多模式及进行联想、推测和记忆的功能,因而,它非常适合应用于各种系统的故障诊断.

神经网络与专家系统的结合有两种方法:1)使用神经网络来构造专家系统,即把传统专家系统的基于符号的推理变成基于数值运算的推理,以提高专家系统的执行效率,并解决专家系统的自学习问题;2)将神经网络理解成一类知识源的表达与处理模式,这类模式与其它知识表达方式,如规则、框架等一起来表达领域专家的知识,并面向不同的推理机制.在本章中将集中讨论第一种结合方式.

与传统专家系统相比,基于神经网络的专家系统具有如下几个主要优点:

1)具有统一的内部知识表示形式,任何知识规则都可通过对范例的学习存储于同一个神经网络的各连接权重中,便于知识库的组织与管理,通用性强;知识容量大,可把大量的知识规则存储于一个相对小得多的神经网络中.

2)便于实现知识的自动获取,能够自适应环境的变化.

3)推理过程为并行的数值计算过程,避免了以往的"匹配冲突"、"组合爆炸"和"无穷递归"等问题;推理速度快.

4)具有联想、记忆、类比等形象思维能力,克服了传统专家系统中存在的"知识窄台阶"问题,可以工作于所学习过的知识以外的范围.

5)实现了知识表示、存储和推理三者融为一体,即都由一个神经网络来实现.

神经网络故障诊断专家系统的结构如图 4.1 所示.

在神经网络结构知识这一模块中,实际上也是一个知识库,因而有人称其为辅助库.它主要是用来定义诊断对象的常用术语或名称,如定义输入、隐含、输出神经元的物理意义.

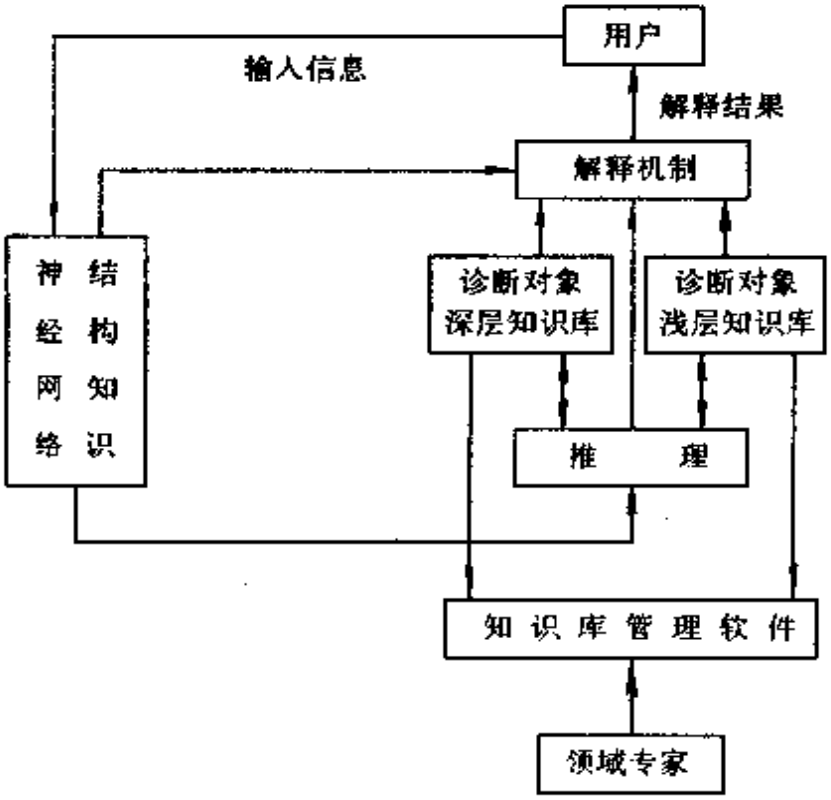


图 4.1 神经网络故障诊断专家系统结构图

注意在上图中，给出的只是神经网络故障诊断的一般结构，在实际的诊断系统中可根据具体情况而有所改变，或增或删一些内容.

4.2 人工神经元模型

神经网络是对人脑神经系统的数学模拟，其目的是学习和模仿人脑的信息处理方式. 人们对神经系统的研究已经有了很长一段历史，早在 19 世纪末，人们就开始认识到，人脑包含着数量大约在 10^{10}—10^{12}之间的神经元，这些神经元存在着复杂的联接，并形成一个整体，使得人脑具有各种智能行为. 尽管在外观形状上，这些神经元各不相同，然而它们都由三个区组成：细胞体、树突、轴突. 如图 4.2 所示：

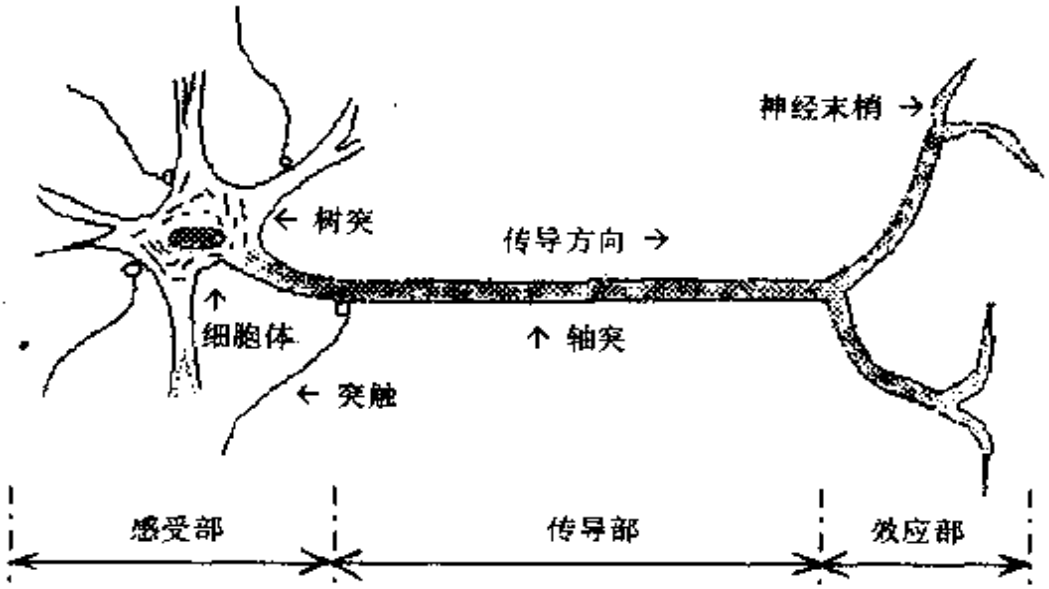


图 4.2　神经元结构图

从图中我们可以看出：一般的神经元有多个树突，它们起感受作用即接受外部(包括其它神经元)传来的信息. 轴突只有一条，与树突相比它显得长而细，用于传递和输出信息. 神经元之间通过突触联结，突触是一个神经元轴突的末梢与另一个神经元的细胞体或树突相接触的地方，每个神经元大约有 10^3—10^4 个突触，换句话说，每个神经元大约与 10^3—10^4 个其它神经元有连接，正是因为这些突触才使得全部大脑神经元形成一个复杂的网络结构. 依据这种理论，我们可以画出神经元的等效模型图 4.3.

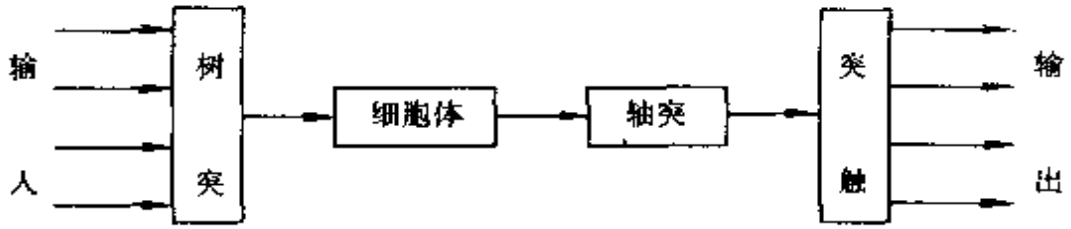


图 4.3　神经元等效模型

从图 4.3 可以看出人脑神经系统的工作原理：外部刺激信号或上级神经元信号经合成后由树突传给神经元细胞体处理，最后由突触输出给下级神经元或作出响应.

基于神经细胞的这种理论知识，自 1943 年 McCulloch 和 Pitts 提出的第一个人工神经元模型以来，人们相继提出了多种人工神经元模型，其中被人们广泛接受并普遍应用的是图 4.4 所示的模型：

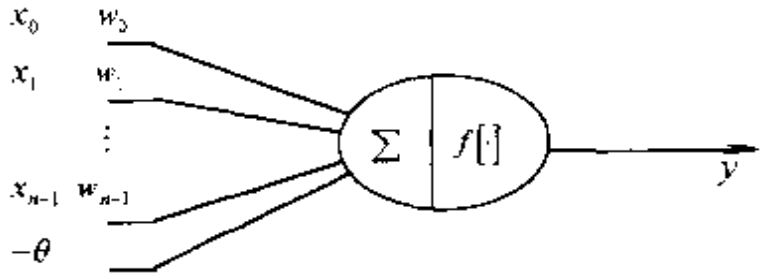


图 4.4　人工神经元模型

图中的 $x_0, x_1, \cdots, x_{n-1}$ 为实连续变量，是神经元的输入，θ 称为阈值（也称为门限），$w_0, w_1, \cdots, w_{n-1}$ 是本神经元与上级神经元的连接权值.

神经元对输入信号的处理包括两个过程：第一个过程是对输入信号求加权和，然后减去阈值变量 θ，得到神经元的净输入 net，即

$$\text{net} = \sum_{i=0}^{n-1} w_i x_i - \theta \tag{4.2.1}$$

从上式可以看出，连接权大于 0 的输入对求和起着增强的作用，因而这种连接又称为兴奋连接，相反连接权小于 0 的连接称为抑制连接.

第二个过程是对净输入 net 进行函数运算，得出神经元的输出 y，即

$$y = f(\text{net}) \tag{4.2.2}$$

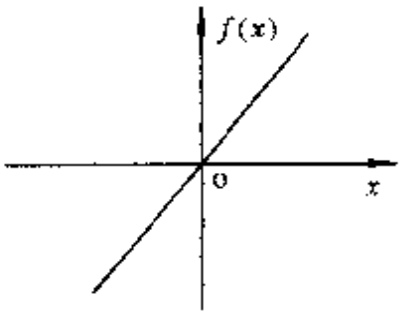
$f[\cdot]$ 通常被称为变换函数（或特征函数），简单的变换函数有图 4.5 所示的几种.

这几种变换函数分别满足如下关系：

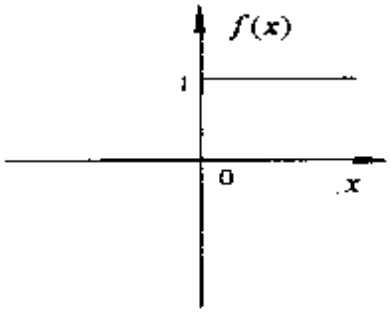
1）线性函数

$$f(x) = kx$$

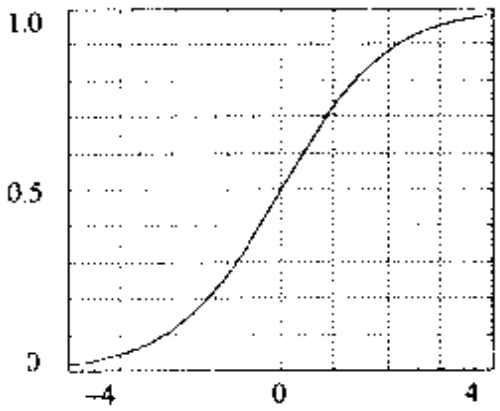
2）阈值函数（硬限幅函数、阶跃函数）



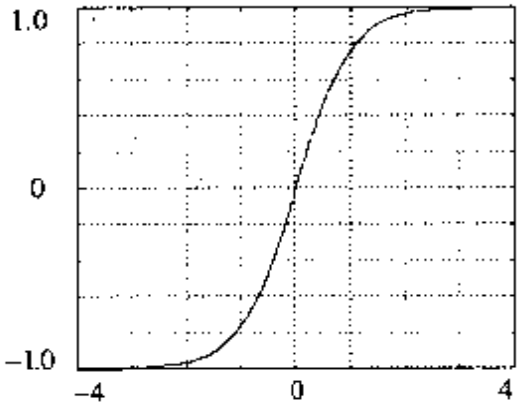
(a) 线性函数



(b) 阈值函数



(c) Sigmoid 函数



(d) 双曲正切函数

图 4.5 神经元的几种常用变换函数

$$f(x)=\begin{cases}1, & x\geqslant 0\\ 0, & x<0\end{cases}$$

3) Sigmoid 函数(S 函数)

$$f(x)=\frac{1}{1+e^{-\lambda x}}$$

在实际的神经网络中,λ 常取为 1.

4) 双曲正切函数

$$f(x)=\mathrm{th}(x)$$

4.3 前向多层神经网络、BP 算法及计算机实现

基于上面介绍的神经元结构,人们又提出了很多种神经网络

结构模型，如 Hopfield 网络、Boltzmann 机、ART 网络和 BAM 网络等. 在故障诊断领域中用得最多也最有成效的是前向多层神经网络. 由于该网络在学习(训练)过程中采用了 BP(Back-Propagation)算法，故有时该网络又称为 BP 网络. 标准的 BP 网络由三层神经元组成，其结构如图 4.6 所示(有一种观点认为图示网络只有两层，其理由是输入层神经元只起数据传输作用，而没有进行任何计算，不影响网络的计算能力，因而在计算网络层数时不应该包括输入层). 最下面为输入层，中间为隐含层，最上面为输出层. 网络中相邻层采取全互连方式连接，同层各神经元之间没有任何连接，输出层与输入层之间也没有直接的连系. 为方便以后的讨论，在此我们假设输入层、隐含层及输出层神经元的个数分别为 L, M, N.

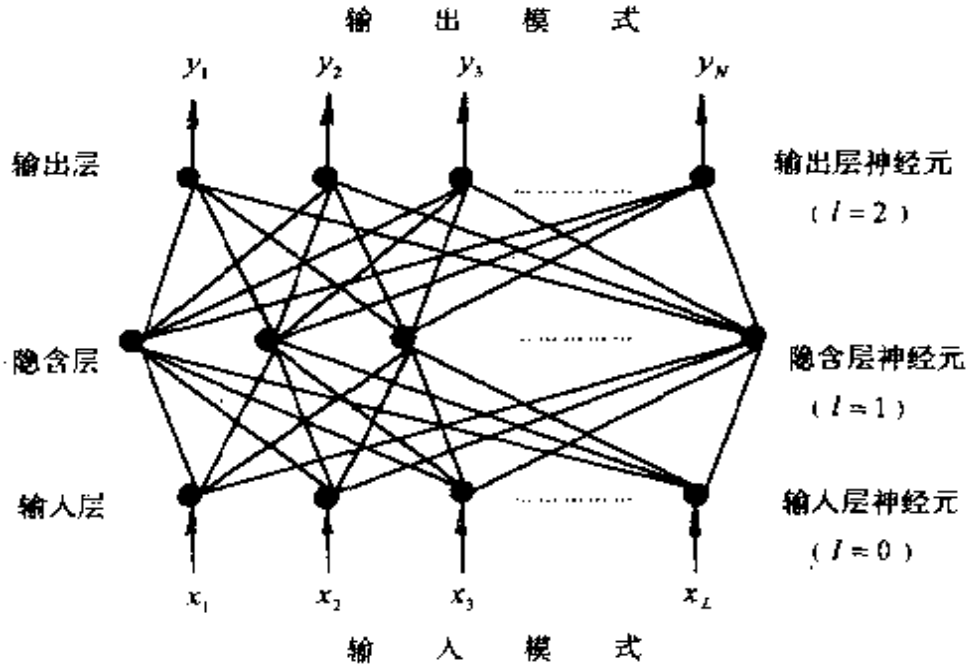


图 4.6 前向多层神经网络(BP 网络)模型

可以证明：在隐含层节点可以根据需要自由设置的情况下，那么用三层前向神经网络可以实现以任意精度逼近任意连续函数.

BP 神经网络中的动力学过程有两类：一类是学习过程，在这

类过程中，神经元之间的连接权将得到调整，使之与环境信息相符合. 连接权的调整方法称为学习算法.

另一类过程是指神经网络的计算过程，在该过程中将实现神经网络的活跃状态的模式变换. 与学习过程相比，计算过程的速度要快得多，因而，计算过程又称为快过程. 与之对应，学习过程通常称为慢过程.

现在，我们来推导前向多层神经网络的学习算法. 设从第 l 层神经元 j 到第 $l-1$ 层神经元 i 的连接权值为 $w_{ji}^{(l)}$，p 为当前学习样本，$o_{pi}^{(l)}$ 为在 p 样本下第 l 层第 i 个神经元的输出，变换函数 $f[\cdot]$ 取为 Sigmoid 函数，即 $f(x)=\dfrac{1}{1+e^{(-x)}}$.

对于第 p 个样本，网络的输出误差 E_p 用下式表示：

$$E_p=\frac{1}{2}\sum_{j=0}^{N-1}(t_{pj}-o_{pj}^{(2)})^2 \tag{4.3.1}$$

上式中 t_{pj} 为输入第 p 个样本时第 j 个神经元的理想输出，$o_{pj}^{(2)}$ 是它的实际输出.

考虑多层神经网络中的 l 层(隐含层或输出层，即 $l=1,2$)，假设第 l 层有 J 个神经元，第 $l-1$ 层有 I 个神经元，具有如下的通用结构：

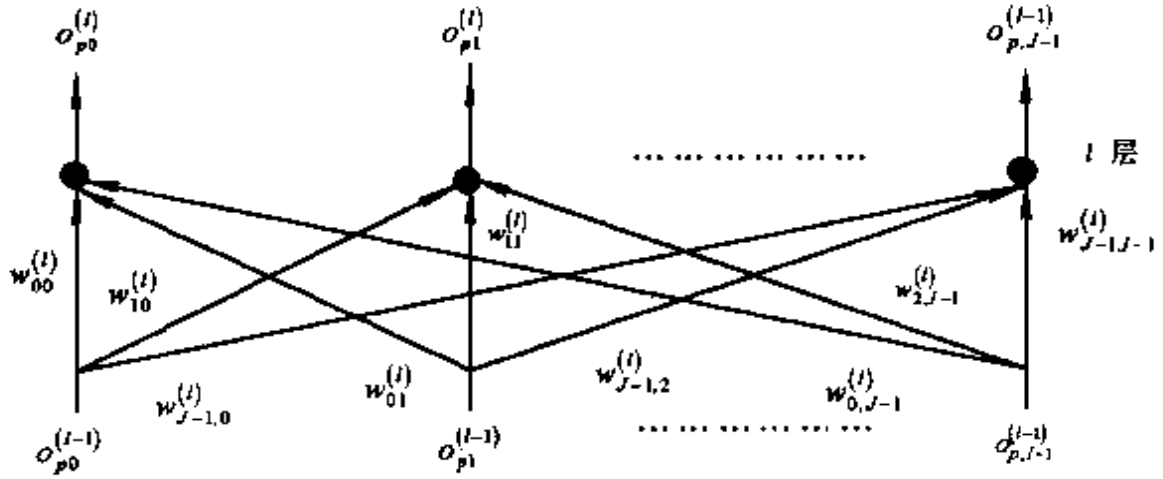


图 4.7 BP 神经网络的通用层结构

为了使系统的实际输出与理想输出相接近，即使 E_p 下降，根据梯度算法，我们可以对 l 层按下式进行调整：

$$\Delta_p w_{ji}^{(l)} \propto -\frac{\partial E_p}{\partial w_{ji}^{(l)}} \quad , \quad l = 1,2 \tag{4.3.2}$$

对于非输入层的神经元具有下面的操作特性：

$$\mathrm{net}_{pj}^{(l)} = \sum_{i=0}^{I-1} w_{ji}^{(l)} o_{pi}^{(l-1)} - \theta_j^{(l)} \tag{4.3.3}$$

$$o_{pj}^{(l)} = f_j(\mathrm{net}_{pj}^{(l)}) \tag{4.3.4}$$

在式(4.3.3)中，如果将 $-\theta_j^{(l)}$ 看作是第 $l-1$ 层的一个虚拟神经元的输出，即设

$$o_{pI}^{(l-1)} = 1 \quad , \quad w_{jI}^{(l)} = -\theta_j^{(l)}$$

则式(4.3.3)可改写为

$$\mathrm{net}_{pj}^{(l)} = \sum_{i=0}^{I} w_{ji}^{(l)} o_{pi}^{(l-1)} \tag{4.3.5}$$

又

$$\frac{\partial E_p}{\partial w_{ji}^{(l)}} = \frac{\partial E_p}{\partial \mathrm{net}_{pj}^{(l)}} \frac{\partial \mathrm{net}_{pj}^{(l)}}{\partial w_{ji}^{(l)}} \tag{4.3.6}$$

由式(4.3.5)可以得到

$$\frac{\partial \mathrm{net}_{pj}^{(l)}}{\partial w_{ji}^{(l)}} = \frac{\partial}{\partial w_{ji}^{(l)}} \sum_{k=0}^{I} w_{jk}^{(l)} o_{pk}^{(l-1)} = o_{pi}^{(l-1)} \tag{4.3.7}$$

定义：

$$\delta_{pj}^{(l)} \equiv -\frac{\partial E_p}{\partial \mathrm{net}_{pj}^{(l)}}$$

综合式(4.3.2)、(4.3.3)、(4.3.5)和(4.3.7)得出

$$\Delta_p w_{ji}^{(l)} = \eta \delta_{pj}^{(l)} o_{pi}^{(l-1)} \tag{4.3.8}$$

$$i = 0,1,2,\cdots,I, \quad j = 0,1,2,\cdots,J-1, \quad l = 1,2$$

可见，为求出调整值，必须先求出 $\delta_{pj}^{(l)}$.

$$\delta_{pj}^{(l)} = -\frac{\partial E_p}{\partial \mathrm{net}_{pj}^{(l)}} = -\frac{\partial E_p}{\partial o_{pj}^{(l)}} \frac{\partial o_{pj}^{(l)}}{\partial \mathrm{net}_{pj}^{(l)}} \tag{4.3.9}$$

由式(4.3.4)得到

$$\frac{\partial o_{pj}^{(l)}}{\partial \operatorname{net}_{pj}^{(l)}} = f_j'(\operatorname{net}_{pj}^{(l)})$$

现在，我们分两种情况来讨论：

1）如果所讨论的神经元为输出神经元，则由式(4.3.1)可得

$$\frac{\partial E_p}{\partial o_{pj}^{(l)}} = -(t_{pj} - o_{pj}^{(2)})$$

代入式(4.3.9)得到

$$\delta_{pj}^{(l)} = (t_{pj} - o_{pj}^{(2)}) f_j'(\operatorname{net}_{pj}^{(l)}) \tag{4.3.10}$$

$$l = 2, \quad j = 0,1,2,\cdots,N-1$$

2）如果所讨论的神经元为隐层神经元，则有

$$\begin{aligned}\frac{\partial E_p}{\partial o_{pj}^{(l)}} &= \sum_{k=0}^{N-1} \frac{\partial E_p}{\partial \operatorname{net}_{pk}^{(l+1)}} \frac{\partial \operatorname{net}_{pk}^{(l+1)}}{\partial o_{pj}^{(l)}} \\ &= \sum_{k=0}^{N-1} \frac{\partial E_p}{\partial \operatorname{net}_{pk}^{(l+1)}} \frac{\partial}{\partial o_{pj}^{(l)}} \sum_{i=0}^{M-1} w_{ki}^{(l+1)} o_{pi}^{(l)} \\ &= \sum_{k=0}^{N-1} \frac{\partial E_p w_{kj}^{(l+1)}}{\partial \operatorname{net}_{pk}^{(l+1)}} \\ &= -\sum_{k=0}^{N-1} \delta_{pk}^{(l+1)} w_{kj}^{(l+1)}\end{aligned}$$

将此结果代入式(4.3.9)得到

$$\delta_{pj}^{(l)} = f_j'(\operatorname{net}_{pj}^{(l)}) \sum_{k=0}^{N-1} \delta_{pk}^{(l+1)} w_{kj}^{(l+1)} \tag{4.3.11}$$

$$l = 1, \quad j = 0,1,2,\cdots,M-1$$

从上式可以看出，为求出隐含层的输出误差系数 $\delta_{pi}^{(1)}$，必须用到输出层的 $\delta_{pj}^{(2)}$，所以这个过程也称为误差反向传播过程(EBP，error back-propagation)。

现在来讨论 $\delta_{pj}^{(l)}$ 项中的 $f_j'(\operatorname{net}_{pj}^{(l)})$，由于 $f[\cdot]$ 采用 Sigmoid 函数，即

$$f_j(\operatorname{net}_{pj}^{(l)}) = \frac{1}{1 + e^{-\operatorname{net}_{pj}^{(l)}}} = o_{pj}^{(l)}$$

由此，我们可以得到

$$f_j(\mathrm{net}_{pj}^{(l)}) = o_{pj}^{(l)}(1 - o_{pj}^{(l)}) \tag{4.3.12}$$

将式(4.3.12)代入式(4.3.10)和式(4.3.11)得到

$$\delta_{pi}^{(l)} = (t_{pj} - o_{pi}^{(2)}) o_{pi}^{(l)}(1 - o_{pi}^{(l)}) \tag{4.3.13}$$

$$l = 2, \quad j = 0,1,2,\cdots,N-1$$

和

$$\delta_{pi}^{(l)} = \left(\sum_{k=0}^{N-1} \delta_{pk}^{(l+1)} w_{kj}^{(l+1)}\right) o_{pi}^{(l)}(1 - o_{pi}^{(l)}) \tag{4.3.14}$$

$$l = 1, \quad i = 0,1,2,\cdots,M-1$$

将式(4.3.13)和式(4.3.14)代入式(4.3.8)得到

当 $l=2$(输出层)时

$$\Delta_p w_{ij}^{(2)} = \eta(t_{pi} - o_{pi}^{(2)}) o_{pi}^{(2)}(1 - o_{pi}^{(2)}) o_{pj}^{(1)} \tag{4.3.15}$$

$$i = 0,1,2,\cdots,N-1, \quad j = 0,1,2,\cdots,M$$

当 $l=1$(隐含层)时

$$\Delta_p w_{ij}^{(1)} = \eta\left(\sum_{k=0}^{N-1} \delta_{pk}^{(2)} w_{ki}^{(2)}\right) o_{pi}^{(1)}(1 - o_{pi}^{(1)}) o_{pj}^{(0)} \tag{4.3.16}$$

$$o_{pj}^{(0)} = x_{pj}, \quad i = 0,1,2,\cdots,M-1, \quad j = 0,1,2,\cdots,L$$

至此为止，推导完了 BP 学习算法，式(4.3.15)与式(4.3.16)为推导的最后结果.

现在我们再来讨论 BP 学习算法中的几个值得注意的问题：

1) 神经网络输入层、输出层的神经元个数可以根据研究对象的输入、输出信息来确定，如何合适选取隐含层神经元的数目无规律可循，然而隐含神经元的数目是否合适对整个网络能否正常工作具有重要意义. 隐含神经元数目如果太少，则网络可能根本无法训练；如果隐含神经元数目刚刚够，则网络可以训练，但鲁棒性不好，抗噪音能力差，无法辨识以前未见过的模式；如果隐含神经元过大，则除了需要很多训练样本外，还可能会建立一个“老祖母”网络，具有了所有模式，而无法认识新的内容，同时训练时也必然耗费更多的时间，并占有更多的内存.

一般情况下可按下式给出：

$$n_H = \sqrt{n_I + n_O} + l \tag{4.3.17}$$

式中

n_H 为隐含层神经元数目；

n_I 为输入层神经元数目；

n_O 为输出层神经元数目；

l 为 1—10 之间的整数.

2) 学习算法中的 η 表示学习速率，或称为步幅，η 较大时，权值的修改量就较大，学习速率比较快，但有时会导致振荡；η 值较小时，学习速率慢，然而学习过程平稳. 这样，在实际的学习过程中，可以将 η 值取为一个与学习过程有关的变量，并且在学习刚开始时 η 值相对大，然后随着学习的深入，η 值逐步减小.

η 值的具体选取方案已有很多种，但迄今为止没有一种是令人信服的. 在一些简单的问题中，η 可取为一个常数，满足 $0<\eta<1$，如 η 取 0.5.

3) 在权值的修改公式中，往往还加入一个惯性项（有时称为动量项）$\alpha\Delta w_{ji}^{(l)}(n-1)$，即

$$\Delta w_{ji}^{(l)}(n) = \eta\delta_{pj}^{(l)}(n)o_{pi}^{(l-1)}(n) + \alpha\Delta w_{ji}^{(l)}(n-1) \tag{4.3.18}$$

式中 $\Delta w_{ji}^{(l)}(n)$ 表示第 l 层第 j 个神经元与上一层第 i 个神经元之间的连接权的当前修改值，$\Delta w_{ji}^{(l)}(n-1)$ 表示上一个学习周期对同一个学习样本的权值修改值.

惯性项校正系数 α 应与 η 协调选取，通常较大的 α 可以改善网络的收敛速度，但对提高网络的收敛精度没有积极的作用.

有文献指出：当 α 依下式取值时，可同时获得较好的收敛速度和精度：

$$\alpha_{pj}^{(l)}(n) = \frac{\delta_{pj}^{(l)2}(n)}{\sum_{m=1}^{n-1}\delta_{pj}^{(l)2}(m)} \tag{4.3.19}$$

此时权值修改公式为

$$\Delta w_{ji}^{(l)}(n) = \eta \delta_{pj}^{(l)}(n) o_{pi}^{(l-1)}(n) + \alpha_{pj}^{(l)}(n-1) \Delta w_{ji}^{(l)}(n-1) \tag{4.3.20}$$

对于简单的情况，α 可取一个常数，如 $\alpha=0.5$.

4）由于单个神经元的转换函数大都是采用 Sigmoid 函数而不是阶跃函数，因而输出层各神经元的实际输出值只能趋近于 1 或者 0，而不可能达到 1 或者 0. 例如：当 net＝1 时，$f(\text{net})=0.73106<1$，因而，在设置各训练样本的理想输出分量时，t_{pj}有时可取为接近 1，0 的数，如 0.9，0.1 等而不直接取为 1，0.

5）在学习开始时，必须给各个连接权赋初值. 我们可以对每个连接权赋一个随机值，但不能使所有的连接权初值都相等.

在实际的网络训练过程中，通常的处理方法是给每一个连接权赋以－1 至 1 之间的一个随机数.

6）BP 算法学习的目的是为了寻找连接权 $w_{ji}^{(l)}$（$l=1$ 时，$i=0,1,2,\cdots,L$，$j=0,1,2,\cdots,M-1$；$l=2$ 时，$i=0,1,2,\cdots,M$，$j=0,1,2,\cdots,N-1$. $L,M\cdot N$ 为输入层、隐含层、输出层的神经元个数）使得 $E=\sum_{p}\sum_{j=0}^{N-1}(t_{pj}-o_{pj}^{(2)})^2$ 趋于全局最小，然而在实际操作时，我们获得的连接权常常不能使 E 趋于全局最小，而只能使之趋于一个相对大一点的 E 值，我们称为局部最优. 如何避免在学习过程中陷入局部最小是 BP 算法的一大难题.

我们知道，当出现局部最优的情况时，表现出来的特征是：各权值收敛到某一稳定值，而误差值却不是最小. 因此，我们可以按下式判定：

$$\begin{cases} |o_{pj}^{(2)}(n+1)-o_{pj}^{(2)}(n)|<\xi \\ |o_{pj}^{(2)}(n)-t_{pj}|>\beta \end{cases}, \quad j=0,1,\cdots,N-1 \tag{4.3.21}$$

式中 $\xi \ll 1$，β 为一小数，通常 $0<\beta<0.2$.

如符合上式，则认为此时 BP 网络陷入局部极小点.

系统陷入局部最小点后的处理方法有很多种，最简单的方法是从头重新做起，此外我们还可以采用模拟退火法和遗传算法(Genetic Algorithms，GA)来消除局部最小问题，在此不作介绍，感兴趣的读者可参阅有关资料.

综合上面的讨论，我们可以按照以下步骤来设计具体的学习过程：

1) 网络结构及学习参数的确定：输入输入层、隐含层、输出层的神经元数目，步长 η 以及惯性项校正系数 α、权值收敛因子 ξ 及误差收敛因子 β.

2) 网络状态初始化：用较小的(绝对值为 1 以内)随机数对网络权值、阈值置初值.

3) 提供学习样本：输入向量 $x_p(p=1,2,\cdots,P)$和目标向量 t_p $(p=1,2,\cdots,P)$.

4) 学习开始：对每一个样本进行如下操作：

a) 计算网络隐含层及输出层各神经元的输出

$$o_{pj}^{(l)} = f_j(\mathrm{net}_{pj}^{(l)}) = f_j\Big(\sum_i w_{ji}^{(l)} o_i^{(l-1)} - \theta_j^{(l)}\Big)$$

b) 计算训练误差

$$\delta_{pj}^{(2)} = o_{pj}^{(2)}(1 - o_{pj}^{(2)})(t_{pj} - o_{pj}^{(2)}) \qquad \text{(输出层)}$$

$$\delta_{pj}^{(1)} = o_{pj}^{(1)}(1 - o_{pj}^{(1)}) \sum_k \delta_{pk}^{(2)} w_{kj}^{(2)} \qquad \text{(隐含层)}$$

c) 修改权值和阈值

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \eta \delta_{pj}^{(l)} o_{pi}^{(l-1)} + \alpha(w_{ji}^{(l)}(n) - w_{ji}^{(l)}(n-1))$$

5) 是否满足 $|o_{pj}^{(l)}(n+1) - o_{pj}^{(l)}(n)| < \xi$? 满足执行第 6)步，否则返回 4).

6) 是否满足 $|o_{pj}^{(l)}(n) - t_{pj}(n)| < \beta$? 若是则执行第 7)步，否则返回第 2)步.

7) 停止.

其程序流程图如下：

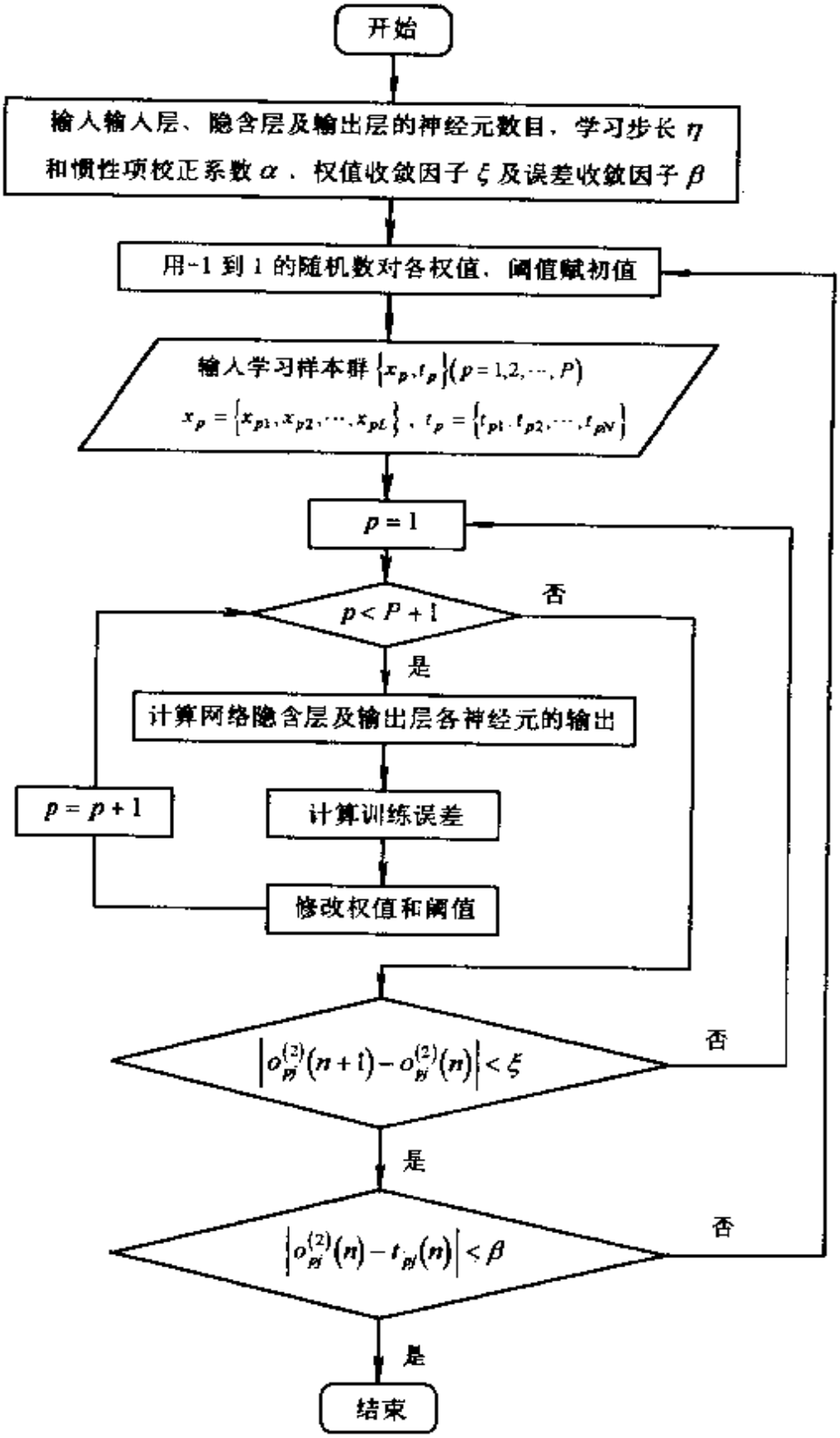

开始
输入输入层、隐含层及输出层的神经元数目，学习步长 η
和惯性项校正系数 α ，权值收敛因子 ξ 及误差收敛因子 β
用-1 到 1 的随机数对各权值、阈值赋初值
输入学习样本群 $\{x_p, t_p\}(p=1,2,\cdots,P)$
$x_p=\{x_{p1},x_{p2},\cdots,x_{pL}\}$ ， $t_p=\{t_{p1},t_{p2},\cdots,t_{pN}\}$
$p=1$
$p<P+1$
否
是
计算网络隐含层及输出层各神经元的输出
$p=p+1$
计算训练误差
修改权值和阈值
$\left|o_{pj}^{(2)}(n+1)-o_{pj}^{(2)}(n)\right|<\xi$
否
是
$\left|o_{pj}^{(2)}(n)-t_{pj}(n)\right|<\beta$
否
是
结束

图 4.8　BP 算法流程图

需要特别指出的是：由于在理论上三层 BP 神经网络在隐含层神经元数目可以任意设定的情况下，可以以任意精度逼近任意连续函数，因此本书在具体设计神经网络故障诊断专家系统时都采用三层神经网络，故在本节中介绍 BP 算法时也只介绍三层（即单隐含层）时的情况. 但是，该算法也同样适用于多隐含层 BP 神经网络的情况.

4.4 神经网络诊断专家系统知识库的组建

神经网络诊断专家系统知识库的组建包括两个过程：知识的获取和知识的存储. 知识的获取表现为训练样本的获得与选择，训练样本来源于同类型诊断对象在正常运转时和带故障运行时的各种特征参数. 训练样本的选择应遵循两条原则：相容性，即样本之间不能有冲突；遍历性，即样本具有代表性.

与传统专家系统以规则等形式显式地将故障诊断知识存储在知识库中相比，神经网络专家系统采取完全不同的知识存储方法：它将诊断知识隐式地分散存储在神经网络的各项连接权值和阈值中，知识存储方法的不同是这两种专家系统的一大区别. 从下面的三条定理可以看出：这种独特的知识存储方法可以有效地解决传统专家系统在运行过程中出现的知识容量与运行速度的矛盾.

定理 1 由 N 个神经元组成的神经网络的信息表达能力上限为

$$C \leqslant (N-1)^2 N$$

定理 2 由 N 个神经元组成的神经网络的信息表达能力下限为

$$C \geqslant \frac{N^3}{24}$$

定理 3 任意由 N 个输入向量构成的任何布尔函数的前向神经网络实现所需权系数数目为

$$W \geqslant \frac{N}{1+\log_2^N}$$

假设某传统故障诊断专家系统有 10000 条规则，现在用神经

网络来重新构造这个专家系统，由定理 1 知道要完全表达这些信息至少需 23 个神经元；由定理 2 得知用 63 个神经元就能确保储存这些规则；由定理 3 得知如果用 BP 神经网络则至少需要 700 个连接权系数. 如果现在采用的 BP 网络输入神经元数目为 25 个，输出神经元数目为 10 个，隐含神经元数目取为 20 个，则该网络完全可以实现这种 10000 条规则的储存.

下面根据诊断对象的故障知识难易复杂程度来分别讨论如何具体建立一个神经网络故障诊断专家系统的知识库.

4.4.1 简单诊断对象的神经网络专家系统知识库的组建

对于简单的诊断对象，故障知识层次结构简单，故障征兆和故障原因都不多也即输入神经元和输出神经元数目比较少，因而可直接用一个 BP 神经网络来组建这个故障诊断专家系统. 具体的组建步骤为

1）根据征兆、故障及训练样本数目确定神经网络的结构.

2）在众多的样本中选择训练样本.

3）对神经网络进行训练获得连接权值，形成知识库.

下面通过一个例子来详细说明如何组建一个神经网络专家系统的知识库.

如图 4.9 所示为一个故障模拟试验台. 振动信号经过垂直、水平两只涡流传感器送至 ZXP-4 型表和 BK2034 频谱分析仪，试验台转速可调.

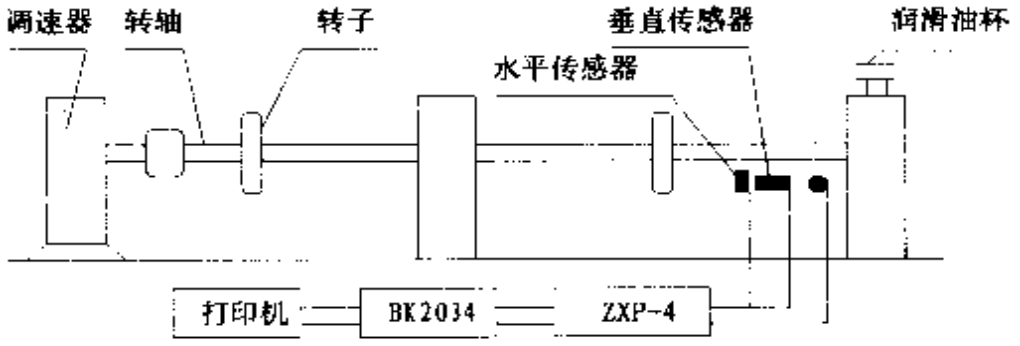


图 4.9 转子试验台结构简图

在该转子试验台上，可以通过调整中间轴承座底部垫片厚度、不平衡重块大小和润滑油粘度、轴承长径比等来实现模拟不平衡、油膜振荡和不对中三种故障.

分别取振动信号频谱图中的 0.4—0.5x，1x，2x，3x，>3x 分量作为特征值. 网络学习和测试时对数据进行归一化处理，即：令

$$x_{\max} = \max(x_i), \qquad \overline{x}_i = \frac{x_i}{x_{\max}}$$

这样，网络所有输入都在[0,1]内.

每种典型单故障可分别选取 5 组频谱值，构成相应三种故障的 15 组学习样本，如表 4.1 所示.

表 4.1　神经网络学习样本

序号	模拟故障	样本输入					样本输出		
		0.4x—0.5x	1x	2x	3x	>3x	F_1	F_2	F_3
1	大油膜振荡	0.88	0.22	0.02	0.04	0.06	0.9	0.1	0.1
2	大油膜振荡	0.90	0.20	0.05	0.02	0.02	0.9	0.1	0.1
3	大油膜振荡	0.92	0.21	0.03	0.01	0.02	0.9	0.1	0.1
4	大油膜振荡	0.85	0.25	0.06	0.02	0.01	0.9	0.1	0.1
5	大油膜振荡	0.82	0.28	0.05	0.04	0.03	0.9	0.1	0.1
6	油膜振荡	0.04	0.98	0.10	0.02	0.02	0.1	0.9	0.1
7	油膜振荡	0.02	1.00	0.08	0.03	0.01	0.1	0.9	0.1
8	油膜振荡	0.05	0.90	0.11	0.05	0.02	0.1	0.9	0.1
9	油膜振荡	0.03	0.96	0.12	0.04	0.03	0.1	0.9	0.1
10	油膜振荡	0.02	0.91	0.08	0.01	0.02	0.1	0.9	0.1
11	大不对中	0.02	0.41	0.43	0.34	0.15	0.1	0.1	0.9
12	大不对中	0.01	0.52	0.40	0.32	0.10	0.1	0.1	0.9
13	大不对中	0.01	0.40	0.47	0.35	0.18	0.1	0.1	0.9
14	大不对中	0.02	0.45	0.42	0.28	0.29	0.1	0.1	0.9
15	大不对中	0.01	0.48	0.48	0.36	0.20	0.1	0.1	0.9

在这个实例中，我们有 5 个征兆数和 3 个故障原因，因而，神

经网络也就有 5 个输入神经元和 3 个输出神经元. 隐含神经元数目的确定一方面要遵循前面提到的原则，另一方面要考虑使网络收敛速度快并对测试样本的识别效果较好. 在这里，可以将隐含层神经元数目取为 5 个，从而得到图 4.10 所示的神经网络结构.

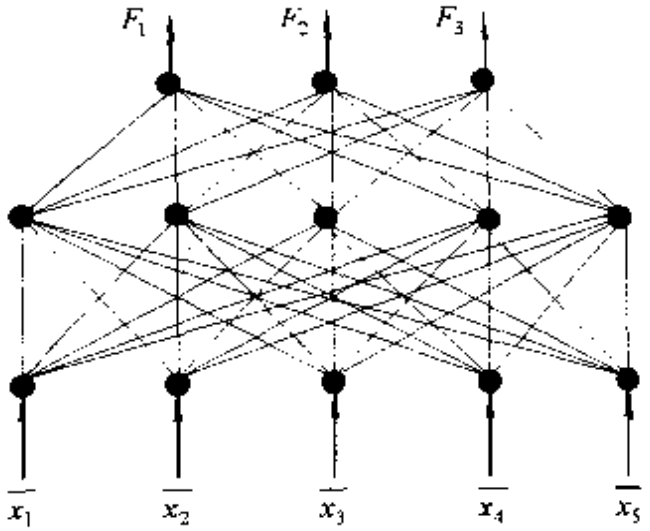


图 4.10 BP 网络结构图

建立神经网络专家系统的第一步就是组建知识库，也就是确定神经网络的各个隐含神经元和输出神经元的权值和阈值以及输入神经元和隐含神经元的权值和阈值. 这实际上就是前面提到的神经网络的学习过程. 在附录中，给出了这个故障诊断神经网络专家系统的全部 C 语言程序，“YJG. C”、“YJG. H”及“BPXL. C”. 读者可以在自己具体编程时参考.

需要特别指出的是：由于神经网络在学习时，首先就必须给每个权值(包括阈值)取一个小的随机数，因此会导致不同的学习过程得出不同的权值和阈值，它们的差异有时还比较大. 表 4.2 和表 4.3 是这个例子的两次不同训练结果. 对于相同的样本进行训练可以获得不同的连接权值和阈值，这一现象表明：神经网络故障诊断专家系统的知识库具有不唯一性. 图 4.11 给出了训练时的一条步长-误差曲线图. 当然，这只是一个特例，通常的 BP 网络的学习步长多达上万步.

表 4.2　权值和阈值数据(一)

神经元	权值数据					阈值
第一隐含神经元	2.481	0.076	2.176	1.840	1.260	−0.292
第二隐含神经元	−0.761	2.443	−0.834	−0.267	0.396	−0.784
第三隐含神经元	2.687	0.542	−2.059	−1.357	−0.781	−0.029
第四隐含神经元	1.277	−0.990	0.991	0.661	0.568	0.336
第五隐含神经元	1.182	−3.133	2.872	2.129	1.195	0.876
第一输出神经元	−2.974	−2.146	2.595	0.592	1.507	−0.861
第二输出神经元	0.100	2.415	0.904	−1.461	−4.319	0.666
第三输出神经元	3.159	−1.282	−3.351	0.052	2.612	−1.196

表 4.3　权值和阈值数据(二)

神经元	权值数据					阈值
第一隐含神经元	−0.999	−2.839	3.699	3.309	2.208	0.274
第二隐含神经元	−0.961	2.273	−0.774	−0.810	0.096	−0.827
第三隐含神经元	2.262	−0.554	−0.638	−0.274	0.328	−0.326
第四隐含神经元	1.896	−0.500	−0.552	0.174	0.160	−0.357
第五隐含神经元	−2.218	2.279	1.069	0.354	0.623	−0.494
第一输出神经元	−1.001	−1.300	2.068	1.685	−2.846	0.216
第二输出神经元	4.345	2.426	−1.016	−0.868	2.104	−0.378
第三输出神经元	5.083	−1.444	−1.447	−0.978	0.581	−1.421

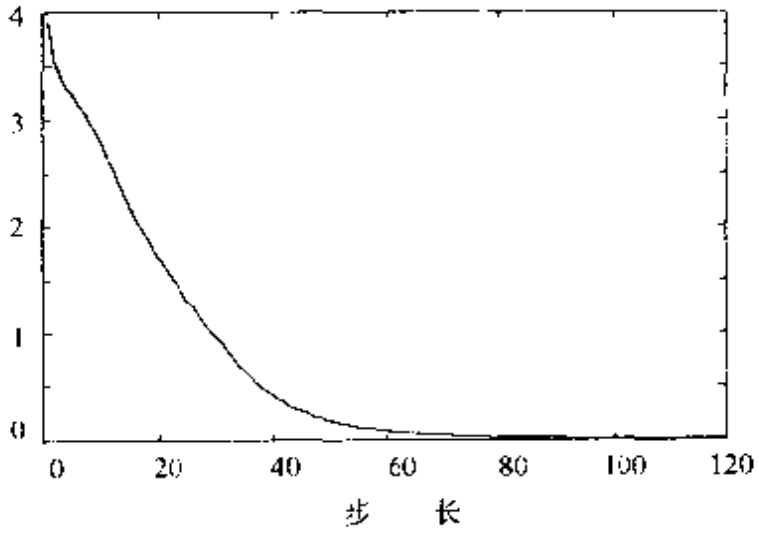


图 4.11　步长-误差曲线图

将所获得的各个权值和阈值数据存入一数据文件，该数据文件即是神经网络诊断专家系统的知识库内容.

4.4.2 复杂诊断对象的神经网络专家系统知识库的组建

对于复杂的诊断对象，如在“传统故障诊断专家系统的组建”一章中所举的汽车故障诊断一例，系统共有 8 个输出、12 个输入. 至于更复杂的系统，输入输出神经元比这还要多得多. 在这种情况下，如果还和以前一样，试图通过一个简单的三层神经网络来构造一个故障诊断专家系统，将出现两大弊端：

1）由于连接权数目很多（如上面说的汽车故障诊断一例中，如取隐含层神经元数目为 12 个，那么，连接权数目（包括阈值）将达到 260），学习样本的组合也将很巨大，这必然影响神经网络的学习效率.

2）更为糟糕的是，这个神经网络一点也不能反映诊断对象的结构知识. 对于一个复杂的诊断对象，往往可以划分成几大块，如在医疗诊断中，人的内脏可以划分成肾、肝、心、脾、肺等组成部分. 如不加考虑地将诊断系统建立在一个简单的 BP 网络上，必将增加诊断的难度且降低诊断的精度并为以后解释程序的设计带来很大的困难.

由此可见，我们有必要对这种简单的神经网络结构进行改进，建立一种含有多个子网络的复杂神经网络故障诊断专家系统. 具体步骤为

1）分析诊断对象故障知识结构以确定神经网络结构模型.

2）与前面介绍的简单诊断系统相似，依次确定各个子网络的训练样本，并分别进行训练，获得各自的连接权值和阈值.

3）存储连接权，形成知识库.

现在仍以上一章的汽车故障诊断专家系统为例来讨论如何具体组建一个复杂神经网络故障诊断专家系统的知识库.

观察图 3.2 可以发现：该系统最下面是三个子系统：电气系统、供油系统和点火系统，上面的 8 个故障可以分成相互比较独立

的三个块：由电气系统故障导致的故障包括电池没电和电池接线故障(称为故障块一)；由供油系统故障导致的故障包括油箱没油、汽油油路故障、汽油泵故障和气化器故障(称为故障块二)；由点火系统故障导致的故障包括点火线路故障和火花塞故障(称为故障块三)．依照这种讨论，我们可以将结构图 3.2 简化成图 4.12.

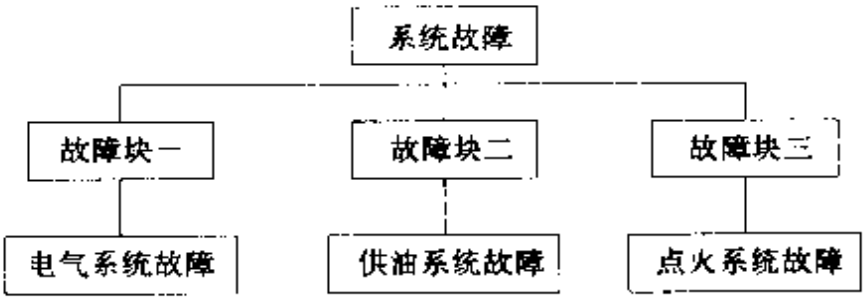


图 4.12　汽车故障结构简图

为方便下面的讨论，现在首先定义输入输出符号如下：

x_1：起动机能否转动，　0 能　1 不能

x_2：电池接线是否正常，　0 正常　1 不正常

x_3：油箱中油量是否正常，　0 正常　1 不正常

x_4：汽油泵入口处有没有汽油，　0 有汽油　1 没汽油

x_5：汽油泵出口处有没有汽油，　0 有汽油　1 没汽油

x_6：火花塞是否正常，　0 正常　1 不正常

x_7：仪表盘是否亮，　0 亮　1 不亮

x_8：照明灯是否亮，　0 亮　1 不亮

x_9：点火火花是否正常，　0 正常　1 不正常

x_{10}：是否来油且畅通，　0 是　1 不是

x_{11}：怠速运转是否熄灭，　0 不熄　1 熄灭

x_{12}：发动机运转是否平稳，　0 平稳　1 不平稳

y_1：电气系统是否正常，　0 正常　1 不正常

y_2：点火系统是否正常，　0 正常　1 不正常

y_3：供油系统是否正常，　0 正常　1 不正常

y_4：电池是否有电，　0 有　1 没有

y_5：电池接线是否正常，　　0 正常　　1 不正常

y_6：油箱是否没有油，　　0 有　　1 没有

y_7：是否汽油油路问题，　　0 不是　　1 是

y_8：汽油泵是否正常，　　0 正常　　1 不正常

y_9：气化器是否正常，　　0 正常　　1 不正常

y_{10}：火花塞是否正常，　　0 正常　　1 不正常

y_{11}：点火线路是否正常，　　0 正常　　1 不正常

根据图 4.12 的这种知识结构，我们可以依照如下步骤设计汽车故障诊断神经网络专家系统的结构：

(1) 建立电气系统子网络

电气系统子网络只有两个输入神经元，一个输出神经元，隐含神经元可以取为 1 个. 训练样本如表 4.4 所示，神经网络结构如图 4.13 所示.

表 4.4　电气系统子网络训练样本

样本号	x_7	x_8	y_1
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

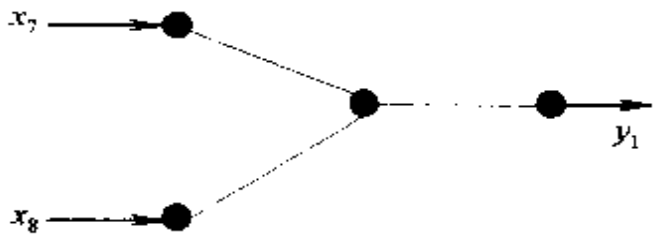


图 4.13　电气系统子网络

(2) 建立供油系统子网络

供油系统子网络有四个输入神经元，一个输出神经元，隐含神经元可以取为 3 个. 部分训练样本如表 4.5 所示，神经网络结构如图 4.14 所示.

表 4.5　供油系统子网络训练样本

样本号	x_1	x_7	x_9	x_{10}	y_2
1	0	0	0	1	1
2	1	0	1	0	0
3	0	1	1	0	0
4	1	1	0	0	0

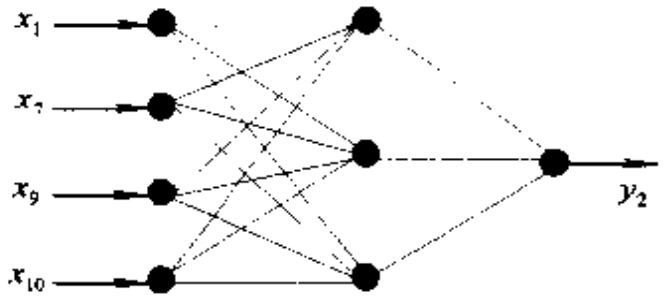


图 4.14　供油系统子网络

(3) 建立点火系统子网络

点火系统子网络与供油系统子网络相似，有四个输入神经元，一个输出神经元，隐含神经元可以取为 3 个. 部分训练样本如表 4.6 所示，神经网络结构如图 4.15 所示.

表 4.6　供油系统子网络训练样本

样本号	x_1	x_3	x_7	x_9	y_3
1	0	0	0	1	1
2	1	0	1	0	0
3	0	1	1	0	0
4	1	1	0	0	0

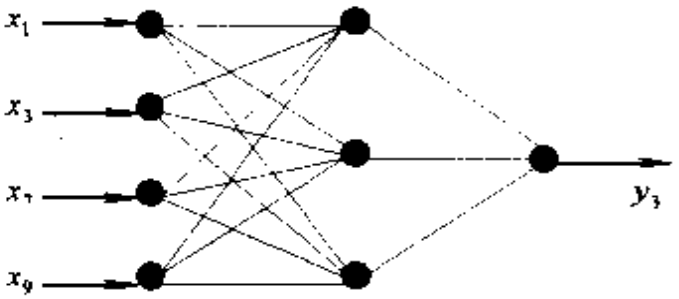


图 4.15　点火系统子网络

(4) 建立故障块—子网络

故障块一子网络有三个输入神经元，两个输出神经元，隐含神经元可以取为 2 个. 部分训练样本如表 4.7 所示，神经网络结构如图 4.16 所示.

表 4.7 故障块一子网络训练样本

样本号	x_3	x_2	y_1	y_4	y_5
1	1	0	1	1	0
2	0	1	1	0	1
3	1	1	1	0	1
4	0	0	0	0	0

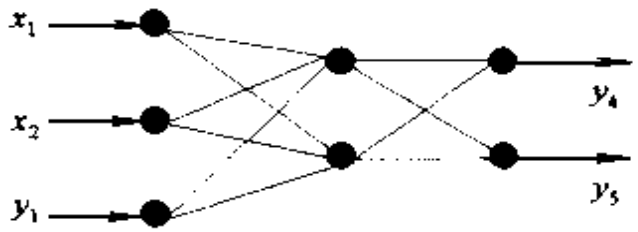


图 4.16 故障块一子网络

(5) 建立故障块二子网络

故障块二子网络是这个诊断专家系统中最大的一个子网络，它具有 5 个输入神经元，4 个输出神经元，隐含神经元可以取为 3 个. 部分训练样本如表 4.8 所示，神经网络结构如图 4.17 所示.

表 4.8 故障块二子网络训练样本

样本号	x_3	x_4	x_5	x_{11}	y_2	y_6	y_7	y_8	y_9
1	0	0	0	0	0	0	0	0	0
2	1	0	0	0	1	1	0	0	0
3	0	1	0	0	1	0	1	0	0
4	0	0	1	0	0	0	0	1	0
5	0	0	0	1	0	0	0	0	1

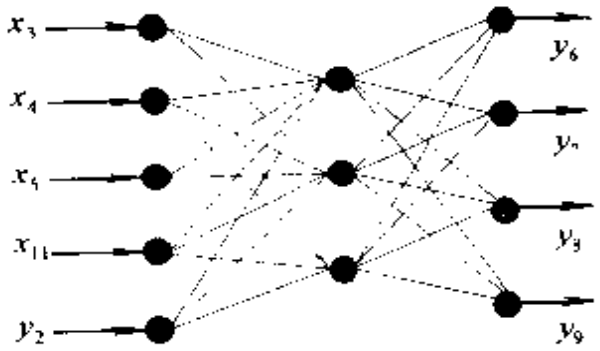


图 4.17　故障块二子网络

(6) 建立故障块三子网络

故障块三子网络与故障块一子网络在结构上相似，它具有 3 个输入神经元，2 个输出神经元，隐含神经元可以取为 2 个. 部分训练样本如表 4.9 所示，神经网络结构如图 4.18 所示.

表 4.9　故障块一子网络训练样本

样本号	x_6	x_{12}	y_3	y_{10}	y_{11}
1	0	1	1	1	0
2	1	0	1	0	1
3	1	1	1	0	1
4	0	0	0	0	0

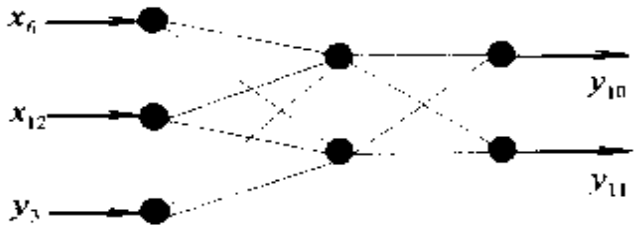


图 4.18　故障块一子网络

(7) 建立合成网络

在前面的(4)、(5)、(6)三步中，我们将神经网络的 8 个输出分别放在 3 个子网络中，这显然还不是我们所需要的. 我们现在再建

立一个神经网络，以这8个输出作为输入，注意到故障块一、故障块二、故障块三3个子网络的输出，发现每一个子网络的输出元中最多输出一个1.因此，为减少神经网络输出神经元的数目，我们可以如下设计合成神经网络：输入神经元为8个，输出神经元为3个，隐含神经元设计为5个.输出神经元输出 y_{12}，y_{13}，y_{14}的含义如下：

y_{12} 0 正常；1. 电池没电；2. 电池接线故障.

y_{13} 0 正常；1. 油箱没油；2. 汽油油路故障；3. 汽油泵故障；4. 气化器故障.

y_{14} 0 正常；1. 点火线路故障；2. 火花塞故障.

部分训练样本如表4.10所示，神经网络的结构如图4.19所示.

表4.10 合成神经网络训练样本表

样本号	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}
1	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	1	0	0
3	0	1	0	0	0	0	0	0	2	0	0
4	0	0	1	0	0	0	0	0	0	1	0
5	0	0	0	1	0	0	0	0	0	2	0
6	0	0	0	0	1	0	0	0	0	3	0
7	0	0	0	0	0	1	0	0	0	4	0
8	0	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	1	0	0	2

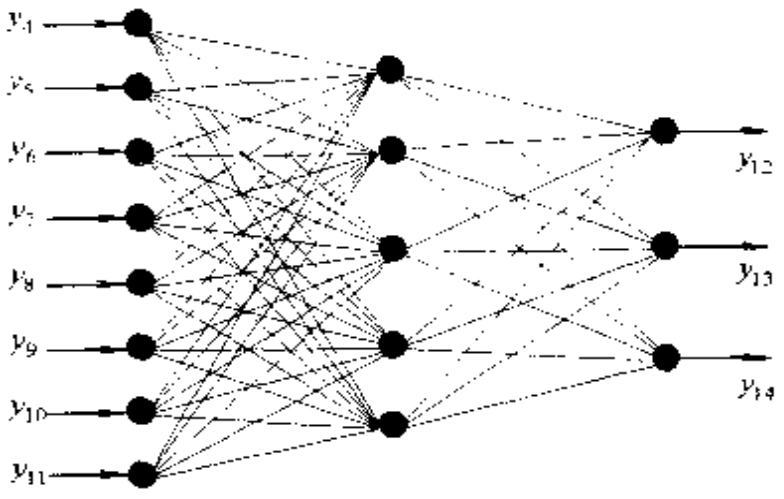


图4.19 合成神经网络结构图

到这里，我们已经设计完了诊断神经网络的结构.现在统计一下这7个神经网络的连接权(包括阈值)数目是168，比上面所提到的260要少得多，因此相关的神经网络运算步数也将减少.

下面要做的工作是对各个网络进行训练.如同简单神经网络故障诊断专家系统，将表4.4到表4.10的训练样本(根据需要可增加训练样本)输入到相应神经网络中进行训练，将得到的权值数据组合在一起存入一数据文件，即为知识库.

4.4.3 深知识在神经网络故障诊断专家系统中的应用例子

在本节前面介绍的知识库内容中，主要是针对浅知识(Shallow Knowlwdge，即领域专家的经验知识).这也是神经网络诊断专家系统发展初期所广泛采用的模式，随着神经网络故障诊断专家系统的发展，深知识(deep knowledge，即诊断对象的结构、性能和功能性知识)也逐渐应用到里面来.在复杂系统的诊断中，通常采用浅知识与深知识相结合的混合知识模型.利用混合知识模型进行诊断，其思路一般都是先利用浅知识推理形成诊断焦点，再用深知识进行确认，然后产生精确的解释.

由于深知识是针对某一种具体的诊断对象的，不同的诊断对象建立的深知识模型会截然不同，因而很难讨论一般的模式，本章在此仅举一个例子作一个大致的说明.

北方交通大学的王飚能等人在利用神经网络对机车电路故障进行诊断时，较好地利用了混合知识模型.

机车电路是一个庞大、复杂的系统，它的故障是机务部门的大敌，据机务部门统计，东风4(DF4)机车电气系统的故障占全部故障的60%以上，成为产生事故的极大隐患，若不能及时修复处理，会给铁路运行的安全直接造成威胁.但是由于电路系统的复杂性，它的几大回路在同一工况下错综耦合，继电器、保护电路和断路器都正常工作时尚较易判断，若出现误动作，开关拒动等情况，会使故障后的系统响应复杂化，从而导致故障识别困难，这时需要较多的启发性知识，并以逻辑判断为主，而且电路故障往往还与非常规

的电器元件(如恒功联合调节器)有关,这就需要涉及机车控制原理较深层的知识.可见,该系统适合采用混合知识模型建立的专家系统来进行诊断.

机车电路故障检测系统是神经网络与专家系统相结合的智能诊断系统,其知识库由电路的组成结构、功能原理用计算机故障模拟建立的深层次知识和由专家提供的实际经验知识即浅层次知识.诊断方法是通过由神经网络(利用浅知识)进行预诊,再由用字典法表示的深知识确诊.

系统的软件组成如图 4.20 所示.其功能原理大致如下:用户通过人机接口(计算机键盘与屏幕)启动管理模块,以确定系统的工作状态(包括自检、诊断及库操作).当首先启动自检模块后,则自动输入典型检测点值并运行该系统,随后与事先设定的诊断结果相对照,两者相符,则说明该系统软、硬件工作是可靠的.当进行诊断时,从所有检测点获得检测信息,建立非正常检测点的故障假

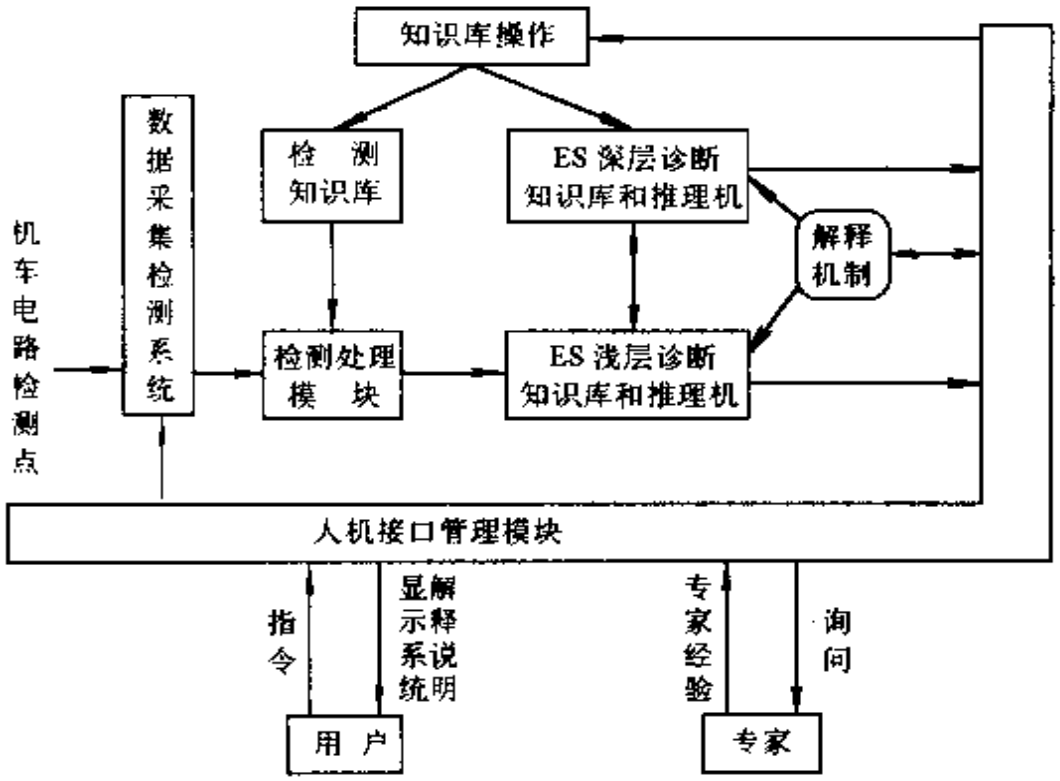


图 4.20 机车故障电路诊断系统软件组成框图

设集.再利用检测知识库判断检测点值是否正常,如有故障,送至动态数据库或相应的人工神经网络(ANN)进行浅层推理,如不成功,就转入深层知识库进行规则逻辑推理.这样,最终得到故障部位、可能原因和维修策略,送至人机接口和解释系统.解释模块给出推理过程和推理根据,解释用户的提问,并以系统询问的方式补充诊断所缺少的信息.知识库操作模块主要用来供专家通过人机接口对知识库进行扩充、修改、删除及自学习等操作.

下面以"无流无压"这个故障为例说明其诊断过程.

这里的"无流无压"是指非"接地"、"过流"、"停机"原因导致的"无流无压"故障现象.

预先,ANN 知识库中存有:

浅层知识 1:一条支路有电,只有当该支路上所含元件均能导通,构成一条通畅的流路,且支路两端有电压降.

浅层知识 2:发电机正常发电,只有当其定子励磁线圈电路通电良好且转子电枢线圈电路通电良好.

浅层知识 3:继电器常开触点通电,只有当线圈电路通电良好且触点接触良好.

事实 1:主发 F 输出电路由主发电枢线圈、主整流 1ZL 牵引电机 1-6D 及导线组成.

事实 2:主发励磁电路由励磁机电枢线圈、副整流 2ZL、LC 主触头、主发 F 定子线圈及导线组成.

事实 3:LC 线圈电路包括 1K→2K→SK→1→2HKf→1J→DJ→LLC 或 1ZL 触头→3、4YJ 或 3ZJ 触点→2ZJ→1-6C 辅助触点→LC 线圈的所有元件及导线.

事实 4:主发 F 是发电机.

事实 5:LC 是继电器.

其次,ES 知识库中存有:

深层知识 1:恒功调节系统的给定环节、比较环节、调整环节、检测环节、执行环节中的故障将可能导致"无流无压"故障.

故障诊断过程开始时，首先进行信号检测，即把所有检测信号送入检测处理模块，在检测知识库中存有该工况下关键信号的正常值，输入值与之相比，判断是否有故障，是则判断是何故障，否则输出显示正常，此时是“无流无压”故障，进入故障诊断模块，首先在 ANN 中搜索推理，由事实 1 和浅层规则 1 推理得出“主发电枢线圈或主整流 1ZL 或牵引电机 1-6D 或导线故障”的结果，若此结果与检测值相符，则推理成功，输出结论；若不相符，继续搜索，与浅层规则 2 相匹配，推理得出“主发励磁电路有故障”的结果，再对此进行判断，对则输出结论，错则进一步推理. 过程与前面相似，由事实 2 和浅层规则 1 得出“励磁机电枢线圈或副整流 2ZL 或 LC 主触头或主发 F 定子线圈或导线有断路故障”的结果，其中若是 LC 主触点故障，由事实 5 和浅层规则 3 得出“LC 线圈电路有故障”的结论，然后，由事实 3 和浅层规则 1 得到“从 1K→2K→SK→1→2HKf→1J→DJ→LLC 或 1ZL 触头→3，4YJ 或 3ZJ 触点→2ZJ→1-6C 辅助触点→LC 线圈的所有元件及导线有电路故障”的结果，然后进行判断，如此推理到测速发电机 CF 的励磁回路. 倘若仍未得出正确结论，那么，就进入 ES 深层知识推理，与深层规则 1 匹配，得出“恒功调节系统故障”的结果，之后返回 ANN 中进行判断，是则输出结论；否则再回到 ES 中，搜索其它规则进行推理. 如此 ANN→ES→ANN 不断交叉往复，依次类推，就推理出“无流无压”故障的各种可能起因，并逐个判断其真实性，最终搜索到故障的真正原因，得到正确的结论.

4.5 神经网络故障诊断专家系统推理机制

在“传统故障诊断专家系统”一章里，我们就讨论过专家系统推理机制有三种：正向推理、反向推理、混合双向推理. 基于神经网络的故障诊断专家系统也能实现这三种推理机制，可以用图 4.21 表示 NN-FD-ES 的通用推理机制.

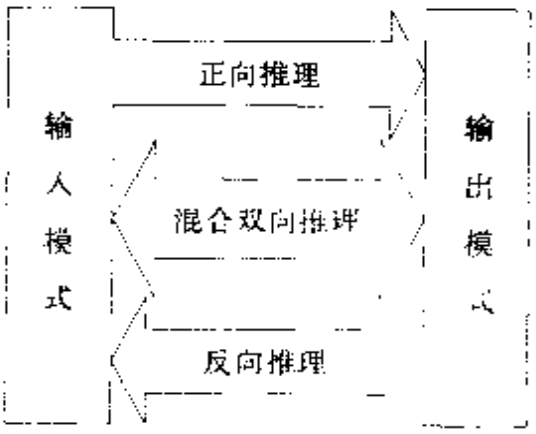


图 4.21 三种推理策略示意图

对于输入模式和输出模式可以有三种类型的数据：实型(整型)数据、由用户定义的符号型数据和集合. 实型(整型)数据有一个给定的实型(整型)范围，符号型数据的域是可能出现的值组成的范围，集合应包括可能出现的所有元素. 如一个简单的医疗诊断专家系统的输入、输出模式可以如下：

输入： 喉疼　　符号型数据(是，否)
　　体温　　实型数据　(36，42)
输出： 病情　　符号型数据(正常，感冒，心绞疼，流感)
　　丧失体能　符号型数据(是，否)
　　治疗药物　集合　　(口服药，青霉素，阿司匹林)

由于符号型数据和集合都可以通过编码将其转换成整型数据，如对于喉疼这一符号型数据，可以用 1 表示是 0 表示否，因此只需要讨论数值型数据一种形式即可. 下面分别就三种推理机制的建立方案予以讨论.

4.5.1 正向推理

神经网络故障诊断专家系统的正向推理过程的实质就是神经网络的计算过程：由已知的输入模式(征兆向量)经过神经计算获得输出模式(故障向量)，因此神经网络专家系统采用正向推理机制就显得特别得心应手. 与传统专家系统的正向推理机制相比，神

经网络的正向推理具有很大的优势，具体的表现在：

(1) 是一种并行推理

神经网络同一层神经元在原理上是并行处理的，层间的处理是串行的，而每一层的神经元数目比神经网络的层数(通常只有三层)要大得多，因而可认为是一种并行推理.

(2) 提高了推理速度

传统专家系统采用"递归"、"回溯"、"匹配"等简单手段来实现推理的目的，速度很慢. 而神经网络专家系统采用比规则数目少得多的权值来构造知识库，推理则通过权值数据与输入数据的运算来完成，即由以前的符号运算转变为现在的数值运算，从而大大地提高了推理速度.

(3) 克服了冲突

传统专家系统中，如果输入的事实与多条规则的前提相匹配，便出现了冲突问题，从而影响了专家系统的求解速度与准确性，而神经网络专家系统采用隐式的知识表示方式，通过神经计算来进行求解的推理策略完全避免了冲突.

假设诊断专家系统采用的神经网络为三层，各层的神经元数目依次为 L,M,N，则正向推理的具体步骤为

1) 调入故障诊断知识库.

2) 输入各项故障征兆值 $\{x_1,x_2,\cdots,x_L\}$.

3) 计算隐含层神经元的输出，计算公式如下：

$$o_i=\frac{1}{1+\exp\left\{-\sum_{j=1}^{L}w_{ij}^{(1)}x_j-\theta_i^{(1)}\right\}},\quad i=1,2,\cdots,M$$

4) 计算输出层神经元的输出，计算公式如下：

$$y_i=\frac{1}{1+\exp\left\{-\sum_{j=1}^{M}w_{ij}^{(2)}o_j-\theta_i^{(2)}\right\}},\quad i=1,2,\cdots,N$$

5) 由给定的法则判定输出神经元的输出，如：

$$\text{Fault}[i]=\begin{cases}\text{严重故障}, y_i \geqslant 0.75\\ \text{一般故障}, 0.5 \leqslant y_i < 0.75\\ \text{正常状况}, y_i < 0.5\end{cases}, i=1,2,\cdots,N$$

从正向推理的算法也可以看出：它能很好地处理模糊输入和多故障诊断情况.

前面 4.4 节中关于旋转机械故障诊断专家系统一例即采用正向推理.

由于神经网络的加权不可分性和针对同一种故障输出的征兆输入模式的非唯一性，使得反向推理和混合双向推理与正向推理相比显得特别复杂，尤其是输入输出为任意模糊值时. 现在就简单的情况：神经网络结构只有三层且输入输出只为简单的二值(0，1)，对这两种推理方法进行一些探讨.

4.5.2 反向推理

在 NN-FD-ES 中，反向推理的指导思想是：假设诊断对象存在某一种故障，现在寻找证据(故障征兆)来证实这一假设是否成立.

设计一个反向推理算法应该考虑两方面的因素，1) 尽可能做到只需要少量的证据就可以得出结论，如果一个反向推理算法需要全部的征兆数据才能得出所作假设是否正确，则该算法无疑是失败的；2) 除非在万不得已的情况下，尽可能不要去寻找代价昂贵或很难获取的征兆值.

由于第二个因素是针对具体的对象而言的，因此在这里，只考虑第一个因素. 反向推理的具体步骤为

1) 调入故障诊断知识库.

2) 输入诊断对象怀疑存在的故障原因，将所有输入层神经元的输入值(征兆值)置为未知，所有输入层及隐含层神经元标识为 UNKNOWN.

3) 在标识为 UNKNOWN 的隐含神经元中找出一个，选择条件是在所有的标识为 UNKNOWN 的隐含层神经元中，它与该故

障输出神经元有最大的连接权绝对值，将该隐含神经元的标识改为 KNOWN，求其输出是否有确定值，如有，执行 4)，否则执行 a).

a) 在标识为 UNKNOWN 的输入层神经元中找出一个，选择条件是在所有的标识为 UNKNOWN 的输入神经元中，它与该隐含层神经元具有最大的连接权绝对值，同时，将该输入神经元的标识改为 KNOWN.

b) 对该输入神经元进行询问，要求用户响应：输入是否具有该症状：0 表示没有该症状，1 代表具有该症状.

c) 判断在已知的标识为 KNOWN 输入神经元下能否决定该隐含神经元的输出，如能执行 d)，否则执行 a).

d) 计算在已知的输入情况下是否还有其它标识为 UNKNOWN 的隐含层神经元的输出能被确定.

4) 判断在已知的标识为 KNOWN 的隐含层神经元下能否决定该输出神经元的输出，不能则执行 3)，否则执行 5).

5) 输出神经元是否输出 1？如是，假设正确；如不是，假设错误.

反向推理的难点在于如何在只知道部分输入信息的情况下确定隐含层神经元或输出层神经元的输出，这里采用如下方法：

1) $\alpha+\beta<\theta$，讨论的神经元肯定输出 0；

2) $\alpha+\gamma>\theta$，讨论的神经元肯定输出 1；

3) 其它，　讨论的神经元的输出不定.

α 是标识为 KNOWN 的前层神经元的输入与该前层神经元与讨论神经元的连接权的乘积的代数和，称之为讨论神经元的确定输入；β 是标识为 UNKOWN 且与讨论神经元具有正连接权的前层神经元与讨论神经元的连接权的代数和，称之为讨论神经元的最大未知输入；γ 是标识为 UNKOWN 且与讨论神经元具有负连接权的前层神经元与讨论神经元的连接权的代数和，称之为讨论神经元的最小未知输入；θ 是讨论神经元的阈值.

图 4.22 是根据上面讨论而画出的简化了的程序流程图.

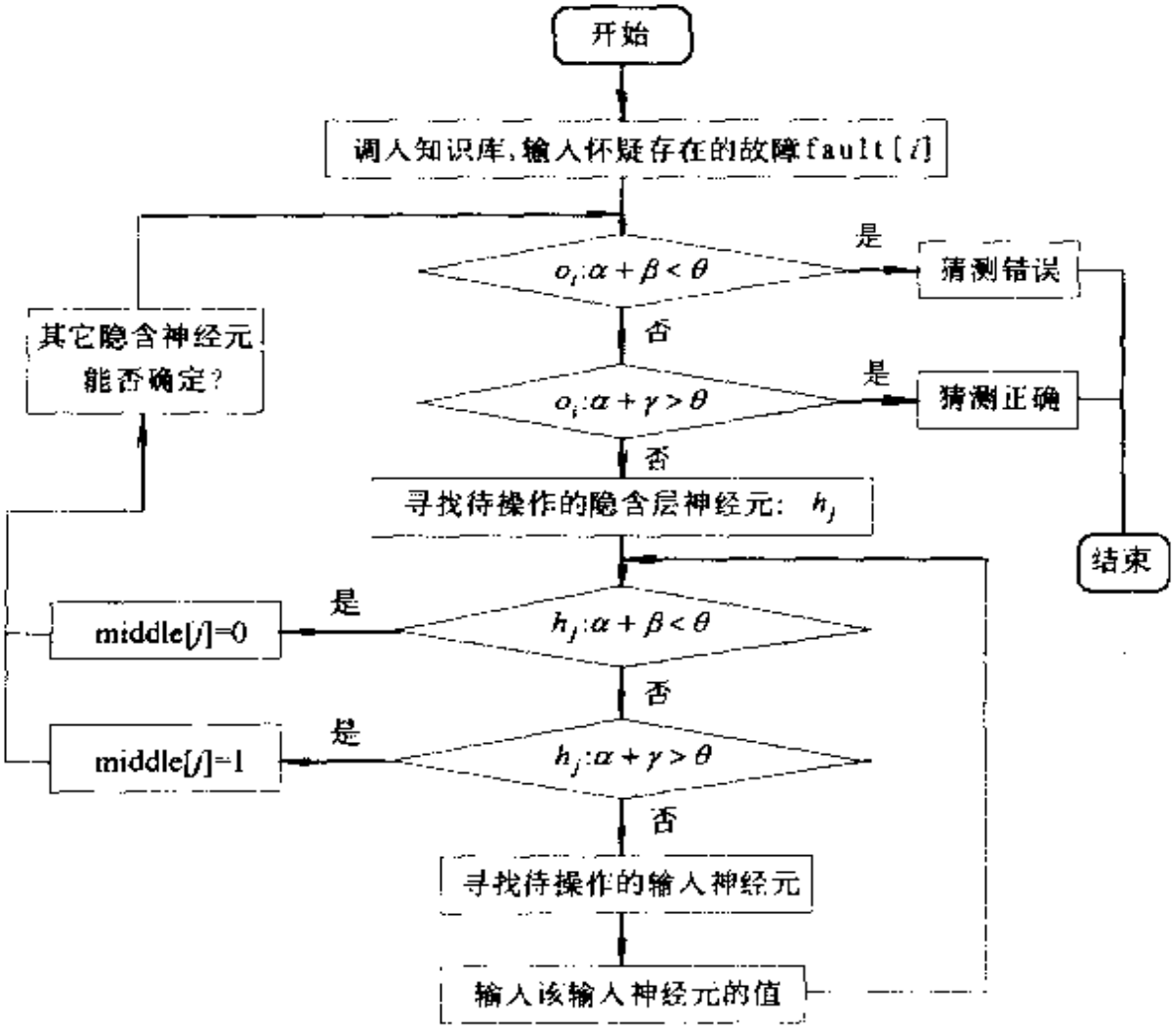


图 4.22　反向推理结构简图

4.5.3　混合双向推理

在神经网络故障诊断专家系统中，混合双向推理的指导思想是：输入已知的部分信息，神经网络根据这些信息提出一个最有可能发生的故障作为诊断对象存在的怀疑故障，然后证实这个假设，若该假设经证实成立，则结束；否则，作出新的假设并继续证实.

在混合双向推理中，一开始便已知的信息往往是凭感官的感觉如观察外表、听声音、闻气味等或凭借极其简单的手段即可获得的信息，如医疗诊断中病人的脸色、心跳等. 在设计混合双向推理算法时，也应该考虑在设计反向推理算法时提出的两条原则.

混合双向推理的具体步骤是：

1）调入故障诊断知识库.

2）输入已有的故障征兆信息，同时将这些输入神经元标识为KNOWN，将其它输入神经元的值赋0，并将其标识为UNKNOWN.

3）在已有的输入下进行神经计算，检查是否有某个输出神经元的输出为1，如有直接给出诊断结果，并执行5），否则执行4）.

4）按控制选择执行下面几步：

a）按顺序将标识为UNKNOWN的输入神经元中的一个的值改为1，再进行神经计算，查看是否有输出神经元的值为1，如有，则将该输出神经元对应故障作为怀疑故障，并对这个输入神经元进行询问，记录询问结果，如是1，则证实怀疑成立并执行5），如是0，则将该输入神经元的标识改为KNOWN，其值为0. 重复执行a），直到最后一个标识为UNKNOWN的神经元.

b）按顺序每次将标识为UNKNOWN的输入神经元中的两个进行组合，将其值都变为1，再进行神经计算，查看是否有输出神经元的值为1，如有，则将该输出神经元对应故障作为怀疑故障，并对这个输入神经元组合进行询问，对响应结果进行如下操作：如果输入都是1，则证实怀疑成立并执行5）；如果输入都是0，则修改这两个输入神经元的标识为KNOWN，其值为0，从头执行b）；如果输入只有1个为1，则修改这两个输入神经元的标识为KNOWN，其值分别为1和0，转去执行a）. 重复执行b）直到最后一组组合.

c）类似b），唯一区别是每次试图改变的输入神经元输入值数目递增1，即每次依次改变3个，4个……并根据用户对询问的响应控制以后的执行步骤.

5）输出推理结果，结束运行.

在混合双向推理的第4）步第c）小步中的处理方法有时可简化：认为输入信息严重不足，而退出诊断.

依照上面的讨论，我们可以简单地画出程序流程图4.23.

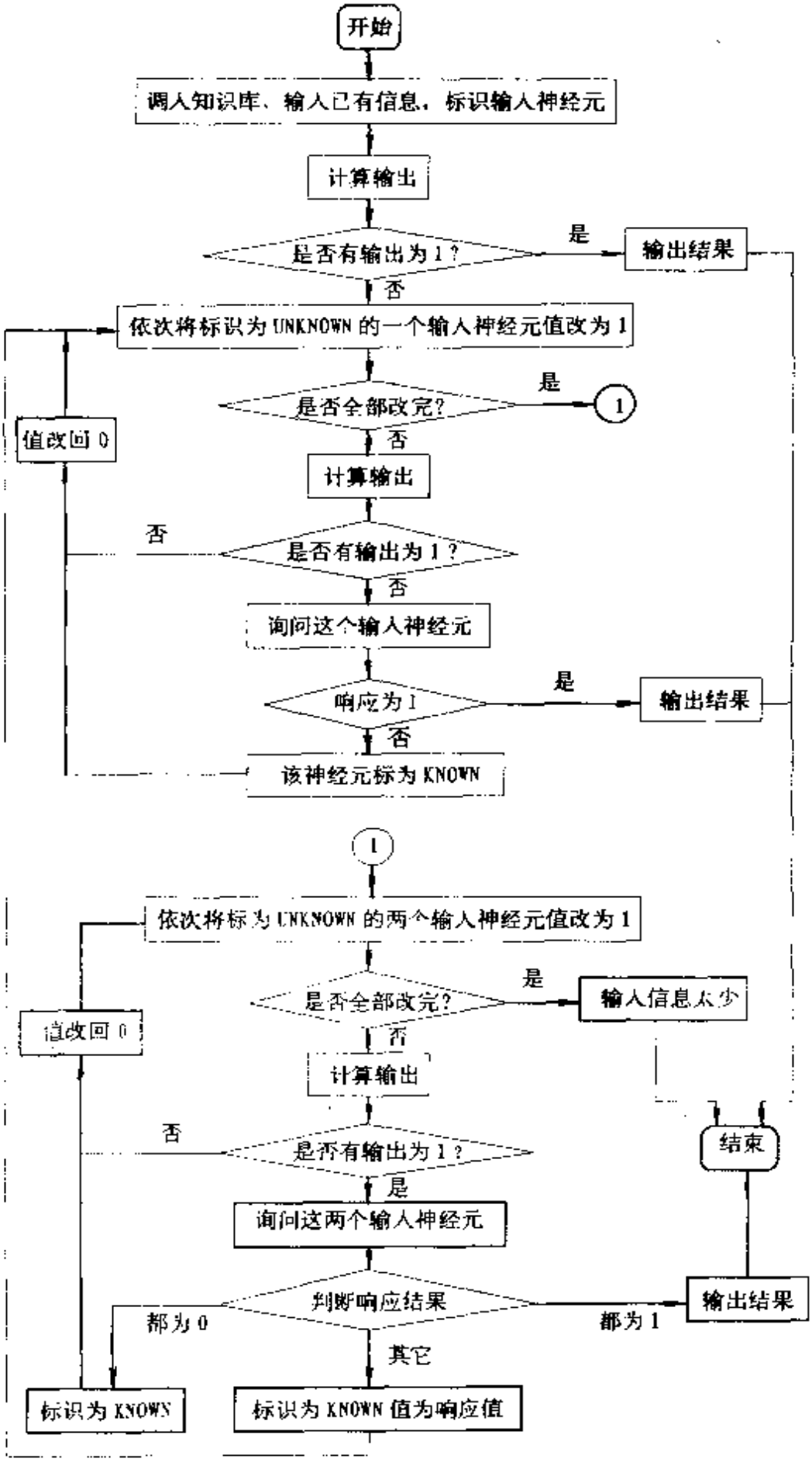


图 4.23　混合双向推理简化流程图

4.5.4 一个使用反向推理和混合双向推理的例子

表 4.11 给出的是某工程机械液压系统的故障征兆与故障原因关系表.根据这个关系表我们采用三层 BP 神经网络设计该液压系统的故障诊断专家系统.

表 4.11 液压系统故障征兆-原因表

关系 / 原因集 / 征兆集		系统排气不彻底 y_1	管道连接松动 y_2	管件破损毛刺 y_3	液压油量不符 y_4	液压油粘度不当 y_5	液压泄老化 y_6	液压油渗有水分 y_7	液压油混入异物 y_8	液压油混入空气 y_9	油箱压力不当 y_{10}	冷却器故障 y_{11}	滤油器故障 y_{12}	液压泵故障 y_{13}	溢流阀故障 y_{14}	换向阀故障 y_{15}	液压缸马达故障 y_{16}	气温异常 y_{17}
不动作	x_1	1	0	1	1	1	0	0	1	0	1	0	1	1	1	1	1	1
动作慢	x_2	0	0	0	1	1	0	0	1	0	1	0	1	1	1	1	1	1
反应迟钝、无力	x_3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
保位无力沉降快	x_4	0	0	1	0	1	0	0	0	0	0	0	0	0	1	1	1	0
抖动	x_5	1	0	0	1	1	0	0	1	1	1	0	1	1	1	1	1	0
不能微动	x_6	1	0	0	0	1	0	0	0	1	1	0	1	1	0	1	1	0
操纵阀失灵	x_7	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0
溢油噪声高	x_8	0	0	1	1	0	0	0	0	1	1	0	1	0	1	0	0	0
油管车体振动	x_9	1	1	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0
液压泵噪声高	x_{10}	1	0	1	1	1	0	1	1	1	1	0	1	1	0	0	1	0
液压马达噪声高	x_{11}	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0
异响	x_{12}	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0
液压缸、阀门系统泄漏	x_{13}	0	0	0	1	0	0	1	0	0	1	1	1	0	0	1	1	1
液压泵马达轴油封泄漏	x_{14}	0	0	0	1	0	0	1	0	0	1	0	0	0	1	0	1	1
高压管路漏	x_{15}	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
低压管路漏	x_{16}	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1
油温过高	x_{17}	0	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	1
磨损粉末过多	x_{18}	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0	1	0

输入层的神经元数目为 18,将隐含层的神经元数目取为 15,输出层的神经元数目为 17.学习样本数目为 17 个.

由于采用硬限幅函数即阶跃函数作为变换函数的多层前馈神经网络至今还没有一种行之有效的学习算法,因此这里采用 Sig-

moid 函数来模拟硬限幅函数. 观察图 4.24 给出的 λ 取不同值时的两条 Sigmoid 函数曲线，曲线(1)为 $\lambda=1$，即

$$f(x)=\frac{1}{1+e^{-x}}$$

的图形，该图形在 $x\in[0,1]$ 内类似一条斜率很小的直线，显然不能用它来模拟硬限幅函数. 曲线(2)为 $\lambda=40$，即

$$g(x)=\frac{1}{1+e^{-40x}}$$

时的图形，该曲线与硬限幅函数极其相似，事实上 $g(0.05)=0.88$，$g(-0.05)=0.12$，$g(0.1)=0.98$，$g(-0.1)=0.02$，因此，可以采用该函数来模拟硬限幅函数.

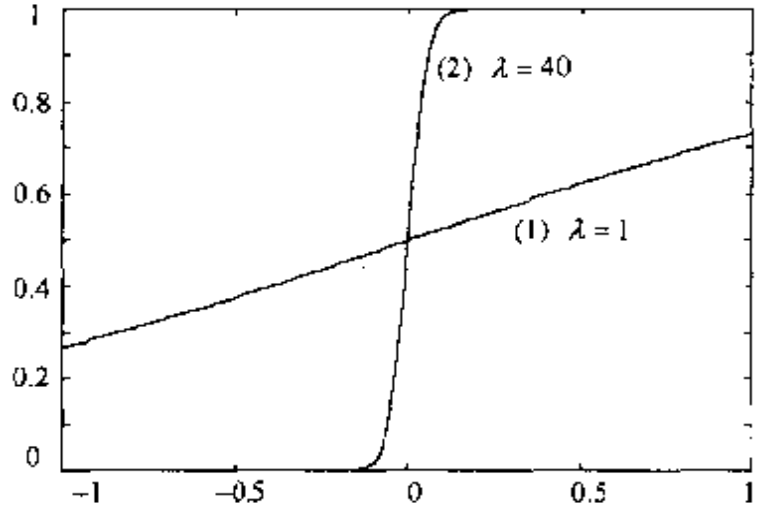


图 4.24 λ 值为 1 和 40 的 Sigmoid 函数

变换函数取为 $g(x)=\dfrac{1}{1+e^{-\lambda x}}\left(\text{net}_j^{(l)}=\sum_i w_{ji}^{(l)}o_i^{(l-1)}-\theta_j^{(l)}\right)$

时的 BP 神经网络学习算法与 $f(x)=\dfrac{1}{1+e^{-x}}$ 的学习算法在原理上是一样的，只是 $g'(\text{net}_j^{(l)})=\lambda o_j^{(l)}(1-o_j^{(l)})$，因而相应的公式为

$$\delta_{pi}^{(2)}=\lambda(t_{pi}-o_{pi}^{(2)})o_{pi}^{(2)}(1-o_{pi}^{(2)}) \qquad (4.5.1)$$

$$i=0,1,2,\cdots,N-1$$

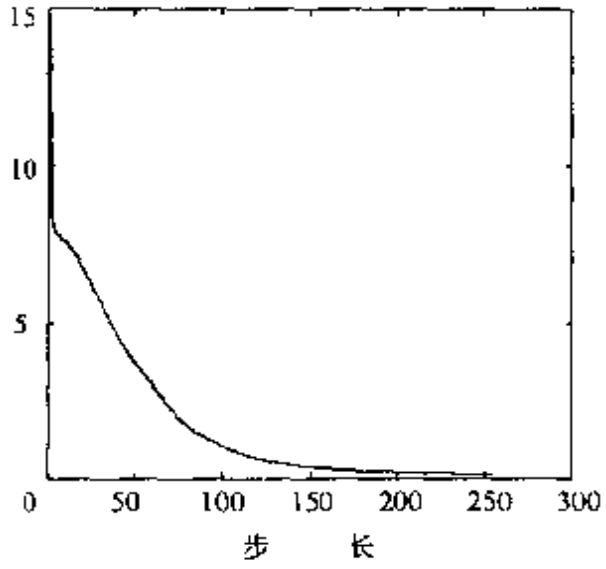
$$\delta_{pi}^{(1)}=\lambda\left(\sum_{k=0}^{N-1}\delta_{pk}^{(2)}w_{ki}^{(2)}\right)o_{pi}^{(1)}(1-o_{pi}^{(1)}) \qquad (4.5.2)$$

$$i=0,1,2,\cdots,M-1$$

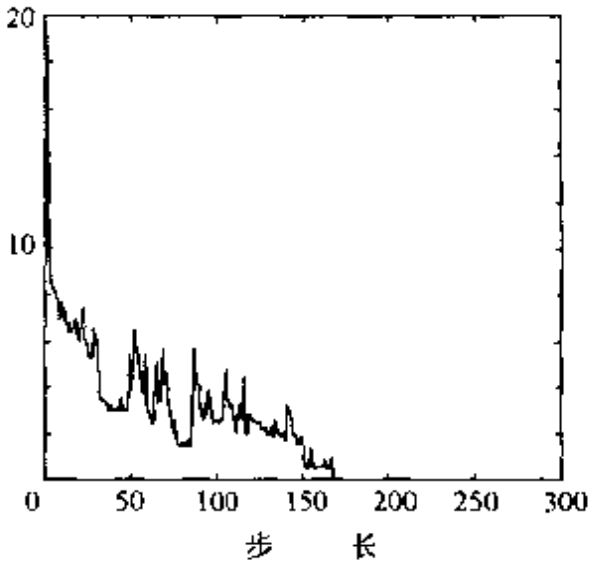
权值修改公式依旧为式(4.3.8).

按照上面的处理方法,结合前面讨论的反向推理和混合双向推理机制我们将这个例子用C语言编成专家系统程序"WJP.C"、"WJP.H"和"BPXL.C"附在后面.

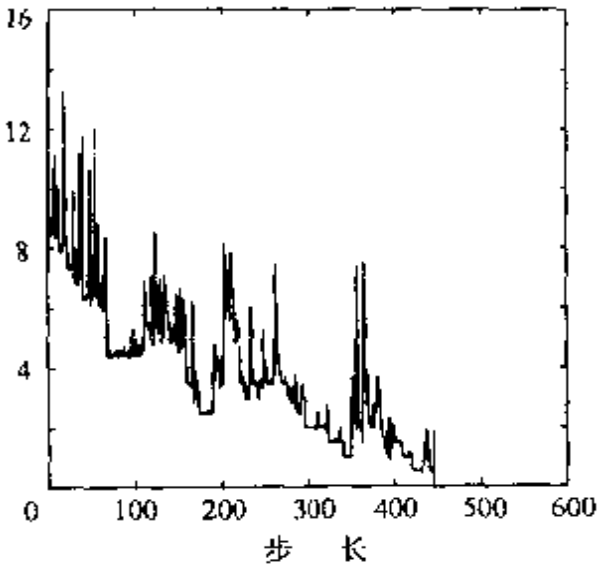
下面,我们来讨论 λ 与收敛速度及收敛效果的关系.图4.25的四个小图分别画出了 $\lambda=1,5,10,40$ 时的误差曲线.



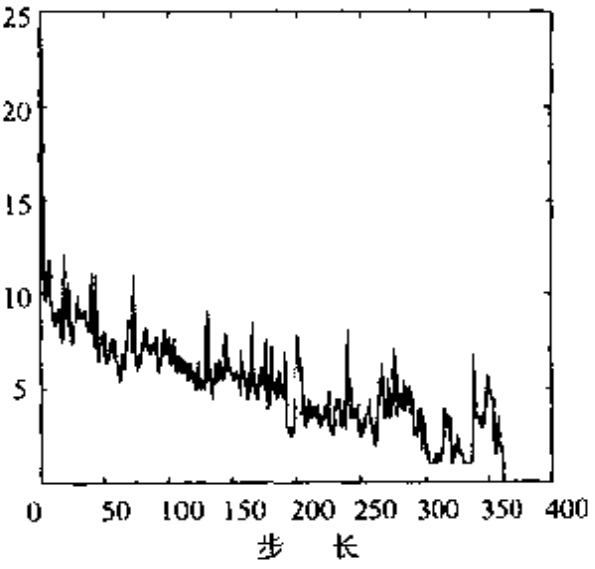
(a) $\lambda=1$ 时的误差收敛曲线



(b) $\lambda=5$ 时的误差收敛曲线



(c) $\lambda=10$ 时的误差收敛曲线



(d) $\lambda=40$ 时的误差收敛曲线

图 4.25 λ 取不同值时的误差曲线图

从上图可以看出 λ 的大小对收敛速度没什么影响，但对收敛效果有很大的影响：λ 小时，收敛曲线平滑，随着 λ 的增大，收敛曲线越来越振荡，不过总体趋势是下降的. 这种现象的直观解释是：当 λ 较大时，权值的细小变化都有可能造成输出的较大变化，从而，使误差曲线发生振荡.

4.6 神经网络故障诊断专家系统的解释机制

一个诊断型神经网络专家系统的性能不仅取决于这个专家系统所能工作的范围和诊断结果的准确程度，而且还取决于它所具有的解释能力. 一个性能优越的神经网络故障诊断专家系统应能对它的推理过程和诊断结果作出合理的解释. 在上一章中，我们看到：在基于规则的传统诊断专家系统中建立这些解释机制是比较简单的，只需要将相关的规则找出并将其转换为自然语言的形式然后摆放在用户面前，就能让用户对推理过程和诊断结果感到心服口服. 然而，在基于神经网络的故障诊断专家系统中，由于知识库中存放的是一些用数字形式隐式表示的连接权值，而不是直接的规则，因而，它的解释机制的建立不可能像传统故障诊断专家系统那样简单，而是要费一番心思.

神经网络故障诊断专家系统对它的推理过程和诊断结果的解释总体来说都是利用网络中的各项数据（包括征兆输入数据、故障输出数据和隐含神经元输出数据）及输入神经元、输出神经元的物理含义并结合知识库中的连接权值来形成规则，其过程相当于神经网络训练的一个逆过程，在训练过程中是将输入信号和教师信号（它们的组合实质上就是规则）作为样本经过训练形成各项权值.

下面，我们只考虑二值三层 BP 神经网络的情况，分别就三种不同的推理机制说明如何对推理过程和诊断结果建立解释机制，对于连续型多层或带有子网络的情况可以对此进行简单推广即可建立起各自的解释机制.

4.6.1 在正向推理中建立对诊断结果的解释

在正向推理中，神经网络必须知道全部的输入值(否则就应采用反向推理或混合双向推理)，因而用户在使用时必须一次性地输入全部征兆数据，而不会提出什么疑问或者说系统设计者可以不给用户提问的机会. 可见，在正向推理中我们只需要讨论如何对诊断结果作出解释，其过程就是如何利用已经知道的输入输出数据形成规则. 确保正确的规则应该按照 4.7 节介绍的方法生成，然后从中找出合适的规则作为解释，只有这样才能保证解释的准确性. 然而，如果真的采用这样方法也就陷入了传统专家系统划定的圈圈之中，而不能突破知识窄台阶的局限性，因此，实际的解释规则生成方法往往如下：

1) 将输出结果为 1 的输出层神经元找出，对其中的某一个 u_i 进行如下五步操作.

2) 考察与该输出神经元具有正的带权输入值的隐含层神经元，即这些神经元满足：

$$p = w_{ij}a_j > 0$$

其中 a_j 为隐含层神经元 j 的输出，将这些神经元按连接权绝对值大小进行降序排列成表 L. 考察与该输出层神经元具有负的带权输入值的隐含层神经元，计算这些隐含层神经元的权值绝对值和 c_0：

$$c_0 = \sum_j |w_{ij}|$$

式中 j 满足 $p=w_{ij}a_j<0$，同时令 $c=0$.

3) 从表 L 中取出排在最前面的隐含层神经元 j，并将这个神经元从表 L 中删除，将其放入表 L_3 中，计算 $c=c+w_{ij}$，判断 $c>c_0+\theta_i$，其中 θ_i 为输出层神经元 i 的阈值，如成立则清除表 L 中的所有元素转 4)，否则返回 3)再执行.

4) 形成规则 1，规则 1 的表示形式是：

IF 表达式 1 AND 表达式 2 AND…AND 表达式 n THEN 结论

规则中的表达式 n 的具体表述形式为某隐含神经元的输出为 1，

结论的具体形式是诊断对象具有故障原因 i.

5) 对于每一个从表 L_3 中提取的隐含层神经元仿照 3)进行操作形成规则：

IF 表达式 1 AND 表达式 2 AND…AND 表达式 n THEN 结论

规则中表达式 n 的具体形式是某个输入神经元输入值为 1 也即具有某种症状，规则中结论的具体形式是某个隐含层神经元的输出为 1.

6) 将 4)和 5)两步产生的规则作为解释语句呈现在用户面前，其中 5)步产生的规则放在前面.

4.6.2 在反向推理中建立对推理过程和诊断结果的解释

在反向推理中，诊断专家系统在运作过程刚开始时，对诊断对象的全部征兆输入值一无所知，它们都是根据推理的需要逐步向用户提出询问，再由用户输入的，这样用户有可能对系统为什么要输入这个值而感到迷惑，因而诊断系统有必要在推理过程中准备回答用户提出的这类问题. 对推理过程作出解释应该结合在推理过程中.

在反向推理过程中，设询问的输入节点为 i，对应故障征兆 i；讨论的隐含节点为 j；怀疑的故障原因为 k，则我们可形成如下的解释语句：

因为是否具有故障征兆 i 关系到隐含层神经元 j 的输出，而隐含层神经元 j 的输出又直接关系到诊断对象是否具有故障原因 k，因此必须输入该征兆值.

在反向推理中设怀疑且经推理得到证实的故障原因为故障 k，对于诊断结果的解释机制可以依照以下步骤进行设计：

1) 找出标识为 KNOWN 且输出为 1 的隐含层神经元，将其分为两部分：与输出神经元 k 具有正的连接权的按降序排列在表 L_1 中；与输出神经元 k 具有负的连接权的排列在表 L_2 中. 设置初始值 $c=0$.

2) 计算 $c_0=a_1+a_2+a_3$，其中 a_1 为表 L_2 中神经元连接权的绝

对值和；a_2 为标识为 UNKNOWN 且与输出神经元 k 具有负连接权的隐含层神经元的连接权(与输出神经元 k)绝对值和；a_3 为输出神经元 k 的阈值.

3) 取出表 L_1 中的排在最前面的隐含神经元 i，设其与输出神经元 k 的连接权为 w_{ki}，将这个神经元从表 L_1 中清除并列进表 L_3 中，计算 $c=c+w_{ki}$，判断 $c>c_0$? 如成立则执行 4)步并将 L_1 中内容全部清除，否则重复执行 3)步.

4) 形成规则. 规则的具体形式为

IF 表达式 1 AND 表达式 2 AND… AND 表达式 n THEN 结论

规则中每个表达式都对应着表 L_3 中的某一项，形式为某隐含神经元输出为 1；结论的具体形式为诊断对象具有故障原因 k.

5) 对于表 L_3 中的每一个隐含层神经元类似前面几步，形成它们各自的规则.

IF 表达式 1 AND 表达式 2 AND…AND 表达式 n THEN 结论

规则中表达式的形式为输入神经元的值为 1，结论的形式为某隐含层神经元输出为 1.

6) 将 4)和 5)两步形成的规则呈现在用户面前，其中 5)产生的规则放在前面.

4.6.3 在混合双向推理中建立对推理过程和诊断结果的解释

在混合双向推理中，系统刚开始时只接受了用户输入的部分征兆信息，这部分信息可能还不能推出诊断对象所具有的故障，因此，在推理过程中需要用户输入更多的征兆信息. 这样，如同反向推理，系统必须准备回答用户提出的类似于为什么需要输入这个值的问题.

在混合双向推理中设询问的节点为 i，怀疑的故障原因为 k，则我们可以形成如下的解释语句作为对推理过程的解释：

诊断对象可能具有故障原因 k，而诊断对象是否具有征兆 i 直接影响到其是否具有故障原因 k，因此我们必须知道征兆 i 的输入.

由于在混合双向推理中，我们将未知的输入都视为 0，因此对

诊断结果进行解释的具体方法与反向推理略有不同.设得到证实的故障原因为故障 k(即对应的输出层神经元 k 输出值为 1),现在对其进行解释,具体的设计步骤为

1) 将标识为 KNOWN 且输出为 1 的隐含层神经元放入表 L 中,再将表 L 分成两部分:具有正的带权输出值(给输出神经元 k)的放在表 L_1 中,表 L_1 中按降序排列;具有负的带权输出值的放在表 L_2 中.设置初始值 $c=0$.

2) 计算表 L_2 中神经元的连接权绝对值和(与输出神经元 k 连接)c_0.

3) 从表 L_1 中取出排在最前面的隐含层神经元 i,计算 $c=c+w_{ki}$,w_{ki}为输出神经元 k 与隐含神经元 i 的连接权,将这个神经元从表 L_1 中删除放入表 L_3 中.判断 $c>c_0+\theta$? θ 为输出神经元的阈值.如成立则删除表 L_1 中全部内容转 4),否则返回 3)重复执行.

4) 形成规则,规则的具体形式为

IF 表达式1 AND 表达式2 AND … AND 表达式n THEN 结论

规则中每个表达式都对应着表 L_2 中的某一项,形式为某隐含神经元输出为 1;结论的具体形式为诊断对象具有故障原因 k.

5) 对于表 L_3 中的每一个隐含层神经元类似前面几步,形成它们各自的规则.

IF 表达式1 AND 表达式2 AND … AND 表达式n THEN 结论

规则中表达式的形式为输入神经元的值为 1,结论的形式为某隐含层神经元输出为 1.

6) 将 4)和 5)两步形成的规则呈现在用户面前,其中 5)产生的规则放在最前面.

从上面的讨论看出:即使我们能对推理过程和诊断结果作出一些解释,但是由于隐含层神经元没有明确的物理意义而在解释过程中又不能绕过它,因而给出的解释也就显得有气无力,远远比不上传统专家系统所能给出的解释.

作为示范,本章后面的附录程序"wjp.c"依照上面的算法给出了部分解释功能.

4.7 传统专家系统与神经网络专家系统的关系

从前面的讨论中我们可以知道，传统专家系统与神经网络专家系统两者相比较各有千秋：传统专家系统在解释机制上比神经网络专家系统要容易实现得多，而且知识的显式表示也直观，让人信服；神经网络专家系统却克服了传统专家系统的致命弱点，具有知识容量大，处理的问题范围广，推理速度快等优势. 正是由于各自都存在优劣两方面，使得两者谁也不能替代谁，而只能同时存在、互相补充.

大家注意到在上一章和本章建立两种不同类型的专家系统时都使用了汽车发动机故障诊断专家系统这一例子，由此可见，对于同一个诊断对象在建立专家系统时我们可以采用两种方案中的任一种，这样，在建立诊断专家系统时便面临一个选择：是建立一个传统诊断专家系统还是建立一个神经网络故障诊断专家系统呢？要解决这个问题必须从以下几个方面进行考虑：

1) 诊断知识的获取途径. 如果领域知识比较容易表示成规则等形式，且领域专家又愿意提供知识给知识工程师，则可以选择建立传统诊断专家系统；反之，如果领域知识是以各种数据的形式表示，且不易总结成规则，则最好采用神经网络故障诊断专家系统.

2) 知识的表现形式. 如果知识是模糊的，而传统专家系统又不易处理模糊知识，因此在这种情况下，最好采用神经网络诊断专家系统或在后面将要介绍的基于模糊逻辑的故障诊断方法.

3) 在线性. 如果诊断专家系统要求在线运行，即要求推理速度很快，这时最好采用神经网络故障诊断专家系统，而不要采用传统诊断专家系统.

4) 是否能获取全部的征兆信息. 传统诊断专家系统采用精确的、易理解的规则表示知识，正是由于这种知识的精确性，使得系统具有解释功能并能在不完全的征兆信息下进行诊断；相反，神经网络诊断专家系统由于单个的神经元没有一般的意义，使得在不

完全的征兆信息下诊断受到限制甚至根本不可能.因此,当很难获得诊断对象的全部征兆信息时,最后采用传统诊断专家系统.

在实际的诊断专家系统中,两者往往是结合在一起的.在一个专家系统中,它的知识库中既包含用规则表示的知识又包含用神经网络连接权表示的知识、推理机制也有两套:神经网络推理和符号推理.这种结合方法是为了充分利用它们各自的优点.

由这两种方案建立的专家系统也可以互相转化,下面将着重讨论这一问题.

4.7.1 从传统专家系统到神经网络专家系统

在基于规则的传统专家系统里,知识是通过规则的方式来表达的;而在神经网络专家系统中,知识是通过对样本的反复学习并在此过程中不断调整网络连接权值,从而使网络误差收敛到全局最小点(此时网络连接权值也稳定)后储存在这些连接权值中的.可见,要实现由传统专家系统到神经网络专家系统的过渡的关键是将规则转换为学习样本.从传统专家系统过渡到神经网络专家系统的具体步骤是:

1) 统计在规则表述中诊断对象可能出现的故障征兆与故障原因数目,分析诊断知识结构,确定神经网络的输入、输出神经元数目及其层次结构.

2) 将传统专家系统知识库中的规则提取出来,形成神经网络的学习样本.

3) 对神经网络进行样本学习,获得各自的连接权值,形成神经网络故障诊断专家系统.

下面我们举一个简单的例子来说明,设某传统诊断专家系统具有下面三条规则:

rule1: IF 诊断对象存在故障征兆 1,2 THEN 诊断对象具有故障 1

rule2: IF 诊断对象存在故障征兆 1,3 THEN 诊断对象具有故障 2

rule3：IF 诊断对象存在故障征兆 2,3 THEN 诊断对象具有故障 3

即系统具有三种征兆，三种故障，且故障知识层次结构简单，不存在中间的故障层次，因而我们可简单地采用单个三层 BP 网络(具有子系统的神经网络专家系统参见 4.3 节)．输入、隐含、输出层的神经元数目都取为 3.

我们以规则中出现的征兆信号作为神经网络的输入，存在的故障作为教师信号，以 1 表示存在该故障征兆或具有该故障的原因，0 表示正常即没有该征兆输入或不存在该故障原因，则我们有下面 3 个学习样本：

样本 1：{{1,1,0},{1,0,0}}

样本 2：{{1,0,1},{0,1,0}}

样本 3：{{0,1,1},{0,0,1}}

通过学习，我们可以获得无穷多种连接权值组合，下图给出一种．

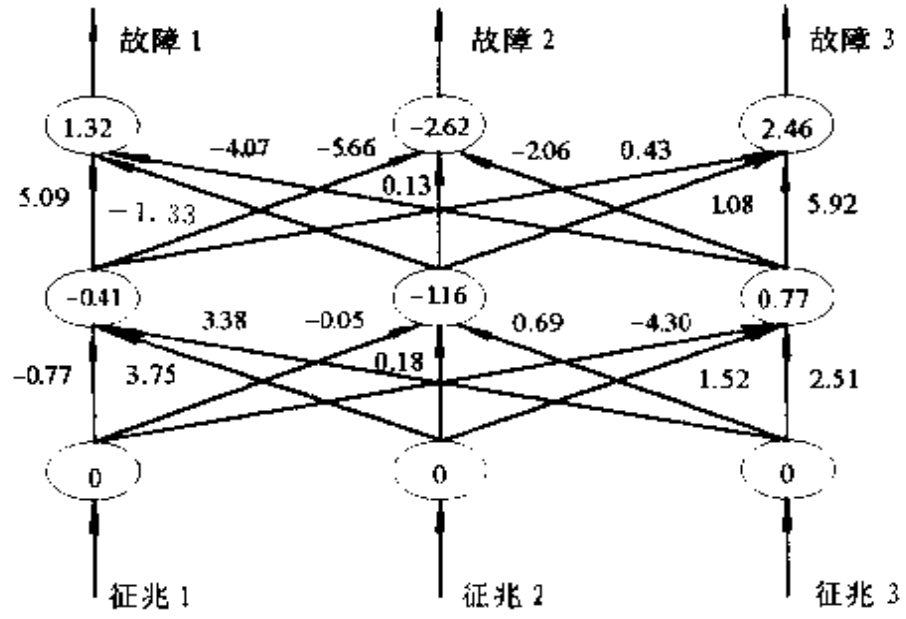


图 4.26　简单故障诊断神经网络结构图

最后，我们将连接权和阈值数据存入数据库中并结合一定的人机接口、推理机制和解释机制即成为一个神经网络专家系统.

4.7.2 由神经网络专家系统到传统专家系统

要实现由神经网络专家系统过渡到传统专家系统就必须从神经网络专家系统现有的连接权知识库中提取出规则来.规则的提取有两个途径:

1) 已知学习样本时,我们可以直接将每一个学习样本转换为一条规则.例如对于神经网络(5 个输入 3 个输出)学习样本:{1,1,0,0,1,0,1,0}相应的规则为

如果诊断对象存在故障征兆 1、故障征兆 2 和故障征兆 5,则诊断对象具有故障 2.

2) 未知学习样本,而只知道连接权值时.这种情况很复杂,因为神经网络具有一个特点是给神经网络以任意一种输入,神经网络都会给出一个输出.这样如果我们以每一种可能的输入输出组合都作为一个样本,同时假设神经网络具有 N 个输入,对于二值逻辑那么我们将获得 2^N 个样本.在这么多的样本里面有些是正确的,有些则是错误的,因为我们只能保证用来训练的样本是正确的,即符合用户要求的.现在我们就简单的情况来讨论如何将用来训练的样本找出来形成规则,这种简单的情况是:神经网络的结构不包含子网络,层数为 3,输入层、隐含层、输出层的神经元数目依次为:L,M,N,所有学习样本的输入输出都只能为 1 或 0,转换函数为

$$f(x)=\frac{1}{1+\exp(-x)}$$

对于学习样本,输出不是非常接近 1 便是非常接近 0(设已知在训练时取教师信号为 0 或 1),而对于非学习样本则不存在这个特性,它们的输出是 0 到 1 之间的连续值,根据这一点,我们可以从神经网络中产生规则,具体算法为

1) 设置各项参数:神经网络结构参数,规则误差限 E_rule,$E.rule\ll 1$.

2) 调入神经网络知识库.

3）对每一种可能输入，进行如下操作：

a）计算输出，并按下式计算误差和：

$$Err\_p = \sum_{i=1}^{N} E\_y_{pi} = \sum_{i=1}^{N} \frac{1}{2} (t_{pi} - y_{pi})^2$$

式中

$$t_{pi} = \begin{cases} 1, & y_{pi} \geqslant 0.5 \\ 0, & y_{pi} < 0.5 \end{cases}, \quad i = 1,2,\cdots,N, \ p = 1,2,\cdots,2^N$$

b）判断在这种输入下是否产生规则，依据是：

是 规 则：if $Err\_p < E\_rule$

不是规则：if $Err\_p > E\_rule$

例如：我们可以从下图所示的权值数据中获得 5 条规则：

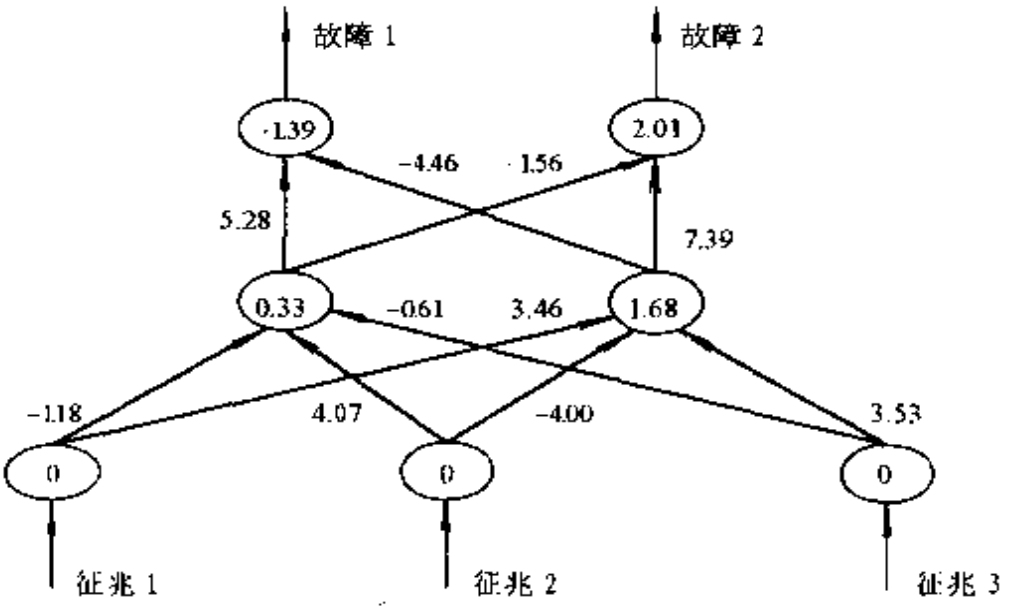


图 4.27 神经网络结构图

rule1：IF 诊断对象存在故障征兆 2 THEN 对象存在故障 1

rule1：IF 诊断对象存在故障征兆 1，2 THEN 对象存在故障 1

rule2：IF 诊断对象存在故障征兆 1，3 THEN 对象存在故障 2

rule3：IF 诊断对象存在故障征兆 2，3 THEN 对象存在故障 1

rule4：IF 诊断对象存在故障征兆 1，2，3 THEN 对象存在故障 1，2

具体的程序请参看 ANN_Rule.c.

通过这两种手段获得规则也就建立了传统专家系统的知识

库，再结合一定的人机接口、推理机制、解释机制，依照上一章的办法就能建立传统故障诊断专家系统.

4.8 其它神经网络模型在故障诊断中的应用

在前面的几节中，我们只讨论了基于前馈多层神经网络的故障诊断专家系统，实际上可用于设计故障诊断专家系统的神经网络模型有很多种，而且所建立起来的专家系统各有优劣，如用BP网络建立的专家系统便既有设计简单，知识容量大等优势，也有易陷入局部最小和不能边学习边工作等不足. 下面对几种常用于故障诊断的神经网络模型作一些介绍.

4.8.1 双向联想记忆(BAM)

联想记忆(associative memory)是人脑记忆的一种重要形式，例如：看到某人的名字时，就会联想到这个人的容貌、性格等特征，听到某首歌的曲调就会联想到它的歌词等. 联想记忆就其输入输出模式上讲可分为自联想记忆和异联想记忆两种.

自联想记忆的功能是实现对自身的联想. 联想的过程是首先在联想记忆中存储一组模式 $A_1, A_2, \cdots, A_m$，然后向联想记忆中输入这 m 个模式中的某一个带有噪声的模式 A_i'，自联想记忆就会输出一个与之相应的模式 A_i，且 A_i 与 A_i' 具有某种定义上的最短距离.

异联想记忆能存储成对的模式 $(A_1, B_1), (A_2, B_2), \cdots, (A_m, B_m)$，$A_i$ 和 B_i 是不同向量空间中的向量. 在联想时，设对异联想记忆输入向量 A，则输出向量为 B，且如果 A 与 A_i 最接近，则 B 就是与 A_i 成对的 B_i. 自联想记忆可以也看作是异联想记忆的一种特例，此时 $A_i = B_i$.

从上面的定义也可以看出：联想记忆的操作可以分为两个阶段，信息存储阶段和联想阶段. 在信息存储阶段即学习阶段，联想记忆通过对输入样本的学习来构造一个学习(存储)矩阵，用以将

提供的模式信息全部包含进去；在联想阶段，联想记忆针对当前的输入模式，找出与之最为接近的存储在联想记忆中的相应模式．在寻找最为接近或具有最短距离的存储模式上，通常要用到贴近度等数学工具．

由 Kosko 提出的双向联想记忆(Bidirectional Associative Memory 简称 BAM)属于异联想记忆中的一种，它是由两层层间互连且同层不连的神经元域 F_A 和 F_B 构成的反馈神经网络，拓扑结构如图 4.28 所示：

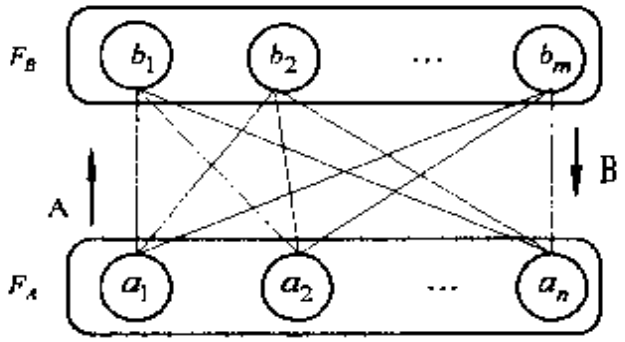


图 4.28　BAM 的拓扑结构

构成 BAM 的是非线性神经元，一般情况下，它们的输出取[0,1]之间的值，但通常情况下域 F_A 中的每一个神经元 a_i 和域 F_B 中的每一个神经元 b_j 的输出值均为二元值{0,1}．

BAM 中所有信息都包含在一个 $n\times m$ 的权值矩阵 M(即前面说过的学习矩阵)中，它的双向联想功能也是通过权值矩阵 M 来实现的．

1．M 矩阵的确定

M 矩阵的确定由以下两步来完成：

1) 对于给定的 p 个学习样本向量(矩阵)对$\{(A_1,B_1),(A_2,B_2),\cdots,(A_p,B_p)\}$中的每一个$(A_i,B_i)(1\leqslant i\leqslant p)$求其双极向量(矩阵)$(X_i,Y_i)$．

所谓双极向量或矩阵是指在二值向量或矩阵的基础上，将 0 代之

以－1 而得到的向量或矩阵. 如二值向量对 $A_i=(1,1,0,0,0,1)$和$B_i=(1,0,1,0)$对应的双极向量对为$X_i=(1,1,-1,-1,-1,1)$和$Y_i=(1,-1,1,-1)$.

2）对于每一个双极向量对计算与之相应的双极伴随矩阵 $X_i^TY_i$，再将它们相加，即得到权值矩阵 W.

$$W=\sum_{i=1}^{p}X_i^TY_i=X_1^TY_1+X_2^TY_2+\cdots+X_p^TY_p \quad (4.8.1)$$

值得注意的是：样本数量 p 必须小于 F_A 和 F_B 中的元素个数，即有公式

$$p<\min(m,n) \quad (4.8.2)$$

因为只有这样才能将不同的样本向量对(A_i,B_i)都放在局部能量最小点上. 由此可见，BAM 网络在存储容量上与 BP 网络相比很小.

下面举一个例子，具体看 BAM 是如何将这些联想对记住的. 设有如下三个记忆对：

$$A_1=(1,0,1,0,0,1)\quad ,\quad B_1=(1,1,0,0)$$
$$A_2=(1,1,0,0,1,1)\quad ,\quad B_2=(1,0,1,0)$$
$$A_3=(0,1,1,1,0,1)\quad ,\quad B_3=(0,1,0,1)$$

由于符合式(4.8.2)，因此 BAM 可以将这些记忆对记住. 现在将这些记忆对变换成联想记忆对，则有

$$X_1=(1,-1,1,-1,-1,1),\quad Y_1=(1,1,-1,-1)$$
$$X_2=(1,1,-1,-1,1,1),\quad Y_2=(1,-1,1,-1)$$
$$X_3=(-1,1,1,1,-1,1),\quad Y_3=(-1,1,-1,1)$$

根据式(4.8.1)我们便可以得到权矩阵：

$$W_3=\sum_{i=1}^{3}X_i^TY_i=\begin{bmatrix}3&-1&1&-3\\-1&-1&1&1\\-1&3&-3&1\\-3&1&-1&3\\1&-3&3&-1\\1&1&-1&-1\end{bmatrix}$$

如果要从 BAM 中擦除某个记忆对，只需要在权矩阵中将与这个记忆对相对应的双极伴随矩阵减掉即可. 如下式便将记忆对 (A_3,B_3)擦除了.

$$W_2=\sum_{i=1}^{2}X_i^TY_i=W_3-X_3^TY_3=\begin{bmatrix}2&0&0&-2\\0&-2&2&0\\0&2&-2&0\\-2&0&0&2\\0&-2&2&0\\2&0&0&-2\end{bmatrix}$$

2. 双向联想功能的实现

BAM 通过以下几步来实现双向联想功能

1) 将输入模式 A 送入 BAM 的域 F_A 中.

2) 计算域 F_B 中各节点的接收值 $b_j(t)$，$j=1,2,\cdots,m$

$$b_j(t)=\sum_{i=1}^{n}a_i(t)w_{ij} \tag{4.8.3}$$

式中 w_{ij}为矩阵中第 i 行 j 列的元素.

计算域 F_A 中各节点的接收值 $a_i(t)$，$i=1,2,\cdots,n$

$$a_i(t)=\sum_{j=1}^{m}b_j(t)w_{ij} \tag{4.8.4}$$

3) 修正 a_i,b_j 的值

$$a_i(t+1)=\begin{cases}1, & a_i(t)>\theta_i\\ a_i(t), & a_i(t)=\theta_i\\ 0, & a_i(t)<\theta_i\end{cases} \tag{4.8.5}$$

$$b_j(t+1)=\begin{cases}1, & b_j(t)>\theta_j\\ b_j(t), & b_j(t)=\theta_j\\ 0, & b_j(t)<\theta_j\end{cases} \tag{4.8.6}$$

式中 θ_i,θ_j 分别为 a_i,b_j 的阈值.

4) 检查 a_i,b_j 是否已稳定(即 a_i,b_j 不再变化)，如不稳定返回 2)，如稳定则结束，此时 F_B 的输出即为 A 的联想结果.

3. BAM 在故障诊断中的应用例子

在 BAM 中，域 F_A 和域 F_B 为不同的向量空间，如果将域 F_A 与故障征兆向量对应，域 F_B 与故障原因向量对应则可以建立故障诊断专家系统.

分析一个生产过程控制系统，结构如图 4.29 所示，整个系统包括调节器、阀门、测量仪表和控制对象. 系统中的某个部分出现故障都将导致整个系统的故障，或者系统异常.

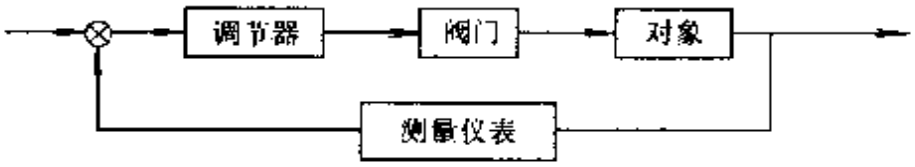


图 4.29　生产过程控制系统

为方便讨论，这里仅诊断调节器和阀门两部分故障. 采用图 4.28 所示的两层 BAM 神经网络，输入层有四个神经元，它们的阈值均置为零，令 $A_i=(a_{i1},a_{i2},a_{i3},a_{i4})$ 为输入向量，输入各种故障现象. 四个输入神经元的定义如下：

$a_{i1}=1$　表示调节器故障

$a_{i2}=1$　表示调节器误指令

$a_{i3}=1$　表示不工作

$a_{i4}=1$　表示阀工误操作

输出层也是四个神经元，它们的阈值同样均置为零，令 $B_j=(b_{j1},b_{j2},b_{j3},b_{j4})$ 为输出向量，输出解决故障的办法. 四个输出神经元的定义如下：

$b_{j1}=1$　表示检查调节器的传动机构

$b_{j2}=1$　表示检修调节器

$b_{j3}=1$　表示检查阀门的传动机构

$b_{j4}=1$　表示检修阀门

存储样本对的内容和含义为

$A_1=(1,1,0,0)\rightarrow B_1=(1,0,0,0)$　调节器误指令→检查调节器的传动机构

$A_2=(1,0,1,0)\rightarrow B_2=(0,1,0,0)$　调节器不工作→检修整个调节器

$A_3=(0,1,0,1)\rightarrow B_3=(0,0,1,0)$　阀门误操作　→检查阀门的传动机构

$A_4=(0,0,1,1)\rightarrow B_4=(0,0,0,1)$　阀门不工作　→检修整个阀门

将这四个样本转化成双极联想记忆对：

$$X_1=(1,1,-1,-1)\rightarrow Y_1=(1,-1,-1,-1)$$

$$X_2=(1,-1,1,-1)\rightarrow Y_2=(-1,1,-1,-1)$$

$$X_3=(-1,1,-1,1)\rightarrow Y_3=(-1,-1,1,-1)$$

$$X_4=(-1,-1,1,1)\rightarrow Y_4=(-1,-1,-1,1)$$

因此它们的联想记忆权矩阵为

$$M=\sum_{i=1}^{4}X_i^TY_i=\begin{bmatrix}2&2&-2&-2\\2&-2&2&-2\\-2&2&-2&2\\-2&-2&2&2\end{bmatrix}$$

现在，假设 BAM 的输入为存入网络中的一个 A_i，那么可以得到

$$A_1M=(4,0,0,-4)\rightarrow(1,0,0,0)=B_1$$

$$A_2M=(0,4,-4,0)\rightarrow(0,1,0,0)=B_2$$

$$A_3M=(0,-4,4,0)\rightarrow(0,0,1,0)=B_3$$

$$A_4M=(-4,0,0,4)\rightarrow(0,0,0,1)=B_4$$

它们的反向信息流分别为

$$B_1M^T=(2,2,-2,-2)\rightarrow(1,1,0,0)=A_1$$

$$B_2M^T=(2,-2,2,-2)\rightarrow(1,0,1,0)=A_2$$

$$B_3M^T=(-2,2,-2,2)\rightarrow(0,1,0,1)=A_3$$

$$B_4M^T=(-2,-2,2,2)\rightarrow(0,0,1,1)=A_4$$

从上面的讨论可以看出：当输入 A 与存入网络中的任一个 A_i

相同时，BAM 只需要一个往返即可以使系统稳定，且输出为相应的 B_i.

当输入不是 A_i 中的一个时，BAM 也可以给出满意的结果. 例如输入 $A=(1,0,0,0)$时，按上面的定义规则，只知道该生产过程控制系统具有调节器故障，但究竟是那一类还不清楚，可以依照

$$N(A,A_i)=1-\frac{1}{4}\sum_{j=1}^{4}|a_j-a_{ij} \qquad (4.8.7)$$

来计算 A 与各个 A_i 的海明贴近度，则有

$$N(A,A_1)=N(A,A_2)=0.75$$

$$N(A,A_3)=N(A,A_4)=0.25$$

说明输入的信息与 A_1 和 A_2 较为接近，这跟人们的直观理解也是一致的. 事实上，

$$AM=(2,2,-2,-2)\rightarrow(1,1,0,0)=B=B_1+B_2$$

$$BM^T=(4,0,0,-4)\rightarrow(1,0,0,0)=A$$

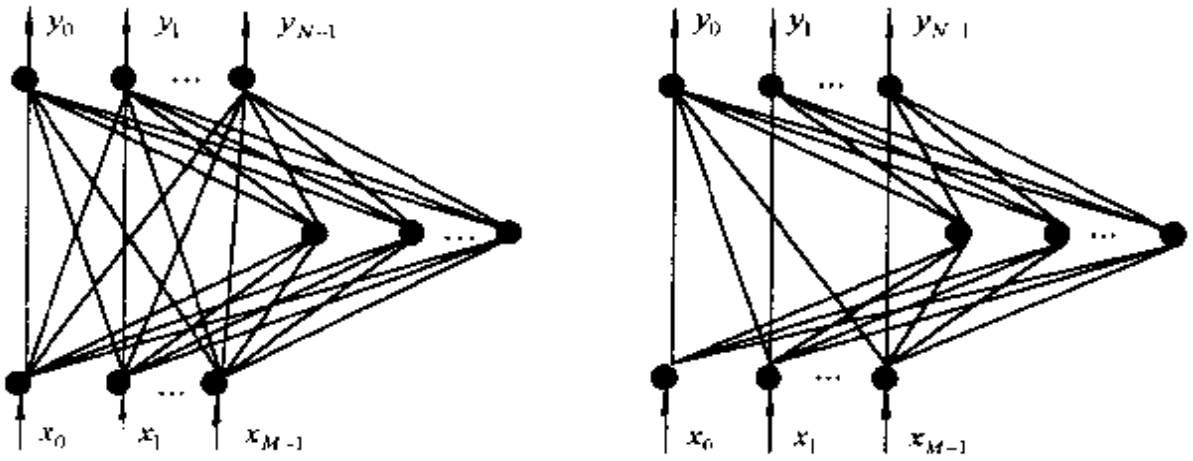
收敛到 $B=(1,1,0,0)$，说明解决办法 B_1 和 B_2 两种都要考虑，这为进行正确的故障诊断提供了依据.

4.8.2 改进的 BP 网络模型

在基于神经网络的故障诊断专家系统中，BP 网络由于具有较强的学习能力及非线性模式识别和联想能力等而受到人们的青睐，但是它也存在着一些缺陷，例如：对于多故障情况，它的联想能力很有限. 针对 BP 网络存在的不足，人们提出了多种改进模型. 作为简单的了解，在这里仅介绍东南大学杨建刚等人提出的两种改进模型.

1. 改进 BP 网络模型 I

改进模型 I 是在标准 BP 网络的基础上增加输入与输出神经元的直接连接，这部分连接采用全互连接方式，如图 4.30(a)所示. 网络层间采用前馈计算公式：



(a) 改进 BP 网络模型I　　(b) 改进 BP 网络模型II

图 4.30　改进 BP 网络模型

对于隐含层节点

$$\mathrm{net}_{pj}^{(1)} = \sum_i w_{ji}^{(1)} x_{pi} - \theta_j^{(1)} \tag{4.8.8}$$

$$o_{pj}^{(1)} = f_j(\mathrm{net}_{pj}^{(1)}) \tag{4.8.9}$$

对于输出层节点

$$\mathrm{net}_{pj}^{(2)} = \sum_i w_{ji}^{(2)} o_{pi}^{(1)} - \theta_j^{(2)} + \sum_l u_{jl} x_{pl} \tag{4.8.10}$$

$$o_{pj}^{(2)} = f_j(\mathrm{net}_{pj}^{(2)}) \tag{4.8.11}$$

式(4.8.10)中 u_{jl} 为输出层与输入层的连接权值.

网络学习仍然采用误差反向传播算法.具体的计算公式为

$$\Delta w_{ij}^{(2)}(n+1) = \eta(t_{pi} - o_{pi}^{(2)}) o_{pi}^{(2)}(1 - o_{pi}^{(2)}) o_{pj}^{(1)} + \alpha \Delta w_{ji}^{(2)}(n) \tag{4.8.12}$$

$$\Delta \theta_i^{(2)}(n+1) = \eta(t_{pi} - o_{pi}^{(2)}) o_{pi}^{(2)}(1 - o_{pi}^{(2)}) + \alpha \Delta \theta_i^{(2)}(n) \tag{4.8.13}$$

$$\Delta w_{ij}^{(1)}(n+1) = \eta\Big(\sum_k (t_{pi} - o_{pi}^{(2)}) o_{pi}^{(2)}(1 - o_{pi}^{(2)}) w_{ki}^{(2)}\Big) \times o_{pi}^{(1)}(1 - o_{pi}^{(1)}) x_{pj} + \alpha \Delta w_{ij}^{(1)}(n) \tag{4.8.14}$$

$$\Delta \theta_i^{(1)}(n+1) = \eta\Big(\sum_k (t_{pi} - o_{pi}^{(2)}) o_{pi}^{(2)}(1 - o_{pi}^{(2)}) w_{ki}^{(2)}\Big) \times o_{pi}^{(1)}(1 - o_{pi}^{(1)}) + \alpha \Delta \theta_i^{(1)}(n) \tag{4.8.15}$$

$$\Delta u_{ij}(n+1)=\eta(t_{pi}-o_{pi}^{(2)})o_{pi}^{(2)}(1-o_{pi}^{(2)})x_{pj}+\alpha\Delta u_{ij}(n) \tag{4.8.16}$$

上面五个式子中，带有 Δ 的项为修改量.

2. 改进 BP 网络模型 Ⅱ

改进 BP 网络模型 Ⅱ 是在改进 BP 网络模型 Ⅰ 的基础上稍加修改而建立的，它的输入与输出两层间不采用全互连接方式，而只是根据实际情况的需要，将有关联的输入输出节点连接，如图 4.30(b)所示. 建立改进 BP 网络模型 Ⅱ 的难点在于如何确定哪些输入输出神经元之间需要连接，哪些不需要连接.

设诊断对象故障集为 $Y=\{y_0,y_1,\cdots,y_{N-1}\}$，征兆集为 $X=\{x_0,x_1,\cdots x_{M-1}\}$，每个故障发生的概率为 $P(y_i)=S(y_i)/S$，$S(y_i)$ 为已有样本中故障 y_i 发生的次数，S 为已有样本总数. 用相应故障 y_i 的熵来描述系统的不确定性.

$$H(y_i)=-P(y_i)\ln[P(y_i)] \tag{4.8.17}$$

用 $P(y_i|x_k)$ 表示征兆 x_k 对故障 y_i 发生的条件概率，$P(y_i|x_k)=S_{ki}/S_k$，S_{ki} 表示所有样本中故障 y_i 在 x_k 出现情况下发生的次数，S_k 为所有样本中 x_k 出现的次数. 定义征兆 x_k 对故障 y_i 发生的条件熵：

$$H(y_i|x_k)=-P(y_i|x_k)\ln[P(y_i|x_k)] \tag{4.8.18}$$

由此，征兆 x_k 对判断故障 y_i 所提供的信息量为

$$I(y_i|x_k)=H(y_i)-H(y_i|x_k) \tag{4.8.19}$$

从上式的定义可知，如果 $I(y_i|x_k)$ 的值越大，那么征兆 x_k 对判断故障 y_i 所起的作用也就越大. 记 C 为阈值，如果 $I(y_i|x_k)\geqslant C$，则称输入节点 x_k 与输出节点 y_i 有关，这两个节点也就需要连接.

网络的权值学习公式同改进模型 Ⅰ，但是输入和输出层间的连接权系数的学习只针对相连的权系数(即 $u_{ij}\neq 0$)进行.

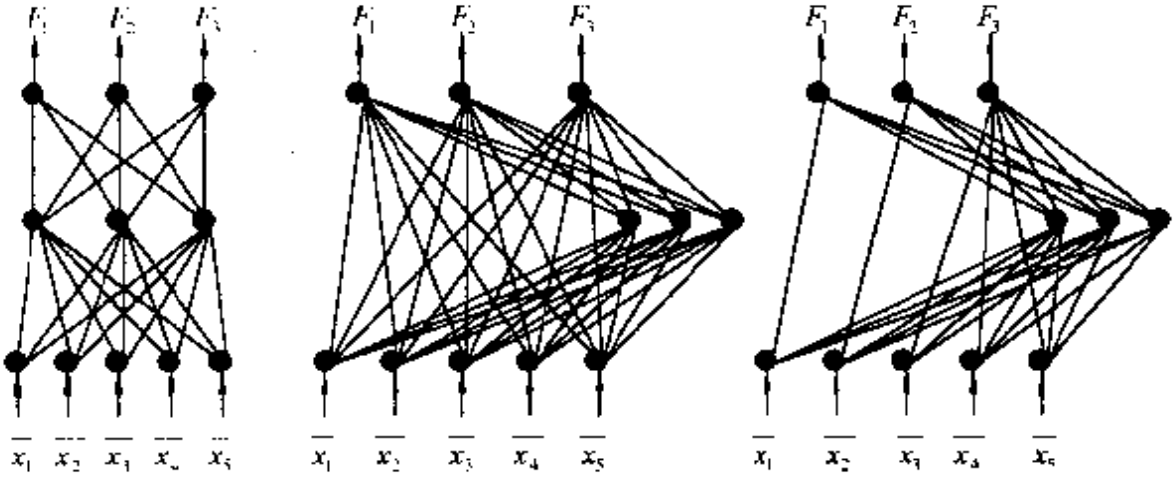
3. 改进 BP 网络在故障诊断中的例子

这里仍然以 4.3 节讲过的故障模拟试验台为例，在改进 BP

网络模型Ⅱ中，可取阈值 $C=0.1$，根据输入神经元与输出神经元的信息量计算公式有

$$I(y_0|x_0)=0.368\geqslant C \quad , \quad I(y_1|x_1)=0.215\geqslant C$$
$$I(y_2|x_2)=0.367\geqslant C \quad , \quad I(y_2|x_3)=0.188\geqslant C$$
$$I(y_2|x_4)=0.188\geqslant C \quad , \quad I(y_i|x_j)_{\text{other}}<C$$

如果隐含层的神经元都取为 3，这样可得到如图 4.31 所示的三种网络模型.



(a) BP 网络　　(b) 改进模型Ⅰ　　(c) 改进模型Ⅱ

图 4.31　模拟试验台故障诊断的三种网络模型

仍然以表 4.1 中的十五个样本对这三个网络进行训练，然后输入表 4.12 所示的六种模拟故障中的征兆数据，得出的诊断结果也列在表中，从表中可以看出：一般的 BP 网络在诊断组合故障时效果不是很好，改进模型Ⅰ和Ⅱ的效果要好得多.

表 4.12　三种网络对组合故障诊断结果的比较

序号	模拟故障	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	$\overline{x_4}$	$\overline{x_5}$	BP 结论	改进Ⅰ的结论	改进Ⅱ的结论
1	B 大，C 大	0.04	0.97	0.45	0.33	0.20	C 中	B 中，C 中	B 大，C 大
2	B 中，C 大	0.04	0.78	0.45	0.35	0.15	C 大	C 大	B 中，C 大
3	A 大，B 大	0.91	0.92	0.02	0.04	0.02	A 大	A 大，B 大	A 大，B 大
4	A 中，B 中	0.62	0.71	0.03	0.02	0.02	A 中	A 中，B 中	A 中，B 大
5	A 大，C 大	0.95	0.32	0.45	0.26	0.25	A 大，C 中	A 大	A 大，C 大
6	A 中，C 大	0.60	0.35	0.46	0.30	0.22	A 中，C 大	A 中，C 大	A 中，C 大

4.8.3 用作分类器的 ART

1. ART 基本知识

自适应共振理论(Adaptive Resonance Theoty 简称为 ART)是由美国 Boston 大学的 S. Grossberg 和 A. Carpentet 提出的. 在 ART 中,当输入与一个已存储的样本充分相似时,我们就说它们是共振的. 由此可见,要确定一个输入样本是否与一个已存储的样本发生共振,就必须首先定义一个参数以确定这两个样本的相似程度是否达到要求,这个参数就是警戒阈值 ρ. $0<\rho\leqslant 1$, ρ 小要求的相似程度就比较低,反之要求的相似程度比较高.

在结构上,ART 系统可分为两个子系统:注意子系统(attentional subsystem)和取向子系统(orienting subsystem). 注意子系统用于处理熟悉的或以前学习过的模型,可对熟悉的事件建立起精确的内部表示;取向子系统用于确定新的模式是不是熟悉的,同时产生识别代码.

目前开发的 ART 主要有三种模型:ART1,ART2 和 ART3. ART1 是针对二进制信号输入的,即输入矢量 X 的每个分量 x_i 只能取 0 或 1;ART2 是针对连续信号输入的,此时输入矢量 X 的每个分量 x_i 可取任意模拟信号;ART3 兼容了前两种的功能,同时将两层神经网络扩展为任意多层神经网络. 下面将要介绍的用作分类器的 ART 属于 ART1.

与大多数神经网络相比,ART 具有如下特点:

1) 可以实时学习. 在 ART 中,学习和工作是不可分的,能实现边学习边工作.

2) 能适应非平稳的工作环境.

3) 可以通过注意子系统来实现对已学习过的对象的稳定、快速识别;通过取向子系统来实现对未学习过的新对象的迅速适应.

4) 具有自归一能力,根据某些特征在全体特征中占的比例,有时将其作为关键特征来处理,有时又将其作为噪声来处理.

5) 学习不需要事先知道样本的结果,属于无监督学习,如果

对环境作出错误反应则自动提高“警觉性”，迅速识别对象.

6）容量不受输入通道数的限制，存储对象也不要求是正交的.

2. 用作分类器的 ART 的拓扑结构

当 ART 用于分类器时，它具有图 4.32 所示的拓扑结构，其中下面是输入节点，上面是输出节点. 输入节点与输出节点之间采取双向的全连接即输入层与输出层之间不仅有自上而下的连接也有自下而上的连接，换句话说，任意两个输入神经元和输出神经元间都存在两个连接权值；输出节点之间也采取双向的全连接. 在该网络中输入信号 x_0, x_1, x_2 和输出信号 y_0, y_1 的取值都只能为 0 或 1.

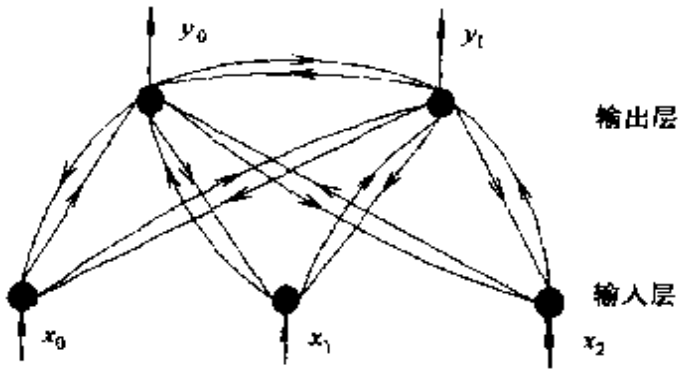


图 4.32 用作分类器的 ART 拓扑结构

3. 用作分类器的 ART 学习算法

设输入层有 N 个神经元，输出层有 M 个神经元，自上而下的连接权值为 W_{ji}，自下而上的连接权值为 W_{ij}，警戒阈值为 ρ，则由 Carpentet 和 Grossberg 提出的 ART 分类学习算法为

1）初始化

$$\begin{cases} W_{ji}(0) = 1, & W_{ij}(0) = \dfrac{1}{N+1} \\ 0 \leqslant i \leqslant N-1, & 0 \leqslant j \leqslant M-1 \\ \text{设置 } \rho, & 0 \leqslant \rho \leqslant 1 \end{cases} \tag{4.8.20}$$

2）给 ART 提供新的输入

3) 计算匹配值

$$\mu_j = \sum_{i=1}^{N-1} W_{ji}(t)x_i, \quad 0 \leqslant j \leqslant M-1 \qquad (4.8.21)$$

其中 μ_j 是输出节点 j 的输出，x_i 是输入节点 i 的输入，取值为 0 或 1.

4) 选择一最佳匹配样本

$$\mu_j^* = \max_j \{\mu_j\} \qquad (4.8.22)$$

这可通过输出节点的扩展抑制权来获得：

5) 警戒线检验

令 $$\| X \| = \sum_{i=0}^{N-1} X_i, \quad \| Y \| = \sum_{i=0}^{N-1} W_{ji}X_i \qquad (4.8.23)$$

判断 $$\| Y \| / \| X \| > \rho \qquad (4.8.24)$$

是否成立，若是执行 7)，否则执行 6).

6) 将 4)步的最佳匹配暂时置为零，不参与比较，转 3)步.

7) 调整网络权值

$$W_{ji}^*(t+1) = W_{ji}^*(t)x_i \qquad (4.8.25)$$

$$W_{ij}^*(t+1) = \frac{W_{ji}^*(t)x_i}{0.5 + \sum_{i=0}^{N-1} W_{ji}^*(t)x_i} \qquad (4.8.26)$$

8) 取消 6)步中节点为零的限制，转第 2)步.

4. 用作分类器的 ART 在故障诊断中的应用例子

由于图 4.32 所示的网络可以有效地将不同的输入模式按其类型分开，因此可以将其运用于故障诊断中. 仍然以故障征兆向量作为输入向量，输出向量对应于不同的故障原因. 为了以后能诊断更多的故障类型，可以适当地增加输出神经元的数目.

我们仍以在本节 BAM 网络中举过的生产过程控制系统为例. 输入神经元取 4 个，输出神经元取 6 个，将 4 个输入样本：

$$X_1 = \{1,1,0,0\}, \ X_2 = \{1,0,1,0\}$$

$$X_3 = \{0,1,0,1\}, \ X_4 = \{0,0,1,1\}$$

依次输入后，可以得到如下的 ART 网络：

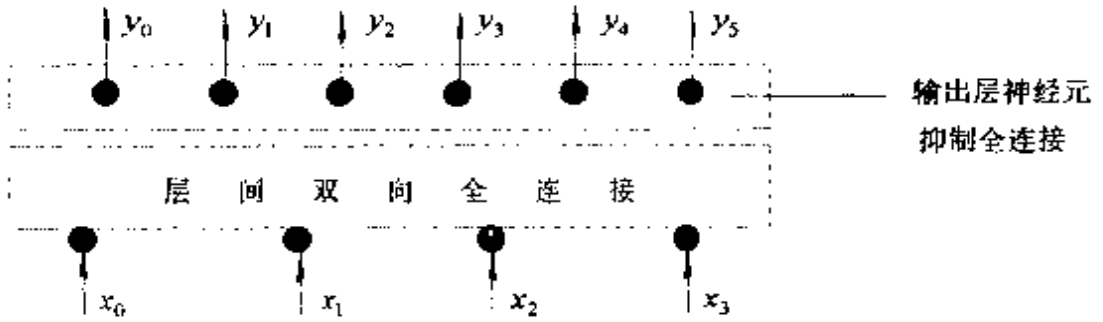


图 4.33 生产过程控制系统故障诊断 ART 网络图

图 4.33 中 $x_0, x_1, x_2, x_3, y_0, y_1, y_2, y_3$ 的定义分别与前面的 $a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4$ 定义相同. y_5, y_6 暂时没有定义留待以后扩充用.

诊断时，如果输入征兆向量为四个已学习样本中的一个，如 $X_1=\{1,1,0,0\}$，则输出 $Y=\{1,0,0,0,0,0\}$ 为一已知故障类型；如果输入为一新样本，则输出可能为一新的故障类型.

附录　神经网络故障诊断专家系统 C语言程序集

在本附录中共有六个文件，其中“YJG.C”，“YJG.H”和“BPXL.C”用于第4.4节中的转子试验台的故障诊断例子，“YJG.H”为头文件，用于定义诊断神经网络的结构参数、学习参数、输入输出神经元的物理意义等，“BPXL.C”为三层BP网络的训练程序，诊断程序中的推理算法采用正向推理，征兆输入可为任意实数；“WJP.C”，“WJP.H”和“BPXL.C”用于第4.5节中的液压系统的故障诊断例子，“WJP.H”为头文件，作用同“YJG.H”，诊断程序的推理算法采用正向推理、反向推理和混合双向推理三种，征兆输入只能为0或1两种，不能为其它；“ANN-RULE.C”为一个独立的小程序，用于将神经网络连接权转变为规则.

```
/**************************************************/
/*      文件名              YJG.C                 */
/*      BP 神经网络旋转机械故障诊断专家系统       */
/*      本神经网络专家系统采用正向推理            */
/*      隐含层和输出层神经元变换函数为            */
/*                Sigmoid 函数                    */
/**************************************************/

#include"YJG.H"
#include"bpxl.c"

/*结论子程序*/
void result()
```

```
{
    int i,j,l;
    double net;
    static char Degree[2][4]={"大","中"};
    char FLT_DGR[12];
    /*计算隐含层的输出*/
    for(j=0;j<HN;j++)
        {
            net=0;
            for(l=0;l<IN;l++)
                net=net+cause[l]*w_h_i[j][l];
            net=net+w_h_i[j][IN];
            o_h[0][j]=sigmoid(net);
        }
    /*计算输出层的输出*/
    for(j=0;j<ON;j++)
        {
            net=0;
            for(l=0;l<HN;l++)
                net=net+o_h[0][l]*w_o_h[j][l];
            net=net+w_o_h[j][HN];
            o_o[0][j]=sigmoid(net);
        }
    /*判断并输出结果*/
    for(j=0;j<ON;j++)
        {
            if(o_o[0][j]>0.7)       l=0;
            else if(o_o[0][j]>0.5)    l=1;
            if(o_o[0][j]>0.5)
                {
                    strcpy(FLT_DGR,Degree[l]);
                    strcat(FLT_DGR,Fault[j]);
                    printf("%s\n",FLT_DGR);
```

```
                }
        }
}

/* 诊断子程序 */
void diagnose()
{
    int i,j,l,choice1;
    char know_nm[10];
    struct fault exam;
    FILE *know_fl;
    printf("请输入知识库文件名:");
    scanf("%s",know_nm);
    if((know_fl=fopen(know_nm,"rb"))==NULL)
        {
            printf("不能打开知识库文件\n");
            exit(0);
        }
    /* 读取权值数据 */
    rewind(know_fl);
    for(j=0;j<HN;j++)
        for(i=0;i<IN+1;i++)
            fread(&w_h_i[j][i],8,1,know_fl);
    for(j=0;j<ON;j++)
        for(i=0;i<HN+1;i++)
            fread(&w_o_h[j][i],8,1,know_fl);
    fclose(know_fl);
    /* 开始诊断 */
    printf("\n\n ********  选择诊断例子  ******** \n\n");
    printf("          1          程序中给出的例子  \n");
    printf("          2          用户自己的例子  \n\n");
    printf(" 您选择.....\n");
    scanf("%d",&choice1);
```

```
    switch(choice1)
        {
            case 1:
              for(i=0;i<SIZE_FLT;i++)
                {
                  printf("第%2d个样本具有的故障是:  \n",i+1);
                  for(j=0;j<IN;j++)
                    cause[j]=samp_flt[i].in_sign[j];
                  result();
                }
              break;
            case 2:
              while(1)
                {
                  printf("请依次输入各种征兆:\n");
                  for(i=0;i<IN;i++)
                    {
                      printf("\n%s 的数据是:",in_fault[i]);
                      scanf("%lf",&cause[i]);
                    }
                  printf("您输入的例子的故障是:  ");
                  result();
                  printf("\n\n ******** 选择功能 ******** \n\n");
                  printf("          1          继续诊断  \n");
                  printf("          2          退出      \n\n");
                  printf(" 您选择.....\n");
                  scanf("%d",&choice1);
                  if(choice1!=1) break;
                }
        }
}

/* 主程序 */
```

```
main()
{
    int choice;
    printf("\n ***** 欢迎进入旋转机械故障诊断专家系统 ***** \n\n\n");
    printf("                功能选择\n\n");
    printf("           1       神经网络学习\n");
    printf("           2       神经网络诊断\n");
    printf("           3       退出\n");
    printf("     您选择.....");
    for(;;)
        {
            scanf("%d",&choice);
            if(choice>0&&choice<4) break;
            printf("输入越界,重输");
            sound(7);
            delay(2000);
        }
    switch(choice)
        {
            case 1:
              learn();
              break;
            case 2:
              diagnose();
              break;
            case 3:
              exit(0);
        }
    return 0;
}

    /**********************************************/
    /*   文件名                YJG.H          */
```

```
/*   本程序为 YJG.C 的头文件                          */
/*   定义有关的神经网络参数和学习诊断样本              */
/*   各种全局变量也在此定义                           */
/*****************************************************/

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<dos.h>
#include<graphics.h>
#include<string.h>
#include<conio.h>
#include<time.h>

#define IN 5                          /* 输入层神经元数目 */
#define HN 5                          /* 隐含层神经元数目 */
#define ON 3                          /* 输出层神经元数目 */
#define SIZE 15                       /* 学习样本数目 */
#define SIZE_FLT 9                    /* 诊断样本数目 */
#define EITA 0.6                      /* 学习步长 */
#define ZITA 0.0001                   /* 权值收敛因子 */
#define BETA 0.1                      /* 误差收敛因子 */
#define NAMTA 1                       /* Sigmoid 函数修正因子 */

static double w_h_i[HN][IN+1];        /* 输入层到隐含层的权值 */
static double w_o_h[ON][HN+1];        /* 隐含层到输出层的权值 */
static double o_h[SIZE][HN];          /* 隐含层的输出 */
static double o_o[SIZE][ON];          /* 输出层的输出 */
float Sum_Err[10000];                 /* 神经网络的总体误差 */
float Sig_Err[SIZE];                  /* 神经网络的单样本误差 */
static double cause[IN];              /* 征兆 */
```

```
/* 故障原因中文描述 */
static char Fault[3][10]={"油膜振荡","不平衡","不对中"};
/* 故障征兆中文描述 */
static char in_fault[IN][10]={"0.4—0.5x","      1x",
    "      2x","      3x","      >3x"};

/* 学习样本 */
struct samples
{
    double in_sign[IN];              /* 输入信号 */
    double tch_sign[ON];             /* 教师信号 */
} samp[SIZE]=
    {{0.88,0.22,0.02,0.04,0.06,0.9,0.1,0.1},
     {0.90,0.20,0.05,0.02,0.02,0.9,0.1,0.1},
     {0.92,0.21,0.03,0.01,0.02,0.9,0.1,0.1},
     {0.85,0.25,0.06,0.02,0.01,0.9,0.1,0.1},
     {0.82,0.28,0.05,0.04,0.03,0.9,0.1,0.1},
     {0.04,0.98,0.10,0.02,0.02,0.1,0.9,0.1},
     {0.02,1.00,0.08,0.03,0.01,0.1,0.9,0.1},
     {0.05,0.90,0.11,0.05,0.02,0.1,0.9,0.1},
     {0.03,0.96,0.12,0.04,0.03,0.1,0.9,0.1},
     {0.02,0.91,0.08,0.01,0.02,0.1,0.9,0.1},
     {0.02,0.41,0.43,0.34,0.15,0.1,0.1,0.9},
     {0.01,0.52,0.40,0.32,0.10,0.1,0.1,0.9},
     {0.01,0.40,0.47,0.35,0.18,0.1,0.1,0.9},
     {0.02,0.45,0.42,0.28,0.29,0.1,0.1,0.9},
     {0.01,0.48,0.48,0.36,0.20,0.1,0.1,0.9}};

/*诊断样本*/
struct fault
{
    double in_sign[IN];              /* 故障征兆 */
} samp_flt[SIZE_FLT]=
```

```
{{0.89,0.20,0.04,0.03,0.01},
 {0.67,0.20,0.03,0.02,0.02},
 {0.01,0.97,0.10,0.02,0.04},
 {0.02,0.70,0.04,0.02,0.02},
 {0.01,0.35,0.17,0.40,0.20},
 {0.01,0.30,0.34,0.22,0.20},
 {0.95,0.32,0.45,0.26,0.25},
 {0.60,0.35,0.46,0.30,0.22},
 {0.68,0.30,0.35,0.26,0.67}};

/**************************************************/
/*      文件名:              WJP.C                */
/*      BP 神经网络液压系统故障诊断专家系统       */
/*      本神经网络专家系统采用                    */
/*      正向推理、反向推理和混合双向推理          */
/*      作为示范本程序对正向推理结果              */
/*          和反向推理过程作出简单的解释          */
/*      隐含层和输出层神经元变换函数为            */
/*        用 Sigmoid 函数模拟的阶跃函数           */
/**************************************************/

#include"WJP.H"
#include"bpxl.c"

/* 程序结束出口 */
void finish()
    {
      printf("\n ********************************** \n");
      printf("          本次诊断到此结束            \n");
      printf("          希望能够让您满意            \n");
      printf(" ********************************** \n");
      exit(0);
```

```
    }

/*计算输出子程序*/
void compute()
    {
      int j,l;
      double net;
      /*计算隐含层的输出*/
      for(j=0;j<HN;j++)
          {
              net=0;
              for(l=0;l<IN;l++)
                  net=net+symptom[l]*w_h_i[j][l];
              net=net+w_h_i[j][IN];
              middle[j]=0;
              if(net>0) middle[j]=1;
          }
      /*计算输出层的输出*/
      for(j=0;j<ON;j++)
          {
              net=0;
              for(l=0;l<HN;l++)
                  net=net+middle[l]*w_o_h[j][l];
              net=net+w_o_h[j][HN];
              trouble[j]=0;
              if(net>0) trouble[j]=1;
          }
    }

/*正向推理诊断结果解释程序*/
void f_expl()
{
    int i,j,k,l,m,tmp;
```

```
int HN_NO=0;          /*记录在规则中起作用的隐含神经元数目*/
int IN_NO;            /*记录在规则中起作用的输入神经元数目*/
int RULE_HN[HN];      /*记录在规则中起作用的隐含神经元*/
int RULE_IN[IN];      /*记录在规则中起作用的输入神经元*/
double c=0,c0;
for(i=0;i<HN;i++) RULE_HN[i]=-1;
/*计算 c0*/
c0=-w_o_h[OUT_NO][HN];
for(i=0;i<HN;i++)
    if(w_o_h[OUT_NO][i]<0&&middle[i]==1)
        c0+=fabs(w_o_h[OUT_NO][i]);
/*形成表 L1,存于数组 RULE_HN 中*/
j=0;
for(i=0;i<HN;i++)
    if(w_o_h[OUT_NO][i]>0&&middle[i]==1)
        {
            RULE_HN[j]=i;
            j++;
        }
/*对表 L1 进行排序*/
if(j!=1)
    {
        for(i=0;i<j-1;i++)
            for(k=i+1;k<j;k++)
                if (w_o_h[OUT_NO][RULE_HN[i]]<w_o_h
                  [OUT_NO][RULE_HN[k]])
                {
                    tmp=RULE_HN[i];
                    RULE_HN[i]=RULE_HN[k];
                    RULE_HN[k]=tmp;
                }
    }
/*形成表 L3 还放在数组 RULE_HN 中*/
```

```
for(i=0;i<j;i++)
  {
        c+=w_o_h[OUT_NO][RULE_HN[i]];
        HN_NO++;
        if(c>c0)  break;
  }
for(k=i+1;k<j;k++)
    RULE_HN[k]=-1;
/*对表 L3 的每一项形成一条规则*/
for(j=0;j<HN_NO;j++)
 {
    c=0;
      for(i=0;i<IN;i++) RULE_IN[i]=-1;
      /*计算 c0*/
    c0=-w_h_i[RULE_HN[j]][IN];
      for(k=0;k<IN;k++)
          if(w_h_i[RULE_HN[j]][k]<0&&symptom[k]==1)
             c0+=fabs(w_h_i[RULE_HN[j]][k]);
      /*形成表 L1,存于数组 RULE_IN 中*/
      l=0;
      for(k=0;k<IN;k++)
        if(w_h_i>0&&symptom[k]==1)
          {
            RULE_IN[l]=k;
            l++;
          }
  /*对表 L1 进行排序*/
  if(l!=1)
       {
         for(k=0;k<l-1;k++)
            for(m=k+1;m<l;m++)
              if (w_h_i[RULE_HN[j]][RULE_IN[k]]<w_h_i
                [RULE_HN[j]][RULE_IN[m]])
```

```
                    {
                      tmp=RULE_IN[k];
                      RULE_IN[k]=RULE_IN[m];
                      RULE_IN[m]=tmp;
                    }
            }
        /* 形成表 L3 还放在数组 RULE_HN 中 */
        IN_NO=0;
        for(k=0;k<1;k++)
            {
                c+=w_h_i[RULE_HN[j]][RULE_IN[k]];
                IN_NO++;
                if(c>c0)  break;
            }
        for(m=IN_NO+1;m<IN;m++)
              RULE_IN[m]=-1;
        /* 写出规则 */
        printf("\n 因为存在征兆\n");
        for(m=0;m<IN_NO;m++)
            {
                printf("          %s",in_fault[RULE_IN[m]]);
                if((m+1)/2*2-m-1==0)    printf("\n");
            }
        printf("导致第%d 隐含层神经元输出为 1",RULE_HN[j]);
    }
    printf("\n 因为\n");
    for(m=0;m<HN_NO;m++)
      {
        printf("      第%2d 个隐含层神经元输出为 1",RULE_HN[m]);
        if((m+1)/2*2-m-1==0)    printf("\n");
      }
    printf("所以存在故障%s",Fault[OUT_NO]);
}
```

```
/* 正向推理子程序 */
void f_diag()
{
    int i;
    char ans;
    OUT_NO=-1;
    printf("\n 请依次输入故障征兆值");
    printf("\n0 表示无此征兆;1 表示有此征兆");
    for(i=0;i<IN;i++)
      {
      printf("\n 系统是否具有征兆:%s?   ",in_fault[i]);
          scanf("%d",&symptom[i]);
      }
    compute();
    for(i=0;i<ON;i++)
      {
          if(trouble[i]==0) continue;
          OUT_NO=i;
          printf("\n 系统具有故障:%s! \n",Fault[i]);
          printf("\n 您是否需要解释(Y/N)");
          getchar();
          scanf("%c",&ans);
          if(ans=='Y'||ans=='y')
             f_expl();
      }
    if(OUT_NO==-1)
       printf("\n 您的液压系统没有故障");
    finish();
}

/* 反向推理子程序 */
void b_diag()
```

```
{
    int i,j,k,reponse;
    int no_i,no_h,no_o;          /* 标识输入、隐含、输出神经元 */
    double real_in;              /* 确定输入 */
    double max_in;               /* 未知的最大输入 */
    double min_in;               /* 未知的最小输入 */
    double max_w;                /* 最大连接权绝对值 */
    for(i=0;i<IN;i++)   strcpy(mark_i[i],"UNKNOWN");
    for(i=0;i<HN;i++)   strcpy(mark_h[i],"UNKNOWN");
    printf("\n 请输入您怀疑存在的故障的编号\n");
    printf("注意：第一个故障编号为 0\nno. =");
    while(1)
       {
            scanf("%d",&no_o);
            if(no_o<ON) break;
            printf("输入越界，重新输入\n");
       }
    while(1)
       {
         /* 判断是否能确定讨论神经元的输出 */
         real_in=max_in=min_in=0;
         for(i=0;i<HN;i++)
             {
                 if(strcmp(mark_h[i],"KNOWN")==0)
                    real_in+=middle[i]*w_o_h[no_o][i];
                 else
                 {
                    if(w_o_h[no_o][i]>0)
                           max_in+=w_o_h[no_o][i];
                    else
                           min_in+=w_o_h[no_o][i];
                  }
             }
```

```
if(real_in+max_in<w_o_h[no_o][HN])
    {
        printf("液压系统没有这种故障\n");
        finish();
    }
if(real_in+min_in>w_o_h[no_o][HN])
    {
        printf("您的猜测正确\n");
        finish();
    }
/*确定与输出层神经元具有最大连接权绝对值的 UNKNOWN
  隐含神经元*/
max_w=0;
for(i=0;i<HN;i++)
  {
    if(strcmp(mark_h[i],"UNKNOWN")==0)
      {
        if(fabs(w_o_h[no_o][i])>max_w)
            {
              max_w=fabs(w_o_h[no_o][i]);
              no_h=i;
            }
      }
  }
/*确定讨论隐含神经元的输出*/
while(1)
{
    /*确定与隐含层神经元具有最大连接权绝对值的 UN-
      KNOWN 输入神经元*/
    max_w=0;
    for(i=0;i<IN;i++)
        {
        if(strcmp(mark_i[i],"UNKNOWN")==0)
```

```
        {
            if(fabs(w_h_i[no_h][i])>max_w)
              {
                max_w=fabs(w_h_i[no_h][i]);
                no_i  i;
              }
        }
    }
/* 对输入神经元进行询问 */
printf("\n 液压系统是否有%21s? \n",in_fault[no_i]);
printf("用户响应[0 没有 1 有 2 为什么]:");
scanf("%d",&reponse);
if(reponse==2)
    {
      printf("\n ***********************************\");
      printf ("\n 因为诊断对象是否具有%21s\n",in_fault
            [no_i]);
      printf("关系到隐含神经元%d 的输出\r",no_h);
      printf ("而隐含神经元%d 的输出直接影响对象是
            否",no_h);
      printf("有故障%s\n",Fault[no_h]);
      printf("因此用户必须对此作出响应.");
      printf("\n ***********************************\");
      printf("\n 用户响应[0 没有 1 有]");
      scanf("%d",&reponse);
    }
strcpy(mark_i[no_i],"KNOWN");
symptom[no_i]=reponse;
/* 计算讨论隐含层神经元的输出 */
real_in=max_in=min_in=0;
for(i=0;i<IN;i++)
    {
      if(strcmp(mark_i[i],"KNOWN")==0)
```

```
                real_in+=symptom[i]*w_h_i[no_h][i];
            else
              {
                if(w_h_i[no_h][i]>0)
                  max_in+=w_h_i[no_h][i];
                else
                  min_in+=w_h_i[no_h][i];
              }
          }
    if(real_in+max_in<w_h_i[no_h][IN])
        {
          middle[no_h]=0;
          strcpy(mark_h[no_h],"KNOWN");
          break;
        }
    if(real_in+min_in>w_h_i[no_h][IN])
        {
          middle[no_h]=0;
          strcpy(mark_h[no_h],"KNOWN");
          break;
        }
  }
/*判断在已有的已知输入下能否确定其它隐含输出*/
for(i=0;i<HN;i++)
  {
    if(strcmp(mark_h[i],"KNOWN")==0) continue;
    real_in=max_in=min_in=0;
    for(j=0;j<IN;j++)
        {
          if(strcmp(mark_i[j],"KNOWN")==0)
            real_in+=symptom[j]*w_h_i[i][j];
          else
            {
```

```
                    if(w_h_i[i][j]>0)
                      max_in+=w_h_i[i][j];
                    else
                      min_in+=w_h_i[i][j];
                  }
              }
              if(real_in+max_in<w_h_i[i][IN])
                  {
                    middle[i]=0;
                    strcpy(mark_h[i],"KNOWN");
                  }
              if(real_in+min_in>w_h_i[i][IN])
                  {
                    middle[i]=1;
                    strcpy(mark_h[i],"KNOWN");
                  }
          }
      }
}

/* 双向推理子程序 */
void d_diag()
{
    int i,j,k,reponse,reponse1,check;
    printf("请用 0,1,2 依次输入故障征兆\n");
    printf("注意：0- - 不存在该征兆  1---存在  2---不详\n");
    for(i=0;i<lN;i++)
       {
          printf("\n 是否存在%21s      响应---",in_fault[i]);
          scanf("%d",&reponse);
          switch(reponse)
           {
              case 0;
```

```
                    strcpy(mark_i[i],"KNOWN");
                    symptom[i]=0;
                    break;
                case 1:
                    strcpy(mark_i[i],"KNOWN");
                    symptom[i]=1;
                    break;
                case 2:
                    strcpy(mark_i[i],"UNKNOWN");
                    symptom[i]=0;
            }
        }
    compute();
    check=0;
    /*检查是否有输出神经元输出为1*/
    for(i=0;i<ON;i++)
        {
            if(trouble[i]==1)
              {
                printf("\n本液压系统具有%20s",Fault[i]);
                check=1;
              }
        }
    if(check==1)  finish();
    else
        {
            /*每次改变一个输入*/
loop:   for(i=0;i<IN;i++)
              {
                if(strcmp(mark_i[i],"KNOWN")==0) continue;
                symptom[i]=1;
                compute();
                for(j=0;j<ON;j++)
```

```
            {
                if(trouble[j]==1)
                 {
                   printf ("\n 本液压系统可能具有%20s\n",Fault
                         [j]);
                   printf("询问：系统是否具有%21s\n 响应[0,
                         1]——",in_fault[i]);
                   scanf("%d",&reponse);
                   if(reponse==0)
                       {
                         printf("\n 我的猜测错了!");
                         strcpy(mark_i[i],"KNOWN");
                         symptom[i]=0;
                         break;
                       }
                   else
                       {
                          printf ("\n 我的猜测正确,很高兴能帮助
                                您!");
                          finish();
                        }
                 }
             }
        }
    /* 每次改变两个权值 */
    for(i=0;i<IN-1;i++)
       for(j=i+1;j<IN;j++)
           {
              if ((strcmp(mark_i[i],"KNOWN")==0)||(strcmp
                (mark_i[j], "KNOWN")==0)) continue;
             symptom[i]=1;
             symptom[j]=1;
             compute();
```

```
for(k=0;k<ON;k++)
  {
     if(trouble[k]--1)
       {
         printf ("\n 本液压系统可能具有%20s",
               Fault[k]);
         printf("\n 询问:系统是否具有%21s\n 响应
               [0,1]----",in_fault[i]);
         scanf("%d",&reponse);
         printf ("\n 询问:系统是否具有%21s\n 响应
               [0,1]——",in_fault[j]);
         scanf("%d",&reponse1);
         if((reponse==1)&&(reponse1==1))
             {
               printf ("\n 我的猜测正确,很高兴能
                     帮助您!");
               finish();
             }
         else if((reponse==0)&&(reponse1==0))
             {
               strcpy(mark_i[i],"KNOWN");
               strcpy(mark_i[j],"KNOWN");
               symptom[i]=0;
               symptom[j]=0;
               break;
             }
         else
            {
              strcpy(mark_i[i],"KNOWN");
              strcpy(mark_i[j],"KNOWN");
            symptom[i]=reponse;
            symptom[j]=reponse1;
            goto loop;
```

```
                    }
                }
            }
        }
    }
    printf("对不起,您所提供的信息不足不能猜测\n");
    finish();
}

/* 诊断子程序 */
void diagnose()
{
    int i,j,l,choice1;
    char know_nm[10];
    FILE *know_fl;
    printf("请输入知识库文件名:");
    scanf("%s",know_nm);
    if((know_fl=fopen(know_nm,"rb"))==NULL)
        {
            printf("不能打开知识库文件\n");
            exit(0);
        }
    /* 读取权值数据 */
    rewind(know_fl);
    for(j=0;j<HN;j++)
        for(i=0;i<IN+1;i++)
            fread(&w_h_i[j][i],8,1,know_fl);
    for(j=0;j<ON;j++)
        for(i=0;i<HN+1;i++)
            fread(&w_o_h[j][i],8,1,know_fl);
    fclose(know_fl);

    /* 开始诊断 */
```

```
    printf("\n\n ********  选择推理方向  ******** \n\n");
    printf("              1          正向推理          \n");
    printf("              2          反向推理          \n");
    printf("              3      混合双向推理        \n\n");
    printf("   您选择.....\n");
    scanf("%d",&choice1);
    switch(choice1)
    {
       case 1:
            f_diag(); break;
       case 2:
            b_diag(); break;
       case 3:
            d_diag();
      }
}

/*    主程序    */
main()
{
    int choice;
    printf("\n ***** 欢迎进入液压系统故障诊断专家系统 ***** \n\n\n");
    printf("                  功能选择\n\n");
    printf("            1       神经网络学习\n");
    printf("            2       神经网络诊断\n");
    printf("            3       退出\n");
    printf("    您选择.....");
    for(;;)
       {
          scanf("%d",&choice);
          if(choice>0&&choice<4) break;
          printf("\n 输入越界,重输");
          sound(7);
```

```
            delay(2000);
          }
    switch(choice)
       {
            case 1:
               learn();
               break;
            case 2:
               diagnose();
               break;
            case 3:
               exit(0);
       }
   return 0;
}

/*********************************************/
/*   文件名:          WJP.H                  */
/*   本程序为 WJP.C 的头文件                 */
/*   定义有关的神经网络参数和学习诊断样本    */
/*   各种全局变量也在此定义                  */
/*********************************************/

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<dos.h>
#include<graphics.h>
#include<string.h>
#include<conio.h>
#include<time.h>
```

```
#define IN 18                          /* 输入层神经元数目 */
#define HN 15                          /* 隐含层神经元数目 */
#define ON 17                          /* 输出层神经元数目 */
#define SIZE 17                        /* 学习样本数目 */
#define EITA 0.5                       /* 学习步长 */
#define ZITA 0.0001                    /* 权值收敛因子 */
#define BETA 0.1                       /* 误差收敛因子 */
#define NAMTA 10                       /* Sigmoid 函数修正因子 */

static float w_h_i[HN][IN+1];          /* 输入层到隐含层的权值 */
static float w_o_h[ON][HN+1];          /* 隐含层到输出层的权值 */
static float o_h[SIZE][HN];            /* 隐含层的输出 */
static float o_o[SIZE][ON];            /* 输出层的输出 */
float Sum_Err[5000];                   /* 神经网络的总体误差 */
float Sig_Err[SIZE];                   /* 神经网络的单样本误差 */

static int OUT_NO;                     /* 故障输出号 */
static int symptom[IN];                /* 征兆 */
static int trouble[ON];                /* 故障 */
static int middle[HN];                 /* 隐含层的值 */
static char mark_i[IN][10];            /* 标识输入神经元 */
static char mark_h[HN][10];            /* 标识隐含神经元 */

/*   故障征兆中文描述   */
static char in_fault[IN][21]=
   {"                不动作","                动作慢",
    "          反应迟钝、无力","          保位无力沉降快",
    "                  抖动","                不能微动",
    "            操纵阀失灵","            溢油噪声高",
    "          油管车体振动","          液压泵噪声高",
    "          液压马达噪声高","                异响",
    "  液压缸、阀门系统泄漏","  液压缸马达抽油封泄漏",
    "              高压管路漏","              低压管路漏",
```

```
    "                油温过高","              磨损粉末过多"};

/*   故障原因中文描述   */
static char Fault[ON][20]=
   {"     系统排气不彻底","              管道连接松劲",
    "       管件破损毛刺","     液压油量过多或过少",
    "     液压油粘度不当","              液压油老化",
    "     液压油渗有水分","          液压油混入异物",
    "     液压油混入空气","     封闭式油箱压力不当",
    "          冷却器故障","              滤油器故障",
    "          液压泵故障","              溢流阀故障",
    "          换向阀故障","     液压缸液压马达故障",
    "            气温异常"};

/*   学习样本   */
struct samples
   {
     float in_sign[IN];        /*输入信号*/
     float tch_sign[ON];       /*教师信号*/
   } samp[SIZE]=
   {{1,0,0,0,1,1,0,0,1,1,1,0,0,0,0,0,0,0,  1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,1,0,0,  0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {1,0,0,1,0,0,0,1,0,1,0,0,0,0,1,1,0,0,  0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {1,1,0,0,1,0,0,1,0,1,0,0,1,1,0,1,1,0,  0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},
    {1,1,0,1,1,1,0,0,0,1,0,0,0,0,0,1,1,1,  0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,  0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,1,0,0,1,0,0,1,1,0,0,0,1,  0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0},
    {1,1,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0},
    {0,0,0,0,1,1,0,1,1,1,1,0,0,0,0,0,1,0,  0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0},
    {1,1,0,0,1,1,0,1,1,1,0,0,1,1,0,0,0,0,  0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,  0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
    {1,1,0,0,1,1,0,1,0,1,0,0,1,0,0,0,0,1,  0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0},
    {1,1,1,0,1,1,0,0,0,1,0,1,0,0,0,0,0,1,  0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0},
```

```
    {1,1,0,1,1,0,1,1,1,0,0,0,0,1,0,0,1,0,  0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0},
    {1,1,0,1,1,1,1,0,1,0,1,1,1,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0},
    {1,1,1,1,1,0,0,0,1,1,0,1,1,0,0,0,1,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0},
    {1,1,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1};

    /* ************************************** */
    /*    文件名          BPXI..C             */
    /*    本程序为三层 BP 神经网络学习程序    */
    /* ************************************** */

/* sigmoid 函数 */
double sigmoid(flag)
double flag;
{
    double ret;
    flag=-NAMTA*flag;
    if(flag>5)  return 0.99;
    if(flag<-5) return 0.01;
    ret=1+exp(-flag);
    ret=1/ret;
    return ret;
}

/* 画图子程序 */
void draw_Err(STEP)
int STEP;
{
    int i,j;
    int gdriver=DETECT,gmode;
    int x_step,y_step;              /* x 轴和 y 轴的幅度 */
    int x_spot,y_spot;              /* 误差点的坐标 */
    double max_err=0;
```

```
char x_string[8],y_string[8];
initgraph(&gdriver,&gmode,"e:\\borlandc\\bgi");
clearviewport();
for(i=1;i<STEP;i++)
    if(Sum_Err[i]>max_err)
      max_err=Sum_Err[i];
for(i=1;i<7;i++)
    if(STEP<=pow10(i))          break;
if(STEP<2*pow10(i-1))           j=2;
else if(STEP<4*pow10(i-1))      j=4;
else if(STEP<8*pow10(i-1))      j=8;
else                            j=10;
x_step=j*pow10(i-1);
for(i=0;i<4;i++)
    if(max_err<=pow10(i))           break;
if(max_err<2*pow10(i-1))            j=2;
else if(max_err<4*pow10(i-1))       j=4;
else if(max_err<8*pow10(i-1))       j=8;
else                                j=10;
y_step=j*pow10(i-1);
/*画坐标轴*/
moveto(20,440);   lineto(600,440);
moveto(60,10);    lineto(60,470);
moveto(56,15);    lineto(60,5);     lineto(64,15);
moveto(595,436);  lineto(605,440);  lineto(595,444);
outtextxy(70,20,"Error");
outtextxy(590,450,"Step");
outtextxy(40,450,"0");
for(i=1;i<=4;i++)
    {
        moveto(60,40+(i-1)*100);
        lineto(65,40+(i-1)*100);
        gcvt(y_step*(5-i)/4,8,y_string);
```

```
            outtextxy(35,40+(i-1)*100,y_string);
        }
    for(i=1;i<=5;i++)
        {
            moveto(60-i*100,440);
            lineto(60+i*100,435);
            itoa(x_step*i/5,x_string,10);
            outtextxy(40+(i)*100,450,x_string);
        }
    moveto(1.0/x_step*500+60,480-(Sum_Err[1]/y_step*400+40));
    for(i=1;i<STEP;i++)
        {
            x_spot=i*1.0/x_step*500+60;
            y_spot=480-(Sum_Err[i]/y_step*400+40);
            lineto(x_spot,y_spot);
        }
     getch();
     clearviewport();
     closegraph();
}

/* 权值初始化函数 */
void initw()
{
    int i,j,l;
    randomize();
    for(i=0;i<=IN;i++)
        for(j=0;j<HN;j++)
            {
               w_h_i[j][i]=random(100)*1.0/100;
               if(random(2)==0) w_h_i[j][i]=-w_h_i[j][i];
            }
    for(j=0;j<=HN;j++)
```

```
        for(l=0;l<ON;l++)
        {
                w_o_h[l][j]=random(100)*1.0/100;
                if(random(2)==0) w_o_h[l][j]=-w_o_h[l][j];
        }
}

/*    训练子程序    */
void learn()
{
   int i,j,k,l,mark,step=0;
   char file_nm[10];
   static double v_h_i[HN][IN+1];   /*前一时刻的隐含层到输入层的
                                      权值*/
   static double v_o_h[ON][HN+1]; /*前一时刻的输出层到隐含层的
                                      权值*/
   double o_o_old[SIZE][ON];        /*前一时刻输出层的输出*/
   double net;
   static double delta_h[HN];       /*隐含层的训练误差*/
   static double delta_o[ON];       /*输出层的训练误差*/

   FILE *data_fl;
   initw();
   printf("请输入保存权值数据的数据库文件名:");
   scanf("%s",file_nm);
   if((data_fl=fopen(file_nm,"wb"))==NULL)
        {
                printf("不能打开数据库文件\n");
                exit(0);
        }
   printf("\n");
   for(i=1;i<=4;i++)
      printf("    步数        总误差    ");
```

```
printf("\n");
while(1)
    {
      /* 保存各项数据 */
      for(i=0;i<SIZE;i++)
         for(j=0;j<ON;j++)
             o_o_old[i][j]=o_o[i][j];
      /* 对每一个学习样本进行如下操作 */
      for(i=0;i<SIZE;i++)
       {
          for(l=0;l<=IN;l++)
              for(j=0;j<HN;j++)
                v_h_i[j][l]=w_h_i[j][l];
          for(j=0;j<=HN;j++)
              for(l=0;l<ON;l++)
                    v_o_h[l][j]=w_o_h[l][j];
          /* 计算隐含层的输出 */
          for(j=0;j<HN;j++)
              {
                net=0;
                for(l=0;l<IN;l++)
                      net=net+samp[i].in_sign[l]*v_h_i[j][l];
                net=net+v_h_i[j][IN];
                o_h[i][j]=sigmoid(net);
              }
          /* 计算输出层的输出 */
          for(j=0;j<ON;j++)
             {
                net=0;
                for(l=0;l<HN;l++)
                      net=net+o_h[i][l]*v_o_h[j][l];
                net=net+v_o_h[j][HN];
                o_o[i][j]=sigmoid(net);
```

```
    }
/*计算训练误差*/
/*计算输出层的训练误差*/
for(j=0;j<ON;j++)
      delta_o[j]=NAMTA*o_o[i][j]*(1-o_o[i][j])*
          (samp[i].tch_sign[j]-o_o[i][j]);
/*计算隐含层的训练误差*/
for(j=0;j<HN;j++)
      {
         delta_h[j]=0;
         for(l=0;l<ON;l++)
           delta_h[j]+=delta_o[l]*v_o_h[l][j];
         delta_h[j]*=NAMTA*o_h[i][j]*(1-o_h[i]
                 *(1-o_h[i][j]);
      }
/*修改权值*/
/*修改隐含层到输出层的权值*/
for(j=0;j<ON;j++)
     {
       for(l=0;l<HN;l++)
           w_o_h[j][l]=v_o_h[j][l]+EITA*delta_o[j]
                  *o_h[i][l];
       w_o_h[j][HN]=v_o_h[j][HN]+EITA
                         *delta_o[j];
     }
/*修改输入层到隐含层的权值*/
for(j=0;j<HN;j++)
     {
       for(l=0;l<IN;l++)
          w_h_i[j][l]=v_h_i[j][l]+EITA*delta_h[j]*
                 samp[i].in_sign[l];
       w_h_i[j][IN]=v_h_i[j][IN]+EITA*delta_h[j];
     }
```

```
}
/*计算输出总误差*/
step++;
for(i=0;i<SIZE;i++)
   {
        Sig_Err[i]=0;
        for(j=0;j<ON;j++)
            Sig_Err[i]+=pow((samp[i].tch_sign[j]-o_o[i]
                [j]),2);
        Sum_Err[step]+=.5*Sig_Err[i];
     }
 if(step-step/10*10==0)
        printf("%6d%14.8f",step,Sum_Err[step]);
/*判定权值是否已经收敛*/
mark=0;
for(i=0;i<SIZE;i++)
   for(j=0;j<ON;j++)
       if(fabs(o_o[i][j]-o_o_old[i][j])>ZITA)
         {
            mark=1;
            break;
         }
if(Sum_Err[step]<0.15) break;
if(mark==1)  continue;
/*判断是否陷入局部最小点*/
for(i=0;i<SIZE;i++)
   for(j=0;j<ON;j++)
       if(fabs(o_o[i][j]-samp[i].tch_sign[j])>BETA)
         {
            printf(" ***************************\n");
            printf(" *很抱歉:                    *\n");
            printf(" *      网络收敛到局部最小点.*\n");
            printf(" *   请选择样本和权值初值重做.*\n");
```

```
                    printf(" ************************** \n");
                    getch();
                    exit(0);
                }
            break;
        }
        /*保存权值数据*/
        for(j=0;j<HIN;j++)
            for(i=0;i<IN+1;i--)
                fwrite(&w_h_i[j][i],8,1,data_fl);
        for(j=0;j<ON;j++)
            for(i=0;i<HIN+1;i++)
                fwrite(&w_o_h[j][i],8,1,data_fl);
        fclose(data_fl);
        getch();
        draw_Err(step);
        printf("\n                    权值及阈值数据");
        for(i=0;i<HIN;i++)
            {
                printf("\n 第%2d 个隐含神经元的权值和阈值为\n",i+1);
                for(j=0;j<=IN;j++)
                    printf("%13.7f",w_h_i[i][j]);
            }
        for(i=0;i<ON;i++)
            {
                printf("\n 第%2d 个输出神经元的权值和阈值为\n",i+1);
                for(j=0;j<=HN;j++)
                    printf("%13.7f",w_o_h[i][j]);
            }
        getch();
}

/ ********************************************** /
```

```
/*    文件名:      ANN_RULE.C                */
/*    作用:从神经网络连接权中产生规则          */
/*****************************************/

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<dos.h>
#include<conio.h>
#include<string.h>

#define IN 3                          /*定义输入神经元数*/
#define HN 2                          /*定义隐含神经元数*/
#define ON 2                          /*定义输出神经元数*/
#define E_rule 0.005                  /*定义规则误差限*/

static double w_h_i[HN][IN+1];        /*输入层到隐含层的权值*/
static double w_o_h[ON][HN+1];        /*隐含层到输出层的权值*/
static double o_h[HN];                /*隐含层的输出*/
static int rule_no=1;                 /*规则号*/
int symptom[IN];                      /*征兆的各种整型输入值*/
double d_fault[ON];                   /*故障的各种输出值*/
int i_fault[ON];                      /*故障的整型输出值*/
/*   故障征兆的中文描述   */
static char in_fault[IN][10]={"征兆1","征兆2","征兆3"};
/*   故障原因的中文描述   */
static char Fault[ON][10]={"故障1","故障2"};

/*sigmoid函数*/
double sigmoid(flag)
double flag;
{
```

```
    double ret;
    ret = 1+exp(-flag);
    ret = 1/ret;
    return ret;
}

/* 计算输出 */
void compute()
{
    int j,l;
    double net;
    /* 计算隐含层的输出 */
    for(j = 0;j<HN;j++)
        {
          net = 0;
          for(l=0;l<IN;l++)
              net=net+symptom[l] * w_h_i[j][l];
          net=net+w_h_i[j][IN];
          o_h[j]=sigmoid(net);
        }
    /* 计算输出层的输出 */
    for(j=0;j<ON;j++)
        {
          net=0;
          for(l=0;l<HN;l++)
              net=net+o_h[l] * w_o_h[j][l];
          net=net+w_o_h[j][HN];
          d_fault[j]= sigmoid(net);
          i_fault[j]=0;
          if(d_fault[j]>0.5) i_fault[j] = 1;
        }
}
```

```
/*判断是否产生规则*/
void rule()
{
   int i;
   double Error=0;
   for(i=0;i<ON;i++)
       Error+=0.5*pow((d_fault[i]-i_fault[i]),2);
   if(Error<E_rule)
     {
       printf("\n 规则%d：如果诊断对象具有",rule_no);
       for(i=0;i<IN;i++)
            if(symptom[i]==1)
              printf("%s",in_fault[i]);
       printf("  则诊断对象具有");
       for(i=0;i<ON;i++)
            if(i_fault[i]==1)
              printf("%s",Fault[i]);
       rule_no++;
     }
}

/*主函数*/
main()
{
   FILE *know_fl;
   int i,Max=1,j,l,k,m=1;
   char know_nm[10];
   printf("请输入知识库文件名:");
   scanf("%s",know_nm);
   if((know_fl=fopen(know_nm,"rb"))==NULL)
      {
          printf("不能打开知识库文件\n");
          exit(0);
```

```
    }
/* 读取权值文件 */
rewind(know_fl);
for(j=0;j<HN;j++)
    for(i=0;i<=IN;i++)
        fread(&w_h_i[j][i],8,1,know_fl);
for(j=0;j<ON;j++)
    for(i=0;i<=HN;i++)
        fread(&w_o_h[j][i],8,1,know_fl);
fclose(know_fl);
for(i=0;i<IN;i++) Max=Max*2;
  for(i=0;i<Max;i++)
    {
        l=i;
        /*产生输入*/
        for(j=IN-1;j>=0;j--)
          {
          m=1;
          for(k=0;k<j;k++) m*=2;
          symptom[j]=0;
          if(l>=m)
              {
                symptom[j]=1;
                l=l-m;
              }
          }
        compute();
        rule();
    }
return 0;
}
```

第五章　模糊神经网络故障诊断专家系统

5.1　概　　述

在日常生活中，我们常常碰到一些事物具有亦此亦彼特性的情形，它们不能简单地表达为非此即彼，此时我们称他们具有模糊性. 例如，如果一个人头上一根头发也没有，那么他无疑是一个秃头，如果他有满头的头发，那么他无疑不是秃头，然而他有半头头发是不是秃头呢？再多一点或少一点是否算秃头呢？问题的回答变得越来越困难. 可见，一个人是否秃头没有一个明确的界限. 从秃到不秃要经历一个从量变到质变的连续过渡过程，这种现象就称为中间过渡性. 由这种中间过渡性形成的划分上的不确定性就是模糊性.

自从美国控制论学者扎德(L. A. Zadeh)于 1965 年提出模糊集合论以来创立的模糊数学就是用来研究这种模糊性的. 由于它在处理难以精确化的大系统、复杂系统中显得简洁而有力，从而在某种程度上弥补了经典数学上的不足，而受到广大科研、工程技术人员的欢迎.

神经网络与模糊逻辑的结合始于 1988 年，它们的结合可以说是历史发展的必然，因为神经网络的优点是具有很强的自学习功能，而模糊逻辑则长于处理显式逻辑问题，它们的结合必将更有力地促进它们的发展.

所谓模糊神经网络(Fuzzy Neural Network，FNN)就是神经网络与模糊数学相结合的产物，它是一种新型的神经网络，既具有学习、联想、自适应性，又能进行模糊推理.

模糊神经网络故障诊断专家系统(FNN-FD-ES)是将神经网络技术引入到模糊故障诊断专家系统中去的结果，也可以说是将

模糊逻辑与神经网络直接应用于故障诊断专家系统中去的结果.一个 FNN-FD-ES 的推理结构通常可表示为下图：

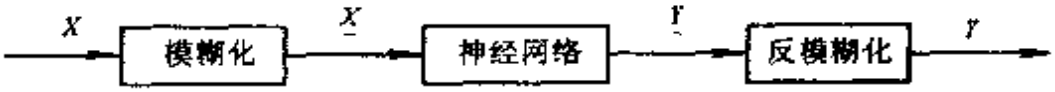


图 5.1　FNN-FD-ES 结构示意图

图中 X 为输入向量，$\tilde{X}$ 是输入模糊向量(也称量化输入)，$\tilde{Y}$ 是输出模糊向量(量化输出)，Y 为输出向量.

5.2　模糊逻辑理论

5.2.1　隶属函数

在我们熟悉的经典集合(相对于模糊集合而言)中，假设 u 为论域 U 中的一个元，A 为 U 的子集 $A \subset U$，那么 u 与 A 的关系可以用特征函数 $C_A(u)$ 来刻划：

$$C_A(u) = \begin{cases} 1, & \text{如果 } u \in A \\ 0, & \text{如果 } u \notin A \end{cases}$$

即 u 要么属于 A 要么不属于 A，两者必取其一，非此即彼. 经典集合中的特征函数只允许取{0,1}两个值，与二值逻辑相对应，它不能表达现实中存在的“亦此亦彼”的现象，因此有必要将特征函数推广到[0,1]闭区间，这就形成了模糊数学中的隶属函数 $A(u)$.

定义 1　设在论域 U 上给定的一个映射

$$A : u \to [0,1] \tag{5.2.1}$$

$$u \to A(u) \tag{5.2.2}$$

则称 A 为 U 上的模糊(Fuzzy)集，$A(u)$ 称为 A 的隶属函数(或称为 u 对 A 的隶属度).

隶属函数的表示方法大致有三种：

一般情形下(如下面的“年老”与“年青”隶属函数即属于此种)表为

$$A = \{(u, A(u)) \mid u \in U\} \tag{5.2.3}$$

如果 U 是有限集或可数集，可表示为

$$A = \sum A(u_i)/u_i \tag{5.2.4}$$

或表示为向量(称为 F 向量)

$$A = (A(u_1), A(u_2), \cdots, A(u_n)) \tag{5.2.5}$$

如果 U 是无限集，可表示为

$$A = \int A(u)/u \tag{5.2.6}$$

例如：扎德曾经给出“年青”和“年老”的隶属函数. 他将论域 U 取为[0,100]，集合 A 和 B 分别表示“年老”和“年青”两个模糊集，则有

$$A(u) = \begin{cases} 0, & 0 \leqslant u \leqslant 50 \\ \left[1 + \left(\dfrac{u-50}{5}\right)^{-2}\right]^{-1}, & 50 < u \leqslant 100 \end{cases}$$

$$B(u) = \begin{cases} 1, & 0 \leqslant u \leqslant 25 \\ \left[1 + \left(\dfrac{u-25}{5}\right)^{2}\right]^{-1}, & 25 < u \leqslant 100 \end{cases}$$

这样，$A(70)=0.94$，说明年龄为 70 时属于“年老”的隶属程度达到 94%；$B(30)=0.5$，说明 30 岁属于“年青”的隶属程度为 50%.

在模糊理论的实际应用中，首先就必须建立 F 集的隶属函数，可见隶属函数的建立是比较重要的. 在故障诊断中常用的确定隶属函数的方法有：专家统计法、模糊统计试验法、加权统计法、二元对比排序法和动态信号分析法等.

5.2.2 截集

截集的概念描述了模糊集与普通集的之间的转换关系.

定义 2 设 $A \in F(U)$，对任意 $\lambda \in [0,1]$，集合

$$A_\lambda = \{u \mid u \in U, A(u) \geqslant \lambda\}$$

称为集合 A 的 λ 截集，λ 称为阈值(或置信水平).

由定义可知，集合 A 为模糊集，A_λ 为普通集，通过阈值 λ 实现了模糊集到普通集的转换. 下面举一个例子对这种集合转换予以说明.

例：某公司有五名职工，分别编号为 $S1, S2, \cdots, S5$. 年龄如表 5.1 所示，根据扎德定义的“青年”隶属度函数，可以得到各人的“青年”隶属度，也列在表中. 假设现在该公司召开一个只需有 3 人参加的小型座谈会，即选取一个 λ，使 A_λ 中有 3 个元素，这时对于任意一名职工 Si 来说，就不存在模糊可言：他要么参加，要么不参加，不可能说以多少程度参加，此时可取 $\lambda=0.6098$，有 $A_{0.6098}=\{1,1,1,0,0\}$；如果召开一个 4 人参加的座谈会，则此时必须选取 $\lambda=0.2$，这时，$A_{0.2}=\{1,1,1,1,0\}$.

表 5.1　普通集与截集的关系

编　号	年　龄	$A(u)$	$A_{0.6098}(u)$	$A_{0.2}(u)$
$S1$	20	1	1	1
$S2$	27	0.8631	1	1
$S3$	29	0.6098	1	1
$S4$	35	0.2000	0	1
$S5$	40	0.1000	0	0

5.2.3　模糊算子、模糊集合的运算

与普通集相似，模糊集也可以定义各种运算. 两个具有相同论域模糊集合之间的运算，实质上就是逐点对其隶属度进行相应的运算.

定义 3　设 $A, B\in F(U)$，对 $\forall u\in U$，规定

$$(A\cup B)(u)=A(u)\ \vee^{*} B(u)$$

$$(A\cap B)(u)=A(u)\ \wedge^{*} B(u)$$

式中 $\vee^{*}$ 和 $\wedge^{*}$ 是[0,1]中的二元运算，简称为模糊算子，用得最多的模糊算子是 $\vee$，$\wedge$ 算子，以后除非特别说明，否则都使用这对算子. 设 $a=A(u), b=B(u)$，则这对算子的定义为

$$a\vee b=\max(a,b),\ a\wedge b=\min(a,b)$$

定义 4　设 $A, B\in F(U)$，则

A 与 B 的并集中的元为

$$(A \cup B)(u) = \max(A(u), B(u)) = A(u) \vee B(u) \tag{5.2.7}$$

A 与 B 的交集中的元为

$$(A \cap B)(u) = \min(A(u), B(u)) = A(u) \wedge B(u) \tag{5.2.8}$$

A 的补集中的元为

$$A^C(u) = 1 - A(u) \tag{5.2.9}$$

定义 4 可以用图 5.2 来表示.

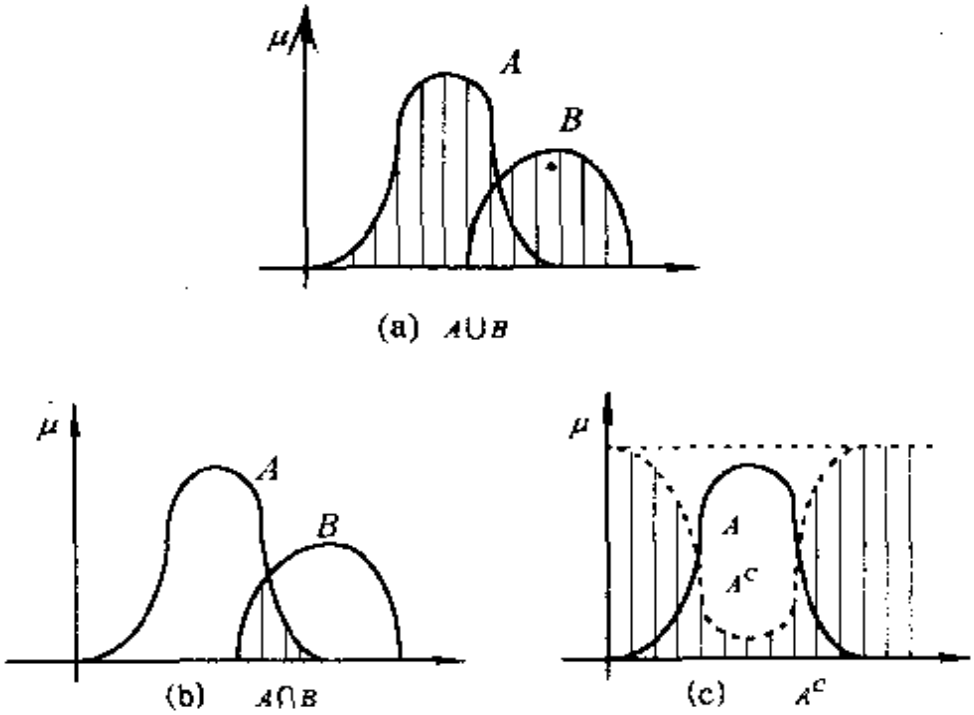


图 5.2 模糊集合运算

模糊集合的并、交、补运算中,除互补律外满足与普通集相同的其它基本性质,即模糊集不存在:

$$A \cup A^C = U \quad \text{和} \quad A \cap A^C = \Phi$$

从图 5.2(c)中可以充分看出这一点.

5.2.4 F 集的贴近度

贴近度是用来衡量两个 F 集 A 和 B 的接近程度,用 $N(A,B)$

表示.贴近度数值越大表明这两者越接近.

定义 5 设 $A,B,C\in F(U)$,若映射

$$N:\quad F(U)\times F(U)\to[0,1]$$

满足条件:

1) $N(A,B)=N(B,A)$;

2) $N(A,A)=1,N(U,\Phi)=0$;

3) 若 $A\subseteq B\subseteq C$,则 $N(A,C)\leqslant N(A,B)\wedge N(B,C)$.

则称 $N(A,B)$为 F 集 A 与 B 的贴近度.N 称为 $F(U)$上的贴近度函数.

考虑论域 U 为有限集的情况(在故障诊断领域中论域都为有限集),常用的有以下三种贴近度.

(1) 海明贴近度

若 $U=\{u_1,u_2,\cdots,u_n\}$,则

$$N(A,B)\triangleq 1-\frac{1}{n}\sum_{i=1}^{n}|A(u_i)-B(u_i)| \tag{5.2.10}$$

(2) 欧几里德贴近度

若 $U=\{u_1,u_2,\cdots,u_n\}$,则

$$N(A,B)\triangleq 1-\frac{1}{\sqrt{n}}\left(\sum_{i=1}^{n}(A(u_i)-B(u_i))^2\right)^{1/2} \tag{5.2.11}$$

(3) 格贴近度

定义 F 集 A,B 的内积

$$A\circ B=\bigvee_{u\in U}(A(u)\wedge B(u)) \tag{5.2.12}$$

若 $U=\{u_1,u_2,\cdots,u_n\}$,则

$$N(A,B)=(A\circ B)\wedge(A^C\circ B^C) \tag{5.2.13}$$

5.2.5 F 关系

定义 6 设 R 是 $U\times V$ 上的一个 F 子集(简称 F 子集),它的隶属函数

$$R:U\times V\to[0,1]$$

$$(u,v) \to R(u,v)$$

确定了 U 中元素 u 与 V 中元素 v 的关系程度，则称 R 为 U 到 V 的一个 F 关系，记

$$U \xrightarrow{R} V$$

任何一个 F 关系都可以通过一个 F 矩阵来确定，任何一个 F 矩阵反过来也对应着一个 F 关系. 所谓 F 矩阵就是这么一种矩阵，它的元素都在区间[0,1]内. 通过这个 F 矩阵，可以实现由 U 上的 F 集 A，经变化得到 V 上的 F 集 B：

$$B = A \circ R \tag{5.2.14}$$

式中 $\circ$ 为算子，例如可这样定义。

$$(A \circ R)(v) \triangleq \bigvee_{u \in U} (A(u) \wedge B(u)),\ v \in V$$

例如设 $U=\{u_1,u_2,u_3\}$，$V=\{v_1,v_2,v_3,v_4\}$

$$R = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \quad R \in F(U \times V)$$

$$A = \frac{0.5}{u_1} + \frac{0.1}{u_2} + \frac{0.3}{u_3}$$

则

$$\begin{aligned} B &= (0.5,0.1,0.3) \circ \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \\ &= (0.5,0.3,0.5,0.1) \\ &= \frac{0.5}{v_1} + \frac{0.3}{v_2} + \frac{0.5}{v_3} + \frac{0.1}{v_4} \end{aligned}$$

5.3 模糊故障诊断

在设备的运转过程中，设备的运行状况由无故障运行到带故障运行是一个渐变过程，在此中间它既不是完全的“完好”，也不是完全的“故障”，而是处于一个中间状态；设备所表现出来的症状也是

如此，因而，设备的各种征兆值和各种故障值应该是一种模糊值.

5.3.1 模糊故障诊断原理

模糊诊断的方法是通过某些征兆的隶属度来求出各种故障原因的隶属度，设诊断对象可能表现的征兆有 m 种，表之为 $x_1, x_2, \cdots, x_m$；可能出现的故障原因有 n 种，表之为 $y_1, y_2, \cdots, y_n$. 故障征兆模糊向量为

$$\underset{\sim}{X} = (\mu_{x_1}, \mu_{x_2}, \cdots, \mu_{x_m})$$

$\mu_{x_i}(i=1,2,\cdots,m)$是对象具有症状 x_i 的隶属度.

故障原因模糊向量

$$\underset{\sim}{Y} = (\mu_{y_1}, \mu_{y_2}, \cdots, \mu_{y_n})$$

$\mu_{y_j}(j=1,2,\cdots,n)$是对象具有故障 y_j 的隶属度，则 $\underset{\sim}{Y}$ 与 $\underset{\sim}{X}$ 具有模糊关系：

$$\underset{\sim}{Y} = \underset{\sim}{X} \circ \underset{\sim}{R} \tag{5.3.1}$$

这就是故障原因与征兆之间的模糊关系方程. 式中“$\circ$”是模糊算子，$\underset{\sim}{R}$ 是体现诊断专家经验知识的模糊诊断矩阵：

$$\underset{\sim}{R} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix} = (r_{ij})_{m\times n} \tag{5.3.2}$$

其中，$0 \leqslant r_{ij} \leqslant 1$，$1 \leqslant i \leqslant m$，$1 \leqslant j \leqslant n$.

模糊诊断矩阵描述了故障征兆与故障原因之间的关系.

5.3.2 模糊算子

设诊断对象故障征兆模糊向量为 $\underset{\sim}{X} = (x_1, x_2, \cdots, x_m)$，故障原因模糊向量为 $\underset{\sim}{Y} = (y_1, y_2, \cdots, y_n)$，将式(5.3.1)展开有

$$\underset{\sim}{Y} = \underset{\sim}{X} \circ \underset{\sim}{R} = (x_1, x_2, \cdots, x_m) \circ \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix}$$

$$= (y_1, y_2, \cdots, y_n)$$

为求出 $\underset{\sim}{Y}$ 有必要将算子“。”具体化，设通用的算子模型用记号 $M(\overset{\bullet}{*}, \overset{+}{*})$ 表示. 其中 M 表示模型；括号内的符合表示合成运算的方式，$\overset{\bullet}{*}$ 表示广义模糊“与”运算，$\overset{+}{*}$ 表示广义“或”运算. 这样我们可得到通用模糊诊断公式：

$$y_j = \overset{m}{\underset{i=1}{\overset{+}{*}}} (x_i \overset{\bullet}{*} r_{ij}), \quad j = 1, 2, \cdots, n \tag{5.3.3}$$

下面对一些具体的算法模型进行讨论.

模型 Ⅰ：$M(\wedge, \vee)$

在这种模型中采用 $\wedge$（取小）来代替“与”运算，采用 $\vee$（取大）来代替“或”运算，此时(5.3.3)式成为

$$y_j = \bigvee_{i=1}^{m} (x_i \wedge r_{ij}), \quad j = 1, 2, \cdots, n \tag{5.3.4}$$

$M(\wedge, \vee)$ 算法的优点是计算起来非常方便、简单，无论 $\underset{\sim}{X}$ 和 $\underset{\sim}{R}$ 是如何产生的，这种算法都是适用的. 缺点是 $(\wedge, \vee)$ 运算太粗糙，诊断中往往丢失大量有价值的信息，所得的诊断结果常常不能令人满意. 如果 $\underset{\sim}{R}$ 和 $\underset{\sim}{X}$ 是稀疏型矩阵，即它们的隶属函数衰减都很快的话，那么选用 $M(\wedge, \vee)$ 能得到较满意的结果.

模型 Ⅱ：$M(\bullet, \vee)$

在这种模型中采用乘法 $\bullet$ 来代替“与”运算，采用 $\vee$（取大）来代替“或”运算，此时(5.3.3)式成为

$$y_j = \bigvee_{i=1}^{m} (x_i r_{ij}), \quad j = 1, 2, \cdots, n \tag{5.3.5}$$

在这种模型中相乘运算 $(x_i r_{ij})$ 不会丢失任何信息；但取大运算 $\bigvee_{i=1}^{m}$ 仍将丢失大量有用的信息. 与模型 Ⅰ 一样，仍是既最大限度地突出主要因素，又最大限度地突出单因素评判的隶属度，两者缺一不可. 模型 Ⅱ 虽然也将丢失大量的信息，但能较好地反映单因素评判的结果和因素的重要程度. 在这方面，比模型 Ⅰ 有所改进.

模型 Ⅲ：$M(\wedge, \oplus)$

在这种模型中采用 $\wedge$（取小）来代替“与”运算，采用 $\oplus$ 来代替

“或”运算，算子⊕定义为有上限 1 的求和，即满足：

$$x \oplus y = \min(1, x + y)$$

此时式(5.3.3)成为

$$y_j = \bigcup_{i=1}^{m} (x_i \wedge r_{ij}), \quad j = 1,2,\cdots,n \tag{5.3.6}$$

记号 $\bigcup_{i=1}^{m}$ 为对 m 个数在⊕运算下的求和. 因此上式又可改写为

$$y_j = \min\left\{1, \sum_{i=1}^{m} (x_i \wedge r_{ij})\right\} = 1 \wedge \sum_{i=1}^{m} (x_i \wedge r_{ij}) \quad j = 1,2,\cdots,n \tag{5.3.7}$$

在这种模式中存在两个缺陷：1) 该模式在进行取小运算($x_i \wedge r_{ij}$)时，仍将会丢失大量有用的信息，以致诊断结果不会令人满意；2) 当 x_i 和 r_{ij} 取值较大时，相应的 y_i 值均可能等于上限 1；当 x_i 和 r_{ij} 取值均较小时，相应的 y_j 均可能等于各 x_i 的和. 这样的诊断结果都是毫无意义的.

模型Ⅳ：$M(\bullet, \oplus)$

该模型用 $\bullet$ 代替“与”运算，用⊕取代“或”运算. 此时式(5.3.3)成为

$$y_j = \bigcup_{i=1}^{m} x_i r_{ij}, \quad j = 1,2,\cdots,n \tag{5.3.8}$$

类似于模型Ⅲ，上式可以改写为

$$y_j = \min(1, \sum_{i=1}^{n} x_i r_{ij}) = 1 \wedge \sum_{i=1}^{n} x_i r_{ij}, \quad j = 1,2,\cdots,n \tag{5.3.9}$$

当 $x_i (i = 1,2,\cdots,m)$ 具有归一性，即具有 $\sum_{i=1}^{m} x_i = 1$ 时，由于 $0 \leqslant r_{ij} \leqslant 1$，因此我们可以得到 $\bigcup_{i=1}^{m} x_i r_{ij} = \sum_{i=1}^{m} x_i r_{ij} \leqslant 1$，这时，算子⊕便脱化成一般的实数相加. 于是，该模型可改写为 $M(\bullet, +)$，即用 $\bullet$ 作为广义“与”运算，用 $+$ 作为广义“或”运算，这是普通的矩阵运算. 这时有

$$y_j = \sum_{i=1}^{m} x_i r_{ij}, \quad j = 1,2,\cdots,n \tag{5.3.10}$$

式中

$$\sum_{i=1}^{m} x_i = 1 \tag{5.3.11}$$

该模型不仅考虑了所有因素的影响，而且保留了单因素评判的全部信息. 在运算时，并不对 x_i 和 $r_{ij}(i=1,2,\cdots,m,\ j=1,2,\cdots,n)$施加上限限制，只是 x_i 必须归一化处理. 这是该模型的显著特点，也是它的优点.

上面的四种模型各有特点，在实际应用时，一般可先考虑 $M(\wedge,\vee)$和 $M(\bullet,\oplus)$，若所得数据偏小，则改用 $M(\wedge,\oplus)$，若所得数据偏大，则改用 $M(\bullet,\vee)$.

5.3.3 诊断原则

经过模糊运算 $Y=X\circ R$ 后，我们得到故障原因模糊向量 $Y=(y_1,y_2,\cdots,y_n)$，现在讨论使用何种原则来确定诊断对象所具有的故障.

(1) 最大隶属度原则

设给定论域 U 上的 n 个模糊子集(模糊模式)$\tilde{A}_1,\tilde{A}_2,\cdots,\tilde{A}_n$，对于 $u_0\in U$ 属于这 n 个模糊子集的隶属度分别为 $\mu_{\tilde{A}_1}(u_0)$，$\mu_{\tilde{A}_2}(u_0),\cdots,\mu_{\tilde{A}_n}(u_0)$，若有 $i\in\{1,2,\cdots,n\}$，使得 $\mu_{\tilde{A}_i}(u_0)=\max(\mu_{\tilde{A}_1}(u_0),\mu_{\tilde{A}_2}(u_0),\cdots,\mu_{\tilde{A}_n}(u_0))$，则认为元素 u_0 应该隶属于 $\tilde{A}_i$，判决 u_0 归属于 $\tilde{A}_i$ 所代表的那个模式，这就是最大隶属度原则.

例 5.1 某种柴油机“负荷运转不足”的五个主要故障原因是 y_1(气门弹簧断)、y_2(喷油头积碳堵塞)、y_3(机油管破裂)、y_4(喷油过迟)、y_5(喷油泵驱动键滚键). 六个症状分别是：x_1(排气过热)，x_2(振动)，x_3(扭矩急降)，x_4(机油压过低)，x_5(机油耗量大)，x_6(转速上不去). 现在某台柴油机的故障症状有：x_3,x_4,x_5 三种，其严重程度分别为 0.5,0.6,0.8，试求其故障原因.

解 症状向量为(0,0,0.5,0.6,0.8,0).

诊断算法取模型Ⅳ即 $M(\bullet,\oplus)$.

根据柴油机的经验资料和机理分析，得出诊断矩阵：

$$\underset{\sim}{R}=\begin{bmatrix}0.6 & 0.4 & 0 & 0.98 & 0\\ 0.8 & 0.98 & 0.3 & 0 & 0\\ 0.95 & 0 & 0.8 & 0.3 & 0.98\\ 0 & 0 & 0.98 & 0 & 0\\ 0 & 0 & 0.9 & 0 & 0\\ 0.3 & 0.6 & 0.9 & 0.98 & 0.95\end{bmatrix}$$

故障原因向量为

$$\underset{\sim}{Y}=\underset{\sim}{X}\circ\underset{\sim}{R}=(0.475,0,1,0.15,0.49)$$

即故障原因向量中各元素的隶属度为

$$\mu_{y_1}=0.475,\ \mu_{y_2}=0,\ \mu_{y_3}=1,\ \mu_{y_4}=0.15,\ \mu_{y_5}=0.49$$

其中 μ_{y_3} 最大，即在柴油机的五种故障原因中具有原因 y_3 的可能性最大，根据最大隶属度原则，可以认为故障原因为 y_3，即机油管破裂.

最大隶属度原则诊断法的优点是简单易行，在计算机上实现容易. 缺点是概括的信息量太少，完全不考虑其余一切隶属度较小的因素对诊断判决应起的作用，而且当最大隶属度值与其它隶属度值之间的差距不是较大时难以作出可靠的诊断结论.

(2) 阈值原则

规定一个阈值水平 $\lambda\in[0,1]$，记 $\alpha=\max(\mu_{A_1}(u),\mu_{A_2}(u),\cdots,\mu_{A_n}(u))$，若 $\alpha<\lambda$，则作“拒识”的判决，说明提供的输入信息(症状向量)不足，在诊断人员补足信息之后再重新诊断；若 $\alpha>\lambda$，则认为诊断可行，此时如果 $\mu_{A_i}(u)>\lambda$，则认为：诊断对象具有与之对应的故障原因. 可见采用阈值原则可能会得出几种故障原因，也可能得不出故障原因，不像最大隶属度原则，每次肯定得出且仅得出一种故障原因.

在实际应用中，有时将阈值原则与最大隶属度原则结合起来使用. 例如，可以先判断是否有 $\alpha\geqslant\lambda$，如没有则拒判，如有则按最

大隶属度原则进行判决. 近年来发展起来的浮动阈值、分级阈值等技术，使诊断精度得到进一步的提高.

有些情况下，可以将单一的阈值改变成阈值向量 $\Lambda=(\lambda_1,\lambda_2,\cdots,\lambda_n)$.

考虑一个诊断实例，某设备加工方式症状论域为

$U=$｛崩刀，刀尖发红，工作表面出现亮点，切屑底面粗糙，切屑颜色变化，出现尖叫声，切削不轻快，刀架有振感｝

故障论域为

$V=$｛刀具磨损，刀具破损，表面粗糙度不合要求，切削振动｝

阈值向量为

$$\Lambda=(0.60,0.60,0.60,0.55)$$

诊断权矩阵为

$$\underset{\sim}{R}=\begin{bmatrix}0.025 & 0.150 & 0.150 & 0.025 & 0.150 & 0.200 & 0.275 & 0.025\\ 0.800 & 0 & 0 & 0 & 0.050 & 0.050 & 0.050 & 0.050\\ 0.200 & 0.100 & 0.050 & 0.200 & 0.100 & 0.100 & 0.100 & 0.150\\ 0.250 & 0.025 & 0.025 & 0.025 & 0.025 & 0.200 & 0.050 & 0.400\end{bmatrix}^T$$

设某次诊断中，症状的隶属度向量为

$$\underset{\sim}{A}=(0.2,0.6,0.4,0.3,0.4,0.8,0.8,0.7)$$

诊断模型采用算子 $M(\bullet,+)$.

这样，可得出故障隶属度输出：

$$\underset{\sim}{Y}=\underset{\sim}{A}\circ\underset{\sim}{R}=(0.620,0.295,0.485,0.5725)$$

通过与阈值向量 Λ 的比较，得出：

$$y_1=0.620>\lambda_1,\ y_2=0.295<\lambda_2$$

$$y_3=0.485<\lambda_3,\ y_4=0.5725>\lambda_4$$

即加工过程中，出现了两种故障：刀具磨损和切削振动.

(3) 择近原则

设 $A_i,B\in F(U),i=1,2,\cdots,n$，若存在 j，使

$$N(A_j,B)=\max\{N(A_1,B),N(A_2,B),\cdots,N(A_n,B)\}\tag{5.3.12}$$

则认为 B 与 A_j 最贴近，即判 B 与 A_j 为一类，这种原则就成为择近原则. 式(5.3.12)中 $N(A_i,B)$为 A_i 与 B 的任意一种贴近度.

例 5.2 汽轮发电机组的状态监测与故障诊断.

某厂一台 50MW 汽轮发电机组近几年来的 6 次测试(6 个样本)数据如表 5.2 所示. 每个样本用 10 个指标表示时域、频域参量.

表 5.2 机组振动信号计算数据

指标 \ 样本		1	2	3	4	5	6
1	波形偏差	23.45	73.67	99.54	83.70	24.37	154.56
2	均方差	194.59	295.32	264.15	156.66	185.94	183.34
3	自相关系数	0.9285	0.9929	0.9951	0.9485	0.9841	0.9627
4	一倍频幅值	251.56	382.41	317.95	172.30	224.79	162.87
5	二倍频幅值	22.87	102.89	96.53	107.56	103.31	148.92
6	三倍频幅值	20.17	25.70	30.31	34.53	29.01	57.86
7	四倍频幅值	5.93	40.74	28.64	26.27	15.75	42.77
8	五倍频幅值	5.70	29.92	10.69	10.68	17.15	11.86
9	六倍频幅值	4.06	32.49	17.20	23.52	13.88	33.15
10	分频幅值	18.45	2.00	4.11	23.60	10.62	10.78

表 5.2 中的“波形偏差”是指时域波形峰-峰值的平均值的中点偏离波形均值的大小，它反映轴承刚度不均匀、不同频率分量的相位不同等因素. 均方差、自相关系数、一倍频至六倍频幅值等与一般习惯定义相同. 这里的分频是指小于一倍频的频率分量. 均方差表示动态信号波动的大小；自相关系数是取 $\tau>50\Delta T$ 后的平均，表示噪声衰减程度或故障周期分量占总能量的比重；各频率分量分别表示失衡、不对中、松动、油膜涡动、刚度不均、非线性等故障原因.

对表 5.2 中的数据建立的模糊子集列于表 5.3 中. 相应的隶属函数选择如下：

属于波形偏差小模糊子集的隶属函数为

$$\mu_1(u)=\begin{cases}0, & u\leqslant 0\\ e^{-0.0002u^2}, & u>0\end{cases}$$

其中 u 是表 5.2 中的波形偏差数值.

属于相关性好的模糊子集的隶属函数为

$$\mu_2(u)=\begin{cases}0, & u\leqslant 0.9\\ 0, & u>1.0\\ 10u-9, & 0.9<u\leqslant 1.0\end{cases}$$

其中 u 是表 5.2 中的自相关系数值.

属于诸频率幅值小和波动小(即方差小)模糊子集的隶属函数为

$$\mu_3(u)=\begin{cases}1, & u\leqslant a_1\\ \dfrac{a_2-u}{a_2-a_1}, & a_1\leqslant u\leqslant a_2\\ 0, & u>a_2\end{cases}$$

其中 u 是一至六倍频幅值及分频幅值和均方差.

用于计算的各隶属函数公式也列于表 5.3 中.

表 5.3 各状态的隶属函数值

样本 / 模糊子集	1	2	3	4	5	6	隶属函数
波形偏差小	0.90	0.34	0.14	0.25	0.88	0.01	$\exp(-0.0002u^2)$
波动(方差)小	0.68	0.35	0.45	0.81	0.71	0.72	$(400-u)/(400-100)$
相关性好	0.29	0.93	0.96	0.49	0.84	0.63	$10u-9$
一倍频幅值小	0.62	0.29	0.46	0.82	0.69	0.84	$(500-u)/(500-100)$
二倍频幅值小	0.89	0.49	0.52	0.46	0.48	0.26	$(200-u)/200$
三倍频幅值小	0.66	0.57	0.50	0.42	0.52	0.04	$(60-u)/60$
四倍频幅值小	0.88	0.18	0.43	0.47	0.69	0.14	$(50-u)/50$
五倍频幅值小	0.87	0.34	0.76	0.76	0.62	0.74	$(45-u)/45$
六倍频幅值小	0.90	0.19	0.57	0.41	0.65	0.17	$(40-u)/40$
分频幅值小	0.39	0.93	0.24	0.21	0.65	0.64	$(60-u)/60$

为了衡量这六次样本的运行情况,必须建立一个理想样本,只

有这样才能进行比较. 根据机组运行情况，知道机组处于理想的运行情况时各指标值必须小于某一值或等于零，从属于表 5.3 的隶属度均为 1. 表 5.4 列出了 6 个样本状态分别与理想状态的两种贴近度值，贴近度 I 为海明贴近度，贴近度 II 为欧几里德贴近度.

表 5.4　各样本状态与理想状态的贴近度

贴近度 \ 样本	1	2	3	4	5	6
贴近度 I	0.708	0.461	0.503	0.510	0.673	0.419
贴近度 II	0.639	0.402	0.451	0.468	0.654	0.343

从表 5.4 中可以看出：海明贴近度与欧几里德贴近度计算结果相近，样本 1 和样本 5 与理想状态接近.

5.3.4　诊断矩阵的确定

前面就说过，若征兆有 m 个，故障有 n 种，则诊断权矩阵是一个 $m\times n$ 维的矩阵. 如何精确地构造这个诊断矩阵 $\underset{\sim}{R}$，是模糊诊断中的核心问题.

$\underset{\sim}{R}$ 阵的建立包含两个阶段：1）由专家经验设定初始值；2）在诊断过程中，根据经验积累对权矩阵进行修改.

在第一个阶段，往往要集中多名专家分析故障集与征兆集的关系，建立每一种故障 i 与各种征兆的联系 W_{ij}，$j=1,\cdots,m$，并满足 $\sum_{j=1}^{m} W_{ij}=1$. 当 $W_{ij}=0$ 时表示征兆 j 的存在对故障 i 的发生不起作用. W_{ij} 的值越大，表示征兆 j 的存在对故障 i 的发生起的作用越大.

由于两方面的原因，使得 $\underset{\sim}{R}$ 在刚建立时不够精确.

1）专家的知识可能会出错，不同专家的知识会发生矛盾就充分地说明了这一点. 即使他们的知识不存在错误，然而要把知识由自然语言的形式转变到 0—1 之间的模糊数表示也很难. 比如说，若专家有知识：故障 i 存在则征兆 j 比较严重，那么，$W_{ij}=0.6$ 还是 $W_{ij}=0.7$?

2）诊断对象所处的环境和本身的状况在经过长时间的运转后，本身的性能和参数都会发生改变. 这些变化也必然会导致故障与征兆关系的不断改变.

为了使诊断矩阵能准确地反映故障与征兆的关系并适应这种关系的变化，有必要在诊断过程中对诊断权矩阵不断进行自适应修正，这便是 $\tilde{R}$ 建立的第二个阶段. 关于这一阶段的具体算法可以参阅文献[1].

5.3.5 模糊故障诊断的流程框图

根据前面的讨论，可以将模糊故障诊断的全过程进行总结，并得出如下的流程图.

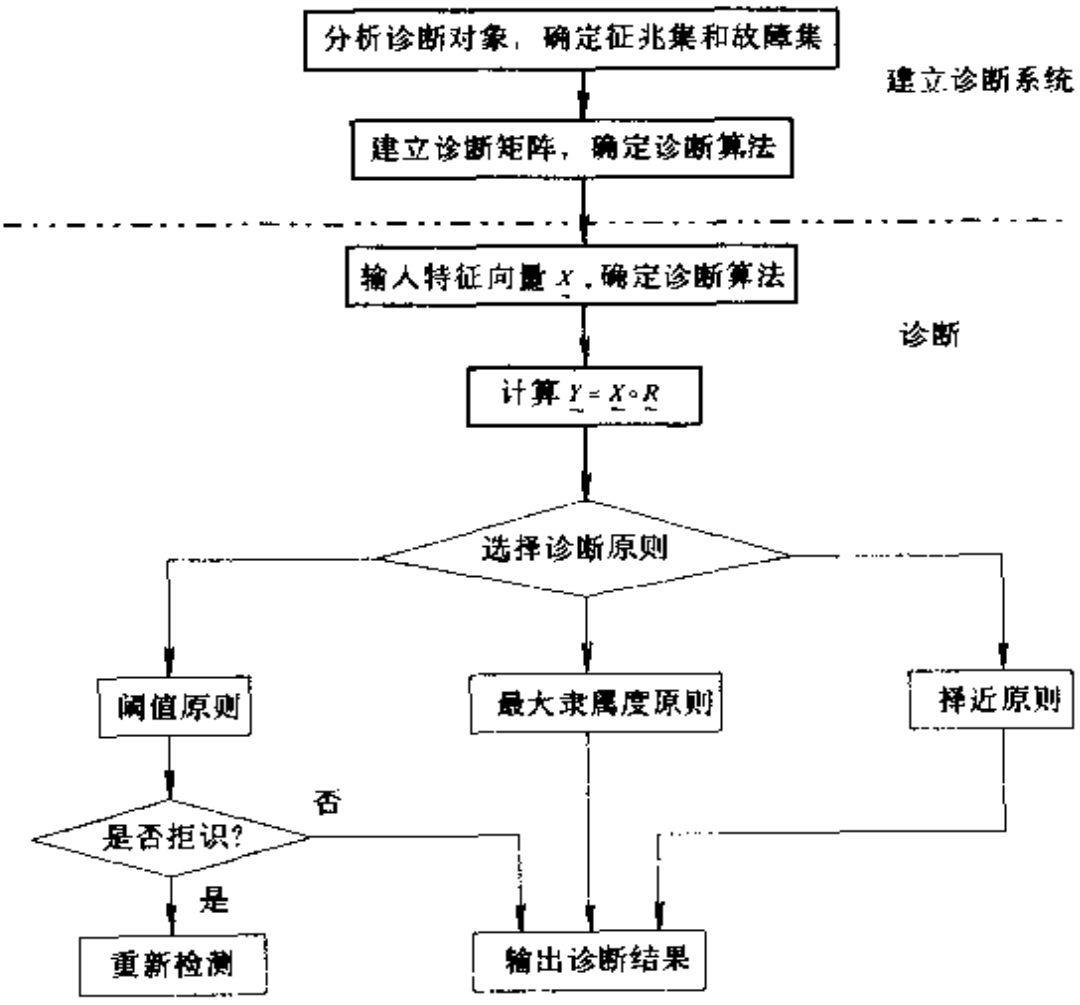


图 5.3 模糊故障诊断流程图

5.3.6 多级模糊诊断模型

如果诊断对象的征兆和故障都比较多，使得很难合理地给出一个模糊诊断权矩阵，这时就有必要采取多级诊断模型. 以二级模型为例：

$$\underset{\sim}{Y}=\underset{\sim}{X}\circ\underset{\sim}{R}=\underset{\sim}{X}\circ\begin{bmatrix}\underset{\sim}{X}_1 & \circ & \underset{\sim}{R}_1\\ \underset{\sim}{X}_2 & \circ & \underset{\sim}{R}_2\\ \vdots & \vdots & \vdots\\ \underset{\sim}{X}_l & \circ & \underset{\sim}{R}_l\end{bmatrix}=\underset{\sim}{X}\circ\begin{bmatrix}\underset{\sim}{B}_1\\ \underset{\sim}{B}_2\\ \vdots\\ \underset{\sim}{B}_l\end{bmatrix}=\underset{\sim}{A}\circ(b_{ij})_{l\times m}$$

式中$\underset{\sim}{X}_i\circ\underset{\sim}{R}_i$为一级诊断模型. 二级诊断模型的具体工作原理与一级诊断模型相同. 如果使用二级诊断模型仍然觉得很复杂，那么可以采用三级诊断模型.

5.3.7 一个诊断实例：内燃机气阀结构故障的模糊诊断

内燃机气阀机构，特别是排气阀，是经常处在高温、高速气流冲刷下工作的零部件. 强化内燃机的气阀温度可达 900℃，另外，气阀落座频繁且冲击力大. 因此，内燃机工作时，排气阀承受很高的热压力和机械压力. 长期以来，排气阀被认为是内燃机最容易发生故障的零部件之一.

在内燃机的故障诊断中，由于存在检测数据中包含大量噪声，内燃机结构的多样性及内燃机故障与征兆之间关系的错综复杂，使得传统的诊断方法在实际应用中暴露出它的局限性.

对 2015 内燃机气阀机构设置异常间隙，在气缸盖上测取振动加速度，求出其 ESD(能量密度谱函数). 考虑内燃机气阀机构的主要故障，其故障论域 X 为

$$X=\{x_1\text{ 间隙异常},\ x_2\text{ 漏气},\ x_3\text{ 阀头翘曲变形}\}$$

其中各元素 x_i 的隶属度 μ_{x_i} 组成模糊向量 $\underset{\sim}{C}$：

$$\underset{\sim}{C}=(\mu_{x_1},\ \mu_{x_2},\ \mu_{x_3})$$

征兆(即气缸盖测点 ESD)论域 U 为

$$U=\{y_1,y_2,y_3,y_4,y_5\}$$

其中 $y_j(j=1,2,\cdots,5)$为频率 f_j 时的 ESD 值,各元素 y_j 的隶属度 μ_{y_j}组成模糊向量 $\underset{\sim}{B}$:

$$\underset{\sim}{B}=(\mu_{y_1},\mu_{y_2},\mu_{y_3},\mu_{y_4},\mu_{y_5})$$

两论域之间存在模糊关系 $\underset{\sim}{R}$.通过大量的试验、分析、测试及经验总结,得到 2105 型内燃机气阀机构的 $\underset{\sim}{R}$ 为

$$\underset{\sim}{R}=\begin{bmatrix}0.024 & 0.915 & 0.884\\ 0.037 & 0.030 & 0.057\\ 0.282 & 0.028 & 0.029\\ 0.301 & 0.015 & 0.016\\ 0.356 & 0.012 & 0.014\end{bmatrix}$$

这样,我们在已知模糊向量 $\underset{\sim}{B}$ 时,便可以根据 $\underset{\sim}{C}=\underset{\sim}{B}\circ\underset{\sim}{R}$,求出 $\underset{\sim}{C}$,从而获得诊断结果.

现在,已知待检模式 $\underset{\sim}{B}=(0.032,0.040,0.277,0.308,0.343)$,可计算得到

$$\underset{\sim}{C}=\underset{\sim}{B}\circ\underset{\sim}{R}=(0.343,0.032,0.040)$$

根据最大隶属度原则可知,待检模式存在故障气阀间隙异常.

5.4 模糊神经网络

在诊断领域中,模糊逻辑理论和神经网络技术在知识表示、知识存贮、推理速度及克服知识窄台阶效应等方面起到了很大的作用.但模糊逻辑和神经网络分别是模仿人脑的部分功能,因此,它们各有偏重:模糊逻辑主要模仿人脑的逻辑思维,具有较强的结构性知识表达能力;神经元网络模仿人脑神经元的功能,具有强大的自学习能力和数据的直接处理能力.具体地讲,可以从表 5.5 来分析模糊逻辑和神经网络的异同:

表 5.5 神经网络与模糊逻辑的比较

	神经网络	模糊逻辑
组成	神经元互连	模糊逻辑模糊规则
映射关系	点与点之间的对应	块与块之间的对应
知识存储方式	连接权值	规则的方式
知识表达能力	弱	强
容错能力	强	较强
学习能力	能进行学习	不能学习
精度比较	高	较高
计算量	多	少
应用	用于建模、模式识别、估计	用于可凭经验处理的系统

从这个表中，可以看出神经网络和模糊逻辑各有优缺点，因此，有必要将模糊逻辑与神经网络融合起来构成模糊神经网络，使之能同时具有模糊逻辑和神经网络的优点，主要表现在既能表示定性知识又能具有强大的自学习能力和数据处理能力.

5.4.1 模糊神经元

要讨论模糊神经网络，首先必须研究模糊神经元.

在第四章中介绍神经元时提出的模型有如下的信息处理能力：

$$\text{net} = \sum_{i=0}^{n-1} w_i x_i - \theta, \quad x_i \in [-\infty, +\infty]$$

$$y = f(\text{net})$$

式中 $x_i, i=0,\cdots,n-1$ 为该神经元的输入，w_i 为对应于输入 x_i 的连接权值，θ 为该神经元的阈值，y 为输出，$f[\cdot]$为一转换函数，通常采用的是 Sigmoid 函数.

现在将这种神经元模型进行推广，使之具有一般表达式：

$$\text{net} = \mathop{\hat{+}}_{i=0}^{n-1} (w_i \mathbin{\hat{\cdot}} x_i) - \theta \tag{5.4.1}$$

$$y = f(\mathrm{net}) \tag{5.4.2}$$

即以算子($\hat{+}$，$\hat{\cdot}$)代替算子(+，·)．算子($\hat{+}$，$\hat{\cdot}$)称为模糊神经元算子，对任意$a,b\in[0,1]$，它满足：

1) 交换率 $a\hat{+}b=b\hat{+}a$

$a\hat{\cdot}b=b\hat{\cdot}a$

2) 结合率 $(a\hat{+}b)\hat{+}c=a\hat{+}(b\hat{+}c)$

$(a\hat{\cdot}b)\hat{\cdot}c=a\hat{\cdot}(b\hat{\cdot}c)$

3) $0\hat{\cdot}a=0$

当$x_i\in[0,1]$，$i=0,\cdots,n-1$时，采用模糊神经元算子，且满足式(5.4.1)和式(5.4.2)的神经元模型即为模糊神经元模型．对于模糊神经元有如下的示意图．

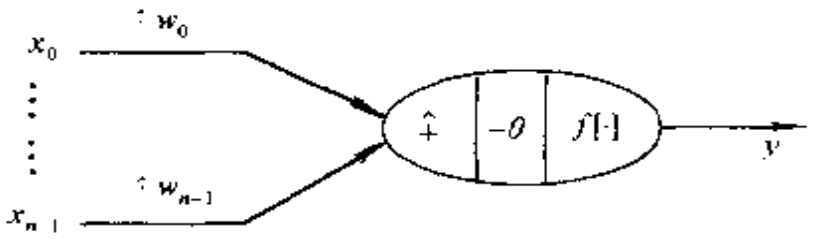


图 5.4 模糊神经元

常用的模糊神经元算子有 6 种，表 5.6 列出了它们的定义．

表 5.6 常用模糊神经元算子

序 号	算子名称	$\hat{+}$	$\hat{\cdot}$
1	和与积	+	·
2	取小与积	∧	·
3	取大与积	∨	·
4	和与取小	+	∧
5	取小与取小	∧	∧
6	取大与取小	∨	∧

这样，我们也可以得到 6 种类型的常用模糊神经元，下面作为

示例介绍其中两种.

例 5.3 取大神经元.

模糊神经元中,各项输入 $x_i\in[0,1]$,$i=0,\cdots,n-1$,算子$(\hat{+},\hat{\cdot})=(\vee,\cdot)$,各项连接权满足 $w_0=w_1=\cdots=w_{n-1}=1$,阈值 $\theta=0$.

$$f(x)=\begin{cases}1, & x\geqslant 1\\ x, & 0<x<1\\ 0, & \leqslant 0\end{cases}$$

则

$$y=f\left[\bigvee_{i=0}^{n-1}(1\cdot x_i)-\theta\right]$$

$$=\bigvee_{i=0}^{n-1}x_i$$

例 5.4 取余神经元.

模糊神经元中,只有一个输入 $x\in[0,1]$,阈值 $\theta=0$,算子$(\hat{+},\hat{\cdot})=(+,\cdot)$,连接权值 $w=1$,$f(x)=1-x$,则有

$$y=f[wx-\theta]=f[x]=1-x$$

5.4.2 模糊神经网络

由两个或两个以上个模糊神经元相互连接而形成的网络就是模糊神经网络(FNN),它是模糊逻辑与神经网络相融合的产物.构成模糊神经网络的方式有两种:

1) 传统神经网络模糊化.这种 FNN 保留原来的神经网络结构,而将神经元进行模糊化处理,使之具有处理模糊信息的能力.

2) 基于模糊模型的 FNN.这种 FNN 的结构与一个模糊系统相对应.

就具体形式而言,FNN 可以分为五大类:

1) FNN_1 神经元之间的运算与常规的神经网络相同,采用 Sigmoid 函数,输入值改为模糊量.

2) FNN_2 神经元之间的运算与常规的神经网络相同,采用

Sigmoid 函数，权值改为模糊量.

3) FNN_3 神经元之间的运算与常规的神经网络相同，采用 Sigmoid 函数，输入值和权值都改为模糊量.

4) HNN 输入、权值与常规的神经网络相同，但是用与、或运算代替符取代 Sigmoid 函数.

5) HFNN 具体地说，这一形式也具有三种方式，分别是在 FNN_1，FNN_2，FNN_3 基础上，采用与、或运算代替符取代 Sigmoid 函数.

5.4.3 模糊 BP 网络

正如 BP 网络是最常用的神经网络一样，模糊 BP 网络也是最常用的模糊神经网络. 一个具有两个输入、两个输出的 FNN_1 型模糊 BP 网络具有下图的结构方式：

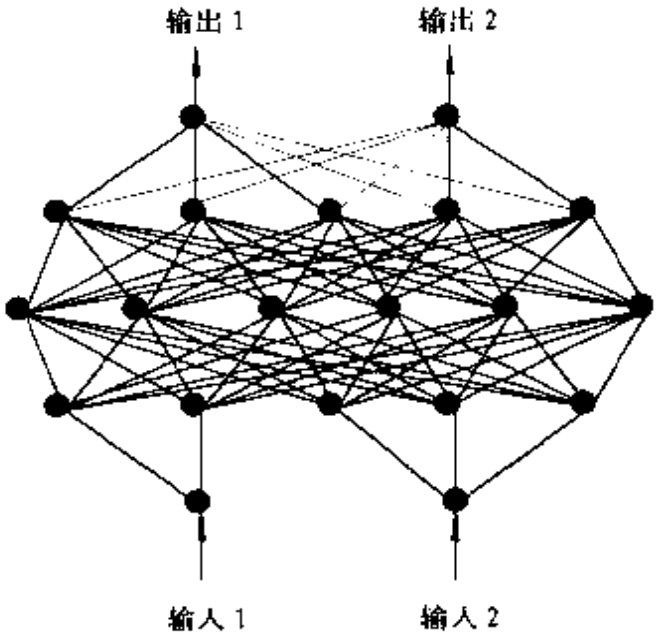


图 5.5 模糊 BP 网络结构图

该诊断网络中共有五层，其中第一层为输入层，它的每一个节点代表一个输入变量，它不加作用地将其输入输出到下层；第二层是量化输入层，作用是将输入变量模糊化；第三层为 BP 网络的隐含层，其作用与普通 BP 网络基本相同，用于实现输入变量模糊

值到输出变量模糊值的映射；第四层为量化输出层，其输出是模糊化数值；第五层是加权输出层，实现输出的清晰化.

从图 5.5 也可以知道，由第二、三、四层所组成的网络，在对权值没有约束的情况下，与普通的 BP 网络没有区别，因此也可以采用前面介绍的 BP 算法进行学习.

在故障诊断中，使用模糊 BP 网络模型时通常是将第四层直接作为最终的输出，每个输出神经元代表一种故障，各个模糊数值代表该故障存在的程度(有时将其转化为一定的语言表达形式，如 0.95 表示严重存在某种故障). 这样一个具有两个征兆和两个故障的 FNN_1 型模糊 BP 网络具有的结构方式往往如下：

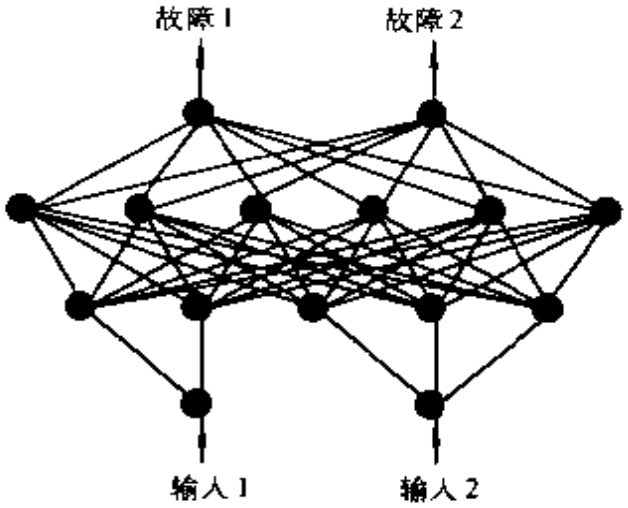


图 5.6 常用于诊断中的模糊 BP 网络

如果不考虑模糊化过程，同时一个输入只对应一个量化输入，则第四章介绍的 BP 网络也可以看作是一种模糊神经网络.

5.4.4 模糊联想记忆

模糊联想记忆(FAM)有两种常用的形式：

1. Max-min 型的模糊联想记忆

这种 FAM 具有如下图所示的网络拓扑结构：

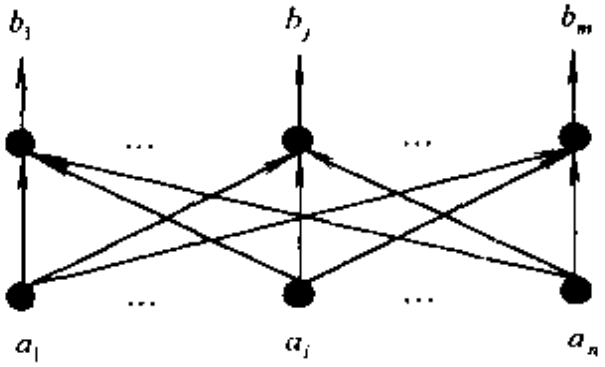


图 5.7　Max_min 型模糊联想记忆的拓扑结构

这种 FAM 中的神经元都采用取大取小模糊神经元，即表 5.6 中所示的第 6 类模糊神经元，在该类神经元中还要求满足各神经元的阈值都为 0，$f[x]=x$. 设现在有 P 个模糊模式对(A_p,B_p)，$p=1,\cdots,P$. 其中 $A_p=(a_p^1,\cdots,a_p^n)$，$B_p=(b_p^1,\cdots,b_p^m)$，$a_p^i\in[0,1]$，$b_p^j\in[0,1]$，对任意 $i=1,\cdots,n$，$j=1,\cdots,m$，$p=1,\cdots,P$.

这种 FAM 也包括学习和联想两个过程. 在学习过程就是寻找权值矩阵 W，使满足：

$$A_p\circ W=B_p,\quad p=1,\cdots,P \tag{5.4.3}$$

式中算子 $\circ$ 为$(\vee,\wedge)$.

在联想阶段，就是给网络提供输入模式 $A=(a_1,\cdots,a_n)$，网络通过取大-取小合成运算，联想出输出模式 $B=(b_1,\cdots,b_m)$. 具体的运算公式为

$$b_j=\bigvee_{i=1}^{n}(a_i\wedge w_{ij}),\quad j=1,\cdots,m \tag{5.4.4}$$

Max_min 型 FAM 模糊联想记忆的学习算法为

1）赋初值

令 $w_{ij}=1$，对$\forall\ i,j;p=1$.

2）给定输入和输出(A_p,B_p)

输入各模糊模式对，输入模式$(a_p^1,\cdots,a_p^n)$，输出模式$(b_p^1,\cdots,b_p^m)$.

3.1）计算实际输出

计算公式为：$(b_p^j)^r=\bigvee_{i=1}^{n}(w_{ij}\wedge a_p^i)$.

式中$(b_p^j)^r$表示第 p 个模糊模式对在训练时第 j 个分量的实际输出，w_{ij}为 F_1 中第 i 个节点到 F_2 中第 j 个节点的权，a_p^i 为输入模式的第 i 个分量.

3.2) 调权

令 $\delta_{pj}=(b_p^j)^r-b_p^j$，则

$$\begin{cases}w_{ij}(t+1)=w_{ij}(t)-\eta\delta_{pj}, & \text{如果}(w_{ij}(t)\wedge a_p^i)>b_p^j\\ w_{ij}(t+1)=w_{ij}(t), & \text{其它}\end{cases}$$

式中 η 为比例因子，满足 $0<\eta\leqslant 1$.

3.3) 验证是否对所有 i,j 都存在 $w_{ij}(t+1)=w_{ij}(t)$？如存在，转 4)，否则返回 3.1)

4) $p=p+1$，重复 2)，直到 $p=P+1$ 结束

2. Rule 型 FAM

这种 FAM 具有如下的系统结构：

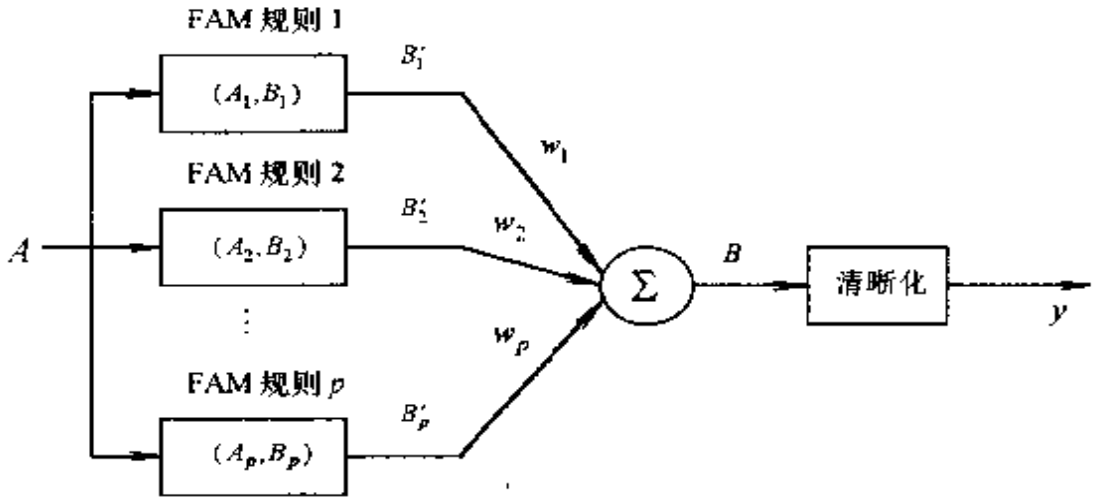


图 5.8 Rule 型 FAM 系统结构

在这种联想记忆系统中，规则库可以写成(A_1,B_1)，(A_2,B_2)，…，(A_p,B_p). 每个输入 A 可以不同程度地作用模糊联想记忆中的每一条规则. 最小模糊联想记忆系统(A_i,B_i)映射输入 A 到 B_i'，B_i'

为 B 的一部分. 相应的输出模糊集合 B 可以通过各部分作用的模糊集合结果的组合来获得，即有公式：

$$B=w_1B_1'+w_2B_2'+\cdots+w_pB_p'$$

其中，w_i 反映模糊联想记忆规则(A_i,B_i)的可信度(或强度). 在实际中，还将输出模糊向量 B 通过清晰化(defuzzify)变成输出 y.

对于这个 FAM，现在关心的问题有三个：如何利用规则(A_i,B_i)和输入模糊向量 A 产生联想结果 B_i'；如何构造有 p 条规则的 FAM；如何合成模糊输出 B.

(1) 形成矩阵 $M_i(i=1,2,\cdots,p)$

由于矩阵 M_i 对应 FAM 规则 i，因此该矩阵必须包含信息(A_i,B_i). 即 M_i 应该满足：

$$A\circ M=B,\ M=A^T\circ B$$

取算子 $\circ$ 为取大取小算子，从而有

$$m_{ij}=\min(a_i,b_j)$$

例 5.4 若 $A=(0.3,0.4,0.8,1)$，$B=(0.8,0.4,0.5)$，则

$$M=A^T\circ B=\begin{bmatrix}0.3\\0.4\\0.8\\1\end{bmatrix}\circ(0.8,0.4,0.5)=\begin{bmatrix}0.3&0.3&0.3\\0.4&0.4&0.4\\0.8&0.4&0.5\\0.8&0.4&0.5\end{bmatrix}$$

可以证明，M 具有联想功能：

$$A\circ M=(0.8,0.4,0.5)=B$$

同时由于，$B\circ M^T=(0.3,0.4,0.8,0.8)\neq A$

即不能实现反向联想.

(2) 构造有 p 条规则的 FAM

M_i 实现了单个规则 $A_i\to B_i$ 的联想，如何将 p 条规则合成在一个 FAM 中？一个自然的想法就是将 M_i 累加即进行运算：

$$M=\bigvee_{i=1}^{p}M_i$$

再以 M 通过计算 $A\circ M=B$ 来实现 $A\to B$ 的联想. 这种想法是错误的，因为 M 并不能实现对联想记忆对(A_i,B_i)，$i=1,2,\cdots,p$ 的

存储. 正确的做法是通过对每条规则的联想结果进行累加，即由 $B_1', B_2', \cdots, B_p'$ 来组合 B. 这样，为实现对 p 条规则的存储，必须有 p 个权值矩阵 $M_1, M_2, \cdots, M_p$.

假设一个单输入单输出的系统，输入经过模糊化后变为 3 个量值输入，输出在清晰化前是 2 个量值输出，同时假设该系统只有 3 条规则，则这个 FAM 可能的实际结构为

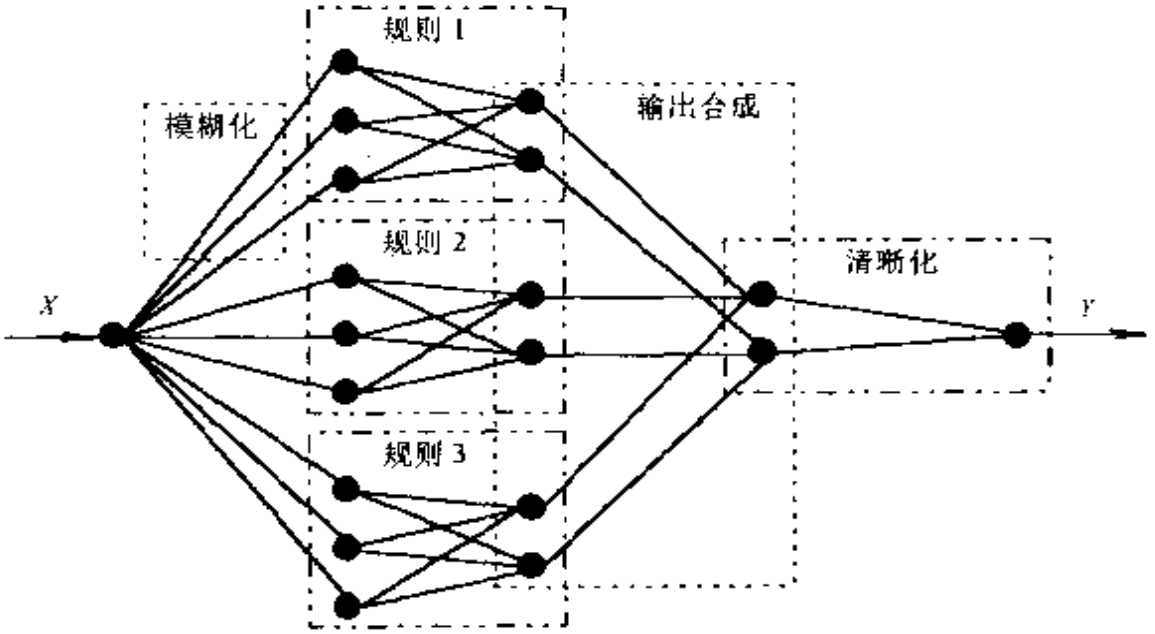


图 5.9 一个 FAM 神经元连接图

从图中可以看出，为了实现对多条规则的存储，需要一个很大的网络结构，所用的内存开销将是十分巨大的，但是这种网络的空间结构十分清晰.

(3) 输出模式的合成

如前面指出，输出向量 B 是对每条规则所产生的向量 B_i' 的加权和：

$$B = \sum_{i=1}^{p} w_i B_i'$$

w_i 为一非负的数据，通常人为确定.

5.5 模糊神经网络在诊断中的应用

本节主要讨论前面 5.4 节介绍的三种模糊神经网络在故障诊断中的应用.

5.5.1 模糊 BP 网络在故障诊断中的应用

这里,我们仍以本章例 3.2 汽轮发电机组的状态监测与故障诊断为例来说明模糊 BP 网络在故障诊断中的应用.

要在故障诊断中使用模糊 BP 网络首先必须确定该神经网络的结构,因此,第一步就必须确定量化输入层的神经元个数.为此,对每一个输入量的模糊化需最先进行.这里,假设每一个输入量都对应着两个量化输入,如表 5.7 所示.当然,也可以对应三个、四个或只对应一个,这可以根据实际情况的需要来确定.如本例的波形偏差这一项,我们也可以使之对应波形偏差小、波形偏差中和波形偏差大三个量化输入.

表 5.7 各状态的隶属函数值

模糊子集 \ 样本		1	2	3	4	5	6	隶属函数
输入	波形偏差小 x_1	0.90	0.34	0.14	0.25	0.88	0.01	$\exp(-0.002u^2)$
	波形偏差大 x_2	0.10	0.66	0.86	0.75	0.12	0.99	$1-\exp(-0.002u^2)$
	波动方差小 x_3	0.68	0.35	0.45	0.81	0.71	0.72	$(400-u)/300$
	波动方差大 x_4	0.32	0.65	0.55	0.19	0.29	0.28	$(u-100)/300$
	相关性好 x_5	0.29	0.93	0.96	0.49	0.64	0.63	$10u-9$
	相关性差 x_6	0.71	0.07	0.04	0.51	0.16	0.37	$10-10u$
	一倍频幅值小 x_7	0.62	0.29	0.46	0.82	0.69	0.84	$(500-u)/400$
	一倍频幅值大 x_8	0.38	0.71	0.54	0.18	0.31	0.16	$(u-100)/400$
	二倍频幅值小 x_9	0.89	0.49	0.52	0.46	0.48	0.26	$(200-u)/200$
	二倍频幅值大 x_{10}	0.11	0.51	0.48	0.54	0.52	0.74	$u/200$

续表 5.7

	样本 / 模糊子集		1	2	3	4	5	6	隶属函数
输入	三倍频幅值小	x_{11}	0.66	0.57	0.50	0.42	0.52	0.04	$(60-u)/60$
	三倍频幅值大	x_{12}	0.34	0.43	0.50	0.58	0.48	0.95	$u/60$
	四倍频幅值小	x_{13}	0.88	0.18	0.43	0.47	0.69	0.14	$(50-u)/50$
	四倍频幅值大	x_{14}	0.12	0.82	0.57	0.53	0.31	0.86	$u/50$
	五倍频幅值小	x_{15}	0.87	0.34	0.76	0.76	0.62	0.74	$(45-u)/45$
	五倍频幅值大	x_{16}	0.13	0.66	0.24	0.24	0.38	0.26	$u/45$
	六倍频幅值小	x_{17}	0.90	0.19	0.57	0.41	0.65	0.17	$(40-u)/40$
	六倍频幅值大	x_{18}	0.10	0.81	0.43	0.59	0.35	0.87	$u/40$
	分频幅值小	x_{19}	0.39	0.93	0.24	0.21	0.65	0.64	$(30-u)/30$
	分频幅值大	x_{20}	0.61	0.07	0.76	0.79	0.35	0.36	$u/30$
输出	贴近度	y	0.639	0.402	0.451	0.468	0.654	0.343	————

第二步工作是确定 BP 网络隐含层的神经元个数和神经网络的整体结构. 选取隐含层的神经元数目可依据第四章中一般 BP 网络的原则. 这里选取的隐含层神经元数目为 5 个. 对于量化输出层,这里因为只要输出汽轮发电机组的运行状态,即与理想运行状态的欧几里德贴近度,因此该层只需要 1 个神经元即可. 模糊输出层可以将运行状况分为三个级别:良好、正常和不正常. 整个网络的结构如图 5.10 所示.

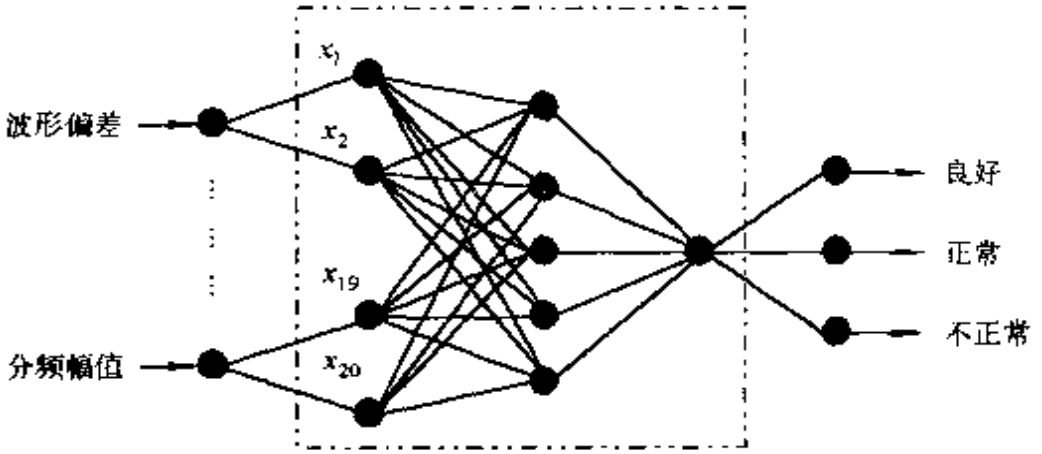


图 5.10 汽轮发电机组状态监测模糊 BP 网络结构图

第三步是对图 5.10 所示虚线框内部分进行训练.训练样本为表 5.7 中的数据,训练算法同第四章的 BP 网络,所得权值数据的集合即为知识库.表 5.8 为一种可能的知识库结构.

表 5.8　汽轮发电机组模糊 BP 网络知识库结构

量化输入层 / 隐含层	1	2	3	4	5	6	7	8	9	10
1	−1.01	0.70	−0.09	0.07	−0.11	0.83	0.51	0.12	0.26	0.88
2	−1.43	0.13	−0.30	−0.94	0.43	0.08	0.58	0.39	−0.78	−0.40
3	−0.73	0.16	0.94	−0.69	−0.08	−0.48	0.07	0.33	0.34	−0.00
4	−0.24	−0.60	1.07	−0.93	0.44	−0.16	−0.35	−0.05	−0.59	−0.37
5	−0.72	−0.56	−1.01	0.13	−0.80	0.49	−0.83	0.54	0.38	0.25

量化输入层 / 隐含层	11	12	13	14	15	16	17	18	19	20	阈值
1	0.37	0.00	−0.35	0.18	0.20	0.22	−0.27	0.80	−0.73	−0.63	−0.99
2	0.72	−0.25	0.23	0.00	0.30	−0.73	0.77	0.89	−0.38	0.15	0.50
3	0.52	−0.86	0.59	0.08	−0.21	−0.22	0.66	−0.15	0.61	0.41	0.80
4	1.11	0.45	0.41	−0.76	−0.29	−0.23	0.49	−0.44	0.37	0.51	0.64
5	0.88	−0.36	0.55	0.04	−0.39	0.63	0.55	0.18	0.26	0.10	−0.06

隐含层 / 量化输出层	1	2	3	4	5	阈值
1	−1.06	−1.09	−0.10	1.14	−0.35	0.76

第四步是利用知识库中的知识和模糊化和反模糊化函数进行

诊断.可以将这一步分解为下面几个小步：

1）输入各个征兆的实际数据.

2）利用表 5.7 中的隶属函数对各个输入进行模糊化，使之成为量化输入.

3）进行 BP 网络计算，得出量化输出.

4）依照一定的反模糊化原则，得到模糊输出.例如，设用 out_d 表示量化输出，out_l 表示模糊输出，则我们可以采用下面的反模糊化原则：

$$\text{out\_l} = \begin{cases} \text{好}, & \text{out\_d} > 0.60 \\ \text{正常}, & 0.6 \geqslant \text{out\_d} \geqslant 0.4 \\ \text{差}, & \text{out\_d} < 0.4 \end{cases}$$

例如现在有待诊断样本{波形偏差、波动(方差)、相关性、一倍频幅值、二倍频幅值、三倍频幅值、四倍频幅值、五倍频幅值、六倍频幅值、分频幅值}={48.50 230.64 0.9580 260.75 59.67 40.64 30.12 18.16 20.18 12.78}.

依照表 5.7 对这些实际数值进行模糊化，得到量化输入$\{x_1, \cdots, x_{26}\}$={0.62 0.48 0.56 0.44 0.58 0.42 0.60 0.40 0.70 0.30 0.32 0.68 0.40 0.60 0.60 0.40 0.50 0.50 0.57 0.43}.

采用表 5.8 中的知识库，进行推理即神经网络计算，得到量化输出 0.545.

将量化输出清晰化，得到输出最后的输出：汽轮发电机组运行状况正常.

如果计算待诊断样本与理想运行状况的欧几里德贴近度，其结果为 0.533，与模糊神经网络的输出相差无几.

5.5.2 Max_min 型 FAM 在故障诊断中的应用

这里以一个模拟的故障诊断系统为例说明其应用，由于该系统有 6 个输入和 2 个输出，因此其网络结构为

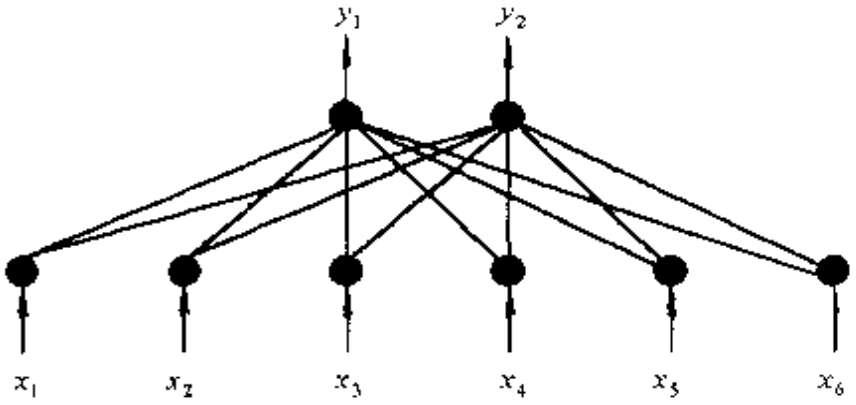


图 5.11　模拟诊断系统网络结构图

以表 5.9 中的 7 个学习样本对作为训练样本，对之进行训练.

表 5.9　模拟诊断系统模糊模式对

样本号	征　兆						故　障	
	1	2	3	4	5	6	1	2
1	0.7	0.2	0.4	0.6	0.3	0.8	0.6	0.4
2	0.9	0.4	0.4	1.0	0.7	0.0	0.9	0.6
3	0.3	0.6	0.5	0.5	0.2	0.9	0.5	0.5
4	0.1	0.7	0.4	0.7	0.7	0.7	0.7	0.6
5	0.3	0.6	0.4	0.8	0.2	0.4	0.8	0.5
6	0.7	0.9	0.6	0.1	0.7	0.4	0.7	0.6
7	0.8	0.1	0.6	0.5	0.4	0.3	0.6	0.4

依照 5.4.4 节介绍的训练算法，取 $\eta=0.6$，得到的连接权矩阵为

$$W=\begin{bmatrix}0.6 & 0.5 & 1.0 & 0.9 & 1.0 & 0.48\\ 0.4 & 0.5 & 0.4 & 0.4 & 0.6 & 0.4\end{bmatrix}^T$$

这样，我们就实现了把模拟系统的模糊模式对全部存储在权矩阵中.

对于 Max_min 型 FAM 诊断阶段也就是回想过程，可以验证，W 权矩阵可以准确地回想上面的 7 个样本中的每一个. 如对于第 6 个样本，则有

$$B_6 = A_6 \circ W = (0.7, 0.9, 0.6, 0.1, 0.7, 0.4) \circ \begin{bmatrix} 0.6 & 0.5 & 1.0 & 0.9 & 1.0 & 0.48 \\ 0.4 & 0.5 & 0.4 & 0.4 & 0.6 & 0.4 \end{bmatrix}$$
$$= (0.7, 0.6)$$

与训练样本 6 的输出完全一样.

对于一组新的样本征兆集,诊断系统也能给出一个合理的结果.

从训练算法和上面的分析也可以得出 Max_min 型 FAM 诊断系统的特点:

1) 系统结构简单,且总能收敛.

2) 与 BP 网络不同的是,该算法所赋初值并非随机值,而是都赋 1,因此具体的权值矩阵数值只与比例因子 η 的取值有关,且 η 的取值对 W 的关系也不是很大,如 $\eta=0.7$ 时模拟诊断系统中的 W 只改变了一个数 0.48 为 0.46.

3) 系统的知识容量小,容错性能不强.

4) 由于采用最大-最小算子,因此对输入输出数值有一定的要求:对于每一个训练样本都必须满足最大的征兆隶属度大于最大的故障隶属度,因而对于第四章的旋转机械试验台一例就不能用 Max_min 型 FAM.

Max_min 的 C 语言实现见附录 5.2.

5.5.3 Rule 型 FAM 在故障诊断中的应用

这里仍举一个模拟对象的例子来说明其应用.设某诊断对象表现出来的征兆有 3 种,故障为 2 类,专家经验知识被总结为 3 条规则.

规则 1:IF 征兆 1(0.9) 征兆 2(0.6) 征兆 3(0.2) THEN 故障 1(0.8) 故障 2(0.5)

规则 2:IF 征兆 1(0.4) 征兆 2(0.9) 征兆 3(0.4) THEN 故障 1(0.3) 故障 2(0.8)

规则 3:IF 征兆 1(0.8) 征兆 2(0.3) 征兆 3(0.7) THEN 故

障 1(0.6) 故障 2(0.7)

规则中括号里的数字表示征兆表现为症状的严重程度或故障存在的严重程度.

由 $M_i=A_i^T \circ B_i$ 得到 3 个矩阵分别表示 3 条规则:

$$M_1=\begin{bmatrix}0.8 & 0.5\\ 0.6 & 0.5\\ 0.2 & 0.2\end{bmatrix},\ M_2=\begin{bmatrix}0.3 & 0.4\\ 0.3 & 0.8\\ 0.3 & 0.4\end{bmatrix},\ M_3=\begin{bmatrix}0.6 & 0.7\\ 0.3 & 0.3\\ 0.5 & 0.7\end{bmatrix}$$

设现在有待诊断样本:征兆 1(0.7),征兆 2(0.2),征兆 3(0.8),即 $A=(0.7,0.2,0.8)$,那么有

$$B_1=A\circ M_1=(0.7,0.5)$$
$$B_2=A\circ M_2=(0.3,0.4)$$
$$B_3=A\circ M_3=(0.6,0.7)$$

将 $\sum_{i=1}^{p} w_i$ 看为一个算子,并将 $B=\sum_{i=1}^{p} w_i B_i'$ 理解为 $B=\bigcup_{i=1}^{p} B_i'$.

由此有

$$B=B_1\cup B_2\cup B_3=(0.7,0.7)$$

附录 5.1　模糊诊断系统 C 语言程序

在本系统中，是为了实现模糊诊断 $Y=X\circ R$. R 为诊断权矩阵，预先确定好，放置在数据文件中. 作为示范，征兆向量 X 的确定方法有四种.

(1) 降半梯形分布

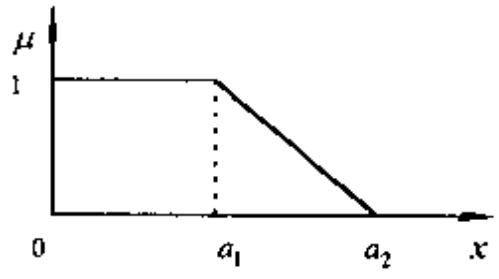
$$\mu_A(x)=\begin{cases}1, & x\leqslant a_1\\ \dfrac{a_2-x}{a_2-a_1}, & a_1<x\leqslant a_2\\ 0, & a_2<x\end{cases}$$

见附图 5.1.

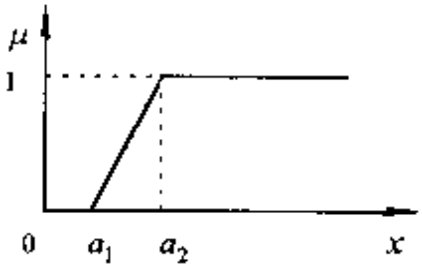
(2) 升半梯形分布

$$\mu_A(x)=\begin{cases}0, & x\leqslant a_1\\ \dfrac{x-a_1}{a_2-a_1}, & a_1<x\leqslant a_2\\ 1, & a_2<x\end{cases}$$

见附图 5.2.



附图 5.1　降半梯形分布

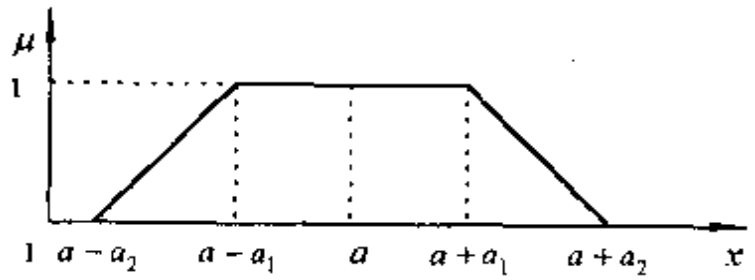


附图 5.2　升半梯形分布

(3) 梯形分布

$$\mu_A(x)=\begin{cases}0, & x\leqslant a-a_2\\ \dfrac{a_2+x-a}{a_2-a_1}, & a-a_2<x<a-a_1\\ 1, & a-a_1<x\leqslant a+a_1\\ \dfrac{a_2+x+a}{a_2-a_1}, & a+a_1<x\leqslant a+a_2\\ 0, & a+a_2<x\end{cases}$$

见附图 5.3.



附图 5.3　梯形分布

(4) 语义征兆

对于语义征兆分为 8 档,即很严重、比较严重、严重、一般、轻微、比较轻微、很轻微、不存在,分布对应隶属度{0.95,0.80,0.65,0.50,0.35,0.20,0.05,0}.

诊断模型四种:模型Ⅰ(∧,∨)、模型Ⅱ(·,∨)、模型Ⅲ(∧,⊕)及模型Ⅳ(·,⊕).

诊断原则两种:最大隶属度原则和阈值原则.

具体的程序编码为

```
/*****************************************/
/*              模糊诊断程序             */
/*              文件名:FLFD.C            */
/*****************************************/
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<dos.h>
```

```
#include<string.h>
#include<conio.h>
#include<time.h>

#define N 6              /*征兆数为 6*/
#define M 5              /*故障数为 5*/
#define THD 0.5          /*阈值置为 0.5*/

float power[N][M];  /*定义诊断权矩阵*/

float mem_dgr(void)
/*隶属度子函数*/
/*本程序具有四种模糊化方法、四种模糊算子、两种判决*/
{   int choice,choice1;
    float a,mm_dgr,real_val,a1,a2;
    printf("请选择隶属度分布函数\n");
    printf("1   降半梯形分布\n");
    printf("2   升半梯形分布\n");
    printf("3   梯形分布\n");
    printf("4   语言变量\n");
    for(;;)
    {     printf("   ......请选择 1—4\n");
          scanf("%d",&choice);
          if(choice<5) break;
          printf("输入越界.....,请选择 1—4");
    }
    if(choice1 =4)
    {     printf("请输入您的实际值");
          scanf("%f",&real_val);
          mm_dgr=0;
    }
    switch(choice)
    {     case  4:printf("请选择征兆存在程度\n");
```

```
        printf("1   很严重\n");
        printf("2   严重\n");
        printf("3   比较严重\n");
        printf("4   一般\n");
        printf("5   比较轻微\n");
        printf("6   轻微\n");
        printf("7   很轻微\n");
        printf("8   不存在\n");
        printf("    请选择 1—8\n");
        scanf("%d",&choice1);
        switch(choice1)
        {   case 1:mm_dgr=0.95;break;
            case 2:mm_dgr=0.80;break;
            case 3:mm_dgr=0.65;break;
            case 4:mm_dgr=0.50;break;
            case 5:mm_dgr=0.35;break;
            case 6:mm_dgr=0.20;break;
            case 7:mm_dgr=0.05;break;
            case 8:mm_dgr=0.00;break;
        }
        break;
case    1:printf("请输入隶属度为 1 的边界 a1=");
        scanf("%f",&a1);
        printf("\n 请输入隶属度为 0 的边界 a2=");
        scanf("%f",&a2);
        if(real_val<=a2) mm_dgr=(a2-real_val)/(a2-a1);
        if(real_val<a1)  mm_dgr=1; break;
case    2:printf("\n 请输入边界 a1=");
        scanf("%f",&a1);
        printf("\n 请输入边界 a2=");
        scanf("%f",&a2);
        if(real_val>a1) mm_dgr=(real_val-a1)/(a2-a1);
        if(real_val>a1)  mm_dgr=1; break;
```

```
    case  3:printf("\n 请输入边界 a=");
          scanf("%f",&a);
          printf("\n 请输入边界 a1=");
          scanf("%f",&a1);
          printf("\n 请输入隶属度为 0 的边界 a2=");
          scanf("%f",&a2);
          if(real_val<=a-a1&&real_val>a-a2)
               mm_dgr=(a2+real_val-a)/(a2-a1);
          if(real_val<=a+a2&&real_val>a+a1)
               mm_dgr=(a2-real_val+a)/(a2-a1);
          if(real_val<=a+a1&&real_val>a-a1)
               mm_dgr=1;
    }
    return (mm_dgr);
}

void diag(void)
/* 诊断子程序 */
{
    int choice,i,j,no;
    float symp[N],flt[M],fault;
    for(;;)
  {     printf("选择诊断算法\n");
        printf("1  诊断模型 Ⅰ\n");
        printf("2  诊断模型 Ⅱ\n");
        printf("3  诊断模型 Ⅲ\n");
        printf("4  诊断模型 Ⅳ\n");
        printf("5  退出诊断\n");
        printf("   您选择 1—5");
        scanf("%d",&choice);
        if(choice==5) exit(0);
        printf("\n 请输入各征兆\n");
        for(i=0;i<N;i++)
```

```
{   printf("请输入第%2d个征兆的情况\n",i+1);
    symp[i]=mem_dgr();
    printf("征兆%2d的隶属度为",i+1);
    printf("%f\n",symp[i]);
}
switch(choice)
{   case 1:for(j=0;j<M;j++)
    {       fault=0;
            for(i=0;i<N;i++)
            fault=max(fault,min(power[i][j],symp[i]));
            flt[j]=fault;
    }
    break;
    case 2:for(j=0;j<M;j++)
    {       fault=0;
            for(i=0;i<N;i++)
            fault=max(fault,power[i][j]*symp[i]);
            flt[j]=fault;
    }
    break;
    case 3:for(j=0;j<M;j++)
    {       fault=0;
            for(i=0;i<N;i++)
            fault=fault+min(power[i][j],symp[i]);
            flt[j]=min(1,fault);
    }
    break;
    case 4:for(j=0;j<M;j++)
    {       fault=0;
            for(i=0;i<N;i++)
            fault=fault+power[i][j]*symp[i];
            flt[j]=min(fault,1);
    }
```

```
        }
        printf("\n 请选择诊断原则\n");
        printf("1   最大隶属度原则\n");
        printf("2   阈值原则\n");
        printf("......您选择 1 or 2");
        scanf("%d",&choice);
        switch(choice)
        {   case 1:fault=flt[0];
                   no=0;
                   for(i=1;i<M;i++)
                   {   if(flt[i]>fault)
                       {   fault=flt[i];
                           no=i;
                   }   }
                   printf("诊断对象具有故障%d",no);
                   break;
            case 2:for(i=0;i<M;i++)
                       if(flt[i]>THD) printf("诊断对象具有故障%d",no);
        }
    }
}

main()
{
   FILE *fp;
   char **endptr;
   char filename[12],cpower[5];
   int i,j;
   printf("*************************************\n");
   printf("*          欢迎进入模糊诊断系统          *\n");
   printf("*************************************\n");
   printf("     请输入诊断权矩阵数据文件名");
   scanf("%s",filename);
```

```
    if((fp=fopen(filename,"r"))==NULL)
    {   printf("不能打开该文件\n");
        exit(0);
    }
    for(i=0;i<N;i++)
        for(j=0;j<M;j++)
        {   fread(cpower,5,1,fp);
            power[i][j]=strtod(cpower,endptr);
            printf("%f\n",power[i][j]);
        }
    fclose(fp);
    diag();
    return 0;
}
```

附录 5.2　Max-min 型 FAM 诊断程序

本系统采用本章 5.4.4 节介绍的学习算法，网络的输入为六种模拟征兆，输出为两种模拟故障，程序仿真表明，该系统对所有的模糊对都能准确地联想.

```
/*************************************************/
/*    文件名                FAM.C                */
/*    本程序采用 Max-min 型模糊联想记忆 FAM        */
/*              故障诊断仿真程序                   */
/*************************************************/

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<dos.h>
#include<string.h>
#include<conio.h>

#define IN 6                    /*输入层神经元数目*/
#define ON 2                    /*输出层神经元数目*/
#define SIZE 7                  /*学习样本数目*/
#define EITA 0.7                /*学习步长*/

static double wgh[ON][IN];      /*输入层到隐出层的权值*/
static double cause[IN];        /*征兆*/

/*   故障原因中文描述   */
static char Fault[ON][6]={"故障 1","故障 2"};
```

```
/*   故障征兆中文描述   */
static char in_fault[IN][6]={"征兆 1","征兆 2","征兆 3","征兆 4","征兆
                             5","征兆 6"};

/*   学习样本   */
struct samples
{
    double in_sign[IN];        /*输入信号*/
    double tch_sign[ON];       /*教师信号*/
} samp[SIZE]=
    {{0.7,0.2,0.4,0.6,0.3,0.8,0.6,0.4},
     {0.9,0.4,0.4,1.0,0.7,0.0,0.9,0.6},
     {0.3,0.6,0.5,0.5,0.2,0.9,0.5,0.5},
     {0.1,0.7,0.4,0.7,0.7,0.7,0.7,0.6},
     {0.3,0.6,0.4,0.8,0.2,0.4,0.8,0.5},
     {0.7,0.9,0.6,0.1,0.7,0.4,0.7,0.6},
     {0.8,0.1,0.6,0.5,0.4,0.3,0.6,0.4}};

void train(void)                    /*学习子程序*/
{  int i,j,p,mark=0;
   double real_sgn[ON],delta[ON];
   char file_nm[10];
   FILE *data_fl;
   printf("请输入保存权值数据的数据库文件名:");
   scanf("%s",file_nm);
   if((data_fl=fopen(file_nm,"wb"))==NULL)
       {printf("不能打开数据库文件\n");
        exit(0);
       }
   for(i=0;i<IN;i++)                /*赋初值*/
       for(j=0;j<ON;j++)
         wgh[j][i]=1;
   for(p=0;p<SIZE;p++)
```

```
        {for(;;)
          {for(j=0;j<ON;j++)      /* 计算实际输出 */
            {real_sgn[j]=0;
             for(i=0;i<IN;i++)
               real_sgn[j]=max(real_sgn[j],min(wgh[j][i],
                          samp[p].in_sign[i]));
            }
           for(j=0;j<ON;j++)
             delta[j]=real_sgn[j]-samp[p].tch_sign[j];
           for(j=0;j<ON;j++)      /* 调权 */
             for(i=0;i<IN;i++)
              {if(min(wgh[j][i],samp[p].in_sign[i])>samp[p].tch_
                    sign[j])
                 {mark=1;
                  wgh[j][i]=wgh[j][i]-EITA*delta[j];
                 }
               else wgh[j][i]=wgh[j][i];
              }
           if(mark==0) break;
           mark=0;
          }  }
    /* 保存权值数据 */
    for(j=0;j<ON;j++)
        {for(i=0;i<IN;i++)
         { fwrite(&wgh[j][i],8,1,data_fl);
           printf("%7.4f",wgh[j][i]);}
         printf("\n");
        }
    fclose(data_fl);
}

void result(void)
{  int i,j,l;
```

```
    double real  sgn[ON];
    static char Degree[2][4]={"大","中"};
    char FLT  DGR[12];
    /*计算输出*/
    for(j=0;j<ON;j++)
        {real_sgn[j]=0;
        for(i=0;i<IN;i++)
          real_sgn [j]= max(real_sgn[j],min(wgh[j][i],cause[i]));
        printf("%2d 的输出为%7.4f",j+1,real_sgn[j]);
        }
    /*判断并输出结果*/
    for(j=0;j<ON;j++)
        {if(real_sgn[j]>0.7)          l=0;
        else if(real_sgn[j]>0.5)      l=1;
        if(real_sgn[j]>0.5)
        {strcpy(FLT_DGR,Degree[l]);
        strcat(FLT_DGR,Fault[j]);
        printf("%s\n",FLT_DGR);
}       }    }

void diagnose(void)
{   int i,j,choice1;
    FILE  *know_fl;
    char know_nm[10];
    printf("请输入知识库文件名");
    scanf("%s",know_nm);
    if((know_fl=fopen(know_nm,"rb"))==NULL)
        {printf("不能打开知识库文件\n");
        exit(0);
        }
    /*读取 FAM 权值数据*/
    rewind(know_fl);
    for(j=0;j<ON;j++)
```

```
        for (i=0;i<IN;i++)
            fread(&wgh[j][i],8,1,know_fl);
    fclose(know_fl);
    while(1)
        {printf("请依次输入各种故障征兆:\n");
        for(i=0;i<IN;i++)
            {printf("\n%s 的数据是: ",in_fault[i]);
             scanf("%lf",&cause[i]);
            }
            printf("您输入例子的故障是: ");
            result();
            printf("\n ******* 选择功能 ******* \n \n");
            printf("              1   继续诊断              \n");
            printf("              2   退出诊断              \n");
            printf("      ....您选择                        \n");
            scanf("%d",&choice1);
        if(choice1!=1) break;
}       }

main()                  /* 主程序 */
{
    int choice;
    printf(" ***************************************** \n");
    printf(" *        欢迎进入 Max-min 型 FAM 诊断系统  * \n");
    printf(" *                  功能选择                * \n");
    printf(" *            1       FAM 网络学习          * \n");
    printf(" *            2       FAM 故障诊断          * \n");
    printf(" *            3       退出系统              * \n");
    printf(" *    ........您选择(1—3)                   * \n");
    printf(" ***************************************** \n");
    for(;;)
        {scanf("%d",&choice);
        if(choice>0&&choice<4) break;
```

```
        printf("输入越界,重输");
        }
    switch(choice)
        {case 1:train();
                break;
         case 2:diagnose();
                break;
         case 3:exit(0);
        }
    return(0);
}
```

第六章　智能故障诊断技术的发展和展望

人工智能在故障诊断领域中的应用，实现了基于人类专家经验知识的设备故障诊断技术，并且将其提高到一个新的水平——智能化诊断水平. 目前，智能故障诊断技术正处于研究热点之中，本章将讨论智能故障诊断技术的发展趋势，展望其发展前景.

6.1　智能故障诊断技术发展面临的问题和解决的途径

智能故障诊断技术的研究目前主要从两个方面展开：基于知识的智能故障诊断技术的研究和基于神经网络的智能故障诊断技术的研究. 下面对它们的研究状况及发展前景进行讨论和分析.

6.1.1　基于知识的智能故障诊断技术的研究

基于知识的智能故障诊断技术是设备诊断领域中最为引人注目的发展方向之一，也是研究最多、应用最广的一类智能诊断技术. 它大致经历了两个发展阶段：基于浅知识(人类专家的经验知识)的第一代故障诊断专家系统和基于深知识(诊断对象的模型知识)的第二代故障诊断专家系统. 近期出现的混合结构的专家系统，是将上述两种方法结合使用，互补不足，相得益彰.

1. 基于浅知识的故障诊断系统

基于浅知识的故障诊断系统是以领域专家和操作者的启发性经验知识为核心，通过演绎推理或产生式推理来获取诊断结果，其目的是寻找一个故障集合使之能对一个给定的征兆(包括存在的和缺席的)集合产生的原因做出最佳解释. 在其推理过程中，一般

要用到两类知识，一类是表达故障与征兆之间联系的因果性符号知识；一类是反映故障与征兆间因果关系成立程度的数值性知识（模糊性度量、信任度、可能度等）.长久以来，人们提出了把这两类知识结合在一起的各种推理方法，如 Shortliffe 和 Buchanan 提出并成功应用于医疗专家系统 MYCIN 中的确定性因子方法；Duda 等提出并应用于地质矿床勘探专家系统 PROSPECTOR 中的主观贝叶斯概率理论；Dempster 和 Shafer 针对主观贝叶斯概率理论中先验概率难以获取的困难而提出的证据理论；在 Zadeh 的可能性理论基础上形成的近似推理方法；Reggia 的节约覆盖集理论以及 Peng 提出的基于节约覆盖集理论的概率因果诊断方法等.

基于浅知识的诊断推理具有知识表达直观、形式统一、模块性强、推理速度快等优点.但是这种方法具有较大的局限性，如知识集不完备，对没有考虑到的情况系统容易陷入困境.

2. 基于模型知识的故障诊断系统

基于模型知识（深知识）的故障诊断系统要求诊断对象的每一个环节具有明确的输入输出表达关系，诊断时首先通过诊断对象的实际输出与期望输出之间的不一致，生成引起这种不一致的原因集合，然后根据诊断对象领域中的第一定律知识（具有明确科学依据的知识）及其内部特定的约束关系，采用一定的算法，找出可能的故障源.

基于模型知识的诊断推理在知识的表达与组织上比基于经验知识的诊断推理具有更大的优越性：知识获取方便，维护简单，易于保证知识库的一致性和完备性.但是搜索空间大，推理速度慢.近年来发展了基于经验知识和模型知识相结合的诊断推理方法，如 Gallanti 和 Fink 提出的集成诊断模型；Peng 的层次因果模型等.

3. 现有知识处理诊断系统的局限性

十几年的研究和实践表示，现有的知识处理系统的确在某些

方面具备智能系统所拥有的能力，在设备故障诊断领域中取得了相当程度的成功应用.但是当代知识处理系统依然存在许多明显的局限性，主要表现在以下几个方面：

(1) 知识获取的瓶颈问题

著名人工智能学者 A. Barr 和 A. Feigenbaum 曾精辟地指出："专家系统的性能水平主要是它拥有的知识数量和质量的函数."与领域专家解决问题的能力相类似，一个专家系统占有的知识越多、质量越高，它解决问题的能力就越强.然而在第一代基于知识的故障诊断专家系统中，知识的获取都是通过知识工程师与领域专家对话，将领域专家的知识总结为规则加入知识库中，这种知识获取是间接的，不但费时费力，而且效率低.另外，领域专家的某些经验知识往往只能意会，不能言传，很难用一定的规则来描述.因此，要把领域专家的经验知识以适当的方式组织成高质量的知识库是很困难的，它已成为开发研制基于知识的故障诊断专家系统的瓶颈问题.

(2) 自适应能力差

当代知识处理诊断系统通常是以专业领域的经验知识为基础进行问题求解，对那些超出系统所拥有的专业领域经验知识的问题，即使问题所涉及到的知识只与现有专业领域知识有细微的偏差，诊断系统就得不出结论，甚至会得出错误的结论.然而在工程实际中，即使是同一型号的设备，由于其安装、检修、负载等因素的影响及设备运行环境的不同，故障模式会存在较大的差别.此时专业领域知识的适用性与可靠性会大大降低.因此，如果诊断系统只具有浅知识，缺乏深知识和原理性知识，诊断系统对诊断结果的自适应性就会变得很差，使系统对诊断问题的求解能力变得脆弱.

(3) 学习能力差

人类专家能够在不断的实践中总结经验(成功或失败)或从专业领域本身的发展中学习新的知识，当代知识处理诊断系统大多不具备这一特性.系统在其运行过程中不能从诊断成败中吸取经验教训，从诊断过的实例中自动学习新的知识、修正并更新原有知

识库中的知识.系统的智能水平取决于系统最初所具备的知识,缺乏学习能力,则限制了系统性能的自我完善、发展和提高.

(4) 实时性差

著名人工智能先驱 Newell 说过:"符号和搜索是人工智能的中心."知识处理系统正是基于符号处理方法,而在符号处理中,问题求解的过程是一个在解空间的搜索过程,对于复杂的诊断对象,搜索空间大,速度慢,难以实现实时诊断的要求.当今许多电子、机械设备要求系统对故障实时检测、实时诊断,不允许得到解的时间很长,这就需要提高现有系统对故障处理的快速反应能力.

4. 知识处理诊断系统的发展方向

上面分析了当代知识处理诊断系统存在的一些局限性,人们正在积极寻求解决这些局限性的方法.例如:采取机器学习的方法解决知识获取的瓶颈问题;在浅层知识的基础上增加深层知识,以增强系统的适应性和强壮性;采取多种知识表示方法及多种求解策略,提高系统的灵活性;采用并行处理和分布式系统结构,提高其实时性等.

知识处理诊断系统的发展方向可以大致归纳为以下几方面:

(1) 由基于规则的系统到基于混合模型的系统

基于规则的诊断系统比较成熟,许多实际运转的诊断系统都是基于规则的.为了吸收各种方法的优点,可综合多种方法,如基于规则、基于功能和深层知识模型的方法,甚至可以采用人工神经网络和模式识别等方法,以实现多形式、多深度诊断知识的推理,或者也可以在诊断的不同阶段采用不同的方法.

(2) 由领域专家提供知识到机器学习

知识获取是建造故障诊断专家系统的瓶颈问题,尤其是知识的自动获取一直是专家系统研究中的难点.解决知识获取问题的途径是机器学习.机器学习研究的主要目标是让机器自身具有获取知识的能力,使其能在实际工作中不断总结成功和失败的经验教训,对知识库中的知识自动进行调整和修改,以丰富、完善系统

的知识.机器学习是提高故障诊断系统智能的主要途径,一旦诊断系统具有学习能力,它就能从环境的变化中学习新知识,不断实现自我完善.

(3) 由非实时诊断到实时诊断

实时诊断就是强调在线数据处理与在线诊断推理.在很多情况下,诊断速度是第一位的,系统要在有限的时间内,自行决策,给出设备的运行工况或可能存在的问题.要达到诊断的实时性,为此要寻求合理的诊断方法,设计合理的诊断软件结构,实行分级进程推理,尽可能提高硬件的处理速度.

(4) 由单机诊断到分布式全系统诊断

现有的设备诊断技术几乎都是面向单台、单机或单一类型的设备的.然而在现代化工业生产中,大型成套设备的使用越来越多,它们在生产中起着关键作用.这些成套设备有一个共同特点,那就是其结构和功能是分布式和多层次的,比如计算机集成制造系统、空间飞行器系统、大型汽轮发电机组等,大多实行主-从机分层控制,并且具有一定的分散度.这一特点决定诊断系统也应是分布式和多层次的.诊断系统由全局(系统级)诊断系统和子诊断系统组成.全局诊断系统负责诊断任务的管理,包括将总体任务分解成子任务和向各子诊断系统分配子诊断任务,这些任务往往是相互耦合的.诊断子任务完成之后,通过对各子诊断系统结论的综合,最终给出全系统的诊断结论.大型成套设备的分布式监视与诊断问题的求解将是基于知识的诊断推理中的一个新的研究重点和发展方向.它具有如下特点:

1) 诊断信息量大,类型多.以数值或图象信息为主,也包括符号信息,因此,相应地也就需要多种数据处理与诊断推理方法的联合.

2) 与设备运行实行主-从机分层控制一样,诊断系统也应是一个分布式的分层控制系统.各层监视与诊断单元之间的关系,如受控性、依赖性等,则视具体应用领域的特征来定.一般来说,各低层监视与诊断单元具有一定的智能水平和决策能力,同时,它在信息的处理方法、监视与诊断目标、对设备的决策干预等方面又受控

于上层诊断单元.

3）大量的诊断信息主要来源于各种传感器，因此，传感器工作的可靠性与信息传输的可靠性必须得到保证.传感器的自诊断问题受到人们的广泛重视.目前已有一些诊断系统具备诊断传感器的能力.传感器的诊断一般是通过自检、冗余和逻辑诊断来实现.在诊断过程中先诊断传感器的故障，后诊断设备故障.

（5）由单一推理控制策略到混合推理控制策略

推理控制策略直接影响专家系统的推理效率.单一的问题求解策略也是当前基于知识的诊断专家系统的一个致命弱点.单一的问题求解策略很容易使系统在推理过程中出现“匹配冲突”、“组合爆炸”和“无穷递归”等问题，从而导致推理速度慢，系统性能低.

目前知识处理系统常用的推理控制策略包括：数据驱动控制和目标驱动控制.前者的主要缺点是盲目推理，后者的主要缺点是盲目选择目标.一个有效的办法是综合二者的优点，通过数据驱动选择目标，通过目标驱动求解该目标，这就形成了双向混合控制策略的基本思想.目前已提出三种典型的双向混合控制模式：

第一种是双向交替控制策略：首先由用户提供尽可能多的事实，调用数据驱动策略，从已知事实演绎出部分结果，然后根据顶层目标，调用目标驱动控制策略，试图证实该目标.为此再收集事实重复上述过程，直到某个顶层目标的权超过阈值（已证实）或收集不到新的事实（失败）为止.

第二种是双向同时控制策略：根据原始数据进行正向演绎推理，但不希望推理一直到达目标为止，同时从目标出发进行反向推理，也不希望该推理一直到达原始证据，而是希望两种推理在原始证据和目标之间的某处“接合”起来.

第三种是生成与测试——广义混合控制策略：其基本思想是根据部分约束条件，生成一批目标，然后再利用全部约束条件逐个“测试”目标.

总之，今后基于知识的故障诊断专家系统的发展除了在知识表示、推理等各部分功能更加完备之外，在机器学习、运行效率和

实时性等方面将会有较大的突破，这是决定一个智能诊断系统的有效性和广泛应用性的关键.

6.1.2 基于神经网络的智能故障诊断技术的研究

利用神经网络的学习功能、联想记忆功能、分布式并行信息处理功能以及极强的非线性映射能力，解决诊断系统的知识表示、获取和并行推理等问题，为智能诊断技术的发展开辟了新的途径. 早在 1988 年 Hoskins 就详细论证了神经网络在故障诊断领域的巨大潜力，Venkatasubramanian 也具体地描述了一些故障诊断的神经网络方法，Gallant 还提出了一种基于联结主义机制专家系统的自动产生策略. 总的来看，神经网络在设备故障诊断领域的应用研究主要集中在三个方面：一是从模式识别角度应用神经网络作为分类器进行故障诊断；二是从预测角度应用神经网络作为动态预测模型进行故障预测；三是从知识处理角度建立基于神经网络的诊断专家系统.

1. 模式识别的故障诊断神经网络

大家知道，状态监测的任务是使机器系统不偏离正常功能，并预防功能失败，在监测的基础上进行诊断；而当系统一旦偏离正常功能，则必须进一步分析故障产生的原因，这时的工作可理解为是故障诊断. 如果事先已对机器可能发生的故障模式进行分类，那么诊断问题就转换为把机器的现行工作状态归入那一类的问题. 因此，故障诊断实质上是一类模式分类和识别的问题.

在传统的模式识别技术中，模式分类的基本方法是利用判别函数来划分每一个类别. 如果模式样本特征空间为 N 维欧氏空间，模式分类属于 M 类，则在数学上模式分类问题就归结为如何定义诸超平面方程把 N 维欧氏空间最佳分割为 M 个决策区域的问题. 对线性不可分的复杂的决策区域，则要求较为复杂的判别函数，并且在许多情况下，由于不容易得到全面的典型参考模式样本，常采用概率模型，在具有输入模式先验概率知识的前提下，选

取适宜的判别函数形式，以提高识别分类的性能. 如何选择有效的判别函数形式，以及在识别过程中如何对判别函数的有关参数进行修正，对于传统的模式识别技术来说，并不是一件容易的事.

人工神经网络作为一种自适应的模式识别技术并不需要预先给出关于模式的先验知识和判别函数，它通过自身的学习机制自动形成所要求的决策区域. 网络的特性由其拓扑结构、节点特性、学习或训练规则所决定，它能充分利用状态信息，对来自不同状态的信息逐一训练以获得某种映射关系，而且网络可连续学习，当环境改变，这种映射关系可以自适应，以求对对象的进一步逼近.

例如，使用来自机器不同状态的振动信号，通过特征选择，找出对于故障反映最敏感的特征信号作为神经网络的输入向量，建立故障模式训练样本集，对网络进行训练；当网络训练完毕，对于每一个新输入的状态信息，网络将迅速给出分类结果. 图 6.1 示出了基于神经网络故障分类诊断的一般流程图.

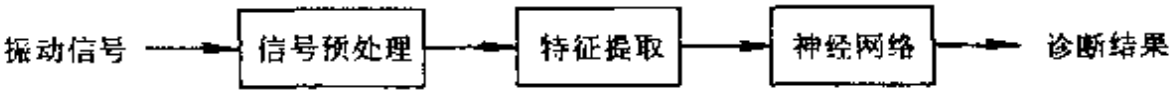


图 6.1　神经网络故障分类诊断

2. 故障预测的神经网络

故障预测的神经网络主要以两种方式实现预测功能，一是以神经网络(如 BP 网络)作为函数逼近器，对机组工况的某参数进行拟合预测；二是考虑输入输出间的动态关系，用带馈连接的动态神经网络对过程或工况参数建立动态模型而进行故障预测.

Funahashi 首先证明了单隐层的感知器网络能以任意精度逼近任一连续映射，表明多层感知器是一个理想的函数逼近器. 但是目前应用较广泛的仍然是基于多层的前馈网络(如 BP 网络). 从系统辨识的角度看，前馈网络只代表了一类可通过代数方程描述的静态映射，且只适用于静态预测，将其应用于设备动态行为的建模与预测则受到很大的限制.

动态神经网络的预测是一个对动态时序建模的过程.人们已经提出了许多有效的网络结构,其中包括全连接网络以及各种具有局部信息反馈结构的网络模型等,这些网络一个共同的特点是其输出不仅取决于当前输入,还依赖于网络过去的状态,网络本身具有相应的动态结构,其预测是动态预测,因而在实际的非线性动态系统的建模和预测中得到了成功的应用.但是动态神经网络在结构上远比前馈网络结构复杂,其样本训练也较困难,因此合理地降低网络结构的复杂性,简化网络的学习算法将是实际应用中需要研究解决的问题.

3. 神经网络故障诊断专家系统

第四章讨论了神经网络与专家系统的结合主要有两种策略:一是将专家系统构成神经网络,把传统专家系统的基于符号的推理变成基于数值运算的推理,以提高专家系统的执行效率并利用其学习能力解决专家系统的学习问题;二是将神经网络视为一类知识源的表达与处理模型,与其它知识表达模型一起去表达领域专家的知识.总之,基于神经网络的故障诊断专家系统是一类新的知识表达体系,与传统的专家系统的高层逻辑模型不同,它是一种低层数值模型,信息处理是通过大量称之为节点的简单处理单元之间的相互作用而进行的.由于它的分布式信息保持方式,为专家知识的获取和表达以及推理提供了全新的方式.通过对经验样本的学习,将专家知识以权值和阈值的形式存储在网络中,并且利用网络的信息保持性来完成不精确诊断推理,较好地模拟了专家凭经验、直觉而不是复杂的计算的推理过程.

综上所述,与传统的诊断方法及专家系统相比,神经网络在故障诊断领域中的应用显示了明显的优越性,但也存在以下一些困难:

(1) 训练样本获取的困难性

神经网络故障诊断是建立在大量的故障样本训练基础之上,系统性能受到所选训练样本的数量及其分布情况的限制.如果样

本选择不当，特别在训练样本少，样本分布不均匀的情况下，很难有良好的诊断能力.然而，在设备诊断中，往往很难得到大量的故障样本，且在所能得到的有限故障样本中，其分布也是很不均匀的.

(2) 忽视了领域专家的诊断经验知识

神经网络利用知识和表达知识的方式单一，通常的神经网络仅仅通过典型实例来获取诊断知识，忽视了领域专家在长期实践中所积累的宝贵经验.而设备故障诊断尤其是大型设备的故障诊断是一项经验性技术，忽视了专家经验，基于单一的典型实例必将导致神经网络故障诊断能力的下降.

(3) 权重形式的知识表达方式难以理解

在专家系统中，人们总是希望系统能对其所得出的结论作出合适的解释.当然，在神经网络故障诊断专家系统中，也希望其具有解释能力.然而由于神经网络所学习到的知识是以权重形式分布在网络之中，这种隐式的数值表示不易被人们理解，且对诊断结果缺乏解释能力.对用户来说，整个诊断系统是一个"黑箱"，不具备基于知识的故障诊断专家系统的透明特性，而在工程实际中，对诊断结果不透明，缺乏可理解性，用户是很难接受的.

目前，为了提高神经网络在实用中的学习和诊断性能，主要是从神经网络模型本身的改进和模块化神经网络诊断策略两个方面开展研究.

(1) 模块化神经网络

模块化神经网络的思想就是用多个相对简单的网络模块以一定的关系协同运作来处理一个大规模问题，这一策略因其在应用中的有效性而越来越受到人们的重视.相对于非模块化神经网络而言，模块化神经网络具有以下优点：

1) 模块化神经网络中各网络模块要比相应非模块化神经网络简单得多，因此，各网络模块易于构造，且其学习性能和泛化特性易于得到保证；

2) 在实际应用中，一般可独立并行对待各网络(网络训练、再

训练或网络结构的再调整等),使得模块化神经网络更具应用的灵活性和现场的适应性;

3) 模块化神经网络的提出更易于解释,因为各网络模块的任务明确,功能确定,且各网络模块间的关系清楚,使得整个网络的工作过程更为明朗;

4) 模块化神经网络思想符合计算机软硬件设计的发展趋势,便于软硬件的实现.

(2) 神经网络模型的改进

任何一种故障诊断模型,对其最基本的要求应是能对下面三个问题给出一个明确的结果,且诊断行为应具有较强的鲁棒性.

1) 正常与否,即诊断模型能判断是否有故障存在;

2) 若有故障存在且为已知(已经学习过的)故障,则能给出确切的诊断结果;

3) 若有故障存在但为未知(未经学习过的)故障,则能辨别其为未知故障,即诊断模型应具有"拒绝"能力.

文献[60]分析了 BP 网络模型在故障诊断应用中的局限性,提出了一种新的高阶网络模型——椭球单元网络模型在故障诊断中的应用.椭球单元网络同 BP 网络一样,都是属于前馈网络的结构.二者不同之处是 BP 网络的神经元为线性单元,单元的激励函数为线性函数:

$$L(X) = W^T X$$

式中 W 为权重向量,X 为输入向量,经 Sigmoid 函数变化,单元输出为

$$O_L(X) = \frac{1}{1 + \exp(L(X))}$$

$O_L(X)$在输入空间形成的是超平面,所以,BP 网络是以超平面划分决策空间的,而作为故障诊断的分类器则存在缺陷,如图 6.2 所示,对已知的三类故障的分类,在某些区域 BP 网络缺乏正确的诊断能力或完全失去诊断能力.比如对于区域 D 中的样本,关于三类故障的网络输出节点均会给出低的输出值,表示该样本均不属

于各故障类，而对于区域 E 中的样本，网络会给出既属于故障 1 又属于故障 3 的诊断结果. 对于区域 F 和区域 G 也存在同样问题.

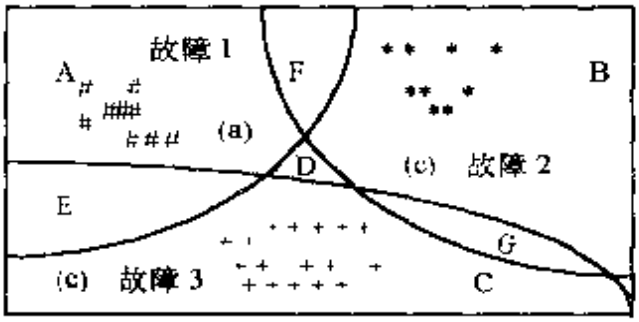


图 6.2 BP 网络对决策空间的划分

椭球单元网络的单元激励函数为

$$E(X) = -(X - M)^T (D^T D)^{-1} (X - M) + 1$$

式中 M 为椭球中心，D 为对角矩阵，$E(X)=0$ 定义一个各主轴平行相应坐标轴的超椭球，代入 Sigmoid 函数，单元输出为

$$O_E(X) = \frac{1}{1 + \exp(-E(X))}$$

$O_E(X)$ 在输入空间形成超椭球，所以，椭球单元网络经训练后以超椭球划分决策空间，形成封闭有界的决策区域，如图 6.3 所示，同 BP 网络相比，椭球单元网络更直观清晰地反映了故障样本在决策空间中的分布，其诊断结果更具有真实性. 因此，椭球单元网络在故障诊断领域中显示出明显的优越性，有着极大的应用潜力.

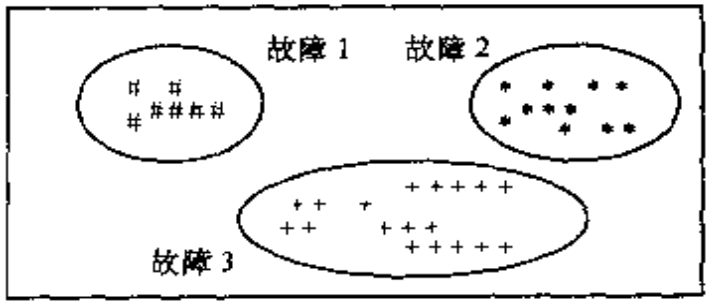


图 6.3 椭球单元网络对决策空间的划分

6.2 智能故障诊断技术未来发展相关的新技术

6.2.1 机器学习

1. 机器学习问题的提出

何谓机器学习？目前众说纷纭，我们认为，学习是一个有特定目的的知识获取过程，从学习的内在行为看，是从未知到知的过程，是知识增加和积累经验的过程；从外在表现看，学习是使系统改进性能、适应环境，使其在下一次完成同样的或类似的任务时比前一次更有效.

当前机器学习已成为人工智能的核心，它的应用遍及人工智能的各个领域，特别是专家系统、模式识别、计算机视觉等. 学习是一切智能行为的基础，但是现存的智能系统都普遍缺乏学习的能力. 例如，当它们遇到错误时，不能自我改正；它们不会通过经验改善自身的性能；它们不能自动产生合理的启发式方法和推理策略；而且它们的推理只限于演绎而缺乏归纳，因此它们至多能够证明已存在的事实、定理，而不能发现新的定律、定理. 为了克服这些局限性，人工智能不得不求助于机器学习. 下面我们以专家系统为例，说明机器学习在智能系统中的作用.

(1) 克服知识获取瓶颈问题

为了建立专家系统的知识库，需要领域专家与知识工程师合作，总结专家经验，然后形式化并编码输入计算机中. 由于领域知识往往很模糊，难于抽取和描述，知识获取是一项耗资费时的老大难问题. 解决这个问题的办法是通过机器学习来自动获取知识. 机器学习的一个解决方法是应用示例学习(learning from examples)，从大量的实例中自动归纳产生描述这些实例的一般规则. 一个成功的例子是 R. L. Chilausky 和 Michalski 提出的一个大豆病害诊断防治专家系统. 该系统用示例学习系统自动产生规则进行诊断，比由专家给出的规则进行诊断的正确率还要高.

(2) 克服知识脆弱性问题

当推理所用的知识稍微超过知识库的范围时，系统运行就会失败. 这个问题原则上可以通过添加新的知识来解决. 但知识库的扩充又会造成不一致性和知识库的重建. 这些问题可以用渐进学习(incremental learning)来克服.

(3) 克服知识库过于庞大和非结构性问题

缺乏一个良结构知识库常常是专家系统受到指责的一个原因. 要彻底解决这一问题，根本的途径是引入工程化的思想对知识软件进行开发，机器学习中的概念聚类(conceptual clustering)和概念获取的方法可以用于结构化知识的组织和管理.

(4) 克服求解方法单一问题

单一的求解方法不能适应领域知识的多样性，不能充分反映专家系统处理问题的思想. 机器学习的一个解决方法是解释学习(explanation based learning). 解释学习通过运用论域内的有关知识，经过对某一实例的详细分析构造解释，产生规则，然后对解释进行推广得到一般性的描述.

(5) 克服系统直觉判断能力差的问题

直觉是当前人工智能和认知心理学学界对一个专家系统是否真正达到专家级水平的争论热点. 而现存专家系统即使遇到过去解决了的问题，但仍需要重新一步步推理. 通过机器学习可以把一个专家系统过去的求解经验积累起来，并加以合理的组织，使知识库做到面向目标的结构化，从而实现直觉推理的目的.

2. 机器学习的发展

机器学习的研究和人工智能同时起步，并一直起着核心的作用. 根据它的研究目标和研究方法来划分，其发展过程可分为三个阶段：

(1) 神经元模型的研究

这一阶段始于 50 年代的中期，主要是研究可适应环境的通用学习系统，即神经网络或自组织系统. 它的基本思想是：如果给系统一组刺激、一个反馈源和修改自身组织的自由度，那么系统就可

以自适应地趋向最优组织.但由于当时计算技术水平的限制,这种想法只能停留在理论上的探讨或专用实验硬件系统的构造上.在此期间有代表性的工作是1957年洛森伯勒特提出的感知器模型,它由阈值性神经元组成,试图模拟动物和人脑的感知及学习能力;最有影响的成果是塞缪尔研制的具有自学习、自组织、自适应能力的跳棋程序,经过反复训练,这个程序达到大师级水平.

1969年明斯基在麻省理工学院发表了颇有影响的论著《感知器》,对神经元模型的研究作出了悲观的论断.鉴于明斯基在人工智能界的地位和影响,以及神经元模型自身的局限性,致使对它的研究开始走向低潮.

(2) 符号学习的研究

这一阶段始于70年代的中期.当时对专家系统的研究已取得了很大成功.迫切需要解决获取知识难的问题,这一需求刺激了机器学习的发展.符号学习的特点是使用符号表示而不是数值表示,目标在于学习表达高级知识的符号描述.这一时期有代表性的工作是莫斯托夫的指导式学习、温斯顿和卡保尼尔的类比学习以及米切尔等人提出的解释学习等.

(3) 连接学习的研究

这一阶段始于80年代.当年从事神经元模型研究的学者们经过10多年的潜心研究,发现了用隐单元来计算与学习非线性函数的方法,从而克服了早期神经元模型的局限性.加之计算机硬件技术的突飞猛进的发展,使得神经网络的实现成为可能.因此,在80年代末期,神经网络迅速崛起,使机器学习进入了连接学习的研究阶段.连接学习是一种以非线性大规模并行处理为主流的神经网络的研究,该研究目前仍在继续进行之中.

在这一阶段中,符号学习的研究也全面发展并日趋成熟,应用领域不断扩大,尤其模拟人类高级思维活动符号学习是不可缺少的.连接学习和符号学习各有所长,具有较大的互补性.就目前的研究情况来看,连接主义适用于连续发音的语言识别及连续模式的识别;而符号学习在离散模式识别及专家系统的规则获取方面

有较多的应用.目前已有人在开展把符号学习与连接学习结合起来进行研究,里奇开发的混合系统就是其中的一个例子.

3. 机器学习方法的分类

正像人有各种各样的学习方法一样,机器学习方法也很多,从不同的角度进行划分,可以得到不同的分类方法.当前常用的分类方法有如下四种:按应用领域分类(如专家系统、问题求解和认知模拟等);按获取的知识表示分类(如逻辑表达式、产生式规则、框架表示等);按推理策略分类(如演绎推理和归纳推理的程度和比重);按系统性分类(综合考虑系统的知识表示、推理策略、应用领域等多种因素).其中按推理策略分类的方法一直被广泛采纳,下面仅介绍这种分类中的若干方法.

1) 机械学习(rote learning):不需要推理,直接编程或存储外部信息.这种学习方法通过记忆和评价外部环境提供的信息达到学习目的.机器学习系统要做的工作是把经过评价取得的知识存储到知识库中,当求解问题时,就从知识库中检索相应的知识直接用来求解问题.例如,设某个计算的输入是$(x_1,x_2,\cdots,x_n)$,计算后的输出是$(y_1,y_2,\cdots,y_m)$,如果经过评价得知该计算是正确的,则机械学习就把联想对:

$$[(x_1,x_2,\cdots,x_n),(y_1,y_2,\cdots,y_m)]$$

存入知识库中.当以后又要对$(x_1,x_2,\cdots,x_n)$作同样的计算时,只要直接从知识库中检索出$(y_1,y_2,\cdots,y_m)$就可以了,不需要再重新计算.

2) 示例学习(learning from examples):它是从许多示例中得出事物的特性或一般规律的一种学习方法,是从特殊到一般的学习过程,所以又称为"一般化"的学习方法.示例学习的高级形式称为发现学习,它通过学习可高度概括出一般性的"定理"及"定律".

在这种学习方式下,外部环境提供的是一组示例(正例和反例),它们具体指明了对于特定的输入应该产生什么样的输出.这些示例实际上是一组特殊的知识,每一个例子表达了仅适用于该

例子的知识.示例学习就是从这些特殊知识中归纳出适用于更大范围的一般性知识,它将覆盖所有正例并排除所有反例.例如,如果我们用一组液压系统故障作为示例,并且告诉学习系统哪一种故障是液压系统的冷却器故障,哪一种故障不是.当示例足够多时,学习系统就能概括出关于"冷却器故障"的概念模型,使自己能识别冷却器故障,并且能把它与其它故障区别开来.这一学习过程就是示例学习.

示例学习的系统模型如图 6.4 所示,其学习过程是首先从示例空间中选择一个训练示例,然后经解释将其转换为知识空间的知识,最后再从示例空间中选择更多的示例对它进行验证.

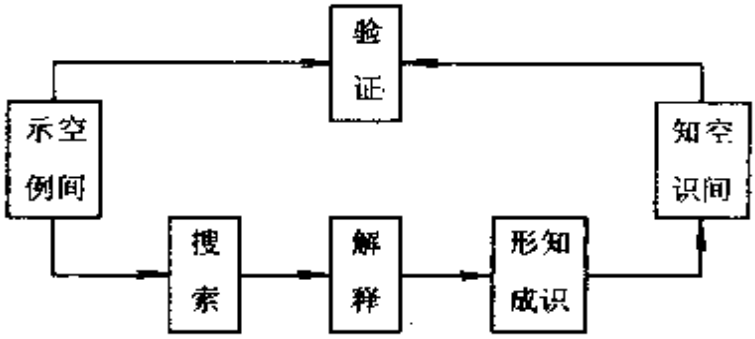


图 6.4　示例学习的系统模型

示例空间是所有可能对系统进行训练的示例集合.与示例空间有关的两个主要问题是示例的质量及示例在示例空间中的组织.关于示例的质量,要求它能准确地反映实际问题的情况,因为它是学习的基础,只有正确的示例才有可能得出正确的知识.关于示例的组织,即它们在示例空间中的排列形式,要求它便于系统对示例的搜索.

搜索就是从示例空间中查找所需的示例.为了提高搜索效率,需要设计合适的搜索算法,并把它与示例空间的组织做统筹考虑.

解释是从搜索到的示例中抽象出所需的有关信息供形成知识使用.当示例空间中的示例与知识表示形式有较大差别时,需要先将其转换为某种适合于形成知识的过渡形式.

形成知识是指把经解释得到的有关信息通过综合、归纳等过

程得出一般性的知识.

知识空间用于存放由上一步形成的知识,这些知识的正确性还有待于进一步验证.

验证的作用是检验由第4步形成的知识的正确性.为此,需要从示例空间选择大量的示例来做验证工作.如在验证中发现形成的知识不正确,则需进一步获得示例,对刚才形成的知识进行修改.重复这一过程,直到形成正确的知识为止.

示例学习方法能够从一些分散的事实中归纳抽象出一般规则,因此,它被看作专家系统中知识自动获取的一种重要手段和有力工具.对于示例学习,目前已提出一些学习方法与系统,其中影响较大的一个算法,是1977年J. R. Quinlan等人提出的基于信息熵的概念及最大信息增益机制的决策学习系统ID3算法.由于该算法比较简单,容易实现而且适于处理规模较大的学习问题,所以被广泛地用于机器学习中.目前在故障诊断专家系统中,大多采用了ID3算法以实现知识获取.但ID3算法具有如下缺点:在不重建整棵故障决策树的条件下,不能方便地对决策树做更改,也就是说,当一个新的故障模式不能被正确地分类时,就需要对树进行修改以适应这一新模式.尽管可以通过局部修补来实现这一要求,但在这种情况下,故障模式决策树却会逐渐失去它作为某些重要概念最有效的表达作用.另外,当实例中含有噪声信号时,噪声信号有可能会成为故障决策树中的一条推理途径,如存在较多的噪声信号时,故障决策树会变得非常庞大,致使该方法完全失去其实用性.因此,针对ID3算法存在的缺陷,目前国内外学者已提出了多种ID3算法的改进算法.

3) 解释学习(explanation-based learning):这是近几年来在机器学习领域中兴起的一种学习方法.解释学习可概括为:已知目标概念,该概念的一个实例和领域理论.用领域理论产生该例子满足目标概念的一个解释(证明树),并将该解释推广以适合新的例子.虽然在学习过程中解释学习也需要实例,但它与示例学习不同,示例学习是基于归纳的学习,它通过大量示例的归纳得出一般

性的知识，而解释学习只需要一个例子，在学习过程中它需要运用论域内的有关知识，经过对实例的详细分析构造解释，然后对解释进行推广得到一般性的描述.

对于解释学习，米切尔将其总结为如下的描述框架：

给定：论域知识　DT

目标概念　TC

训练实例　E

操作性准则　C

找出：满足 C 的关于 TC 的充分条件.

其中，论域知识(Domain Theory，DT)包含一组事实和规则，用于证明(解释)训练实例如何满足目标概念；目标概念(Target Concept，TC)是待学概念的一个非操作性描述；训练实例(Example，E)是为解释学习提供的一个例子，解释学习正是从该例子出发，通过运用论域知识进行证明，最终推广出目标概念的学习描述的；操作准则(Operationality Criterion，OC)则用于指明哪些测试在运行时容易判定，指导学习系统对用来描述目标的概念进行取舍.

对解释学习方法的一个扩充是采用多个实例. 在多个类似实例同时存在时，对各实例构造解释树，然后找出它的最大树，从而导出规则. 这种方法可能克服只解释一个实例时得出的规则太特别的问题，可能适用性更广.

在论域理论正确的情况下，得出的描述也一定是正确的. 实际使用中，领域理论常是不完善的. 不完善理论有三种：1)不完备理论：不能完全解释某些实例；2)不一致理论：可能有几种不一致解释；3)不可行理论：受计算时空复杂性的限制解释不能完成.

必须指出，实际领域常常无法给出一个完善领域理论，而只有在学习过程中不断完善，这样把现有解释学习方法用到一个复杂性较大的领域就有困难. 近期有些工作对解释学习方法作了改进，研究了领域理论重组和修正的策略，以弥补领域理论不完善的地方. 使用多个实例也可能对不完善领域理论有所帮助.

总之，发展和完善现有的机器学习方法，开展新的学习方法的

研究，建立实用的机器学习系统，特别是多种学习方法协同工作的集成化系统的研究，将是机器学习不断深入研究的课题，我们期待着这些方面工作有所突破.

4. 机器学习与智能系统

智能系统的核心问题就是它的学习能力. 学习不仅是任何一个计算机系统提高智能的主要途径，同时也是衡量一个系统智能程度的主要标志. 如果一个智能系统不具有学习能力，它就很难实现自我完善，一旦有了错误就会永远重复相同的错误，不能自拔，更不要说提高性能，这样的系统显然是低能的. 由此可以看出机器学习在智能系统中所处的地位及重要作用. 专家系统是一种智能系统，同样需要得到机器学习的支持.

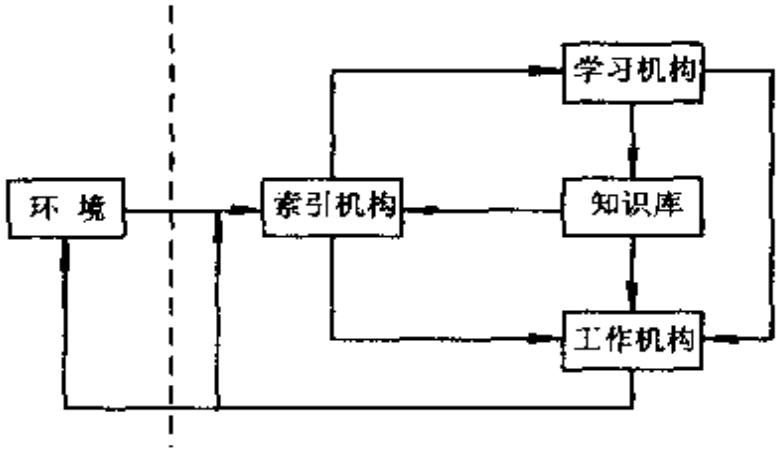


图 6.5 智能系统

智能系统可表示为图6.5所示模式. 在这个模式中，虚线右边为智能系统，左边为系统所处的环境，虚线表示系统与环境的界面，箭头表示信息流动的方向. 这里所说的环境不是指通常意义下的物理条件，而是智能系统进行学习和应用时的信息来源. 例如，当把智能系统用于专家系统的自动知识获取时，环境就是领域专家及有关的文字资料、图像等；当把它用于故障诊断系统时，环境就是大量的故障记录或一个具体设备的征兆、监测结果等. 智能系统的学习流程可归纳为

1）系统首先根据环境当前状态，由索引机构与知识库比较，判断是否为系统所经历的状态，若是，交由工作机构处理；否则转2）.

2）学习机构判断当前状态与知识库内容是否有矛盾，若有，形成问题，交由工作机构处理；若无矛盾，系统把新获得的信息归纳分类，存入知识库.

3）系统用原有知识对新获得的内容进行一定程度的假设性或联想性补充，形成问题给予解决. 如原有事物与新输入事物具有相同的结构，我们可假设新事物也具有原有事物具有的某一属性.

4）系统根据新获得的知识，扫描尚未解决的问题，若能对其有启发，则送工作机构给予解决.

智能系统应具有如下特点：

1）开放性：系统的能力应在实际使用过程中，在同环境进行信息交互的过程中不断进化.

2）结构性：系统必须具备适当的结构来记忆已学到的东西，即能够修改和完善知识表示与组织的形式.

3）有效性：系统学习到的知识应受到实践的检验，新知识必须对改善系统的行为起有益的作用.

4）工作机构与学习机构的正反馈性：学习的知识越多，工作机构的性能就得以提高，工作机构的性能提高又加快了学习速度.

5）系统的相互作用性：包括人与机器以及机器与机器间的相互作用性，特别是自然语言的理解，这是智能系统最重要的属性.

前面我们讨论了机器学习对智能系统包括专家系统的支持与帮助. 事实上，支持总是相互的，智能系统的研究与发展又为机器学习提出了新的研究课题，促进了机器学习的发展.

6.2.2 智能计算机

1. 智能计算机概述

所谓智能计算机就是用来模拟、延伸、扩展人类智能的一种新型计算机. 它与目前人们使用的冯·诺伊曼型计算机相比，无论在体系结构、运行模式还是功能上都有本质的不同. 传统的冯·诺伊

曼型计算机虽然在科学与工程计算、工程控制与现代化管理等领域取得了惊人成就，极大地推动了人类社会与科学技术的发展与进步，但与人脑相比，冯·诺伊曼型计算机存在着如下一些局限性：

1）人的记忆与思维是相随相伴而不可分的，信息是分布式存储的．但冯·诺伊曼型计算机的数据处理与存储是完全分离的，在处理器与存储器之间仅仅通过一条狭窄的通道逐字地交换数据，这就与大脑中记忆与思维合一及信息分布的方式不一致，不能满足模拟人类记忆与思维的需要．

2）人的思维过程是串行与并行共存且以并行为主的，这就使得人们不仅可以顺序地处理问题，而且可以同时应付不同的场景，把问题的不同侧面、不同因素密切地联系起来，进行多方位的综合性思考．但目前计算机的工作方式是顺序的、串行的．因此它们所能执行的算法都是串行算法．当人们需要计算机求解问题时，必须事先用某种程序设计语言编制程序，具体地指出先做什么，后做什么以及怎样做．计算机只能按程序规定的次序顺序地执行，只能完成程序指定的工作．

3）现实世界中的事物并非都是确定性的两态逻辑，更多的是多态逻辑、非确定性的、模糊的，人脑可以很自然地处理这样的事物．但目前的计算机是基于两态逻辑的，任何数据或符号在计算机内部都是用二进制表示的．因此，需要用计算机求解的问题必须先把它转化为一系列的布尔代数运算，这样才能在计算机上实现．对于不确定性信息的处理，还需要从软件上想办法，这就增加了软件设计的复杂性及难度．

4）人的思维方式除了逻辑思维外，还有非逻辑思维．人的感知过程主要是形象思维，这是单靠逻辑思维做不到的．人在处理问题时，通常是把逻辑思维与形象思维结合起来进行的．但是目前的计算机只能进行符号处理，任何要在计算机上进行处理的问题都必须表示为一串符号序列，并且还要给出处理这些符号的规则．因此它所能解决的问题仅仅局限于逻辑思维所能解决的问题范畴内，对于那些需要形象思维的问题，只能望尘莫及．

根据研究，人类大脑的左半脑和右半脑是有分工的，左半脑承担推理思考，右半脑承担感知、认识、学习. 目前的冯·诺伊曼型计算机，从模拟人类大脑的功能方面看，是属于承担推理思考能力、进行数值计算的左半脑型，它在视觉、听觉、触觉及形象思维方面的功能特别差. 智能计算机应该是研究一个包括左半脑计算机和右半脑计算机在内，更接近人类智能的完整计算机系统.

5) 人们具有多种形式的表达能力及行为能力，能对外部刺激及时作出反应，这也是目前计算机做不到的.

总之，目前的冯·诺伊曼型计算机已经为人类做出了巨大的贡献，它不仅具有非凡的计算能力，丰富的记忆能力，而且具有判断能力. 据此，我们可以说计算机是人类大脑的延伸. 但是它与人类大脑相比还相差甚远. 为了实现人类智能在计算机上的模拟、延伸、扩展，必须对其体系结构、工作模式、处理能力、接口方式等进行彻底的变革，这样造出来的计算机才能称为智能计算机. 然而，至今人类对大脑的认识还很肤浅，智能计算机不是一朝一夕就可以实现的，许多问题需要今后的科学家去探索.

智能计算机的研究目标大体上是：

1) 像人脑一样能进行自然信息/数据的处理，而不象现代(第四代)计算机那样，单纯地把信息和数据数字(二值)化后再进行计算或处理.

2) 像人脑一样处理的信息/数据既可取数字形式，也可取模拟的形式.

3) 像人脑一样采用并行、分布的模式处理信息/数据.

4) 信息/数据存储及处理的硬件均采用分布式，密集式互连的巨量处理机的构架.

为了实现上述要求，让计算机用仿人脑的方式来处理自然信息，近年来，各经济发达国家开展了人工神经网络的研究. 人工神经网络是采用大量的、比较简单的人工神经元作为基本单元，依靠单元之间复杂繁多的连接关系构成具有良好功能的网络，它是一种动态非线性系统，以分布式存储和广泛并行协同处理为特征，具

有容错、联想记忆、自学习进化等特性. 同传统的冯·诺伊曼型计算机相比，人工神经网络在人工智能和形象思维等相关领域具有明显优势，被公认为是解决下一代(第六代)智能型计算机的主要途径.

根据实现人工神经网络计算机的不同技术，可把它分为电子神经网络计算机、光学神经网络计算机和生物分子计算机等类型.

2. 电子神经网络计算机

电子神经网络计算机是指利用超大规模集成电路 VLSI 技术研制和开发的人工神经网络计算机. 因为现代半导体技术(微电子技术)比较发达，因此电子神经网络计算机的研究是目前各种人工神经网络计算机研究技术中最成熟的一个领域. 电子神经网络计算机的实现可分为三个层次：

1) 最低层次：基于当代(第四代)计算机的纯软件仿真方式；

2) 中间层次：软硬件结合的虚拟实现方式；

3) 最高层次：全硬件实现方式.

以下分别介绍这三种实现方式的近年研究和发展概况.

(1) 纯软件仿真实现的神经网络计算机

在现有通用计算机的软件环境上编制神经网络程序，就得到一台软件仿真的神经网络计算机，可在其上运行模型及算法验证. 这种方式具有成本低、使用方便、灵活性强、可以充分利用现有的软硬件资源等优点，但由于某一瞬间计算机只能模拟一个神经元的连接运算，因此把神经网络大量的并行运算又通过串行的方式来加以实现，使其从根本上失去了神经网络广泛并行处理信息的基本特征，在处理速度上远远不能满足研究和应用的要求. 尽管这样，用神经网络仿真软件进行理论研究或非实时性应用研究，仍然是当前最广泛采用的方式. 基于 Unix 或 Windows 环境下的商品神经网络仿真软件或共享软件，可在市场上购买或在 Internet 网上下载. 此外，一些著名的数值计算软件包(如 MATLAB 软件包)中，也开始把各种神经网络模型及学习算法以及相关的遗传算法

等内容包括在内，使用户获得很大的方便.

据不完全统计，目前国外已推出了几十种神经网络仿真模型（如 BP 模型等），当用户按系统约定输入自己的数据后，则能得到相应的计算结果. 有时还提供神经网络描述语言，使用户能方便地构造自己的模型. 目前世界上最大的计算机企业 IBM 公司也进入了神经网络市场. 1990 年推出了 AS400 工作站，上面安装了一个神经网络仿真开发环境，也可以在 Microsoft 公司的 Windows 3.x以及本公司的 OS/2 支持下运行，可为商业用户提供图形识别、预测建模和预报能力. 在国内，90 年代初开始即有不少单位开展了神经网络仿真软件的研制工作，并取得了一批成果，发挥了一定的效益.

(2) 软硬件结合虚拟实现的神经网络计算机

如果用 P 个物理单元去实现由 N 个神经元组成的神经网络的计算，则只要 $P<N$，我们就称它为神经网络的虚拟实现. 在虚拟实现中，若干个神经元要映射到一个物理处理单元，软件的采用增加了虚拟实现的通用性和灵活性. 根据目前技术，兼顾速度与一定程度通用性的虚拟电子神经网络计算机仍是神经网络计算机的主攻方向，它虽然没纯软件模拟那样高的灵活通用性，但有高得多的模拟速度. 纯软件仿真实现神经网络计算机视主机速度不同约可达到 10—500KIPS(IPS：每秒钟互连数)，而虚拟实现神经网络计算机一般可达到 300KIPS—6MIPS 的速度.

众所周知，典型的神经网络计算（如 BP 网络训练）和应用（如图像、语言识别）涉及到数以万亿次计的运算，对于实时及嵌入式应用来讲，用单处理器的计算机来进行神经网络计算，哪怕是采用巨型机也难以满足实际需要. 在实现过程中需要在灵活性、网络规模和网络速度等多方面作适当的选择平衡，这就形成了两条主要的开发途径. 一条是研制通用的神经网络处理机，其优点是网络结构灵活、易于修改权重、具有自学习功能. 另一条是研制专用神经网络固化芯片，以其为核心来开发神经网络计算机. 这里芯片是把已经经过计算机模拟得到确定权值和固定网络模型的一个具有特

定用途的神经网络固化在大规模集成芯片上.以上两条开发途径的中心思想都是用硬件即空间复杂度为代价换取时间复杂度的降低来提高神经信息处理的速度.

通用神经网络处理机是神经网络研究与产品开发的手段,其发展将是使神经网络计算机研究走向深入和实用的最有效途径.目前通用神经网络处理机所用ASIC芯片存在三种实现方式:

1) 全部以模拟量进行运算.这种结构每个神经元的乘法、求和、非线性处理等运算电路非常简单,易于集成,但权重的存储和修改十分困难,自学习功能与灵活性很差.宜于全并行处理,且适合于实现小规模的神经网络.

2) 全部以数字量进行运算.这种结构的神经网络处理机具有精度高、灵活性强,易于通用微机联结,易于实现串并行处理系统模拟大规模神经网络等优点,但其线路复杂,集成难度很大,成本很高.

3) 模拟量与数字量混合运算.这种结构容易解决大批量权值的快速存储和修改问题,使之可以具有全数字运算方式的灵活性和自学习功能,同时又兼有全模拟量运算方式,乘、加电路简单,易于集成的特点,是一种较理想实现通用神经网络处理机的途径.

目前数模结合的新型通用神经网络计算机已经问世.在国外,美国加利福尼亚大学(伯克莱分校)和匈牙利科学院自动化所联合提出的模逻(Analogic)计算机,以点格神经网络(CNN)为硬件设计基础,构造了一种既可作模拟信息处理,又可作逻辑运算的通用信息处理机.现在,核心芯片GAPU(全局模逻程序单元)和CNN(含神经元、局部的通信控制单元、模拟存储器、逻辑存储器、模逻输出单元、局部逻辑单元)芯片都在加紧开发并走向商品化阶段.在国内,中国科学院半导体所神经网络组研制了一种适合于我国神经网络需要的通用神经网络处理机.它以通用微机作为宿主机,大量采用高速SRAM作为权值和神经元状态存储器,并以模拟量和数字量混合处理方法实现的串并行处理机.

(3) 全硬件实现的神经网络计算机

全硬件实现的神经网络计算机是指计算机体系结构本身就是按照神经网络原理设计的，一个处理器代表一个神经元，用大量的神经元来构造神经网络计算机. 它的优点是速度快，可以满足许多应用的实时要求. 但是全硬件实现中各处理单元之间的连接方式一般难以改变，因此全硬件实现往往在专用神经网络计算机中采用，缺乏通用性、灵活性和可编程性.

目前，不可能设想由成千上万个通用微处理器通过复杂互连形成网络来实现全硬件的神经网络计算机. 这是因为人脑的神经元数目数量在万亿个量级，神经元之间突触互连数则以万计，要完全用硅芯片进行一一对应的模仿实现，在可见的将来仍有相当困难. 以晶体管为例，2000 年左右在硅片上采用深亚微米工艺大约可集成一亿个晶体管，但到时多层布线技术仅可做到六至七层，要实现单元电路间的高度互连是难以达到的. 因此，希望制造出一种计算机，使它的功能接近人脑在语言识别、联想记忆等方面的功能这不是一朝一夕就可以实现的，还需要走一段漫长的路. 神经网络计算机不能取代常规计算机已成为学术界的共识. 许多专家认为，未来的智能计算机(第六代计算机)应由具有较粗粒度的巨型计算机与具有细粒度的神经网络计算机共同组成. 神经网络计算机的一个基本特征是全息方式的存储，它适于低层次的传感处理，而传统的计算机和人工智能技术主要基于算法(包括非确定算法)、数据及知识结构. 如何将这两方面结合起来，充分发挥两者的长处，通过大规模并行处理来实现人类智能的有关计算，这正是所谓结构化连接模型(structured connectionist model)的目标.

3. 光学神经网络计算机

电子技术与光学技术相比，精度高，抗噪声能力强，便于程序控制. 但当采用 VLSI 技术来具体实现各种复杂的逻辑电路和处理芯片时，众多的处理器芯片中的电子通过电磁场发生强烈的相互作用，容易造成信号之间的相互干扰，从而成为采用 VLSI 技术进行复杂的高密度连接的严重制约因素. 这一缺点又恰恰是光学

系统的优点：光线传播可以任意交叉而完全没有相互作用；光学运算拥有真实的高速并行性. 当然光学方法实现也存在其特有困难，如光学记录介质的动态范围小、灵敏度低、光通过介质的损耗、噪声等. 但近年来一系列光学器件的发展，如光调制器、光学双稳阵列、发光器件、幅度探测器等，为光学神经网络计算机的实现提供了必要的技术准备.

与电子神经网络计算机相比，光学神经网络计算机具有以下3个重要特征：

1) 光具有空间并行性，这一点与具有并行机制的神经网络计算机相吻合.

2) 光波的传播无交叉失真，传播容量大.

3) 有可能实现超高速的运算.

上述特征 1)是最重要的. 在神经网络计算机中，当神经元总数为 n，作全互连时，则互连数为 $n(n-1)/2$，即正比于 n^2. 例如由 10^4 个神经元构成的神经网络有可能需要 10^8 根连接线，这样大的数目，用现有的硅 VLSI 技术很难实现. 另一方面，当神经网络计算机进行学习时，需要动态地改变突触结合强度，当神经网络数目很多时，用硅 VLSI 实现有很大困难. 如果利用光技术有希望动态地实现神经元之间的大量连接.

下面简要介绍光神经网络计算机的基本原理，可以通过图6.6所示的光学矩阵向量乘法器来了解光学系统怎样实现神经网络中频繁用到的 $\Sigma W_{ij}V_j$ 计算.

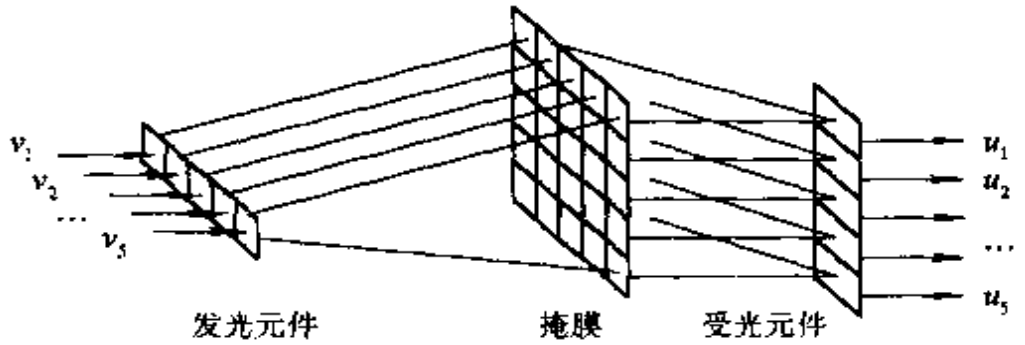


图 6.6　光学矩阵向量乘法器

输入信号 $V=(v_1,v_2,v_3,v_4,v_5)$ 由液晶、发光二极管(或激光二极管)等发光元件转换成光信号,通过透镜成扇面展开,投影在光学掩膜的 j 列上.掩膜各区域透光率不同,对应于权值 W_{ij} 的不同,故其输出光强与 $W_{ij}V_{ij}$ 成正比.掩膜的输出光经过透镜聚焦于光敏二极管等受光元件,即将掩膜 i 行的输出聚焦于列向量 U 的 i 分量处,从而有 $u_i=\Sigma W_{ij}V_{ij}$.

若将 M 个输入模式经过正交变换,形成正交的模式簇,受光器件换成全息胶片,经 M 次曝光后,便在全息胶片上记忆了 M 个模式的信息.由于已经过了正交化,故全息图中每一空间位置上的实际曝光次数将基本上与 M 无关,从而大大扩展了记忆的容量.

当待识模式(可以是加噪声后的记忆模式或是有残缺的记忆模式)输入后,由全息片读出的再现图像经 CCD 器件探测器转换为电信号取阈值后,再由液晶转换成光信号回到全息片的光路中去.经多次循环迭代后,系统将识别出此输入模式并在监视器上给出完整的记忆图像(噪声被去除,残缺被补足).图 6.7 给出了上述过程的示意图.

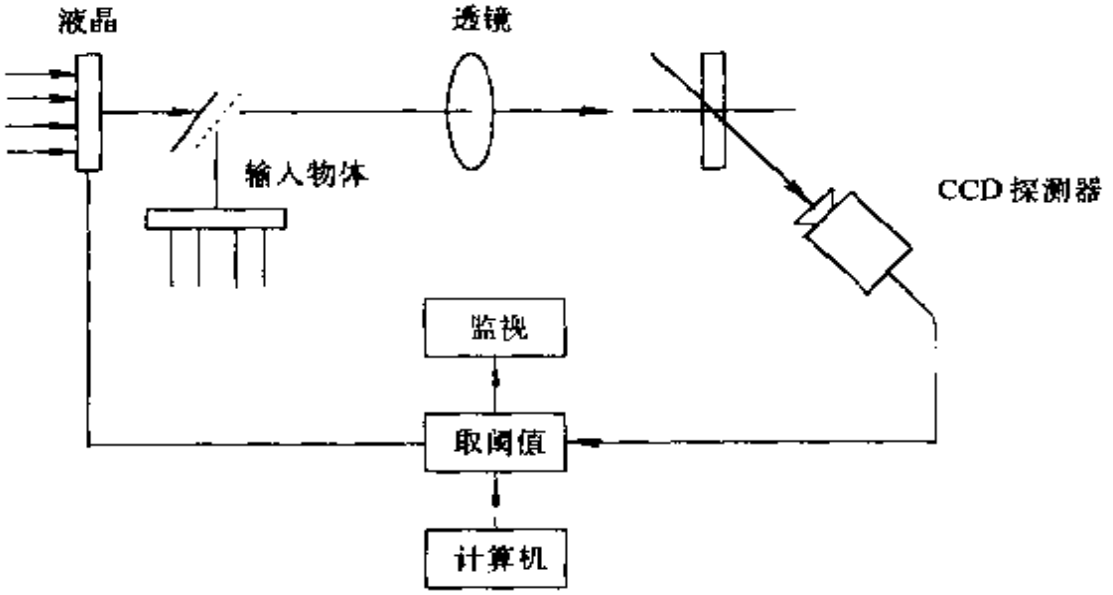


图 6.7 光学联想存储器

尽管光学神经网络计算机的实现还存在许多难题,但由于摆

脱了传统计算机的设计思想体系以及光的许多独特性能与神经网络计算机的特征相近似，所以光学神经网络计算机的研究一直令人注目，并有着广阔的发展前景.它对新一代的智能计算机——神经网络计算机的研究将会有十分重大的意义.

需要指出，光学神经网络虽然具有高速并行处理、高密度互连等优点，但是光学元件零散、笨重、固定和校正都很困难.近年来从理论、结构到技术等方面开展了光电集成的研究.用光电集成技术来实现神经网络兼有电子神经网络的固体化和集成化等优点，又能保持光学神经网络快速、高密度互连等优异性能.因此用光电集成技术实现神经网络已显示出其旺盛的生命力，成为实现神经网络计算机最富有前景的研究方向和途径.

4. 生物分子计算机

生物分子计算机是一种模拟人脑神经网络的新型计算机.它的实现途径可能有两条，一是从分子线路的设计与制作出发，形成一个体积小、速度快、存储容量大而成本低的数字分子计算机；另一是模拟生物图像信息处理的快速模式识别、自组织和自学习能力，构造感触式的分子计算机.

生物微电子学的先驱科学家从1983年就开始进行理论和实验的研究工作，1984年在美国加利福尼亚举行了第一届生物芯片跨学科会议，此后，分子计算机往往成为许多会议的主题.我国东南大学已在实验中培养出具有受激、传输性能的分子器件.目前，要研制一台实用的分子计算机，虽然道路还漫长，需要不断探索，但是可以预料，人工智能及人工神经网络技术与自然智能之间的距离会随着生物分子计算机研究的深入发展而逐渐缩短.

参 考 文 献

[1] 吴今培编著，模糊诊断理论及其应用，科学出版社，1995.

[2] 沈清、胡德文、时春编著，神经网络应用技术，国防科技大学出版社，1993.

[3] 胡守仁、余少波、戴葵编著，神经网络导论，国防科技大学出版社，1993.

[4] 王永庆，人工智能原理·方法·应用，西安交通大学出版社，1994.

[5] 杨叔子、史铁林等，基于知识的诊断推理，清华大学出版社，1993.

[6] 石博强、李畅著，大功率汽车发动机故障智能诊断，冶金工业出版社，1995.

[7] 吴今培著，实用时序分析，湖南科技出版社，1989.

[8] 刘有才、刘增良编著，模糊专家系统原理与设计，北京航空航天大学出版社，1994.

[9] 万建伟等，一种新的采用神经网络的图象分类方法研究，系统工程与电子技术，1995 年第 2 期.

[10] 许飞云，基于行为的智能化故障诊断理论和方法的研究，东南大学博士学位论文，1996.

[11] 虞厥邦，硅神经网络计算机的研究与发展近况，神经网络理论与应用研究 '96，西南交通大学出版社.

[12] 吴今培、肖健华，故障诊断专家系统的一种新设计方案，振动、测试与诊断，1996 年第 4 期.

[13] 吴今培、刘智勇，模糊时间序列模型的辨识，模糊系统与数学，1995 年第 4 期.

[14] Wu J. P.（吴今培），A Fuzzy Diagnosis Method for the Diesel Engine，Proceedings of the International Conference on Structural Dynamics，Vibration，Noise and Control，Hong Kong Polytechnic University，pp. 1316—1321，Dec. 1995.

[15] Fu L. M.，Rule Generation from Neural Networks，IEEE Trans. on SMC，Vol. 24，No. 8，pp. 1114—1124，1994.

[16] Doherty N. F.，Kochher A. K. and Main R.，Knowledge-based Approaches to Fault Diagnosis：a Practical Evaluation of the Relative Merits of Deep and Shallow Knowledge，Proc Instn Mech Engrs，Vol. 208，Part B：Journal of Engineering Manufacture，pp. 39—45，1994.

[17] Frederick W. R. and Michael L. D.，Human Cognition and the Expert System Interface：Mental Models and Inference Explanations，IEEE Trans. on SMC，Vol. 23，No. 6，pp. 1649—1661，1993.

[18] Si-Zhao Q.，Hong-Te S. and Thomas J. M.，Comparison of Four Neural Net Learning Methods for Dynamics System Identification，IEEE Trans. on Nns，Vol. 3，No. 1，pp. 122—130，1992.

[19] Buckley J. J. and Hayashi Y. Fuzzy Neural Networks：A Survey，Fuzzy Sets and Systems Vol. 66，No. 1，pp. 1—13，1994.

[20] Sankar K. P. and Sushmita M.，Multilayer Perceptron，Fuzzy Sets and Classification，IEEE Trans. on Nns，Vol. 3，No. 5，pp. 683—697，1992.

[21] Huang J. X. et al.，Fuzzy Art Properties，Neural Networks，Vol. 8，No. 2，pp. 203—213，1995.

[22] Sushmita M. and Sankar K. P., Fuzzy Multi-Layer Perception, Inferencing and Rule Generation, IEEE Trans. on Nns, Vol. 6, No. 1, pp. 51—63, 1995.
[23] Tang Z. Y. and Gary J. K., Deterministic Global Optimal FNN Training Algorithms, Neural Networks, Vol. 7, No. 2, pp. 301—311, 1994.
[24] Sushmita M. and Sankar K. P., Logical Operation Based Fuzzy MLP for Classification and Rule Generation, Neural Networks, Vol. 7, No. 2, pp. 353—373, 1994.
[25] Vincent W. P. et al, Alternative Neural Network Training Methods, IEEE Expert, pp. 16—22, 1995.
[26] Chung F. L. and Lee T., Fuzzy Competitive Learning, Neural Networks, Vol. 7, No. 3, pp. 539—551, 1994.
[27] Benedikt K. H., Improving Back Propagation with a New Error Function, Neural Networks, Vol. 7, No. 8, pp. 1191—1192, 1994.
[28] Ronald R. Y., Modeling and Formulating Fuzzy Knowledge Bases Using Neural Networks, Neural Networks, Vol. 7, No. 8, pp. 1273—1283, 1994.
[29] Neil C. R., Artificial Intelligence Through Prolog, Prentice Hall, 1988.
[30] Hayes R. F. et al., Building Expert System, Addison-wesley Publishing Company, Inc. 1983.
[31] Kosko B., Fuzzy Associative Memories, In A. Kandel (ed). Fuzzy Expert System Reading, MA: Addison Wesley, 1987.
[32] Kosko B., Neural Networks and Fuzzy Systems, Prentice Hall, Inc. 1992.
[33] Venkatasubramanian V., Recall and Generalization Performances of Neural Networks for Process Fault Diagnosis, In Proc. Fourth Int. Conf. on Chemical Processes, South Padre Island, TX, 1991.
[34] Venkatasubramanian V. et al., Process Fault Detection and Diagnosis Using Neural Networks: I. Steady State Processes, Computers Chemical Engng, Vol. 14, pp. 699—722, 1990.
[35] Gallant S. I., Connectionist Expert Systems, Communications of the ACM, Vol. 31, No. 2, pp. 152—169, 1988.
[36] Funahashi K., On the Approximate Realization of Continuous Mappings by Neural Networks, Neural Networks, Vol. 2, No. 1, pp. 183—192, 1989.
[37] Girosi F., Regularization Theory and Neural Networks Architectures, Neural Computation, Vol. 7, pp. 219—269, 1995.
[38] Connor J. T. et al., Recurrent Neural Networks and Robust Time Series Prediction, IEEE Trans. on Nns, Vol. 5, No. 2, pp. 240—254, 1994.
[39] Atiya A. and Parlos A. G., Identification of Nonlinear Dynamics Using a General Spatio-temporal Networks, Mathl. Comput. Modeling, Vol. 21, No. 1, pp. 53—71, 1995.
[40] Brooks R. A., Intelligence without Representation, Artificial Intelligence, Vol. 47, pp. 139—159, 1991.
[41] Brooks R. A., Intelligence without Reason, Proc. IJCAI-91, Sydney, Australia, 1991.
[42] Peng Y. and Reggia J. A., Diagnostic Problem Solving with Causal Chaining, Int. J. Intelligence System, Vol. 2, pp. 265—302, 1987.

[43] Reggia J. A. et al. ,A Formal Model of Diagnosis Inference——Part I:Problem Formulation and Decomposition,Information Sci. ,Vol. 37,pp. 227—256,1985.
[44] Shortliffe E. ,Computer-based Medical Consultations:MYCIN. American Elsevier Publishing Inc. ,1976.
[45] Shafer G. ,A Mathematical Theory of Evidence,Princeton,NJ. Princeton Univ. Press. ,1976.
[46] Peng Y. and James A. ,A Probabilistic Causal Model for Diagnosis Problem Solving——Part I: Integrating Symbol Causal Inference with Numeric Probabilistic Inference,IEEE Trans. on SMC, Vol. 17,No. 2,pp. 146—162,1987.
[47] Botanna F. and Bahamonde A. ,Shape:A Machine Learning System form Examples,Int. J. Human Computer Studies,Vol. 42,pp. 137—155,1995.
[48] Lansdale M. ,New Method for Discovering Rules from Examples,Man-Machine Studies,Vol. 36, pp. 127—143,1992.
[49] Davis R. and Lenat D. ,Knowledge-based System in Artificial Intelligence,McGraw-Hill,New York,1980.
[50] Winston P. H. ,Artificial Intelligence,Second Edition,Addison-Wesley,1984.
[51] Fink P. K. and Lusth J. C. ,Expert System and Diagnostic Expertise in the Mechanical and Electrical Domains,IEEE Trans. on SMC,Vol. 17,No. 3,pp. 340—349,1987.
[52] 杨建刚等,改进 BP 网络在旋转机械诊断中的应用,振动工程学报,1995 年第 4 期.
[53] 张良杰、李衍达,模糊神经网络技术的新近发展,信息与控制,1995 年第 1 期.
[54] 谭民、疏松桂,神经元网络在故障诊断中的双向联想记忆法,自动化学报,1991 年第 1 期.
[55] 梅胜敏等,故障诊断专家系统中的模糊推理方法,南京航空航天大学学报,1995 年第 4 期.
[56] 黄洪钟等,神经网络技术在机械工程中的应用与展望,机械科学与技术,1995 年第 4 期.
[57] 许飞云等,旋转机械工况辨识神经网络方法,东南大学学报,1995 年第 5 期.
[58] Janson[德国]著,李聪译,TURBO PROLOG 与专家系统,电子工业出版社,1994.
[59] 吴泉原、刘江宁编著,人工智能与专家系统,国防科技大学出版社,1995.
[60] 施鸿宝、王秋荷编,专家系统,西安交通大学出版社,1990.
[61] 何永勇,基于神经网络提高故障诊断精度策略的研究——旋转机械故障诊断理论、方法及系统的研究,东南大学博士学位论文,1997.
[62] 陈兆乾、潘金贵、谢俊元编译,Turbo Prolog 程序设计,南京大学出版社,1991.
[63] 李晓忠、汪培庄、罗承忠编著,模糊神经网络,贵州科技出版社,1994.
[64] 罗四维、张彤,一个基于模糊理论的神经网,中国神经网络首届学术大会论文,1988.
[65] 李强等,焊接质量的模糊综合评判,北方交通大学学报,1996 年第 4 期.
[66] 张杰等,专家系统的发展及其在化工生产中的应用,化工自动化及仪表,1996 年第 3 期.
[67] 陈国金、成自维,内燃机气阀机构故障的模糊诊断,振动工程学报,1396 年第 2 期.

[68] 王鹰舵等，基于专家系统和神经网络的机车电路故障诊断系统研究，北方交通大学学报，1996 年第 4 期.
[69] 沈建强、李平，神经模糊技术的研究现状与展望，控制与决策，1996 年第 5 期.
[70] 张雪江等，汽轮发电机组故障诊断专家系统知识处理技术的研究，振动工程学报，1996 年第 3 期.
[71] 李洪兴等编著，工程模糊数学方法及应用，天津科学技术出版社，1993.
[72] 范俊波、靳蕃、史燕，模糊联想记忆的一种有效学习算法，电子学报，1996 年第 1 期.
[73] 张雪江，机械设备故障诊断系统知识自动获取及更新的研究，东南大学博士学位论文，1997.
[74] 刘嵘，周鸣歧，故障诊断专家系统综述，测控技术，1994 年第 2 期.
[75] 费宗铭、吕建等，机器学习，计算机科学，1991 年第 1 期.
[76] 洪家荣，机器学习——回顾与展望，计算机科学，1991 年第 1 期.
[77] 阎平凡，人工神经网络的容量、学习与计算复杂性，电子学报，1995 年第 5 期.
[78] 王继成、吕维雪，一个基于规则的神经网络系统，电子学报，1995 年第 2 期.
[79] 张铃、张钹，神经网络中的 BP 算法分析，模式识别与人工智能，1994 年第 3 期.
[80] 史铁林、王雪等，层次分类诊断模型，华中理工大学学报，1993 年第 1 期.
[81] 吴今培、应立军，模糊时序分析的理论与方法，系统工程，1992 年第 6 期.
[82] 黄可鸣，专家系统导论，东南大学出版社，1988.
[83] 吴今培、刘智勇，模糊时序模型辨识的约束最小二乘法，系统工程理论与实践，1997 年第 2 期.
[84] Ruckley J. J. and Hayashi Y., Neural Nets for Fuzzy Systems, Fuzzy Sets and Systems, Vol. 71, pp. 265—276, 1995.
[85] Castro J. L., Fuzzy Logic Controllers' are Universal Approximation, IEEE Trans. on SMC, Vol. 25, No. 4, pp. 629—635, 1995.
[86] Chao C. T. and Teng C. C., Implementation of a Fuzzy Inference System Using a Normalized Fuzzy Neural Networks, Fuzzy Sets and Systems, Vol. 75, pp. 17—31, 1995
[87] Archer K. P. and Wang S., Fuzzy Set Representation of Neural Network Classification, IEEE Trans. on SMC, Vol. 21, No. 4, pp. 735—742, 1991.
[88] Timo S. and Heikki N. K. and Hannu K., Neural Networks in Process Fault Diagnosis, IEEE Trans. on SMC, Vol. 21, No. 4, pp. 315—825, 1991.
[89] Vohhan P., Adaptive Pattern Recognition and Neural Network, Addsion Wesley Inc., 1989.
[90] Zimmermann H. J., Fuzzy Set Theory and Its Applications, Boston, Kluwer Academic Publishers, 1985.
[91] Harmon P. and Sawyer B., Creating Expert Systems for Business and Industry, John Wiley & Sons, Inc., 1990.
[92] Negojta C. V., Expert Systems and Fuzzy Systems, London, 1985.
[93] Hart A., Knowledge Acquisition for Expert Systems, McGraw-Hill Book Company, 1986.
[94] Sanchez E. and Zadeh L. A., Approximate Reasoning in Intelligent Systems, Decision and Control, Pergamon, 1988.
[95] Tanaka H. and Ishibuchi H., Identification of Possibilistic Linear Systems by

Quadratic Membership Function of Fuzzy Parameters, Fuzzy Sets and Systems, Vol. 41, pp. 145—150, 1991.

[96] Masumi I., Learning of Modular Structured Networks, Artificial Intelligence, Vol. 75, pp. 51—62, 1995.

[97] Nelson M. and Herve A. B., Neural Networks for Statistical Recognition of Continuous Speech, Proc. Of the IEEE, Vol. 83, No. 5, pp. 742—770, 1995.

[98] Rangachari A. et al., Efficient Classification for Multiclass Problems Using Modular Neural Networks, IEEE Trans. on Nns, Vol. 6, No. 1, pp. 117—124, 1995.

[99] Ripley B. D., Neural Networks and Related Methods for Classification, J. R. Stutist. Soc. B, Vol. 56, No. 3, pp. 409—456, 1994.

[100] Rivas C. et al., Dynamic Modeling Using a Multiple Neural Networks Architecture, IEEE Conf. Pub. Vol. 2, No. 3, pp. 978—982, 1994.

[101] Smith D. J., A Artificial Intelligence—— Today's New Design and Diagnostic Tool, Power Engineering, Vol. 93, No. 1, pp. 26—30, 1989.

[102] Traven H. G., A Neural Networks Approach to Statistical Pattern Classification by Semiparametric Estimation of Probability Density Function, IEEE Trans. on Nns, Vol. 2, pp. 366—375, 1991.

[103] Tzafestas S. G. and Dalianis P. J., Fault Diagnosis in Complex Systems Using Artificial Neural Networks, Proc. of the IEEE Conf. on Control Applications, Vol. 2, pp. 877—882, 1994.

[104] Ungar L. H. et al., Adaptive Networks for Fault Diagnosis and Process Control, Computers Chem. Engng, Vol. 14, pp. 561—573, 1990.

[105] Van O. A. and Nienhuis B., Improving the Convergence of the Back-propagation Algorithm, Nns, Vol. 5, pp. 465—471, 1994.

Quadratic Membership Function of Fuzzy Parameters, Fuzzy Sets and Systems, Vol. 41, pp. 145—150, 1991.

[96] Masumi I., Learning of Modular Structured Networks, Artificial Intelligence, Vol. 75, pp. 51—62, 1995.

[97] Nelson M. and Herve A. B., Neural Networks for Statistical Recognition of Continuous Speech, Proc. Of the IEEE, Vol. 83, No. 5, pp. 742—770, 1995.

[98] Rangachari A. et al., Efficient Classification for Multiclass Problems Using Modular Neural Networks, IEEE Trans. on Nns, Vol. 6, No. 1, pp. 117—124, 1995.

[99] Ripley B. D., Neural Networks and Related Methods for Classification, J. R. Stutist. Soc. B, Vol. 56, No. 3, pp. 409—456, 1994.

[100] Rivas C. et al., Dynamic Modeling Using a Multiple Neural Networks Architecture, IEEE Conf. Pub. Vol. 2, No. 3, pp. 978—982, 1994.

[101] Smith D. J., A Artificial Intelligence—— Today's New Design and Diagnostic Tool, Power Engineering, Vol. 93, No. 1, pp. 26—30, 1989.

[102] Traven H. G., A Neural Networks Approach to Statistical Pattern Classification by Semiparametric Estimation of Probability Density Function, IEEE Trans. on Nns, Vol. 2, pp. 366—375, 1991.

[103] Tzafestas S. G. and Dalianis P. J., Fault Diagnosis in Complex Systems Using Artificial Neural Networks, Proc. of the IEEE Conf. on Control Applications, Vol. 2, pp. 877—882, 1994.

[104] Ungar L. H. et al., Adaptive Networks for Fault Diagnosis and Process Control, Computers Chem. Engng, Vol. 14, pp. 561—573, 1990.

[105] Van O. A. and Nienhuis B., Improving the Convergence of the Back-propagation Algorithm, Nns, Vol. 5, pp. 465—471, 1994.